# PREFACE

Thank you for purchasing Pro-face's ladder logic programing software "Pro-Control Editor Ver. 4.0" for use with GLC series units.

To ensure correct use of this product, be sure to read the included manuals 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 the "Pro-Control Editor Ver. 4.0" (hereinafter referred to as "this product") software are reserved by the Digital Electronics Corporation. Digital grants the use of this product to its users as described in the "Software Lisence Conditions". Any actions violating the above-mentioned conditions are 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, contact your local representative.

(3) Please be aware that Digital Electronics Corporation shall not be held liable by the user for any damages, losses, or third party claims arising from the uses of this product..

(4) Differences may occur between the descriptions found in this manual and the actual functioning of this product. Therefore, the latest information on this product is provided in data files (i.e. Readme.txt files, etc.) and/or separate documents. Please consult 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.

(6) Please be aware that Digital Electronics Corporation shall not be held liable by the user for any damages, losses, or third party claims arising from the use of this product.



Digital Electronics Corporation December 2001

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

# 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 omits individual descriptions of each of these rights.

| Trademark / Tradename | Right Holder |
|---|---|
| Microsoft, MS, MS-DOS, Windows, Windows 95, Windows 98, Windows Me, Windows NT, Windows 2000, Window XP, Windows Explorer, Microsoft Excel | Microsoft, U.S. |
| Intel, Pentium | Intel, U.S. |
| Pro-face, Flex Network | Digital Electronics Corporation (in Japan and other countries) |
| Ethernet | Western Digital, U.S. |
| IBM compatible | IBM, U.S. |
| Adobe, Acrobat | Adobe Systems Incorporated |

The following terms used in this manual differ from the above mentioned formal trade names and trademarks.

| 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 NT | Microsoft® Windows NT® Operating System |
| Windows Me | Microsoft® Windows Me® Operating System |
| Windows 2000 | Microsoft® Windows 2000® Operating System |
| Windows XP | Microsoft® Windows XP® Operating System |
| Acrobat Reader | Adobe® Acrobat® Reader |

# LIST OF SUPPORTED MODELS

The following table lists the models compatible with Pro-Control Editor Ver.4.0. The following series names or product names are used in the descriptions contained in this manual. The "GP Type" listed in the table below refers to the name to be selected via 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-SC41-24V | 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-Package01 Manual comprises seven individual volumes. For the contents of each volume, refer to the table below. The manual is also available on CD-ROM 2 in PDF format. (The "Installation Guide" is not included as a PDF file.) Supplemental explanations and additional and revised information on functions may be provided as data files. To read the data files, click the [Start] button, and choose [Programs]-[Pro-face]-[ProPB3 C-Package]. Click [Read me] and read the contents thoroughly.

For detailed descriptions of hardware manufactured by Digital Electronics Corporation, please read the specific user manual (sold separately) for the applicable device.

| **GP-PRO/PB III C-Package01** | |
|---|---|
| Installation Guide | Describes the installation and basic operation procedures. |
| **Pro-Control Editor Ver. 4.0** | |
| User Manual | Describes the software settings for combining with the GLC, variables, and instructions. |
| Operation Manual (this manual) | Provides exercises for learning the basic functions from installation to operation and a list of error messages. Describes procedures using the variables registered by the Pro-Control Editor for use by the GP-PRO/PB III. |
| **GP-PRO/PB III for Windows Ver. 6.0** | |
| Operation Manual | Describes the installation, operating procedures and software functions for using the GP screen creation software. (PDF manual) |
| Tag Reference Manual | Explains "tags" for specifying on-screen functions of the GP. |
| Parts List | Describes the parts and symbols provided in the software for creating GP screens. |
| Device/PLC Connection Manual | Describes procedures for connecting the GP to PLCs, temperature controllers and inverters of other manufacturers. |

For your convenience, a layout sheet is installed as part of the standard installation. Use this layout sheet for specifying the PLC registers when setting the tag addresses. The layout sheet comprises two files: "List of Device Assignments" and "Tag Layout Sheet." The two files are installed separately as Microsoft® Excel files. The locations and file names of each file are listed on the next page.

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

---

* *The GP-PRO/PB III Manual describes the procedures for developing GP screens. The steps for developing GLC screens are identical; simply substitute "GLC" for "GP."*

* *As a supplement to the manuals listed above, detailed explanations are available in the on-line help.*

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

Adobe® Acrobat® Reader is required to view the manuals contained in PDF file format on CD-ROM.

# TABLE OF CONTENTS

# MANUAL SYMBOLS AND TERMINOLOGY

This manual uses the following symbols and terminology.
If you have any questions about the contents of this manual, please contact your local Pro-face distributor. Also, if you have any question about your personal computer or the Windows® software, please contact your local distributor or manufacturer.

## ■ Safety Symbols and Terms

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

| Symbol | Description |
|---|---|
|   | Incorrect operation resulting from negligence of this instruction may cause death or serious injury. |
|   | Incorrect operation resulting from negligence of this instruction may cause personal injury or damage to equipment. |
|   | Failure to observe this instruction may cause abnormal operation of equipment or data loss. |
|   | Instructions / procedures that must be performed to ensure correct product use. |
|   | Actions / procedures that should not be performed. |

## ■ General Information Symbols and Terms

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

| Symbol | Description |
|---|---|
|   | Provides hints on correct use, or supplementary information. |
| Reference | Indicates related (manual name, page number) information. |
| *1, *2, (etc.) | Indicates related supplemental information. |
| Pro-Control Editor | Software for editing, transferring, and monitoring a GLC unit's ladder logic program. |
| Controller | The control function of a GLC unit. |
| GP-PRO/PBIII (screen creation software) | The screen creation software GP-PRO/PBIII for Windows Ver. 6.0. |
| GLC | Indicates the "GLC series" of graphic logic controllers manufactured by the Digital Electronics Corporation. |
| External Communication Device | Indicates peripheral devices including PLCs (programmable logic controller), temperature controllers, and inverters. Note that devices connected through Flex Network and DIO are not included. |

## ■ Keyboard Compatibility List

These keys may vary depending on the type of personal computer keyboard you are using.

This manual uses the following symbols to indicate a personal computer's keys:

| Type / Symbol | IBM Compatible 101 keyborad |
|---|---|
| Esc | Esc |
| Tab | Tab ⇆ |
| Ctrl | Ctrl |
| Shift | ⬆ Shift |
| Alt | Alt |
| Delete | Delete |
| Back space | Back Space |

## ■ Typical System Configuration

This manual describes this software's operating procedures and functions based on the typical PC system configuration shown below.

If you use a different system configuration from this one, the screen shown on your PC, as well as various item names may be different. In this case, substitute a functionally equivalent item for the one(s) shown here.

| Item | Specification | Remarks |
|---|---|---|
| Personal Computer | PC/AT compatible machine with Pentium processor | |
| Memory | 32 MB | |
| Mouse | Windows 95 compatible mouse | |
| OS | Windows 95 | |
| GLC | GLC 2300 Series | |
| GLC Connection Cable | RS-232C | Digital's GPW-CB02 cable is required. |

# PRECAUTIONS

## ■ Product Usage Precautions

To prevent program malfunctions or accidents, be sure to observe the following:


- **Example circuits and applications shown in this manual are only for your reference. Please be sure to check if all units and system equipment are operating correctly and safely before using.**
- **Digital Electronics Corporation does not assume the use of this product for applications requiring extremely high degrees of reliability and safety, such as with equipment or systems for transportation, moving, medicine, aerospace, nuclear, or for undersea data communication. Do not use this product for these applications.**
- **Touch panel switches should NOT be used for a device's Emergency Stop Switch. Generally speaking, all industrial machinery/systems must be equipped with a mechanical, manually operated emergency stop switch. Also, for other kinds of systems, similar mechanical switches must be provided to ensure safe operation of those systems.**
- **When a GLC unit problem would cause a serious or fatal accident, or would seriously damage equipment, please install your own backup or fail-safe*1 system.**


- This product is not designed or manufactured for use in a machine or system that is to be used under circumstances where human life is potentially at risk.
- Do not turn off your PC's power switch during the performance of a program.
- Do not modify the contents of this product's project files using the text editor feature, etc.
- Do not transfer screens to the GLC which contain features the GLC series unit does not support.

*1 Preparations for minimizing defects caused by wrong operation or data errors from sensors/controllers.*

## ■ CD-ROM Usage Precautions


To prevent CD-ROM damage or CD-ROM drive malfunctions, please observe the following instructions:

- Make sure to remove the CD-ROM before turning on and off your PC.
- Do not remove the CD-ROM disk from the CD-ROM drive while the drive's operation lamp is lit.
- Do not touch the CD-ROM disk's recording surface.
- Do not place CD-ROMs in a place where they may be exposed to extremely high or low temperatures, high humidity or dust.
- Do not place floppy disks near stereo speakers, TVs, or magnetic therapy equipment.

## ■ Product Restrictions

This product has the following restrictions:


- **GLC100/300 series units do not support the Pro-Server with Pro-Studio for Windows (2-way Driver) software.**
- **The GP-PRO/PBIII software displays screen data using your personal computer's fonts and graphic functions. Therefore, there may be a slight difference between data displayed on your personal computer and the same data displayed on the GLC unit.**
- **GP-PRO/PBIII functions which cannot be used with GP-370 series units (i.e. AUX Output, Inching Tags, t-Tag AUX Output, Backup Function etc.) cannot be used on the GLC100.**
- **The device codes and address codes used to specify indirect addresses for GP-PRO/PBIII E-tags and K-tags cannot be used with the Pro-Control Editor since the Editor is not equipped with the variables associated with these device/address codes.**
- **If the GLC's logic time (scan time) becomes too long, the sampling time designated for the trend graph may not be accurately maintained.**
- **When using arrays with the Pro-Control Editor, do not delete any array-elements in GP-PRO/PBIII.**
- **Only real numbers can be used with the E-tag and K-tag's "Float" function. However, there may be some error due to differences in tag precision with the GLC variable.**
- **GLC variables cannot be used for the trend graph's Block Indirect Display when M to M is selected as the PLC type.**
- **GLC variables are handled using 32 bit-device Low/High order.**
- **With the GLC100, the Q-tag's Sub Display feature cannot be used.**
- **If a GLC's Logic time (scan time) period is too long, sound file reproduction may be interrupted during playback.**
- **When you are designating a bit using an Integer-type Variable, if a T-tag or a W-tag's bit (except the "REVERSE" setting) is written to, all bits will be changed to "0" except for the one that has been designated using an Integer-type Variable.**

- **When placing multiple T-tags used to reverse a bit's action (e.g. ON or OFF) on a (Base) screen, if the same integer variable (i.e. "01") is used to designate the bit position used by more than one of these T-tags, only the T-tag placed last (top-most) will be enabled.**
- **All GLC Retentive Variable data is retained by SRAM backup memory that uses a lithium battery. The battery's back up period lasts approximately 60 days in its initial condition (fully charged), and approximately 6 days when the battery life is almost finished. If you need to back up data for a longer period, you need to either use back up data in your host computer, or configure the Editor system so that the Editor can back up data.**
- **With the GLC2400, AUX can only be used for reset input.**
- **Online editing edits the logic program stored on the SRAM. Though all the data on the SRAM may be lost during battery loss at off-state, backup data will be reloaded from the FEPROM. Be sure to "copy to FEPROM (at off-line menu of GLC)" or backup the logic program as PRW file using the Pro-Control Editor.**
- **Due to differences in PC and GLC Real value accuracy, the values displayed during "Monitoring Mode" may differ.**

# Precautions - Compatibility with Earlier Versions

Please read the following precautions if you are currently using the Pro-Control Editor versions earlier than 3.0.

Logic programs are saved in WLL format with Pro-Control Editor versions earlier than 3.0. However, with Ver. 4.0, logic programs are included in the Project Files of the GP-PRO/PB III and saved in PRW format.

When using logic programs created with versions earlier than 3.0, you are required to import the WLL files to PRW files.

Reference *2.14 Importing/Exporting a Logic Program*

# 1 Pro-Control Editor Fundamentals

## 1.1 About Pro-Control Editor

Pro-Control Editor Ver.4.0 (hereafter referred to as the “Editor”) is a logic programming software for use with GLC Series units.

This software contains many features, such as:

- a GLC DIO unit driver
- a GLC Flex Network I/F unit driver
- a Ladder logic program editor
- Ladder logic program transfer feature
- Cross reference reports
- Monitoring feature
- Online Edit Function*1
- Communication via Ethernet *1

The Editor allows you to develop logic programs compliant with the international standard IEC61131-3 in an easy-to-use Windows environment.
Logic programs created with the Editor function on the GLC unit can be used after being downloaded to the GLC.
The variables created with the Editor can be transferred to the screen creation software GP-PRO/PB III for Windows Ver. 6.0 and used in common with the display functions (switches and lamps) of the GLC.

*1 Supported by GLC2000 Series only.

*MEMO*

# 2 Creating a Program (Introductory Tutorial)

This chapter provides step-by-step instructions on using the Editor to create a logic program in Programming mode.

## 2.1 Before starting the tutorial

Please read the "Installation Guide" and "Pro-Control Editor User's Manual" provided with the Pro-Control Editor before starring the tutorial. After understanding the instructions, variables, and functions used with the GLC, go through the tutorial examples in this Operation Manual to learn the functions and operation procedures.

Each lesson in this chapter describes the operating procedures of the Editor based on tutorial examples.

This section describes how to use the Editor to create a logic program that controls the operation of soft drink machines used in fast food restaurants. The machine features the following functions:

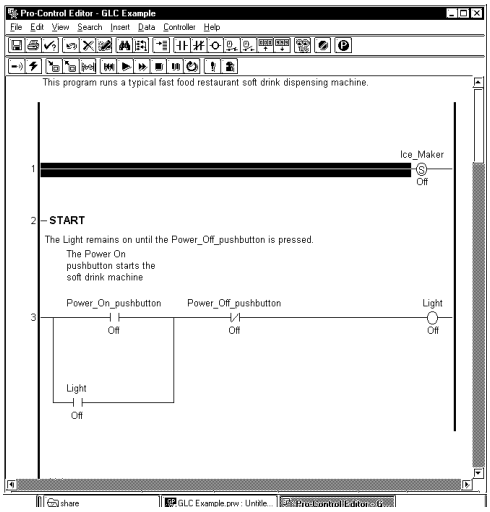
- Pressing the button one time automatically loads a large/medium/small cup.
- Dispenses ice cubes or soda only when a cup is placed under the dispenser.
- Counts the number of cups loaded by the machine after the power is turned on.

### ◆ Examples of Completed Logic Program and Screen

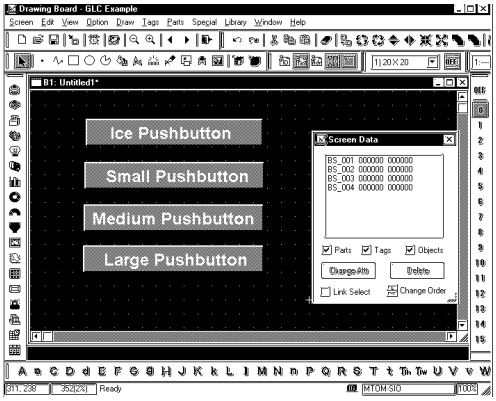
The example logic program and project file used in this lesson can be found in the "Soda.prw" file in the "C:\Program Files\Pro-face\ProPBWin\GLC_Samples" folder.

Refer to this file if you have problems with the procedure or for practicing search operation. Please refer to the on-line help for a detailed description of the features of the Editor.

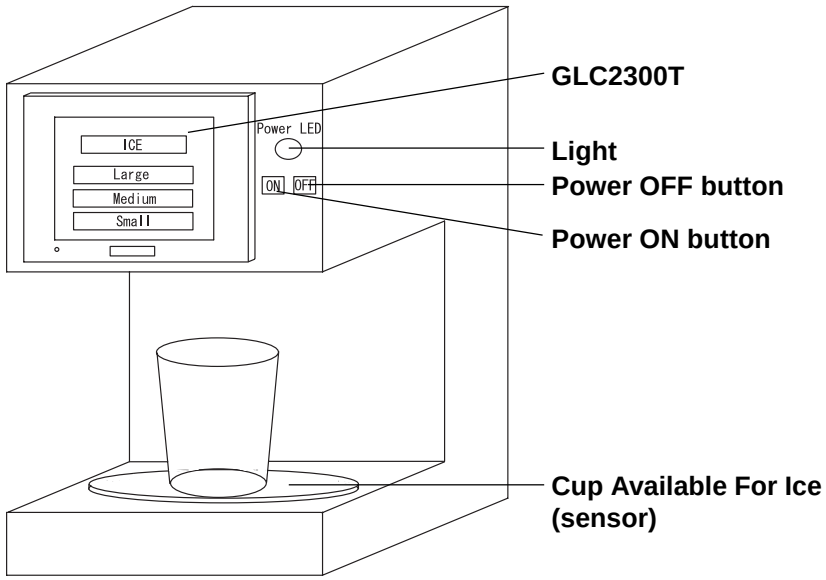
**<Logic Program>**



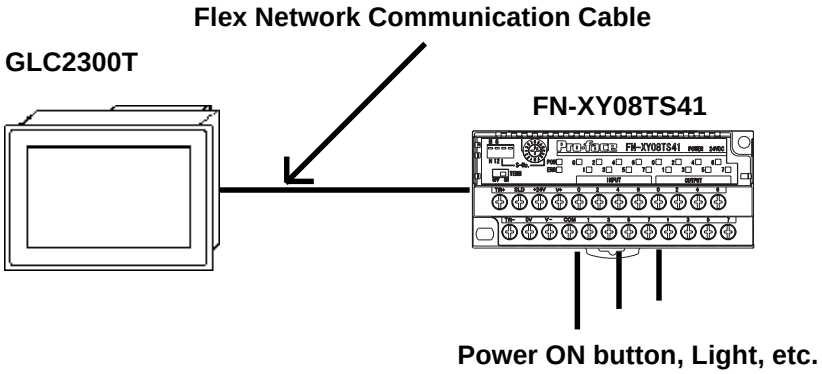

**<Screen>**



## ◆ Soft Drink Machine




## ◆ Hardware Design




## ◆ Assignment of the I/O Configuration

The "Ice_pushbutton", "Large_pushbutton", "Medium_pushbutton", and "Small_pushbutton" will be placed on the GLC screen for touch-panel inputs. These buttons are not assigned to the terminals.

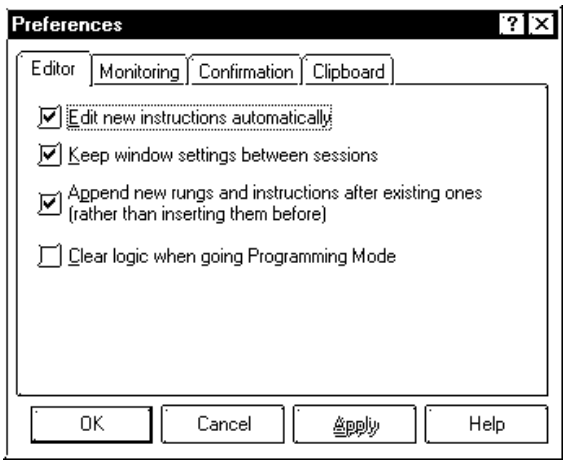
| Variable Name | Terminal Type | Terminal No. |
|---|---|---|
| Power_ON_pushbutton | Input | I0 |
| Cup_Present_for_Ice | Input | I2 |
| Power_OFF_pushbutton | Input | I6 |
| Light | Output | Q0 |
| Ice | Output | Q1 |
| Soda_valve | Output | Q2 |

## 2.1.1 Preference Area Settings (Prior to Creating a Logic Program)

Prior to creating a logic program, you can designate the general settings used in order to customize your program creation/operation.

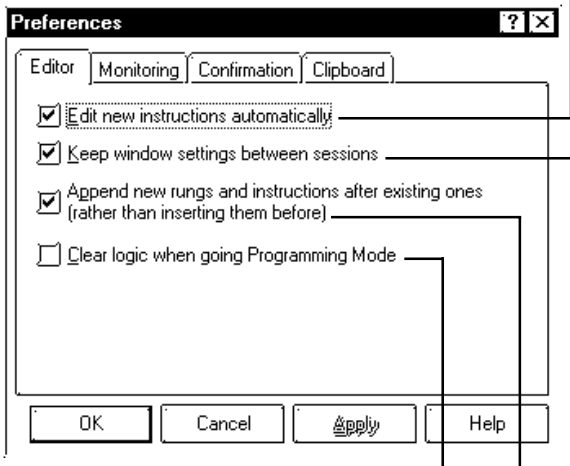
## ■ Designating Settings

1. Select [Preferences] from the [File] menu and the [Preferences] dialog box will appear.




2. Click on each check box to select or deselect a setting. The followings page's data explains each tab setting.

### ◆ Editor Tab



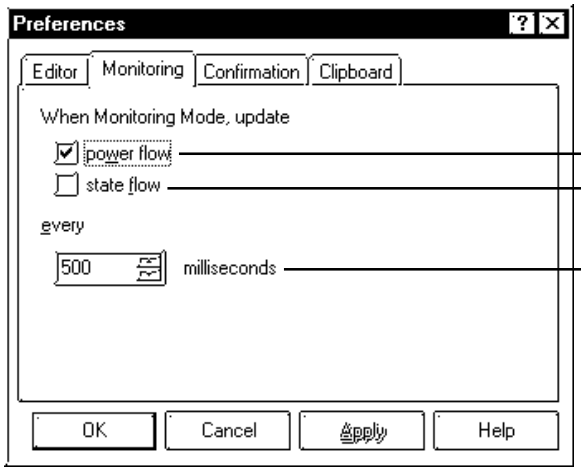

If selected, the [**Instruction Parameter**] box is automatically opened for any new instructions inserted in your program.
(Default: selected)

If selected, the Editor opens at start up all windows that were open at the end of the last session. Settings (such as window size and position) for any windows open during your editing session are retained. This also applies to the [Data Watch] window which retains its contents when the current program runs On-line.
(Default: selected)

If selected, new instructions are appended to the right of the [**focus**]. Objects (including rungs, labels, and subroutines) are appended below the [**focus**]. If cleared, new instructions are inserted to the left of the [**focus**]. Objects are inserted above the [**focus**]. If the [**focus**] is on a [**shunt**], new instructions are inserted on the [**shunt**].
(Default: selected)

If selected, the ladder logic screen will be cleared when going Programming Mode from Monitoring Mode.
(Default: not selected)

### ◆ Monitoring Tab

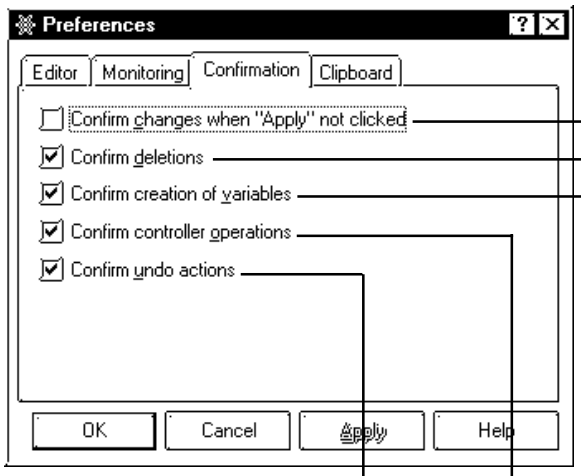



If selected, [**power flow**] is displayed while the Controller is running. The **[power flow]** feature highlights the rung (a line connecting instructions) that is executed during the RUN mode of the controller.
(Default: clear)

If selected, [**state flow**] is displayed while the Controller is running. [**state flow**] is the display feature that emphasizes the command run by the controller. [**power flow**] and [**state flow**] can be displayed together on the same screen.
(Default: not selected)

Specifies how often the Editor requests new data from the Controller to update [**power flow**], [**state flow**], data values, and the [**status bar**].
(Default: 500 msec)

### ◆ Confirmation Tab




If selected, the Editor accepts changes you make only when you click [**Apply**]. If cleared, Editor accepts changes immediately but asks for confirmation.
(Default: not selected)

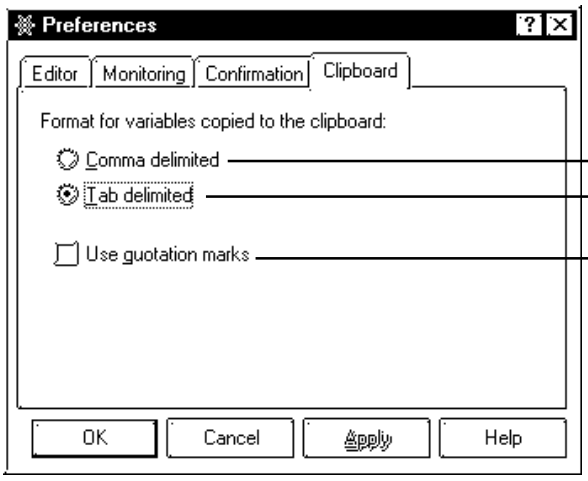
If selected, Editor asks for confirmation for all deletions when you are creating your program.
(Default: selected)

If selected, Editor asks you to confirm the creation of every new variable in your program. This applies only to the Programming Mode environment.
(Default: selected)

If selected, Editor asks you to confirm any change in the Controller operation (i.e., Start/Stop, Read/Write.)
(Default: selected)

If selected, Editor asks you to confirm any undo action.
(Default: selected)

### ◆ Clipboard Tab




- If selected, the fields copied from the variable list of the Editor to the clipboard are separated by commas.
  Ex., My_variable, Discrete, adescription
  (Default: not selected)
- If selected, the fields copied from the variable list of the Editor to the clipboard are separated by tabs.
  Ex.,My_variable[TAB]Discrete[TAB]adescription
  (Default: selected)
- If selected, the fields copied from the variable list of the Editor to the clipboard are separated by delimiter and enclosed in double quotes.
  Ex.,"My_variable", "Discrete", "adescription"
  (Default: selected)

**Note: In this tutorial, be sure to use the default settings. Click on [Cancel] to close the [Preference dialog] box and preserve the default settings.**

# 2.2 Tutorial Overview

1. Start the GP-PRO/PB III C-Package.
   Reference *2.3 How to Start the GP-PRO/PB III C-Package*

2. Select the GLC Type and PLC Type in the [**New**] dialog box.
   Reference *2.3 How to Start the GP-PRO/PB III C-Package*

3. Develop a logic program.
   1. Determine variables.

      This section describes how to designate the functions of the logic program to be used in the Editor as well as how to create and delete variables and set the initial values.

      Reference *2.4 Creating Variables*

   2. Create a logic program.

      This section describes how to create rungs, insert instructions and branches, and how to delete rungs, instructions and branches associated with the rungs.

      Reference *2.5 Inserting Rungs, Instructions, and Branches*

   3. Assign variables to the logic program.

      This section describes how to assign variables to the instructions in the logic program.

      Reference *2.6 Assigning Variables to Instructions*

   4. Insert descriptions.

      This section describes how to label the logic program with descriptions. The descriptions include the procedures for documenting the entire program, specific rungs, and individual instructions.

      Reference *2.7 Documenting a Ladder Logic Program*

   5. Edit.

      This section describes how to copy, cut and paste rungs.

      Reference *2.8 Copying, Cutting, and Pasting Rungs*

   6. Subroutine

      This section describes how to insert subroutines and labels in the logic program.

      Reference *2.9 Subroutines and Labels*

7. Search.

   This section describes how to search and go to the desired circuit quickly in the logic program.

   Reference *2.10 Navigating a Ladder Logic Program*

8. Assign I/O.

   This section describes how to assign the logical variables in the logic program to the actual I/O terminals.

   Reference *2.11 I/O Configuration*

9. Check errors.

   This section describes how to check for errors in the logic program.

   Reference *2.12 Checking the Validity of a Program*

10. Print.

    This section describes how to print out the logic program.

    Reference *2.13 Printing Your Ladder Logic Program*

11. Import and export.

    This section describes how to "read" and "write to" the logic program.

    Reference *2.14 Importing/Exporting a Logic Program*

4. Develop a screen program.

   Use the GP-PRO/PB III and create a screen linked with the logic program.

   Reference *2.15 Developing a Screen Program*

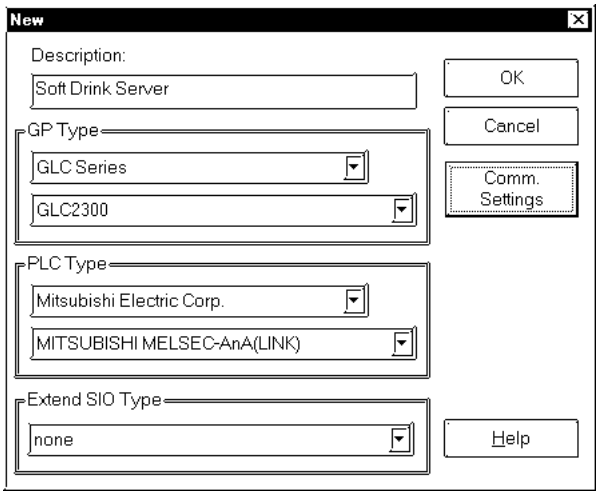
# 2.3 How to Start the GP-PRO/PB III C-Package

Activate the Project Manager prior to creating a logic program with the Editor.

1. Click the [**Start**] button, and go to [**Programs**]-[**Pro-face**]-[**ProPB3 C-Package**], and then click [**Project Manager**].

2. The Project Manager starts up.

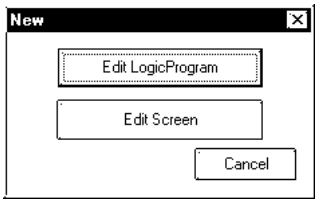

3. In the Project Manager screen, select [**New**] from the [**Project**] menu, or click the [**New**] button. Designate the settings as follows, and press the [**OK**] button.




Description : Soft Drink Server
Display Type : GLC series
GLC2300
PLC Type : Digital Electronics Corporation
Memory Link SIO type*1
Extended SIO Type: None

*1 Select "Memory Link SIO Type" and "Digital Electronics Corporation" for the "Device/PLC Type" when using just the GLC.

4. A window appears asking whether you will create a Logic Program or Screen. Click [**Edit LogicProgram**] to activate the Editor.




# 2.4 Creating Variables

This section describes how to designate the functions of the Editor as well as how to create and delete variables and set the initial values used on the Editor.

The completed sample of the tutorial program created in this lesson is located in the "Soda.prw" file in the "C:\ProgramFiles\Pro-face\ProPBWin\GLC_Samples" folder.

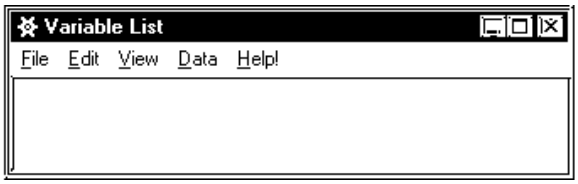
## 2.4.1 Creating a Variable List

You can add variables at any point while creating a ladder logic program. For convenience, create a list of the variables you will use in the tutorial now. Variables are addressable units of data you create and map to logic program instructions.

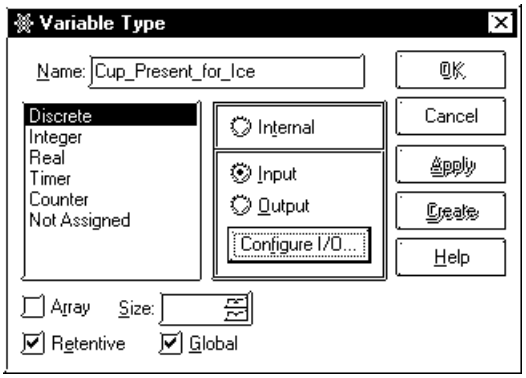
### ■ Creating a List

Please refer to the on-line help for detailed descriptions of the menu items.

1. From the [**Data**] menu, select [**Variable List**]. The Variable List window is displayed.




2. From the [**Edit**] menu, select [**Add Variable**], and the [**Variable Type**] dialog box will appear.




3. Type "**Cup_Present_for_Ice**" in the Name field.


***Variable names cannot begin with numerical characters or contain spaces. For example, you could not name a variable "1Switch" or "Switch 1", but you could name it "Switch1"or "Switch_1". Variable names are not case sensitive. For a variable name, you can use only alphabetical/numerical characters and "_". A maximum of 20 characters can be used. Array/Timer/Counter element values and Integer/Real bit values are included in these characters.***

## 2.4.2 Selecting Variable Types

The variable "**Cup_Present_for_Ice**" is now displayed in the [**Variable Type**] dialog box. The words "**Not Assigned**" are highlighted in the list below it. There in no variable type assigned to "**Cup_Present_for_Ice**". Therefore, it needs to be assigned as a discrete input. Please refer to **"Pro-Control Editor User's Manual 2.2 Variable Types"** for details on the variable types.

### ■ Assigning Variable Types

1. Select [**Discrete**] from the [**Variable Type**] list.
2. Select [**Input**].
3. Click on the [**Retentive**] box to deselect it. Data will not be retained if the power supply is cut, or the GLC unit is reset.
4. Click on [**Create**]. "**Cup_Present_for_Ice**" has now been assigned as a discrete input.

   Note that the variable type change that you made to "**Cup_Present_for_Ice**" in the [**Variable Type**] dialog box has now taken effect in the [**Variable List**] window and that the Variable Type dialog box is still open. If you had clicked on [**OK**], the changes would still have occurred in the [**Variable List**] window, but the [**Variable Type**] dialog box would have closed. The advantages of leaving these dialog boxes open becomes apparent as you begin inserting rungs and instructions as well as using Editor's drag & drop, click, and insert features.


**You can select the variable types you want to view in the [Variable List] window by selecting [View], then selecting the variable types you want displayed. A check mark appears beside the selected variable types.**

Now you have learned how to create a variable and assign a variable type to it, create the list of variables shown in the following table. Variables can be created directly in the [**Variable Type**] dialog box.

| Variable Name | Variable Type | I/O Type | Hold/Release | Global |
|---|---|---|---|---|
| Power_On_pushbutton | Discrete | Input | Release | Local |
| Cup_Present_for_Ice | Discrete | Input | Release | Local |
| Ice_pushbutton | Discrete | Internal | Release | Global |
| Large_pushbutton | Discrete | Internal | Release | Global |
| Medium_pushbutton | Discrete | Internal | Release | Global |
| Small_pushbutton | Discrete | Internal | Release | Global |
| Power_Off_pushbutton | Discrete | Input | Release | Local |
| Ice | Discrete | Output | Release | Local |
| Soda_valve | Discrete | Output | Release | Local |
| Fill_Timer | Timer | Internal | Hold | Local |
| Number_of_Larges | Counter | Internal | Release | Local |
| Number_of_Mediums | Counter | Internal | Release | Local |
| Number_of_Smalls | Counter | Internal | Release | Local |

Close the [**Variable Type**] dialog box when you have finished.


**If you typed a variable name incorrectly, simply rename it using the [Rename] option in the [Edit] menu's [Variable List] window. To create variables faster in the [Variable List] window, press the INSERT key.**

## 2.4.3 Saving Your Program

To ensure the safety of created data, it is recommended that you save your logic program periodically. When a logic program is saved, the global variables created with the Editor are automatically registered in the Symbol Editor as GLC symbols and can be used in common with the GP-PRO/PB III display.

Reference *4.2.5 Symbol Editor in the GP-PRO/PB III Operation Manual*

### ■ To Save the Program

Select [**Save**] from the [**File**] menu on the Editor screen.


**You can also save your program by clicking**  **on the toolbar or by pressing the CTRL+S keys.**

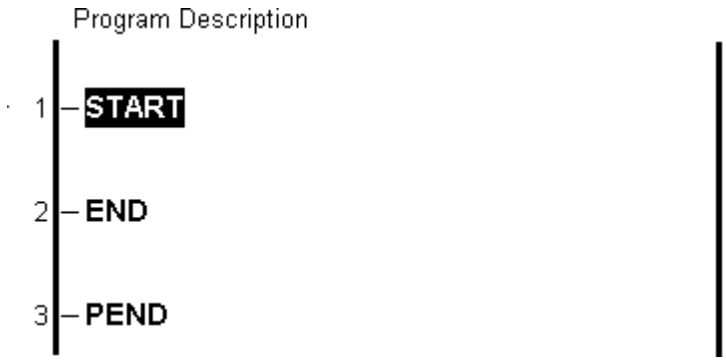
### < *Summary* >

In this section you have learned how to:

- create variables and use dialog boxes associated with the variables
- determine variable types
- save a program

# 2.5 Inserting Rungs, Instructions and Branches

The first step in creating a ladder logic program is to insert a rung. The screen initially shows a blank program as shown below. The completed sample of the tutorial program used in this lesson is located in the "Soda.prw" file in the "C:\ProgramFiles\Proface\ProPBWin\GLC_Samples" folder.




## 2.5.1 Inserting a Rung

Create a new logic program.

Down the left side of each new program are three rungs labelled START, END and PEND:

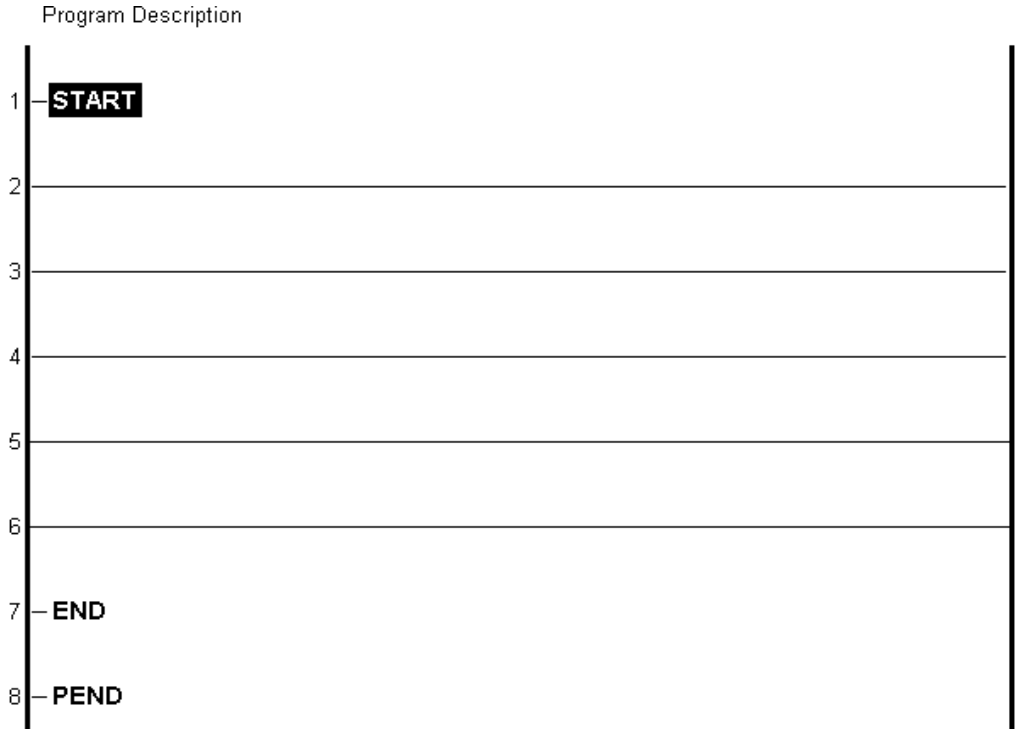
- The START rung indicates the start of the main program area.
- The END rung indicates the end of the main program area.
- The PEND rung indicates the end of the total program area. No rungs can be inserted after the PEND rung.

The rungs between START and END are executed every scan. Any rungs inserted above START are initialization logic and are executed on the first scan only. The area between the END and PEND rung is reserved for subroutines.

Reference *See the "**Programmer's Reference**" in the "**Editor Help**" area for a detailed explanation of the START, END, and PEND rungs.*

### ■ To insert a rung

1. Click on the rung number 1 left of the word START. Rung 1 is selected.
2. Right click once. A shortcut menu appears.
3. Select [**Insert Rung**]. (Or select [**Rung**] from the [**Insert**] menu.) A new rung appears at number 2, below the START rung.
4. Using the above method, insert five (5) more rungs below the START rung. The screen will be like the picture shown below.

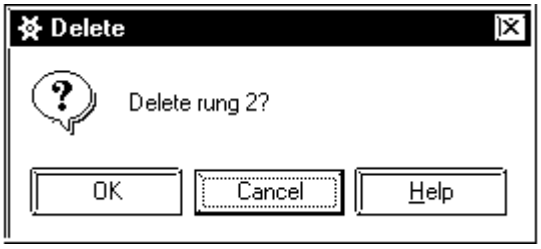



**You can also insert a rung by selecting [Rung] from the [Insert] menu, or by clicking on in the toolbar.**

## 2.5.2 Deleting a Rung

### ■ To delete a rung

1. Select the rung you want to delete. In this example click on the number "2" (the rung number) on the left side of rung 2.
2. Press the DELETE key, or right-click on the rung and click on the [**Delete Rung]** selection. The [**Delete**] dialog box will appear.




3. Click on [**OK**].

**As with other Windows applications, the Editor has an "Undo" command. From the [Edit] menu, select [Undo Changes to XX], or click on in the toolbar.**

## 2.5.3 Inserting Instructions

There are many ways to insert instructions into an Editor ladder logic program and assign variables to them. As you create the ladder logic program in the tutorial, these methods are described and used.

### ■ Selecting a rung to insert instructions

1. Here, you are inserting instructions on rung 2. Click on anywhere on the rung 2 line to select it, but not on the number "2" itself. The selected rung will then be highlighted, as shown below.


2. Once you have selected this rung, you can insert instructions. One way to do that is from the toolbar.

The Editor toolbar contains the following buttons.

Click on these buttons to insert instructions into a selected rung. The meaning of these buttons is as follows.

| | |
|---|---|
|  | Normally Open Contact (NO) |
|  | Normally Closed Contact (NC) |
|  | Coil (OUT) |
|  | Timer On Delay (TON) |
|  | Timer Off Delay (TOF) |
|  | Up Counter (CTU) |
|  | Down Counter (CTD) |

## ■ Method 1: Insert instructions from the toolbar

1. Click on the button. The following box will appear.


The instruction now appears on the selected rung. Also, there is a box above it with a flashing cursor inside. This is the "Instruction Parameter Box" and is where you enter a variable to associate with the instruction. This will be explained in more detail later in this chapter.

2. Click on the button. This places an output coil on the right side of rung 2. Though the "Instruction Parameter Box" is still flashing, please ignore it for now.

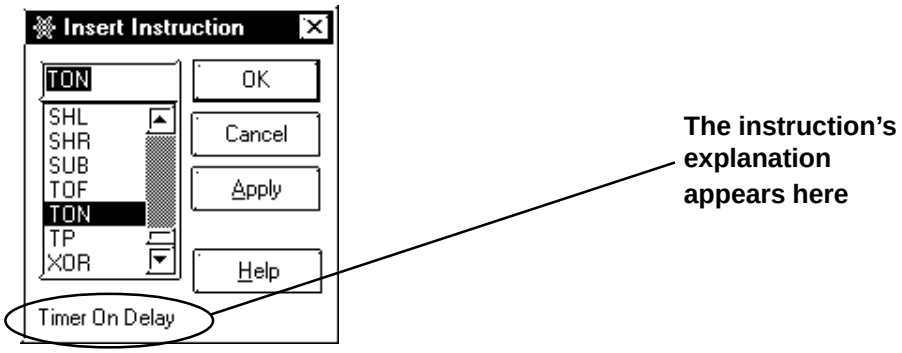
Reference *For Variable entry information, refer to "**2.6 Assigning Variables to Instructions**".*


3. Click on rung 2, between the **NO** and **OUT** instructions.
4. Click on the "**Normally Closed" (NC)** button , and that symbol will appear.


Note: **For a description of each toolbar button's feature, place the cursor over the button and read information that appears in the status bar. Though the toolbar offers an easy way to insert frequently used instructions, it does not include all Editor instructions available within Editor. You can also insert instructions using the following two methods.**

## ■ Method 2: Insert instructions from the [Insert Instruction] dialog box

1. Right click anywhere on rung 3 and a shortcut menu will appear.
2. Select [**Insert Instruction**]. The [**Insert Instruction**] dialog box appears.




This dialog box contains all instructions available to create a ladder logic program with the Editor. As you type or click each instruction, a descriptor of the instruction appears at the bottom of the dialog box.

**Note:** **You can also bring up the [Insert Instruction] dialog box by selecting [Instruction] from the [Insert] menu or by pressing INSERT key after you have selected a rung. To view detailed information on each instruction, click the [Help] button while selecting the desired instruction.**

3. Scroll through the instruction list until "**TON**" is found.
4. Select "**TON**".

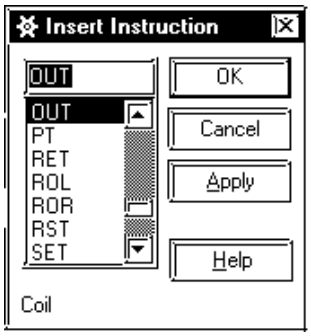
As with the [**Variable Type**] dialog box, you have a choice of clicking on either [**OK**] or [**Apply**] to register your selection. Since you are entering other instructions in your ladder logic program in this tutorial, the [**Insert Instruction**] dialog box needs to remain open. To do this, click on [**Apply**].


5. Click on rung 3, to the left of the TON instruction.
6. Scroll through the list of Editor instructions until you find "NO".
7. Double-click on "**NO" and** that symbol will appear.

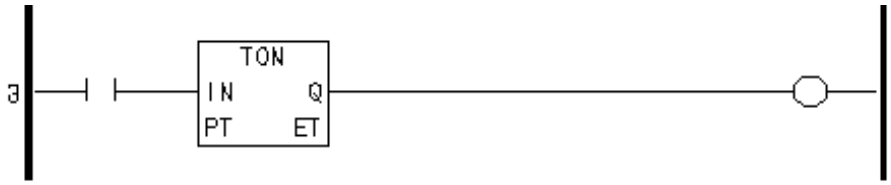
### ■ Method 3: Insert instructions by typing in the [Insert Instruction] dialog box

1. Type "out" in the field above the instruction list.

**Note:** **The instruction list automatically scrolls until the "OUT" instruction appears at the top of the list. Also, its name appears in the bottom left hand corner of the dialog box.**




2. Click on the rung section to the right of the TON instruction.
3. Click on [**Apply**] and the TON box will appear.

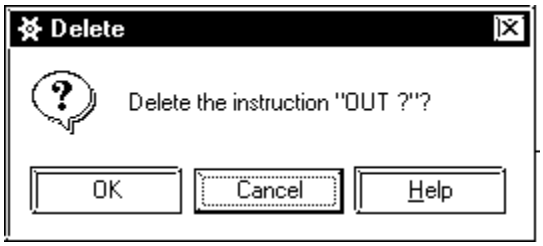



## 2.5.4 Deleting Instructions

Here, you will delete the OUT instruction you just inserted into rung 3.

### ■ To delete an instruction

1. Right click on the rung 3's OUT instruction and a shortcut menu will appear.
2. Select [**Delete**]. A dialog box will appear to confirm that the instruction is to be deleted.




3. Click on [**OK**].

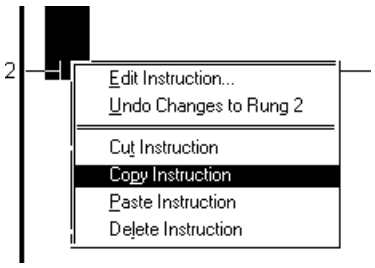
**Note:** **You can also delete an instruction by selecting it and pressing the DELETE key, or clicking on ☒ in the toolbar.**

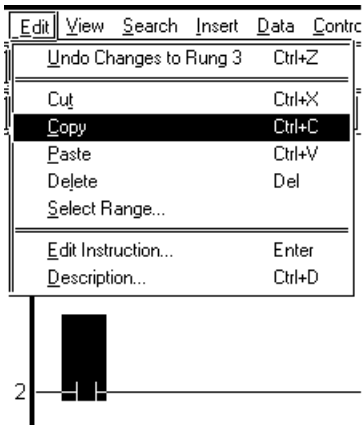
## 2.5.5 Copying and Pasting Instructions

Here, you will copy the instruction inserted into a rung and paste this instruction into another rung.

### ■ To copy an instruction

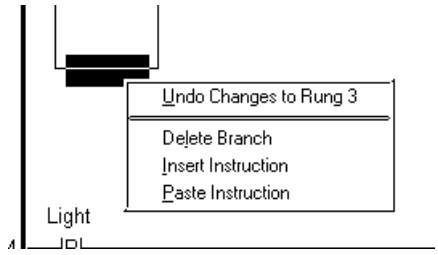
1. Click on the instruction you wish to copy.
2. Right-click and select [**Copy Instruction**], or select the [**Editor**] menu's [**Copy**].

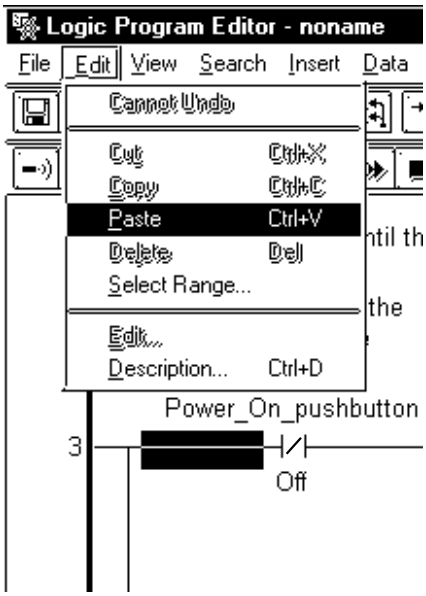


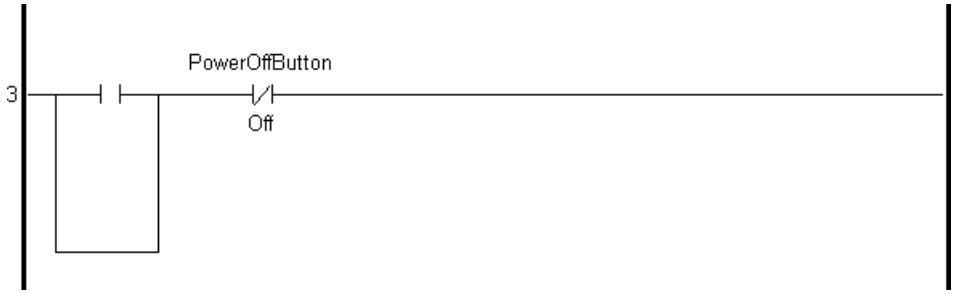

### ■ To paste an instruction

1. Click on the place you wish to insert the copied instruction.
2. Right-click on the [**Paste Instruction**] or click on the [**Edit**] menu's [**Paste**].




3. Now the copied instruction is pasted (inserted) into the desired rung.

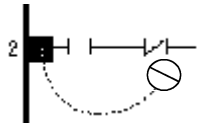


## 2.5.6 Inserting Branches

This section explains how you can insert a branch on rung 2 between the NO and the NC instructions. This branch is designed to turn the light in the soda pop machine.

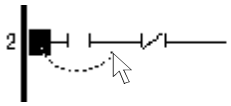
### ■ To insert a branch

1. Place the cursor at the point on the rung where you want the branch to begin. In this case, directly to the left of the NO instruction.
2. Click and drag the mouse to the right. The cursor has turned into a ⊘ with a dotted line attached to it.

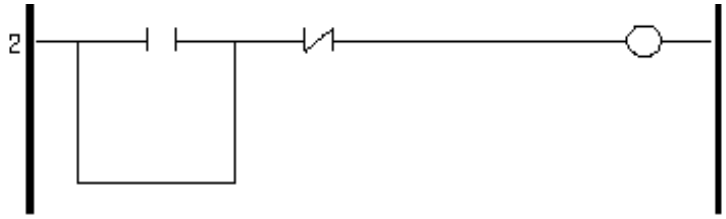



Whenever the end point of the branch is in an incorrect location, the Editor changes your cursor to a ⊘. Also, whenever the end point of the branch is in a valid location, the cursor returns to normal. If you release the cursor while it is normal, a branch is inserted between the starting point and where you released the mouse. If you release the mouse when the cursor is a ⊘, a branch will not be created.

3. Click and drag the mouse to the right until the cursor is between the NO and NC instructions and is not a ⊘.

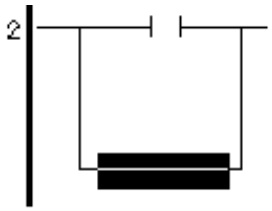



4. Release the mouse and a branch appears between the NO and NC instructions.



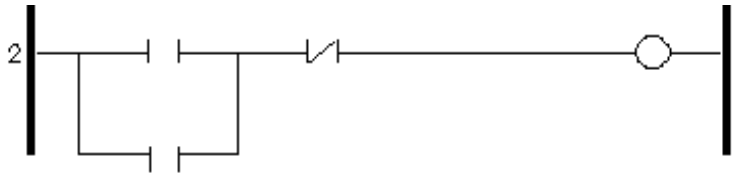

### ■ To add an instruction to a branch

1. Select the branch by clicking on the bottom of it.




2. The [**Insert Instruction**] dialog box should still be open. If it is not, open it using any of the previously described methods

3. Select the NO instruction from the [**Insert Instruction**] dialog box and insert it using any of the previously described methods. Rung 2 will appear like this:




**Note:** **To delete a branch containing instructions you must first select and delete each instruction.**

## 2.5.7 Initialization Logic

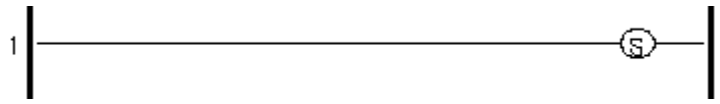
Logic inserted above the START rung is called initialization logic. It is executed only once when the Controller is started.

### ■ To insert initialization logic

1. Right click on "**Program Description**" field located above the START rung. If it is not visible, select [**Descriptions**] from the [**View**] menu, and then select [**Program**].
2. Select [**Insert Rung**] from the shortcut menu, and a rung is inserted above the START rung.

**Note:** **In the following examples the rungs have been moved down one position (i.e. the rung which was previously number 2 is now rung 3).**

3. Right click on the initialization rung (rung1).
4. Select [**Insert Instruction**] from the shortcut menu.
5. Select the SET instruction from the [**Insert Instruction**] window and click on [**OK**].



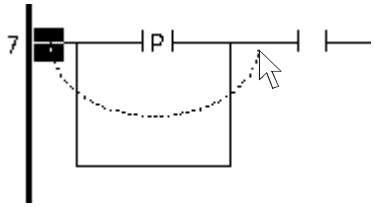

This rung is used to turn the ice machine of a soft drink dispenser ON. It remains ON while this exercise shows how to assign variables to instructions. is started up and only needs to be set once.

**Note:** **If you do <u>not</u> have [Append New Rungs and Instructions] selected in the [Preferences] dialog box, you must select the START rung to insert any initialization rungs. These rungs will appear below the program description.**

You have now completed rungs 3 and 4 of the ladder logic program as well as one rung of initialization logic. Please complete rungs 5-7, as shown on the following page. To help you complete your program, please remember that the |P| instruction is the Positive Transition (PT) instruction.

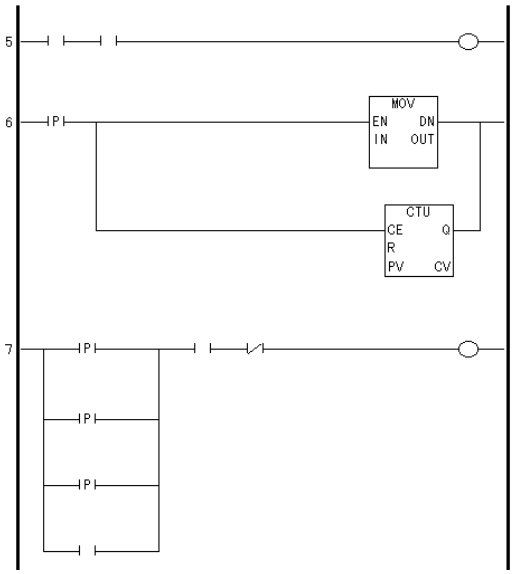
## ■ To insert multiple branches into rung 7:

1. Insert the first branch as previously described.
2. Insert the next branch by starting to click and drag from the same point as the previous branch.
3. Drag the cursor around the previous branch to the point on the rung where you want the branch to be inserted.




When the mouse is released, a new branch will be inserted over the previous branch, when is then pushed down.

In the example below, instructions have been inserted on rungs 5-7.




### < *Summary* >

In this section, you have learned how to:

- insert and delete rungs
- insert and delete instructions
- insert and delete branches

# 2.6 Assigning Variables to Instructions

This exercise shows how to assign variables to instructions.
In 2.4 Creating Variables you created a variable list which includes some of the variables used in the tutorial ladder logic program. Please re-open the [**Variable List**] dialog box now.

## ■ To open the Variable List dialog box

1. From the [**Data**] menu, select [**Variable List**].
2. Move this dialog box to the lower left corner of your screen. If the [**Insert Instruction**] dialog box is still open, close it by clicking on [**Cancel**].

## 2.6.1 Instruction Parameter Box

In the previous section, a field appeared with a flashing cursor inside it when you first inserted an instruction on a rung. This is the **Instruction Parameter Box** and is where you enter the variables you want associated with the instruction.

## ■ To access the Instruction Parameter Box of a basic level instruction:

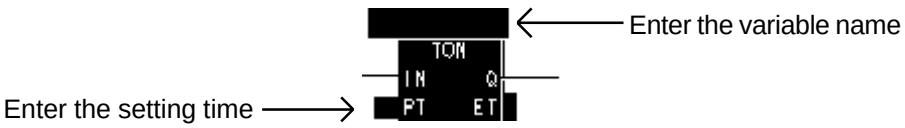
1. Double-click on rung 3's OUT instruction. A text field will open above the instruction with a flashing cursor inside of it. This is the "Instruction Parameter Box".

**Note: The "Instruction Parameter Box" can also be accessed by clicking on the instruction and pressing the ENTER key or by right clicking on the instruction and selecting [Edit Instruction] from the shortcut menu.**

General instructions (non-basic level instruction) have more than one **"**Instruction Parameter Box". For example, a TIMER ON DELAY (TON) instruction has two (2). One is where you assign a variable, and the other is where you enter the preset time in milliseconds.

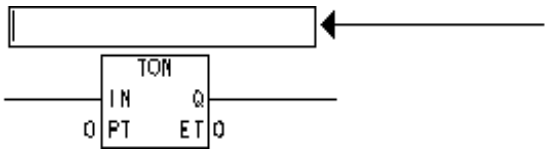
## ■ To access the Instruction Parameter Boxes of general instructions

1. Click on rung 4's TON instruction. The TON instruction then changes as follows:




Above the TON instruction a black highlighted area will appear. This is where you enter the variable to be assigned to the TON instruction. Next to the Preset (PT) element is another black highlighted area. This is where you enter the preset time in milliseconds.

2. Double-click on the black highlighted area above the TON instruction to select the "Instruction Parameter Box". Here, you can assign a timer variable to the instruction.



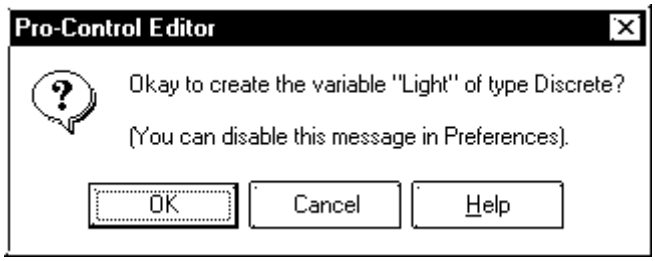

3. Next, double-click on the area immediately to the left of the PT element in the TON instruction. The [**Data Value**] dialog box opens. Here, enter the preset time in milliseconds that will elapse before output (Q) is turned ON. (Assigning variables and other operands to instructions will be discussed in the next section.)
4. Close the [**Data Value**] dialog box.

## 2.6.2 Entering Variables

One method of entering a variable into an Instruction Parameter Box is to type directly into the box.

### ■ To enter text in the Instruction Parameter Box

1. Double-click on the OUT instruction's Instruction Parameter Box on rung 3.
2. Type "Light" in the box.
3. Press the ENTER key. The following dialog box appears asking you to confirm the creation of the variable.

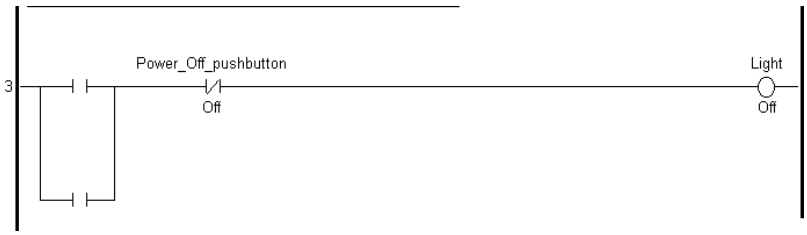



4. Click on [**OK**]. In the [**Variable List**] dialog box the variable "Light" appears in the list. The Editor has automatically assigned it a *variable type*. In this case it has assigned it as an internal discrete variable.


**The Editor automatically assigns variable types to any new instruction variables created. You can also type a variable that already exists in your variable list directly into an Instruction Parameter Box. The variable is assigned automatically when you finished entering it.**

5. Using the above mentioned method, assign the operand "Power_Off_pushbutton" to the NC instruction on rung 3.

Rung 3 should look like this:



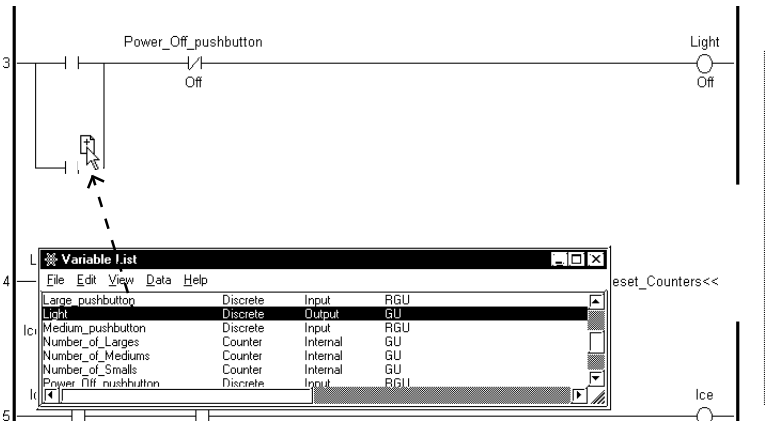

**If you change the variables assigned to "Coil" instructions (i.e. OUT, SET, RST, NEG) to "Retentive", the "Coil" instructions also automatically change to "Retentive" type (i.e. M, SM, RM, NM).**

6. Assign the variable "Ice_Maker" to the SET coil on the first initialization rung. This variable can be created by typing it directly into the "Instruction Parameter Box". After it is typed, the initialization rung appears as follows:


Another method of assigning variables to instructions is to simply drag the variable from the [**Variable List**] dialog box to the instruction itself. This method is very convenient if there are many instructions which need to have the same variables assigned to them. The advantages of using this method will be explained in 2.11 I/O Configuration.

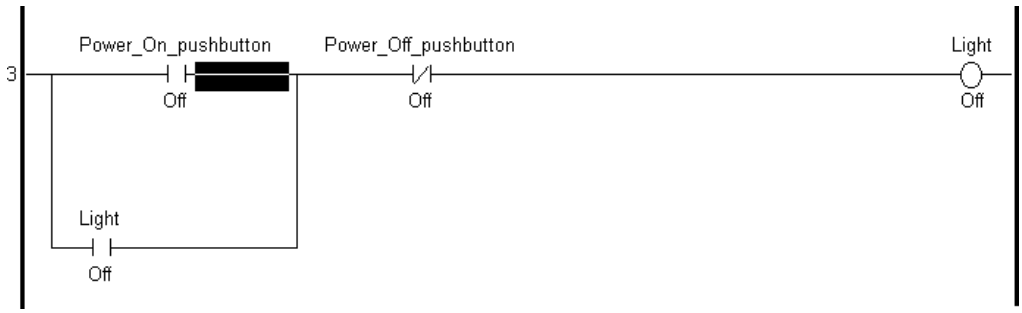
## ■ To assign a variable using the Variable List dialog box

1. Call up the [**Variable List**] dialog box.
2. Click on "Light" in the [**Variable List**] dialog box but do not release the mouse button.
3. With the mouse button still pressed, drag "Light" to the NO instruction located on the branch on rung 3. As when inserting branches, note that your cursor initially becomes a ⃠. When the cursor is in this state you cannot assign the variable to any instruction.
4. When you reach the No instruction, your cursor will change to a ✝ mark.




The variable is then assigned when the cursor is released. As long as the cursor appears as a ✝, you can assign the variable to an instruction.

4. Release the mouse button. The variable “Light” is now assigned to the NO instruction.
5. Click on and then drag the “Power_On_pushbutton” variable to the other NO instruction on rung 3. Rung 3 should now appear as follows:




In general, variables which are expressions, constants are assigned to instructions in exactly the same way as basic type variables, however, they must be typed in manually since there is no window to drag them from.

## 2.6.3 Completing the Operation

Since you have learned how to assign variables to instructions, you can now complete the remaining rungs of the program. A diagram of the completed rungs is presented on the following page.

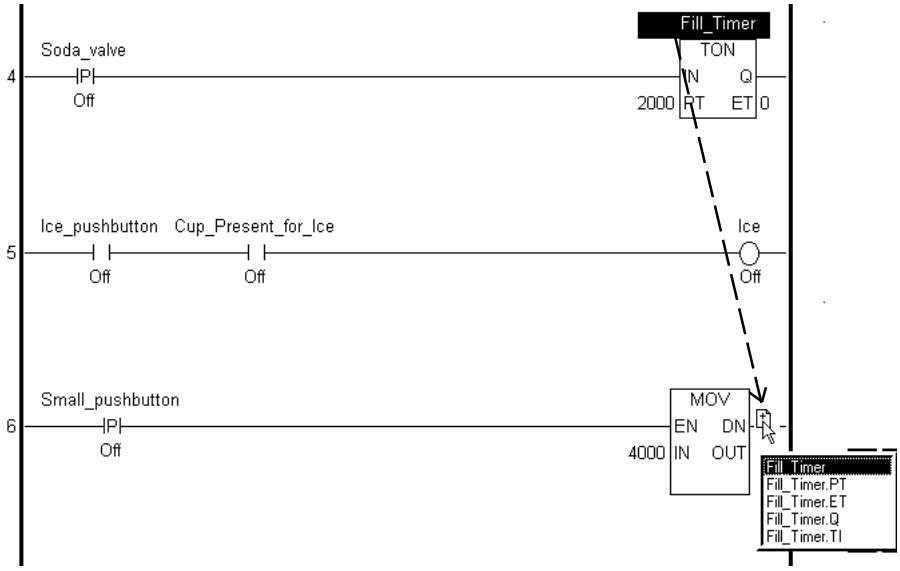
Notice that the MOV instruction on rung 6 and the NC instruction on rung 7 contain the variables "Fill_Timer.PT" and "Fill_Timer.Q" respectively. These variables refer to the "PT" and "Q" elements of the Timer with the "Fill_Timer" variable assigned to it.

The following three procedures are available for entering these variables. You can either:

- select the Instruction Parameter Box and type the "Fill_Timer" variable in directly.
- click on and drag the "Fill_Timer" variable from the [**Variable List**] dialog box and add the ".PT" and ".Q" extensions in the Instruction Parameter Box.
- drag the Instruction Parameter Box to the instruction you want to copy, and enter a variable selected from the special Variable List.

The following are detailed instructions for this procedure.

1. Select the source Instruction Parameter Box you want to copy from.
2. Drag the counter and timer variables to the destination instruction you ant to copy.
3. Select and double click on the desired parameter from the special Variable List Box.



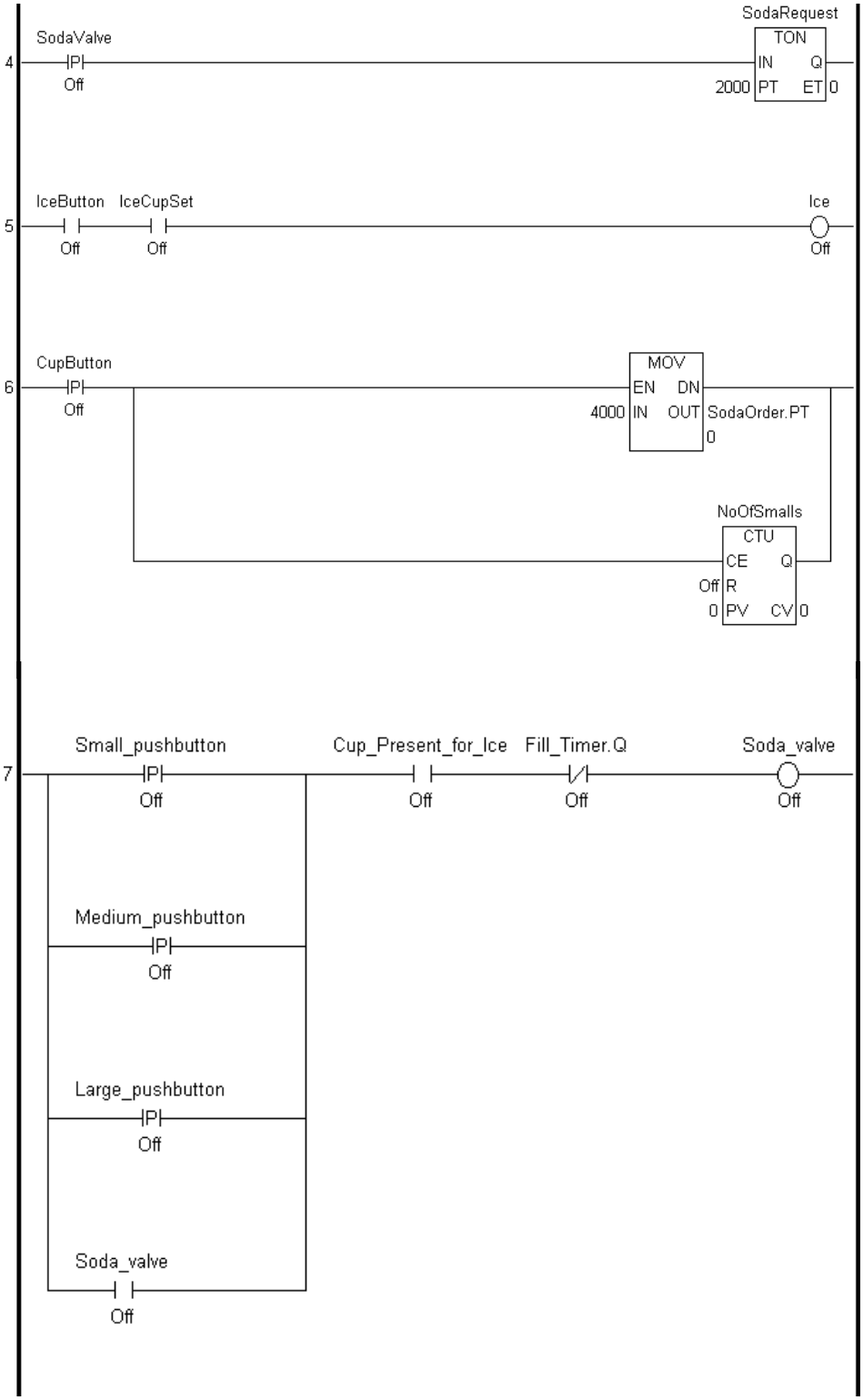

Important

***These methods are used with rungs 6,7, and onwards. The application instructions' exclusive variables such as "Fill_Timer.PT" or "Fill_Timer.Q" consist of a variable name and a file extension:***

| | |
|---|---|
| ***.CV | *(Current value)* |
| ***.PT | *(Set value)* |
| ***.Q | *(Output bit)* |
| ***.R | *(Reset bit)* |

Reference *Pro-Control Editor User Manual "**2.2 Variable Type**"*

**< Tutorial Program Sample >**




## < *Summary* >

In this section, you have learned how to assign variable to instructions.

# 2.7 Documenting a Ladder Logic Program

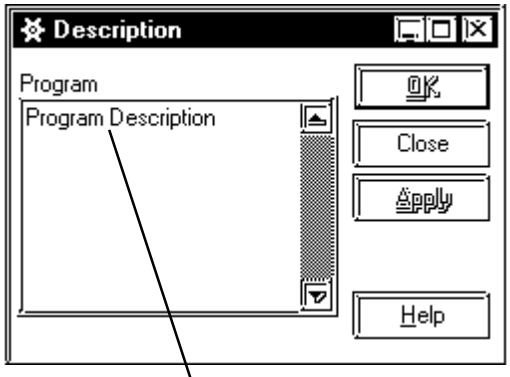
It is recommended that you document your ladder logic program. This data explains to users exactly how the program and each of its elements perform and is useful when the program needs to be altered or debugged later on. In the Editor, you can document how the program performs, how each rung operates and what specific variables are used for.

## 2.7.1 Adding a Program Description

The first description to add to your ladder logic program is a description explaining the program's features.

### ■ To add a program description

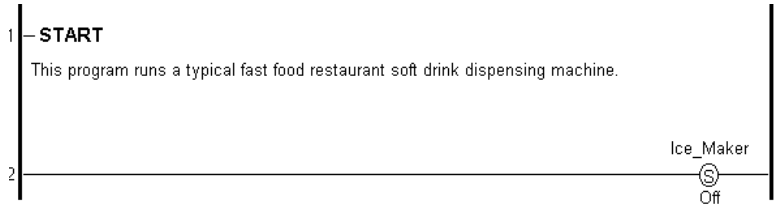
1. Double-click on the "**Program Description**" field at the top of the screen, and the [**Description**] dialog box will appear.




All Editor descriptions are entered here.

**The word "Program", above the text field in the description dialog box, indicates that the text field contains a description of the program.**

2. Click on the "**Program Description**" text.
3. Type "**This program runs a typical fast food restaurant soft drink dispensing machine**".
4. Click on [**OK**]. This description now appears at the very top of the ladder logic program. (You may need to scroll up to see it.)




**You can also add or edit a Program Description by double-clicking the lower left-hand panel of the status bar.**

## 2.7.2 Adding a Rung Description

Via the Editor, you can add descriptions to each rung of your program. In the following example, a description is added to rung 5.

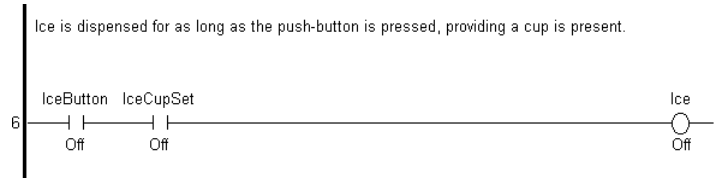
### ■ To add a rung description

1. Right-click on rung 5's left side number.
2. Select [**Description**] from the shortcut menu and the [**Description**] dialog box opens. It is the same dialog box you opened previously, however, the descriptor above the text field now says **Rung 5** instead of **Program**.

**Note: You can also open the [Description] dialog box by selecting [Description] from the [Edit] menu or by clicking on in the toolbar.**

Rung 5 controls the ice dispenser.

3. Click on the text field of the [**Description**] dialog box.
4. Type "**Ice is dispensed for as long as the push-button is pressed, providing a cup is present**".
5. Click on [**Apply**].




To add descriptions to the remaining rungs of your program easily, keep the [**Description**] dialog box open.

### ■ To add a description to rung 3

1. Click anywhere on rung 3, outside of the **Instruction Parameter Boxes**. The descriptor at the top of the [**Description**] dialog box now says Rung 3.
2. Click on the text field.
3. Type "**The Light remains on until the Power_Off_Pushbutton is pressed**".
4. Click on [**Apply**]. In this tutorial only the comments for rungs 3 and 5 are explained.

## 2.7.3 Adding Descriptions to Variables

Descriptions can also be added to each of the variables in your ladder logic program.

You cannot however, add descriptions to labels or constants.

### ■ To add a description to a variable

1. The [**Variable List**] dialog box should be open. If it is not, open it now by selecting [**Variable List**] from the [**Data**] menu.
2. The [**Description**] dialog box should also be open. If it is not, open it now by selecting [**Description**] from the [**Edit**] menu.
3. Click on any Instruction Parameter Box containing the variable "Fill_Timer". Note that not only does the [**Description**] dialog box contain the descriptor "Fill_Timer", but that "Fill_Timer" is also highlighted in the [**Variable List**] dialog box.
4. Click on the text field of the [**Description**] dialog box.
5. Type "The Soda Request decides how long to keep the soda valve open. The preset time changes depending on the size selected".
6. Click on [**Apply**].


**You can also add descriptions to a variable by selecting the variable in the [Variable List] dialog box, instead of selecting it from the ladder logic program.**

### ■ To add a description

Here you will add a description to the variable "Power_On_pushbutton".

1. Click on the variable "Power_On_pushbutton" in the [**Variable List**] dialog box. The [**Description**] dialog box now contains the descriptor "Power_On_pushbutton".
2. Click on the text field of the [**Description**] dialog box.
3. Type **"The Power On pushbutton starts the soft drink machine"**.
4. Click on [**Apply**].

In this tutorial, descriptions are added to only the "Fill_Timer" and "Power_On_Pushbutton" variables. Descriptions for other variables can be created by simply repeating the procedure described here.

## 2.7.4 Description List Dialog Box

The [**Description List**] dialog box displays brief, one line descriptions of all variables and rungs in the program.

### ■ To bring up the Description List dialog box

- From the [**View**] menu, select [**Description List**].

### ■ To view a detailed description from the Description List dialog box

- Double-click the "Fill_Timer" variable in the [**Description List**] dialog box. The [**Description**] dialog box displays the detailed description of the "Fill_Timer" variable.

  The [**Variable List**], [**Description**], and [**Description List**] dialog box displays change to reflect the rungs and variables selected in the ladder logic program. However, the opposite is not possible; for example, if a variable in the [**Variable List**] dialog box or a description from [**Description**] or [**Description List**] dialog boxes is selected, the corresponding choice is not reflected in the ladder logic. In the Editor, there are methods which can be used to easily find specific variables for programming your ladder logic. This will be explained in more detail in 2.10 Navigating a Ladder Logic Program.

### < *Summary* >

You have learned how to add descriptions to the program, to rungs and to variables as well as how to call up the [**Description List**] dialog box.

# 2.8 Copying, Cutting and Pasting Rungs

When creating a ladder logic program, you may find you have to duplicate sequences of instructions on several rungs. You can speed up your work by copying and pasting completed rungs.

## 2.8.1 Copying a Rung

In the following exercise, two rungs are added between rungs 5 and 7. These additional rungs contain the same instructions as rung 6 with different variables assigned to them.

### ■ To copy a rung

1. Select rung number 6 by clicking on the left side number.
2. From the [**Edit**] menu, select [**Copy**].


**If you wish to select a range of rungs to be cut or copied, click on the rung number of the first rung you wish to select. Hold the [SHIFT] key down and select the rung number of the last rung you wish to select. All rungs between the two are then selected and can be cut or copied. Copying is limited to approximately 25 rungs. (The number varies depending on the number of instructions in the rung.)**

## 2.8.2 Pasting a Rung

The Editor pastes rung(s) below the current rung, as long as all the current rung is not selected. If [**Append new rungs and instructions**] is not selected in the [**Preferences**] dialog box, the copied rung is inserted above the current rung.

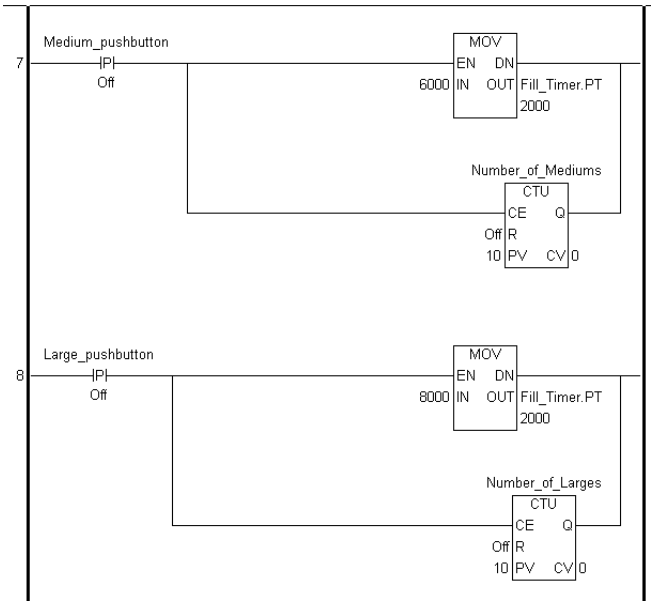

***A rung cut and pasted is loaded to the Editor's internal clipboard, then copied to the program. If you select an entire rung when pasting from the clipboard, the Editor replaces the rung you have selected with the rung in your clipboard.***

### ■ To paste a rung

1. Click anywhere on rung 6.
2. From the [**Edit**] menu, select [**Paste**]. Rungs 6 and 7 are now identical.
3. Click anywhere on rung 6.
4. From the [**Edit**] menu, select [**Paste**]. Rungs 6 to 8 are now all alike.


**When pasting a rung, all variables and descriptions associated with that rung are also pasted. Be aware that you may have to edit the pasted rung.**

**The variables on rungs 7 and 8 should now be changed, according to the following example.**




5. Change the variable name of the PT instruction on the rung as shown in the example above.

## 2.8.3 Cut Command

The Editor's Cut command allows you to take a rung or section of rungs out of one part of your program and move them to another. In the following tutorial, rung 4 is to be moved to the last rung of your program.

### ■ To use the "Cut" command

1. Click on rung 4.
2. From the [**Edit**] menu, select [**Cut**]. The rung is now taken from the ladder logic program and placed on the clipboard.
3. Click anywhere on rung 8.
4. From the [**Edit**] menu, select [**Paste**]. Rung 4 is now appended to below rung 8. The end of the program now appears as follows:



**To move an entire rung to another part of the program, first select the rung and drag it using the middle of the rung to the new location.**

***<Summary>***

In this section, you have learned how to copy, cut, and paste rungs.

# 2.9 Subroutines and Labels

When a [**JSR**] (jump to subroutine) or [**JMP**] (jump) instruction is inserted in a rung, it tells the Controller to resume scanning starting at that subroutine or label. The main difference between a subroutine and a label is that Editor executes a subroutine and then returns to the point in the ladder logic directly after the [**JSR**] instruction. If Editor jumps to a label (through the use of the [**JMP**] instruction), it continues executing the ladder logic program at that point and does not return to the [**JMP**] instruction during that scan.

Reference *For more information on the [**JMP**] and [**JSR**] instructions, see the Pro-Control Editor User Manual "**4.2.43 JMP (jump)/ 4.2.44 JSR (jump to subroutine**" .*

## 2.9.1 Inserting a Subroutine

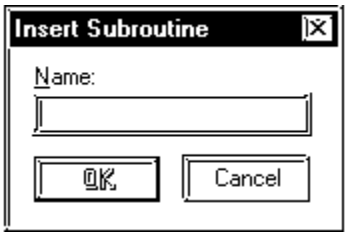
At the bottom of every Editor program are two rungs labelled "**END**" and "**PEND**".

The "**END**" label signifies the end of the main program area. The Editor executes the instructions between "**START**" and "**END**" with every scan. The area between the "**END**" label and the "**PEND**" (Program End) label is reserved for subroutines.

In the following tutorial, a subroutine is added.

### ■ To insert a subroutine

1. Click on the [**END**] label.
2. From the [**Insert**] menu, select [**Subroutine**]. The [**Insert Subroutine**] dialog box appears.




3. Type "**Reset_Counters**" in the [**Name**] field of the [**Insert Subroutine**] dialog box. A maximum of 32 characters, numbers, or underscore characters, can be used for a subroutine name. Variable names cannot begin with numerical characters and cannot contain spaces.

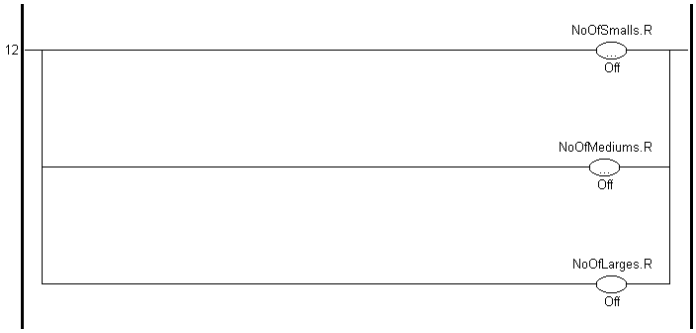
Reference *2.4.1 Creating a Varialble List*

4. Click on [**OK**]. At the end of your program the subroutine will appear.


Here, you insert your subroutine between the two new rungs labelled "**SUB START CounterReset**" and "**SUB END CounterReset**".

5. Right click on the "**SUB START CounterReset**" label.
6. Select [**Insert Rung**] from the shortcut menu to insert a rung between the "**SUB START**" and "**SUB END**" rungs.
7. Right click on the rung "**SUB START**" and "**SUB END**".
8. Insert an "**OUT**" instruction in the rung.
9. Insert 2 branches around the "**OUT**" instruction.
10. Insert an "**OUT**" instruction on each branch. The following is the completed subroutine.

This routine will reset each of the Counters every time the machine is turned ON.




Each of the variables you see here should be assigned to each of the "**OUT**" instructions. Assign these variables now.

This completes the subroutine you can add more than one subroutine to a ladder logic program by selecting either the "**SUBSTART**" or "**PEND**" rungs and repeating steps 2 through 6.

If you want a subroutine to be executed at some point in your ladder logic program you must insert a [**JSR**] instruction. This is explained in the following tutorial.

This subroutine is executed as soon as the **'Light' OUTPUT COIL** on rung 3 turns ON. Therefore, the [**JSR**] instruction must be placed on rung 4.

## ■ To insert a [JSR] instruction:

1. Select rung 3.
2. From the [**Insert**] menu, select [**Rung**].
3. Insert a [**PT**] instruction on rung 4.
4. Assign the variable 'Light' to the [**PT**] instruction.

5. Insert a [**JSR**] instruction to the right of the [**PT**] instruction. This is done from the [**Insert Instruction**] dialog box.
6. Type '**ResetCounter**', the name of the subroutine, in the [**Instruction Parameter Box**] of the [**JSR**] instruction. The rung appears as follows:


Whenever the [**JSR**] instruction "**ResetCounter**" receives power, it will jump to the subroutine "**ResetCounter**". Execution will resume from rung 5 once the subroutine has finished execution.

**Note: To delete a subroutine, you must first delete the individual rungs. After that, delete the "SUB START" rung. The "SUB END" rung will then be automatically deleted when the "SUB START" rung is deleted.**

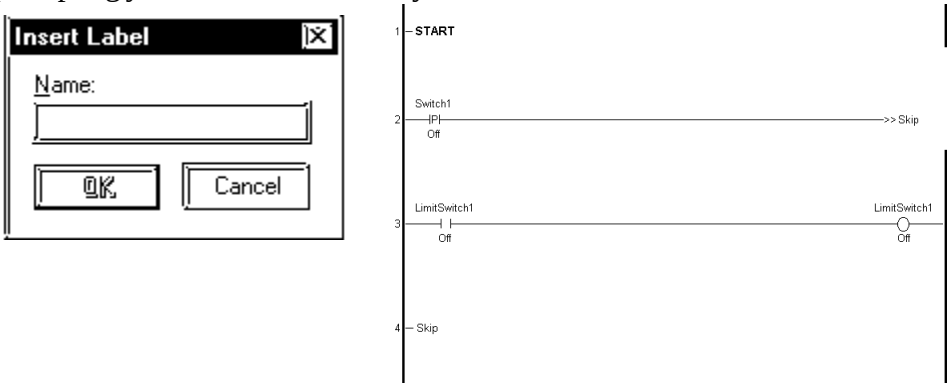
## 2.9.2 Inserting Labels

A label, which is combined with a [**JMP**] (Jump) instruction, can be inserted in any part of a ladder logic program. When the Controller executes a [**JMP**] instruction, it jumps to the designated label and begins executing the program at that point.

Labels are inserted above or below the selected rung depending if [**Append new rungs and instructions**] is selected in the [**Preference**] dialog box. This tutorial does not use any labels. However, to insert one, the following procedure is used.

### ■ To assign a label to your ladder logic program:

1. Click anywhere on the rung.
2. From the [**Insert**] menu, select [**Label**]. The [**Insert Label**] dialog box appears prompting you to insert a name for your label.




This is the name that is designated in the [**JMP**] instruction in your ladder logic. The same rules that apply to naming variables apply to naming labels.

### ■ To insert a [JMP] instruction

1. Right click on the right of the last instruction on the rung and select [**Insert Instruction**] from the shortcut menu.
2. Double click the [**JMP**] instruction in the [**Insert Instruction**] dialog box. The [**JMP**] instruction is inserted as the last instruction on the rung. Whenever the Editor sees this instruction in your program, it jumps to the designated label.

#### *< Summary >*

This section explained how to create subroutines and labels and insert [**JMP**] (jump) and [**JSR**] (jump to subroutine) instructions.

# 2.10 Navigating a Ladder Logic Program

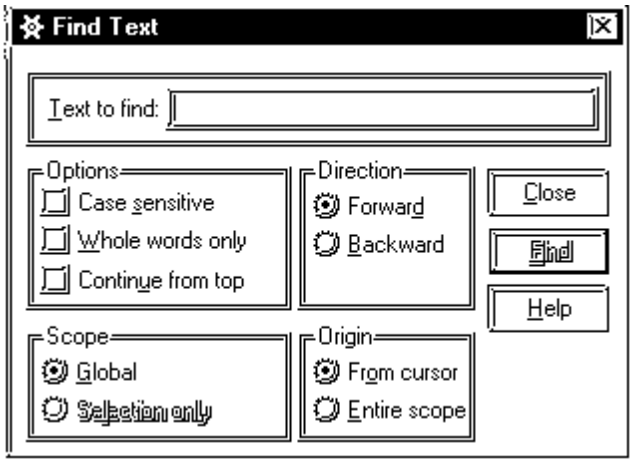
If a ladder logic program is large, using the scroll bars to locate specified points in your logic can take quite a bit of time. As a result, there are features available to help you find specified item in your program much more quickly. These are the [**Find**], [**References**], [**Bookmark**], [**Go to Rung**] and [**Go to Label**] commands.

## 2.10.1 The [Find] Command

The [**Find**] command allows you to locate specific textual references in your ladder logic.

### ■ To use the Find command:

1. If you have any windows open, close them before you use the [**Find**] command.
2. From the [**Search**] menu, select [**Find**]. The [**Find Text**] dialog box appears:




**Note: The [Find Text] dialog box can also be opened by clicking in the tool bar.**

#### ◆ Specifying the type of matching to apply to the search

- You can specify the type of matching to apply to the search. If you were trying to find the word 'Fill', the Editor would find all instances of that word, even if it found it as a lower case 'fill' or as part of another word such as 'Fillet'.
- If you selected [**Case sensitive**], Editor would find 'Fill' but not 'fill'. If you selected [**Whole words only**] Editor would find 'Fill' but not 'Fillet'.

#### ◆ Specifying the scope and direction of the search

- You can specify the scope and direction of the search. If [**Selection only**] is selected, the scope is limited to the highlighted portion of your program.
- Selecting [**Global**] includes the entire program. You can begin the search from the top of the selected scope by selecting [**Entire scope**] or from a given position by selecting [**From cursor**]. This tutorial starts the search from the beginning of the program.

3. Select the [**START**] label in your program.
4. Click the [**Text to find**] field of the [**Field Text**] dialog box.
5. Type 'FILL'.

6. Select [**Global**], [**Forward**], and [**From cursor**].
7. Click on the [**Find**] button. The "focus" moves to the first match found, a part of the 'Fill_Timer' variable.
8. Click on the [**Find**] button again. The "focus" moves to the next match found. When you have reached a point in your program where there are no more instances of the items you are trying to locate, a beep sounds.


**After the first [Find] operation. You can locate subsequent occurrences of a text match by selecting [Find Next] from the [Search] menu.**

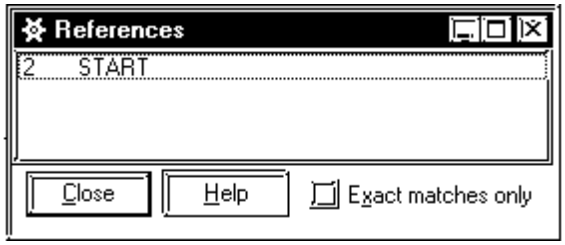
## 2.10.2 The [References] Command

The [**References**] command allows you to locate all occurrences of a specific variable in your ladder logic program. It identifies the rung numbers and the instructions the variable appears on.

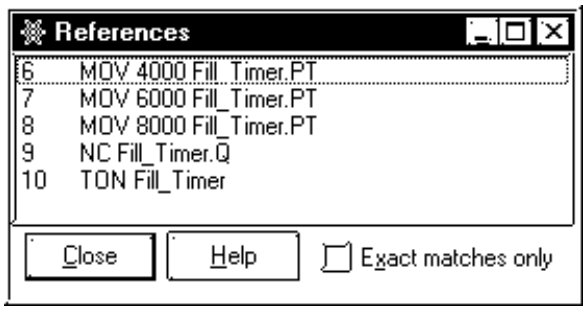
For this tutorial, you will select the [**START**] label. However, the [**References**] command can be implemented from any point in your program.

### ■ To use the Reference command:

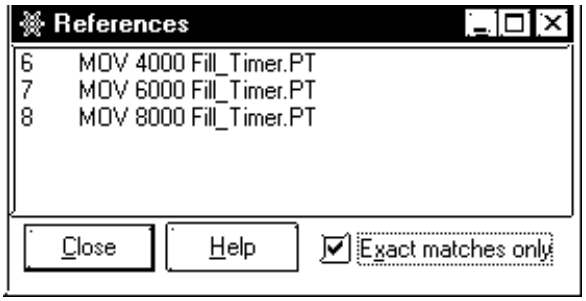
1. Click on the [**START**] label.
2. From the [**Search**] menu, select [**References**]. The [**References**] dialog box appears:




3. Re-size and move the [**References**] dialog box to the lower right hand corner of your screen.
4. Click on the rung 6's 'Fill_Timer.PT' variable and the [**References**] dialog box will appear as follows:




5. Select [**Exact matches only**].



In the [**References**] dialog box display:

- The number at the left of the line signifies the rung number the variable appears on. This display tells you the 'Fill_Timer' variable appears on rung 6,7,8,9 and 10. When [**Exact matches only**] is selected, the display shows that 'Fill_Timer.PT' occurs on rung 6,7 and 8.
- The next column on the line is the instruction type. This is the instruction that this variable has been assigned to on this rung. This display tells you the 'Fill_Timer' variables has been referred by three (3) [**MOV**] instructions, one [**NC**] instruction and a [**TON**] instruction.
- The last column on the line lists the parameter that has been assigned to this instruction, including the variable you initially referenced. In this display, you can see the integers 4000, 6000 and 8000 assigned to the IN elements, and 'Fill_Timer.PT' assigned to OUT elements.

The [**References**] dialog box changes in accordance with your selection every time you click on a variable in your ladder logic program. One advantage is when you click on any of the lines in its display, the corresponding point in your ladder logic appears.


**You must click on the parameter itself, not the instruction for the corresponding information to be displayed in the [References] dialog box.**

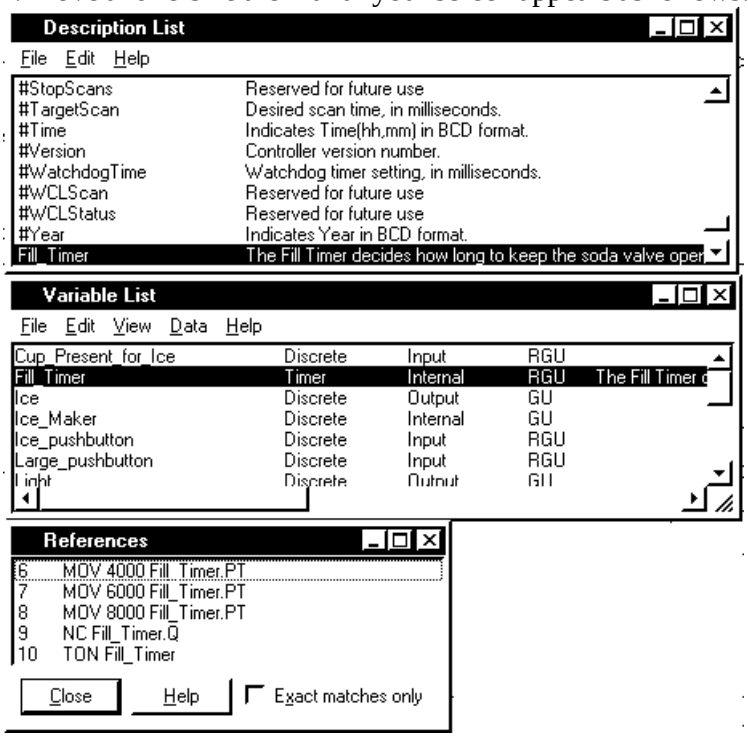
## 2.10.3 Using [References] Dialog Box with Other Dialog Boxes

Using only the [**Reference**] dialog box when you do not know where at least one instance of the desired variable is located is not the most convenient search method. You can also use the [**Find**] command to locate it, however, there is an even quicker method. You can use the [**References**] dialog box in conjunction with the [**Variable List**] and/or the [**Description List**] dialog box.

### ■ To use the references dialog box with other dialog boxes.

1. Open the [**Variable List**], [**Description list**] and [**References**] dialog boxes.

2. Move and re-size them until your screen appears as follows:




3. Click on the variable 'Fill_Timer' in the [**Variable List**] dialog box.

**Note:** **The displays of the [Description List] and [References] dialog box will change according to your selection. The [References] dialog box now displays every instance of the variable 'Fill_Timer'. Also, note that even though you change a dialog box's display, the logic program's display does not change. The corresponding point in your logic will appear when you select any variable line in the [References] dialog box.**

4. Click on the first line in the [**References**] dialog box. Your ladder logic program now displays that variable highlighted on the rung and the instruction you specified.

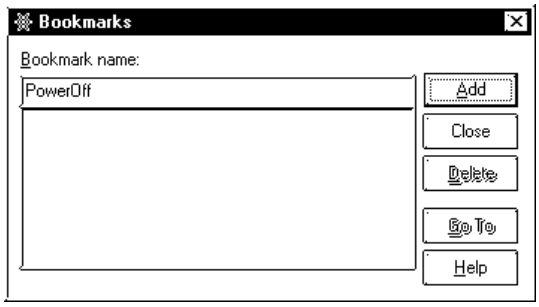
## 2.10.4 Using Bookmarks

If you are constantly referring back to a specific point in your ladder logic program, using a [**Bookmark**] saves you repeatedly scrolling the screen.

To set a [**Bookmark**], you must signify the exact point where you wish to return to. Anything you can select or highlight can be a [**Bookmark**]. For this demonstration, the [**NORMALLY CLOSED CONTACT (NC)**] instruction on rung 3 is set as a [**Bookmark**].

### ■ To set a [Bookmark]:

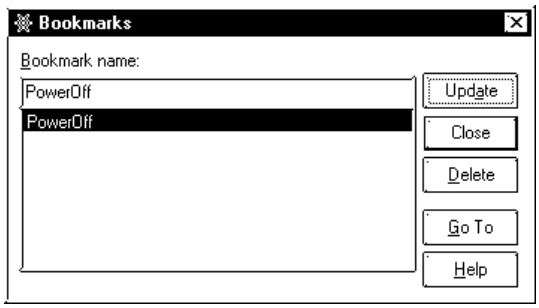
1. Click on the [**NC**] instruction on rung 3.
2. From the [**Search**] menu, select [**Bookmark**]. The [**Bookmarks**] dialog box appears.




3. Type '**Power Off**' in the [**Bookmark name**] field, then click on [**ADD**]. The [**Bookmark**] has now been set. Thus, whenever you select '**Power Off**' and click on [**Go To**] to return to your [**Bookmark**], you will return to the [**NC**] instruction on rung 3. If you wish to set a new [**Bookmark**], simply select a new point on the ladder logic and repeat steps 1 through 3. The Editor supports the use of multiple [**Bookmarks**].

### ■ To go to a [Bookmark]

1. From the [**Search**] menu, select [**Bookmarks**]. The [**Bookmarks**] dialog box appears.




2. Select a [**Bookmark Name**] from the list, then click on [**Go To**]. Wherever you are in your ladder logic program, the Editor automatically takes you back to where you placed the [**Bookmark**].

**Note:** **You can use the [CTRL] + [M ] keys to open the [Bookmarks] dialog box.**

### ■ To change the position of a [Bookmark]

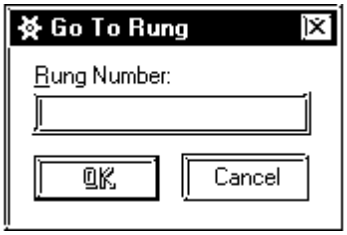
1. Select the new position in the ladder logic program.
2. Select the [**Bookmark name**] you wish to re-position.
3. Click on [**Update**] in the [**Bookmarks**] dialog box.

## 2.10.5 Using the [Go To Rung] Command

The [**Go To Rung**] command allows you to move the "focus" to a specified rung in your ladder logic program.

### ■ To use the [Go to Rung] command

1. From the [**Search**] menu, select [**Go To Rung**] and the following dialog box will appear:



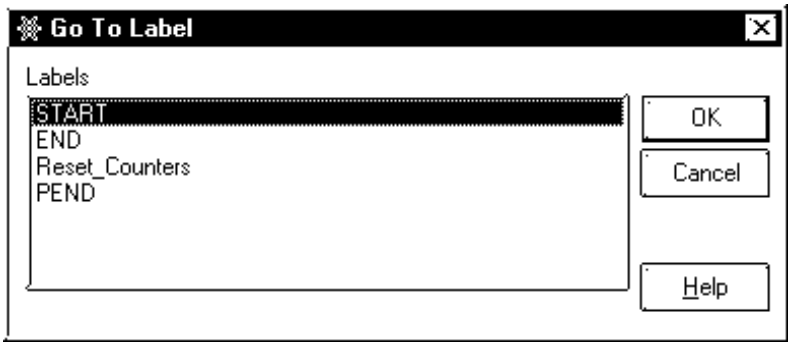

2. Enter a [**Rung Number**].
3. Click on [**OK**]. You are now positioned at the specified rung.

## 2.10.6 Using the [Go To Label] Command

The [**Go to Label**] command allows you to jump to a specific "label" in your ladder logic program.

### ■ To use the [Go to Label] command:

1. From the [**Search**] menu, select [**Go TO Label**]. The [**Go To Label**] dialog box appears:




2. Select the label to go to.
3. Click on [**OK**]. You are now positioned at the specified label.

***< Summary >***

This section has explained how to use [**Find**], [**References**], [**Bookmark**], [**Go To Rung**] and [**Go To Label**] commands.

# 2.11 I/O Configuration

Once you have finished constructing a ladder logic program, you must assign I/O to selected variables. In this tutorial, variables were created first and I/O assigned after the ladder logic program was completed. This was done in order to present the various features of the Editor in a logical order. If you know what your I/O will be before beginning programing, you can specify your I/O first and then assign it to your variables as you create your program. Both methods are demonstrated in this section.

## 2.11.1 Assigning Variables to I/O

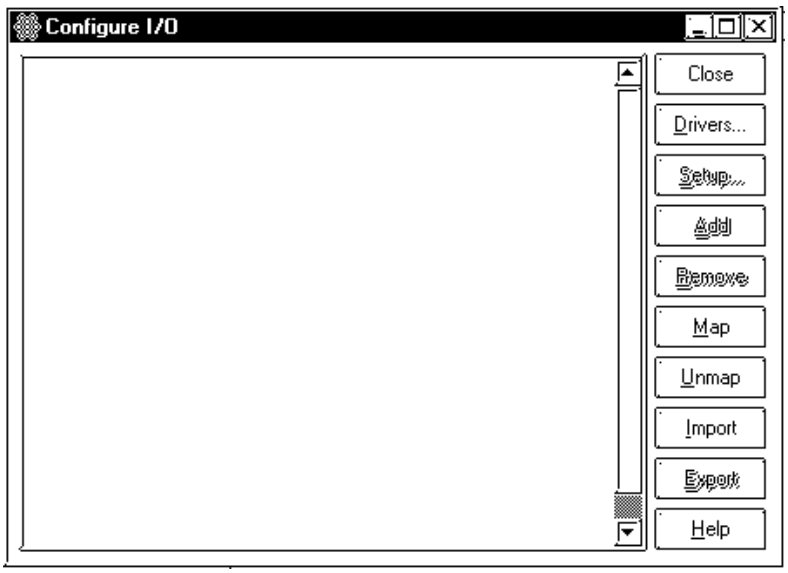
Once you have created variables in a ladder logic program, there are a number of methods you can use to assign them to your I/O.

The "Ice_pushbutton", "Large_pushbutton", "Medium_pushbutton", and "Small_pushbutton" will be placed on the GLC screen for touch-panel inputs. These buttons are not assigned to the terminals.

| Variable Name | Terminal Type | Terminal No. |
|---|---|---|
| Power_ON_pushbutton | Input | I0 |
| Cup_Present_for_Ice | Input | I2 |
| Power_OFF_pushbutton | Input | I6 |
| Light | Output | Q0 |
| Ice | Output | Q1 |
| Soda_valve | Output | Q2 |

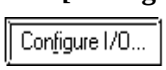
### ■ To open the [Configure I/O] window:

From the [**Data**] menu, choose [**Configure I/O**] and the following window will appear.



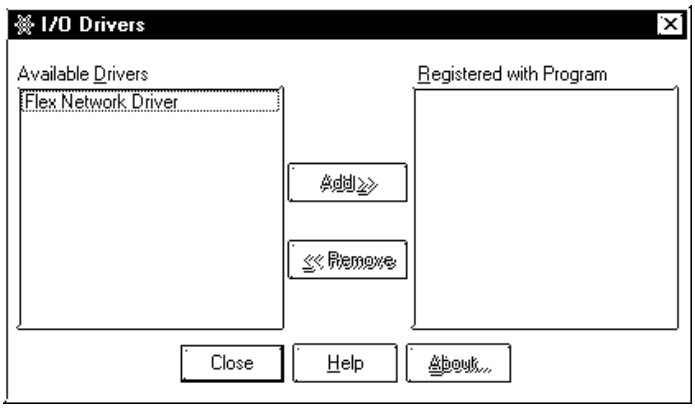

* The [Import] and [Export] buttons are not supported with the current setting.

Currently there are no drivers registered with your program.

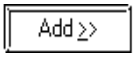

**You can also open the [Configure I/O] dialog box by clicking on**  **on the tool bar or by clicking on**  **in the [Variable Type] dialog box.**

## ■ To specify a driver:

1. Click on [**Drivers**] in the [**Configure I/O**] dialog box. The [**I/O Drivers**] dialog box appears.




The left side of this dialog box lists all [**Available Drivers**]. The right side of the dialog box lists the drivers [**Registered with Program**]. Currently there are no registered drivers.

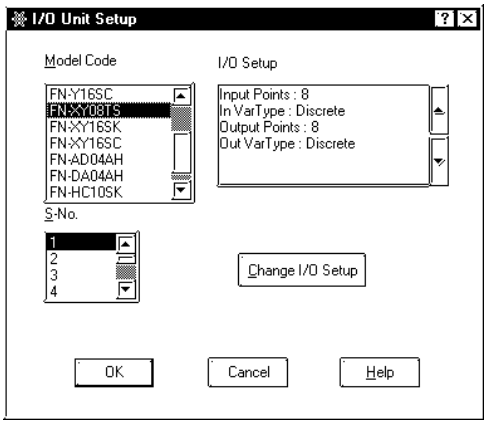
2. Select '**Flex Network Driver**' in the [**Available Drivers**] section of the [**I/O Drivers**] dialog box.
3. Click on 

, or double-click on the driver's title and the selected driver will appear in the [**Register with Program**] list.
4. Click on [**Close**]. The [**Configure I/O**] window shown on the following page will appear.

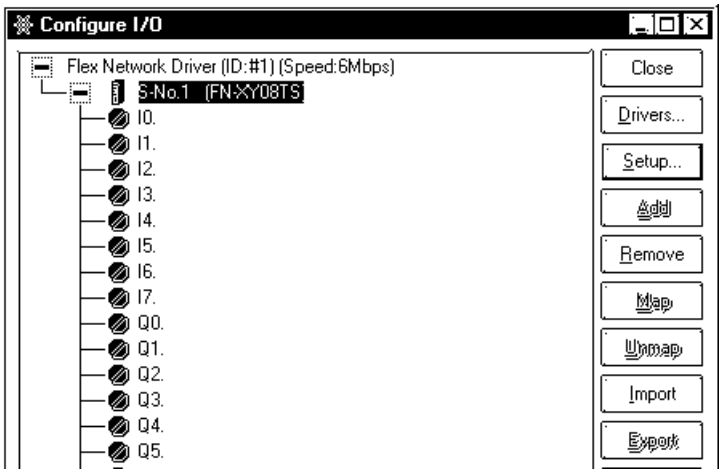
In the default settings, the "Model Type" is set to "FN-X16TS", and the "S-No. (Machine Number)" is set to "1". In this example, set the "Model Type" to "FN-XY08TS", and "S-No." to "1". The FX-XY08TS features 8 points each for input and output.

## ■ To set up the Flex Network driver:

1. Select **"S-No. 1 (FN-XY08TS)"**.
2. Click on [**Setup**]. The [**I/O Unit Setup**] dialog box appears:

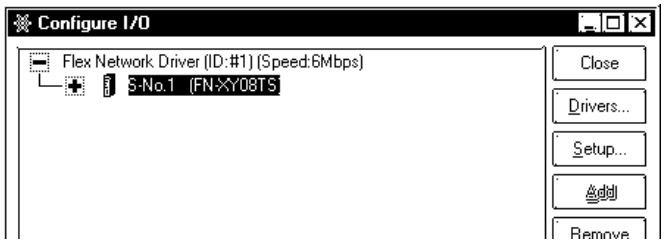



3. Select **"FN-XY08TS"** from the **"Model Code"** field.
4. Click on [**OK**]. The [**Configure I/O**] window appears as follows:




Displayed underneath 'S-No.1 (FN-XY08TS)' are 8 input terminals and one output terminal (for a 8-bit word) associated with the Flex Network module displayed. You will assign variables to them later in this tutorial.

5. Click on ⊟ next to 'S-No.1 (FN-XY08TS)'. The terminals are hidden and ⊞ appears in place of ⊟ .

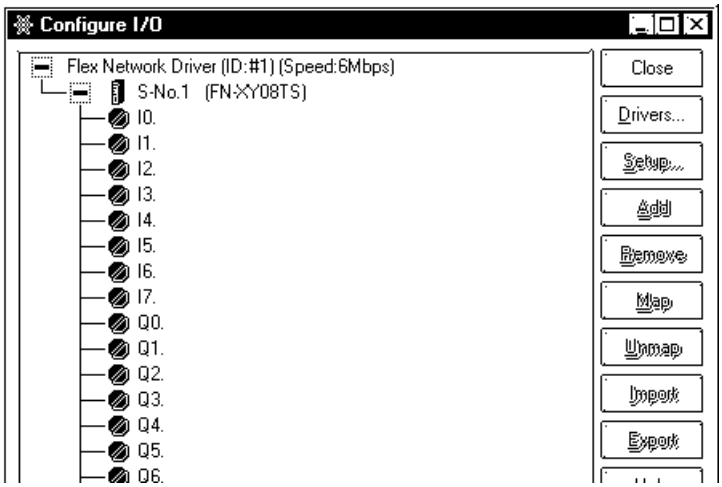



6. Up to 63 units (when 2 lines are used) can be connected with the Flex Network driver. Use the same method for selecting a module for another unit.

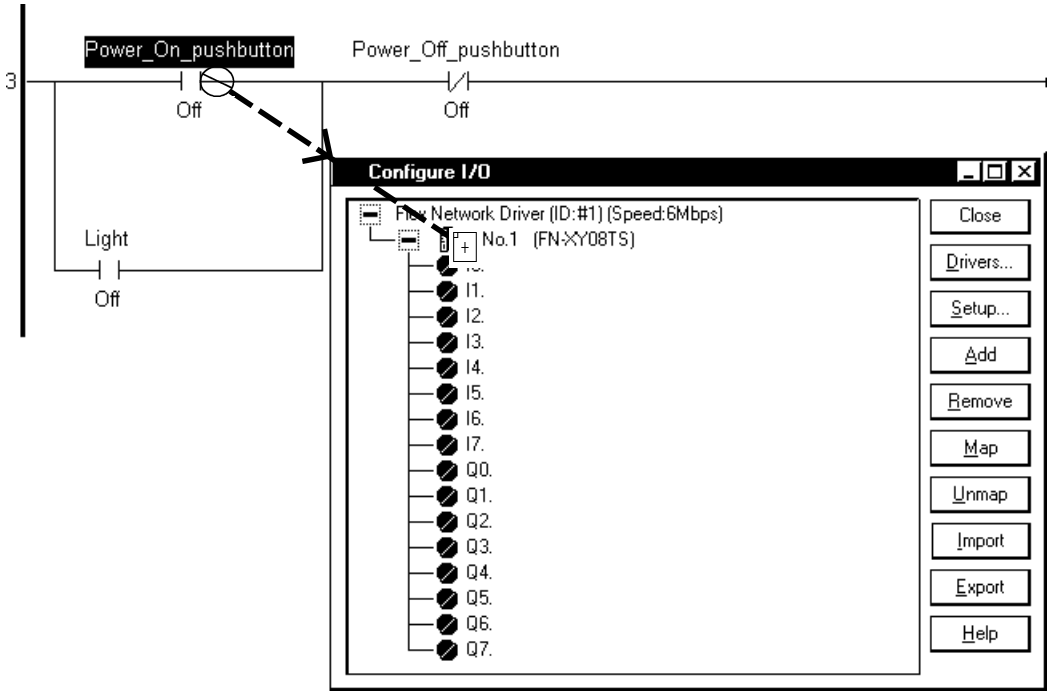
## ■ To click and assign variables to the I/O terminals:

1. Click on ✚ next to 'S-No.1 (FN-XY08TS)'. The [**Configure I/O**] window appears as follows:

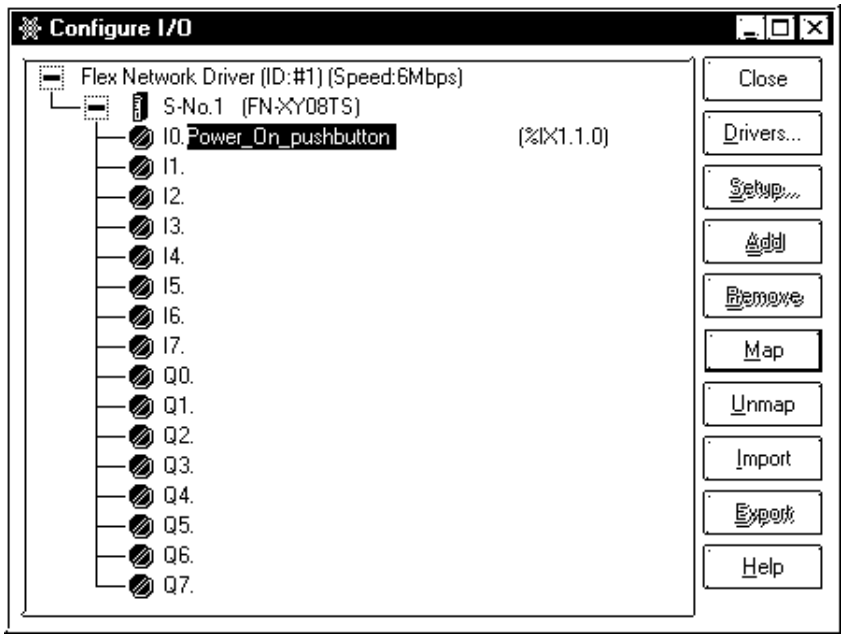
   Now that a driver is set up, you can assign variables to the I/O terminals. The variables used come directly from the program created in the tutorial. There are several ways to assign variables to I/O. You can use the first 8 terminals for input with S-No.1 (FN-XY08TS).



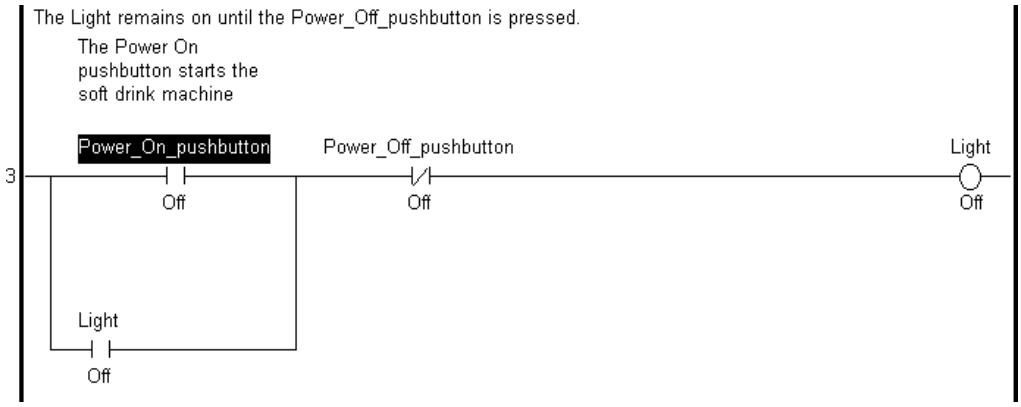

2. Locate the variable 'Power_On_pushbutton' on the NO instruction of rung 3.
3. Click and drag 'Power_On_pushbutton' toward terminal I0. As well as when inserting branches, note that your cursor initially becomes a ⃠ . When the cursor is in this state you cannot assign the variable to any I/O terminal.



4. Drag the cursor over terminal 0 and release the mouse. The variable ‘Power_On_pushbutton’ is now assigned to terminal I0.



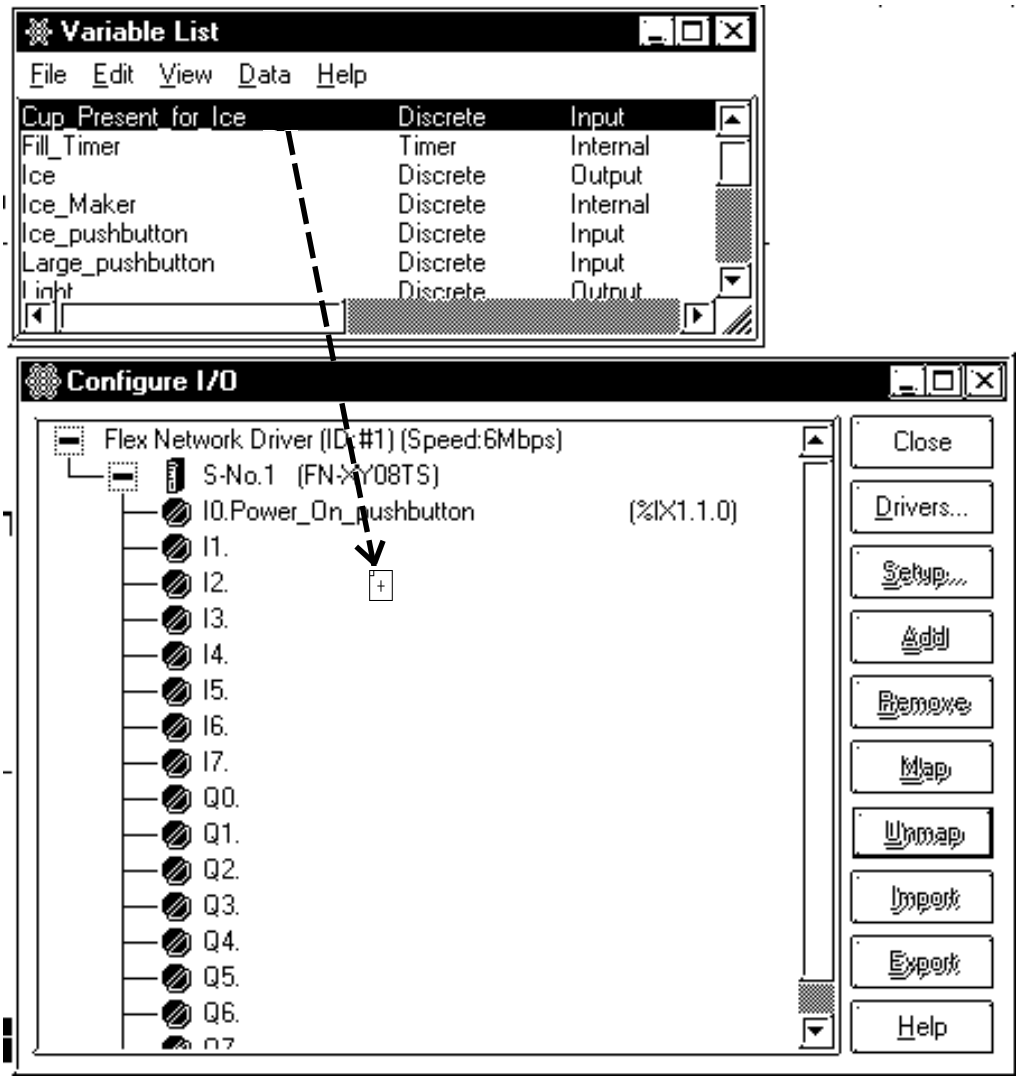

The variable ‘Power_On_pushbutton’ on the NO instruction of rung 3 now has a series of digits and letters above it. This is the IEC I/O address of that variable.



## ■ To click and drag variables to I/O terminals from the [Variable List] dialog box:

1. Open the [**Variable List**] dialog box. The [**Configure I/O**] window should still be open.
2. Arrange the dialog boxes so that both can be viewed.
3. From the [**Variable List**] dialog box, click and drag the variable 'Cup_Present_for_Ice' to terminal I2 in the [**Configure I/O**] window.
4. Release the mouse. The variable 'Cup_Present_for_Ice' is now assigned to input terminal I2.

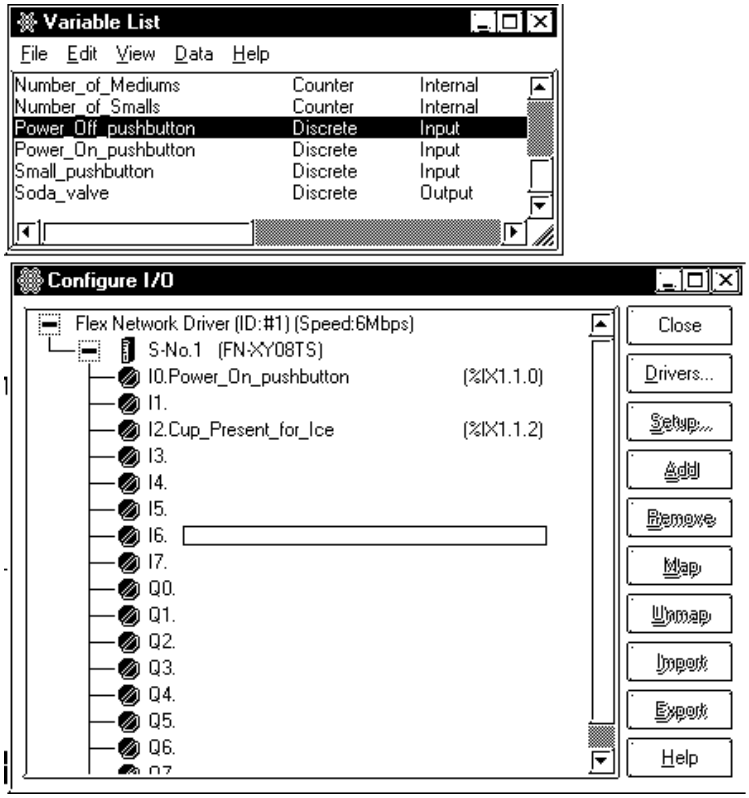
Note: **You can also use the above procedure to assign variables to I/O from the [Description List] dialog box.**



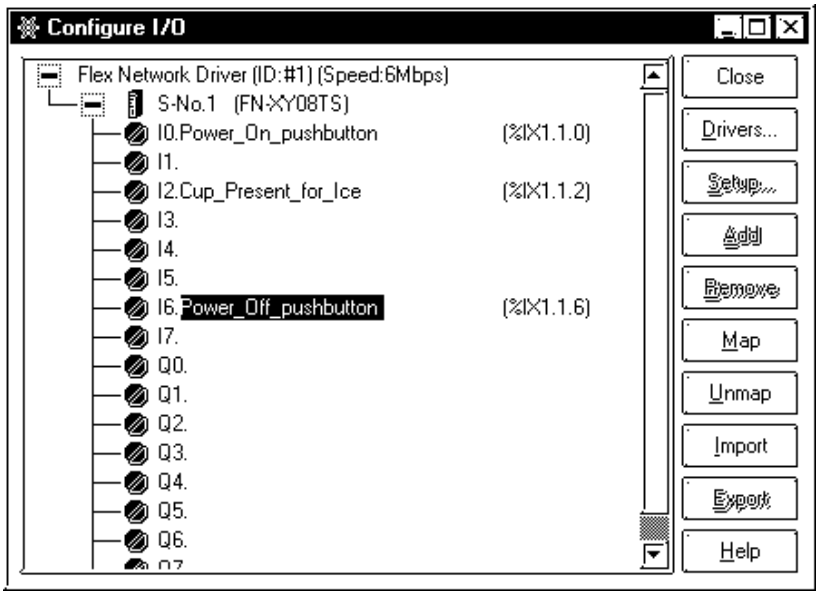

Caution **When you assign (click and drag) a variable to [Configure I/O] from the [Variable List] or [Description List] window, that I/O attribute is enabled and any other variable attribute will be changed to Input/Output.**

## ■ To assign variables via text entry:

1. Click on terminal I6.
2. Press the [**Enter**] key. The terminal test field is activated.




3. Type ‘Power-Off-pushbutton’.
4. Press the [**Enter**] key. ‘Power-Off-pushbutton’ is now assigned to input terminal I6.

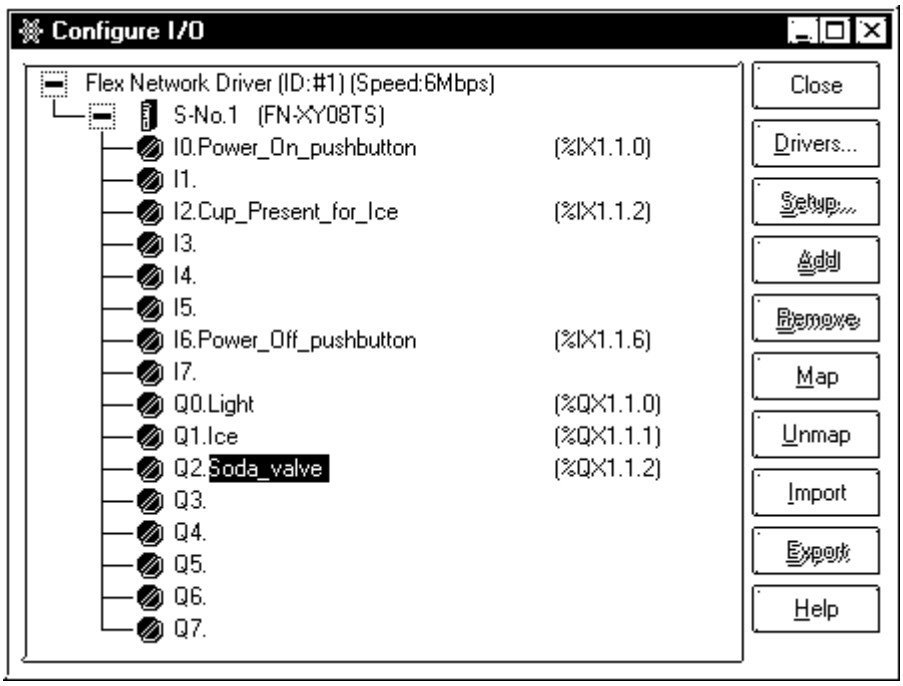



**Note:** **When variables are assigned to I/O via text entry, the variables will be automatically listed in the [Variable List] dialog box.**

Assigning variables to output terminals is the same as assigning them to input terminals. Use the above procedures to assign variables from the following table to the input and output terminals.

| Variable Name | Terminal Type | Terminal # |
|---|---|---|
| Light | Output | Q0 |
| Ice | Output | Q1 |
| Soda_valve | Output | Q2 |

The input and output modules are displayed in the [Configure I/O] dialog box as shown here:



## 2.11.2 Unassigning Variables from the [Configure I/O] Dialog Box

### ■ To unassign a variable from the [Configure I/O] window:

1. Click on terminal I0 in the [**Configure I/O**] window.
2. Click on [**Unmap**]. The 'Power_On_pushbutton' is now unassigned from terminal I0 and can be assigned to any other terminal you select. In this tutorial, assign it back to terminal I0.

## 2.11.3 Using Variables Assigned to I/O with Instructions

The easiest way to configure I/O for new programs is to type the variables directly into the terminals. They are then automatically created, configured and mapped to the correct I/O point. In this case, when you configure your I/O first and then construct your ladder logic program, creating your I/O points is explained.

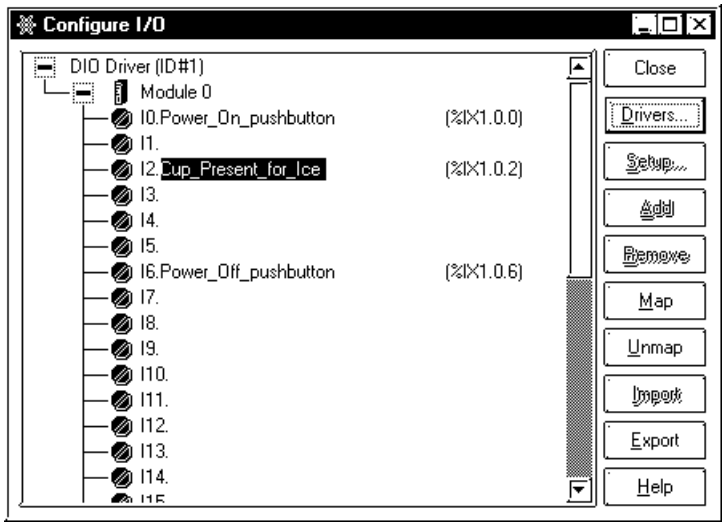
### ■ To use variables assigned to I/O with Instructions:

1. Click the target variable and drag to the I/O terminals as described above to assign variables to the input and output terminals of your driver.
2. Construct your ladder logic program.
3. Click and drag the variables from the [**Configure I/O**] dialog box to the instructions you want I/O assigned to.

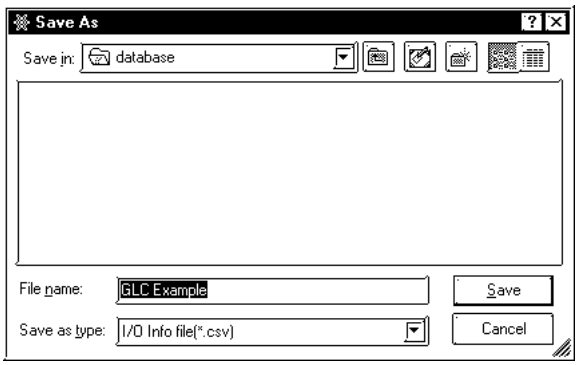
## 2.11.4 Converting I/O Configuration Data

You can automatically convert variables allocated to the DIO unit to the Flex Network. The Flex Network FN-X32TS, FN-XY16SK and FN-XY16SC I/O 32 point units support this feature. Prior to converting this data, the following setup steps are required.

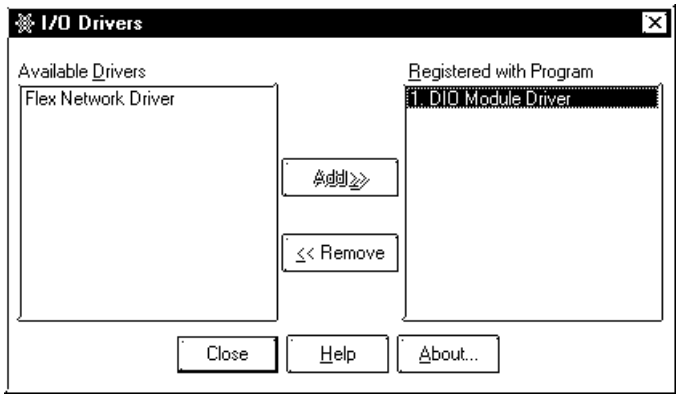
This example converts data from the DIO unit to a Flex Network [FN-XY16SK] unit.



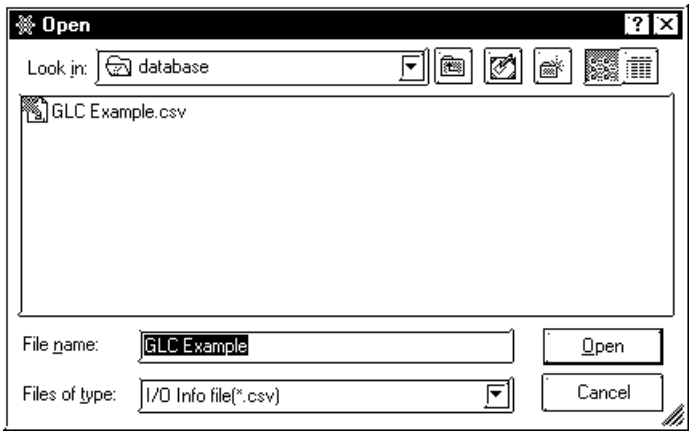

1. Click on the [**Configure I/O**] window's [**Export**] button. Then, export (save) the DIO driver's allocated variables to a CSV file.



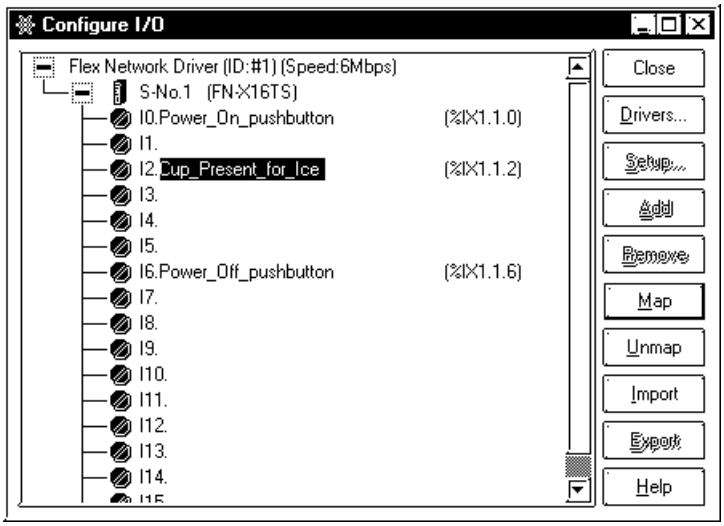

2. Click on the [**Driver**] button and use the [**Remove**] button to delete the DIO unit driver. Next, use the [**Add**] button to add the Flex Network driver.



3. Click on the Flex Network Driver's [**S-No.1(FN-XY16SK)**] item, and then click on the [**Import**] button. Select the previously saved CSV file and click on [**Open**].




4. After the variables are read in from the CSV file, allocate them to the Flex Network Driver.




***< Summary >***

This section explained how to:

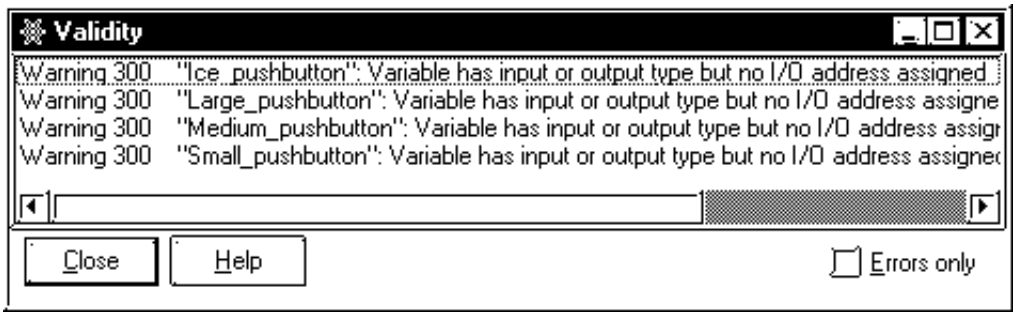
- select an I/O driver,
- configure the Flex Network driver,
- use variables assigned to I/O.

# 2.12 Checking the Validity of a Program

Before running an Editor ladder logic program online, use a validity check to make sure the program is free of errors.

## ■ To run a validity check:

- From the [**File**] menu, select [**Check Validity**] and the following dialog box will appear.




The [**Validity**] dialog box lists all errors and possible trouble spots the Editor can detect in your program. Trouble spots are listed as “warnings”.

In the lower right hand corner of the dialog box is a check box marked [**Errors only**]. If this box is selected, only the “errors” that the Editor detects in your program are displayed; the “warnings” are not. The Editor can run a program that contains “warnings” in the Controller, however, it cannot run a program that contains errors. These errors must be corrected first.

**Note: A validity check can also be performed by clicking on ✓? in the tool bar.**

The [**Validity**] dialog box displays “errors” and “warnings” in the order they appear in your ladder logic program. In other words, the “errors” in rung 1 are presented first, then rung 2 and so on. If you double-click on the “errors” or “warnings” in the [**Validity**] dialog box you can go directly to the problem.

- If it is a logic problem, that part of your program is displayed.
- If it is a problem with assigning I/O, the [**Configure I/O**] dialog box is displayed.

As previously mentioned, there can be a variety of “error” types displayed in the [**Validity**] dialog box. Your validity check will show the following error.

Error 200 Rung 9: Parameter should be a Discrete

## ■ To fix an error:

1. Double-click on the “error” line in the [**Validity**] dialog box. The [**Instruction Parameter Box**] of the instruction on rung 9 is highlighted, indicating there is no variable assigned to it.
2. Enter ‘Soda_valve’ as the instruction variable.

Reference *For more information on specific errors and warnings, refer to the **Editor Help** system or “**Appendix A: Errors and Warnings**” in this manual.*

When you have corrected the “errors” listed in the [**Validity**] dialog box, run a validity check again. Any errors that exist are displayed. If they have all been corrected your program can be written to the Controller.

### *< Summary >*

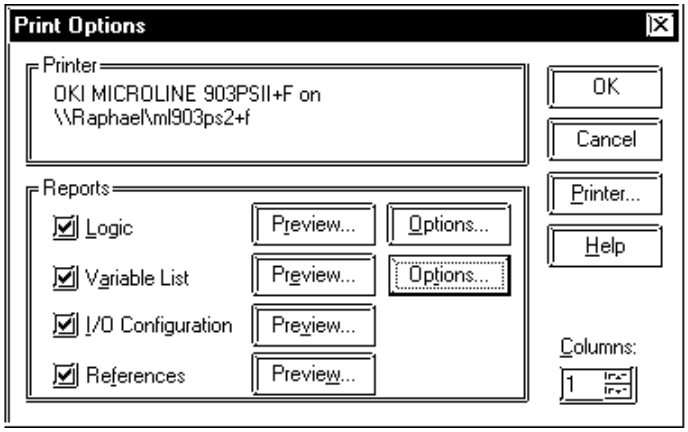
In this section you have learned how to check the validity of an Editor ladder logic program. The preparation for transferring a program to the GLC for execution is complete. The details of the procedures hereafter are explained in 3.1 Configuring the GLC Controller.

# 2.13 Printing Your Ladder Logic Program

With Editor, you can print different aspects of your ladder logic program.

## ■ To print a ladder logic program:

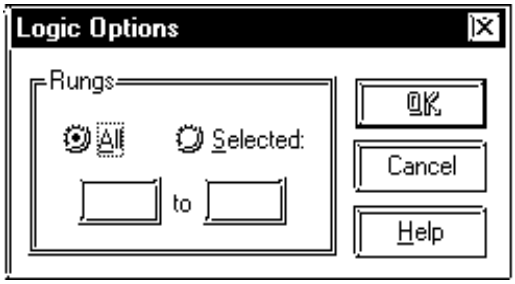
From the [**File**] menu, select [**Print**] and the following dialog box is displayed. You can view the logic program on the screen before it is printed using the Preview function.




You can select the number of columns (1 to 4) into which your report will be formatted. Under the [**Reports**] section there are four check boxes labelled [**Logic**], [**Variable List**], [**I/O Configuration**] and [**References**]. These check boxes provide the following options when printing your ladder logic program:

◆ [**Logic**]:

This option allows you to print the rungs of your ladder logic program. If you click on [**Options**] next to it, the following dialog box appears:

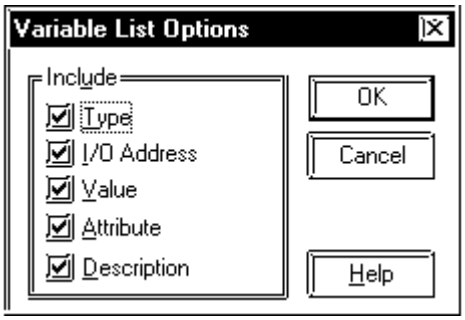



Select [**All**] to print all the rungs of the program or, click on [**Selected**] and type in the range of rungs you wish to print.

Use the [**View**] menu to adjust the logic program's printout size.

◆ [**Variable List**]:

This option allows you to print a variable list. Click on [**Options**] to select the items you wish to include in that variable list.




| Option | Description |
|---|---|
| Type | Displays the variable type. |
| I/O Address | Displays the I/O addresses of all assigned variables. |
| Value | Displays the data value of all variables. |
| Attribute | Displays the Retentive and Global settings |
| Descriptions | Displays any descriptions given to the variables. |

◆ [**I/O Configuration**]:

This option allows you to print your I/O configuration.

◆ [**References**]:

This option allows you to print a cross reference report showing all instances of all variables.

**You can also print your program by clicking on**  **in the tool bar.**

***< Summary >***

This section explained how to select which aspects of your ladder logic program you wish to print.

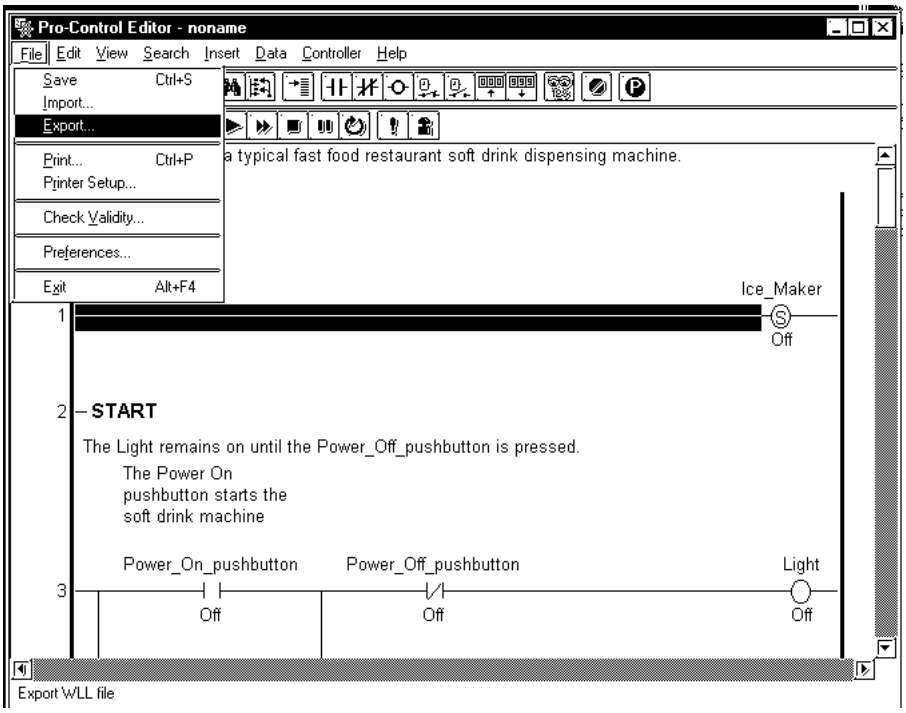
# 2.14 Importing/Exporting a Logic Program

The Editor allows you to export a logic program exclusively and save it as a Logic Program File (*.wll).

The logic program file can be imported and used as a logic program in another Project File (*prw) or vice versa.

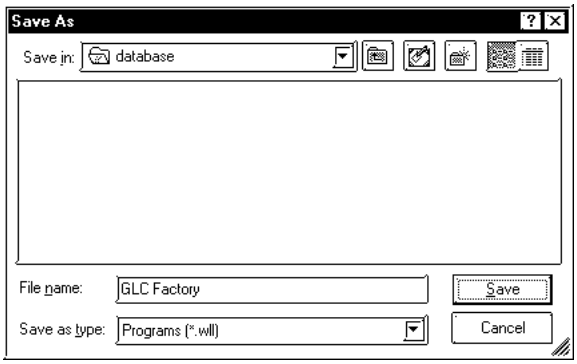
## ■ To Export a Logic Program

1. Select the [**Export**] command from the [**File**] menu.



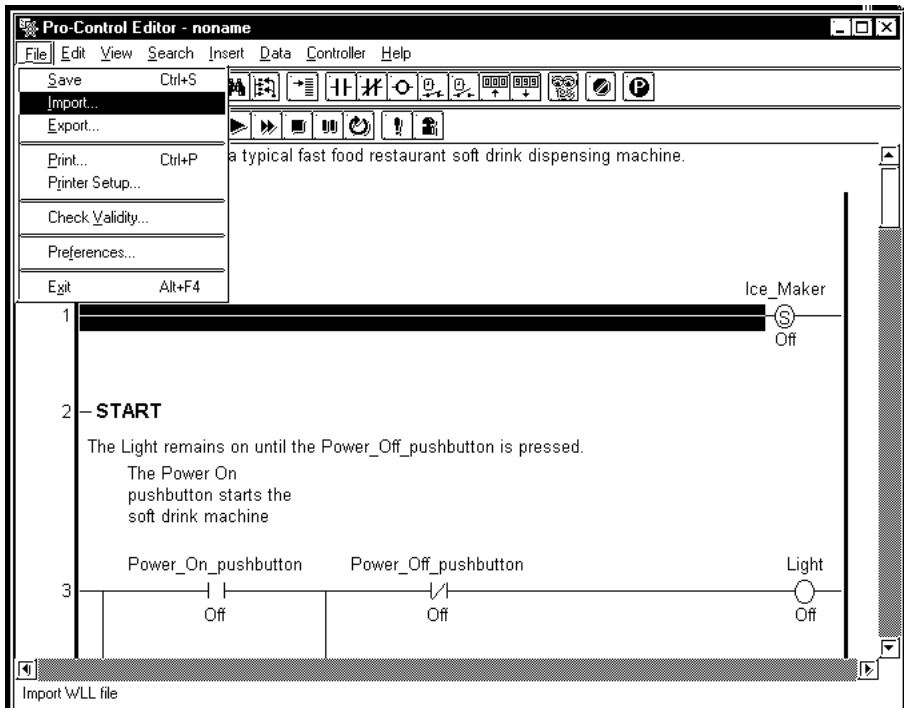

2. Enter a file name in the [**Save As**] window.
3. Click [**Save**].

   The Logic Program is saved in WLL format.



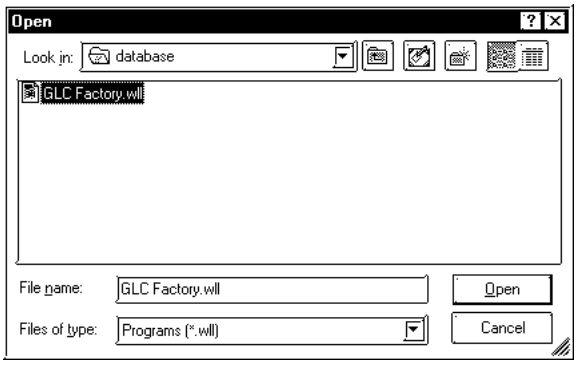
## ■ To Import a Logic Program

1. Select the [**Import**] command from the [**File**] menu.




2. Select the WLL file you want to import in the [**Open**] window.
3. Click [**Open**].

   The specified Logic Program is imported, and the variables used in the Logic Program are registered to the Variable List.




4. Saving the Logic Program will register a global variable in the Symbol Editor as a logic symbol.

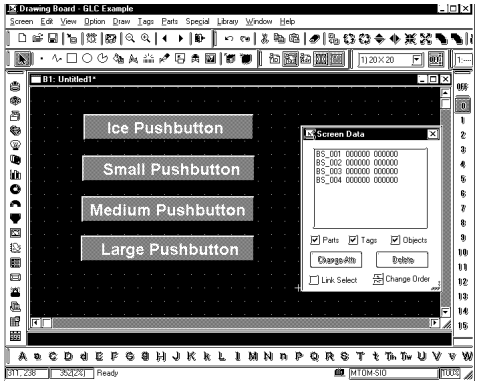
Reference *4.7 Symbol Editor in the Operation Manual*

### *< Summary >*

In this section, you have learned how to import and export a logic program.

# 2.15 Developing a Screen Program

Create the "Ice_Pushbutton", "Large_Pushbutton", "Medium_Pushbutton", and "Small_Pushbutton" with the GP-PRO/PB III. The illustration below is the completed sample screen.

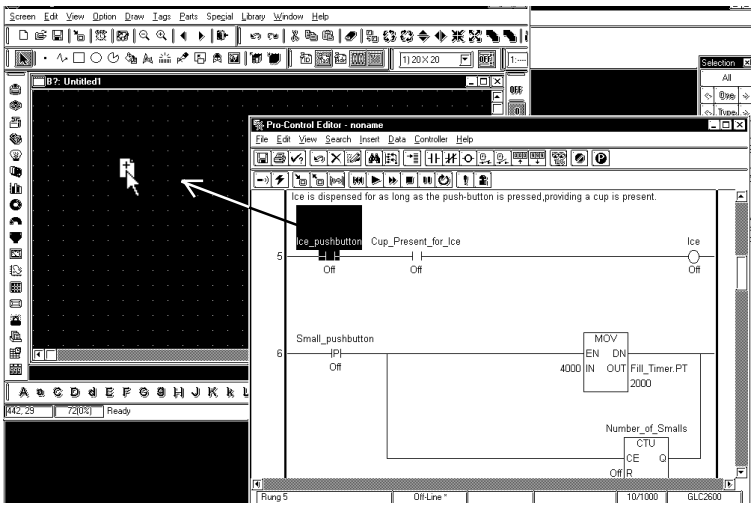



## ■ To Start the GP-PRO/PB III

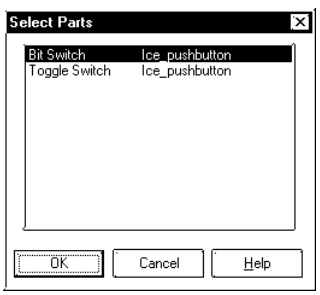
1. In the Project Manager's window, click [**Draw/Screen**] to activate the GP-PRO/PB III.
2. Click [**Screen/New**] on the Menu Bar. Check that "Base Screen" is selected, and click the [**OK**] button.

## ■ To Draw using Drag and Drop Operations

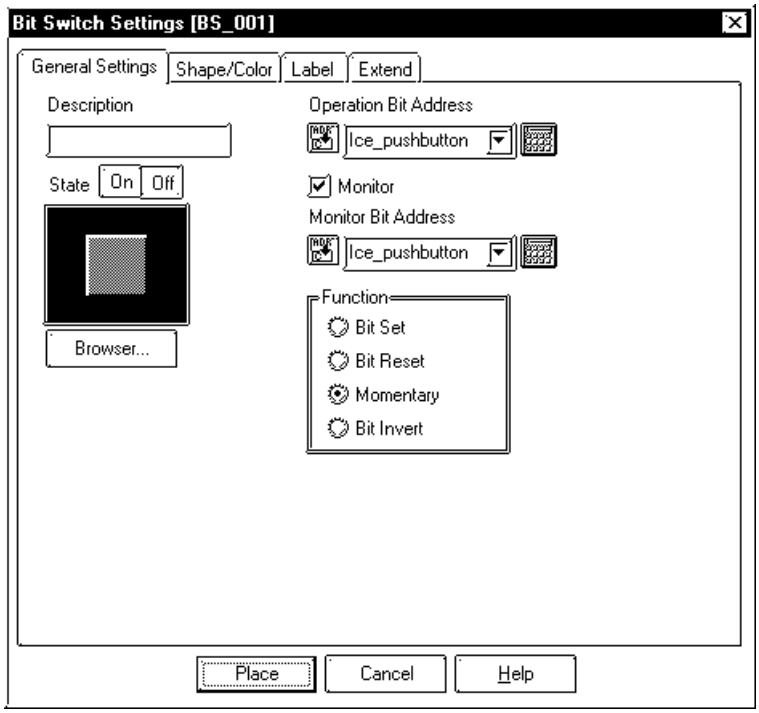
1. Select the "Ice-pushbutton" in the Logic Program, and drag it to the Screen Editor of the GP-PRO/PB III.



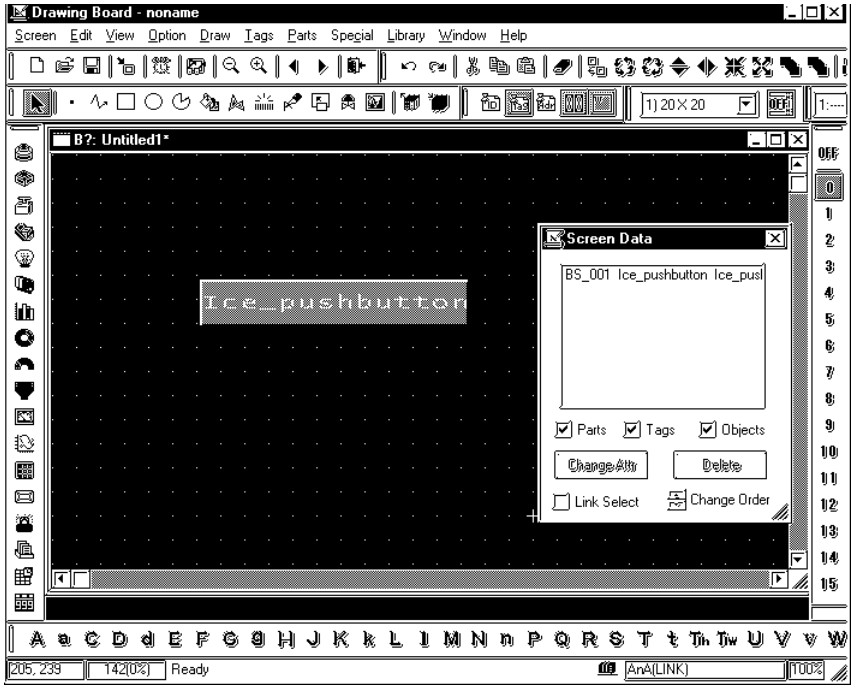

2. Drop the button on the Screen Editor. The "Select Parts" dialog box will appears on the screen. Select "Bit Switch", and click the [**OK**] button.



3. The "Bit Switch Settings" dialog box appears on the screen. Select "Momentary" from the "Function" field. Check that the "Operation Bit Address" is set to "Add Ice", and then click the [**Place**] button to place the pushbutton.




4. The "Add Ice" is completed. Create the "Large_pushbutton", "Medium_pushbutton", and "Small_pushbutton" using the same procedure.




MEMO

# 3 Running the Ladder Logic Program

Once you have developed a ladder logic program that is free of errors, it can be run by the GLC Controller.

This chapter explains how to configure the GLC Controller, send (write) a program to it and run the program online.

## 3.1 Configuring the GLC Controller

Before writing a ladder logic program to a GLC Controller, please be sure that the controller is configured properly. For a controller running on a GLC platform there are three aspects of the Controller which can be configured: [**Tuning**], [**Connection**] and [**Memory**].

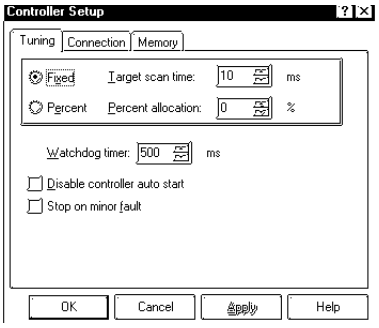
### ■ To Configure the Controller:

From the [**Controller**] menu, select [**Setup**], which calls up the following screen.

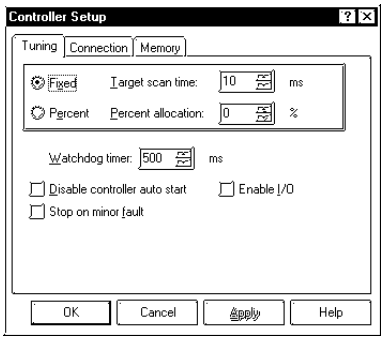
#### ◆ Tuning

Select the **[Tuning]** tab.

<GLC100/300>




<GLC2000 Series>




When you set parameters on the [**Tuning**] tab, you are setting the parameters the ladder logic program uses when it is written to the GLC Controller. From this point onward, whenever this particular program is run, the GLC Controller uses these settings, unless they are changed. These settings are unique to this program. Controller [**Tuning**] options are explained below.

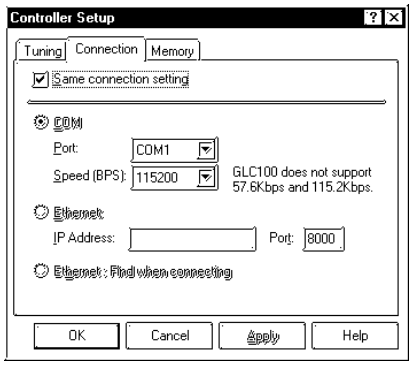
| Option | Description |
|---|---|
| **Target Scan Time** | In [**Target Scan Time**] (System Variable: "#TargetScan"), enter the amount of time in milliseconds you would like each scan of your program to take. (Note): If the logic time exceeds the 50% of the scan time, the "Scan" operation is not guaranteed. Specify the setting in 10-ms increments. |
| **Percent Allocation** | In [**Percent allocation**] (System Variable: "#PercentAlloc"), enter a value in % to designate the scan time by the percentage of the whole dealing time. The 1-ms place of the calculated scan time is round up. |
| **Watchdog Timer** | When a logic program alarm occurs that delays the scan so that the value entered here is exceeded, a Major Fault alert occurs. The system variable #WatchdogTime also can be used for this setting. See Pro-Control Editor User Manual Chapter 3 System Variables |
| **Disable Controller Auto Start** | Only when the GLC OFFLINE mode's "MODE WHEN POWER IS ON" selection is set to [**DEFAULT**] is this feature enabled.*1 When the controller is restarted after being stopped, this feature will automatically prevent the Logic Program from restarting. The system variable #DisableAutoStart can also be used for this setting. See Pro-Control Editor User Manual Chapter 3 System Variables |
| **Stop on Minor Fault** | This setting designates if the logic program is stopped when a minor controller fault occurs. The system variable #FaultOnMinor can also be used for this setting. See Pro-Control Editor User Manual Chapter 3 System Variables |
| **Enable I/O (GLC2000 Series)** | This function enables the inputs/outputs to the GLC main unit and external I/O of the I/O unit. In normal operation, the input/output of the external I/O is disabled when the GLC is set to RUN mode after performing a Logic Program download. For safety reasons, this function prevents the possibility of accidental startups of machines caused by errors in operation and logic programs. |

**1 The [Mode When Power is ON] setting is set via the [GP SETUP] -> [CONTROLLER MENU] -> [CONTROLLER SETTING1] screen. The setting set here (RUN orSTOP) is given priority and the setting used in Pro Control Editor is ignored.*


- **For details on Target Scan Time and Percent Allocation, refer to Chapter 1: Controller Features in the Pro-Control Editor User's Manual.**
- **For details on the system variables, refer to Chapter 3: System Variables in the Pro-Control Editor User's Manual.**
- **The "Enable I/O" feature can be selected when starting and stopping the controller. For details, refer to 3.2 Starting and Stopping the Controller.**

### ◆ Connection

Click the [**Connection**] tab.




- **Same connection settings**

  When this item is selected, the settings designated with the [**Transfer/Transfer Settings**] on the GP-PRO/PB III become active.

- **COM**

  Designate the Port and Speed for serial communications. Note that the GLC100 does not support 115.2 kbps.

  This enables logic program writing/reading and monitoring mode execution via on Ethernet network.

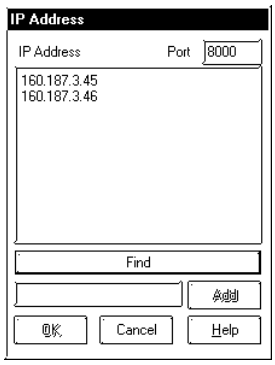
### ■ Transmission Settings

Select [**Setting**] at the [**Controller Menu**]. Select “Ethernet” or ”Ethernet: Automatic Search” with the [**Communication Setting**] on the [**Setting Menu**]. Selecting the "Use the Transfer Settings" option enables the "Transfer" settings on the GP-PRO/PB III.

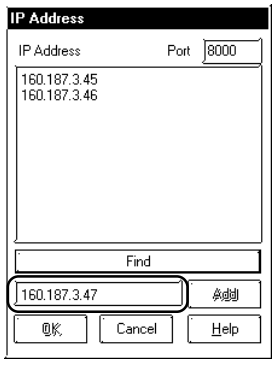
| Item | Content |
|---|---|
| **Ethernet** | Input the IP address and port no. of the GLC for communication. Communication via Ethernet begins when you execute [**Write to controller**], [**Read from controller**] or [**Monitoring Mode**]. |
| **Ethernet: Automatic addressing** | A list of GLC units connected to the Ethernet network is displayed when you click on [**Write to controller**], [**Read from controller**] or [**Monitoring Mode**]. Communication begins when you select the GLC for communication and click on [**OK**]. Multiple GLC can be selected with [**Wite to controller**]. |

**(Example) Ethernet: Automatic Detect**

A list of the GLC units currently connected to the Ethernet network will appear.

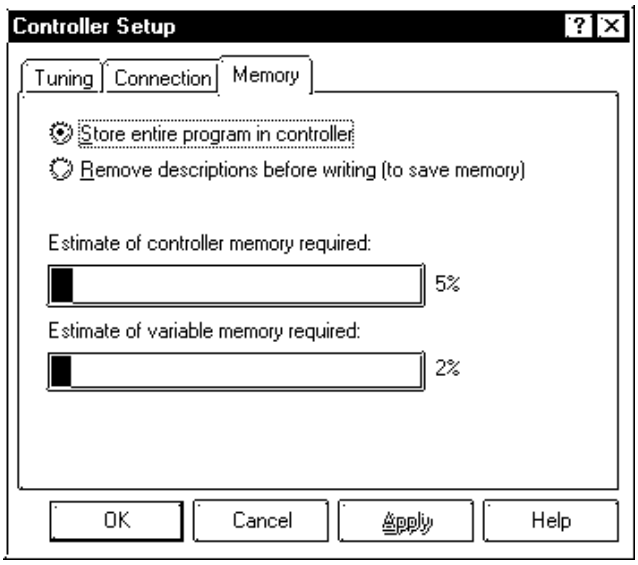



It is possible to search for GLC with a designated address by designating the IP address and selecting [**Add**].



### ◆ Memory

The [**Memory**] tab shows the percentages of [**Estimate of controller memory required**] and [**Estimate of variable memory required**] with bar graphs.




[**Store entire program in controller**]

Transmits the entire logic program, including comments. Comments for the logic program can be read when reading is done from the GLC.

[**Remove descriptions before writing (to save memory)**]

Reduces the size of the file you are downloading to the GLC, therefore, when the file is uploaded from the GLC, there will not be any description data.

[**Estimate of controller memory required**]

Shows the relationship of the current program's memory to the GLC's usable memory, as a percentage.

[**Estimate of variable memory required**]

Shows the relationship of the total memory of all variables currently registered to the GLC's usable memory, as a percentage.

## 3.1.1 Writing to the Controller

After you have completed creating a ladder logic program with the Editor and it is free of errors, you can write it to the GLC and run it online.

To write a logic program to a GLC, you can either:

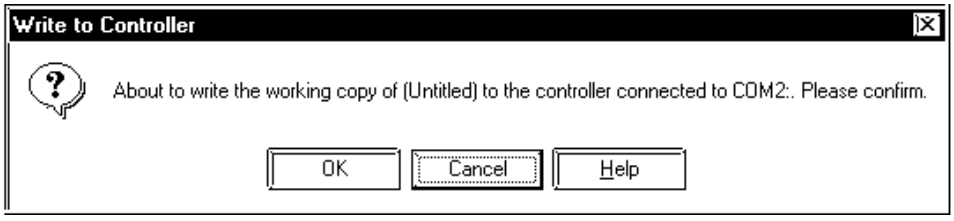
- transfer the screen data and logic program via the "Transfer" window of the GP-PRO/PB III.
- transfer the logic program exclusively via the Editor.

Make sure to set up your GLC before writing a logic program to a GLC. To set up a GLC, transfer the system along with a Project File via the "Transfer" window of the Editor. For details on transferring data, refer to Chapter 7: Transferring Data in the GP-PRO/PB III Operation Manual.

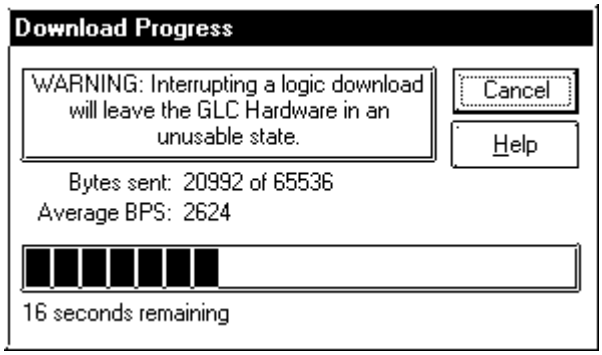
This section describes how to transfer a logic program exclusively using the Editor.

### ■ To Write to the Controller:

1. From the [**Controller**] menu, select [**Write to Controller**] and the following dialog box appears, prompting you for your OK before writing to the Controller. Before a program is written to the Controller, the Editor automatically runs a validity check. A program containing errors cannot be written to the Controller.




2. Click on [**OK**]. The [**Download Progress**] dialog box appears and displays the status of the download of data to the GLC.



**Note:**

- **The Flex Network driver software etc. will be downloaded (if needed) when you write your. PRW file to the controller. If no changes in the driver have occurred since the last download, the download of the driver is skipped.**
- **The size of the downloaded file can be reduced by removing descriptions before transferring.**

Reference *3.1 Configuring the GLC Controller*

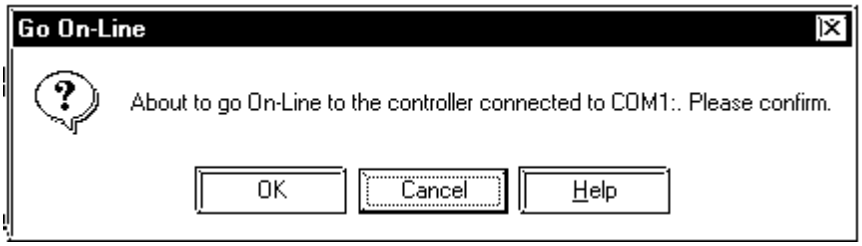
**Important**

***With GLC300/GLC2000 Series units, previous data will be erased when the program is written to the Controller.***

## 3.1.2 Going to Monitoring Mode

### ■ To Go to Monitoring Mode:

1. From the [**Controller**] menu, select [**Monitoring Mode**]. A dialog box then appears asking if you wish to go online.




2. Click on [**OK**]. You are now online and can start the Controller.

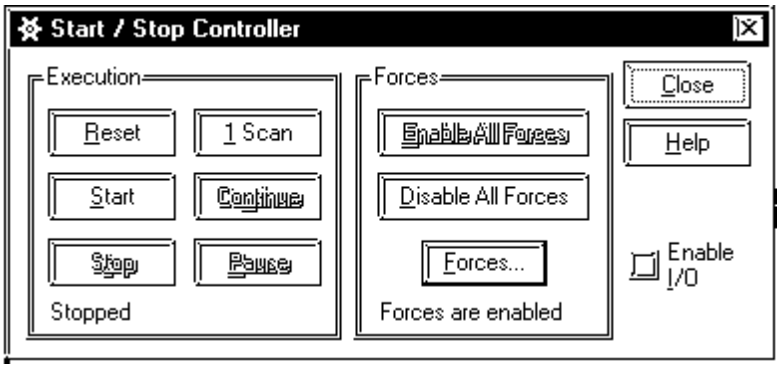
# 3.2 Starting and Stopping the Controller

Once you are online, you can start the Controller. It is at this point that your program starts solving logic. As mentioned previously, you must be online to the Controller before you can use the start/stop, or online editing functions.

## ■ To Start/Stop the Controller:

1. From the [**Controller**] menu, select [**Start/Stop**]. If you are in Programming Mode, however, this option is unavailable. The [**Start/Stop Controller**] window is displayed.

**Note: If you click on [Reset], all Editor variables will be reset except retentive variables. Use the MOV instruction etc. if any values need special initialization.**

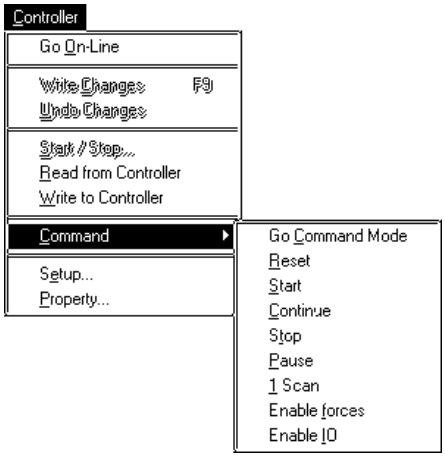



The functionality of the [**Start/Stop Controller**] window is explained below.

| Option | Description |
|---|---|
| **Start** | The [**Start**] button starts the Controller. Once it starts, it scans from the beginning of the program and executes all logic sequentially. The first scan executes any initialization logic. |
| **Stop** | The [**Stop**] button stops the Controller. |
| **Reset** | The Reset button causes the Controller to reload the ".PRW" file, initialize any I/O and then stop. |
| **1 Scan** | Press this button to perform a single scan of logic. This function is useful for troubleshooting or debugging an application. |
| **Pause** | Pause button stops the Controller from scanning logic but leaves the I/O enabled. |
| **Continue** | The Continue option is available after the Pause button has been pressed. It allows the Controller to continue executing logic (or a single scan) with the current data values. |
| **Enable All Forces** | Enables the forced variables. |
| **Disable All Forces** | Disables the forced variables. |
| **Forces** | Lists all forced variables in the ladder logic program. |
| **Enable I/O** | This function enables the inputs/outputs to the GLC main unit and external I/O of the I/O unit. In normal operations, the input/output of the external I/O is disabled when the GLC is set to the RUN mode after performing a Logic Program download. For safety reasons, this function prevents the possibility of accidental startups of machines caused by errors in operation and logic programs. |

**Note: When the setting is changed from Start/Stop, the system internally checks the status for the "Enable IO" setting. Therefore, "Enable IO" setting changes made during the "Start" mode will not be reflected. Be sure to change the setting to "Stop" before changing the "Enable IO" setting, and then return to the "Start" mode.**

You can also select these items from the [**Controller**] menu's [**Command**].




| | |
|---|---|
|  | Go Command Mode |
|  | Go Online |
|  | Write to controller |
|  | Read from controller |
|  | Write change (Enabled with the On-Line Edit only.) |
|  | Reset |
|  | Start |
|  | Continue |
|  | Stop |
|  | Pause |

| | |
|---|---|
|  | 1 Scan |
|  | Enable Forces |
|  | Enable IO |

## 3.3 Troubleshooting Using System Variables

System variables can be used to help troubleshoot for an application if it does not perform as expected.

The system variables are the most useful for detecting problems with either the Controller or the I/O are #Fault, #IOFault, #IOStatus and #ScanCount.

| | |
|---|---|
| **#FaultCode** | #FaultCode identifies the most recent fault condition. It is reset to 0 when the first scan operates after the Controller started. |
| **#Faultrung** | #Faultrung detects the rung number which has a fault. |
| **#IOFault** | #IOFault is a discrete variable that is turned ON when a fault is detected in your I/O system. |
| **#IOStatus** | #IOStatus is an array which displays I/O specific errors. These errors are indexed with a numeric code. This code differs from driver to driver. (Reference: For a detailed explanation of the error see the driver's Help system.) An error is displayed in #IOStatus only if #IOFault has been turned ON. |
| **#ScanCount** | #ScanCount indicates the number of scans the Controller has executed since it was last started. When monitored, this variable should constantly be increasing. If it is not, the Controller is not running. |

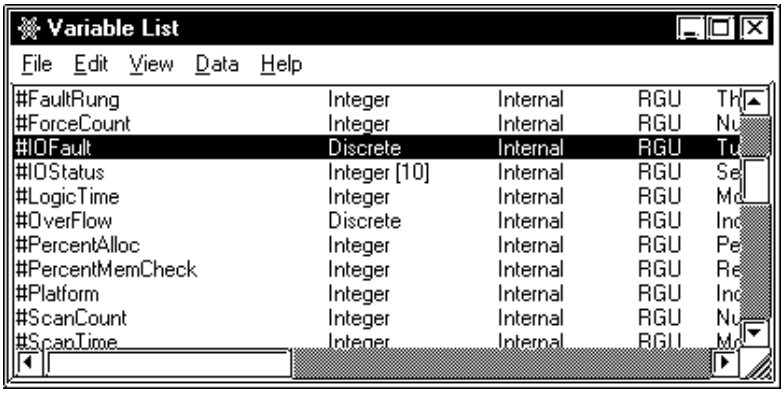
Reference *For details about System's variables, refer to* ***Pro-Control Editor User Manual, "chapter 3 System Variables"***

# 3.4 Viewing System Variables

You can view the system variables to show information about I/O status, scan time and controller status.

## ■ To View System Variables:

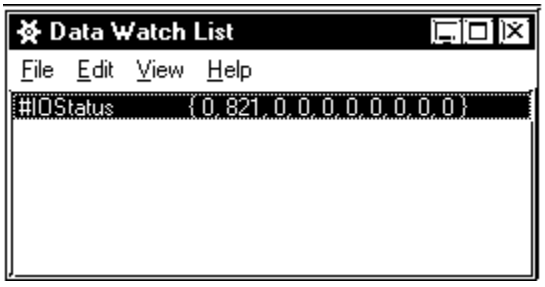
1. From the [**Data**] menu, select [**Variable List**] and the [**Variable List**] window appears. All Editor system variables (variables which begin with #) should be displayed. If they are not, select [**System**] from the [**View**] menu.




2. From the [**Data**] menu, select [**Data Watch List**]. The [**Data Watch List**] window appears.
3. Click and drag the system variables you wish to monitor from the [**Variable List**] window to the [**Data Watch List**] window.

These monitored variables display the appropriate errors if they occur while the logic is being scanned.

In the following example, I/O error 821 has occurred with driver one. The #IOFault is turned ON.



# 3.5 Reading from the Controller

To edit and save a logic program located in the GLC unit, read out the program from the Controller.

To read a logic program from the GLC, you can either:

- receive the screen data and logic program via the "Transfer" window of the GP-PRO/PB III.
- receive the logic program exclusively via the Editor.

This section describes how to receive a logic program exclusively using the Editor. For details on how to receive a logic program via the "Transfer" window in the GP-PRO/PB III, refer to Chapter 7 Data Transfer in the GP-PRO/PB III Operation Manual.

## ■ To Read from the Controller:

1. If the Controller is online, from the [**Controller**] menu, select [**Go Off-Line**].

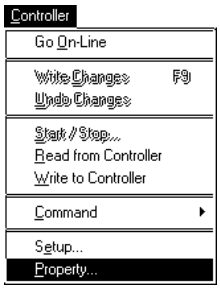

***The Controller must be stopped before doing a "read from controller" if the program contains values that are not initialized.***

2. From the [**Controller**] menu, select [**Read from Controller**]. A copy of the program written to the Controller will be opened.

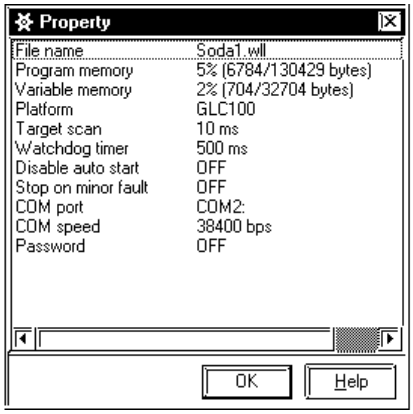
   You can now make changes to the program and /or save it as a ".PRW file".

# 3.6 Property

Select the [**Controller**] menu's [**Property**]. The GLC program's property information list box will appear.




The [**Property**] box is shown below.



*MEMO*

# 4 On-Line Editing

The Editor allows you to make online changes to a program running in the Controller and have these changes take effect immediately. For the demonstrations and examples in this chapter use the 'Soda.prw' file, located in 'C:\Program Files\Pro-face\ProPBWin\GLC_Samples'. All examples used here assume that the ladder colors and preferences use the system default.

## 4.1 Before Editing

This chapter uses the Soda.prw file as an example.

### ■ To execute the example program:

1. Open 'Soda.prw' file. It is included as an Editor sample program and is located in 'C:\Program Files\Pro-face\ProPBWin\GLC_Samples'.
2. Write this program to the Controller.
3. Go online to the Controller.
4. Start the Controller.

Reference *For Controller operation, refer to "**Chapter 3 Running the Ladder Logic Program**".*

### ■ Program changes which can be made online to a GLC

On-line editing features are restricted for the GLC platform, however, the following changes can be made to a program while it is running online in the Controller.

• Turning ON/OFF discrete variables

• Integer value changes

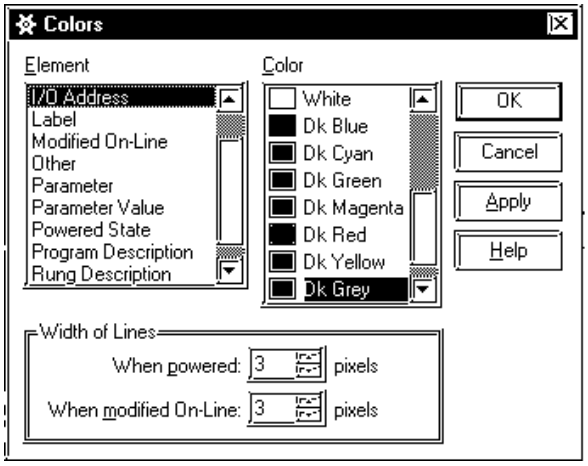
## 4.2 Using Colors for On-line Editing

The Editor uses default colors to indicate specific aspects and changes to a ladder logic program while running online. The default colors are:

| Colors | Messages |
|---|---|
| Green | Circuit is on. |
| Red | Error is occurred at Rung |
| Purple | During On-line Edit |

### ■ To Change the Color Defaults:

1. From the [**View**] menu, select [**Colors**] and the [**Colors**] dialog box appears.



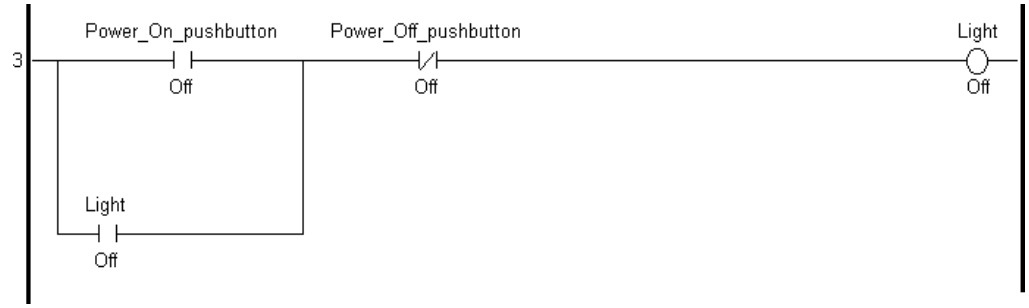

2. Select the [**Element**] and then the [**Color**] you want associated with that element, then click on [**Apply**].

## 4.3 Turning Discretes ON and OFF

Discrete variables can be manually turned ON or OFF while the logic is running. A discrete that has been turned ON is not the same as a discrete that has been forced ON, since its state can be affected by the program as it is scanned.

### ■ To Turn a Discrete ON or OFF:

1. Right-click on the variable 'Light' assigned to the output coil on rung 2.
2. Select [**Turn ON**] from the short cut menu. The 'Light' variable turns ON and the power flow indicates that power is flowing through the rung.




3. Right-click on the variable 'Light' assigned to the output coil on rung 2.
4. Select [**Turn OFF**] from the short cut menu. The 'Light' variable now turns OFF and the power flow disappears, indicating that power no longer flows through the rung.

**Note:** **Power flow is not displayed in your logic if the [Power Flow] check box is not selected in the [Monitoring] section of the [Preferences] dialog box.**

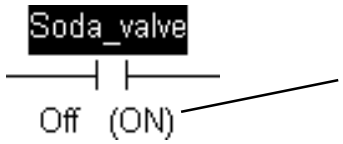
Reference *2.1.1 Preference Area Settings (Prior to Creating a Logic Program)*

# 4.4 Forcing Discretes ON and OFF

Discretes can be forced ON or OFF while you are online in the Controller. The difference between turning a discrete ON or OFF and forcing it ON or OFF is that if you force it, the variable does not change its state until the force is manually changed. The program logic and I/O cannot change its state.

## ■ To Force a Discrete ON or OFF:

1. Right-click on the variable 'Soda_valve' on the output coil on rung 9.
2. Select [**Force ON**] from the short cut menu.
3. Click on [**OK**] in the [**Force**] dialog box.




The variable turns ON and cannot be turned OFF by the logic program.

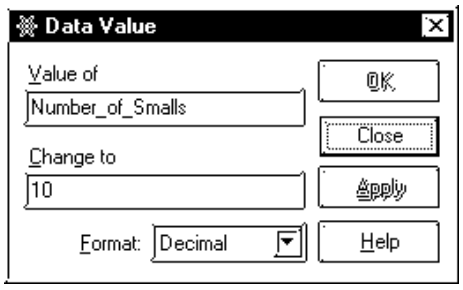
**Note:** **If you find that forced variables have no effect in your ladder logic program, they have probably been disabled in the Editor. To enable forces click on the [Enable All Forces] button in the [Start/Stop Controller] dialog box, or use the [Controller] menu and the toolbar.**

# 4.5 Changing Variable Values

While you are online to the Controller you can set the value of any Editor variable included in your ladder logic program.

## ■ Changing a Variable Value:

1. From the [**Data**] menu, select [**Value**]. The [**Data Value**] dialog box appears.
2. Click on the variable 'Number_of _Smalls' in the ladder logic. The [**Data Value**] dialog box appears as follows:




3. Select the '0' in the [**Change to**] field, then type '5'.
4. Click on [**Apply**].

The value of 'Number_of _Smalls' is now 5. You can change other values or close the [**Data Value**] dialog box by clicking on [**Close**].

**Note:**

- **You can enter data values in Decimal, Hexadecimal, Octal or Binary number format. Simply select one from the [Format] list.**
- **Use the [Variable List] or [Data Watch List] in conjunction with the [Data Value] dialog box to quickly find and set Editor variables.**

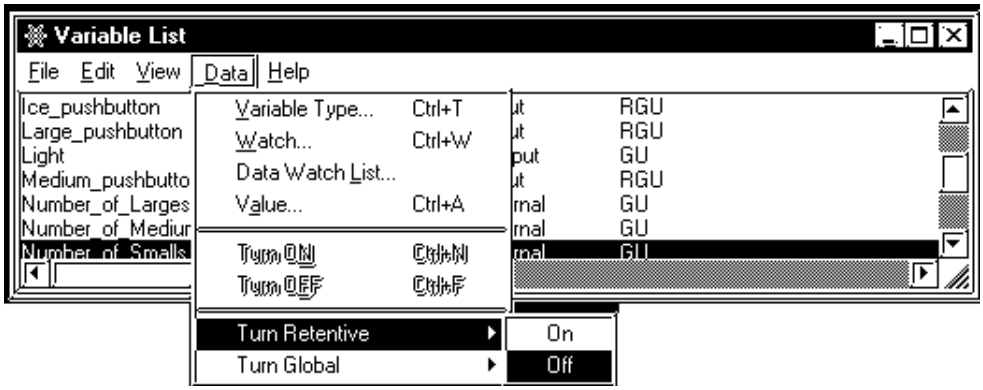
# 4.6 Changing Variable Attributes

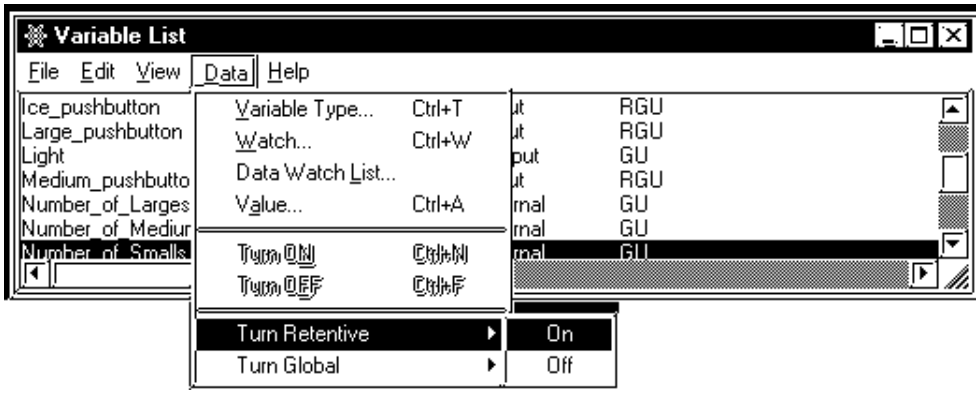
You can use the [**Data**] menu to change the variable attributes (Retentive/Global.)

This menu is enabled only when you are in the programming mode.

## ■ Changing a Variable Attribute (Retentive)

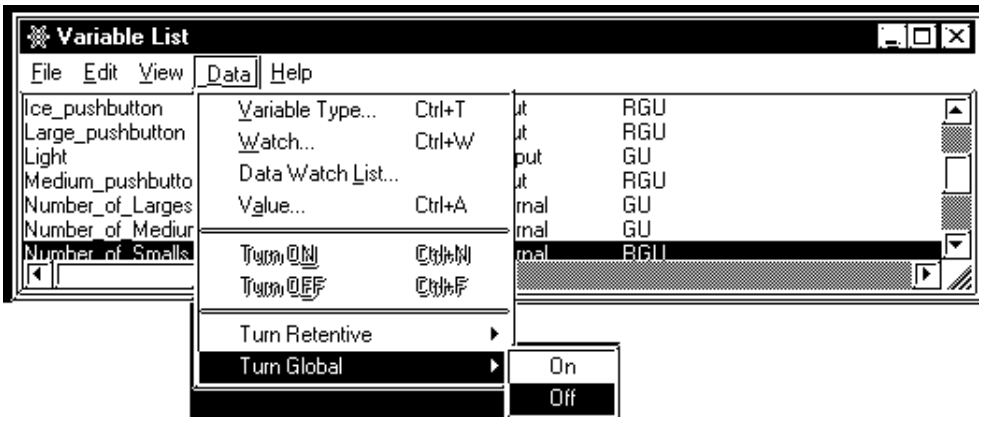
Select the [**Data**] menu's [**Variable List**]. The [**Variable List**] window will appear. Select the variable you wish to change its attribute, and change the attribute using this window as shown below. However, the system variable's "retentive" cannot be changed.

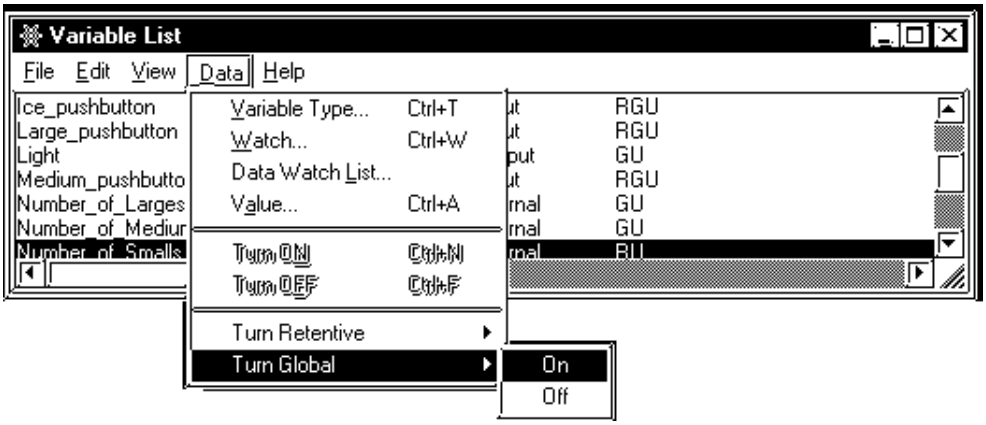


## ■ Changing a Variable Attribute (Global)

Select the [**Data**] menu's [**Variable List**]. The [**Variable List**] window will appear. Select the variable you wish to change its attribute, and change the attribute using this window as shown below.

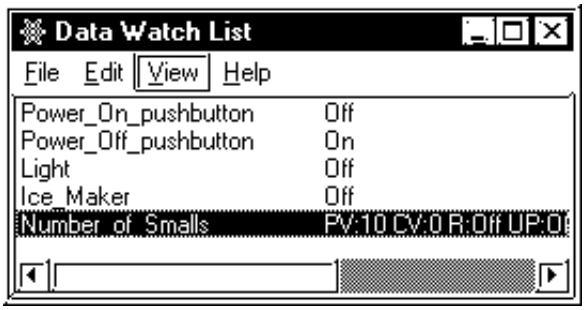


# 4.7 Data Watch List

## ■ To change the display mode of all selected variables at the same time

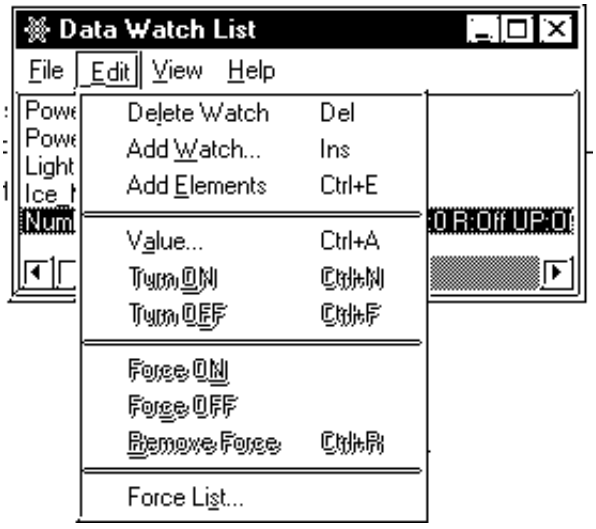
Select the [**Data**] menu's [**Data Watch List**], and select a display mode in the [**View**] list box. This allows you to change all of the selected variables' display mode to the designated display mode at the same time.




## ■ To Display Array Elements

When creating an array via [**Data Watch List**], you can display array counter/timer's values by element.

1. Select the [**Data**] menu's [**Variable List**] and [**Data Watch List**].
2. Select the [**Data Watch List**] menu's [**Edit**], and select [**Add Elements**] from the [**Edit**] box.



# 4.8 Online Edit (GLC model: GLC2000 Series)

The GLC2000 Series allows you, while in monitoring mode, to change the logic program as it is being executed.

In online edit, 6 types of editing functions are available.

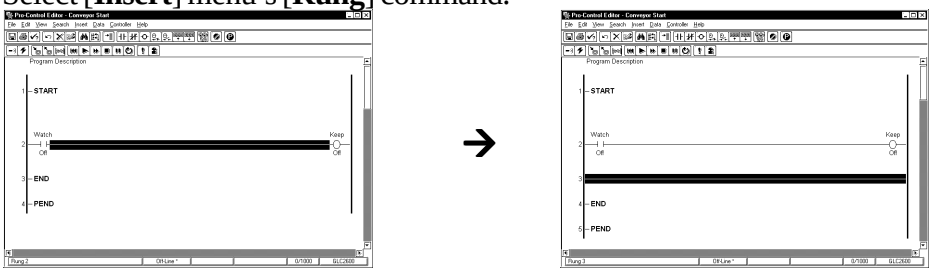
1. Add Rungs
2. Replace Rungs
3. Delete Rungs
4. Add Labels
5. Add Subroutines
6. Add Variables

## 4.8.1 Editing Functions in Online Edit

### ■ Add Rungs

This adds a single line ladder circuit between designated rungs.

Select [**Insert**] menu's [**Rung**] command.

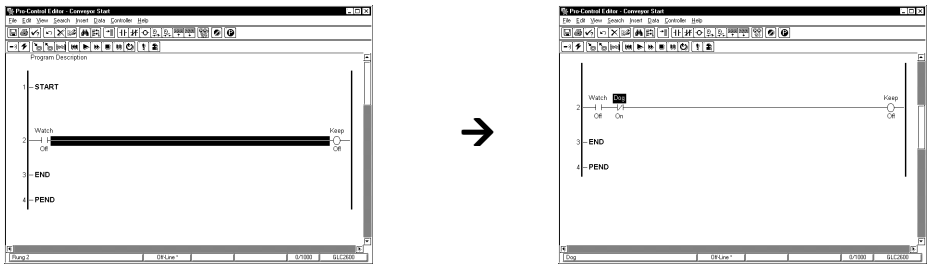



If a variable is added at this time, the variable add instruction is executed at the same time.

### ■ Replace Rungs

This edits the ladder circuit of an existing line.

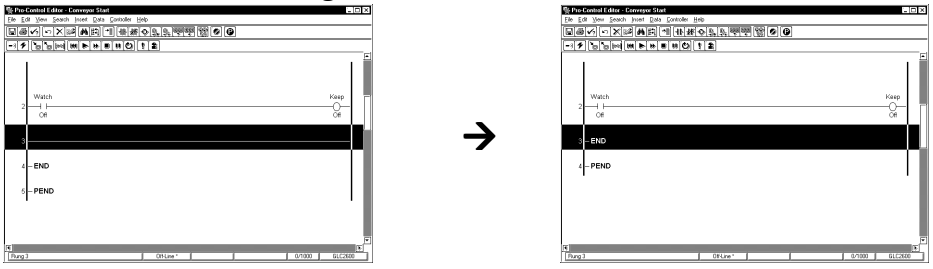
Instructions can be inserted, replaced or deleted.




If a variable is added at this time, the variable add instruction is executed at the same time.

### ■ Delete Rungs

This deletes a selected rung.

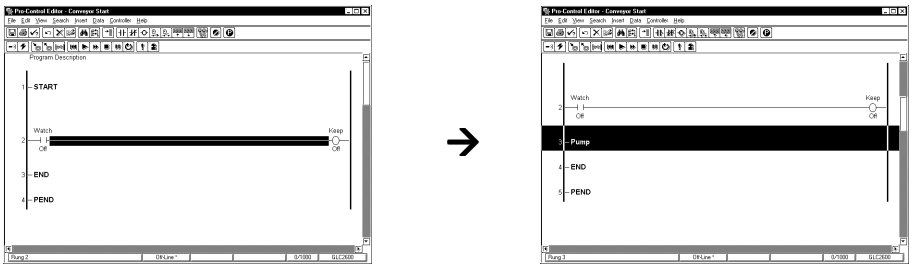



Variables are not deleted at this time.

## ■ Add Labels

This adds a label.

Select [**Insert**] menu's [**Label**] command.

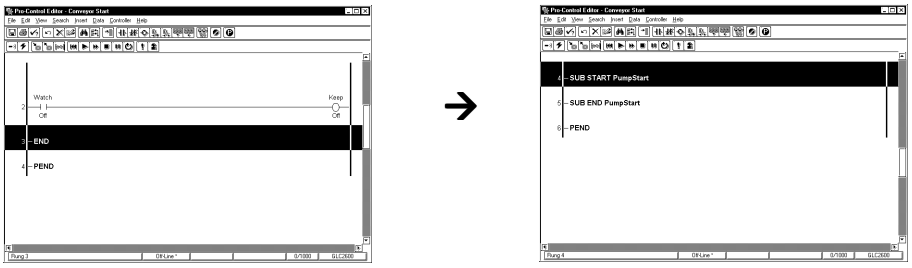


## ■ Add Subroutines

This adds a subroutine.

Subroutine is inserted between END label and PEND label.

Select [**Insert**] menu's [**Subroutine**] command.



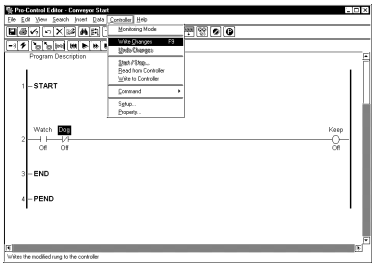
## ■ Add Variables

This adds a new variable.

Addition can be made during inserting [**Variable type**] of [**Data**] or command.

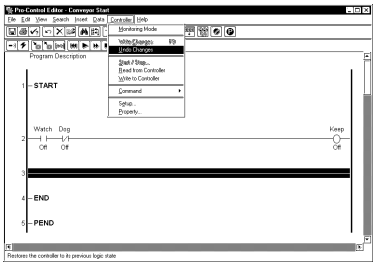
## ■ Writing an Edited Logic Program

This writes the logic program edited by [**Write change**] of [**Controller**] to the GLC unit. Logic program will be written when editing other rung after editing.



## ■ Restoring an Edited Logic Program

This restores the edited program (rung unit) to its previous state.



## 4.8.2 Saving Data

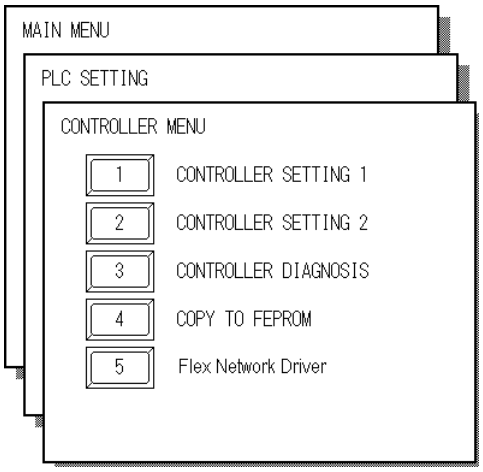
After creating a logic program with the Editor, write it once to the FEPROM using the [Write to Controller] command. After sending the logic program to the GLC and starting it up, the content of FEPROM is copied to the GLC's SRAM. With online edit, this logic program in SRAM is edited. The logic program saved in SRAM may be lost due to a dead battery when the power supply is OFF. In this case, the logic program stored in the FEPROM is read at the next start-up. Therefore, be sure to backup the edited logic program with the [**Copy to FEPROM**] command in the GLC OFFLINE menu or save as an PRW file using Editor.

### ■ Copy to FEPROM

Select [**Copy to FEPROM**] from the GLC OFFLINE menu.

When entering the OFFLINE menu, the GLC stops the logic program and display functions and starts from the initial conditions.

When an edited logic program is copied to FEPROM, the system can continue operation by reading the logic program from FEPROM, even if the logic program saved in SRAM is lost.





**If [Copy to FEPROM] is not performed after editing the logic program with online edit, the warning message "No Backup logic program in FEPROM" will be displayed at GLC start-up. If copying is not done to FEPROM and the logic program saved in SRAM is lost, the system will execute the logic program saved in FEPROM prior to editing with online edit. So, be sure to copy to FEPROM.**

---

**1 A Lithium battery's lifetime is:*

*10 years when the battery's ambient temperature is under 40ºC*

*4.1 years when the battery's ambient temperature is under 50ºC*

*1.5 years when the battery's ambient temperature is under 60ºC*

*When used for backup:*

*Approximately 60 days, with a fully charged battery.*

*Approximately 6 days, with a half-charged battery.*


**When data in SRAM is lost, the logic program is read from FEPROM automatically. However, a minor error will occur in this case, and with some systems there may be a problem in automatic execution using the logic program in FEPROM. In such systems, select the [Continue Error Switch], and set so the logic program is not automatically executed.**

### ■ Saving with Editor

After finishing online editing with the Editor, the file can be saved as a PRW file by switching to programming mode, and performing [**Save**] for the edited logic program. The edited logic program can be executed by downloading the logic program saved in SRAM to the GLC.

# 5 Using the Editor and GP-PRO/PBIII

GP-PRO/PBIII allows you to create operation screens that are linked with the parameters created by Editor. These screens allow you to operate or monitor the ladder logic program and the controller.

This chapter focuses on how to create operation screens for the GLC unit using the GP-PRO/PBIII software. Here, a short tutorial is given that uses a ladder program to pump water from a tank.

## 5.1 Importing the I/O Symbols to GP-PRO/PBIII

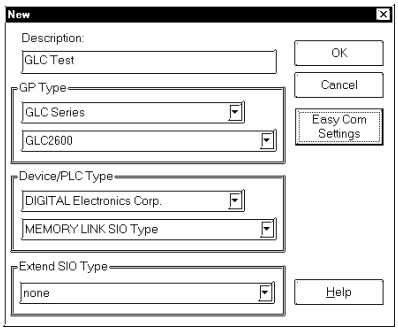
The followings steps explain how to import GLC parameters into GP-PRO/PB III.

### 5.1.1 To Start up the Editor

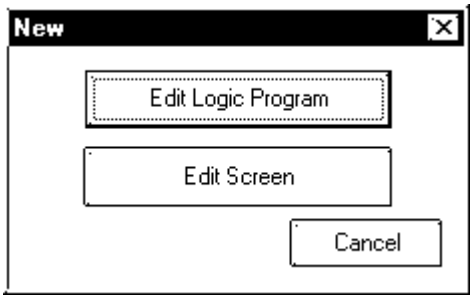
1. Click the [**Start**] button, go to [**Programs**]-[**Pro-face**]-[**ProPB3 C-Package**], and then click [**Project Manager**].
2. The Project Manager starts up.


#### ■ To Create a New Project

1. In the Project Manager screen, select **[New]** from the **[Project]** menu, or click the **[New]** button.
2. Designate the settings for Description, Display (GP) Type, PLC Type, and Extended SIO Type, and then press **[OK]** to execute the command.



3. A window appears asking whether you will create a Logic Program or a Screen.




Logic Program : Starts up the Editor.

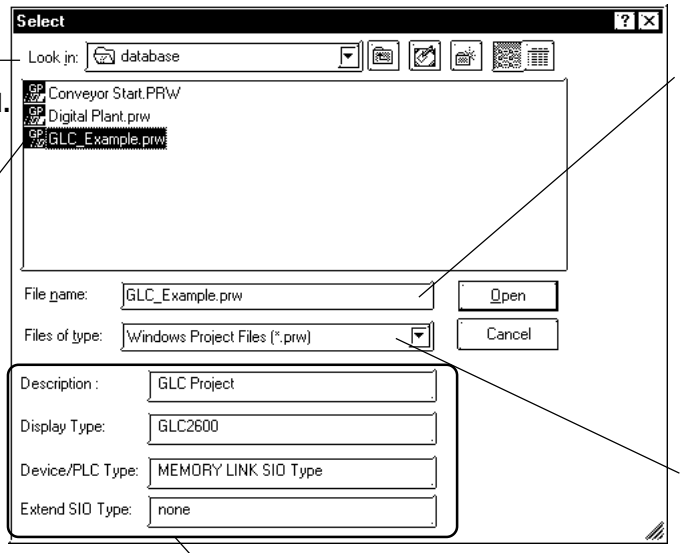
Screen : Starts up the Screen Editor

Cancel : Returns to the Project Manager

**Note: When you attempt to create or select a different Project File after creating a new Project File, a confirmation window appears asking if you want to save the file. Clicking [Yes] will display the "Save As" dialog box on the screen. Clicking [No] will close the screen without saving the file.**

## ■ Select from Existing Projects

Select the **[Select]** command from the **[Project]** menu on the Project Manager's window, or click the **[Select]** icon.




**Select the folder where the desired Project File is located.**

**This area displays the currently selected folder and names of the existing Projects in a list.**

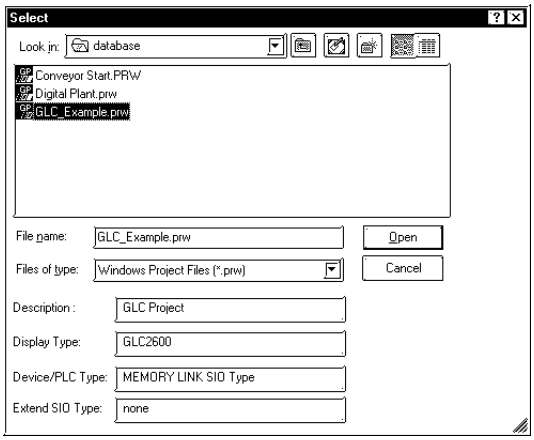
**This field displays the name of the Project File selected from the list. You are also allowed to specify the file by entering the desired file name.**

**Select the type of file.**

**This field displays the Description documented on the selected Project File, the current settings for the Display (GP) Type, PLC Type, and Extended SIO Type.**

**Important Select "MEMORY LINK SIO TYPE" when no external device is (PLCs, temperature controllers, and inverters, etc.) is connected.**

1. Select the **[Select]** command from the **[Project]** menu on the Project Manager's window, or click the **[Select]** icon.
2. Select the desired Project File from the list, or enter the name of the desired Project File.




3. Click **[Open]** to execute the command.

## ■ Startup Icons of a Logic Program

Create : Allows you to create a logic program with the Editor.

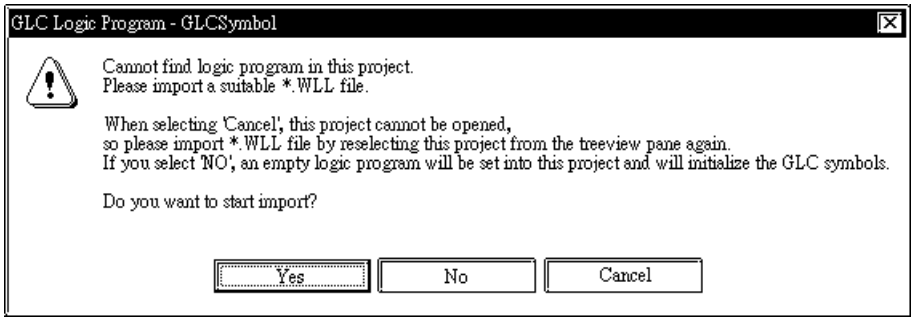
Monitor : Allows you to monitor a logic program.

I/O : Designate the I/O configuration.

Variable : Displays the Variable List

**Note: The icons of a logic program are enabled only when the GLC series supported Display (GP) Type is selected.**

To select a Project File created with an earlier program version, you are required to assign the WLL file when a Project that has imported a WLL file is selected. In this case, the following dialog box appears on the screen.



## 5.1.2 Pasting Instruction Data

First, save the logic program you created earlier in order to assign variables to the addresses of parts and tags used for screen creation. The variables are imported to the GP-PRO/PB III by saving the program.

You can place parts corresponding to the instructions by copying the desired instructions in the logic program created with the Editor and pasting them to the Screen Editor.

You can also insert the instructions corresponding to the parts by copying the parts placed on the screen and pasting them to the logic program.

### ■ Conversion between Instructions and Parts

Each instruction corresponds to specific parts.

#### ◆ Conversion from Instructions to Parts

The table below shows the types of parts to which instructions are converted.

| Pro-Control Editor Instruction → | GP-PRO/PBIII for Windows Parts |
|---|---|
| NO (a Contact) | Bit Switch |
| NC (b Contact) | Bit Switch |
| PT (Start Up Contact) | Bit Switch |
| NT (Start Down Contact) | Bit Switch |
| OUT/M (Out Coil) | Light |
| NEG/NM (Reverse Coil) | Light |
| SET/SM (Set Coil) | Light |
| RST/RM (Reset Coil) | Light |
| CTU (Up Counter) | Numeric Display/Graph/Keypad Input Display |
| CTD (Down Counter) | Numeric Display/Graph/Keypad Input Display |
| CTUD (Updown Counter) | Numeric Display/Graph/Keypad Input Display |
| TON (On Delay Timer) | Keypad Input Display |
| TOF (Off Delay Timer) | Keypad Input Display |
| TP (Pulse Timer) | Keypad Input Display |

#### ◆ Conversion from Parts to Instructions

The table below shows the types of instructions to which parts are converted.

| GP-PRO/PBIII for Windows Parts → | Pro-Control Editor Instruction |
|---|---|
| Bit/Toggle Switch | NO (a Contact), NC (b Contact), PT (Start Up Contact), NT (Start Down Contact) |
| Lamp | NO (a Contact), NC (b Contact), PT (Start Up Contact), NT (Start Down Contact), OUT/M (Out Coil), NEG/NM (Reverse Coil), SET/SM (Set Coil), RST/RM (Reset Coil) |
| Numeric Display/Graph/Keypad Input Display | CTU (Up Counter), CTD (Down Counter), CTUD (Updown Counter) |
| Keypad Input Display | TON (On Delay Timer), TOF (Off Delay Timer), TP (Pulse Timer) |

## ■ Pasting Logic Program Instructions to a Screen

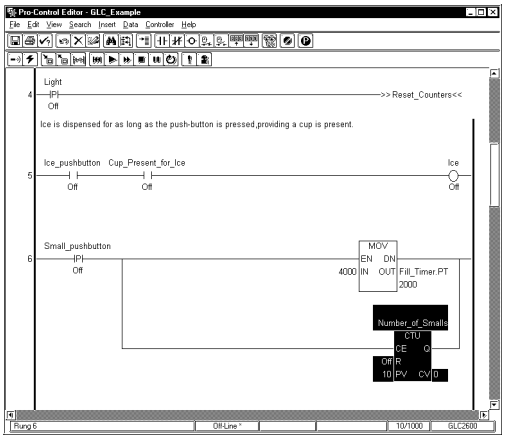
Copy a logic program instruction and paste it to a screen. When pasting an instruction, select the type of parts to which the instruction is converted.

Important

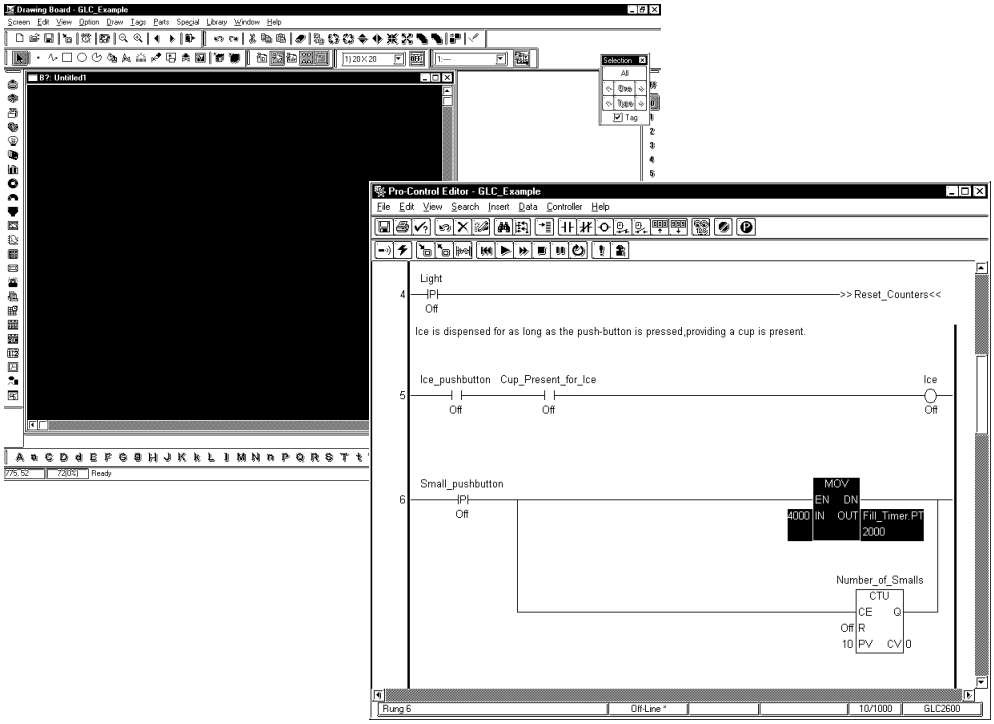
- ***Prior to copying an instruction, you are required to assign a variable to the instruction. An instruction to which a variable is not assigned cannot be pasted on a screen.***
- ***Make sure to save the logic program before pasting an instruction.***

Note: **When the instruction is modified via the logic program after pasting the instruction on the screen, the change will not be reflected to the pasted instruction.**

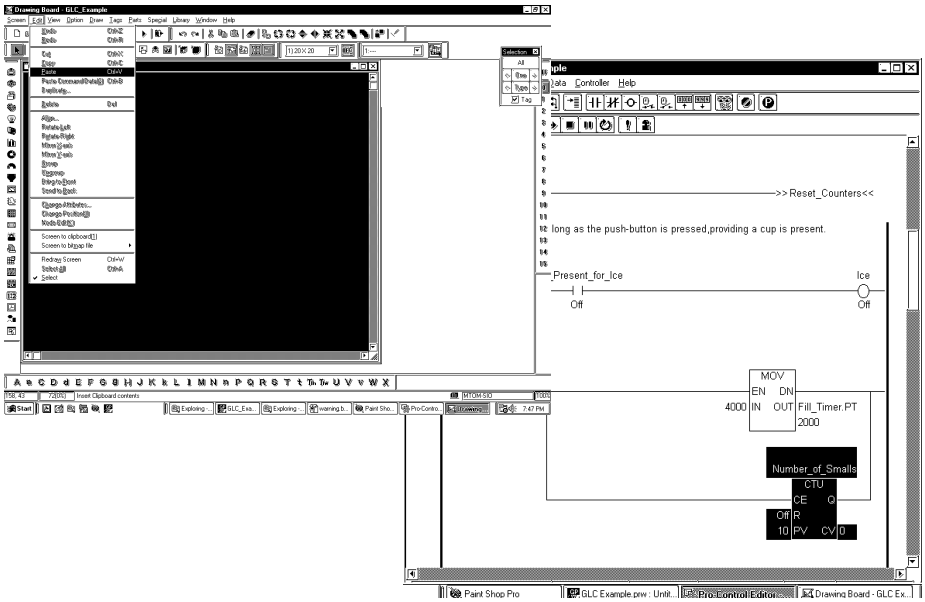
1. Select the desired instruction on the Editor.



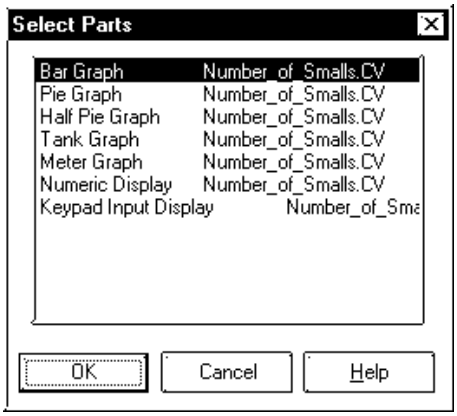

2. On the Editor's screen, select the **[Copy]** command from the **[Edit]** menu. The selected instruction is copied to the Clipboard.



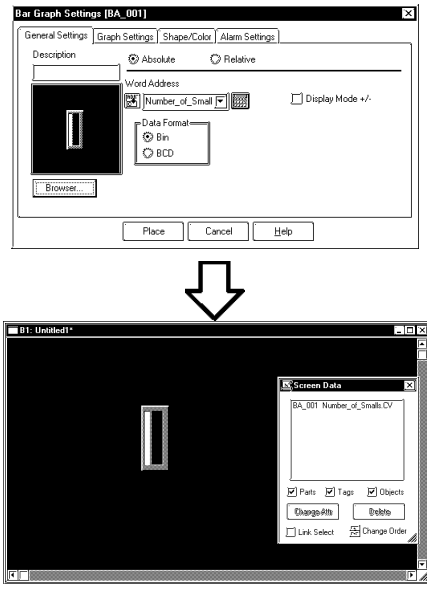
3. On the Screen Editor's screen, select the **[Paste]** command from the **[Edit]** menu.




4. Select the Part to which the instruction is converted, and click **[OK]** to confirm the selection. The Parts selection menu shows the parts corresponding to the copied instruction in the list. When the available part for the instruction is fixed, the menu will not appear.




5. On the Screen Editor's screen, select the **[Paste]** command from the **[Edit]** menu.



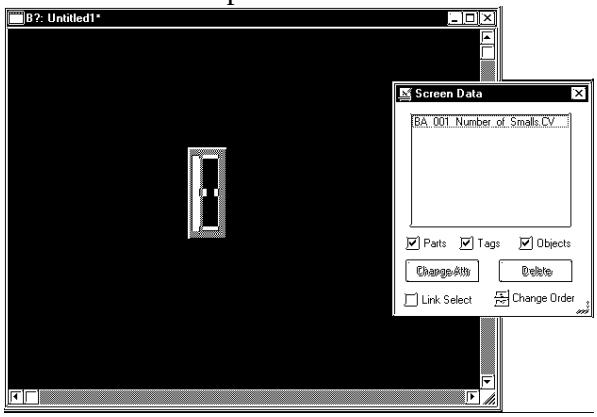
## ■ Pasting Parts Placed on a Screen to a Logic Program

Copy a part placed on a screen, and paste it to a logic program. When pasting a part, select the type of instruction to which the part is converted.

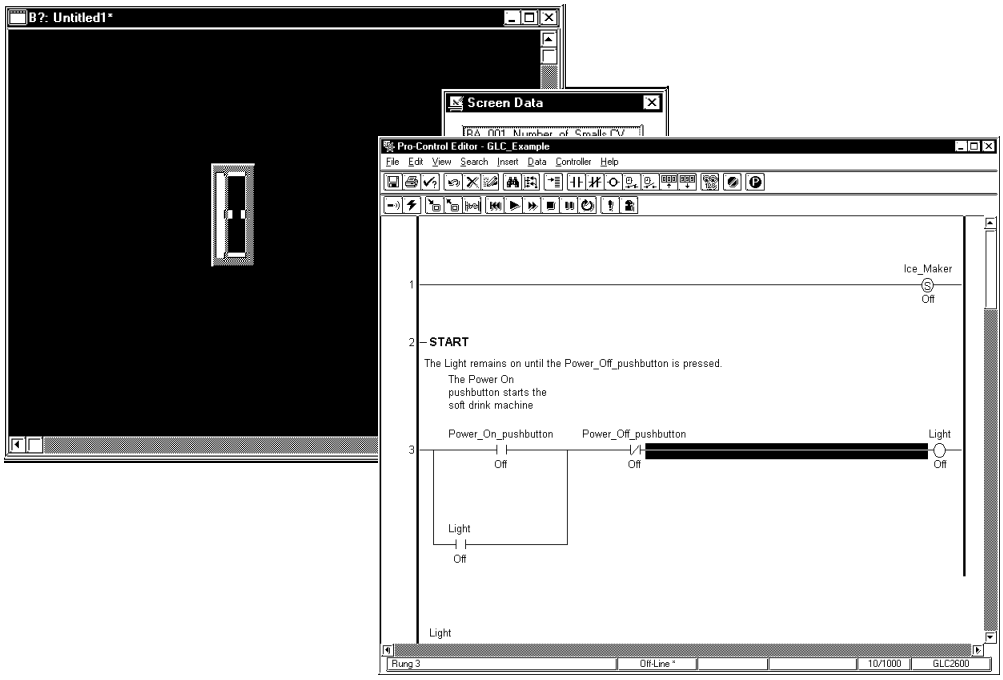
***Important*** ***Prior to copying a part, you are required to assign a variable (GLC symbol) to the part. A part to which a variable is not assigned cannot be pasted on a logic program.***

**Note:**
- **Variables are registered in the Symbol Editor as GLC symbols by saving the logic program.**
- **When the instruction is modified via screen after pasting the instruction on the logic program screen, the change will not be reflected to the pasted instruction.**

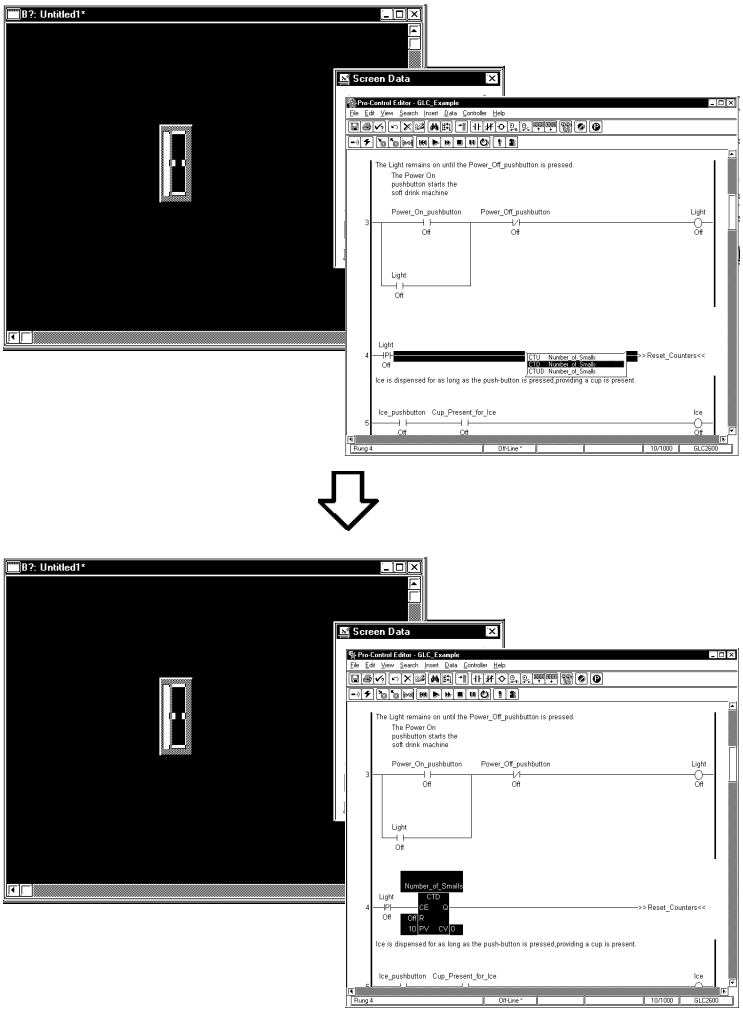
1. Select the desired part on the Screen Editor.




2. On the Screen Editor's screen, select the [Copy] command from the [Edit] menu. The selected part is copied to the Clipboard.
3. On the Logic Program Editor, select the rung you want to insert the instruction on, and then select the [Paste] command from the [Edit] menu.



4. Select the instruction to which the part is converted, and double click on it to confirm the selection. A list of instructions to which the part is converted appears on the screen corresponding to the copied part.




## ■ Drag and Drop Operation

You can copy and paste instruction data and parts using drag and drop.

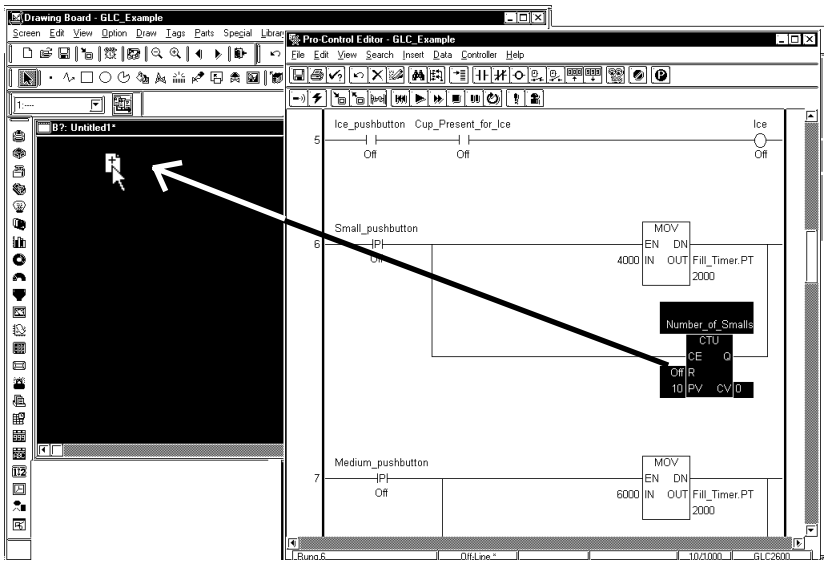
***Before using the Drag and Drop feature, variables must be assigned to instructions and/or Parts. A part or instruction to which a variable is not assigned cannot be processed using drag and drop.***

### ◆ Drag and Drop from a Logic Program Instruction to a Part

You can place parts corresponding to the instructions by dragging the desired logic program instructions created with the Editor to the Screen Editor.

- ***Make sure to save the logic program before dragging and dropping.***

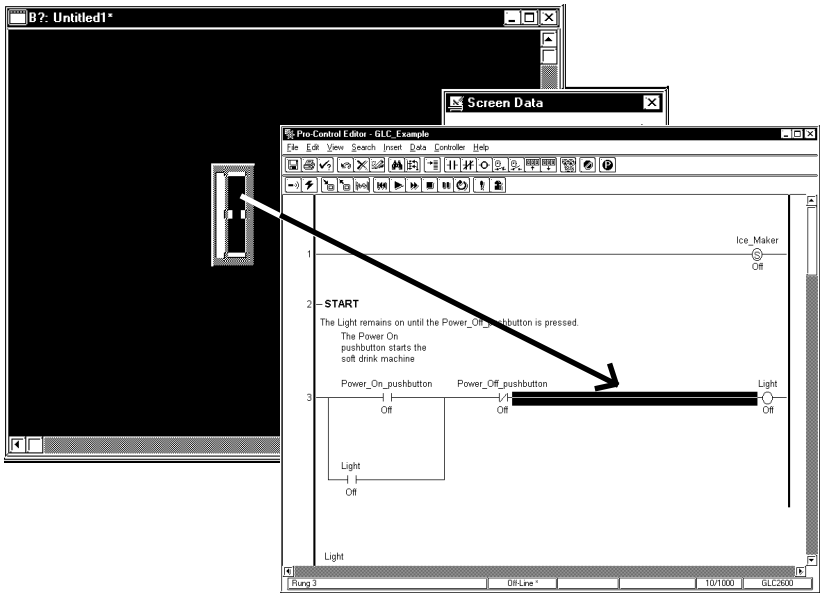
**When the instruction is modified via the logic program after pasting the instruction on the screen, the change will not be reflected to the pasted instruction.**




## ◆ Drag and Drop from a Part to an Instruction

You can insert the instructions corresponding to the parts by dragging the parts placed on the screen to the logic program. Drag the desired part while pressing down the **[Ctrl]** key.

**When the instruction is modified via screen after pasting the instruction on the logic program screen, the change will not be reflected to the pasted instruction.**

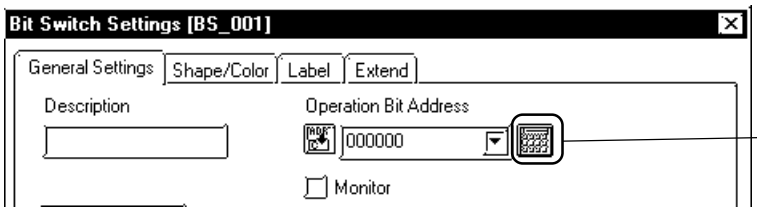


## ■ Input from the Address Keypad

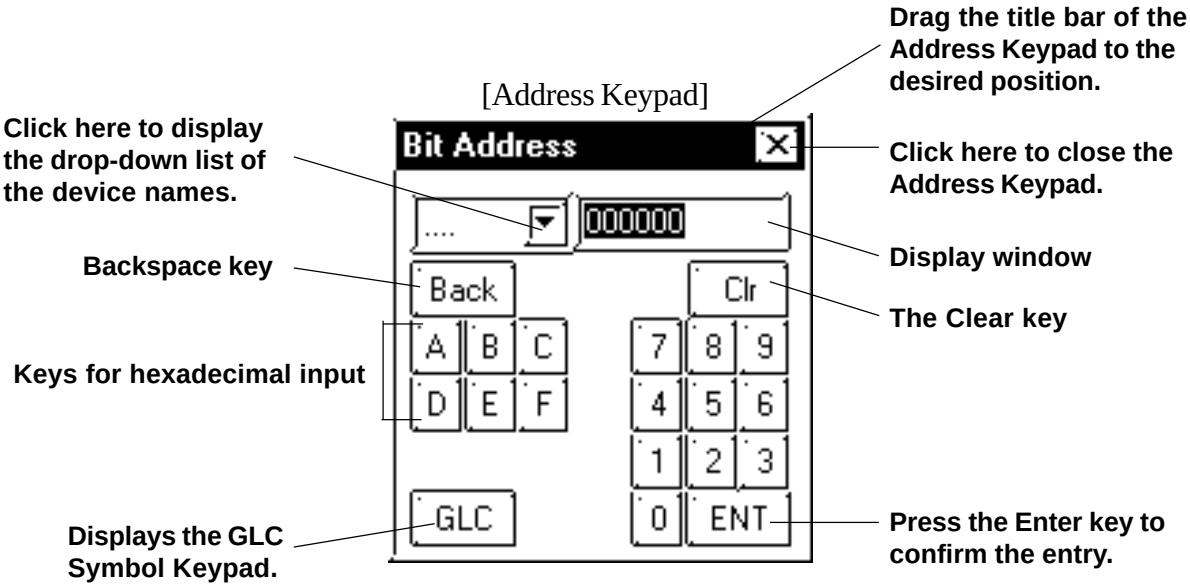
The Editor allows you to designate a bit by adding an extension (.X [m]) to an integer variable. Also you can designate a bit on the GP-PRO/PB III by adding an extension (.X [m]) to an imported integer variable.

Reference *2.3 Accessing Variables in the Pro-Control Editor User's Manual*

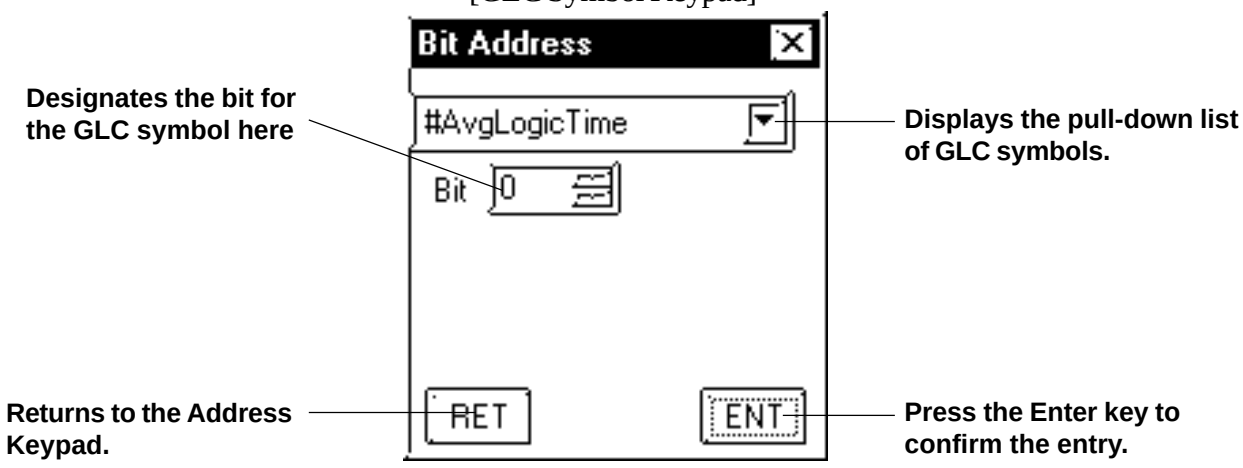
To designate the bit for a variable (GLC symbol) of a logic program, display the GLC Symbol Keypad by clicking the [Logic] button on the Address Keypad.




[Address Keypad]




[GLC Symbol Keypad]




**Note: Regardless of whether you designate a bit for an integer variable on the Editor or not, only the normal integer variables will be imported to the GP-PRO/PB III. If you wish to access to an integer variable in bit unit in the GP-PRO/PB III, designate a bit via the GP-PRO/PB III.**

## ■ Variable Restrictions

The following are restrictions on GLC variables used in the GP-PRO/PB III.

- When exporting normal symbols, the GLC symbols will not be output.
- When copying and pasting normal symbols, the variables located in the Controller variables cannot be designated.
- When entering normal symbols, variables located in the GLC symbols cannot be designated.
- If the Display (GP) Type is changed from a GLC to a non-GLC type, the GLC symbols will be changed to normal variables and the automatically allocated addresses will be cancelled when GLC symbols are designated in the original GLC type. In this case, the screen containing the GLC symbols settings is automatically changed to the status requiring preparations for transfer. Review the GLC symbol allocation.
- When performing a simulation of a screen containing GLC symbols, the device information field on the Simulation screen will not display the devices designated with GLC symbols.
- The GLC series does not support a device type for the Editor variables, which means that the device type and address used for indirect designation of GP-PRO/PB III E-tags and K-tags cannot be specified.
- The GLC variables are handled in the Low-High order of a 32-bit device.
- The GLC arrays shown with "[ ]" are shown with "< >" on the GP-PR/PB III.
- The number of GLC variables that can be used with the GP-PRO/PB III software is limited to 2048. One element in an array is counted as one variable. If the number of global variables exceeds 2048, register the variables you will not use with tags and parts of the GP-PR/PB III as non-global variables.

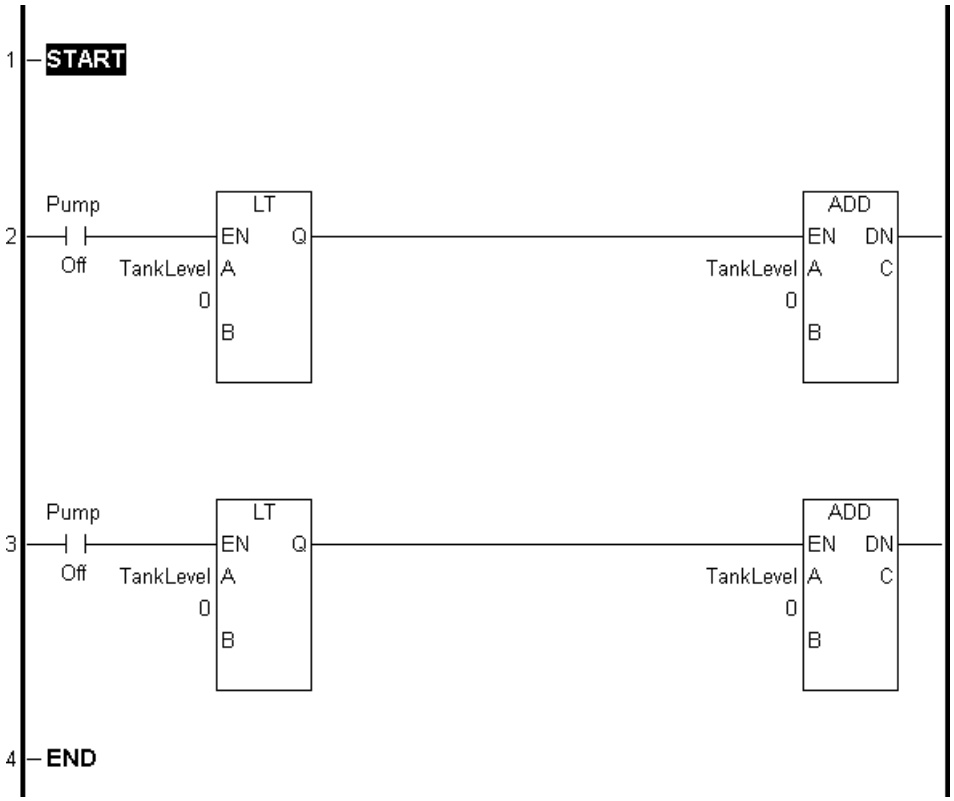
## 5.1.3 Screen Creation Example using the "Pump Tutorial"

The Pump Tutorial shows you how use the GP-PRO/PB III to create a screen linked with a logic program that is designed to draw up water from a tank using a pump.

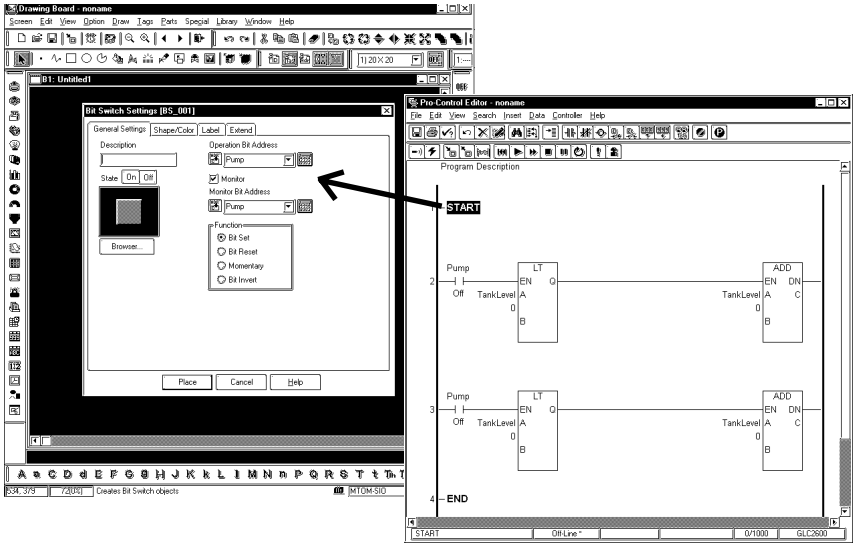
### ■ To Start up the Editor

1. Click **[Logic Program/Create]** on the Project Manager's window and then click the **[Draw/Screen]** command to open the Editor window and Screen Editor window.

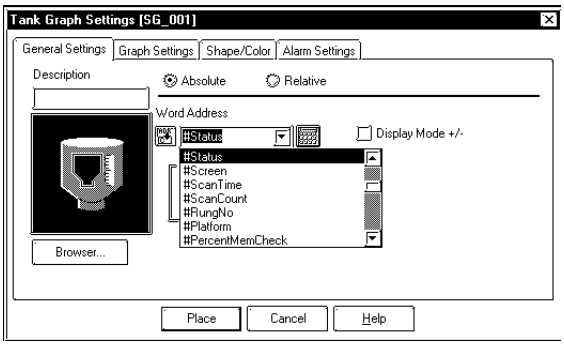

2. Create a logic program as shown below using the Editor. Refer to the completed sample of the screen in the "pump.prw" file.

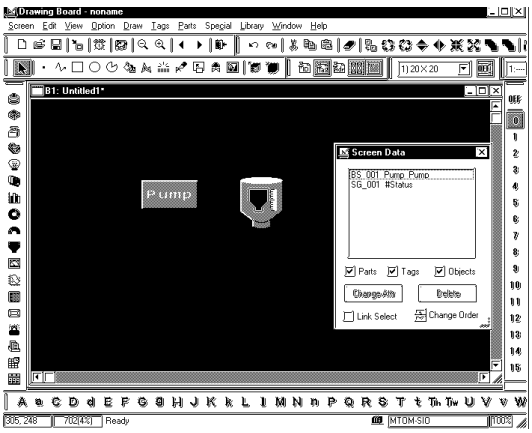


3. Save the logic program so that the Editor variables are imported. The variables are enabled on the Screen Editor.
4. Select the "Pump" instruction in the Editor, and drag it to the Screen Editor of the GP-PRO/PB III. Drop the instruction on the Screen Editor. The Part Selection dialog box will appear on the screen. Select "Bit Switch", and click **[OK]**.




5. Check that the **"Operation Bit Address"** is set to **"Pump"**, and click the **[Place]** button. The **"Pump"** Bit Switch is created on the Screen Editor.
6. Create a tank graph with the Screen Editor. Click on the **[Tank Graph]** command from the Tool Bar, or click **[Parts/Tank Graph]** from the Menu Bar.
7. Select "#Status" from the pull-down menu, and click the **[Place]** button.

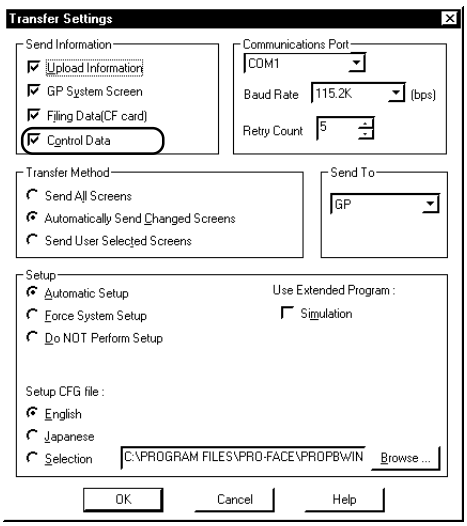


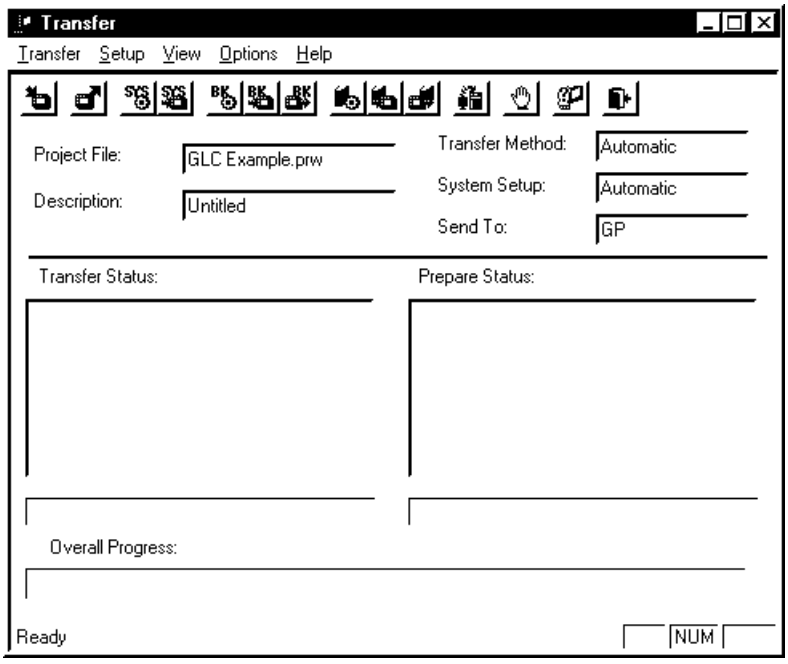



# 5.2 Transferring Screens to the GLC

## ■ Transferring GP-PRO/PBIII operation screens to the GLC

1. Click on the GP-PRO/PBIII screen's [**Transfer**] icon.
2. Select the **[Transfer Settings]** command from the **[Setup]** menu, and check that a check mark is put to the **"Control Data"** option in the **"Send Information"** area on the Transfer Setting dialog box, and then transfer the data via the **"Transfer"** window. If an error occurs during data transfer to the GLC a relevant error message will appear, in addition to the current "**Overall Progress**" display.




**While transferring data to the GLC, do not use the communication port for any other purpose.**

# 5.3 Operating the "Pump Project"

## ■ Downloading the Ladder Logic Program of Controller Objects to the GLC

1. Start the GP-PRO/PBIII.
2. Select **"Pump.prw"**, and click the **[Create Control]** command..
3. Select [**Write to Controller**] from the [**Controller**] menu.
4. The [**Download Progress**] menu will appear briefly.
5. Select [**Monitoring Mode**] from the [**Controller**] menu.
6. Select [**Start/Stop**] from the [**Controller**] menu. The [**Controller**]'s screen will appear.
7. Click on [**Start**].


***Even though this tutorial does not use examples of external I/O, when you wish to connect an external I/O device, be sure to set the [Controller/Command] menu's [Command] area [Enable IO] selection to ON.***

Reference *"3.2, Starting and Stopping the Controller"*

Now, the ladder logic program "**Pump.prw**" can be operated with the operation screens downloaded from your personal computer.

Reference *"Chapter 3, Running the Ladder Logic Program"*

## ■ Check the Project

The GP-PRO/PBIII project has been correctly designed and downloaded if it performs as follows:

1. Touch the GLC screen's [**ON**] button, and watch the fluid level shown by the bar graph on the screen drop as the pump empties the tank.
2. Touch the GLC screen's [**OFF**] button, and watch the fluid level rise because the pump is no longer emptying the tank.

If the project does not operate as explained above you need to modify the program.

### < Summary >

This chapter has explained how to:

- open the project manager.
- create the GP-PRO/PB III project linked with the controller.
- import Editor ladder logic program variables to a GP-PRO/PBIII project.
- associate the Editor's variables with GP-PRO/PBIII screen creation objects (i.e. Parts, Tags)
- download and then run a combined GP-PRO/PBIII /Editor application on a GLC.

*MEMO*

# 6 Pro-Control Editor and Pro-Server

**(GLC model: GLC2000 Series)**

When the Pro-Server is used, GLC variable read/writing and 2-way functions (communication, action functions etc.) can be executed via the Ethernet.

This Chapter explains how to use GLC variables with Pro-Server.

*Reference* *For details on Pro-Server, see the Operation Manual for Pro-Server with Pro-Studio for Windows.*

This Section explains how to import GLC variables with Pro-Server.

## 6.1 Importing GLC Variables

In order to use GLC variables in Pro-Studio, it is necessary to read in previously imported [GP-PRO/PB] GLC symbols to Pro-Server.

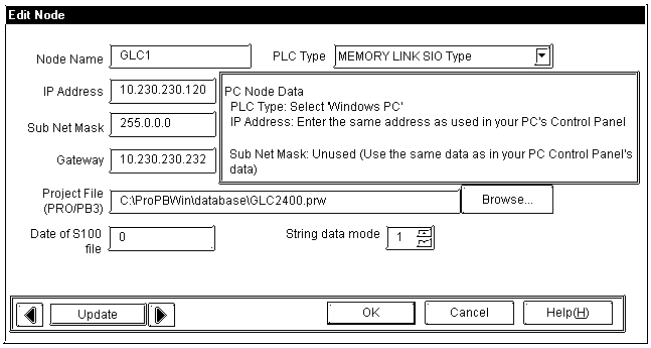
GLC local symbols:

- Are symbols which exist only for participating GLC stations
- Cannot be edited or deleted
- Have only the bit or 32-bit HEX device types.

### ■ To Import GLC variables

This section explains how to import GLC symbols into Pro-Server.

1. Start Pro-Studio.
2. Create a network project file.
3. Select **[Edit]** menu's **[Store participating station]** command.
   Select a PRW file to be linked to the **[PRO/PB III project file]**




4. Select **[Tool]** menu's **[Import GLC variable]** command.

The "S100 file date" is displayed as a property of the participating station.

5. The symbol name is displayed under **[Symbol name (Item)]** on the right of the main screen.

# 6.2 S100 File Check

If there is a difference in the content of any of the S100 files, i.e. those which are stored in each participating station, stored in the PRW file to be linked, and those that are imported to the GLC, it is possible that an incorrect device is being accessed. As a result, the system checks the dates of each S100 file and provides a warning display if they are different.

**1. Pro-Server (Pro_API) side**

When the network project file has been loaded, the system compares the property of the [S100 file date] of each participating station with the date of the linked PRW file. If there is a difference, a warning dialog is displayed.

**2. 2-Way Driver (GLC Unit)**

At start-up, the system compares the property of the [S100 file date] of each participating station with the date of the S100 file imported in the GLC. If there is a difference, a warning dialog is displayed.

# A.1 Errors and Warnings

Error or warning displays may appear in the [**Validity**] dialog box when a validity check is done on a program. These errors and warnings may be related to a problem with the program's logic, variables or I/O. The errors are indexed numerically, with each numeral being part of a specific range. Each range specifies a general area for you to focus on when determining why the error or warning has occurred.

## 200-299: Logic errors and warnings

*Reference For information on any of the ladder logic instruction, select it in the main window, and then from the [**Help**] menu select [**Context**], or press the [**F1**] key.*

◆ **Error 200 – Parameter should be a Discrete**

The instruction requires a Discrete operand. This can be:

• A Discrete variable,

• An element of a Discrete array, or

• A Discrete element of an Integer variable.

◆ **Error 201 – Parameter should be a Counter**

The instruction requires a Counter variable.

◆ **Error 202 – Parameter should be a Timer**

The instruction requires a Timer variable.

◆ **Error 203 – Parameter should be an Integer or Real**

The instruction requires an Integer of Real, either as a variable or a constant.

◆ **Error 204 – Parameter should be a non-constant Integer or Real**

The instruction requires an Integer or Real variable. It cannot be a constant.

◆ **Error 205 - Parameter should be an Integer**

The instruction requires an Integer as a variable or a constant.

◆ **Error 206 – Parameter should be an Integer but not an array**

The instruction requires an Integer, either as a variable or a constant. It cannot be an array.

◆ **Error 207 – Parameter should be a non-constant Integer**

The instruction requires an Integer variable. It cannot be a constant.

◆ **Error 208 – Parameter should be a label**

The instruction requires a label name, and a label with that name must exist.

◆ **Error 209 – Parameter should be a subroutine**

The instruction requires a subroutine name.

◆ **Error 210 – Label is out of scope**

The specified label exists, but cannot be reached from here.

◆ **Error 211- Subroutine cannot call itself**

The Jump Subroutine instruction is attempting to call the subroutine that contains it. This is not allowed.

◆ **Error 212 – X should be the same type as Y**

The two parameters should have the same type (Integer, Real, etc.).

◆ **Error 213 – X should be the same size as Y**

The two parameters must be the same size. That is, both must be either:

• Arrays with the same number of elements, or

• Non-arrays.

◆ **Error 214 – X should be the same size as Y or be an Integer.**

The two parameters must be the same size or the second can be an Integer that is treated as if it is the larger size.

◆ **Error 215 – X should be an Integer, a Real or a Discrete array**

The instruction requires an Integer, Real or Discrete, either as a simple variable or a complete array.

◆ **Error 216 – X should be a non-constant Integer, Real or Discrete array.**

The instruction requires an Integer, Real or Discrete, either as a simple variable or a complete array. It cannot be a constant.

◆ **Warning 217 – Both parameters are constants**

The instruction is comparing two constants.

◆ **Warning 218 – Input parameter used on output instruction**

The variable is marked as an input (refer to [Variable Type] window), however, it is used in an output instruction. Double-check its I/O assignment.

◆ **Warning 219 – Preset value is zero**

The preset value of the counter is set to zero.

◆ **Warning 220 – Preset time is zero**

The preset time of the timer is set to zero.

◆ **Warning 224 – Parameter should not be retentive**

The variables assigned to the instruction parameter cannot be “Hold” type.

◆ **Warning 225 – X should be an Integer Array**

The instruction requires Integer as a complete array.

◆ **Error 250 – Duplicate labels are not allowed**

The same label is defined more than once. This is not allowed, even in different sections of the program.

◆ **Warning 251 – Empty subroutines have no effect**

The subroutine contains no rungs. If you do not alter the empty subroutine it will have no effect on your program.

◆ **Warning 252 – Empty rungs have no effect**

The rung contains no instructions. If you do not alter the empty rung it will have no effect on your program.

◆ **Warning 253 – Empty branches have no effect**

The branch contains no instructions. If you do not alter the empty branch it will have no effect on your program.

◆ **Error 254 – Control instruction should be last on rung.**

The instruction cannot have any others to the right of it.

◆ **Warning 255 – X is used by more than one timer instruction**

The timer variable is used by more than one timer instruction. The results are indefinite.

Reference *You can use the [**References**] window to find the other instruction(s).*

◆ **Error 256 – X is used by more than one counter instruction**

The Counter variable is used by more than one counter instruction. The results are indefinite.

Reference *You can use the [**References**] window to find the other instruction(s).*

◆ **Error 257 – Last instruction on rung should be an output**

The instruction is not an output instruction (i.e., it does not change the values of its parameters).

◆ **Error 258 – Multiple outputs are not allowed**

An output instruction cannot have other instructions to the right of it.

◆ **Error 259 – Last instruction on branch should be an output**

An output instruction cannot have other instructions to the right of it.

◆ **Error 260 – Maximum level of nesting exceeded**

The rung has too many levels of branches (the maximum number of levels is 25). Try dividing the rung into several smaller ones.

◆ **Error 262 – Program is too large (by xx %), see Controller | Setup | Memory**

The program size is larger than the GLC Flash Memory.

◆ **Warning 263 - X is used by more than one coil.**

The variable is used by more than one coil. The result of the instruction with the most recently allocated variable by the logic program will be valid.

◆ **Error 264 -NEXT instruction not found.**

The NEXT instruction corresponding to the FOR instruction is not found.

◆ **Error 265 -FOR instruction not found.**

The FOR instruction corresponding to the NEXT instruction is not found.

◆ **Error 266 -FOR and NEXT instructions cannot be on the rung containing other instructions.**

Move other instructions from the rung containing a FOR or NEXT instruction.

◆ **Error 267 -The current platform does not support the instruction.**

The instruction cannot be used on the selected GLC type.

◆ **Error 268 -FOR-NEXT does not exist.**

JSR instruction or RET instruction exists between the FOR-NEXT instructions.

## 300-399: Variable errors and warnings

◆ **Warning 300 – Variable has input or output type but no I/O address assigned**

The variable is marked as an input or output (refer to the [**Variable Type**] window), however, it is not mapped to any I/O.

◆ **Error 301 – Type not assigned**

The variable has not been assigned a variable type. To assign a variable type use the [**Variable Type**] window.

◆ **Error 302 – Label not found**

The Jump Subroutine instruction refers to a label that does not exist.

◆ **Error 303 – Variable referenced should be a Timer or Counter**

You have specified an element of a Timer or Counter variable, however, the variable is actually of a different type. Refer to the [**Variable Type**] window.

◆ **Error 304 – Variable(s) referenced should be Integer type**

You have used a variable to specify an array element or modifier. This variable must be an Integer. Refer to the [**Variable Type**] window.

◆ **Error 305 – Array reference to non-array variable**

You have specified an element of an array, however, the variable is not designated as an array. Refer to the [**Variable Type**] window.

◆ **Error 306 – Array reference is beyond size of array**

You have specified an element of an array using a constant that is equal to or larger than the array's size. (Note that the valid elements are numbered 0 to size-1). You can change the size in the [**Variable Type**] window.

◆ **Error 308 – Modifier reference is out of range**

You have specified a bit, byte or word element that is out of range.

◆ **Error 309 – Reference is invalid for the variable**

You have specified a timer reference for a counter variable, or vice versa.

◆ **Warning 310 – …Already exists and will be replaced**

A variable by that name already exists. The new one will replace the original one if you click on [**OK**] in the [**Variable Import Status**] window.

◆ **Error 311 – The clipboard buffer is not a recognized format**

The current contents of the clipboard are not suitable for pasting into the [**Variable List**] window.

◆ **Error 312 – Too many warnings**

The [**Variable Import Status**] window only shows a certain number of warnings. If you see this message, there may be more warnings that does not show.

◆ **Warning 313 – Missing ]**

An array type requires the size enclosed in square brackets. For example, Integer [10].

◆ **Warning 314 – Array size is invalid …Assuming a size of 1**

This variable apparently is intended to be an array, however, the size is not recognizable. The size should be an integer within square brackets. For example, Integer [10].

◆ **Warning 315 – Unknown type …will be Not Assigned**

The text is not recognized as an Editor variable type. Possible causes are:

1. It is spelled incorrectly

2. It has leading and / or trailing blanks.

◆ **Warning 316 – Unsupported array type … Ignoring array settings**

That variable cannot be an array.

◆ **Error 317 – Invalid variable name…**

You have entered an invalid variable name.

◆ **Error 318 – Too many errors**

The [**Variable Import Status**] window only shows a certain number of errors. If you see this message, there may be more that it does not show.

◆ **Error 320 – Too many variables**

You have attempted to assign too many variables.

◆ **Error 321 – Too many variables**

You have attempted to assign too many variables. Reduce the number of variables.

## 400-499: Editor I/O errors and warnings

◆ **Error 400 – Variable Name has already been mapped**

The variable is mapped to more than one I/O point. Refer to the [**Configure I/O**] window.

## 500-549: Generic I/O driver errors

◆ **Error 501 – Internal variable mapped to I/O terminal**

The variable is marked as "internal", however, it is mapped to an I/O terminal. Refer to the [**Variable Type**] window.

◆ **Error 502 – Input variable mapped to output terminal**

The variable is marked as an input, however, it is mapped to an output terminal. Refer to the [**Variable Type**] window.

◆ **Error 503 – Output variable mapped to input terminal**

The variable is marked as an output, however, it is mapped to an input terminal. Refer to the [**Variable Type**] window.

◆ **Error 504 – Discrete variable mapped to analog terminal**

A Discrete variable cannot be mapped to an analog terminal.

◆ **Error 505 – Integer variable mapped to discrete terminal**

An Integer variable cannot be mapped to a discrete terminal.

◆ **Error 506 – Variable type not supported by I/O driver**

The I/O driver requires a different type of variable to be mapped to this terminal.

## 800-899: Specific I/O driver errors

Reference *For information about any errors pertaining to your I/O driver, refer to your I/O driver user guide.*

## 900-1000: Specific I/O driver warnings

Reference *For information about any warnings pertaining to your I/O driver, refer to your I/O driver's Help system.*

# A.2 Glossary of Terms

### ■ Array

A Discrete, Integer or Real variable can be designated as an array. This means that multiple elements of that type are allocated under a single name.


***If using a variable, and it goes out of range, a major fault is triggered.***


**To clarify the terminology, 'LimitSwitches' is the variable, and 'LimitSwitches[5]' is an expression representing one of its elements.**

### ■ Bit

The basic storage element, its value may be either 1 or 0.

### ■ Bookmark

An invisible marker that can be placed anywhere in your logic, allowing you to instantly return to that portion of your program.

### ■ Branch

A parallel path of execution on a rung.

### ■ Byte

A storage element containing 8 bits of information. A byte may be assigned values from 0 to 255. An Editor integer is composed of 4 bytes.

### ■ Clipboard

A temporary storage place maintained by Windows for copying and pasting data. This can be done between applications or within a single application.

### ■ Data Watch List Window

Shows data values as they change. You can adjust the update rate in the [**Preferences**] dialog box.

### ■ Demonstration Mode

A limited mode of operation that is suitable for demonstrating the features of Editor, however, not for developing and running a full application. A full-featured copy of Editor operates in demonstration mode if it is not authorized.

### ■ Descriptions

A description can be any amount of text, up to 32767 characters, that describes some part of your program. A summary of descriptions may be viewed with the [**Description Lis**t] window.

### ■ Discrete point

A point that can have one of two states: OFF or ON.

### Drag

To press and hold down the left mouse button, move the mouse, then release. The mouse pointer indicates whether this is a valid place to let go.

### Element

An element is a name for some part of a variable, rather than the whole thing. This part can be:

- An element of a Timer or Counter variable,
- An element of an array, or
- Part of an Integer; see Modifiers.

### Error (Fault Conditions)

There are three types: **Major**, **Minor**, and **I/O**.

A Major Fault is serious. When this occurs, the Controller stops executing logic immediately. The editor shows the state as "**MAJOR FAULT**". To clear the condition, the Controller must be reset using the [**Start/Stop**] window.

A Minor Fault is one that can be safely ignored.

An I/O Fault is a failure to read or write I/O in.

### Focus

A black rectangle that highlights a selection in the ladder logic

### Forces

Discrete points can be forced either ON or OFF. This overrides any actions the logic may take. For example, if a variable is forced OFF, but the logic is trying to turn it on, it stays off. A list of the forces in your program can be viewed with the [**Force List**] window.

### GLC Controller

The GLC Controller executes ladder logic and drives I/O. The Controller is invisible. It runs as a system service inside Windows and communicates with the user through the Editor. The GLC Controller's name is WALTZCTL.EXE.

### Hexadecimal

A base-16 representation of an integer value. These can be entered with 16# in front. For example, 16#FF is 255.

### IEC 61131-3

A standard developed by the International Electrotechnical Commission defining the printed and displayed representation of five control languages including: Instruction List (IL), Ladder Logic Diagrams (LD), Function Block Diagrams (FBD), Structured Text (ST), and Sequential Function Charts (SFC).

The smallest component in a rung which instructs the Editor Controller to perform a specific function (i.e., Discrete, Bit operand, Data control, Operand, Timer/Counter, and Program control instructions). Instructions in Editor are based on the IEC 61131-3 specification.

### ■ Instruction

The smallest component in a rung which instructs the Editor Controller to perform a specific function (i.e., Discrete, Bit operand, Data control, Operand, Timer/Counter, and Program control instructions). Instructions in Editor are based on the IEC 61131-3 specification.

### ■ Integer

A storage element containing 32 bits of information. An integer may be assigned values ranging form –2147483648 to 2147483647 (16#0000000 to 16#FFFFFFF in hexa-decimal). Integers cannot contain decimal points.

### ■ Internal Variable

A variable that is not mapped to an I/O point.

### ■ I/O

Input/Output . The Editor Controller connects to physical (real-world) devices through I/O hardware supplied by third parties.

### ■ I/0 Address

An address assigned to a variable when it is mapped to an I/O device. The format of an I/O address depends on the driver it is mapped to.

### ■ Label Name

A name containing up to 32 characters that identify or label a position within the ladder logic. It cannot start with a digit.

### ■ Ladder Logic

The collection of rungs that make up your application. So called because it looks vaguely like a ladder.

### ■ Off-Line

When Off-Line, the Editor works with the disk file '.WLL' containing a ladder logic program. This program is developed Off-Line and then run On-Line with the Controller.

### ■ On-Line

The Editor monitors a program which is running 'live' in the Controller. For example; Power_of_pushbutton, ResetButton, ALARM2 etc.

### ■ Parameter

An input to or output from an instruction. Parameters are entered into the Instruction Parameter Box.

### ■ Power Flow

The path power is taking through the ladder logic program.

Pro-Control Editor (Editor)

The Pro-Control Editor is the front end to the GLC Controller. All program development and monitoring take place here. The Editor's name is WALTZED.EXE.

### ■ Real

Any number containing a decimal point or being represented in scientific notation. The range for a real in Editor is ±2.25e-308 to ±1.79e-308. It can have up to 15 significant digits.

### ■ State Flow

Highlights individual instructions based on their parameters. Each contact is highlighted if it is able to pass power (as opposed to whether it actually gets power), based on the state of its parameter.

### ■ Subroutine

A group of rungs in a separate, named area.

Subroutines are placed between the END and PEND (Program End) markers, and cannot be placed within other subroutines. When you click on [**Subroutine**] from the [**Insert**] menu, both a "Subroutine Start" and a "Subroutine End" markers are created. You can then insert logic between the two.

Subroutine are called with a "Jump Subroutine (JSR)" instruction. The advantage is that they can be called from many places, and the code only needs to be written once. A subroutine name is required.

### ■ Subroutine Name

A Subroutine Name consists of up to 32 letters, digits, and / or underscores. It can only start with a letter.

### ■ System Variables

System Variables are special, predefined variables that provide information about the controller's status or affect its operation. They perform like ordinary variables, except that they are created automatically and cannot be deleted.

### ■ Variable

Storage locations for data values are called variables. Easy-to-understand names are recommended to use, rather than using numbered addresses. A variable name is up to 20 letters, digits, and / or underscores. It cannot start with a digit. Some valid examples are; Power_Off_pushbutton, ResetButton and ALARM2, etc. Editor creates an appropriate type of variable automatically as soon as a new variable name is entered either in [**Parameter Box**] or the [**Configure I/O**] window.

### ■ Watchdog Timer

Detects an error if the program did not finish running up to the "END" rung within a certain length of time. To set "Watchdog Timer", select [**Setup**] from the [**Controller**] menu, and enter time in millisecond in the [**Watchdog Timer**] box in the [**Tuning**] tab.

### ■ Word

A storage element containing 16 bits of information. A word may be assigned values ranging from 0.