

# PREFACE

Thank you for purchasing Pro-face's ladder logic programming software, Pro-Control Editor Ver. 5.1.

To ensure the safe and correct use of this product, be sure to read all related materials carefully and keep them nearby so that you can refer to them whenever required.

## NOTE

1. The copyrights to all programs and manuals included in Pro-Control Editor Ver. 5.1 (hereinafter referred to as "this product") are reserved by Digital Electronics Corporation. Digital Electronics Corporation grants the use of this product to its users as described in the "Software Licence Agreement" (included with the CD-ROM). Any violation of the abovementioned conditions is prohibited by both Japanese and foreign regulations.
2. The contents of this manual have been thoroughly inspected. However, if you should find any errors or omissions in this manual, please contact your local sales representative.
3. Regardless of the above clause, Digital Electronics Corporation shall not be held responsible for any damages, losses or third-party claims resulting from the use of this product.
4. Differences may exist between the descriptions found in this manual and the actual functioning of this software. Therefore, the latest information on this software is provided in the form of data files (Readme.txt files, etc.) and/or separate documents. Refer to these sources as well as this manual prior to use.
5. Even though the information contained in and displayed by this product may be related to intangible or intellectual properties of Digital Electronics Corporation or third parties, Digital Electronics Corporation shall not warrant or grant the use of said properties to any users or other third parties. Also, Digital Electronics Corporation shall not be liable for problems related to intellectual properties of the third party caused by using the information contained in and displayed by this product.

© 2004 Digital Electronics Corporation. All rights reserved.

Digital Electronics Corporation, August 2004.

For information about the rights to trademarks and trade names, see "TRADE-MARK RIGHTS."

# TABLE OF CONTENTS

PREFACE .....	1
APPLICABLE PRODUCTS .....	8
HOW TO USE THIS MANUAL .....	9
PRODUCT USAGE PRECAUTIONS .....	10
FOR GLC2400/GLC2600 USERS .....	11
DOCUMENTATION CONVENTIONS .....	12

## CHAPTER1      CONTROLLER FEATURES

<b>1.1    Operating the GLC .....</b>	<b>1-1</b>
1.1.1    Controller Feature Overview .....	1-3
1.1.2    RUN Mode .....	1-5
1.1.3    GLC Scan Overview .....	1-5

## CHAPTER2      VARIABLES

<b>2.1    Variable Names .....</b>	<b>2-1</b>
<b>2.2    Variable Types .....</b>	<b>2-3</b>
<b>2.3    Accessing Variables .....</b>	<b>2-6</b>

## CHAPTER3      SYSTEM VARIABLES

<b>3.1    System Variable List .....</b>	<b>3-1</b>
<b>3.2    System Variable Details .....</b>	<b>3-3</b>
3.2.1    #AvgLogicTime .....	3-3
3.2.2    #AvgScanTime .....	3-3
3.2.3    #LogicTime .....	3-4
3.2.4    #ScanCount .....	3-4
3.2.5    #ScanTime .....	3-5
3.2.6    #WatchdogTime .....	3-5
3.2.7    #PercentAlloc .....	3-5
3.2.8    #TargetScan .....	3-6
3.2.9    #ForceCount .....	3-6
3.2.10    #IOStatus .....	3-7
3.2.11    #Platform .....	3-7
3.2.12    #Status .....	3-8

3.2.13	#Version .....	3-9
3.2.14	#FaultCode .....	3-10
3.2.15	#FaultRung .....	3-11
3.2.16	#IOFault .....	3-12
3.2.17	#Overflow .....	3-12
3.2.18	#DisableAutoStart .....	3-13
3.2.19	#Fault .....	3-13
3.2.20	#FaultOnMinor .....	3-14
3.2.21	#Command .....	3-14
3.2.22	#Screen .....	3-15
3.2.23	#Clock100ms .....	3-16
3.2.24	#Year .....	3-17
3.2.25	#Month .....	3-17
3.2.26	#Day .....	3-18
3.2.27	#Time .....	3-18
3.2.28	#Weekday .....	3-19
3.2.29	#LadderMonitor .....	3-20
3.2.30	#RungNo .....	3-21

<b>CHAPTER4</b>	<b>INSTRUCTIONS</b>
-----------------	---------------------

<b>4.1</b>	<b>Instruction List .....</b>	<b>4-1</b>
<b>4.2</b>	<b>Instruction Details .....</b>	<b>4-7</b>
4.2.1	NO (Normally Open) .....	4-7
4.2.2	NC (Normally Closed) .....	4-9
4.2.3	OUT/M (Output Coil) .....	4-9
4.2.4	NEG (Negated Coil) .....	4-11
4.2.5	SET (Set Coil) .....	4-11
4.2.6	RST (Reset Coil) .....	4-13
4.2.7	PT (Positive Transition Contact) .....	4-13
4.2.8	NT (Negative Transition Contact) .....	4-14
4.2.9	AND (And) .....	4-15
4.2.10	OR (Or) .....	4-17
4.2.11	XOR (Exclusive OR) .....	4-17
4.2.12	NOT (Bit Invert) .....	4-19
4.2.13	MOV (Transfer) .....	4-19
4.2.14	BMOV (Block Transfer) .....	4-20
4.2.15	FMOV (Fill Transfer) .....	4-22

4.2.16	SUM (Sum Total)	4-24
4.2.17	AVE (Average)	4-26
4.2.18	BCNT (Bit Count)	4-28
4.2.19	ROL (Rotate Left)	4-29
4.2.20	ROR (Rotate Right)	4-30
4.2.21	SHL (Shift Left)	4-31
4.2.22	SHR (Shift Right)	4-33
4.2.23	RCL (Left Rotation with Carry)	4-35
4.2.24	RCR (Right Rotation with Carry)	4-36
4.2.25	SAL (Arithmetic Shift Left)	4-37
4.2.26	SAR (Arithmetic Shift Right)	4-38
4.2.27	ADD (Add)	4-39
4.2.28	SUB (Subtract)	4-40
4.2.29	MUL (Multiply)	4-41
4.2.30	DIV (Divide)	4-42
4.2.31	MOD (Modulus)	4-43
4.2.32	INC (Increment)	4-44
4.2.33	DEC (Decrement)	4-44
4.2.34	SQRT (Square Root)	4-45
4.2.35	EQ (Compare: = )	4-46
4.2.36	GT (Compare: > )	4-47
4.2.37	LT (Compare: < )	4-47
4.2.38	GE (Compare: >= )	4-49
4.2.39	LE (Compare: <= )	4-49
4.2.40	NE (Compare: <> )	4-49
4.2.41	TON (Timer ON Delay)	4-50
4.2.42	TOF (Timer OFF Delay)	4-52
4.2.43	TP (Timer Pulse)	4-55
4.2.44	CTU (UP Counter)	4-57
4.2.45	CTD (DOWN Counter)	4-57
4.2.46	CTUD (UP/DOWN Counter)	4-59
4.2.47	BCD (BCD Conversion)	4-59
4.2.48	BIN (Binary Conversion)	4-60
4.2.49	ENCO (Encode)	4-61
4.2.50	DECO (Decode)	4-62
4.2.51	RAD (Radian conversion)	4-62

4.2.52	DEG (Degree Conversion) .....	4-63
4.2.53	SCL (Scale conversion) .....	4-63
4.2.54	JMP (Jump) .....	4-65
4.2.55	JSR (Jump Subroutine) .....	4-66
4.2.56	RET (Return Subroutine) .....	4-67
4.2.57	FOR/NEXT (Repeat) .....	4-67
4.2.58	PID (PID Calculation).....	4-68
4.2.59	SIN (sine function).....	4-81
4.2.60	COS (cosine function) .....	4-81
4.2.61	TAN (tangent function).....	4-82
4.2.62	ASIN (Arc Sine).....	4-82
4.2.63	ACOS (Arc Cosine) .....	4-83
4.2.64	ATAN (Arc Tangent).....	4-83
4.2.65	COT (Cotangent).....	4-84
4.2.66	EXP (Exponent) .....	4-84
4.2.67	LN (Natural Logarithm).....	4-85

**CHAPTER5 LS AREA REFRESH**

<b>5.1</b>	<b>LS Area Refresh Overview .....</b>	<b>5-1</b>
<b>5.2</b>	<b>LS Area Refresh Settings.....</b>	<b>5-2</b>
5.2.1	LS Area - When not using a Device/PLC .....	5-4
<b>5.3</b>	<b>GLC and External Device Data Sharing .....</b>	<b>5-7</b>
5.3.1	LS Area Refresh Cautions.....	5-9

**CHAPTER6 GLC LADDER MONITOR FEATURE**

<b>6.1</b>	<b>Overview of the GLC Ladder Monitor Feature.....</b>	<b>6-1</b>
<b>6.2</b>	<b>Starting/Exiting the GLC Ladder Monitor .....</b>	<b>6-3</b>
6.2.1	Preparing to operate the GLC Ladder Monitor.....	6-3
6.2.2	Starting the GLC Ladder Monitor .....	6-4
6.2.3	Exiting the GLC Ladder Monitor .....	6-4
<b>6.3</b>	<b>Various GLC Ladder Monitor Features .....</b>	<b>6-5</b>
6.3.1	Online Monitor Feature (Normal Display) .....	6-5
6.3.2	Rung Jump/Scroll Features .....	6-7
6.3.3	Instruction Enlarge Feature (Zoom Display) .....	6-7
6.3.4	GLC Variable Monitor Feature .....	6-8
6.3.5	Setup Value Edit Feature .....	6-9
6.3.6	Variable/Instruction Search Feature .....	6-11

**CHAPTER7      BACKUP**

**7.1    Overview of the Backup Feature ..... 7-1**  
**7.2    Backup Operation Procedure ..... 7-2**  
    7.2.1    Backup ..... 7-2  
    7.2.2    Recovery ..... 7-3

**CHAPTER8      I/O DRIVERS**

**8.1    I/O Drivers Overview ..... 8-1**  
**8.2    Flex Network Interface Driver ..... 8-2**  
    8.2.1    Flex Network Interface Unit Self-Diagnosis ..... 8-2  
    8.2.2    Communication Check ..... 8-3  
    8.2.3    Error S-No. Display ..... 8-4  
    8.2.4    I/O Monitor (I/O Connection Check) ..... 8-5  
    8.2.5    Flex Network Troubleshooting ..... 8-11  
**8.3    DIO Driver ..... 8-13**  
    8.3.1    DIO Unit Self-Diagnosis ..... 8-13  
    8.3.2    I/O Monitor (I/O Connection Check) ..... 8-15  
    8.3.3    DIO Troubleshooting ..... 8-17

**CHAPTER9      ERROR MESSAGES**

**9.1    Error Message List ..... 9-1**  
**9.2    Error Codes ..... 9-3**  
**9.3    Program Errors ..... 9-4**

**APPENDICES**

**Appendix 1 Instruction List ..... A-1**  
**Appendix 2 System Variable List ..... A-3**

**INDEX**

# TRADEMARK RIGHTS

The company names and product names used in this manual are the trade names, trademarks (including registered trademarks), and service marks of their respective companies. This product does not include individual descriptions pertaining to the rights held by each company.

<b>Trademark / Tradename</b>	<b>Rights Holder</b>
Microsoft, MS, MS-DOS, Windows, Windows 95, Windows 98, Windows Me, Windows NT, Windows 2000, Windows XP, Windows Explorer, Microsoft Excel	Microsoft Corporation, USA
Intel, Pentium	Intel Corporation, U.S.A.
Pro-face, Flex Network	Digital Electronics Corporation (worldwide)
Ethernet	Western Digital Electric Corporation, USA
Adobe, Acrobat	Adobe Systems Corporation

The following terms used in this manual differ from the official trade names and trademarks (listed above).

<b>Term used in this manual</b>	<b>Formal Tradename or Trademark</b>
Windows 95	Microsoft® Windows® 95 Operating System
Windows 98	Microsoft® Windows® 98 Operating System
MS-DOS	Microsoft® MS-DOS® Operating System
Windows Me	Microsoft® Windows® Me Operating System
Windows NT	Microsoft® Windows NT® Operating System
Windows 2000	Microsoft® Windows® 2000 Operating System
Windows XP	Microsoft® Windows XP® Operating System

# APPLICABLE PRODUCTS

The following is a list of products used with Pro-Control Editor Ver. 5.1 software. In this manual, the following names are used to describe series units and products. "GP Type" refers to GP-PRO/PB III for Windows Ver. 7.1.

Series		Product Name	Model	GP Type			
GLC100 Series		GLC100L	GLC100-LG41-24V	GLC100L			
		GLC100S	GLC100-SC41-24V	GLC100S			
GLC300 Series		GLC300T	GLC300-TC41-24V	GLC300T			
GLC2000 Series	GLC2300 Series	GLC2300L	GLC2300-LG41-24V	GLC2300L			
		GLC2300T	GLC2300-TC41-24V	GLC2300			
	GLC2400 Series	GLC2400T	GLC2400-TC41-24V	GLC2400 <sup>*1</sup>			
				"Rev.* - None, 1"			
	GLC2500 Series	GLC2500T	GLC2500-TC41-24V GLC2500-TC41-200V	GLC2400 <sup>*1</sup>			
				"Rev.* - Above2"			
	GLC2600 Series	GLC2600T	GLC2600-TC41-24V GLC2600-TC41-200V	GLC2500			
				GLC2600 <sup>*1</sup>			
				"Rev.* - Above2"			
	LT Series	LT Type A Series	LT Type A1	GLC150-BG41-XY32SK-24V	LT TypeA		
LTC Type A1			GLC150-SC41-XY32SK-24V	LTC TypeA			
LT Type A2			GLC150-BG41-XY32SC-24V	LT TypeA			
LT Type B/B+ Series		LT Type B	GLC150-BG41-FLEX-24V	LT TypeB/B+			
		LT Type B+	GLC150-BG41-XY32KF-24V				
		LTC Type B+	GLC150-SC41-XY32KF-24V		LTC TypeB+		
LT Type C Series		LT Type C	GLC150-BG41-RSFL-24V	LT TypeC			
LT Type H Series		LT Type H1	GLC150-BG41-ADK-24V GLC150-BG41-ADPK-24V GLC150-BG41-ADTK-24V	LT TypeH			
					LTC Type H1	GLC150-SC41-ADK-24V GLC150-SC41-ADPK-24V GLC150-SC41-ADTK-24V	LTC TypeH

\*1 For how to distinguish "Revisions", refer to "For GLC2400/GLC2600 Users".



# HOW TO USE THIS MANUAL

The GP-PRO/PB III C-Package03 manuals consist of seven volumes. A description of each is found in the table below. These PDF manuals are located in disk 1 of your C-Package03 CD set. (The “Setup Guide” is not included as a PDF file.) Supplemental explanations and additional or revised information about functions may be provided as data files. To read the data files, click the [Start] button, point to [Programs], [Pro-face], and [ProPB3 C-Package], then click [ReadMe] to view this information.

For detailed information on Pro-face products, please refer to that product’s user manual (sold separately).

<b>GP-PRO/PB III C-Package03</b>	
Setup Guide	Describes software installation and basic application development procedures.
<b>Pro-Control Editor Ver. 5.1</b>	
User Manual (this manual)	Describes the software settings for combining with the GLC, variables, and instructions.
Operation Manual	Provides exercises for learning the basic functions, from installation to operation, and a list of error messages. Describes procedures using the variables registered by the Pro-Control Editor for use by the GP-PRO/PB III.
<b>GP-PRO/PB III for Windows Ver. 7.1</b>	
Operation Manual (PDF)	Describes the installation, operating procedures, and software functions of the GP screen creation software.
Tag Reference Manual	Explains "tags" for specifying on-screen functions of the GP.
Parts List	Describes the parts and symbols provided in the software for creating GP screens.
Device/PLC Connection Manual	Describes procedures for connecting the GP to PLCs, temperature controllers, and inverters of other manufacturers.

For your convenience, after you install the screen editor software, screen layout sheets can be found in the Pro-face folder described below. You can use these layout sheets for specifying the PLC registers when setting the tag addresses. The layout sheets consist of two files: List of Device Assignments and Tag Layout Sheet. The location and name of each file is shown in the following table.

For directions on using Microsoft® Excel, refer to the manuals supplied with the software.

- 
- \* *The abovementioned GP-PRO/PB III manuals describe the procedures for developing GP screens. The steps for developing GLC/LT screens are identical; simply substitute “GLC/LT” for “GP.”*
  - \* *As a supplement to the manuals listed above, detailed explanations are available in the GP-PRO/PB III online help.*

## Preface

Folder Name	File Name	Contents
Pro-face\propbwin\sheet	Device1E.xls	List of device assignments
	TAG1E.xls	Tag layout sheet
	TAG2E.xls	
	TAG3E.xls	
	TAG4E.xls	

Adobe® Acrobat® Reader is required to view the CD-ROM's PDF manuals.

# PRODUCT USAGE PRECAUTIONS



## WARNING

**Do NOT use the GLC unit for control in situations where a life-threatening accident or major machine damage could occur.**

### Disk Media Usage Precautions

To prevent CD-ROM or floppy disk damage or data loss, be sure to observe the following instructions:



- Be sure to remove the disk media from its disk drive prior to turning the PC ON or OFF.



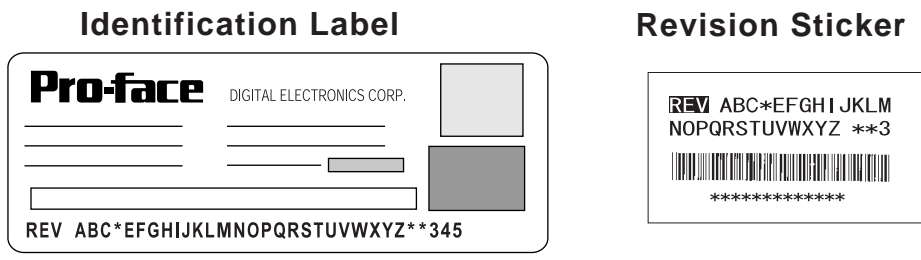
- Do NOT remove the disk media from its drive while the drive operation lamp is lit.
- Do NOT touch the disk media's (CD-ROM or floppy disk) recording surface.
- Do NOT place the disk(s) where they may be exposed to extreme temperatures, high humidity, or dust.

# FOR GLC2400/GLC2600 USERS

The revision code can be easily found using the GLCunit’s rear face identification label or revision sticker. In the area titled “REV”, the code is indicated by asterisks (\*) or marked with a marker pen.

■ **How to Read the Code**

In the example below, asterisks (\*) are placed at positions “D”, “1”, and “2”, which indicates the revision version as “D-2”.



■ **Revision Categories**

Revision Types	Meaning
"Rev.* - None, 1"	The revision code is not used, or is "1".
"Rev.* - Above2"	The revision code is "2" or higher.






# DOCUMENTATION CONVENTIONS

This manual uses the following symbols and terminology.

If you have any questions about the contents of this manual, please contact your local Pro-face sales distributor. If you have any question about your personal computer or the Windows® software, please contact your local distributor or manufacturer.



## ■ Safety Symbols and Terms

This manual uses the following symbols and terms for important information related to the correct and safe operation of this product.

Symbol	Description
 <b>Warning</b>	Incorrect operation resulting from negligence of this instruction may cause death or serious injury.
 <b>Caution</b>	Incorrect operation resulting from negligence of this instruction may cause personal injury or damage to equipment.
 <b>Important</b>	Failure to observe this instruction may cause abnormal operation of equipment or data loss.
 <b>Careful!</b>	Instructions / procedures that must be performed to ensure correct product use.
 <b>STOP</b>	Actions / procedures that should NOT be performed.

## General Information Symbols and Terms

This manual uses the following symbols and terms for general information.

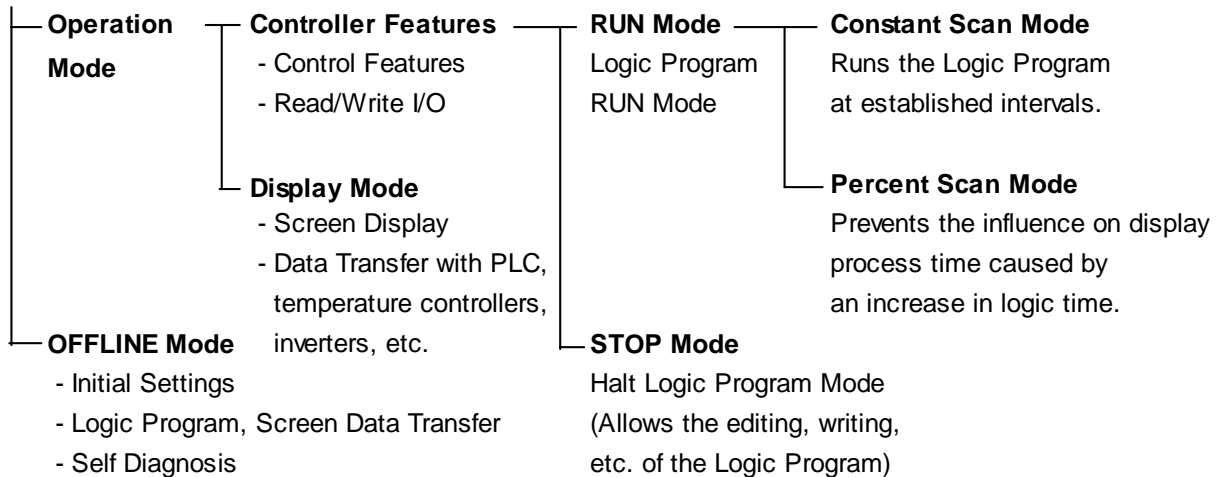
Symbol	Description
 <b>Note:</b>	Provides hints on correct use or supplementary information.
 <b>Reference</b>	Indicates related information (manual name, page number).
*1, *2, (etc.)	Indicates related supplemental information.
Pro-Control Editor	Referred to in this manual as the "Editor." Software for editing, transferring, and monitoring a GLC/LT unit's ladder logic program.
Controller	The control function of a GLC/LT unit.
GP-PRO/PB III	Screen creation software GP-PRO/PBIII for Windows Ver. 7.1.
GLC	Indicates the "GLC/LT series" of graphic logic controllers manufactured by the Digital Electronics Corporation.
External Communication Device	Indicates peripheral devices including PLCs (programmable logic controller), temperature controllers, and inverters. Note that devices connected through Flex Network and DIO are not included.

# 1 Controller Features

## 1.1 Operating the GLC

The GLC contains both screen display and I/O control features. These features and their respective modes are described below.

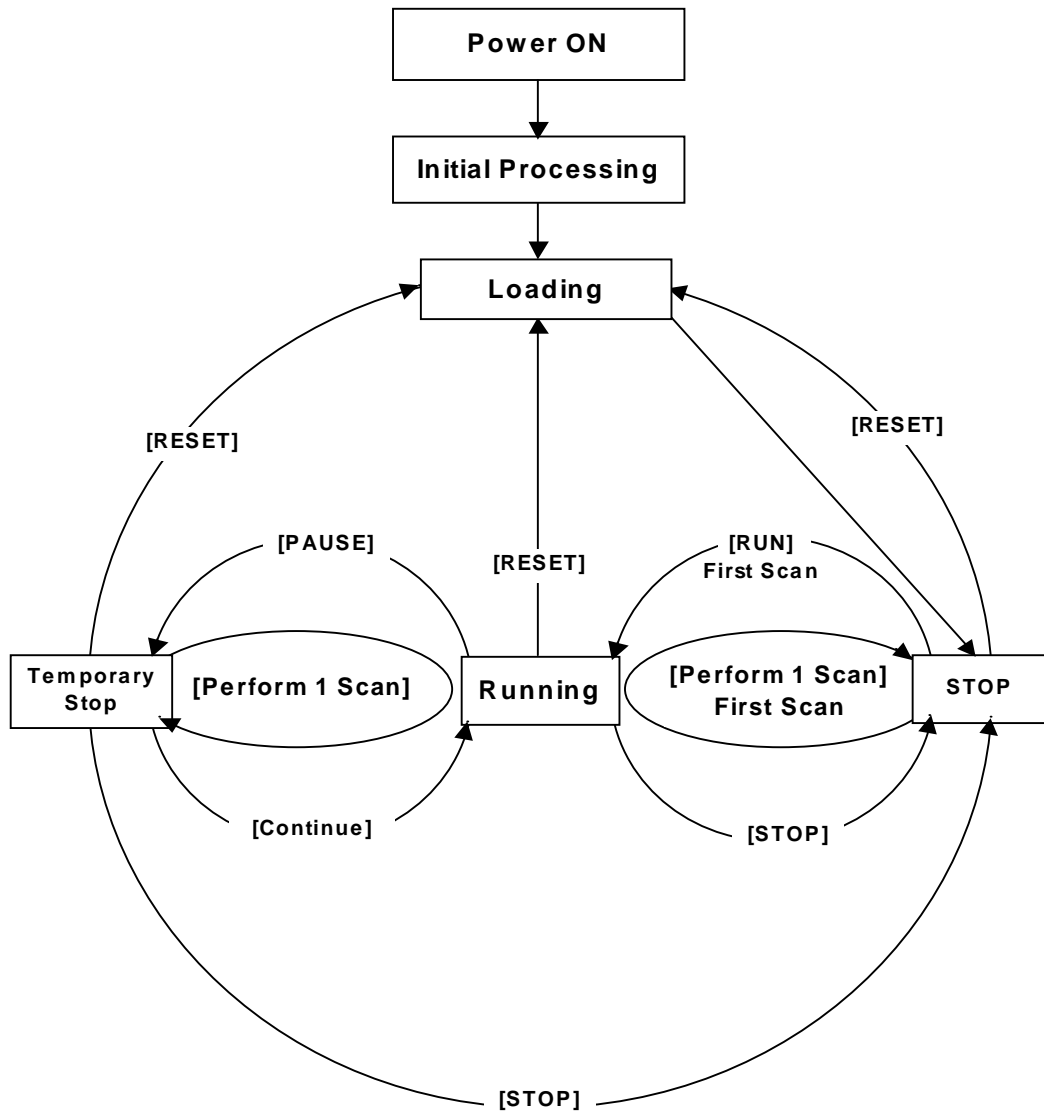
### GLC Features



- **Understanding the GLC unit's operation modes is critical for designing a system. Please read this chapter thoroughly to understand the operation and to design the system in consideration of safety issues.**
- **When OFFLINE mode is entered, the controller will stop. Re-entering RUN mode will reset the GLC.**

**1.1.1 Controller Feature Overview**

The Controller feature functions as follows. The following pages provide detailed descriptions of each step.



■ **Initial Processing**

This is the initial state that the engine uses to perform the Logic Program. Once initialization is finished, the Controller enters the “Loading” state.

■ **Loading**

Here, the actual reading-in of the Logic Program from memory is performed. After a check determines whether the Logic Program is successfully loaded or not. If an error has occurred, error processing is performed. If Loading is successful, the program enters the [STOP] state. If the [Power ON] Operation Mode has been set to [START], the [RUN] instruction is automatically performed.

### ■ STOP

In this state, the Controller is waiting to receive another instruction. Once the [RESET], [Perform 1 Scan], [Continue], or [PAUSE] instruction is received, the Controller will change to that state.

- The [RESET] instruction will change the program to the [Loading] state.

At this time, variables are initialized. Retentive variables maintain data before the power shuts down or the GLC resets. However, when triggering Controller reset in Monitoring mode\*<sup>1</sup>, or when using #Command, the value set in Programming Mode\*<sup>2</sup> is used as the initial value. The [RUN] and [Perform 1 Scan] instructions clear non-retentive variables to zero (0).

- The [RUN] instruction will change the program state to [Running].
- The [Perform 1 Scan] instruction will perform the program once.

### ■ First Scan

Executes the I/O Read, performs any Logic Program that is higher than the START level, and executes the I/O write.

### ■ Running

This is the logic program execution engine's continuous performance mode. In this mode, it executes I/O Reads, performs Logic Programs, executes I/O writes, and updates System Variables (such as #AvgLogicTime, #AvgScanTime).

- The [RESET] instruction will change the program to the [Loading] state.
- The [STOP] instruction will change the program to the [STOP] state.
- The [PAUSE] instruction will change the program to the [Temporary Stop] state.

### ■ Temporary Stop

The logic program execution engine is temporarily stopped in this state. To avoid an I/O watchdog timeout, the system executes an I/O read and I/O write. However, the logic program is not executed, so the output state does not change. When a command is received, the system switches to the appropriate state.

- The [RESET] instruction will change the program to the [Loading] state.
- The [Perform 1 Scan] instruction will perform the program once.
- The [STOP] instruction will change the program to the [STOP] state.
- The [Continue] instruction will change the program to the [Running] state.

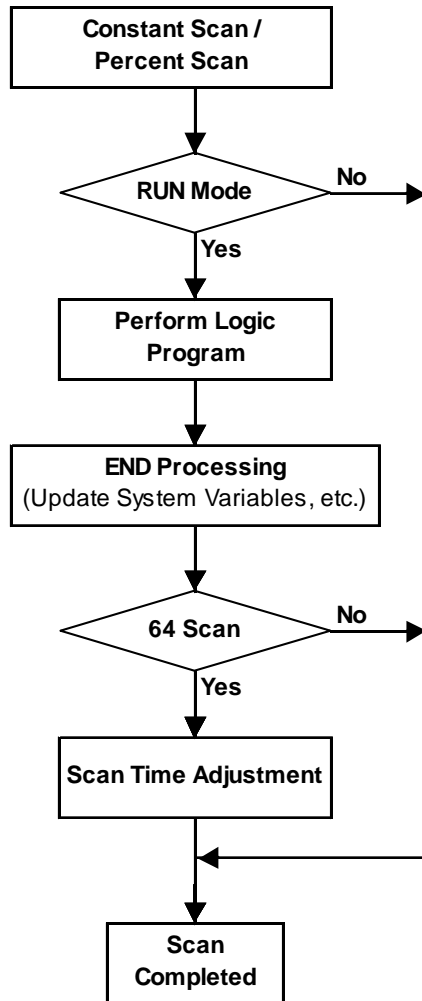
---

1. This mode is used to edit the program currently being executed by the controller.

2. This mode is used to create a program.

**1.1.2 RUN Mode**

RUN Mode uses the following steps.



■ **Scan Time Adjustment**

This adjustment is performed every 64 scans. The various types of adjustments are described below for Constant Scan Time and Percent Scan Time.

◆ **Constant Scan Time Mode**

$$\text{GLC scan time} = (\#AvgLogicTime \times 100) / 50$$

◆ **Percent Scan Time Mode**

$$\text{GLC scan time} = (\#AvgLogicTime \times 100) / \#PercentAlloc$$

**Reference** For information about #AvgLogicTime or #PercentAlloc, see Chapter 3 – “System Variables.”



**The GLC or LT unit’s Scan Time includes an error.**

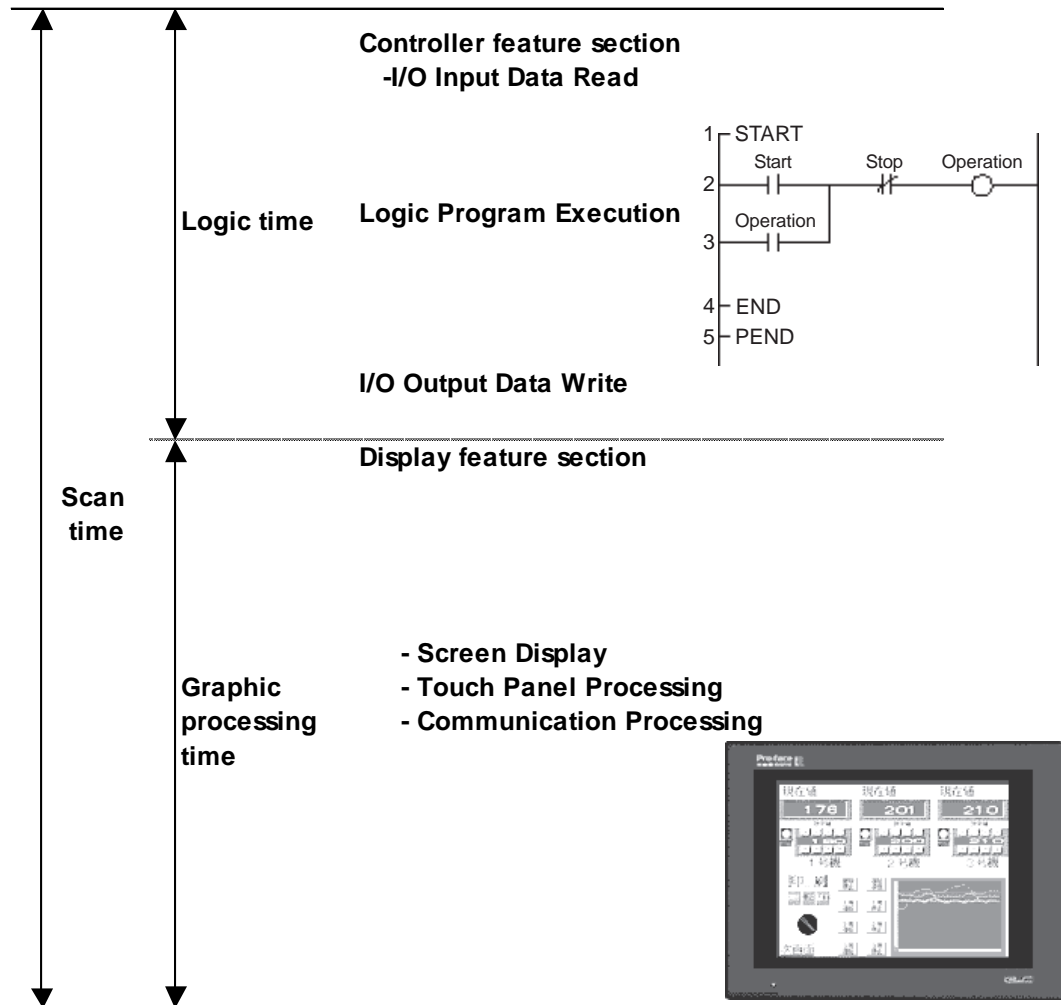
Model	Difference
GLC100 Series	approx. - 0.2%
GLC300 Series GLC2000 Series LT Series	approx. + 0.3%



**1.1.3 GLC Scan Overview**

GLC Scan time includes the Controller feature (logic program execution time) and the Display feature (screen display/touch panel processing/external device communication processing time), as follows.

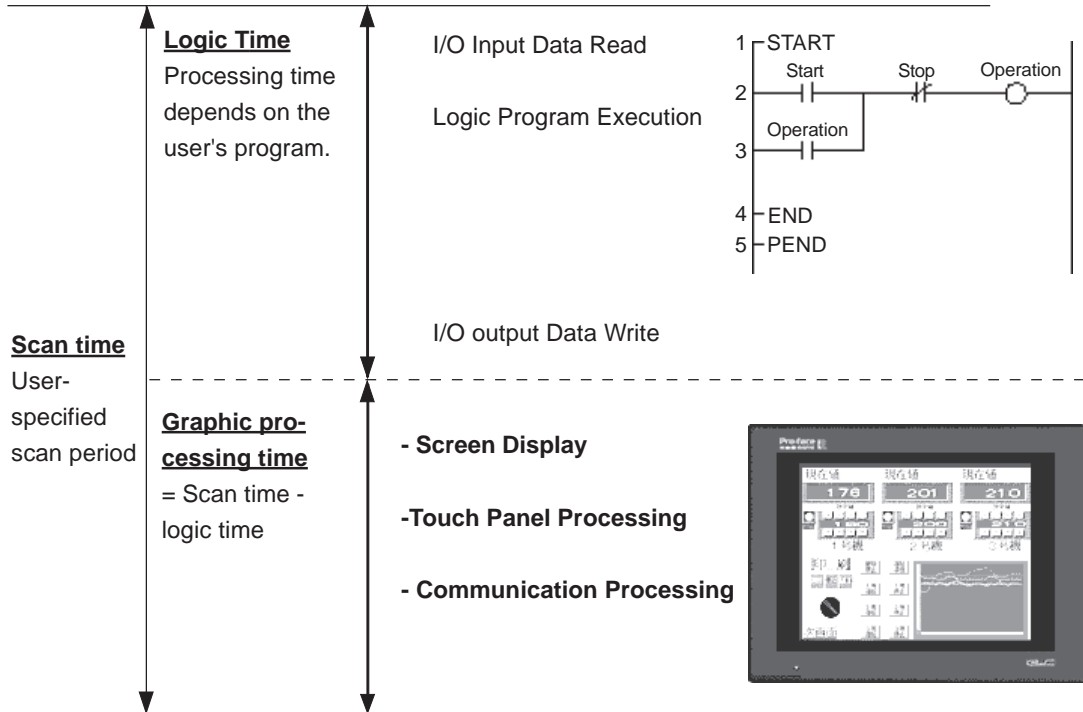
It has two modes: Constant Scan Time mode and Percent Scan Time mode.



**Reference** See 1.1.2 – “RUN Mode.”

## ■ Constant Scan Mode

When this setting is used, the logic program is processed periodically at specified intervals. This mode is suitable for a system that places a higher priority on the control (logic program), in which the screen is mainly used for monitoring (data display) and used for control less.



Graphic processing time = Setting time for constant scan time mode (ms) – logic time (variable)

E.g.: If constant scan time is set to 50ms and logic executing time is 20ms

$$\begin{aligned} \text{Graphic processing time} &= 50\text{ms} - 20\text{ms} \\ &= 30\text{ms} \end{aligned}$$

The longer the logic time, the shorter the graphic processing time will become.

Though GLC display response will be slower, the logic program will execute continuously.



- **Be sure to enter the setting value for the scan time in 10ms increments.**
- **When the logic time exceeds 50 % of the designated setting value for constant scan mode, the scan time is automatically adjusted so that it is twice as long as the logic time.**

**E.g.: When the logic time is 30 ms and the constant scan mode is 50 ms, the scan time is 60 ms.**



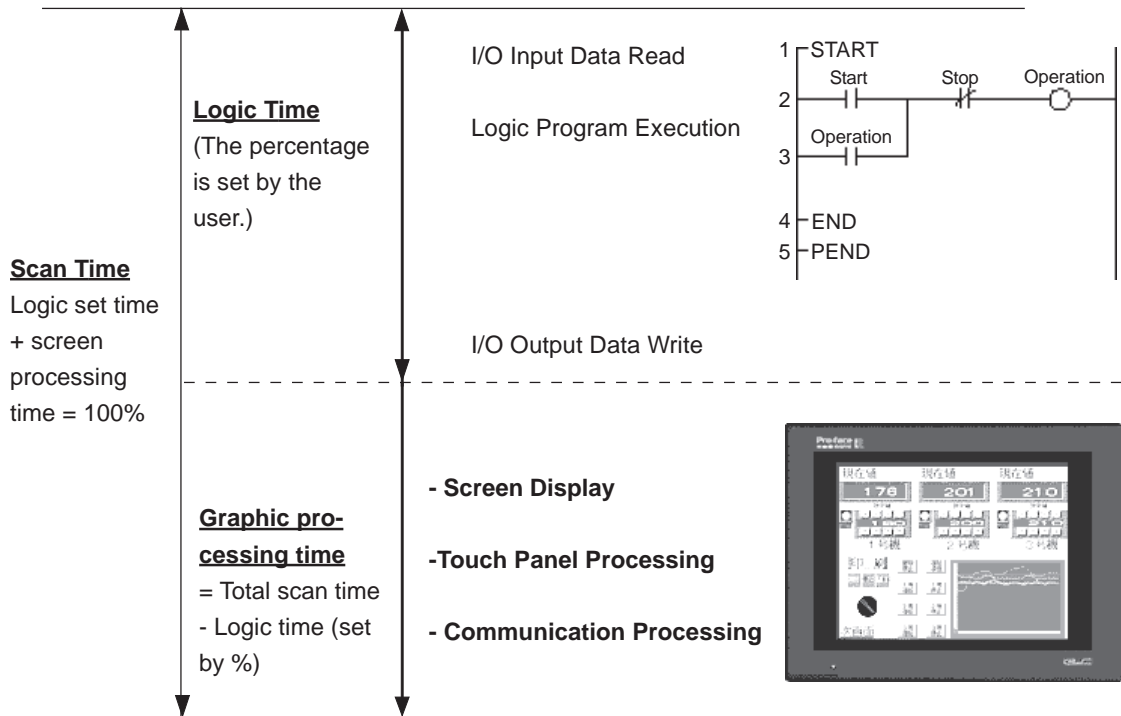
- When determining the value for the setting time, use the #AvgScanTime value obtained from a test run of the GLC.

**Reference** See 3.2.2 – “#AvgScanTime.”

■ **Percent Scan Mode**

This mode is used to vary the scan time by setting the percentage (%) of the logic time in the total scan time.

Since this mode can prevent the graphic processing time from being compressed due to an increase of the logic time, it is well suited to a system where priority is given to screen operation speed and screen switching speed.



$$\text{Scan time} = \text{Logic time} / \text{Percent scan set time (\%)}$$

E.g.: If percent scan time is set to 40% and logic execution time is 20ms

$$\begin{aligned} \text{Scan time} &= (20 \div 40) \times 100 \\ &= 50\text{ms} \end{aligned}$$

$$\begin{aligned} \text{Graphic processing time} &= 50\text{ms} - 20\text{ms} \\ &= 30\text{ms} \end{aligned}$$

When logic time increases, display processing time also increases, resulting in longer scan times.

The longer the logic time, the longer the time allocated to display processing. Therefore, the display is updated more quickly, but the logic program processing cycle slows.



- ***Set the percent scan value so that the scan time is set in 10ms increments.***
- ***There is no change in the processing time for one instruction in a logic program.***
- ***The percent scan setting value cannot be set to more than 50%.***
- ***When the percent scan setting is set to 50%, the display and logic program are processed at the same time. The display process will not be given priority.***

## 2 Variables

This chapter explains the different types of variables used by Pro-Control Editor. Using hardware-independent variables enhances the reusability of your programs.

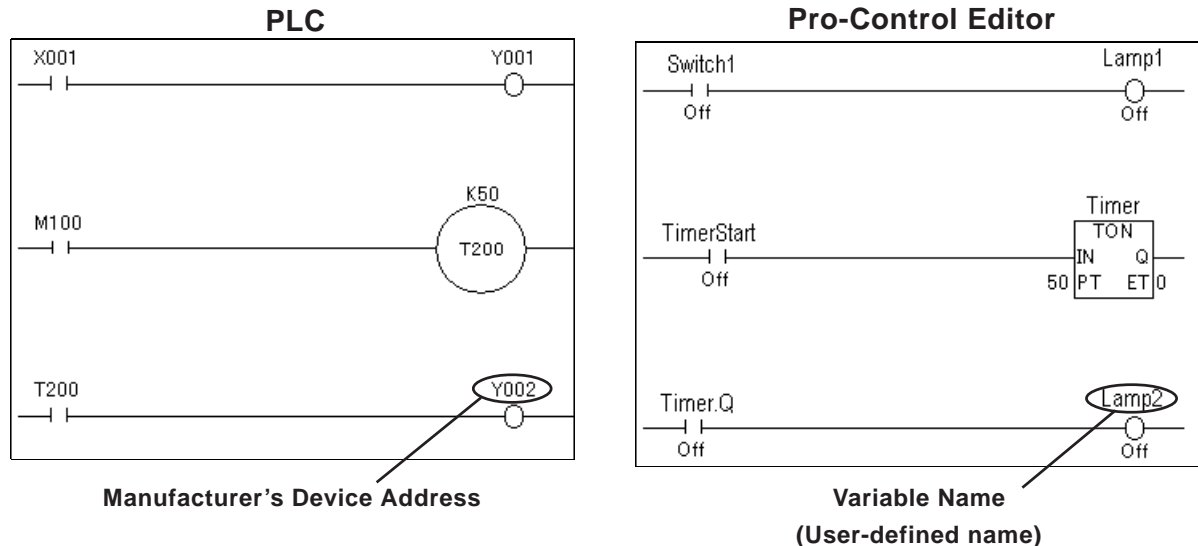
### 2.1 Variable Names

Pro-Control Editor uses variables to store I/O and counter data. Variables are user-designated and are used as-is in the logic program.

In a conventional PLC, the area used to store data is called a device address. These addresses are given specific names by each PLC manufacturer.

PLC Manufacturer (Examples)	External I/O	Internal Relay	Timer	Data Register
Company "M"	X001	M100	T200	D00001
Company "O"	O1	I001	TIM000	DM0000
Digital Electronics Corporation	Switch1	Timerstart	Timer	Operating Time

With Pro-Control Editor, you can assign names to these device addresses and use them as variables in a logic program.



When designating variable names, be aware of the following limitations.

- Maximum Variable Name length is 20 characters (20 bytes).
- No differentiation is made between upper-case and lower-case characters. If duplicates are created, only the first word registered will be enabled (valid).

E.g.: If "TANK" was registered before "tank," "tank" will be invalid, even though entering it will not create an error.

## Chapter 2 – Variables

- Except for the first character, variable names can use numbers.
- Variable names cannot contain spaces.
- The underscore ( \_ ) is the only special character that can be used. However, consecutive underscores ( \_\_ ) cannot be used. (OK: tank\_1; Not OK: tank\_\_1).
- The “#” sign cannot be used, since it is a reserved character, .
- “LS” and “LSS” are reserved variable names for use in the System Data Area, the Read Area, and for Special Relays. Therefore, they cannot be used for user-defined variable names.

**Reference** See Chapter 5 – “LS Area Refresh.”



**Note:** When creating variable names, Pro-face recommends using the underscore character to divide the variables into blocks, or groups. This will make the variable names easier to find in the Pro-Control Editor’s variable list.

E.g.: If you have several conveyor belts in your factory system (Conveyor A, Conveyor B, Conveyor C, etc.), include an identifying character in the motor and sensor variable names:

Conveyor A variables:

A\_Motor

A\_Sensor

You could also name a Discrete (bit) as B, Integer as I, floating point as F:

AB\_MotorStartingSwitch

AI\_MotorRotationNumber

AF\_MotorPowerRatio

Here, the variables used for contacts and coils are distinguished from the variables used for basic mathematical operations.

- You can also use an array to set up variable names for each of your PLC’s devices.

Example

PLC Device	Pro-Control Editor	
	Array Variable	Variable Type
External Input	X[100]	Discrete
External Output	Y[100]	Discrete
Internal Relay	M[100]	Discrete
Data Register	D[100]	Integer

**Reference** For information about Variable Settings, refer to the *Pro-Control Editor Operation Manual, 2.4 – “Creating Variables.”*

For information about reserved System Variables, see Chapter 3 – “System Variables” Variable names can be designated by the user.

## 2.2 Variable Types

The Pro-Control Editor uses three types of variables: Discrete (bit), Integer, and Real. Within these types, Timers and Counters are also used. Arrays can be defined and used with each type of variable.

**Reference** For details on defining arrays, see 2.3 – “Accessing Variables.”

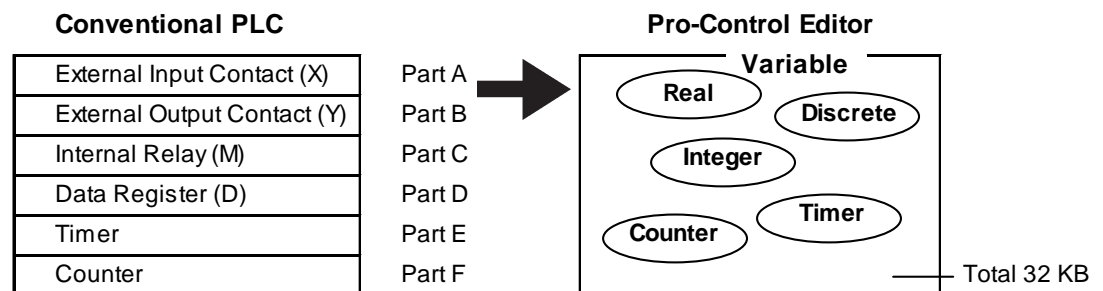
The maximum size of an array (the number of elements it contains) is 65535. However, the actual number of elements that can be used by any application is limited by the size of the GLC unit’s variable storage area. However, the GLC unit’s variable storage area varies depending on the model as described below. Be sure to design your system so that the number of variables used does not exceed the GLC unit’s memory limit.

Series	Variable Storage Area
GLC100/GLC300/GLC2400/GLC2600 “Rev.* - None, 1” *1	32KB
LT/GLC2300/GLC2500/GLC2400/GLC2600 “Rev.* - Above 2” *1	64KB

Use the following table to find the amount of memory used by each variable.

Variable Type	Memory Used (unit:byte)
Discrete	12
Discrete Array	20 + (No. of elements x 12)
Integer	8
Integer Array	20 + (No. of elements x 8)
Real	16
Real Array	20 + (No. of elements x 16)
Timer	48
Counter	80

In the PLC, the number of variables that can be used by each device is limited. In the GLC, however, variables can be registered, regardless of type, as long as the overall limit of 32 KB is not exceeded.



PLC Device and Pro-Control Variable Comparison

\*1 **Reference** For how to distinguish "Revisions", refer to "For GLC2400/GLC2600 Users".

## Chapter 2 – Variables

### ■ Discrete Variables

These variables are used to define a discrete condition, (ON or OFF), using a single bit and the values “0” or “1.”

### ■ Integer Variables

These variables use 32 bits to define integer values from -2147483648 to 2147483647.

### ■ Real Variables

These variables use 64 bits to define floating decimal point values from +/-2.25e-308 to +/-1.79e+308, and “0”.

### ■ Timer/Counter

The Timer and Counter consist of multiple special-purpose variables.

Each dedicated variable’s type is set up individually.

#### ◆ Timer

The following four dedicated variables are used for Timer instructions.

**Reference** For details, see 4.2 – “*Instruction Details.*”

Special-Purpose Variables	Description* <sup>1</sup>	Variable Type
Variable.PT	Preset Value	Integer
Variable.ET	Current Value	Integer
Variable.Q	Timer Output Bit	Discrete
Variable.TI	Timer Measuring Bit	Discrete

1. Any names can be used for the Special-Purpose Variables.



**Note:**

Even when a timer is designated as non-retentive, the special-purpose variable “Timer.PT” will retain data.

**Reference** For a list of retentive/non-retentive variables, see ■ “*Variable Attributes.*”

#### ◆ Counter

The following seven dedicated variables are used for the Counter instructions.

**Reference** For details, see 4.2 – “*Instructions Details.*”

Special Purpose Variables	Description* <sup>1</sup>	Variable Type
Variable.PV	Preset Value	Integer
Variable.CV	Current Value	Integer
Variable.R	Counter Reset	Discrete
Variable.UP	UP Counter	Discrete
Variable.QU	UP Counter Output	Discrete
Variable.QD	DOWN Counter Output	Discrete
Variable.Q	Counter Output	Discrete

1. Any names can be used for the Special-Purpose Variables.





- Even when a counter is designated as non-retentive, the special-purpose variable “Counter.PV” will retain data.
- A scan update will not be performed for a counter when it is reset. One scan is required for resetting the counter.

**Reference** *For retentive/non-retentive variable details, see ■ “Variable Attributes.”*

## ■ Variable Attributes

Variables have the following attributes, in addition to the variable type.

### ◆ Internal

Used internally by the GLC. It cannot be used for external input/output. Internal variables are equivalent to PLC internal relays (internal registers).

### ◆ Input/Output

External input/output is available. Assign variables to I/O in the [Configure I/O] window. This feature is equivalent to the input/output relays of the PLC.

**Reference** *For I/O configuration details, refer to the Pro-Control Editor Operation Manual, 2.11 – “Assigning I/O.”*

### ◆ Retentive

Retentive-type variables use the GLC unit’s SRAM, which preserves data values in the case of a power failure. The initial values for these variables are set via Programming mode. When the GLC unit is powered down or reset, all current data is retained. However, when the GLC unit’s Controller is reset in Monitoring mode or by using #Command, or when logic programs are downloaded, all data is initialized using Programming mode preset values.

In addition, reading the GLC unit’s .PRW files will save the execution results to the Editor. However, be careful when using retentive-type variables as initial values. If these variables are designed to vary while the logic program is being executed, the predetermined initial values will be lost when the data is loaded into the Editor. Non-retentive variables are either cleared to 0 or set to OFF.



**After GP/GLC unit power is turned OFF and the backup battery runs down, data stored in SRAM will be lost. When this happens, all SRAM data is re-initialized to the value(s) set in Programming mode.**

### ◆ Global

These variables can be designated as either global or non-global. Specify "global" for variables that are used to display Drawing Board Parts. Global variables are automatically registered as GLC symbols in the Symbol Editor when you save the ladder logic program. These variables can also be shared with the Drawing Board’s display feature. Global/non-global settings of multiple variables can be performed simultaneously by selecting the desired variables from the Variable List. Up to 2048 global variables can be set.

**Reference** *Refer to the Pro-Control Editor Operation Manual, 4.6 – “Changing Variable Attributes.”*



Preprogrammed system variables are set to “global” in the GLC unit’s initial settings.

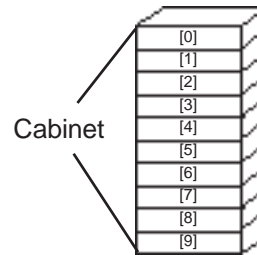
## 2.3 Accessing Variables

This section explains how to access variable array elements, bits, bytes and words with Pro-Control Editor.

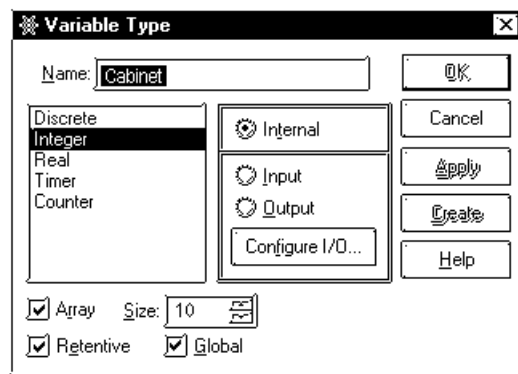
An array is a method of declaring and handling multiple elements with a single variable name. Variables of the same type can be registered as one group using an array.

One analogy is the drawers of a cabinet.

The array variable Cabinet[10] has 10 drawers, numbered from [0] to [9]. These drawers are called Cabinet[0], Cabinet[1], . . . Cabinet[9]. Each drawer corresponds to an individual data register in the PLC.



When using 10 locations of Cabinet memory, first declare the variable name of Cabinet and the array size (number of elements) of 10. The variable type settings are listed as follows:



### ■ Accessing a Discrete Array

To access the elements of a Discrete array, a modifier [n] must be attached to each element. To access the modifier, it is assigned an element number, but the first element number in an array must be “0.”

E.g.: The Discrete array “MotorSetting” is a Discrete array of 10 elements. The seventh element controls the output coil Fan. When the seventh element is turned ON, the output coil turns ON. To access the seventh element of MotorSetting, enter MotorSetting[6].



■ **Accessing an Integer/Integer Array**

Integers and Integer Arrays can be accessed via array elements, bits, bytes, and words.

To access an array’s element, add [n] to the end of the variable name. To access using bits, bytes, and words, the following suffixes are used. The modifier [m] is used to denote the position of the element in the array being accessed.

Access Item/Unit	Suffix
Bit	.X[m]
Byte	.B[m]
Word	.W[m]

◆ **To Access an Element with the Integer Array**

An Integer Array can be used for numerical calculation, tracking of repetitive information, and data logging.

E.g.: To record the number of sodas sold in one month in the Integer Array **Water\_Sales**, design your array as follows.

The array consists of 31 Integer type elements that correspond to the 31 days in a month.

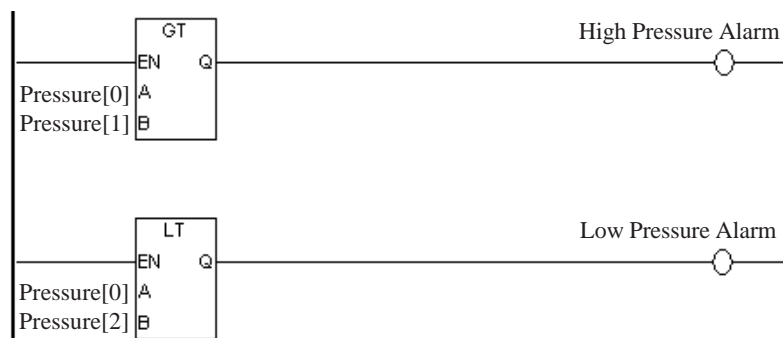
[Day 1]	Water_Sales[0]
[Day 2]	Water_Sales[1]
[Day 3]	Water_Sales[2]
[Day 4]	Water_Sales[3]
	—
	—
	—
[Day 28]	Water_Sales[27]
[Day 29]	Water_Sales[28]
[Day 30]	Water_Sales[29]
[Day 31]	Water_Sales[30]

The following diagram is an example of the Integer Array **Pressure**, using three elements.

- **Pressure[0]** represents the current pressure of the boiler.
- **Pressure[1]** represents the pressure upper limit value.
- **Pressure[2]** represents the pressure lower limit value.

When the pressure is higher or lower than the pressure limit, an alarm turns ON.

Current Pressure	Pressure[0]
Pressure Upper Limit Value	Pressure[1]
Pressure Lower Limit Value	Pressure[2]

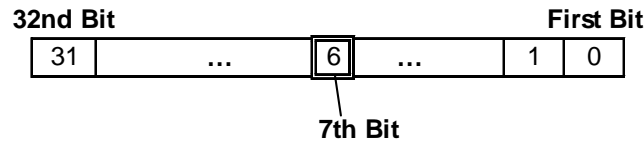


## Chapter 2 – Variables

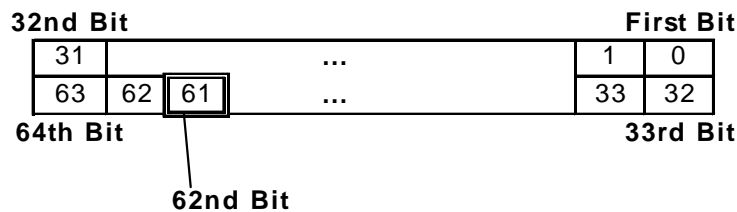
### ◆ To Access an Integer Array using bits

As is the case with the discrete array variables, integer arrays can be accessed via bits, bytes, and words. To access the **m+1st** bit of the **n+1st** element in the **Integer\_Array\_Variable\_Drink Sales**, enter **Drink\_Sales[n].X[m]**.

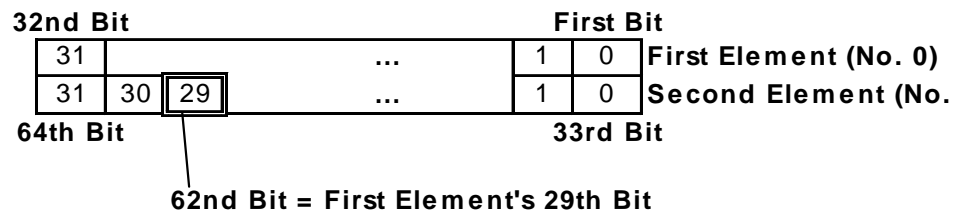
E.g.: • To access the Integer array **Alarm**'s seventh bit, type **Alarm.X[6]**.



- To access the 62nd bit of the Integer array variable **Water\_Sales**, type **Water\_Sales.X[61]**.



Also, for **Water\_Sales[1].X[29]**:



As a result, since **Water\_Sales.X[61] = Water\_Sales[1].X[29]**, both can be used to access the 62nd bit of the Integer array **Water\_Sales**.

- To access the 6th byte of the Integer array variable **Water\_Sales**, both **Water\_Sales.B[5]** and **Water\_Sales[1].B[1]** can be used.
- To access the 5th word of the Integer array variable **Water\_Sales**, both **Water\_Sales.W[4]** and **Water\_Sales[2].W[0]** can be used.



**Note:** **Water\_Sales.X[61]** and **Water\_Sales[0].X[61]** have the same meaning.

In the following example, the 3rd bit of the system variable **#Status** is used as a NO instruction variable. The third bit of **#Status** identifies whether the GLC unit has an I/O error or not. Therefore, when the third bit is turned ON, the output coil's **IO\_Error** is turned ON, which provides notification that an I/O error has occurred.



### ■ Accessing a Real Array

Real Arrays can be accessed using array elements. To access the elements of a Real array, the modifier (n) must be attached to each element, which represents the element number. Also, “0” is used for the first element in the array.

E.g.: To access the 5th element in the Real array **SolutionTemperature**, you would use "**SolutionTemperature[4]**".



**Note:** Pro-Control Editor can handle up to 2048 GLC variables. The elements of the array become single variables. Thus, an array with five elements becomes five variables.

A Real array can be used for numerical calculation, tracking of repetitive information and data logging.

E.g.: To record the temperature of a solution every 24 hours in the Real array **Solution\_Temperature**, the structure of data is as follows.

The array consists of 24 Real type elements that correspond to each hour of a day.

Real element 0 corresponds to the temperature data at 0:00.

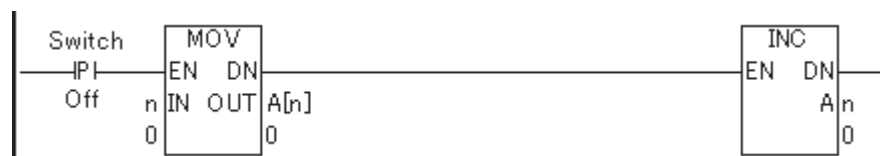
Solution_Temperature[0]
Solution_Temperature[1]
Solution_Temperature[2]
Solution_Temperature[3]
—
—
—
Solution_Temperature[20]
Solution_Temperature[21]
Solution_Temperature[22]
Solution_Temperature[23]

### ■ Accessing an Indirect Array

Array elements[n] can be indirectly accessed by an Integer variable. Numbers in the square brackets [ ] of suffixes such as .X[m], B[m], and W[m] can also be indirectly accessed.

For example, when a switch is operated as shown in the circuit below, with the INC instruction, N increases by one every time the switch is turned on, and a value added by the MOV instruction (Transfer) is assigned to A [N]. The result of the ADD instruction (the sum of "N" and "1"), is then assigned to A[N]. After five scans have been performed, "1" is assigned to A[0], "2" to A[1], "3" to A[2], "4" to A[3], and "5" to A[4].

Note that the initial value of "N" is 0.



# *Memo*

## 3 System Variables

The following table provides a list of the Controller's predefined System Variables.

### 3.1 System Variable List

System Variables are used to display the Controller's current state, and affect its operation. System variables are similar to variables and perform similarly. Since system variables are preprogrammed and defined, they cannot be deleted and their names cannot be changed.

Group	System Variable	Description	Initial Value	Variable Name	
ScanTime	#AvgLogicTime	Displays the average Logic Time (Read, Perform, Write) once every 64 scans. (Unit: ms)	0	Integer	Read Only
	#AvgScanTime	Displays the latest Logic Time (Read, Perform, Write, Display processing) once every 64 scans. (Unit: ms)	0	Integer	
	#LogicTime	Displays the latest Logic Scan Time (Read, Perform, Write). (Unit: ms)	0	Integer	
	#ScanCount	Excluding the current scan, counts the number of scans performed.	0	Integer	
	#ScanTime	Displays the latest Logic Scan Time (Read, Perform, Write, Display processing). (Unit: ms)	0	Integer	
	#WatchdogTime	Displays the Watchdog Timer' value set either in the editor or in offline mode (Unit: ms)	-	Integer	
	#PercentAlloc	Calculates the Percent Scan's percentage. (Unit: %)	0	Integer	Write Only
	#TargetScan	Sets the Constant Scan Time. (Unit: ms)	-	Integer	
Status	#ControllInfo	Not currently used by GLC.	-	Integer	Read Only
	#ForceCount	Counts the number of times a variable is forced ON or OFF.	0	Integer	
	#IOInfo	Not currently used by GLC.	-	Integer	
	#IOStatus	Displays the I/O Driver's condition.	-	Integer [10]	
	#Platform	Indicates the controller's platform.	-	Integer	
	#Status	Indicates controller's current status.	-	Integer	
	#Version	Displays the controller's version data.	-	Integer	
	#FaultCode	Displays the latest error code	-	Integer	
	#FaultRung	Displays the rung where the error occurred.	-	Integer	
	#IOFault	Turns ON when an error occurs.	-	Discrete	
#Overflow	Turns ON when an overflow occurs due to mathematical commands or Real-to-Integer variable conversion.	0	Discrete		

## Chapter 3 – System Variables

Group	System Variable	Description	Initial Value	Variable Name	
Status	#DisableAutoStart	Defines the mode entered when the GLC starts up.	-	Discrete	Write Only
	#Fault	Used to stop the performance of an Error Handler subroutine.	0	Discrete	
	#FaultOnMinor	Controls the completion of the logic performed when a minor error occurs.	0	Discrete	
Command	#Command	Changes the controller's mode.	0	Integer	Write Only
	#Screen	Switches GLC screens by assigning screen numbers. (BIN/BCD)	0	Integer	
Time	#Clock100ms	Create 0.1s clock.	-	Discrete	Read Only
	#Year	Stores Year data as BCD two digits.	-	Integer	
	#Month	Stores Month data as BCD two digits.	-	Integer	
	#Day	Stores Day data as BCD two digits.	-	Integer	
	#Time	Stores Time data as BCD two digits.	-	Integer	
	#WeekDay	Stores Day data as an integer value between 0 and 6	-	Integer	
Others	#EditCount	Not currently used by GLC.	-	Integer	Read Only
	#StopPending	Not currently used by GLC.	-	Discrete	
	#WCLScan	Not currently used by GLC.	-	Integer	
	#WCLStatus	Not currently used by GLC.	-	Integer	
	#LadderMonitor	Starts and runs the GLC Ladder Monitor Feature.	-	Integer	Write Only
	#PercentMemCheck	Not currently used by the GLC.	-	Integer	
	#RungNo	Sets the starting rung number to be displayed by the GLC Ladder Monitor Feature.	-	Integer	
	#StopScans	Not currently used by the GLC.	-	Integer	



### Note:

#Year, #Month, #Day and #Time are saved as the GLC unit's time data. Time data changes are performed via the GLC unit's Initial settings, or the System Data Area's Write settings.

**Reference** Refer to the *GLC Series User Manual (sold separately)*, *GP-PRO/PB III PLC/Device Connection Manual*.



## 3.2 System Variable Details

This section describes each system variable in detail.

### 3.2.1 #AvgLogicTime

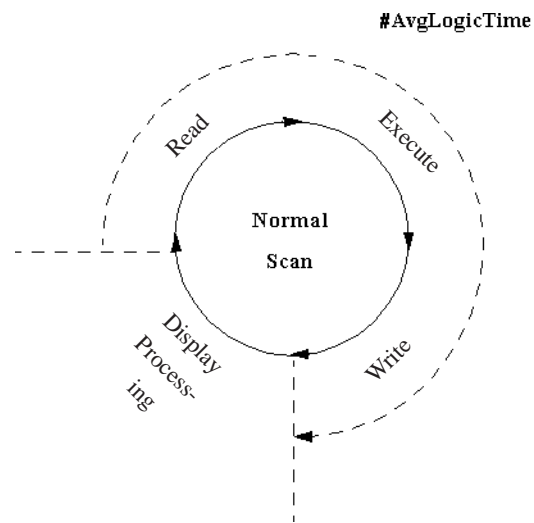
#AvgLogicTime stores the average logic time in ms units.

The average logic time refers to the average time required in one scan, to read I/O, execute the ladder logic program, and read I/O. Every 64 scans, this system variable updates the average logic time since its last calculation.

Variable Type: Integer

Set by: Controller

Read Only



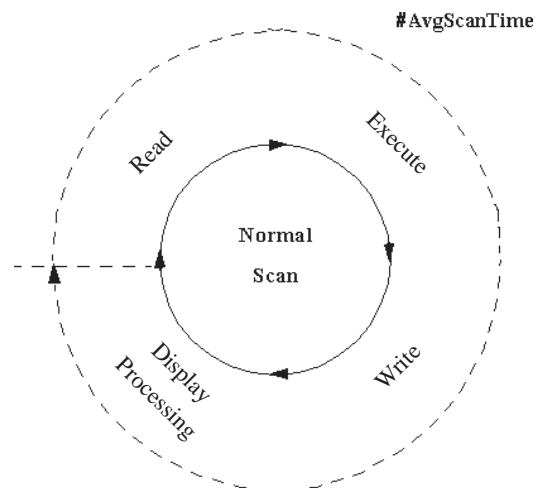
### 3.2.2 #AvgScanTime

#AvgScanTime stores the average amount of time, in milliseconds, that the controller uses to read inputs, execute logic, write outputs, and perform display processing in a single scan. Every 64 scans, this system variable updates the average scan time.

Variable Type: Integer

Set by: Controller

Read Only



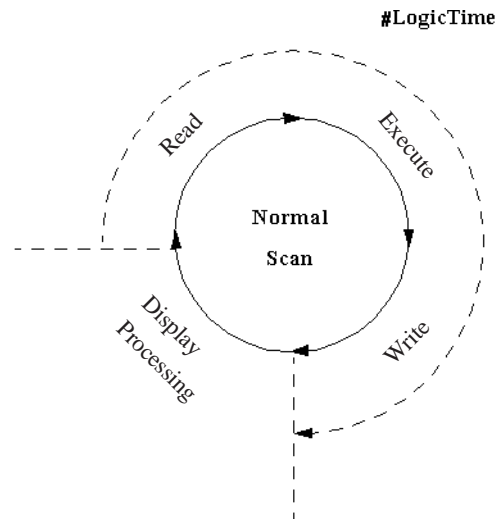
### 3.2.3 #LogicTime

#LogicTime indicates the length of time, in milliseconds, that the controller uses in a single scan to read inputs, execute logic, and write outputs of the previous scan. Logic time does not include the display processing time allowed by the controller for other programs to execute.

Variable Type: Integer

Set by: Controller

Read Only



### 3.2.4 #ScanCount

#ScanCount is a counter incremented by the controller at the end of each scan.

The value range of #ScanCount is 0 – 16#FFFFFFFF. When the counter value exceeds the maximum value (16#FFFFFFFF), the #ScanCount value is incremented again starting at 0.

Variable Type: Integer

Set by: Controller

Read Only



**Note:** Whether or not a logic program is running can be easily checked using #ScanCount.

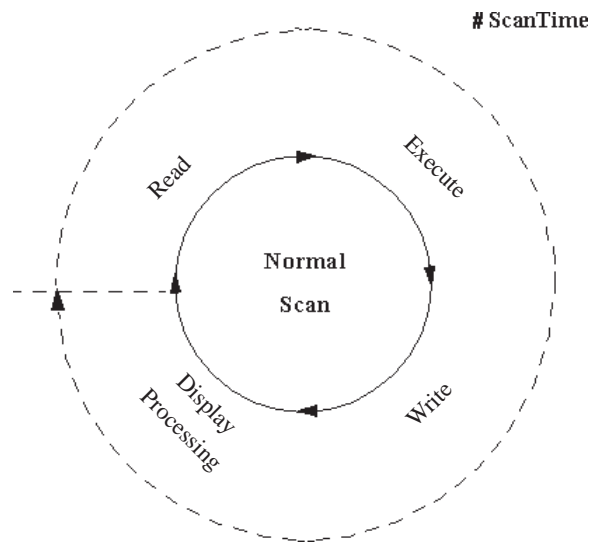
### 3.2.5 #ScanTime

#ScanTime stores the amount of time, in milliseconds, that the controller uses during its last complete scan, to read I/O, execute logic, write I/O, and display processing.

Variable Type: Integer

Set by: Controller

Read Only



### 3.2.6 #WatchdogTime

#WatchdogTime is used to set the value of the watchdog timer, in milliseconds. When #ScanTime exceeds this value, a major fault occurs.

▼ **Reference** ▲ See 9.2 – “Error Codes.”

#WatchdogTime can be set in the initial settings or the configuration settings when the controller is in RUN mode. #WatchdogTime is usually set up in the Setup dialog box.

Variable Type: Integer

Set by: User

Initial Value: 500ms

Writable

### 3.2.7 #PercentAlloc

#PercentAlloc is used when the controller is set to the Percent Scan mode. It sets the percentage of the GLC unit’s total CPU time available to the controller. Set a scan time value in multiples of 10ms.

#PercentAlloc can be set in the initial settings or the configuration settings when the controller is in RUN mode. #PercentAlloc can usually be set up in the Setup dialog box.

▼ **Reference** ▲ See 1.1.2 – “RUN Mode.”

Variable Type: Integer

Set by: User

Range: 0 to 50%

Initial Value: 50

Writable

**3.2.8 #TargetScan**

#TargetScan is used when the controller is set to the Constant Scan mode. The #TargetScan variable is designated in multiples of 10ms units. When the logic time is constant, increasing the value in #TargetScan means that the display processing time will be longer. Decreasing the value in #TargetScan means that the display processing time will be shorter. This is because most of the processing time is used by the controller. #TargetScan can be set in the initial settings or the configuration settings when the controller is in RUN mode. Typically, #TargetScan can be set up in the Setup dialog box.

**Reference** See 1.1.2 – “RUN Mode.”

- Variable Type: Integer
- Set by: User
- Range: 10–2000ms
- Initial Value: 10ms
- Writable

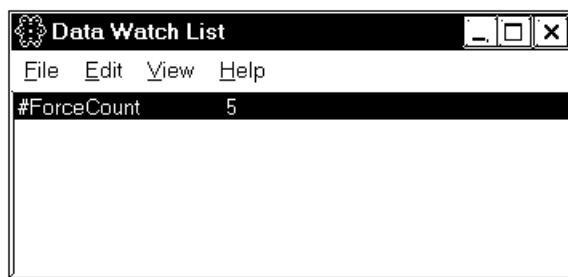
**3.2.9 #ForceCount**

#ForceCount stores the number of variables that are forced ON or OFF in the current ladder program.

**Reference** Refer to the *Pro-Control Editor Operation Manual, Section 3.2 – “Starting and Stopping the Controller.”*

- Variable Type: Integer
- Set by: Controller
- Read Only

The Data Watch List window indicates the five variables that are forced ON or OFF in the logic program.



**3.2.10 #IOStatus**

#IOStatus is set by the I/O driver, and stores the I/O driver’s current status in #IOStatus[1].

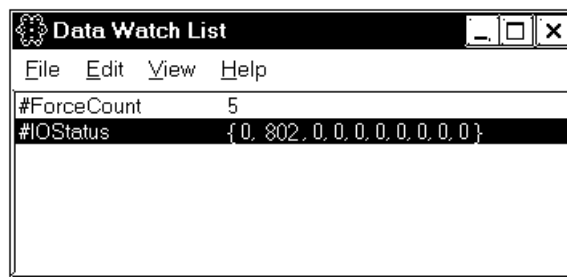
A value of 0 indicates that the I/O is normal. The status indicated by a value other than 0 differs, depending on the I/O driver.

Variable Type: Integer[10]

Set by: Controller

Read Only

The Data Watch List window shows that Error 802 occurred in the I/O driver 1.



**Reference** For I/O driver error code descriptions, see *Chapter 8 – “I/O Drivers.”*

**3.2.11 #Platform**

#Platform is used to store the platform number.

Variable Type: Integer

Set by: Controller

Initial Value: 1

Read Only

**GLC Series**

Value	Platform
16#04	GLC100
16#14	GLC300
16#84	GLC2300
16#44	GLC2400 (Rev. *-None,1) <sup>*1</sup>
16#4C	GLC2400 (Rev. *-Above2) <sup>*1</sup>
16#3C	GLC2500
16#94	GLC2600 (Rev. *-None,1) <sup>*1</sup>
16#9C	GLC2600 (Rev. *-Above2) <sup>*1</sup>

**LT Series**

Value	Platform
16#54	LT Type A
16#64	LT Type B/Type B+
16#74	LT Type C
16#114	LT Type H-AD
16#144	LT Type H-ADP
16#154	LT Type H-ADT

\*1 For how to distinguish "Revisions", refer to "For GLC2400/GLC2600 Users".

**3.2.12 #Status**

#Status indicates the controller’s status. Within #Status system variable:

- Byte 0 indicates the current fault conditions of the controller.
- Byte 1 is used to show the fault status history, and is reset to 0 only when the controller is reset.
- Byte 2 indicates the current operating status of the controller.
- Byte 3 is reserved.

Variable Type: Integer

Set by: Controller

Read Only



**Note:** Intermittent errors can be detected by using the latch fault flag. Be sure to display controller status (#Status) using hexadecimal characters.

**Definition of #Status (32bit)**

For details of each item, refer to the next page.

<b>Byte3</b> Reserved	<b>Byte2</b> Current operation status	<b>Byte1</b> Error status history	<b>Byte0</b> Current error status
--------------------------	--	--------------------------------------	--------------------------------------

When the following fault flags become 1, the corresponding conditions are indicated as follows:

		Fault Flags
<b>Byte0</b>	Bit0	Major fault
	Bit1	Minor fault
	Bit2	I/O fault
	Bit3	Reserved
	Bit4	Read error
	Bit5	Reserved
	Bit6	Scan time error
	Bit7	Reserved

		Latched Fault Flags
<b>Byte1</b>	Bit8	Major fault
	Bit9	Minor fault
	Bit10	I/O fault
	Bit11	Reserved
	Bit12	Read error
	Bit13	Reserved
	Bit14	Scan time error
	Bit15	Reserved

		Controller Status
<b>Byte2</b>	Bit16	Running
	Bit17	I/O Enabled/Disabled
	Bit18	Forces Enabled/Disabled
	Bit19	Paused
	Bit20	Reserved
	Bit 21-23	Reserved

<b>Byte3</b>	Reserved
--------------	----------

*Major fault, Minor fault* : See 3.2.14 – “#FaultCodes.”

*I/O fault* : See 3.2.16 – “#IOFault.”, See 3.2.10 – “#IOStatus.”

*Read error* : The Editor’s program cannot be written to the Controller. This can be due to any validity problem encountered when the Controller evaluates the downloaded program. For example:

- Missing or wrong I/O driver.
- Corrupt program file.

*Scan time error* : Occurs if the average logic time, #AvgLogicTime, exceeds 50% of #TargetScan.

### 3.2.13 #Version

#Version indicates the version number of the controller. #Version is displayed in hexadecimal format.

Variable Type: Integer

Set by: Controller

Read Only

Byte No.	Description	Ver. 1.0.0
Byte3	Major version	01
Byte2	Minor version	00
Byte1	Reserved	–
Byte0	Reserved	–

**3.2.14 #FaultCode**

#FaultCode identifies the most recent fault status. A controller resets all these values to 0.

**Reference** See 8.2 – “Error Codes.”

Variable Type: Integer

Set by: Controller

Read Only

Code	Type	Cause
0	Normal	No fault.
1	Minor	Overflow resulting from a mathematical operation or a Real-to-Integer conversion.
2	Major	Array reference is out of bounds.
3	Major	Bit reference of the Integer (32 bits) is out of bounds.
4	Major	Stack overflow.
5	Major	Invalid instruction code.
6		Reserved by the system
7	Major	Scan time exceeds watchdog time.
8		Reserved by the system
9	Major	Software error – typically a malfunctioning custom function block – may require a system reboot to recover.
10		Reserved by the system
11		Reserved by the system
12	Minor	BCD/BIN conversion error
13	Minor	ENCD/DECO error* <sup>1</sup>
14		Reserved by the system
15	Minor	Backup memory's logic program (SRAM) is damaged. Logic program in FEPRAM will now be executed.* <sup>1</sup>

1. An error occurs only in the GLC2000 Series and LT Series units.



In the Data Watch List window, #FaultCode 7 is displayed.

This indicates that the scan time has exceeded the watchdog time.



**3.2.15 #FaultRung**

#FaultRung stores the rung number where a fault occurred. #FaultRung is set to 0 if there are no faults.

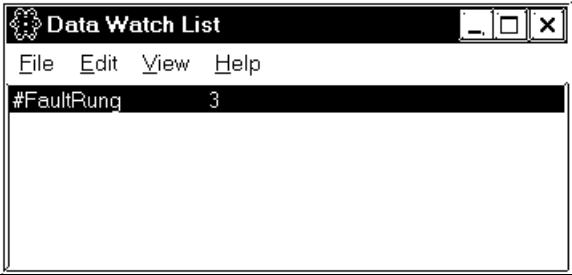
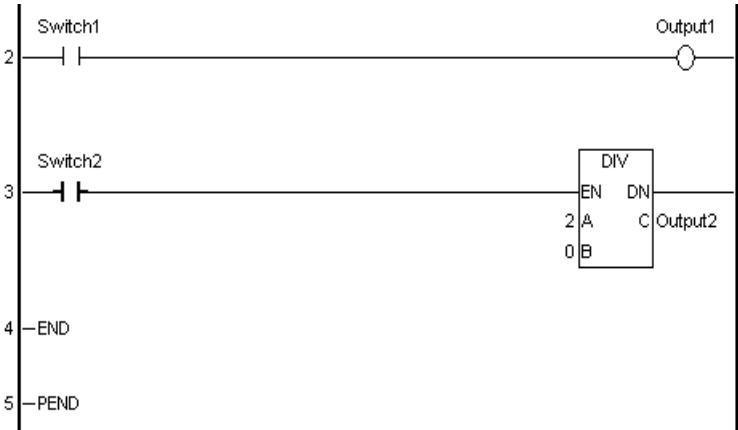
The following example shows when an error occurred at Rung 3.

This error is caused by subtracting the Integer by 0 when DIV Instruction is executed. This error remains until the next error occurs or the controller is reset.

Variable Type: Integer

Set by: Controller

Read Only



### 3.2.16 #IOFault

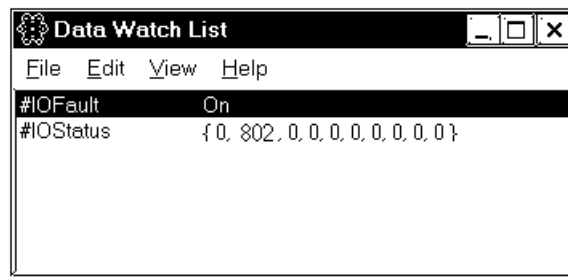
#IOFault turns ON when an I/O fault occurs with the I/O driver. This error remains until it is restored or the controller is reset. Check the #IOStatus variable for detailed status of the I/O driver.

#IOFault is able to confirm from Data Watch List window.

Variable Type: Discrete

Set by: Controller

Read Only



**Reference** For I/O driver error code descriptions and troubleshooting, see **Chapter 8 – “I/O Drivers.”**

### 3.2.17 #Overflow

#Overflow turns ON when a mathematical fault occurs. #Overflow stays ON until the next mathematical instruction or conversion.

Mathematical faults include instruction overflows, Real-to-Integer conversion overflows, and divide by zero errors.

When a mathematical fault occurs, a minor fault also occurs, which executes an ErrorHandler subroutine, if one exists.

**Reference** See 9.2 – “Error Codes.”

The ErrorHandler subroutine is an error process subroutine, and must first be created under the name “ErrorHandler.”

The value in the #Fault system variable defines whether the controller will stop or continue execution of the logic program.

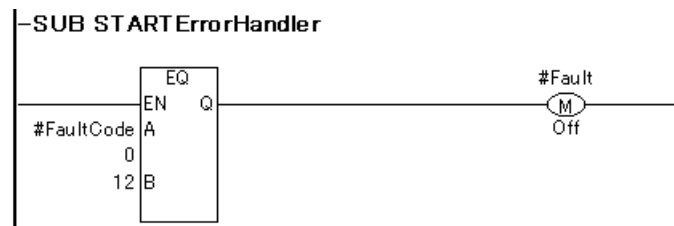
**Reference** See 3.2.19 – “#Fault.”

Variable Type: Discrete

Set by: Controller

Read Only

In the following example, the ErrorHandler subroutine detects BCD/BIN conversion errors and stops execution of the logic program.



**Note:** If an overflow does not occur during Real-to-Integer conversion, #Overflow will not turn ON.

### 3.2.18 #DisableAutoStart

If the power is turned ON while #DisableAutoStart is ON, the controller starts up in the STOP mode.

If the power is turned ON while #DisableAutoStart is OFF, the controller starts up in the state that it was in (START or STOP) prior to shutdown.

The above settings are enabled only when the Controller State setting is set to Default in the GLC unit's initial settings.

Variable Type: Discrete

Set by: User

Initial Value: OFF

Writable

### 3.2.19 #Fault

#Fault is referred to by the controller as to whether the logic program will stop or continue to execute at the completion of the ErrorHandler subroutine.

By turning #Fault ON, the controller will be able to stop executing the logic program.

**Reference** For information about ErrorHandler subroutines, see 3.2.17 – “#Overflow.”

Variable Type: Discrete

Set by: User

Initial Value: OFF

Writable



**Note:** #Fault has no meaning when there is no ErrorHandler subroutine.

**3.2.20 #FaultOnMinor**

#FaultOnMinor is checked by the controller to determine whether the logic program will stop or continue to execute when a minor fault occurs and there is no ErrorHandler subroutine in the logic program.

Turning ON #FaultOnMinor allows you to pause the execution of a ladder logic program.

**Reference** See 9.2 – “Error Codes.”

*For information about the ErrorHandler subroutine, see 3.2.17 – “#Overflow.”*

Variable Type: Discrete

Set by: User

Initial Value: OFF

Writable

**3.2.21 #Command**

#Command is an Integer variable used as a controller command. After the controller reads #Command, its value is reset to 0, but bit 7 remains unchanged. When multiple bits are ON, the lowest bit takes precedence.

Variable Type: Integer

Set by: User

Initial Value: OFF (all bits)

Writable

Bit0 (=1)	Stop Controller
Bit1 (=2)	Run Controller
Bit2 (=4)	Reset Controller
Bit3 (=8)	Execute single scan
Bit4 (=16)	Continue
Bit5 (=32)	Pause
Bit7 (=128)	Enable I/O

**3.2.22 #Screen**

#Screen is used to change GLC unit screens. This screen change variable's operation differs from [Change Screen Check] as follows.

If the [Change Screen Check] feature is enabled and the screen change number is entered in #Screen, after the screen change is completed the value is reset to "0".

**Reference** *Pro-Control Editor Operation Manual Ch. 2 Creating a Program*

Variable Type: Integer

Set by: User/Controller

Initial Value: 0

Writable



**Note:**

- The screen number set in #Screen defines which base screen to display. This number is not the currently displayed screen number. The currently displayed screen number is stored in the System Data Area — in LS[15] when using the Memory Link Communication Method, and in LS[0] when using the Direct Access Communication Method.

- #Screen is a write-only system variable. Therefore, DO NOT use #Screen in applications that determine screen change, etc. To determine the screen number currently displayed, refer to the system data area (LS area).

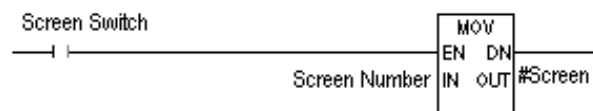
**Reference** *see 5.2 - "LS Area Refresh Settings"*

- #Screen is available only with the GLC2000 Series and LT Series units.



**Important**

- **When changing screens using #Screen, use the #Screen in the logic program. Do NOT write directly to the #Screen using touch input. Change screens using the logic program diagram below as an example.**



- **After power is turned ON, if the #Screen variable is used to change the initial screen, be sure to wait more than 200ms or use the LSS[0].x[3] (LS2032's bit 3) bit rising (0 -> 1) timing.**

**3.2.23 #Clock100ms**

#Clock100ms generates clock in milliseconds. Do not change the clock value since this is used for read in only. An initial value is undefined.

Variable Type: Discrete

Set by: Controller

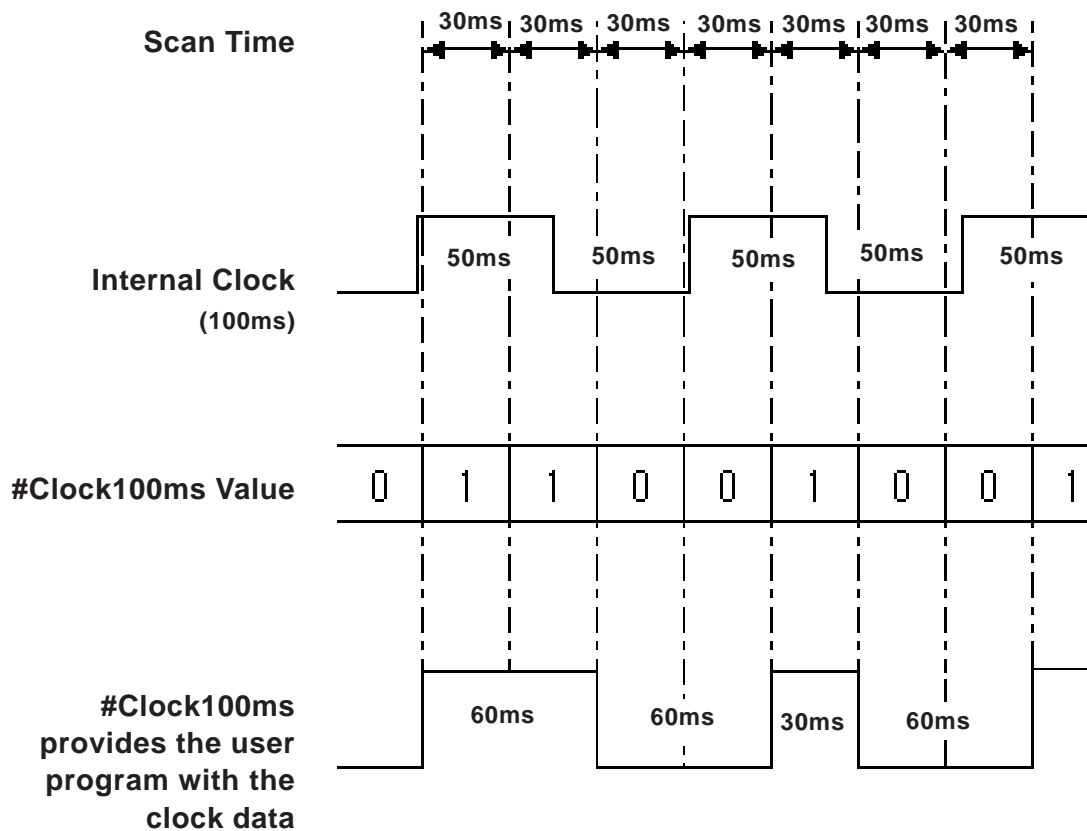
Read Only



**Note:**

- If a GLC unit’s scan time exceeds 50ms, #Clock100ms clock will not be guaranteed.
- If the #Clock100ms clock reads in the internal clock 100ms at the beginning of each GLC scan, an error will occur.
- #Clock100ms is available only with the GLC2000 Series and LT Series units.

**Scan Time Every 30ms**



- The #Clock100ms includes an approximate amount of error equal to the scan time.

**3.2.24 #Year**

#Year displays Year data, as set in the controller, using two digits in BCD format.

Variable Type: Integer

Set by: Controller

Read Only

Year, Month, Day, and Time data are displayed using the following system variables.

E.g.: July 14, 2001 at 6:19 a.m.

	Year	Month	Day	Time
<b>System Variable</b>	#Year	#Month	#Day	#Time
<b>Value</b>	1	7	14	619



- #Year is available only with the GLC2000 Series and LT Series units.
- When changing the Year, Month, Day, Time, use the System Data area.

**Reference** GP-PRO/PBIII Device/PLC Connection Manual

**3.2.25 #Month**

#Month displays the Month data, as set in the controller, using two digits in BCD format.

Variable Type: Integer

Set by: Controller

Read Only

Year, Month, Day, and Time data are displayed using the following system variables:

E.g.: July 14, 2001 at 6:19 a.m.

	Year	Month	Day	Time
<b>System Variable</b>	#Year	#Month	#Day	#Time
<b>Value</b>	1	7	14	619



- #Month is available only with the GLC2000 Series and LT Series units.
- When changing the Year, Month, Day, Time, use the System Data area.

**Reference** GP-PRO/PBIII Device/PLC Connection Manual

**3.2.26 #Day**

#Day displays the Day data, as set by the controller, using two digits in BCD format.

Variable Type: Integer

Set by: Controller

Read Only

Year, Month, Day, and Time data are displayed using the following system variables:

E.g.: July 14, 2001 at 6:19 a.m.

	Year	Month	Day	Time
<b>System Variable</b>	#Year	#Month	#Day	#Time
<b>Value</b>	1	7	14	619



**Note:** #Day is available only with the GLC2000 Series and LT Series units.

**3.2.27 #Time**

#Time displays Time data, as set in the controller, using four digits in BCD format.

Variable Type: Integer

Set by: Controller

Read Only

Year, Month, Day, and Time data are displayed using the following system variables.

E.g.: July 14, 2001 at 6:19 a.m.

	Year	Month	Day	Time
<b>System Variable</b>	#Year	#Month	#Day	#Time
<b>Value</b>	1	7	14	619



**Note:** #Time is available only with the GLC2000 Series and LT Series units.



**3.2.28 #Weekday**

#Weekday displays present Weekday data, with a value of 0 to 6.

Variable Type: Integer

Set by: Controller

Read Only

#Weekday reflects the number LS2054.

LS2054 cycles a value of 0 to 6 (.. 5→6→0→1→..) at the time-of-day change (23:59 to 00:00).

When the power is plugged in, values of 0 to 6 are reflected by default. It is not necessarily reflect that Sunday is 0 and Monday is 1. To coordinate Weekday data with the actual day of the week, use a keypad input display to enter values of 0 to 6 to LS2062.



**#Weekday is available only with the GLC2000 Series and LT Series units.**

**3.2.29 #LadderMonitor**

#LadderMonitor is used to start up and operate the GLC Ladder Monitor Feature. Each operation is shown in the table below.

Variable Type: Integer

Set by: User

Writable

Bit Location	Items
0	Start/Exit bit
1	Scroll Left
2	Scroll Right
3	Scroll Up
4	Scroll Down
5	Screen/Rung changeover (0: Screen; 1: Rung)
6	Decimal/Hexadecimal change (0: Decimal; 1: Hexadecimal)
7	Reserved
8	Search
9	Up
10	Down
11	Variable Search (0: Do NOT perform; 1: Perform)
12	Variable Selection (List Display)
13	Instruction Search (0: Do NOT perform; 1: Perform)
14	Instruction Selection (List Display)
15	Reserved
16	Controller (0: STOP; 1: RUN)
17	Clear Designated Search Item
18	Reserved
19	Reserved
20	Screen (0: Others; 1: GLC Ladder Monitor Normal Display)
21–30	Reserved
31	GLC Ladder Monitor is starting up.



**Note:** #LadderMonitor is available only with the GLC2000 Series and LT Series units.

**3.2.30 #RungNo**

When the GLC Ladder Monitor is running, the starting rung number is stored in #RungNo. If the GLC Ladder Monitor is not running, writing the rung number in #RungNo, turns the GLC Ladder Monitor Starting Bit (#LadderMonitor's bit 0) from OFF to ON, the GLC Ladder Monitor starts up using the assigned rung number as the starting rung.

Variable Type: Integer

Set by: User

Read Only (When GLC Ladder Monitor is started up.)

Writable (When GLC Ladder Monitor is not started.)



**#RungNo is available only with the GLC2000 Series and LT Series units.**

# *Memo*

# 4 Instructions

## 4.1 Instruction List

The Instructions supported by the Pro-Control Editor software are as follows. This chapter describes instructions used by the Pro-Control Editor. You cannot use some instructions depending on the model and the revision. For details, refer to "For GLC2400/GLC2600 Users" and "Appendix Instruction List".

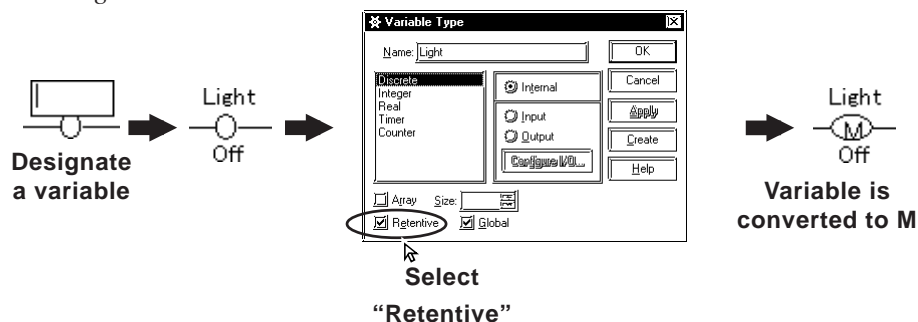
### ■ Discrete Instructions

Instruction	Type	Symbol	Function
NO	Normally Open	$\text{—} \text{   } \text{—}$	Allows power to pass when the contact turns ON.
NC	Normally Closed	$\text{—} \text{   } \text{—} \text{ / }$	Allows power to pass when the contact turns OFF.
OUT/M <sup>*1</sup>	Output Coil / Retention Coil	$\text{—} \text{ ( ) } \text{—} / \text{—} \text{ (M) } \text{—}$	Turns physical output devices or internal discrete variables and expressions ON or OFF.
NEG/NM <sup>*1</sup>	Negated Coil / Negated Retention Coil	$\text{—} \text{ ( / ) } \text{—} / \text{—} \text{ ( / M) } \text{—}$	Turns a variable OFF if the coil receives power, and ON if it does not receive power.
SET/SM <sup>*1</sup>	Latch Coil / Latch Retention Coil	$\text{—} \text{ ( S ) } \text{—} / \text{—} \text{ ( S M) } \text{—}$	Turns a variable ON if the coil receives power. Power remains ON until it receives another explicit instruction.
RST/RM <sup>*1</sup>	Unlatch Coil / Unlatch Retention Coil	$\text{—} \text{ ( R ) } \text{—} / \text{—} \text{ ( R M) } \text{—}$	Turns a variable OFF if the coil receives power. Power remains OFF until it receives another explicit instruction.
PT <sup>*2</sup>	Positive Transition	$\text{—} \text{   } \text{—} \text{ / }$	Allows power to pass if the variable was OFF during the previous scan, but is currently ON.
NT <sup>*2</sup>	Negative Transition	$\text{—} \text{   } \text{—} \text{ / }$	Allows power to pass if the variable was ON during the previous scan, but is currently OFF.

- For the instructions listed above, when a variable is retentive, it automatically changes to one of the right-side instructions. Therefore, when inserting instructions in this screen, be sure to use one of the left-side (non-retentive) instructions.

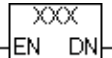
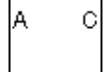

**Reference** See 2.2 – “Variable Types.”

In the following example, when an OUT instruction's variable is retentive, the screen icon changes to M.

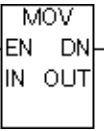
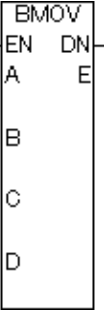
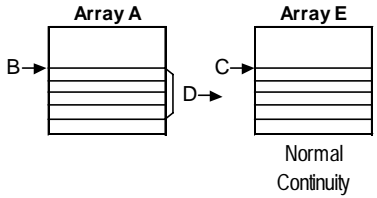

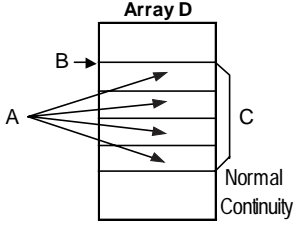
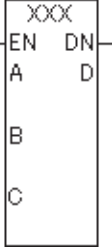
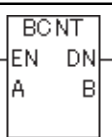


- There is a limit to the maximum number of PT/NT instructions. See “Pro Control Editor Operation Manual - Ch.3 Running the Ladder Logic Program”

■ Arithmetic Operation Instructions

Instruction	Type	Symbol	Function
AND	Logical Multiply		A and B → C Normal Continuity
OR	Logical Add		A or B → C Normal Continuity
XOR	Exclusive Logical Add		A xor B → C Normal Continuity
NOT	Bit Negation		$\bar{A}$ → C Normal Continuity

■ Movement Instructions\*1

Instruction	Type	Symbol	Function
MOV	Transfer		IN → OUT Normal Continuity
BMOV	Block Transfer		 Normal Continuity
FMOV	Fill Transfer		 Normal Continuity
SUM	Sum		Sums up B through C, and stores it to D.
AVE	Average		Calculates the average of B through C, and stores it to D.
BCNT	Bit count		Stores the ON bit count of A to B.

1. There are some restrictions on instructions that you can use depending on the model and the revision. Refer to "For GLC2400/GLC2600 Users" and "Appendix Instruction List".

■ Shift Instructions\*1

Instruction	Type	Symbol	Function
ROL	Rotate Left		<span style="float: right;">▶ C Normal Continuity</span>
ROR	Rotate Right		<span style="float: right;">▶ C Normal Continuity</span>
SHL	Shift Left		<span style="float: right;">▶ C Normal Continuity</span>
SHR	Shift Right		<span style="float: right;">▶ C Normal Continuity</span>
RCL	Left rotation with carry		<span style="float: right;">▶ D Normal Continuity</span>
RCR	Right rotation with carry		<span style="float: right;">▶ D Normal Continuity</span>
SAL	Arithmetic shift left		<span style="float: right;">▶ C Normal Continuity</span>
SAR	Arithmetic shift right		<span style="float: right;">▶ C Normal Continuity</span>

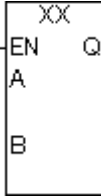
1. There are some restrictions on instructions that you can use depending on the model and the revision. Refer to "For GLC2400/GLC2600 Users" and "Appendix Instruction List".

■ Mathematical Instructions\*1

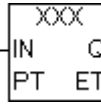
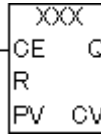
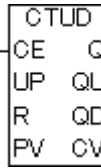
Instruction	Type	Symbol	Function
ADD	Add		$A + B \rightarrow C$ Normal Continuity
SUB	Subtract		$A - B \rightarrow C$ Normal Continuity
MUL	Multiply		$A \times B \rightarrow C$ Normal Continuity
DIV	Divide		$A \div B \rightarrow C$ Normal Continuity
MOD	Residual Processing		$A \% B \rightarrow C$ Normal Continuity
INC	Increment		$A + 1 \rightarrow A$ Normal Continuity
DEC	Decrement		$A - 1 \rightarrow A$ Normal Continuity
SQRT	Square root		$\sqrt{A} \rightarrow B$ Normal Continuity

1. There are some restrictions on instructions that you can use depending on the model and the revision. Refer to "For GLC2400/GLC2600 Users" and "Appendix Instruction List".

■ Comparison Instructions

Instruction	Type	Symbol	Function
EQ	Equal To (=)		When A = B, Continuity
GT	Greater Than (>)		When A > B, Continuity
LT	Less Than (<)		When A < B, Continuity
GE	Greater Than or Equal To (>=)		When A > or = B, Continuity
LE	Less Than or Equal To (<=)		When A < or = B, Continuity
NE	Not Equal (<>)		When A < > B, Continuity

■ Timer and Counter Instructions

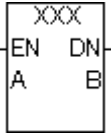
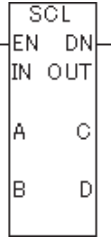
Instruction	Type	Symbol	Function
TON	ON Delay Timer		See 4.2.41 – "TON (ON Delay Timer)."
TOF	OFF Delay Timer		See 4.2.42 – "TOF (OFF Delay Timer)."
TP	Timer Pulse		See 4.2.43 – "TP (Timer Pulse)."
CTU	UP Counter		See 4.2.44 – "CTU (UP Counter)."
CTD	DOWN Counter		See 4.2.45 – "CTD (DOWN Counter)."
CTUD	UP/DOWN Counter		See 4.2.46 – "CTUD (UP/DOWN Counter)."



**The Timer Instruction includes an approximate amount of error equal to the scan time.**

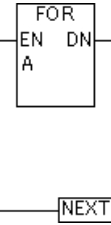


■ **Convert Instructions**<sup>\*1</sup>

Instruction	Type	Symbol	Function
BCD	BCD Conversion		A → BCD conversion → B Normal Continuity
BIN	Binary Conversion		A → Binary conversion → B Normal Continuity
ENCO	Encode		A → Encode conversion → B Normal Continuity
DECO	Decode		A → Decode conversion → B Normal Continuity
RAD	Radian conversion (Degrees → Radians)		A → Radian conversion → B Normal Continuity
DEG	Degree conversion (Radian → Degree)		A → Degree conversion → B Normal Continuity
SCL	Scale Conversion		Converts IN value according to the ratio of an input lower-upper limit range (between A and B) to an output lower-upper limit range (between C and D), and outputs the result to OUT.  Normal Continuity

1. There are some restrictions on instructions that you can use depending on the model and the revision. Refer to "For GLC2400/GLC2600 Users" and "Appendix Instruction List".

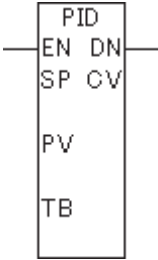
■ **Program Control Instructions**<sup>\*1</sup>

Instruction	Type	Symbol	Function
JMP	Jump	->>label name	Jumps to a label
JSR	Jump to Subroutine	->>Subroutine Name<<-	Jumps to subroutine
RET	Return from Subroutine	<-RETURN>-	Returns to called JSR command.
FOR, NEXT	Repeat		Repeats execution of the logic program between FOR and NEXT for the number of times assigned at A.

1. There are some restrictions on instructions that you can use depending on the model and the revision. Refer to "For GLC2400/GLC2600 Users" and "Appendix Instruction List".

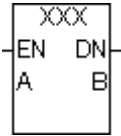
## Chapter 4 – Instructions

### ■ Special Instructions\*1\*2

Instruction	Type	Symbol	Function
PID	PID Calculation		<p><u>When EN is energized:</u> SP and PV perform the PID calculation, and output via CV.</p> <p><u>When EN is not energized:</u> TB and CV go to MOV.</p>

1. There are some restrictions on instructions that you can use depending on the model and the revision. Refer to "For GLC2400/GLC2600 Users" and "Appendix Instruction List".
2. A maximum of 100 special instructions can be used within a project.

### ■ Function Control Instructions\*1

Instruction	Type	Symbol	Function
SIN	sine function		A (Radians) $\rightarrow$ sin(A) $\rightarrow$ B Normal Continuity
COS	cosine function		A(Radians) $\rightarrow$ cos(A) $\rightarrow$ B Normal Continuity
TAN	tangent function		A(Radians) $\rightarrow$ tan(A) $\rightarrow$ B Normal Continuity
ASIN	Arc sine		A $\rightarrow$ sin <sup>-1</sup> (A) $\rightarrow$ B[Radian] Normal Continuity
ACOS	Arc cosine		A $\rightarrow$ cos <sup>-1</sup> (A) $\rightarrow$ B[Radian] Normal Continuity
ATAN	Arc tangent		A $\rightarrow$ tan <sup>-1</sup> (A) $\rightarrow$ B[Radian] Normal Continuity
COT	Cotangent		A[Radian] $\rightarrow$ 1/tan(A) $\rightarrow$ B Normal Continuity
EXP	Exponent		A $\rightarrow$ e <sup>A</sup> $\rightarrow$ B Normal Continuity
LN	Natural logarithm		A $\rightarrow$ log <sub>e</sub> A $\rightarrow$ B Normal Continuity

1. There are some restrictions on instructions that you can use depending on the model and the revision. Refer to "For GLC2400/GLC2600 Users" and "Appendix Instruction List".

## 4.2 Instruction Details

This section describes the detail of each instruction. It explains the variable type (integer, integer array and so on) and the operation example in the variable mode, which you can assign to each instruction operand. When you develop the logic program in the fixed variable mode, you should assign the appropriate operand type while comparing the table below with the variable type in the variable mode.

For details of the fixed variable mode, refer to "**Appendix Fixed Variable Mode**" of "**Pro-Control Editor Operation Manual**".

### Devices in the Fixed Variable Mode

Symbol Mark	Device Name	Type	Kind
D	Data Register	Integer	Internal Variable
W	Word Register	Real	Internal Variable
M	Subsidiary Relay	Bit	Internal Variable
T	Timer	Timer	Internal Variable
C	Counter	Counter	Internal Variable
P	PID	Integer Array	Internal Variable
X	Input	Bit	Input
XW		Integer	Input
Y	Output	Bit	Output
YW		Integer	Output



**Note:**

You can use the LS and LSS areas. For details, refer to "**Chapter 5 LS Area Refresh**".

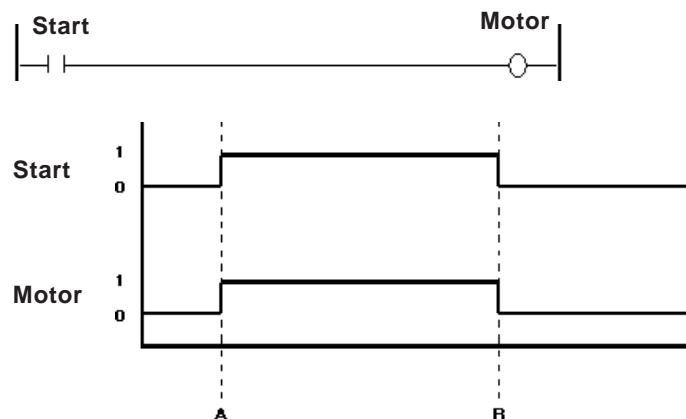
### 4.2.1 NO (Normally Open)

Variable



The NO instruction allows power to pass when the variable is ON.

The following diagram is an example of the NO instruction's function.



A: When the Start variable turns ON, the Motor variable turns ON.

B: When the Start variable turns OFF, the Motor variable turns OFF.

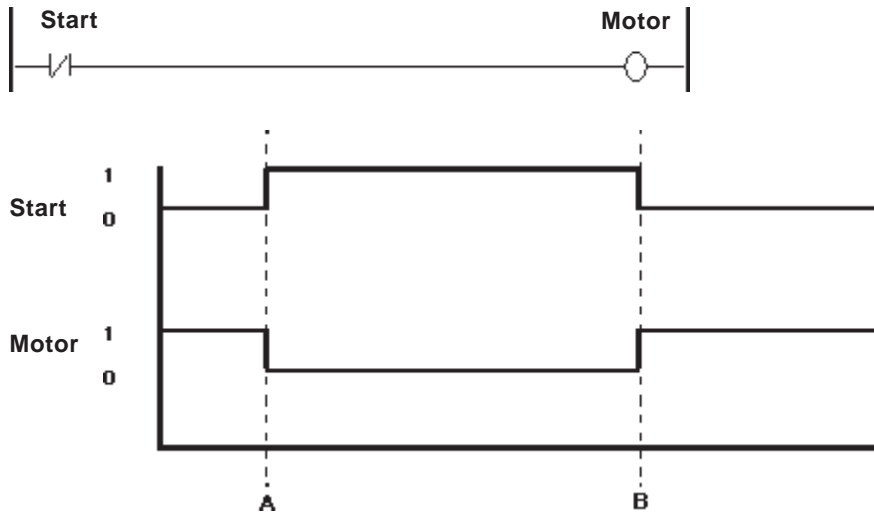
### 4.2.2 NC (Normally Closed)

Variable



The NC instruction allows power to pass when the variable is OFF.

The following diagram is an example of the NC instruction's function.

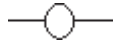


A: When the Start variable turns ON, the Motor variable turns OFF.

B: When the Start variable turns OFF, the Motor variable turns ON.

### 4.2.3 OUT/M (Output Coil)

Variable



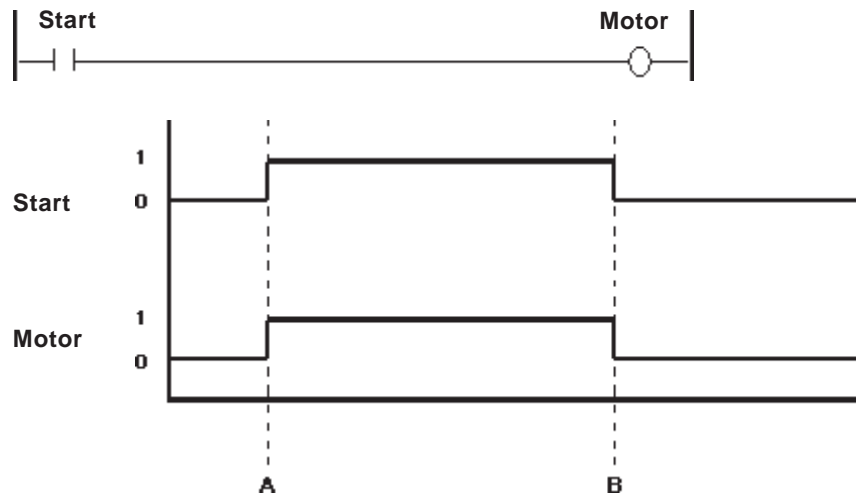
The OUT instruction is used to turn ON/OFF the variables mapped to the I/O, or the Discrete variables in the internal memory .

Since this instruction is a coil-type output instruction, only one instruction can be used for each rung. Other instructions cannot be used on the right side of the output instruction. The output instruction should be placed to the immediate left of the right-most power line.

When the variable mapped to the OUT instruction is retentive, the following symbol is displayed in the logic program.



The following diagram is an example of the OUT instruction's function .



A: When the Start variable turns ON, the Motor variable turns ON.

B: When the Start variable turns OFF, the Motor variable turns OFF.



**Note:**

The OUT instruction can be used only with non-retentive variables. With retentive variables, use the M (Retention Coil) instruction.

**4.2.4 NEG (Negated Coil)**

Variable



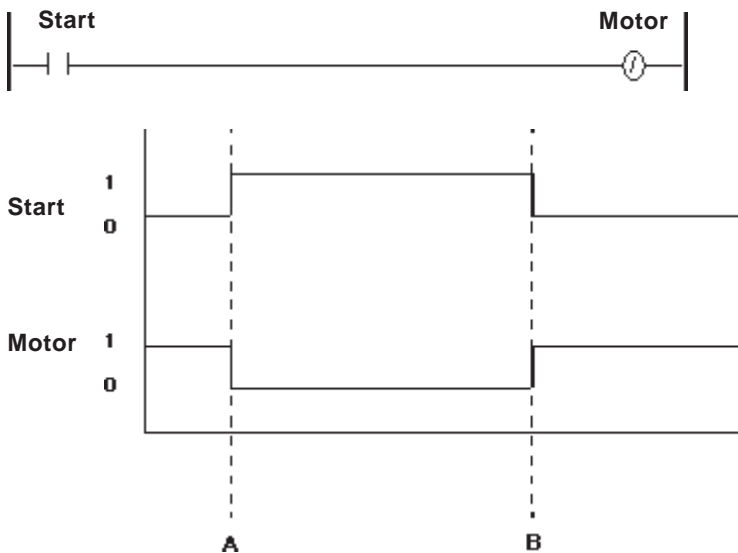
When the NEG instruction is executed, the variable turns OFF when the coil receives power, and ON when the coil does not receive power.

Since this instruction is a coil-type output instruction, only one instruction can be used for each rung. Other instructions cannot be used on the right side of the output instruction. The output instruction should be placed to the immediate left of the right-most power line.

When the variable mapped to NEG instruction is retentive, the following symbol is displayed in the logic program.



The following diagram is an example of the NEG instruction's function.



A: When the Start variable turns ON, the Motor variable turns OFF.

B: When the Start variable turns OFF, the Motor variable turns ON.



**Note:** The NEG instruction can be used only with non-retentive variables.

### 4.2.5 SET (Set Coil)

Variable



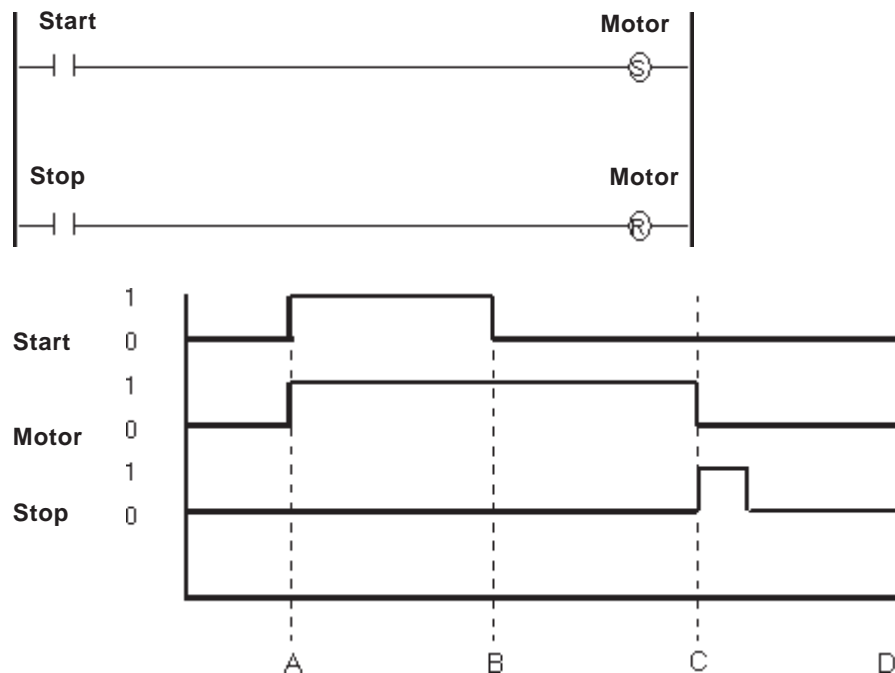
When the SET instruction is executed after the coil receives power, the variable turns ON. The variable will remain ON until explicitly turned OFF by another instruction (such as an RST instruction).

Since this instruction is a coil-type output instruction, only one instruction can be used for each rung. Other instructions cannot be used on the right side of the output instruction. The output instruction should be placed to the immediate left of the right-most power line.

When the variable mapped to SET instruction is retentive, the following symbol is displayed in the logic program.



The following diagram is an example of the SET instruction's function.



A: When the Start variable turns ON, the Motor variable turns ON.

B: The Start variable turns OFF, but does not affect the Motor variable.

C: The Stop variable turns ON, the Motor variable resets.

D: The Motor variable stays reset until the Start variable turns ON.



**Note:**

The SET instruction can be used only with non-retentive variables. With retentive variables, use the SM (Latch Retention Coil) instruction.

**4.2.6 RST (Reset Coil)**

Variable



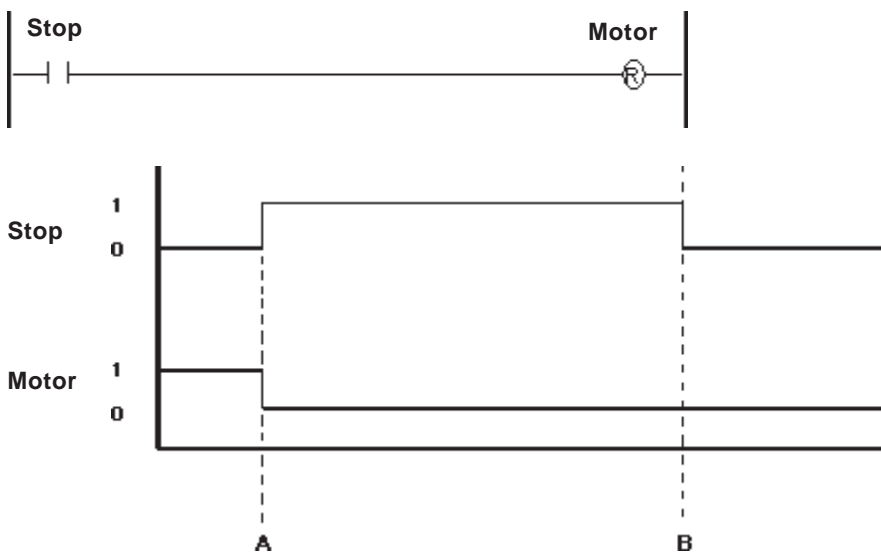
When the coil receives power after the RST instruction is executed, the variable turns OFF. The variable remains OFF until explicitly turned ON by another instruction (such as a SET instruction).

Since this instruction is a coil-type output instruction, only one instruction can be used for each rung. Other instructions cannot be used on the right side of the output instruction. The output instruction should be placed to the immediate left of the right-most power line.

When the variable mapped to the RST instruction is retentive, the following symbol is displayed in the logic program.



The following diagram is an example of the RST instruction's function.



A: When the Stop variable turns ON, the Motor variable resets.

B: When the Stop variable turns OFF, the Motor variable reset by the RST instruction will remain OFF until another instruction turns it ON.



**Note:**

- The RST instruction can be used only with non-retentive variables. With retentive variables, use the RM (Unlatch Retention Coil) instruction.
- Real and Integer variables cannot be reset (set to zero) with an RST instruction.



### 4.2.7 PT (Positive Transition Contact)

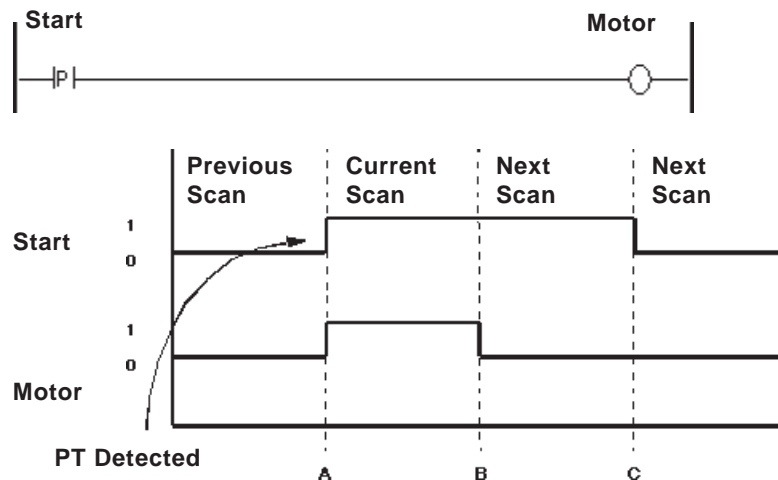
Variable

—|P|—

When the PT instruction is executed, if the variable was OFF during the previous scan but is currently ON, power is allowed to pass for a single scan.

When starting up the program, the state of positive transition contact during the previous scan is considered to have been OFF.

The following diagram is an example of the PT instruction's function.



A: When the Start variable turns ON, the Motor variable turns ON.

B: After one scan (the current scan), the Motor variable turns OFF.

C: Since the rising edge of the variable Start is not detected, the variable Motor remains OFF.

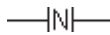


**Be careful when using PT (Rising-type contacts) and NT (Falling-type contacts) instruction operands for indirect addressing of elements in arrays or bit designations via variables.**

**The condition of variables set via operands and used during previous program execution and those variables set for operands are compared and then executed. Therefore, when designated variable values differ, the condition comparison object also differs.**

**4.2.8 NT (Negative Transition Contact)**

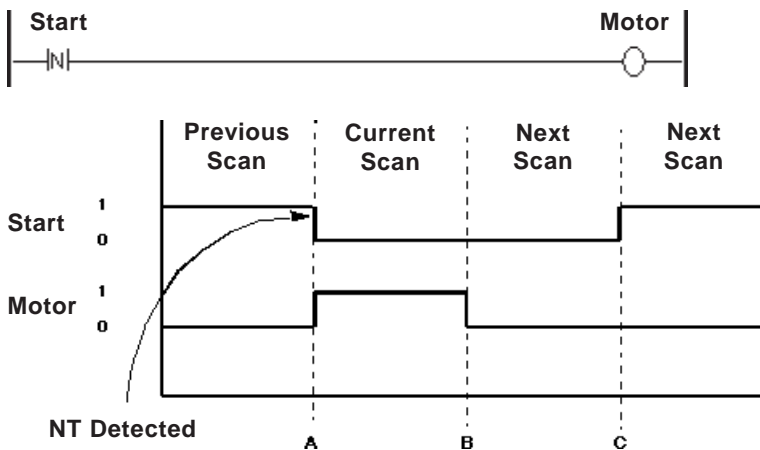
Variable



When the NT instruction is executed, if the variable was ON during the previous scan but is currently OFF, power is allowed to pass for a single scan.

During the first scan, the state of transition during the previous scan is considered to have been OFF. Therefore, the NT instruction does not pass power during the first scan.

The following diagram is an example of the NT instruction's function.



A: When the Start variable turns OFF, the Motor variable turns ON.

B: After one scan, the Motor variable turns OFF.

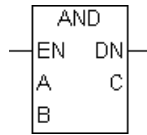
C: Since the falling edge of the variable Start is not detected, the variable Motor remains OFF.



**Be careful when using PT (Rising-type contacts) and NT (Falling-type contacts) instruction operands for indirect addressing of elements in arrays or bit designations via variables.**

**The condition of variables set via operands used during previous program execution and those variables set for operands are compared and then executed. Therefore, when designated variable values differ, the condition comparison object also differs.**

**4.2.9 AND (And)**



When the AND instruction is executed, the bit in C turns ON if the corresponding bit in both A and B is ON. Otherwise, the bit in C is turned OFF.

A	Operator	B	C
ON	AND	ON	ON
ON		OFF	OFF
OFF		ON	OFF
OFF		OFF	OFF

<b>Integer A</b>	0 1 1 0 ... 1 1 0 0
<b>Integer B</b>	1 1 0 0 ... 0 0 0 1
<b>Integer C</b>	0 1 0 0 ... 0 0 0 0

The AND instruction always passes power.

The following table lists the combinations of A, B and C that can be used with an AND instruction.

A	B	C
Integer	Integer	Integer
Integer Array	Integer Array	Integer Array
Integer	Integer Constant	Integer
Integer Array	Integer Constant	Integer Array

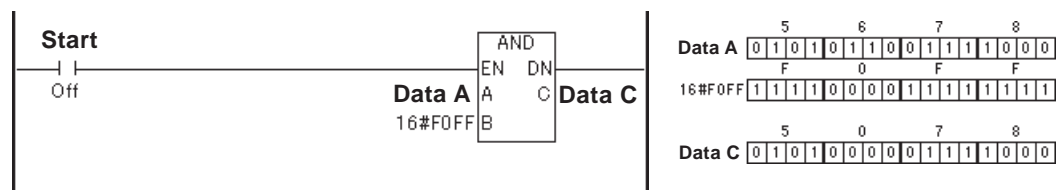
There are three types of AND instructions:

1. When all the variables are not array variables, a simple 32-bit AND operation is performed.
2. When A and C are array variables and B is not an integer array, AND operations are performed for each element of A and B, and the results are stored the corresponding elements of C. Make sure that the size of A and C arrays are the same.
3. When the three variables are arrays of the same size, AND operations of array A and array B are performed. The results are stored in array C.

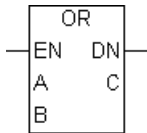
◆ **Operation Example**

When Start is ON, the 3rd digit of the 4-digit BCD data of Data A is masked to "0", and the result is stored in Data C.

E.g.: When Data A is "16#5678" (5678 in hexadecimal system), "16#5078" is stored in Data C.



**4.2.10 OR (Or)**



When the OR instruction is executed, the bit in C turns ON if the corresponding bit in A and/or B is ON. Otherwise, the bit in C is turned OFF.

A	Operator	B	C
ON	OR	ON	ON
ON		OFF	ON
OFF		ON	ON
OFF		OFF	OFF

<b>Integer A</b>	0 1 1 0 ... 1 1 0 0
<b>Integer B</b>	1 1 0 0 ... 0 0 0 1
<b>Integer C</b>	1 1 1 0 ... 1 1 0 1

The OR instruction always passes power.

The following table lists the combinations of A, B and C in which OR instructions can be executed.

A	B	C
Integer	Integer	Integer
Integer Array	Integer Array	Integer Array
Integer	Integer Constant	Integer
Integer Array	Integer Constant	Integer Array

There are three types of OR instructions:

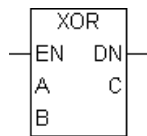
1. When both variables A and B are integers, simple 32-bit OR operation is performed.
2. When A and C are array variables and B is not an integer array, logical OR operations are performed for each element of A and B, and the results are stored the corresponding elements of C. Make sure that the size of A and C arrays are the same.
3. When the three variables are arrays of the same size, logical OR operations of array A and array B are performed. The results are stored in array C.

◆ **Operation Example**

When Start is ON, the result of the logical OR operation of Data A and Data B is stored in Data C.



**4.2.11 XOR (Exclusive OR)**



When the XOR instruction is executed, the bit in C turns ON if the corresponding bit in A or B is ON. Otherwise, the bit in C is turned OFF.

A	Operator	B	C
ON	XOR	ON	OFF
ON		OFF	ON
OFF		ON	ON
OFF		OFF	OFF

<b>Integer A</b>	0 1 1 0 ... 1 1 0 0
<b>Integer B</b>	1 1 0 0 ... 0 0 0 1
<b>Integer C</b>	1 0 1 0 ... 1 1 0 1

The XOR instruction always passes power.

The following table lists the combinations of A, B and C in which XOR instructions can be executed.

A	B	C
Integer	Integer	Integer
Integer Array	Integer Array	Integer Array
Integer	Integer Constant	Integer
Integer Array	Integer Constant	Integer Array

There are three types of XOR instructions:

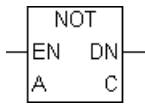
1. When both variables A and B are integers, simple 32-bit exclusive OR operations are performed.
2. When A and C are array variables and B is not an integer array, exclusive OR operations are performed for each element of A and B, and the results are stored the corresponding elements of C. Make sure that the size of A and C arrays are the same.
3. When the three variables are arrays of the same size, exclusive OR operations of array A and array B are performed. The results are stored in array C.

◆ **Operation Example**

When Start is ON, the result of the exclusive OR operation of Data A and Data B is stored in Data C.



**4.2.12 NOT (Bit Invert)**



When the NOT instruction is executed, the bit in C turns ON if the corresponding bit in A is OFF.

The NOT instruction turns OFF the bit in C if the corresponding bit in A is ON.

A	Operator	C
ON	NOT	OFF
OFF		ON

Integer A    0 1 1 0 ... 1 1 0 0

Integer C    1 0 0 1 ... 0 0 1 1

The NOT instruction always passes power.

The following table lists the combinations of A and C in which NOT instructions can be executed.

A	C
Integer	Integer
Integer Array	Integer Array

There are two types of NOT instruction:

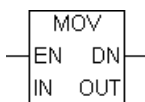
1. When the A variables are integers, simple 32-bit bit conversion is performed.
2. When the A variables are an array, bit conversion is performed for the entire A array. The result is stored in C. Make sure that the size of A and C arrays are the same.

◆ **Operation Example**

When Start is ON, the result of the NOT operation of Data A and Data B is stored in Data C.



**4.2.13 MOV (Transfer)**



When the MOV instruction is executed, IN is copied to OUT.

If IN and OUT are different variable types, the resulting type will be converted to the same type as OUT. To transfer arrays, both IN and OUT must be identical in type and size.

The MOV instruction normally passes power. The following table lists the combinations of IN and OUT in which MOV instructions can be executed.

IN Type	OUT Type
Discrete Array	Discrete array same size as IN
Integer	Variable or Array in Integer or Real
Integer Array	Integer array or variable that is the same size as IN
Integer Constant	Variable or Array in Integer or Real
Real	Variable or Array in Integer or Real
Real Array	Integer array or variable that is the same size as IN
Real Constant	Variable or Array in Integer or Real

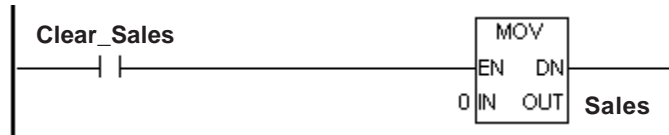


**Note:** #Overflow will turn ON if the operation involves a Real-to-Integer data-type conversion, and the value is too large to transfer. In this case, the result will be undefined.

The following examples illustrate how to use the MOV instruction.

**Example 1: Clear a variable**

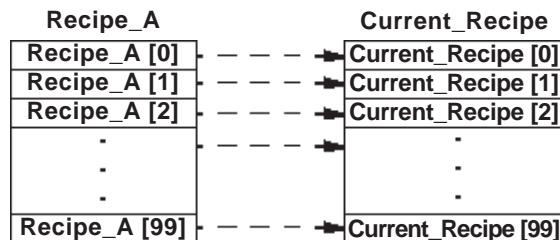
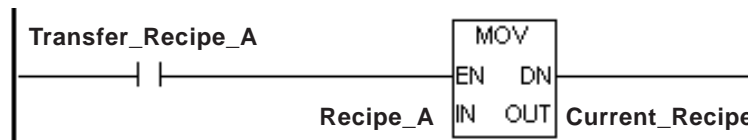
A variable can be cleared with the MOV instruction by transferring a “0” into the variable.



**Example 2: Block-transfer an array**

A block transfer can be performed with the MOV instruction by specifying two arrays of the same type and size.

For example, when transferring Recipe A, which consists of 100 elements, to the Current\_Recipe of the same type and size, simply transfer Recipe A with a MOV instruction.



**Note:** When designating an entire array, enter only the variable names.

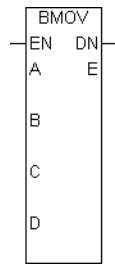
E.g.: OK : Recipe\_A

Not OK : Recipe\_A [\*]

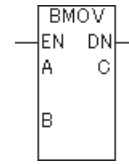
Not OK : Recipe\_A [100]

**4.2.14 BMOV (Block Transfer)**

**Variable Mode**



**Fixed Variable Mode**



- A: Source variable
- B: Start from Array A[B]
- C: To Array E[C]
- D: Amount of data to be transferred
- E: Destination variable

- A: Source device
- B: Amount of data to be transferred
- C: Destination device

**Variable Mode**

When the BMOV instruction is executed, elements of one array can be copied into elements of another array. Specifically, the D elements are copied from index B in array A to index C in array E.

The BMOV instruction is valid for Integer arrays only. When transferring, arrays can be different sizes.

The BMOV instruction always passes power. The following table lists the types of A, B, C, D, and E that can execute BMOV instructions.

A and E	B, C, and D
Integer Array	Integer
	Integer Constant

**Fixed Variable Mode**

Although the operation of the BMOV instruction is equal to that in the variable mode, operands to be set (A, B, E and so on) differ because the fixed variable mode does not have the concept of an "Array".

If you execute the BMOV instruction in the fixed variable mode, elements as many as B starting from the source initial address A will be copied to elements as many as B starting from the destination initial address C.

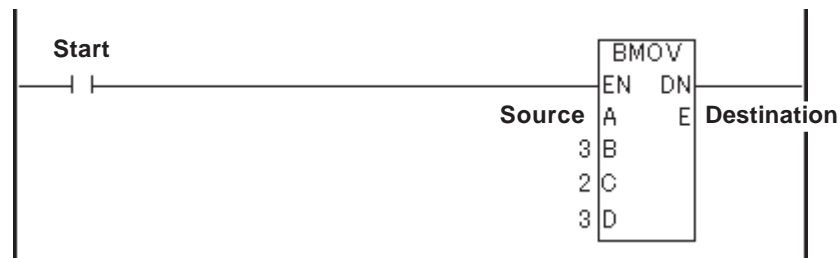
This instruction always continues. A, B and C that you can execute the BMOV instruction are as the following.

A and B	C
Integer, Integer Constant	Integer

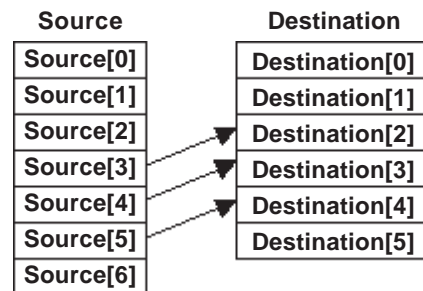


◆ **Example**

When copying, Source [3], [4], and [5] of the seven source integer array elements are copied to Destination [2], [3], and [4] of the six destination array elements. This data transfer is performed as follows.



Source [3] is copied to Destination [2].  
 Source [4] is copied to Destination [3].  
 Source [5] is copied to Destination [4].

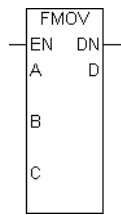


While the program is running, the controller checks whether references to array A and E elements exist in the BMOV instruction. If an invalid array is referred to, a major error occurs and #FaultCode is set to 2.

▼ **Reference** ▼ See 3.2.14 – “#FaultCode.”

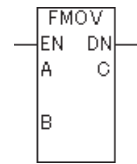
**4.2.15 FMOV (Fill Transfer)**

**Variable Mode**



- A: Source data
- B: Start from Array D[B]
- C: Amount of data to be transferred
- D: Variable name of destination array

**Fixed Variable Mode**



- A: Source device
- B: Amount of data to be transferred
- C: Destination device

**Variable Mode**

When the FMOV instruction is executed, the C elements, starting at index B of Integer array D, are filled with value A.

The FMOV instruction is valid for Integer arrays only. The FMOV instruction always passes power.

The following table lists the types of A, B, C and D in which FMOV instructions can be executed.

A, B, and C	D
Integer	Integer Array
Integer Constant	

**Fixed Variable Mode**

Although the operation of the FMOV instruction is equal to that in the variable mode, operands to be set (A, B, D and so on) differ because the fixed variable mode does not have the concept of an "Array".

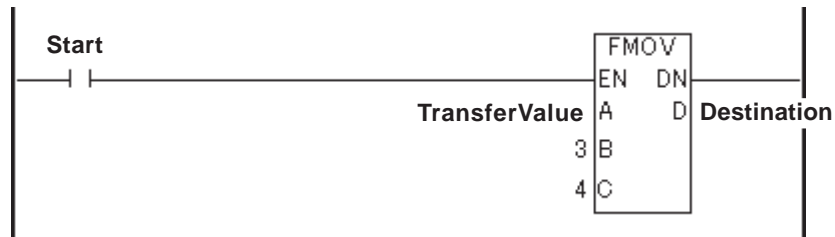
If you execute the FMOV instruction in the fixed variable mode, A will be stored in elements as many as B starting from the destination initial address C.

The following table lists the types of A, B, and C in which FMOV instructions can be executed.

A and B	C
Integer, Integer Constant	Integer

◆ **Example**

When copying, the values are transferred to Destination [3], [4], [5], and [6] of the seven destination array elements. The transfer operates as follows.

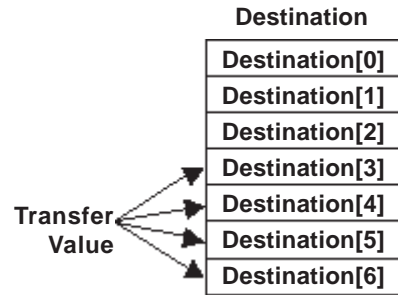


TransferValue is copied to Destination[3].

TransferValue is copied to Destination[4].

TransferValue is copied to Destination[5].

TransferValue is copied to Destination[6].

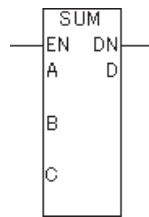


While the program is running, the controller checks whether references to array D elements exist in the FMOV instruction. If an invalid array is referred to, a major error will occur and #FaultCode is set to 2.

▼ **Reference** ▲ See 3.2.14 – “#FaultCode.”

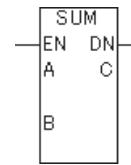
**4.2.16 SUM (Sum Total)**

**Variable Mode**



- A: Array to calculate the sum total
- B: Initial number of the array element range to be calculated (offset)
- C: Element count to be calculated
- D: Variable name of the storing destination

**Fixed Variable Mode**



- A: Device to calculate the sum total
- B: Number of elements to be calculated
- C: Storing destination device

**Variable Mode**

If you execute the SUM instruction, it stores the sum total of elements as many as C starting from B in the array A[n] to D. This instruction always continues.

A, B, C and D that you can execute the SUM instruction are as the following.

Type of A	Type of B	Type of C	Type of D
Integer Array	Integer, or Integer Constant	Integer, or Integer Constant (excluding zero or below)	Integer

**Fixed Variable Mode**

Although the operation of the SUM instruction is equal to that in the variable mode, operands to be set (A, B, D and so on) differ because the fixed variable mode does not have the concept of an "Array".

If you execute the SUM instruction in the fixed variable mode, the sum total of elements as many as B starting from the initial address A will be stored in C.

A, B and C that you can execute the SUM instruction are as the following.

Type of A and C	Type of B
Integer	Integer, Integer Constant



**Note:**

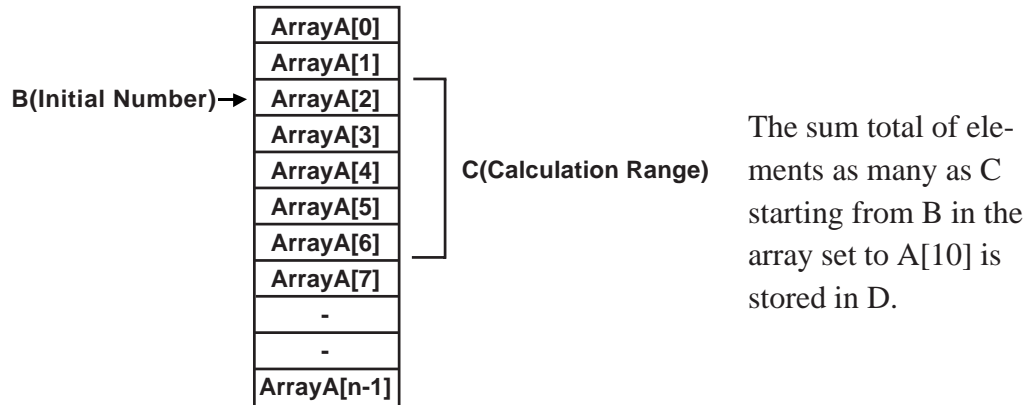
- If the number of elements to be calculated is equal to 0 (zero), it means the sum total of 0 (zero) elements and the result will be equal to 0 (zero).
- If the result is out of range, which can be expressed in the variable type, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

◆ **Example**

The following describes the case when "A[10]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, B=2, C=5".

$$D=(2+3+4+5+6)=20$$

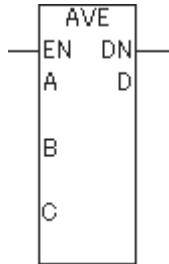


The controller checks whether the SUM instruction refers to any inexistent elements of the array A during RUN or not. If an invalid array is referred to, a major error will occur and #FaultCode will be set to "2".

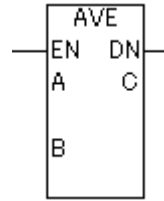
▼ **Reference** ▲ See 3.2.14 “#FaultCode.”

**4.2.17 AVE (Average)**

**Variable Mode**



**Fixed Variable Mode**



- A: Array to calculate the average
- B: Initial number of the array element range to be calculated (offset)
- C: Element count to be calculated
- D: Variable name of the storing destination

- A: Device to calculate the average
- B: Number of elements to be calculated
- C: Storing destination device

**Variable Mode**

If you execute the AVE instruction, it stores the average of elements as many as C starting from B in the array A[n] to D. This instruction always continues.

The following table lists the types of A, B, C and D in which AVE instructions can be executed.

A	B	C	D
Integer Array	Integer, or Integer Constant	Integer, or Integer Constant (excluding zero or below)	Real

**Fixed Variable Mode**

Although the operation of the AVE instruction is equal to that in the variable mode, operands to be set (A, B, D and so on) differ because the fixed variable mode does not have the concept of an "Array".

If you execute the AVE instruction in the fixed variable mode, the average of elements as many as B starting from the initial address A will be stored in C.

The following table lists the types of A, B, and C in which AVE instructions can be executed.

A	B	C
Integer	Integer, Integer Constant	Real



**Note:**

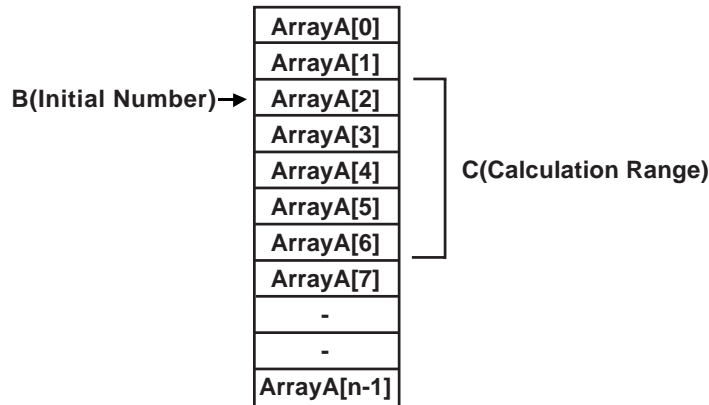
- Decimals are omitted.
- If the result is out of range, which can be expressed in the variable type, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

◆ Example

The following describes the case when "A[10]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, B=2, C=5".

$$D=(2+3+4+5+6)/5=4$$



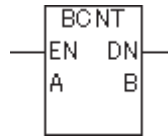
The controller checks whether the AVE instruction refers to any inexistent elements of the array A during RUN or not. If an invalid array is referred to, a major error will occur and #FaultCode will be set to "2".

▼ **Reference** ▲ See 3.2.14 “#FaultCode.”

**4.2.18 BCNT (Bit Count)**

A: Variable name to be calculated

B: Storing destination variable name



If you execute the BCNT instruction, the ON bit count of A will be stored in B.

This instruction always continues.

The following table lists the types of A and B in which BCNT instructions can be executed.

A	B
Integer	Integer
Integer Constant	Integer



**Note:**

- If the result B is out of range, which can be expressed in the variable type of B, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

◆ **Example**

The following describes the case when "A=38" (decimal) and "A=100110" (binary).

B=3 (decimal)



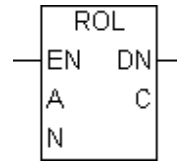


### 4.2.19 ROL (Rotate Left)

A: Variable name to be rotated

N: Number of bit positions to shift

C: Destination variable



The ROL instruction left-shifts the bits in A by N positions. Bits are rotated from the left end (most significant bit) to the right end (least significant bit). The result is placed in C.

The ROL instruction always passes power.

The following table lists the types of A, N, and C in which ROL instructions can be executed.

A	N	C
Integer	Integer or Integer Constant	Integer
Integer Array	Integer or Integer Constant	Integer Array is same size as A
Integer Constant	Integer or Integer Constant	Integer

There are two types of ROL instructions:

1. If both A and C are Integers, a simple 32-bit rotation is performed. N must range from 0 to 31.
2. If both A and C are Integer arrays of the same size, the array is treated as a large Integer.

Bits are shifted from one element to the next, rather than rotating only within each element. N must range from 0 to  $[(32 \times \text{array size}) - 1]$ , inclusive.



**Note:**

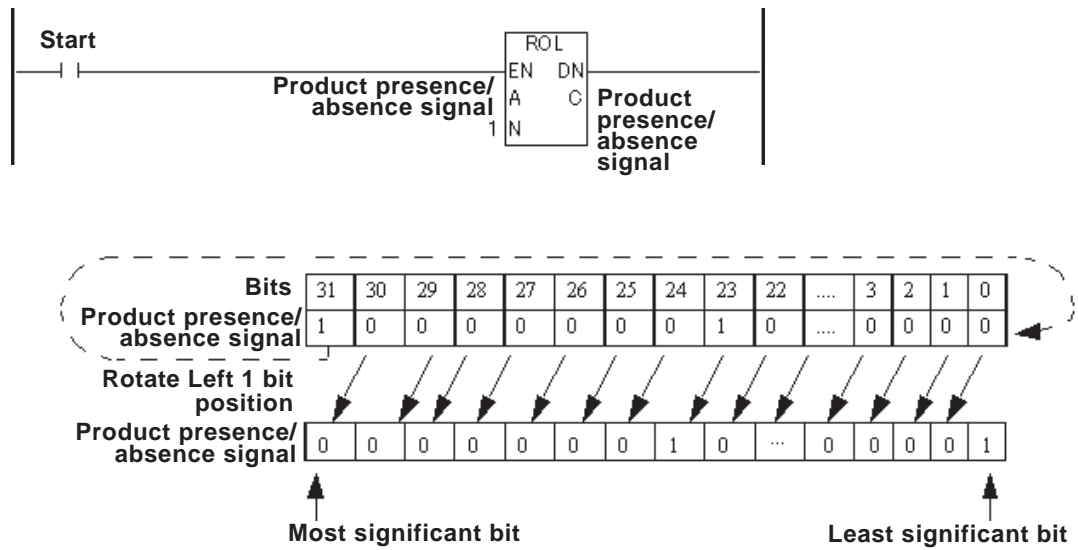
#Overflow is turned ON if N is out of range. The result is undefined.

**Reference** See 3.2.17 “#Overflow.”

## Chapter 4 – Instructions

### ◆ Example

The following example describes the operation of a 1-bit rotation using a product presence/absence signal.

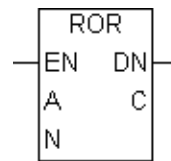


### 4.2.20 ROR (Rotate Right)

A: Variable name to be rotated

N: Number of bit positions to shift

C: Destination variable



The ROR instruction right-shifts the bits in A by N positions. Bits are rotated from the right end (least significant bit) to the left end (most significant bit). The result is placed in C.

The ROR instruction always passes power.

The following table lists the types of A, N, and C in which ROR instructions can be executed.

A	N	C
Integer	Integer or Integer Constant	Integer
Integer Array	Integer or Integer Constant	Integer Array is same size as A
Integer Constant	Integer or Integer Constant	Integer

There are two types of ROR instruction.

1. If neither A nor C is an array, a simple 32-bit rotation is performed. N must range from 0 to 31.
2. If both A and C are Integer arrays of the same size, the array is treated as a large Integer.

Bits are shifted from one element to the next, rather than rotating only within each element. N must range from 0 to [(32 x array size) – 1], inclusive.

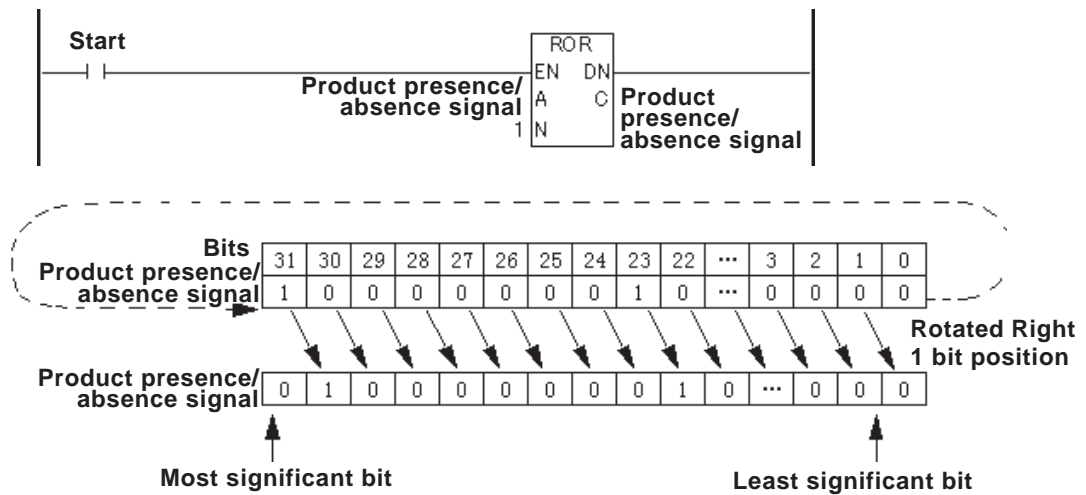


**Note:** #Overflow is turned ON if N is out of range. The result is undefined.

**Reference** See 3.2.17 – “#Overflow.”

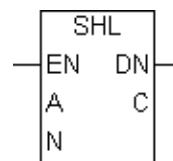
◆ **Example**

The following example describes the operation of 1-bit rotation using the signal of product presence/absence.



**4.2.21 SHL (Shift Left)**

- A: Variable name to be shifted
- N: Number of bit positions to shift
- C: Destination variable



The SHL instruction left-shifts the bits in A by N positions. Bits are dropped from the left end (most significant bit) of the element, and 0 is inserted in the now-vacant bit positions at the right end (least significant bit). The result is placed in C.

The SHL instruction always passes power.

## Chapter 4 – Instructions

The following table lists the types of A, N and C in which SHL instructions can be executed.

A	N	C
Integer	Integer or Integer Constant	Integer
Integer Array	Integer or Integer Constant	Integer Array is same size as A
Integer Constant	Integer or Integer Constant	Integer

There are two types of SHL instruction.

1. If neither A nor C is an array, a simple 32-bit shift is performed. N must range from 0 to 31.
2. If both A and C arrays are the same size, the A array is treated as a large Integer.

Bits are shifted from one element to the next, rather than the most significant bit being dropped from the left end of each element. Only the most significant bit of the highest-numbered element within the array is dropped. N must range from 0 to  $[(32 \times \text{array size}) - 1]$ , inclusive.



**Note:** #Overflow is turned ON if N is out of range. The result is undefined.

**Reference** See 3.2.17 – “#Overflow.”

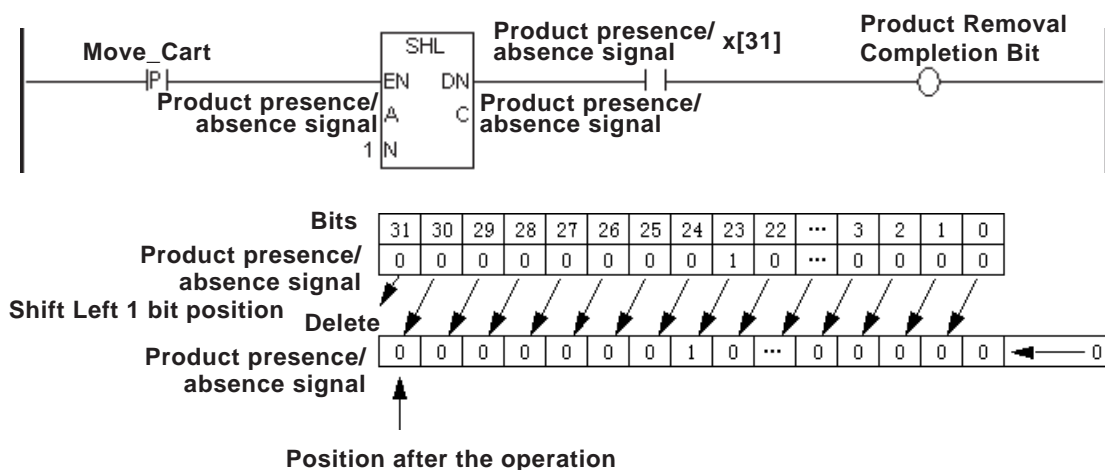
### ◆ Example

The following diagram is an example of a one-bit left shift, used to track the position of a bit.

Each bit in the product presence/absence signal represents the actual position of the product.

When "Move\_Cart" is turned ON, bit is shifted left to the next position.

When the bit reaches the final bit position in the variable (31), the Product Removal Completion Bit is turned ON, indicating that the operation is completed.

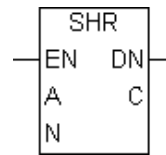


### 4.2.22 SHR (Shift Right)

A: Variable name to be shifted

N: Number of bit positions to shift

C: Destination variable



The SHR instruction right-shifts the bits in A by N positions. Bits are dropped from the right end (least significant bit) of the element, and 0 is inserted in the now-vacant bit positions at the left end (most significant bit). The result is placed in C.

The SHR instruction always passes power. The following table lists the types of A, N, and C in which SHR instructions can be executed.

A	N	C
Integer	Integer or Integer Constant	Integer
Integer Array	Integer or Integer Constant	Integer Array is same size as A
Integer Constant	Integer or Integer Constant	Integer

There are two types of SHR instructions.

1. If neither A nor C is an array, a simple 32-bit shift is performed. N must range from 0 to 31.
2. If both A and C arrays are the same size, the A array is treated as a large Integer.

Bits are shifted from one element to the next, rather than the least significant bit being dropped from the right end of each element. Only the least significant bit of the lowest-numbered element within the array is dropped. N must range from 0 to  $[(32 \times \text{array size}) - 1]$ , inclusive.



**Note:**

#Overflow is turned ON if N is out of range. The result is undefined.

**Reference** See 3.2.17 – “#Overflow.”

#### ◆ Example

##### • When Using Bits

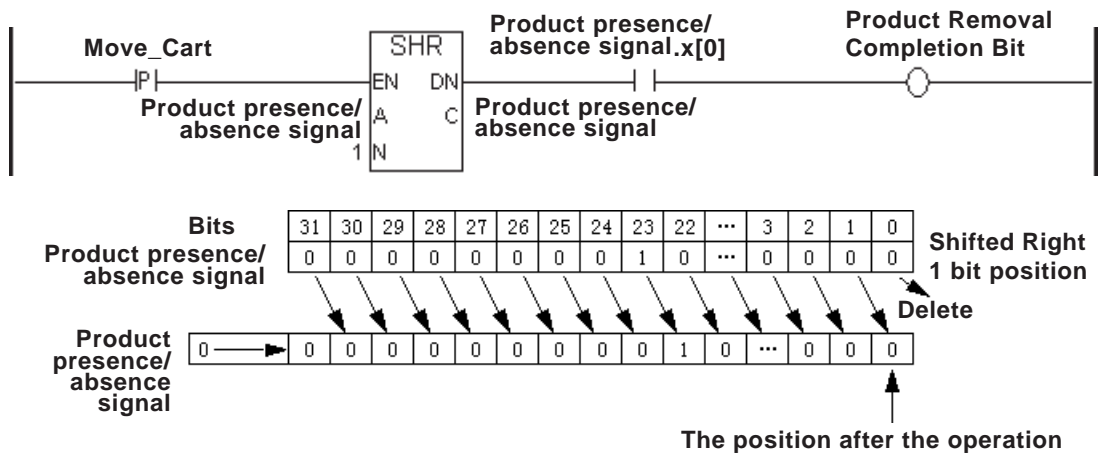
The following diagram is an example of a one-bit right shift, used to track the position of a bit.

Each bit in the product presence/absence signal represents the actual position of the product.

When "Move\_Cart" is turned ON, bit is shifted right to the next position.

When the bit reaches the final bit position in the variable (0), the Product Removal Completion Bit is turned ON, indicating that the operation is completed.

## Chapter 4 – Instructions

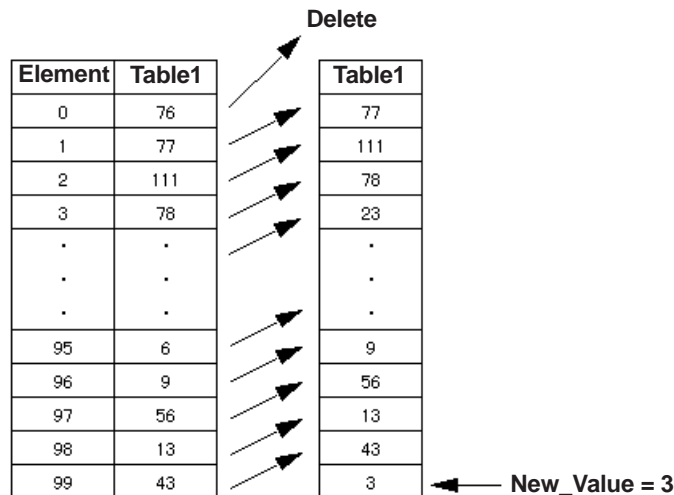
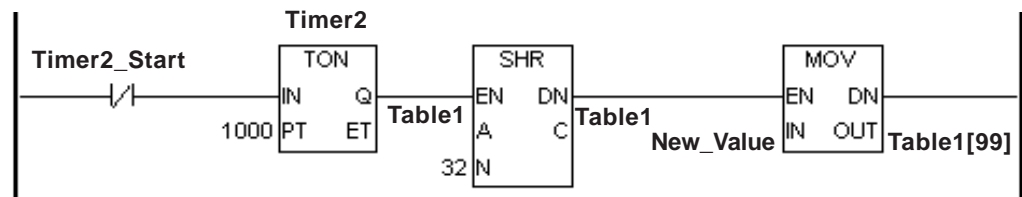


- **When Using Arrays**

The following diagram is an example an SHR instruction being used to transfer values of each element in an Integer array.

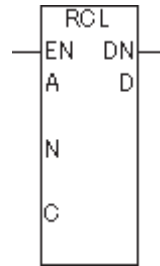
A 32-bit shift rotates the entire 32-bit Integer.

Every second, the “Table1” Integer array’s values are moved up one position towards Element 0, and a new value becomes the last element “Table1[99]” in the “Table1” Integer array.



**4.2.23 RCL (Left Rotation with Carry)**

A: Name of the variable to be rotated  
 N: Shift bit count  
 C: Name of the variable for carry  
 D: Storing destination variable name



If you execute the RCL instruction, the A bit will be shifted to the left direction by N bits.

The far left bit (most significant bit) is stored in a carry, and carry information (1 or 0) is rotated to the far right bit (least significant bit).

This instruction always continues.

The combination of A, N, C and D that you can execute the RCL instruction is as the following.

Type of A	Type of N	Type of C	Type of D
Integer	Integer, or Integer Constant	Bit	Integer
Integer Constant	Integer, or Integer Constant	Bit	Integer



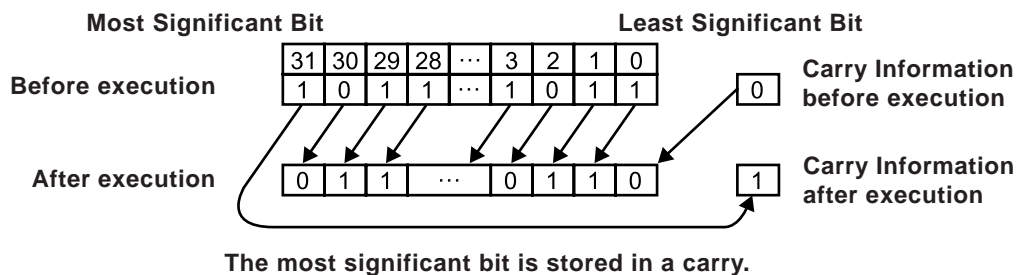
- You should set N to be 0 (zero) or greater and to be 32 or less.
- If N is out of range, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

◆ **Example**

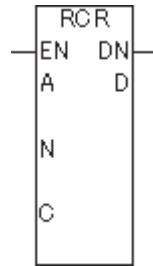
The figure below indicates the operation example of a single bit rotation (when N=1). When the RCL instruction is executed, each bit will be shifted to the left direction by 1 (one) bit.

At this time, bit information stored in a carry before execution is stored in the least significant bit, and the most significant bit is stored in a carry as new carry information.



**4.2.24 RCR (Right Rotation with Carry)**

- A: Name of the variable to be rotated
- N: Shift bit count
- C: Name of the variable for carry
- D: Storing destination variable name



If you execute the RCR instruction, the A bit will be shifted to the right direction by N bits.

The far right bit (most significant bit) is stored in a carry, and carry information (1 or 0) is rotated to the far left bit (least significant bit).

This instruction always continues.

The combination of A, N, C and D that you can execute the RCR instruction is as the following.

Type of A	Type of N	Type of C	Type of D
Integer	Integer, or Integer Constant	Bit	Integer
Integer Constant	Integer, or Integer Constant	Bit	Integer



**Note:**

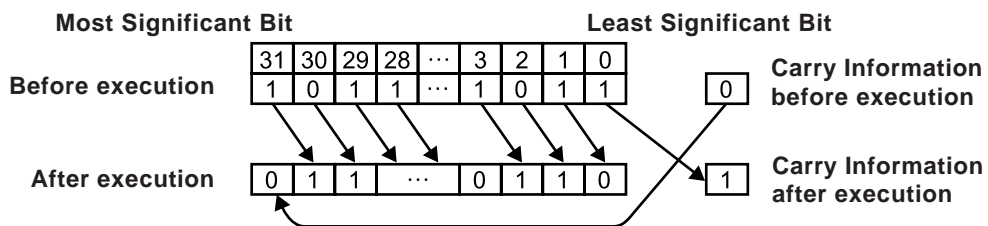
- You should set N to be 0 (zero) or greater and to be 32 or less.
- If N is out of range, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

◆ **Example**

The figure below indicates the operation example of a single bit rotation (when N=1). When the RCR instruction is executed, each bit will be shifted to the right direction by 1 (one) bit.

At this time, bit information stored in a carry before execution is stored in the most significant bit, and the least significant bit is stored in a carry as new carry information.



Carry information is stored in the most significant bit.

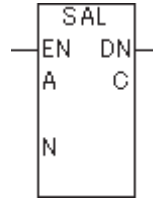


**4.2.25 SAL (Arithmetic Shift Left)**

A: Name of the variable to be shifted

N: Shift bit count

C: Storing destination variable name



If you execute the SAL instruction, the A bit will be shifted to the left direction by N bits.

When it is shifted by 1 (one) bit, the far left bit (most significant bit) remains as it is, the 2nd bit from the far left bit is lost and 0 (zero) is stored in a vacant bit on the far right.

This instruction always continues.

The combination of A, N and C that you can execute the SAL instruction is as the following.

Type of A	Type of N	Type of C
Integer	Integer, or Integer Constant	Integer
Integer Constant	Integer, or Integer Constant	Integer



**Note:**

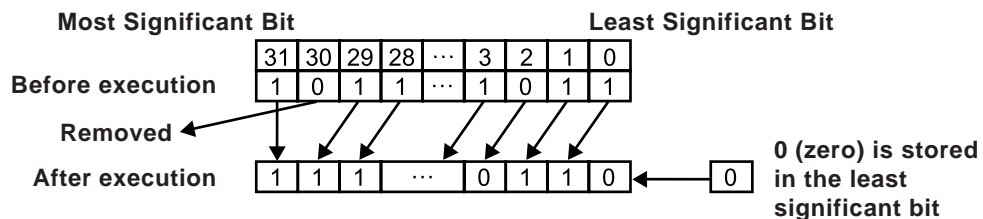
- You should set N to be 0 (zero) or greater and to be 31 or less.
- If N is out of range, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

**◆ Example**

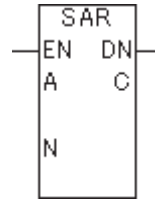
The figure below indicates the operation example of a single bit rotation (when N=1). When the SAL instruction is executed, each bit will be shifted to the left direction by 1 (one) bit.

At this time, only the most significant bit is not shifted, it is stored in the most significant bit as it is after execution and 0 (zero) is stored in the least significant bit.



**4.2.26 SAR (Arithmetic Shift Right)**

A: Name of the variable to be shifted  
 N: Shift bit count  
 C: Storing destination variable name



If you execute the SAR instruction, the A bit will be shifted to the right direction by N bits. When it is shifted by 1 (one) bit, the far right bit (least significant bit) is lost and the most significant bit information before execution is stored in a vacant bit on the far left.

This instruction always continues.

The combination of A, N and C that you can execute the SAR instruction is as the following.

Type of A	Type of N	Type of C
Integer	Integer, or Integer Constant	Integer
Integer Constant	Integer, or Integer Constant	Integer



**Note:**

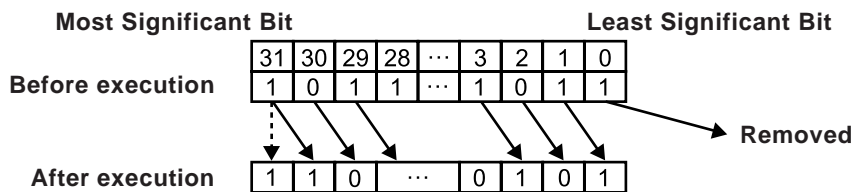
- You should set N to be 0 (zero) or greater and to be 31 or less.
- If N is out of range, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

**◆ Example**

The figure below indicates the operation example of a single bit rotation (when N=1). When the SAR instruction is executed, each bit will be shifted to the right direction by 1 (one) bit.

When it is shifted by 1 (one) bit, the far right bit (least significant bit) is lost and the most significant bit information before execution is stored in a vacant bit on the far left.

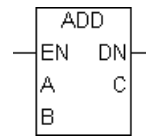


## 4.2.27 ADD (Add)

A: Data

B: Data

C: Destination Variable



When the ADD instruction is executed, A and B are added, and the result is placed in C.

If both A and B are Integers or Integer constants, the ADD instruction performs an Integer addition. Otherwise, the instruction performs a floating-point instruction, which may reduce the processing speed.

The ADD instruction always passes power. The following table lists the combinations of A, B and C in which ADD instructions can be executed.

A	B	C
Integer	Integer	Integer or Real
Integer Constant	Integer Constant	Integer or Real
Real	Real	Integer or Real
Real Constant	Real Constant	Integer or Real

**Note:**

- If the result C exceeds the range expressed with the variable data type in C, #Overflow turns ON and the result of ADD is undefined.

**Reference** See 3.2.17 – “#Overflow.”

- If either A or B are Real, both are converted to Real prior to the addition. However, if C is an Integer, the number is truncated after the decimal point, since the result is placed in C.

## ◆ Example

When Start is turned ON, Data A and Data B are added and the result of the operation is stored in Data C.

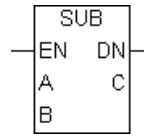


**4.2.28 SUB (Subtract)**

A: Data

B: Data

C: Destination Variable



When the SUB instruction is executed, B is subtracted from A, and the difference is placed in C.

If both A and B are Integers or Integer constants, the SUB instruction performs an Integer subtraction. Otherwise, the instruction performs a floating-point instruction, which may reduce the processing speed.

The SUB instruction always passes power. The following table lists the types of A, B and C in which SUB instructions can be executed.

A	B	C
Integer	Integer	Integer or Real
Integer Constant	Integer Constant	Integer or Real
Real	Real	Integer or Real
Real Constant	Real Constant	Integer or Real

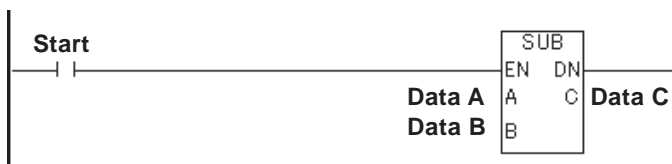


**Note:**

- If the result C exceeds the range expressed with the variable data type in C, #Overflow turns ON and the result of SUB is undefined.  
**Reference** See 3.2.17 – “#Overflow.”
- If either A or B are Real, both are converted to Real prior to the subtraction. However, if C is an Integer, the number is truncated after the decimal point, since the result is placed in C.

◆ **Example**

When Start is turned ON, Data B is subtracted from Data A and the result of the operation is stored in Data C.

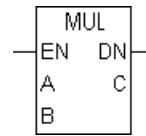


## 4.2.29 MUL (Multiply)

A: Data

B: Data

C: Destination variable



When the MUL instruction is executed, A is multiplied by B, and the result is placed in C. If both A and B are Integers or Integer constants, the MUL instruction performs an Integer multiplication .

Otherwise, the instruction performs a floating-point instruction, which may reduce the processing speed.

The MUL instruction always passes power. The following table lists the combinations of A, B and C in which MUL instructions can be executed.

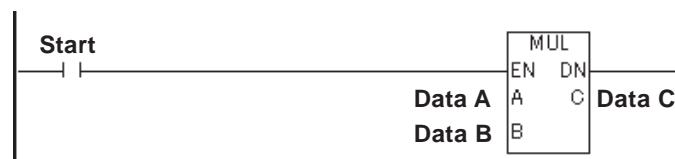
A	B	C
Integer	Integer	Integer or Real
Integer Constant	Integer Constant	Integer or Real
Real	Real	Integer or Real
Real Constant	Real Constant	Integer or Real

**Note:**

- If the result C exceeds the range expressed by the variable data type in C, #Overflow turns ON and the result of MUL is undefined.  
**Reference** See 3.2.17 – “#Overflow.”
- If either A or B are Real, both are converted to Real prior to the multiplication. However, if C is an Integer, the number is truncated after the decimal point, since the result is placed in C.

## ◆ Example

When Start is turned ON, Data A is multiplied by Data B, and then the result of the operation is stored in Data C.

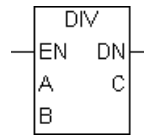


**4.2.30 DIV (Divide)**

A: Data

B: Data

C: Destination variable



When the DIV instruction is executed, A is divided by B, and the quotient is placed in C.

If both A and B are Integers or Integer constants, the DIV instruction performs an Integer division. Otherwise, the instruction performs a floating-point instruction, which may reduce the processing speed.

The DIV instruction always passes power.

The following table lists the combinations of A, B and C in which DIV instructions can be executed.

A	B	C
Integer	Integer	Integer or Real
Integer Constant	Integer Constant	Integer or Real
Real	Real	Integer or Real
Real Constant	Real Constant	Integer or Real



**Note:**

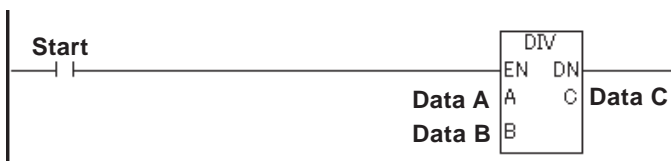
- If B is zero or if the result C exceeds the range expressed by the variable data type in C, #Overflow turns ON and the result of DIV is undefined.

**Reference** See 3.2.17 – “#Overflow.”

- If either A or B are Real, both are converted to Real prior to the division. However, if C is an Integer, the number is truncated after the decimal point, since the result is placed in C.

◆ **Example**

When Start is turned ON, Data A is divided by Data B and the result of the operation is stored in Data C.

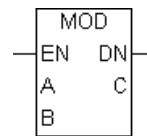


**4.2.31 MOD (Modulus)**

A: Data

B: Data

C: Destination variable



When the MOD instruction is executed, A is divided by B, and the remainder is placed in C. The MOD instruction performs only Integer or Integer Constant operations.

The MOD instruction always passes power.

The following table lists the combinations of A, B and C in which MOD instructions can be executed.

A	B	C
Integer Constant	Integer	Integer
Integer	Integer Constant	Integer



**Note:** #Overflow is turned ON when divided by zero, and the result C is undefined.

**Reference** See 3.2.17 – “#Overflow.”

◆ **Example**

When Start is turned ON, Data A is divided by Data B and the remainder is stored in Data C.

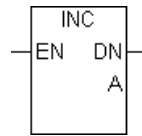


The following example is an Integer (27) divided by 5, and the result (2) is placed in C.

$$\begin{array}{r}
 \text{A}=27 \quad 5 \overline{)27} \\
 \text{B}=5 \quad \underline{25} \\
 \quad \quad 2 \leftarrow \text{C}=2
 \end{array}$$

**4.2.32 INC (Increment)**

A: Data



When the INC instruction is executed, one (1) is added to A, and the result is then placed in A.

The INC instruction always passes power.

The following table lists the combinations of A in which INC instructions can be executed.

<b>A</b>
Integer



**Note:** #Overflow is set if A increments from 0x7FFFFFFF to 0x80000000.

**Reference** See 3.2.17 – “#Overflow.”

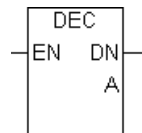
◆ **Example**

When Start is turned ON, "1" is added to Data A.



**4.2.33 DEC (Decrement)**

A: Data



When the DEC instruction is executed, one (1) is subtracted from A, and the result is then placed in A.

The DEC instruction always passes power.

The following table lists the combinations of A in which DEC instructions can be executed.

<b>A</b>
Integer



**Note:** #Overflow is set if A decrements from 0x80000000 to 0x7FFFFFFF.

**Reference** See 3.2.17 – “#Overflow.”

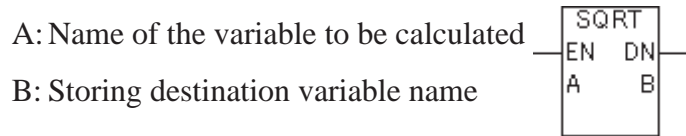


◆ Example

When Start is turned ON, "1" is subtracted from Data A.



**4.2.34 SQRT (Square Root)**



If you execute the SQRT instruction, a square root of an input will be calculated. The result B is a real number between + 2.225e-308 and + 1.79e+308. This instruction always continues.

A and B that you can execute the SQRT instruction are as the following.

Type of A	Type of B
Positive Integer, or Positive Integer Constant	Real
Positive Real, or Positive Real Constant	Real



**Note:** If A is negative and the solution is an imaginary number, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

◆ Example

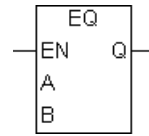
When "Start" is set to ON, the square root of "Data A" will be stored in "Data B".



**4.2.35 EQ (Compare: = )**

A: Data

B: Data



The EQ instruction passes power if A is equal to B.

The following table lists the combinations of A and B in which EQ instructions can be executed.

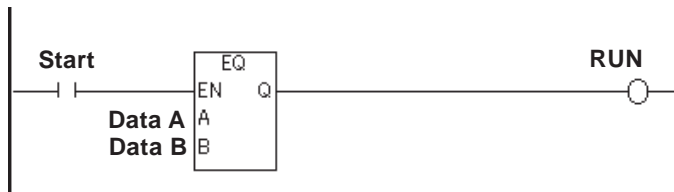
A	B
Integer	Integer
Integer Constant	Integer Constant
Real	Real
Real Constant	Real Constant



**Note:** Real values need to be compared very carefully. For example, a calculation might result in 1.9999999999, which is not equal to 2.0000000000.

◆ **Example**

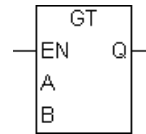
RUN mode is triggered when the values of Data A and Data B are equal after Start is turned ON.



### 4.2.36 GT (Compare: > )

A: Data

B: Data



The GT instruction passes power if A is greater than B.

The following table lists the combinations of A and B in which GT instructions can be executed.

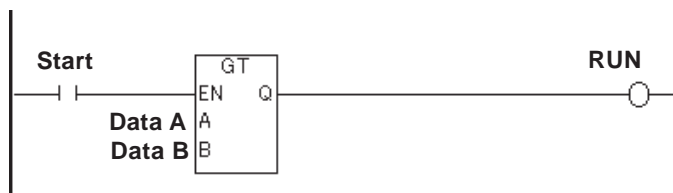
A	B
Integer	Integer
Integer Constant	Integer Constant
Real	Real
Real Constant	Real Constant



**Note:** Real values need to be compared very carefully. For example, a calculation might result in 2.000000000001, which is greater than 2.

#### ◆ Example

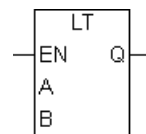
Run mode is triggered when the value of Data A is greater than that of Data B after Start is turned ON.



### 4.2.37 LT (Compare: < )

A: Data

B: Data



The LT instruction passes power if A is less than B.

The following table lists the combinations of A and B in which LT instructions can be executed.

A	B
Integer	Integer
Integer Constant	Integer Constant
Real	Real
Real Constant	Real Constant

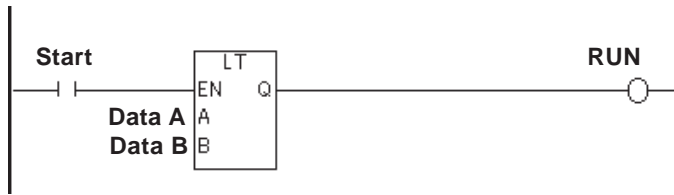


**Note:** Real values need to be compared very carefully. For example, a calculation might result in 1.999999999999, which is less than 2.

## Chapter 4 – Instructions

### ◆ Example

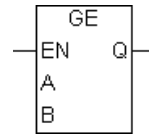
Run mode is triggered when the value of Data A is smaller than that of Data B after Start is turned ON.



### 4.2.38 GE (Compare: >= )

A: Data

B: Data



The GE instruction passes power if A is greater than or equal to B.

The following table lists the combinations of A and B in which GE instructions can be executed.

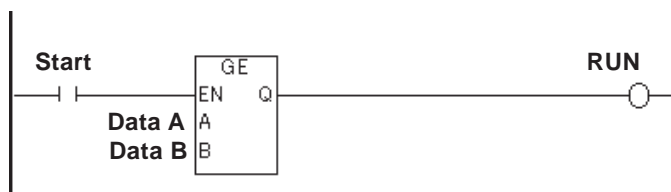
A	B
Integer	Integer
Integer Constant	Integer Constant
Real	Real
Real Constant	Real Constant



**Note:** Real values need to be compared very carefully. For example, a calculation might result in 1.9999999999, which is not greater than or equal to 2.

### ◆ Example

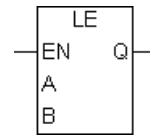
Run mode is triggered when the value of Data A is equal to or greater than that of Data B after Start is turned ON.



### 4.2.39 LE (Compare: <= )

A: Data

B: Data



The LE instruction passes power if A is less than or equal to B.

The following table lists the combinations of A and B in which LE instructions can be executed.

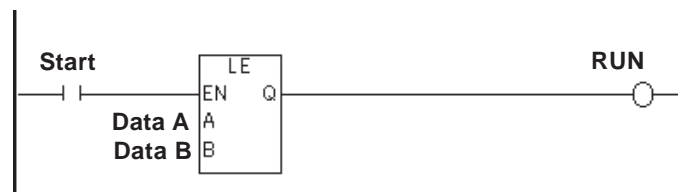
A	B
Integer	Integer
Integer Constant	Integer Constant
Real	Real
Real Constant	Real Constant

**Note:**

Real values need to be compared very carefully. For example, a calculation might result in 2.000000000001, which is not less than or equal to 2.

#### ◆ Example

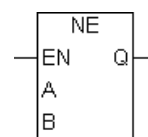
Run mode is triggered when the value of Data A is equal to or smaller than that of Data B after Start is turned ON.



### 4.2.40 NE (Compare: <> )

A: Data

B: Data



The NE instruction passes power if A is not equal to B.

The following table lists the combinations of A and B in which NE instructions can be executed.

A	B
Integer	Integer
Integer Constant	Integer Constant
Real	Real
Real Constant	Real Constant

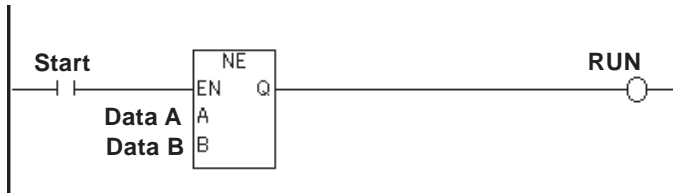
**Note:**

Real values need to be compared very carefully. For example, a calculation might result in 1.999999999999, which is not equal to 2.

## Chapter 4 – Instructions

### ◆ Example

After Start is turned ON, Run mode is triggered when the values of Data A and Data B are not equal.



### 4.2.41 TON (Timer ON Delay)

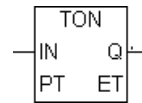
IN: Timer starting bit

PT: Preset time of timer

Q: Time up flag

ET: Elapsed time

**Variable**



After the timer input bit (IN) receives power, the TON instruction turns the timer output bit (Q) ON when the elapsed time (ET) equals the preset time (PT) in milliseconds.

### ◆ Overview

Special Variable	Description	Variable Type
Variable. PT	Preset Value	Integer
Variable. ET	Elapsed Time Value	Integer
Variable. Q	Timer Output Bit	Discrete
Variable. TI	Timing Bit	Discrete

When power is passed to the timer starting bit (IN), the TON instruction starts, and:

- Variable.ET (the elapsed time) begins to increment in milliseconds.
- Variable.TI (the timing bit) turns ON.
- Variable.Q (the timer output bit) turns OFF.

When the elapsed time (Variable.ET) increments and equals the preset time (Variable.PT):

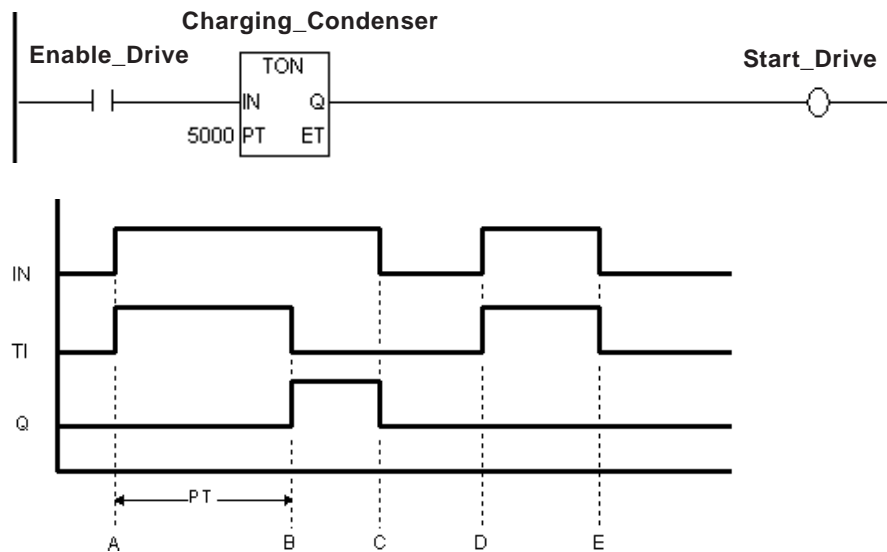
- Variable.ET (the elapsed time) holds the current value.
- Variable.TI (the timing bit) turns OFF.
- Variable.Q (the timer output bit) turns ON, and the instruction passes power.

When the timer starting bit (IN) stops passing power to start the TON instruction:

- Variable.ET (the elapsed time) is reset to zero.
- Variable.TI (the timing bit) turns OFF.
- Variable.Q (the timer output bit) turns OFF.

### ◆ Example

In the following example, the drive will be started 5 seconds after "Enable\_Drive" is turned ON.



- A: When power is applied to the timer input bit (IN), the timing bit (TI) turns ON, the timer begins timing, and the elapsed time (ET) increments. The timer output bit (Q) remains OFF.
- B: The elapsed time (ET) equals the preset time (PT), the timer output bit (Q) turns ON, and the elapsed time (ET) stays fixed at the preset time. The timing bit (TI) turns OFF.
- C: The timer input bit (IN) turns OFF, the timer output bit (Q) turns OFF, and the elapsed time (ET) is reset to 0.
- D: The timer input bit (IN) turns ON, and the timing bit (TI) turns ON. The timer begins timing, and the elapsed time (ET) increments.
- E: The timer input bit (IN) is turned OFF before the elapsed time (ET) equals preset time (PT), the timer output bit (Q) remains OFF, the elapsed time (ET) is reset to 0.

**4.2.42 TOF (Timer OFF Delay)**

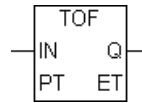
IN: Timer starting bit

PT: Preset time of timer

Q: Time up flag

ET: Elapsed time

**Variable**



After the timer input bit (IN) stops receiving power, the TOF instruction turns the timer output bit (Q) OFF when the elapsed time (ET) equals the preset time (PT) in milliseconds.

◆ **Overview**

Special Variable	Description	Variable Type
Variable. PT	Preset Value	Integer
Variable. ET	Elapsed Time Value	Integer
Variable. Q	Timer output bit	Discrete
Variable. TI	Timing bit	Discrete

When power is passed to the timer starting bit (IN), the TOF instruction starts, and:

- Variable.ET (the elapsed time) is reset to zero.
- Variable.TI (the timing bit) turns OFF.
- Variable.Q (the timer output bit) turns ON, and the instruction passes power.

When the timer starting bit (IN) stops passing power to start the TOF instruction:

- Variable.ET (the elapsed time) begins to increment, in milliseconds.
- Variable.TI (the timing bit) turns ON.
- Variable.Q (the timer output bit) remains ON.

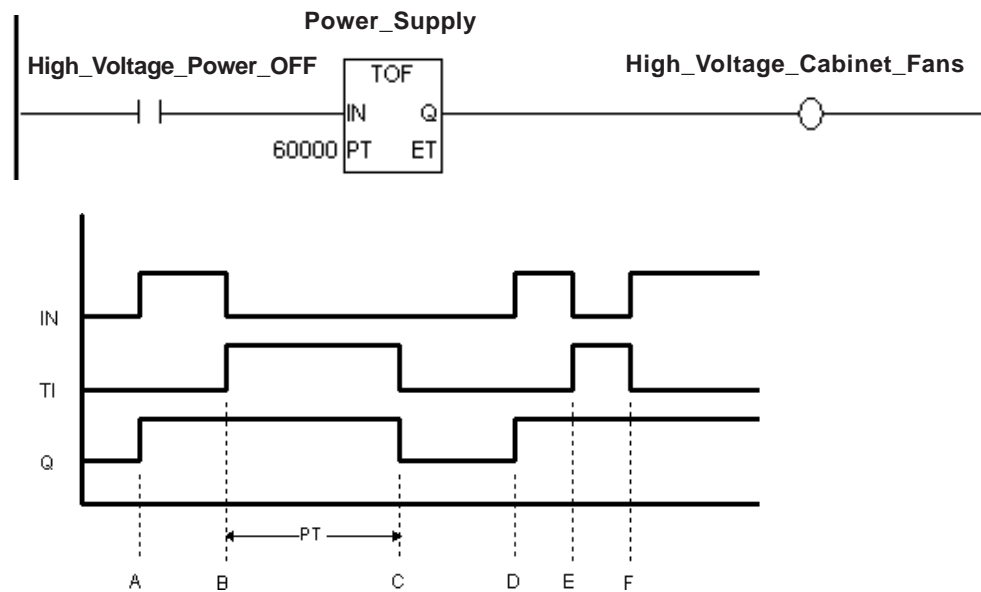
When the elapsed time (Variable.ET) increments and equals the preset time (Variable.PT):

- Variable.ET (the elapsed time) stays fixed at the preset value.
- Variable.TI (the timing bit) turns OFF.
- Variable.Q (the timer output bit) turns OFF.



## ◆ Example

The following diagram is an example of high-voltage cabinet fans that are kept running for 1 minute (60,000ms) after the high voltage turns OFF.



A: The timer input bit (IN) turns ON, the timing bit (TI) remains OFF, the timer output bit (Q) turns ON, and the elapsed time (ET) is reset to 0.

B: The timer input bit (IN) turns OFF, the timer starts timing (TI turns ON), and the timer output bit (Q) remains ON.

C: When the elapsed time (ET) equals the preset time (PT), the timer output bit (Q) turns OFF, the timer stops timing (TI turns OFF), and the elapsed time stays fixed at preset time (ET=PT).

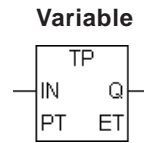
D: The timer input bit (IN) turns ON, the timing bit (TI) remains OFF, the timer output bit (Q) turns ON, and the elapsed time (ET) is reset to 0.

E: The timer input bit (IN) turns OFF, the timer starts timing (TI turns ON), and the timer output bit (Q) remains ON.

F: Before the elapsed time (ET) equals the preset time (PT), the timer input bit (IN) turns ON, and the timer stops timing (TI turns OFF). The timer output bit (Q) remains ON, and the elapsed time (ET) is reset to 0.

**4.2.43 TP (Timer Pulse)**

IN: Timer starting bit  
 PT: Preset time of timer  
 Q: Time up flag  
 ET: Elapsed time



When the timer input bit (IN) receives power one time, the TP instruction turns the output bit (Q) ON for the duration of the preset time (PT), in milliseconds.

◆ **Overview**

Special Variable	Description	Variable Type
Variable. PT	Preset Value	Integer
Variable. ET	Elapsed Time Value	Integer
Variable. Q	Timer output bit	Discrete
Variable. TI	Timing bit	Discrete

When power is passed to the timer starting bit (IN), the TP instruction starts, and:

- Variable.ET (the elapsed time) begins to increment in milliseconds.
- Variable.TI (the timing bit) turns ON.
- Variable.Q (the timer output bit) turns ON as the instruction passes power.

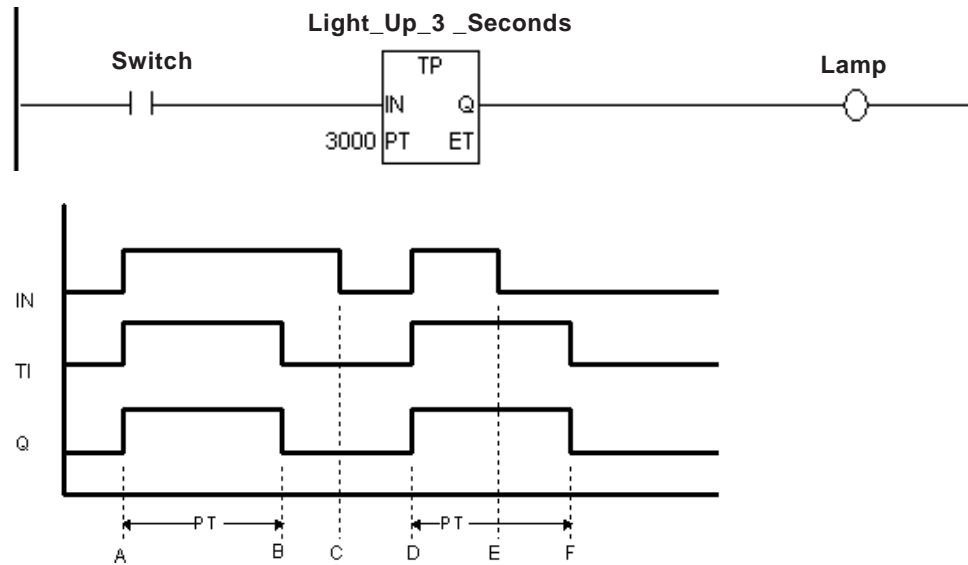
When the elapsed time (Variable.ET) equals the preset time (Variable.PT):

- Variable.ET (the elapsed time) stays fixed at the preset value if the TP instruction is still receiving power.
- Variable.ET (the elapsed time) resets immediately to zero if the instruction stops receiving power.
- Variable.TI (the timing bit) turns OFF.
- Variable.Q (the timer output bit) turns OFF and stops receiving power.

When the timer starting bit (IN) stops passing power to start the TP instruction, the elapsed time (Variable.ET) is reset to zero, and the timer output bit (Variable.Q) turns OFF — only if it has already reached the value of the preset time (Variable.PT). Otherwise, it continues timing, and the timer output bit (Variable.Q) remains ON.

◆ **Example**

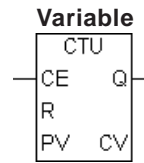
The following diagram is an example of a lamp that lights up for three seconds when the switch is pressed.



- A: The timer input bit (IN) turns ON, the timer starts timing (TI turns ON), and the timer output bit (Q) turns ON.
- B: When the elapsed time (ET) equals the preset time (PT), the timer output bit (Q) turns OFF, the timer stops timing (TI turns OFF), and the elapsed time stays fixed at the preset time (ET=PT).
- C: The timer input bit (IN) turns OFF, and the elapsed time (ET) is reset to 0.
- D: The timer input bit (IN) turns ON, the timer starts timing (TI turns ON), and the timer output bit (Q) turns ON.
- E: The timer input bit (IN) turns OFF, the timer continues timing (TI remains ON), and the timer output bit (Q) remains ON.
- F: When the elapsed time (ET) equals the preset time (PT), the timer output bit (Q) turns OFF, the timer stops timing (TI turns OFF), and since the timer input bit (IN) is OFF, the elapsed time (ET) is reset to 0.

**4.2.44 CTU (UP Counter)**

- CE: Counter starting bit
- R: Counter reset bit
- PV: Preset value of counter
- Q: Counter output
- CV: Current value of counter



◆ **Overview**

Special Variable	Description	Variable Type
Variable. PV	Preset Value	Integer
Variable. CV	Current Value	Integer
Variable. R	Counter Reset	Discrete
Variable. UP	UP Counter	Discrete
Variable. QU	UP Counter Output	Discrete
Variable. QD	Down Counter Output	Discrete
Variable. Q	Counter Output	Discrete

When the counter input bit (CE) passes power, the current value (Variable .CV) is incremented by one if the counter reset bit (Variable.R) is OFF and the current value (Variable .CV) is smaller than Preset value (Variable PV).

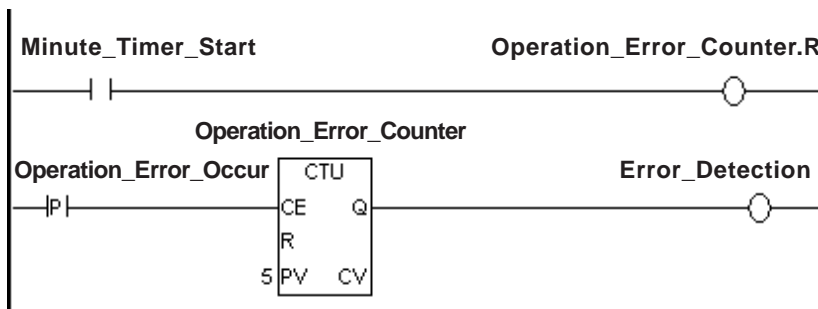
When the current value (Variable.CV) is equal to the preset value (Variable.PV), the counter output bit (Variable.Q) is turned ON, and the instruction passes power.

When the counter reset bit (Variable.R) is ON, the current value (Variable.CV) is reset to zero.

The counter output bit (Variable.Q) is also turned OFF.

◆ **Example**

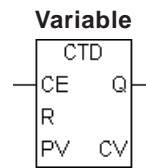
The following diagram is an example of the CTU instruction notifying the Error\_Detection output when five errors have been counted during a one-minute period.



**Note:** The counter is reset every scan. To count an event like the example above, be sure that the PT instruction is positioned before the CTU instruction's position. The CTU instruction is a level input.

**4.2.45 CTD (DOWN Counter)**

- CE: Counter starting bit
- R: Counter reset bit
- PV: Preset value of counter
- Q: Counter output
- CV: Current value of counter



◆ **Overview**

Special Variable	Description	Variable Type
Variable.PV	Preset Value	Integer
Variable.CV	Current Value	Integer
Variable.R	Counter Reset	Discrete
Variable.UP	UP Counter	Discrete
Variable.QU	UP Counter Output	Discrete
Variable.QD	Down Counter Output	Discrete
Variable.Q	Counter Output	Discrete

When the counter input bit (CE) passes power, the current value (Variable.CV) is decremented by one if the counter reset bit (Variable.R) is OFF.

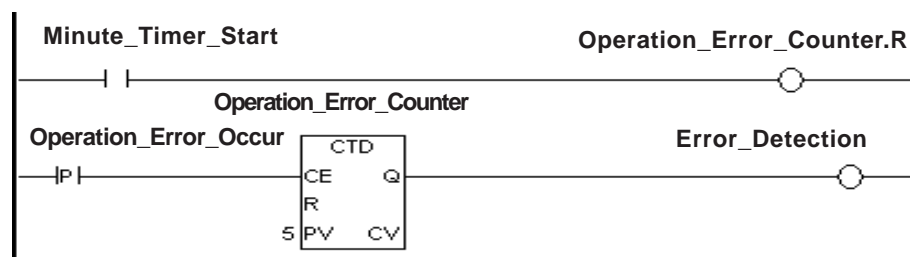
When the current value (Variable.CV) becomes equal to or less than zero after decrementing, the counter output bit (Variable.Q) is turned ON, and the instruction passes power.

When the counter reset bit (Variable.R) is ON, the preset value (Variable.PV) is set to the current value (Variable.CV).

The counter output bit (Variable.Q) is also turned OFF.

◆ **Example**

The following diagram is an example of the CTD instruction passing power and notifying the Error\_Detection output when five errors have been counted during a one-minute period. The timer resets the counter every minute.

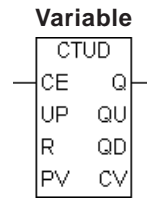


**Note:**

The counter is reset every scan. To count an event like the example above, be sure that the PT instruction is positioned before the CTD instruction's position. The CTD instruction is a level input.

**4.2.46 CTUD (UP/DOWN Counter)**

- CE: Counter starting bit
- UP: Counter Up Instruction
- R: Counter reset bit
- PV: Preset value of counter
- Q: Counter output
- QU: UP Counter flag
- QD: Down Counter flag
- CV: Current value of counter



◆ **Overview**

Special Variable	Description	Variable Type
Variable.PV	Preset Value	Integer
Variable.CV	Current Value	Integer
Variable.R	Counter Reset	Discrete
Variable.UP	UP Counter	Discrete
Variable.QU	UP Counter Output	Discrete
Variable.QD	Down Counter Output	Discrete
Variable.Q	Counter Output	Discrete

When executing the CTUD instruction while the counter up instruction Variable.UP is ON, the execution is similar with the CTU instruction (up-counter).

When Variable.UP is OFF, the execution is similar with the CTD (down-counter) instruction.

After executing the CTUD instruction:

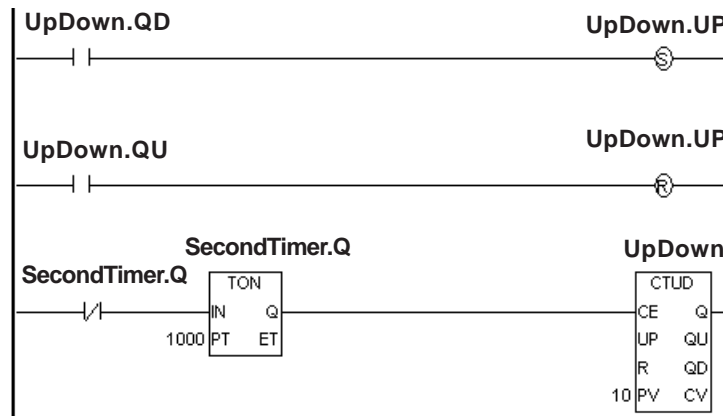
- If the current value (Variable.CV) is equal to or greater than the preset value (Variable.PV), the Counter Output and UP Counter Output (Variable.Q and Variable.QU) are turned ON.
- If the current value (Variable.CV) is equal to or less than zero, the Counter Output and Down Counter Output (Variable.Q and Variable.QD) are turned ON.

◆ **Example**

The following diagram is an example of the CTUD instruction continuously counting up, from 0 to 10, and then down from 10 to 0.

The SecondTimer outputs a pulse to the Up/Down Counter every second.

The UP bit turns ON when the Up/Down Counter reaches 0, and turns OFF when the Up/Down counter reaches 10 (the preset value).

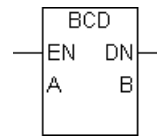


**Note:** If the counter reset bit (Variable.R) turns ON when the Counter Up instruction (Variable.UP) is ON, the current value (Variable.CV) is set to zero. If the counter reset bit (Variable.R) turns ON when the Counter Up instruction (Variable.UP) is OFF, the preset value (Variable.PV) is set to the current value (Variable.CV).

#### 4.2.47 BCD (BCD Conversion)

A: Data

B: Result to be stored



When the BCD instruction is executed, a binary number assigned to A is converted to binary-coded decimal format, and the result is placed in B.

The BCD instruction does not pass power if an error occurs. The following table lists the combinations of A and B in which BCD instructions can be executed.

A	B
Integer	Integer
Integer Constant	

The largest value of A that can be converted is 0 x 5F5E0FF. If A is too large, #FaultCode is updated with the error code, and #Overflow is turned ON.

**Reference** See 3.2.14 – “#FaultCode” and 3.2.17 – “#Overflow.”



**Note:** If the value cannot be converted, the value in B is undefined.

#### ◆ Example

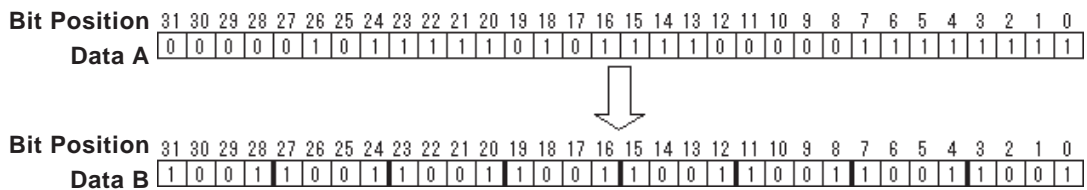
When Start is turned ON, Data A is converted to BCD and stored in Data B.



## Chapter 4 – Instructions

### ◆ Example

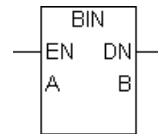
BIN data "99999999" is designated for Data A, and BCD conversion is performed.



### 4.2.48 BIN (Binary Conversion)

A: Data

B: Result to be stored



When the BIN instruction is executed, a binary coded decimal number assigned to A is converted to binary format, and the result is placed in B.

The BIN instruction does not pass power if an error occurs. The following table lists the combinations of A and B in which BIN instructions can be executed.

A	B
Integer	Integer
Integer Constant	

If A is not a valid BCD number, #FaultCode will be updated with the error code, and #Overflow will turn ON.

**Reference** See 3.2.14 “#FaultCode” and 3.2.17 – “#Overflow.”



**Note:**

If the value cannot be converted, the value in B is undefined.

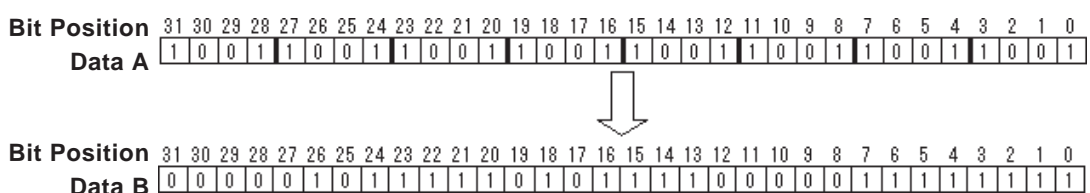
### ◆ Example

When Start is turned ON, Data A is converted to BIN and stored in Data B.



### ◆ Example

BIN data "99999999" is designated for Data A, and BCD conversion is performed.

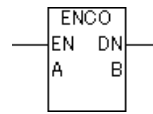




**4.2.49 ENCO (Encode)**

A: Data

B: Result to be stored



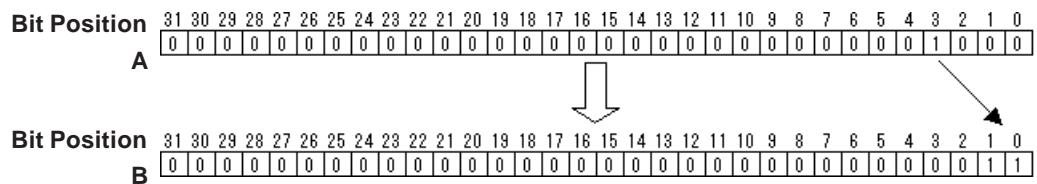
The value entered in A is encoded and output to B. The ENCO instruction reads the 32 bits in A for the bit position that is ON, and this position is output to B as a binary value. If several bits in A are ON, the most significant bit position is output to B.

The ENCO instruction always passes power.

The combinations of valid variable data types for the ENCO instruction are as follows:

A	B
Integer	Integer
Integer Array	Integer Array (same size as A)
Integer Constant	Integer

E.g.: If 0x00000008 is entered in A, the output B is 0x00000003.



- If 0 is entered in Input A, the error code “13” is set to #FaultCode as a minor error (#Overflow).

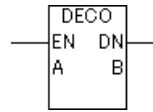
**Reference** See 3.2.17 – “#Overflow.”

- The ENCO instruction does not support variable modifiers (assigned bit, word, or byte).

**4.2.50 DECO (Decode)**

A: Data

B: Result to be stored



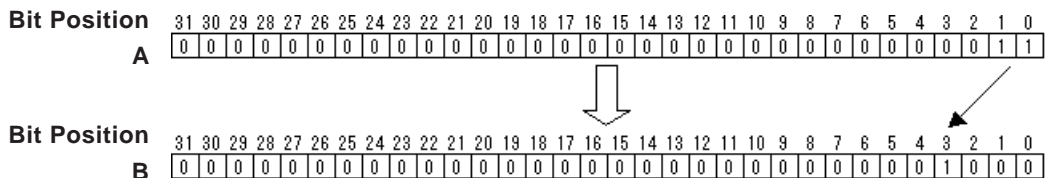
The value entered in A is decoded and output to B. The DECO instruction reads A as a binary value, and the corresponding bit position in B is up. 0 to 31 are available for input.

The DECO instruction always passes power.

The combinations of valid variable data types for the DECO instruction are as follows:

A	B
Integer	Integer
Integer Array	Integer Array (same size as A)
Integer Constant	Integer

E.g.: If 0x00000003 is entered in A, the output B is 0x00000008.



**Note:**

- If a value other than 0 to 31 is entered in Input A, the error code “13” is set to #FaultCode as a minor error (#Overflow).

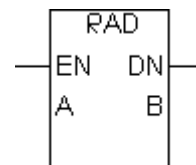
**Reference** See 3.2.17 – “#Overflow.”

- The DECO instruction does not support variable modifiers (assigned bit, word, or byte).

**4.2.51 RAD (Radian conversion)**

A: Data (In degrees)

B: Variable that stores the result (In radian units)



The RAD instruction converts a degree value to a radian value and stores the result in B.

This instruction is normally ON. The following table lists the types of A and B data that can be used for this instruction.

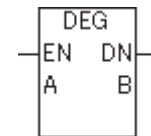
If A is...	B must be...
Integer or integer constant	Real
Real or real constant	Real



**Note:**  $\pi = 3.1415926535897$ .

### 4.2.52 DEG (Degree Conversion)

- A: Data (Radian)
- B: Result storing destination (Degree)



If you execute the DEG instruction, "Radian" of an angle unit will be converted to "Degree" and the result will be stored in B.

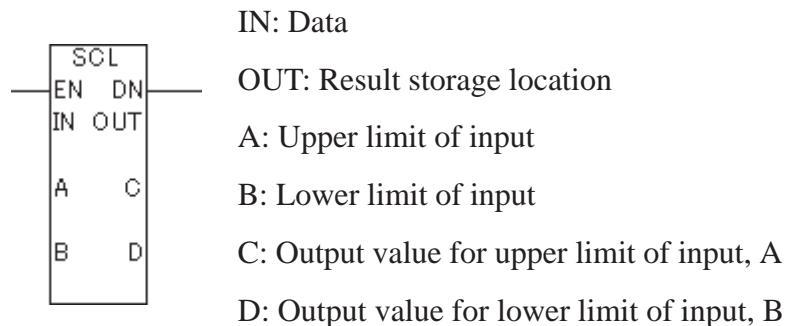
This instruction always continues. The combination of A and B that you can execute the DEG instruction is as the following.

Type of A	Type of B
Integer, or Integer Constant	Real
Real, or Real Constant	Real



**Note:**  $\pi = 3.1415926535897$ .

### 4.2.53 SCL (Scale conversion)



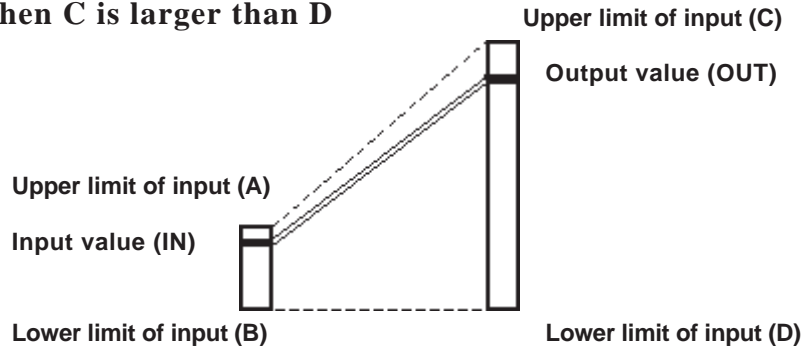
The integral value entered for IN is converted according to the ratio of the range between the upper and lower limits of input (between A and B) to the range between the upper and lower limits of output (between C and D), as shown in the figure below. The result is then stored in OUT.

This instruction is always energized. The combinations of A, B, C, and D that allow the SCL instruction to be executed are shown below.

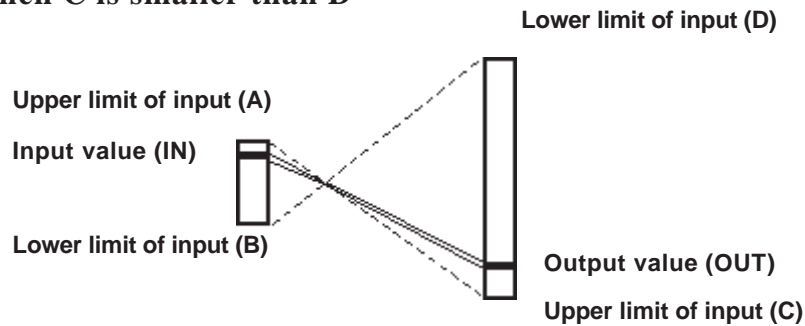
Type of IN	Type of OUT	Type of A
Integer	Integer	Integer constant
Integer constant	Integer	Integer constant

Type of B	Type of C	Type of D
Integer constant	Integer constant	Integer constant
Integer constant	Integer constant	Integer constant

◆ When C is larger than D

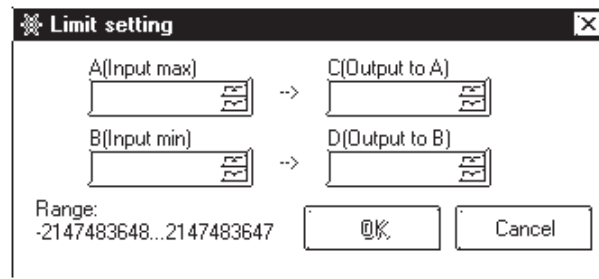


◆ When C is smaller than D



- When double-clicking on the SCL Conversion Instruction, the [RANGE SETTING] dialog box shown below is displayed.

This dialog box can also be used to set parameters A, B, C, and D.



◆ Operation example

To convert the analog input value (0 to 4095) into a "current value" between 4 and 20 [mA] and calculate down to three decimal places, set SCL instruction's parameters A, B, C, and D as described below.

**A (upper limit of input): 4095**  
**B (lower limit of input): 0**

**C (output value for A): 20000**  
**D (output value for B): 4000**

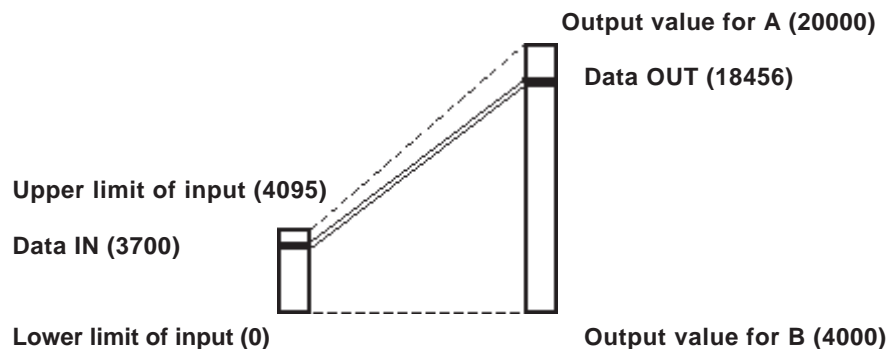
		SCL		
		EN	DN	
Data_IN		IN	OUT	Data_OUT
3700				18456
4095	A	C		20000
0	B	D		4000

When "3700" is entered for data IN, "18456" is stored in data OUT.

Set the display of the decimal point using the GP-PRO/PB III so that the value is displayed as "18.456".

**Reference** For details, see GP-PRO/PB III Operation Manual 2.1.21 – “Numeric Display”.

$$\begin{aligned} \text{Data OUT} &= \text{Data IN} \times (C - D)/(A - B) + (A \times D - B \times C)/(A - B) \\ &= 3700 \times (20000 - 4000)/(4095 - 0) + (4095 \times 4000 - 0 \times 20000)/(4095 - 0) \\ &\doteq 18456.7 \end{aligned}$$



#### 4.2.54 JMP (Jump)

—>> LabelName

When the JMP instruction receives power, control jumps to the specified label. Unlike the JSR instruction, control does not automatically return to the rung following the JMP rung.

A jump cannot be made over a START, SUB START or SUB END label.

Jumping upward can create an infinite loop.



**Note:** Be sure that the time required to execute the entire program will not exceed the value of the Watch Dog Timer.

**Reference** See 3.2.6 – “#WatchDogTime.”

#### ◆ Example

If the Jump Instruction is ON, rung 3’s instruction will be skipped and will not be executed. Control will jump to rung 4 with the label "Operation Disabled", and instructions below rung 4 will be executed.



### 4.2.55 JSR (Jump Subroutine)

—>> SubroutineName<<

When the JSR instruction receives power, the control jumps to the specified subroutine. After the subroutine executes, control returns to the rung that follows the JSR instruction and continues to execute that rung’s instruction. A subroutine name can not be duplicated.

JSR must be the last instruction on a rung.

#### ◆ Restrictions

- A maximum of 128 subroutine jumps from a subroutine can be executed.
- One (1) stack is used per jump in the JSR instruction. The total number of stacks that can be used in a logic program is 128. The only other instruction that uses stacks is the FOR/NEXT instruction, which uses two (2) stacks.

**Reference** See 4.2.57 – “FOR/NEXT (Repeat).”



**Note:**

Be sure that the time required to execute the entire program will not exceed the value of the Watch Dog Timer.

**Reference** See 3.2.6 – “#WatchdogTime”

#### ◆ Control System Error Handler Subroutine

Be sure to create a subroutine called ErrorHandler. Control should jump to this subroutine when calculation errors or other minor faults occur.

**Reference** See 3.2.17 – “#Overflow”

### 4.2.56 RET (Return Subroutine)

—<RETURN>—

When the RET instruction receives power, control is forced to return from a subroutine to its original location. Execution continues from the rung that follows the Jump Subroutine (JSR) instruction.

When a subroutine is completed, the SUB END instruction forces the program to automatically return to the jump point. As a result, the RET instruction is not always needed to perform this function.

The RET instruction must be the last instruction on a rung.

### 4.2.57 FOR/NEXT (Repeat)

—

FOR
EN DN
A

—

—

NEXT
------

—

The FOR/NEXT instruction repeats the logic program between corresponding FOR and NEXT instructions, for the number of times specified in A. After executing the logic program between FOR and NEXT the specified number of times (A), the step that follows the NEXT instruction will be processed.

If A is equal to or less than 0, the logic program between FOR and NEXT is not executed, but jumps to the step that follows the NEXT instruction.

The FOR/NEXT instruction always passes power.

Valid variable data types for the FOR/NEXT instruction are as follows:

<b>A</b>
Integer
Integer Constant

#### ◆ Restrictions

- Each FOR instruction requires a NEXT instruction.
- Do not insert instructions before or after FOR and NEXT instructions on the same rung.
- Up to 64 nests can be included in each instruction.
- The number of executions in a FOR/NEXT instruction cannot be changed.
- A FOR/NEXT instruction cannot be removed while it is being executed.
- If the instruction exceeds more than 64 nests, a major error occurs and error code “4” is displayed in #FaultCode.

## Chapter 4 – Instructions

- Two (2) stacks are used for one nesting. The total number of stacks that can be used in a logic program is 128. The only other instruction that uses stacks is the JSR instruction, which uses one (1) stack.

**Reference** See 4.2.55 – “JSR (Jump Subroutine).”



**Note:**

- For information about the errors or warnings displayed by the Editor’s error check, refer to the Pro-Control Editor Operation Manual, Chapter 7, Appendix 1 – “Errors and Warnings.”

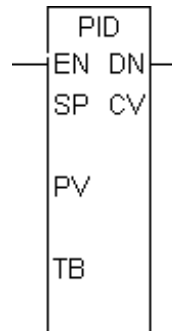
**Reference** For information about #FaultCode error codes, see 3.2.14 – “#FaultCode.”

- When specifying the number of nests, the time required for the program’s entire execution must NOT exceed the value of Watchdog Timer.

**Reference** See 3.2.6 – “#WatchdogTime.”

### 4.2.58 PID (PID Calculation)

SP: Setpoint  
 PV: Process Variable  
 TB: Tieback  
 CV: Control Variable



The PID (Proportional Integral Derivative) instruction compares a measured value (Process Variable), from the analog input or temperature input, with a preset value (Setpoint). The PID then adjusts the Control Variable to eliminate the difference between the Process Variable and the Setpoint.

When performing the PID control, the proportional (P), the integral (I), and the derivative (D) controls can be combined freely. By setting each parameter (described later in this section), these controls can be executed.

The control value calculated by the PID control can be expressed in the following equation.

$$CV = KC(E + \text{Reset} \int_0^t (E)dt + \text{Rate} \frac{d(E)}{dt} )$$

- KC : Proportional Coefficient\*1
- E : Error Signal (SP-PV or PV-SP)
- Reset: Integral Time\*1
- Rate : Derivative Time\*1

\*1 This is set in the “Tuning” tab, explained on the following page. This is not a control block variable value.



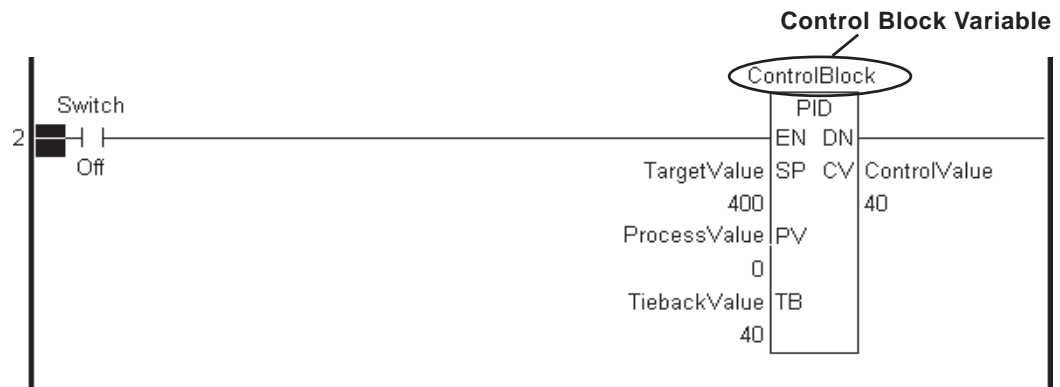
By adjusting the sampling period in the Tune tab which is described later, the effect of noise on the error signal can be reduced. The filtered error signal can be expressed in the following equation.

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

- EF : Filtered Error Signal
- Tloop : Data Refresh Period (Loop update time)
- TFilter : Sampling Time (DFTC)
- E : Error Signal (SP-PV or PV-SP)

### Overview

When the PID instruction passes power, it adjusts the PID output (Automatic Mode). If the PID instruction is not passing power, a constant control amount is output (Manual mode). In Manual mode, the Control Variable is set to the Tieback value.



When using the PID instruction in a logic program, map variables to the control block, SP, PV, TB, and CV variables.

### Parameter and Variable Type

Parameter	Description	Variable Type
SP	Setpoint	Integer, Integer Constant, Integer Array
PV	Process variable	Integer, Integer Array
TB	Tieback When the instruction doesn't receive power, value set in this is output.	Integer, Integer Constant, Integer Array
CV	Controlled variable	Integer, Integer Array

## Chapter 4 – Instructions

### ◆ Control Block Variable

When a variable is mapped to the PID instruction, an array with seven elements (see following table) is mapped to the variable. Element [0] represents the current status, and Elements [1] to [6] are used for the PID control to make fine-tuning adjustments.

Element Number	Description	
0	Bit 0	Mode Switch Flag
	Bit 1	PID instruction process completion flag
	Bit 2	PID deadband flag
	Bit 3	Control variable exceeds upper limit
	Bit 4	Control variable exceeds lower limit
	Bit 5	Exceeded the number of integration process times
1		Proportional coefficient
2		Integration times per minute
3		Derivative time per cycle
4		PID deadband
5		Offset
6		Sampling time



The variable type of a control block variable will be retentive.



*The values in the control block variable for the proportional coefficient, the number of integral times per minute, and the derivative time per cycle are 1000 times the values of proportional coefficient, integral times, and derivative time set in the Tune tab.*

### ■ Control Block Variable Element [0] Status

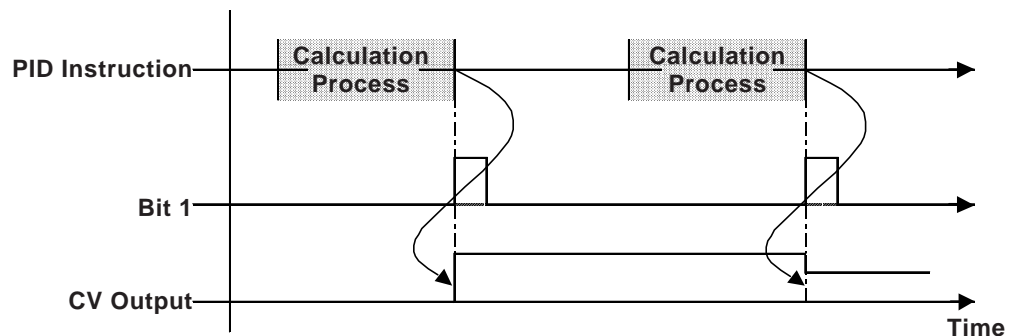
#### ◆ Mode Switch Flag (Bit 0)

When the PID instruction in a logic program passes power, bit 0 turns ON.

Bit 0	Mode
ON	Automatic Mode (PID Calculation)
OFF	Manual Mode

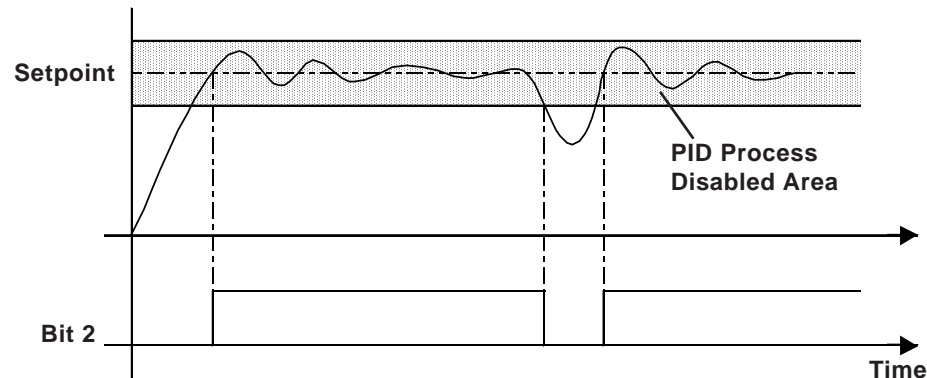
#### ◆ PID Instruction Process Completion Flag (Bit 1)

When the calculation process is finished and the CV is output, bit 1 turns ON. Bit 1 stays ON during one scan.



◆ **PID Deadband Flag (Bit 2)**

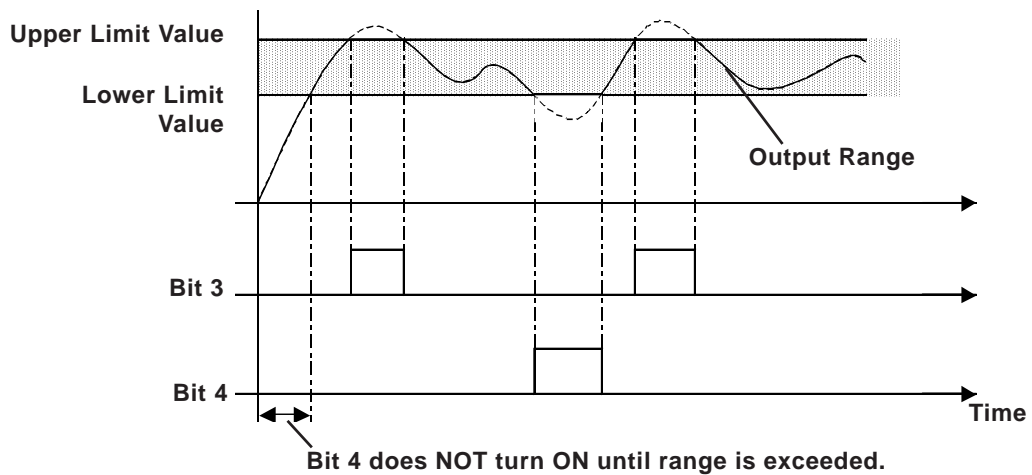
When the control variable is within the range specified in the Tune tab of the [PID] dialog box, or in Element [4] in the control block variable, bit 2 turns ON when the process variable reaches the setpoint. Bit 2 turns OFF when the process variable is outside the range.



**Note:** For details about the Tune tab in the [PID] dialog box, see the following ■ *Fine-Tuning Adjustments and Monitoring of PID Control* section.

◆ **Control Variable Exceeds the Upper Limit (Bit 3) or Lower Limit (Bit 4)**

When there is an output at the upper limit specified in the [Tune] tab of the [PID] dialog box, bit 3 turns ON. When there is an output at the lower limit, bit 4 turns ON. Even if a status bit is turned ON, PID calculation is performed and either the upper limit value or lower limit value will be output.



**Note:** For details about the Setup tab in the [PID] dialog box, see the following ■ *Fine-Tuning Adjustments and Monitoring of PID Control* section.

◆ **Exceeding the Number of Integration Process Times (Bit 5)**

When processing is performed for an integration frequency that is outside the range assigned in the Tune tab of the [PID] dialog box, bit 5 turns ON. Even if this status bit is turned ON, PID calculation is performed and the value is output at the upper limit.



**Note:** For details about the Setup tab in the [PID] dialog box, see the following ■ *Fine-Tuning Adjustments and Monitoring of PID Control* section.

■ **Control Block Variable Elements [1]–[6] Status**

Elements [1]-[6] perform fine-tuning adjustments of PID control.

▼ **Reference** ▲ For details, see the following ■ *Fine-Tuning Adjustments and Monitoring of PID Control* section.

■ **Fine-Tuning Adjustments and Monitoring of PID Control**

Clicking the instruction after setting up special variables and control block variables to the PID instruction displays the following [PID] dialog box. Fine-tuning adjustments and monitoring of the PID control settings are available in this dialog box.

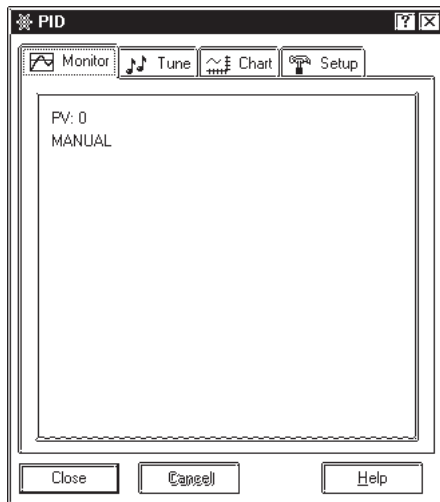


**Note:** There is no tuning feature to automatically adjust each parameter.

◆ **Monitor**

While in Monitoring mode, use the Monitor feature to monitor the PID instruction execution result.

▼ **Reference** ▲ *Pro-Control Editor Operation Manual Chapter 4 Online Editing*



**Type of Chart Lines**

The following table lists the types of lines and colors of each item monitored.

Item	Types of Lines/Colors	
Setpoint	Black dotted line	— · — · — · — ·
Process Variable	Black solid line	—————
Control Variable	Blue solid line	—————

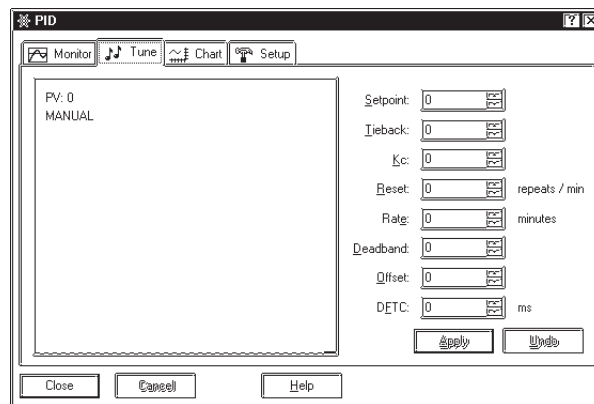


**Note:** Types and colors of graph lines cannot be changed.

## ◆ Tune

Each value can be adjusted while monitoring. The values set here reflect the special variables or control block variable elements [1] to [6].

**Reference** *Pro-Control Editor Operation Manual Chapter 4 Online Editing*



### Setpoint

Sets the target value.

### Tieback

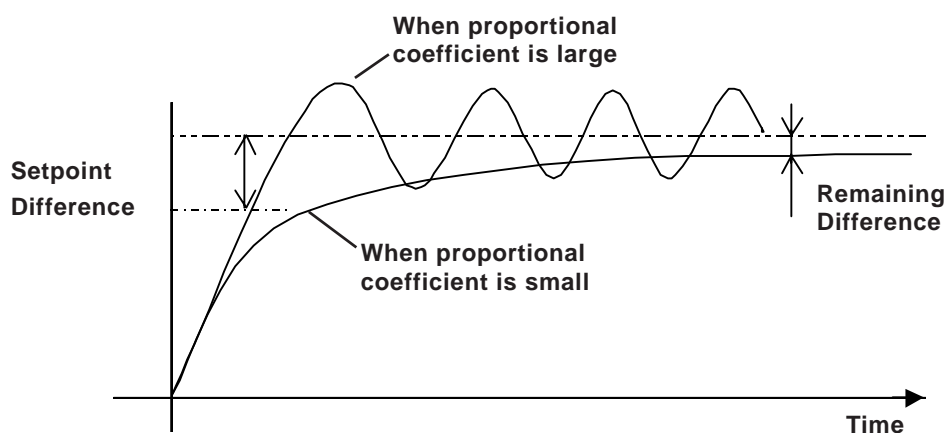
When the PID instruction in the logic program does NOT pass power, the value set here will be output.

### Proportional Coefficient [KC]

The Control Variable's output is based on the difference between the Process Variable and the Setpoint.

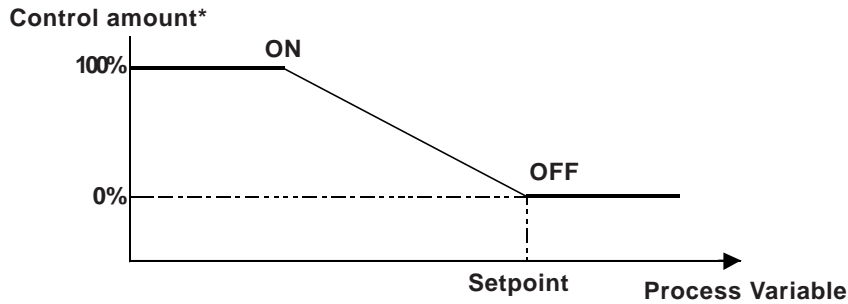
When the proportional coefficient is decreased, the control amount for bringing the Process Variable closer to the target value decreases, and overshoot is prevented. However, this may increase the remaining difference.

When the proportional coefficient is increased, the control amount for bringing the current value closer to the target value increases, and the length of time to reach the target value will shorten. However, this may cause hunting.





When using proportional control, when the Process Variable is smaller than the Setpoint and the control amount reaches its maximum limit at 100%, if the Setpoint and the Process Variable match (no difference), the control amount becomes 0%.



\* Control amount: output per unit time

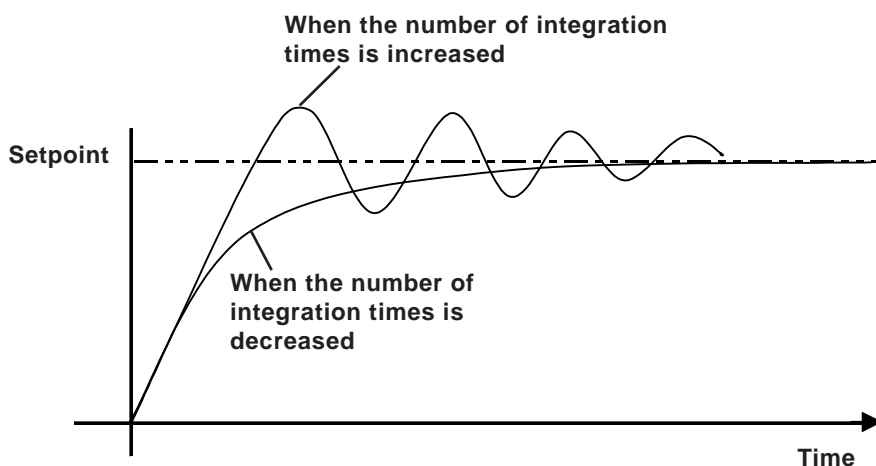
### Number of Integration Times [Reset]

The control amount alone can never completely eliminate the difference, since the control amount (control output) becomes too small when it gets close to the Setpoint. Using integral control, that remaining difference can be eliminated.

This control method makes adjustments based on the accumulated difference, over time, between the Process Variable and the Setpoint. If it reaches a certain level, it affects the output to reduce the difference.

When the number of integration times is increased, the control amount to reduce the difference increases. The length of time to reach the Setpoint will shorten. However, this may cause overshoot and hunting.

When the number of integration times is decreased, the control amount to reduce the difference decreases. Overshoot and hunting are eliminated. However, the length of time to reach the Setpoint will be greater.



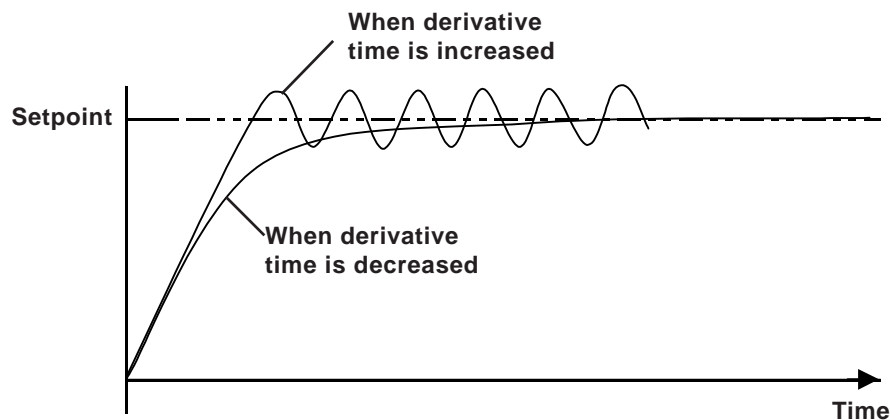
### Derivative Time [Rate]

Proportional control or integral control that requires a certain time (time constant) cannot respond quickly to a disturbance, and cannot return to the target value quickly.

Derivative control monitors the difference against the disturbance, and when a difference is large compared to the previous difference, a large amount of control is given to provide a quick response.

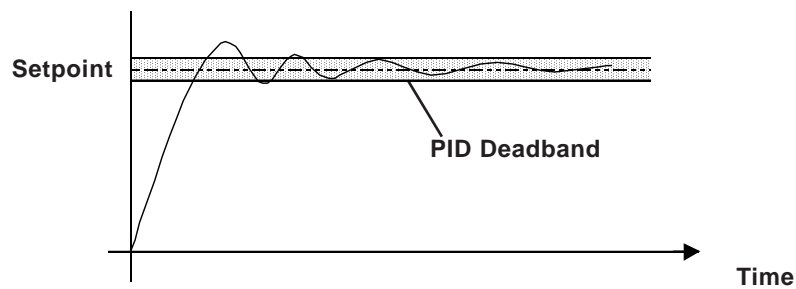
When the derivative time is increased, recovery time from the disturbance is shortened. However, this may cause overshoot and short cycle hunting.

When the derivative time is decreased, overshoot and hunting are eliminated. However, the recovery time from the disturbance will be greater.



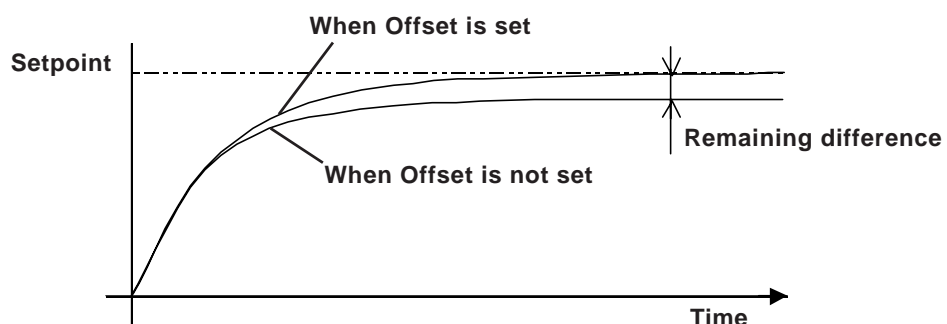
### Deadband

PID control is not performed in the deadband, and the minimum control variable value is output, which provides smooth control without hunting.

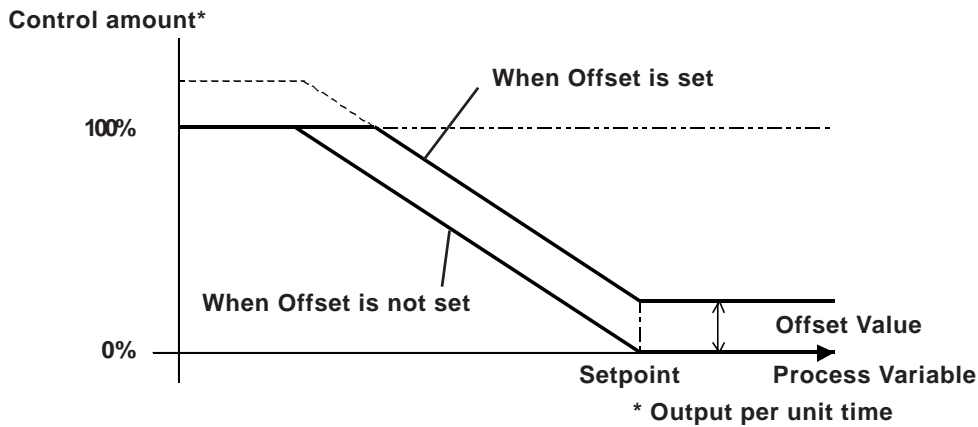


### Offset

Set offset value. Offset can reduce the remaining difference created by the proportional control.



## Chapter 4 – Instructions



### Sampling Time [DFTC]

Provides noise reduction of the connected device's measured data acquired by the Loop Update Time. Calculates a running average of the previous filtering result and the currently acquired data.

Setting the sampling time allows for the measured data to contain unexpected measurements. If the previously measured data is calculated as an average, the effect of unexpected measurements on the output value will be minor.

Sampling Time should be set to a larger value than the Loop Update Time. Also, setting the sampling value to "0" will disable the filter.

For loop update time information, refer to "Control" in the "Setup" tab.

### Reflect Tune tab setting to Control Block Variable

Each setting in the Tune tab is reflected in the parameter variables (SP and TB) and the elements [1] to [6] of the control block variable.

The following tables compare the Tune tab and Parameter variables, and the Tune tab and control block variable. 1000 times the values in the proportional coefficient, integral times, and derivative period are written in the control block variable.

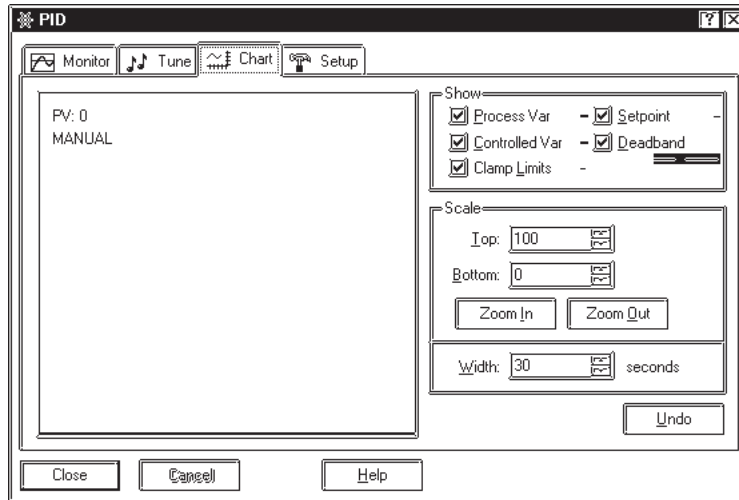
Tune Tab	Scale Factor	➔	Parameter Variables	
Setpoint	x 1		SP	Variable
Tieback	x 1		TB	Variable

Tune Tab	Scale Factor	➔	Control Block Variable	
Proportional Coefficient	x 1000		Proportional Coefficient	Control Block Variable [1]
Integral number of times	x 1000		Integral number of times per minute	Control Block Variable [2]
Derivative period	x 1000		Derivative number of times per operation	Control Block Variable [3]
Deadband	x 1		PID deadband	Control Block Variable [4]
Offset	x 1		Offset	Control Block Variable [5]
Sampling time	x 1		Sampling time	Control Block Variable [6]



◆ **Chart**

Process Variable (PV), Setpoint (SP), Control Variable (CV), Deadband, and Clamp Limits can be monitored. The monitoring setup is available in the Chart tab.



**Show**

Chart types and lines of each item monitored are listed the following table.

Item	Types of Lines/Colors
Setpoint	Black dotted line — — — — —
Process Variable	Black solid line —————
Controlled Variable	Blue solid line —————
Clamp Limits	Red dotted line - - - - -
Deadband	Gray zone



**Note:** Types of lines/colors cannot be changed.

**Scale**

Top: Set the upper limit of the chart

Bottom: Set the lower limit of the chart

Width: Set the width of the chart in seconds. Sampling time can be changed in the Preference area of the Monitoring tab by clicking Option in the Editor’s File menu.

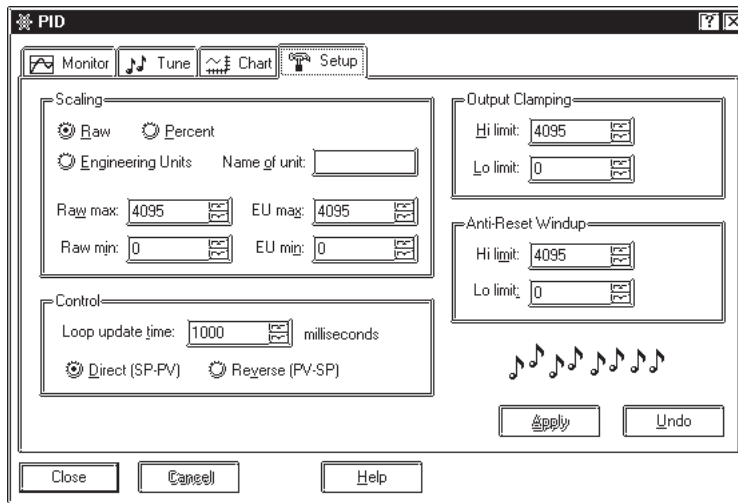


**Note:** Previous data cannot be monitored.

## Chapter 4 – Instructions

### ◆ Setup

You can preset the range (upper and lower limits) set to all parameters during the programming mode.



**Note:**

These settings are not available during the monitoring mode.

### Scaling

The Raw, Percent, and Engineering Units options set the conversion rate of PV data values with values, such as monitoring, in the display area. Raw max and Raw min values specify PV data values, and EU max and EU min values specify values, such as monitoring, in the display area.

**Raw:** All Input/Output values to the connected device are shown in raw form with a conversion rate of 0.

When Raw is selected, set the values of Raw max, Raw min, EU max, and EU min as follows.

- Raw max=EU max
- Raw min=EU min

**Percent:** Values in percent are set in the display area.

When Percent is selected, set Raw max, Raw min, EU max, and EU min as follows.

- Raw max and Raw min values = user-defined value by the connected device
- EU max = 100
- EU min = 0

**Engineering Units:** Values of n mole fraction, defined by the user, are set in the display area.

When Engineering Units is selected, set Raw max, Raw min, EU max, and EU min as follows.

- Raw max and Raw min values = user defined value by the connected device
- EU max = n
- EU min = 0

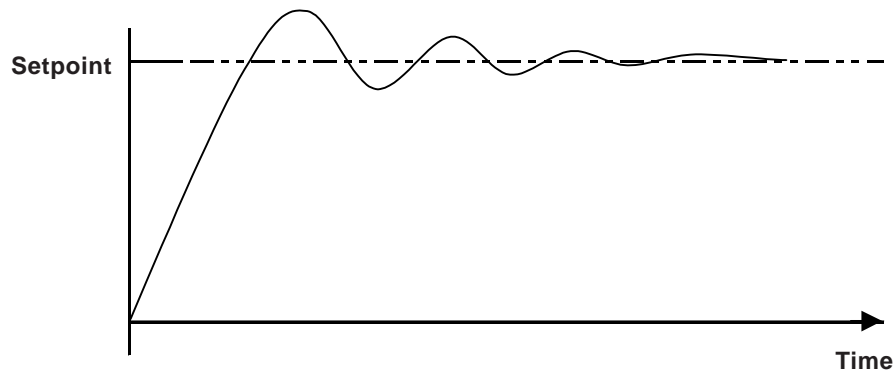
**Control**

Loop Update Time: Set the time cycle to acquire data from the connected device.

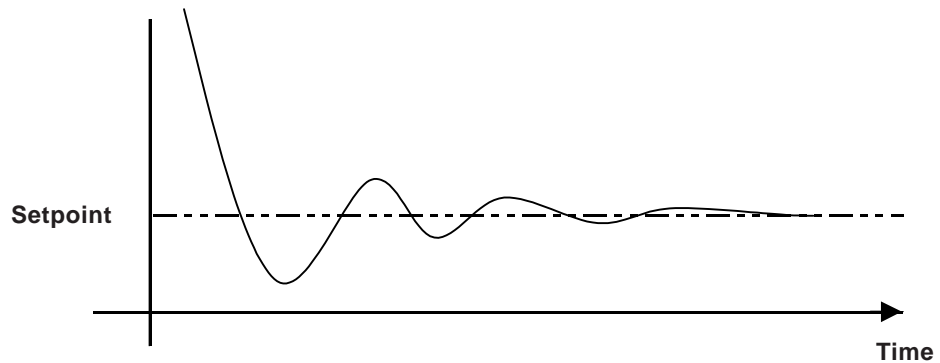
The Loop Update Time becomes the output update time/period.

Setting the Sampling Time allows you to use the Filter feature, however, the Sampling Time should be set to a larger value than the Loop Update Time.

Direct (SP-PV): Specify to perform control to increase the control amount output when the Process Variable is smaller than the Setpoint (such as heater).



Reverse(PV-SV): Specify to perform control to decrease the control amount output when the Process Variable is greater than the Setpoint (such as cooler).



## Chapter 4 – Instructions

### Output Clamping

Sets the highest limit and the lowest limit of the Control Variable. When the Control Variable is outside this range, the highest limit or the lowest limit is output, and the status bit of Bit 3 or Bit 4 in Element [0] of the control block variable turns ON.

**Reference** For details, see • *Control Variable Exceeds the Upper Limit (Bit 3) or Lower Limit (Bit 4) in the ■ Control Block Variable section.*

### Anti-Reset Windup

Sets the highest and lowest limits of the Number of Integration Times per minute of Element [2] of the control block variable.

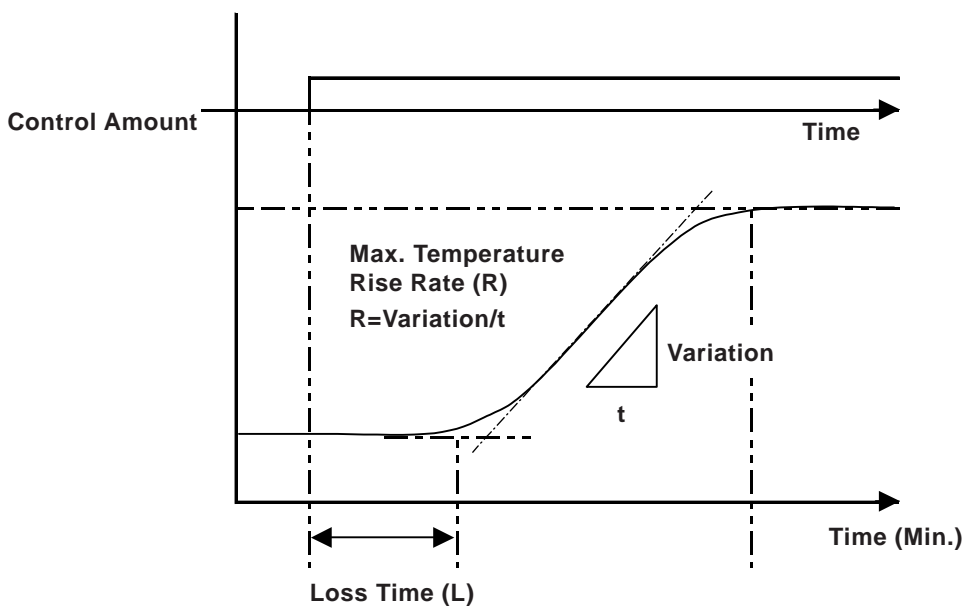
## ■ How to Adjust the PID Constant

This section explains how to adjust the PID constant using temperature control as an example. To obtain optimum PID control results, each constant of P (Proportional Element), I (Integral Element), and D (Derivative Element) has to be set to its optimum value. Step Response is one method of adjusting the PID constant against various control targets, and is based on temperature characteristics.

The optimum value might not be obtained with the Step Response method, depending on the control target used. In these cases, adjust the values in the Tune tab of the [PID] dialog box.

### Step Response

Step Response sets the Setpoint, and 100% of the control amount for the control target is output in steps. The following example, based on the chart of temperature characteristics, measures the maximum temperature slope (R) and the loss of time (L).



You can calculate constants of the proportional coefficient, the number of integral times, and the derivative time by substituting the measured values of the maximum temperature slope (R) and the loss time (L) in the following equation.

Please enter the calculated values in the Tune tab of the PID dialog box.

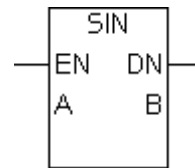
“Proportional Coefficient” =  $100/(0.83 * R * L)$  [%]

“Number of Integration Times” =  $1/(2 * L)$  [cycles/min.]

“Derivative Time” =  $0.5 * L$  [min.]

### 4.2.59 SIN (sine function)

- A: Data (In radians)
- B: Variable that stores the result



The SIN instruction computes  $\sin(A)$  and stores the value in B. Enter A data (values) in radian units.

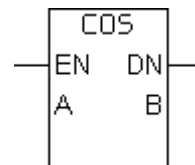
**Arithmetic Formula :  $B = \sin(A)$**

This instruction is normally ON. The following table lists the types of A and B data that can be used for this instruction.

If A is...	B must be...
Integer or integer constant	Real
Real or real constant	Real

### 4.2.60 COS (cosine function)

- A: Data (In radians)
- B: Variable that stores the result



The COS instruction computes  $\cos(A)$  and stores the value in B. Enter A data (values) in radian units.

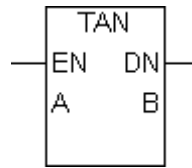
**Arithmetic Formula :  $B = \cos(A)$**

This instruction is normally ON. The following table lists the types of A and B data that can be used for this instruction.

If A is...	B must be...
Integer or integer constant	Real
Real or real constant	Real

**4.2.61 TAN (tangent function)**

- A: Data (In radians)
- B: Variable that stores the result



The TAN instruction computes  $\cos(A)$  and stores the value in B. Enter A data (values) in radian units.

**Arithmetic Formula :  $B=\tan(A)$**

This instruction is normally ON. The following table lists the types of A and B data that can be used for this instruction.

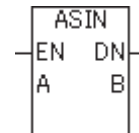
If A is...	B must be...
Integer or integer constant	Real
Real or real constant	Real

When the value used for A is in the vicinity of  $(2n-1) \times \pi/2$ , where n is an integer, B cannot be displayed. Therefore, #Overflow turns ON, and the solution is uncertain. ( $\pi = 3.1415926535897$ )

**Reference** see 3.2.17 - “#Overflow”

**4.2.62 ASIN (Arc Sine)**

- A: Data
- B: Result storing destination (Radian)



If you execute the ASIN instruction,  $\sin^{-1}(A)$  will be stored in B. The input A is between -1.0 and 1.0, the result B is output in radian and it is the real number between  $-\pi/2$  and  $\pi/2$ .

**Arithmetic Formula :  $B=\sin^{-1}(A)$**

This instruction always continues. The combination of A and B that you can execute the ASIN instruction is as the following.

Type of A	Type of B
Integer, or Integer Constant	Real
Real, or Real Constant	Real

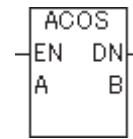


- $\pi = 3.1415926535897$
- If A is out of range, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

### 4.2.63 ACOS (Arc Cosine)

- A: Data  
B: Result storing destination (Radian)



If you execute the ACOS instruction,  $\cos^{-1}(A)$  will be stored in B. The input A is between -1.0 and 1.0, the result B is output in radian and it is the real number between 0 (zero) and  $\pi$ .

**Arithmetic Formula :  $B = \cos^{-1}(A)$**

This instruction always continues. The combination of A and B that you can execute the ACOS instruction is as the following.

Type of A	Type of B
Integer, or Integer Constant	Real
Real, or Real Constant	Real

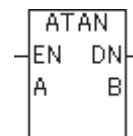


- $\pi = 3.1415926535897$ .
- If A is out of range, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

### 4.2.64 ATAN (Arc Tangent)

- A: Data  
B: Result storing destination (Radian)



If you execute the ATAN instruction,  $\tan^{-1}(A)$  will be stored in B. The result B is output in radian and it is the real number between  $-\pi/2$  and  $\pi/2$ .

**Arithmetic Formula :  $B = \tan^{-1}(A)$**

This instruction always continues. The combination of A and B that you can execute the ATAN instruction is as the following.

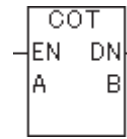
Type of A	Type of B
Integer, or Integer Constant	Real
Real, or Real Constant	Real



- $\pi = 3.1415926535897$

**4.2.65 COT (Cotangent)**

A: Data (Radian)  
 B: Result storing destination



If you execute the COT instruction,  $1/\tan(A)$  will be stored in B. The input A is in radian. Although the absolute value of the result B becomes greater as A becomes an approximation of a multiple of  $f\hat{I}$ , the expressible range is approximately between  $+2.225e_{-308}$  and  $+1.79e+308$  (real number).

**Arithmetic Formula :  $B=1/\tan(A)$**

This instruction always continues. The combination of A and B that you can execute the COT instruction is as the following.

Type of A	Type of B
Integer, or Integer Constant	Real
Real, or Real Constant	Real

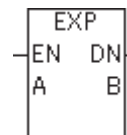


**Note:**  $\pi = 3.1415926535897$

**Reference** See 3.2.17 – “#Overflow.”

**4.2.66 EXP (Exponent)**

A: Data  
 B: Result storing destination



If you execute the EXP instruction, the exponential function of A will be stored in B. The result B is output as a real number of e to the "A"th power.

**Arithmetic Formula :  $B=e^A$**

This instruction always continues. The combination of A and B that you can execute the EXP instruction is as the following.

Type of A	Type of B
Integer, or Integer Constant	Real
Real, or Real Constant	Real



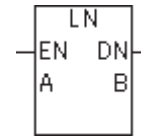
- $e = 2.7182818284590$ .
- If the result B is out of range, which can be expressed by the variable type of B, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”



### 4.2.67 LN (Natural Logarithm)

- A: Data  
B: Result storing destination



If you execute the LN instruction, the natural logarithm function of A will be stored in B. The result B is output as a real number where e to the "B"th power is equal to A.

#### Arithmetic Formula : $B = \log_e A$

This instruction always continues. The combination of A and B that you can execute the LN instruction is as the following.

Type of A	Type of B
Integer, or Integer Constant	Real
Real, or Real Constant	Real



#### Note:

- $e = 2.7182818284590$ .
- If the result B is out of range, which can be expressed by the variable type of B, #Overflow will be set to ON. The result in this case is "Undefined".

**Reference** See 3.2.17 – “#Overflow.”

# *Memo*

# 5 LS Area Refresh

## 5.1 LS Area Refresh Overview

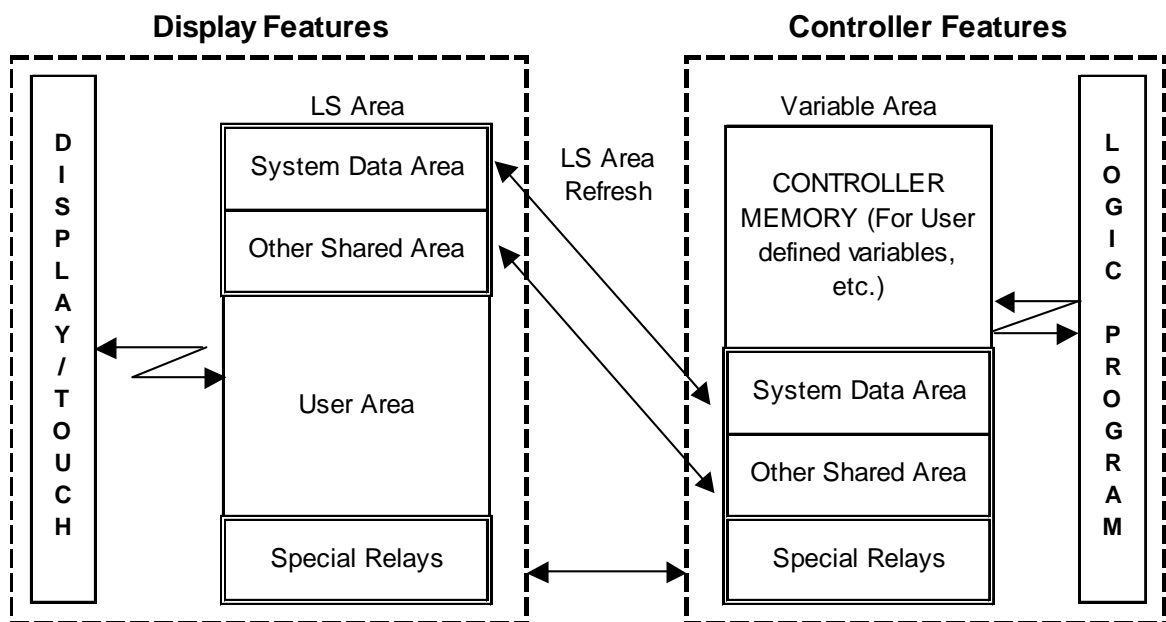
### ■ LS Area Refresh Feature

The GLC unit uses the LS Area's System Data Area to control the changing of screens, the sounding of buzzers, etc. These are processed as GLC display features.

Therefore, when using the functions assigned to the System Data Area (such as the screen change and clock function) via the Controller Feature, the data in the LS Area should be shared between the Display Features and Controller Features by registering the LS Area as variables.

This is defined as the LS Area Refresh.

It is also possible to use an area outside of the System Data Area if the GLC unit's controller features or display features need to share data.



The timing of the LS Area Refresh and the Logic Symbol data update is not synchronized. When designing your logic program to use either of these as a trigger for data update, be sure to include an interlock feature.

## 5.2 LS Area Refresh Settings

When using a logic program to designate the LS Area, the desired variable must first be registered in Pro-Control Editor. This section describes this procedure.

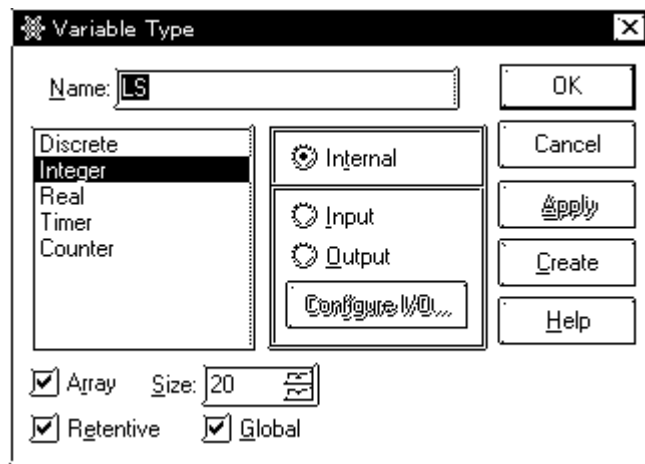
### ■ Variable Registration

In Pro-Control Editor’s Data menu, click Variable Type to open the Variable Type dialog box.

This section describes how to register a variable with a variable name "LS" as an internal integer array.

The size should be calculated by combining the number of words of data to be shared with the size (20 words) of the System Data Area.

E.g.: When sharing 16 words of data with the System Data Area, enter the size as “36” words (20 words for the System Data Area plus 16 words to be shared).



**Note:**

- The Special Relay Area is called the LSS area.
- The maximum LS size is 276 words.

The relationship between variables and addresses are listed in the following table.

Variable Name*1	Address	LS Address	
LS[0]	0	LS0000	} System Data Area
LS[1]	1	LS0001	
:	:	:	
LS[19]	19	LS0019	
:	:	:	} Other Shared Data
LS[275]	275	LS0275	
LSS[0]	2032	LS2032	} Special Relays
LSS[1]	2033	LS2033	
:	:	:	
LSS[15]	2047	LS2047	

1. Variable Name: System Variables managed by the GLC unit's ladder logic program

**Reference** For details on the "LS Area" and "Special Relays", refer to the **External Device Connection Manual** (provided with Pro-Control Editor).

■ **GLC Screen Number Display Confirmation Example**

When confirming the GLC’s screen display number, the LS Area to be accessed varies depending on the access method used. Two access methods are available: Direct Access method, and Memory Link method.

Screen Display Number : LS[0] Direct Access method  
 : LS[15] Memory Link method

In the following example logic program, the “Message” coil turns ON when the screen number switches to “5”.

**Direct Access method (Connected to External Communication Device)**



**Memory Link method (Not connected to External Communication Device)**



**5.2.1 LS Area - When not using a Device/PLC**



**Do NOT use any areas designated as Reserved.**

**■ System Data Area**

You can use controller features in your logic program to update this area and control the "GLC Screen Change" and "Backlight ON/OFF".



**This area can be accessed by registering Pro-Control Editor's internal "[LS]" integer array variables.**

Address	Var. *1 Name	Detail	Function	Bit	Particulars
1	LS[1]	Status		0, 1	Reserved
				2	Now Printing
				3	Writes a set value
				4 to 9	Reserved
				10	Backlight Burnout Detection
				11 to 15	Reserved
2	LS[2]	Error Status  Each bit changes according to the GLC error function. When an error occurs, the corresponding bit will turn on.		0, 1	Unused
				2	System ROM/RAM
				3	Screen Memory Checksum
				4	SIO Framing
				5	SIO Parity
				6	SIO Overrun
3	LS[3]	A bit that has turned ON remains ON until the power is turned OFF and back ON, or until RUN mode is re-entered from OFFLINE mode.		7, 8	Unused
				9	Initialization of Internal Memory Checksum Necessary
				10	Timer Lock Error
				11 to 15	Unused
4	LS[4]	Clock Data (Year)	"Year / Month / Day / Hour / Minute " Data is stored in BCD's 2digits. (E.g.) 98/02/01 17:15	0 to 7	Stores the last 2 digits of the Calendar year
5	LS[5]	Clock Data (Month)		8 to 15	Unused
				0 to 7	Stores 01 to 12 (Month) as 2 BCD digits
6	LS[6]	Clock Data (Day)		8 to 15	Unused
				0 to 7	Stores 00 to 31 (Day) as 2 BCD digits
7	LS[7]	Clock Data (Hour)		8 to 15	Unused
				0 to 7	Stores 00 to 23 (Hour) as 2 BCD digits
8	LS[8]	Clock Data (Minute)		8 to 15	Unused
			0 to 7	Stores 00 to 59 (Minute) as 2 BCD digits	
10	LS[10]	Interrupt Output (Touch OFF)			If you use a Switch Part to write in word data, the bottom 8 bits will be output as an interrupt code after Touch OFF. FFh will not be output.

1. Variable names used when accessing the GLC.

LS Address	Var. *1 Name	Detail	Function	Bit	Particulars
11	LS[11]	Control		0	Backlight
				1	Buzzer ON
				2	Starts Printing
				3	Reserved
				4	Buzzer - - - 0:enabled 1: disabled
				5	Reserved
				6	Interrupt Output when touching panel to turn the display ON. (Interrupt Code:FFh) 0: Disabled 1: Enabled
				7 to 10	Reserved
				11	Hard copy output - 0: Enabled 1: Disabled
				12 to 15	Reserved
12	LS[12]	Screen Display ON/OFF	FFFFh : Screen clears almost immediately 0h: Screen turns ON		
13	LS[13]	Interrupt Output	Using a Switch Part or other method to write absolute value data from LT causes an output of the interrupt code using the contents of the bottom 8 bits ( Will not output FFh)		
15	LS[15]	Screen Display No.	Write the Screen No. in binary to change the	0 to 14	Screen change number, 1 to 8999. ( 1 to 1999 when using BCD input)
				15	Forced Screen Change 0: Normal 1: Forced Screen Change
16	LS[16]	Window Control		0	Display :0:OFF, 1:ON
				1	Invert the window overlap order 0: Possible, 1: Not possible
				2 to 15	Reserved
17	LS[17]	Window Registration Number	Window number used for indirectly designated Global windows. (either BIN or BCD)		
18	LS[18]	Window display position (x-coordinate)	Window number used for indirectly designated Global window display location. (either BIN or BCD)		
19	LS[19]	Window display position (y-coordinate)			

1. Variable names used when accessing the GLC.

## Chapter 5 – LS Area Refresh

### ■ Special Relay



***This area can be accessed by registering Pro-Control Editor's internal "[LSS]" integer array.***

LS Address	Var. Name *1	Contents
2032	LSS [0]	Share Relay Data
2033	LSS [1]	Base Screen information
2034	LSS [2]	Reserved
2035	LSS [3]	Binary Counter - 1 second
2036	LSS [4]	Tag Scan Time
2037	LSS [5]	Reserved
2038	LSS [6]	Tag Scan Counter
2039	LSS [7]	Reserved
2040	LSS [8]	
2041	LSS [9]	
2042	LSS [10]	
2043	LSS [11]	
2044	LSS [12]	
2045	LSS [13]	
2046	LSS [14]	
2047	LSS [15]	

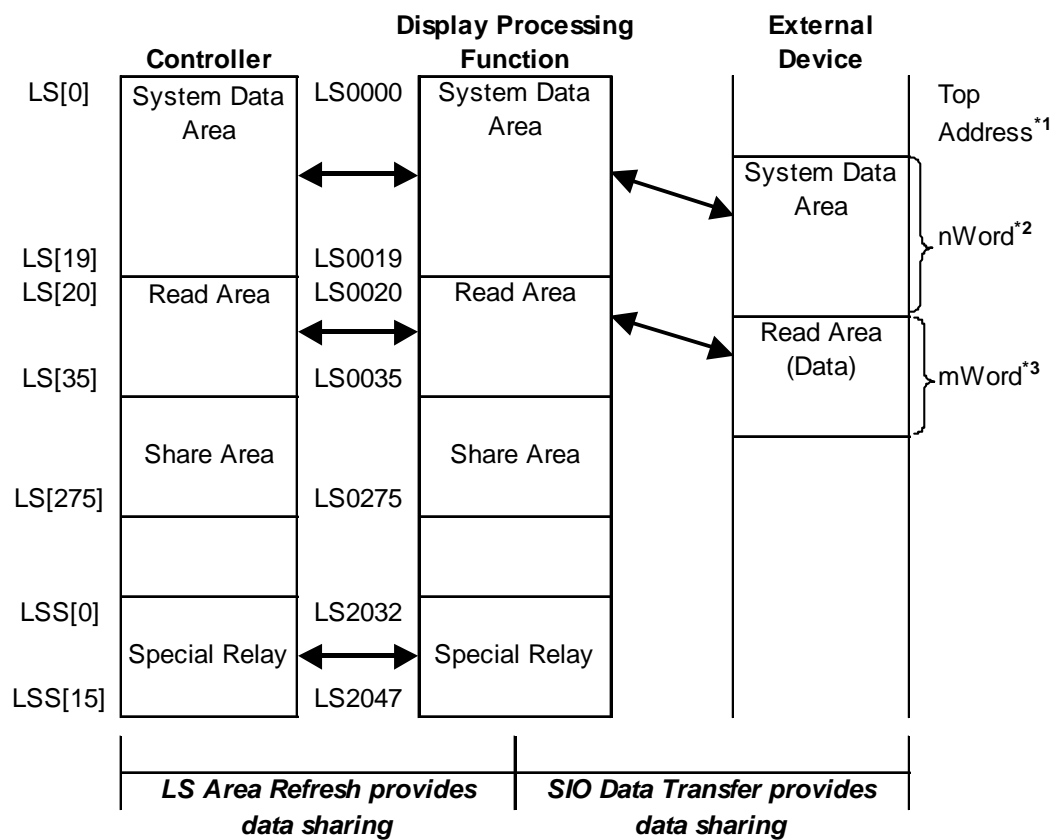
1. Variable names used when accessing the GLC.



## 5.3 GLC and External Device Data Sharing

When using external communication device data with the controller features, the data is shared via the LS Area. However, if data sharing between the controller functions and the external device's data register exceeds a size of 16 words, the performance of screen display functions may deteriorate.

**Reference** For details on System Data Area, see "GP-PRO/PBIII for Windows Ver. 7.0 Device/PLC Connection Manual Mitsubishi Electric Corporation Direct Access Communication Contents and Range of System Data Area".

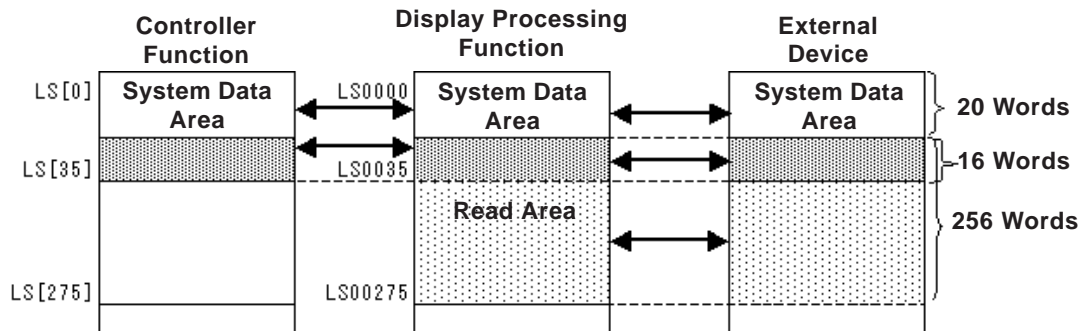


1. Start Address defined in Initial Settings.
2.  $n = 0$  to 20 Depends on the System Data Area setting items selected in Initial Settings.
3.  $m = 0$  to 16 Depends on the size of the Read Area designated in Initial Settings.

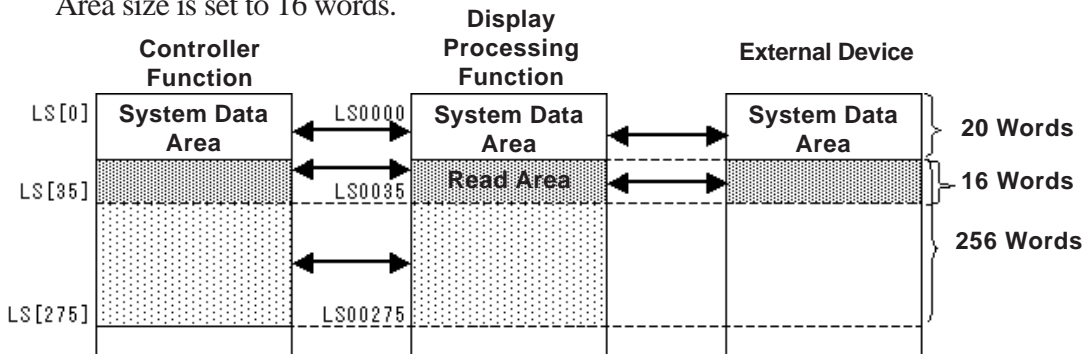
## Chapter 5 – LS Area Refresh

If you want the Read Area and Variable LS to exceed 16 words, the Read Area can be set with up to 256 words, and Variable LS can be set with up to 276 words. A maximum size of 16 words is recommended when setting data that is shared between the controller, display processing function, and external devices.

In the following example, the Variable LS size is set to 36 words and the Read Area size is set to 256 words.



In the following example, the Variable LS size is set to 276 words and the Read Area size is set to 16 words.

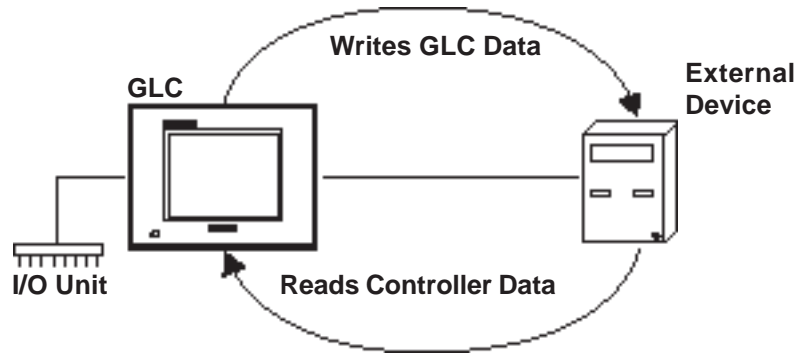


- **When the controller's logic program, the tags or Parts on the Display Processing function, or a logic program from an external device unit, attempts to update the same variable at the same time, priority is determined by the timing.**
- **When writing data to the GLC unit's Read Area, be sure that the Write Via Parts and Write Via Logic Program functions of the Controller do not conflict.**
- **Be sure to designate Controller variables LS and LSS as retentive variables. Designating these variables as non-retentive will clear them to 0 when the logic program starts. The display processing function's LS Area will be cleared to 0, depending on LS Refresh.**

**Note:**

Using the Read Area to share data between the GLC and external devices allows you to use the GLC as:

- the expansion unit of an external device
- POP machines for factory automation
- an I/O information terminal for production control



### 5.3.1 LS Area Refresh Cautions

Use the LS Area Refresh feature:

- to control the system area using the controller feature
- to view Read Data from an external communication device.

Digital Electronics Corporation recommends that you use the data send/receive related Initialize area or the Operation Designation Change parameter settings to control the refreshing of data in this area. Avoid refreshing the data intermittently, via the controller feature, in addresses LS000-to-LS0035 and LS2032-to-LS2047.

If the frequency of the LS Area's data refresh is increased, the LS Area Refresh may not be executed within one scan. As a result, External Device communication errors may occur.

Variable LS is an integer variable, and is 32 bits in length.

When the System Data Area is 16 bits in length, the low 16 bits are enabled.

*Memo*

# 6

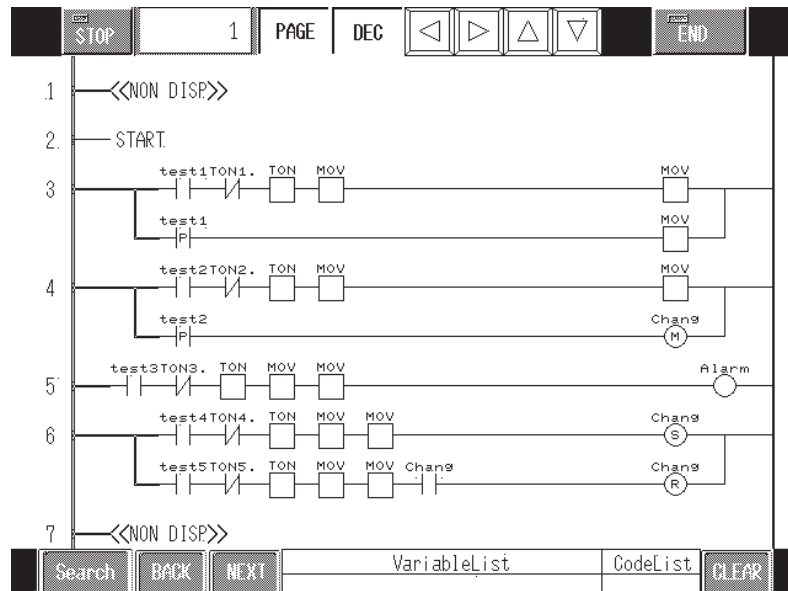
# GLC Ladder Monitor Feature

The GLC ladder monitor feature is also available for LT series. This chapter describes the feature based on GLC. The contents are also useful to LT users by replacing the word "GLC" with "LT".

## 6.1 Overview of the GLC Ladder Monitor Feature

Pro-Control Editor features a ladder monitor that can be used with the GLC unit in order to improve maintenance of a logic program. All programs (such as a logic program) are executed during the ladder monitoring.

The logic program is displayed on the screen of the GLC unit. The GLC2000 Series and the LT Series models support this feature.



**Note:** GLC ladder monitor screens cannot be monitored using GP-Web or GP-Viewer software.

## 6.2 Starting/Exiting the GLC Ladder Monitor

### 6.2.1 Preparing to operate the GLC Ladder Monitor

To operate the GLC Ladder Monitor feature, the project file for the GLC Ladder Monitor, as well as the editor screen and the logic program, must be transferred to the GLC unit.

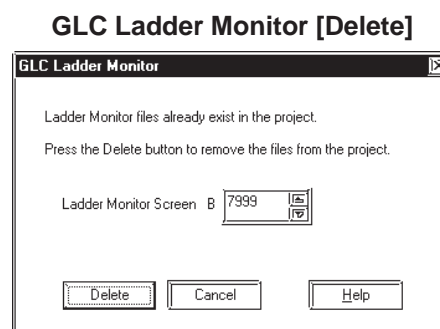
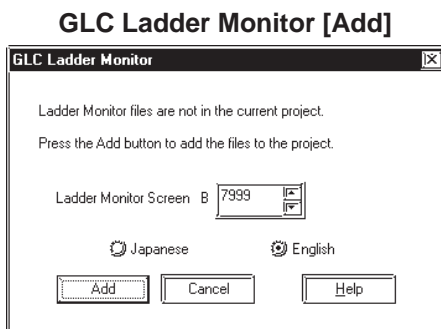
This section describes how to set up a project file, and precautions to use.

#### ■ Set Up the Project File

1. From the Project Manager’s [Screen/Setup] menu, click [GLC Ladder Monitor] to open the [GLC Ladder Monitor] dialog box, and to check that ladder monitor screens (base screens used in the ladder monitor) exist in the project.



2. If the screen is not registered, the [GLC Ladder Monitor] dialog box (as shown on the left) will appear. If the screen is already registered, the [GLC Ladder Monitor] dialog box (as shown on the right) will appear.

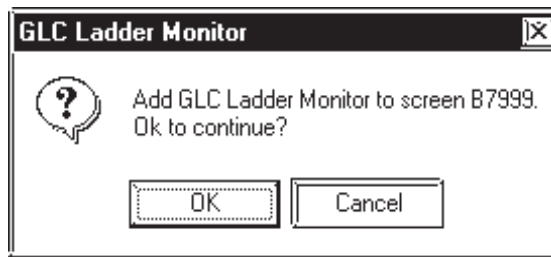


Assign the base screen number used by the GLC ladder monitor, and click [Add]. Set the screen number from 1 to 7999.

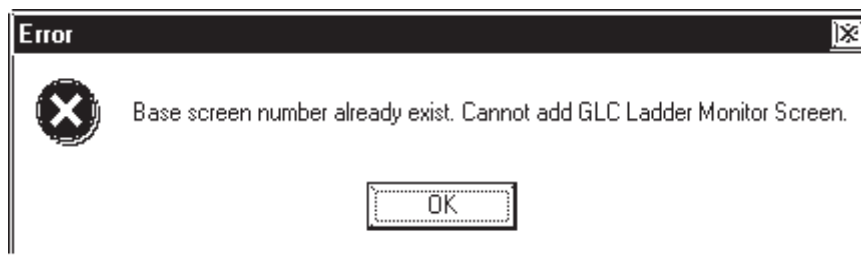


**Note:** The base screen number used by the GLC ladder monitor should not be the same as other existing screens.

- Click [Add] to add the GLC Ladder Monitor to the screen. When registering the Ladder Monitor, the following confirmation message appears.



If the assigned screen already exists, the following message is displayed and the screen of the GLC Ladder Monitor is not registered. To re-register the GLC Ladder Monitor screen to the assigned screen number, first delete the existing screen, and then register the screen again.



- The GLC Ladder Monitor screen can be edited since it is automatically generated to a base screen. However, if the GLC Ladder Monitor screen is edited in the Drawing Board even once, the registration of the GLC ladder monitor screen is automatically cancelled, since the edited GLC Ladder Monitor will not be recognized as the GLC Ladder Monitor screen.***

***Delete the existing GLC Ladder Monitor screen from the [Open Screen] dialog box (open the Drawing Board's Screen menu and click Open Screen), and then add the GLC Ladder Monitor screen again.***

- The GLC Ladder Monitor screen will not be recognized when it is copied to another project.***

***Delete the existing GLC Ladder Monitor screen from the [Open Screen] dialog box (open the Drawing Board's Screen menu and click Open Screen), and then add the GLC Ladder Monitor screen again.***

- When the GLC type (GLC2400, GLC2600) is changed, the screen will not be recognized as the registered GLC Ladder Monitor screen. Delete the existing GLC Ladder Monitor screen, then add the GLC Ladder Monitor screen again.***

### 6.2.2 Starting the GLC Ladder Monitor

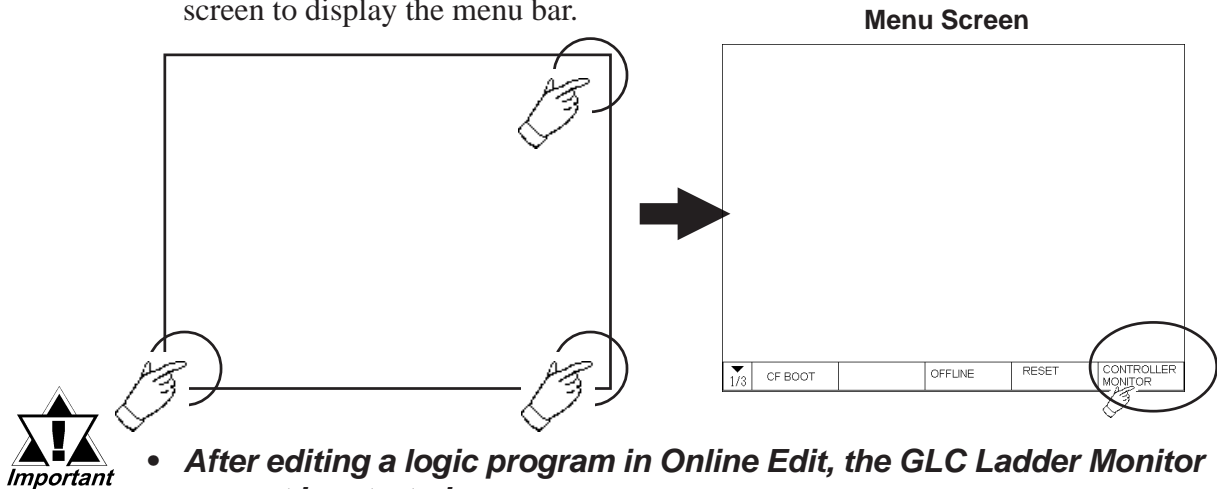
The two methods of starting the GLC Ladder Monitor are as follows.

- Turn ON Bit 0 of the System Variable #LadderMonitor.

**Reference** See 3.2.29 – “#LadderMonitor.”

*(When #Ladder Monitor Bit Operation is used and the GLCLadder Monitor is started, [Screen Level Change] mode cannot be used.)*

- Touch the [Controller Monitor] on the menu bar, then touch three corners of the screen to display the menu bar.



- **After editing a logic program in Online Edit, the GLC Ladder Monitor cannot be started.**
- **While the GLC Ladder Monitor is running, Online Edit cannot be executed.**
- **When a password is given to the project file that is transferred to the GLC unit, the GLC Ladder Monitor cannot be started.**

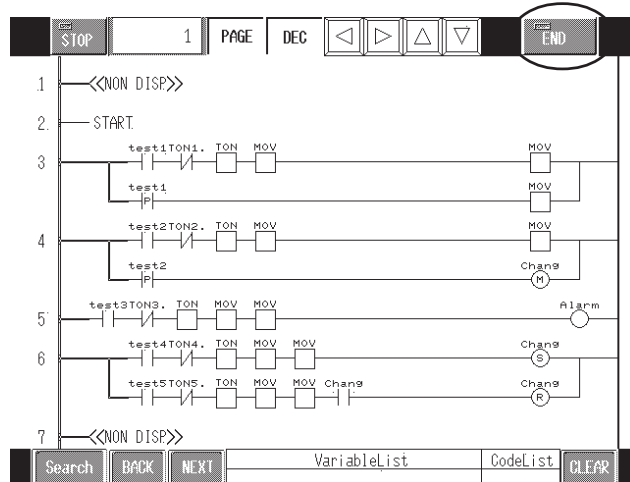
### 6.2.3 Exiting the GLC Ladder Monitor

The three methods of exiting the GLC Ladder Monitor are as follows.

- Turn OFF Bit 0 of the #LadderMonitor system variable and press [END] on the GLC Ladder Monitor screen.

**Reference** See 3.2.29 – “#Ladder Monitor.”

- Automatically switch the screen from the PLC.
- Press [END] on the GLC Ladder Monitor screen.



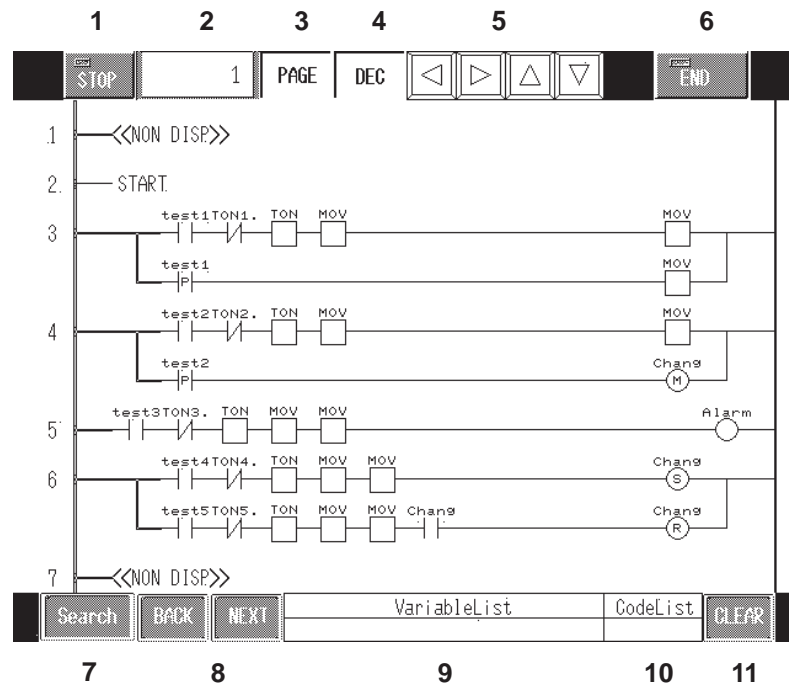


## 6.3 Various GLC Ladder Monitor Features

### 6.3.1 Online Monitor Feature (Normal Display)

A logic program is displayed on the GLC unit, and the rung that the power passes through is displayed with a green bolded line.

The number of instructions can be displayed on one screen are 18 horizontal instructions and 13 vertical rungs (four instructions can be viewed by scrolling).



1. RUN/STOP Switch button  
Switches the state of the GLC controller between RUN/STOP.
2. Rung Number Display button  
Displays the starting rung number of the screen being displayed.
3. Scroll Unit Switch button  
Selects the unit of screen scroll between rung/page.
4. Display Base Switch button  
Switches the current value display of the variable between decimal/hexadecimal.
5. Scroll button  
Scrolls the screen being displayed in Up and Down and Right and Left.
6. GLC Ladder Monitor Switch  
Switches between starting/exiting the GLC ladder monitor.
7. Search button  
Starts a search after assigning the instruction and variable.

## Chapter 6 – GLC Ladder Monitor Feature

8. Search Again button  
In the search mode, repeat the search back and next.
9. Variable List button  
Starts up the variable monitoring screen.
10. Code list button  
Changes the instruction assigning screen to do the instruction search.
11. Clear button  
Exits from the search mode.



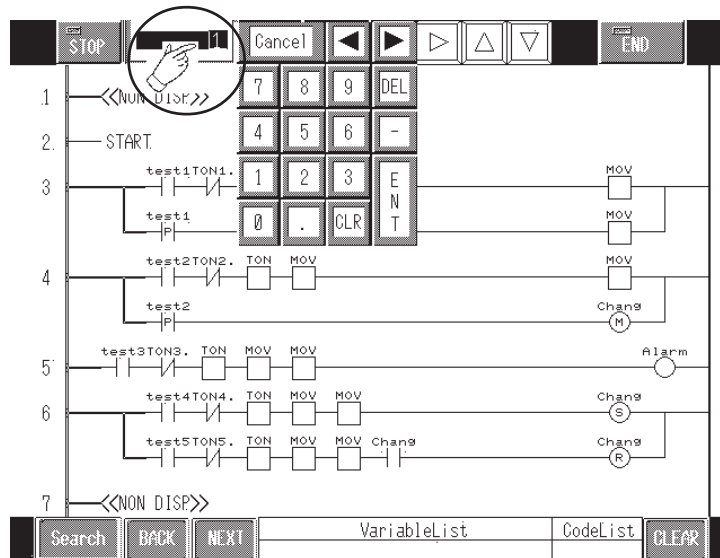
- **When more than 18 instructions are placed along the horizontal power flow, those following the 18th instruction will not be displayed, even by scrolling.**
- **Rungs exceeding 13 lines are not displayed and "NON DISP" appears on the screen instead.**
- **On a rung that has several vertical lines, if not all rungs are displayed by scrolling, the screen will display "NON DISP."**
- **The first five characters will display for the names of a contact and coil instruction's variables.**

### 6.3.2 Rung Jump/Scroll Features

This feature is used to move the screen to the rung you want to display when all logic program cannot be displayed in one screen.

#### ■ Rung Jump Feature

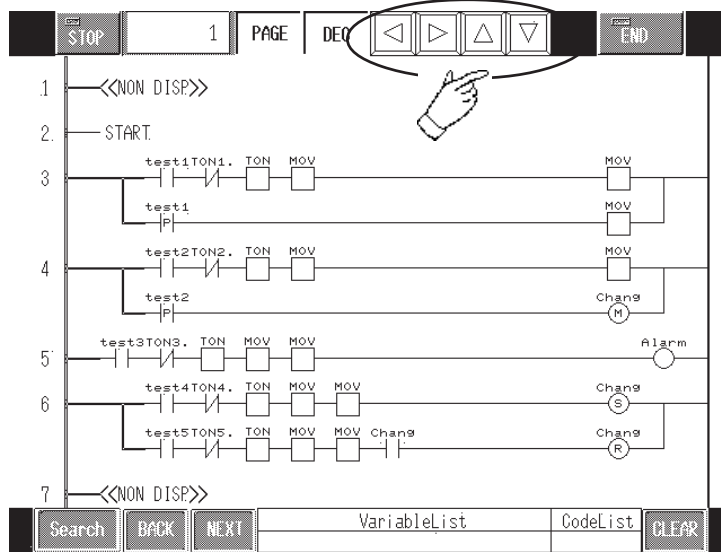
1. Touch the rung number display button to display the keyboard.



2. Enter the rung number to be displayed.
3. Press the [ENT] button to jump to the screen of the entered rung number.

■ Scroll Feature

Pressing the scroll keys to move the display screen.

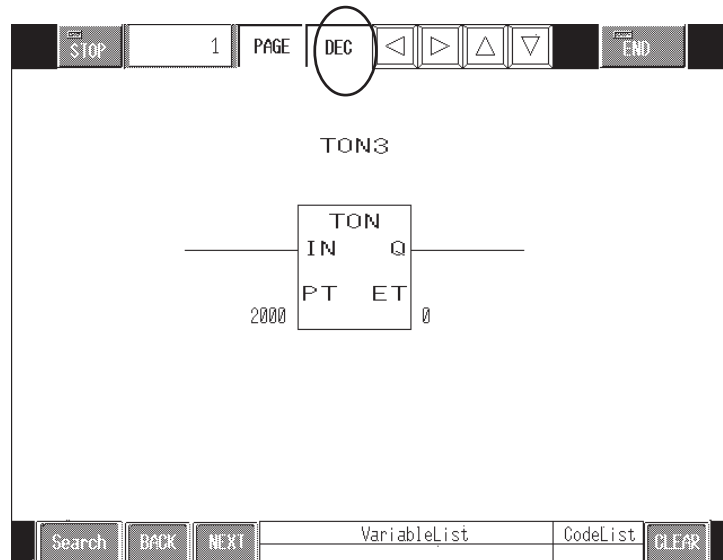


**6.3.3 Instruction Enlarge Feature (Zoom Display)**

The instruction can be zoomed in on by touching the instruction on the normal view screen of the GLC Ladder Monitor. The variable name is displayed in full view by zooming in on it.

Also, for the Timer, Counter, or MOV instructions, current value and preset value are also displayed, and the Display Base Switch button can switch the current value display to decimal or hexadecimal.

Once again, by touching the screen, it will go back to the normal view.



**Note:**

- Up to 32 characters can be viewed for variable name or variable value.
- When the array variable is assigned by BMOV, the current value of the array variable's starting element is displayed.

## Chapter 6 – GLC Ladder Monitor Feature

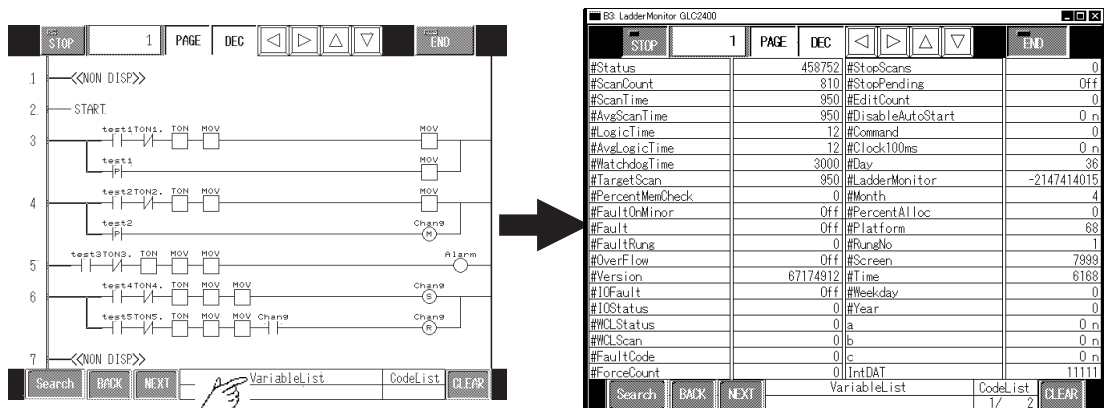
### PID Instruction Parameters (GLC2400/GLC2500/GLC2600 Series supported)

Zooming in on the PID instruction displays 7 parameters in the screen's lower right area. The parameters are displayed in decimal/hexadecimal format, similar to element 7's array that has been allocated to the control block variable.

- Status : Displays the current status.
- Kc : Proportional Coefficient
- Reset : Integral Time
- Rate : Derivative Time
- Deadband : PID Deadband Range
- Offset : Displays the offset
- DFTC : Displays the sampling time

### 6.3.4 GLC Variable Monitor Feature

Displays a list of variables and current values of each variable.



**Note:** To return to a normal screen from the variable list screen, touch “Clear.”

#### ◆ Display Color of Variable Name (For color monitors)

Variable Attributes	Display Color
System Variable	Purple
User Defined Variable	White

#### ◆ Display Color of Variable Value (For color monitors)

Variable Type	Display Color
Discrete	Green
Integer	White
Real	Light Blue
Timer	Purple
Counter	Yellow



- **The GLC2600 unit displays up to 32 characters for a variable name. However, if the 32nd character is for the 1st byte of a double-byte character, 31 characters are displayed at maximum.**
- **For GLC2300/GLC2400/GLC2500/LT, they display up to 24 single-byte characters for a variable name. However, if the 24th character is for the 1st byte of a double-byte character, 23 characters are displayed at maximum.**
- **Touch the corresponding variable to display the exclusive variable such as the timer or the counter variable.**
- **Touch the corresponding variable to display all elements of an array variable. The value displayed in the variable list is the current value of the header element. Use [CLEAR] button to go back to the variable list screen.**

### 6.3.5 Setup Value Edit Feature

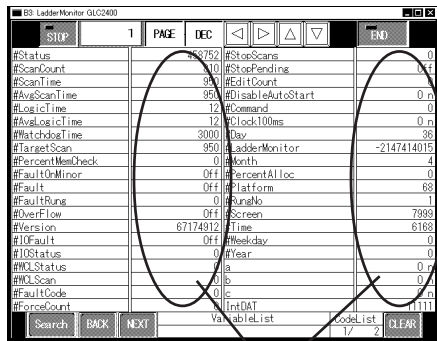
This allows you to switch ON/OFF contact points and to modify setup values of the bit, integer, timer and counter variables.



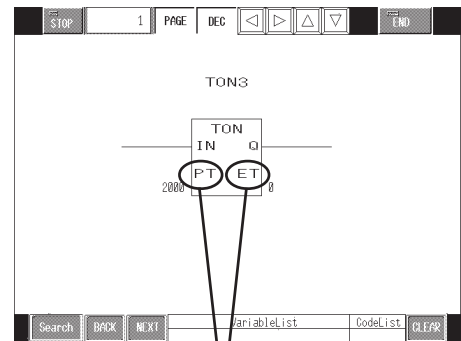
- **Corresponding GLC units are GLC2300 Series, GLC2400 Series (Rev . \*-Above2\*1), GLC2500 Series, GLC2600 Series (Rev . \*-Above2\*1).**

#### ■ Switching to the Setup Value Edit Screen

To edit setup values, touch either the current value display area in the variable list or the operand being zoomed in to switch to "Setup Value Edit Screen".



Current Value Display Area



Operand

#### ■ Operations in the Setup Value Edit Screen

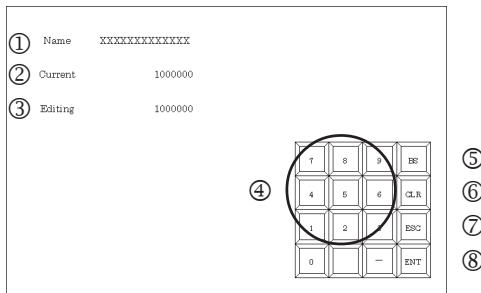
The setup value edit screen for integer variables differs from that for bit variables as shown in the figure below. Use following operation buttons to switch the contact point and to edit setup values.

The screen shown below is that of GLC2400/GLC2500/GLC2600. The screen sizes of GLC2300 and LT are smaller than this, but the operation method of each button is same as described below.

\*1 For how to distinguish "Revisions", refer to "For GLC2400/GLC2600 Users".

## Chapter 6 – GLC Ladder Monitor Feature

### Integer Variable



① **Variable Name**

Displays the variable name.

② **Current Value**

Displays the current value.

③ **Edit Value**

Displays the value being edited.

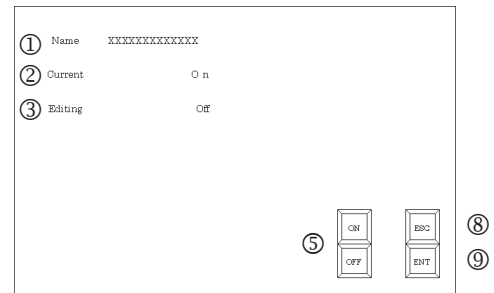
④ **0-9 Buttons (10-Keys)**

For entering values to edit.

⑤ **ON/OFF Buttons**

Switches the contact point of the bit variable.

### Bit Variable



⑥ **BS Button**

Deletes the last number of the value being edited.

⑦ **CLR Button**

Clears the value being edited to 0 (zero).

⑧ **ESC Button**

Exits without saving the setup value edit.

⑨ **ENT Button**

Saves the setup value edit and then exits.

### 6.3.6 Variable/Instruction Search Feature

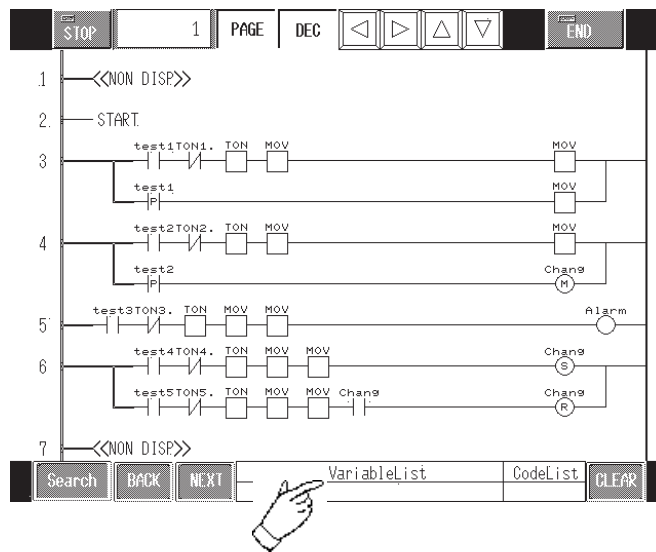
Searches a variable in the Variable List, or an instruction from a logic program specified in the Code List. The matching variable or instruction will be highlighted with a light blue box.

#### ■ Search from Variable

Searches the variables specified in the Variable List screen.

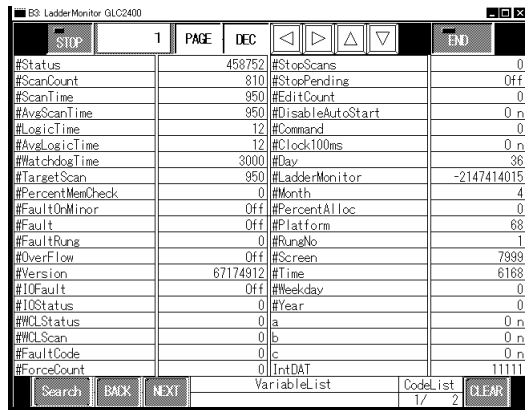
The search method is as follows.

1. Press Variable List on the bottom of the normal display screen, and display the Variable List screen.

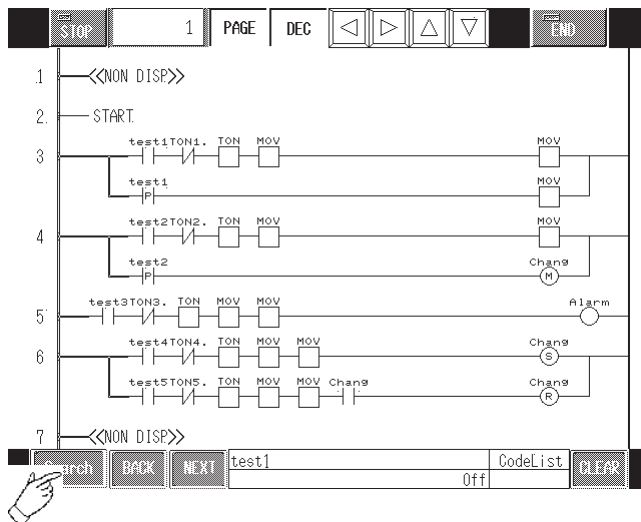


## Chapter 6 – GLC Ladder Monitor Feature

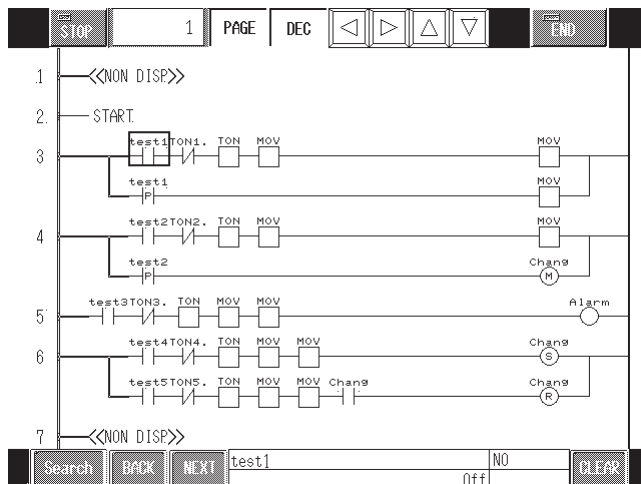
- In the variable list screen, system variables and all variables included in a logic program are displayed. Select by touching the variable you want to search for.



- After the variable is selected in the variable list screen, the screen automatically returns to the normal display screen. Make sure that the variable name being search is blinking at the bottom of the screen, then press “Search” to start the Search.



- The search will result in the matching variable being highlighted with a light blue box. To exit the Search, press CLEAR.



**Note:**

In order to continue your search, press “BACK” or “NEXT” to move to the adjacent variable in the search results.

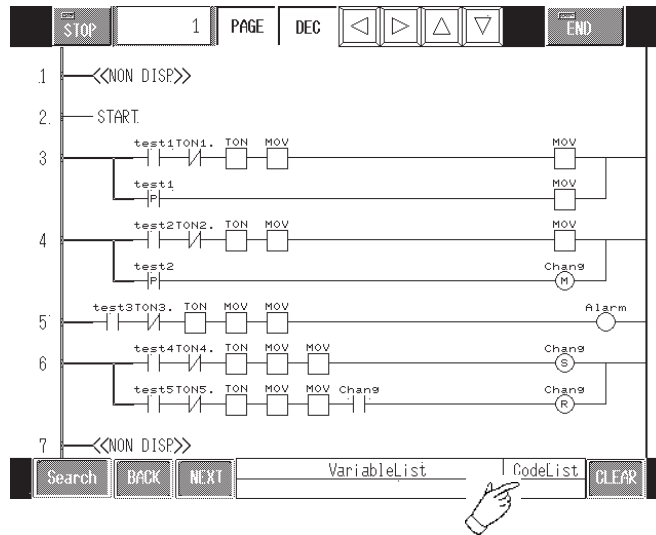


### ■ Search from Instruction

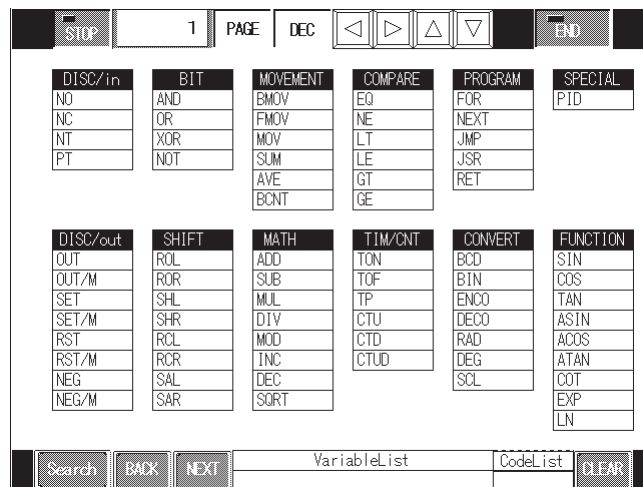
Searches the instruction specified in the Code List screen.

The Search method is as follows.

1. Press Code List on the bottom of the normal display screen, and the Instruction List screen will display.

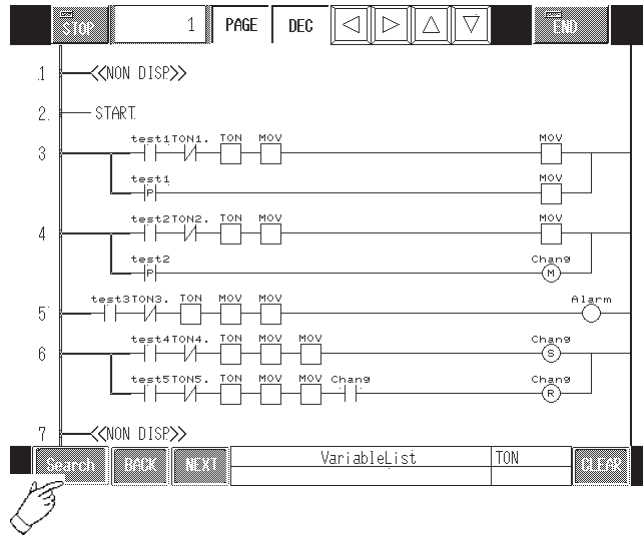


2. In the Code List screen, all instructions that can be used in the logic program are displayed. Select the variable you want to search.

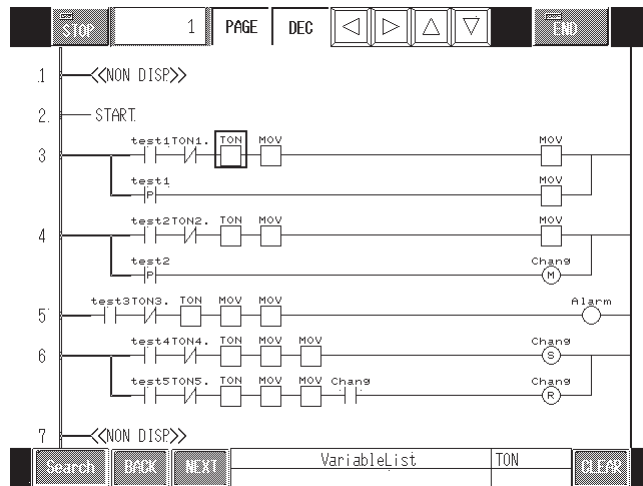


## Chapter 6 – GLC Ladder Monitor Feature

- When the instruction is selected in the code list screen, it automatically returns to the normal displayed screen. Make sure that the instruction to be searched is blinking at the bottom of the screen, then press “Search” to start the search.



- The search will result in the matching variable being highlighted with a light blue box. To exit the search, press CLEAR.



**Note:**

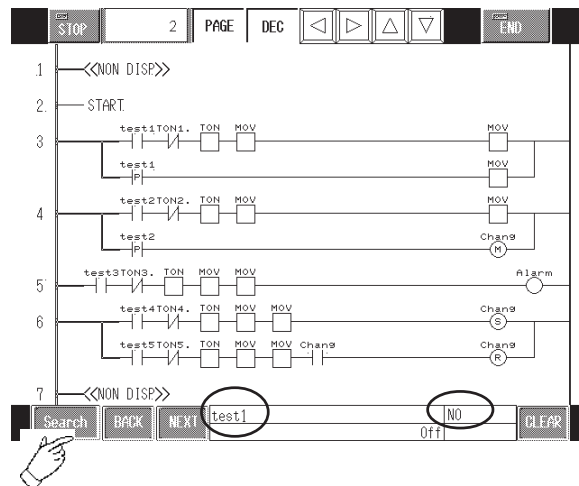
In order to continue your search, press “BACK” or “NEXT” to move to the adjacent variable in the search results.

■ Search from Variable and Instruction (AND Search)

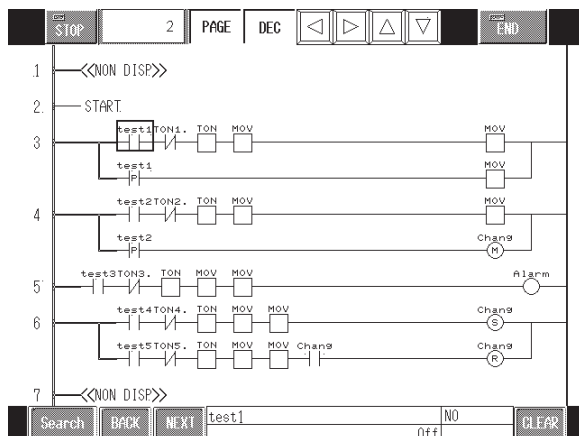
Combining a variable search and an instruction search by using an AND search narrows the search. Designate the variable and instruction you want to search for from the list screen of each.

Use the following search method.

1. Specify the search subject from either the variable or the instruction list screen.
2. When it returns to the normal screen, press the other list screen button and specify the search subject from the list screen.
3. After returning to the normal screen again, make sure that both the variable and instruction search subjects are on the bottom of the screen. Press the Search button to start the search. The following screens show examples of an AND search when “Test1” is selected from the variable list screen and “NO” is selected from the code list screen.



4. The search will result in the matching variable being highlighted with a light blue box. To exit from the search, press Clear.



**Note:** In order to continue your search, press “BACK” or “NEXT” to move to the adjacent variable or instruction in the search results.

# *Memo*

# 7 Backup

The data contents and the backup destination for GLC differ from those for LT. Be sure to check your model when you refer to this chapter.

## 7.1 Overview of the Backup Feature

You can easily save the data backed up to SRAM. You can also execute the recovery procedure using the backup data.

By this way, you can recover the SRAM data when the backup SRAM data disappeared due to power OFF status continued for a long time or when the backup SRAM data have been damaged.

The backup data contents and the backup destination for GLC differ from those for LT as described below.

### GLC Backup

To back up GLC data, use the Memory Loader Tool.

**GLC Data Backup Destination:** CF Card

**Backup Data Contents:** Variable values, OFFLINE controller setting data, Driver backup data, Data transfer protocols, Expansion programs, Screen data, backup SRAM data.

**Reference** *GP-PRO/PB III Operation Manual - 10.7 CF Memory Loader Tool*

### LT Backup

**LT Data Backup Destination:** Internal FROM

**Backup Data Contents:** Variable values, OFFLINE controller setting data, Driver backup data

This chapter describes the procedure to save LT backup SRAM controller information.

# 7.2 Backup Operation Procedure

## 7.2.1 Backup

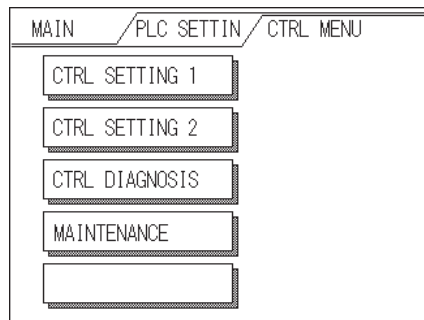
This section describes the operation procedure to execute data backup of the backup SRAM data to FROM.



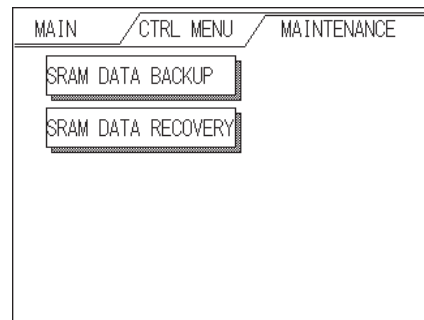
- **You cannot execute backup automatically. You cannot edit or analyze backup data either.**

### ■ Backup Operation Procedure

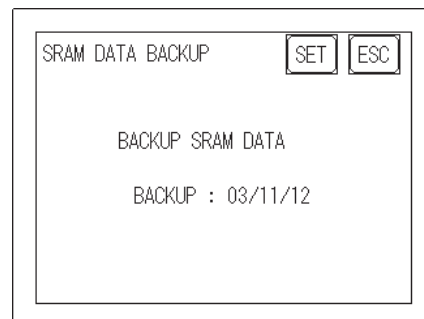
1. Select [Maintenance] of the offline Controller menu.



2. Select [SRAM Data Backup] of the Maintenance menu.



3. If you press the [Start] button, the backup procedure will start. When it completed, the "Backup Completed" message appears.



- **If power is turned off while data is being backed up, data may not be saved correctly.**



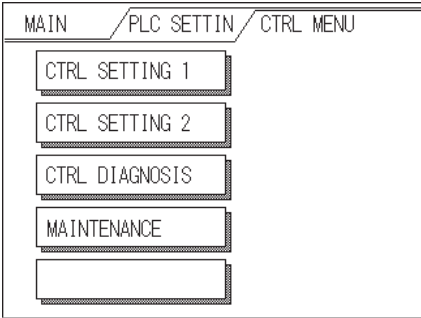
Press the [Cancel] button to go back to the previous menu.

**7.2.2 Recovery**

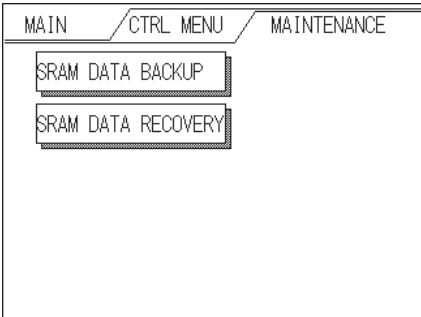
This section describes the operation procedure to recover the backup SRAM data from FROM.

**■ Recovery Operation Procedure**

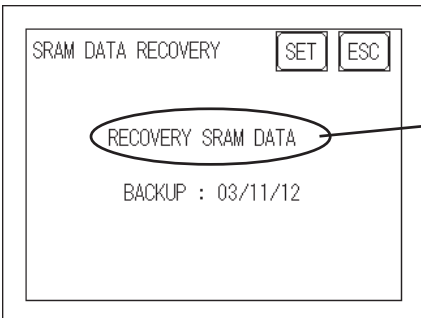
- 1. Select [Maintenance] of the offline Controller menu.



- 2. Select [SRAM Data Recovery] of the Maintenance menu.



- 3. If you press the [Start] button, the recovery procedure will start. When it completed, the "Recovery Completed" message appears.



The last backup date is displayed.



- *The recovery data is shown by the last backup date. Be sure to check it before you execute recovery.*
- *If no previously backed up data exists, or if data different from the backed up data has been sent to the LT, the message "No data." will appear.*



Press the [Cancel] button to go back to the previous menu.

*Memo*



# 8 I/O Drivers

This chapter describes I/O drivers that are required when using the built-in I/O in the GLC unit.

For the driver of LT Type H Series , refer to "LT Type H Series I/O Setup User's Manual".

## 8.1 I/O Drivers Overview

To perform external I/O, the GLC unit's I/O unit must be attached and its related I/O drivers must be installed.

**Reference** For detailed I/O Driver information, refer to the *Pro-Control Editor Operation Manual, 2.9 – "I/O Configuration."*

The following table lists the GLC-supported drivers:

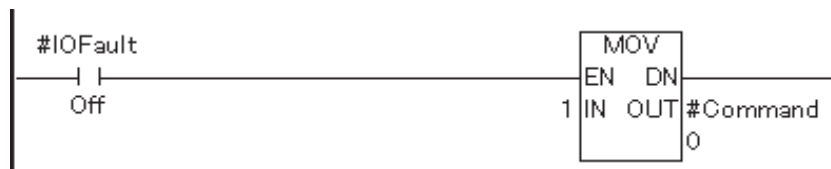
Series		Supported Drivers
GLC100 Series		DIO Driver
GLC300 Series		Flex Network Driver
GLC2000 Series		Flex Network Driver
GLC2300 Series		Flex Network Driver
GLC2400 Series		
GLC2600 Series		
LT Series	LT Type A Series	DIO Driver
	LT Type B Series	Flex Network Driver
	LT Type B+ Series	
	LT Type C Series	
	LT Type H Series	Refer to "LT Type H Series I/O Setup User's Manual".



**Note:**

- When an I/O error occurs and the controller stops, create the following logic program. There will be a delay of approximately one scan, from the time the error is detected until the time the logic program stops.

In the following example, an I/O error is detected with #IOFault, and logic execution is stopped by assigning 1 to #Command.



- When an I/O error occurs, #IOFault will turn ON. Detailed information can be checked by #IOStatus.

**Reference** See 3.2.16 – “#IOFault” and 3.2.21 – “#Command.”

## 8.2 Flex Network Interface Driver

This section describes the Flex Network driver menus in the GLC unit's OFFLINE mode.

Prior to executing any Flex Network Driver menu instructions, be sure to download the Flex Network driver from Pro-Control Editor software in your PC. Also, for GLC100 and GLC300 models, make sure that the Flex Network I/F unit is attached to the back of your GLC unit. GLC2000 Series, LT Type B/B+ and LT Type C Series have been equipped with the built-in Flex Network I/Fs.

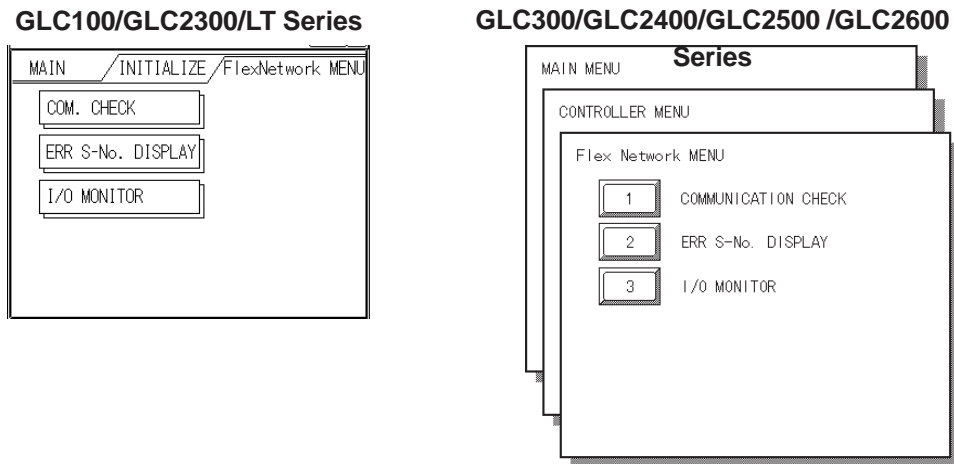
**Reference** To return to the GLC unit's OFFLINE mode, refer to the GLC unit's user manual (sold separately), LT unit's user manual (sold separately).

### 8.2.1 Flex Network Interface Unit Self-Diagnosis

This section describes how to operate the self-diagnostics of the Flex Network I/F unit.

**Reference** For details on the self-diagnostics of the GLC main unit, refer to the GLC unit's user manual (sold separately), LT unit's user manual (sold separately).

Select FLEX NETWORK DRIVER in the CONTROLLER MENU. The following FLEX NETWORK DRIVER MENU window will then appear.



**When a logic program changes from the RUN mode to either the OFFLINE or RESET mode, the GLC or the I/O signal will operate as follows, regardless of the Output Hold setting. Be sure to take this into consideration when changing to either the OFFLINE or RESET mode.**

	→		
<b>GLC Condition</b>	RUN	OFFLINE	RUN
<b>I/O Signal</b>	Output from Logic Program	No Output	Output from Logic Program
Output			
No Output			



**The RESET mode's I/O signal OFF timing is NOT fixed.**

**8.2.2 Communication Check**

The number of the Flex Network I/O units that have been connected to the Flex Network I/F units, as well as the S-Nos. that have been connected to each I/O unit, will be checked.

Via the communication check operation, the following items can be checked:

- currently connected I/O units
- currently malfunctioning I/O units (connection section)

◆ **Communication Check Procedure**

1. Press the COMMUNICATION CHECK button, and the COMMUNICATION CHECK SETUP window will appear.

Set Communication Speed to either 6 or 12. Setting the communication speed faster may cause the unit to be easily influenced by noise. Generally, set this speed to 6Mbps.

**GLC100/GLC2300/LT Series**

**GLC300/GLC2400/GLC2500/GLC2600**

2. Press the NEXT button, and the COMMUNICATION CHECK window will appear.

Press START to begin the communication check.

The currently connected I/O unit's S-No. will be displayed in reverse color.

**GLC100/GLC2300/LT Series**

**GLC300/GLC2400/GLC2500/GLC2600**

To return to the FLEX NETWORK MENU window, press the RET button.

**8.2.3 Error S-No. Display**

If the Communication Error (Error Code No. 841) occurs while a logic program is being executed, the S-Nos. of the I/O units that have been excluded from the communication circuit and malfunctioning I/O units will be checked.

**Reference** See 7.2.3 – “Flex Network I/F Unit Troubleshooting.”

**◆ Error S-No. Procedure**

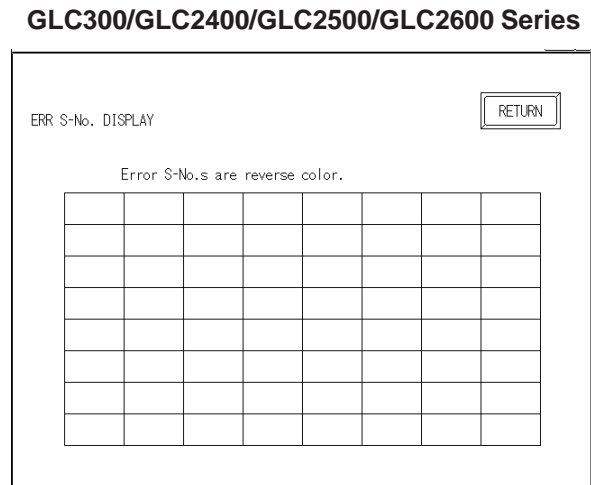
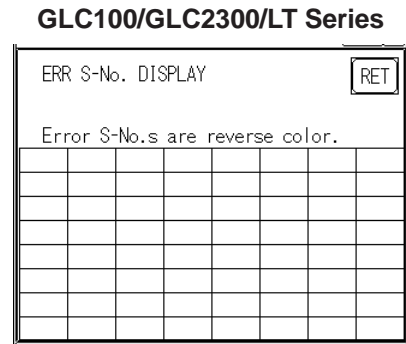
1. Touch the CONTROLLER MENU window’s FLEX NETWORK DRIVER selection.

The FLEX NETWORK DRIVER MENU will appear.

2. Press the FLEX NETWORK DRIVER MENU's ERROR S-NO. DISPLAY.

The ERROR S-NO. DISPLAY window will appear, and the error check will begin.

The currently connected I/O unit's S-Nos. will appear, and the I/O unit S-No. with the error will be shown in reverse color.



**8.2.4 I/O Monitor (I/O Connection Check)**

Check each Input and Output terminal between the GLC and I/O unit. To check inputs, monitor the I/O unit of output signals on the GLC. To check outputs, monitor the GLC unit’s output signals on the I/O unit.

**■ I/O Monitor Check Procedure**

1. Select the CONTROLLER MENU window's FLEX NETWORK DRIVER, and the FLEX NETWORK DRIVER MENU will appear.
2. Select the FLEX NETWORK DRIVER MENU window's I/O MONITOR, and the following I/O MONITOR SETUP window will appear.

**GLC100/GLC2300/LT Series**

I/O MONITOR SETUP		NEXT	ESC
TRANSFER SPEED (Mbps)	6		
S-No.	1		
MODEL CODE	FN-X16TS		
VARIABLE TYPE	DISCRETE		

**GLC300/GLC2400/GLC2500/GLC2600 Series**

I/O MONITOR SETUP				NEXT	CANCEL
TRANSFER SPEED (Mbps)	6	12			
S-No.	[ 1 ]				
MODEL CODE (FN-)	X16TS	Y08RL	Y16SK	Y16SC	XY08TS
	AD02AH	AD04AH	AD04AH	DA02AH	DA04AH
VARIABLE TYPE	DISCRETE	WORD			
1	2	3	4	5	6
7	8	9	0		

**TRANSFER SPEED**

Set TRANSFER SPEED to either 6 or 12 Mbps. Setting a faster transfer speed may result in interference caused by noise. Normally, set this speed to 6Mbps.

**S-No. (Station no.)**

Select S-No. from 1 to 63.

**MODEL CODE**

Select one of the following models: X16TS, Y08RL, Y16SK, Y16SC, XY08TS, AD04AH, DA04AH, AD02AH and DA02AH.

The FN-X32TS, FN-XY16SK, FN-XY16SC, FN-XY32SK, and FN-XY32SC models are not included in the selection. Therefore, select a substitute model, from the table below, that can check the I/O monitor’s connection.

Models Performing I/O Monitoring			FN-X32TS	FN-XY16SK FN-XY16SC	FN-XY32SKS FN-XY32SCS*1	
Substitute Models Performing I/O Monitoring			X16TS	X16TS	Y16SK or Y16SC XY08TS	
S-No.	+0	Input	0-15	0-15	–	0-7
		Output	–	–	0-15	0-7
	+1	Input	16-31	–	–	8-15
		Output	–	–	–	8-15
	+2	Input	–	–	–	16-23
		Output	–	–	–	16-23
	+3	Input	–	–	–	24-31
		Output	–	–	–	24-31

- Contact your local distributor regarding purchasing these products.

**◆ Monitoring the FN-X32TS**

Use X16TS as a substitute.

Lower 16 bits (0-15 bits) can be monitored by assigning the station number set in the I/O unit to the S-No.

Upper 16 bits (16-31 bits) can be monitored by assigning values created by adding 1 to the station number set in the I/O unit to the S-No.

◆ **Monitoring the FN-XY16SK or the FN-XY16SC**

Use X16TS as a substitute for input, and Y16SK or Y16SC as a substitute for output.

Input and Output cannot be monitored simultaneously.

◆ **Monitoring the FN-SY32SK, FN-XY32SC**

Use XY08TS as a substitute.

Input and output of bits 0–7 can be monitored by assigning the station number set in the I/O unit to the S-No.

Input and output of bits 8–15 can be monitored by assigning the values by adding 1 to the station number set in the I/O unit to the S-No.

Input and output of bits 16–23 can be monitored by assigning to the S-No. the values created by adding 2 to the station number set in the I/O unit.

Input and output of bits 24–31 can be monitored by assigning to the S-No. the values created by adding 3 to the station number set in the I/O unit.

**VARIABLE TYPE**

Select either DISCRETE or INTEGER.

\* Only the Integer setting can be used for FN-AD04AH, FN-DA04AH, FN-AD02AH and FN-DA02AH.

- Press the NEXT button, and the following I/O MONITOR window will appear. This window's items will vary depending on the VARIABLE TYPE selected.

**Reference** Please refer to the information for the corresponding I/O unit model(s).



This I/O monitor cannot be used with the high-speed counter and single-axis positioning unit.

■ **I/O MONITOR (when VARIABLE TYPE is set to BIT or INTEGER)**

◆ For FN-X16TS/FN-XY08TS/FN-Y08RL/FN-Y16SK/FN-Y16SC/FN-XY16SK/FN-XY16SC/FN-X32TS/FN-XY32SK/FN-XY32SC

**I/O MONITOR (When VARIABLE TYPE is set to BIT)**

The INPUT area terminal numbers where data has been entered will appear in reverse color. Touching an Output area terminal number will output the data and reverse that number's color.

**GLC100/GLC2300/LT Series**

I/O MONITOR		S-No.1						RET
INPUT								
0	1	2	3	4	5	6	7	
8	9	10	11	12	13	14	15	
OUTPUT								
0	1	2	3	4	5	6	7	
8	9	10	11	12	13	14	15	

**GLC300/GLC2400/GLC2500/GLC2600 Series**

I/O MONITOR		S-No.1						RETURN
INPUT								
0	1	2	3	4	5	6	7	
8	9	10	11	12	13	14	15	
OUTPUT								
0	1	2	3	4	5	6	7	
8	9	10	11	12	13	14	15	
1	2	3	4	5	6	7	8	
9	0					1	BS	
						←	→	

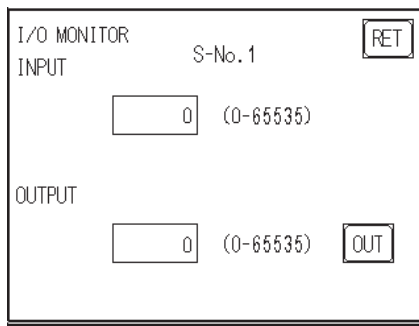
The above windows display the maximum input/output points of an I/O unit in the Flex Network system. The number of input/output points depends on each I/O unit model. Use the range of I/O points within each unit, beginning with “0.”

When using an input-only I/O unit, use only the input area of the window, and when using an output-only I/O unit, use only the output area. When using a unit with inputs and outputs, use both the input and output areas.

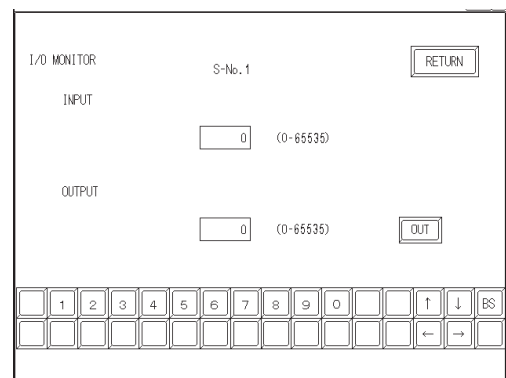
**I/O MONITOR (When VARIABLE TYPE is set to INTEGER)**

The input data, if any, will be displayed in the input field. Enter the necessary data in the output section via the ten-key keypad. When using the GLC100, GLC2300 and LT Series, touch the data entry field, and a ten-key keypad will appear. After entering data, touch the OUT key to output the data. Data will be displayed in the decimal format.

**GLC100/GLC2300/LT Series**



**GLC300/GLC2400/GLC2500/GLC2600 Series**

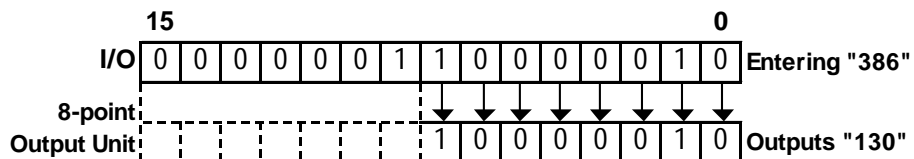


- **Enter data within the output range, according to the number of the I/O points in each I/O unit.**

I/O Points	I/O Range
8	0 to 255
16	0 to 65535



- **Data will be output to the I/O unit for the number of I/O points according to the MODEL selected on the I/O MONITOR SETUP window.**
- **If data that cannot be expressed in the 8-bit system is entered in an 8-point output I/O unit, excess data will be ignored.**



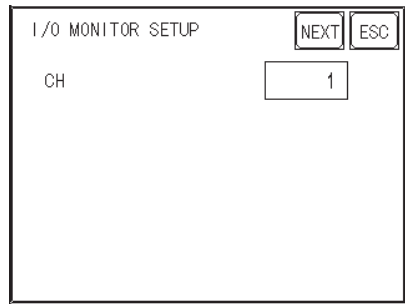
■ Analog I/O MONITOR

◆ For FN-AD04AH/FN-DA04AH/FN-AD02AH/FN-DA02AH  
I/O MONITOR (Channel Setting)

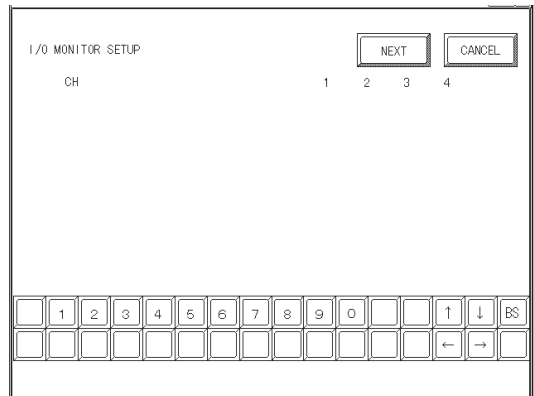
The system switches successively through the selectable settings when the channel area is pressed.

For FN-AD04AH/FN-DA04AH

GLC100/GLC2300/LT Series



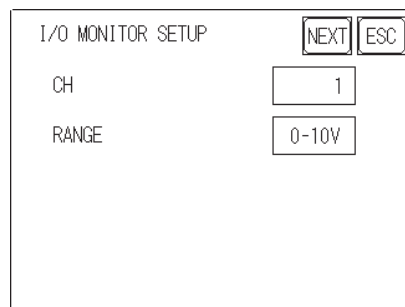
GLC300/GLC2400/GLC2500/GLC2600 Series



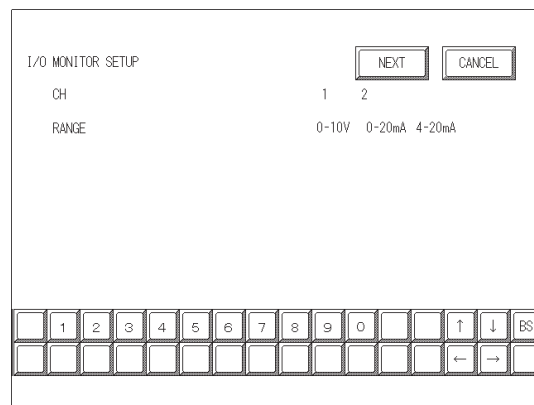
For FN-AD02AH/FN-DA02AH

When the NEXT button is pressed, the system switches to the next I/O MONITOR screen. The screen is different for FN-AD04AH and FN-DA04AH.

GLC100/GLC2300/LT Series



GLC300/GLC2400/GLC2500/GLC2600 Series



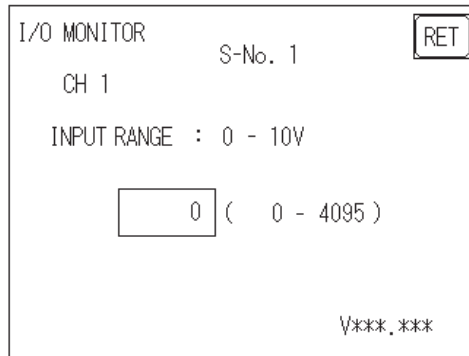


◆ For FN-AD04AH/FN-AD02AH

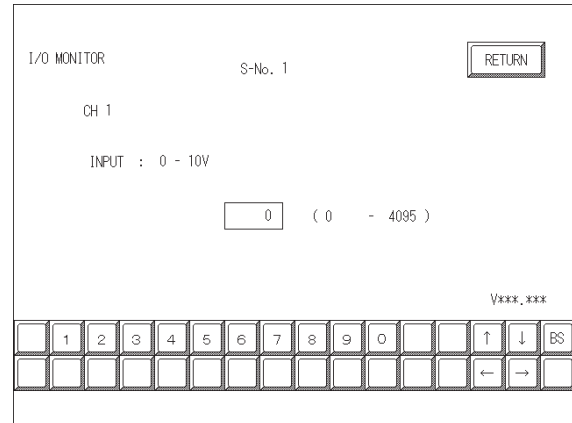
I/O MONITOR

This displays input data. The FN-AD02AH is compatible with the GLC2000 series and LT series.

GLC100/GLC2300/LT Series



GLC300/GLC2400/GLC2500/GLC2600 Series



**Note:**

- The version information at the bottom right corner of the I/O monitor is displayed only on the FN-AD04AH.

A/D Conversion Table

Input Range Setting	Input Range	
	FN-AD04AH	FN-AD02AH
0~ 5V	0~4095	-
1~5V	0~4095	-
0~10V	0~4095	0~4095
-5~5V	-2048~2047	-
-10~10V	-2048~2047	-
0~20mA	0~4095	0~4095
4~20mA	0~4095	0~4095



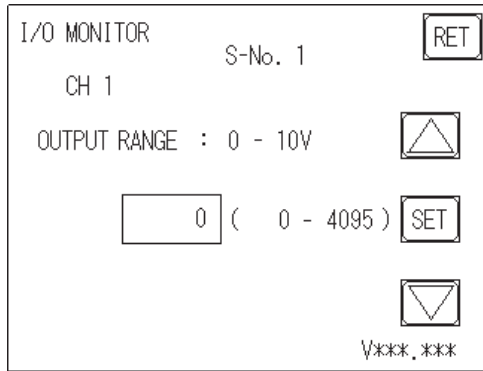
- **On the FN-AD04AH, settings other than maximum/minimum, A/D conversion sample count, and the file type operate with the set content stored on the I/O unit side. To change the settings saved on the I/O unit side, change the settings in Pro-Control Editor and download the logic program to the GLC. The logic program will then be set to RUN mode, and the settings will be enabled.**
- **On the FN-AD02AH, a filter type operates without calculating a running average.**
- **On the FN-AD04AH, the settings of the range changeover switch on the I/O unit side are read in the internal unit when the I/O unit's power cord is plugged in. To change the settings of the range changeover switch, be sure to turn the I/O unit's power OFF and then ON again.**
- **On the FN-AD04AH, the settings of the range changeover switch on the I/O unit side are read in when a logic program is switched to RUN mode. To change the settings of the range changeover switch, change the logic program to STOP mode and then to RUN mode. If the ranges do not match, the data cannot be read correctly.**

◆ For FN-DA04AH/FN-DA02AH

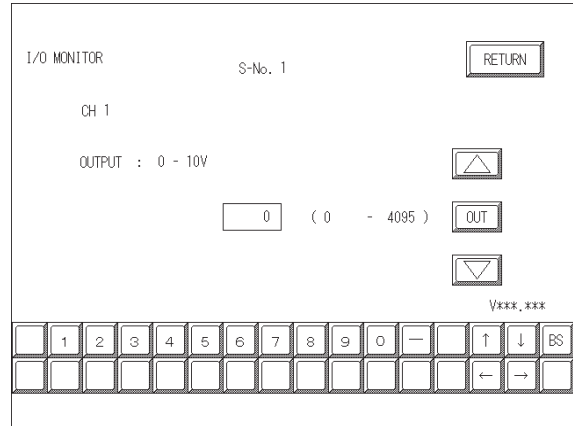
I/O MONITOR

Enter data with the keypad. Touching the screen’s data display will call up the keypad. After entering all data, push the OUT button to output the data. All data is displayed in decimal format. The FN-AD02AH is compatible with the GLC2000 series and LT series.

GLC100/GLC2300/LT Series



GLC300/GLC2400/GLC2500/GLC2600 Series



Important

- Touch the up and down arrow to increase/decrease the range value. Each time the value is changed, the new value is output to the I/O unit.
- Pressing the RET(URN) button will clear the current data, even if the output hold setting in the I/O unit is ON.
- The version information at the bottom right corner of the I/O monitor is displayed only on the FN-DA04AH.



Note:

D/A Conversion Table

Output Range Setting	Output Range	
	FN-DA04AH	FN-DA02AH
0~ 5V	0~4095	-
1~5V	0~4095	-
0~10V	0~4095	0~4095
-5~5V	-2048~2047	-
-10~10V	-2048~2047	-
0~20mA	0~4095	0~4095
4~20mA	0~4095	0~4095



Important

- On the FN-DA04AH, the settings of the range changeover switch on the I/O unit side are read in the internal unit when the I/O unit’s power is plugged in. To change the settings of the range changeover switch, be sure to turn the I/O unit’s power OFF and then ON again.
- On the FN-DA04AH, the settings of the range changeover switch on the I/O unit side are read in when a logic program is switched to RUN mode. To change the settings of the range changeover switch, change the logic program to STOP mode and then to RUN mode. If the ranges do not match, the data cannot be written correctly.

## 8.2.5 Flex Network Troubleshooting

The following is a description of possible problems that may occur when using the Flex Network I/F unit, and their solutions.

### ■ Flex Network I/F unit I/O Errors

**Reference** *For a detailed explanation of Flex Network I/F unit I/O errors, please refer to the Flex Network User Manual (sold separately).*

### ■ Error Codes

I/O errors include those that occur during writing and reading. When one of these errors occurs, the controller writes an error code to #IOStatus.

### ◆ Setting Errors

Error Code	Description	Solution
501	Internal variable error mapped to I/O terminal.	Reset the variable used.
502	External variable error mapped to I/O terminal.	
503	Output variable error mapped to I/O terminal.	
504	Discrete variable error mapped to analog terminal.	
505	Integer variable error mapped to discrete terminal.	
506	Variable type not supported by driver.	Correct the variable type.
507	Variable is not mapped to terminal.	Map the variable to all terminals.
801	Terminal numbers are duplicated.	Two or more terminals are using the same terminal number, possibly causing transfer failure. Download the project file again.
802	Multiple S-No. exist.	Two or more areas are using the same area number, possibly causing transfer failure. Download the project file again.
803	S-No. is outside of accepted range.	Two or more terminals are using the same terminal number, possibly causing transfer failure. Download the project file again.
804	S-No. range overlap at the analog unit.	Two or more I/O units are using the same S-No. The analog unit has S-Nos. for four stations. Reset so there is no S-No. overlap.
805	S-No. range overlap at the high-speed counter unit.	Two or more I/O units are using the same S-No. The high-speed counter unit has S-Nos. for eight stations. Reset so there is no S-No. overlap.
806	S-No. range overlap at the single-axis positioning unit.	Two or more I/O units are using the same S-No. The positioning unit has S-Nos. for four stations. Reset so there is no S-No. overlap.

◆ Initialization Errors

Error Code	Problem	Solution
821	There is no Flex Network I/F unit attached.	The ID Number loaded from the GLC unit's built-in Flex Network I/F is invalid. The Flex Network I/F unit may be broken. Write down the error code and contact your local Pro-face distributor.
822	Initialization Error. Initialization failed to synchronize the Flex Network I/F unit and the Flex Network driver.	The Flex Network I/F unit may be broken. Write down the error code and contact your local Pro-face distributor.
823	Analog unit setting error (4 Channel)	There may be a break in the communication cable, the I/O unit is not
824	Analog unit setting error (2 Channel)	

◆ Runtime Errors

Error Code	Description	Solution
841	There is an I/O unit error (loose connector, malfunction, etc.)	Check all related wiring. <b>Reference</b> Refer to the <i>Flex Network User Manual</i> (sold separately).
842	Disconnected output signal line of sensor for input to the analog unit (A/D conversion unit) (4 Channel/2 Channel)	This is likely due to disconnection in the output signal line. Check the output signal line of the sensor. The error code will remain until the controller is reset.
843	Error in the high-speed counter unit	The High-Speed Counter unit detected an error. <b>Reference</b> Refer to the <i>Flex Network High-Speed Counter User Manual</i> (sold separately).
844	Initial error in the high-speed counter unit	Check to see if communication line is disconnected, power is not supplied to the I/O unit, or the I/O unit is malfunctioning.
845	Communication error with the high-speed counter unit	Check to see if communication line is disconnected, power is not supplied to the I/O unit, or the I/O unit is malfunctioning.
846	Error in the single-axis positioning unit	The positioning unit detected an error. <b>Reference</b> Refer to the <i>Flex Network Single-Axis Positioning Unit User Manual</i> (sold separately).
847	Communication error with the single-axis positioning unit.	Check to see if communication line is disconnected, power is not supplied to the I/O unit, or the I/O unit is malfunctioning.
848	Communication error with the Analog unit. (2 Channel)	

◆ Internal Errors

Error Code	Contents	Solution
850	Driver Error. A major system error has occurred.	Reset the GLC. If an error code still appears, try to identify if the error is due to the GLC itself, or to a related/connected device.
...		<b>Reference</b> Write down the error code and refer to your <i>GLC User Manual</i> .
859		Contact your local Pro-face distributor.

## 8.3 DIO Driver

This section explains the GLC OFFLINE mode’s DIO menu. Be sure the DIO unit is securely attached prior to using any of the DIO unit’s features.

**Reference** For instructions on how to move to the OFFLINE mode screen, refer to the *GLC Series User Manual* (sold separately), *LT Series User Manual* (sold separately).

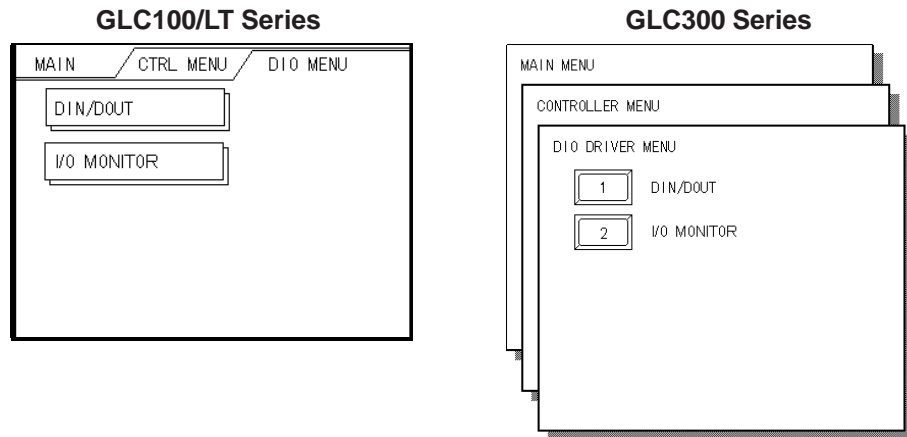
### 8.3.1 DIO Unit Self-Diagnosis

This section explains how to use the DIO unit’s Self-Diagnosis feature.

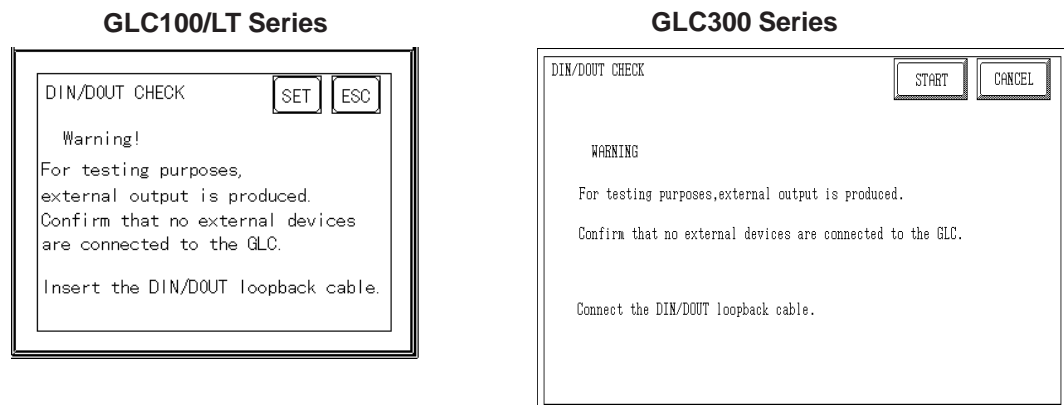
**Reference** For detailed information, refer to the *GLC Series User Manual* (sold separately), *LT Series User Manual* (sold separately).

<When DIO Driver has been Selected>

Touch the OFFLINE screen’s Controller Menu to open the DIO Menu area.



Touch the DIN/DOUT key to open the following screen.



Touch either the SET or START key to start the self-diagnosis.

## Chapter 8 – I/O Drivers

This check sends an output signal from the output unit to the input unit. Therefore, prior to performing this check, be sure to attach the DIN/DOUT loopback cable.



**When switching to the OFFLINE mode or when resetting from a logic program's RUN state, the I/O signal may turn to OFF. Be aware of the possibility that the I/O signal will turn OFF.**

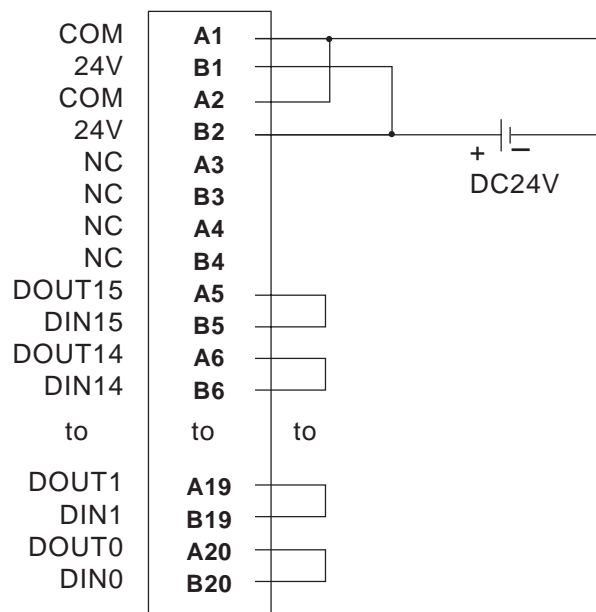
GLC Condition	→		
	RUN	OFFLINE	RUN
Analog Output	Output from Logic Program	No Analog Output	Output from Logic Program
I/O Signal			
No Analog Output			



**The RESET mode's I/O signal OFF timing is NOT fixed.**

### ■ Loopback Cable Creation

Use the following diagram when creating the DIN/DOUT loopback cable.



### ◆ Recommended Products

Connection Type	Manufacturer	Model Number
Soldered Type	Fujitsu	FCN-361J040-AU (Connector) FCN-360C040-B (Cover)
	Digital	FN-IFCN01 <sup>*1</sup> (Connector/Cover)
Crimped Type	Fujitsu	FCN-363J040 FCN-363J-AU/S FCN-360C0404-B
Terminal Block Unit Type	Mitsubishi	A6TBX36 (Terminal Block) AC**TB (Cable) (* = cable length)
	Yokogawa	TA40-ON

1. This product is similar to Fujitsu's product.

**8.3.2 I/O Monitor (I/O Connection Check)**

On the DIO driver menu touch I/O Monitor to call up the following screens.

◆ **When I/O Monitor has been Selected**

**GLC100/LT Series**

**GLC300 Series**

Select the Module No. — either 0 or 1. (The “0” unit is attached directly to the GLC, and the “1” unit is attached to the back of the “0” unit.

Select the Input Variable Type — either Discrete or Word.

Select the Output Variable Type — either Discrete or Word.

For example, if you enter “0” as the Module No., “Discrete” as the Input Variable Type, and “Word” as the Output Variable Type, then touch the RUN button in the screen’s top-right corner, the I/O Monitor screen will appear.

**GLC100/LT Series**

**GLC300 Series**

When the Input Variable Type is Discrete, the input terminal (S-No.) will appear in reverse color. When the Output Variable Type is WORD, use the ten-key keypad to enter the data. When using the GLC100 and LT Series units, touch the data entry field and the ten-key keypad will appear. After entering data, touch the OUT key to output the data. Data will be displayed in decimal format.

**8.3.3 DIO Troubleshooting**

This area explains how to solve possible DIO unit problems.

◆ **DIO Unit Input Errors**

<b>Error Type</b>	<b>Possible Cause</b>	<b>Solution</b>
Input monitor lamp is ON, but no input can be performed.	DIO Unit is defective.	Contact your local Pro-face distributor.
	[Enable I/O] box is not selected.	Set the [Enable I/O].
	Program is incorrect.	Correct program
Input monitor lamp is OFF and no input can be performed.	DIO Unit is defective	Contact your local Pro-face distributor.
	Input common line is incorrectly wired.	Common line wiring check. Common line breakage check. Common terminal looseness check.
	External input power is incorrect.	Provide the correct voltage.
	Connector is not securely attached.	Attach the connector securely.
All input lines do not turn OFF	DIO Unit is defective.	Contact your local Pro-face distributor.
Designated Input lines do not turn ON.	DIO Unit is defective.	Contact your local Pro-face distributor.
	Program is incorrect	Correct the program.
	Input wiring is incorrect.	Check common line wiring. Check common line breakage. Check common terminal for looseness.
	External unit is defective.	Replace the unit.
	Input ON period is too short.	Lengthen the Input ON time.
Designated Input lines do not turn OFF.	DIO Unit is defective.	Contact your local Pro-face distributor.
	Program is incorrect.	Correct the program.
Input area randomly turns ON or OFF.	External Input voltage is incorrect.	Provide the correct voltage.
	Input terminal screws are loose.	Tighten the terminal screws.
	Program is incorrect.	Correct the program.
	Connector is not securely attached.	Attach the connector securely.
	Noise is causing unit mis-operation.	Reduce the noise level. Attach a surge killer. Use a shielded cable.



## ◆ DIO Unit Output Errors

Error Type	Possible Cause	Solution
Output monitor lamp is ON, but no output can be performed	DIO unit is defective.	Contact your local Pro-face distributor.
	Output common line is incorrectly wired.	Output line wiring check. Output line breakage check. Output terminal looseness check.
	Load current is incorrect.	Provide the correct current.
	Connector is not securely attached.	Attach the connector securely.
Output monitor lamp is OFF and no output can be performed	DIO unit is defective.	Contact your local Pro-face distributor.
	Program is incorrect Output area is completely OFF.	Correct program.
	[Enable I/O] box is not selected.	Set the [Enable I/O].
Output lines do not turn OFF	DIO unit is defective.	Contact your local Pro-face distributor.
Designated output lines do not turn ON	DIO unit is defective.	Contact your local Pro-face distributor.
	Output wiring is incorrect.	Check output line wiring. Check output line breakage. Check output terminal for looseness.
	External unit is defective.	Replace unit.
Designated output lines do not go OFF	DIO unit is defective.	Contact your local Pro-face distributor.
	Current leakage, residual voltage causes incorrect recurrence.	Change design of external device. I.e. Attach dummy resistor, etc.
Output area randomly turns ON/OFF	Load voltage is incorrect.	Correct voltage load.
	Output terminal screws are loose.	Tighten the terminal screws.
	Program is incorrect. Output commands are overlapping.	Correct the program.
	Connector is not securely attached.	Attach the connector securely.
	Noise is causing unit operation error.	Reduce the noise level. Attach a surge killer. Use a shielded cable.

■ **Error Codes**

I/O errors are Read/Write errors. When I/O errors occur, the controller writes an error code to the #IOStatus variable. The logic program continues to operate. The following table describes possible error causes and solutions for when the DIO unit is attached to the GLC.

◆ **Setting Errors**

Error Code	Description	Solution
501	Internal variable error allocated to I/O terminal.	Reset the variable used.
502	External variable error allocated to I/O terminal.	
503	Output variable error allocated to I/O terminal.	
504	Discrete variable error allocated to analog terminal.	
505	Integer variable error allocated to discrete terminal.	
506	Variable type not supported by driver.	Correct the variable type.
801	Terminal numbers are duplicated.	Two or more terminals are using the same terminal number, possibly causing transfer failure. Download the project file again.
802	Multiple modules are used.	Two DIO units are using the same module number. Reset these numbers so they do not overlap.
803	Module number has exceeded 1.	Set a module number from 0 to 1.
804	Unit number starts from 1.	Set the DIO unit nearest the GLC unit's rear face to "0".

◆ **Runtime Errors**

Error Code	Description	Solution
840	Module "0" read-out data is incorrect. After two successive read attempts, the GLC has detected that the value of DIO Module "0" is incorrect.	Increase the time of the Input signal's ON period. The error code will remain until the controller is reset.
841	Module "1" read-out data is incorrect. After two successive read attempts, the GLC has detected that the value of the DIO Module "1" is incorrect.	Increase the time of the Input signal's ON period. The error code will remain until the controller is reset.
842	Module "0" output data is incorrect. Incorrect output data was detected by an internal loopback check from DIO Module "0".	Ensure that there are no noise-related or other adverse effects. The error code will remain until the controller is reset.
843	Module "1" output data is incorrect. Incorrect output data was detected by an internal loopback check from DIO Module "1".	Ensure that there are no noise-related or other adverse effects. The error code will remain until the controller is reset.

◆ Initialization Errors

Error Code	Description	Solution
821	The number of DIO units registered in the WLL file and the actual number of DIO units connected are different.	Correct the number of connected DIO units.
822	Module "0" does not exist. DIO Module "0" does not exist.	Confirm that the DIO unit is securely connected to the GLC and correct the DIO driver settings.
823	Module "1" does not exist. DIO Module "1" does not exist.	Confirm that the DIO unit is securely connected to the GLC and correct the DIO driver settings.

◆ Internal Errors


Error Code	Contents	Solution
850	Driver Error	Record the Error Number and contact your local Pro-face distributor.
...	A major system error has occurred.	
864		

# *Memo*

# 9 Error Messages

## 9.1 Error Message List

This chapter describes error messages that can appear on the GLC unit. The error messages described here are those related to the Pro-Control program only.

**Reference**  For further information concerning GLC error messages, refer to the *GLC Series User Manual* (sold separately).

Error Message	Cause	Solution
"Invalid ladder file"	The GLC's logic program file is not downloaded, or the file is damaged.	Download another copy of the project file from the LogiTouch Editor.
"Fatal Error: Drive check Failed"	The GLC's current I/O driver is incorrect.	Check that the I/O driver set in the logic program file and the driver installed in the GLC are the same.
"Global Data Area Too Small"	The downloaded file's data may be damaged.	Download the project file again. If this does not fix the problem, contact your local Pro-face dealer.
"Can't Set Priority"	The GLC's system file is incorrect. The file may have been damaged during downloading.	Download the project file again.
"Exception nnn:[mmm:ooo]"	A fatal error has occurred in the ladder logic program.	Write down the error message details and consult your local Pro-face dealer.
"SRAM checksum error"	The project file stored in SRAM is damaged. (GLC2000 Series only)	Download the project file again from Pro-Control Editor.
"SRAM data broken"	The battery for SRAM back-up may have run out. This is a warning message. (GLC2000 Series only)	Execute from the project file in FEPRM. Using online edit, check that no changes have been made in the logic program.
"Watchdog Error"	The Constant Scan Time is longer than the Watchdog time.	Reset the Watchdog time so that it is longer than the Constant Scan Time. If doing so exceeds the Watchdog Timer's limit, then the Constant Scan Time (program) should be changed.

## Chapter 9 – Errors

Error Message	Cause	Solution
"Bad Var: xxx"	Unable to find variable "XXX". Either the logic program file has not been downloaded, or a variable that does not exist in the logic program file on the screen, is used.	Save the project file, and download screen data again.
"Bad Array: xxx"	The number of elements used in the screen file's array variables and those used in the logic program file's array variables are different.	Download the project file again.
"Bad Type xxx"	The Logic program variable "XXX"s type is different from the screen's variable type.	Download the project file again.
"Unknown register type"	This variable type does not exist.	Download the project file again.
"Register is missing"	Cannot find variable used for Writing.	
"S100 file index is out of range"	Cannot find variable used for Reading.	
"Too many entries in the S100 file"	Too many variables are being used. Limit is 2048.	
"S100 file is missing"	Cannot find S100 (variable storage file).	
"Over Compile count MAX"	Too many Parts are being used.	Reduce the number of Parts and then download the project to the GLC again.
"Exception 65532 [xxxx : xxx] " "Exception 65533 [xxxx : xxx] " "Exception 65534 [xxxx : xxx] " "Exception 65535 [xxxx : xxx] "	GLC heap memory is insufficient. Memory for storing programs and variables is sufficient, however logic program memory is insufficient.	Setup the GLC unit again with the Pro-Control Editor after reducing the logic program size, or the number of variables and labels. Also reduce the number of array variable elements, or shorten the name of variables and labels.
"No Backup logic program in EEPROM"	The logic program was edited, and data downloaded, during monitoring mode. Therefore, the data was saved to the SRAM, and not to the default FEPROM.	Switch to offline mode, and follow the procedure below to copy data from the SRAM to the FEPROM. [Main Menu] -> [INITIALIZE] -> [PLC SETTING] -> [CTRL SETTING] -> [COPY TO FEPROM]

## 9.2 Error Codes

The following table lists #FaultCode errors that are written in when errors occur.

Error Code	Level	Cause
0	Normal	No errors
1	Minor	The calculated result, or the conversion of a Real variable to an Integer variable has resulted in an overflow.
2	Major	A reference was used for an area outside the array's range.
3	Major	A reference was used for a bit outside the Integer's (32 bit) range
4	Major	The stack has overflowed.
5	Major	Incorrect command code is being used.
6	—	Reserved for System.
7	Major	The Scan time is now longer than the Watchdog time.
8	—	Reserved for System.
9	Major	Software Error. Depending on the type of problem, the system may need to be restarted.
10	—	Reserved for System.
11	—	Reserved for System.
12	Minor	BCD/BIN Conversion Error
13	Minor	ENCO/DECO Error*1
14	—	Reserved for System.
15	Minor	The logic program of the backup memory (SRAM) is damaged. The logic program of FEPRM will be executed.*1

1. This error occurs only with GLC2000 Series units.



- **[Major Faults]**

*When a major fault occurs, the controller immediately stops executing the logic program. The main unit's LED turns red, and the buzzer sounds continuously.*

- **[Minor Faults]**

*Logic program execution can continue when minor faults occur.*

## 9.3 Program Errors

The following table lists Pro-Control Editor’s program operation errors.

Error Type	Possible Problem	Solution
Control Memory power is cut.	Battery Alarm	Exchange Unit
Keep Area data is not preserved.	Memory Alarm	Exchange Unit
Program is not operating normally.	Program transfer mistake.	Use GP-PRO/PB III to download the project file again. <b>Reference</b> Refer to the <i>Pro-Control Editor Operation Manual, 5.2 – "Transferring Preparation Screens to the GLC."</i>
Data is output from I/O even in STOP mode.	When output data performs RUN/STOP switchover, I/O output hold is enabled.	Disable this feature. <b>Reference</b> Refer to <i>Pro-Control Editor's Online Help.</i>
Soon after entering RUN mode unit changes to STOP mode.	A Command Execution Alarm has occurred. Or, a major fault has occurred.	Check the contents of System variable #FaultCode data and modify the program. <b>Reference</b> Refer to the <i>Pro-Control Editor Operation Manual, 3.4 – "Viewing System Variables."</i>  Check if the System variable #Command has been written, and modify the program. <b>Reference</b> See <i>3.2.14 – "#FaultCode,"</i> and <i>3.2.21 – "#Command."</i>
Pro-Control Editor cannot enter configuration settings.	The data transfer cable used to send data from GP-PRO/PB III to the GLC unit may be loose or disconnected. Also, the PC or GLC unit's power may have dropped, causing excessive noise and possible destruction of the contents.	Check whether the data transfer cable is unplugged or if there is noise influence. If the problem continues, please contact your local Pro-face distributor for assistance.
The logic program file cannot be downloaded from Pro-Control Editor.		
The project (.prw) file cannot be downloaded from GP-PRO/PB III.		
Data cannot write to or read from the I/O.	Enable I/O*1 is not selected.	Set the enable I/O.

1. *Enable I/O is used to input and output data between the GLC and I/O units. After downloading a logic program to the GLC unit, the external I/O devices cannot be performed in RUN mode. (As a safety precaution, the I/O is not enabled in the default setting.) It is necessary to set up the Enable I/O beforehand to write and read data to the I/O.*

**Reference** For information on how to set up, refer to the *Pro-Control Editor Operation Manual, 3.1 – "Controller Configuration"* and *3.2 – "Starting and Stopping the Controller."*



# Appendices

## Appendix 1 Instruction List

For details of each instruction, refer to "Chapter 4 Instructions".

Instruction List			Supported Models					
Category	Instruction	Process	GLC100, GLC300	LT	GLC2400/GLC2600 "Rev.* - None, 1"	GLC2300, GLC2500, GLC2400/GLC2600 "Rev.* - Above2"		
Discrete Instructions	NO	Normally Open	○	○	○	○		
	NC	Normally Closed						
	OUT/M	Output Coil/Retention Coil						
	NEG/NM	Negated Coil/ Negated Retention Coil						
	SET/SM	Latch Coil/Latch Retention Coil						
	RST/RM	Unlatch Coil/ Unlatch Retention Coil						
	PT	Positive Transition						
	NT	Negative Transition						
Arithmetic Operation Instructions	AND	Logical Multiply	○	○	○	○		
	OR	Logical Add						
	XOR	Exclusive Logical Add						
	NOT	Bit Negation						
Movement Instructions	MOV	Transfer	○	○	○	○		
	BMOV	Block Transfer						
	FMOV	Fill Transfer						
	SUM	Sum						
	AVE	Average					×	
	BCNT	Bit Count						
Shift Instructions	ROL	Rotate Left	○	○	○	○		
	ROR	Rotate Right						
	SHL	Shift Left						
	SHR	Shift Right						
	RCL	Left Rotation with Carry					×	×
	RCR	Right Rotation with Carry						
	SAL	Arithmetic Shift Left						
	SAR	Arithmetic Shift Right						
Mathematical Instructions	ADD	Add	○	○	○	○		
	SUB	Subtract						
	MUL	Multiply						
	DIV	Divide						
	MOD	Residual Processing						
	INC	Increment						
	DEC	Decrement						
	SQRT	Square Root					×	×

1. For how to distinguish "Revisions", refer to "For GLC2400/GLC2600 Users".

## Appendices

Instruction List			Supported Models					
Category	Instruction	Process	GLC100, GLC300	LT	GLC2400/GLC2600 "Rev.* - None, 1"	GLC2300, GLC2500, GLC2400/GLC2600 "Rev.* - Above2"		
Comparison Instructions	EQ	Equal To (=)	○	○	○	○		
	GT	Greater Than (>)						
	LT	Less Than (<)						
	GE	Greater Than or Equal To (>=)						
	LE	Less Than or Equal To (<=)						
	NE	Not Equal (< >)						
Timer and Counter Instructions	TON	ON Delay Timer	○	○	○	○		
	TOF	OFF Delay Timer						
	TP	Timer Pulse						
	CTU	UP Counter						
	CTD	DOWN Counter						
	CTUD	UP/DOWN Counter						
Convert Instructions	BCD	BCD Conversion	○	○	○	○		
	BIN	Binary Conversion						
	ENCO	Encode	×		○		○	
	DECO	Decode						
	RAD	Radian conversion (Degrees→Radians)			×		○	○
	DEC	Degree Conversion (Radians→Degrees)						
	SCL	Scale Conversion						
Program Control Instructions	JMP	Jump	○	○	○	○		
	JSR	Jump to Subroutine						
	RET	Return from Subroutine						
	FOR,NEXT	Repeat						
Special Instructions	PID	PID Calculation	×	○	○	○		
Function Control Instructions	SIN	sine function	×	○	○	○		
	COS	cosine function						
	TAN	tangent function						
	ASIN	Arc sine			×		○	
	ACOS	Arc cosine						
	ATAN	Arc tangent						
	COT	Cotangent						
	EXP	Exponent						
	LN	Natural logarithm						

1. For how to distinguish "Revisions", refer to "For GLC2400/GLC2600 Users".

## Appendix 2 System Variable List

For details of each system variable, refer to "Chapter 3 System Variables".

Category	System Variable	Process
ScanTime	#AvgLogicTime	Displays the average Logic Time (Read, Perform, Write) once every 64 scans. (Unit: ms)
	#AvgScanTime	Displays the latest Logic Time (Read, Perform, Write, Display processing) once every 64 scans. (Unit: ms)
	#LogicTime	Displays the latest Logic Scan Time (Read, Perform, Write). (Unit: ms)
	#PercentAlloc	Calculates the Percent Scan's percentage. (Unit: % )
	#ScanCount	Excluding the current scan, counts the number of scans performed.
	#ScanTime	Displays the latest Logic Scan Time (Read, Perform, Write, Display processing). (Unit: ms)
	#TargetScan	Sets the Constant Scan Time. (Unit: ms)
	#WatchdogTime	Displays the Watchdog Timer' value set either in the editor or in offline mode (Unit: ms)
Status	#DisableAutoStart	Defines the mode entered when the GLC starts up.
	#ForceCount	Counts the number of times a variable is forced ON or OFF.
	#IOStatus	Displays the I/O Driver's condition.
	#Platform	Indicates the controller's platform.
	#Status	Indicates controller's current status.
	#Version	Displays the controller's version data.
	#Fault	Used to stop the performance of an Error Handler subroutine.
	#FaultCode	Displays the latest error code
	#FaultOnMinor	Controls the completion of the logic performed when a minor error occurs.
	#FaultRung	Displays the rung where the error occurred.
	#IOFault	Turns ON when an error occurs.
	#Overflow	Turns ON when an overflow occurs due to mathematical commands or Real-to-Integer variable conversion.
Command	#Command	Changes the controller's mode.
	#Screen	Switches GLC screens by assigning screen numbers. (BIN/BCD)
Time	#Clock100ms	Create 0.1s clock.
	#Year	Stores Year data as BCD two digits.
	#Month	Stores Month data as BCD two digits.
	#Day	Stores Day data as BCD two digits.
	#Time	Stores Time data as BCD two digits.
	#WeekDay	Stores Day data as an integer value between 0 and 6
Others	#LadderMonitor	Starts and runs the GLC Ladder Monitor Feature.
	#RungNo	Sets the starting rung number to be displayed by the GLC Ladder Monitor Feature.

# *Memo*

# Index

## A

Arithmetic Operation Instructions 4-2  
Arrays 2-3, 2-7

## B

Base Screen Number 6-3  
BCD/BIN Conversion 3-12, 4-59  
Bit Operation Instructions 4-1  
Bit Positions 4-31, 4-33

## C

Channel Setting (I/O Monitor) 8-9  
Communication, Flex Network I/F 8-3  
Constant Scan Mode 1-1, 1-5, 1-7  
Controller 1-1, 3-8, 5-1  
Controller States 1-3  
Copyrights 1  
Counter Instructions 4-4  
Counter Variables 2-5

## D

Damages or Third-Party Claims 1  
Data Files 9  
Data Sharing 5-7  
Data Watch List 3-6  
Device Addresses 2-1  
Device/PLC Connection Manual 9  
Digital Electronics Corporation 1, 5-9  
DIN/DOOUT Loopback Cable 8-15  
DIO Unit  
    DIO Driver 8-13  
    Error Codes 8-19  
    I/O Errors 817  
    Self-Diagnosis 8-13  
    Troubleshooting 8-17  
Discrete Arrays 2-7  
Discrete Variables 2-3  
Disk Media Usage Precautions 10  
Display Features 1-1, 5-1

## E

Enable I/O 9-4  
Error Codes 9-3  
Error S-Nos. 8-5

## Errors

BCD/BIN conversion 3-12, 4-59  
DIO Unit 819  
External Device Communication 5-9  
Fault Errors 9-3  
Flex Network I/O Units 8-11, 8-13  
I/O Unit 8-1  
Initialization 8-19  
Internal 8-19  
Messages 9-1  
Operation 9-4  
Pro-Control Editor 9-4  
Program Operation 9-4  
Runtime 8-19  
Setting 8-19

## External Devices,

LS Area Refresh Cautions and 5-9

## F

### Faults

Fault Errors 9-3  
Fault Flags 3-8  
Mathematical 3-11  
Minor 3-11, 3-13  
Status Codes 3-9  
Status History 3-8

### Flex Network Interface

Communication 8-3  
Driver 8-3  
I/O Unit Settings 8-3  
Self-Diagnosis 8-3  
Troubleshooting 8-11

Floating-Point Instruction 4-39, 4-41, 4-42

Foreign Regulations 1

## G

General Information Symbols and Terms 12  
GLC Features 1-1  
GLC Scan Time 1-5  
GLC Variable Monitor Feature,  
Ladder Monitor 6-8

## Index

### I

- I/O Connection Check 8–5, 8–15
- I/O Drivers 8–1
- I/O Points, maximum 8–7
- I/O Unit
  - and Error S-No. 8–5
  - Flex Network Communication Check 8–3
  - Flex Network Settings 8–3
  - GLC I/O Drivers 8–1
  - I/O Monitor Connection Check 8–5
- Indirect Arrays 2–9
- Infinite Loops 4–63
- Input Terminal (S-No.) 8–15
- Input-Only I/O Unit 8–7
- Instruction Enlarge Feature, Ladder Monitor 6–7
- Instruction Search Feature, Ladder Monitor 6–13
- Instructions
  - Arithmetic operation 4–2
  - Bit operation 4–1
  - Convert 4–5
  - Counter 4–4
  - Floating-point 4–39, 4–41, 4–42
  - Mathematical 4–3, 4–4
  - Movement 4–2
  - Overflow 3–11
  - Program Control 4–5, 4–6
  - Timer 4–4
- Integer Arrays 2–7, 2–9
- Integer Variables 2–3
- Integers 2–7
- Intellectual Properties 1
- Internal Clock 3–15

### L

- Ladder Monitor 3–19, 6–1
- Ladder Monitor Features 6–5
- Latch Fault Flag 3–8
- Liability 1
- Logic Program Mode Change 8–2
- Loopback Cable 8–15
- LS Area Refresh 5–1, 5–7, 5–9

### N

- Negative Transition Contact 4–14
- Nests 4–65
- Normal Display Feature, Ladder Monitor 6–5

### O

- OFFLINE Mode 1–1, 8–3
- Online Edit 6–4
- Online Monitor Feature, Ladder Monitor 6–5
- Operating Status, Controller 3–8
- Operation Manual 9
- Operation Mode 1–1
- Output-Only I/O Unit 8–7
- Overflows 3–11

### P

- Parts List 9
- Percent Scan Mode 1–1, 1–5, 1–7
- Precautions and Warnings 10
- Pro-Control Editor
  - Compatible Products 8
  - Error messages 9–1
  - Program errors 9–4
- Pro-face 1

### R

- Range Changeover Switch 8–9, 8–11
- Read Area and Variable LS Size 5–8
- ReadMe File 1, 9
- Real Arrays 2–9
- Real Values 4–45, 4–46, 4–49
- Real Variables 2–5
- Real-to-Integer Conversion 3–11, 3–12, 4–19
- Registered Trademarks 9
- RESET Mode 8–2
- Retentive / Non-retentive Variables 4–1
- Rollover 3–4
- RUN Mode States 1–5
- Rung Jump Feature, Ladder Monitor 6–7

### S

- S-No. (Input Terminal) 8–15
- Safety Symbols and Terms 12, 11
- Scan Time Adjustment, RUN Mode 1–5
- Screen Layout Sheets 9
- Scroll Feature, Ladder Monitor 6–7
- Setup Guide 9
- SIO Data Transfer 5–7
- Software Licence Agreement 1
- Special-Purpose Variables 2–5
- Stacks 4–65
- STOP Mode 1–1
- System Data Area 5–1
- System Variables 3–1

**T**

Tag Reference Manual 9  
 Third-Party Claims or Damages 1  
 Timer Instructions 4–4  
 Timer Variables 2–5  
 Trademarks, Registered 9  
 Transfer Speed (I/O Monitor) 8–5  
 Troubleshooting  
   DIO Unit Errors 8–17  
   Flex Network Interface 8–11  
   I/O Unit Errors 8–11

**U**

User Manual 9

**V**

Variable AND Instruction Search Feature,  
   Ladder Monitor 6–15  
 Variable LS and Read Area Size 5–8  
 Variable Names 2–2  
 Variable Search Feature, Ladder Monitor 6–9, 6–11  
 Variables 2–1  
   Array elements 2–7  
   Arrays 2–3  
   Attributes 2–5  
   Available Memory in the GLC 2–3  
   Counter 2–5  
   Device Addresses and Variable Names 2–1  
   Discrete 2–3  
   Hardware-independent 2–1  
   Integer 2–3  
   Naming 2–2  
   Real 2–5  
   Registering 5–7  
   Special Purpose 2–5  
   Storing in the GLC 2–3  
   System 3–1  
   Timer 2–5

**W**

Warnings and Precautions 10  
 Watchdog Timer 4–66

**Z**

Zoom Display Feature, Ladder Monitor 6–7

# *Memo*