1508, TFDEN-010-008, 001

NACHi

FD/CFD CONTROLLER INSTRUCTION MANUAL SOFTWARE PLC

8th edition

- Before attempting to operate the robot, please read through this operating manual carefully, and comply with all the safety-related items and instructions in the text.
- The installation, operation and maintenance of this robot should be undertaken only by those individuals who have attended one of our robot course.
- When using this robot, observe the low related with industrial robot and with safety issues in each country.
- This operating manual must be given without fail to the individual who will be actually operating the robot.
- Please direct any queries about parts of this operating manual which may not be completely clear or any inquiries concerning the after-sale service of this robot to any of the service centers listed on the back cover.

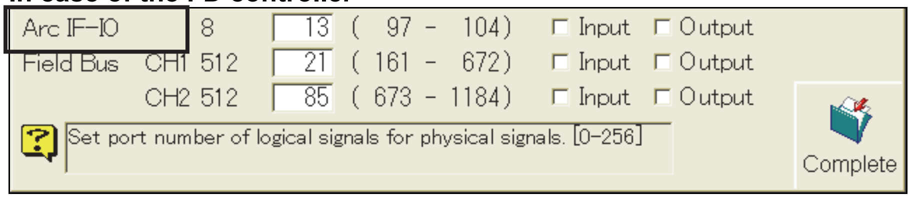
NACHI-FUJIKOSHI CORP.

For customers using CFD controller

(1) The name of the "Arc IF-IO" will change to "Mini I/O". (The address is the same)

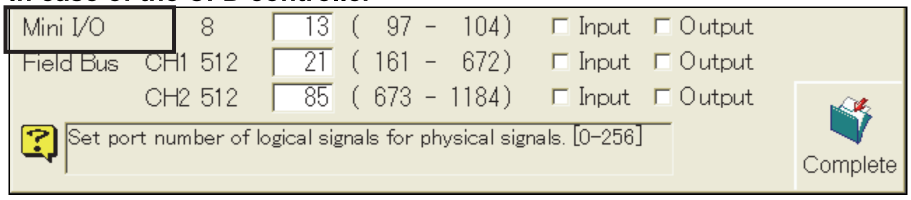
The screen examples in the <Constant Setting> - [6 Signals] [15 Hardware setting] are shown as below.

In case of the FD controller



```
Arc IF-IO        8    13  (  97 -  104)  □ Input  □ Output
Field Bus  CH1 512    21  ( 161 -  672)  □ Input  □ Output
           CH2 512    85  ( 673 - 1184)  □ Input  □ Output
Set port number of logical signals for physical signals. [0-256]
```

In case of the CFD controller



```
Mini I/O         8    13  (  97 -  104)  □ Input  □ Output
Field Bus  CH1 512    21  ( 161 -  672)  □ Input  □ Output
           CH2 512    85  ( 673 - 1184)  □ Input  □ Output
Set port number of logical signals for physical signals. [0-256]
```

(2) Because of the hardware structure, the "I/O-3" (Digital I/O board 3) is not available in the CFD controller.

(3) The outline of the I/O signals (factory setting) is shown as below.

Input signals

Logical input	Physical input	Remarks
I1～I32	X0000～X0031	Digital I/O board 1 (CNIN)
I33～I63	X0064～X0095	Digital I/O board 2 (CNIN)
I64～I96	X0096～X0127	Not used
I97～I104	X0128～X0135	Mini I/O board
I105～I160		No connection to the physical input signals
I161～I672	X1000～X1511	Fieldbus input CH1(512 points)
I673～I1184	X1512～X2023	Fieldbus input CH2(512 points)
I1185～I1696	X2024～X2535	Fieldbus input CH3(512 points)
I1697～I2048	X2536～X3047	Fieldbus input CH4(352 points)

(NOTE) The following address is used for the ***"System input signals (Fixed input signals)"***
X0032～X0063

Output signals

Logical output	Physical output	Remarks
O1～O32	Y0000～Y0031	Digital I/O board 1 (CNOUT)
O33～O63	Y0064～Y0095	Digital I/O board 2 (CNOUT)
O64～O96	Y0096～Y0127	Not used
O97～O104	Y0128～Y0135	Mini I/O board
O105～O160		No connection to the physical input signals
O161～O672	Y1000～Y1511	Fieldbus output CH1(512 points)
O673～O1184	Y1512～Y2023	Fieldbus output CH2(512 points)
O1185～O1696	Y2024～Y2535	Fieldbus output CH3(512 points)
O1697～O2048	Y2536～Y3047	Fieldbus output CH4(352 points)

(NOTE) The following address is used for the ***"System output signals (Fixed output signals)"***
Y0032～Y0063

Table of Contents

Chapter 5 Monitors

Chapter 6 Input/output relay lists

Chapter 7 List of command words

Chapter 1 General description

This chapter provides a general description of the software PLC.

1.1 General description of the software PLC

1.1.1 General description of the software PLC

The PLC (Programmable Logic Controller) is a device that controls various devices by incorporating input signals and previously created programs to switch contacts within output circuits ON/OFF.
The software PLC is software incorporated into the robot controller that has all of the functions of the PLC, and can be programmed using the teach pendant. In this way, the need to provide a special external PLC is obviated, thereby helping to reduce costs.

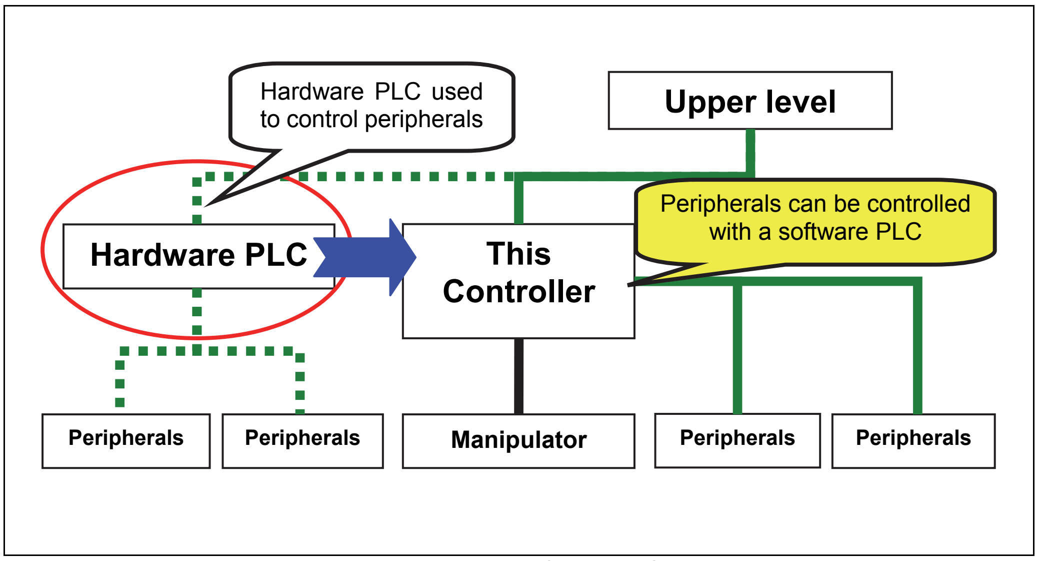


Fig. 1.1.1 Utilizing the software PLC

As in Fig. 1.1.2, the software PLC occupies a position between the inside and outside of the robot controller. The physical signals to and from the devices outside the controller connected by the parallel I/O, field buses, etc. are connected to the logical signals through the PLC programs.

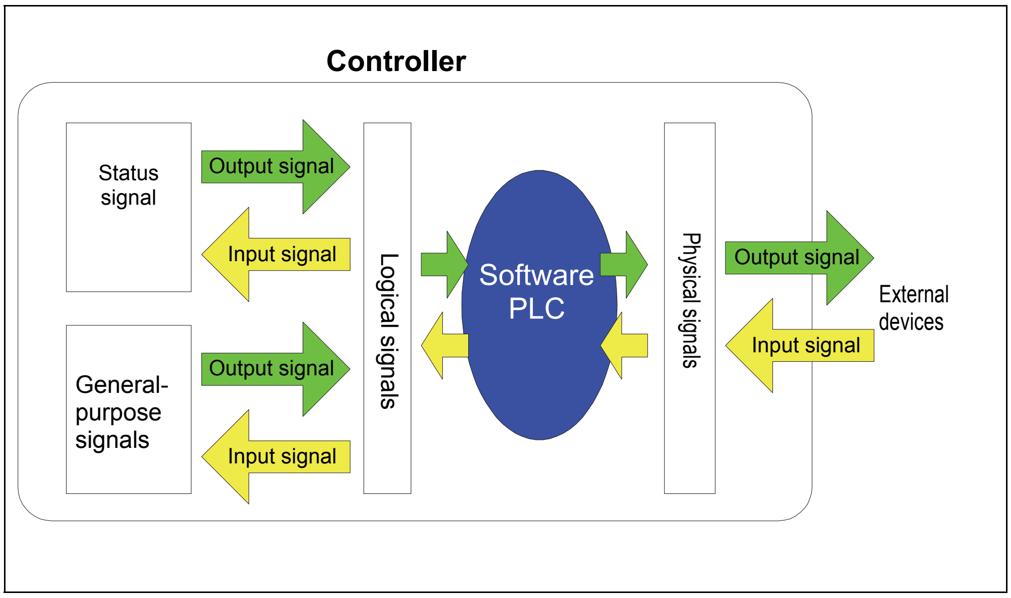
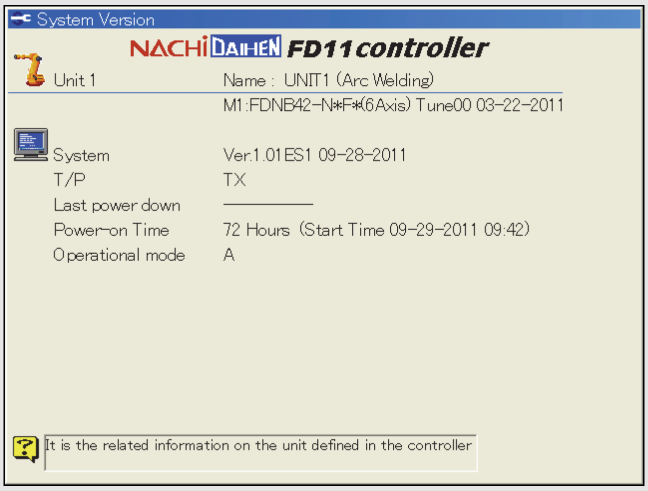


Fig. 1.1.2 Flow of input/output signals through software PLC

POINT

Software PLC Factory Settings

The factory settings for the software PLC differ depending on the operation mode.
The current operating mode can be checked on the system environment screen displayed with shortcut code "286". For details, see “Basic Operations Chapter 1 Introduction” in the Instruction Manual.


System Version

NACHi DAIHEN FD11controller

Unit 1	Name : UNIT1 (Arc Welding)
	M1:FDNB42-N*F*(6Axis) Tune00 03-22-2011
System	Ver.1.01ES1 09-28-2011
T/P	TX
Last power down	————
Power-on Time	72 Hours (Start Time 09-29-2011 09:42)
Operational mode	A

It is the related information on the unit defined in the controller

Irrespective of the operation mode, both cases are directly connected with logical signals and physical signals.

Operation mode A:	Shipped in the <startup> state using the software PLC. Logical and physical signals are connected directly by the PLC program "Default.stf" installed in the factory.
Operation mode S:	Shipped in the disconnected state, not using the software PLC. In the disconnected state, logical and physical signals are directly connected.

PLC programs are scanned normally irrespective of the work program. Use of the software PLC can be switched on and off in <Constant settings> - [1 Control Environment] – [4 Built-in PLC].

1.1.2 Software PLC capabilities

The software PLC of this controller uses the ISaGRAF-PRO run time (see <*Note*>). This PLC supports the LD (ladder) language and the five programming languages specified in the IEC61131-3 international standard. The capabilities of the software PLC are listed below.

<Note>
ISaGRAF and ISaGRAF-PRO are registered trademarks of ICS Triplex ISaGRAF Inc.

Table 1.1.1 Software PLC capabilities

Control system	Cyclic scanning system	
Programming languages	Full support for five languages specified in IEC1131-3 • LD (ladder) • FBD (Function Block Diagram) • SFC (Sequential Flow Chart) • ST (Structured Text) • IL (Instruction List) <Note> The ISaGRAF Workbench is an additional requirement if a language other than LD (ladder) is to be used. LD (ladder) is the only language that can be displayed and edited using the teach pendant.	
Types of commands	Basic commands: 12 Function commands: 92	
Program capacity	32 kwords (=3.2 kwords per file x 10 files)	
Processing time	5 to 30 ms	
Input/output relays (Input/output variables)	Logical input	2048 signals
	Logical output	2048 signals
	Physical input	2176 signals
	Physical output	2176 signals
Internal relays (Internal variables)	BOOL variables	2000 signals (Including 500 retained variables)
	Real variables	200 signals (All retained variables)
	DINT variables	500 signals (All retained variables)
	SINT variables	100 signals (All retained variables)
	Timer variables	500 signals (All retained variables)
	Character variables	10 signals (All retained variables)
Operation/Editing	Operation	Specification of disconnect, stop, start supported
	Monitors	Channel monitor, ladder monitor, rung monitor, etc. supported
	Forced Output	Possible
Programming tool	Teach pendant ISaGRAF Workbench (Product that supports ISaGRAF-PRO Ver 4.2 is required)	

For details on the functions of the input/output relays and internal relays, refer to “1.2 Input/output relay”.

<Note> Concerning the program capacity
In the above table, the value of one "word" is equivalent to one-half of the simple circuit of (LOAD+OUT) which counts as 2 words. The maximum capacity per display image file (STF file) for editing is 3.2K (equivalent to 16,000 circuits), and since up to 10 files can be joined together, this in turn yields a capacity of 3.2K x 10 = 32K (equivalent to 16,000 circuits).
The size of 64Kbytes/file for intermediate files (TIC files) that have been compiled is the upper limit in practice.
Refer to Fig. 1.1.4 in order to explain this in simple terms, the foregoing is presented in the form of display images. The maximum number of circuits varies depending on the content of the circuits which are created.

<Note> Processing time
Although the processing time can be set in the range presented in the above table, the factory setting is 30ms, and this is the setting that is normally used. When the actual scan time has exceeded this time setting, "scan time exceeded" is detected.

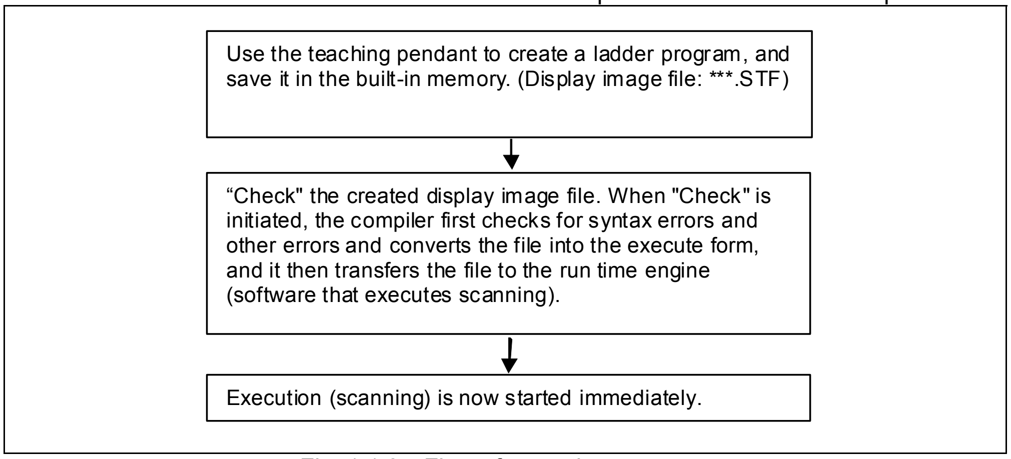
(Reference)
A brief description on the programming languages other than LD (ladder) will now be given.

Table 1.1.2 Five programming languages specified by IEC61131-3

Programming languages	General description
LD ***(Ladder Diagram)***	(Ladder Diagram) This is used for description in the step-by-step method which has been widely used in the past. It is ideal for expressing PLC sequence programs.
FBD ***(Function Block Diagram)***	(Function block Diagram) It is a new structured programming technique. A format that resembles the circuit diagrams of printed circuit boards is featured. A function block (a single function) can be described in any way, and it can be re-used over and over again.
SFC ***(Sequential Function Chart)***	(Sequential Function Chart) It is a new structured programming technique. One by one, the function blocks are prepared using ladders, etc., and multiple blocks can be linked together and written from top to bottom as in a flowchart for easy visual presentation.
ST ***(Structured Text)***	(Structured Text) The same format as ordinary software is featured. This language is ideal for programming numeric algorithms.
IL ***(Instruction List)***	(Instruction List) This languages uses LD, AND, OR and other mnemonic formats.

1.1.3 The structure of the software PLC

The structure of the software PLC is now described at each step of the flow of the actual operation.



Use the teaching pendant to create a ladder program, and save it in the built-in memory. (Display image file: ***.STF)

“Check" the created display image file. When "Check" is initiated, the compiler first checks for syntax errors and other errors and converts the file into the execute form, and it then transfers the file to the run time engine (software that executes scanning).

Execution (scanning) is now started immediately.

Fig. 1.1.3 Flow of operation

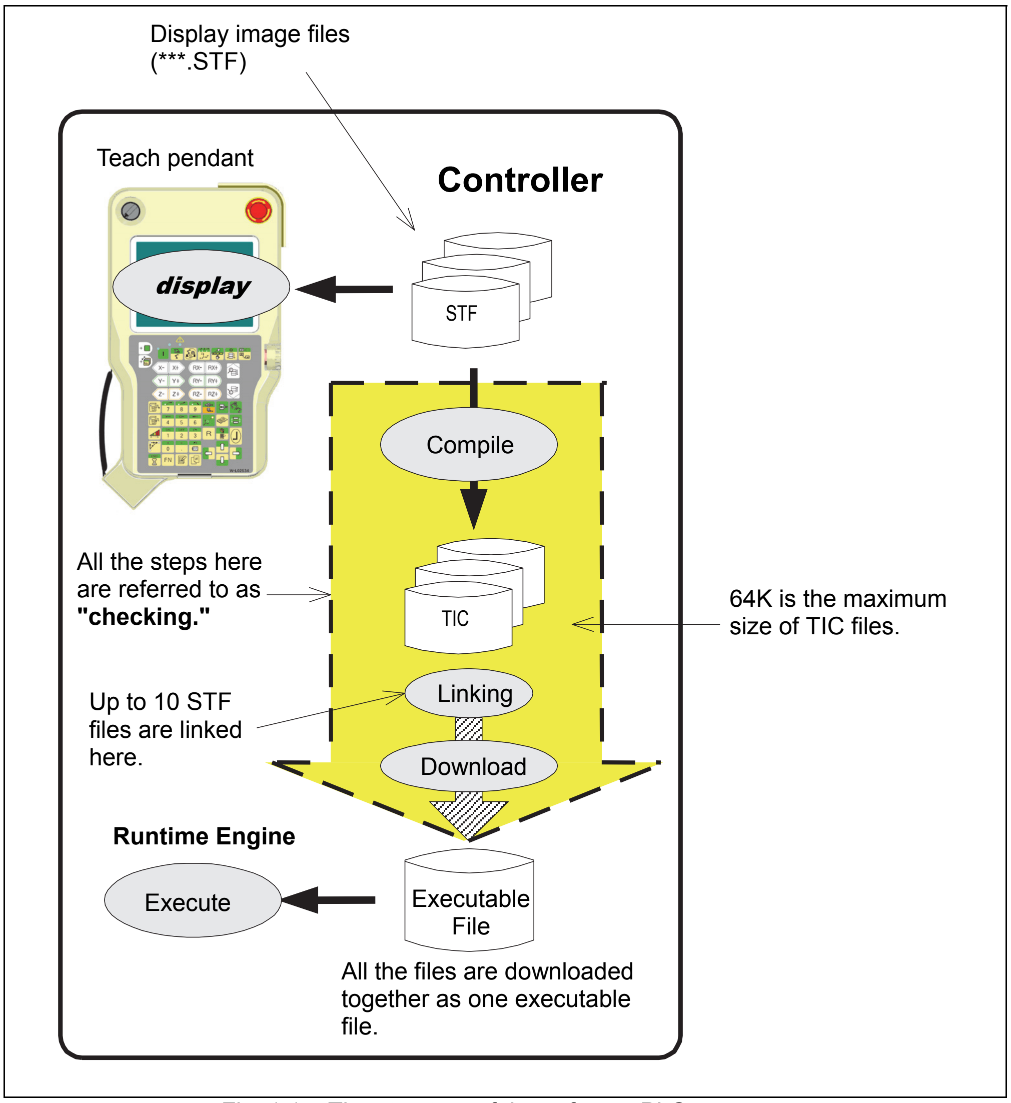

Display image files (***.STF)
Teach pendant
Controller
display
STF
Compile
TIC
Linking
Download
All the steps here are referred to as **"checking."**
64K is the maximum size of TIC files.
Up to 10 STF files are linked here.
Runtime Engine
Execute
Executable File
All the files are downloaded together as one executable file.

Fig. 1.1.4 The structure of the software PLC

IMPORTANT

The file name for the "`****.stf`" has a limitation like the following. The limitation depends on the software version.

V3.21 or before: 16 letters (Even if the filename is more than 16 letters, it can be executed. But monitor function cannot work for the file.)

V3.22 or after: 124 letters

Table 1.1.3 Flow of software PLC operation

1. Use the teaching pendant to create a ladder program, and save it in the built-in memory. (Display image file: ∗∗∗.stf)	
(1)	A multiple number of ladder programs (display image files) can be stored in the internal memory. Programs for each application including spot welding or arc welding can be always kept in the memory so that the one required can be selected and run.
(2)	A long ladder program is divided up into a multiple number of blocks and stored. By so doing, programs can be easily edited and maintained. A program can be divided up into a maximum of 10 blocks. These blocks are linked up later to create executable files. (For the limits per file, refer to Table 1.1.1.)
(3)	A ladder program can be given any name.
2. "Check" the created display image file. When "Check" is initiated, the compiler first checks for syntax errors and other errors and converts the file into the execute form, and it then transfers the file to the run time engine (software that executes scanning).	
(1)	Select the required ladder program from among the ladder programs (display image files), and perform "Check." → Fig. 1.1.5 If a multiple number of ladder programs (display image files) are to be used, the sequence in which they are to be executed must be specified at this time.
(2)	The files are first checked for syntax errors and other errors by the compiler and then converted into TIC files (temporary executable image files). If errors are found, TIC files are not generated.
(3)	When all the ladder programs (display image files) are successfully compiled, the TIC files (temporary executable image files) are linked together to form a single executable file.
3. Execution (scanning) is now started immediately.	
(1)	Scanning starts immediately for the executable file which has been transferred to the run time engine (software that executes scanning).
(2)	The progress made in the ladder scanning and the status of each channel can be displayed on the service monitor for monitoring.

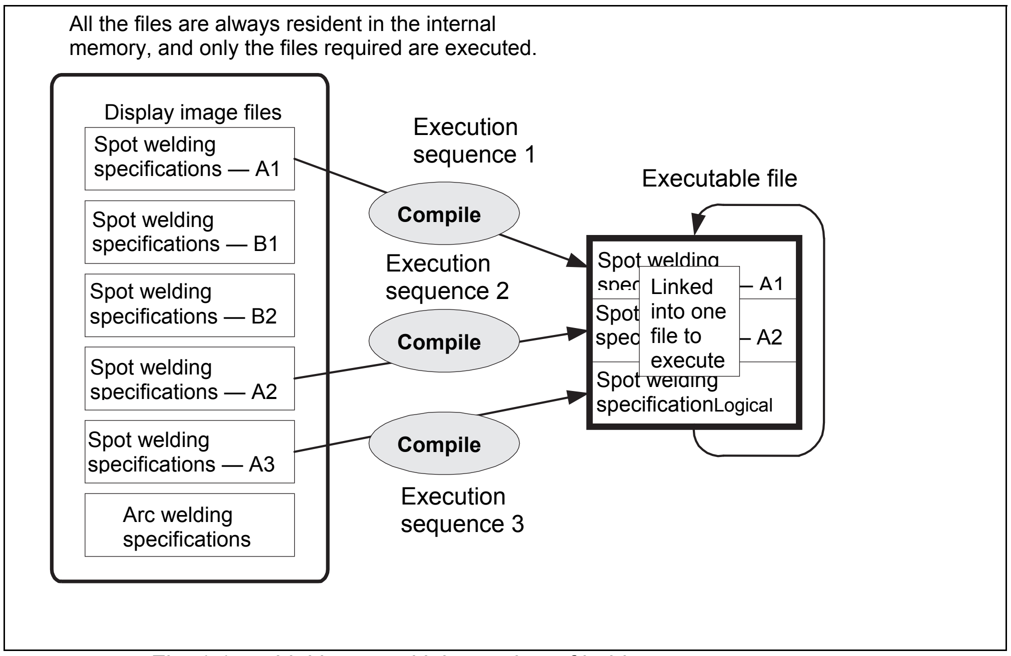

All the files are always resident in the internal memory, and only the files required are executed.
Display image files
Spot welding specifications — A1
Spot welding specifications — B1
Spot welding specifications — B2
Spot welding specifications — A2
Spot welding specifications — A3
Arc welding specifications
Execution sequence 1
Compile
Execution sequence 2
Compile
Compile
Execution sequence 3
Executable file
Spot welding spec — A1
Spot spec — A2
Spot welding specificationLogical
Linked into one file to execute

Fig. 1.1.5 Linking a multiple number of ladder programs

1.2 Input/output relays

The input/output relays and internal relays are described here. Relays are called “variables” as far as the software PLC is concerned. The ON/OFF coil contacts and those with integer data are all treated equally.

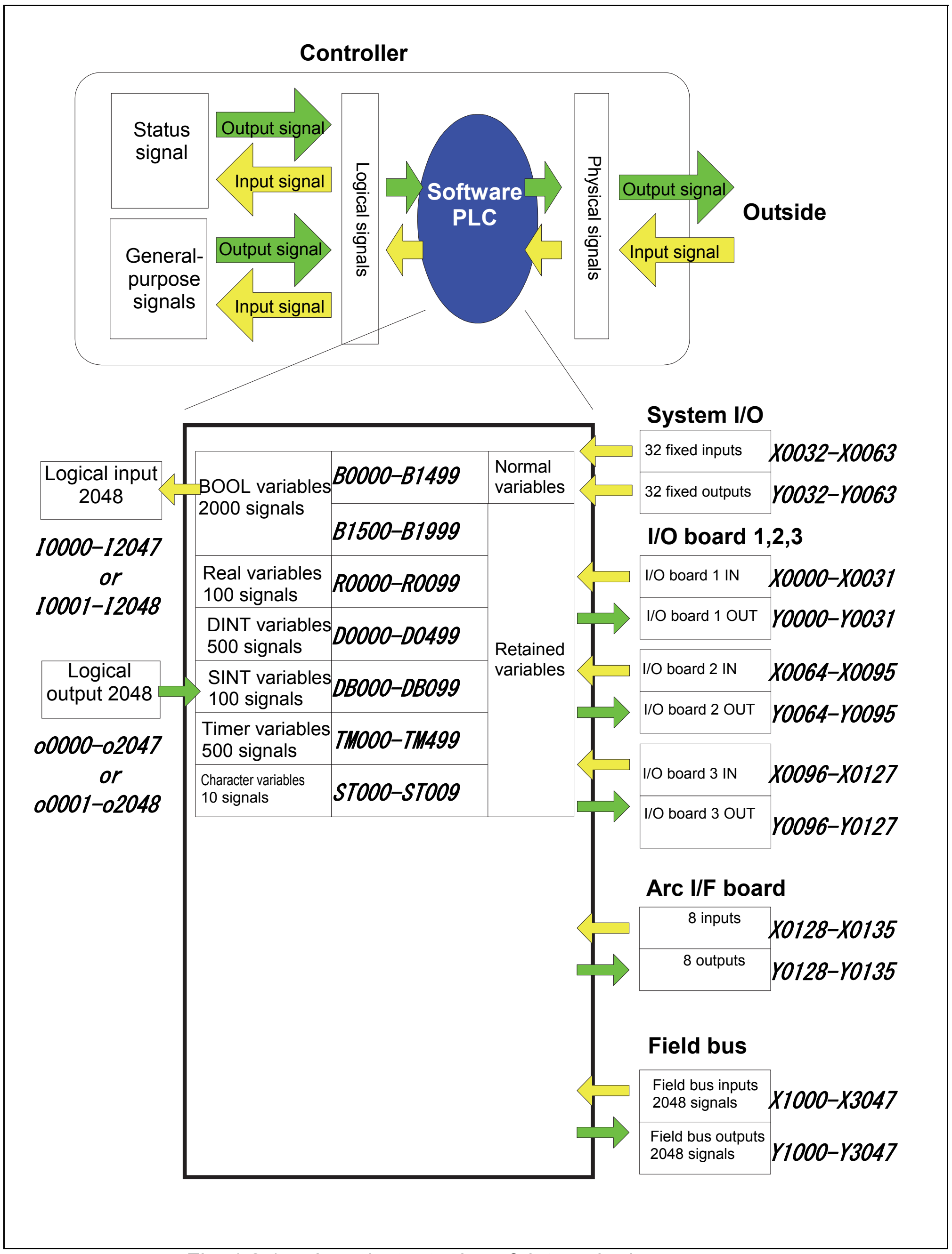


Fig. 1.2.1 Input/output relays & internal relays

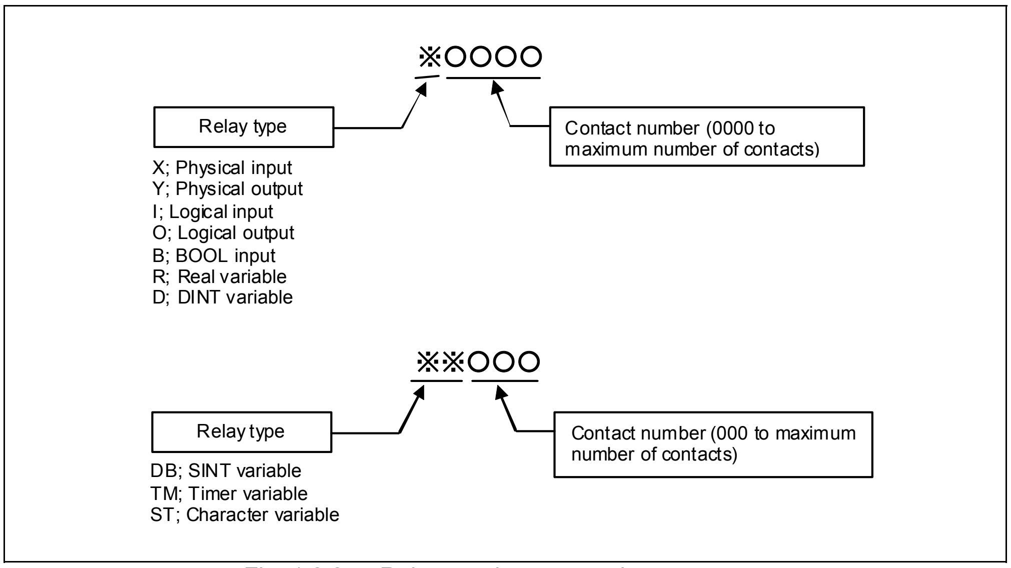


Fig. 1.2.2 Relay number expressions

1.2.1 Input/output relays

Table 1.2.1 Types of input/output relays

Names of input/output relays	Number of inputs / outputs	Explanation
Logical inputs	2048 signals	As seen from the software PLC, these are the input/output signals of the controller.These are all user I/O signals.
Logical outputs	2048 signals	
Fixed inputs	32 signals	These are physical input/output signals. These are the servo ON/OFF and other input/output signals which are used to control the operations inside the controller. They can only be referenced by the software PLC.
Fixed outputs	32 signals	
I/O board 1 IN	32 signals	These are physical input/output signals. These are the input/output signals for the CNIN (input) and CNOUT (output) connectors of the I/O board 1 provided as an optional accessory.
I/O board 1 OUT	32 signals	
I/O board 2 IN	32 signals	These are physical input/output signals. These are the input/output signals for the CNIN (input) and CNOUT (output) connectors of the I/O board 2 provided as an optional accessory.
I/O board 2 OUT	32 signals	
I/O board 3 IN	32 signals	These are physical input/output signals. These are the input/output signals for the CNIN (input) and CNOUT (output) connectors of the I/O board 3 provided as an optional accessory.
I/O board 3 OUT	32 signals	
Arc I/F board IN	8 signals	These are physical input/output signals. These are the input/output signals of the Arc I/F board provided as an optional accessory.
Arc I/F board OUT	8 signals	
Field bus inputs	2048 signals	These are physical input/output signals. These are the input/output signals for the DeviceNet, Profibus, RIO and other field buses provided as optional accessories. Field buses can be installed for up to four channels.
Field bus outputs	2048 signals	

POINT

The I/O board 1,2, and 3 are optional item.

POINT

In Fixed input/output signals, the software PLC can refer to signals shown below only. (There are unused signals in part.) For details, refer to “6.2.1 Fixed input/output”.

Fixed inputs	X0032 to X0063	32signals
Fixed outputs	Y0032 to Y0047	16signals

1.2.2 Internal relays

Table 1.2.2 Types of internal relays

Names of internal relays	Number of inputs / outputs	Explanation
BOOL variables	2000 signals	These variables have a status which is either ON or OFF (TRUE or FALSE). They are equivalent to the internal auxiliary relays used in the past.
Real variables	100 signals	These variables have real (floating-point) type consecutive values. (Short floating point)
DINT variables	500 signals	These variables have integer type consecutive values. (– 2147483648 to 2147483647)
SINT variables	100 signals	These variables have short integer type consecutive values. (–128 to +127)
Timer variables	500 signals	These variables have timer type consecutive values. (0 to 23h59m59s999ms)
Character variables	10 signals	These are character (ASCII code) strings. Max. 255 characters
Normal variables		These variables are reset when the power is turned off. Some BOOL variables (B0000 to B1499) are normal variables.
Retained variables		The data of these variables is retained in the memory even when the power is turned off. All variables except the B0000 to B1499 variables mentioned above are retained variables.

1.2.3 Field bus input/output

For details on field bus input/output, see “Chapter 6 Input/Output Relay List” and “Device Net Functions” in the options instruction manual.

1.3 When using Workbench

If the ISaGRAF Workbench is used, programming can be undertaken off-line (without using the robot controller). Connect Workbench with the robot controller using Ethernet, use Workbench for programming, download the programs to the robot controller, and execute them. Workbench is also used to make modifications and show monitor displays.
The programs created using Workbench cannot be displayed and edited using the robot teach pendant without taking further action. This is because the names of the I/O relays and internal relays used by Workbench and by the robot controller are different. In order to ensure complete compatibility at the display image file (STF file) level, the same database (shown in the figure below) containing the definitions for the I/O relays and internal relays as the one in the robot controller must be registered in Workbench ahead of time.
Furthermore, when a language other than LD (ladder) is used by Workbench, the programs cannot be displayed using the robot teach pendant even when the above action has been taken.

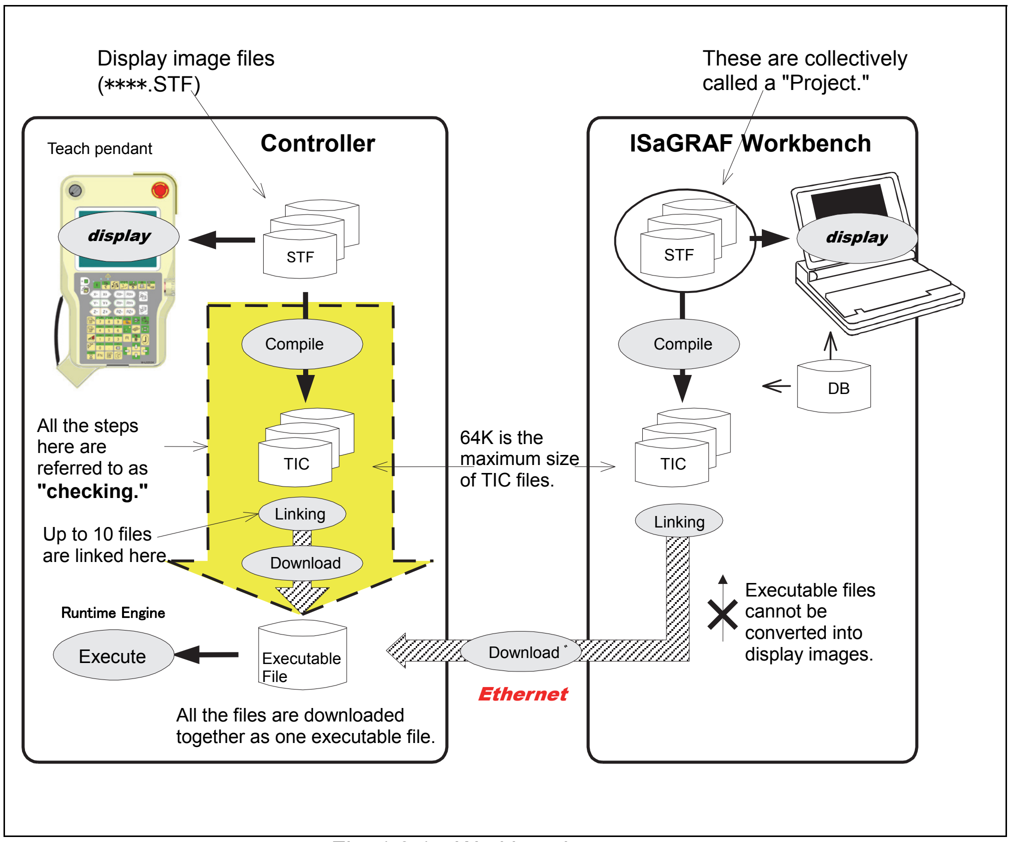

Display image files
(****.STF)
These are collectively called a "Project."
Teach pendant
Controller
ISaGRAF Workbench
display
STF
STF
display
Compile
Compile
DB
All the steps here are referred to as "checking."
TIC
64K is the maximum size of TIC files.
TIC
Linking
Linking
Up to 10 files are linked here.
Download
Runtime Engine
Execute
Executable File
Download
Executable files cannot be converted into display images.
Ethernet
All the files are downloaded together as one executable file.

Fig. 1.3.1 Workbench

The ISaGRAF Workbench must be purchased separately from *ICS Triplex ISaGRAF Inc.*

Chapter 2 Create program

The method used to create PLC programs using the ladder editor is described in this chapter.

2.1 Creating programs

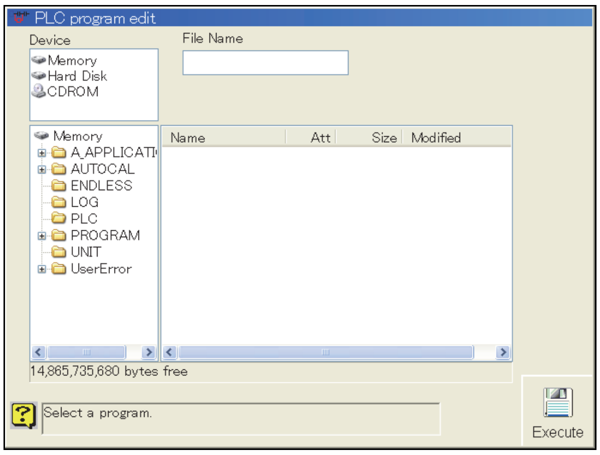
2.1.1 Starting the ladder editor

Use the ladder editor to create new PLC programs or editing programs which have already been taught. The ladder (LD language) displays appear on the teach pendant, enabling direct editing. Either the teach or playback mode may be established.
Any number of PLC programs can be recorded in the memory in the form of display image files.

Service Utilities

1 Select <Service Utilities> - [14 PLC Program Edit], and select [1 PLC program edit] from the menu items displayed.

>> A list of the ladder programs (display image files) is now displayed as follows.
Depending on the factory settings, Default.stf may be incorporated. For details, see "Chapter 1 Overview".



14,865,735,680 bytes free

Select a program.

A new ladder programs can be edited regardless of whether the scanning of a ladder program is in progress.
The programs listed here on the display are display image files which have been recorded in the memory, and they are not the files which are actually being scanned by the run time engine.

2 The first step which must be taken when a new program is created is to decide on the filename of the ladder program.
Align the cursor with the filename field, and press [ENABLE] + [EDIT].

>> The soft keyboard screen is now displayed so register the filename using this keyboard.
Hiragana, Katakana and Kanji can be used as well as alphanumerics for the file name.

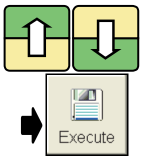
After inputting the filename, press f12 [Complete]. →4

IMPORTANT

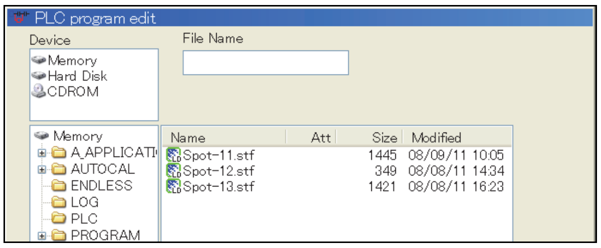
The file name for the "****.stf" has a limitation like the following. The limitation depends on the software version.

V3.21 or before: 16 letters (Even if the filename is more than 16 letters, it can be executed. But monitor function cannot work for the file.)

V3.22 or after: 124 letters



3 **If the ladder programs have already been recorded, use the up or down cursor key to select the file to be edited, and press f12 [Execute].**



PLC program edit

Device

Memory
Hard Disk
CDROM

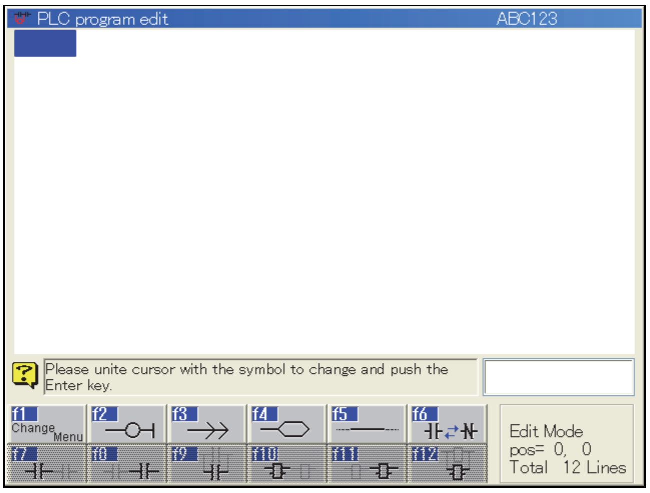
File Name

Memory
A_APPLICATI
AUTOCAL
ENDLESS
LOG
PLC
PROGRAM

Name	Att	Size	Modified
Spot-11.stf		1445	08/09/11 10:05
Spot-12.stf		349	08/08/11 14:34
Spot-13.stf		1421	08/08/11 16:23

Ladder programs (display image files) have the "stf" extension. A list of all the stf files already stored in the memory now appears on the display.

4 **When the file to be edited is entered, the ladder editor starts up, and a ladder-editing screen such as the one shown below appears.**



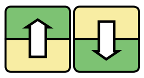
PLC program edit ABC123

Please unite cursor with the symbol to change and push the Enter key.

Edit Mode
pos= 0, 0
Total 12 Lines

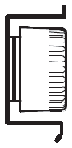
When a ladder program (display image file) already stored in the memory has been selected, the head of this program is displayed.

The "Insert" status is always established for ladder editing. No data is overwritten.



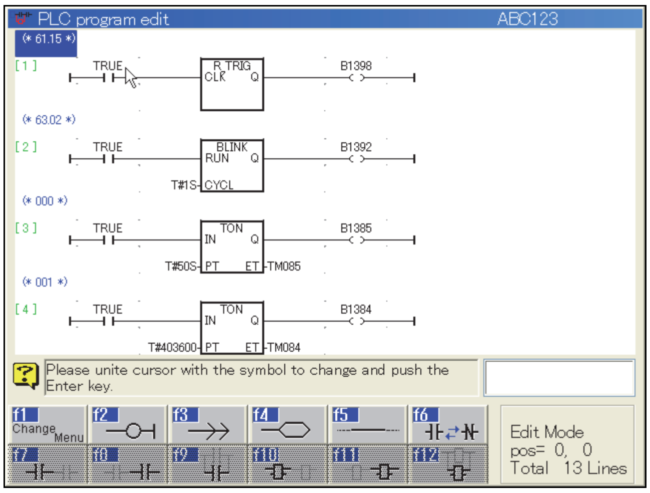
[Up/Down]

Or



[Jog Dial]

Scrolling upward and downward can be done using.

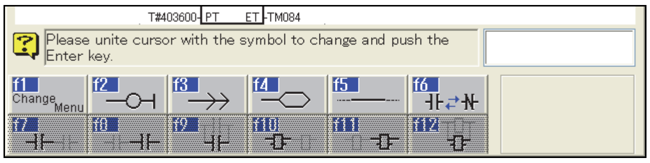


PLC program edit
ABC123
(* 61.15 *)
[1] TRUE R_TRIG CLK Q B1398
(* 63.02 *)
[2] TRUE BLINK RUN Q B1392 T#1S CYCL
(* 000 *)
[3] TRUE TON IN Q B1385 T#50S PT ET TM085
(* 001 *)
[4] TRUE TON IN Q B1384 T#403600 PT ET TM084
f1 Change Menu
f2
f3
f4
f5
f6
f7
f8
f9
f10
f11
f12
Ladder editing
Function key menu
Guide message input range

Please unite cursor with the symbol to change and push the Enter key.

Edit Mode
pos= 0, 0
Total 13 Lines

There are f keys for two screens: select the applicable menu items using f1 <Change Menu> to switch between the two screens, and edit the ladder program.



Please unite cursor with the symbol to change and push the Enter key.

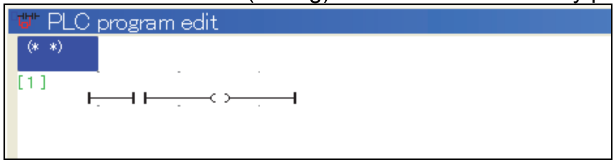
Fig. 2.1.1 Ladder program editing screen

2.1.2 Inputting a new rung

The "rung" is a closed circuit. Input a new rung according to the procedure shown below.

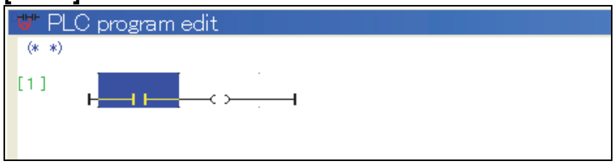
1 **Select the f keys with f1 <Operation Menu>, and then press [Enable] + f10 <Insert Rung> in a blank area.**

>> A new closed circuit (1 rung) is inserted and displayed as shown below.
The closed circuit (1 rung) can be also inserted by pressing [Enable] + [Enter].

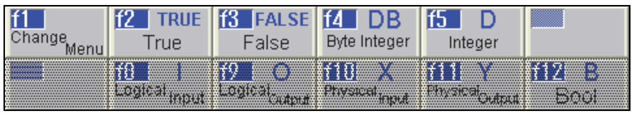


The area with a blue background represents the cursor position (area subject to editing).
A rung comment (i.e., a comment describable in units of closed circuits) is displayed inside (* *), while the rung number (i.e., the serial closed circuit number starting from "1") is displayed inside [ ].

2 **To input the contact number, put the cursor on the contact, and then press [Enter].**

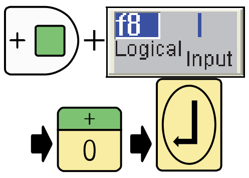


>> The f keys are now switched to the parameter selection menu as shown below.



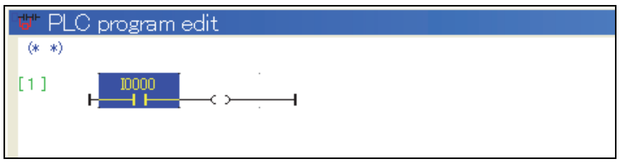
1st Page	
Function Name	Input example/Description
f1<Variable switch>	Switches parameter candidates.
f2<TRUE>	Insertion of TRUE constant (always ON)
f3<FALSE>	Insertion of FALSE constant (always OFF)
f4<SINT variable>	E.g: DB123 Specify the bit number as the relay number + "." + bit number 0~7.
f5<DINT variable>	E.g: D1234 Specify the bit number as the relay number + "." + bit number 0~31.
f6	
f7	
f8<Logic input>	E.g: I1234
f9<Logic output>	E.g: O1234
f10<Physical input>	E.g: X1234
f11<Physical output>	E.g: Y1234
f12<BOOL variable>	E.g: B1234

2nd Page	
Function Name	Input example/Description
f1<Variable switch>	Switches parameter candidates.
f8<String variable>	E.g: ST123
f9<Time variable>	E.g:TM123
f10<Real variable>	E.g: R1234



3 **To input the logical input I0000, press [Enable] + f8 <Logical Input>, and then [0] [Enter].**

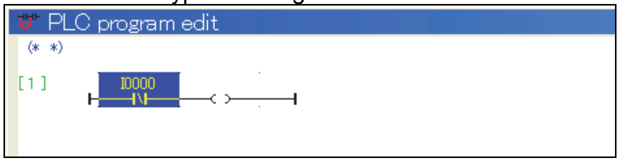
>> "I0000" is displayed above the contact symbol.



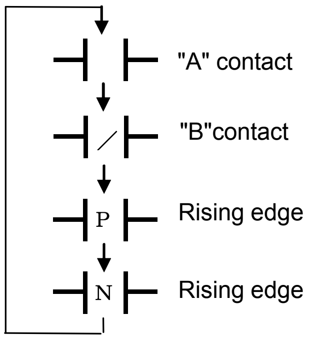
The f keys return to the original menu placement..

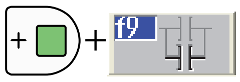
4 **Now try to change the contact type. Press f6 <Switch type> once.**

>>The contact type is changed to the "B".



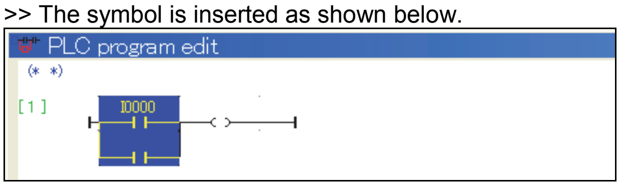
Every time f6 [Switch type] is pressed, the contact type is changed as shown below.



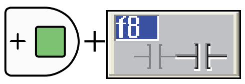


5 **Now try to insert the OR circuit. Press [Enable] + f9 <Insert OR>.**

>> The symbol is inserted as shown below.

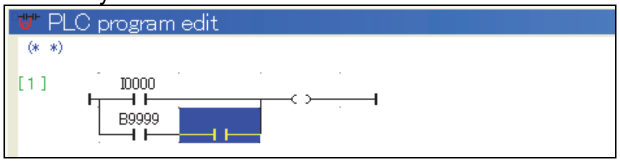


Input the contract number that was inserted according to the same procedure as those in the Steps 2 to 3.

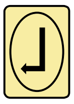


6 **Now try to insert the relay contact to the right.**
Press [Enable] + f8 <Insert a relay to the right of the cursor>.

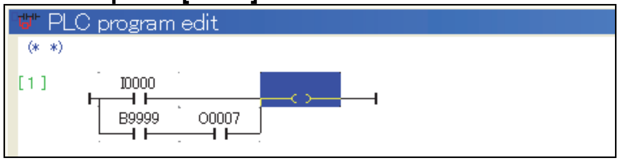
>> The symbol is inserted as shown below.



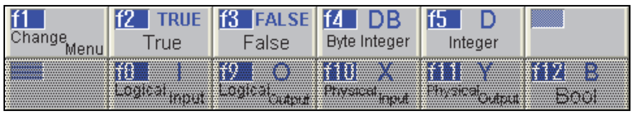
Input the contract number that was inserted according to the same procedure as those in the Steps 2 to 3.

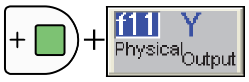


7 **Then, input the coil number.**
As in the case of inputting the contact number, put the cursor on the coil, and then press [Enter].

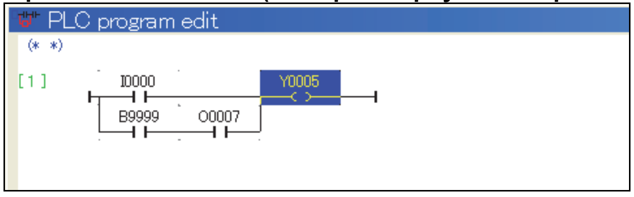


>> The f keys are switched to the Parameter Selection screen as shown below.

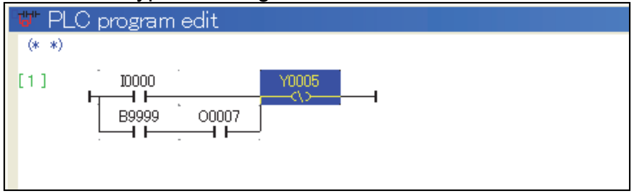




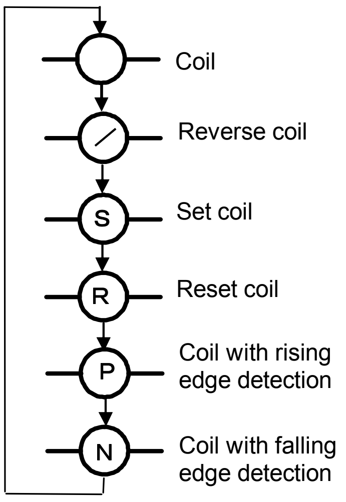
8 **According to the same procedure as that for inputting the contact number, input the coil number. (Example for physical output Y0005)**

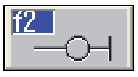


9 **Now try to change the coil type. Press f6 <Switch type> once.**
>> The coil type is changed to the reverse coil as shown below.

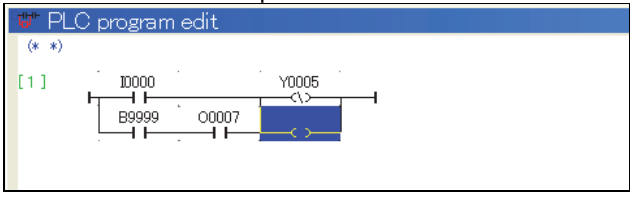


Every time f6 <Switch type> is pressed, the coil type is changed as shown below.


Coil
Reverse coil
S
Set coil
R
Reset coil
P
Coil with rising edge detection
N
Coil with falling edge detection



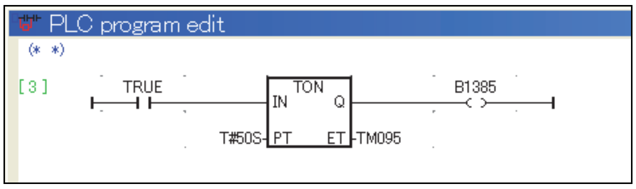
10 **To output multiple coils, put the cursor on the coils respectively, and then press f2 <Coil>.**
>> The coil number is input one after another in the same manner.



2.1.3 Inputting LD blocks

The LD block is a generic term used to refer to function commands such as comparison commands and timer commands. The LD blocks are input using numbers or names.
In the software PLC in this controller, all LD blocks use “Function blocks”. The "function blocks" are functions with input and output expressed in the form of a rectangle (block). Unlike conventional PLCs, the LD blocks are not handled as simple coils.

The following section describes a LD block taking a "timer command" as an example. When LD block No. 24 = TON (rising delay time), the "timer command" is programmed as shown below.

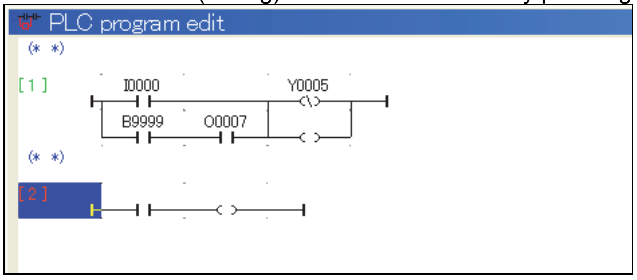


The square part that is located in the middle and marked with "TON" serves as the LD block (function block) having the timer function. When rise in the BOOL variable connected to the "IN" is detected, incrementing the timer value is initiated. When the timer completes its predetermined time, the BOOL output "Q" changes from FALSE to TRUE. To reference the status using a different rung, use the B1385 contact connected to the output "Q". Specify the setting of time-out period with the timer type constant connected to the "PT". In this case, the timer is set to 50 seconds. Furthermore, the timer variable connected to the "ET" output enables monitoring of the current elapsed timer value.
Thus, the function blocks feature ability to be used anywhere in the rungs and also to have output.

For details on what types of LD blocks are available, refer to information in “Chapter 7 List of command words”.
The following section describes how to input the LD blocks.

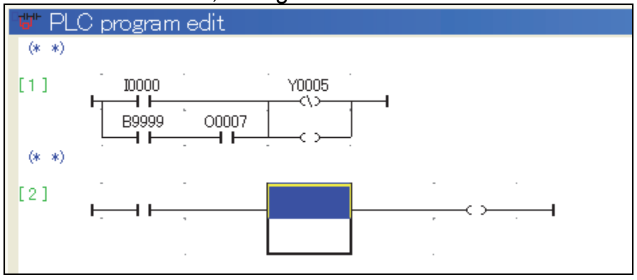
+ 🟩 + f10 Insert Rung

1 **Press [Enable] + f10 <Insert Rung> to add a new rung.**
The closed circuit (1 rung) can be also inserted by pressing [Enable] + [Enter].



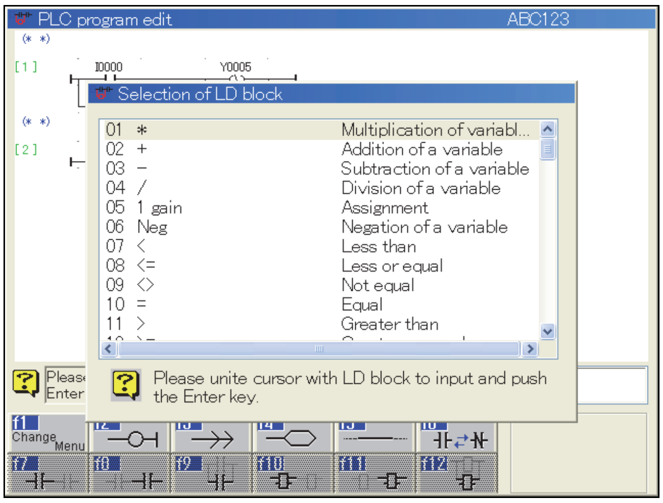
+ 🟩 + f11

2 **Put the cursor on the contact, and then press [Enable] + f11 <Insert LD block to the right>.**
>> As shown below, a single LD block is inserted between the contact and the coil.



3 **Put the cursor on the LD block, and then press [Enter].**

>> The list of LD block shown below is displayed.



PLC program edit ABC123

Selection of LD block

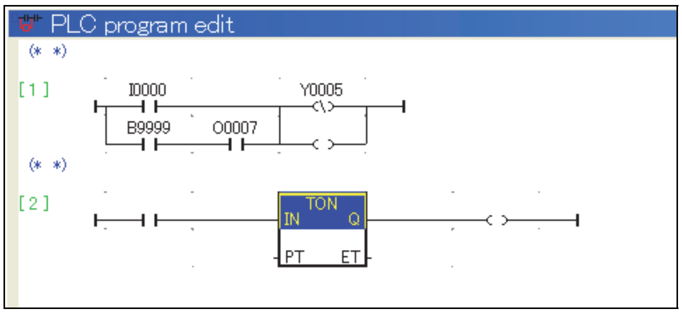
01	*	Multiplication of variabl...
02	+	Addition of a variable
03	−	Subtraction of a variable
04	/	Division of a variable
05	1 gain	Assignment
06	Neg	Negation of a variable
07	<	Less than
08	<=	Less or equal
09	<>	Not equal
10	=	Equal
11	>	Greater than

Please unite cursor with LD block to input and push the Enter key.

The LD blocks are displayed in ascending order by number.

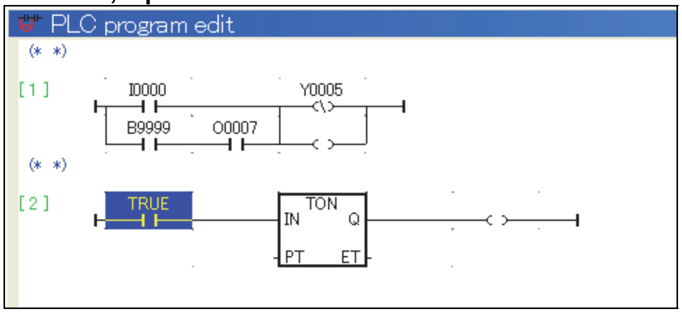
4 **Put the cursor on the LD block to be input, and then press [Enter].**
Alternatively, directly input the LD block number displayed on the extreme left.
For example, select No. 24 TON (Rising delay time).

>> The LD block display changes to TON as shown below.



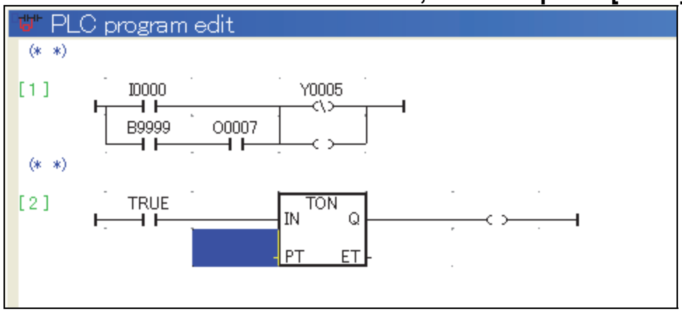
Consequently, it is apparent that the LD block "TON" must have two inputs (IN and PT) and two outputs (Q and ET). These must be all properly input.

5 **Put the cursor on the contact connected to the "IN", and then press [Enter].**
After that, input the desired contact.



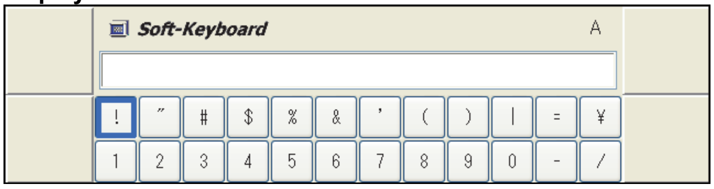
The example above shows that the TRUE (meaning a contact that is normally ON) has been input. The LD block "TON" initiates incrementing the timer value when rise in the input IN is detected. Consequently, the LD block "TON" can be said to be a timer that counts up at the time when the power supply is turned ON.

6 **Input 500msec as a time-out value.**
Put the cursor on the left of the "PT", and then press [Enter].



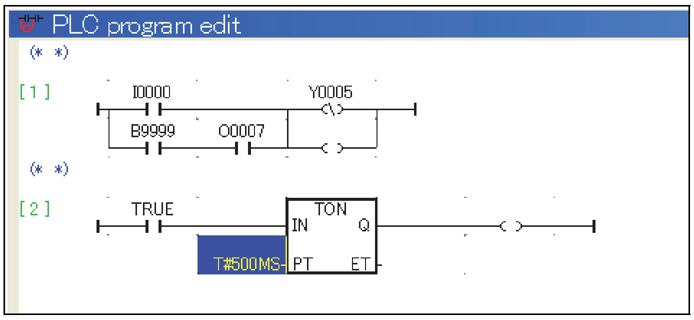
f6 Soft Keybord
or
+ +

7 **The display changes to the Parameter Input screen. Press f6 <Soft keyboard> to display the Character Input screen. Alternatively, press [Enable] + [Edit] to display this screen.**



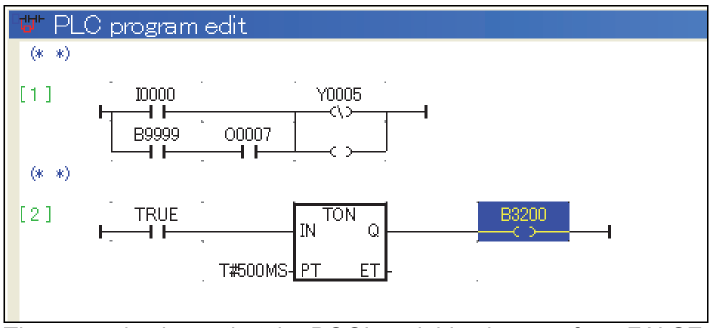
8 **To specify the time-out value, use the timer type constant starting with "T#" and ending with "S" or "MS". To specify the time-out value of 500msec, input "T#500MS" on the Character Input screen, and then press f12 <Complete>.**

>> The time-out value is displayed as shown below.

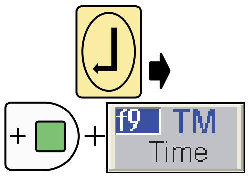


The timer type constant is available for the specifications in two units, milliseconds and seconds.
For 500 msec ... T#500MS
For 2 sec ... T#2S

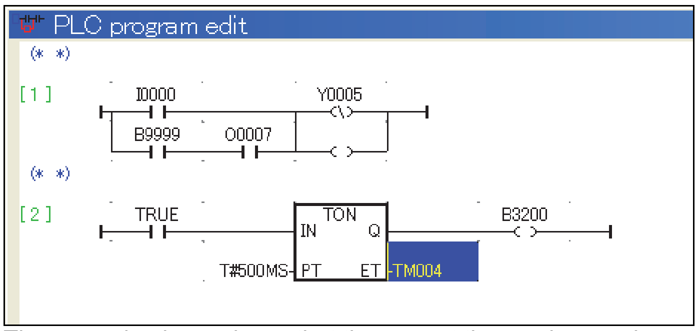
9 **Specify the output.**
Put the cursor on the coil connected to the output "Q", and then press [Enter]. After that, input the desired variable (or the BOOL variable for the LD block "TON").



The example shows that the BOOL variable changes from FALSE to TRUE when the timer times out.



10 **Lastly, specify the timer value monitor.**
Put the cursor on the right of the output "ET", and then press [Enter]. After that, select [Enable] + f9 <Time> from the parameter candidates to input the number of the timer variable.

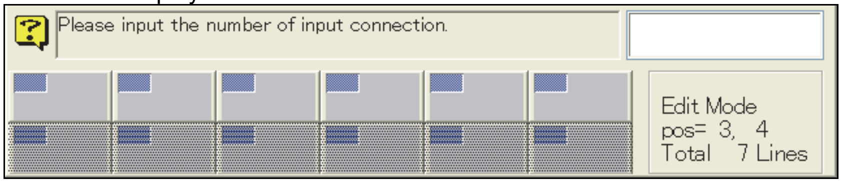


The example above shows that the current timer value can be monitored with the timer variable TM004.
Even if the timer value monitor is not used, it must be specified.

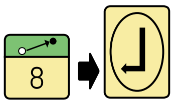
11 **With that, inputting the LD block "TON" is complete.**

To use LD blocks with the variable number of inputs and outputs, follow the procedure shown below.

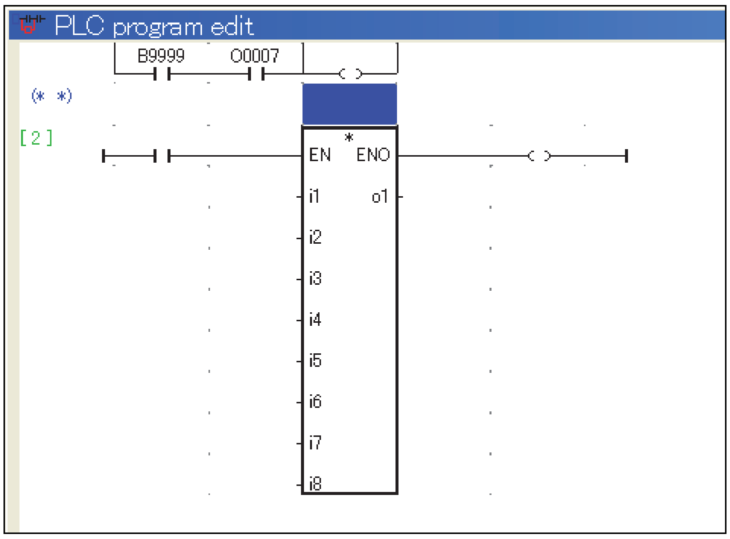
12 **Select the LD block 01 "Multiplying two or more variables".**
>> As shown below, a guide message requesting the number of input connecting wires is displayed.


Please input the number of input connection.

Edit Mode
pos= 3, 4
Total 7 Lines



13 **Input the number of input connecting wires in the range of 2 to 20, and then press [Enter].**
For example, input 8 for the number of input connecting wires.
>> As shown below, the LD block 01 having 8 inputs is displayed.

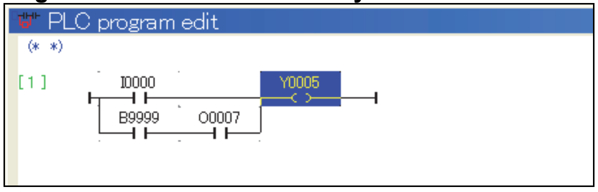


14 **Subsequently, specify inputs and outputs according to the same procedure as that aforementioned.**

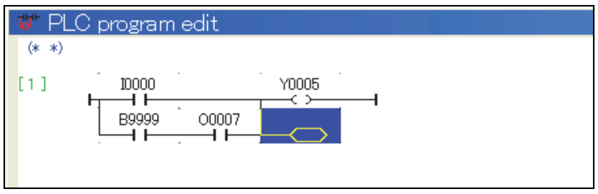
2.1.4 Inputting the return command

The return command is for returning to the start of a program without executing the subsequent circuits. When the return command is executed (when its logic state is TRUE), the operation returns to the start of the program without executing the subsequent circuits, and the program is executed from the start.

1 **Align the cursor with the coil symbol.**



2 **Press f4 <Return>.**
>> The return command has been added as shown below.

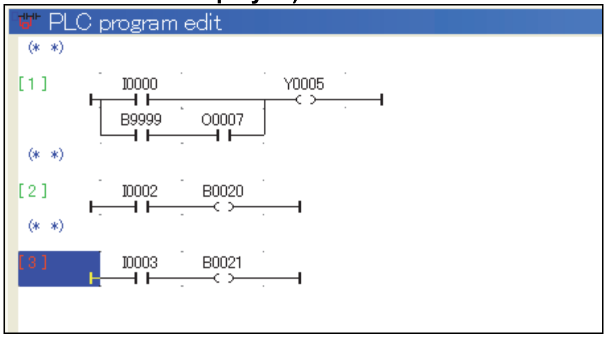


In the example given here, as soon as the Y0005 state becomes true, the return is executed, and the subsequent rungs are not executed. It is only after the Y0005 state becomes false that operation moves to execute the subsequent rungs.

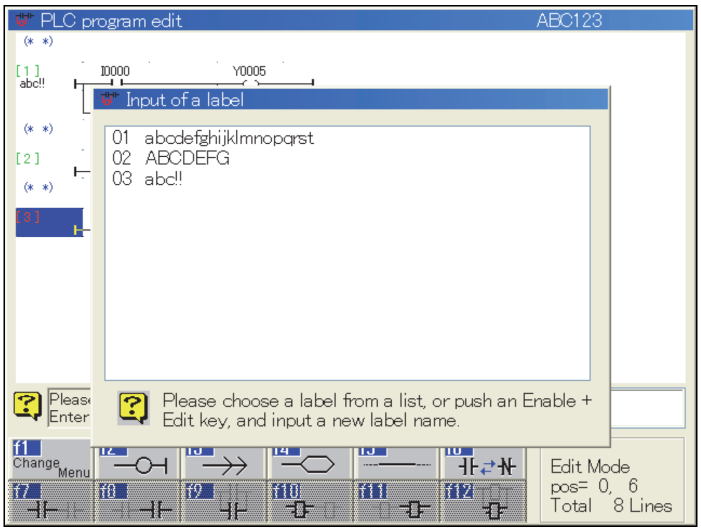
2.1.5 Inputting the jump command and label

Using a combination of the jump command and label, the program execution can be skipped to a desired rung. When the jump command is executed (as soon as its logic state is TRUE), operation jumps to the location of the label provided there without executing the subsequent circuits, and execution starts from that location.
Up to 1,000 labels can be input in an entire program.(40 alphanumerics)

1 **First, input the label.**
Start by aligning the cursor with the head of the circuit (where the rung number at the far left is displayed).



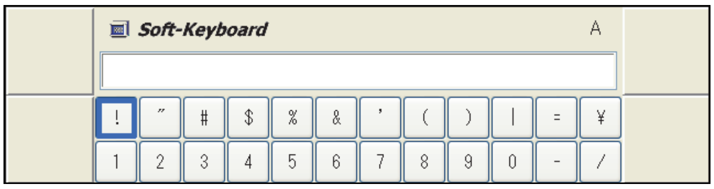
2 **Now press [Enter].**
>> A display listing the label names already registered now appears as shown below.
(The display is blank if not a single label has been registered.)



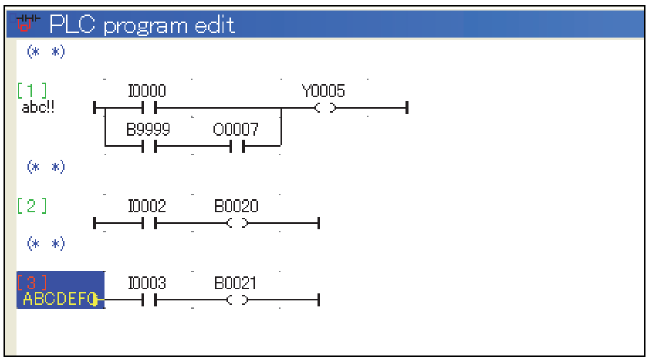
01 abcdefghijklmnopqrst
02 ABCDEFG
03 abc!!

Please choose a label from a list, or push an Enable + Edit key, and input a new label name.

3 **To register a new label, press [ENABLE] + [EDIT].**
>> The soft keyboard now starts up so input the name of the label to be registered.

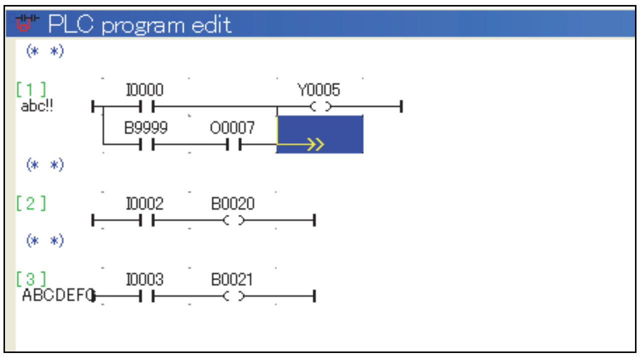


4 **After inputting the label name, the soft keyboard closes, and the registered label name is displayed at the far left as shown below.(Example: "ABCDEFG")**



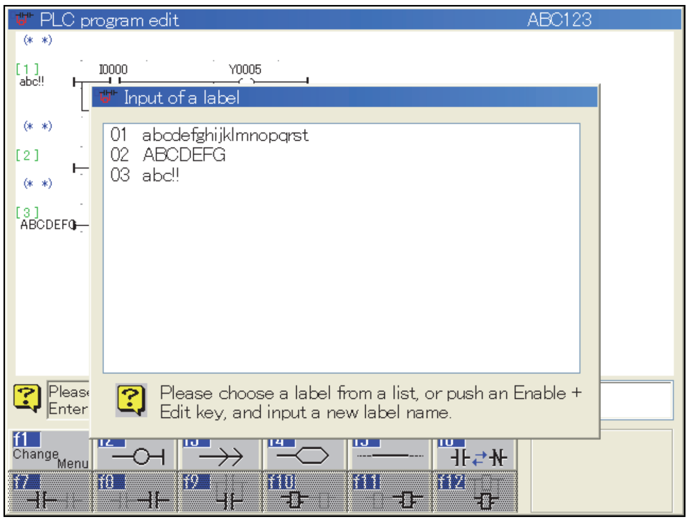
5 **Next, input the jump command.**
Align the cursor with the coil symbol, and press f3 [Jump].

>> The jump symbol is now input in an OR circuit as shown below.



6 **Align the cursor with the jump command, and press [Enter].**

>> A list of the registered labels such as the one below now appears on the display.

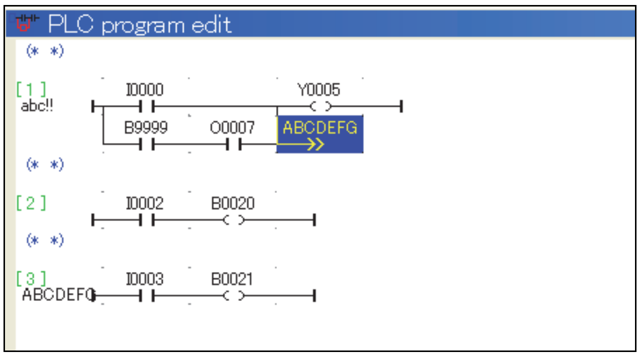


01 abcdefghijklmnopqrst
02 ABCDEFG
03 abc!!

Please choose a label from a list, or push an Enable + Edit key, and input a new label name.

7 Select the jump destination label using [Up/Down], and press [Enter].

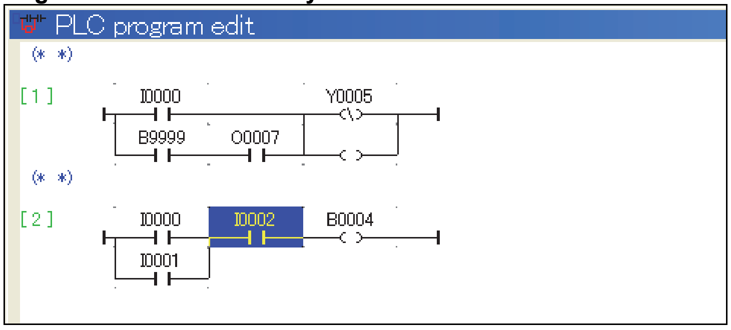
>> The name of the jump destination label is now input as shown below.(Example: "ABCDEFG")



In the example given here, the jump is executed as soon as the Y0005 state becomes TRUE, and rung number [2] is not executed. Operation is executed again from label ABCDEFG or, in other words, rung number [3].

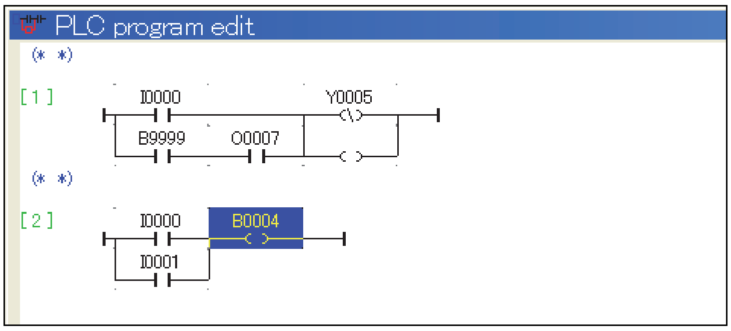
2.1.6 Deleting symbols

1 Align the cursor with the symbol to be deleted.

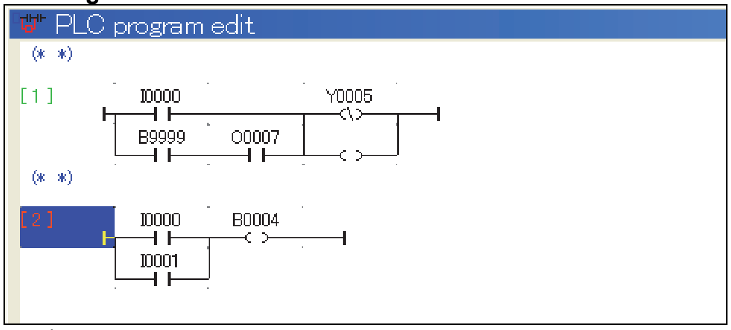


2 Press [DEL].

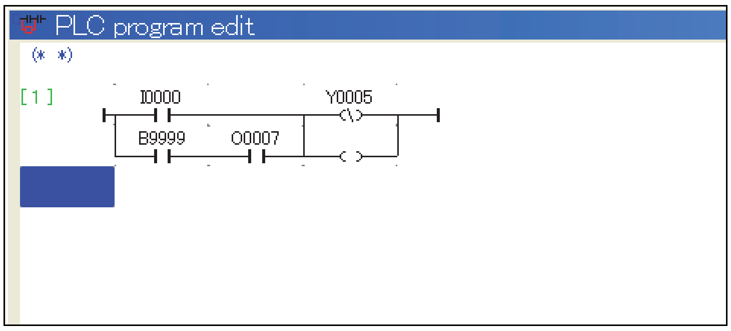
>> The symbol at the cursor position is deleted, and what is to the right side of that position is moved to the left and displayed.



3 **Align the cursor with the rung number field at the far left (number contained in the green brackets), and press [DEL]. The number is now deleted together with the rung.**



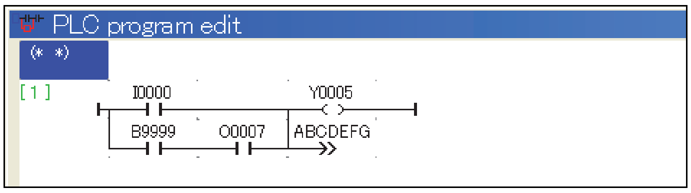
↓ [Delete]



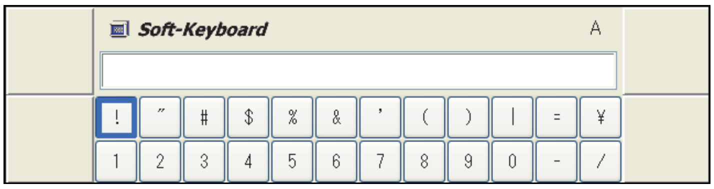
2.1.7 Inputting rung comments

A comment can each be input for all the rungs (closed circuits).
Up to 200 alphanumerics can be input as the rung comment.

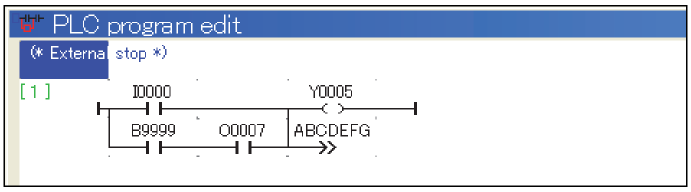
1 **Align the cursor with the rung comment field at the far left (area enclosed in the blue (* *)).**



2 **Now press [Enter].**
>> The soft keyboard now starts up so input the comment using this keyboard.



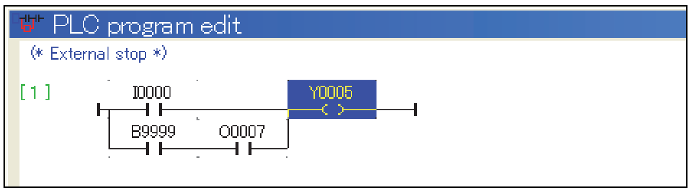
3 **After inputting the comment, the soft keyboard closes, and the rung comment is displayed as shown below.(Example: "External stop" is input)**



2.1.8 Inputting and display alias names

An alias is a comment provided for each and every symbol.
Ladder programs can be read with greater ease if the names of the contact applications and functions are set in advance.
Up to 8 alphanumerics can be input for an alias.
Simply by touching the f keys, it is possible to switch between the normal display (variable display) and alias name display.
Alias names are managed in ladder program (display image file) increments. If the same alias name is used in a multiple number of ladder programs, after one ladder program in which the alias name has been registered is created, the entire file is copied, and the copied file is edited.

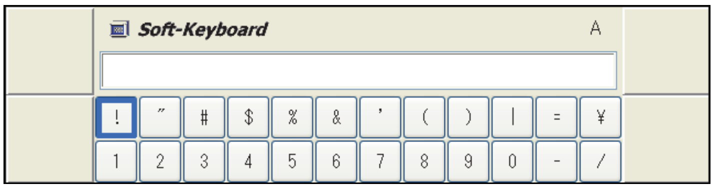
1 Align the cursor with the symbol whose alias name is to be set.



f6 Edit Alias

2 Now press f6 <Edit Alias>. If the f keys are not displayed, they will appear if the f key group is switched using f1 <Change Menu>.

>> The soft keyboard now starts up so input the alias name (comment) using this keyboard.



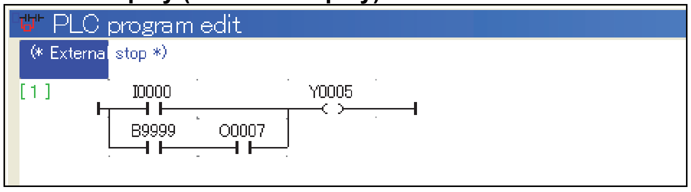
If the alias name has already been set, the name which has been set is displayed. Modifications can be made here.

3 After inputting the comment, the soft keyboard closes, and the operation returns to the ladder display screen.

f5 Disp Alias

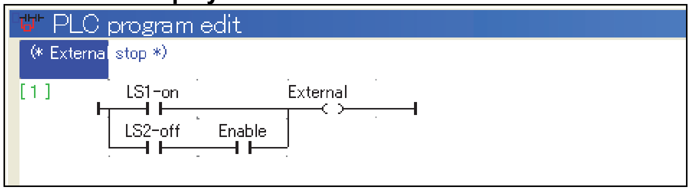
4 If f6 <Disp Alias> is pressed, the display is switched alternately between the alias name display and normal display. Symbols for which alias names have not been set are displayed with the same variable names as on the normal display.

Normal display (Variable display)



↑ ↓ f5 <Disp Alias>

Alias name display

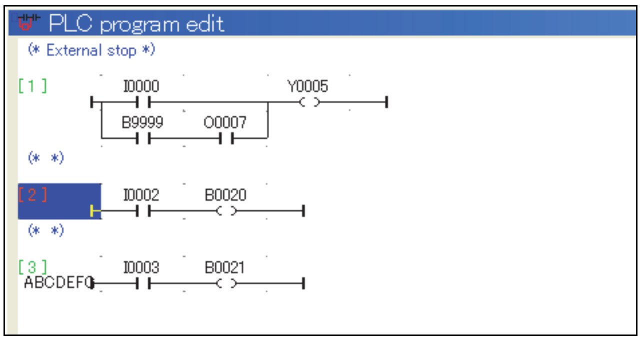


Due to screen space limitations, only the first 7 characters from the beginning are displayed for the alias names.

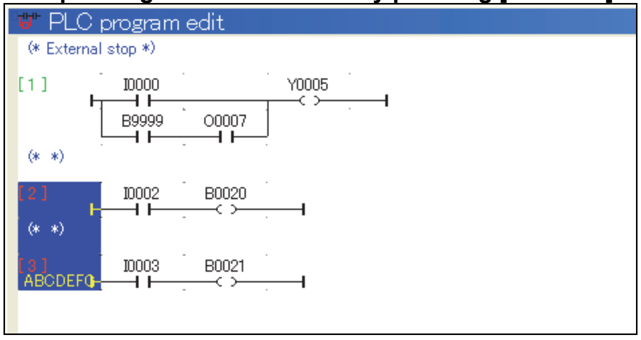
2.1.9 Cut & pasting

This function enables a multiple number of rungs (closed circuits) to be cut (deleted) or copied (stored in the paste buffer and then copied altogether). It is useful for editing.

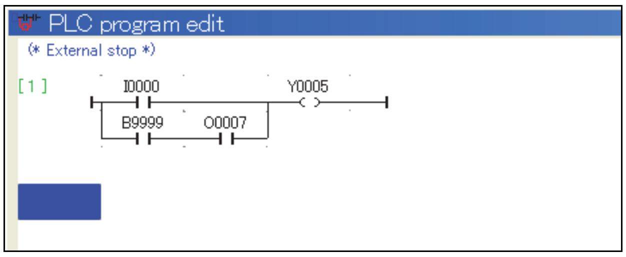
1 Align the cursor with the beginning (far left) of the rungs (closed circuits).



2 Multiple rungs can be selected by pressing [ENABLE] and [Up/Down] together.

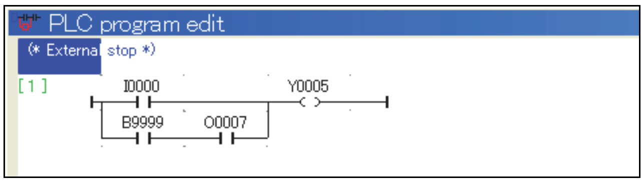


3 If [Enable] + f7 <Cut> is now pressed, the multiple number of rungs selected are deleted altogether.



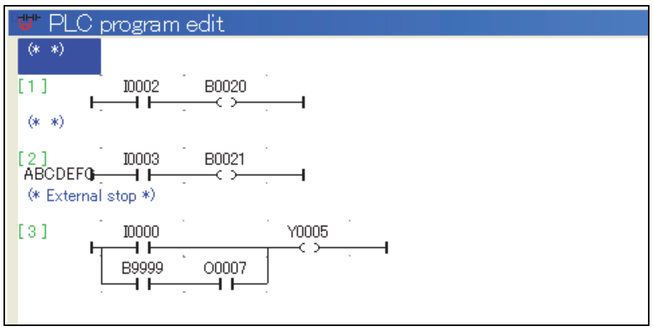
The multiple number of rungs deleted at this time are stored in the paste buffer.

4 Move the cursor to the location where the rungs are to be pasted.
Align the cursor with the beginning (far left) of the rungs (closed circuits).



+ + f9 Paste

5 **If [Enable] + f9 <Paste> is now pressed, the multiple number of rungs stored in the paste buffer are inserted above the cursor position.**



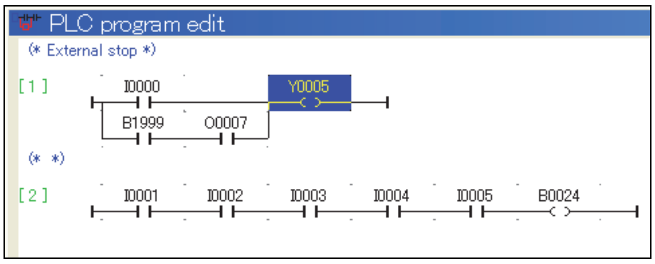
+ + f8 Copy

6 **If [Enable] + f8 <Copy> is pressed at step 3 above instead of [Enable] + f7 <Cut>, the multiple number of rungs can be stored in the paste buffer instead of being deleted (cut).**
In the same way, they can be inserted using f9 <Paste>.

2.1.10 Making it easier to view the ladders

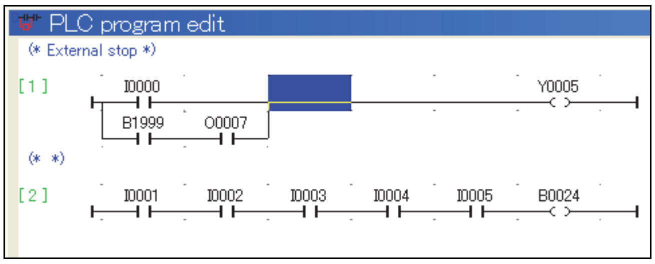
With the software PLC of this controller, the coil (far right) display position changes freely in accordance with the number of contacts. The “Horizontal bar” is added to align the coil (far right) display position.
The "horizontal bar" has nothing to do with the run time of the program.

1 **Align the cursor with the position where the "horizontal bar" is to be inserted.**



f5

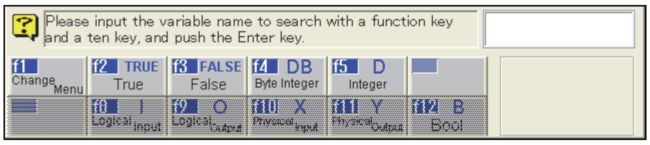
2 **Press f5 <Horizontal bar>.**
>> When this key is pressed once, the "horizontal bar" is added by amount equivalent to one column. In the example given here, if the key is pressed three times, the coil locations are aligned perpendicularly for easier viewing.



2.1.11 Searching

1 **Press f2 <Find>, wherever the cursor is positioned.**

>> The f key display shown below appears.

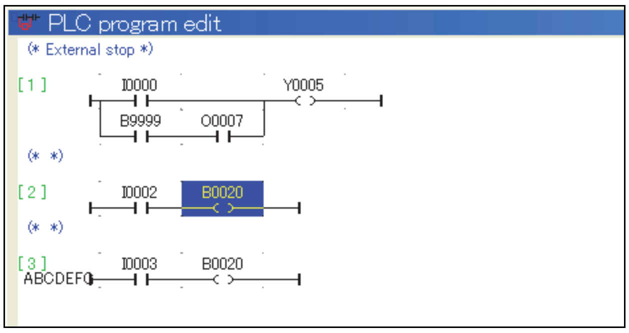


Please input the variable name to search with a function key and a ten key, and push the Enter key.

2 **To search the BOOL variable B0020, press [Enable] + f12 <BOOL> first, and then [0], [0], [2], [0], and [Enter] in the order described**

>> Searching is initiated in the backward direction from the current cursor position.
The cursor moves to a position in which the B0020 has been located.

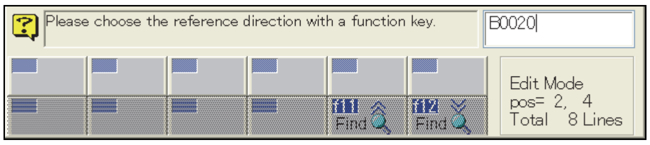
Number



POINT

Pressing **[Enable] + f12** <BOOL> and then specifying [2] and [0] disables searching. **When searching, input without abbreviating [O].**

3 **The f key display changes as shown below.**



Please choose the reference direction with a function key.

B0020

Edit Mode
pos= 2, 4
Total 8 Lines

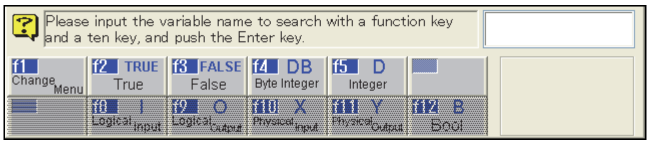
If [Enable] + f11 <Find> is pressed,
searching is continued in the forward direction to move the cursor to the next B0020.
If [Enable] + f12 <Find> is pressed,
searching is continued in the backward direction to move the cursor to the next B00200.

4 **To end searching, press [Reset/R].**

2.1.12 Replacing

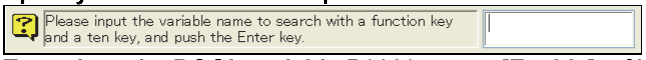
1 **Press f3 <Replace>, wherever the cursor is positioned.**

>> The f key display shown below appears.


Please input the variable name to search with a function key and a ten key, and push the Enter key.

2 **Specify the variable to be replaced.**

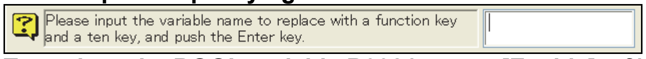
Number


Please input the variable name to search with a function key and a ten key, and push the Enter key.

To replace the BOOL variable B0020, press [Enable] + f12 <BOOL> first, and then [0], [0], [2], [0], and [Enter] in the order described.

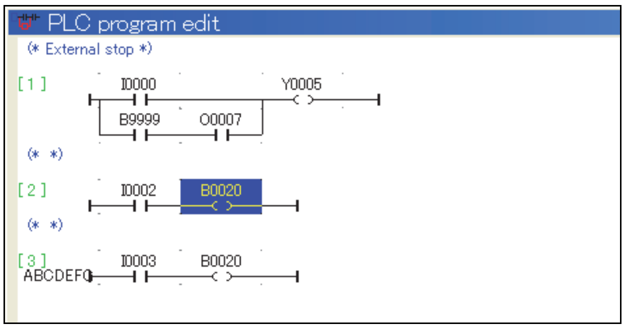
3 **Follow up with specifying the variable after one that has been replaced.**

Number


Please input the variable name to replace with a function key and a ten key, and push the Enter key.

To replace the BOOL variable B0021, press [Enable] + f12 <BOOL> first, and then [0], [0], [2], [1], and [Enter] in the order described.

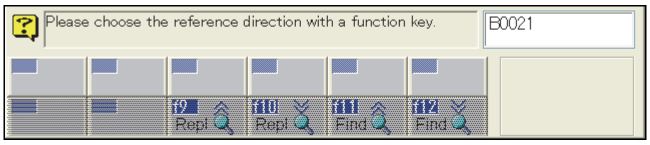
>> Searching is initiated in the backward direction from the current cursor position. The cursor moves to a position in which the B0020 to be replaced has been located.



POINT

Pressing [Enable] + f12 <BOOL> and then specifying [2] and [1] disables replacing. When searching, input without abbreviating [O].

4 **The f key display changes as shown below.**


Please choose the reference direction with a function key.

B0021

If [Enable] + f9 <Repl.> is pressed,
the symbol B0020 shown above is replaced with B0021 to continue searching in the forward direction, thus moving the cursor to the next B0020.
If [Enable] + f10 <Repl.> is pressed,
the symbol B0020 shown above is replaced with B0021 to continue searching in the backward direction, thus moving the cursor to the next B0020.
If [Enable] + f11 <Find> is pressed,
replacing is not executed to continue searching in the forward direction, thus moving the cursor to the next B0020.
If [Enable] + f12 <Find> is pressed,
replacing is not executed to continue searching in the backward direction, thus moving the cursor to the next B0020.

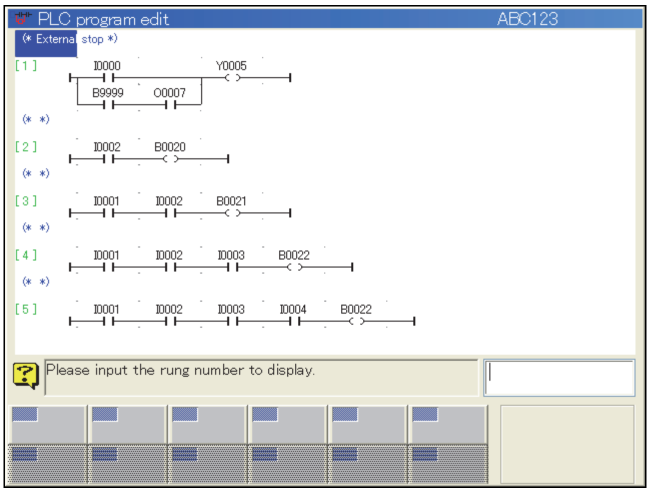
5 **To end searching and replacing, press [Reset/R].**

2.1.13 Jumping using a rung number

By specifying the rung number (closed circuit number) to be displayed, a jump can be initiated to that rung instantly.
This function is very useful when editing long ladder programs.

f4 Jump

1 **Press f4 <Jump> at any cursor position.**

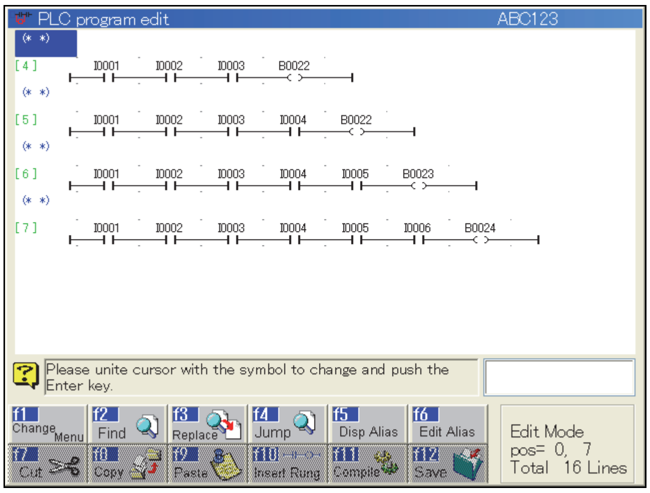


Please input the rung number to display.

Number

2 **Input the rung number (closed circuit number) to be displayed, and press [Enter].**
The rung number (closed circuit number) is the number enclosed in the green [ ] brackets displayed at the far left.

>> The specified rung (closed circuit) is displayed at the top of the screen.



Please unite cursor with the symbol to change and push the Enter key.

Edit Mode
pos= 0, 7
Total 16 Lines

2.2 Checking for any syntax errors

Finished PLC programs are descriptions of display images, which are not available for direct execution. To execute these programs, the display image file needs to be converted into an executable form. This conversion is called “compile”. Compliers (i.e.,software that executes compilation) are used to check whether or not display image files have any syntax errors and are executable. The files are converted into executable forms only after they pass those checks.
This section explains the method for checking whether compiling can be performed with the ladder editor open and specifying error sections.

INFO.

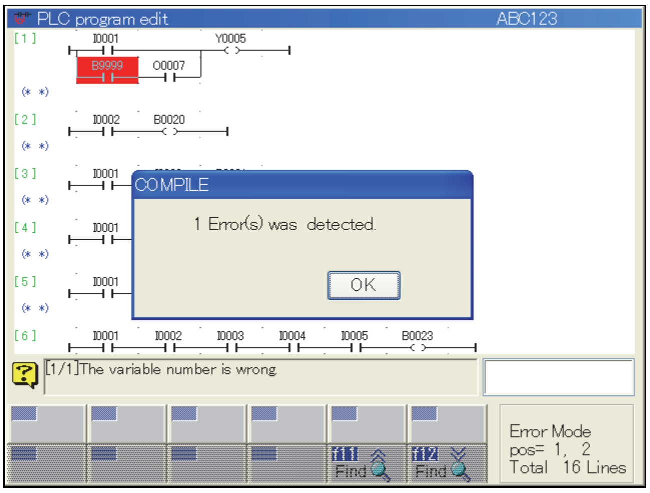
The created PLC program can be compiled and downloaded using the same operation as “3.2 From compile to download” from the ladder editor.

+ + f11 Compile

1 **Press [ENABLE] + f11 <Compile>.**
>> The compilation of a ladder program in the editing process is initiated. After a short while, the compilation result is displayed as shown below.

\>>When there are no compile errors
The window in step 5 is displayed. Proceed to step 6.

\>>When there are compile errors



[1/1]The variable number is wrong

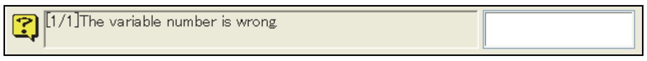
Error Mode
pos= 1, 2
Total 16 Lines

If there are compile errors, the total number of errors is displayed in the pop up window, and the first error location is highlighted in red.

+ + f11 Find

+ + f12 Find

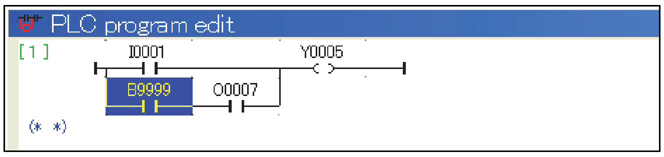
2 **Every time [Enable] + f11 <Find> or f12 <Find> is pressed, the cursor moves to the location of the corresponding error.**
The guide message area displays the error contents, respectively.
Figures inside "[ ]" represent the current cursor position/a total number of errors.



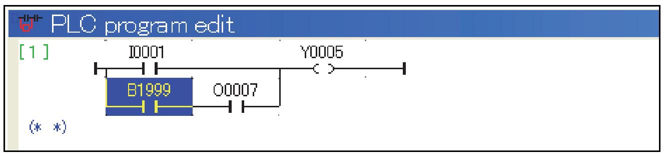
[1/1]The variable number is wrong

R

3 **To make corrections to the compile error, press [Enter] to clear the pop-up window, and then press [Reset/R] to return to the normal editing screen.**
>> The reverse video display in red that indicates the error locations returns to blue video display that indicates the editing status.

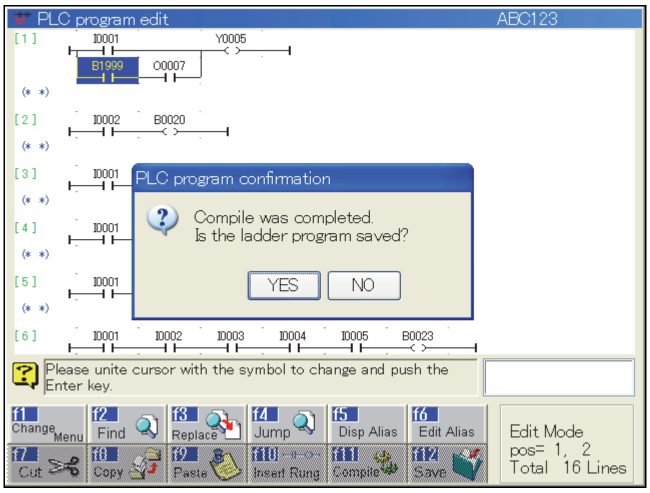


4 The example below shows that the BOOL variable number has deviated from the operating range (to 2000 at maximum). Correct B9999 to B1999.



5 Press [Enable] + f11 <Compile> again.

>> If no compilation errors are detected at all, the message shown below is displayed.



Compile was completed.
Is the ladder program saved?

Please unite cursor with the symbol to change and push the Enter key.

Edit Mode
pos= 1, 2
Total 16 Lines

6 To continue editing without directly saving the program, use [Right/Left] to select [NO], and then press [Enter].

Selecting [YES] makes it possible to save the program in the editing process.
For details, refer to information in the next section "2.3 Saving program in editing process".

2.3 Saving program in editing process

Save the finished PLC program in the memory in the form of display image.

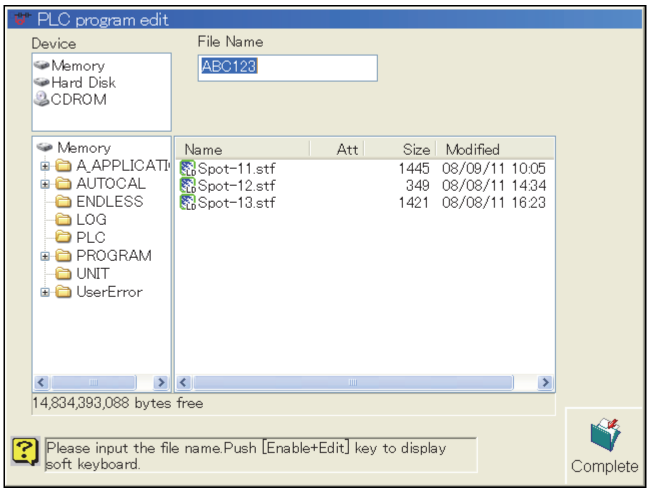
INFO.

Compiling and download of the created PLC program can also be done from the ladder editor. Operations are the same as "3.2 From compile to download".

+ + f12 Save

1 **Press [Enable] + f12 <Save>, wherever the cursor is positioned.**

>> The ladder program (display image file) on display is saved in the memory.



PLC program edit

Device: Memory, Hard Disk, CDROM

File Name: ABC123

Memory: A_APPLICATI, AUTOCAL, ENDLESS, LOG, PLC, PROGRAM, UNIT, UserError

Name	Att	Size	Modified
Spot-11.stf		1445	08/09/11 10:05
Spot-12.stf		349	08/08/11 14:34
Spot-13.stf		1421	08/08/11 16:23

14,834,393,088 bytes free

Please input the file name.Push [Enable+Edit] key to display soft keyboard.

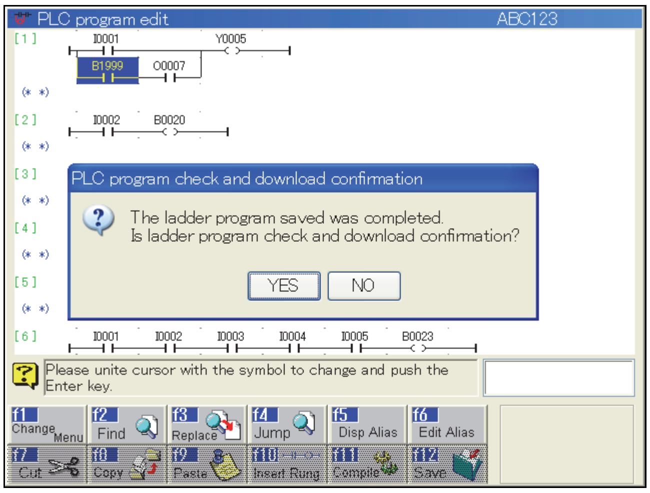
Complete

Make sure the display image file with a name that was specified before the editing was initiated is still on the List.

Complete

2 **Press f12 <Complete>.**

\>>The following pop-up message is displayed.



PLC program edit ABC123

PLC program check and download confirmation

The ladder program saved was completed.
Is ladder program check and download confirmation?

YES NO

Please unite cursor with the symbol to change and push the Enter key.

3 **To continue editing without checking and transferring the program, use [Right/Left] to select [NO], and then press [Enter].**

Selecting [YES] makes it possible to check and transfer the program.

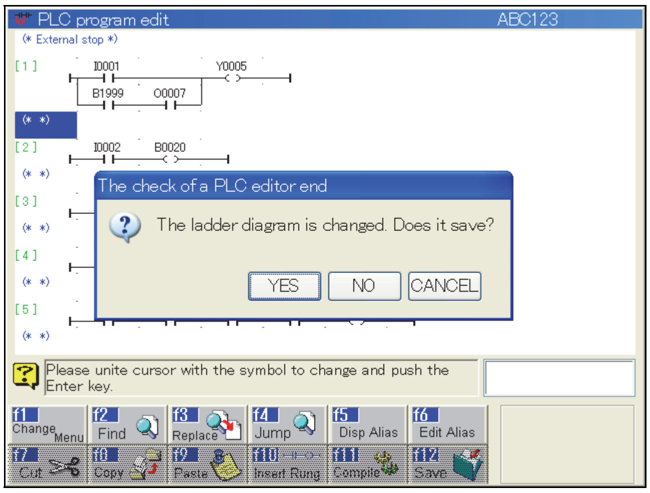
For details on program checking and transfer, see Chapter 3 "Program Check".

2.4 Exiting editor

Exit the ladder editor.

1 **Press [Reset/R] at any cursor position.**

>> If the ladder program has been changed, the message shown below is displayed.



PLC program edit ABC123

(* External stop *)

The check of a PLC editor end

The ladder diagram is changed. Does it save?

Please unite cursor with the symbol to change and push the Enter key.

If the ladder program has not been changed, the editor is exited without displaying the message.

2 **To exit the editor without saving the ladder program, use [Right/Left] to select [NO], and then press [Enter].**

Selecting [YES] makes it possible to save the ladder program in the editing process sand then exit the editor.

Selecting [Cancel] does not exit the editor.

NOTE

Chapter 3 Program check

This chapter describes the series of operations which are performed up to the start of the execution of the ladder program (display image file) which has been created.

3.1 General outline of program check

In order to execute the program file (display image file) which has been created, it must first be converted into an executable format. This process is known as compiling.
Next, the file is transferred to the run time engine (software that executes scanning). This process is known as downloading.
With this controller, this series of operation is collectively called "checking."

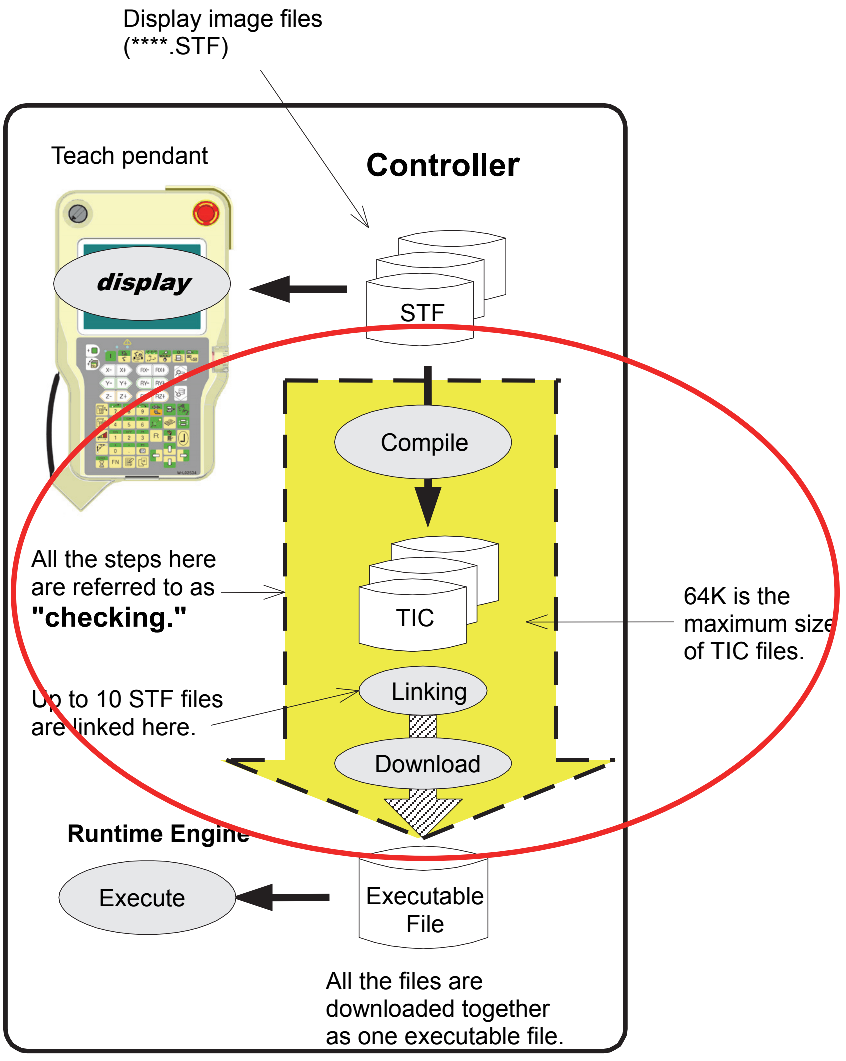

Display image files
(****.STF)
Teach pendant
Controller
display
STF
Compile
TIC
All the steps here are referred to as "checking."
64K is the maximum size of TIC files.
Up to 10 STF files are linked here.
Linking
Download
Runtime Engine
Execute
Executable File
All the files are downloaded together as one executable file.

Fig. 3.1.1 The structure of the software PLC

A multiple number of ladder programs (display image files) can be stored in the internal memory. As shown in Fig. 3.1.2, first store all the ladder programs for the intended application in the memory, and then select the ones to be run, and proceed with the compiling and downloading steps.

Up to ten ladder programs can be linked together and run using the software PLC of this controller. Editing of long programs is facilitated by dividing them up. In the example given in Fig. 3.1.2, three ladder programs are linked together.
If multiple ladder programs are linked together and used in this manner, the sequence in which they are to be run must be specified at the compiling stage.

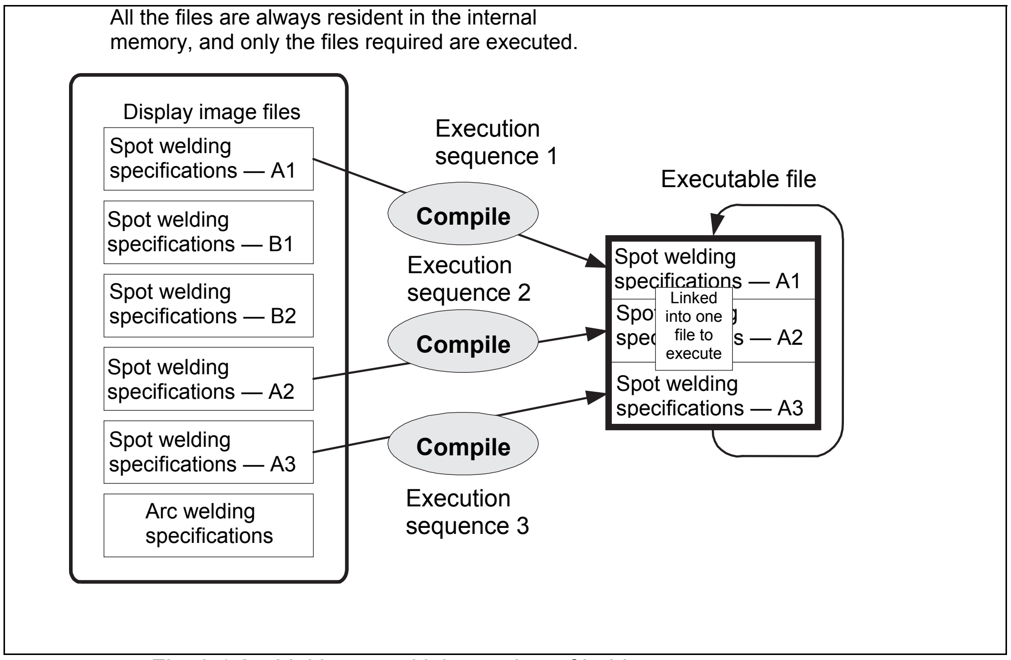

All the files are always resident in the internal memory, and only the files required are executed.
Display image files
Spot welding specifications — A1
Spot welding specifications — B1
Spot welding specifications — B2
Spot welding specifications — A2
Spot welding specifications — A3
Arc welding specifications
Execution sequence 1
Compile
Execution sequence 2
Compile
Compile
Execution sequence 3
Executable file
Spot welding specifications — A1
Spot welding specifications — A2
Linked into one file to execute
Spot welding specifications — A3

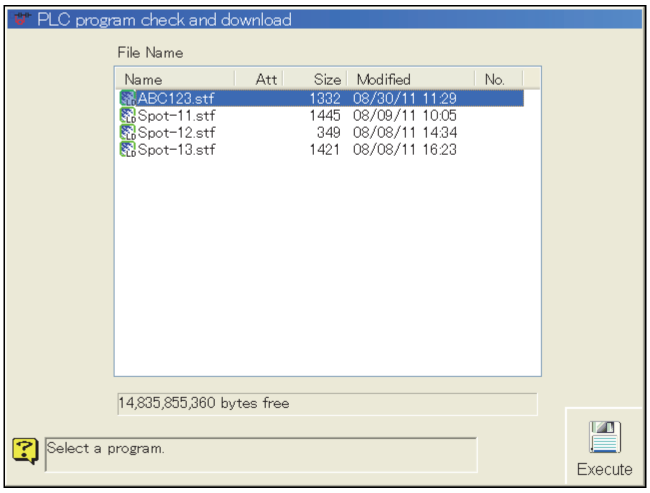
Fig. 3.1.2 Linking a multiple number of ladder programs

3.2 From compiling to downloading

This section describes the operations involved in selecting ladder programs (display image files) stored in the internal memory, compiling them while specifying the running sequence, and downloading.

1 **Select <Service Utilities> - [14 PLC Program Edit], and select [3 PLC program check] from the menu items displayed.**

>> A list of the ladder programs (display image files) is now displayed as follows.



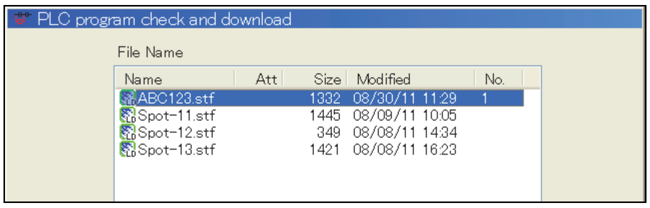
Name	Att	Size	Modified	No.
ABC123.stf		1332	08/30/11 11:29	
Spot-11.stf		1445	08/09/11 10:05	
Spot-12.stf		349	08/08/11 14:34	
Spot-13.stf		1421	08/08/11 16:23	

14,835,855,360 bytes free

Select a program.

2 **Select the ladder program to be compiled using [Up/Down], and press [Enter].**

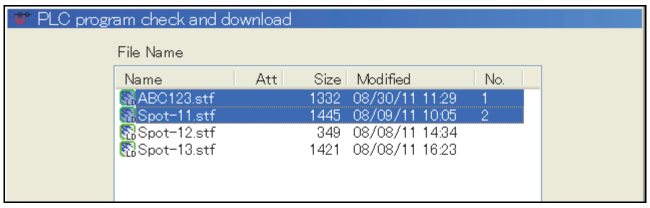
>> The selected ladder program is highlighted in blue, and the figure "1" appears on its right. This indicates that when a multiple number of ladder programs are to be linked together and run, this file is the ladder program which will be run first.



Name	Att	Size	Modified	No.
ABC123.stf		1332	08/30/11 11:29	1
Spot-11.stf		1445	08/09/11 10:05	
Spot-12.stf		349	08/08/11 14:34	
Spot-13.stf		1421	08/08/11 16:23	

3 **Select the next ladder program to be compiled using [Up/Down], and press [Enter].**

>> The selected ladder program is highlighted in blue, and the figure "2" appears on its right. This indicates that when a multiple number of ladder programs are to be linked together and run, this file is the ladder program which will be run second.



Name	Att	Size	Modified	No.
ABC123.stf		1332	08/30/11 11:29	1
Spot-11.stf		1445	08/09/11 10:05	2
Spot-12.stf		349	08/08/11 14:34	
Spot-13.stf		1421	08/08/11 16:23	

In this way, the ladder programs are selected in the sequence in which they will be run.
If only one executable file is sufficient, select just the one file.

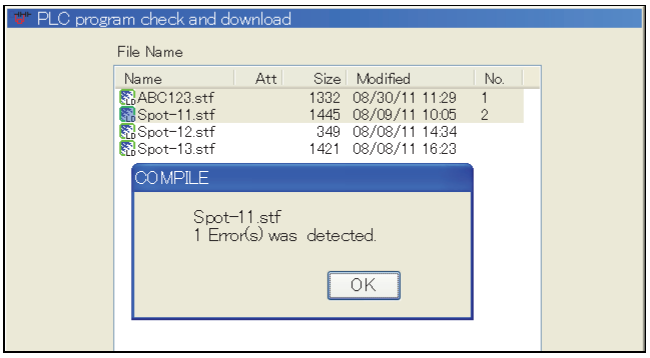
4 **If a mistake was made in specifying the execution sequence, move the cursor to the ladder program concerned, and press [BS].**

\>>The display of the ladder program number in the execution sequence is now cleared.

Execute

5 **After all the ladder programs have been selected, press f12 <Execute>.**

>>The programs are compiled in sequence starting with execution sequence number 1. Upon completion of the compiling, the results are displayed as shown below.


PLC program check and download

File Name

Name	Att	Size	Modified	No.
ABC123.stf		1332	08/30/11 11:29	1
Spot-11.stf		1445	08/09/11 10:05	2
Spot-12.stf		349	08/08/11 14:34	
Spot-13.stf		1421	08/08/11 16:23	

COMPILE

Spot-11.stf
1 Error(s) was detected.

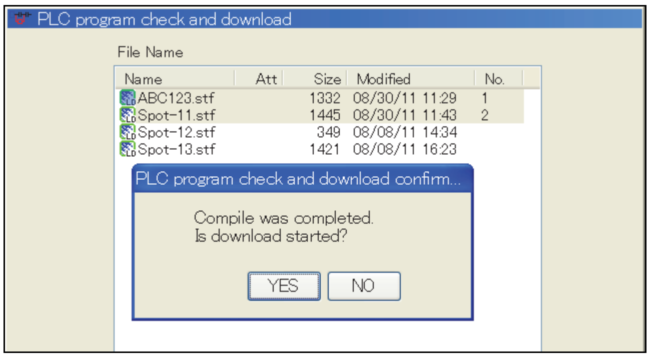
OK

In the example given here, file ABC456.stf is error-free but one compilation error was found in file ABC123.stf.

R

6 **If one or more compilation errors have been found, use [Reset/R] to exit the program check menu.**
Start the ladder editor using [1 PLC program edit], resume compiling, pinpoint the error locations and make corrections.

7 **If there are no compilation errors, the message shown below appears.**


PLC program check and download

File Name

Name	Att	Size	Modified	No.
ABC123.stf		1332	08/30/11 11:29	1
Spot-11.stf		1445	08/30/11 11:43	2
Spot-12.stf		349	08/08/11 14:34	
Spot-13.stf		1421	08/08/11 16:23	

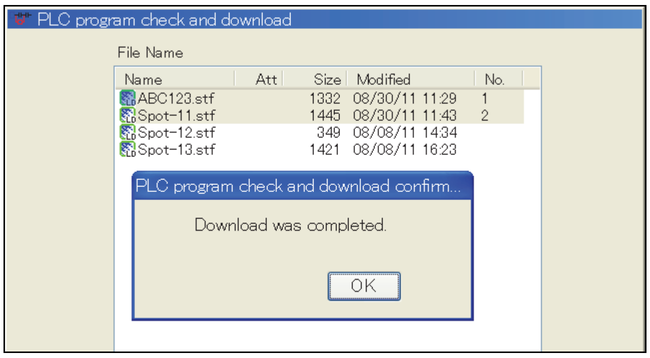
PLC program check and download confirm...

Compile was completed.
Is download started?

YES NO

8 **Select "Yes" using [Right/Left], and press [Enter].**

>> The multiple number of ladder programs which have been compiled are linked together into a single program and downloaded to the run time engine. When the file downloading is completed properly, the message shown below appears.


PLC program check and download

File Name

Name	Att	Size	Modified	No.
ABC123.stf		1332	08/30/11 11:29	1
Spot-11.stf		1445	08/30/11 11:43	2
Spot-12.stf		349	08/08/11 14:34	
Spot-13.stf		1421	08/08/11 16:23	

PLC program check and download confirm...

Download was completed.

OK

The program has now been transferred to the run time engine.
If the PLC (run time engine) is already started, it stops scanning temporarily, and after the program has been downloaded, resumes scanning.

R

9 **Exit the menu using [Reset/R].**

>> If PLC (run time engine) is not started, set it to start using the procedure in “3.3 Program start, stop, and disconnect”.

3.3 Program start, stop and disconnect

This section describes the procedures for starting and stopping the scanning of the ladder program downloaded from the run time engine. This operation is done from the constant menu.
Normally, if the PLC status setting is set to “Running”, it need not be changed later. Scanning is started by turning on the power of the controller.
Furthermore, it is possible for the software PLC not to be used (=disconnected) temporarily depending on the connection status of the peripheral devices etc.

The PLC status setting has the following three status settings.

Table 3.3.1 Program start, stop and disconnect

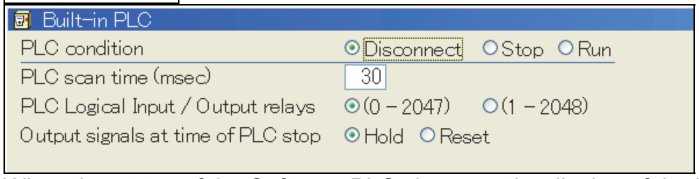
Setting choice	Explanation
Disconnect	The built-in PLC is not used. In other words, the logical inputs/outputs and physical inputs/outputs are connected on a one-on-one basis in this status. The PLC (run time engine) does not scan the downloaded programs.
Stop	The built-in PLC is used. However, since the PLC (run time engine) does not scan downloaded programs, no further changes occur in the physical I/O statuses.
Run	The built-in PLC is used. The PLC (run time engine) scans the downloaded programs. <Start> is not established automatically even when the downloading is initiated in the program check. The status must be changed to <Start> without fail using this menu.

Constant Setting

1 **In the teach mode, select f 5 <Constant Setting> - [1 Control Constants], and select [6 Built-in PLC] from the menu items displayed.**

>> The setting menu related to the built-in PLC now appears as shown below.

V3.33 or before

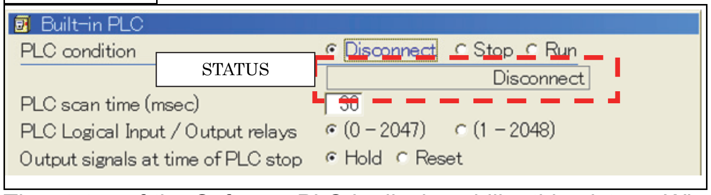


Built-in PLC

PLC condition	[✓ Disconnect] Stop Run
PLC scan time (msec)	30
PLC Logical Input / Output relays	[✓ (0 – 2047)] (1 – 2048)
Output signals at time of PLC stop	[✓ Hold] Reset

When the status of the Software PLC changes, the display of the “PLC condition” will change.

V3.34 or after



Built-in PLC

PLC condition	[✓ Disconnect] Stop Run
STATUS	Disconnect
PLC scan time (msec)	30
PLC Logical Input / Output relays	[✓ (0 – 2047)] (1 – 2048)
Output signals at time of PLC stop	[✓ Hold] Reset

The status of the Software PLC is displayed like this picture. When the condition of the Software PLC changes (e.g. “I/O simulation” function is enabled, etc.), this status display will also change.

2 **Select “Disconnect/Stop/Run” by pressing [ENABLE] + [Right/Left] together.**

Complete

3 **Press f12 [Complete].**

>> Operation is immediately transferred to the “Disconnect/Stop/Run” control status.

DANGER

If the PLC status setting is changed to "Running", the PLC program input/output signal status also changes accordingly. It is extremely dangerous to perform this operation while work piece is gripped or while there is interference with any of the peripheral devices. Take sufficient care when performing this operation.

NOTE

Chapter 4 Program verify

This chapter describes how the programs of the software PLC are verified.
"Verify" refers to checking whether the programs displayed and edited by the ladder editor are identical to the programs actually scanned.

4.1 General outline of program verify

As set forth in Fig. 4.1.1, the display image files (STF files) displayed by the ladder editor are different from the executable files actually scanned by the run time engine. Therefore, even when changes have been made to a program by the ladder editor and the file has been stored, if this program is not downloaded, the two programs will remain different, and the I/O and ladder monitors will not function properly.

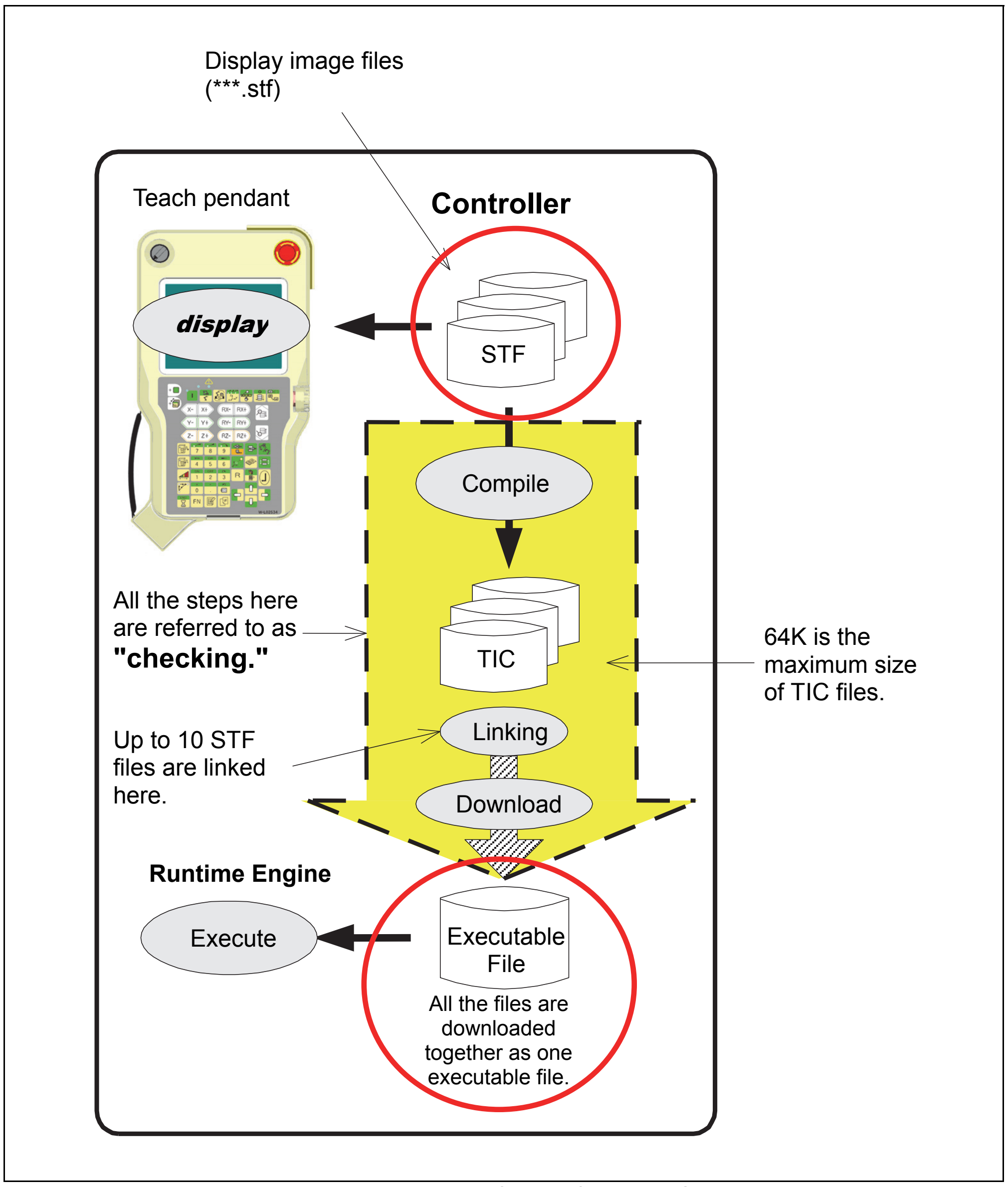

Display image files
(***.stf)
Teach pendant
Controller
display
STF
Compile
All the steps here are referred to as "checking."
TIC
64K is the maximum size of TIC files.
Up to 10 STF files are linked here.
Linking
Download
Runtime Engine
Execute
Executable File
All the files are downloaded together as one executable file.

Fig. 4.1.1 The structure of the software PLC

Thus, it is possible to verify whether the ladder programs (display image files) displayed and edited by the ladder editor are identical to the programs that the run time engine is actually being scanned. This process is known as "program verify." Programs can be verified even while PLC scanning is underway.
Programs are verified by temporarily compiling and linking (multiple) STF files targeted for verify, and then comparing them with the executable files in the run time engine.

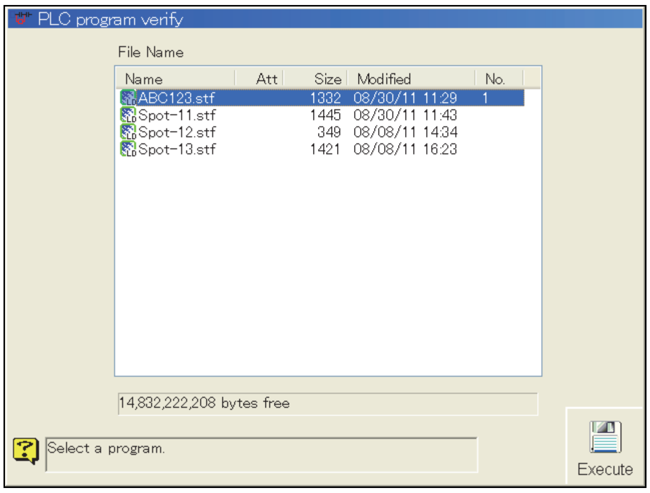
Even when a mismatch is detected during program verify, it is not possible to specify its location.

4.2 Program verify

This section describes the operations involved in selecting the ladder programs (display image files) stored in the internal memory, compiling them while specifying the execution sequence, and browsing the results of comparing them with what has already been downloaded in the run time engine.

1 **Select <Service Utilities> - [14 PLC Program Edit], and select [2 PLC program verify] from the items menu displayed.**

>> A list of the ladder programs (display image files) is now displayed as follows.


PLC program verify

File Name

Name	Att	Size	Modified	No.
ABC123.stf		1332	08/30/11 11:29	1
Spot-11.stf		1445	08/30/11 11:43	
Spot-12.stf		349	08/08/11 14:34	
Spot-13.stf		1421	08/08/11 16:23	

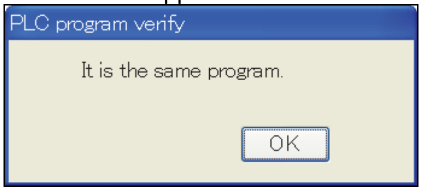
14,832,222,208 bytes free

Select a program.

2 **Select the ladder program to be collated using the up or down cursor key, and press [Enter].**
In exactly the same way as when checking (compiling) the programs, keep pressing [Enter] in the sequence in which the programs are to be run.

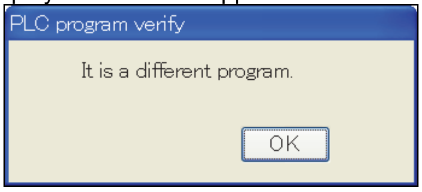
3 **After all the ladder programs have been selected, press f12 <Execute>.**

>>The procedure from compiling to linking is carried out in the same was as with checking (compiling), and a comparison is made to verify whether the generated executable files are the same as those in the run time engine.
If the results of the comparison indicate that the programs are identical, the display shown below appears.


PLC program verify

It is the same program.

If the results of the comparison indicate that the programs are not identical, the display shown below appears.


PLC program verify

It is a different program.

4 **If a compilation error is found in a selected ladder program, the program cannot be collated.**
As with checking (compiling), an error message is displayed. Make the corrections, and resume verify.

5 **Exit the menu using the [Reset/R] key.**

Chapter 5 Monitors

The monitor functions of the software PLC are described in this chapter. A number of other monitors in addition to the ladder monitor are available, and can be used for the intended objectives.

5.1 General outline of monitors

When the software PLC is in the "start" status, the contact and coil ON/OFF statuses can be displayed in real time.
A number of monitor functions such as displaying contact or coil status on the ladder diagrams or displaying specific coils are available. Use them according to the intended objective.
The monitor refresh cycle is 200msec. Therefore, contacts or coils that switch ON/OFF at high speed may not be displayed correctly.

POINT

When the software PLC is in the “disconnected” status, nothing is displayed on the monitors.

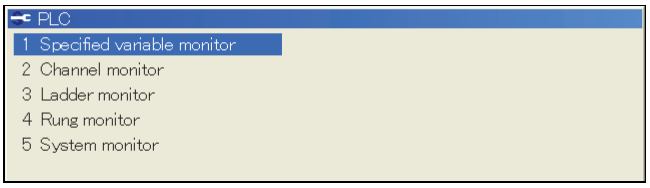
IMPORTANT

The limitation of the filename length depends on the software version. In case of V3.21 or before, several monitor functions (ladder monitor and lung monitor) do not work when the filename length of the ladder program exceeds 16 letters. Please use short name.

Service Utilities

1 **Select <Service Utilities> - [3 Monitor 1] to [6 Monitor 4], and select [32 PLC] from the menu items displayed.**

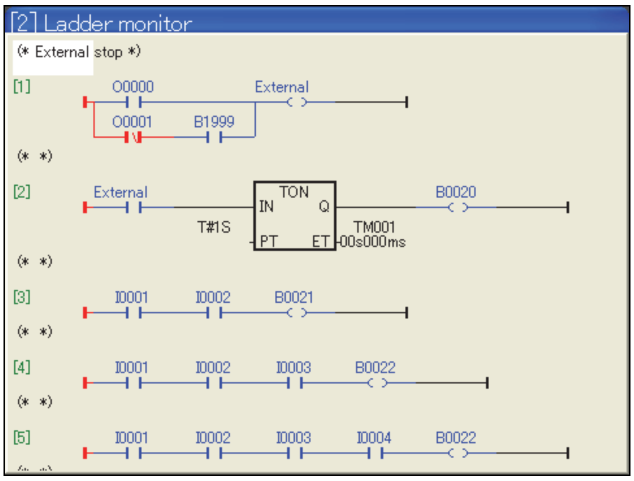
>> A menu listing the built-in PLC monitors resembling the one shown below is displayed.



PLC
1 Specified variable monitor
2 Channel monitor
3 Ladder monitor
4 Rung monitor
5 System monitor

2 **A total of five monitors are provided.**
Use [Up/Down] to select the monitor menu item to be used, and press [Enter].

>> If the ladder monitor is to be used, for example, the following display appears.


[2] Ladder monitor

A detailed description of each monitor menu item is provided in the following pages.

3 **To close the monitor, place the monitor screen in the active status using the same [CLOSE] as with the other monitors, and then press [ENABLE] and [CLOSE] together.**

5.2 Specified variable monitor

This monitor enables up to any 16 variables to be monitored simultaneously. The 16 variables can be specified as desired irrespective of type.
The monitor data is displayed in text as shown below.

1. General-purpose input, general-purpose output, physical input, physical output

I0000	**TRUE**
IO number	TRUE/FALSE

2. BOOL variables (normal variables, retained variables)

B0000	**TRUE**
BOOL number	TRUE/FALSE

3. Real variables (normal variables, retained variables)

R0001	**2345.0005**
Real variable number	Real-number display

4. DINT variables (normal variables, retained variables)
   How the third portion of the integer display is to be displayed whether in hexadecimal and binary notation can be switched by inputting the left or right key.
   When hexadecimal notation is selected

D0001	**12345**	**00003039 H**
DINT variable number	integer display	hexadecimal notation

   When binary notation is selected

D0001	**12345**	**00000000 00000000 00110000 00111001 B**
DINT variable number	integer display	binary notation

5. SINT variables (normal variables, retained variables)
   How the third portion of the short integer display is to be displayed whether in hexadecimal and binary notation can be switched by inputting the left or right key.
   When hexadecimal notation is selected

DB0001	**123**	**7B H**
SINT variable number	integer display	hexadecimal notation

   When binary notation is selected

DB0001	**12345**	**01111011 B**
SINT variable number	integer display	binary notation

6. Timer variables (normal variables, retained variables)

TM001	**00h00m12s345ms**
Timer variable number	Timer display

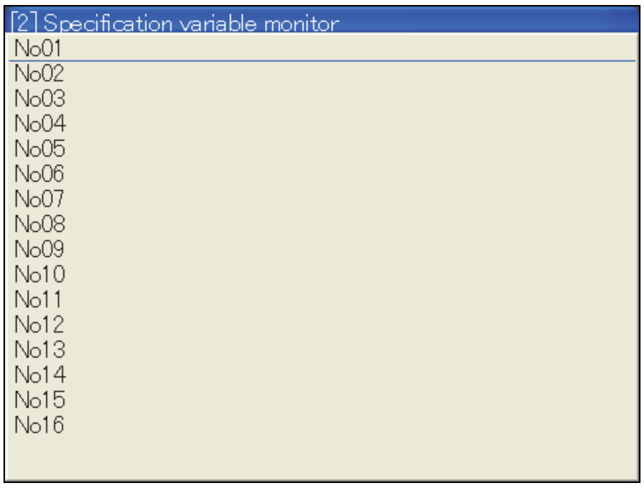
7. Character variables (normal variables, retained variables)
   What appears on the display portion of the character string display can be moved one character at a time by inputting the left or right key.

ST001	**ABCDEFGHIJKLMNOPQRSTUVWXYZ012345**	**255**
String variables	Character display	Number of characters

5.2.1 Displaying the specified variable monitor

1 **Select [1 Specification variable monitor] from the built-in PLC monitor menu.**

>> The monitor screen such as the one shown below appears.

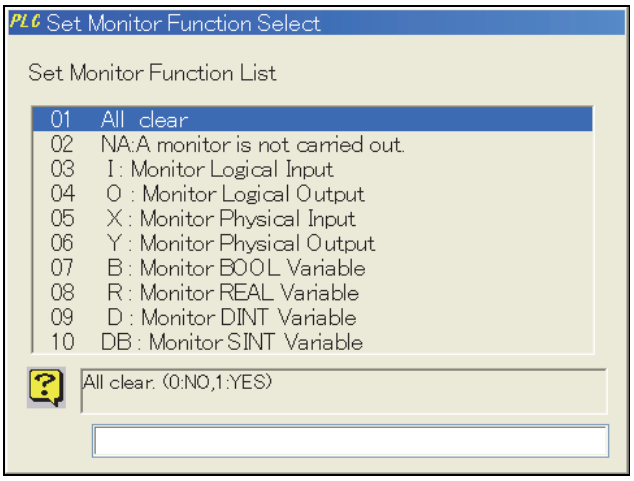


[2] Specification variable monitor

No01
No02
No03
No04
No05
No06
No07
No08
No09
No10
No11
No12
No13
No14
No15
No16

2 **Select the applicable number (such as No01) using the up or down cursor key, and press [Enter].**

>> The dialog box for selecting the variable type such as the one shown below appears on the display.
In this dialog box, input the variable to be monitored on the display.



PLC Set Monitor Function Select

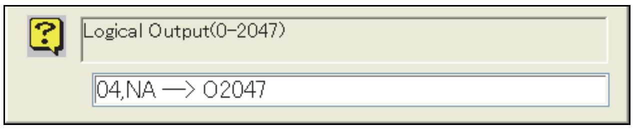
Set Monitor Function List

01 All clear
02 NA:A monitor is not carried out.
03 I : Monitor Logical Input
04 O : Monitor Logical Output
05 X : Monitor Physical Input
06 Y : Monitor Physical Output
07 B : Monitor BOOL Variable
08 R : Monitor REAL Variable
09 D : Monitor DINT Variable
10 DB : Monitor SINT Variable

All clear. (0:NO,1:YES)

Number

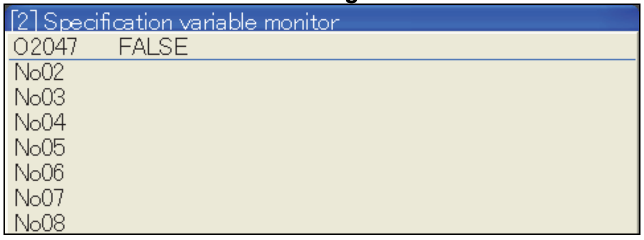
3 **If logical output 2047 is to be monitored, for example, align the cursor with "04 O: Monitor Logical output," press [Enter], input "2047" using the number keys, and finally press [Enter] again.**



Logical Output(0-2047)

04,NA —> O2047

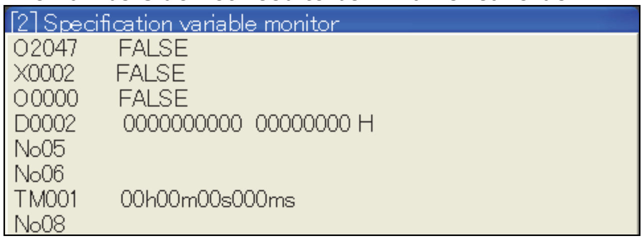
4 **The monitor screen now changes to the one shown below.**



```
[2] Specification variable monitor
O2047     FALSE
No02
No03
No04
No05
No06
No07
No08
```

Monitor No01 is replaced with logical output O2047, and the monitor data (false in this case) appears on the right.

5 **In the same way, specify up to 16 variables from No02 to No16 one after the other.**
The numbers do not need to be in numerical order.



```
[2] Specification variable monitor
O2047     FALSE
X0002     FALSE
O0000     FALSE
D0002      0000000000  00000000 H
No05
No06
TM001      00h00m00s000ms
No08
```

The monitor display settings are stored even after the power is turned off.

6 **To return a number for which the monitor display has been set to the no-monitoring status, select "02 NA: A monitor is not carried out."**

7 **To abort all the monitor display settings, select "01 All clear," and select "1" (Yes).**

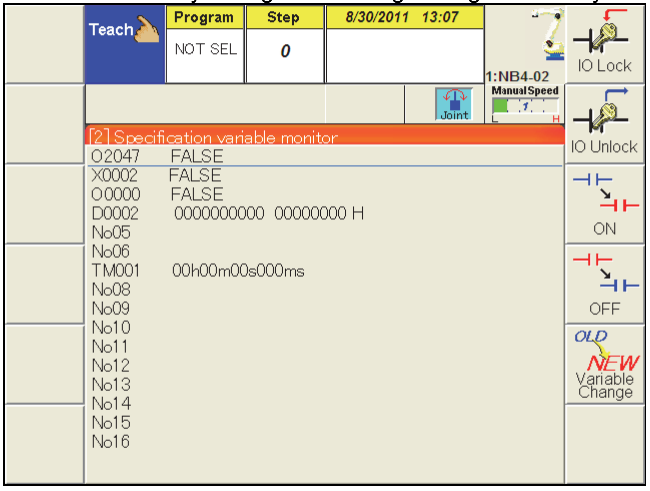
5.2.2 Forcibly changing the settings

The contents of variables can be forcibly changed on the specified variable monitor screen.
The operations involved are described below.

1 **With the monitor screen in the active status, press [EDIT].**

>> The title bar turns red to indicate that the editing status has been established, and the f key display changes as shown below.

Proceed to forcibly change the settings using these f keys.



```
Teach   Program   Step   8/30/2011  13:07
        NOT SEL   0
[2] Specification variable monitor
O2047     FALSE
X0002     FALSE
O0000     FALSE
D0002      0000000000  00000000 H
No05
No06
TM001      00h00m00s000ms
No08
No09
No10
No11
No12
No13
No14
No15
No16
```

2 **Use [Up/Down] to align the cursor with the variable whose setting is to be forcibly changed.**

IO Lock

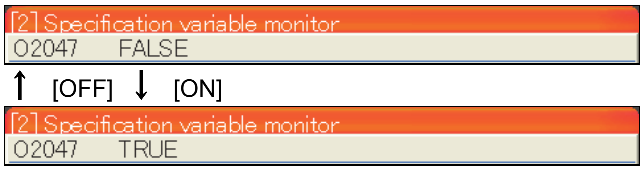
3 **To forcibly change the on/off statuses of logical inputs/outputs and physical inputs/outputs, first press f7 <IO Lock>.**

>> The selected variable is temporarily disconnected from the PLC engine. This does not mean that scanning stops.

ON

OFF

4 **To forcibly set to ON (TRUE), press f9 <ON>; conversely, to forcibly set to OFF (FALSE), press f10 <OFF>.**


[2] Specification variable monitor
O2047 FALSE

↑ [OFF] ↓ [ON]

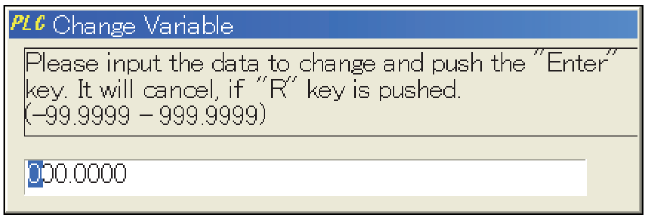
[2] Specification variable monitor
O2047 TRUE

Even if the monitor screen is now closed with no other operations performed, the forced setting status (I/O lock status) remains unchanged even if other PLC monitor screens are opened.

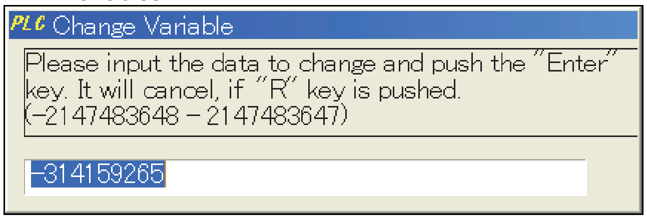
OLD NEW Variable Change

5 **In the same way, the settings of the Real variables, DINT variables, SINT variables, Timer variables and String variables can be forcibly changed using f11 <Variable Change>.**
With each change, the message shown below appears. Following the guide message displayed, input a number and then press [Enter].
For forcibly changing the String variables, the soft keyboard is displayed so use its keys to input the new character string, and press f12 [Complete] key.

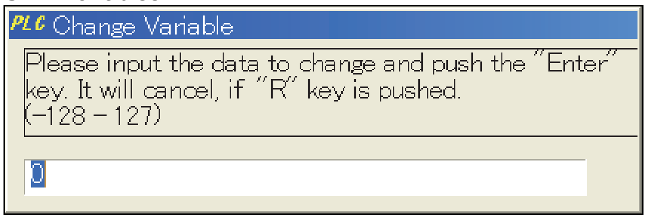
Real variables


PLC Change Variable
Please input the data to change and push the "Enter" key. It will cancel, if "R" key is pushed.
(-99.9999 - 999.9999)
000.0000

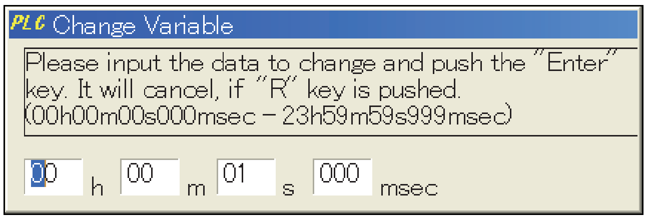
DINT variables


PLC Change Variable
Please input the data to change and push the "Enter" key. It will cancel, if "R" key is pushed.
(-2147483648 - 2147483647)
-314159265

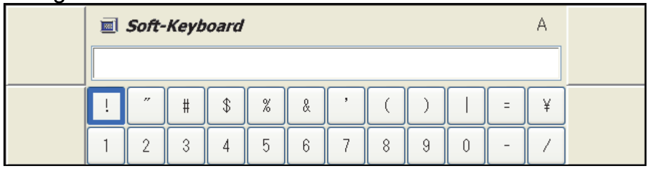
SINT variables


PLC Change Variable
Please input the data to change and push the "Enter" key. It will cancel, if "R" key is pushed.
(-128 - 127)
0

Timer variables


PLC Change Variable
Please input the data to change and push the "Enter" key. It will cancel, if "R" key is pushed.
(00h00m00s000msec - 23h59m59s999msec)
00 h 00 m 01 s 000 msec

String variables



R

6 **To cancel the input, press [Reset/R].**

IO Unlock

7 **After having forcibly changed the settings, press f8 <IO Unlock>.**

>> The temporarily disconnected status is released, and the regular scanning status is restored.
Do NOT forget to perform this operation.

R

8 **The original monitor status screen is restored by pressing [Reset/R].**

5.2.3 Initializing variables in a batch

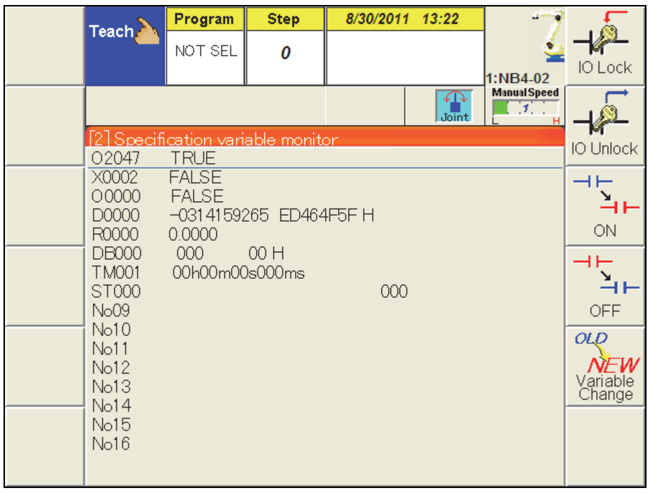
On the Specified Variable Monitor screen, the contents of the variables (power failure holding devices) can be initialized in a batch. The following section describes the initializing procedure.

1 **Activate the Monitor screen, and then press [Edit].**

+

2 **Press [Enable].**

>> The f key display changes as shown below.



```
[2]Specification variable monitor
O2047    TRUE
X0002    FALSE
O0000    FALSE
D0000    -0314159265  ED464F5F H
R0000    0.0000
DB000     000       00 H
TM001    00h00m00s000ms
ST000                                000
No09
No10
No11
No12
No13
No14
No15
No16
```

Variable All Clear

3 **Press f11 [Batch variable initialization].**

>> The variables (power failure holding devices) are initialized.

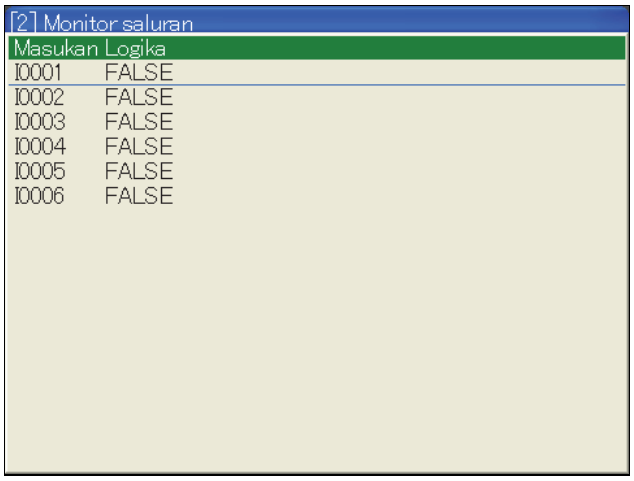
5.3 Channel monitor

The channel monitor displays a list of only the variables that are being used by the program that is currently being executed by the software PLC. Each of the “Logical input”, "Logical output”, “Physical input”, “Physical output”, “BOOL variable”, “SINT variable”, “DINT variable”, “Real variable”, “Timer variable” and "String variable" are displayed.
Furthermore, the monitor data is displayed in the same format text as the specification variable monitor.

5.3.1 Displaying the channel monitor

1 **Select [2 Channel monitor] from the monitor menu items.**

>> The monitor screen such as the one shown below appears. (The figure below shows the whole screen display.)
The statuses of all the "logical input" variables used by the program now being run by the PLC are listed on the display.

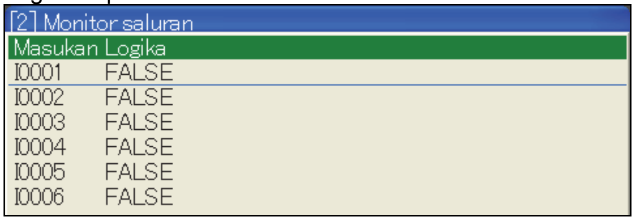


```
[2] Monitor saluran
Masukan Logika
I0001     FALSE
I0002     FALSE
I0003     FALSE
I0004     FALSE
I0005     FALSE
I0006     FALSE
```

2 **Press [ENABLE] + [Right/Left].**

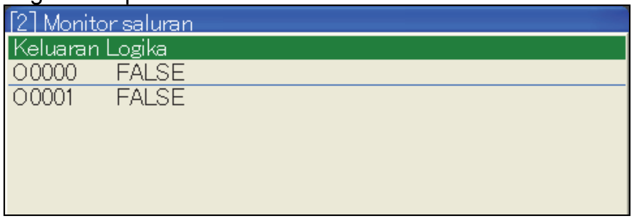
>> The monitor screen is switched in the following sequence: "Logical Input"–"Logical Output"–"Physical Input"–"Physical Output"–"BOOL Variable"–"Real variable"–" DINT Variable"–"SINT Variable"–"TIME Variable"–"STRING Variable."
On each and every screen, the statuses of only those variables used by the program being run by the PLC are listed on the display.
If no variables have been recorded for a certain type, the display will be blank.

Logical Input



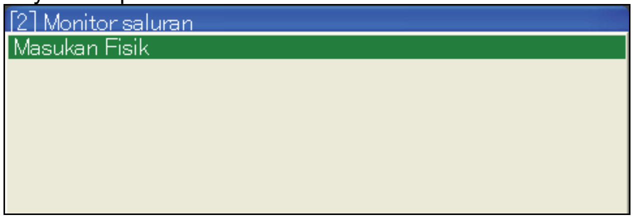
```
[2] Monitor saluran
Masukan Logika
I0001     FALSE
I0002     FALSE
I0003     FALSE
I0004     FALSE
I0005     FALSE
I0006     FALSE
```

Logical Output



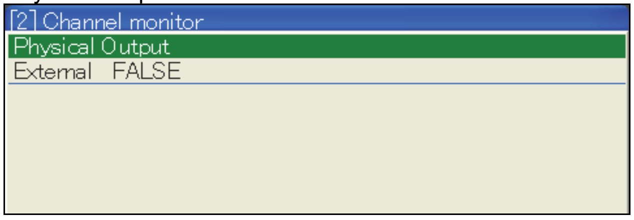
```
[2] Monitor saluran
Keluaran Logika
O0000     FALSE
O0001     FALSE
```

Physical Input



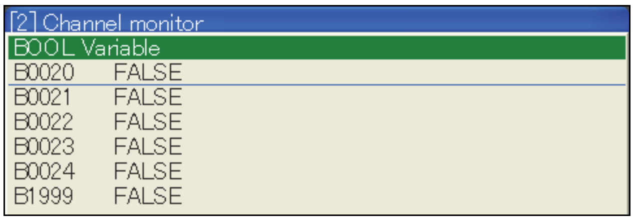
```
[2] Monitor saluran
Masukan Fisik
```

Physical Output



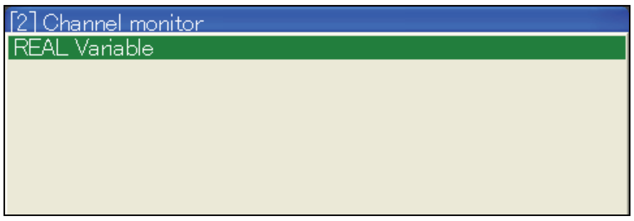
```
[2] Channel monitor
Physical Output
External   FALSE
```

BOOL Variable



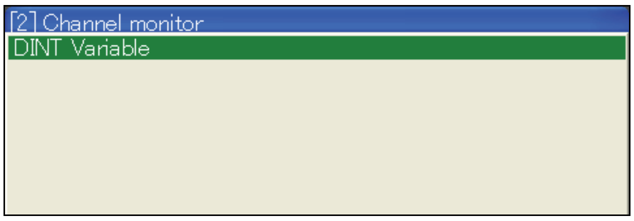
```
[2] Channel monitor
BOOL Variable
B0020      FALSE
B0021      FALSE
B0022      FALSE
B0023      FALSE
B0024      FALSE
B1999      FALSE
```

Real Variable



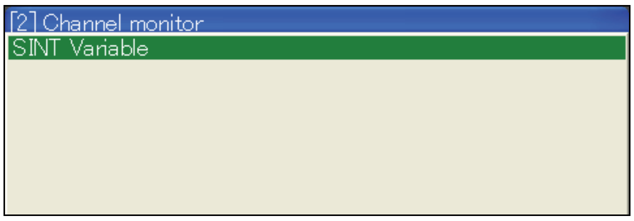
```
[2] Channel monitor
REAL Variable
```

DINT Variable



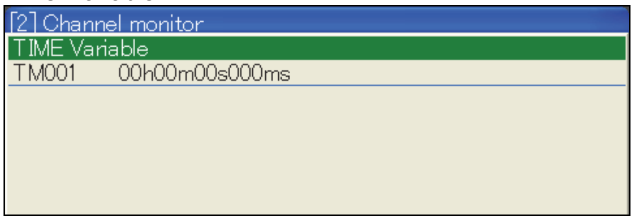
```
[2] Channel monitor
DINT Variable
```

SINT Variable



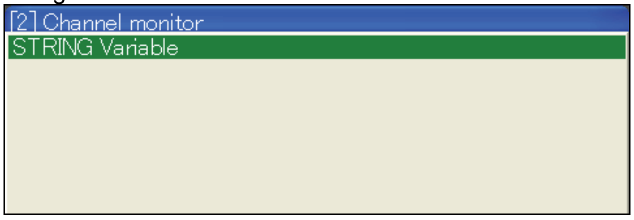
```
[2] Channel monitor
SINT Variable
```

Timer variable

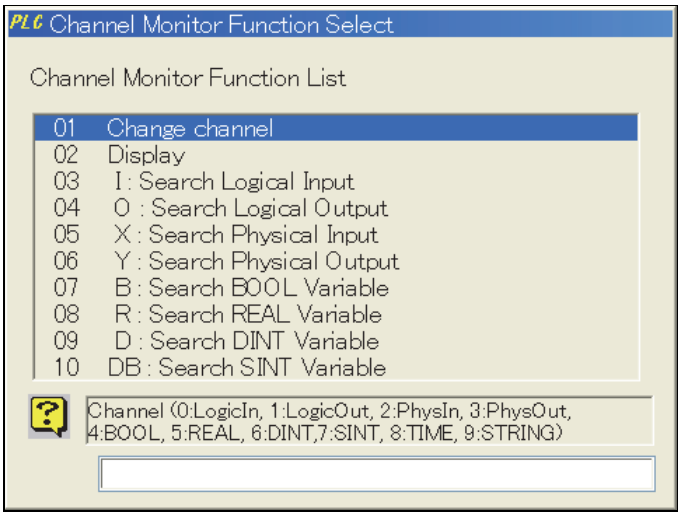


```
[2] Channel monitor
TIME Variable
TM001      00h00m00s000ms
```

String variable


[2] Channel monitor
STRING Variable

3 **Whether particular variables are being used in the program now being run can be searched, and operation can also be made to jump to the monitor screen concerned.**
On the display screen of the channel concerned, press [Enter].
>> The dialog box such as the one shown below now appears.


Channel Monitor Function Select

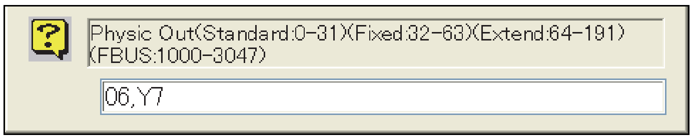
Channel Monitor Function List

```
01   Change channel
02   Display
03   I : Search Logical Input
04   O : Search Logical Output
05   X : Search Physical Input
06   Y : Search Physical Output
07   B : Search BOOL Variable
08   R : Search REAL Variable
09   D : Search DINT Variable
10   DB : Search SINT Variable
```

Channel (0:LogicIn, 1:LogicOut, 2:PhysIn, 3:PhysOut, 4:BOOL, 5:REAL, 6:DINT,7:SINT, 8:TIME, 9:STRING)

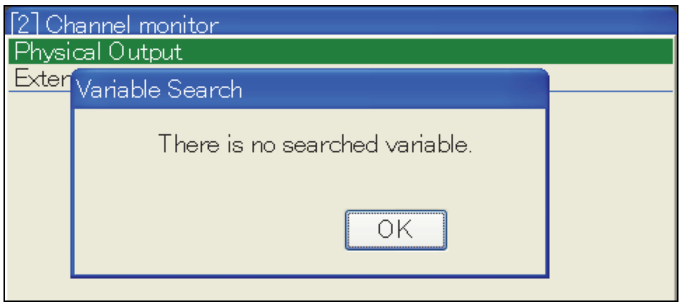
Number

4 **Specify here the variables to be searched (= variable to be monitored on the display).**
If physical output Y0007 is to be searched (monitored on the display), select "06 Y: Search Physical Output," and press [7] and [Enter].


Physic Out(Standard:0-31)(Fixed:32-63)(Extend:64-191)(FBUS:1000-3047)

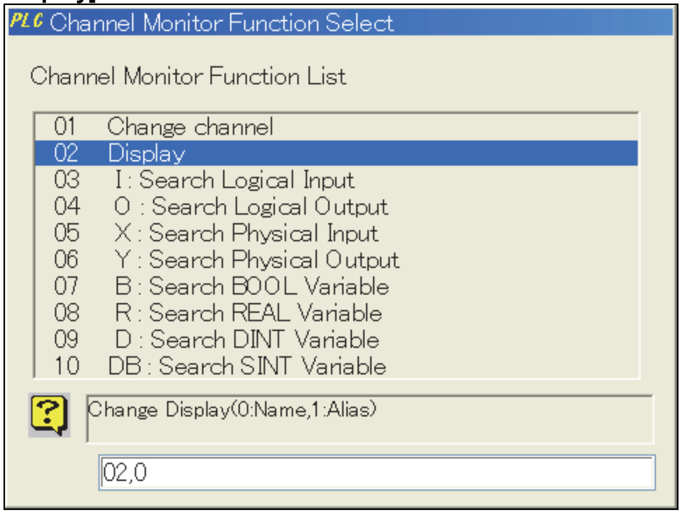
06,Y7

5 **The monitor screen changes to "Physical Output," and the cursor is positioned at the monitor display line for the physical output Y0007 which was specified.**
If physical output Y0007 is not used inside the program, the following message is displayed.


[2] Channel monitor
Physical Output
Exter
Variable Search
There is no searched variable.
OK

6 **Press [Enter] to clear the message.**

7 **On the channel monitor, aliases can be shown on the monitor displays. To do this, press [Enter] on any channel display screen, and select [02 Display].**



PLC Channel Monitor Function Select

Channel Monitor Function List

01	Change channel
02	Display
03	I : Search Logical Input
04	O : Search Logical Output
05	X : Search Physical Input
06	Y : Search Physical Output
07	B : Search BOOL Variable
08	R : Search REAL Variable
09	D : Search DINT Variable
10	DB : Search SINT Variable

Change Display(0:Name,1:Alias)

02,0

8 **For displaying the names of the variables, press [0] and [Enter]; for displaying the aliases, press [1] and [Enter].**

>> The displayed channel is displayed by the names of the variables (or aliases). The variables are arranged in alphabetical order so the display sequence for the variables names differs from the display sequence for aliases.

5.3.2 Forcibly changing the settings

On the channel monitor screen, changes can be forcibly made to the contents of the variables.
The operation method is the same as for the specified variable monitor.

5.3.3 Initializing variables in a batch

On the Channel Monitor screen, the contents of the variables (power failure holding devices) can be initialized in a batch.
The initializing procedure is the same as that for the specified variable monitor.

5.4 Ladder monitor

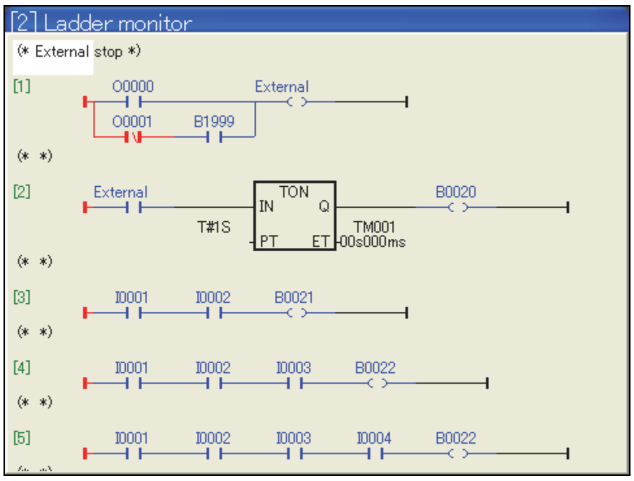
The program now being run by the PLC can be monitored on the display in the ladder display status.

5.4.1 Displaying the ladder monitor

1 **Select [3 Ladder monitor] from the monitor menu items.**

>> The monitor screen such as the one shown below appears. (The figure below shows the whole screen display.)

The program now being run by the PLC is displayed from its start.
The area with the white background is the cursor position. Move across the display using [Cursor keys].



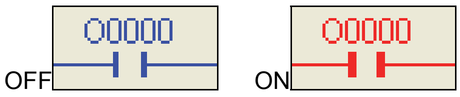
[2] Ladder monitor
(* External stop *)
[1] 00000 External 00001 B1999
(* *)
[2] External TON IN Q T#1S PT ET TM001 00s000ms B0020
(* *)
[3] I0001 I0002 B0021
(* *)
[4] I0001 I0002 I0003 B0022
(* *)
[5] I0001 I0002 I0003 I0004 B0022

Linking and executing multiple programs

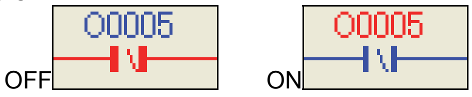
1 program (display image file; *.stf) can be displayed for 1 ladder monitor. If multiple programs are being linked and executed, a dialog box for selecting the program for monitor display appears. Select the programs to be displayed.**
Up to 4 monitors can be displayed (opened) simultaneously. To monitor multiple programs at the same time, display the ladder monitor in a different monitor window, and select a different program in that window.

The contact and coil ON/OFF, timer and other current values are displayed as shown below.

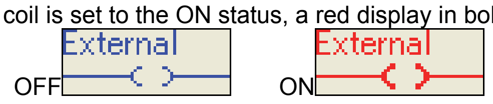
When the A (normally open) contact is set to the ON status, a red display in bold lines appears.



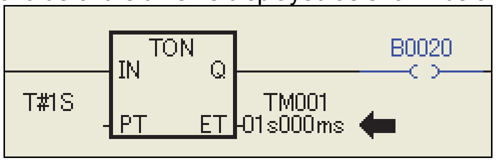
When the B (normally open) contact is set to the OFF status, a red display in bold lines appears.



When the coil is set to the ON status, a red display in bold lines appears.



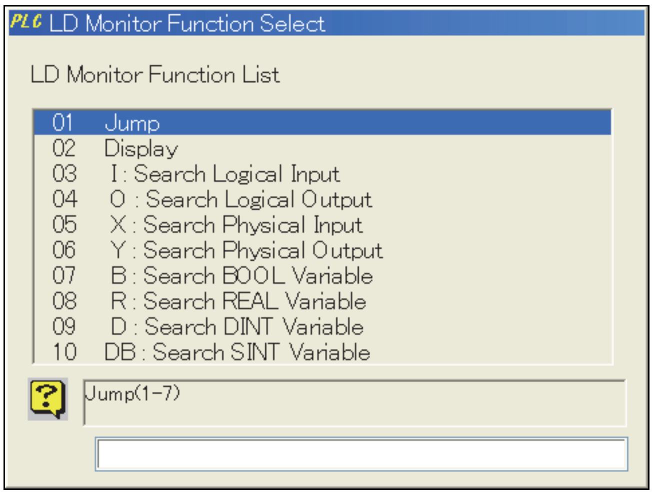
The current value of the timer is displayed as shown below. (Area indicated by arrow)



In the case of the example given above, whether or not the time-up setting has been reached is connected to output Q, and this corresponds to the ON/OFF status of BOOL variable B0020.

2 **On the ladder monitor screen, a rung number or particular variable can be specified so that a jump can be made to where it is displayed.**
Press [Enter] at any cursor position.

>> The dialog box such as the one shown below now appears.



LD Monitor Function Select

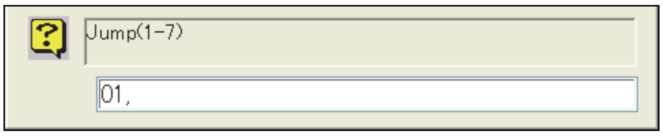
LD Monitor Function List

```
01   Jump
02   Display
03    I : Search Logical Input
04    O : Search Logical Output
05    X : Search Physical Input
06    Y : Search Physical Output
07    B : Search BOOL Variable
08    R : Search REAL Variable
09    D : Search DINT Variable
10   DB : Search SINT Variable
```

Jump(1-7)

Number

3 **To specify a rung number and jump to the rung concerned, select "01 Jump," and press [Enter].**
Then input the number of the rung to be displayed, and press [Enter].



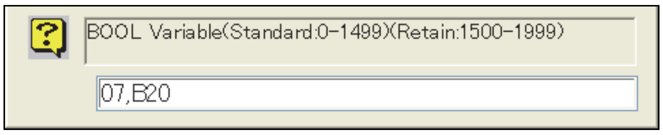
Jump(1-7)

01,

>> The ladder monitor screen appears, and the cursor is positioned at the beginning of the specified rung (closed circuit).

Number

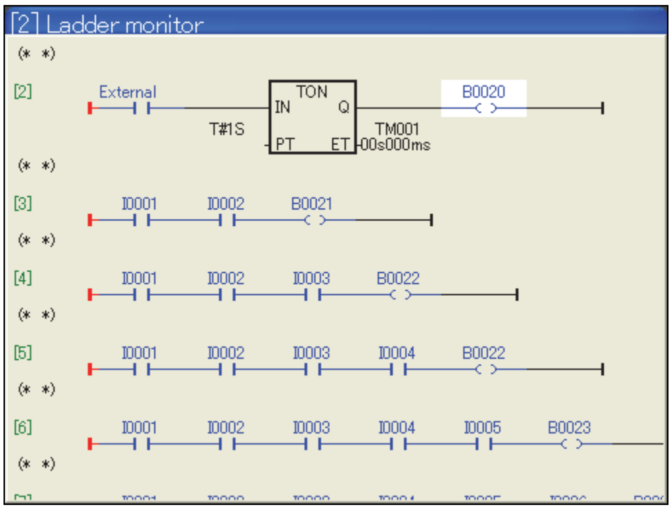
4 **When searching and jumping to a variable,**
For example, if physical output B0020 is to be searched (and a jump made to B0020), select “07 B; BOOL Variable Search”, and enter “value” + [Enter].



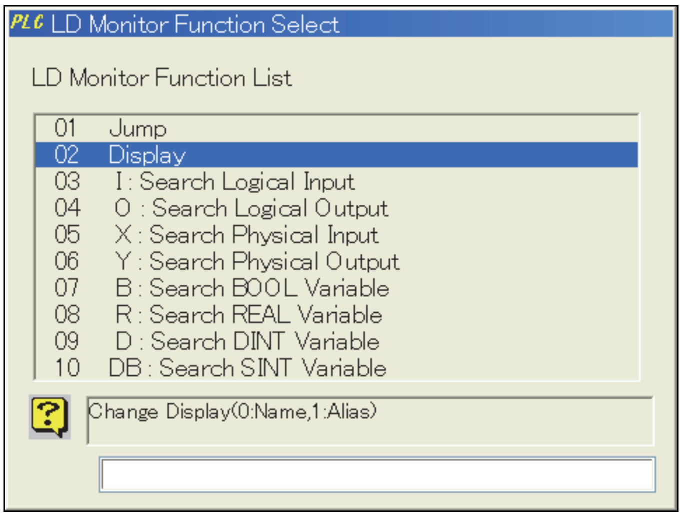
BOOL Variable(Standard:0-1499)(Retain:1500-1999)

07,B20

>> The ladder monitor screen appears, and the cursor is positioned at the specified variable.



5 **With the ladder monitor, aliases can be shown on the monitor displays.**
To do this, press [Enter] on the ladder monitor screen, and select [02 Display].



PLC LD Monitor Function Select

LD Monitor Function List

01	Jump
02	Display
03	I : Search Logical Input
04	O : Search Logical Output
05	X : Search Physical Input
06	Y : Search Physical Output
07	B : Search BOOL Variable
08	R : Search REAL Variable
09	D : Search DINT Variable
10	DB : Search SINT Variable

Change Display(0:Name,1:Alias)

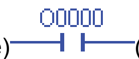
6 **For displaying the names of the variables, press [0] and [Enter];**
for displaying the aliases, press [1] and [Enter].

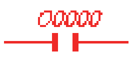
7 **Press [Shift] + [Up] / [Down] keys.**
>>Downward / Upward search can restart.
If the search operation has not been executed, the search will not restart.

8 **Press [Shift] + [Left] / [Right] keys.**
>>Downward / Upward search can restart. (Only coils)
If the search operation has not been executed, the search will not restart.

5.4.2 Forcibly changing the settings

On the ladder monitor screen, changes can be forcibly made to the contents of the variables.
The operation method is the same as for the specified variable monitor and channel monitor.
The text is set in italic characters while Forcibly changing.

(Normal state)

(Forcibly changing state) 

5.4.3 Initializing variables in a batch

On the Ladder Monitor screen, the contents of the variables (power failure holding devices) can be initialized in a batch.

5.5 Rung monitor

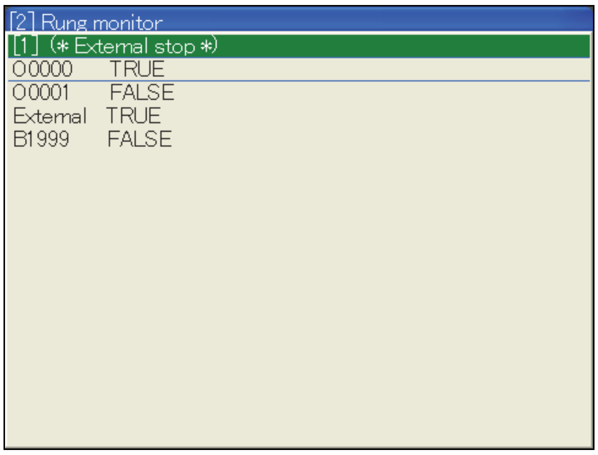
With the rung monitor, all the variables used for each rung can be monitored on the display for each of the rungs (closed circuits).
The monitor data is displayed with text that takes the same form as the text used by the specified variable monitor.

5.5.1 Displaying the rung monitor

1 **Select [4 Rung monitor] from the monitor menu items.**

>> The monitor screen such as the one shown below appears. (The figure below shows the whole screen display.)

All the variables used in the first rung of the program now being run by the PLC are displayed.
The rung number and rung comment are displayed at the top of the screen.



[2] Rung monitor
[1] (* External stop *)

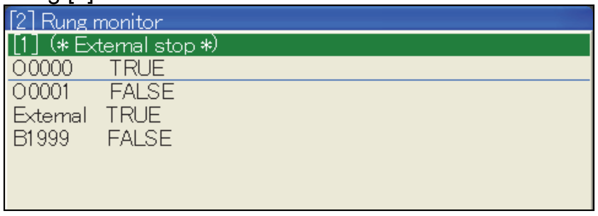
```
O0000     TRUE
O0001     FALSE
External  TRUE
B1999     FALSE
```

2 **Press [ENABLE] and [Right/Left].**

>> The monitor screen is switched in sequence starting from the first rung.

On each and every screen, the statuses of all the variables used in the rung concerned are listed on the display.

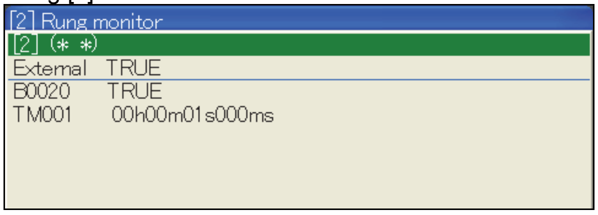
Rung [1]



[2] Rung monitor
[1] (* External stop *)

```
O0000     TRUE
O0001     FALSE
External  TRUE
B1999     FALSE
```

Rung [2]



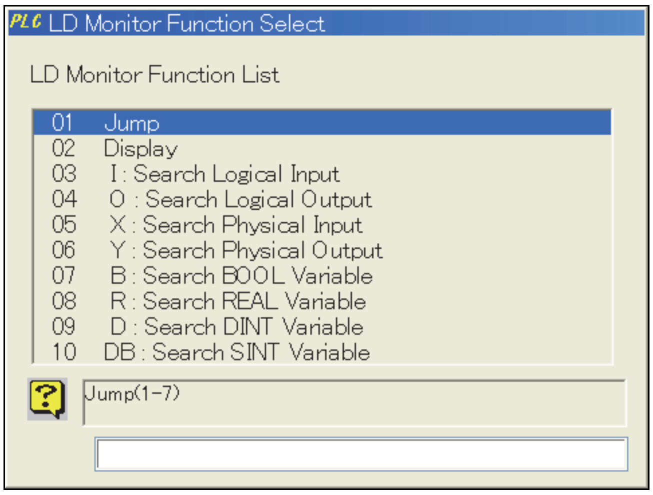
[2] Rung monitor
[2] (* *)

```
External  TRUE
B0020     TRUE
TM001     00h00m01s000ms
```

Rung [3]
. . .
Rung [7]

The display does not return to the first rung from the last rung.

3 **Whether particular variables are being used in the program now being run can be searched, and operation can also be made to jump to the monitor screen concerned.**
Press [Enter] on any rung monitor screen.
>> The dialog box such as the one shown below now appears.


PLC LD Monitor Function Select

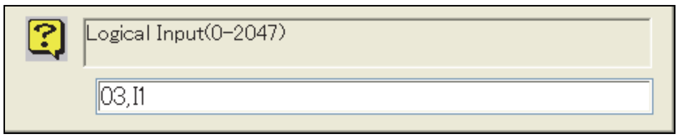
LD Monitor Function List

```
01   Jump
02   Display
03   I : Search Logical Input
04   O : Search Logical Output
05   X : Search Physical Input
06   Y : Search Physical Output
07   B : Search BOOL Variable
08   R : Search REAL Variable
09   D : Search DINT Variable
10   DB : Search SINT Variable
```

Jump(1-7)

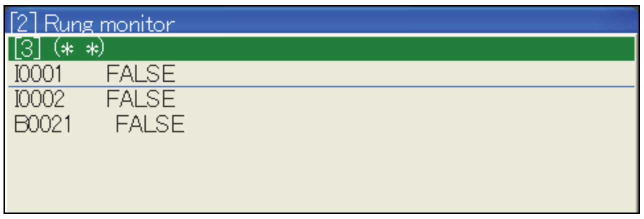
Number

4 **Specify here the variables to be searched (= variable to be monitored on the display).**
To search logical input I0001 (to monitor it on the display), select "03 I: Search Logical Input," and press [1] and [Enter].


Logical Input(0-2047)

03,I1

5 **A search is performed in the backward direction starting from the rung now displayed.**
The screen is switched to the display of the rung found first, and the cursor is positioned at the specified I0001.


[2] Rung monitor
[3] (* *)

```
I0001     FALSE
I0002     FALSE
B0021     FALSE
```

6 **With the rung monitor, aliases can be shown on the monitor displays.**
The operation method is the same as for the channel and other monitors.

5.5.2 Forcibly changing the settings

On the rung monitor screen, changes can be forcibly made to the contents of the variables.
The operation method is the same as for the specified variable monitor, channel monitor and ladder monitor.

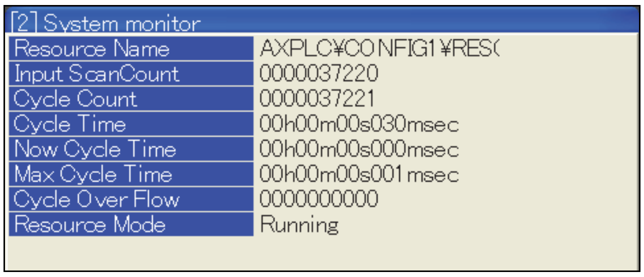
5.5.3 Initializing variables in a batch

On the Rung Monitor screen, the contents of the variables (power failure holding devices) can be initialized in a batch.
The initializing procedure is the same as that for the specified variable monitor, the channel monitor, and the ladder monitor.

5.6 System monitor

The system monitor refers to the monitor screen on which the software PLC statuses are diagnosed.

1 **Select [5 System monitor] from the monitor menu items.**
>> The monitor screen such as the one shown below appears.


[2] System monitor

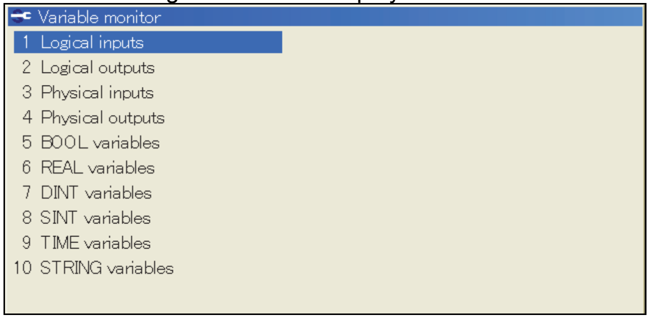
Resource Name	AXPLC¥CO NFIG1 ¥RES(
Input ScanCount	0000037220
Cycle Count	0000037221
Cycle Time	00h00m00s030msec
Now Cycle Time	00h00m00s000msec
Max Cycle Time	00h00m00s001 msec
Cycle Over Flow	0000000000
Resource Mode	Running

Table 5.6.1 System monitor display item

Display item	Details	
Number of scans input	This is the number of scans input to date since the previous program was downloaded to the run time engine. It is displayed in an unsigned 32-bit format, and when the maximum value is reached, the number returns to zero.	
Executed cycle count	This is the number of cycles executed to date since the previous program was downloaded to the run time engine. It is displayed in an unsigned 32-bit format, and when the maximum value is reached, the number returns to zero.	
Set cycle time value	This is the PLC scanning interval (time from scan start to the next scan start) which has been set. The factory setting is 20 ms.	
Execution cycle time value	This is the time actually taken for scanning.	
Maximum cycle time value	This is the maximum time actually taken for scanning.	
Number of exceeded cycle times	This value represents the total number of times that the time actually taken for scanning has exceeded the "set cycle time value."	
Resource execution mode	This indicates the execution mode of the run time engine.	
	No resources	No resources (programs) have been downloaded.
	Execution wait	The execution of the resources (programs) has been stopped.
	In Process	A resource (program) is now being executed.
	Break point	The run time engine has stopped at a break point during execution. (Break points are possible only with the ISaGRAF Workbench.)
	Fault	A fatal error has occurred.

5.7 Variables monitor

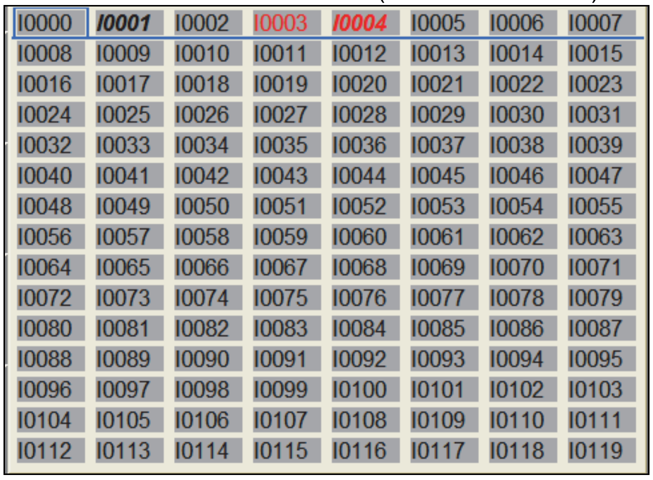
1 **Choose <Service Utilities> [Monitor] [32 PLC] [6 Variable monitor]**
>> The following menu will be displayed.


Variable monitor
1 Logical inputs
2 Logical outputs
3 Physical inputs
4 Physical outputs
5 BOOL variables
6 REAL variables
7 DINT variables
8 SINT variables
9 TIME variables
10 STRING variables

5.7.1 Logical inputs monitor

1 **Choose [1 Logical inputs] in the variables monitor menu.**
>> The following monitor will be displayed. (This example is full screen mode)

This monitor displays the condition of the all "Logical input signals".
The conditions of *I0000* to *I2047* (or *I0001* to *I2048*) are displayed.



I0000	***I0001***	I0002	I0003	***I0004***	I0005	I0006	I0007
I0008	I0009	I0010	I0011	I0012	I0013	I0014	I0015
I0016	I0017	I0018	I0019	I0020	I0021	I0022	I0023
I0024	I0025	I0026	I0027	I0028	I0029	I0030	I0031
I0032	I0033	I0034	I0035	I0036	I0037	I0038	I0039
I0040	I0041	I0042	I0043	I0044	I0045	I0046	I0047
I0048	I0049	I0050	I0051	I0052	I0053	I0054	I0055
I0056	I0057	I0058	I0059	I0060	I0061	I0062	I0063
I0064	I0065	I0066	I0067	I0068	I0069	I0070	I0071
I0072	I0073	I0074	I0075	I0076	I0077	I0078	I0079
I0080	I0081	I0082	I0083	I0084	I0085	I0086	I0087
I0088	I0089	I0090	I0091	I0092	I0093	I0094	I0095
I0096	I0097	I0098	I0099	I0100	I0101	I0102	I0103
I0104	I0105	I0106	I0107	I0108	I0109	I0110	I0111
I0112	I0113	I0114	I0115	I0116	I0117	I0118	I0119

The display style of the letter will change depending on the condition of the signal.

Signal condition OFF	Background color Gray	Signal condition ON	Background color Yellow
Not used in the PLC program	Normal letters I0002	Used in the PLC program	***Bold letters*** ***I0001***
IO Lock : OFF	Black letters	IO Lock : ON	Red letters

POINT

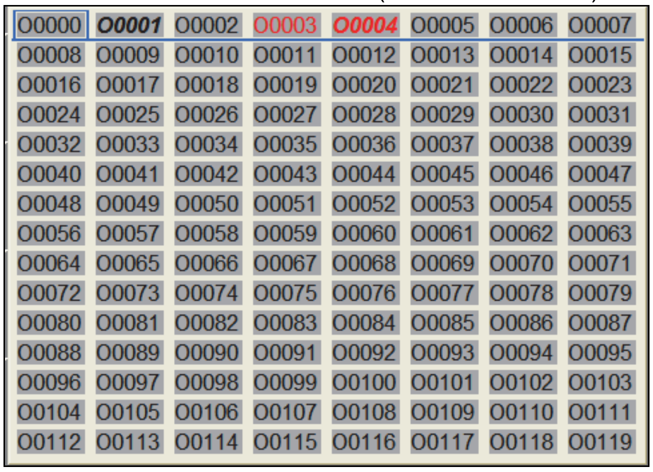
The displayed numbers depends on the setting of <Constant Setting> [1 Control Constants] [6 Built-in PLC] "PLC Logical Input / Output relays".

(0 – 2047): *I0000* to *I2047* are displayed in this monitor.
(1 – 2048): *I0001* to *I2048* are displayed in this monitor.

5.7.2 Logical outputs monitor

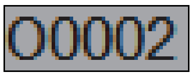
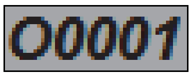
1 **Choose [2 Logical outputs] in the variables monitor menu.**
>> The following monitor will be displayed. (This example is full screen mode)

This monitor displays the condition of the all "Logical output signals".
The conditions of ***O0000*** to ***O2047*** (or ***O0001*** to ***O2048***) are displayed.



O0000	**O0001**	O0002	O0003	***O0004***	O0005	O0006	O0007
O0008	O0009	O0010	O0011	O0012	O0013	O0014	O0015
O0016	O0017	O0018	O0019	O0020	O0021	O0022	O0023
O0024	O0025	O0026	O0027	O0028	O0029	O0030	O0031
O0032	O0033	O0034	O0035	O0036	O0037	O0038	O0039
O0040	O0041	O0042	O0043	O0044	O0045	O0046	O0047
O0048	O0049	O0050	O0051	O0052	O0053	O0054	O0055
O0056	O0057	O0058	O0059	O0060	O0061	O0062	O0063
O0064	O0065	O0066	O0067	O0068	O0069	O0070	O0071
O0072	O0073	O0074	O0075	O0076	O0077	O0078	O0079
O0080	O0081	O0082	O0083	O0084	O0085	O0086	O0087
O0088	O0089	O0090	O0091	O0092	O0093	O0094	O0095
O0096	O0097	O0098	O0099	O0100	O0101	O0102	O0103
O0104	O0105	O0106	O0107	O0108	O0109	O0110	O0111
O0112	O0113	O0114	O0115	O0116	O0117	O0118	O0119

The display style of the letter will change depending on the condition of the signal.

Signal condition OFF	Background color Gray	Signal condition ON	Background color Yellow
Not used in the PLC program	Normal letters  O0002	Used in the PLC program	**Bold letters**  ***O0001***
IO Lock : OFF	Black letters	IO Lock : ON	Red letters

POINT

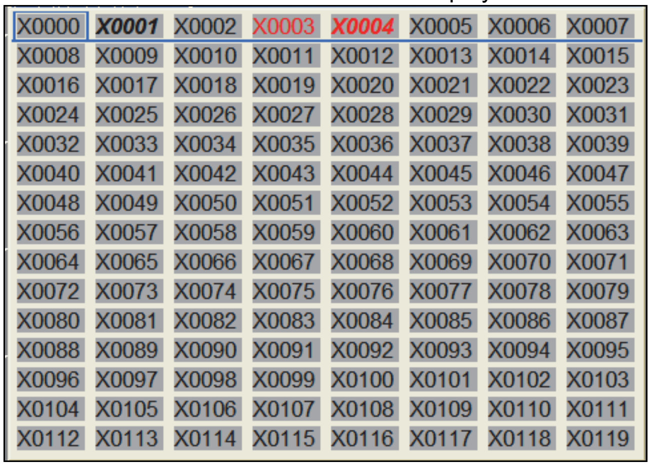
The displayed numbers depends on the setting of <Constant Setting> [1 Control Constants] [6 Built-in PLC] "PLC Logical Input / Output relays".

`(0 – 2047):` ***O0000*** to ***O2047*** are displayed in this monitor.
`(1 – 2048):` ***O0001*** to ***O2048*** are displayed in this monitor.

5.7.3 Physical inputs monitor

1 **Choose [3 Physical inputs] in the variables monitor menu.**
>> The following monitor will be displayed. (This example is full screen mode)

This monitor displays the condition of the all “Physical input signals”.
The conditions of `X0000` to `X3047` are displayed.



X0000	**X0001**	X0002	X0003	**X0004**	X0005	X0006	X0007
X0008	X0009	X0010	X0011	X0012	X0013	X0014	X0015
X0016	X0017	X0018	X0019	X0020	X0021	X0022	X0023
X0024	X0025	X0026	X0027	X0028	X0029	X0030	X0031
X0032	X0033	X0034	X0035	X0036	X0037	X0038	X0039
X0040	X0041	X0042	X0043	X0044	X0045	X0046	X0047
X0048	X0049	X0050	X0051	X0052	X0053	X0054	X0055
X0056	X0057	X0058	X0059	X0060	X0061	X0062	X0063
X0064	X0065	X0066	X0067	X0068	X0069	X0070	X0071
X0072	X0073	X0074	X0075	X0076	X0077	X0078	X0079
X0080	X0081	X0082	X0083	X0084	X0085	X0086	X0087
X0088	X0089	X0090	X0091	X0092	X0093	X0094	X0095
X0096	X0097	X0098	X0099	X0100	X0101	X0102	X0103
X0104	X0105	X0106	X0107	X0108	X0109	X0110	X0111
X0112	X0113	X0114	X0115	X0116	X0117	X0118	X0119

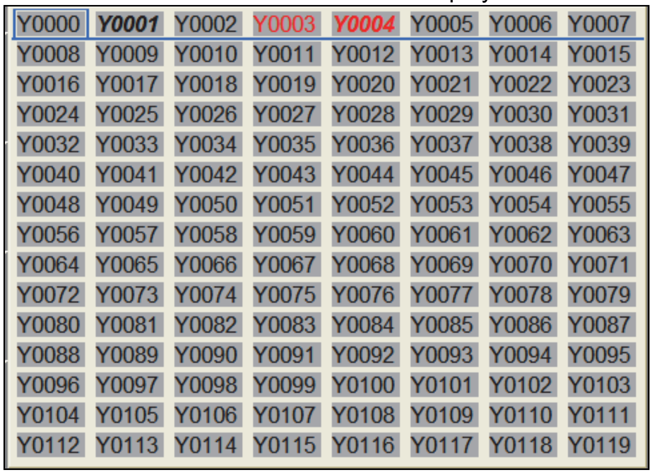
The display style of the letter will change depending on the condition of the signal.

Signal condition OFF	Background color Gray	Signal condition ON	Background color Yellow
Not used in the PLC program	Normal letters X0002	Used in the PLC program	**Bold letters** ***X0001***
IO Lock : OFF	Black letters	IO Lock : ON	Red letters

5.7.4 Physical outputs monitor

1 **Choose [4 Physical outputs] in the variables monitor menu.**
>> The following monitor will be displayed. (This example is full screen mode)

This monitor displays the condition of the all “Physical output signals”.
The conditions of *Y0000* to *Y3047* are displayed.



Y0000	**Y0001**	Y0002	Y0003	**Y0004**	Y0005	Y0006	Y0007
Y0008	Y0009	Y0010	Y0011	Y0012	Y0013	Y0014	Y0015
Y0016	Y0017	Y0018	Y0019	Y0020	Y0021	Y0022	Y0023
Y0024	Y0025	Y0026	Y0027	Y0028	Y0029	Y0030	Y0031
Y0032	Y0033	Y0034	Y0035	Y0036	Y0037	Y0038	Y0039
Y0040	Y0041	Y0042	Y0043	Y0044	Y0045	Y0046	Y0047
Y0048	Y0049	Y0050	Y0051	Y0052	Y0053	Y0054	Y0055
Y0056	Y0057	Y0058	Y0059	Y0060	Y0061	Y0062	Y0063
Y0064	Y0065	Y0066	Y0067	Y0068	Y0069	Y0070	Y0071
Y0072	Y0073	Y0074	Y0075	Y0076	Y0077	Y0078	Y0079
Y0080	Y0081	Y0082	Y0083	Y0084	Y0085	Y0086	Y0087
Y0088	Y0089	Y0090	Y0091	Y0092	Y0093	Y0094	Y0095
Y0096	Y0097	Y0098	Y0099	Y0100	Y0101	Y0102	Y0103
Y0104	Y0105	Y0106	Y0107	Y0108	Y0109	Y0110	Y0111
Y0112	Y0113	Y0114	Y0115	Y0116	Y0117	Y0118	Y0119

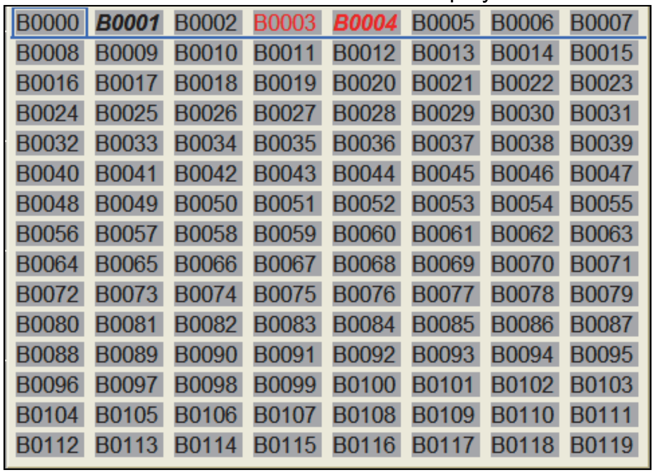
The display style of the letter will change depending on the condition of the signal.

Signal condition OFF	Background color Gray	Signal condition ON	Background color Yellow
Not used in the PLC program	Normal letters Y0002	Used in the PLC program	**Bold letters** ***Y0001***
IO Lock : OFF	Black letters	IO Lock : ON	Red letters

5.7.5 BOOL variables monitor

1 **Choose [5 BOOL variables] in the variables monitor menu.**
>> The following monitor will be displayed. (This is full screen mode)

This monitor displays the condition of the all “BOOL variables”.
The conditions of `B0000` to `B1999` are displayed.



B0000	***B0001***	B0002	B0003	***B0004***	B0005	B0006	B0007
B0008	B0009	B0010	B0011	B0012	B0013	B0014	B0015
B0016	B0017	B0018	B0019	B0020	B0021	B0022	B0023
B0024	B0025	B0026	B0027	B0028	B0029	B0030	B0031
B0032	B0033	B0034	B0035	B0036	B0037	B0038	B0039
B0040	B0041	B0042	B0043	B0044	B0045	B0046	B0047
B0048	B0049	B0050	B0051	B0052	B0053	B0054	B0055
B0056	B0057	B0058	B0059	B0060	B0061	B0062	B0063
B0064	B0065	B0066	B0067	B0068	B0069	B0070	B0071
B0072	B0073	B0074	B0075	B0076	B0077	B0078	B0079
B0080	B0081	B0082	B0083	B0084	B0085	B0086	B0087
B0088	B0089	B0090	B0091	B0092	B0093	B0094	B0095
B0096	B0097	B0098	B0099	B0100	B0101	B0102	B0103
B0104	B0105	B0106	B0107	B0108	B0109	B0110	B0111
B0112	B0113	B0114	B0115	B0116	B0117	B0118	B0119

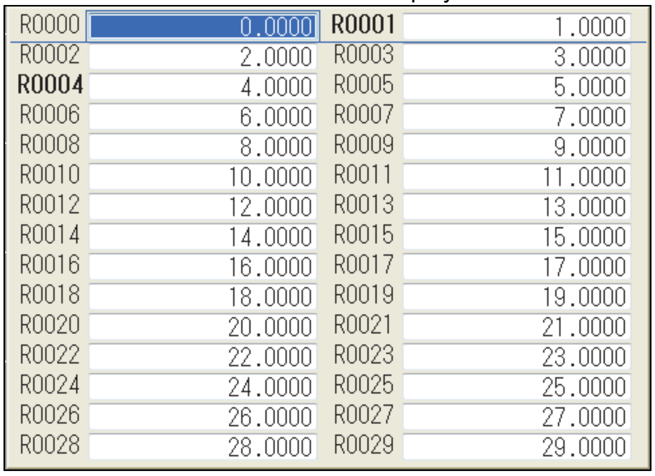
The display style of the letter will change depending on the condition of the signal.

Signal condition OFF	Background color Gray	Signal condition ON	Background color Yellow
Not used in the PLC program	Normal letters B0002	Used in the PLC program	**Bold letters** ***B0001***

5.7.6 Real variables monitor

1 **Choose [6 REAL variables] in the variables monitor menu.**
>> The following monitor will be displayed. (This is full screen mode)

This monitor displays the condition of the all "Real variables".
The values of *R0000* to *R0099* are displayed.



R0000	0.0000	**R0001**	1.0000
R0002	2.0000	R0003	3.0000
R0004	4.0000	R0005	5.0000
R0006	6.0000	R0007	7.0000
R0008	8.0000	R0009	9.0000
R0010	10.0000	R0011	11.0000
R0012	12.0000	R0013	13.0000
R0014	14.0000	R0015	15.0000
R0016	16.0000	R0017	17.0000
R0018	18.0000	R0019	19.0000
R0020	20.0000	R0021	21.0000
R0022	22.0000	R0023	23.0000
R0024	24.0000	R0025	25.0000
R0026	26.0000	R0027	27.0000
R0028	28.0000	R0029	29.0000

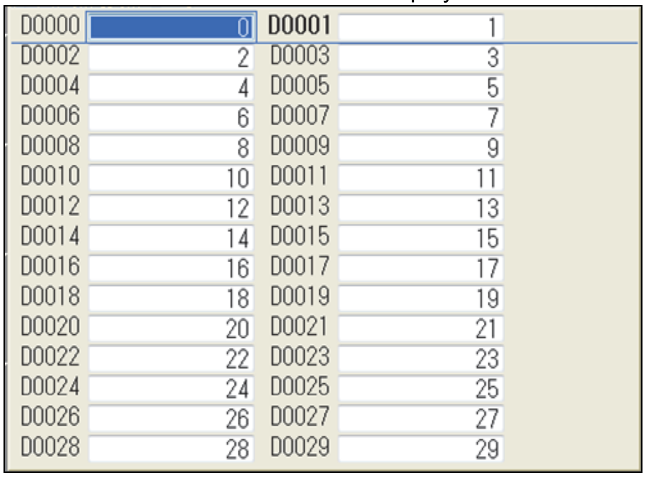
The display style of the letter will change depending on the condition of the signal.

Not used in the PLC program	Normal letters R0000	Used in the PLC program	**Bold letters** **R0001**

5.7.7 DINT variables monitor

1 **Choose [7 DINT variables] in the variables monitor menu.**
>> The following monitor will be displayed. (This is full screen mode)

This monitor displays the condition of the all “DINT variables”.
The values of *D0000* to *D0499* are displayed.

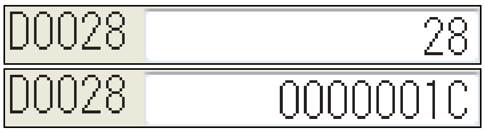


D0000	0	D0001	1
D0002	2	D0003	3
D0004	4	D0005	5
D0006	6	D0007	7
D0008	8	D0009	9
D0010	10	D0011	11
D0012	12	D0013	13
D0014	14	D0015	15
D0016	16	D0017	17
D0018	18	D0019	19
D0020	20	D0021	21
D0022	22	D0023	23
D0024	24	D0025	25
D0026	26	D0027	27
D0028	28	D0029	29

The display style of the letter will change depending on the condition of the signal.

Not used in the PLC program	Normal letters D0000	Used in the PLC program	**Bold letters** **D0001**

2 **Press [Shift] + [Left] / [Right] keys.**
>> The decimal value display and the hexadecimal display can be selected.

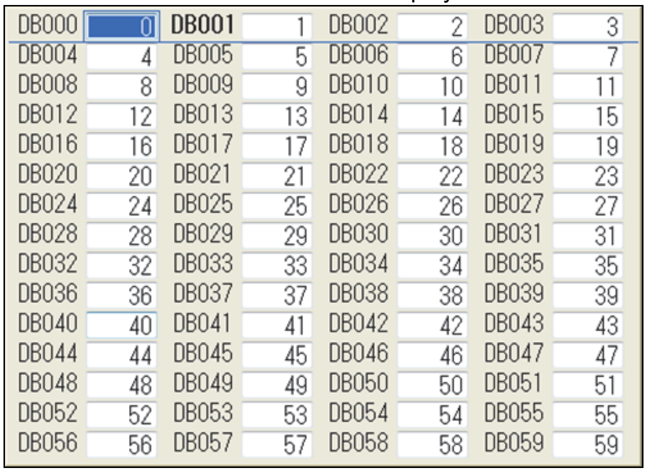


Decimal value display	D0028	28
Hexadecimal value display	D0028	0000001C

5.7.8 SINT variables monitor

1 **Choose [8 SINT variables] in the variables monitor menu.**
>> The following monitor will be displayed. (This is full screen mode)

This monitor displays the condition of the all “SINT variables”.
The values of *DB000* to *DB099* are displayed.



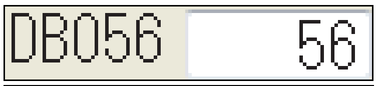
DB000	0	**DB001**	1	DB002	2	DB003	3
DB004	4	DB005	5	DB006	6	DB007	7
DB008	8	DB009	9	DB010	10	DB011	11
DB012	12	DB013	13	DB014	14	DB015	15
DB016	16	DB017	17	DB018	18	DB019	19
DB020	20	DB021	21	DB022	22	DB023	23
DB024	24	DB025	25	DB026	26	DB027	27
DB028	28	DB029	29	DB030	30	DB031	31
DB032	32	DB033	33	DB034	34	DB035	35
DB036	36	DB037	37	DB038	38	DB039	39
DB040	40	DB041	41	DB042	42	DB043	43
DB044	44	DB045	45	DB046	46	DB047	47
DB048	48	DB049	49	DB050	50	DB051	51
DB052	52	DB053	53	DB054	54	DB055	55
DB056	56	DB057	57	DB058	58	DB059	59

The display style of the letter will change depending on the condition of the signal.

Not used in the PLC program	Normal letters DB000	Used in the PLC program	**Bold letters** **DB001**

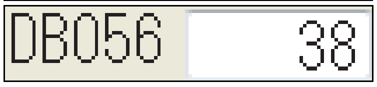
2 **Press [Shift] + [Left] / [Right] keys.**
>> The decimal value display and the hexadecimal display can be selected.

Decimal value display



DB056 56

Hexadecimal value display

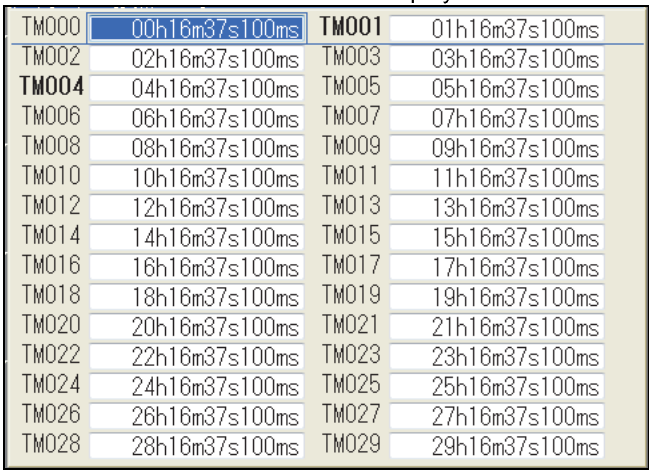


DB056 38

5.7.9 Timer variables monitor

1 **Choose [9 TIME variables] in the variables monitor menu.**
>> The following monitor will be displayed. (This is full screen mode)

This monitor displays the condition of the all “Timer variables”.
The values of ***TM000*** to ***TM499*** are displayed.



TM000	00h16m37s100ms	**TM001**	01h16m37s100ms
TM002	02h16m37s100ms	TM003	03h16m37s100ms
TM004	04h16m37s100ms	TM005	05h16m37s100ms
TM006	06h16m37s100ms	TM007	07h16m37s100ms
TM008	08h16m37s100ms	TM009	09h16m37s100ms
TM010	10h16m37s100ms	TM011	11h16m37s100ms
TM012	12h16m37s100ms	TM013	13h16m37s100ms
TM014	14h16m37s100ms	TM015	15h16m37s100ms
TM016	16h16m37s100ms	TM017	17h16m37s100ms
TM018	18h16m37s100ms	TM019	19h16m37s100ms
TM020	20h16m37s100ms	TM021	21h16m37s100ms
TM022	22h16m37s100ms	TM023	23h16m37s100ms
TM024	24h16m37s100ms	TM025	25h16m37s100ms
TM026	26h16m37s100ms	TM027	27h16m37s100ms
TM028	28h16m37s100ms	TM029	29h16m37s100ms

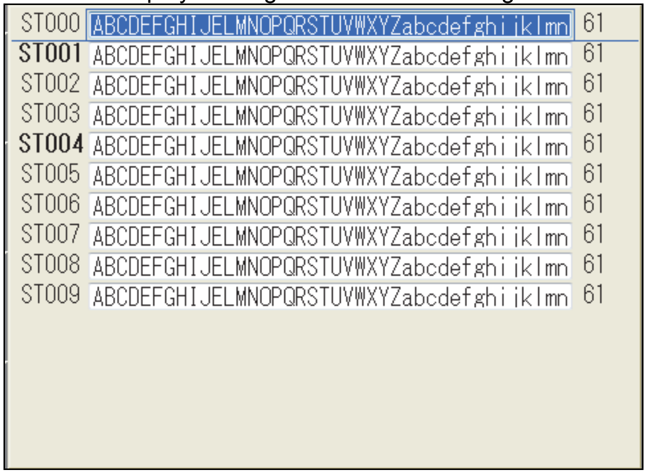
The display style of the letter will change depending on the condition of the signal.

Not used in the PLC program	Normal letters TM000	Used in the PLC program	**Bold letters** **TM001**

5.7.10 STRING variables monitor

1 **Choose [10 STRING variables] in the variables monitor menu.**
>> The following monitor will be displayed. (This is full screen mode)

This monitor displays the condition of the all "String variables".
The contents of *ST000* to *ST009* are displayed.
The value displayed at right side shows the length of the string variable.



```
ST000 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn 61
ST001 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn 61
ST002 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn 61
ST003 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn 61
ST004 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn 61
ST005 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn 61
ST006 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn 61
ST007 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn 61
ST008 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn 61
ST009 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn 61
```

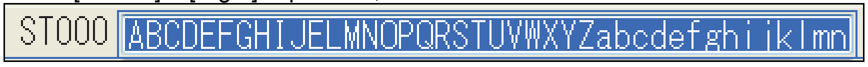
The display style of the letter will change depending on the condition of the signal.

Not used in the PLC program	Normal letters ST002	Used in the PLC program	**Bold letters** **ST001**

2 **Press [Enable] + [Left] / [right] keys.**
>> Displayed content is shifted 1 letter.

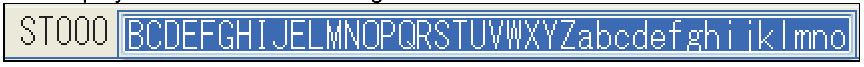
(Example)

When [Enable] + [Right] is pressed,



```
ST000 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn
```

The displayed content will be changed like this.

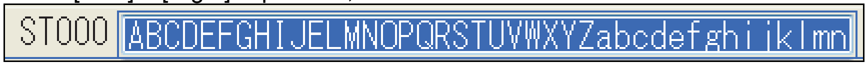


```
ST000 BCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmno
```

3 **Press [Shift] + [Left] / [right] keys.**
>> Displayed content is shifted 1 page.

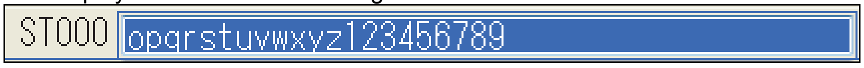
(Example)

When [Shift] + [Right] is pressed,



```
ST000 ABCDEFGHIJELMNOPQRSTUVWXYZabcdefghiiklmn
```

The displayed content will be changed like this.



```
ST000 opqrstuvwxyz123456789
```

NOTE

Chapter 6 Input/output relay lists

The correspondences between the logical input/output signals, physical input/output signals and the numbers of the input/output relays of the software PLC are described in this chapter.

6.1 Logical input/output relays

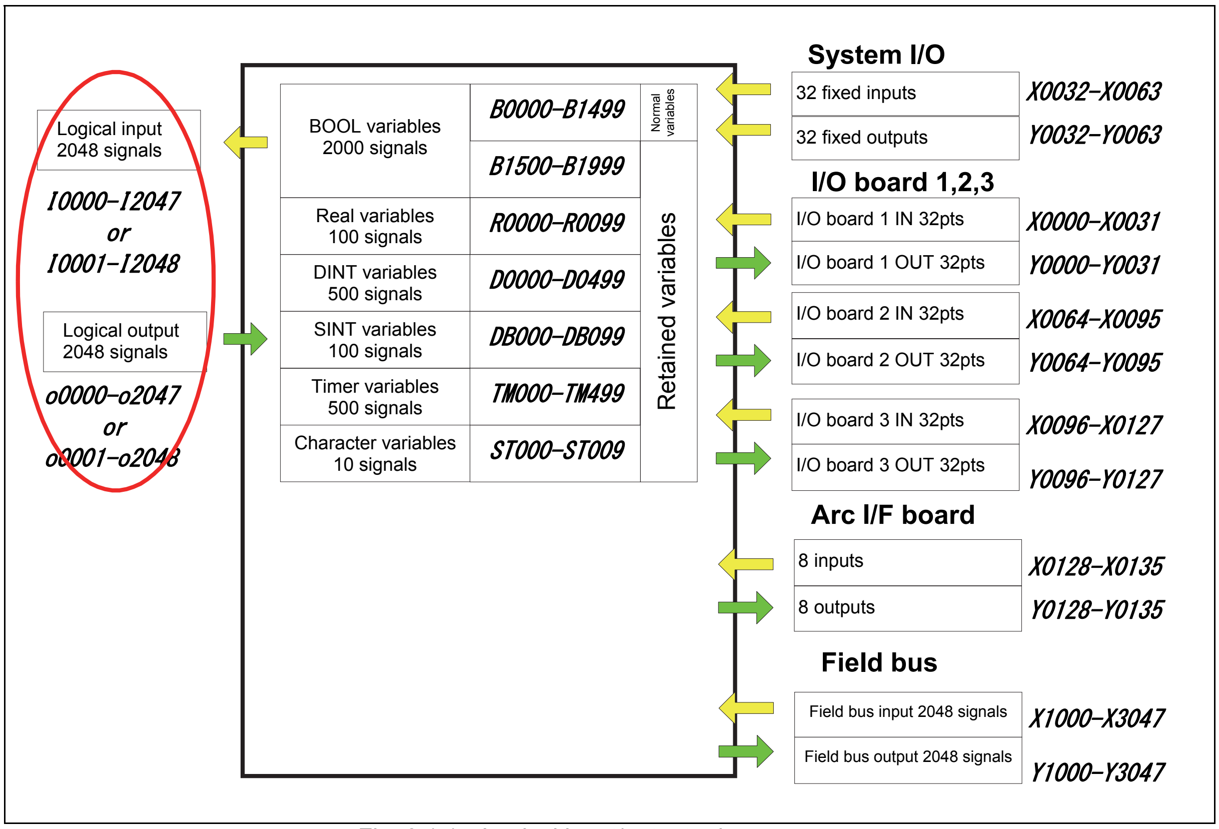


Fig. 6.1.1 Logical input/output relays

The logical input/output signals are the input/output signals of this controller as seen from the software PLC. The logical input/output relays are identified using a code starting with I (input) or O (output) so that their identification is consistent with the input/output signals which are actually recorded in the programs.

6.1.1 Relay numbers

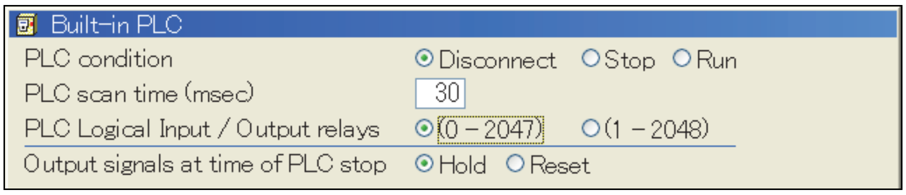
This function enables the settings of logical input/output relay numbers with the Constant menu.
Select the numbers from the two types listed in Table 6.1.1. The factory setting is made to the range of "0 to 2047".

Table 6.1.1 Logical input and output

Selection range	Description
0 - 2047	Used to make settings of relay numbers in the range of 0 to 2047. Signal numbers are assigned in the range of 1 to 2048. Consequently, "Logical input relay number = Logical input signal number - 1" "Logical output relay number = Logical output signal number - 1"
1 - 2048	Used to make settings of relay numbers in the range of 1 to 2048, like the signal numbers. "Logical input relay number = Logical input signal number" "Logical output relay number = Logical output signal number"

Constant Setting

1 **While in teach mode, select f5 <Constant Setting> - [1 Control Constants] and [4 Built-in PLC] from the menu on display.**
>> The setting menu related to the built-in PLC is as shown below. Move the cursor to the PLC logical input/output.

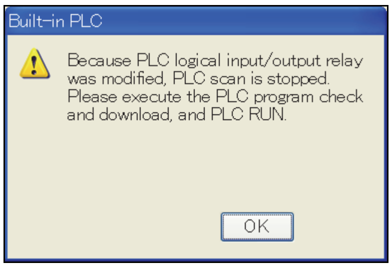

Built-in PLC

PLC condition	[✓ Disconnect] Stop Run
PLC scan time (msec)	30
PLC Logical Input / Output relays	[✓ (0 – 2047)] (1 – 2048)
Output signals at time of PLC stop	[✓ Hold] Reset

2 **Press [Enable] + [Right/Left] at a time to select either 0-2047 or 1-2048.**

Complete

3 **Press f12 <Complete>.**
>> If the PLC status is set to "Start", the message shown below is displayed.


Built-in PLC

Because PLC logical input/output relay was modified, PLC scan is stopped. Please execute the PLC program check and download, and PLC RUN.

Pressing [Enter] exits the Setting screen. At this time, the PLC status is set to "Stop". Be sure to execute checking according to information in Chapter 3 "Program check".

6.2 Physical input/output relays

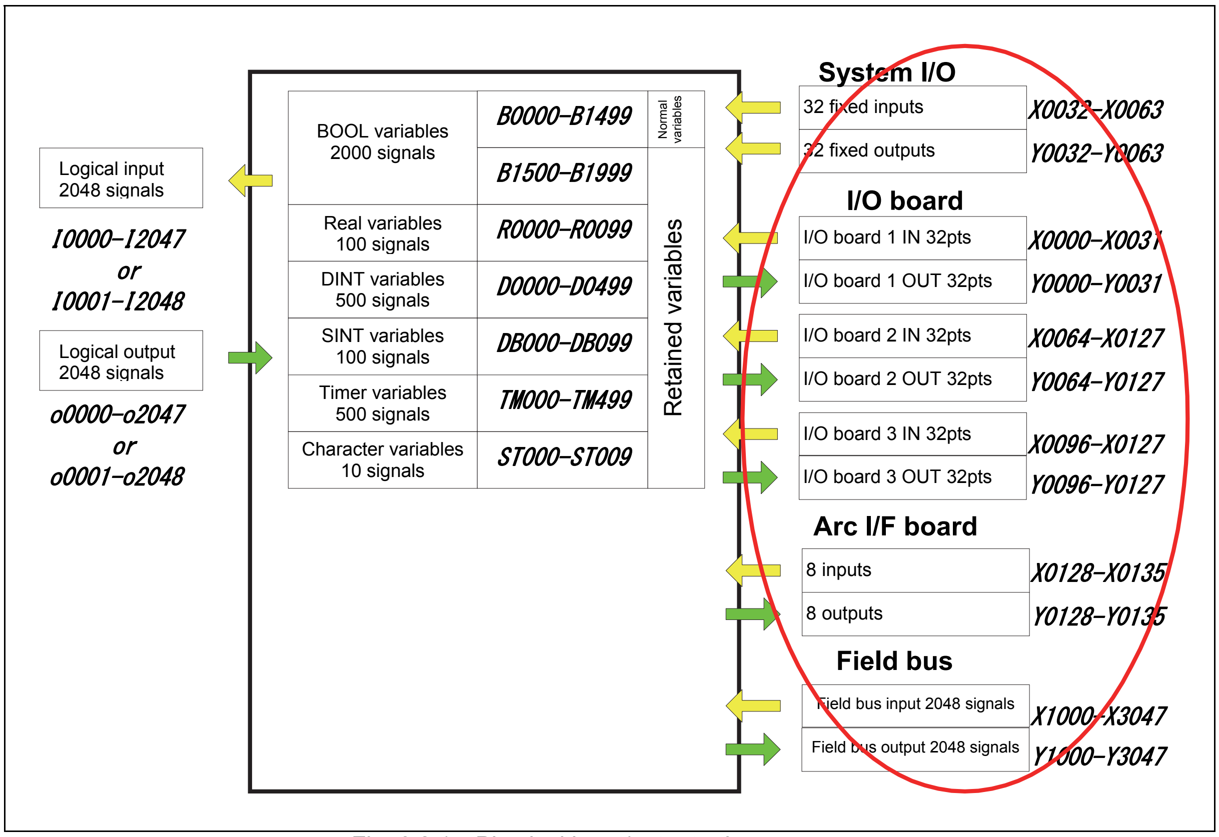


Fig. 6.2.1 Physical input/output relays

Table 6.2.1 The relationship between the relay numbers and the hardwares

Input		Output	
Relay No.	Hardware	Relay No.	Hardware
X0000 – X0031	I/O board 1	***Y0000 – Y0031***	I/O board 1
X0032 – X0063	Fixed input (System I/O)	***Y0032 – Y0063***	Fixed output (System I/O)
X0064 – X0095	I/O board 2	***Y0064 – Y0095***	I/O board 2
X0096 – X0127	I/O board 3	***Y0096 – Y0127***	I/O board 3
X0128 – X0135	Arc I/F board	***Y0128 – Y0135***	Arc I/F board
X1000 – X1511	Fieldbus CH1	***X1000 – X1511***	Fieldbus CH1
X1512 – X2023	Fieldbus CH2	***X1512 – X2023***	Fieldbus CH2
X2024 – X2535	Fieldbus CH3	***X2024 – X2535***	Fieldbus CH3
X2536 – X3047	Fieldbus CH4	***X2536 – X3047***	Fieldbus CH4

6.2.1 Fixed input/output (System I/O)

These are the servo ON/OFF and other input/output signals which are used to control the operations inside the controller. The fixed input/output signals can only be referenced by the software PLC. An error results during compiling if a program calling for signals to be output to the settled input/output signals is created.

Table 6.2.2 List of Fixed input/output relay numbers

Settled input signal name		Relay number	Settled output signal name		Relay number
1	Motors-ON	***X0032***	1	Motors-ON lamp	***Y0032***
2	G-STOP	***X0033***	2	Motors-ON request	***Y0033***
3	Start 1	***X0034***	3	Start lamp 1	***Y0034***
4	Start 2	***X0035***	4	Start lamp 2	***Y0035***
5	Start 3	***X0036***	5	Start lamp 3	***Y0036***
6	Start 4	***X0037***	6	Start lamp 4	***Y0037***
7	Stop	***X0038***	7	Stop lamp	***Y0038***
8	Playback mode	***X0039***	8	TP enable release	***Y0039***
9	Mat switch	***X0040***	9	Motors-ON enable	***Y0040***
10	—	***X0041***	10	Magnet-ON enable	***Y0041***
11	High-speed Teach	***X0042***	11	Internal/External	***Y0042***
12	P1 correct	***X0043***	12	WPS E-STOP ctrl	***Y0043***
13	Ext Emergency stop	***X0044***	13	CPU failure	***Y0044***
14	Emergency stop	***X0045***	14	TP mode	***Y0045***
15	Safety plug	***X0046***	15	Ext motors-ON	***Y0046***
16	Confirm motors-ON	***X0047***	16	Motors-ON lamp	***Y0047***
17	TP Emergency stop	***X0048***	—	—	—
18	Teach mode	***X0049***	—	—	—
19	—	***X0050***	—	—	—
20	TP enable SW	***X0051***	—	—	—
21	—	***X0052***	—	—	—
22	CR ON	***X0053***	—	—	—
23	Servo-ON	***X0054***	—	—	—
24	Servo enable	***X0055***	—	—	—
25	—	***X0056***	—	—	—
26	—	***X0057***	—	—	—
27	—	***X0058***	—	—	—
28	Magnet-ON	***X0059***	—	—	—
29	—	***X0060***	—	—	—
30	Weld detection	***X0061***	—	—	—
31	Inconsistency	***X0062***	—	—	—
32	—	***X0063***	—	—	—
33	Inconsist(GSTOP)	—	—	—	—
34	Inconsist(mode)	—	—	—	—
35	Inconsist(MAT-SW)	—	—	—	—
36	Inconsist(HI-SP)	—	—	—	—
37	Inconsist(Ext ES)	—	—	—	—
38	Inconsist(E.S.)	—	—	—	—
39	Inconsist(S.plug)	—	—	—	—
40	Inconsist(TP-ES)	—	—	—	—
41	Inconsist(ENB-SW)	—	—	—	—
42	Inconsist(CRON)	—	—	—	—
43	—	—	—	—	—
44	—	—	—	—	—
45	—	—	—	—	—
46	—	—	—	—	—
47	—	—	—	—	—
48	—	—	—	—	—

6.2.2 I/O board 1 (Option)

These are the input/output signals for the CNIN (input) and CNOUT (output) connectors of the I/O PCB provided as an optional accessory.

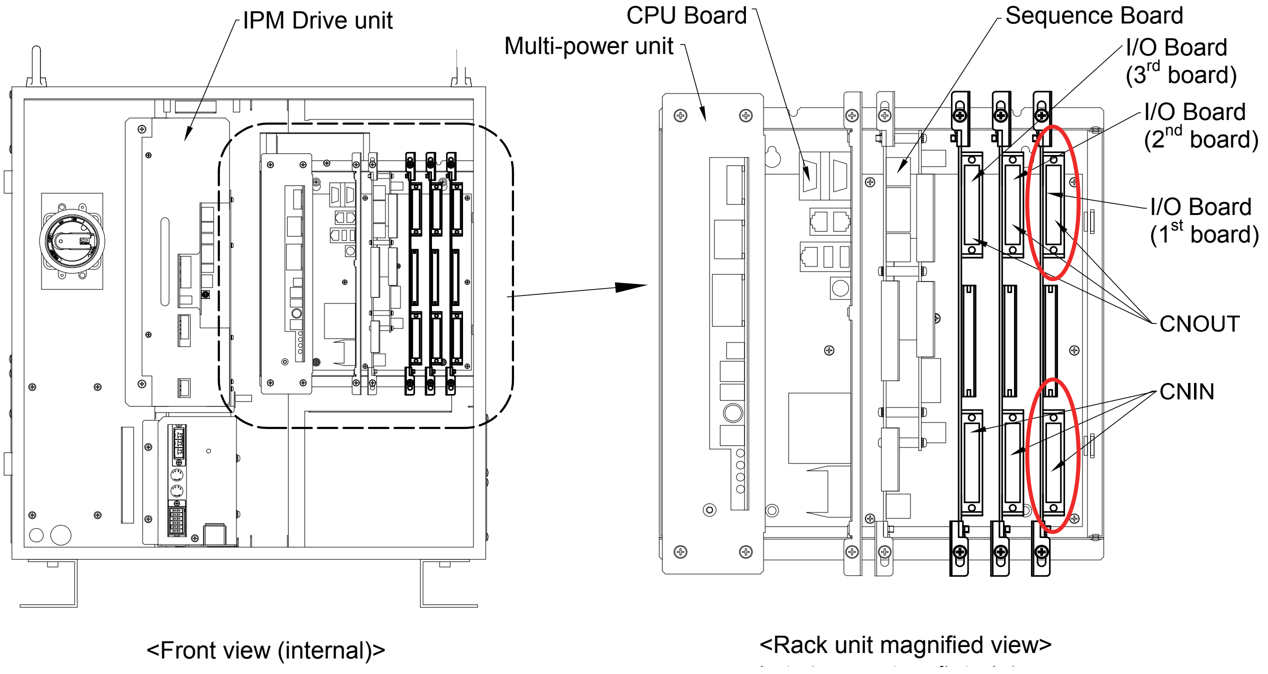


Fig. 6.2.2 Positions of I/O board signal connectors

Table 6.2.3 List of I/O board 1 input/output relay numbers

Input connector CNIN		Output connector CNOUT	
Connector pin number	Relay number	Connector pin number	Relay number
1	***X0000***	1	***Y0000***
2	***X0001***	2	***Y0001***
3	***X0002***	3	***Y0002***
4	***X0003***	4	***Y0003***
5	***X0004***	5	***Y0004***
6	***X0005***	6	***Y0005***
7	***X0006***	7	***Y0006***
8	***X0007***	8	***Y0007***
9	–	9	–
10	***X0008***	10	***Y0008***
11	***X0009***	11	***Y0009***
12	***X0010***	12	***Y0010***
13	***X0011***	13	***Y0011***
14	***X0012***	14	***Y0012***
15	***X0013***	15	***Y0013***
16	***X0014***	16	***Y0014***
17	***X0015***	17	***Y0015***
18	–	18	–
19	***X0016***	19	***Y0016***
20	***X0017***	20	***Y0017***
21	***X0018***	21	***Y0018***
22	***X0019***	22	***Y0019***
23	***X0020***	23	***Y0020***
24	***X0021***	24	***Y0021***
25	***X0022***	25	***Y0022***
26	***X0023***	26	***Y0023***
27	–	27	–
28	–	28	–
29	–	29	–
30	–	30	–
31	–	31	–
32	–	32	–
33	***X0024***	33	***Y0024***
34	***X0025***	34	***Y0025***
35	***X0026***	35	***Y0026***
36	***X0027***	36	***Y0027***
37	***X0028***	37	***Y0028***
38	***X0029***	38	***Y0029***
39	***X0030***	39	***Y0030***
40	***X0031***	40	***Y0031***
41	–	41	–
42	–	42	–
43	–	43	–
44	–	44	–
45	–	45	–
46	–	46	–
47	–	47	–
48	–	48	–
49	–	49	–
50	–	50	–

POINT

The I/O board 1,2, and 3 are optional items.

6.2.3 I/O board 2 and 3 (Option)

The input/output signals for the 2nd and subsequent optional boards correspond to the relay numbers below.

Table 6.2.4 Standard Input/Output Relay Number List

Board	Relay Number	Type
I/O Board 2	***X0064-X0095***	**INPUT**
	Y0064-Y0095	**OUTPUT**
I/O Board 3	***X0096-X0127***	**INPUT**
	Y0096-Y0127	**OUTPUT**

Furthermore, the input/output signals for the optional arc I/F board are as shown below.

6.2.4 Arc I/F board (Option)

The input/output signals for the Arc I/F board (option) correspond to the relay numbers below.

Table 6.2.5 Arc I/F Board Input/Output Relay Number List

Input Connector TBIN		Output Connector TBOUT	
Connector Pin number	Relay number	Connector Pin number	Relay number
1	***X00128***	1	**-**
2	***X00129***	2	**-**
3	***X00130***	3	***Y00128***
4	***X00131***	4	***Y00129***
5	**-**	5	***Y00130***
6	***X00132***	6	***Y00131***
7	***X00133***	7	**-**
8	***X00134***	8	***Y00132***
9	***X00135***	9	***Y00133***
10	**-**	10	***Y00134***
11	**-**	11	***Y00135***
12	**-**	12	**-**
13	**-**	**-**	**-**
14	**-**	**-**	**-**

6.2.5 Field bus input/output (Option)

The field bus input/output area is the area of the input/output signals used exclusively for the field buses of the DeviceNet, Profibus, RIO, etc. which are provided as optional accessories. Field buses can be installed for up to four channels. This means that different applications can be allocated to the channels: for instance, it is possible to allocate the slaves for use by the upstream controller and the master for use by the downstream control device.
When a multiple number of channels are to be used, the input/output signal area to be used by the channels will be as in Fig. 6.2.3. The start address (the lowest input/output signal number) is settled for each channel, and how many addresses from the start address are to be used is set.

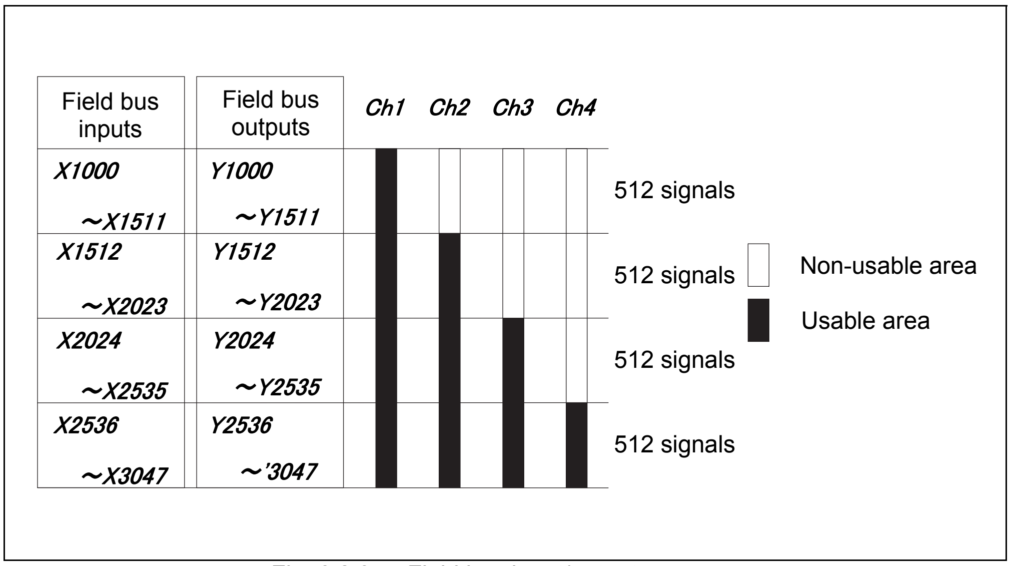


Fig. 6.2.3 Field bus input/output

For instance, when two channelsóone master channel and one slave channelóare to be used simultaneously by DeviceNet, 1,024 signals can be used for each by allocating the applications as follows.

Table 6.2.6 Example of settings to use 2 channels simultaneously

Channel number	Settings	Number of inputs/outputs	Input	Output
Ch1	DeviceNet / Slave	1024 signals	X1000-X2023	Y1000-Y2023
Ch2	Disabled	0	-	-
Ch3	DeviceNet / Master	1024 signals	X2024-X3047	Y2024-Y3047
Ch4	Disabled	0	-	-

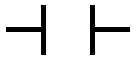
For details, refer to "DeviceNet Functions" in the options instruction manual and other documentation related to the field buses.

Chapter 7 List of command words

This chapter provides details of the basic commands and function commands (LD blocks) which can be used by the software PLC.

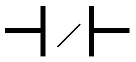
7.1 Basic commands

7.1.1 Contact A



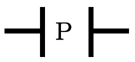
The status of the connecting wire on the right side of the contact is the logical product (AND) of the connecting wire on the left side and the variable allotted to the contact.

7.1.2 Contact B



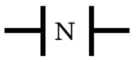
The status of the connecting wire on the right side of the contact is the logical product (AND) of the reverse of the connecting wire on the left side and the variable allotted to the contact.

7.1.3 Rise contact



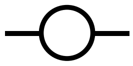
The status of the connecting wire on the right side of the contact is TRUE only when the connecting wire on the left side is TRUE and the BOOL variable has risen from FALSE to TRUE. At all other times, it is FALSE.

7.1.4 Fall Contact



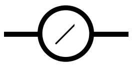
The status of the connecting wire on the right side of the contact is TRUE only when the connecting wire on the left side is TRUE and the BOOL variable has fallen from TRUE to FALSE. At all other times, it is FALSE.

7.1.5 Coil



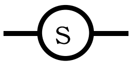
The status of the connecting wire on the left side is replaced by the variable allotted to the coil.
The status of the connecting wire on the left side is relayed as is to the connecting wire on the right side. The connecting wire on the right side is normally connected to the right bus.

7.1.6 Reverse coil



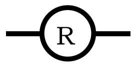
The reverse status of the connecting wire on the left side is replaced by the variable allotted to the coil.
The status of the connecting wire on the left side is relayed as is to the connecting wire on the right side. The connecting wire on the right side is normally connected to the right bus.

7.1.7 Set coil



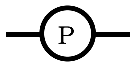
The values of the allotted variables are set to TRUE when the status of the connecting wire on the left side has been set to TRUE. The output variable is maintained until this status is reversed by the reset coil.
The status of the connecting wire on the left side is relayed as is to the connecting wire on the right side. The connecting wire on the right side is normally connected to the right bus.

7.1.8 Reset coil



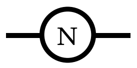
The values of the allotted variables are reset to FALSE when the status of the connecting wire on the left side has been set to TRUE. The output variable is maintained until this status is reversed by the set coil.
The status of the connecting wire on the left side is relayed as is to the connecting wire on the right side.The connecting wire on the right side is normally connected to the right bus.

7.1.9 Coil with rising edge detection



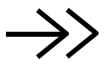
The allotted variables are set to TRUE when the status of the connecting wire on the left side has risen from FALSE to TRUE. At all other times, they are reset to FALSE.
The status of the connecting wire on the left side is relayed as is to the connecting wire on the right side. The connecting wire on the right side is normally connected to the right bus.

7.1.10 Coil with falling edge detection



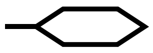
The allotted variables are set to TRUE when the status of the connecting wire on the left side has fallen from TRUE to FALSE. At all other times, they are reset to FALSE.
The status of the connecting wire on the left side is relayed as is to the connecting wire on the right side. The connecting wire on the right side is normally connected to the right bus.

7.1.11 Jump



When the Boolean status of the connecting wire on the left side of the jump symbol is TRUE, the program jumps to the corresponding label symbol and it executed from this symbol onward.

7.1.12 Return



When the Boolean status of the connecting wire on the left side of the return label is TRUE, the program returns without the subsequent programs being executed.

7.2 Function commands (LD blocks)

With this software PLC, the comparison command, timer commands and other function commands provided as a standard feature are referred to as "LD blocks," and they are input using numbers or names. LD blocks are expressed as square blocks (function blocks), and they must always have inputs and outputs.

For instance, the "timer command" is LD block No. 24 = TON (rise delay time). The "TON" function block is identical to what is supported as a standard feature by ICS Triplex IsaGRAF Inc. (ISaGRAF–PRO). "LD block No. 24" is a number provided independently by this controller as a way of facilitating teaching, and it is not supported by ICS Triplex IsaGRAF Inc. (ISaGRAF–PRO).

IMPORTANT

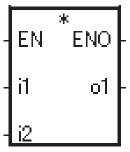
The variables used with the LD block are initialized when turning ON the primary power of the robot controller or when starting the Software PLC scanning process.

Therefore, the variables that are assigned to the LD block outputs may not be kept at timings like those.

The blocks in which the variables are initialized are as follows;
- RS (RS flip-flop)
- SR (SR flip-flop)
- CTD (count down)
- CTU (count up)
- CTUD (up/down counter)
- HYSTER (Boolean type value for hysteresis of two real number values)
- INTEGRAL (time-base-related interation)
- LIM_ALRM (upper and lower limit alarms with hysteresis)
- STACKINT (integer stack)

Each function command will now be described in detail in turn.
With the permission of ICS Triplex IsaGRAF, the ISaGRAF (ISaGRAF–PRO) the details given below as the text are based on the user guide. Many languages (such as ST and IL) besides the LD (ladder) language are used in the explanatory text. They are used for the purpose of describing the behavior of the commands, and they cannot be displayed or edited by the teach pendant of this controller. To program with a language other than the LD (ladder) language, use the ISaGRAF Workbench.

7.2.1 No.1 : *(multiplication)



Note: The number of inputs for this command can be changed to 3 or more.

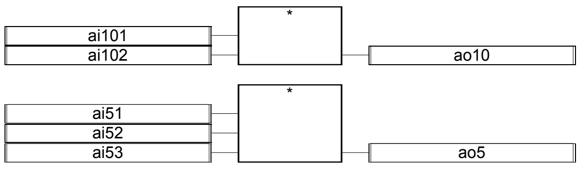
Argument:

Input **i***	Integer/real number type	(All inputs are the same type.)
Output **o1**	Integer/real number type	Signed multiplication of inputs

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
Multiplication of multiple integer/real number variables

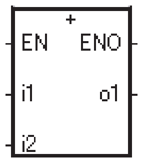
(* FBD example using multiplication block *)



```
(* Equivalent ST language: *)
ao10 := ai101 * ai102;
ao5 := (ai51 * ai52) * ai53;

(* Equivalent IL language: *)
LD      ai101
MUL     ai102
ST      ao10
LD      ai51
MUL     ai52
MUL     ai53
ST      ao5
```

7.2.2 No.2：+(addition)



Note: The number of inputs for this command can be changed to 3 or more.

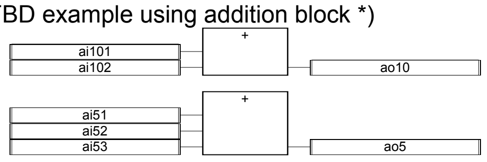
Argument:

Input **i***	Integer/real number type	(All inputs are the same type.)
Output **o1**	Integer/real number type	Signed addition of all inputs

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
Signed addition of multiple integer/real number variables

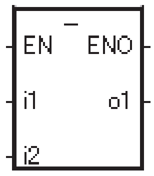
(* FBD example using addition block *)



```
(* Equivalent ST language: *)
ao10 := ai101 + ai102;
ao5 := (ai51 + ai52) + ai53;

(* Equivalent IL language: *)
LD      ai101
ADD     ai102
ST      ao10
LD      ai51
ADD     ai52
ADD     ai53
ST      ao5
```

7.2.3 No.3：-(subtraction)



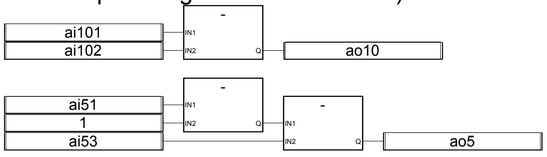
Argument:

Input **i1* i2**	Integer/real number type	(IN1 and IN2 are the same type.)
Output **o1**	Integer/real number type	Subtraction result (IN1 – IN2)

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
Subtraction of two integer/real number variables (first – second)

(* FBD example using subtraction block *)

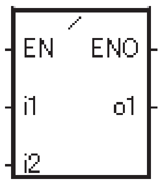


(* Equivalent ST language: *)
ao10 := ai101 – ai102;
ao5 := (ai51 – 1) – ai53;

(* Equivalent IL language: *)

```
LD      ai101
SUB     ai102
ST      ao10
LD      ai51
SUB     1
SUB     ai53
ST      ao5
```

7.2.4 No.4 : / (devision)



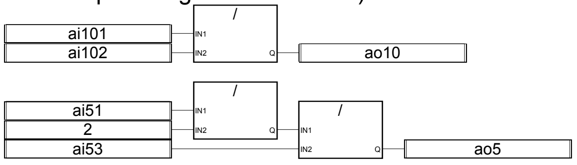
Argument:

Input **il, i2**	Integer/real number type	The divisor must not be zero. (IN1 and IN2 are the same type.)
Output **o1**	Integer/real number type	Signed division (IN1/IN2)

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
Division of two integer/real number variables (first/second)

(* FBD example using division block *)



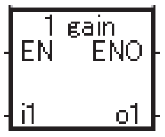
(* Equivalent ST language: *)
ao10 := ai101 / ai102;
ao5 := (ai5 / 2) / ai53;

(* Equivalent IL language: *)

```
LD      ai101
DIV     ai102
ST      ao10
```

```
LD      ai51
DIV     2
DIV     ai53
ST      ao5
```

7.2.5 No.5 : 1 gain (assign)



Argument:

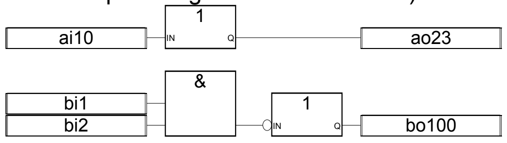
Input **i1**	All types
Output **o1**	All types

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes one variable with another.

This block connects the input and output directly.

(* FBD example using substitution block *)

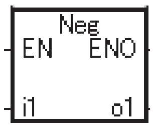


(* Equivalent ST language: *)
ao23 := ai10;
bo100 := NOT (bi1 AND bi2);

(* Equivalent IL language: *)

```
LD      ai10
ST      ao23
LD      bi1
AND     bi2
LDN     bo100
```

7.2.6 No.6 : Neg (signed reversal of integers)



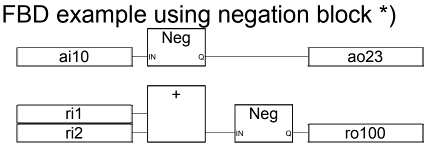
Argument:

Input **i1**	Integer/real number type	The input and output parameters are the same type.
Output **o1**	Integer/real number type	The input and output parameters are the same type.

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes the reverse of variables (signed reversal).

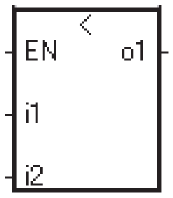
(* FBD example using negation block *)



```
(* Equivalent ST language: *)
ao23 := – (ai10);
ro100 := – (ri1 + ri2);

(* Equivalent IL language: *)
LD      ai10
MUL     –1
ST      ao23
LD      ri1
ADD     ri2
MUL     –1.0
LD      ro100
```

7.2.7 No.7 : < (less than)



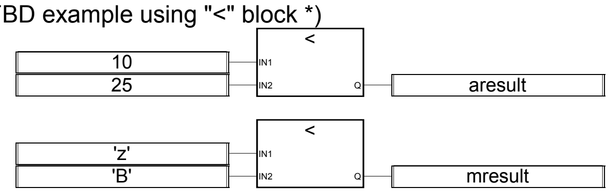
Argument:

Input **i1, i2**	Integer type, real number type, timer type, character string type	The two inputs are the same type.
Output **o1**	Boolean type	TRUE if I1 < I2

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.

Explanation:
This command compares whether a particular variable is less than another variable.

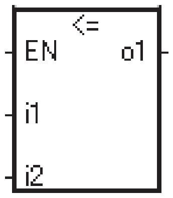
(* FBD example using "<" block *)



```
(* Equivalent ST language: *)
aresult := (10 < 25); (* aresult is TRUE *)
mresult := ('z' < 'B'); (* mresult is FALSE *)

(* Equivalent IL language: *)
LD      10
LT      25
ST      aresult
LD '    z'
LT      'B'
ST      mresult
```

7.2.8 No.8 : <= (less than or equal to)

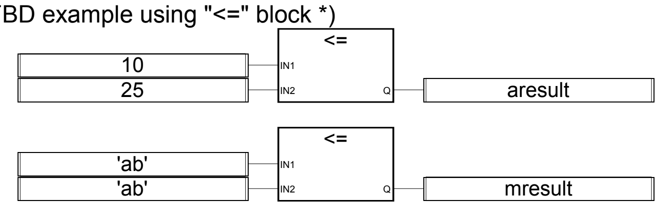


Argument:

Input **i1, i2**	Integer type, real number type, character string type	The two inputs are the same type.
Output **o1**	Boolean type	TRUE if I1 <= I2

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.

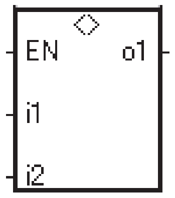
Explanation:
This command compares whether a particular variable is less than or equal to another variable.
(* FBD example using "<=" block *)



(* Equivalent ST language: *)
aresult := (10 <= 25); (* aresult is TRUE *)
mresult := ('ab' < 'ab'); (* mresult is TRUE *)

(* Equivalent IL language: *)
```
LD      10
LE      25
ST      aresult
LD      'ab'
LE      'ab'
ST      mresult
```

7.2.9 No.9 : <> (not equal to)



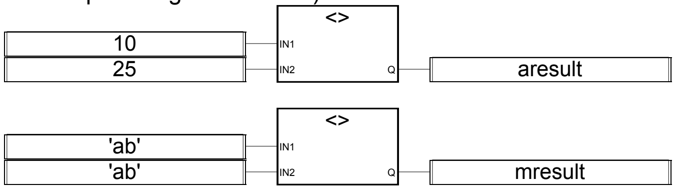
Argument:

Input **i1, i2**	Integer type, real number type, character string type	The two inputs are the same type.
Output **o1**	Boolean type	TRUE if I1 <> I2

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.

Explanation:
This command compares whether a particular variable is not equal to another variable.

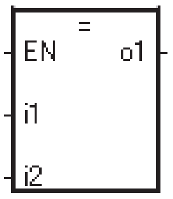
(* FBD example using "<>" block *)



(* Equivalent ST language: *)
aresult := (10 <> 25); (* aresult is TRUE *)
mresult := ('ab' <> 'ab'); (* mresult is FALSE *)

(* Equivalent IL language: *)
```
LD      10
NE      25
ST      aresult
LD      'ab'
NE      'ab'
ST      mresult
```

7.2.10 No.10 : = (equal to)



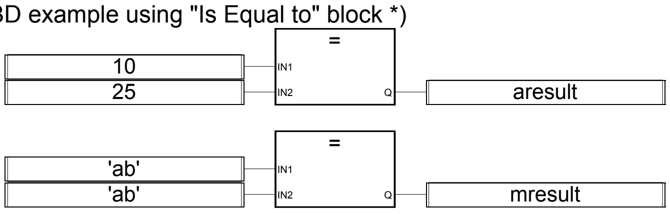
Argument:

Input **i1, i2**	Integer type, real number type, character string type	The two inputs are the same type.
Output **o1**	Boolean type	TRUE if IN1 = IN2

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.

Explanation:
This command compares whether a particular variable is equal to another variable.

(* FBD example using "Is Equal to" block *)

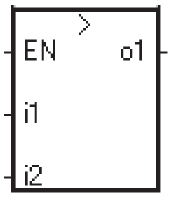


(* Equivalent ST language: *)
aresult := (10 = 25); (* aresult is FALSE *)
mresult := ('ab' = 'ab'); (* mresult is TRUE *)

(* Equivalent IL language: *)

```
LD      10
EQ      25
ST      aresult
LD      'ab'
EQ      'ab'
ST      mresult
```

7.2.11 No.11 : > (greater than)



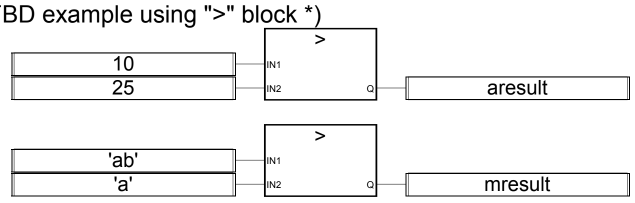
Argument:

Input **i1, i2**	Integer type, real number type, timer type, character string type	The two are of the same type.
Output **o1**	Boolean type	TRUE if IN1 > IN2

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.

Explanation:
This command compares whether a particular variable is greater than another variable.

(* FBD example using ">" block *)

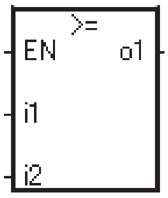


(* Equivalent ST language: *)
aresult := (10 > 25); (* aresult is FALSE *)

```
mresult := ('ab' > 'a'); (* mresult is TRUE *)

(* Equivalent IL language: *)
LD      10
GT      25
ST      aresult
LD      'ab'
GT      'a'
ST      mresult
```

7.2.12 No.12：>= (greater than or equal to)



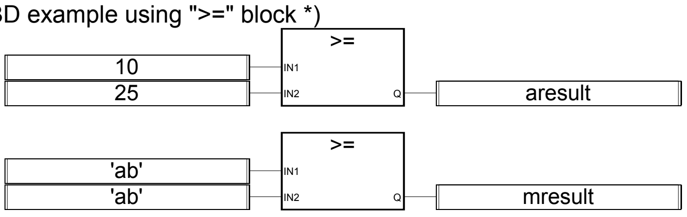
Argument:

Input **i1, i2**	Integer type, real number type, character string type	The two inputs are the same type.
Output **o1**	Boolean type	TRUE if IN1 >= IN2

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.

Explanation:
This command compares whether a particular variable is greater than or equal to another variable.

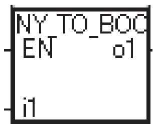
(* FBD example using ">=" block *)



```
(* Equivalent ST language: *)
aresult := (10 >= 25); (* aresult is FALSE *)
mresult := ('ab' > 'ab'); (* mresult is TRUE *)

(* Equivalent IL language: *)
LD      10
GE      25
ST      aresult
LD      'ab'
GE      'ab'
ST      mresult
```

7.2.13 No.13：ANY_TO_BOOL (conversion to Boolean type)



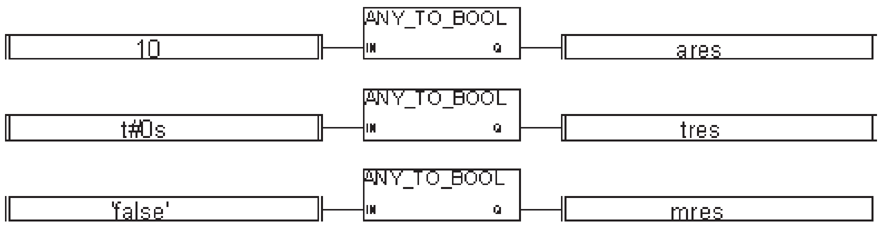
Argument:

Input **i1**	ANY	All non–integer values
Output **o1**	Boolean type	TRUE: Value other than 0
		FALSE: 0
		TRUE: 'TRUE' character string
		FALSE: 'FALSE' character string

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.

Explanation:
This command converts variables into Boolean types.

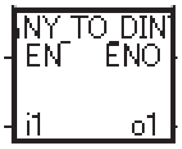
(* FBD example using Boolean type conversion block *)



```
(* Equivalent ST language: *)
ares := ANY_TO_BOOL (10);        (* ares is TRUE *)
tres := ANY_TO_BOOL (t#0s);      (* tres is FALSE *)
mres := ANY_TO_BOOL ('false');   (* mres is FALSE *)
```

```
(* Equivalent IL language: *)
LD              10
ANY_TO_BOOL
ST              ares
LD              t#0s
ANY_TO_BOOL
ST              tres
LD              'false'
ANY_TO_BOOL
ST              mres
```

7.2.14 No.14：ANY_TO_DINT (conversion to interger type)



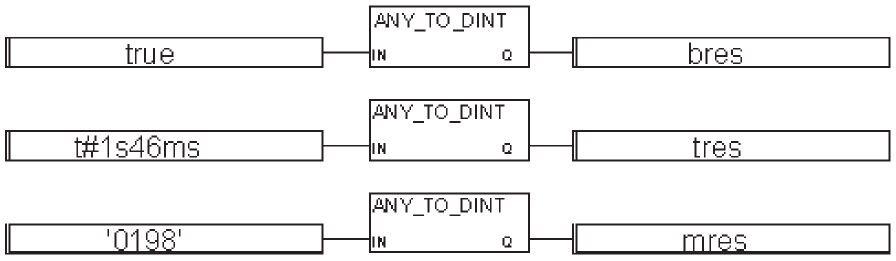
Argument:

Input **i1**	ANY	All non–integer values
Output **o1**	integer type	0 : if IN is FALSE,
		1 : if IN is TRUE
		Timer ms value
		Integer part of real number variable
		Decimal expression of character string

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command converts variables into integer types.

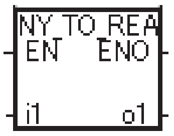
(* FBD example using integer type conversion block *)



```
(* Equivalent ST language: *)
bres := ANY_TO_DINT (true);        (* bres is 1 *)
tres := ANY_TO_DINT (t#1s46ms);    (* tres is 1046 *)
mres := ANY_TO_DINT ('0198');      (* mres is 198 *)
```

```
(* Equivalent IL language: *)
LD              true
ANY_TO_DINT
ST              bres
LD              t#1s46ms
ANY_TO_DINT
ST              tres
LD              '0198'
ANY_TO_DINT
ST              mres
```

7.2.15 No.15：ANY_TO_REAL (conversion to real number type)



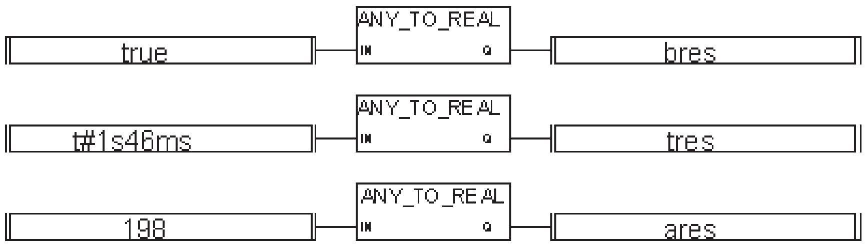
Argument:

Input **i1**	ANY	All non–integer values
Output **o1**	real number type	0.0 : IN is FALSE
		1.0 : IN is TRUE
		Numerical value of timer in millisecond units
		Same numerical value as integer type

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command converts variables into real number types.

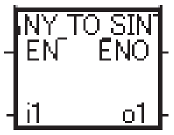
(* FBD example using real number type conversion block *)



```
(* Equivalent ST language: *)
bres := ANY_TO_REAL (true);          (* bres is 1.0 *)
tres := ANY_TO_REAL (t#1s46ms);      (* tres is 1046.0 *)
ares := ANY_TO_REAL (198);           (* ares is 198.0 *)
```

```
(* Equivalent IL language: *)
LD              true
ANY_TO_REAL
ST              bres
LD              t#1s46ms
ANY_TO_REAL
ST              tres
LD              198
ANY_TO_REAL
ST              ares
```

7.2.16 No.16 : ANY_TO_SINT (conversion to short interger type)



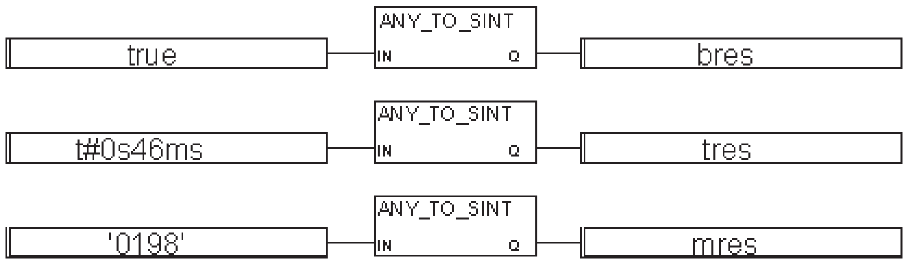
Argument:

Input **i1**	ANY	All non–integer values
Output **o1**	real number type	0 if IN is FALSE, 1 if IN is TRUE
		Numerical value of timer in millisecond units
		With real number type, integer part
		Decimal notation expressed by character string

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

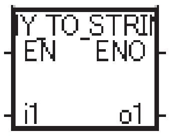
Explanation:
This command converts variables into short integers.

(* FBD example using "conversion to short integer" operator *)



```
(* Equivalent ST language: *)
bres := ANY_TO_SINT (true);          (* bres is 1 *)
tres := ANY_TO_SINT (t#0s46ms);      (* tres is 46 *)
mres := ANY_TO_SINT ('0198');        (* mres is 198 *)
```

7.2.17 No.17 : ANY_TO_STRING (conversion to character string type)

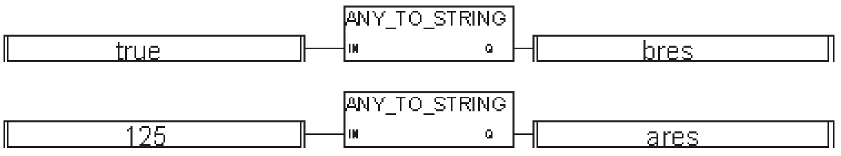


Argument:

Input **i1**	any	All non–character string values
Output **o1**	character string type	'false' or 'true' when I1 is a Boolean type
		Expression in decimal notation when I1 is an integer or real number type

Explanation:
This command converts variables into character string types.

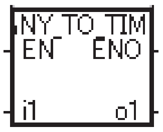
(* FBD example using variable length character string type conversion block *)



```
(* Equivalent ST language: *)
bres := ANY_TO_STRING (true);        (* bres is 'TRUE' *)
ares := ANY_TO_STRING (125);         (* ares is '125' *)
```

```
(* Equivalent IL language: *)
LD              true
ANY_TO_STRING
ST              bres
LD              125
ANY_TO_STRING
ST              ares
```

7.2.18 No.18：ANY_TO_TIME (conversion to tiomer type)



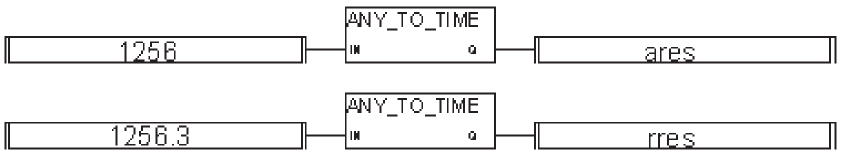
Argument:

Input **i1**	any	All non–timer values Numerical value in millisecond units
Output **o1**	timer type	Timer value expressed by I1 When I1 is a real number: Integer part of the number

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command converts integer and real number type variables into timer values.

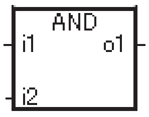
(* FBD example using timer type conversion block *)



```
(* Equivalent ST language: *)
ares := ANY_TO_TIME (1256);        (* ares := t#1s256ms *)
rres := ANY_TO_TIME (1256.3);      (* rres := t#1s256ms *)
```

```
(* Equivalent IL language: *)
LD              1256
ANY_TO_TIME
ST              ares
LD              1256.3
ANY_TO_TIME
ST              rres
```

7.2.19 No.19：AND (Boolean AND)

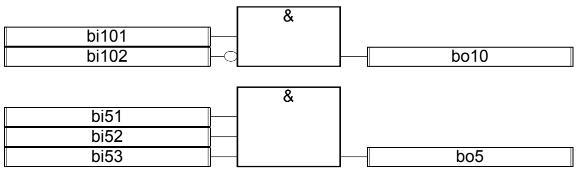


Note: The number of inputs for this command can be changed to 3 or more.
Argument:

Input **i1, i2**	Boolean type
Output **o1**	Boolean type Logical product of input parameters

Explanation:
Logical product of two or more input parameters

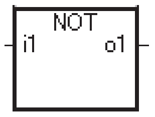
(* FBD example using "AND" block *)



```
(* Equivalent ST language: *)
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) AND bi53;

(* Equivalent IL language: *)
LD      bi101     (* Current result := bi101 *)
ANDN    bi102     (* Current result := bi101 AND not(bi102) *)
ST      bo10      (* bo := Current result *)
LD      bi51      (* Current result := bi51;
&       bi52      (* Current result := bi51 AND bi52 *)
&       bi53      (* Current result := bi51 AND bi52 *) AND bi53 *)
ST      bo5       (* bo := Current result *)
```

7.2.20 No.20：NOT (Boolean NOT)

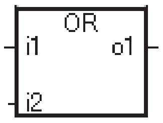


Argument:

Input **i1**	Boolean type
Output **o1**	Boolean type TRUE if I1 is FALSE, FALSE if I1 is TRUE

Explanation:
This command returns NOT of the Boolean expression

7.2.21 No.21：OR (Blooean OR)



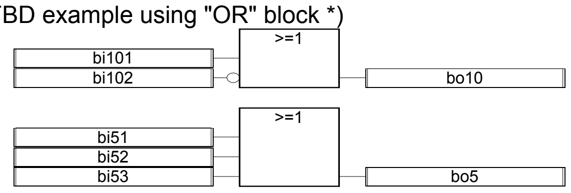
Note: The number of inputs for this command can be changed to 3 or more.

Argument:

Input **i1, i2**	Boolean type	
Output **o1**	Boolean type	Logical sum of input parameters

Explanation:
Logical sum of two or more input parameters

(* FBD example using "OR" block *)

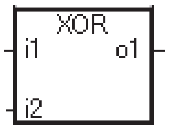


```
(* Equivalent ST language: *)
bo10 := bi101 OR NOT (bi102);
bo5 := (bi51 OR bi52) OR bi53;

(* Equivalent IL language: *)
LD      bi101
ORN     bi102
```

```
ST      bo10
LD      bi51
OR      bi52
OR      bi53
ST      bo5
```

7.2.22 No.22：XOR (Boolean Exclusive OR)

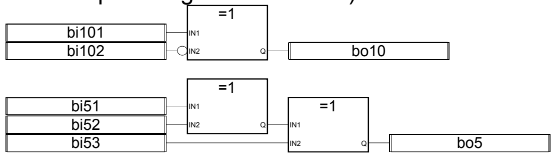


Argument:

Input **i1, i2**	Boolean type	
Output **o1**	Boolean type	Exclusive OR of two Boolean inputs

Explanation:
Exclusive OR of two Boolean values

(* FBD example using "XOR" block *)

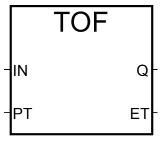


(* Equivalent ST language: *)
bo10 := bi101 XOR NOT (bi102);
bo5 := (bi51 XOR bi52) XOR bi53;

(* Equivalent IL language: *)

```
LD      bi101
XORN    bi102
ST      bo10
LD      bi51
XOR     bi52
XOR     bi53
ST      bo5
```

7.2.23 No.23：TOF (fall delay time)

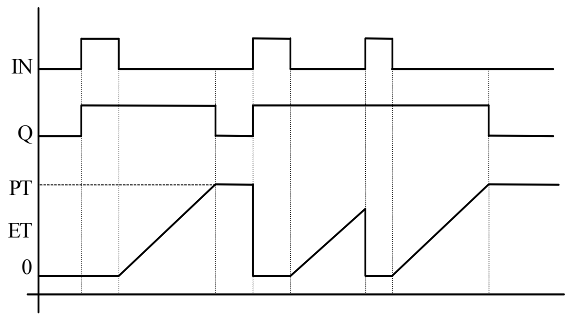


Argument:

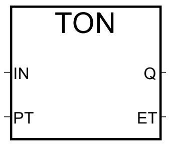
IN	Boolean type	Starts the incrementing of the timer value at the fall detection. Stops or resets the timer value at the rise detection.
PT	timer type	Timer value set
Q	Boolean type	When TRUE, the timer value has not been reached.
ET	timer type	Current elapsed timer value

Explanation:
Increments the internal timer value to the specified value (PT).

Time chart:



7.2.24 No.24：TON (rise delay time)

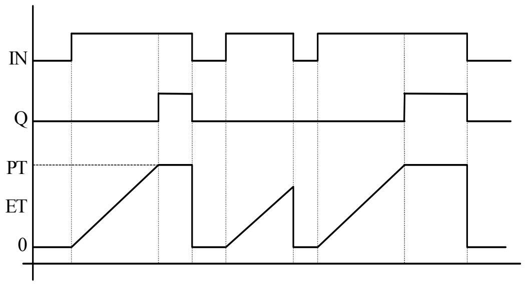


Argument:

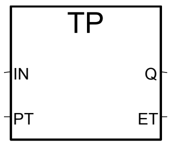
IN	Boolean type	Starts the incrementing of the timer value at the rise detection. Stops or resets the timer value at the fall detection.
PT	timer type	Time–up value set
Q	Boolean type	When TRUE, the time–up value set has been reached.
ET	timer type	Current elapsed timer value

Explanation:
Increments the internal timer value to the specified value (PT).

Time chart:



7.2.25 No.25：TP (pulse timing)

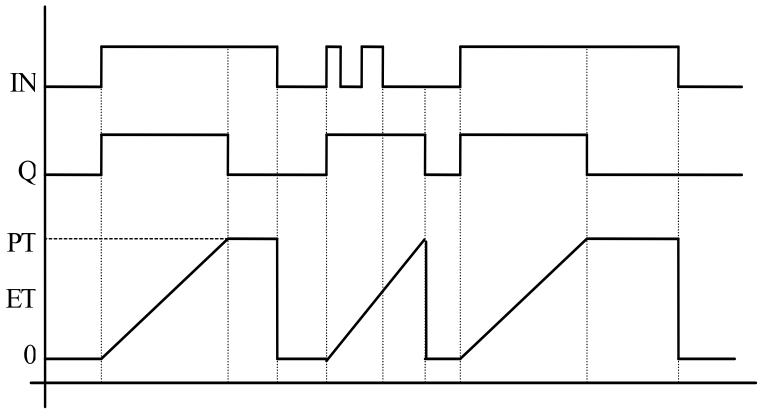


Argument:

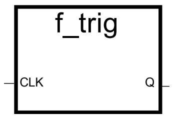
IN	Boolean type	Starts the incrementing of the timer value at the rise detection. (unless the value is already being incremented). If the timer value set has been reached in the FALSE status, the timer value is reset. Any change in IN is ignored while the timer time is being incremented.
PT	timer type	Timer value set
Q	Boolean type	When TRUE, the timer time is being incremented.
ET	timer type	Current elapsed timer value

Explanation:
Increments the internal timer value to the specified value (PT).

Time chart:



7.2.26 No.26 : F_TRIG (falling edge detection)

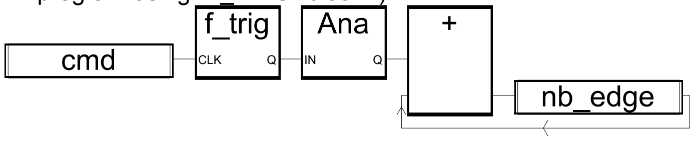


Argument:

CLK	Boolean type	
Q	Boolean type	TRUE: When CLK has fallen from TRUE to FALSE FALSE: In all other cases

Explanation:
This command detects the fall of the Boolean type integer. (It makes a comparison with the value in the previous cycle.)

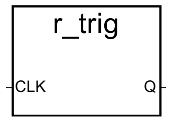
(* FBD program using "F_TRIG" block *)



```
(* Equivalent ST language: F_TRIG1 is the instance of the F_TRIG block. *)
F_TRIG1 (cmd);
nb_edge := ANA (F_TRIG1.Q) + nb_edge; 0
```

```
(* Equivalent IL language: *)
LD      cmd
ST      F_TRIG1.clk
CA      F_TRIG1
LD      F_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

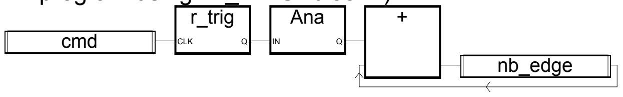
7.2.27 No.27 : R_TRIG (rising edge detection)



Argument:

CLK	Boolean type	
Q	Boolean type	TRUE : When CLK has risen from FALSE to TRUE FALSE: In all other cases

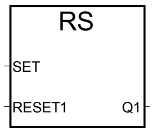
Explanation:
This command detects the rise of the Boolean type integer. (It makes a comparison with the value in the previous cycle.)
(* FBD program using "R_TRIG" block *)



(* Equivalent ST language: R_TRIG1 is the instance of the R_TRIG block. *)
R_TRIG1(cmd);
nb_edge := ANA (R_TRIG1.Q) + nb_edge;

(* Equivalent IL language: *)
```
LD      cmd
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

7.2.28 No.28：RS (RS flip-flop)



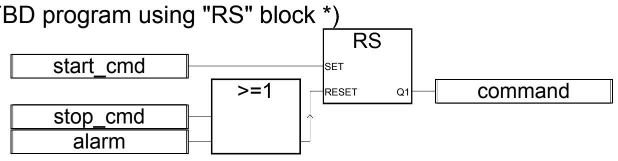
Argument:

SET	Boolean type	If TRUE, Q1 is set to TRUE.
RESET1	Boolean type	At TRUE, Q1 is reset to FALSE (priority).
Q1	Boolean type	Boolean type memory status

Explanation:
Reset–priority bistable multivibrator: Refer to table below.

Set	Reset1	Q1	result Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

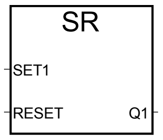
(* FBD program using "RS" block *)



(* Equivalent ST language: RS1 is the instance of the RS block. *)
RS1 (start_cmd, (stop_cmd OR alarm));
command := RS1.Q1;

(* Equivalent IL language: *)
```
LD      start_cmd
ST      RS1.set
LD      stop_cmd
OR      alarm
ST      RS1.reset1
CA      RS1
LD      RS1.Q1
ST      command
```

7.2.29 No.29 : SR (SR flip-flop)



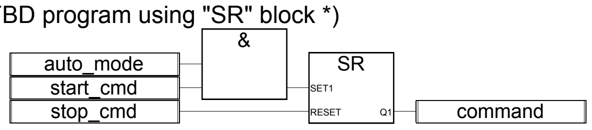
Argument:

SET1	Boolean type	If TRUE, Q1 is set to TRUE (priority).
RESET	Boolean type	If TRUE, Q1 is reset to FALSE.
Q1	Boolean type	Boolean type memory status

Explanation:
Set–priority bistable multivibrator: Refer to table below.

Set1	Reset	Q1	result Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

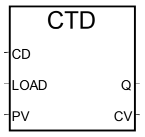
(* FBD program using "SR" block *)



```
(* Equivalent ST language: SR1 is the instance of the SR block. *)
SR1 ((auto_mode & start_cmd), stop_cmd);
command := SR1.Q1;

(* Equivalent IL language: *)
LD      auto_mode
AND     start_cmd
ST      SR1.set1
LD      stop_cmd
ST      SR1.reset
CAL     SR1
LD      SR1.Q1
ST      command
```

7.2.30 No.30 : CTD (countdown)



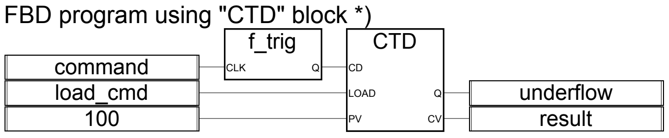
Argument:

CD	Boolean type	Count input (Countdown when CD is TRUE)
LOAD	Boolean type	Load command (priority) (CV = PV when LOAD is TRUE)
PV	Integer type	Initial value of counter
Q	Boolean type	Underflow TRUE when CV = 0
CV	Integer type	Counter result

CAUTION:
CTD does not detect the rise or fall of the input CD. The pulse counter must be created using "R_TRIG" or "F_TRIG" without fail.

Explanation:
The count (integer type) decreases in increments of 1 from the PV value down to zero.

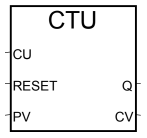
(* FBD program using "CTD" block *)



```
(* Equivalent ST language: F_TRIG1 is the instance of the F_TRIG block. CTD1 is the instance of the CTD block.*)
CTD1 (F_TRIG1(command),load_cmd,100);
underflow := CTD1.Q;
result := CTD1.CV;
```

```
(* Equivalent IL language: *)
LD      command
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ST      CTD1.cd
LD      load_cmd
ST      CTD1.load
LD      100
ST      CTD1.pv
CAL     CTD1
LD      CTD1.Q
ST      underflow
LD      CTD1.cv
ST      result
```

7.2.31 No.31 : CTU (count up)



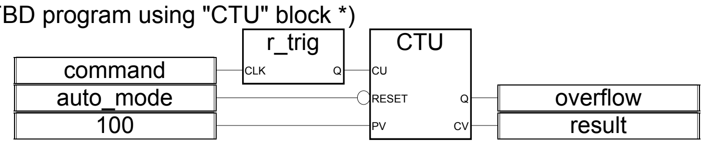
Argument:

CU	Boolean type	Count input (counting when CU is TRUE)
RESET	Boolean type	Reset command (priority)
PV	Integer type	Maximum count value
Q	Boolean type	Overflow: TRUE if CV = PV
CV	Integer type	Current count result

CAUTION:
CTU does not detect the rise or fall of the input CU. The pulse counter must be created using "R_TRIG" or "F_TRIG" without fail.

Explanation:
The count (integer type) increases in increments of 1 from zero.

(* FBD program using "CTU" block *)

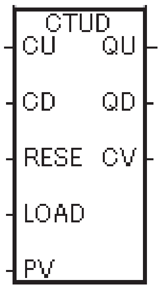


```
(* Equivalent ST language: R_TRIG1 is the instance of the R_TRIG block. CTU1 is the instance of the CTU block. *)
CTU1 (R_TRIG1(command), NOT (auto_mode),100);
overflow := CTU1.Q;
result := CTU1.CV;
```

```
(* Equivalent IL language: *)
LD      command
ST      R_TRIG1.clk
```

```
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTU1.cu
LDN     auto_mode
ST      CTU1.reset
LD      100
ST      CTU1.pv
CAL     CTU1
LD      CTU1.Q
ST      overflow
LD      CTU1.cv
ST      result
```

7.2.32 No.32：CTUD (up/down counter)



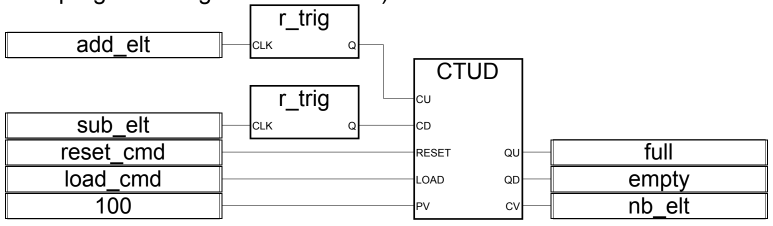
Argument:

CU	Boolean type	Up counter input (counting when CU is TRUE)
CD	Boolean type	Down counter input (counting when CD is TRUE)
RESET	Boolean type	Reset command (priority) (CV = 0 when RESET is TRUE)
LOAD	Boolean type	Load command (CV = PV when LOAD is TRUE)
PV	Integer type	Maximum count value
QU	Boolean type	Overflow: TRUE when CV = PV
QD	Boolean type	Underflow TRUE when CV = 0
CV	Integer type	Count result

CAUTION:
CTUD does not detect the rise or fall of the input CU or CD. The pulse counter must be created using "R_TRIG" or "F_TRIG" without fail.

Explanation:
The count (integer type) increases in increments of 1 from zero to the PV value or it decreases in increments of 1 from the current value down to zero.

(* FBD program using "CTUD" block *)



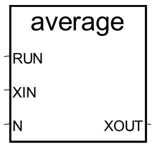
(* Equivalent ST language: R_TRIG1 and R_TRIG2 is the instance of the R_TRIG block. CTUD1 is the instance of the CTUD block. *)
CTUD1 (R_TRIG1(add_elt), R_TRIG2 (sub_elt), reset_cmd, load_cmd,100);
full := CTUD1.QU;
empty := CTUD1.QD;
nb_elt := CTUD1.CV;

(* Equivalent IL language: *)

```
LD      add_elt
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTUD1.cu
LD      sub_elt
```

```
ST      R_TRIG2.clk
CAL     R_TRIG2
LD      R_TRIG2.Q
ST      CTUD1.cd
LD      load_cmd
ST      CTUD1.load
LD      100
ST      CTUD1.pv
CAL     CTUD1
LD      CTUD1.QU
ST      full
LD      CTUD1.QD
ST      empty
LD      CTUD1.CV
ST      nb_elt
```

7.2.33 No.33：AVERAGE (average of N samples)



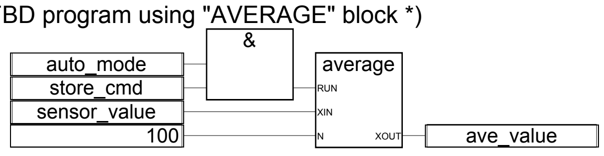
Argument:

RUN	Boolean type	TRUE ; Execution / FALSE ; Reset
XIN	Real number type	Real number value input
N	Integer type	Number of samples for obtaining the average (1 input per cycle)
XOUT	Real number type	Moving average value of sample number (N units) of XIN value

Explanation:
This command captures XIN in each cycle and calculates the average of the past sample values (sample number: N). The latest N–unit data is stored.

The maximum sample number **N** is 128. When the **"RUN"** command is **FALSE** (reset mode), the output value is the same as the input value.
When the stored sample number reaches N, the value stored first is cleared.

(* FBD program using "AVERAGE" block *)

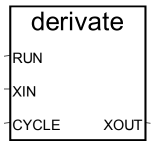


(* Equivalent ST language: AVERAGE1 is the instance of the AVERAGE block. *)
AVERAGE1 ((auto_mode & store_cmd), sensor_value, 100);
ave_value := AVERAGE1.XOUT;

(* Equivalent IL language: *)

```
LD      auto_mode
AND     store_cmd
ST      AVERAGE1.run
LD      sensor_value
ST      AVERAGE1.xin
LD      100
ST      AVERAGE1.N
CAL     AVERAGE1
LD      AVERAGE1.XOUT
ST      ave_value
```

7.2.34 No.34 : DERIVATE (time base-related derivationvalues))



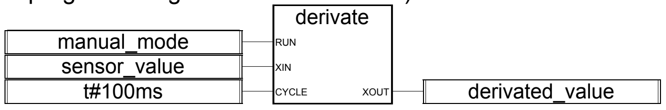
Argument:

RUN	Boolean type	If mode is TRUE, derivation execution; if FALSE, reset
XIN	Real number type	Input value
CYCLE	Timer type	Sampling period
XOUT	Real number type	Deviation result

Explanation:
This command derives real number values.

If the **"CYCLE"** parameter value is shorter than the ISaGRAF cycle time, the sampling interval is matched to the cycle time.

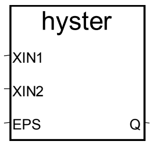
(* FBD program using "DERIVATE" block *)



```
(* Equivalent ST language: DERIVATE1 is the instance of the DERIVATE block. *)
DERIVATE1 (manual_mode, sensor_value, t#100ms);
derivated_value := DERIVATE1.XOUT;

(* Equivalent IL language: *)
LD      manual_mode
ST      DERIVATE1.run
LD      sensor_value
ST      DERIVATE1.XIN
LD      t#100ms
ST      DERIVATE1.CYCLE
CAL     DERIVATE1
LD      DERIVATE1.XOUT
ST      derivated_value
```

7.2.35 No. 35: HYSTER (Boolean type value for hysteresis of two real number values)

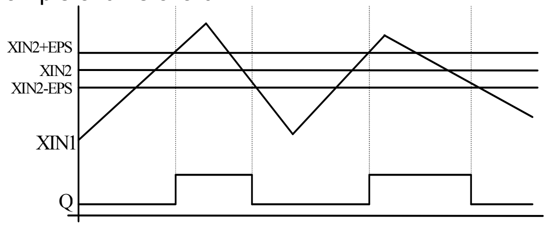


Argument:

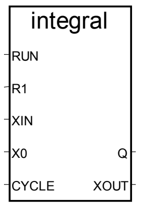
XIN1	Real number type	Real number
XIN2	Real number type	Real number (whether XIN1 has exceeded XIN2+EPS is tested)
EPS	Real number type	Hysteresis value (>0)
Q	Boolean type	TRUE when XIN1 is more than XIN2+EPS and not yet less than XIN–EPS

Explanation:
Hysteresis for upper limit level of real number value

Example of time chart:



7.2.36 No.36：INTEGRAL (time base-related interation)



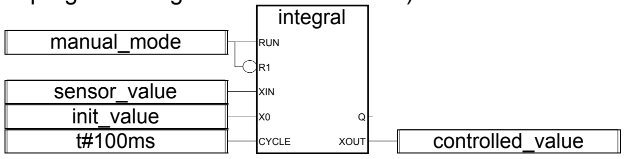
Argument:

RUN	Boolean type	If mode is TRUE, integration; if FALSE, hold
R1	Boolean type	Overwrite and reset
XIN	Real number type	Input value
X0	Real number type	Initial value
CYCLE	Timer type	Sampling period
Q	Boolean type	Reverse of R1
XOUT	Real number type	Integration result

Explanation:
This command integrates real number values.

For each specified sampling period, the result of multiplying the input (XIN) value by the time (in millisecond units) elapsed since the previous sample is added to the previous output, and this is then output from XOUT.
If the "CYCLE" parameter value is shorter than the ISaGRAF cycle time, the sampling interval is matched to the cycle time.

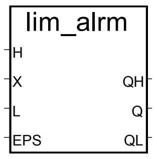
(* FBD program using "INTEGRAL" block *)



```
(* Equivalent ST language: INTEGRAL1 is the instance of the INTEGRAL block. *)
INTEGRAL1 (manual_mode, NOT (manual_mode), sensor_value, init_value, t#100ms);
controlled_value := INTEGRAL1.XOUT;
```

```
(* Equivalent IL language: *)
LD      manual_mode
ST      INTEGRAL1.run
STN     INTEGRAL1.R1
LD      sensor_value
ST      INTEGRAL1.XIN
LD      init_value
ST      INTEGRAL1.X0
LD      t#100ms
ST      INTEGRAL1.CYCLE
CAL     INTEGRAL1
LD      INTEGRAL1.XOUT
ST      controlled_value
```

7.2.37 No.37：LIM_ALRM (upper and lower limit alarms with hysteresis)



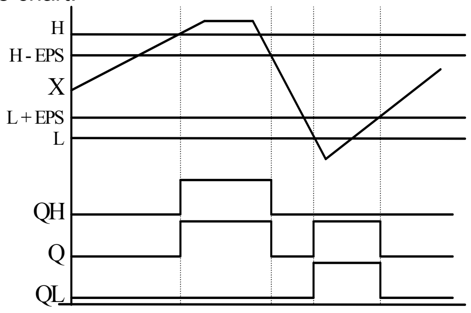
Argument:

H	Real number type	Upper limit setting
X	Real number type	Input value
L	Real number type	Lower setting
EPS	Real number type	Hysteresis value (>0)
QH	Boolean type	Upper limit alarm: TRUE if X has risen above upper limit value H
Q	Boolean type	Alarm: TRUE if the upper or lower limit has been exceeded
QL	Boolean type	Lower limit alarm: TRUE if X has fallen below lower limit value L

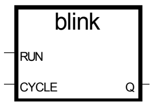
Explanation:
This command applies the hysteresis in respect of the upper or lower limits for real number values.

Hysteresis is applied to the upper or lower limit. The hysteresis delta value used for the upper and lower limits is one–half of the EPS parameter.

Time chart:



7.2.38 No.38：BLINK (blink Boolean type signal)

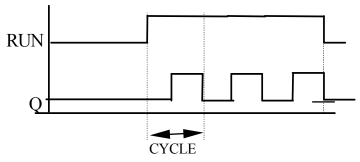


Argument:

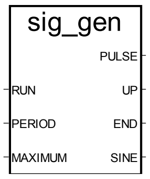
RUN	Boolean type	Mode: TRUE = blink; FALSE = output is reset
CYCLE	Timer type	Blink period
Q	Boolean type	Blink output signal

Explanation:
This command generates the blink signal.

Time chart:



7.2.39 No.39：SIG_GEN (sine signal generator)



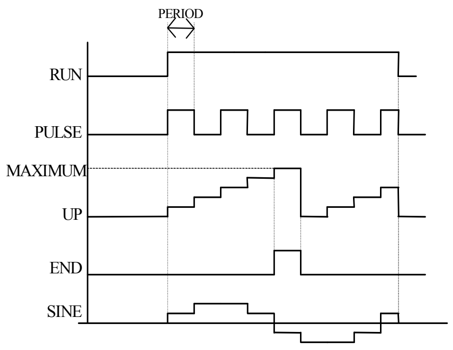
Argument:

RUN	Boolean type	Mode: TRUE = execution; FALSE = reset to FALSE
PERIOD	Timer type	Single sample time (pulse width)
MAXIMUM	Integer type	Maximum count value
PULSE	Boolean type	This reverses with each single sample time.
UP	Integer type	Up counter with each single sample time
END	Boolean type	TRUE when up counter has finished
SINE	Real number type	Sine wave signal (waveform half period = up counting time)

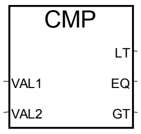
Explanation:
This command simultaneously generates various signals (Boolean square waves, integer type up counter, real number type sine waves).

When the counter reaches the maximum value, it is restarted from zero.The END signal is TRUE only during the time equivalent to one period.

Time chart:



7.2.40 No.40：CMP (comparison of function block)

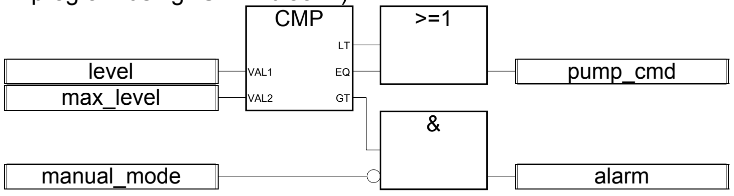


Argument:

VAL1	Integer type	Signed integer
VAL2	Integer type	Signed integer
LT	Boolean type	VAL1 < TRUE when VAL2
EQ	Boolean type	TRUE when VAL1 = VAL2
GT	Boolean type	VAL1 > TRUE when VAL2

Explanation:
This command compares two integer type values. It takes one of three results. (<, =, >)

(* FBD program using "CMP" block *)

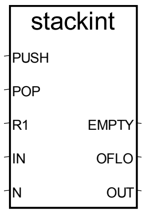


(* Equivalent ST language: CMP1 is the instance name of the CMP block. *)
CMP1 (level, max_level);

```
pump_cmd := CMP1.LT OR CMP1.EQ;
alarm := CMP1.GT AND NOT (manual_mode);

(* Equivalent IL language: *)
LD      level
ST      CMP1.val1
LD      max_level
ST      CMP1.val2
CAL     CMP1
LD      CMP1.LT
OR      CMP1.EQ
ST      pump_cmd
LD      CMP1.GT
ANDN    manual_mode
ST      alarm
```

7.2.41 No.41：STACKINT (integer stack)



Argument:

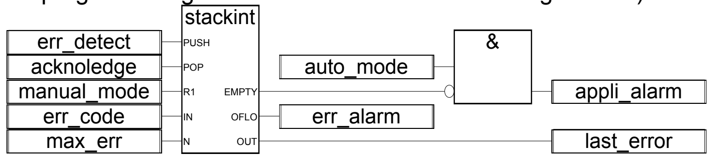
PUSH	Boolean type	Push command: The IN value is added to the top of the stack by the rise detection.
POP	Boolean type	Pop command: The head of the stack (last value pushed) is cleared by the rise detection.
R1	Boolean type	This resets the stack to the empty state.
IN	Integer type	Value to be pushed
N	Integer type	Stack size
EMPTY	Boolean type	TRUE when the stack is empty
OFLO	Boolean type	Overflow: TRUE when the stack is full
OUT	Integer type	Value at the head of the stack

Explanation:
This command manages the integer type value stack.

The STACKINT function block detects the rise for both the push and pop inputs. **The maximum size of the stack is 128.** Stack size **N must be a value between 1 and 128.**

The OFLO value is valid only after the stack has been reset. (In other words, R1 must be returned to FALSE after it has been set to TRUE.)

(* FBD program using "STACKINT" block : Error management *)



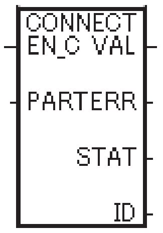
```
(* Equivalent ST language: STACKINT1 is the instance of the STACKINT block. *)
STACKINT1 (err_detect, acknoledge, manual_mode, err_code, max_err);
appli_alarm := auto_mode AND NOT (STACKINT1.EMPTY);
err_alarm := STACKINT1.OFLO;
last_error := STACKINT1.OUT;

(* Equivalent IL language: *)
LD      err_detect
ST      STACKINT1.push
LD      acknoledge
ST      STACKINT1.pop
LD      manual_mode
ST      STACKINT1.r1
LD      err_code
ST      STACKINT1.IN
LD      max_err
ST      STACKINT1.N
```

```
CAL     STACKINT1
LD      auto_mode
ANDN    STACKINT1.empty
ST      appli_alarm
LD      STACKINT1.OFLO
ST      err_alarm
LD      STACKINT1.OUT
ST      last_error
```

7.2.42 No.42：CONNECT (connection to resource)

This command cannot be used at the present time.



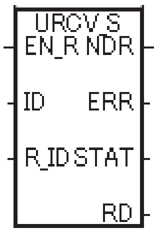
Argument:

EN_C	BOOL	This enables the connection.
PARTNER	STRING	Name of remote communication partner
VALID	BOOL	If TRUE, the connection ID is valid.
ERROR	BOOL	If TRUE, a new status which is not zero is received.
STATUS	DINT	Status detected last
ID	DINT	Communication channel identifier (ID)

Explanation:
This command connects with the remote or local resource (of the current or different project), and manages the exchange based on the USEND_S block and URCV_S block.
In this way, the communication channel identifier (ID) is created.
This identifier is required by all the other communication function blocks (URCV_S or USEND_S).

7.2.43 No.43：URCV_S (transmission of message to resource)

This command cannot be used at the present time.



Argument:

EN_R	BOOL	This enables the data to be received.
ID	DINT	Communication channel identifier
R_ID	STRING	Identifier of remote SFB inside channel
NDR	BOOL	If TRUE, a new character string is received at RD.
ERROR	BOOL	If TRUE, a new status which is not zero is received.
STATUS	DINT	Status detected last
RD	STRING	Character string which has been received

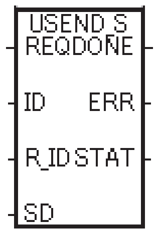
Explanation:
This command receives character strings from the remote or local resource (of the current or different project).

Note: In the current cycle, the "Connect" block must be called before "URCV_S" is called.

This function block receives the character string from one USEND_S instance. The character string received previously is now overwritten. When the character string is received properly, NDR is set to TRUE for one cycle. When an error is found, the ERROR output parameter is set to TRUE, and its status is set in the STATUS parameter.

7.2.44 No.44：USEND_S (reception of message from resource)

This command cannot be used at the present time.



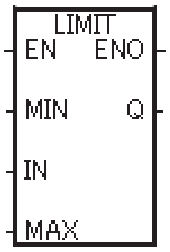
Argument:

REQ	BOOL	This transmits the request at the rising edge.
ID	DINT	Communication channel identifier
R_ID	STRING	Identifier of remote CFB inside channel
SD	TRING	Character string to be transmitted
DONE	BOOL	If TRUE, the operation is completed normally.
ERROR	BOOL	If TRUE, a new status which is not zero is received.
STATUS	DINT	Status detected last

Explanation:
This command transmits character strings to the remote or local resource (of the current or different project).

Note: In the current cycle, the "Connect" block must be called before "USEND_S" is called.
This CFB transmits the character string to one URCV_S at the rising edge of REQ. When the character string is transmitted correctly, DONE is set. When an error is found, the ERROR output parameter is set to TRUE, and its status is set in the STATUS parameter.

7.2.45 No.45：LIMIT (limit value)



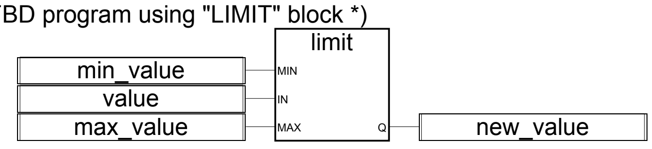
Argument:

MIN	Integer type	Minimum setting value
IN	Integer type	Integer input value
MAX	Integer type	Maximum setting value
Q	Integer type	The input values are kept within the range between the maximum and minimum for this.

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command sets the upper and lower limits for the input values. When the input value is between the maximum and minimum, it remains unchanged. If it is above the maximum, it is set to the maximum; conversely, if it is below the minimum, it is set to the minimum.

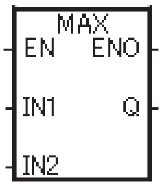
(* FBD program using "LIMIT" block *)



(* Equivalent ST language:*)
new_value := LIMIT (min_value, value, max_value);
(* The value is kept within the range between the maximum and minimum. *)

```
(* Equivalent IL language: *)
LD       min_value
LIMIT    value, max_value
ST       new_value
```

7.2.46 No.46 : MAX (maximum value)



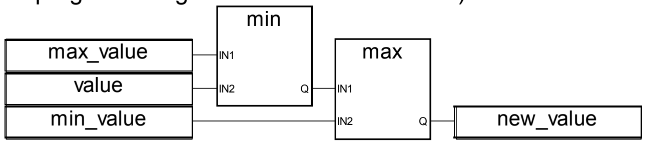
Argument:

IN1	Integer type	Integer value
IN2	Integer type	Integer value
Q	Integer type	Maximum value of two inputs

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
Maximum of two integer values

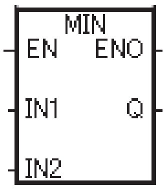
(* FBD program using "MIN" and "MAX" block *)



```
(* Equivalent ST language:*)
new_value := MAX (MIN (max_value, value), min_value);
(* The value is kept within the range between the maximum and minimum. *)
```

```
(* Equivalent IL language: *)
LD       max_value
MIN      value
MAX      min_value
ST       new_value
```

7.2.47 No.47 : MIN (minimum)



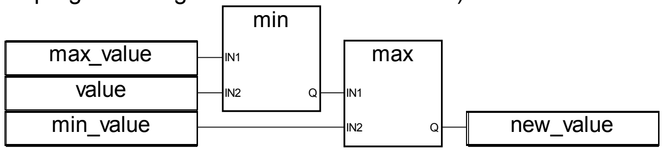
Argument:

IN1	Integer type	Integer value
IN2	Integer type	Integer value
Q	Integer type	Minimum value of two inputs

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This provides the minimum of two integer values.

(* FBD program using "MIN" and "MAX" block *)

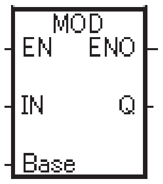


(* Equivalent ST language:*)
new_value := MAX (MIN (max_value, value), min_value);
(* The value is kept within the range between the maximum and minimum. *)

(* Equivalent IL language: *)

```
LD      max_value
MIN     value
MAX     min_value
ST      new_value
```

7.2.48 No.48 : MOD (residue calculation)



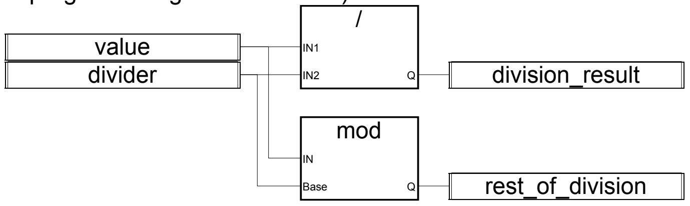
Argument:

IN	Integer type	Input value
Base	Integer type	Input value (greater than 0)
Q	Integer type	Residue calculation result –1 when base is equal to or less than 0

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command calculates the residues of integer values.

(* FBD program using "MOD" block *)

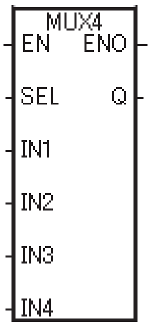


(* Equivalent ST language:*)
division_result := (value / divider); (* division of integer number*)
rest_of_division := MOD (value, divider); (* Remainder*)

(* Equivalent IL language: *)

```
LD      value
DIV     divider
ST      division_result
LD      value
MOD     divider
ST      rest_of_division
```

7.2.49 No.49：MUX4 (multiplexer 4)



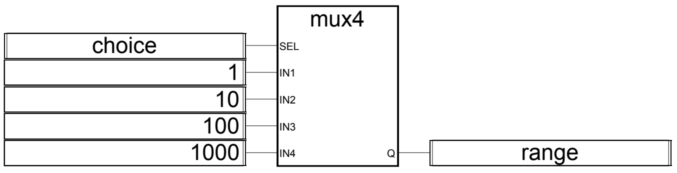
Argument:

SEL	Integer type	Select input value [0 to 3]
IN1...IN4	Integer type	Input value
Q	Integer type	IN1 when = SEL = 0 IN2 when = SEL = 1 IN3 when = SEL = 2 IN4 when = SEL = 3 = 0 when SEL is other than the above

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
Multiplexer for 4 inputs: This command selects one of the four integer input values.

(* FBD program using "MUX4" block *)

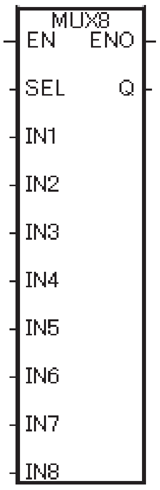


(* Equivalent ST language:*)
range := MUX4 (choice, 1, 10, 100, 1000);
(* One of the four inputs is selected. When "choice" is 1, the "range" is 10. *)

(* Equivalent IL language: *)

```
LD      choice
MUX4    1,10,100,1000
ST      range
```

7.2.50 No.50：MUX8 (multiplexer 8)



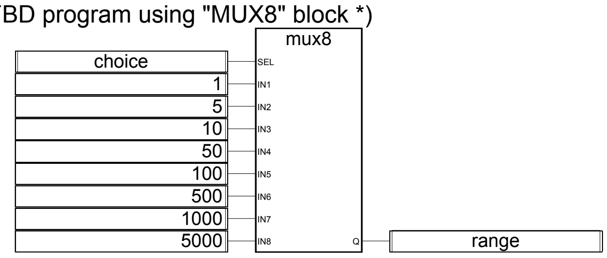
Argument:

SEL	Integer type	Select input value [0 to 7]
IN1 ... IN8	Integer type	Input value
Q	Integer type	IN1 when = SEL = 0
		IN2 when = SEL = 1
		IN3 when = SEL = 2
		.....
		IN8 when = SEL = 7
		= 0 when SEL is other than the above

The command is executed only when the EN input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
Multiplexer for 8 inputs: This command selects one of the four integer input values.

(* FBD program using "MUX8" block *)

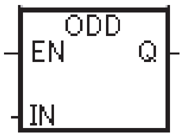


(* Equivalent ST language:*)
range := MUX8 (choice, 1, 5, 10, 50, 100, 500, 1000, 5000);
(* One of the 8 inputs is selected. When "choice" is 3, the "range" is 50. *)

(* Equivalent IL language: *)

```
LD      choice
MUX8    1,5,10,50,100,500,1000,5000
ST      range
```

7.2.51 No.51：ODD (odd parity)



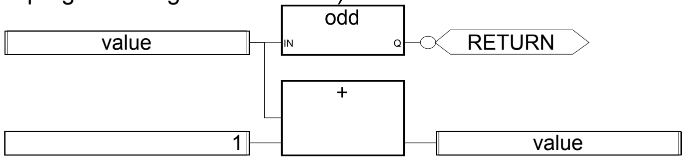
Argument:

IN	Integer type	Integer input
Q	Boolean type	TRUE: When the input value is an odd number FALSE: When the input value is an even number

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.

Explanation:
Parity check of integer value: This command determines whether the parity is odd or even.

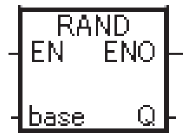
(* FBD program using "ODD" block *)



```
(* Equivalent ST language:*)
If Not (ODD (value)) Then Return; End_if;
value:= value +1;
(* "value" is always an even number.)
```

```
(* Equivalent IL language: *)
LD      value
ODD
RETNC
LD      value
ADD     1
ST      value
```

7.2.52 No.52：RAND (random value)



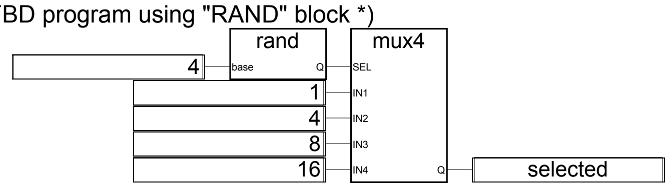
Argument:

base	Integer type	Maximum random value
Q	Integer type	Random value [0 to base –1]

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command provides random values in the range specified by the integer values.

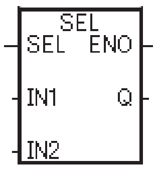
(* FBD program using "RAND" block *)



```
(* Equivalent ST language:*)
selected := MUX4 ( RAND (4), 1, 4, 8, 16 );
(*
Depending on the random value generated, any one of the four inputs is selected.
The numerical values generated by RAND are 0 to 3. The following is output to "selected" of MUX4 as the result.
1:      RAND output is 0.
4:      RAND output is 1.
8:      RAND output is 2.
16:     RAND output is 3.
*)

(* Equivalent IL language: *)
LD      4
RAND
MUX4    1.40.80.16
ST      selected
```

7.2.53 No.53 : SEL (binary selector)



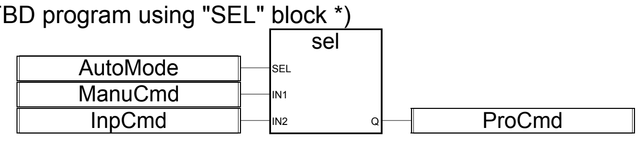
Argument:

SEL	Boolean type	For selection purposes
IN1, IN2	Integer type	Integer values
Q	Integer type	= IN1: When SEL is FALSE
		= IN2: When SEL is TRUE

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
Binary selection: This command selects one of two integer values.

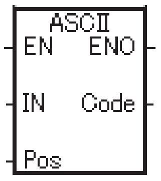
(* FBD program using "SEL" block *)



```
(* Equivalent ST language:*)
ProcCmd := SEL (AutoMode, ManuCmd, InpCmd);
(* Select the command to be processed. *)

(* Equivalent IL language: *)
LD      AutoMode
SEL     ManuCmd,InpCmd
ST      ProCmd
```

7.2.54 No.54：ASCII (characters → ASCII kode)



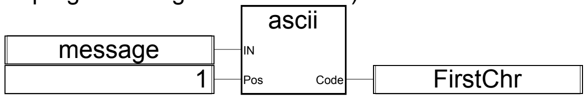
Argument:

IN	Character string type	Character string which is not blank
Pos	Integer type	Position in character string of character to be converted [1 to len] (where "len" is the length of character string IN)
Code	Integer type	ASCII code of character selected [0 to 255] 0 if "Pos" is longer than the length of the character string

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command provides ASCII codes for specified characters in character strings.

(* FBD program using "ASCII" block *)

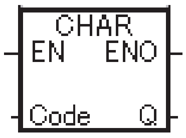


(* Equivalent ST language:*)
FirstChr := ASCII (message, 1);
(* "FirstChr" is the ASCII code of the first character in the character string. *)

(* Equivalent IL language: *)

```
LD      message
ASCII   1
ST      FirstChr
```

7.2.55 No.55：CHAR (ASCII code → characters)



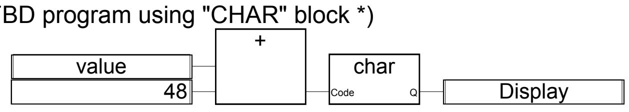
Argument:

Code	Integer type	ASCII codes [0 to 255]
Q	Character string type	Character string of one character The characters which correspond to ASCII codes ("Code") are used. ASCII codes which remain after division by 256 are used.

The command is executed only when the EN input is TRUE. When each scan is executed, connect directly to the bus.
The ENO output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command provides characters which correspond to the ASCII codes provided.

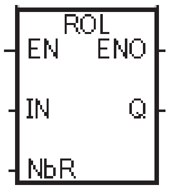
(* FBD program using "CHAR" block *)



(* Equivalent ST language:*)
Display := CHAR (value + 48);
(* value: 0 to 9*)
(* 48 is the ASCII code for '0'. *)

```
(* The result is a character from '0' to '9'. *)
(* Equivalent IL language: *)
LD      value
ADD     48
CHAR
ST      Display
```

7.2.56 No.56：ROL (leftward dotation)

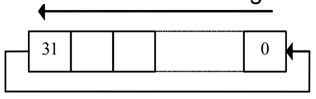


Argument:

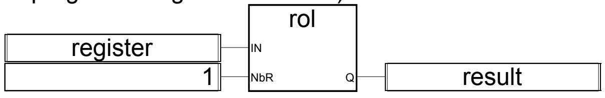
IN	Integer type	Integer value
NbR	Integer type	Number of bits by which the value is to be rotated [1 to 31]
Q	Integer type	Result of leftward rotation
		No change when nb_rotation is equal to or less than 0

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
Leftward bit rotation of integer value Integer values are rotated by 32 bits.



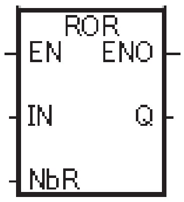
(* FBD program using "ROL" block *)



```
(* Equivalent ST language:*)
result := ROL (register, 1);
(* register = 2#0100_1101_0011_0101*)
(* result = 2#1001_1010_0110_1010*)

(* Equivalent IL language: *)
LD      register
ROL     1
ST      result
```

7.2.57 No.57：ROR (rightward rotation)

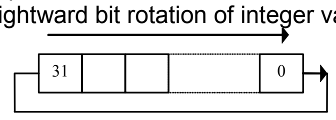


Argument:

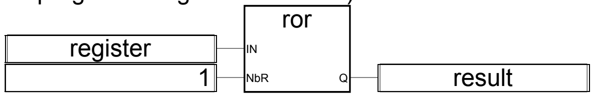
IN	Integer type	Integer value
NbR	Integer type	Number of bits by which the value is to be rotated [1 to 31]
Q	Integer type	Rotation result
		No change when nb_rotation is equal to or less than 0

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
Rightward bit rotation of integer value Integer values are rotated by 32 bits.



(* FBD program using "ROR" block *)

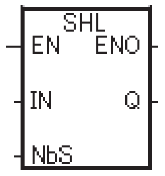


(* Equivalent ST language:*)
result := ROR (register, 1);
(* register = 2#0100_1101_0011_0101*)
(* result = 2#1010_0110_1001_1010 *)

(* Equivalent IL language: *)

LD	register
ROR	1
ST	result

7.2.58 No.58：SHL (leftward shift)

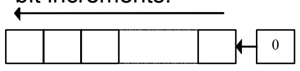


Argument:

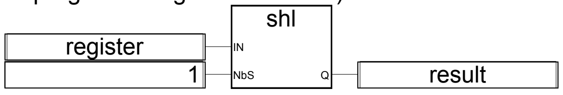
IN	Integer type	Integer value
NbS	Integer type	Bit shift number [1 to 31]
Q	Integer type	Result of leftward shift No change when nb_shift is equal to or less than 0. 0 is inserted into the lowest bit.

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command performs leftward shifts using the bit expressions of the integer values. The shift is performed in 32– bit increments.



(* FBD program using "SHL" block *)

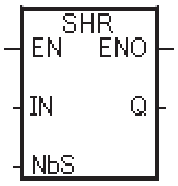


(* Equivalent ST language:*)
result := SHL (register, 1);
(* register = 2#0100_1101_0011_0101*)
(* result = 2#1001_1010_0110_1010*)

(* Equivalent IL language: *)

LD	register
SHL	1
ST	result

7.2.59 No.59：SHR (rightward shift)

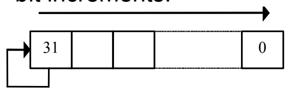


Argument:

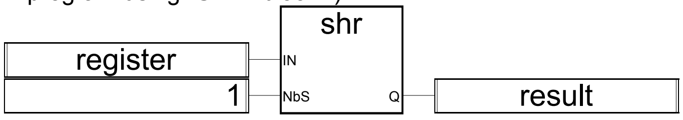
IN	Integer type	Integer value
NbS	Integer type	Bit shift number [1 to 31]
Q	Integer type	Result of rightward shift No change when nb_shift is equal to or less than 0. The highest bit remains unchanged.

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command performs leftward shifts using the bit expressions of the integer values. The shift is performed in 32– bit increments.



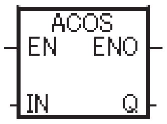
(* FBD program using "SHR" block *)



```
(* Equivalent ST language:*)
result := SHR (register, 1);
(* register = 2#1100_1101_0011_0101*)
(* result = 2#1110_0110_1001_1010 *)

(* Equivalent IL language: *)
LD      register
SHR     1
ST      result
```

7.2.60 No.60：ACOS (arc cosine)



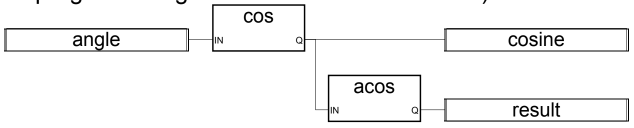
Argument:

IN	Real number type	This must be from –1.0 to +1.0.
Q	Real number type	Arc cosine of input value [0.0 to π] = 0.0: When the input value is illegal

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command calculates the arc cosines of real number values.

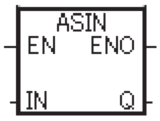
(* FBD program using "COS" and "ACOS" block *)



```
(* Equivalent ST language:*)
cosine := COS (angle);
result := ACOS (cosine); (* Result is equal to angle. *)

(* Equivalent IL language: *)
LD      angle
COS
ST      cosine
ACOS
ST      result
```

7.2.61 No.61 : ASIN (arc sine)



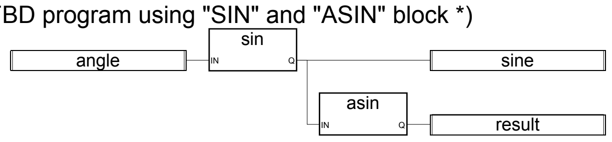
Argument:

IN	Real number type	This must be from –1.0 to +1.0.
Q	Real number type	Arc cosine of input value [– π /2 to + π /2]
		= 0.0: When the input value is illegal

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command calculates the arc sines of real number values.

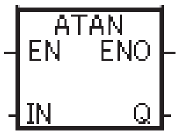
(* FBD program using "SIN" and "ASIN" block *)



```
(* Equivalent ST language:*)
sine := SIN (angle);
result := ASIN (sine); (* Result is equal to angle. *)

(* Equivalent IL language: *)
LD      angle
SIN
ST      sine
ASIN
ST      result
```

7.2.62 No.62 : ATAN (arc tangent)



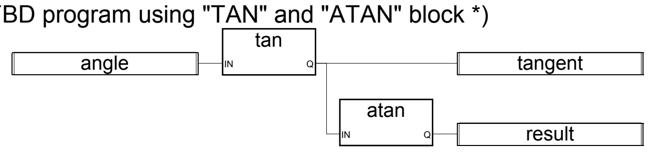
Argument:

IN	Real number type	Real number value
Q	Real number type	Arc tangent of input value [– π /2 to + π /2]
		= 0.0: When the input value is illegal

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command calculates the arc tangents of real number values.

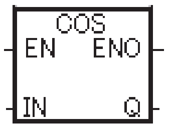
(* FBD program using "TAN" and "ATAN" block *)



(* Equivalent ST language:*)
tangent:= TAN (angle);
result := ATAN (tangent); (* Result is equal to angle. *)

(* Equivalent IL language: *)
```
LD      angle
TAN
ST      tangent
ATAN
ST      result
```

7.2.63 No.63 : COS (cosine)

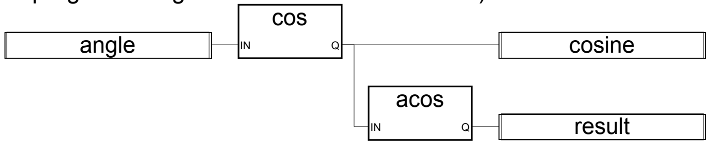


Argument:

IN	Real number type	Real number value
Q	Real number type	Cosine of input value [–1.0 to +1.0]

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

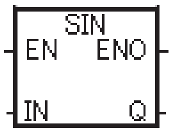
Explanation:
This command calculates the cosines of real number values.
(* FBD program using "COS" and "ACOS" block *)



(* Equivalent ST language:*)
cosine := COS (angle);
result := ACOS (cosine); (* Result is equal to angle. *)

(* Equivalent IL language: *)
```
LD      angle
COS
ST      cosine
ACOS
ST      result
```

7.2.64 No.64 : SIN (sine)



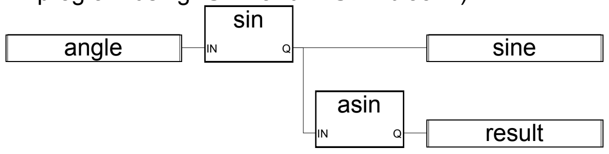
Argument:

IN	Real number type	Real number value
Q	Real number type	Sine of input value [–1.0 to +1.0]

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command calculates the sines of real number values.

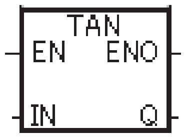
(* FBD program using "SIN" and "ASIN" block *)



```
(* Equivalent ST language:*)
sine := SIN (angle);
result := ASIN (sine); (* Result is equal to angle. *)
```

```
(* Equivalent IL language: *)
LD      angle
SIN
ST      sine
ASIN
ST      result
```

7.2.65 No.65：TAN (tangent)



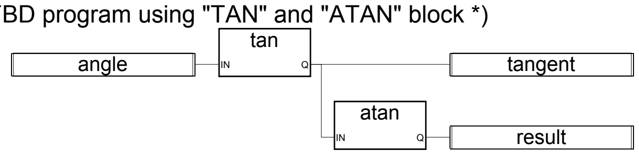
Argument:

IN	Real number type	The remainder after division by π must not be π/2.
Q	Real number type	Tangent of input value For an illegal input, 1E+38

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command calculates the tangents of real number values.

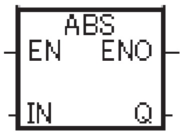
(* FBD program using "TAN" and "ATAN" block *)



```
(* Equivalent ST language:*)
tangent:= TAN (angle);
result:= ATAN (tangent); (* Result is equal to angle. *)
```

```
(* Equivalent IL language: *)
LD      angle
TAN
ST      tangent
ATAN
ST      result
```

7.2.66 No.66：ABS (absolute value)



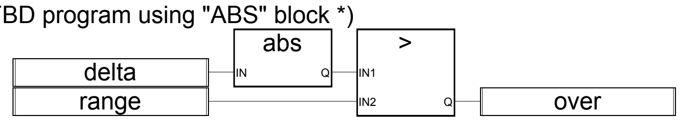
Argument:

IN	Real number type	Real number value input
Q	Real number type	Absolute value (always 0 or more)

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command provides the absolute values of real numbers.

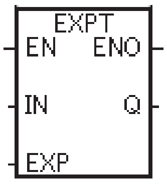
(* FBD program using "ABS" block *)



(* Equivalent ST language: *)
over := (ABS (delta) > range);

(* Equivalent IL language: *)
```
LD      delta
ABS
GT      range
ST      over
```

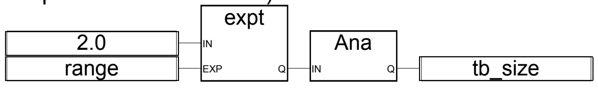
7.2.67 No.67：EXPT (exponential)



Argument:
IN Real number type Signed real number
EXP Integer type Integer (exponential part)
Q Real number type (IN ^{EXP})
The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command provides the arithmetic results of (base ^{exponent}) in real numbers.'base' is the integer used in the first argument, and 'exponent' is the integer used in the second argument.

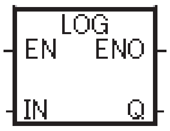
(*Example of "EXPT" block *)



(* Equivalent ST language:*)
tb_size := ANA (EXPT (2.0, range));

(* Equivalent IL language: *)
```
LD      2.0
EXPT    range
ANA
ST      tb_size
```

7.2.68 No.68：LOG (common logarithm)



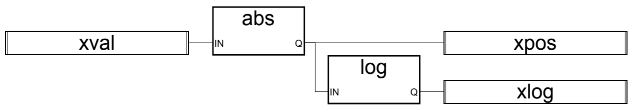
Argument:

IN	Real number type	Real number value greater than 0
Q	Real number type	Common logarithm of input value (with base of 10)

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command provides the common logarithms of real number inputs.

(* FBD program using "LOG" block *)

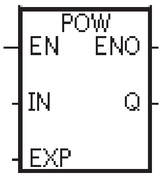


(* Equivalent ST language: *)
xpos := ABS (xval);
xlog := LOG (xpos);

(* Equivalent IL language: *)

```
LD      xval
ABS
ST      xpos
LOG
ST      xlog
```

7.2.69 No.69：POW (power calculation)



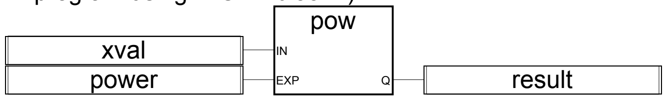
Argument:

IN	Real number type	Real number value (non–operand)
EXP	Real number type	Real number value (power)
Q	Real number type	(IN ^{EXP})
		1.0 : IN<> 0.0 if EXP= 0.0
		0.0 : IN= 0.0 if EXP < 0.0
		0.0 : IN =0.0 if EXP = 0.0
		0.0 : If IN < 0

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command provides the powers (base ^{exponent}) of real numbers.'base' is the real number used in the first argument, and 'exponent' is the real number used in the second argument.

(* FBD program using "POW" block *)

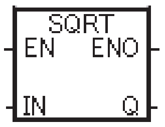


(* Equivalent ST language: *)
result := POW (xval, power);

(* Equivalent IL language: *)

```
LD      xval
POW     power
ST      result
```

7.2.70 No.70：SQRT (square root)



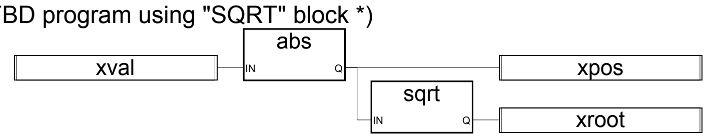
Argument:

IN	Real number type	Real number value (>= 0)
Q	Real number type	Square root of input value

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command calculates the square roots of real number values.

(* FBD program using "SQRT" block *)

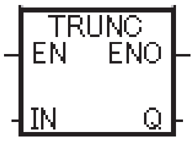


(* Equivalent ST language: *)
xpos := ABS (xval);
xroot := SQRT (xpos);

(* Equivalent IL language: *)

```
LD      xval
ABS
ST      xpos
SQRT
ST      xrout
```

7.2.71 No.71：TRUNC (truncation of decimal places)



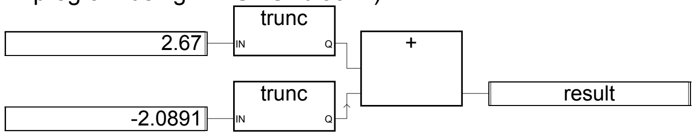
Argument:

IN	Real number type	Real number value
Q	Real number type	IN>0, maximum integer value less than input value IN<0, minimum integer value greater than input value

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command provides the integer parts of real number values.

(* FBD program using "TRUNC" block *)

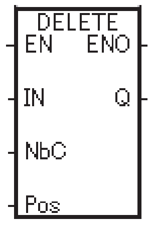


(* Equivalent ST language: *)
result:= TRUNC (+2.67) + TRUNC (–2.0891);
(* significance: result:= 2.0 + (–2.0) := 0.0; *)

(* Equivalent IL language: *)

```
LD        2.67
TRUNC
ST        temporary (* First TRUNC result *)
LD        –2.0891
TRUNC
ADD       temporary
ST        result
```

7.2.72 No.72：DELETE (delection of character strings)



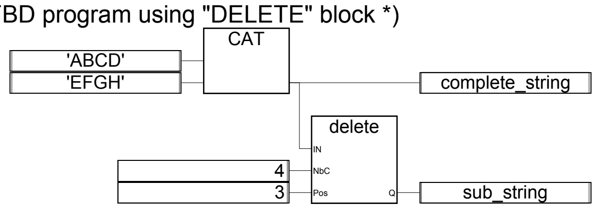
Argument:

IN	Character string type	Character string which is not blank
NbC	Integer type	Number of characters to be deleted from character string
Pos	Integer type	Position of first character string to be deleted ("1" is the position at the head of the character string.)
Q	Character string type	Character string resulting from deletion of specified character string When Pos < 1, empty character string When Pos > IN character string, first character When NbC <=0, first character

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command deletes parts of character strings.

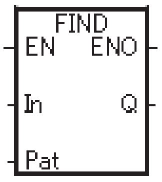
(* FBD program using "DELETE" block *)



(* Equivalent ST language: *)
complete_string := 'ABCD' + 'EFGH'; (* complete_string is 'ABCDEFGH' *)
sub_string := DELETE (complete_string, 4, 3); (* sub_string is 'ABGH' *)

```
(* Equivalent IL language: *)
LD          'ABCD'
ADD         'EFGH'
ST          complete_string
DELETE      4,3
ST          sub_string
```

7.2.73 No.73 : FIND (character string search)



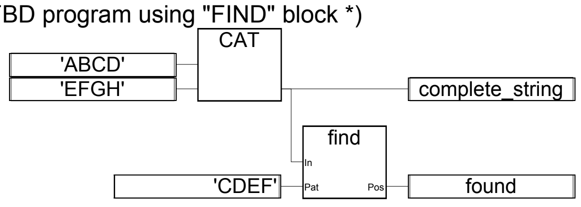
Argument:

In	Character string type	Input character string
Pat	Character string type	Character string pattern to be searched Must not be blank.
Pos	integer type	= 0: When the character string pattern cannot be found = Position where the character string searched was found first ("1" if first character) **Differentiation is made between upper– and lower–case characters.**

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command searches the specified character string from among the character strings available, and if that string is found, it returns the position of the string.

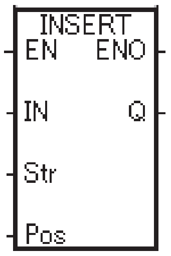
(* FBD program using "FIND" block *)



(* Equivalent ST language: *)
complete_string := 'ABCD' + 'EFGH'; (* complete_string is 'ABCDEFGH' *)
found := FIND (complete_string, 'CDEF'); (* found is 3 *)

```
(* Equivalent IL language: *)
LD      'ABCD'
ADD     'EFGH'
ST      complete_string
FIND    'CDEF'
ST      found
```

7.2.74 No.74 : INSERT (insertion of character string)



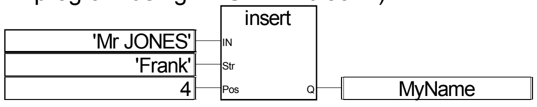
Argument:

IN	Character string type	Original character string
Str	Character string type	Character string to be inserted
Pos	Integer type	Insertion position The character string to be inserted is provided before the number." 1" is the position at the head of the character string.
Q	Character string type	Character string resulting from insertion When Pos <=0, empty character string When "Pos" exceeds the length of character string IN, this becomes two character strings joined together.

The command is executed only when the EN input is TRUE. When each scan is executed, connect directly to the bus.
The ENO output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command inserts another character string at the position specified in one character string.

(* FBD program using "INSERT" block *)

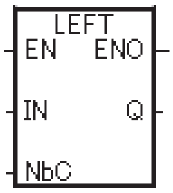


(* Equivalent ST language: *)
MyName := INSERT ('Mr JONES', 'Frank', 4);
(* MyName : 'Mr Frank JONES". *)

(* Equivalent IL language: *)

```
LD              'Mr JONES'
INSERT          'Frank',4
ST              MyName
```

7.2.75 No.75 : LEFT (acquisition of left part of chatacter string)



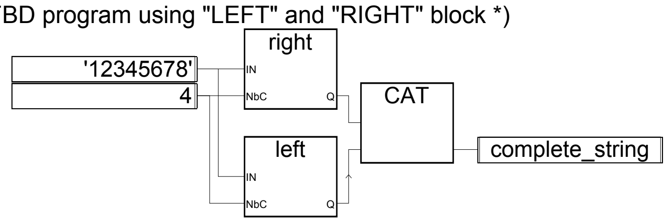
Argument:

IN	Character string type	Character string which is not blank
NbC	Integer type	Number of characters to be taken out (This must be less than the length of character string IN.)
Q	Character string type	Left–side character string of character string IN (character string length; NbC) When Nbc <=0, empty character string. When NbC >= IN length, entire character string IN

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command takes out only the specified number of characters from the left side of the original character string.

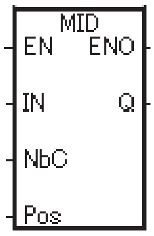
(* FBD program using "LEFT" and "RIGHT" block *)



```
(* Equivalent ST language: *)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string : '56781234'
Character string taken out by RIGHT: '5678'
Character string taken out by LEFT: '1234'
*)
```

```
(* Equivalent IL language: This calls LEFT first. *)
LD      '12345678'
LEFT    4
ST      sub_string (* Intermediate result *)
LD      '12345678'
RIGHT   4
ADD     sub_string
ST      complete_string
```

7.2.76 No.76：MID (acquisition of middle part of character string)



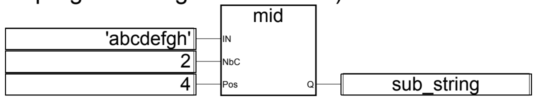
Argument:

IN	Character string type	Character string which is not blank
NbC	Integer type	Number of characters to be taken out (This must be less than the length of character string IN.)
Pos	Integer type	Position where characters are taken out ("1" for head position)
Q	Character string type	Character string which has been taken out (character string length: NbC) Empty character string if specified argument is illegal

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command takes out character strings with the specified lengths from the specified positions.

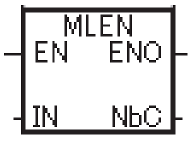
(* FBD program using "MID" block *)



```
(* Equivalent ST language: *)
sub_string := MID ('abcdefgh', 2, 4);
(* sub_string : 'de' *)
```

```
(* Equivalent IL language: *)
LD      'abcdefgh'
MID     2,4
ST      sub_string
```

7.2.77 No.77：MLEN (acquisition of character string length)



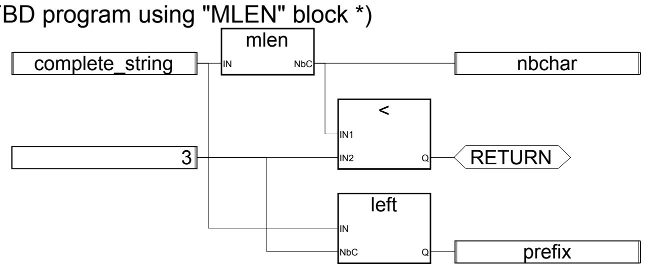
Argument:

IN	Character string type	Character string
NbC	Integer type	Character string length of character string IN

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command calculates the character string lengths of character strings.

(* FBD program using "MLEN" block *)

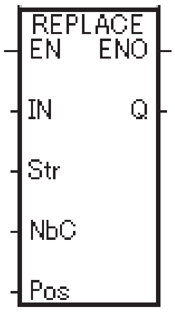


(* Equivalent ST language: *)
nbchar := MLEN (complete_string);
If (nbchar < 3) Then Return; End_if;
prefix := LEFT (complete_string, 3);
(* This program extracts from the character string three characters on the left side, and it inserts them into the prefix character string. Nothing is done if the character string length is under three characters. *)

(* Equivalent IL language: *)

```
LD      complete_string
MLEN
ST      nbchar
LT      3
RETC
LD      complete_string
LEFT    3
ST      prefix
```

7.2.78 No.78：REPLACE (character string replacement)



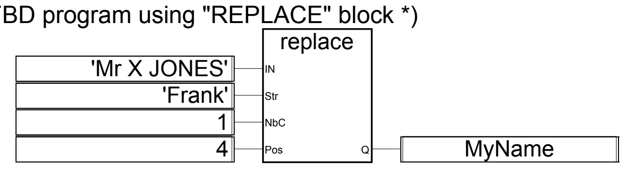
Argument:

IN	Character string type	Original character string
Str	Character string type	Character string to be inserted (the characters equivalent to NbC are replaced)
NbC	Integer type	Number of characters to be deleted
Pos	Integer type	Position of the first character which is to be replaced ("1" for head position)
Q	Character string type	Character string resulting from replacement – First, the characters equivalent to NbC are deleted from the "Pos" position. – Next, character string "Str" is inserted at this position. When Pos <=0, empty character string When Pos = >character string IN length, this becomes two character strings (IN + Str) joined together. When NbC <= 0, it becomes the original character string.

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command replaces a part of one character string with another character string.

(* FBD program using "REPLACE" block *)

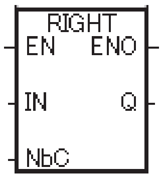


(* Equivalent ST language:*)
MyName := REPLACE ('Mr X JONES, 'Frank', 1, 4);
(* MyName becomes 'Mr Frank JONES". *)

(* Equivalent IL language: *)

```
LD              'Mr X JONES'
REPLACE         'Frank',1,4
ST              MyName
```

7.2.79 No.79：RIGHT (acquisition of right part of character string)



Argument:

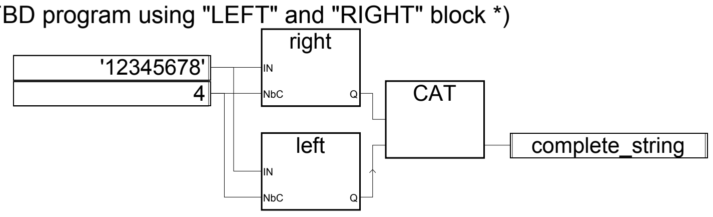
IN	Character string type	Character string which is not blank
NbC	Integer type	Number of characters to be taken out (less than character string IN length)
Q	Character string type	Right–side character string of character string IN (character string length; NbC) When Nbc <=0, empty character string. When NbC >= IN character string length, entire character string IN

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to

the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command takes out only the specified number of characters from the right side of the character string.

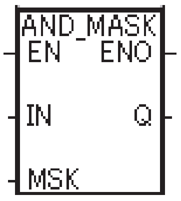
(* FBD program using "LEFT" and "RIGHT" block *)



```
(* Equivalent ST language: *)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string is '56781234'
Character string extracted by RIGHT: '5678'
Character string extracted by LEFT: '1234'*)
```

```
(* Equivalent IL language: First done is call to LEFT *)
LD '     12345678'
LEFT    4
ST      sub_string (* Intermediate result *)
LD      '12345678'
RIGHT   4
ADD     sub_string
ST      complete_string
```

7.2.80 No.80：AND_MASK (AND mask for each interger bit)



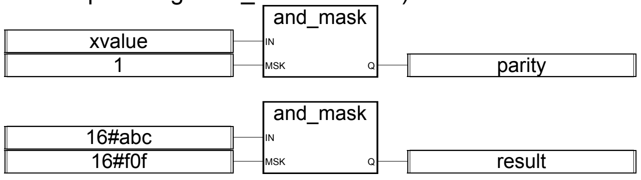
Argument:

IN	Integer type	
MSK	Integer type	
Q	Integer type	Logical product (AND) for each IN bit and MSK bit

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This is the logical product command of integer type variables which ANDs each of the IN bits and each of the MSK bits.

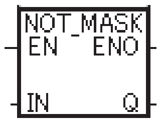
(* FBD example using AND_MASK block *)



```
(* Equivalent ST language: *)
parity := AND_MASK (xvalue, 1); (* This is "1" if "xvalue" is an odd number. *)
result := AND_MASK (16#abc, 16#f0f); (* it become 16#a0c*)
```

```
(* Equivalent IL language: *)
LD            xvalue
AND_MASK      1
ST            parity
LD            16#abc
AND_MASK      16#f0f
ST            result
```

7.2.81 No.81：NOT_MASK (NOT for each integer bit)



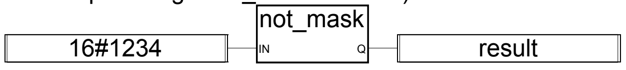
Argument:

IN	Integer type	
Q	Integer type	Reversal of each bit in IN which is expressed with 32 bits

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command reverses each of the bits in integer type variables.

(* FBD example using NOT_MASK block *)

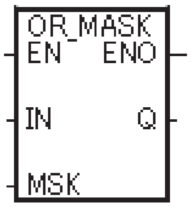


(* Equivalent ST language: *)
result := NOT_MASK (16#1234);
(* result is 16#FFFF_EDCB *)

(* Equivalent IL language: *)

```
LD              16#1234
NOT_MASK
ST              result
```

7.2.82 No.82：OR_MASK (OR mask for each interger bit)



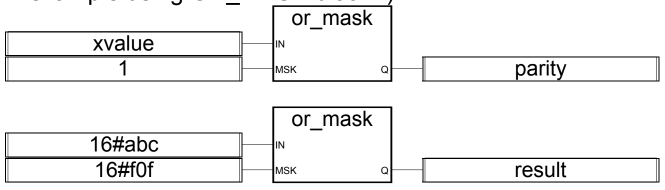
Argument:

IN	Integer type	
MSK	Integer type	
Q	Integer type	Logical sum (OR) of each IN bit and MSK bit

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This is the logical sum command for integer type variables which ORs each of the IN bits and each of the MSK bits.

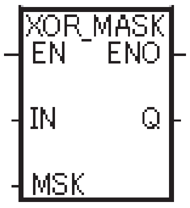
(* FBD example using OR_MASK block *)



(* Equivalent ST language: *)
is_odd := OR_MASK (xvalue, 1); (* This must be an odd number. *)
result := OR_MASK (16#abc, 16#f0f); (* it becomes 16#fbf *)

```
(* Equivalent IL language: *)
LD            xvalue
OR_MASK       1
ST            is_odd
LD            16#abc
OR_MASK       16#f0f
ST            result
```

7.2.83 No.83：XOR_MASK (XOR mask for each interger bit)



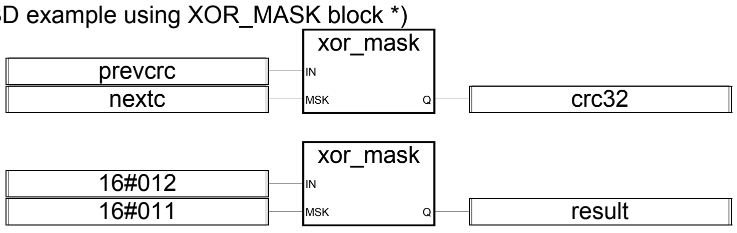
Argument:

IN	Integer type	
MSK	Integer type	
Q	Integer type	Exclusive OR for each IN bit and MSK bit

The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This is the exclusive OR command for integer type variable which exclusively ORs each of the IN bits and each of the MSK bits.

(* FBD example using XOR_MASK block *)

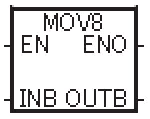


```
(* Equivalent ST language: *)
crc32 := XOR_MASK (prevcrc, nextc);
result := XOR_MASK (16#012, 16#011); (* it becomes 16#003 *)
```

```
(* Equivalent IL language: *)
LD            prevcrc
XOR_MASK      nextc
ST            crc32
LD            16#012
XOR_MASK      16#011
ST            result
```

7.2.84 No.84：MOV8 (movement of 8 Boolean data)

This command is unique to this controller.

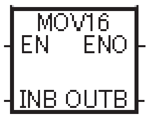


Argument:
INB Boolean type
OUTB Boolean type
The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes the values of 8 consecutive BOOL variables with INB at the head for the 8 BOOL variables with OUTB at the head.

7.2.85 No.85：MOV16 (movement of 16 Boolean data)

This command is unique to this controller.

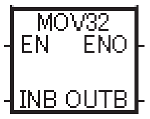


Argument:
INB Boolean type
OUTB Boolean type
The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes the values of 16 consecutive BOOL variables with INB at the head for the 16 BOOL variables with OUTB at the head.

7.2.86 No.86：MOV32 (movement of 32 Boolean data)

This command is unique to this controller.

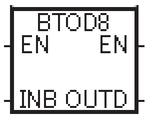


Argument:
INB Boolean type
OUTB Boolean type
The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes the values of 32 consecutive BOOL variables with INB at the head for the 32 BOOL variables with OUTB at the head.

7.2.87 No.87：BTOD8 (movement of 8 Boolean data to integer variables)

This command is unique to this controller.

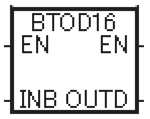


Argument:
INB Boolean type
OUTD Integer type
The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes the values of 8 consecutive BOOL variables with INB at the head for the integer variables indicated by OUTD, and it substitutes INB for LSB.

7.2.88 No.88：BTOD16 (movement of 16 Boolean data to integer variables)

This command is unique to this controller.

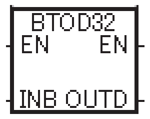


Argument:
INB Boolean type
OUTD Integer type
The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes the values of 16 consecutive BOOL variables with INB at the head for the integer variables indicated by OUTD, and it substitutes INB for LSB.

7.2.89 No.89：BTOD32 (movement of 32 Boolean data to integer variables)

This command is unique to this controller.

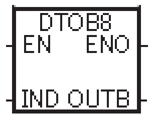


Argument:
INB Boolean type
OUTD Integer type
The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes the values of 8 consecutive BOOL variables with INB at the head for the integer variables indicated by OUTD, and it substitutes INB for LSB.

7.2.90 No.90：DTOB8 (substitution of 8 interger variables for BOOL variables)

This command is unique to this controller.

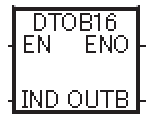


Argument:
IND Integer type
OUTB Boolean type
The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes the lower 8 bit values of the variable indicated by IND for 8 BOOL variables with OUTB at the head, and it substitutes OUTB for LSB.

7.2.91 No.91：DTOB16 (substitution of 16 integer variables for BOOL variables)

This command is unique to this controller.

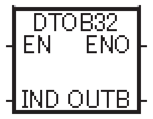


Argument:
IND Integer type
OUTB Boolean type
The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes the lower 16 bit values of the variable indicated by IND for 16 BOOL variables with OUTB at the head, and it substitutes OUTB for LSB.

7.2.92 No.92：DTOB32 (substitution of 32 integer variables for BOOL variables)

This command is unique to this controller.



Argument:
IND Integer type
OUTB Boolean type
The command is executed only when the **EN** input is TRUE. When each scan is executed, connect directly to the bus.
The **ENO** output always has the same status as the first input of the block. Connect the BOOL variable.

Explanation:
This command substitutes the values of the variable indicated by IND for 32 BOOL variables with OUTB at the head, and it substitutes OUTB for LSB.

Chapter 8 Troubleshooting

This chapter describes the troubleshooting.

8.1 Troubleshooting

IMPORTANT

When the following errors are detected, the Software PLC (Built-in PLC) scanning will stop because the robot controller regards that there are problems in the environment of the software PLC function or the ladder programs. Please check those and restart the software PLC scanning after fixing the problems.

Error

E367 The error was detected by Built-in PLC

[Cause]	It becomes this error when Built-in PLC detects an error.
[Remedy]	Check the environment of Built-in PLC of operation.

Error

E550 It is scan time over of Built-in PLC.

[Cause]	It detects, when the scanning time of Built-in PLC is too long.
[Remedy]	Change the "PLC scan time" or correct the Ladder program to shorten the scan time.

POINT

If the "E177 Primary power supply fault." occurs with the E550, the reason of the scan time over may be the primary power shut down.

Alarm

E2367 Scan of Built-in PLC has stopped.

[Cause]	Although Built-in PLC status is except <Disabled>, when scan of Built-in PLC has stopped, it becomes this error at the time of robot operation.
[Remedy]	Built-in PLC status is set as <RUN> by constant mode, and scan is started.

NOTE

NACHI
NACHI-FUJIKOSHI CORP.

http://www.nachi-fujikoshi.co.jp/

Japan Main Office	Phone: +81-3-5568-5245	Fax: +81-3-5568-5236	Shiodome Sumitomo Bldg. 17F, 1-9-2 Higashi-Shinbashi Minato-ku, TOKYO, 105-0021 JAPAN
Nachi Robotic Systems Inc. (NRS)		**http://www.nachirobotics.com/**	
North America Headquarters	Phone: 248-305-6545	Fax: 248-305-6542	42775 W. 9 Mile Rd. Novi, Michigan 48375, U.S.A
Indiana Service Center	Phone: 248-305-6545	Fax: 248-305-6542	Greenwood, Indiana
Ohio Service Center	Phone: 248-305-6545	Fax: 248-305-6542	Cincinnati, Ohio
South Carolina Service Center	Phone: 248-305-6545	Fax: 248-305-6542	Greenville, South Carolina
Canada Branch Office	Phone: 905-760-9542	Fax: 905-760-9477	89 Courtland Ave., Unit No.2, Concord, Ontario, L4K 3T4, CANADA
Mexico Branch Office	Phone :+52-555312-6556	Fax:+52-55-5312-7248	Urbina No.54, Parque Industrial Naucalpan, Naucalpan de Juarez, Estado de Mexico C.P. 53489, MEXICO
NACHI EUROPE GmbH		**http://www.nachi.de/**	
Central Office Germany	Phone: +49-2151-65046-0	Fax: +49-2151-65046-90	Bischofstrasse 99, 47809, Krefeld, GERMANY
U.K. branch	Phone: +44-0121-423-5000	Fax: +44-0121-421-7520	Unit 3, 92, Kettles Wood Drive, Woodgate Business Park, Birmingham B32 3DB, U.K.
Czech branch	Phone: + 420-255-734-000	Fax: +420-255-734-001	Obchodni 132, 251 01 Cestlice, PRAGUE-EAST CZECH REPUBLIC
NACHI AUSTRALIA PTY. LTD.		**http://www.nachi.com.au/**	
Robotic Division & Victoria office	Phone: +61-(0)3-9796-4144	Fax: +61-(0)3-9796-3899	38, Melverton Drive, Hallam, Victoria 3803, , AUSTRALIA
Sydney office	Phone: +61-(0)2-9898-1511	Fax: +61-(0)2-9898-1678	Unit 1, 23-29 South Street, Rydalmere, N.S.W, 2116, AUSTRALIA
Brisbane office	Phone: +61-(0)7-3272-4714	Fax: +61-(0)7-3272-5324	7/96 Gardens Dr,Willawong,QLD 4110, , AUSTRALIA
NACHI SHANGHAI CO., LTD.		**http://www.nachi.com.cn/**	
Shanghai office	Phone: +86-(0)21-6915-2200	Fax: +86-(0)21-6915-2200	11F Royal Wealth Centre, No.7 Lane 98 Danba Road Putuo District, Shanghai 200062, China
NACHI KOREA		**http://www.nachi-korea.co.kr/**	
Seoul office	Phone: +82-(0)2-469-2254	Fax: +82-(0)2-469-2264	2F Dongsan Bldg. 276-4, Sungsu 2GA-3DONG, Sungdong-ku, Seoul 133-123, KOREA

Robot Division

1-1-1, FUJIKOSHIHONMACHI, TOYAMA CITY, JAPAN 930-8511
Phone +81-76-423-5137
Fax +81-76-493-5252

In case that an end user uses this product for military purpose or production of weapon, this product may be liable for the subject of export restriction stipulated in the Foreign Exchange and Foreign Trade Control Law. Please go through careful investigation and necessary formalities for export.

Original manual is written in Japanese.