

PREFACE

Thank you for purchasing Pro-face's ladder logic programming software, Pro-Control Editor Ver. 4.0, for use with Pro-face's GLC series of graphical logic controllers.

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. 4.0 (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 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.

© Copyright 2002 Digital Electronics Corporation. All rights reserved.
Digital Electronics Corporation, January 2002.

For the rights to trademarks and trade names, see "TRADEMARK RIGHTS."

TABLE OF CONTENTS

PREFACE	1
TABLE OF CONTENTS	2
APPLICABLE PRODUCTS	6
TRADEMARK RIGHTS	6
HOW TO USE THIS MANUAL	7
PRODUCT USAGE PRECAUTIONS	8
DOCUMENTATION CONVENTIONS	9

CHAPTER 1 CONTROLLER FEATURES

1.1 Operation Mode Overview	1-1
1.1.1 GLC Scan Overview	1-1
1.1.2 Controller Feature Overview	1-2
1.1.3 RUN Mode	1-4

CHAPTER 2 VARIABLES

2.1 Variable Names	2-1
2.2 Variable Types	2-3
2.3 Accessing Variables	2-6

CHAPTER 3 SYSTEM VARIABLES

3.1 System Variable List	3-1
3.1.1 How to Use System Variables	3-2
3.2 System Variable – Details	3-3
3.2.1 #AvgLogicTime	3-3
3.2.2 #AvgScanTime	3-3
3.2.3 #Clock100ms	3-4
3.2.4 #Day	3-5
3.2.5 #ForceCount	3-5
3.2.6 #IOStatus	3-6
3.2.7 #LogicTime	3-6
3.2.8 #Month	3-7
3.2.9 #Platform	3-7
3.2.10 #ScanCount	3-7
3.2.11 #ScanTime	3-8
3.2.12 #Status	3-8
3.2.13 #Time	3-9

3.2.14 #Version	3-10
3.2.15 #Year	3-10
3.2.16 #FaultCode	3-11
3.2.17 #FaultRung	3-12
3.2.18 #IOFault	3-12
3.2.19 #Overflow	3-13
3.2.20 #Command	3-14
3.2.21 #DisableAutoStart	3-14
3.2.22 #Fault	3-14
3.2.23 #FaultOnMinor	3-15
3.2.24 #PercentAlloc	3-15
3.2.25 #Screen	3-15
3.2.26 #TargetScan	3-16
3.2.27 #WatchdogTime	3-16

CHAPTER 4 INSTRUCTIONS

4.1 Instruction List.....	4-1
4.2 Instruction Details	4-5
4.2.1 NO (Normally Open).....	4-5
4.2.2 NC (Normally Closed)	4-6
4.2.3 OUT/M (Output Coil)	4-7
4.2.4 NEG (Negated Coil)	4-8
4.2.5 SET (Set Coil)	4-9
4.2.6 RST (Reset Coil)	4-10
4.2.7 PT (Positive Transition Contact)	4-11
4.2.8 NT (Negative Transition Contact)	4-12
4.2.9 AND (And)	4-13
4.2.10 OR (Or)	4-14
4.2.11 XOR (Exclusive OR)	4-15
4.2.12 NOT (Bit Invert)	4-16
4.2.13 MOV (Transfer)	4-16
4.2.14 BMOV (Block Transfer)	4-18
4.2.15 FMOV (Fill Transfer).....	4-19
4.2.16 ROL (Rotate Left)	4-20
4.2.17 ROR (Rotate Right).....	4-21
4.2.18 SHL (Shift Left).....	4-22
4.2.19 SHR (Shift Right)	4-23
4.2.20 ADD (Add)	4-26

4.2.21 SUB (Subtract).....	4-26
4.2.22 MUL (Multiply).....	4-27
4.2.23 DIV (Divide).....	4-28
4.2.24 MOD (Modulus).....	4-29
4.2.25 INC (Increment).....	4-29
4.2.26 DEC (Decrement).....	4-30
4.2.27 EQ (Compare: =).....	4-30
4.2.28 GT (Compare: >).....	4-31
4.2.29 LT (Compare: <).....	4-31
4.2.30 GE (Compare: >=).....	4-32
4.2.31 LE (Compare: <=).....	4-32
4.2.32 NE (Compare: <>).....	4-33
4.2.33 TON (Timer ON Delay).....	4-33
4.2.34 TOF (Timer OFF Delay).....	4-35
4.2.35 TP (Timer Pulse).....	4-37
4.2.36 CTU (UP Counter).....	4-39
4.2.37 CTD (DOWN Counter).....	4-40
4.2.38 CTUD (UP/DOWN Counter).....	4-41
4.2.39 BCD (BCD Conversion).....	4-42
4.2.40 BIN (Binary Conversion).....	4-43
4.2.41 ENCO (Encode).....	4-43
4.2.42 DECO (Decode).....	4-44
4.2.43 JMP (Jump).....	4-45
4.2.44 JSR (Jump Subroutine).....	4-45
4.2.45 RET (Return Subroutine).....	4-46
4.2.46 FOR/NEXT (Repeat).....	4-46

CHAPTER 5 LS AREA REFRESH

5.1 LS Area Refresh Overview.....	5-1
5.2 LS Area Refresh Settings.....	5-2
5.3 Sharing Data with External Devices.....	5-3
5.3.1 LS Area Refresh Cautions.....	5-5

CHAPTER 6 I/O DRIVERS

6.1 I/O Drivers Overview.....	6-1
6.2 Flex Network I/F Driver.....	6-2
6.2.1 Flex Network I/F Unit Self-Diagnosis.....	6-2
6.2.2 I/O Monitor (I/O Connection Check).....	6-5
6.2.3 Troubleshooting.....	6-10

6.3 DIO Driver	6-12
6.3.1 DIO Unit Self-Diagnosis	6-12
6.3.2 I/O Monitor (I/O Connection Check)	6-14
6.3.3 Troubleshooting	6-15

CHAPTER 7 ERROR MESSAGES

7.1 Error Message List	7-1
7.2 Error Codes	7-3
7.3 Program Errors	7-4

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.

Trademark / Tradename	Rights Holder
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).

Term used in this manual	Formal Tradename or Trademark
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. 4.0 software. In this manual, the following names are used to describe series units and products. “GP Type” refers to the GP-PRO/PB III for Windows Ver. 6.0.

Series		Product Name	Model	GP Type
GLC100 Series	GLC100 Series	GLC100L	GLC100-LG41-24V	GLC100L
		GLC100S	GLC100-SG41-24V	GLC100S
GLC300 Series	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
	GLC2600 Series	GLC2600T	GLC2600-TC41-24V	GLC2600

HOW TO USE THIS MANUAL

The GP-PRO/PB III C-Pack01 comprises seven manuals. Refer to the following table for a summary of the contents of these manuals, which are included in the CD-ROM (Disc 2) as PDF files. (The Installation Guide PDF file is not included.)

In addition to these manuals, data files containing supplemental information on updated functions are also provided. To access these additional data files, click the Start button on your Windows OS main screen. On the Programs | Pro-face | ProPB3 C-Package, click ReadMe.

Reference *For detailed information on Digital Electronics Corporation hardware products, refer to each model's user manual (sold separately).*

GP-PRO/PB III C-Package 01	
Setup Guide	Describes the procedures for installation and basic operation of this product.
Pro-Control Editor Ver. 4.0	
User Manual (this manual)	Describes the software settings, variables, and commands used for GLC series units.
Operation Manual	Includes the tutorial for preparation through operation of the product, and an extensive warning/error message list. Also describes the procedures for using the variables registered in Pro-Control Editor on the GP-PRO/PB III screen.
GP-PRO/PB III for Windows Ver. 6.0	
Operation Manual	Describes the operating procedures and software functions used to create the GP screen.
Tag Reference Manual	Includes detailed descriptions of the Tags used to specify functions of the GP unit.
Parts List	Describes both the pre-designed Parts included with GP-PRO/PB III and the symbols that can be called up.
Device/PLC Connection Manual	Describes connections between GP series units and other products, such as manufacturer-specific PLCs, temperature controllers, and inverters.



Note:

- The GP-PRO/PB III user manual is a GP screen creation reference. However, when creating GLC screens for use when operating the GP-PRO/PB III, refer to “GP” as “GLC.”
- Refer to the Online Help guide, in addition to the PDF manual listed above, for detailed explanations about this product.

The layout sheets that are installed as part of the GP-PRO/PB III for Windows standard installation are useful for designing tag address settings, etc.

Use the “Device Allocation Table” and “Tag Layout Sheet” layout sheets that are installed as Microsoft Excel data format.

Each file location and name is listed in the following table.

Reference *For information about the use of Microsoft Excel, refer to Microsoft Excel software's user manual.*

Folder Name	File Name	Contents
Pro-face/propbwin/sheet	Device1E.xls	Device Allocation Table
	TAG1E.xls	Tag Layout Sheet
	TAG2E.xls	
	TAG3E.xls	
	TAG4E.xls	

The PDF file manuals included in the CD-ROM can be viewed using Adobe Acrobat Reader.

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.

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 GLC distributor.

Also, if you have any question about your personal computer, Windows 95, Windows 98, or Windows NT, please contact your local distributor or manufacturer.


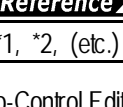
SAFETY SYMBOLS AND TERMS

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

Symbol	Description
 Warning	Incorrect operation resulting from negligence of this instruction may cause death or serious injury.
 Caution	Incorrect operation resulting from negligence of this instruction may cause injury or damage to equipment.
 Important	Failure to observe this instruction may cause abnormal operation of equipment or data loss.
 Careful!	This instruction/procedure must be performed to ensure correct product use.
 STOP	This action/procedure should NOT be performed.

GENERAL INFORMATION SYMBOLS AND TERMS

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

Symbol	Description
 Note:	Provides hints on correct use or supplementary information.
 Reference	Indicates related information (manual name, chapter, section, page number).
*1, *2, (etc.)	Indicates footnotes.
Pro-Control Editor	Software used for editing, transferring, and monitoring of a GLC ladder logic program.
Controller	Indicates the GLC unit's built-in control feature.
GP-PRO/PB III	The screen creation software GP-PRO/PB III for Windows Ver. 6.0.
GLC	Indicates the GLC series of graphic logic controller manufactured by Digital Electronics Corporation.
External Data Communication Device	Indicates peripheral devices such as PLCs (Programmable Logic Controllers), Temperature Controllers, and Inverters. However, the devices connected with Flex Network, Uniwire, or DIO are not included.

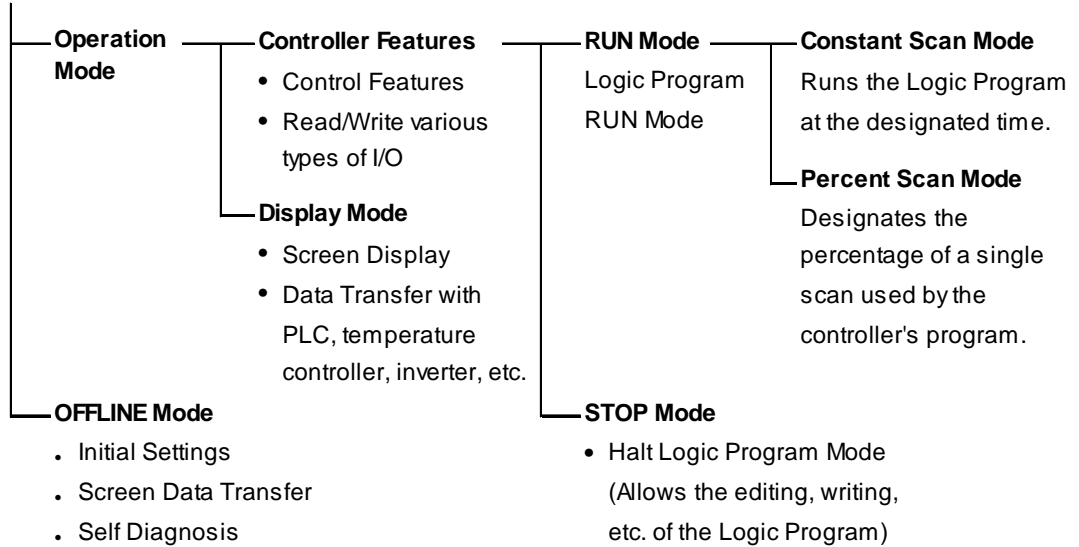
Memo

1 Controller Features

1.1 Operation Mode Overview

The GLC contains both screen display and I/O control features. The following overview describes the GLC operation modes.

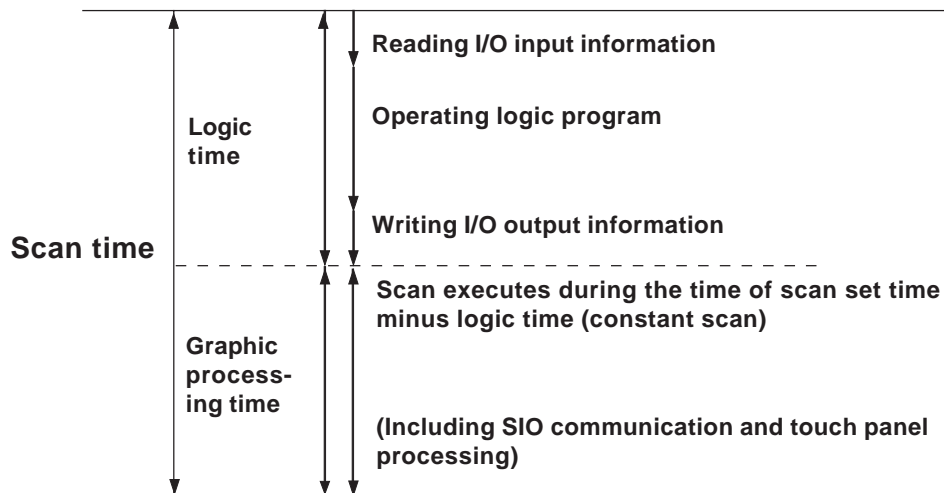
GLC Features



When OFFLINE mode is entered, the controller will stop. Re-entering RUN mode will reset the GLC.

1.1.1 GLC Scan Overview

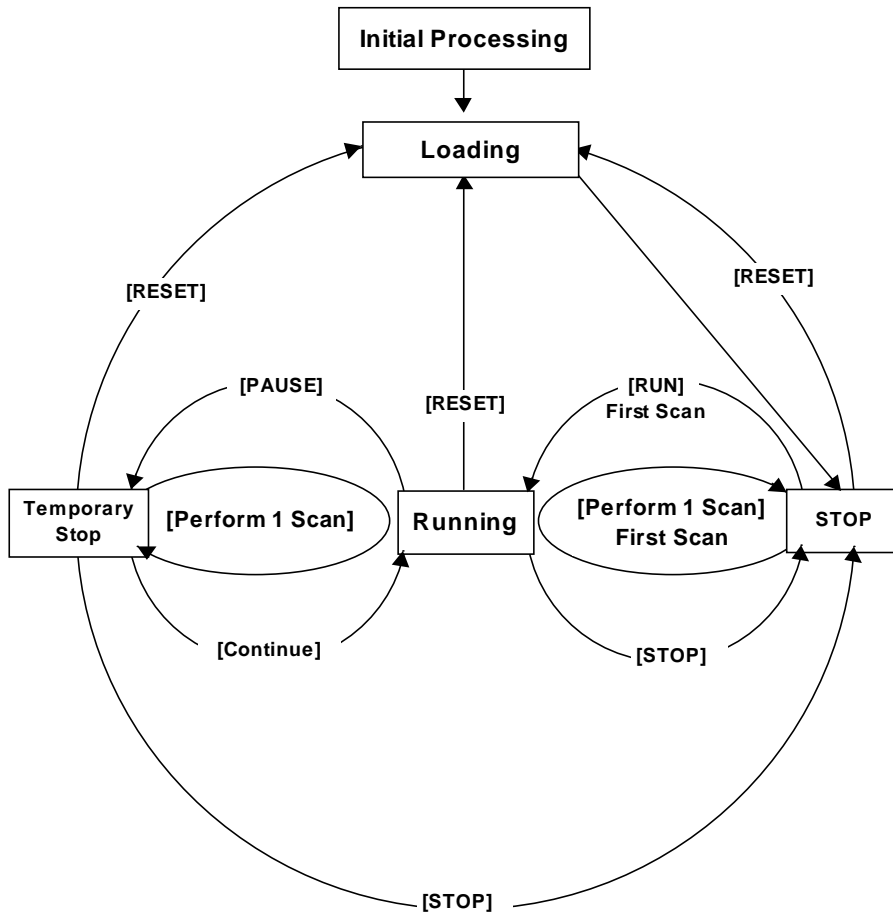
GLC Scan time includes ladder circuit execution time, screen processing time, SIO communication time and touch panel processing time, as follows:



Reference See 1.1.3 – “RUN Mode.”

1.1.2 Controller Feature Overview

The controller feature functions as follows. The following page provides details of each step.



INITIAL PROCESSING

Initial Processing is the original state of the engine used to perform the logic program. Once initialization is finished, the controller enters the Loading state.

LOADING

When the controller enters the Loading state, the logic program is read in from the stored memory of the program to the memory that can perform RUN. After a check is performed to determine whether the logic program is successfully loaded or not, error processing is performed if an error has occurred. If Loading is successful, the program enters the STOP state. If the CONTROLLER STATE is set to START in the GLC OFFLINE mode’s controller settings, the RUN instruction is automatically performed.

Reference For Controller Settings information, refer to the specific GLC unit’s

STOP

In this condition the controller is waiting to receive another instruction. Once the RESET, Perform 1 Scan, Continue, or PAUSE instructions are received, the controller changes to that condition.

- The RESET instruction changes the program to the Loading condition.
At this time, variables are initialized. Retentive variables maintain data before the power shuts down or the GLC resets. However, when the controller is reset by Configuration settings*¹ or #Command, use the value set in the Programming Mode*² as an initial value. Non-retentive variables are cleared to zero.
- The RUN instruction changes the program to the Running condition.
- The Perform 1 Scan instruction performs the program once.

FIRST SCAN

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 performance engine's continuous performance condition. Executes the I/O Read, performs the logic program, executes the I/O Write, and updates the System Variables. (#AvglogicTime, #AvgscanTime, etc.)

- The RESET instruction changes the program to the Loading condition.
- The STOP instruction changes the program to the STOP condition.
- The PAUSE instruction changes the program to the Temporary Stop condition.

TEMPORARY STOP

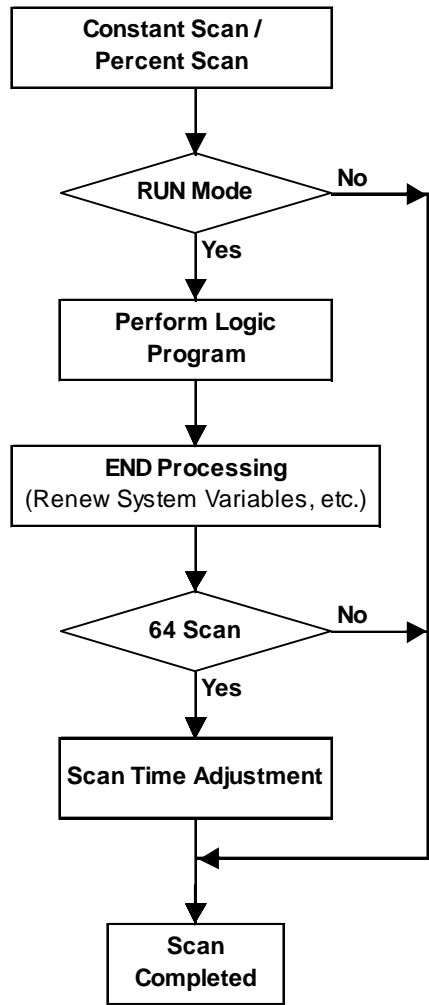
The logic program execution engine is temporarily stopped in this state. To avoid an I/O watchdog timeout, the system executes an I/O Read and I/O Write. However, the logic program is not executed, so the output state does not change. When a command is received, the system switches to the appropriate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

- The RESET instruction changes the program to the Loading condition.
- The Perform 1 Scan instruction performs the program once.
- The STOP instruction changes the program to the STOP condition.
- The Continue instruction changes the program to the Running condition.

1. Mode used to implement the program executed by the controller on the Editor.
2. Mode used to create a program.

1.1.3 RUN Mode

RUN Mode uses the following steps:



SCAN TIME ADJUSTMENT

Scan Time 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.”

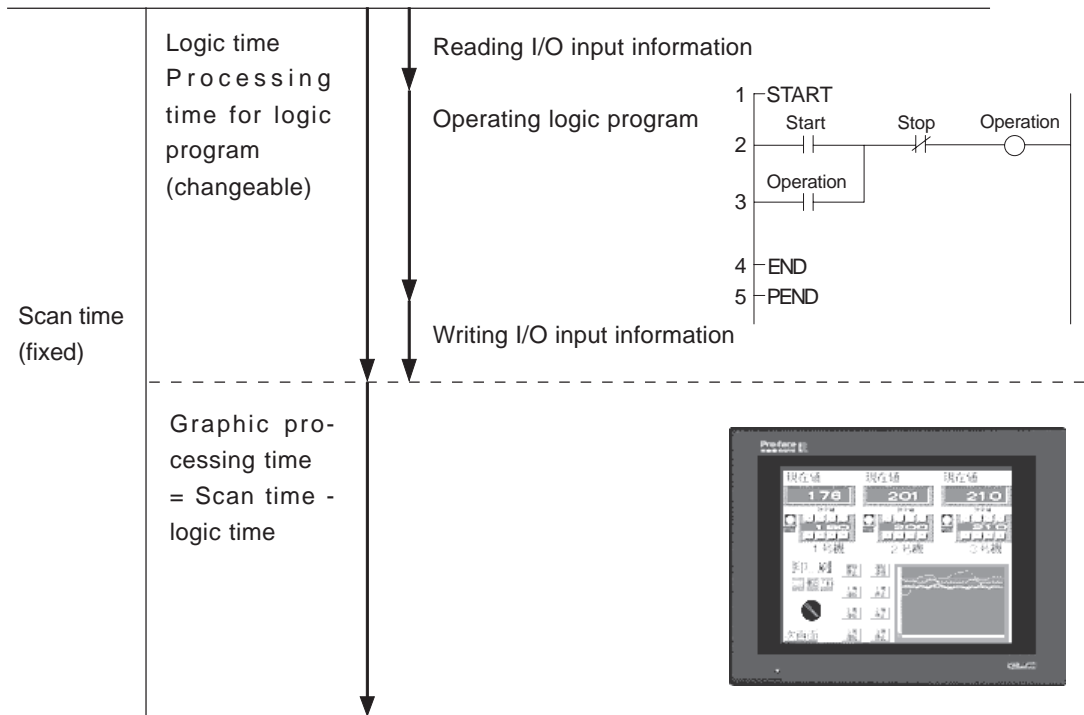


The GLC unit’s ScanTime includes the following error:

Model	Difference
GLC100 Series	approx. -0.2%
GLC300 Series GLC2000 Series	approx. -0.02%

CONSTANT SCAN TIME MODE

Constant Scan Time Mode constantly executes the program during the set scan time. During Constant Scan Time Mode, the screen is used mainly for data display and less for operation, and the control (logic program) is the priority.



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

E.g.: If the constant scan time is set to 50ms and logic executing time is 30ms, the Graphic processing time = 50ms – 20ms (30ms).

A longer logic executing time will result in a shorter Graphic processing time. Therefore, although the GLC unit’s display response time will be slower, the logic program will execute continuously.



If the logic execution time exceeds 50% of the setting time, the logic time will automatically adjust to 50% of the scan time. E.g., when the setting time for constant scan is 50ms and the logic time is 30 ms, the scan time will be 60 ms.

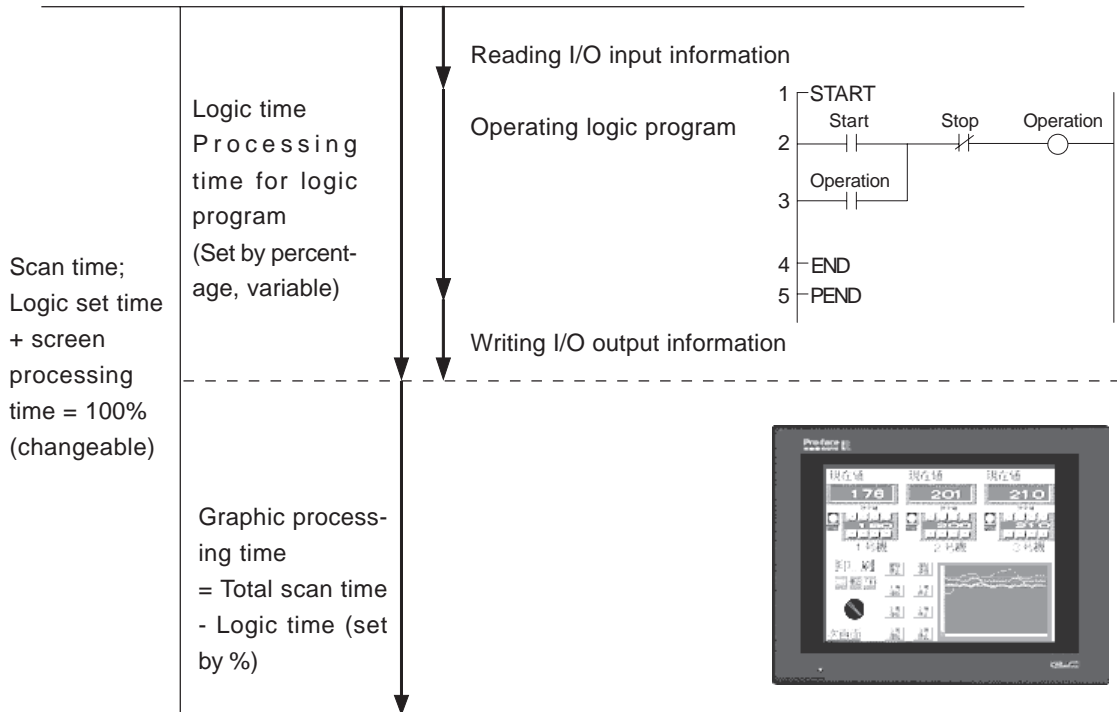


Set the scan time in multiples of 10ms.

PERCENT SCAN TIME MODE

Percent Scan Time Mode varies the scan time according to the percentage set by the logic time.

This feature sets the priority to the operation speed and switching speed of the display and varies the scan time according to the control time (logic program).



Scan time = Logic time / Percent scan setting (%)

E.g.: If the percent scan setting is set to 40%, and the logic executing time is 20ms,
Scan time $(20 \div 40) \times 100 = 50\text{ms}$

Graphic processing time = $50\text{ms} - 20\text{ms} = 30\text{ms}$

A longer logic executing time will result in a shorter display processing time increases, resulting in increased scan time. Therefore, the longer the logic time, the longer the time allocated to display processing; therefore, the display is updated more quickly on the GLC, but the logic program processing cycle slows.



- There is no change in the processing time for one instruction in the logic program.
- The percent scan setting (%) cannot be set over 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.



Set the percent scan value so that the scan time is set every 10ms unit.

2 Variables

This chapter explains the variables used by Pro-Control Editor.

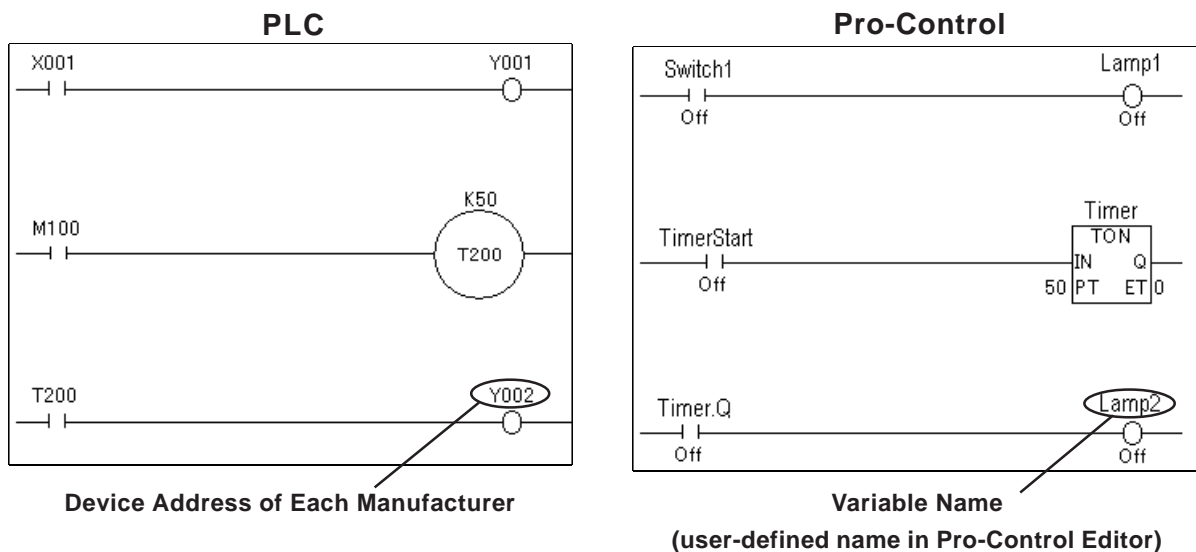
2.1 Variable Names

Pro-Control Editor uses variables to store I/O or counter data. Variables are user designated, and use the designated names in a logic program.

In a conventional PLC, the area that stores data is called a device address, which has a specific name for each PLC manufacturer.

Company-Specific PLC	External I/O	Internal Relay	Timer	Data Register
Mitsubishi	X001	M100	T200	D00001
Omron	01	1001	TIM000	DM0000
Digital	Switch1	Timerstart	Timer	Operating Time, etc.

For Pro-Control Editor, give arbitrary names to these device addresses, and use them as **variables** in the logic program.



Variable names can be designated by the user. When designating variable names, be aware of the following limitations.

- Variable names can be up to 20 characters (20 bytes).
- No differentiation is made between upper- and lower-case characters. However, the order in which words are registered will determine their validity.

E.g.: If the word “TANK” has been entered prior to the word “tank,” the word “tank” will be invalid, even though it can be entered.

- Variable names can use numbers, except for the first character.

Chapter 2 – Variables

- Variable names cannot contain any spaces.
- The underscore (_) is the only special character that can be used.
- Double underscores (_ _) cannot be used (OK: tank_1; Not OK: tank__1).
- Since it is a reserved character, the # sign cannot be used.
- Since the names LS and LSS are reserved for use by the GLC unit’s system in the System Data Area, the Read Area, and for Special Relays, they cannot be used for variable names.

Reference See Chapter 5 – “LS Area Refresh.”



Note:

- If variable names are grouped according to data type, the variables are easily found when searching the variable lists in Pro-Control Editor. (It is easy to see an underscore [_] that is entered between the group name and the variable name.)

E.g.: When several conveyer belts are in the system (Conveyer A, Conveyer B, Conveyer C, etc.), name the motor and sensor variables according to their particular conveyer:

Conveyor A variables:

A_Motor

A_Sensor

Also, name Discrete (bit) as B, Integer as I, floating point as F:

AB_MotorStartingSwitch

AI_MotorRotationNumber

AF_MotorPowerRatio

The variables used for a contact point and a coil are distinguished from the variables used for basic mathematical operations.

- It is convenient to use arrays to set up variable names that are the same as the corresponding PLC device.

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.”

2.2 Variable Types

The Pro-Control software uses three types of variable — Discrete (bit), Integer, and Real. Timer and Counter data types that comprise these variable types are also used. Arrays can be defined and used within each Discrete, Integer, and Real variable type.

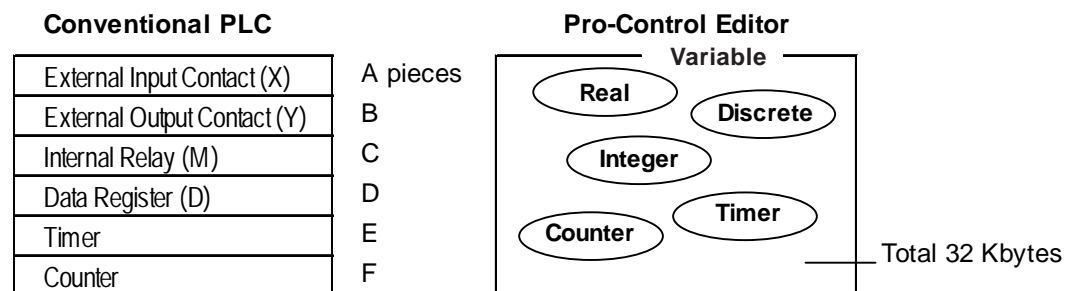
Reference *For details about defining arrays, see 2.3 – “Access to Variables.”*

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. The amount of memory available to the GLC for variables is limited to 32Kbytes. Be sure to design your system so that the number of variables used does not exceed the GLC unit’s available memory limit.

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+ (for each element x 12)
Integer	8
Integer Array	20+ (for each element x 8)
Real	16
Real Array	20+ (for each element x 16)
Timer	48
Counter	80

In the PLC, the number of variable is limited in each device. In the GLC, however, variables can be registered, regardless of type, as long as each takes up no more than 32 Kbytes in the variable storage area.



DISCRETE VARIABLES

These variables use a single bit, with a value of “0” or “1” to define a Discrete condition (i.e., ON or OFF).

INTEGER VARIABLES

These variables use 32 bits to define Integer values from -2147483648 to 214783647.

Chapter 2 – Variables

REAL VARIABLES

These variables use 64 bits to define floating decimal point values ranging from $\pm 2.25e^{-308}$ to $\pm 1.79e^{+308}$, and “0.”



Note:

- To display a Real variable on the GLC screen, set the display data type of the GP-PRO/PB III E tag to Float (32 bits).
- An error occurs when the Real data is converted from 64 bits to 32 bits.
- Because the Integer variable is 32 bits in length, when using a Real variable that is 16 bits in length with the GLC unit's display feature, only the lower 16 bits are available.

Reference Refer to the *GP-PRO/PB III Tag Reference Manual E Tag*.

TIMER/COUNTER

Timer and Counter have a number of special variables.

Each special variable's type is set up individually.

Timer

The following four special variables are used for the Timer instructions.

Special Variables	Description	Variable Type
PT	Preset Value	Integer
ET	Current Value	Integer
Q	Timer Output Bit	Discrete
TI	Timer Measuring Bit	Discrete

By adding a period and a special variable name at the end of the variable name, you can refer to the special variable.

E.g.: Timer.ET

Reference For more information, see 4.2 – “Instruction Details.”



Note:

- When a Timer variable is designated as non-retentive, the special variable Timer.PT remains retentive.

COUNTER

The following seven special variables are used for Counter instructions:

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

By adding a period and a special variable name at the end of the variable name, you can refer to the special variable.

E.g.: Counter.CV

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



Note:

- When a Counter variable is designated as non-retentive, the special variable (PV) remains retentive.
- At scan when the counter is reset, the counter is not updated. One scan is required for resetting a counter.

VARIABLE ATTRIBUTES

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

This section describes each attribute.

Internal

- Used in the internal GLC.
- Cannot be used for the external I/O.
- Is equivalent to the PLC’s Internal relay (internal register).

I/O

- External I/O can be used.
- Variables mapped to I/O in the I/O configuration.
- Is equivalent of the PLC’s I/O relay.

Reference *Refer to the Pro-Control Editor Operation Manual, 2.11 – “I/O Configuration.”*

Retentive

The variable value is retained when the power is down since a retentive variable is managed by static memory. The retentive variable has the value set at the programming mode as an initial value. When the power is shutdown or the GLC unit is reset, the data prior to it is stored. However, when the controller is reset by monitoring mode or #Command, the value set at the programming mode is initialized. Also, reading in CLC PRW file, the result of execution can be saved in the Editor. However, when the retentive variable is used as an initial value, the initial value set at reading in the Editor will be lost if it is designed to change the variables during the execution of the logic program. Be sure to design the system carefully. Non-retentive data will be cleared to zero or OFF.

Global

There are Global and Non-Global variables.

For variables used for tags or parts in the GP-PRO/PB III display feature, set the property to Global. By saving the logic program, the global variables are registered as the GLC symbol in the symbol editor, and can be shared with the GP-PRO/PB III display features. Batch conversion is available by selecting a multiple number of variables in the variable list.

Reference *Refer to the Pro-Control Editor Operation Manual, 2.4 – “Creating Variables.”*

2.3 Accessing Variables

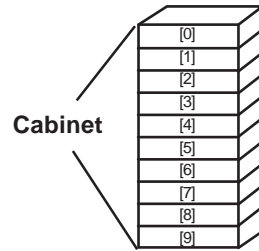
This section explains how to access variable array elements, bits, bytes and words.

ARRAY VARIABLES

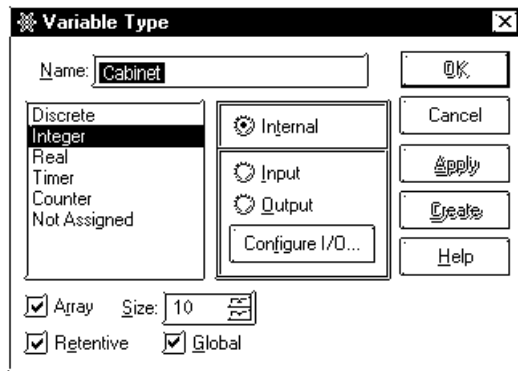
An array is a method of declaring and handling multiple elements with a single variable name. This method allows the user to register same-type variables simultaneously.

An example or analogy could be 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 that is Cabinet and size (number of elements) 10 array. 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, however 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 unit, 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 Integer Array’s Element

Numerical calculation, tracking of the repetitive information, logging of data are available using the Integer Array.

E.g.: To record the number of sodas sold in one month in the Integer Array **Water_Sales**, the structure of data is as follows.

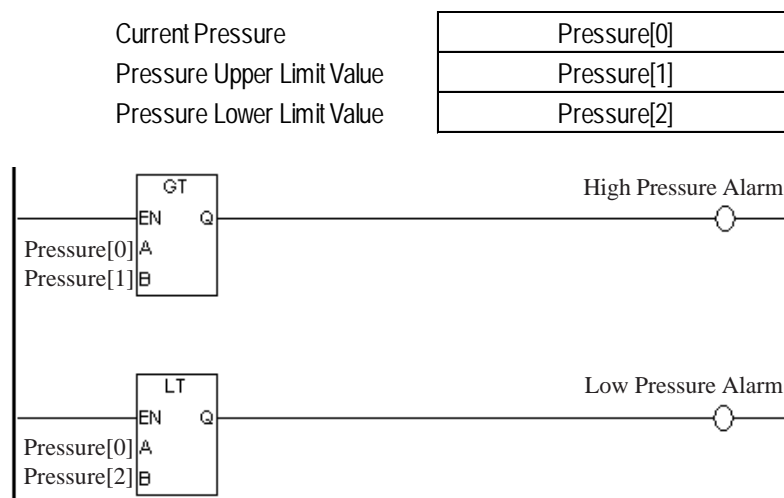
The array consists of 31 Integer type elements which correspond to each day of one month (31 days).

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

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

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

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

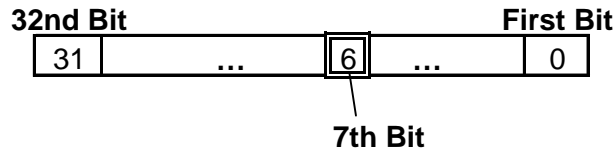


Chapter 2 – Variables

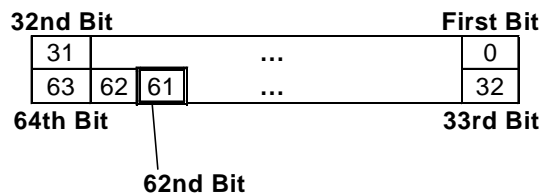
To access a bit with the Integer Array

Also, as with the Discrete array, the modifier [n] can also be used to access any of the Integer array's elements. This method can also be combined with the bit, byte, and word access method. Thus, in order to access the Integer array variable **Water_Sales'** n+1 element's m+1 bit, the wording **Water_Sales[n].X[m]** is used.

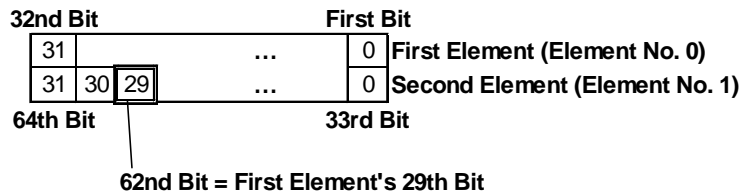
E.g.: To access the Integer array **Alarm's** seventh bit, type **Alarm.X[6]**.



To access the Integer array variable **Water_Sales'** 62nd bit, type **Water_Sales.X[61]**.



Also, for **Water_Sales[1].X[29]**:



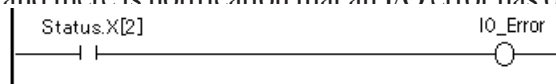
Since **Water_Sales.X[61] = Water_Sales[1].X[29]**, both can be used to access the Integer array **Water_Sales'** 62nd bit.

- When accessing the Integer array variable **Water_Sales'** sixth byte, both **Water_Sales.B[5]** and **Water_Sales[1].B[1]** can be used.
- When accessing the Integer array variable **Water_Sales'** fifth word, 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]** mean the same.

The third bit of the system variable #Status is used as a NO instruction variable in the following example. The third bit of #Status notifies 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 and there is 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, a modifier (n) must be attached to each element, which represents the element number. A “0”, however, is used for the first element in the array.

E.g.: When accessing the Real array Solution_Temperature’s fifth element, type **Solution_Temperature[4]**.



Note: GP-PRO/PB III can handle 2048 GLC variables. The elements of the array become single variables. For example, an array with five elements becomes five variables.

Up to 2048 variables can be used in GP-PRO/PBIII for Windows.

Numerical calculation, tracking of the repetitive information, and logging of data are available using Real Arrays.

E.g.: To record the temperature of 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 24-hour day.

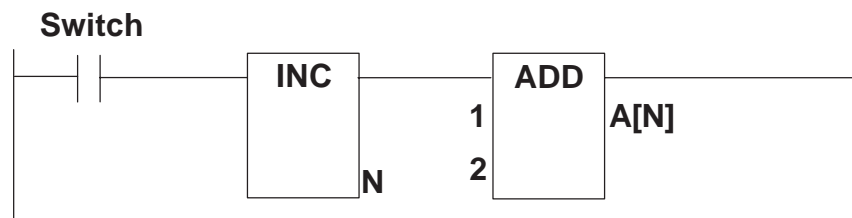
Real element 0 corresponds to the temperature data at 0:00.

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

ARRAY INDIRECT ACCESS

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, if a switch is pressed, N in an INC instruction increments once every scan and 1 is added to 2 with an ADD instruction and then substituted in A[N], then 3 is assigned to A[1]. If five scans are performed, A[1], A[2], A[3], A[4], A[5] are substituted to 3. However, this works out only the initial value of the N value stayed (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 condition, and effect its operation. System variables perform like normal variables, however, since they are reserved, they cannot be automatically created and deleted.

Group	System Variable	Explanation	Initial Value	Variable Name	
Data	#AvgLogTime	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). (Unit: ms)	0	Integer	
	#Clock100ms	Create 0.1s clock.	-	Discrete	
	#Day	Stores Day data as BCD two digits.	-	Integer	
	#EditCount	Currently not used by GLC	-	Integer	
	#ForceCount	Counts the number of times a variable is forced ON or OFF.	0	Integer	
	#IOStatus	Displays the I/O Driver's condition.	-	Integer [10]	
	#LogicTime	Displays the latest Logic Scan Time (Read, Perform, Write). (Unit: ms)	0	Integer	
	#Month	Stores Month data as BCD two digits.	-	Integer	
	#PlatForm	Indicates the controller's platform.	-	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	
	#Status	Indicates the controller's current status.	-	Integer	
	#StopPending	Currently not used by GLC	-	Discrete	
	#Time	Stores Time data as BCD four digits.	-	Integer	
	#Version	Displays the controller's version data.	-	Integer	
	#WCLScan	Currently not used by GLC	-	Integer	
#WCLStatus	Currently not used by GLC	-	Integer		
#Year	Stores Year data as BCD two digits.	-	Integer		

Chapter 3 – System Variables

Group	System Variable	Explanation	Initial Value	Variable Name	
Errors	#FaultCode	Displays the latest error code.	-	Integer	Read Only
	#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 conversion of a variable from Real to Integer.	0	Discrete	
Settings	#Command	Changes the controller's mode.	0	Integer	Write Only
	#DisableAutoStart	Defines the mode entered when the GLC starts up.	-	Discrete	
	#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	
	#LadderMonitor	Reserved for Manufacturer	0	Integer	
	#PercentAlloc	Calculates the Percent Scan's percentage. (Unit: %)	0	Integer	
	#PercentMemCheck	Not currently used by the GLC	-	Integer	
	#RungNo	Reserved for Manufacturer	0	Integer	
	#Screen	Switches GLC screens by assigning screen numbers.	-	Integer	
	#StopScans	Not currently used by the GLC	-	Integer	
	#TargetScan	Sets the Constant Scan Time. (Unit: ms)	-	Integer	
	#WatchdogTime	Sets the Watchdog Timer's value. (Unit: ms)	-	Integer	



Note:

- #Year, #Month, #Day, #Time variables set the GLC unit's Clock data. Set or change the clock data by writing in the initial settings of the GLC unit or in the System Data Area.

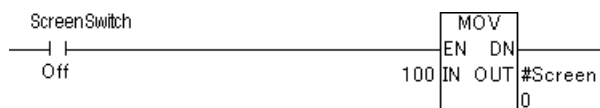
Reference Refer to applicable *GLC User Manual* and the *GP-PRO/PB III Device/PLC Connection Manual*.

- #Clock100ms, #Day, #Month, #Time, #Year, and #Screen are available only with the GLC2000 Series unit.

3.1.1 How to Use System Variables

This section uses a #Screen example to explain how to use the system variables.

The following logic program switches the screen to the base screen (B100), which is screen number 100. Pressing the switch changes the screen by substituting 100 in the #Screen.



3.2 System Variable – Details

This section describes each system variable in detail.

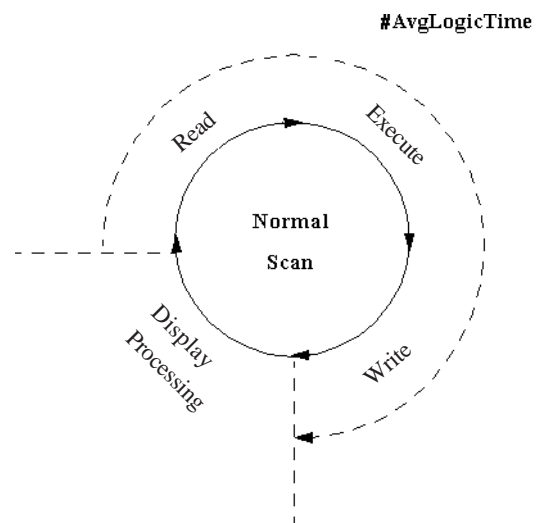
3.2.1 #AvgLogicTime

#AvgLogicTime stores the average amount of time, in milliseconds, that the controller uses in a single scan to read inputs, execute logic, and write outputs. Every 64 scans, this system variable updates the average logic time since its last calculation.

Variable Type: Integer

Set by: Controller

Read Only



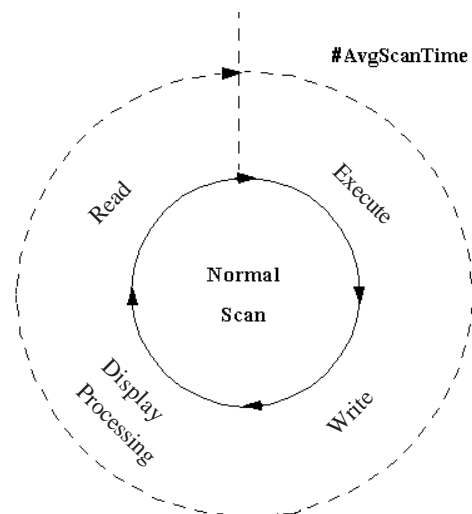
3.2.2 #AvgScanTime

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

Variable Type: Integer

Set by: Controller

Read Only



3.2.3 #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.

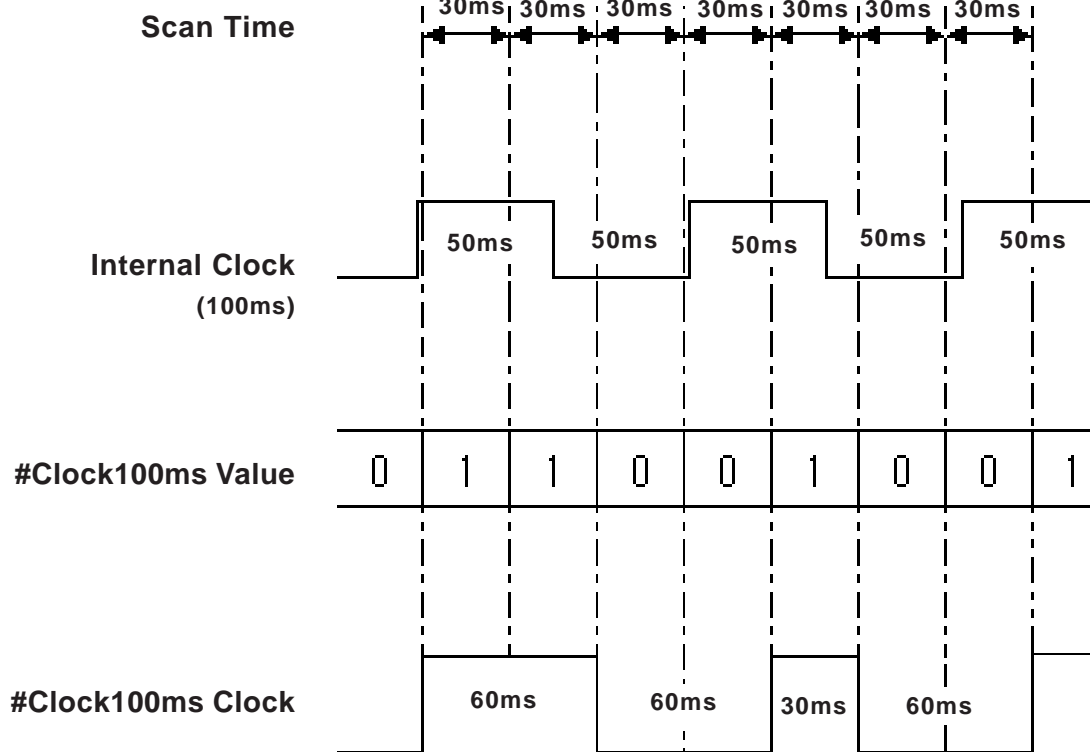
Variable Type: Discrete
 Set by: Controller
 Read Only



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 GLC 2000 Series unit.

Scan Time Every 30ms



3.2.4 #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
System Variable	#Year	#Month	#Day	#Time
Value	01	07	14	0619

- #Day is available only with the GLC2000 Series unit.

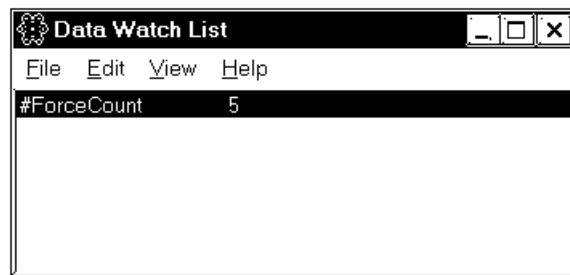
3.2.5 #ForceCount

#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 4.4 – “Forcing Discrete ON/OFF.”*

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.



3.2.6 #IOStatus

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

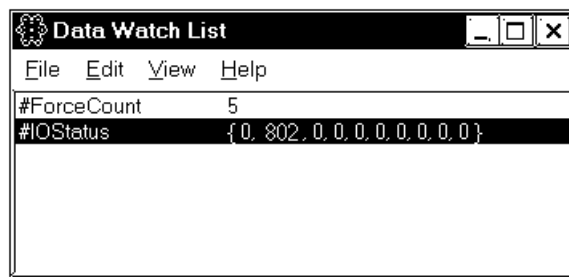
A value of 0 indicates that the I/O is normal. The status indicated by a value other than 0 differs, depending on the I/O driver.

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 6 – “I/O Drivers.”*

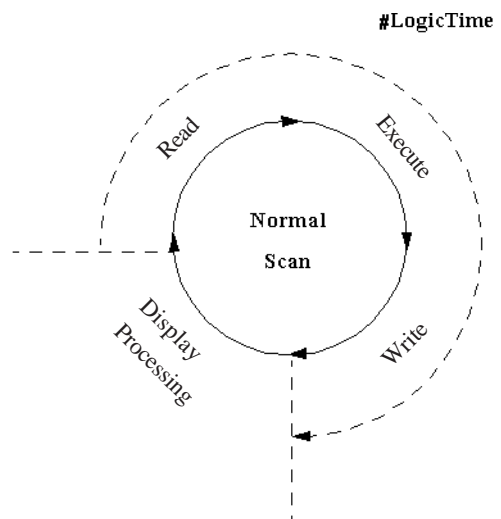
3.2.7 #LogicTime

#LogicTime indicates the amount 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



3.2.8 #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
System Variable	#Year	#Month	#Day	#Time
Value	01	07	14	0619

- #Month is available only with the GLC2000 Series unit.

3.2.9 #Platform

#Platform displays which platform the controller is running on.

Variable Type: Integer

Set by: Controller

Initial Value: 1

Read Only

Value	Platform
2	GLC100
4	GLC300
11	GLC2400
12	GLC2300
13	GLC2600

3.2.10 #ScanCount

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

The value range of #ScanCount is 0 – 4294967295. When the counter value exceeds the maximum value (4294967295), the value of #ScanCount is set to zero (functioning as a Rollover, but without setting the Overflow variable).

Variable Type: Integer

Set by: Controller

Read Only



- Whether or not the logic program is running can be easily checked using #ScanCount.

3.2.11 #ScanTime

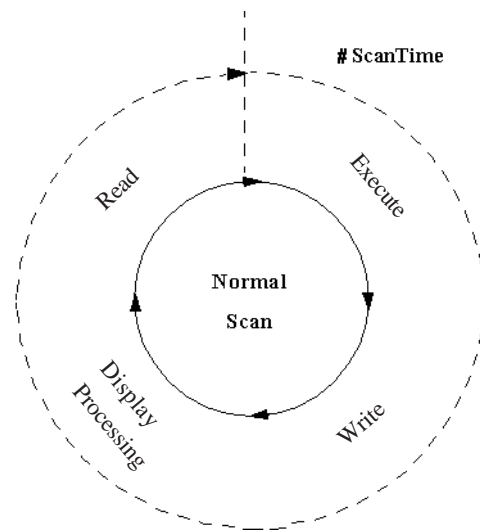
#ScanTime stores the amount of time, in milliseconds, that the controller uses during its last complete scan, to read I/O, execute logic, write I/O, and display processing.

Variable Type: Integer

Set by: Controller

Initial Value: 1

Read Only



3.2.12 #Status

#Status indicates the controller's status.

Within the #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.

Variable Type: Integer

Set by: Controller

Read Only



Note:

Intermittent errors can be detected by using the latch fault flag. Use hexadecimal format for #Status.

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

		Fault Flags
Byte0	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
Byte1	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
Byte2	Bit16	Running
	Bit17	I/O Enabled/Disabled
	Bit18	Forces Enabled/Disabled
	Bit19	Paused
	Bit20	Reserved
	Bit 21-23	Reserved

Byte3	Reserved
--------------	----------

3.2.13 #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
System Variable	#Year	#Month	#Day	#Time
Value	01	07	14	0619

- #Time is available only with the GLC2000 Series unit.

3.2.14 #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. 4.0.0
Byte3	Major version	04
Byte2	Minor version	00
Byte1	Reserved	-
Byte0	Reserved	-

3.2.15 #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
System Variable	#Year	#Month	#Day	#Time
Value	01	07	14	0619

- #Year is available only with the GLC2000 Series unit.

3.2.16 #FaultCode

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

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	Major	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 ¹
14		Reserved by the system
15	Minor	Backup memory's logic program (SRAM) is damaged. Logic program in FEPROM will now be executed. ¹

1. An error occurs only in the GLC2000 Series unit.



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

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

3.2.17 #FaultRung

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

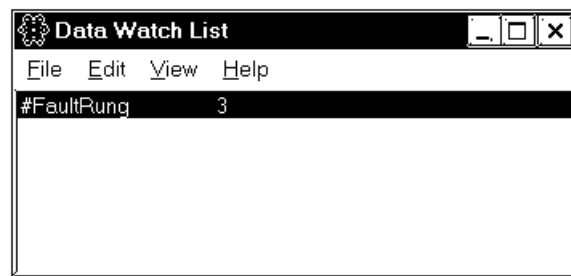
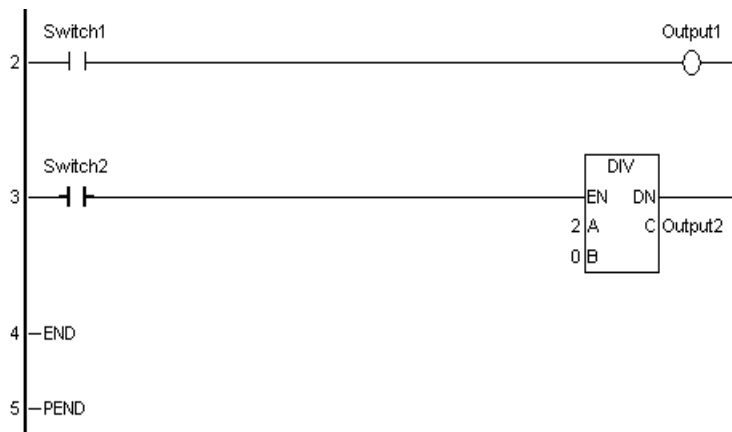
The following example shows when an error occurred at Rung 3.

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

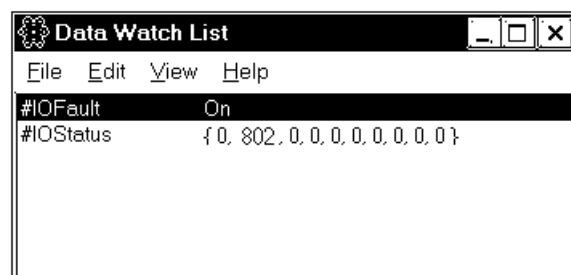
Read Only



3.2.18 #IOFault

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

When #IOFault turns ON, #IOFault is displayed in the Data Watch List window.



Variable Type: Discrete
 Set by: Controller
 Read Only

Reference For I/O driver error code descriptions, see *Chapter 6 – “I/O Drivers.”*

3.2.19 #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.

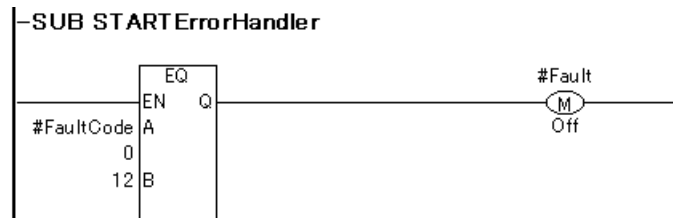
When a mathematical fault occurs, a minor fault also occurs, which executes an ErrorHandler subroutine, if one exists. The ErrorHandler subroutine is an error process subroutine, and must first be created under the name “ErrorHandler.”

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

Reference See 3.2.22 – “#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.

3.2.20 #Command

#Command is an Integer variable used as a controller command. After the controller reads #Command, it resets the value to 0. 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

3.2.21 #DisableAutoStart

#DisableAutoStart is a Discrete variable.

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 it was in (START or STOP) prior to shutdown.



Note: 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

3.2.22 #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 “3.2.19 – #Overflow.”

Variable Type: Discrete
 Set by: User
 Initial Value: OFF
 Writable



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

3.2.23 #FaultOnMinor

#FaultOnMinor is referred to by the controller as to 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.

Reference *For information about ErrorHandler subroutine, see “3.2.19 – #Overflow.”*

Variable Type: Discrete

Set by: User

Initial Value: OFF

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

3.2.24 #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. Usually, #PercentAlloc can be set up in the Setup dialog box.

Reference *See 1.1.3 – “RUN Mode.”*

Variable Type: Integer

Set by: User

Range: 0–50%

Initial Value: 50

Writable

3.2.25 #Screen

The controller stores the screen number set by #Screen.

Variable Type: Integer

Set by: User

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.
- #Screen is available only with the GLC2000 Series unit.



When changing screens, 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.



3.2.26 #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 1.1.3 – “RUN Mode.”

Variable Type: Integer

Set by: User

Range: 10–2000ms

Initial Value: 10ms

Writable

3.2.27 #WatchdogTime

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

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

Variable Type: Integer

Set by: User

Initial Value: 500ms

Writable

4 Instructions

This chapter describes the Pro-Control Editor instructions.

4.1 Instruction List

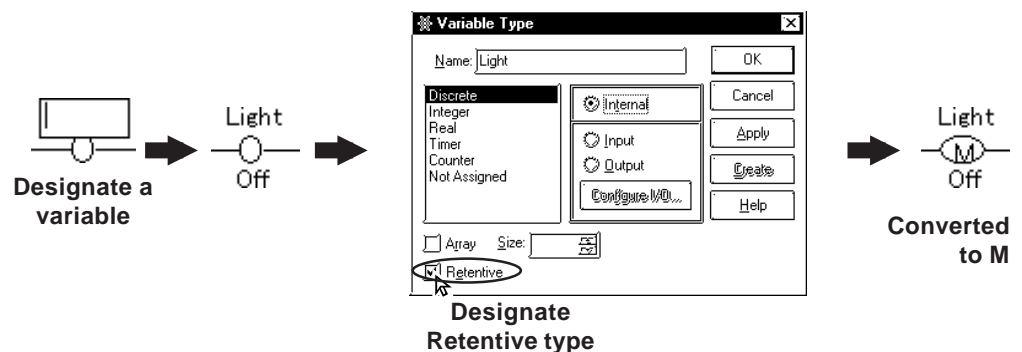
Instructions supported by the Pro-Control Editor software are as follows:

BIT OPERATION INSTRUCTIONS

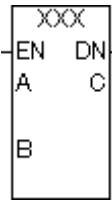
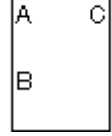
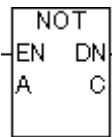
Instruction	Type	Symbol	Function
NO	Normally Open		Allows power to pass when the contact turns ON.
NC	Normally Closed		Allows power to pass when the contact turns OFF.
OUT/M ^{*1}	Output Coil / Retention Coil		Turns physical output devices or internal discrete variables and expressions ON or OFF.
NEG/NM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

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

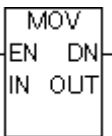
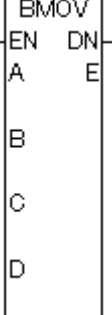
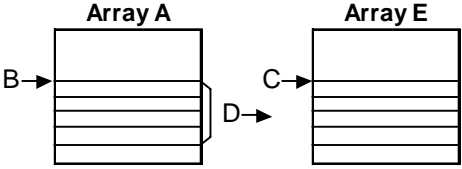
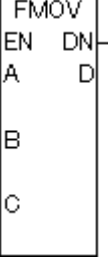
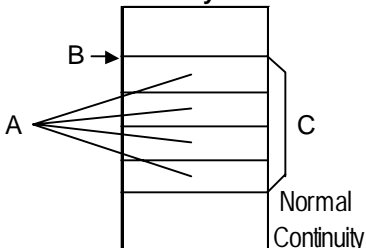
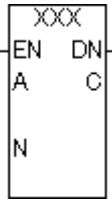
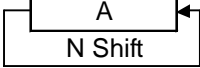
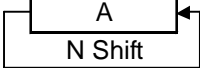
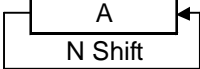
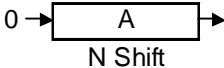
In the following example, when an OUT instruction's variable is retentive, the screen icon changes to M.



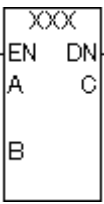
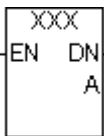
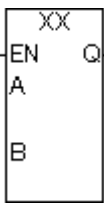
MATHEMATICAL OPERATION INSTRUCTIONS

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



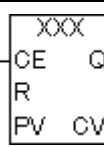
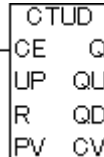
MOVEMENT INSTRUCTIONS

Instruction	Type	Symbol	Function
MOV	Transfer		IN → OUT Normal Continuity
BMOV	Block Transfer		 Normal Continuity
FMOV	File Transfer		 Normal Continuity
ROL	Rotate Left		 → C Normal Continuity
ROR	Rotate Right		 → C Normal Continuity
SHL	Shift Left		 → C Normal Continuity
SHR	Shift Right		 → C Normal Continuity

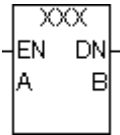
MATHEMATICAL INSTRUCTIONS

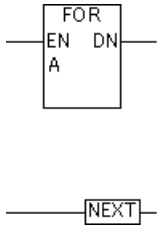
Instruction	Type	Symbol	Function
ADD	Add		$A + B \rightarrow C$ Normal Continuity
SUB	Subtract		$A - B \rightarrow C$ Normal Continuity
MUL	Multiply		$A \times B \rightarrow C$ Normal Continuity
DIV	Divide		$A \div B \rightarrow C$ Normal Continuity
MOD	Residual Processing		$A \% B \rightarrow C$ Normal Continuity
INC	Increment		$A + 1 \rightarrow A$ Normal Continuity
DEC	Decrement		$A - 1 \rightarrow A$ Normal Continuity
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 (\geq)		When $A > \text{ or } = B$, Continuity
LE	Less Than or Equal To (\leq)		When $A < \text{ or } = B$, Continuity
NE	Not Equal (<>)		When $A < > B$, Continuity

TIMER AND COUNTER INSTRUCTIONS

Instruction	Type	Symbol	Function 
TON	Timer ON Delay		See 4.2.33 – "TON (Timer ON Delay)."
TOF	Timer OFF Delay		See 4.2.34 – "TOF (Timer OFF Delay)."
TP	Timer Pulse		See 4.2.35 – "TP (Timer Pulse)."
CTU	UP Counter		See 4.2.36 – "CTU (UP Counter)."
CTD	DOWN Counter		See 4.2.37 – "CTD (DOWN Counter)."
CTUD	UP/DOWN Counter		See 4.2.38 – "CTUD (UP/DOWN Counter)."

CONVERT INSTRUCTIONS

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

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.



Note: ENCO, DECO, FOR, and NEXT are supported with GLC2000 Series units only.

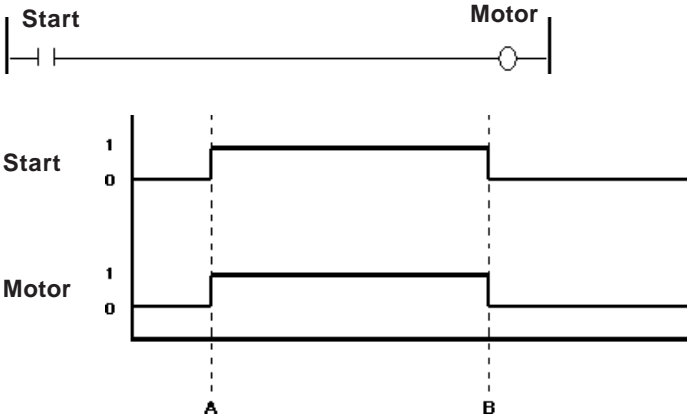
4.2 Instruction Details

This section describes each instruction in detail.

4.2.1 NO (Normally Open)

Variable
—| —

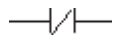
The NO instruction allows power to pass when the variable is ON. The following diagram is an example of the NO instruction's function.



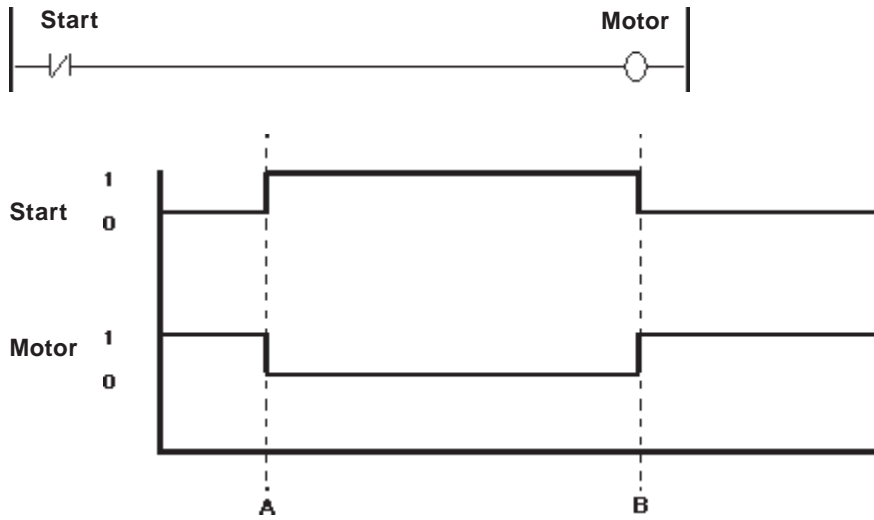
- A: When the Start variable turns ON, the Motor variable turns ON.
- B: When the Start variable turns OFF, the Motor variable turns OFF.

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



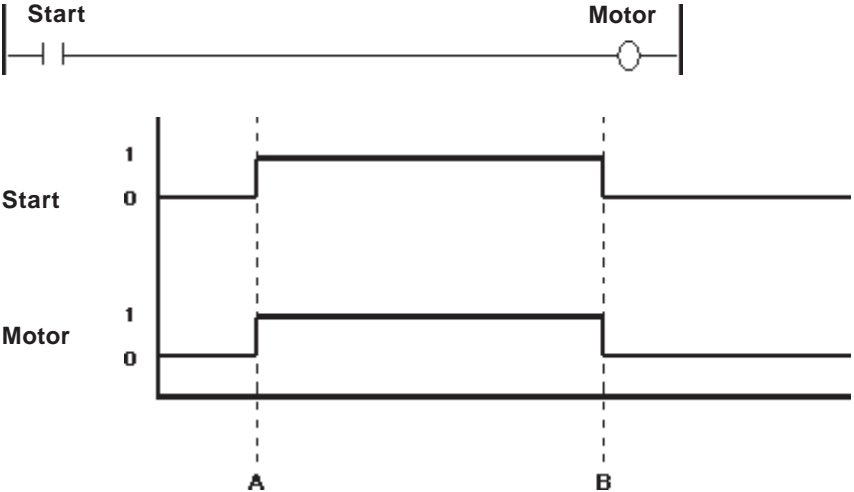
The OUT instruction is used to turn ON/OFF the variables mapped to the I/O, or the Discrete variables in the internal memory .

Because OUT is a coil-type output instruction, and can be used only once per rung, it should appear at the end of a rung.

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



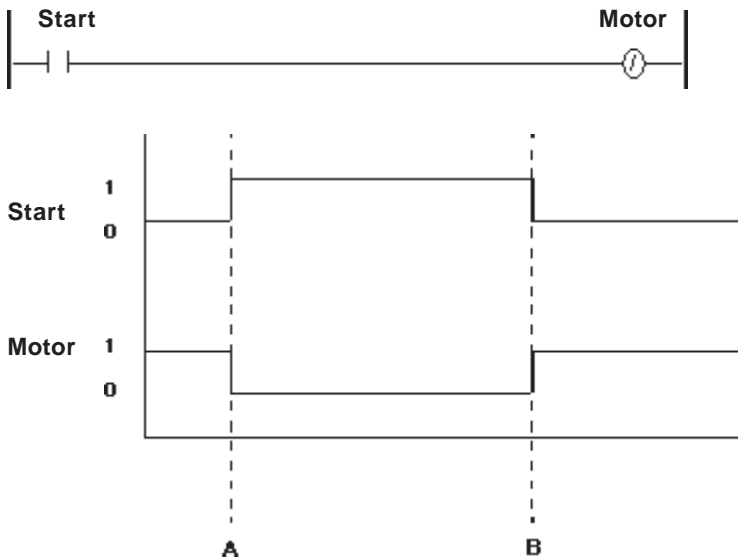
When the NEG instruction is executed, the variable turns OFF when the coil receives power, and ON when the coil does not receive power.

Because NEG is a coil-type output instruction, and can be used only once per rung, it should appear at the end of the rung.

When the variable mapped to NEG instruction is retentive, the following symbol is displayed in the logic program. (Retentive NEG instructions are NOT supported by the GLC100 unit.)



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.



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

4.2.5 SET (Set Coil)

Variable



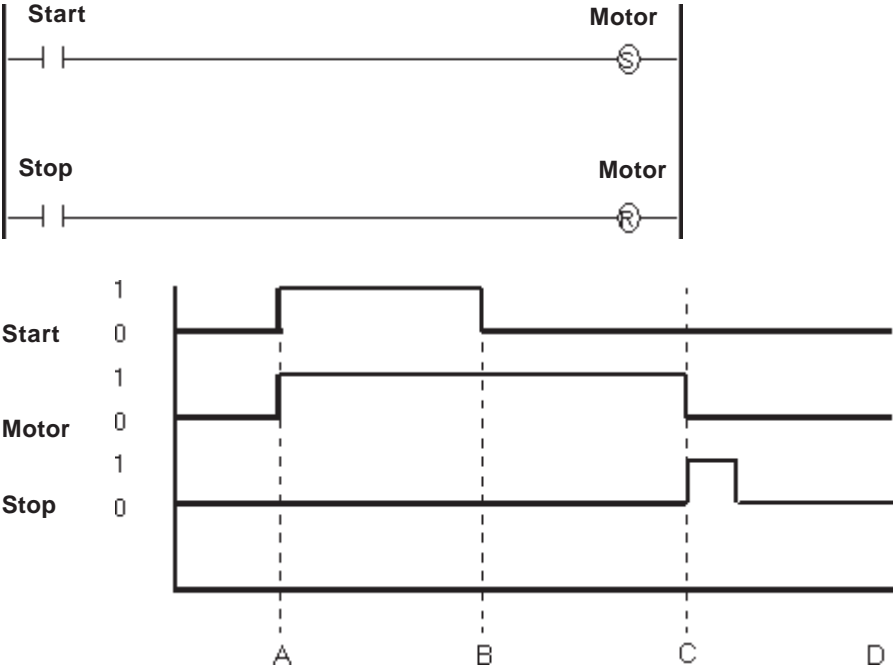
When the SET instruction is executed after the coil receives power, the variable turns ON. The variable will remain ON until explicitly turned OFF by another instruction (such as an RST instruction).

Because SET is a coil-type output instruction, and can be used only once per rung, it should appear at the end of the rung.

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



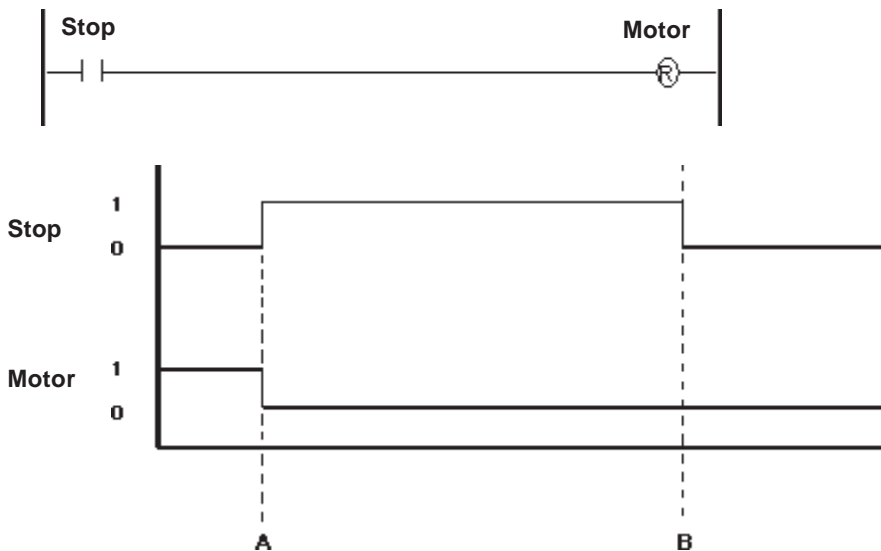
When the coil receives power after the RST instruction is executed, the variable turns OFF. The variable remains OFF until explicitly turned ON by another instruction (such as a SET instruction).

Because RST is a coil-type output instruction, and can be used only once per rung, it should appear at the end of the rung.

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.



- 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.

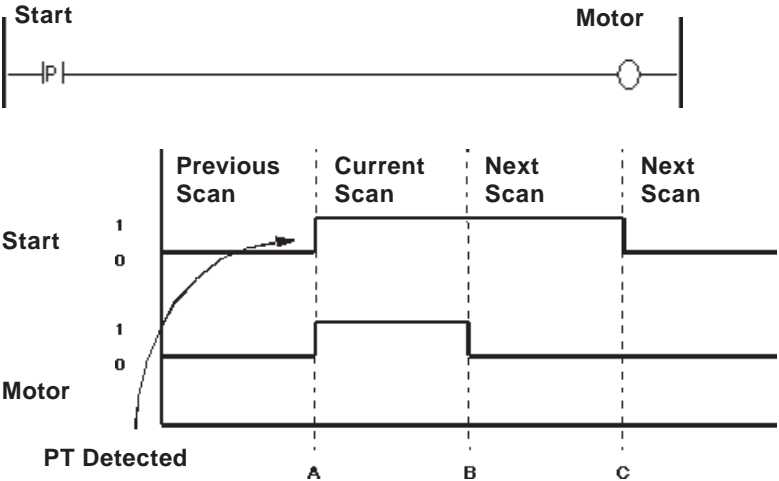
4.2.7 PT (Positive Transition Contact)

Variable
—|P|—

When the PT instruction is executed, it allows power to pass if the variable was OFF during the previous scan but is currently ON.

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: A positive transition contact of the Start variable was not detected during the current scan, and the Motor variable remains OFF.

4.2.8 NT (Negative Transition Contact)

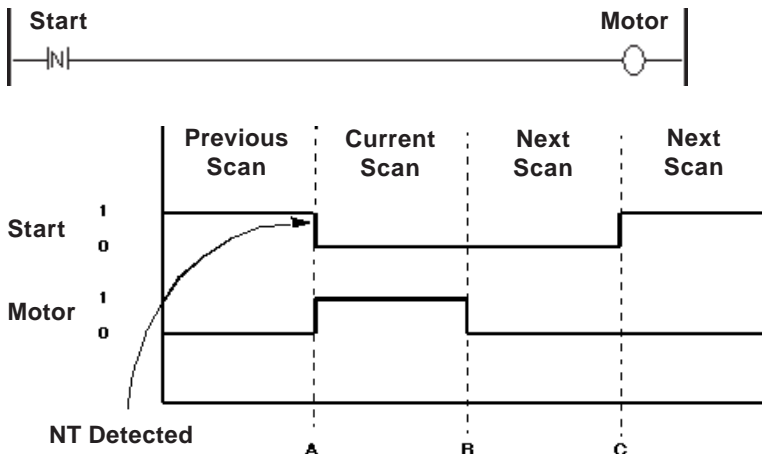
Variable



When the NT instruction is executed, it allows power to pass if the variable was ON during the previous scan but is currently OFF.

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.



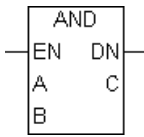
A: When the Start variable turns OFF, the Motor variable turns ON.

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

C: Since a negative transition contact of the Start variable was not detected, the Motor variable remains OFF.

These examples apply when the scan detects the NT instruction.

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	Integer A	0 1 1 0 ... 1 1 0 0
ON	AND	ON	ON	Integer B	1 1 0 0 ... 0 0 0 1
ON		OFF	OFF		
OFF		ON	OFF	Integer C	0 1 0 0 ... 0 0 0 0
OFF		OFF	OFF		

There are three types of AND instruction:

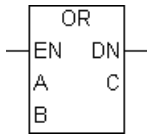
1. When none of the variables are in an array, a simple 32-bit AND operation is performed.
2. When A and C are arrays and B is not in an Integer array, an AND operation is carried out for each element of A and B. The result of each calculation is stored in the corresponding element in C. Both A and C arrays must be the same size.
3. When all three variables have arrays that are the same size, an AND operation is performed on array A and array B. The results are stored in array C.

The AND instruction always passes power.

The AND instruction can be performed with the following combinations of variable types.

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

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	Integer A	Integer B	Integer C
ON	OR	ON	ON	0 1 1 0 ... 1 1 0 0	1 1 0 0 ... 0 0 0 1	1 1 0 0 ... 1 1 0 1
ON		OFF	ON			
OFF		ON	ON			
OFF		OFF	OFF			

There are three types of OR instruction:

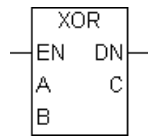
1. When both A and B are Integer variables, a simple 32-bit OR operation is performed.
2. When A and C are arrays and B is not in an Integer array, an OR operation is carried out for each element of A and B. The result of each calculation is stored in the corresponding element in C. Both A and C arrays must be the same size.
3. When all three variables have arrays that are the same size, an OR operation is performed on array A and array B. The results are stored in array C.

The OR instruction always passes power.

The OR instruction can be performed with the following combinations of variable types.

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

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	Integer A
ON	XOR	ON	OFF	0 1 1 0 ... 1 1 0 0
ON		OFF	ON	1 1 0 0 ... 0 0 0 1
OFF		ON	ON	1 0 1 0 ... 1 1 0 1
OFF		OFF	OFF	

There are three types of XOR instruction:

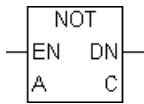
1. When both A and B are Integer variables, a simple 32-bit XOR instruction is performed.
2. When A and C are arrays and B is not in an Integer array, an XOR operation is carried out for each element of A and B. The result of each calculation is stored in the corresponding element in C. Both A and C arrays must be the same size.
3. When all three variables have arrays that are the same size, an XOR operation is performed on array A and array B. The results are stored in array C.

The XOR instruction always passes power.

The XOR instruction can be performed with the following combinations of variable types.

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

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	Integer A	0 1 1 0 ... 1 1 0 0
ON	NOT	OFF	Integer C	1 0 0 1 ... 0 0 1 1
OFF		ON		

There are two types of NOT instruction:

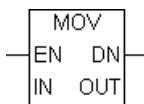
1. When variable A is an Integer, a simple 32-bit NOT operation is performed.
2. When variable A is an array, the NOT operation is performed on array A. The result is stored in array C. Both A and C arrays must be the same size.

The NOT instruction always passes power.

The NOT instruction can be performed with the following combinations of variable types.

A	C
Integer	Integer
Integer Array	Integer Array

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 always passes power.

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

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



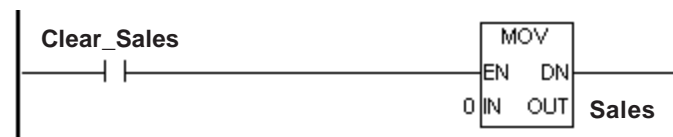
Note: #Overflow will turn ON if the operation involves a Real-to-Integer data-type conversion, and the value is too large to transfer. In this case, the result will be undefined.

The following examples illustrate how to use the MOV instruction..

- The first diagram describes how to clear a variable.
- The second diagram describes how to block transfer arrays.

Example 1: Clear

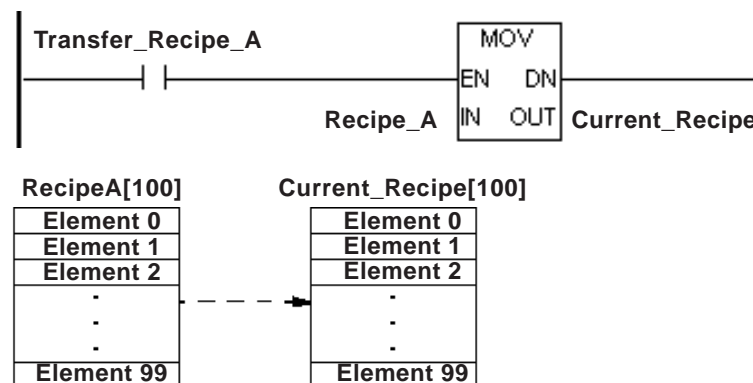
A variable can be cleared with the MOV instruction by transferring a “0” into the variable.



Example 2: Block Transfer

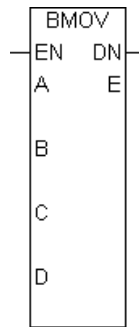
A block transfer can be performed with the MOV instruction by specifying two arrays of the same type and size.

When an array (“Recipe_A”) is transferred to an array that is the same type (“Current_Recipe”), it can be transferred with a single MOV instruction.



4.2.14 BMOV (Block Transfer)

- A: Source variable
- B: Start from Array A[B]
- C: To Array E[C]
- D: Number of Data
- E: Destination variable



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.

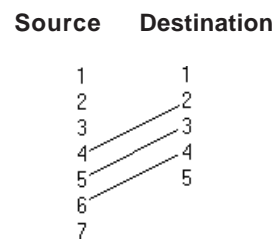
The combinations of valid variable data types for the BMOV instruction are as follows:

A and E	B, C, and D
Integer Array	Integer
	0
	Integer Constant

Operating Example

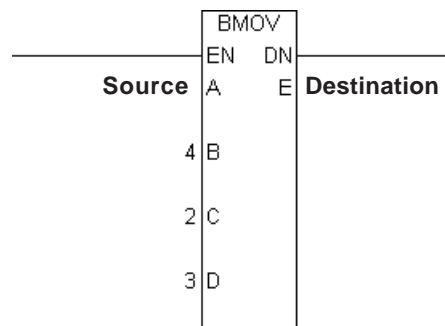
The following diagram is an example of some of the seven Integer array “Source” elements being copied into some of the six Integer array “Destination” elements.

- Source[4] is copied to Destination[2].
- Source[5] is copied to Destination[3].
- Source[6] is copied to Destination[4].



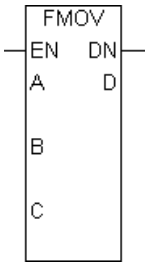
The BMOV instruction is defined as follows:

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.



4.2.15 FMOV (Fill Transfer)

- A: Data
- B: Start from Array D[B]
- C: Number of data
- D: Destination variable



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.

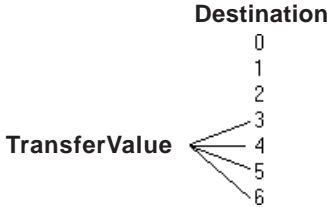
The combinations of valid variable data types for the FMOV instruction are as follows:

A, B, and C	D
Integer	Integer Array
0	
Integer Constant	

Operating Example

The following diagram is an example of an initial value (“TransferValue”) being copied into some of the seven Integer array elements (“Destination”).

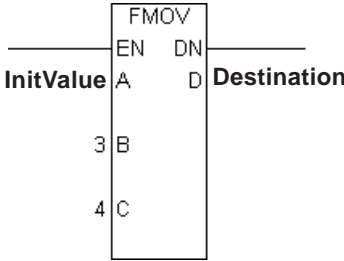
- InitValue is copied to Destination[3].
- InitValue is copied to Destination[4].
- InitValue is copied to Destination[5].
- InitValue is copied to Destination[6].



The FMOV instruction is defined as follows:

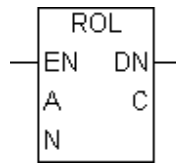
While the program is running, the controller checks whether references to array A and E elements exist in the FMOV instruction. If an invalid array is referred to, a major error will occur and #Faultcode is set to 2.

Reference See 3.2.16 – “#FaultCode.”



4.2.16 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.

There are two types of ROL instruction:

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 x array size) – 1], inclusive.

The ROL instruction always passes power.

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

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

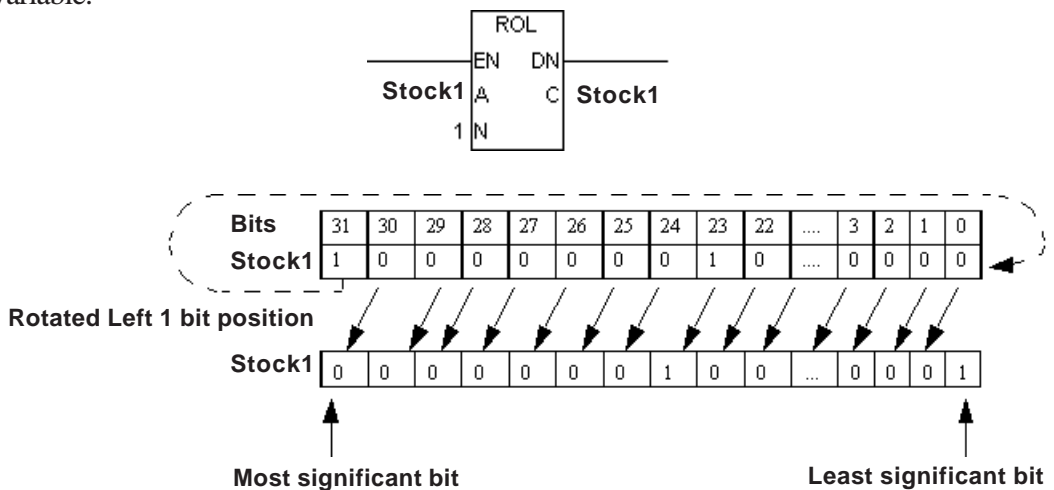


Note: #Overflow is turned ON if N is out of range. The result is undefined.

Reference See 3.2.19 – “#Overflow.”

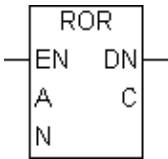
Operating Example

The following diagram is an example of a one-bit left rotation of the “Stock1” Integer variable.



4.2.17 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.

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.

The ROR instruction always passes power.

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

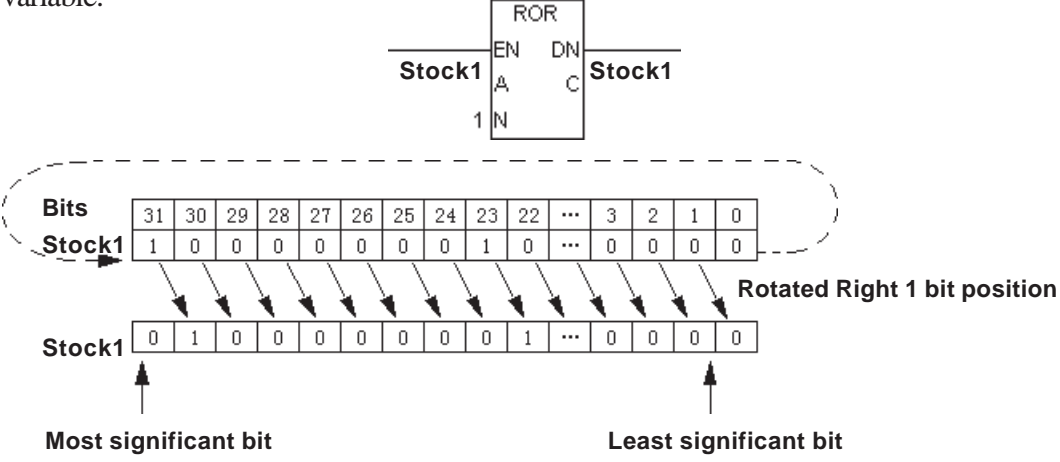
A	N	C
Integer	Integer or Integer Constant	Integer
Integer Array	Integer or Integer Constant	Integer Array is same size as A
Integer Constant	Integer or Integer Constant	Integer

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

Reference See 3.2.19 – “#Overflow.”

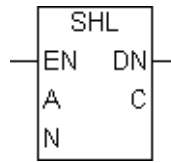
Operating Example

The following diagram is an example of a one-bit right rotation of the “Stock1” Integer variable.



4.2.18 SHL (Shift Left)

- A: Variable name to be rotated
- 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.

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 x array size) – 1], inclusive.

The SHL instruction always passes power.

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

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



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

Reference See 3.2.19 – “#Overflow.”

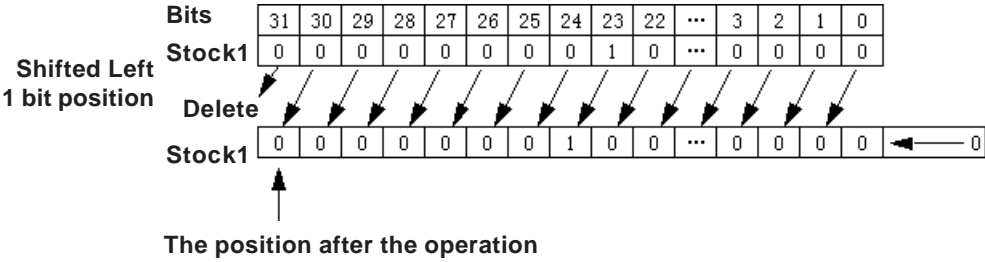
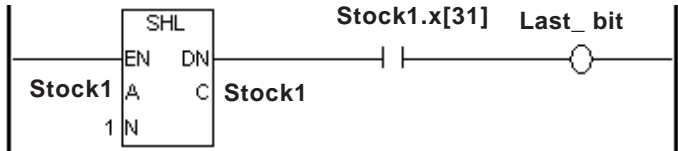
Operating Example

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

Each bit in the “Stock1” Integer indicates the current position.

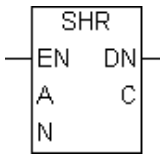
At every program scan (or fixed interval), the bit is left-shifted to the next position.

When the bit reaches the last bit position (31), the “Last_bit” variable is turned ON, signaling the completion of the operation .



4.2.19 SHR (Shift Right)

A: Variable name to be rotated
 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.

There are two types of SHR 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 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 x array size) – 1], inclusive.

The SHR instruction always passes power.

Chapter 4 – Instructions

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

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



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

Reference See 3.2.19 – “#Overflow.”

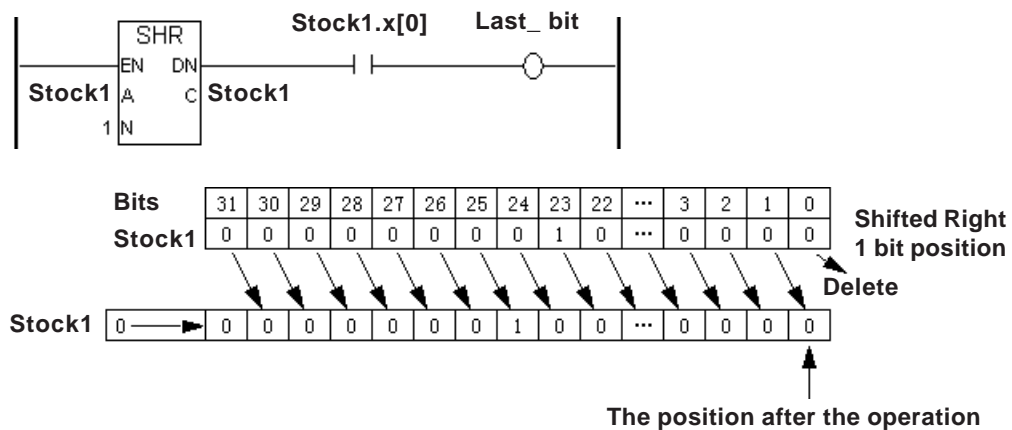
Operating Example – When Using Bits

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

Each bit in the “Stock1” Integer indicates the current position.

At every program scan (or fixed interval), the bit is right-shifted to the next position.

When the bit reaches the last bit position (0), the “Last_bit” variable is turned ON, signaling the completion of the operation.

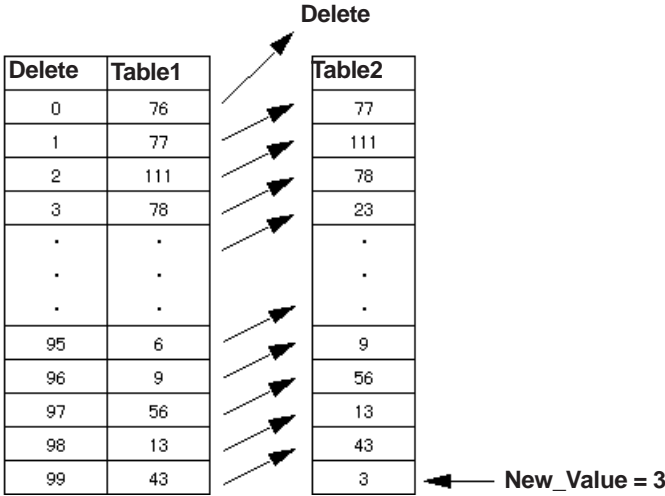
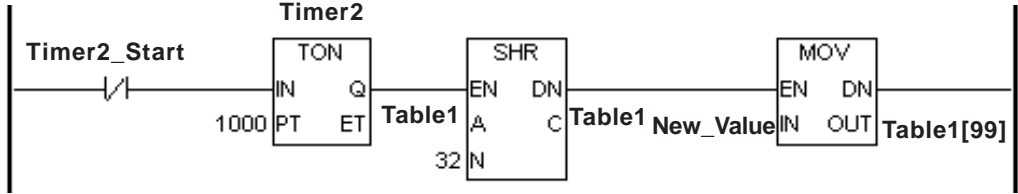


Operating Example – When Using Arrays

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

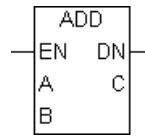
A 32-bit shift rotates the entire 32-bit Integer.

Every second, the “Table1” Integer array’s values are moved up one position towards 0, and a new value is placed at the end of the elements “Table1[99]” in the “Table1” Integer array.



4.2.20 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 combinations of valid variable data types for the ADD instruction are as follows:

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:

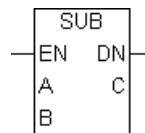
- If the result C exceeds the range expressed with the variable data type in C, #Overflow turns ON and the result of ADD is undefined.

Reference See 3.2.19 – “#Overflow.”

- If either A or B are Reals, both are converted to Reals 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.

4.2.21 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 combinations of valid variable data types for the SUB instruction are as follows:

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:

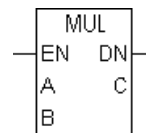
- If the result C exceeds the range expressed with the variable data type in C, #Overflow turns ON and the result of SUB is undefined.

Reference See 3.2.19 – “#Overflow.”

- If either A or B are Reals, both are converted to Reals 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.

4.2.22 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 combinations of valid variable data types for the MUL instruction are as follows:

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:

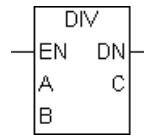
- If the result C exceeds the range expressed by the variable data type in C, #Overflow turns ON and the result of MUL is undefined.

Reference See 3.2.19 – “#Overflow.”

- If either A or B are Reals, both are converted to Reals 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.

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

The DIV instruction always passes power.

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

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



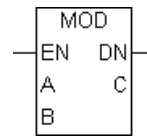
- If B is zero or if the result C exceeds the range expressed by the variable data type in C, #Overflow turns ON and the result of DIV is undefined.

Reference See 3.2.19 – “#Overflow.”

- If either A or B are Reals, both are converted to Reals 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.

4.2.24 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 combinations of valid variable data types for the MOD instruction are as follows:

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.

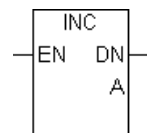
Reference See 3.2.19 – “#Overflow.”

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

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

4.2.25 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 combinations of valid variable data types for the INC instruction are as follows:

A
Integer

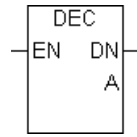


Note: #Overflow is set if A increments from 0x7FFFFFFF to 0x80000000.

Reference See 3.2.19 – “#Overflow.”

4.2.26 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.

Valid variable data types for the DEC instruction are as follows:

A
Integer



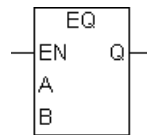
Note: #Overflow is set if A decrements from 0x80000000 to 0x7FFFFFFF.

Reference See 3.2.19 – “#Overflow.”

4.2.27 EQ (Compare: =)

A: Data

B: Data



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

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

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

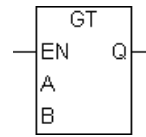


Note: Real values need to be compared very carefully. For example, a calculation might result in 1.9999999999, which is not equal to 2.0000000000.

4.2.28 GT (Compare: >)

A: Data

B: Data



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

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

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

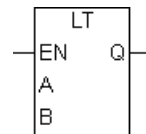
**Note:**

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

4.2.29 LT (Compare: <)

A: Data

B: Data



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

The combinations of valid variable data types for the LT instructions are as follows:

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

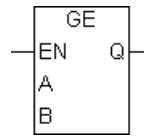
**Note:**

Real values need to be compared very carefully. For example, a calculation might result in 1.99999999999, which is less than 2.

4.2.30 GE (Compare: >=)

A: Data

B: Data



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

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

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



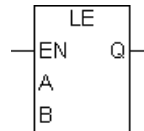
Note:

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

4.2.31 LE (Compare: <=)

A: Data

B: Data



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

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

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



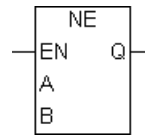
Note:

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.

4.2.32 NE (Compare: <>)

A: Data

B: Data



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

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

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.

4.2.33 TON (Timer ON Delay)

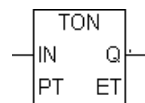
IN: Timer starting bit

PT: Preset time of timer

Q: Time up flag

ET: Present value of timer

Variable



When the timer input bit (IN) receives power, the TON instruction adds the preset time (PT), in milliseconds, and the timer output bit (Q) turns ON.

Operation Overview

Special Variable	Description	Variable Type
PT	Preset Value	Integer
ET	Present Value	Integer
Q	Timer Output Bit	Discrete
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.

Chapter 4 – Instructions

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

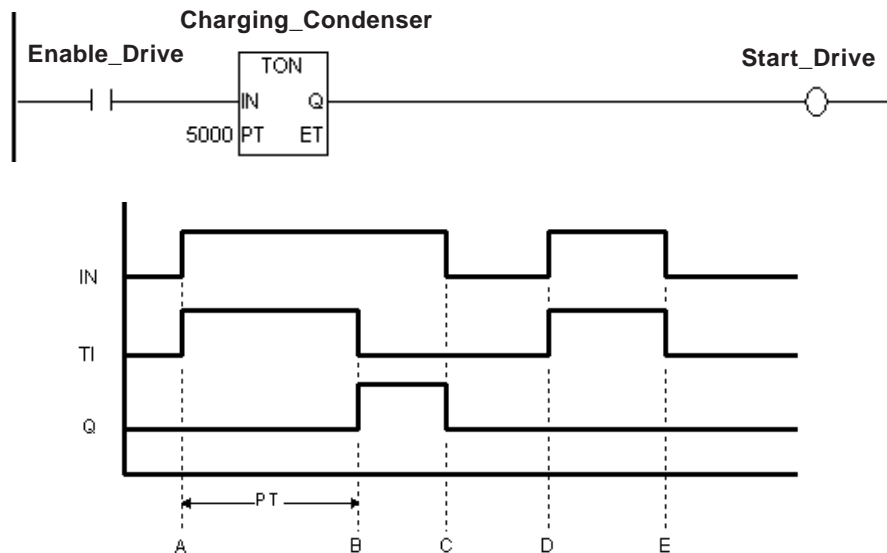
- Variable.ET (the elapsed time) holds the current value.
- Variable.TI (the timing bit) turns OFF.
- Variable.Q (the timer output bit) turns ON, and the instruction passes power.

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.

Operating Example

The following diagram is an example of a drive that starts five (5) seconds after the power is turned ON.

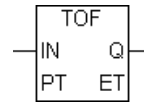


- A: When power is applied to the timer input bit (IN), the timing bit (TI) turns ON, the timer begins timing, and the elapsed time (ET) increments. The timer output bit (Q) remains OFF.
- B: The elapsed time (ET) equals the preset time (PT), the timer output bit (Q) turns ON, and the elapsed time (ET) 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.

4.2.34 TOF (Timer OFF Delay)

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

Variable



When the timer input bit (IN) stops receiving power, the TOF instruction adds the preset time (PT), in milliseconds, and the timer output bit (Q) turns OFF.

Operating Overview

Special Variable	Description	Variable Type
PT	Preset Value	Integer
ET	Present Value	Integer
Q	Timer output bit	Discrete
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.

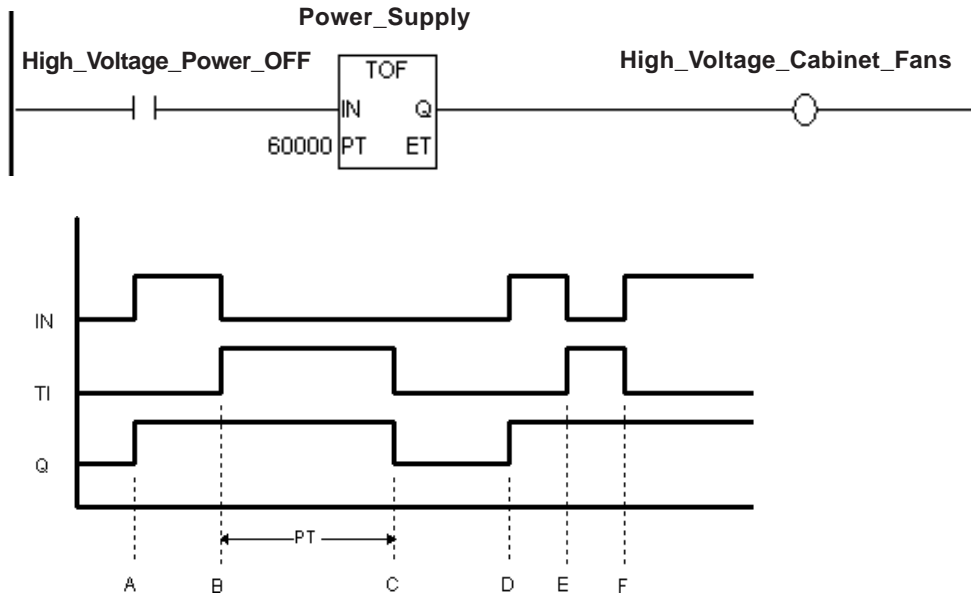
When the elapsed time (Variable.ET) increments and equals the preset time (Variable.PT):

- 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.

Chapter 4 – Instructions

Operating 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.

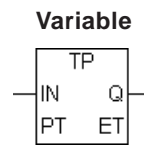
4.2.35 TP (Timer Pulse)

IN: Timer starting bit

PT: Preset time of timer

Q: Time up flag

ET: Present value of timer



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

Operation Overview

Special Variable	Description	Variable Type
PT	Preset Value	Integer
ET	Present Value	Integer
Q	Timer output bit	Discrete
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):

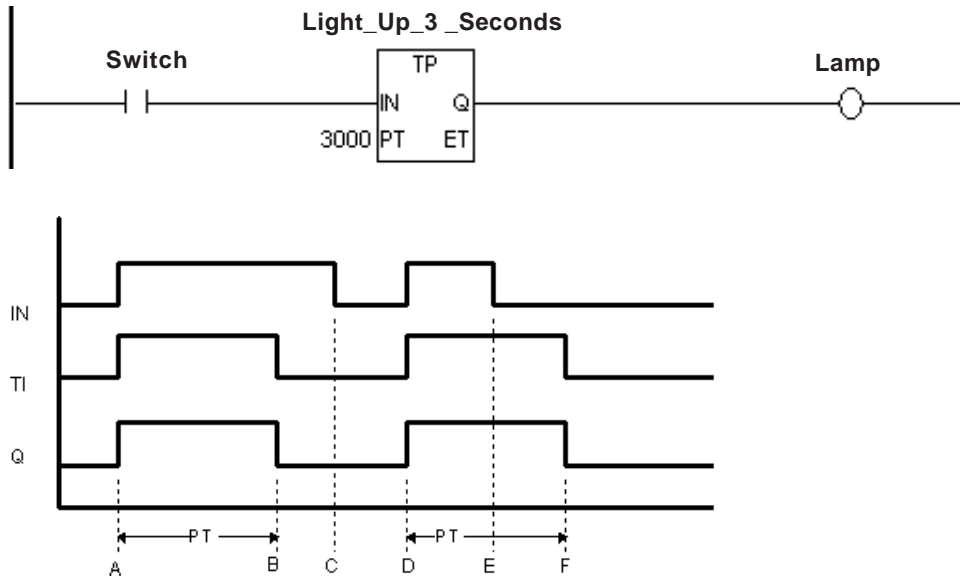
- Variable.ET (the elapsed time) stays fixed at the preset value if the TP instruction is still receiving power.
- Variable.ET (the elapsed time) resets immediately to zero if the instruction stops receiving power.
- Variable.TI (the timing bit) turns OFF.
- Variable.Q (the timer output bit) turns OFF.

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.

Chapter 4 – Instructions

Operating Example

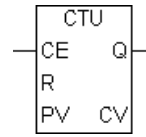
The following diagram is an example of a lamp that lights up for three seconds when the switch is pressed.



- 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.

4.2.36 CTU (UP Counter)

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



Operation Overview

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

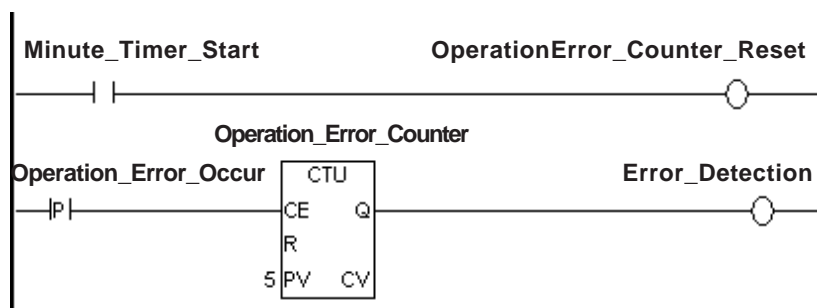
When the current value (Variable.CV) is equal to the preset value (Variable.PV), the counter output bit (Variable.Q) is turned ON, and the instruction passes power.

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.

Operating Example

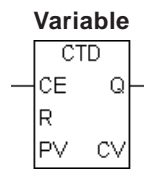
The following diagram is an example of the CTU instruction notifying the Error_Detection output when five errors have been counted during a one-minute period.



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.

4.2.37 CTD (DOWN Counter)

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



Operation Overview

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

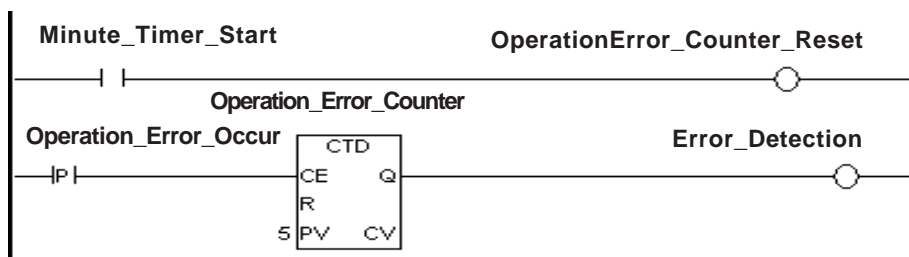
When the current value (Variable.CV) becomes equal to or less than zero after decrementing, the counter output bit (Variable.Q) is turned ON, and the instruction passes power.

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.

Operating 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 CTU instruction's position. The CTD instruction is a level input.

4.2.38 CTUD (UP/DOWN Counter)

CE: Counter starting bit

UP: UP Counter flag

R: Counter reset bit

PV: Preset value of counter

Q: Counter output

QU:UP Counter Output

QD:Down Counter Output

CV: Present value of counter

Variable

CTUD	
CE	Q
UP	QU
R	QD
PV	CV

Operation Overview

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

When the UP Counter enable flag (Variable.UP) is ON, the CTUD instruction operates the same as the CTU (UP Counter) instruction.

When the UP Counter enable flag (Variable.UP) is OFF, the CTUD instruction operates the same as 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.

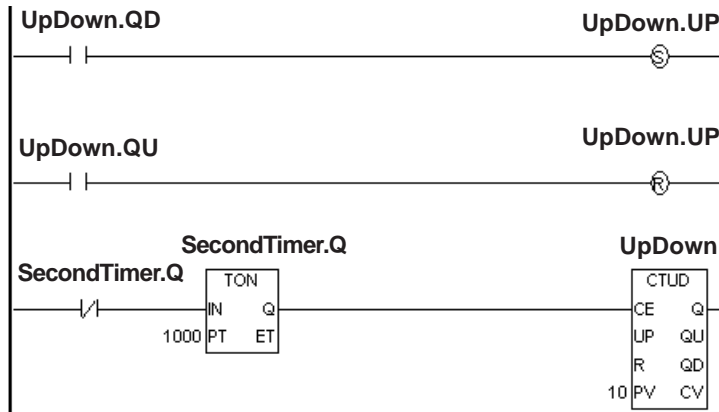
Chapter 4 – Instructions

Operating 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).



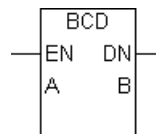
Note:

If the counter reset bit (Variable.R) turns ON when the Counter Up enable flag (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 enable flag (Variable.UP) is OFF, the preset value (Variable.PV) is entered to the current value (Variable.CV).

4.2.39 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.

A	B
Integer	Integer
0	
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 3.2.16 – “#Faultcode,” and 3.2.19 – “#Overflow.”

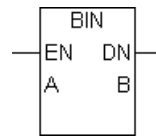


Note:

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

4.2.40 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.

A	B
Integer	Integer
0	
Integer Constant	

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

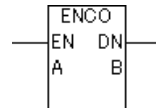
Reference See 3.2.16 – “#Faultcode,” and 3.2.19 – “#Overflow.”



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

4.2.41 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.

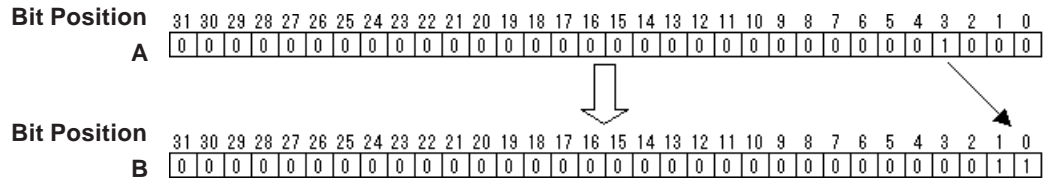
The ENCO instruction always passes power.

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

Chapter 4 – Instructions

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



Note:

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

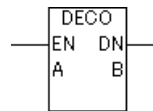
Reference See 3.2.19 – “#Overflow.”

- The ENCO instruction does not support variable modifiers (assigned bit, word, or byte).
- The ENCO instruction is supported by GLC2000 Series units only.

4.2.42 DECO (Decode)

A: Data

B: Result to be stored



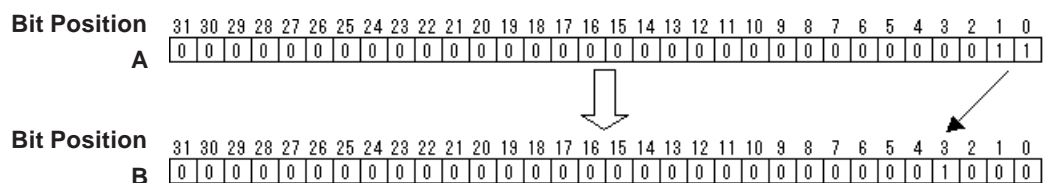
The value entered in A is decoded and output to B. The DECO instruction reads A as a binary value, and the corresponding bit position in B is up. 0 to 31 are available for input.

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.





- 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 3.2.19 – “#Overflow.”

- The DECO instruction does not support variable modifiers (assigned bit, word, or byte).
- The DECO instruction is supported by GLC2000 Series units only.

4.2.43 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 , SUB END, ACT START or ACT END label.

Jumping upward might create an infinite loop.

Execute control to periodically reach the END rung and reset the watchdog timer.

JMP must be the last instruction on a rung.

4.2.44 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

Up to 128 jumps from the subroutine can be used. One subroutine jump uses one stack.

When using the FOR/NEXT instruction , calculate the total size of stacks.



- The number of stacks that can be used in the logic program is 128. Only the FOR/NEXT instruction and the JSR instruction use nests.

Reference See 4.2.46 – “FOR/NEXT Instruction.”

4.2.45 RET (Return Subroutine)

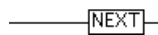
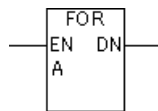


When the RET instruction receives power, control is forced from a subroutine and is returned to its original location . Execution continues from the rung that follows the Jump Subroutine (JSR) instruction.

Because the SUB END instruction returns control when the subroutine is completed, the RET instruction is not always necessary.

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

4.2.46 FOR/NEXT (Repeat)



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

If A is equal to or less than 0, the logic program flow 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:

A
Integer
Integer Array
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.

If the instruction exceeds more than 64 nests, a major error occurs and error code “4” is displayed in #FaultCode. Two stacks are used for one nesting. When using the JSR instruction, calculate the total size of stacks.



Note:

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

For information about #FaultCode error codes, refer to 3.2.25 – "#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 3.2.27 – "#WatchdogTime."*

- The number of nests that can be used in a logic program is 128. Only the FOR/NEXT instruction and the JSR instruction use nests.

Reference *See 4.2.44 – "JSR instruction."*

- The FOR/NEXT instruction is available only with GLC 2000 Series units.

Memo

5 LS Area Refresh

5.1 LS Area Refresh Overview

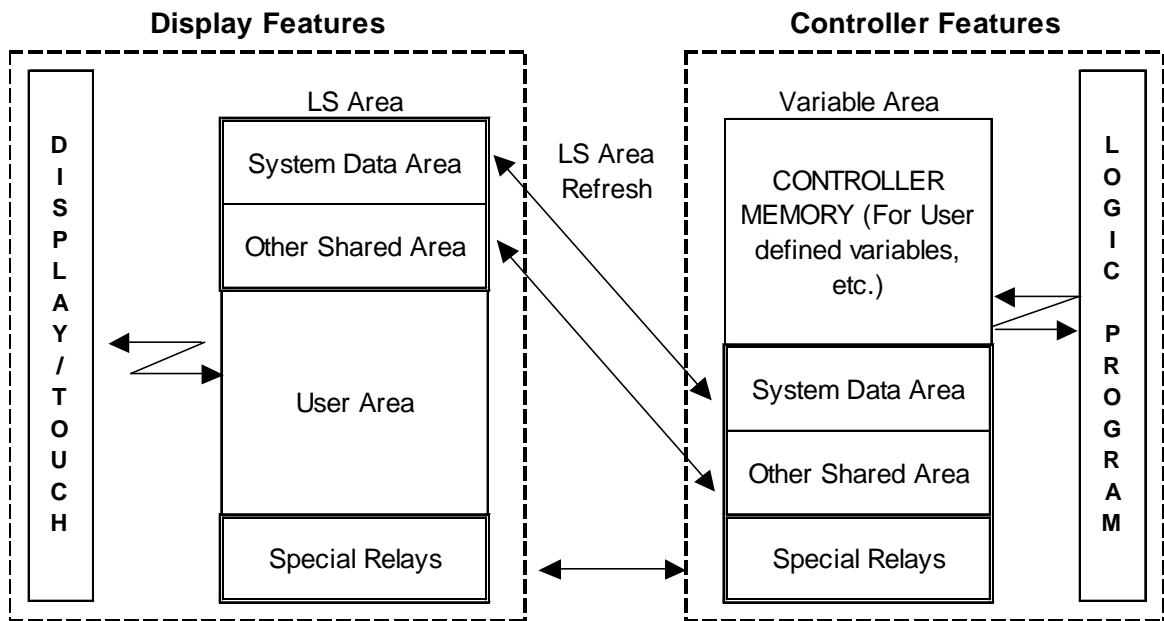
LS AREA REFRESH FEATURE

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, to use the above screen change and buzzer functions with the GLC unit's controller functions (the System Data Area's "mapped" functions), the LS Area must be registered as a variable, with the controller and display features operating via the sharing of LS area data.

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 controller features or display features need to share data.



5.2 LS Area Refresh Settings

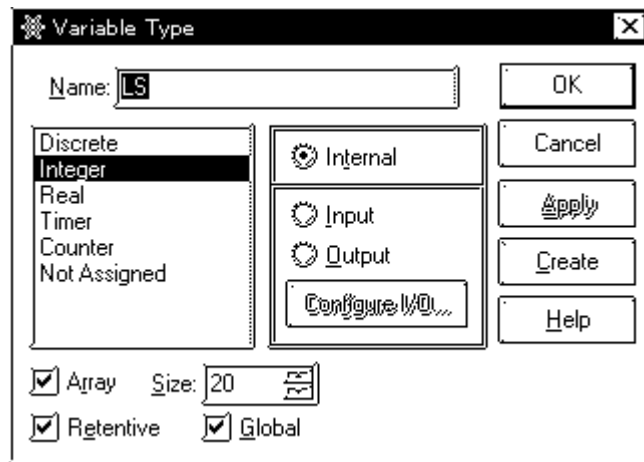
When using the logic program to designate the LS Area, the desired variable must first be registered in Pro-Control Editor. This section describes this procedure.

VARIABLE REGISTRATION

In Pro-Control Editor’s Data menu, click Variable Type to open the Variable Type dialog box.

The variables handled in the LS Area are registered as an internal Integer and array.

In the following example, the size of a System Area array is 20 words, and any additional data to be shared will be added to that amount. Therefore, if the user wants 16 words of data to be shared outside of the System Data Area, the calculation is 16 words of data plus the System Data Area’s 20 words, for a total of 36 words.



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.

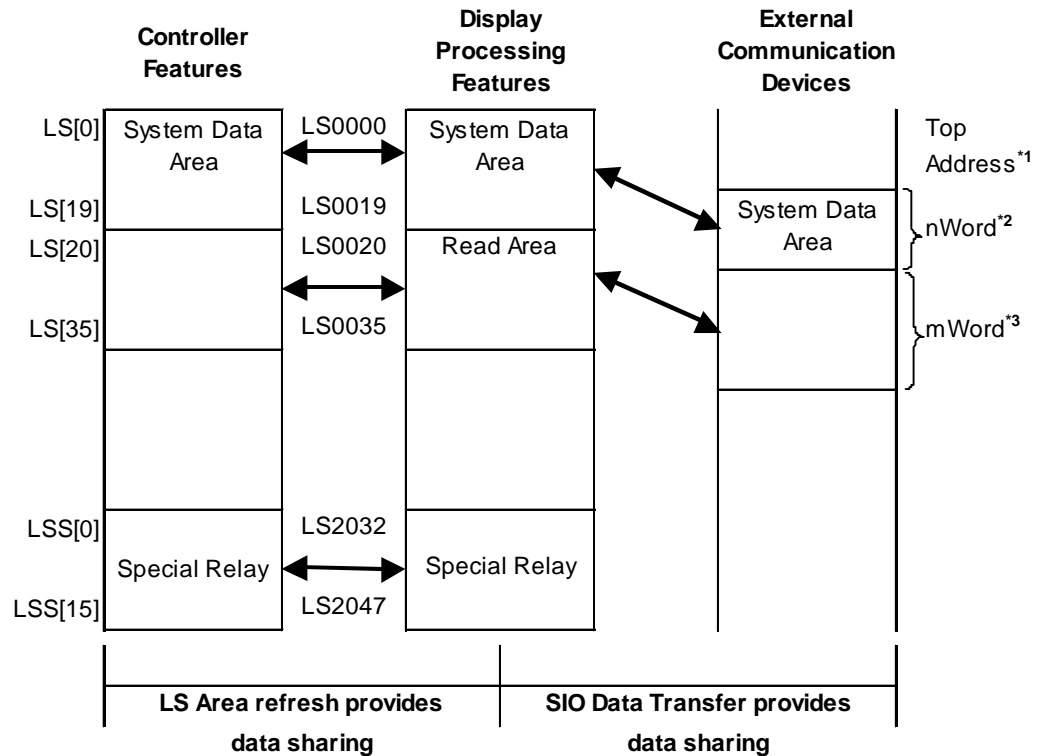
Variable Name*1	Address	LS Address	
LS[0]	0	LS0000	} System Data Area
LS[1]	1	LS0001	
—	—	—	
LS[19]	19	LS0019	
—	—	—	} Other Shared Data
LS[275]	275	LS0275	
LSS[0]	2032	LS2032	} Special Relays
LSS[1]	2033	LS2033	
—	—	—	
LSS[15]	2047	LS2047	

Reference For detailed information about the LS Area and Special Relays, refer to the *Device/PLC Connection Manual*.

1. Names of system variables that are used with the GLC unit’s logic program.

5.3 Sharing Data with External Devices

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 features and the external communication device data register exceeds 16 words, the performance of screen display features may deteriorate.

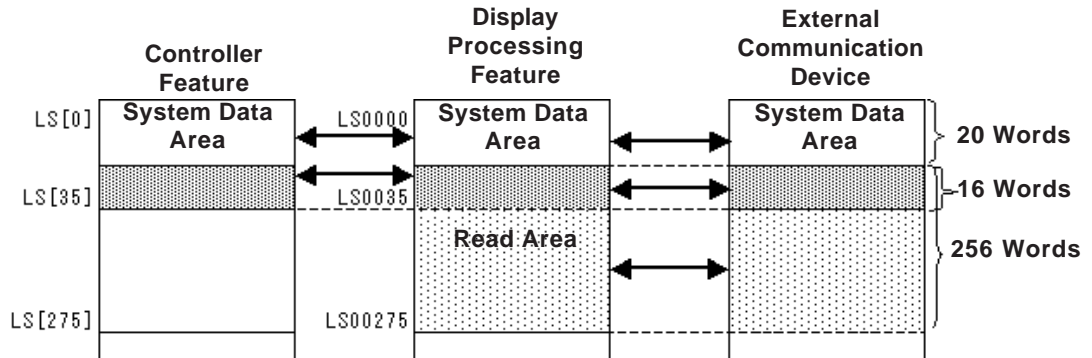


1. Start Address defined in the Initial Settings of the GLC unit.
2. $n = 0$ to 20, depending on the System Data Area setting items selected in the Initial Settings of the GLC unit.
3. $m = 0$ to 16, depending on the size of the Read Area designated in the Initial Settings of the GLC unit.

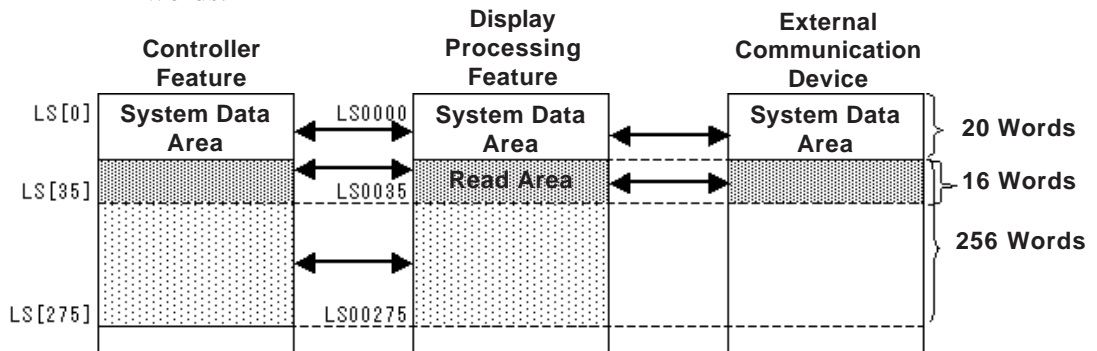
Chapter 5 – LS Area Refresh

To set the Read Area and Variable LS to exceed 16 words, the Read Area can be set to 256 words, and Variable LS can be set to 276 words. A maximum of 16 words is recommended when setting data that is shared with the controller, display processing features, and external communication device.

Example: When the Variable LS size is set to 36 words and the Read Area is set to 256 words.



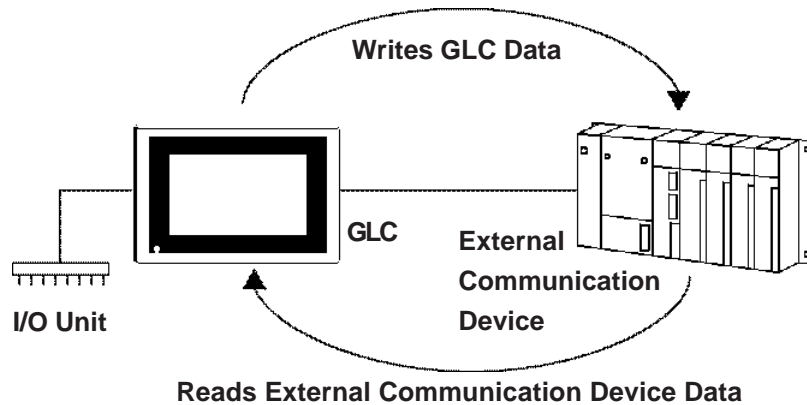
Example: When the Variable LS is set to 276 words and the read area is set to 16 words.



- When the controller's logic program, tags used to update the Display Processing feature, and the logic program from an external I/O unit attempt to change the same variable at same time, priority is determined by the timing.
- When writing data to the Read Area in the GLC, be sure that data written from tag setup and data written from the controller's logic program do NOT overlap or conflict.

**Note:**

When the Read Area is used efficiently and the GLC and external communication device share data, the GLC can be used as the external device's slave device, which also allows the use of an FA type POP unit, or an I/O data collection unit.



5.3.1 LS Area Refresh Cautions

Use the LS Area Refresh feature to control the system area using the controller feature or 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, rather than refreshing the data in addresses LS000 to LS0035 and LS2032 to LS2047 intermittently via the controller feature.

If the frequency of the LS Area's data refresh is increased, the LS Area Refresh may not be executed within one scan. Be aware that errors, such as an External Communication Device communication error, may occur.

Because the Variable LS — an Integer variable — is 32 bits in length, when the System Data Area is 16 bits in length, only the lower 16 bits are available.

Memo

6 I/O Drivers

6.1 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, 2.11 – “I/O Configuration.”***

The following table lists the GLC-supported drivers:

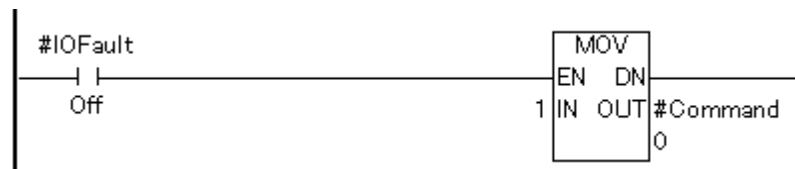
Models	Supported Drivers
GLC100 Series	DIO Driver
GLC300 Series	Flex Network Driver
GLC2300 Series	Flex Network Driver
GLC2400 Series	
GLC2600 Series	



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 3.2.18 – “#IOFault” and 3.2.20 – “#Command.”*

6.2 Flex Network I/F 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, be sure to confirm that the Flex Network I/F unit is attached to the back of your GLC unit. The Flex Network I/F unit is equipped with GLC2300, GLC2400, and GLC2600.

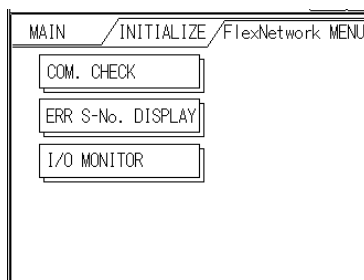
Reference To return to the GLC unit's OFFLINE mode, refer to the GLC unit's user manual (sold separately).

6.2.1 Flex Network I/F Unit Self-Diagnosis

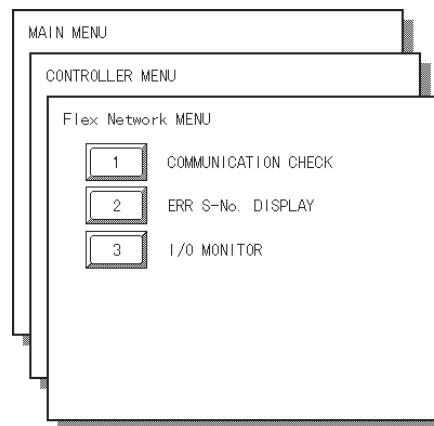
Select FLEX NETWORK DRIVER in the GLC OFFLINE mode's CONTROLLER MENU. The following FLEX NETWORK DRIVER MENU window will then appear.

To select communication check:

GLC100/GLC2300 Series



GLC300/GLC2400/GLC2600 Series



When the 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 consider this when changing to either the OFFLINE or RESET mode.

GLC Condition	→		
	RUN	OFFLINE	RUN
Analog Output	Output from Logic Program	No Analog Output	Output from Logic Program
I/O Signal	Output from Logic Program	No Analog Output	Output from Logic Program
No Analog Output			



The RESET mode's I/O signal OFF timing is NOT fixed.

Here, 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 SETTINGS window will appear.
2. Set Communication Speed to either 6 or 12. Setting the communication speed faster may cause the unit to be easily influenced by noise. Normally, set this speed to 6Mbps.

GLC100/GLC2300 Series

COM. CHECK SETUP NEXT ESC

TRANSFER SPEED (Mbps)

When this test is performed, all connected I/O unit S-No.s are reverse color. When wiring the I/O units, be sure all S-No.s use original settings and are unique.

GLC300/GLC2400/GLC2600 Series

COMMUNICATION CHECK SETUP NEXT CANCEL

TRANSFER SPEED (Mbps)

When this test is performed, all connected I/O unit S-No.s are reverse color. When wiring the I/O units, be sure all S-No.s use original settings and are unique.

3. Press the NEXT button, and the COMMUNICATION CHECK window will appear.

4. Press the START button to begin the communication check.

The currently connected I/O unit's S-No. will be displayed in reverse color.

GLC100/GLC2300 Series

COM. CHECK SET RET

Total Connected I/O units

Connected S-No.s are reverse color.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	

GLC300/GLC2400/GLC2600 Series

COMMUNICATION CHECK START RETURN

Total connected I/O units

Connected S-No.s are reverse color.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	

5. To return to the FLEX NETWORK MENU window, press the RET button.

Chapter 6 – I/O Drivers

To Select Error S-No. Display

When the Error Code No. 841 occurs while the 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 6.4.3 – “Flex Network I/F Unit Troubleshooting.”

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.

GLC100/GLC2300 Series

ERR S-No. DISPLAY										RET
Error S-No.s are reverse color.										

GLC300/GLC2400/GLC2600 Series

ERR S-No. DISPLAY										RETURN
Error S-No.s are reverse color.										

6.2.2 I/O Monitor (I/O Connection Check)

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.

I/O Monitor Settings (when VARIABLE TYPE is set to DISCRETE)

GLC100/GLC2300 Series	GLC300/GLC2400/GLC2600 Series
<div style="border: 1px solid black; padding: 5px;"> I/O MONITOR SETUP <input type="button" value="NEXT"/> <input type="button" value="ESC"/> TRANSFER SPEED (Mbps) <input type="text" value="6"/> S-No. <input type="text" value="1"/> MODEL CODE <input type="text" value="X16TS11"/> VARIABLE TYPE <input type="text" value="DISCRETE"/> </div>	<div style="border: 1px solid black; padding: 5px;"> I/O MONITOR SETUP <input type="button" value="NEXT"/> <input type="button" value="CANCEL"/> TRANSFER SPEED (Mbps) 6 12 S-No. [1] MODEL CODE (FN-) X16TS Y08RL Y16SK Y16SC XY08TS AD04AH DA04AH VARIABLE TYPE DISCRETE WORD <div style="display: flex; justify-content: space-between;"> <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="3"/> <input type="button" value="4"/> <input type="button" value="5"/> <input type="button" value="6"/> <input type="button" value="7"/> <input type="button" value="8"/> <input type="button" value="9"/> <input type="button" value="0"/> <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="BS"/> </div> </div>

Communication speed

Set TRANSFER SPEED to either 6 or 12 Mbps. Setting the communication speed faster may cause the unit to be easily influenced by noise. Normally, set this speed to 6Mbps.

S-No. (Station no.)

Select S-No. from 1 to 63.

Model

Select from FN-X16TS, FN-XY08TS, FN-Y08RL, FN-Y16SK, FN-Y16SC, FN-AD04AH, and FN-DA04AH.

- When using FN-XY16SK and FN-XY16SC, select FN-X16TS for input, and FN-Y16SKS or FN-Y16SC for output.
- When using FN-X32TS, select FN-XY16TS.

Designate the S-No. set by the I/O unit for the lower 16 bits.

Designate the S-No. for the higher 16 bits by adding 1 to the S-No. set by the I/O unit.

- * FN-XY16SK, FN-XY16SC, and FN-X32TS can be used with the GLC2000 Series unit.

Variable type

Select VARIABLE TYPE from DISCRETE and WORD.

- * Only the Word setting can be used for FN-AD04AH and FN-DA04AH.

3. Press the NEXT button, and the following I/O MONITOR window will appear. This window's items will vary depending on the selected VARIABLE TYPE.

**FN-X16TS/FN-XY08TS/FN-Y08RL/FN-Y16SK/FN-Y16SC/FN-XY16SK/
FN-XY16SC/FN-X32TS>**

I/O Monitor (when VARIABLE TYPE is set to DISCRETE)

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 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/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	↑	↓	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 will vary depending on each I/O unit model. Use each unit within the range of its I/O points, beginning from "0".

When using an input-only I/O unit, use only 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 area.

I/O Monitor (when the VARIABLE TYPE is set to WORD)

The input data will be displayed in the input section, if any. Enter the necessary data in the output section via the ten-key pad. When using the GLC100 and GLC2300 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 Series

I/O MONITOR		S-No. 1		RET
INPUT				
	0	(0-65535)		
OUTPUT				
	0	(0-65535)		OUT

GLC300/GLC2400/GLC2600 Series

I/O MONITOR		S-No. 1						RETURN				
INPUT												
	0	(0-65535)										
OUTPUT												
	0	(0-65535)					OUT					
1	2	3	4	5	6	7	8	9	0	↑	↓	BS
										←	→	

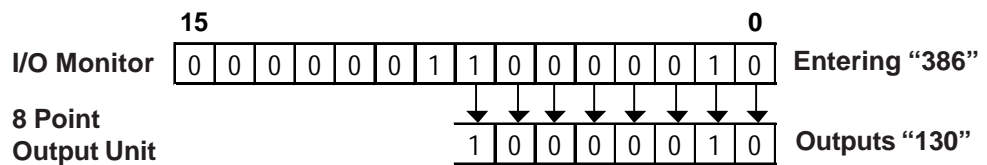


Enter data within the output range, according to the number of the I/O points in each I/O unit.

I/O Points	I/O Range
8	0 to 255
16	0 to 65535

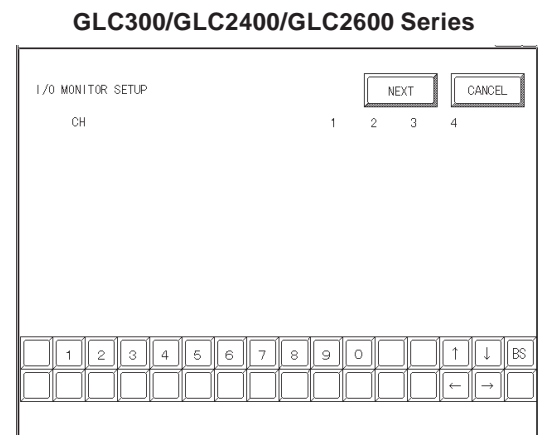
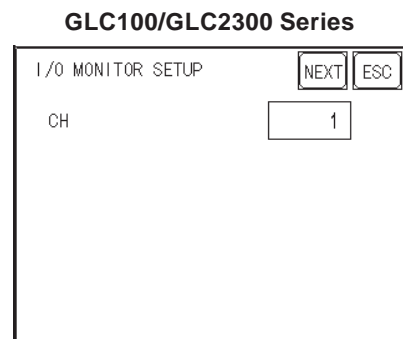


- Data will be output to the I/O unit for the number of I/O points according to the MODEL selected on the I/O MONITOR SETUP window.
- If data that cannot be expressed in the 8-bit system is entered in an 8-point output I/O unit, excess data will be ignored.



FOR FN-AD04AH/FN-DA04AH I/O Monitor (Channel Setting)

The system switches successively through the selectable settings when the channel area is pressed.



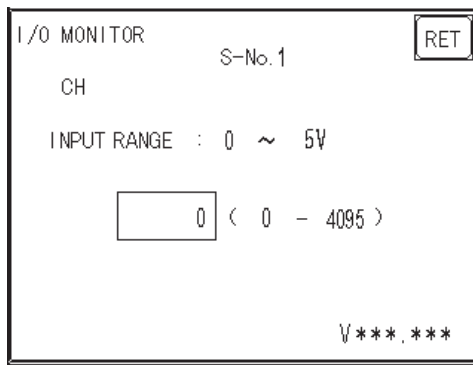
When the NEXT button is pressed, the system switches to the next I/O MONITOR screen. The screen is different for FN-AD04AH and FN-DA04AH.

FOR FN-AD04AH

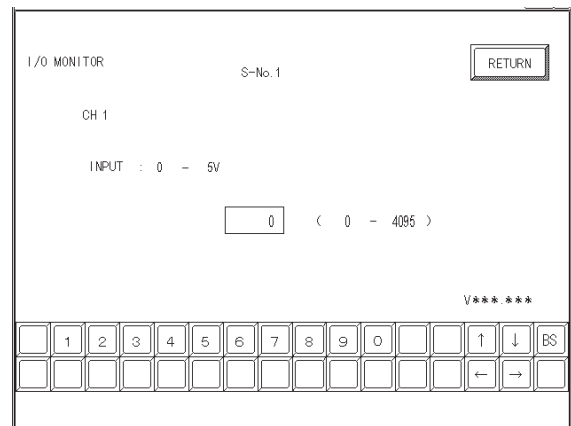
I/O Monitor

This displays input data.

GLC100/GLC2300 Series



GLC300/GLC2400/GLC2600 Series



Pressing the RET(URN) button returns control to the I/O MONITOR screen.

A/D Conversion Table

Input Range Setting	Input Range
0 ~ 5V	0 ~ 4095
1 ~ 5V	0 ~ 4095
0 ~ 10V	0 ~ 4095
-5 ~ 5V	-2048 ~ 2047
-10 ~ 10V	-2048 ~ 2047
0 ~ 20mA	0 ~ 4095
4 ~ 20mA	0 ~ 4095



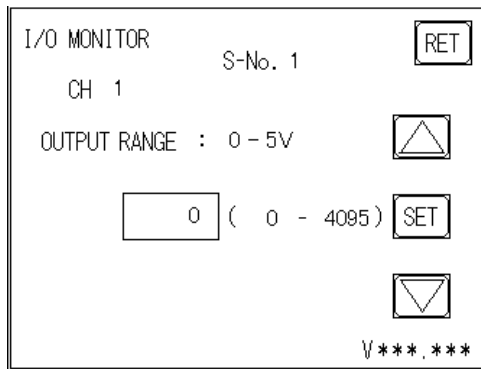
- **Settings other than maximum/minimum, A/D conversion sample count, and the file type operate with the set content stored on the I/O unit side. To change the settings saved on the I/O unit side, change the settings in Pro-Control Editor and download the logic program to the GLC. The logic program will then be set to RUN mode, and the settings will be enabled.**
- **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.**
- **The settings of the range changeover switch on the I/O unit side are read in when the 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

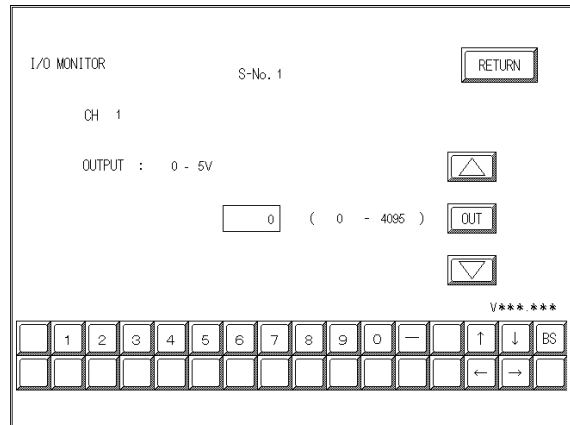
I/O Monitor

Enter data with the keypad. With the GLC100 and GLC2300 Series unit, 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.

GLC100/GLC2300 Series



GLC300/GLC2400/GLC2600 Series



- 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.

D/A Conversion Table

Input Range Setting	Input Range
0 ~ 5V	0 ~ 4095
1 ~ 5V	0 ~ 4095
0 ~ 10V	0 ~ 4095
-5 ~ 5V	-2048 ~ 2047
-10 ~ 10V	-2048 ~ 2047
0 ~ 20mA	0 ~ 4095
4 ~ 20mA	0 ~ 4095



- 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.
- The settings of the range changeover switch on the I/O unit side are read in when the 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.

6.2.3 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 unit's Users Manual.

ERROR CODES

I/O errors include those occurring during writing and reading. When one of these errors occurs, the controller writes an error code to #IOStatus.

Setting Errors

Error Code	Contents	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, possible 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, possible 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	Contents	Solution
821	There is no Flex Network unit attached.	The ID number read from the Flex Network unit is not correct. Occurs when the unit is not attached.
822	Initial Error. Initialization failed to synchronize the Flex Network I/F unit and the unit's driver.	A hardware error may have occurred. Reference For details, refer to the Flex Network unit's user manual. Please contact your local Pro-Face distributor.
823	Analog unit setting error	Check to see if the communication line is disconnected, power is not supplied to the I/O unit, or the I/O unit is malfunctioning.

Runtime Errors

Error Code	Contents	Solution
841	There is an I/O unit error (loose connector, malfunction, etc.)	Check all related wiring. Reference 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)	This is likely due to disconnection in the output signal line. Check the output signal line of the sensor.
843	Error in the high-speed counter unit	The High-Speed Counter unit detected an error. Reference 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. Reference 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.

Internal Errors

Error Code	Contents	Solution
850 ... 859	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. Reference Write down the error code and refer to your GLC User Manual. Contact your local Pro-face distributor.

6.3 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.

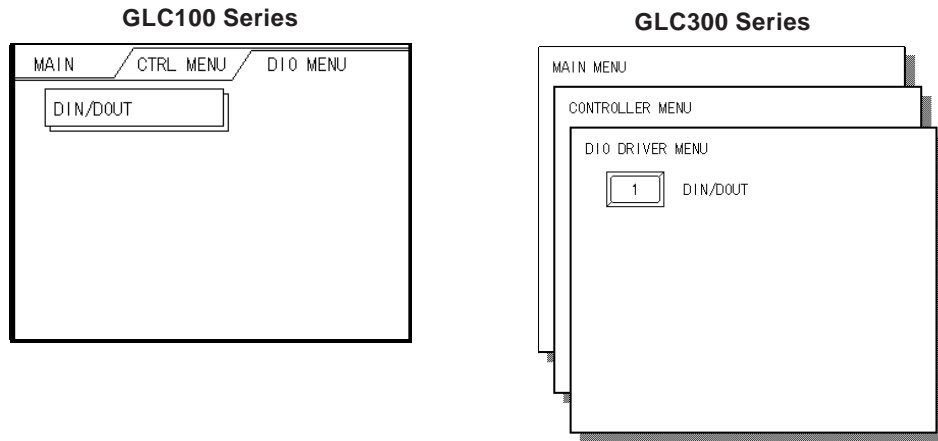
Reference For instructions on how to move to the OFFLINE mode screen, refer to the *GLC Series User Manual* (sold separately).

6.3.1 DIO Unit Self-Diagnosis

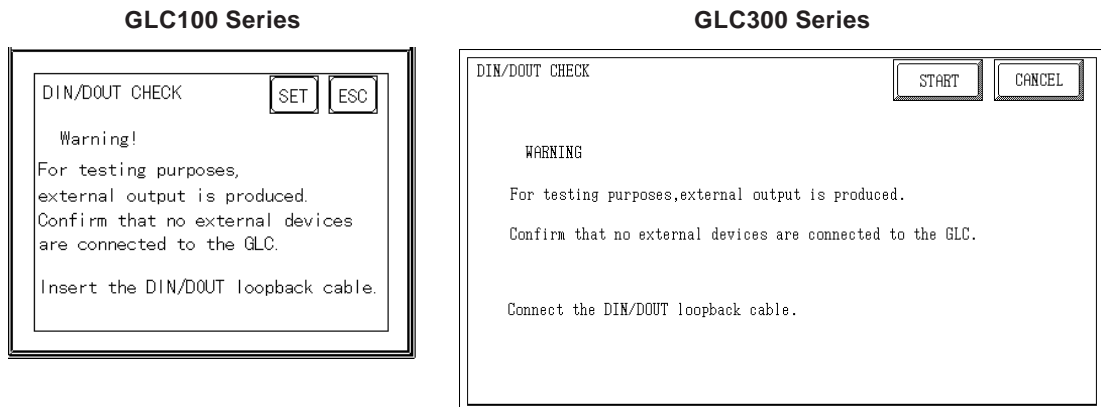
This section explains how to use the DIO unit’s Self-Diagnosis feature.

Reference For detailed information, refer to the *GLC Series User Manual* (sold separately).

1. Touch the OFFLINE screen’s Controller Menu to open the DIO Menu area.



2. Touch the DIN/DOUT key to open the following screen.

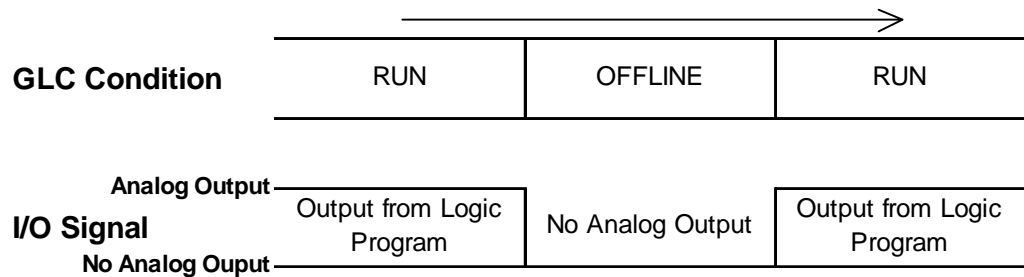


3. Touch either the Set or Start key to start the self-diagnosis.

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.

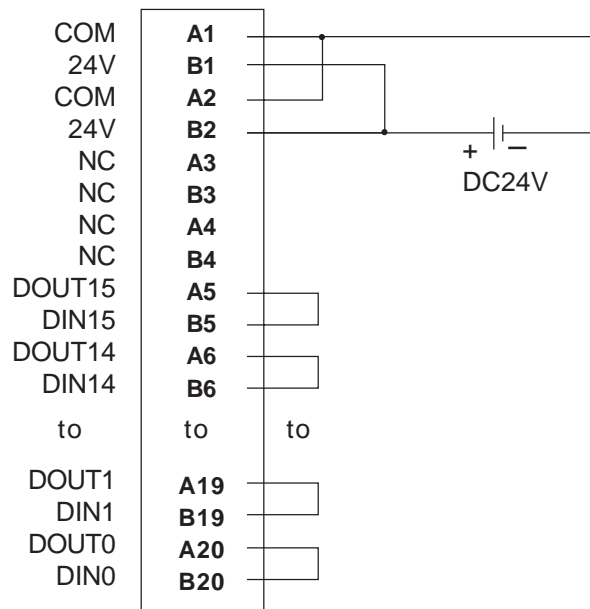


When switching to the OFFLINE mode or when resetting from the logic program's RUN state, the I/O signal may turn to OFF. Be aware of the possibility that the I/O signal will turn OFF.



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)
Crimped Type	Fujitsu	FCN-363J040 FCN-363J-AU/S FCN-360C040-B
Terminal Block Unit Type	Mitsubishi	A6TBX36 (Terminal Block) AC**TB (Cable) (** = cable length)
	Yokogawa	TA40-ON

6.3.2 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 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 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 a GLC100 series unit, 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 the decimal format.

6.3.3 Troubleshooting

This area explains how to solve possible DIO unit problems.

DIO UNIT INPUT ERRORS

Error Type	Possible Cause	Solution
Input monitor lamp is ON, but no input can be performed.	DIO Unit is defective.	Replace DIO Unit
	Enable I/O is not selected.	Select Enable I/O.
	Program is incorrect	Correct program
Input monitor lamp is OFF and no input can be performed.	DIO Unit is defective	Replace DIO Unit
	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.
	DIO unit is not correctly attached.	Attach the DIO unit securely.
	Connector is not securely attached.	Attach the connector securely.
All input lines do not turn OFF	DIO Unit is defective	Replace DIO Unit
Designated Input lines do not turn ON.	DIO Unit is defective	Replace DIO Unit
	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	Replace DIO Unit
	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	Replace DIO unit
	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	Replace DIO unit
	Program is incorrect Output area is completely OFF.	Correct program.
	[Use I/O] box is not selected.	Set the [Use I/O].
	DIO unit is not correctly attached.	Attach the DIO unit securely.
Output lines do not turn OFF	DIO unit is defective	Replace DIO unit
Designated output lines do not turn ON	DIO unit is defective	Replace DIO unit
	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	Replace DIO unit
	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 mis-operation.	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 explanation of possible error causes and solutions for when the DIO unit is attached to the GLC.

Setting Errors

Error Code	Contents	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 WLL 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".

Initialization Errors

Error Code	Contents	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.

Runtime Errors

Error Code	Contents	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.
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.
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.
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.

Internal Errors

Error Code	Contents	Solution
850 ... 864	Driver Error A major system error has occurred.	Record the Error Number and contact your local Pro-face distributor.

7 Error Messages

7.1 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 logic program file has not been downloaded to the GLC or the GLC unit's logic program file is damaged.	Download another copy of the project file from Pro-Control Editor.
"Fatal Error: Drive Check Failed"	The GLC unit's current I/O driver is incorrect.	Check that the I/O driver designated 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 distributor.
"Can't Set Priority"	The GLC unit's system file is incorrect. The file may have been damaged during downloading.	Check that the model type set in the GP-PRO/PB III software file is a "GLC" type, and re-transmit the project file.
"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 distributor.
"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, the Constant Scan Time (program) should be changed.
"Bad Var: xxx"	Unable to find variable "XXX". Either the logic program file has not been downloaded, or the GP-PRO/ PB III is using a variable that does not exist in the logic program file.	Download the project file again.

Chapter 7 – Errors

Error Message	Cause	Solution
"Bad Array: xxx"	The number of elements used in the GP-PRO/PB III array variables and those used in the logic program file's array variables are different.	After saving the logic program file, download the project file again to the GLC.
"Bad Type xxx"	The GLC variable "XXX"'s type is different from the GP-PRO/PB III variable type.	After saving the logic program file, download the project file again to the GLC.
"Unknown register type"	This variable type does not exist.	After saving the logic program file, download the project file again to the GLC.
"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 Tags or Parts are being used.	Reduce the number of Tags or Parts and then download the project to the GLC again.
"Logic Program is Empty"	The logic program file has not been downloaded to the GLC, or the logic program file in the GLC (FEPROM) is damaged. (GLC2000 Series only)	Download the logic program file again from Pro-Control Editor.
"No backup logic program in FEPROM"	The project file has not been copied to FEPROM after online editing. This is a warning message. (GLC2000 Series only)	Copy the project file to FEPROM using GLC offline menu.
"SRAM checksum error"	WLL 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 FEPROM. Using online edit, check that no changes have been made in the logic program.
"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.	Set up the GLC unit again with GP-PRO/PB III 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.
"Exception 137 [xxxx : xxx]"	Incompatible I/O has been set.	Check the I/O configuration and re-allocate the I/O.

7.2 Error Codes

The following table lists the #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	Major	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 FEPROM will be executed.* ¹

1. This error occurs only with GLC2000 Series units.



Note: Major Faults and Minor Faults

- When a major error occurs, the controller immediately stops executing the logic program.
- When a minor error occurs, the controller is able to continue executing the logic program.
- Check the cause of the error.

7.3 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. ▼Reference▼ 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. ▼Reference▼ 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. ▼Reference▼ 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. ▼Reference▼ See 3.2.25 – "#FaultCode," and 3.2.29 – "#Command."
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 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 the 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 *Pro-Control Editor Operation Manual, 3.1 – "Controller Configuration"* and 3.2 – "Starting and Stopping the Controller."

Index

A

- A/D Conversion Table 6–8
- Arrays 2–3
 - Accessing 2–7
 - Block Transferring 4–17
 - Defining 2–6
 - Fill Transferring 4–19
 - Indirectly accessing 2–9
 - Shifting right 4–25
- Available Memory for Variable Storage 2–3

B

- BCD/BIN Conversion
 - Errors 3–13
 - Instructions 4–42
- Bit Access Method 2–8
- Bit Operation Instructions 4–1
- Bit Positions
 - Rotating left 4–20
 - Rotating right 4–21
 - Shifting left 4–22
 - Shifting right 4–23
- Block Transferring Arrays 4–17
- Byte Access Method 2–8

C

- Channel Setting (I/O Monitor) 6–7
- Coil-Type Output Instructions 4–7, 4–8, 4–10
- Communication Check Procedure 6–3
- Connectors, recommended 6–13
- Constant Scan Time Mode 1–1, 1–4, 1–5
- Controller
 - Current condition 3–1
 - Data, shared 5–1
 - Fault conditions 3–8
 - Fault status history 3–8
 - Features 1–1, 5–1
 - Operating status 3–8
- Conversion, Real-to-Integer 4–17
- Convert Instructions 4–4
- Copyright 1
- Counter Data Variables 2–3, 2–4
- Counter Instructions 4–3

D

- D/A Conversion Table 6–9
- Damages or Third-Party Claims 1
- Data Watch List 3–5, 3–6
- Device Address 2–1
- Device Allocation Table Layout Sheet 7
- Digital Electronics Corporation 1, 7, 2–1
- DIN/DOOUT Loopback Cable 6–13

- DIO Unit 6–12
 - Input errors 6–15
 - Output errors 6–16
- Discrete Arrays, accessing 2–6
- Discrete Variable Setting (I/O Monitor) 6–6
- Discrete Variables 2–3
- Disk Media Usage Precautions 8
- Display Area Data, shared 5–1
- Display Features 1–1, 2–4, 5–1
- Divide by Zero Errors 3–13

E

- Enable I/O 7–4
- Error Codes
 - DIO unit
 - initialization errors 6–18
 - internal errors 6–18
 - Runtime errors 6–18
 - setting errors 6–17
 - FaultCode errors 7–3
 - Flex Network I/O unit
 - initialization errors 6–11
 - internal errors 6–11
 - Runtime errors 6–11
 - setting errors 6–10
 - Troubleshooting 6–10
- Error Messages, Pro-Control 7–1
- Error Process Subroutine 3–13
- Error S-No. Display 6–4
- Errors
 - #FaultCode 7–3
 - BCD/BIN conversion 4–42
 - Divide by Zero 3–13
 - Divide by zero 3–13
 - External Communication Device communication 5–5
 - Intermittent 3–8
 - Pro-Control Editor 7–4
- External Communication Device Data, shared 5–3, 5–5

F

- Fault Flags 3–9
- Fault Status History 3–8
- Faults 7–3
- Fill Transferring Integer Arrays 4–19
- Flex Network
 - Communication check 6–2
 - Communication speed 6–3
 - Driver menu 6–2
 - I/O unit errors 6–10
 - Maximum input/output points 6–6

Index

Floating Decimal Point Values 2–4
Floating-Point Instruction 4–26, 4–27, 4–28
FOR/NEXT Restrictions 4–46
Forced Variables 3–5
Foreign Regulations 1

G

GLC
 Available memory limit 2–3
 Clock data variables 3–2
 Scan time 1–1
GP-PRO/PB III C-Pack01 7
Graphic Processing Time 1–5, 1–6

I

I/O Connection Check 6–5, 6–14
I/O Driver Status 3–6
I/O Errors 6–1
 DIO unit 6–17
 Flex Network 6–10
I/O Monitor Settings 6–5
I/O Points, Maximum Available 6–6
I/O Read / I/O Write 1–3
I/O Watchdog Timeout 1–3
Infinite Loops 4–45
Initialization Errors 6–11
 DIO unit 6–18
 Flex Network I/F unit 6–11
Input Data Display (I/O Monitor) 6–8
Input Terminal (S-No.) 6–14
Input-Only I/O Unit 6–6
Instruction Overflows 3–13
Instructions
 Bit operation 4–1
 Convert 4–4
 Counter 4–3
 Floating-point 4–26, 4–27, 4–28
 Mathematical 4–2, 4–3
 Movement 4–2
 Timer 4–3
Integer Arrays, accessing 2–7
Integer Variables 2–3
Intellectual Properties 1
Internal Clock 3–4
Internal Errors
 DIO unit 6–18
 Flex Network I/F unit 6–11

K

Keypad Data Entry (I/O Monitor) 6–9

L

Ladder Circuit Execution Time 1–1
Latch Fault Flag 3–8
Layout Sheets
 Device Allocation Table 7
 Microsoft Excel data format 7
 Tag Layout Sheet 7

Liability 1
Logic Program
 Execution time 1–5
 Features 1–2
Loopback Cable, creating 6–5, 6–6, 6–13
LS Area Refresh 5–1
LSS Area 5–2

M

Mathematical Faults 3–13
Mathematical Instructions 4–2
Memory, Variable Storage 2–3
Microsoft Excel Data Format 7
Minor Faults 3–13, 3–15
Mitsubishi 2–1
Models (I/O Monitor) 6–5
Movement Instructions 4–2

N

Negative Transition Contact 4–12
Nests 4–45, 4–46

O

OFFLINE Mode 1–1, 6–2
Omron 2–1
Operating Status, Controller 3–8
Operation Mode 1–1
Output-Only I/O Unit 6–6
Overflows
 Instruction 3–13
 Real-to-Integer conversion 3–13

P

Percent Scan Time Mode 1–1, 1–4, 1–6
PLC Device Address 2–1
PLC Manufacturers 2–1
Positive Transition Contact 4–11
Precautions and Warnings 8
Pro-Control Editor 2–1
 Error messages 7–1
 Program errors 7–4
Programming Mode 1–3

R

Range Changeover Switch 6–8, 6–9
Read Area 5–3
ReadMe.txt 1, 7
Real Arrays, accessing 2–9
Real Values, comparing 4–30, 4–31, 4–32, 4–33
Real Variables 2–3, 2–4
Real-to-Integer Conversion 3–13, 4–17
Real-to-Integer Conversion Overflows 3–13
Registered Trademarks 6
RESET Mode 6–2
Retentive / Non-retentive Variables 1–3, 4–1
Rollover 3–7
RUN Mode 1–1

Runtime Errors
DIO unit 6–18
Flex Network I/F unit 6–11

S

S-No. (I/O Monitor) 6–5
S-No. (Input Terminal) 6–14
Safety Symbols and Terminology 9
Scan Time Adjustment 1–4
Screen Display Features 5–3
Screen Processing Time 1–1
Setting Errors
DIO unit 6–17
Flex Network I/F unit 6–10
Shared Data 5–1
SIO Communication Time 1–1
Special Relay Area 5–2
Stacks 4–45, 4–46
STOP Mode 1–1
System Data Area 3–2, 5–1, 5–2, 5–3
System Variables 1–3, 3–1, 3–3

T

Tag Layout Sheet 7
Third-Party Claims or Damages 1
Timer Data Variables 2–3, 2–4
Timer Instructions 4–3
Touch Panel Processing Time 1–1
Trademarks, Registered 6
Transfer Speed (I/O Monitor) 6–5
Troubleshooting
DIO unit input errors 6–15
DIO unit output errors 6–16
Flex Network I/O errors 6–10

V

Variable Storage Area 2–3
Variables 2–1, 6–5
Accessing 2–6
Clearing 4–17
Counter data 2–3
Forced 3–5
GLC clock data 3–2
Naming 2–1
Non-retentive 1–3
Registration 5–2
System 1–3
Timer 2–3

W

Warnings and Precautions 8
Watchdog Timeout 1–3
Watchdog Timer 4–47
Word Access Method 2–8
Word Variable Setting (I/O Monitor) 6–6

Memo