Technical Information

TI 34P02K13-02E

STARDOM

FCN-500/FCN-RTU Primer – Fundamentals

YOKOGAWA
Yokogawa Electric Corporation

TI 34P02K13-02E
1st Edition: Apr.28, 2016
3rd Edition: Jun. 6, 2018

Blank Page

Introduction

■ About This Manual

This manual is targeted at users who are about to built their first control application on the Autonomous Controller FCN-500 and FCN-RTU. It not only provides theory, but also includes hands-on exercises for creating a simple control application so that a user may, through the exercises, understand the fundamentals of building an application, as well as learn how to use the relevant tools.

We recommend that you read Chapter 1 to understand the FCN-500 and FCN-RTU architecture before starting the hands-on exercises so that you may better understand why things are defined a certain way during the hands-on sessions.

Detailed procedures are provided in the hands-on exercises so that a user who has basic knowledge of Windows operation (keyboard and mouse operation) should have no difficulty in carrying out the exercises.

For the sake of simplicity, this manual uses the least terminologies and the most basic functions. For more details on functions and specifications, please refer to other available documentation (IM, TI, GS or online help).

● Related Technical Information

- Introduction to Engineering Know-how:
  FCN/FCJ Know-how
  (TI34P02K31-01E)
- Description of connection of VDS and FCN/FCJ:
  VDS Primer-Fundamentals
  (TI34P02K12-01E)

IMPORTANT

Notations in this document:

- The term “FCN-500” refers to the autonomous controllers with NFCP501/NFCP502 CPU module.
- The term “FCN-RTU” refers to the low power autonomous controllers with NFCP050 CPU module.

■ Organization of This Manual

Chapter 1: Introduces the minimum knowledge that is required for configuring a FCN-500 and FCN-RTU control application. You should read this chapter to understand the FCN-500 and FCN-RTU architecture before starting on the hands-on exercise.

Chapter 2: Introduces the problem example for the control application to be built in the hands-on exercises.

Chapter 3: Briefly describes how to perform hardware setup using the Resource Configurator.

Chapter 4: Briefly describes how to create a control loop and a sequence based on FBD (Function Block Diagram) using Logic Designer.

Appendices: Briefly describes how to perform setup for operating an FCN-500 or FCN-RTU from VDS, as well as how to create and delete global variables.

■ Useful Tips

Tips are provided throughout the hands-on exercises to enhance understanding of the FCN-500 and FCN-RTU functions.

STARDOM

FCN-500/FCN-RTU Primer – Fundamental

3rd Edition: TI 34P02K13-02E

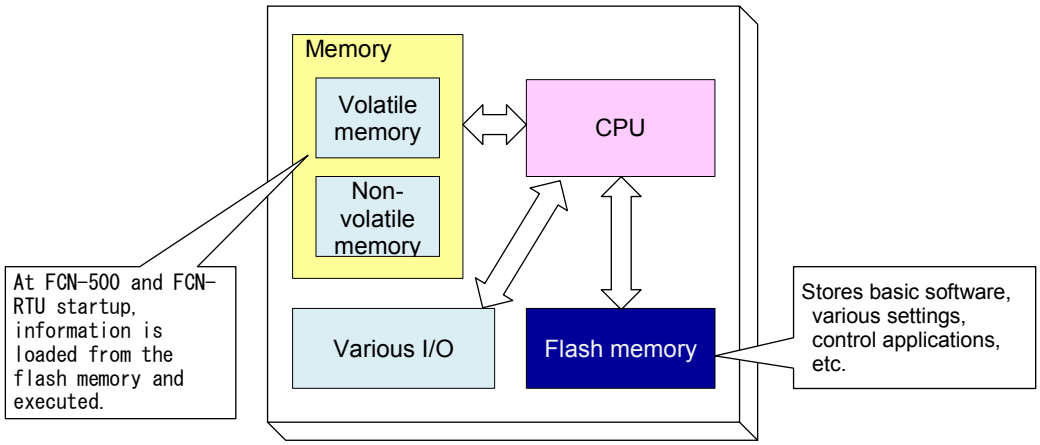
CONTENTS

1. Configuration of FCN-500, FCN-RTU

This chapter briefly describes the functions that are required for building a FCN-500 and FCN-RTU control application.

1.1 Internal Architecture of FCN-500, FCN-RTU

Before describing the configuration of a control application, which is the main subject of this manual, let us first introduce the internal architecture of an FCN-500 and FCN-RTU.
Within an FCN-500 and FCN-RTU, the CPU module operates by accessing various I/O and the on-board flash memory as shown in the figure below.

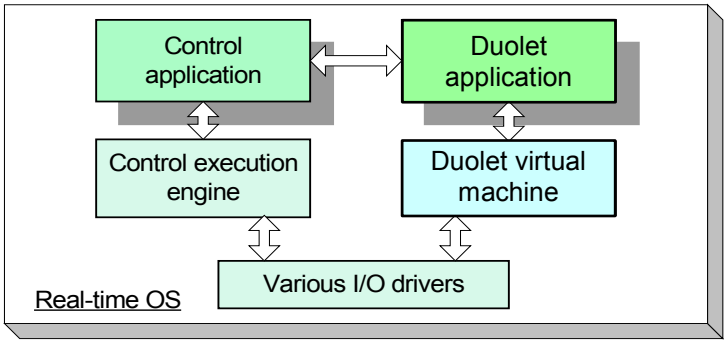


Whole information is pre-stored in the flash memory. Such information includes the basic software, namely, a Real-time OS, a control execution engine and a Duolet/Java virtual machine, as well as various hardware setup information, control applications (*1) and Duolet applications (*2).

*1: Control application is IEC 61131-3 programing language application.
*2: Duolet application is Java language application.

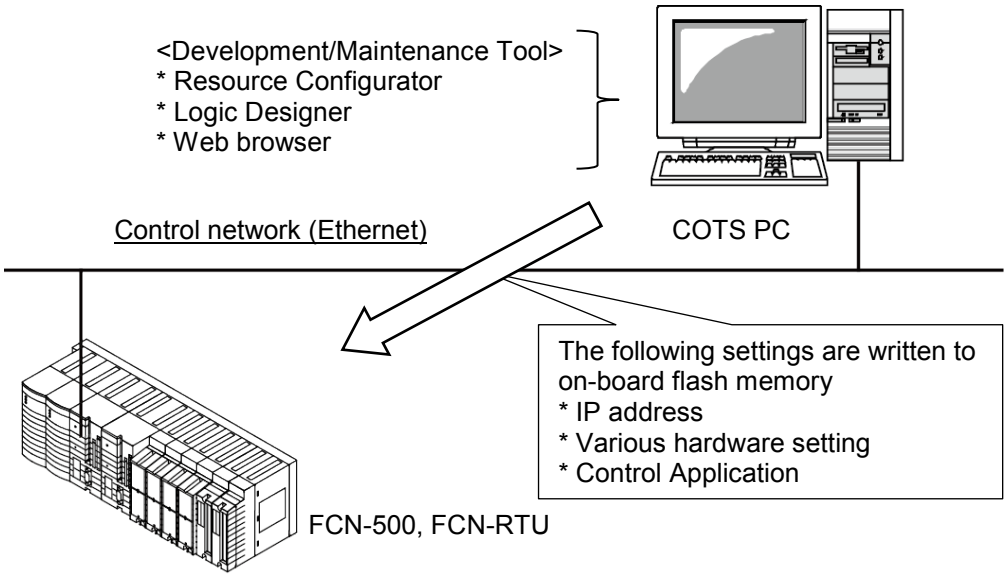
At FCN-500 and FCN-RTU startup, this information is loaded from the flash memory to the volatile memory, and executed. The non-volatile memory stores retentive data, which are values of variables specified to be retained, and RAS information.

The figure below shows the software architecture of an FCN-500 and FCN-RTU. Two engines, one for executing user-created control applications and the other for executing user-created Duolet applications communicate with various I/O drivers.



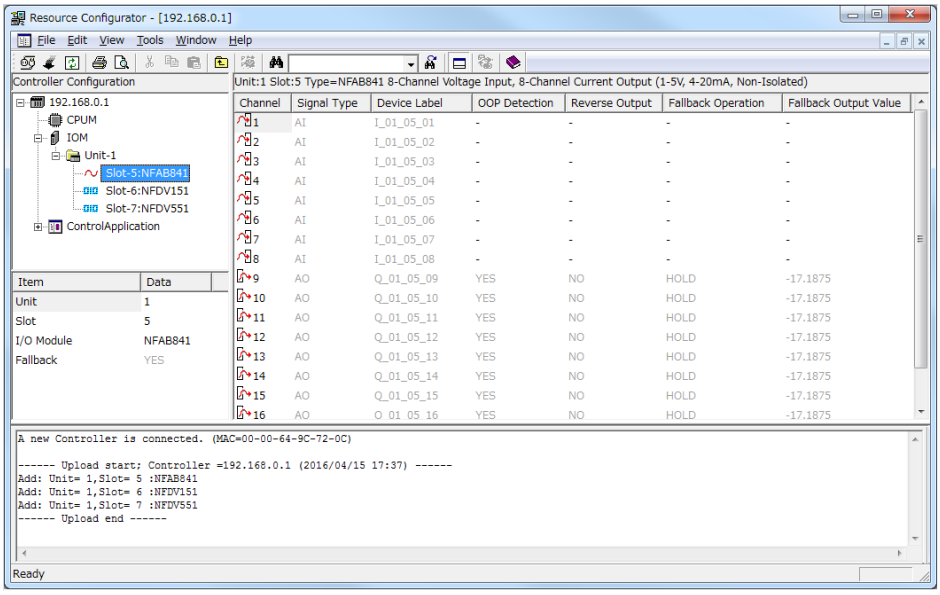
1.2 FCN-500, FCN-RTU Development and Maintenance Tools

Resource Configurator and Logic Designer are software tools to be installed on a PC for FCN-500 and FCN-RTU development and maintenance. Routine and advanced setup by accessing an FCN-500 and FCN-RTU as a Web server can also be carried out using a Web browser such as Internet Explorer.



1.2.1 Resource Configurator

The Resource Configurator tool can be used to perform basic setup such as setup of IP address of FCN-500 and FCN-RTU, setup of various hardware including I/O modules.



- **Downloading Setup Information**

  Downloading setup information to the FCN-500 and FCNRTU using Resource Configurator writes the setup information to the flash memory.

  In addition, setup information for each I/O module is downloaded to the I/O module.

  (Only setup information that has been changed is downloaded.)

TIP: I/O Module Setup Information

Setup information downloaded to an I/O module is stored in its memory. Hence, such setup information would need to be rewritten when an I/O module is replaced.
(When replacing an I/O module, you may specify to have the system automatically write setup information stored on the flash memory to the memory of the I/O module).

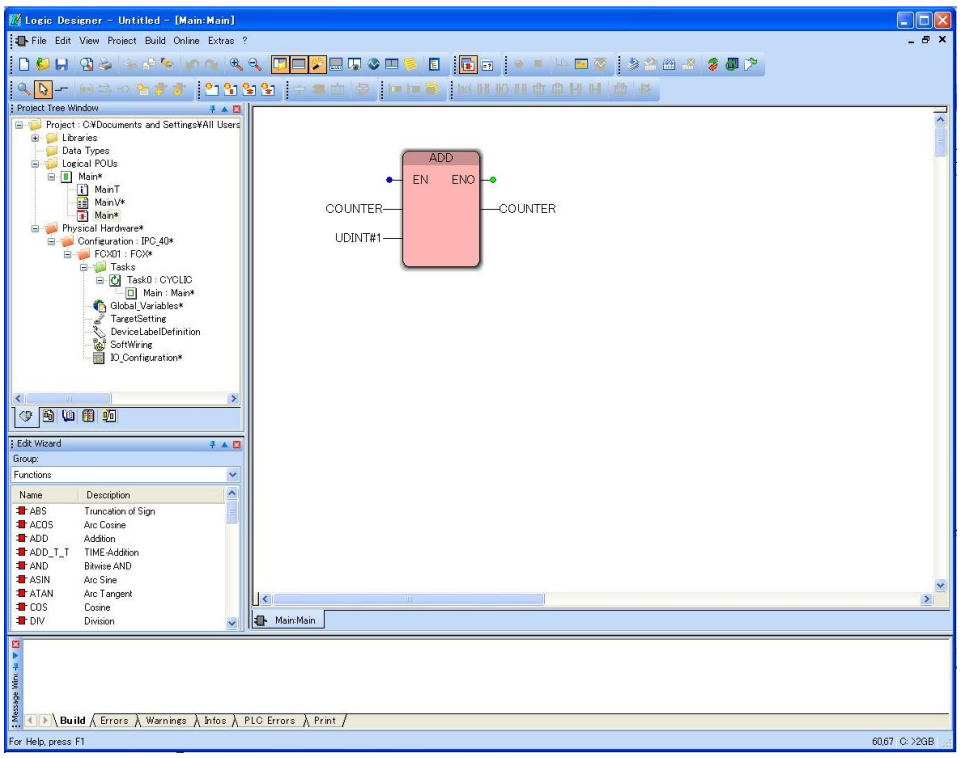
- **What Happens When You Connect to an FCN-500 and FCN-RTU**

  When you connect to an FCN-500 and FCN-RTU using Resource Configurator, the following information is automatically uploaded and displayed.

  - Configuration information of hardware and software
  - RAS information
  - Setup information stored in the flash memory

1.2.2 Logic Designer

The Logic Designer tool can be used to create and debug control applications to be executed on an FCN-500 and FCN-RTU.



● Programming Languages

The following 5 programming languages, all compliant to the IEC61131-3 International Standard are available, and may be selected based on your application requirements and preferences.

- FBD (Function Block Diagram)
  FBD implements data processing by modularizing various functions into blocks and connecting the blocks using signal lines. By supporting system development according to data flow, FDB is most suited for continuous control of analog signals.
- LD (Ladder Diagram)
  Ladder Diagram is the most widely used programming language for PLC. Using symbols of contacts and coils as basic elements, it enables graphical coding of logic operations.
- SFC (Sequential Function Chart)
  SFC is suited for building process sequence applications. It enables clear description of sequential control. Combining SFC describing sequential control and LD describing non-sequential control such as exception handling allows building of neat sequence programs.
- IL (Instruction List)
  Instruction list is a standardized mnemonic language. By allowing one operator and its operands to be coded on a single line, it avoids ambiguous expressions but is cumbersome for coding complex logical relationships.
- ST (Structured Text)
  ST is a textual language similar to high-level programming languages such as PASCAL, which allows multi-branch process applications to be coded using IF-THEN-ELSE statements, as well as complex expressions to be coded easily.

TIP: Combining Programming Languages

In principle, only one language is allowed in a POU (program). FBD and LD, however, can be intermixed, as well as be coded in SFC actions. SFC transitions and actions can be coded in FBD, LD, IL and ST.

● Encapsulation and Reuse of Control Application

The control application can be made into parts of functional units and can be easily reused.(we can made black box parts that have been hidden by a password).

By encapsulation and reuse, the total engineering efficiency including debugging becomes good.

● Debugging Support

Logic Designer provides the following functions, which greatly simplifies debugging and maintenance of control applications.

- Software Wiring Function
  Software wiring to I/O modules using connection definitions enables debugging without real external signal input.
- Online Layout and Value Display
  Values of variables in a program can be displayed and changed while a program is displayed.
- Setting Breakpoints
  Breakpoints can be set within a program to stop program execution. Stepwise execution can be carried out subsequently.
- Logic Analyzer Function
  The logic analyzer function records variable values at specified time intervals, and displays the result graphically.
- Watch Window
  Registering variables to a Watch Window enables monitoring of their values.

● Online Download Function

The online download function can be used to change or modify a control application without interrupting control.

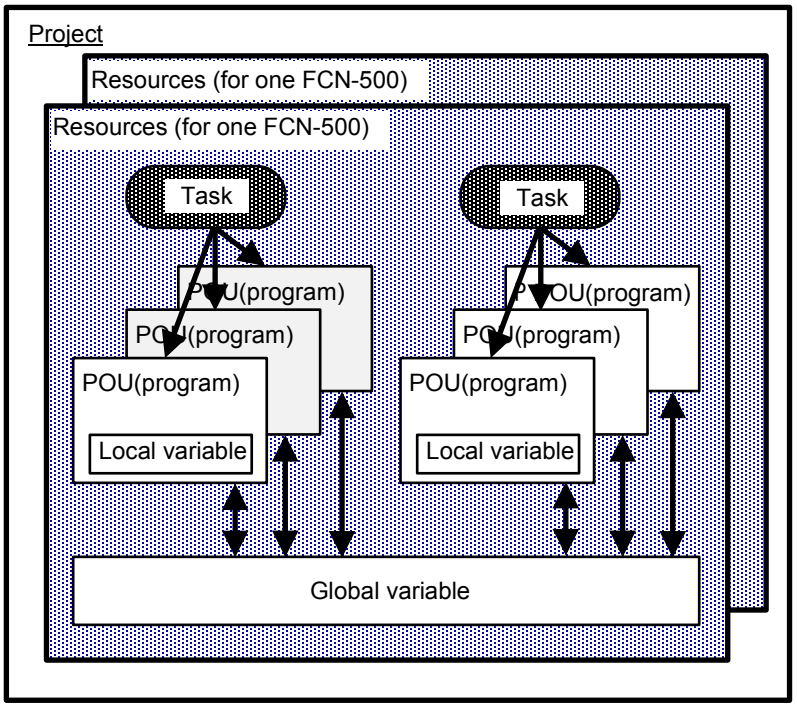
The following table shows whether online downloading is allowed for each type of modification in Logic Designer.

Modification	Allowed (Yes/No)
Add, delete or modify a comment	Yes
Add, delete or modify a library	Yes
Add, delete or modify a data type definition	Yes
Add, delete or modify a variable	Yes
Add, delete or modify a logical POU	Yes
Add, delete or modify a POU instance	Yes
Add, delete or modify program code	Yes
Add or delete a configuration	Yes
Add or delete a resource	Yes
Add, delete or modify a global variable	Yes
Add, delete or modify a device label variable	Yes
Modify PLC type	No
Modify processor type	No
Add, delete or modify a task	No
Modify priority	No
Modify watchdog time	No

1.3 Components of a Control Application

1.3.1 Overall Structure

Let's look the overall structure of a FCN-500 and FCN-RTU control application created using Logic Designer.



- **Project**

  A project is a collection of control applications created using Logic Designer. Control applications for multiple controllers can be stored within one project.

- **Resource**

  Each resource is associated with one FCN-500 or FCN-RTU unit. Resources contain multiple programs and various variables, which can be grouped arbitrarily.

- **POU (Program)**

  Programs are control applications created using any of the five programming languages described earlier. A control application can be arbitrarily divided into function units, known as POU (Program Organization Unit) in IEC61131-3 nomenclature.

SEE ALSO

For more details on POU, see Section 1.3.2, "Control Application Unit (POU)".
For more details on local variables and global variables, see Section 1.3.3, "Local Variables and Global Variables".

- **Task**

  A task manages the scheduling of control applications (POU) based on execution period, watchdog time or other parameters. A created program cannot be executed unless it is assigned to a task. One or more POUs can be assigned to a single task.

1.3.2 Control Application Unit (POU)

As described earlier, a control application can be arbitrarily divided into function units known as POUs. A POU can be classified into any of the following three types according to its form.

- **Program**

  Of the three types of POUs, only program POUs can be assigned to a task. A control application is usually built hierarchically by combining function blocks and functions, which are described below.

- **Function Block**

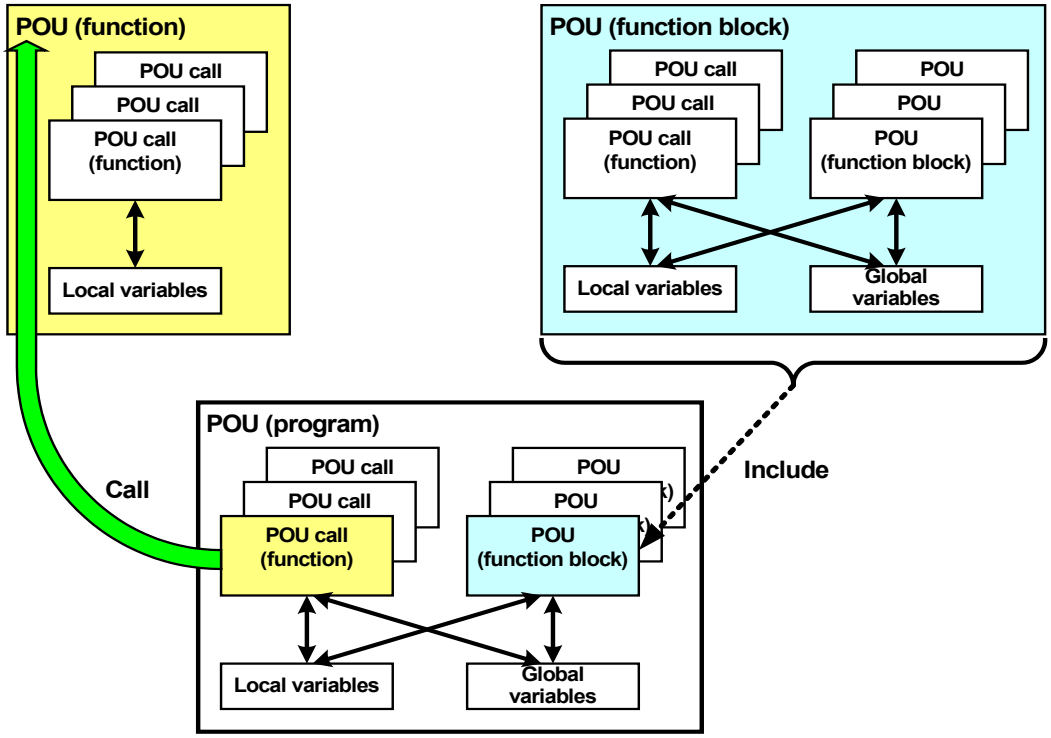
  Like a program, a function block can consist of other function blocks or functions, but it cannot be assigned to as task and be executed on its own. It can only be executed by including it within a program.

- **Function**

  A Function (such as ADD and NOT) can have one or more input parameters, but can only have one output parameter. It has no internal memory, and thus always produces the same output value given the same set of input values.

  Functions cannot be built from function blocks, but can contain other functions. Global variables cannot be used within functions.

  Unlike function blocks, functions are not executed by combining it with other function blocks or programs but are executed through function calls.



1.3.3 Local Variables and Global Variables

Variables used in control applications are classified as global variables or local variables according to their scope.

- **Local Variables**

  A local variable can only be used within a POU (program, function block or function) where it is defined.

- **Global Variables**

  A global variable is accessible by all executing POUs within the same resource.

  Each variable has several properties, in addition to its scope.

  - RETAIN property
    A variable with its RETAIN property turned on retain its value when the FCN-500 or FCN-RTU power supply is cut off. (To retain the value of a variable at power off, this property must be turned on using Resource Configurator).
  - OPC property
    A variable with its OPC property turned on is accessible (read/write) by SCADA or other software outside of the FCN-500 or FCN-RTU.

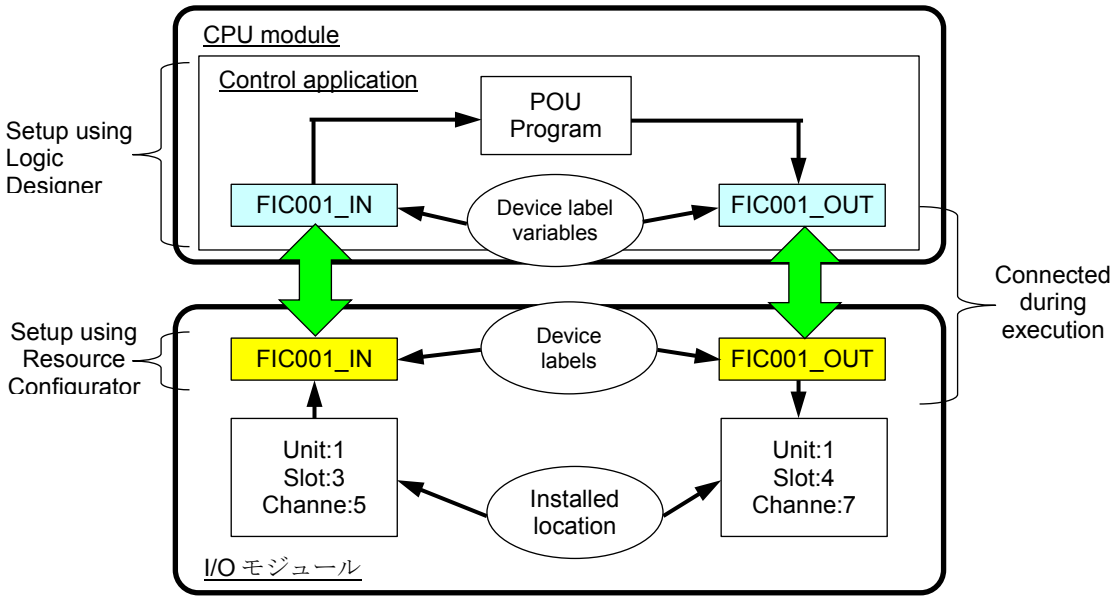
1.4 Connecting Control Applications and I/O Terminals

Development of control applications and definition of I/O assignment on FCN-500 and FCN-RTU can be carried out independently.
On an FCN-500 and FCN-RTU, physical inputs and outputs of an I/O module (or I/O terminals of an FCN-RTU itself) can be defined with names known as "device labels".
(Device labels may be defined using the Resource Configurator, or default device labels may be used.)

- Default Device Label
  I/O Module of FCN-500 or FCN-RTU : x_uu_ss_cc
  (x: signal type; uu: unit number; ss: slot number; cc: channel number)
  Built-in I/O of FCN-RTU : x_d_uu_ss_cc
  (x: signal type; d: device type; uu: unit number; ss: slot number;
  cc: channel number)
  x: signal type (I: input signal, O: output signal)
  d: device type (A: analog input/output, D: digital output,
  P: pulse input)

On the other hand, logical I/O names known as "device label variables" are used when control applications are created using Logic Designer. This allows creation of control applications without knowledge of the physical installed location of the I/O module.

Each device label variable is then connected to the device label with the same name when the control application is executed.



2. Introduction of Hands-on Exercise

This chapter describes the system configuration and control application to be built in the hands-on exercises described in Chapters 3 to 4.

2.1 System Configuration

The system configuration to be used for the hands-on sessions is shown below. When doing the exercises on your own, you may need to modify the IP address or other parameters to match your specific environment. Procedures for installing the various tools and registering software licenses are not described in this manual.

- **FCN-500 or FCN-RTU (1 unit)**

  The exercises are described assuming the use of an FCN-500.

  In the exercises, use the PAS portfolio and analog I/O function. Prepare the FCN-500 or FCN-RTU.

  - FCN-500 : NFCP501-W☐☐ or NFCP502-W☐☐ CPU module (With extended functions) (*1), analog input module and analog output module.
    *1: NFCP501-S☐☐ or NFCP502-S☐☐CPU module (standard type) is not available for this exercise.
  - FCN-RTU : NFCP050-S1☐ CPU module

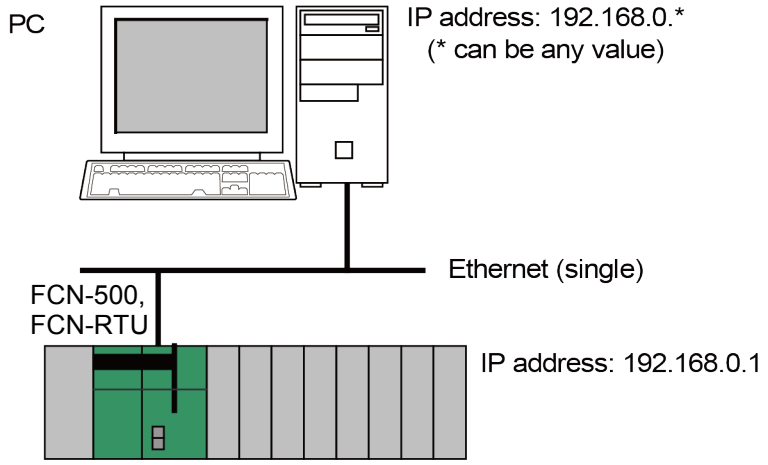
- **Personal Computer**

  The following software must be installed for building control applications:

  - Resource Configurator
  - Logic Designer
  - VDS (used only for checking operation)

- **Network**

  The exercises are described assuming a single Ethernet configuration.

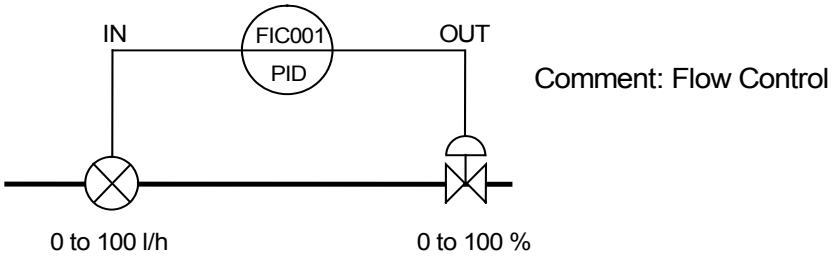


2.2 Control Application Example

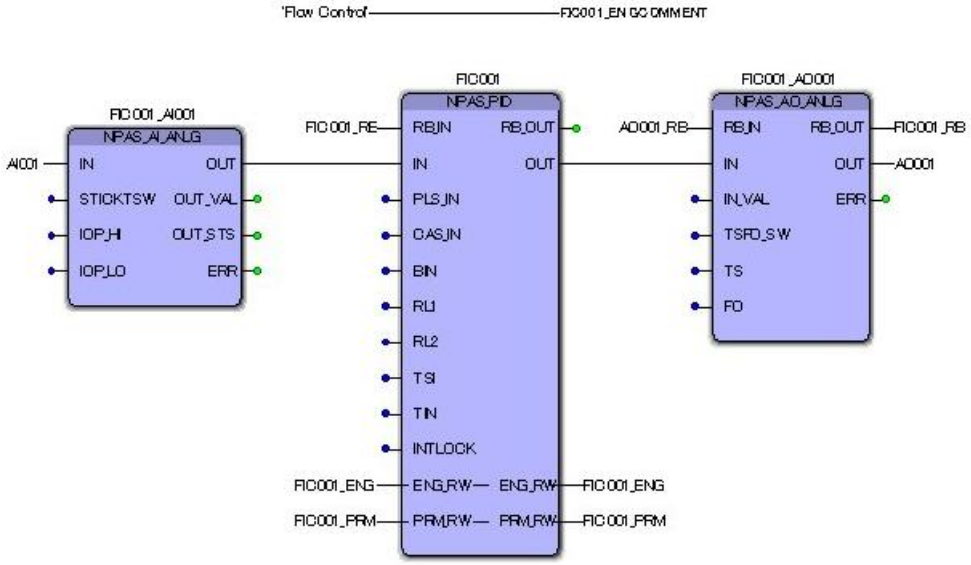
The problem example for the control application to be built in the exercises is described below.

- **Regulatory Control**

  The following single-loop PID controller will be registered in this exercise.



The figure below shows the control application that will be built.

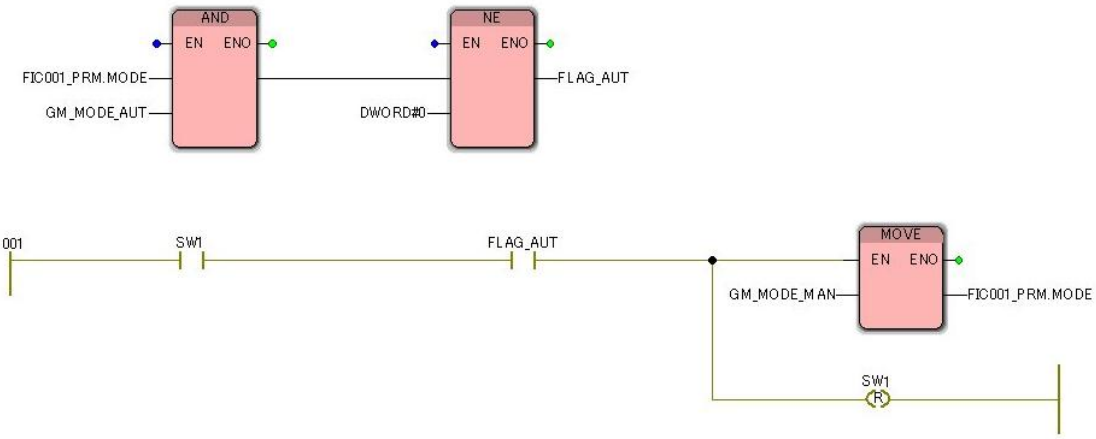


TIP

The first-order lag filter is omitted from the control application hands-on exercise for simplicity sake.

- **Sequence Control**

  We will build a sequence control that switches a PID controller to MAN mode when an internal flag (SW1) turns on with the PID controller in AUT mode.



TIP

The SW1 contact type is assumed to be status instead of one-shot type in the sequence application hands-on exercise for simplicity sake.

Blank Page

3. Hands-on with Hardware Setup

Let's do a hands-on exercise together.
In this chapter, we shall perform hardware setup using Resource Configurator.

3.1 Hands-on: Online Setup

Before creating a control application, we shall describe how to set up various hardware of the FCN-500 or FCN-RTU. These operations are done using Resource Configurator but can be carried out either online or offline. Section 3.1 introduces how to perform online setup with the FCN-500 or FCN-RTU connected to the Ethernet, while Section 3.2 introduces how to perform offline setup with the FCN-500 or FCN-RTU unconnected, and subsequently download the setup after connection.

3.1.1 Starting Resource Configurator

Installing the Resource Configurator software creates a "Resource Configurator" icon on the desktop.

Double-clicking this icon starts Resource Configurator.

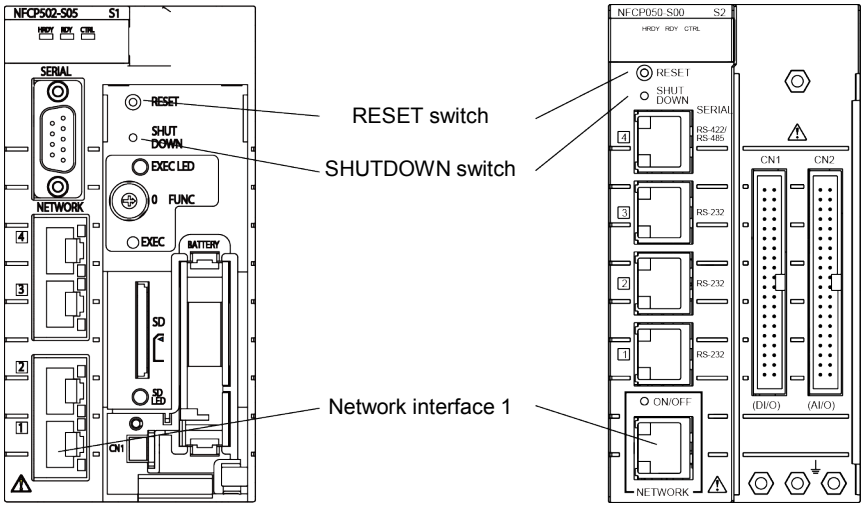
3.1.2 Setting IP Address

First, we shall set up the IP address of the FCN-500 or FCN-RTU. For a newly purchased FCN-500 or FCN-RTU, the system automatically goes to step (4) of the procedure below when the power is turned on, so you can continue the procedure, starting from step (5).

(1) First, connect an Ethernet cable to "Network Interface 1" of the FCN-500 or FCN-RTU.

(2) Power on the FCN-500 or FCN-RTU, or press the [RESET] button.

(3) After the "HRDY" and "RDY" LEDs starts blinking, push the [SHUT DOWN] button of the FCN-500 or FCN-RTU once within 3 seconds. This will bring the FCN-500 or FCN-RTU into IP address setup mode.

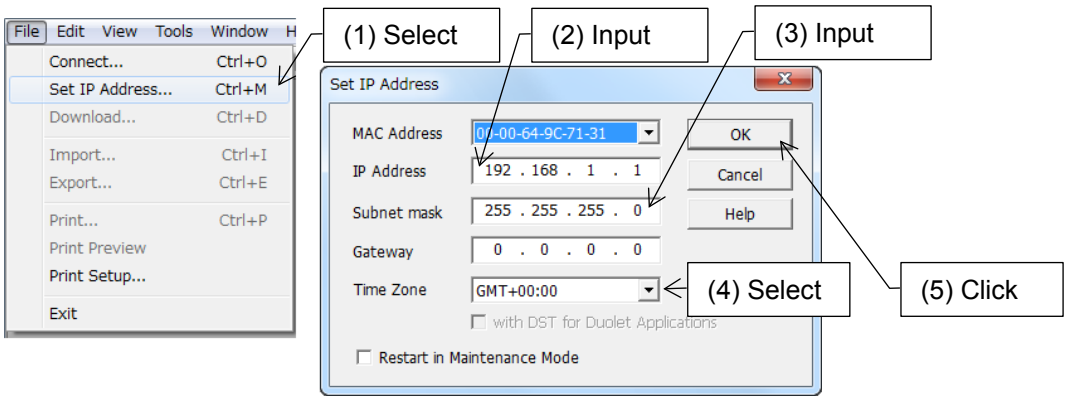


(4) The blinking of the RDY LED slows down, and the following message is displayed in the message window at the bottom of the Resource Configurator: "A new controller is connected."
(If the message is not displayed, go back to the step (2).)

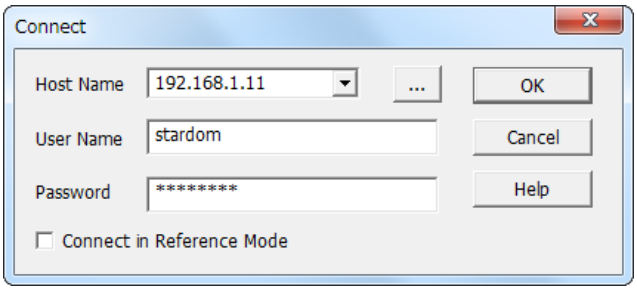
(5) Select [File]-[Set IP Address] from the menu bar to display the [Setting IP address] dialog.

(6) Set the IP address to "192.168.0.1" and the subnet mask to "255.255.255.0" as required for the hands-on exercise.

(7) Set Time Zone to local time zone and click [OK].



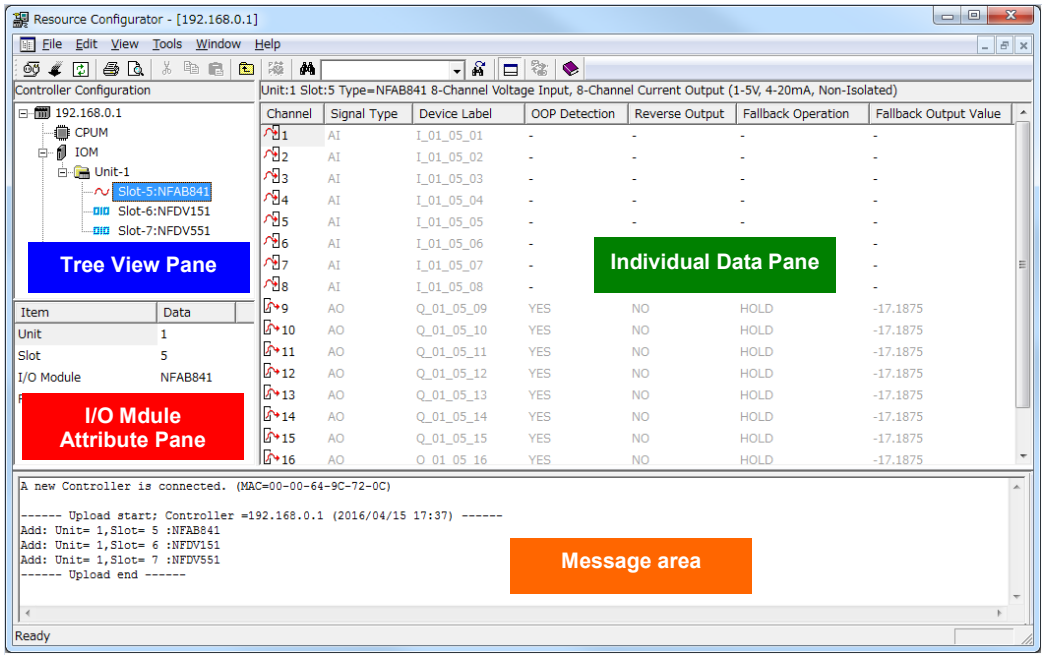
(8) After data is written on the flash memory, the Connect Dialog appears. Enter "User Name" and "Password" and click [OK] to continue the settings.



TIP: Connecting to FCN-500 or FCN-RTU

To connect to the FCN-500 or FCN-RTU at any other time, select [File]-[Connect] from the menu bar. The default user name is "stardom", and the user password is "YOKOGAWA". After connection completes, an "Upload end" message is displayed in the message window.

(9)The following screen is displayed when the connection has been completed.



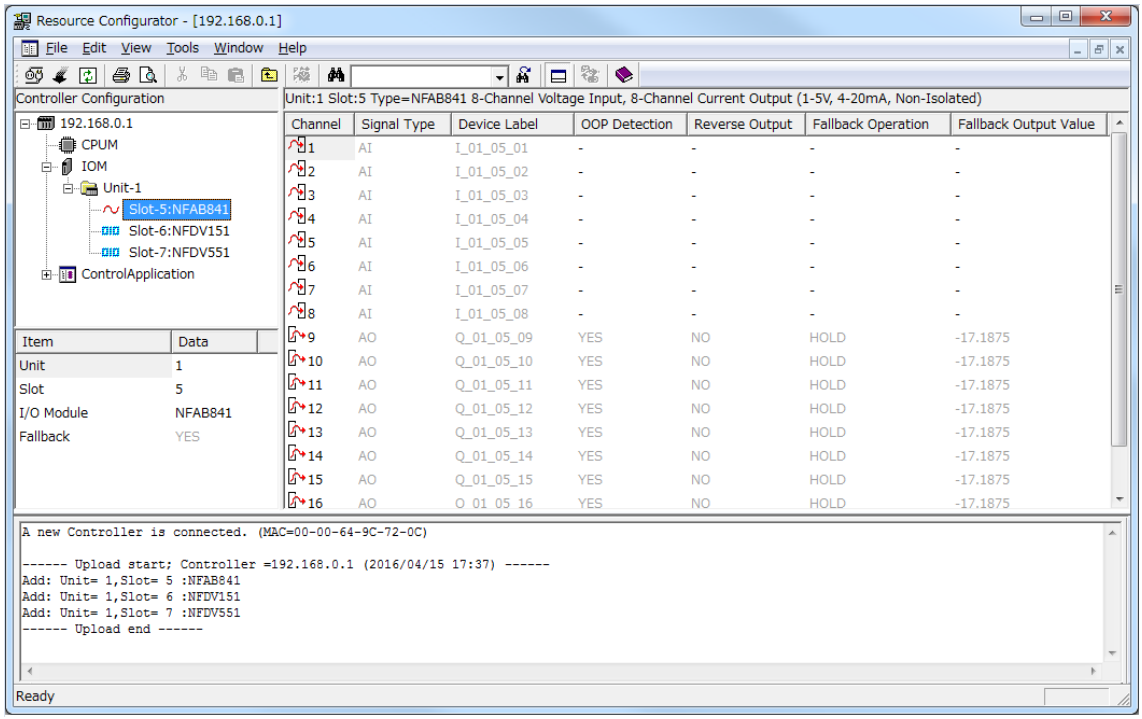
3.1.3 Defining Device Labels

We shall now define arbitrary names (known as device labels) for each physical I/O of the I/O module (or built-in I/O of the FCN-RTU).
This is the procedure in the case of analog input and output modules NFAB841 (8-point input/8-point output) is attached to the slot 5 of the unit 1.
Depending on the actual module of the system to be used, please read as unit number, slot number and channel number.

SEE ALSO

For more details on device labels, see Section 1.4, "Connecting Control Applications and I/O Terminals."

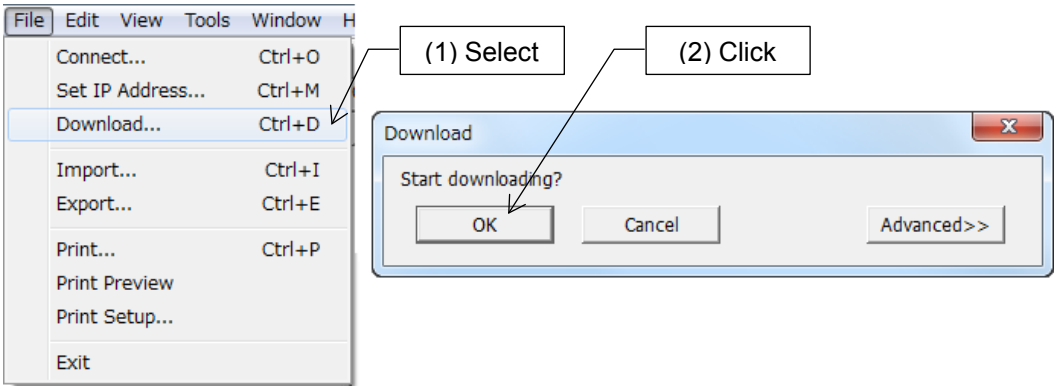
(1) Click "IOM" on tree view pane.
(2) Click "Unit-1" to display the I/O modules on the Unit-1.
(3) Click "Unit-5:NFAB841" to display the AI/AO definition information.
(4) Set up the device labels in this exercise:
- Define device label (I_01_05_01) of channel 1 as "AI001".
- Define device label (O_01_05_09) of channel 9 as "AO001".



3.1.4 Downloading Setup Information to FCN-500, FCN-RTU

Let's download basic setup of the CPU module or I/O module, to the FCN-500 or FCN-RTU.

(1) Select [File]-[Download] from the menu.
(2) When prompted with the download message, select [OK].



TIP: Writing IP Address

IP address is not written when setup information is downloaded, but can be written to the system card using step (8) of Subsection 3.1.2.

TIP: Setup Information in Resource Configurator

In addition to network setup and checking, and device label definition, Resource Configurator also provides functions for "CPU Module Setup", "I/O Module Setup" and "RAS Information Display".

CPU Module Setup

CPU module setup includes single/duplex network (only NFCP-500) setup and enabling/disabling of Duolet feature. CPU module setup also includes setup of retentive data, whose values are retained when power is switched off.

I/O Module Setup

I/O module setup includes device label definition, as well as setup for output open detection, reverse output definition, fallback operation (I/O operation when disconnected from CPU) and burnout..

RAS Information Display

RAS information display enables checking of various RAS-related information.

3.2 Hands-on: Offline Setup

If you are creating a control application without a target FCN-500 or FCN-RTU, you should follow the procedure described in this section instead of the procedure given in Section 3.1.

3.2.1 Starting Resource Configuration Editor

When performing offline hardware setup, use Resource Configuration Editor instead of Resource Configurator described in Section 3.1.

(1) Select [Program]-[YOKOGAWA FCN-FCJ]-[Support Tools]-[Resource Configuration Editor] from the Start menu to start Resource Configuration Editor.

3.2.2 Basic Setup of CPU Module and I/O Modules

We shall now perform basic setup of the CPU Module, as well as define device labels for an I/O module.

(1) Select [File]-[New] from the menu, followed by the appropriate controller type.
(2) Click "CPUM" on the tree view pane, and specify basic setup information for the CPU module on the displayed screen.
(3) Next, click each I/O module within "IOM" on the tree view pane, and define basic I/O module setup information such as device labels on the displayed screen.

SEE ALSO

For more details on device labels, see Section 1.4, "Connecting Control Applications and I/O Terminals."
For more details on how to define device labels, see Subsection 3.1.4, "Defining Device Labels".

(4) After completing the definitions, select [File]-[Save As] from the menu, and enter any desired file name for saving the definitions.

3.2.3 Downloading to FCN-500, FCN-RTU

After connecting online to the target FCN-500 or FCN-RTU, set up and download various pieces of information to the FCN-500 or FCN-RTU using the following procedure.

(1) Start Resource Configurator using the procedure given in Subsection 3.1.1.
(2) Set up the IP address of the target FCN-500 or FCN-RTU using the procedure given in Subsection 3.1.2.
(3) Select [File]-[Import] from the menu to load the definition file saved in Subsection 3.2.2.
(4) Download setup information to the FCN-500 or FCN-RTU using the procedure given in Subsection 3.1.4.

TIP: Functions Available Only in Online Mode

IP address setup can only be carried out in online mode.
Similarly, RAS information display is only available in online mode.

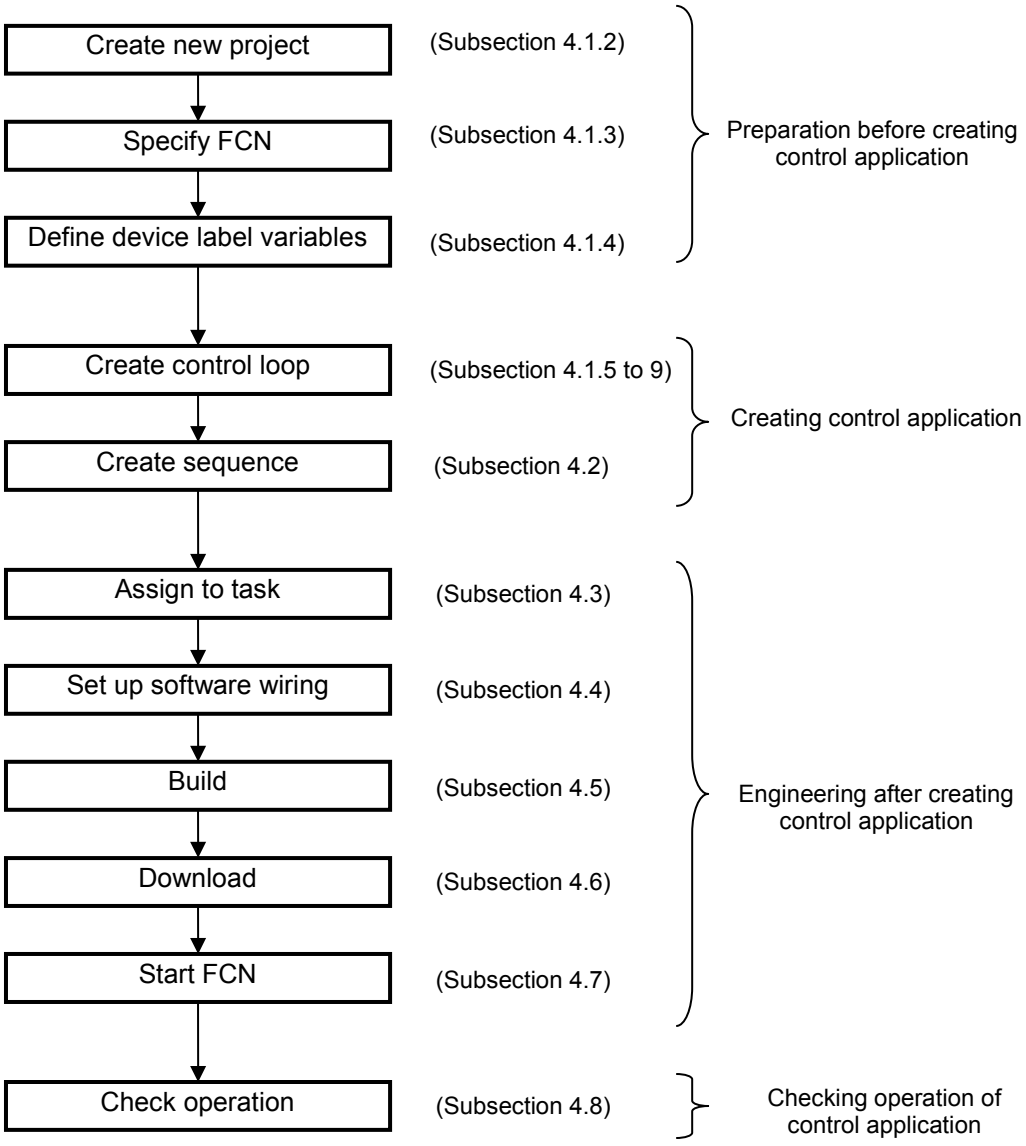
Blank Page

4. Hands-on with Logic Designer

Logic Designer can be used to create a control application.
In this chapter, we shall briefly describe how to build a control loop and a sequence using FBD (Function Block Diagram).

■ Engineering Procedure

The following flowchart shows the engineering work that will carried out in this chapter.



4.1 Hands-on: Creating a Control Application (Control Loop)

We shall now create the control application example described in Chapter 2. Logic Designer can be used to create control applications to be run on an FCN-500 or FCN-RTU.

4.1.1 Starting Logic Designer

Start Logic Designer either by double-clicking the Logic Designer icon on the desktop, or by selecting [Start]-[Programs]-[YOKOGAWA FCN-FCJ]-[Logic Designer].

TIP: Project Displayed When Logic Designer Is Started

It is possible to set that Logic Designer opens the previously opened project automatically, when it starts.
"Open Last Project on Program Start" is the default setting.
To do so, select [Extras]-[Options] from the menu after starting Logic Designer, and then perform the required setup on the [General] tab of the Options dialog.

4.1.2 Creating a New Project

A project is a collection of control applications to be created using Logic Designer. Control applications for multiple controllers can be stored within one project. We will now create a new project.

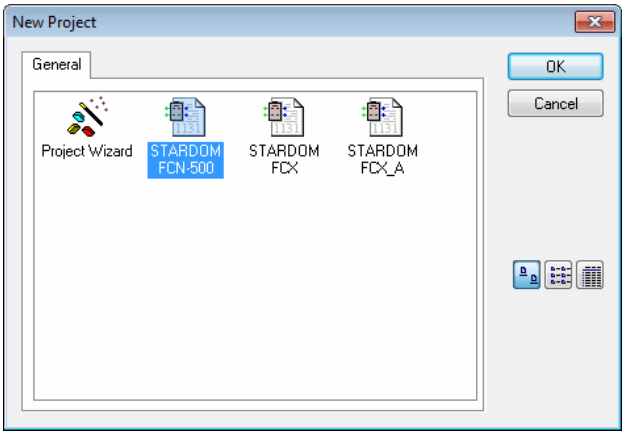
(1) Select [File]-[New Project] from the menu to display the New Project dialog.

Logic Designer provides template projects for different control application types to facilitate development. For this exercise, select "STARDOM NPAS" to make use of NPAS POUs of the PAS Portfolio: e.g. PID Controller.

TIP: PAS Portfolio

PAS Portfolio is a piece of optional software for FCN-500 (NFCP501-W☐☐, NFCP502-W☐☐ CPU module with extended functions model) and FCN-RTU. It provides as Function Blocks for various regulatory control instruments such as PID controllers or manual loaders. PAS Portfolio provides two types of POU, namely NPAS POU and PAS POU. NPAS POUs are normally used to build control applications.

(2)Select the “STARDOM FCN-500” icon for FCN-500 or “STARDOM FCX_A” icon for FCN-RTU, and then click [OK].



TIP: Types of Template Projects

Template project contains pre-defined required information, and thus enables simple and correct project creation. The templates on following table are used for creating control applications that use NPAS POUs and serial communication function blocks. “Project Wizard” is normally not used.

Template	Description	Controller
STARDOM FCN-500	Processor type: FCX_B This template is expanded the retain data size than FCX_A.	FCN-500
STARDOM FCX (*1)	Processor type: FCX This is for control applications that do not use the NFLF-111 /NFGS813/NFGP813/NFLP121/NFLC121.	FCN-RTU
STARDOM FCX_A (*1)	Processor type: FCX_A This template is expanded the number of devices label than FCX. Time required for it to download will be longer. This is for control applications using NFLF-111/NFGS813 /NFGP813/NFLP121/NFLC121.	FCN-RTU

*1: If using the "STARDOM FCX" or "STARDOM FCX_A" templates to FCN-500 controller, the maximum size of the retain data that can be used will be the half of the hardware specifications. Usually, for a new project created for the FCN-500, use the template project "STARDOM FCN-500".

For the project after creation, the processor type definition can't be changed. To use another processor type definition, create a new resource. For the procedure, refer to the online help of the Logic Designer.

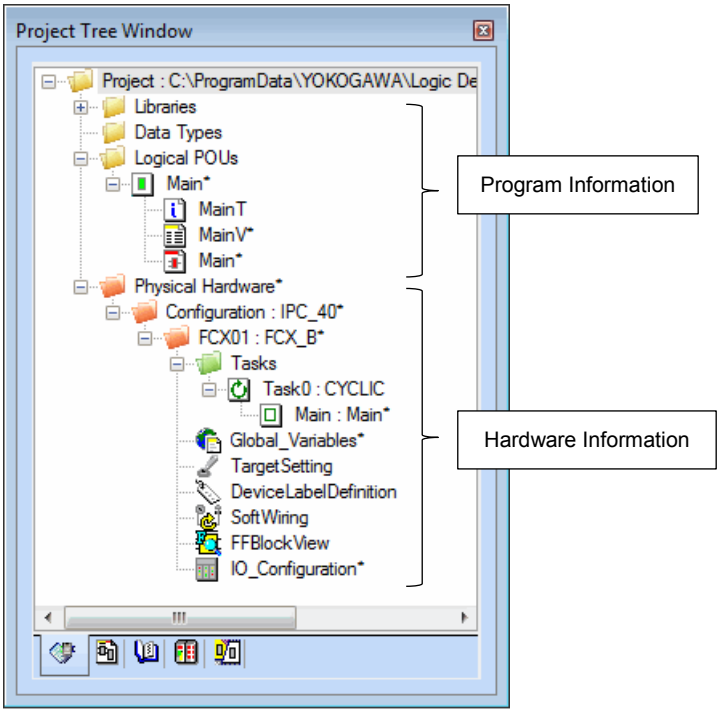
Processor type: FCX_C
This processor type is device label extended version of FCX_B, for NFCP501/NFCP502 with three or more extension units and NFLF111/NFLP121/NFLC121.
When using it, create a new resource with processor type FCX_C.

	FCX_C	FCX_B	FCX_A	FCX
Maximum IO group number (*1)	120	100	100	25
Maximum device label variable definition number (*2)	7680	6400	6400	1600

*1: Number of input groups only. The number of output groups is also the same.
*2: Number of device label variable definitions of logic designer. Number of input variables only. The number of output variables is also the same. Convert with DI variable.

TIP: Project Configuration

Project information can be broadly classified into program-related information and information on hardware to which program-related information is to be downloaded.



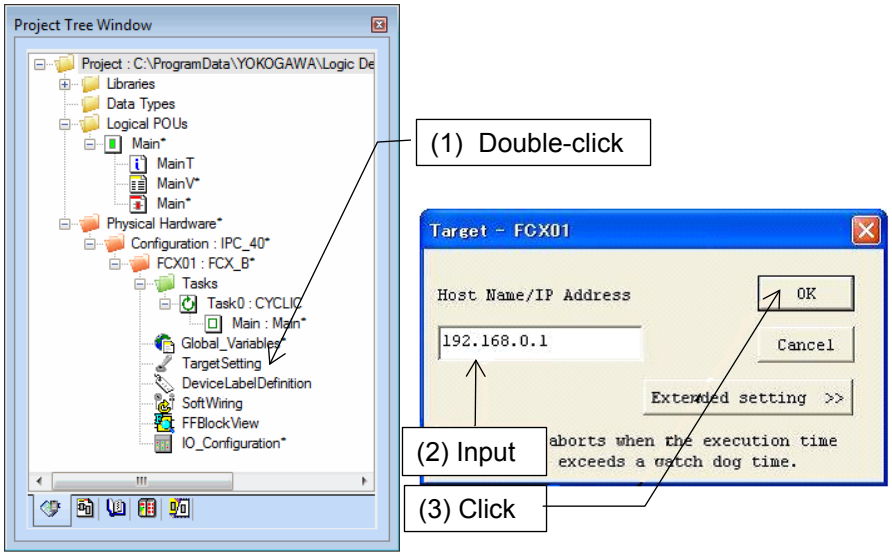
TIP: Hardware Information

PLC type for FCN-RTU is not "IPC_40" but "SH04-40".

4.1.3 Specifying Target FCN-500, FCN-RTU (by Specifying IP Address)

We will now specify the target FCN-500 and FCN-RTU for the control application to be created by specifying the IP address of the FCN-500 and FCN-RTU. In this exercise, we shall select the controller configured with IP address "192.168.0.1" in Section 3.1.2 as the target FCN-500 and FCN-RTU.

(1) Double-click [TargetSetting] on the project tree to display the Target dialog box.
(2) Enter "192.168.0.1" for the IP address, and then click [OK].



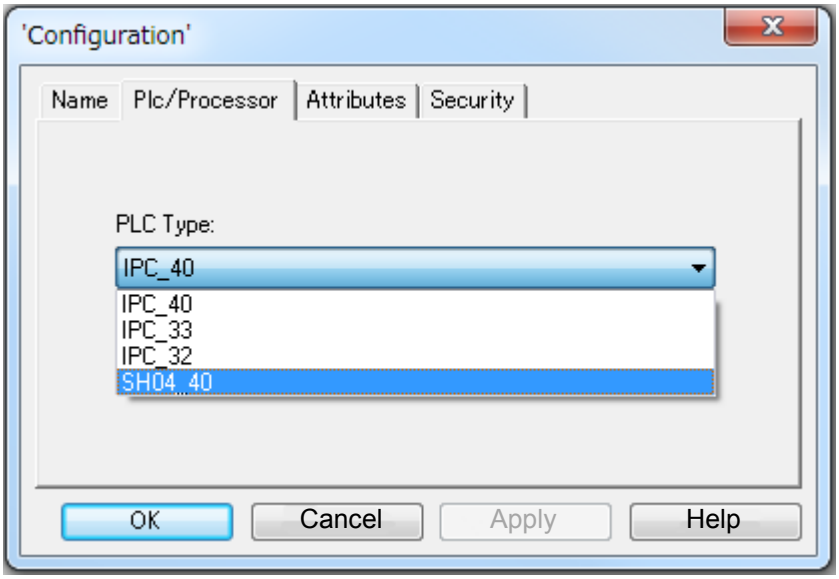
4.1.4 Changing PLC Type

When a project for the RTU is created, you need to change the PLC type.

(1) Right-click on [Configuration: IPC_40] on the project tree, and select the [Properties].

(2) Select the "PLC/Processor" tab:

(3) Change the PLC type from [IPC_40] to [SH04_40], and then click the [OK] button. Click the [OK] button for the confirmation message.



IMPORTANT

When debugging by the simulator, change the PLC type to “SH04_40”.
When debugging is complete, return the PLC type to the “SH04_S40”.

TIP: Hardware Information

If you change the PLC type for the compiled project, select [Build] - [Re-compile] in menu bar to re-compile.

Table The combination of PLC type and controller

PLC type	Description
IPC_40	For the FCN-500 and FCN/FCJ simulator.
SH04_40	FCN-RTU only.

4.1.5 Defining Device Label Variables

In this section, we shall define logical I/O known as device label variables for the I/O to be used in the control application to be created later.

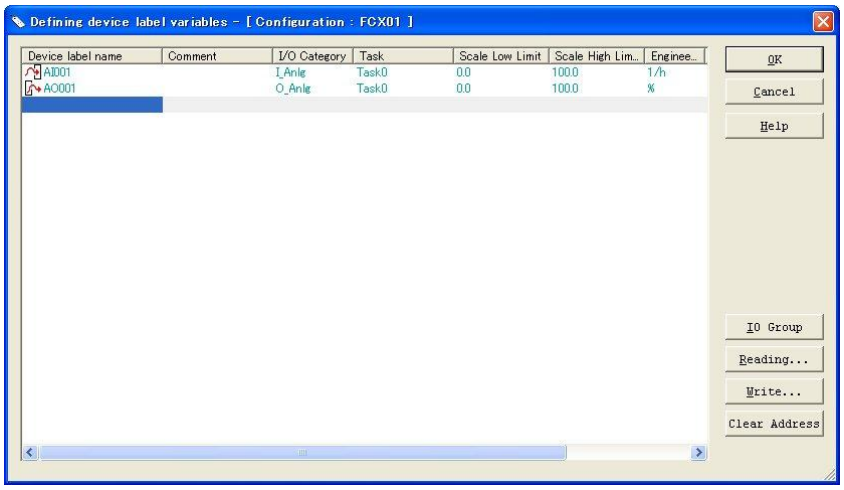
SEE ALSO

For more details on device label variables, see Section 1.4, "Connecting Control Applications and I/O Terminals."

(1) Double-click [DeviceLabelDefinition] on the project tree to display the [Defining device label variables] dialog.

(2) We shall define the following two device label variables in this exercise:

[Analog input]	[Analog output]
Device label name: AI001	Device label name: AO001
I/O Category: I_Anlg	I/O Category: O_Anlg
Task: Task0	Task: Task0
Scale Low Limit: 0.0	Scale Low Limit: 0.0
Scale High Limit: 100.0	Scale High Limit: 100.0
Engineering Unit: l/h	Engineering Unit: %



(3) After completing input, click [OK], and then click [Yes] on the displayed confirmation dialog to proceed.

4.1.6 Creating a Program

We will now build a control application using POU units as described in Chapter 1.
We will describe how to add POUs (program) to a project, and then create a control application using the POUs (program) using the FBD or LD language.

SEE ALSO

For more details on POUs, see Subsection 1.3.2, "Control Application Unit (POU)".

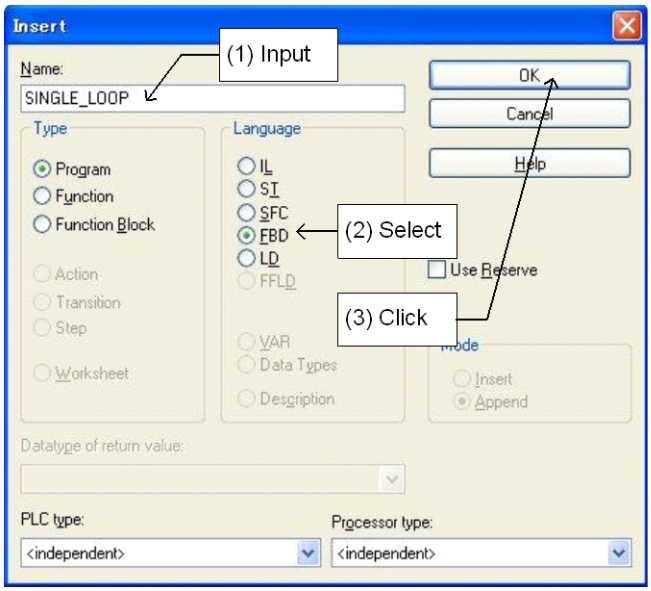
■ Adding a Program

We shall now add a new POU (program) to the project created in Subsection 4.1.2.

TIP: Default POU

The template project contains a program named "Main". This program can be used as is, but in this exercise, we shall describe how to add a new POU (program).

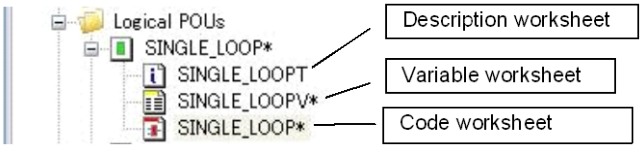
(1) Right-click [Logical POUs] on the project tree, and then select [Insert]-[Program] to display the Insert dialog box.
(2) In this exercise, enter "SINGLE_LOOP" in the [Name] field, select [FBD] for [Language], and then click [OK].



From the project tree, confirm that a POU (program) named "SINGLE_LOOP" has been created with three worksheets.

A control application can be built by programming the three worksheets.

Description worksheet	: Contains description for documentation.
Variable worksheet	: Contains local variable declarations.
Code worksheet	: Contains codes of program logic.



TIP: Deleting a Program

To delete a program, right-click the POU (program) to be deleted below [Logical POUs] on the project tree, and then select [Delete].

TIP: Asterisk ('*') Marks Beside Programs and Worksheets

An asterisk ('*') mark displayed beside a program or worksheet on the project tree indicates that the item has not yet been built.

■ Placing the PID Controller Function Block

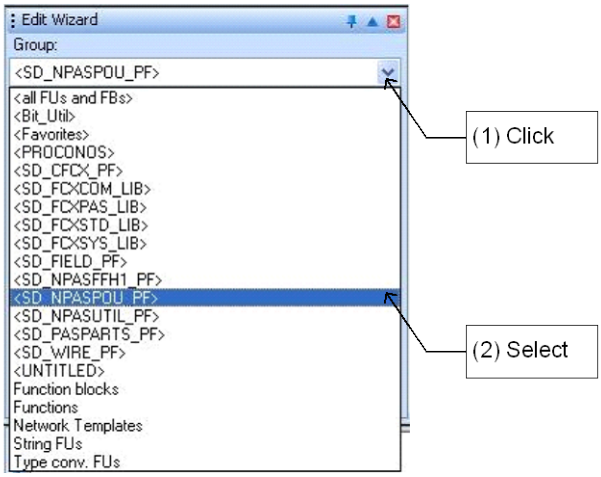
Let us now register the PID Controller function block on the POU (program) created in the previous subsection.

(1) Double-click the [SINGLE_LOOP] code worksheet on the project tree to open the code worksheet.

(2) Select "<SD_NPASPOU_PF>" (library) from the [Group] combo box of the [Edit Wizard].

TIP: Edit Wizard

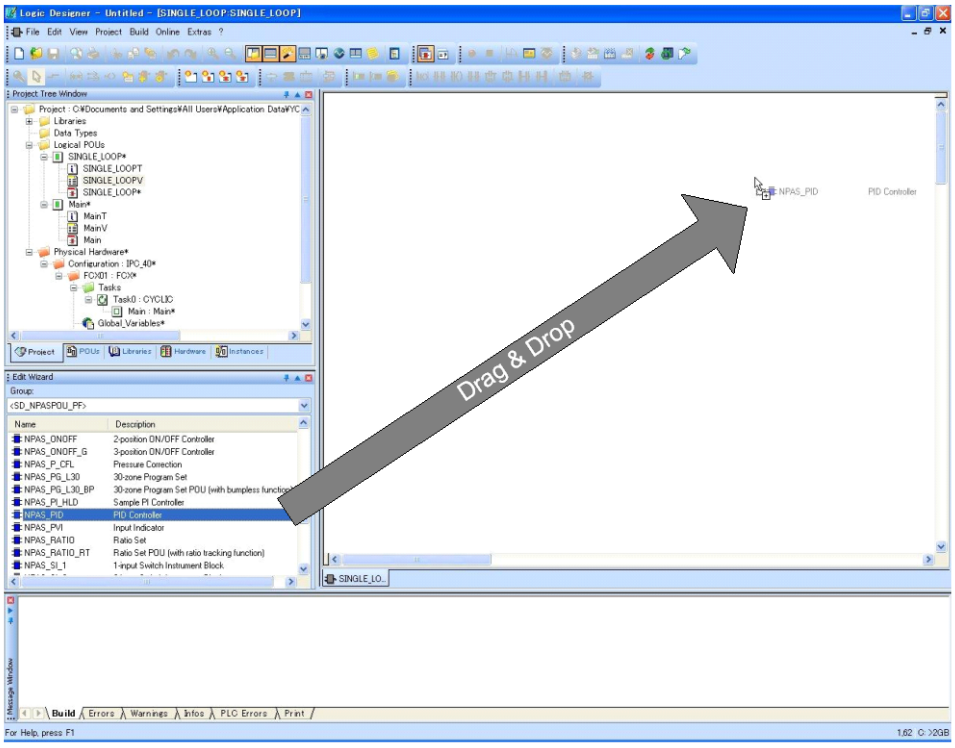
If the Edit Wizard is not displayed, you can display it by selecting [View]-[Edit Wizard] from the menu.



TIP: <SD_NPASPOU_PF>

<SD_NPASPOU_PF> is one of the libraries of POU function blocks provided in the PAS Portfolio. It stores various NPAS POUs including input/output data processing POUs, regulatory control POUs, arithmetic calculation POUs and sequence POUs.

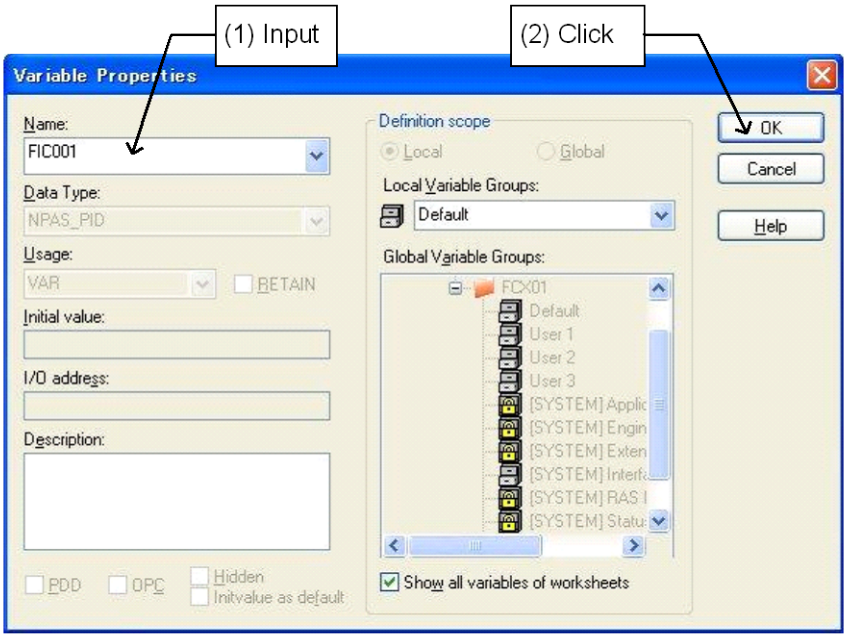
(3) Left-click [NPAS_PID (PID Controller)] in the [Group] combo box, hold the mouse button down and drag the object from the [Edit Wizard] into the desired target worksheet. Release the mouse button when the cursor is at the desired target position. The Variable Properties dialog is displayed.

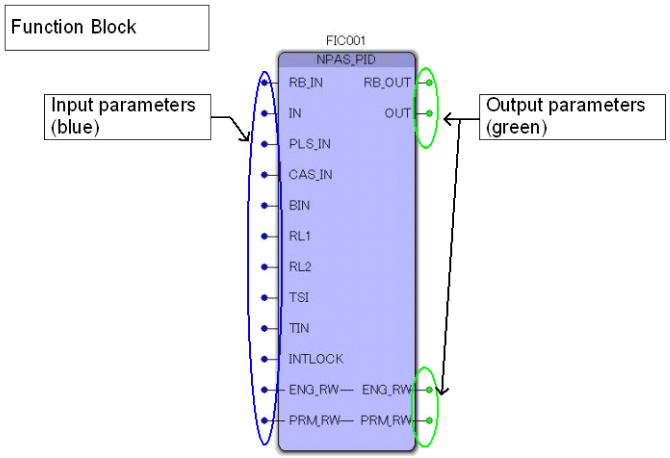


TIP: Definition of Function Blocks and Functions

Without using Drag & Drop function, Function Blocks and Functions are defined on the cord worksheet. In the above figure, by typing "NPAS_PID" directly on the position to be placed on the cord worksheet, the Variable Properties dialog appears. (Inline Edit Box Function)

(4) For this exercise, enter "FIC001" in the [Name] field and click [OK]. The function block is placed on the worksheet.





TIP: NPAS_PID

NPAS_PID (PID Controller POU) provides PID control and calculation processing, as well as alarm processing.

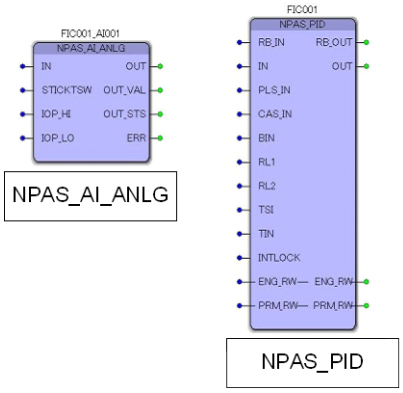
■ Placing the Standard Analog Input Function Block

Next, we shall register the Standard Analog Input function block, which performs IOP alarm detection and input signal conversion.

(1) Select "<SD_NPASPOU_PF>" (library) in the [Group] combo box of the [Edit Wizard].

(2) Left-click [NPAS_AI_ANLG (Standard Analog Input)] in the [Group] combo box, hold the mouse button down and drag the object from the [Edit Wizard] into the desired target worksheet. Release the mouse button when the cursor is at the desired target position. The Variable Properties dialog is displayed.

(3) For this exercise, enter "FIC001_AI001", and click [OK]. The function block is placed on the worksheet.



TIP: Digital Filter

NPAS_DGFLT (digital filter), a first-order lag filter is normally inserted between OUT of NPAS_AI_ANLG and IN of NPAS_PID to suppress noise. It is however omitted from this exercise for simplicity sake.

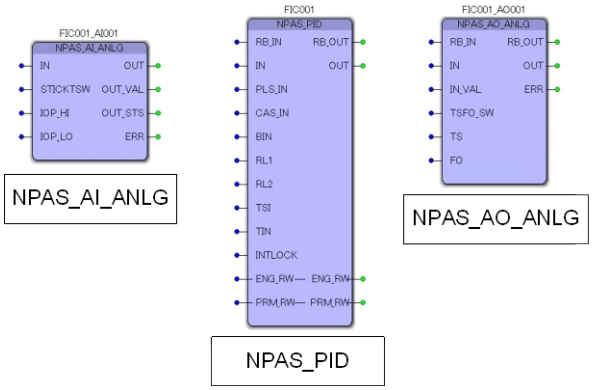
■ Placing the Standard Analog Output Function Block

Next, we shall register the Standard Analog Output function block, which performs output signal conversion and tight-shut/full-open processing.

(1) Select "<SD_NPASPOU_PF>" (library) in the [Group] combo box of the [Edit Wizard].

(2) Left-click [NPAS_AO_ANLG (Standard Analog Output)] in the [Group] combo box, hold the mouse button down and drag the object from the [Edit Wizard] into the desired target worksheet. Release the mouse button when the cursor is at the desired target position. The Variable Properties dialog is displayed.

(3) For this exercise, enter "FIC001_AO001" and click [OK]. The function block is placed on the worksheet.



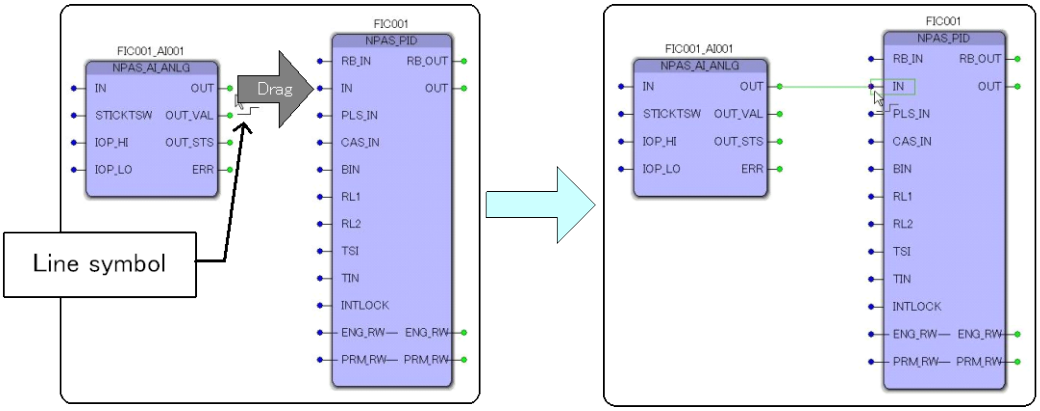
■ Connecting Function Blocks

Let us now connect the POUs placed on the work sheet.

(1) Move the cursor over the connection point [OUT] of [FIC001_AI001]. A line symbol is displayed to the cursor.

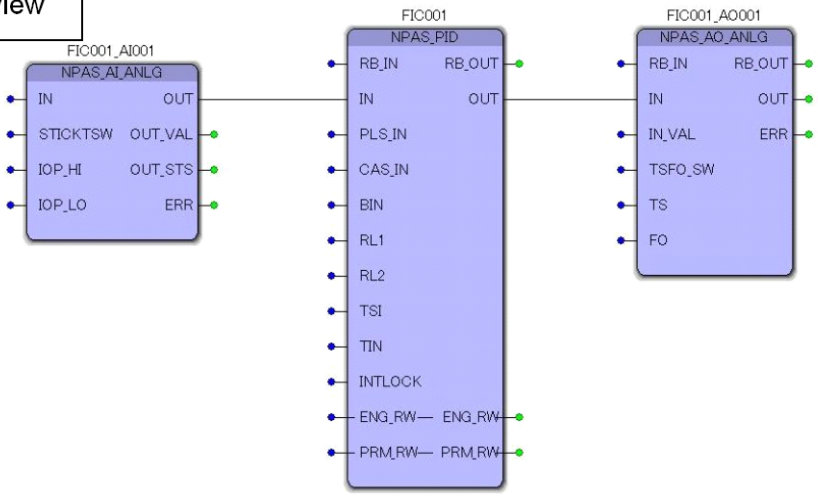
(2) Left-click, hold the mouse button down, and drag the mouse directly to the destination connection point [IN] of [FIC001_AO001].

Note: If the connection is permitted, the line becomes green. If the connection between the two objects is not permitted, the line is displayed in red. In this case, no connection can be established.



(3) Release the mouse button to create the connection. The editor automatically determines the path for the connection line (autorouting).
(4) Connect [OUT] of [FIC001] and [IN] of [FIC001_AO001] similarly.

Over all view



■ Connecting Device Label Variables

Let us now connect device label variable [AI001] to the Standard Analog Input function block.

(1) Double-click the blue circle for [IN] of [FIC001_AI001]. The Variable Properties dialog is displayed.
(2) To display device label variable "AI001" on the variable list, you must turn on [Show all variables of worksheet] checkbox.

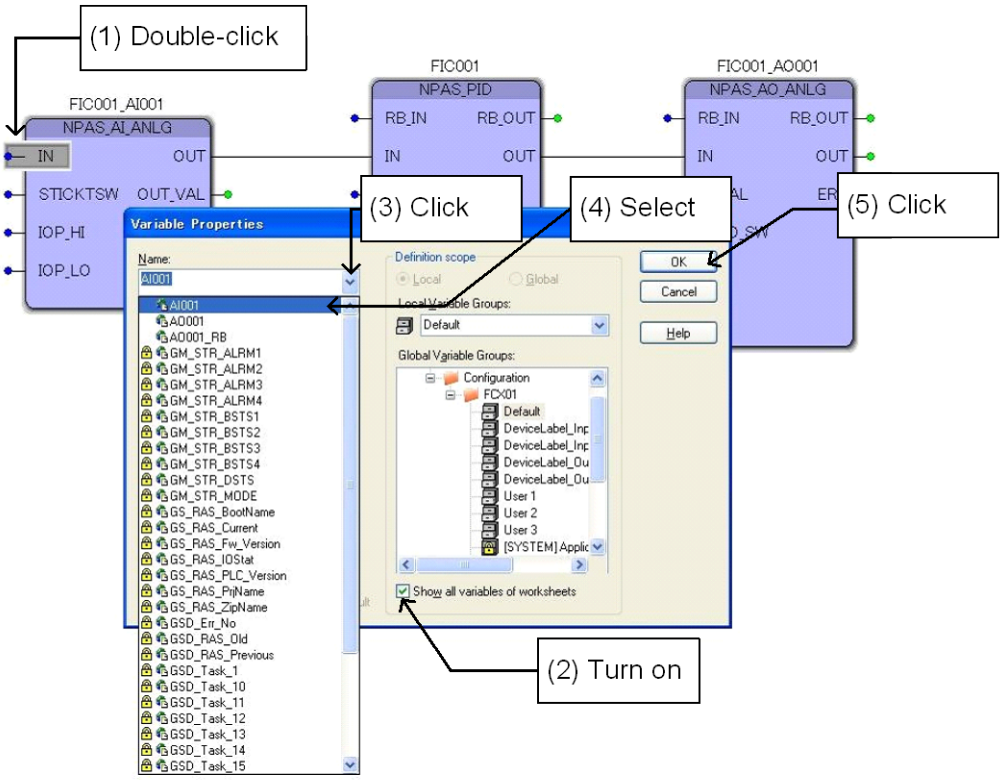
TIP: [Show all variables of worksheet] Checkbox

- When the [Show all variables of worksheet] checkbox is turned on, all global variables are displayed in the variable list regardless of the global variable group specified.
- To display the variable list of a specific global variable group, turn off the [Show all variables of worksheet] checkbox, select "Global" as "Definition scope" and specify the required global variable group (e.g. “DeviceLabel_Input_A” or “User1”).
- To display the variable list of a local variable group only, turn off the [Show all variables of worksheet] checkbox and select Local as Definition Scope.

SEE ALSO

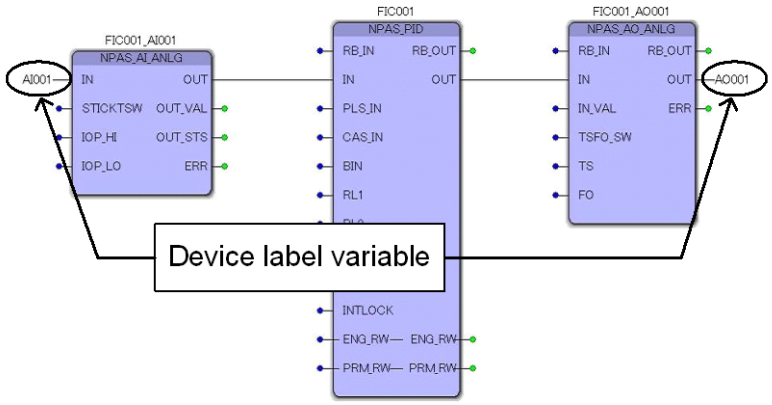
For more details on local variables and global variables, see Subsection 1.3.3, "Local Variables and Global Variables".

(3) Click the [Name] combo box, and select [AI001] from the displayed list of global variables of the resource specified in step (2). Click [OK].



(4) Next, we will connect device label variable "AO001" as output of "FIC001" similarly. Double-click the green circle for [OUT] of [FIC001_AO001]. The Variable Properties dialog is displayed.

(5) Since the scope has been selected in step (2), we leave it unchanged. Select [AO001], and click [OK].



TIP: Specify Variables

Without using "Variable Properties" dialog, declared variables can be specified.
By typing the first character (or several characters) on the connection point, listed name is automatically completed.

■ Connecting Read Back Variables

To avoid sudden changes in the output signal during switching o operation according to the state of the output destination in regulatory control, knowledge of the state of the function block or output module of the output destination is required.

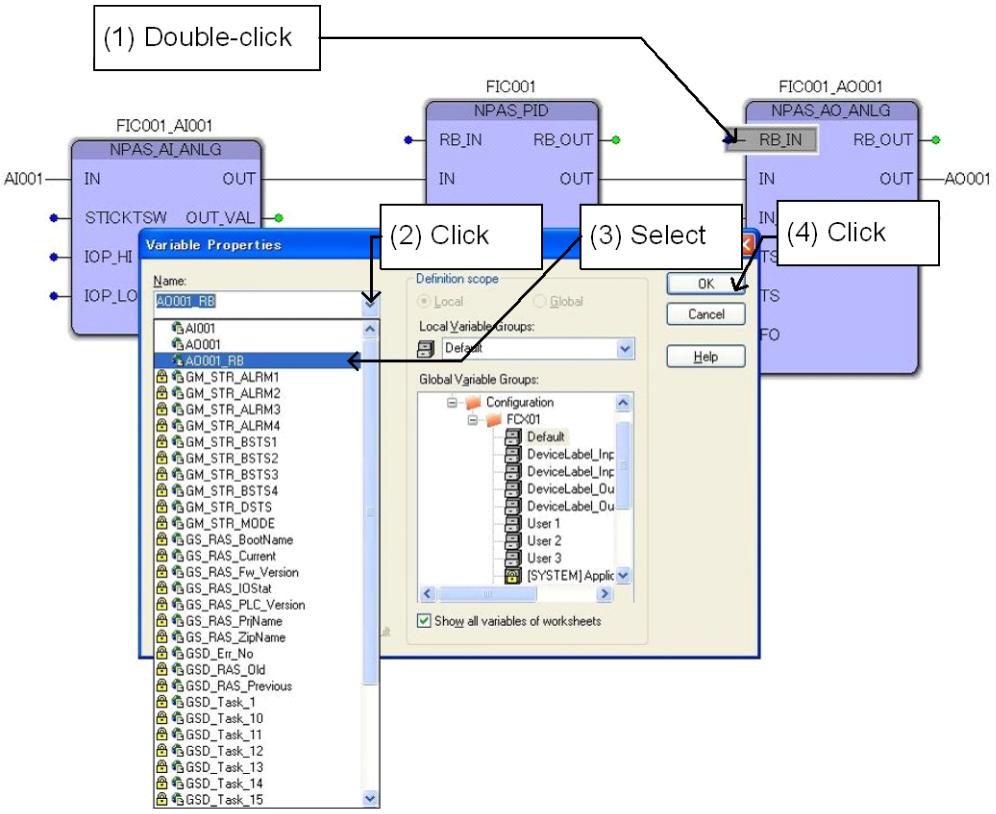
Nevertheless, the IEC standard only allows unidirectional transfer of data among connected function blocks from an output parameter to an input parameter. To perform bi-directional data exchange, read back connection, which goes from secondary to primary is required.

We shall define a read back connection that will provide feedback on the state of the output destination.

TIP: Necessity of Read Back Variables

In the current loop system, OOP information, output range and fallback detection information from the output module will be read back to the analog output function block and the primary PID controller function block.
Such read back will enable automatic MODE switching at OOP, calculation of output value according to the output range and avoid sudden output changes during fallback recovery.

(1) Let us define a read back for the state of output destination "FIC001_AO001". Double-click the blue circle for [RB_IN] of [FIC001_AO001]. The Variable Properties dialog is displayed.
(2) Click the [Name] combo box, and select [AO001_RB] from the displayed list of global variables. Click [OK].

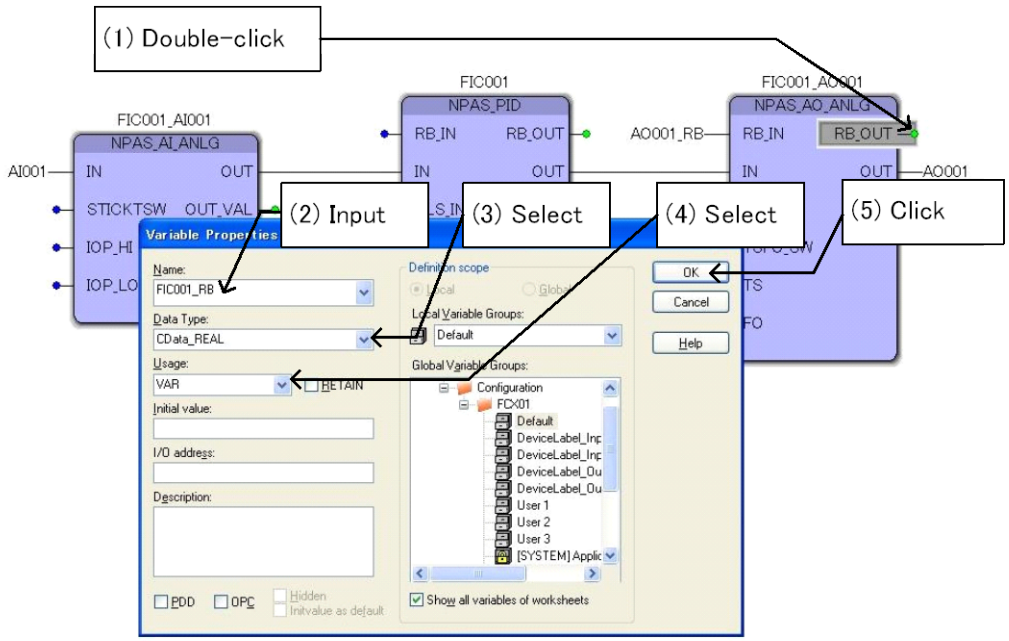


TIP: Parameter for Read Back

Of the input/output parameters of the function block, "RB_IN" and "RB_OUT" are used for read back. (RB: Read Back)

Up to now, we have defined read back of the state of device label variable "AO001" to "FIC001_AO001". Let us proceed to define read back of the state of "FIC001_AO001" to "FIC001".

(3) Double-click [RB_OUT] of [FIC001_AO001]. The Variable Properties dialog is displayed.
(4) Enter [FIC001_RB] for [Name], select [CData_REAL] for [Data Type], select [VAR] for [Usage], and click [OK]. (The local variable for read back is added.)



TIP: Data Type (CData_REAL)

The CData_REAL data type is a structure data type that stores scale limits or engineering unit for the PAS Portfolio, as well as current value and other information.

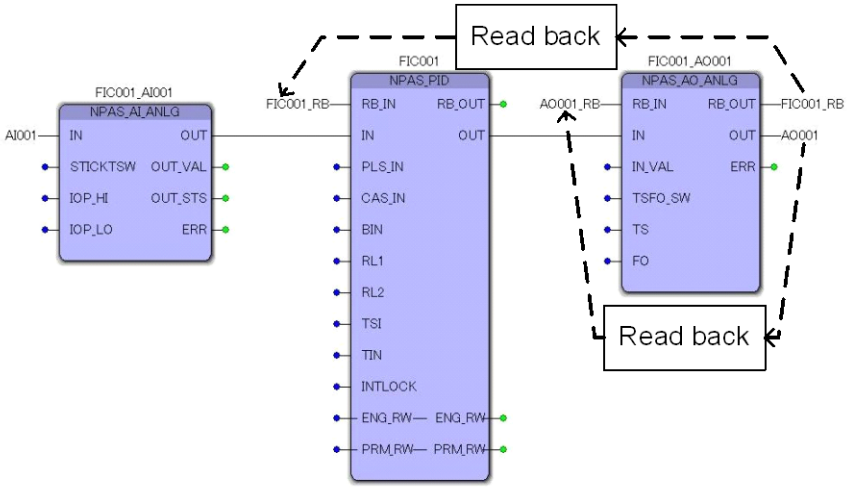
TIP: Usage

Specify the variable declaration keyword for "Usage". When "Show all variables of worksheets" is selected, "VAR" or "VAR_GLOBAL" can be selected.
By selecting either "VAR" or "VAR_GLOBAL in the "Usage" combo box, the usage of the variables is defined as follows:

- VAR : Local variables declaration
- VAR_GLORBAL : Global variables declaration

(5) Next, double-click [RB_IN] of [FIC001]. The Variable Properties dialog is displayed.
(6) Select [FIC001_RB] in the [Name] combo box, and click [OK].

The figure below shows the read back definitions that have been completed.



TIP: Warning of Backward Connection Line

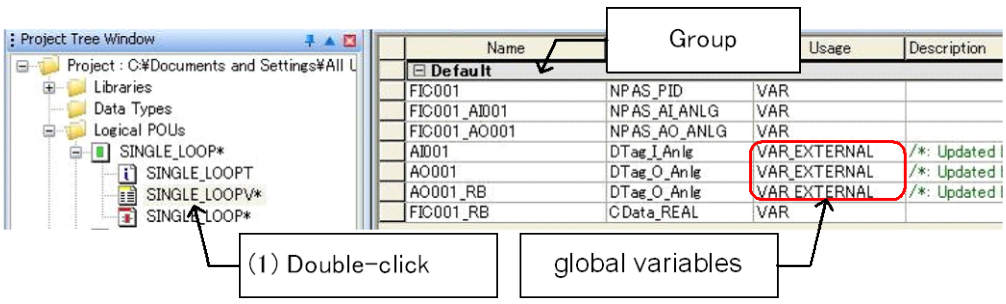
In this exercise, read back is achieved by connecting the same variable to both "RB_OUT" of "FIC001_AO001" and "RB_IN" of "FIC001".
If "RB_OUT" and "RB_IN" are connected directly using a connection line instead of indirectly via a variable, execution order of control applications may be changed because it depends on the POU positions.
And a warning will be generated when the project is complied because the connection line runs in the reverse direction (from right to left).

■ Checking Variables

Let us now check the device label variables that we have placed as global variables, as well as FIC001_RB, which we have registered as a local variable.

(1) Double-click variable worksheet [SINGLE_LOOPV] on the project tree to open the variable worksheet.

Variables displayed with type [VAR_EXTERNAL] on the variable worksheet are global variables.



After finished checking, switch from the variable worksheet to the code worksheet.

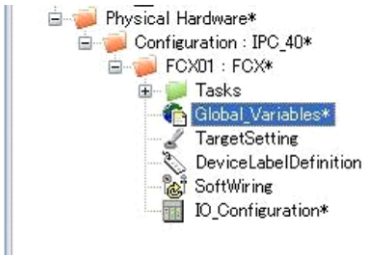
TIP: Adding and Deleting Variables

Besides adding variables from the code worksheet using the Variable Properties dialog, you can also add variables from the variable worksheet.
You can also delete variables or add variable groups from the variable worksheet.
(Deleting a variable from a code worksheet does not delete it from the variable worksheet. To completely delete the variable, you should also manually delete it from the variable worksheet.)

TIP: Grouping Variables

When moving a worksheet to another program, variables used on the worksheet must also be moved. To simplify the operation, we recommend that you group variables by worksheets. (This is easier than selecting and moving the variables one at a time.)

You can display the list of global variables by selecting [Global Variables], which is several levels under [Physical Hardware] on the project tree. The list contains device label variables, as well as system global variables.



SEE ALSO

For more details on the relationship between global variable declarations on the variable worksheet of a POU (program) and the global variable definitions on the “Global_Variables” global variable worksheet, see Appendix 2, “Creating and Deleting Global Variables.”

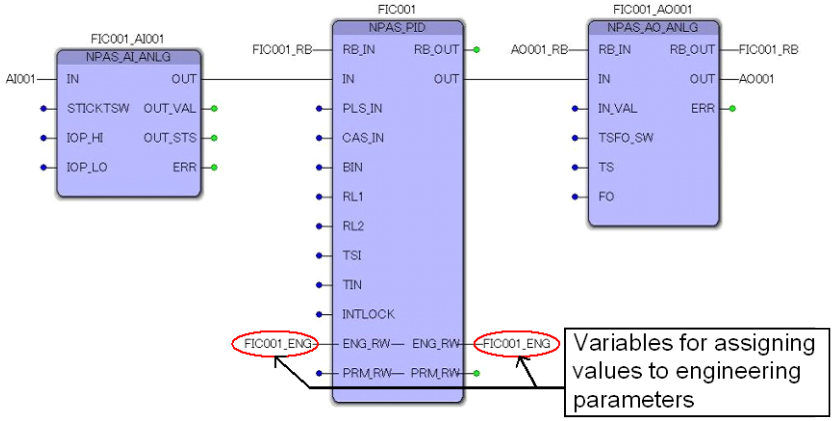
4.1.7 Saving a Project

Let us now save the work that we have done so far.

(1) Select [File]-[Save Project As/Zip Project As] from the menu. A dialog is displayed.
(2) Enter "training" for the project name, and then click [Save].

4.1.8 Assigning Values to Engineering Parameters

Engineering parameters can be used to set up a comment, high limit hysteresis, low limit hysteresis and other information for a NPAS POU when creating a control application.
All engineering parameters of NPAS POUs are pre-defined with default values for convenience. These default values can be changed as required using a user-defined variable referred to as the "Variable for assigning values to engineering parameters".



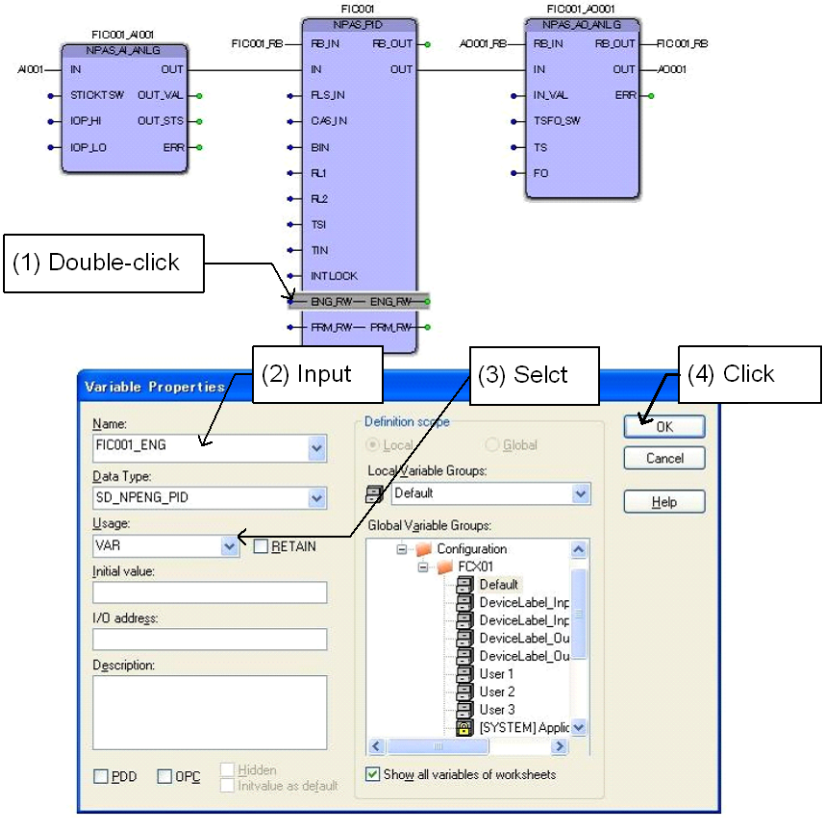
■ Defining Variables for Assigning Values to Engineering Parameters

As its name implies, a "variable for assigning values to engineering parameters" is used to assign values to engineering parameters.

To change the default values of engineering parameters, we need to define a "variable for assigning values to engineering parameters", and then change the value of this variable.

Let us first define a variable for assigning values to engineering parameters.

(1) Double-click the "SINGLE_LOOP" code worksheet on the project tree to open the code worksheet.
(2) Double-click [ENG_RW] of [FIC001]. The Variable Properties dialog is displayed.
(3) Enter [FIC001_ENG] in the [Name] field, select [VAR] for [Usage] and click [OK]. (The [Data Type] is automatically set to [SD_NPENG_PID].)



TIP: Local and Global Scope

The Local or Global selection for [Scope] on the Variable Properties dialog follows the value for the most recently defined variable. When creating a new variable, you should always check that the global/local selection is appropriate before defining the variable.

TIP: ENG_RW

"ENG_RW" is known as an I/O parameter, whose allocated variable is used for both input and output. By allocating a variable for assigning values to engineering parameters ("FIC001_ENG") to the input terminal of an I/O parameter as shown above, the same variable is also connected automatically to the output terminal.

■ Assigning Values to Engineering Parameters

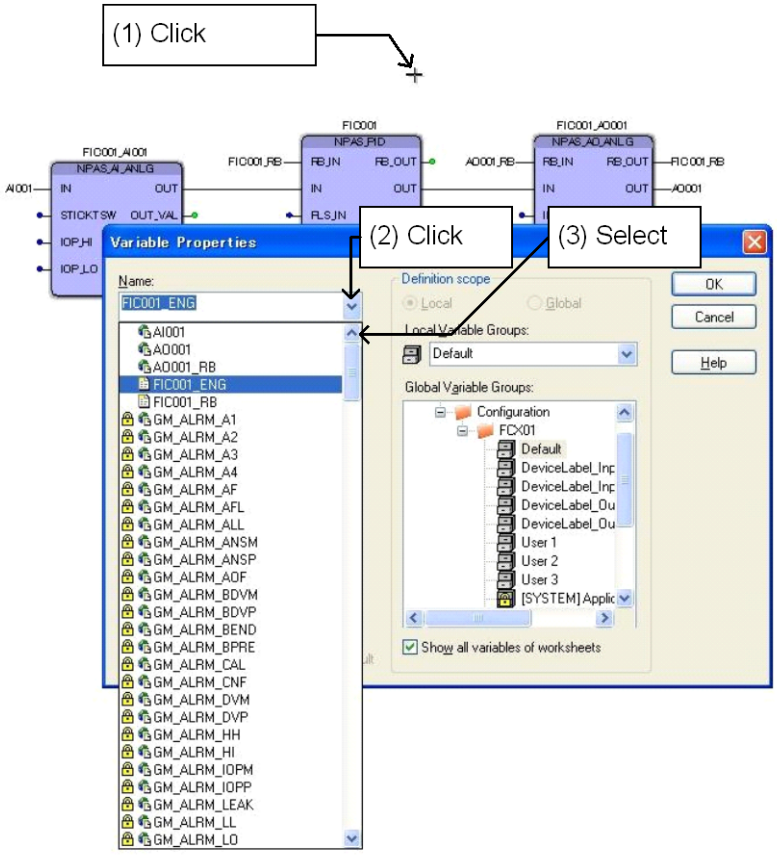
Let us assign a value to the Comment of engineering parameter.

To assign a value to an engineering parameter, you simply enter the data to the variable for assigning values to engineering parameters.

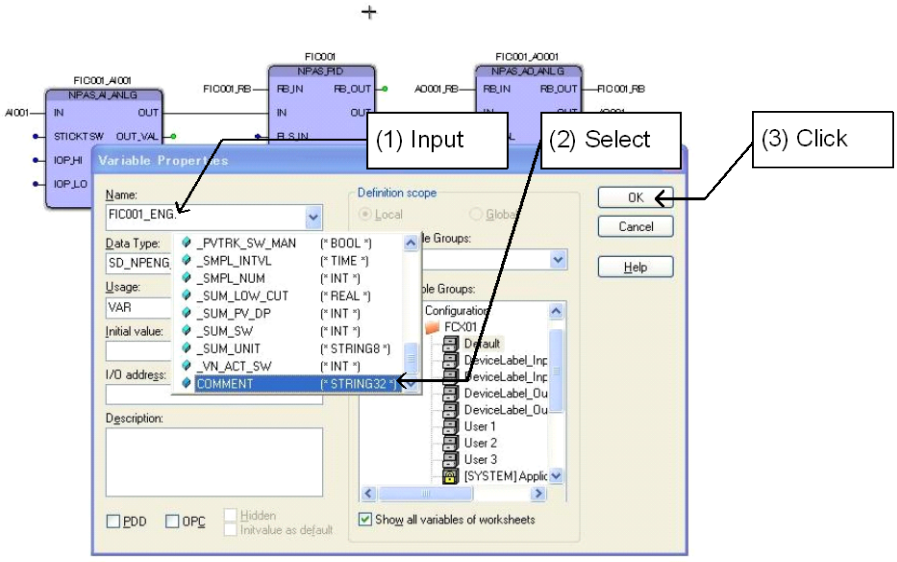
(1) Click a location above [FIC001] to specify the position for inserting the variable for assigning values to engineering parameters.

(2) Click the right mouse button, and select [Variables] from the displayed menu. The Variable Properties dialog is displayed.

(3) Click the [Name] combo box, and then select [FIC001_ENG] from the displayed local variable list.



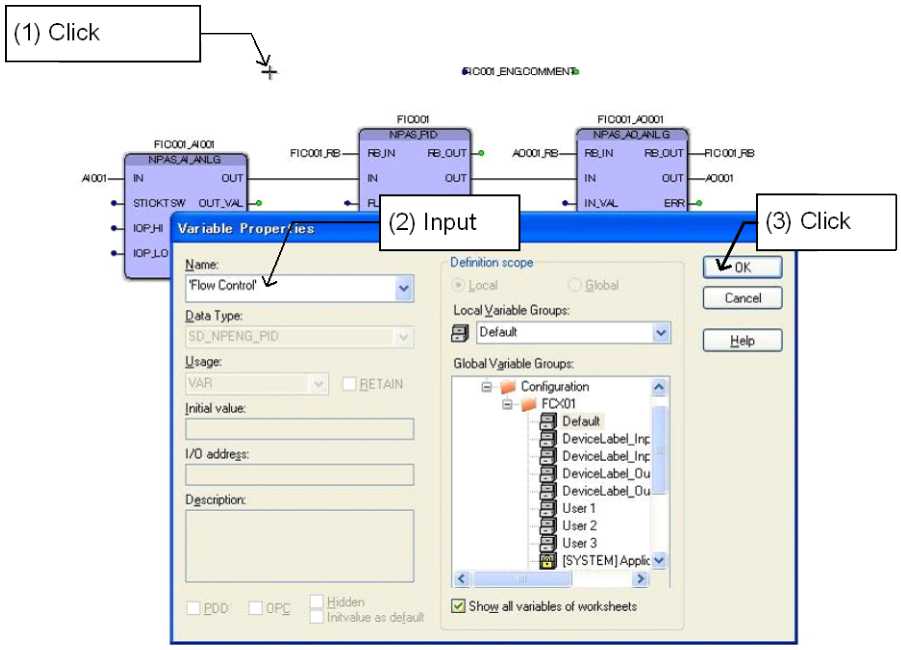
(4) Enter a period ('.') character after "FIC001_ENG", and select [COMMENT] from the displayed menu of the variable for assigning values to engineering parameters. Click [OK].



(5) Next, click a location on the worksheet to the left of [FIC001_ENG.COMMENT] to specify the location for placing the value to be assigned to the user variable.

(6) Click the right mouse button, and select [Variables] from the displayed menu. The Variable Properties dialog is displayed.

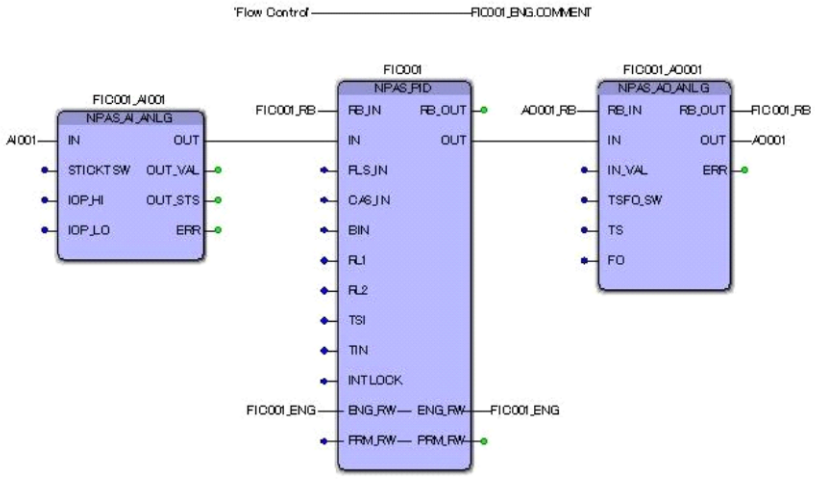
(7) For this exercise, enter "'Flow Control'", and click [OK]. Note that the entered string must be enclosed within single quote (') characters.



TIP: Assigning Comments

When assigning a value to a comment, a string value is entered directly in the [Name] field instead of a variable name by enclosing the string within single quote (') characters.
A comment may be up to 32 characters long.

(8) Finally, connect 'Flow Control' with FIC001_ENG.COMMENT.



This completes the setup of engineering parameters.

TIP: Engineering Parameters

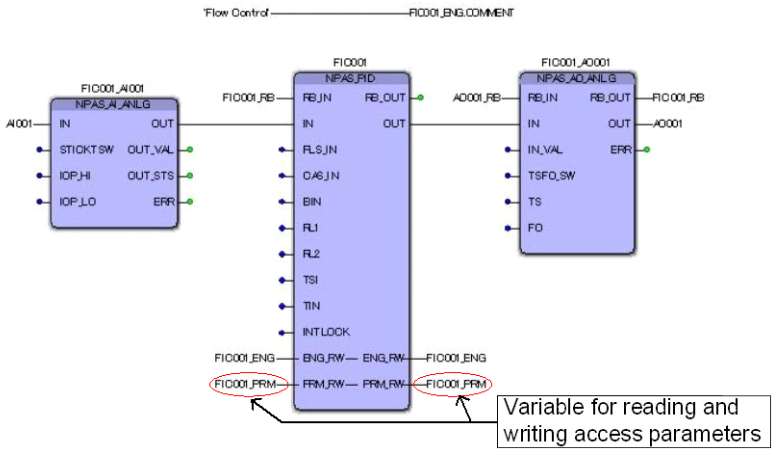
Default values are applied to the unassigned engineering parameters of NPAS POUs.

SEE ALSO

For a list of engineering parameters of an NPAS POU and their respective default values, see the online help for the corresponding POU.

4.1.9 Defining Variables for Reading and Writing Access Parameters

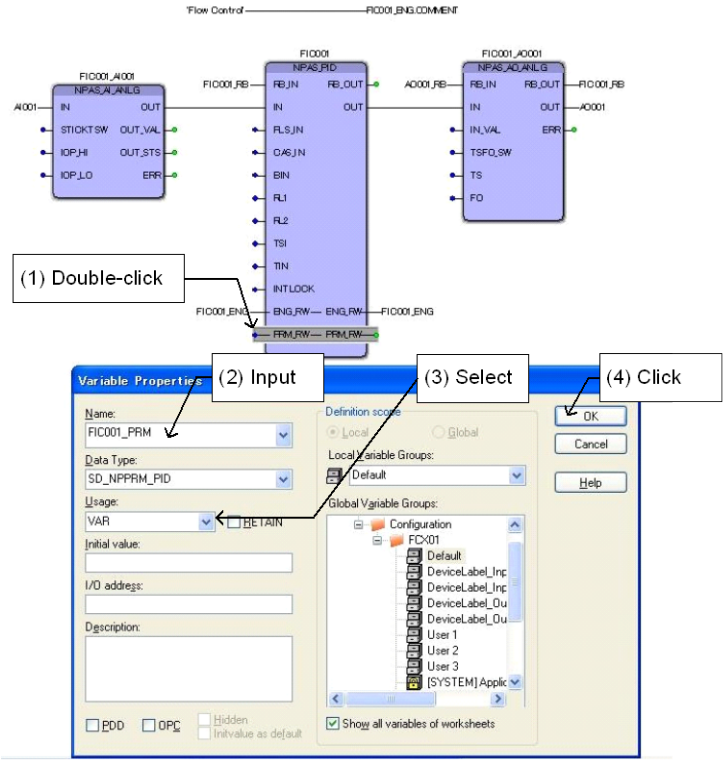
Access parameters can be used to display or set PV, SV, MV, MODE, PH, PL, P, I, D and other parameters during operation. To read and write these access parameters from other control applications, a user-defined variable referred to as the "variable for reading and writing access parameters" is used.



We shall now define a variable for reading and writing access parameters.

(1) Double-click the blue circle for [PRM_RW] of [FIC001]. The Variable Properties dialog is displayed.

(2) For this exercise, enter [FIC001_PRM] in the [Name] field, select [VAR] for [Usage] and click [OK]. (The [Data Type] is automatically set to [SD_NPPRM_PID].)



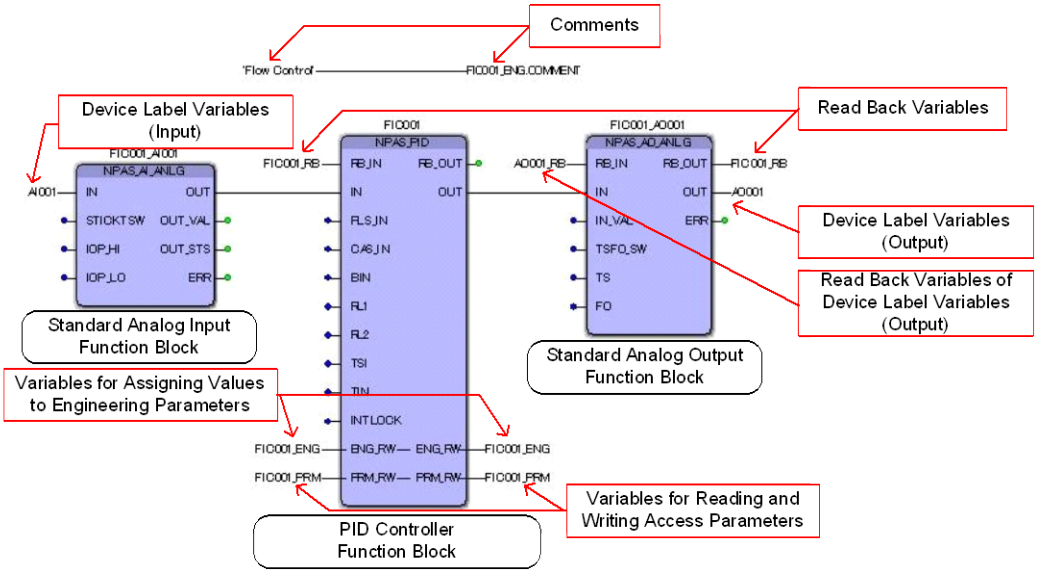
TIP: Defining Variables for Reading and Writing Access Parameters

We have defined a variable for reading and writing access parameters. For more details on reading and writing access parameters, see Section 4.2, "Creating Control Applications (Sequence)."

This completes the setup of a variable for reading and writing access parameters.

4.1.10 Checking and Saving a Created Control Loop

We have so far finished creating a control loop.
The last thing we need to do is to check the control loop we have created.



TIP: Creating Control Loops

Read back variables must be set up to ensure correct operation of a control loop. Even though a control loop without read back variables may compile without errors, the actual control will not work properly.
In addition, the variable for assigning values to engineering parameters and the variable for reading and writing access parameters must be defined even if they are not to be used. Omitting their definitions will result in compilation errors.

TIP: Range Setup for Function Blocks

In NPAS POUs, the range of data values of a function block are automatically determined from the data range of input data or read back variable. Hence, for NPAS POUs, no range setup is required for function blocks.

Let us save the work that we have done so far.
Select [File]-[Save] from the menu.

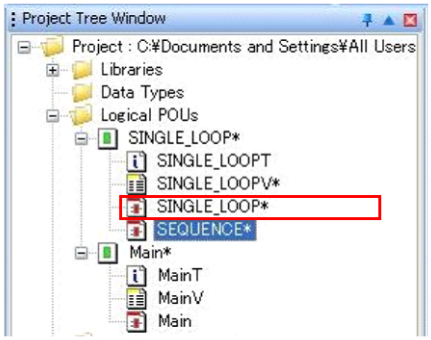
4.2 Hands-on: Creating A Control Application (Sequence)

In Section 4.1, we have described how to create a control application using FBD, starting from creating a project and a program to defining a control loop. In this section, we shall describe the procedure for building a sequence application.

4.2.1 Adding a Worksheet

We can add the sequence application to be created later to the "SINGLE_LOOP" code worksheet constructed in Section 4.1, but in this exercise, let us add a new code worksheet to the "SINGLE_LOOP" program, and build our control application on the worksheet.

(1) Right-click "SINGLE_LOOP" code worksheet (not the program) on the project tree, and then select [Insert]-[Code Worksheet] to display the Insert dialog box.



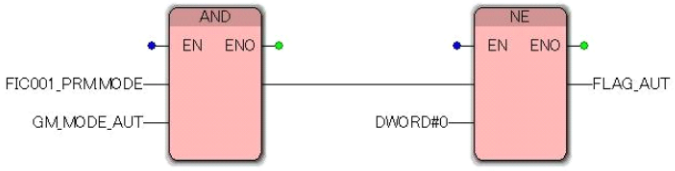
(2) Enter "SEQUENCE" in the [Name] field for this exercise, and click [OK].

TIP: Language for a New Worksheet

When adding a worksheet, the language of the new worksheet is determined by the language specified for the POU (program). FBD and LD can be intermixed, as well as be coded in SFC actions.

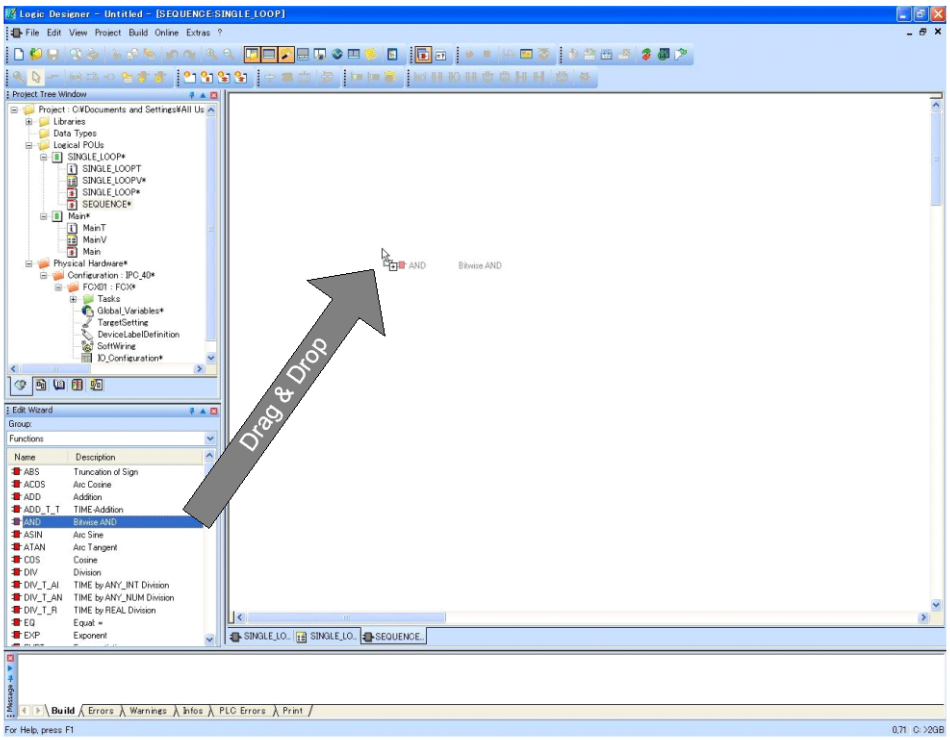
4.2.2 Reading Access Parameters

Let us build the upper part of the sequence application described in Chapter 2, which reads access parameters (see figure below).



■ Placing the AND Function

(1) Double-click the "SEQUENCE" code worksheet on the project tree to open the code worksheet.
(2) Select [Functions] from the [Group] combo box of the Edit Wizard.
(3) Left-click [AND] in the [Group] combo box, hold the mouse button down and drag the object from the [Edit Wizard] into the desired target worksheet. Release the mouse button when the cursor is at the desired target position. The AND function is placed.

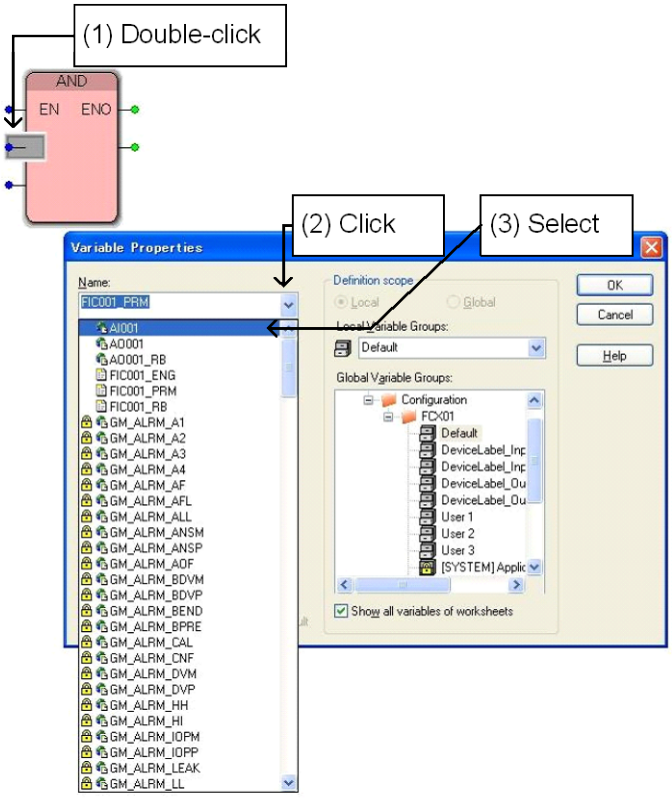


■ Setting Up Parameters of AND Function

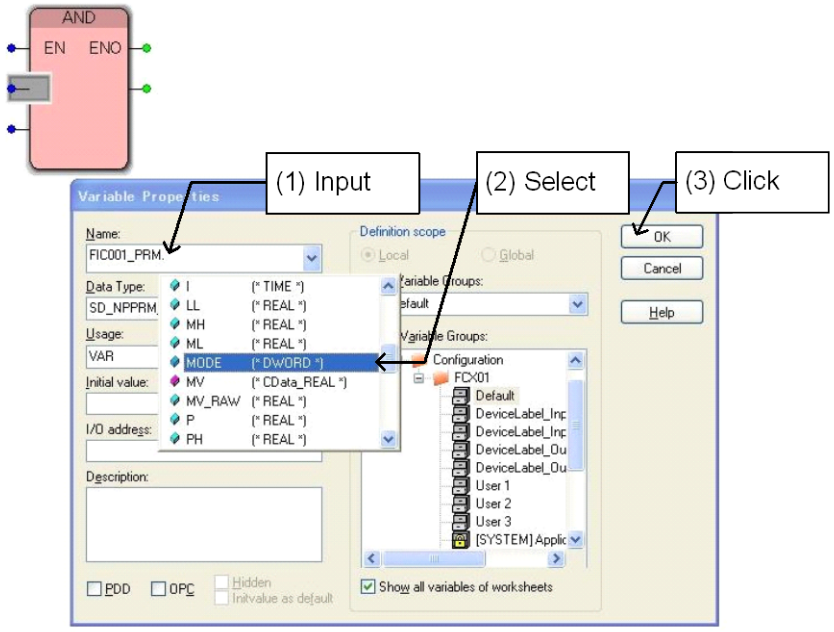
Next, we shall set up the parameters of the AND function placed earlier.

(1) Double-click one of the input parameters of the AND function. The Variable Properties dialog is displayed.

(2) Click the [Name] combo box, and then select [FIC001_PRM], which is the variable for reading and writing access parameters, from the displayed local variable list.

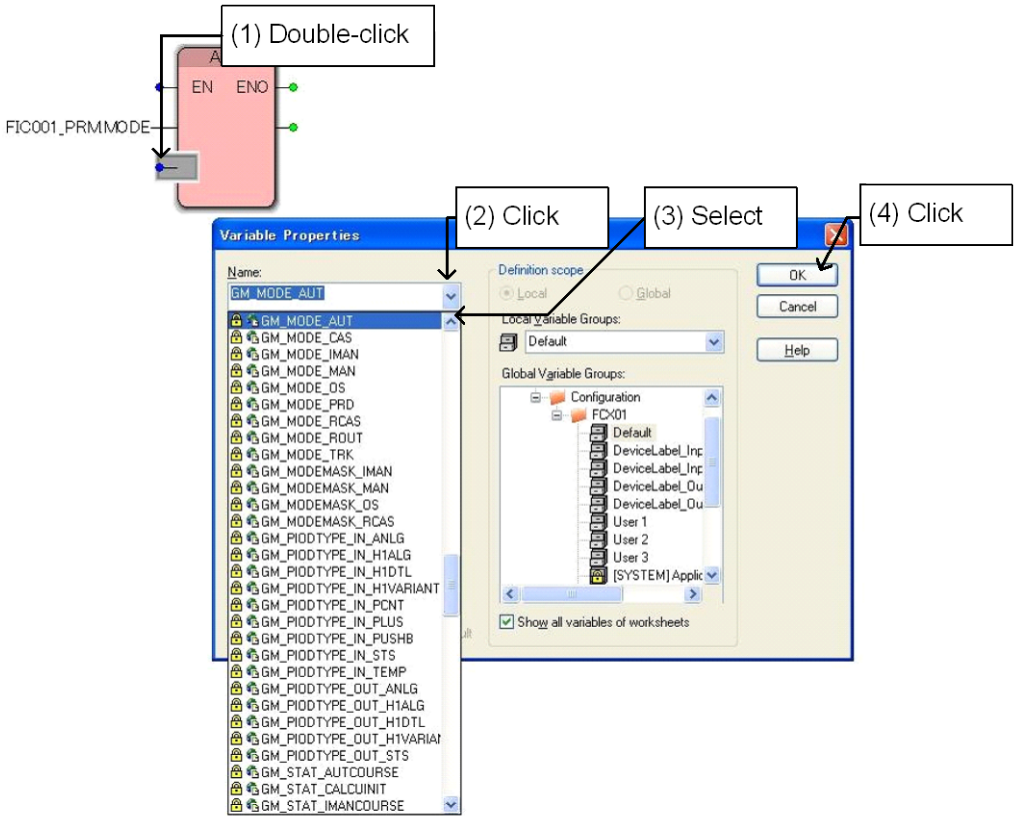


(3) Enter a period ('.') character after "FIC001_PRM", and select [MODE] from the displayed member list of the variable for reading and writing access parameters. Click [OK].



(4)Double-click the other input parameter. The Variable Properties dialog is displayed.

(5) Click the [Name] combo box, and select [GM_MODE_AUT] from the displayed list of global variables. Click [OK].



TIP: GM_MODE_AUT

GM_MODE_AUT is a DWORD-type global variable, which stores a system-defined constant (hexadecimal value 00400000) representing the AUT block mode of NPAS POUs.

This completes the assigning of parameters to the AND function.

With this setup, the AND function performs a logical AND of FIC001_PRM.MODE (DWORD value of the mode of FIC001) and global variable GM_MODE_AUT (DWORD value representing AUT mode), and outputs the result as a DWORD value from the output terminal of the AND function.

TIP: EN Terminal of Function

The EN terminal of a function can be used to enable or disable the function by turning the terminal on or off respectively. In our example, nothing is linked to the EN terminal so its input is always TRUE and the function is always enabled.

■ Placing the NE Function

Next, we shall place the NE (Not Equal) function to the right of the AND function.

(1) Select [Functions] in the [Group] combo box of the [Edit Wizard].

(2) Left-click [NE] in [Group] combo box, hold the mouse button down and drag the object from the [Edit Wizard] into the desired target worksheet. Release the mouse button when the cursor is at the desired target position. The NE function is placed.

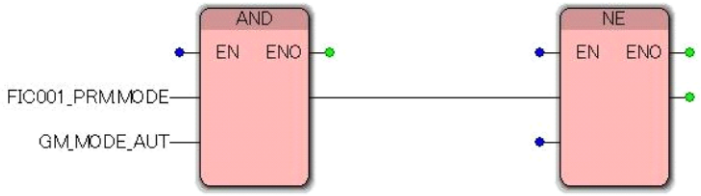
■ Setting Up Parameters of NE Function

We shall link to the NE function, and set up its parameters.

(1) Move the cursor over the connection point of the output parameter of the AND function. A line symbol is displayed to the cursor.

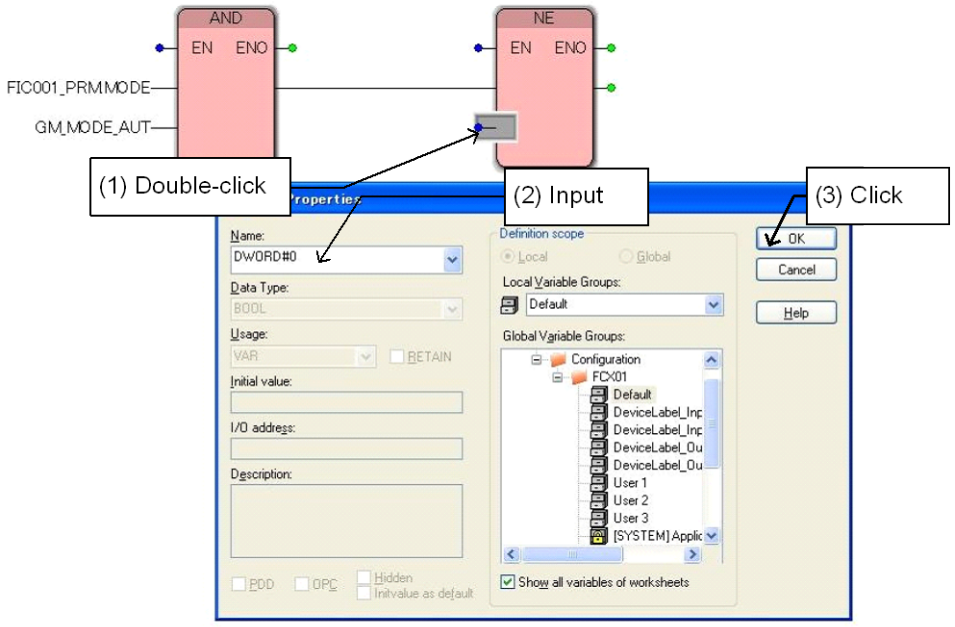
(2) Left-click, hold the mouse button down, and drag the mouse directly to the destination connection point of the input of the NE function.

Note: If the connection is permitted, the line becomes green. If the connection between the two objects is not permitted, the line is displayed in red. In this case, no connection can be established.



(3) Release the mouse button to create the connection. The editor automatically determines the path for the connection line (autorouting).

(4) Double-click the other input parameter of the NE function. The Variable Properties dialog is displayed.
(5) Enter "DWORD#0" in the [Name] field, and click [OK].

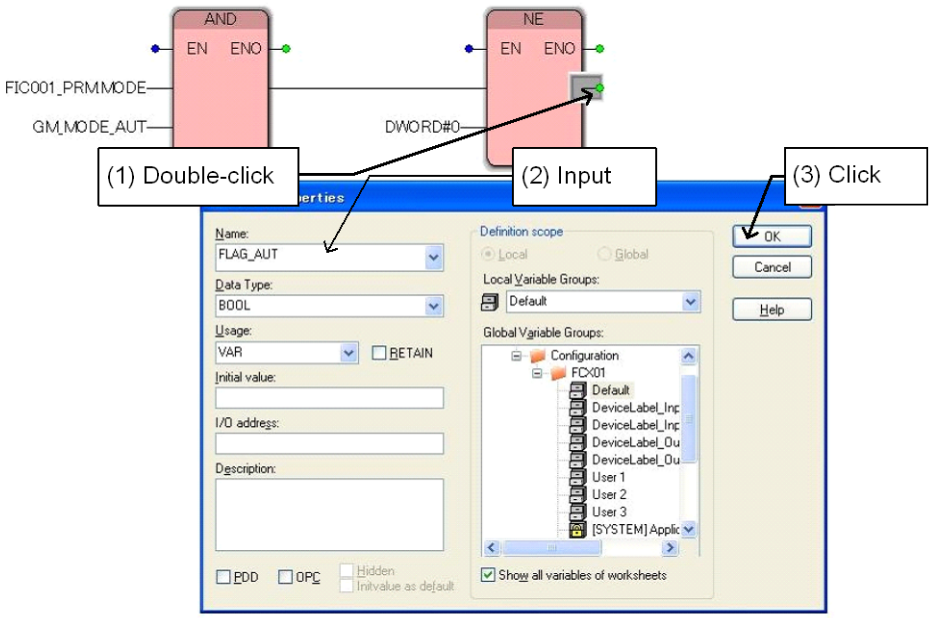


TIP: DWORD#0

Here, we have entered a numerical value "DWORD#0" directly in the [Name] field of the Variable Properties dialog instead of entering a variable name. "DWORD#0" denotes a zero value of DWORD data type (with hexadecimal value 00000000).

(6)Next, we will feed the output of the NE function to a new BOOL-type variable named "FLAG_AUT".
Double-click the output parameter of the NE function. The Variable Properties dialog is displayed.

(7) Enter [FLAG_AUT] in the [Name] field, and click [OK].



This completes the assignment of parameters to the NE function.

With this setup, the NE function determines whether the output of the AND function is not equal to zero, and outputs the result to FLAG_AUT, a BOOL type variable.

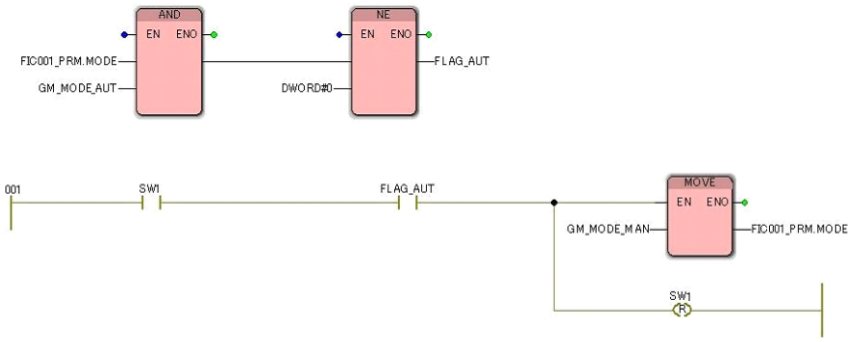
In summary, we have completed a logical operation that sets the BOOL type variable named "FLAG_AUT" to TRUE when FIC001 is in AUT mode, but sets the variable to FALSE when FIC001 is not in AUT mode.

TIP: Reading Block Mode

The sequence we have constructed compares the block mode of FIC001 to the global variable GM_MODE_AUT by combining an AND function with a NE function. One EQ (Equal) function is usually sufficient for comparing against a single DWORD value. In the case of the block mode of a NPAS POU, it can assume more than two modes at the same time (e.g. IMAN and AUT), so we need to first perform a logical AND operation using an AND function, and then check whether the result is non-zero to determine whether the NPAS POU is in AUT mode.

4.2.3 Writing Access Parameters

Next, let us build the lower part of the sequence application (see figure below) that is described in Chapter 2. It writes to an access parameter.



■ Placing Contacts

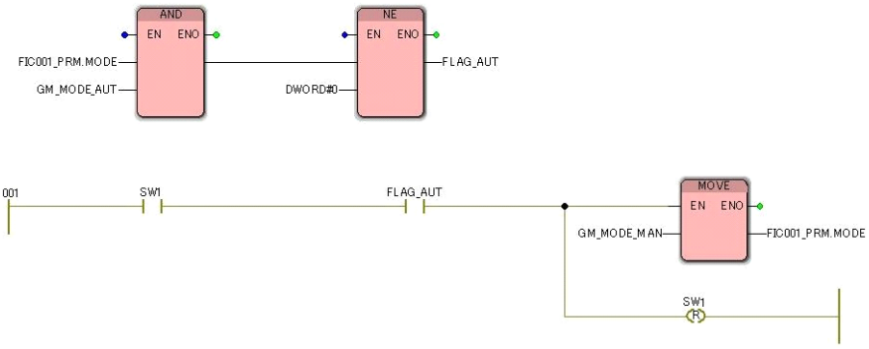
(1) Click a location below the AND function on the worksheet to specify the position for placing a contact.
(2) Click [Contact right] on the toolbar. A contact is placed on the worksheet.

Contact right

(3) Double-click the contact. The Contact/Coil Properties dialog is displayed.
(4) For this exercise, enter "SW1" in the [Name] field, and click [OK].
(5) Next, with the contact selected, click [Left power rail] on the toolbar to place a power rail to the left of the contact.

Left power rail

(6) Select contact "SW1" and click [Contact right] on the toolbar. A contact is placed on the worksheet.
(7) Double-click the contact. The Contact/Coil Properties dialog is displayed.
(8) Click the [Name] combo box, and select [FLAG_AUT] created earlier from the displayed list of local variables. Click [OK].



■ Placing the MOVE Function

Next, we shall place a MOVE function.

(1) Select [Functions] in the [Group] combo box of the [Edit Wizard].
(2) Left-click [MOVE] in the [Group] combo box, hold the mouse button down and drag the object from the [Edit Wizard] into the desired target worksheet. Release the mouse button when the cursor is at the desired target position. The MOVE function is placed.

■ Setting Up Parameters of MOVE Function

We shall link to the MOVE function, and set up its parameters.

(1) Move the cursor over the connection point of the right terminal of the [FLAG_AUT]. A line symbol is displayed to the cursor.
(2) Left-click, hold the mouse button down, and drag the mouse directly to the destination connection point of the [EN] input parameter of the MOVE function.

Note: If the connection is permitted, the line becomes green. If the connection between the two objects is not permitted, the line is displayed in red. In this case, no connection can be established.

(3) Release the mouse button to create the connection. The editor automatically determines the path for the connection line (autorouting).

(4) Double-click the other input parameter of the MOVE function. The Variable Properties dialog is displayed.
(5) Click the [Name] combo box, and select [GM_MODE_MAN] from the displayed list of global variables. Click [OK].
(6) Double-click the output parameter of the MOVE function. The Variable Properties dialog is displayed.
(7) Click the [Name] combo box, and then select [FIC001_PRM], which is the variable for reading and writing access parameters, from the displayed local variable list.
(8) Enter a period ('.') character after "FIC001_PRM", and select [MODE] from the displayed member list of the variable for reading and writing access parameters. Click [OK].

This completes assignment of parameters for the MOVE function.

We have so far created a sequence circuit that switches the FIC001 to MAN mode when both the FLAG_AUT contact (flag indicating whether FIC001 is in AUT mode) and SW1 contact (switch for activating this sequence) are TRUE.

TIP: Writing to Access Parameters

As shown above, a MOVE function can be used to write to an access parameter. To do so, specify the value to be written at the input parameter of the MOVE function, and specify "<Variable for reading and writing access parameters>.<parameter name>" at the output parameter of the MOVE function.

■ Placing Coils

Lastly, we will place a coil for resetting the SW1 contact.

(1) Click a location below the MOVE function to specify the position for placing a coil.

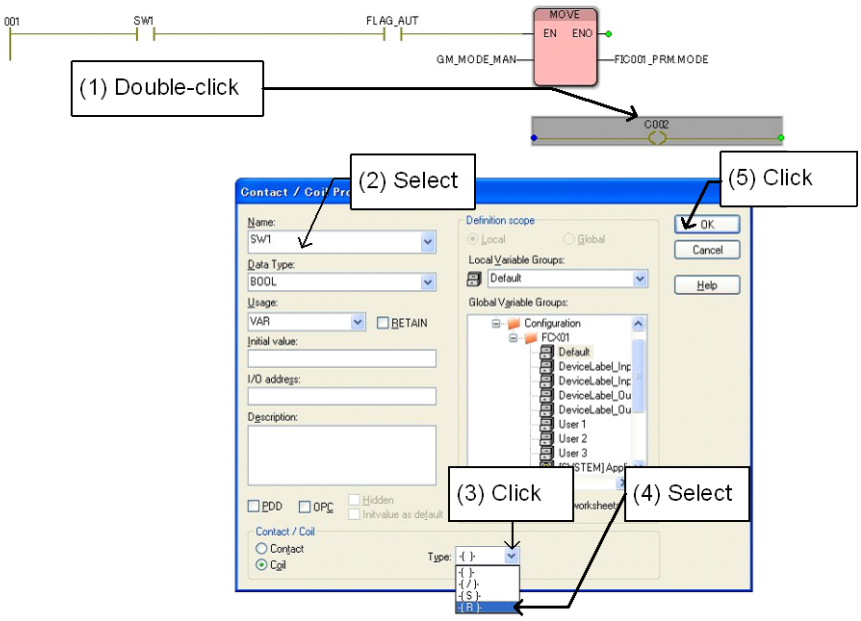
(2) Click [Coil right] on the toolbar. A coil is placed on the worksheet.

Coil right

(3) Double-click the coil. The Contact/Coil Properties dialog is displayed.

(4) Click the [Name] combo box, and select [SW1] from the displayed list of local variables.

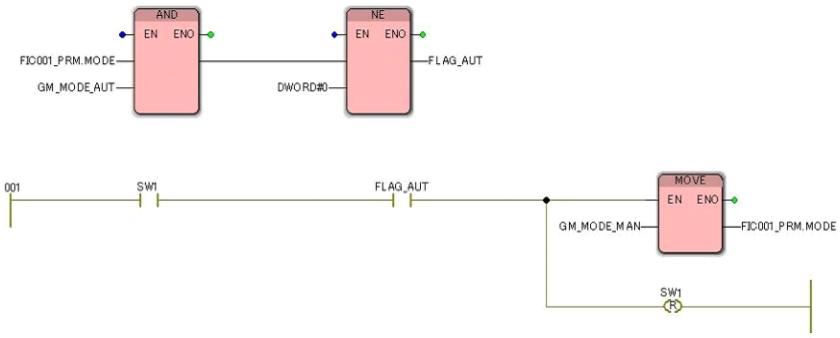
(5) Select [-(R)-] (RESET) for the coil type, and click [OK].



(6) With the coil selected, click [Right power rail] on the toolbar to place a power rail on the right of the coil.

Rigth power rail

(7) Finally, link the FLAG_AUT contact and the coil. Select the Connect Objects icon from the toolbar, and click the line on the right of the FLAG_AUT contact, followed by the blue circle on the left of the coil.



This completes the creation of our sequence application.

This sequence control switches FIC001 to MAN mode, and resets SW1 when SW1 becomes TRUE and FIC001 is in AUT mode.

■ Saving

(1) Select [File]-[Save] from the menu bar.

As shown above, when creating a sequence application, you not only can use functions such as AND function, NE function or MOVE function, but also can construct a sequence as a ladder circuit using contacts and coils.

4.3 Hands-on: Assigning to a Task

We have constructed a control application in Sections 4.1 and 4.2, but to execute the created program, we need to first assign it to a task. Let us assign the "SINGLE_LOOP" program to a task.

4.3.1 Checking Registered Tasks

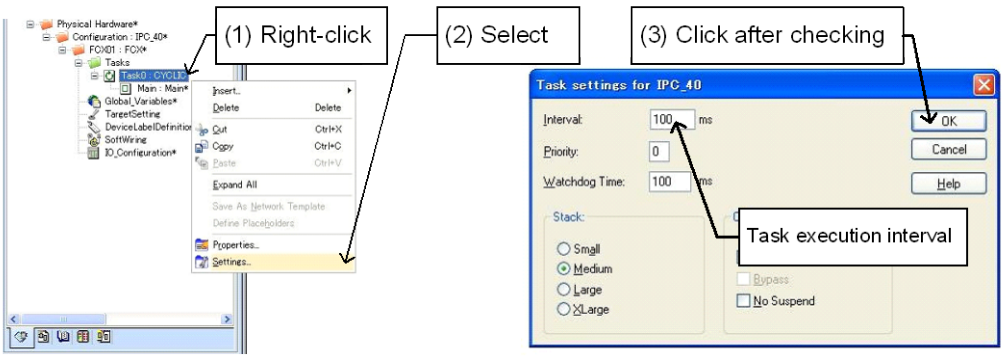
The new template project that we have created in Subsection 4.1.2 is provided with a default task. Let us check the settings of this task.

(1) Select [Task0] on the project tree. Click the right mouse button, and select [Properties] from the displayed menu. Task0 is a cyclic task.
After checking, click [Cancel] to close the dialog.

TIP: Types of Tasks

There are three types of tasks.
- Default task
  A default task has the lowest priority amongst all user tasks, and is executed only when no other user task is running.
- Cyclical Task
  A cyclical task is executed at the specified interval according to its defined priority.
- System task
  A system task is called automatically by the system when an error or some event occurs during FCN-500 or FCN-RTU operation.

(2) Select [Task0] on the project tree. Click the right mouse button, and select [Settings] from the displayed menu. The [Task settings for IPC_40] dialog (or [Task settings for SH04_40] dialog) is displayed.
In this exercise, we will use the default task interval and watchdog time settings.
After checking the values, click [Cancel] to close the dialog.

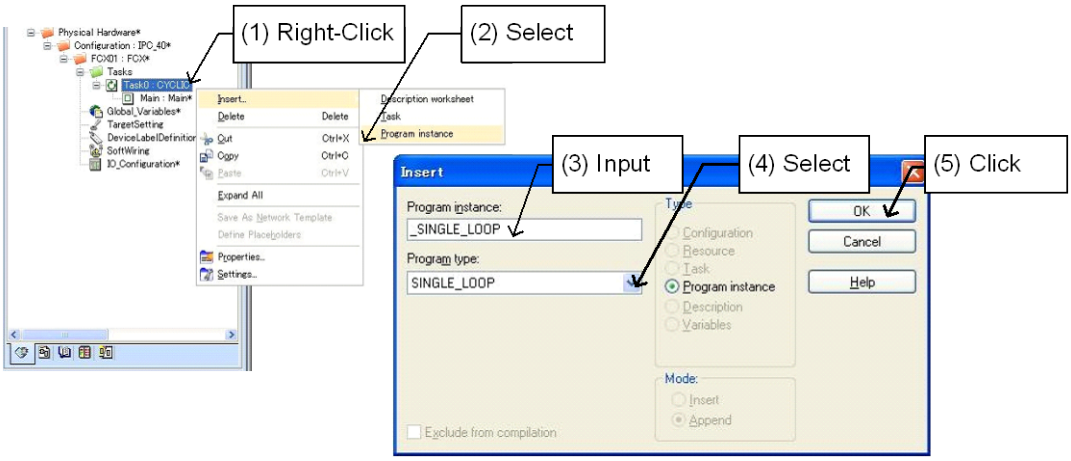


4.3.2 Assigning to a Task

Let us assign the "SINGLE_LOOP" program to a task.

(1) Select [Task0] on the project tree. Click the right mouse button, and select [Insert]-[Program instance] from the popup menu. The Insert dialog is displayed.

(2) Enter "_SINGLE_LOOP" in the [Program instance] field, and select [SINGLE_LOOP] for [Program type]. Click [OK].



TIP: Program Instance Name

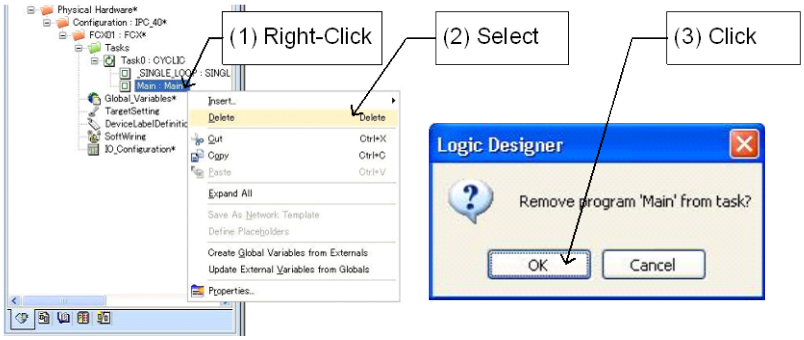
By prefixing a program instance name with the underscore ('_') character, the program instance name will be omitted from the object name when defined variables are imported into the VDS data server when a control application is created.

4.3.3 Deleting Programs Assigned to a Task

If the "Main" program registered in a template project is deleted, the task to which the "Main" program is assigned also becomes unnecessary.

(1) Select the [Main] program instance. Click the right mouse button, and select [Delete] from the displayed menu.

(2) Click [OK] when a dialog confirming whether to delete is displayed.

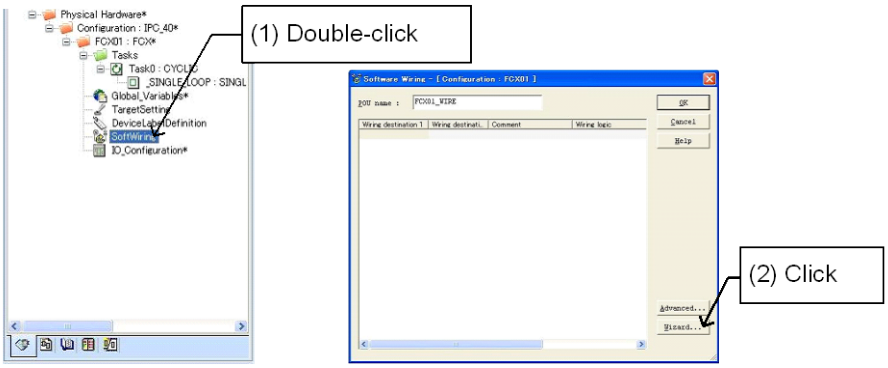


4.4 Hands-on: Software Wiring Setup

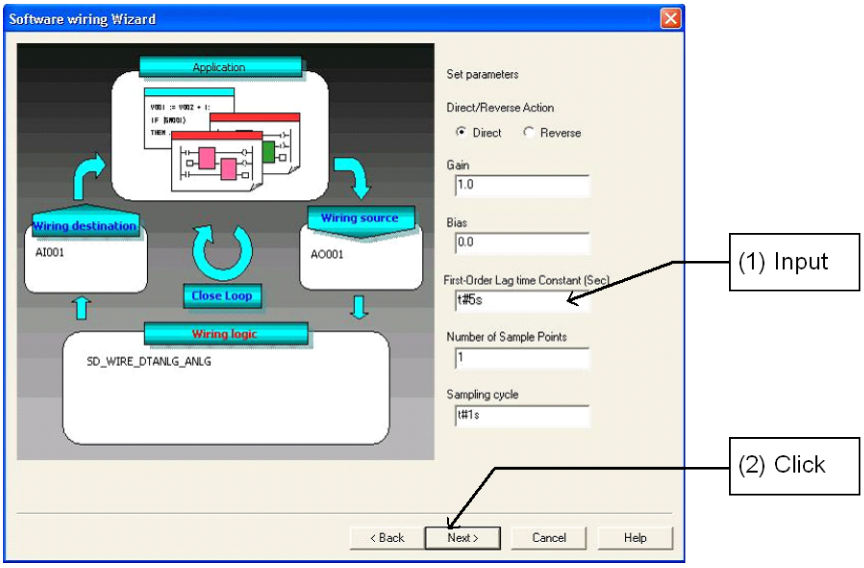
This section briefly describes how to use the software wiring function to perform debugging without using external signal input. The software wiring function can be used to connect the input and output of a control logic by software.

4.4.1 Defining Software Wiring

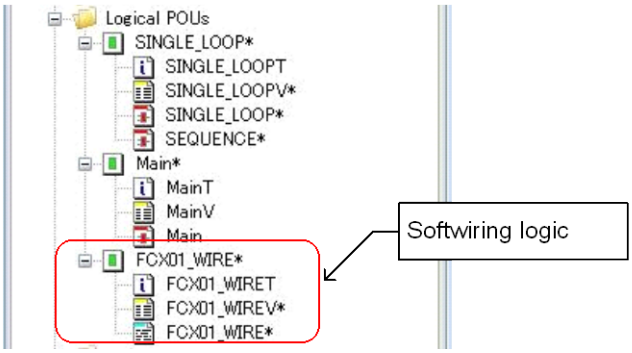
(1) Double-click "SoftWiring" on the project tree to open the Software Wiring dialog.
(2) Click [Wizard].



(3) Select [AI001] for [Wiring destination 1]. Click [Next].
(4) Select [Close Loop] for [Input form]. Click [Next].
(5) Select [AO001] for [Wiring source 1]. Click [Next].
(6) Select [SD_WIRE_DTANLG_ANLG] for [Wiring logic]. Click [Next].
(7) Enter "t#5s" (5 seconds) in [First-Order Lag time Constant]. Leave the other default values unchanged. Click [Next].



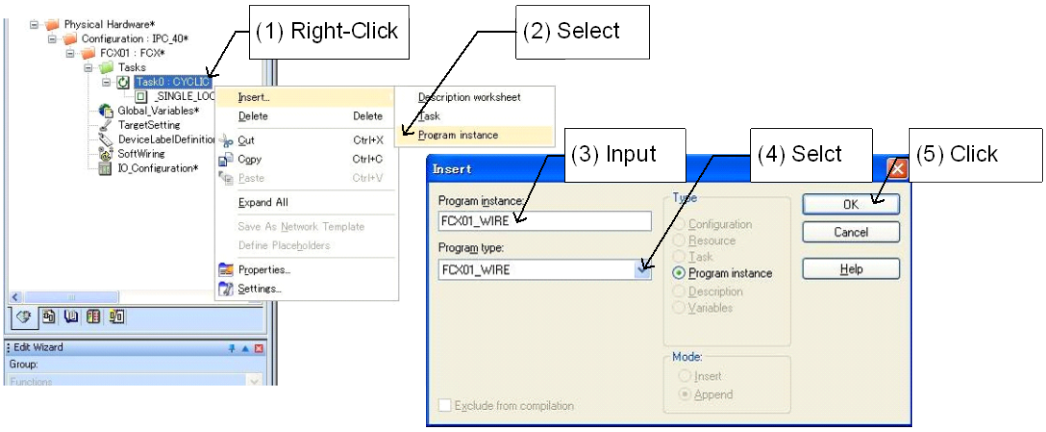
(8) Click [Finish]. The wizard exits.
(9) Click [OK] to close the Software Wiring dialog. [FCX01_WIRE] is added under [Logical POUs] in the project tree.



4.4.2 Assigning Software Wiring to a Task

Like programs, a software wiring definition cannot be executed unless it is assigned to a task.

(1) Select [Task0] on the project tree. Click the right mouse button, and select [Insert]-[Program instance] from the popup menu. The Insert dialog is displayed.
(2) Enter "FCX01_WIRE" in the [Program instance] field, and select [FCX01_WIRE] for [Program type]. Click [OK].



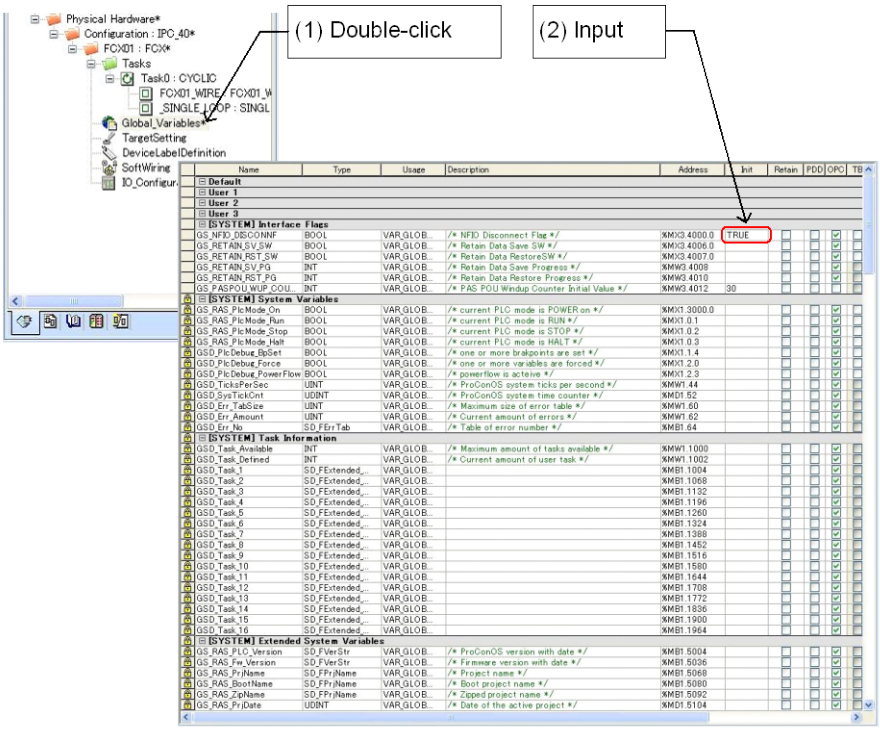
4.4.3 Disconnecting I/O

To use software wiring in place of process I/O, I/O must first be disconnected. This can be done by changing the value of the switch for disconnecting I/O, which is provided as a global variable.

(1) Double-click [Global_Variables] on the project tree to open the variable worksheet.

(2) Change the initial value of [GS_NFIO_DISCONNF], which is the switch for disconnecting I/O, to TRUE.

This completes I/O disconnection.



TIP: Disabling Software Wiring Function

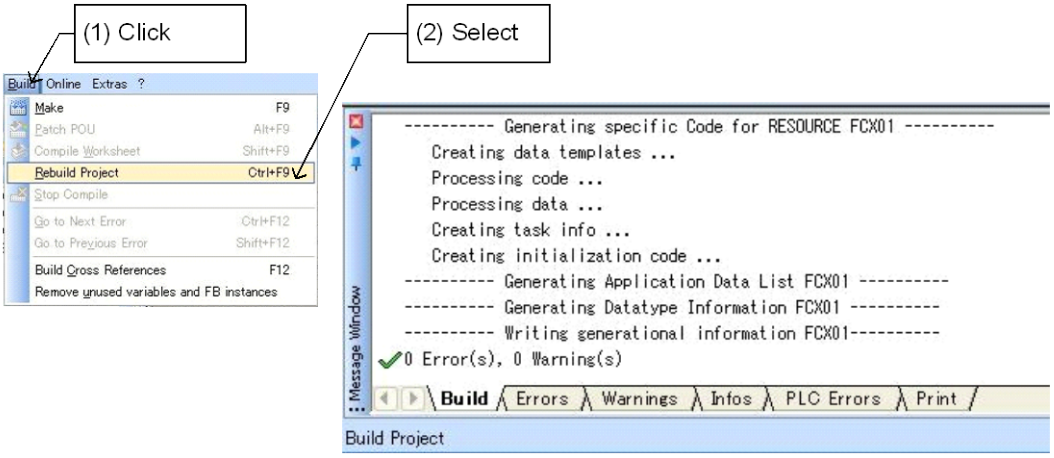
To disable the software wiring function, delete the software wiring definition assigned to a task, and restore the initial value of GS_NFIO_DISCONNF to FALSE.
Setting GS_NFIO_DISCONNF to FALSE in debug mode disables the software wiring function, as well as cancels I/O disconnection at the same time.
To not use a software wiring but leave I/O disconnected, delete the software wiring definition from the task.

4.5 Hands-on: Building a Project

We shall build the created project to convert it to executable code for an FCN-500 or FCN-RTU.

(1) Select [Build]-[Rebuild Project] from the menu bar. A message dialog showing the progress of compilation is displayed.

(2) When compilation completes, the compilation result is displayed in the message window. If the display indicates zero error, it means the compilation is successful.

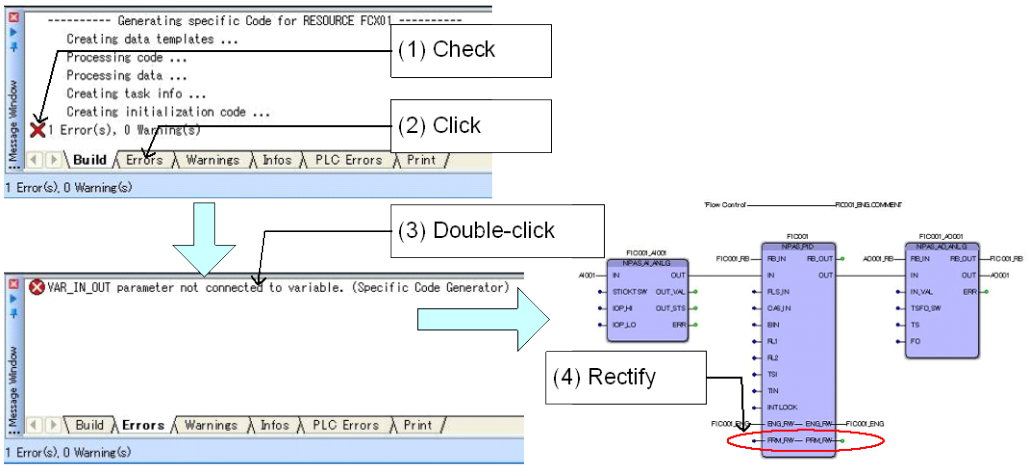


TIP: Fixing Compilation Errors and Warnings

If errors are detected during compilation, you can view individual error information by clicking the [Errors] tab on the message window. Double-clicking an error message displays the location of the error. Rectify each error by referring to its error message.

You can ignore all warnings, or click the [Warnings] tab on the message window, and go through the warnings the same way as errors.

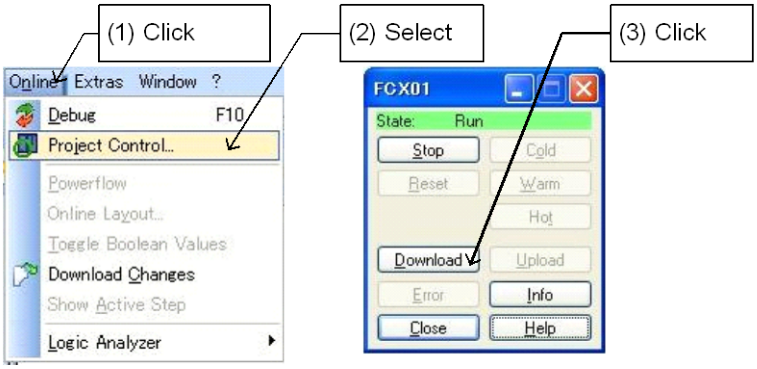
Repeat this process of program modification and compilation until no more errors and warnings are detected.



4.6 Hands-on: Downloading a Project

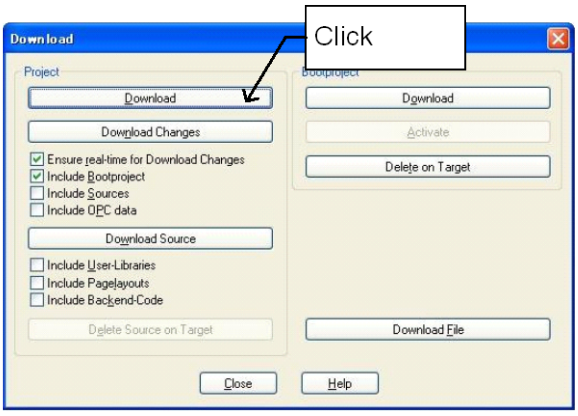
Let us download the compiled project to the FCN-500 or FCN-RTU.

(1) Select [Online]-[Project Control] from the menu bar. The Resource Control dialog is displayed.



(2) Click [Download]. The Download dialog is displayed.

(3) Click the [Download] button under [Project].



(4) If another project has been downloaded earlier, a dialog is displayed to confirm whether to overwrite the existing project. Select [Yes].

(5) The progress of downloading is shown in the status bar. Wait for downloading to complete (when 100% is downloaded).

TIP: Downloading to Flash Memory

In the above procedure, the checkbox for [Include Bootproject] is selected. When selected, the project is downloaded to both the volatile memory and the flash memory on the CPU module. When the checkbox is not selected, the project is downloaded only to the volatile memory. When the FCN-500 or FCN-RTU is powered off, project in the volatile memory disappears. When the FCN-500 or FCN-RTU is next powered on, the bootproject stored in the flash memory on the CPU module is loaded. To start the same project when the FCN-500 or FCN-RTU is switched on, turn on the [Include Bootproject] checkbox before downloading.

TIP: Scope for Downloading

When a project contains multiple resources, you can select which resource is to be downloaded using the Project Control dialog.

TIP: Online Download

Clicking [Download Changes] on the Download dialog performs online downloading without interrupting control. For more details on online downloading, see Section 1.2.2, “Logic Designer.”

4.7 Hands-on: Starting a Project

Downloading a control application does not execute it. To execute the project, you need to select one of three available start modes.
You may specify any of the following three modes during debugging depending on how you prefer variable values to be handled.

Table Start Modes of Control Engine

Start Mode	Non-retentive Variables	Retentive Variables	Remarks
Cold start	Values are initialized. Variables with specified initial values are initialized accordingly. Variables with no initial values specified are initialized with default values (0).	Values are initialized. Variables with specified initial values are initialized accordingly. Variables with no initial values specified are initialized with default values (0).	
Warm start	Values are initialized. Variables with specified initial values are initialized accordingly. Variables with no initial values specified are initialized with default values (0).	Values are retained. (*1)	
Hot start	Values are retained.	Values are retained.	Hot start is not allowed if reset processing or downloading of control application has been performed after the execution of a control application had stopped.

*1: Retentive variables are initialized if its configuration in the volatile memory has been changed.
If retentive data has been saved, the saved data is restored after initialization.

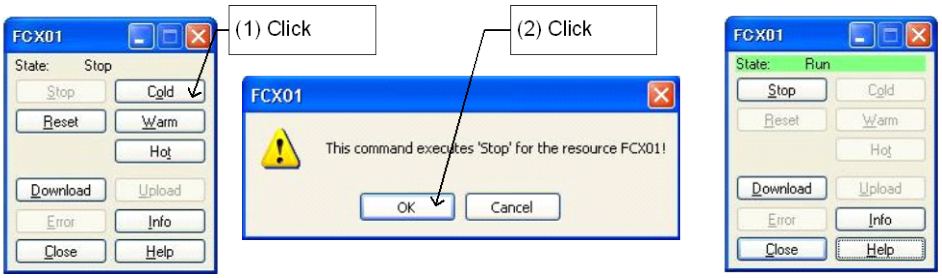
TIP

When an FCN-500 or FCN-RTU is switched on, it always performs a warm start.

In this exercise, we briefly describe how to perform a cold start (*), assuming that a new project has already been downloaded.

*: In accordance with the intended use, select the "start mode".

(1) Click the [Cold] button on the Project Control dialog. The displayed state on the dialog changes to “Run”.
The Project Control dialog closes.



We have now executed the created control application on the FCN-500 or FCN-RTU.
(Execution is carried out using software wiring as the initial value of GS_NFIO_DISCONNF has been set to TRUE.)

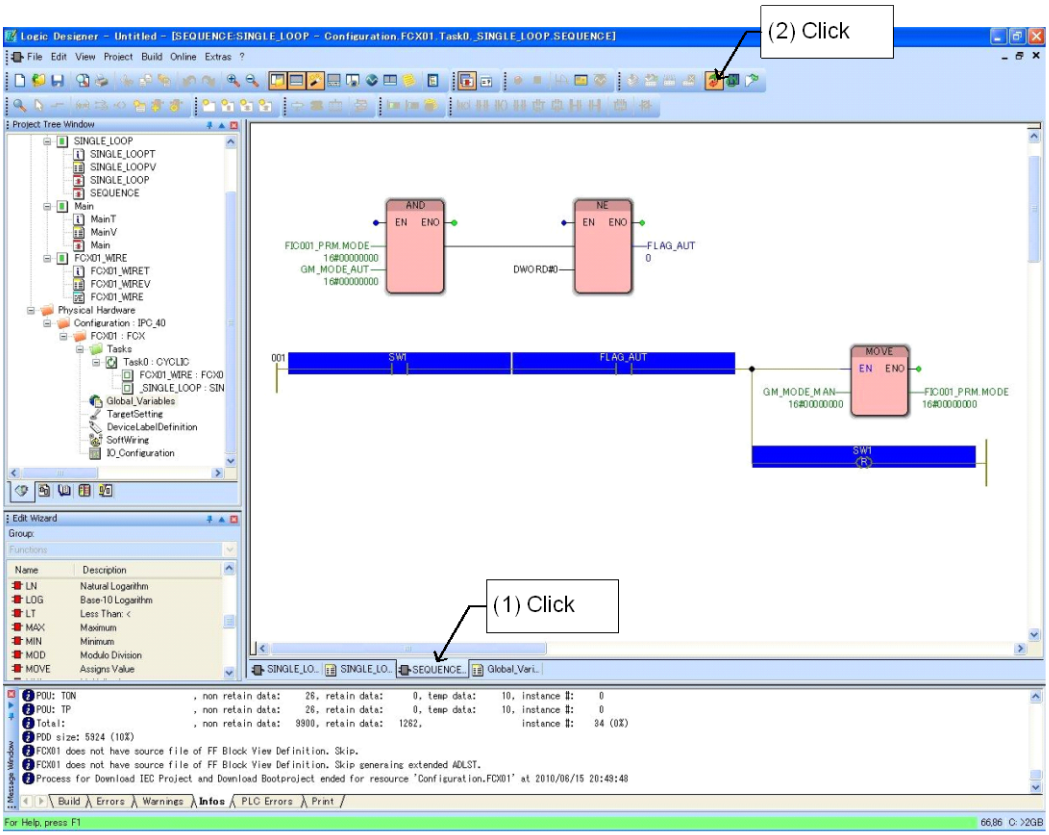
4.8 Hands-on: Checking Operation

Let us check whether the control application that is running on the FCN-500 or FCN-RTU is operating correctly.
To check the operation, we will first switch the Logic Designer to Debug Mode.

4.8.1 Switching Logic Designer to Debug Mode

The Debug Mode of Logic Designer is used during debugging to display the status of a control application running on an FCN-500 or FCN-RTU.
To switch to the Debug Mode, use the following procedure.

(1) Click the [Debug on/off] icon on the toolbar.
(2) Display the "SEQUENCE" code worksheet.



This completes switching to Debug Mode.

In Debug mode, the bottom area of the Logic Designer is displayed in green.

In addition, a BOOL type variable is displayed in blue when it is FALSE, but displayed in red when it is TRUE. Any other type of variable is displayed in green with its current value displayed below the variable.

4.8.2 Displaying Access Parameters

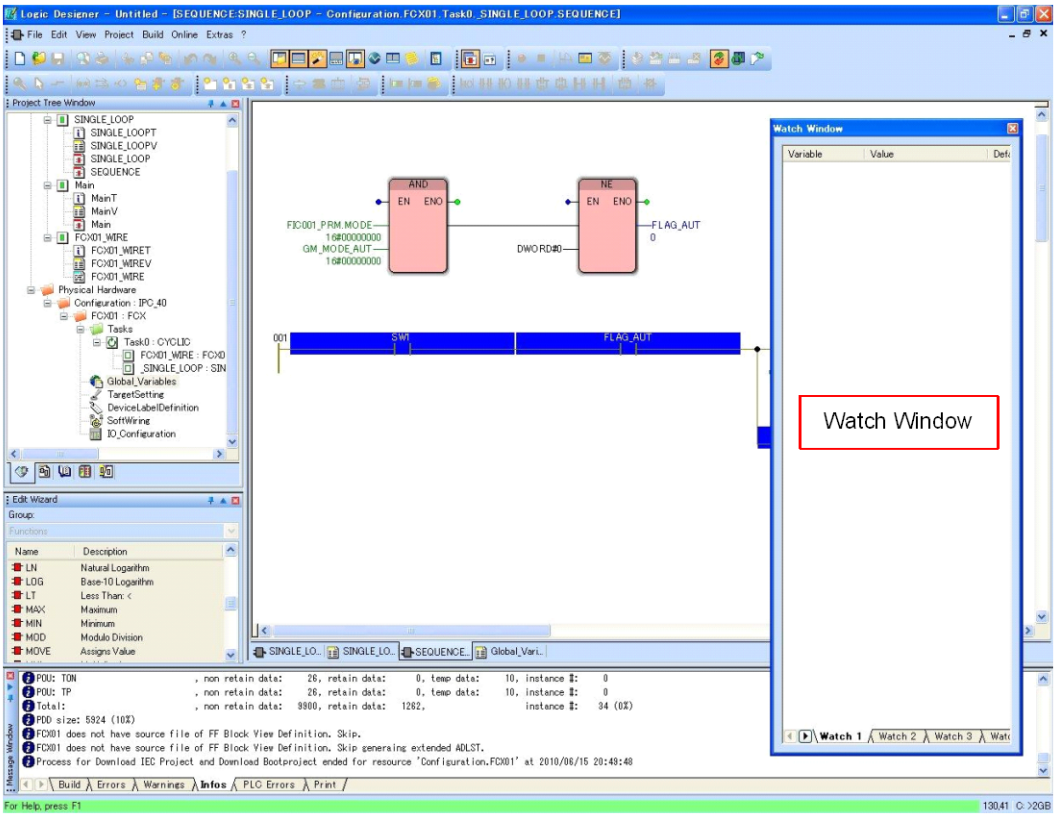
Next, we shall display access parameters of the "FIC001" controller to check the values of PV, SV and MV.

■ Displaying a Watch Window

A Watch Window is used to display access parameters.

The procedure below shows how to display the Watch Window.

(1) Select [View]-[Watch Window] from the menu to display the Watch Window.



TIP

By default, the Watch Window is displayed as a tiled window like the Project Tree window. To display it as an independent window like shown above, right-click in the Watch Window, and remove the [Allow Docking] checkmark on the displayed menu.

■ Adding the Variable for Reading and Writing Access Parameters to the Watch Window

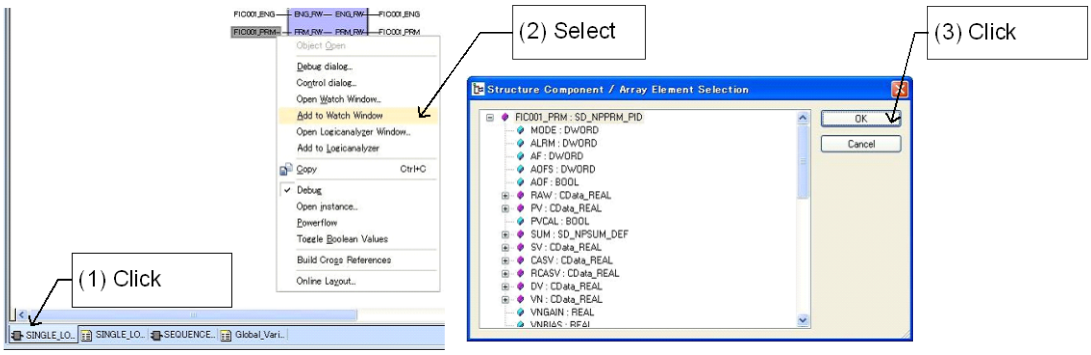
To display values of access parameters on the Watch Window, you need to first assign the variable for reading and writing access parameters to the Watch Window.

The procedure below describes how to assign the variable for reading and writing access parameters to the Watch Window.

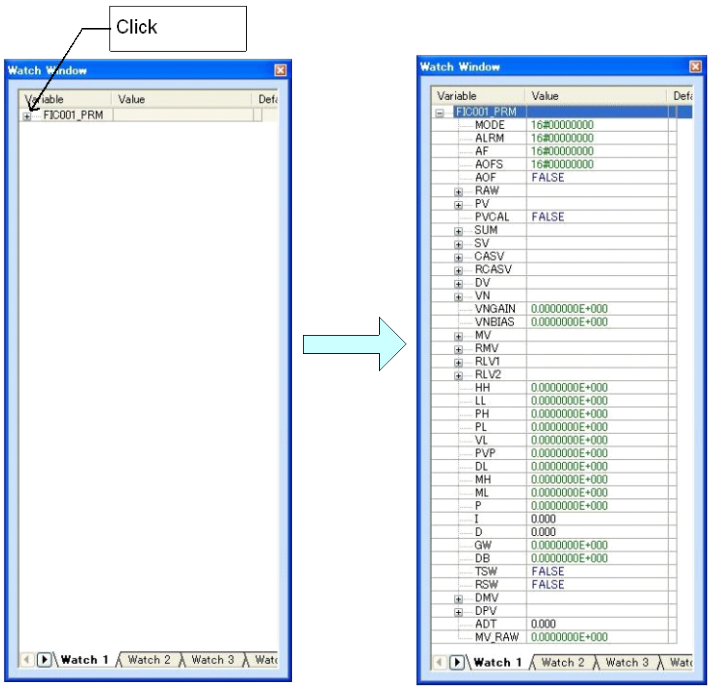
(1) Display the "SINGLE_LOOP" code worksheet.

(2) Select the variable for reading and writing access parameters (“FIC001_PRM”) of "FIC001" on the worksheet. Right-click and select [Add to Watch Window] from the popup menu.

(3) Click [OK]. "FIC001_PRM" is added to Watch Window



(4) Next, we shall expand [FIC001_PRM] to display values for PV, SV and MV.



We have displayed values of access parameters of the "FIC001" controller.
The PV, SV, MV values are displayed in their respective Value fields.

TIP

The mode of a NPAS POU block is shown as a hexadecimal value in the MODE field on the Watch Window. The following list shows typical block modes with their corresponding DWORD values. (The prefix "16#" denotes a hexadecimal value)

- MAN: 16#00800000
- AUT: 16#00400000
- CAS: 16#00200000

4.8.3 Checking Operation of Control Loop

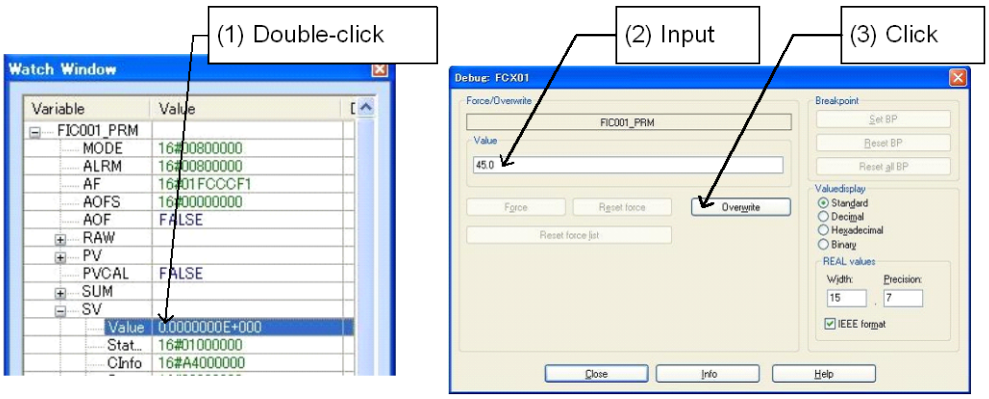
We shall now check whether the "FIC001" controller is operating normally by changing values of access parameters of the control loop.

■ Changing SV Value

Let us change the SV value using the Watch Window.

(1) Double-click the [FIC001_PRM]-[SV]-[Value] item in the Watch Window. The "Debug: FCX01" window is displayed.

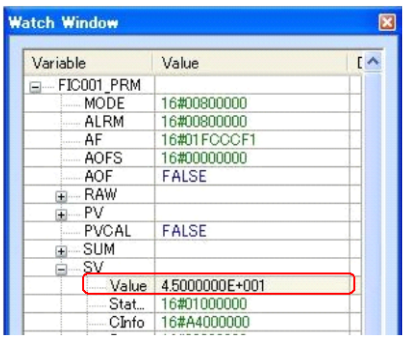
(2) Enter "45.0" for [Value], and click the [Overwrite] button.



TIP

When changing the value of a REAL type variable such as SV, always enter the new value, including a decimal point (e.g. “45.0”).

The SV value is changed to 45.0 l/h.



TIP

Member PV, SV or MV of the variable for reading and writing access parameters (FIC001_PRM) is not a single data value, but contains, in addition to the data value, scale high/low limits and status information. Therefore, to set the SV value, you should set the value to the [Value] field of SV. To set the SV value from a sequence, use the expression "FIC001_PRM.SV.Value".

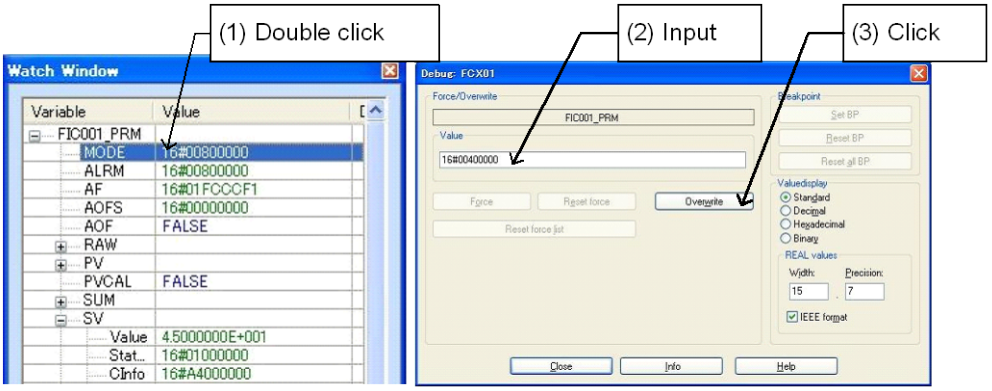
■ Changing Block Mode

We have changed the SV value of the "FIC001" controller but the other values remain unchanged because the current block mode has the value "MAN" (16#00800000).

Next, we shall switch to "AUT" mode to check whether the controller is operating normally.

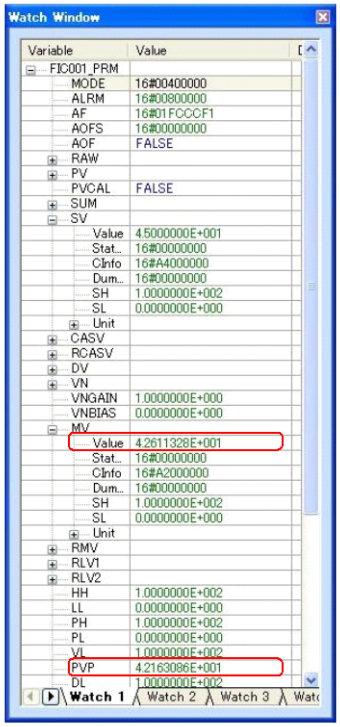
(1) Double-click the [FIC001_PRM]-[MODE] item in the Watch Window. The "Debug: FCX01" window is displayed.

(2) Enter "16#00400000" for [Value], and click the [Overwrite] button.



We have changed the block mode of the "FIC001" controller to "AUT".

Verify that the MV value changes and that the PV value follows according to the software wiring.



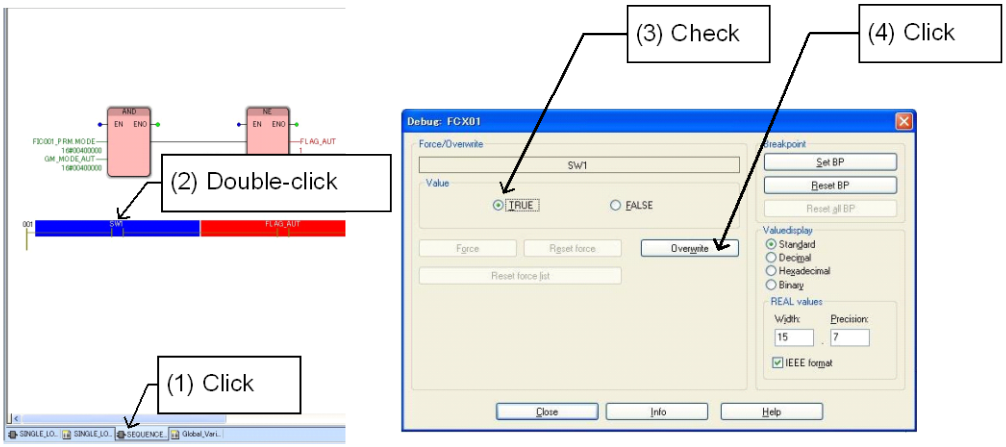
4.8.4 Checking Operation of Sequence

The last thing we need to do is to check the operation of the sequence.

(1) Display the "SEQUENCE" code worksheet.

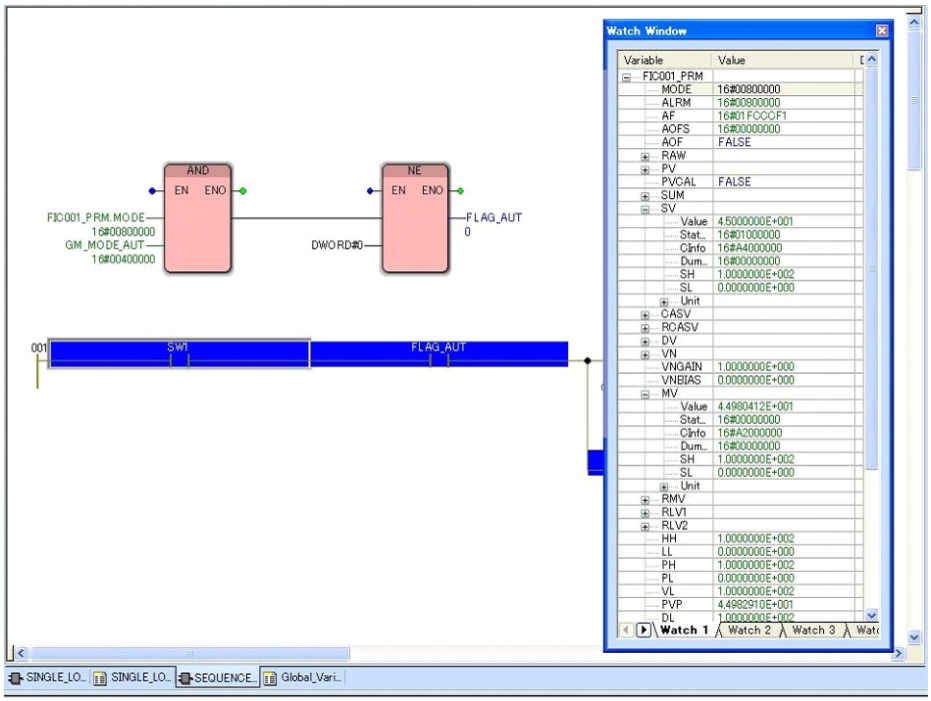
(2) Double-click the "SW1" contact. The [Debug: FCX01] window is displayed.

(3) Check that the [TRUE] option is selected under [Value], and click the [Overwrite] button.



The above operation sets SW1 to TRUE and executes the sequence.

Check the execution result of the sequence by verifying that the block mode of the "FIC001" controller has been changed to 'MAN' (16#00800000) and the "SW1" contact has been reset.



This completes checking of the operation of the control application.

Finally, click the Debug On/Off icon on the toolbar to exit from Debug mode.

Blank Page

Appendix 1 Operating FCN-500, FCN-RTU from VDS

This appendix describes how to perform the required definition for operating the control application built in Chapter 4 on the FCN-500 or FCN-RTU from VDS.
It briefly explains how to operate the PID loop using the VDS Tuning Panel.

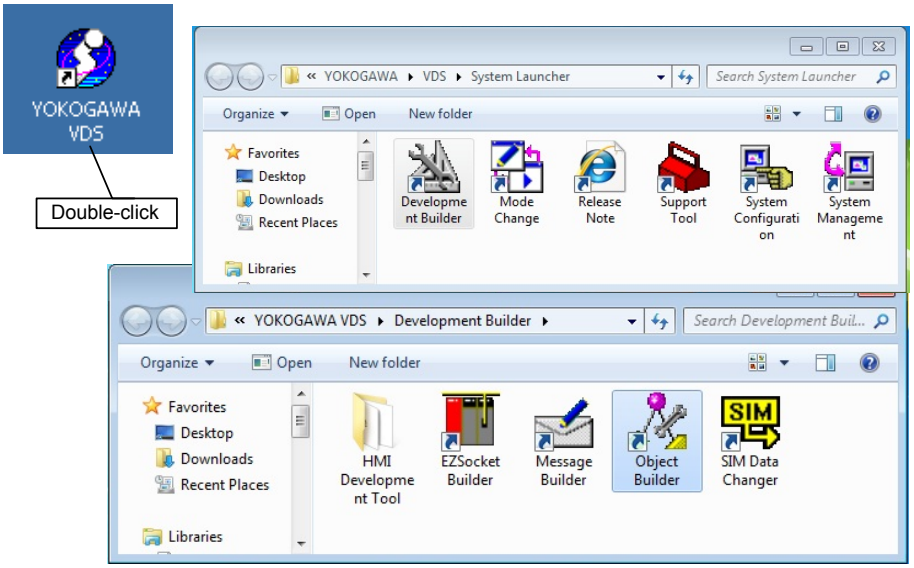
Appendix 1.1 Defining Connection between VDS and FCN-500, FCN-RTU

We shall explain how to obtain information defined in a control application as a monitoring tag in VDS for the purpose of operating a PID loop using a VDS Tuning Panel.

■ Starting Object Builder

First, we need to start Object Builder. Object Builder is a tool for performing data acquisition definition on VDS.

(1) Start Object Builder either from the YOKOGAWA VDS icon on the desktop by selecting [System Launcher]-[Development Builder]-[Object Builder], or from the start menu by selecting [Start]-[Programs]-[YOKOGAWA ASTMAC VDS]-[Development Builder]-[Object Builder].



■ Importing Definition Information

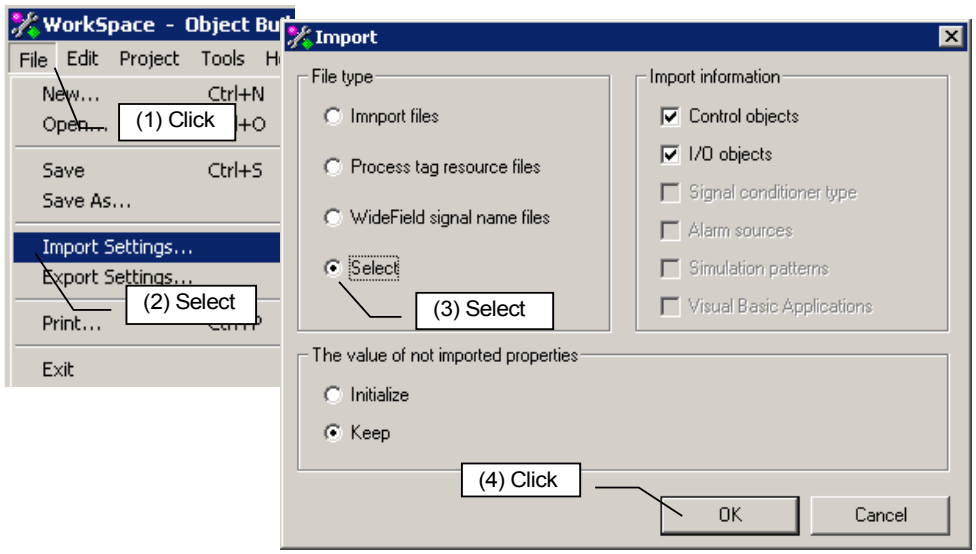
Object Builder can be used to read definition information generated by Logic Designer.

Logic Designer generates a CSV file known as an application data list (ADLST) during compilation. By importing the ADLST file into Object Builder, we can then define acquisition of FCN-500 and FCN-RTU data by VDS.

TIP: Variables Registered in ADLST

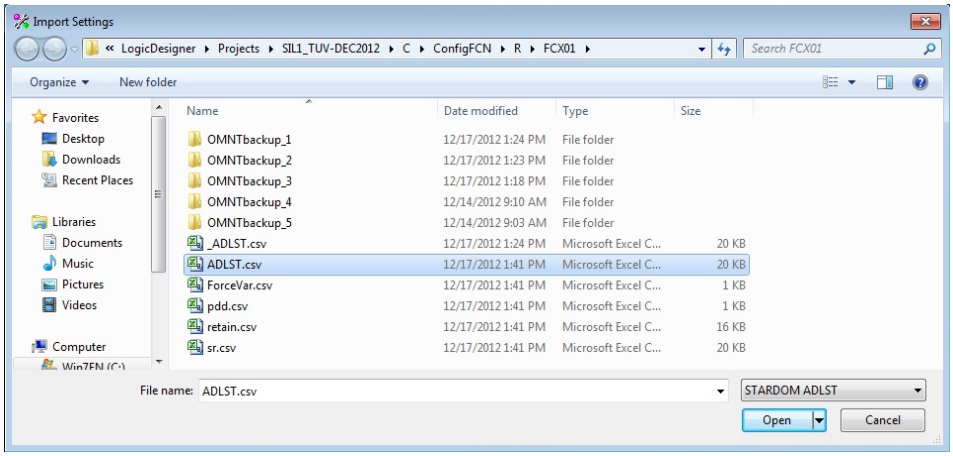
While device label variables and engineering parameters are registered in ADLST by default, user-defined variables are not registered in ADLST by default and hence are not accessible from VDS. To access a user-defined variable from VDS, you need to turn on its [OPC Property] during variable definition.

(1) Select [File]-[Import Settings] from the menu bar. The Import dialog is displayed.
(2) Select the [Select] option under [File type], and click [OK]. The [Select CSV type] dialog is displayed.

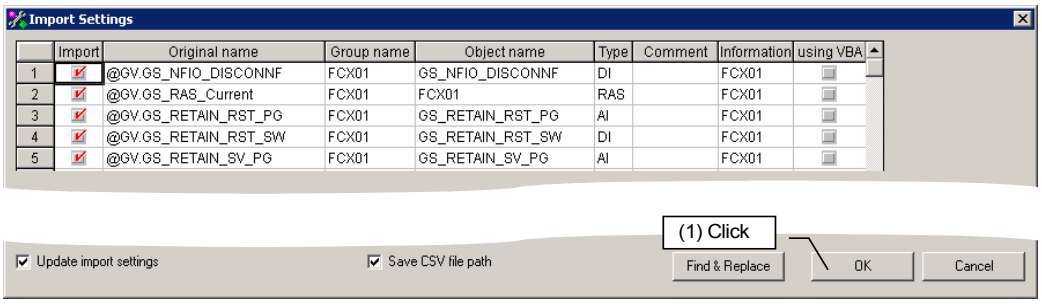


(3) Verify that [STARDOM ADLST] is selected. Click [OK]. The [Import Settings] dialog is displayed.

(4) Specify the following folder for the file location.
C:\YOKOGAWA\FCN-FCJ\LogicDesigner\Projects\training\C\Configuration\R\FCX01
(This location should be modified if Logic Designer was not installed in the default directory)



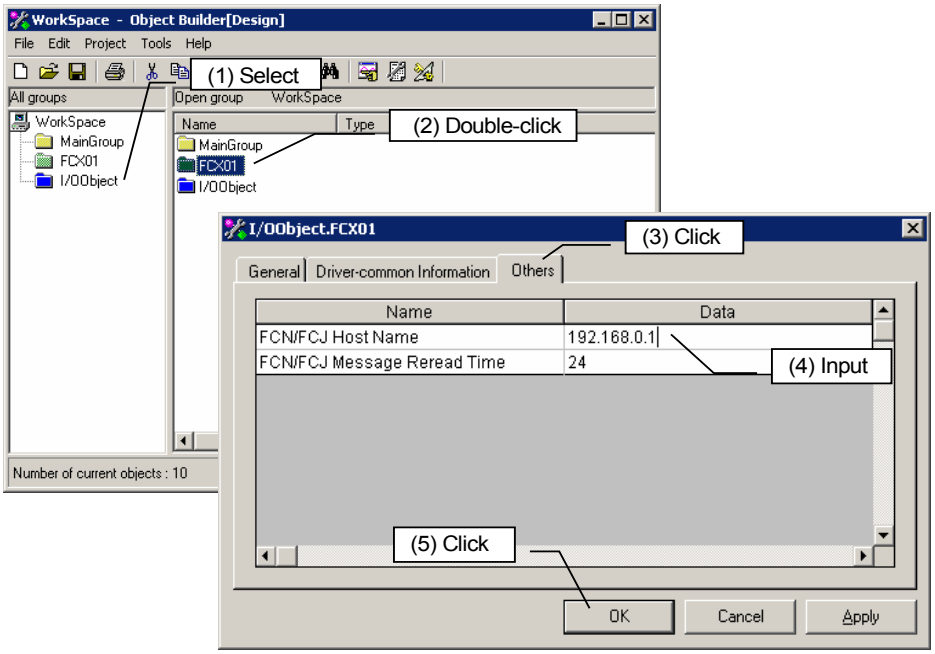
(5) Select filename [ADLST.csv], and click the [Open] button. The [Import Settings] dialog is displayed.
(6) Click [OK]. Data is imported, and the group [FCX01] is added to the window.



■ Communication Setup

Communication setup configures I/O objects (communication driver) for communication with the FCN-500 or FCN-RTU.

(1) In the Object Builder window, click [I/OObject] to open the group.
(2) Double-click the [FCX01] I/O object to open its property window.
(3) Select the [Others] tab. Enter "192.168.01" for FCN FCJ Host Name, and click [OK]. The property window closes.



TIP: Controller Object

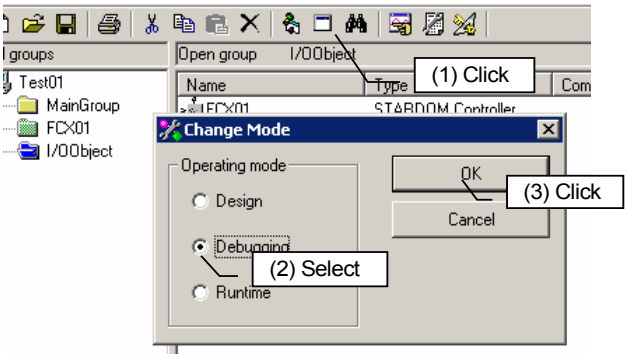
The above operation also sets up the controller object to be accessed in the POU on the FCN-500 or FCN-RTU. The specified controller object can be displayed by selecting [WorkSpace]-[FCX01] on the Object Builder window.

■ Changing Mode of Object Builder

Let us switch the Object Builder to Debug Mode. Communication with FCN-500 or FCN-RTU begins when the Object Builder is switched to Debug Mode.

(1) Select [File]-[Save As] from the menu bar. The Save As dialog is displayed.
(2) Enter "Test01" for the filename, and then click [Save].
(3) Select [Tool]-[Change Mode] from the menu bar. The Change Mode dialog is displayed.
(4) Select [Debugging] for the operating mode, and click [OK].

Switching to Debug Mode starts the VisualBasicEditor automatically. Since we are not using the editor, click the [Close] button to quit the editor.



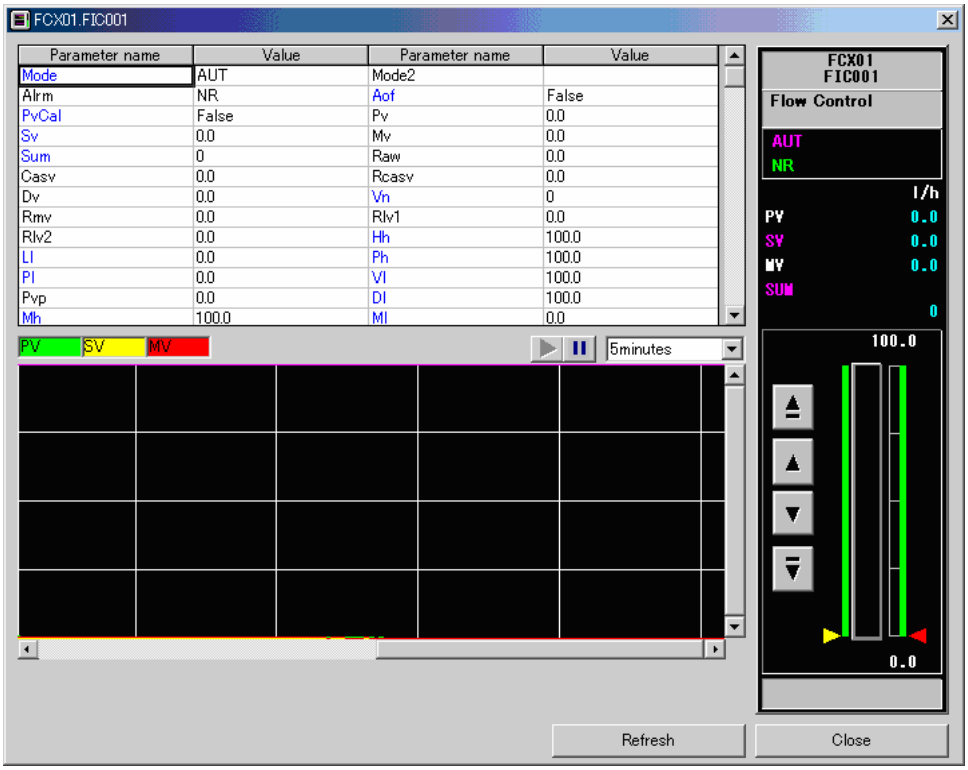
Appendix 1.2 Operating the PID Controller

■ Starting Tuning Panel

(1) Start the FCN/FCJ Tuning Panel by selecting [System Launcher]-[Support Tool]-[FCN FCJ Tuning Panel].

(2) Select group [FCX01], and double-click object [FIC001] from the displayed objects to open the Tuning Panel for FIC001.

(3) The following screen is displayed when step (2) has been completed.



■ Operation

Let us check the operation of the PID controller by changing its SV value and mode using the Tuning Panel.

● Changing SV Value

Click the numerical value of SV. An input dialog is displayed.

Enter the desired numerical value, and click the [Input] button.

● Changing Mode

Click the "MAN" string. A dialog for changing mode is displayed.

Select the desired mode, and click [OK].

Check that the PV value changes with the MV value to confirm that the FIC001 is running.

This ends our description on how to operate an FCN-500 or FCN-RTU from VDS.

Appendix 2 Creating and Deleting Global Variables

This appendix describes how global variables are represented in the system and how to create and delete global variables.

■ System Representation of Global Variables

● Global Variable and Local Variable

Variables in Logic Designer are classified into two types, namely, global variables and local variables.

A global variable can be referenced and used by all POUs on an FCN-500 or FCN-RTU, while a local variable can be referenced and used only within the POU (program, function block or function) where it is defined.

The figure below illustrates the concepts of global variables and local variables.

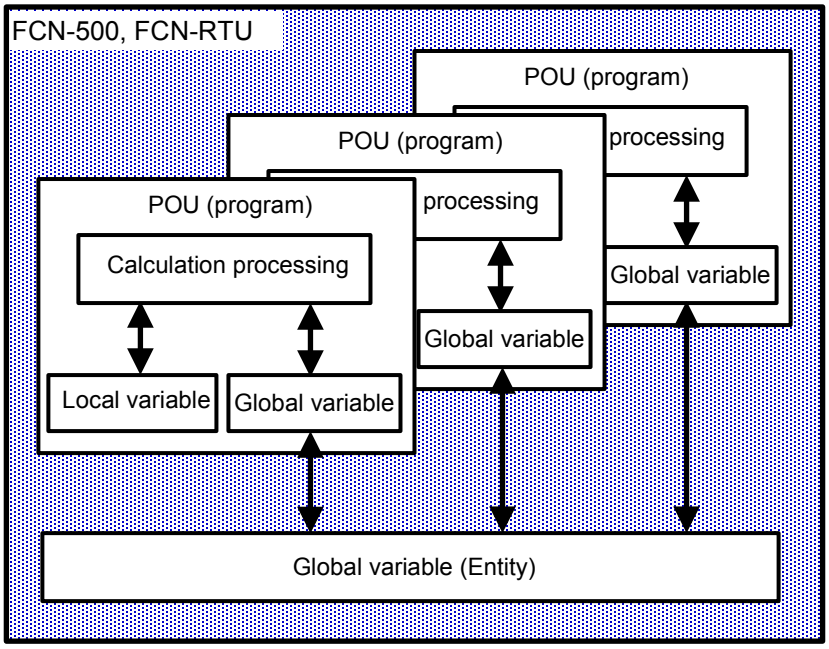


Fig. Global Variable and Local Variable

As shown in the above figure, a local variable defines a variable that is accessible only within a POU (program), while a global variable defines a variable that is accessible and shared by all POUs (programs). All POUs referencing the same global variable name read and write to the same global variable entity on the FCN-500 or FCN-RTU.

Device label variables connected to I/O signals (physical I/O) of an FCN-500 or FCN-RTU are also defined as global variables.

● Representation of Global Variables and Local Variables on Worksheets

The diagram below shows how global variables and local variables are represented on the various types of worksheets.

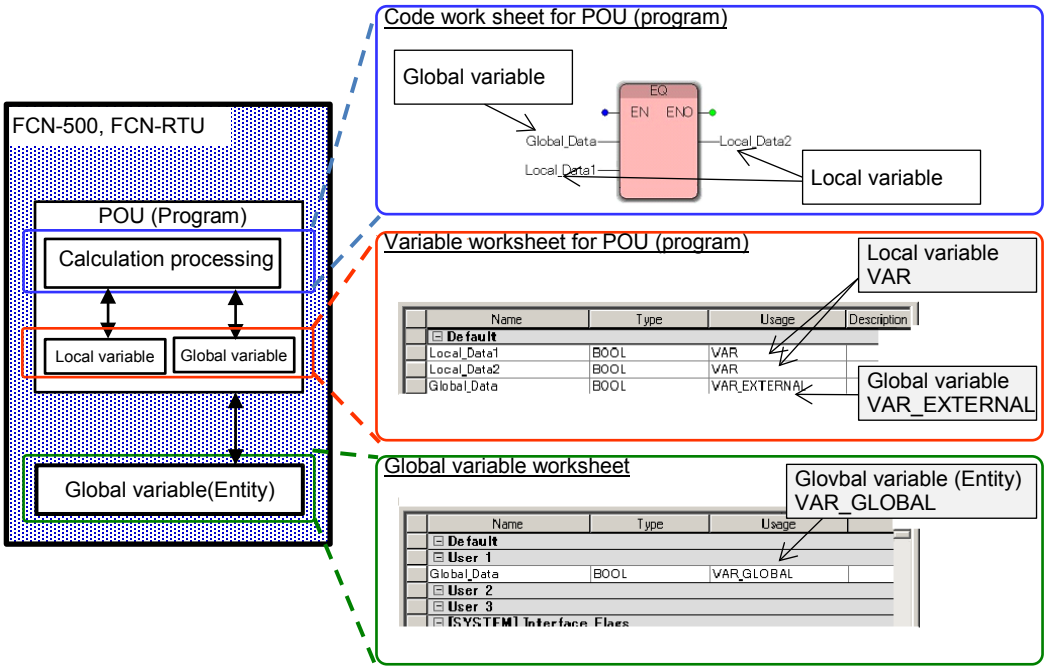


Fig. Representation of Global Variables and Local Variables on Worksheets

The sample program in the above figure compares whether a user-created global variable and local variable have the same value, and outputs the result of the comparison to another local variable.

The global variable named “Global_Data” and the local variable named “Local_Data1” are inputs to an “EQ” (equal) function, which outputs the comparison result to a local variable named “Local_Data2.”

In the figure, the global variable named “Global_Data” is defined using “VAR_GLOBAL” in the global variable worksheet. It is declared using “VAR_EXTERNAL” on the variable worksheet of a POU (program), and finally referenced within actual calculation on a code worksheet.

TIP

In actual operation, when you specify a global variable, which is already defined on a global variable worksheet, on a code worksheet of a POU, the system automatically creates a “VAR_EXTERNAL” declaration on the variable worksheet of the POU (program).
Similarly, when you create a new global variable on a code worksheet of a POU, the system automatically creates a “VAR_EXTERNAL” declaration on the variable worksheet of the POU and a “VAR_GLOBAL” definition on the global variable worksheet.

You can also see from the above figure that local variables named “Local_Data1” and “Local_Data2” exist only within the POU (program).

Appendix 2.1 Creating a New Global Variable

This section describes how to create a new global variable.

A new global variable can be created either on the global variable worksheet or from a code worksheet of a POU (program) that references the global variable in actual calculation. We describe below the latter procedure.

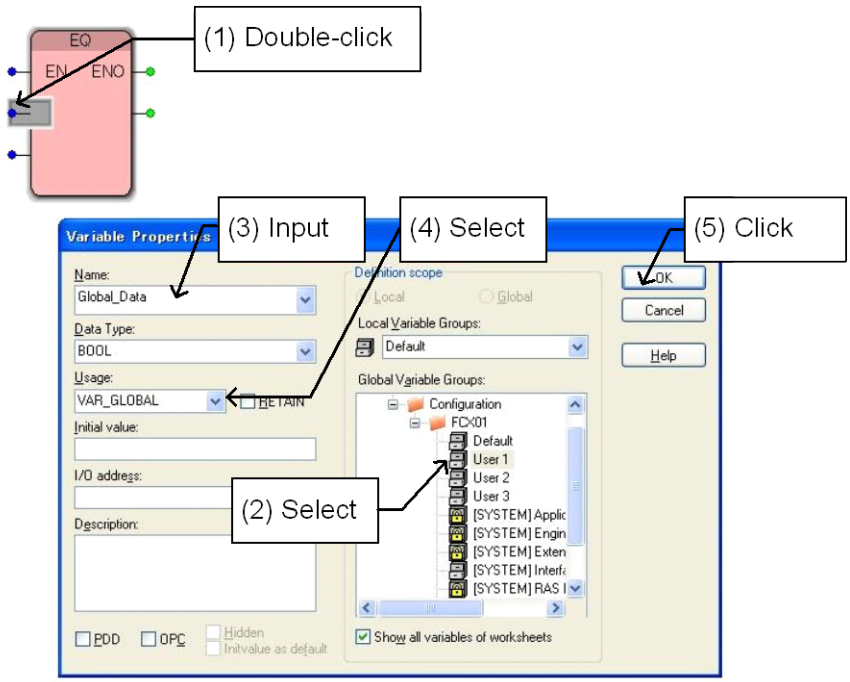


Fig. Creating a New Global Variable

Double-click the IN terminal on the code worksheet to display the Variable Properties dialog. Select global variable group “User1” and select [Global] for scope. Enter “Global_Data” as the variable name. Select [VAR] for [Usage]. Finally, click [OK]. The global variable is created.

At this point, the system automatically generates a “VAR_EXTERNAL” declaration on the variable worksheet of the POU (program) and a “VAR_GLOBAL” definition on the global variable worksheet.

TIP

Instead of “User1”, you can also select a different global variable group in the above procedure. (Beware that the dialog retains any specified setting).
Select either “Default”, “User1”, “User2”, “User3” or any user-created group as the global variable group for a user-created global variable. Do not select any group that begins with “DeviceLabel” or "[System]" for a user-created global variable as these are system-reserved groups.

IMPORTANT

The Variable Properties dialog retains previous setting values. Therefore, always remember to revert the selection of step (3) shown in the figure to [Local] when defining a local variable after defining a global variable.

In particular, remember to revert the selection of step (3) shown in the figure to [Local] when defining a local variable after selecting a global variable group such as “DeviceLabel_Input_A” to define a device label variable. Beware that the device label definition dialog will not open if a user-created variable is defined in a global variable group such as “DeviceLabel_Input_A”.

Appendix 2.2 Deleting a Global Variable

This section describes how to delete a user-created global variable.
When deleting a user-created global variable, you must not only delete its references from the code worksheet, but also delete its "VAR_EXTERNAL" declaration on the variable worksheet of the POU (program) and its "VAR_GLOBAL" definition on the global variable worksheet. The "VAR_EXTERNAL" declaration and "VAR_GLOBAL" definition may have been automatically generated by the system earlier.
The actual procedure is given below.

First, delete the global variable from the code worksheet.

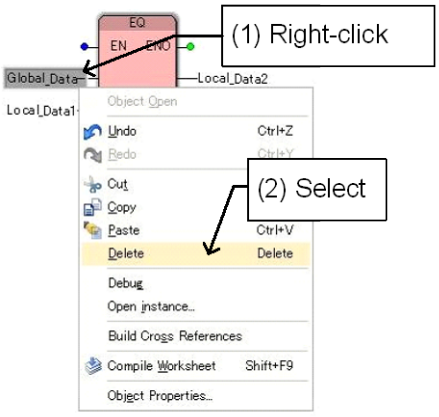


Fig. Deleting a Global Variable from a Code Worksheet

Next, delete the global variable from the variable worksheet of the POU (program).

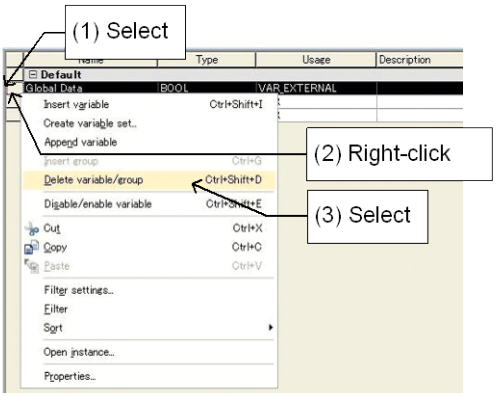


Fig. Deleting a Global Variable from the Variable Worksheet of a POU (Program)

TIP

When a user deletes a global variable reference from a code worksheet of a POU (program), the system does not automatically delete the declaration of the global variable from the variable worksheet. This system behavior takes into consideration the fact that one POU (program) can have multiple code worksheets so the global variable may be referenced from another code worksheet.
Similarly, deleting a local variable from a code worksheet does not automatically delete the local variable from the variable worksheet.

Finally, delete the global variable from the global variable worksheet.

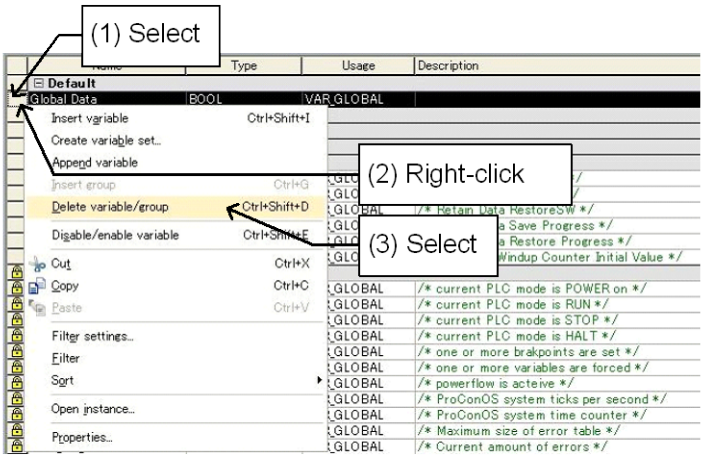


Fig. Deleting a Global Variable from the Global Variable Worksheet

TIP

When a user deletes a global variable from a code worksheet or variable worksheet of a POU (program), the system does not automatically delete the global variable from the global variable worksheet. This system behavior takes into consideration the fact that the global variable may be used by another POU (program).
Hence, to completely delete a global variable, remember to also delete it from the global variable worksheet.

The above procedure can be used to completely delete a global variable.

Revision Information

Title: FCN-500/FCN-RTU Primer – Fundamentals
Document No.: TI 34P02K13-02E

Apr. 2016/1st Edition
Newly published

May 2017/2nd Edition for R4.10 or later
Revised
- Support for Duolet on FCN-500.

Jun. 2018/3rd Edition for R4.20 or later*
Revised
- Added FCX_C. processor type

*: Denotes the release version of the software corresponding to the contents of the technical information in question. The revised contents are valid until the next edition is issued.

■ For Questions and More Information
If you have any questions, you can send an E-mail to the following address.
E-mail:stardom_info@cs.jp.yokogawa.com
■ Written by Yokogawa Electric Corporation
■ Published by Yokogawa Electric Corporation
2-9-32 Nakacho, Musashino-shi, Tokyo 180-8750, JAPAN