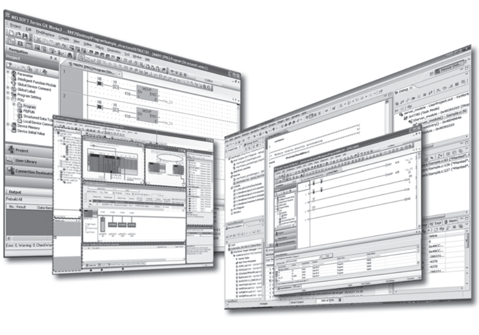
MITSUBISHI ELECTRIC

MELSOFT

Engineering Software

PX Developer Version 1 Programming Manual (1/2)

-SW1D5C-FBDQ-E
-SW1D5C-FBDQMON-E



• SAFETY PRECAUTIONS •

(Always read these instructions before using this product.)

Before using this product, thoroughly read this manual and the relevant manuals introduced in this manual and pay careful attention to safety and handle the products properly.
The precautions given in this manual are concerned with this product. For the safety precautions of the programmable controller system, refer to the User's Manual for the CPU module.
In this manual, the safety precautions are ranked as "⚠WARNING" and "⚠CAUTION".

⚠ WARNING	Indicates that incorrect handling may cause hazardous conditions, resulting in death or severe injury.
⚠ CAUTION	Indicates that incorrect handling may cause hazardous conditions, resulting in minor or moderate injury or property damage.

Note that the ⚠CAUTION level may lead to serious consequences according to the circumstances. Always follow the precautions of both levels because they are important for personal safety.

Please save this manual to make it accessible when required and always forward it to the end user.

[Security Precautions]

⚠WARNING

- To maintain the security (confidentiality, integrity, and availability) of the programmable controller and the system against unauthorized access, denial-of-service (DoS) attacks, computer viruses, and other cyberattacks from external devices via the network, take appropriate measures such as firewalls, virtual private networks (VPNs), and antivirus solutions.

[Startup/Maintenance Precautions]

⚠ CAUTION

- The online operations have to be executed after the manual has been carefully read and the safety has been ensured.
  Failure to do so may cause a miss operation which results in machine damage or an accident.

● CONDITIONS OF USE FOR THE PRODUCT ●

(1) MELSEC programmable controller ("the PRODUCT") shall be used in conditions;
i) where any problem, fault or failure occurring in the PRODUCT, if any, shall not lead to any major or serious accident; and
ii) where the backup and fail-safe function are systematically or automatically provided outside of the PRODUCT for the case of any problem, fault or failure occurring in the PRODUCT.

(2) The PRODUCT has been designed and manufactured for the purpose of being used in general industries.
MITSUBISHI ELECTRIC SHALL HAVE NO RESPONSIBILITY OR LIABILITY (INCLUDING, BUT NOT LIMITED TO ANY AND ALL RESPONSIBILITY OR LIABILITY BASED ON CONTRACT, WARRANTY, TORT, PRODUCT LIABILITY) FOR ANY INJURY OR DEATH TO PERSONS OR LOSS OR DAMAGE TO PROPERTY CAUSED BY the PRODUCT THAT ARE OPERATED OR USED IN APPLICATION NOT INTENDED OR EXCLUDED BY INSTRUCTIONS, PRECAUTIONS, OR WARNING CONTAINED IN MITSUBISHI ELECTRIC USER'S, INSTRUCTION AND/OR SAFETY MANUALS, TECHNICAL BULLETINS AND GUIDELINES FOR the PRODUCT. ("Prohibited Application")
Prohibited Applications include, but not limited to, the use of the PRODUCT in;
- Nuclear Power Plants and any other power plants operated by Power companies, and/or any other cases in which the public could be affected if any problem or fault occurs in the PRODUCT.
- Railway companies or Public service purposes, and/or any other cases in which establishment of a special quality assurance system is required by the Purchaser or End User.
- Aircraft or Aerospace, Medical applications, Train equipment, transport equipment such as Elevator and Escalator, Incineration and Fuel devices, Vehicles, Manned transportation, Equipment for Recreation and Amusement, and Safety devices, handling of Nuclear or Hazardous Materials or Chemicals, Mining and Drilling, and/or other applications where there is a significant risk of injury to the public or property.

Notwithstanding the above, restrictions Mitsubishi Electric may in its sole discretion, authorize use of the PRODUCT in one or more of the Prohibited Applications, provided that the usage of the PRODUCT is limited only for the specific applications agreed to by Mitsubishi Electric and provided further that no special quality assurance or fail-safe, redundant or other safety features which exceed the general specifications of the PRODUCTs are required. For details, please contact the Mitsubishi Electric representative in your region.

(3) Mitsubishi Electric shall have no responsibility or liability for any problems involving programmable controller trouble and system trouble caused by DoS attacks, unauthorized access, computer viruses, and other cyberattacks.

REVISIONS

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Dec., 2002	SH (NA)-080371E-A	First edition
Apr., 2003	SH (NA)-080371E-B	Correction Section 2.11.1, Section 7.6.20, Appendix 1.2
Oct., 2003	SH (NA)-080371E-C	Addition Appendix 5.1 Correction Section 2.2.4, Section 2.3.1, Section 2.10, Section 2.11.1, Section 5.1.1, Section 5.1.2, Section 7.9.1 to 7.9.4, Chapter 8, Section 8.1.3, Section 8.1.4, Section 8.2.1, Section 8.2.2, Section 8.2.3, Appendix 1.1, Appendix 5
Jun., 2004	SH (NA)-080371E-D	Model Addition Q12PRHCPU, Q25PRHCPU Addition Section 2.14 Correction Terms, Section 1.1, Section 1.3.1, Section 2.2.4, Section 2.3.1, Section 2.10 (Whole), Section 4.10, Section 5.5 (Whole), Section 7.4.7, Section 7.6, Chapter 8, Appendix 1, Appendix 2, Appendix 5
Jun., 2004	SH (NA)-080371E-E	Correction Section 7.6.7, Section 7.6.12
Feb., 2005	SH (NA)-080371E-F	Addition Section 4.9.5 Correction Chapter 7, Section 7.1.1, Section 7.1.2, Section 7.6.20, Appendix 1.1, Appendix 4, Appendix 5, Index

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Feb., 2006	SH (NA)-080371E-G	Addition Section 7.5.4, Section 7.6.7, Section 7.6.8, Section 7.6.11, Section 7.6.12, Section 7.8.9, Section 7.8.10, Section 7.8.13, Section 7.8.14, Appendix 3.14 Correction Section 1.2, Section 2.9.3, Section 2.9.4, Section 2.11.13, Section 2.14.2, Chapter 3, Section 4.1.16, Section 4.1.17, Section 4.1.19, Section 7.1.1, Section 7.1.2, Section 7.4.6, Section 7.5.1, Section 7.5.2, Section 7.5.6, Section 7.6.17, Section 7.6.18, Section 7.6.19, Section 7.7.1, Section 7.8.19, Section 7.8.20, Section 7.8.35, Appendix 1, Appendix 2, Appendix 3, Appendix 5 Section 7.5.4 to 7.5.7 changed to Section 7.5.5 to 7.5.8 Section 7.6.13 to 7.6.14 changed to Section 7.6.5 to 7.6.6 Section 7.6.5 to 7.6.6 changed to Section 7.6.9 to 7.6.10 Section 7.6.7 to 7.6.12 changed to Section 7.6.13 to 7.6.18 Section 7.6.15 to 7.6.23 changed to Section 7.6.19 to 7.6.27 Section 7.8.13 to 7.8.16 changed to Section 7.8.5 to 7.8.8 Section 7.8.5 to 7.8.6 changed to Section 7.8.11 to 7.8.12 Section 7.8.7 to 7.8.12 changed to Section 7.8.15 to 7.8.20 Section 7.8.17 to 7.8.31 changed to Section 7.8.21 to 7.8.35
Mar., 2007	SH (NA)-080371E-H	Addition Section 7.6.25, Section 7.8.36, Section 8.1.4, Section 8.1.6, Section 8.1.11, Appendix 3.15 Correction Section 2.9.3, Section 2.9.4, Section 2.10, Section 5.5.1, Section 5.5.2, Section 7.5.4, Section 7.6.7, Section 7.6.8, Section 7.6.25, Section 7.7.1, Section 7.8.9, Section 7.8.10, Section 7.8.31, Section 7.8.32, Section 7.8.33, Section 7.8.36, Chapter 8, Section 8.2.1, Section 8.2.2, Section 8.2.3, Section 8.4.3, Section 8.4.4, Appendix 1, Appendix 1.1, Appendix 1.2, Appendix 1.3, Appendix 3.3, Appendix 3.14, Appendix 5, INDEX Section 7.6.25 to 7.6.27 changed to Section 7.6.26 to 7.6.28 Section 8.1.4 changed to Section 8.1.5 Section 8.1.5 to 8.1.8 changed to Section 8.1.7 to 8.1.10
Jun., 2008	SH (NA)-080371E-I	Model Addition Q02PHCPU, Q06PHCPU Addition Section 7.1.8, Section 8.2.2, Section 8.2.5, Appendix 3.11 Correction MANUALS, GENERIC TERMS, ABBREVIATIONS, AND TERMS, Section 1.2, Section 2.4, Section 2.9.1, Section 2.10, Section 7.1.7, Section 7.5.1, Section 7.7.1, Chapter 8, Section 8.2.1 Section 8.4.2, Appendix 1.2 to 1.3, Appendix 3.10, Appendix 5 Section 8.2.2 to 8.2.3 changed to Section 8.2.3 to 8.2.4 Appendix 3.11 to 3.15 changed to Appendix 3.12 to 3.16

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Jan., 2009	SH (NA)-080371E-J	Addition Section 7.5.9, Section 7.8.37, Appendix 3.12 Correction Section 2.2.5, Section 2.9.3, Section 2.10, Section 2.14.1, Section 4.2.1, Section 7.5.1, Section 7.6.7 to 7.6.10, Section 7.8.1 to 7.8.28, Section 8.2.2, Appendix 1, Appendix 5 Appendix 3.12 to 3.16 changed to Appendix 3.13 to 3.17
Dec., 2009	SH (NA)-080371E-K	Addition CONDITIONS OF USE FOR THE PRODUCT, Section 7.1.9, Section 7.6.29 to 7.6.31, Section 7.8.38 to 7.8.40, Section 7.9.9, Section 8.1.12, Appendix 3.18 Correction SAFETY PRECAUTIONS, Section 2.9.3, Section 2.10, Section 7.5.4, Section 7.5.9, Section 7.6.7, Section 7.6.8, Section 7.8.9, Section 7.8.10, Section 8.1.10, Appendix 1 to 1.3, Appendix 5
Dec., 2010	SH (NA)-080371E-L	Addition Section 2.9 to 2.9.7, Section 4.2.7, Section 4.3.6, Section 8.2.29, Section 9.1.34, Appendix 3.19 Correction MANUALS, GENERIC TERMS, ABBREVIATIONS, AND TERMS, Section 2.1, Section 2.3.1, Section 2.10.3, Section 2.13, Section 2.15.3, Section 4.7.1, Section 4.10.5, Section 8.1.4, Section 8.2.26 to 8.2.28, Section 10.1.10, Appendix 1.1, Appendix 1.3, Appendix 3.14, Appendix 5 Section 2.9 to 2.14 changed to Section 2.10 to 2.15 Section 7.5 to 7.7.1 changed to Chapter 8 Section 7.8 to 7.11.1 changed to Chapter 9 Chapter 8 changed to Chapter 10
Oct., 2011	SH (NA)-080371E-M	Addition Section 7.4.11, Section 9.1.42, Section 9.1.43, Appendix 4 Correction Section 2.1, Section 2.2.1, Section 2.2.5, Section 2.10.3, Section 2.15.3, Section 8.1.1, Section 8.2.19, Section 8.3.1, Appendix 1 to 1.3, Appendix 6 Appendix 4 to 5.1 changed to Appendix 5 to 6.1

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Feb., 2014	SH (NA)-080371E-N	Addition Section 4.3.7, Section 9.3.2, Section 9.4.2, Section 10.1.13 Correction GENERIC TERMS, ABBREVIATIONS, AND TERMS, Section 1.3.1, Section 2.2.4, Section 2.3.1, Section 2.7.4, Section 2.10.3, Section 2.11 to 2.11.2, Section 2.12.1, Section 2.15 to 2.15.4, Section 4.1.1 to 4.1.21, Section 4.2.1 to 4.2.7, Section 4.3.1 to 4.3.6, Section 4.4.1, Section 4.4.2, Section 4.5.1, Section 4.6.1 to 4.6.4, Section 4.7.1, Section 4.8.1 to 4.8.8, Section 4.9.1 to 4.9.5, Section 4.10.1, Section 5.4.1. Section 5.4.3, Section 5.4.5, Section 6.1.1 to 6.1.5, Section 8.2.7, Section 8.2.8, Section 8.2.30 to 8.2.32, Section 9.1.39 to 9.1.41, Chapter 10, Section 10.1.1 to 10.1.12, Section 10.2.1 to 10.2.5, Section 10.3.1, Section 10.3.2, Section 10.5.1 to 10.5.4, Appendix 1 to Appendix 1.3, Appendix 3.7, Appendix 3.12, Appendix 4, Appendix 6, Appendix 6.1
Jul., 2015	SH (NA)-080371E-O	Correction Section 2.6.1, Section 4.7.1, Section 7.4.6, Section 7.4.7, Section 8.2.25
Jan., 2017	SH (NA)-080371E-P	Correction Section 8.1.8, Section 8.2.1, Section 8.2.3, Section 8.2.5, Section 8.2.7, Section 8.2.9, Section 8.2.11, Section 8.2.13, Section 8.2.15, Section 8.2.17, Section 8.2.19, Section 8.2.20, Section 8.2.22, Section 8.2.27, Section 8.2.28, Section 8.2.29, Section 9.1.1, Section 9.1.3, Section 9.1.5, Section 9.1.7, Section 9.1.9, Section 9.1.11, Section 9.1.13, Section 9.1.15, Section 9.1.17, Section 9.1.19, Section 9.1.21, Section 9.1.23, Section 9.1.25, Section 9.1.32, Section 9.1.33, Section 9.1.34, Section 9.1.42, Section 9.1.43
Apr., 2019	SH (NA)-080371E-Q	Model Addition Q04UDPVCPU, Q06UDPVCPU, Q13UDPVCPU, Q26UDPVCPU Correction GENERIC TERMS, ABBREVIATIONS, AND TERMS, Section 2.2.5, Section 2.4, Section 2.15.3, Section 4.10.5
Apr., 2020	SH (NA)-080371E-R	Correction Section 2.11.1, Section 10.1.1 to 10.1.13, Section 10.2.1 to 10.2.5, Section 10.3.1, Section 10.3.2
Oct., 2021	SH (NA)-080371E-S	Correction SAFETY PRECAUTIONS, CONDITIONS OF USE FOR THE PRODUCT, Section 9.1.29, Section 9.1.30

Japanese Manual Version SH-080261- AO

INTRODUCTION

Thank you for purchasing the engineering software, MELSOFT series.
Read this manual and make sure you understand the functions and performance of MELSOFT series thoroughly in advance to ensure correct use.

CONTENTS

Programming Manual (2/2)

MANUALS

The following manuals are also related to this product.
Refer to the following table for ordering a manual.

Related manuals

Manual name	Manual number (model code)
PX Developer Version 1 Programming Manual Details of programming with PX Developer, lists of FB parts, and the PID instructions (this manual) (Sold separately.)	SH-080371E (13JW00)
PX Developer Version 1 Operating Manual (Programming Tool) FBD language programming, compilation, online operation and debug methods with PX Developer (Sold separately.)	SH-080369E (13JU38)
PX Developer Version 1 Operating Manual (Monitor tool) Operation methods of the monitor tool and methods for monitoring and controlling DDC processing with tag FB (Sold separately.)	SH-080370E (13JU39)
PX Developer Version 1 Operating Manual (GOT Screen Generator) Generation procedure for GOT screen project and details about generated screen (Sold separately.)	SH-080772ENG (13JU61)
PX Developer Version 1 Operating Manual (InTouch Interaction) Interaction between PX Developer monitor tool and SCADA software (InTouch) (Sold separately.)	SH-080773ENG (13JU62)
PX Developer Version 1 Operating Manual (JoyWatcherSuite Interaction) Interaction between PX Developer monitor tool and SCADA software (JoyWatcherSuite) (Sold separately.)	SH-080976ENG (13JU70)

CAUTION

- Please note that we do not guarantee commercially available software compatible with Microsoft® Windows® Operating System introduced in this manual.
- The software copyright of this product belongs to Mitsubishi Electric Corporation.
- No contents in this manual can be reproduced or duplicated in any form or by any means without permission.
- Although we make utmost efforts, this manual may not completely follow the revisions of the software and hardware.
- In principle, this software should be purchased by one set per personal computer or by license purchase.
- This product (including this manual) can only be used under the software license agreement.
- Please note that we are not responsible for any influence resulting from operating this product (including this manual).
- The contents of this manual are subject to change without notice.

HOW TO USE THIS MANUAL

"HOW TO USE THIS MANUAL" is arranged according to different needs in using: Please refer to the following contents when using this manual:

(1) Hoping to learn features, product configuration and project flow (☞ Chapter 1)
Features are described in Section 1.1; product configuration is illustrated in Section 1.2; and the project flow in Section 1.3.

(2) Hoping to learn the programming method of FBD language (☞ Chapter 2)
FBD language and its programming method are described in Chapter 2.

(3) Programming with FBD parts (☞ Chapter 3 to Chapter 10, Appendix 1)
- Reading method of instructions after Chapter 4 is described in Chapter 3.
- Input/output pins parameter, function and program example of general functions are described in Chapter 4.
- Input/output pins parameter, function and program example of general FB are described in Chapter 5.
- Input/output pins parameter, function and program example of process function are described in Chapter 6.
- Input/output pins parameter, public variable, function, and program example of process FB are described in Chapter 7, Chapter 8, and Chapter 9.
- Input/output pins, public variable, function, and program example of module FB are described in Chapter 10.
- The tag data list and its detailed information are in Appendix 1.

(4) Hoping to learn the contents of error codes for process control (☞ Appendix 2)
The check method and contents of error codes for process control are elaborated in Appendix 2.

(5) Hoping to learn process-related functions (☞ Appendix 3, Appendix 5)
- Process-related functions are elaborated in Appendix 3.
- Relative terms are elaborated in Appendix 5.

GENERIC TERMS, ABBREVIATIONS, AND TERMS

The following table shows the generic terms, abbreviations, and terms in this manual.

(1) Generic terms and abbreviations

Generic term/abbreviation	Description
PX Developer	Generic term for PX Developer Version 1 (SW1D5C-FBDQ-E) and PX Developer Monitor Tool (SW1DNC-FBDQMON-E) For PX Developer, Programming Tool and Monitor Tool are installed. For PX Developer Monitor Tool, only Monitor Tool is installed.
GX Works2	Abbreviation for GX Works2 Version 1 (SW1DNC-GXW2-E Version 1.98C) or later
GX Developer	Abbreviation for GX Developer Version 7 (SW7D5C-GPPW-E Version 7.20W) or later
GX Simulator	Abbreviation for GX Simulator Version 7 (SW7D5C-LLT-E Version 7.27D) or later
GX application	Generic term for GX Works2 and GX Developer which are interacted with PX Developer
GX project	Generic term for GX Works2 project and GX Developer project included in PX Developer project
FBD program	Generic term for a program created in FBD language
FBD part	Generic term for parts (FB part, function part, variable part, constant part, comment part, etc.) used by the programming tool
Global part	Generic term for module FB, tag FB, and global variable
Peripheral device	Generic term for the personal computer on which PX Developer can be used
QCPU	Generic term for Q00JCPU, Q00UJCPU, Q00CPU, Q00UCPU, Q01CPU, Q01UCPU, Q02CPU, Q02HCPU, Q02PHCPU, Q02UCPU, Q03UDCPU, Q03UDECPU, Q03UDVCPU, Q04UDHCPU, Q04UDEHCPU, Q04UDVCPU, Q04UDPVCPU, Q06HCPU, Q06PHCPU, Q06UDHCPU, Q06UDEHCPU, Q06UDVCPU, Q06UDPVCPU, Q10UDHCPU, Q10UDEHCPU, Q12HCPU, Q12PHCPU, Q12PRHCPU, Q13UDHCPU, Q13UDEHCPU, Q13UDVCPU, Q13UDPVCPU, Q20UDHCPU, Q20UDEHCPU, Q25HCPU, Q25PHCPU, Q25PRHCPU, Q26UDHCPU, Q26UDEHCPU, Q26UDVCPU, Q26UDPVCPU, Q50UDEHCPU, and Q100UDEHCPU
Process CPU	Generic term for Q02PHCPU, Q06PHCPU, Q12PHCPU, and Q25PHCPU
Universal model process CPU	Generic term for Q04UDPVCPU, Q06UDPVCPU, Q13UDPVCPU, and Q26UDPVCPU
Redundant CPU	Generic term for Q12PRHCPU and Q25PRHCPU
CPU module	Generic term for the Process CPU, Universal model process CPU, and Redundant CPU
ACPU	Generic term for the PLC CPU that can be used with MELSEC-A series
Redundant type extension base unit	Abbreviation for Q65WRB extension base unit for redundant system
CC-Link IE Controller Network	Abbreviation for CC-Link IE Controller Network system compatible with the Q series
MELSECNET/H	Abbreviation for MELSECNET/H network system compatible with the Q series
MELSECNET/10	Abbreviation for MELSECNET/10 network system compatible with the AnU, QnA/Q4AR
MELSECNET/10 compatible mode	Abbreviation for function and performance-compatible mode so that the MELSECNET/H network system can have upward compatibility to existing MELSECNET/10 network system
CC-Link IE Controller Network board	Abbreviation for CC-Link IE Controller Network interface board
MELSECNET/H board	Abbreviation for MELSECNET/H interface board
MELSECNET/10 board	Abbreviation for MELSECNET/10 interface board
Ethernet board	Generic term for Ethernet PC card and Ethernet interface board supported by Windows®
Personal computer	Generic term for IBM-PC/AT-compatible personal computer
Programming tool	Abbreviation for PX Developer programming tool
Monitor tool	Abbreviation for PX Developer monitor tool

(2) Terms

Term	Description
DDC	Abbreviation for Direct Digital Control A control of controller functions with a digital device.
FB	Abbreviation for Function Block A block with a specific function used in a program.
FBD	Abbreviation for Function Block Diagram defined in IEC61131-3 Programs are created by connecting variables, constants, and blocks containing specific processing, according to the flow of data signal.
ST	Abbreviation for Structured Text defined in IEC61131-3 Programs are created by writing arithmetic operations and logical operations in text format.
Project	Unit that gathers and manages a series of data necessary for configuration of FBD programs executed by the CPU module
Tag	Identification symbol attached to each DDC processing defined by JIS This can be likened to a tag attached to process control equipment.
Sequence control	Control that processes each control step according to preset order and procedures
Loop control	Control method that repeatedly executes processing of specific parts
Member	Basic data items in structure type data
Tag data	Data that data attached to DDC processing indicated with a tag (process condition data/process status data) is summarized Accessing the tag data can monitor status and set conditions of the relevant DDC.
Tag FB	Function block works as a controller or an indicator containing tag data
Module FB	Function block for inputting/outputting data of analog I/O module, digital I/O module, and high-speed counter module connected to the base unit on which the PLC is mounted or CC-Link field bus
Faceplate	Gauge window on which an indicator such as a controller is displayed in image format. Values assigned to tag data are manipulated.
System resource	PLC device required for executing FBD programs, used for automatically assigning variables (This cannot be used in ladder programs.)
Ladder program	Program method designed so that contact sequence can be applied to PLC language Draw two vertical control bus lines and describe a contact between the buses for programming.
Identifier	Used for setting various element names (variable name, FB variable name, structure name, etc.) Some unusable characters cannot be used for the identifier.
Reserved word	Part names (such as VAR) that cannot be used as various element names (variable name, FB variable name, structure name, etc.)
Operation mode	Mode for determining the operation method of the redundant system The following three modes are available. • Backup mode • Separate mode • Debug mode
Backup mode	Mode for normal operation of the redundant system If a failure or an error occurs in the control system, the standby system switches to the control system to continue the control of the redundant system. The operation mode can be switched to the separate mode using GX application.
Separate mode	Mode for maintaining a system (partial modification of a program, replacement of modules mounted on the main base unit) without stopping the control during run of the redundant system During this mode, different programs can be executed in the control system and standby system. System switching cannot be made in this mode (User switching is possible). The operation mode can be switched to the backup mode using GX application.
Debug mode	Mode for performing a debug using a single system prior to redundant system operation This permits operations without connecting tracking cables. In this mode, the CPU module is fixed to system A, control system. (Tracking of the redundant system is not performed.) Set/cancel this mode in the redundant parameter setting of GX application.
Operation mode change	Switching of the operation mode for system A and system B using GX application while the redundant system is running The operation mode can be switched between the backup mode and separate mode.
System A	System to which system A connector for tracking cable is connected in the redundant system
System B	System to which system B connector for tracking cable is connected in the redundant system

Term	Description
System switching System switching User switching	Control switching to backup system to continue system control and network communication when a trouble occurs in the system that performs control in the redundant system (when a failure or an error occurs in the power supply system, mounted module, or network) (Switching between control system and standby system to avoid system down) The following two types are available. • System switching Automatic system switching by the redundant system when a trouble occurs • User switching System switching by sequence program/GX application
Control system	A system that performs program operation, system control, and network communication in the redundant system When system A and system B start concurrently in the backup mode, the system A will be the control system (Concurrent startup: One system starts within three seconds after the other system has started.) When the system A and system B start separately, a system that starts first will be the control system.
Standby system	Backup system to continue system control in case of a failure or an error in the module in the control system in the redundant system (The CPU module in the standby system does not calculate programs.) When system A and system B start concurrently in the backup mode, the system B will be the standby system. (Concurrent startup: One system starts within three seconds after the other system has started.) When the system A and system B start separately, a system that starts later will be the standby system.
Tracking transfer function	Data transfer function that keeps the data of control system and standby system consistent This function enables the standby system to serve as the control system to continue the system control in case of system down of the control system. The Redundant CPU can perform tracking transfer without making the tracking settings, as it tracking transfer setting data has been set by default. (Change tracking transfer setting data using GX application.)
Redundant system	System configured using Redundant CPUs This system consists of two basic systems including CPU modules, power supply modules, and network modules. (If module error occurs in one system, the other system continues the system control. Thus, system reliability is improved.) To configure the redundant system, prepare two sets of the systems where the above modules of the same models are mounted on the base unit, and connect the CPU modules with tracking cables.
Redundant parameter	Parameter for setting operation mode of Redundant CPU system and tracking transfer setting data (tracking setting) Use GX application to set the parameter.

1

1 OVERVIEW

This manual covers some relative contents: the programming specification, function, instruction and programming method for programming with Function Block Diagram language (abbreviation: "FBD language") on PX Developer.

1.1 Features

The features of PX Developer are as follows:

(1) Enhance program productivity
It is more convenient to create DDC processing program by FBD language than by ladder program, which has been quite complicated. Therefore program productivity is enhanced. (*1)
*1 FBD language conforms to international standard specification IEC61131-3.

(2) Reduce the work-hour of creating DDC processing program by offering various parts.
PX Developer provides abundant function blocks (tag FB) for loop processing, which are loaded with CPU module dedicated instructions and tag data.
Creating FBD program with above-mentioned parts can reduce work-hour of DDC processing program.

(3) It can be used for creating user-defined FB
By combining various FB and functions, the individual FBs can be created according to their needs.

(4) Variables used by FBD program can be assigned to PLC device automatically.
Variables used by FBD program can be assigned to PLC device automatically, thus trivial device assigning work is saved.

(5) Compatibility with ladder diagram program
In the batch system that combines sequence control and loop control, ladder program applicable for sequence control processing description and FBD program that is easy to describe the loop control can be executed in one CPU module simultaneously.

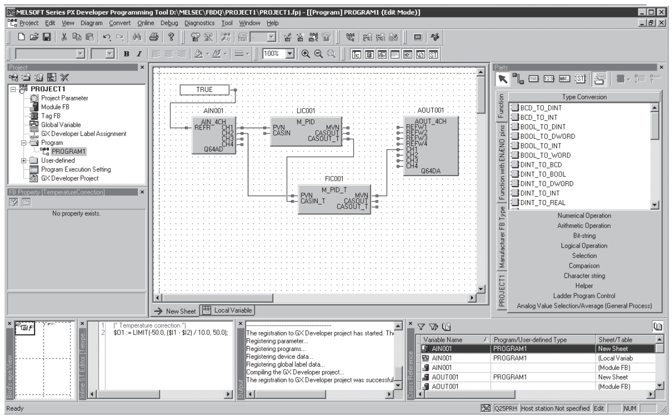
(6) Compatibility with Redundant CPU system
Programming applicable for Redundant CPU system is enabled.

1.2 Product Configuration

PX Developer consists of programming tool and monitor tool.

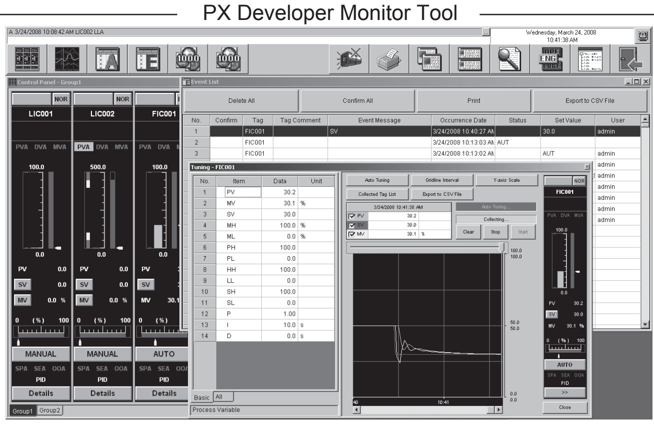
(1) Programming tool
The tool can be used for programming with FBD language (FBD program editing function), converting program edited by FBD into ladder program (compile function), as well as for monitoring and debugging.
For details about PX Developer programming tool, refer to the following manual.
- PX Developer Version 1 Operating Manual (Programming Tool)

PX Developer Programming Tool



(2) Monitor tool
The monitor tool can be used to monitor and control DDC processing that is being executed on CPU module (DDC monitor function).
For the details about PX Developer monitor tool, refer to the following manual.
- PX Developer Version 1 Operating Manual (Monitor Tool)

PX Developer Monitor Tool



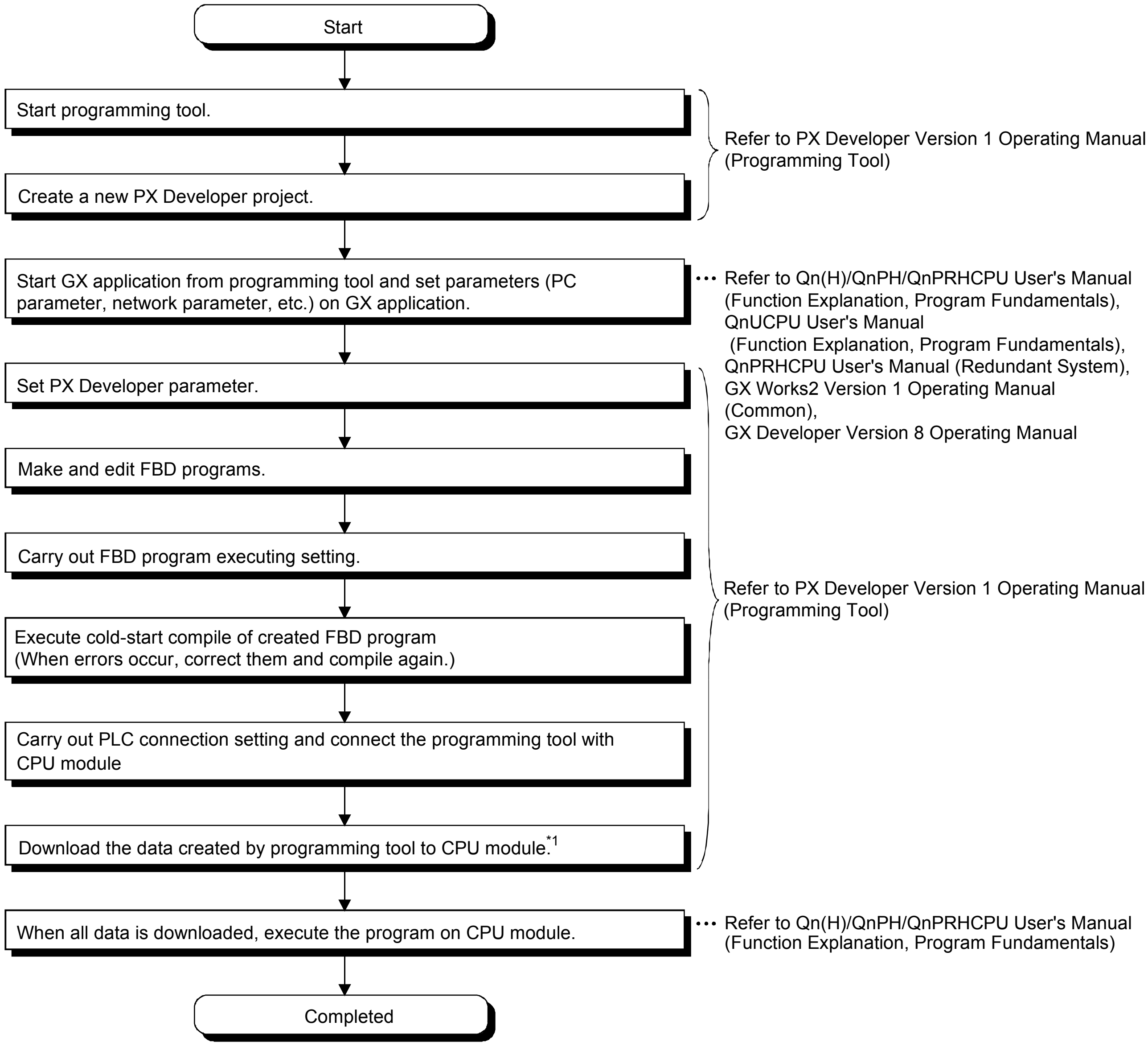
1.3 Engineering Flow

The section explains methods for creating FBD program by PX Developer and executing monitor of DDC processing.

1.3.1 Programming Procedure of FBD Program

The following paragraphs describe the sequence for executing FBD program creation and online monitor in using programming tool.

(1) Creating and executing a project



*1 When PLC download is performed with the programming tool, reload the monitor target project with the monitor tool.

(2) Project online monitor

Start

↓

Start programming tool.

↓

Open the PX Developer project executed as monitor target on the CPU module.

↓

Connect the programming tool with CPU module.

↓

Open various editing windows to start monitoring.

↓

Completed

Refer to PX Developer Version 1 Operating Maunal (Programming Tool)

1.3.2 Monitor Procedure of DDC Processing

The sequence of DDC processing by monitor tool is as follows:

Start

↓

Start monitor tool.

↓

Switch the monitor tool into engineering mode.

↓

Execute the monitor target project setting .

↓

Set various monitor if necessary.

↓

Carry out PLC connection setting and connect the monitor tool with CPU module.

↓

Perform test operation and make tuning.

↓

Switch the monitor tool into any mode.

↓

Open various monitor windows and monitor DDC processing.

↓

Completed

Refer to PX Developer Version 1 Operating Manual (Monitor Tool)

2 PROGRAMMING SPECIFICATION

This chapter explains how to use programming tool to make FBD programs.

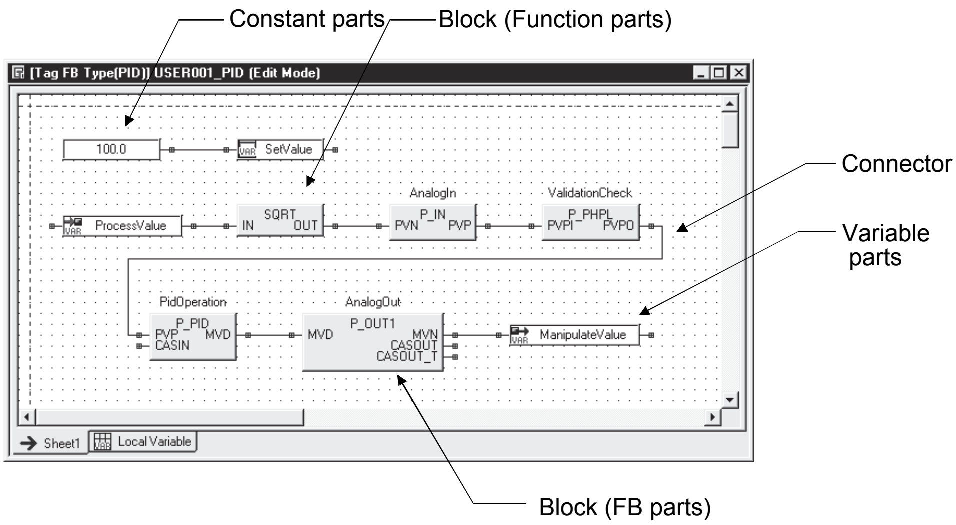
2.1 FBD Program

(1) What is FBD program

(a) Apply the FBD language specified in IEC61131-3 as the standard language for making program by programming tool.
FBD program is the program that uses FBD language.

(b) FBD language is a kind of graphical language, highly visualized and easy to understand.
Make program by connecting the blocks (including function parts, FB parts, and inline ST parts), variable parts and constant parts that are for special processing along the flow of data and signals.
Blocks can be reused and placed anywhere in FBD program, in addition, new blocks can be defined.

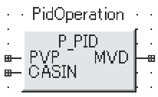
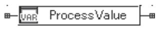
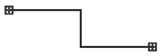
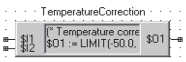
(Example) Program example for FBD program.



As showed in the above illustration of blocks connected by connectors, which seems quite like an electric circuit, actually, data flows from the output of blocks, variable parts or constant parts to the input of variable parts of other blocks.

(c) Parts which compose FBD programs are called FBD parts.
These parts can be used for programming.

(2) FBD parts
A FBD program consists of various FBD parts.
The program can be created by connecting FBD parts.
FBD parts described below.

FBD parts name	FBD parts graph	Description
Function parts		It indicates execution of function parts. Left pin is for input and right for output. Function part name is at the top center.
FB parts		It indicates execution of function parts. Left pin is for input, while right pin for output. Function part name is at the top center. Above the part is FB variable name. * If it is module FB, the FB module name is at the bottom center.
Variable parts		It indicates variable. Value is acquired and saved. Variable name is shown at the centre of the part.
Constant parts		It indicates constant part. Value or character string is directly set on the part. Value of the constant is shown at the center of the part.
Connector		It indicates the data flow. Used to connect parts. Data flows from left to right. Data types of the connected parts must be the same.
Comment parts		Any comment can be entered. This will not influence the execution code of compile result. (This will not affect FBD program)
Inline ST parts		It indicates execution of inline ST program. Left pin is for input, while right pin for output. Inline ST part name is displayed on the upper part of the part.

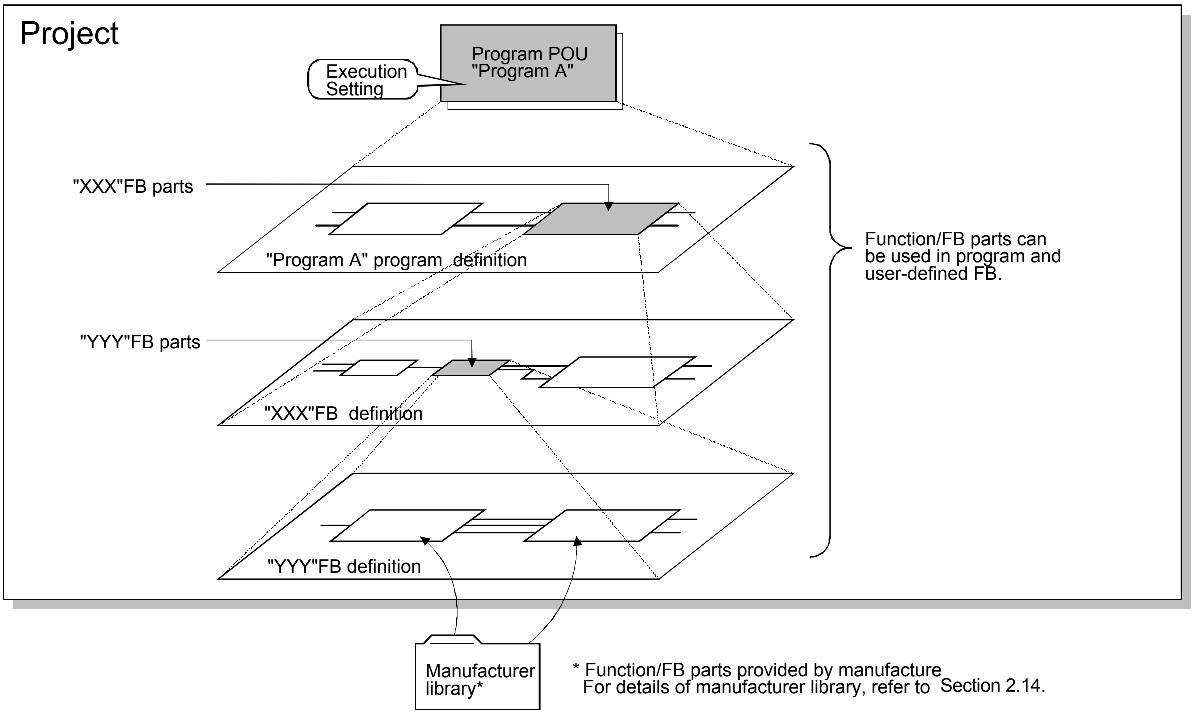
2.2 Configuration of FBD Program

This section explains on the configuration of FBD programs made by programming tool.

2.2.1 Program Organization Units

Elements that construct a FBD program are called Program Organization Unit (abbreviation: "POU").
POUs may be classified into three types: program, FB and function.

(1) The structured design of FBD program
A FBD program is actually a hierarchical structure of several POUs.



(a) When programming using FBD language for a certain processing on CPU module, create one or multiple FBD programs and define the processing contents in the FBD program. Define FBD program by combining FB parts, function parts, inline ST parts, etc.

(b) FB parts to be used in FBD program can be newly defined. In user-defined FB parts, the defined FB parts, function parts, etc. can be combined for use.

(c) In the hierarchical structure of FBD program, the lowest layer can be manufacturer function, FB or tag FB.

(d) Project manages all the user-defined elements (such as POU definition, structure definition and global variable) to convert the FBD program into ladder program that can be executed on CPU module.

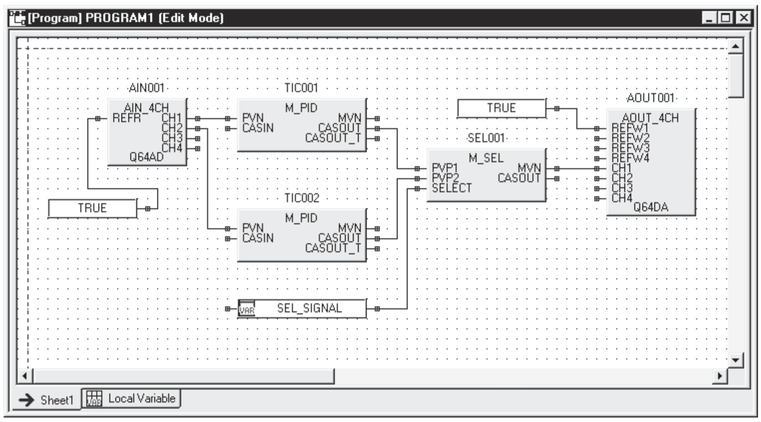
(2) Program

(a) Program is configured the processing which is executed in CPU module by combining functions, FBs, and inline STs that are explained later.
Program is at the top hierarchy of the parts configuring FBD programs.

(b) Program processes according to the executing conditions specified in Program Execution Setting of programming tool.

(c) The maximum number of programs for 1 project is 200.

(Example) Program definition window

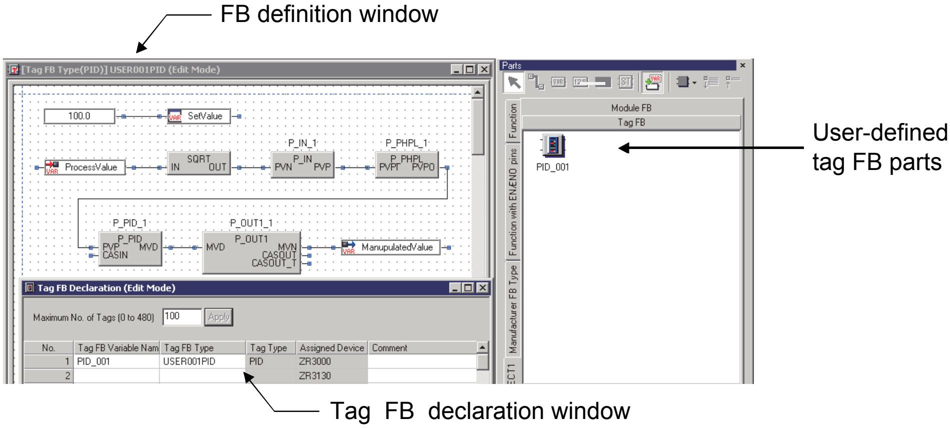


(3) FB

(a) With its own internal memory, FB conducts control processing according to the status of input and internal memory. It is the part that conducts control processing by using Program/User-defined FB parts.

(b) FB parts are used after being given variable names respectively.
FB parts with variable name can independently conduct control processing.

(c) There are two kinds of FB parts: FB to be newly defined and FB provided by manufacturers.
(Example) User-defined tag FB



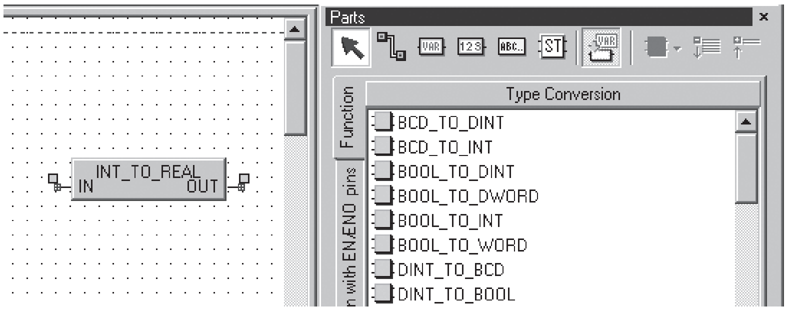
(4) Function

(a) Function is used to conduct certain control processing to the input. Program/User-defined FB parts conduct control processing by using function parts.

(b) Without internal memory, functions can only conduct a single processing to a single input. Function parts can operate independently without variable name.

(c) Function parts can only be provided by manufacturers. The function parts cannot be newly defined.

(Example) Function parts



(5) Inline ST

(a) Inline ST is used to conduct a control processing with such as conditional judgment and arithmetic operation by programming in text format. It is the part that conducts control processing by using Program/User-defined FB parts.

(b) Inline ST parts are used after being given names respectively. Inline ST parts with name can independently conduct control processing.

2.2.2 Definition of POU Interface

POU has both input variable interface and output variable interface.

(1) Input variable and output variable

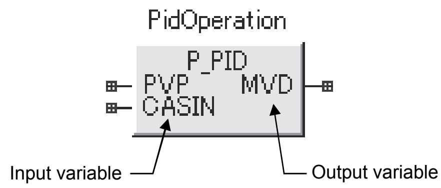
(a) Program

Program is at the highest POU hierarchy in a FBD program. With no data exchange transferred as parameters, it has no input variable and output variable.

(b) Function/FB

Function/FB has input variable as well as output variable. (However, there is also Function/FB with no input or output variable.)

- Input variable : The variables that receive data while Function/FB parts are conducting processing.
- Output variable : The variables that transfer the data as the function/FB parts processing.

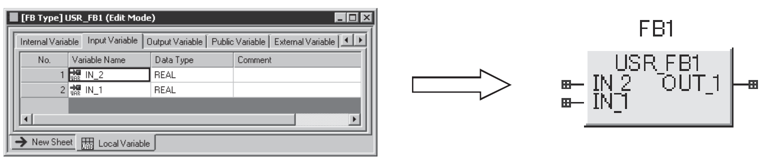


(2) Definition of input variable and output variable of user-defined FB/tag FB parts.
Define input and output variable in user-defined FB/tag FB parts.

(a) Input and output variable may define the variable parts with Input/Output variable type in FB Definition Window.

(b) The input variables and output variables inserted are automatically reflected on the local variable sheet in the corresponding order as the Input/Output pins on the user-defined FB/tag FB parts.

(Example) Input and output variable definition of user-defined FB part.



The arrangement of input/output pins on user-defined FB/tag FB parts is corresponding to the column order of input and output variables on the local variable sheet.

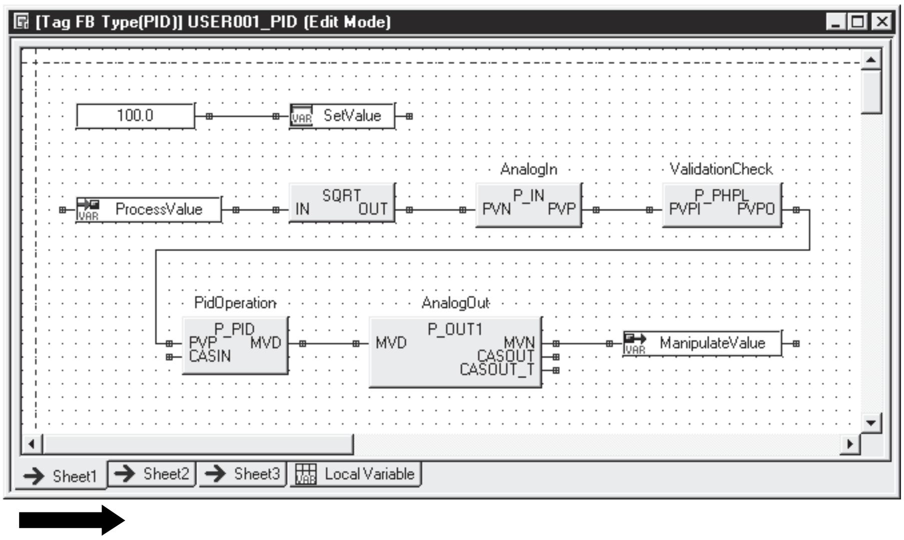
2.2.3 Definition of POU Processing Contents

POU processing contents can be defined through the creation of block diagram indicating processing actions on FBD sheets of Program/FB definition window.
POUs that can be newly defined are program and FB (including tag FB).

(1) Programming of POU definition
In the process of POU definition programming, different FBD parts: including function/FB parts, user-defined FB parts, and variable parts are inserted into a FBD sheet then are connected with connector according to control actions.

(2) FBD sheet
FBD sheet is an operation area where FBD parts are inserted and connected.
While describing POU definition in FBD language, up to 32 FBD sheets can be added so as to improve the visibility of the definition.
When more than one FBD sheets are used in a program, tag FBD sheets are executed one by one from left to right.

(Example) POU processing contents (user-defined FB)



POU processing is executed from the left tag FB sheet to the right tag FB sheet.

2.2.4 Relation with GX application

When GX project is started in PX Developer project, ladder program creation and various parameter settings can be performed with programming tool.
For more details on how to start GX application from programming tool, refer to "PX Developer Version 1 Operating Manual (Programming Tool)".
The relation of FBD program in a PX Developer project with the user-created ladder program is as follows.

(1) About ladder programming

(a) Ladder program (called user-created ladder hereafter) in GX application can be used to describe the processing that is difficult to be described in a FBD program (such as interlock processing).

(b) Through GX label assignment setting in programming tool, the global variable in a FBD program can be used as global label of GX application. This allows the user to program with the variables of FBD programs on a user ladder without paying attention to devices.
For details on GX label assignment, refer to "PX Developer Version 1 Operating Manual (Programming Tool)".

GX Works2 Label Assignment (Edit Mode)

No.	GX Works2 Global Label	PX Developer Global Variable Name	Comment
1	LIC001_SV	LIC001_SV	Tag FB (LIC001) SV
2	TIC001_SV	TIC001_SV	Tag FB (TIC001) SV
3	PID001_SV	PID001_SV	Tag FB (PID001) SV

IMPORTANT

The QDRSET(P) instruction (setting of file for file register) must not be included in the user ladder. If included, FBD program will not normally operate when the file for file register is renamed by the QDRSET(P) instruction.

(2) Download to PLC

Please use PLC download function of programming tool when downloading user-created ladder or setting parameters compiled in GX application.
For details on the operation methods of PLC download of programming tool, refer to "PX Developer Version 1 Operating Manual (Programming Tool)".

2.2.5 Compiling FBD Program

FBD program can be compiled with programming tool, then transferred into codes that can be executed on CPU module (ladder program, PLC parameter etc.)
There are three methods of compile: cold-start compile and hot-start compile, and on-line change compile.
For more details on various compile methods and their functions please refer to "PX Developer Version 1 Operating Manual (Programming Tool)".

(1) Cold-start compile
Cold-start compile is a compile process which reassigns all the assigned devices of the currently existed variables from the very beginning. (All the variable values are changed into initial values.)
When compiling FBD program, cold-start compile is executed first.
Additionally, when executing PLC download after cold-start compile, CPU module is in STOP mode. Under this mode, FBD programs and user ladders stop executing; all the outputs (Y) are OFF, while analog output can be retained. (Module side setting is possible)

(2) Hot-start compile
Hot-start compile is a compile process without changing the assigned devices of the currently existed variables. (The variable value will be kept)
During compile, current status is kept, this kind of compile can be used to make changes additions to FBD programs.
Additionally, when executing PLC download after hot-start compile, CPU module is in PAUSE mode. Under this mode, FBD programs and user ladders stop executing; output (Y) will remain the previous status, and analog output can be retained. (Module side setting is possible)

(3) Online change compile

Online change compile is to compile without changing the assigned devices of present variables, and to download the project during RUN without stopping or pausing CPU module.

Online change compile is mainly used in the occasion in which it is wanted to use processing such as FBD program modification/addition with no need to stop system. (Like hot-start compile, the variable value will be kept.)

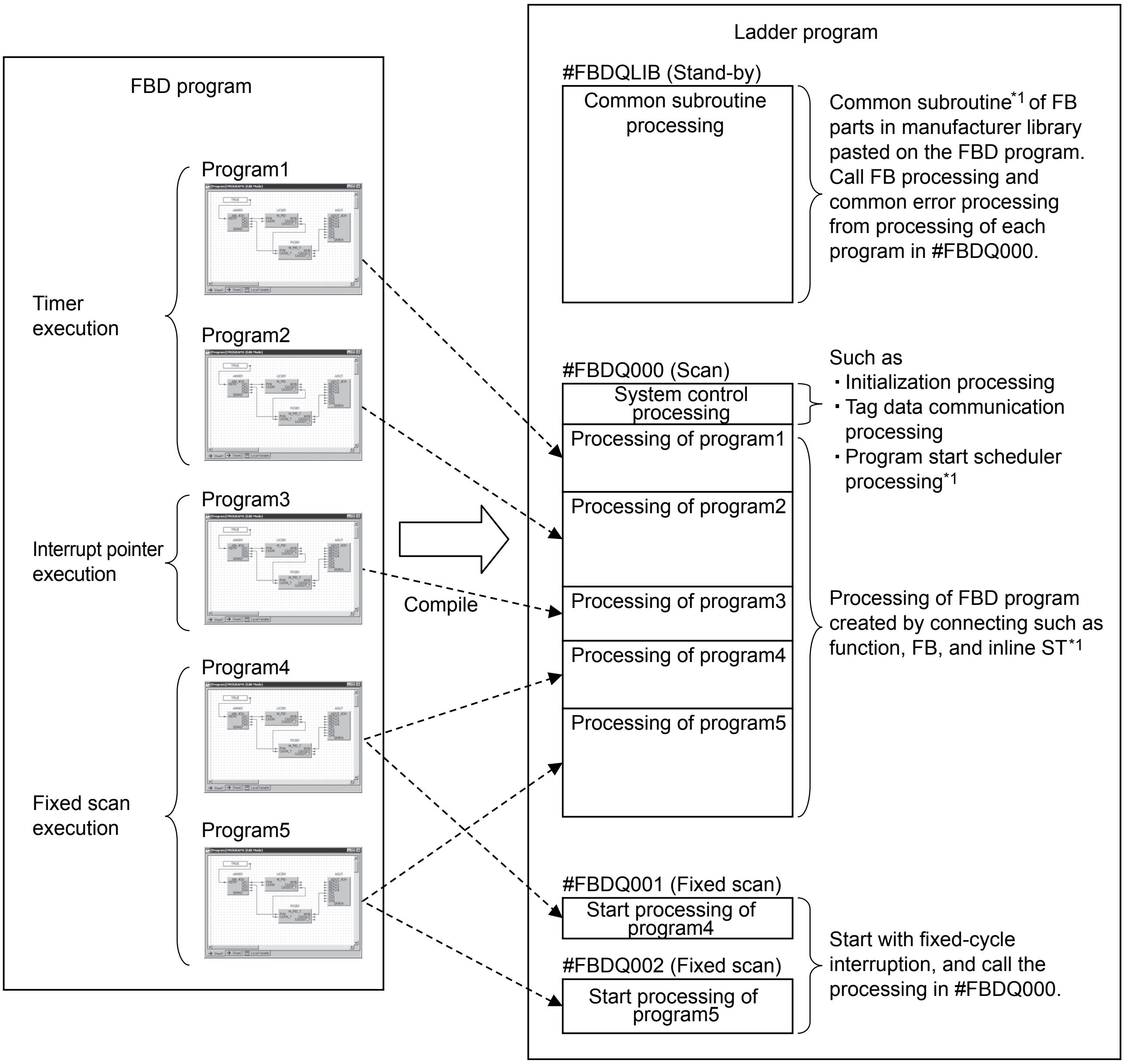
POINT

- In the case of hot-start compile and online change compile, do not change the file register setting in PLC parameter.
  To change the setting of file register in PLC parameter, PC download cannot be executed after hot-start compile and online change compile.
- When executing online change, the scan time will be prolonged as follows, pay attention to this.

Item	PX Developer → Write during CPU module RUN
When blank area can be reserved on program memory	The maximum scan prolonged time (ms) = 4.0 × (k step number of #FBDQ000) + 0.8 However, 97ms will be the maximum scan prolonged time if the calculated time is less than 97ms.
When blank area can be reserved on memory card (excluding ATA card*1)	The maximum scan prolonged time (ms) = 5.1 × (k step number of #FBDQ000) + 0.8 However, 97ms will be the maximum scan prolonged time if the calculated time is less than 97ms.

*1 In the case of using ATA card, the scan time per 30k step will be prolonged by 1.25s.
Therefore, it is suggested to use SRAM card instead of ATA card in online change.

(4) Converting FBD program into ladder program by compilation
Compiling FBD program with Programming Tool converts into ladder program of common subroutine processing, system control processing and FBD program processing.
The following explains the relation between FBD program and ladder program executed in process CPU, Universal model process CPU, or Redundant CPU.

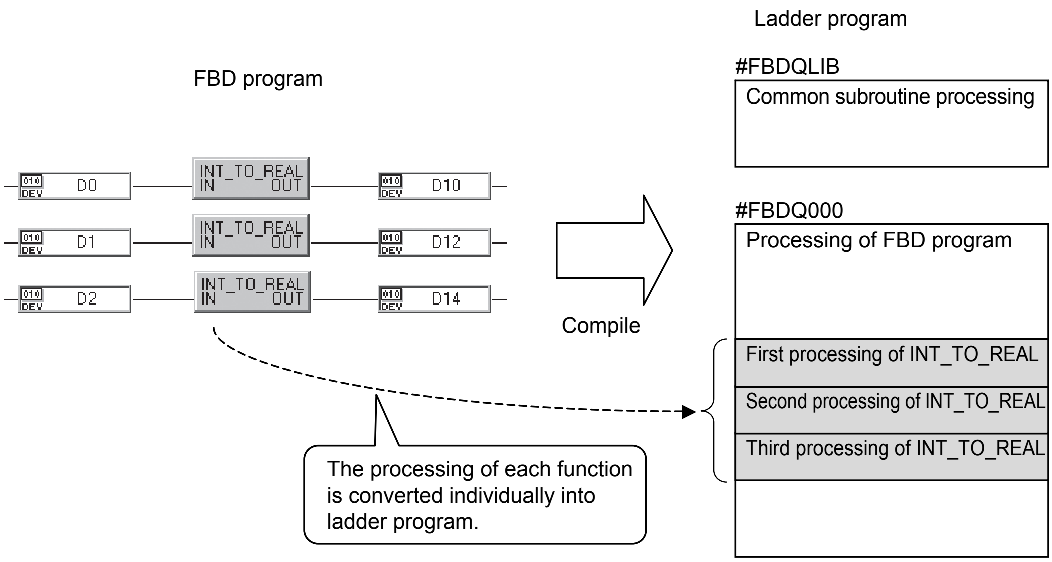


*1 For details of approximate number of steps for the system control processing and functions/FBs in manufacturer library, refer to Appendix 4.

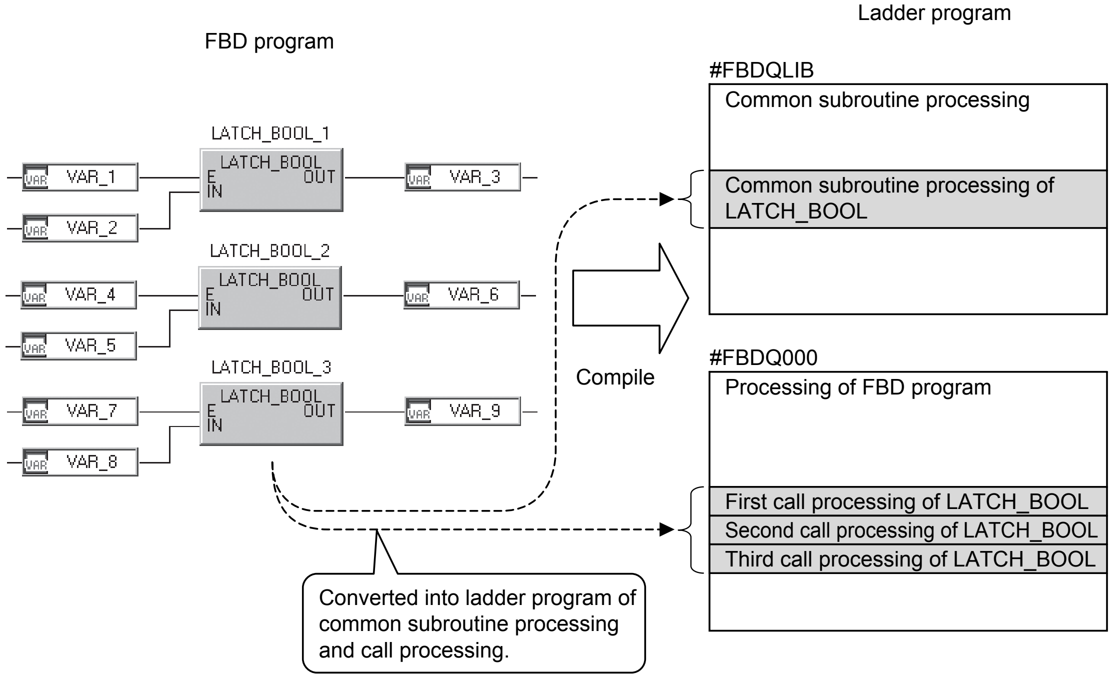
When compiling a FBD program in Programming Tool, the conversion method into a ladder program differs depending on a type of FBD part.
The following describes how a FBD part is converted into a ladder program in compilation.

Type of FBD part	Conversion method
Manufacturer function	Converted into ladder programs individually at the positions where FBD parts are used.
Manufacturer FB/ Tag FB, Module FB	Converted into one common subroutine processing and call processing. For example, when the same type of manufacturer FBs are used at the multiple positions, converted into one common subroutine processing and call processing for the amount of positions to be used.
User-defined FB/Tag FB	Converted into ladder programs individually at the positions where FBD parts are used.
Inline ST	Converted into ladder programs individually at the positions where FBD parts are used.

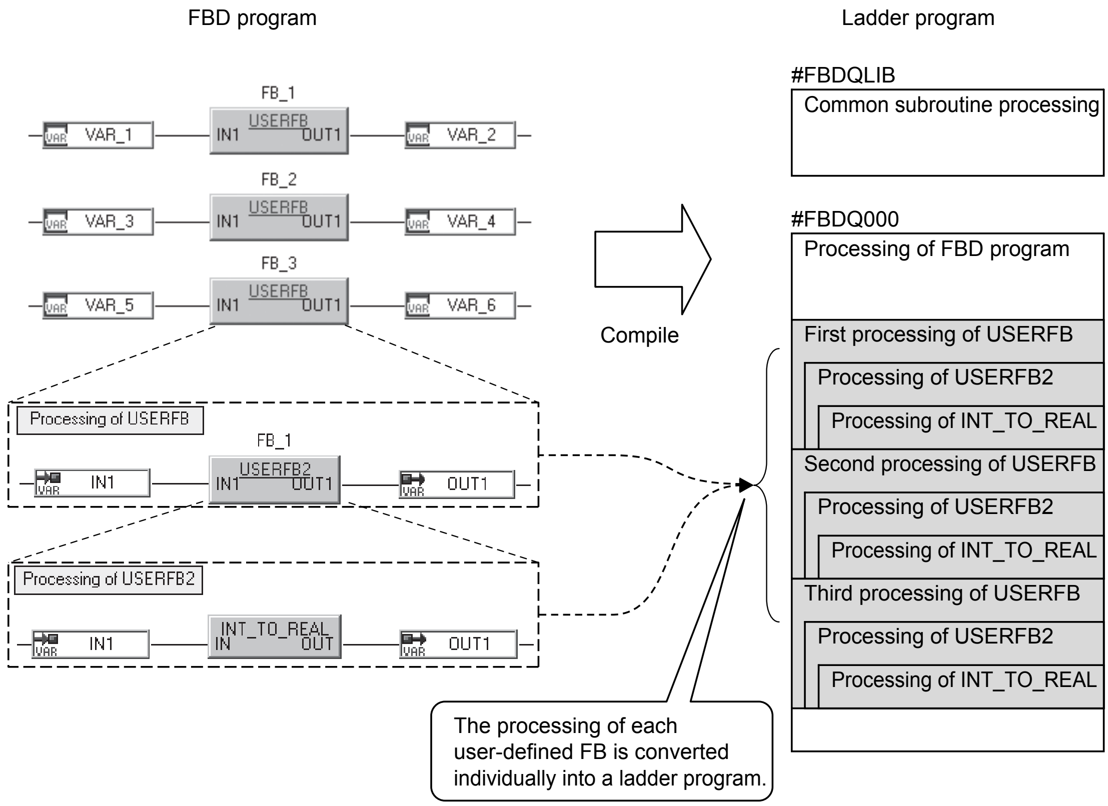
- Manufacturer function
  Manufacturer functions are converted individually into ladder programs as shown below example diagram.



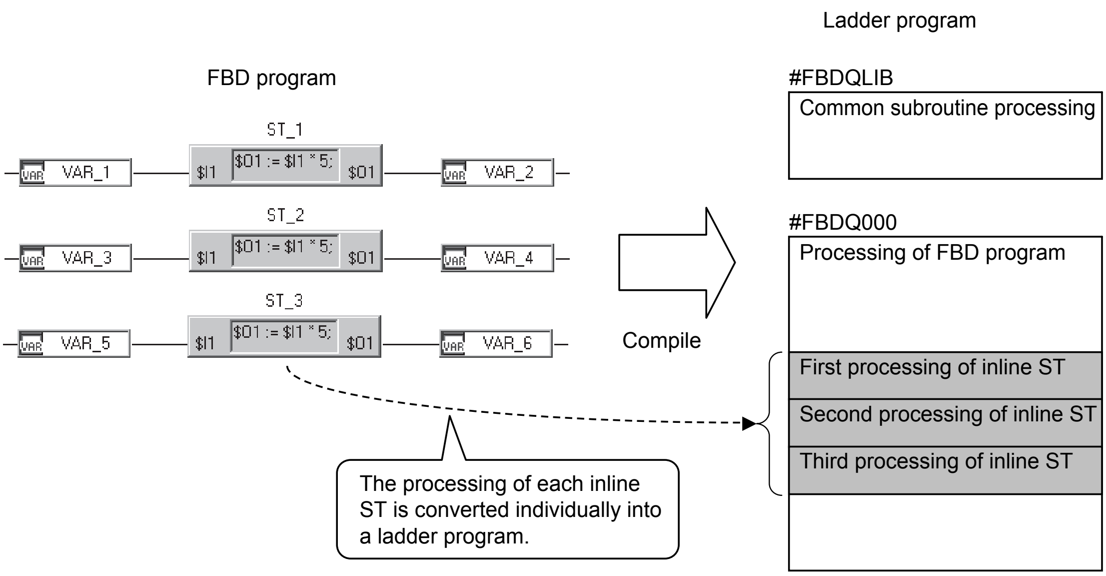
- Manufacturer FB/Tag FB, Module FB
  Manufacturer FB/Tag FB, Module FB are converted into ladder programs of common subroutine processing and call processing as shown below example diagram.



- User-defined FB/Tag FB
  User-defined FB/tag FB are converted individually into ladder programs as shown below example diagram.



- Inline ST
  Inline STs are converted individually into ladder programs as shown below example diagram.



2.2.6 When Power Supply Is OFF → ON or Doing the Reset Operation

Variables are assigned to the file register by programming tool, so their values will remain unchanged when switching power supply from OFF to ON and executing reset operation.
Therefore, to initialize variables and restart CPU module, perform cold-start compile mentioned in Section 2.2.5 (1) then RUN CPU module after PLC download.

2.3 Variable

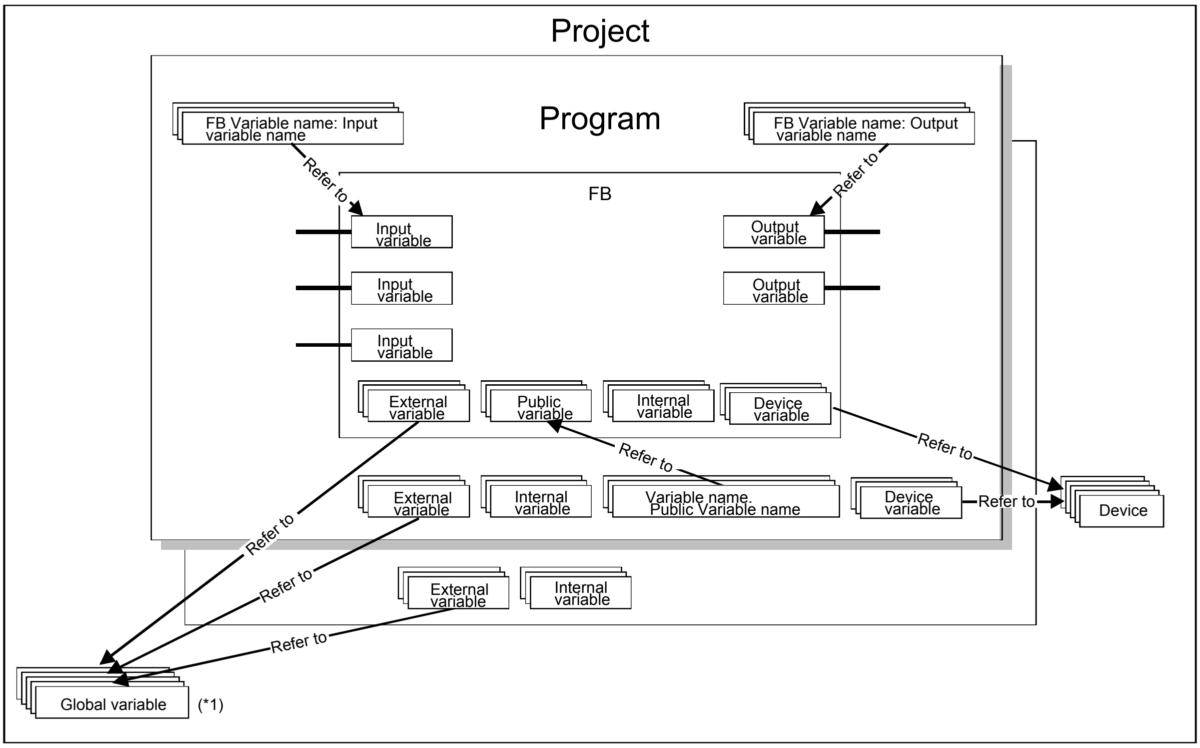
Variable is an area where various values are stored.
Data type of a variable must be declared before its value is operated.

2.3.1 Local Variable and Global Variable

Variables can be classified into local variables, which are dedicated to the use of each POU; and global variables, which can be used publicly among several POUs.
The contents about local and global variables are as follows.

Item	Local variable		Global variable
	(Other than device variable)	(Device variable)	
Variable declaration	Declare through defining variable name and data type of variable parts. The declaration is shown on local variable sheet.		Declare through defining global variable name and data type in Global Variable Declaration Window of programming tool.
Available data type	Elementary data type, structure type.	Elementary data type	Elementary data type, structure type.
Number of declaration available in a project	Limited by the setting range of file register (ZR) assigned to the variable.	Limited by the device setting range of PLC parameter.	Up to 32000 declarations can be made. But the number is limited by the setting range of file register (ZR) assigned to the variable.
Initial value setting	'0' is stored. (In case of character string, null (" ") is stored.) Initial value cannot be set. However, initial value of public variable can only be set in FB property window.	Initial value cannot be set.	Initial value setting is allowed for elementary data type. Without initial value setting, '0' is stored. (In case of character string, null (" ") is stored.) Initial value setting is not allowed for structure type.
Assignment device setting	Devices cannot be assigned. Devices are automatically assigned by cold-start compile (with different devices each time).	Device with variable name is assigned automatically	Devices can be assigned when it is elementary date type or member of structure type. Intelligent function module (U○\G○) and link direct device (J○\○○) can also be specified as assigned device when the variable is not string type. Devices are automatically assigned by cold-start compile (with different devices each time) if not assigned.
Use of variable	Variables can be used when variable parts stated as local variable are connected.		Variable parts are assigned in FBD sheet. Variables can be used through defining external variable with the same name and data type as global variable.
Variable definition modification	The modification is made through editing local variable sheet, and reflected on all the variables with the same name in the same Program/FB Definition Window.		The modification is made through editing Global Variable Declaration Window. External variables with reference to the new global variable must be changed accordingly.

(1) Relation between local variable and global variable
The relation between local variable and global variable is shown below.



*1 The global variable value in program/FB is applied through external variable having the same name with the global variable. While using programming tools, external variable will automatically increase if global variable parts are dragged & dropped from parts window to FBD sheet.

(2) Local variable
Local variables are the variables stated in each FBD program (program, user-defined FB type/tag FB type) and used only in this program.
Local variables consist of the following variable types.

Variable type	Description
Internal variable	The variable can only be used in the program and user-defined FB type/tag FB type which the internal variable is declared. External program cannot access it. Stored data will be retained as internal memory.
Input variable	The variable can be used as input value (input pin of user-defined FB type/tag FB type) in user-defined FB type/Tag FB type. The other part is same as public variable.
Output variable	The variable can be used as output value (output pin of user-defined FB/tag FB) in user-defined FB type/tag FB type. The other part is the same as public variable.
Public variable	All the variables inside FB/tag FB can be accessed by FB/tag FB and the nearest outer POU. Stored data will be retained as internal memory.
External variable	The variable used in program and user-defined FB type/tag FB type which the external variable is declared (with reference to global variable having the same variable name). Data type of the external variable and the global variable to which the external variable has reference must be matched. With no data memory, for it is only a reference to global variable.
Device variable	A variable which reads/writes the PLC device values. Use a device name as a variable name. Declare device variables for each FBD which uses the device.

FBD program type and data type that local variables can operate on varies with types as following table.

Variable type	FBD program type		Data type of variables
	Program	User-defined FB type/ tag FB type	
Internal variable	○	○	Elementary data type, structure type, FB type (except tag/module FB type)
Input variable	×	○	Elementary data type, structure type
Output variable	×	○	Elementary data type, structure type
Public variable	×	○	Elementary data type, structure type
External variable	○	○	Elementary data type, structure type, tag/module FB type
Device variable	○	○	Elementary data type

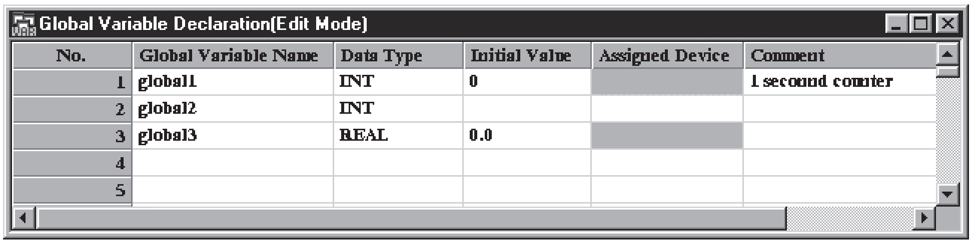
○: Available ×: Not available

(3) Global variable
Global variables are the variables declared in global variable declaration window of programming tool, to which all the FBD programs in PX Developer can be referred.
Each FBD program being referred to global variable through external variable of local variable statement.
With global variable, data can be exchanged with other different FBD programs.
Up to 32000 global variables can be defined.

(4) Device assignment of variable

(a) The data operated by local variable and global variable is stored in the file register of CPU module.
The stored devices are automatically assigned during cold-start compile by programming tool (Devices assigned are different each time).
During hot-start compile or online change compile, the assigned devices of the currently existed variables are not changed. Devices will only be assigned to those newly added variables.

(b) When declaring the global variables, the assigned devices can also be specified.
Through device assignment specification, the user may read/write in devices using global variables.
Device assignment of global variables is conducted in Global Variable Declaration Window of programming tool.


Global Variable Declaration(Edit Mode)

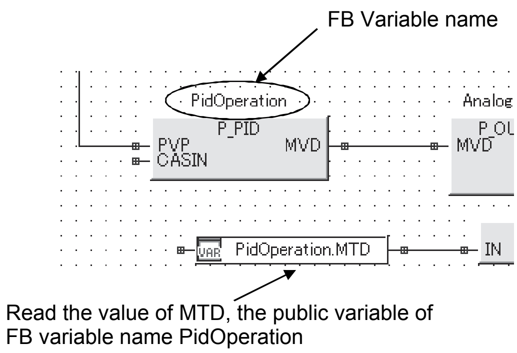
No.	Global Variable Name	Data Type	Initial Value	Assigned Device	Comment
1	global1	INT	0		1 second counter
2	global2	INT			
3	global3	REAL	0.0		
4					
5					

(5) Access to data in FB parts in reference operator

All the internal data of each FB parts, input variable, output variable and public variable can be accessed from the nearest outer POU using reference operator symbols (.).
While accessing input variable, output variable and public variable with reference operator symbols, follow the methods below.

(a) Specify input variable, output variable and public variable of FB parts

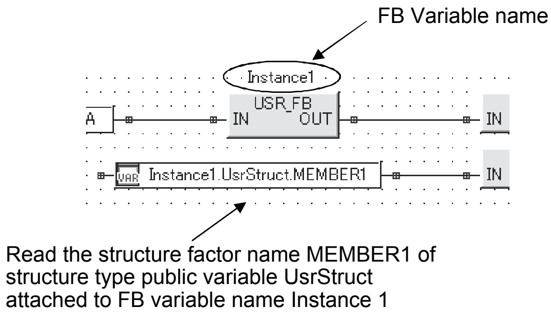
[Specified form]	'FB variable name'. 'Input variable name' 'FB variable name'. 'Output variable name' 'FB variable name'. 'Public variable name' For FB variable name, please specify declared FB variable name, to the I/O variable and public variable which have been read out.
[Specified example]	PidOperation.MTD



(b) When input variable, output variable and public variable of a FB part belong to structure type, refer to Section 2.5 for relevant details.

[Specified form] 'FB variable name' . 'Structure type input variable name'. 'Structure member name'
'FB variable name' . 'Structure type output variable name'.'Structure member name'
'FB variable name' . 'Structure type public variable name' .'Structure member name'
For 'FB variable name', specify FB variable name that makes declaration to the objects read out.

[Specified example] Instance1.UsrStruct.MEMBER1



(6) Variable initial value setting

The initial value is set through programming tool when executing PLC download on CPU module.

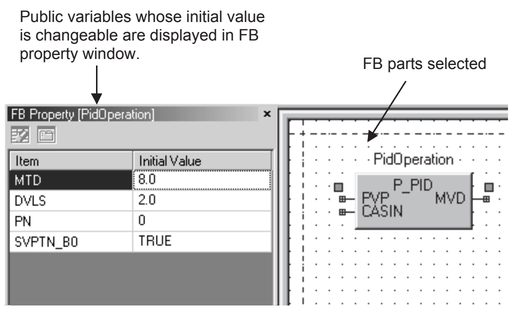
During cold-start compile, hot-start compile or online change compile of programming tool, the file registers are used by assignment target device of automatically assigned device. Therefore, variable values will remain unchanged after power off and reset operation. (Even latch clear operation cannot initialize the values.)

To change variable initial value, use programming tool. Then start cold-start compile to execute PLC download on CPU module.

Initial values of variables of different types are shown in the following table.

Variable type		Description
Local variable	Internal variable, input variable, output variable	'0' is stored. (In case of character string, null (" ") is stored.)
	Public variable	Each FB/tag FB stores its own initial value. (Initial value cannot be changed when public variables are of structure type.) Initial values can be changed in FB Property Window.
Global variable		The initial value or '0' is stored. *1 (In case of character string, null (" ") is stored.)

*1 When an assigned device is set, initialization will not be executed.



(7) Write forbidden device
The file register (R) cannot be used with the programming tool. Use the ZR device.
The file register (R) can be used in user ladders.
However, when using the file register (R) in a user ladder, do not use the file register (ZR) in the range set by the system resource of the project parameter of the programming tool.
In addition, the following devices can be read /written during program execution on programming tool.
Do not change the value of these devices from global variable or user ladder.

Devices used in programming tool	Change-forbidden range of device value
ZR (or R)	Range set with the system resource in the project parameter setting*1. (However, the items of tag data*2 within the range can be changed by specifying with ZR.)
T	Range set with the system resource in the project parameter setting*1
P	P3500 to P4095
M	Range set with the system resource in the project parameter setting*1
Z	Z0 to Z6 (However, the device value can be changed when the check box of "High speed execution" is cleared in "interrupt program/Fixed scan program setting" of PLC parameter of GX application.) *3
SD	SD0 to SD3 SD5 to SD8 SD16 to SD19 SD203 SD1500 to SD1501 SD1502 to SD1505
SM	SM1 SM390 SM701 SM1500, SM1501, SM1552 to 1583

*1 Refer to "PX Developer Version 1 Operating Manual (Programming Tool)".
*2 The device assigned to tag data can be checked from the tag FB declaration window of the programming tool.
*3 For details, refer to Section 2.15.3 (3).

2.4 Elementary Data Type

The elementary data types that can be applied in program tool are shown as follows:

Data type	Description	Range
INT	16 bits integer with sign	-32768 to 32767
DINT	32 bits integer with sign	-2147483648 to 2147483647
REAL	32 bits real number (single precision floating decimal)	$\pm1.17549^{-38}$ to $\pm3.40282^{+38}$,0
STRING	Variable length character string	0 to 255 bytes
BOOL	1 bit data	TRUE, FALSE
WORD	16 bits data	0_H to $FFFF_H$
DWORD	32 bits data	0_H to $FFFFFFFF_H$
ADR_REAL	Applied in tracking with cascade connection	—

REMARK

The variables of the elementary data types occupy the following memory capacity:

1. INT type and WORD type
   1 word.
2. DINT type, DWORD type, REAL type, and ADR_REAL type
   2 continuous words.
3. STRING type
   N continuous words.
   N is the number of ((the maximum storage character string length of STRING type variable +1*1) ÷2). (Round off the numbers at the right side of the decimal point)
   *1 +1 represents NULL code addition.
4. BOOL type
   Store the bit specification of word device as 1 bit.
   (Example: ZR5012.2)

POINT

For REAL type, an information loss error of single-precision floating-point operation may occur in the integration operation (current value + integration value up to the previous time).
"Information error" is an error that is caused by rounding (round down/round up) the lower digit of extremely small value when adding an extremely small value to an extremely large value. Generally, it occurs in computer systems that execute a floating-point operation. Real numbers in a PLC are represented by single-precision floating-point number.
The number of significant digits of this real number is approximately six to seven digits (when represented in decimal). Therefore, when the following real number operation is carried out, an error occurs in the operation result.

(Example of information loss on single-precision floating-point operation)
0.013333 + 32768.0 = 32768.013333 → 32768.012
↑ Current value ↑ Integration value up to the previous time
As indicated above, the logical operation result is 32768.013333; however, the number is rounded (round down) to 32768.012. As a result, the original increment of 0.013333 becomes 0.012 so that the increment amount reduces. Additionally, the number of significant digits of decimal part decreases as the number of integer digits of integration value increases.

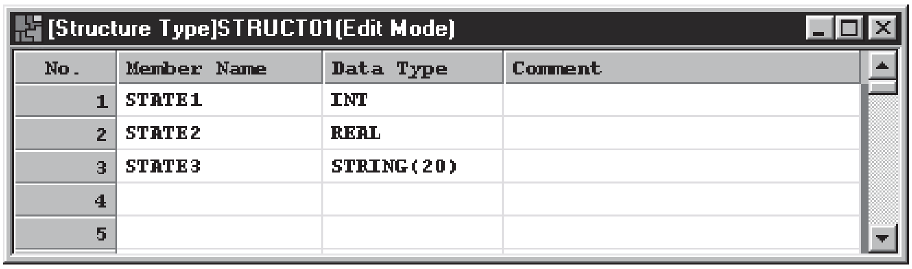
> **POINT**
> In REAL type operation, the operation result may not be exactly the same between Process CPU/Redundant CPU and Universal model process CPU.
> When changing PLC type from Process CPU or Redundant CPU to Universal model process CPU, make sure that there is no problem in the actual system.

2.5 Structure Type

Structure type can merge maximum 255 elementary data type variables as members, and is used for merging variables of relevance.
Structure type handles variables of different elementary data types.
However, only basic data types can be declared as data types. Structure type and FB type cannot be declared as data type.

(1) Definition of structure type.
Define the structure type members in structure type definition window.
[Start procedure]
Select "User-defined" → "Structure Type" In project window.
For setting methods, please refer to "PX Developer Version 1 Operating Manual (Programming Tool)".

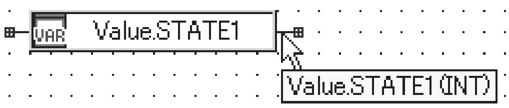
[Setting window]
(Example) The following is a case that 3 members are defined in structure type STRUCT01.


[Structure Type]STRUCT01(Edit Mode)

No.	Member Name	Data Type	Comment
1	STATE1	INT	
2	STATE2	REAL	
3	STATE3	STRING(20)	
4			
5			

(2) Application of structure type member.
When structure type members are applied, reference operator (.) and member names should be attached after structure type variable name.
(Example) When applying the member [STATE1] of structure type [STRUCT01].

[Specified format]	'Structure Type Variable Name'. 'Member Name'
[Specified example]	Value.STATE1
	Select STRUCT01 for data type



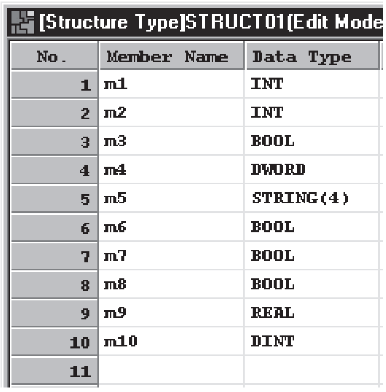
REMARK

The memory structure of structure type is as follows:

- The members of structure type are assigned in the continuous area of word devices orderly.
- The members of elementary data type except BOOL type are assigned in the same structure as shown in the REMARK in Section 2.4.
  The member of BOOL type is stored in word device as 1 bit unit.
  However, in case of continuous definition of BOOL type members, the maximum storage, beginning from the lowest bit unit, is 16 units.

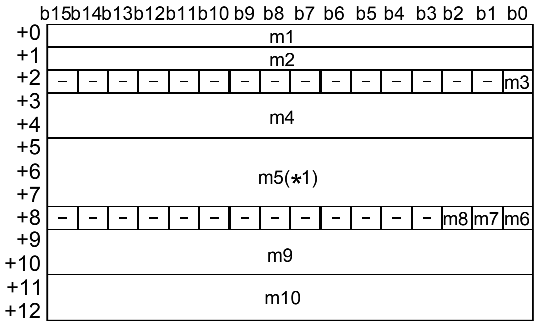
(Example) Definition window of structure type and memory structure

(Definition window of structure type)


[Structure Type]STRUCT01[Edit Mode

No.	Member Name	Data Type
1	m1	INT
2	m2	INT
3	m3	BOOL
4	m4	DWORD
5	m5	STRING(4)
6	m6	BOOL
7	m7	BOOL
8	m8	BOOL
9	m9	REAL
10	m10	DINT
11		

(Memory structure)



*1 Store 4 bytes of STRING type and NULL code.

2.6 Constant

2.6.1 Constant Format

Constant does not have particular data type which the value is input itself.
The data type of constant is specified by the input variable data type of function part/FB part that is connected to the constant part by connector.
The elementary data types corresponding to constant input formats are described in the following table.

Input format *1	INT	DINT	REAL	WORD	DWORD	STRING	BOOL *2	Display format example
Character string	—	—	—	—	—	○	—	"ABC"
Decimal integer	○	○	○	○	○	—	○	100
Hexadecimal integer number	○	○	—	○	○	—	○	H123A
Real number	—	—	○	—	—	—	—	(Displayed in radix point) 100.0 (Displayed in exponential form) 2E+010
TRUE/FALSE	—	—	—	—	—	—	○	TRUE

○: Applicable —: Not applicable

*1 Input format is as follows.

Character string	:	The character string within (" ") that is no more than 32 characters. However, the following characters cannot be used in the character string. ● double quotation marks (") ● Comma (,) ● Horizontal tab
Decimal integer number	:	The value that consists of signs (+, -) and numbers.
Hexadecimal integer number	:	The value that begins with "H" and consists of numbers and "A" to "F".
Real number	:	The value that is displayed with radix point (Ex: 100.0) or exponent (Ex: 2E+010).
Truth/False	:	"TRUE" or "FALSE"

*2 Input TRUE and FALSE of BOOL type by following methods

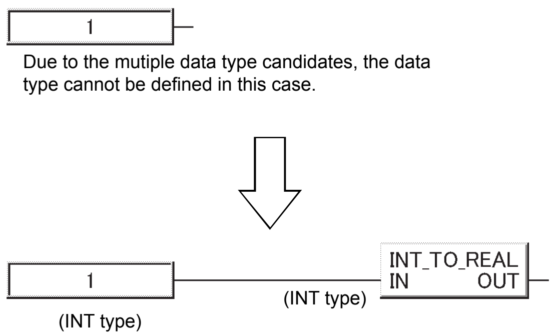
In the case of decimal integer number	:	0 : FALSE, 1 : TRUE
In the case of hexadecimal integer number	:	H0: FALSE, H1: TRUE

2.6.2 Constant Data Type

In the case of constant parts, the data type of constant value is not defined when inputting the constant value but defined when connecting constant part and FBD part by connector.
The data type of constant value is the same as that of the connecting FBD part by connector.

(Example) When 1 is input into constant value

For example, if 1 is input to constant value, there are 6 possibilities of candidate data types, they are, INT, DINT, WORD, DWORD, REAL, and BOOL type. So the data type cannot be decided under such circumstance.
The connection of constant part with FBD part by connector makes the data type of FBD part (connection target) input pins.



When connector connects constant part with FBD part, the data type can be decided.
(Becomes the data type of the input pins of the connected FBD part.)

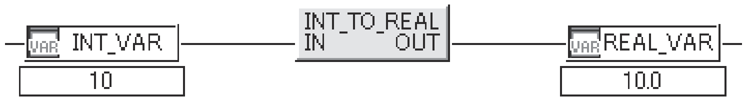
2.7 Function

2.7.1 Function

Functions perform the same operation on the input values of input variables, and output the results through output variables. Functions do not have internal memory.
Functions include the following types:

- Overload function
- Input pin number changeable function
- Function with EN/ENO pins

(Example) In case of type conversion function (INT_TO_REAL)
When input 10 by INT type, output 10.0 by REAL type.



Input value: 10 (INT type)

Output value: 10.0 (REAL type)

POINT

- Functions must connect all of their input pins to other FBD parts. (Except BIND (_E), CALL_DINT (_E), CALL_REAL (_E))
  If the input pins are not connected, compile errors will occur.
- If the data type of input variable and output variable of function has been defined, the FBD parts connected to input and output pins must correspond to it.

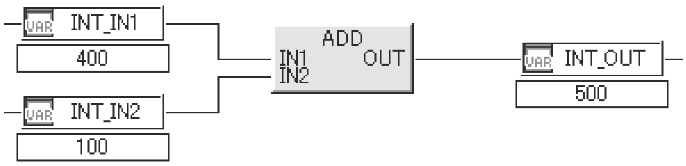
2.7.2 Overload Function

Overload function can handle several basic data types for a single input pin or output pin.

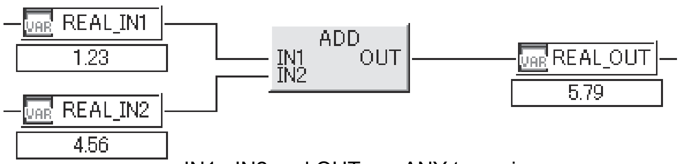
Such type that can handle several elementary data types, and decide data types automatically according to the data type of connected variable part or constant part is called ANY type.

(Example) The following is an example in which an "ADD" function connects INT type and REAL type variable.

<When connected with an INT type variable>



<When connected with a REAL type variable>



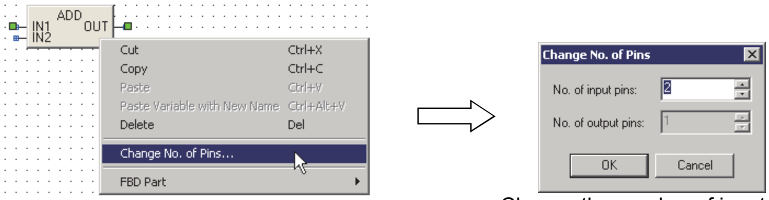
IN1 , IN2 and OUT are ANY type pins

POINT

- If there are several ANY type input pins, all of the elementary data types of the variable parts and constant parts connected to the ANY type input pins shall be set as the same type.
  If input pins are set as different elementary data types, compile errors will occur.
- If input and output pins are all of ANY type, all of the elementary data types of the variable parts and constant parts connected to the pins of ANY type shall be set as the same type.
  If input and output pins are set as different elementary data types, compile errors will occur.

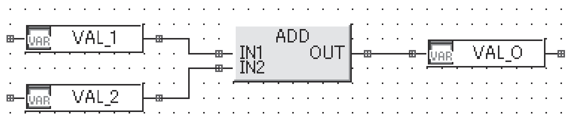
2.7.3 Input Pins Changeable Function

There is some functions whose input pin number can be changed.
Change the input pin number on the setting screen of input pin number.



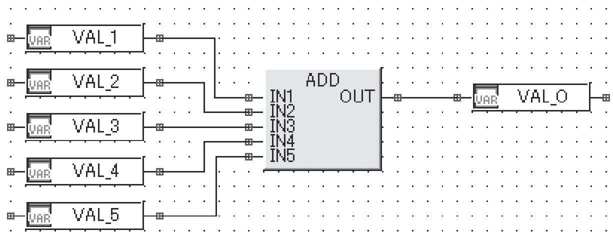
Change the number of input pins

<The number of input pins:2>



The number of input pins is changed to 5

<The number of input pins:5>



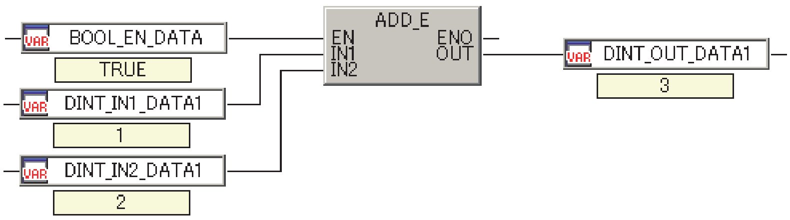
2.7.4 Function Execution Control (Function with EN/ENO Pins)

There are 2 kinds of functions: general function and the function with EN/ENO pins (with EN/ENO pins).
The function with EN/ENO pins can perform function operation control.
The input variable EN inputs the function operation conditions.
The output variable ENO outputs the status.
The operation conditions and the operation results are as follows:

Operation condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error)	Undefined value
FALSE (Operation stop)	FALSE	Undefined value

The following shows the substitution processing from OUT of the function with EN/ENO pins to variable.

1) When ENO is TRUE, substitute the output value.
   (Example) Example of substituting the output value when EN is TRUE

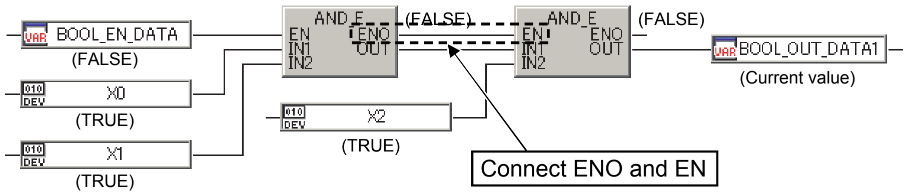


2) When ENO is FALSE, control the execution of the substitution processing depending on the FBD part connected to OUT.

FBD parts	Execution control of substitution processing when ENO is FALSE
Variable parts	Do not substitute (the value of variable connected to OUT is not changed)
FB parts	Do not substitute (the value of variable connected to OUT is not changed)
Inline ST parts	Do not substitute (the value of variable connected to OUT is not changed)
Function parts	Substitute

POINT

- For the program connected a function with EN/ENO pins to function, connect ENO and EN using a function with EN/ENO pins in order to prevent a function from using undefined value.



- For a programming tool version 1.31H or earlier, when compiling and downloading a program that includes function parts with EN/ENO pins, and their output values ENO become FALSE, the values substituted to FBD parts which are connected to OUT will be undefined value. Therefore, the program without using the value substituted to OUT is required. For details, refer to "PX Developer Version 1 Operating Manual (Programming Tool).

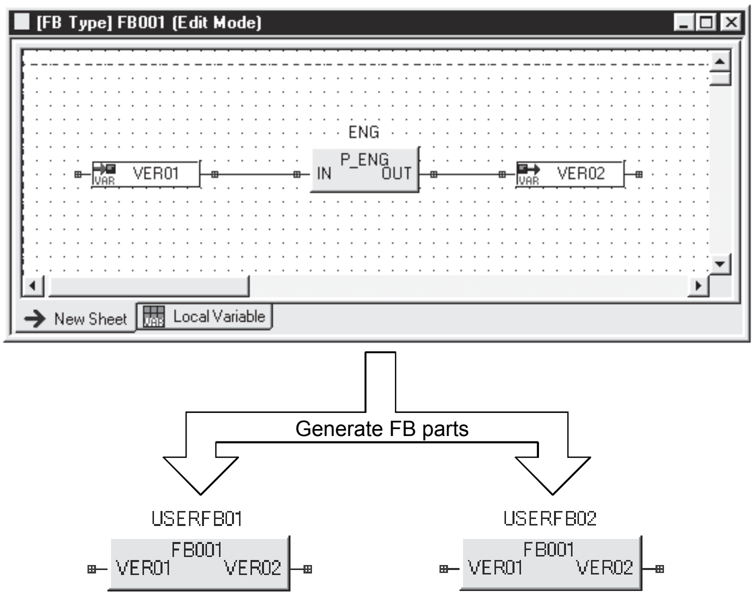
REMARK

The name of the function with EN/ENO pins is "Function name_E".

2.8 FB

2.8.1 FB

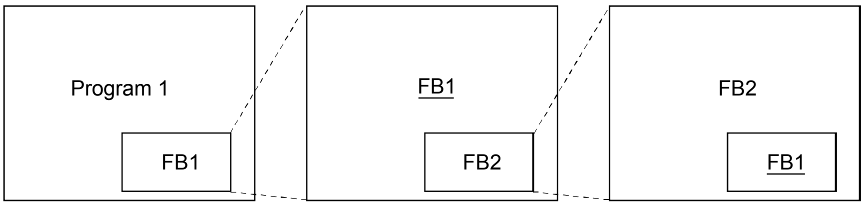
FB is used when variable of each part is named.
The FB with variable name has internal memory of its own, operates through input memory and input value of input variable, and outputs operation results through output variables.
FB is different from functions. Even if there are input pins that are not connected to FBD part by connector, compile errors will not occur.



USERFB01 and USERFB02 both perform the processing user-defined FB as "FB001".
USERFB01 and USERFB02 work separately by using internal memories.

2.8.2 Recursively Call

Recursive usage in FBD program is inhibited in programming tool.
In case of structured programming, the FB defined in upper hierarchy cannot be arranged on the lower hierarchy. (Recursively call) In case of programming by user-defined FB/ tag FB, please be careful to recursive call.
(Example) FB1 is recursively called.

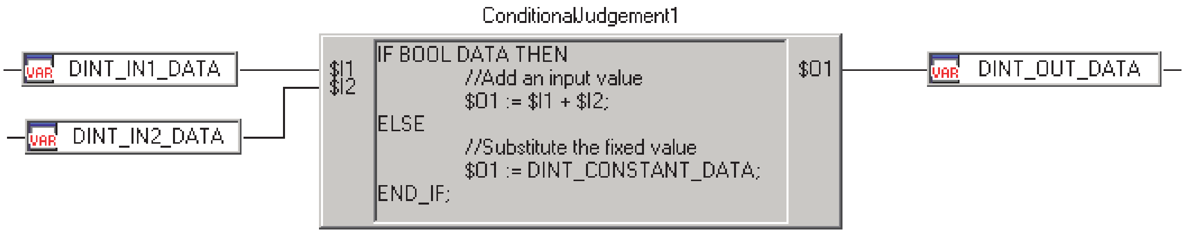


2.9 Inline ST

This section explains a programming of inline ST program to be set to inline ST part.

2.9.1 Inline ST

Inline ST part is one of FBD parts and enables to write inline ST program on a FBD sheet.
Since inline ST is a text format program which uses conditional text and operator, it simplify the writing of such as conditional judgment and arithmetic operation which are complicated to write a program with FBD language.



POINT

Inline ST parts must connect all of their input pins to other FBD parts.
If the input pins are not connected, compile errors will occur.

2.9.2 Data Exchange with FBD Program

Exchange data with FBD program with local variable and input/output variable of inline ST part as shown below.

(1) Local variable
Refer only local variables which are declared in POU.
When structure type variable members and input variable/output variable/public variable of FB type variable are applied, reference operator (.) should be attached.

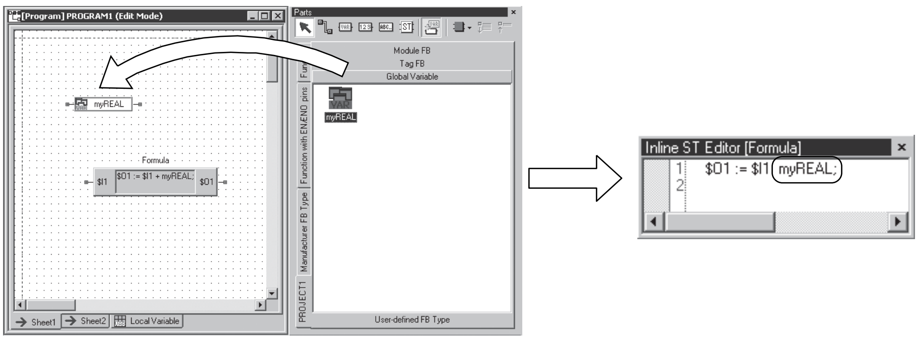
(2) Input/output variable of inline ST part
Refer with the formats as shown below.

Variable type	Format	Data type	Initial value
Input variable	$I1, $I2, …, $I8 Effective range (1 to 8) is an input pin number	ANY type *1	FBD part output value being connected to input pin is stored.
Output variable	$O1, $O2, …, $O8 Effective range (1 to 8) is an output pin number	ANY type *1	'0' is stored. (In case of character string, null (" ") is stored.)

*1 For the ANY type of input/output variable, the elementary data is any of the following.
BOOL, DINT, DWORD, INT, REAL, STRING, WORD
The data type of input/output variable is the same data type.

POINT

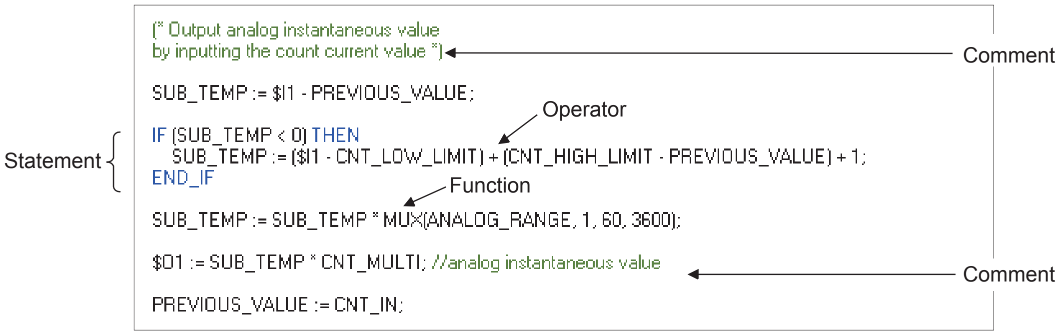
- The number of input/output variables can be set for each inline ST part.
- Defining variable parts being arranged on a FB sheet allows to use the defined variables in inline ST program.
  <When using a global variable>



- The ANY type of input/output variable is decided by FBD parts of connection destination.
- The data types which can be used in inline ST program are elementary data type and structured data type.

2.9.3 Inline ST Program Writing

The following shows an inline structure text program example.



Inline ST program is composed of statements which are written with prescribed grammar/format.

The following shows the components and references which constitute an inline ST program.

Component	Reference
Variable	Section 2.3
Constant	Section 2.6
Operator	Section 2.9.4
Statement	Section 2.9.5
Function	Section 2.9.6
Comment	Section 2.9.7

POINT

- Statements with expression only cannot be executed.
  For example, a compile error occurs with statements which do not refer the value of expression, as shown below.
  realV1;
  1.23;
  realV1 + 1.23;
  ABS(1.23);
- Inline ST program cannot call FB.
  Arrange FB parts on FBD sheet to execute FB.

2.9.4 Operator

The following shows the operators which can be used in inline ST program, prioritization of operators and functions corresponding to the operators.
The data type of value/variable to be a target of operation will be the same data type as input/output variable of functions corresponding to each operator.

Operator	Description	Example	Priority *1	Corresponding function*2
( )	Parenthesized expression	(1+2)*(3+4)	Highest	
Function ( )	Function (Parameter list)	ABS(input01);		Function to the left of the parentheses ()
**	Exponentiation	re01 := 2.0 ** 4.4		POW_
-	Sign reversal	re01 := -re01		NEG_
NOT	Logical negation	NOT bo01		NOT *3
*	Multiplication	3 * 4		MUL
/	Division	12 / 3		DIV
MOD	Modulus operation	13 MOD 3		MOD *3
+	Addition	in01 + in02		ADD
-	Subtraction	in01 - in02		SUB
<	Comparison	in01 < in02		< *3
>		in01 > in02		> *3
<=, =<		in01 <= in02		<= *3
=>, >=		in01 => in02		>= *3
=	Equality	in01 = in02		= *3
<>	Inequality	in01 <> in02		<> *3
AND, &	Logical AND	bo01 & bo02		AND *3
XOR	Exclusive OR	bo01 XOR bo02		XOR *3
OR	Logical OR	bo01 OR bo02	Lowest	OR *3

*1 The prioritization of the operators in the same frame is the same.
If one expression includes multiple operators in the same priority, the operation is performed from the leftmost operator.

*2 When an error occurs with an operator of ST program in CPU module, the function to the operator is displayed on the area of error in the FBD program diagnose dialog box.
Ex: When an error occurs with division by zero with "/", "Program 1.DIV" is displayed as an error area, the area where corresponding operator "/" is used is displayed with error JUMP.

*3 Cannot be called as a function. (Section 2.9.6)

POINT

Operators are recognized without distinction between uppercase/lowercase.

2.9.5 Statement

The following shows the statement which can be used in inline ST program.

Type of statement	Description
Assignment statement	Assignment statement
Conditional statement	IF THEN conditional statement, IF ELSE conditional statement, and IF ELSIF conditional statement

Conditional statement evaluates a Boolean expression and then executes the specific statement in accordance with TRUE or FALSE which Boolean expression results.
An expression which results TRUE or FALSE can be used. (Example: int1 > 1)

(1) Assignment statement

(a) Format

```
<Left side> := <Right side>;
```

(b) Description

The assignment statement assigns the result of the right side expression to the variable of the left side.
The result of the right side expression and data type of the left side need to obtain the same data when using the assignment statement.

(c) Example

```
intV1 := 1;
intV2 := intV2 + 2;
```

(2) IF THEN conditional statement

(a) Format

```
IF <Boolean expression> THEN
<Statement ...>;
END_IF;
```

(b) Description

The statement is executed when the value of Boolean expression (conditional expression) is TRUE. The statement is not executed if the value of Boolean expression is FALSE.

(c) Example

```
IF bool1 THEN
  intV1 := 1;
  intV2 := intV2 + 2;
END_IF;
```

(3) IF ELSE conditional statement

(a) Format

```
IF <Boolean expression> THEN
  <Statement 1 ...>;
ELSE
  <Statement 2 ...>;
END_IF;
```

(b) Description

Statement 1 is executed when the value of Boolean expression (conditional expression) is TRUE.
Statement 2 is executed when the value of Boolean expression is FALSE.

(c) Example

```
IF int1 > 1 THEN
  intV2 := intV2 + 2;
ELSE
  intV3 := intV3 + 3;
END_IF;
```

(4) IF ELSIF conditional statement

(a) Format

```
IF <Boolean expression 1> THEN
  <Statement 1 ...>;
ELSIF <Boolean expression 2> THEN
  <Statement 2 ...>;
ELSIF <Boolean expression 3> THEN
  <Statement 3 ...>;
END_IF;
```

(b) Description

Statement 1 is executed when the value of Boolean expression (conditional expression) 1 is TRUE.
Statement 2 is executed when the value of Boolean expression 1 is FALSE and the value of Boolean expression 2 is TRUE.
Statement 3 is executed when the value of Boolean expression 1 and 2 are FALSE and the value of Boolean expression 3 is TRUE.

(c) Example

```
IF bool1 THEN
  intV1 := intV1 + 1;
ELSIF int1 > 1 THEN
  intV2 := intV2 + 2;
ELSIF (0 < int2) & (int2 < 10) THEN
  intV3 := intV3 + 3;
END_IF;
```

POINT

When comparing Boolean expression with constant of TRUE or FALSE and equality (=) or inequality (<>), programming with the format of (1), (2) decreases the number of ladder program steps which are generated with compilation.

(1) A format with Boolean expression itself and without equality (=) or inequality (<>) when evaluating Boolean expression is TRUE.

IF <Boolean expression> = TRUE THEN or IF <Boolean expression> <> FALSE THEN	→	IF <Boolean expression> THEN
Ex: IF INT_TO_BOOL (intV1) = TRUE THEN	→	IF INT_TO_BOOL (intV1) THEN

(2) A format with "NOT" operator when evaluating Boolean expression is FALSE.

IF <Boolean expression> = FALSE THEN or IF <Boolean expression> <> TRUE THEN	→	IF NOT <Boolean expression> THEN
Ex: IF INT_TO_BOOL (intV1) = FALSE THEN	→	IF NOT INT_TO_BOOL (intV1) THEN

2.9.6 Function

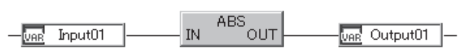
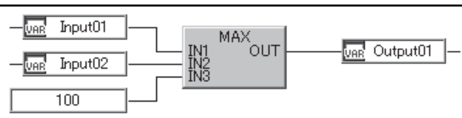
In inline ST program, function parts can be called as functions.
The following shows a method of calling functions and functions can be used.

(1) Method of calling functions
The following description is used to call a function

```
Function name (Argument1, Argument2, ...)
```

Enclose the arguments by '( )' after the function name.
When using multiple variables, delimit them by ','.
The argument order is the order of function part input pins.
The arguments can specify expressions.

Example

	Inline ST	FBD language
Calling a function with one input variable (Ex: ABS)	Output01 := ABS(Input01);	
Calling a function with multiple input variables (Ex: MAX)	Output01 := MAX(Input01, Input02, 100);	

(2) Functions which can be used in inline ST program
In inline ST program, functions which have only one output variable can be used.

However, functions whose name is the same as operator cannot be used.
(NOT, MOD, AND, XOR, OR, <, >, <=, >=, =, <>)

The following shows a list of functions can be used.
Arrange the function parts on a FBD sheet to execute the functions which are not in the list.

Classification	Function name			
Type conversion function	BCD_TO_DINT	BCD_TO_INT	BOOL_TO_DINT	BOOL_TO_DWORD
	BOOL_TO_INT	BOOL_TO_WORD	DINT_TO_BCD	DINT_TO_BOOL
	DINT_TO_DWORD	DINT_TO_INT	DINT_TO_REAL	DINT_TO_STRING
	DINT_TO_WORD	DWORD_TO_BOOL	DWORD_TO_DINT	DWORD_TO_INT
	DWORD_TO_WORD	INT_TO_BCD	INT_TO_BOOL	INT_TO_DINT
	INT_TO_DWORD	INT_TO_REAL	INT_TO_STRING	INT_TO_WORD
	REAL_TO_DINT	REAL_TO_INT	REAL_TO_STRING	REAL_TO_STRING_EX
	STRING_TO_DINT	STRING_TO_INT	STRING_TO_REAL	WORD_TO_BOOL
	WORD_TO_DINT	WORD_TO_DWORD	WORD_TO_INT	
Numerical operation function	ABS	ACOS	ASIN	ATAN
	COS	EXP	LN	LOG
	NEG_	SIN	SQRT	TAN
Arithmetic operation function	ADD	DIV	MUL	POW_
	SUB			
Bit-string function	ROL	ROR	SHL	SHR
Logical operation function *1	—			
Selection function	LIMIT	MAX	MIN	MUX
	SEL			
Comparison function *1	—			
Character string function	CONCAT	DELETE	FIND	INSERT
	LEFT	LEN	MID	REPLACE
	RIGHT			
Helper function	BIND	HI_WORD	IS_CONNECTED_	LO_WORD
	MAKE_DWORD			
Ladder program control function	PLOW_E	POFF_E	PSCAN_E	PSTOP_E
Analog value selection and average value function	P_AVE			

*1 Functions cannot be called in inline ST program.

POINT

- Specify all input variables. (For input pins changeable function, specify the number of input pins to be used.)
- Functions are recognized with distinction between uppercase/lowercase.

2.9.7 Comment

This section explains the comments can be used in inline ST program.

(1) Multiple line comment
  (a) Format
    (* <Comment> *)

  (b) Description
    Start with a parenthesis and an asterisk, and regard the character strings as comment until the next asterisk and parenthesis appear.
    Comment can be used for more than one line.

  (c) Example
    (* Execute the temperature correction.
    Temperature correction = (Design temperature + Bias temperature) /
    (Measured temperature + Bias temperature) *)

(2) Single line comment
  (a) Format
    // <Comment>

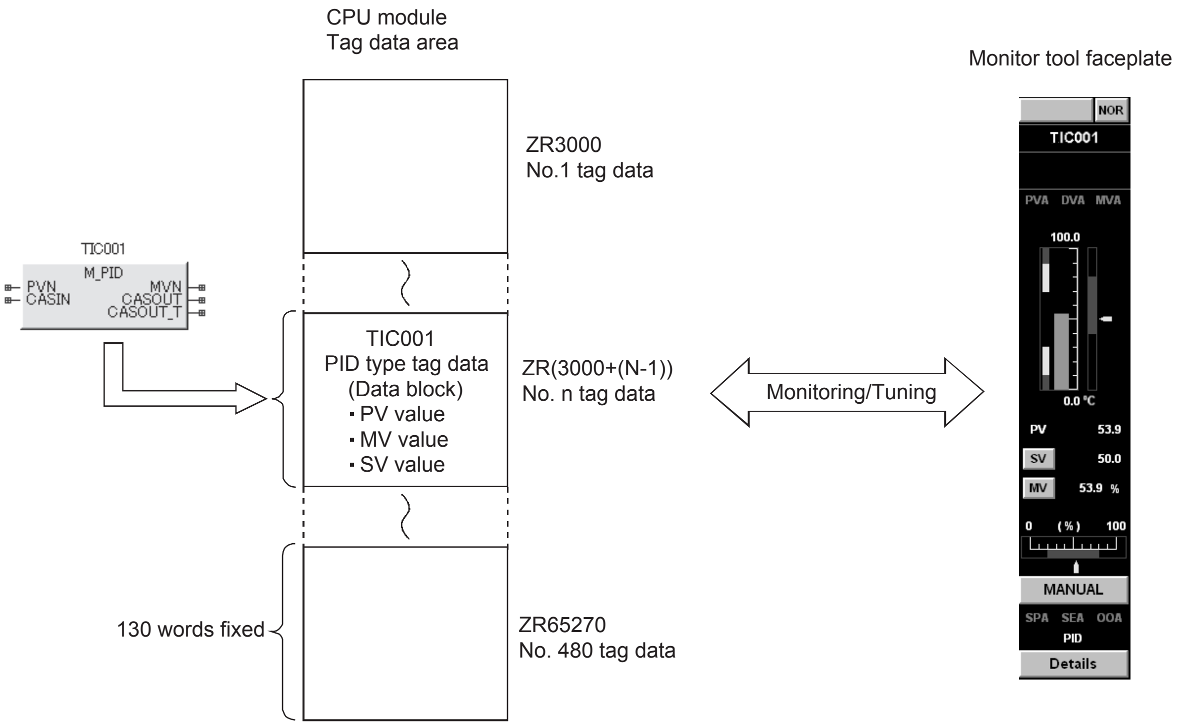
  (b) Description
    Start with 2 slash marks and regard the character strings as comment until a line feed character appears.

  (c) Example
    intV1 := 0; // Variable 1

2.10 Tag

2.10.1 Overview of Tag

Tag is the identifier for all kinds of the DDC processing of process control system.
Tag data is the pack of the data that are related to the DDC processing shown by tag.
It is easy to implement tags by using tag FB.
Tag data is attached inside the tag FB part. The DDC processing status can be monitored through monitoring the tag data with monitor tool.
Tag data area is reserved in the device area of CPU module. The size of tag data area is set as maximum no. of tags in Tag FB definition window of programming tool. Tag data of each tag FB has a fixed head device address.
For details about monitor tool, refer to "PX Developer Version 1 Operating Manual (Monitor Tool)".



2.10.2 Tag FB

Tag FB is the extended POU, and it has tag data.
The differences between tag FB and FB are shown in following table:

Item	Tag FB	FB
Tag data	All of the tag types have tag data whose structure has been defined	No tag data
Variable declaration	Global declaration	Only for local declaration
User-defined	It can be defined by user with all function parts and FB parts.	It can be defined by user with all function parts and FB parts except tag access FB.

REMARK

As the structure of tag data area of CPU module has been defined, the tag data can be read once in batch and displayed on the faceplate of monitor tool.

2.10.3 Tag Type

(1) About tag type

Tag FB has the property called tag type.
Tag data structure attached to the tag FB and the faceplate type of monitor tool can be specified by the tag type.
There are 4 tag types:

Tag type	Description
Loop tag	Tags used for loop control processing. It is equivalent to loop tags of process control instructions of CPU module.
Status tag	Tags used for monitoring and controlling ON/OFF status
Alarm tag	Tags used for alarm notification
Message tag	Tags used for guidance message notification

(2) List of tag type and manufacturer tag FB

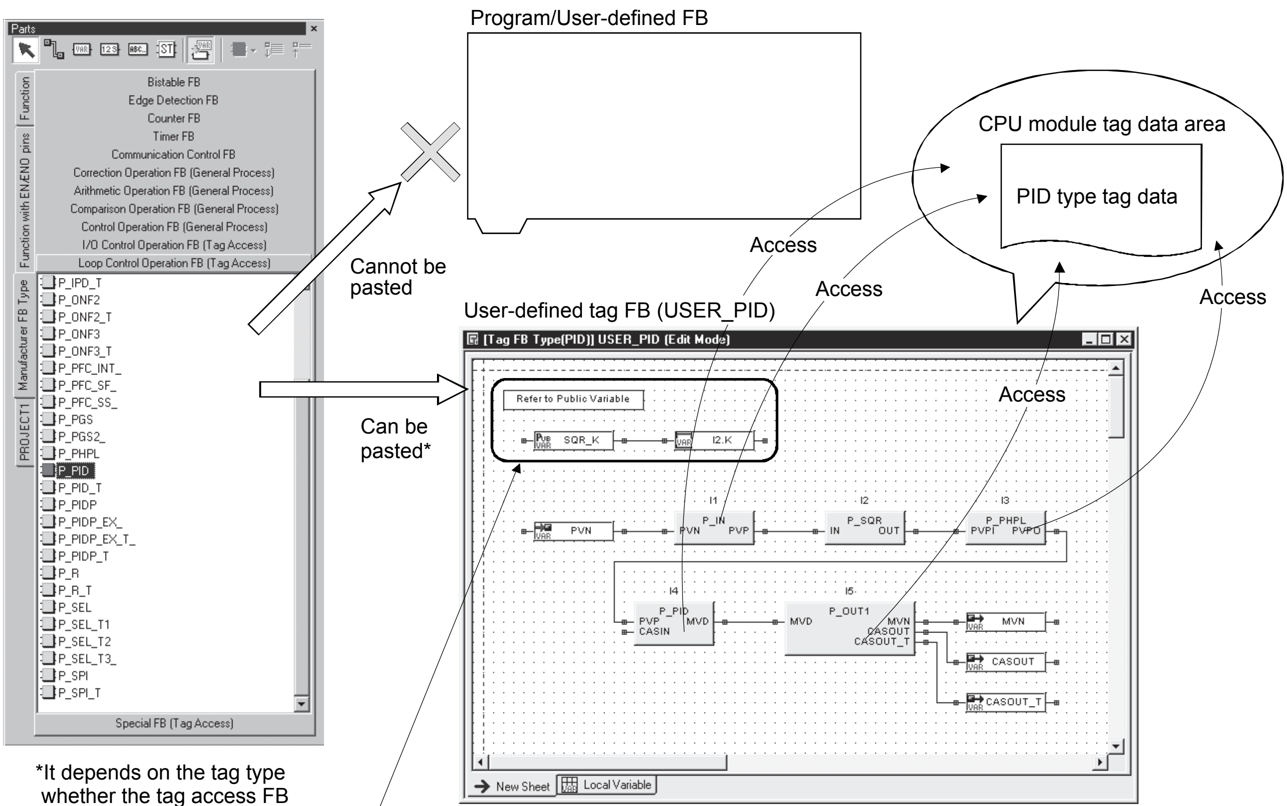
Using manufacturer Tag FB part, the definition of tag processing is not needed and the implementation of tag is realized easily.

Classification	Tag type	Name	Manufacturer Tag FB
Loop tag	PID	PID control	M_PID(_T), M_PID_DUTY(_T)
	2PID	2-degree-of-freedom PID control	M_2PID(_T), M_2PID_DUTY(_T)
	2PIDH	2-degree-of-freedom advanced PID control	M_2PIDH(_T)_
	PIDP	Position type PID control	M_PIDP(_T), M_PIDP_EX(_T)_
	SPI	Sample PI control	M_SPI(_T)
	IPD	I-PD control	M_IPD(_T)
	BPI	Blend PI control	M_BPI(_T)
	R	Ratio control	M_R(_T)
	ONF2	2 position ON/OFF control	M_ONF2(_T)
	ONF3	3 position ON/OFF control	M_ONF3(_T)
	PFC_SF	Predictive functional control (simple first order lag)	M_PFC_SF_
	PFC_SS	Predictive functional control (simple second order lag)	M_PFC_SS_
	PFC_INT	Predictive functional control (integral process)	M_PFC_INT_
	PGS	Program setter	M_PGS
	PGS2	Multi-point program setter	M_PGS2_
	MOUT	Manual output	M_MOUT
	MONI	Monitor	M_MONI
	SWM	Manual setter with monitor	M_SWM_
	MWM	Manual output with monitor	M_MWM
	SEL	Loop selector	M_SEL(_T1) (_T2) (_T3_)
	BC	Batch counter	M_BC
	PSUM	Pulse integrator	M_PSUM
	PVAL	Position-proportional output	M_PVAL_T_
	HTCL	Heating and cooling output	M_HTCL_T_
Status tag	NREV	Monitor irreversible control	M_NREV
	REV	Monitor reversible control	M_REV
	MVAL1	On/OFF control 1 (without intermediate value)	M_MVAL1
	MVAL2	On/OFF control 2 (with intermediate value)	M_MVAL2
	PB	Push button operation	M_PB_
	TIMER 1	Timer 1 (Timer stops when COMPLETE flag is on.)	M_TIMER1
	TIMER 2	Timer 2 (Timer continues when COMPLETE flag is on.)	M_TIMER2
	COUNT1	Counter 1 (Counter stops when COMPLETE flag is on.)	M_COUNTER1
	COUNT2	Counter 2 (Counter continues when COMPLETE flag is on.)	M_COUNTER2
Alarm tag	ALM	Alarm	M_ALARM
	ALM_64PT	64-points alarm	M_ALARM_64PT_
Message tag	MSG	Message	M_MESSAGE
	MSG_64PT	64-points message	M_MESSAGE_64PT_

2.10.4 User-defined Tag FB and Tag Access FB

(1) User-defined tag FB
When implementing tags, tag FB supplied by manufacturer and user-defined tag FB are both applicable.
The user-defined tag FB has the corresponding tag data structure with the tag type.
Thus the processing contents can be programmed using tag access FB, FB or function.

(2) Tag access FB
Tag access FB can be used only on user-defined tag FB.
Tag access FB part accesses the tag data of user-defined tag FB in which the tag access FB is pasted when it is executed.
Thus, tag access FB cannot be used in FB or program that do not have tag data.



*It depends on the tag type whether the tag access FB can be pasted.

When user-defined tag FB "USER_PID" is pasted to FBD table, public variables of FB used in USER_PID cannot be displayed on FB property window.
For method of setting the initial values of these public variable on FB property window, please refer to "User-defined tag FB type" (2) in "PX Developer Operating Manual (Programming Tool)".
The above graph is an example of setting the initial value of public variable K of I 2 (P_SQR) in FBproperty window.

POINT

When tag access FB parts are used in user-defined tag FB, the applicable tag access FB parts depend on the tag type of user-defined tag FB.
For tag type and applicable tag access FB parts, refer to Appendix 1.3.

2.10.5 Initial Setting of Tag Data and Operation Constant

The information of setting initial values of tag data and operation constant is shown here.

(1) About initial value setting

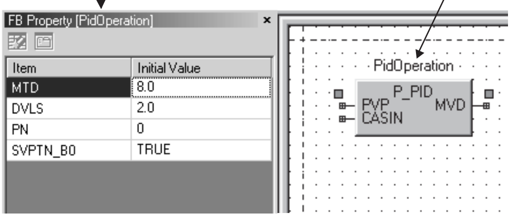
Setting the initial value of tag data	......	The initial value of tag data must be set for each tag
Setting the initial value of operation constant	......	It is necessary to set operation constant in the tag access FB part and FB part that has encapsulated the process control instructions of CPU module, such as tag access FB.

(2) About the setting methods

Tag data and operation constant are handled as the public variables that are attached to tag FB parts. Therefore, the initial value of tag FB parts can be set on FB property window.

Public variables whose initial values can be changed are displayed on the FB property window.

The selected FB part.



2.11 Module FB

Module FB performs input /output processing of data from the module that is connected to the base unit of PLC.
In using module FB, program for data exchange can be realized without being conscious of the address of input/output X/Y device and buffer memory. The names of modules that can be used by programming tool are shown in the following table.

Classification		Names of corresponding modules
Digital Input/output module		QX10, QX28, QX40, QX40-S1, QX41, QX41-S1, QX42, QX42-S1, QX50, QX70, QX71, QX72, QX80, QX81, QX82, QX82-S1, QY10, QY18A, QY22, QY40P, QY41P, QY42P, QY50, QY68A, QY70, QY71, QY80, QY81P, QH42P, QX48Y57
Analog module		Q64AD, Q64AD2DA, Q68ADV, Q68ADI, Q62AD-DGH, Q66AD-DG, Q64AD-GH, Q68AD-G, Q62DA, Q62DAN, Q64DA, Q64DAN, Q68DAV, Q68DAVN, Q68DAI, Q68DAIN, Q62DA-FG, Q66DA-G, Q68CT
Temperature input module		Q64TD, Q64TDV-GH, Q68TD-G-H01, Q68TD-G-H02, Q64RD, Q64RD-G, Q68RD3-G
Counter module		QD62, QD62E, QD62D, QD60P8-G
Remote Module via CC-Link master module *1	Master module	QJ61BT11, QJ61BT11N
	For remote I/O station	General CC-Link Remote Station (occupy 1 to 4 station)
	For remote device station	General CC-Link Remote Station (occupy 1 to 4 station)

*1 Incompatible with CC-Link Ver. 2.

2.11.1 Requirements to Use Module FB

Prior to use the module FB, complete the following operations, startup of the module, and the settings necessary to use the module (☞ (1) in this section).
Then, declare the module FB with the programming tool. This allows the module FB to be used in FBD programs. (☞ (2) in this section)

> **POINT**
> - For items which can be set with both intelligent function module operation and module FB (conversion enable/disable setting, for example), set the items to be the same between the module operation and FB.
> - For the module that uses the module FB to perform data I/O processing, do not execute the auto refresh function on the PLC devices using intelligent function module operation. When the auto refresh function is executed, the output values of the module FB will be illegal.

(1) Settings with intelligent function module operation

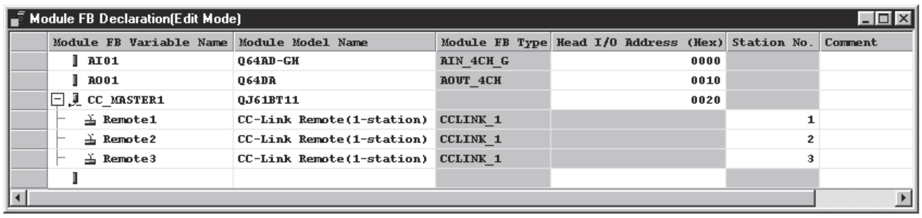
Perform the following settings which are necessary to use the analog module, temperature input module, or counter module with GX Works2 or GX Developer and GX Configurator.

- Intelligent function module switch setting
- Parameter (initial setting)
- Auto refresh

For the settings with intelligent function module operation, start GX application from the project window of the programming tool.
Use GX application to make the network parameter settings necessary to use the CC-Link remote module.
For details of the setting procedure and method of each module, refer to the corresponding user's manual.

(2) Settings with programming tool

Module FB parts are generated in Parts window automatically when they are declared in module FB declaration window.
For the setting methods of module FB declaration window, please refer to "PX Developer Version 1 Operating Manual (Programming Tool)".



2.11.2 Access to MELSECNET/H Remote I/O Station

This section explains creating FBD programs by using data stored in buffer memory of the intelligent function module mounted to a MELSECNET/H remote I/O station.

POINT

- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
- For details of access to buffer memory of the intelligent function module mounted to a MELSECNET/H remote I/O station, refer to the following manual.
  Q Corresponding MELSECNET/H Network System Reference Manual (Remote I/O network)
  Intelligent function module manual

(1) Accessing MELSECNET/H remote I/O station

Read from/Write to an intelligent function module using either of the following two methods.

(a) Using intelligent function module operation

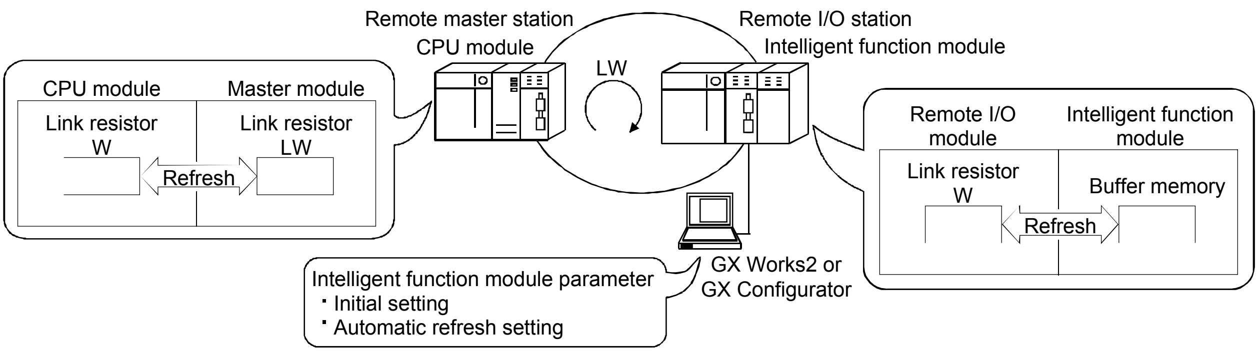
Use the auto refresh function of GX Works2 or GX Configurator to enable data to be read/written between a CPU module and intelligent function module.

(GX Works2 or GX Configurator setting)

Make the settings so that the remote I/O module link register (W) will automatically refresh the target buffer memory in the automatic refresh setting of intelligent function module parameter.
The setting example is shown in (2) in this section.
GX Works2 or GX Configurator for the corresponding intelligent function module is required.

(FBD program)

Set the link register (W) including the above automatic refresh setting in the PX Developer global variable declaration window.



POINT

When decreasing the number of link register (W) points to be set in link parameter, use data register (D) as automatic refresh target device.
In this case, make the settings using the remote I/O station device transfer parameter in order that the necessary data will be transferred from the automatic refresh setting data register (D) to link register (W).

(b) Using link-dedicated instruction

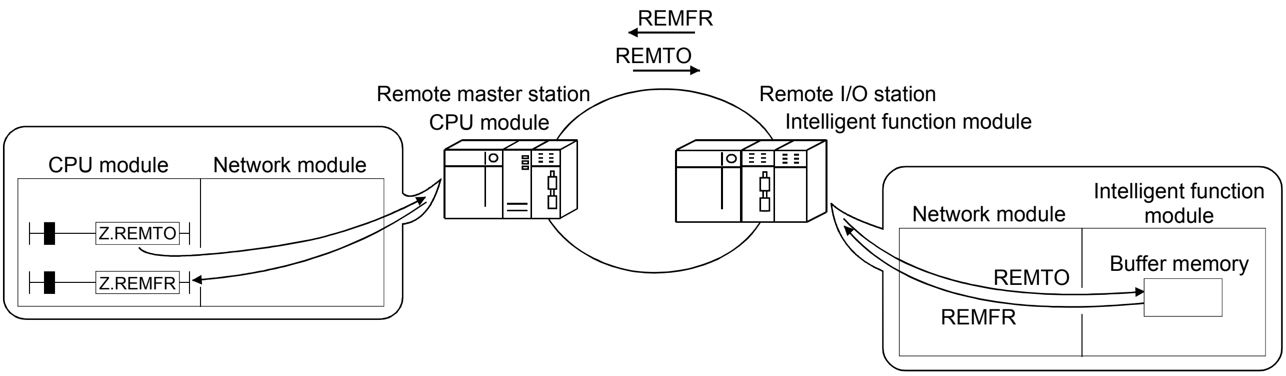
By executing a sequence program, FBD programs can be created using data read to a CPU module.

(Creating sequence program)

Use the REMER instruction/REMTO instruction to create a sequence program that reads from/writes to the intelligent function module buffer memory.

(FBD program)

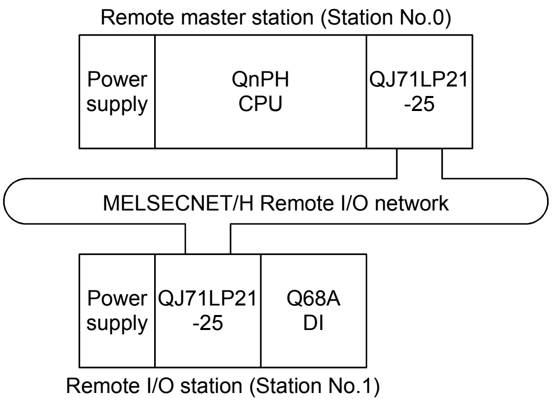
Set the register that stores data which have been read from/written to the buffer memory in the PX Developer global variable window.



(2) Setting example

The following shows the example for setting link register (W) to automatic refresh target device.

(a) System configuration



(Network parameter Network range assignment)
- Remote master station Remote I/O station
  Input: X100 to X10F ← Input: X000 to X00F
  Output: Y100 to Y10F → Output: Y000 to Y00F
- Remote master station ← Remote I/O station
  Link register: W0 to W7

(b) Remote master station network parameter setting

Set network type, head I/O No., network No., number of total slave stations and mode. And then, set network range assignment and refresh parameters.

(Network range assignment)

- XY setting

StationNo.	M station -> R station						M station <- R station					
	Y			Y			X			X		
	Points	Start	End	Points	Start	End	Points	Start	End	Points	Start	End
1	16	0100	010F	16	0000	000F	16	0100	010F	16	0000	000F

- BW setting

StationNo.	M station -> R station			M station <- R station			M station -> R station			M station <- R station		
	B			B			W			W		
	Points	Start	End	Points	Start	End	Points	Start	End	Points	Start	End
1										8	0000	0007

(Refresh parameter setting)

	Link side					PLC side			
	Dev. name	Points	Start	End		Dev. name	Points	Start	End
Transfer SB	SB	512	0000	01FF	↔	SB	512	0000	01FF
Transfer SW	SW	512	0000	01FF	↔	SW	512	0000	01FF
Random cyclic	LB				↔				
Random cyclic	LW				↔				
Transfer1	LB	8192	0000	1FFF	↔	B	8192	0000	1FFF
Transfer2	LW	8192	0000	1FFF	↔	W	8192	0000	1FFF
Transfer3	LX	512	0000	01FF	↔	X	512	0000	01FF
Transfer4	LY	512	0000	01FF	↔	Y	512	0000	01FF
Transfer5					↔				
Transfer6					↔				

(c) Remote I/O station parameter setting

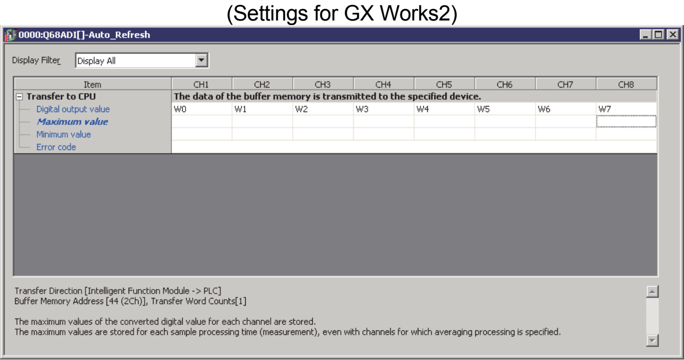
[Setting procedure for GX Works2]

1. Specify "QJ72LP25/QJ72BR15 (Remote I/O)" as PLC type and newly create a project.
2. Select [Project] – [Intelligent Function Module] – [New Module] menu of GX Works2
3. Select [Switch Setting]/[Parameter] from [Intelligent Function Module] – [(module)] in the project view, as necessary.
4. Select [Intelligent Function Module] – [(module)] – [Auto Refresh] on the Project view, assign the link register (W) that transfers buffer memory in the auto refresh setting.
5. Write parameters into the remote I/O station.

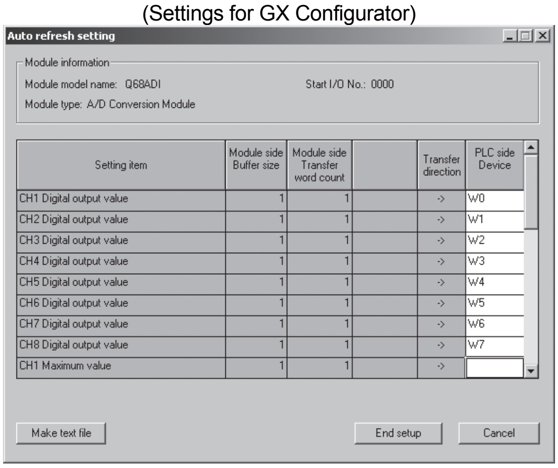
[Setting procedure for GX Configurator]

1. Specify "Remote I/O" as PLC type and newly create a project.
2. Start the intelligent function module utility.
   Select [Tool] – [Intelligent function module utility] in the [Start] menu.
3. Make the initial settings such as sampling and averaging processing specification, as necessary.
4. Assign the link register (W) that transfers buffer memory in the auto refresh setting.
5. Write parameters into the remote I/O station.

(Settings for GX Works2)

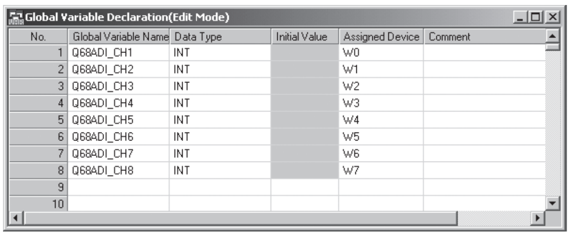


(Settings for GX Configurator)



(d) FBD program setting

Register the link register (W) that includes auto refresh setting onto the PX Developer global variable window.
Register input relay (X) and output relay (Y) that correspond to the intelligent function module I/O signals, as necessary.



2.12 Execution of FBD Program

This section describes execution of FBD program.

2.12.1 Execution Type and Priority/Phase of Program

The execution cycle, timing and priority of FBD program can be set.
The execution types of FBD program will be explained below.

(1) How to execute a FBD program
FBD program can be executed in two ways: timer execution and interrupt execution. Choose method according to different needs for applying programs.
The execution types will be explained below.

(a) Timer execution
Timer execution is an execution method that uses the scan time of CPU module to multiply the execution time.
Users can choose from the four types of execution time for timer execution.

Type of execution time	Description
High-speed	Program is executed once every 200ms
Normal	Program is executed once every [(high-speed execution type cycle 200ms)×n1] ms (n1=2,3,4,5)
Low-speed	Program is executed once every [(high-speed execution type cycle 200ms)×n2] ms (n2=5,10,20,25,50)
Scan	Program is executed once for each scan, beginning from the scan after the execution of initial execution type program (that is executed once in switching from power ON to RUN or from STOP to RUN)

*1 n1 and n2 can be set with programming tool
For details, refer to "PX Developer Version 1 Operating Manual (Programming Tool)".
*2 Priority and execution conditions can be set for timer executing.
Phase can be set for normal and low-speed execution.
For details, refer to (2) in this section.

POINT

(1) Timer execution at any timing other than those based on scan types will cause an error of up to +1 scan time.
Timer execution of other than the scan type has a larger error than interrupt execution.
(2) Programs including general process FB, tag access FB, and loop tag FB cannot be set as scan type timer execution.
For more precautions with scan starting, refer to Section 2.12.1 (4).
(3) To keep the fixed scan cycle of the program, the scan time must be within 200ms. (The scan time can be checked with GX application.)

(b) Interrupt execution

Interrupt execution is an execution type that inserts execution program in the execution process of "(a) Timer execution" program.

There are two kinds of interrupt execution: fixed scan execution based on fixed scan execution program of CPU module and interrupt pointer execution by use of interrupt pointer (I).

Execution type	Description
Fixed scan execution	Program is executed at set intervals (execution time).
Interrupt pointer execution	It is executed after stopping other programs for a while when interruption factors indicated in interrupt pointer (I) of CPU module occur.

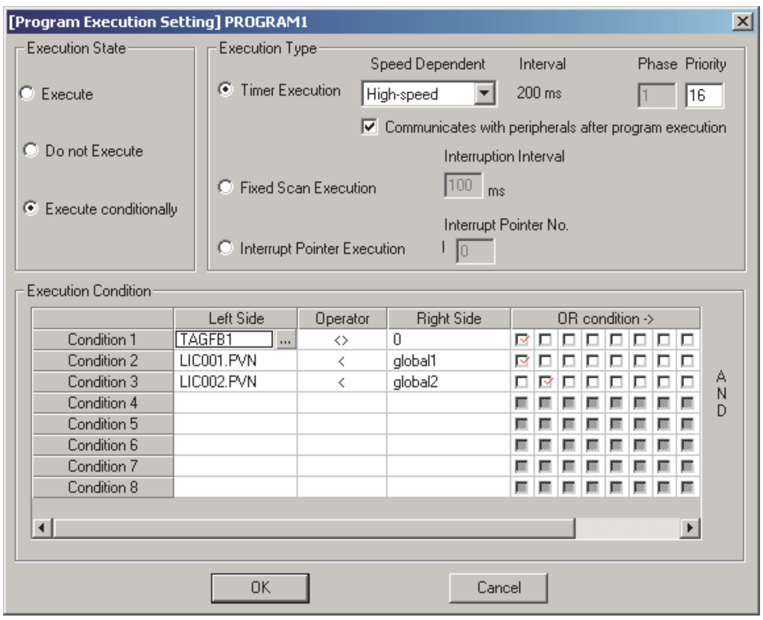
POINT

Programs that contain general process FB, tag access FB, and loop tag FB shall not be set as interrupt pointer execution. For precautions of applying interrupt execution, refer to Section 2.12.1 (4).

(2) Priority and Phase

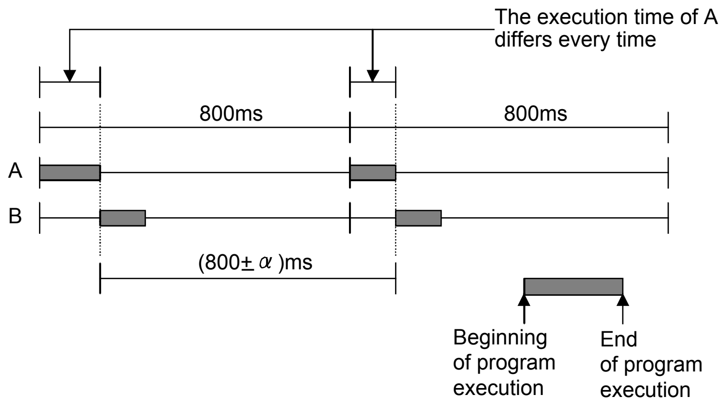
The following paragraphs explain priority and phase that can be set for timer execution.

- Priority .... when there are more than one programs to be executed using the same execution method, priority shall be set to decide which program is to be executed first.
  The closer to "0", the earlier the program is to be executed.
  Priority is only valid for programs that use the same execution method.
- Phase ..... stagger the program execution at a constant interval of 200ms.



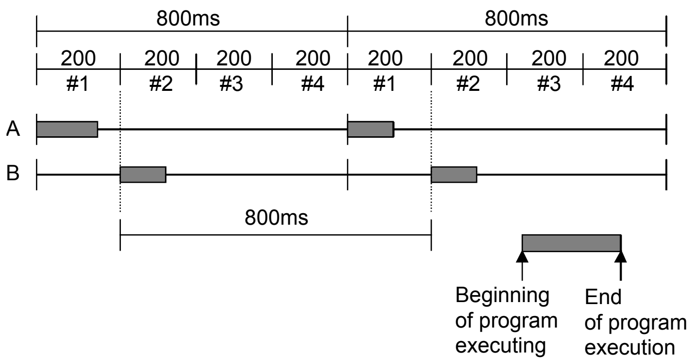
(a) Phase setting
When several programs with the same execution cycle are executed simultaneously, the program with higher priority is started prior to the other programs. This may influence the on-time performance of program. (Example 1)
As shown in (Example 2), if we divide the execution cycle into 200ms units (phase), by specifying the execution time and phase in program execution setting, the degrading of on-time performance can be effectively prevented.

(Example 1) Constant period execution of program without phase.



Suppose program A and program B have the same execution cycle of 800ms, and are executed simultaneously, program A, with higher priority, is first executed at an interval of 800ms, and program B is executed immediately after program A finishes executing. Therefore, the execution time of program A decides execution cycle of program B, which will not be exactly 800ms. Fixed scanning performance thus degrades. (In the diagram on the left, the undetermined factor (± α) in program B execution cycle results from the fluctuation of program A execution time.)

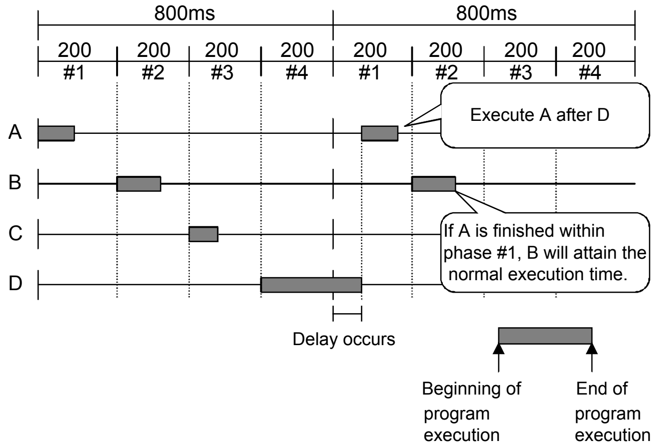
(Example 2) Constant period execution of program with phase



Divide the execution time 800ms into 200ms × 4, set program A to start at phase #1 and program B at #2. As long as the execution time of A is below 200ms, the execution time of program B will hold 800ms, without influenced by the execution time of A.

(b) Precautions for phase setting
As for interrupt execution, the execution time of program may be longer than normal and the next execution time will exceed the phase time. In this case, the fixed scanning performance will degrade, which is worth user's attention. However, if the exceeded part of execution time and the next execution time is within one phase, the program will be executed in normal execution time hereafter.

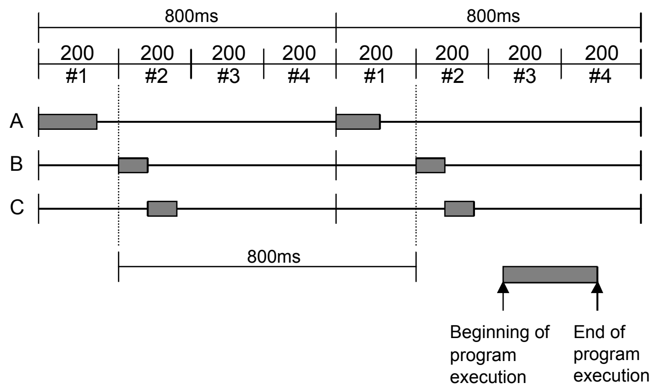
(Example 3) Fixed scan execution of program when delay occurs.



Program A is executed in phase #1, program B in phase #2, C In #3, D in #4. As for interrupt execution, execution time of program D exceeds 200ms. Accordingly, program A cannot be executed in phase #1until program D execution ends, resulting in longer program A execution time. Therefore, the execution time of program A is not 800ms, and the on-time performance will degrade.
However, if the total sum of the exceeding time of program D and the execution time of program A is no more than 200ms, program B can be executed within normal execution time.

(c) Precautions for specifying several programs with same execution cycle and phase.
When specifying several programs with same phase, programs with higher priority will be executed first despite of the same execution time requirements.
In the case of total execution time exceeding 200ms, timeout delay occurs and program execution schedule will be disrupted. (This kind of delay cannot be avoided)
Pay attention to the total sum of execution time and scan time when setting several programs with a single phase.

(Example 4) Constant period execution of programs with the same execution cycle and phase.



When setting program B and C with a single phase, program C is not executed until program B with higher priority finishes executing.

(3) Precautions for use of high-speed/normal-speed/low-speed timer execution
The execution timings of normal-speed and low-speed execution are determined by the execution timing of a high-speed execution type program.
The execution cycle of a high-speed execution type program is fixed at 200ms.
If the execution cycle of a high-speed execution type program cannot be kept to 200ms, this affects the execution cycles of all speed execution type programs.
The following expression shows the execution time relation between a high-speed execution type program and a normal/low-speed execution type program.
For example, when the execution cycle of 1000ms is set to a normal-speed execution type program, the normal-speed execution type program will be executed once every five times the high-speed execution type program is executed.

$$\frac{\text{Execution cycle setting of normal-speed execution type}}{\text{Execution cycle of high-speed execution type (fixed)}} = \frac{1000\text{ms}}{200\text{ms}}$$

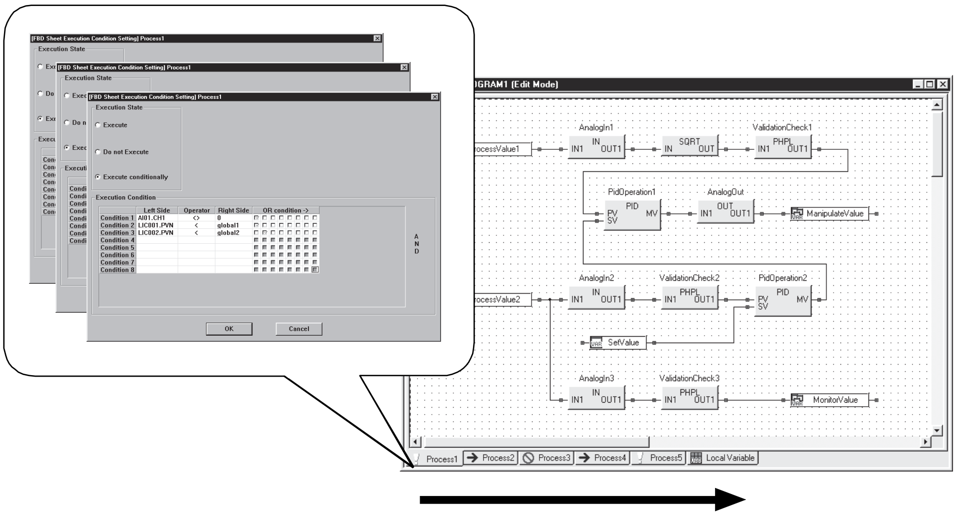
If the scan time is 300ms, the high-speed execution type program is executed every 300ms. (This is because the program execution timing is controlled by adding up the scan times.)
When the execution cycle of the high-speed execution type program is 300ms, the execution cycle of the normal-speed execution type program is as follows:
300ms×5 = 1500ms, producing an error. (Error 500ms)
Hence, to keep the program cycle fixed, the scan time must be within 200ms.
The scan time can be checked using GX application.

(4) Precautions for scan type timer execution and interrupt pointer execution
Do not paste general process FB, tag access FB or loop tag FB in scan type timer execution and interrupt pointer execution programs.
As general process FB, tag access FB and loop tag FB must execute operation processing based on execution time, setting program as timer execution scan type and interrupt execution without execution time may result in invalid control for it.

2.12.2 Setting of FBD Sheet Execution Conditions

User-defined FB/program is executed according to the conditions setting in execution condition setting dialog box.
If user-defined FB/program execution conditions are described in several FBD sheets, these FBD sheets are executed one by one from left to right.
FBD sheet will not be executed if condition is not satisfied.
The execution condition setting of FBD sheet can be used as interlock of FBD sheet unit in the processing contents of POU definition.
For the setting method for execution condition setting dialog box and the details of setting items, refer to "PX Developer Version 1 Operating Manual (Programming Tool)".

Execution conditions can also be set in FBD sheets

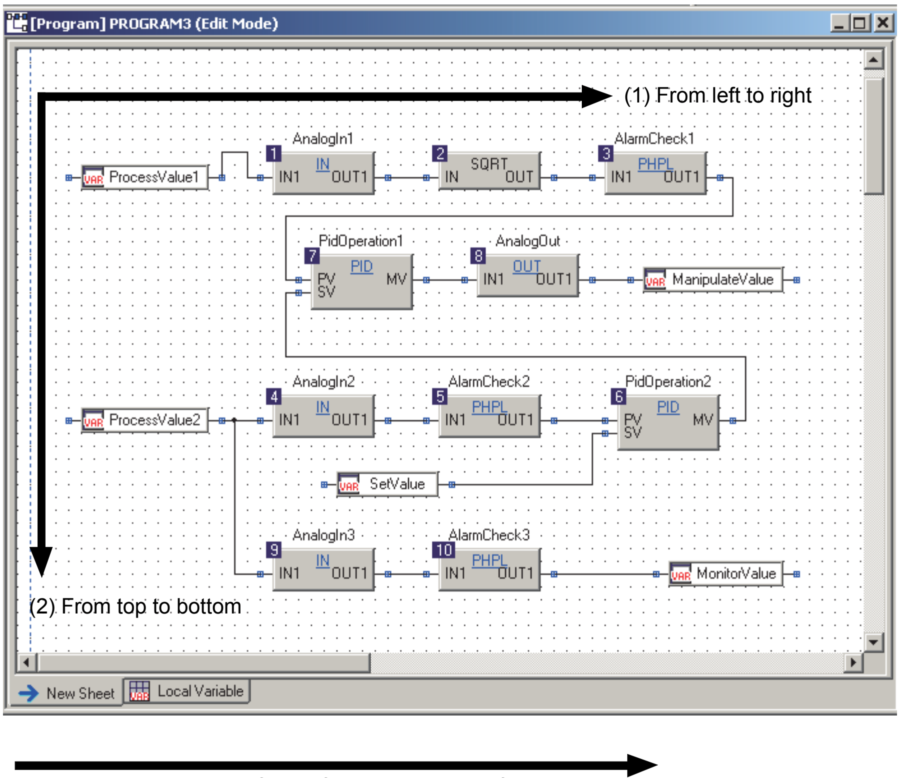


After execution condition setting for each FBD sheet, these sheets will be executed one by one from left to right if condition is satisfied.

2.12.3 Executing Order of FBD Parts

Inserting, arranging and connecting various FBD parts on FBD sheet shall be executed in the sequence from (1) to (3) as described in the following graph.

(Example 1) Display execution order of FBD parts



(3) From the FBD sheet of the left tabs to the one of the right

POINT

(1) For all the parts arranged on a FBD sheet, variable/FB parts shall have specified variable name/data type; constant parts should have specified value/data type.

(2) FBD parts to be input with value shall be executed after executing the FBD parts that are connected to the input pin. (Whether inputting value is finished shall be assessed in order from top to bottom)

(3) The execution order of FBD parts inserted and connected in FBD sheet can be confirmed in [Display Execution Order of FBD parts] of programming tool. For details, refer to PX Developer Version 1 Operating Manual (Programming Tool).

(4) Recursive call and closed loop are not allowed.

2.13 Identifier and Reserved Words

(1) Identifier
Identifier is used in the names of various programming tool members (such as variable names, FB variable names, structure type names etc.)
Qualified identifier should meet the following requirements.
(a) Specify with a string of less than 32 characters.
(b) Do not use reserved words. (Refer to (2) below)
(c) Use the following specified characters:
Alphabet number, underscore (_), backslash (\ ) *1.
*1 It can be used as a part of variable names only for device variables.
(Example: U□\G□)
(d) Do not use underscore (_) at the end.
Do not use two underscores or more continuously.
(e) Do not use space.
(f) Do not use a number as initial character.
(g) Constant is not allowed.
(For identifier that starts with "H" or "h", if "H" or "h" is followed by consecutive hexadecimal (0 to F) (up to 9 digits including "H" or "h", (excluding 0 that directly follows H/h) are processed as a constant. (Ex:"hab0"))
(h) Names with elementary data type cannot be used.
(i) Function/FB parts names are not allowed.

(2) Reserved words
This is the invalid character strings for identifier when make program by programming tool.
The reserved words are character strings that are used by system and cannot be used as identifier.
Error occurs only when identifier is identical with the reserved words listed as below, independently of the upper/lower case.

	Reserved words (in alphabetical order)
A	ACTION, ANY, ANY_BIT, ANY_DATE, ANY_DERIVED, ANY_ELEMENTARY, ANY_INT, ANY_MAGNITUDE, ANY_NUM, ANY_REAL, ANY_SIMPLE, ANY_STRING, ARRAY, AT
B	BOOL, BY, BYTE
C	CASE, CONFIGURATION, CONSTANT
D	DATE, DATE_AND_TIME, DEVICE, DINT, DO, DS, DT, DWORD
E	ELSE, ELSIF, ELSEIF, EN, END_ACTION, END_CASE, END_CONFIGURATION, END_FOR, END_FUNCTION, END_FUNCTION_ BLOCK, END_IF, END_PROGRAM, END_REPEAT, END_RESOURCE, END_STEP, END_STRUCT, END_TRANSITION, END_TYPE, END_VAR, END_WHILE, ENO, EXIT
F	FALSE, F_EDGE, FOR, FROM, FUNCTION, FUNCTION_BLOCK
I	IF, INT, INITIAL_STEP
L	LINT, LREAL, LWORD
O	OF, ON
P	PDD, PROGRAM
R	READ_ONLY, READ_WRITE, REAL, R_EDGE, REPEAT, RETAIN, RETURN, RESOURCE
S	SINT, STEP, STRING, STRUCT
T	TASK, THEN, TIME, TIME_OF_DAY, TO, TOD, TRANSITION, TRUE, TYPE
U	UDINT, UINT, ULINT, UNTIL, USINT
V	VAR, VAR_ACCESS, VAR_CONSTANT, VAR_EXT, VAR_EXTERNAL, VAR_EXTERNAL_CONSTANT, VAR_DEVICE, VAR_EXTERNAL_FB, VAR_ EXTERNAL_PG, VAR_GLOBAL, VAR_GLOBAL_CONSTANT, VAR_GLOBAL_FB, VAR_GLOBAL_PG, VAR_IN_OUT, VAR_INPUT, VAR_OUTPUT, VAR_PUBLIC, VAR_TEMP
W	WORD, WHILE, WITH, WSTRING

2.14 Manufacturer Library

Library list provided by the manufacturer is as follows.
The following function/FB parts will be shown in parts window of programming tool.
For more details about each library, refer to "Reference".

Library classification		Command classification	Reference
General function (with EN/ENO)		Type conversion function (with EN/ENO)	Chapter 4
		Numerical operation function (with EN/ENO)	
		Arithmetic operation function (with EN/ENO)	
		Bit-string function (with EN/ENO)	
		Logical operation function (with EN/ENO)	
		Selection function (with EN/ENO)	
		Comparison function (with EN/ENO)	
		Character string function (with EN/ENO)	
		Helper function (with EN/ENO)	
		Ladder program control function (with EN/ENO)	
General FB		Bistable FB	Chapter 5
		Edge detection FB	
		Counter FB	
		Timer FB	
		Communication control FB	
Process function (with EN/ENO)		Analog value selection and average value function (with EN/ENO)	Chapter 6
Process FB	General process FB	Correction operation FB	Chapter 7
		Arithmetic operation FB	
		Comparison operation FB	
		Control operation FB	
	Tag access FB	Input/output control FB	Chapter 8
		Loop control FB	
		Special FB	
	Tag FB	Loop tag FB	Chapter 9
		Status tag FB	
		Alarm tag FB	
		Message tag FB	
Module FB		Analog module FB	Chapter 10
		Temperature input module FB	
		Pulse input module FB	
		Digital input module FB	
		CC-Link module FB	

POINT

- Whether to continue/stop CPU processing when operation error occurs is based on the PLC parameter settings.
- When operation error occurs on the function, the output of corresponding function becomes undefined value.
- When operation error occurs on FB, the output of corresponding FB holds the previous value.

2.15 Precautions When Using GX application

The following provides the precautions when using GX application.

POINT

Handle the PX Developer project using GX application while paying attention to precautions given in the "PX Developer Version 1 Operating Manual (Programming Tool)".

2.15.1 GX application/PX Developer Version

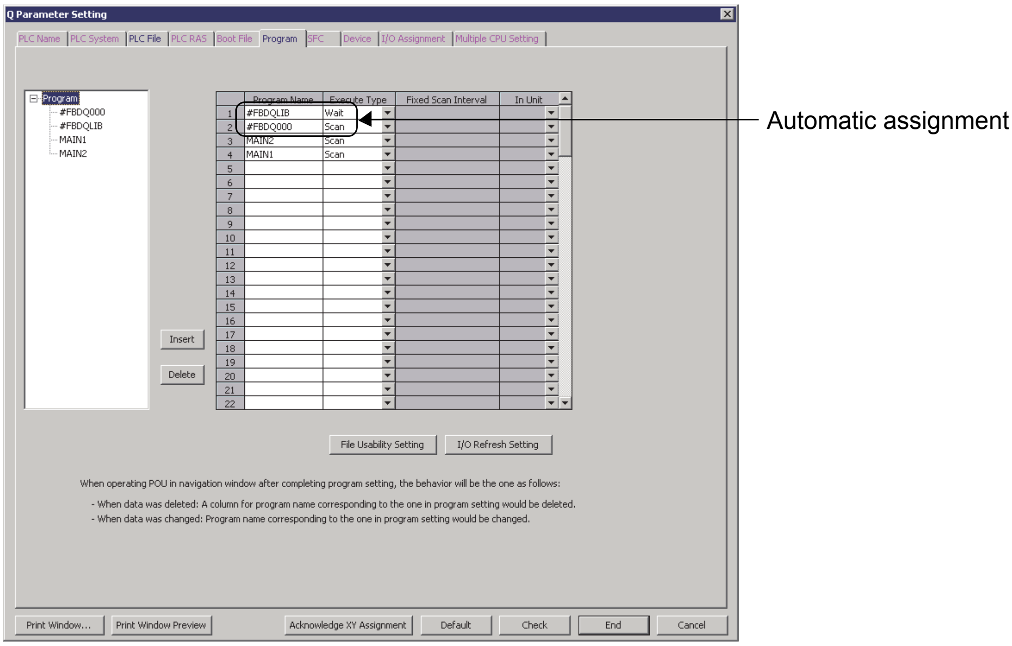
GX application is required to use the PX Developer programming tool.
Note that, the combination of GX application and PX Developer varies with the PLC type, used function or network configuration. For details, refer to "PX Developer Version 1 Operating Manual (Programming Tool)".

2.15.2 PLC Parameters

(1) About program setting of PLC parameter

(a) Program name and execution type on a user-created ladder must be set in program setting of PLC parameter.

(b) With the compile functions of programming tool, FBD programs can be converted into ladder program files (#FBDQ □) for its own use, and program setting of PLC parameter can be automatically assigned. (*1)
When FBD programs and user ladders are mixed together, FBD programs are first recompiled with the programming tool, then automatically assigned at the end of the program setting.
However, when the "#FBDQ…" files are assigned in PLC parameter, the order of the program setting will not be changed even if the programs are recompiled. (The "#FBDQ…" files will not move to the last.)



*1 According to the execution type of program execution, FBD programs automatically create files by following methods.

Execution type	PC parameter program setting	
	Program name	Execution type
Scan	#FBDQ000	Scan
Stand-by	#FBDQLIB	Wait
Fixed scan starting	#FBDQ***	Fixed scan

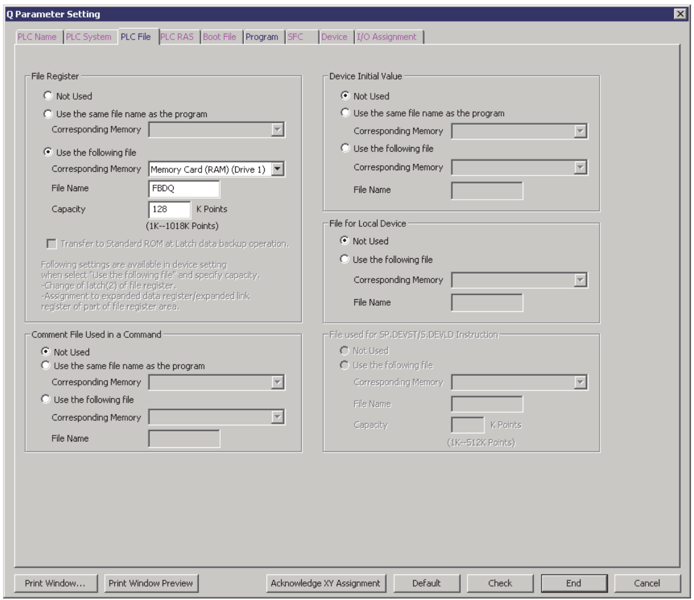
- In "#FBDQ***", *** is assigned with a consecutive number starting from 001 during each compile.

POINT

Make sure no changes are made on setting screen of the FBD program (#FBDQ□) automatically assigned by PLC parameter program setting.

(2) About PLC file setting of PLC parameter

(a) File registers used in FBD programs are automatically set with programming tool.



(b) Through PLC file setting of PLC parameter of GX application, the "Corresponding memory", "File name", and "Capacity" of the currently used file register can be changed.
File used can be chosen from "Use the following file".

IMPORTANT

The number of the file register compiled in target memory must be set to 1.
If a file register already exists in the target memory, it should be deleted before file register of other name is written in.

POINT

During hot-start compile, do not change PLC parameter file register setting.
When PLC parameter file register setting is changed, do not execute downloading after hot-start compile or online change compile.
For details of hot-start compile or online change compile, refer to Section 2.2.5.

(3) PLC system setting using PLC parameters

(a) Keep the low speed timer limit to 100ms in "Timer limit setting".

(b) Set "Common pointer No." within the range of 0 to 3500.
(The default value is 3500.)

2.15.3 Ladder Programming

Precautions for various parameter settings, etc. made by user ladder creation or using GX application will be explained.

(1) Creation of user ladder or settings using GX application

When creating user ladder*1 or settings using GX Works2 or GX Developer and GX Configurator, start the GX project from the project window of the programming tool.

*1 Including comment and device memory editing.

(2) Devices that must not be rewritten from user ladder

The following devices can be read /written during program execution on programming tool.
Do not change the value of these devices from global variable or user ladder.

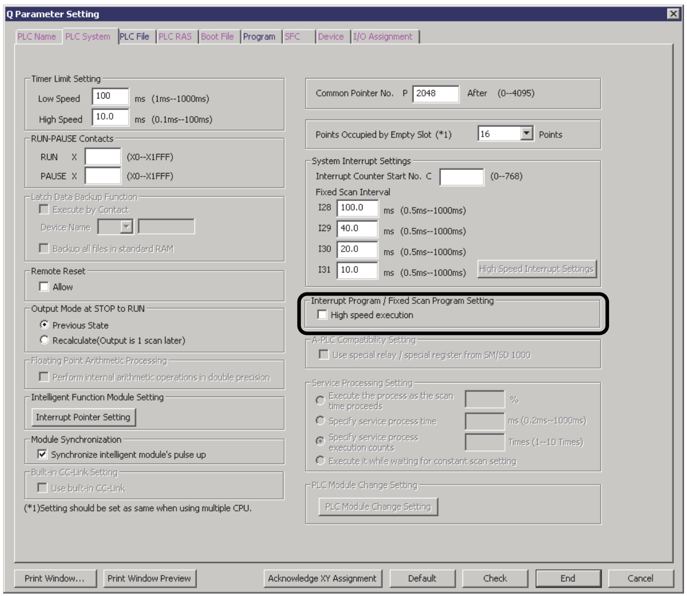
Devices used in programming tool	Change-forbidden range of device value
ZR (or R)	Range set with the system resource in the project parameter setting*1. (However, the items of tag data*2 within the range can be changed by specifying with ZR.)
T	Range set with the system resource in the project parameter setting*2
P	P3500 to P4095
M	Range set with the system resource in the project parameter setting*2
Z	Z0 to Z6 (However, the device value can be changed when the check box of "High speed execution" is cleared in "interrupt program/Fixed scan program setting" of PLC parameter of GX application.) *3
SD	SD0 to SD3 SD5 to SD8 SD16 to SD19 SD203 SD1500 to SD1501 SD1502 to SD1505
SM	SM1 SM390 SM701 SM1500, SM1501, SM1552 to 1583

*1 Refer to "PX Developer Version 1 Operating Manual (Programming Tool)".
*2 The device assigned to tag data can be checked from the tag FB declaration window of the programming tool.
*3 For details, refer to (3) in this section.

(3) Precautions on using index registers

The following are the considerations for index registers in a user ladder.

(a) Uncheck the "High speed execution" item under "Interrupt Program/Fixed Scan Program Setting" on the <<PLC System>> of the PLC parameter in GX application. (Default setting: unchecked)

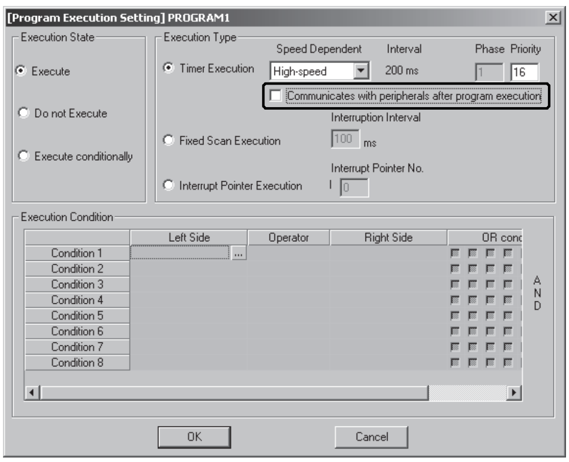


(b) When the CPU is switched from STOP to RUN again immediately after the CPU is stopped by an error, values of index register become undefined values. Initialize index registers correctly through the SM402 (after RUN, ON for 1 scan) contact.

(c) When the "Communicates with peripherals after program execution" item is checked (default setting: checked) in the "Program Execution Setting" dialog box of PX Developer, the operation is as shown below.

- Values of index register cannot be monitored normally.
- When the value of index register is changed with device test, etc., the operation of FBD program will be incorrect.

In this case, uncheck the "Communicates with peripherals after program execution" item for all FBD programs. However, the monitoring performance of peripherals (such as monitor tool) may become low.



(4) Instructions that must not be used in user ladder

The QDRSET(P) instruction (setting of file for file register) must not be included in the user ladder. If included, FBD program will not normally operate when the file for file register is renamed by the QDRSET(P) instruction.

(5) Number of user ladders that can be created

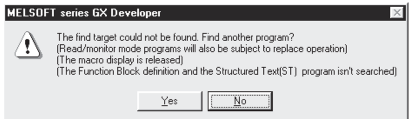
Up to 122 ladder programs can be created. However, if the fixed scan type program is created using the programming tool, this will affect the number of ladder programs, i.e., it will decrease by the number of the created fixed scan type programs.

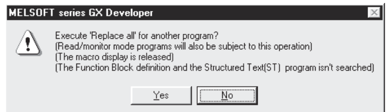
(6) Ladder program that must not be edited in GX Developer project

In the GX Developer project started from the programming tool, do not edit or delete the ladder program "#FBDQ***".
If it is edited or deleted, FBD programs will not operate normally.

IMPORTANT

If Replace device is performed for any program other than "#FBDQ***" in Replace device on GX Developer, the devices in the "#FBDQ***" program may also be replaced.





If the above dialog box is displayed in Replace device, click the "No" button.
If the "Yes" button is clicked, the devices in "#FBDQ***" are also replaced and FBD programs will not operate normally.
If the devices in "#FBDQ***" have been replaced by clicking the "Yes" button, re-compile the project again with the programming tool.

(7) Compatibility with the function block of GX application

There is no compatibility between the function block that can be created by label programming of GX application and the function block used with the programming tool.

(8) Overwrite by global variable (label) setting of GX Developer at compile

When the project is compiled with the programming tool, the settings made in the GX Developer label assignment window of the programming tool are overwritten by the global variable (label) setting of the GX Developer project.

At this time, when Auto External setting*1 has been performed in the global variable (label) setting of the GX Developer, the settings are also overwritten by the local variables (labels) of the reflection destination.*2

*1 Auto External setting reflects the settings made by the global variable (label) setting of the GX Developer on all local variable (label) settings or the specified local variable (label) setting.

*2 The settings are automatically overwritten only when using PX Developer Version 1.04E or later and GX Developer Version 8.03D or later.

(9) Auto device setting of GX application

Set the ranges of the devices automatically assigned to labels by GX application so that they do not overlap the devices that must not be rewritten from user ladder indicated in (2) in this section.

POINT

When the device range (file register: ZR, timer: T, Internal relay: M) of the system resource has been changed in the project parameter setting of the programming tool, confirm the contents of the automatic assignment device setting of GX application in the following procedure.

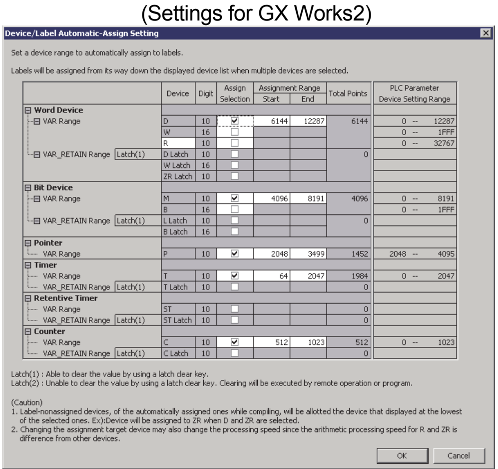
If it overlaps the assignment range of the automatic assignment device, change the setting so that it does not overlap the assignment range.

[Setting procedure]

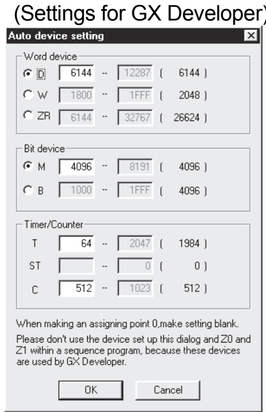
1. Open the GX project from the programming tool.
2. Select [Device/Label Automatic-Assign Setting]/[Auto device setting] menu of GX application. *1
   For GX Developer, display the Global variable (label) setting window before the operation.
3. Set the device range in the Auto device setting window so that it does not overlap the device ranges of the device that must not be rewritten from user ladder, indicated in (2) in this section, and the project parameter setting.

*1 For GX Works2, select [Tool] → [Device/Label Automatic-Assign Setting], as for GX Developer, select [Edit] → [Auto device setting].

(Settings for GX Works2)



(Settings for GX Developer)



(10) Simulation of FBD program

For simulation of FBD program, the following are required.

- When GX project type is GX Works2 project
  <When Process CPU or Redundant CPU is selected as PLC type>
  GX Works2 Version 1 (SW1DNC-GXW2 Version 1.98C or later)
  <When Universal model process CPU is selected as PLC type>
  GX Works2 Version 1 (SW1DNC-GXW2 Version 1.501X or later)
- When GX project type is GX Developer project
  GX Developer Version 8 (SW8D5C-GPPW 8.94Y or later)
  GX Simulator Version 7 (SW7D5C-LLT 7.27D or later)

For details, refer to "PX Developer Version 1 Operating Manual (Programming Tool)".

(11) Auto refresh by intelligent function module operation when module FB is used

In the module where the module FB performs data I/O processing, do not use the auto refresh function for the PLC devices using GX Works2 or GX Configurator.
If the auto refresh function is used, the output value of the module FB will be illegal.

2.15.4 Redundant Parameters

Redundant parameters are for continuing the system control by switching system (system A ↔ system B) when the system control by Redundant CPU fails.
The following shows the redundant parameters.
Set either of them using GX application. (Batch setting is not allowed for internal devices in the tracking setting of redundant parameter.)

Menu	Parameter	Parameter outline	Reference
Redundant parameter	Operation mode setting	Makes the settings for power-on and debug mode/backup mode.	GX Works2 Version 1 Operating Manual (Common) GX Developer Version 8 Operating Manual
	Tracking setting	Transfers system information and device memory information used to continue the system control by Redundant CPU at the time of system switching from control system to standby system.	(1) to (4) in this section GX Works2 Version 1 Operating Manual (Common) GX Developer Version 8 Operating Manual

(1) Tracking device setting

Tracking target device range is divided as follows.

(a) Tracking device range needs to be set manually

The following device rages must be set manually.

Tracking block No.	Tracking target device range	Remark
Either of No.1 to 32	Output Y device	Transfer triggers (SN1520 to SM1551) corresponding to tracking block No.1 to 32 need to be ON in user ladder program.*2
	B device, W device for host station transmission	
	Various devices used in user ladder program*1	

*1 The device range to be set as the PX Developer system resource is extruded.

*2 For tracking block No.1, automatic transfer (SM1520 automatically turns ON) can be selected.

(b) Tracking device range to be automatically set by the programming tool

The following are automatically registered in the tracking setting of redundant parameter of GX application.

Tracking block No.	Tracking target device range	File register file setting	Remark
No. 33 to 36	None (setting is cleared)	None (setting is cleared)	Reserved for future use of PX Developer
No.64	System resource device range (ZR device*1, T device)	File register target memory and file name set in the PLC parameter	Tracking is performed in backup mode only.

*1 The device range that stores the information peculiar to system A/system B (project identification code) is extruded.

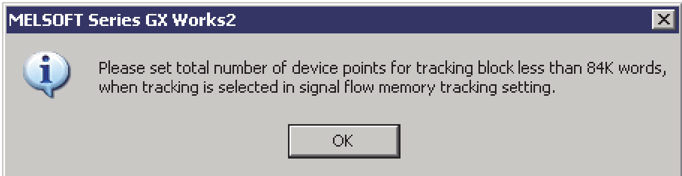
(2) Tracking setting for signal flow (rising/falling execution instruction history)

Basically, tracking of signal flow is not required when FBD program is executed using PX Developer.

However, make the settings if tracking of signal flow is required to execute user ladder program.

POINT

When tracking of signal flow is selected in the redundant parameter setting window of GX application, the tracking block No.64 device range setting may appear with the following dialog box.



In this case, make the settings in order that the number of tracking points within the system resource device range will be 84kwords or less, as shown below.

[Setting procedure]

1. Open the project parameter setting window from the programming tool and select <System resource> tab.
2. Decrease the number of system resource device points (ZR device, T device) so that the following condition will be satisfied.

   {(Number of ZR device setting points)
   + (Number of T device setting points) x 9/8} ≦ 86016

3. Execute a compile using the programming tool.
4. Open the redundant parameter setting window of GX application after the compile is completed, and select the tracking setting of signal flow again.

(3) Adjustment for tracking time reduction

Scan time is extended by the following time according to the number of device points in the programming tool system resource.*1

Reference of tracking time within the system resource device range {(Number of ZR device setting points) + (Number of T device setting points) x 9/8} x 0.35*2µs

*1 Actual total delay time due to tracking increases by the device range (tracking block No.1 to 32) used for user ladder program and transfer time of signal flow or the like.

*2 Indicates the time constant when creating the file register within standard RAM.

The delay of scan time can be decreased by decreasing the number of system resource device points as shown on the next page. However, note that insufficient system resource may cause a compile error.

[Adjustment procedure]

1. Reduce "Maximum No. of Tags" in the tag FB window to the minimum number required.*3
2. Execute a compile and confirm the number of remaining system resource points displayed in the output window.
3. Open the project parameter setting window by referring to the number of remaining system resource points, select <System resource> tab, and reduce the number of system resource device points (ZR device, T device) to the minimum number required. *4

*3 When changing "Maximum No. of Tags", make sure to consider the tags required for future use, as this requires cold start compile.

*4 When changing the number of system resource device points, make sure to consider the device points required for future use, as read to CPU during RUN will be disabled.

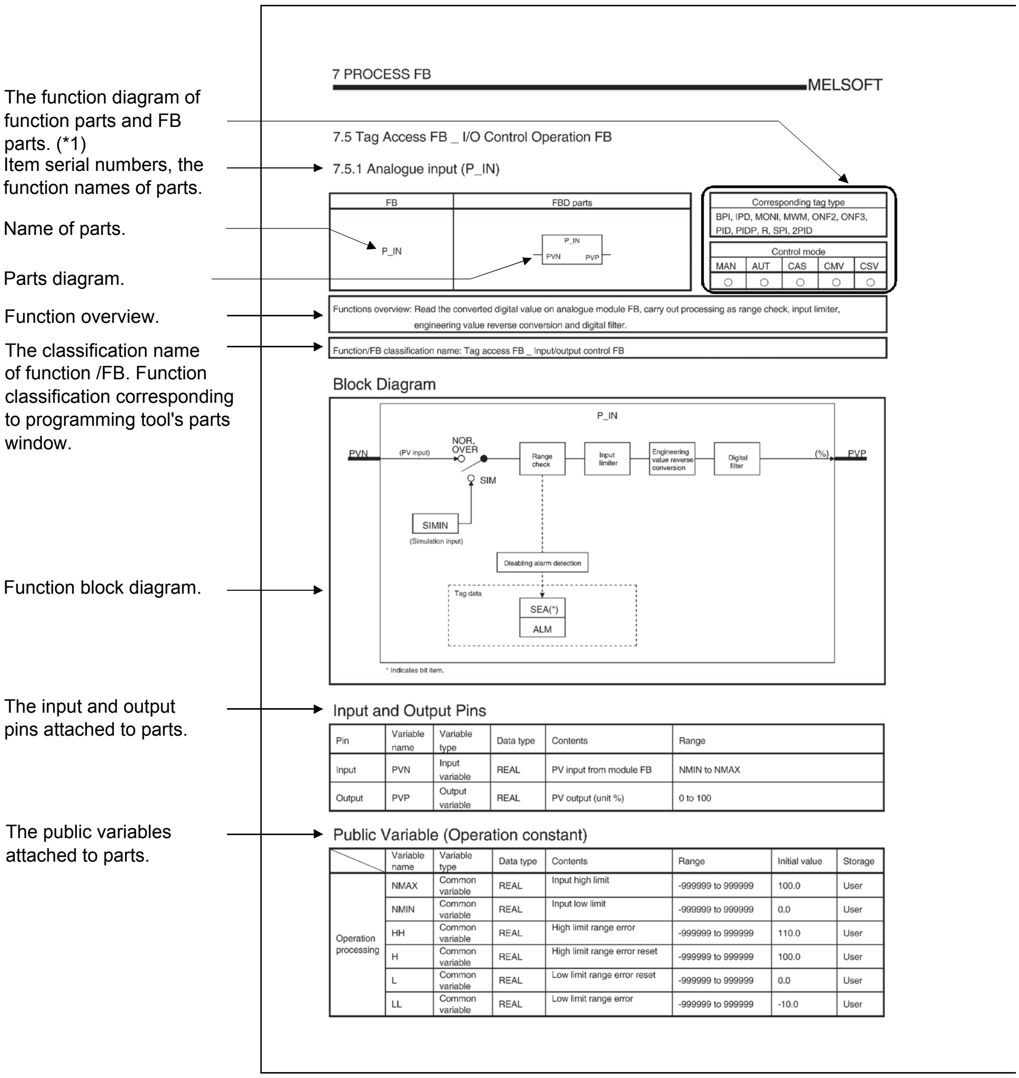
(4) Tracking transfer mode setting

By default, the tracking transfer mode is set to tracking synchronous mode. Switch to program priority mode as necessary.

MEMO

3 ABOUT COMPREHENDING FUNCTION PARTS AND FB PARTS

From next chapter on, function parts and FB parts are illustrated through the following format:



7 PROCESS FB MELSOFT

7.5 Tag Access FB _ I/O Control Operation FB

7.5.1 Analogue input (P_IN)

FB	FBD parts
P_IN	P_IN

Corresponding tag type
BPI, IPD, MONI, MWM, ONF2, ONF3, PID, PIDP, R, SPI, 2PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Read the converted digital value on analogue module FB, carry out processing as range check, input limiter, engineering value reverse conversion and digital filter.

Function/FB classification name: Tag access FB _ Input/output control FB

Block Diagram

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	PV input from module FB	NMIN to NMAX
Output	PVP	Output variable	REAL	PV output (unit %)	0 to 100

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Common variable	REAL	Input high limit	-999999 to 999999	100.0	User
	NMIN	Common variable	REAL	Input low limit	-999999 to 999999	0.0	User
	HH	Common variable	REAL	High limit range error	-999999 to 999999	110.0	User
	H	Common variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	L	Common variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	LL	Common variable	REAL	Low limit range error	-999999 to 999999	-10.0	User

*1 Shows the function parts and FB parts correspond to each function.

(a) General function/process function (☞ Chapter 4 and Chapter 6)
For details of function, refer to Section 2.7.

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	2 to 8

1) Shows functions with EN/ENO pins.
2) Shows overload functions.
3) Shows the input pin changeable range of input pin changeable function.

7 PROCESS FB
MELSOFT

Public Variable (others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User

*1 Please read and write them with program.
It will not be displayed in FB property window of PX Developer.
*2 It means simulation processing.

Tag Data — Parts have tag data

Please refer to the tag data list of tag type in Appendix 1.1 about tag data which is written/ read in the tag access FB.

Function — Functions shown in the block diagram.

Item	Contents

Other Functions — Functions not shown in the block diagram.

Item	Contents

Processing Operation — Function operations determined by control mode.

Control mode \ Processing	Range check	Input limiter	Engineering data reverse conversion	Digital filter	Alarm
MAN, CMV, AUT, CAS, CSV	○	○	○	○	○ (*1)

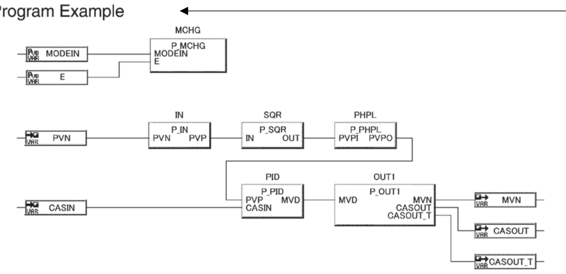
○ : Execute ×: Not execute

*1 When the bit of "Disable Alarm Detection" (INH) which corresponds to an alarm in TRUE, the alarm is not detected.

Error — Error contents and error codes.

Error may occur in the following case, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
Please refer to Appendix 2 for details of process control error code.
- When overflow occurs during operation. (Error code: 4100)

Program Example — Sample programs of parts.



(b) Tag access FB/tag FB (☞ Chapter 8 to Chapter 9)
For details of tag access FB/tag FB, refer to Section 2.10.
1) Shows the tag type corresponding to tag access FB/ tag FB.
For the tag type corresponding to tag access FB/ tag FB, refer to Appendix 1.3.
2) Shows the selectable control mode.
However, the module is not available when the corresponding bit of "Disable Mode Change (MDIH)" of the tag data is TRUE.
For control mode, refer to Appendix 3.8.
Cascade direct (CASDR) mode of control mode is selectable when the tag type is 2PIDH.

Corresponding tag type
BPI, IPD, MONI, MWM, ONF2, ONF3, PID, PIDP, R, SPI, 2PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

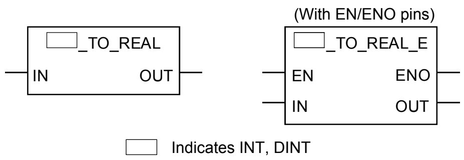
4 GENERAL FUNCTION

General functions are classified as follows.

Classification name	Description	Reference
Type conversion function	Conversion among data types	Section 4.1
Numerical Operation	Output the absolute value, square root, operation results of (natural logarithm, common logarithm and natural exponential), ASIN value, ACOS value, and ATAN value.	Section 4.2
Arithmetic operation function	Output the sum, product, difference, quotient and remainder of the input value.	Section 4.3
Bit-string function	Shift or rotate the input value to the left or right by n bits.	Section 4.4
Logical operation function	Output the AND, OR, XOR and NOT of the input value.	Section 4.5
Selection function	Select the output method for input value	Section 4.6
Comparison function	Output the comparison results of the input data value	Section 4.7
Character string function	Execute string length detection, middle character, concatenation, inserting, deleting, replacing and searching for characters.	Section 4.8
Helper function	Select the output methods for various data types	Section 4.9
Ladder program control function	Sub-routine program call, program scan execution registration, program standby execution, program output OFF standby instruction, program low-speed execution registration.	Section 4.10

4.1 Type Conversion Function

4.1.1 INT/DINT Type → REAL Type Conversion (INT_TO_REAL(_E), DINT_TO_REAL(_E))

Function	FBD parts
INT_TO_REAL DINT_TO_REAL INT_TO_REAL_E DINT_TO_REAL_E	

With EN/ENO pins	○
Overload	—
Input pin number changeable (range)	—

Function overview: Converts data type INT/DINT to REAL.

Function/ FB classification name: Type conversion function

4

Input and Output Pins

(1) INT_TO_REAL INT_TO_REAL_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	INT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	REAL	Output

(2) DINT_TO_REAL DINT_TO_REAL_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DINT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	REAL	Output

Function

Item	Contents
Operation processing	(1) INT_TO_REAL INT_TO_REAL_E (a) Convert the INT type data input from input variable IN to REAL type data, and output from output variable OUT. $\underbrace{\boxed{1234}}_{\text{INT type}} \Rightarrow \underbrace{\boxed{1234.0}}_{\text{REAL type}}$ (b) The value input from input variable IN is INT type data value. (2) DINT_TO_REAL DINT_TO_REAL_E (a) Convert the DINT type data input from input variable IN to REAL type data, and output from output variable OUT. $\underbrace{\boxed{16543521}}_{\text{DINT type}} \Rightarrow \underbrace{\boxed{16543521.0}}_{\text{REAL type}}$ (b) The value input from input variable IN is DINT type data value. (c) REAL type data is processed by 32 bits single precision, so its valid digits are about 7. Therefore, when integer value exceeds the range of –16777216 to 16777215, the converted value error will occur. (Rounding error)

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>(1) Functions without EN/ENO pins
Execute operation processing. Output the operation output value from OUT pin.

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

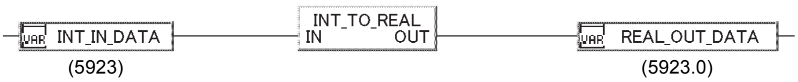
Error

There is no operation error caused by INT_TO_REAL(_E), DINT_TO_REAL(_E).

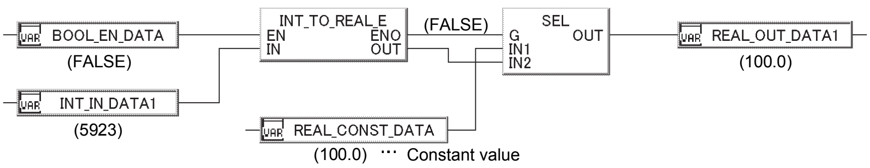
Program Example

(1) The program that converts INT type data input from input variable IN to REAL type data, and output from output variable OUT.

(a) Basic program example (INT_TO_REAL)

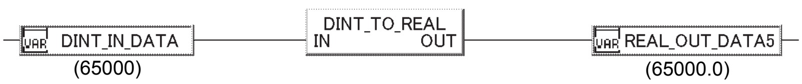


(b) The program example that outputs constant value when input variable EN is FALSE, (INT_TO_REAL_E)



(2) The program that converts DINT data input from input variable IN to REAL type data, and output from output variable OUT.

(a) Basic program example (DINT_TO_REAL)



4.1.2 INT Type → DINT Type Conversion INT_TO_DINT(_E))

Function	FBD parts
INT_TO_DINT INT_TO_DINT _E	INT_TO_DINT: IN → OUT (With EN/ENO pins) INT_TO_DINT_E: EN → ENO, IN → OUT

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Converts data type INT to DINT.

Function/ FB classification name: Type conversion function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	INT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DINT	Output

Function

Item	Contents
Operation processing	(1) Convert the INT type data input from input variable IN to DINT type data, and output from output variable OUT. 1234 (INT type) ⇒ 1234 (DINT type) (2) The value input from input variable IN is INT type data.
Operation results	(1) Functions without EN/ENO pins Execute operation processing. Output the operation output value from OUT pin. (2) Functions with EN/ENO pins The execution conditions and the operation results are as follows:

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

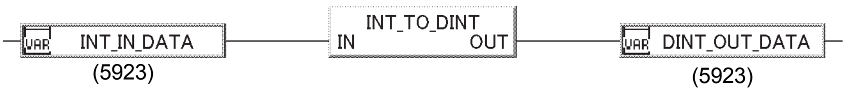
Error

There is no operation error caused by INT_TO_DINT(_E).

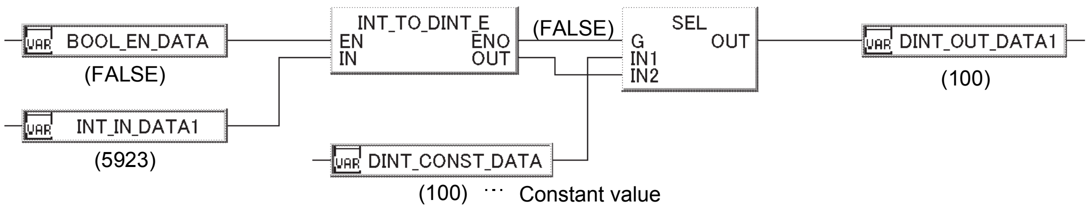
Program Example

The program that converts the INT type data input from input variable IN to DINT type data, and output from the output variable OUT.

(1) Basic program example (INT_TO_DINT)



(2) The program example that outputs constant value when input variable EN is FALSE, (INT_TO_DINT_E)



4.1.3 DINT Type → INT Type Conversion (DINT_TO_INT(_E))

Function	FBD parts
DINT_TO_INT DINT_TO_INT_E	DINT_TO_INT: IN → OUT (With EN/ENO pins) DINT_TO_INT_E: EN → ENO, IN → OUT

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Converts data type DINT to INT.

Function/FB classification name: Type conversion function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DINT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output

Function

Item	Contents
Operation processing	(1) Convert the INT type data input from input variable IN to DINT type data, and output from output variable OUT. 1234 (DINT type) ⇒ 1234 (INT type) (2) The value input from input variable IN is INT type data.
Operation results	(1) Functions without EN/ENO pins The operation results are as follows: (see table A) (2) Functions with EN/ENO pins The execution conditions and the operation results are as follows: (see table B) * When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Operation result	OUT
No operation error	Operation output value
Operation error occur	Undefined value

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error occur) (*)	Undefined value
FALSE (Operation stop)	FALSE (*)	Undefined value

Error

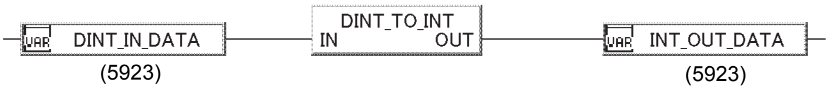
Error may occur in the following case, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When input value is not in the range of –32768 to 32767. (Error code: Refer to Appendix 2)

Program Example

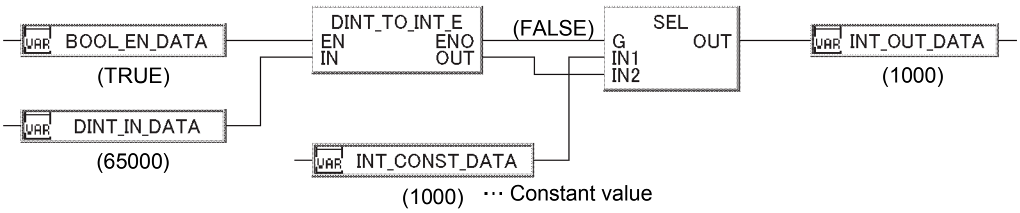
The program that converts DINT type data input from input variable IN to INT type data, and output from output variable OUT.

(1) Basic program example (DINT_TO_INT)

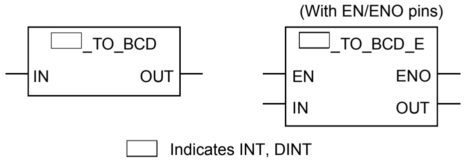


(2) The program example that outputs constant value when input variable EN is FALSE, or operation errors occur, (DINT_TO_INT_E)

(Example) When operation errors occur



4.1.4 INT/DINT Type → BCD Type Conversion (INT_TO_BCD(_E), DINT_TO_BCD(_E))

Function	FBD parts
INT_TO_BCD DINT_TO_BCD INT_TO_BCD_E DINT_TO_BCD_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Converts data type INT/DINT to REAL.

Function/ FB classification name: Type conversion function

Input and Output Pins

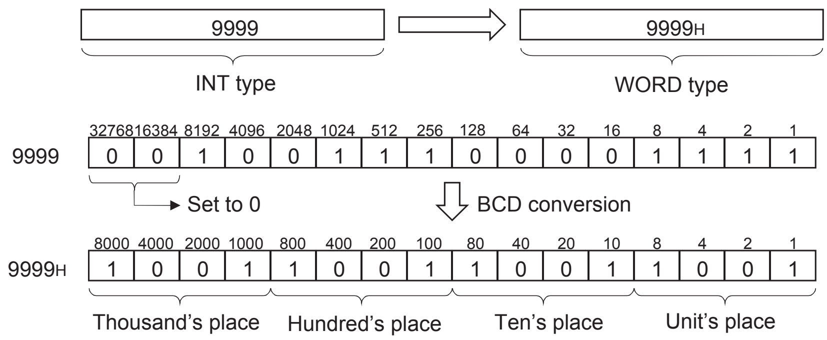
(1) INT_TO_BCD INT_TO_BCD_E

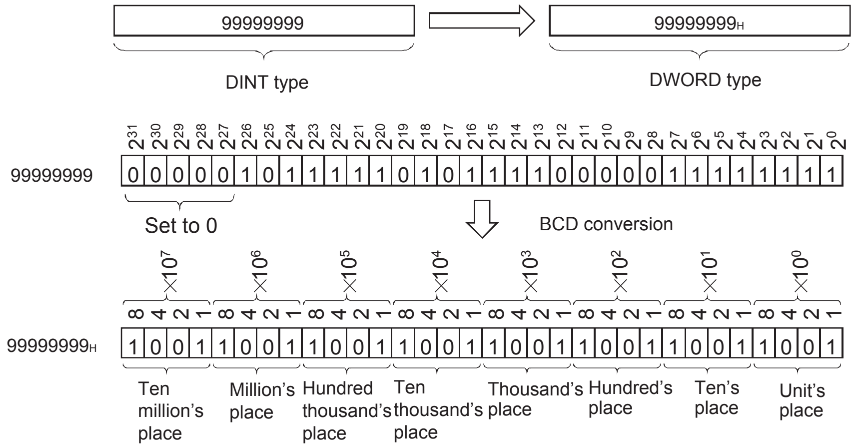
Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	INT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	WORD	Output

(2) DINT_TO_BCD DINT_TO_BCD_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DINT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DWORD	Output

Function

Item	Contents
Operation processing	(1) INT_TO_BCD INT_TO_BCD_E (a) Convert the INT type data input from input variable IN to BCD type data, and output the data from output variable OUT.  (b) The value input from input variable IN is INT type data value, and its range is 0 to 9999.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(2) DINT_TO_BCD DINT_TO_BCD_E
(a) Convert the DINT type data input from input variable IN to BCD type data, and output the data from output variable OUT.


(b) The value input from input variable IN is DINT type data value, and its range is 0 to 99999999.
</td></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occur</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occur) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

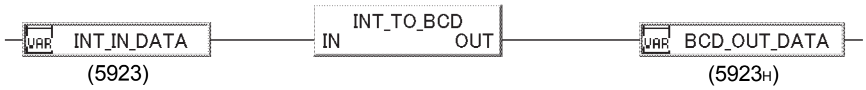
Error may occur in the following case, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- INT_TO_BCD_(_E)): When input value is out of 0 to 9999 range (Error code: Refer to Appendix 2)
- DINT_ TO_BCD_(_E)): When input value is out of 0 to 99999999 range (Error code: Refer to Appendix 2)

Program Example

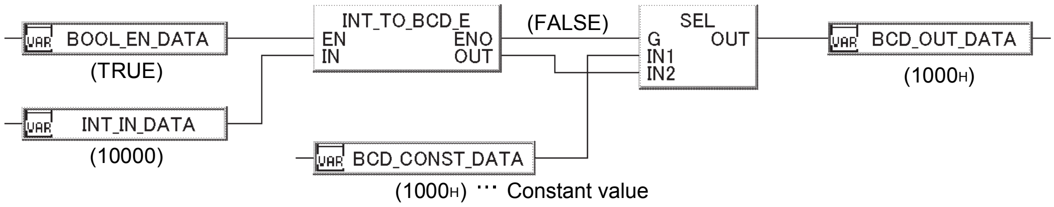
(1) The program that converts INT type data input from input variable IN to BCD type data, and output the data from output variable OUT.

(a) Basic program example (INT_TO_BCD)



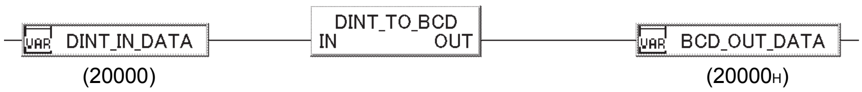
(b) The program example that outputs constant value when input variable EN is FALSE, or operation errors occur, (INT_TO_BCD_E)

(Example) When operation errors occur

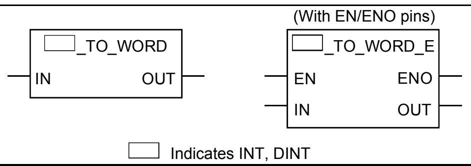


(2) The program that converts DINT type data input from input variable IN to BCD type data, and output the data from output variable OUT.

(a) Basic program example (DINT_TO_BCD)



4.1.5 INT/DINT Type → WORD Type Conversion (INT_TO_WORD(_E), INT_TO_WORD(_E))

Function	FBD parts
INT_TO_WORD DINT_TO_WORD INT_TO_WORD_E DINT_TO_WORD_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Converts data type INT/DINT to WORD.

Function/ FB classification name: Type conversion function

Input and Output Pins

(1) INT_TO_WORD INT_TO_WORD_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	INT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	WORD	Output

(2) DINT_TO_WORD DINT_TO_WORD_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DINT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	WORD	Output

Function

Item	Contents
Operation processing	(1) INT_TO_WORD INT_TO_WORD_E (a) Convert the INT type data input from input variable IN to WORD type data, and output the data from variable OUT. 22136 ⇨ 5678H INT type WORD type (b) The value input from input variable IN is INT type data value. (2) DINT_TO_WORD DINT_TO_WORD_E (a) Convert the DINT type data input from input variable IN to WORD type data, and output the data from output variable OUT. High-order 16 bit information of the input is abandoned. 12345678 ⇨ 614EH DINT type WORD type 12345678: 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 614EH: 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 High-order16 bit information is abandoned. (b) The value input from input variable IN is DINT type data value.

Item	Contents
Operation results	(1) Functions without EN/ENO pins Execute operation processing. Output the operation output value from OUT pin. (2) Functions with EN/ENO pins The execution conditions and the operation results are as follows:

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

POINT

Information on the high-order 16 bits of the DINT type data input from input variable IN is abandoned while executing DINT_TO_WOED_(E)

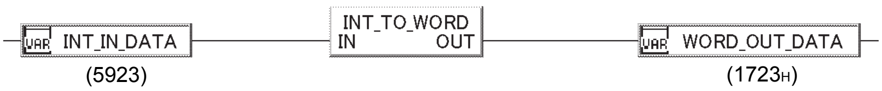
Error

There is no operation error caused by INT_TO_WORD(_E), DINT_TO_WORD(_E).

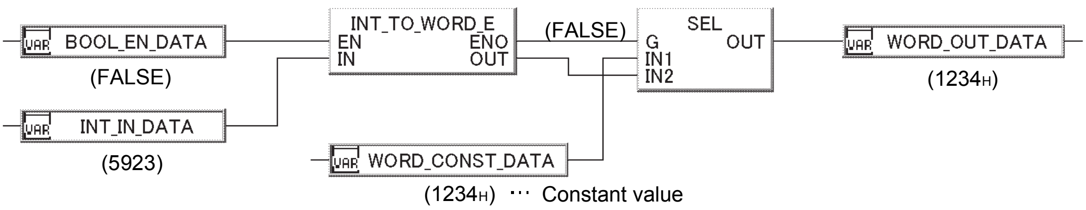
Program Example

(1) The program that converts INT data input from input variable IN into WORD data, and output the data from output variable OUT.

(a) Basic program example (INT_TO_WORD)

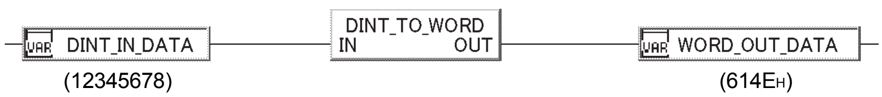


(b) The program example that outputs constant value when input variable EN is FALSE, (INT_TO_WORD_E)

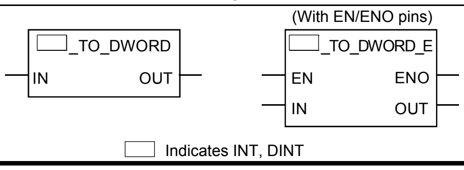


(2) The program that converts DINT data input from input variable IN into WORD data, and output the data from output variable OUT.

(a) Basic program example (DINT_TO_WORD)



4.1.6 INT/DINT Type → DWORD Type Conversion (INT_TO_DWORD(_E), DINT_TO_DWORD(_E))

Function	FBD parts
INT_TO_DWORD DINT_TO_DWORD INT_TO_DWORD_E DINT_TO_DWORD_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Converts data type INT/DINT to DWORD.

Function/ FB classification name: Type conversion function

Input and Output Pins

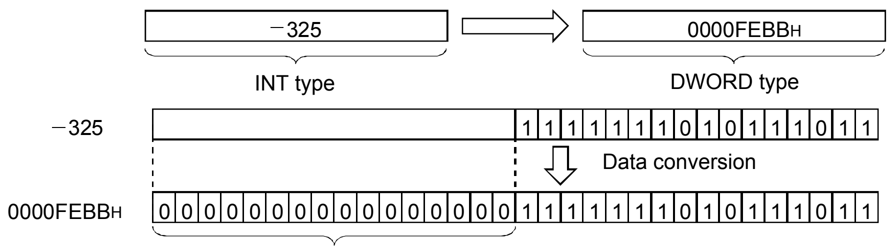
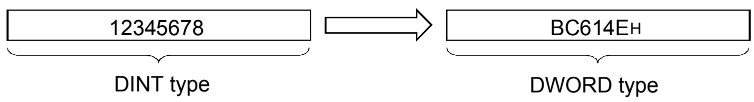
(1) INT_TO_DWORD INT_TO_WORD_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	INT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DWORD	Output

(2) DINT_TO_DWORD DINT_TO_DWORD_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DINT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DWORD	Output

Function

Item	Contents
Operation processing	(1) INT_TO_DWORD INT_TO_DWORD_E (a) Convert the INT type data input from input variable IN to DWORD type data, and output the data from output variable OUT.  When the data is converted, high-order 16 digits become 0. (b) The value input from input variable IN is INT type data. (2) DINT_TO_DWORD DINT_TO_DWORD_E (a) Convert the DINT type data input from input variable IN to DWORD type data, and output the data from output variable OUT.  (b) The value input from input variable IN is DINT type data.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
Execute operation processing. Output the operation output value from OUT pin.

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

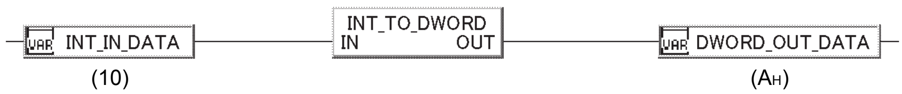
Error

There is no operation error caused by INT_TO_DWORD(_E), DINT_TO_DWORD(_E)

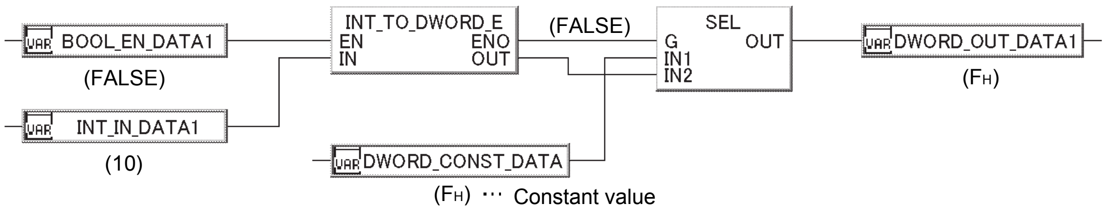
Program Example

(1) The program that converts INT type data input from input variable IN to DWORD type data, and output the data from output variable OUT.

(a) Basic program example (INT_TO_DWORD)

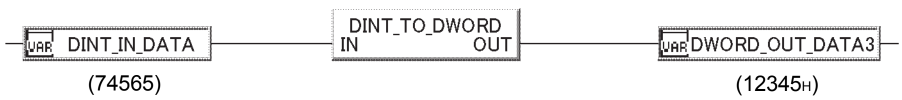


(b) The program example that outputs constant value when input variable EN is FALSE, (INT_TO_DWORD_E)

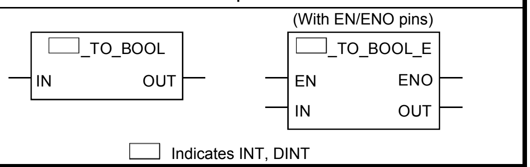


(2) The program that converts DINT type data input from input variable IN to DWORD type data, and output the data from output variable OUT.

(a) Basic program example (DINT_TO_DWORD)



4.1.7 INT/DINT Type → BOOL Type Conversion (INT_TO_BOOL(_E), DINT_TO_BOOL(_E))

Function	FBD parts
INT_TO_BOOL DINT_TO_BOOL INT_TO_BOOL_E DINT_TO_BOOL_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Converts data type INT/DINT to BOOL.

Function/ FB classification name: Type conversion function

Input and Output Pins

(1) INT_TO_BOOL INT_TO_BOOL_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	INT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	BOOL	Output

(2) DINT_TO_BOOL DINT_TO_BOOL_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DINT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	BOOL	Output

Function

Item	Contents
Operation processing	(1) INT_TO_BOOL INT_TO_BOOL_E (a) Convert the INT type data input from input variable IN to BOOL type data, and output the data from output variable OUT. When input value is 0, it will output FALSE. When input value is not 0, it will output TRUE. 0 ⇒ FALSE 1567 ⇒ TRUE INT type → BOOL type (b) The value input from input variable IN is INT type data. (2) DINT_TO_BOOL DINT_TO_BOOL_E (a) Convert the DINT type data input from input variable IN to BOOL type data, and output the data from output variable OUT. When input value is 0, it will output FALSE. When input value is not 0, it will output TRUE. 0 ⇒ FALSE 12345678 ⇒ TRUE DINT type → BOOL type (b) The value input from input variable IN is DINT type data.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
Execute operation processing. Output the operation output value from OUT pin.

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

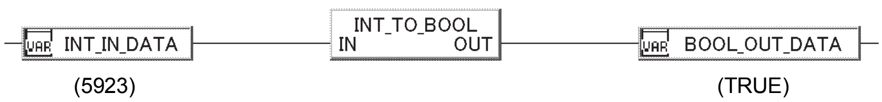
Error

There is no operation error caused by INT_TO_BOOL(_E), DINT_TO_ BOOL(_E)

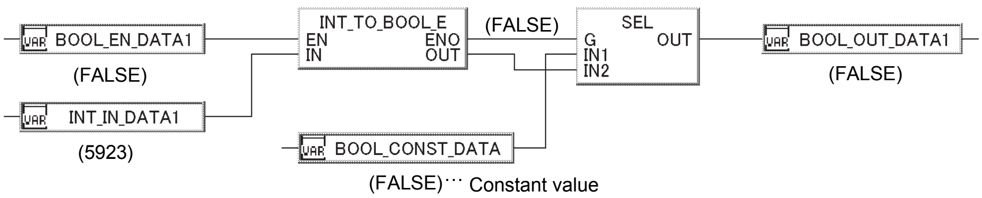
Program Example

(1) The program that converts INT type data input from input variable IN to BOOL type data, and output the data from output variable OUT.

(a) Basic program example (INT_TO_ BOOL)

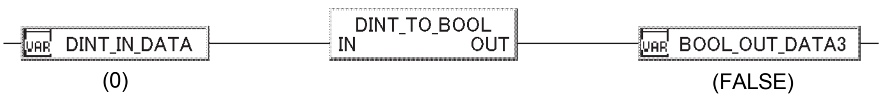


(b) The program example that outputs constant value when input variable EN is FALSE, (INT_TO_BOOL_E)

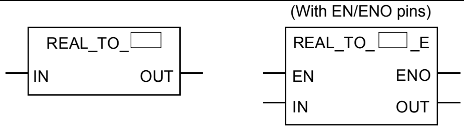


(2) The program that converts DINT type data input from input variable IN to BOOL type data, and output the data from output variable OUT.

(a) Basic program example (DINT_TO_BOOL)



4.1.8 REAL Type → INT/DINT Type Conversion (REAL_TO_INT(_E), REAL_TO_DINT(_E))

Function	FBD parts
REAL _TO_ INT REAL _TO_ DINT REAL _TO_ INT _E REAL_TO_DINT_E	 Indicates INT, DINT

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Converts data type REAL to INT/DINT.

Function/ FB classification name: Type conversion function

Input and Output Pins

(1) REAL_TO_INT REAL_TO_INT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output

(2) REAL_TO_DINT REAL_TO_DINT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DINT	Output

Function

Item: Operation processing

Contents:

(1) REAL_TO_INT REAL_TO_NT_E

(a) Convert the REAL type data input from input variable IN to INT type data, and output the data from output variable OUT.

1234.0	⇨	1234
REAL type		INT type

(b) The value input from input variable IN is REAL type data, its range is –32768 to 32767.

(c) The converted data is the value that comes from rounding up REAL type data's first digit after the decimal point.

(2) REAL_TO_DINT REAL_TO_DINT_E

(a) Convert the REAL type data input from input variable IN to DINT type data, and output the data from output variable OUT.

16543521.0	⇨	16543521
REAL type		DINT type

(b) The value input from input variable IN is REAL type data, its range is -2147483648 to 2147483647.

(c) The converted data is the value that comes from rounding up REAL type data's first digit after the decimal point.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occur</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occur) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

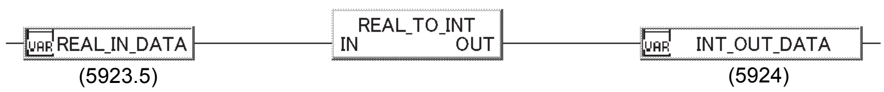
Error may occur in the following case, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- REAL_TO_INT(_E): When input value is in the range other than –32768 to 32767 (Error code: Refer to Appendix 2)
- REAL_TO_DINT(_E): When input value is in the range other than –2147483648 to 2147483647 (Error code: Refer to Appendix 2)

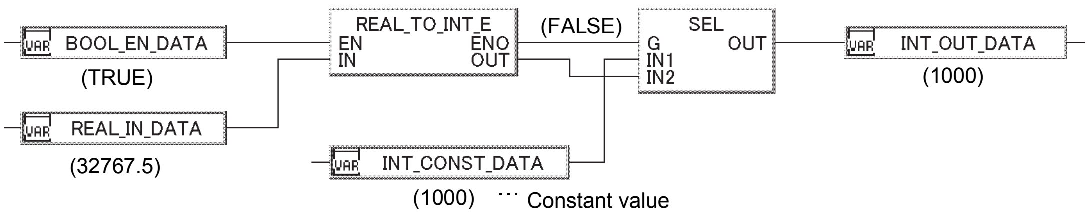
Program Example

(1) The program that converts REAL type data input from input variable IN to INT type data, and output the data from output variable OUT.

(a) Basic program example. (REAL_TO_INT)

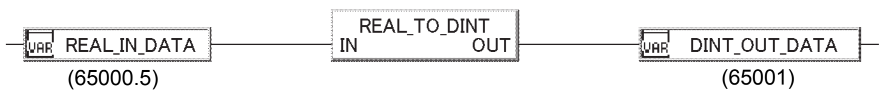


(b) The program example that outputs constant value when input variable EN is FALSE, or operation errors occur. (REAL_TO_INT_E)
(Example) When operation errors occur

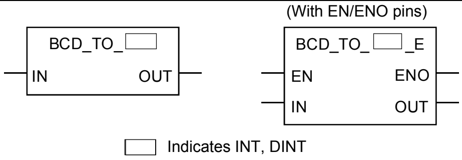


(2) The program that converts REAL type data input from input variable IN to DINT type data, and output the data from output variable OUT.

(a) Basic program example (REAL_TO_DINT)



4.1.9 BCD Type → INT/DINT Type Conversion (BCD_TO_INT(_E), BCD_TO_DINT(_E))

Function	FBD parts
BCD _TO_ INT BCD _TO_ DINT BCD _TO_ INT _E BCD_TO_DINT_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Converts data type BCD to INT/DINT.

Function/ FB classification name: Type conversion function

Input and Output Pins

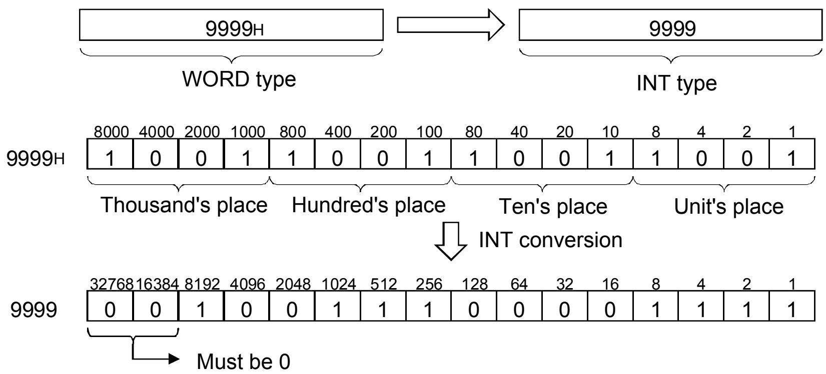
(1) BCD_TO_INT BCD_TO_INT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	WORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output

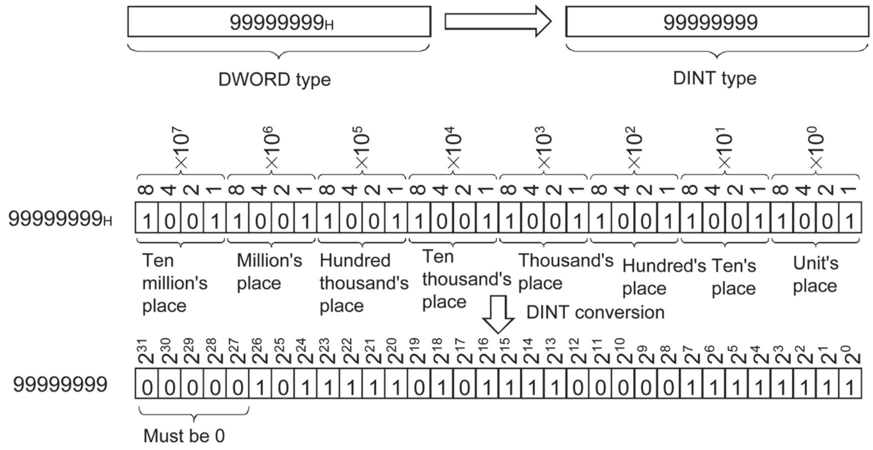
(2) BCD_TO_DINT BCD_TO_DINT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DWORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DINT	Output

Function

Item	Contents
Operation processing	(1) BCD_TO_INT BCD_TO_INT_E (a) Convert the BCD type data input from input variable IN to INT type data, and output the data from output variable OUT.  (b) The value input from input variable IN is WORD type data value, its range is 0H to 9999H (the range of each digit is 0 to 9).

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(2) BCD_TO_DINT BCD_TO_DINT_E
(a) Convert the BCD type data input from input variable IN to BCD type data, and output the data from output variable OUT.



(b) The value input from input variable IN is DWORD type data, its range is 0H to 99999999H (the range of each digit is 0 to 9).
</td></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occur</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occur) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

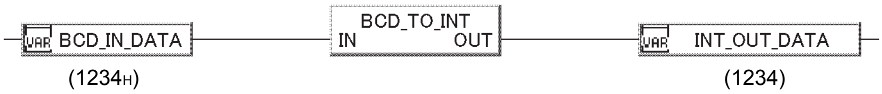
Error may occur in the following case, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- The value of each digit of the input value is in the range other than 0 to 9. (Error code: Refer to Appendix 2)

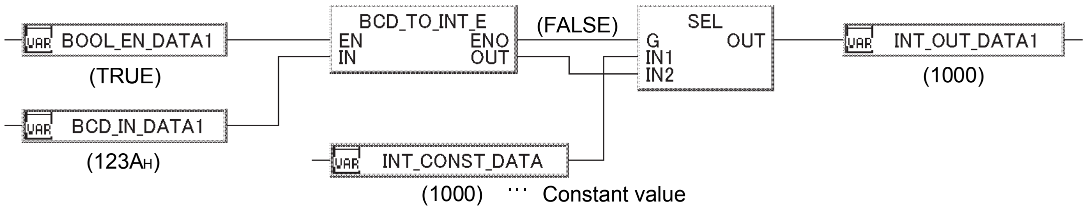
Program Example

(1) The program that converts BCD type data input from input variable IN to INT type data, and output the data from output variable OUT.

(a) Basic program example (BCD_TO_INT)

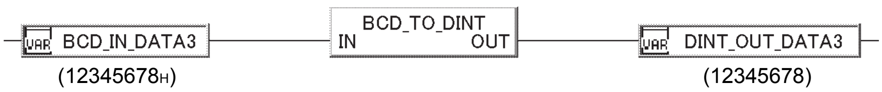


(b) The program example that outputs constant value when input variable EN is FALSE, or operation errors occur, (BCD_TO_INT_E)

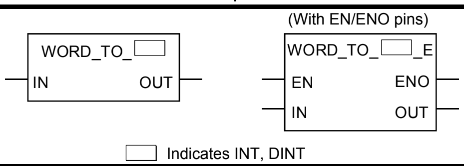


(2) The program that converts BCD type data input from input variable IN into DINT type data, and output the data from output variable OUT.

(a) Basic program example (BCD_TO_DINT)



4.1.10 WORD Type → INT/DINT Type Conversion (WORD_TO_INT(_E), WORD_TO_DINT(_E))

Function	FBD parts
WORD _TO_ INT WORD _TO_ DINT WORD _TO_ INT _E WORD_TO_DINT_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Converts data type WORD to INT/DINT.

Function/ FB classification name: Type conversion function

Input and Output Pins

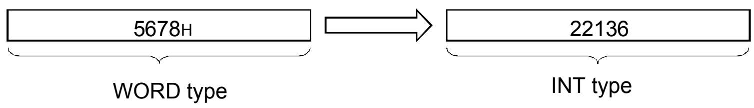
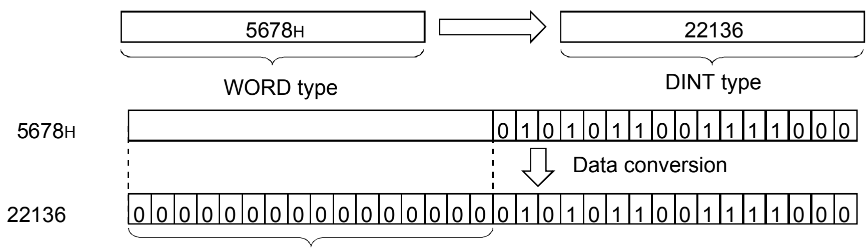
(1) WORD_TO_INT WORD_TO_INT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	WORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output

(2) WORD_TO_DINT WORD_TO_DINT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	WORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DINT	Output

Function

Item	Contents
Operation processing	(1) WORD_TO_INT WORD_TO_INT_E (a) Convert the WORD type data input from input variable IN to INT type data, and output the data from output variable OUT.  (b) The value input from input variable IN is WORD type data value. (2) WORD_TO_DINT WORD_TO_DINT_E (a) Convert the WORD type data input from input variable IN to DINT type data, and output the data from output variable OUT. After the data conversion, high-order 16 bits will become 0.  High-order 16 bits will become 0 when the data are converted. (b) The input value of input variable IN is WORD type data value.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>(1) Functions without EN/ENO pins
Execute operation processing. Output the operation output value from OUT pin.

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

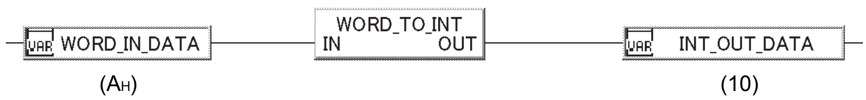
Error

There is no operation error caused by WORD_TO_INT(_E), WORD_TO_DINT(_E).

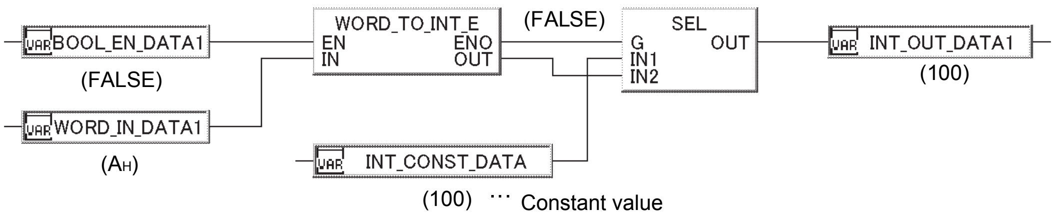
Program Example

(1) The program that converts WORD type data input from input variable IN to INT type data, and output the data from output variable OUT

(a) Basic program example (WORD_TO_INT)

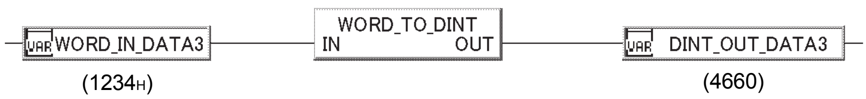


(b) The program example that outputs constant value when input variable EN is FALSE, or operation errors occur (WORD_TO_INT_E)

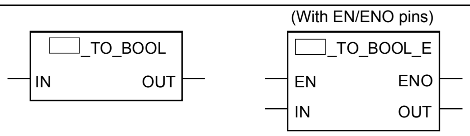


(2) The program that converts WORD type data input from input variable IN to DINT type data, and output the data from output variable OUT

(a) Basic program example (WORD_TO_DINT)



4.1.11 WORD/DWORD Type → BOOL Type Conversion (WORD_TO_BOOL(_E), DWORD_TO_BOOL(_E))

Function	FBD parts
WORD _TO_BOOL DWORD_TO_BOOL WORD_TO_BOOL_E DWORD_TO_BOOL_E	 Indicates INT, DINT

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Converts data type WORD/DWORD to BOOL.

Function/ FB classification name: Type conversion function

Input and Output Pins

(1) WORD_TO_BOOL WORD_TO_BOOL_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	WORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	BOOL	Output

(2) DWORD_TO_BOOL DWORD_TO_BOOL_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DWORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	BOOL	Output

Function

Item	Contents
Operation processing	(1) WORD_TO_BOOL WORD_TO_BOOL_E (a) Convert the WORD type data input from input variable IN to BOOL type data, and output the data from output variable OUT. When input value is 0H, it will output FALSE. When input value is not 0H, it will output TRUE. 0H ⇒ FALSE 1567H ⇒ TRUE WORD type BOOL type (b) The value input from input variable IN is WORD type data value.

Item	Contents
Operation processing	(2) DWORD_TO_BOOL DWORD_TO_BOOL_E (a) Convert the DWORD type data input from input variable IN to BOOL type data, and output the data from output variable OUT. When input value is 0H, it will output FALSE. When input value is not 0H, it will output TRUE. 0H ⇒ FALSE 12345678H ⇒ TRUE DWORD type → BOOL type (b) The value input from input variable IN is DWORD type data value.
Operation results	(1) Functions without EN/ENO pins Execute operation processing. Output the operation output value from OUT pin. (2) Functions with EN/ENO pins The execution conditions and the operation results are as follows: (see table below) * When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE (*)	Undefined value

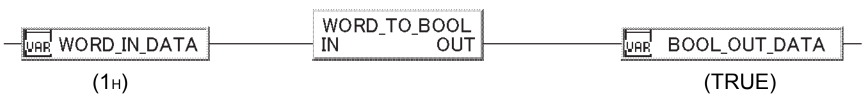
Error

There is no operation error caused by WORD_TO_BOOL (_E), DWORD_TO_BOOL (_E).

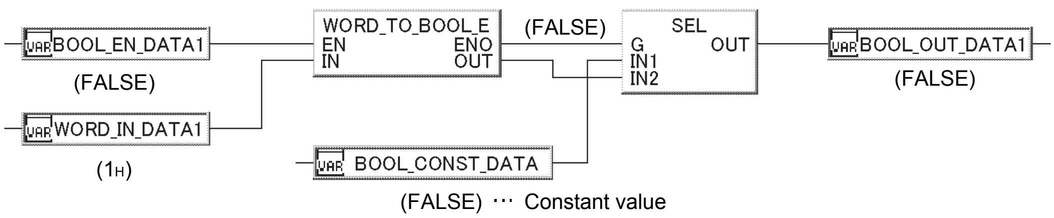
Program Example

(1) The program that converts WORD type data input from input variable IN to BOOL type data, and output the data from output variable OUT.

(a) Basic program example (WORD_TO_BOOL)

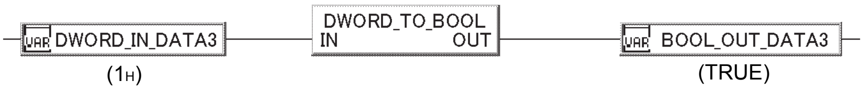


(b) The program example that outputs constant value when input variable EN is FALSE (WORD_TO_BOOL_E)

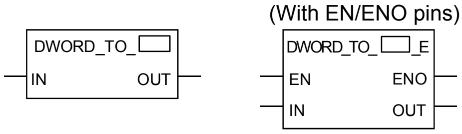


(2) The program that converts DWORD type data input from input variable IN to BOOL type data, and output the data from output variable OUT

(a) Basic program example (DWORD_TO_BOOL)



4.1.12 DWORD Type → INT/DINT Type Conversion (DWORD_TO_INT(_E), DWORD_TO_DINT(_E))

Function	FBD parts
DWORD _TO_INT DWORD_TO_DINT DWORD_TO_INT_E DWORD_TO_DINT_E	 Indicates INT, DINT

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Functions overview: Converts data type DWORD to INT/DINT.

Function/FB classification name: Type conversion function

Input and Output Pins

(1) DWORD_TO_INT DWORD_TO_INT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DWORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output

(2) DWORD_TO_DINT DWORD_TO_DINT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DWORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DINT	Output

Function

Item	Contents
Operation Processing	(1) DWORD_TO_INT DWORD_TO_INT_E (a) Convert the DWORD type data input from the input variable IN to INT type and output from output variable OUT. High-order 16 bit information of the input is abandoned. BC614EH ⇒ 24910 DWORD type INT type BC614EH 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 24910 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 The high-order 16 bit information of the input is abandoned. (b) The input value of input variable IN is DWORD type data. (2) DWORD_TO_DINT DWORD_TO_DINT_E (a) Convert the Data of DWORD type input from input variable IN to DINT type data and output from output variable OUT. BC614EH ⇒ 12345678 DWORD type DINT type (b) The input value of input variable IN is DWORD type data.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
Execute operation processing. Output the operation output value from OUT pin.

(2) Functions With EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

POINT

High-order 16 bit information of the DWORD type data input from input variable IN is abandoned in executing DWORD_TO_INT_(_E)

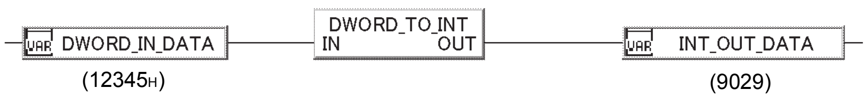
Error

There is no operation error caused by DWORD_TO_INT(_E) and DWORD_TO_DINT(_E).

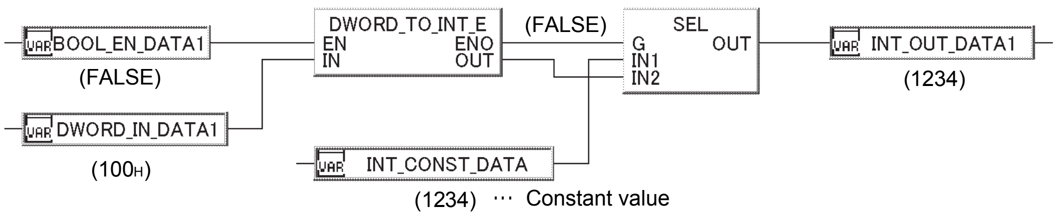
Program Example

(1) The program that converts DWORD type data input from input variable IN to INT type data and output from output variable OUT.

(a) Basic program example (DWORD_TO_INT)

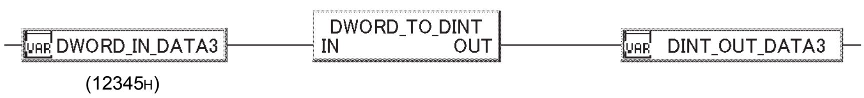


(b) The program example that outputs constant value when input variable EN is FALSE. (DWORD_TO_INT_E)

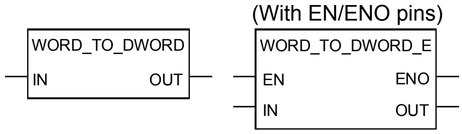


(2) The program that converts DWORD type data input from input variable IN to DINT type data, and output from output variable OUT.

(a) Basic program example (DWORD_TO_DINT)



4.1.13 WORD Type → DWORD Type Conversion (WORD_TO_DWORD(_E))

Function	FBD parts
WORD_TO_DWORD WORD_TO_DWORD_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Functions overview: Converts data type WORD to DWORD.

Function/FB classification name: Type conversion function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	WORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DWORD	Output

Function

Item	Contents
Operation processing	(1) Convert the WORD type data from input variable IN to DWORD type data and output it from output variable OUT. After the conversion the 16 high-order bits become 0. $\underbrace{\boxed{5678_H}}_{\text{WORD type}} \Rightarrow \underbrace{\boxed{00005678_H}}_{\text{DWORD type}}$ (2) The value input from input variable IN is WORD type data.
Operation results	(1) Functions without EN/ENO pins Execute operation processing. Output the operation output value from OUT pin. (2) Functions With EN/ENO pins The execution conditions and the operation results are as follows: (see table below) * When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE (*)	Undefined value

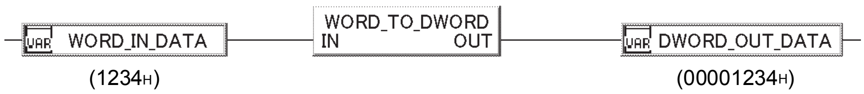
Error

There is no operation error caused by WORD_TO_DWORD(_E).

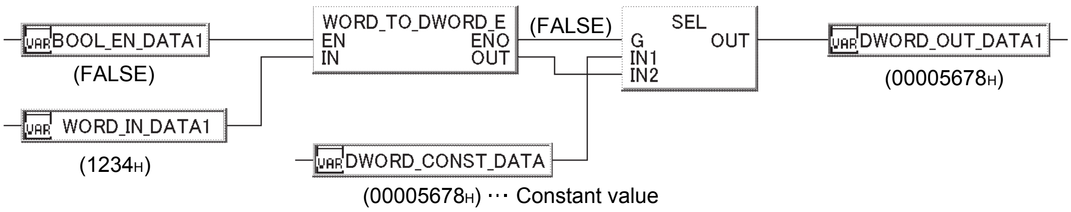
Program Example

The program that converts WORD type data input from variable IN to DWORD type data, and output from output variable OUT.

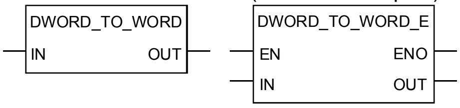
(1) Basic program example (WORD_TO_DWORD)



(2) The program example that outputs constant value when input variable EN is FALSE.
(WORD_TO_DWORD_E)



4.1.14 DWORD Type → WORD Type Conversion (DWORD_TO_WORD(_E))

Function	FBD parts
DWORD_TO_WORD DWORD_TO_WORD_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

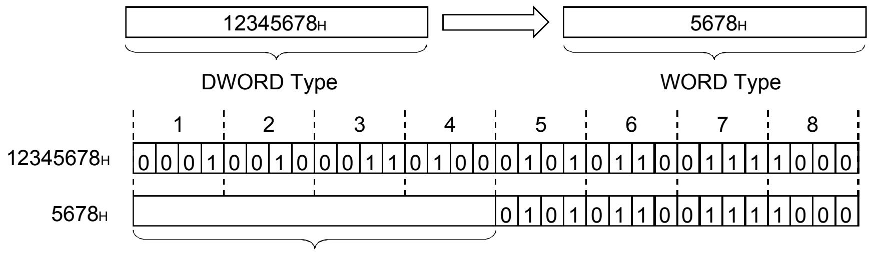
Functions overview: Converts data type DWORD to WORD.

Function/FB classification name: Type conversion function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DWORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	WORD	Output

Function

Item	Contents
Operation processing	(1) Convert the DWORD type data input from input variable IN to WORD type data and output from the output variable OUT. High-order 16 bit information of the input is abandoned  The high-order 16 bit information of the input are abandoned (2) The data input from input variable IN is DWORD type data.
Operation results	(1) Functions without EN/ENO pins Execute operation processing. Output the operation output value from OUT pin. (2) Functions With EN/ENO pins The execution conditions and the operation result are as follows: Execution condition EN / Operation result ENO, OUT: TRUE (Operation execution): TRUE, Operation output value FALSE (Operation stop): FALSE (*), Undefined value * When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE (*)	Undefined value

POINT

High-order 16 bit information of the DWORD type data input from input variable IN is abandoned in executing DWORD_TO_ WORD (_E)

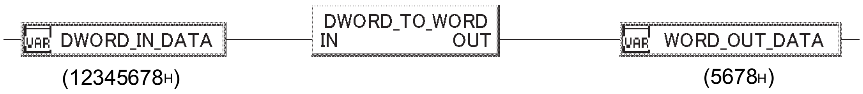
Error

There is no operation error caused by DWORD_TO_WORD(_E)

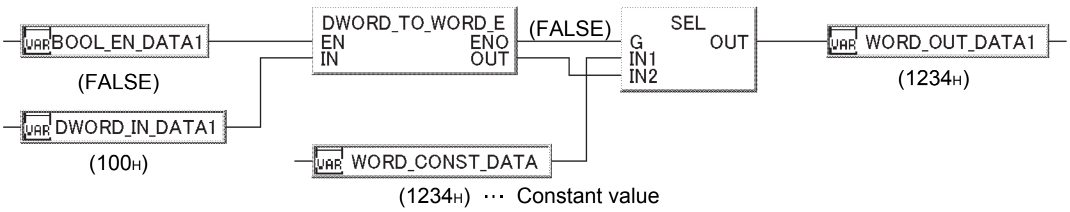
Program Example

The program that converts DWORD type data input from input variable IN to WORD type data, and output from output variable OUT

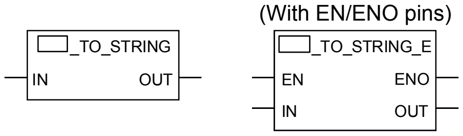
(1) Basic program example (DWORD_TO_WORD)



(2) The program example that outputs constant value when input variable EN is FALSE. (DWORD_TO_WORD_E)



4.1.15 INT/DINT Type → STRING Type Conversion (INT_TO_STRING(_E), DINT_TO_STRING(_E))

<table>
<tr><th>Function</th><th>FBD parts</th></tr>
<tr><td>INT_TO_STRING
DINT_TO_STRING
INT_TO_STRING_E
DINT_TO_STRING_E</td><td>(With EN/ENO pins)

Indicater INT,DINT</td></tr>
</table>

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Functions overview: Converts data type INT/DINT to STRING.

Function/FB classification name: Type conversion function

Input and Output Pins

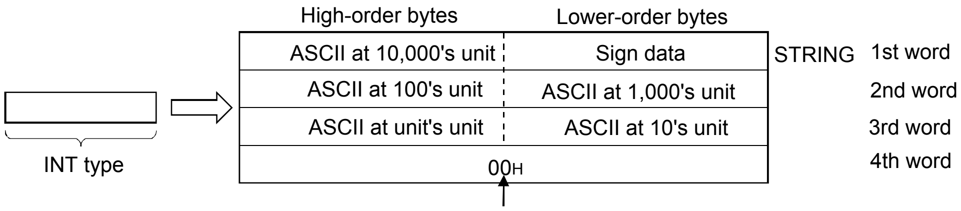
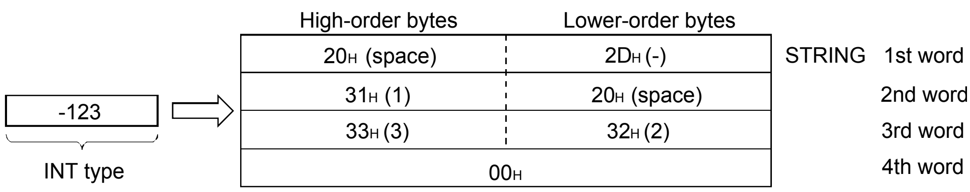
(1) INT_TO_STRING INT_TO_STRING_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	INT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	STRING (6)	Output

(2) DINT_TO_STRING DINT_TO_STRING_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DINT	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	STRING (11)	Output

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>(1) INT_TO_STRING INT_TO_STRING_E
(a) Convert the INT type data input from input variable IN to STRING type data, and output from output variable OUT.

Automatically stored at the end of the string
(b) The value input from input variable IN is INT type data.
(c) "20H (space)" is stored into "Sign data" if the input value is positive; "2DH (-)" is stored if negative.
(d) When there are not so many valid No. of bytes, "20H (space)" is stored in high-order bytes.
(Example) Suppose that the input is -123.

(e) "00H" is automatically stored at the end of the string. (The 4th word)</td></tr>
</table>

Item	Contents
Operation processing	(2) DINT_TO_STRING DINT_TO_STRING_E

(a) Convert the DINT type data input from input variable IN to STRING type data and output from output variable OUT.

DINT type ⇒

High-order digits	Lower-order digits	STRING	
ASCII at 1,000,000,000's	Sign data		1st character
ASCII at 10,000,000	ASCII at 100,000,000's		2nd character
ASCII at 100,000's	ASCII at 1,000,000's		3rd character
ASCII at 1,000's	ASCII at 10,000's		4th character
ASCII at 10's	ASCII at 100's		5th character
00H	ASCII at unit's		6th character

Automatically stored at the end of the string

(b) The value input from input variable IN is DINT type data.

(c) "20H (space)" is stored to "Sign data" if the input value is positive; "2DH (-)" is stored if negative.

(d) When there are not so many valid No. of bytes, "20H (space)" is stored to high-order bytes.
(Example) Suppose the input is -123456

-123456 (DINT type) ⇒

High-order bytes	Lower-order bytes	STRING	
20H(space)	2DH(–)		1st character
20H(space)	20H(space)		2nd character
31H (1)	20H(space)		3rd character
33H (3)	32H (2)		4th character
35H (5)	34H (4)		5th character
00H	36H (6)		6th character

(e) "00H" is automatically stored at the end of the string. (The 6th word High-order bytes)

Operation results

(1) Functions without EN/ENO pins
Execute operation processing. Output the operation output value from OUT pin.

(2) Functions With EN/ENO pins
The execution conditions and the operation results are as follows:

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

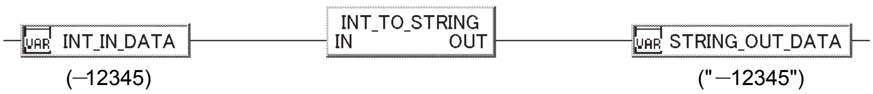
Error

There is no operation error caused by INT_TO_STRING(_E), DINT_TO_STRING(_E).

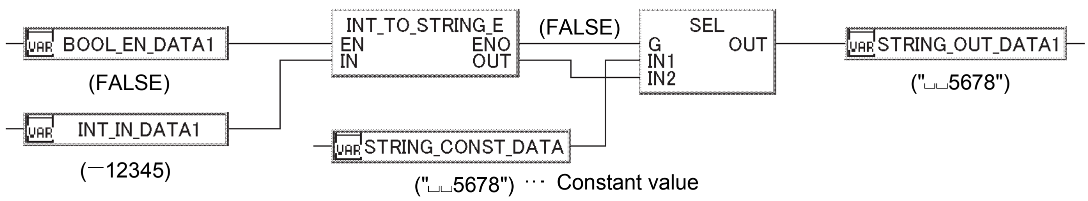
Program Example

(1) The program that converts the INT type data input from input variable IN to STRING type data, and output from the output variable OUT

(a) Basic program example. (INT_TO_STRING)

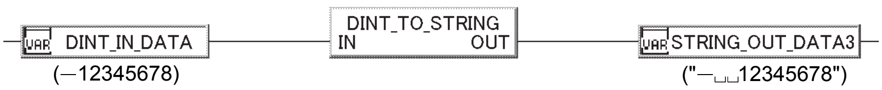


(b) The program example that output constant value when input variable EN is FALSE (INT_TO_STRING_E)

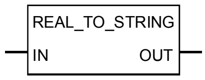
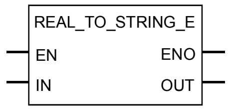


(2) The program that converts the DINT type data input from input variable IN to STRING type data and output from the output variable OUT

(a) Basic program example (DINT_TO_STRING)



4.1.16 REAL Type → STRING Type (Exponent Form) Conversion (REAL_TO_STRING(_E))

Function	FBD parts
REAL_TO_STRING REAL_TO_STRING_E	 (With EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

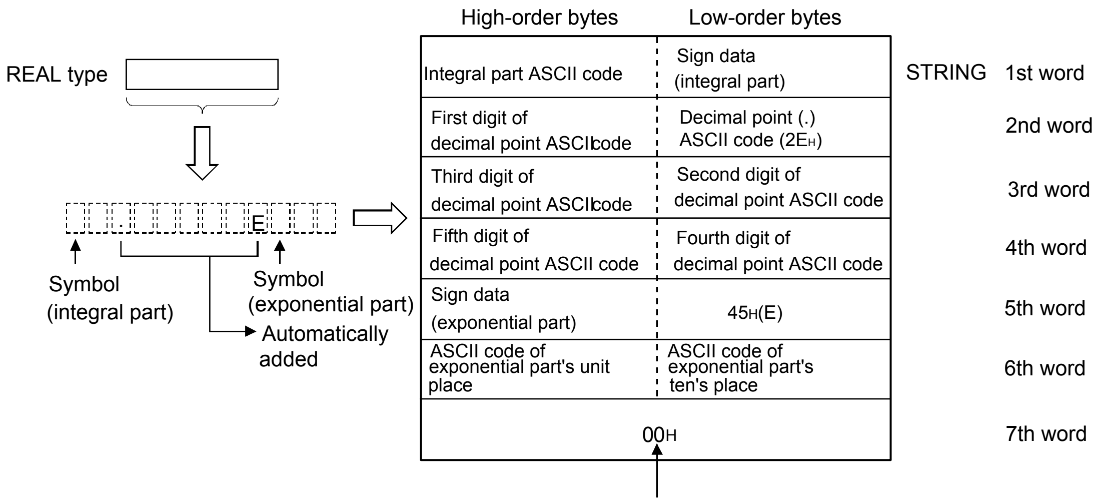
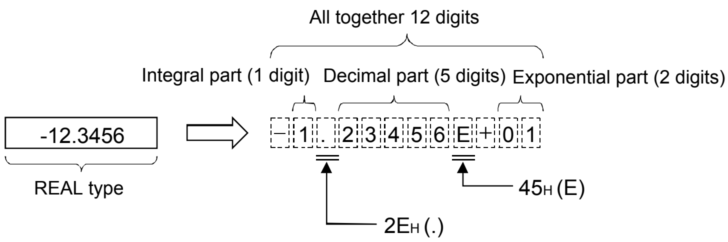
Functions overview: Converts data type REAL to STRING (Exponent form).

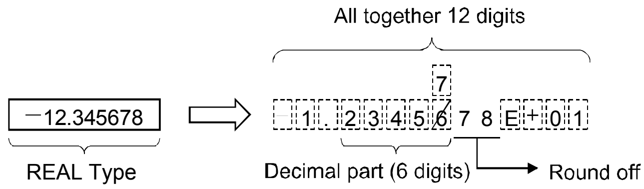
Function/FB classification name: Type conversion function

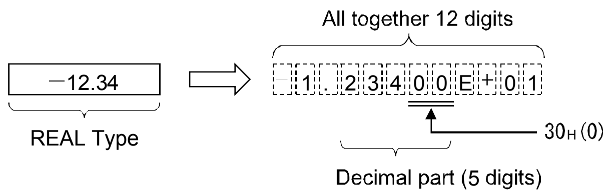
Input and Output Pins

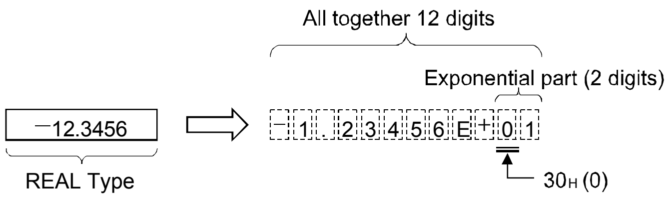
Lead pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute, FALSE: Stop)
	IN	Input variable	REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal, FALSE: Abnormal)
	OUT	Output variable	STRING (12)	Output

Function

Item	Contents
Operation processing	(1) Convert the REAL type data input from input variable IN to STRING (exponential form) type data and output from output variable OUT.  (2) The value input from input variable IN is REAL type data. (3) Converted data of string type are output from output variable OUT in the following manner. (a) The digit's numbers of integral part, decimal part and exponential part are fixed. (Integral part: 1 digit, Decimal part: 5 digit, Exponential part: 2 digit). "2EH"(.) and "45H"(E) are stored automatically in the 3rd and 9th byte. 

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(b) "20H"(space) is stored to "Sign data (integral part)" if the input value is positive; "2DH"(-) is stored if negative.
(c) All the digits after the 6th digit below the decimal part are rounded.


(d) When there are not so many valid No. of digits, "30H"(0) is stored in decimal part.


(e) "2BH"(+) is stored in Sign data (exponential part) if the exponential is positive; "2DH"(-) is stored if negative.
(f) "30H"(0) is stored in ten's place of the exponential part if the exponential only has one digit.


(4) "00H" is automatically stored at the end of the string. (The 7th word)
</td></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation results</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occur</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occur) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

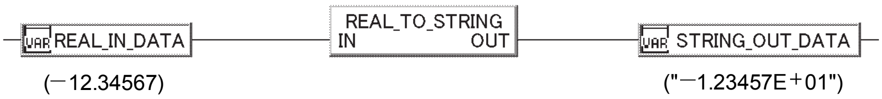
Error may occur in the following cases, and the error code will be displayed on the FBD program diagnosis screen of PX Developer programming tool.

- Input values other than "0" do not fall in the range of $\pm1.17549^{-38}$ to $\pm3.40282^{+38}$. (Error code: Refer to Appendix 2)

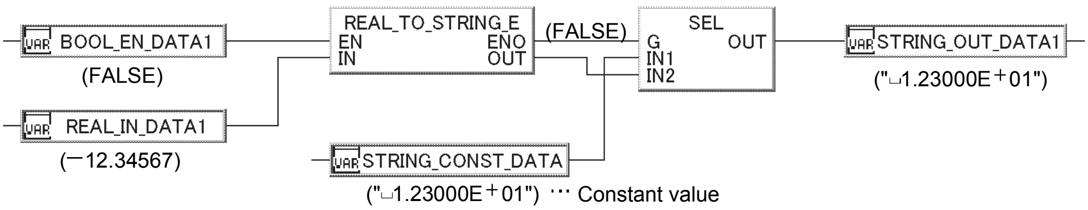
Program Example

The program that converts the REAL type data input from input variable IN to STRING type data, and output from the output variable OUT. (Exponent form)

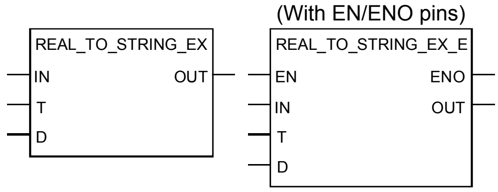
(1) Basic program example (REAL_TO_STRING).



(2) The program example that outputs constant value when input variable EN is FALSE, or operation error occurs. (REAL_TO_STRING_E)
(Example) When the input variable EN is FALSE



4.1.17 REAL Type → STRING Type (Decimal Point Form) Conversion (REAL_TO_STRING_EX(_E))

Function	FBD parts
REAL_TO_STRING_EX REAL_TO_STRING_EX_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Functions overview: Converts data type REAL to STRING (Decimal Point Form).

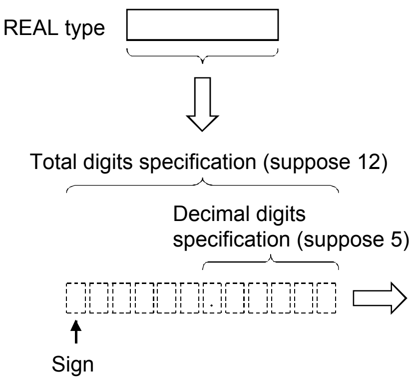
Function/FB classification name: Type conversion function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Executing condition (TRUE: Execute, FALSE: Stop)
	IN	Input variable	REAL	Input
	T	Input variable	INT	Total digits specification
	D	Input variable	INT	Decimal digits specification
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal, FALSE: Abnormal)
	OUT	Output variable	STRING (24)	Output

Function

Item	Contents
Operation Processing	(1) Convert the REAL type data input from input variable IN to STRING (decimal point) type data and output from output variable OUT.



	High-order digits	Low-order digits
STRING 1st word	(Total digits specification value-1) digit of ASCII code	Sign data
2nd word	(Total digits specification value-3) digit of ASCII code	(Total digits specification value-2) digit of ASCII code
3rd word	(Total digits specification value-5) digit of ASCII code	(Total digits specification value-4) digit of ASCII code
4th word	(Total digits specification value-6) digit of ASCII code (Decimal point part)	Decimal point (.) ASCII code (2EH)
5th word	(Total digits specification value-8) digit of ASCII code (Decimal point part)	(Total digits specification value-7) digit of ASCII code (Decimal point part)
6th word	(Total digits specification value-10) digit of ASCII code (Decimal point part)	(Total digits specification value-9) digit of ASCII code (Decimal point part)
7th word	00H	

Automatically storde at the end of the string

(2) The value input from input variable IN is REAL type data.

(3) Data input from input variable T (total digits specification) is specified as below.
Sign and decimal point shall be counted in the total digits number.
When decimal digit is 0, number of total digits (maximum: 24) ≧2.
When decimal digit is not 0, number of total digits (maximum: 24) ≧ (Number of decimal digits +3)

Item	Contents

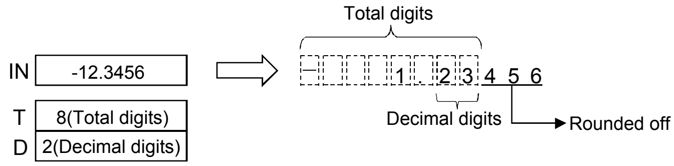
Operation processing

(4) Data input from input variable D (decimal digits specification) shall be ranged between 0 and 7. (Make sure that number of decimal digits ≦ (Number of total digits - 3))

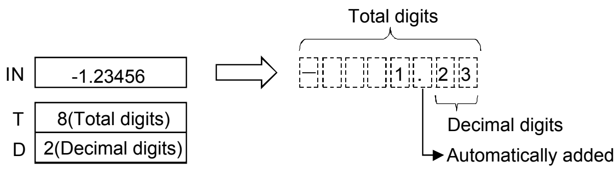
(5) Converted data of string type are output from the output variable OUT in the following manner.

(a) "20_H"(space) is stored to sign data (integral part) if the input value is positive; "$2D_H$"(-) is stored if negative.

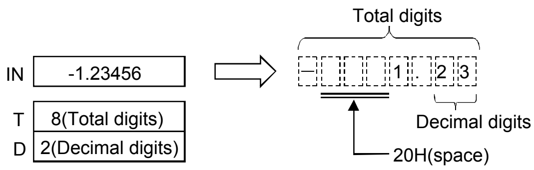
(b) If the decimal digits of a REAL type data exceed the valid range specified in decimal digits specification, the Low-order digits beyond the range are rounded off.



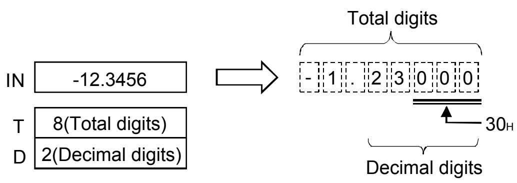
(c) When decimal digit is set other than "0", "$2E_H$"(.) is automatically stored to the first digit after the specified decimal part.
When decimal digit is "0", "$2E_H$"(.) is not stored.



(d) "20_H"(space) is stored to each excess integral digit between sign digit and integral part when the total digits minus one digit for sign, one for decimal point and decimal digits for decimal part exceed the required digits for the REAL type integral part.



(e) When there are not so many valid No. of digits, "30_H"(0) is stored to in decimal part.



(6) "00_H" is automatically stored at the end of the string.

Operation results

(1) Functions without EN/ENO pins.
The operation results are as follows:

Operation results	OUT
No operation error	Operation output value
Operation error occur	Undefined value

(2) Functions With EN/ENO pins.
The execution conditions and the operation results are as follows:

Executing condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error occurs) (*)	Undefined value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Error

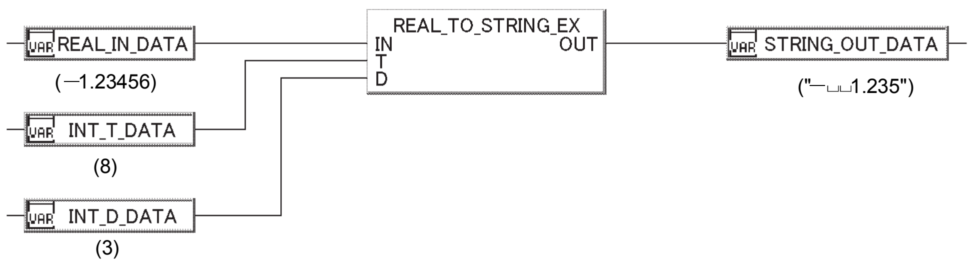
Operation error is incurred in the following condition. Error code will be displayed on FBD program diagnosis screen of PX developer programming tool.

- Input values other than "0" do not fall in the range of $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$. (Error code: Refer to Appendix 2)
- Data input to the input variable IN exceeds length of string the total digits specified after type conversion.
- Data input to the input variable T (total digits specification) is beyond the valid range specified as below: (Error code: Refer to Appendix 2)
  When decimal digit is "0", number of total digits (maximum: 24) ≧ 2.
  When decimal digit is not "0", number of total digits (maximum: 24) ≧ (Number of decimal digits +3)
- Data input to the input variable D (decimal digits specification) is beyond the valid range specified as below: (Error code: Refer to Appendix 2)
  Number of decimal digits does not exceed total digits minus 3

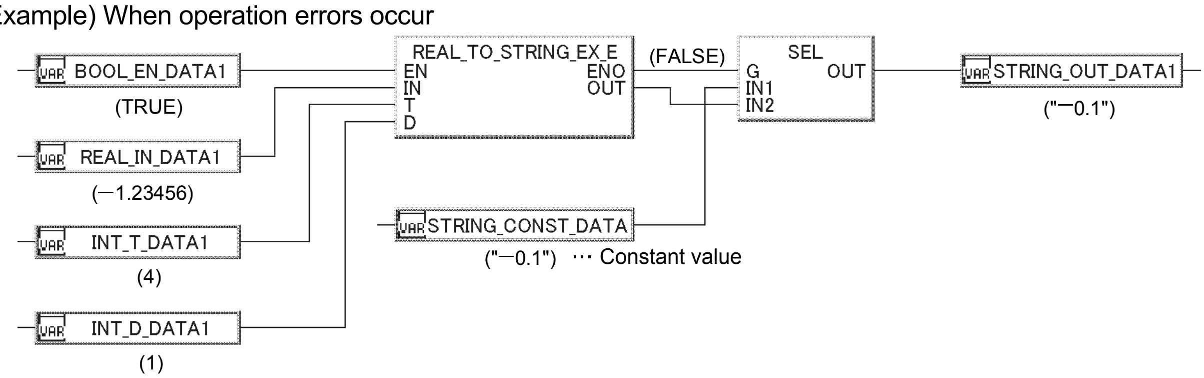
Program Example

The program converts the REAL type data input from input variable IN to STRING (decimal point) type data and output from the output variable OUT.

(1) Basic program example (REAL_TO_STRING_EX)



(2) The program example that outputs constant value when the input variable EN is FALSE, or operation error occurs. (REAL_TO_STRING_EX_E)
(Example) When operation errors occur



4.1.18 STRING Type → INT/DINT Type Conversion (STRING_TO_INT(_E), STRING_TO_DINT(_E))

Function	FBD parts
STRING_TO_INT STRING_TO_DINT STRING_TO_INT_E STRING_TO_DINT_E	STRING_TO_☐ (IN, OUT) (With EN/ENO pins) STRING_TO_☐_E (EN, ENO, IN, OUT) ☐ Indicates INT, DINT

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Functions overview: Converts data type STRING to INT/DINT

Function/FB classification name: Type conversion function

Input and Output Pins

(1) DWORD STRING_TO_INT STRING_TO_INT_E

Lead pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (6)	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output

(2) STRING_TO_DINT STRING_TO_DINT_E

Lead pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (11)	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DINT)	Output

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation Processing</td><td>
(1) STRING_TO_INT STRING_TO_INT_E
(a) Data of STRING type input to input variable (IN) are converted into data of INT type and output from the output variable (OUT)
<table>
<tr><th></th><th></th><th>High-order byte</th><th>Low-order byte</th></tr>
<tr><td>STRING</td><td>1st word</td><td>ASCII at 10,000's place</td><td>Sign data</td></tr>
<tr><td></td><td>2nd word</td><td>ASCII at 100's place</td><td>ASCII at 1,000's place</td></tr>
<tr><td></td><td>3rd word</td><td>ASCII at unit's place</td><td>ASCII at 10's place</td></tr>
<tr><td></td><td>4th word</td><td colspan="2">00H(indicates the end of the string)</td></tr>
</table>
⇒ INT type
(b) Data input from input variable IN are of STRING type. ASCII of these data may be in the following range: "30H" to "39H","20H", "2DH", "00H". Figure of STRING type should be ranged between -32768 to 32767.
</td></tr>
</table>

Item	Contents
Operation processing	(2) STRING_TO_DINT STRING_TO_DINT_E

(a) Convert the STRING type data input from input variable IN to DINT type data and output from the output variable OUT.

STRING	High-order digits	Lower-order digits
1st word	ASCII at 1,000,000,000's place	Sign data
2nd word	ASCII at 10,000,000's place	ASCII at 100,000,000's place
3rd word	ASCII at 100,000's place	ASCII at 1,000,000's place
4th word	ASCII at 1,000's place	ASCII at 10,000's place
5th word	ASCII at 10's place	ASCII at 100's place
6th word	00H	ASCII at unit's place

Indlcates the end of the string

⇨ DINT type

(b) Data input from input variable (IN) are of STRING type. ASCII of these data may be in the following range: "30H" to "39H","20H", "2DH", "00H". Figure of STRING type should be ranged between -2147483648 to 2147483647.

Operation results

(1) Functions without EN/ENO pins

The operation results are as follows:

Operation results	OUT
No operation error	Operation output value
Operation error occur	Undefined value

(2) Functions With EN/ENO pins

The execution conditions and the operation results are as follows:

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error occurs) (*)	Undefined value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Error

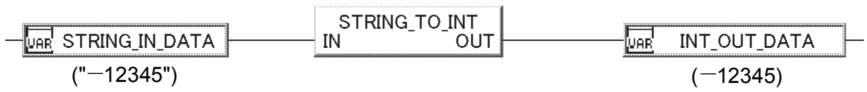
Operation error is incurred in the following condition. Error code will be displayed on FBD program diagnostic screen in PX developer programming tool.

- ACSII of the input is not "30H" to "39H", "20H", "2DH", "00H" (Error code: Refer to Appendix 2)
- ACSII of the input is beyond the range specified as below: (Error code: Refer to Appendix 2)
  STRING_TO_INT(_E): "-32768 to 32767"
  STRING_TO_DINT(_E): "-2147483648 to 2147483647"

Program Example

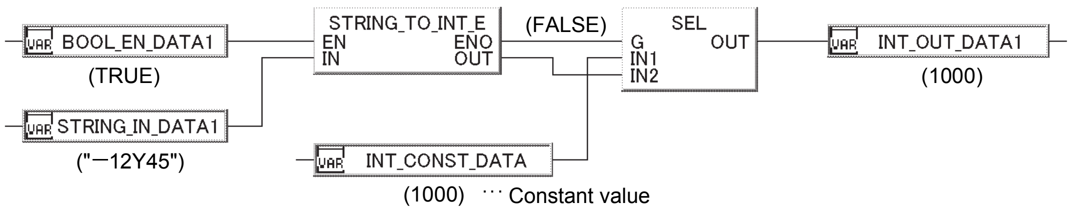
(1) The program that converts STRING type data input from input variable IN to INT type data, and output from output variable OUT.

(a) Basic program example (STRING_TO_INT)



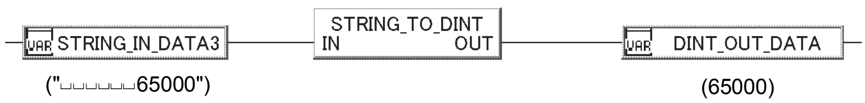
(b) The program example that outputs constant value when input variable EN is FALSE, or operation error occurs. (STRING_TO_INT_E)

(Example) When operation errors occur



(2) The program that converts STRING type data input from input variable IN to DINT type data, and output from the output variable OUT

(a) Basic program example (STRING_TO_DINT)



4.1.19 STRING Type → REAL Type Conversion (STRING_TO_REAL(_E))

Function	FBD parts
STRING_TO_REAL STRING_TO_REAL_E	STRING_TO_REAL: IN → OUT (With EN/ENO pins) STRING_TO_REAL_E: EN → ENO, IN → OUT

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Functions overview: Converts data type STRING to REAL.

Function/FB classification name: Type conversion function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (12)	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	REAL	Output

Function

Item: Operation processing

(1) Convert the STRING type data input from input variable IN to REAL type data, and output from output variable OUT.

STRING	High-order byte	Low-order byte
1st word	1st character ASCII code	Sign data
2nd word	3rd character ASCII code	2nd character ASCII code
3rd word	5th character ASCII code	4th character ASCII code
4th word	7th character ASCII code	6th character ASCII code
5th word	9th character ASCII code	8th character ASCII code
6th word	11th character ASCII code	10th character ASCII code
7th word	00H (Implying the end of the string)	

⇨ REAL type

(2) Data of string type may be converted into real type of both decimal form and exponent form.

(a) In decimal point form

STRING	High-order byte	Low-order byte
1st word	31H (1)	2DH (−)
2nd word	33H (3)	2EH (.)
3rd word	30H (0)	35H (5)
4th word	34H (4)	33H (3)
5th word	00H	

− 1 . 3 5 0 3 4

⇨ −1.35034

REAL type

(b) In exponent form

STRING	High-order byte	Low-order byte
1st word	31H (1)	2DH (−)
2nd word	33H (3)	2EH (.)
3rd word	30H (0)	35H (5)
4th word	34H (4)	33H (3)
5th word	2DH (−)	45H (E)
6th word	30H (0)	31H (1)
7th word	00H	

− 1 . 3 5 0 3 4 E − 1 0

⇨ −1.35034E −10

REAL type

Item	Contents

Operation processing

(3) Convert STRING type data that has 6 valid digits (excluding symbol digit, decimal point and exponential part). Digits after the sixth one are all rounded off in conversion processing.

(a) In decimal point form -1.35034 1 2 0 2 3 ⇒ -1.35034
Cut
REAL Type

(b) In exponent form -1.35034 1 2 E -1 0 ⇒ -1.35034E-10
Cut
Real type

(4) When abbreviating a sign or specifying "2BH"(+) to a sign in decimal point form, the STRING type data is converted to a positive value.
Whereas, when specifying "2DH"(-) as a sign, the STRING type data is converted to a negative value.

(5) When abbreviating a sign or specifying "2BH"(+) to a sign of exponential part in exponent form, the STRING type data is converted to a positive value.
Whereas, when specifying "2DH"(-) in an exponential part, the STRING type data is converted to a negative value.

(6) If there is "20H"(space) or "30H"(0) at the beginning of the STRING type data except "0", it will be neglected in the conversion processing.

(a) In decimal point form - 0 1 . 3 5 0 3 4 ⇒ -1.35034
Neglected
REAL type

(b) In exponent form - 0 1 . 3 5 0 3 4 E -1 0 ⇒ -1.35034E-10
Neglected
REAL type

(7) "30H"(0) between "E" and exponent in a string type data (exponent form) is neglected in the conversion processing.
-1 . 3 5 0 3 4 E - 0 2 ⇒ -1.35034E-2
Neglected
REAL type

(8) When "20H"(space) is included in the character string, the conversion is executed with ignoring "20H".

(9) The maximum number of characters that can be input for a string type data is 24.
"20H" (space) and "30H"(0) in the character string are counted as one character.

(10) ASCII of a STRING type data input to the input variable IN shall be ranged as follows:
"30H" to "39H ", "45H ", "2BH ", "2DH ", "2EH ", "20H ", "00H ".

Operation results

(1) Functions without EN/ENO pins
The operation results are as follows:

Operation results	OUT
No operation error	Operation output value
Operation error occur	Undefined value

(2) Functions With EN/ENO pins
The execution conditions and the operation results are as follows:

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE (without operation error)	Operation output value
	FALSE (Operation error occurs) (*)	Undefined value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Error

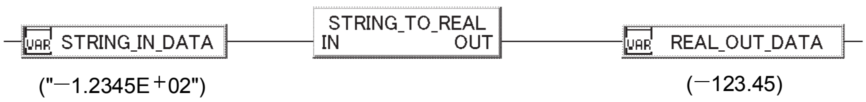
The following situation indicate operation error occurrence. In this case, error code will be displayed on the FBD program diagnosis screen of PX Developer programming tool.

- There is character of the integral/decimal part is beyond the range of "30H"(0) to "39H"(9). (Error code: Refer to Appendix 2)
- There exists two or more "2EH"(.). (Error code: Refer to Appendix 2)
- There is character whose exponential part is not "45H2BH"(E+), or "45H2DH"(E-), or there are several exponential parts. (Error code: Refer to Appendix 2)
- Converted data is except 0 or does not fall in the range of $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$. (Error code: Refer to Appendix 2)
- There are no character or more than 24 characters. (Error code: Refer to Appendix 2)

Program Example

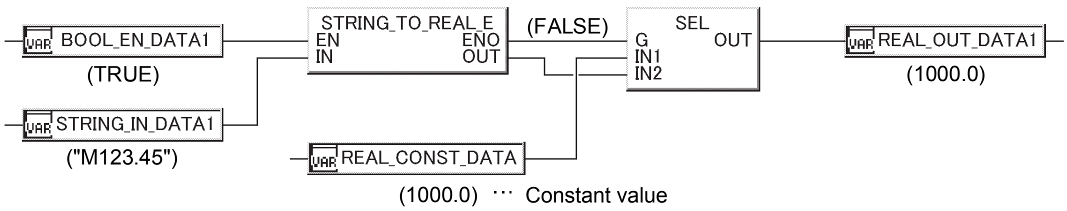
The program that converts STRING type data input from input variable IN to REAL type data, and output from output variable OUT.

(1) Basic program example (STRING_TO_REAL)

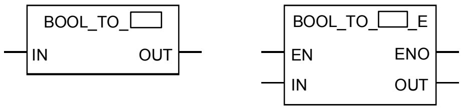


(2) The program example that outputs constant value when input variable EN is FALSE, or operation error occurs. (STRING_TO_REAL_E)

(Example) When operation errors occur



4.1.20 BOOL Type → INT/DINT Type Conversion (BOOL_TO_INT(_E), BOOL_TO_DINT(_E))

Function	FBD parts
BOOL_TO_INT BOOL_TO_DINT BOOL_TO_INT_E BOOL_TO_DINT_E	(With EN/ENO pins)  □ Indicates INT, DINT

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Functions overview: Converts data type BOOL to INT/DINT.

Function/FB classification name: Type conversion function

Input and Output Pins

(1) BOOL_TO_INT BOOL _TO_INT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	BOOL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output

(2) BOOL _TO_DINT BOOL _TO_DINT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	BOOL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DINT	Output

Function

Item	Contents
Operation processing	(1) BOOL_TO_INT BOOL _TO_INT_E (a) Converts the BOOL type data input from input variable IN to INT type data, and output from output variable OUT. When input data is FALSE, '0' (INT type) is output. When input data is TRUE, '1' (INT type) is output. FALSE ⇨ 0 TRUE ⇨ 1 BOOL type → INT type (b) The value input of input variable IN is BOOL type data. (2) BOOL _TO_DINT BOOL _TO_DINT_E (a) Convert the BOOL type data input from input variable IN to DINT type data, and output from output variable OUT. When input data is FALSE, '0' (DINT type) is output. When input data is TRUE, '1' (DINT type) is output. FALSE ⇨ 0 TRUE ⇨ 1 BOOL type → DINT type (b) The value input of input variable IN is BOOL type data.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation result</td><td>(1) Functions without EN/ENO pins.
Execute operation processing and output the operation output value from OUT.

(2) Functions With EN/ENO pins.
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

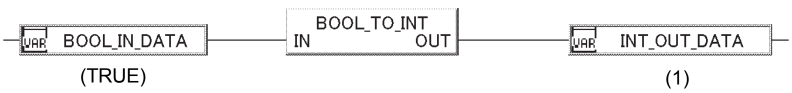
Error

There is no operation error caused by BOOL_TO_INT(_E), BOOL_TO_DINT(_E).

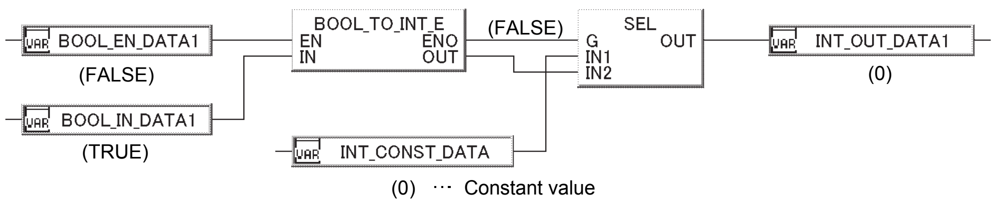
Program Example

(1) The program that converts BOOL type data input from input variable IN to INT type data, and output from output variable OUT.

(a) Basic program example (BOOL _TO_INT)

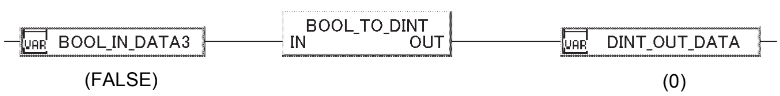


(b) The program example that outputs constant value when input variable EN is FALSE. (BOOL_TO_INT_E)



(2) The program that converts BOOL type data input from input variable IN to DINT type data, and output from the output variable OUT.

(a) Basic program example (BOOL_TO_DINT)



4.1.21 BOOL Type → WORD/DWORD Type Conversion (BOOL_TO_WORD(_E), BOOL_TO_DWORD(_E))

Function	FBD parts
BOOL_TO_WORD BOOL_TO_DWORD BOOL_TO_WORD_E BOOL_TO_DWORD_E	BOOL_TO_☐ (IN, OUT) (With EN/ENO pins) BOOL_TO_☐_E (EN, ENO, IN, OUT) ☐ Indicates WORD, DWORD

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Functions overview: Converts data type BOOL to WORD/DWORD.

Function/FB classification name: Type conversion function

Input and Output Pins

(1) BOOL_TO_WORD BOOL_TO_WORD_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	BOOL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	WORD	Output

(2) BOOL_TO_DWORD BOOL_TO_DWORD_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	BOOL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DWORD	Output

Function

Item	Contents
Operation processing	(1) BOOL_TO_WORD BOOL_TO_WORD_E (a) Convert the BOOL type data input from input variable IN to WORD type data and output from the output variable OUT When input data is FALSE, '0H' (WORD type) is output. When input data is TRUE, '1H' (WORD type) is output. FALSE ⇒ 0H TRUE ⇒ 1H BOOL type WORD type (b) The value input from input variable IN is BOOL type data. (2) BOOL_TO_DWORD BOOL_TO_DWORD_E (a) Convert the STRING type data input from input variable IN to DWORD type data, and output from the output variable OUT When input data is FALSE, '0H' (DWORD type) is output. When input data is TRUE, '1H' (DWORD type) is output. FALSE ⇒ 0H TRUE ⇒ 1H BOOL type DWORD type (b) The value input from input variable IN is BOOL type data.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation result</td><td>(1) Functions without EN/ENO pins.
Execute operation processing and output the operation output value from OUT pin.

(2) Functions With EN/ENO pins.
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

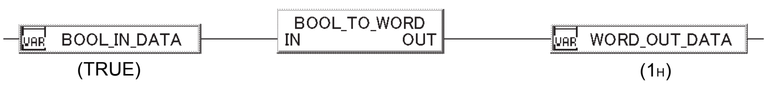
Error

There is no operation error caused by BOOL_TO_WORD(_E), BOOL_TO_DWORD(_E).

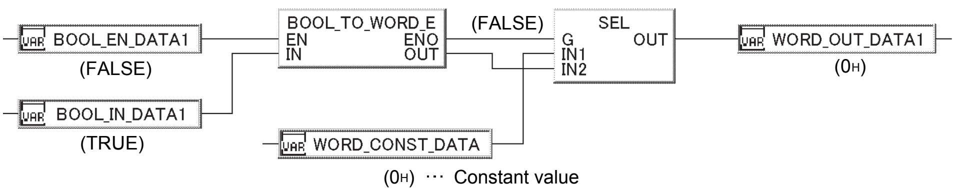
Program Example

(1) The program that converts BOOL type data input from input variable IN to WORD type data, and output from output variable OUT.

(a) Basic program example (BOOL _TO_WORD)

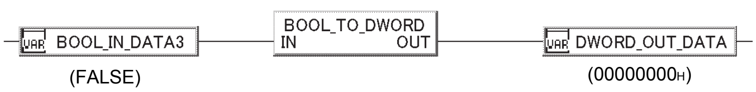


(b) The program example that outputs constant value when input variable EN is FALSE. (BOOL_TO_WORD_E)



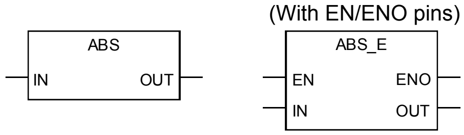
(2) The program that converts BOOL type data input from input variable IN to DWORD type data, and output from output variable OUT.

(a) Basic program example (BOOL_TO_DWORD)



4.2 Numerical Operation Function

4.2.1 Absolute Value (ABS(_E))

Function	FBD parts
ABS ABS_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

Function overview: Output absolute value of input value.

Function/FB classification name: Numerical operation function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	INT	Input
			DINT	
			REAL	
Output	ENO	Output variable	BOOL	Output condition (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output
			DINT	
			REAL	

Function

Item	Contents
Operation processing	(1) Through the same data type as input variable IN from the output variable OUT, output the absolute value of INT/DINT/REAL type data that have been input from input variable IN. The following equality is enabled assuming that the input value is A and operation output value is B. B=|A| (2) The input value from input variable IN is of INT/DINT/REAL type data value (3) If the operation result is outside the range of data type of output variable OUT, the input value is output as it is from OUT. If the data type of input variable IN is INT type and the input value is –32768, the output value from the output variable OUT will be –32768. If the data type of input variable IN is DINT type and the input value is –2147483648, the output value from the output variable OUT will be –2147483648. (The operation processing is error-free. Additionally, in the case of ABS_E, the output value from output variable ENO will be TRUE.)
Operation results	(1) Function without EN/ENO pins Execute operation processing and output operation output value from OUT. (2) Function With EN/ENO pins The execution conditions and the operation results are as follows: <table><tr><th>Execution condition</th><th colspan="2">Operation result</th></tr><tr><th>EN</th><th>ENO</th><th>OUT</th></tr><tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr><tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr></table>* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

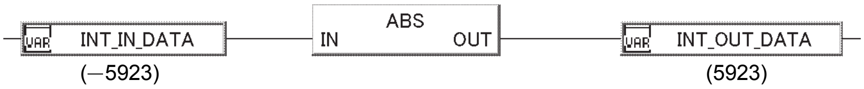
Error

There is no operation error caused by ABS(_E).

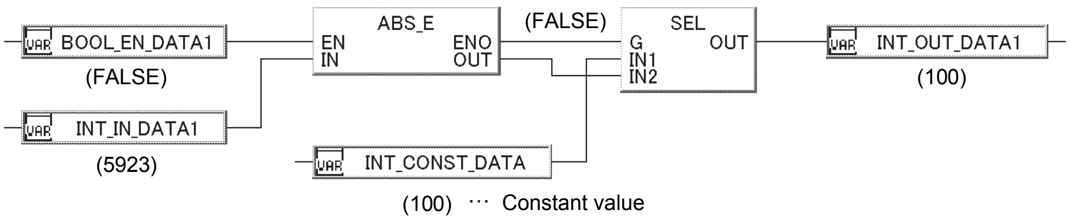
Program Example

In following program examples, the absolute value (of INT/DINT/REAL type data that are input from input variable IN through the same data type as input variable IN) is output from output variable OUT.

(1) Basic program example (ABS)



(2) The program example in which the output value is constant value when the input variable EN is FALSE. (ABS_E)



4.2.2 Square Root (SQRT(_E))

Function	FBD parts
SQRT SQRT_E	SQRT: IN → OUT (With EN/ENO pins) SQRT_E: EN → ENO, IN → OUT

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Output square root of input value.
Function/FB classification name: Numerical operation function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	REAL	Output

Function

Item	Contents
Operation processing	(1) From output variable OUT, output the square root of the REAL type data that are input from input variable IN. The following equality is enabled assuming that the input value is A and the operation output value is B. $B=\sqrt{A}$ (2) The input value of input variable IN is of REAL type within the range of positive number.
Operation results	(1) Function without EN/ENO pins The operation results are as follows: (see table 1) (2) Function with EN/ENO pins The execution conditions and the operation results are as follows: (see table 2)

(1) Function without EN/ENO pins

Operation result	OUT
No operation error	Operation output value
Operation error occurs	Undefined value

(2) Function with EN/ENO pins

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error occurs) (*)	Undefined value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Error

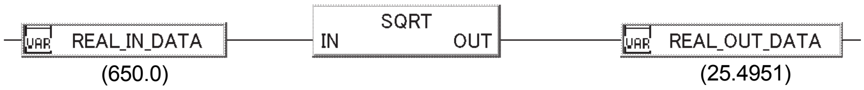
Following situations indicate operation error occurring, and the error codes will be displayed on the FBD program diagnostic screen of PX Developer programming tool.

- When the input value is negative number. (Error code: Refer to Appendix 2)

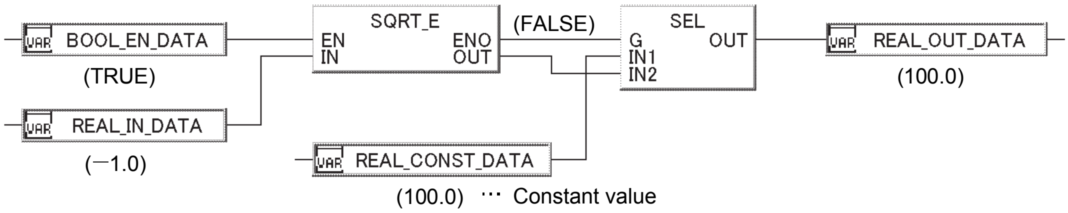
Program Example

In following program examples, output the square root of REAL type data from output variable OUT that are input from input variable IN.

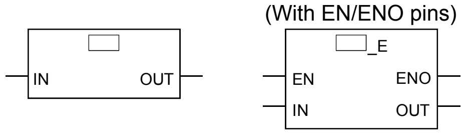
(1) Basic program example (SQRT)



(2) The program example (SQRT_E) in which the output is constant value when the input variable EN is FALSE or operation error occurs.
(Example) When operation errors occur



4.2.3 Natural Logarithm/Common Logarithm (LN(_E), LOG(_E))

Function	FBD parts
LN LOG LN_E LOG_E	(With EN/ENO pins)  □ indicates LN,LOG

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: LN(_E) Output natural logarithm operation result of input value. LOG(_E) Output common logarithm operation result of input value.

Function/FB classification name: Numerical operation function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	REAL	Output

Function

Item	Contents
Operation processing	(1) LN LN_E This processing performs natural logarithm (using (e) as its base number) on the REAL type data that are input from input variable IN and outputs the result from the operation output variable OUT. The following equality is enabled assuming that the input value is A and the operation output value is B. $B=\log e^{A}$ In natural logarithm operation, the base number (e) is "2.71828" (2) LOG LOG_E This processing performs common logarithm (using (10) as its base number) on the REAL type data that are input from input variable IN and outputs result from the output variable OUT. The following equality is enabled assuming that the input value is A and the output value is B. $B=\log 10^{A}$ (3) If the input value of input variable IN is REAL type data value, it should be in positive number range.
Operation results	(1) Function without EN/ENO pins The operation results are as follows: (see table 1 below) (2) Function with EN/ENO pins The execution conditions and the operation results are as follows: (see table 2 below) * When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Operation result	OUT
No operation error	Operation output value
Operation error occurs	Undefined value

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error occurs) (*)	Undefined value
FALSE (Operation stop)	FALSE (*)	Undefined value

Error

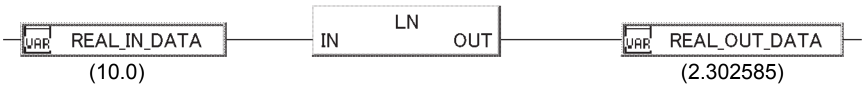
Following situations indicate operation error occurring, and the error codes will be displayed on the FBD program diagnostic screen of PX Developer programming tool.

- When input value is negative number. (Error code: Refer to Appendix 2)
- When the converted data are 0 or out of the range"±1.17549^{-38} to ±3.40282^{+38}". (Error code: Refer to Appendix 2)

Program Example

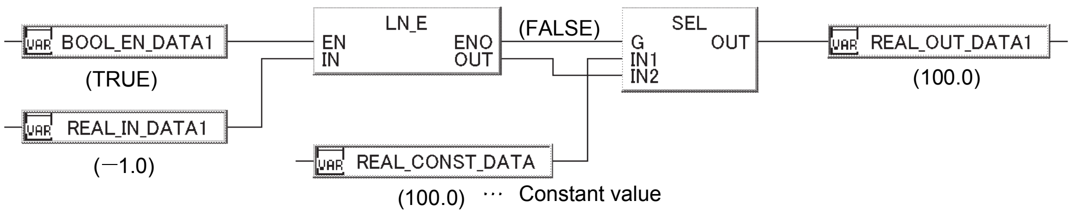
(1) Following are the program examples in which natural logarithm operation (of REAL type data input from input variable IN with (e) as the base number) is executed, and the result is output from the output variable OUT.

(a) Basic program example (LN)



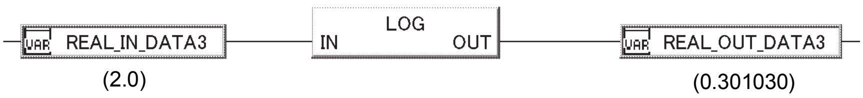
(b) This is a program example in which the output is constant value when the input variable EN is FALSE or operation error occurs. (LN_E)

(Example) When operation errors occur

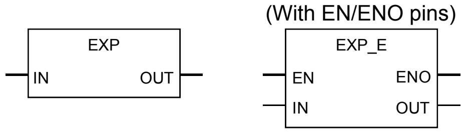


(2) This is a program example in which the common logarithm (of REAL type data input from input variable IN with (10) as the base number) is executed, and the result is output from the output variable OUT.

(a) Basic program example (LOG)



4.2.4 Natural Exponential (EXP(_E))

Function	FBD parts
EXP EXP_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Output the natural exponential result of input value.

Function/FB classification name: Numerical operation function

Input and Output Pins

Pins	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	REAL	Output

Function

Item	Contents
Operation processing	(1) This processing performs natural exponential on REAL type data input from input variable IN and outputs the result from the output variable OUT. The following equality is enabled assuming that the input value is A and the operation output value is B. $B=e^{A}$ In natural exponential, the base number (e) is "2.71828". (2) The input value of input variable IN is REAL type data.
Operation results	See below

(1) Function without EN/ENO pins

The operation results are as follows:

Operation result	OUT
No operation error	Operation output value
Operation error occurs	Undefined value

(2) Function with EN/ENO pins

The execution conditions and the operation results are as follows:

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error occurs) (*)	Undefined value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Error

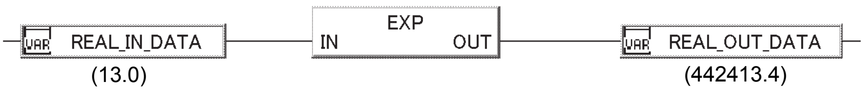
Following situations indicate operation error occurring, and the error codes will be displayed on the FBD program diagnostic screen of PX Developer programming tool.

- When the converted data are beyond the range of $\pm1.17549^{-38}$ to $\pm3.40282^{+38}$. (Error code: Refer to Appendix 2)

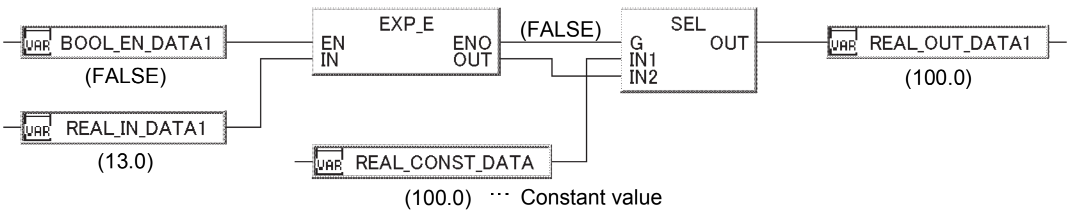
Program Example

Following are the program examples in which natural exponential (of REAL type data input from input variable IN) is executed, and the result is output from output variable OUT.

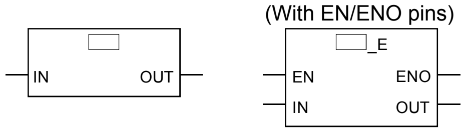
(1) Basic program example (EXP)



(2) This is a program example in which the output is constant value when input variable EN is FALSE or operation error occurs. (EXP_E)
(Example) When input variable EN is FALSE



4.2.5 SIN/COS/TAN Operation (SIN(_E), COS(_E), TAN(_E))

Function	FBD parts
SIN COS TAN SIN_E COS_E TAN_E	(With EN/ENO pins)  ☐ indicates SIN, COS, TAN

With EN/ENO pins	○
Overload	—
Input pin number changeable (range)	—

Function overview: SIN(_E) outputs the SIN (sine) of input value
COS(_E) outputs the COS (cosine) of input value
TAN(_E) outputs the TAN (tangent) of input value

Function/FB classification name: Numerical operation function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	REAL	Output

Function

Item	Contents
Operation processing	(1) SIN SIN_E This processing performs SIN (sine) operation on the REAL type data input from input variable IN and outputs the result from the output variable OUT. The following equality is enabled assuming that the input value is A and the operation output value is B. B=SIN A (2) COS COS_E This processing performs COS (cosine) operation on the REAL type data input from input variable IN and outputs the result from the output variable OUT. The following equality is enabled assuming that the input value is A and the operation output value is B. B=COS A (3) TAN TAN_E This processing performs TAN (tangent) operation on the REAL type data input from input variable IN and outputs the result from the output variable OUT. The following equality is enabled assuming that the input value is A and the operation output value is B. B=TAN A Take care of radian value operation error that is hard to avoid even if the input value is π/2 radian or (3/2) π radian, so the error should be neglected here. (4) The value (angle) input from input variable IN is REAL type data. Please input value in radian unit (angle $\times$ π/180).

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Function without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Function with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

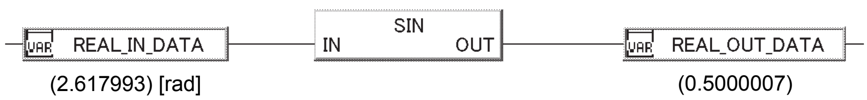
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the input value is –0 (Error code: Refer to Appendix 2)
- In the case of TAN(_E), the converted data are 0 or out of the range" $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$".
  (Error code: Refer to Appendix 2)

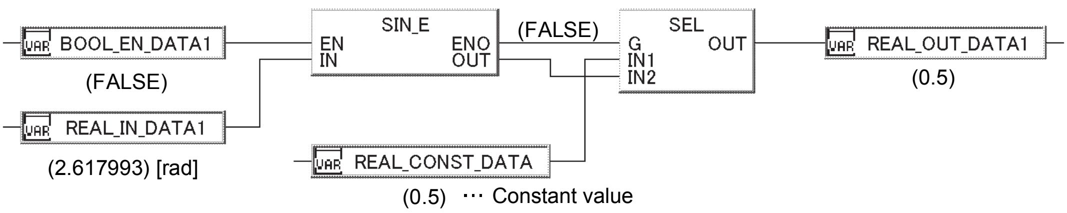
Program Example

(1) Following are the program examples in which SIN (sine) operation (of the REAL type data (angle) input from the input variable IN) is executed, and the result is output from output variable OUT.

(a) Basic program example (SIN)

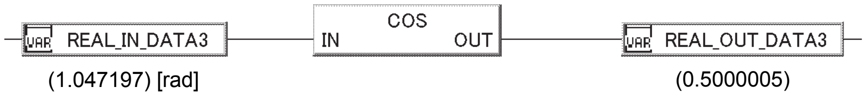


(b) This is a program example in which the output result is constant value when the input variable EN is FALSE or operation error occurs.
(Example) When the input variable EN is FALSE



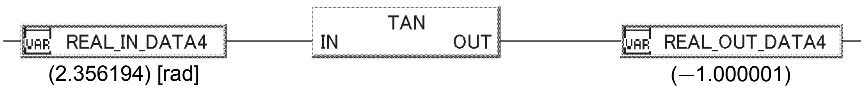
(2) This is a program example in which COS (cosine) operation (of the REAL type data (angle) input from the input variable IN) is executed, and the result is output from the output variable OUT.

(a) Basic program example (COS)

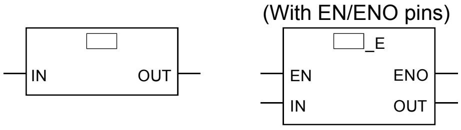


(3) This is a program example in which TAN (tangent) operation (of the REAL type data (angle) input from the input variable IN) is executed, and the result is output from the output variable OUT.

(a) Basic program example (TAN)



4.2.6 ASIN/ACOS/ATAN Operation (ASIN(_E), ACOS(_E), ATAN(_E))

Function	FBD parts
ASIN ACOS ATAN ASIN_E ACOS_E ATAN_E	(With EN/ENO pins)  ☐ indicates ASIN, ACOS, ATAN

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: $\boxed{\text{ASIN(\_E)}}$ outputs the SIN^{-1} (principal arc sine) of input value $\boxed{\text{ACOS(\_E)}}$ outputs the COS^{-1} (principal arc cosine) of input value $\boxed{\text{ATAN(\_E)}}$ outputs the TAN^{-1} (principal arc tangent) of input value

Function/FB classification name: Numerical operation function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	REAL	Output

Function

Item	Contents
Operation processing	(1) $\boxed{\text{ASIN}}$ $\boxed{\text{ASIN\_E}}$ This processing performs ASIN (arc sine) operation on the REAL type data input from input variable IN and outputs the result from output variable OUT. The following equality is enabled assuming the input value is A and the operation output value is B. B=SIN^{-1}A (2) $\boxed{\text{ACOS}}$ $\boxed{\text{ACOS\_E}}$ This processing performs ACOS (arc cosine) operation on the REAL type data input from input variable IN and outputs the result from output variable OUT. The following equality is enabled assuming the input value is A and the operation output value is B. B=COS^{-1}A (3) $\boxed{\text{ATAN}}$ $\boxed{\text{ATAN\_E}}$ This processing performs ATAN (arc tangent) operation on the REAL type data input from input variable IN and outputs the result from output variable OUT. The following equality is enabled assuming the input value is A and the operation output value is B. B=TAN^{-1}A (4) The range of the REAL type data input into input variable IN is shown as follows: ASIN(_E), ACOS(_E) : -1.0 to 1.0 ATAN(_E) : $\pm1.17549^{-38}$ to $\pm3.40282^{+38}$ (5) The output value (angle) of output variable OUT uses radian (angle $\times$ π/180) as its unit.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Function without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Function with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

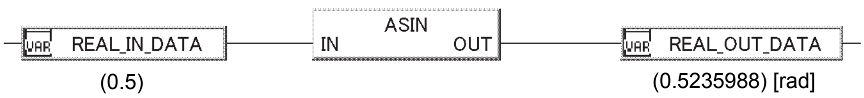
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the input value is -0. (Error code: Refer to Appendix 2)
- In the case of ASIN(_E) and ACOS(_E), the input value is beyond the range of -1.0 to 1.0. (Error code: Refer to Appendix 2)

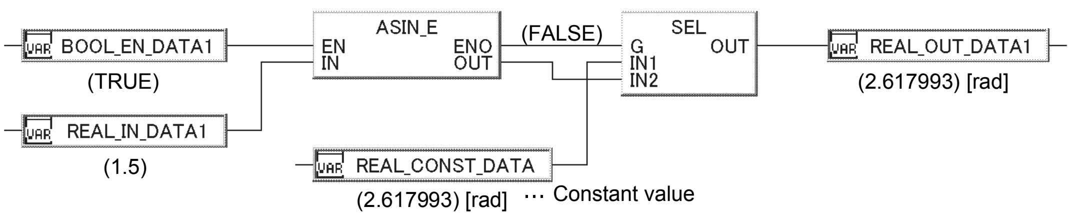
Program Example

(1) Following are the program examples in which ASIN (arc sine) operation (of the REAL type data input from input variable IN) is executed, and the result is output from output variable OUT.

(a) Basic program example (ASIN)

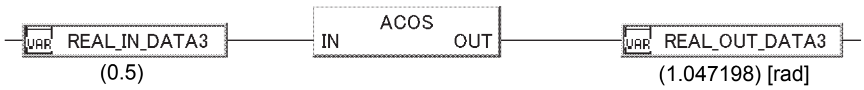


(b) This is a program example in which the output is constant value when the input variable EN is FALSE or an operation error occurs. (ASIN_E)
(Example) When operation errors occur



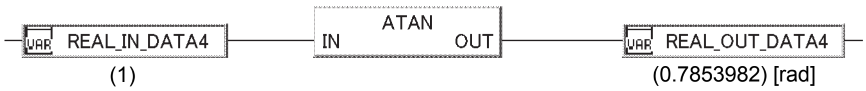
(2) This is a program example in which ACOS (arc cosine) operation (of the REAL type data input from input variable IN) is executed, and the result is output from the output variable OUT.

(a) Basic program example (ACOS)

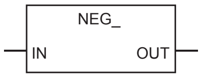
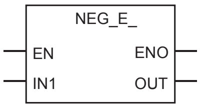


(3) This is a program example in which ATAN (arc tangent) operation (of the REAL type data input from input variable IN) is executed, and the result is output from the output variable OUT.

(a) Basic program example (ATAN)



4.2.7 Sign Reversal (NEG(_E)_)

Function	FBD parts
NEG_ NEG_E_	 (With EN/ENO pins) 

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

Function overview: Reverses the sign of an input data.

Function/FB classification name: Numerical operation function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	INT DINT REAL	Input
Output	ENO	Output variable	BOOL	Output condition (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT DINT REAL	Output

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(1) Reverse sign of INT/DINT/REAL type data that have been input from input variable IN, output the same data type as input variable IN from the output variable OUT.
The following equality is enabled assuming that the input value is A and operation output value is B.
B= -A
(2) The input value from input variable IN is of INT/DINT/REAL type data value
(3) If the operation result is outside the range of data type of output variable OUT, the input value is output as it is from OUT.
If the data type of input variable IN is INT type and the input value is –32768, the output value from the output variable OUT will be –32768.
If the data type of input variable IN is DINT type and the input value is –2147483648, the output value from the output variable OUT will be –2147483648.
(The operation processing is error-free. Additionally, in the case of NEG_E_, the output value from output variable ENO will be TRUE.)
</td></tr>
<tr><td>Operation results</td><td>
(1) Function without EN/ENO pins
Execute operation processing and output operation output value from OUT.
(2) Function With EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

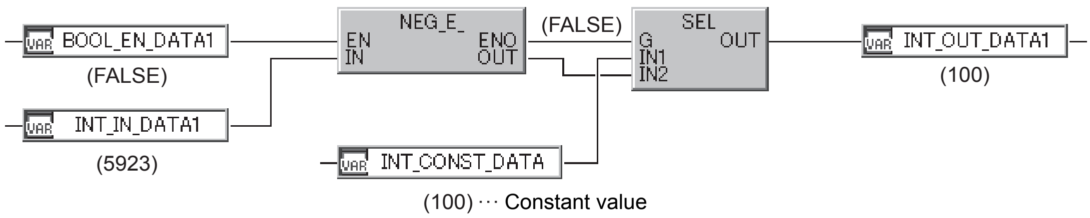
There is no operation error caused by NEG(_E)_.

Program Example

A program which reverses a sign of INT/DINT/REAL type data that have been input from input variable IN, output the same data type as input variable IN from the output variable OUT.

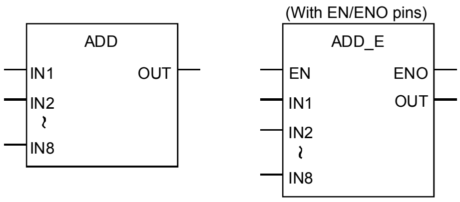
(1) Basic program example (NEG_)

(2) The program example in which the output value is constant value when the input variable EN is FALSE. (NEG_E_)



4.3 Arithmetic Operation Function

4.3.1 Addition (ADD(_E))

Function	FBD parts
ADD ADD_E	 Pin number of input variable IN can be changed in the range of 2 to 8.

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	2 to 8

Function overview: Output the sum of input value (IN1+IN2+ • • • +IN8)

Function/FB classification name: Arithmetic operation function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN1 to IN8	Input variable	INT	Input
			DINT	
			REAL	
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output
			DINT	
			REAL	

Function

Item	Contents
Operation processing	(1) This processing performs addition operation (IN1+IN2+...+IN8) on the REAL type data input from input variables IN1 to IN8 and outputs the result from the output variable OUT in the same data type as that of input variable IN. (Example) when data type is INT $\underbrace{\boxed{1234}}_{\text{IN1 (INT type)}} + \underbrace{\boxed{5678}}_{\text{IN2 (INT type)}} \Rightarrow \underbrace{\boxed{6912}}_{\text{INT type}}$ (2) The value input from input variables IN1 to IN8 is INT/DINT/REAL type value. (3) Pin number of input variable IN can be changed in the range of 2 to 8. (4) In the case of the underflow/overflow of operation results, the output status of output variable OUT is shown as follows: (a) When the data type is INT Operation error will not occur even if underflow/overflow occurs. Besides, TRUE will be output from output variable ENO in the case of ADD_E. 32767+ 2 = -32767 It is a negative value because the highest bit is 1. (7FFFH) (0002H) (8001H) -32767+ (-2) = 32766 It is a positive value because the highest bit is 0. (8000H) (FFFEH) (7FFEH)

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>(b) When the data type is DINT
Operation error will not occur even if underflow/overflow occurs.
Besides, TRUE will be output from output variable ENO in the case of ADD_E.
2147483647 + 2 = -2147483647 It is negative number because the highest digit is 1.
(7FFFFFFFH) (0002H) (80000001H)

-2147483648 + (-2) = 2147483646 It is positive number because the highest digit is 0.
(80000000H) (FFFEH) (7FFFFFFEH)

(c) When the data type is REAL
An operation error occurs, and undefined value is output.</td></tr>
<tr><td>Operation results</td><td>(1) Function without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Execution result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Function with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

Error

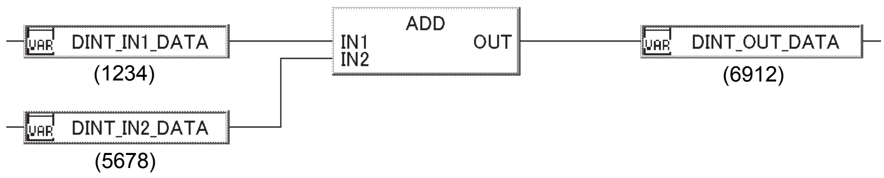
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the data type is REAL type, the input value and output value are not 0 and beyond the range of $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$. (Error code: Refer to Appendix 2)

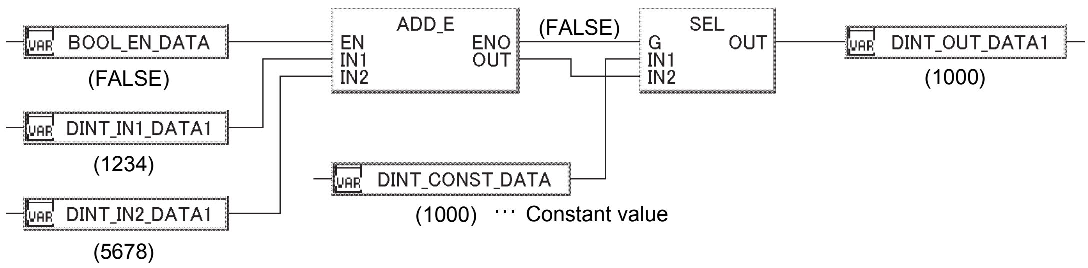
Program Example

Following are the program example in which addition operation (IN1+IN2+...+IN8) (of the INT/DINT/REAL type data input from input variable IN1 to IN8) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

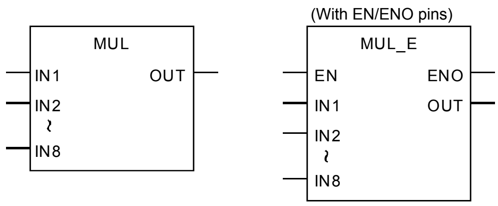
(1) Basic program example (ADD)



(2) This is a program example in which the output is constant value when the input variable EN is FALSE or operation error occurs.
(Example) When the input variable EN is FALSE



4.3.2 Multiplication (MUL(_E))

Function	FBD parts
MUL MUL_E	 Pin number of input variable IN can be changed within range of 2 to 8.

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	2 to 8

Function overview: Output the product (IN1 × IN2 × • • • × IN8) of input value.

Function/FB classification name: Arithmetic operation function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN1 to IN8	Input variable	INT DINT REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT DINT REAL	Output

Function

Item	Contents
Operation processing	(1) This processing performs multiplication operation (IN1 × IN2 × ... × IN8) on the INT/DINT/REAL type data input from input variables IN1 to IN8 and outputs the operation result from output variable OUT in the same data type as that of the input variable IN. 100 × 15 ⇒ 1500 IN1 (INT type) IN2 (INT type) INT type (2) The value input from input variables IN1 to IN8 is INT/DINT/REAL type value. (3) Pin number of input variable IN can be changed in the range of 2 to 8. (4) In the case of underflow/overflow of operation results, the output status of output variable OUT is as follows: (a) When the data type is INT type Operation error will not occur in operation even in the case of underflow/overflow occurs. Besides, TRUE will be output from output variable ENO in the case of MUL_E. If the operation result exceeds the range of INT type data, the output will still be INT type data. (The operation result is of DINT type but the output will be INT type data after the deletion of high-order 16 bits) If the operation result exceeds INT type data range, please convert the result into DINT type data via INT_TO_DINT before performing operation.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(b) When the data type is DINT type
Operation error will not occur even in the case of underflow/overflow occurs.
Besides, TRUE will be output from output variable ENO in the case of MUL_E.
If the operation result exceeds the range of DINT type data, the output will still be DINT type data.
(The operation result is 64-bits data but the output data will be DINT type data after the deletion of high-order 32 bits)
If the operation result exceeds the DINT type data range, please convert the REAL type data via DINT_TO_REAL before performing operation.

(c) When the data type is of REAL type
An operation error occurs, and undefined value is output.
</td></tr>
<tr><td>Operation results</td><td>
(1) Function without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Execution result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Function with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

POINT

When the operation result exceeds the data type range, please convert the data type of the input value before performing operation.

Error

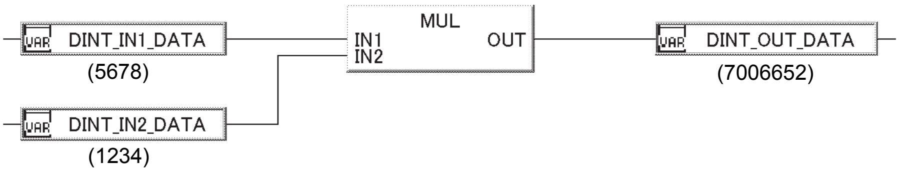
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the data type is REAL, the input value and output value are not 0 and beyond the range of $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$. (Error code: Refer to Appendix 2)

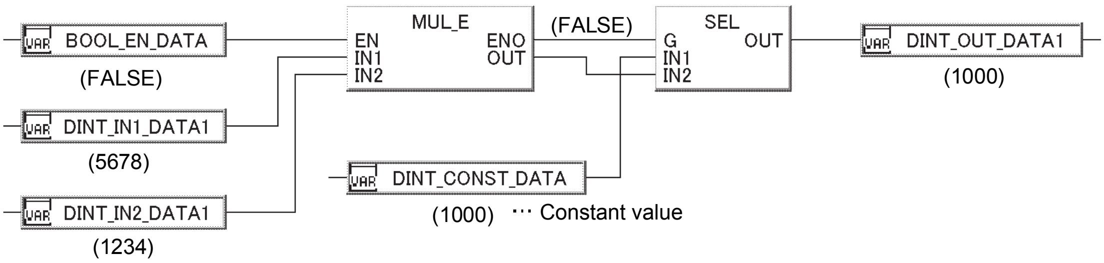
Program Example

Following are the program examples in which multiplication operation (IN1×IN2×...×IN8) (of the IN/DINT/REAL type data input from input variables IN1 to IN8) is executed, then the operation result is output from the output variable OUT in the same data type with input variable IN.

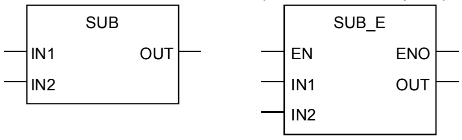
(1) Basic program example (MUL)



(2) This is a program example in which the output is constant value when the input variable EN is FALSE or operation error occurs.
(Example) When the input variable EN is FALSE



4.3.3 Subtraction (SUB(_E))

Function	FBD parts
SUB SUB_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

Function overview: Output the difference (IN1-IN2) of the input values

Function/FB classification name: Arithmetic operation function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN1 IN2	Input variable	INT DINT REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT DINT RAL	Output

Function

Item	Contents
Operation processing	(1) This processing performs subtraction operation (IN1-IN2) on the INT/DINT/REAL type data input from input variables IN1 and IN2 and outputs the operation result from output variable OUT in the same data type as that of input variable IN. (Example) When data type is INT type [12345] – [6789] ⇒ [5556] IN1 (INT type) IN2 (INT type) INT type (2) The input value of variables IN1 and IN2 is INT/DINT/REAL type data value. (3) In the case of underflow/overflow, the output status of output variable OUT is as follows: (a) When the data type is INT type Operation error will not occur even in the case of underflow/overflow occurs. Besides, TRUE will be output from output variable ENO in the case of SUB_E. 32767 - (-2) = -32767 It is a negative value because the highest bit is 1 (7FFFH) (FFFEH) (8001H) -32767 - 2 = 32766 It is a positive value because the highest bit is 0. (8000H) (0002H) (7FFEH)

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(b) When the data type is DINT type
Operation error will not occur even in the case of underflow/overflow occur.
Besides, TRUE will be output from output variable ENO in the case of SUB_E.
2147483647 + 2 = -2147483647 It is a negative value because the highest bit is 1.
(7FFFFFFFH) (0002H) (80000001H)
-2147483648 + (-2) = 2147483646 It is a positive value because the highest bit is 0.
(80000000H) (FFFEH) (7FFFFFFEH)
(c) When the data type is REAL
An operation error occurs, and undefined value is output.
</td></tr>
<tr><td>Operation results</td><td>
(1) Function without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Function With EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

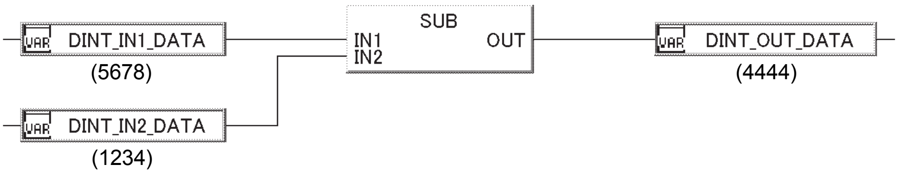
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer.

- When the data type is REAL, input/output values are not 0 and beyond the range of $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$. (Error code: Refer to Appendix 2)

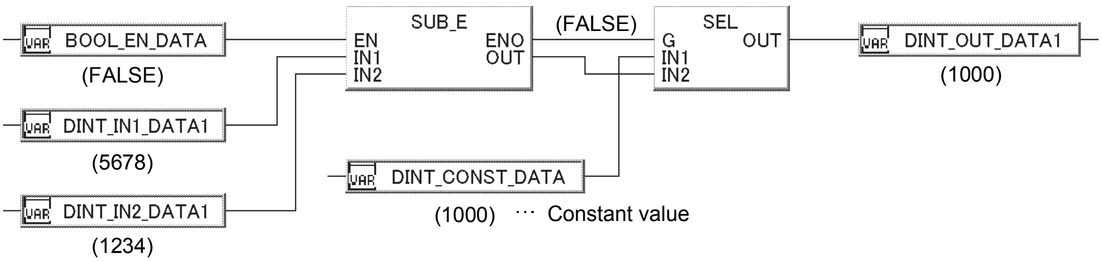
Program Example

Following are the program example in which subtraction operation (IN1-IN) (of the INT/DINT/REAL type data input from input variable IN1, IN2) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

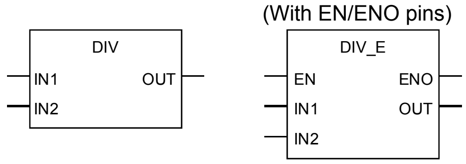
(1) Basic program example (SUB)



(2) This is a program example in which the output is constant value when the input variable EN is FALSE or operation error occurs. (SUB_E)
(Example) When the input variable EN is FALSE



4.3.4 Division (DIV(_E))

Function	FBD parts
DIV DIV_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

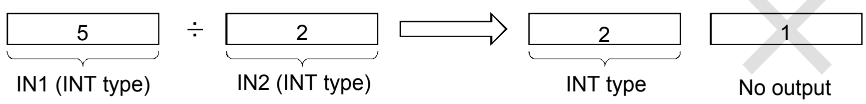
Function overview: Output the quotient (IN1÷IN2) of input values.

Function/FB classification name: Arithmetic operation function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN1 IN2	Input variable	INT DINT REAL	Input
Output	END	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT, DINT, REAL	Output

Function

Item	Contents
Operation processing	(1) This processing performs division operation (IN1 ÷ IN2) on the INT/DINT/REAL type data input from input variables IN1, IN2, and outputs the result of quotient from the output variable OUT in the same data type as that of input variable IN. (Example) When the data type is INT type  (2) The value input from input variable IN1 and IN2 is INT/DINT/REAL type value. (However, the input value of input variable IN2 cannot be 0)
Operation results	(1) Function without EN/ENO pins The operation results are as follows: (see table A) (2) Function with EN/ENO pins The execution conditions and the operation results are as follows: (see table B) * When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Operation result	OUT
No operation error	Operation output value
Operation error occurs	Undefined value

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error occurs) (*)	Undefined value
FALSE (Operation stop)	FALSE (*)	Undefined value

Error

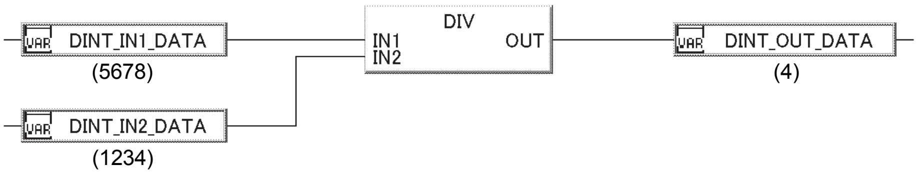
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the data type is REAL or the input/output value is not within the following range: (Error code: Refer to Appendix 2)
  Input value: $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$
  Output value: 0, $\pm 1.17549^{-38}$ to $\pm 3.40282^{+38}$
- When input value of input variable IN2 is 0. (Error code: Refer to Appendix 2)

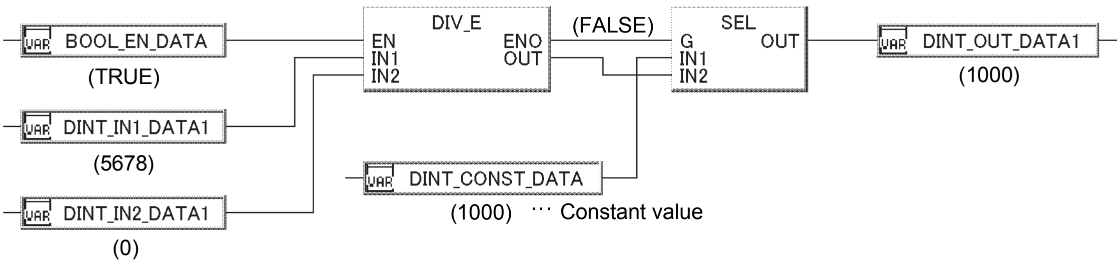
Program Example

Following are the program example in which division operation (IN1 ÷ IN2) (of the INT/DINT/REAL type data input from input variable IN1 and IN2) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

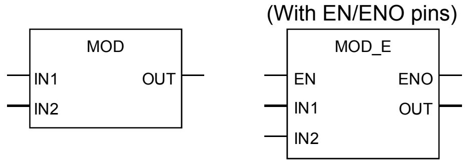
(1) Basic program example (DIV)



(2) This is a program example in which the output is constant value when the input variable EN is FALSE or operation error occurs. (DIV_E)
(Example) When operation errors occur



4.3.5 Modulus Operation (MOD(_E))

Function	FBD parts
MOD MOD_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

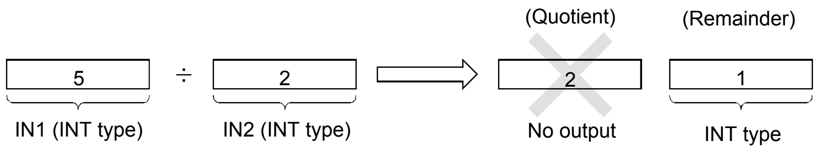
Function overview: Output the remainder value (IN1÷IN2) of input values

Function/FB classification name: Arithmetic operation function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN1 IN2	Input variable	INT DINT REAL	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT DINT	Output

Function

Item	Contents
Operation processing	(1) This processing performs division operation (IN1 ÷ IN2) on the REAL type data input from input variables IN1 and IN2 and outputs the result from the output variable OUT in the same data type as that of input variable IN. (Example) When the data type is INT type  (2) The value input from input variables IN1 and IN2 is INT/DINT type value. (However, the input value of input variable IN2 cannot be 0.)
Operation results	(1) Function without EN/ENO pins The operation results are as follows: Operation result / OUT No operation error / Operation output value Operation error occurs / Undefined value (2) Function With EN/ENO pins The execution conditions and operation results are as follows: Execution condition EN / Operation result ENO / OUT TRUE (Operation execution) / TRUE (No operation error) / Operation output value TRUE (Operation execution) / FALSE (Operation error occurs) (*) / Undefined value FALSE (Operation stop) / FALSE (*) / Undefined value * When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Error

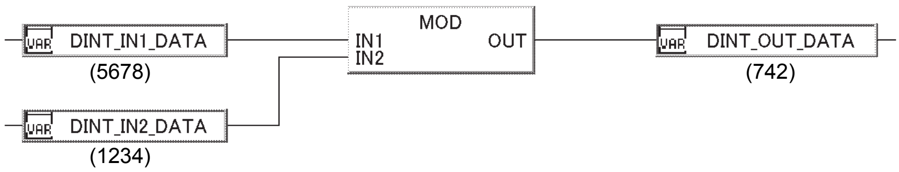
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When input value of input variable IN2 is 0. (Error code: Refer to Appendix 2)

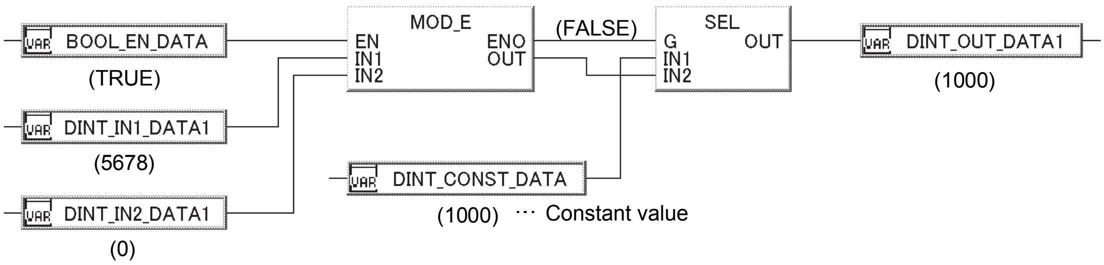
Program Example

Following are the program example in which division operation (IN1÷IN2) (of the INT/DINT/REAL type data input from input variable IN1 and IN2) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

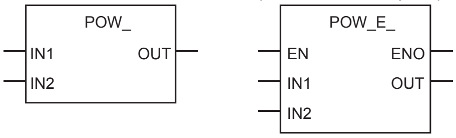
(1) Basic program example (MOD)



(2) This is a program example in which the output is constant value when the input variable EN is FALSE or operation error occurs. (MOD_E)
(Example) When operation errors occur



4.3.6 Exponentiation (POW(_E)_)

Function	FBD parts
POW_ POW_E_	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

Function overview: Returns the value of IN1 to the power IN2.

Function/FB classification name: Arithmetic operation function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN1 IN2	Input variable	INT	Input
			DINT	
			REAL	
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output
			DINT	
			REAL	

Function

Item	Contents
Operation processing	(1) This processing performs exponentiation operation ($IN1^{IN2}$) on the INT/DINT/REAL type data input from input variables IN1 and IN2 and outputs the operation result from output variable OUT in the same data type as that of input variable IN. Input value IN1 is A, IN2 is B, operation output value is C is described as shown below. $C=A^B$ (2) The input value of variables IN1 and IN2 is INT/DINT/REAL type data value. (3) When data type is INT/DINT type, and input negative number to IN2, 1 is output from output variable OUT. (4) In the case of underflow/overflow, the output status of output variable OUT is as follows: (a) When the data type is INT type An operation error occurs, and undefined value is output. If the operation result exceeds the INT type data range, convert the DINT type data via INT_TO_DINT before performing operation. (b) When the data type is DINT type If the operation result exceeds the DINT type data range, the output will still be the DINT type data. (The operation result is 64-bits data, however, the output data will be the DINT type data after the deletion of high-order 32 bits) If the operation result exceeds the DINT type data range, convert the REAL type data via DINT_TO_REAL before performing operation. (c) When the data type is REAL type An operation error occurs, and undefined value is output.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Function without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Function With EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

POINT

When the operation result exceeds the data type range, convert the data type of the input value before performing operation.

Error

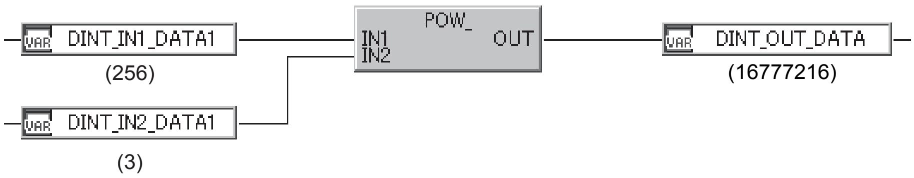
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer.

- When the data type is DINT, and IN2 is other than -32768 to 32767. (Error code: Refer to Appendix 2)
- When the data type is REAL, and input value (IN1) is negative number. (Error code: Refer to Appendix 2)
- When the data type is REAL, and operation result is not the range of less 2^{128}. (Error code: Refer to Appendix 2)

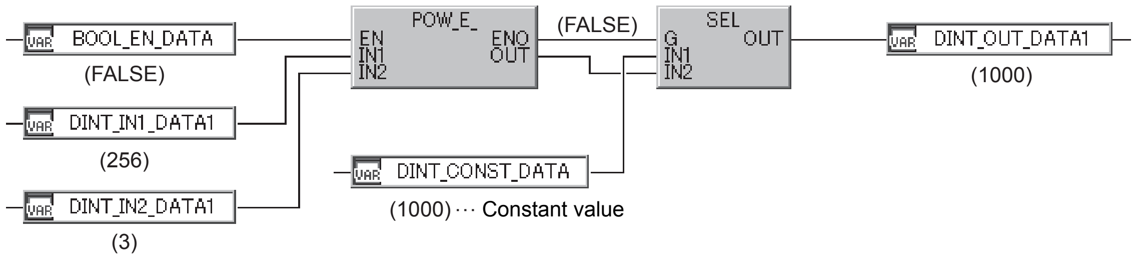
Program Example

Following are the program example in which exponentiation operation ($IN1^{IN2}$) (of the INT/DINT/REAL type data input from input variable IN1, IN2) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

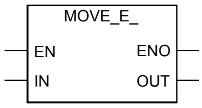
(1) Basic program example (POW_)



(2) This is a program example in which the output is constant value when the input variable EN is FALSE or operation error occurs. (POW_E_)
(Example) When the input variable EN is FALSE



4.3.7 Transfer (MOVE_E_)

Function	FBD parts
MOVE_E_	

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

Function overview: Output the input value (IN).

Function/FB classification name: Arithmetic operation function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	BOOL INT DINT WORD DWORD REAL ADR_REAL STRING(255)	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	BOOL INT DINT WORD DWORD REAL ADR_REAL STRING(255)	Output

Function

Item	Contents
Operation processing	(1) Output the value of input variable IN with the same data type as input variable IN from the output variable OUT. (2) The input value of variable IN is BOOL/INT/DINT/WORD/DWORD/REAL/ADR_REAL/STRING type data value.
Operation results	The execution conditions and the operation results are shown as follows: (see table below) * When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE (*)	Undefined value

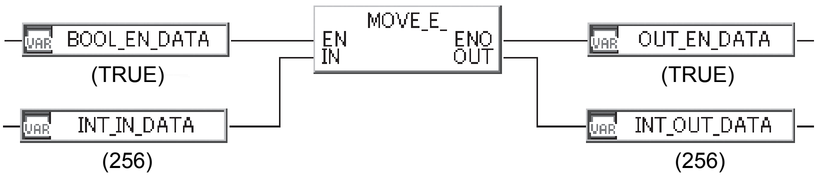
Error

There is no operation error caused by MOVE_E_.

Program Example

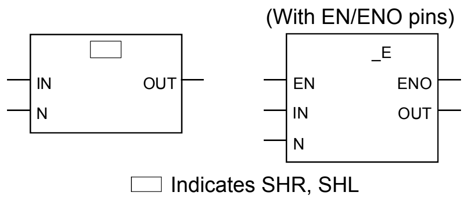
The program that outputs INT type data input to input variable IN from output variable OUT when input variable EN is TRUE.

(1) Basic program example (MOVE_E_)



4.4 Bit-string Function

4.4.1 Shift Left, Shift Right (SHL(_E), SHR(_E))

Function	FBD parts
SHL SHR SHL _E SHR_E	

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

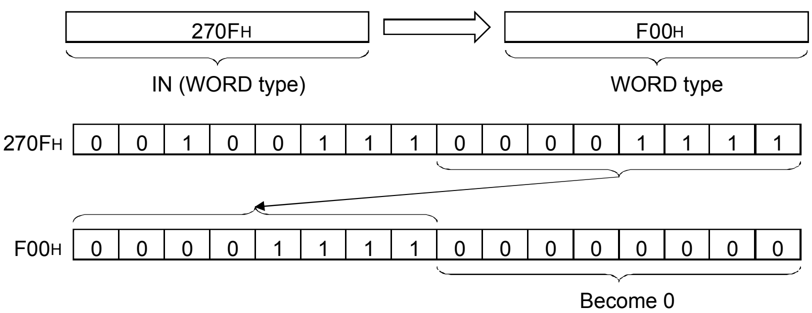
Function overview: SHL(_E) shifts the input value to the left by n bits, and then outputs the result.
SHR(_E) shifts the input value to the right by n bits, and then outputs the result.

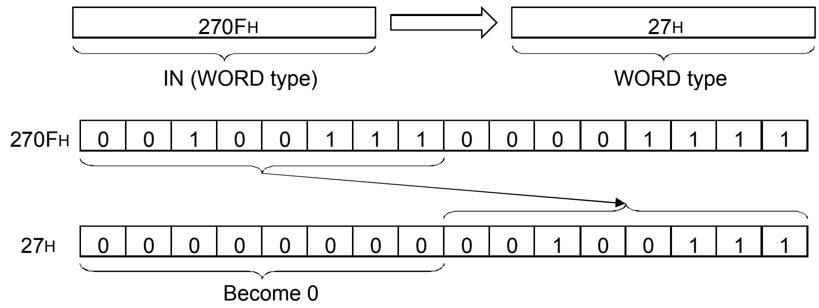
Function/ FB classification name: Bit-string function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	WORD DWORD	Input
	N	Input variable	INT	Shift bit number specification
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	WORD DWORD	Output

Function

Item	Contents
Operation processing	(1) SHL SHL_E (a) Shift to the left by n bits the WORD/DWORD type data that is input from input variable IN, and output the result from output variable OUT in the same data type as that of input variable IN. The bit number n for shifting left is the input value of input variable N. (Example) The following is an example when the data type of input variable IN is WORD type and the input value of input variable N is 8.  (b) The first "n" bits become 0.

Item	Contents
Operation processing	(2) SHR SHR_E (a) Shift to the right by n bits the WORD/DWORD type data that is input from input variable IN, and output the result from output variable OUT in the same data type as that of input variable IN. The bit number n for shifting left is the input value of input variable N. (Example) The following is an example when the data type of input variable IN is WORD type and the input value of input variable IN is 8.  (b) The first "n" bits become 0. (3) The value input from input variable IN is WORD/ DWORD type data. (4) The value input from input variable N (shift bit number specification) is INT type data, and its range is as follows. (a) When the data type of input variable IN is WORD type The input value of variable N is within 0 to 15. Applying the 4 low-order bits of the value input from the input variable N. Example When input value is 6 : 6 When input value is 22 : 6 (b) When the data type of input variable IN is DWORD type The input value of variable N is within 0 to 31. Applying the 5 low-order bits of the value input from the input variable N. Example When input value is 6 : 6 When input value is 22 : 22
Operation results	(1) Function without EN/ENO pins Execute operation processing and output operation output value from OUT. (2) Function with EN/ENO pins The execution conditions and the operation results are as follows: [see table below] * When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE (*)	Undefined value

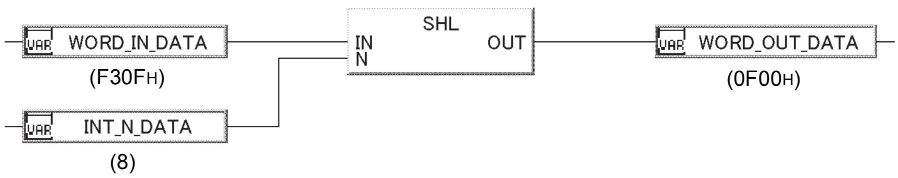
Error

There is no operation error caused by SHL(_E), SHR(_E).

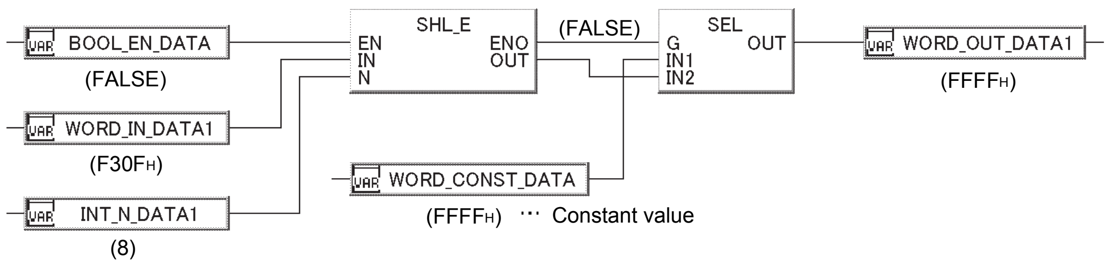
Program Example

(1) Following are the program example in which left shift operation (of the WORD/DWORD type data input from input variable IN) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

(a) Basic program example (SHL)

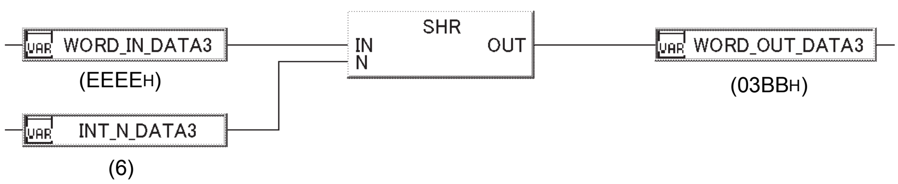


(b) This is a program example in which the output is constant value when the input variable EN is FALSE. (SHL_E).

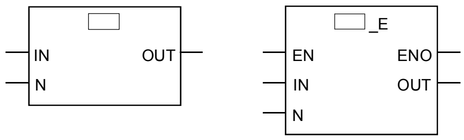


(2) Following is the program example in which right shift operation (of the WORD/DWORD type data input from input variable IN) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

(a) Basic program example (SHR)



4.4.2 Rotate Left, Rotate Right (ROL(_E), ROR(_E))

Function	FBD parts
ROL ROR ROL _E ROR_E	(With EN/ENO pins)  ▭ Indicates ROL, ROR.

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

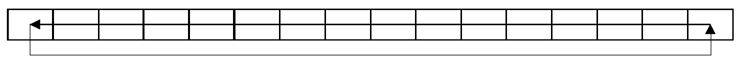
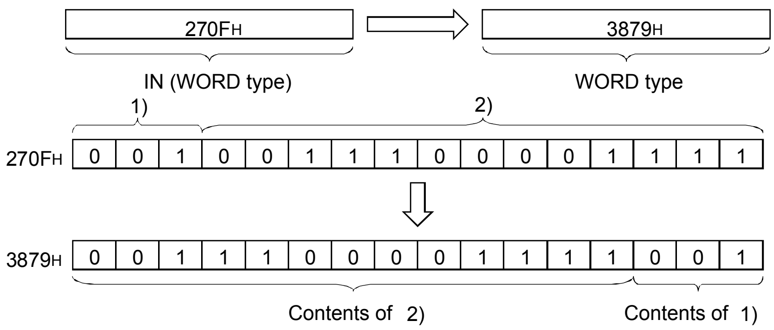
Function overview: ROL(_E) rotate the input value to the left by n bits, and then output the result.
ROR(_E) rotate the input value to the right by n bits, and then output the result.

Function/ FB classification name: Bit-string function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: stop)
	IN	Input variable	WORD DWORD	Input
	N	Input variable	INT	Shift bit number specification
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	WORD DWORD	Output

Function

Item	Contents
Operation processing	(1) ROL ROL_E Rotates (rotation) the WORD/DWORD type data to the left by n bits that is input from input variable IN, and outputs the result from output variable OUT by the same data type as that of input variable IN. The bit number n for rotation left is the input value of input variable N. WORD type, DWORD type  n bit rotation (Example) The following is an example in which the data type of input variable IN is WORD type and the input value of input variable N is 3. (Rotates 3 bits to left) 

Item	Contents

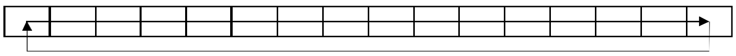
Operation processing

(2) ROR ROR_E

Rotates (rotation) the WORD/DWORD type data to the right by n bits that is input from input variable IN, and outputs the result from output variable OUT by the same data type as that of input variable IN.

The bit number n for rotation left is the input value of input variable N.

WORD type, DWORD type



n bit rotation

(Example) The following is an example in which the data type of input variable IN is WORD type and the input value of input variable N is 3. (Rotates to the right by 3 bits)

$\underbrace{\boxed{270F_H}}_{\text{IN (WORD type)}} \Rightarrow \underbrace{\boxed{E4E1_H}}_{\text{WORD type}}$

	2)													1)		
$270F_H$	0	0	1	0	0	1	1	1	0	0	0	0	1	1	1	1

⇩

$E4E1_H$	1	1	1	0	0	1	0	0	1	1	1	0	0	0	0	1
	Contents of 1)			Contents of 2)												

(3) The value input from the input variable IN is WORD/ DWORD type data.

(4) The value input from the input variable N (shift bit number specification) is INT type data, and its range is as follows.

(a) When the data type of input variable N is WORD type
The input value of variable N is within 0 to 15.
Applying the 4 low-order bits of the value input from the input variable N.
Example When input value is 6 : 6
When input value is 22 : 6

(b) When the data type of input variable IN is DWORD type
The input value of variable N is within 0 to 31.
Applying the 5 low-order bits of the value input from the input variable N.
Example When input value is 6 : 6
When input value is 22 : 22

Operation results

(1) Function without EN/ENO pins
Execute operation processing and output operation output value from OUT.

(2) Function with EN/ENO pins
The execution conditions and the operation results are as follows:

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE	Operation output value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

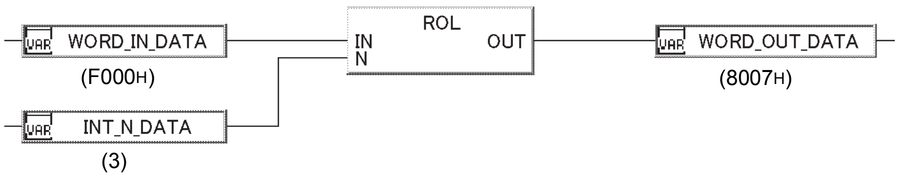
Error

There is no operation error caused by ROL(_E), ROR(_E).

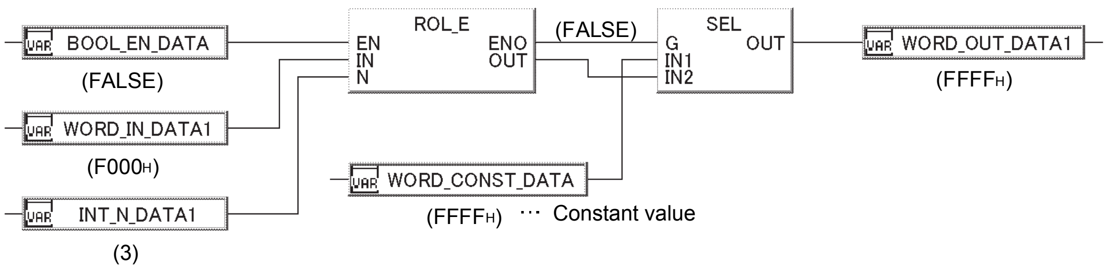
Program Example

(1) Following is the program example in which rotate left by n nits operation (rotation) (of the WORD/DWORD type data input from input variable IN) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

(a) Basic program example (ROL)

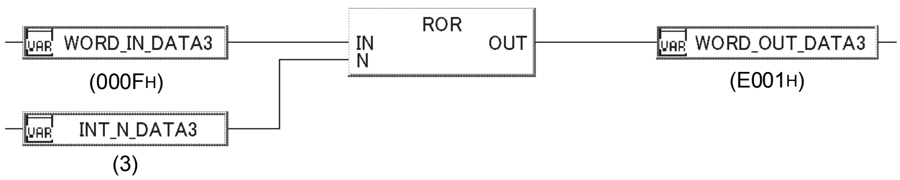


(b) This is a program example in which the output is constant value when the input variable EN is FALSE. (ROL_E)



(2) Following is the program example in which rotate right by n bits operation (rotation) (of the WORD/DWORD type data input from input variable IN) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

(a) Basic program example (ROR)



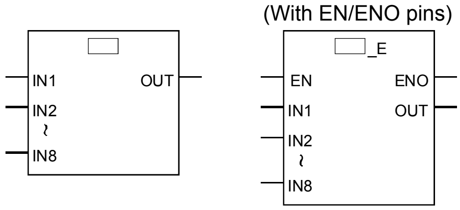
4.5 Logical Operation Function

4.5.1 AND, OR, XOR and NOT (AND(_E), OR(_E), XOR(_E), NOT(_E))

Function: AND, OR, XOR, AND _E, OR _E, XOR _E

FBD parts:

(With EN/ENO pins)



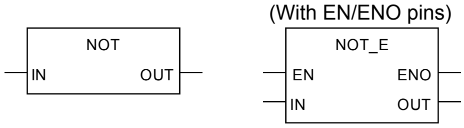
Indicates AND, OR, XOR.
Pin number of input variable IN: 2 to 8.

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	2 to 8

Function: NOT, NOT_E

FBD parts:

(With EN/ENO pins)



With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

Function overview: AND(_E) outputs AND of the input value. OR(_E) outputs the OR of the input value.
XOR(_E) outputs XOR of the input value. NOT(_E) outputs NOT of the input value.

Function/ FB classification name: Logical operation function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: stop)
	IN1 to IN8 (NOT(_E) is IN)	Input variable	BOOL WORD DWORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	BOOL WORD DWORD	Output

Function

Item	Contents
Operation processing	(1) AND AND_E (a) The processing performs AND operation on the BOOL/WORD/DWORD type data input from input variables IN1 to IN8 and outputs the result from the output variable OUT by the same data types as that of input variable IN. (Example) When the data type is WORD type IN1: 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 AND IN2: 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 ⇩ AND OUT: 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 (b) The pin number of input variable IN is within 2 to 8.

Item	Contents
Operation processing	(see below)

(2) OR OR_E

(a) The processing performs OR operation on the BOOL/WORD/DWORD type data input from input variables IN1 to IN8 and outputs the result from the output variable OUT by the same data types as that of input variable IN.

(Example) When the data type is WORD

IN1	1	1	0	0	1	1	1	1	0	0	0	0	1	1	1	1
OR																
IN2	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	0
⇩ OR																
OUT	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	1

(b) The pin number of input variable IN is within 2 to 8.

(3) XOR XOR_E

(a) The processing performs XOR operation on the BOOL/WORD/DWORD type data input from input variables IN1 to IN8 and outputs the result from the output variable OUT by the same data types as that of input variable IN.

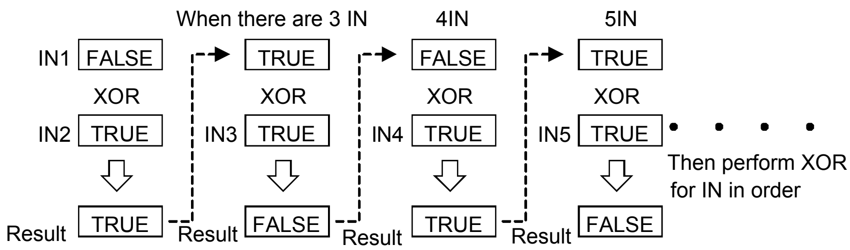
(Example) When the data type is WORD

IN1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
OR																
IN2	0	0	0	1	1	0	1	1	1	1	1	1	0	0	0	0
⇩ XOR																
OUT	1	0	1	1	0	0	0	1	0	1	0	1	1	0	1	0

(b) The pin number of input variable IN is within 2 to 8

(c) When there are 3 or more input variable IN, XOR operation should be executed on IN3 with the operation result of IN1 and IN2.
Besides, if IN4 exits, XOR operation shall be executed on IN4 with the operation result of IN3.
Then, perform XOR on IN5 and IN6 in sequence according to the allocated quantity of input variable IN.

(Example) When data type is BOOL



(4) NOT NOT_E

The processing performs NOT operation on the BOOL/WORD/DWORD type data input from input variables IN and outputs the result from the output variable OUT by the same data types as that of input variable IN.

(Example) When the data type is WORD

IN	0	1	1	0	1	0	1	1	0	0	0	0	1	1	1	1
NOT																
OUT	1	0	0	1	0	1	0	0	1	1	1	1	0	0	0	0

(5) The value input from the input variable IN is BOOL/ WORD/DWORD type data.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
Execute operation processing and output operation output value from OUT.
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

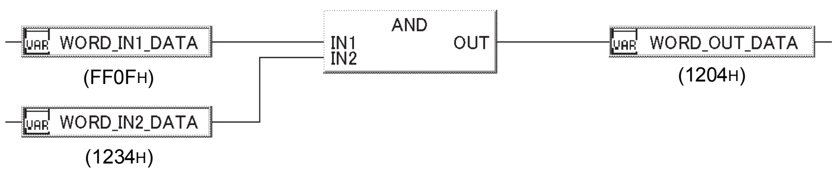
Error

There is no operation error caused by AND(_E), OR(_E), XOR(_E), NOT(_E).

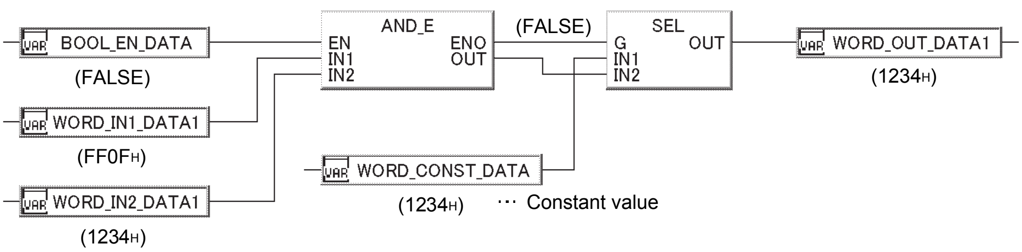
Program Example

(1) Following is the program example in which AND operation (of the BOOL/WORD/DWORD type data input from input variable IN1 to IN8) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

(a) Basic program example (AND)

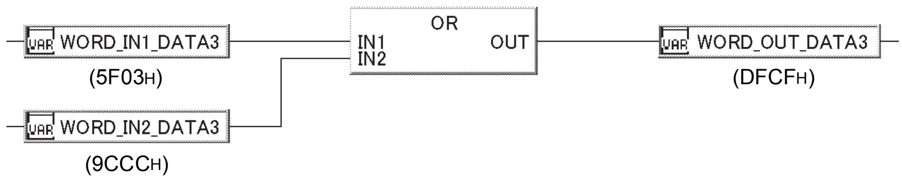


(b) This is a program example in which the output is constant value when the input variable EN is FALSE. (AND_E)



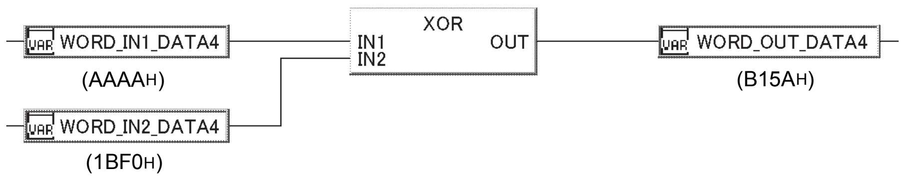
(2) Following is the program example in which OR operation (of the BOOL/WORD/DWORD type data input from input variable IN1 to IN8) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

(a) Basic program example (OR)



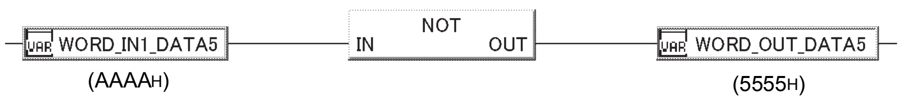
(3) Following is the program example in which XOR operation (of the BOOL/WORD/DWORD type data input from input variable IN1 to IN8) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

(a) Basic program example (XOR)



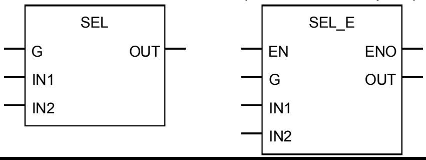
(4) Following is the program example in which NOT operation (of the BOOL/WORD/DWORD type data input from input variable IN) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

(a) Basic program example (NOT)



4.6 Selection Function

4.6.1 Input Value Selection (SEL(_E))

Function	FBD parts
SEL SEL_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

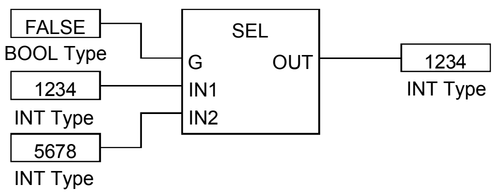
Function overview: Output the selected input value.

Function/ FB classification name: Selection function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Operation condition (TRUE: Execute FALSE: Stop)
	G	Input variable	BOOL	Output condition (TRUE: IN2 output FALSE: IN1 output)
	IN1 IN2	Input variable	BOOL INT DINT WORD DWORD REAL ADR_REAL STRING (255)	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	BOOL INT DINT WORD DWORD REAL ADR_REAL STRING (255)	Output

Function

Item	Contents
Operation processing	(1) According to the value input from the input variable G, output one of the output values input from input variable IN1 or IN2 from OUT in the same data type as input variable IN. When input variable G inputs FALSE, output the input value of input variable IN1 from output variable OUT. When input variable G inputs TRUE, output the input value of input variable IN2 from output variable OUT. (Example) When the data type of input variable IN1 and IN2 is INT 

Item	Contents
Operation processing	(2) The input value from input variable G is BOOL type data. (3) The values input from input variable IN1 and IN2 are BOOL/INT/DINT/WORD/DWORD/REAL/ADR_REAL/STRING type data.
Operation result	(1) Functions without EN/ENO pins Execute operation processing and output operation output value from OUT. (2) Functions with EN/ENO pins The execution conditions and the operation results are as follows: <table><tr><th>Execution condition</th><th colspan="2">Operation result</th></tr><tr><th>EN</th><th>ENO</th><th>OUT</th></tr><tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr><tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr></table>* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

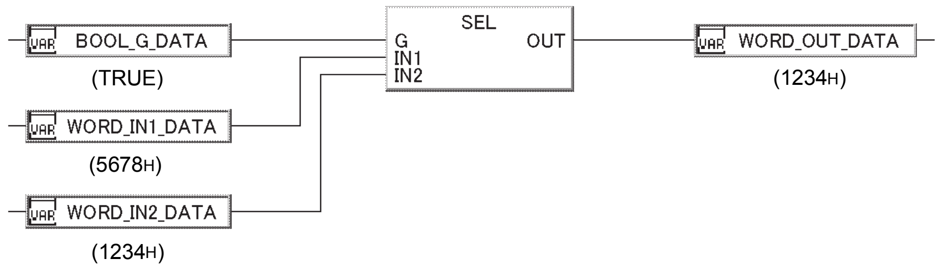
Error

There is no operation error caused by SEL(_E).

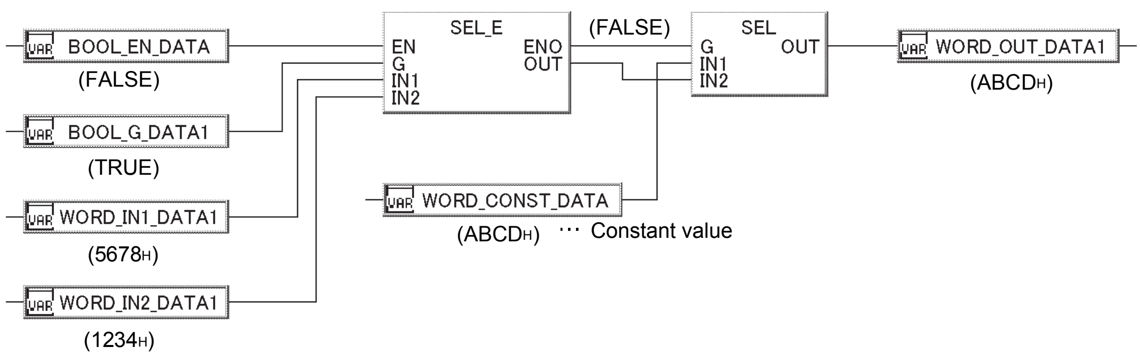
Program Example

Following is the program example in which one of the value from IN1 and IN2 which is input to input variable G , and the result is output from the output variable OUT in the same data type with input variable IN.

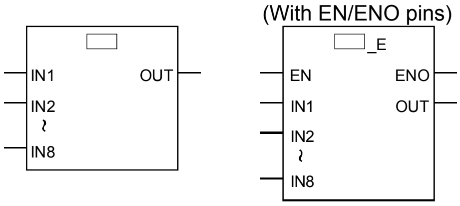
(1) Basic program example (SEL)



(2) This is a program example in which the output is constant value when the input variable EN is FALSE. (SEL_E)



4.6.2 Maximum/Minimum Value Selection (MAX(_E), MIN(_E))

Function	FBD parts
MAX MIN MAX_E MIN_E	(With EN/ENO pins)  Indicates MAX, MIN Pin number of input variable IN can be changed in the range of 2 to 8.

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	2 to 8

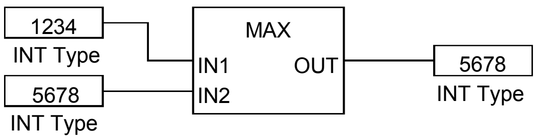
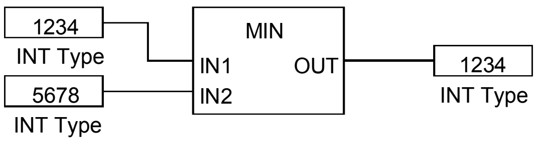
Function overview: MAX(_E) outputs the maximum value of the input value.
MIN(_E) outputs the minimum value of the input value.

Function/ FB classification name: Selection function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: stop)
	IN1 to IN8	Input variable	INT	Input
			DINT	
			REAL	
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output
			DINT	
			REAL	

Function

Item	Contents
Operation processing	(1) MAX MAX_E Output the maximum value of type INT/DINT/REAL data which are input from the input variable IN1 to IN8 from output variable OUT in the same data type as IN. (Example) When the data type is INT  (2) MIN MIN_E Output the minimum value of type INT/DINT/REAL data which are input from the input variable IN1 to IN8 from output variable OUT in the same data type as IN. (Example) When the data type is INT  (3) The value input from the input variable IN1 to IN8 is INT/DINT/REAL type data. (4) The pin number of variable IN is within the range 2 to 8.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>(1) Functions without EN/ENO pins
Execute operation processing and output operation output value from OUT.

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

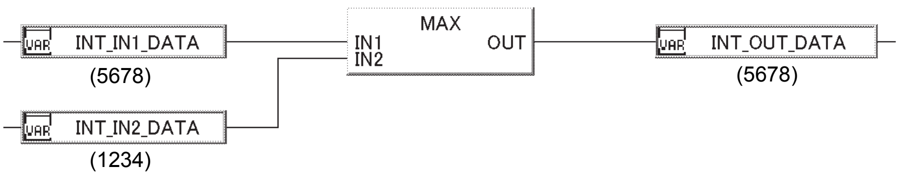
Error

There is no operation error caused by MAX(_E), MIN(_E).

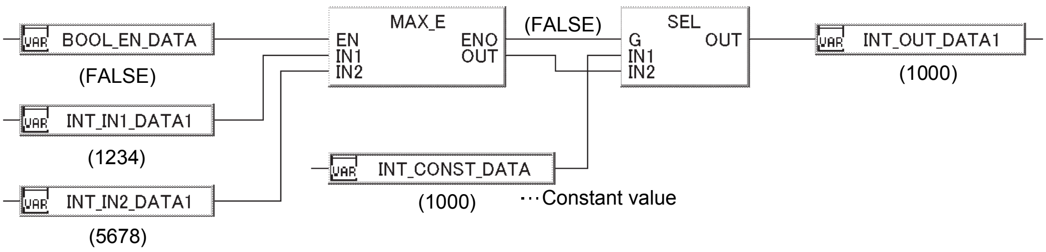
Program Example

(1) Following is the program example in which maximum value (of the INT/DINT/REAL type data input from input variable IN1 to IN8) is output from the output variable OUT in the same data type with input variable IN.

(a) Basic program example (MAX)

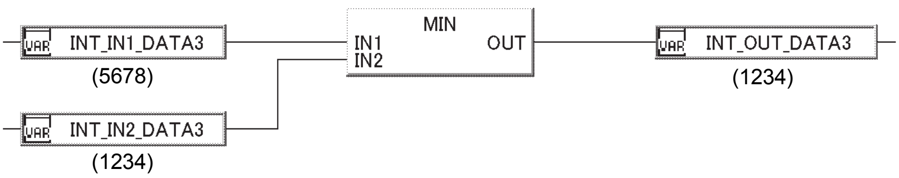


(b) This is a program example in which the output is constant value when the input variable EN is FALSE. (MAX_E)

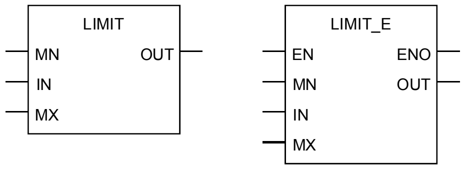


(2) Following is the program example in which minimum value (of the INT/DINT/REAL type data input from input variable IN1 to IN8) is output from the output variable OUT in the same data type with input variable IN.

(a) Basic program example (MIN)



4.6.3 High/Low Limit Control (LIMIT(_E))

Function	FBD parts
LIMIT LIMIT_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

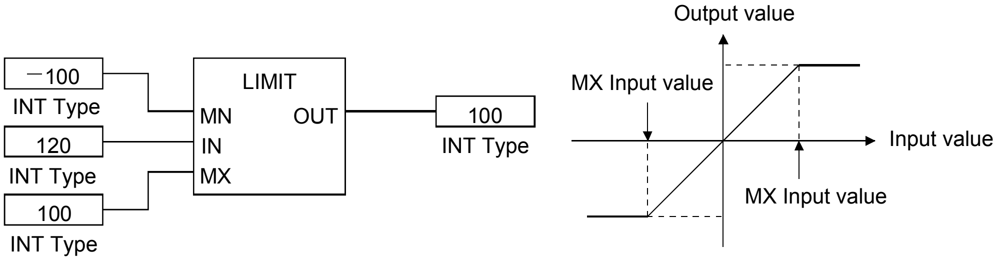
Function overview: Output the input value through the high/low limit control.

Function/ FB classification name: Selection function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: stop)
	MN	Input variable	INT DINT REAL	Low limit value (minimum output limit value)
	IN	Input variable	INT DINT REAL	The input value controlled through high/low limit control
	MX	Input variable	INT DINT REAL	High limit value (maximum output limit value)
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT DINT REAL	Output

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(1) Output to output variable OUT in the same data type as the input variable according to the INT/DINT/REAL data input from the input variable MN, IN and MX.
(a) When input value of IN > input value of MX, output the input value of input variable MX from output variable OUT.
(b) When input value of IN < input value of MN, output the input value of input variable MN from output variable OUT.
(c) When input value of MN ≦ input value of IN ≦ input value of MX, output the input value of input variable IN from output variable OUT.
(Example) When the data type is INT


(2) The values input from the input variable MN, IN, MX are INT/DINT/REAL type data.
(In the case of input value of MN < input value of MX)
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occur</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the data type is INT/DINT, and input value of MN > input value of MX. (Error code: Refer to Appendix 2)

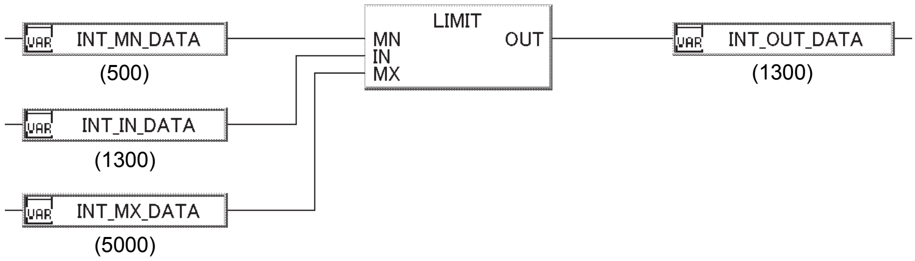
POINT

When data type is REAL, input value of MN > input value of MX, it is not an operation error.
However, the operation result will be undefined value.
Furthermore, for functions with EN/ENO, ENO will be FALSE.

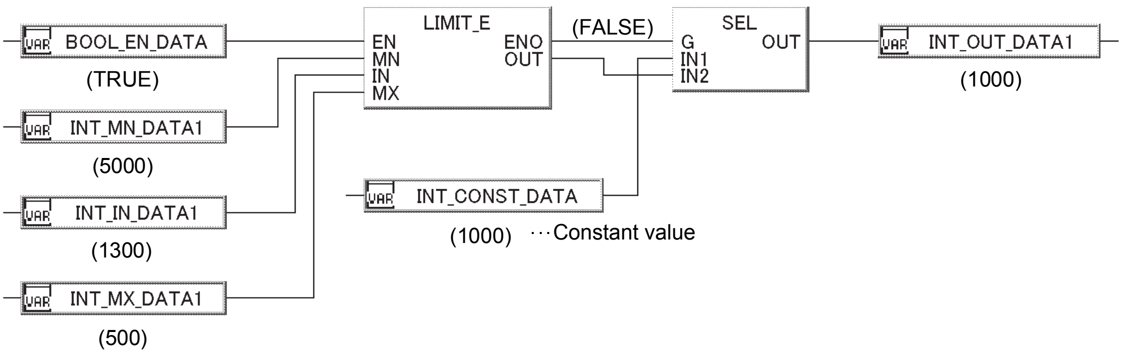
Program Example

Following is the program example in which outputs the value input from input variable IN1 and IN2 to output variable OUT in the same data type with input variable IN according to the input INT/DINT/REAL type data of input variable MN, IN and MX.

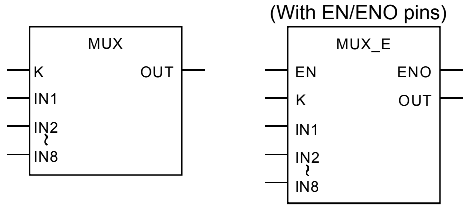
(1) Basic program example (LIMIT)



(2) This is a program example in which the output is constant value when the input variable EN is FALSE. (LIMIT_E)
(Example) When operation errors occur



4.6.4 Multiplexer (MUX(_E))

Function	FBD parts
MUX MUX_E	(With EN/ENO pins)  Pin number of input variable IN can be changed in the range of 2 to 8.

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	2 to 8

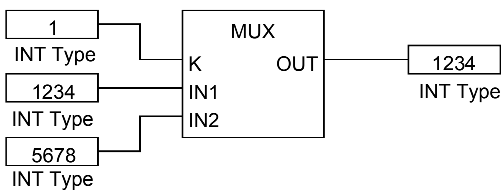
Function overview: Output one of the multiple input values.

Function/ FB classification name: Selection function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: stop)
	K	Input variable	INT	Select the output value
	IN1 to IN8	Input variable	BOOL	Input
			INT	
			DINT	
			WORD	
			DWORD	
			REAL	
			ADR_REAL	
			STRING (255)	
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	BOOL	Output
			INT	
			DINT	
			WORD	
			DWORD	
			REAL	
			ADR_REAL	
			STRING (255)	

Function

Item	Contents
Operation processing	(1) Output one of the values input from input variable IN1 to IN8 from output variable OUT in the same data type as variable IN, according to the value input from the input variable K. When input value of K is 1, output the input value of IN1 from output variable OUT. When input value of K is n, output the input value of INn from output variable OUT. (Example) When the data type is INT. 

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>(2) When the number of input values of the input variable K is out of range of the pins of variable IN, output undefined value from output variable OUT.
(3) The input value of input variable K is of type INT and within 1 to 8. (But it should be within the pin number range of input variable IN)
(4) The input value of input variable IN is of BOOL/INT/DINT/WORD/DWORD/REAL/ADR_REAL/STRING type.
(5) The pin number of input variable IN is within the range of 2 to 8.</td></tr>
<tr><td>Operation results</td><td>(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occur</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs)) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

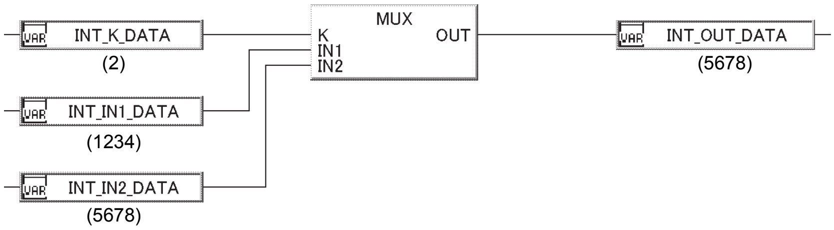
Error

There is no operation error caused by MUX(_E).

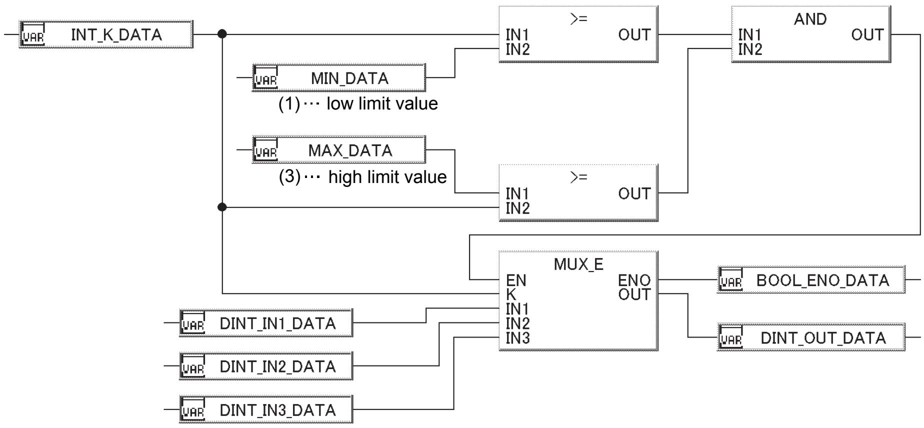
Program Example

Following is the program example in which one of the input value from input variable IN1 to IN8 (input from input variable K) is input, and the result is output from the output variable OUT in the same data type with input variable IN.

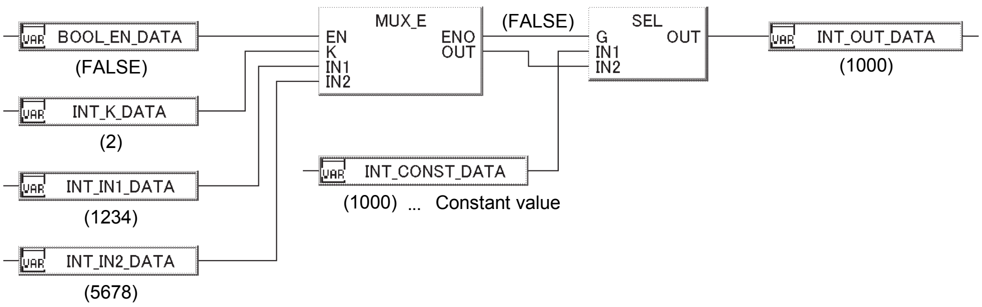
(1) Basic program example (MUX)



(2) This is a program example that checks the input value of input variable K in advance. (MUX_E)

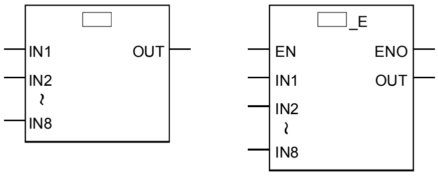
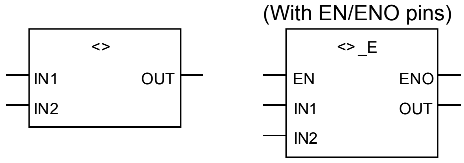


(3) This is a program example in which the output is constant value when the input variable EN is FALSE. (MUX_E)



4.7 Comparison Function

4.7.1 Comparison (>(_E), > =(_E), =(_E), < =(_E), <(_E), <>(_E))

Function	FBD parts
> >_E >= >=_E = =_E <= <=_E < <_E	(With EN/ENO pins)  Indicates >,>=,=,<=,<. Pin number of input variable IN can be changed in the range of 2 to 8.
<> <>_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	2 to 8

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

Function overview: Output comparison results of the input data.

Function/ FB classification name: Comparison function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: stop)
	IN1 to IN8	Input variable	INT	Input
			DINT	
			REAL	
			WORD(*1)	
			DWORD(*1)	
			BOOL (*2)	
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	BOOL	Output (TRUE: True value FALSE: False value)

*1 In the case of >(_E), >=(_E), <(_E), and <=(_E), when using for WORD type/DWORD type input, the number of steps involved in compilation and generation of ladder program increase. If the input values are in the range of 0_H to $7FFF_H$, 0_H to $7FFFFFFF_H$, it is recommended to convert data type using WORD_TO_INT/DWORD_TO_INT functions and to compare the data type with INT type/DINT type.

*2 Input of BOOL type is possible only for =(_E), <>(_E)

Function

Item	Contents
Operation processing	(1) This process performs comparison operation for the values input from the input variable IN, and output the results from output variable OUT with BOOL type. (a) > >_E Compare [IN1> IN2]&[IN2> IN3]&…&[IN $_{(n-1)}$ > IN $_{(n)}$] ● When all IN $_{(n-1)}$ > IN $_{(n)}$, output TRUE. ● When any IN $_{(n-1)}$ ≦ IN $_{(n)}$, output FALSE. (b) >= >=_E Compare [IN1≧ IN2]&[IN2≧IN3]&…&[IN $_{(n-1)}$ ≧ IN $_{(n)}$] ● When all IN $_{(n-1)}$ ≧ IN $_{(n)}$, output TRUE. ● When any IN $_{(n-1)}$ < IN $_{(n)}$, output FALSE.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(c) = =_E Compare [IN1= IN2]&[IN2=IN3]&…&[IN (n-1) = IN (n)]
● When all IN (n-1) = IN (n), output TRUE.
● When any IN (n-1) ≠IN (n), output FALSE.

(d) <= <=_E Compare [IN1 ≦ IN2]&[IN2 ≦ IN3]&…&[IN (n-1) ≦ IN (n)]
● When all IN (n-1) ≦ IN (n), output TRUE.
● When any IN (n-1) > IN (n), output FALSE.

(e) < <_E Compare [IN1< IN2]&[IN2<IN3]&…&[IN (n-1) < IN (n)]
● When all IN (n-1) < IN (n), output TRUE.
● When any IN (n-1) ≧ IN (n), output FALSE.

(f) <> <>_E Compare [IN1≠ IN2]
● When IN1≠ IN2, output TRUE.
● When IN1= IN2, output FALSE.
(2) The value input from input variable IN is INT/DINT/REAL type data.

(3) The pin number of input variable IN is within 2 to 8. (But the pins of the input variable IN of <> (_E) are fixed as IN1, IN2.)
</td></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation result</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occur</td><td>Undefined value</td></tr>
</table>

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs)) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

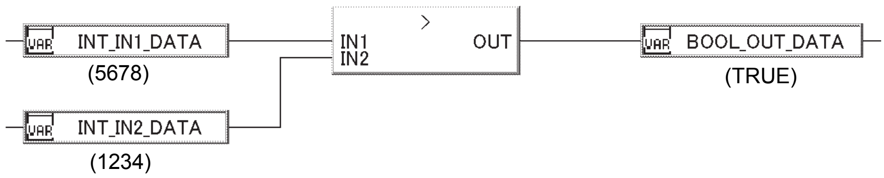
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the input value is -0. (Error code: Refer to Appendix 2)

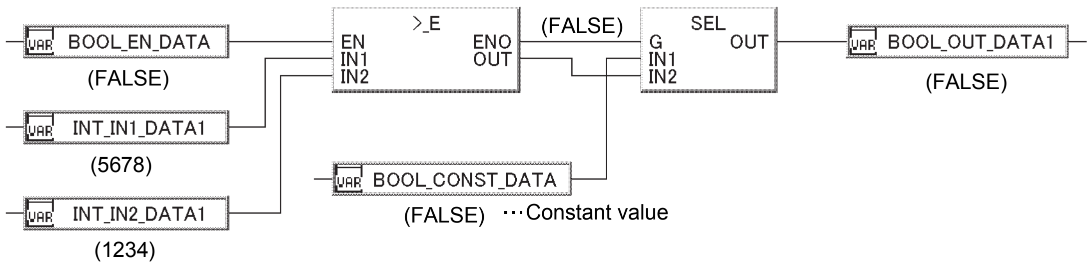
Program Example

Following is the program example in which comparison operation (input from input variable IN) is executed, and the result is output from the output variable OUT in the same data type with input variable IN.

(1) Basic program example (>)



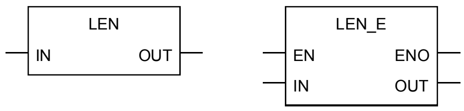
(2) This is a program example in which the output is constant value when the input variable EN is FALSE or operation errors occur. (>_E)
(Example) When the input variable EN is FALSE



4.8 Character String Function

4.8.1 String Length (LEN(_E))

Function	FBD parts
LEN LEN_E	(With EN/ENO pins)



With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Detect and output the input string length

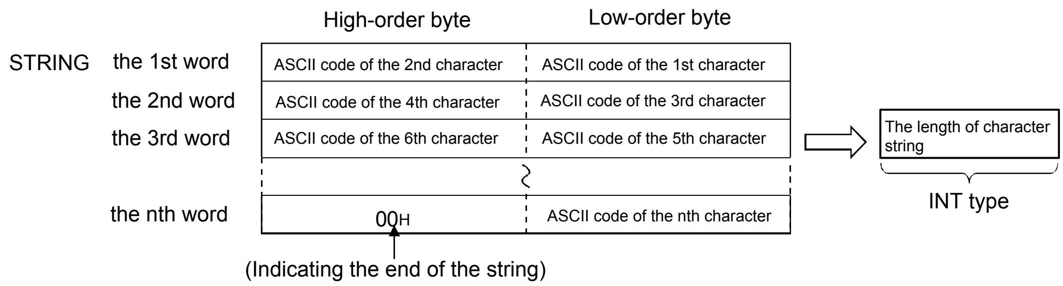
Function/FB classification name: Character string function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (255)	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output

Function

Item	Contents
Operation processing	(1) Detect the string length input from the input variable IN, and output it from output variable OUT. (2) The value input from input variable IN is STRING type within the range of 0 to 255 bytes.



Operation results

(1) Functions without EN/ENO pins
The operation results are as follows:

Operation result	OUT
No operation error	Operation output value
Operation error occurs	Undefined value

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:

Execution condition	Operation result	
EN	ENO	OUT
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error occurs) (*)	Undefined value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Error

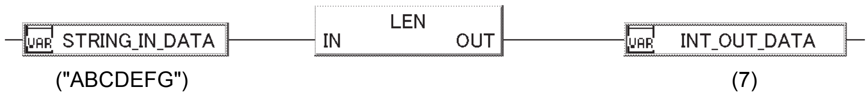
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- "00H" is not included in the string input from input variable IN. (Error code: Refer to Appendix 2)

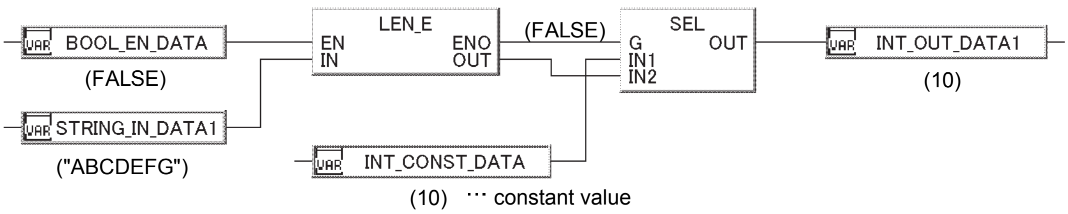
Program Examples

The following are programs that will detect the string length input from the input variable IN and output from output variable OUT.

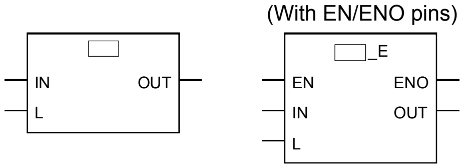
(1) Basic program example (LEN)



(2) This is a program example in which the output is constant value when the input variable EN is FALSE or operation error occurs. (LEN_E)
(Example) When input variable EN is FALSE



4.8.2 Leftmost/Rightmost Characters (LEFT(_E), RIGHT(_E))

Function	FBD parts
LEFT RIGHT LEFT_E RIGHT_E	(With EN/ENO pins)  □ Indicate LEFT, RIGHT

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: LEFT(_E) Output specified number of characters from leftmosted
RIGHT(_E) Output specified number of characters from rightmosted

Function/FB classification name: Character string function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Content
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (255)	Input
	L	Input variable	INT	The specification of character number extraction
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	STRING (255)	Output

Function

Item	Contents
Operation processing	(1) LEFT LEFT_E Specified character number data (input from the input variable IN) is extract and output from the output variable OUT. The characters extract is specified by the value input from input variable L. (Example) When the value input from input variable L is 7 "ABCDEF12345" ⇒ "ABCDEF1"

Input:

	High-order byte	low-order byte
the 1st word	42H(B)	41H(A)
the 2nd word	44H(D)	43H(C)
the 3rd word	46H(F)	45H(E)
the 4th word	32H(2)	31H(1)
the 5th word	34H(4)	33H(3)
the 6th word	00H	35H(5)

7 characters number of extracting (L)

Output:

High-order byte	low-order byte	
42H(B)	41H(A)	the 1st word
44H(D)	43H(C)	the 2nd word
46H(F)	45H(E)	the 3rd word
00H	31H(1)	the 4th word

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(2) RIGHT RIGHT_E
Specified character number data (input from the input variable IN) is extract and output from the output variable OUT.
The characters extract is specified by the value input from input variable L.
(Example) When the value input from input variable L is 5
"ABCDEF12345" ⇨ "12345"
<table>
<tr><th></th><th>High-order bits</th><th>low-order bits</th></tr>
<tr><td>the 1st word</td><td>42H(B)</td><td>41H(A)</td></tr>
<tr><td>the 2nd word</td><td>44H(D)</td><td>43H(C)</td></tr>
<tr><td>the 3rd word</td><td>46H(F)</td><td>45H(E)</td></tr>
<tr><td>the 4th word</td><td>32H(2)</td><td>31H(1)</td></tr>
<tr><td>the 5th word</td><td>34H(4)</td><td>33H(3)</td></tr>
<tr><td>the 6th word</td><td>00H</td><td>35H(5)</td></tr>
</table>
Character number of extracting (L) 5 characters
<table>
<tr><th>High-order bits</th><th>low-order bits</th><th></th></tr>
<tr><td>32H(2)</td><td>31H(1)</td><td>the 1st word</td></tr>
<tr><td>34H(4)</td><td>33H(3)</td><td>the 2nd word</td></tr>
<tr><td>00H</td><td>35H(5)</td><td>the 3rd word</td></tr>
</table>
(3) The values input from input variable IN is STRING type data within the range of 0 to 255 bytes.
(4) The values input from input variable L is INT type within the range of 0 to 255.
(Pay attention that it cannot exceed the character number of the character string input from the input variable IN.)
</td></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation results</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

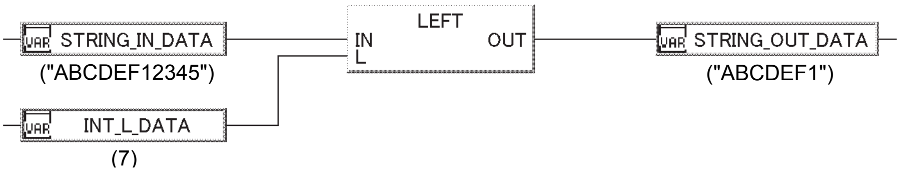
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When "00H" is not included in the string input from input variable IN. (Error code: Refer to Appendix 2)
- The value input from the input variable L is beyond the range of character number of the characters input from the input variable IN. (Error code: Refer to Appendix 2)

Program Examples

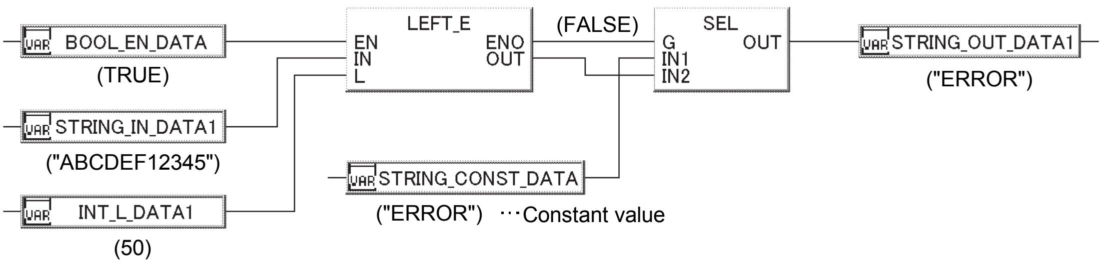
(1) Specified character number data (input from the input variable IN) is extract and output from the output variable OUT.

(a) Basic program example (LEFT)



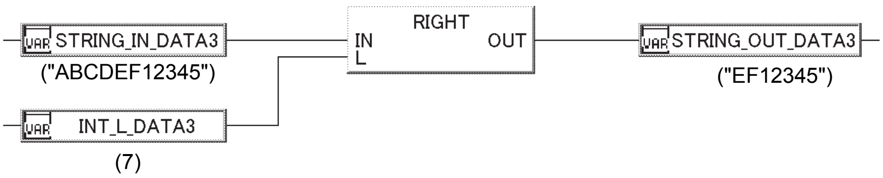
(b) This is a program example in which the output is constant value when the input variable EN is FALSE or operation error occurs. (LEFT_E)

(Example) When operation errors occur

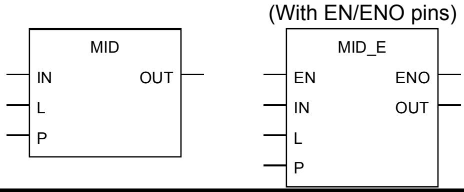


(2) Specified character number data (input from the input variable IN) is extract and output from the output variable OUT.

(a) Basic program example (RIGHT)



4.8.3 Middle Characters (MID(_E))

Function	FBD parts
MID MID_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

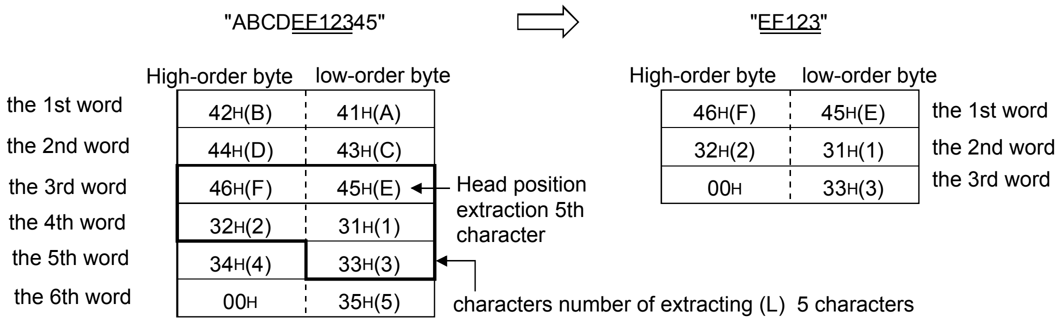
Function overview: Output the specified number of characters beginning from any position of the input character string.

Function/FB classification number: Character string function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Content
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (255)	Input
	L	Input variable	INT	The specification of character number extraction
	P	Input variable	INT	The specification of head position extraction
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	STRING (255)	Output

Function

Item	Contents
Operation processing	(1) Specified character number data (input from the input variable IN) is extracted and output from the output variable OUT. The number of characters extracted is specified by the value input from input variable L. The head position of extracted character string is specified by the input value to the input variable P. (Example) When the value input from input variables L and P are both 5 

(2) The input value from input variable IN is STRING type data within the range of 0 to 255 bytes.

(3) The input value from input variable L is of INT type data within the range of 0 to 255.
(Pay attention that it cannot exceed the character number of the character string input from the input variable IN.)

(4) The input value to input variable P is of INT type within the range of 1 to 255,
(Provided that it does not exceed the character number of the characters input from the input variable IN.)

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation results</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

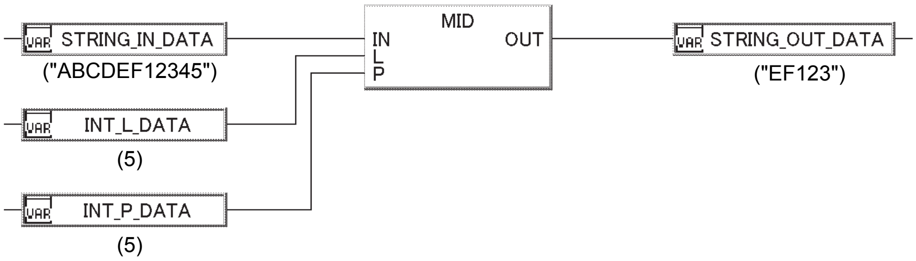
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- "00H"' is not included in the string input from input variable IN. (Error code: Refer to Appendix 2)
- The input value from the input variable L is beyond the range of character number of the characters input from the input variable IN. (Error code: Refer to Appendix 2)

Program Examples

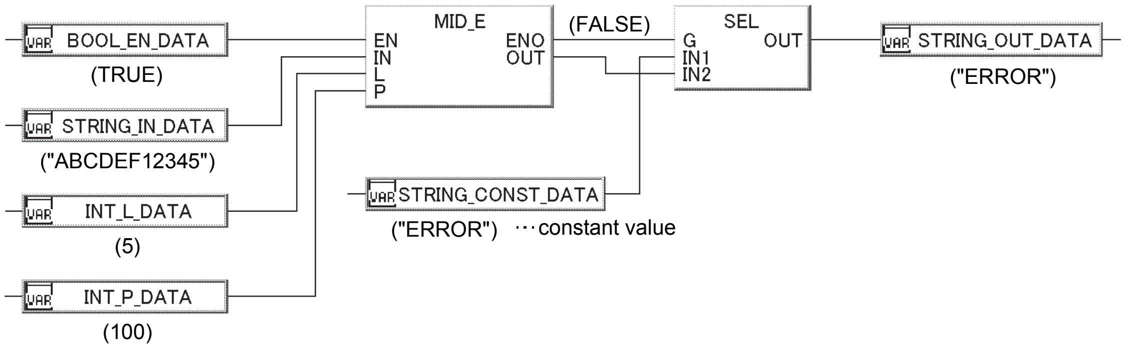
Following are the programs that will extract specified character number data (input from the input variable IN) and output from the output variable OUT.

(1) Basic program example (MID)

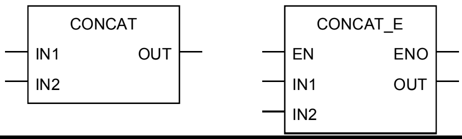


(2) This is a program example in which the output is constant value when the input variable EN is FALSE or operation error occurs. (MID_E)

(Example) When operation errors occur



4.8.4 Concatenation (CONCAT(_E))

Function	FBD parts
CONCAT CONCAT_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Concatenate two characters and output the combined characters

Function/FB classification: Character string function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Content
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN1 IN2	Input variable	STRING (255)	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	STRING (255)	Output

Function

Item	Contents
Operation processing	(1) The character string input from input variable IN2 is concatenated to the end of those input from input variable IN1. Then the concatenated string will be output from the output variable OUT. While concatenating two characters, "00H" indicating the end of the characters input to IN1 is ignored, the second character IN2 is closely concatenated. If the concatenated character strings have over 255 bytes, maximum 255 bytes will be output. (see tables below) (2) The value input from input variable IN1, IN2 are STRING type data within the range of 0 to 255 bytes.

"ABCDE" + "123456" ⇒ "ABCDE123456"

	High-order byte	low-order byte
the 1st word	42H(B)	41H(A)
the 2nd word	44H(D)	43H(C)
the 3rd word	00H	45H(E)

	High-order byte	low-order byte
the 1st word	32H(2)	31H(1)
the 2nd word	34H(4)	33H(3)
the 3rd word	36H(6)	35H(5)
the 4th word	00H	

	High-order byte	low-order byte
the 1st word	42H(B)	41H(A)
the 2nd word	44H(D)	43H(C)
the 3rd word	31H(1)	45H(E)
the 4th word	33H(3)	32H(2)
the 5th word	35H(5)	34H(4)
the 6th word	00H	36H(6)

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation results</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

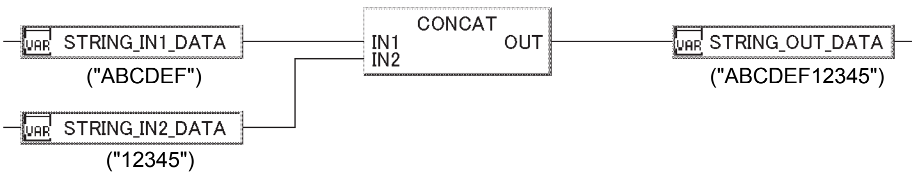
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- "00H" is not included in input value of input variable IN1, IN2. (Error code: Refer to Appendix 2)

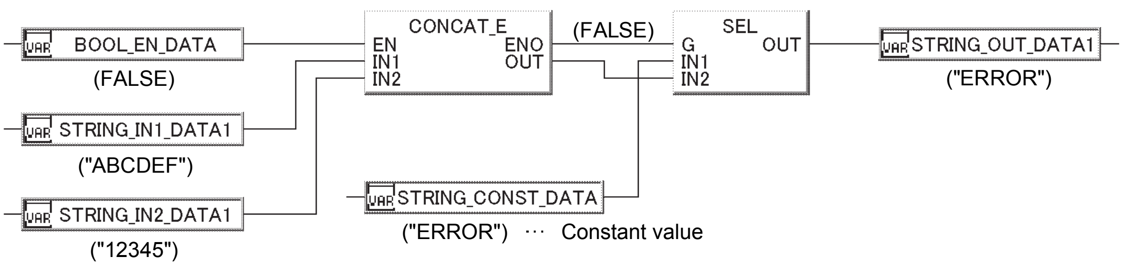
Program Example

Following are the programs that will output character string from input variable IN2 to output variable OUT concatenated it to the end of characters input from IN1.

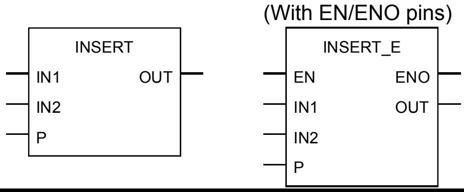
(1) Basic program example (CONCAT)



(2) This is the program example in which the output is constant value when the input variable EN is FALSE, or operation error occurs. (CONCAT _E)
(Example) When input variable EN is FALSE



4.8.5 Inserting Characters (INSERT(_E))

Function	FBD parts
INSERT INSERT_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

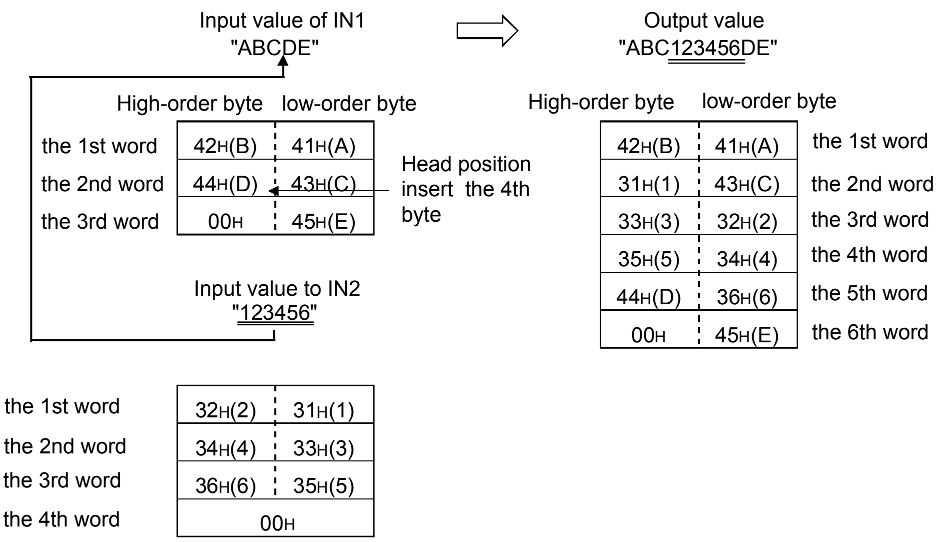
Function overview: Insert characters into character strings and output the finished one.

Function/FB classification name: Character string function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Content
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN1 IN2	Input variable	STRING (255)	Input
	P	Input variable	INT	The specification of head position insert
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	STRING (255)	Output

Function

Item	Contents
Operation processing	(1) Insert the character string, which is input to input variable IN2, to the Pth character count from the first character (the inserted head position) of the character string, which is input to input variable IN1. Then the inserted string will be output from the output variable OUT. After inserting characters into IN1's characters, "00H" indicating the end of the characters input to IN1 is ignored. If the inserted character strings have over 255 bytes, maximum 255 bytes will be output. (Example) When the input value to input variable P is 4  (2) The values input from input variable IN1, IN2 is STRING type data within the range of 0 to 255. (3) The values input from input variable P is INT type within the range of 1 to 255. (Note that it cannot exceed the character number of the character string input from input variable IN1.)

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation results</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

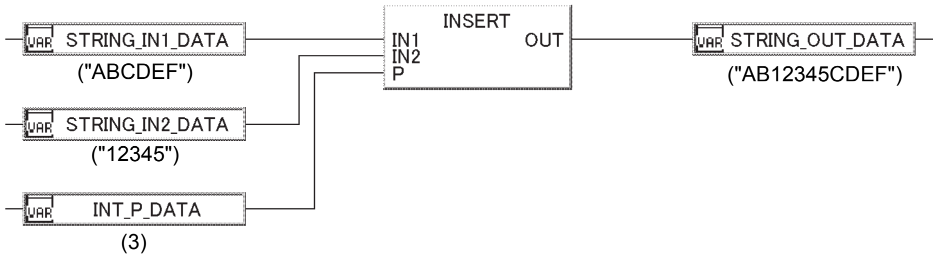
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- "00H" is not included in the input value from the input variable IN1, IN2. (Error code: Refer to Appendix 2)
- The input value to input variable P exceeds the character number of the string input to IN1 +1.
  (Error code: Refer to Appendix 2)

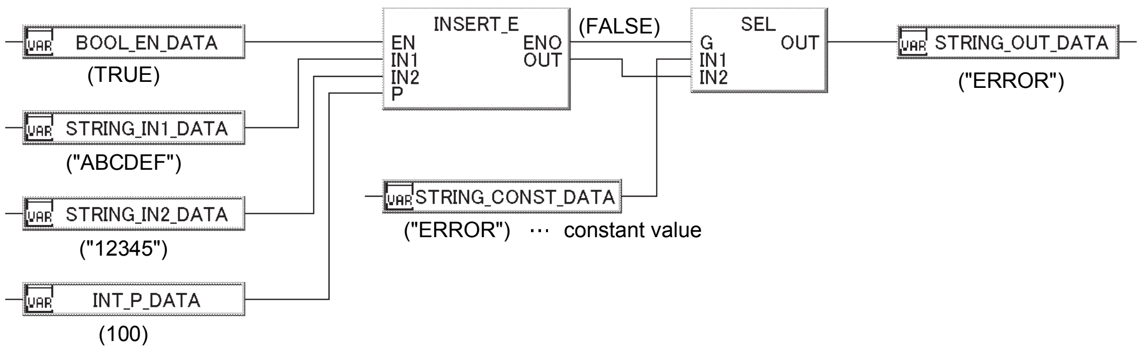
Program Example

The followings are the programs that will insert the character string, which is input to input variable IN2, to the Pth character count from the first character (the inserted head position) of the character string, which is input to input variable IN1, and output from the output variable OUT.

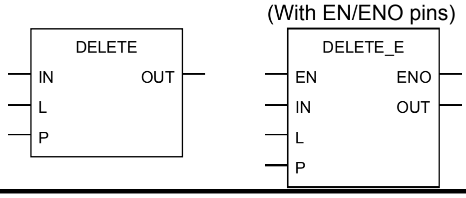
(1) Basic program example (INSERT)



(2) This is the program example in which the output is constant value when the input variable EN is FALSE, or operation error occurs. (INSERT _E)
(Example) When operation errors occur



4.8.6 Deleting Substring (DELETE(_E))

Function	FBD parts
DELETE DELETE_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

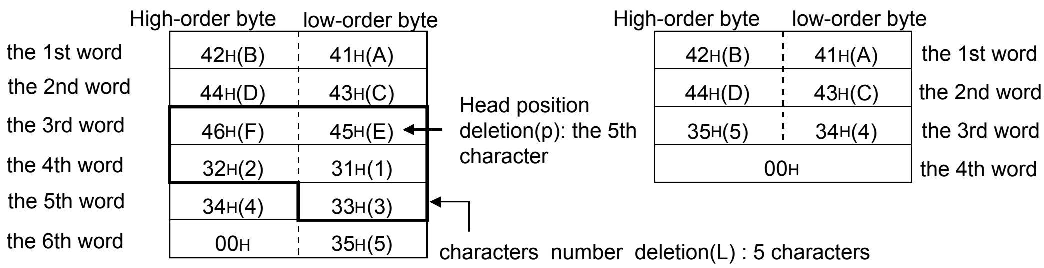
Function overview: Delete substring within any range and output the result.

Function/FB classification name: Character string function

Input and output pins

Pin	Variable name	Variable type	Data type	Content
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (255)	Input
	L	Input variable	INT	The specification of character number of deletion
	P	Input variable	INT	The specification of head position deletion
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	STRING (255)	Output

Function

Item	Contents
Operation processing	(1) Specified character number data (input from the input variable IN) is deleted and output from the output variable OUT. The deleted characters are specified by the value input from input variable L. The head position of character string to be deleted is specified by the input value to the input variable P. (Example) When the input value of the input variable L, P is 5 "ABCDEF12345" ⇨ "ABCD45"  (2) The values input from input variable IN are STRING type data within the range of 0 to 255. (3) The values input from input variable L are INT type within the range of 0 to 255. (Pay attention that it cannot exceed the character number of the character string input from input variable) (4) The values input from input variable P are INT type within the range of 1 to 255. (Pay attention that it cannot exceed the character number of the character string input from input variable)

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation results</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

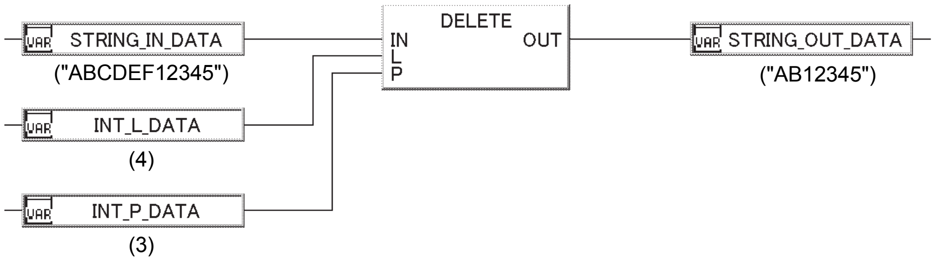
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- "00_H" is not included in the input value from the input variable IN1, IN2. (Error code: Refer to Appendix 2)
- The input value to input variable P exceeds the character number of the string input to IN1 +1.
  (Error code: Refer to Appendix 2)

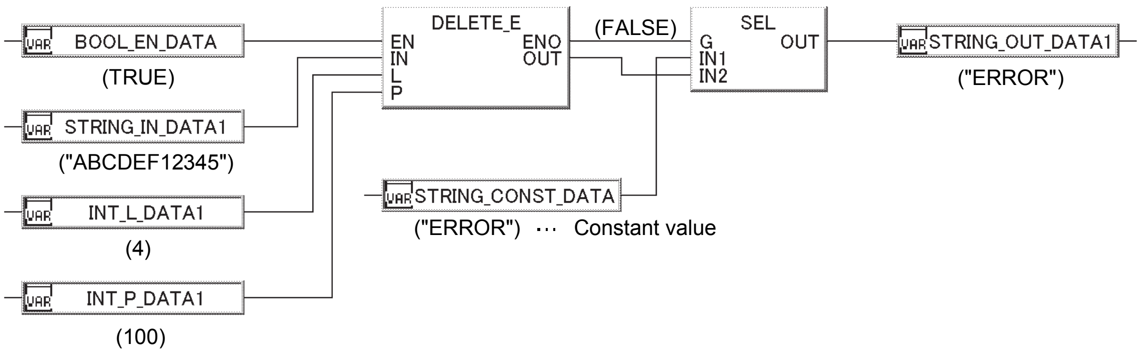
Program Example

The following are the programs that will delete the string length input from the input variable IN and output from output variable OUT.

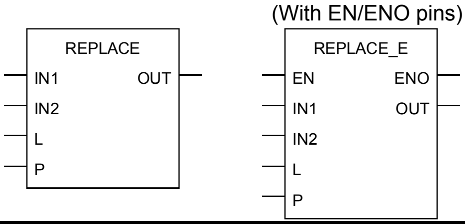
(1) Basic program example (DELETE)



(2) This is the program example in which the output is constant value when the input variable EN is FALSE, or operation error occurs. (DELETE _E)
(Example) When operation errors occur



4.8.7 Replacing Characters (REPLACE(_E))

Function	FBD parts
REPLACE REPLACE_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

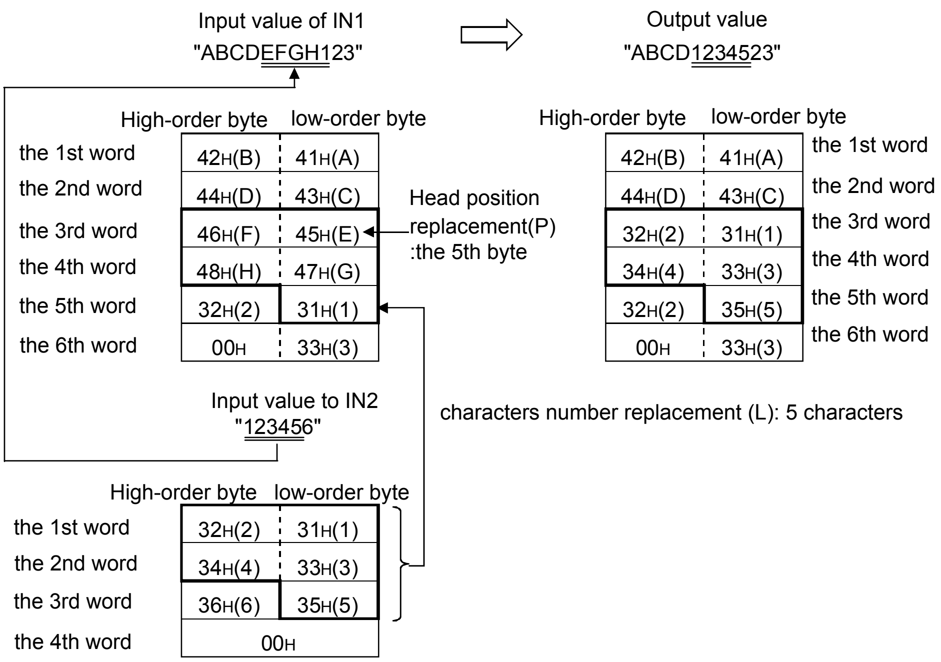
Function overview: Replace characters within any range and output the result.

Function/FB classification name: Character string function

Input and output pins

Pin	Variable name	Variable type	Data type	Content
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN1 IN2	Input variable	STRING (255)	Input
	L	Input variable	INT	The specification of character number replacement
	P	Input variable	INT	The specification of head position replacement
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	STRING (255)	Output

Function

Item	Contents
Operation processing	(1) After the character data with specified number from any position of the characters input from input variable IN1are replaced with the characters input from input variable IN2, they are output from output variable OUT. The character number to be replaced is decided by input value of input variable L. Replacement head position is decided by the input value to input variable P. (Example) When the value input from input variable L and P are both 5 

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>(2) The values input from input variable IN is STRING type data within the range of 0 to 255.

(3) The values input from input variable L is INT type within the range of 0 to 255.
(Pay attention that it cannot exceed the character number of the character string input from IN1)

(4) The values input from input variable P is INT type within the range of 1 to 255.
(Pay attention that it cannot exceed the character number of the character string input from IN1)</td></tr>
<tr><td>Operation results</td><td>(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation results</th><th>OUT</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occurs</td><td>Undefined value</td></tr>
</table>

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occurs) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

Error

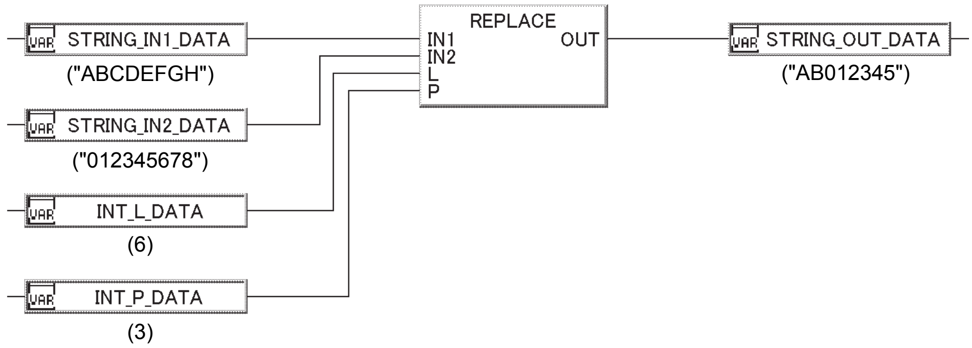
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- "00H" is not included in the input value from input variable IN1, IN2. (Error code: Refer to Appendix 2)
- The input value of input variable P exceeds the character number range of the string input from input variable IN1. (Error code: Refer to Appendix 2)

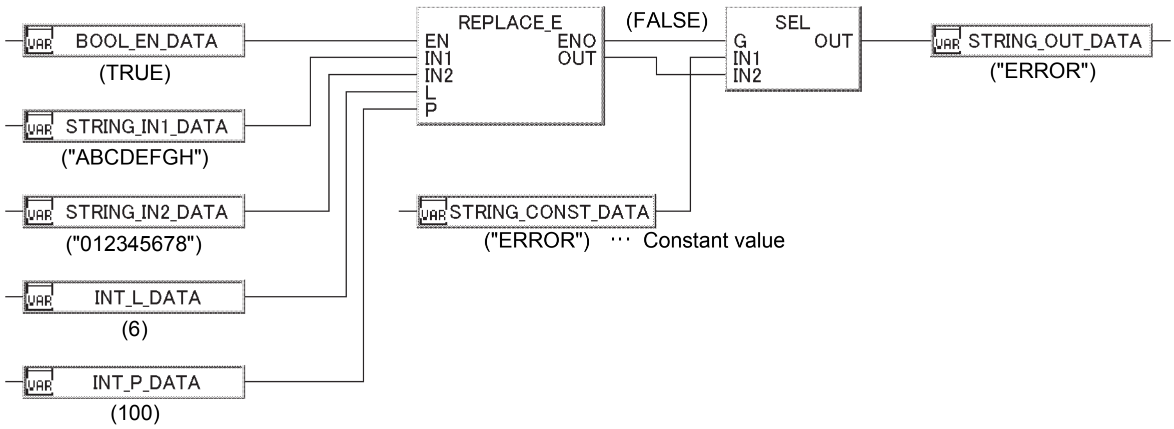
Program Example

The following are the programs that will delete the specified number of character data input from IN and output from output variable OUT.

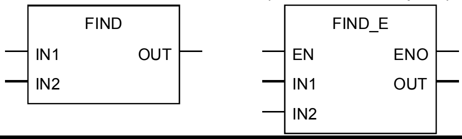
(1) Basic program example (REPLACE)



(2) This is the program example in which the output is constant value when the input variable EN is FALSE, or operation error occurs. (REPLACE _E)
(Example) When operation errors occur



4.8.8 Finding Characters (FIND(_E))

Function	FBD parts
FIND FIND_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

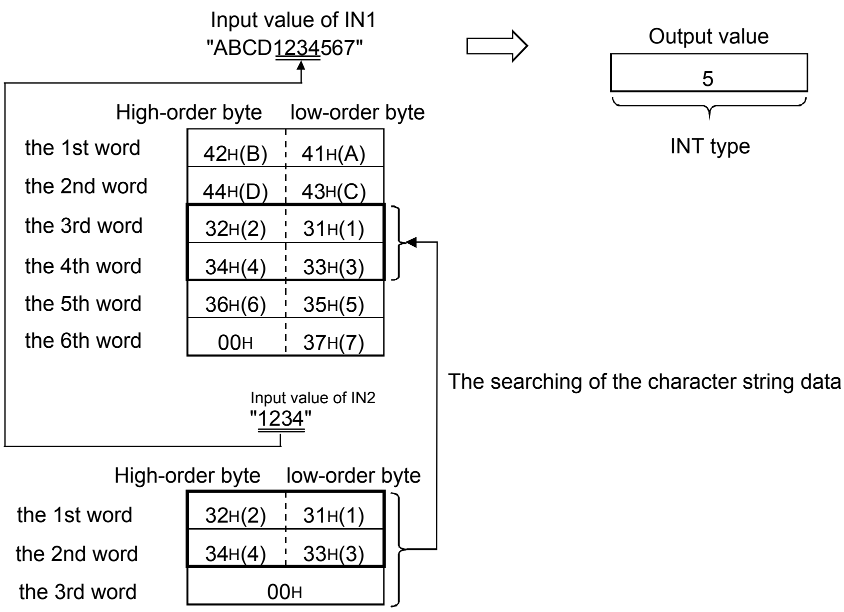
Function overview: Search the characters and output the searching result.

Function/FB classification name: Character string function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN1 IN2	Input variable	STRING (255)	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	INT	Output

Function

Item	Contents
Operation processing	(1) Search the characters string input from input variable IN1 from its head. Output the searching result from the output variable OUT in case of finding the characters input from input variable IN2. The searching result is output to the head character position of the firstly-searched characters. If the IN2 characters are nowhere to be found from character string IN1, "0" will be output.  (2) The values input from input variable IN1 and IN2 are STRING type data within the range of 0 to 255.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>(1) Functions without EN/ENO pins
Execute operation processing and output operation output value from OUT.

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

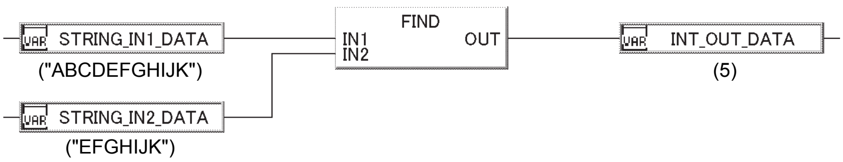
Error

There is no operation error caused by FIND(_E).

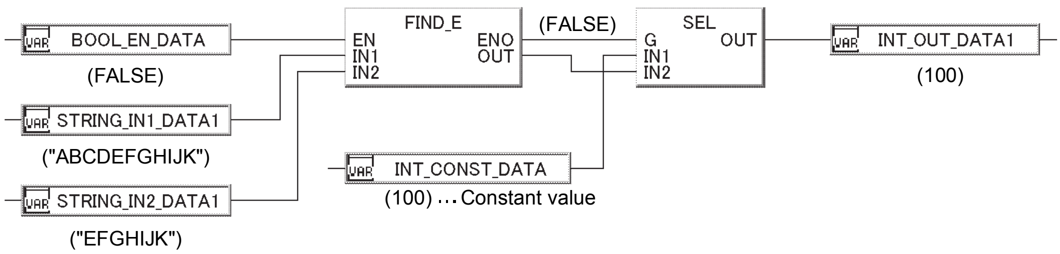
Program Example

The following are the programs that will search the string (input from IN2) from the head position of the string input from IN1 and output from output variable OUT.

(1) Basic program example (FIND)

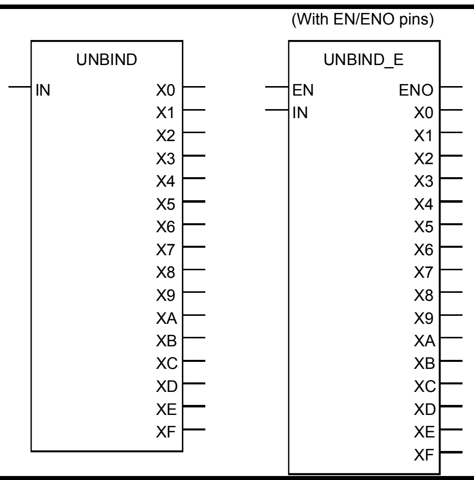


(2) This is the program example in which the output is constant value when the input variable EN is FALSE. (FIND_E)
(Example) When input variable EN is FALSE



4.9 Helper Function

4.9.1 WORD→16BOOL Unbinding (UNBIND(_E))

Function	FBD parts
UNBIND UNBIND_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

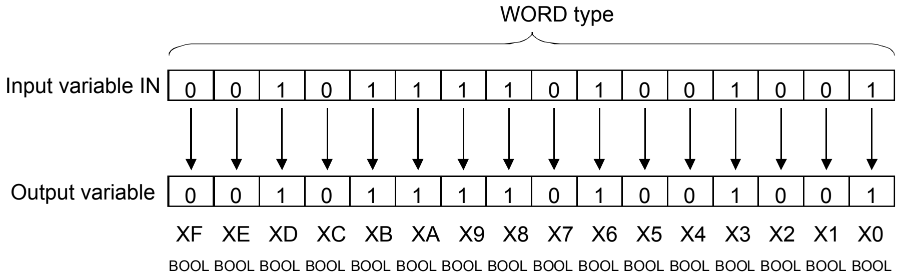
Function overview: Unbind WORD type data into 16 BOOL type data then output the result.
It is applicable in unbinding the WORD type data output from module FB (CCLINK_, CCLINK_2, CCLINK_3, CCLINK_4) to BOOL type data.

Function/FB classification name: Helper function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	WORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	X0 to XF	Output variable	BOOL	Output

Function

Item	Contents
Operation Processing	(1) This processing performs the operation of unbinding the WORD type data input from input variable IN into 16 BOOL type data and output them from the output variable X0 to XF.  (2) The input value of input variable IN is of WORD type data value.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>(1) Functions without EN/ENO pins
Execute operation processing and output the operation output value from OUT.

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>X0 to XF</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FASLE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

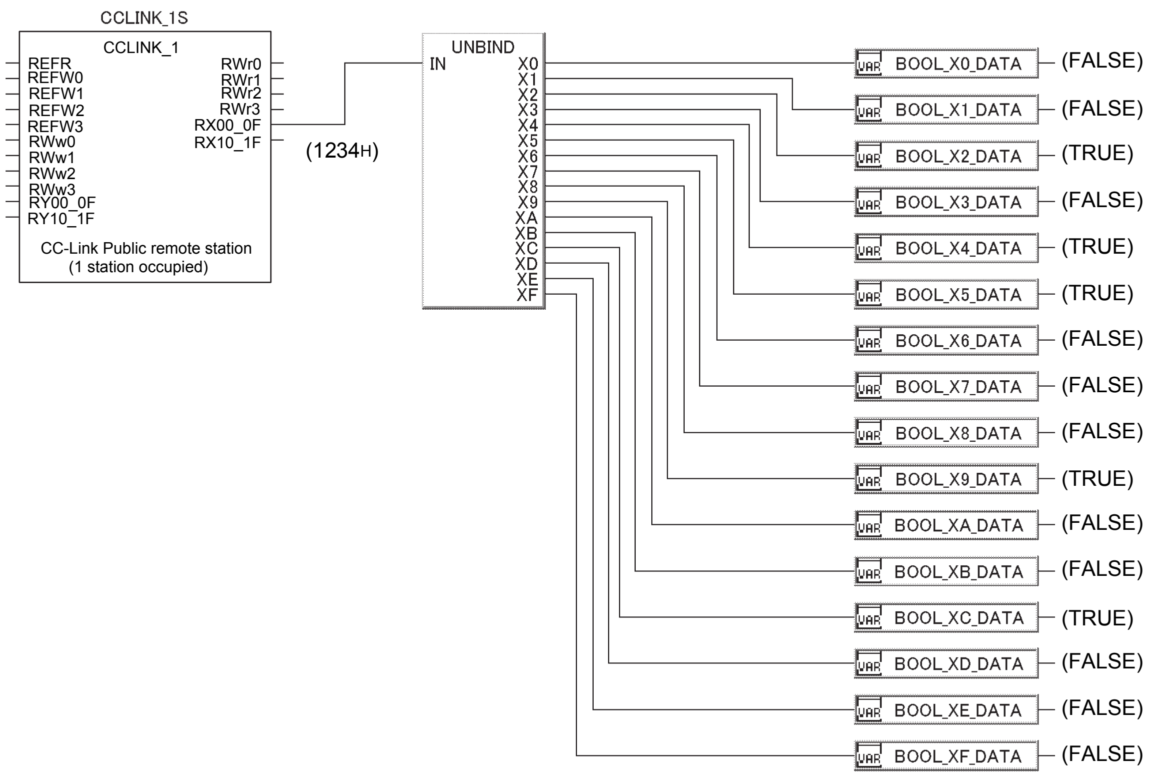
Error

There is no operation error caused by UNBIND(_E).

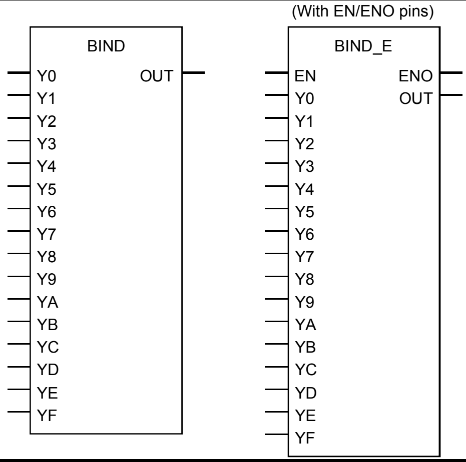
Program Example

Following is the program example in which the WORD type data input from input variable IN is unbound into 16 BOOL type data, and then the result is output from output variable X0 to XF.

(1) Basic program example (UNBIND)



4.9.2 16 BOOL→WORD/DWORD (BIND(_E))

Function	FBD parts
BIND BIND_E	

With EN/ENO pins	○
Overload	○
Input pin number changeable (range)	–

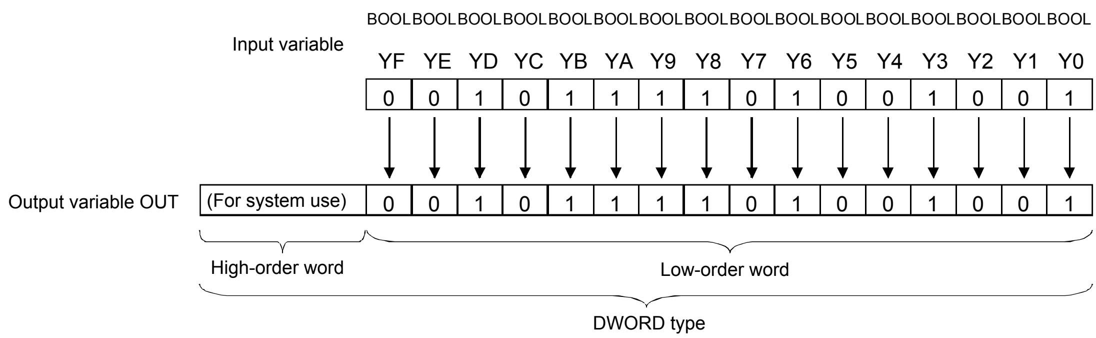
Function overview: Output the 16 BOOL type data in the data type (WORD/DWORD type) connected to output pin OUT.
It is applicable in inputting DWORD type data to input pins (RY00_0F etc.) of module FB (CCLINK_1, CCLINK_2, CCLINK_3, CCLINK_4).

Function/FB classification name: Helper function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	Y0 to YF	Input variable	BOOL	Input
Output	ENO	Output variable	BOOL	Execution status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	WORD DWORD	Output

Function

Item	Contents
Operation Processing	(1) This processing outputs the BOOL type data input from input variable Y0 to YF in the data type (WORD/DWORD type) connected to output pin OUT. (a) When the data that is connected to output pin OUT is DWORD type Arrange the BOOL type data that is input from input variable Y0 to YF into DWORD type data (low-order word), and output them from output variable OUT. The high-order word of the DWORD type data that is output from output variable OUT is used by system. 

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation Processing</td><td>(b) When the data type connected to the output pin OUT is WORD type.
Arrange the BOOL type data input from input variable Y0 to YF into WORD type data and output them from output variable OUT.

(2) The input value of input variable Y0 to YF is BOOL type data value

(3) Compile error will not occur in BIND(_E) even if variable, as well as constant, is not connected to input pins (Y0 to YF).</td></tr>
<tr><td>Operation results</td><td>(1) Functions without EN/ENO pins
Execute operation processing and output the operation output value from OUT.

(2) Functions With EN/ENO pins
The execution conditions and the operation results are as follows:
<table><tr><th>Execution condition</th><th colspan="2">Operation result</th></tr><tr><th>EN</th><th>ENO</th><th>OUT</th></tr><tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr><tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr></table>* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)</td></tr>
</table>

POINT

Compile error will not occur even if variable/constant is not connected to the input pins (Y0 to YF) of BIND(_E). So please connect variable with output pins (OUT). However, compile error will occur when variable is not connected to output pins (OUT).

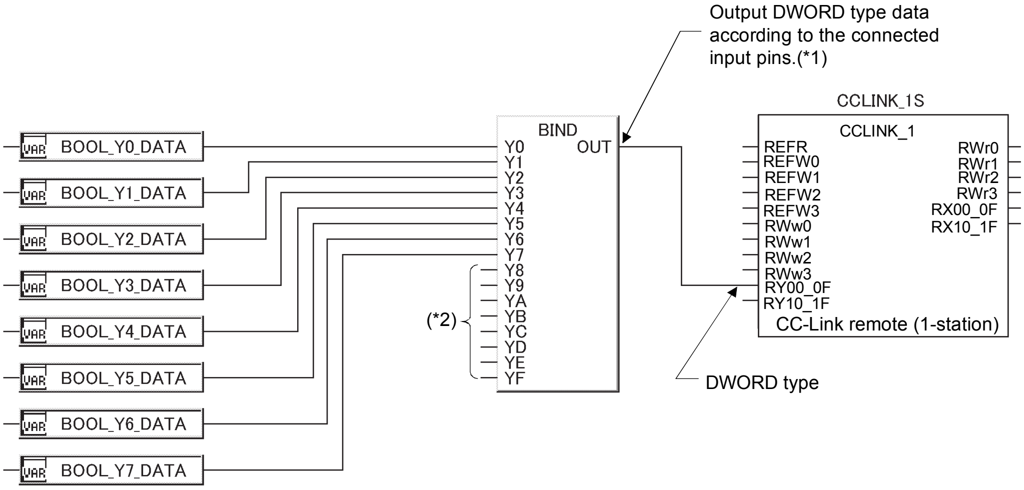
Error

There is no operation error caused by BIND(_E).

Program Example

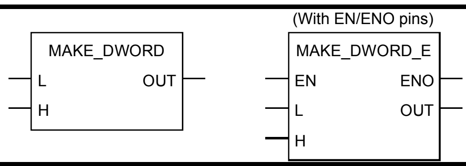
Following is the program example in which data is output to the remote output (RY) of module FB "CCLINK_1S".
Output from FBD program to RY00 to RY07.
Output from ladder program to RY08 to RY0F.

(1) Basic program example: (BIND)



*1 The high-order word of DWORD type data is used by system.
*2 Please do not connect variable to the corresponding pins (Y8 to YF in the above figure) to the remote output (RY) from ladder program.

4.9.3 2WORD→DWORD (MAKE_DWORD(_E))

Function	FBD parts
MAKE_DWORD MAKE_DWORD_E	

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

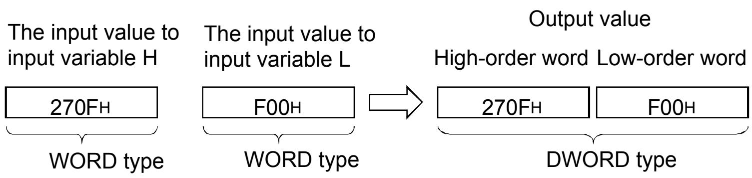
Function overview: Merge 2 WORD type data into 1 DWORD type data and output the result.

Function/FB classification name: Helper function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	L	Input variable	WORD	Input (low-order word)
	H	Input variable	WORD	Input (high-order word)
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	DWORD	Output

Function

Item	Contents
Operation Processing	(1) Following is the processing in which the WORD type data that are input from input variable L and H are merged into DWORD type data and outputs result from output variable OUT. The low-order word of output value is the input value to input variable L and the high-order word of output value is the input value to input variable H.  (2) The input value to input variable L, H is WORD type data value.
Operation results	(1) Functions without EN/ENO pins Execute operation processing and output operation output value from OUT. (2) Functions with EN/ENO pins The execution conditions and the operation results are as follows: Execution condition: EN / Operation result: ENO, OUT TRUE (Operation execution) — TRUE — Operation output value FALSE (Operation stop) — FALSE (*) — Undefined value * When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed. The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins.(☞ Section 2.7.4)

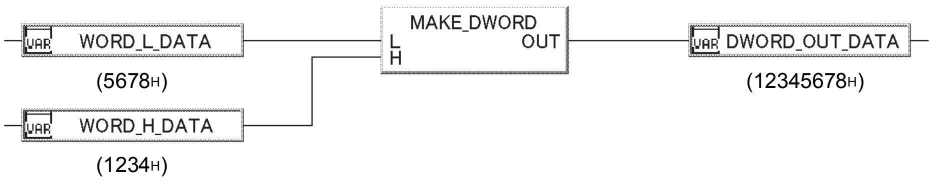
Error

There is no operation error caused by MAKE_DWORD(_E)

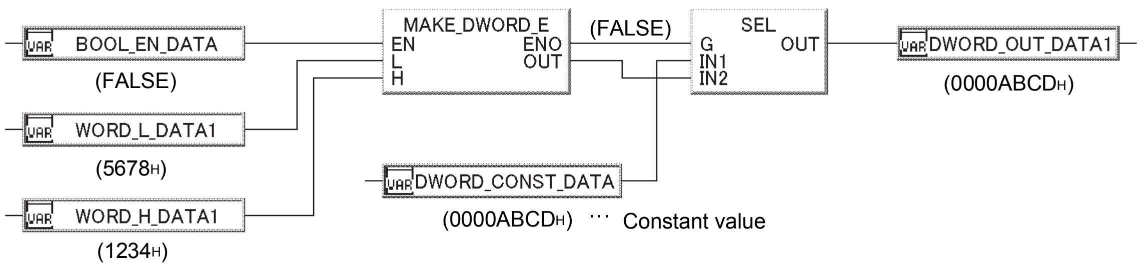
Program Example

The following are the programs that will merge the WORD type data and DWORD type data input from input variable L and H, and output from output variable OUT.

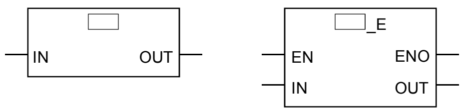
(1) Basic program example (MAKE_DWORD)



(2) This is the program example in which the output is constant value when the input variable EN is FALSE. (MAKE_DWORD_E)



4.9.4 High-order/Low-order Output of DWORD Type Data (HI_WORD(_E), LO_WORD(_E))

Function	FBD parts
HI_WORD LO_WORD HI_WORD_E LO_WORD_E	(With EN/ENO pins)  ☐ Indicates HI_WORD, LO_WORD

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: HI_WORD(_E) outputs the high-order word of DWORD type data.
LO_WORD(_E) outputs the low-order word of DWORD type data.

Function/FB classification name: Helper function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	DWORD	Input
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	WORD	Output

Function

Item	Contents
Operation processing	(1) HI_WORD HI_WORD_E Outputs from the output variable OUT the high-order word of DWORD type data that is input from input variable IN. Input value of input variable IN High-order word Low-order word → Output value 270FH F00H ⇨ 270FH DWORD type → WORD type (2) LO_WORD LO_WORD_E Outputs from the output variable OUT the low-order word of DWORD type data that is input from input variable IN. Input value of input variable IN High-order word Low-order word → Output value 270FH F00H ⇨ F00H DWORD type → WORD type (3) The input value of input variable IN is DWORD type data.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO pins
Execute operation processing and output the operation output value from OUT.

(2) Functions With EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td>TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

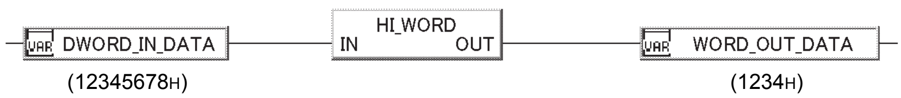
Error

There is no operation error caused by HI_WORD(_E), LO_WORD(_E).

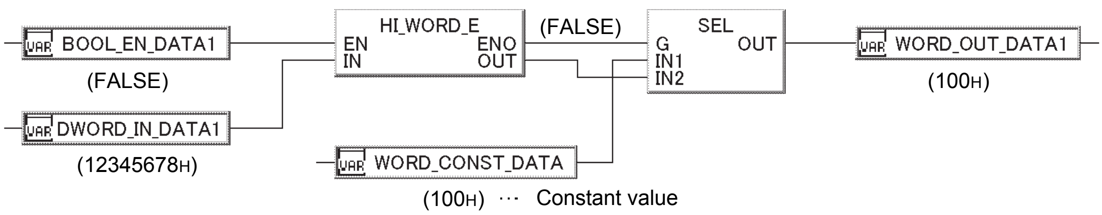
Program Example

(1) Following are the program examples in which the high-order word (of DWORD type data that is input from input variable IN) is output from the output variable OUT.

(a) Basic program example (HI_WORD)

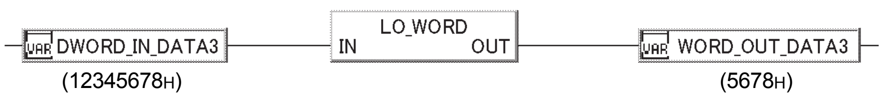


(b) This is the program example in which the output is constant value when the input variable EN is FALSE, or operation error occurs. (HI_WORD_E)

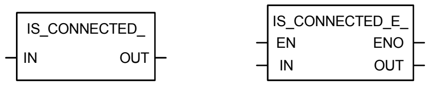


(2) Following is the program example in which the low-order word (of DWORD type data that is input from input variable IN) is output from the output variable OUT.

(a) Basic program example (LO_WORD)



4.9.5 Input Pins Connection Status Acquisition (IS_CONNECTED(_E)_)

Function	FBD parts
IS_CONNECTED_ IS_CONNECTED _E_	(with EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

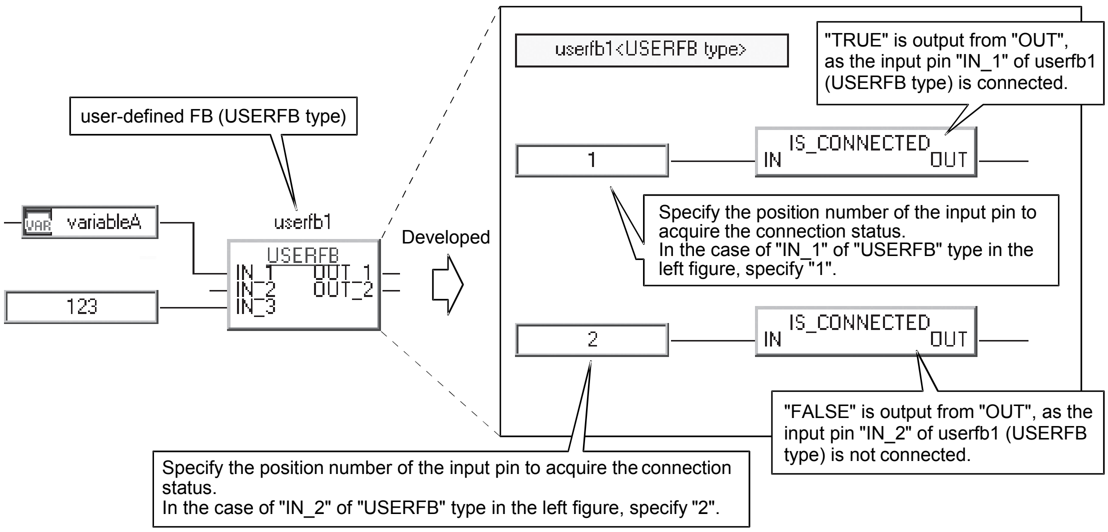
Function overview: Output the connection statuses of input pins of the user-defined FB/user-defined tag on which this function is pasted.

Function/FB classification name: Helper function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	INT	Input in position number (1 to 64)
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)
	OUT	Output variable	BOOL	Connection status (TRUE: Connected FALSE: Unconnected)

Function

Item	Contents
Operation processing	(1) TRUE is output when the input pin of the user-defined FB or user-defined tag FB, which this function is used in, is connected. [Operation Procedure] 1) Paste this function onto the user-defined FB or user-defined tag FB. 2) Input the position number, which is for identifying the input pin position*1, to the input variable IN of this function. *1 The position number is used for identifying the FB input pin position, and assigned to input pins starting from the top one in ascending order (1, 2, …). The number corresponds to the No. of input variable in the local variable sheet for user-defined FB/user-defined tag FB. [Output Results] The result, whether or not the input pin specified by the position number is connected, is output from the output variable OUT of this function. If it is connected, TRUE is output; if not connected, FALSE. [Application Example] 

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(2) This function determines whether maximum of 64 input pins (position number 1 to 64) per user-defined FB/user-defined tag FB are connected or not. For the user-defined FB/user-defined tag FB that has 65 or more input pins, the connection status of 65th or later pin cannot be determined.
(3) If the position number of the input pin that does not exit in the input variable IN is specified (Example: the number smaller than "0" or greater than "4" is input when the target has 3 input pints), an undefined value will be output from the output OUT.
(It is not regarded as an operation error. In the case of IS_CONNECTED_E_, FALSE will be output from the output variable ENO.)
(4) This function is applicable for the user-defined FB or user-defined tag FB only.
(If this function is pasted onto a program, an undefined value will be output from the output variable OUT. In the case of IS_CONNECTED_E_, FALSE will be output from the output variable ENO.)
</td></tr>
<tr><td>Operation result</td><td>
(1) Functions without EN/ENO pins
Execute operation processing and output the operation output value from OUT.
(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE</td><td>Operation output value</td></tr>
<tr><td>FALSE (*1)</td><td rowspan="2">Undefined value (*2)</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE</td></tr>
</table>
*1 If the position number of the input pin that does not exit in the input variable IN is specified, ENO will become FALSE.
*2 When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4))
</td></tr>
</table>

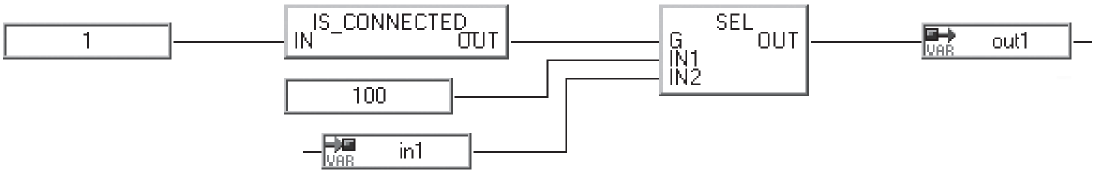
Error

There is no operation error caused by IS_CONNECTED(_E)_

Program Example

(1) The following provides the examples of creating and using the user-defined FB that outputs the corresponding value if the input pin is connected with a connector, and outputs the predetermined value if it is not connected.

1) Example of creating the user-defined FB



The SEL function selects the value of "in1" in the input variable "in1" is connected, and selects the predetermined value "100" if it is not connected, and then, outputs the value from the output variable "out1".

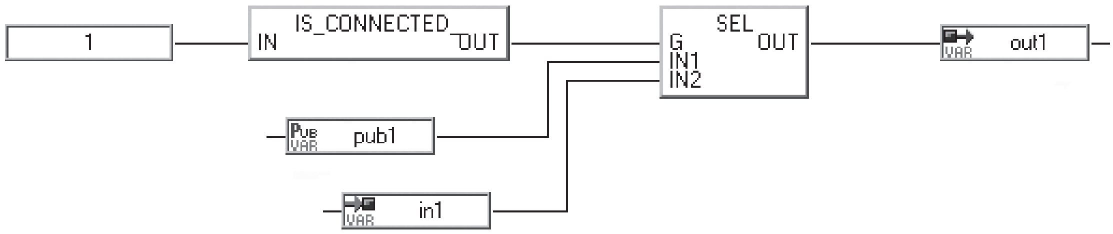
2) Example of using the above user-defined FB

"123" will be output from "out1", as the constant value "123" is connected to "in1".

The predetermined value "100" will be output from "out1", as "in1" is not connected with a connector.

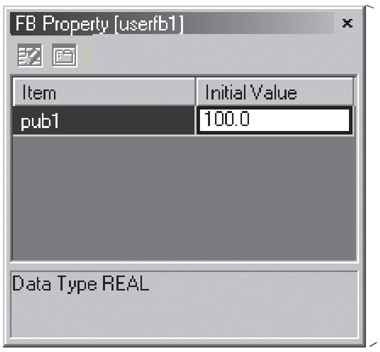
(2) The following provides the examples of creating and using the user-defined FB for setting the initial value to an input variable.
(The user-defined FB outputs the initial value set in the FB property window if the input pin is not connected with a connector, outputs the corresponding value if it is connected.)

1) Example of creating the user-defined FB

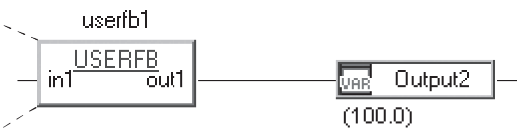


The SEL function selects the value of "in1" if the input variable "in1" is connected with a connector, and selects the initial value (public variable "pub1") set in the FB property window if it is not connected, and then outputs the value from the output variable "out1" .

2) Example of using the above user-defined FB



Set the initial value "pub1" of the input variable "in1" on the FB property window of "userfb1".



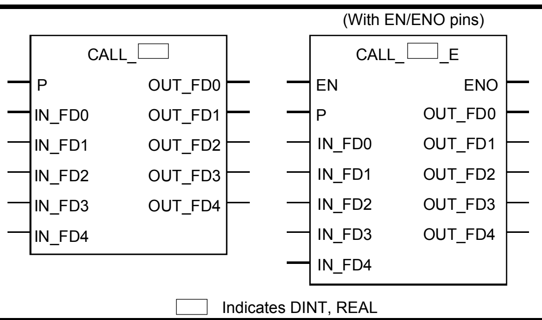
The initial value (100) set in the FB property window will be output from "out1", as "in1" is not connected with a connector.

"200" will be output from "out1", as the constant value "200" is connected to "in1".

4.10 Ladder Program Control Function

4.10.1 Sub-routine Program Call (DINT/REAL Type Argument) (CALL_DINT(_E), CALL_REAL(_E))

Function	FBD parts
CALL_DINT CALL_REAL CALL_DINT_E CALL_REAL_E	(see figure below)



With EN/ENO pins	○
Overload	—
Input pin number changeable (range)	—

Functions summary: CALL_DINT(_E) Subroutine program call. Input argument (DINT type) into input variable IN_FD0 to IN_FD4. (Execute the same processing as CALL instruction of sequent program).

CALL_REAL(_E) Subroutine program call. Input argument (REAL type) into input variable IN_FD0 to IN_FD4. (Execute the same processing as CALL instruction of sequent program).

Function/FB classification name: Ladder program control function

Input and Output Pins

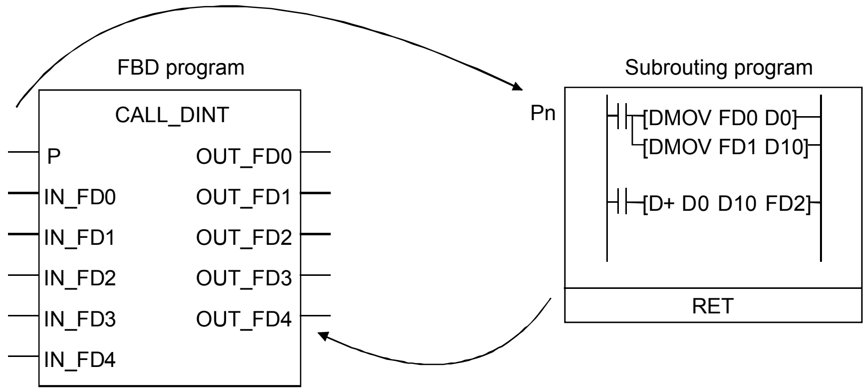
(1) CALL_DINT CALL_DINT_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute, FALSE: Stop)
	P	Input variable	INT	The common pointer number of subroutine program
	IN_FD0 to IN_FD4	Input variable	DINT	Argument input of subroutine program
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal, FALSE: Abnormal)
	OUT_FD0 to OUT_FD4	Output variable	DINT	Argument (return value) output of subroutine program

(2) CALL_REAL CALL_REAL_E

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute, FALSE: Stop)
	P	Input variable	INT	The common pointer number of subroutine program
	IN_FD0 to IN_FD4	Input variable	REAL	Argument input of subroutine program
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal, FALSE: Abnormal)
	OUT_FD0 to OUT_FD4	Output variable	REAL	Argument (return value) output of subroutine program

Function

Item	Contents
Operation Processing	(1) This processing executes the subroutine program whose pointer is specified by the input value of input variable P. (CALL_DINT(_E) and CALL_REAL(_E) can execute the subroutine program specified in the common pointer. Execute subroutine program call, and return to the next step of CALL_DINT(_E) and CALL_REAL(_E) after terminating subroutine program by RET instruction.)  (2) The input value from input variable P is INT type data value whose range is 0 to 3499. (The data value beyond 3500 is used by system) The pointer of subroutine program executed in (CALL_DINT(_E) or CALL_REAL(_E) (input value to input variable P) should be within the range of the common pointer. (The starting number of the common pointer is set in the PLC system setting of PLC parameter of GX application.) (3) The input value of input variable IN of CALL_DINT(_E) is DINT type data value. The input value of input variable IN of CALL_REAL(_E) is REAL type data value. The argument is input from input variable IN_FD0 to IN_FD4 during subroutine program execution. Maximum 5 argument (IN_FD0 to IN_FD4) specified. 0 is input when no argument input (when input pin is not connected) (4) There are no compile errors even if variables/constants are not connected to input pins (IN_FD0 to IN_FD4) in CALL_DINT(_E) and CALL_REAL(_E). (5) The operation to execute CALL_DINT(_E) and CALL_REAL(_E) is described below (a) The argument input from input variable IN will be transmitted to function device of subroutine program before executing subroutine program (b) The contents of function device will be transmitted to the corresponding OUT_FD0 to OUT_FD4 (return value) after executing subroutine program (c) Please use FD for function device. Assign the DINT/REAL type data value specified by input variable (IN_FD0 to IN_FD4) to FD.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation results</td><td>
(1) Functions without EN/ENO
The execution results are as follows.
<table>
<tr><th>Execution result</th><th>OUT_FD0 to OUT_FD4</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error occur</td><td>Undefined value</td></tr>
</table>
(2) Functions With EN/ENO pins
The execution conditions and the execution results are as follows.
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT_FD0 to OUT_FD4</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error occur) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

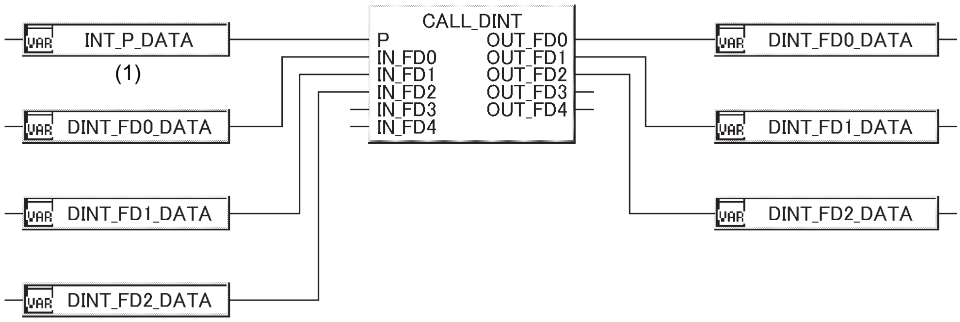
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When RET instruction is executed between CALL_DINT(_E), CALL_REAL(_E) execution and END, FEND, GOEND, STOP instruction execution. (Error code: 4211)
- When RET instruction is executed before executing CALL_DINT(_E) or CALL_REAL(_E). (Error code: 4212)
- When the 17 level nesting is executed. (Error code: 4213)
- When the subroutine program of the pointers specified by CALL_DINT(_E) or CALL_REAL(_E) does not exist. (Error code: 4210)

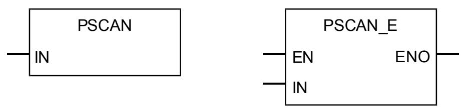
Program Example

Following is the program example in which the subroutine program whose pointers are specified by input value of input variable P is executed.

(1) Basic program example (CALL_DINT)



4.10.2 Program Scan Execution Registration (PSCAN(_E))

Function	FBD parts
PSCAN PSCAN_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

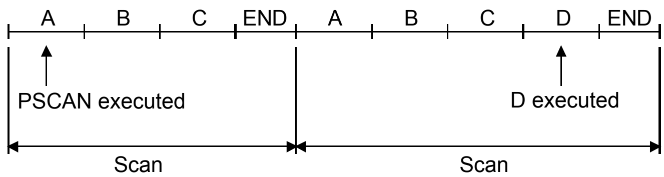
Function overview: Sets a sequence program into the scan execution mode.

Function/FB classification name: Ladder program control function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (8)	Program file name input
Output	ENO	Output variable	BOOL	Execution result (TRUE: Normal FALSE: Abnormal)

Function

Item	Contents
Operation Processing	(1) Set the file name program input from input variable IN as scan execution mode. (2) Only the program saved in program memory (Whose driver number is 0) can be set as scan execution mode. (3) The specified program switches to scan execution mode in END processing. (Example) Assuming there are programs A, B and C, the following is the situation when FBD program A executes "PSCAN" on ladder program D.  (4) This function has the priority even if the execution format is specified by parameter. (5) It is not necessary to specify the extension (.QPG) for file name. (For .QPG file only) (6) In case of PSCAN_E, FALSE will be output from output variable ENO when the input variable EN is FALSE or operation errors occur.

Restrictions on redundant system operation and the applicable corrective action

Restrictions	Corrective action
Executing the ladder program control function does not carry out the ladder program control if the following occurs: a Redundant CPU stop error occurs and the system is switched before the END processing is executed.	Execute the ladder program control function in the new control system immediately after system switching, as necessary.

Error

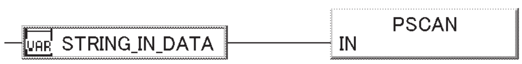
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the program with specified file name does not exist. (Error code: 2410)

Program Example

Following is the program example in which the file name program input from input variable IN is set as scan execution status.

(1) Basic program example (PSCAN)



("SCANSUB")

4.10.3 Program Standby Instruction (PSTOP(_E))

Function	FBD parts
PSTOP PSTOP_E	PSTOP: IN (With EN/ENO) pins PSTOP_E: EN, IN, ENO

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Sets a sequence program into standby mode.

Function/FB classification name: Ladder program control function

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (8)	Program file name input
Output	ENO	Output variable	BOOL	Execution result (TRUE: Normal FALSE: Abnormal)

Function

Item	Contents
Operation processing	(1) Set the file name program input from input variable IN as standby mode. (2) Only the program saved in program memory (Whose driver number is 0) can be set as standby mode. (3) The specified program switches to standby mode in END processing. (4) This function has the priority even if the execution format is specified by parameter. (5) It is not necessary to specify the extension (.QPG) for file name. (For .QPG file only) (6) In case that PSTOP_E, FALSE will be output from output variable ENO when the input variable EN is FALSE or operation errors occur.

Restrictions on redundant system operation and the applicable corrective action

Restrictions	Corrective action
Executing the ladder program control function does not carry out the ladder program control if the following occurs: a Redundant CPU stop error occurs and the system is switched before the END processing is executed.	Execute the ladder program control function in the new control system immediately after system switching, as necessary.

Error

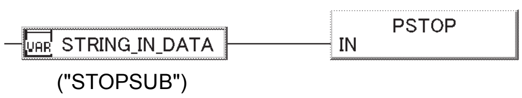
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the program with specified file name does not exist. (Error code: 2410)

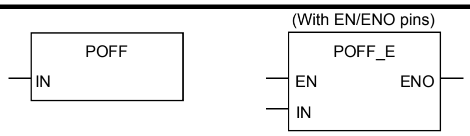
Program Example

Following is the program example in which the file name program input from input variable IN is set as standby mode.

(1) Basic program example (PSTOP)



4.10.4 Program Output Standby Instruction (POFF(_E))

Function	FBD parts
POFF POFF_E	

With EN/ENO pins	○
Overload	—
Input pin number changeable (range)	—

Function overview: Sets a sequence program into standby mode including reset of the outputs.

Function/FB classification name: Ladder program control function.

Input and output pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (8)	Program file name input
Output	ENO	Output variable	BOOL	Execution result (TRUE: Normal FALSE: Abnormal)

Function

Item	Contents
Operation Processing	(1) Change the execution type of the file name program input from input variable IN. Scan execution type : Reset of output (non-executed processing) in the next scanning. Switches into standby mode after the next scanning. Low-speed execution type: Interrupt the low-speed type execution and reset of output in next scanning. Switches into standby mode after the next scanning. (2) Under the non-executed status, only the programs saved in program memory (Whose driver number is 0) can be set as standby mode. (3) This function has the priority even if the execution type is specified by parameter. (4) It is not necessary to specify the extension (.QPG) in file name. (For .QPG file only) (5) In case of POFF_E, FALSE will be output from output variable ENO when the input variable EN is FALSE or operation errors occur.

REMARK

(1) Non-executed processing performs coil instructions through the same processing with the one whose condition setting is OFF.

(2) The operation results of all coil instructions after non-executed processing have nothing to do with ON/OFF of condition contact.

- OUT instruction .... Forced OFF
- SET instruction
- RST instruction
- SFT instruction
- Basic instruction
- Application instruction .... Hold status
- PLS instruction
- Pulse generate (□P) .... The same processing as condition contact OFF
- Current value of low-speed/high-speed timer 0
- Current value of integrate timer
- Current value of counter .... Hold

Restrictions on redundant system operation and the applicable corrective action

Restrictions	Corrective action
Executing the ladder program control function does not carry out the ladder program control if the following occurs: a Redundant CPU stop error occurs and the system is switched before the END processing is executed.	Execute the ladder program control function in the new control system immediately after system switching, as necessary.

Error

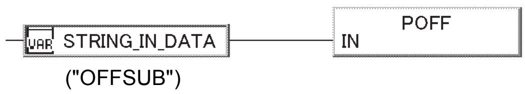
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the program with specified file name does not exist. (Error code: 2410)

Program Example

Following are the program examples in which the execution type (of file name program input from input variable IN) is changed.

(1) Basic program example (POFF)



4.10.5 Program Low-speed Execution Registration (PLOW(_E))

Function	FBD parts
PLOW PLOW_E	PLOW: IN (With EN/ENO pins) PLOW_E: EN, IN / ENO

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

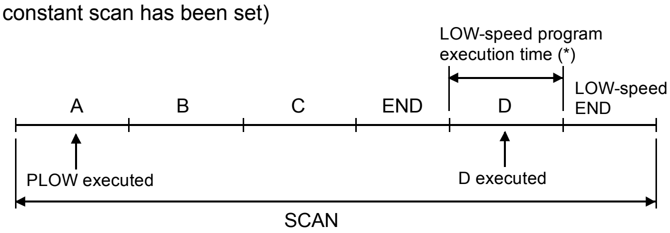
Function overview: Sets a sequence program into the low-speed execution mode.

Function/FB classification name: Ladder program control function

Input and output pins

Pin	Variable name	Variable type	Data type	Contents
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)
	IN	Input variable	STRING (8)	Program file name input
Output	ENO	Output variable	BOOL	Execution result (TRUE: Normal FALSE: Abnormal)

Function

Item	Contents
Operation processing	(1) Set the file name program input from input variable IN as low-speed execution mode. (2) Only the programs saved in the program memory (Whose driver number is 0) can be set as low-speed execution mode. (3) The specified program switches into low-speed execution mode in END processing. (Example) Provided that there are programs A, B and C, the following is the situation when FBD program A executes "PLOW" on ladder program D. (Provided that the constant scan has been set)  * Set the LOW-speed program execution time on PLC parameter beforehand. LOW-speed program will not be executed without setting. (4) This function has the priority even if the execution type is specified by parameter. (5) It is not necessary to specify the extension (.QPG) for file name. (For .QPG file only) (6) In case of PLOW_E, FALSE will be output from output variable ENO when the input variable EN is FALSE or operation errors occur.

Restrictions on redundant system operation and the applicable corrective action

Restrictions	Corrective action
This instruction is unsupported by Universal model process CPU and Redundant CPU.	Do not use the PLOW(_E) for a project for which Universal model process CPU or Redundant CPU has been set as CPU type, as this may cause a compile error.

Error

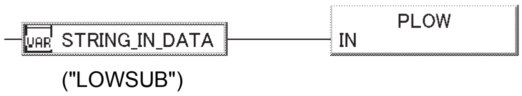
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When the program with specified file name does not exist. (Error code: 2410)
- When the program with specified file name contains the CHK instruction. (Error code: 4235)

Program Example

Following is the program example in which the file name program input from input variable IN is set as low-speed execution status.

(1) Basic program example (PLOW)



MEMO

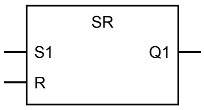
5 GENERAL FB

General FB can be classified as follows.

Classification name	Description	Reference
Bistable FB	Sets the priority of set/reset and outputs the latch processing result	Section 5.1
Edge Detection FB	Output rising/Falling edge detection	Section 5.2
Counter FB	Output current value +1(addition) or current value -1 (subtraction).	Section 5.3
Timer FB	Performs pulse timer processing and ON/OFF delay timer processing	Section 5.4
Communication control FB	Sends/Receives data to/from PLC CPU of other stations	Section 5.5

5.1 Bistable FB

5.1.1 Set-Dominant Flip-Flop (SR)

FB	FBD parts
SR	

Function overview: SR flip-flop. If the input value of input variable S1 and R are both TRUE, the set (TRUE) has the priority.

Function/FB classification name: Bistable FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	S1	Input variable	BOOL	Set instruction
	R	Input variable	BOOL	Reset instruction
Output	Q1	Output variable	BOOL	Output

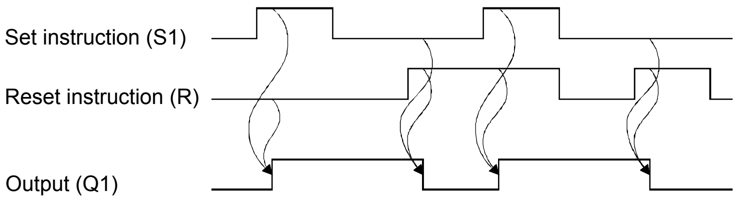
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IR	Public variable	BOOL	Reset request when CPU runs at the first time TRUE : At a CPU module startup (power OFF →ON, RESET/STOP→RUN), the output value of the output variable Q1 is reset. FALSE: At a CPU module startup (power OFF → ON, RESET/STOP→RUN), the output value of the output variable Q1 is held.	TRUE, FALSE	FALSE	User

Function

Item	Contents
Operation processing	(1) This processing outputs the result from output variable Q1 according to two input variables S1 and R. (a) If the input variable S1 is TRUE, the output from output variable Q1 will be TRUE. The output value (TRUE) remains until TRUE is input into input variable R. (b) If the input variable R is TRUE, the output from output variable Q1 will be FALSE. The output value (FALSE) remains until TRUE is input into input variable S1. (c) If the input variable S1 and R are both TRUE, the output from output variable Q1 will be TRUE. (Set-dominant) (d) If the input variable S1 and R are both FALSE, the output from output variable Q1 will be the previous value.

Input variable		Output variable
S1	R	Q1
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	FALSE	Previous value



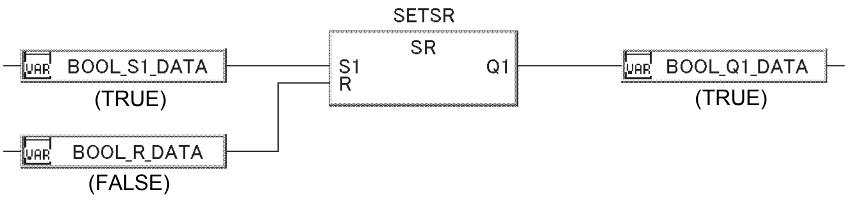
(2) The input value of input variable S1 and R should be BOOL type data. (Default value: FALSE)
(3) When the public variable IR is TRUE, the output value of the output variable Q1 is reset at a CPU module startup (power OFF→ON, RESET/STOP→RUN). When the public variable IR is FALSE, the output value of the output variable Q1 is held.

Error

There are no operation errors caused by SR.

Program Example

(1) Basic program example



5.1.2 Reset-Dominant Flip-Flop (RS)

FB	FBD parts
RS	

Function overview: RS flip-flop. If the input value to input variable S and R1 are both TRUE, the reset (FALSE) has the priority.

Function/FB classification name: Bistable FB

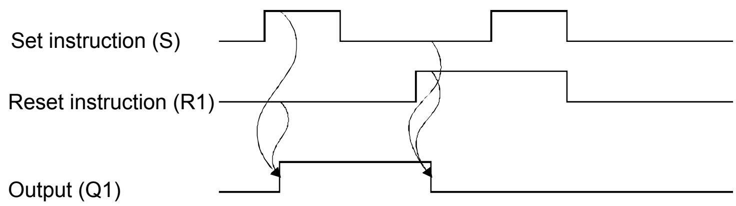
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	S	Input variable	BOOL	Set instruction
	R1	Input variable	BOOL	Reset instruction
Output	Q1	Output variable	BOOL	Output

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IR	Public variable	BOOL	Reset request when CPU runs at the first time TRUE : At a CPU module startup (power OFF →ON, RESET/STOP→RUN), the output value of the output variable Q1 is reset. FALSE: At a CPU module startup (power OFF → ON, RESET/STOP→RUN), the output value of the output variable Q1 is held.	TRUE, FALSE	FALSE	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(1) This processing outputs the result from output variable Q1 according to two input variables S and R1.
(a) If the input variable S is TRUE, the output from output variable Q1 will be TRUE. The output value (TRUE) remains until TRUE is input into input variable R1.
(b) If the input variable R1 is TRUE, the output from output variable Q1 will be FALSE. The output value (FALSE) remains until TRUE is input into input variable S.
(c) If the input variable S and R1 are both TRUE, the output from output variable Q1 will be TRUE. (Priority set)
(d) If the input variable S and R1 are both FALSE, the output from output variable Q1 will be the previous value.
<table>
<tr><th colspan="2">Input variable</th><th>Output variable</th></tr>
<tr><th>S</th><th>R1</th><th>Q1</th></tr>
<tr><td>TRUE</td><td>FALSE</td><td>TRUE</td></tr>
<tr><td>FALSE</td><td>TRUE</td><td>FALSE</td></tr>
<tr><td>TRUE</td><td>TRUE</td><td>FALSE</td></tr>
<tr><td>FALSE</td><td>FALSE</td><td>Previous value</td></tr>
</table>


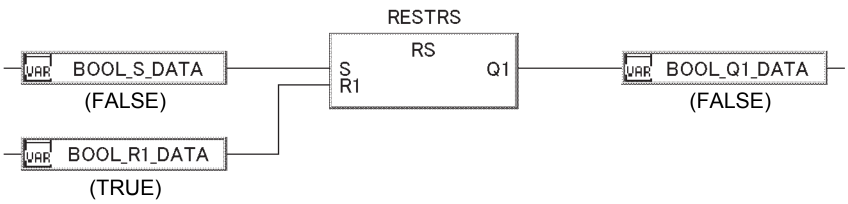
(2) The input value of input variable S and R1 should be BOOL type data. (Default value: FALSE)
(3) When the public variable IR is TRUE, the output value of the output variable Q1 is reset at a CPU module startup (power OFF→ON, RESET/STOP→RUN). When the public variable IR is FALSE, the output value of the output variable Q1 is held.
</td></tr>
</table>

Error

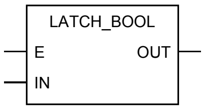
There are no operation errors caused by RS.

Program Example

(1) Basic program example



5.1.3 Latch FB (BOOL Type) (LATCH_BOOL)

FB	FBD parts
LATCH_BOOL	

Function overview: Latch FB (BOOL type). Outputs the result of latch processing.

Function/FB classification name: Bistable FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	E	Input variable	BOOL	Latch processing (TRUE: No latch, FALSE: Latch)
	IN	Input variable	BOOL	Input
Output	OUT	Output variable	BOOL	Output

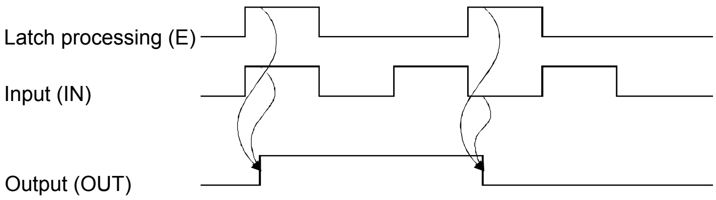
Function

Item: Operation processing

Contents:

(1) This processing outputs operation result of the input of two input variables E and IN from output variable OUT.
   (a) If the input variable E is TRUE, the input value of input variable IN is output from output variable.
   (b) If the input variable E is FALSE, no matter what the input value of input variable IN is, the output from output variable OUT will be the previous value. (The previous value remained)

Input variable		Output variable
E (BOOL type)	IN (BOOL type)	OUT (BOOL type)
TRUE	TRUE	TRUE
	FALSE	FALSE
FALSE	TRUE	Previous value
	FALSE	



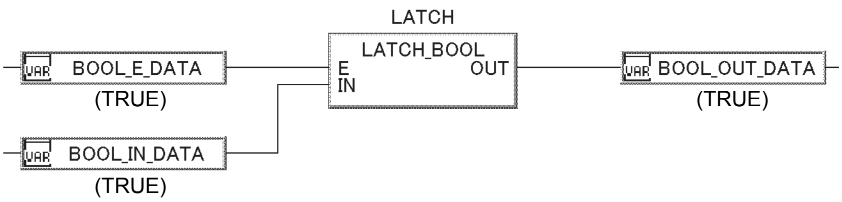
(2) The input value of input variable E and IN should be BOOL type data. (Default: FALSE)

Error

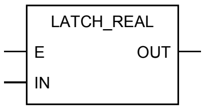
There are no operation errors caused by LATCH_BOOL.

Program Example

(1) Basic program example



5.1.4 Latch FB (REAL Type) (LATCH_REAL)

FB	FBD parts
LATCH_REAL	

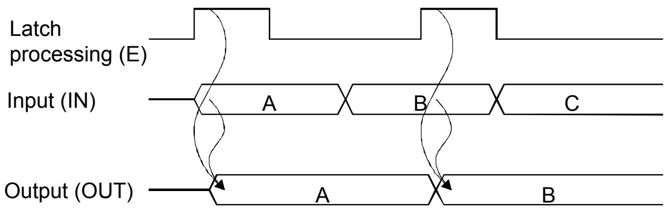
Function overview: Latch FB (REAL type). Outputs the operation result of latch processing.

Function/FB classification name: Bistable FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	E	Input variable	BOOL	Latch processing (TRUE: No latch, FALSE: Latch)
	IN	Input variable	REAL	Input
Output	OUT	Output variable	REAL	Output

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(1) This processing outputs the result from output variable OUT according to input value of input variables E and IN.
(a) If the input variable E is TRUE, the output of output variable OUT will be the input value of input variable IN.
(b) If the input variable E is FALSE, no matter what the input value of input variable IN is, the output value of output variable OUT will be the previous value. (The previous value remained)
<table>
<tr><th colspan="2">Input variable</th><th>Output variable</th></tr>
<tr><th>E
(BOOL type)</th><th>IN
(REAL type)</th><th>OUT
(REAL type)</th></tr>
<tr><td>TRUE</td><td>Input value</td><td>Input value</td></tr>
<tr><td>FALSE</td><td>Input value</td><td>Previous value</td></tr>
</table>


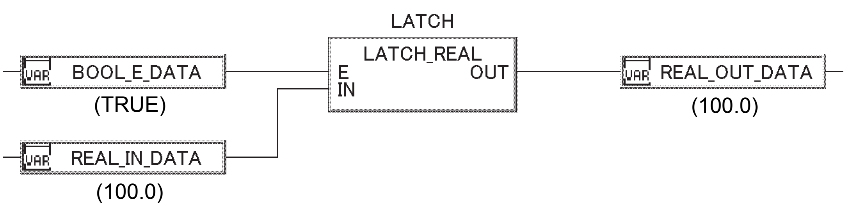
(2) The input value of input variable E should be BOOL type data. (Default value: FALSE)
(3) The input value of input variable IN should be REAL type data. (Default value: 0.0)
</td></tr>
</table>

Error

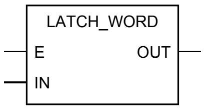
There are no operation errors caused by LATCH_REAL.

Program Example

(1) Basic program example



5.1.5 Latch FB (WORD Type) (LATCH_WORD)

<table>
<tr><th>FB</th><th>FBD parts</th></tr>
<tr><td>LATCH_WORD</td><td></td></tr>
</table>

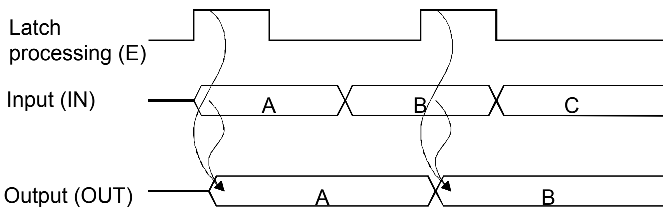
Function overview: Latch FB (WORD type). Outputs the result of latch processing.

Function/FB classification name: Bistable FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	E	Input variable	BOOL	Latch processing (TRUE: No latch, FALSE: Latch)
	IN	Input variable	WORD	Input
Output	OUT	Output variable	WORD	Output

Function

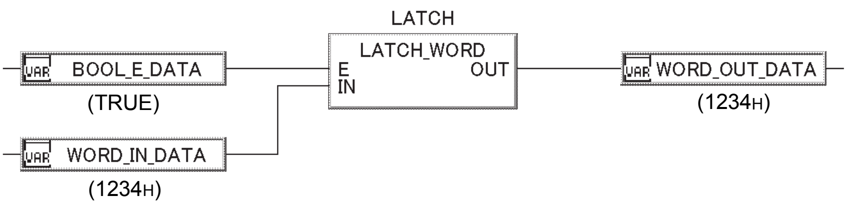
<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(1) This processing outputs the result to output variable OUT according to input value of input variables E and IN.
(a) If the input variable E is TRUE, the input value of input variable IN will be output from output variable OUT.
(b) If the input variable E is FALSE, no matter what the input value of input variable IN is, the output value of output variable OUT will be the previous value. (The previous value remained)
<table>
<tr><th colspan="2">Input variable</th><th>Output variable</th></tr>
<tr><th>E
(BOOL type)</th><th>IN
(WORD type)</th><th>OUT
(WORD type)</th></tr>
<tr><td>TRUE</td><td>Input value</td><td>Input value</td></tr>
<tr><td>FALSE</td><td>Input value</td><td>Previous value</td></tr>
</table>

(2) The input value of input variable E should be BOOL type data. (Default value: FALSE)
(3) The input value of input variable IN should be WORD type data. (Default value: 0H)
</td></tr>
</table>

Error

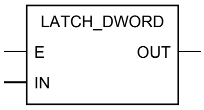
There are no operation errors caused by LATCH_WORD.

Program Example

(1) Basic program example



5.1.6 Latch FB (DWORD Type) (LATCH_DWORD)

FB	FBD parts
LATCH_DWORD	

Function overview: Latch FB (DWORD type). Outputs the result of latch processing.

Function/FB classification name: Bistable FB

Input and Output Pins

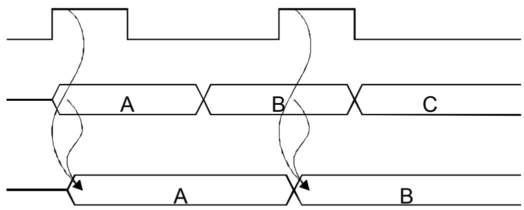
Pin	Variable name	Variable type	Data type	Contents
Input	E	Input variable	BOOL	Latch processing (TRUE: No latch, FALSE: Latch)
	IN	Input variable	DWORD	Input
Output	OUT	Output variable	DWORD	Output

Function

Item	Contents
Operation processing	(1) This processing outputs operation result of the input value of input variables E and IN from output variable OUT. (a) If the input variable E is TRUE, the output of output variable OUT will be the input value of input variable IN. (b) If the input variable E is FALSE, no matter what the input value of input variable IN is, the output value of output variable OUT will be the previous value. (The previous value remained)

Input variable		Output variable
E (BOOL type)	IN (DWORD type)	OUT (DWORD type)
TRUE	Input value	Input value
FALSE	Input value	Previous value

Latch processing (E)
Input (IN)
Output (OUT)



(2) The input value of input variable E should be BOOL type data. (Default value: FALSE)

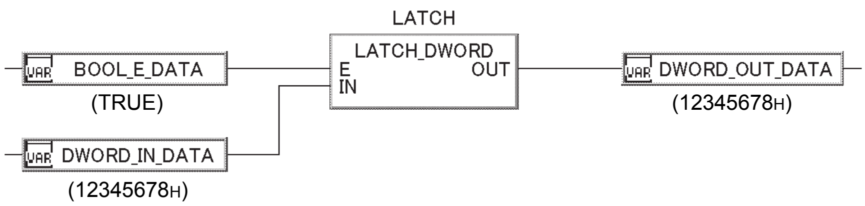
(3) The input value of input variable IN should be DWORD type data. (Default value: 0_H)

Error

There are no operation errors caused by LATCH_ DWORD.

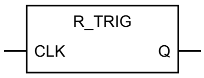
Program Example

(1) Basic program example



5.2 Edge Detection FB

5.2.1 Rising Edge Detector (R_TRIG)

FB	FBD parts
R_TRIG	

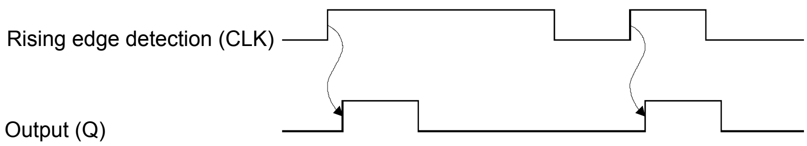
Function overview: Outputs TRUE when the rising edge is detected. (When input variable CLK changes from FALSE to TRUE)

Function/FB classification name: Edge detection FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	CLK	Input variable	BOOL	Rising edge detection
Output	Q	Output variable	BOOL	Output

Function

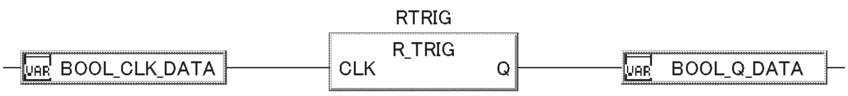
Item	Contents
Operation processing	(1) When the input variable CLK changes from FALSE to TRUE, TRUE will be output from output variable Q. When the input variable CLK does not change from FALSE to TRUE (e.g. TRUE→TRUE, TRUE→FALSE or FALSE→FALSE), FALSE will be output from output variable Q.  (2) The input value of the input variable CLK should be BOOL type data. (Default value: FALSE)

Error

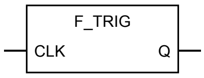
There are no operation errors caused by R_TRIG.

Program Example

(1) Basic program example



5.2.2 Falling Edge Detector (F_TRIG)

FB	FBD parts
F_TRIG	

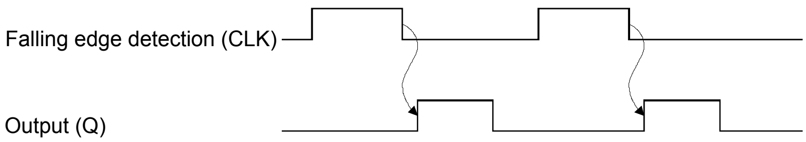
Function overview: Outputs TRUE when the Falling edge is detected. (When input variable CLK changes from TRUE to FALSE)

Function/FB classification name: Edge check FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	CLK	Input variable	BOOL	Falling edge detection
Output	Q	Output variable	BOOL	Output

Function

Item	Contents
Operation processing	(1) When the input variable CLK changes from TRUE to FALSE, TRUE will be output from output variable Q. When the input variable CLK does not change from TRUE to FALSE (e.g. FALSE→FALSE, FALSE→TRUE or TRUE→TRUE), FALSE will be output from output variable Q.  (2) The input value of input variable CLK should be BOOL type data. (Default value: FALSE)

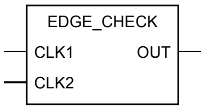
Error

There are no operation errors caused by F_TRIG.

Program Example

(1) Basic program example

5.2.3 Edge Detection Input (EDGE_CHECK)

FB	FBD parts
EDGE_CHECK	

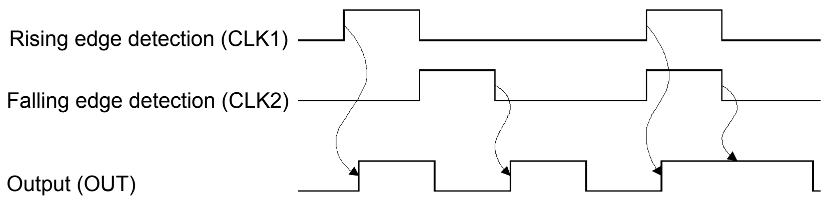
Function overview: Outputs TRUE when rising/falling edge is detected. (When the input variable CLK1 changes from FALSE to TRUE or the input variable CLK2 changes from TRUE to FALSE.)

Function/FB classification name: Edge detection FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	CLK1	Input variable	BOOL	Rising edge detection
	CLK2	Input variable	BOOL	Falling edge detection
Output	OUT	Output variable	BOOL	Output

Function

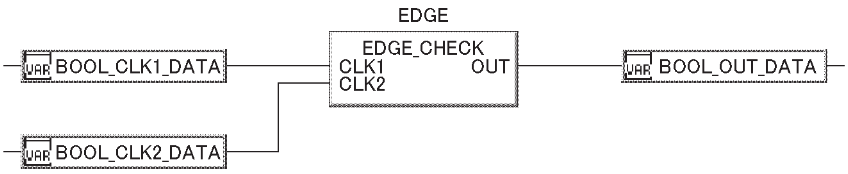
Item	Contents
Operation processing	(1) When the input variable CLK1 changes from FALSE to TRUE or the input variable CLK2 changes from TRUE to FALSE, TRUE will be output from output variable OUT. In the following conditions, FALSE will be output from output variable OUT: When the input value of input variable CLK1 does not change from FALSE to TRUE (TRUE→TRUE, TRUE→FALSE, FALSE→FALSE) When the input value of input variable CLK2 does not change from TRUE to FALSE (FALSE→FALSE, FALSE→TRUE, TRUE→TRUE)  (2) The input value of input variables CLK1 and CLK2 should be BOOL type data value. (Default value: FALSE)

Error

There are no operation errors caused by EDGE_CHECK.

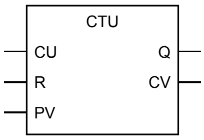
Program Example

(1) Basic program example



5.3 Counter FB

5.3.1 Up-counter (CTU)

FB	FBD parts
CTU	

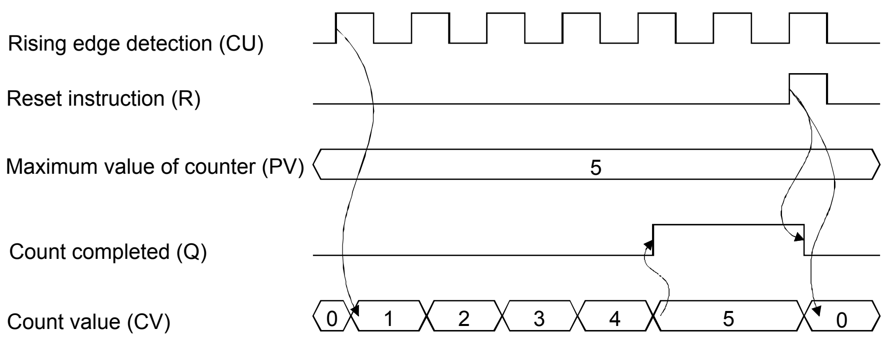
Function overview: Up-counter. When the input variable CU changes from FALSE to TRUE, current value+1 (Count value) will be output.

Function/FB classification name: Counter FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	CU	Input variable	BOOL	Rising edge detection
	R	Input variable	BOOL	Reset instruction
	PV	Input variable	INT	Maximum value of counter
Output	Q	Output variable	BOOL	Count completed
	CV	Output variable	INT	Count value

Function

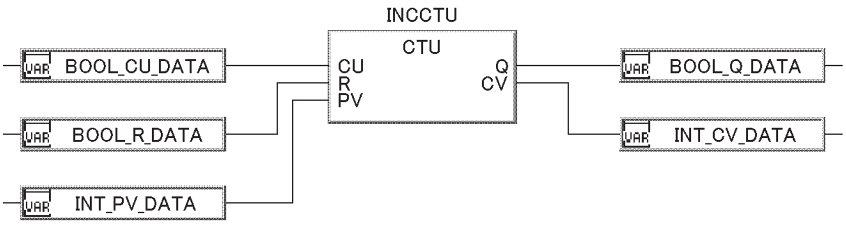
Item	Contents
Operation processing	(1) If the input variable CU changes from FALSE to TRUE, the current value+1 will be output from output variable CV. (2) When output variable CV reaches the input value of input variable PV, TRUE will be output from output variable Q and the count up will be stopped. When the output variable Q is TRUE, the input value of input variable PV will be output from output variable CV. (CV is kept) (3) When the input variable R is TRUE, the output of output variable Q will be FALSE and that of output variable CV will be 0. (4) When the input variable CU does not changes from FALSE to TRUE (e.g. TRUE→TRUE, TRUE→FALSE or FALSE→FALSE), the output from output variable CV will be the previous value. (5) The input value of input variable CU and R should be BOOL type data. (Default value: FALSE) (6) The input value of input variable PV should be INT type data value within the range of 0 to 32767. (Default value: 0) 

Error

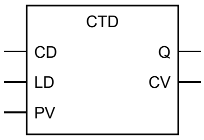
There are no operation errors caused by CTU.

Program Example

(1) Basic program example



5.3.2 Down-counter (CTD)

FB	FBD parts
CTD	

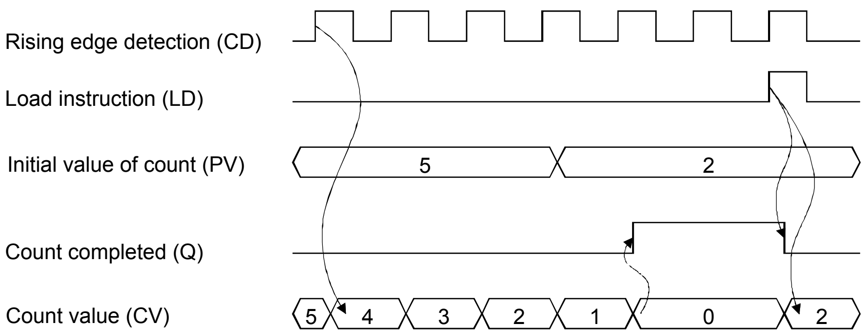
Function overview: Down-counter. When the input variable CD changes from FALSE to TRUE, current value-1 (Count value) will be output.

Function/FB classification name: Counter FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	CD	Input variable	BOOL	Rising edge detection
	LD	Input variable	BOOL	Load instruction
	PV	Input variable	INT	Initial value of counter
Output	Q	Output variable	BOOL	Count completed
	CV	Output variable	INT	Count value

Function

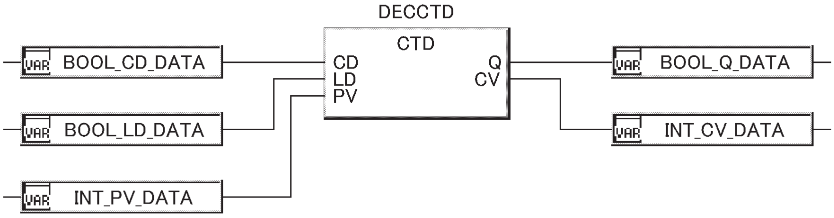
Item	Contents
Operation processing	(1) If the input variable CD changes from FALSE to TRUE, the current value-1 (count value) will be output from output variable CV. The initial value of counter is the input value of input variable PV. (2) When output variable CV becomes 0, TRUE will be output from output variable Q and the count down will be stopped. When output variable Q is TRUE, 0 will be output from output variable CV. (CV is kept) (3) When the input variable LD is TRUE, the output of output variable Q will be FALSE. The initial value of counter will be output from output variable CV (input value of PV) (4) When the input variable CD does not changes from FALSE to TRUE (TRUE→TRUE, TRUE→FALSE or FALSE →FALSE), the output of output variable CV will be the previous value. (5) If the input value of input variable PV is changed in count, when the input variable LD is TRUE next time, the changed input value of input variable PV is valid. (6) The input value of input variable CD, LD should be BOOL type data. (Default value: FALSE) (7) The input value of input variable PV should be INT type data value within the range of 0 to 32767. (Default value: 0) 

Error

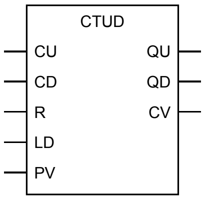
There are no operation errors caused by CTD.

Program Example

(1) Basic program example



5.3.3 Up-down-counter (CTUD)

FB	FBD parts
CTUD	

Function overview: Up-down-counter. When the input variable CU changes from FALSE to TRUE, current value+1 (Count value) will be output. When the input variable CD changes from FALSE to TRUE, current value-1 (Count value) will be output.

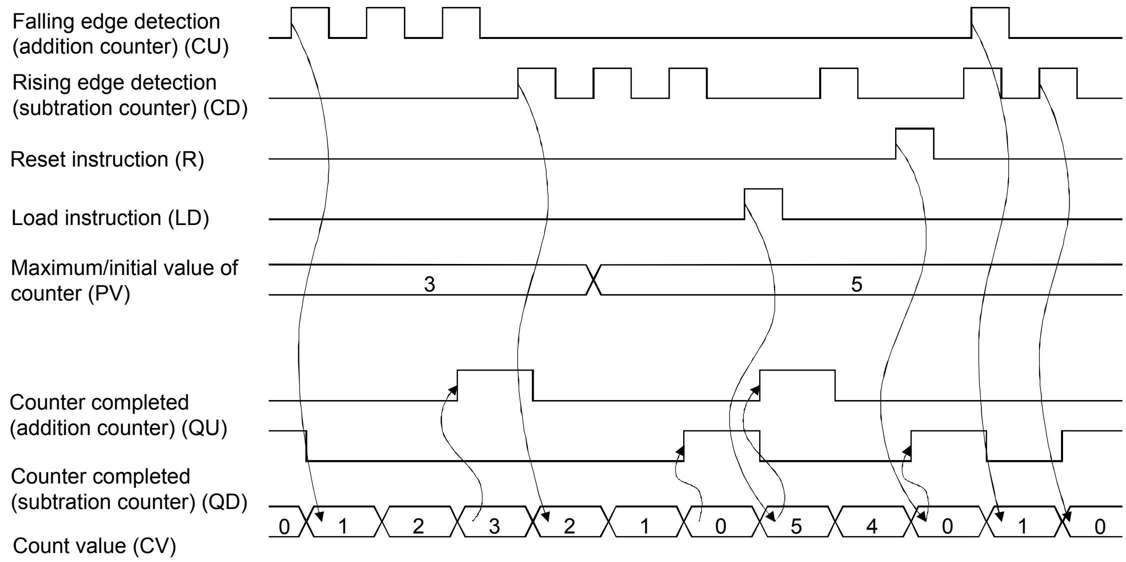
Function/FB classification name: Counter FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	CU	Input variable	BOOL	Rising edge detection (up-counter)
	CD	Input variable	BOOL	Rising edge detection (down-counter)
	R	Input variable	BOOL	Reset instruction
	LD	Input variable	BOOL	Load instruction
	PV	Input variable	INT	Maximum/initial value of counter
Output	QU	Output variable	BOOL	Count completed (up-counter)
	QD	Output variable	BOOL	Count completed (down-counter)
	CV	Output variable	INT	Count value

Function

Item	Contents
Operation processing	(1) Up-counter (a) When the input variable CU changes from FALSE to TRUE, current value+1 (Count value) will be output from output variable CV. (b) When the output variable CV reaches the input value of input variable PV, TRUE will be output from output variable QU and counter up will be stopped. When the output variable QU is TRUE, the input value of input variable PV will be output from output variable CV. (CV is kept) (c) When the input variable R is TRUE, FALSE will be output from output variable QU and 0 will be output from output variable CV. (d) When the input variable CU does not change from FALSE to TRUE (TRUE→TRUE, TRUE→FALSE, FALSE→FALSE), the previous value will be output from output variable CV. (e) The input value of input variables CU and R should be BOOL type data. (Default value: FALSE) (f) The input value of input variable PV should be INT type data within the range of 0 to 32767. (Default value: 0) (2) Down-counter (a) When the input variable CD changes from FALSE to TRUE, current value-1 (Count value) will be output from output variable CV. The initial value of counter is the input value of input variable PV. (b) When the output variable CV becomes 0, TRUE will be output from output variable QD and counter up will be stopped. When the output variable QD is TRUE, 0 will be output from output variable CV. (CV is kept) (c) When the input variable LD is TRUE, FALSE will be output from output variable QD and initial value (input value of PV) of counter will be output from output variable CV. (d) When the input variable CD does not change from FALSE to TRUE (but TRUE→TRUE, TRUE→FALSE or FALSE→FALSE), the previous value will be output from output variable CV. (e) If the input value of input variable PV is changed in counter when input variable LD is TRUE next time, the changed input value of input variable PV is valid. (f) The input value of input variables CD and LD are BOOL type data value. (Default value: FALSE) (g) The input value of input variable PV is INT type data value whose range is 0 to 32767. (Default value: 0)

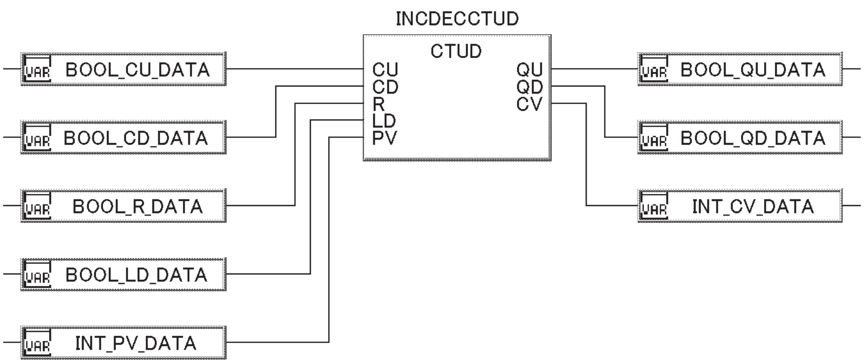
Item	Contents
Operation processing	(3) When TRUE is input from input variables CU and CD simultaneously, CU has the priority and current value+1(Count value) will be output from output variable CV. (4) When TRUE is input into input variables R and LD simultaneously, R has the priority and FALSE will be output from output variable QU and 0 will be output from output variable CV. 

Error

There are no operation errors caused by CTUD.

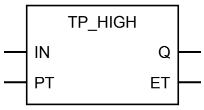
Program Example

(1) Basic program example



5.4 Timer FB

5.4.1 Pulse Timer (High-speed Timer) (TP_HIGH)

FB	FBD parts
TP_HIGH	

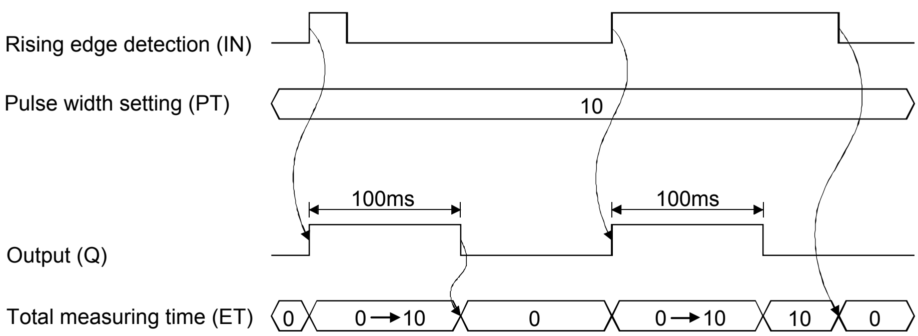
Function overview: Pulse timer (high-speed timer). When the input variable IN changes from FALSE to TRUE, TRUE will be output from output variable Q for the time set by input variable PT.

Function/FB classification name: Timer FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	IN	Input variable	BOOL	Rising edge detection
	PT	Input variable	INT	Pulse width setting
Output	Q	Output variable	BOOL	Output
	ET	Output variable	INT	Total measuring time

Function

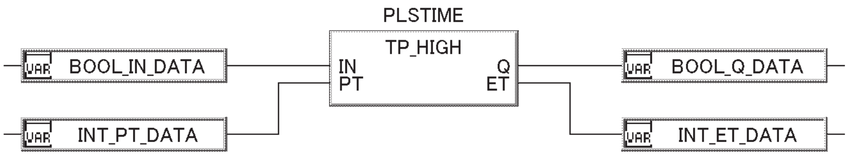
Item	Contents
Operation processing	(1) When the input variable IN changes from FALSE to TRUE, TRUE will be output from output variable Q for the time set by input variable PT. Output variable Q has nothing to do with the status of input variable IN, TRUE is output for the time set by input variable PT. (2) The output of output variable ET is total measuring time. (Output the time when output variable Q is TRUE) When output variable Q changes from TRUE to FALSE and FALSE is input into input variable IN, the output variable ET is reset. (Zero clear) (3) The input value of input variable IN should be BOOL type data. (Default value: FALSE) (4) The input value of input variable PT should be INT type data whose range is 1 to 32767. (Default value: 0) The input value of input variable PT is set as follows: Pulse width time=pulse width setting(PT)×high-speed timer interval (*1) *1 High-speed timer interval can be set in unit of 0.1ms within the range of 0.1ms to 100ms. (Default value: 10ms) The settings can be performed by selecting [Parameter] → [PLC parameter] → [PLC system setting] → "Timer limit setting" of GX application. (Example) In the following sample, the pulse width is set as 10 (when high-speed timer interval is 10ms) 

POINT

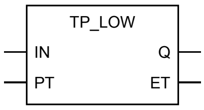
The output variables Q and ET are output in the execution of TP_HIGH.
After the pulse time, the time error when output variable Q changes to FALSE is not greater than a TP_HIGH execution cycle.

Error
There are no operation errors caused by TP_HIGH.

Program Example
(1) Basic program example



5.4.2 Pulse Timer (Low-speed Timer) (TP_LOW)

FB	FBD parts
TP_LOW	

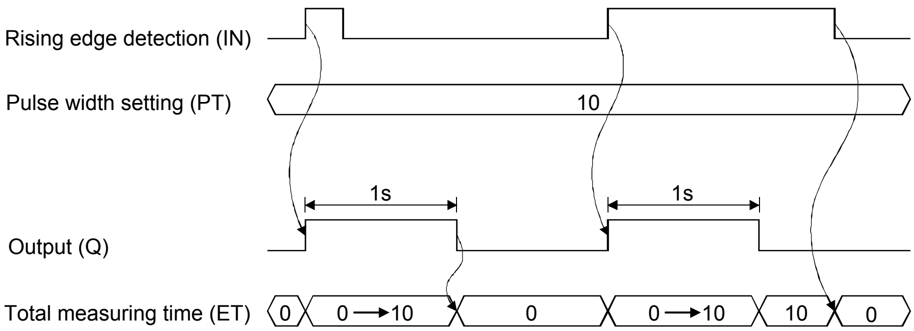
Function overview: Pulse timer (low-speed timer). When the input variable IN changes from FALSE to TRUE, TRUE will be output from output variable Q for the time set by input variable PT.

Function/FB classification name: Timer FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	IN	Input variable	BOOL	Rising edge detection
	PT	Input variable	INT	Pulse width setting
Output	Q	Output variable	BOOL	Output
	ET	Output variable	INT	Total measuring time

Function

Item	Contents
Operation processing	(1) When the input variable IN changes from FALSE to TRUE, TRUE will be output from output variable Q for the time set by input variable PT. Output variable Q has nothing to do with the status of input variable IN, TRUE is output for the time set by input variable PT. (2) The output of output variable ET is total measuring time. (Output the time when output variable Q is TRUE) When output variable Q changes from TRUE to FALSE and FALSE is input into input variable IN, the output variable ET is reset. (Zero clear) (3) The input value of input variable IN should be BOOL type data. (Default value: FALSE) (4) The input value of input variable PT should be INT type data whose range is 1 to 32767. (Default value: 0) The input value of input variable PT is set as follows: Pulse width time=pulse width setting (PT) $\times$100ms(low-speed timer interval (*1)) *1 Low-speed timer interval must be 100ms if FBD program is used. (Example) In the following sample, the pulse width is set as 10 (when low-speed timer interval is 100ms) 

IMPORTANT

If FBD program is used, please do not change the default interval (100ms) of low-speed timer. If it is changed, it may work abnormally.

POINT

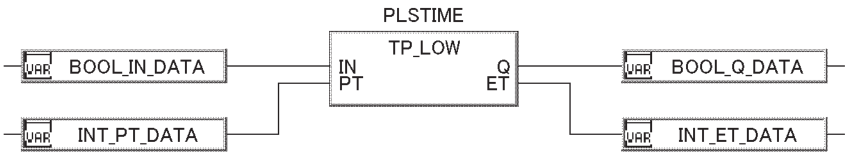
The output variables Q and ET are output in the execution of TP_LOW.
After the pulse time, the time error when output variable Q changes to FALSE is not greater than a TP_LOW execution cycle.

Error

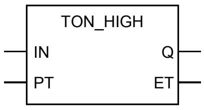
There are no operation errors caused by TP_LOW.

Program Example

(1) Basic program example



5.4.3 ON Delay Timer (High-speed Timer) (TON_HIGH)

FB	FBD parts
TP_HIGH	

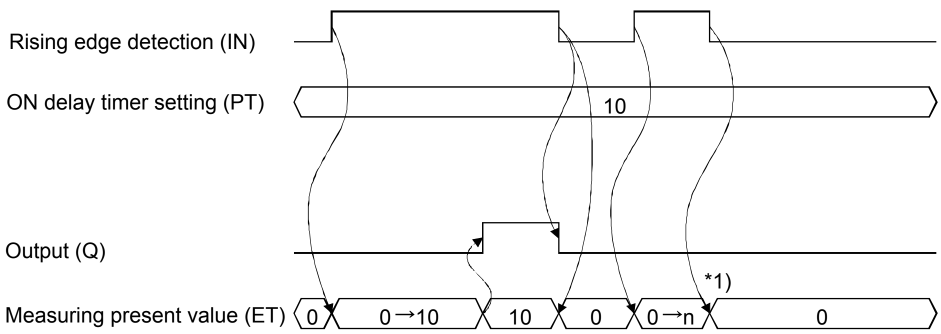
Function overview: ON delay timer (high-speed timer). It starts measurement when input variable IN changes from FALSE to TRUE. When the measuring time reaches the time (output value of ET≧input value of PT) set by the input variable PT, TRUE will be output from output variable Q.

Function/FB classification name: Timer FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	IN	Input variable	BOOL	Rising edge detection
	PT	Input variable	INT	ON delay timer setting
Output	Q	Output variable	BOOL	Output
	ET	Output variable	INT	Measuring current value

Function

Item	Contents
Operation processing	(1) It starts measurement when input variable IN changes from FALSE to TRUE. When the measuring time reaches the time (output value of ET≧input value of PT) set by the input variable PT, TRUE will be output from output variable Q. The output of output variable Q will be TRUE unless the input variable IN changes from TRUE to FALSE. (2) The output of output variable ET is the current value of measuring time. The data of output variable ET does not change after output variable becomes TRUE. ET is reset (Zero clear) when input variable IN changes from TRUE to FALSE. (3) The output variable ET is reset (timer stops) if input variable IN becomes FALSE before output variable Q becomes TRUE. (Illustrated in the following figure *1) (4) The input value of input variable IN is BOOL type data. (Default value: FALSE) (5) The input value of input variable PT is INT type data whose range is 1 to 32767. (Default value: 0) The input value of input variable PT should be set with the following value: ON delay timer time = OFF delay timer setting (PT) ×high-speed timer limit (*2) *2 High-speed timer limit can be set within the range of 0.1ms to 100ms in unit of 0.1ms. (Default value: 10ms) The settings can be performed by selecting [Parameter] → [PLC parameter] → [PLC system] → "Timer limit setting" of GX application. (Example) In the following case, the ON delay timer setting is 10 (when the high-speed timer limit is 10ms) 

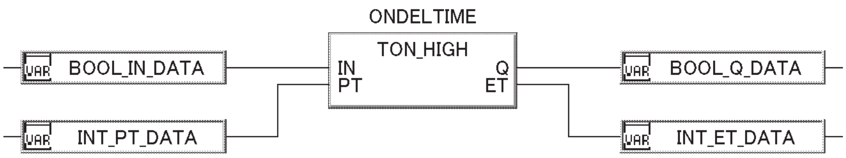
> **POINT**
>
> The output variables Q and ET are output in the execution of TON_HIGH.
> After the pulse time, the time error when output variable Q changes to FALSE is not greater than a TP_HIGH execution cycle.

Error

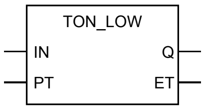
There are no operation errors caused by TON_HIGH.

Program Example

(1) Basic program example



5.4.4 ON Delay Timer (Low-speed Timer) (TON_LOW)

FB	FBD parts
TON_LOW	

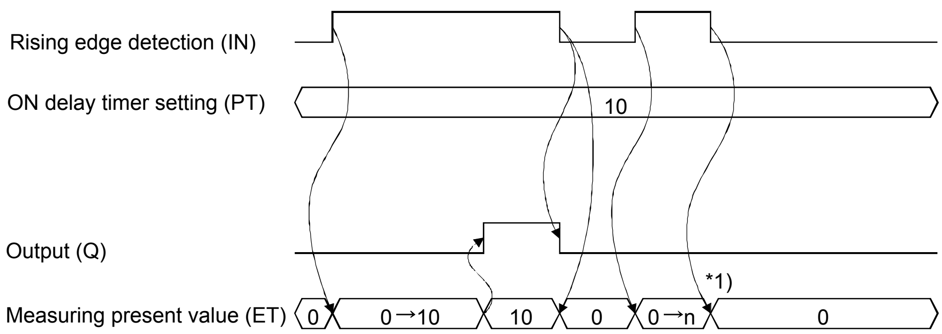
Function overview: ON delay timer (low-speed timer). It starts measurement when input variable IN changes from FALSE to TRUE. When the measuring time reaches the time (output value of ET≧input value of PT) set by the input variable PT, TRUE will be output from output variable Q.

Function/FB classification name: Timer FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	IN	Input variable	BOOL	Rising edge detection
	PT	Input variable	INT	ON delay timer setting
Output	Q	Output variable	BOOL	Output
	ET	Output variable	INT	Measuring current value

Function

Item	Contents
Operation processing	(1) It starts measurement when input variable IN changes from FALSE to TRUE. When the measuring time reaches the time (output value of ET≧input value of PT) set by the input variable PT, TRUE will be output from output variable Q. (2) The output of output variable ET is the current value of measuring time. (3) The output variable ET is reset (timer stops) if input variable IN becomes FALSE before output variable Q becomes TRUE. (Illustrated in the following figure *1) (4) The input value of input variable IN is BOOL type data value. (Default value: FALSE) (5) The input value of input variable PT is INT type data whose range is 1 to 32767. (Default value: 0) The input value of input variable PT should be set with the following value: ON delay timer time=OFF delay timer setting (PT) × 100ms(low-speed timer limit (*2)) *2 Low-speed timer limit is fixed in 100ms if FBD program is used. (Example) In the following case, the ON delay timer setting is 10 (when the low-speed timer limit is fixed in 100ms) 

IMPORTANT

Please do not change the default value 100ms of low-speed timer limit when using FBD program. Otherwise, it cannot work normally.

POINT

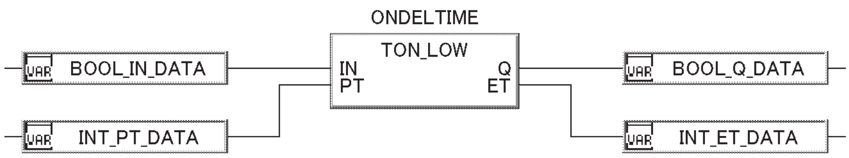
The output variables Q and ET are output in the execution of TON_LOW.
After the pulse time, the time error when output variable Q changes to FALSE is not greater than a TON_LOW execution cycle.

Error

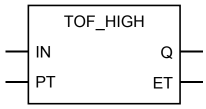
There are no operation errors caused by TON_LOW.

Program Example

(1) Basic program example



5.4.5 OFF Delay Timer (High-speed Timer) (TOF_HIGH)

FB	FBD parts
TOF_HIGH	

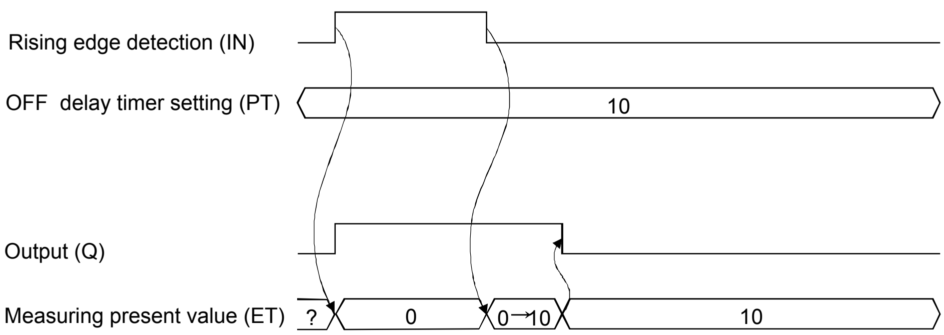
Function overview: OFF delay timer (high-speed timer). TRUE will be output from the output variable Q when the input variable IN changes from FALSE to TRUE. It starts measurement when the input variable IN changes from TRUE to FALSE, and TRUE is output from the output variable Q until the measurement time reaches the time set by the input variable PT (output value of ET ≧ input value of PT).

Function/FB classification name: Timer FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	IN	Input variable	BOOL	Rising edge detection
	PT	Input variable	INT	OFF delay timer setting
Output	Q	Output variable	BOOL	Output
	ET	Output variable	INT	Measuring current value

Function

Item	Contents
Operation processing	(1) TRUE will be output from the output variable Q when the input variable IN changes from FALSE to TRUE. It starts measurement when the input variable IN changes from TRUE to FALSE, and TRUE is output from the output variable Q until the measurement time reaches the time set by the input variable PT (output value of ET > input value of PT). (2) Current value of measuring time will be output from output variable ET when input variable IN changes from TRUE to FALSE. Output variable ET keeps its output when output variable Q changes from TRUE to FALSE, while reset when input variable IN changes from FALSE to TRUE. (3) The input value of input variable IN is BOOL type data. (Default value: FALSE) (4) The input value of input variable PT is INT type data whose range is 1 to 32767. (Default value: 0) The input value of input variable PT should be set with the following value: OFF delay timer time=OFF delay timer setting (PT) × high-speed timer limit (*1) *1 High-speed timer limit can be changed within the range of 0.1ms to 100ms in unit of 0.1ms. (Default value: 10ms) The settings can be performed by selecting [Parameter] → [PLC parameter] → [PLC system] → "Timer limit setting" of GX application. (Example) In the following case, the OFF delay timer setting is 10 (when the high-speed timer limit is 10ms) 

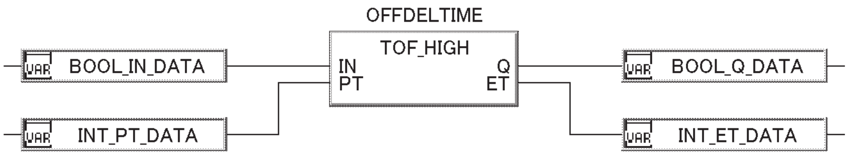
POINT
The output variables Q and ET are output in the execution of TOF_HIGH. After the OFF delay time, the time error when output variable Q changes to TRUE is not greater than a TOF_HIGH execution cycle.

Error

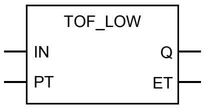
There are no operation errors caused by TOF_HIGH.

Program Example

(1) Basic program example



5.4.6 OFF Delay Timer (Low-speed Timer) (TOF_LOW)

FB	FBD parts
TOF_LOW	

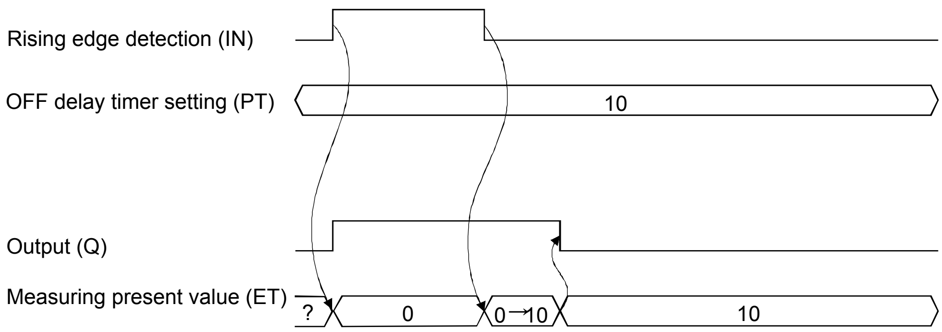
Function overview: OFF delay timer (low-speed timer). TRUE will be output from output variable Q when input variable IN changes from FALSE to TRUE. Measuring of output variable Q starts when input variable IN changes from TRUE to FALSE. Before the measuring time reaches the time (output value of ET≧input value of PT) set by input variable PT, TRUE will be output from output variable Q.

Function/FB classification name: Timer FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	IN	Input variable	BOOL	Rising edge detector
	PT	Input variable	INT	OFF delay timer setting
Output	Q	Output variable	BOOL	Output
	ET	Output variable	INT	Measuring current value

Function

Item	Contents
Operation processing	(1) TRUE will be output from output variable Q when input variable IN changes form FALSE to TRUE. The measurement of output variable starts from the time when input variable IN changes from TRUE to FALSE. when the measuring time reaches the time (value of ET≧input value of PT) set by input variable PT, TRUE will be output from output variable (2) Current value of measuring time will be output from output variable ET when input variable IN changes from TRUE to FALSE. Output variable ET keeps its output when output variable Q changes from TRUE to FALSE, while reset when input variable IN changes from FALSE to TRUE. (3) The input value of input variable IN is BOOL type data. (Default value: FALSE) (4) The input value of input variable PT is INT type data whose range is 1 to 32767. (Default value: 0) The input value of input variable PT should be set with the following value: OFF delay timer time=OFF delay timer setting (PT) × 100ms (low-speed timer limit (*1)) *1 Low-speed timer limit is fixed in 100ms if FBD program is used. (Example) In the following case, the OFF delay timer setting is 10 (when the low-speed timer limit is fixed in100ms) 

IMPORTANT

Please do not change the default value 100ms of low-speed timer limit when using FBD program. Otherwise, it cannot work normally.

POINT

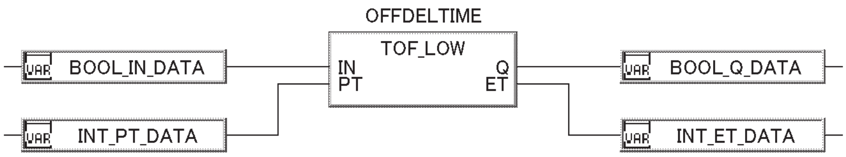
The output variables Q and ET are output in the execution of TOF_LOW.
After the ON delay time, the time error when output variable Q changes to FALSE is not greater than a TOF_LOW execution cycle.

Error

There are no operation errors caused by TOF_LOW.

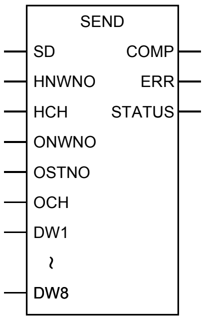
Program Example

(1) Basic program example



5.5 Communication Control FB

5.5.1 Sending Data to PLC CPUs of Other Stations (SEND)

FB	FBD parts
SEND	

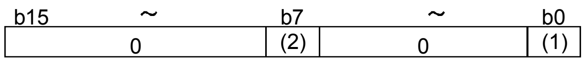
Function overview: Send data to PLC CPUs of other stations. (Performs the same processing as JP.SEND instruction of PLC)

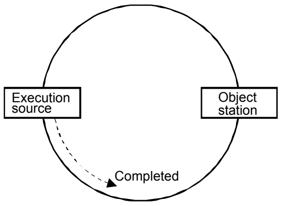
Function/FB classification name: Communication control FB

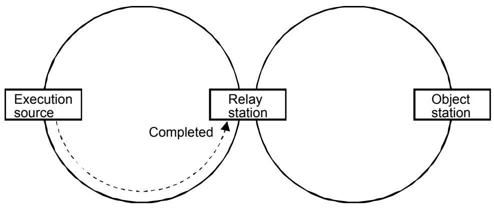
Input and Output Pins

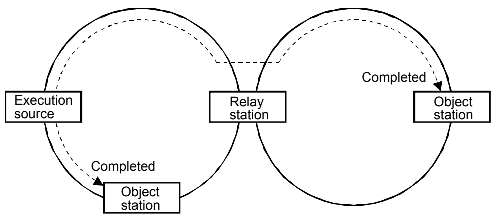
Pin	Variable name	Variable type	Data type	Contents
Input	SD	Input variable	BOOL	Sending instructions
	HNWNO	Input variable	INT	Network No. of host station 1 to 239 : network No. 254 : network specified by valid module during other station access
	HCH	Input variable	INT	Channel used by host station 1 to 8: Channel
	ONWNO	Input variable	INT	Network No. of object station 1 to 239 : network No. 254 : network specified by valid module during other station access
	OSTNO	Input variable	INT	Object station number 1 to 64: station number of object station 81H to 89H: all stations of this group number (When the execution type of public variable EXETYPE is "0: No arrival confirmation" it is settable.) FFH: all stations (broadcasting) of object network No (host station excluded) (When the execution type of public variable EXETYPE is "0: No arrival confirmation" it is settable.)
	OCH	Input variable	INT	Object station storage channel 1 to 8: Channel
	DW1 to DW8	Input variable	DWORD	Transmitting data Set transmitting data DW1 to DW8.
Output	COMP	Output variable	BOOL	Transmission completed TRUE: Completed FALSE: uncompleted
	ERR	Output variable	BOOL	Transmission completed status TRUE: Completed abnormally FALSE: completed normally
	STATUS	Output variable	WORD	Completion status 0H: normal other than 0H: abnormal (error code)

Public Variable

<table>
<tr><th>Variable name</th><th>Data type</th><th>Contents</th><th>Range</th><th>Storage</th></tr>
<tr><td>EXETYPE</td><td>WORD</td><td>[Execution/abnormal completion type]


(1) Execution type (0 bit)
0: No arrival confirmation
When the object station is in local network ...... Transmission is completed when host station sends data.


When the object station is in other network ..... Transmission is completed when the relay station in local network receives data.


1: With arrival confirmation
Transmission is completed when the data is stored in the specified channel of object station.


(2) Completion type when an error occurs (7th bit)
To set whether timer data is stored in the public variable CLOCKSET to ERRST when an error occurs.
0: Not set timer data.
1: Set timer data.</td><td>0000H
0001H
0080H
0081H</td><td>User</td></tr>
<tr><td>RETRY</td><td>INT</td><td>[Re-transmission (retry) times]
It is valid when the execution type of public variable EXETYPE is "1: With arrival confirmation".
(1) When instruction is under execution: 0 to 15 (times)
To set the re-transmission times when transmission is not completed in the time specified by public variable ARRTIME.
(2) When instruction is completed: 0 to 15 (times)
To store the re-transmission times.</td><td>0 to 15</td><td>User
system</td></tr>
</table>

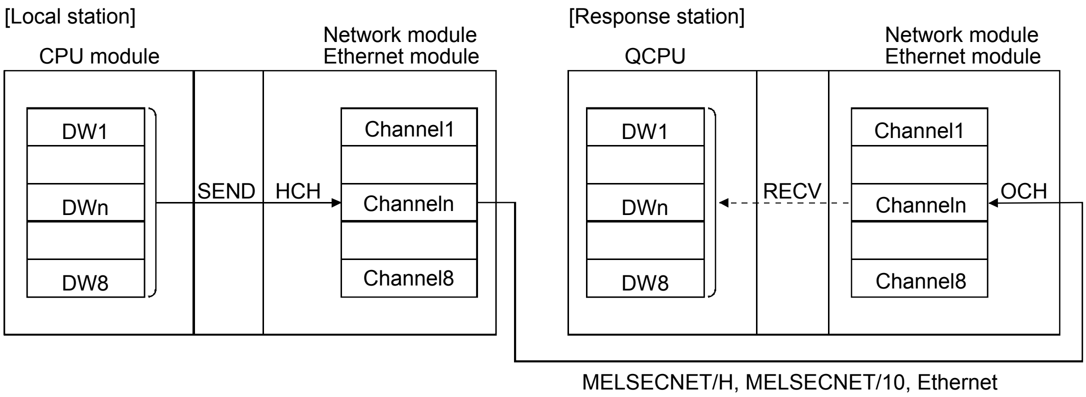
Variable name	Data type	Contents	Range	Storage
ARRTIME	INT	[Arrival WDT Time] It is valid when the execution type of public EXETYPE is "1: With arrival confirmation". If it is uncompleted in the Arrival WDT time, it will be re-transmitted in times specified by public variable RETRY. (1) Set the WDT time till the instruction is completed via MELSECNET/H or MELSECNET/10 network system. 0 : 10 seconds 1 to 32767: 1 to 32767 seconds	0 to 32767	User
		(2) Set the WDT time above TCP retry timer value till the instruction is completed via Ethernet interface module. 0 to TCP transmission retry timer value: WDT time is equivalent to TCP transmission retry timer value. (TCP transmission retry timer value+1) to 16383: WDT time (unit: second)	1 to 16383 0 to transmission retry time value	
CLOCKSET	INT	[Clock set flag] To store the valid/invalid status of public variable CLOCKDATA1 to ERRST data. 0: invalid 1:valid	—	System (*1)
CLOCKDATA1	WORD	[Clock data (only when errors occur)] High-order 8 bit: month (01H to 12H), low-order 8 bit: year (00H to 99H) Rightmost two digits.	—	System (*1)
CLOCKDATA2	WORD	[Clock data (only when errors occur)] High-order 8 bit: hour (00H to 23H), low-order 8 bit: day (01H to 31H)	—	System (*1)
CLOCKDATA3	WORD	[Clock data (only when errors occur)] High-order 8 bit: second (00H to 59H), low-order 8 bit: minute (00H to 59H)	—	System (*1)
CLOCKDATA4	WORD	[Clock data (only when errors occur)] High-order 8 bit: year (00H to 99H) leftmost 2 digits, low-order 8 bit: Week (00H(Sunday) to 06H(Saturday))	—	System (*1)
ERRNW	INT	[Station No. where the error was detected] To store the network No. of detected abnormal station. However, if the completion status of output variable STATUS is "Channel in use (F7C1H or C085H)", it will not be stored. 1 to 239 (network No.)	—	System (*1)
ERRST	INT	[Station No. where the error was detected.] To store the station No. of detected abnormal station. However, if the completion status of output variable STATUS is "Channel in use (F7C1H or C085H)", it will not be stored. 1 to 64 (station No.)	—	System (*1)
CHGSYS	BOOL	It turns TRUE in the new control system immediately after system switching, and then returns FALSE at the first FB execution, in redundant system. (It turns TRUE/FALSE for each execution.)	TRUE, FALSE	System (*1)

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

Restrictions on redundant system operation and the applicable corrective action

Restrictions	Corrective action
If tracking of signal flow has not been executed, data transmission will not be executed even when the SEND instruction SD is turned from FALSE to TRUE at the first FB execution in the new control system immediately after system switching.	Input "FALSE ⟶ TRUE" in the SEND instruction SD and execute data transmission in the new control system immediately after system switching, as necessary. (Whether system switching has been performed or not can be found by using the public variable CHGSYS.)
If the system is switched before TRUE is output to the transmission completion COMP when "FALSE ⟶ TRUE" has been input in the SEND instruction SD and executed, the data transmission may not be executed. In addition, the COMP may not turn TRUE in the new control system.	
This FB cannot be used for an Ethernet module mounted to a redundant type extension base unit of Redundant CPU.	Mount an Ethernet module to a main base unit for execution of the FB.

Function

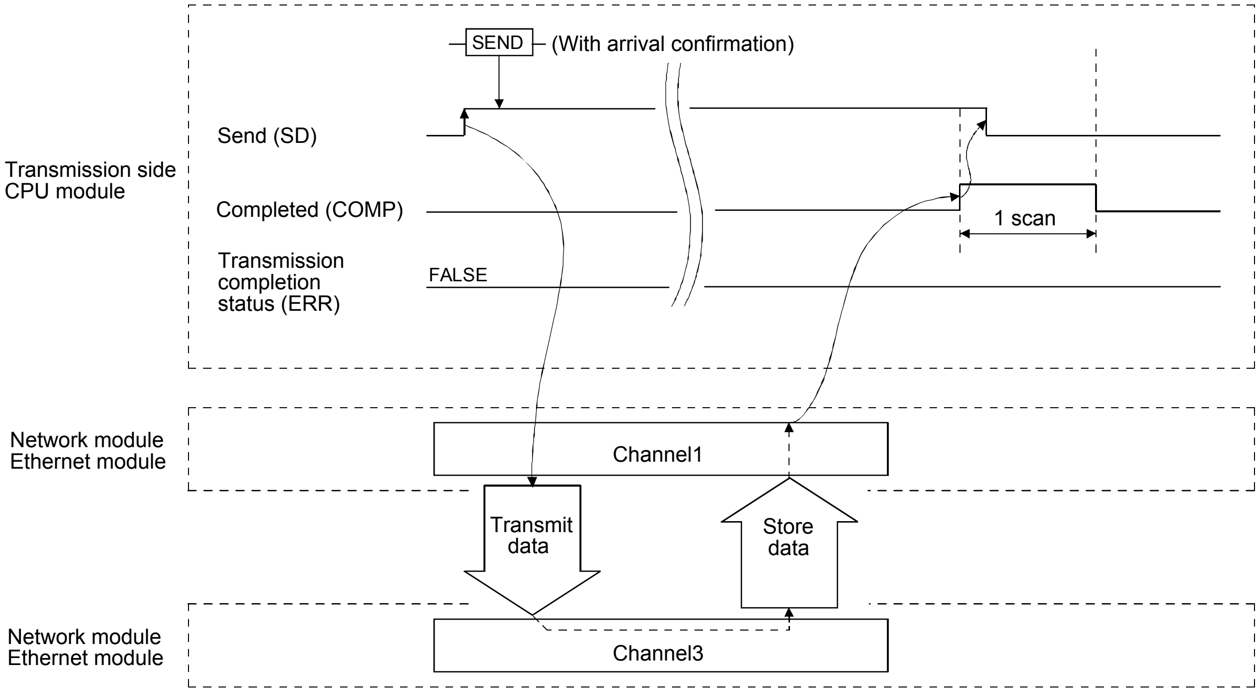
Item	Contents
Operation processing	(1) Transmit the input data of input variable DW1 to DW8 to the MELSECNET/H or MELSECNET/10 network module or Ethernet interface module specified by the input value of input variable ONWNO and OSTNO when the input variable SD changes from FALSE to TRUE.  (a) When input variable SD changes from FALSE to TRUE, the transmission processing is performed only once. (b) The transmission data input into input variable DW1 to DW8 are stored in the channel specified by input variable HCH and transmitted. (c) The transmission data is stored in the channel specified by input variable OCH. (d) Read information data from object station with RECV instruction. (2) The information data can be transmitted not only to the connected stations of host station but also to the connected stations specified by MELSECNET/H or MELSECNET/10 or Ethernet. (3) To a single channel, communication control FB cannot be performed in more than two places simultaneously. If execution conditions in more than two places are satisfied simultaneously, the later instructions have to wait for the availability of channel as automatic handshakes are carried out in the channel. (4) The status of SEND instruction such as under execution, normal/abnormal completion can be identified by the output value of output variable. (a) SEND instruction completed. (Output variable COMP) TRUE will be output when SEND instruction is completed and FALSE will be output when SEND instruction is executed next time. (b) Status display when SEND instruction is completed (output variable ERR) TRUE/FALSE will be output to indicate the status when SEND instruction is completed. In normal completion : output FALSE. In abnormal completion : output TRUE and FALSE will be output when SEND instruction is executed next time. When SEND instruction is completed abnormally, TRUE will be output from output variable ERR and error code will be output from output variable STATUS. Please refer to the following manuals for error codes for confirmation/disposal of errors. <Error code> Lower than 4FFFH : <<QCPU User's Manual (Hardware Design/Maintenance and Inspection)>> C000H to : <<Q Corresponding Ethernet Interface Module User's Manual (Basic edition)>> F000H to : <<Q Corresponding MELSECNET/H Network System Reference Manual (PLC to PLC network)>>

POINT

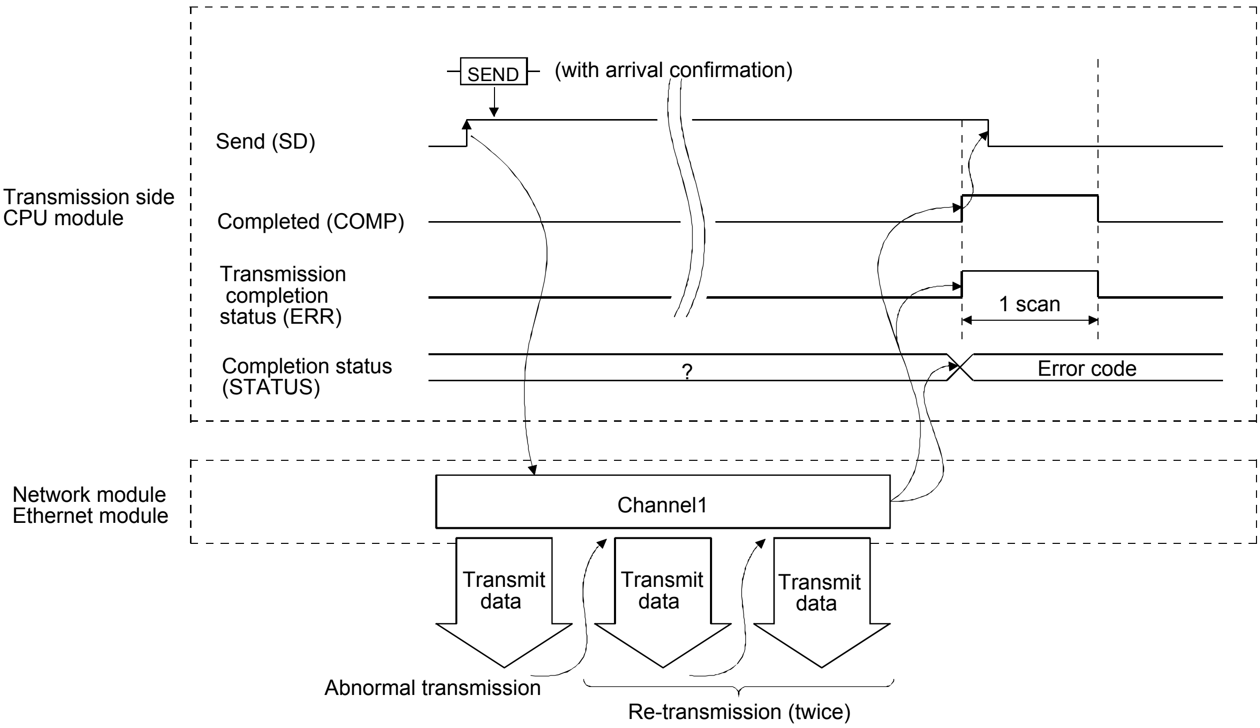
When the input variable SD becomes TRUE, the SEND type FB executes the JP. SEND instruction, which is a rise instruction, in the FB. Hence, send processing is not performed if the online change of the SEND type FB is executed with the input variable SD in a TRUE status. To perform send processing, change the input variable SD from FALSE to TRUE.

Instruction Execution Timing

(1) Completion



(2) Completion with errors



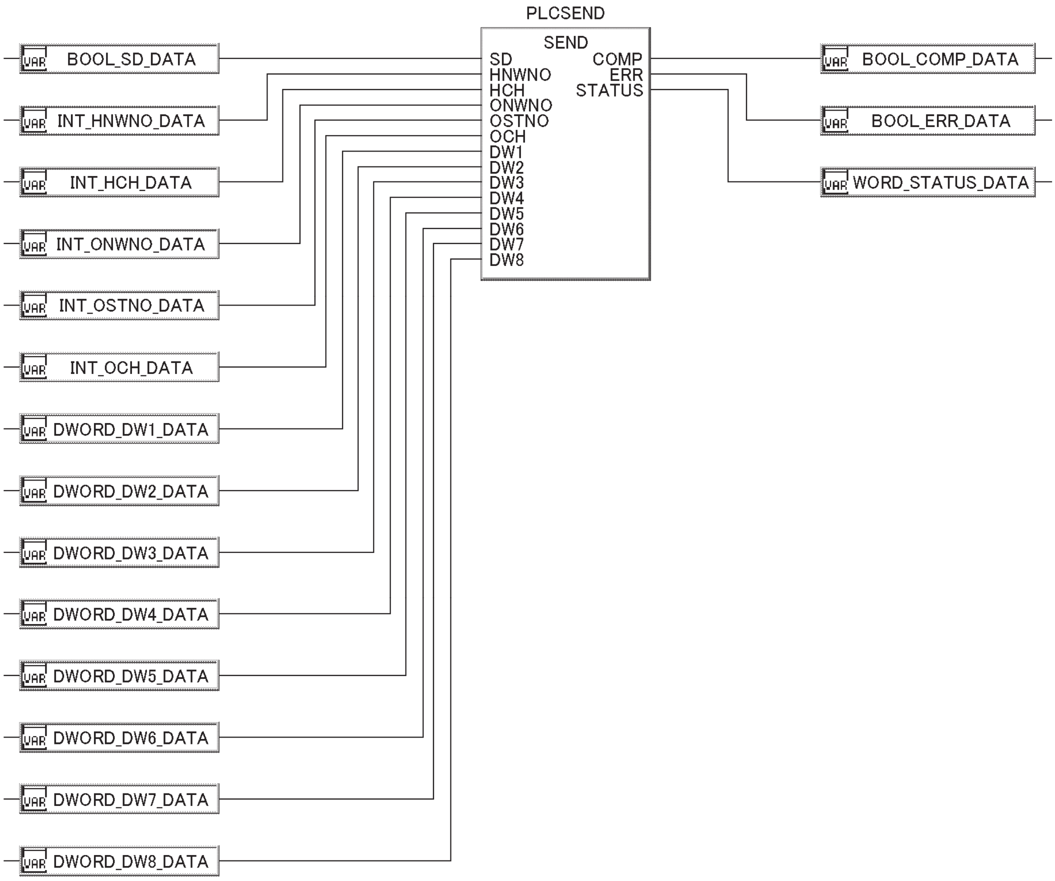
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

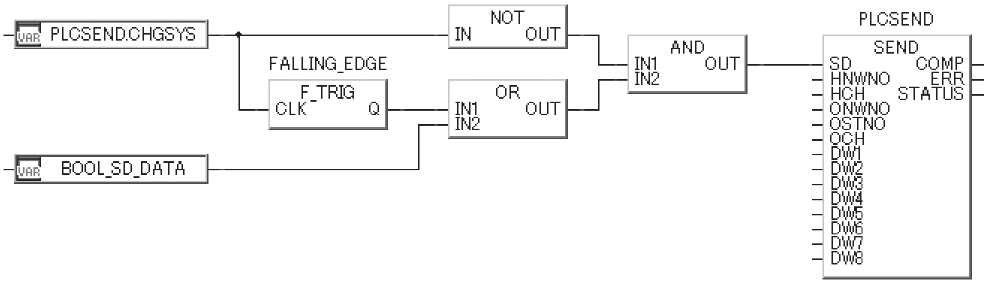
- When the input value of input variable HNWNO, HCH, ONWNO, OSTNO or OCH is out of range. (Error code: Refer to Appendix 2)
- Object station is not connected with the host station. (Error code: 4102)

Program Example

(1) Basic program example



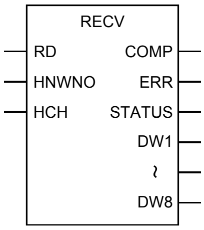
(2) The following shows an example of a program that re-executes the SEND type FB in redundant system by turning the input variable SD from FALSE to TRUE at system switching.



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB for setting initial value of communication control FB pasted on user-defined FB/Tag FB and reading/writing them in program.

5.5.2 Receiving Data from PLC CPUs of Other Stations (RECV)

FB	FBD parts
RECV	

Function overview: Receive data from other PLC CPUs. (To execute the equivalent processing with JP.RECV instruction of PLC program.)

Function/FB classification name: Communication control FB

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents
Input	RD	Input variable	BOOL	RECV instruction execution requesting flag
	HNWNO	Input variable	INT	Network No. of host station 1 to 239: network No. 254 : network specified in valid unit in accessing other stations
	HCH	Input variable	INT	Host station storage channel 1 to 8: Channel
Output	COMP	Output variable	BOOL	Reception completed. TRUE: Completed FALSE: uncompleted.
	ERR	Output variable	BOOL	Reception completed status TRUE: abnormal completion FALSE: normal completion.
	STATUS	Output variable	BOOL	Completion status 0H: normal Other than 0H: abnormal (error code)
	DW1 to DW8	Output variable	DWORD	Receiving data Output the received data DW1 to DW8 from corresponding channels.

Public Variable

Variable name	Data type	Contents	Range	Storage
EXETYPE	WORD	[Abnormal completion type] b15 ~ b8 b7 b6 ~ b0 0 ~ 0 (1) 0 ~ 0 (1) Abnormal completion type (the 7th bit) To set whether timer data is stored in the public variable CLOCKSET to ERRST when an error occurs. 0: not set timer data 1: set timer data	0000H 0080H	User
OCH	INT	[Channel used for object station] To store the channels used in transmission station. 1 to 8 (channel)	1 to 8	System (*1)
ONWNO	INT	[Network No. of object station] To store the network No. of transmission station. 1 to 239: network No.	1 to 239	System (*1)
OSTNO	INT	[Station No. of object station] To store the station No. of transmission stations. 1-64: station No. (receive from station with station No.) FFH: all stations (receive according to broadcasting)	1 to 64 FFH	System (*1)

Variable name	Data type	Contents	Range	Storage
ARRTIME	INT	[Arrival WTD time] (1) Set WDT time till the instruction is completed via MELSECNET/H or MELSECNET/10 network system. If the instruction is not completed within the WDT time, it will be regarded as abnormal completion. 1 : 10 seconds 1 to 32767 : 1 to 32767 seconds	0 to 32767	User
		(2) Set the WDT time above TCP retry timer value till the instruction is completed via Ethernet interface module. If the instruction is not completed within the WDT time, it will be regarded as abnormal completion. 0 to TCP transmission retry timer value: monitor time is equivalent to TCP transmission retry timer value. (TCP transmission retry timer value+1) to 16383: WDT time (unit: second)	1 to 16383 0 to TCP transmission retry timer value	
LENGTH	INT	[Length of received data] To store the received data that has been stored in DW1 to DW8. However, if the received data is more than the stored data of DW1 to DW8, the excess data will be lost in reading. 0 : no received data 1 to 480 : bit number of received data	0 to 480	System (*1)
CLOCKSET	INT	[Clock set flag] To store the valid/invalid status of public variable CLOCKDATA1 to ERRST data. 0: invalid 1:valid	—	System (*1)
CLOCKDATA1	WORD	[Clock data (only when errors occur)] High-order bit 8 bit: month (01_H to 12_H), low-order bit 8 bit: year (00_H to 99_H) Rightmost two digits.	—	System (*1)
CLOCKDATA2	WORD	[Clock data (only when errors occur)] High-order bit 8 bit: hour (00_H to 23_H), low-order bit 8 bit: day (01_H to 31_H)	—	System (*1)
CLOCKDATA3	WORD	[Clock data (only when errors occur)] High-order bit 8 bit: second (00_H to 59_H), low-order bit 8 bit: minute (00_H to 59_H)	—	System (*1)
CLOCKDATA4	WORD	[Clock data (only when errors occur)] High-order bit 8 bit: year (00H to 99H) leftmost 2 digits, low-order bit 8 bit: Week (00H(Sunday) to 06H(Saturday))	—	System (*1)
ERRNW	INT	[Detected abnormal network No.] To store the network No. of detected abnormal station. However, if the completion status of output variable STATUS is "Channel in use (F7C1H or C085H)", it will not be stored. 1 to 239 (network No.)	—	System (*1)
ERRST	INT	[Detected abnormal station No.] To store the station No. of detected abnormal station. However, if the completion status of output variable STATUS is "Channel in use (F7C1H or C085H)", it will not be stored. 1 to 64 (station No.)	—	System (*1)
CHGSYS	BOOL	It turns TRUE in the new control system immediately after system switching, and then returns FALSE at the first FB execution, in redundant system. (It turns TRUE/FALSE for each execution.)	TRUE, FALSE	System (*1)

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

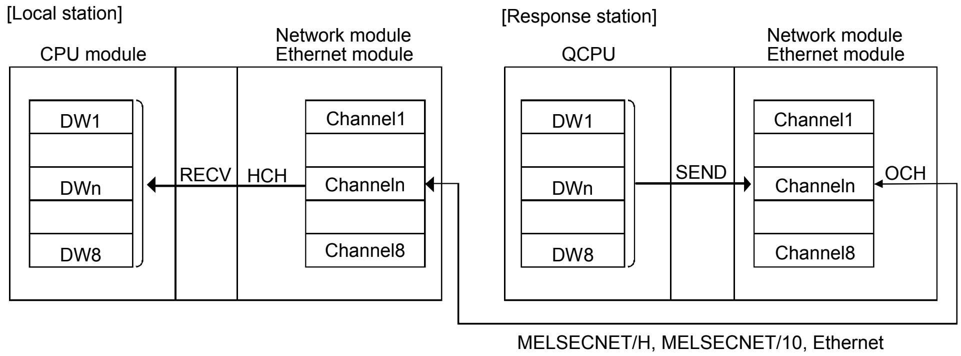
POINT

If the received data number exceeds the storage data number (maximum 16 words) of output variable DW1 to DW8, the excess data will be lost.

Restrictions on redundant system operation and the applicable corrective action

Restrictions	Corrective action
If tracking of signal flow has not been executed, data reception will not be executed even when the RECV instruction RD is turned from FALSE to TRUE at the first FB execution in the new control system immediately after system switching.	Input "FALSE ⟶ TRUE" in the RECV instruction RD and execute data reception in the new control system immediately after system switching, as necessary. (Whether system switching has been performed or not can be found by using the public variable CHGSYS.)
If the system is switched before TRUE is output to the receiving completion COMP when "FALSE ⟶ TRUE" has been input in the RECV instruction RD and executed, the data reception may not be executed. In addition, the COMP may not turn TRUE in the new control system.	
This FB cannot be used for an Ethernet module mounted to a redundant type extension base unit of Redundant CPU.	Mount an Ethernet module to a main base unit for execution of the FB.

Function

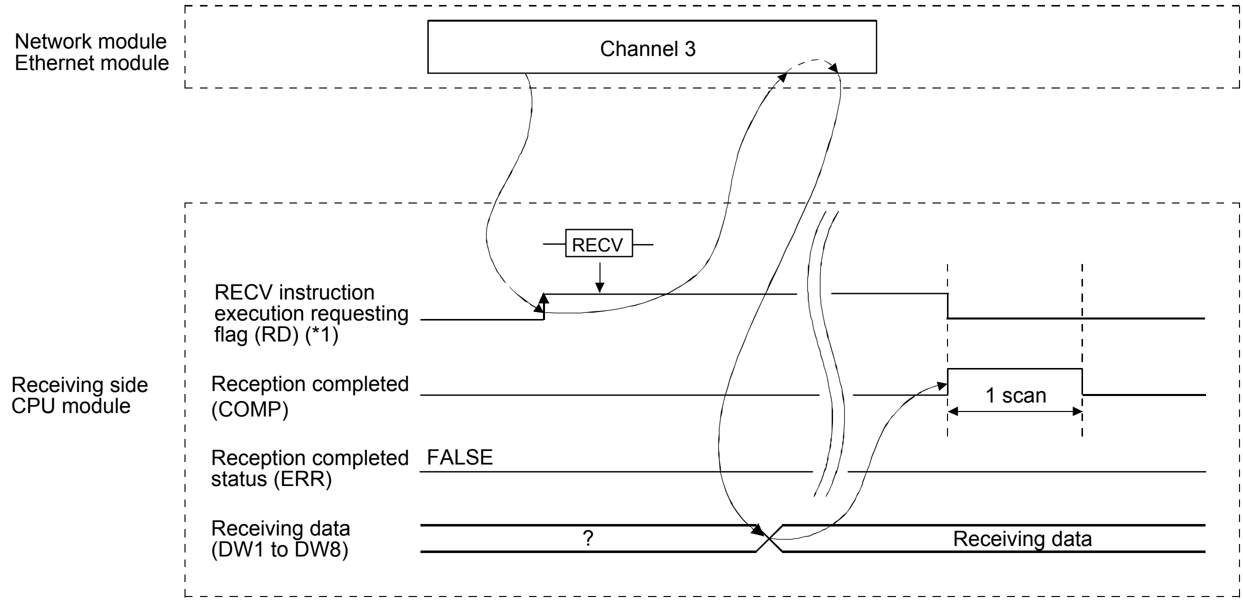
Item	Contents
Operation processing	(1) When input variable RD changes from FALSE to TRUE, the stored data in used channel that is specified by input value of input variable HCH will be output from output variable DW1 to DW8.  (a) When input variable RD changes from FALSE to TRUE, the transmission processing is performed only once. (b) In case of MELSENCNET/H and MELSECNET/10 network module, data transmission is executed by SEND instruction. Once data from transmission station are received, they will be stored in host station channels specified by the transmission station. Special link relay (SBA0 to A7) corresponding to the channels is switched on. The OFF→ON timing of input variable RD is marked by the special link relay (SBA0 to A7) and the received data is then read out from the received data storage channel. (c) In case of Ethernet interface module, SEND instruction is used to execute data transmission. Once data are received from transmission station, the received data will be stored in the host station channels. Corresponding bits of RECV instruction requesting execution area (address: 205) of Ethernet interface module buffer memory are switched on. The OFF→ON timing of input variable RD is marked by the special link relay (SBA0 to A7) and the received data is then read out from the received data storage channel. (2) To a single channel, communication control FB cannot be performed in more than two places simultaneously. If execution conditions in more than two places are satisfied simultaneously, the later instructions have to wait for the availability of channel as automatic handshakes are carried out in the channel. (3) In reading received data of a single channel by RECV instruction of communication control FB, it cannot be used along with RECVS instructions of sequent control (for interrupt program use). (4) The status of RECV instruction such as under execution, normal/abnormal completion can be identified by the output value of output variable. (a) RECV instruction completed. (Output variable COMP) TRUE will be output when RECV instruction is completed and FALSE will be output when RECV instruction is executed next time. (b) Status display when RECV instruction is completed (output variable ERR) TRUE/FALSE will be output to indicate the status when RECV instruction is completed. In normal completion : output FALSE. In abnormal completion : output TRUE and FALSE will be output when RECV instruction is executed next time. (5) When RECV instruction is completed abnormally, TRUE will be output from output variable ERR and error code will be output from output variable STATUS. Please refer to the following manuals for error codes for confirmation/disposal of errors. <Error code> Lower than 4FFFH : <<QCPU User's Manual (Hardware Design/Maintenance and Inspection)>> C000H to : Q Corresponding <<Ethernet Interface Module User's Manual (Basic edition)>> F000H to : Q Corresponding << MELSECNET/H Network System Reference Manual (PLC to PLC network)>>

POINT

When the input variable RD becomes TRUE, the RECV type FB executes the JP. RECV instruction, which is a rise instruction, in the FB. Hence, send processing is not performed if the online change of the RECV type FB is executed with the input variable RD in a TRUE status. To perform send processing, change the input variable RD from FALSE to TRUE.

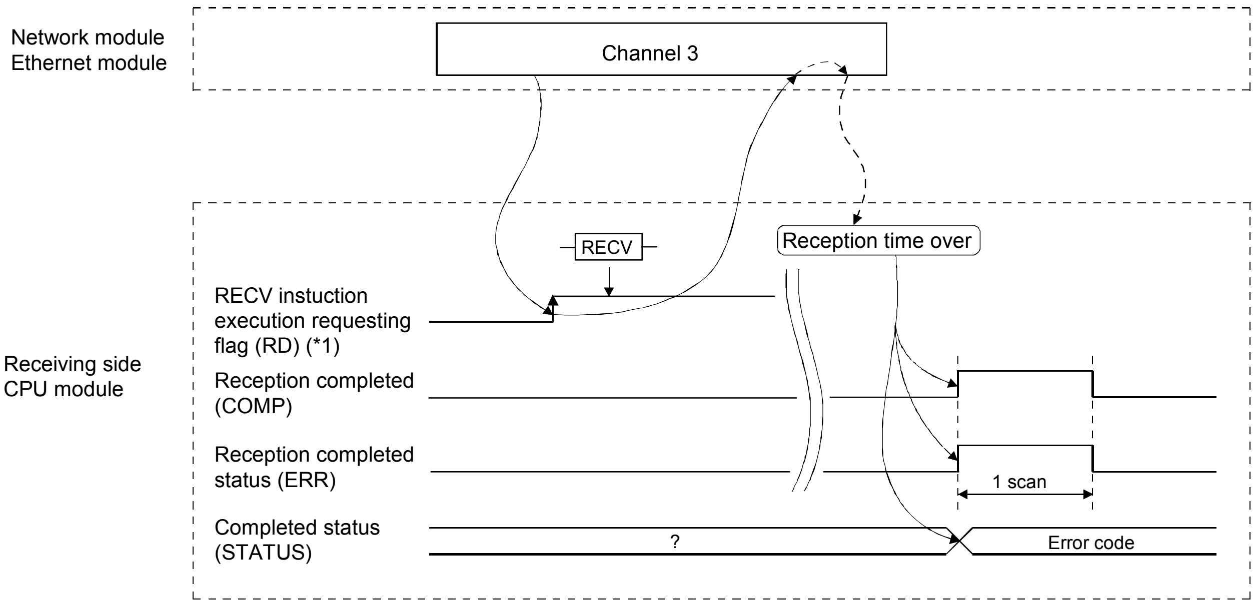
Instruction Execution Timing

(1) Completion



*1 The instruction execution requesting flag (RD) of channel 3 for receiving side use is shown as follows:
In case of network module : SB2 of special link relay
In case of Ethernet module: RECV instruction execution request area bit 2 of buffer memory(address: 205)

(2) Abnormal completion



*1 The instruction execution requesting flag (RD) of channel 3 for receiving side use is shown as follows:
In case of network module : SB2 of special link relay.
In case of Ethernet module: RECV instuction execution requesting area bit 2 of buffer memory (address: 205).

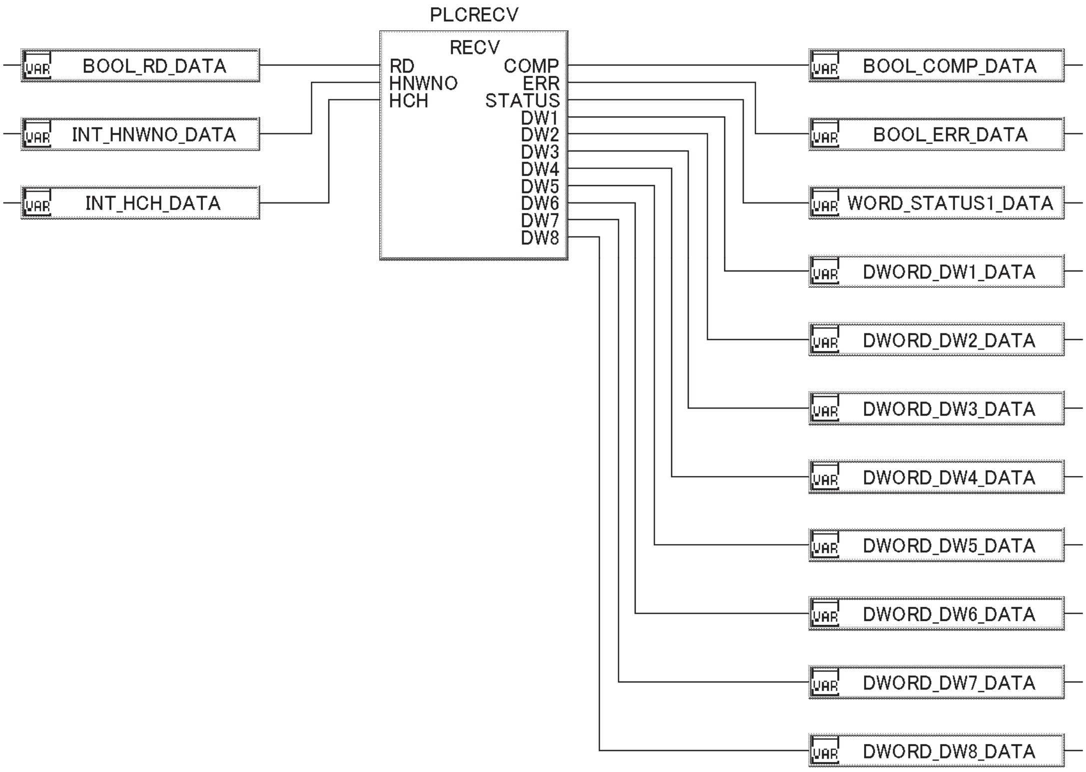
Error

Error may occur in the following cases, the error codes will be displayed in FBD program diagnostics screen of PX Developer programming tool.

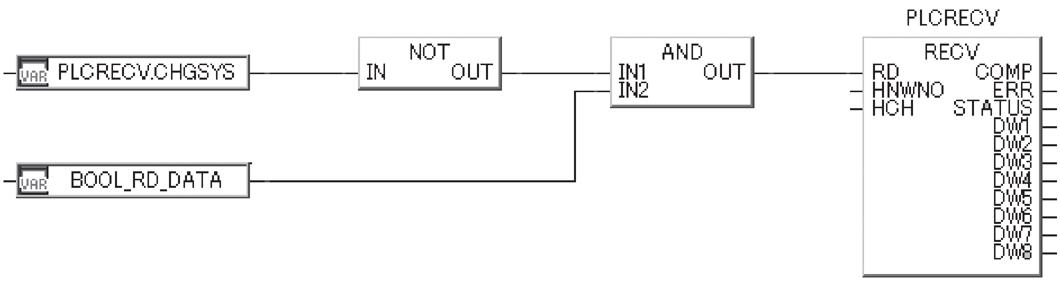
- When the input value of input variables HNWNO and HCH is out of the range. (Error code: Refer to Appendix 2)
- When the object station network is not connected with host station. (Error code: 4102)

Program Example

(1) Base Program example



(2) The following shows an example of a program that re-executes the RECV type FB in redundant system by turning the input variable RD from FALSE to TRUE at system switching when the RECV instruction execution requesting flag (BOOL_RD_DATA) is TRUE.



POINT

It is necessary to refer to public variable on user-defined FB/tag FB for setting initial value (of communication control FB public variable that is pasted on user-defined FB/tag FB) on the FB property window and reading/writing programs.

6 PROCESS FUNCTION

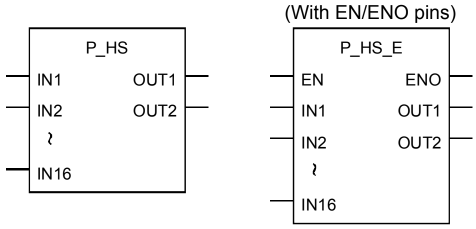
Process functions are classified as follows.

Classification name	Description	Reference
Analog value selection and Average value function	Output the maximum, minimum, intermediate, average, and absolute values of the input values.	Section 6.1

6

6.1 Analog Value Selection and Average Value Function

6.1.1 High Selector (P_HS (_E))

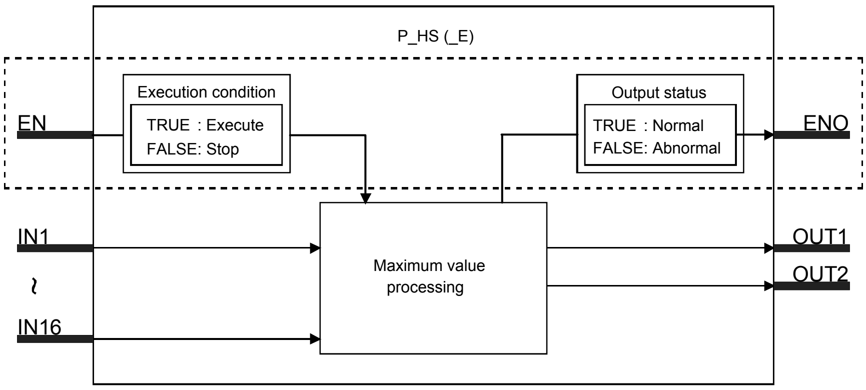
Functions	FBD parts
P_HS P_HS_E	 Number of input pins for variable IN may vary from 2 to 16.

With EN/ENO pins	○
Overload	—
Input pin number changeable (range)	2 to 16

Function overview: Output the maximum of the input values

Function/FB classification name: Analog value selection and average value function

Block Diagram

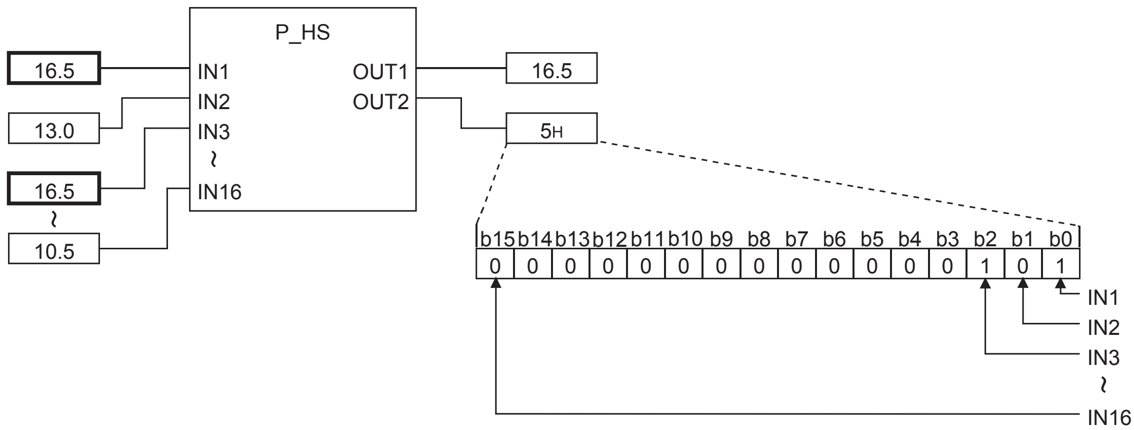


Processing in the dashed part is only carried out in fuctions with EN/ENO pins.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execution, FALSE: Stop)	TRUE, FALSE
	IN1 to IN16	Input variable	REAL	Input	-999999 to 999999
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal, FALSE: Abnormal)	TRUE, FALSE
	OUT1	Output variable	REAL	Output	-999999 to 999999
	OUT2	Output variable	WORD	Output selection	0H to FFFFH

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(1) The maximum input value of input variables IN1toIN16 is output from the output variable OUT1.
Set the corresponding bit of the maximum value within IN1toIN16 as 1, then output it from variable OUT2. (In case that multiple maximum values exist, all the corresponding bits are output from variable OUT2 after being set as 1.)
(Example) When the input values of the input variables IN1and IN3 are both of maximum values.


(2) The input values of the input variables IN1 to IN16 are all of REAL type within the range of -999999 to 999999.
(3) The pin number of input variable IN may vary from 2 to 16.
</td></tr>
<tr><td>Operation result</td><td>
(1) Functions without EN/ENO pins
The operation results are as follows:
<table>
<tr><th>Operation results</th><th>OUT1, OUT2</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error</td><td>Undefined value</td></tr>
</table>
(2) Functions With EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT1, OUT2</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error)(*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

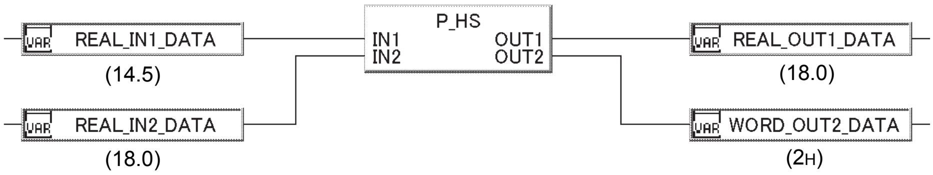
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation (Error code: Refer to Appendix 2)

Program Example

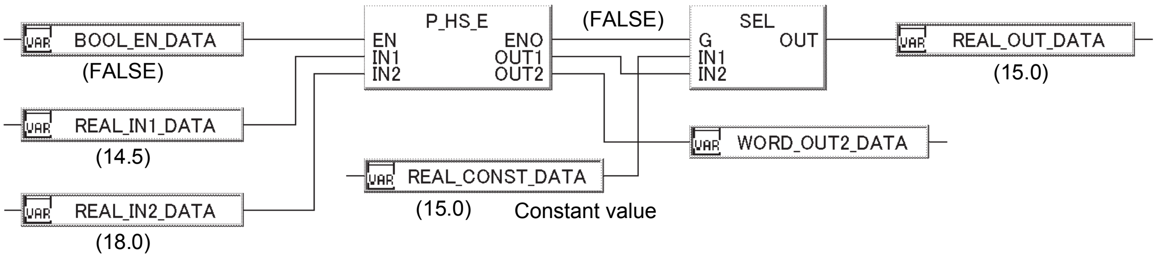
The following are the programs in which the maximum input value of input variables IN1 to IN16 is output.

(1) Basic program examples (P_HS).

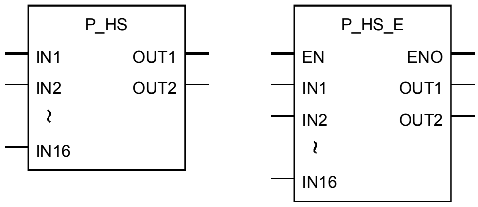


(2) This is a program example in which a constant value is output when the input variable EN is FALSE, or operation errors occur. (P_HS_E).

(Example) When the input variable EN is FALSE



6.1.2 Low Selector (P_LS (_E))

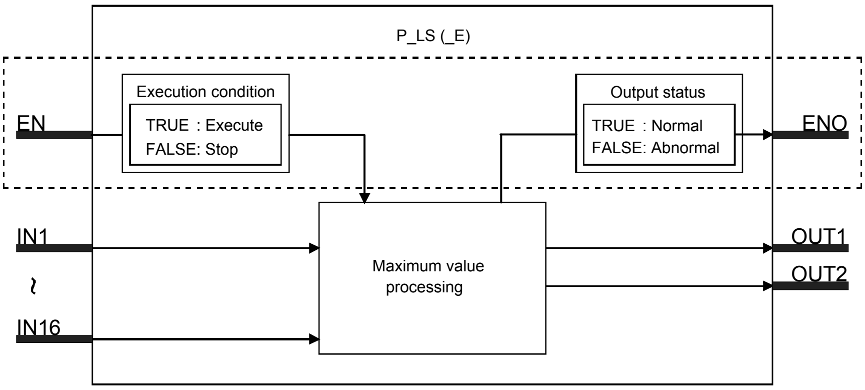
Functions	FBD parts
P_LS P_LS_E	(With EN/ENO pins)  Number of input pins for variable IN may vary from 2 to 16.

With EN/ENO pins	○
Overload	—
Input pin number changeable (range)	2 to 16

Function overview: Output the minimum input value

Function/FB classification name: Analog value selection and average value function

Block Diagram

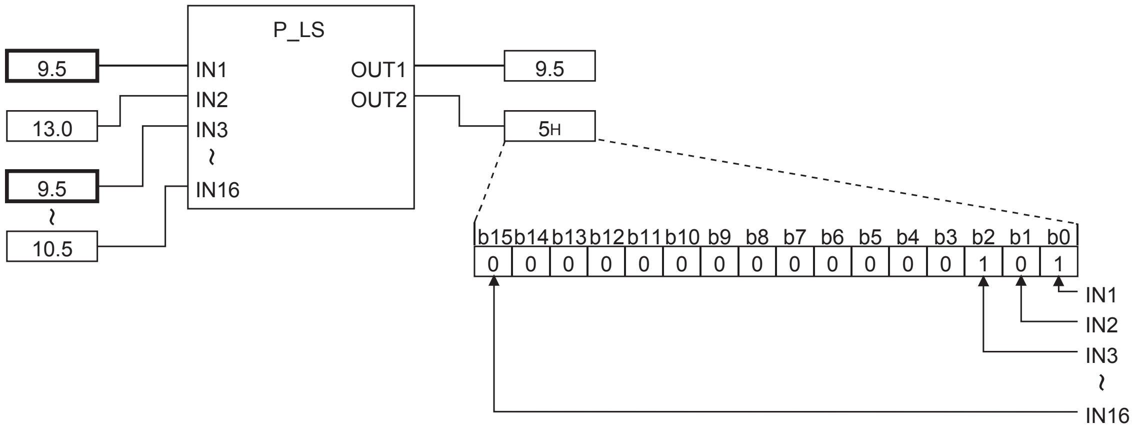


Processing in the dashed part is only carried out in fuctions with EN/ENO pins.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	IN1 to IN16	Input variable	REAL	Input	-999999 to 999999
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)	TRUE, FALSE
	OUT1	Output variable	REAL	Output	-999999 to 999999
	OUT2	Output variable	WORD	Output selection	0_H to $FFFF_H$

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Operation processing</td><td>
(1) The minimum input value of input variables IN1 to IN16 is output through the output variable OUT1.
Set the corresponding bit of the maximum value within IN1 to IN16 as 1, then output it from variable OUT2. (In case that more than one minimum value exists, all the corresponding bits are output from variable OUT2 after being set as 1.)
(Example) Suppose the input values of the input variables IN1 and IN3 are both minimum values.


(2) The input values of the input variables IN1 to IN16 are all of REAL type within the range of –999999 to 999999.

(3) The pin number of input variable IN may vary from 2 to 16.
</td></tr>
<tr><td>Operation result</td><td>
(1) Functions without EN/ENO pins.
The operation results are as follows:
<table>
<tr><th>Operation results</th><th>OUT1, OUT2</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error</td><td>Undefined value</td></tr>
</table>
(2) Functions With EN/ENO pins
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT1, OUT2</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error)(*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

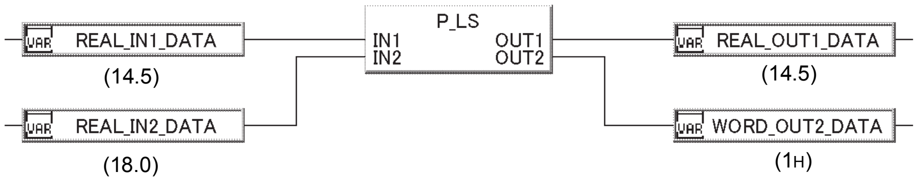
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

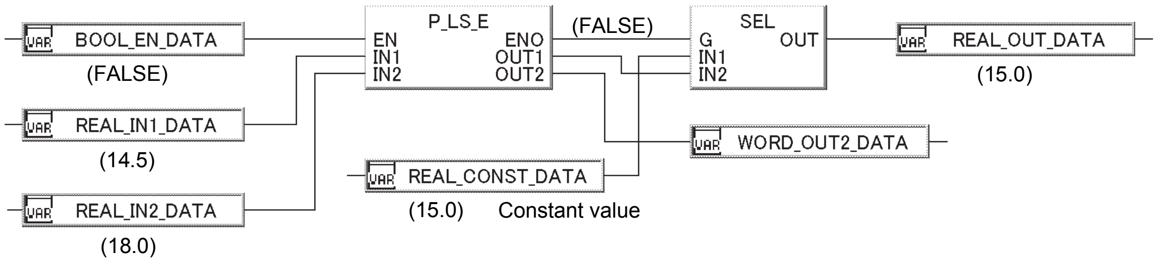
Program Example

The following are the programs in which the minimum input value of input variables IN1 to IN16 is output.

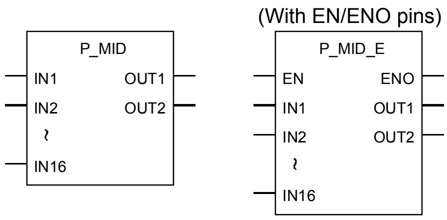
(1) Basic program examples (P_LS)



(2) This is a program example in which a constant value is output when the input variable EN is FALSE, or operation errors occur. (P_LS _E)
(Example) When the input variable EN is FALSE



6.1.3 Middle Value Selection (P_MID (_E))

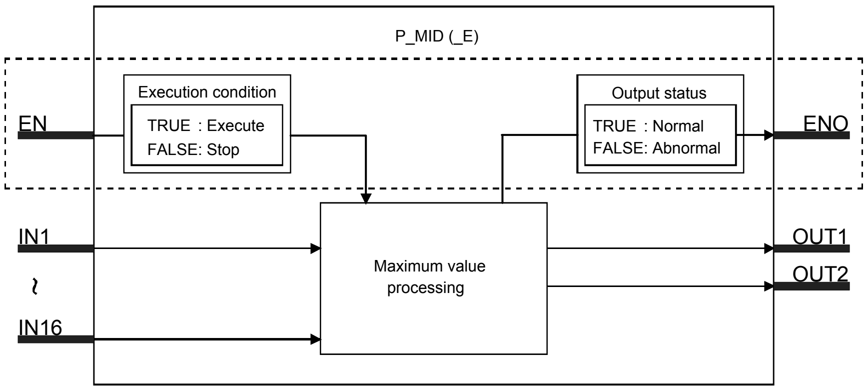
Functions	FBD parts
P_MID P_MID_E	 Number of input pins for variable IN may vary from 2 to 16.

With EN/ENO pins	○
Overload	—
Input pin number changeable (range)	2 to 16

Function overview: Output the intermediate value of the input values.

Function/FB classification name: Analog value selection and average value function

Block Diagram



Processing in the dashed part is only carried out in fuctions with EN/ENO pins.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	IN1 to IN16	Input variable	REAL	Input	-999999 to 999999
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal, FALSE: Abnormal)	TRUE, FALSE
	OUT1	Output variable	REAL	Output	-999999 to 999999
	OUT2	Output variable	WORD	Output selection	0_H to $FFFF_H$

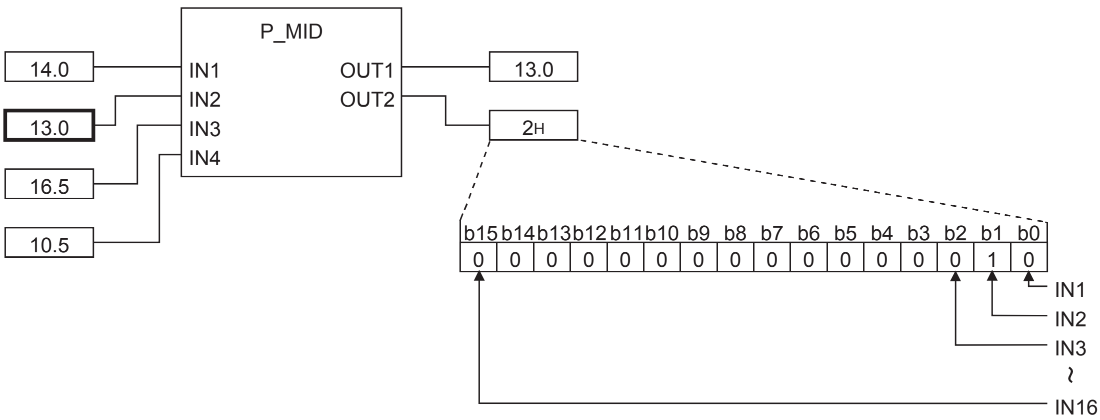
Function

Item	Contents
Operation processing	See below
Operation result	See below

Operation processing

(1) The intermediate value of input values input to input variables IN1 to IN16 is output from the output variable OUT1.
  (a) The input values are rearranged from the small to the big, and their intermediate value is output.
    (Example) 1,3,4,5,1→1,1,3,4,5
    In this case, '3' is output as the intermediate value.
  (b) If the pin number of input variables IN1 to IN16 is even, the smaller one is output.

(2) Set the corresponding bit of the input value (of middle value selection) within IN1 to IN16 as 1, then output it from variable OUT2.
(In case that more than one intermediate value exists, all the corresponding bits are output from variable OUT2 after being set as 1.)
(Example) The intermediate value of input variables IN1 to IN4 is to be output.



(3) The input values of input variables IN1 to IN16 are of REAL type within the range of –999999 to 999999.

(4) The pin number of input variable IN may vary from 2 to 16.

Operation result

(1) Functions without EN/ENO pins.
The operation results are as follows:

Operation results	OUT1, OUT2
No operation error	Operation output value
Operation error	Undefined value

(2) Functions with EN/ENO pins.
The execution conditions and the operation results are as follows:

Execution condition	Operation result	
EN	ENO	OUT1, OUT2
TRUE (operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error)(*)	Undefined value
FALSE (operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Error

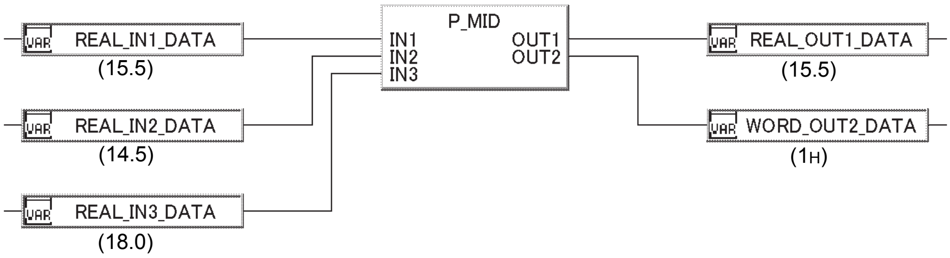
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

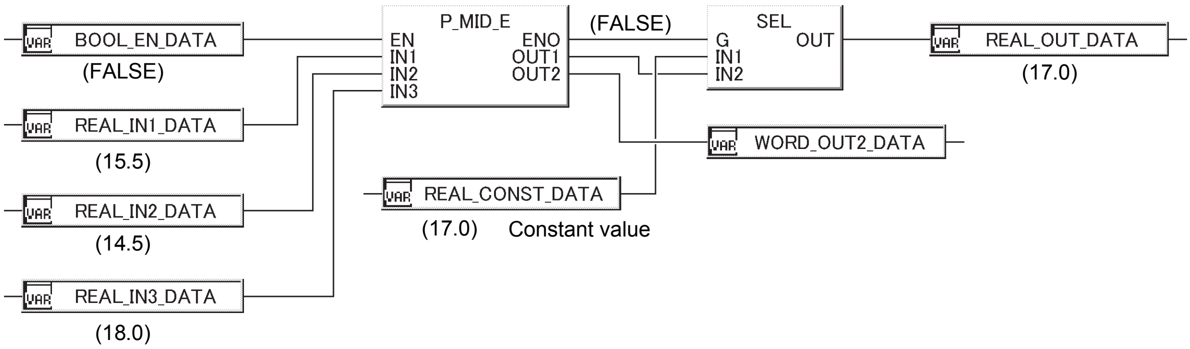
Program Example

The following are the programs in which the intermediate value of the input values input from the input variables IN1 to IN16 is output.

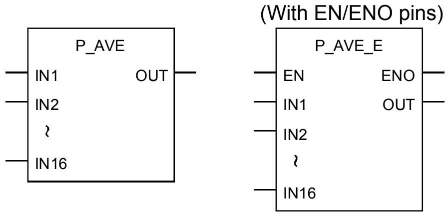
(1) Basic program examples (P_MID).



(2) This is a program example in which a constant value is output when the input variable EN is FALSE, or operation errors occur. (P_MID_E).
(Example) When the input variable EN is FALSE



6.1.4 Average Value (P_AVE (_E))

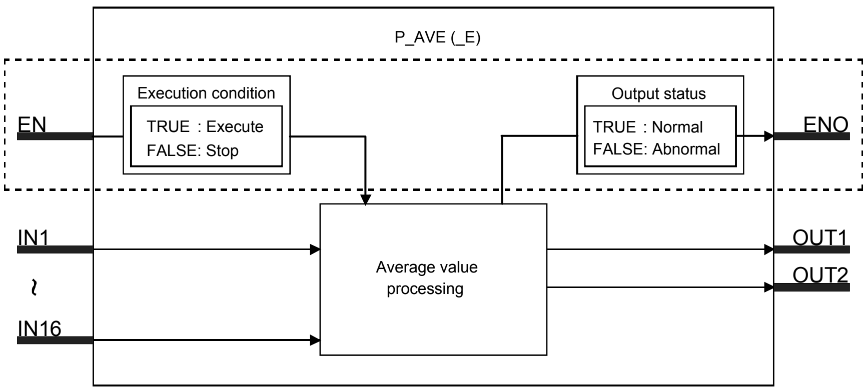
Functions	FBD parts
P_AVE P_AVE_E	(With EN/ENO pins)  Number of input pins for variable IN may vary from 2 to 16.

With EN/ENO pins	○
Overload	—
Input pin number changeable (range)	2 to 16

Function overview: Output the average of the input values.

Function/FB classification name: Analog value selection and average value function

Block Diagram



Processing in the dashed part is only carried out in fuctions with EN/ENO pins.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	IN1 to IN16	Input variable	REAL	Input	-999999 to 999999
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)	TRUE, FALSE
	OUT	Output variable	REAL	Output	-999999 to 999999

Function

Item	Content
Operation processing	(1) The average of the values input to the input variables IN1 to IN16 is output through the output variable OUT OUT = (IN1+IN2+IN3+⋯⋯IN16) ÷ N IN1 to IN16: Input values, OUT: Output value, N: Input pin number. (2) The input values of input variables IN1 to IN16 are REAL type within the range of -999999 to 999999. (3) The pin number of input variables may vary from 2 to 16.

<table>
<tr><th>Item</th><th>Content</th></tr>
<tr><td>Operation result</td><td>
(1) Functions without EN/ENO pins.
The operation results are as follows:
<table>
<tr><th>Operation results</th><th>OUT1, OUT2</th></tr>
<tr><td>No operation error</td><td>Operation output value</td></tr>
<tr><td>Operation error</td><td>Undefined value</td></tr>
</table>
(2) Functions with EN/ENO pins.
The execution conditions and the operation results are as follows:
<table>
<tr><th>Execution condition</th><th colspan="2">Operation result</th></tr>
<tr><th>EN</th><th>ENO</th><th>OUT</th></tr>
<tr><td rowspan="2">TRUE (Operation execution)</td><td>TRUE (No operation error)</td><td>Operation output value</td></tr>
<tr><td>FALSE (Operation error) (*)</td><td>Undefined value</td></tr>
<tr><td>FALSE (Operation stop)</td><td>FALSE (*)</td><td>Undefined value</td></tr>
</table>
* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)
</td></tr>
</table>

Error

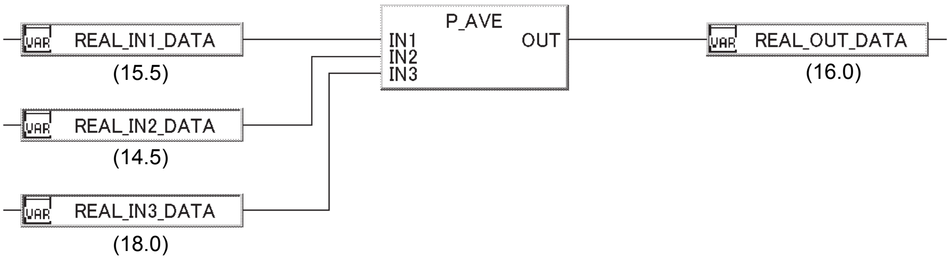
Error may occur in the following cases, error codes will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

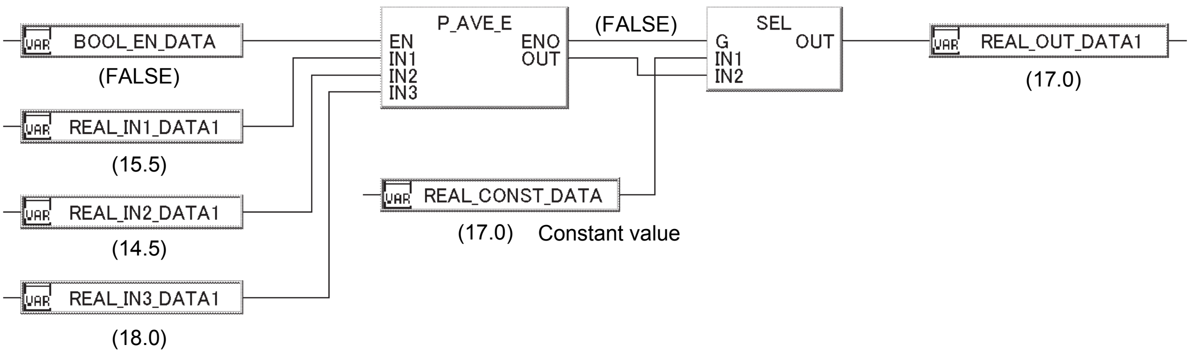
Program Example

The following are the programs in which the average of the values input from the input variables IN1 to IN16 is output.

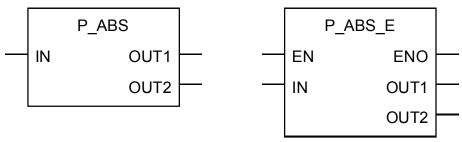
(1) Basic program examples (P_AVE).



(2) This is a program example in which a constant value is output when the input variable EN is FALSE, or operation errors occur. (P_AVE_E).
(Example) When input variable EN is FALSE



6.1.5 Absolute Value (P_ABS (_E))

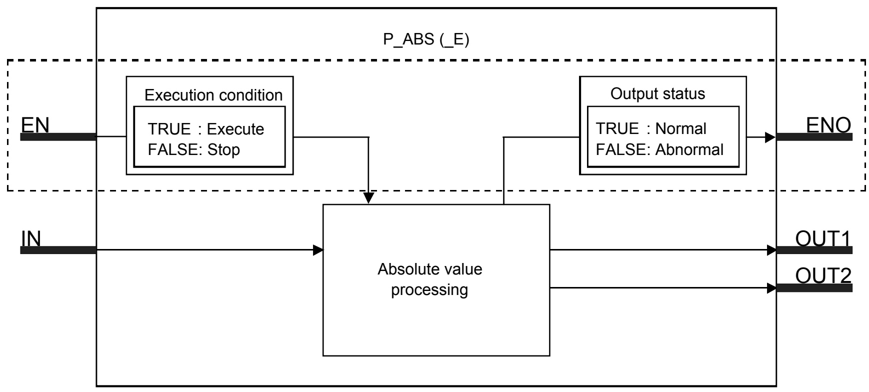
Functions	FBD parts
P_ABS P_ABS_E	(With EN/ENO pins) 

With EN/ENO pins	○
Overload	–
Input pin number changeable (range)	–

Function overview: Output the absolute value of the input value.

Function/FB classification name: Analog value selection and average value function

Block Diagram



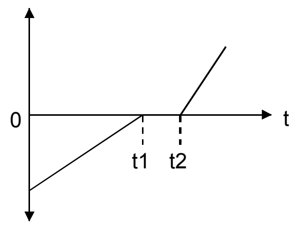
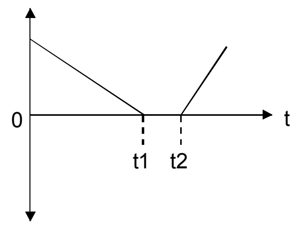
Processing in the dashed part is only carried out in fuctions with EN/ENO.

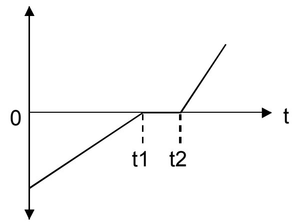
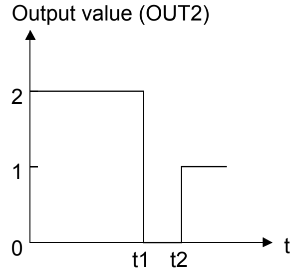
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	EN	Input variable	BOOL	Execution condition (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	IN	Input variable	REAL	Input	-999999 to 999999
Output	ENO	Output variable	BOOL	Output status (TRUE: Normal FALSE: Abnormal)	TRUE, FALSE
	OUT1	Output variable	REAL	Output	-999999 to 999999
	OUT2	Output variable	WORD	Input value sign detection (IN=0: 0_H IN>0: 1_H IN<0: 2_H)	0_H to 2_H

Function

Item	Contents
Operation processing	(1) The absolute value of the values input to the input variable IN is output from the output variable OUT1, and the sign of the input value is output from the output variable OUT2.

	Input value (IN)	Processing	Output value (OUT1)
Absolute value processing	Input value (IN) 	OUT1 = $\lvert IN \rvert$	Output value (OUT1) 

	Input value (IN)	Processing	Output value (OUT2)
Input value sign detection processing	Input value (IN) 	IN=0: OUT2= 0 IN>0: OUT2= 1 IN<0: OUT2= 2	Output value (OUT2) 

(2) The input value of the input variables IN1 to IN16 are REAL type within the range of –999999 to 999999.

Operation result

(1) Functions without EN/ENO pins
The operation results are as follows:

Operation results	OUT1, OUT2
No operation error	Operation output value
Operation error	Undefined value

(2) Functions with EN/ENO pins
The execution conditions and the operation results are as follows:

Executing condition	Operation result	
EN	ENO	OUT1, OUT2
TRUE (Operation execution)	TRUE (No operation error)	Operation output value
	FALSE (Operation error) (*)	Undefined value
FALSE (Operation stop)	FALSE (*)	Undefined value

* When ENO is FALSE, the values of variable parts connected to OUT pin and input variables of FB parts/inline ST parts are not changed.
The value of input variable of function part connected to OUT pin will be undefined value. When connected to function part, connect ENO pin and EN pin using a function block with EN pins. (☞ Section 2.7.4)

Error

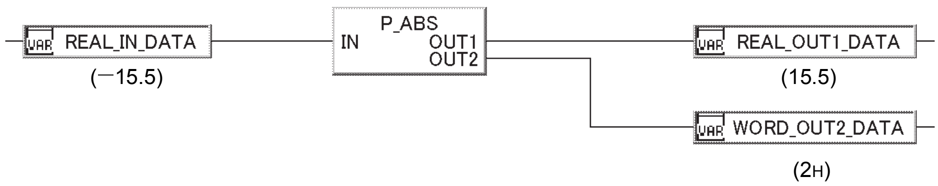
Error may occur in the following cases, error codes will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

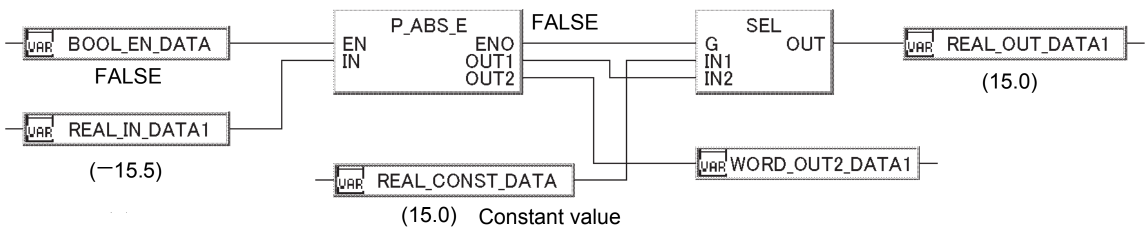
Program Example

The following are programs in which the absolute value (of value input from the input variable IN) from output variable OUT1and the input symbol detection outcome are output through output variable OUT2.

(1) Basic program examples (P_ABS).



(2) This is a program example in which a constant value is output when the input variable EN is FALSE, or operation errors occur. (P_ABS_E).
(Example) When input variable EN is FALSE



7 PROCESS FB_GENERAL PROCESS FB

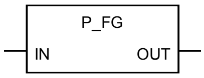
General process FB is the instructions used for process control. It can be classified into following types.

Classification Name		Description	Reference
General process FB	Correction operation FB	Operate broken line correction, standard filter, engineering value conversion, temperature/pressure correction, and integration, etc.	Section 7.1
	Arithmetic operation FB	Operate addition/subtraction, multiplication, division and square root, etc.	Section 7.2
	Comparison operation FB	Operate comparison operation (≧, >, =, <, ≦).	Section 7.3
	Control operation FB	Control operation of lead-lag, integral, derivative, high/low limiter, variation rate limiter, dead band, bumpless transfer and analog memory, etc.	Section 7.4

7

7.1 General Process FB_Correction Operation FB

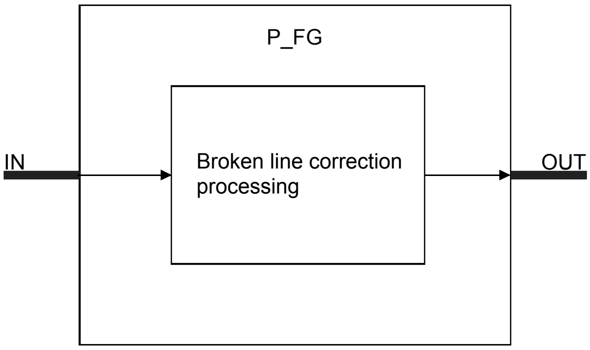
7.1.1 Function Generator (P_FG)

FB	FBD parts
P_FG	

Function overview: Output (OUT) the value from the input (IN) that follows the broken line pattern that consists of SN points.

Function/FB division name: General process FB_Correction operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	SN	Public variable	INT	Number of points	0 to 48	0	User
	X1 to X48	Public variable	REAL	Input coordinates (X Coordinates)	-999999 to 999999	0.0	User
	Y1 to Y48	Public variable	REAL	Output coordinates (Y Coordinates)	-999999 to 999999	0.0	User

7

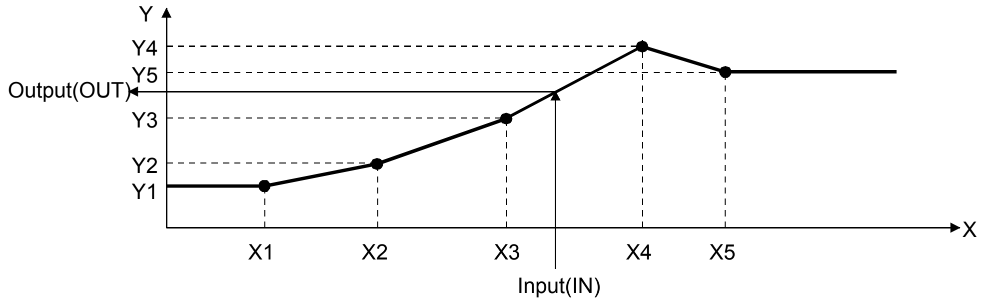
Function

Item	Contents

Broken line correction

For the input value from the input variable IN, output from output variable (OUT) the value that follows the broken line pattern that consists of SN (n=0 to 48) points.

(Example) When there are 5 points



(1) Processing contents
Execute the following operation.

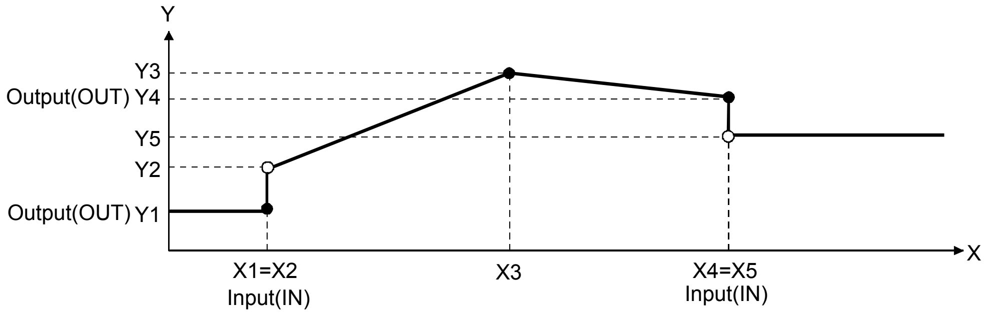
Input (IN)	Output (OUT)
IN≦X1	OUT=Y1
Xi-1<IN≦Xi (i=2 to n)	$OUT = \frac{Y_i - Y_{i-1}}{X_i - X_{i-1}} \times (IN - X_{i-1}) + Y_{i-1}$
IN>Xn	OUT=Yn

Xi: Input coordinate, Yi: Output coordinate, IN: Input value, OUT: Output value.

(a) Please ensure Xi ≦ Xi+1 for the input coordinate (Xi) setting.
(When Xi >Xi+1, take the value to Xi are valid as points.)

(b) If there are several Yi for a single Xi, those Yi with smaller Xi shall be applied.

(Example) When X1=X2, X4=X5
For input (IN), when input the input coordinate (X1=X2), output coordinate (Y1) is output.
For input (IN), when input the input coordinate (X4=X5), output coordinate (Y4) is output.



(2) Relation between the number of points range and the processing.

Number of points (SN)	Processing
SN=0	No processing
0< SN≦48	Broken line correction processing
SN< 0 or SN> 48	Operation error

Initial value setting using FB property page

The Initial value concerning function generator (P_FG) can be easily set in the FB property page of PX Developer programming tool. The following shows the setting details.

Group	Item	Variable name	Content
—	Number of points	SN	Set the number of points used in the broken line correction processing.
Coordinate	Broken points coordinate	Xn	Set the Input coordinates of the broken line correction processing.
	Broken points coordinate	Yn	Set the output coordinates of the broken line correction processing.

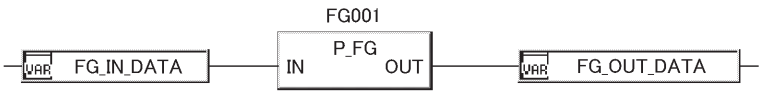
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

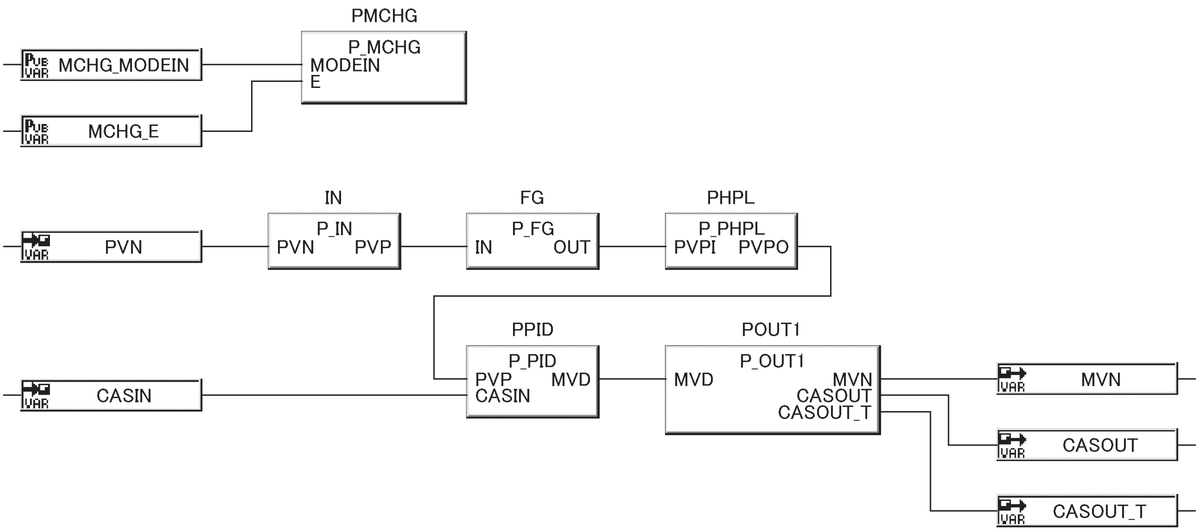
- When overflow occurs during operation (Error code: Refer to Appendix 2)
- The number of points (SN): When SN< 0 or SN> 48 (Error code: Refer to Appendix 2)

Program Example

(1) Program example 1

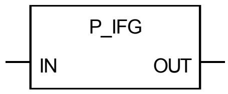


(2) Program example 2



> **POINT**
> It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

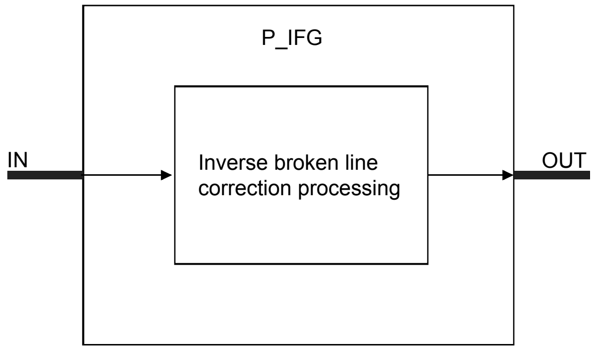
7.1.2 Inverse Function Generator (P_IFG)

FB	FBD parts
P_IFG	

Function overview: Output (OUT) the value from the input (IN) that follows the broken line pattern that consists of SN points.

Function/FB division name: General process FB_Correction operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

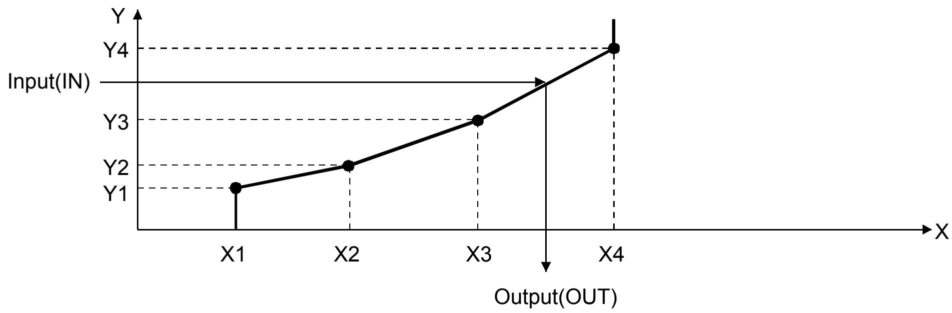
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	SN	Public variable	INT	Number of points	0 to 48	0	User
	X1 to X48	Public variable	REAL	Output coordinate (X coordinate)	-999999 to 999999	0.0	User
	Y1 to Y48	Public variable	REAL	Input coordinate (Y coordinate)	-999999 to 999999	0.0	User

Function

Item	Contents
Broken line correction	See below

For the input value from the input variable IN, output from output variable (OUT) the value that follows the broken line pattern that consists of SN (n=0 to 48) points.

(Example) When there are 4 points



(1) Processing contents
Execute the following operation.

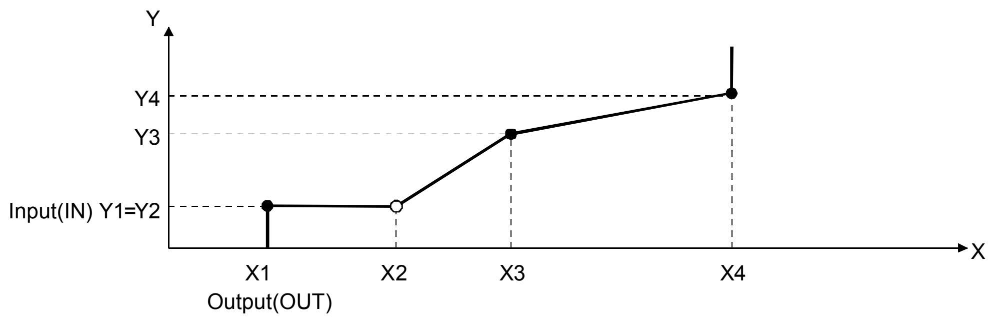
Input (IN)	Output (OUT)
IN≦Y1	OUT=X1
Yi-1< IN ≦Yi (i=2 to n)	$OUT = \frac{Yi - Yi\text{-}1}{Xi - Xi\text{-}1} \times (IN - Xi\text{-}1) + Yi\text{-}1$
IN> Yn	OUT=Xn

Xi: Output coordinate, Yi: Input coordinate, IN: Input value, OUT: Output value.

(a) Please ensure Yi ≦ Yi+1 for the input coordinate (Yi) setting.
(When Yi >Yi+1, the values to Yi are valid as points.)

(b) If there are several Xi for a single Yi, those Xi with smaller Yi shall be applied.

(Example) When Y1=Y2
For input (IN), when input the input coordinate (Y1=Y2), output coordinate (X1) is output.



(2) Relation between number of points range and the processing.

Number of points (SN)	Process
SN=0	No processing
0< SN≦48	Broken line correction processing
SN< 0 or SN> 48	Operation error

Initial value setting using FB property page

The Initial value concerning inverse function generator (P_IFG) can be easily set in the FB property page of PX Developer programming tool. The following shows the setting details.

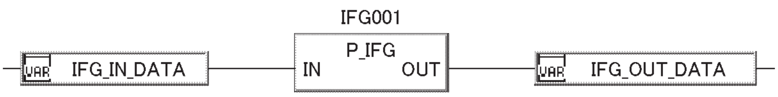
Group	Item	Variable name	Content
—	Number of points	SN	Set the number of points used in the broken line correction processing.
Coordinate	Broken points coordinate	Xn	Set the output coordinates of the broken line correction processing.
	Broken points coordinate	Yn	Set the input coordinates of the broken line correction processing.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation (Error code: Refer to Appendix 2)
- The number of points (SN): When SN< 0 or SN> 48 (Error code: Refer to Appendix 2)

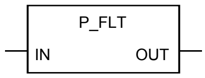
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

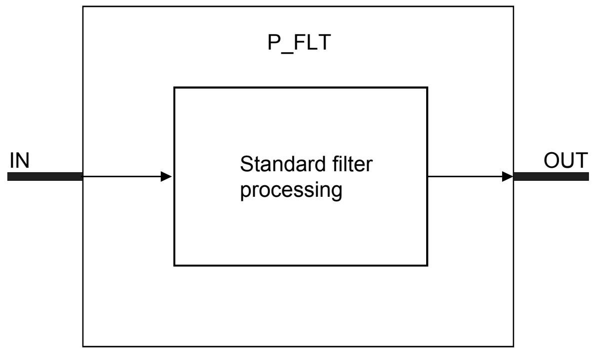
7.1.3 Standard Filter (Moving Average) (P_FLT)

FB	FBD parts
P_FLT	

Function overview: Output (OUT) the average value of 'SN' pieces of input (IN) data that are sampling collected at data collection interval ST.

Function/FB classification name: General process FB_Correction operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	ST	Public variable	REAL	Data collection interval (unit: s)	0 to 999	1.0	User
	SN	Public variable	INT	Sampling number	0 to 48	0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Standard filter process</td><td>
Output (OUT) the average value of 'SN' pieces of input (IN) data that are collected at data collection interval.
(1) Processing contents
Execute the following operation.

$$OUT = \frac{IN1 + IN2 + IN3 + \cdots + IN_{SN}}{SN}$$

SN: Sampling number, IN1 to IN_{SN}: Input value, OUT: Output value
(a) Data updating period is set as $\frac{ST}{\triangle T}$ ($\triangle$T: Execution cycle). (The data after the decimal point will be rounded off)
(b) Before input (IN) reaches the sampling (SN) number, output the average value of sampling input (IN) that is collected so far.
(c) Please set data collection interval (ST) as ST=n $\times$ $\triangle$T. (n is integer).

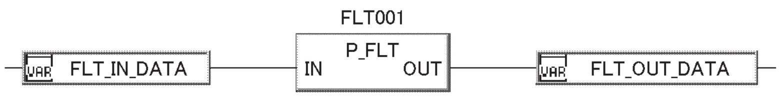
(2) Relation between range of sampling number (SN) and the processing.
<table>
<tr><th>Sampling number (SN)</th><th>Processing</th></tr>
<tr><td>0< SN ≦ 48</td><td>Standard filter processing</td></tr>
<tr><td>SN=0</td><td>Output (OUT)=0</td></tr>
<tr><td>SN< 0 or SN> 48</td><td>Operation error</td></tr>
</table>
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)
- The number of sampling number (SN): When SN< 0 or SN> 48. (Error code: Refer to Appendix 2)

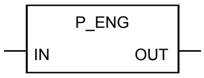
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

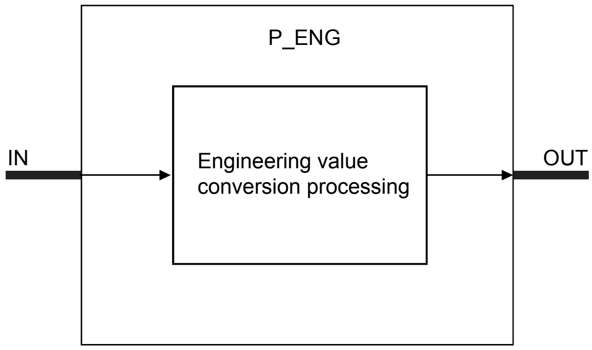
7.1.4 Engineering Value Conversion (P_ENG)

FB	FBD parts
P_ENG	

Function overview: Convert the input data (%) into temperature or pressure engineering value and output (OUT).

Function/FB division name: General process FB_Correction operation FB

Block Diagram



Input and Output Pins

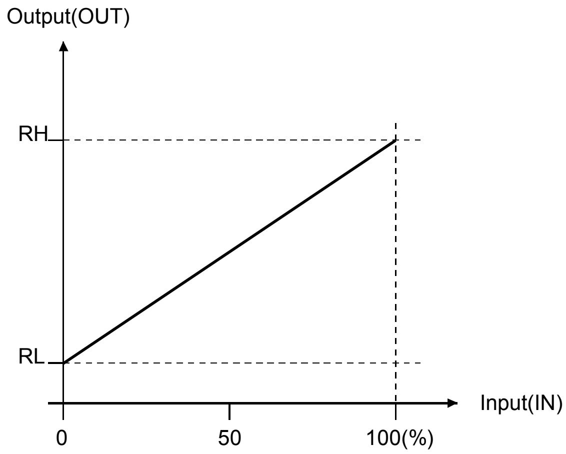
Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	RH	Public variable	REAL	Engineering value high limit	-999999 to 999999	100.0	User
	RL	Public variable	REAL	Engineering value low limit	-999999 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Engineering value conversion processing</td><td>
Convert the data which is input from input variable IN using percentage (%) type into temperature or pressure engineering value (%) and output the result from OUT.



(1) Processing contents
Execute the following operation.

$$OUT = \{[RH\text{-}RL] \times \frac{IN}{100}\} + RL$$

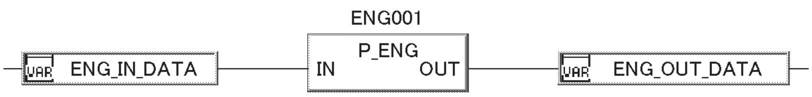
RH: Engineering value high limit, RL: Engineering value low limit,
IN: Input value (%) (0% to 100%), OUT: Output value
● This expression is applicable in case of RH≦RL, too.
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

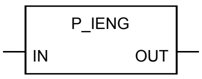
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

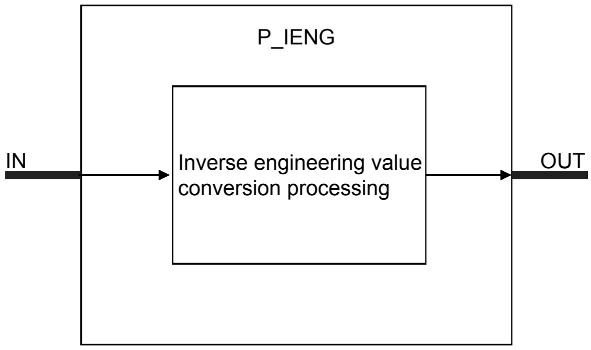
7.1.5 Inverse Engineering Value Conversion (P_IENG)

FB	FBD parts
P_IENG	

Function overview: Convert the input engineering value such as temperature and pressure into percentage (%) and output the result from OUT.

Function/FB division name: General process FB_Correction operation FB

Block Diagram



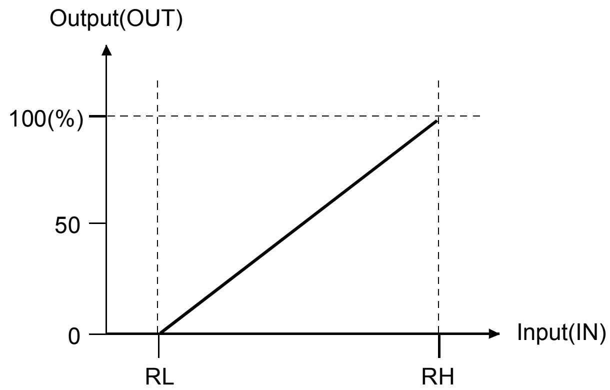
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	RH	Public variable	REAL	Engineering value high limit	-999999 to 999999	100.0	User
	RL	Public variable	REAL	Engineering value low limit	-999999 to 999999	0.0	User

Function

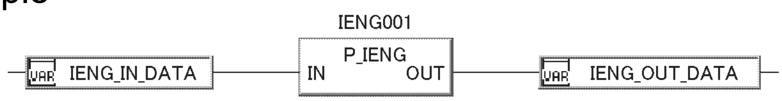
Item	Contents
Inverse engineering value conversion processing	Convert the value input from input variable IN, such as temperature or pressure into percentage (%) and output the result from OUT.  (1) Processing contents Execute the following operation. $$\mathrm{OUT}(\%) = \frac{\mathrm{IN} - \mathrm{RL}}{\mathrm{RH} - \mathrm{RL}} \times 100$$ RH: Engineering value high limit, RL: Engineering value low limit, IN: Input value, OUT: Output value (0% to 100%) ● This expression is applicable in case of RH≦RL, too.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

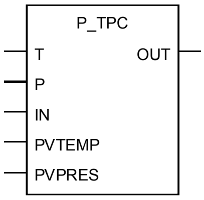
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

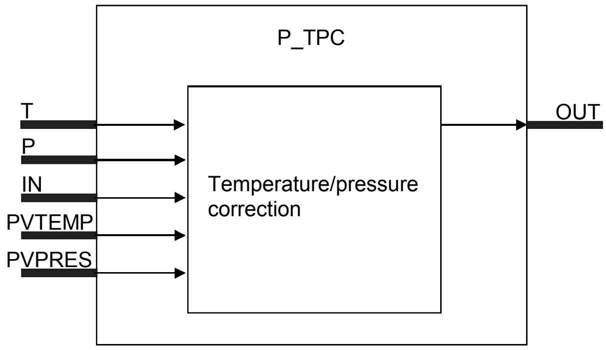
7.1.6 Temperature/Pressure Correction (P_TPC)

FB	FBD parts
P_TPC	

Function overview: For the differential pressure input (IN), execute temperature/pressure correction (temperature correction or pressure correction) operation and output the result from OUT.

Function/FB division name: General process FB_Correction operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	T	Input variable	BOOL	Temperature correction selection (TRUE: Used, FALSE: Not used)	TRUE, FALSE
	P	Input variable	BOOL	Pressure correction selection (TRUE: Used, FALSE: Not used)	TRUE, FALSE
	IN	Input variable	REAL	Differential pressure input (%)	0 to 100
	PVTEMP	Input variable	REAL	Measured temperature (engineering value)	-999999 to 999999
	PVPRES	Input variable	REAL	Measured pressure (engineering value)	-999999 to 999999
Output	OUT	Output variable	REAL	Output (%)	0 to 100

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	TEMP	Public variable	REAL	Design temperature T' (engineering value)	-999999 to 999999	0.0	User
	B1	Public variable	REAL	Bias temperature (engineering value)	-999999 to 999999	273.15	User
	PRES	Public variable	REAL	Design pressure P' (engineering value)	-999999 to 999999	0.0	User
	B2	Public variable	REAL	Bias pressure (engineering value)	-999999 to 999999	10332.0	User

Function

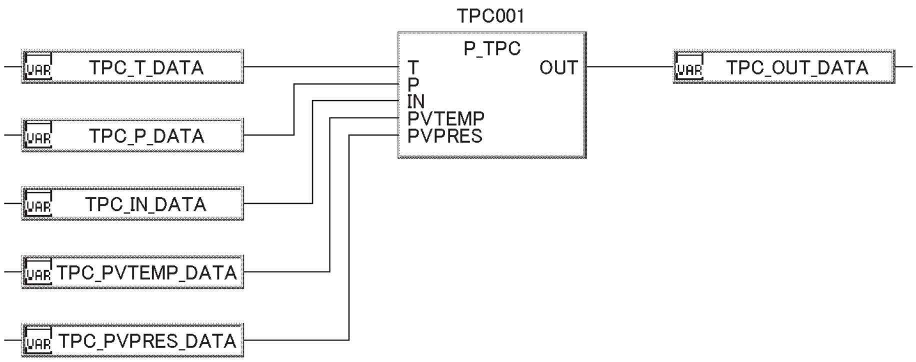
<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Temperature/ pressure correction processing</td><td>
For the differential pressure input (IN), execute temperature/pressure correction (temperature correction or pressure correction) operation and output the result from OUT.
(1) Processing contents
Operate following items.
<table>
<tr><th colspan="2">Temperature and pressure correction selection</th><th rowspan="2">A1</th><th rowspan="2">A2</th><th rowspan="2">Output (OUT)</th></tr>
<tr><th>Temperature correction (T)</th><th>Pressure correction (P)</th></tr>
<tr><td>TRUE</td><td>TRUE</td><td>$\frac{TEMP + B1}{PVTEMP + B1}$</td><td>$\frac{PVPRES + B2}{PRES + B2}$</td><td rowspan="3">OUT=IN×A1×A2</td></tr>
<tr><td>FALSE</td><td>TRUE</td><td>1.0</td><td>$\frac{PVPRES + B2}{PRES + B2}$</td></tr>
<tr><td>TRUE</td><td>FALSE</td><td>$\frac{TEMP + B1}{PVTEMP + B1}$</td><td>1.0</td></tr>
</table>
T: Temperature correction selection, P: Pressure correction selection, IN: Differential pressure input (%),
PVTEMP: Measured temperature (engineering value), PVPRES: Measured pressure (engineering value),
TEMP: Design temperature T' (engineering value), PRES: Design pressure P' (engineering value),
B1: Bias temperature (engineering value), B2: Bias pressure (engineering value), OUT: Output (%)
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

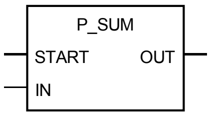
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

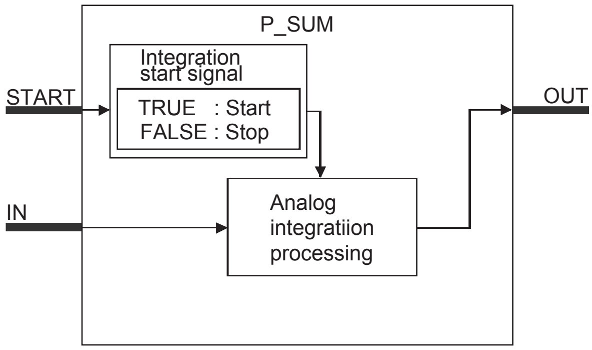
7.1.7 Summation (P_SUM)

FB	FBD parts
P_SUM	

Function overview: When integration start signal (START) is TRUE, execute integration operation on the input (IN) and outputs (OUT) the result.

Function/FB division name: General process FB_Correction operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	START	Input variable	BOOL	Integration start signal (TRUE: Start, FALSE: Stop)	TRUE, FALSE
	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	ILC	Public variable	REAL	Input low cut-off value	-999999 to 999999	0.0	User
	A	Public variable	REAL	Initial value	-999999 to 999999	0.0	User
	RANGE	Public variable	INT	Input range 1:/s 2:/min 3:/hour	1 to 3	1	User

Function

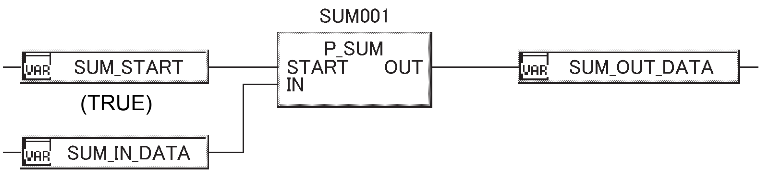
<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Analog integration processing</td><td>When integration start signal (START) is TRUE, accumulates the input value that comes from input variable IN, and outputs the result from variable OUT.
(1) Processing contents
Execute the following operation.
<table>
<tr><th>Integration start signal (START)</th><th>Input (IN)</th><th>Output (OUT)</th></tr>
<tr><td>FALSE: Stop</td><td>Optional</td><td>OUT=Initial value (A)</td></tr>
<tr><td rowspan="2">TRUE: Start</td><td>IN≦ILC</td><td>OUT=Previous value</td></tr>
<tr><td>IN>ILC</td><td>$OUT = (IN \times \frac{\triangle T}{T}) + \text{previous value}$</td></tr>
</table>
△T: Execution cycle, ILC: Input low cut-off value, A: Initial value,
T: When RANGE=1, T=1(s),
RANGE=2, T=60(s),
RANGE=3, T=3600(s)
(Example) When input 0 to 5m^3/min, the setting should be RANGE=2 due to input range"/min".
Besides, multiplying factor is ×1 m^3.</td></tr>
<tr><td>Integration start signal</td><td>When integration start signal (START) is FALSE: Stop integration
When integration start signal (START) is TRUE: Start integration</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)
- When input range (RANGE) is not within1 to 3. (Error code: Refer to Appendix 2)

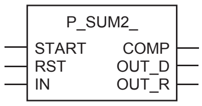
Program Example



POINT

(1) It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.
(2) Use P_SUM2_ to reduce the influence of information loss in the single-precision floating-point operation. P_SUM is used to keep the compatibility with existing programs.
For the information loss, refer to Section 2.4.

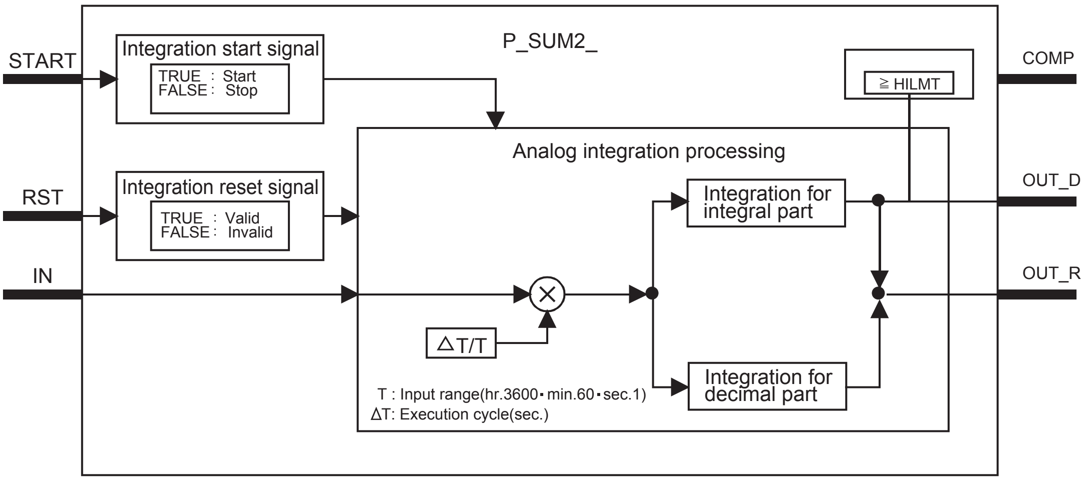
7.1.8 Summation (Internal Integer Integration) (P_SUM2_)

FB	FBD parts
P_SUM2_	

Function overview: When integration start signal (START) is TRUE, execute integration operation on the input (IN) and outputs the result.
Internal integration for the integral part is executed by signed 32-bit integer.

Function/FB division name: General process FB_Correction operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	START	Input variable	BOOL	Integration start signal (TRUE: Start, FALSE: Stop)	TRUE, FALSE
	RST	Input variable	BOOL	Integration reset signal (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE
	IN	Input variable	REAL	Input	-999999 to 999999
Output	COMP	Output variable	BOOL	Integration complete signal (TRUE: Complete, FALSE: Unreached)	TRUE, FALSE
	OUT_D	Output variable	DINT	Integration value output (integral part)	-2147483648 to 2147483647
	OUT_R	Output variable	REAL	Integration value real number output	-2147483648 to 2147483647

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	ILC	Public variable	REAL	Input low cut-off value	-999999 to 999999	0.0	User
	A	Public variable	REAL	Initial value	-999999 to 999999	0.0	User
	RANGE	Public variable	INT	Input range 1:/s 2:/min 3:/hour	1 to 3	1	User
	HILMT	Public variable	DINT	Integration high limit	1 to 2147483647	1000000	User
	CYCLIC	Public variable	BOOL	TRUE : Returns to 0 when CYCLIC is more than the integration high limit *1 FALSE: Keeps the high limit value when CYCLIC is more than the integration high limit.	TRUE, FALSE	TRUE	User

*1 Integration value (OUT_R) will be the value to which a surplus to the integration high limit is added.
(Example) Integration value will be 10 for the following condition: HILMT=1000, Last integration value=990, Current value (IN×ΔT/T) =20.

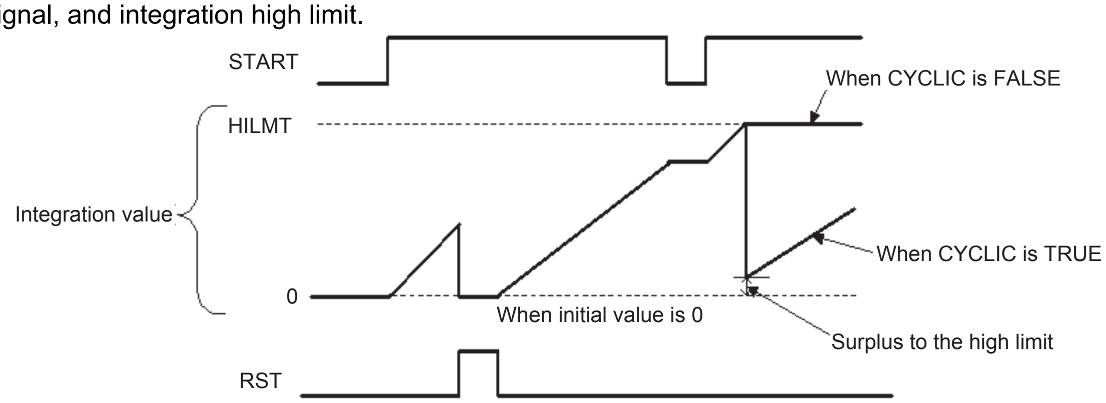
POINT

REAL type output value (OUT_R output variable) is processed by 32-bit single-precision floating-point, so that the number of significant digits is six to seven digits. Consequently, a rounding error occurs when the integral value exceeds the number of significant digits range, and the integral part may not match with DINT type output value (OUT_D output variable).

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Analog integration processing</td><td>When integration start signal (START) is TRUE and integration reset signal (RST) is FALSE, accumulates the input value that comes from input variable IN, and outputs the result from variable.
(1) Processing contents
Execute the following operation.
<table>
<tr><th>Integration start signal (START)</th><th>Integration reset signal (RST)</th><th>Input (IN)</th><th>Output (OUT)</th></tr>
<tr><td rowspan="2">FALSE: Stop</td><td>FALSE: Invalid</td><td>Optional</td><td>OUT= Previous value</td></tr>
<tr><td>TRUE: Valid</td><td>Optional</td><td>OUT=Initial value (A)</td></tr>
<tr><td rowspan="3">TRUE: Start</td><td rowspan="2">FALSE: Invalid</td><td>IN≦ILC</td><td>OUT=Previous value</td></tr>
<tr><td>IN>ILC</td><td>OUT = $(IN \times \frac{\triangle T}{T})$ + previous value</td></tr>
<tr><td>TRUE: Valid</td><td>Optional</td><td>OUT=Initial value (A)</td></tr>
</table>
△T: Execution cycle, ILC: Input low cut-off value, A: Initial value,
T: When RANGE=1, T=1(s),
RANGE=2, T=60(s),
RANGE=3, T=3600(s)
(Example) When input 0 to 5m^3/min, the setting should be RANGE=2 due to input range"/min".
Besides, multiplying factor is ×1 m^3.</td></tr>
<tr><td>Integration start signal</td><td>When integration start signal (START) is FALSE: Stop integration
When integration start signal (START) is TRUE: Start integration</td></tr>
<tr><td>Integration reset signal</td><td>When integration reset signal (RST) is FALSE: No integration value reset (initial value)
When integration reset signal (RST) is TRUE: Reset the integration value and output the initial value</td></tr>
<tr><td>Integration complete signal</td><td>When integration value output (integral part)(OUT_D) ≧ Integration high limit(HILMT): Integration complete signal is TRUE*1
When integration value output (integral part)(OUT_D) < Integration high limit(HILMT): Integration complete signal is FALSE
*1 When CYCLIC is TRUE, TRUE is output for one cycle only.</td></tr>
</table>

The following shows the timing chart for the integration start signal (START), integration value, integration reset signal, and integration high limit.

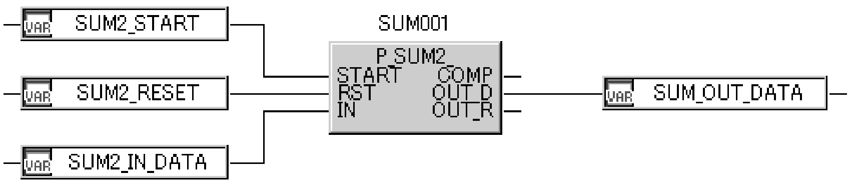


Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

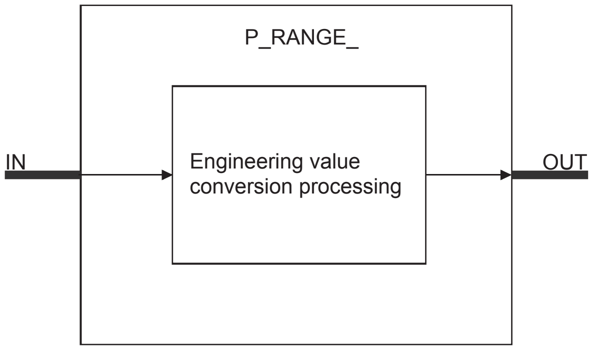
7.1.9 Range Conversion (P_RANGE_)

FB	FBD parts
P_RANGE_	

Function overview: Converts the input data (IN) into the specified value (OUT).

Function/FB division name: General process FB_Correction operation FB

Block Diagram



Input and Output Pins

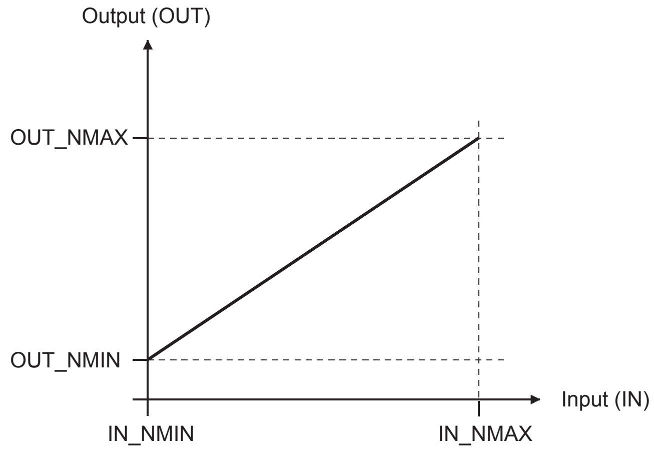
Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	OUT_NMAX	Public variable	REAL	Output high limit	-999999 to 999999	100.0	User
	OUT_NMIN	Public variable	REAL	Output low limit	-999999 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Engineering value conversion processing</td><td>
Range-convert the data which is input from input variable IN and output the result from OUT.



(1) Processing contents
Execute the following operation.

$$OUT=\frac{(OUT\_NMAX-OUT\_NMIN)\times(IN-IN\_NMIN)}{IN\_NMAX-IN\_NMIN}+OUT\_NMIN$$

OUT_NMAX: Output high limit, OUT_NMIN: Output low limit, IN_NMIN: Input low limit, IN_NMAX: Input high limit, IN: Input value (%) (0% to 100%), OUT: Output value
● This expression is applicable in case of OUT_NMAX≦OUT_NMIN or IN_NMAX≦IN_NMIN, too.
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

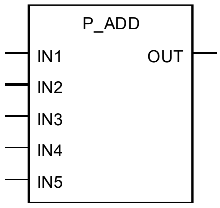
Program Example

POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

7.2 General Process FB_Arithmetic Operation FB

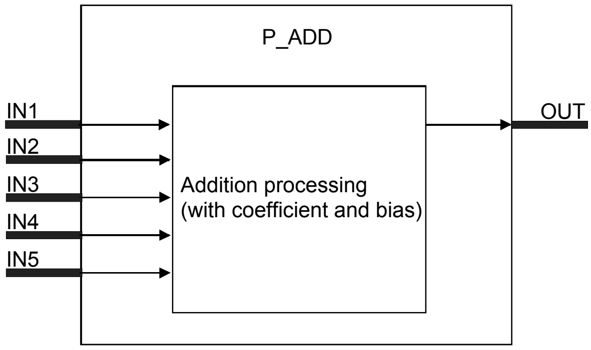
7.2.1 Addition (With Coefficient) (P_ADD)

FB	FBD parts
P_ADD	

Function overview: For the input (IN1 to IN5), add the input data with coefficient and bias, and outputs (OUT) the result.

Function/FB division name: General process FB_Arithmetic operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN1	Input variable	REAL	Input1	-999999 to 999999
	IN2	Input variable	REAL	Input2	-999999 to 999999
	IN3	Input variable	REAL	Input3	-999999 to 999999
	IN4	Input variable	REAL	Input4	-999999 to 999999
	IN5	Input variable	REAL	Input5	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

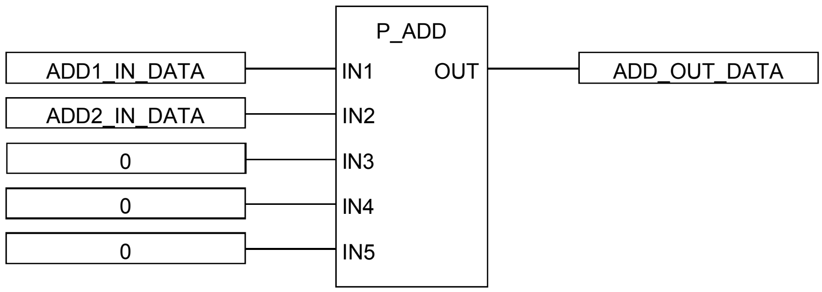
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	K1 to K5	Public variable	REAL	Coefficient 1: Coefficient of IN1 data to Coefficient 5: Coefficient of IN5 data	-999999 to 999999	1.0	User
	B	Public variable	REAL	Bias	-999999 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Addition processing</td><td>
For the input value from the input variable IN1 to IN5, add the input data (with coefficient and bias), and output from variable OUT.

(1) Processing contents
Execute the following operation.
<table>
<tr><th>Input (IN)</th><th>Output (OUT)</th></tr>
<tr><td>IN1 to IN5 (*1)</td><td>OUT=(K1×IN1)+(K2×IN2)+(K3×IN3)+(K4×IN4)+(K5×IN5)+B</td></tr>
</table>
IN1 to IN5: Input value, K1 to K5: Coefficient, B: Bias.

*1 When there is no input data for input (IN1 to IN5), please input 0.
(Example) When only IN1 and IN2 exist.
Output (OUT) will be OUT=(K1×IN1)+(K2×IN2)+B.


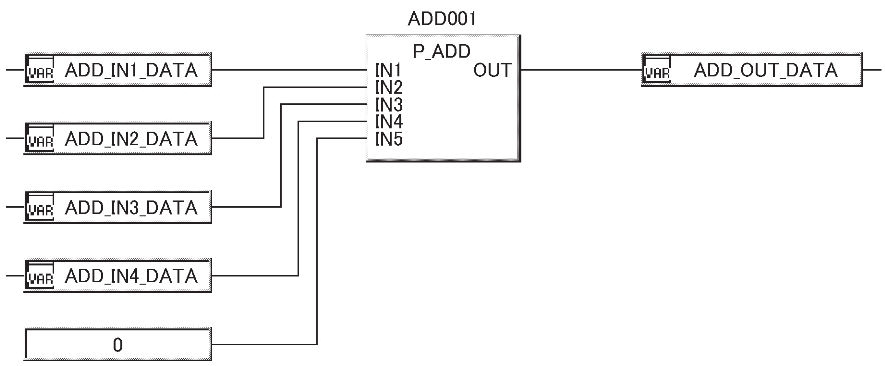
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

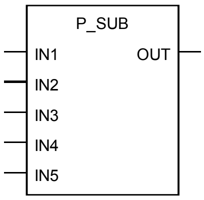
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

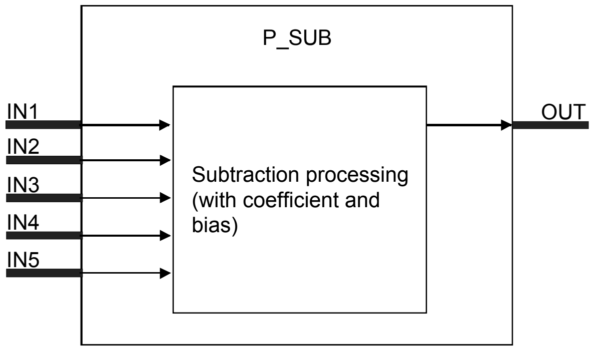
7.2.2 Subtraction (With Coefficient) (P_SUB)

FB	FBD parts
P_SUB	

Function overview: For the input (IN1 to IN5), subtract the data (with coefficient and bias), and outputs (OUT) the result.

Function/FB division name: General process FB_Arithmetic operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN1	Input variable	REAL	Input1	-999999 to 999999
	IN2	Input variable	REAL	Input2	-999999 to 999999
	IN3	Input variable	REAL	Input3	-999999 to 999999
	IN4	Input variable	REAL	Input4	-999999 to 999999
	IN5	Input variable	REAL	Input5	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

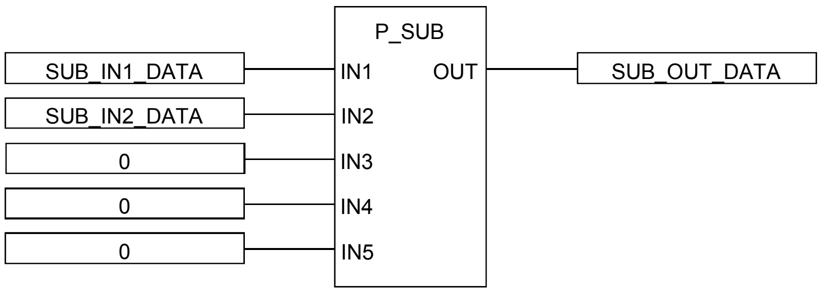
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	K1 to K5	Public variable	REAL	Coefficient 1: Coefficient of IN1 data to Coefficient 5: Coefficient of IN5 data	-999999 to 999999	1.0	User
	B	Public variable	REAL	Bias	-999999 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Subtraction processing</td><td>For the input value from the input variable IN1 to IN5, subtract the input data with coefficient and bias, and output from variable OUT.

(1) Processing contents
Execute the following operation.
<table><tr><th>Input (IN)</th><th>Output (OUT)</th></tr><tr><td>IN1 to IN5 (*1)</td><td>OUT=(K1 × IN1)-(K2 × IN2)-(K3 × IN3)-(K4 × IN4)-(K5 × IN5)+B</td></tr></table>IN1 to IN5: Input value, K1 to K5: Coefficient, B: Bias.

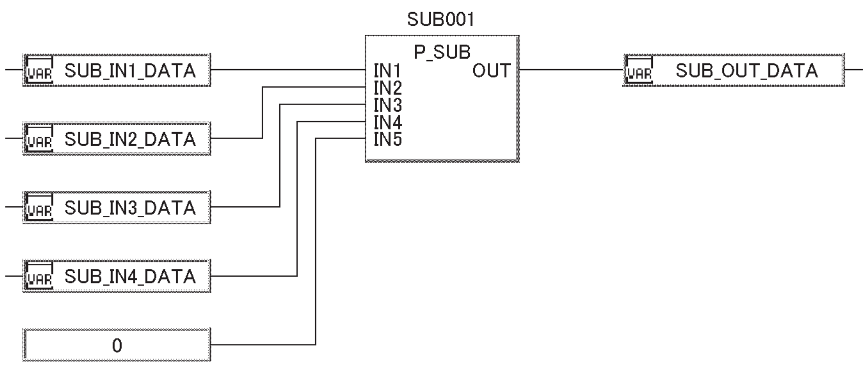
*1 When there is no input data for input (IN1 to IN5) please input 0.
(Example) When only IN1 and IN2 exist.
Output (OUT) will be OUT=(K1 × IN1)-(K2 × IN2)+B.

</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

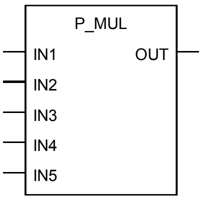
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

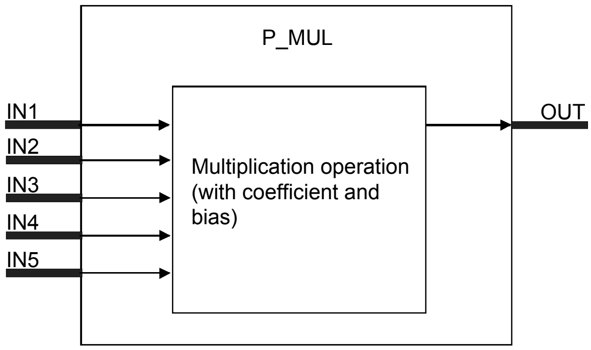
7.2.3 Multiplication (With Coefficient) (P_MUL)

FB	FBD parts
P_MUL	

Function overview: For the input (IN1 to IN5), multiply the input data with coefficient and bias, and output (OUT) the result.

Function/FB division name: General process FB_Arithmetic operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN1	Input variable	REAL	Input1	-999999 to 999999
	IN2	Input variable	REAL	Input2	-999999 to 999999
	IN3	Input variable	REAL	Input3	-999999 to 999999
	IN4	Input variable	REAL	Input4	-999999 to 999999
	IN5	Input variable	REAL	Input5	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

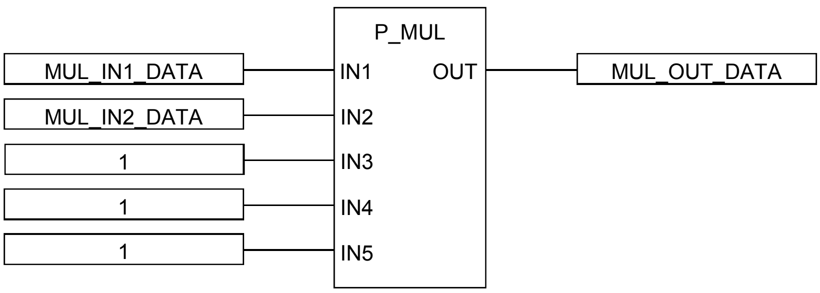
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	K1 to K5	Public variable	REAL	Coefficient 1: Coefficient of IN1 data to Coefficient 5: Coefficient of IN5 data	-999999 to 999999	1.0	User
	B	Public variable	REAL	Bias	-999999 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Multiplication processing</td><td>
For the input value from the input variable IN1 to IN5, multiply the input data with coefficient and bias, and output the result from variable OUT.

(1) Processing contents
Execute the following operation.
<table>
<tr><th>Input (IN)</th><th>Output (OUT)</th></tr>
<tr><td>IN1 to IN5 (*1)</td><td>OUT=(K1×IN1) × (K2×IN2) × (K3×IN3) × (K4×IN4) × (K5×IN5)+B</td></tr>
</table>
IN1 to IN5: Input value, K1 to K5: coefficient, B: Bias.

*1 When there is no input data for input (IN1 to IN5), please input 1 for input and coefficient.
(When either input or coefficient is set to 0, bias (B) will be output from OUT.)
(Example) When only IN1 and IN2 exist.
Output (OUT) will be OUT=(K1×IN1) × (K2×IN2)+B.


</td></tr>
</table>

POINT

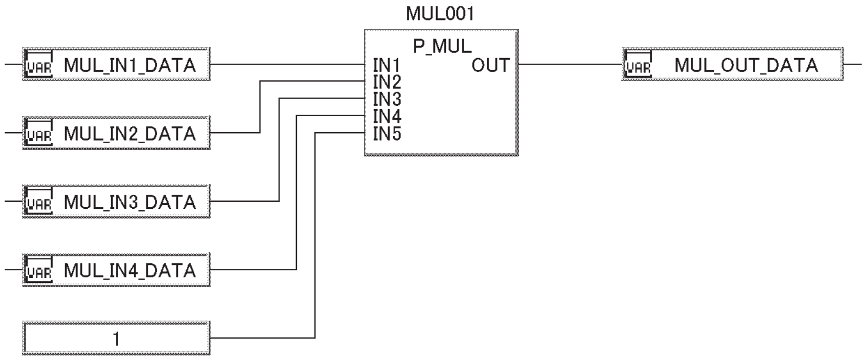
When there is no input data in input (IN1 to IN5), please input 1 to both input and coefficient.
When either input or coefficient is set to 0, bias (B) will be output from OUT.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

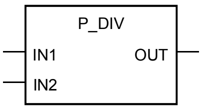
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

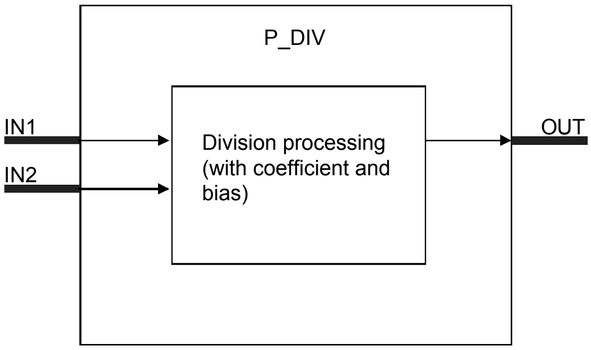
7.2.4 Division (With Coefficient) (P_DIV)

FB	FBD parts
P_DIV	

Function overview: For the input (IN1, IN2), divide the input data with coefficient and bias, and output (OUT) the result.

Function/FB division name: General process FB_Arithmetic operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN1	Input variable	REAL	Input1	-999999 to 999999
	IN2	Input variable	REAL	Input2	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	A	Public variable	REAL	Coefficient	-999999 to 999999	1.0	User
	K1	Public variable	REAL	Coefficient 1: Coefficient of IN1 data	-999999 to 999999	1.0	User
	K2	Public variable	REAL	Coefficient 2: Coefficient of IN2 data	-999999 to 999999	1.0	User
	B1	Public variable	REAL	IN1 data bias	-999999 to 999999	0.0	User
	B2	Public variable	REAL	IN2 data bias	-999999 to 999999	0.0	User
	B3	Public variable	REAL	Bias	-999999 to 999999	0.0	User

Function

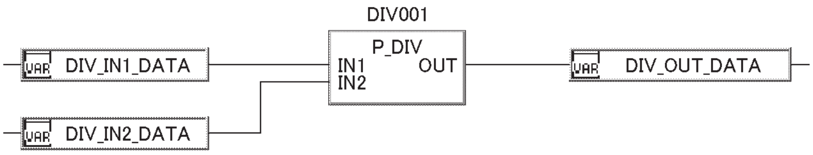
<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Division processing</td><td>For the input value from the input variable IN1 and IN2, divide the input data with coefficient and bias, and output from variable OUT.
(1) Processing contents
Execute the following operation.
<table>
<tr><th>Input (IN2), coefficient (K2), bias (B2): Denominator</th><th>Output (OUT)</th></tr>
<tr><td>If K2 $\times$ IN2+B2 is not 0 (denominator≠0)</td><td>$OUT = A \times \frac{K1 \times IN1 + B1}{K2 \times IN2 + B2} + B3$</td></tr>
<tr><td>If K2 $\times$ IN2+B2 is 0 (denominator=0)</td><td>OUT=B3</td></tr>
</table>
IN1 to IN2: Input value, A and K1 to K5: Coefficient, B1 to B3: Bias.</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

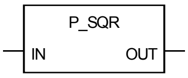
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

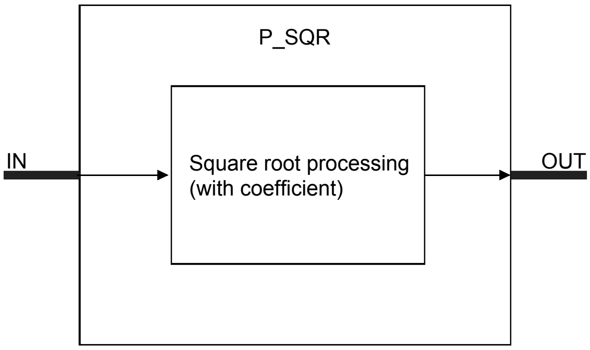
7.2.5 Square Root (With Coefficient) (P_SQR)

FB	FBD parts
P_SQR	

Function overview: Execute square root extraction for the input (IN) with coefficient, and output (OUT) the result.

Function/FB division name: General process FB_Arithmetic operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

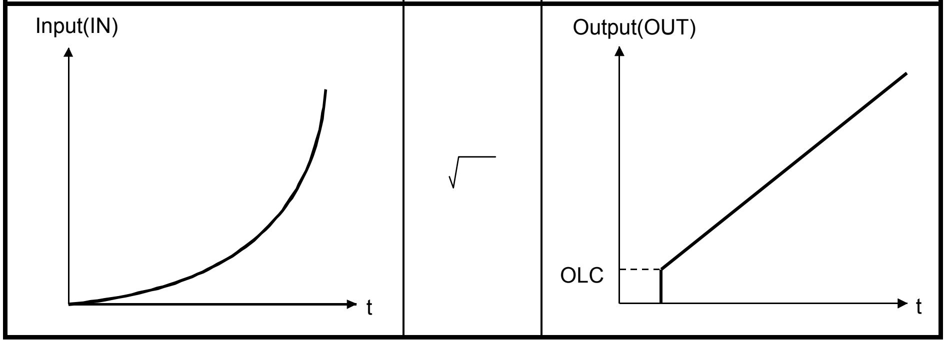
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	OLC	Public variable	REAL	Output low cut-off value	0 to 999999	0.0	User
	K	Public variable	REAL	Coefficient	0 to 999999	10.0	User

Function

Item	Contents
Square root processing	For the input value from the input variable IN, perform square root extraction with coefficient, and output from variable OUT.

(1) Processing contents
Execute the following operation.

Input (IN)	Processing	Output (OUT)

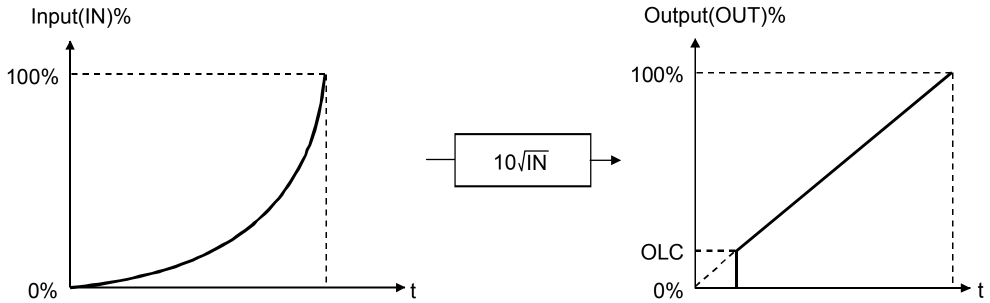


Input (IN)	Output low cut-off value (OLC)	Output (OUT)
IN≧0	$K\times\sqrt{IN} > OLC$	$OUT = K\times\sqrt{IN}$
	$k\times\sqrt{IN} \leqq OLG$	OUT = 0
IN<0		OUT = 0

IN: Input value, K: Coefficient

POINT

(1) When input (IN) is in percentage (%), please set the coefficient (K) to K=10.
Extract square root with K=10($10\sqrt{IN}$), make 0 to 100% of input (IN) corresponding to 0 to 100% of output (OUT). (When input=100%, output=$10\sqrt{100}$=100%).



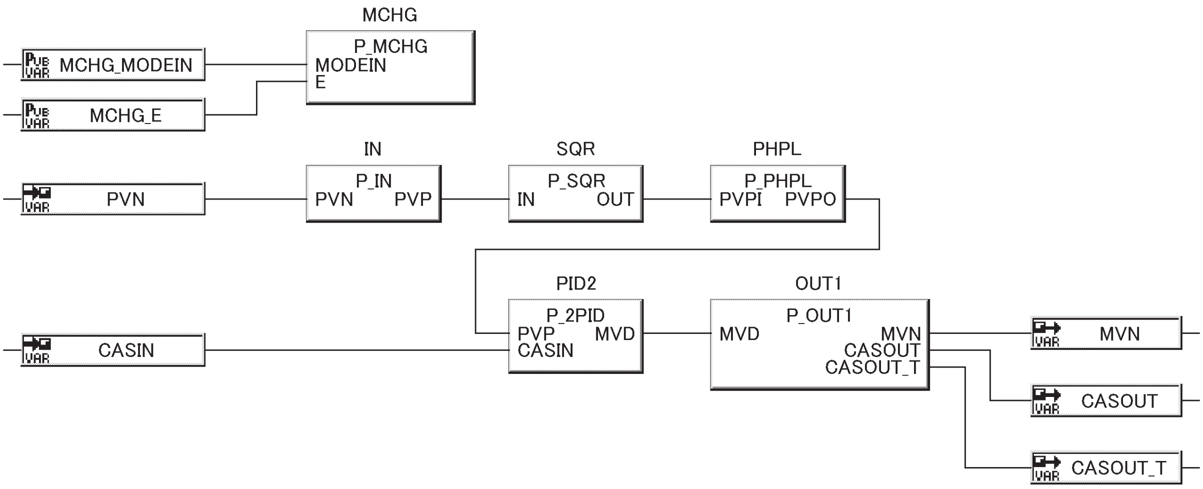
(2) Generally, output low cut-off value (OLC) is applied when input (IN) is in percentage (%). (Output low cut-off value (OLC) shall be set in 10 (%) level)

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

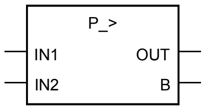


POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

7.3 General Process FB_Comparison Operation FB

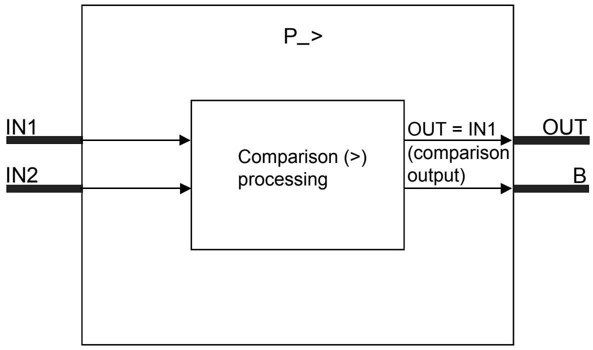
7.3.1 Compare Greater Than (With Setting Value) (P_>)

FB	FBD parts
P_>	

Function overview: Compare (>) input1 (IN1) with input2 (IN2) using setting value and hysteresis, and output result from comparison output (B). Additionally, the input1 (IN1) is always output from OUT.

Function/FB division name: General process FB_Comparison operation FB

Block Diagram



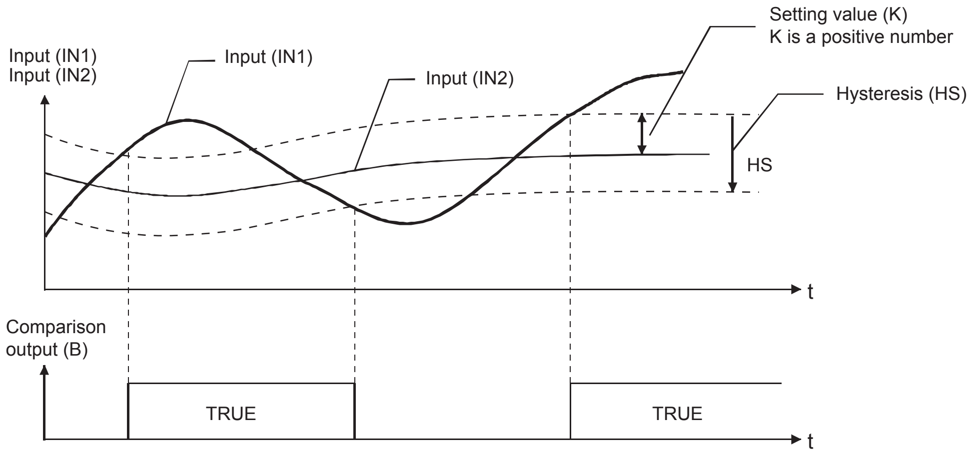
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN1	Input variable	REAL	Input1	-999999 to 999999
	IN2	Input variable	REAL	Input1	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999
	B	Output variable	BOOL	Comparison Output	TRUE, FALSE

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	K	Public variable	REAL	Setting value	-999999 to 999999	0.0	User
	HS	Public variable	REAL	Hysteresis	0 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Comparison processing (>)</td><td>
Compare (>) the input value from the input variable IN1 with that from the input variable 2 with setting value and hysteresis, then output the comparison result from output variable B.
(At the same time, input variable IN1 is always output from variable OUT.)


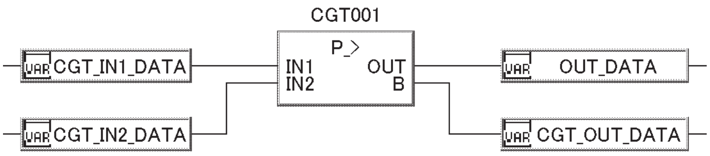
(1) Processing contents
Execute the following operation.
<table>
<tr><th>Condition</th><th>Comparison output (B)</th><th>Output (OUT)</th></tr>
<tr><td>IN1>IN2+K</td><td>B=TRUE</td><td rowspan="3">OUT=IN1</td></tr>
<tr><td>IN1≦IN2+K-HS</td><td>B=FALSE</td></tr>
<tr><td>IN2+K-HS<IN1≦IN2+K</td><td>Previous value</td></tr>
</table>
IN1: Input value1, IN2: Input value 2, K: Setting value, HS: Hysteresis
</td></tr>
</table>

Error

Error may occur in the following case, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

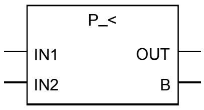
- When overflow occurs during operation. (Error code: Refer to Appendix 2)
- When hysteresis (HS)<0. (Error code: Refer to Appendix 2)

Program Example



> **POINT**
>
> It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

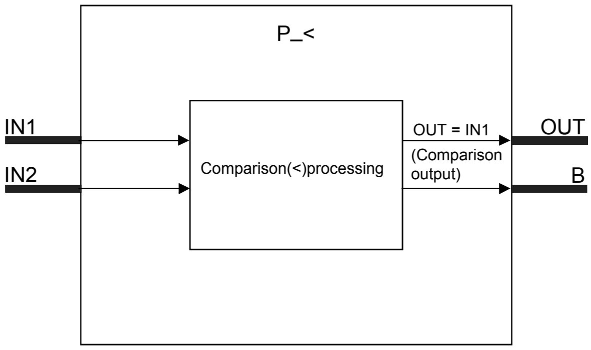
7.3.2 Compare Less Than (With Setting Value) (P_ <)

FB	FBD parts
P_<	

Function overview: Compare input1 (IN1) with input2 (IN2) using setting value and hysteresis (<), and output from comparison output (B). Additionally, the input1 (IN1) is always output from OUT.

Function/FB division name: General process FB_Comparison operation FB

Block Diagram



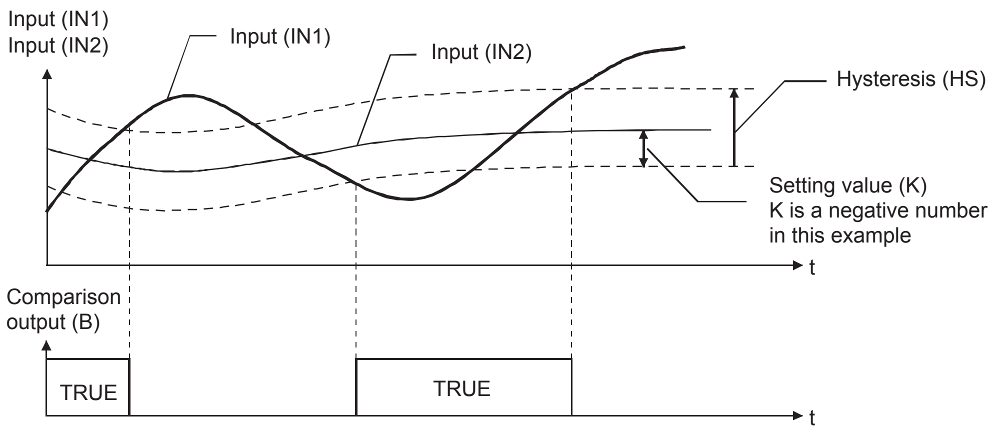
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN1	Input variable	REAL	Input1	-999999 to 999999
	IN2	Input variable	REAL	Input1	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999
	B	Output variable	BOOL	Comparison Output	TRUE, FALSE

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	K	Public variable	REAL	Setting value	-999999 to 999999	0.0	User
	HS	Public variable	REAL	Hysteresis	0 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Comparison processing (<)</td><td>
Compare (<) the input value from the input variable IN1 with that from the input variable 2 with setting value and hysteresis, then output the comparison result from output variable B.
(At the same time, input variable IN1 is output from output variable OUT.)


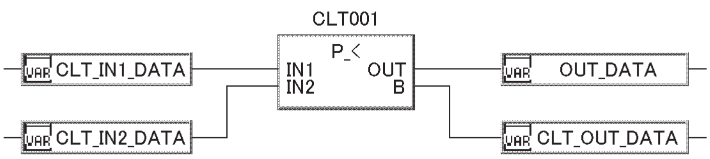
(1) Processing contents
Execute the following operation.
<table>
<tr><th>Condition</th><th>Comparison output (B)</th><th>Output (OUT)</th></tr>
<tr><td>IN1<IN2+K</td><td>B=TRUE</td><td rowspan="3">OUT=IN1</td></tr>
<tr><td>IN1≧IN2+K+HS</td><td>B=FALSE</td></tr>
<tr><td>IN2+K≦IN1+K+HS</td><td>Previous value</td></tr>
</table>
IN1: Input value1, IN2: Input value 2, K: Setting value, HS: Hysteresis
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)
- When hysteresis (HS)<0 (Error code: Refer to Appendix 2)

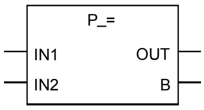
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

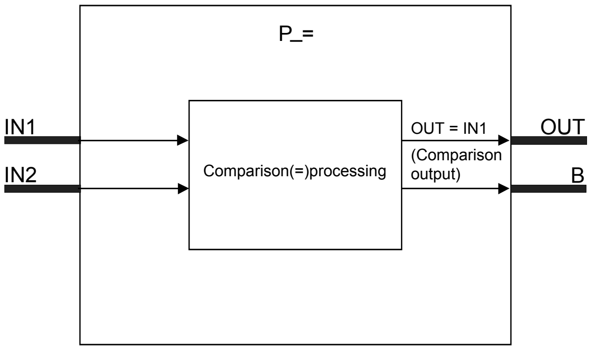
7.3.3 Compare Equal Than (With Setting Value) (P_=)

FB	FBD parts
P_=	

Function overview: Compare (=) input1 (IN1) with input2 (IN2) using setting value and hysteresis, and output result from comparison output (B). Additionally, input1 (IN1) is output from OUT.

Function/FB division name: General process FB_Comparison operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN1	Input variable	REAL	Input1	-999999 to 999999
	IN2	Input variable	REAL	Input2	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999
	B	Output variable	BOOL	Comparison Output	TRUE, FALSE

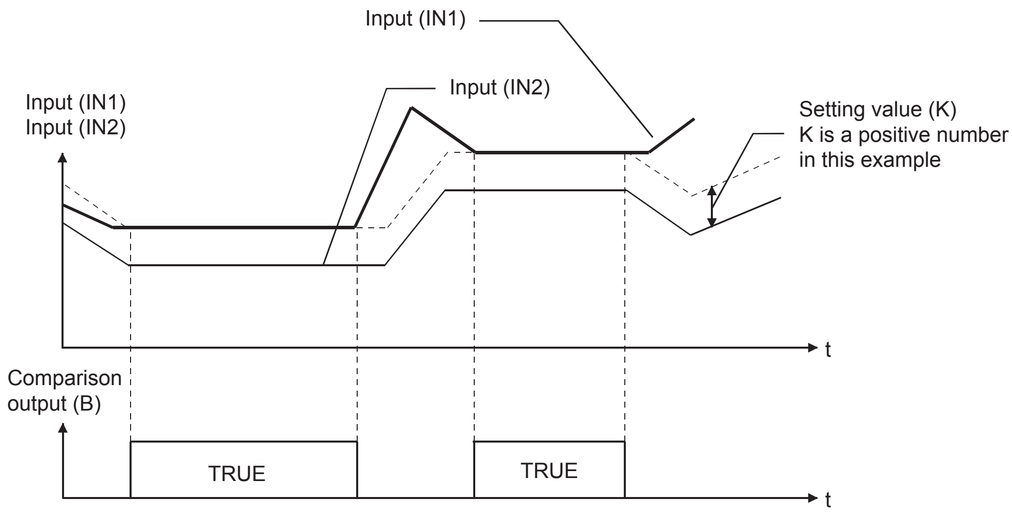
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	K	Public variable	REAL	Setting value	-999999 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Comparison processing (=)</td><td>

Compare (=) the input value from the input variable IN1 with that from the input variable 2 with setting value, and output the comparison result from output variable B.
At the same time, input variable IN1 is always output from output variable OUT.



(1) Processing contents
Execute the following operation.

Condition	Comparison output (B)	Output (OUT)
IN1=IN2+K	B=TRUE	OUT=IN1
IN1≠IN2+K	B=FALSE	

IN1: Input value1, IN2: Input value 2, K: Setting value

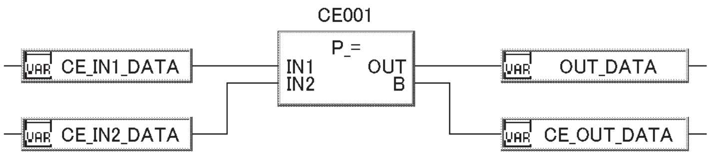
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

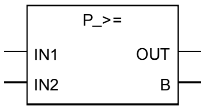
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

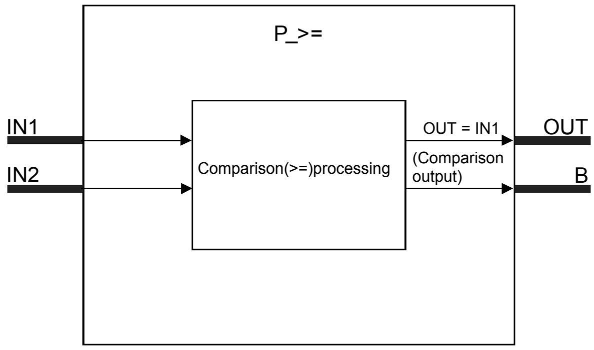
7.3.4 Compare Greater Or Equal (With Setting Value) (P_>=)

FB	FBD parts
P_>=	

Function overview: Compare (≧) input1 (IN1) with input2 (IN2) using setting value and hysteresis, and output from comparison output (B). Additionally, input1 (IN1) is output from output (OUT).

Function/FB division name: General process FB_Comparison operation FB

Block Diagram



Input and Output Pins

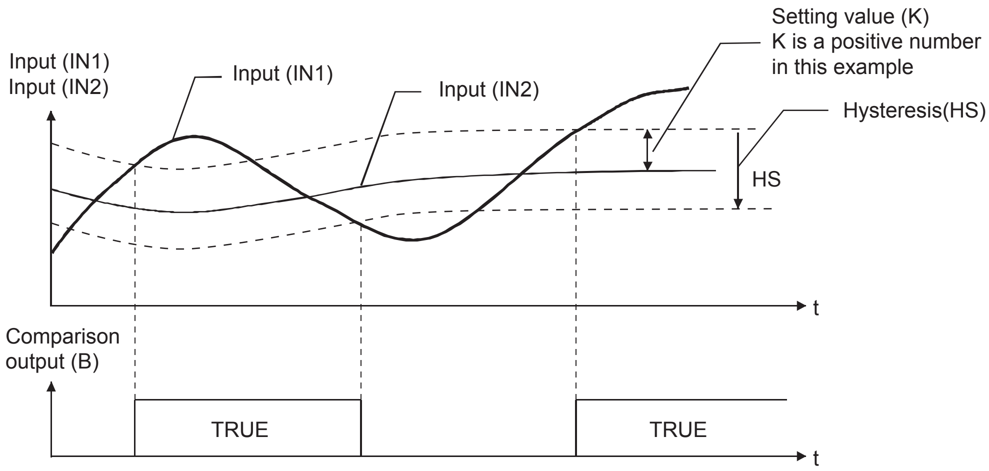
Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN1	Input variable	REAL	Input1	-999999 to 999999
	IN2	Input variable	REAL	Input1	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999
	B	Output variable	BOOL	Comparison Output	TRUE, FALSE

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	K	Public variable	REAL	Setting value	-999999 to 999999	0.0	User
	HS	Public variable	REAL	Hysteresis	0 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Comparison processing (≧)</td><td>
Compare (≧) the input value from the input variable IN1 with that from the input variable 2 with setting value and hysteresis, and output the comparison result from output variable B.
Additionally, input variable IN1 is always output from variable OUT.



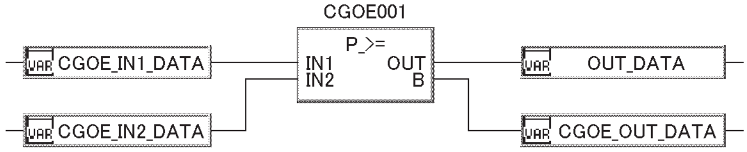
(1) Processing contents
Execute the following operation.
<table>
<tr><th>Condition</th><th>Comparison output (B)</th><th>Output (OUT)</th></tr>
<tr><td>IN1≧IN2+K</td><td>B=TRUE</td><td rowspan="3">OUT=IN1</td></tr>
<tr><td>IN1<IN2+K-HS</td><td>B=FALSE</td></tr>
<tr><td>IN2+K-HS≦IN1< IN2+K</td><td>Previous value</td></tr>
</table>
IN1: Input value 1, IN2: Input value 2, K: Setting value, HS: Hysteresis
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)
- When hysteresis (HS)<0 (Error code: Refer to Appendix 2)

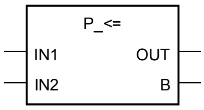
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

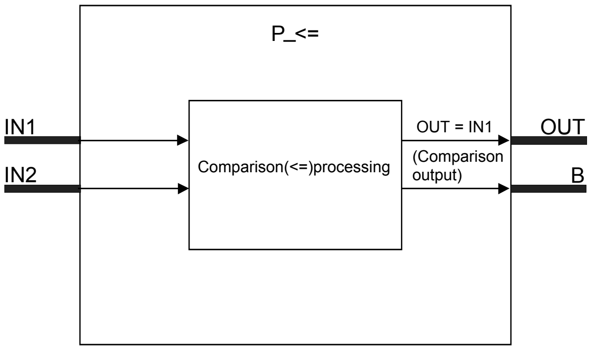
7.3.5 Compare Less Or Equal (With Setting Value) (P_<=)

FB	FBD parts
P_<=	

Function overview: Compare (≦) input1 (IN1) with input2 (IN2) using setting value and hysteresis, and output from comparison output (B). Additionally, input1 (IN1) is always output from output (OUT).

Function/FB division name: General process FB_Comparison operation FB

Block Diagram



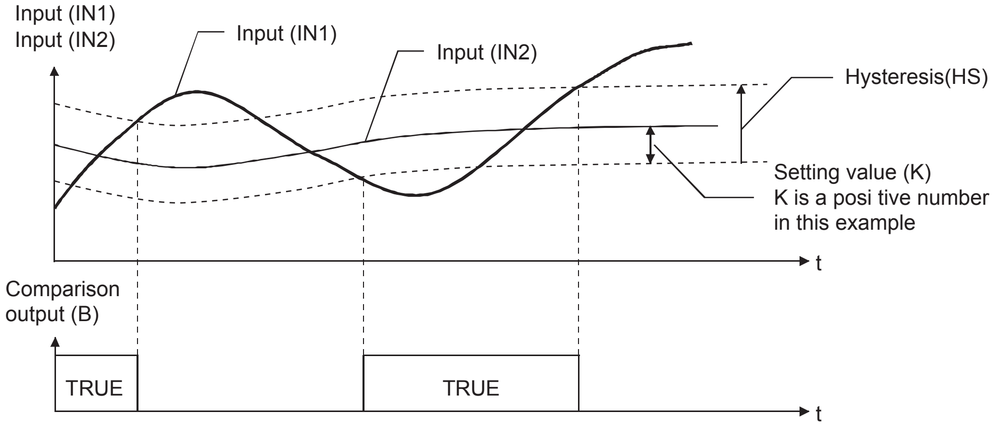
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN1	Input variable	REAL	Input1	-999999 to 999999
	IN2	Input variable	REAL	Input1	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999
	B	Output variable	BOOL	Comparison Output	TRUE, FALSE

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	K	Public variable	REAL	Setting value	-999999 to 999999	0.0	User
	HS	Public variable	REAL	Hysteresis	0 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Comparison processing (≦)</td><td>
Compare (≦) the input value from the input variable IN1 with that from the input variable 2 with setting value and hysteresis, and output the comparison result from output variable B.
Additionally, input variable IN1 is always output from variable OUT.


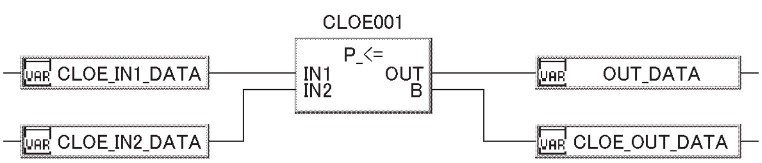
(1) Processing contents
Operate following items.
<table>
<tr><th>Condition</th><th>Comparison output (B)</th><th>Output (OUT)</th></tr>
<tr><td>IN1≦IN2+K</td><td>B=TRUE</td><td rowspan="3">OUT=IN1</td></tr>
<tr><td>IN1>IN2+K+HS</td><td>B=FALSE</td></tr>
<tr><td>IN2+K<IN1≦IN2+K+HS</td><td>Previous value</td></tr>
</table>
IN1: Input value1, IN2: Input value 2, K: Setting value, HS: Hysteresis
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)
- When hysteresis (HS)<0. (Error code: Refer to Appendix 2)

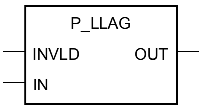
Program Example



> **POINT**
>
> It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

7.4 General Process FB_Control Operation FB

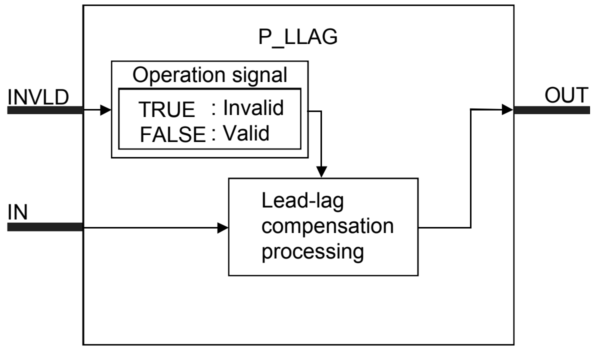
7.4.1 Lead-Lag (P_LLAG)

FB	FBD parts
P_LLAG	

Function overview: When operation signal (INVLD) is FALSE, perform lead-lag compensation for input (IN), and output (OUT) the result.

Function/FB division name: General process FB_ Control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	INVLD	Input variable	BOOL	Operation signal (TRUE: Invalid, FALSE: valid)	TRUE, FALSE
	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	T1	Public variable	REAL	Lag time (lag time constant) (unit: s)	0 to 999999	1.0	User
	T2	Public variable	REAL	Lead time (lead time constant) (unit: s)	0 to 999999	1.0	User

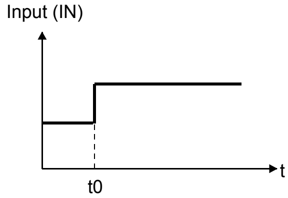
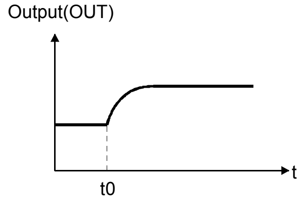
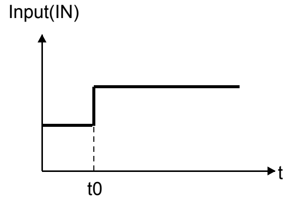
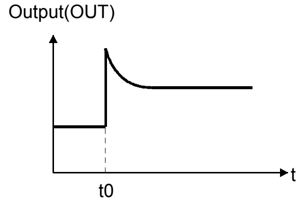
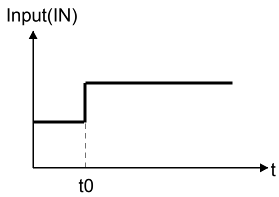
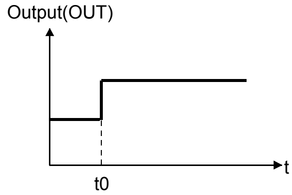
Function

Item	Contents

Lead-lag compensation

When operation signal (INVLD) is FALSE, perform lead-lag compensation for the value from input variable (IN), and output from variable OUT.

(1) Processing contents
Execute the following operation.

Condition		Input (IN)	Lead-lag compensation	Output (OUT)
Operation signal (INVLD)	Lag time (T1) lead time (T2)			
FALSE	T1>T2		$\frac{1 + T2 \cdot S}{1 + T1 \cdot S}$	
	T1<T2		$\frac{1 + T2 \cdot S}{1 + T1 \cdot S}$	
TRUE	T1>T2 T1<T2		None	Output value = Input value 

Operation signal (INVLD)	Output (OUT)
FALSE (valid)	$OUT = \frac{1}{T1 + \triangle T}\{[T2 \times (IN - INn\text{-}1)] + (T1 \times OUTn\text{-}1) + (\triangle T \times IN)\}$ ● When T1+△T=0,OUT=0 ● When lead time (lead time constant) T2=0, it will be a first-order lag
TRUE (invalid)	OUT=IN

IN: Input value, OUT: Output value, INn+1: Previous input value, OUTn-1: Previous output value
T1: Lag time (lag time constant)(s), T2: Lead time (lead-time constant)(s), △T: Execution cycle(s),
S: Laplace operator

Item	Contents
Operation signal	When operation signal (INVLD) is FALSE: Lead-lag compensation is valid When operation signal (INVLD) is TRUE: Lead-lag compensation is invalid

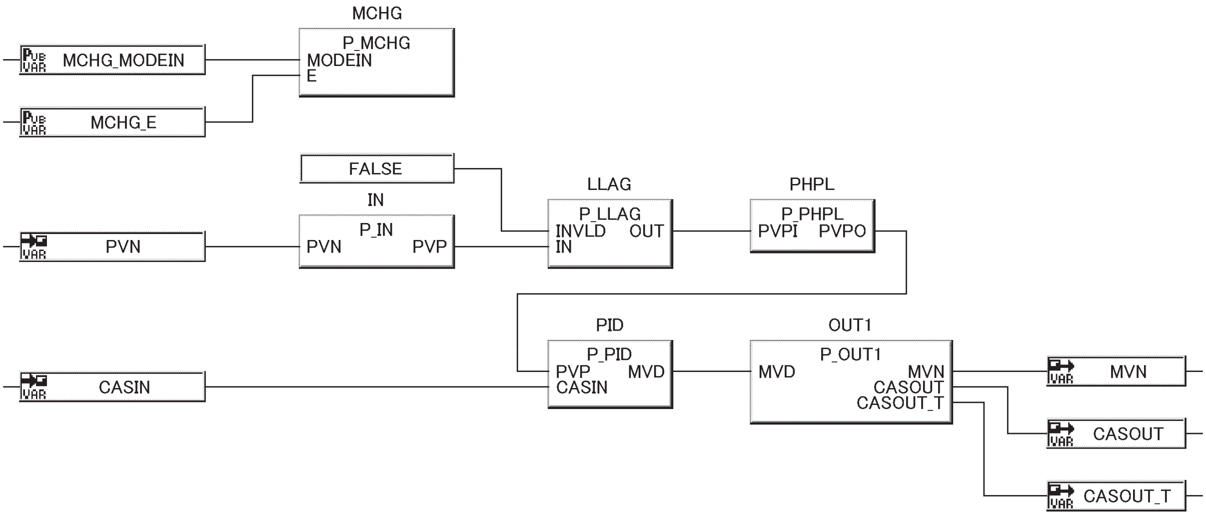
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

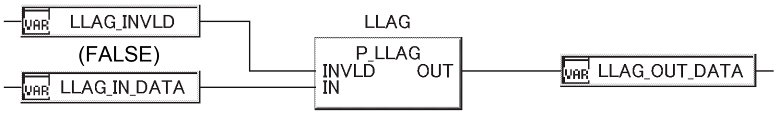
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(1) Program Example 1



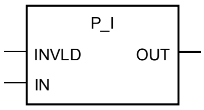
(2) Program Example 2



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

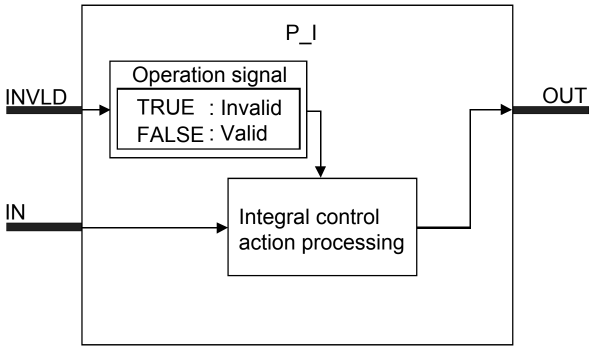
7.4.2 Integral (P_I)

FB	FBD parts
P_I	

Function overview: When operation signal (INVLD) is FALSE, perform integral control action for input (IN), and output (OUT) the result.

Function/FB division name: General process FB_Control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	INVLD	Input variable	BOOL	Operation signal (TRUE: Invalid, FALSE: valid)	TRUE, FALSE
	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

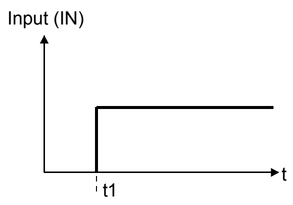
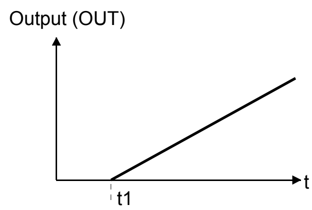
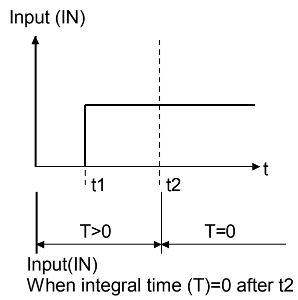
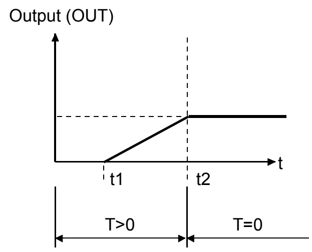
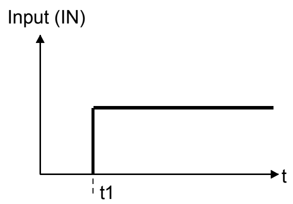
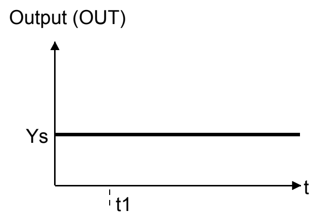
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	T	Public variable	REAL	Integral time (unit: s)	0 to 999999	1.0	User
	Ys	Public variable	REAL	Initial output value	-999999 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Integral control action processing</td><td>
When operation signal (INVLD) is FALSE, perform integral control action for the input value from input variable IN, and output from variable OUT.

(1) Processing contents
Execute the following operation.

<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Input (IN)</th><th rowspan="2">Integral control action</th><th rowspan="2">Output (OUT)</th></tr>
<tr><th>Operation signal (INVLD)</th><th>Integral time (T)</th></tr>
<tr><td rowspan="2">FALSE</td><td>T1 > 0</td><td>
</td><td>$\frac{1}{T \cdot S}$</td><td>
</td></tr>
<tr><td>T = 0</td><td>
</td><td>$\frac{1}{T \cdot S}$</td><td>
</td></tr>
<tr><td>TRUE</td><td>T ≧ 0</td><td>
</td><td>Ys</td><td>
</td></tr>
</table>

<table>
<tr><th>Operation signal (INVLD)</th><th>Integral time (T)</th><th>Output (OUT)</th></tr>
<tr><td>FALSE (valid)</td><td>T > 0</td><td>$OUT = \frac{\triangle T \times IN}{T} + OUTn\text{-}1$</td></tr>
<tr><td>FALSE (valid)</td><td>T = 0</td><td>OUT = OUTn-1</td></tr>
<tr><td>TRUE (invalid)</td><td></td><td>OUT = Ys</td></tr>
</table>

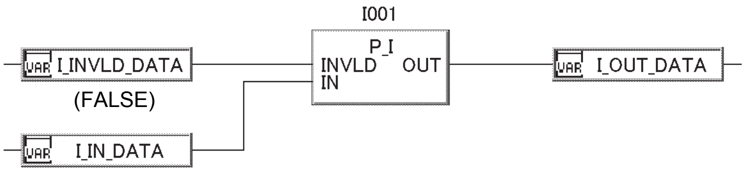
△T: Execution cycle, T: Integral time(s), IN: Input value, OUT: Output value, OUT_{n-1}: Previous output value, Ys: Initial output value, S: Laplace operator.
</td></tr>
<tr><td>Operation signal</td><td>When operation signal (INVLD) is FALSE: Integral control action is valid
When operation signal (INVLD) is TRUE: Integral control action is invalid</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

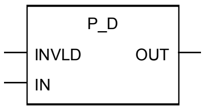
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

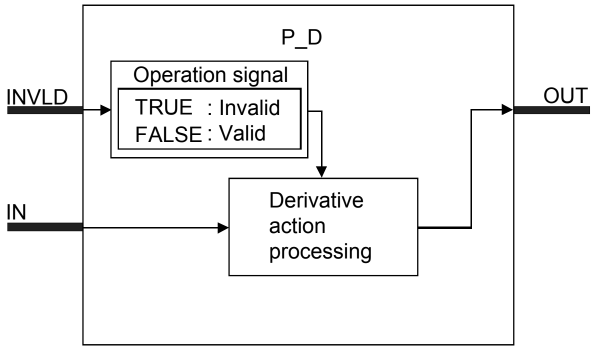
7.4.3 Derivative (P_D)

FB	FBD parts
P_D	

Function overview: When operation signal (INVLD) is FALSE, perform derivative action for input (IN), and output (OUT) the result.

Function/FB division name: General process FB_Control operation FB

Block Diagram



Input and Output Pins

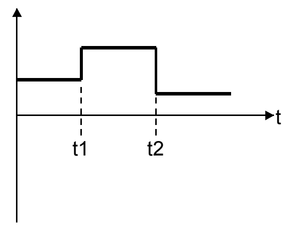
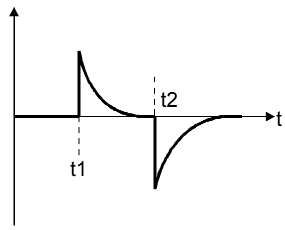
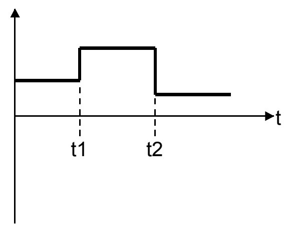
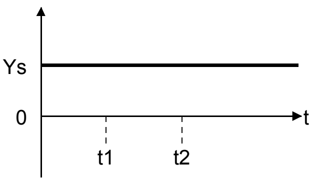
Pin	Variable name	Variable type	Data type	Contents	Range
Input	INVLD	Input variable	BOOL	Operation signal (TRUE: Invalid, FALSE: valid)	TRUE, FALSE
	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	T	Public variable	REAL	Derivative time (unit: s)	0 to 999999	1.0	User
	Ys	Public variable	REAL	Initial output value	-999999 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Derivative action</td><td>
When operation signal (INVLD) is FALSE, execute derivative action for the value input from input (IN), and output from variable OUT.

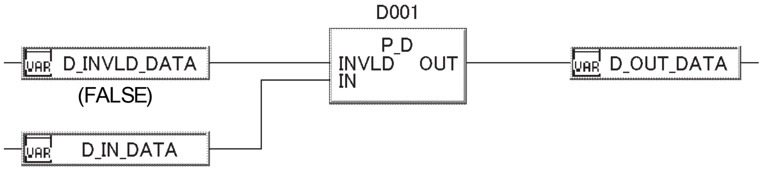
(1) Processing contents
Execute the following operation.
<table>
<tr><th>Condition
Operation signal (INVLD)</th><th>Input (IN)</th><th>Integral control action</th><th>Output (OUT)</th></tr>
<tr><td>FALSE</td><td>Input (IN)
</td><td>$\frac{T \cdot S}{1 + T \cdot S}$</td><td>Output (OUT)
</td></tr>
<tr><td>TRUE</td><td>Input (IN)
</td><td>Ys</td><td>Output (OUT)
</td></tr>
</table>
<table>
<tr><th>Operation signal (INVLD)</th><th>Output (OUT)</th></tr>
<tr><td>FALSE (valid)</td><td>$OUT = \frac{T}{T + \triangle T} \times (OUT_{n-1} - IN_{n-1} + IN)$</td></tr>
<tr><td>TRUE (invalid)</td><td>OUT=Ys</td></tr>
</table>
$\triangle T$: Execution cycle, T: Derivative time (s), IN: Input value, IN_{n-1}: Previous input value, OUT: Output value, OUT_{n-1}: Previous output value, Ys: Initial output value, S: Laplace operator.
</td></tr>
<tr><td>Operation signal</td><td>When operation signal (INVLD) is FALSE: Derivative action is valid
When operation signal (INVLD) is TRUE: Derivative action is invalid</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

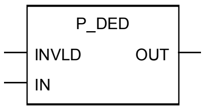
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

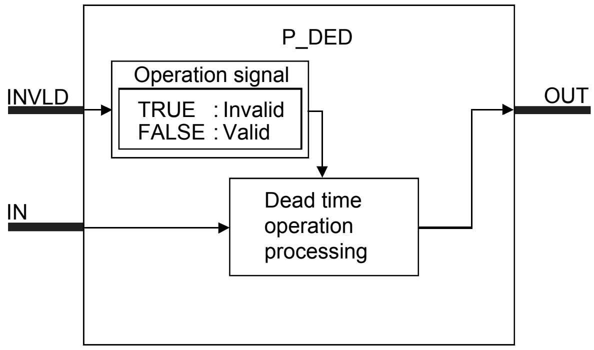
7.4.4 Dead Time (P_DED)

FB	FBD parts
P_DED	

Function overview: When operation signal (INVLD) is FALSE, execute invalid time operation for input (IN), and output (OUT).

Function/FB division name: General process FB_Control operation FB

Block Diagram



Input and Output Pins

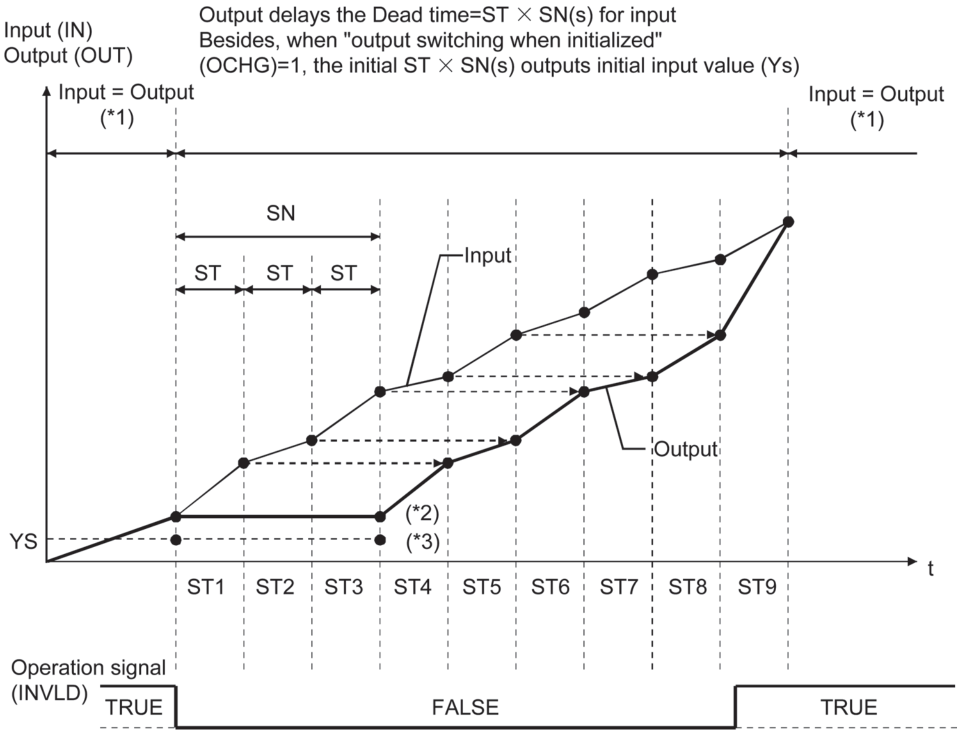
Pin	Variable name	Variable type	Data type	Contents	Range
Input	INVLD	Input variable	BOOL	Operation signal (TRUE: Invalid, FALSE: valid)	TRUE, FALSE
	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	ST	Public variable	REAL	Data collection interval (unit: s)	0 to 9999	1.0	User
	SN	Public variable	INT	Sampling number	0 to 48	0	User
	Ys	Public variable	REAL	Initial output value	-999999 to 999999	0.0	User
	OCHG	Public variable	INT	Output switching when initialized	0,1	0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Dead time processing</td><td>
When operation signal (INVLD) is FALSE, make the value from input variable IN delayed as invalid time, and output from variable OUT.

(Example) When sampling number (SN)=3


(1) Processing contents
Execute the following operation.
<table>
<tr><th>Operation signal (INVLD)</th><th>Output switching when initialized (OCHG)</th><th>Dead time</th><th>Output (OUT)</th></tr>
<tr><td>TRUE (invalid)</td><td>Random (0 or 1)</td><td></td><td>Output (OUT) the input (IN) value. (The above figure*1)</td></tr>
<tr><td rowspan="2">FALSE (valid)</td><td>0</td><td>ST × SN</td><td>From the beginning of operation to the SN time, output the input value (IN) when the operation signal (INVLD) TRUE→FALSE. (The above figure*2)
After SN time, output OUT=IN_{N-SN}.</td></tr>
<tr><td>1</td><td>ST × SN</td><td>From the beginning of operation to the SN time, output (OUT) the initial output value (Ys)(the above figure*3).
After SN time, output OUT=IN_{N-SN}.</td></tr>
</table>
INVLD: Operation signal, OCHG: Output switching when initialized, Ys: Initial output value, IN: Input value, OUT: Output value, ST: Data collection interval(s), SN: Sampling number

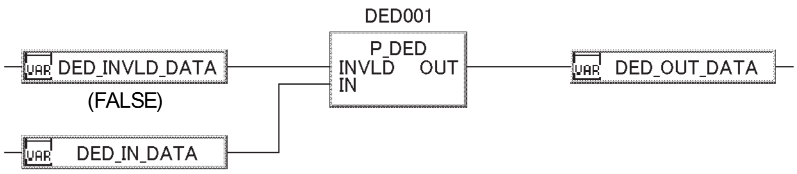
● When sampling number (SN)=0, output (OUT) value =input (IN) value.
</td></tr>
<tr><td>Operation signal</td><td>When operation signal (INVLD) is FALSE: Dead time operation is valid
When operation signal (INVLD) is TRUE: Dead time operation is invalid</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)
- When sampling number (SN): SN< 0 or SN> 48. (Error code: Refer to Appendix 2)

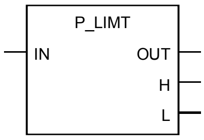
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

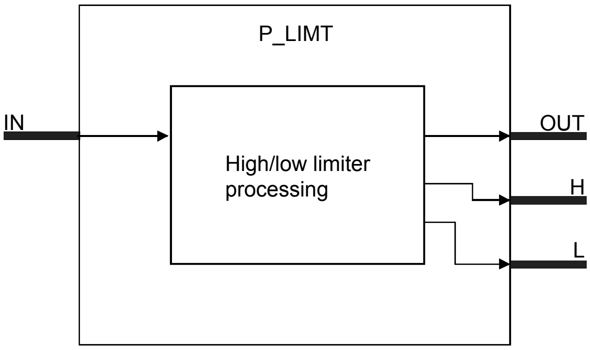
7.4.5 High/Low Limiter (P_LIMT)

FB	FBD parts
P_LIMT	

Function overview: For the input (IN), execute high/low limiter processing with hysteresis, and output (OUT) it.

Function/FB division name: General process FB_Control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
	OUT	Input variable	REAL	Output	-999999 to 999999
Output	H	Output variable	BOOL	High limit over detection (TRUE: Occur, FALSE: Reset)	TRUE, FALSE
	L	Output variable	BOOL	Low limit over detection (TRUE: Occur, FALSE: Reset)	TRUE, FALSE

Public Variable (Operation constant)

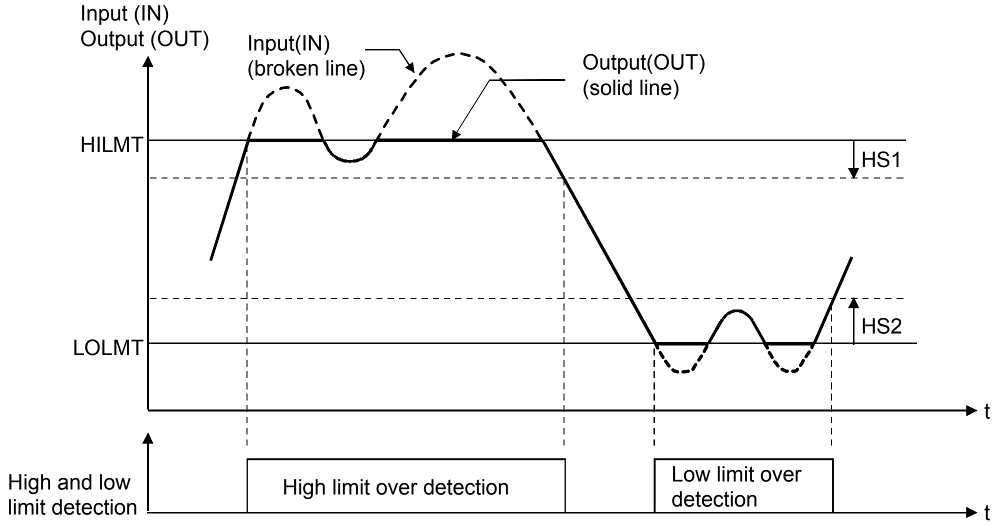
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	HILMT	Public variable	REAL	High limit value	-999999 to 999999	100.0	User
	LOLMT	Public variable	REAL	Low limit value	-999999 to 999999	0.0	User
	HS1	Public variable	REAL	High limit hysteresis	0 to 999999	0.0	User
	HS2	Public variable	REAL	Low limit hysteresis	0 to 999999	0.0	User

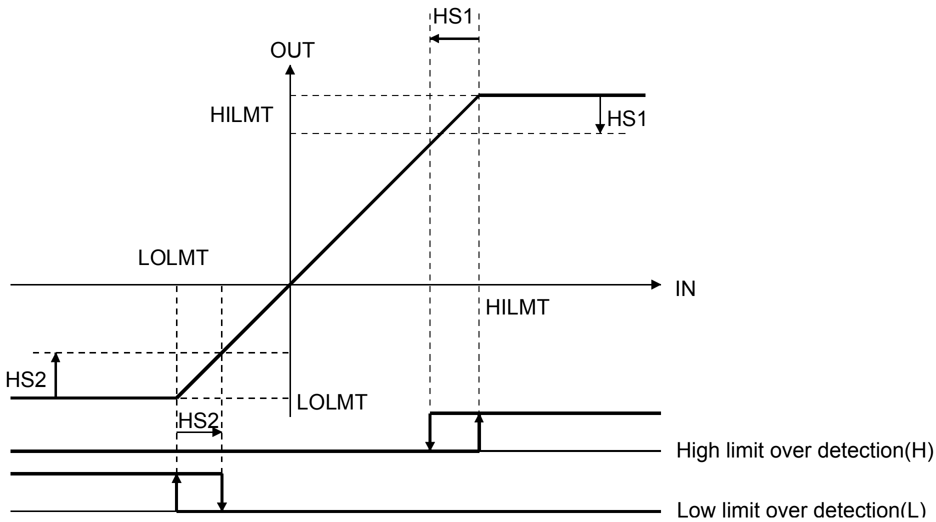
Function

Item	Contents

High/low limiter processing

For the value input from the variable IN, execute high/low limiter processing with hysteresis, and output it to output variable OUT.





IN: Input value, OUT: Output value, HILMT: High limit value, LOLMT: Lower limit value, HS1: High limit hysteresis, HS2: Low limit hysteresis.

(1) Processing contents
Operate following items.

Input (IN)	Output (OUT)	High limit OVER detection (H)	Low limit OVER detection(L)
IN≧HILMT	HILMT	TRUE (occur)	FALSE (reset)
LOLMT+HS2<IN <HILMT-HS1	IN	FALSE (reset)	FALSE (reset)
IN≦LOLMT	LOLMT	FALSE (reset)	TRUE (occur)
Others	IN	Previous value	Previous value

IN: Input value, OUT: Output value, HILMT: High limit value, LOLMT: low limit value, HS1: high limit hysteresis, HS2: low limit hysteresis.

(a) Please set high limit value (HILMT) and low limit value (LOLMT) as HILMT≧LOLMT.
(b) When HS1< 0 or HS2< 0, errors will occur.

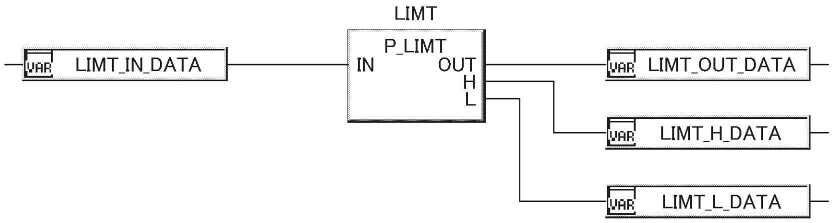
	HS2≧0	HS2<0
HS1≧ 0	Normal	Operation error
HS1< 0	Operation error	Operation error

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)
- When high limit hysteresis (HS1)<0, or lower limit hysteresis (HS2)<0. (Error code: Refer to Appendix 2)

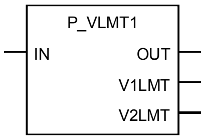
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

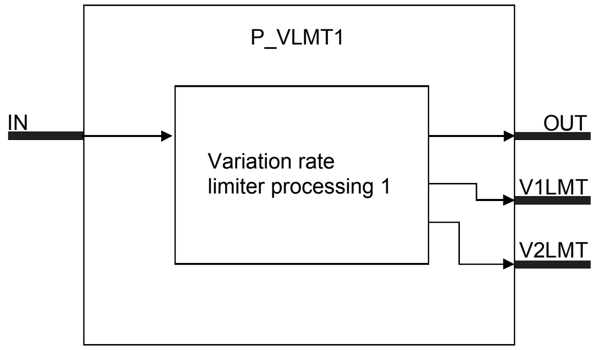
7.4.6 Variation Rate Limiter1 (P_VLMT1)

FB	FBD parts
P_VLMT1	

Function overview: Limit variation rate for input (IN) and outputs (OUT) the result.

Function/FB division name: General process FB_Control operation FB

Block Diagram



Input and Output Pins

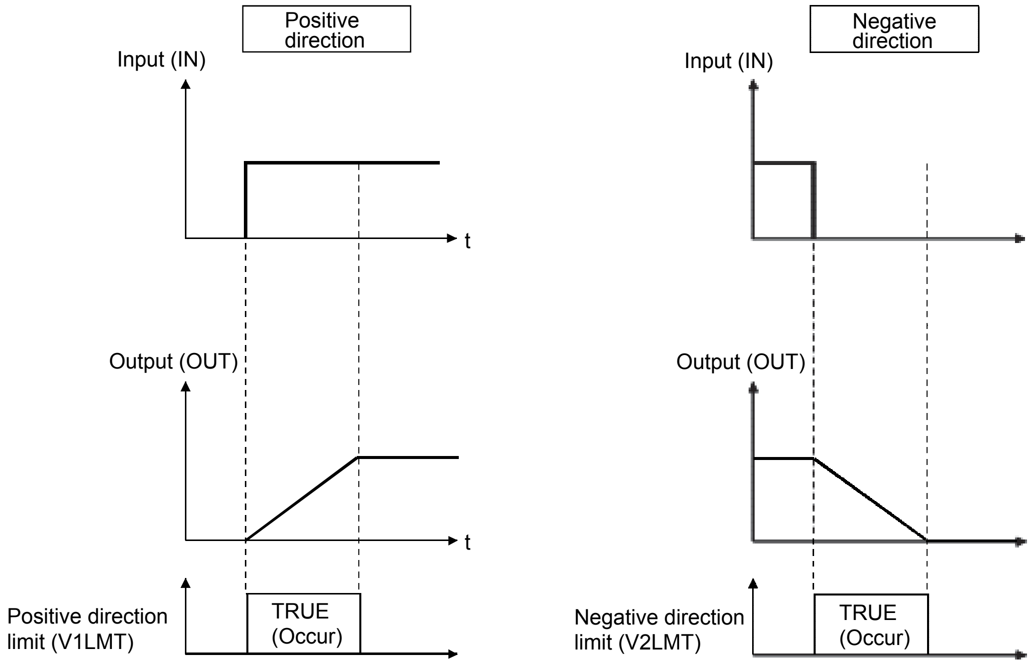
Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999
	V1LMT	Output variable	BOOL	Positive direction limit (TRUE: Occur, FALSE: Reset)	TRUE, FALSE
	V2LMT	Output variable	BOOL	Negative direction limit (TRUE: Occur, FALSE: Reset)	TRUE, FALSE

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	V1	Public variable	REAL	Positive direction limit value (unit: /s)	0 to 999999	100.0	User
	V2	Public variable	REAL	Negative direction limit value (unit: /s)	0 to 999999	100.0	User
	HS1	Public variable	REAL	Positive direction hysteresis	0 to 999999	0.0	User
	HS2	Public variable	REAL	Negative direction hysteresis	0 to 999999	0.0	User

Function

Item	Contents
Processing of variation rate limiter 1	Limit the variation rate for the input (IN) which is input from input variable and output the result from variable OUT.



(1) Processing contents
Execute the following operation.

		Input (IN-OUT)	Output (OUT)	Positive direction (V1LMT)	Negative direction (V2LMT)
Positive direction IN≧ OUT		IN-OUT≧ V1 × △T	OUT=OUT+V1 × △T	TRUE (occur)	FALSE (reset)
		IN-OUT<V1 × △T-HS1	OUT=IN	FALSE (reset)	FALSE (reset)
		Other	OUT=IN	Previous value	Previous value
Negative direction IN< OUT		OUT-IN≧V2 × △T	OUT=OUT-V2 × △T	FALSE (reset)	TRUE (occur)
		OUT-IN<V2 × △T-HS2	OUT=IN	FALSE (reset)	FALSE (reset)
		Other	OUT=IN	Previous value	Previous value

△T: Execution cycle, IN: Input value, OUT: Output value, V1: Positive direction limit value (/s), V2: Negative direction limit value (/s), HS1: Positive direction limit hysteresis, HS2: Negative direction limit hysteresis

(a) If HS1< 0 or HS2< 0, errors will occur.

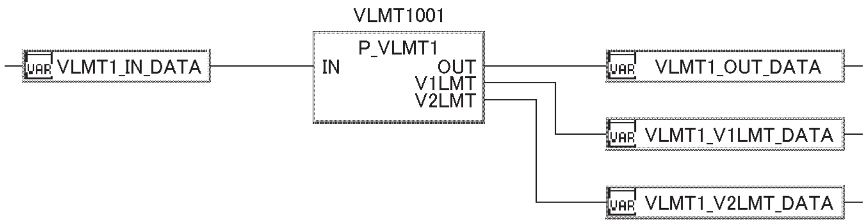
	HS2 ≧ 0	HS2< 0
HS1≧ 0	Normal	Operation error
HS1< 0	Operation error	Operation error

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)
- When positive direction hysteresis (HS1)<0, or negative direction hysteresis (HS2)<0. (Error code: Refer to Appendix 2)

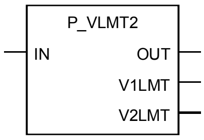
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

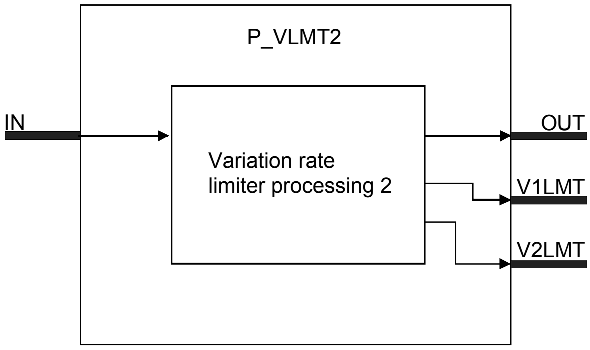
7.4.7 Variation Rate Limiter2 (P_VLMT2)

FB	FBD parts
P_VLMT2	

Function overview: Limit variation rate for input (IN) and output (OUT) it.

Function/FB division name: General process FB_Control operation FB

Block Diagram



Input and Output Pins

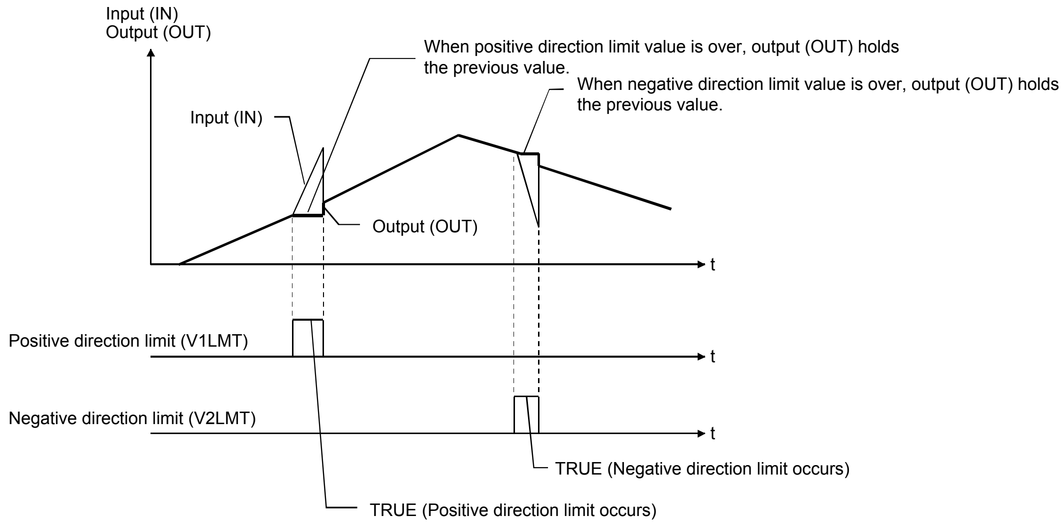
Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999
	V1LMT	Output variable	BOOL	Positive direction limit (TRUE: Occur, FALSE: Reset)	TRUE, FALSE
	V2LMT	Output variable	BOOL	Negative direction limit (TRUE: Occur, FALSE: Reset)	TRUE, FALSE

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	V1	Public variable	REAL	Positive direction limit value (unit: /s)	0 to 999999	100.0	User
	V2	Public variable	REAL	Negative direction limit value (unit: /s)	0 to 999999	100.0	User
	HS1	Public variable	REAL	Positive direction hysteresis	0 to 999999	0.0	User
	HS2	Public variable	REAL	Negative direction hysteresis	0 to 999999	0.0	User

Function

Item	Contents
Processing of variation rate limiter 2	Limit the variation rate for the input variable IN and output it to variable OUT.



(1) Processing contents

Execute the following operation

When the input (IN) variation rate is beyond limit value, limit the output (OUT) by the above mentioned method.

Hold the previous value if the variation rate limit value is exceeded.

Cancel the keeping of the previous value when within the variation rate limit value.

	Input (IN-OUT)	Output (OUT)	Positive direction (V1LMT)	Negative direction (V2LMT)
Positive direction IN≧ OUT	IN-OUT≧V1 × △T	OUT=OUT	TRUE (occur)	FALSE (reset)
	IN-OUT<V1 × △T-HS1	OUT=IN	FALSE (reset)	FALSE (reset)
	Other	OUT=OUT	Previous value	Previous value
Negative direction IN< OUT	OUT-IN≧V2 × △T	OUT=OUT	FALSE (reset)	TRUE (occur)
	OUT-IN<V2 × △T-HS2	OUT=IN	FALSE (reset)	FALSE (reset)
	Other	OUT=OUT	Previous value	Previous value

△T: Execution cycle, IN: Input value, OUT: Output value, V1: Positive direction limit value (/s),
V2: Negative direction limit value (/s), HS1: Positive direction hysteresis, HS2: Negative direction hysteresis

(a) When HS1< 0 or HS2< 0, errors will occur.

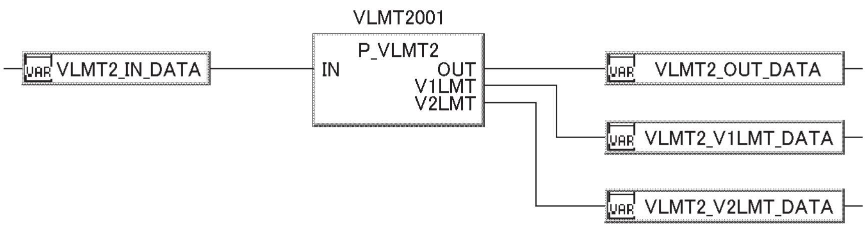
	HS2≧0	HS2<0
HS1≧0	Normal	Operation errors occur
HS1<0	Operation errors occur	Operation errors occur

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)
- When Positive direction hysteresis (HS1)<0, or Negative direction hysteresis (HS2)<0. (Error code: Refer to Appendix 2)

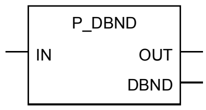
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

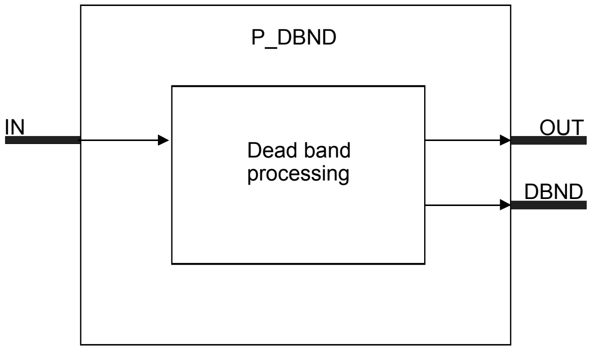
7.4.8 Dead Band (P_DBND)

FB	FBD parts
P_DBND	

Function overview: Set dead band for input (IN), and output (OUT) the result.

Function/FB division name: General process FB_Control operation FB

Block Diagram



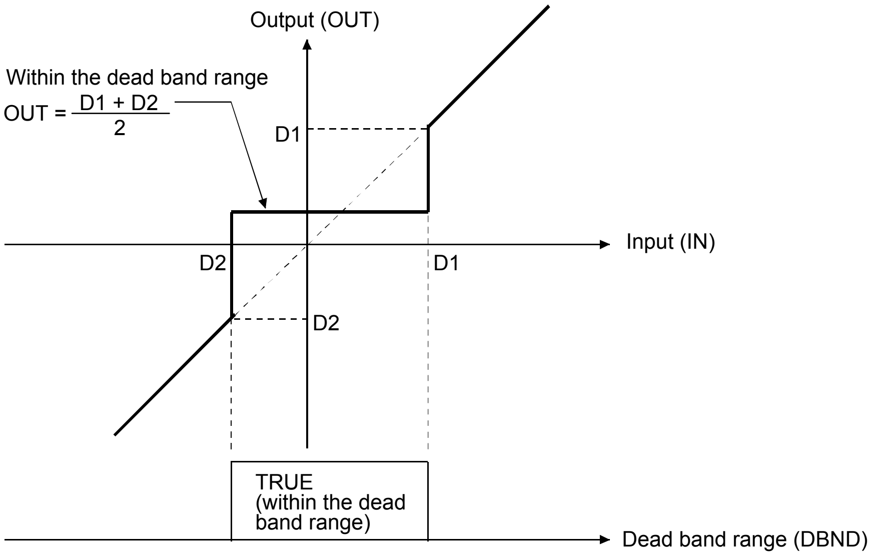
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	IN	Input variable	REAL	Input	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999
	DBND	Output variable	BOOL	Dead band range (TRUE: Within range FALSE: Out of range)	TRUE, FALSE

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	D1	Public variable	REAL	Dead band high limit	-999999 to 999999	0.0	User
	D2	Public variable	REAL	Dead band low limit	-999999 to 999999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Dead band processing</td><td>
Set dead band for input value from the variable (IN), and output it to the variable OUT.


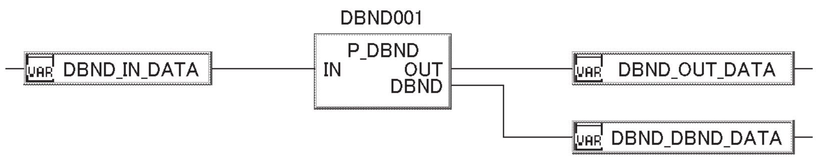
(1) Processing contents
Operate following items.
<table>
<tr><th>Input (IN)</th><th>Output (OUT)</th><th>Dead band range (DBND)</th></tr>
<tr><td>D2≦IN≦D1</td><td>$\frac{D1 + D2}{2}$</td><td>TRUE (within the range)</td></tr>
<tr><td>IN<D2 or IN>D1</td><td>IN</td><td>FALSE (Out of the range)</td></tr>
</table>
D1: Dead band high limit, D2: Dead band low limit, IN: Input value, OUT: Output value, DBND: Dead band range.
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

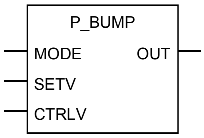
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

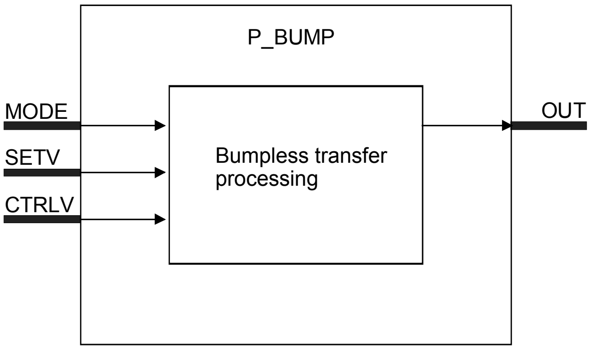
7.4.9 Bumpless Transfer (P_BUMP)

FB	FBD parts
P_BUMP	

Function overview: When mode (MODE) is changed from FALSE (MANUAL) to TRUE (AUTO), change the output from control value CTRLV to output setting value SETV smoothly.

Function/FB classification name: General process FB_Control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	MODE	Input variable	BOOL	Mode switching (TRUE: AUTO, FALSE: MANUAL)	TRUE, FALSE
	SETV	Input variable	REAL	Output setting value	-999999 to 999999
	CTRLV	Input variable	REAL	Output control value	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	T	Public variable	REAL	Delay Time (unit: s)	0 to 999999	1.0	User
	a	Public variable	REAL	Delay band	0 to 999999	1.0	User

Function

Item	Contents
Bumpless transfer processing	When input variable MODE (mode switching) changes from FALSE (MANUAL) → TRUE (AUTO), change the output value (OUT) from output control value CTRLV to output setting value SETV smoothly.

(1) Processing contents
Execute the following items.
(a) Output (OUT) approaches the output setting value (SETV) at a ratio set by delay time (T).
However, approach the output setting value (SETV) by first-order lag taking SETV as a bench mark if it is within the range set by delay band (a).

Condition		Xp	Output (OUT)
Mode switching (MODE)	|Xp|		
FALSE (MANUAL)	–	Xq=CTRLV-SETV Xp=CTRLV-SETV	OUT=CTRLV
TRUE (AUTO)	|Xp|>a	$Xp = Xp' - \frac{\Delta T}{T} \times Xq$	OUT=SETV + Xp When OUT=SETV, Xp=Xp' $\lvert Xp \rvert \leqq (\frac{\Delta T}{T}) \times \lvert Xq \rvert$
	|Xp|≦a	$Xp = \frac{T}{T + \Delta T} \times Xp'$	OUT=SETV+Xp When OUT=SETV, Xp=Xp' |Xp|≦0.0001

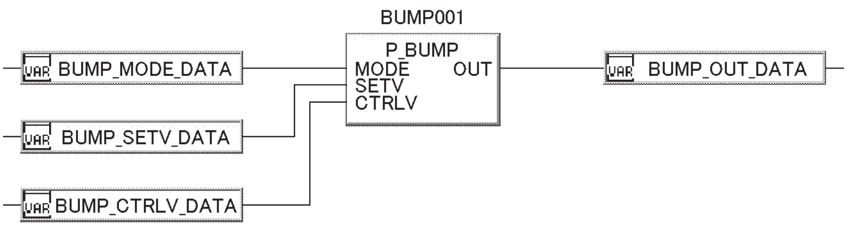
MODE: Mode switching, OUT: Output value, SETV: Output setting value, CTRLV: Output control value, Xq: Initial deviation, Xp: deviation △T: Execution cycle, T: Delay time(s), a: Delay band.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

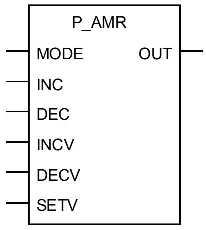
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

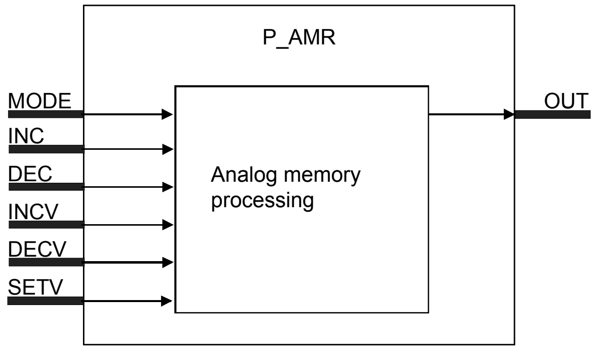
7.4.10 Analog Memory (P_AMR)

FB	FBD parts
P_AMR	

Function overview: Increase or decrease output (OUT) by certain ratio.

Function/FB division name: General process FB_Control operation FB

Block Diagram



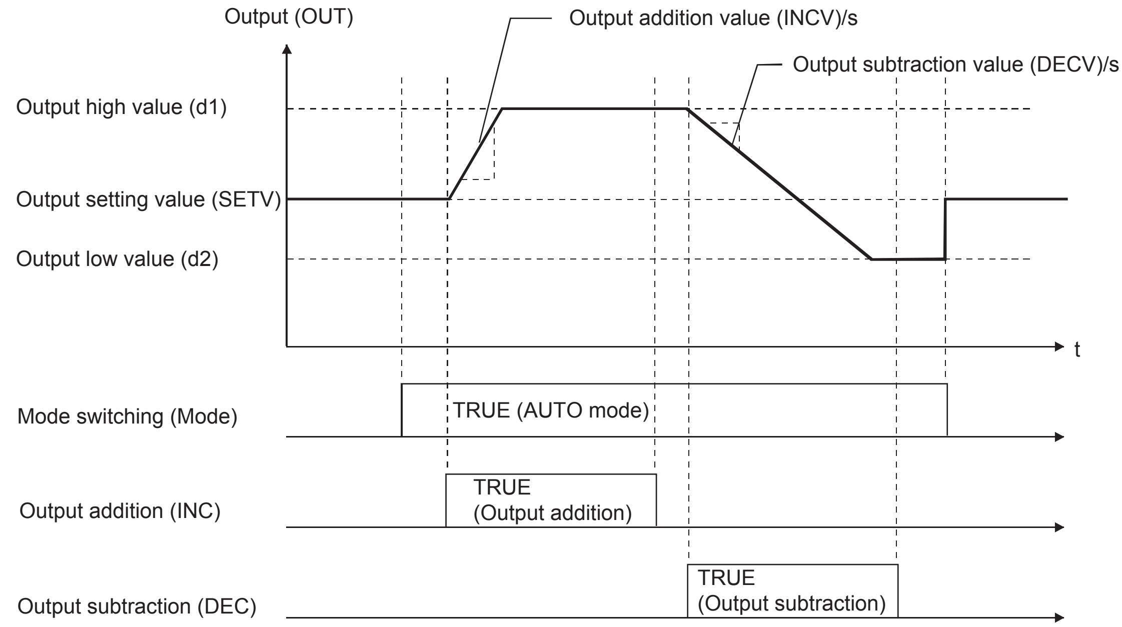
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	MODE	Input variable	BOOL	Mode switching (TRUE: AUTO, FALSE: MANUAL)	TRUE, FALSE
	INC	Input variable	BOOL	Output addition (TRUE: Used, FALSE: Not used)	TRUE, FALSE
	DEC	Input variable	BOOL	Output subtraction (TRUE: Used, FALSE: Not used)	TRUE, FALSE
	INCV	Input variable	REAL	Output addition value	-999999 to 999999
	DECV	Input variable	REAL	Output subtraction value	-999999 to 999999
	SETV	Input variable	REAL	Output setting value	-999999 to 999999
Output	OUT	Output variable	REAL	Output	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	d1	Public variable	REAL	Output high Limit value	0 to 999999	1.0	User
	d2	Public variable	REAL	Output low limit value	0 to 999999	1.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Analog memory processing</td><td>
Increase or decrease by certain ratio and output the result from the output variable OUT.


(1) Processing contents
Operate following items.
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (OUT)</th></tr>
<tr><th>Mode switching (MODE)</th><th>Output addition (INC)</th><th>Output subtraction (DEC)</th></tr>
<tr><td>FALSE (MANUAL)</td><td>—</td><td>—</td><td>OUT=SETV</td></tr>
<tr><td rowspan="4">TRUE (AUTO)</td><td>TRUE</td><td>FALSE</td><td>OUT=OUT+|INCV|×△T

When OUT is beyond d1, OUT=d1.</td></tr>
<tr><td>FALSE</td><td>TRUE</td><td>OUT=OUT-|DECV|×△T

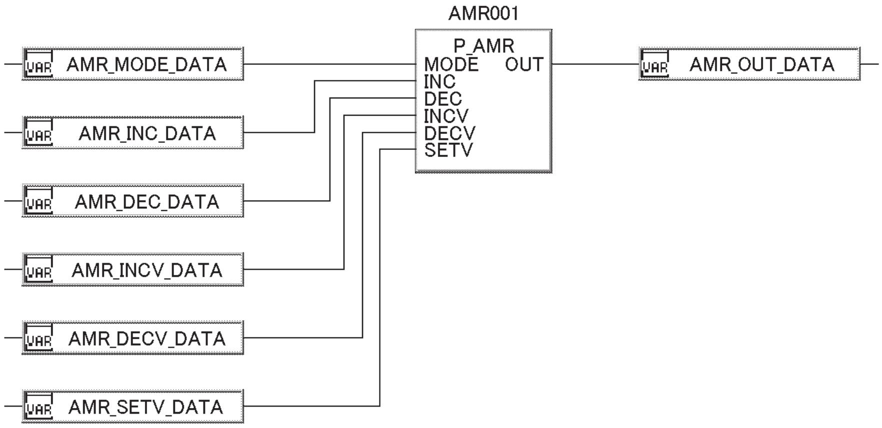
When OUT is below d2, OUT=d2.</td></tr>
<tr><td>TRUE</td><td>TRUE</td><td rowspan="2">OUT=OUT</td></tr>
<tr><td>FALSE</td><td>FALSE</td></tr>
</table>
MODE: Mode switching, OUT: Output value, INC: Output addition signal, DEC: Output subtraction signal, INCV: Output addition value, DECV: Output subtraction value, SETV: Output setting value, d1: Output high limit value, d2: Output low limit value, △T: Execution cycle
</td></tr>
</table>

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

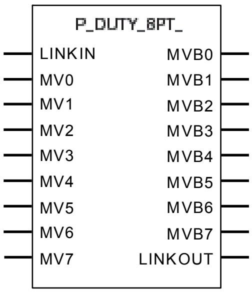
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

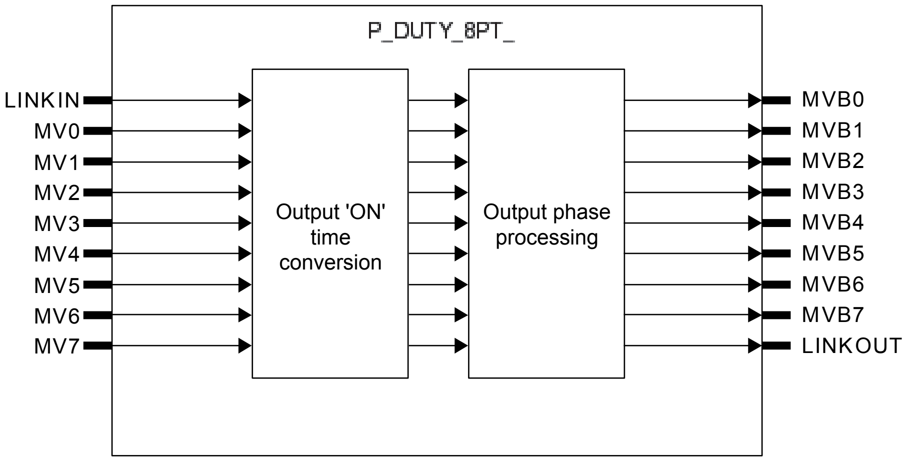
7.4.11 8 Points Time Proportional Output (P_DUTY_8PT_)

FB	FBD parts
P_DUTY_8PT_	

Function overview: For the input value, execute processing as output ON time conversion, and output in bit. Phase of output cycle is adjusted automatically to inhibit peak current.

Function/FB division name: General process FB_Control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	LINKIN	Input variable	ADR_REAL	Link input	–
	MV0 to MV7	Input variable	REAL	MV input (unit: %)	-10 to 110
Output	MVB0 to MVB7	Output variable	BOOL	Bit ON/OFF duty output	TRUE, FALSE
	LINKOUT	Output variable	ADR_REAL	Link output	–

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	PRIMARY	Public variable	BOOL	Lead FB specified	TRUE, FALSE	TRUE	User
	CTDUTY	Public variable	REAL	Control output cycle	0 to 9999	1.0	User
	DELAY	Public variable	REAL	Output ON delay time	0 to 9999	0.0	User

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output 'ON' time conversion /output phase processing</td><td>

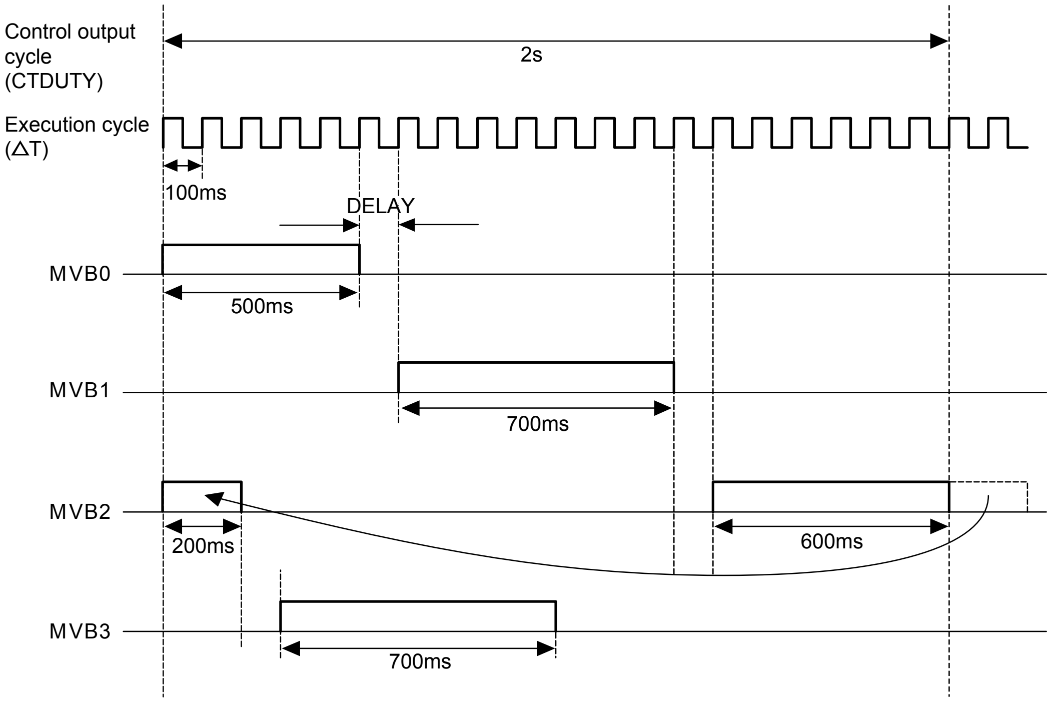
Capture an input value (MVn) every control output cycle (CTDUTY), and output a duty manipulated variable (MVBn) for input value. At the same time, adjust the phase of output cycle automatically to inhibit peak current.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Duty manipulated variable (MVBn) 'on' time</td><td>If the 'ON' time execution cycle count is defined as $\left(\frac{CTDUTY \times MVn}{\Delta T \times 100}\right)$ with the first digit after decimal point rounded off, then:
Duty manipulated variable (MVBn) 'on' time='on' time execution cycle count × △T</td></tr>
<tr><td>Duty manipulated variable (MVBn) 'off' time</td><td>If the 'off' time execution cycle count is defined as (execution cycle count in control cycle) –('on' time execution cycle count) then:
Duty manipulated variable (MVBn) 'off' time='off' time execution cycle count × △T</td></tr>
</table>

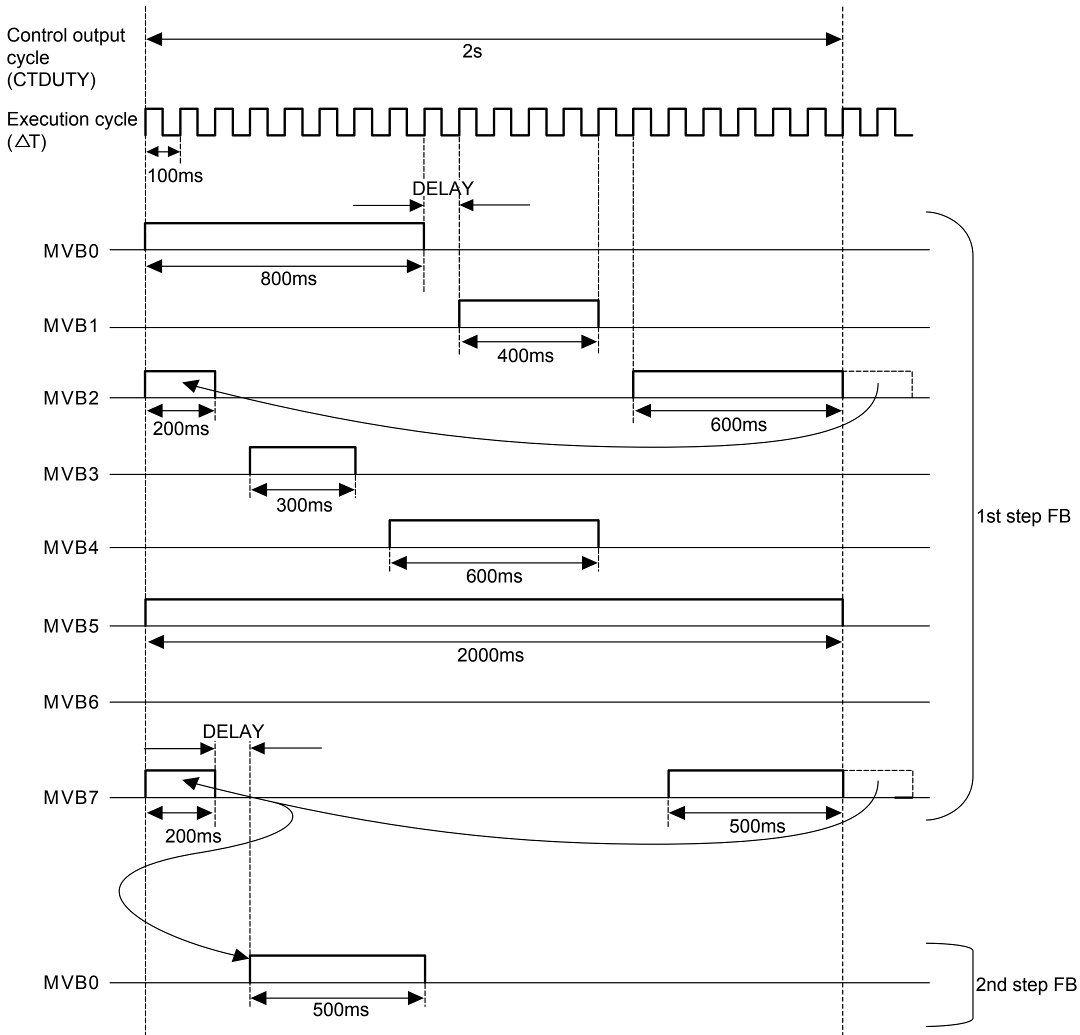
CTDUTY: Control output cycle, △T: Execution cycle, MVn: Input to nth pin (%),
MVBn: Output from nth pin (BOOL)

● Interval of bit ON/OFF duty output
Output of pin starts with delay for the time set with DELAY after the previous pin output OFF.
When the input value from an input pin is 0%, the delay time for the pin is ignored.

(Example) Execution cycle: △T=100ms, Control output cycle: CTDUTY = 2.0s
Input: MV0 = 25%, MV1 = 35%, MV2 = 40%, MV3 = 35%
Output ON Delay Time: DELAY = 0.1s
Lead FB specified: PRIMARY = TRUE



</td></tr>
</table>

Item	Contents
Output 'ON' time conversion /output phase processing (continued)	● When changing 9 or more bit ON/OFF duty output phases Concatenate this FB to change 9 or more bit ON/OFF duty output phases. When using this FB with concatenation, connect LINKOUT of lead FB with LINKIN of following FB, set PRIMARY of lead FB to TRUE, and PRIMARY of following FB to FALSE. The value of lead FB is applied to CTDUTY and DELAY of following FB. (Example) Execution cycle: △T=100ms ● 1st step FB Control output cycle: CTDUTY = 2.0s Input: MV0 = 40%, MV1 = 20%, MV2 = 40%, MV3 = 15%, MV4 = 30%, MV5 = 100%, MV6 = 0%, MV7 = 35% Output ON delay time: DELAY = 0.1s Lead FB specified: PRIMARY = TRUE ● 2nd step FB Control output cycle: same as lead FB CTDUTY(2.0s in this example) Input: MV0 = 25% Output ON delay time: same as lead FB DELAY(0.1s in this example) Lead FB specified: PRIMARY = FALSE 

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

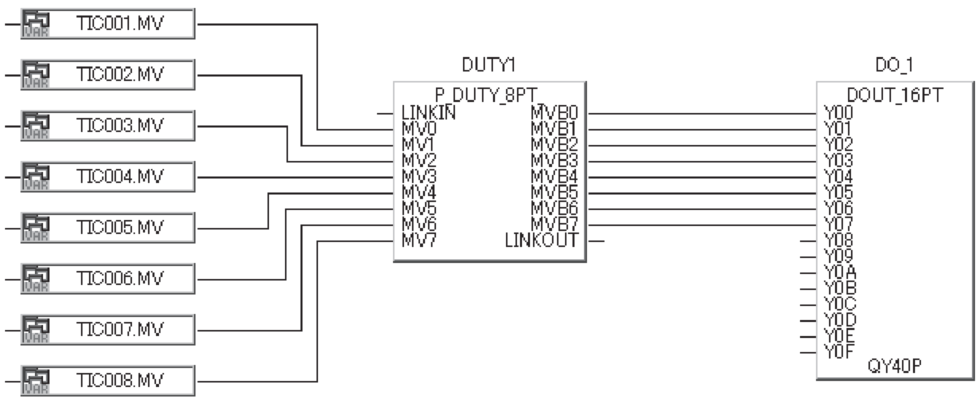
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(1) Single use of P_DUTY_8PT_

- Precautions on settings

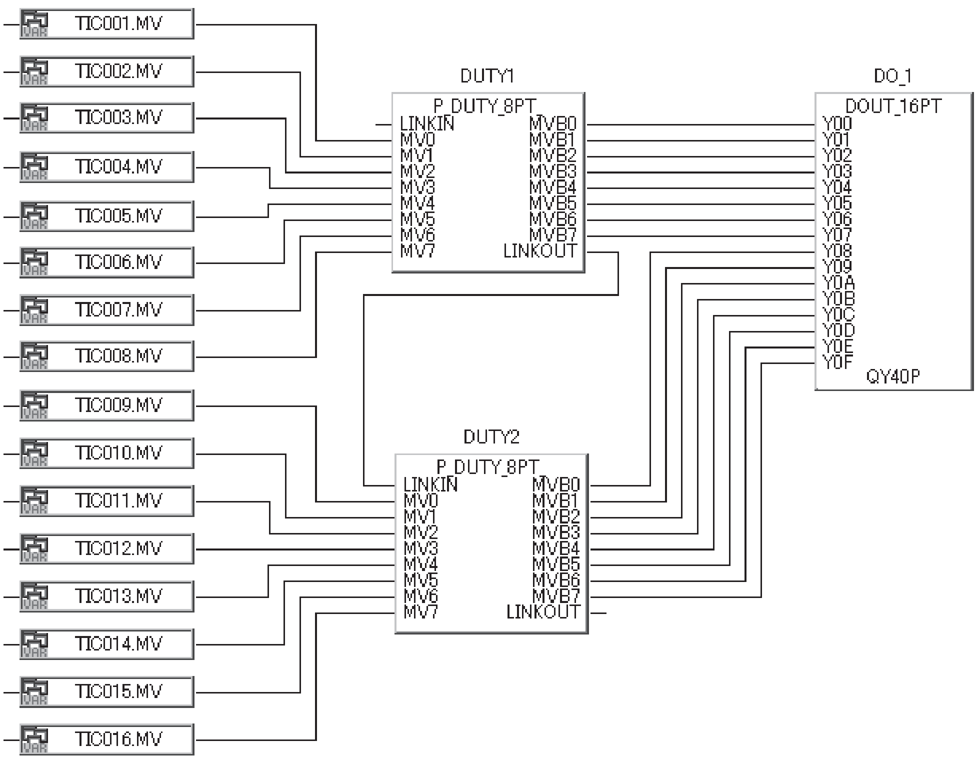
Variable type/Pin	Variable name	Contents	Setting/connection method
Public variable	PRIMARY	Lead FB specified	TRUE
Input pin	LINKIN	Link input	Not connected
Output pin	LINKOUT	Link output	Not connected



(2) When using this FB with concatenating P_DUTY_8PT_ (when changing 9 or more bit ON/OFF duty output phases)

● Precautions on settings

Target FB	Variable type/Pin	Variable name	Contents	Setting/connection method
Lead FB	Public variable	PRIMARY	Lead FB specified	TRUE
	Input pin	LINKIN	Link input	Not connected
	Output pin	LINKOUT	Link output	Connected with LINKIN of following FB
Following FB	Public variable	PRIMARY	Lead FB specified	FALSE
	Input pin	LINKIN	Link input	Connected with LINKOUT of lead FB
	Output pin	LINKOUT	Link output	Connected with LINKIN of following FB
Last FB	Public variable	PRIMARY	Lead FB specified	FALSE
	Input pin	LINKIN	Link input	Connected with LINKOUT of lead FB
	Output pin	LINKOUT	Link output	Not connected



POINT

- It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.
- When using this FB with concatenation, control output cycle and output ON delay time of following FB used in the operation execution can be checked with the value of public variable CTDUTY and DELAY of lead FB.

MEMO

INDEX

C-2

C-3

[I]

[L]

C-4

C-5

[N]

[O]

[P]

C-6

C-7

C-10

WARRANTY

Please confirm the following product warranty details before using this product.

1. Gratis Warranty Term and Gratis Warranty Range

If any faults or defects (hereinafter "Failure") found to be the responsibility of Mitsubishi occurs during use of the product within the gratis warranty term, the product shall be repaired at no cost via the sales representative or Mitsubishi Service Company.
However, if repairs are required onsite at domestic or overseas location, expenses to send an engineer will be solely at the customer's discretion. Mitsubishi shall not be held responsible for any re-commissioning, maintenance, or testing on-site that involves replacement of the failed module.

[Gratis Warranty Term]
The gratis warranty term of the product shall be for one year after the date of purchase or delivery to a designated place. Note that after manufacture and shipment from Mitsubishi, the maximum distribution period shall be six (6) months, and the longest gratis warranty term after manufacturing shall be eighteen (18) months. The gratis warranty term of repair parts shall not exceed the gratis warranty term before repairs.

[Gratis Warranty Range]

(1) The range shall be limited to normal use within the usage state, usage methods and usage environment, etc., which follow the conditions and precautions, etc., given in the instruction manual, user's manual and caution labels on the product.

(2) Even within the gratis warranty term, repairs shall be charged for in the following cases.

1. Failure occurring from inappropriate storage or handling, carelessness or negligence by the user. Failure caused by the user's hardware or software design.
2. Failure caused by unapproved modifications, etc., to the product by the user.
3. When the Mitsubishi product is assembled into a user's device, Failure that could have been avoided if functions or structures, judged as necessary in the legal safety measures the user's device is subject to or as necessary by industry standards, had been provided.
4. Failure that could have been avoided if consumable parts (battery, backlight, fuse, etc.) designated in the instruction manual had been correctly serviced or replaced.
5. Failure caused by external irresistible forces such as fires or abnormal voltages, and Failure caused by force majeure such as earthquakes, lightning, wind and water damage.
6. Failure caused by reasons unpredictable by scientific technology standards at time of shipment from Mitsubishi.
7. Any other failure found not to be the responsibility of Mitsubishi or that admitted not to be so by the user.

2. Onerous repair term after discontinuation of production

(1) Mitsubishi shall accept onerous product repairs for seven (7) years after production of the product is discontinued. Discontinuation of production shall be notified with Mitsubishi Technical Bulletins, etc.

(2) Product supply (including repair parts) is not available after production is discontinued.

3. Overseas service

Overseas, repairs shall be accepted by Mitsubishi's local overseas FA Center. Note that the repair conditions at each FA Center may differ.

4. Exclusion of loss in opportunity and secondary loss from warranty liability

Regardless of the gratis warranty term, Mitsubishi shall not be liable for compensation to:

(1) Damages caused by any cause found not to be the responsibility of Mitsubishi.

(2) Loss in opportunity, lost profits incurred to the user by Failures of Mitsubishi products.

(3) Special damages and secondary damages whether foreseeable or not, compensation for accidents, and compensation for damages to products other than Mitsubishi products.

(4) Replacement by the user, maintenance of on-site equipment, start-up test run and other tasks.

5. Changes in product specifications

The specifications given in the catalogs, manuals or technical documents are subject to change without prior notice.

SH(NA)-080371E(1/2)-S(2110)KWIX
MODEL:SW1D5C-FBDQ-P-E
MODEL CODE: 13JW00

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : TOKYO BUILDING, 2-7-3 MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS : 1-14 , YADA-MINAMI 5-CHOME , HIGASHI-KU, NAGOYA , JAPAN

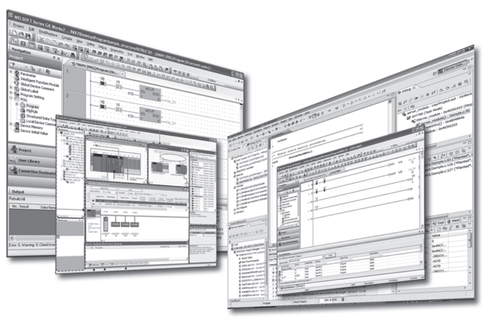
MELSOFT

MITSUBISHI ELECTRIC

Engineering Software

PX Developer Version 1 Programming Manual (2/2)

-SW1D5C-FBDQ-E
-SW1D5C-FBDQMON-E



• SAFETY PRECAUTIONS •

(Always read these instructions before using this product.)

Before using this product, thoroughly read this manual and the relevant manuals introduced in this manual and pay careful attention to safety and handle the products properly.
The precautions given in this manual are concerned with this product. For the safety precautions of the programmable controller system, refer to the User's Manual for the CPU module.
In this manual, the safety precautions are ranked as "⚠WARNING" and "⚠CAUTION".

⚠ WARNING	Indicates that incorrect handling may cause hazardous conditions, resulting in death or severe injury.
⚠ CAUTION	Indicates that incorrect handling may cause hazardous conditions, resulting in minor or moderate injury or property damage.

Note that the ⚠CAUTION level may lead to serious consequences according to the circumstances. Always follow the precautions of both levels because they are important for personal safety.

Please save this manual to make it accessible when required and always forward it to the end user.

[Security Precautions]

⚠WARNING

- To maintain the security (confidentiality, integrity, and availability) of the programmable controller and the system against unauthorized access, denial-of-service (DoS) attacks, computer viruses, and other cyberattacks from external devices via the network, take appropriate measures such as firewalls, virtual private networks (VPNs), and antivirus solutions.

[Startup/Maintenance Precautions]

⚠ CAUTION

- The online operations have to be executed after the manual has been carefully read and the safety has been ensured.
  Failure to do so may cause a miss operation which results in machine damage or an accident.

● CONDITIONS OF USE FOR THE PRODUCT ●

(1) MELSEC programmable controller ("the PRODUCT") shall be used in conditions;
i) where any problem, fault or failure occurring in the PRODUCT, if any, shall not lead to any major or serious accident; and
ii) where the backup and fail-safe function are systematically or automatically provided outside of the PRODUCT for the case of any problem, fault or failure occurring in the PRODUCT.

(2) The PRODUCT has been designed and manufactured for the purpose of being used in general industries.
MITSUBISHI ELECTRIC SHALL HAVE NO RESPONSIBILITY OR LIABILITY (INCLUDING, BUT NOT LIMITED TO ANY AND ALL RESPONSIBILITY OR LIABILITY BASED ON CONTRACT, WARRANTY, TORT, PRODUCT LIABILITY) FOR ANY INJURY OR DEATH TO PERSONS OR LOSS OR DAMAGE TO PROPERTY CAUSED BY the PRODUCT THAT ARE OPERATED OR USED IN APPLICATION NOT INTENDED OR EXCLUDED BY INSTRUCTIONS, PRECAUTIONS, OR WARNING CONTAINED IN MITSUBISHI ELECTRIC USER'S, INSTRUCTION AND/OR SAFETY MANUALS, TECHNICAL BULLETINS AND GUIDELINES FOR the PRODUCT. ("Prohibited Application")
Prohibited Applications include, but not limited to, the use of the PRODUCT in;
- Nuclear Power Plants and any other power plants operated by Power companies, and/or any other cases in which the public could be affected if any problem or fault occurs in the PRODUCT.
- Railway companies or Public service purposes, and/or any other cases in which establishment of a special quality assurance system is required by the Purchaser or End User.
- Aircraft or Aerospace, Medical applications, Train equipment, transport equipment such as Elevator and Escalator, Incineration and Fuel devices, Vehicles, Manned transportation, Equipment for Recreation and Amusement, and Safety devices, handling of Nuclear or Hazardous Materials or Chemicals, Mining and Drilling, and/or other applications where there is a significant risk of injury to the public or property.

Notwithstanding the above, restrictions Mitsubishi Electric may in its sole discretion, authorize use of the PRODUCT in one or more of the Prohibited Applications, provided that the usage of the PRODUCT is limited only for the specific applications agreed to by Mitsubishi Electric and provided further that no special quality assurance or fail-safe, redundant or other safety features which exceed the general specifications of the PRODUCTs are required. For details, please contact the Mitsubishi Electric representative in your region.

(3) Mitsubishi Electric shall have no responsibility or liability for any problems involving programmable controller trouble and system trouble caused by DoS attacks, unauthorized access, computer viruses, and other cyberattacks.

REVISIONS

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Dec., 2002	SH (NA)-080371E-A	First edition
Apr., 2003	SH (NA)-080371E-B	Correction Section 2.11.1, Section 7.6.20, Appendix 1.2
Oct., 2003	SH (NA)-080371E-C	Addition Appendix 5.1 Correction Section 2.2.4, Section 2.3.1, Section 2.10, Section 2.11.1, Section 5.1.1, Section 5.1.2, Section 7.9.1 to 7.9.4, Chapter 8, Section 8.1.3, Section 8.1.4, Section 8.2.1, Section 8.2.2, Section 8.2.3, Appendix 1.1, Appendix 5
Jun., 2004	SH (NA)-080371E-D	Model Addition Q12PRHCPU, Q25PRHCPU Addition Section 2.14 Correction Terms, Section 1.1, Section 1.3.1, Section 2.2.4, Section 2.3.1, Section 2.10 (Whole), Section 4.10, Section 5.5 (Whole), Section 7.4.7, Section 7.6, Chapter 8, Appendix 1, Appendix 2, Appendix 5
Jun., 2004	SH (NA)-080371E-E	Correction Section 7.6.7, Section 7.6.12
Feb., 2005	SH (NA)-080371E-F	Addition Section 4.9.5 Correction Chapter 7, Section 7.1.1, Section 7.1.2, Section 7.6.20, Appendix 1.1, Appendix 4, Appendix 5, Index

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Feb., 2006	SH (NA)-080371E-G	Addition Section 7.5.4, Section 7.6.7, Section 7.6.8, Section 7.6.11, Section 7.6.12, Section 7.8.9, Section 7.8.10, Section 7.8.13, Section 7.8.14, Appendix 3.14 Correction Section 1.2, Section 2.9.3, Section 2.9.4, Section 2.11.13, Section 2.14.2, Chapter 3, Section 4.1.16, Section 4.1.17, Section 4.1.19, Section 7.1.1, Section 7.1.2, Section 7.4.6, Section 7.5.1, Section 7.5.2, Section 7.5.6, Section 7.6.17, Section 7.6.18, Section 7.6.19, Section 7.7.1, Section 7.8.19, Section 7.8.20, Section 7.8.35, Appendix 1, Appendix 2, Appendix 3, Appendix 5 Section 7.5.4 to 7.5.7 changed to Section 7.5.5 to 7.5.8 Section 7.6.13 to 7.6.14 changed to Section 7.6.5 to 7.6.6 Section 7.6.5 to 7.6.6 changed to Section 7.6.9 to 7.6.10 Section 7.6.7 to 7.6.12 changed to Section 7.6.13 to 7.6.18 Section 7.6.15 to 7.6.23 changed to Section 7.6.19 to 7.6.27 Section 7.8.13 to 7.8.16 changed to Section 7.8.5 to 7.8.8 Section 7.8.5 to 7.8.6 changed to Section 7.8.11 to 7.8.12 Section 7.8.7 to 7.8.12 changed to Section 7.8.15 to 7.8.20 Section 7.8.17 to 7.8.31 changed to Section 7.8.21 to 7.8.35
Mar., 2007	SH (NA)-080371E-H	Addition Section 7.6.25, Section 7.8.36, Section 8.1.4, Section 8.1.6, Section 8.1.11, Appendix 3.15 Correction Section 2.9.3, Section 2.9.4, Section 2.10, Section 5.5.1, Section 5.5.2, Section 7.5.4, Section 7.6.7, Section 7.6.8, Section 7.6.25, Section 7.7.1, Section 7.8.9, Section 7.8.10, Section 7.8.31, Section 7.8.32, Section 7.8.33, Section 7.8.36, Chapter 8, Section 8.2.1, Section 8.2.2, Section 8.2.3, Section 8.4.3, Section 8.4.4, Appendix 1, Appendix 1.1, Appendix 1.2, Appendix 1.3, Appendix 3.3, Appendix 3.14, Appendix 5, INDEX Section 7.6.25 to 7.6.27 changed to Section 7.6.26 to 7.6.28 Section 8.1.4 changed to Section 8.1.5 Section 8.1.5 to 8.1.8 changed to Section 8.1.7 to 8.1.10
Jun., 2008	SH (NA)-080371E-I	Model Addition Q02PHCPU, Q06PHCPU Addition Section 7.1.8, Section 8.2.2, Section 8.2.5, Appendix 3.11 Correction MANUALS, GENERIC TERMS, ABBREVIATIONS, AND TERMS, Section 1.2, Section 2.4, Section 2.9.1, Section 2.10, Section 7.1.7, Section 7.5.1, Section 7.7.1, Chapter 8, Section 8.2.1 Section 8.4.2, Appendix 1.2 to 1.3, Appendix 3.10, Appendix 5 Section 8.2.2 to 8.2.3 changed to Section 8.2.3 to 8.2.4 Appendix 3.11 to 3.15 changed to Appendix 3.12 to 3.16

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Jan., 2009	SH (NA)-080371E-J	Addition Section 7.5.9, Section 7.8.37, Appendix 3.12 Correction Section 2.2.5, Section 2.9.3, Section 2.10, Section 2.14.1, Section 4.2.1, Section 7.5.1, Section 7.6.7 to 7.6.10, Section 7.8.1 to 7.8.28, Section 8.2.2, Appendix 1, Appendix 5 Appendix 3.12 to 3.16 changed to Appendix 3.13 to 3.17
Dec., 2009	SH (NA)-080371E-K	Addition CONDITIONS OF USE FOR THE PRODUCT, Section 7.1.9, Section 7.6.29 to 7.6.31, Section 7.8.38 to 7.8.40, Section 7.9.9, Section 8.1.12, Appendix 3.18 Correction SAFETY PRECAUTIONS, Section 2.9.3, Section 2.10, Section 7.5.4, Section 7.5.9, Section 7.6.7, Section 7.6.8, Section 7.8.9, Section 7.8.10, Section 8.1.10, Appendix 1 to 1.3, Appendix 5
Dec., 2010	SH (NA)-080371E-L	Addition Section 2.9 to 2.9.7, Section 4.2.7, Section 4.3.6, Section 8.2.29, Section 9.1.34, Appendix 3.19 Correction MANUALS, GENERIC TERMS, ABBREVIATIONS, AND TERMS, Section 2.1, Section 2.3.1, Section 2.10.3, Section 2.13, Section 2.15.3, Section 4.7.1, Section 4.10.5, Section 8.1.4, Section 8.2.26 to 8.2.28, Section 10.1.10, Appendix 1.1, Appendix 1.3, Appendix 3.14, Appendix 5 Section 2.9 to 2.14 changed to Section 2.10 to 2.15 Section 7.5 to 7.7.1 changed to Chapter 8 Section 7.8 to 7.11.1 changed to Chapter 9 Chapter 8 changed to Chapter 10
Oct., 2011	SH (NA)-080371E-M	Addition Section 7.4.11, Section 9.1.42, Section 9.1.43, Appendix 4 Correction Section 2.1, Section 2.2.1, Section 2.2.5, Section 2.10.3, Section 2.15.3, Section 8.1.1, Section 8.2.19, Section 8.3.1, Appendix 1 to 1.3, Appendix 6 Appendix 4 to 5.1 changed to Appendix 5 to 6.1

* The manual number is given on the bottom left of the back cover.

Print Date	* Manual Number	Revision
Feb., 2014	SH (NA)-080371E-N	Addition Section 4.3.7, Section 9.3.2, Section 9.4.2, Section 10.1.13 Correction GENERIC TERMS, ABBREVIATIONS, AND TERMS, Section 1.3.1, Section 2.2.4, Section 2.3.1, Section 2.7.4, Section 2.10.3, Section 2.11 to 2.11.2, Section 2.12.1, Section 2.15 to 2.15.4, Section 4.1.1 to 4.1.21, Section 4.2.1 to 4.2.7, Section 4.3.1 to 4.3.6, Section 4.4.1, Section 4.4.2, Section 4.5.1, Section 4.6.1 to 4.6.4, Section 4.7.1, Section 4.8.1 to 4.8.8, Section 4.9.1 to 4.9.5, Section 4.10.1, Section 5.4.1. Section 5.4.3, Section 5.4.5, Section 6.1.1 to 6.1.5, Section 8.2.7, Section 8.2.8, Section 8.2.30 to 8.2.32, Section 9.1.39 to 9.1.41, Chapter 10, Section 10.1.1 to 10.1.12, Section 10.2.1 to 10.2.5, Section 10.3.1, Section 10.3.2, Section 10.5.1 to 10.5.4, Appendix 1 to Appendix 1.3, Appendix 3.7, Appendix 3.12, Appendix 4, Appendix 6, Appendix 6.1
Jul., 2015	SH (NA)-080371E-O	Correction Section 2.6.1, Section 4.7.1, Section 7.4.6, Section 7.4.7, Section 8.2.25
Jan., 2017	SH (NA)-080371E-P	Correction Section 8.1.8, Section 8.2.1, Section 8.2.3, Section 8.2.5, Section 8.2.7, Section 8.2.9, Section 8.2.11, Section 8.2.13, Section 8.2.15, Section 8.2.17, Section 8.2.19, Section 8.2.20, Section 8.2.22, Section 8.2.27, Section 8.2.28, Section 8.2.29, Section 9.1.1, Section 9.1.3, Section 9.1.5, Section 9.1.7, Section 9.1.9, Section 9.1.11, Section 9.1.13, Section 9.1.15, Section 9.1.17, Section 9.1.19, Section 9.1.21, Section 9.1.23, Section 9.1.25, Section 9.1.32, Section 9.1.33, Section 9.1.34, Section 9.1.42, Section 9.1.43
Apr., 2019	SH (NA)-080371E-Q	Model Addition Q04UDPVCPU, Q06UDPVCPU, Q13UDPVCPU, Q26UDPVCPU Correction GENERIC TERMS, ABBREVIATIONS, AND TERMS, Section 2.2.5, Section 2.4, Section 2.15.3, Section 4.10.5
Apr., 2020	SH (NA)-080371E-R	Correction Section 2.11.1, Section 10.1.1 to 10.1.13, Section 10.2.1 to 10.2.5, Section 10.3.1, Section 10.3.2
Oct., 2021	SH (NA)-080371E-S	Correction SAFETY PRECAUTIONS, CONDITIONS OF USE FOR THE PRODUCT, Section 9.1.29, Section 9.1.30

Japanese Manual Version SH-080261- AO

INTRODUCTION

Thank you for purchasing the engineering software, MELSOFT series.
Read this manual and make sure you understand the functions and performance of MELSOFT series thoroughly in advance to ensure correct use.

CONTENTS

Programming Manual (2/2)

MANUALS

The following manuals are also related to this product.
Refer to the following table for ordering a manual.

Related manuals

Manual name	Manual number (model code)
PX Developer Version 1 Programming Manual Details of programming with PX Developer, lists of FB parts, and the PID instructions (this manual) (Sold separately.)	SH-080371E (13JW00)
PX Developer Version 1 Operating Manual (Programming Tool) FBD language programming, compilation, online operation and debug methods with PX Developer (Sold separately.)	SH-080369E (13JU38)
PX Developer Version 1 Operating Manual (Monitor tool) Operation methods of the monitor tool and methods for monitoring and controlling DDC processing with tag FB (Sold separately.)	SH-080370E (13JU39)
PX Developer Version 1 Operating Manual (GOT Screen Generator) Generation procedure for GOT screen project and details about generated screen (Sold separately.)	SH-080772ENG (13JU61)
PX Developer Version 1 Operating Manual (InTouch Interaction) Interaction between PX Developer monitor tool and SCADA software (InTouch) (Sold separately.)	SH-080773ENG (13JU62)
PX Developer Version 1 Operating Manual (JoyWatcherSuite Interaction) Interaction between PX Developer monitor tool and SCADA software (JoyWatcherSuite) (Sold separately.)	SH-080976ENG (13JU70)

CAUTION

- Please note that we do not guarantee commercially available software compatible with Microsoft® Windows® Operating System introduced in this manual.
- The software copyright of this product belongs to Mitsubishi Electric Corporation.
- No contents in this manual can be reproduced or duplicated in any form or by any means without permission.
- Although we make utmost efforts, this manual may not completely follow the revisions of the software and hardware.
- In principle, this software should be purchased by one set per personal computer or by license purchase.
- This product (including this manual) can only be used under the software license agreement.
- Please note that we are not responsible for any influence resulting from operating this product (including this manual).
- The contents of this manual are subject to change without notice.

HOW TO USE THIS MANUAL

"HOW TO USE THIS MANUAL" is arranged according to different needs in using: Please refer to the following contents when using this manual:

(1) Hoping to learn features, product configuration and project flow (☞ Chapter 1)
Features are described in Section 1.1; product configuration is illustrated in Section 1.2; and the project flow in Section 1.3.

(2) Hoping to learn the programming method of FBD language (☞ Chapter 2)
FBD language and its programming method are described in Chapter 2.

(3) Programming with FBD parts (☞ Chapter 3 to Chapter 10, Appendix 1)
- Reading method of instructions after Chapter 4 is described in Chapter 3.
- Input/output pins parameter, function and program example of general functions are described in Chapter 4.
- Input/output pins parameter, function and program example of general FB are described in Chapter 5.
- Input/output pins parameter, function and program example of process function are described in Chapter 6.
- Input/output pins parameter, public variable, function, and program example of process FB are described in Chapter 7, Chapter 8, and Chapter 9.
- Input/output pins, public variable, function, and program example of module FB are described in Chapter 10.
- The tag data list and its detailed information are in Appendix 1.

(4) Hoping to learn the contents of error codes for process control (☞ Appendix 2)
The check method and contents of error codes for process control are elaborated in Appendix 2.

(5) Hoping to learn process-related functions (☞ Appendix 3, Appendix 5)
- Process-related functions are elaborated in Appendix 3.
- Relative terms are elaborated in Appendix 5.

GENERIC TERMS, ABBREVIATIONS, AND TERMS

The following table shows the generic terms, abbreviations, and terms in this manual.

(1) Generic terms and abbreviations

Generic term/abbreviation	Description
PX Developer	Generic term for PX Developer Version 1 (SW1D5C-FBDQ-E) and PX Developer Monitor Tool (SW1DNC-FBDQMON-E) For PX Developer, Programming Tool and Monitor Tool are installed. For PX Developer Monitor Tool, only Monitor Tool is installed.
GX Works2	Abbreviation for GX Works2 Version 1 (SW1DNC-GXW2-E Version 1.98C) or later
GX Developer	Abbreviation for GX Developer Version 7 (SW7D5C-GPPW-E Version 7.20W) or later
GX Simulator	Abbreviation for GX Simulator Version 7 (SW7D5C-LLT-E Version 7.27D) or later
GX application	Generic term for GX Works2 and GX Developer which are interacted with PX Developer
GX project	Generic term for GX Works2 project and GX Developer project included in PX Developer project
FBD program	Generic term for a program created in FBD language
FBD part	Generic term for parts (FB part, function part, variable part, constant part, comment part, etc.) used by the programming tool
Global part	Generic term for module FB, tag FB, and global variable
Peripheral device	Generic term for the personal computer on which PX Developer can be used
QCPU	Generic term for Q00JCPU, Q00UJCPU, Q00CPU, Q00UCPU, Q01CPU, Q01UCPU, Q02CPU, Q02HCPU, Q02PHCPU, Q02UCPU, Q03UDCPU, Q03UDECPU, Q03UDVCPU, Q04UDHCPU, Q04UDEHCPU, Q04UDVCPU, Q04UDPVCPU, Q06HCPU, Q06PHCPU, Q06UDHCPU, Q06UDEHCPU, Q06UDVCPU, Q06UDPVCPU, Q10UDHCPU, Q10UDEHCPU, Q12HCPU, Q12PHCPU, Q12PRHCPU, Q13UDHCPU, Q13UDEHCPU, Q13UDVCPU, Q13UDPVCPU, Q20UDHCPU, Q20UDEHCPU, Q25HCPU, Q25PHCPU, Q25PRHCPU, Q26UDHCPU, Q26UDEHCPU, Q26UDVCPU, Q26UDPVCPU, Q50UDEHCPU, and Q100UDEHCPU
Process CPU	Generic term for Q02PHCPU, Q06PHCPU, Q12PHCPU, and Q25PHCPU
Universal model process CPU	Generic term for Q04UDPVCPU, Q06UDPVCPU, Q13UDPVCPU, and Q26UDPVCPU
Redundant CPU	Generic term for Q12PRHCPU and Q25PRHCPU
CPU module	Generic term for the Process CPU, Universal model process CPU, and Redundant CPU
ACPU	Generic term for the PLC CPU that can be used with MELSEC-A series
Redundant type extension base unit	Abbreviation for Q65WRB extension base unit for redundant system
CC-Link IE Controller Network	Abbreviation for CC-Link IE Controller Network system compatible with the Q series
MELSECNET/H	Abbreviation for MELSECNET/H network system compatible with the Q series
MELSECNET/10	Abbreviation for MELSECNET/10 network system compatible with the AnU, QnA/Q4AR
MELSECNET/10 compatible mode	Abbreviation for function and performance-compatible mode so that the MELSECNET/H network system can have upward compatibility to existing MELSECNET/10 network system
CC-Link IE Controller Network board	Abbreviation for CC-Link IE Controller Network interface board
MELSECNET/H board	Abbreviation for MELSECNET/H interface board
MELSECNET/10 board	Abbreviation for MELSECNET/10 interface board
Ethernet board	Generic term for Ethernet PC card and Ethernet interface board supported by Windows®
Personal computer	Generic term for IBM-PC/AT-compatible personal computer
Programming tool	Abbreviation for PX Developer programming tool
Monitor tool	Abbreviation for PX Developer monitor tool

(2) Terms

Term	Description
DDC	Abbreviation for Direct Digital Control A control of controller functions with a digital device.
FB	Abbreviation for Function Block A block with a specific function used in a program.
FBD	Abbreviation for Function Block Diagram defined in IEC61131-3 Programs are created by connecting variables, constants, and blocks containing specific processing, according to the flow of data signal.
ST	Abbreviation for Structured Text defined in IEC61131-3 Programs are created by writing arithmetic operations and logical operations in text format.
Project	Unit that gathers and manages a series of data necessary for configuration of FBD programs executed by the CPU module
Tag	Identification symbol attached to each DDC processing defined by JIS This can be likened to a tag attached to process control equipment.
Sequence control	Control that processes each control step according to preset order and procedures
Loop control	Control method that repeatedly executes processing of specific parts
Member	Basic data items in structure type data
Tag data	Data that data attached to DDC processing indicated with a tag (process condition data/process status data) is summarized Accessing the tag data can monitor status and set conditions of the relevant DDC.
Tag FB	Function block works as a controller or an indicator containing tag data
Module FB	Function block for inputting/outputting data of analog I/O module, digital I/O module, and high-speed counter module connected to the base unit on which the PLC is mounted or CC-Link field bus
Faceplate	Gauge window on which an indicator such as a controller is displayed in image format. Values assigned to tag data are manipulated.
System resource	PLC device required for executing FBD programs, used for automatically assigning variables (This cannot be used in ladder programs.)
Ladder program	Program method designed so that contact sequence can be applied to PLC language Draw two vertical control bus lines and describe a contact between the buses for programming.
Identifier	Used for setting various element names (variable name, FB variable name, structure name, etc.) Some unusable characters cannot be used for the identifier.
Reserved word	Part names (such as VAR) that cannot be used as various element names (variable name, FB variable name, structure name, etc.)
Operation mode	Mode for determining the operation method of the redundant system The following three modes are available. • Backup mode • Separate mode • Debug mode
Backup mode	Mode for normal operation of the redundant system If a failure or an error occurs in the control system, the standby system switches to the control system to continue the control of the redundant system. The operation mode can be switched to the separate mode using GX application.
Separate mode	Mode for maintaining a system (partial modification of a program, replacement of modules mounted on the main base unit) without stopping the control during run of the redundant system During this mode, different programs can be executed in the control system and standby system. System switching cannot be made in this mode (User switching is possible). The operation mode can be switched to the backup mode using GX application.
Debug mode	Mode for performing a debug using a single system prior to redundant system operation This permits operations without connecting tracking cables. In this mode, the CPU module is fixed to system A, control system. (Tracking of the redundant system is not performed.) Set/cancel this mode in the redundant parameter setting of GX application.
Operation mode change	Switching of the operation mode for system A and system B using GX application while the redundant system is running The operation mode can be switched between the backup mode and separate mode.
System A	System to which system A connector for tracking cable is connected in the redundant system
System B	System to which system B connector for tracking cable is connected in the redundant system

Term	Description
System switching System switching User switching	Control switching to backup system to continue system control and network communication when a trouble occurs in the system that performs control in the redundant system (when a failure or an error occurs in the power supply system, mounted module, or network) (Switching between control system and standby system to avoid system down) The following two types are available. • System switching Automatic system switching by the redundant system when a trouble occurs • User switching System switching by sequence program/GX application
Control system	A system that performs program operation, system control, and network communication in the redundant system When system A and system B start concurrently in the backup mode, the system A will be the control system (Concurrent startup: One system starts within three seconds after the other system has started.) When the system A and system B start separately, a system that starts first will be the control system.
Standby system	Backup system to continue system control in case of a failure or an error in the module in the control system in the redundant system (The CPU module in the standby system does not calculate programs.) When system A and system B start concurrently in the backup mode, the system B will be the standby system. (Concurrent startup: One system starts within three seconds after the other system has started.) When the system A and system B start separately, a system that starts later will be the standby system.
Tracking transfer function	Data transfer function that keeps the data of control system and standby system consistent This function enables the standby system to serve as the control system to continue the system control in case of system down of the control system. The Redundant CPU can perform tracking transfer without making the tracking settings, as it tracking transfer setting data has been set by default. (Change tracking transfer setting data using GX application.)
Redundant system	System configured using Redundant CPUs This system consists of two basic systems including CPU modules, power supply modules, and network modules. (If module error occurs in one system, the other system continues the system control. Thus, system reliability is improved.) To configure the redundant system, prepare two sets of the systems where the above modules of the same models are mounted on the base unit, and connect the CPU modules with tracking cables.
Redundant parameter	Parameter for setting operation mode of Redundant CPU system and tracking transfer setting data (tracking setting) Use GX application to set the parameter.

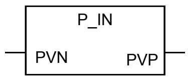
8 PROCESS FB_TAG ACCESS FB

Tag access FB is the instructions used for process control. It can be classified into following types.

Classification Name		Description	Reference
Tag access FB	I/O control FB	Execute analog input, output, and pulse integration, batch counter, etc.	Section 8.1
	Loop control operation FB	Ratio control, various PID control, 2 position ON/OFF, 3 position ON/OFF, program setter and loop selector, etc.	Section 8.2
	Special FB	Change control mode.	Section 8.3

8.1 Tag Access FB_I/O Control Operation FB

8.1.1 Analog Input Processing (P_IN)

FB	FBD parts
P_IN	

Corresponding tag type
PID, 2PID, 2PIDH, PIDP, SPI, IPD, BPI, R, ONF2, ONF3, PFC_SF, PFC_SS, PFC_INT, MONI, SWM, MWM, PVAL

Control mode				
MAN	AUT	CAS*1	CMV	CSV
○	○	○	○	○

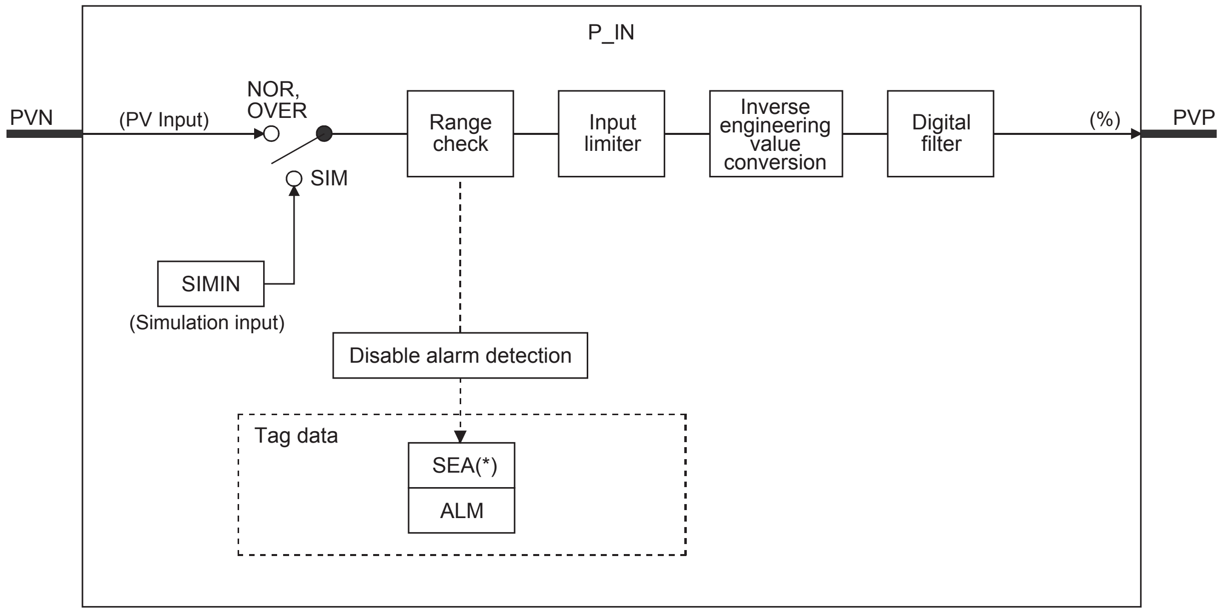
*1 Transition to CASDR is possible.

Functions overview: Read the converted digital value on analog module FB, carry out processing as range check, input limiter, inverse engineering value conversion and digital filter.
The input limiter processing can be enabled or disabled by using the project parameter.*2

Function/FB classification name: Tag access FB_I/O control FB

*2 Requires Process CPU or Redundant CPU whose upper five digits of serial No. are 10042 or later.

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	PV input from module FB	NMIN to NMAX
Output	PVP	Output variable	REAL	PV output (unit: %)	0 to 100

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User

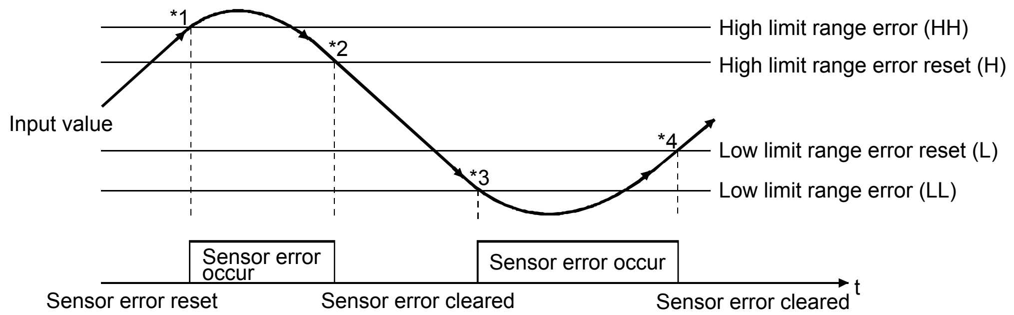
*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

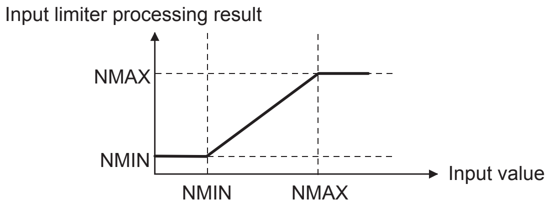
Function

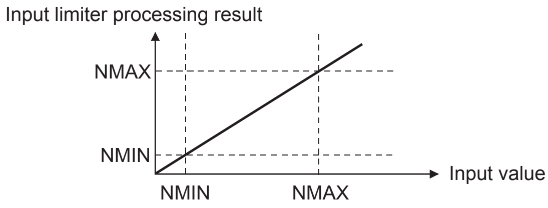
Item	Contents
Range check	Execute range check processing to the input value.

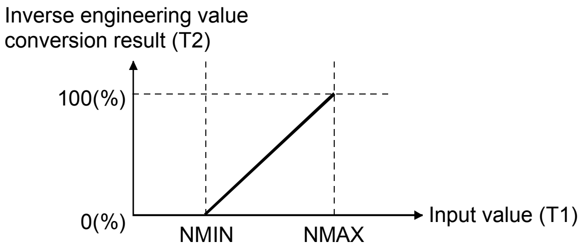


Condition	Alarm (ALM) / Sensor error (SEA)
Input value ≧ HH (above *1)	TRUE (occur)
Input value ≦ H (above *2)	FALSE (reset)
Input value ≦ LL (above *3)	TRUE (occur)
Input value ≧ L (above *4)	FALSE (reset)

HH: High limit range error, H: High Limit range error reset,
LL: Low limit range error, L: low limit range error reset.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input limiter</td><td>Execute input limiter processing to the input value


Operation when input limiter processing is enabled.


Operation when input limiter processing is disabled.
<table>
<tr><th>Condition</th><th>Input limiter processing result</th></tr>
<tr><td>Input value ≧ NMAX</td><td>NMAX</td></tr>
<tr><td>Input value ≦ NMIN</td><td>NMIN</td></tr>
<tr><td>NMIN < Input value < NMAX</td><td>Input value</td></tr>
</table>
NMAX: Input high limit, NMIN: Input low limit</td></tr>
<tr><td>Inverse engineering value conversion</td><td>Convert the engineering value input from A/D conversion module and output the result (%).


$$T2\ (\%) = \frac{T1-NMIN}{NMAX-NMIN} \times 100(\%)$$
T1: Input value, T2: Inverse engineering value conversion processing result (%)
NMAX: Input high limit, NMIN: Input low limit</td></tr>
<tr><td>Digital filter</td><td>Execute digital filter processing.
Digital filter processing result = T2 + α (Previous digital filter processing value -T2)
α: Filter coefficient, T2: Inverse engineering value conversion processing result.</td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether enable alarm detection or not in range check:
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" of tag data:
If the following items of "Disable Alarm Detection" (INH) of tag data are TRUE, the SEA of alarm (ALM) will not be detected.
● ERRI, SEI
(2) "Disable alarm detection" by loop stop processing:
Please refer to loop stop processing in the following contents</td></tr>
</table>

Other Functions

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Holding processing</td><td>Set whether to hold output of P_IN when sensor error (SEA) occurs due to the high/low limit range error during the range check. The setting can be made through PX Developer project parameter setting.
[Setting procedure] [Project Parameter] → [I/O Control] → Holding processing
● "Hold the output of P_IN" selected: Hold output
● "Hold the output of P_IN" unselected: Continue operation</td></tr>
<tr><td>Loop stop processing</td><td>The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE.
1) Hold the output (PVP).
2) Change the control mode automatically to MANUAL.
3) Reset SEA when the SEA of alarm (ALM) occurs or reset SPA when TSTP is TRUE.
4) Alarm is not detected in range check.</td></tr>
</table>

Processing Operation

Processing / Control mode	Range check	Input limiter	Inverse engineering value conversion	Digital filter	Alarm
MAN, CMV, AUT, CAS, CSV, CASDR	○	○	○	○	○ (*1)

○ : Execute ×: Not execute

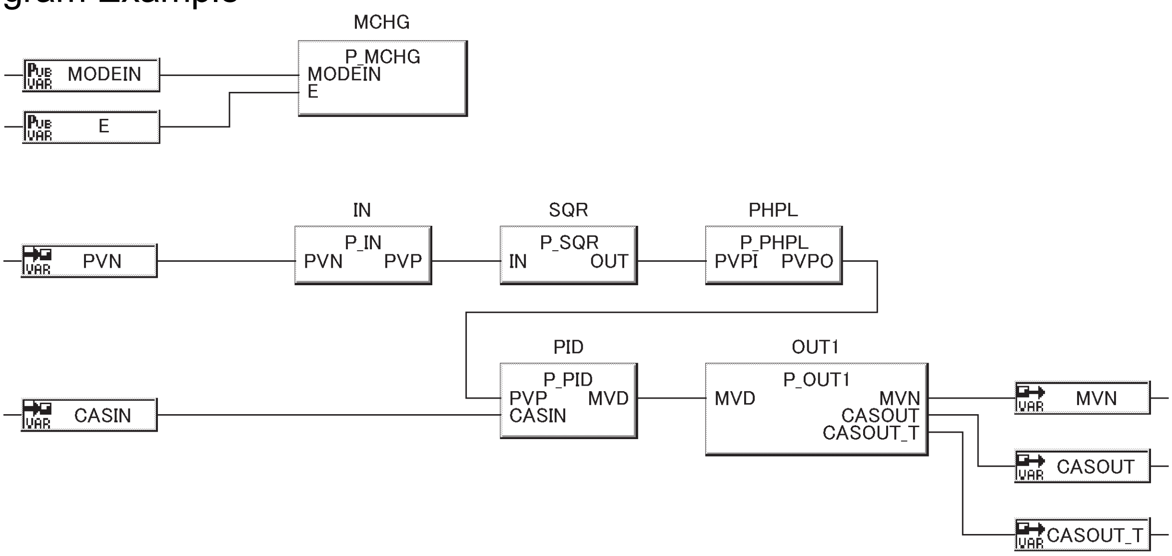
*1 When the bit of "Disable Alarm Detection" (INH) which corresponds to an alarm in TRUE, the alarm is not detected.

Error

Error may occur in the following case, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

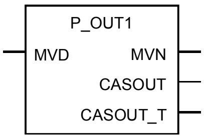
Program Example



POINT

- It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.
- Initial values of range error/range error reset are based on the default input range of an analog input module. The value is a digital value converted to percentage. When changing the input range, or treating an I/O value of an analog module FB as a digital value, change the value as required. For the setting examples of converting digital values to percentage, refer to Appendix 3.12.

8.1.2 Output Processing-1 with Mode Switching (With Input Addition) (P_OUT1)

FB	FBD parts
P_OUT1	

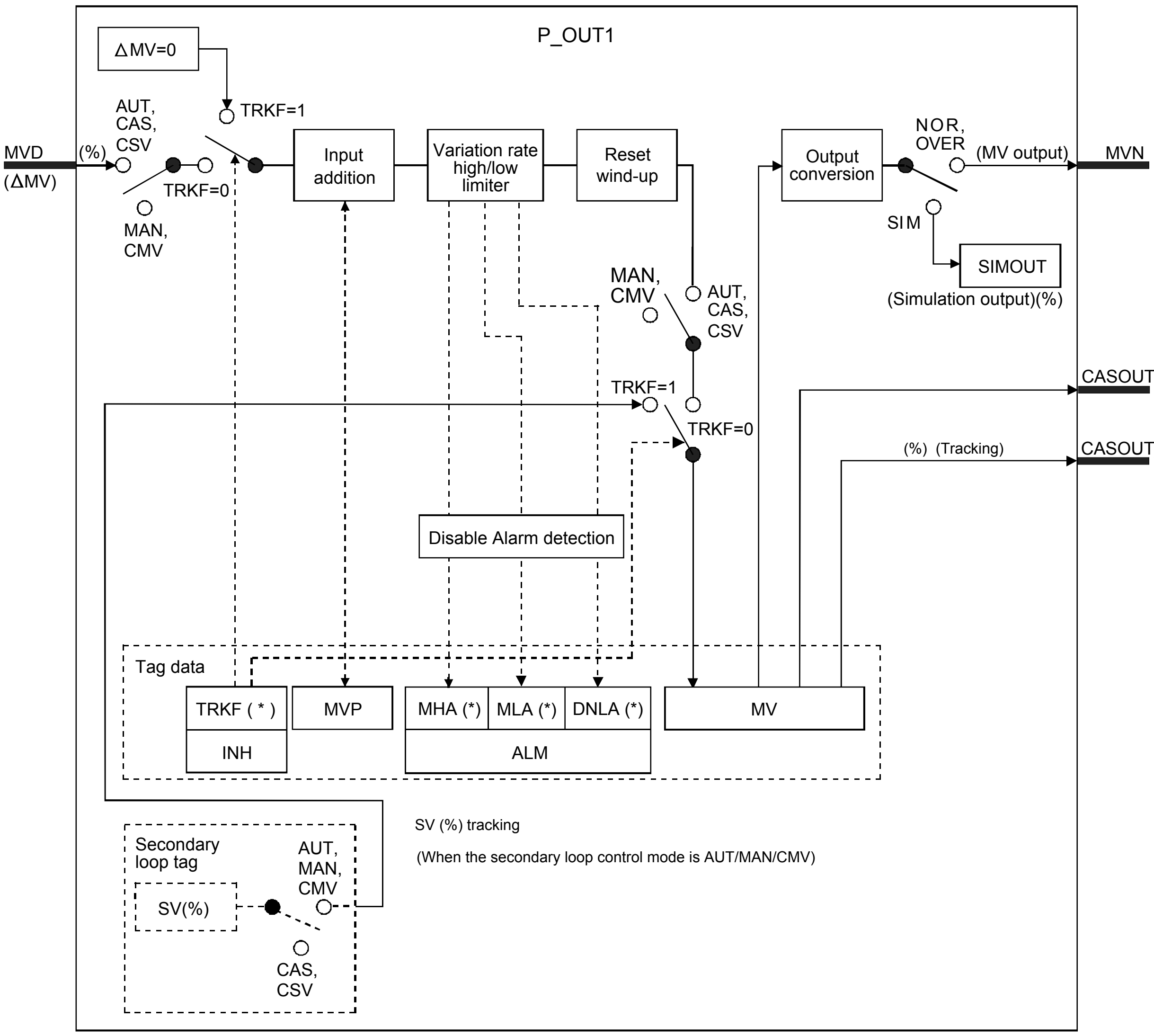
Corresponding tag type
BPI,IPD,PID,SPI,2PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Execute processing to the input value (ΔMV) as input addition, variation rate limiter and high/low limiter, reset windup, and output conversion and then output the MV.

Function/FB classification name: Tag access FB_I/O control FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	MVD	Input variable	REAL	ΔMV input (unit: %)	-999999 to 999999
Output	MVN	Output variable	REAL	MV output to module FB	NMIN to NMAX
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade output (unit: %) (With tracking)	0 to 100

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

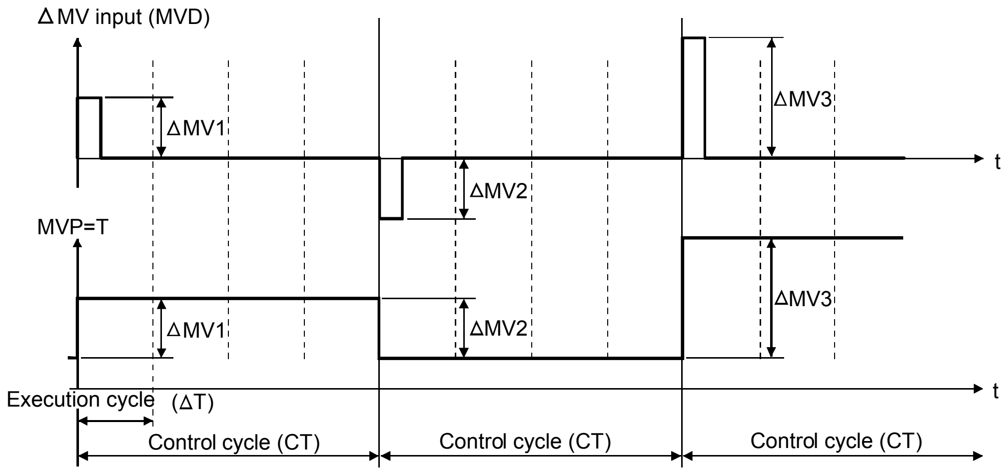
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.

Tag Data

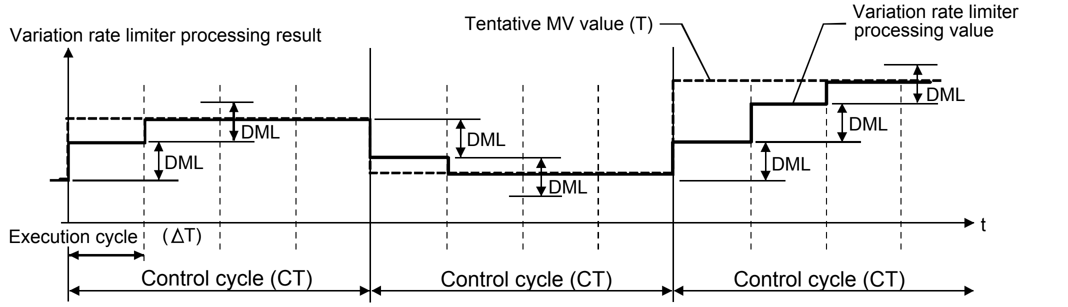
For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

Item	Contents
Input addition	Tentative MV value is calculated based on the input values (△MV) (△ MV is output in every control cycle (CT) by FB parts (P_PID and etc.) connected to the input variable)  T=△MV+MVP MVP=T △MV: Input value to MVD, MVP: MV internal operation value, T: Tentative MV value ● When the MAN, CMV mode are changed to AUT, CSV, manipulated variable (MV) is stored in MV internal operation value (MVP) to avoid sudden MV change. ● While tacking, △MV becomes 0 and manipulated variable is stored into the MV internal operation value (MVP).

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Variation rate limiter and high/low limiter</td><td>
Variation rate limiter and high/low limit of the input value are checked.

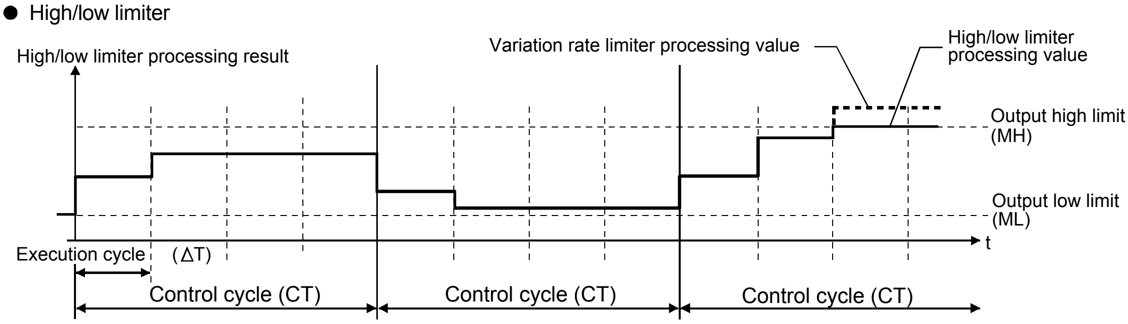
● Variation rate limiter



<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">Variation rate limiter processing result</th><th>Alarm (ALM)</th></tr>
<tr><th>Output variation rate limit (DMLA)</th></tr>
<tr><td>|T-MV|≦DML</td><td>T</td><td>FALSE (reset)</td></tr>
<tr><td>T-MV>DML</td><td>MV+DML</td><td>TRUE (occur)</td></tr>
<tr><td>T-MV< -DML</td><td>MV-DML</td><td>TRUE (occur)</td></tr>
</table>

T: Tentative MV value, MV: Manipulated variable, DML: Output change high limit

● High/low limiter

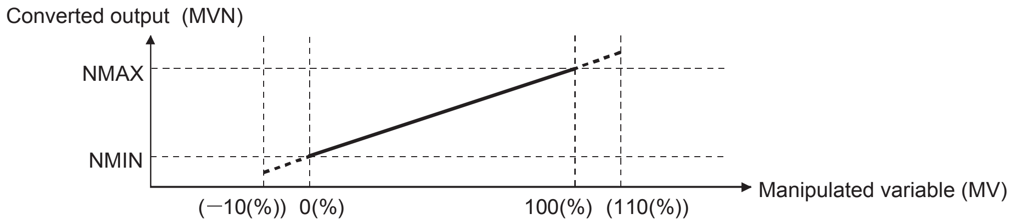


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter processing result</th><th colspan="2">Alarm (ALM)</th></tr>
<tr><th>Output low limit (MLA)</th><th>Output high limit MHA)</th></tr>
<tr><td>Variation rate limiter processing result >MH</td><td>MH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter processing result <ML</td><td>ML</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>ML≦ Variation rate limiter processing result ≦MH</td><td>Variation rate limiter processing value</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>

MH: Output high limit, ML: Output low limit
</td></tr>
<tr><td>Reset windup</td><td>
Countermeasures against reset windup include: Return MV to high/low limit when MV exceeds the high/low limit; Respond quickly when deviation is inverted.

<table>
<tr><th>Condition</th><th>Countermeasure processing against reset windup</th></tr>
<tr><td>Variation rate limiter processing result > MH and $\frac{\Delta T}{Ti} \leqq 1$</td><td>$MVP= \frac{\Delta T}{Ti} \times (MH-T)+T$</td></tr>
<tr><td>Variation rate limiter processing result < ML and $\frac{\Delta T}{Ti} \leqq 1$</td><td>$MVP= \frac{\Delta T}{Ti} \times (ML-T)+T$</td></tr>
</table>

MH: Output high limit value, ML: Output low limit value, MVP: MV internal operation value, △T: Execution cycle,
T: Tentative MV value, Ti: Integral time
If Ti=0, no countermeasures against reset windup will be performed.
</td></tr>
</table>

Item	Contents
Output conversion	Output conversion processing is carried out.  $\boxed{\text{Converted output (MVN)}=\{(NMAX-NMIN)\times\frac{MV}{100}+NMIN}$ NMAX: Output conversion high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%), MVN: Converted output
Disable Alarm Detection	Set whether enable Alarm Detection or not in variation rate limiter and high/low limiter. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected. ● ERRI, DMLI, MHI, MLI (2) Disable Alarm Detection by control mode: If the control mode is MAN and CMV, reset DMLA, MHA and MLA of alarm (ALM) and DMLA, MHA and MLA will not be detected. (3) Disable Alarm Detection by stop processing: Please refer to loop stop processing in the following contents.

Other Functions

Item	Contents
Holding processing	Set whether to hold output of P_OUT1 when sensor error (SEA) occurs in P_IN of tag access FB. Hold processing is to execute PX Developer project parameter setting. The setting can be made through PX Developer project parameter setting. [Setting procedure] [Project Parameter] → [Program Execution] → $\boxed{\text{Holding processing}}$ ● "Hold the output of P_OUT1, P_OUT2, P_DUTY" selected: Hold output ● "Hold the output of P_OUT1, P_OUT2, P_DUTY" unselected: Continue operation
Loop stop processing	The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occur. 4) Alarm is not detected in variation rate limiter and high/low limiter.

Processing Operation

Processing / Control mode	Input addition	Variation rate limiter and high/low limiter	Reset windup	Output conversion	Alarm
MAN, CMV,	×	×	×	○	× (*1)
AUT, CAS, CSV	○	○	○	○	○ (*2)

○: Execute ×: Not execute

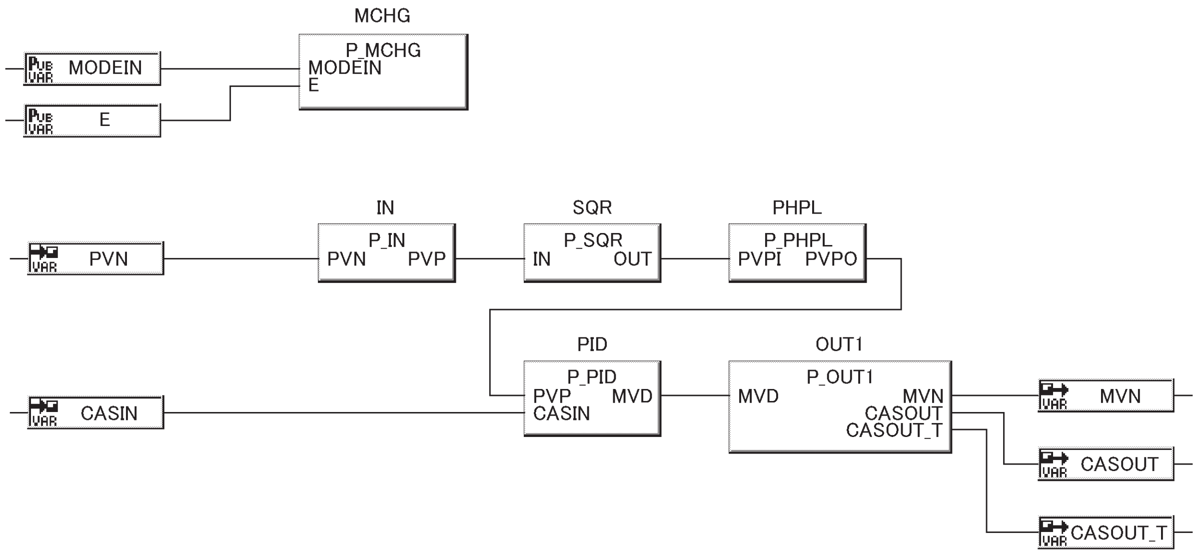
*1 An alarm (ALM) whose corresponding bit is TRUE (occurred) is reset, and the alarm cannot be detected.
*2 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

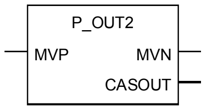
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.1.3 Output Processing-2 with Mode Switching (Without Input Addition) (P_OUT2)

FB	FBD parts
P_OUT2	

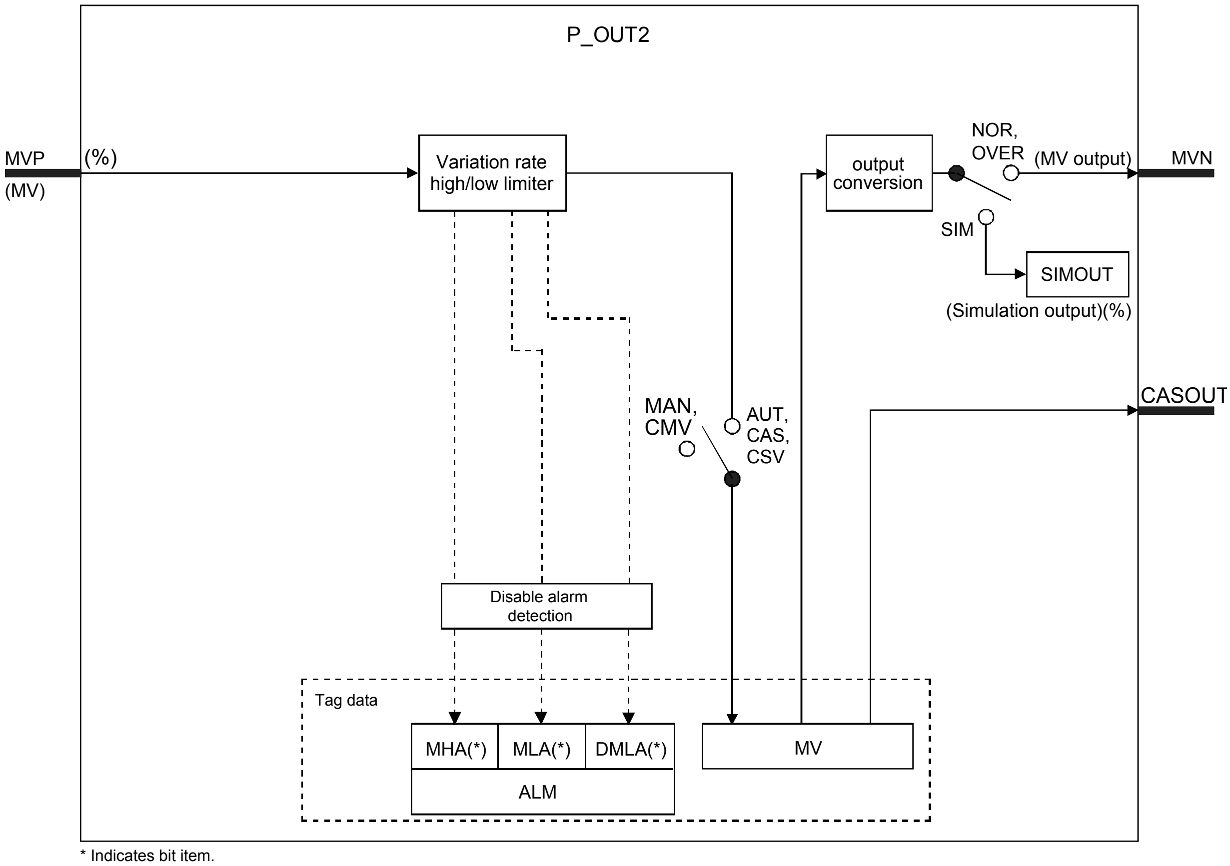
Corresponding tag type
R

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Carry out processing as variation rate & high/low limiter, output conversion to the input (MV); and then output MV.

Function/FB classification name: Tag access FB_I/O control FB

Block Diagram



Input and output pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	MVP	Input variable	REAL	MV input (unit: %)	0 to 100
Output	MVN	Output variable	REAL	MV output to module FB	NMIN to NMAX
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100

Public Variable (operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.

Tag Data

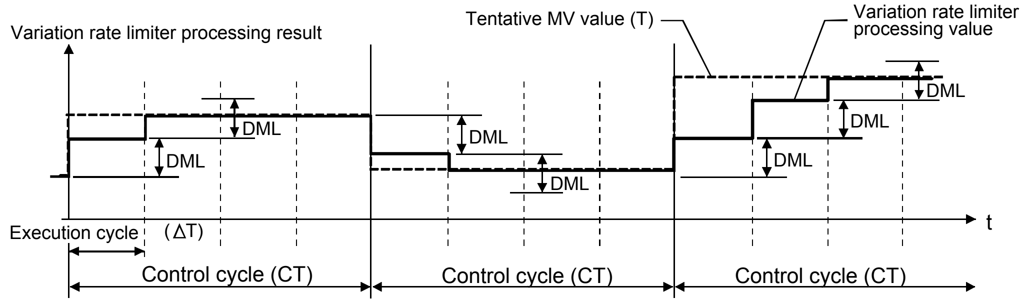
For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

Item	Contents
Variation rate limiter and high/low limiter	Variation rate limiter and high/low limit of the input value are checked.

Variation rate limiter and high/low limit of the input value are checked.

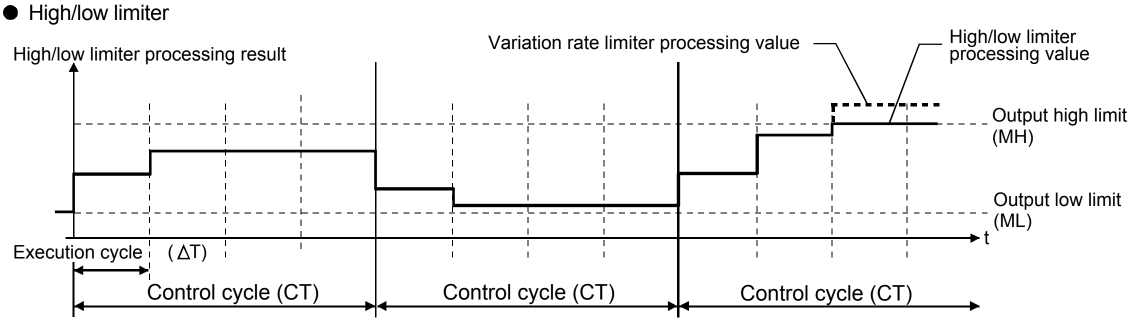
- Variation rate limiter.



Condition	Variation rate limiter processing result	Alarm (ALM) Output variation rate limit (DMLA)
|T-MV|≦DML	T	FALSE (reset)
T-MV>DML	MV+DML	TRUE (occur)
T-MV<-DML	MV-DML	TRUE (occur)

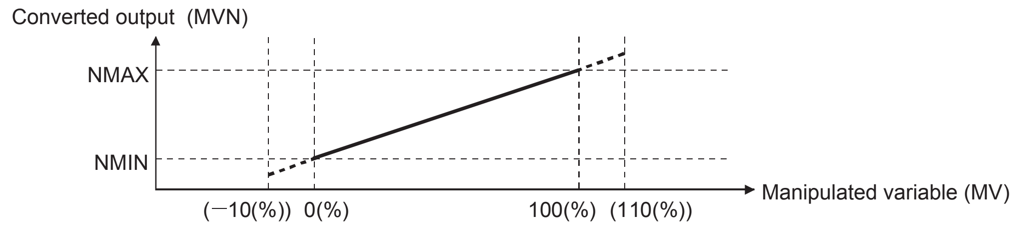
T: Tentative MV value, MV: Manipulated variable, DML: Output variation rate high limit

- High/low limiter



Condition	High/low limiter processing result	Alarm (ALM)	
		Output low limit (MLA)	Output high limit (MHA)
Variation rate limiter processing result >MH	MH	FALSE (reset)	TRUE (occur)
Variation rate limiter processing result <ML	ML	TRUE (occur)	FALSE (reset)
ML≦ Variation rate limiter processing result ≦ MH	Variation rate limiter processing value	FALSE (reset)	FALSE (reset)

MH: Output high limit value, ML: Output low limit value

Item	Contents
Output conversion	Execute output conversion processing.  Converted output (MVN)={(NMAX-NMIN) $\times \frac{MV}{100}$}+NMIN NMAX: Output Conversion high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%), MVN: Converted output
Disable Alarm Detection	Set whether enable Alarm Detection or not in variation rate limiter and high/low limiter. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected. ● ERRI, DMLI, MHI, MLI (2) Disable Alarm Detection by control mode: If the control mode is MAN and CMV, reset DMLA, MHA and MLA of alarm (ALM) and DMLA, MHA and MLA will be not detected. (3) Disable Alarm Detection by stop processing: Please refer to loop stop processing in the following contents.

Other Functions

Item	Contents
Holding processing	Set whether to hold output of P_OUT2 when sensor error (SEA) occurs on tag access P_IN. The setting can be made through PX Developer project parameter setting. [Setting procedure][Project Parameter] → [Program Execution] → Holding processing ● "Hold the output of P_OUT1, P_OUT2, P_DUTY" selected: Hold output ● "Hold the output of P_OUT1, P_OUT2, P_DUTY" unselected': Continue operation
Loop stop processing	The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occur. 4) Alarm is not detected in variation rate limiter and high/low limiter.

Processing Operation

Processing / Control mode	Variation rate limiter and high/low limiter	Output conversion	Alarm
MAN, CMV	×	○	× (*1)
AUT, CAS, CSV	○	○	○ (*2)

○: Execute ×: Not execute

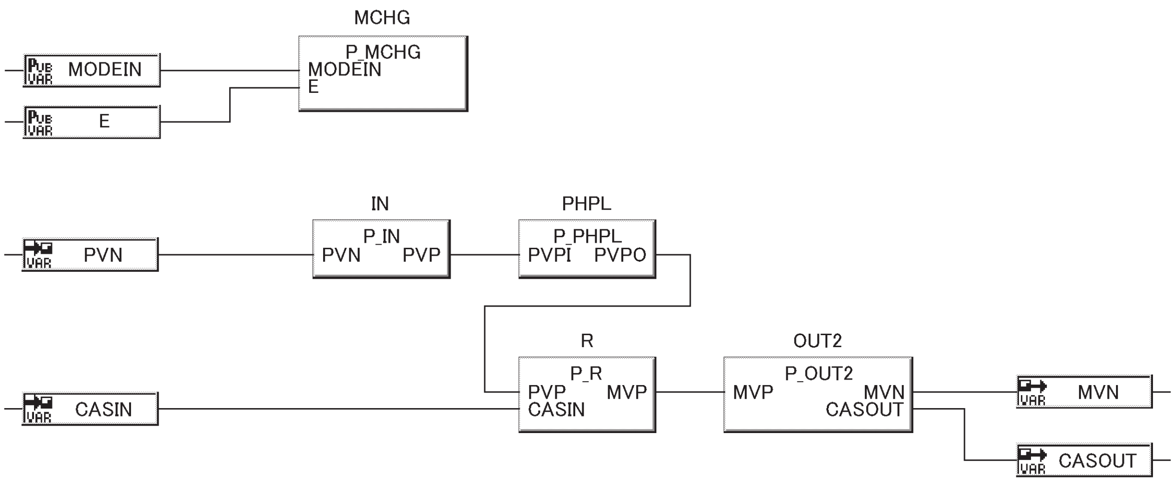
*1 An alarm (ALM) whose corresponding bit is TRUE (occurred) is reset, and the alarm cannot be detected.
*2 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

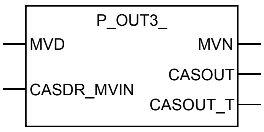
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



> **POINT**
>
> It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.1.4 Output Processing-3 with Mode Switching (With Input Addition and Compensation) (P_OUT3_)

FB	FBD parts
P_OUT3_	

Corresponding tag type
2PIDH

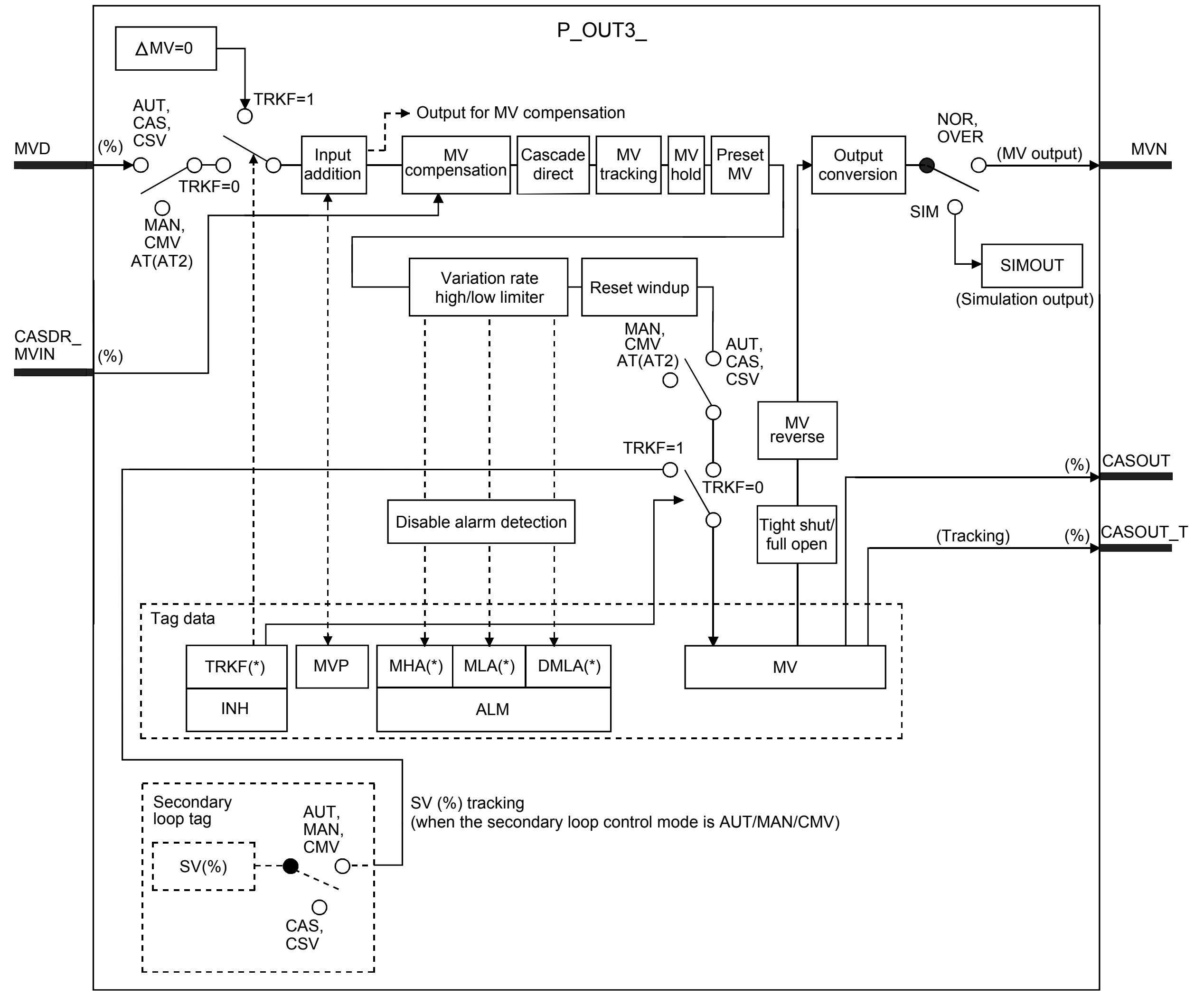
Control mode				
MAN	AUT	CAS*1	CMV	CSV
○	○	○	○	○

*1 Transition to CASDR is possible.

Functions overview: Executes processing to the input value (△MV) as input addition, MV compensation, preset MV, MV hold, MV tracking, variation rate limiter and high/low limiter, reset windup, tight shut/full open, MV reverse, and output conversion and then output the MV.

Function/FB classification name: Tag access FB_I/O control FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	MVD	Input variable	REAL	△MV input (unit: %)	-999999 to 999999
	CASDR_MVIN	Input variable	REAL	MV input for cascade direct (unit: %)	0 to 100
Output	MVN	Output variable	REAL	MV output to module FB	NMIN to NMAX
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade output (unit: %) (With tracking)	0 to 100

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	MVCMP_EN	Public variable	BOOL	MV compensation execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	MV_CMPIN	Public variable	REAL	MV compensation value (Unit: %)	-999999 to 999999	0.0	User
	MVCMP_MODE	Public variable	INT	MV compensation mode (0: Addition, 1: Replacement)	0 to 1	0	User
	PREMV_EN	Public variable	BOOL	Preset MV execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PREMV_V	Public variable	REAL	Preset MV value (Unit: %)	0 to 100	0.0	User
	MVHLD_EN	Public variable	BOOL	MV hold execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	MVTRK_EN	Public variable	BOOL	MV tracking execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	MV_TRKIN	Public variable	REAL	MV tracking input (Unit: %)	0 to 100	0	User
	STP_OTYPE	Public variable	INT	Output in loop stop or tag stop (0: Hold, 1: Preset value)	0 to 1	0	User
	SEA_OTYPE	Public variable	INT	MV output selection when SEA occurs (0: Hold, 1: Preset MV output, 2: Neither hold nor preset MV output is executed.)	0 to 2	0	User
	ARW_EX_EN	Public variable	BOOL	MV value instantaneous pullback when MV internal operation high/low limit value is over (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User
	MVPH	Public variable	REAL	MV internal operation high limit value (Unit: %)	MH to 999999	100.0	User
	MVPL	Public variable	REAL	MV internal operation low limit value (Unit: %)	-999999 to ML	0.0	User
	MVREV_EN	Public variable	BOOL	MV reverse execution condition (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User
	FOTS_EN	Public variable	BOOL	Tight shut/full open execution condition (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User
	MVFO	Public variable	REAL	Output value for full open (Unit: %)	100 to 125	112.5	User
	MVTS	Public variable	REAL	Output value for tight shut (Unit: %)	-25 to 0	-16.82	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MV compensation processing	MV_CMPOUT	Public variable	REAL	Output for MV compensation (Unit: %)	-999999 to 999999	0.0	System

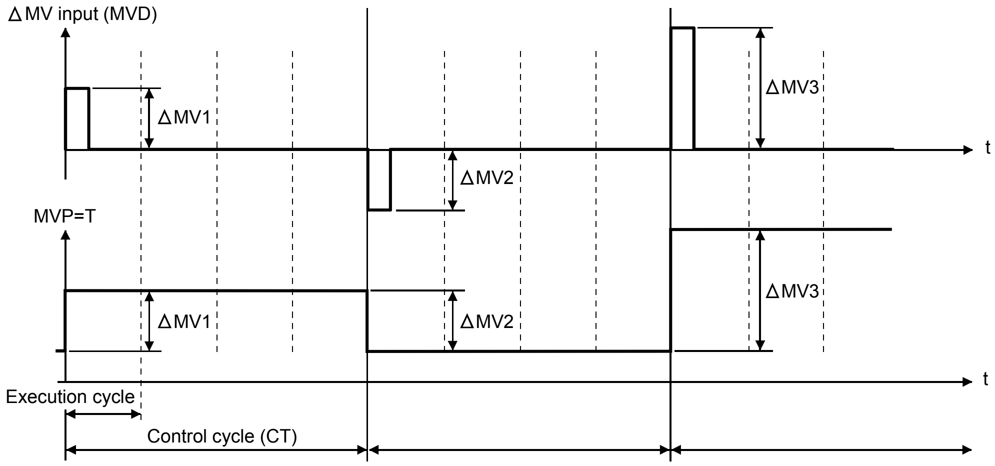
*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

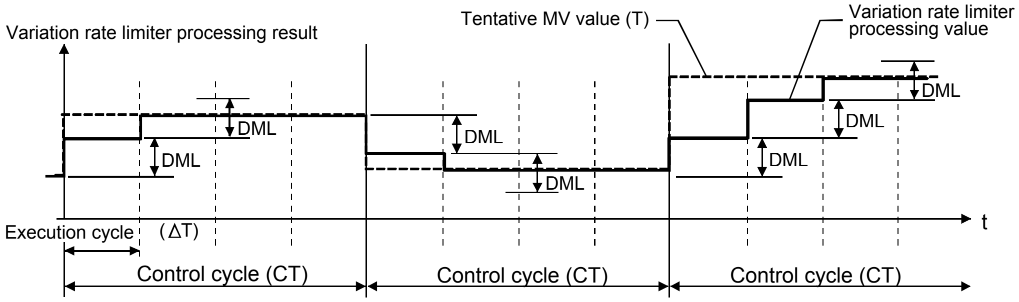
Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input addition</td><td>
Tentative MV value is calculated based on the input values (△MV).
(△MV is output in every control cycle (CT) by FB parts (P_PID and etc.) connected to the input variable).

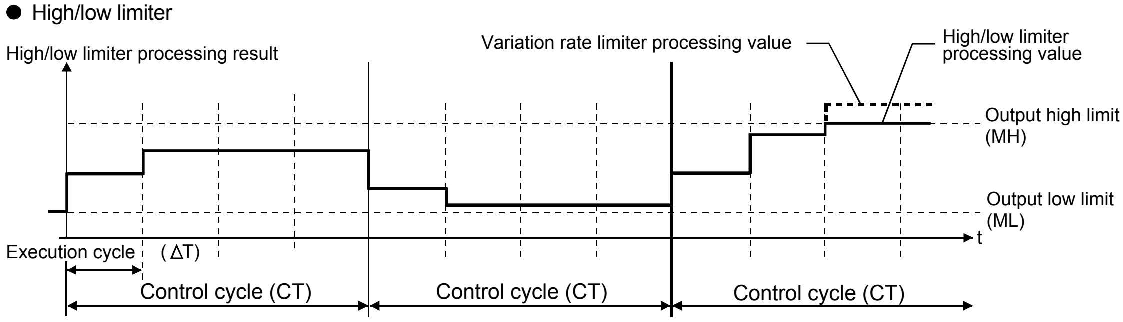


T= △MV+MVP
MVP=T
△MV: Input value to MVD, MVP: MV internal operation value, T: Tentative MV value
● When the MAN, CMV mode are changed to AUT, CSV, manipulated variable (MV) is stored in MV internal operation value (MVP) to avoid sudden MV change.
● While tacking, △MV becomes 0 and manipulated variable is stored into the MV internal operation value (MVP).
</td></tr>
<tr><td>MV compensation</td><td>
Executes compensation processing to the tentative MV.
<table>
<tr><th colspan="2">Condition</th><th>Processing result</th></tr>
<tr><td rowspan="2">MVCMP_EN = TRUE</td><td>MVCMP_MODE = 0 (Addition)</td><td>T + MV_CMPIN</td></tr>
<tr><td>MVCMP_MODE = 1 (Replacement)</td><td>MV_CMPIN</td></tr>
<tr><td rowspan="2">MVCMP_EN = FALSE</td><td>MVCMP_MODE = 0 (Addition)</td><td>T</td></tr>
<tr><td>MVCMP_MODE = 1 (Replacement)</td><td>T</td></tr>
</table>
MVCMP_EN: MV compensation execution condition, T: Tentative MV value, MV_CMPIN: MV compensation value, MVCMP_MODE: MV compensation mode
</td></tr>
<tr><td>CASCADE DIRECT</td><td>
Sets the MV value of primary loop to tentative MV.
<table>
<tr><th>Condition</th><th>Processing result</th></tr>
<tr><td>Control mode = CASCADE DIRECT (CAS=TRUE and CASDR = TRUE)</td><td>CASDR_MVIN</td></tr>
<tr><td>Control mode ≠ CASCADE DIRECT (CASDR=FALSE)</td><td>T</td></tr>
</table>
CASDR_MVIN: MV input for cascade direct, T: Tentative MV value
</td></tr>
<tr><td>MV tracking</td><td>
Switches MV value to tracking input.
<table>
<tr><th>Condition</th><th>Processing result</th></tr>
<tr><td>MVTRK_EN = TRUE</td><td>MV_TRKIN</td></tr>
<tr><td>MVTRK_EN = FALSE</td><td>T</td></tr>
</table>
MVTRK_EN: MV tracking execution condition, T: Tentative MV value, MV_TRKIN: Tracking input
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>MV hold</td><td>Retains MV value to the value of at that point.
<table>
<tr><th>Condition</th><th>Processing result</th></tr>
<tr><td>MVHLD_EN = TRUE</td><td>MVn-1</td></tr>
<tr><td>MVHLD_EN = FALSE</td><td>T</td></tr>
</table>
MVHLD_EN: MV hold execution condition, T: Tentative MV value, MVn-1: Previous MV value</td></tr>
<tr><td>Preset MV</td><td>Switches tentative MV to preset MV value.
<table>
<tr><th>Condition</th><th>Processing result</th></tr>
<tr><td>PREMV_EN = TRUE</td><td>PREMV_V</td></tr>
<tr><td>PREMV_EN = FALSE</td><td>T</td></tr>
</table>
PREMV_EN: Preset MV execution condition, T: Tentative MV value, PREMV_V: Preset MV value</td></tr>
<tr><td>Variation rate limiter and high/low limiter</td><td>Variation rate limiter and high/low limit of the input value are checked.
● Variation rate limiter

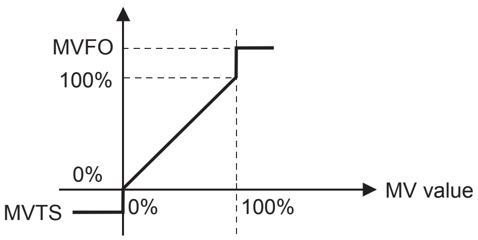


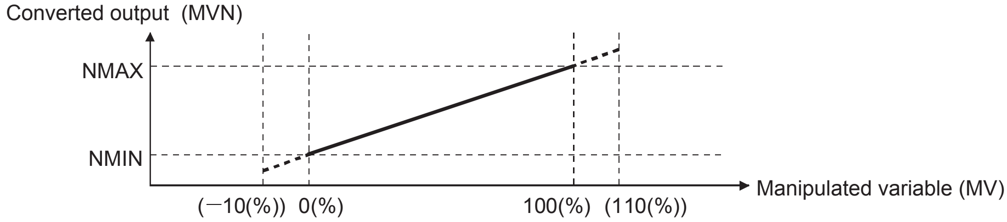
<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">Variation rate limiter processing result</th><th>Alarm (ALM)</th></tr>
<tr><th>Output variation rate limit (DMLA)</th></tr>
<tr><td>|T-MV| ≦ DML</td><td>T</td><td>FALSE (reset)</td></tr>
<tr><td>T-MV > DML</td><td>MV+DML</td><td>TRUE (occur)</td></tr>
<tr><td>T-MV < -DML</td><td>MV-DML</td><td>TRUE (occur)</td></tr>
</table>
T: Tentative MV value, MV: Manipulated variable, DML: Output change high limit
● High/low limiter



<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter processing result</th><th colspan="2">Alarm (ALM)</th></tr>
<tr><th>Output low limit (MLA)</th><th>Output high limit MHA)</th></tr>
<tr><td>Variation rate limiter processing result > MH</td><td>MH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter processing result < ML</td><td>ML</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>ML ≦ Variation rate limiter processing result ≦ MH</td><td>Variation rate limiter processing value</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
MH: Output high limit, ML: Input low limit</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Reset windup</td><td>
Countermeasures against reset windup include: Return MV to high/low limit when MV exceeds the high/low limit; Respond quickly when deviation is inverted.
<table>
<tr><th>Condition</th><th>Countermeasure processing against reset windup</th></tr>
<tr><td>MHA=1 and $\frac{\Delta T}{Ti} \leqq 1$</td><td>MVP= $\frac{\Delta T}{Ti} \times$ (MH-T)+T</td></tr>
<tr><td>MLA=1 and $\frac{\Delta T}{Ti} \leqq 1$</td><td>MVP= $\frac{\Delta T}{Ti} \times$ (ML-T)+T</td></tr>
</table>
MHA: Output high limit, MLA: Output low limit, MH: Output high limit value, ML: Output low limit value,
MVP: MV internal operation value, △T: Execution cycle, T: Tentative MV value, Ti: Integral time
If Ti=0, no countermeasures against reset windup will be performed.

● When ARW_EX_EN is TRUE
Countermeasures against reset windup include: Immediately returns to MV internal operation high/low limit value when MV value exceeds the limit value; Respond quickly when deviation is inverted.
<table>
<tr><th>Condition</th><th>Countermeasure processing against reset windup</th></tr>
<tr><td>MVP > MVPH</td><td>MVP = MVPH</td></tr>
<tr><td>MVP < MVPL</td><td>MVP = MVPL</td></tr>
</table>
MVP: MV internal operation value, MVPH: MV internal operation high limit value,
MVPL: MV internal operation low limit value
However, when MVPH is less than MH, the condition and the pullback value are processed as MH.
When MVPH is more than ML, the condition and the pullback value are processed as ML.
</td></tr>
<tr><td>Tight shut/ full open</td><td>
Use the tight shut/full open function to open or close the control valve completely and absolutely.
Reduce the processing result to the output value for tight shut when the MV value is 0% or lower, and raise it to the output value for full open when it is 100% or higher.
● When FOTS_EN is TRUE


MVTS: Output value for tight shut (%), MVFO: Output value for full open (%)
</td></tr>
<tr><td>MV reverse</td><td>
Executes MV value inversion processing (100-MV).
<table>
<tr><th>Condition</th><th>Processing result</th></tr>
<tr><td>MVREV_EN = TRUE</td><td>MVREV = 100-MV</td></tr>
<tr><td>MVREV_EN = FALSE</td><td>MVREV = MV</td></tr>
</table>
MVREV: Output after processing of MV reverse for internal operation (%), MV: Manipulated variable (%)
</td></tr>
<tr><td>Output conversion</td><td>
Output conversion processing is carried out.


$$\text{Converted output (MVN)}=\{(\text{NMAX-NMIN})\text{x}\ \frac{\text{MVREV}}{100}\}+\text{NMIN}$$
NMAX: Output conversion high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%),
MVREV: Output after processing of MV reverse for internal operation (%), MVN: Converted output
</td></tr>
</table>

Item	Contents
Disable Alarm Detection	Set whether enable Alarm Detection or not in variation rate limiter and high/low limiter. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected. ● ERRI, DMLI, MHI, MLI (2) Disable Alarm Detection by control mode: If the control mode is MAN and CMV, reset DMLA, MHA and MLA of alarm (ALM) and DMLA, MHA and MLA will not be detected. (3) Disable Alarm Detection by stop processing: Please refer to loop stop processing in the following contents.

Other Functions

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output processing at sensor alarm occurrence</td><td>When a sensor error (SEA) occurs in P_IN of tag access FB, select the output of P_OUT3_ from any of the following.
<table>
<tr><th>Condition</th><th>Processing result</th></tr>
<tr><td>SEA_OTYPE = 0</td><td>MV value hold</td></tr>
<tr><td>SEA_OTYPE = 1</td><td>Preset MV output</td></tr>
<tr><td>SEA_OTYPE = 2</td><td>Neither hold nor preset MV output is executed.</td></tr>
</table></td></tr>
<tr><td>Loop stop processing</td><td>The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE.
1) When the output at loop stop is the previous value (STP_OTYPE=0), holds the output (MVN).
When the output at loop stop is the preset value (STP_OTYPE=1), presets the output (MVN).
2) Change the control mode automatically to MANUAL.
3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occur.
4) Alarm is not detected in variation rate limiter and high/low limiter.</td></tr>
<tr><td>Auto tuning (AT2)</td><td>If auto tuning with the Limit Cycle method is in execution, this FB performs in the same way with MANUAL mode.</td></tr>
</table>

Processing Operation

Processing / Control mode	Loop stop	Mode judgment	Input addition	MV compen-sation	CASCADE DIRECT	Preset MV	MV hold	MV tracking	Variation rate limiter and high/ low limiter	Reset windup	MV reverse	Alarm	Auto tuning (AT2)	Output Conver-sion
MAN, CMV	○	○	×	×	×	×	×	×	×	×	○	× (*1)	○	○
AUT, CAS, CSV	○	○	○	○	×	○	○	○	○	○	○	○ (*2)	○	○
CASDR	○	○	○	×	○	○	○	○	○	○	○	○ (*2)	×	○

○: Execute ×: not execute

*1 An alarm (ALM) whose corresponding bit is TRUE (occurred) is reset, and the alarm cannot be detected.
*2 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Output function priority

The following shows the priority in each output function.

Priority	Output function
1	Preset MV
2	MV hold
3	MV tracking
4	Cascade direct
5	MV compensation

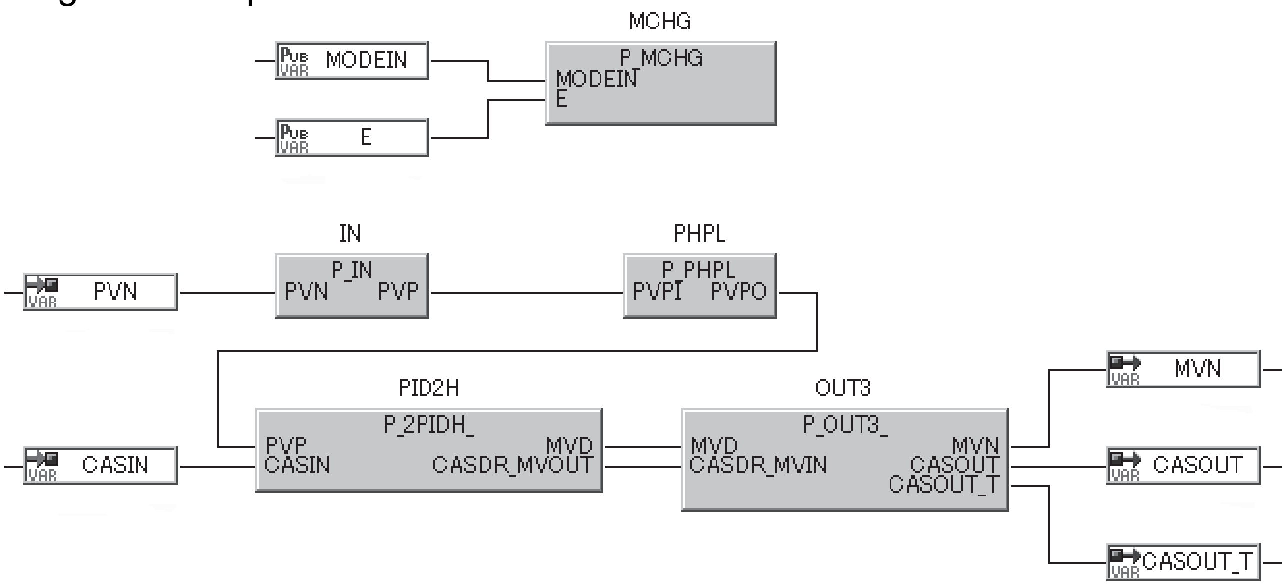
Example) When both preset MV and MV tracking are valid (PREMV_EN=TRUE, MVHLD_EN=TRUE), preset MV value is output as it has higher priority.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

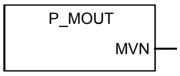
Program Example



POINT

- Module FB It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.
- It is recommended to use the tight shut/full open function with the isolated analog output module having the range setting (extended mode) which can keep an outputable range wider than the normal range setting (4 to 20mA, 1 to 5V).
- For module without extended mode in the range setting, set to 0 to 20mA 0 to 5V in the range setting, and reset output conversion high/low limit value of the FB to enable tight shut/full open function. For details of setting, refer to Appendix 3.19.

8.1.5 Manual Output (P_MOUT)

FB	FBD parts
P_ MOUT	

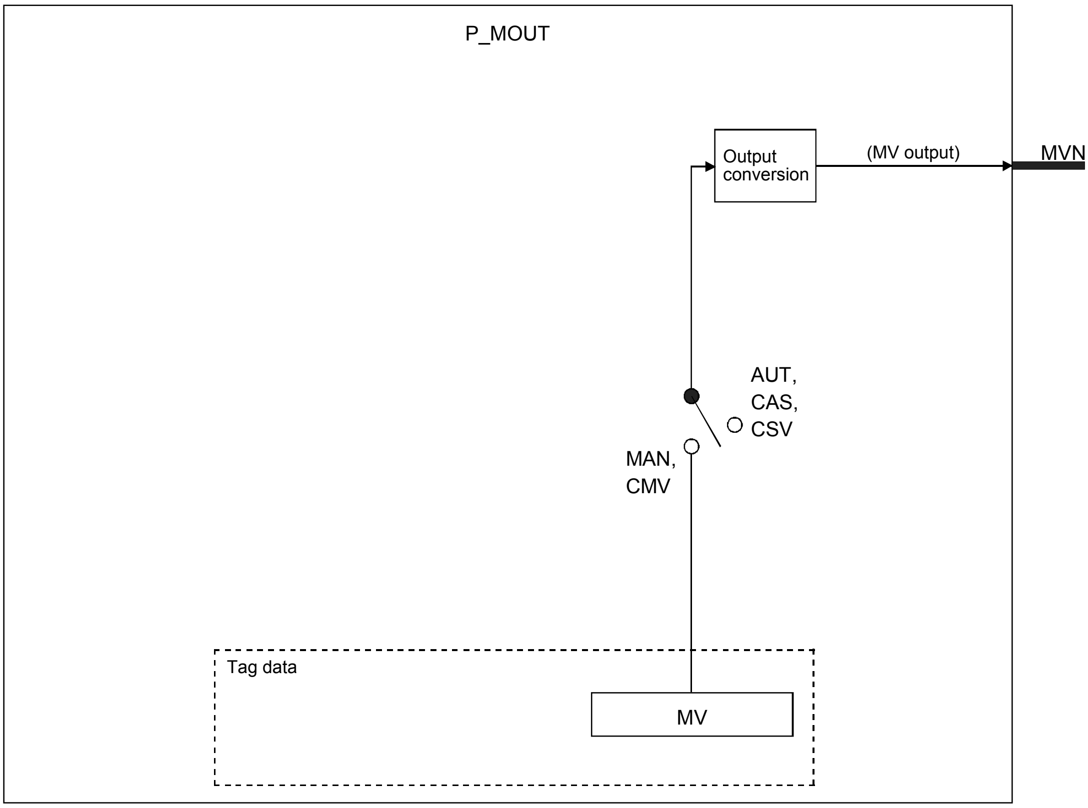
Corresponding tag type
MOUT, MWM

Control mode				
MAN	AUT	CAS	CMV	CSV
○	—	—	○	—

Functions overview: Read the manipulated variable (MV) of the tag data, execute output conversion processing and output MV.

Function/FB classification name: Tag access FB_I/O control FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Output	MVN	Output variable	REAL	MV output to module FB	NMIN to NMAX

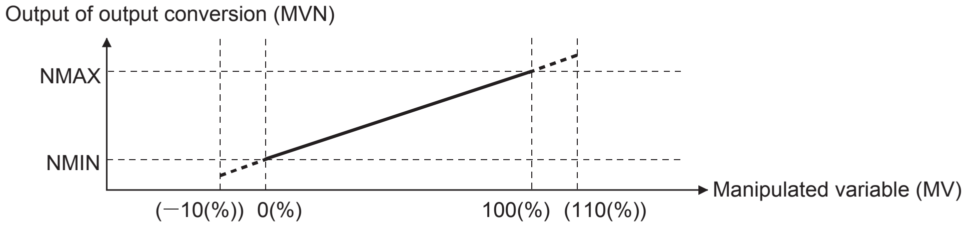
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output conversion</td><td>Execute output conversion processing.

Converted output $(MVN)=\{(NMAX-NMIN)\times\frac{MV}{100}\}+NMIN$
NMAX: Output conversion high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%),
MVN: Converted output</td></tr>
</table>

Processing Operation

Processing / Control mode	Output conversion
MAN, CMV	○
AUT, CAS, CSV	×

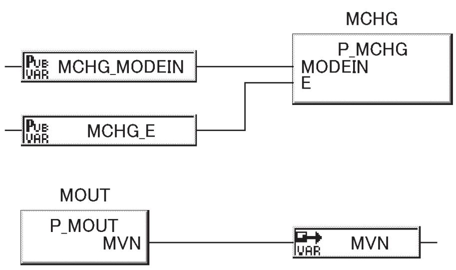
○: Execute ×: Not execute

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.1.6 Time Proportioning Output (P_DUTY)

FB	FBD parts
P_ DUTY	P_DUTY: MVD → MVB, CASOUT, CASOUT_T

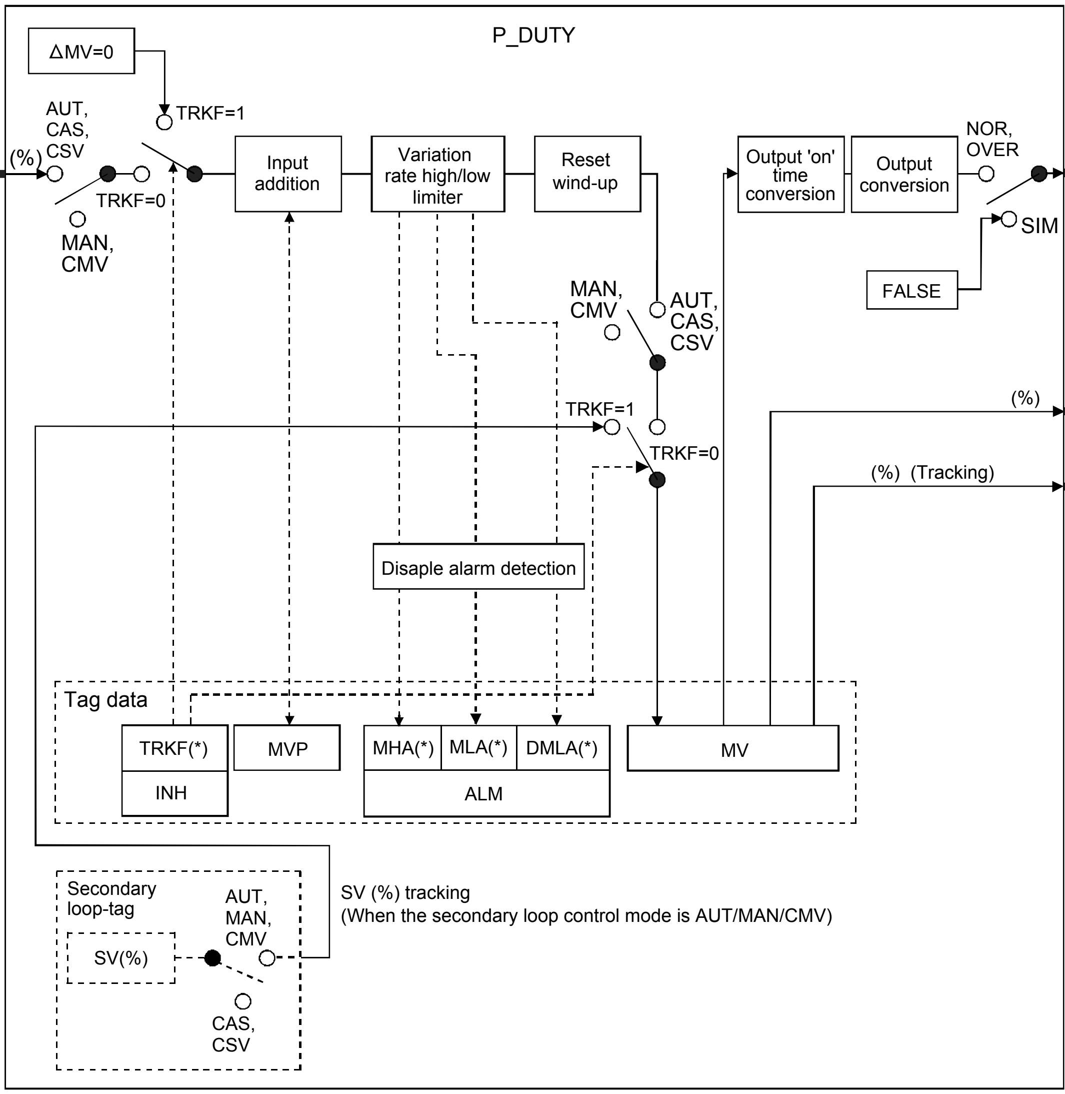
Corresponding tag type
BPI, I PD, PID, SPI, 2PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: For the input value (ΔMV), execute processing as input addition, variation rate limiter and high/low limiter, reset wind-up, output ‘ON’ time conversion and output conversion and output in bit.

Function/FB classification name: Tag access FB_I/O control FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	MVD	Input variable	REAL	△MV input (unit: %)	-999999 to 999999
Output	MVB	Output variable	BOOL	Bit ON/OFF duty output to module FB	TRUE, FALSE
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade output (unit: %) (With tracking)	0 to 100

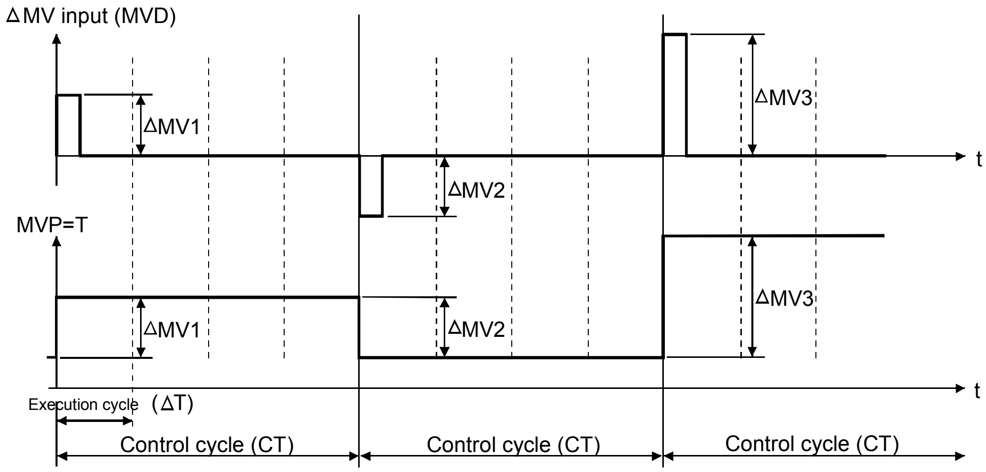
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

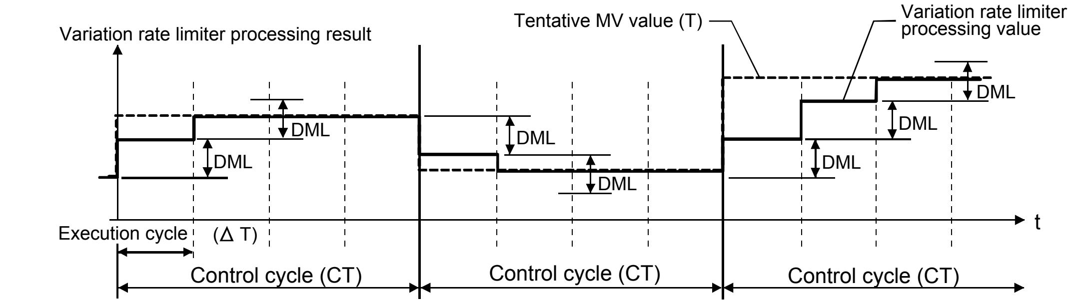
Item	Contents
Input addition	Tentative MV value is calculated based on the input values (△MV) (△MV is output in every control cycle (CT) by FB parts (P_PID and etc.) connected to the input variable.)  T=△MV+MVP MVP=T △MV: Input value to MVD, MVP: MV internal operation value, T: Tentative MV value

Item	Contents

Variation rate limiter and high/low limiter

Variation rate limiter and high/low limiter of input value are checked.

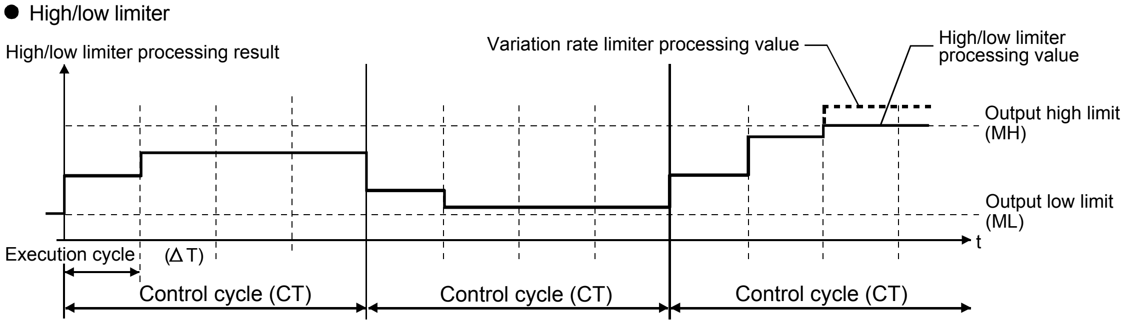
- Variation rate limiter.



Condition	Variation rate limiter processing result	Alarm (ALM)
		Output variation rate limit (DMLA)
$\lvert T-MV \rvert \leqq DML$	T	FALSE (reset)
T-MV >DML	MV+DML	TRUE (occur)
T-MV<-DML	MV-DML	TRUE (occur)

T: Tentative MV value, MV: Manipulated variable, DML: Output variation rate high limit

- High/low limiter



Condition	High/low limiter processing result	Alarm (ALM)	
		Output low limit (MLA)	Output high limit (MHA)
Variation rate limiter processing result>MH	MH	FALSE (reset)	TRUE (occur)
Variation rate limiter processing result<ML	ML	TRUE (occur)	FALSE (reset)
ML ≦ Variation rate limiter processing result ≦ MH	Variation rate limiter processing value	FALSE (reset)	FALSE (reset)

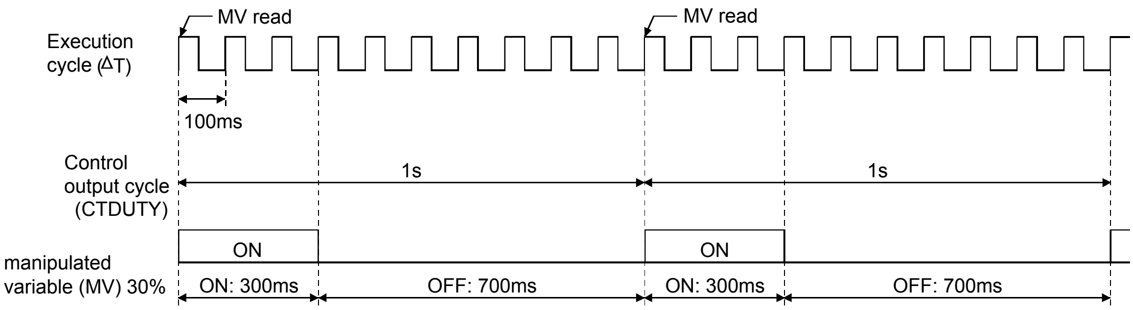
MH: Output high limit, ML: Output low limit

Reset windup

Countermeasures against reset windup include:
Return MV to high/low limit when MV exceeds the high/low limit;
Respond quickly when deviation is inverted.

Condition	Countermeasure processing against reset windup
Variation rate limiter processing result > MH and $\frac{\Delta T}{Ti} \leqq 1$	$MVP=\frac{\Delta T}{Ti} \times (MH-T)+T$
Variation rate limiter processing result < ML and $\frac{\Delta T}{Ti} \leqq 1$	$MVP=\frac{\Delta T}{Ti} \times (ML-T)+T$

MH: Output high limit value, ML: Output low limit value, MVP: MV internal operation value, △T: Execution cycle, T: Tentative MV value, Ti: Integral time
If Ti=0, no countermeasure against reset windup will be taken.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output 'ON' time conversion /Output conversion</td><td>
Duty manipulated variable (MVB) to manipulated variable (MV) is output.
<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Duty manipulated variable (MVBn) 'on' time</td><td>If the 'ON' time execution cycle count is defined as $\left(\frac{CTDUTY \times MV}{\Delta T \times 100}\right)$ with the first digit after decimal point rounded off, then:
Duty manipulated variable (MVB) 'on' time='on' time execution cycle count × △T</td></tr>
<tr><td>Duty manipulated variable (MVBn) 'off' time</td><td>If the 'off' time execution cycle count is defined as (execution cycle count in control cycle) –('on' time execution cycle count) then:
Duty manipulated variable (MVB) 'off' time='off' time execution cycle count × △T</td></tr>
</table>
CTDUTY: Control output cycle, △T: Execution cycle, MV: Manipulated variable (%)
(Example) Suppose execution cycle: △T=100ms, control output cycle: CTDUTY =1.0s,
manipulated variable: MV= 30%


</td></tr>
<tr><td>Disable Alarm Detection</td><td>
Set whether enable Alarm Detection or not in variation rate limiter and high/low limiter.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected.
● ERRI, DMLI, MHI, MLI
(2) Disable Alarm Detection by control mode:
If the control mode is MAN and CMV, reset DMLA, MHA and MLA of alarm (ALM) and DMLA, MHA and MLA will be detected.
(3) Disable Alarm Detection by stop processing:
Please refer to loop stop processing in the following contents.
</td></tr>
</table>

Other Functions

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Holding processing</td><td>
Set whether to hold output of P_DUTY when sensor error (SEA) occurs on tag access P_IN. The setting can be made through PX Developer project parameter setting.
[Setting procedure][Project Parameter] → [Program Execution] → Holding processing
● "Hold the output of P_OUT1, P_OUT2, P_DUTY" selected: Hold output
● "Hold the output of P_OUT1, P_OUT2, P_DUTY" unselected': Continue operation
</td></tr>
<tr><td>Loop stop processing</td><td>
The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE.
1) Hold the output (MVB).
2) Change the control mode automatically to MANUAL.
3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occurs.
4) Alarm is not detected in range check.
</td></tr>
</table>

Processing Operation

Control mode \ Processing	Input processing	Variation rate limiter and high/low limiter	Reset windup	Output 'ON' time conversion	Output conversion	Alarm
MAN, CMV	×	×	×	○	○	× (*1)
AUT, CAS, CSV	○	○	○	○	○	○ (*2)

○ : Execute × : Not execute

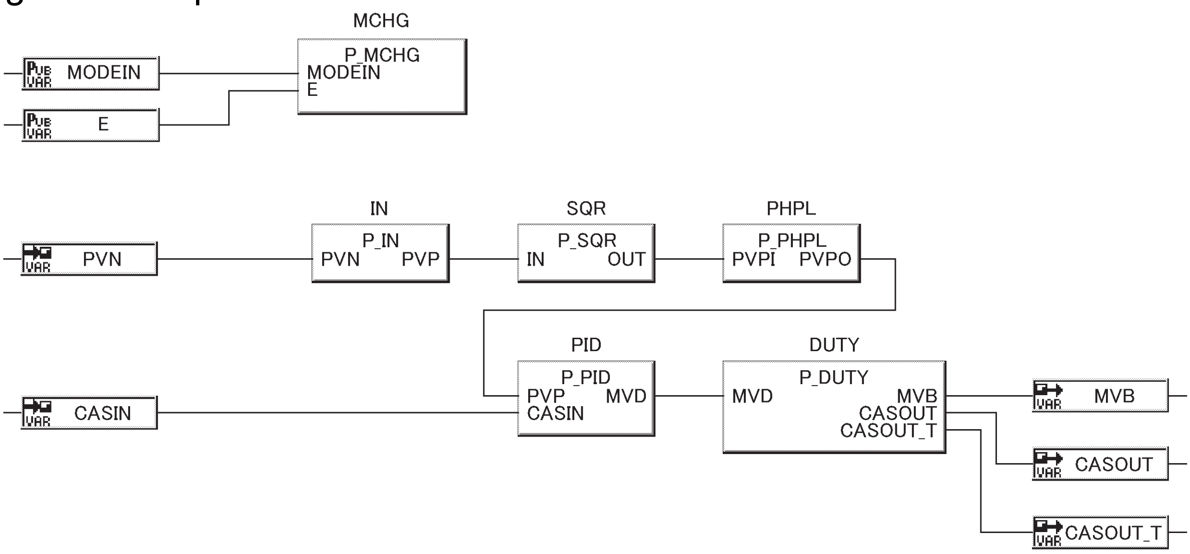
*1 An alarm (ALM) whose corresponding bit is TRUE (occurred) is reset, and the alarm cannot be detected.
*2 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

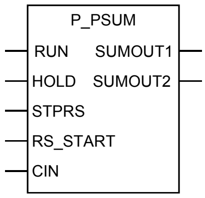
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.1.7 Pulse Integration (P_PSUM)

FB	FBD parts
P_ PSUM	

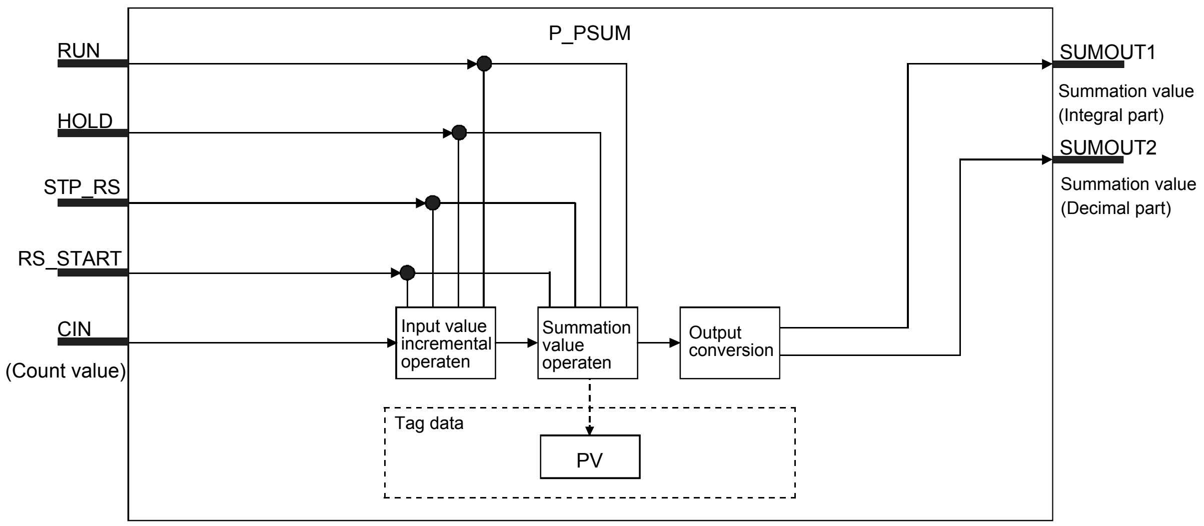
Corresponding tag type
PSUM,BC

Control mode				
MAN	AUT	CAS	CMV	CSV
–	–	–	–	–

Functions overview: It executes the input value incremental operation, integration value operation and output conversion for the count value when the integration start signal (RUN) is TRUE, then output the result.

Function/FB classification name: Tag access FB_I/O control FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	RUN	Input variable	BOOL	Integration start signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	HOLD	Input variable	BOOL	Integration pause signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	STPRS	Input variable	BOOL	Reset signal after integration pause (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	RS_START	Input variable	BOOL	Start signal after integration reset (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	CIN	Input variable	DINT	Count value	-2147483648 to 2147483647 ring counter (pulse increment for each execution should be less than 32767)
Output	SUMOUT1	Output variable	DINT	Integration value output (integral part)	0 to HILMT
	SUMOUT2	Output variable	DINT	Integration value output (decimal part)	0 to 999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	W	Public variable	INT	Weight per pulse	1 to 999	1	User
	U	Public variable	INT	Unit conversion constant	1,10,100,1000	1	User
	HILMT	Public variable	DINT	Integration high limit	0 to 2147483647	2147483647	User
	SUMPTN	Public variable	INT	Integration pattern: 0: return to 0 if over integration high limit. 1: hold the integration high limit value if over integration high limit	0,1	0	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

Item	Contents
Input condition	If the input variable RUN is TRUE, integration processing is carried out to the input (CIN) and integration value is exported. If the input variable HOLD is TRUE, the integration processing to the input (CIN) is held. If the input variable STPRS is TRUE, integration processing is stopped and integration value is cleared. If the input variable RS_START is TRUE, integration processing is restarted after resetting the integration processing.
Input value incremental operation	Execute the following operations to the input (CIN): (see table below)
Integration value operation	Execute the following operations to the input increment that is calculated by input incremental operation. (see table below)

Input value incremental operation:

Integration start signal (RUN)	Integration pause signal (HOLD)	Input value incremental processing result (T1)
FALSE	FALSE	–
FALSE	TRUE	–
TRUE	FALSE	CIN − CINn-1
TRUE	TRUE	–

CIN: Count value, $CINn_{-1}$: Previous count value, T1: Input value incremental processing result

Integration value operation:

Integration start signal (RUN)	Integration pause signal (HOLD)	Integration value operation processing result (T2: Integration value (integral part), T3: Integration value (decimal part))
FALSE	FALSE	T2=0,T3=0
FALSE	TRUE	T2=0,T3=0
TRUE	FALSE	T4= Quotient of {(T1 × W)/U} (integral part) T5= Modulus of {(T1 × W)/U}(decimal part) T2= Quotient of PV+T4+[{SUM2+T5}/U](integral part) T3= Modulus of [{SUM2+T5}/U] (decimal part)
TRUE	TRUE	T2=PV, T3=SUM2

T1: Input value incremental processing result, T2: Integration value (integral part), T3: Integration value (decimal part), T4: Integration value increment (integral part), T5: Integration value increment (decimal part), W: Weight per pulse, U: Unit conversion constant, PV: Integration value (integral part), SUM2: Integration value (decimal part)

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output conversion</td><td>Execute the following operation to the integration value that is calculated by integration value increment processing.
<table>
<tr><th rowspan="2">Integration pattern (SUMPTN)</th><th rowspan="2">Condition</th><th colspan="2">Output variable (SUMOUT1, SUMOUT2)</th><th colspan="2">Tag data (PV,SUM2)</th></tr>
<tr><th>Integration value (integral part) (SUMOUT1)</th><th>Integration value (decimal part) (SUMOUT2)</th><th>Integration value (integral part) (PV)</th><th>Integration value (decimal part) (SUM2)</th></tr>
<tr><td rowspan="2">0</td><td>T2>HILMT</td><td>SUMOUT1= T2－HILMT－1</td><td>SUMOUT2=T3</td><td>PV= T2－HILMT－1</td><td>SUM2=T3</td></tr>
<tr><td>Else</td><td>SUMOUT1=T2</td><td>SUMOUT2=T3</td><td>PV=T2</td><td>SUM2=T3</td></tr>
<tr><td rowspan="2">1</td><td>T2>HILMT</td><td>SUMOUT1=HILMT</td><td>SUMOUT2=0</td><td>PV=HILMT</td><td>SUM2=0</td></tr>
<tr><td>Else</td><td>SUMOUT1=T2</td><td>SUMOUT2=T3</td><td>PV=T2</td><td>SUM2=T3</td></tr>
</table>
T2: Integration value (integral part), T3: Integration value (decimal part), PV: Integration value (integral part), SUM2: Integration value (decimal part), SUMOUT1: Integration value (integral part) output, SUMOUT2: Integration value (decimal part) output</td></tr>
</table>

Processing operation

Processing / Control mode	Input value incremental operation	Integration operation	Output conversion
—	○	○	○

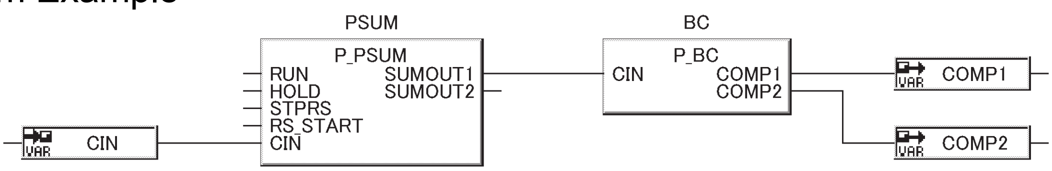
○: Execute ×: Not execute

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.1.8 Batch Counter (P_BC)

FB	FBD parts
P_BC	P_BC: CIN → COMP1, COMP2

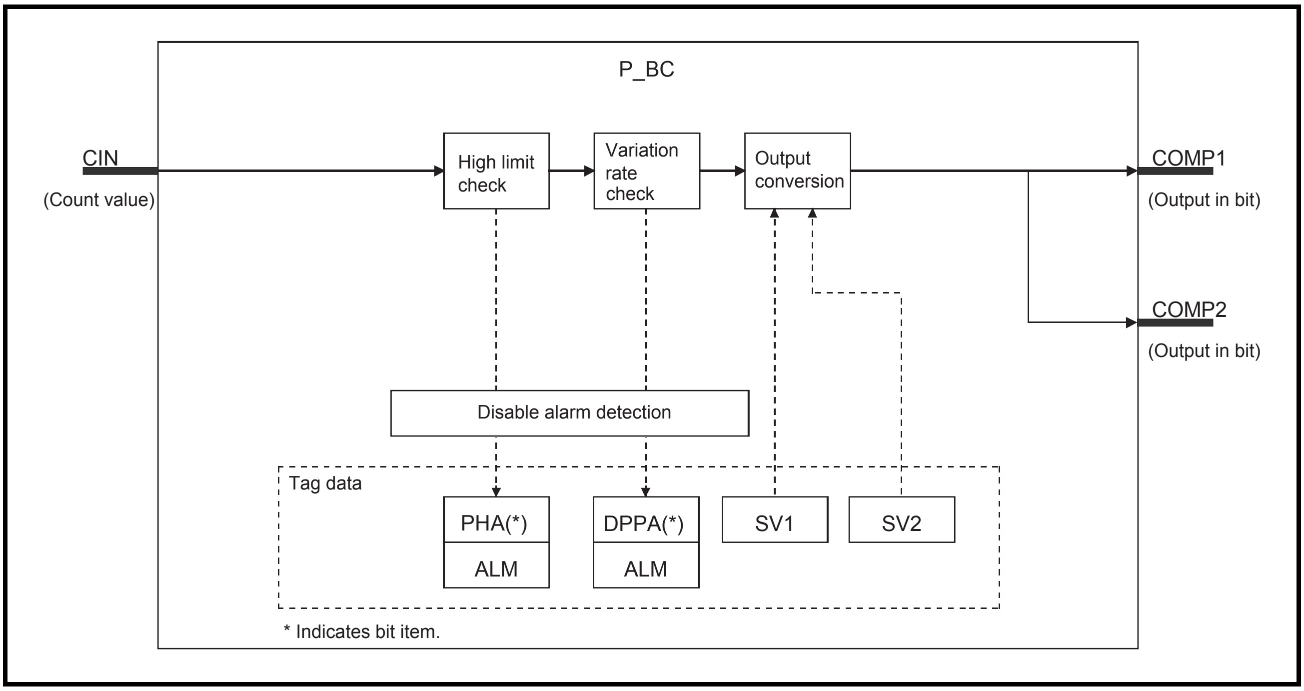
Corresponding tag type
BC

Control mode				
MAN	AUT	CAS	CMV	CSV
–	–	–	–	–

Functions overview: Compare the input (CIN) with setting value 1 and setting value 2. Complete signal is output when the input reaches setting value. Carry out high limit check, variation rate check, and output conversion processing.

Function/FB classification name: Tag access FB_I/O control FB

Block Diagram



Input and Output Pins

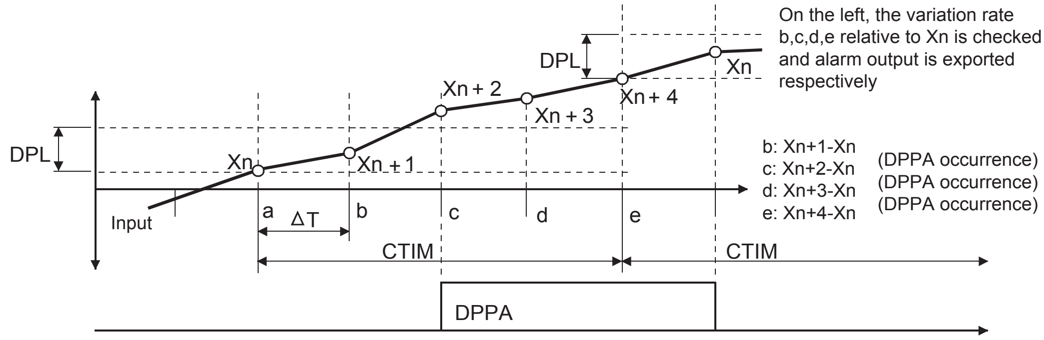
Pin	Variable name	Variable type	Data type	Contents	Range
Input	CIN	Input variable	DINT	Count value	0 to 99999999
Output	COMP1	Output variable	BOOL	Setting value 1 (SV1) completed output (TRUE:ON, FALSE:OFF)	TRUE, FALSE
	COMP2	Output variable	BOOL	Setting value 2 (SV2) completed output (TRUE:ON, FALSE:OFF)	TRUE, FALSE

Tag Data

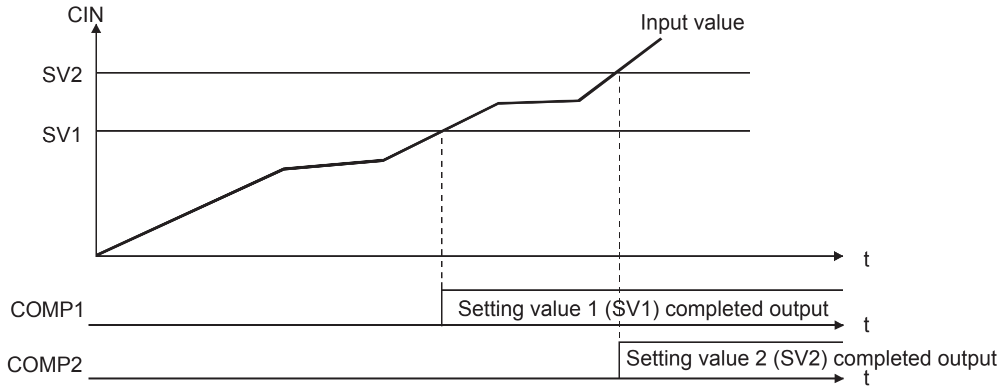
For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>High limit check</td><td>
Execute the high limit check to the input value (CIN).
<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Input high limit (PHA)</th></tr>
<tr><td>CIN > PH</td><td>TRUE (occur)</td></tr>
<tr><td>Else</td><td>FALSE (reset)</td></tr>
</table>
CIN: Count value, PH: PV high limit alarm value
</td></tr>
<tr><td>Variation rate check</td><td>
For a variation rate alarm check time period (CTIM), the change of the input (DPL) in each execution cycle (△T) is compared with the variation rate alarm value, thus variation rate alarm is detected.



<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Positive variation rate (DPPA)</th></tr>
<tr><td>(Xn+m)-Xn ≧ DPL</td><td>TRUE (occur)</td></tr>
<tr><td>Else</td><td>FALSE (reset)</td></tr>
</table>
DPL: Variation rate alarm value (%), m: Variation rate monitor counter= $\frac{CTIM}{\Delta T}$ (*1) , △T: Execution cycle,
CTIM: Variation rate alarm check time
*1 Set the CTIM and ΔT so that 'm' is greater than or equal to 2.
</td></tr>
<tr><td>Output conversion</td><td>
Execute output conversion processing.



<table>
<tr><th>Condition</th><th>Setting value 1 (SV1) completed output (COMP1)</th><th>Setting value 2 (SV2) completed output (COMP2)</th></tr>
<tr><td>CIN<0</td><td>FALSE</td><td>FALSE</td></tr>
<tr><td>0≦CIN<SV1</td><td>FALSE</td><td>–</td></tr>
<tr><td>CIN≧SV1</td><td>TRUE</td><td>–</td></tr>
<tr><td>0≦CIN<SV2</td><td>–</td><td>FALSE</td></tr>
<tr><td>CIN≧SV2</td><td>–</td><td>TRUE</td></tr>
</table>
CIN: Count value, SV1: Setting value 1, SV2: Setting value 2, COMP1: Setting value 1 (SV1) completed output value, COMP2: Setting value 2 (SV2) completed output value
</td></tr>
</table>

Item	Contents
Disable Alarm Detection	Set whether enable Alarm Detection or not in variation rate detection and high/low detection. If the following bit items of Disable Alarm Detection (INH) of tag data are TURE, alarm (ALM) of PHA, and DPPA will not be detected. ● ERRI, PHI, DPPI

Processing Operation

Processing / Control mode	Input value incremental operation	Integration operation	Output conversion	Alarm
—	○	○	○	○ (*1)

○: Execute ×: Not execute

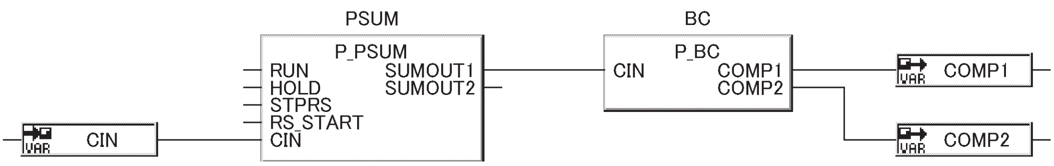
*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

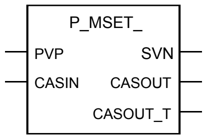
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.1.9 Manual Setter (P_MSET_)

FB	FBD parts
P_MSET_	

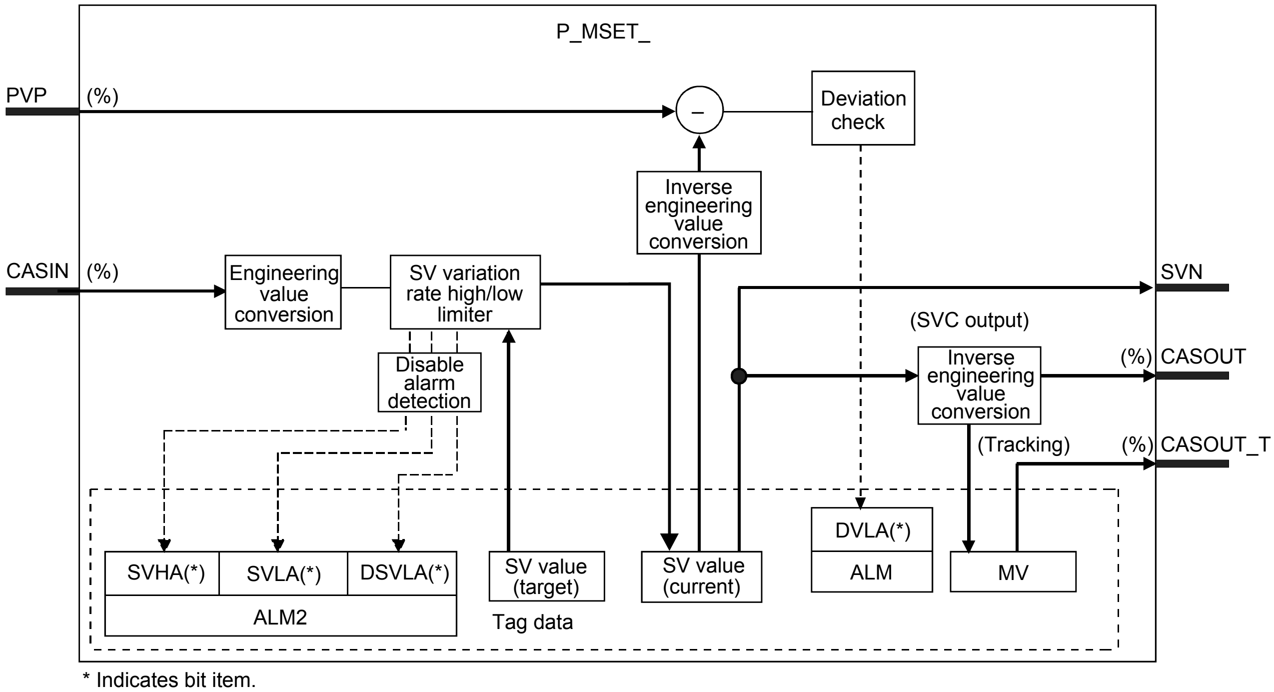
Corresponding tag type
SWM

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	–	○

Functions overview: Executes the processing of SV variation rate and high/low limiter, and set the value after processing it to Setting value (current) (SVC) of tag data. After that, output SVC.

Function/FB classification name: Tag access FB_I/O control FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	SVN	Output variable	REAL	SV output to module FB	-999999 to 999999
	CASOUT	Output variable	REAL	Cascade SV output (unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade SV output (unit: %) (With tracking)	0 to 100

Public Variable (Operation constant)

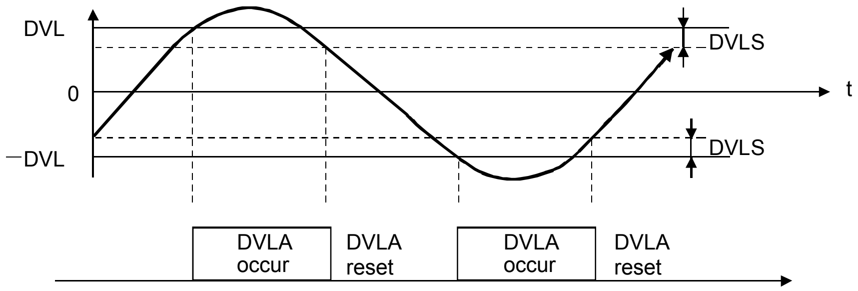
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used*1 (TRUE: Not used, FALSE: Use)	TRUE, FALSE	TRUE	User
	SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

*1 When SVPTN_B0 is TRUE, even if the mode is changed to the CAS mode, the CASIN input cannot be used.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
(1) Execute deviation check processing.


<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL < |DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>
DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
(2) Deviation for direct/reverse action (DV) is calculated as follows.
<table>
<tr><th>Condition</th><th>Deviation (DV)</th></tr>
<tr><td>Direct action (PN = 1)</td><td>DV (%) = PVP (%) – SVC (%)</td></tr>
<tr><td>Reverse action (PN = 0)</td><td>DV (%) = SVC (%) – PVP (%)</td></tr>
</table>
DV: Deviation (%), PVP (%): PV input value (%), SVC (%) = $\frac{100}{RH-RL} \times (SVC - RL)$,
RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting value (current)
</td></tr>
<tr><td>Engineering value conversion</td><td>
Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.
<table>
<tr><td>SV = (RH – RL) × Setting value (%) from the primary loop + RL</td></tr>
</table>
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>SV variation rate high/low limiter</td><td>
Checks variation rate high/low limiter to SV value (target value) in every control cycle (CT).
(1) Variation rate limiter
SV variation rate high limit value inputted in % is converted to engineering value and the processing will be executed.
DSVL → DSVLT (DSVL: SV variation rate high limit value, DSVLT: value converted to engineering value from SV variation rate high limit value)
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>| SV - SVC | ≦ DSVLT</td><td>SV</td><td>FALSE (reset)</td></tr>
<tr><td>SV - SVC > DSVLT</td><td>SVC + DSVLT</td><td>TRUE (occur)</td></tr>
<tr><td>SV - SVC < - DSVLT</td><td>SVC - DSVLT</td><td>TRUE (occur)</td></tr>
</table>
SV: Setting value (target), SVC: Setting value (current)
If DSVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, DSVLA will be FALSE.

(2) High/low limiter
• SVLMT_EN is TRUE.
<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter result</th><th colspan="2">Alarm2 (ALM2)</th></tr>
<tr><th>Target lower limit (SVLA)</th><th>Target upper limit (SVHA)</th></tr>
<tr><td>Variation rate limiter result > SH</td><td>SH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter result < SL</td><td>SL</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>SL ≦ variation rate limiter result ≦ SH</td><td>Variation rate limiter result</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
If SVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVLA will be FALSE.
If SVHI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SHLA will be FALSE.
High/low limiter result is stored to SVC (setting value (current)).
• SVLMT_EN is FALSE.
Variation rate limiter result is stored to SVC (setting value (current)).
</td></tr>
<tr><td>Inverse engineering value conversion</td><td>
Convert the setting value (SV) of engineering value to percentage MV (%).
$$MV\ (\%) = \frac{100}{RH - RL} \times (SV - RL)$$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value, MV: Manipulated variable
</td></tr>
<tr><td>Tracking processing</td><td>When the tracking is requested by the following FB via CASOUT_T, the value entered to MV is converted in engineering value and the tracking is performed to SV and SVC.</td></tr>
</table>

Other Function

Item	Contents
Loop stop processing	The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE. 1) Change the control mode automatically to MANUAL. 2) Reset DLVA when DLVA of alarm (ALM) occurs. Reset DSVLA, SVLA, and SVHA when DSVLA, SVLA, and SVHA of alarm2 (ALM2) occur. 3) Alarm is not detected in deviation check and variation rate high/low limiter.

Processing Operation (*1)

Control mode \ Processing	Deviation check	Engineering value conversion	Alarm	SV variation rate high/low limiter
MAN, AUT	○	×	○ (*2)	○ (*3)
CAS, CSV	○	○	○ (*2)	○

○: Execute ×: Not execute

*1 The processing operation of the tag access FB is executed in every control cycle (CT).

*2 When the bit of Disable Alarm Detection (INH) which corresponding to an alarm is TRUE, the alarm is not detected.

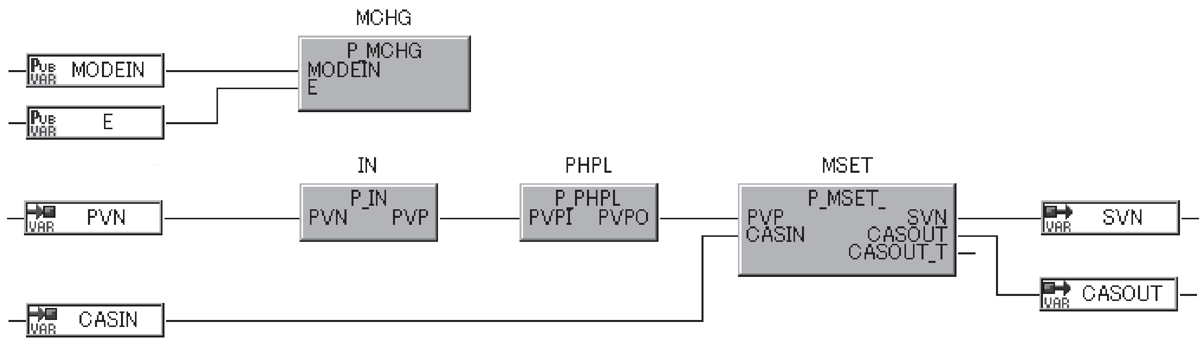
*3 When the control mode is MAN, SV variation rate limiter processing is not executed.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

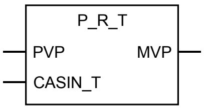


POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2 Tag Access FB _ Loop Control Operation FB

8.2.1 Ratio Control (With Tracking to primary loop) (P_R_T)

FB	FBD parts
P_R_T	

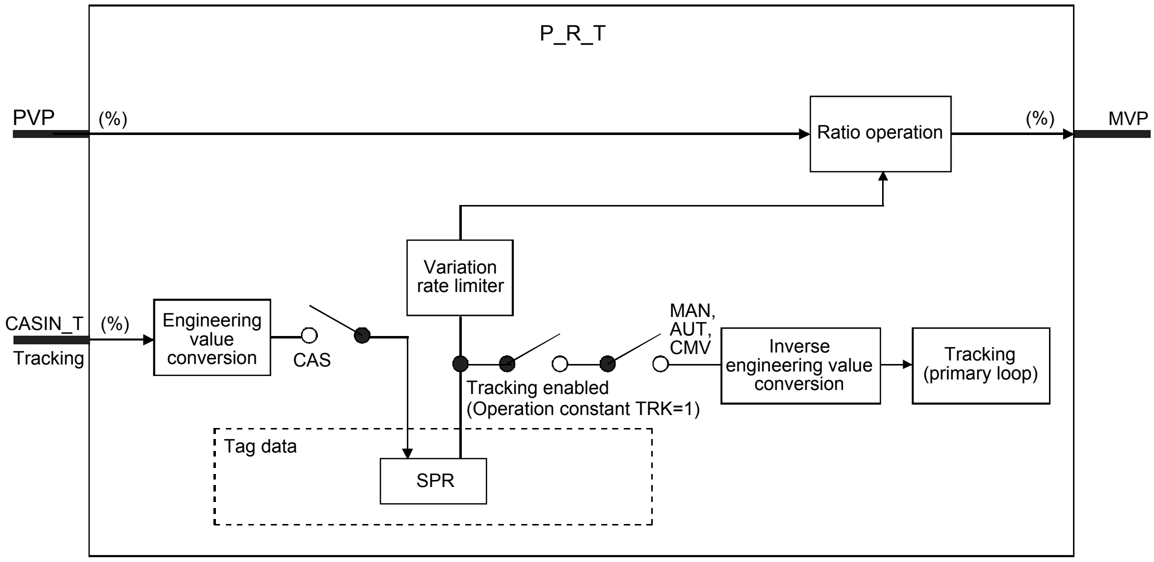
Corresponding tag type
R

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Control 2 control volumes at a constant ratio and output (△MV).

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	Cascade SV input (unit: %) (With tracking)	0 to 100
Output	MVP	Output variable	REAL	MV output (unit: %)	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TRUE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

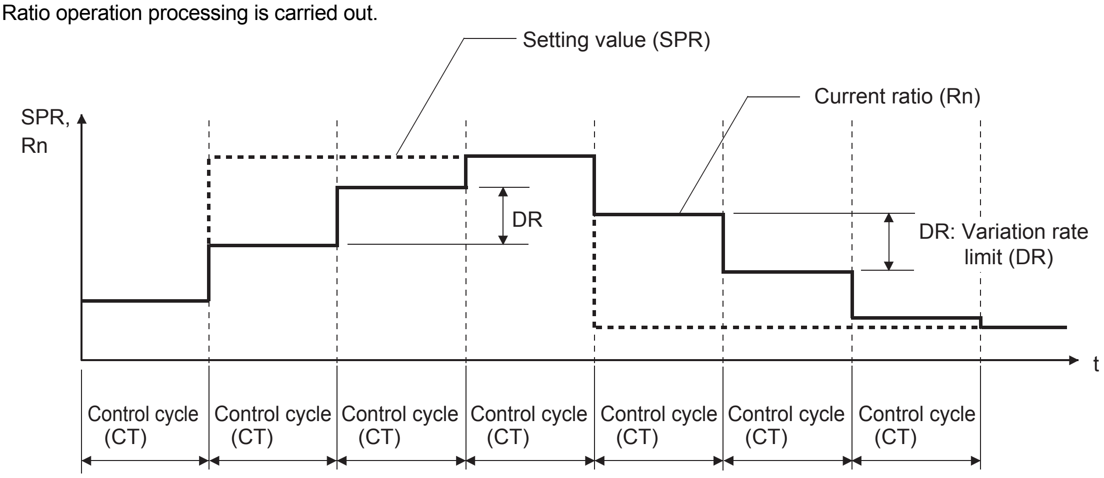
Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Engineering value conversion</td><td>Execute engineering value conversion.

$$SPR=\frac{RMAX-RMIN}{100}\times CASIN+RMIN$$

SPR: Setting value (%), RMIN: Ratio low limit (%), RMAX: Ratio high limit (%),
CASIN: Setting value from upper system (%)</td></tr>
<tr><td>Variation rate limiter</td><td>Execute variation rate limiter processing.
<table>
<tr><th>Condition</th><th>Variation rate limiter processing result</th></tr>
<tr><td>(SPR-Rn)≧DR</td><td>Rn=Rn-1+DR</td></tr>
<tr><td>(SPR-Rn)≦DR</td><td>Rn=Rn-1-DR</td></tr>
<tr><td>|SPR-Rn|<DR</td><td>Rn=SPR</td></tr>
</table>
SPR: Setting value (%), DR: Variation rate limit (%), Rn: Current ratio value (%), Rn-1: Previous ratio value</td></tr>
<tr><td>Ratio operation</td><td>Ratio operation processing is carried out.


$$\text{Ratio operation value (MVP)(\%)} = \frac{Rn-RMIN}{RMAX-RMIN}\times PVP+BIAS$$

Rn: current ratio (%), RMIN: Ratio low limit (%), RMAX: Ratio high limit (%), PVP: PV input (%), BIAS: bias</td></tr>
<tr><td>Inverse engineering value conversion</td><td>Execute engineering value conversion processing when tracking flag (TRK) is 1.

$$SV(\%)=\frac{SPR-RMIN}{RMAX-RMIN}\times 100(\%)$$

SV: Tracking data of primary loop (%), SPR: Setting value (%), RMIN: Ratio low limit (%), RMAX: Ratio high limit (%)</td></tr>
<tr><td>Tracking processing</td><td>Whether execute tracking processing to the input variable CASIN_T is described in the following table:
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SVPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table></td></tr>
</table>

Other Functions

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE. 1) Hold the output (MVP). 2) Change the control mode automatically to MANUAL.

Processing Operation

Control mode \ Processing	Ratio operation	Variation rate limiter	Engineering value conversion	Tracking
MAN, CMV	○	○	×	○ (*1)
AUT, CAS, CSV	○	○	○	×

○: Execute ×: Not execute

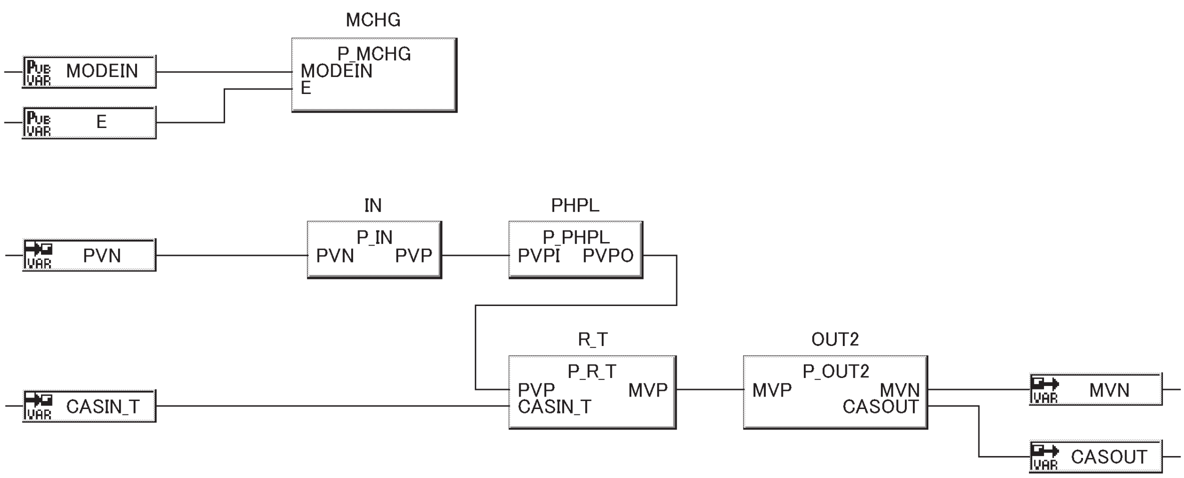
*1 Tracking is executed when the tracking flag (TRK) is 1.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

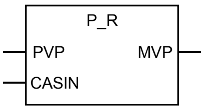
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.2 Ratio Control (Without Tracking to primary loop) (P_R)

FB	FBD parts
P_R	

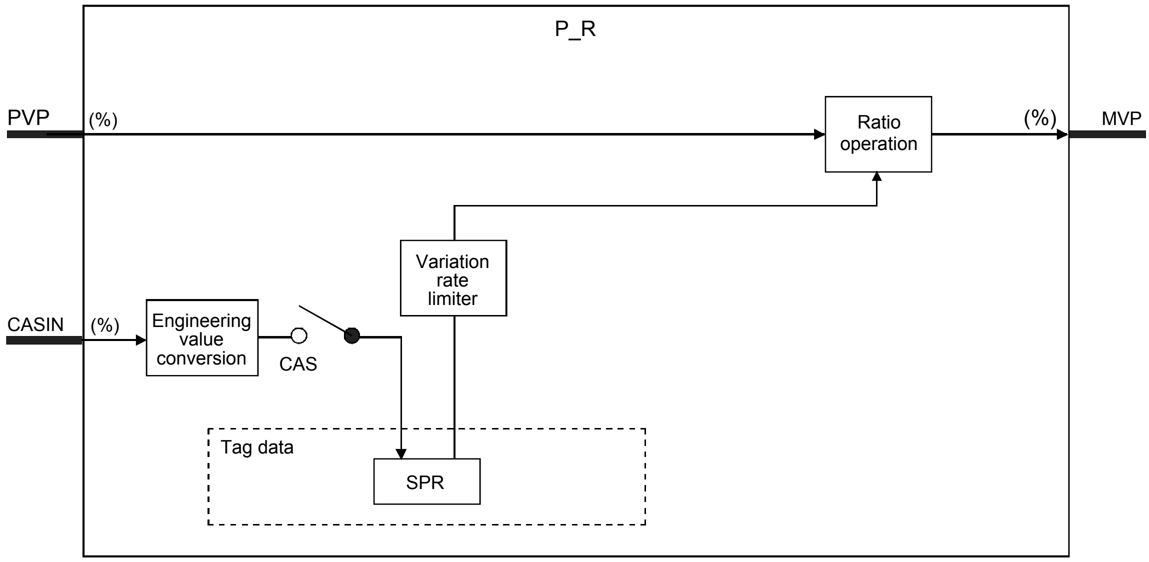
Corresponding tag type
R

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Control 2 control variables at a constant ratio.

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVP	Output variable	REAL	MV output (unit: %)	-999999 to 999999

Public Variable (Operation constant)

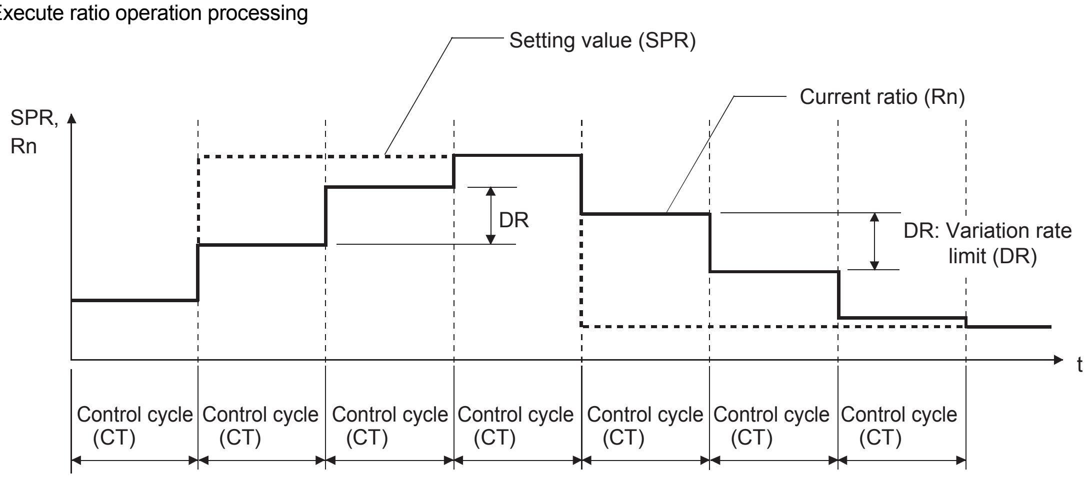
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Engineering value conversion</td><td>Execute engineering value conversion processing.

$$SPR= \frac{RMAX-RMIN}{100} \times CASIN+RMIN$$
SPR: Setting value (%), RMIN: Ratio low limit (%), RMAX: Ratio high limit (%), CASIN: Setting value from host (%)</td></tr>
<tr><td>Variation rate limiter</td><td>Execute the variation rate limiter processing.
<table>
<tr><th>Condition</th><th>Variation rate limiter processing result (Rn)</th></tr>
<tr><td>(SPR − Rn) ≧ DR</td><td>Rn = Rn_{-1} + DR</td></tr>
<tr><td>(SPR − Rn) ≦ DR</td><td>Rn = Rn_{-1} - DR</td></tr>
<tr><td>|SPR—Rn| < DR</td><td>Rn = SPR</td></tr>
</table>
SPR: Setting value (%), DR: Variation rate limit (%), Rn: Current ratio value (%), Rn_{-1}: Previous ratio value</td></tr>
<tr><td>Ratio operation</td><td>Execute ratio operation processing


$$\text{Ratio operation value (MVP)(\%)} = \frac{Rn-RMIN}{RMAX-RMIN} \times PVP+BIAS$$
Rn: Current ratio (%), RMIN: Ratio low limit (%), RMAX: Ratio high limit (%), PVP: PV input (%), BIAS: Bias</td></tr>
</table>

Other Functions

Item	Contents
Loop stop processing	Execute the following operation when the stop alarm (SPA) of alarm (ALM) is TRUE. 1) Hold the output (MVP). 2) Automatically change the control mode into Manual (MANUAL).

Processing Operation

Control mode \ Processing	Ratio operation	Variation rate limiter	Engineering value conversion
MAN, CMV	○	○	×
AUT, CAS, CSV	○	○	○

○: Execute ×: Not execute

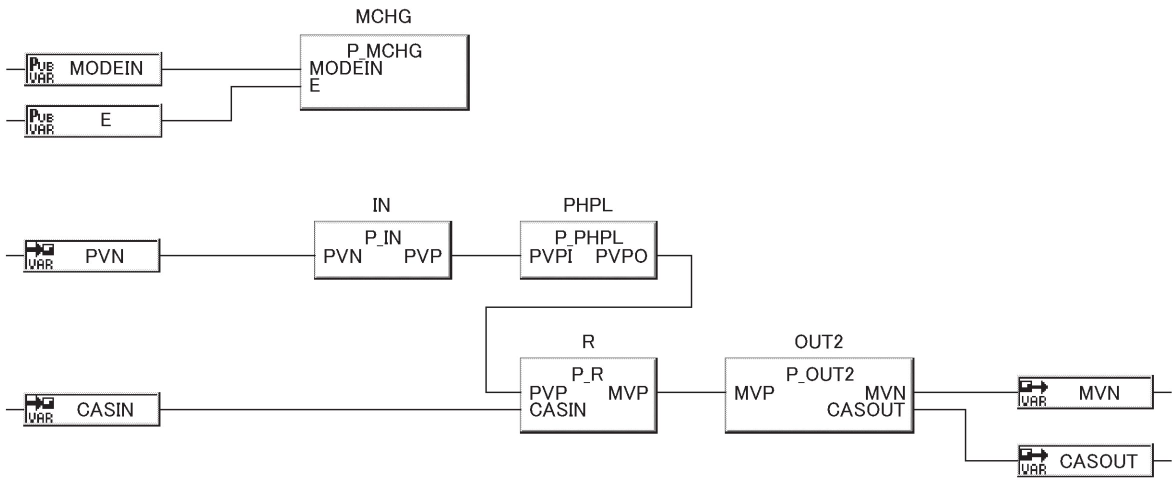
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.3 Velocity Type PID Control (With Tracking to primary loop) (P_PID_T)

FB	FBD parts
P_PID_T	P_PID_T PVP — MVD CASIN_T

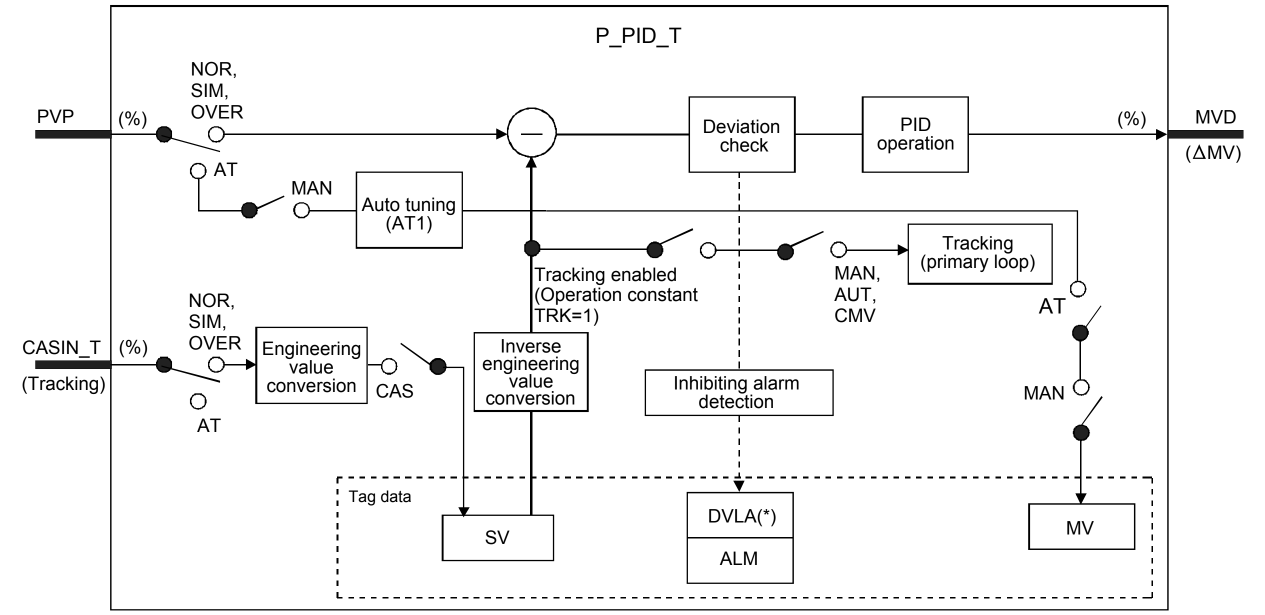
Corresponding tag type
PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Execute PID operation by use of PV- derivative, imperfect derivative, velocity type, and output (ΔMV).

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	Cascade SV input (unit: %) (With tracking)	0 to 100
Output	MVD	Output variable	REAL	ΔMV output (unit: %)	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Use)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TRUE	User

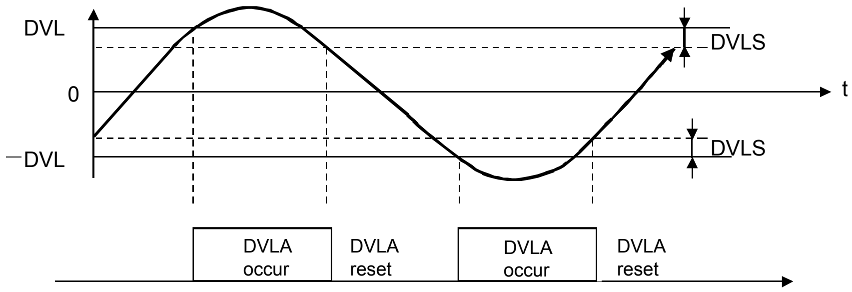
*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
Execute deviation check processing



<table>
<tr><td rowspan="2">Condition</td><td>Alarm (ALM)</td></tr>
<tr><td>Large deviation (DVLA)</td></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV|≦ (DVL−DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

Item	Contents
PID operation	

(1) Gain (Kp) is calculated as below:

$$Kp = K \times P$$

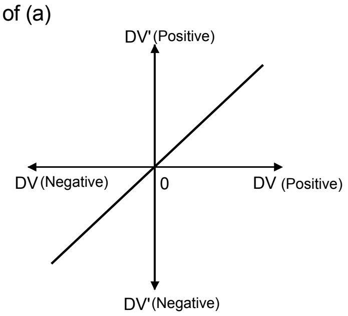
K: Output gain, P: Gain

(2) Output gain (K) is calculated as below:

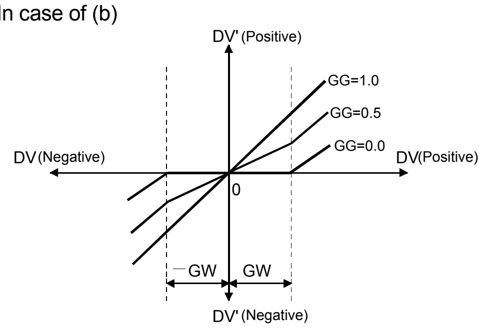
Condition		Output gain (K)
(a) When gap width (GW) = 0, K value to the deviation (DV).		K=1
(b) When gap width (GW) > 0, K value to the deviation (DV)	$\lvert DV \rvert \leqq GW$	K=GG
	$\lvert DV \rvert > GW$	$K=1-\dfrac{(1-GG)\times GW}{\lvert DV \rvert}$

DV: Deviation (%), GW: Gap width (%)=the rate of gap width to deviation, GG: Gap gain

In case of (a)



In case of (b)



(3) Deviation for PID operation (DV') is calculated as below.

Condition	Deviation for PID operation (DV')
DV<-GW	DV'= - (GG × GW)+(DV+GW)
$\lvert DV \rvert \leqq GW$	DV'=GG × DV
DV>GW	DV'=GG × GW+(DV-GW)

DV': Deviation for PID operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as below.

Condition	Deviation (DV)
Direct action (PN=1)	DV (%)=PVP (%)- SV (%)
Reverse action (PN=0)	DV (%)=SV (%)- PVP (%)

DV: Deviation (%), PVP (%): PV input (%), SV (%) = $\dfrac{100}{RH-RL} \times$ (SV-RL), RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PID operation (continued)</td><td>

(5) PID operation is conducted as below.

<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (ΔMV)</td><td colspan="2">

$$\Delta MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{(DVn - DV_{n-1})}_{\text{Proportional}} + \underbrace{\frac{CT}{Ti} \times DVn}_{\text{Integral}} + \underbrace{Bn}_{\text{Derivative(imperfect derivative)}} \}$$

The proportional item, integral item and derivative item of ΔMV are as follows.

- Proportional item : $\Delta MV = Kp \times (DVn - DV_{n-1})$,
- Integral item : $\Delta MV = Kp \times \frac{CT}{Ti} \times DVn$,
- Derivative item : $\Delta MV = Kp \times$ Bn (see below)

</td></tr>
<tr><td>Bn</td><td>

$$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ (PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{Td} \}$$

</td><td>

$$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ -(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{Td} \}$$

</td></tr>
</table>

Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Previous deviation, PVn: Process variable, PV_{n-1}: Previous process variable, PV_{n-2}: Process variable before last, SVn: Engineering value conversion processing result

(a) Integral item and derivative item are listed below corresponding to each condition.

<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>Td=0, or control mode being either of MAN and CMV</td><td>$B_n = 0$</td></tr>
<tr><td>Any of 1), 2), 3)
1) Ti=0
2) When either MH or ML error occurs, MVP>MH and $\frac{CT}{Ti} \times DV_n > 0$
3) When either MH or ML error occurs, MVP<ML and $\frac{CT}{Ti} \times DV_n < 0$</td><td>$\frac{CT}{Ti} \times DV_n = 0$</td></tr>
</table>

Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit, ML: Output low limit,
MVP: MV internal operation value

(b) Control cycle (CT) should be set to be the integral number multiple of execution cycle (△T).

(c) Integral constant should be set to be 0.0 or over control cycle (CT).

(d) PID operation of the tag access FB is executed once in control cycle (CT), (output △MV). For, otherwise execution cycle (△T), the previous output value is kept (△MV=0)

</td></tr>
<tr><td>Engineering value conversion</td><td>

When the control mode is CAS/CSV, the setting value (%) from the primary loop is converted into engineering value.

$$SV = \frac{RH - RL}{100} \times \text{Setting value (\%) from the primary loop} + RL$$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

</td></tr>
<tr><td>Inverse engineering value conversion</td><td>

The setting value (SV) of engineering value is converted to percentage SV (%)

$$SV (\%) = \frac{100}{RH - RL} \times (SV - RL)$$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Tracking processing</td><td>Whether execute tracking processing to input variable CASIN_T.
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SVPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table></td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether enable Alarm Detection or not in deviation check.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DVLA will not be detected.
● ERRI, DVLI
(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
<tr><td>Auto tuning (ATI)</td><td>The proportional gain (Kp), integral time (Ti), derivative time (Td) is automatically calculated by use of the dynamic characteristics by automatic tuning. For details of auto tuning, refer to Appendix 3.1.
(1) The aim of Auto tuning is the initial value setting of proportional gain (Kp), integral time (Ti) and derivative time (Td) of PID control. ZN method (Step response method by Ziegler-Nichols) is being used here.
(2) Auto tuning can only be executed when the control mode is manual (MANUAL).</td></tr>
</table>

Other Functions

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) is TRUE. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Reset DLVA when DLVA of alarm (ALM) occurs. 4) Alarm is not detected in deviation check.

Processing Operation

Processing / Control mode	Deviation check	PID operation	Engineering value conversion	Inverse engineering value conversion	Tracking	Alarm	Auto tuning
MAN, CMV, AUT	○	○	×	○	○ (*1)	○ (*2)	○ (*3)
CAS, CSV	○	○	○	○	×	○ (*2)	×

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

*2 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

*3 Auto tuning can only be executed when the control mode is Manual (MANUAL).

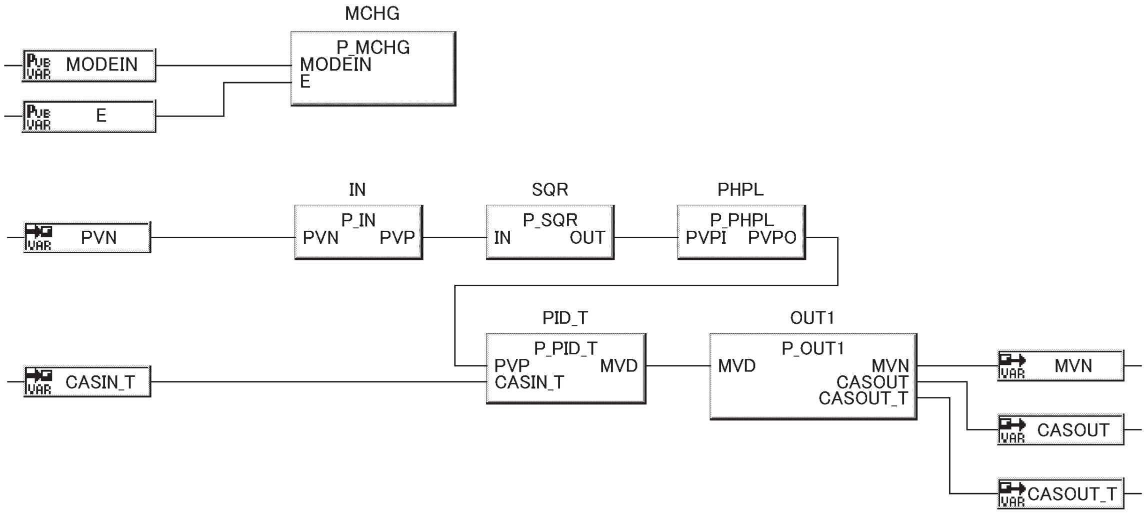
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

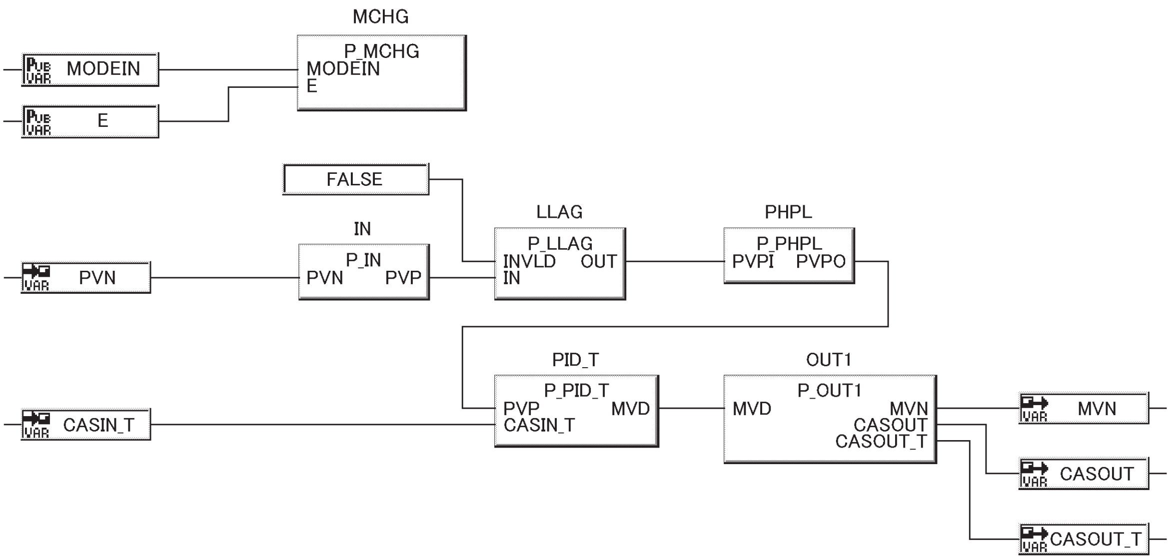
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(1) Example 1

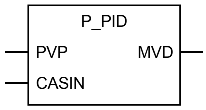


(2) Example 2



POINT
It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.4 Velocity Type PID Control (Without Tracking to primary loop) (P_PID)

FB	FBD parts
P_PID	

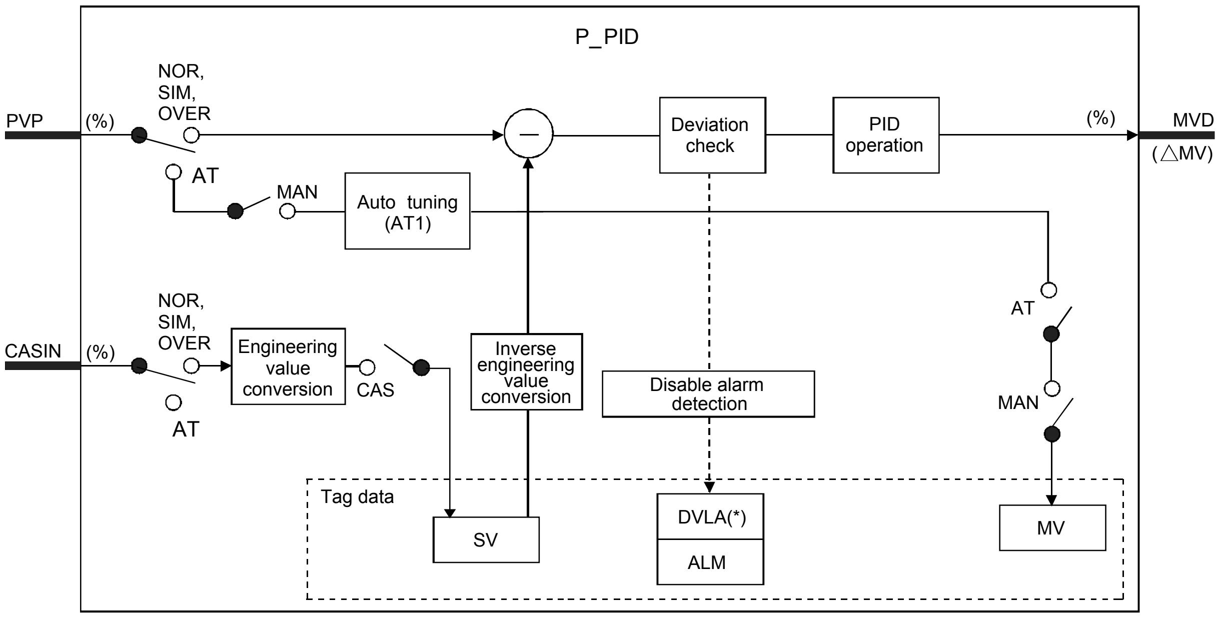
Corresponding tag type
PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Execute PID operation by use of PV-derivative, imperfect derivative and velocity type, and output (△MV).

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVD	Output variable	REAL	△MV output (unit: %)	-999999 to 999999

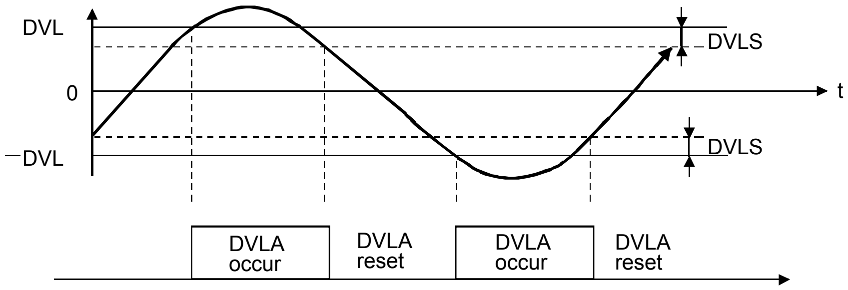
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Direct action and reverse action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Used, FALSE: Not used)	TRUE, FALSE	TRUE	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>Execute deviation check processing.


<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Deviation large (DVLA)</th></tr>
<tr><td>DVL < |DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>
DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)</td></tr>
</table>

Item	Contents
PID operation	

(1) Gain (Kp) is calculated as follows:

$$Kp=K \times P$$

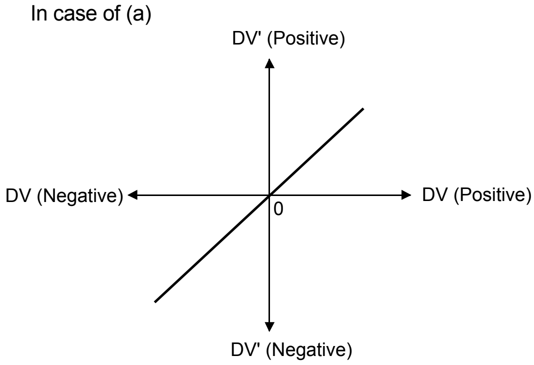
K: Output gain, P: Gain

(2) Output gain (K) is calculated as follows:

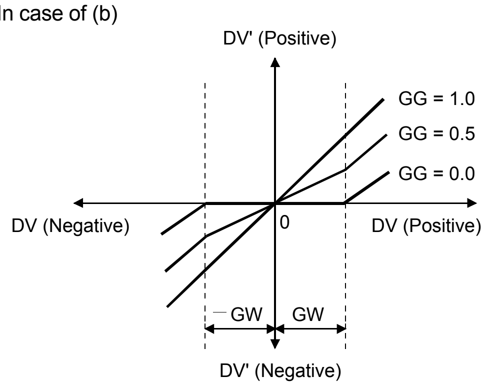
Condition		Output gain (K)
(a) When gap width (GW) = 0, the value of K relative to deviation (DV).		K=1
(b) When gap width (GW) >0, the value of K corresponding to deviation (DV).	$\|DV\| \leqq GW$	K=GG
	$\|DV\| > GW$	$K = 1 - \frac{(1 - GG) \times GM}{\|DV\|}$

DV: Deviation (%), GW: Gap width (%)=Rate of gap width to deviation, GG: Gap gain

In case of (a)



In case of (b)



(3) Deviation for PID operation (DV') is calculated as follows:

Condition	Deviation for PID operation (DV')
DV < -GW	DV'= - (GG × GW)+(DV+GW)
$\|DV\| \leqq GW$	DV'=GG × DV
DV > GW	DV'=GG × GW+(DV-GW)

DV': Deviation for PID operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation value (DV) for direct/reverse action is calculated as follows:

Condition	Deviation (DV)
Direct action (PN=1)	DV (%) = PVP (%) –SV (%)
Reverse action (PN=0)	DV (%) = SV (%) –PVP (%)

DV: Deviation (%), PVP (%): PV input value (%), SV (%) = $\frac{100}{RH-RL} \times$ (SV-RL),
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PID operation (continued)</td><td>
(5) PID operation is calculated as follows:
<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>DVn=PVn-SVn</td><td>DVn=SVn-PVn</td></tr>
<tr><td>Output variation (△MV)</td><td colspan="2">
$$\triangle MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{(DVn - DVn\text{-}1)}_{\text{Proportional}} + \underbrace{\frac{CT}{Ti} \times DVn}_{\text{Integral}} + \underbrace{Bn}_{\text{Derivative (imperfect derivative)}} \}$$
The proportional item, integral item and derivative term of △MV are as follows
● Proportional item : $\triangle MV = Kp \times (DVn\text{-}DVn\text{-}1)$,
● Integral item : $\triangle MV = Kp \times \frac{CT}{Ti} \times DVn$,
● Derivative item : $\triangle MV = Kp \times$ Bn(see below)
</td></tr>
<tr><td>Bn</td><td>$Bn = Bn\text{-}1 + \frac{Md \times Td}{Md \times CT + Td} \times \{(PVn - 2PVn\text{-}1 + PVn\text{-}2) - \frac{CT \times Bn\text{-}1}{Td}\}$</td><td>$Bn = Bn\text{-}1 + \frac{Md \times Td}{Md \times CT + Td} \times \{-(PVn - 2PVn\text{-}1 + PVn\text{-}2) - \frac{CT \times Bn\text{-}1}{Td}\}$</td></tr>
</table>
Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DVn-1: Previous deviation, PVn: Process variable, PVn-1: Previous process variable, PVn-2: Process variable before last, SVn: Engineering value conversion processing result
(a) Integral item and derivative item are listed below corresponding to each condition.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>When Td=0, or control mode is either MAN or CMV</td><td>Bn=0</td></tr>
<tr><td>Any of the following 1), 2), 3)
1) Ti=0
2) When either MH or ML error occurs, MVP>MH and $\frac{CT}{Ti} \times DVn > 0$
3) When either MH or ML error occurs, MVP<ML and $\frac{CT}{Ti} \times DVn < 0$</td><td>$\frac{CT}{Ti} \times DVn = 0$</td></tr>
</table>
Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit value, ML: Output low limit value, MVP: MV internal operation value
(b) Control cycle (CT) should be set as the integral multiple of execution cycle (△T).
(c) Integral constant shall be set to 0.0 or over control cycle (CT).
(d) PID operation of the tag access FB is executed in every control cycle (CT) (△MV output).
For other execution cycle (△T), the previous value shall be applied. (△MV=0)
</td></tr>
<tr><td>Engineering value conversion</td><td>
When the control mode is CAS/ CSV, the setting value (%) from the primary loop is converted to engineering value.
$$SV = \frac{RH - RL}{100} \times \text{Setting value (\%) from the primary loop} + RL$$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Inverse engineering value conversion</td><td>
The setting value (SV) of engineering value is converted to percentage SV (%).
$$SV (\%) = \frac{100}{RH - RL} \times (SV\text{-}RL)$$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Disable Alarm Detection</td><td>
Set whether Enable Alarm Detection or not in deviation check.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of "Disable Alarm Detection"(INH) of tag data are TRUE, alarm (ALM) of DVLA will not be detected.
● ERRI, DVLI
(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the next page.
</td></tr>
</table>

Item	Contents
Auto tuning (AT1)	The proportional gain (Kp), integral time (Ti), derivative time (Td) is automatically calculated by use of the dynamic characteristics through automatic tuning. For details of auto tuning, refer to Appendix 3.1. (1) The aim of Auto tuning is the initial value setting of proportional gain (Kp), integral time (Ti) and derivative time (Td) of PID control. ZN method (Step response method by Ziegler-Nichols) is being used here. (2) Auto tuning can only be executed when the control mode is manual (MANUAL).

Other Function

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) is TRUE. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Reset DVLA when DVLA of alarm (ALM) occurs. 4) Alarm is not detected in deviation check

Processing Operation

Processing / Control mode	Deviation check	PID operation	Engineering value conversion	Inverse engineering value conversion	Alarm	Auto tuning
MAN, CMV, AUT	○	○	×	○	○ (*1)	○ (*2)
CAS, CSV	○	○	○	○	○ (*1)	×

○: Execute ×: Not execute

*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

*2 Auto tuning can only be executed when the control mode is Manual (MANUAL).

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.

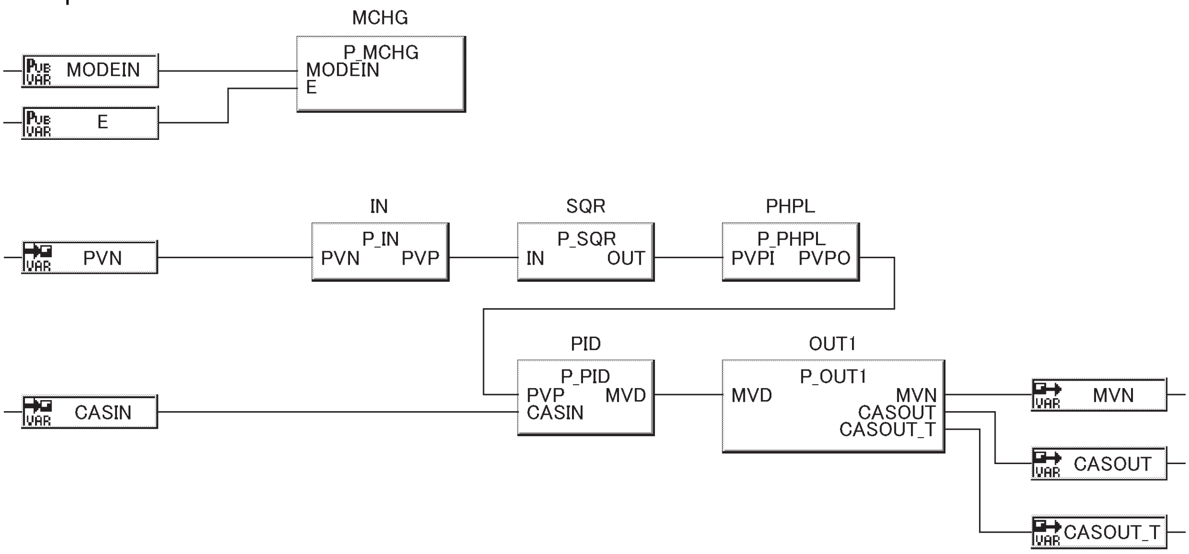
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.

For details of process control error code, refer to Appendix 2.

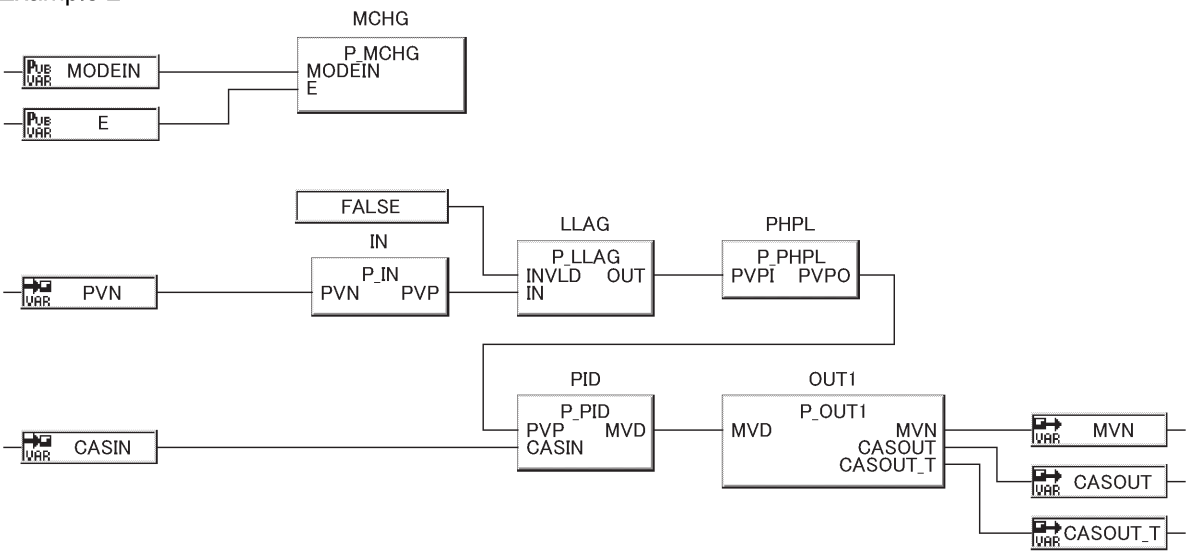
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(1) Example 1



(2) Example 2

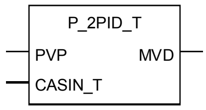


POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in FB property window.

8-56

8.2.5 2-Degree-of-Freedom PID Control (With Tracking to primary loop) (P_2PID_T)

FB	FBD parts
P_2PID_T	

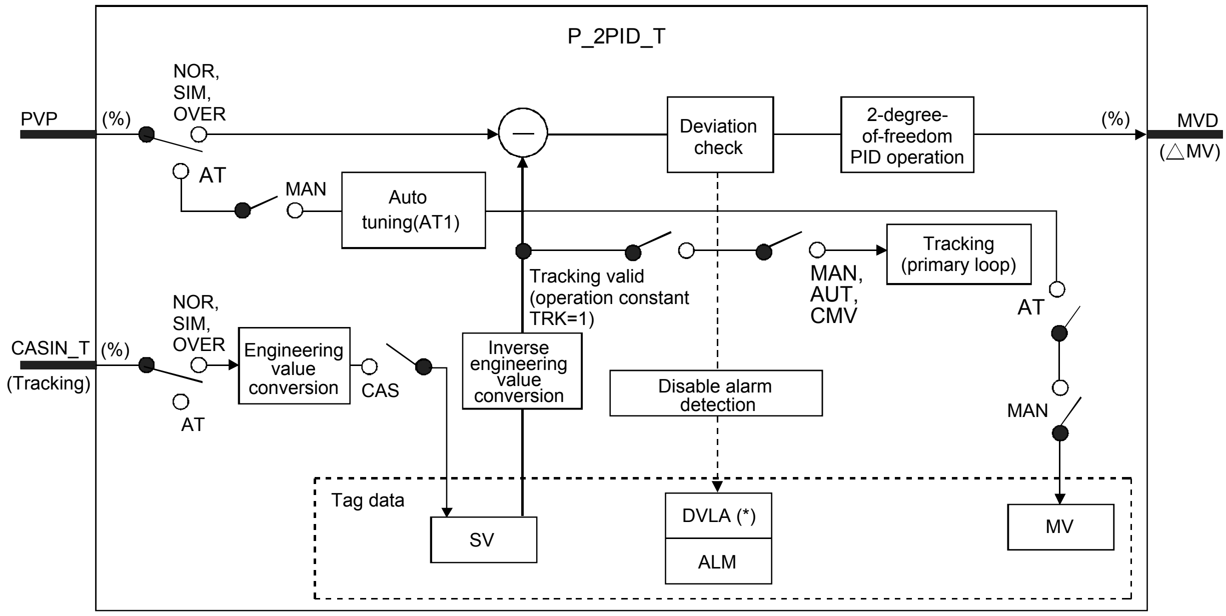
Corresponding tag type
2PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Optimize the response performance (target tracking) for setting value change and disturbance response, and output (△MV).

Function/FB classification: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Content	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	Cascade SV input (unit: %) (With tracking)	0 to 100
Output	MVD	Output variable	REAL	△MV output (unit: %)	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Content	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	TRK(*1)	Public variable	INT	Tracking flag (0: Not executed, 1: executed)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not host MV, FALSE: Host MV)	TRUE, FALSE	TRUE	User

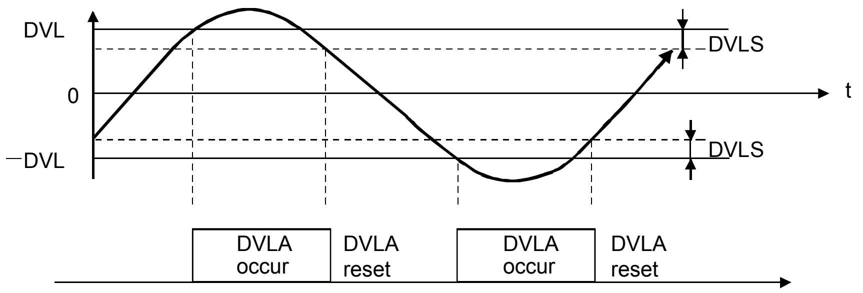
*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
Execute deviation check processing.



<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL < |DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

Item	Contents
2-degree-of-freedom PID operation	(1) Gain (Kp) is calculated as follows:

$$Kp = K \times P$$

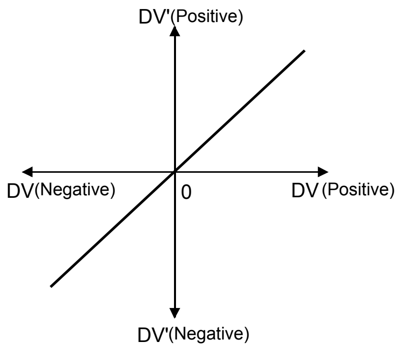
K: Output gain, P: Gain

(2) Output gain (K) is calculated as follows:

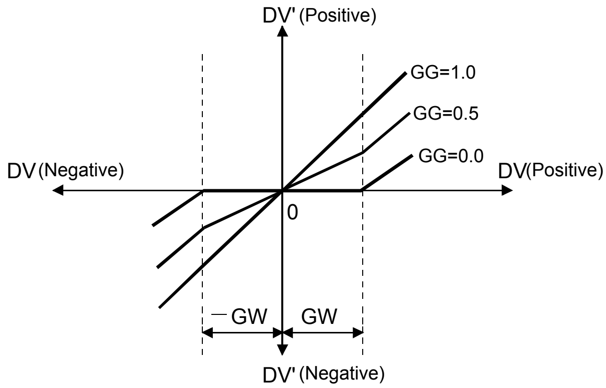
Condition		Output gain (K)
(a) When gap width (GW) = 0, the value of K corresponding to deviation (DV).		K=1
(b) When gap width (GW) > 0, the value of K corresponding to deviation (DV).	$\lvert DV\rvert \leqq GW$	K=GG
	$\lvert DV\rvert > GW$	$K=1-\dfrac{(1-GG)\times GW}{\lvert DV\rvert}$

DV: Deviation (%), GW: Gap width (%)= the rate of gap width corresponding to deviation, GG: Gap gain

In the case of (a)



In the case of (b)



(3) Deviation for PID operation (DV') is calculated as follows.

Condition	Deviation for PID operation (DV')
DV < – GW	DV' = – (GG × GW) + (DV+GW)
$\lvert DV\rvert \leqq GW$	DV' = GG × DV
DV > GW	DV' = GG × GW + (DV–GW)

DV': Deviation for PID operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as follows.

Condition	Deviation (DV)
Direct action (PN=1)	DV (%) = PVP (%) –SV (%)
Reverse action (PN=0)	DV (%) = SV (%) –PVP (%)

DV: Deviation (%), PVP (%): PV input value (%), SV (%) = $\dfrac{100}{RH-RL} \times (SV-RL)$,

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>2-degree-of-freedom PID operation (continued)</td><td>

(5) 2-degree-of-freedom PID operation is executed as follows.

<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (△MV)</td><td colspan="2">$$\triangle MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{(1-\alpha) \times (DV_n - DV_{n-1})}_{\text{Proportional}} + \underbrace{\frac{CT}{Ti} \times DV_n}_{\text{Integral}} + \underbrace{(1-\beta) \times B_n}_{\text{Derivative}} + \underbrace{\alpha \times C_n + \beta \times D_n}_{\text{Feed forward compensation}} \}$$</td></tr>
<tr><td>Bn</td><td colspan="2">$$B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ (DV_n - 2DV_{n-1} + DV_{n-2}) - \frac{CT \times B_{n-1}}{Td} \}$$</td></tr>
<tr><td>Dn</td><td>$$D_n = D_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ (PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{Td} \}$$</td><td>$$D_n = D_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ - (PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{Td} \}$$</td></tr>
<tr><td>Cn</td><td>$PV_n - PV_{n-1}$</td><td>$-(PV_n - PV_{n-1})$</td></tr>
</table>

Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Last deviation, DV_{n-2}: Deviation before last, PVn: Process variable, PV_{n-1}: Previous process variable, PV_{n-2}: Process variable before last, SVn: The processing result of engineering value conversion, α: 2-degree-of-freedom parameter (feedforward proportional), β: 2-degree-of-freedom parameter (feed forward derivative)

(a) Integral items and derivative items are listed below corresponding to each condition.

<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>When Td=0, or control mode is either MAN or CMV</td><td>Bn=0</td></tr>
<tr><td>Any of following 1), 2), 3)
1) Ti=0
2) When either MH or ML error occurs, MVP>MH and $\frac{CT}{Ti} \times DV_n > 0$
3) When either MH or ML error occurs, MVP<ML and $\frac{CT}{Ti} \times DV_n < 0$</td><td>$\frac{CT}{Ti} \times DV_n = 0$</td></tr>
</table>

Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit value, ML: Output low limit value, MVP: MV internal operation value

(b) Control cycle (CT) shall be set to be the integral multiple of execution cycle (△T).

(c) Integral constant should be set to be 0.0 or over control cycle (CT).

(d) PID operation of the tag access FB is executed in every control cycle (CT) (△MV output).
For other execution cycle (△T), hold the previous value. (△MV=0)

</td></tr>
<tr><td>Engineering value conversion</td><td>

Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.

$$SV = \frac{RH - RL}{100} \times \text{Setting value (\%) from the primary loop} + RL$$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

</td></tr>
<tr><td>Inverse engineering value conversion</td><td>

Convert the setting value (SV) of engineering value to percentage SV (%).

$$SV (\%) = \frac{100}{RH - RL} \times (SV - RL)$$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Tracking processing</td><td>Following indicates whether execute tracking processing to input variable CASIN_T.
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SVPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table></td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether Enable Alarm Detection or not in deviation check.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DVLA will not be detected.
● ERRI, DVLI
(2) "Disable Alarm Detection" by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
<tr><td>Auto tuning (ATI)</td><td>The proportional gain (Kp), integral time (Ti), derivative time (Td) is automatically calculated by use of the dynamic characteristics by automatic tuning. For details of auto tuning, refer to Appendix 3.1.
(1) The aim of Auto tuning is the initial value setting of proportional gain (Kp), integral time (Ti) and derivative time (Td) of PID control. ZN method (Step response method by Ziegler-Nichols) is being used here.
(2) Auto tuning can only be executed when the control mode is manual (MANUAL).</td></tr>
</table>

Other Function

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) is TRUE. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Reset DLVA when DLVA of alarm (ALM) occurs. 4) Alarm is not detected in deviation check.

Processing Operation

Processing / Control mode	Deviation check	2-degree-of-freedom PID operation	Engineering value conversion	Inverse engineering value conversion	Tracking	Alarm	Auto tuning
MAN, CMV, AUT	○	○	×	○	○ (*1)	○ (*2)	○ (*3)
CAS, CSV	○	○	○	○	×	○ (*2)	×

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

*2 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

*3 Auto tuning can only be executed when the control mode is Manual (MANUAL).

Error

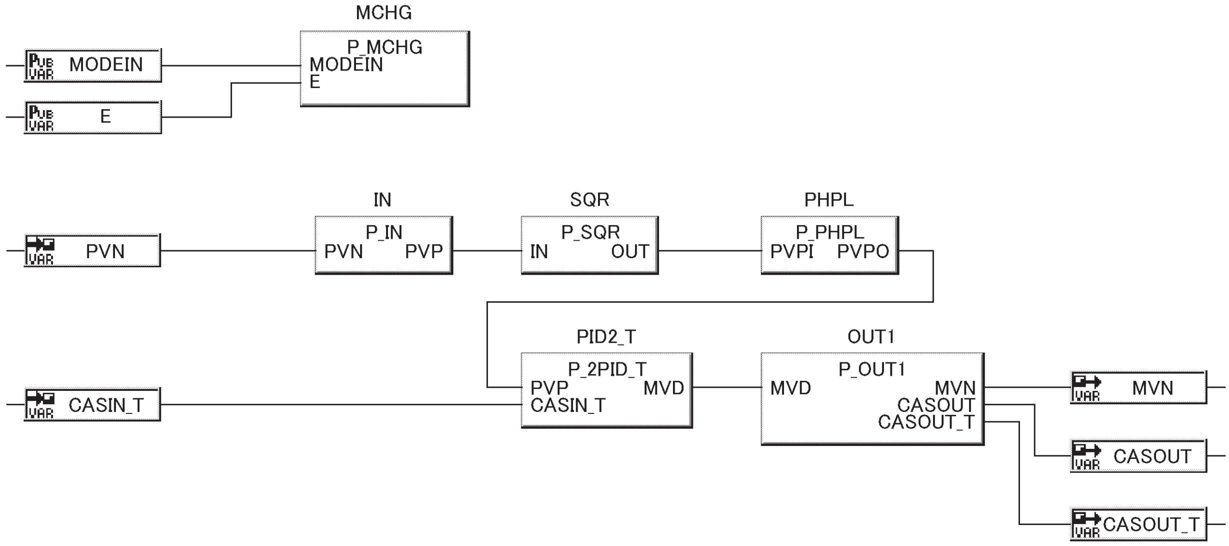
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

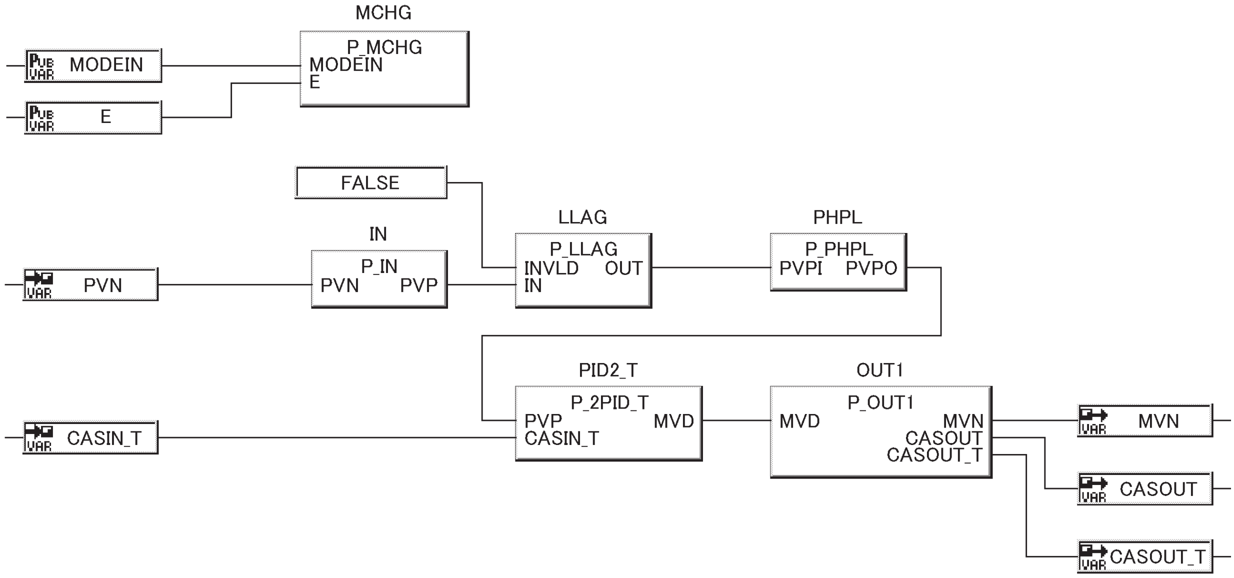
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(1) Example 1



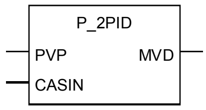
(2) Example 2



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.6 2-Degree-of-Freedom PID Control (Without Tracking to primary loop) (P_2PID)

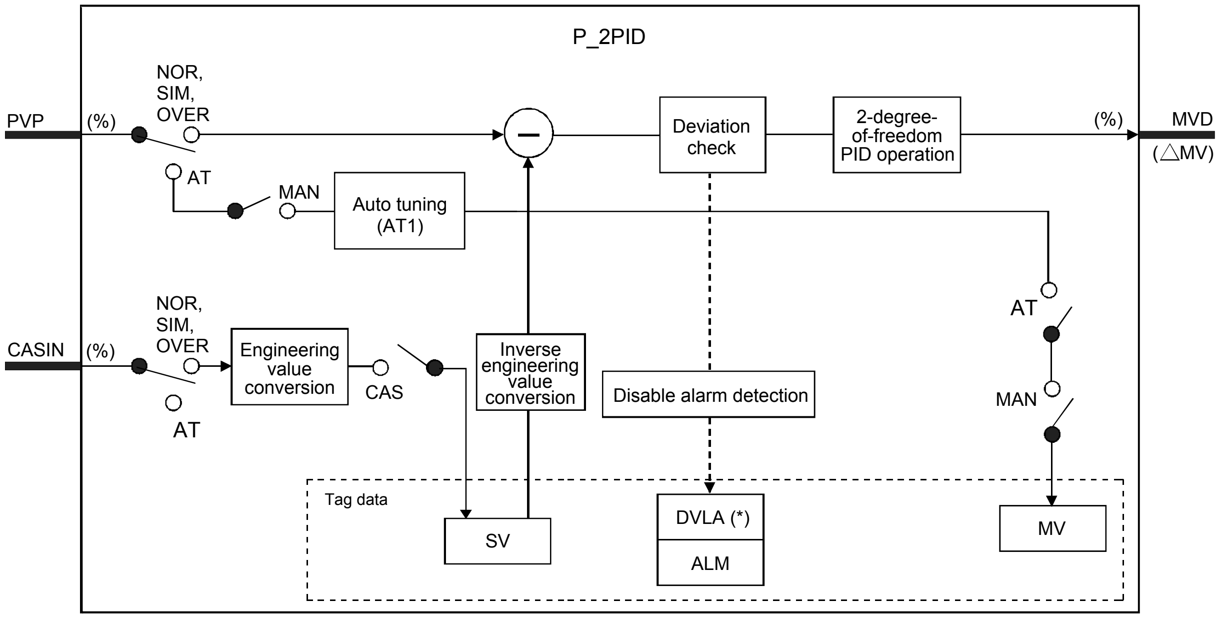
FB	FBD parts
P_2PID	

Corresponding tag type				
2PID				
Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Optimize the response performance (target tracking) for setting value change and disturbance response, and output (△MV).

Function/FB classification: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Content	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVD	Output variable	REAL	∆MV output (unit: %)	-999999 to 999999

Public Variable (Operation constant)

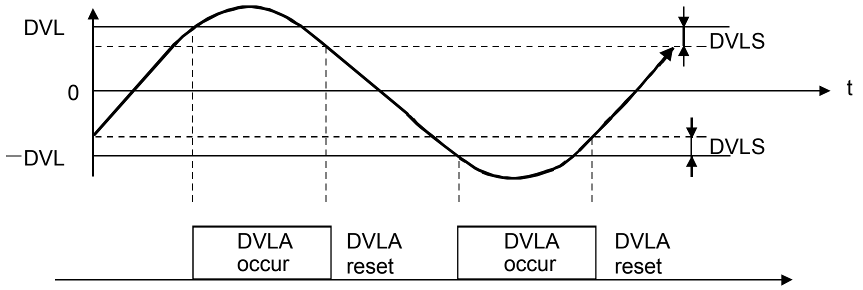
	Variable name	Variable type	Data type	Content	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
Execute deviation check processing.



<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL < |DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

Item	Contents
2-degree-of-freedom PID operation	(see below)

(1) Gain (Kp) is calculated as follows:

$$Kp = K \times P$$

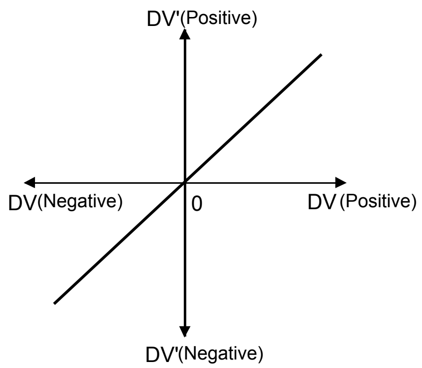
K: Output gain, P: Gain

(2) Output gain (K) is calculated as follows:

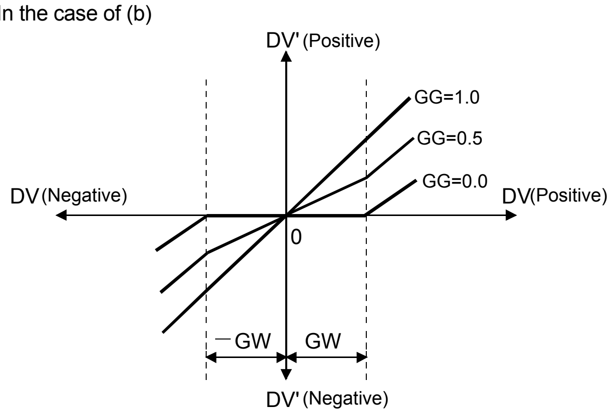
Condition		Output gain (K)
(a) When gap width (GW) = 0, the value of K corresponding to deviation (DV).		K=1
(b) When gap width (GW) > 0, the value of K corresponding to deviation (DV).	$\lvert DV \rvert \leqq GW$	K=GG
	$\lvert DV \rvert > GW$	$K = 1 - \dfrac{(1-GG) \times GW}{\lvert DV \rvert}$

DV: Deviation (%), GW: Gap width (%)= the rate of gap width corresponding to deviation, GG: Gap gain

In the case of (a)



In the case of (b)



(3) Deviation for PID operation (DV') is calculated as follows.

Condition	Deviation for PID operation (DV')
DV < – GW	DV' = – (GG × GW) + (DV+GW)
$\lvert DV \rvert \leqq GW$	DV' = GG × DV
DV > GW	DV' = GG × GW + (DV – GW)

DV': Deviation for PID operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as follows.

Condition	Deviation (DV)
Direct action (PN=1)	DV (%) = PVP (%) – SV (%)
Reverse action (PN=0)	DV (%) = SV (%) – PVP (%)

DV: Deviation (%), PVP (%): PV input value (%), SV (%) = $\dfrac{100}{RH-RL} \times (SV-RL)$,

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>2-degree-of-freedom IPD operation (continued)</td><td>
(5) 2-degree-of-freedom IPD operation is conducted as follows.
<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (△MV)</td><td colspan="2">$\triangle MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{(1-\alpha) \times (DV_n - DV_{n-1})}_{\text{Proportional}} + \underbrace{\frac{CT}{Ti} \times DV_n}_{\text{Integral}}$
$+ \underbrace{(1-\beta) \times B_n}_{\text{Derivative}} + \underbrace{\alpha \times C_n + \beta \times D_n}_{\text{Feed forward compensation}} \}$</td></tr>
<tr><td>Bn</td><td colspan="2">$B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ (DV_n - 2DV_{n-1} + DV_{n-2}) - \frac{CT \times B_{n-1}}{Td} \}$</td></tr>
<tr><td>Dn</td><td>$D_n = D_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times$
$\{ (PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{Td} \}$</td><td>$D_n = D_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times$
$\{ - (PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{Td} \}$</td></tr>
<tr><td>Cn</td><td>$PV_n - PV_{n-1}$</td><td>$-(PV_n - PV_{n-1})$</td></tr>
</table>
Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Previous deviation, DV_{n-2}: Deviation before last, PVn: Process variable, PV_{n-1}: Previous process variable, PV_{n-2}: Process variable before last, SVn: The processing result of engineering value conversion, α: 2-degree-of-freedom parameter (feedforward proportional), β: 2-degree-of-freedom parameter (feedforward derivative)

(a) Integral items and derivative items are listed below corresponding to each condition.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>When Td=0, or control mode is either MAN or CMV</td><td>Bn=0</td></tr>
<tr><td>Any of following 1), 2), 3)
1) Ti=0
2) When either MH or ML error occurs, MVP>MH and $\frac{CT}{Ti} \times DV_n > 0$
3) When either MH or ML error occurs, MVP<ML and $\frac{CT}{Ti} \times DV_n < 0$</td><td>$\frac{CT}{Ti} \times DV_n = 0$</td></tr>
</table>
Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit value, ML: Output low limit value, MVP: MV internal operation value

(b) Control time (CT) should be set to be the integral multiple of execution cycle (△T).

(c) Integral constant should be set to be 0.0 or over control cycle (CT).

(d) PID operation of the tag access FB is executed in every control cycle (CT) (△MV output).
For other execution cycle (△T), the previous value shall be applied. (△MV=0)
</td></tr>
<tr><td>Engineering value conversion</td><td>Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.

$SV = \frac{RH-RL}{100} \times$ Setting value (%) from the primary loop + RL

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value</td></tr>
<tr><td>Inverse engineering value conversion</td><td>Convert the setting value (SV) of engineering value to percentage SV (%).

SV (%) = 100/(RH-RL) $\times$ (SV – RL)

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value, MV: Manipulated variable</td></tr>
</table>

Item	Contents
Disable Alarm Detection	Set whether to Enable Alarm Detection or not in deviation check. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DVLA will not be detected. ● ERRI, DVLI (2) Disable Alarm Detection by loop stop processing: Please refer to loop stop processing in the following contents.
Auto tuning (ATI)	The proportional gain (Kp), integral time (Ti), derivative time (Td) is automatically calculated by use of the dynamic characteristics by automatic tuning. For details of auto tuning, refer to Appendix 3.1. (1) The aim of Auto tuning is the initial value setting of proportional gain (Kp), integral time (Ti) and derivative time (Td) of PID control. ZN method (Step response method by Ziegler-Nichols) is being used here. (2) Auto tuning can only be executed when the control mode is manual (MANUAL).

Other Function

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) is TRUE. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Reset DLVA when DLVA of alarm (ALM) occurs. 4) Alarm is not detected in deviation check.

Processing Operation

Processing / Control mode	Deviation check	2-degree-of-freedom PID operation	Engineering value conversion	Inverse engineering value conversion	Alarm	Auto tuning
MAN, CMV, AUT	○	○	×	○	○ (*1)	○ (*2)
CAS, CSV	○	○	○	○	○ (*1)	×

○: Execute ×: Not execute

*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

*2 Auto tuning can only be executed when the control mode is Manual (MANUAL).

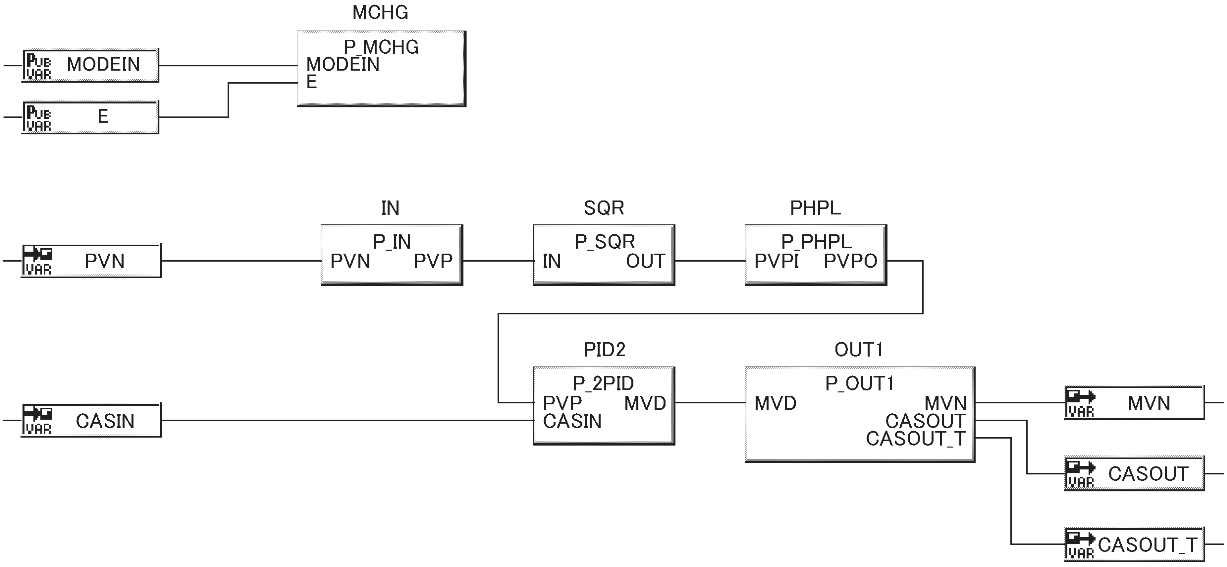
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

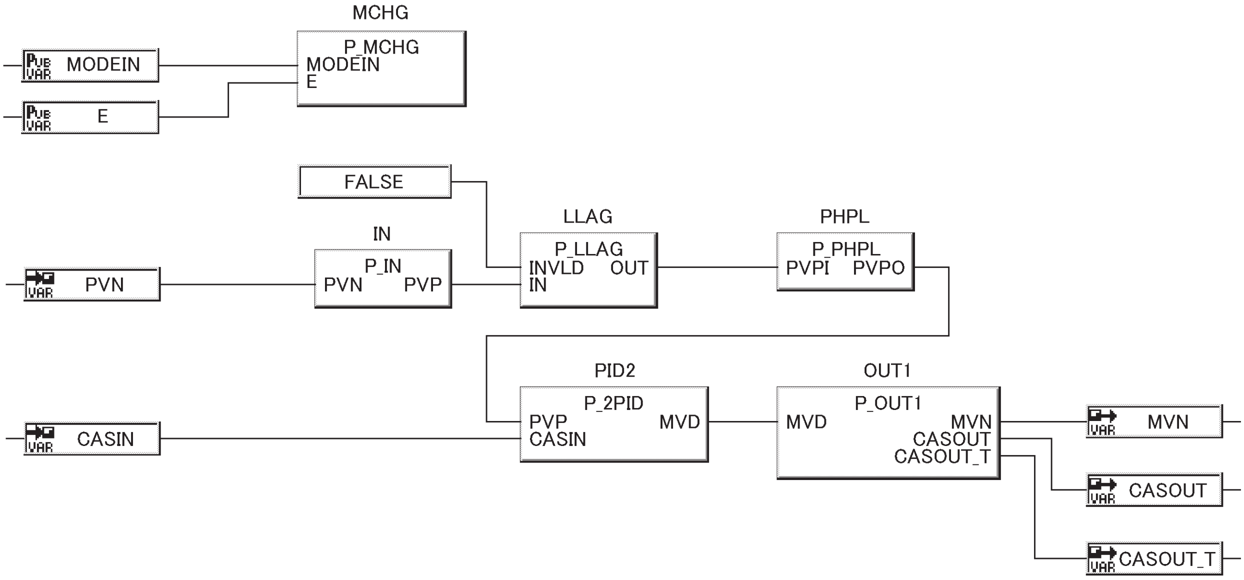
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(1) Example 1



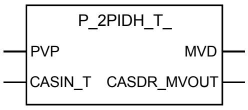
(2) Example 2



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.7 2-Degree-of-Freedom Advanced PID Control (With Tracking to primary loop) (P_2PIDH_T_)

FB	FBD parts
P_2PIDH_T_	

Corresponding tag type
2PIDH

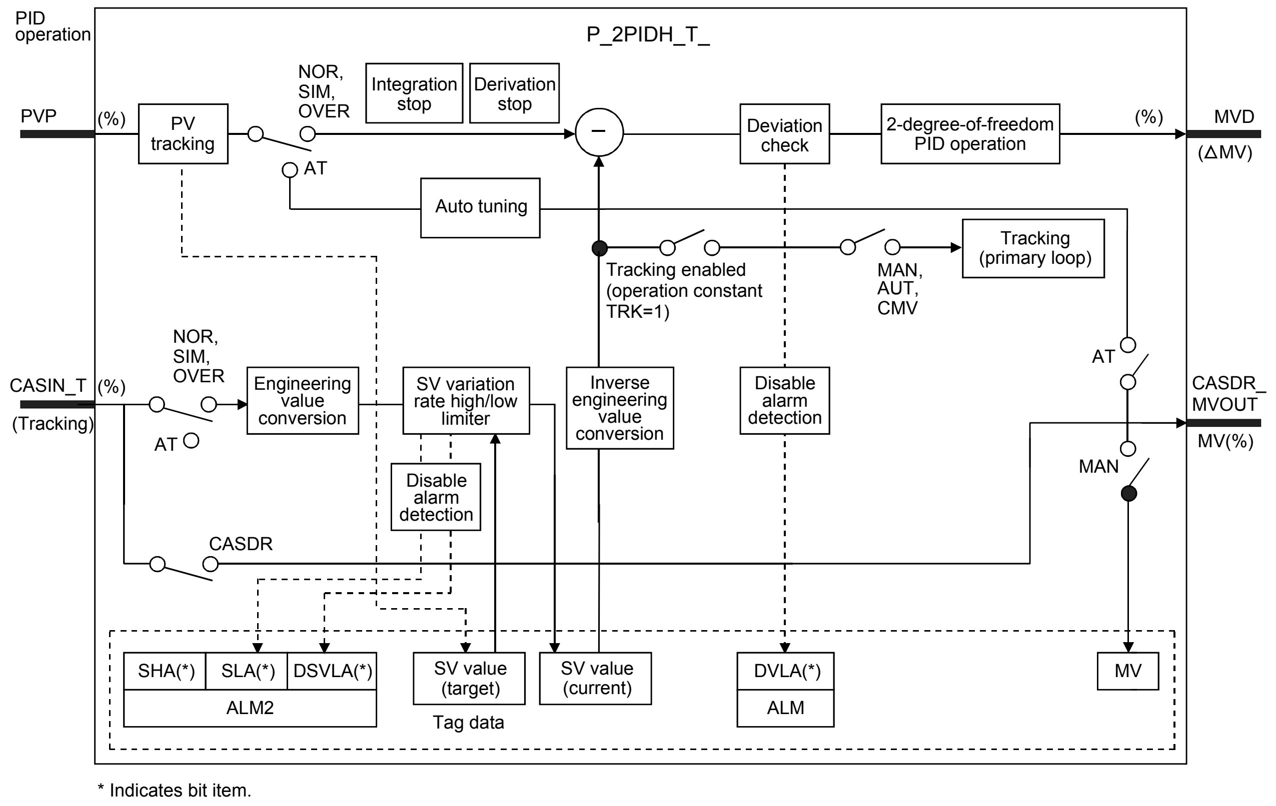
Control mode				
MAN	AUT	CAS*1	CMV	CSV
○	○	○	○	○

*1 Transition to CASDR is possible.

Functions overview:	Optimizes the response performance (target tracking) for setting value change and disturbance response, and output (∆MV). Executes 2-degree-of-freedom PID Operation, PV tracking, integration stop, derivation stop, SV variation rate and high/low limiter processing.

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	Cascade SV input (unit: %) (With tracking)	0 to 100
Output	MVD	Output variable	REAL	∆MV output (unit: %)	-999999 to 999999
	CASDR_MVOUT	Output variable	REAL	MV output for cascade direct (Unit: %)	0 to 100

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Content	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	TRK(*1)	Public variable	INT	Tracking flag (0: Not executed, 1: executed)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not host MV, FALSE: Host MV)	TRUE, FALSE	TRUE	User
	PVTRK_EN	Public variable	BOOL	PV tracking execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	ISTP	Public variable	BOOL	Integration stop signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	DSTP	Public variable	BOOL	Derivation stop signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	LMT_ISTP	Public variable	BOOL	Integration stop selection when MV value variation rate limiter alarm occurs (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User
	SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

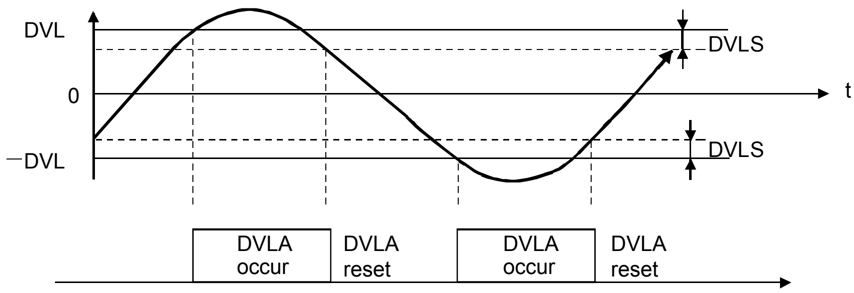
*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
Execute deviation check processing.



<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL < |DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>2-degree-of-freedom PID operation</td><td>

(1) Gain (Kp) is calculated as follows:

$$Kp = K \times P$$

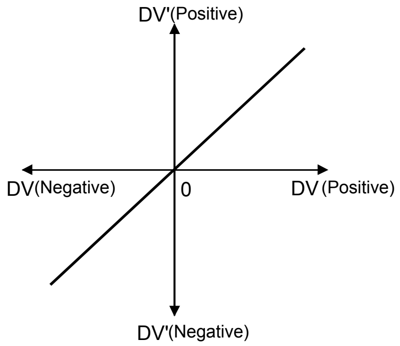
K: Output gain, P: Gain

(2) Output gain (K) is calculated as follows:

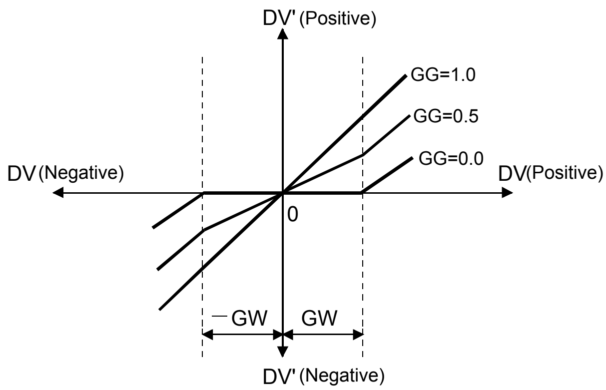
<table>
<tr><th colspan="2">Condition</th><th>Output gain (K)</th></tr>
<tr><td colspan="2">(a) When gap width (GW) = 0, the value of K corresponding to deviation (DV).</td><td>K=1</td></tr>
<tr><td rowspan="2">(b) When gap width (GW) > 0, the value of K corresponding to deviation (DV).</td><td>$|DV| \leqq GW$</td><td>K=GG</td></tr>
<tr><td>$|DV| > GW$</td><td>$K = 1 - \frac{(1-GG) \times GW}{|DV|}$</td></tr>
</table>

DV: Deviation (%), GW: Gap width (%)= the rate of gap width corresponding to deviation, GG: Gap gain

In the case of (a)



In the case of (b)



(3) Deviation for PID operation (DV') is calculated as follows.

<table>
<tr><th>Condition</th><th>Deviation for PID operation (DV')</th></tr>
<tr><td>DV < – GW</td><td>DV' = – (GG × GW) + (DV+GW)</td></tr>
<tr><td>|DV| ≦ GW</td><td>DV' = GG × DV</td></tr>
<tr><td>DV > GW</td><td>DV' = GG × GW+ (DV–GW)</td></tr>
</table>

DV': Deviation for PID operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as follows.

<table>
<tr><th>Condition</th><th>Deviation (DV)</th></tr>
<tr><td>Direct action (PN=1)</td><td>DV (%) = PVP (%) –SVC (%)</td></tr>
<tr><td>Reverse action (PN=0)</td><td>DV (%) = SVC (%) –PVP (%)</td></tr>
</table>

DV: Deviation (%), PVP (%): PV input value (%), SVC (%) = $\frac{100}{RH-RL} \times$ (SVC-RL),

RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting Value (Current)

</td></tr>
</table>

Item	Contents
2-degree-of-freedom PID operation (continued)	(5) 2-degree-of-freedom PID operation is executed as follows.

	Direct action	Reverse action
Deviation (DVn)	$DV_n = PV_n - SV_n$	$DV_n = SV_n - PV_n$
Output variation (△MV)	$\triangle MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{(1-\alpha)\times(DV_n - DV_{n-1})}_{\text{Proportional}} + \underbrace{\frac{CT}{Ti}\times DV_n}_{\text{Integral}} + \underbrace{(1-\beta)\times B_n}_{\text{Derivative}} + \underbrace{\alpha\times C_n + \beta\times D_n}_{\text{Feed forward compensation}} \}$	
Bn	$B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ (DV_n - 2DV_{n-1} + DV_{n-2}) - \frac{CT \times B_{n-1}}{Td} \}$	
Dn	$D_n = D_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ (PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{Td} \}$	$D_n = D_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ -(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{Td} \}$
Cn	$PV_n - PV_{n-1}$	$-(PV_n - PV_{n-1})$

Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Last deviation, DV_{n-2}: Deviation before last, PVn: Process variable, PV_{n-1}: Previous process variable, PV_{n-2}: Process variable before last, SVn: The processing result of engineering value conversion, α: 2-degree-of-freedom parameter (feedforward proportional), β: 2-degree-of-freedom parameter (feed forward derivative)

(a) Integral items and derivative items are listed below corresponding to each condition.

Condition	Processing
When Td=0, or control mode is either MAN or CMV	Bn=0
Any of following 1), 2), 3) 1) Ti=0 2) When either MH or ML error occurs, MVP>MH and $\frac{CT}{Ti} \times DV_n > 0$ 3) When either MH or ML error occurs, MVP<ML and $\frac{CT}{Ti} \times DV_n < 0$	$\frac{CT}{Ti} \times DV_n = 0$

Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit value, ML: Output low limit value, MVP: MV internal operation value

(b) Control cycle (CT) shall be set to be the integral multiple of execution cycle (△T).

(c) Integral constant should be set to be 0.0 or over control cycle (CT).

(d) PID operation of the tag access FB is executed in every control cycle (CT) (△MV output).
For other execution cycle (△T), hold the previous value (△MV=0).

Item	Contents
Engineering value conversion	Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value. $SV = \frac{RH - RL}{100} \times$ Setting value (%) from the primary loop + RL RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value (Target)
Inverse engineering value conversion	Convert the setting value (SVC) of engineering value to percentage SVC (%). $SVC (\%) = \frac{100}{RH - RL} \times (SVC - RL)$ RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting value (Current)

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Tracking processing</td><td>Following indicates whether execute tracking processing to input variable CASIN_T.
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SVPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table></td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether enable Alarm (ALM) detection or not in deviation check, SV variation rate high/low limiter
(1) Disable alarm detection processing with the tag data setting of "Disable Alarm Detection" (INH) and "Disable Alarm2 Detection" (INH2)
If the following bit items of disable alarm detection (INH) and disable alarm2 detection (INH2) of tag data are TRUE, DVLA of alarm (ALM) and DSVLA of alarm2 (ALM2), SVHA and SVLA will not be detected.
• ERRI, DVLI, DSVLI, SVHI, SVLI
(2) Disable alarm detection by loop stop processing:
Refer to loop stop processing in this section.
(3) Disable alarm detection when the control mode is CASDR
Alarm detection will not be executed.</td></tr>
<tr><td>Auto tuning (AT1, AT2)</td><td>Dynamic characteristics is detected and proportional gain (Kp), integral time (Ti), and derivative time (Td) are automatically calculated using auto tuning.
Select either the Step Response method or the Limit Cycle method for auto tuning.
(1) AT1 (Step Response method)
This method calculates proportional gain (Kp), integral time (Ti), and derivative time (Td) for PID with ZN method (Step Response method by Ziegler-Nichols) and sets their initial values.
Executable control modes are MAN and CMV.
For details, refer to Appendix 3.1.1.
(2) AT2 (Limit Cycle method)
This method calculates proportional gain (Kp), integral time (Ti), and derivative time (Td) for PID with oscillation amplitude and oscillation period by repeatedly operating MV at two positions and generating process variable cycle operation.
Executable control modes are MAN, AUT, CAS, CMV, and CSV.
For details, refer to Appendix 3.1.2.</td></tr>
<tr><td>PV tracking function</td><td>To avoid the sudden MV change in mode switching (MAN → AUTO), matches SV value (target) with PV value when control mode is MAN or CMV and maintains the accordance
<table>
<tr><th>Condition</th><th>PV tracking function</th></tr>
<tr><td>PVTRK_EN = TRUE and control mode = "MAN (CMV)"</td><td>SV value (target) = PV value
SV value (current) = PV value</td></tr>
<tr><td>PVTRK_EN = FALSE or control mode ≠ "MAN (CMV)"</td><td>No processing</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>SV variation rate high/low limiter</td><td>
Checks variation rate high/low limiter to SV value (target value) in every control cycle (CT).
(1) Variation rate limiter
• The control mode is AUT or CAS or CSV.
SV variation rate high limit value inputted in % is converted to engineering value and the processing will be executed.
DSVL → DSVLT (DSVL: SV variation rate high limit value, DSVLT: value converted to engineering value from SV variation rate high limit value)
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>| SV - SVC | ≦ DSVLT</td><td>SV</td><td>FALSE (reset)</td></tr>
<tr><td>SV - SVC > DSVLT</td><td>SVC + DSVLT</td><td>TRUE (occur)</td></tr>
<tr><td>SV - SVC < - DSVLT</td><td>SVC - DSVLT</td><td>TRUE (occur)</td></tr>
</table>
SV: Setting value (target), SVC: Setting value (current)
If DSVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, DSVLA will be FALSE.
• The control mode is MAN or CMV or CASDR.
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>No</td><td>SV</td><td>FALSE (reset)</td></tr>
</table>
(2) High/low limiter
• The control mode is MAN, AUT, CAS, CMV and SVLMT_EN is TRUE.
<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter result</th><th colspan="2">Alarm2 (ALM2)</th></tr>
<tr><th>Target lower limit (SVLA)</th><th>Target upper limit (SVHA)</th></tr>
<tr><td>Variation rate limiter result > SH</td><td>SH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter result < SL</td><td>SL</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>SL ≦ variation rate limiter result ≦ SH</td><td>Variation rate limiter result</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
If SVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVLA will be FALSE.
If SVHI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVHA will be FALSE.
High/low limiter result is stored to SVC (setting value (current)).
• The control mode is CASDR or SVLMT_EN is FALSE.
Variation rate limiter result is stored to SVC (setting value (current)).
</td></tr>
<tr><td>Integration stop</td><td>
Stops the operation of integral element.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>ISTP = TRUE</td><td>Stops the operation of integral element.</td></tr>
<tr><td>ISTP = FALSE</td><td>No processing</td></tr>
</table>
</td></tr>
<tr><td>Derivation stop</td><td>
Stops the operation of differential element.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>DSTP = TRUE</td><td>Stops the operation of differential element.</td></tr>
<tr><td>DSTP = FALSE</td><td>No processing</td></tr>
</table>
</td></tr>
<tr><td>Stop integration in MV variation rate limiter occurrence</td><td>
Stops the operation of integral element when MV variation rate limiter occurs.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>LMT_ISTP = TRUE and DMLA alarm occurs</td><td>Stops the operation of integral element when MV variation rate limiter occurs and the integral component is the same with the limit direction.</td></tr>
<tr><td>LMT_ISTP = FALSE</td><td>No processing</td></tr>
</table>
</td></tr>
</table>

Other Function

Item	Contents
Loop stop processing	The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE. 1) Set $\triangle$MV to 0. 2) Change the control mode automatically to MANUAL. 3) Resets DVLA when the DVLA of alarm (ALM) occurs. Reset DSVLA, SVLA, and SVHA when the DSVLA, the SVLA, and the SVHA of alarm2 (ALM2) occur. 4) Alarm is not detected in deviation check, SV variation rate high/low limiter.

Processing Operation

Processing / Control mode	Deviation check	2-degree-of-freedom PID operation	Engineering value conversion	Inverse engineering value conversion	Alarm	Auto tuning (AT1)	Auto tuning (AT2)	PV tracking	SV variation rate high/low limiter	Integration stop	Derivation stop
MAN, CMV	○	○	×	○	○ (*1)	○	○	○	○ (*2, *3)	○	○
AUT	○	○	×	○	○ (*1)	×	○	×	○ (*3)	○	○
CAS, CSV	○	○	○	○	○ (*1)	×	○	×	○ (*3)	○	○
CASDR	×	○	○	○	×	×	×	×	×	○	○

○: Execute ×: Not execute

*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

*2 When the control mode is MAN, SV variation rate limiter processing is not executed.

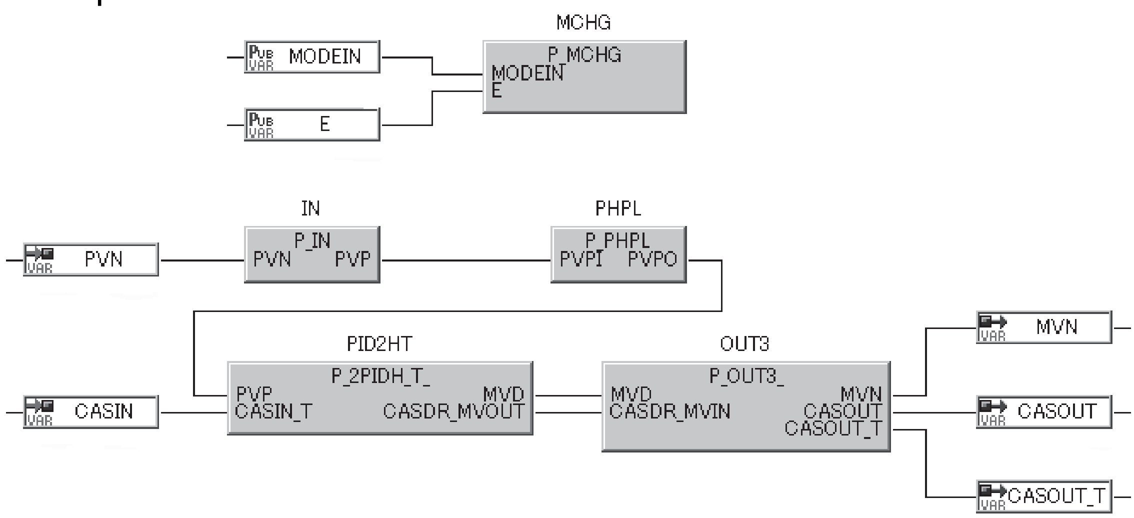
*3 When sensor error (SEA) occurs and "Hold the output" is selected, processing is not executed. SVC (setting value (current)) is not updated as well.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

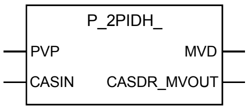
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.8 2-Degree-of-Freedom Advanced PID Control (Without Tracking to primary loop) (P_2PIDH_)

FB	FBD parts
P_2PIDH_	

Corresponding tag type
2PIDH

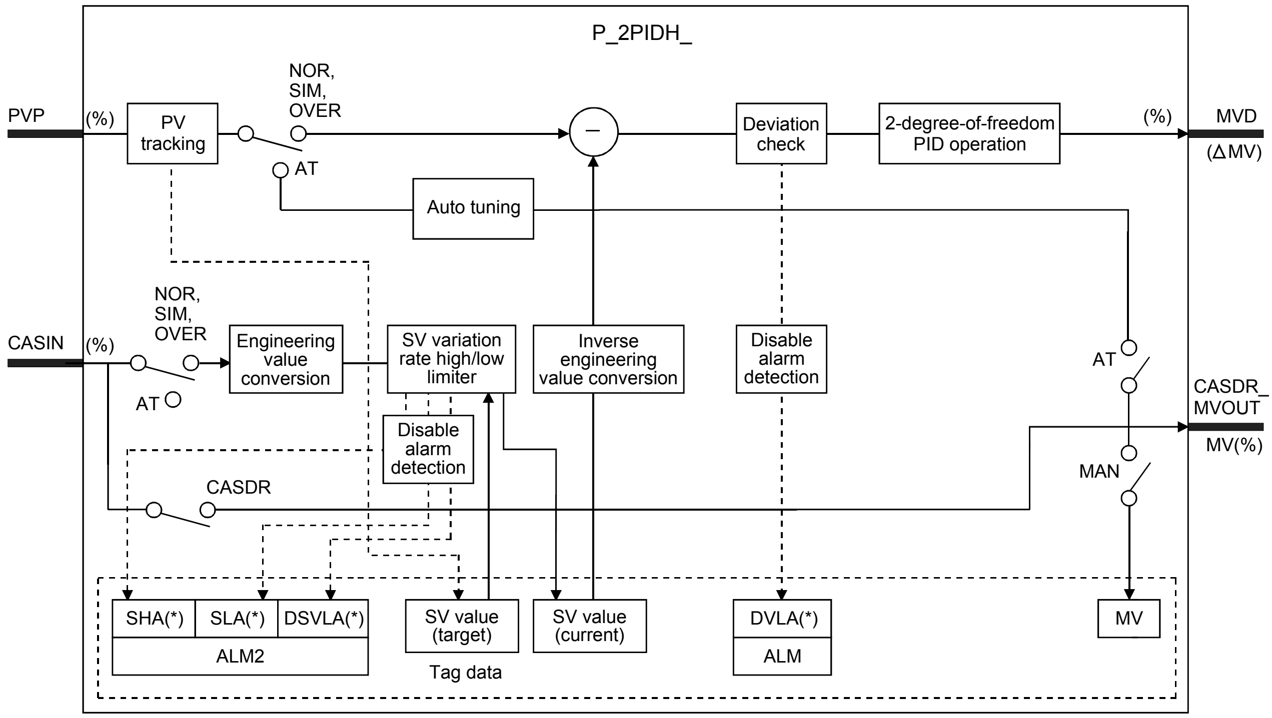
Control mode				
MAN	AUT	CAS[*1]	CMV	CSV
○	○	○	○	○

*1 Transition to CASDR is possible.

Functions overview: Optimizes the response performance (target tracking) for setting value change and disturbance response, and output (∆MV). Executes 2-degree-of-freedom PID Operation, PV tracking, integration stop, derivation stop, SV variation rate and high/low limiter processing.

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



*Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %) (tracking)	0 to 100
Output	MVD	Output variable	REAL	∆MV output (unit: %)	-999999 to 999999
	CASDR_MVOUT	Output variable	REAL	MV output for cascade direct (Unit: %)	0 to 100

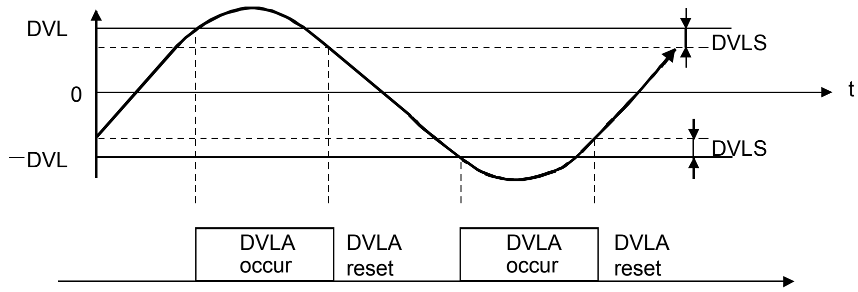
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Content	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PVTRK_EN	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not host MV, FALSE: Host MV)	TRUE, FALSE	FALSE	User
	ISTP	Public variable	BOOL	Integration stop signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	DSTP	Public variable	BOOL	Derivation stop signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	LMT_ISTP	Public variable	BOOL	When MV variation rate limiter alarm occurred, selects stop integration. (TRUE: Stop, FALSE: Not stop)	TRUE, FALSE	FALSE	User
	SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>Execute deviation check processing.


<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL < |DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>
DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)</td></tr>
</table>

Item	Contents
2-degree-of-freedom PID operation	(see below)

2-degree-of-freedom PID operation

(1) Gain (Kp) is calculated as follows:

$$Kp=K\times P$$

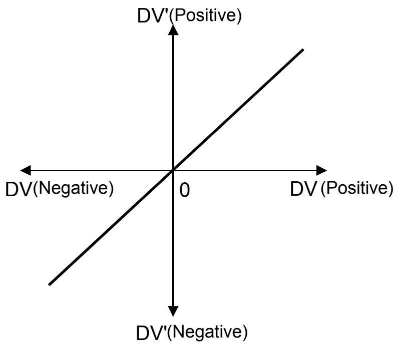
K: Output gain, P: Gain

(2) Output gain (K) is calculated as follows:

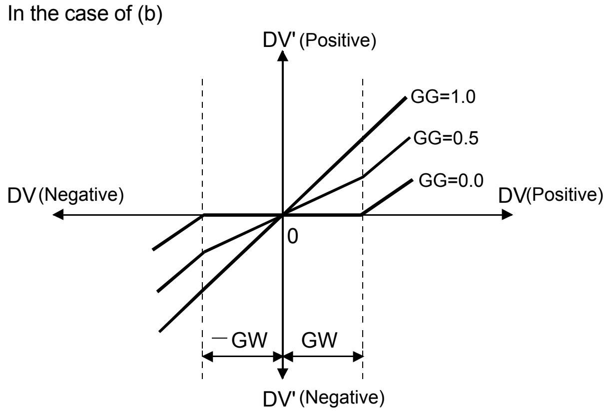
Condition		Output gain (K)
(a) When gap width (GW) = 0, the value of K corresponding to deviation (DV).		K=1
(b) When gap width (GW) > 0, the value of K corresponding to deviation (DV).	$\lvert DV\rvert \leqq GW$	K=GG
	$\lvert DV\rvert > GW$	$K=1-\frac{(1-GG)\times GW}{\lvert DV\rvert}$

DV: Deviation (%), GW: Gap width (%)= the rate of gap width corresponding to deviation, GG: Gap gain

In the case of (a)



In the case of (b)



(3) Deviation for PID operation (DV') is calculated as follows.

Condition	Deviation for PID operation (DV')
DV < – GW	DV' = – (GG × GW) + (DV+GW)
$\lvert DV\rvert \leqq GW$	DV' = GG × DV
DV > GW	DV' = GG × GW + (DV–GW)

DV': Deviation for PID operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as follows.

Condition	Deviation (DV)
Direct action (PN = 1)	DV (%) = PVP (%) – SVC (%)
Reverse action (PN = 0)	DV (%) = SVC (%) – PVP (%)

DV: Deviation (%), PVP (%): PV input value (%), SVC (%) = $\frac{100}{RH\text{-}RL}\times(SVC - RL)$,

RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting value (Current)

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>2-degree-of-freedom PID operation (continued)</td><td>
(5) 2-degree-of-freedom PID operation is executed as follows.

<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (△MV)</td><td colspan="2">$$\triangle MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{(1-\alpha)\times(DV_n - DV_{n-1})}_{\text{Proportional}} + \underbrace{\frac{CT}{Ti}\times DV_n}_{\text{Integral}} + \underbrace{(1-\beta)\times B_n}_{\text{Derivative}} + \underbrace{\alpha\times C_n+\beta\times D_n}_{\text{Feed forward compensation}} \}$$</td></tr>
<tr><td>Bn</td><td colspan="2">$$B_{n-1} + \frac{Md\times Td}{Md\times CT + Td}\times\{(DV_n - 2DV_{n-1} + DV_{n-2}) - \frac{CT\times B_{n-1}}{Td}\}$$</td></tr>
<tr><td>Dn</td><td>$$D_n = D_{n-1} + \frac{Md\times Td}{Md\times CT + Td}\times\{(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT\times D_{n-1}}{Td}\}$$</td><td>$$D_n = D_{n-1} + \frac{Md\times Td}{Md\times CT + Td}\times\{-(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT\times D_{n-1}}{Td}\}$$</td></tr>
<tr><td>Cn</td><td>$PV_n - PV_{n-1}$</td><td>$-(PV_n - PV_{n-1})$</td></tr>
</table>

Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Last deviation, DV_{n-2}: Deviation before last, PVn: Process variable, PV_{n-1}: Previous process variable, PV_{n-2}: Process variable before last, SVn: The processing result of engineering value conversion, α: 2-degree-of-freedom parameter (feedforward proportional), β: 2-degree-of-freedom parameter (feed forward derivative)

(a) Integral items and derivative items are listed below corresponding to each condition.

<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>When Td=0, or control mode is either MAN or CMV</td><td>Bn=0</td></tr>
<tr><td>Any of following 1), 2), 3)
1) Ti=0
2) When either MH or ML error occurs, MVP>MH and $\frac{CT}{Ti}\times DV_n > 0$
3) When either MH or ML error occurs, MVP<ML and $\frac{CT}{Ti}\times DV_n < 0$</td><td>$\frac{CT}{Ti}\times DV_n = 0$</td></tr>
</table>

Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit value, ML: Output low limit value, MVP: MV internal operation value

(b) Control cycle (CT) shall be set to be the integral multiple of execution cycle (△T).

(c) Integral constant should be set to be 0.0 or over control cycle (CT).

(d) PID operation of the tag access FB is executed in every control cycle (CT) (△MV output).
For other execution cycle (△T), hold the previous value (△MV=0).
</td></tr>
<tr><td>Engineering value conversion</td><td>
Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.

$$SV = \frac{RH - RL}{100}\times \text{Setting value (\%) from the primary loop} + RL$$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value (Target)
</td></tr>
<tr><td>Inverse engineering value conversion</td><td>
Convert the SVC of engineering value to percentage SVC (%).

$$SVC\ (\%) = \frac{100}{RH - RL}\times(SVC - RL)$$

RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting value (Current)
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether enable Alarm (ALM) detection or not in deviation check, SV variation rate high/low limiter
(1) Disable alarm detection processing with the tag data setting of "Disable Alarm Detection" (INH) and "Disable Alarm2 Detection" (INH2)
If the following bit items of disable alarm detection (INH) and disable alarm2 detection (INH2) of tag data are TRUE, DVLA of alarm (ALM) and DSVLA of alarm2 (ALM2), SVHA and SVLA will not be detected.
• ERRI, DVLI, DSVLI, SVHI, SVLI
(2) Disable alarm detection by loop stop processing:
Refer to loop stop processing in this section.
(3) Disable alarm detection when the control mode is CASDR
Alarm detection will not be executed.</td></tr>
<tr><td>Auto tuning (AT1, AT2)</td><td>Dynamic characteristics is detected and proportional gain (Kp), integral time (Ti), and derivative time (Td) are automatically calculated using auto tuning.
Select either the Step Response method or the Limit Cycle method for auto tuning.
(1) AT1 (Step Response method)
This method calculates proportional gain (Kp), integral time (Ti), and derivative time (Td) for PID with ZN method (Step Response method by Ziegler-Nichols) and sets their initial values.
Executable control modes are MAN and CMV.
For details, refer to Appendix 3.1.1.
(2) AT2 (Limit Cycle method)
This method calculates proportional gain (Kp), integral time (Ti), and derivative time (Td) for PID with oscillation amplitude and oscillation period by repeatedly operating MV at two positions and generating process variable cycle operation.
Executable control modes are MAN, AUT, CAS, CMV, and CSV.
For details, refer to Appendix 3.1.2.</td></tr>
<tr><td>PV tracking function</td><td>To avoid the sudden MV change in mode switching (MAN → AUTO), matches SV value (target) with PV value when control mode is MAN or CMV and maintains the accordance.
<table>
<tr><th>Condition</th><th>PV tracking function</th></tr>
<tr><td>PVTRK_EN=TRUE and control mode = "MAN (CMV)"</td><td>SV value (target) = PV value
SV value (current) = PV value</td></tr>
<tr><td>PVTRK_EN=FALSE or control mode ≠ "MAN (CMV)"</td><td>No processing</td></tr>
</table></td></tr>
<tr><td>SV variation rate high/low limiter</td><td>Checks variation rate high/low limiter to SV value (target value) in every control cycle (CT).
(1) Variation rate limiter
• The control mode is AUT or CAS or CSV.
SV variation rate high limit value inputted in % is converted to engineering value and the processing will be executed.
DSVL → DSVLT (DSVL: SV variation rate high limit value, DSVLT: value converted to engineering value from SV variation rate high limit value)
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>| SV - SVC | ≦ DSVLT</td><td>SV</td><td>FALSE (reset)</td></tr>
<tr><td>SV - SVC > DSVLT</td><td>SVC + DSVLT</td><td>TRUE (occur)</td></tr>
<tr><td>SV - SVC < - DSVLT</td><td>SVC - DSVLT</td><td>TRUE (occur)</td></tr>
</table>
SV: Setting value (target), SVC: Setting value (current)
If DSVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, DSVLA will be FALSE.
• The control mode is MAN or CMV or CASDR.
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>No</td><td>SV</td><td>FALSE (reset)</td></tr>
</table>
(2) High/low limiter
• The control mode is MAN, AUT, CAS, CMV and SVLMT_EN is TRUE.
<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter result</th><th colspan="2">Alarm2 (ALM2)</th></tr>
<tr><th>Target lower limit (SVLA)</th><th>Target upper limit (SVHA)</th></tr>
<tr><td>Variation rate limiter result > SH</td><td>SH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter result < SL</td><td>SL</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>SL ≦ variation rate limiter result ≦ SH</td><td>Variation rate limiter result</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
If SVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVLA will be FALSE.
If SVHI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVHA will be FALSE.
High/low limiter result is stored to SVC (setting value (current)).
• The control mode is CASDR or SVLMT_EN is FALSE.
Variation rate limiter result is stored to SVC (setting value (current)).</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Integration stop</td><td>Stops the operation of integral element.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>ISTP = TRUE</td><td>Stops the operation of integral element.</td></tr>
<tr><td>ISTP = FALSE</td><td>No processing</td></tr>
</table></td></tr>
<tr><td>Derivation stop</td><td>Stops the operation of differential element.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>DSTP = TRUE</td><td>Stops the operation of differential element.</td></tr>
<tr><td>DSTP = FALSE</td><td>No processing</td></tr>
</table></td></tr>
<tr><td>Stop integration in MV variation rate limiter occurrence</td><td>Stops the operation of integral element when MV variation rate limiter occurs.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>LMT_ISTP = TRUE and DMLA alarm occurs</td><td>Stops the operation of integral element when MV variation rate limiter occurs and the integral component is the same with the limit direction.</td></tr>
<tr><td>LMT_ISTP = FALSE</td><td>No processing</td></tr>
</table></td></tr>
</table>

Other Function

Item	Contents
Loop stop processing	The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Resets DVLA when the DVLA of alarm (ALM) occurs. Reset DSVLA, SVLA, and SVHA when the DSVLA, the SVLA, and the SVHA of alarm2 (ALM2) occur. 4) Alarm is not detected in deviation check, SV variation rate high/low limiter.

Processing Operation

Processing / Control mode	Deviation check	2-degree-of-freedom PID operation	Engineering value conversion	Inverse engineering value conversion	Alarm	Auto tuning (AT1)	Auto tuning (AT2)	PV tracking	SV variation rate high/low limiter	Integration stop	Derivation stop
MAN, CMV	○	○	×	○	○ (*1)	○	○	○	○ (*2, *3)	○	○
AUT	○	○	×	○	○ (*1)	×	○	×	○ (*3)	○	○
CAS, CSV	○	○	○	○	○ (*1)	×	○	×	○ (*3)	○	○
CASDR	×	○	○	○	×	×	×	×	×	○	○

○: Execute ×: Not execute

*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

*2 When the control mode is MAN, SV variation rate limiter processing is not executed.

*3 When sensor error (SEA) occurs and "Hold the output" is selected, processing is not executed. SVC (setting value (current)) is not updated as well.

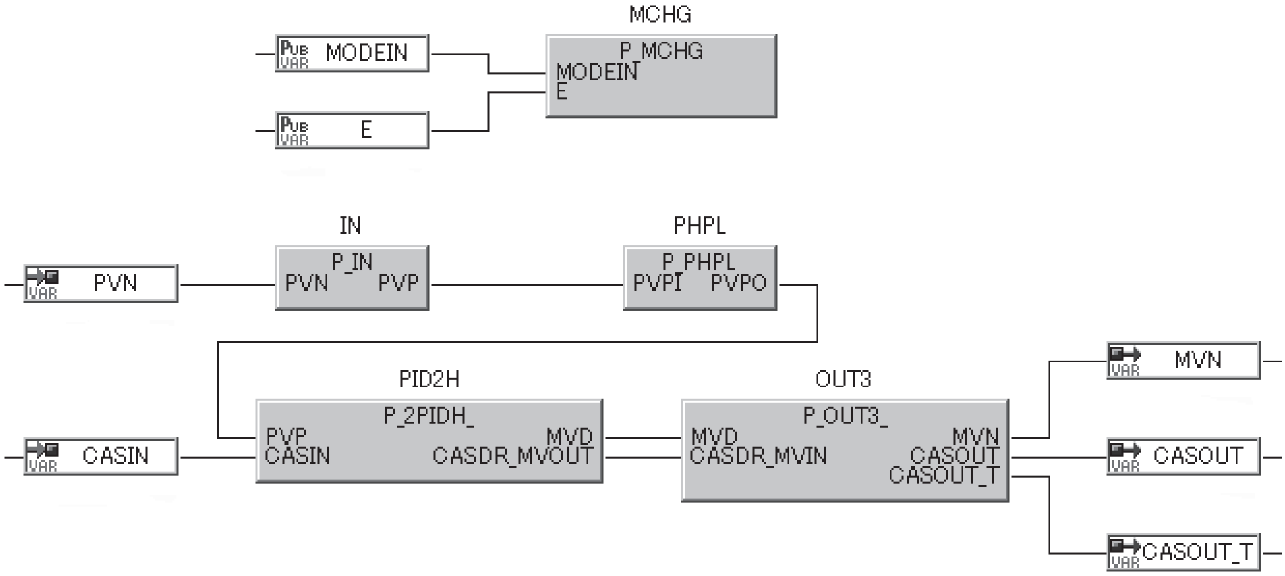
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

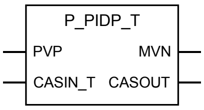
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



POINT
It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.9 Position Type PID Control (With Tracking to primary loop, Without Tracking from secondary loop) (P_PIDP_T)

FB	FBD parts
P_PIDP_T	

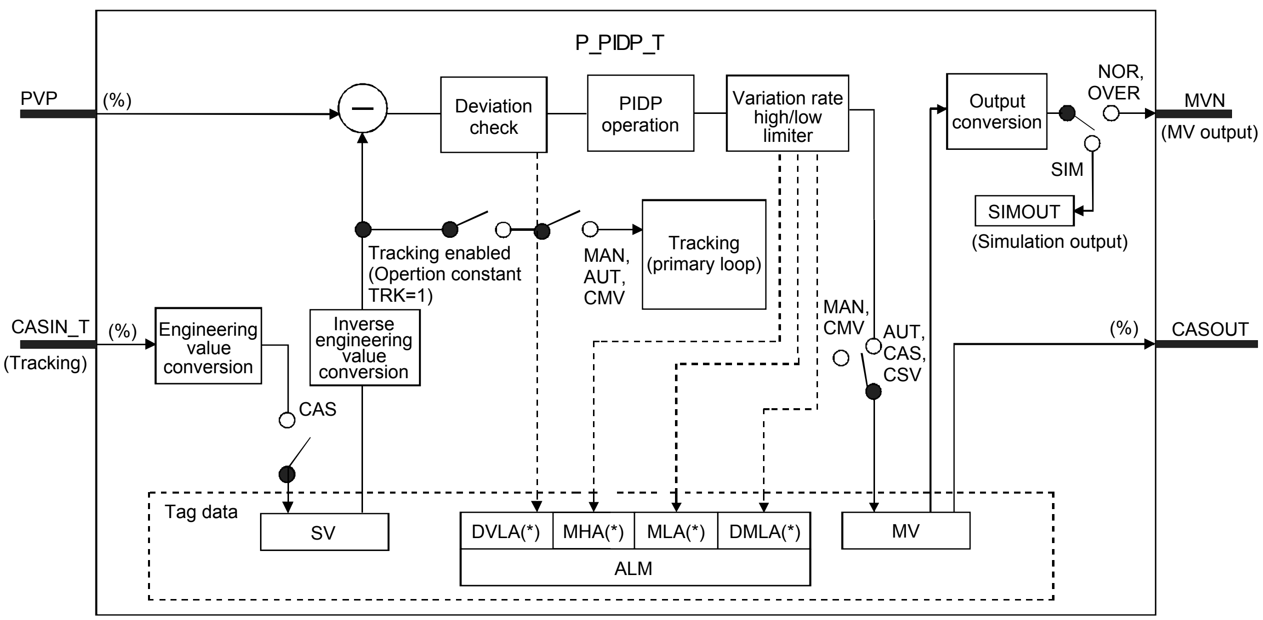
Corresponding tag type
PIDP

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Execute PID operation by PV-derivative, imperfect derivative and position type, and output the result.

Function/FB classification name: Tag access FB _ loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	Cascade SV input (unit: %) (With tracking)	0 to 100
Output	MVN	Output variable	REAL	MV output	NMIN to NMAX
	CASOUT	Output variable	REAL	Cascade SV output (unit: %)	0 to 100

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not host MV, FALSE: Host MV)	TRUE, FALSE	TRUE	User
	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variables (Others) (*1)

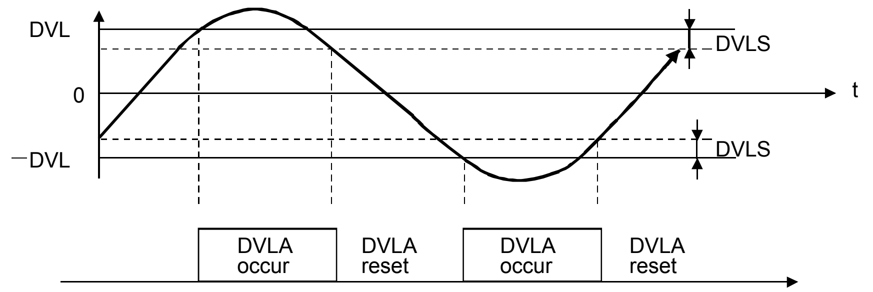
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>Execute deviation check processing.


<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>
DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)</td></tr>
</table>

Item	Contents
PIDP operation	

(1) Gain (Kp) is calculated as follows:

$$Kp = K \times P$$

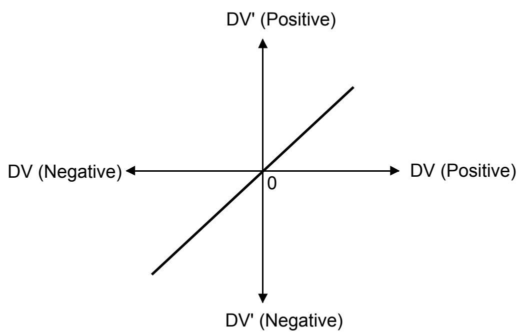
K: Output gain, P: Gain

(2) Output gain (K) is calculated as follows:

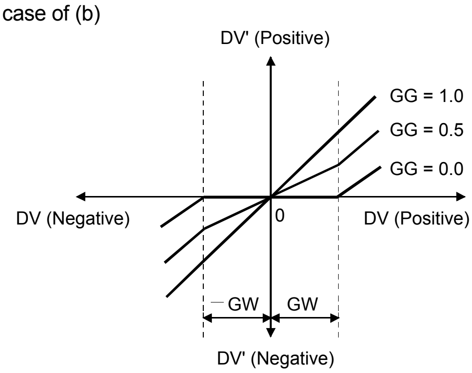
Condition		Output gain (K)
(a) When gap width (GW) = 0, the value of K corresponding to deviation (DV).		K=1
(b) When gap width (GW) > 0, the value of K corresponding to deviation (DV).	$\lvert DV \rvert \leqq GW$	K=GG
	$\lvert DV \rvert > GW$	$K = 1 - \frac{(1-GG) \times GW}{\lvert DV \rvert}$

DV: Deviation (%), GW: Gap width (%)=Rate of gap width to deviation, GG: Gap gain

In case of (a)



In case of (b)



(3) Deviation for PIDP operation (DV') is calculated as follows.

Condition	Deviation for PIDP operation (DV')
DV < – GW	DV' = – (GG × GW) + (DV+GW)
$\lvert DV \rvert \leqq GW$	DV' = GG × DV
DV > GW	DV' = GG × GW + (DV – GW)

DV': Deviation for PIDP operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as follows.

Condition	Deviation (DV)
Direct action (PN=1)	DV (%) = PVP (%) – SV (%)
Reverse action (PN=0)	DV (%) = SV (%) – PVP (%)

DV: Deviation (%), PVP (%): PV input value (%), SV (%) = $\frac{100}{RH-RL} \times$ (SV-RL),

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PIDP operation (continued)</td><td>
(5) PIDP operation is calculated as follows
<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (MV)</td><td colspan="2">$MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{DV_n}_{\text{Proportional}} + \underbrace{I_n}_{\text{Integral}} + \underbrace{B_n}_{\text{Derivative (imperfect derivative)}} \}$
(I_n, B_n are as follows)</td></tr>
<tr><td>I_n</td><td colspan="2">$I_n = I_{n-1} + \frac{CT}{Ti} \times DV_n$</td></tr>
<tr><td>B_n</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ (PV_n - PV_{n-1}) - \frac{CT \times B_{n-1}}{Td} \}$</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ -(PV_n - PV_{n-1}) - \frac{CT \times B_{n-1}}{Td} \}$</td></tr>
</table>
Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Previous deviation, PVn: Process variable, PV_{n-1}: Previous process variable, SVn: Engineering value conversion processing result.

(a) Integral item and derivative item are listed below corresponding to each condition.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>Td=0, or control cycle is any of MAN, CMV</td><td>Bn=0</td></tr>
<tr><td>Any of 1), 2), 3)
1) Ti=0
2) When MH error occurs $\frac{CT}{Ti} \times DVn > 0$
3) When ML error occurs $\frac{CT}{Ti} \times DVn < 0$</td><td>$\frac{CT}{Ti} \times DVn = 0$</td></tr>
</table>
Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit value, ML: Output low limit value

(b) Control cycle (CT) should be set to be the integral multiple of execution cycle (△T).

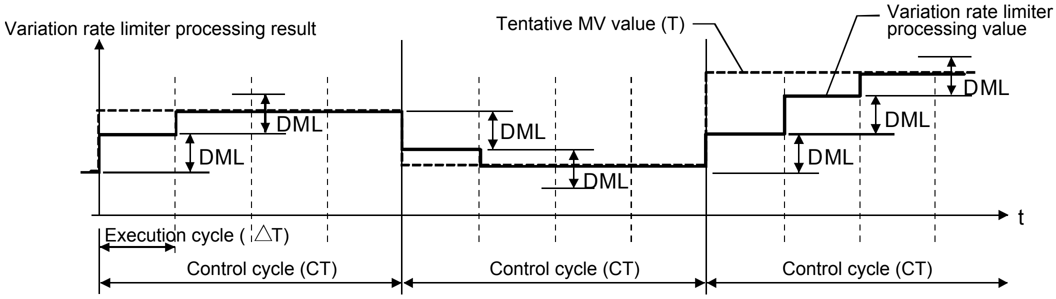
(c) Integral constant should be set to be 0.0 or over control cycle (CT).

(d) PIDP operation of the tag access FB is executed in every control cycle (CT), (MV output).
For other execution cycle (△T), the previous value shall be applied. (MV=0)
</td></tr>
<tr><td>Engineering value conversion</td><td>
Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.
<table><tr><td>$SV = \frac{RH - RL}{100} \times$ Setting value (%) from the primary loop +RL</td></tr></table>
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Inverse engineering value conversion</td><td>
Convert the setting value (SV) of engineering value to percentage MV (%).
<table><tr><td>$SV (\%) = \frac{100}{RH - RL} \times (SV - RL)$</td></tr></table>
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Tracking processing</td><td>
Tracking operation for input variable CASIN_T is executed as follows:
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SCPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table>
</td></tr>
</table>

Item	Contents
Variation rate and high/low limiter	Execute variation rate limiter and high/low limit check to the input value.

Execute variation rate limiter and high/low limit check to the input value.

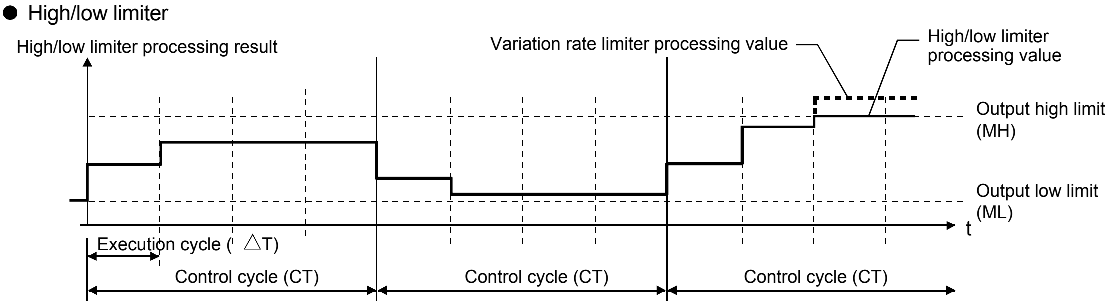
● Variation rate limiter



Condition	Variation rate limiter processing result	Alarm (ALM)
		Output variation rate limit (DMLA)
|T-MV| ≦ DML	T	FALSE (reset)
T-MV>DML	MV+DML	TRUE (occur)
T-MV<-DML	MV-DML	TRUE (occur)

T: Tentative MV value, MV: Manipulated variable, DML: Output variation rate high limit

● High/low limiter

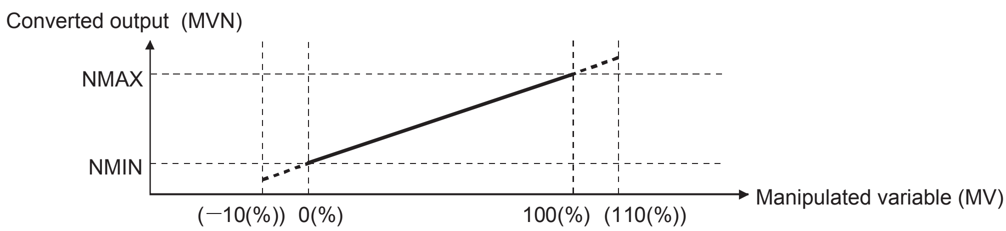


Condition	High/low limiter processing result	Alarm (ALM)	
		Output low limit (MLA)	Output high limit (MHA)
Variation rate limiter processing result > MH	MH	FALSE (reset)	TRUE (occur)
Variation rate limiter processing result < ML	ML	TRUE (occur)	FALSE (reset)
ML ≦ Variation rate limiter processing result ≦ MH	Variation rate limiter processing value	FALSE (reset)	FALSE (reset)

MH: Output high limit, ML: Input low limit

Output conversion

Conversion processing output is carried out.



$$\text{Converted output (MVN)}=\{(\text{NMAX-NMIN}) \times \frac{\text{MV}}{100}\}+\text{NMIN}$$

NMAX: Converted output high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%), MVN: Converted output value

Item	Contents
Disable Alarm Detection	Set whether Enable Alarm Detection or not in variation rate limiter and high/low limiter. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected. ● ERRI, DMLI, MHI, MLI (2) Disable Alarm Detection by control mode: If the control mode is MAN and CMV, reset DMLA, MHA and MLA of alarm (ALM) and DMLA, MHA and MLA will be detected. (3) Disable Alarm Detection by loop stop processing: Please refer to loop stop processing in the following contents.

Other Function

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occurs. 4) Alarm is not detected in variation rate limiter and high/ low limiter.

Processing Operation

Processing / Control mode	Deviation check	PIDP operation	Engineering value conversion	Inverse engineering value conversion	Variation rate Limiter and high/low limiter	Output conversion	Tracking	Alarm
MAN, CMV	○	○	×	○	×	○	○ (*1)	○ (*2)
AUT	○	○	×	○	○	○	○ (*1)	○ (*2)
CAS, CSV	○	○	○	○	○	○	×	○ (*2)

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

*2 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

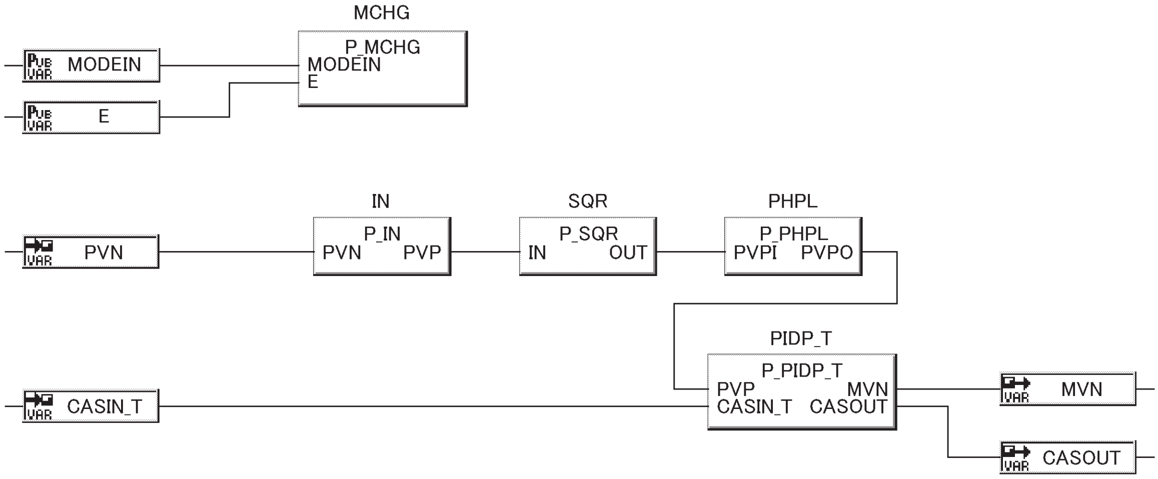
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.

Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

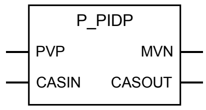
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.10 Position Type PID Control (Without Tracking to primary loop, Without Tracking from secondary loop) (P_PIDP)

FB	FBD parts
P_PIDP	

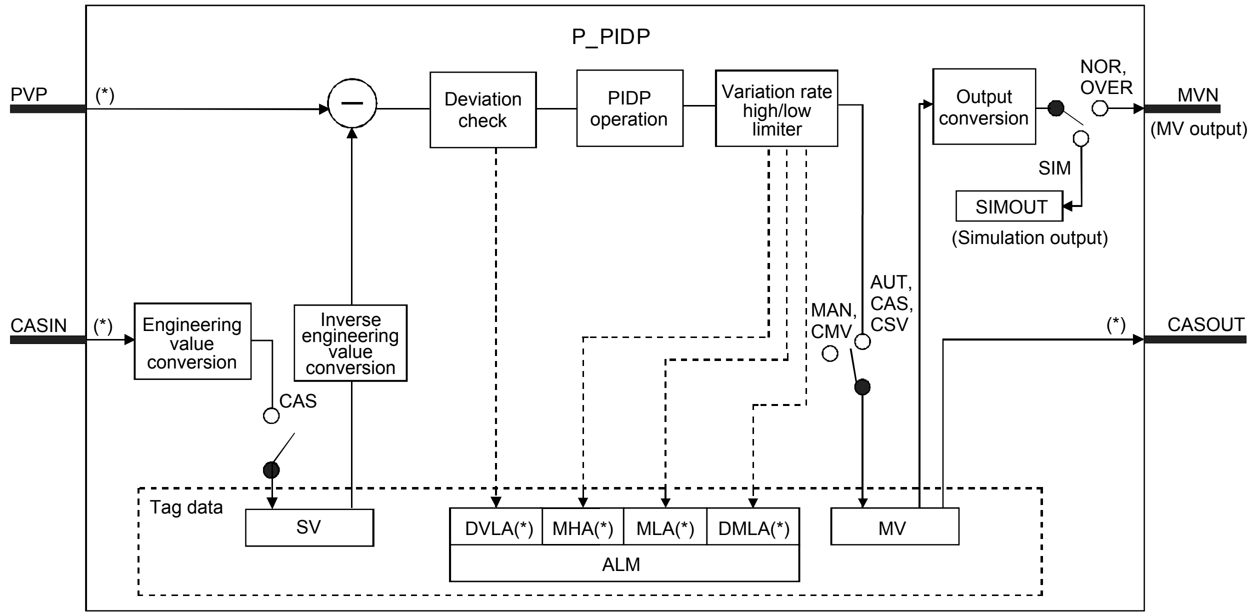
Corresponding tag type
PIDP

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Execute PID operation by PV-derivative, imperfect derivative and position type, and output the result.

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVN	Output variable	REAL	MV output	NMIN to NMAX
	CASOUT	Output variable	REAL	Cascade SV output (unit: %)	0 to 100

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

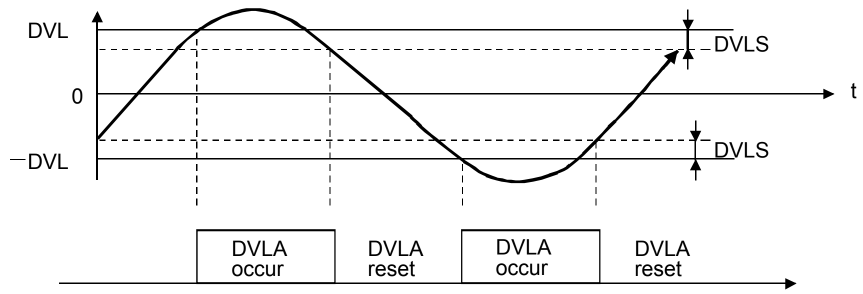
*2 Indicates the simulation processing.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>Execute deviation check operation.



<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)</td></tr>
</table>

Item	Contents
PIDP operation	

(1) Gain (Kp) is calculated as below:

$$Kp = K \times P$$

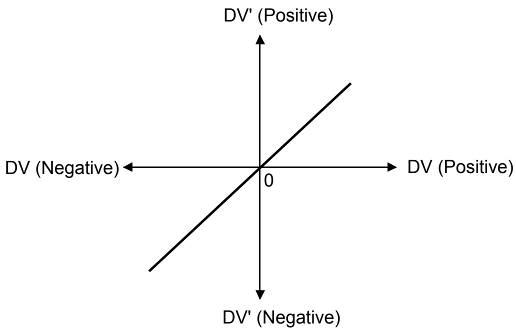
K: Output gain, P: Gain

(2) Output gain (K) is calculated as below:

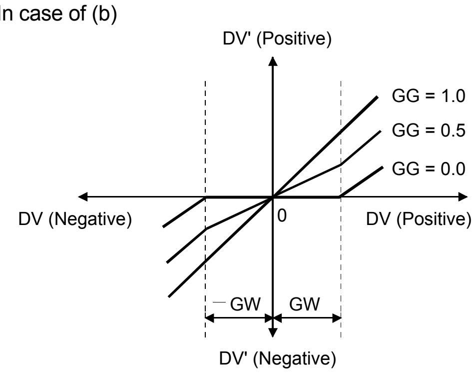
Condition		Output gain (K)
(a) When gap width (GW) = 0, the K value corresponding to the deviation (DV)		K=1
(b) When gap width (GW) > 0, the K value corresponding to the deviation (DV)	$\lvert DV \rvert \leqq GW$	K=GG
	$\lvert DV \rvert > GW$	$K = 1 - \dfrac{(1 - GG) \times GW}{\lvert DV \rvert}$

DV: Deviation (%), GW: Gap width (%)=Rate of gap width to deviation, GG: Gap gain

In case of (a)



In case of (b)



(3) Deviation for PIDP operation (DV') is calculated as below.

Condition	Deviation for PIDP operation (DV')
DV<-GW	DV'= - (GG × GW)+(DV+GW)
$\lvert DV \rvert \leqq GW$	DV'=GG × DV
DV>GW	DV'=GG × GW+(DV-GW)

DV': Deviation for PIDP operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as below.

Condition	Deviation (DV)
Direct action (PN=1)	DV (%) = PVP (%) –SV (%)
Reverse action (PN=0)	DV (%) + SV (%) –PVP (%)

DV: Deviation (%), PVP (%): PV input (%), SV (%) = $\dfrac{100}{RH-RL} \times$ (SV-RL), RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PIDP operation (continued)</td><td>
(5) PIDP operation is conducted as below.

<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DV_n)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (MV)</td><td colspan="2">$MV = \underbrace{K_p}_{\text{Gain}} \times \{ \underbrace{DV_n}_{\text{Proportional}} + \underbrace{I_n}_{\text{Integral}} + \underbrace{B_n}_{\text{Derivative (imperfect derivative)}} \}$
(See below for details about I_n, B_n)</td></tr>
<tr><td>I_n</td><td colspan="2">$I_n = I_{n-1} + \frac{CT}{Ti} \times DV_n$</td></tr>
<tr><td>B_n</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ (PV_n - PV_{n-1}) - \frac{CT \times B_{n-1}}{Td} \}$</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ -(PV_n - PV_{n-1}) - \frac{CT \times B_{n-1}}{Td} \}$</td></tr>
</table>

K_p: Gain, T_i: Integral time, T_d: Derivative time, M_d: Derivative gain, CT: Control cycle, DV_n: Deviation, DV_{n-1}: Previous deviation, PV_n: Process variable, PV_{n-1}: Previous process variable, SV_n: Engineering value conversion processing result

(a) Integral term and derivative term are listed below corresponding to each condition.

<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>Td=0, or control mode being either MAN or CMV</td><td>Bn = 0</td></tr>
<tr><td>Any of 1), 2), 3)
1) Ti=0
2) When MH error occurs $\frac{CT}{Ti} \times DVn > 0$
3) When ML error occurs $\frac{CT}{Ti} \times DVn < 0$</td><td>$\frac{CT}{Ti} \times DVn = 0$</td></tr>
</table>

Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit, ML: Output low limit

(b) Control cycle (CT) should be set to be the integral multiple of execution cycle ($\triangle$T).

(c) Integral constant should be set to be 0.0 or over control cycle (CT).

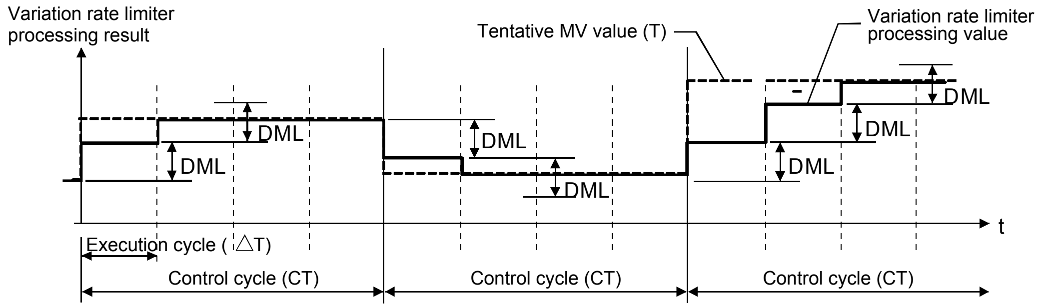
(d) PIDP operation of the tag access FB is executed in every control cycle (CT), (output MV). For other execution cycle ($\triangle$T), the last execution value of MV is held.
</td></tr>
<tr><td>Engineering value conversion</td><td>
When the control mode is CAS/CSV, the setting value (%) from the primary loop is converted into engineering value.

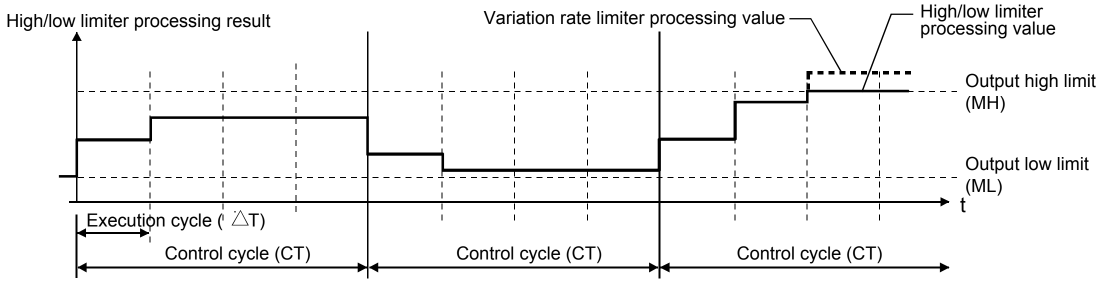
$$SV = \frac{RH - RL}{100} \times \text{Setting value (\%) from the primary loop} + RL$$

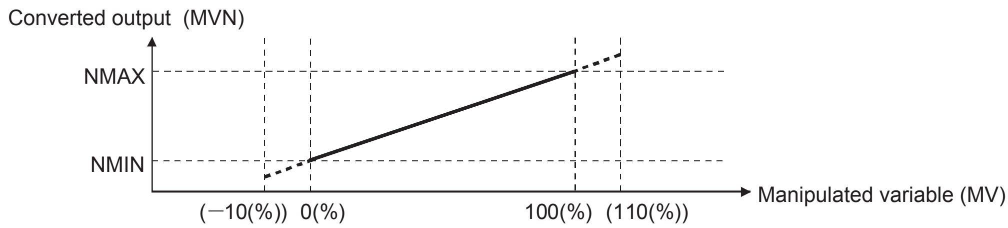
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Inverse engineering value conversion</td><td>
The setting value (SV) of engineering value is converted to percentage SV (%).

$$SV (\%) = \frac{100}{RH - RL} \times (SV - RL)$$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Variation rate and high/low limiter</td><td>
Execute variation rate check and high/low limit check to the input value.
● Variation rate limiter


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">Variation rate limiter processing result</th><th>Alarm (ALM)</th></tr>
<tr><th>Output variation rate limit (DMLA)</th></tr>
<tr><td>|T-MV| ≦ DML</td><td>T</td><td>FALSE (reset)</td></tr>
<tr><td>T-MV>DML</td><td>MV+DML</td><td>TRUE (occur)</td></tr>
<tr><td>T-MV<-DML</td><td>MV-DML</td><td>TRUE (occur)</td></tr>
</table>
T: Tentative MV value, MV: Manipulated variable, DML: Output variation rate high limit
● High/low limiter


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter processing result</th><th colspan="2">Alarm (ALM)</th></tr>
<tr><th>Output low limit (MLA)</th><th>Output high limit (MHA)</th></tr>
<tr><td>Variation rate limiter processing result > MH</td><td>MH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter processing result < ML</td><td>ML</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>ML≦ Variation rate limiter processing result ≦ MH</td><td>Variation rate limiter processing value</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
MH: Output high limit, ML: Output low limit
</td></tr>
<tr><td>Output conversion</td><td>
Execute output conversion procession.


$$\text{Converted output (MVN)} = \{(\text{NMAX-NMIN}) \times \frac{\text{MV}}{100}\}\text{+NMIN}$$

NMAX: Output conversion high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%), MVN: Converted output value
</td></tr>
</table>

Item	Contents
Disable Alarm Detection	Set whether Enable Alarm Detection or not in variation rate limiter and high/low limiter. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected. ● ERRI, DMLI, MHI, MLI (2) Disable Alarm Detection by control mode: If the control mode is MAN and CMV, reset DMLA, MHA and MLA of alarm (ALM) and DMLA, MHA and MLA will be detected. (3) Disable Alarm Detection by loop stop processing: Please refer to loop stop processing in the following contents.

Other Function

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occurs. 4) Alarm is not detected in variation rate limiter and high/ low limiter.

Processing Operation

Processing / Control mode	Deviation check	PIDP operation	Engineering value conversion	Inverse engineering value conversion	Variation rate limiter and high/low limiter	Output conversion	Alarm
MAN, CMV	○	○	×	○	×	○	○ (*1)
AUT	○	○	×	○	○	○	○ (*1)
CAS, CSV	○	○	○	○	○	○	○ (*1)

○: Execute ×: Not execute

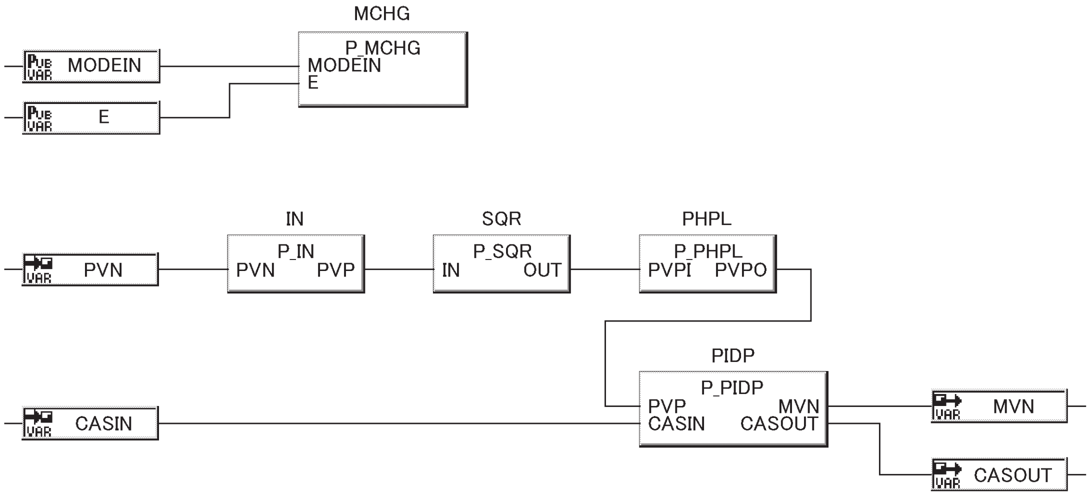
*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

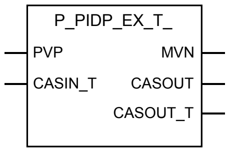
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.11 Position Type PID Control (With Tracking to primary loop, With Tracking from secondary loop) (P_PIDP_EX_T_)

FB	FBD parts
P_PIDP_EX_T_	

Corresponding tag type
PIDP

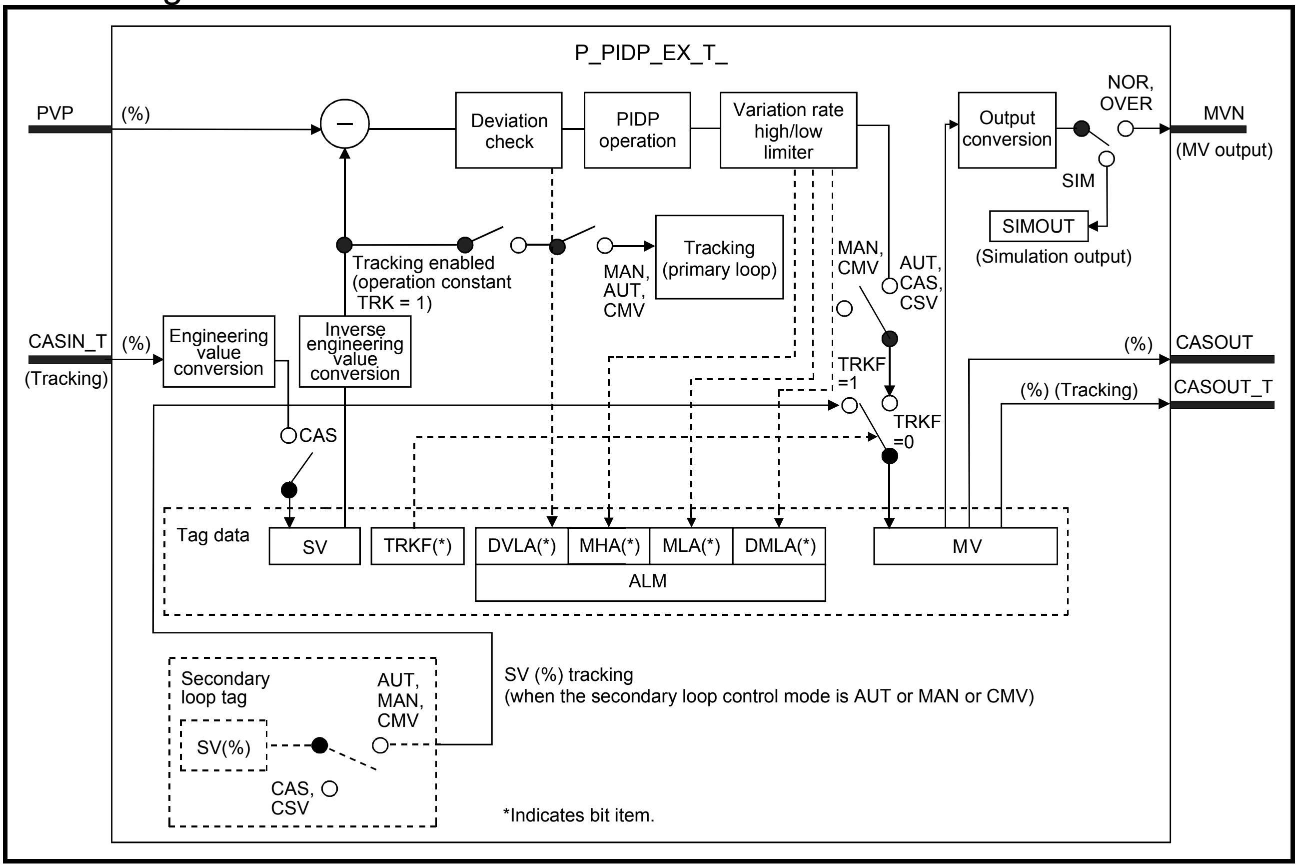
Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Execute PID operation by PV- derivative, imperfect derivative and position type and output the result.
MV value bumpless switching and tracking from the secondary loop at control mode change are also possible. *1

Function/FB classification name: Tag access FB _ loop control operation FB

*1 Requires Process CPU or Redundant CPU whose upper five digits of serial No. are 07032 or later.

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	Cascade SV input (unit: %) (With tracking)	0 to 100
Output	MVN	Output variable	REAL	MV output	NMIN to NMAX
	CASOUT	Output variable	REAL	Cascade SV output (unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (unit: %) (With tracking)	0 to 100

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not host MV, FALSE: Host MV)	TRUE, FALSE	TRUE	User
	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variables (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

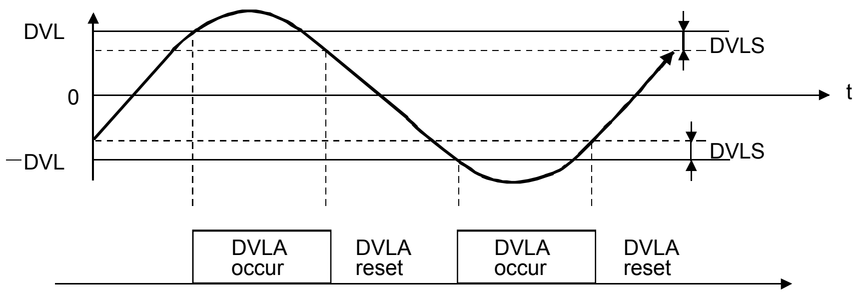
*2 Indicates the simulation processing.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
Execute deviation check processing.



<table>
<tr><td rowspan="2">Condition</td><td>Alarm (ALM)</td></tr>
<tr><td>Large deviation (DVLA)</td></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PIDP operation</td><td>

(1) Gain (Kp) is calculated as follows:

$$Kp=K\times P$$

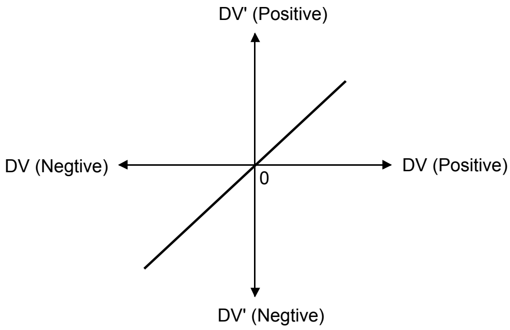
K: Output gain, P: Gain

(2) Output gain (K) is calculated as follows:

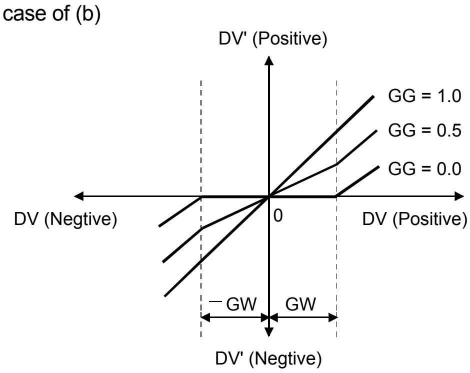
<table>
<tr><th colspan="2">Condition</th><th>Output gain (K)</th></tr>
<tr><td colspan="2">(a) When gap width (GW) = 0, the value of K corresponding to deviation (DV).</td><td>K=1</td></tr>
<tr><td rowspan="2">(b) When gap width (GW) > 0, the value of K corresponding to deviation (DV).</td><td>$|DV| \leqq GW$</td><td>K=GG</td></tr>
<tr><td>$|DV| > GW$</td><td>$K=1-\frac{(1-GG)\times GW}{|DV|}$</td></tr>
</table>

DV: Deviation (%), GW: Gap width (%)=Rate of gap width to deviation, GG: Gap gain

In case of (a)



In case of (b)



(3) Deviation for PIDP operation (DV') is calculated as follows.

<table>
<tr><th>Condition</th><th>Deviation for PIDP operation (DV')</th></tr>
<tr><td>DV < –GW</td><td>$DV' = -(GG\times GW) + (DV+GW)$</td></tr>
<tr><td>$|DV| \leqq GW$</td><td>$DV' = GG\times DV$</td></tr>
<tr><td>DV > GW</td><td>$DV' = GG\times GW + (DV-GW)$</td></tr>
</table>

DV': Deviation for PIDP operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as follows.

<table>
<tr><th>Condition</th><th>Deviation (DV)</th></tr>
<tr><td>Direct action (PN=1)</td><td>DV (%) = PVP (%) – SV (%)</td></tr>
<tr><td>Reverse action (PN=0)</td><td>DV (%) = SV (%) – PVP (%)</td></tr>
</table>

DV: Deviation (%), PVP (%): PV input value (%), SV (%) = $\frac{100}{RH-RL}\times(SV-RL)$,

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PIDP operation (continued)</td><td>

(5) PIDP operation is calculated as follows.

<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (MV)</td><td colspan="2">$MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{DV_n}_{\text{Proportional}} + \underbrace{I_n}_{\text{Integral}} + \underbrace{B_n}_{\text{Derivative (imperfect derivative)}} \}$
(I_n, B_n are as follows)</td></tr>
<tr><td>In</td><td colspan="2">$I_n = I_{n-1} + \frac{CT}{Ti} \times DV_n$</td></tr>
<tr><td>Bn</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ (PV_n - PV_{n-1}) - \frac{CT \times B_{n-1}}{Td} \}$</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ -(PV_n - PV_{n-1}) - \frac{CT \times B_{n-1}}{Td} \}$</td></tr>
</table>

Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Previous deviation, PVn: Process variable, PV_{n-1}: Previous process variable, SVn: Engineering value conversion processing result

(a) Integral item and derivative item are listed below corresponding to each condition.

<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>Td=0, or control cycle is any of MAN, CMV</td><td>Bn=0</td></tr>
<tr><td>Any of 1), 2), 3), 4)
1) Ti=0
2) When MH error occurs $\frac{CT}{Ti} \times DVn > 0$
3) When ML error occurs $\frac{CT}{Ti} \times DVn < 0$
4) Control mode is either MAN or CMV</td><td>$\frac{CT}{Ti} \times DVn = 0$</td></tr>
<tr><td>Any of 1), 2)
1) When tracking from the secondary loop
2) When the control mode is changed from MAN or CMV to any of AUT, CAS and CSV</td><td>$I_{n-1} = \frac{MV}{Kp} - (DV_n + B_n)$</td></tr>
</table>

Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit value, ML: Output low limit value

(b) Control cycle (CT) should be set to be the integral multiple of execution cycle (△T).

(c) Integral constant should be set to be 0.0 or over control cycle (CT).

(d) PIDP operation of the tag access FB is executed in every control cycle (CT), (MV output).
For other execution cycle (△T), the previous value shall be applied. (MV=0)

</td></tr>
<tr><td>Engineering value conversion</td><td>

Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.

$$SV = \frac{RH - RL}{100} \times \text{Setting value (\%) from the primary loop} + RL$$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

</td></tr>
<tr><td>Inverse engineering value conversion</td><td>

Convert the setting value (SV) of engineering value to percentage SV (%).

$$SV (\%) = \frac{100}{RH - RL} \times (SV - RL)$$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

</td></tr>
<tr><td>Tracking processing</td><td>

Tracking operation for input variable CASIN_T is executed as follows:

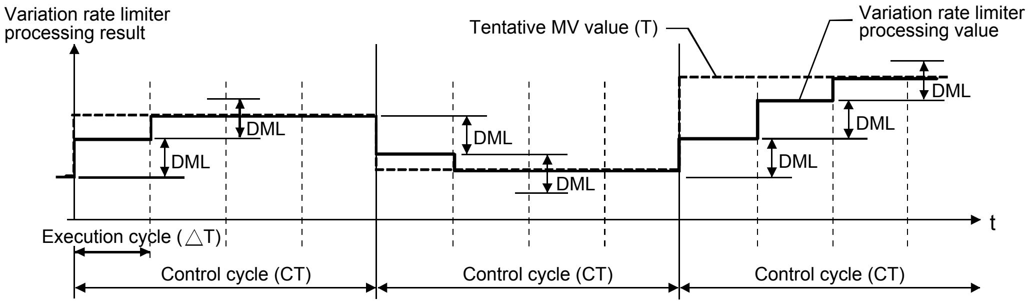
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SCPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table>

</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Variation rate and high/low limiter</td><td>

Execute variation rate limiter and high/low limit check to the input value.

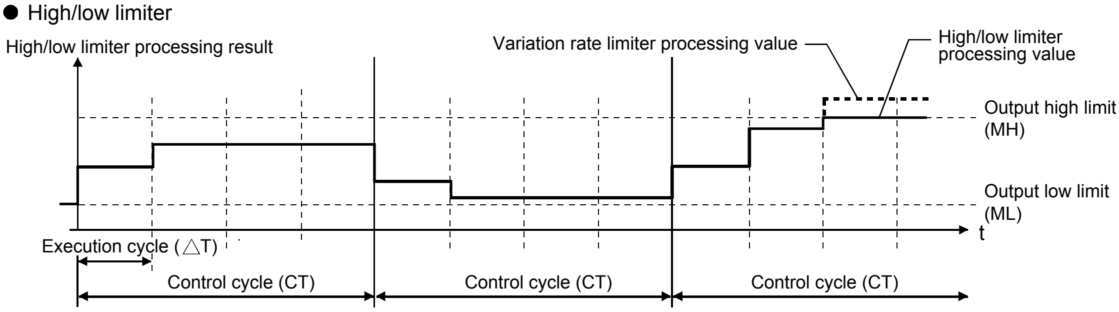
● Variation rate limiter



<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">Variation rate limiter processing result</th><th>Alarm (ALM)</th></tr>
<tr><th>Output variation rate limit (DMLA)</th></tr>
<tr><td>|T-MV| ≦ DML</td><td>T</td><td>FALSE (reset)</td></tr>
<tr><td>T-MV > DML</td><td>MV+DML</td><td>TRUE (occur)</td></tr>
<tr><td>T-MV <-DML</td><td>MV-DML</td><td>TRUE (occur)</td></tr>
</table>

T: Tentative MV value, MV: Manipulated variable, DML: Output variation rate high limit

● High/low limiter

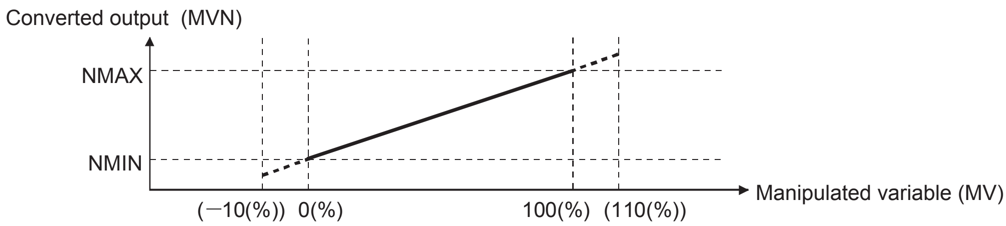


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter processing result</th><th colspan="2">Alarm (ALM)</th></tr>
<tr><th>Output low limit (MLA)</th><th>Output high limit (MHA)</th></tr>
<tr><td>Variation rate limiter processing result > MH</td><td>MH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter processing result < ML</td><td>ML</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>ML ≦ Variation rate limiter processing result ≦ MH</td><td>Variation rate limiter processing value</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>

MH: Output high limit, ML: Input low limit

</td></tr>
<tr><td>Output conversion</td><td>

Conversion processing output is carried out.



$$\text{Converted output (MVN)}=\{(\text{NMAX-NMIN}) \times \frac{\text{MV}}{100}\}+\text{NMIN}$$

NMAX: Converted output high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%), MVN: Converted output value

</td></tr>
</table>

Item	Contents
Disable Alarm Detection	Set whether Enable Alarm Detection or not in variation rate limiter and high/low limiter. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected. ● ERRI, DMLI, MHI, MLI (2) Disable Alarm Detection by control mode: If the control mode is MAN and CMV, reset DMLA, MHA and MLA of alarm (ALM) and DMLA, MHA and MLA will be detected. (3) Disable Alarm Detection by loop stop processing: Please refer to loop stop processing in the following contents.

Other Function

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occurs. 4) Alarm is not detected in variation rate limiter and high/ low limiter.

Processing Operation

Processing / Control mode	Deviation check	PIDP operation	Engineering value conversion	Inverse engineering value conversion	Variation rate and high/low limiter	Output conversion	Tracking	Alarm
MAN, CMV	○	○	×	○	×	○	○ (*1)	× (*2)
AUT	○	○	×	○	○	○	○ (*1)	○ (*3)
CAS, CSV	○	○	○	○	○	○	×	○ (*3)

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

*2 An alarm (ALM) whose corresponding bit is TRUE (occurred) is reset, and the alarm cannot be detected.

*3 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Restriction

This FB supports the Process CPU and the Redundant CPU whose upper five digits of serial No.s are 07032 or later.

If either of the CPU whose upper five digits of serial No. are 07031 or earlier is used, the MV value bumpless switching and tracking processing from the secondary loop at control mode change will not be executed.

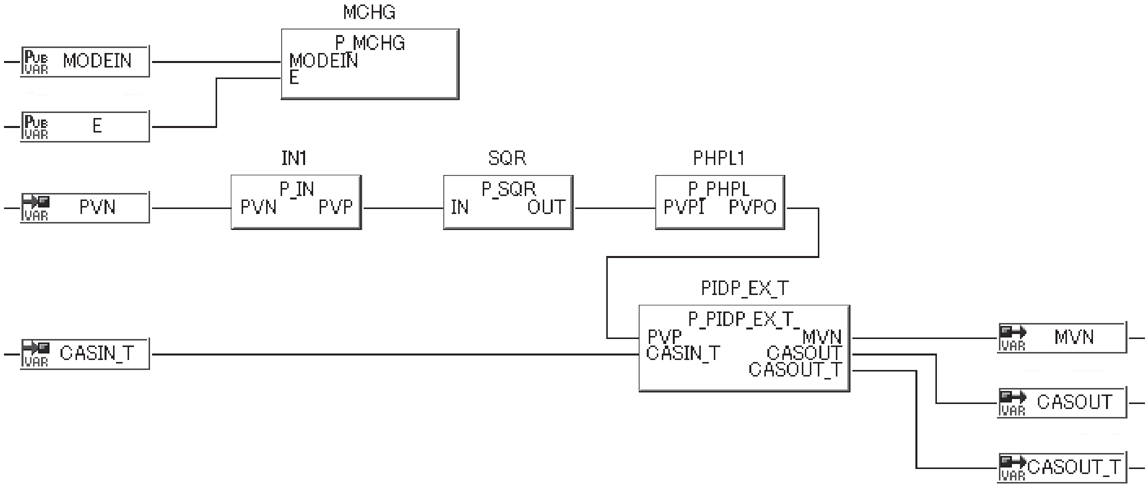
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.

Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

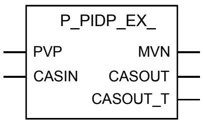
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



POINT
It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.12 Position Type PID Control (Without Tracking to primary loop, With Tracking from secondary loop) (P_PIDP_EX_)

FB	FBD parts
P_PIDP_EX_	

Corresponding tag type
PIDP

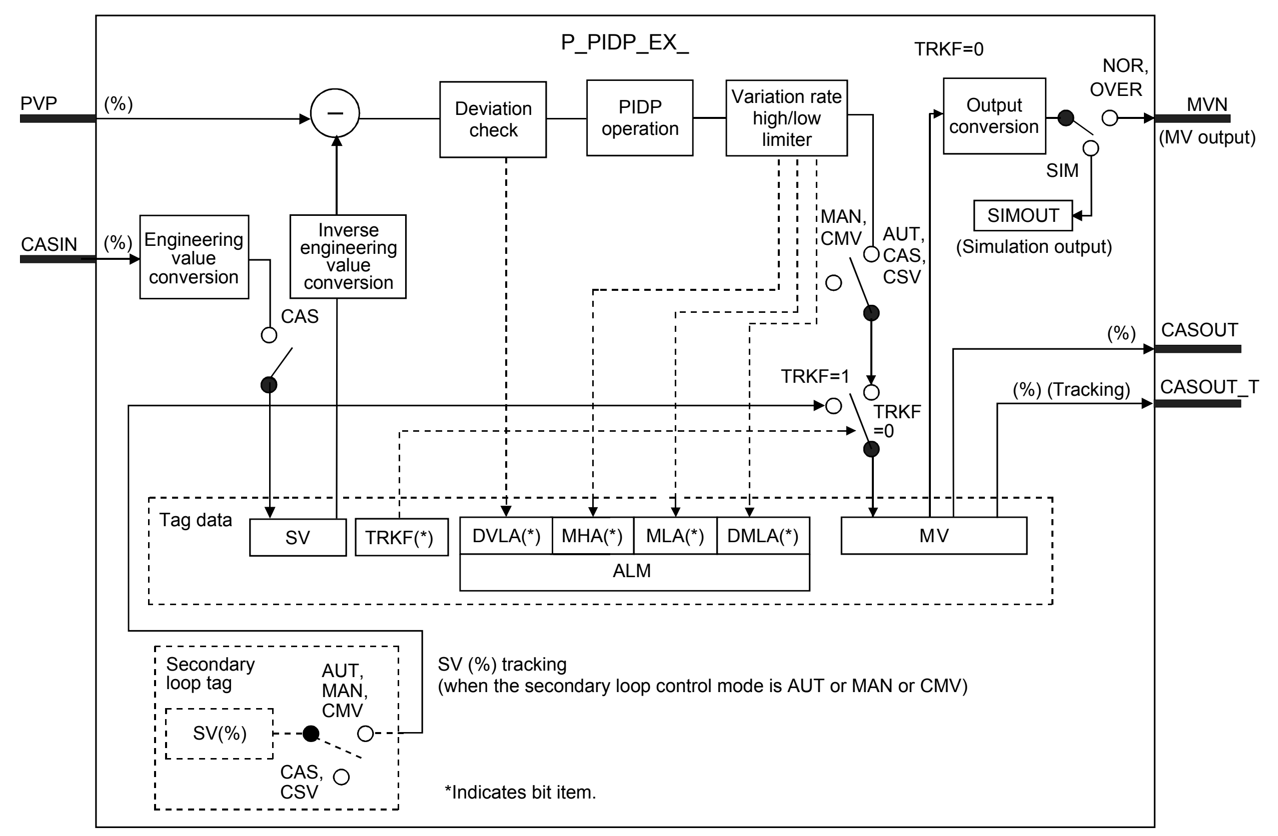
Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Execute PID operation by PV-derivative, imperfect derivative and position type, and output the result.
MV value bumpless switching and tracking from the secondary loop at control mode change are also possible. *1

Function/FB classification name: Tag access FB _ Loop control operation FB

*1 Requires Process CPU or Redundant CPU whose upper five digits of serial No. are 07032 or later.

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVN	Output variable	REAL	MV output	NMIN to NMAX
	CASOUT	Output variable	REAL	Cascade SV output (unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (unit: %)(With tracking)	0 to 100

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

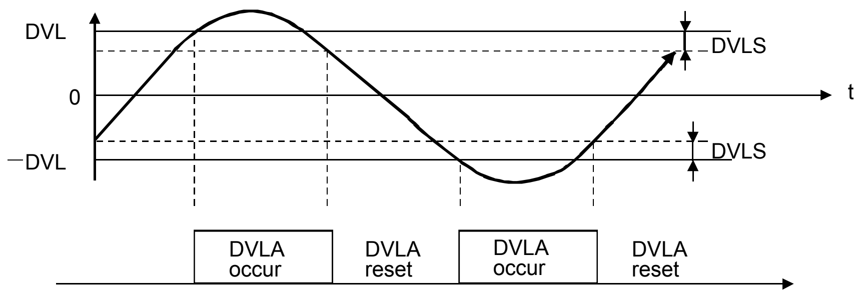
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.

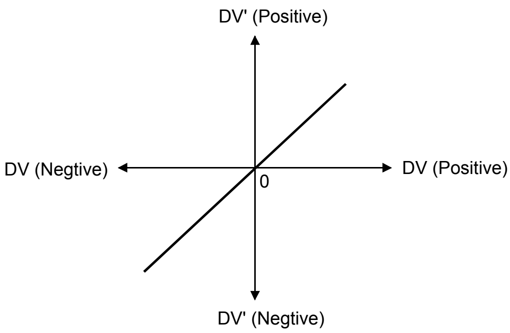
Tag Data

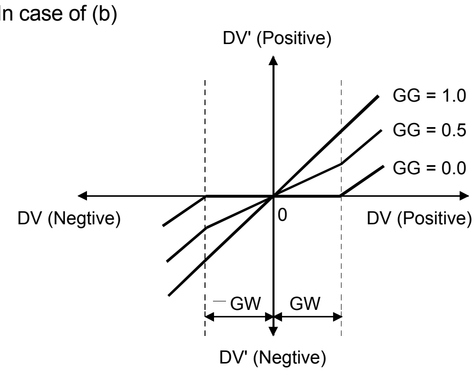
For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>Execute deviation check operation.


<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>
DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PIDP operation</td><td>
(1) Gain (Kp) is calculated as below:
$Kp=K \times P$
K: Output gain, P: Gain
(2) Output gain (K) is calculated as below:
<table>
<tr><th colspan="2">Condition</th><th>Output gain (K)</th></tr>
<tr><td colspan="2">(a) When gap width (GW) = 0, the K value corresponding to the deviation (DV)</td><td>K=1</td></tr>
<tr><td rowspan="2">(b) When gap width (GW) > 0, the K value corresponding to the deviation (DV)</td><td>$|DV| \leqq GW$</td><td>K=GG</td></tr>
<tr><td>|DV|>GW</td><td>$K=1-\frac{(1-GG) \times GW}{|DV|}$</td></tr>
</table>
DV: Deviation (%), GW: Gap width (%)=Rate of gap width to deviation, GG: Gap gain
In case of (a)


In case of (b)


(3) Deviation for PIDP operation (DV') is calculated as below.
<table>
<tr><th>Condition</th><th>Deviation for PIDP operation (DV')</th></tr>
<tr><td>DV < –GW</td><td>$DV' = -(GG \times GW) + (DV+GW)$</td></tr>
<tr><td>$|DV| \leqq GW$</td><td>$DV' = GG \times DV$</td></tr>
<tr><td>DV > GW</td><td>$DV' = GG \times GW + (DV-GW)$</td></tr>
</table>
DV': Deviation for PIDP operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain
(4) Deviation for direct/reverse action (DV) is calculated as below.
<table>
<tr><th>Condition</th><th>Deviation (DV)</th></tr>
<tr><td>Direct action (PN=1)</td><td>DV (%) = PVP (%) – SV (%)</td></tr>
<tr><td>Reverse action (PN=0)</td><td>DV (%) + SV (%) – PVP (%)</td></tr>
</table>
DV: Deviation (%), PVP (%): PV input (%), SV (%) = $\frac{100}{RH-RL} \times (SV-RL)$, RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PIDP operation (continued)</td><td>

(5) PIDP operation is conducted as below.

<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (MV)</td><td colspan="2">$MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{DV_n}_{\text{Proportional}} + \underbrace{I_n}_{\text{Integral}} + \underbrace{B_n}_{\text{Derivative (imperfect derivative)}} \}$
(See below for details about I_n, B_n)</td></tr>
<tr><td>In</td><td colspan="2">$I_n = I_{n-1} + \frac{CT}{Ti} \times DV_n$</td></tr>
<tr><td>Bn</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ (PV_n - PV_{n-1}) - \frac{CT \times B_{n-1}}{Td} \}$</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{ -(PV_n - PV_{n-1}) - \frac{CT \times B_{n-1}}{Td} \}$</td></tr>
</table>

Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Previous deviation, PVn: Process variable, PV_{n-1}: Previous process variable, SVn: Engineering value conversion processing result

(a) Integral term and derivative term are listed below corresponding to each condition.

<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>Td=0, or control mode being either MAN or CMV</td><td>Bn=0</td></tr>
<tr><td>Any of 1), 2), 3), 4)
1) Ti=0
2) When MH error occurs $\frac{CT}{Ti} \times DVn > 0$
3) When ML error occurs $\frac{CT}{Ti} \times DVn < 0$
4) Control mode is either MAN or CMV</td><td>$\frac{CT}{Ti} \times DVn = 0$</td></tr>
<tr><td>Any of 1), 2)
1) When tracking from the secondary loop
2) When the control mode is changed from MAN or CMV to any of AUT, CAS and CSV</td><td>$I_{n-1} = \frac{MV}{Kp} - (DV_n + B_n)$</td></tr>
</table>

Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit, ML: Output low limit

(b) Control cycle (CT) should be set to be the integral multiple of execution cycle (△T).

(c) Integral constant should be set to be 0.0 or over control cycle (CT).

(d) PIDP operation of the tag access FB is executed in every control cycle (CT), (output MV). For other execution cycle (△T), the last execution value of MV is held.

</td></tr>
<tr><td>Engineering value conversion</td><td>

When the control mode is CAS/CSV, the setting value (%) from the primary loop is converted into engineering value.

$SV = \frac{RH - RL}{100} \times$ Setting value (%) from the primary loop + RL

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

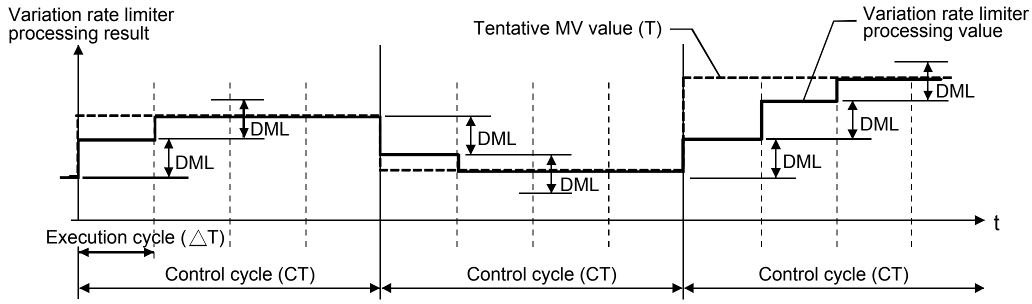
</td></tr>
<tr><td>Inverse engineering value conversion</td><td>

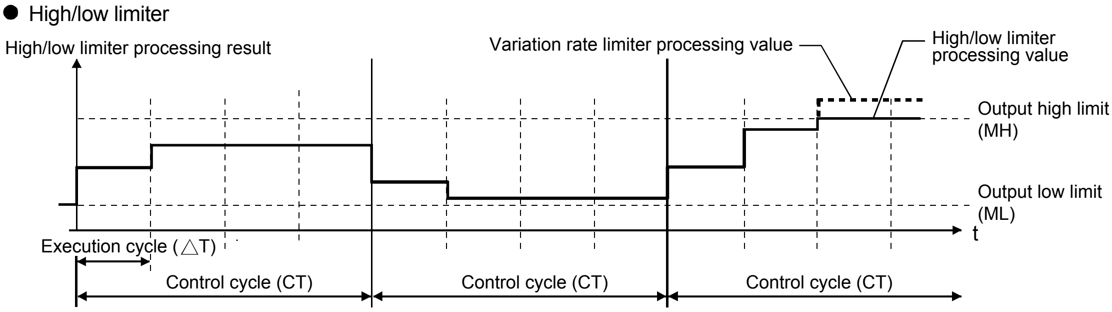
The setting value (SV) of engineering value is converted to percentage SV (%).

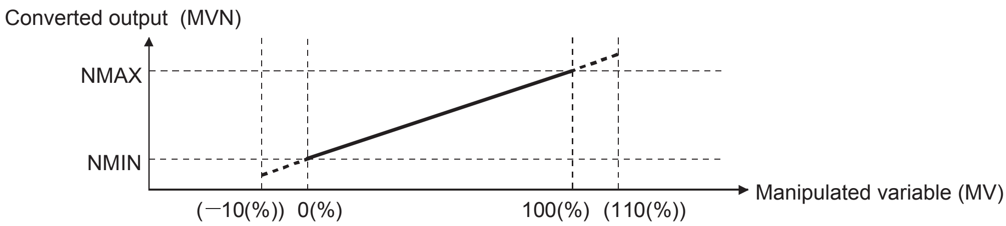
$SV (\%) = \frac{100}{RH - RL} \times (SV - RL)$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Variation rate and high/low limiter</td><td>
Execute variation rate check and high/low limit check to the input value.
● Variation rate limiter


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">Variation rate limiter processing result</th><th>Alarm (ALM)</th></tr>
<tr><th>Output variation rate limit (DMLA)</th></tr>
<tr><td>|T-MV| ≦ DML</td><td>T</td><td>FALSE (reset)</td></tr>
<tr><td>T-MV>DML</td><td>MV+DML</td><td>TRUE (occur)</td></tr>
<tr><td>T-MV<-DML</td><td>MV-DML</td><td>TRUE (occur)</td></tr>
</table>
T: Tentative MV value, MV: Manipulated variable, DML: Output variation rate high limit
● High/low limiter


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter processing result</th><th colspan="2">Alarm (ALM)</th></tr>
<tr><th>Output low limit (MLA)</th><th>Output high limit (MHA)</th></tr>
<tr><td>Variation rate limiter processing result > MH</td><td>MH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter processing result < ML</td><td>ML</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>ML≦ Variation rate limiter processing result ≦ MH</td><td>Variation rate limiter processing value</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
MH: Output high limit, ML: Output low limit
</td></tr>
<tr><td>Output conversion</td><td>
Execute output conversion procession.


<table><tr><td>Converted output (MVN) = $\{(\text{NMAX-NMIN}) \times \frac{\text{MV}}{100}\}$+NMIN</td></tr></table>
NMAX: Output conversion high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%),
MVN: Converted output value
</td></tr>
</table>

Item	Contents
Disable Alarm Detection	Set whether Enable Alarm Detection or not in variation rate limiter and high/low limiter. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected. ● ERRI, DMLI, MHI, MLI (2) Disable Alarm Detection by control mode: If the control mode is MAN and CMV, reset DMLA, MHA and MLA of alarm (ALM) and DMLA, MHA and MLA will be detected. (3) Disable Alarm Detection by loop stop processing: Please refer to loop stop processing in the following contents.

Other Function

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occurs. 4) Alarm is not detected in variation rate limiter and high/ low limiter.

Processing Operation

Processing / Control mode	Deviation check	PIDP operation	Engineering value conversion	Inverse engineering value conversion	Variation rate and high/low limiter	Output conversion	Alarm
MAN, CMV	○	○	×	○	×	○	× (*1)
AUT	○	○	×	○	○	○	○ (*2)
CAS, CSV	○	○	○	○	○	○	○ (*2)

○: Execute ×: Not execute

*1 An alarm (ALM) whose corresponding bit is TRUE (occurred) is reset, and the alarm cannot be detected.

*2 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Restriction

This FB supports the Process CPU and the Redundant CPU whose upper five digits of serial No.s are 07032 or later.

If either of the CPU whose upper five digits of serial No. are 07031 or earlier is used, the MV value bumpless switching and tracking processing from the secondary loop at control mode change will not be executed.

Error

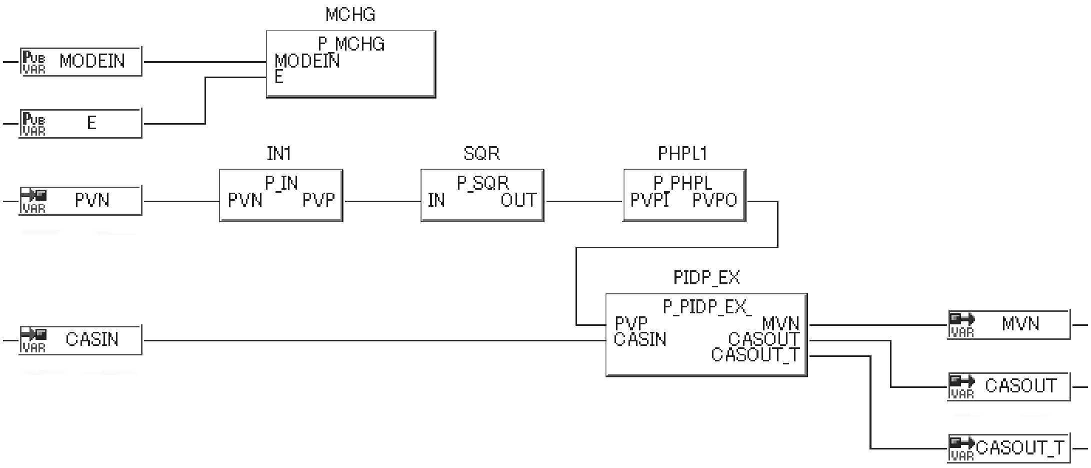
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.

Additionally, process control error code as well as the detailed error information, will be displayed on the screen.

For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

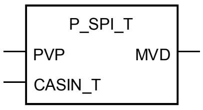
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.13 Sample PI Control (With Tracking to primary loop) (P_SPI_T)

FB	FBD parts
P_SPI_T	

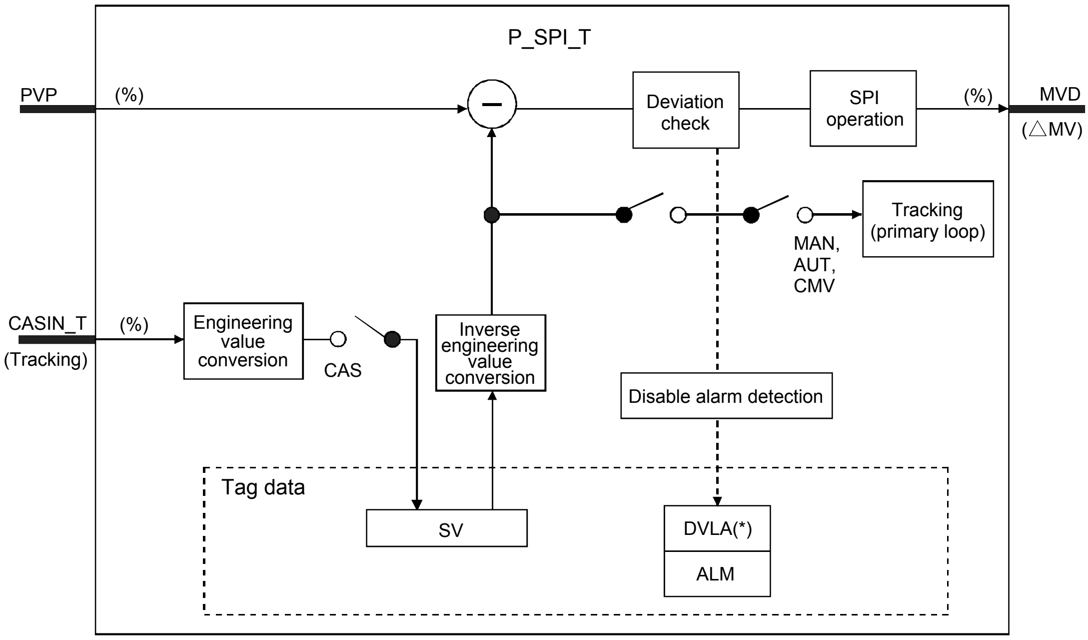
Corresponding tag type
SPI

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Execute PI control and output (ΔMV) during operating time (ST). During hold time (HD), hold the output (ΔMV=0).

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	Cascade SV input (unit: %) (With tracking)	0 to 100
Output	MVD	Output variable	REAL	ΔMV output (unit: %)	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not host MV, FALSE: Host MV)	TRUE, FALSE	TRUE	User

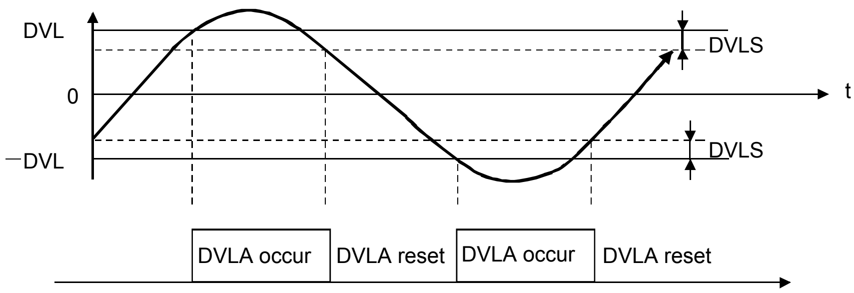
*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
Execute deviation check processing.



<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

Item	Contents
SPI operation	(see below)

SPI operation

(1) Gain (Kp) is calculated as below:

$$Kp = K \times P$$

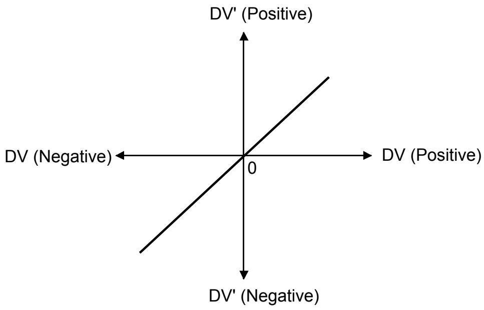
K: Output gain, P: Gain.

(2) Output gain (K) is calculated as below:

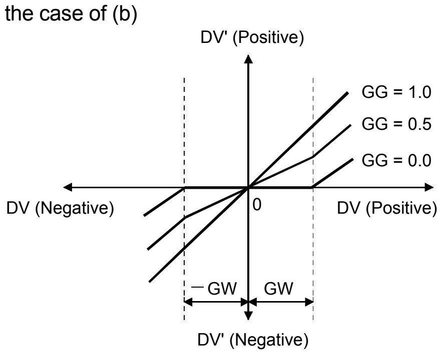
Condition		Output gain (K)
(a) When gap width (GW) = 0, the K value corresponding to the deviation (DV)		K=1
(b) When gap width (GW) > 0, the K value corresponding to the deviation (DV)	$\lvert DV \rvert \leqq GW$	K=GG
	$\lvert DV \rvert > GW$	$K = 1 - \frac{(1 - GG) \times GW}{\lvert DV \rvert}$

DV: Deviation (%), GW: Gap width (%)=Rate of gap width to deviation, GG: Gap gain

In the case of (a)



In the case of (b)



(3) Deviation for SPI operation (DV') is calculated as below.

Condition	Deviation for SPI operation (DV')
DV<-GW	DV'= - (GG × GW)+(DV+GW)
$\lvert DV \rvert \leqq GW$	DV'=GG × DV
DV>GW	DV'=GG × GW+(DV-GW)

DV': Deviation for SPI operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

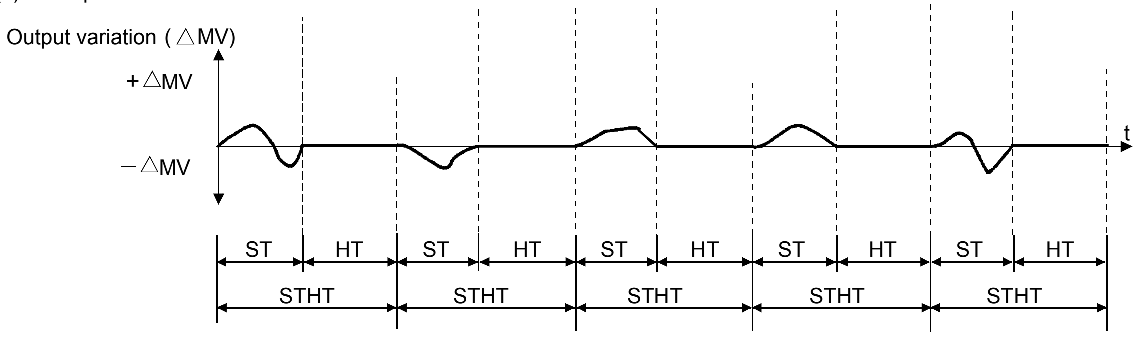
(4) Deviation for direct/reverse action (DV) is calculated as below.

Condition	Deviation (DV)
Direct action (PN=1)	DV (%)=PVP (%)-SV (%)
Reverse action (PN=0)	DV (%)=SV (%)-PVP (%)

DV: Deviation (%), PVP (%): PV input (%), SV (%) = $\frac{100}{RH-RL} \times (SV - RL)$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>SPI operation (continued)</td><td>
(5) SPI operation is executed as below.



ST: Operating time, STHT: Sample time, HT: Hold time (=STHT–ST)

<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>DVn=PVn-SVn</td><td>DVn=SVn-PVn</td></tr>
<tr><td>Output variation (△MV) during operating time (ST)</td><td colspan="2">$\Delta MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{(DV_n - DV_{n-1})}_{\text{Proportional}} + \underbrace{\frac{\Delta T}{Ti} \times DV_n}_{\text{Integral}} \}$</td></tr>
<tr><td>Output variation (△MV) during hold time (HT=STHT-ST)</td><td colspan="2">△MV=0</td></tr>
</table>

Kp: Gain, Ti: Integral time, DVn: Deviation, DVn-1: Previous deviation, PVn: Process variable, SVn: Engineering value conversion processing result, △T: Execution cycle, ST: Operating time, STHT: Sample time, HT: Hold time (=STHT–ST)

(a) Integral items are listed below corresponding to each condition.

<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>Any of 1), 2), 3)
1) Ti=0
2) When either MH or ML error occurs, MVP>MH and $\frac{\Delta T}{Ti} \times DVn > 0$
3) When either MH or ML error occurs, MVP<ML and $\frac{\Delta T}{Ti} \times DVn < 0$</td><td>$\frac{\Delta T}{Ti} \times DVn = 0$</td></tr>
</table>

Ti: Integral time, △T: Control cycle, DVn: Deviation, MH: Output high limit, ML: Output low limit, MVP: MV internal operation value

(b) In case of $\frac{STHT}{\Delta T} \leqq \frac{ST}{\Delta T}$, considering hold time (HT) = 0, PI control is continuously carried out
</td></tr>
<tr><td>Engineering value conversion</td><td>
When the control mode is CAS/CSV, the setting value (%) from the primary loop is converted into engineering value.

$SV = \frac{RH-RL}{100} \times$ Setting value (%) from the primary loop + RL

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Inverse engineering value conversion</td><td>
The setting value (SV) of engineering value is converted to percentage SV (%).

$SV (\%) = \frac{100}{RH-RL} \times (SV-RL)$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Tracking processing</td><td>Whether execute tracking processing to the input variable CASIN_T is described in the following table:
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag</th><th>Setting value (SV) used (SVPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table>
</td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether Enable Alarm Detection or not in deviation check.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DVLA will not be detected.
● ERRI, DVLI
(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
</table>

Other Functions

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) is TRUE. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Reset DVLA when DVLA of alarm (ALM) occurs. 4) Alarm is not detected in deviation check.

Processing Operation

Processing / Control mode	Deviation check	SPI operation	Engineering value conversion	Inverse engineering value conversion	Tracking	Alarm
MAN, CMV, AUT	○	○	×	○	○ (*1)	○ (*2)
CAS, CSV	○	○	○	○	×	○ (*2)

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

*2 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

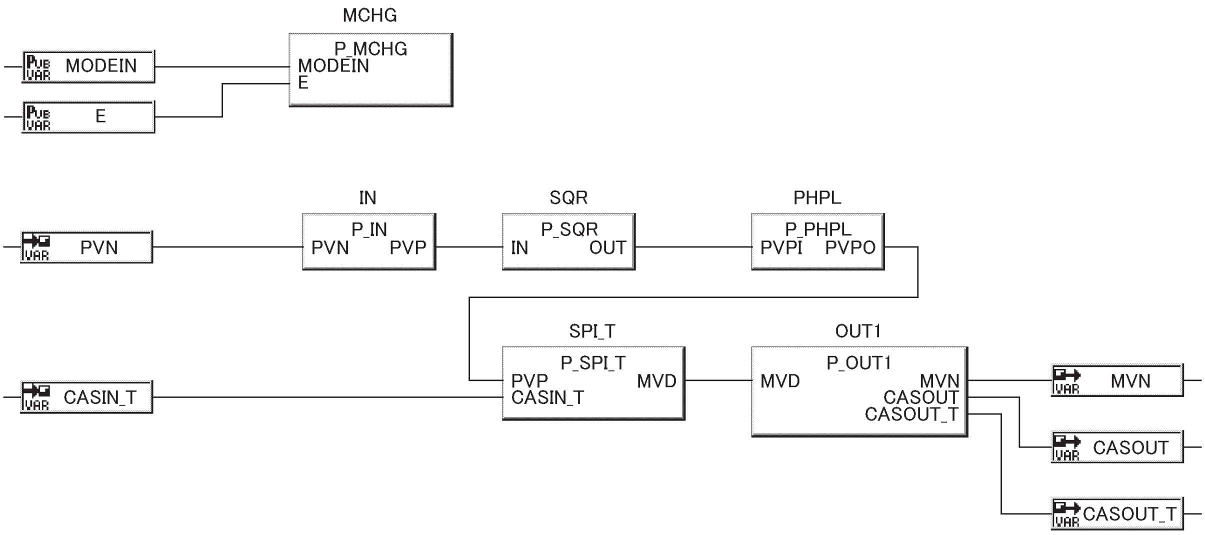
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.

Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

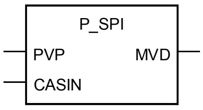
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.14 Sample PI Control (Without Tracking to primary loop) (P_SPI)

FB	FBD parts
P_SPI	

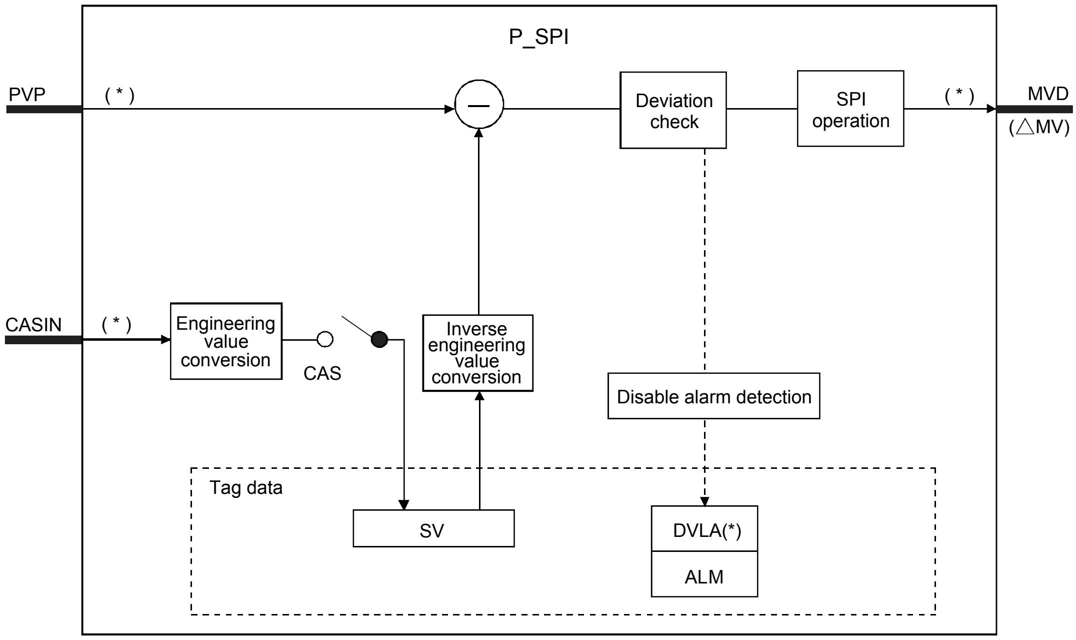
Corresponding tag type
SPI

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: Execute PI control and output (△MV) during operating time (ST). During hold time (HD), hold the output (△MV=0).

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pins	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVD	Output variable	REAL	△MV output (unit: %)	-999999 to 999999

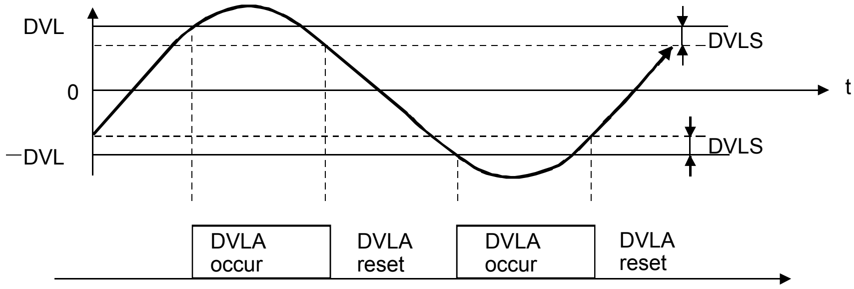
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
Execute the deviation check processing.


<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL < |DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>
DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>SPI operation</td><td>

(1) Gain (Kp) is calculated as below:

$$Kp = K \times P$$

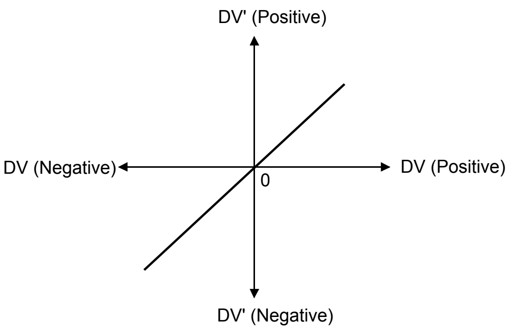
K: Output gain, P: Gain

(2) Output gain (K) is calculated as below:

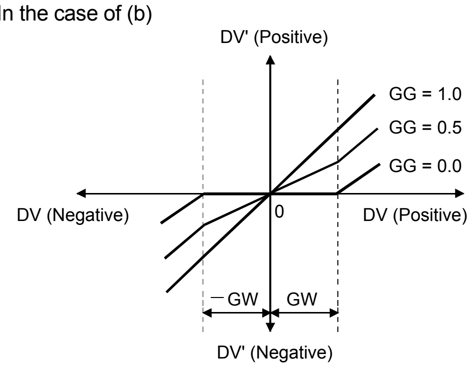
<table>
<tr><th colspan="2">Condition</th><th>Output gain (K)</th></tr>
<tr><td colspan="2">(a) When gap width (GW) = 0, the K value corresponding to the deviation (DV)</td><td>K=1</td></tr>
<tr><td rowspan="2">(b) When gap width (GW) > 0, the K value corresponding to the deviation (DV)</td><td>|DV| ≦ GW</td><td>K=GG</td></tr>
<tr><td>|DV| > GW</td><td>$K = 1 - \frac{(1 - GG) \times GW}{|DV|}$</td></tr>
</table>

DV: Deviation (%), GW: Gap width (%)=Rate of gap width to deviation, GG: Gap gain

In the case of (a)



In the case of (b)



(3) Deviation for SPI operation (DV') is calculated as below.

<table>
<tr><th>Condition</th><th>Deviation for SPI operation (DV')</th></tr>
<tr><td>DV<-GW</td><td>DV'= - (GG × GW)+(DV+GW)</td></tr>
<tr><td>|DV| ≦ GW</td><td>DV'=GG × DV</td></tr>
<tr><td>DV>GW</td><td>DV'=GG × GW +(DV-GW)</td></tr>
</table>

DV': Deviation for SPI operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as below.

<table>
<tr><th>Condition</th><th>Deviation (DV)</th></tr>
<tr><td>Direct action (PN=1)</td><td>DV (%) = PVP (%) – SV (%)</td></tr>
<tr><td>Reverse action (PN=0)</td><td>DV (%) = SV (%) – PVP (%)</td></tr>
</table>

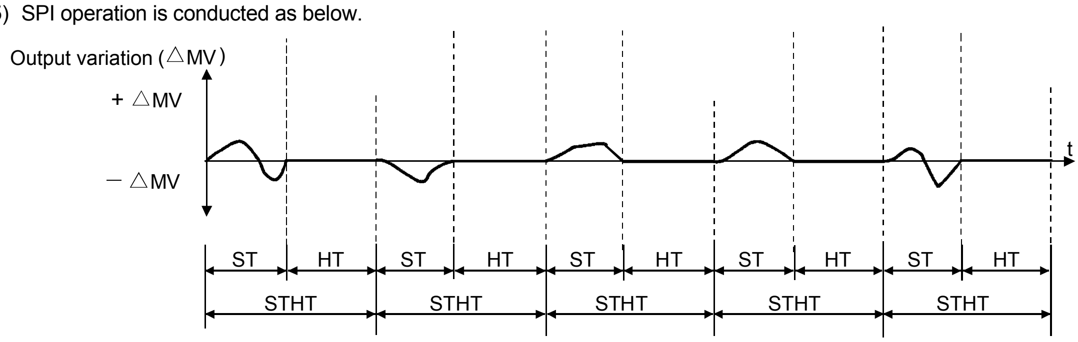
DV: Deviation (%), PVP (%): PV input (%), SV (%) = $\frac{100}{RH-RL} \times (SV - RL)$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

</td></tr>
</table>

Item	Contents
SPI operation (continued)	(5) SPI operation is conducted as below. (see below)

(5) SPI operation is conducted as below.



ST: Operation time, STHT: Sample period, HT: Hold time(STHT−ST)

	Direct action	Reverse action
Deviation (DVn)	$DV_n = PV_n - SV_n$	$DV_n = SV_n - PV_n$
Output variation (△MV) during operation time (ST)	$\triangle MV = \underbrace{Kp}_{\text{Gain}} \times \{\underbrace{(DV_n - DV_{n-1})}_{\text{Proportional}} + \underbrace{\frac{\triangle T}{Ti} \times DV_n}_{\text{Integral}}\}$	
Output variation (△MV) during hold time (HT=STHT-ST)	$\triangle MV = 0$	

Kp: Gain, Ti: Integral time, DVn: Deviation, DV_{n-1}: Previous deviation, PVn: Process variable, SVn: Engineering value conversion processing result, △T: Execution cycle, ST: Operating time, STHT: Sample time, HT: Hold time (=STHT–ST)

(a) Integral item is listed below corresponding to each condition.

Condition	Processing
Any of 1), 2), 3) 1) Ti=0 2) When either MH or ML error occurs, MVP>MH and $\frac{\triangle T}{Ti} \times DVn > 0$ 3) When either MH or ML error occurs, MVP<ML and $\frac{\triangle T}{Ti} \times DVn < 0$	$\frac{\triangle T}{Ti} \times DVn = 0$

Ti: Integral time, △T: Control cycle, DVn: Deviation, MH: Output high limit, ML: Output low limit, MVP: MV internal operation value

(b) In case of $\frac{STHT}{\Delta T} \leqq \frac{ST}{\Delta T}$, considering hold time (HT) = 0, PI control is continuously carried out.

Engineering value conversion

When the control mode is CAS/CSV, the setting value (%) from the primary loop is converted into engineering value.

$$SV = \frac{RH-RL}{100} \times \text{Setting value (\%) from the primary loop} + RL$$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

Inverse engineering value conversion

The setting value (SV) of engineering value is converted to percentage SV (%).

$$SV(\%) = \frac{100}{RH-RL} \times (SV - RL)$$

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

Disable Alarm Detection

Set whether Enable Alarm Detection or not in deviation check.

(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DVLA will not be detected.
- ERRI, DVLI

(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the next page.

Other Functions

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) is TRUE. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Reset DVLA when DVLA of alarm (ALM) occurs. 4) Alarm is not detected in deviation check.

Processing Operation

Control mode \ Processing	Deviation check	SPI operation	Engineering value conversion	Inverse engineering value conversion	Alarm
MAN, CMV, AUT	○	○	×	○	○ (*1)
CAS, CSV	○	○	○	○	○ (*1)

○: Execute ×: Not execute

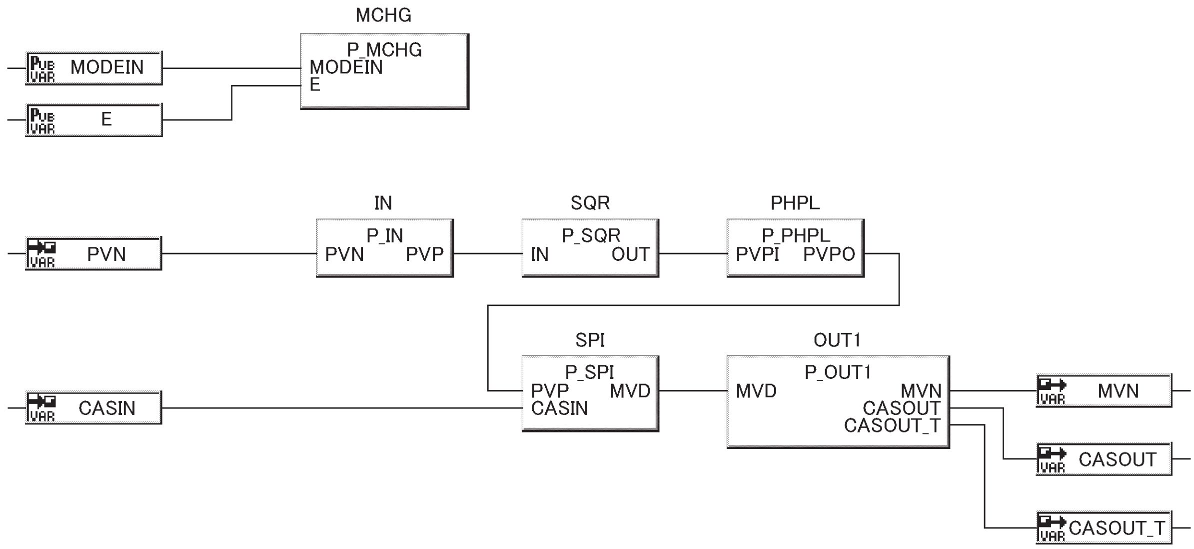
*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

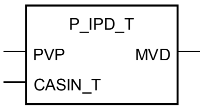
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.15 I-PD Control (With Tracking to primary loop) (P_IPD_T)

FB	FBD parts
P_IPD_T	

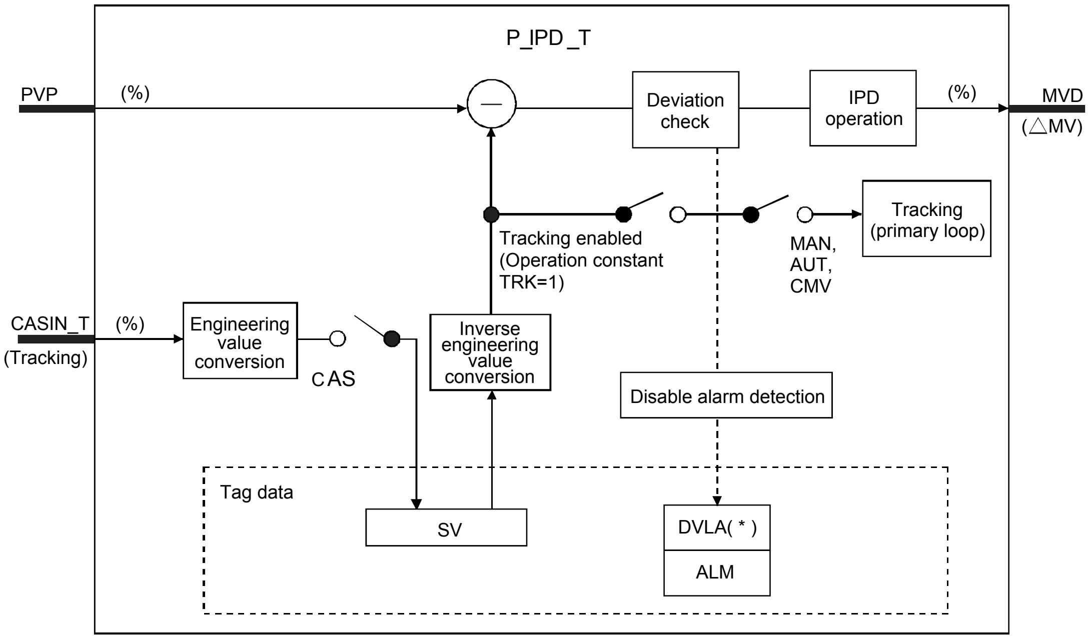
Corresponding tag type
IPD

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: In the I-PD control, as process variable is used in proportional and derivative terms, a step change in the set point does not result in shock in the output and enable slow response. (△MV).

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	Cascade SV input (unit: %) (With tracking)	0 to 100
Output	MVD	Output variable	REAL	ΔMV output (unit: %)	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not host MV, FALSE: Host MV)	TRUE, FALSE	TRUE	User

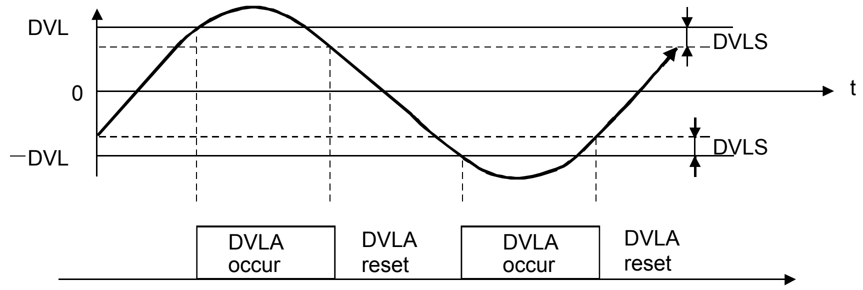
*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
Execute deviation check processing.



<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL- DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

Item	Contents
IPD operation	See below

(1) Gain (Kp) is calculated as below:

$Kp = K \times P$

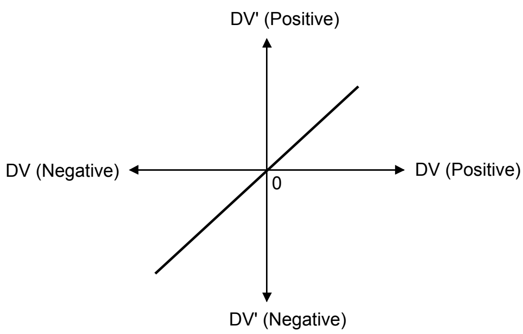
K: Output gain, P: Gain

(2) Output gain (K) is calculated as below:

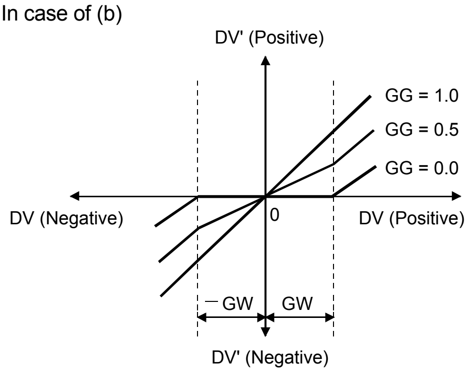
Condition		Output gain (K)
(a) When gap width (GW) = 0, the K value corresponding to the deviation (DV)		K=1
(b) When gap width (GW) > 0, the K value corresponding to the deviation (DV)	$\lvert DV\rvert \leqq GW$	K=GG
	$\lvert DV\rvert > GW$	$K = 1 - \frac{(1 - GG) \times GW}{\lvert DV \rvert}$

DV: Deviation (%), GW: Gap width (%)=Rate of gap width to deviation, GG: Gap gain

In case of (a)



In case of (b)



(3) Deviation for IPD operation (DV') is calculated as below.

Condition	Deviation for IPD operation (DV')
DV<-GW	DV'= - (GG × GW)+(DV+GW)
|DV| ≦ GW	DV'=GG × DV
DV>GW	DV'=GG × GW +(DV-GW)

DV': Deviation for IPD operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for positive/reverse operation (DV) is calculated as below.

Condition	Deviation (DV)
Direct action (PN=1)	DV (%)=PVP (%)- SV (%)
Reverse action (PN=0)	DV (%)=SV (%)- PVP (%)

DV: Deviation (%), PVP (%): PV input (%), SV (%) = $\frac{100}{RH-RL} \times (SV-RL)$,

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>IPD operation (continued)</td><td>
(5) IPD operation is executed as below.
<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (△MV)</td><td>$\triangle MV = \underbrace{Kp}_{Gain} \times \{\underbrace{\frac{CT}{Ti} \times DV_n}_{Integral} + \underbrace{(PV_n - PV_{n-1})}_{Proportional} + \underbrace{B_n}_{Derivative}\}$</td><td>$\triangle MV = \underbrace{Kp}_{Gain} \times \{\underbrace{\frac{CT}{Ti} \times DV_n}_{Integral} - \underbrace{(PV_n - PV_{n-1})}_{Proportional} + \underbrace{B_n}_{Derivative}\}$</td></tr>
<tr><td>Bn</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times B_{n-1}}{Td}\}$</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{-(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times B_{n-1}}{Td}\}$</td></tr>
</table>
Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Previous deviation, PVn: Process variable, PV_{n-1}: Previous process variable, PV_{n-2}: Process variable before last, SVn: Engineering value conversion processing result
(a) Integral item and derivative term are listed below corresponding to each condition.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>Td=0, or control mode being either MAN or CMV</td><td>Bn=0</td></tr>
<tr><td>Any of 1), 2), 3)
1) Ti=0
2) When either MH or ML error occurs, MVP>MH and $\frac{CT}{Ti} \times DVn > 0$
3) When either MH or ML error occurs, MVP<ML and $\frac{CT}{Ti} \times DVn < 0$</td><td>$\frac{CT}{Ti} \times DVn = 0$</td></tr>
</table>
Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit, ML: Output low limit, MVP: MV internal operation value
(b) Control cycle (CT) should be set to be the integral multiple of execution cycle (ΔT)
(c) Integral constant should be set to be 0.0 or over control cycle (CT).
(d) PID operation of the tag access FB is executed in every control cycle (CT) period, (output △MV). For other execution cycle (△T), the last execution value is held (△MV=0)
</td></tr>
<tr><td>Engineering value conversion</td><td>
When the control mode is CAS/CSV, the setting value (%) from the primary loop is converted to engineering value.
$SV = \frac{RH-RL}{100} \times$ Setting value (%) from the primary loop + RL
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Inverse engineering value conversion</td><td>
The setting value (SV) of engineering value is converted to percentage SV (%).
$SV (\%) = \frac{100}{RH-RL} \times (SV - RL)$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Tracking processing</td><td>Whether execute tracking processing to input variable CASIN_T.
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag</th><th>Setting value (SV) used (SVPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table></td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether Enable Alarm Detection or not in variation rate limiter and high/low limiter.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DVLA will not be detected.
● ERRI, DVLI
(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
</table>

Other Functions

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) is TRUE. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Reset DVLA when DVLA of alarm (ALM) occurs. 4) Alarm is not detected in deviation check.

Processing Operation

Processing / Control mode	Deviation check	IPD operation	Engineering value conversion	Inverse engineering value conversion	Tracking	Alarm
MAN, CMV, AUT	○	○	×	○	○ (*1)	○ (*2)
CAS, CSV	○	○	○	○	×	○ (*2)

○ : Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

*2 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

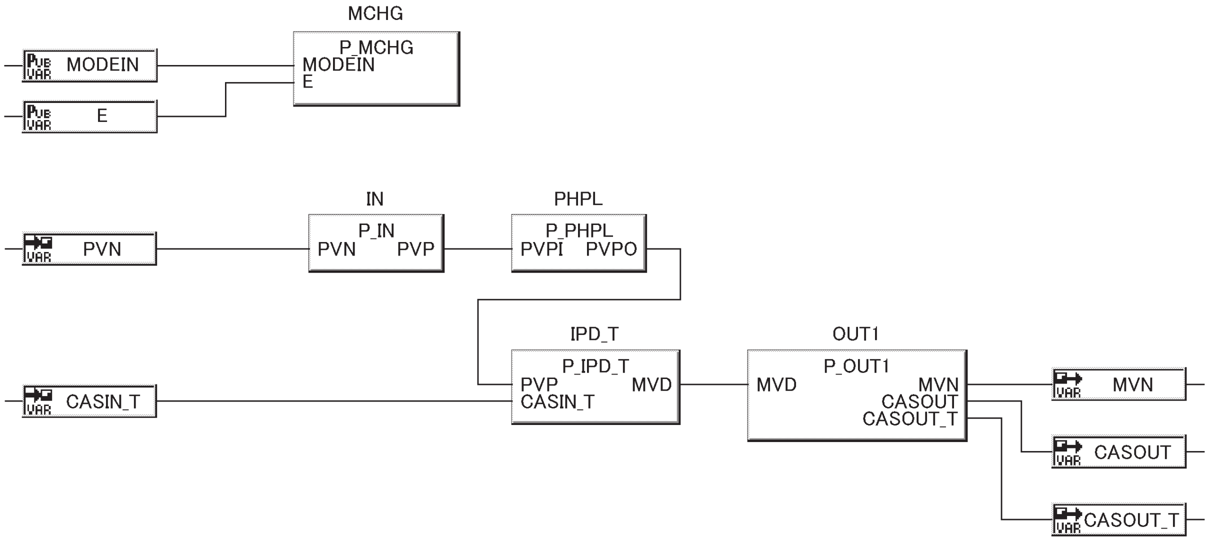
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.

Additionally, process control error code as well as the detailed error information, will be displayed on the screen.

For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

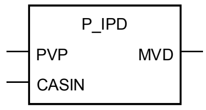
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.16 I-PD Control (Without Tracking to primary loop) (P_IPD)

FB	FBD parts
P_IPD	

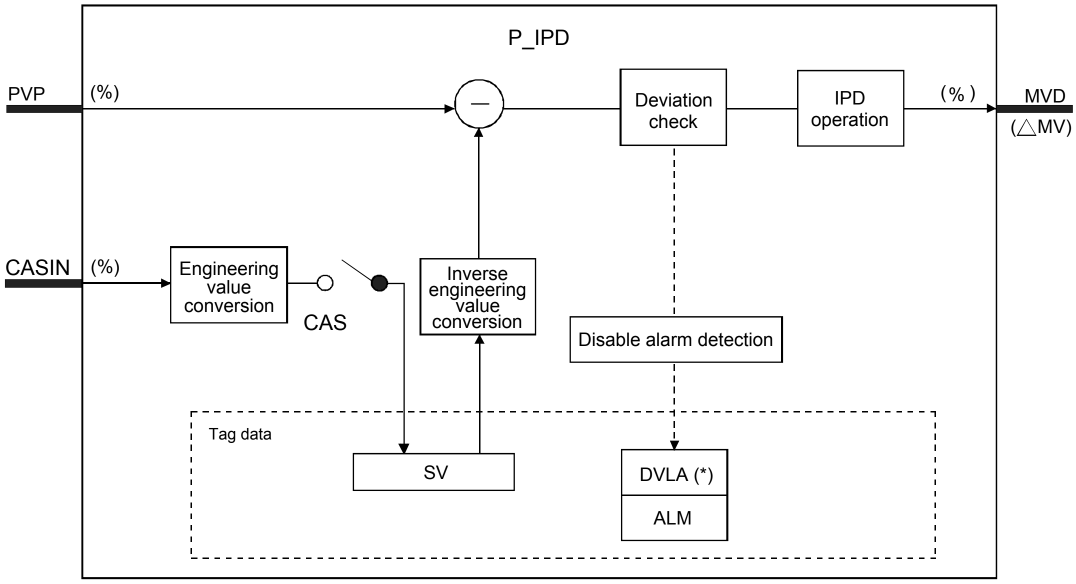
Corresponding tag type
IPD

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: In the I-PD control, as process variable is used in proportional and derivative terms, a step change in the set point does not result in shock in the output and enable slow response. (△MV).

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVD	Output variable	REAL	△MV output (unit: %)	-999999 to 999999

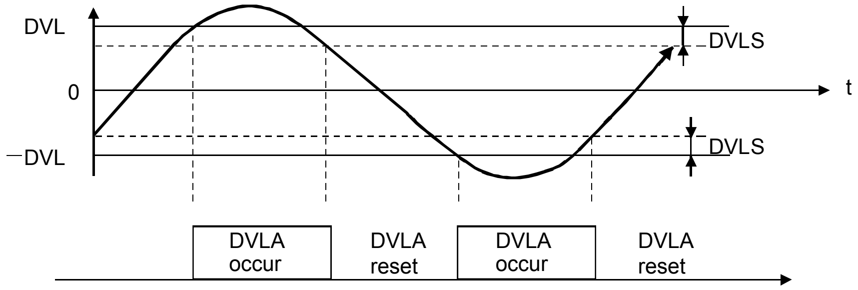
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>Execute deviation check processing.


<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL < |DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>
DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)</td></tr>
</table>

Item	Contents
IPD operation	See below

(1) Gain (Kp) is calculated as follows:

$$Kp = K \times P$$

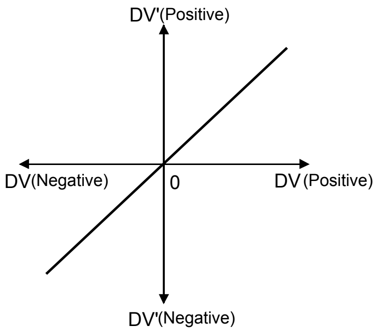
K: Output gain, P: Gain

(2) Output gain (K) is calculated as follows:

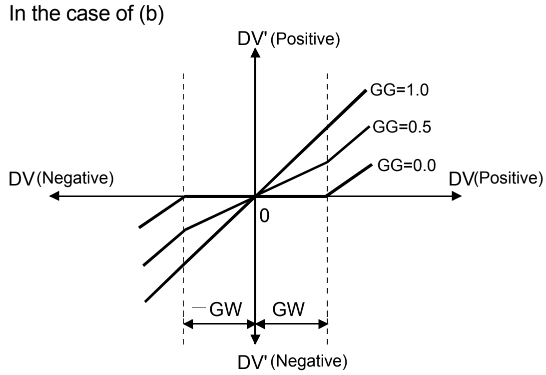
Condition		Output gain (K)
(a) When gap width (GW) = 0, the value of K corresponding to deviation (DV).		K=1
(b) When gap width (GW) > 0, the value of K corresponding to deviation (DV).	$\lvert DV \rvert \leqq GW$	K=GG
	$\lvert DV \rvert > GW$	$K = 1 - \frac{(1 - GG) \times GW}{\lvert DV \rvert}$

DV: Deviation (%), GW: Gap width (%)=Rate of gap width corresponding to deviation, GG: Gap gain

In the case of (a)



In the case of (b)



(3) Deviation for IPD operation (DV') is calculated as follows.

Condition	Deviation for IPD operation (DV')
DV < -GW	DV'= - (GG × GW)+(DV+GW)
$\lvert DV \rvert \leqq GW$	DV'=GG × DV
DV > GW	DV'=GG × GW+(DV−GW)

DV': Deviation for IPD operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as follows.

Condition	Deviation (DV)
Direct action (PN=1)	DV (%) = PVP (%) – SV (%)
Reverse action (PN=0)	DV (%) = SV (%) – PVP (%)

DV: Deviation (%), PVP (%): PV input value (%), SV (%) = $\frac{100}{RH-RL} \times (SV-RL)$,

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>IPD operation (continued)</td><td>
(5) IPD operation is executed as follows.
<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (△MV)</td><td>$\triangle MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{\frac{CT}{Ti} \times DV_n}_{\text{Integral}} + \underbrace{(PV_n - PV_{n-1})}_{\text{Proportional}} + \underbrace{B_n}_{\text{Derivative}} \}$</td><td>$\triangle MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{\frac{CT}{Ti} \times DV_n}_{\text{Integral}} - \underbrace{(PV_n - PV_{n-1})}_{\text{Proportional}} + \underbrace{B_n}_{\text{Derivative}} \}$</td></tr>
<tr><td>Bn</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times B_{n-1}}{Td}\}$</td><td>$B_n = B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{-(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times B_{n-1}}{Td}\}$</td></tr>
</table>
Kp: Gain, Ti: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DVn-1: Previous deviation, PVn: Process variable, PVn-1: Previous process variable, PVn-2: The process variable before last, SVn: The processing result of engineering value conversion
(a) Integral items and derivative items are listed below corresponding to each condition.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>When Td=0, or control mode is either MAN or CMV</td><td>Bn=0</td></tr>
<tr><td>Any of 1), 2), 3)
1) Ti=0
2) When either MH or ML error occurs, MVP > MH and $\frac{CT}{Ti} \times DV_n > 0$
3) When either MH or ML error occurs, MVP < ML and $\frac{CT}{Ti} \times DV_n < 0$</td><td>$\frac{CT}{Ti} \times DV_n = 0$</td></tr>
</table>
Ti: Integral time, CT: Control cycle, DVn: Deviation, MH: Output high limit value, ML: Output low limit value, MVP: MV internal operation value
(b) Control cycle (CT) shall be set to be the integral multiple of execution cycle (△T).
(c) Integral constant shall be set to be 0.0 or over control cycle (CT).
(d) PID operation of the tag access FB is executed in each control cycle (CT) (△MV output).
For other execution cycles (△T), hold the previous value. (△MV=0)
</td></tr>
<tr><td>Engineering value conversion</td><td>
When the control mode is CAS/CSV, the setting value (%) from the primary loop is converted to engineering value.
$SV = \frac{RH-RL}{100} \times \text{Setting value (\%) from the primary loop} + RL$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Inverse engineering value conversion</td><td>
The setting value (SV) of engineering value is converted to percentage SV (%).
$SV (\%) = \frac{100}{RH-RL} \times (SV-RL)$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Disable Alarm Detection</td><td>
Set whether Enable Alarm Detection or not in deviation check.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DVLA will not be detected.
● ERRI, DVLI
(2) "Disable Alarm Detection" by loop stop processing:
Please refer to loop stop processing in the next page.
</td></tr>
</table>

Other Functions

Item	Content
Loop stop processing	The following processing is executed if the stop alarm (SPA) is TRUE. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Reset DLVA when DLVA of alarm (ALM) occurs. 4) Alarm is not detected in deviation check.

Processing Operation

Control mode \ Processing	Deviation check	IPD operation	Engineering value conversion	Inverse engineering value conversion	Alarm
MAN, CMV, AUT	○	○	×	○	○ (*1)
CAS, CSV	○	○	○	○	○ (*1)

○: Execute ×: Not execute

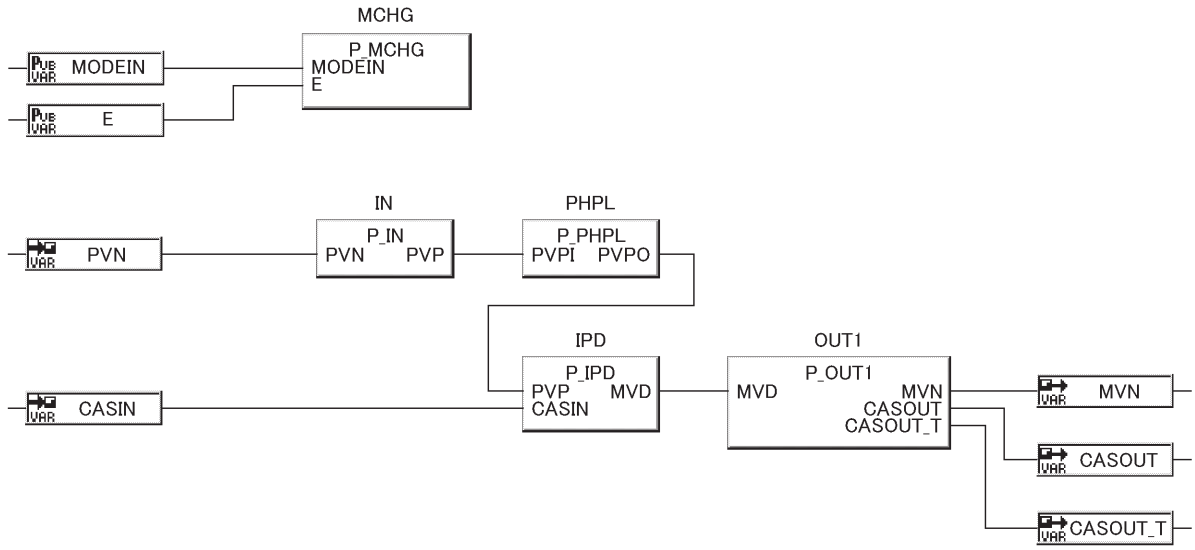
*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

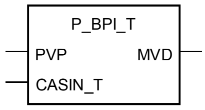
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



> **POINT**
> It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.17 Blend PI Control (With Tracking to primary loop) (P_BPI_T)

FB	FBD parts
P_BPI_T	

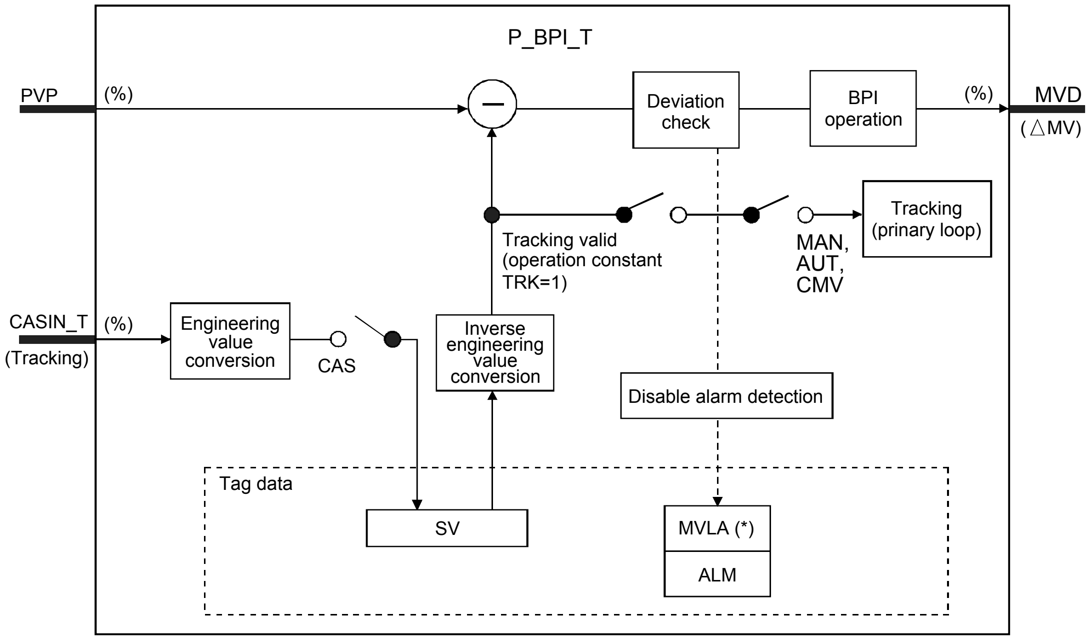
Corresponding tag type
BPI

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: The control when the control volume is stable during a long period even if it vibrates in a short period.

Function/FB classification name: Tag access FB _ Loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Content	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	Cascade SV input (unit: %) (With tracking)	0 to 100
Output	MVD	Output variable	REAL	△MV output (unit: %)	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Content	Range	Initial value	Storage
Operation processing	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Use)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TRUE	User
	RST_SDV_ON_CHGMODE	Public variable	BOOL	DV cumulative value reset in control mode change TRUE: Resets DV cumulative value (SDV) in control mode change (MAN/CMV → AUT/CAS/CSV) FALSE: Not reset DV cumulative value (SDV)	TRUE, FALSE	FALSE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variables (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	RST_SDV	Public variable	BOOL	DV cumulative value reset FALSE → TRUE: DV cumulative value (SDV) reset	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

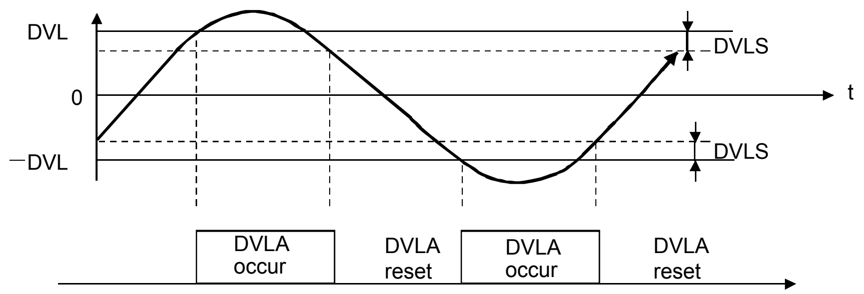
Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>

Execute deviation check processing.



<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL < |DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

Item	Contents

BPI operation

(1) Gain (Kp) is calculated as follows:

$$Kp=K \times P$$

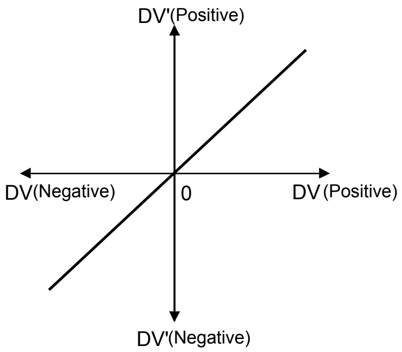
K: Output gain, P: Gain

(2) Output gain (K) is calculated as follows:

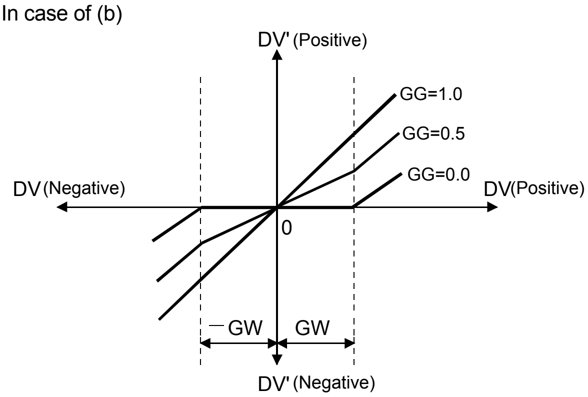
Condition		Output gain (K)
(a) When gap width (GW) = 0, the value of K corresponding to deviation (DV).		K=1
(b) When gap width (GW) > 0, the value of K corresponding to deviation (DV).	$\lvert DV\rvert \leqq GW$	K=GG
	$\lvert DV\rvert > GW$	$K=1-\frac{(1-GG)\times GW}{\lvert DV\rvert}$

DV: Deviation (%), GW: Gap width (%)=Rate of gap width corresponding to deviation, GG: Gap gain

In case of (a)



In case of (b)



(3) Deviation for BPI operation (DV') is calculated as follows.

Condition	Deviation for BPI operation (DV')
DV < - GW	DV'= - (GG × GW)+(DV+GW)
$\lvert DV\rvert \leqq GW$	DV'=GG × DV
DV > GW	DV'=GG × GW+(DV–GW)

DV': Deviation for BPI operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation for direct/reverse action (DV) is calculated as follows.

Condition	Deviation (DV)
Direct action (PN=1)	DV (%) = PVP (%) –SV (%)
Reverse action (PN=0)	DV (%) = SV (%) –PVP (%)

DV: Deviation (%), PVP (%): PV input value (%), SV (%) = $\frac{100}{RH-RL} \times$ (SV-RL),

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>BPI operation (continued)</td><td>
(5) BPI operation is executed as follows.
<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (△MV)</td><td colspan="2">$\triangle MV = Kp \times \triangle T \times \{DV_n + \frac{CT}{Ti} \times \Sigma DV_i\}$</td></tr>
</table>
Kp: Gain, Ti: Integral time, △T: Execution cycle, CT: Control cycle, Σ DVi: The cumulative value of DVn, DVn: Deviation, PVn: Process variable, SVn: The processing result of engineering value conversion
(a) Integral item and derivative item are listed below corresponding to each condition.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>Either of 1), 2).
1) Ti=0
2) MLA or MHA is TRUE</td><td>$\frac{CT}{Ti} \times \Sigma DV_i$ = Hold the previous value</td></tr>
<tr><td>Ti ≠ 0</td><td>$\frac{CT}{Ti} \times \Sigma DV_i = \frac{CT}{Ti} \times (\Sigma DV_i + DV_n)$</td></tr>
</table>
Ti: Integral time, CT: Control cycle, Σ Dvi: The cumulative value of DVn, DVn: Deviation, MLA: Output low limit value, MHA: Output high limit value
(b) Control cycle (CT) should be set to be integral multiple of execution cycle (△T).
(c) Integral constant should be set to be 0.0 or over control cycle (CT).
(d) PID operation of the tag access FB is executed in every control cycle (CT), (△MV output).
For other execution cycle (△T), the previous value shall be held. (△MV=0)
</td></tr>
<tr><td>Engineering value conversion</td><td>
Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.
$SV = \frac{RH-RL}{100} \times$ Setting value (%) from the primary loop +RL
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Inverse engineering value conversion</td><td>
Convert the setting value (SV) of engineering value to percentage SV (%).
$SV (\%) = \frac{100}{RH-RL} \times (SV-RL)$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
</td></tr>
<tr><td>Tracking processing</td><td>
Tracking operation for input variable CASIN_T is executed as follows:
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SCPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table>
</td></tr>
<tr><td>Disable Alarm Detection</td><td>
Set whether Enable Alarm Detection or not in deviation check.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DVLA will not be detected.
● ERRI, DVLI
(2) "Disable Alarm Detection" by loop stop processing:
Please refer to loop stop processing in the next page.
</td></tr>
</table>

Other Functions

Item	Content
Loop stop processing	The following processing is executed if the stop alarm (SPA) is TRUE. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Reset DLVA when DLVA of alarm (ALM) occurs. 4) Alarm is not detected in deviation check.

Processing Operation

Control mode \ Processing	Deviation check	BPI operation	Engineering value conversion	Inverse engineering value conversion	Tracking	Alarm
MAN, CMV, AUT	○	○	×	○	○ (*1)	○ (*2)
CAS, CSV	○	○	○	○	×	○ (*2)

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

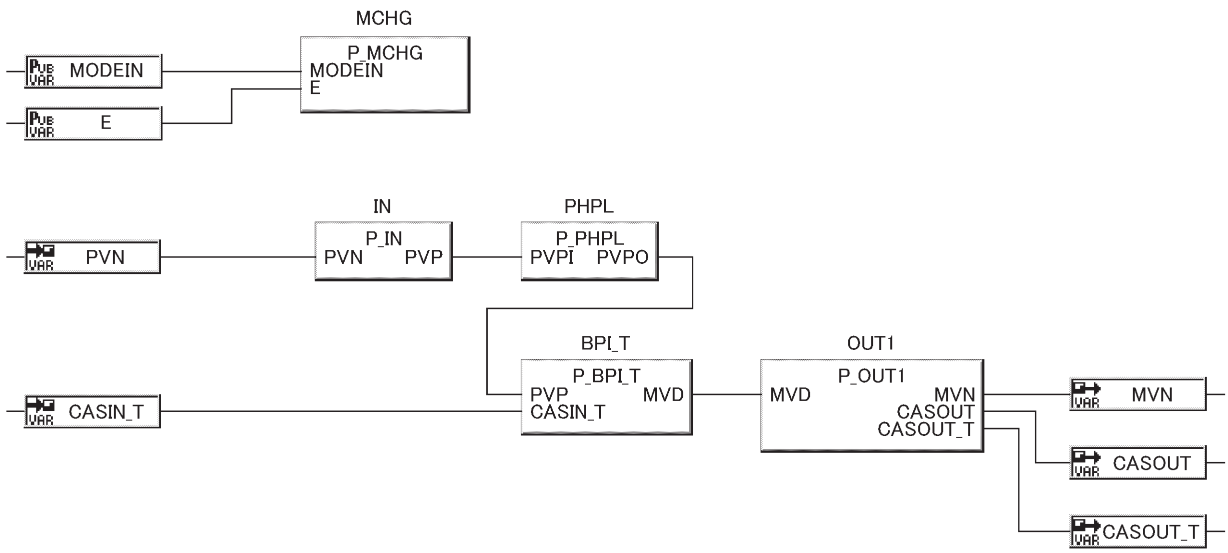
*2 When the bit of "Disable Alarm Detection" (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

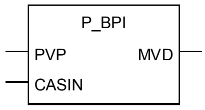
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



> **POINT**
>
> It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.18 Blend PI Control (Without Tracking to primary loop) (P_BPI)

FB	FBD parts
P_BPI	

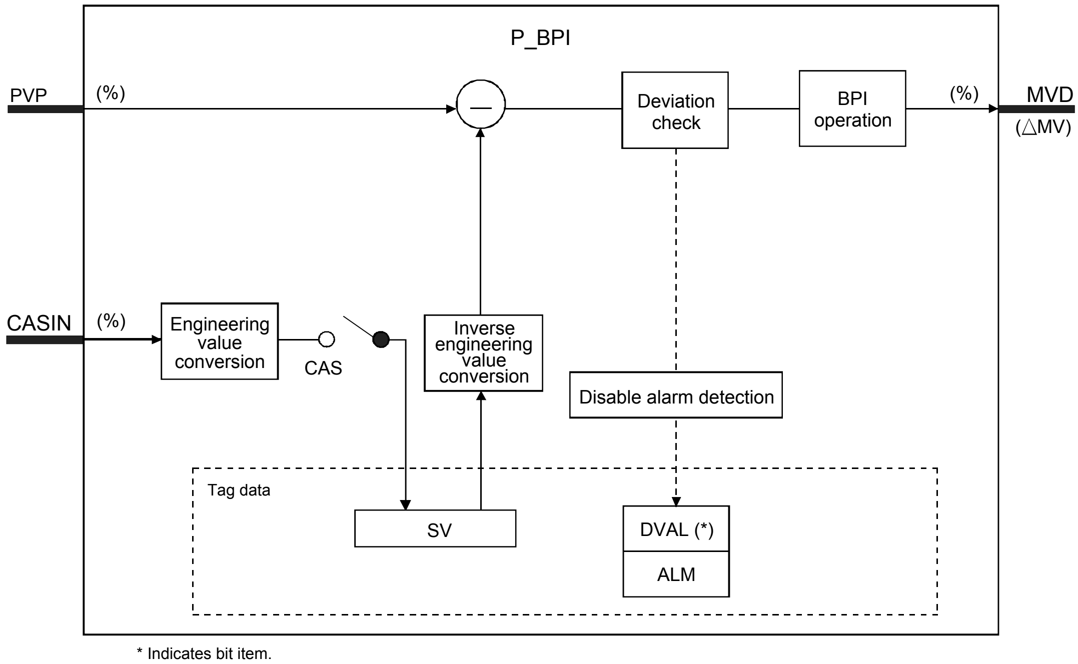
Corresponding tag type
BPI

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Functions overview: The control when the control volume is stable during a long period even if it vibrates in a short period.

Function/FB classification: Tag access FB _ loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Content	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVD	Output variable	REAL	△MV output (unit: %)	-999999 to 999999

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Content	Range	Initial value	Storage
Operation processing	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action・direct action (0: Reverse action, 1:Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	RST_SDV_ON_CHGMODE	Public variable	BOOL	DV cumulative value reset in control mode change TRUE: Resets DV cumulative value (SDV) in control mode change (MAN/CMV → AUT/CAS/CSV) FALSE: Not reset DV cumulative value (SDV)	TRUE, FALSE	FALSE	User

Public Variables (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	RST_SDV	Public variable	BOOL	DV cumulative value reset FALSE → TRUE: DV cumulative value (SDV) reset	TRUE, FALSE	FALSE	User

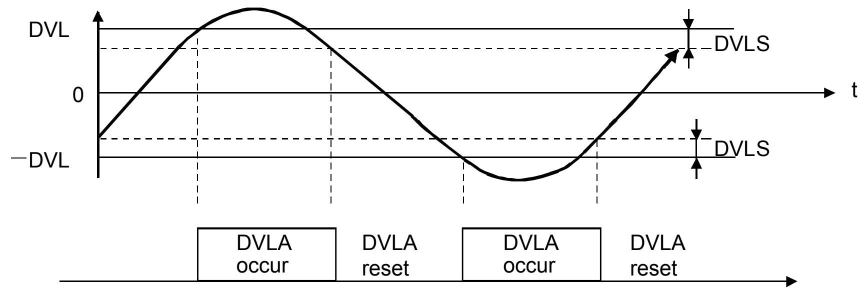
*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
Execute deviation check processing.



<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL < |DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

Item	Contents
BPI operation	

(1) Gain (Kp) is calculated as follows:

$$Kp=K \times P$$

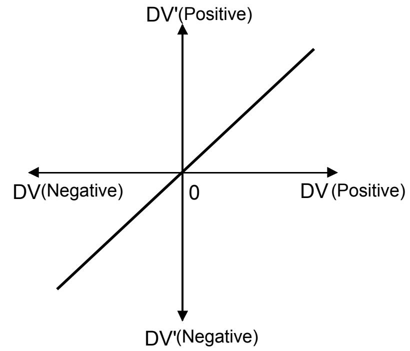
K: Output gain, P: Gain

(2) Output gain (K) is calculated as follows:

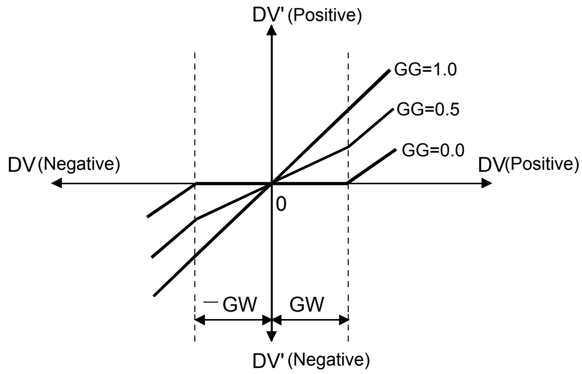
Condition		Output gain (K)
(a) When gap width (GW) = 0, the value of K corresponding to deviation (DV).		K=1
(b) When gap width (GW) > 0, the value of K corresponding to deviation (DV).	$\vert DV\vert\vert \leqq W$	K=GG
	$\vert\vert DV\vert > GW$	$K=1-\frac{(1-GG)\times GW}{\vert DV\vert}$

DV: Deviation (%), GW: Gap width (%)= the rate of gap width corresponding to deviation, GG: Gap gain

In case of (a)



In case of (b)



(3) Deviation for BPI operation (DV') is calculated as follows.

Condition	Deviation for BPI operation (DV')
DV < -GW	DV'=-(GG × GW)+(DV+GW)
$\vert DV\vert \leqq GW$	DV'=GG × DV
DV > GW	DV'=GG × GW+(DV–GW)

DV': Deviation for BPI operation (%), DV: Deviation (%), GW: Gap width (%), GG: Gap gain

(4) Deviation (DV) for direct/reverse action is calculated as follows.

Condition	Deviation (DV)
Direct action (PN=1)	DV (%) = PVP (%) –SV (%)
Reverse action (PN=0)	DV (%) = SV (%) –PVP (%)

DV: Deviation (%), PVP (%): PV input value (%), SV (%) = $\frac{100}{RH-RL} \times$ (SV-RL),

RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>BPI operation (continued)</td><td>(5) BPI operation is conducted as follows.
<table>
<tr><th></th><th>Direct action</th><th>Reverse action</th></tr>
<tr><td>Deviation (DVn)</td><td>$DV_n = PV_n - SV_n$</td><td>$DV_n = SV_n - PV_n$</td></tr>
<tr><td>Output variation (△MV)</td><td colspan="2">$\triangle MV = Kp \times \triangle T \times \{DV_n + \frac{CT}{Ti} \times \Sigma DV_i\}$</td></tr>
</table>
Kp: Gain, Ti: Integral time, △T: Execution cycle, CT: Control cycle, Σ DVi: The cumulative value of DVn, DVn: Deviation, PVn: Process variable, SVn: The processing result of engineering value conversion
(a) Integral items and derivative items are listed below corresponding to each condition.
<table>
<tr><th>Condition</th><th>Processing</th></tr>
<tr><td>Either of 1), 2).
1) Ti=0
2) MLA or MHA is TRUE</td><td>$\frac{CT}{Ti} \times \Sigma DV_i$ = Hold the previous value</td></tr>
<tr><td>Ti ≠ 0</td><td>$\frac{CT}{Ti} \times \Sigma DV_i = \frac{CT}{Ti} \times (\Sigma DV_i + DV_n)$</td></tr>
</table>
Ti: Integral time, CT: Control cycle, Σ DVi: The cumulative value of DVn, DVn: Deviation, MLA: Output low limit value, MHA: Output high limit value
(b) Control cycle (CT) shall be set to be integral multiple of execution cycle (△T).
(c) Integral constant shall be set to be 0.0 or over control cycle (CT).
(d) PID operation of the tag access FB is executed in every control cycle (CT), (△MV output).
For other execution cycle (△T), the previous value shall be held. (△MV=0)</td></tr>
<tr><td>Engineering value conversion</td><td>Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.
$SV = \frac{RH - RL}{100} \times$ Setting value (%) from the primary loop +RL
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value</td></tr>
<tr><td>Inverse engineering value conversion</td><td>Convert the setting value (SV) of engineering value to percentage SV (%).
$SV (\%) = \frac{100}{RH - RL} \times (SV - RL)$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value</td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether Enable Alarm Detection or not in deviation check.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DVLA will not be detected.
● ERRI, DVLI
(2) "Disable Alarm Detection" by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
</table>

Other Functions

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) is. 1) Set △MV to 0. 2) Change the control mode automatically to MANUAL. 3) Reset DLVA when DLVA of alarm (ALM) occurs. 4) Alarm is not detected in deviation check.

Processing Operation

Control mode \ Processing	Deviation check	BPI operation	Engineering value conversion	Inverse engineering value conversion	Alarm
MAN, CMV, AUT	○	○	×	○	○ (*1)
CAS, CSV	○	○	○	○	○ (*1)

○: Execute ×: Not execute

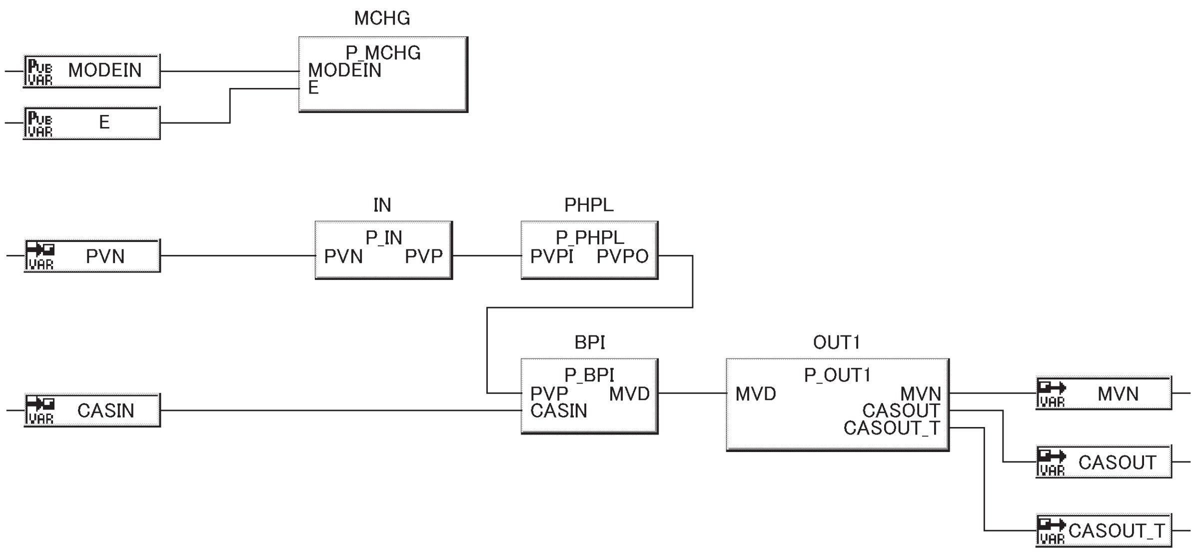
*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

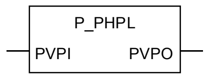
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.19 High/Low Limit Alarm Check (P_PHPL)

FB	FBD parts
P_PHPL	

Corresponding tag type
PID, 2PID, 2PIDH, PIDP, SPI, IPD, BPI, R, ONF2, ONF3, PFC_SF, PFC_SS, PFC_INT, MONI, SWM, MWM, PVAL

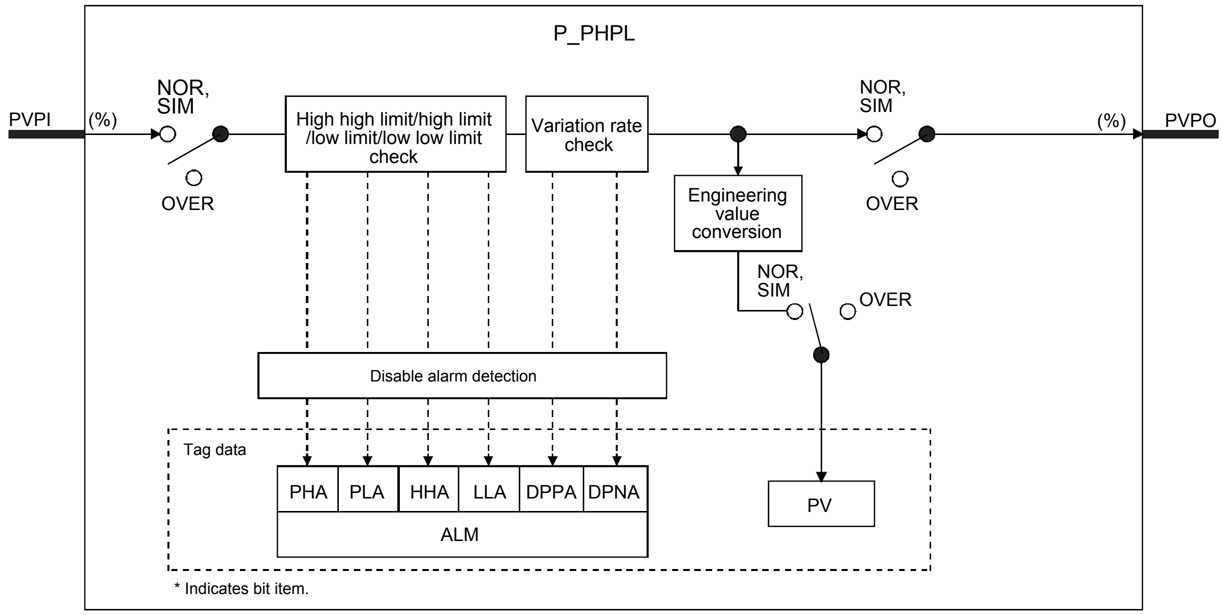
Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○*1	○	○

*1 Transition to CASDR is possible.

Functions overview: Execute high high/high/low/low low limit check and variation rate check to the input (PVPI) and output. Alarm occurs if check range is exceeded.

Function/FB classification: Tag access FB _ Loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Content	Range
Input	PVPI	Input variable	REAL	PV input (unit: %)	0 to 100
Output	PVPO	Output variable	REAL	PV output (unit: %)	0 to 100

Tag Data

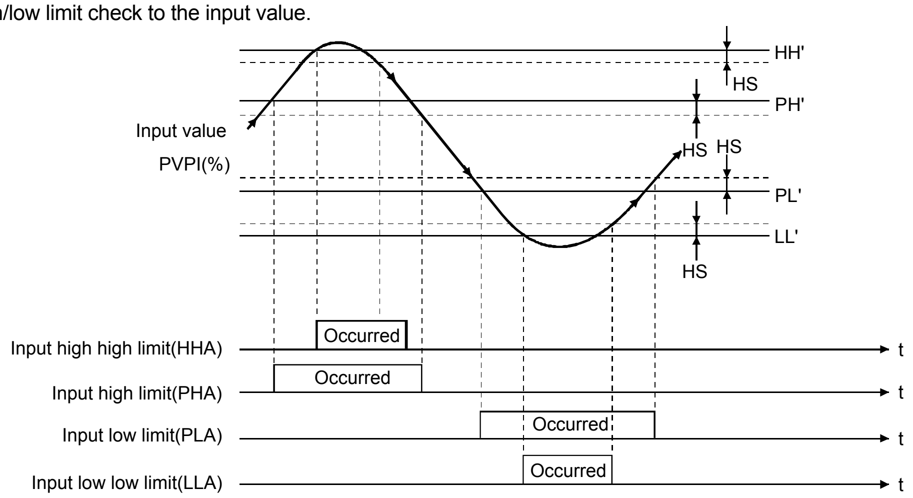
For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

Item	Content
High high limit/high limit/low limit/ low low limit check	Execute high/low limit check to the input value. (see below)
Variation rate check	During variation rate alarm check time (CTIM), compare the input variation rate with the variation rate alarm value (DPL) in each execution cycle (△T) and execute variation rate alarm check. (see below)

High high limit/high limit/low limit/ low low limit check

Execute high/low limit check to the input value.



		The input condition when alarm	
		TRUE (occur)	FALSE (reset)
Alarm (ALM)	Input high high limit (HHA)	Input value (%) > HH'	Input value (%) ≦ HH'-HS
	Input high limit (PHA)	Input value (%) > PH'	Input value (%) ≦ PH'-HS
	Input low limit (PLA)	Input value (%) < PL'	Input value (%) ≧ PL'+HS
	Input low low limit (LLA)	Input value (%) < LL'	Input value (%) ≧ LL'+HS

HH: High high limit alarm value, PH: High limit alarm value, PL: Low limit alarm value, LL: Low low limit alarm value, HH': High high limit alarm value (%), PH: High limit alarm value (%), PL: Low limit alarm value (%), LL: Low low limit alarm value (%), HS: High/low limit alarm hysteresis (%)

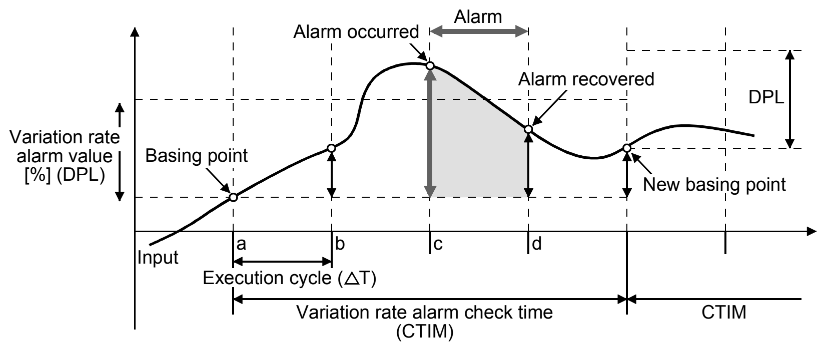
(1) Convert the alarm value of high high/high/low/low low limit to percentage (%) with the input information. (Inverse engineering value conversion)

$$HH'=\frac{HH-RL}{RH-RL}\times 100(\%),\ \ PH'=\frac{PH-RL}{RH-RL}\times 100(\%),\ \ PL'=\frac{PL-RL}{RH-RL}\times 100(\%),\ \ LL'=\frac{LL-RL}{RH-RL}\times 100(\%)$$

RH: Engineering value high limit, RL: Engineering value low limit

Variation rate check

During variation rate alarm check time (CTIM), compare the input variation rate with the variation rate alarm value (DPL) in each execution cycle (△T) and execute variation rate alarm check.



		Input condition in alarm occurrence	
		TRUE (occurred)	FALSE (restored)
Alarm (ALM)	Positive variation rate (DPPA)	(Xn+m) − Xn ≧ DPL	Beyond the left range
	Negative variation rate (DPNA)	(Xn+m) − Xn ≦ –DPL	Beyond the left range

DPL: Variation rate alarm value (%), m: Variation rate monitor counter=CTIM/△T(*1), △T: Execution cycle, CTM: Variation rate alarm check time, Xn: Basing point
*1: Set the CTIM and ΔT so that 'm' is greater than or equal to 2.

Item	Contents
Engineering value conversion	Convert the process variable (%) to engineering value. $PV = \frac{RH-RL}{100} \times \text{Input value (\%)} + RL$ RH: Engineering value high limit, RL: Engineering value low limit, PV: Process variable
Disable Alarm Detection	Set whether Enable Alarm Detection or not in high high/high/low/low low and variation rate check. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of HHA, LLA, PHA, PLA, DPPA, DPNA will not be detected. ● ERRI, HHI, LLI, PHI, PLI, DPPI, DPNI (2) "Disable Alarm Detection" by loop stop processing: Please refer to loop stop processing in the following contents.

Other Function

Item	Content
Loop stop processing	The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE. 1) Output (PVPO)= $\frac{RV-RL}{RH-RL} \times 100$ (%) 2) Automatically switches the control mode to MANUAL. 3) Reset HHA, LLA, PHA, PLA, DPPA and DPNA when alarm (ALM) HHA, LLA, PHA, PLA, DPPA and DPNA of alarm (ALM) occur. 4) Alarm is not detected in high high/high/low/low low limiter check.

Processing Operation

Control mode \ Processing	High high/high/low/ low low limit check	Variation rate check	Engineering value conversion	Alarm
MAN, CMV, AUT CAS, CSV, CASDR	○	○	○	○(*1)

○: Execute ×: Not execute

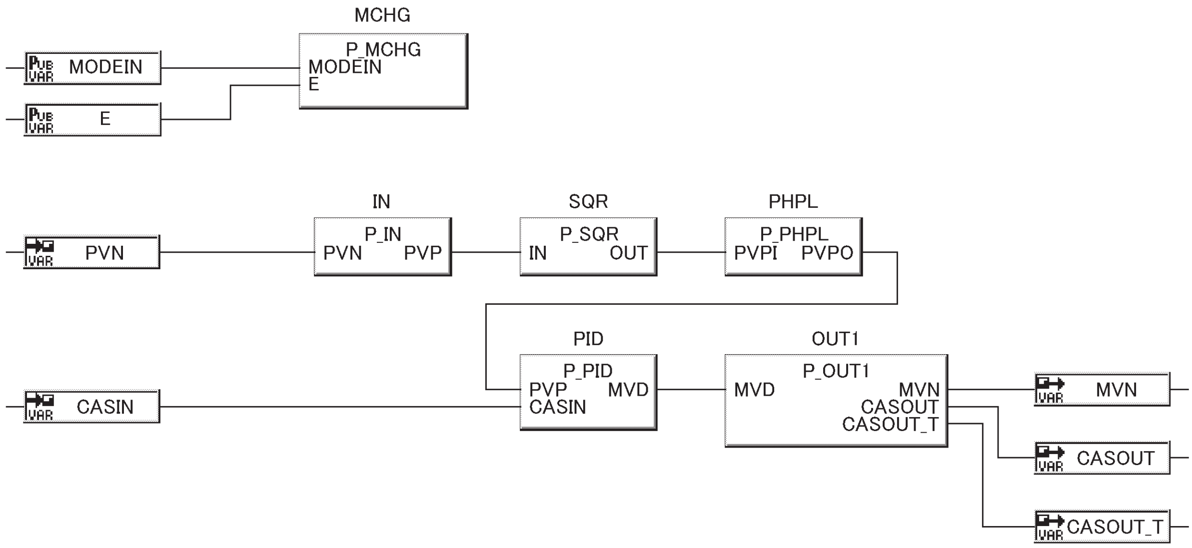
*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

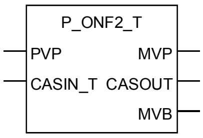
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.

8.2.20 2 Position ON/OFF (With Tracking to primary loop) (P_ONF2_T)

FB	FBD parts
P_ONF2_T	

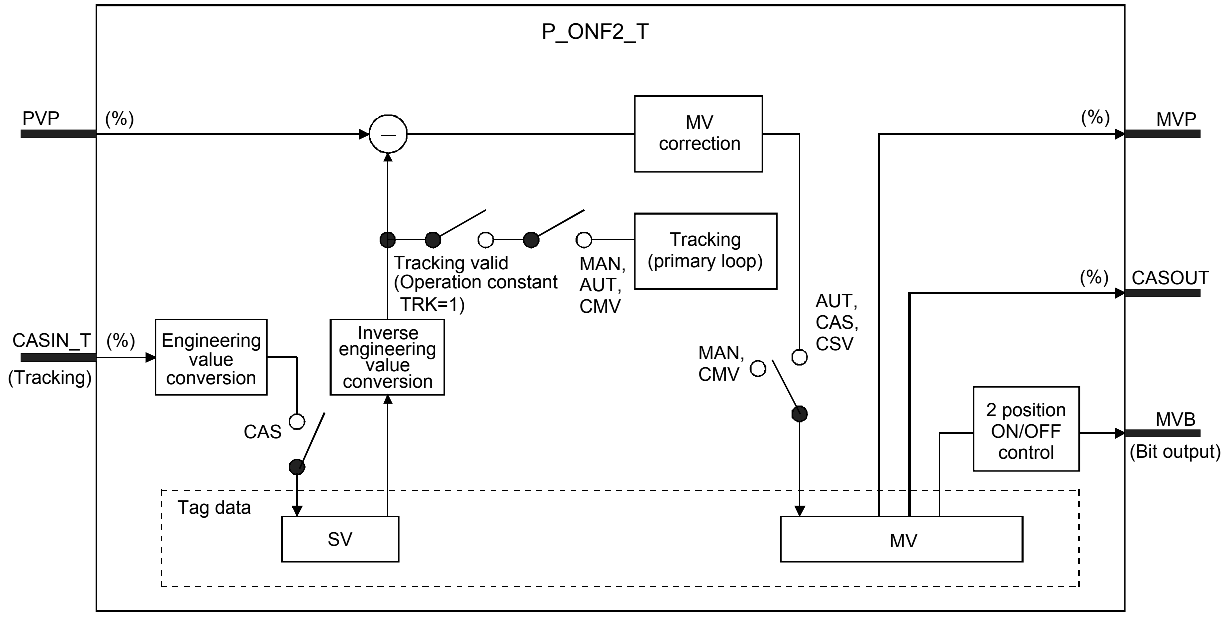
Corresponding tag type
ONF2

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 2 position ON/OFF control.

Function/FB classification name: Tag access FB_Loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	AR_REAL	Cascade input (unit: %) (With tracking)	0 to 100
Output	MVP	Output variable	REAL	MV output (unit: %)	0 to 100
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100
	MVB	Output variable	BOOL	ON/OFF output (ON if MV ≧ 50%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	TRK(*1)	Public variable	INT	Tracking flag (0:Not execute, 1:Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not upper, MV FALSE: Upper MV)	TRUE, FALSE	TRUE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

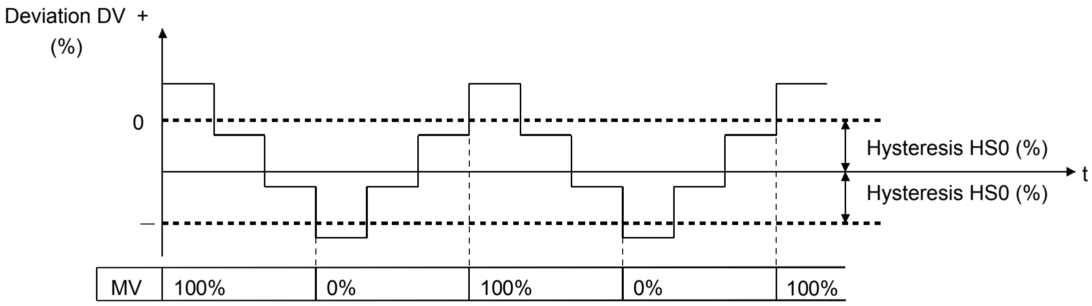
Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>MV correction</td><td>

Calculate deviation (DV)



Condition	Deviation (DV)
Direct action (PN=1)	DV (%)=PV (%)-SV (%)
Reverse action (PN=0)	DV (%)=SV (%)-PV (%)

DV: Deviation (%), HS0: Hysteresis (%), MV: MV output

$$SV(\%)=\frac{\text{SV-Engineering range low limit}}{\text{Engineering range high limit-Engineering range low limit}}\times 100$$

$$PV(\%)=\frac{\text{PV-Engineering range low limit}}{\text{Engineering range high limit-Engineering range low limit}}\times 100$$

Hysteresis is (%) is percentage to (Engineering range high limit-Engineering range low limit).

</td></tr>
<tr><td>2 position ON/OFF control</td><td>

Control 2 position ON/OFF according to MV value.

Condition	2 position ON/OFF output (MVB)
$\lvert MV\rvert \geqq 50(\%)$	TRUE
$\lvert MV\rvert < 50(\%)$	FALSE

MVB: 2 position ON/OFF output, MV: MV output

● Control cycle (CT) ought to be set to integral multiple of execution cycle ($\triangle$T).

</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Engineering value conversion</td><td>Convert setting value (%) from the primary loop in CAS or CSV mode (control mode) into engineering value.
$SV = \frac{RH-RL}{100} \times$ Setting value (%) from the primary loop + RL
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value</td></tr>
<tr><td>Inverse engineering value conversion</td><td>Convert setting value (SV) of engineering value into percentage SV (%)
$SV(\%) = \frac{100}{RH-RL} \times (SV-RL)$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value</td></tr>
<tr><td>Tracking processing</td><td>Whether the tracking processing is executed to the input variable CASIN_T is described as follows:
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SVPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table></td></tr>
</table>

Other Functions

Item	Contents
Loop stop processing	The following processing is exceeded if the stop alarm (SPA) is TRUE. 1) Hold output (MVP). 2) Change the control mode automatically to MAUNAL.

Processing Operation

Control mode \ Processing	MV correction	2 position ON/OFF control	Engineering value conversion	Inverse engineering value conversion	Tracking
MAN, CMV, AUT	○	○	○	○	○ (*1)
CAS, CSV	○	○	○	○	×

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

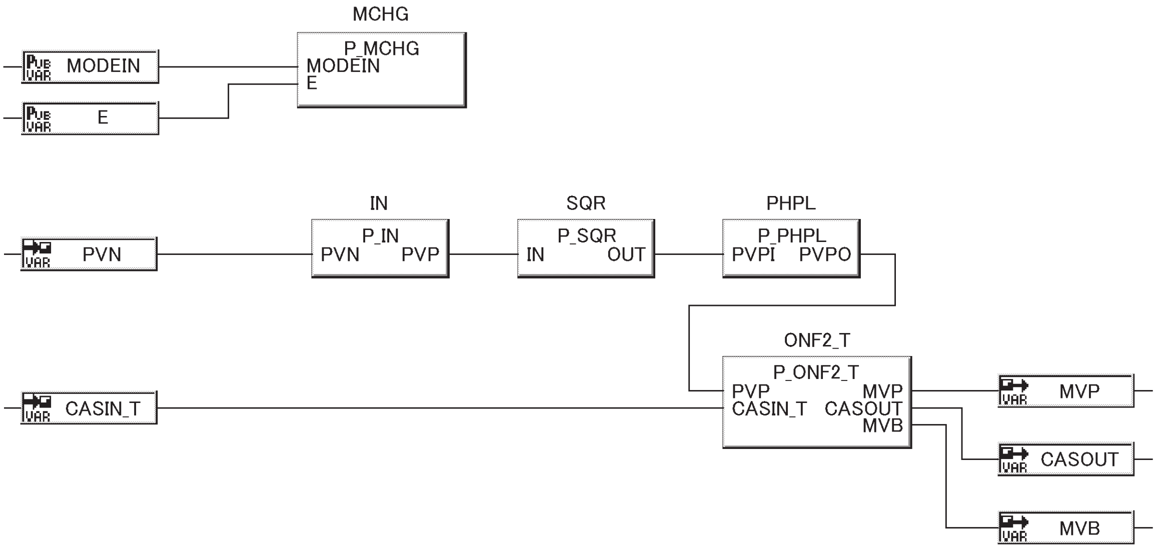
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

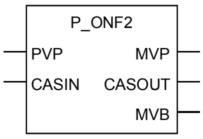
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



POINT
It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.21 2 Position ON/OFF (Without Tracking to primary loop) (P_ONF2)

FB	FBD parts
P_ONF2	

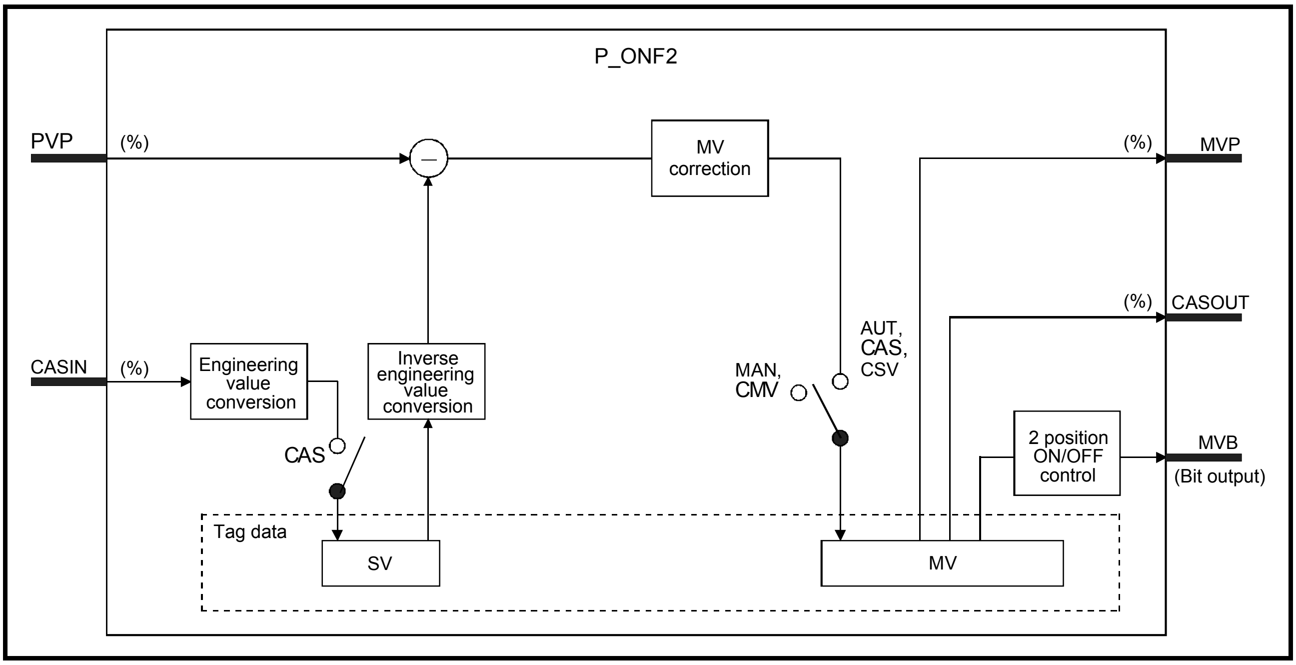
Corresponding tag type
ONF2

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 2 position ON/OFF control.

Function/FB classification name: Tag access GB_Loop control operation FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade input (unit: %) (Tracking)	0 to 100
Output	MVP	Output variable	REAL	MV output (unit: %)	0 to 100
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100
	MVB	Output variable	BOOL	ON/OFF output (ON if MV ≧ 50%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE

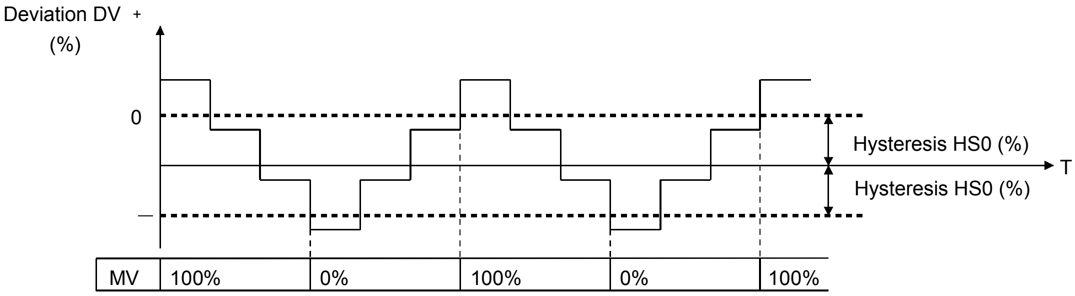
Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>MV correction</td><td>
Calculate deviation (DV)


<table>
<tr><th>Condition</th><th>Deviation (DV)</th></tr>
<tr><td>Direct action (PN=1)</td><td>DV (%)=PV (%)-SV (%)</td></tr>
<tr><td>Reverse action (PN=0)</td><td>DV (%)=SV (%)-PV (%)</td></tr>
</table>
DV: Deviation (%), HS0: Hysteresis (%), MV: MV output
$$SV(\%)=\frac{\text{SV- Engineering range low limit}}{\text{Engineering range high limit-Engineering range low limit}}\times 100$$
$$PV(\%)=\frac{\text{PV- Engineering range low limit}}{\text{Engineering range high limit-Engineering range low limit}}\times 100$$
Hysteresis (%) is percentage to (Engineering range high limit-Engineering range low limit).
</td></tr>
<tr><td>2 position ON/OFF control</td><td>
Control 2 position ON/OFF according to MV value.
<table>
<tr><th>Condition</th><th>2 position ON/OFF output (MVB)</th></tr>
<tr><td>|MV| ≧ 50(%)</td><td>TRUE</td></tr>
<tr><td>|MV| < 50(%)</td><td>FALSE</td></tr>
</table>
MVB: 2 position ON/OFF output MV: MV output
● Control cycle (CT) ought to be set as integral multiple of execution cycle (△T).
</td></tr>
</table>

Item	Contents
Engineering value conversion	Convert setting value (%) from the primary loop in CAS or CSV mode (control mode) into engineering value. $SV = \frac{RH-RL}{100} \times \text{Setting value (\%) from primary loop} + RL$ RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
Inverse engineering value conversion	Convert setting value (SV) of engineering value into percentage SV (%) $SV(\%) = \frac{100}{RH-RL} \times (SV-RL)$ RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

Other Functions

Item	Contents
Loop stop processing	The following processing is exceeded if the stop alarm (SPA) is TRUE. 1) Hold output (MVP). 2) Change the control mode automatically to MAUNAL.

Processing Operation

Control mode \ Processing	MV correction	2 position ON/OFF control	Engineering value conversion	Inverse engineering value conversion
MAN, CMV, AUT	○	○	×	○
CAS, CSV	○	○	○	○

○: Execute ×: Not execute

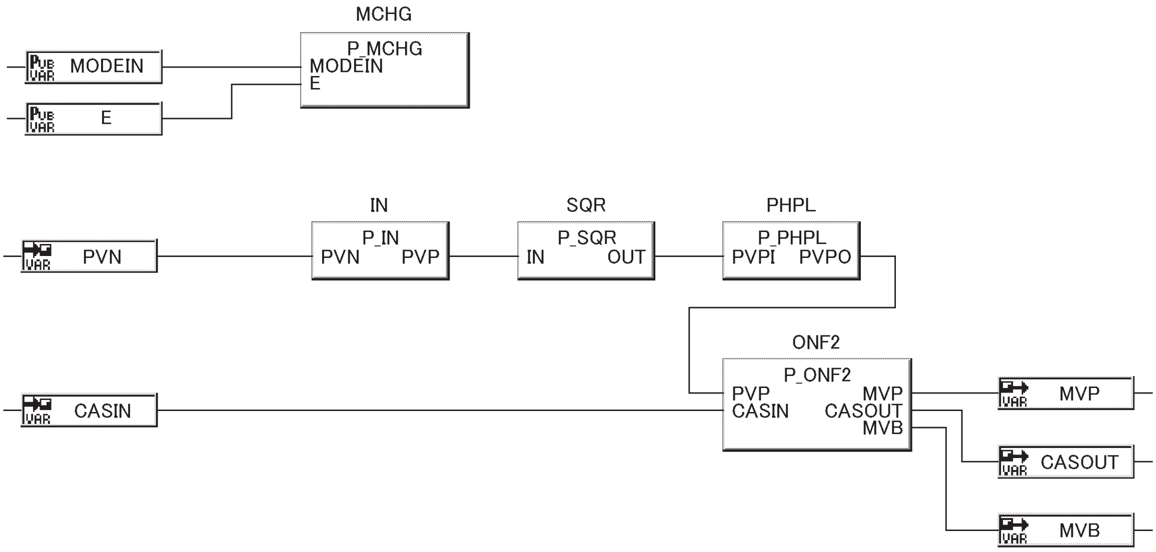
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

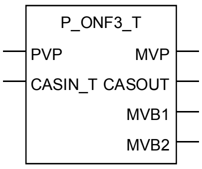
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.22 3 Position ON/OFF (With Tracking to primary loop) (P_ONF3_T)

FB	FBD parts
P_ONF3_T	

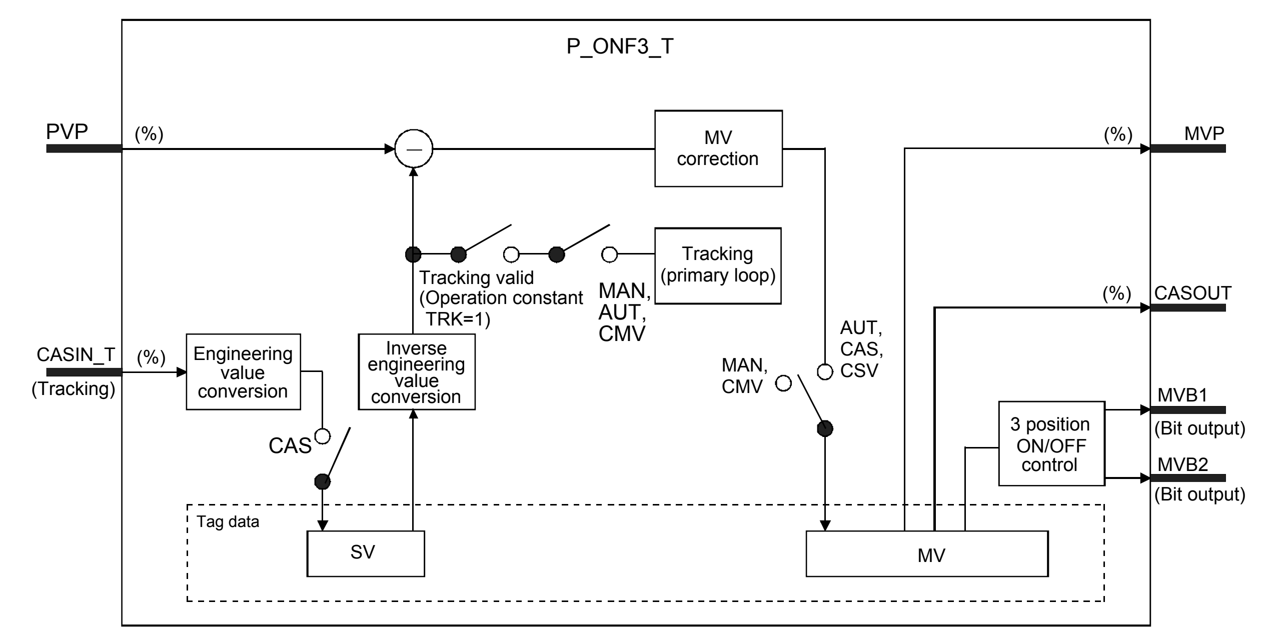
Corresponding tag type
ONF3

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 3 position ON/OFF control.

Function/FB classification name: Tag access FB_Loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	Cascade input (unit: %) (With tracking)	0 to 100
Output	MVP	Output variable	REAL	MV output (unit: %)	0 to 100
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100
	MVB1	Output variable	BOOL	ON/OFF output (ON if MV ≧ 75%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE
	MVB2	Output variable	BOOL	ON/OFF output (ON if MV < 25%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE

Public Variable (Operation Constant)

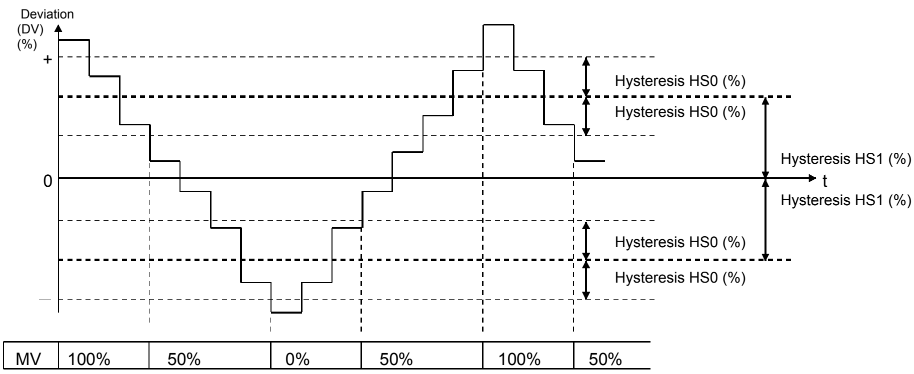
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: direct action)	0 to 1	0	User
	TRK(*1)	Public variable	INT	Tracking flag (0:Not execute, 1:Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TURE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TRUE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>MV correction</td><td>
Calculate deviation (DV)


<table>
<tr><th>Condition</th><th>Deviation (DV)</th></tr>
<tr><td>Direct action (PN=1)</td><td>DV (%)=PV (%)-SV (%)</td></tr>
<tr><td>Reverse action (PN=0)</td><td>DV (%)=SV (%)-PV (%)</td></tr>
</table>
DV: Deviation (%), HS0: Hysteresis (%), MV: MV output

$$SV(\%)= \frac{\text{SV- Engineering range low limit}}{\text{Engineering range high limit-Engineering range low limit}} \times 100$$

$$PV(\%)= \frac{\text{PV- Engineering range low limit}}{\text{Engineering range high limit-Engineering range low limit}} \times 100$$

Hysteresis (%) is percentage to (Engineering range high limit-Engineering range low limit).
</td></tr>
<tr><td>3 position ON/OFF control</td><td>
Control 3 position ON/OFF according to MV value.
<table>
<tr><th rowspan="2">Condition</th><th colspan="2">3 position ON/OFF output (MVB)</th></tr>
<tr><th>MVB1</th><th>MVB2</th></tr>
<tr><td>MV ≧ 75 (%)</td><td>TRUE</td><td>FALSE</td></tr>
<tr><td>25(%) ≦ MV < 75(%)</td><td>FALSE</td><td>FALSE</td></tr>
<tr><td>MV < 25(%)</td><td>FALSE</td><td>TRUE</td></tr>
</table>
MVB1, MVB2: 3 position ON/OFF output, MV: MV output
● Control cycle (CT) ought to be set as integral multiple of execution cycle (ΔT).
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Engineering value conversion</td><td>Convert setting value (%) from the primary loop in CAD or CSV mode (control mode) into engineering value.
$SV = \frac{RH-RL}{100} \times$ Setting value (%) from primary loop + RL
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value</td></tr>
<tr><td>Inverse engineering value conversion</td><td>Convert setting value (SV) of engineering value into percentage SV (%)
$SV (\%) = \frac{100}{RH-RL} \times (SV-RL)$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value</td></tr>
<tr><td>Tracking processing</td><td>Whether the tracking processing will be executed to the input variable CASIN_T is described as follows:
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SVPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td rowspan="2">Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td><td>Input variable CASIN_T does not perform tracking.</td></tr>
</table>
</td></tr>
</table>

Other Functions

Item	Contents
Loop stop processing	The following processing is exceeded if the stop alarm (SPA) is TRUE. 1) Hold output (MVP). 2) Change the control mode automatically to MAUNAL.

Processing Operation

Processing / Control mode	MV correction	3 position ON/OFF control	Engineering value conversion	Inverse engineering value conversion	Tracking
MAN, CMV, AUT	○	○	○	○	○ (*1)
CAS, CSV	○	○	○	○	×

○: Execute ×: Not execute

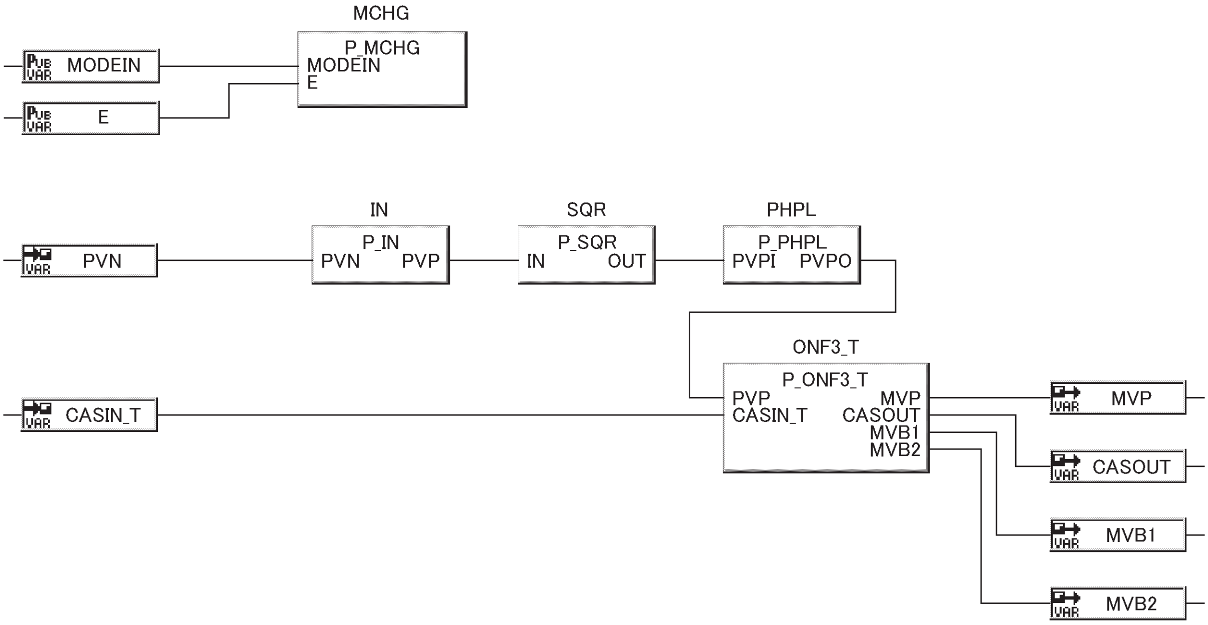
*1 Tracking is executed when the tracking flag (TRK) is 1.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

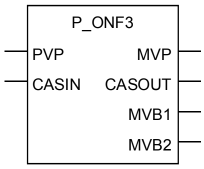
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.23 3 Position ON/OFF (Without Tracking to primary loop) (P_ONF3)

FB	FBD parts
P_ONF3	

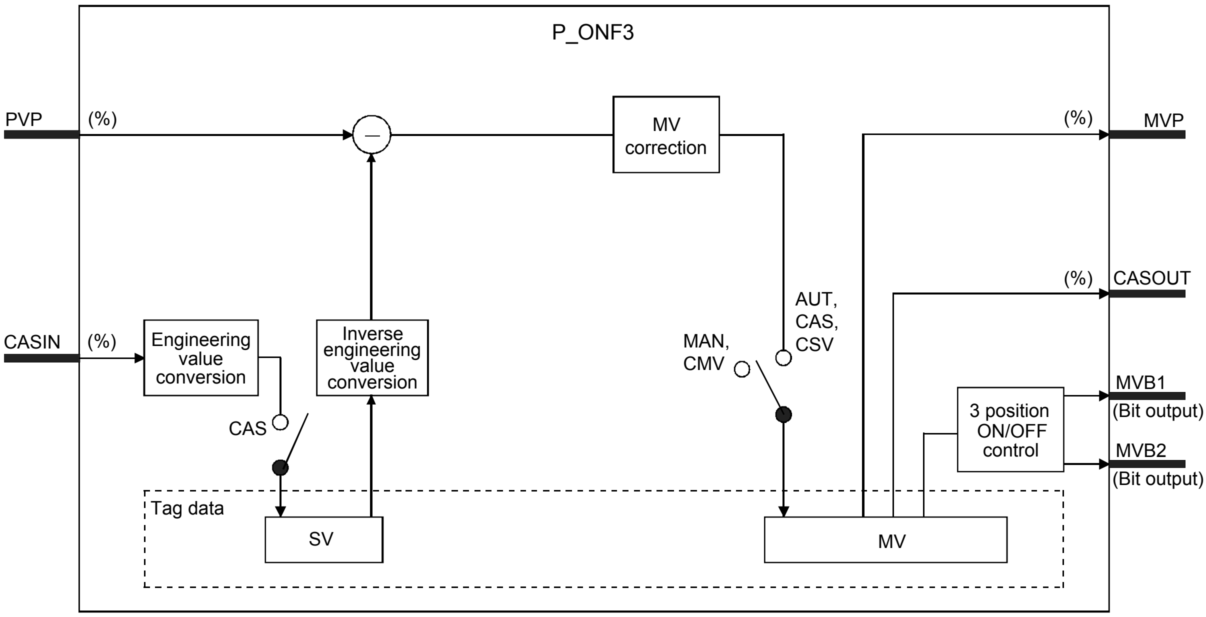
Corresponding tag type
ONF3

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 3 position ON/OFF control.

Function/FB classification name: Tag access FB_loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade input (unit: %) (Tracking)	0 to 100
Output	MVP	Output variable	REAL	MV output (unit: %)	0 to 100
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100
	MVB1	Output variable	BOOL	ON/OFF output (ON if MV ≧ 75%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE
	MVB2	Output variable	BOOL	ON/OFF output (ON if MV < 25%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User

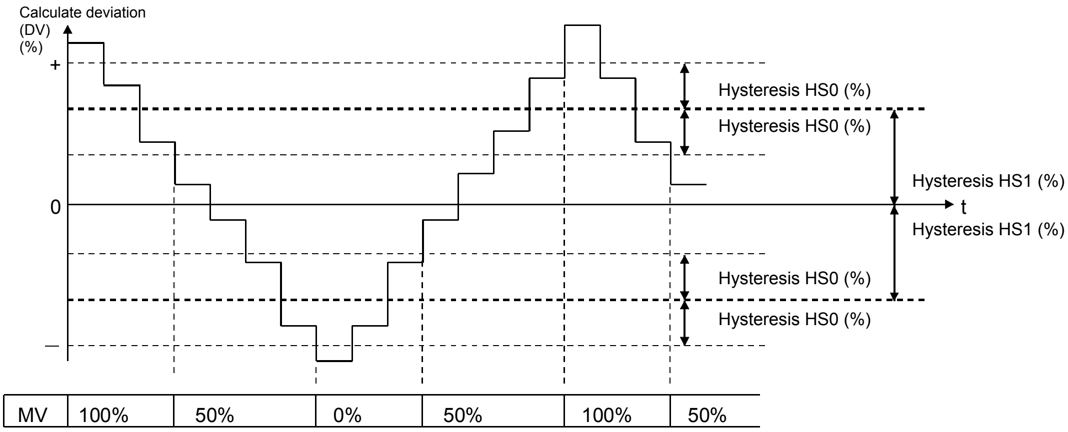
Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>MV correction</td><td>

Calculate deviation (DV)



<table>
<tr><th>Condition</th><th>Deviation (DV)</th></tr>
<tr><td>Direct action (PN=1)</td><td>DV (%)=PV (%)-SV (%)</td></tr>
<tr><td>Reverse action (PN=0)</td><td>DV (%)=SV (%)-PV (%)</td></tr>
</table>

DV: Deviation (%), HSO: Hysteresis (%), MV: MV output

$$SV(\%)=\frac{\text{SV - Engineering range low limit}}{\text{Engineering range high limit-Engineering range low limit}}\times 100$$

$$PV(\%)=\frac{\text{PV - Engineering range low limit}}{\text{Engineering range high limit-Engineering range low limit}}\times 100$$

Hysteresis (%) is percentage to (Engineering range high limit-Engineering range low limit).

</td></tr>
<tr><td>3 position ON/OFF control</td><td>

Control 3 position ON/OFF according to MV value.

<table>
<tr><th rowspan="2">Condition</th><th colspan="2">3 position ON/OFF output (MVB)</th></tr>
<tr><th>MVB1</th><th>MVB2</th></tr>
<tr><td>MV ≦ 75(%)</td><td>TRUE</td><td>FALSE</td></tr>
<tr><td>25(%) ≦ MV < 75(%)</td><td>FALSE</td><td>FALSE</td></tr>
<tr><td>MV < 25(%)</td><td>FALSE</td><td>TRUE</td></tr>
</table>

MVB1, MVB2: 3 position ON/OFF output MV: MV output

- Control cycle (CT) ought to be set as integral multiple of execution cycle (ΔT).

</td></tr>
</table>

Item	Contents
Engineering value conversion	Convert setting value (%) from the primary loop in CAS or CSV mode (control mode) into engineering value. $SV = \frac{RH-RL}{100} \times$ Setting value (%) from primary loop + RL RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value
Inverse engineering value conversion	Convert setting value (SV) of engineering value into percentage SV (%) $SV (\%) = \frac{100}{RH-RL} \times (SV-RL)$ RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value

Other Functions

Item	Contents
Loop stop processing	The following processing is exceeded if the stop alarm (SPA) is TRUE. 1) Hold output (MVP). 2) Change the control mode automatically to MAUNAL.

Processing Operation

Processing / Control mode	MV correction	3 position ON/OFF control	Engineering value conversion	Inverse engineering value conversion	Tracking
MAN, CMV, AUT	○	○	○	○	○ (*1)
CAS, CSV	○	○	○	○	×

○: Execute ×: Not execute

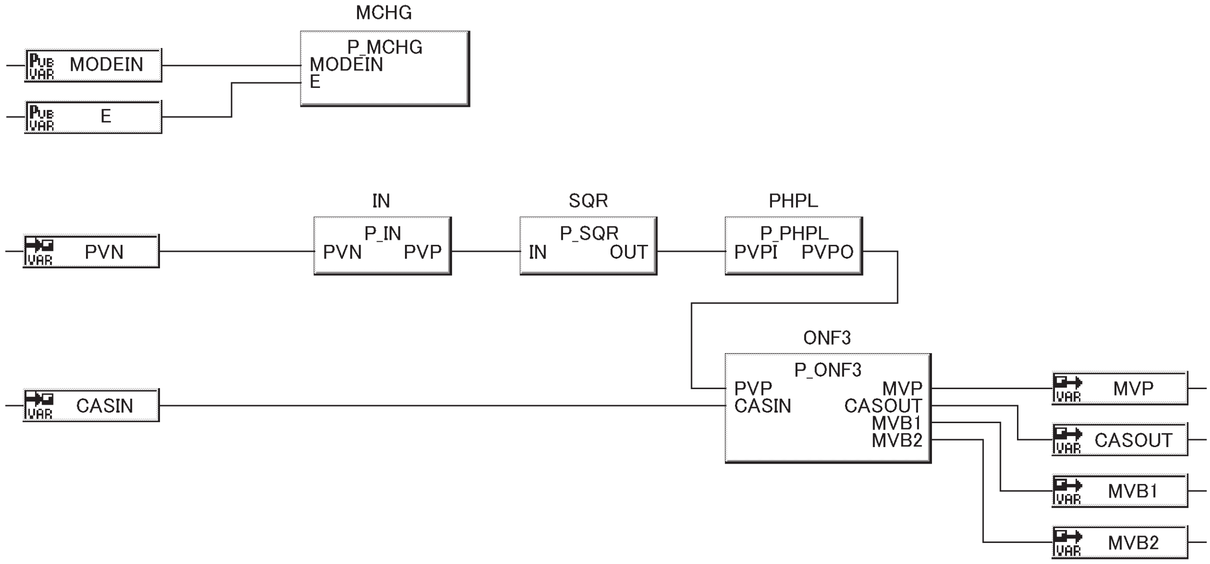
*1 Tracking is executed when the tracking flag (TRK) is 1.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

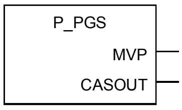
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



POINT
It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.24 Program Setter (P_PGS)

FB	FBD parts
P_PGS	

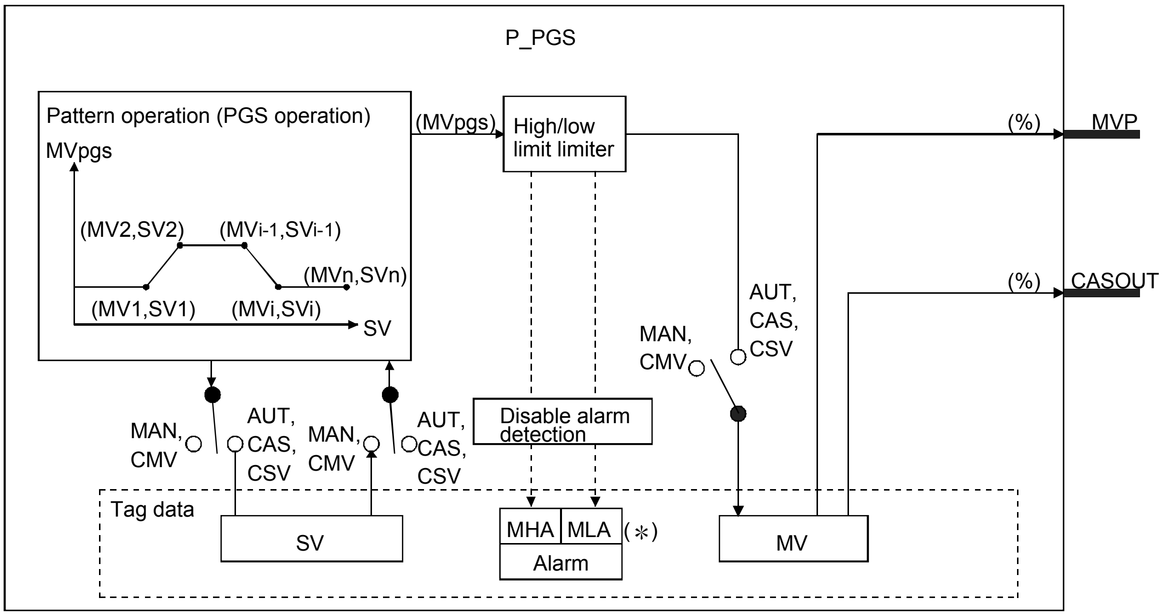
Corresponding tag type
PGS

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Set slope and setting value to time and control the program.

Function/FB classification name: Tag access FB_Loop control operation

Block Diagram



* Indicates bit item.

Input and Output Pins

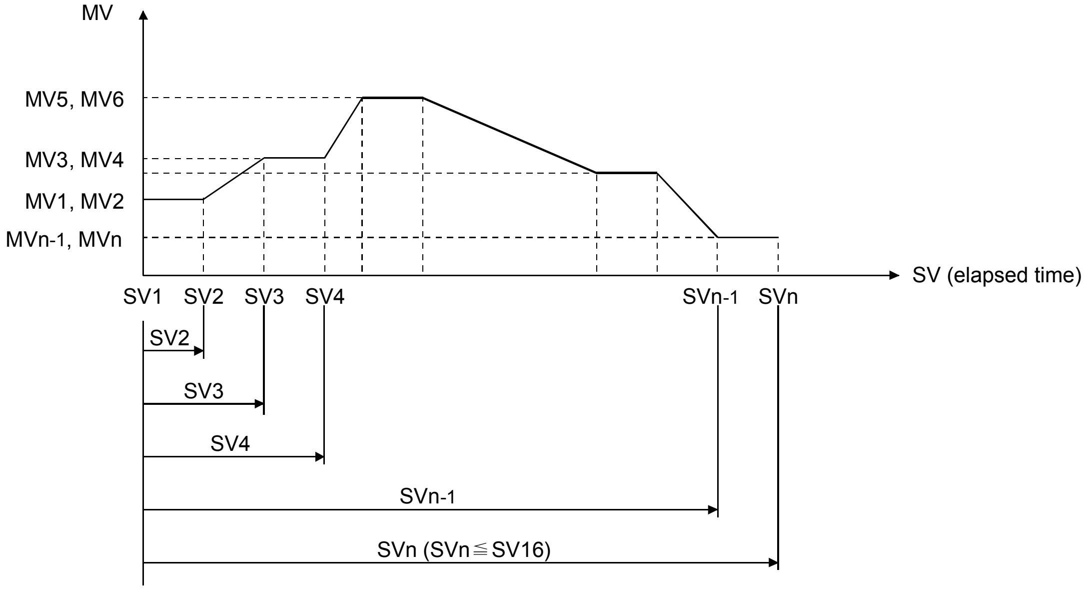
Pin	Variable name	Variable type	Data type	Contents	Range
Output	MVP	Output variable	REAL	MV output (unit: %)	0 to 100
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

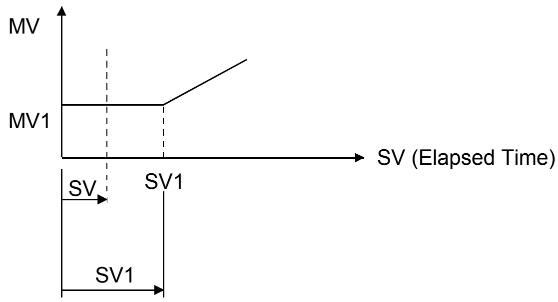
Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PGS operation</td><td>
Output the setting value which is fixed in advance according to elapsed time.
Operation type consists of hold/return/cyclic types.
If the Stop alarm (SPA) of alarm (ALM) is FALSE, following processing will be executed. (loop run processing)

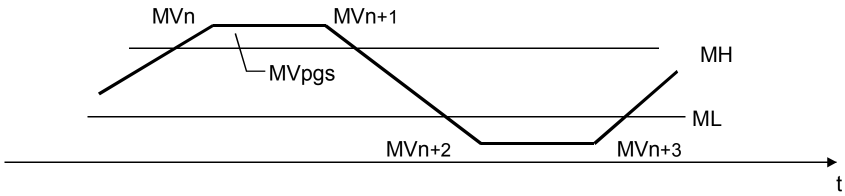


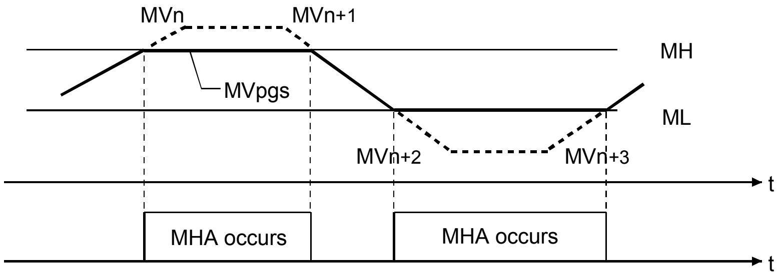
<table>
<tr><td colspan="2">Type</td><td>Hold</td><td>Return</td><td>Cyclic</td></tr>
<tr><td colspan="2">Control mode</td><td>AUT</td><td>AUT</td><td>CAS, CSV</td></tr>
<tr><td rowspan="2">MVpgs calculation</td><td>Elapsed time<SV1</td><td colspan="3">MV1</td></tr>
<tr><td>SVi-1 ≦ Elapsed time<SVi</td><td colspan="3">$\frac{(MVi-MVi-1)}{(SVi-SVi-1)} \times$ (Elapsed time -SVi-1) + MVi-1</td></tr>
<tr><td rowspan="4">Elapsed time > SVn (*1)</td><td>Control mode transition</td><td>MANUAL</td><td>MANUAL</td><td>No transition</td></tr>
<tr><td>SV (Elapsed time)</td><td>Previous value (SVn)</td><td>0</td><td>0</td></tr>
<tr><td>MV</td><td>Previous value (MVn)</td><td>Previous value (MVn)</td><td>MV1</td></tr>
<tr><td>Restart</td><td>MAN → AUT operation after set SV (Elapsed time)</td><td>MAN → AUT operation</td><td>Automatically restart</td></tr>
</table>

*1 If SV1>o second, the result is shown as the following.



</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>High/low limit check</td><td>Execute high/low limit check.
<table>
<tr><th>Control mode</th><th>Contents</th></tr>
<tr><td>MAN, CMV</td><td>

(1) MVpgs is not clamped to MH and ML, even if MVn is above MH or below ML.
(2) MHA and MLA Alarm (ALM) are not detected, no matter MVpgs is above MH or below.</td></tr>
<tr><td>AUT, CAS, CSV</td><td>

(1) MVpgs is clamped by MH and ML, if MVn is above MH or below ML.
(When MVn is programmed like the case of broken line in above diagram, the MVpgs output will be clamped to the solid time and output as a real.)
(2) MHA and MLA of Alarm (ALM) occur if MVn is above MH or below ML.</td></tr>
</table>
MVn: MV output, MH: output high limit value, ML: output low limit value, MHA: output high limit alarm, MLA: output low limit alarm</td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether Enable Alarm Detection or not in variation rate limiter and high/low limiter.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMA, MHA and MLA will not be detected.
● ERRI, MHI, MLI
(2) Disable Alarm Detection by control mode:
If the control mode is MHA or MLA of alarm (ALM) and MHA and MLA will be detected.
(3) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
</table>

Other Functions

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset MHA and MLA when MHA, and MLA of alarm (ALM) occurs. 4) Alarm is not detected in variation rate limiter and high/low limiter.

POINT

If operation constant "number of points" (PTNO) of tag data is 0, the same processing as loop stop processing is executed.

Processing Operation

Control mode \ Processing	PGS mode operation	High/low limit detection	Alarm
MAN (*1), CMV	×	×	× (*2)
AUT,CAS,CSV	○	○	○ (*3)

○: Execute ×: Not execute

*1 If operation constant number of points (PTNO) is 0, the same processing as loop stop processing is executed and control mode changes to MANUAL.

*2 An alarm (ALM) whose corresponding bit is TRUE (occurred) is reset, and the alarm cannot be detected.

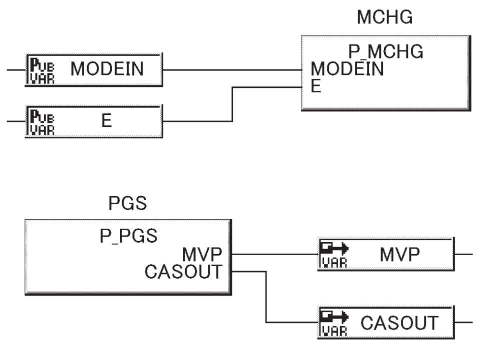
*3 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

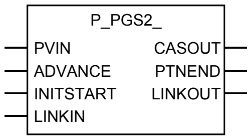
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.25 Multi-Point Program Setter (P_PGS2_)

FB	FBD parts
P_PGS2_	

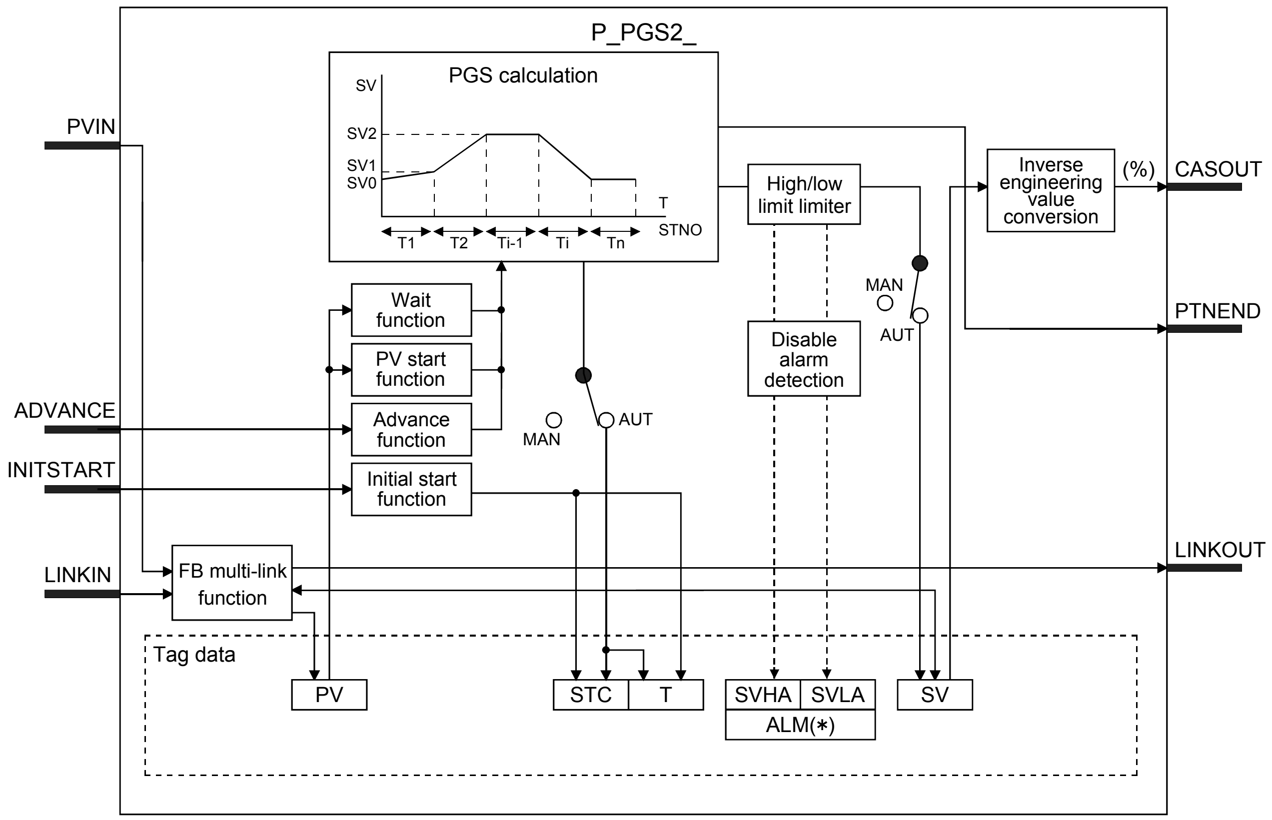
Corresponding tag type
PGS2

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	–	–	–

Function overview: Registers up to 32 steps specified with the time span and the setting value program, and calculates the setting values correspond to the passing time for each step in linear interpolation.

Function/FB classification name: Tag access FB_Loop control operation

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVIN	Input variable	REAL	Process input (Engineering value)	-32768 to 32767
	ADVANCE	Input variable	BOOL	Advance command	TRUE, FALSE
	INITSTART	Input variable	BOOL	Initial start command	TRUE, FALSE
	LINKIN	Input variable	ADR_REAL	Link input	–
Output	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100
	PTNEND	Output variable	BOOL	Pattern end output	TRUE, FALSE
	LINKOUT	Output variable	ADR_REAL	Link output	–

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	PVSTARTNO	Public variable	INT	PV start search start step	1 to 32	1	User
	PVENDNO	Public variable	INT	PV start search end step	1 to 32	32	User
	PRIMARY	Public variable	BOOL	Lead FB specified (TRUE: Lead, FALSE: Following)	TRUE, FALSE	TRUE	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
P_PGS2_ processing	TCNT	Public variable	INT	Second counter for minute mode	0 to 59	0	System
	TMCNT	Public variable	INT	Millisecond counter for second mode	0 to 999	0	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

Tag data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

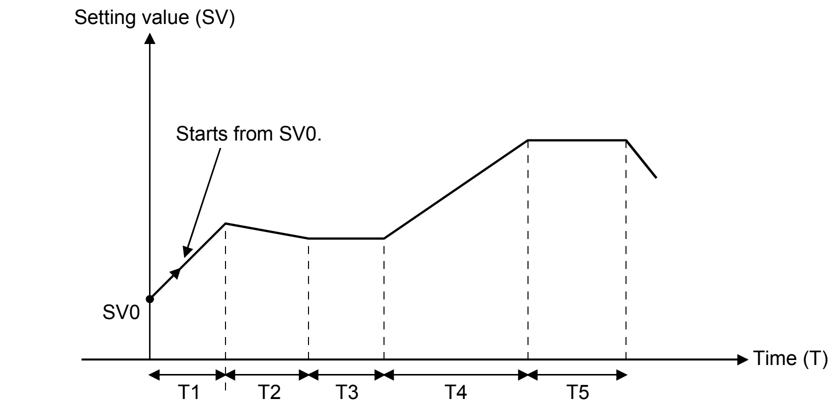
Function

Item	Contents
PV start	When starting the control (AUT mode switch), the difference from the PV value can be minimized by referencing the PV value and adjusting the control start point. The adjusted control start point is set as the current start point (SV0C).

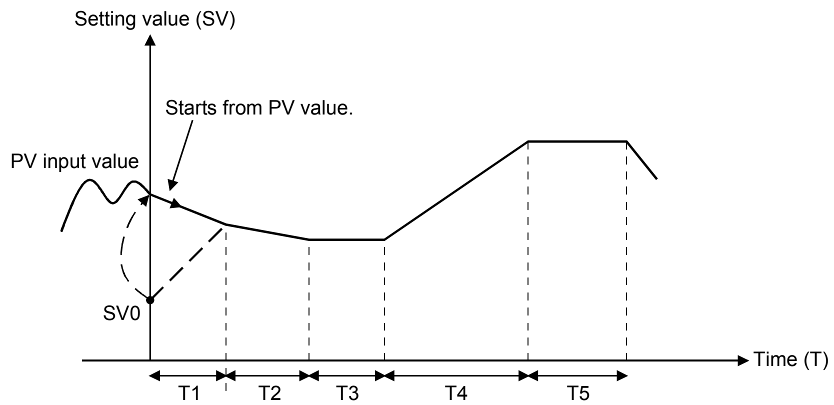
PV start type	Control start condition				
	STNO=0	STNO ≠ 0			
		STC = 0 and T= 0	STC = 0 and T ≠ 0	STC ≠ 0	After the second cycle at cyclic
PVSTART=0	☞ (1) in this section.	SV0 start (☞ (2) in this section).	☞ (5) in this section.	☞ (6) in this section.	☞ (7) in this section.
PVSTART=1		PV start 1 (☞ (3) in this section).			
PVSTART=2		PV start 2 (☞ (4) in this section).			

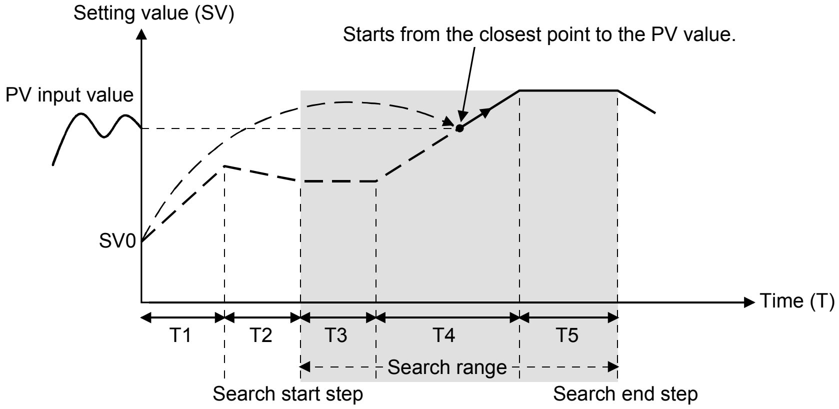
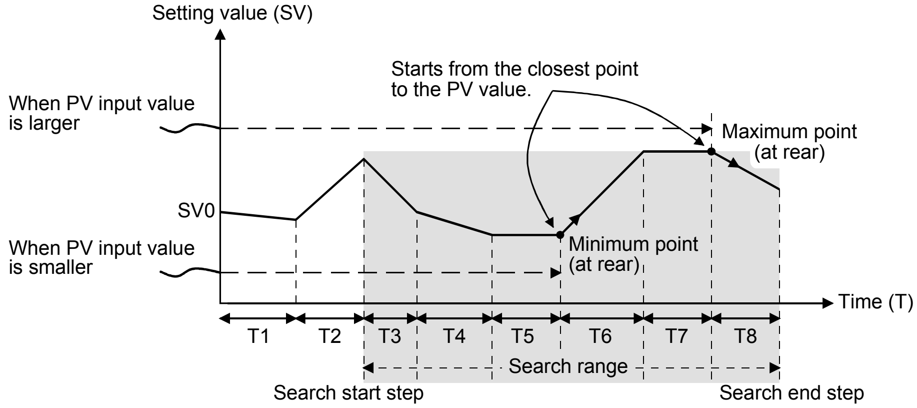
(1) When the Number of step setting (STNO) is 0
Since step setting is empty, when switching to AUT mode, the system switches it to MAN mode and turns pattern completion output (PTNEND) ON for one cycle.

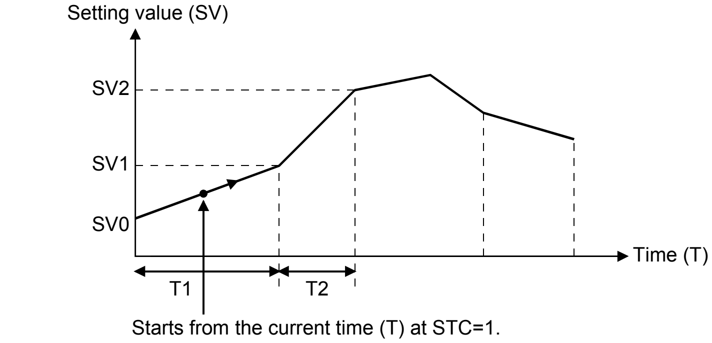
(2) SV0 start (Fixed start point: PVSTART = 0)
Starts the control with setting the start point (SV0) as the control start point, without referring the PV value.

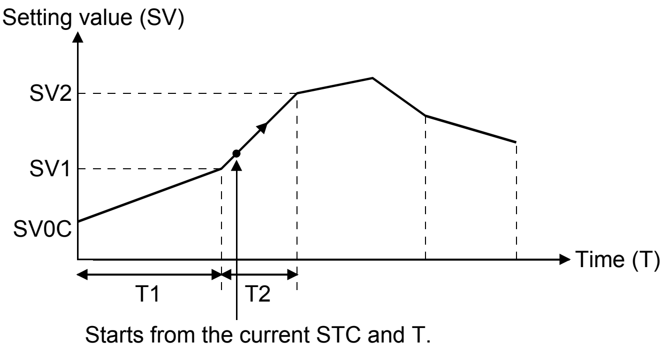


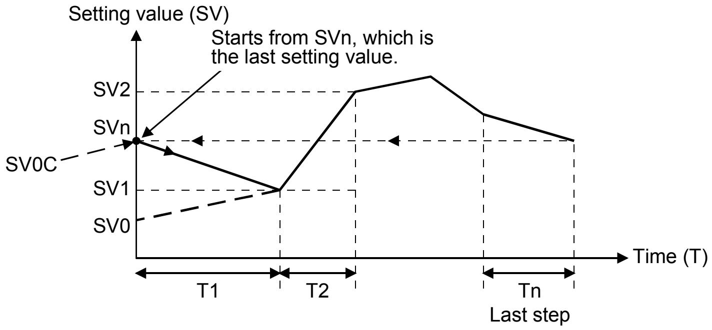
(3) PV start 1 (Start point correction: PVSTART = 1)
Starts the control with setting PV value as the control start point.



Item	Contents
PV start (continued)	(4) PV start 2 (Start point searching: PVSTART=2) Searches the matched point with the PV value in the range specified between PV start search start step (PVSTARTNO) and PV start search end step (PVENDNO), and starts the control from the found step number or time. Note that the closest point (maximum point or minimum point) is used when the matched point was not found. If multiple maximum points or minimum points were found, the rear point is used as a priority. When the matched point is the end point of the last step, the control is completed immediately, and the value prior to the AUT mode switch is maintained for SV value. • When the matched point was found in the search range  • When the matched point was not found in the search range 

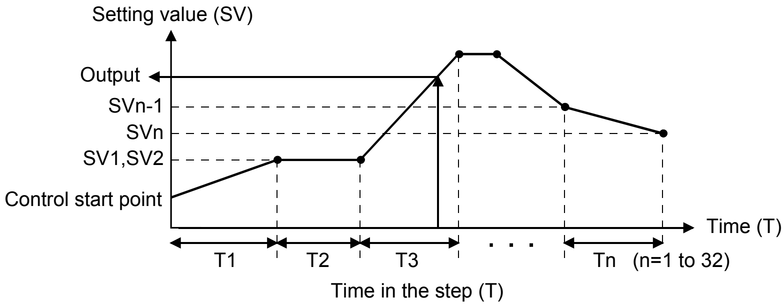
<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PV start
(continued)</td><td>
(5) Starting the control at STC = 0 and T ≠ 0
Sets the start point (SV0) as the control start point, and starts the control from the current time (T) at STC=1.


(6) Starting the control at STC ≠ 0
Starts the control from the current STC and T, using the control start point (current start point (SV0C)) from the previous control (AUT mode).


(7) After the second cycle at cyclic
Starts the control using the last setting value (SVn) as the next control start point (current start point (SV0C)).


</td></tr>
</table>

Item	Contents
PGS calculation	Outputs the setting values correspond to the time specified for each step in advance according to the passing time in AUT mode. The operation type has three types: HOLD, RETURN, and CYCLIC. Can register time spans and setting values in (Tn, Svn) format for each step up to 32 points. Note that the control start point follows the setting of the PV start type (PVSTART).



(1) Executing step specification
In AUT mode, the setting value jumps to the head of the specified step (T = 0), if the executing step number (STC) is changed.
Additionally, changing the time in the step (T) jumps the setting value to the time in the same step.

To jump to the last in the same step (T=Tn) by changing the time in the step (T) when the Wait function is enabled, refer to the Wait function.

(2) Step management
Processes the progress of the time in the step (T) and the executing step number (STC) in AUT mode.

Condition		Processing result	
		Time in the step (T)	Executing step number (STC)
STC ≦ 0		0	1
STC>0	T<0	0	Previous value
	0 ≦ T < Ti	T + △T *1	Previous value
	Ti ≦ T *2	0	Transition to next step (STC + 1)

△T: Execution cycle, i: Executing step number (STC)

*1 The addition of the execution cycle for the time in the step (T) is calculated in real numbers by the resolution to the units of milliseconds when the second is specified for the Unit of time (TUNIT), and to the units of seconds when the minute is specified for the Unit of time (TUNIT).

*2 For using the enabled Wait function, refer to the Wait function.

(3) SV_{PGS} calculation
Calculates the SV value corresponds to the executing step number (STC) and the time in the step (T) in AUT mode.

Condition	Processing result
Ti ≦ 0	SV_i
0 < T ≦ Ti	$\frac{SV_i - SV_{i-1}}{T_i} \times T + SV_{i-1}$ *1

i: Executing step number (STC)

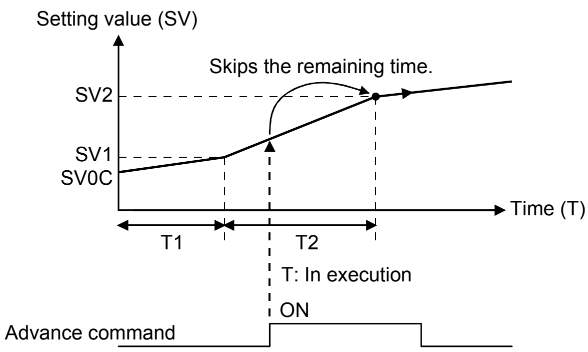
*1 SV_{PGS} calculation is calculated in real numbers.

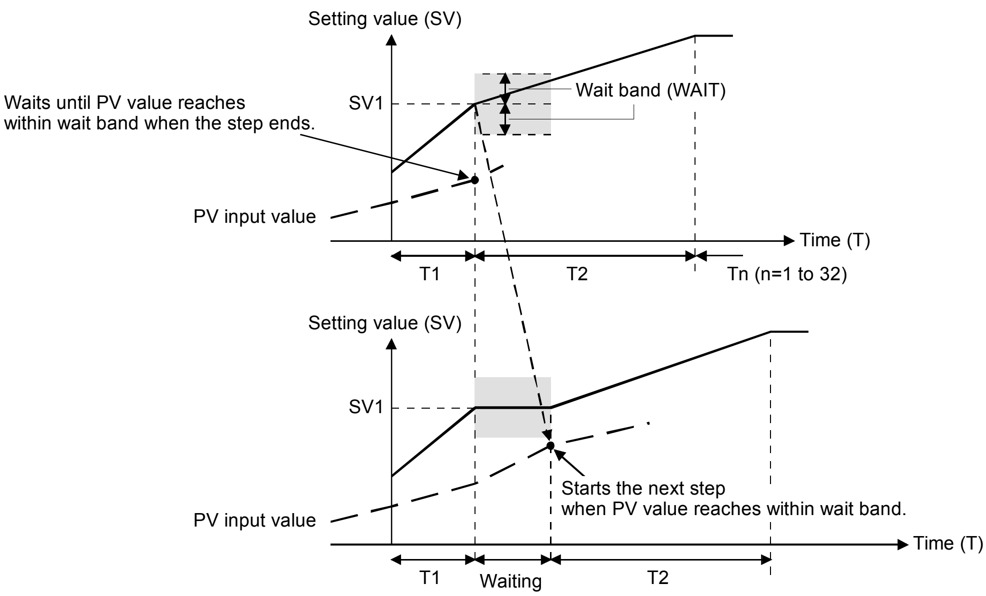
(4) Processing at the completion of the last step
Turns ON the pattern end output (output variable PTNEND) for one cycle and perform the processing shown in the following table, when the pattern is performed throughout and the last step is ended in AUT mode.

	Operation type		
	HOLD	RETURN	CYCLIC
Control mode transition	Transition to MAN	Transition to MAN	No transition
Time in the step (T)	Last value	0	0
Executing step number (STC)	Last value	0	1
SV_{PGS} calculation	SV output value of last step (SVn)	SV output value of last step (SVn)	Restarts from step 1 with setting SV output value of last step (SVn) as the control start point.

POINT

- Setting the time span from T1 to Tn in PGS calculation
  Set the number of seconds or minutes to the loop tag in only integer for time span.
  Unit of time is common to all steps and is specified by Units of time (TUNIT) in tag data.
  The maximum setting value of the time span is 32767 seconds (approximately 9 hours) or 32767 minutes (approximately 22 days) for each step.
- Setting the time span from SV1 to SVn in PGS calculation
  The setting values are set with engineering values. The available setting range is from -32768 to 32767, and the values are set as engineering values. A real number cannot be specified.
  When the setting after the decimal point is required depending on the engineering value range, for example, when the setting value is 1.5 MPa, convert its unit to 1500 kPa to fit in the range from -32768 to 32767.

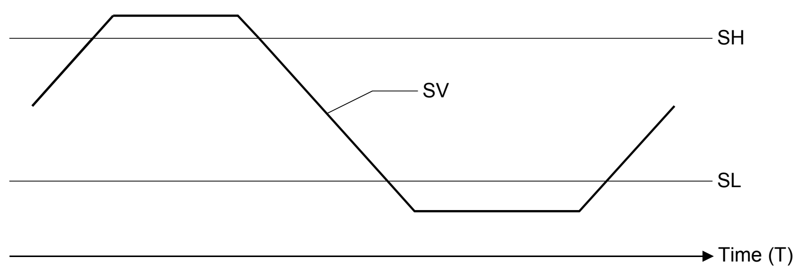
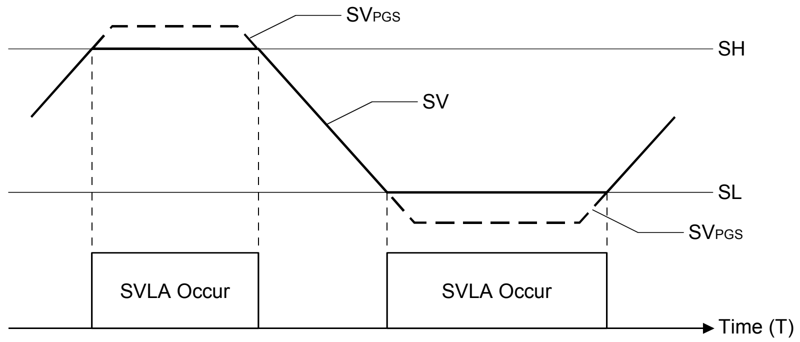
Item	Contents
Advance function	Proceeds to the next step by ending the executing step forcibly in AUT mode. This function is executed at the rising pulse of input variable ADVANCE command and proceeds by one step.  The advance command executed during the Wait function is disabled, and moves to the next step. Note when the advance command is executed at the last step, the control is completed immediately, and the value prior to the advance command is maintained for SV value.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Wait function</td><td>
Checks if the process variable (PV) follows the setting value (SV), and controls the process of steps when each step is completed in AUT mode.
The setting of wait band is common to all steps.
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Processing result</th></tr>
<tr><th>Control mode</th><th>Wait band</th><th>Process variable</th></tr>
<tr><td>MAN</td><td>—</td><td>—</td><td rowspan="2">The Wait function does not operate.</td></tr>
<tr><td rowspan="3">AUT</td><td>WAIT ≦ 0</td><td>—</td></tr>
<tr><td rowspan="2">WAIT > 0</td><td>| PV – SV | > WAIT</td><td>Stops the transition to the next step.
Maintains the setting value (SV) at the last value of the step.</td></tr>
<tr><td>| PV – SV | ≦ WAIT</td><td>Moves to the next step.</td></tr>
</table>
*1: When the step is completed by changing the time in the step (T=Tn), not the last value of the step but the setting value (SV_{PGS}) immediately before changing the time in the step is maintained. To maintain the last value of the step (SVn), change the time in the step to the time immediately before the last in the same step (T=Tn - 1).


</td></tr>
<tr><td>Disable alarm detection</td><td>
Sets whether to detect an alarm (ALM) in the setting value high/low limiter.
(1) Disable alarm detection processing with the Disable alarm detection (INH) setting of tag data:
If the following bit items in Disable alarm detection (INH) of tag data are TRUE, the SVHA and SLVA of the alarm (ALM) are not detected.
• ERRI, SVHI, SVLI
(2) "Disable alarm detection" by loop stop processing:
Refer to loop stop processing in the following contents.
</td></tr>
</table>

Item	Contents
Setting value high/low limiter	Checks the high/low limits of setting value in AUT mode.

Condition		Setting value (SV)	Alarm (ALM)	
			SV low limit (SVLA)	SV high limit (SVHA)
MAN		SV_{PGS}	FALSE (Reset)	FALSE (Reset)
AUT	SV_{PGS} > SH	SH	FALSE (Reset)	TRUE (Occur)
	SV_{PGS} < SL	SL	TRUE (Occur)	FALSE (Reset)
	SL ≦ SV_{PGS} ≦ SH	SV_{PGS}	FALSE (Reset)	FALSE (Reset)

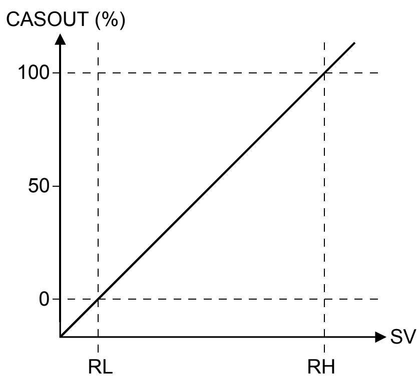
Control mode	Contents
MAN	
AUT	

SH: SV high limit value, SL: SV low limit value, SVHA: SV high limit alarm, SVLA: SV low limit alarm

Item	Contents
Inverse engineering value conversion	Converts the setting value (SV) in percentage (%) within the engineering value range from RL to RH.

$$\text{CASOUT (\%)} = \frac{SV - RL}{RH - RL} \times 100$$

CASOUT: Cascade output, RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value



<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Initial start function</td><td>Initializes the executing step number (STC) and the time in the step (T) at the rising pulse of INITSTART command, and switches to AUT mode in MAN mode.
<table>
<tr><th>Type</th><th>Variable name</th><th>Data type</th><th>Contents</th><th>Value to be initialized</th></tr>
<tr><td rowspan="2">Loop tag</td><td>STC</td><td>INT</td><td>Executing step number</td><td>0</td></tr>
<tr><td>T</td><td>INT</td><td>Time in the step</td><td>0</td></tr>
<tr><td rowspan="2">Public variable</td><td>TCNT</td><td>INT</td><td>Second counter for minute mode</td><td>0</td></tr>
<tr><td>TMCNT</td><td>INT</td><td>Millisecond counter for second mode</td><td>0</td></tr>
</table></td></tr>
<tr><td>FB multi-link function</td><td>When creating a program with over 32 steps, the program functions as a single program setting device by multi-linked FBs*1, and performs following processes.
• Set the same SV value among FBs to keep the SV value to be the last output value after changing SV value of any FBs.
• Manages the control mode with latter priority for two or more FBs not to be in AUT mode when switching to AUT mode.
• Copies the PV value input to the head FB to the PV value of the following FB.
*1 Indicates the tag FB of M_PGS2 type or user-defined tag FB of PGS2 type.
For the examples of programs having multi-linked FBs, refer to M_PGS2_ function.</td></tr>
</table>

Other Functions

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Loop stop processing</td><td>The loop process is executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE.
<table>
<tr><th colspan="6">Loop stop processing result</th></tr>
<tr><th>Input (PV)</th><th>Executing step number (STC) /Time in the step (T)</th><th>Output (SV)</th><th>Mode</th><th>Alarm reset *1</th><th>Alarm detection *2</th></tr>
<tr><td>Hold</td><td>Hold</td><td>Hold</td><td>MAN</td><td>Reset</td><td>No detection</td></tr>
</table>
*1 Recovers SVLA and SVHA when SVLA and SVHA of the alarm (ALM) occur.
*2 Alarms are not detected in SV high/low limiter.</td></tr>
</table>

Processing Operation

Processing / Control mode	PV start	PGS calculation	Advance function	Wait function	Alarm	Setting value high/low limiter	Inverse engineering value conversion	Initial start function	FB multi-link function
MAN	×	×	×	×	×	×	○	○	○
AUT	○	○	○	○	○ (*1)	○	○	×	○

○: Execute ×: Not execute

*1 The detection of the alarm whose corresponding bit of Disable alarm detection (INH) is TRUE (Enabled) is disabled.

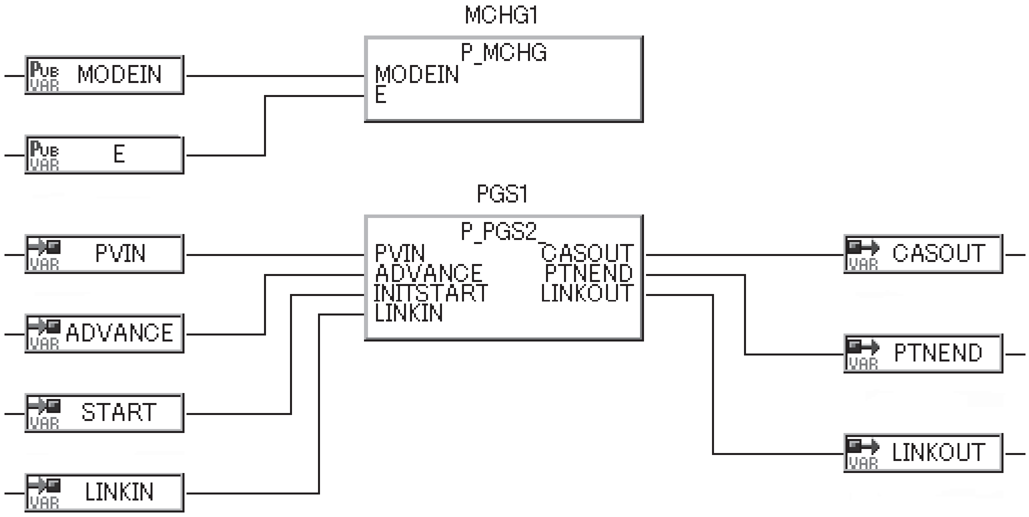
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

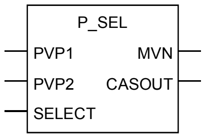
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error codes, refer to Appendix 2.

- When an overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



8.2.26 Loop Selector (Without Tracking to primary loop) (P_SEL)

FB	FBD parts
P_SEL	

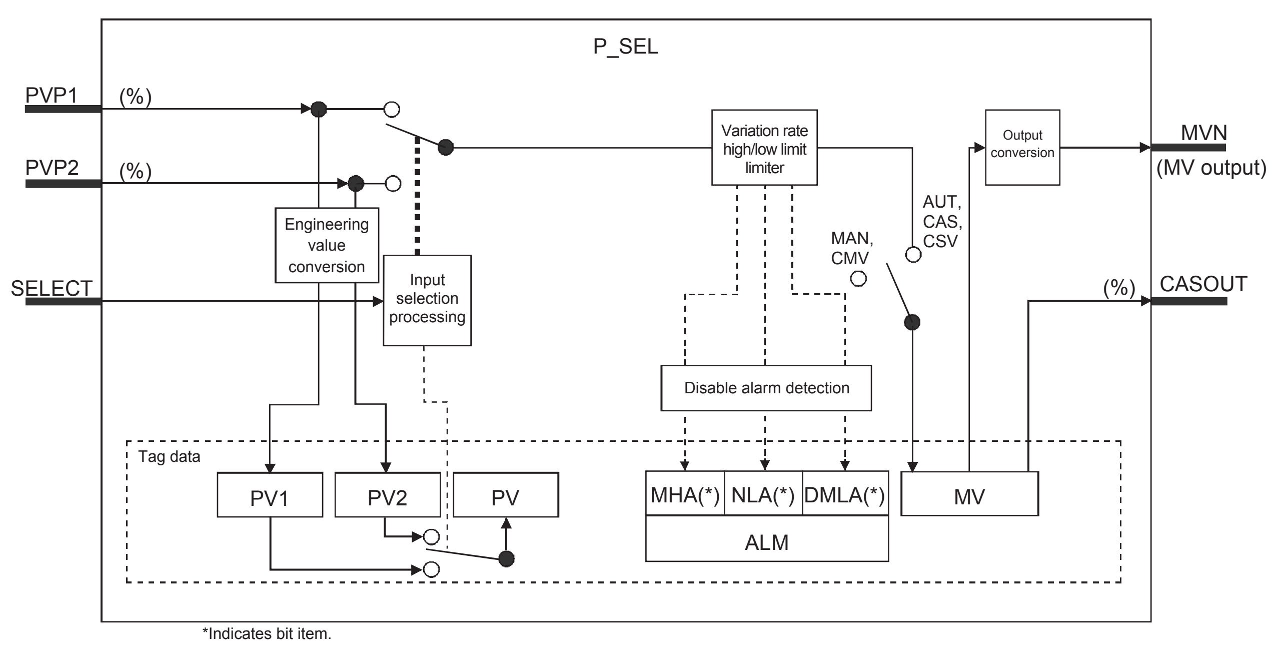
Corresponding tag type
SEL

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Selects the input value according to selection signal and output it.

Function/FB classification name: Tag access FB_Loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP1	Input variable	REAL	PV input (unit: %)	0 to 100
	PVP2	Input variable	REAL	PV input (unit: %)	0 to 100
	SELECT	Input variable	BOOL	Selection signal (TRUE: PVP2, FALSE: PVP1)	TRUE, FALSE
Output	MVN	Output variable	REAL	MV output to module FB	NMIN to NMAX
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	TRK	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

Item	Contents

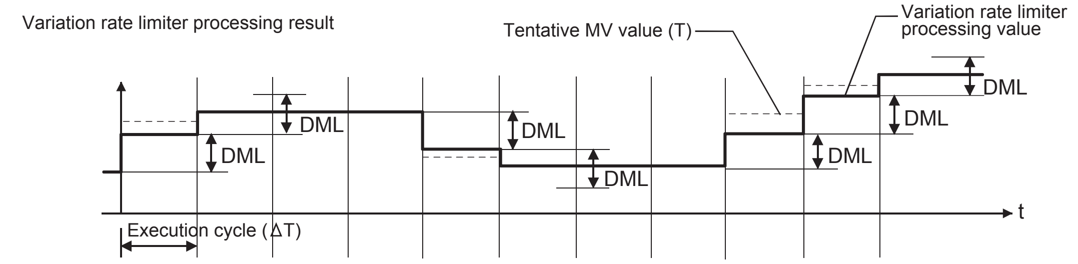
Input selection processing

Selection signal	Result
FALSE	Select input PVP1 (%)
TRUE	Select output PVP2 (%)

Variation rate and high/low limiter

Execute variation rate limiter and high/low limit check to input value.

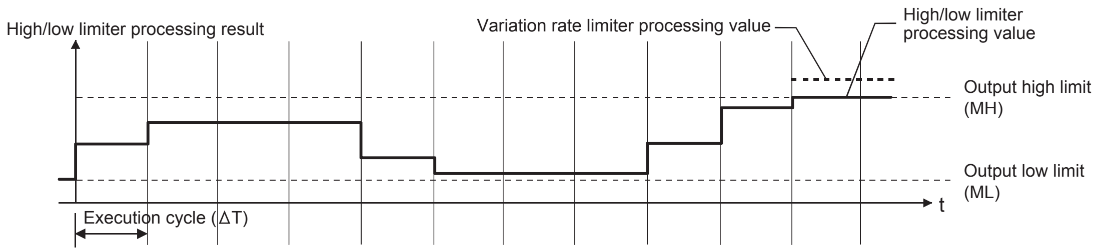
- Variation rate limiter



Condition	Variation rate limiter processing result	Alarm (ALM)
		Output variation rate limit (DMLA)
|T-MV| ≦ DML	T	FALSE (reset)
T-MV > DML	MV + DML	TRUE (occur)
T-MV < -DML	MV - DML	TRUE (occur)

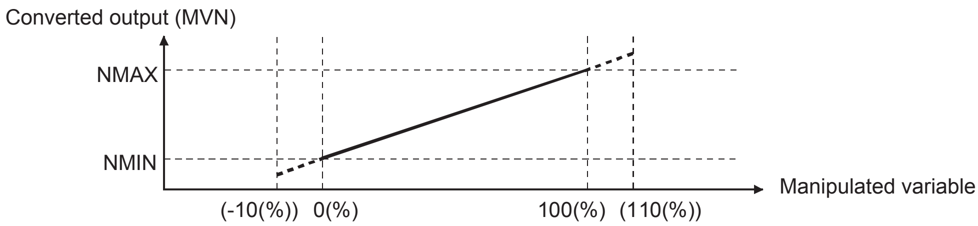
T: tentative MV value, MV: Manipulated variable, DML: High limit of output change

- High/low limiter



Condition	High/low limiter processing result	Alarm (ALM)	
		Output low limit (MLA)	Output high limit (MHA)
Variation rate limiter processing result > MH	MH	FALSE (reset)	TRUE (occur)
Variation rate limiter processing result < ML	ML	TRUE (occur)	FALSE (reset)
ML ≦ variation rate limiter processing result ≦ MH	Variation rate limiter processing value	FALSE (reset)	FLALSE (reset)

MH: Output high value, ML: Output low value

Item	Contents
Engineering value conversion	Convert the input value (%) into engineering value. $PV_n = \frac{RH-RL}{100} \times PV_n(\%) + RL$ RH: Engineering value high limit, RL: Engineering value low limit, PVPn: PV input value (%), PVn: PV1,PV2
Output conversion	Perform output conversion processing.  $\text{Converted output (MVN)} = \{ (NMAX - NMIN) \times \frac{MV}{100} \} + NMIN$ NMAX: High limit value of output conversion, NMIN: Low limit value of output conversion, MV: Manipulated variable (%), MVN: output value of output conversion
Disable Alarm Detection	Set whether Enable Alarm Detection or not in deviation check. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected. ● ERRI, DMLI, MHI, MLI (2) Disable Alarm Detection by loop stop processing: Please refer to loop stop processing in the following contents.

Other Functions

Item	Contents
Loop stop processing	The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occurs. 4) Alarm is not detected in variation rate limiter and high/ low limiter.

Processing Operation

Processing / Control mode	Engineering value conversion	Variation rate and high/low limiter	Output conversion	Alarm
MAN, CMV	○	×	○	○ (*1)
AUT, CAS, CSV	○	○	○	○ (*1)

○: Execute ×: Not execute

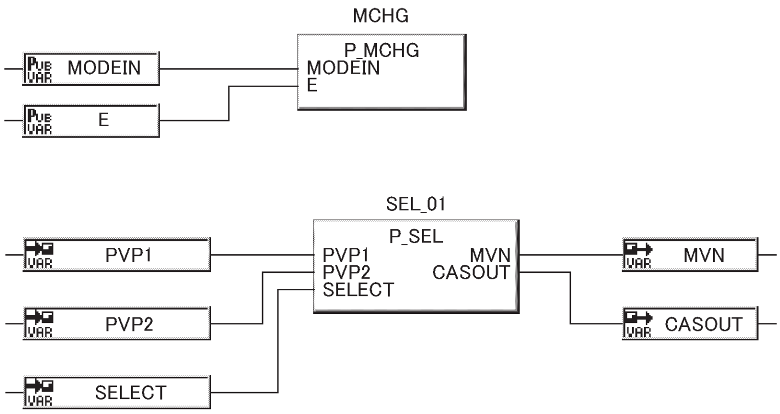
*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

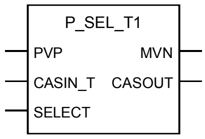
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.27 Loop Selector (With Tracking to primary loop) (P_SEL_T1)

FB	FBD parts
P_SEL_T1	

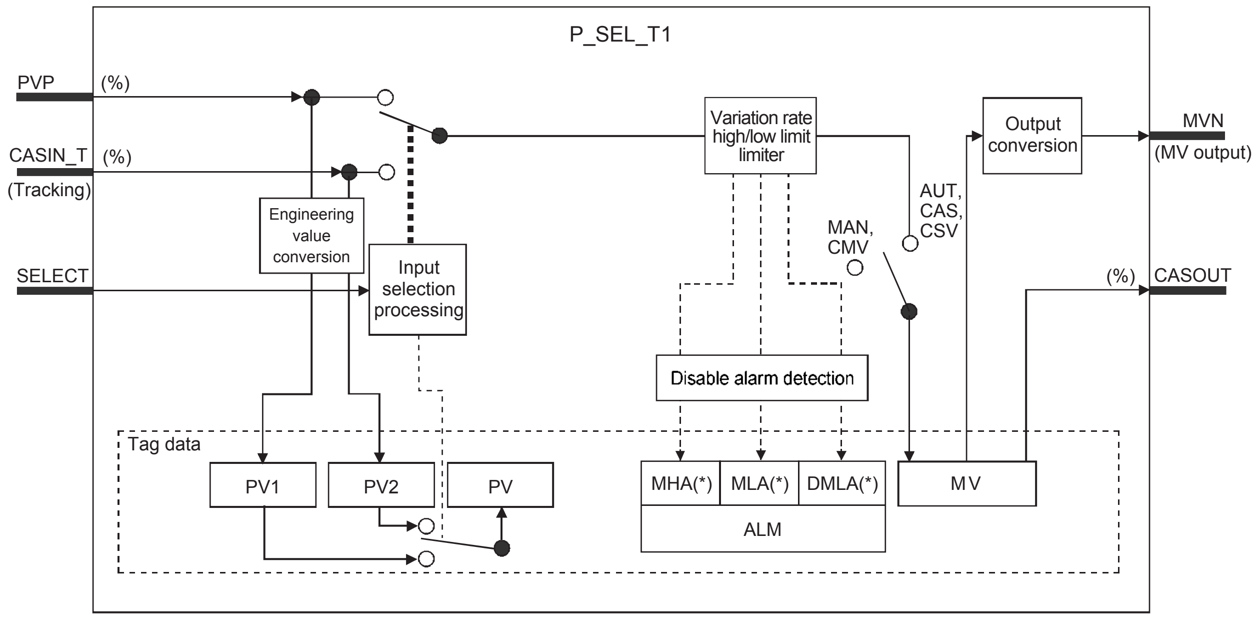
Corresponding tag type
SEL

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Selects the input value according to selection signal and output it. (With tracking)

Function/FB classification name: Tag access FB_loop control operation FB

Block Diagram



*Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	PV input (unit: %) (With tracking)	0 to 100
	SELECT	Input variable	BOOL	Selection signal (TRUE: CASIN_T, FALSE: PVP)	TRUE, FALSE
Output	MVN	Output variable	REAL	Output to unit FB	NMIN to NMAX
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Public variable	REAL	Converted output high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Converted output low limit	-999999 to 999999	0.0	User
	TRK(*1)	Public variable	INT	Tracking flag (0:Not execute, 1:Execute)	0 to 1	0	User
	SVPTN_B4	Public variable	BOOL	CASIN_T pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TURE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

Item: Input selection processing

Contents:

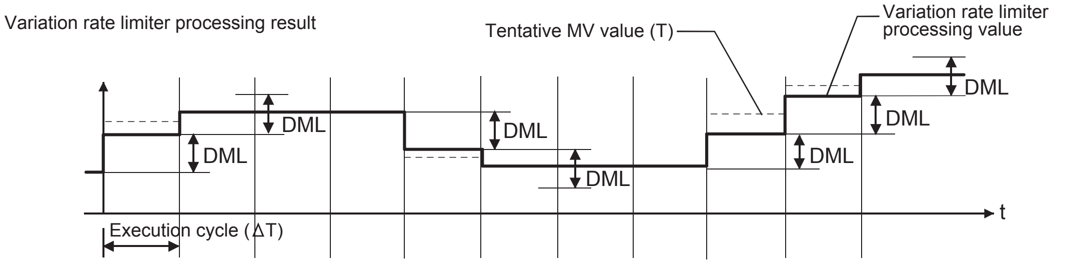
Selection signal	Result
FALSE	Select input PVP1 (%)
TRUE	Select output CASIN_T (%)

Item: Variation rate limiter and high/low limiter

Contents:

Execute variation rate limiter and high/low limiter check to input value.

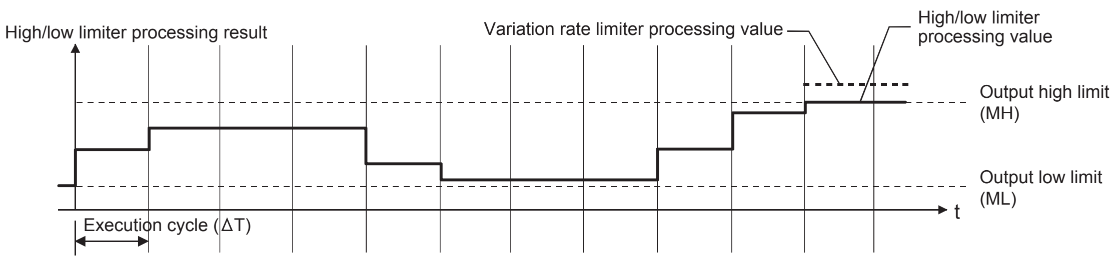
- Variation rate limiter



Condition	Variation rate limiter processing result	Alarm (ALM) Output variation rate limit (DMLA)
|T-MV| ≦ DML	T	FALSE (reset)
T-MV > DML	MV + DML	TRUE (occur)
T-MV < -DML	MV - DML	TRUE (occur)

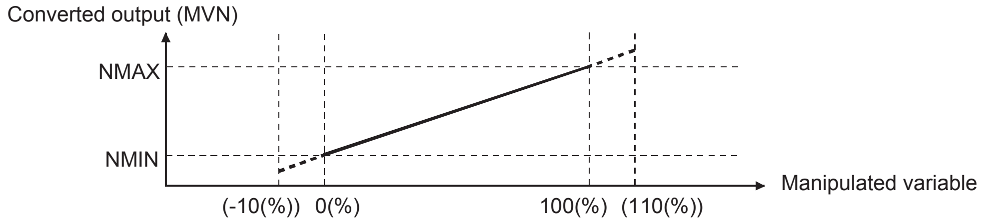
T: Tentative MV value, MV: Manipulated variable, DML: High limit of output change

- High/low limiter



Condition	High/low limiter processing result	Alarm (ALM)	
		Output low limit (MLA)	Output high limit (MHA)
Variation rate limiter processing result > MH	MH	FALSE (reset)	TRUE (occur)
Variation rate limiter processing result < ML	ML	TRUE (occur)	FALSE (reset)
ML ≦ Variation rate limiter processing result ≦ MH	Processing value of variation rate limiter	FALSE (occur)	FLALSE (reset)

MH: Output high value   ML: Output low value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Engineering value conversion</td><td>Convert the input value (%) to engineering value.
$$PV_n = \frac{RH-RL}{100} \times PVP_n\ (\%) + RL$$
RH: Engineering value high limit, RL: Engineering value low limit, PVPn: PV input value (%), PVn: PV1, PV2</td></tr>
<tr><td>Output conversion</td><td>Perform output conversion processing.


$$\text{Converted output (MVN)} = \{ (NMAX - NMIN) \times \frac{MV}{100} \} + NMIN$$
NMAX: High limit value of output conversion, NMIN: Low limit value of output conversion, MV: Manipulated variable (%), MVN: Output value of output conversion</td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether Enable Alarm Detection or not in deviation check.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected.
● ERRI, DMLI, MHI, MLI
(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
</table>

Other Functions

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Loop stop processing</td><td>The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE.
1) Hold the output (MVN).
2) Change the control mode automatically to MANUAL.
3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occurs.
4) Alarm is not detected in variation rate limiter and high/ low limiter.</td></tr>
<tr><td>Tracking processing</td><td>Whether to execute tracking processing on input variable CASIN_T is described in the following table:
<table>
<tr><th>Condition
Tracking flag (TRK)</th><th>Result</th></tr>
<tr><td>1</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>0</td><td>Input variable CASIN_T does not perform tracking.</td></tr>
</table></td></tr>
</table>

Processing Operation

Control mode \ Processing	Engineering value conversion	Variation rate limiter and high/low limiter	Output conversion	Tracking	Alarm
MAN, CMV	○	×	○	○ (*1)	○ (*3)
AUT, CAS, CSV	○	○	○	○ (*2)	○ (*3)

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

*2 Tracking is executed when the tracking flag (TRK) is 1 and DMLA, MHA and MLA of alarm (ALM) occur.

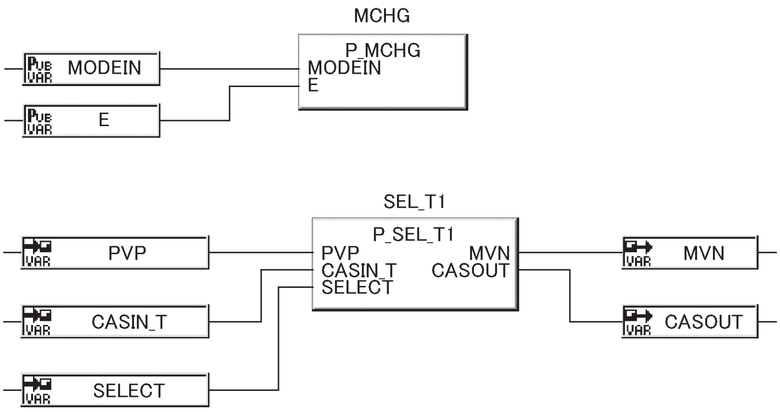
*3 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

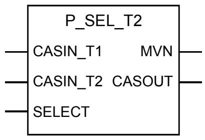
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.28 Loop Selector (With Tracking to primary loop) (P_SEL_T2)

FB	FBD parts
P_SEL_T2	

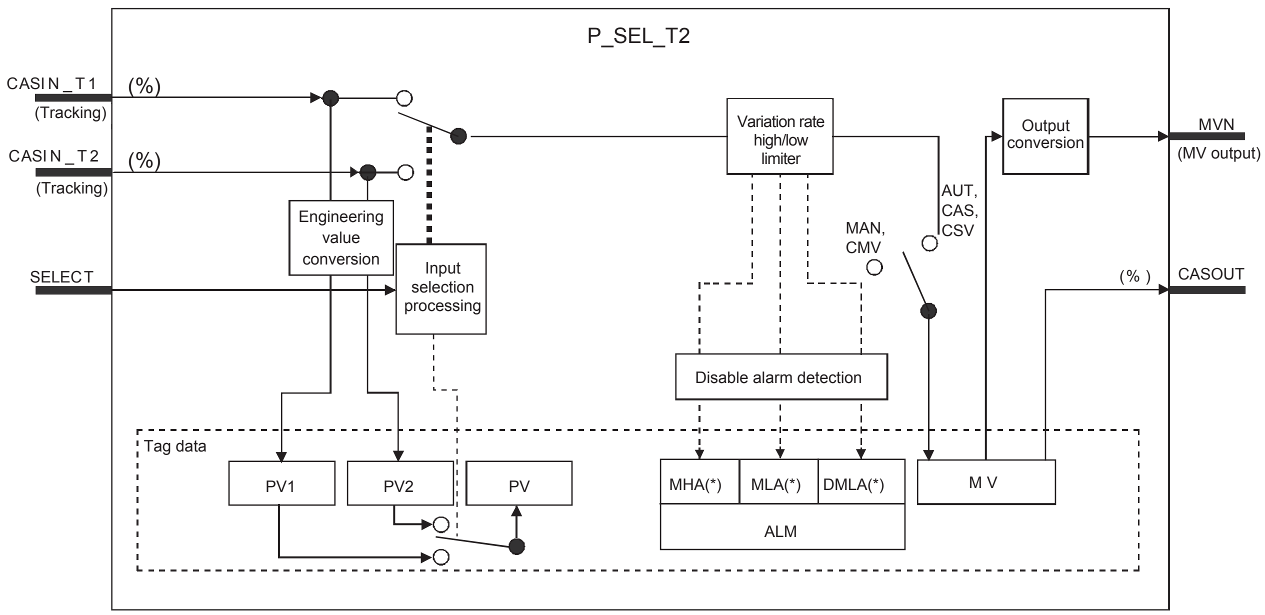
Corresponding tag type
SEL

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Selects the input value according to selection signal and output it. (With tracking)

Function/FB classification name: Tag access FB_loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	CASIN_T1	Input variable	ADR_REAL	PV input (unit: %) (With tracking)	0 to 100
	CASIN_T2	Input variable	ADR_REAL	PV input (unit: %) (With tracking)	0 to 100
	SELECT	Input variable	BOOL	Selection signal (TRUE: CASIN_T2, FALSE: CASIN_T1)	TRUE, FALSE
Output	MVN	Output variable	REAL	Output MV to module FB	NMIN to NMAX
	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100

Public Variable (Operation Constant)

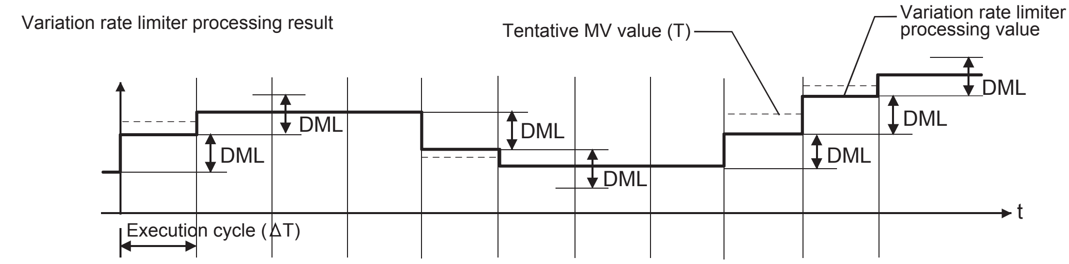
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	TRK(*1)	Public variable	INT	Tracking flag (0:Not execute, 1:Execute)	0 to 1	0	User
	SVPTN_B1	Public variable	BOOL	CASIN_T1 used (TRUE: Not use, FALSE: Use)	TRUE, FALSE	TRUE	User
	SVPTN_B2	Public variable	BOOL	CASIN_T2 used (TRUE: Not use, FALSE: Use)	TRUE, FALSE	TRUE	User
	SVPTN_B3	Public variable	BOOL	CASIN_T1 pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TRUE	User
	SVPTN_B4	Public variable	BOOL	CASIN_T2 pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TRUE	User

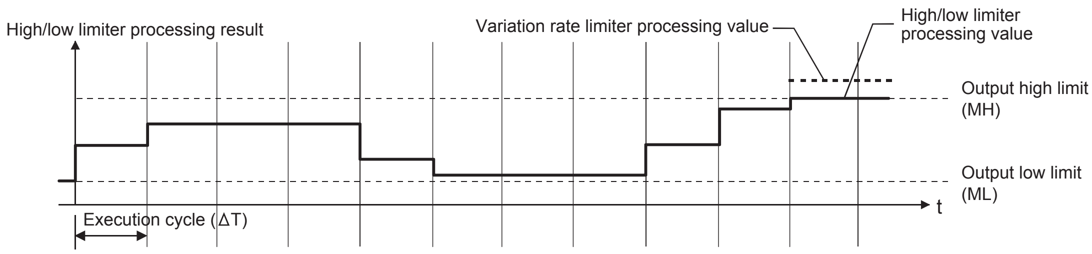
*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Tag Data

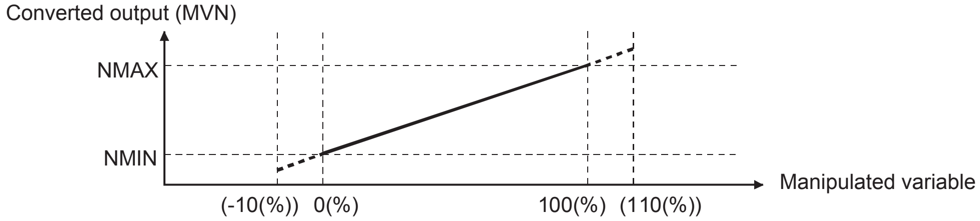
For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input selection processing</td><td>
<table>
<tr><th>Selection signal</th><th>Result</th></tr>
<tr><td>FALSE</td><td>Select output CASIN_T1 (%)</td></tr>
<tr><td>TRUE</td><td>Select output CASIN_T2 (%)</td></tr>
</table>
</td></tr>
<tr><td>Variation rate high/low limiter</td><td>
Execute variation rate limiter and high/low limiter check to input value.
● Variation rate limiter


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">Variation rate limiter processing result</th><th>Alarm (ALM)</th></tr>
<tr><th>Output variation rate limit (DMLA)</th></tr>
<tr><td>|T-MV| ≦ DML</td><td>T</td><td>FALSE (recover)</td></tr>
<tr><td>T-MV > DML</td><td>MV + DML</td><td>TRUE (occur)</td></tr>
<tr><td>T-MV < -DML</td><td>MV - DML</td><td>TRUE (occur)</td></tr>
</table>
T: Tentative MV value, MV: Manipulated variable, DML: High limit of output change
● High/low limiter


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter processing result</th><th colspan="2">Alarm (ALM)</th></tr>
<tr><th>Output low limit (MLA)</th><th>Output high limit (MHA)</th></tr>
<tr><td>Variation rate limiter processing result > MH</td><td>MH</td><td>FALSE (recover)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter processing result < ML</td><td>ML</td><td>TRUE (occur)</td><td>FALSE (recover)</td></tr>
<tr><td>ML ≦ Variation rate limiter processing result ≦ MH</td><td>Processing value of variation rate limiter</td><td>FALSE (recover)</td><td>FLALSE (recover)</td></tr>
</table>
MH: Output high value, ML: Output low value
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Engineering value conversion</td><td>Convert the input value (%) to engineering value.
$$PV_n = \frac{RH-RL}{100} \times PV_n\ (\%) + RL$$
RH: Engineering value high limit, RL: Engineering value low limit, PVPn: PV input value (%), PVn: PV1, PV2</td></tr>
<tr><td>Output conversion</td><td>Execute output conversion processing.


$$\text{Converted output (MVN)} = \{(NMAX - NMIN) \times \frac{MV}{100}\} + NMIN$$
NMAX: High limit value of output conversion, NMIN: Low limit value of output conversion, MV: Manipulated variable (%) MVN: Output value of converted output</td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether Enable Alarm Detection or not in deviation check.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected.
● ERRI, DMLI, MHI, MLI
(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
</table>

Other Functions

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Loop stop processing</td><td>The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE.
1) Hold the output (MVN).
2) Change the control mode automatically to MANUAL.
3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occurs.
4) Alarm is not detected in variation rate limiter and high/ low limiter.</td></tr>
<tr><td>Tracking processing</td><td>Whether to execute tracking processing on input variables CASIN_T1, CASIN_T2 is described in the following table:
(1) Tracking of input variable CASIN_T1
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>CASIN_T1 used (SVPTN_B1)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td rowspan="2">Input variable CASIN_T1 performs tracking.</td></tr>
<tr><td>TRUE</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td><td>Input variable CASIN_T1 does not perform tracking.</td></tr>
</table>
(2) Tracking of input variable CASIN_T2
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>CASIN_T2 used (SVPTN_B2)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td rowspan="2">Input variable CASIN_T2 performs tracking.</td></tr>
<tr><td>TRUE</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td><td>Input variable CASIN_T2 does not perform tracking.</td></tr>
</table>
</td></tr>
</table>

Processing Operation

Control mode \ Processing	Engineering value conversion	Variation rate and high/low limiter	Output conversion	Tracking	Alarm
MAN, CMV	○	×	○	○ (*1)	○ (*3)
AUT, CAS, CSV	○	○	○	○ (*2)	○ (*3)

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

*2 Tracking is executed when the tracking flag (TRK) is 1 and DMLA, MHA and MLA of alarm (ALM) occur.

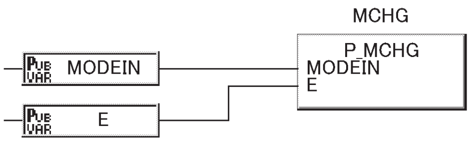
*3 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

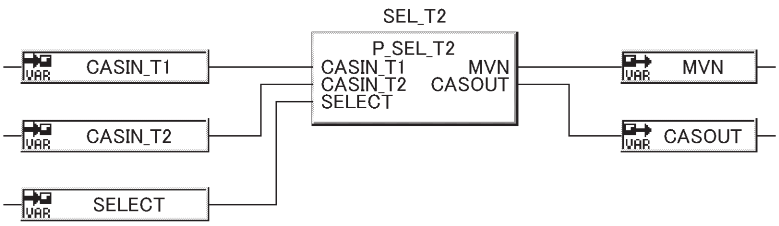
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

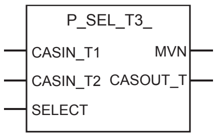




POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.29 Loop Selector (With Tracking from secondary loop to primary loop) (P_SEL_T3_)

FB	FBD parts
P_SEL_T3_	

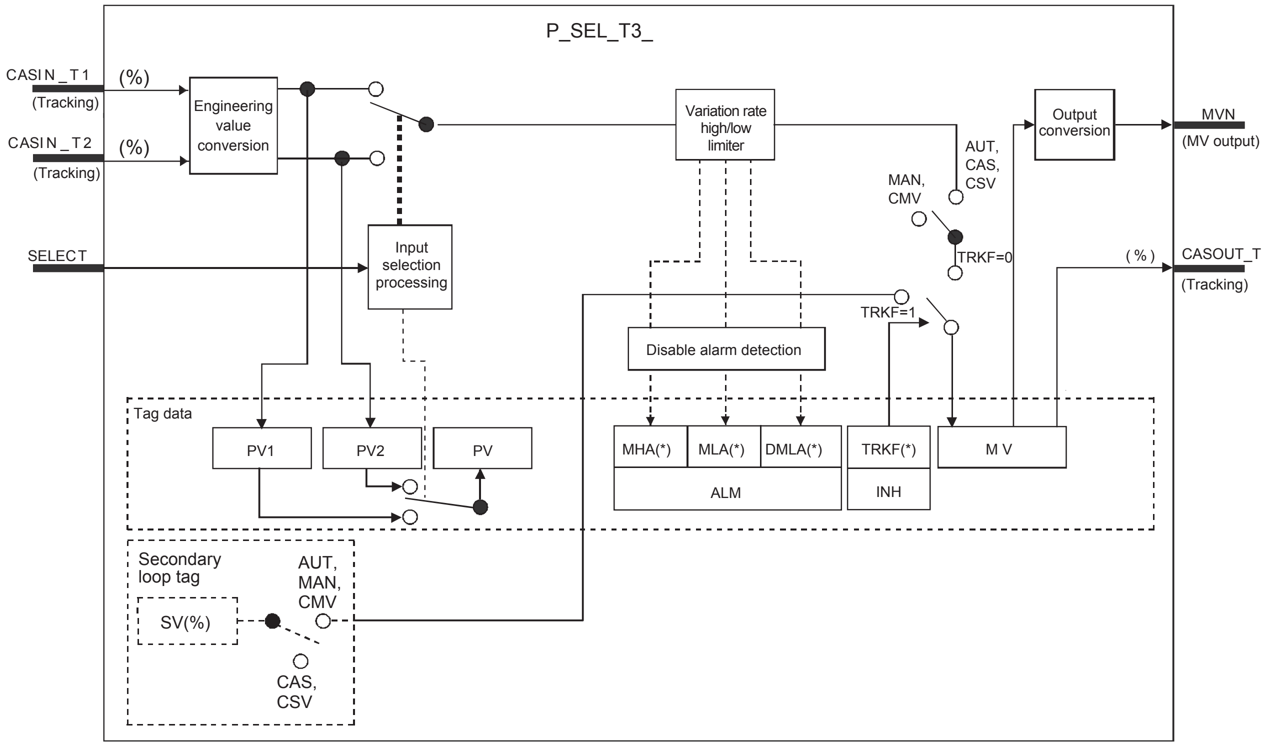
Corresponding tag type
SEL

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Selects the input value according to selection signal and then output it.

Function/FB classification name: Tag access FB_loop control operation FB

Block Diagram



* Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	CASIN_T1	Input variable	ADR_REAL	PV input (unit: %) (With tracking)	0 to 100
	CASIN_T2	Input variable	ADR_REAL	PV input (unit: %) (With tracking)	0 to 100
	SELECT	Input variable	BOOL	Selection signal (TRUE: CASIN_T2, FALSE: CASIN_T1)	TRUE, FALSE
Output	MVN	Output variable	REAL	Output MV to module FB	NMIN to NMAX
	CASOUT_T	Output variable	ADR_REAL	Cascade output (unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

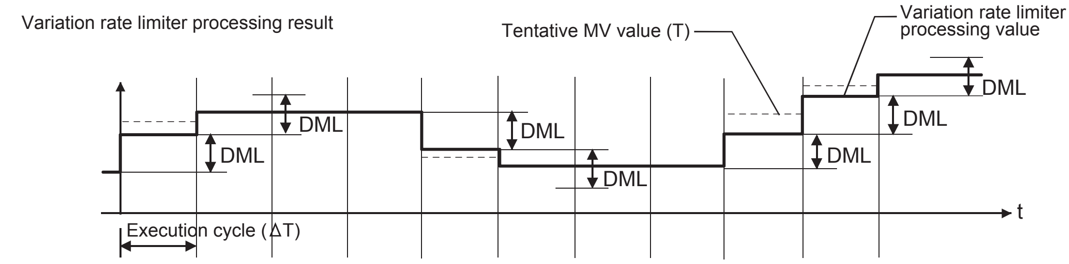
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	TRK(*1)	Public variable	INT	Tracking flag (0:Not execute, 1:Execute)	0 to 1	0	User
	SVPTN_B1	Public variable	BOOL	CASIN_T1 used (TRUE: Not use, FALSE: Use)	TRUE, FALSE	TRUE	User
	SVPTN_B2	Public variable	BOOL	CASIN_T2 used (TRUE: Not use, FALSE: Use)	TRUE, FALSE	TRUE	User
	SVPTN_B3	Public variable	BOOL	CASIN_T1 pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TRUE	User
	SVPTN_B4	Public variable	BOOL	CASIN_T2 pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TRUE	User
	SVPTN_B5	Public variable	BOOL	Tracking to Non-selected loop (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

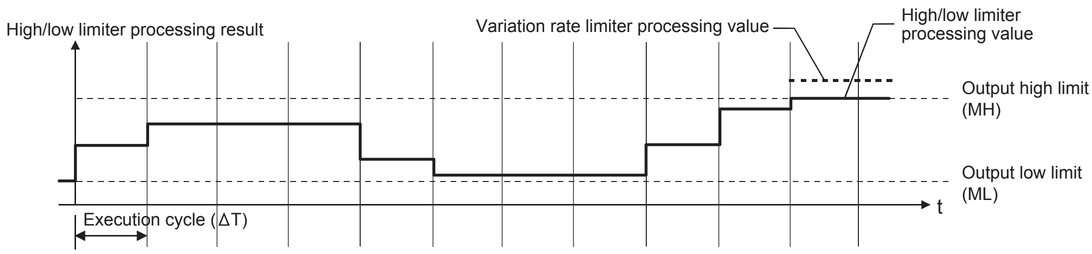
*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Tag Data

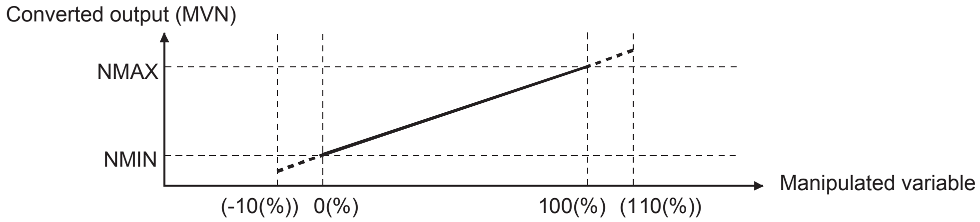
For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input selection processing</td><td>
<table>
<tr><th>Selection signal</th><th>Result</th></tr>
<tr><td>FALSE</td><td>Select output CASIN_T1 (%)</td></tr>
<tr><td>TRUE</td><td>Select output CASIN_T2 (%)</td></tr>
</table>
</td></tr>
<tr><td>Variation rate high/low limiter</td><td>
Execute variation rate limiter and high/low limiter check to input value.
● Variation rate limiter


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">Variation rate limiter processing result</th><th>Alarm (ALM)</th></tr>
<tr><th>Output variation rate limit (DMLA)</th></tr>
<tr><td>|T-MV| ≦ DML</td><td>T</td><td>FALSE (recover)</td></tr>
<tr><td>T-MV > DML</td><td>MV + DML</td><td>TRUE (occur)</td></tr>
<tr><td>T-MV < -DML</td><td>MV - DML</td><td>TRUE (occur)</td></tr>
</table>
T: Tentative MV value, MV: Manipulated variable, DML: High limit of output change
● High/low limiter


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter processing result</th><th colspan="2">Alarm (ALM)</th></tr>
<tr><th>Output low limit (MLA)</th><th>Output high limit (MHA)</th></tr>
<tr><td>Variation rate limiter processing result > MH</td><td>MH</td><td>FALSE (recover)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter processing result < ML</td><td>ML</td><td>TRUE (occur)</td><td>FALSE (recover)</td></tr>
<tr><td>ML ≦ Variation rate limiter processing result ≦ MH</td><td>Processing value of variation rate limiter</td><td>FALSE (recover)</td><td>FLALSE (recover)</td></tr>
</table>
MH: Output high value, ML: Output low value
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Engineering value conversion</td><td>Convert the input value (%) to engineering value.
$PV_n = \frac{RH-RL}{100} \times PVP_n\ (\%) + RL$
RH: Engineering value high limit, RL: Engineering value low limit, PVPn: PV input value (%), PVn: PV1, PV2</td></tr>
<tr><td>Output conversion</td><td>Execute output conversion processing.


$\text{Converted output (MVN)} = \{(NMAX - NMIN) \times \frac{MV}{100}\} + NMIN$
NMAX: High limit value of output conversion, NMIN: Low limit value of output conversion, MV: Manipulated variable (%) MVN: Output value of converted output</td></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether Enable Alarm Detection or not in deviation check.
(1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data:
If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA, MHA and MLA will not be detected.
● ERRI, DMLI, MHI, MLI
(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
</table>

Other Functions

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Loop stop processing</td><td>The following processing is executed if the stop alarm (SPA) of alarm (ALM) is TRUE.
1) Hold the output (MVN).
2) Change the control mode automatically to MANUAL.
3) Reset DMLA, MHA, and MLA when DMLA, MHA, and MLA of alarm (ALM) occurs.
4) Alarm is not detected in variation rate limiter and high/ low limiter.</td></tr>
<tr><td>Tracking processing</td><td>Whether to execute tracking processing on input variables CASIN_T1, CASIN_T2 is described in the following table:
(1) Tracking of input variable CASIN_T1
<table>
<tr><th colspan="4">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>CASIN_T1 used (SVPTN_B1)</th><th>SELECT</th><th>SVPTN_B5</th></tr>
<tr><td rowspan="4">1</td><td rowspan="3">FALSE</td><td>FALSE</td><td>(*1)</td><td rowspan="2">Input variable CASIN_T1 performs tracking.</td></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td></tr>
<tr><td>FALSE</td><td rowspan="3">Input variable CASIN_T1 does not perform tracking.</td></tr>
<tr><td>TRUE</td><td>(*1)</td><td>(*1)</td></tr>
<tr><td>0</td><td>(*1)</td><td>(*1)</td><td>(*1)</td></tr>
</table>
(2) Tracking of input variable CASIN_T2
<table>
<tr><th colspan="4">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>CASIN_T2 used (SVPTN_B2)</th><th>SELECT</th><th>SVPTN_B5</th></tr>
<tr><td rowspan="4">1</td><td rowspan="3">FALSE</td><td rowspan="2">FALSE</td><td>TRUE</td><td>Input variable CASIN_T2 performs tracking.</td></tr>
<tr><td>FALSE</td><td>Input variable CASIN_T2 does not perform tracking.</td></tr>
<tr><td>TRUE</td><td>(*1)</td><td>Input variable CASIN_T2 performs tracking</td></tr>
<tr><td>TRUE</td><td>(*1)</td><td>(*1)</td><td rowspan="2">Input variable CASIN_T2 does not perform tracking.</td></tr>
<tr><td>0</td><td>(*1)</td><td>(*1)</td><td>(*1)</td></tr>
</table>
*1 TRUE or FALSE
Set the value of program execution cycle and control cycle (CT) to be the same as that of secondary loop.</td></tr>
</table>

Processing Operation

Control mode \ Processing	Engineering value conversion	Variation rate and high/low limiter	Output conversion	Tracking	Alarm
MAN, CMV	○	×	○	○ (*1)	○ (*3)
AUT, CAS, CSV	○	○	○	○ (*2)	○ (*4)

○: Execute ×: Not execute

*1 Execute tracking to selected loop and non-selected loop when the tracking flag (TRK) is 1.

*2 Execute tracking to non-selected loop when the tracking flag (TRK) is 1.
Furthermore, execute tracking to selected loop when DMLA, MHA, MLA of alarm (ALM) occur.

*3 Restore the alarm whose corresponding bit of alarm (ALM) is TRUE (occurrence), and alarm detection will not be executed.

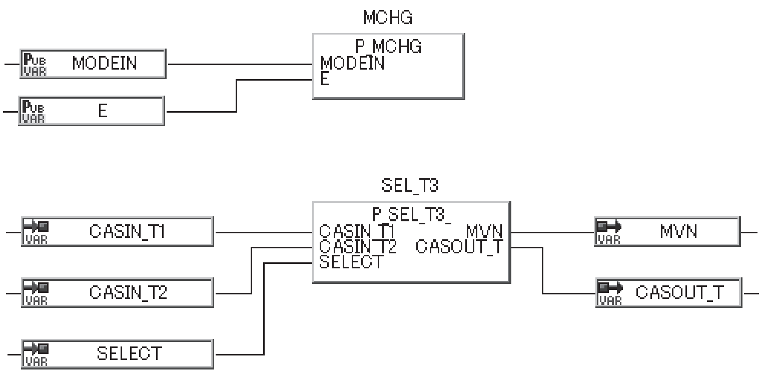
*4 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

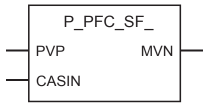
Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

8.2.30 Predictive Functional Control (Simple First Order Lag) (P_PFC_SF_)

FB	FBD parts
P_PFC_SF_	

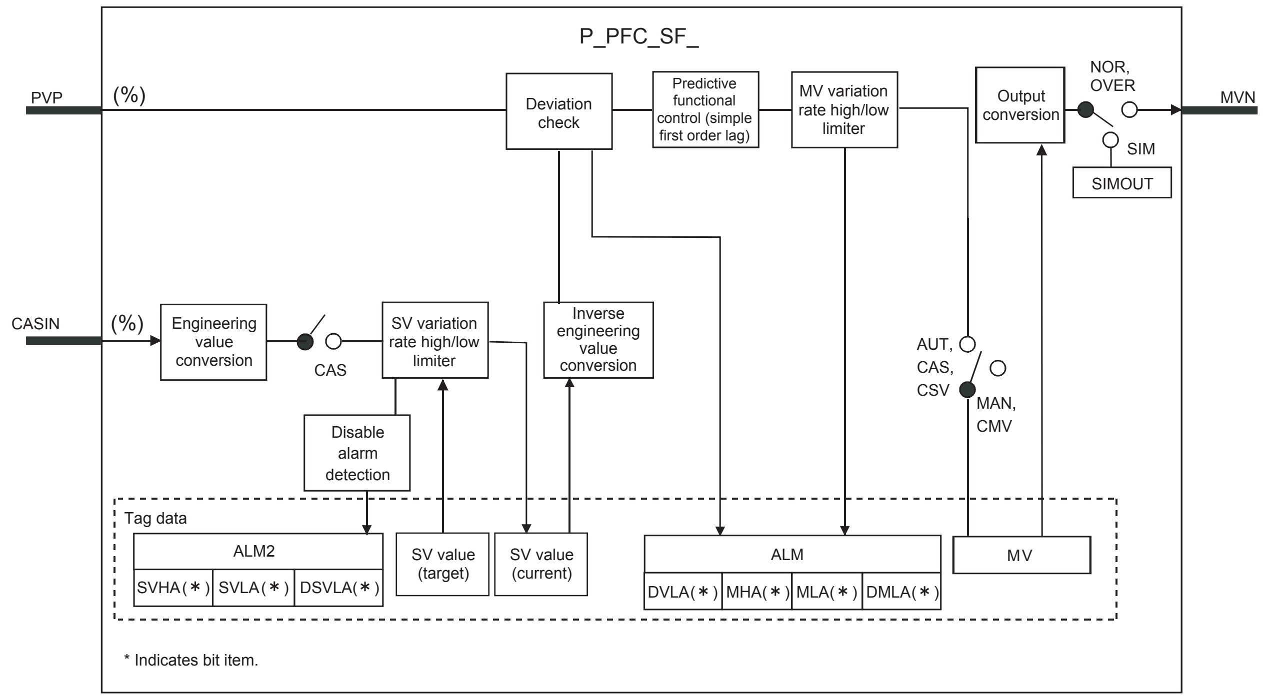
Corresponding tag type
PFC_SF

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Predicts the change in the process variable based on an internal model (first order lag), and outputs the manipulated variable so that the process variable corresponds to the setting value.

Function/FB classification name: Tag access FB_loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVN	Output variable	REAL	MV output	NMIN to NMAX

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used*1 (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	MODEL_INIT	Public variable	BOOL	Initialize Model*2 TRUE: Initialize internal model FALSE: Do not initialize internal model	TRUE, FALSE	FALSE	User
	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

*1 When SVPTN_B0 is TRUE, even if the mode is changed to the CAS mode, the CASIN input cannot be used.
*2 After changing PFC parameter, set TRUE when initializing the model which is used in PFC control.
When the variable is set to TRUE, this flag turns FALSE after the initialization of internal model has been completed in the system.

Public Variables (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System

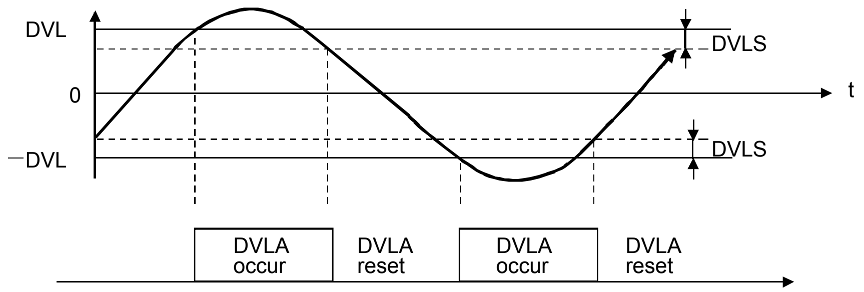
*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

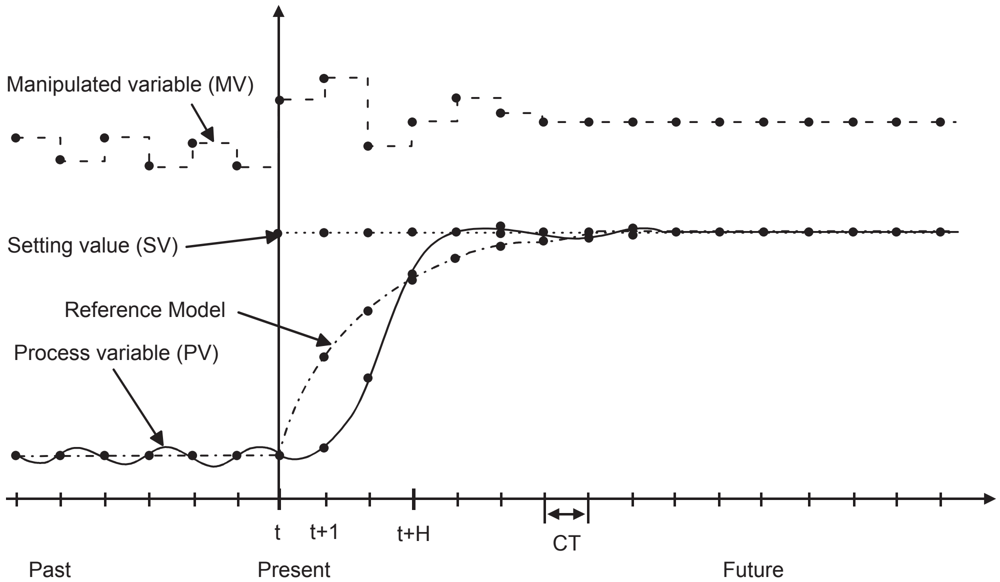
Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>(1) Execute deviation check processing.

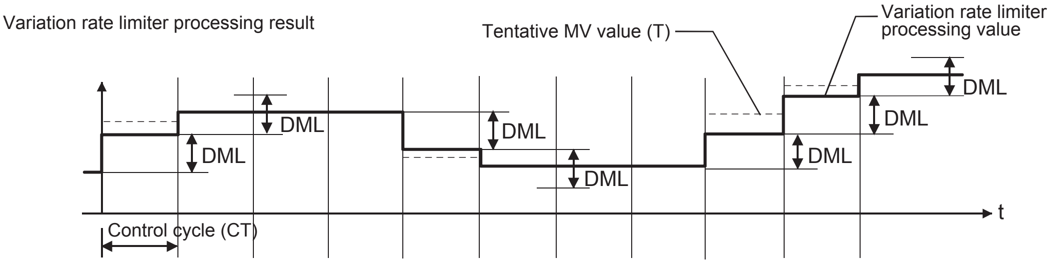


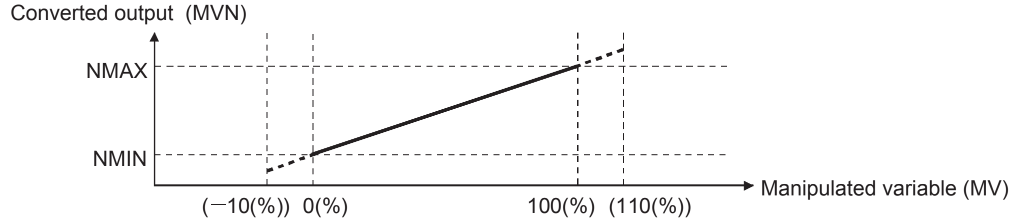
<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>

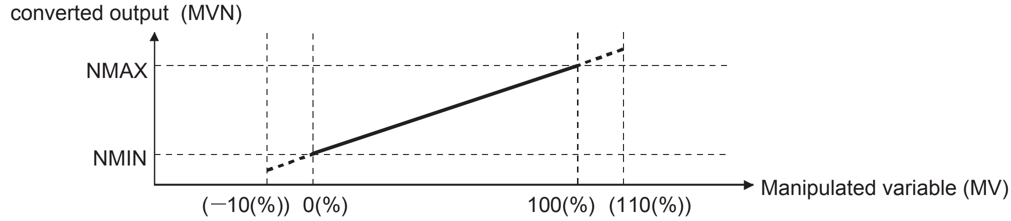
DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check (continued)</td><td>(2) Deviation for direct/reverse action (DV) is calculated as follows.
<table>
<tr><th>Condition</th><th>Deviation (DV)</th></tr>
<tr><td>Direct action (PN = 1)</td><td>DV (%) = PVP (%) – SVC (%)</td></tr>
<tr><td>Reverse action (PN = 0)</td><td>DV (%) = SVC (%) – PVP (%)</td></tr>
</table>
DV: Deviation (%), PVP (%): PV input value (%), SVC (%) = $\frac{100}{RH-RL} \times (SVC - RL)$,
RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting value (current)</td></tr>
<tr><td>Engineering value conversion</td><td>Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.
$\boxed{SV = \frac{RH-RL}{100} \times \text{Setting value (\%) from the primary loop} + RL}$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value (target)</td></tr>
<tr><td>Inverse engineering value conversion</td><td>Convert the setting value (SVC) of engineering value to percentage SVC (%).
$\boxed{SVC\ (\%) = \frac{100}{RH-RL} \times (SVC-RL)}$
RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting value (current)</td></tr>
<tr><td>Predictive functional control</td><td>Predict the change in the process variable based on an internal model (first order lag), and output the manipulated variable so that the process variable corresponds to the setting value.


PV: Process variable (%), MV: Manipulated variable (%), H: HORIZON (Coincidence Horizon), CT Control cycle</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>MV variation rate and high/low limiter</td><td>
Execute variation rate limiter and high/low limit check to the output value.
However, both variation rate limiter and high/low limiter are executed only in every control cycle.
● Variation rate limiter


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">Variation rate limiter processing result</th><th>Alarm (ALM)</th></tr>
<tr><th>Output variation rate limit (DMLA)</th></tr>
<tr><td>|T-MV| ≦ DML</td><td>T</td><td>FALSE (reset)</td></tr>
<tr><td>T-MV>DML</td><td>MV+DML</td><td>TRUE (occur)</td></tr>
<tr><td>T-MV<-DML</td><td>MV-DML</td><td>TRUE (occur)</td></tr>
</table>
T: Tentative MV value, MV: Manipulated variable, DML: Output variation rate high limit
● High/low limiter


<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter processing result</th><th colspan="2">Alarm (ALM)</th></tr>
<tr><th>Output low limit (MLA)</th><th>Output high limit (MHA)</th></tr>
<tr><td>Variation rate limiter processing result > MH</td><td>MH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter processing result < ML</td><td>ML</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>ML ≦ Variation rate limiter processing result ≦ MH</td><td>Variation rate limiter processing value</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
MH: Output high limit, ML: Input low limit
</td></tr>
<tr><td>Output conversion</td><td>
Conversion processing output is carried out.


Converted output (MVN)={(NMAX-NMIN) $\times \frac{MV}{100}$ }+NMIN
NMAX: Converted output high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%),
MVN: Converted output value
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether Enable Alarm Detection or not in deviation check, MV variation rate high/low limiter, and SV variation rate high/low limiter.
(1) Disable Alarm Detection with the settings of "Disable Alarm Detection" and "Disable Alarm Detection 2" of tag data:
If the following items of Disable Alarm Detection (INH) / Disable Alarm Detection 2 (INH2) of tag data are TRUE, the DVLA, DMLA, MHA and MLA of alarm (ALM) and the DSVLA, SVHA, and SVLA of alarm 2 (ALM2) will not be detected.
● ERRI, DVLI, DMLI, MHI, MLI, DSVLI, SVHI, SVLI
(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
<tr><td>SV variation rate high/low limiter</td><td>Checks variation rate high/low limiter to SV value (target value) in every control cycle (CT).
(1) Variation rate limiter
● The control mode is AUT or CAS or CSV.
SV variation rate high limit value inputted in % is converted to engineering value and the processing will be executed.
DSVL → DSVLT (DSVL: SV variation rate high limit value, DSVLT: value converted to engineering value from SV variation rate high limit value)
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>| SV - SVC | ≦ DSVLT</td><td>SV</td><td>FALSE (reset)</td></tr>
<tr><td>SV - SVC > DSVLT</td><td>SVC + DSVLT</td><td>TRUE (occur)</td></tr>
<tr><td>SV - SVC < - DSVLT</td><td>SVC - DSVLT</td><td>TRUE (occur)</td></tr>
</table>
SV: Setting value (target), SVC: Setting value (current)
If DSVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, DSVLA will be FALSE.
● The control mode is MAN or CMV.
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>No</td><td>SV</td><td>FALSE (reset)</td></tr>
</table>
(2) High/low limiter
● When SVLMT_EN is TRUE.
<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter result</th><th colspan="2">Alarm2 (ALM2)</th></tr>
<tr><th>Target low limit (SVLA)</th><th>Target upper limit (SVHA)</th></tr>
<tr><td>Variation rate limiter result > SH</td><td>SH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter result < SL</td><td>SL</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>SL ≦ variation rate limiter result ≦ SH</td><td>Variation rate limiter result</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
If SVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVLA will be FALSE.
If SVHI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVHA will be FALSE.
High/low limiter result is stored to SVC (setting value (current)).
● When SVLMT_EN is FALSE.
Variation rate limiter result is stored to SVC (setting value (current)).</td></tr>
</table>

Other Function

Item	Contents
Loop stop processing	The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset DVLA, DMLA, MHA, and MLA when DVLA, DMLA, MHA, and MLA of alarm (ALM) occurs. Reset DSVLA, SVLA and SVHA when DSVLA, SVLA and SVHA of alarm2 (ALM2) occurs. 4) Alarm is not detected in deviation check, MV variation rate high/low limiter, and SV variation rate high/low limiter.

Processing Operation

Control mode \ Processing	Deviation check	Predictive functional control	Engineering value conversion	Inverse engineering value conversion	MV variation rate limiter high/low limiter	Output conversion	Alarm	SV variation rate limiter high/low limiter
MAN, CMV	○	○	×	○	×	○	○ (*1)	○ (*2)
AUT	○	○	×	○	○	○	○ (*1)	○
CAS, CSV	○	○	○	○	○	○	○ (*1)	○

○: Execute ×: Not execute

*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

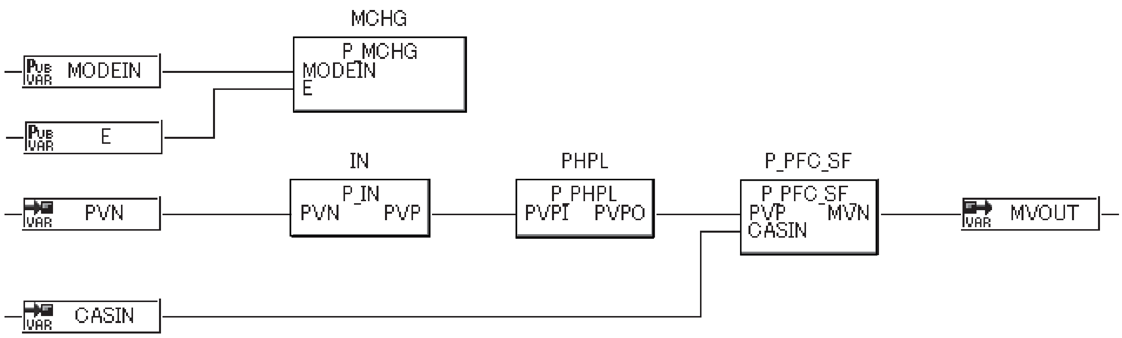
*2 When the control mode is MAN, SV variation rate limiter processing is not executed.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

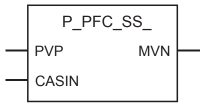
Program Example



POINT

- It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.
- After changing the value of control cycle (CT) or changing the value of Dead time (DM), Gain (KM), or Time contrast (TM) significantly, initialize the model by turning Initialize Model (MODEL_INIT) TRUE from FALSE.
  Note that, however, if the model initialization is performed during the process is not stable (MV is fluctuating), the control may not stable until the dead time is passed.
- When initializing a predictive functional control FB used in the project created with PX Developer version 1.34L or earlier after opening the project using PX Developer 1.42U or later, compilation (cold-start compile/hot-start compile/compile (online change)) is required.

8.2.31 Predictive Functional Control (Simple Second Order Lag) (P_PFC_SS_)

FB	FBD parts
P_PFC_SS_	

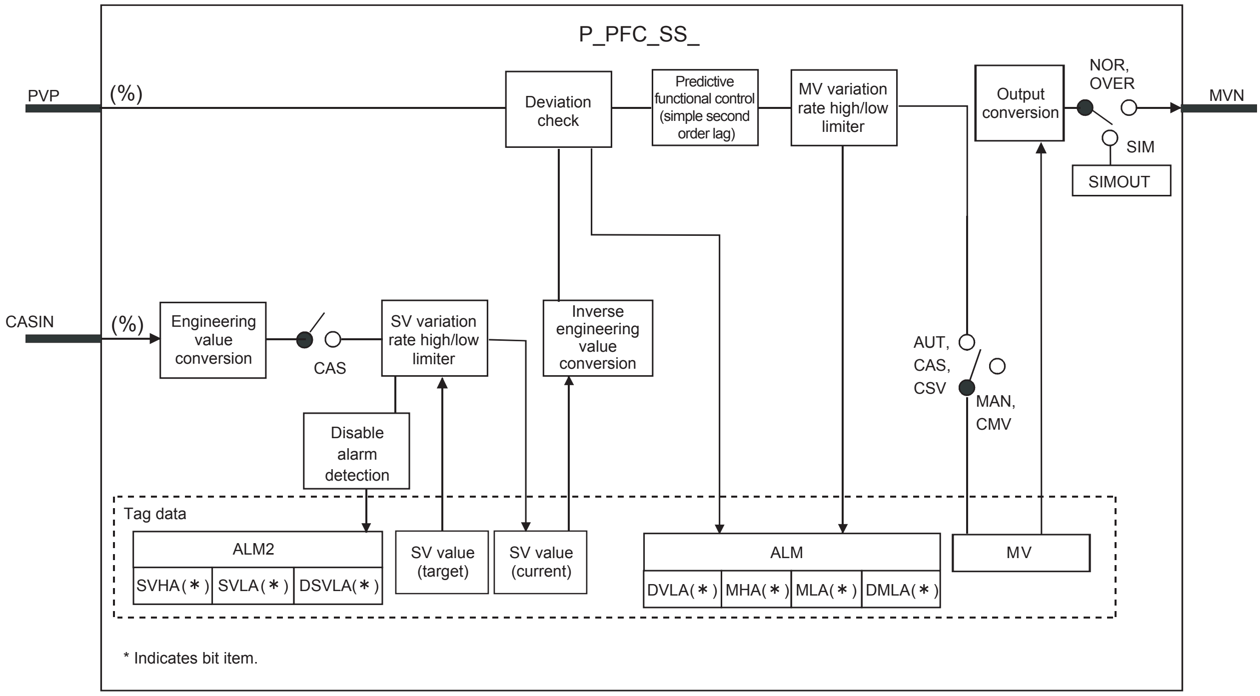
Corresponding tag type
PFC_SS

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Predicts the change in the process variable based on an internal model (second order lag), and outputs the manipulated variable so that the process variable corresponds to the setting value.

Function/FB classification name: Tag access FB_loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVN	Output variable	REAL	MV output	NMIN to NMAX

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used*1 (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	MODEL_INIT	Public variable	BOOL	Initialize Model*2 TRUE: Initialize internal model FALSE: Do not initialize internal mode	TRUE, FALSE	FALSE	User
	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

*1 When SVPTN_B0 is TRUE, even if the mode is changed to the CAS mode, the CASIN input cannot be used.
*2 After changing PFC parameter, set TRUE when initializing the model which is used in PFC control.
When the variable is set to TRUE, this flag turns FALSE after the initialization of internal model has been completed in the system.

Public Variables (Others) (*1)

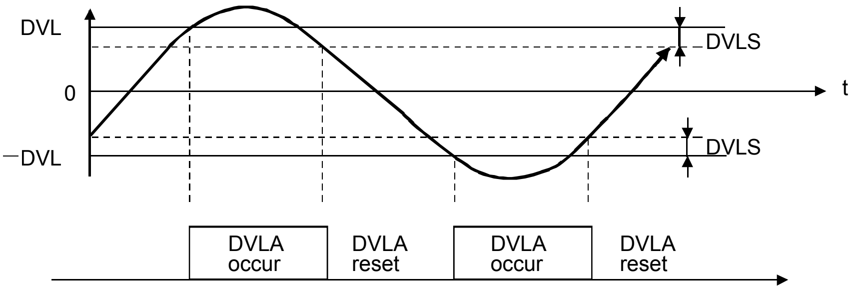
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.

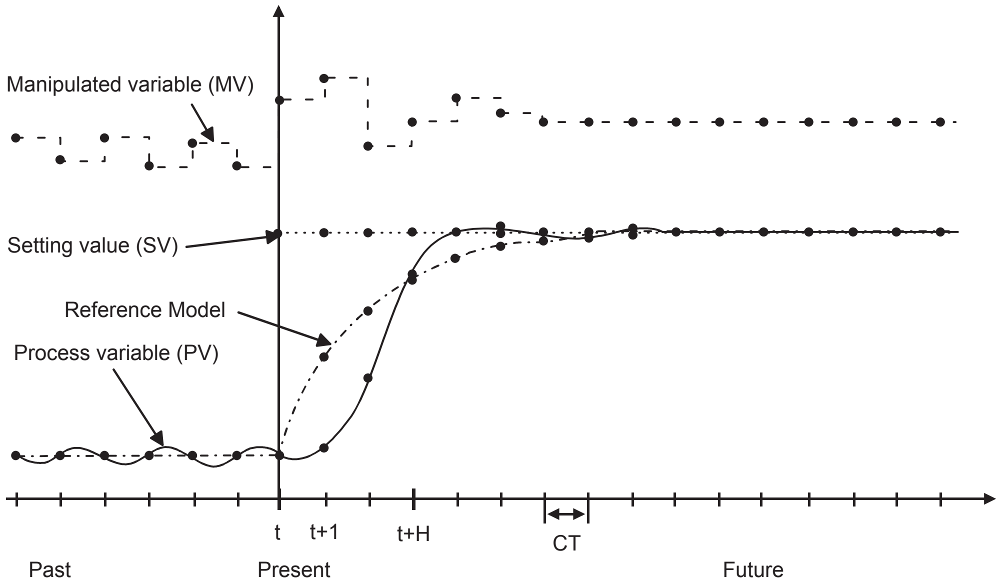
Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
(1) Execute deviation check processing.


<table>
<tr><td rowspan="2">Condition</td><td>Alarm (ALM)</td></tr>
<tr><td>Large deviation (DVLA)</td></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>
DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check (continued)</td><td>(2) Deviation for direct/reverse action (DV) is calculated as follows.
<table>
<tr><th>Condition</th><th>Deviation (DV)</th></tr>
<tr><td>Direct action (PN = 1)</td><td>DV (%) = PVP (%) – SVC (%)</td></tr>
<tr><td>Reverse action (PN = 0)</td><td>DV (%) = SVC (%) – PVP (%)</td></tr>
</table>
DV: Deviation (%), PVP (%): PV input value (%), SVC (%) = $\frac{100}{RH-RL} \times (SVC - RL)$,
RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting value (current)</td></tr>
<tr><td>Engineering value conversion</td><td>Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.
$$SV = \frac{RH-RL}{100} \times \text{Setting value (\%) from the primary loop} + RL$$
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value (target)</td></tr>
<tr><td>Inverse engineering value conversion</td><td>Convert the setting value (SVC) of engineering value to percentage SVC (%).
$$SVC\ (\%) = \frac{100}{RH-RL} \times (SVC-RL)$$
RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting value (current)</td></tr>
<tr><td>Predictive functional control</td><td>Predict the change in the process variable based on an internal model (second order lag), and output the manipulated variable so that the process variable corresponds to the setting value.


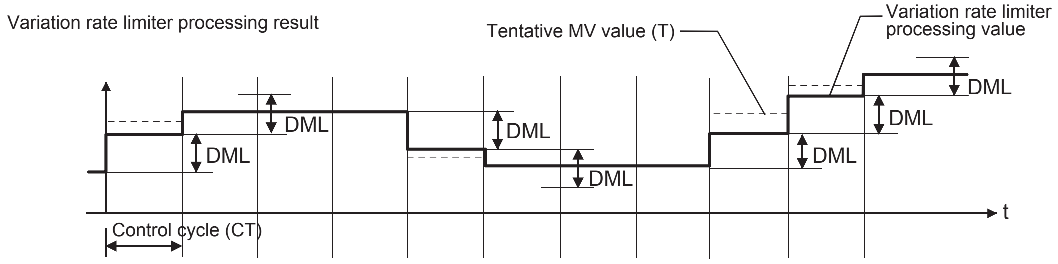
PV: Process variable (%), MV: Manipulated variable (%), H: HORIZON (Coincidence Horizon), CT Control cycle</td></tr>
</table>

Item	Contents
MV variation rate and high/low limiter	(see below)
Output conversion	(see below)

Item: MV variation rate and high/low limiter

Execute variation rate limiter and high/low limit check to the output value.
However, both variation rate limiter and high/low limiter are executed only in every control cycle.

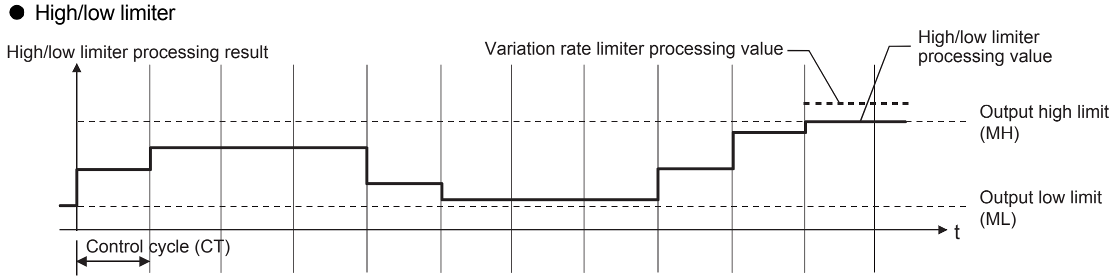
● Variation rate limiter



Condition	Variation rate limiter processing result	Alarm (ALM)
		Output variation rate limit (DMLA)
$\lvert T-MV \rvert \leqq DML$	T	FALSE (reset)
T-MV>DML	MV+DML	TRUE (occur)
T-MV<-DML	MV-DML	TRUE (occur)

T: Tentative MV value, MV: Manipulated variable, DML: Output variation rate high limit

● High/low limiter

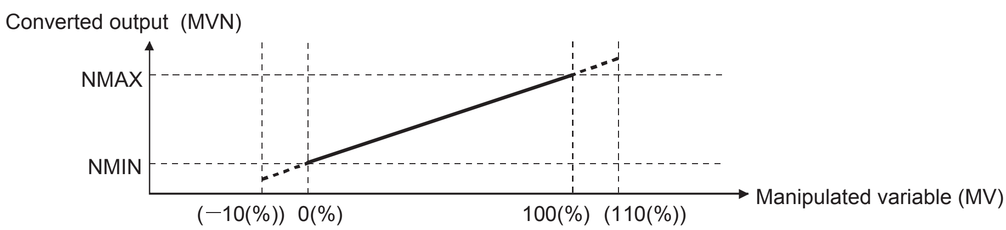


Condition	High/low limiter processing result	Alarm (ALM)	
		Output low limit (MLA)	Output high limit (MHA)
Variation rate limiter processing result > MH	MH	FALSE (reset)	TRUE (occur)
Variation rate limiter processing result < ML	ML	TRUE (occur)	FALSE (reset)
ML ≦ Variation rate limiter processing result ≦ MH	Variation rate limiter processing value	FALSE (reset)	FALSE (reset)

MH: Output high limit, ML: Input low limit

Item: Output conversion

Conversion processing output is carried out.



$$\text{Converted output (MVN)}=\{(\text{NMAX}-\text{NMIN}) \times \frac{\text{MV}}{100}\}+\text{NMIN}$$

NMAX: Converted output high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%),
MVN: Converted output value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Disable Alarm Detection</td><td>
Set whether Enable Alarm Detection or not in deviation check, MV variation rate high/low limiter, and SV variation rate high/low limiter.
(1) Disable Alarm Detection with the settings of "Disable Alarm Detection" and "Disable Alarm Detection 2" of tag data:
If the following items of Disable Alarm Detection (INH) / Disable Alarm Detection 2 (INH2) of tag data are TRUE, the DVLA, DMLA, MHA and MLA of alarm (ALM) and the DSVLA, SVHA, and SVLA of alarm 2 (ALM2) will not be detected.
● ERRI, DVLI, DMLI, MHI, MLI, DSVLI, SVHI, SVLI
(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the following contents.
</td></tr>
<tr><td>SV variation rate high/low limiter</td><td>
Checks variation rate high/low limiter to SV value (target value) in every control cycle (CT).
(1) Variation rate limiter
● The control mode is AUT or CAS or CSV.
SV variation rate high limit value inputted in % is converted to engineering value and the processing will be executed.
DSVL → DSVLT (DSVL: SV variation rate high limit value, DSVLT: value converted to engineering value from SV variation rate high limit value)
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>| SV - SVC | ≦ DSVLT</td><td>SV</td><td>FALSE (reset)</td></tr>
<tr><td>SV - SVC > DSVLT</td><td>SVC + DSVLT</td><td>TRUE (occur)</td></tr>
<tr><td>SV - SVC < - DSVLT</td><td>SVC - DSVLT</td><td>TRUE (occur)</td></tr>
</table>
SV: Setting value (target), SVC: Setting value (current)
If DSVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, DSVLA will be FALSE.
● The control mode is MAN or CMV.
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>No</td><td>SV</td><td>FALSE (reset)</td></tr>
</table>
(2) High/low limiter
● When SVLMT_EN is TRUE.
<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter result</th><th colspan="2">Alarm2 (ALM2)</th></tr>
<tr><th>Target low limit (SVLA)</th><th>Target upper limit (SVHA)</th></tr>
<tr><td>Variation rate limiter result > SH</td><td>SH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter result < SL</td><td>SL</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>SL ≦ variation rate limiter result ≦ SH</td><td>Variation rate limiter result</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
If SVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVLA will be FALSE.
If SVHI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVHA will be FALSE.
High/low limiter result is stored to SVC (setting value (current)).
● When SVLMT_EN is FALSE.
Variation rate limiter result is stored to SVC (setting value (current)).
</td></tr>
</table>

Other Function

Item	Contents
Loop stop processing	The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset DVLA, DMLA, MHA, and MLA when DVLA, DMLA, MHA, and MLA of alarm (ALM) occurs. Reset DSVLA, SVLA and SVHA when DSVLA, SVLA and SVHA of alarm2 (ALM2) occurs. 4) Alarm is not detected in deviation check, MV variation rate high/low limiter, and SV variation rate high/low limiter.

Processing Operation

Processing / Control mode	Deviation check	Predictive functional control	Engineering value conversion	Inverse engineering value conversion	MV variation rate limiter high/low limiter	Output conversion	Alarm	SV variation rate limiter high/low limiter
MAN, CMV	○	○	×	○	×	○	○ (*1)	○ (*2)
AUT	○	○	×	○	○	○	○ (*1)	○
CAS, CSV	○	○	○	○	○	○	○ (*1)	○

○: Execute ×: Not execute

*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

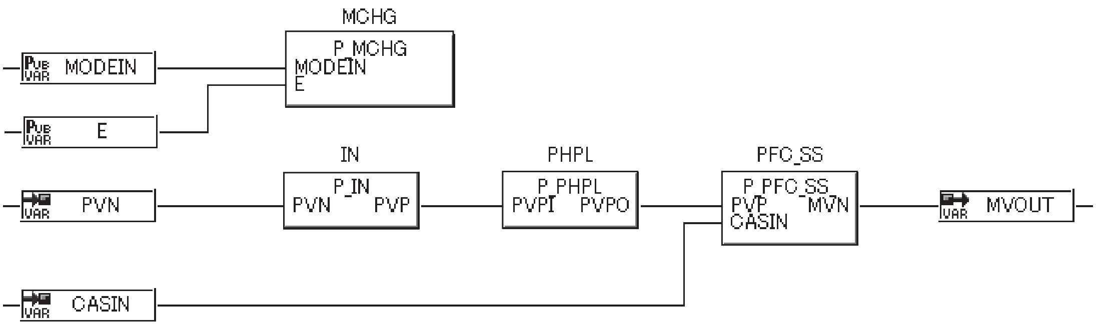
*2 When the control mode is MAN, SV variation rate limiter processing is not executed.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

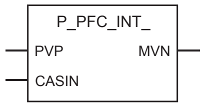
Program Example



POINT

- It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.
- After changing the value of control cycle (CT) or changing the value of Dead time (DM), Gain (KM), or Time contrast (TM1), or Time contrast (TM2) significantly, initialize the model by turning Initialize Model (MODEL_INIT) TRUE from FALSE. Note that, however, if the model initialization is performed during the process is not stable (MV is fluctuating), the control may not stable until the dead time is passed.
- When initializing a predictive functional control FB used in the project created with PX Developer version 1.34L or earlier after opening the project using PX Developer 1.42U or later, compilation (cold-start compile/hot-start compile/compile (online change)) is required.

8.2.32 Predictive Functional Control (Integral Process) (P_PFC_INT_)

FB	FBD parts
P_PFC_INT_	

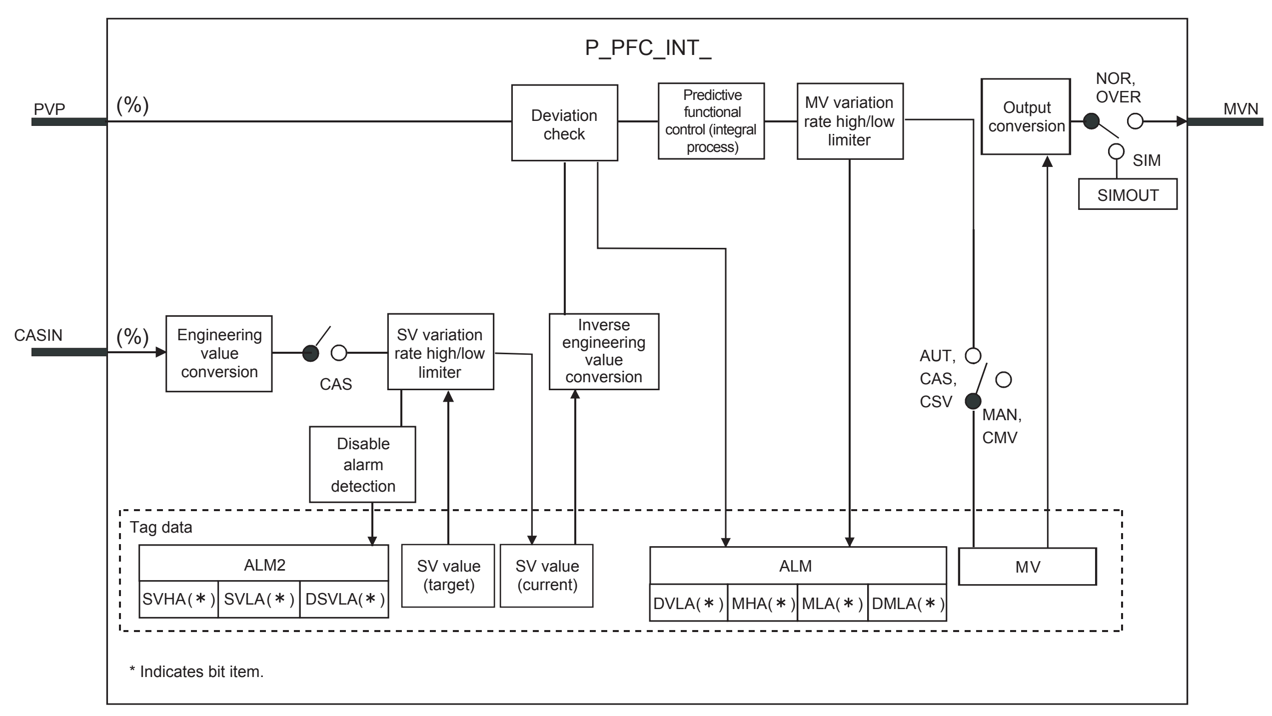
Corresponding tag type
PFC_INT

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Predicts the change in the process variable based on an internal model (integral process), and outputs the manipulated variable so that the process variable corresponds to the setting value.

Function/FB classification name: Tag access FB_loop control operation FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (unit: %)	0 to 100
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	MVN	Output variable	REAL	MV output	(2 × NMIN – NMAX) to NMAX

Public Variable (Operation constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used*1 (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	MODEL_INIT	Public variable	BOOL	Initialize Model*2 TRUE: Initialize internal model FALSE: Do not initialize internal mode	TRUE, FALSE	FALSE	User
	NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

*1 When SVPTN_B0 is TRUE, even if the mode is changed to the CAS mode, the CASIN input cannot be used.
*2 After changing PFC parameter, set TRUE when initializing the model which is used in PFC control.
When the variable is set to TRUE, this flag turns FALSE after the initialization of internal model has been completed in the system.

Public Variables (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM*2	SIMOUT	Public variable	REAL	Simulation output	(2 × NMIN – NMAX) to NMAX	0.0	System

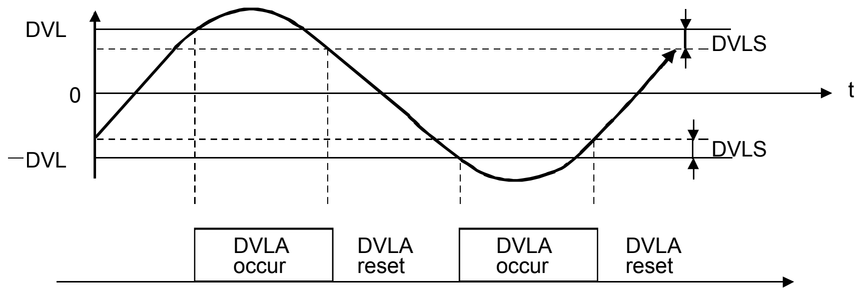
*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

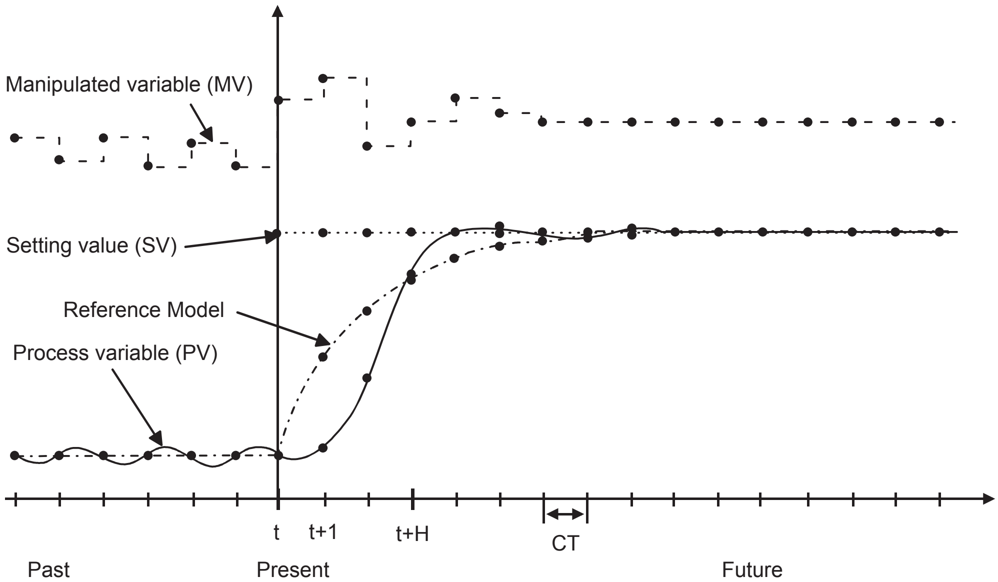
Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>
(1) Execute deviation check processing.



<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV| ≦ (DVL-DVLS)</td><td>FALSE (reset)</td></tr>
</table>

DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check (continued)</td><td>(2) Deviation for direct/reverse action (DV) is calculated as follows.
<table><tr><th>Condition</th><th>Deviation (DV)</th></tr><tr><td>Direct action (PN = 1)</td><td>DV (%) = PVP (%) – SVC (%)</td></tr><tr><td>Reverse action (PN = 0)</td><td>DV (%) = SVC (%) – PVP (%)</td></tr></table>DV: Deviation (%), PVP (%): PV input value (%), SVC (%) = $\frac{100}{RH-RL} \times (SVC - RL)$,
RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting value (current)</td></tr>
<tr><td>Engineering value conversion</td><td>Convert the setting value (%), which is from primary loop control when the control mode is CAS or CSV, to engineering value.
$SV = \frac{RH-RL}{100} \times$ Setting value (%) from the primary loop +RL
RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value (target)</td></tr>
<tr><td>Inverse engineering value conversion</td><td>Convert the setting value (SVC) of engineering value to percentage SVC (%).
$SVC\ (\%) = \frac{100}{RH-RL} \times (SVC-RL)$
RH: Engineering value high limit, RL: Engineering value low limit, SVC: Setting value (current)</td></tr>
<tr><td>Predictive functional control</td><td>Predict the change in the process variable based on an internal model (integral process), and output the manipulated variable so that the process variable corresponds to the setting value.


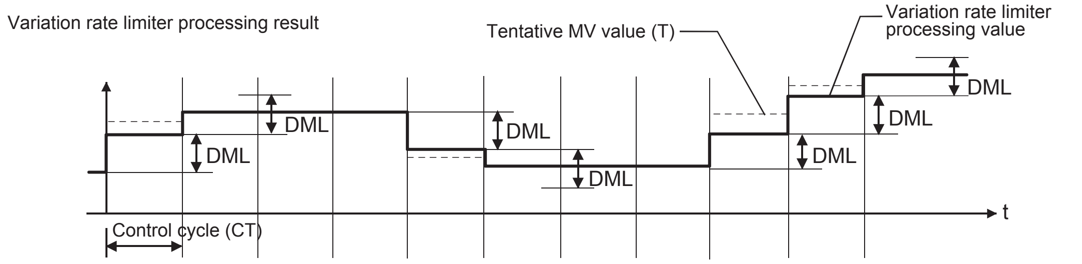
PV: Process variable (%), MV: Manipulated variable (%), H: HORIZON (Coincidence Horizon), CT Control cycle</td></tr>
</table>

Item	Contents

MV variation rate and high/low limiter

Execute variation rate limiter and high/low limit check to the output value.
However, both variation rate limiter and high/low limiter are executed only in every control cycle.

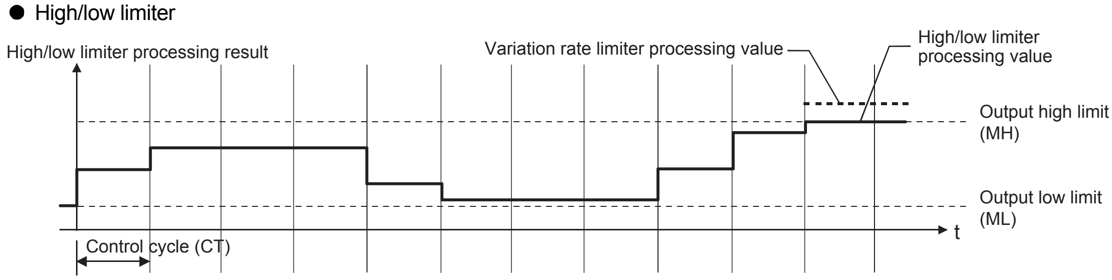
● Variation rate limiter



Condition	Variation rate limiter processing result	Alarm (ALM)
		Output variation rate limit (DMLA)
|T-MV| ≦ DML	T	FALSE (reset)
T-MV>DML	MV+DML	TRUE (occur)
T-MV<-DML	MV-DML	TRUE (occur)

T: Tentative MV value, MV: Manipulated variable, DML: Output variation rate high limit

● High/low limiter

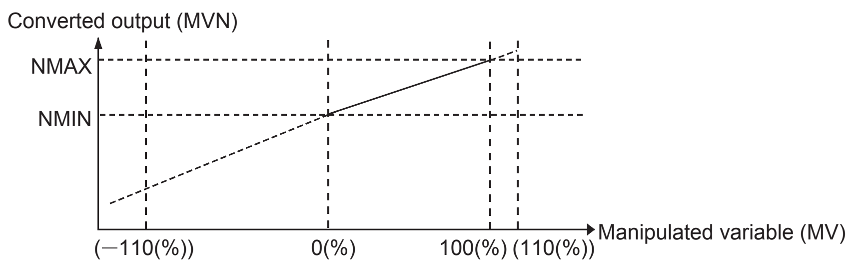


Condition	High/low limiter processing result	Alarm (ALM)	
		Output low limit (MLA)	Output high limit (MHA)
Variation rate limiter processing result > MH	MH	FALSE (reset)	TRUE (occur)
Variation rate limiter processing result < ML	ML	TRUE (occur)	FALSE (reset)
ML ≦ Variation rate limiter processing result ≦ MH	Variation rate limiter processing value	FALSE (reset)	FALSE (reset)

MH: Output high limit, ML: Output low limit

Output conversion

Conversion processing output is carried out.



$$\text{Converted output (MVN)}=\{(\text{NMAX-NMIN})\times\frac{\text{MV}}{100}\}+\text{NMIN}$$

NMAX: Converted output high limit, NMIN: Output conversion low limit, MV: Manipulated variable (%), MVN: Converted output value

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Disable Alarm Detection</td><td>Set whether Enable Alarm Detection or not in deviation check, MV variation rate high/low limiter, and SV variation rate high/low limiter.
(1) Disable Alarm Detection with the settings of "Disable Alarm Detection" and "Disable Alarm Detection 2" of tag data:
If the following items of Disable Alarm Detection (INH) /Disable Alarm Detection 2 (INH2) of tag data are TRUE, the DVLA, DMLA, MHA and MLA of alarm (ALM) and the DSVLA, SVHA, and SVLA of alarm 2 (ALM2) will not be detected.
● ERRI, DVLI, DMLI, MHI, MLI, DSVLI, SVHI, SVLI
(2) Disable Alarm Detection by loop stop processing:
Please refer to loop stop processing in the following contents.</td></tr>
<tr><td>SV variation rate high/low limiter</td><td>Checks variation rate high/low limiter to SV value (target value) in every control cycle (CT).
(1) Variation rate limiter
● The control mode is AUT or CAS or CSV.
SV variation rate high limit value inputted in % is converted to engineering value and the processing will be executed.
DSVL → DSVLT (DSVL: SV variation rate high limit value, DSVLT: value converted to engineering value from SV variation rate high limit value)
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>| SV - SVC | ≦ DSVLT</td><td>SV</td><td>FALSE (reset)</td></tr>
<tr><td>SV - SVC > DSVLT</td><td>SVC + DSVLT</td><td>TRUE (occur)</td></tr>
<tr><td>SV - SVC < - DSVLT</td><td>SVC - DSVLT</td><td>TRUE (occur)</td></tr>
</table>
SV: Setting value (target), SVC: Setting value (current)
If DSVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, DSVLA will be FALSE.
● The control mode is MAN or CMV.
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>No</td><td>SV</td><td>FALSE (reset)</td></tr>
</table>
(2) High/low limiter
● When SVLMT_EN is TRUE.
<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter result</th><th colspan="2">Alarm2 (ALM2)</th></tr>
<tr><th>Target low limit (SVLA)</th><th>Target upper limit (SVHA)</th></tr>
<tr><td>Variation rate limiter result > SH</td><td>SH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter result < SL</td><td>SL</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>SL ≦ variation rate limiter result ≦ SH</td><td>Variation rate limiter result</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
If SVLI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVLA will be FALSE.
If SVHI of disable alarm2 detection or ERRI of disable alarm detection is TRUE, SVHA will be FALSE.
High/low limiter result is stored to SVC (setting value (current)).
● When SVLMT_EN is FALSE.
Variation rate limiter result is stored to SVC (setting value (current)).</td></tr>
</table>

Other Function

Item	Contents
Loop stop processing	The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE. 1) Hold the output (MVN). 2) Change the control mode automatically to MANUAL. 3) Reset DVLA, DMLA, MHA, and MLA when DVLA, DMLA, MHA, and MLA of alarm (ALM) occurs. Reset DSVLA, SVLA and SVHA when DSVLA, SVLA and SVHA of alarm2 (ALM2) occurs. 4) Alarm is not detected in deviation check, MV variation rate high/low limiter, and SV variation rate high/low limiter.

Processing Operation

Processing / Control mode	Deviation check	Predictive functional control	Engineering value conversion	Inverse engineering value conversion	MV variation rate limiter high/low limiter	Output conversion	Alarm	SV variation rate limiter high/low limiter
MAN, CMV	○	○	×	○	×	○	○ (*1)	○ (*2)
AUT	○	○	×	○	○	○	○ (*1)	○
CAS, CSV	○	○	○	○	○	○	○ (*1)	○

○: Execute ×: Not execute

*1 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

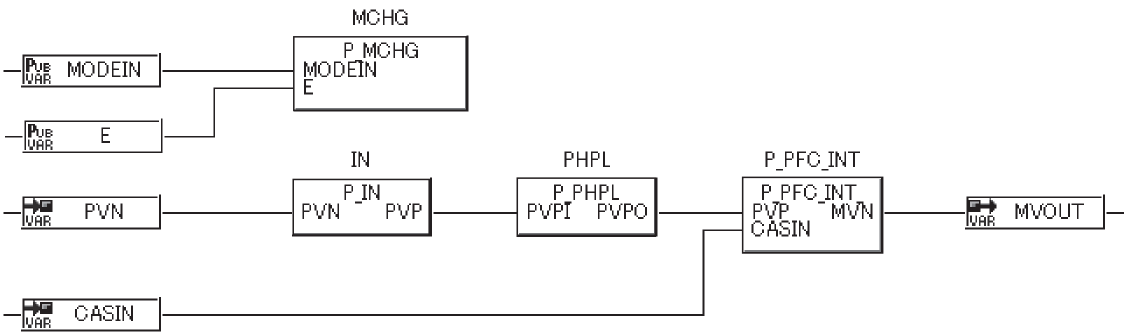
*2 When the control mode is MAN, SV variation rate limiter processing is not executed.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

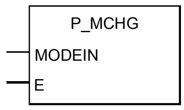


POINT

- It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with program in the FB property window.
- After changing the value of control cycle (CT) or changing the value of Dead time (DM), Gain (KM) significantly, initialize the model by turning Initialize Model (MODEL_INIT) TRUE from FALSE.
  Note that, however, if the model initialization is performed during the process is not stable (MV is not 0% but output), the control may not stable until the dead time is passed.
- When initializing a predictive functional control FB used in the project created with PX Developer version 1.34L or earlier after opening the project using PX Developer 1.42U or later, compilation (cold-start compile/hot-start compile/compile (online change)) is required.

8.3 Tag Access FB_Tag Special FB

8.3.1 Control Mode Change (P_MCHG)

FB	FBD parts
P_MCHG	

Corresponding tag type
PID, 2PID, 2PIDH, PIDP, SPI, IPD, BPI, R, ONF2, ONF3, PFC_SF, PFC_SS, PFC_INT, PGS, PGS2, MOUT, SWM, MWM, SEL, PVAL, HTCL, NREV, REV, MVAL1, MVAL2, PB

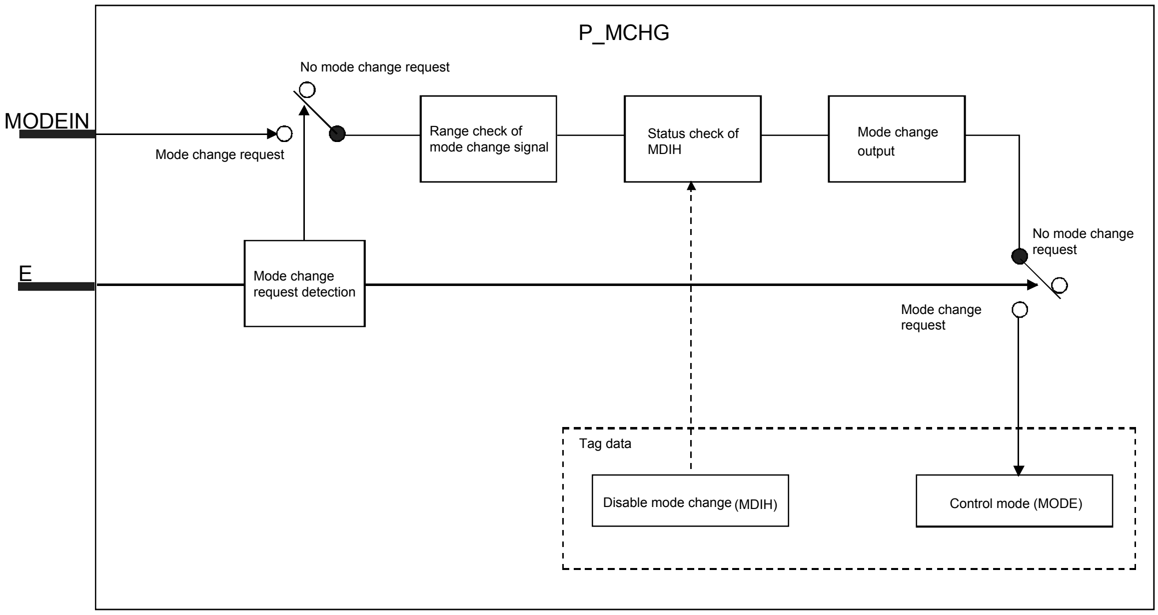
Control mode				
MAN	AUT	CAS*1	CMV	CSV
○	○	○	○	○

*1 Transition to CASDR is possible.

Function overview: Change MAN/AUT/CAS/CMV/CSV/CASDR mode corresponding to mode selection signal.

Function/FB classification name: Tag access FB_tag special FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	MODEIN	Input variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV, 6: CASDR)	1 to 6
	E	Input variable	BOOL	Mode change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE

Tag Data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Mode selection signal range check</td><td>
Check mode change signal range.
(1) The mode change signal is only valid in 1 to 6.
Match table of mode selection signal/mode output
<table>
<tr><td>MODEIN (mode selection signal)</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td></tr>
<tr><td>MODE (control mode)</td><td>CASDR</td><td>CSV</td><td>CMV</td><td>CAS</td><td>AUT</td><td>MAN</td></tr>
</table>
CASDR :CASCADE DIRECT
CSV :Computer SV setting
CMV :Computer MV setting
CAS :Cascade
AUT :Automatic
MAN :Manual
(2) The mode selection transition disabled check and mode selection output processing will not be performed if MODEIN is not within the range of 1 to 6.
The tag data control mode (MODE) holds the previous value.
(3) When “Change the control mode to MANUAL” is checked in the <<I/O Control>> tab of the project parameter setting, the control mode changes to MANUAL if a sensor error or output open alarm occurs. However, the control mode does not change when the disable manual bit is ON.
</td></tr>
<tr><td>Mode selection transition disabled check</td><td>If the corresponding bit of control mode inhibition (MDIH) of tag data is TRUE (valid), execute mode selection disabled on it. (The mode selection output processing is not performed.)</td></tr>
<tr><td>Selection request and mode selection output</td><td>
If the selection request (E) is TRUE and the corresponding mode of mode selection signal is changed, the corresponding bit of the control mode (MODE) of tag data is set as TRUE. (For details of the corresponding bit, refer to Appendix 1.2.)
<table>
<tr><td colspan="5">Condition</td><td rowspan="2">Control mode (MODE) of tag data</td></tr>
<tr><td>Mode selection request (E)</td><td>Mode selection signal (MODEIN)</td><td>Mode selection signal range check</td><td>Mode selection transition disabled check</td><td>Mode selection output</td></tr>
<tr><td rowspan="2">FALSE</td><td>1 to 6</td><td>Valid</td><td>Stop</td><td>Stop</td><td>Hold previous value</td></tr>
<tr><td>Beyond 1 to 6</td><td>Invalid</td><td>Stop</td><td>Stop</td><td>Hold previous value</td></tr>
<tr><td rowspan="2">TRUE</td><td>1 to 6</td><td>Valid</td><td>Execute</td><td>Execute</td><td>The corresponding bit will be TRUE.</td></tr>
<tr><td>Beyond 1 to 6</td><td>Invalid</td><td>Stop</td><td>Stop</td><td>Hold previous value</td></tr>
</table>
</td></tr>
</table>

Processing Operation

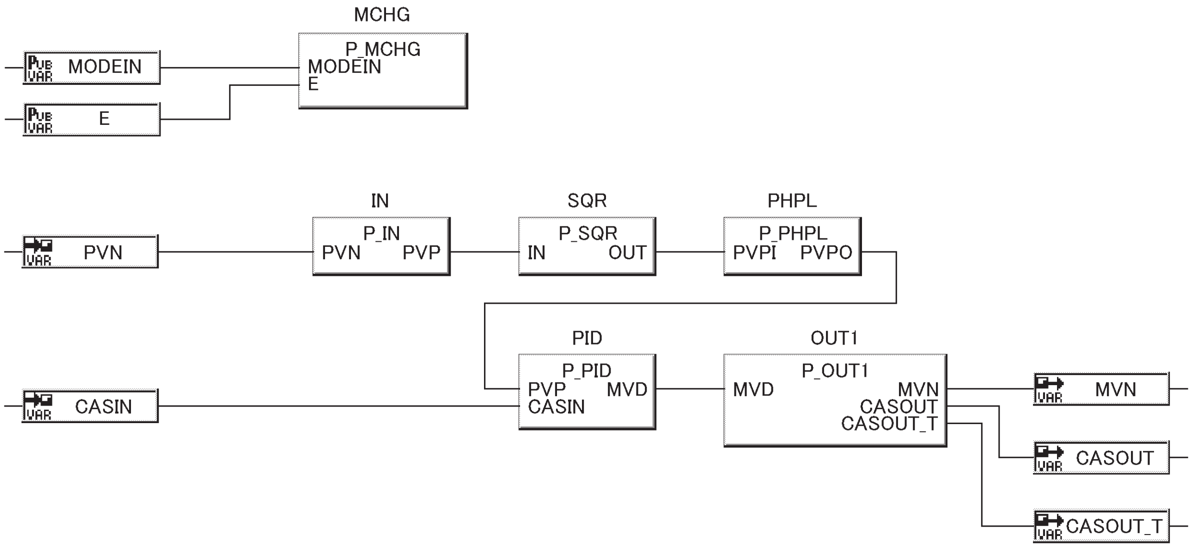
Processing / Control mode	Range check of mode change signal	Status check of "Disable Mode Change" (MDIH)	Change request detection	Mode change output
MAN, CMV, AUT, CAS, CSV, CASDR	○	○	○	○

○: Execute ×: Not execute

Error

There is no error caused by P_MCHG.

Program Example



POINT

It is necessary to refer to public variable on user-defined FB/Tag FB in order to read/write the initial values of public variables of the general process FB which is arranged on user-defined FB/Tag FB with the FB property window or programs.

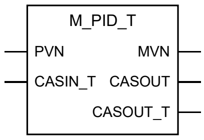
9 PROCESS FB_TAG FB

Tag FB is the instructions used for process control. It can be classified into following types.

Classification Name		Description	Reference
Tag FB	Loop tag FB	Ratio control, PID control, 2 position ON/OFF, 3 position ON/OFF, program setter and loop selector, etc.	Section 9.1
	Status tag FB	Reversible and irreversible operation, ON/OFF operation, timer and counter, etc.	Section 9.2
	Alarm tag FB	Execute alarm notification.	Section 9.3
	Message tag FB	Execute message notification.	Section 9.4

9.1 Tag FB_Loop Tag FB

9.1.1 Velocity Type PID Control (With Tracking to primary loop) (M_PID_T)

FB	FBD parts
M_PID_T	

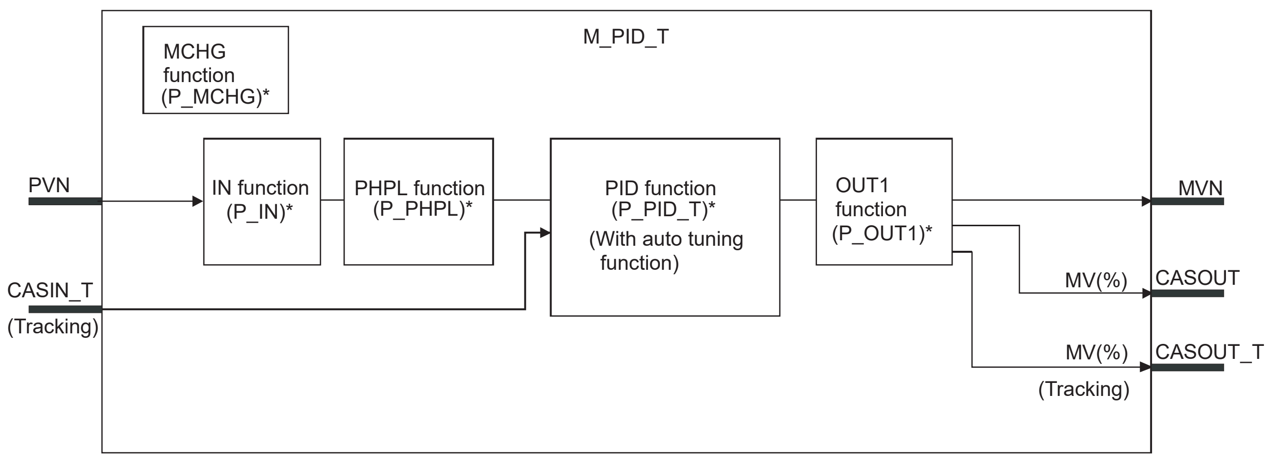
Corresponding tag type
PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute velocity type basic PID control taking function of P_IN+P_PHPL+P_PID_T+P_OUT1 as a single FB.

Function/FB classification name: Tag FB_Loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT1_NMIN to OUT1_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PID_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PID_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PID_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PID_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	PID_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PID_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	OUT1_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT1_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PID function	P_PID_T	Section 8.2.3
OUT1 function	P_OUT1	Section 8.1.2
MCHG function	P_MCHG	Section 8.3.1

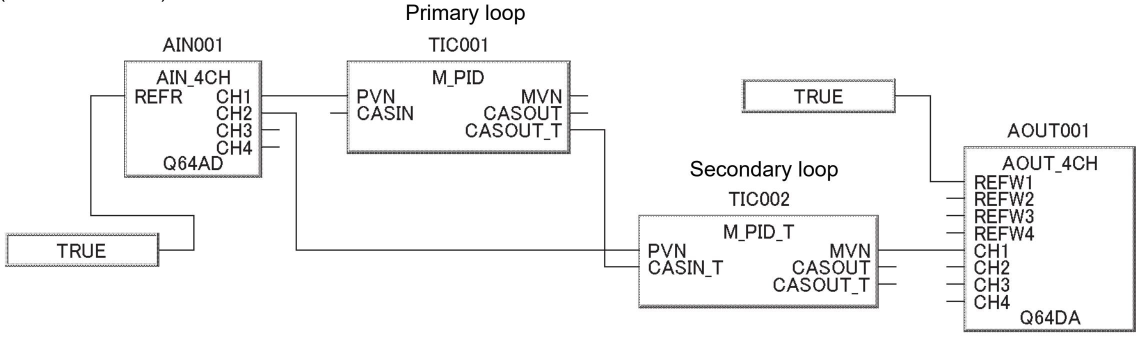
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

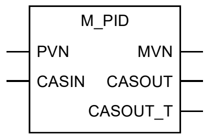
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(Cascade control)



9.1.2 Velocity Type PID Control (Without Tracking to primary loop) (M_PID)

FB	FBD parts
P_PID	

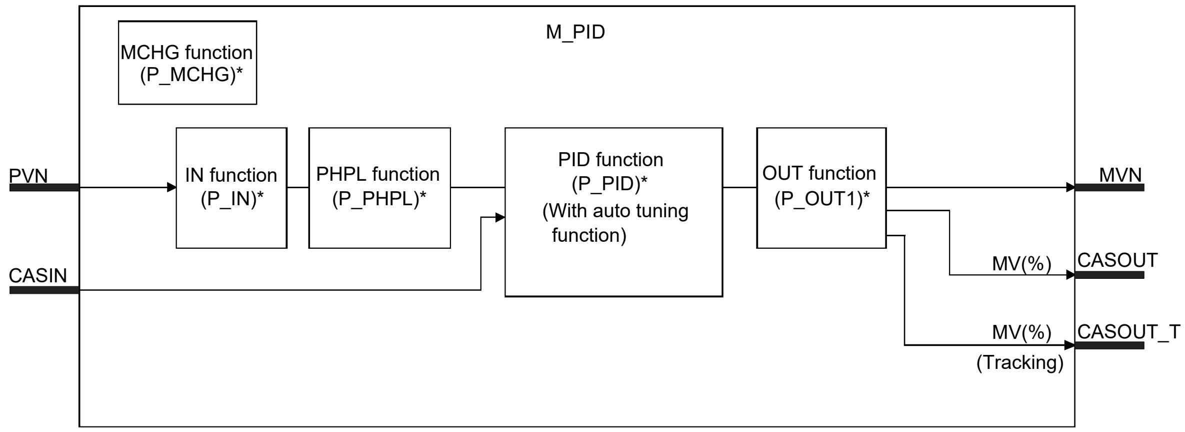
Corresponding tag type
PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute velocity type basic PID control taking function of P_IN+P_PHPL+P_PID+P_OUT1 as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT1_NMIN to OUT1_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PID_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PID_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PID_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PID_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	OUT1_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT1_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

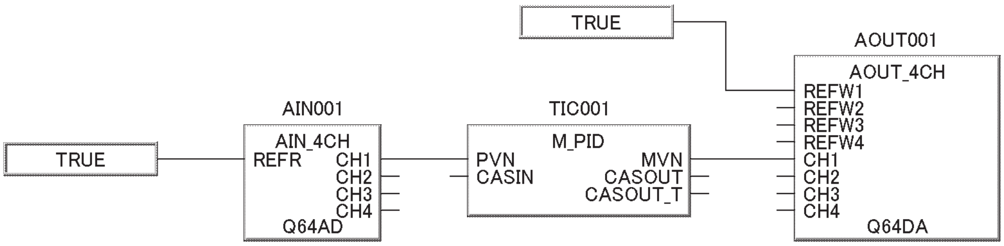
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PID function	P_PID	Section 8.2.4
OUT1 function	P_OUT1	Section 8.1.2
MCHG function	P_MCHG	Section 8.3.1

Error

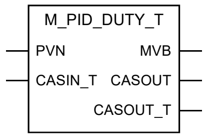
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.3 Velocity Type PID Control and Duty Output (With Tracking to primary loop) (M_PID_DUTY_T)

FB	FBD parts
M_PID_DUTY_T	

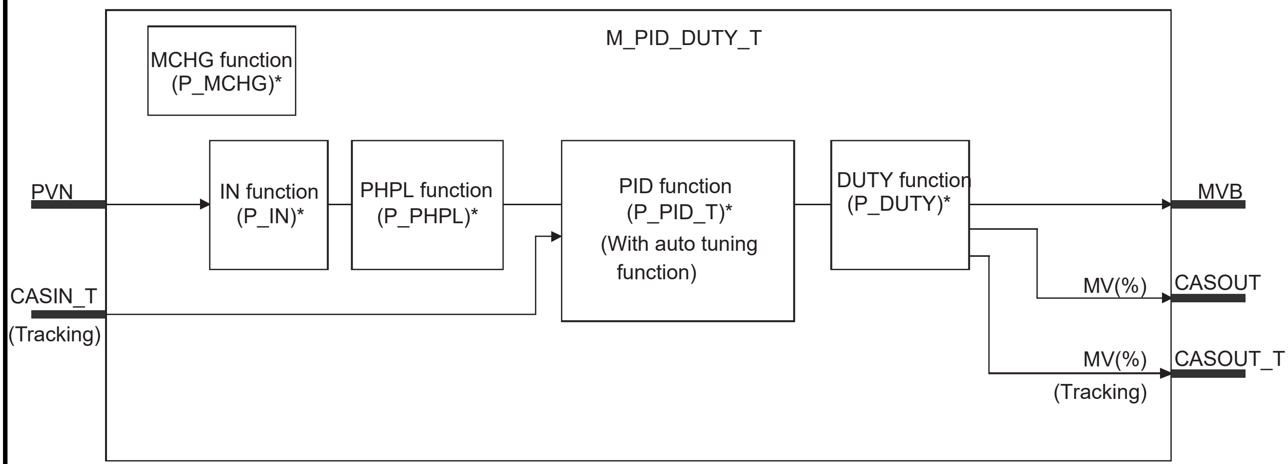
Corresponding tag type
PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute velocity type basic PID control taking function of P_IN+P_PHPL+P_PID_T+P_DUTY as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVB	Output variable	REAL	Bit output to module FB	TRUE, FALSE
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PID_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PID_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PID_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PID_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	PID_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PID_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PID function	P_PID_T	Section 8.2.3
DUTY function	P_DUTY	Section 8.1.6
MCHG function	P_MCHG	Section 8.3.1

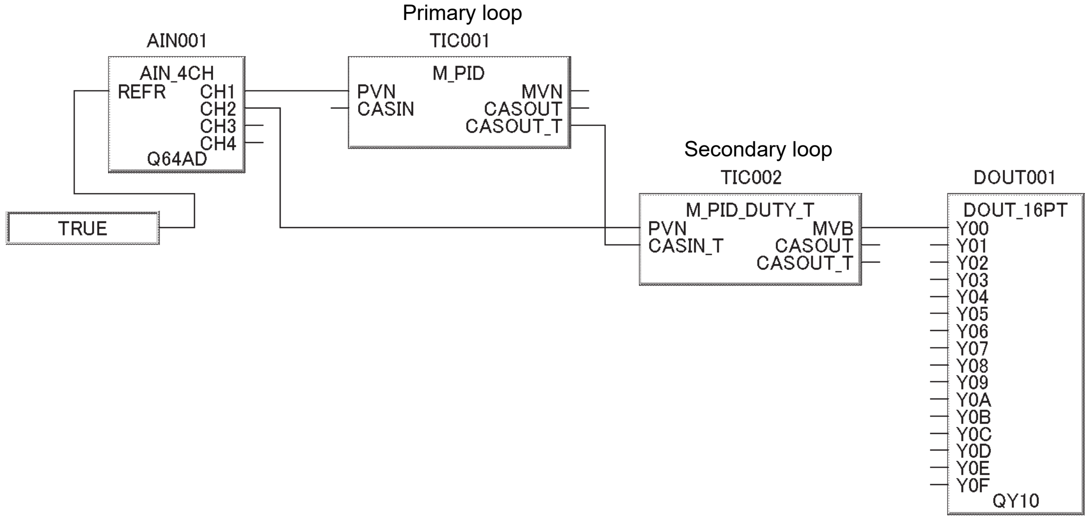
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

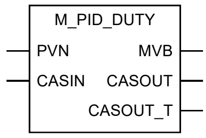
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(Cascade control)



9.1.4 Velocity Type PID Control and Duty Output (Without Tracking to primary loop) (M_PID_DUTY)

FB	FBD parts
M_PID_DUTY	

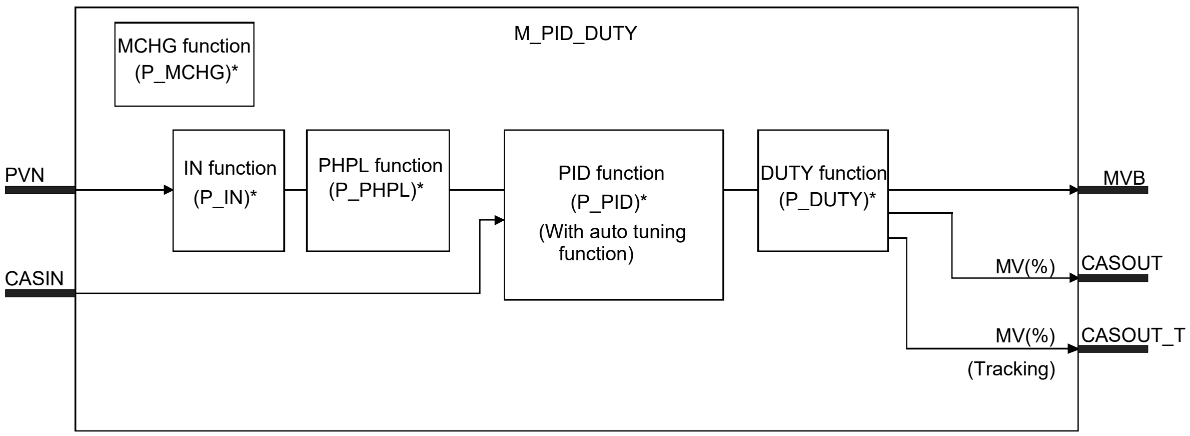
Corresponding tag type
PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute velocity type basic PID control taking function of P_IN+P_PHPL+P_PID +P_DUTY as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVB	Output variable	BOOL	Bit output to module FB	TRUE, FALSE
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PID_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PID_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PID_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PID_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the simulation processing.

*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

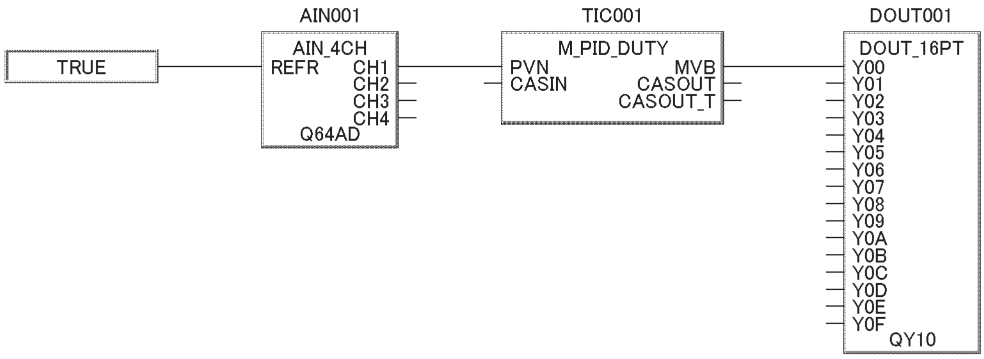
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PID function	P_PID	Section 8.2.4
DUTY function	P_DUTY	Section 8.1.6
MCHG function	P_MCHG	Section 8.3.1

Error

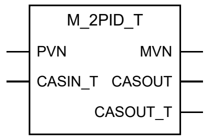
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.5 2-Degree-of-Freedom PID Control (With Tracking to primary loop) (M_2PID_T)

FB	FBD parts
M_2PID_T	

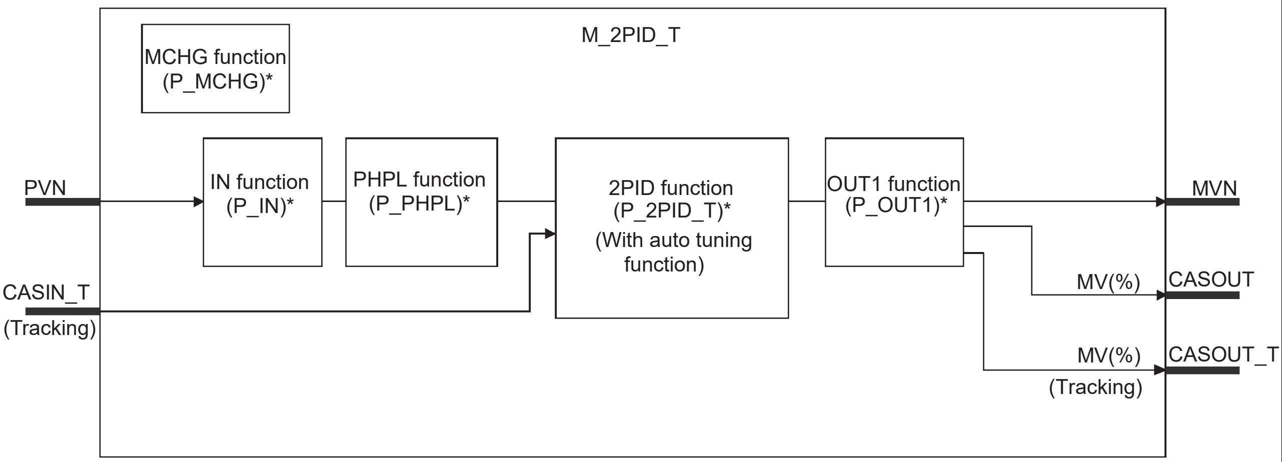
Corresponding tag type
2PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 2-degree-of-freedom PID control taking function of P_IN+P_PHPL+P_2PID_T+P_OUT1 as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT1_NMIN to OUT1_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PID2_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PID2_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PID2_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PID2_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	PID2_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PID2_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	OUT1_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT1_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
2PID function	P_2PID_T	Section 8.2.5
OUT1 function	P_OUT1	Section 8.1.2
MCHG function	P_MCHG	Section 8.3.1

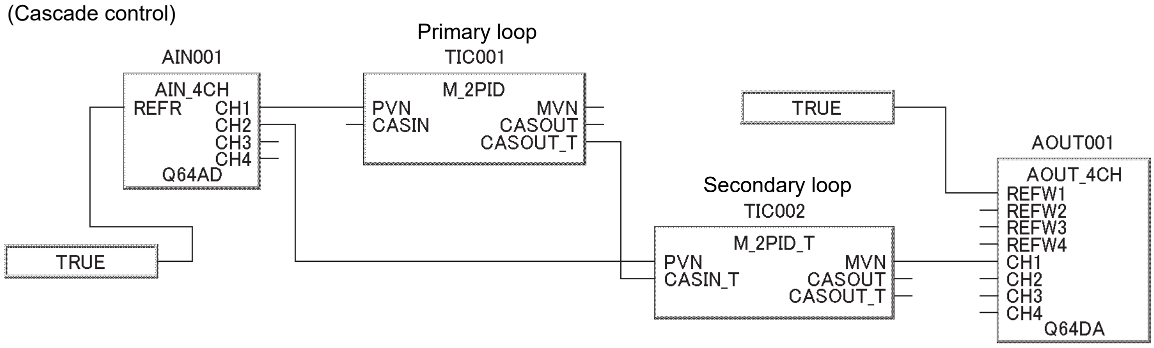
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

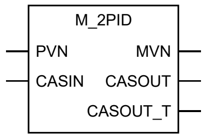
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(Cascade control)



9.1.6 2-Degree-of-Freedom PID Control (Without Tracking to primary loop) (M_2PID)

FB	FBD parts
M_2PID	

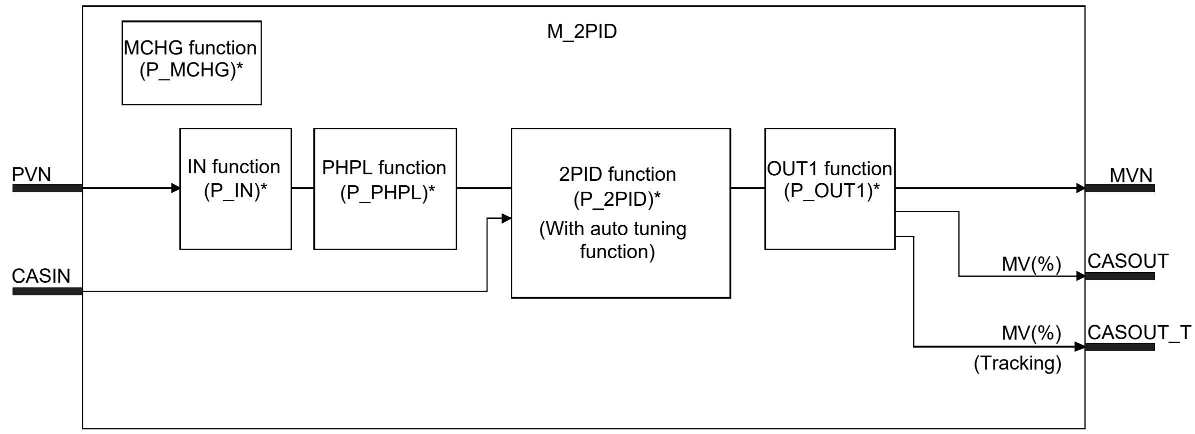
Corresponding tag type
2PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 2-degree-of-freedom PID control taking function of P_IN+P_PHPL+P_2PID+P_OUT1 as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT1_NMIN to OUT1_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PID2_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PID2_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PID2_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PID2_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	OUT1_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT1_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

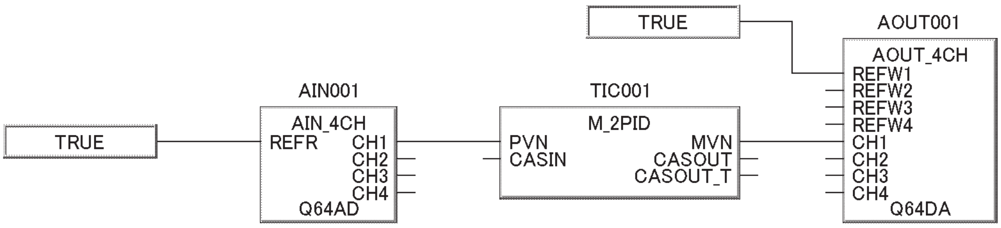
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
2PID function	P_2PID	Section 8.2.6
OUT1 function	P_OUT1	Section 8.1.2
MCHG function	P_MCHG	Section 8.3.1

Error

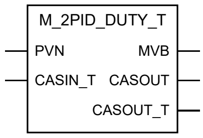
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.7 2-Degree-of-Freedom PID Control and Duty Output (With Tracking to primary loop) (M_2PID_DUTY_T)

FB	FBD parts
M_2PID_DUTY_T	

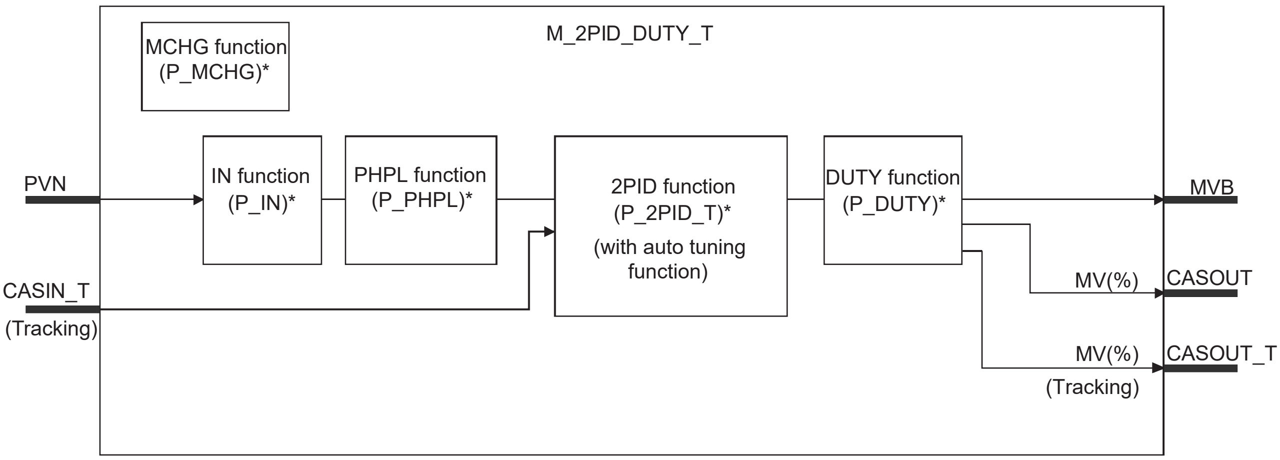
Corresponding tag type
2PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 2-degree-of-freedom PID control and Duty output taking function of P_IN+P_PHPL+P_2PID_T+P_DUTY as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVB	Output variable	BOOL	Bit output to module FB	TRUE, FALSE
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PID2_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PID2_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PID2_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PID2_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	PID2_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PID2_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing *2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
MCHG processing *3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

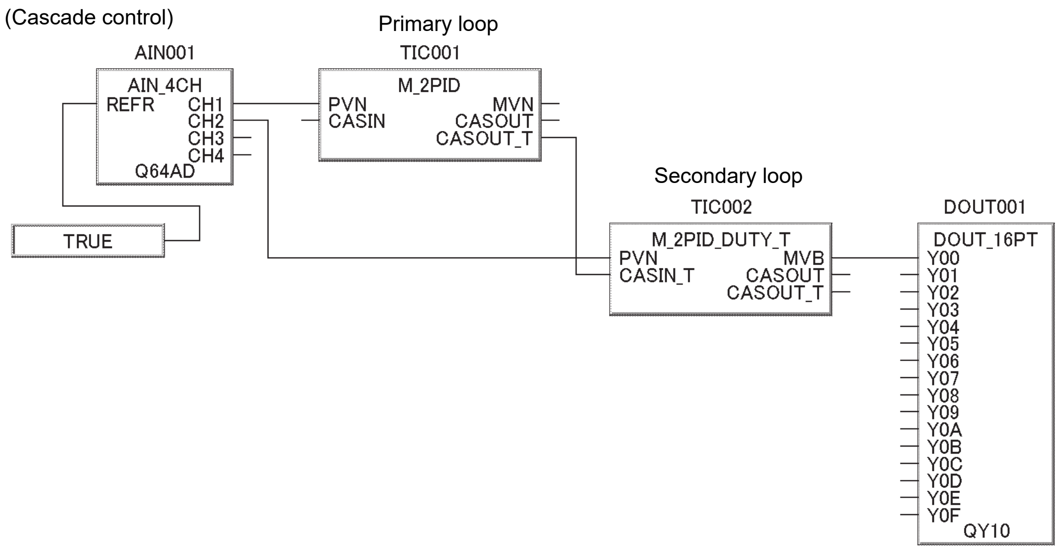
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
2PID function	P_2PID_T	Section 8.2.5
DUTY function	P_DUTY	Section 8.1.6
MCHG function	P_MCHG	Section 8.3.1

Error

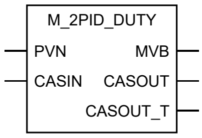
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.8 2-Degree-of-Freedom PID Control and Duty Output (Without Tracking to primary loop) (M_2PID_DUTY)

FB	FBD parts
M_2PID_DUTY	

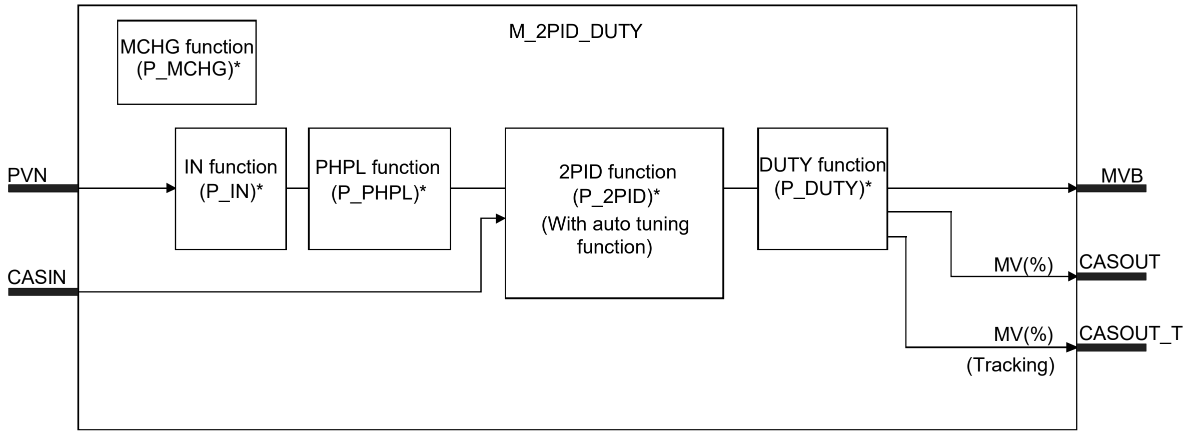
Corresponding tag type
2PID

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 2-degree-of-freedom PID control and Duty output taking function of P_IN+P_PHPL+P_2PID +P_DUTY as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVB	Output variable	BOOL	Bit output to module FB	TRUE, FALSE
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PID2_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PID2_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PID2_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PID2_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

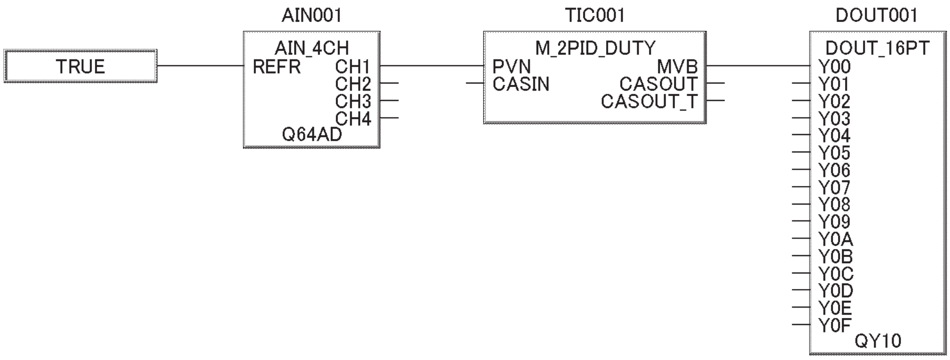
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
2PID function	P_2PID	Section 8.2.6
DUTY function	P_DUTY	Section 8.1.6
MCHG function	P_MCHG	Section 8.3.1

Error

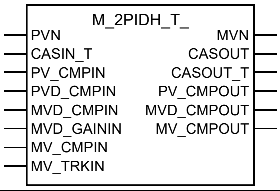
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.9 2-Degree-of-Freedom Advanced PID Control (With Tracking to primary loop) (M_2PIDH_T_)

FB	FBD parts
M_2PIDH_T_	

Corresponding tag type
2PIDH

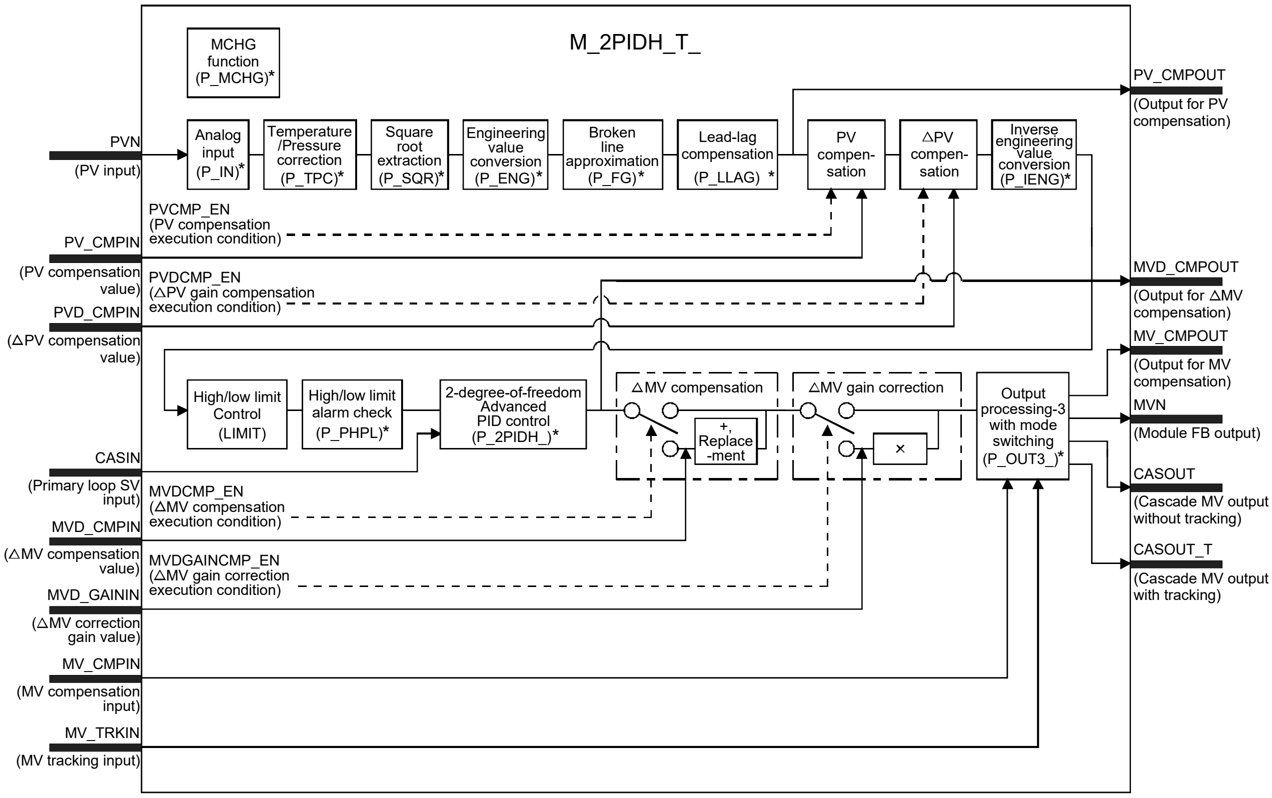
Control mode				
MAN	AUT	CAS*1	CMV	CSV
○	○	○	○	○

*1 Transition to CASDR is possible.

Function overview: Executes 2-degree-of-freedom PID control taking function of P_IN+P_PHPL+P_2PIDH_T_+P_OUT3_ as a single FB and with PV/MV Correction.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_□) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
	PV_CMPIN	Input variable	REAL	PV compensation value	-999999 to 999999
	PVD_CMPIN	Input variable	REAL	△PV compensation value	-999999 to 999999
	MVD_CMPIN	Input variable	REAL	△MV compensation value (Unit: %)	-100 to 100
	MVD_GAININ	Input variable	REAL	△MV correction gain value	-999999 to 999999
	MV_CMPIN	Input variable	REAL	MV compensation value (Unit: %)	-999999 to 999999
	MV_TRKIN	Input variable	REAL	MV tracking input (Unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT3_NMIN to OUT3_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100
	PV_CMPOUT	Output variable	REAL	Output for PV compensation	-999999 to 999999
	MVD_CMPOUT	Output variable	REAL	Output for △MV compensation (Unit: %)	-100 to 100
	MV_CMPOUT	Output variable	REAL	Output for MV compensation (Unit: %)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	TPC_SQR	Public variable	INT	Temperature/pressure correction pattern 0: None 1: Square root extraction 2: Temperature correction+ Square root extraction 3: Pressure correction + Square root extraction 4: Temperature/pressure correction + Square root extraction	0 to 4	0	User
	TPC_PVTEMP	Public variable	REAL	Temperature/pressure correction: Measured temperature (engineering value)	-999999 to 999999	0.0	User
	TPC_PVPRES	Public variable	REAL	Temperature/pressure correction: Measured pressure (engineering value)	-999999 to 999999	0.0	User
	TPC_TEMP	Public variable	REAL	Temperature/pressure correction: Design temperature	-999999 to 999999	0.0	User
	TPC_B1	Public variable	REAL	Temperature/pressure correction: Bias temperature	-999999 to 999999	273.15	User
	TPC_PRES	Public variable	REAL	Temperature/pressure correction: Design pressure	-999999 to 999999	0.0	User
	TPC_B2	Public variable	REAL	Temperature/pressure correction: Bias pressure	-999999 to 999999	10332.0	User
	SQR_OLC	Public variable	REAL	Square root extraction: Output low cut-off value	0 to 999999	0.0	User
	SQR_K	Public variable	REAL	Square root extraction: Coefficient	0 to 999999	10.0	User
	SQR_DENSITY	Public variable	REAL	Square root extraction: Density correction value	0 to 999999	1.0	User
	FG_SN	Public variable	INT	Function generator: Number of points	0 to 48	0	User
	FG_X1 to FG_X48	Public variable	REAL	Function generator: Input coordinates (X-coordinates)	-999999 to 999999	0.0	User
	FG_Y1 to FG_Y48	Public variable	REAL	Function generator: Output coordinates (Y-coordinates)	-999999 to 999999	0.0	User

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing (continued)	LLAG_EN	Public variable	BOOL	First order lag: Execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	LLAG_T1	Public variable	REAL	First order lag: Lag time (second)	0 to 999999	1.0	User
	PVCMP_EN	Public variable	BOOL	PV compensation execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PVCMP_MODE	Public variable	INT	PV compensation mode (0: Addition, 1: Replacement)	0 to 1	0	User
	PVDCMP_EN	Public variable	BOOL	ΔPV compensation execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PID2H_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PID2H_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PID2H_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PID2H _TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	PID2H_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Use)	TRUE, FALSE	TRUE	User
	PID2H_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TRUE	User
	PID2H_PVTRK_EN	Public variable	BOOL	PV tracking execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PID2H_ISTP	Public variable	BOOL	Integration stop signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PID2H_DSTP	Public variable	BOOL	Derivation stop signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PID2H_LMT_ISTP	Public variable	BOOL	Stop Integration, when MV variation rate limiter alarm occurred (TRUE: Stop, FALSE: Not stop)	TRUE, FALSE	FALSE	User
	PID2H_SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User
	MVDCMP_EN	Public variable	BOOL	ΔMV compensation execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	MVDCMP_MODE	Public variable	INT	ΔMV compensation mode (0: Addition, 1: Replacement)	0 to 1	0	User
	MVDGAINCMP_EN	Public variable	BOOL	ΔMV gain correction execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT3_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	OUT3_MVCMP_EN	Public variable	BOOL	MV compensation execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_MVCMP_MODE	Public variable	INT	MV compensation mode (0: Addition, 1: Replacement)	0 to 1	0	User
	OUT3_PREMV_EN	Public variable	BOOL	Preset MV execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_PREMV_V	Public variable	REAL	Preset MV value (Unit: %)	0 to 100	0.0	User
	OUT3_MVHLD_EN	Public variable	BOOL	MV hold execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_MVTRK_EN	Public variable	BOOL	MV tracking execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_STP_OTYPE	Public variable	INT	Output selection when loop stop/tag stop is executed (0: Hold, 1: Preset value)	0 to 1	0	User
	OUT3_SEA_OTYPE	Public variable	INT	MV output selection when SEA occurred (0: Hold, 1: Preset MV output, 2: Do not hold and output Preset MV)	0 to 2	0	User
	OUT3_ARW_EX_EN	Public variable	BOOL	Pull MV internal operation value back, when it exceeds MV internal operation high/low limit value (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User
	OUT3_MVPH	Public variable	REAL	MV internal operation high limit value (Unit: %)	MH to 999999	100.0	User
	OUT3_MVPL	Public variable	REAL	MV internal operation low limit value (Unit: %)	-999999 to ML	0.0	User
	OUT3_MVREV_EN	Public variable	BOOL	MV reverse execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing (continued)	OUT3_FOTS_EN	Public variable	BOOL	Tight shut/full open execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_MVFO	Public variable	REAL	Output value for full open (Unit: %)	100 to 125	112.5	User
	OUT3_MVTS	Public variable	REAL	Output value for tight shut (Unit: %)	-25 to 0	-16.82	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2:AUT, 3:CAS, 4:CMV, 5:CSV, 6: CASDR)	1 to 6	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB and general process FB.

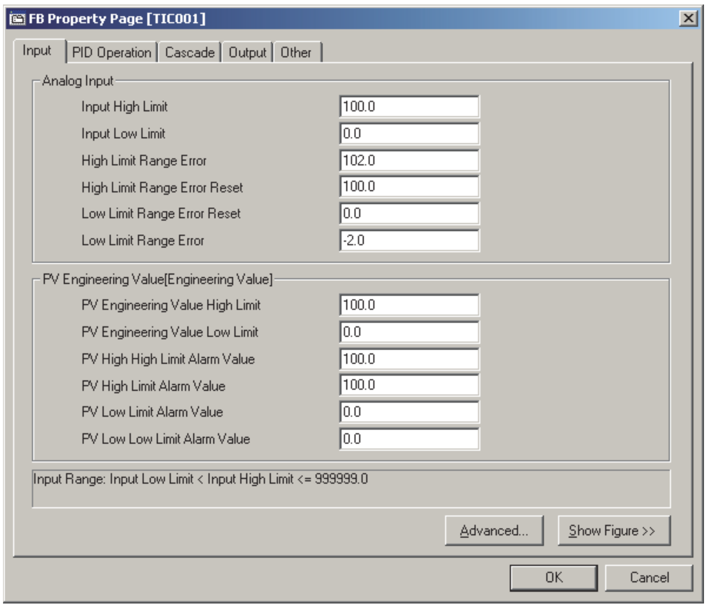
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
TPC function	P_TPC	Section 7.1.6
SQR function	P_SQR	Section 7.2.5
ENG function	P_ENG	Section 7.1.4
FG function	P_FG	Section 7.1.1
LLAG function	P_LLAG	Section 7.4.1
IENG function	P_IENG	Section 7.1.5
LIMIT function	LIMIT	Section 4.6.3
PHPL function	P_PHPL	Section 8.2.19
2PIDH function	P_2PIDH_T_	Section 8.2.7
OUT3 function	P_OUT3_	Section 8.1.4
MCHG function	P_MCHG	Section 8.3.1

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PV compensation</td><td>The compensation value from the external is added to or replaces PV value.
<table>
<tr><th colspan="2">Condition</th><th>Processing result</th></tr>
<tr><td rowspan="2">PVCMP_EN = TRUE</td><td>PVCMP_MODE = 0 (Addition)</td><td>IN + PV_CMPIN</td></tr>
<tr><td>PVCMP_MODE = 1 (Replacement)</td><td>PV_CMPIN</td></tr>
<tr><td>PVCMP_EN = FALSE</td><td>—</td><td>IN</td></tr>
</table>
IN: Input value (PV value), PV_CMPIN: Compensation value, PVCMP_MODE: Compensation mode</td></tr>
<tr><td>△PV compensation</td><td>Add △PV compensation value (PVD_CMPIN) to internal addition value (Σ PVD_CMPIN) when PVDCMP_EN is valid.
Add Σ PVD_CMPIN to PV value.</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>△MV compensation</td><td>The compensation value from the external is added to or replaces △MV.
<table>
<tr><th colspan="2">Condition</th><th>Processing result</th></tr>
<tr><td rowspan="2">MVDCMP_EN = TRUE</td><td>MVDCMP_MODE = 0 (Addition)</td><td>IN + MVD_CMPIN</td></tr>
<tr><td>MVDCMP_MODE = 1 (Replacement)</td><td>MVD_CMPIN</td></tr>
<tr><td>MVDCMP_EN = FALSE</td><td>—</td><td>IN</td></tr>
</table>
IN: Input value (△MV value), MVD_CMPIN: Compensation value, MVDCMP_MODE: Compensation mode</td></tr>
<tr><td>△MV gain correction</td><td>Multiply △MV by gain correction value.
<table>
<tr><th>Condition</th><th>Processing result</th></tr>
<tr><td>MVDGAINCMP_EN = TRUE</td><td>IN ×MVD_GAININ</td></tr>
<tr><td>MVDGAINCMP_EN = FALSE</td><td>IN</td></tr>
</table>
IN: Input value (△MV value), MVD_GAININ: Gain correction value</td></tr>
</table>

Initial setting in FB property page

Initial setting concerning 2-degree-of-freedom Advanced PID control FB (M_2PIDH (_T) _) can be displayed by function on the FB property page of PX Developer programming tool so that the settings are easy. The following explains about the initial value set in public variable and tag data according to the classification on the FB property page.



<FB property page for the setting of 2-degree-of-freedom Advanced PID control FB>

The following shows the function classification in the FB property page.

Tab name	Advanced setting window tab name/Other setting window name	Reference
Input	PV engineering value, Temperature/Pressure correction, Function generator, First order lag, PV Compensation	(1) in this section
PID operation	2-Degree-of-Freedom PID Operation, SV Setting	(2) in this section
Cascade	—	(3) in this section
Output	MV Output, MV Output Selection, MV Compensation	(4) in this section
Other	Mode Disablement, Alarm disregard, Alarm Level, Monitor Tool Display	(5) in this section

(1) Inputs

(a) Basic operations

Set basic items regarding inputs such as the range of A/D conversion value input from an analog input module, PV engineering value scale. The setting details are shown below.

Function	Group	Item	Variable name	Contents
Input	Analog input	Input high limit	IN_NMAX	Set high limit value for the range of A/D conversion values (such as 0 to 4000, 0 to 8000) input from an analog input module. (Example) When using the range of 0 to 64000 for Q64AD-GH → Set "64000". After range error check, limiter processing (high limit) is performed with the input high limit value.
		Input low limit	IN_NMIN	Set low limit value for the range of A/D conversion values (such as 0 to 4000, 0 to 8000) input from an analog input module. (Example) When using the range of 0 to 64000 for Q64AD-GH → Set "0". After range error check, limiter processing (low limit) is performed with the input low limit value.
		High limit range error	IN_HH	Set reference value of high limit exceeding error (range high limit error) for A/D conversion values input from an analog input module. When AD conversion value is greater than this value, high limit range error (Sensor alarm SEA) occurs.
		High limit range error reset	IN_H	Set reference value of error reset performed after high limit range error occurrence. When A/D conversion value is smaller than this value, the high limit range error (Sensor alarm SEA) is reset.
		Low limit range error reset	IN_L	Set reference value of error reset performed after low limit range error occurrence. When A/D conversion value is greater than this value, the low limit range error (Sensor alarm SEA) is reset.
		Low limit range error	IN_LL	Set reference value of low limit exceeding error (range low limit error) for A/D conversion values input from an analog input module. When A/D conversion value is smaller than this value, low limit range error (Sensor alarm SEA) occurs.
	PV engineering value	PV engineering value high limit	RH	Set high limit value for using A/D conversion value inputs from an analog input module as PV engineering values. (Example) When using PV engineering value of 0 to 200℃ → Set "200". The PV engineering value high limit corresponds to the input high limit of analog inputs.
		PV engineering value low limit	RL	Set low limit value for using A/D conversion value inputs from an analog input module as PV engineering values. (Example) When using PV engineering value of 0 to 200℃ → Set "0". The PV engineering value low limit corresponds to the input low limit of analog inputs.
		PV high high limit alarm value	HH	Set reference value of high high limit exceeding alarm for PV engineering value. When PV engineering value is greater than this value, the input high high limit alarm (HHA) occurs.
		PV high limit alarm value	PH	Set reference value of high limit exceeding alarm for PV engineering value. When PV engineering value is greater than this value, the input high limit alarm (PHA) occurs.
		PV low limit alarm value	PL	Set reference value of low limit exceeding alarm for PV engineering value. When PV engineering value is smaller than this value, the input low limit alarm (PLA) occurs.
		PV low low limit alarm value	LL	Set reference value of low low limit exceeding alarm for PV engineering value. When PV engineering value is smaller than this value, the input low low limit alarm (LLA) occurs.

(b) PV engineering value

Set items as filter coefficient, high/low limit alarm hysteresis, and variation rate check for PV engineering values. The setting details are shown below.

Function	Group	Item	Variable name	Contents
PV engineering value	—	PV filter coefficient	ALPHA	Set filter coefficient for digital filtering processing to be performed against input values. Digital filtering processing is a simpler processing compared to the first order lag filtering. When the first order lag filtering is required, enable the first order lag filter on the "First Order Lag setting screen" and set lag time.
		PV high/low limit alarm hysteresis	HS	Set hysteresis width for alarm restoration for the case input high limit, high high limit, low limit or low low limit exceeding alarm occurs. Set it with a percentage value (0 to 100%) of the range from the PV engineering value low limit to the PV engineering value high limit. In the case the input high limit alarm has occurred, for example, the input high limit alarm is restored when PV engineering value becomes smaller than a value obtained by subtracting the hysteresis width from the PV high limit alarm value.
	Variation rate check	Variation rate alarm check time	CTIM	This is used for checking the variation rate or PV value. The variation rate alarm is checked within this period. Specify a period (seconds) with a multiplied (by an integral number) value of the execution cycle ΔT (Execution cycle in unit of FBD program). Set the CTIM and ΔT so that CTIM / ΔT is greater than or equal to 2.
		Variation rate alarm value	DPL	Set change range for checking the variation rate of PV value. Set it with a percentage value (0 to 100%) of the range from the PV engineering value low limit to the PV engineering value high limit. The PV value will not be restricted even when the variation rate alarm occurs.

(c) Temperature/pressure correction

Temperature/pressure correction is required when the conditions (temperature, pressure, density) of the fluid, to which the differential pressure is measured with a differential pressure type flow meter, are different from the design conditions. Perform the temperature/pressure correction when measuring gas. Also, perform the square root extraction since the measured differential pressure has the characteristics of the squared flow quantity.
Set items regarding design conditions for performing the temperature/pressure correction. The setting details are shown below.

Function	Group	Item	Variable name	Contents
Temperature/ pressure correction	—	Temperature/ pressure correction pattern	TPC_SQR	Set correction pattern. 0: None, 1: Square root extraction, 2: Temperature correction + Square root extraction, 3: Pressure correction + Square root extraction, 4: Temperature/pressure correction + Square root extraction
	Temperature correction	Design temperature	TPC_TEMP	Set the temperature specified in the design specification. Use the same unit as measured temperature.
		Bias temperature	TPC_B1	Set the bias temperature to perform the correction calculation with absolute temperature. Set "273.15" when Celsius is used for the design temperature and measured temperature.
	Pressure correction	Design pressure	TPC_PRES	Set the pressure specified in the design specification. Use the same unit as measured pressure.
		Bias pressure	TPC_B2	Set the bias pressure to perform the correction calculation with absolute pressure. Measured variables of equipment are input in gauge pressure (The atmosphere pressure is 0.), normally. Set "101.3" when kilo Pascal (kPa) is used for the design pressure and measured pressure.
	Square root	Coefficient	SQR_K	For process FB, the input value internal operation is performed in percentage (%). Set "10.0".
		Output low cut-off value	SQR_OLC	Output is cut off when the value becomes unstable due to small input value. When the input value is 1% (When coefficient is "10.0"), the output low cut-off value is "10.0".

When applying density correction, substitute the density correction value to the public variable "SQR_DENSITY" in the program. When not applying density correction, set the value to "1.0" on the property window.

Example) In the case of gas flow quantity
Density correction value = Design density / Measured density

The calculation formula to be used when applying all corrections in flow quantity measurement is as follows.

$$\text{Coefficient} \times \sqrt{\text{Differential pressure input} \times \frac{\text{Design temperature + Bias temperature}}{\text{Measured temperature + Bias temperature}} \times \frac{\text{Measured pressure + Bias pressure}}{\text{Design pressure + Bias pressure}} \times \text{Density correction value}}$$

(d) Function generator

Approximate and correct by broken line correction processing when the input value and the actual PV engineering value are not in direct proportion to each other. Also set the items regarding number of points and coordinates of broken line correction processing. The setting details are shown below.

Function	Group	Item	Variable name	Contents
Function generator	—	Number of points	FG_SN	Set the number of points used in the broken line correction processing. The correction is not executed when the number of points is 0.
		Input coordinates (X-coordinates)	FG_X1 to FG_X48	Set the input coordinates (X-coordinates) of the broken line correction processing in engineering value.
		Output coordinates (Y-coordinates)	FG_Y1 to FG_Y48	Set the output coordinates (Y-coordinates) of the broken line correction processing in engineering value.

(e) First order lag

Use the first order lag filter to suppress the sudden change and the noise of input value so that the PV engineering value is stable. And set the items regarding the lag time of the first order lag filter (lag time constant). The setting details are shown below.

Function	Group	Item	Variable name	Contents
First order lag	First order lag	Enable first order lag filter	LLAG_EN	Set the Enable/Disable setting of first order lag filter function.
		Lag time (second)	LLAG_T1	Set the lag time (second) of the first order lag filter (lag time constant).

(f) PV compensation

The compensation value from the external (Example: Smith's dead time compensation method) is added to or replaces PV engineering value. Also the compensation value to be added is input as the velocity type in △PV compensation. Set the items regarding Enable/Disable of PV compensation and △PV compensation. The setting details are shown below.

Function	Group	Item	Variable name	Contents
PV compensation	PV compensation	Enable PV compensation	PVCMP_EN	Set the Enable/Disable setting of PV compensation function.
		PV compensation mode	PVCMP_MODE	Set the mode to execute PV compensation. Select either the addition or the replacement.
	—	Enable △PV compensation	PVDCMP_EN	Set the Enable/Disable setting of △PV compensation function. If △PV compensation is enabled, internally estimate the compensation value input in velocity type. Then, add the integration value to PV engineering value. Therefore, even if compensation value becomes 0 due to the effect of break, the sudden change of PV engineering value can be avoided. When △PV compensation is set Disable, the internal integration value of the compensation value is reset (set to 0).

(2) PID operation

(a) Basic operations

Set the items regarding 2-degree-of-freedom PID operation. The setting details are shown below.

Function	Group	Item	Variable name	Contents
PID operation	2-degree-of-freedom PID operation	Reverse action/ direct action	PID2H_PN	Set PID operation pattern. Reverse action increases the manipulated variable (MV) when the process variable (PV) decreases more than the setting value (SV). Direct action increases the manipulated variable (MV) when the process variable (PV) increases more than the setting value (SV).
		Control cycle	CT	Indicate PID operation cycle and set the time (second) that is the integral number multiple of execution cycle ΔT (The default is 200ms in the execution cycle of FBD program).
	PID constant	Proportional gain	P	Set the proportional gain in P operation. Set in not proportional band but proportional gain. Proportional gain equals 100/proportional band (%). When it is 0, proportioning, integral and derivative controls are not executed.
		Integral time	I	Set the integral time in I operation. Integral control is not executed if the integral time is 0.
		Derivative time	D	Set the derivative time in D operation. Derivative control is not executed if the derivative time is 0.
	SV high/low limit	SV high limit value	SH	Set the high limit value of high/low limiter processing to SV value (target). SV high/low limiter processing is executed when "Enable SV high/low limiter" in the SV setting screen is selected.
		SV low limit value	SL	Set the low limit value of high/low limiter processing to SV value (target). SV high/low limiter processing is executed when "Enable SV high/low limiter" in the SV setting screen is selected.

(b) 2-degree-of-freedom PID operation

2-degree-of-freedom PID operation is a method for optimizing both disturbance response and target tracking using the 2-degree-of-freedom PID parameter α and β.
PID control with gap is a method for reducing deviation used in PID operation by increasing the gap width.
The deviation between PV engineering value and SV value (current) is examined in deviation check and raise a large deviation alarm if the deviation exceeds the limit value.
The items regarding parameters, PID control with gap and large deviation alarm in 2-degree-of-freedom PID operation are set. The setting details are shown below.

Function	Group	Item	Variable name	Contents
2-degree-of-freedom PID operation	—	2-degree-of-freedom parameter Alpha	ALPHA2	Set the value of 2-degree-of-freedom PID parameter α (feed forward proportional). If α is tuned up, the manipulated variable in relation to setting value changing will become smaller, and it will take a time to be stable.
		2-degree-of-freedom parameter Beta	BETA2	Set the value of 2-degree-of-freedom PID parameter β (feed forward derivative). If β is tuned down, the derivative effect in relation to setting value changing will become bigger, and short-time period oscillation will occur, sometimes the system will be unstable.
		Derivative gain	PID2H_MTD	Derivative gain is a constant to determine the characteristics of imperfect derivative. The number is normally needless to change (change only when imperfect derivative characteristics should be adjusted strictly).
		Gap width	GW	Set the gap width (0 to 100%) when executing PID control with gap. PID control with gap will be executed if |Actual deviation| ≦ Gap width.
		Gap gain	GG	Set gap gain when executing PID control with gap. Also set the gain in relation to the actual deviation (0 to 100%) for executing PID control with gap. Actual deviation × Gap gain is the deviation used in PID operation.
		Large deviation alarm hysteresis	PID2H_DVLS	Set hysteresis width for recovering alarm after large deviation alarm (DVLA) occurred. Set it with a percentage value (0 to 100%) that is to the value subtracts PV engineering value low limit from PV engineering value high limit. Large deviation alarm will be recovered when |Actual deviation| ≦ (Deviation limit value - Large deviation alarm hysteresis) is established after large deviation alarm occurred.
		Deviation limit value	DVL	Set the allowable variation range of deviation in deviation check. Set the variation range with a percentage value (0 to 100%) that is to the value subtracts PV engineering value low limit from PV engineering value high limit. Although large deviation alarm (DVLA) occurs when |Deviation| > Deviation limit value is established, deviation value limit will not be executed.

(c) SV setting

Set the items to SV value (target) such as initial value, variation rate high limit, high/low limiter Enable/Disable and PV tracking Enable/Disable. The setting details are shown below.

Function	Group	Item	Variable name	Contents
SV setting	—	Initial SV value	SV	Set the initial value of SV value (target).
		SV variation rate high limit value	DSVL	Set the high limit value of variation rate limiter processing to SV value (target). Set it with a percentage value (0 to 100%) of the range from the PV engineering value low limit to the PV engineering value high limit.
		Enable SV high/low limiter	PID2H_SVLMT_EN	Set Enable/Disable of SV high/low limiter processing. If the processing is enabled, the SV value (current) is limited within the range between the SV high limit value and the SV low limit value.
		Enable PV tracking	PID2H_PVTRK_EN	Set Enable/Disable of PV tracking processing. PV tracking is the function that matches SV value (target) and PV value when the control mode is either manual or computer MV to avoid the sudden change of MV value in mode switching (Manual → Auto).

(3) Cascade

Set the items regarding cascade connection. The setting details are shown below.

Function	Group	Item	Variable	Contents
Cascade	Cascade connection	Do not use/Use as the secondary loop	PID2H_SVPTN_B0 PID2H_SVPTN_B1	Set whether or not to use as the secondary loop.
		Enable to execute tracking*1	PID2H_TRK	Set whether or not to execute tracking (transfer) of the SV value in the secondary loop to the MV value of primary loop if the control mode is other than cascade and cascade direct mode. This setting avoids the sudden change of the SV value of secondary loop in switching the control mode to cascade.

*1 Settable only when the tag type is M_2PIDH_T_.

(4) Output

(a) Basic operations

Set the range of D/A conversion value to be written to an analog output module as the basic items regarding outputs. The setting details are shown below.

Function	Group	Item	Variable name	Contents
Output	Analog output	Output conversion high limit	OUT3_NMAX	Set the high limit value for the range of D/A conversion values (such as 0 to 4000, 0 to 8000) for writing to an analog output module. (Example) When using the range of 0 to 12000 for Q64DA → Set "12000".
		Output conversion low limit	OUT3_NMIN	Set the low limit value for the range of D/A conversion values (such as 0 to 4000, 0 to 8000) for writing to an analog output module. (Example) When using the range of 0 to 12000 for Q64DA → Set "0".
	Tight shut/full open	Enable tight shut/full open	OUT3_FOTS_EN	Set whether or not to enable tight shut/full open.
		Full open output value	OUT3_MVFO	Set the output value for full open.
		Tight shut output value	OUT3_MVTS	Set the output value for tight shut.

(b) MV output

Set the items regarding MV high/low limit, MV value instantaneous pullback and MV variation rate. The setting details are shown below.

Function	Group	Item	Variable name	Contents
MV output	—	Initial MV value	MV	Set the initial MV value.
	MV high/low limit	MV high limit value	MH	Set the high limit value for MV high/low limiter processing. When MV value after output variation rate limit > MV high limit value is established, output high limit alarm (MHA) occurs and the MV value is limited by the MV high limit value (output high limiter). Output high limit alarm (MHA) recovers when MV value ≦ MV high limit value.
		MV low limit value	ML	Set the low limit value for MV high/low limiter processing. When MV value after output variation rate limit < MV low limit value is established, output low limit alarm (MLA) occurs and the MV value is limited by the MV low limit value (output low limiter). Output low limit alarm (MLA) recovers when MV value ≧ MV low limit value.
	MV value instantane-ous pullback	Pull MV internal operation value back, when it exceeds MV internal operation high/low limit value	OUT3_ARW_EX_EN	Set whether or not to enable pull MV internal operation value back, when it exceeds MV internal operation high/low limit value. Use when considerably increasing proportional gain value. Set to disabled in normal control.
		MV internal operation high limit value	OUT3_MVPH	Set the high limit for MV internal operation value. Set the value so as to exceed MV high limit value. If MV internal operation value (MVP) is set enabled for MV value instantaneous pullback, execute limiter processing (high limit).
		MV internal operation low limit value	OUT3_MVPL	Set the low limit for MV internal operation value. Set the value so as not to exceed MV low limit value. If MV internal operation value (MVP) is set enabled for MV value instantaneous pullback, execute limiter processing (low limit).
	MV variation rate	Output variation rate high limit value	DML	Set MV allowable variation range as output variation rate high limit value. Set it with a percentage value (0 to 100%) that is to MV (%). MV variation range is checked in every execution cycle ∆T. When |MV variation range | > Output variation rate high limit value, output variation rate limit alarm (DMLA) occurred and MV variation range is limited by output variation rate high limit value (After ∆T, previous MV value + Output variation rate high limit value = Current MV value is established). This enables to convert MV into ramp status when SV is rapidly changed and not to output rapid variation manipulated variable. Output variation rate limit alarm (DMLA) recovers when |MV variation range | ≦ DML.
		Stop integration, when MV variation rate limiter alarm occurred	PID2H_LMT_ISTP	Set whether or not to stop integration when MV variation rate limiter alarm occurred. Difference exists between MV internal operation value and MV value when the alarm occurs, stop integral operation as countermeasures against reset windup.

(c) MV output selection

Set the items regarding preset MV value, output selection in abnormal occasions and MV reverse output. The setting details are shown below.

Function	Group	Item	Variable name	Contents
MV output selection	—	Preset MV value	OUT3_PREMV_V	When selecting preset MV, set MV value when outputting preset MV in abnormal occasions.
		Output selection when loop stop/tag stop is executed	OUT3_STP_OTYPE	Set the method for MV output in loop stop or tag stop. Select either hold or preset value.
		MV output selection when sensor error occurred	OUT3_SEA_OTYPE	Set the method for MV output in the occurrence of sensor alarm (SEA). Select from "Hold", "Preset MV output", and "Do not hold and output preset MV". When selecting "Do not hold and output preset MV", the result of PID operation + Output addition processing is output.
		Enable MV reverse output	OUT3_MVREV_EN	Set whether or not to output MV reverse. When selecting MV reverse output, output conversion processing is executed by the MV value after inversion processing (100 - MV).

(d) MV compensation

Combine feedforward control when variation of an operation is clear since time lag occurs when responding to disturbance in feedback control.

Set output quantity of feedforward control to the compensation value of △MV compensation or MV compensation. Set the items regarding △MV compensation and MV compensation related to MV compensation. The setting details are shown below.

Function	Group	Item	Variable name	Contents
MV compensation	△MV compensation	Enable △MV compensation	MVDCMP_EN	Set Enable/Disable of △MV compensation. Substitute velocity type compensation value to input variable MVD_CMPIN.
		△MV compensation mode	MVDCMP_MODE	Set the mode for △MV compensation execution. Select either addition or replacement. Addition or replacement operation is executed to △MV that will be input for P_OUT3_.
	—	Enable △MV gain correction	MVDGAINCMP_EN	Set Enable/Disable of gain correction to △MV. Substitute gain correction value to △MV to input variable MVD_GAININ.
	MV compensation	Enable MV compensation	OUT3_MVCMP_EN	Set Enable/Disable of MV compensation. Substitute position type compensation value to input variable MV_CMPIN.
		MV compensation mode	OUT3_MVCMP_MODE	Set the mode for MV compensation execution. Select either addition or replacement. Addition or replacement operation is executed to MV internal operation result of P_OUT3_.

(5) Other

Set the items regarding mode disablement, disable alarm detection, alarm level and monitor tool display setting.

(a) Mode disablement

The setting details are shown below.

Function	Group	Item	Variable name	Contents
Mode disablement	Disable control mode changing	Change to MANUAL mode	MANI	Set the transition to MANUAL mode as "Disable". When disabling transition, transition operation to MANUAL mode in faceplate and mode change FB (P_MCHG type FB) is inhibited.
		Change to AUTO mode	AUTI	Set the transition to AUTO mode as "Disable". When disabling transition, transition operation to AUTO mode in faceplate and mode change FB (P_MCHG type FB) is inhibited.
		Change to CASCADE mode	CASI	Set the transition to CASCADE mode as "Disable". When disabling transition, transition operation to CASCADE mode in faceplate and mode change FB (P_MCHG type FB) is inhibited.
		Change to COMPUTER MV mode	CMVI	Set the transition to COMPUTER MV mode as "Disable". When disabling transition, transition operation to COMPUTER MV mode in faceplate and mode change FB (P_MCHG type FB) is inhibited.
		Change to COMPUTER SV mode	CSVI	Set the transition to COMPUTER SV mode as "Disable". When disabling transition, transition operation to COMPUTER SV mode in faceplate and mode change FB (P_MCHG type FB) is inhibited.
		Change to CASCADE DIRECT mode	CASDRI	Set the transition to CASCADE DIRECT mode as "Disable". When disabling transition, transition operation to CASCADE DIRECT mode in faceplate and mode change FB (P_MCHG type FB) is inhibited.
	Disable I/O mode changing	Change to TAG STOP mode	TSTPI	Set the transition to TAG STOP mode as "Disable". When disabling transition, transition operation to TAG STOP mode in faceplate is inhibited.
		Change to OVERRIDE mode	OVRI	Set the transition to OVERRIDE mode as "Disable". When disabling transition, transition operation to OVERRIDE mode in faceplate is inhibited.
		Change to SIMULATION mode	SIMI	Set the transition to SIMULATION mode as "Disable". When disabling transition, transition operation to SIMULATION mode in faceplate is inhibited.
	—	Change to AUTO TUNING mode	ATI	Set the transition to AUTO TUNING mode as "Disable". When disabling transition, auto tuning operation in tuning mode screen of monitor tool is inhibited.

(b) Alarm Disregard

The setting details are shown below.

Function	Group	Item	Variable name	Contents
Alarm Disregard	—	Disregard all alarms	ERRI	Make disable detection setting of all alarms. When disabling detection, all alarms are not detected without relation to individual setting of disable alarm detection.
	Alarm disregard items	Input high high limit alarm	HHI	Make disable detection setting of input high high limit alarm. When disabling detection, input high high limit alarm (HHA) is not detected.
		Input high limit alarm	PHI	Make disable detection setting of input high limit alarm. When disabling detection, input high limit alarm (PHA) is not detected.
		Input low limit alarm	PLI	Make disable detection setting of input low limit alarm. When disabling detection, input low limit alarm (PLA) is not detected.
		Input low low limit alarm	LLI	Make disable detection setting of input low low limit alarm. When disabling detection, input low low limit alarm (LLA) is not detected.
		Sensor error alarm	SEI	Make disable detection setting of sensor error alarm. When disabling detection, sensor error alarm (SEA) is not detected.
		Positive variation rate alarm	DPPI	Make disable detection setting of positive variation rate alarm. When disabling detection, positive variation rate alarm (DPPA) is not detected.
		Negative variation rate alarm	DPNI	Make disable detection setting of negative variation rate alarm. When disabling detection, negative variation rate alarm (DPNA) is not detected.
		Large deviation alarm	DVLI	Make disable detection setting of large deviation alarm. When disabling detection, large deviation alarm (DVLA) is not detected.
		SV high limit alarm	SVHI	Make disable detection setting of SV high limit alarm. When disabling detection, SV high limit alarm (SVHA) is not detected.
		SV low limit alarm	SVLI	Make disable detection setting of SV low limit alarm. When disabling detection, SV low limit alarm (SVLA) is not detected.
		SV variation rate limit alarm	DSVLI	Make disable detection setting of SV variation rate limit alarm. When disabling detection, SV variation rate limit alarm (DSVLA) is not detected.
		Output high limit alarm	MHI	Make disable detection setting of output high limit alarm. When disabling detection, output high limit alarm (MHA) is not detected.
		Output low limit alarm	MLI	Make disable detection setting of output low limit alarm. When disabling detection, output low limit alarm (MLA) is not detected.
		Output variation rate limit alarm	DMLI	Make disable detection setting of output variation rate limit alarm. When disabling detection, output variation rate limit alarm (DMLA) is not detected.

(c) Alarm level

The setting details are shown below.

Function	Group	Item	Variable name	Contents
Alarm level setting	—	Stop alarm level	SPL	Set alarm level of stop alarm. Select major alarm or minor alarm.
		Input high high limit alarm level	HHL	Set alarm level of input high high limit alarm. Select major alarm or minor alarm.
		Input high limit alarm level	PHL	Set alarm level of input high limit alarm. Select major alarm or minor alarm.
		Input low limit alarm level	PLL	Set alarm level of input low limit alarm. Select major alarm or minor alarm.
		Input low low limit alarm level	LLL	Set alarm level of input low low limit alarm. Select major alarm or minor alarm.
		Sensor error alarm level	SENL	Set alarm level of sensor error alarm. Select major alarm or minor alarm.
		Positive variation rate alarm level	DPPL	Set alarm level of positive variation rate alarm. Select major alarm or minor alarm.
		Negative variation rate alarm level	DPNL	Set alarm level of negative variation rate alarm. Select major alarm or minor alarm.
		Large deviation alarm level	DVLL	Set alarm level of large deviation alarm. Select major alarm or minor alarm.
		SV high limit alarm level	SVHL	Set alarm level of SV high limit alarm. Select major alarm or minor alarm.
		SV low limit alarm level	SVLL	Set alarm level of SV low limit alarm. Select major alarm or minor alarm.
		SV variation rate limit alarm level	DSVLL	Set alarm level of SV variation rate limit alarm. Select major alarm or minor alarm.
		Output high limit alarm level	MHL	Set alarm level of output high limit alarm. Select major alarm or minor alarm.
		Output low limit alarm level	MLL	Set alarm level of output low limit alarm. Select major alarm or minor alarm.
		Output variation rate limit alarm level	DMLL	Set alarm level of output variation rate limit alarm. Select major alarm or minor alarm.

(d) Monitor tool display

The setting details are shown below.

Function	Group	Item	Variable name	Contents
Monitor tool display setting	Unit setting	Index number	UNIT	Set the index Number of Engineering Values displayed in Monitor Tool. Please fill the same number set in the "Unit Setting" window of monitor tool.
	No. of digits after the decimal point setting	No. of digits after the decimal point	N	Set the number of digits after decimal point of Engineering Values displayed on monitor tool.

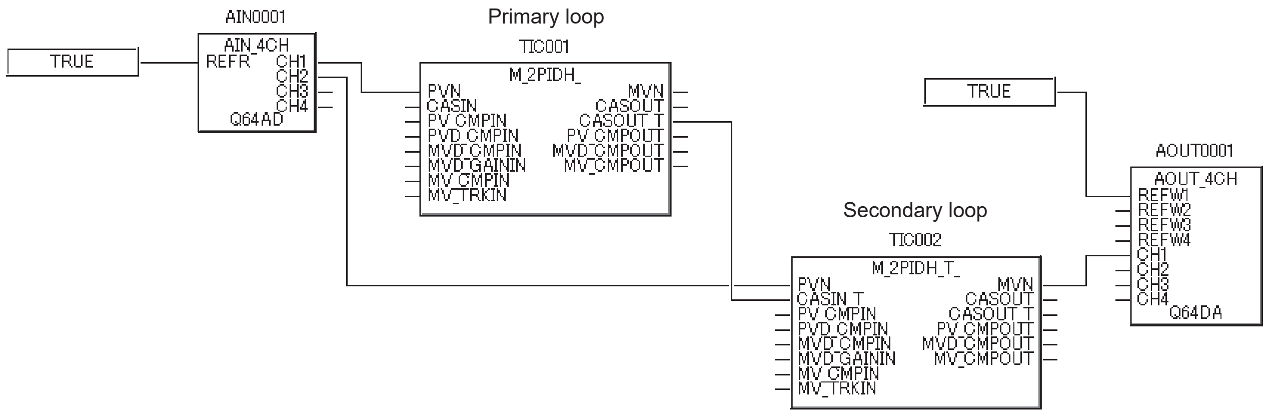
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

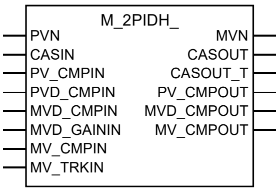
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(Cascade control)



9.1.10 2-Degree-of-Freedom Advanced PID Control (Without Tracking to primary loop) (M_2PIDH_)

FB	FBD parts
M_2PIDH_	

Corresponding tag type
2PIDH

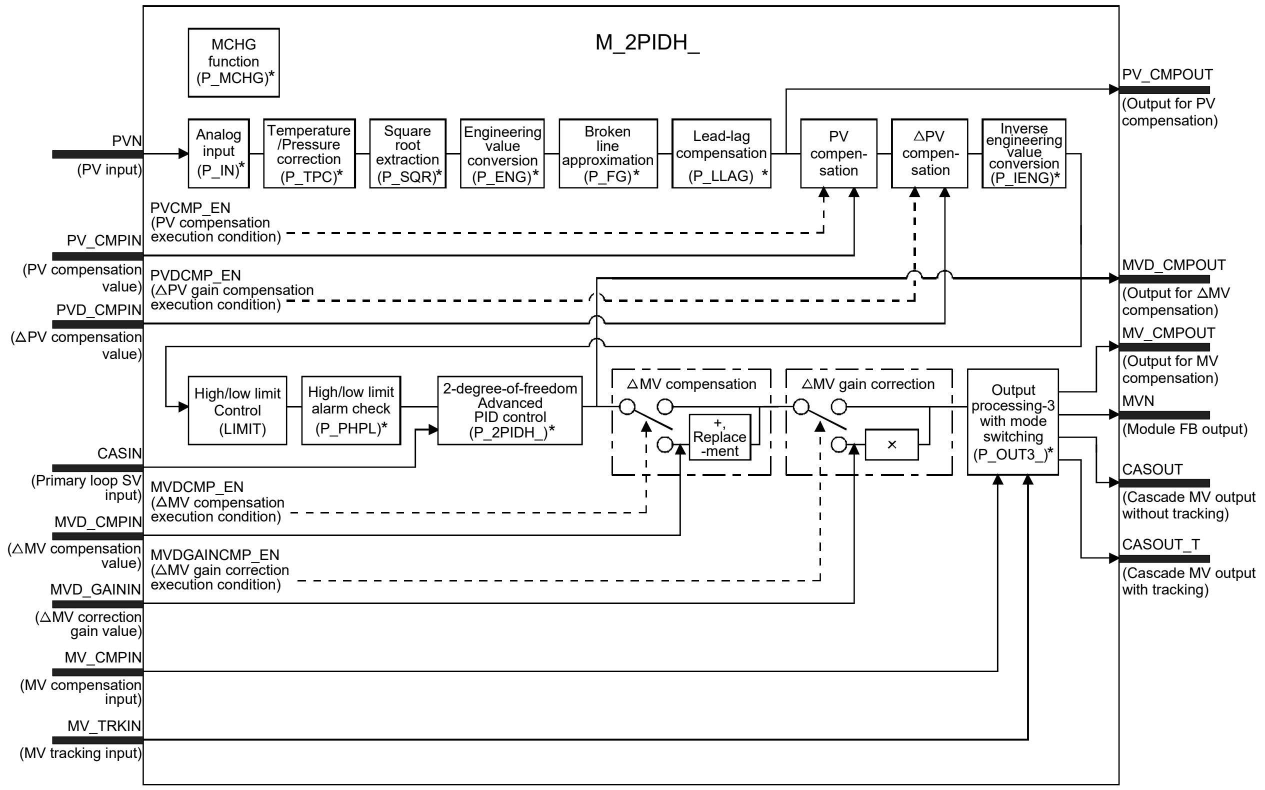
Control mode				
MAN	AUT	CAS*1	CMV	CSV
○	○	○	○	○

*1 Transition to CASDR is possible.

Function overview: Executes 2-degree-of-freedom PID control taking function of P_IN+P_PHPL+P_2PIDH_+P_OUT3_ as a single FB and with PV/MV Correction.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_□) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
	PV_CMPIN	Input variable	REAL	PV compensation value	-999999 to 999999
	PVD_CMPIN	Input variable	REAL	△PV compensation value	-999999 to 999999
	MVD_CMPIN	Input variable	REAL	△MV compensation value (Unit: %)	-100 to 100
	MVD_GAININ	Input variable	REAL	△MV correction gain value	-999999 to 999999
	MV_CMPIN	Input variable	REAL	MV compensation value (Unit: %)	-999999 to 999999
	MV_TRKIN	Input variable	REAL	MV tracking input (Unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT3_NMIN to OUT3_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100
	PV_CMPOUT	Output variable	REAL	Output for PV compensation	-999999 to 999999
	MVD_CMPOUT	Output variable	REAL	Output for △MV compensation (Unit: %)	-100 to 100
	MV_CMPOUT	Output variable	REAL	Output for MV compensation (Unit: %)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	TPC_SQR	Public variable	INT	Temperature/pressure correction pattern 0: None 1: Square root extraction 2: Temperature correction+ Square root extraction 3: Pressure correction + Square root extraction 4: Temperature/pressure correction + Square root extraction	0 to 4	0	User
	TPC_PVTEMP	Public variable	REAL	Temperature/pressure correction: Measured temperature (engineering value)	-999999 to 999999	0.0	User
	TPC_PVPRES	Public variable	REAL	Temperature/pressure correction: Measured pressure (engineering value)	-999999 to 999999	0.0	User
	TPC_TEMP	Public variable	REAL	Temperature/pressure correction: Design temperature	-999999 to 999999	0.0	User
	TPC_B1	Public variable	REAL	Temperature/pressure correction: Bias temperature	-999999 to 999999	273.15	User
	TPC_PRES	Public variable	REAL	Temperature/pressure correction: Design pressure	-999999 to 999999	0.0	User
	TPC_B2	Public variable	REAL	Temperature/pressure correction: Bias pressure	-999999 to 999999	10332.0	User
	SQR_OLC	Public variable	REAL	Square root extraction: Output low cut-off value	0 to 999999	0.0	User
	SQR_K	Public variable	REAL	Square root extraction: Coefficient	0 to 999999	10.0	User
	SQR_DENSITY	Public variable	REAL	Square root extraction: Density correction value	0 to 999999	1.0	User
	FG_SN	Public variable	INT	Function generator: Number of points	0 to 48	0	User
	FG_X1 to FG_X48	Public variable	REAL	Function generator: Input coordinates (X-coordinates)	-999999 to 999999	0.0	User
	FG_Y1 to FG_Y48	Public variable	REAL	Function generator: Output coordinates (Y-coordinates)	-999999 to 999999	0.0	User

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	LLAG_EN	Public variable	BOOL	First order lag: Execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	LLAG_T1	Public variable	REAL	First order lag: Lag time (second)	0 to 999999	1.0	User
	PVCMP_EN	Public variable	BOOL	PV compensation execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PVCMP_MODE	Public variable	INT	PV compensation mode (0: Addition, 1: Replacement)	0 to 1	0	User
	PVDCMP_EN	Public variable	BOOL	△PV compensation execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PID2H_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PID2H_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PID2H_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PID2H_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Use)	TRUE, FALSE	TRUE	User
	PID2H_PVTRK_EN	Public variable	BOOL	PV Tracking execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PID2H_ISTP	Public variable	BOOL	Integration stop signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PID2H_DSTP	Public variable	BOOL	Derivation stop signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	PID2H_LMT_ISTP	Public variable	BOOL	Stop Integration, when MV variation rate limiter alarm occurred (TRUE: Stop, FALSE: Not stop)	TRUE, FALSE	FALSE	User
	PID2H_SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User
	MVDCMP_EN	Public variable	BOOL	△MV compensation execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	MVDCMP_MODE	Public variable	INT	△MV compensation mode (0: Addition, 1: Replacement)	0 to 1	0	User
	MVDGAINCMP_EN	Public variable	BOOL	△MV gain correction execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT3_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	OUT3_MVCMP_EN	Public variable	BOOL	MV compensation execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_MVCMP_MODE	Public variable	INT	MV compensation mode (0: Addition, 1: Replacement)	0 to 1	0	User
	OUT3_PREMV_EN	Public variable	BOOL	Preset MV execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_PREMV_V	Public variable	REAL	Preset MV value (Unit: %)	0 to 100	0.0	User
	OUT3_MVHLD_EN	Public variable	BOOL	MV hold execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_MVTRK_EN	Public variable	BOOL	MV tracking execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_STP_OTYPE	Public variable	INT	Output selection when loop stop/tag stop is executed (0: Hold, 1: Preset value)	0 to 1	0	User
	OUT3_SEA_OTYPE	Public variable	INT	MV output selection when SEA occurred (0: Hold, 1: Preset MV output, 2: Do not hold and output Preset MV)	0 to 2	0	User
	OUT3_ARW_EX_EN	Public variable	BOOL	Pull MV internal operation value back, when it exceeds MV internal operation high/low limit value (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User
	OUT3_MVPH	Public variable	REAL	MV internal operation high limit value (Unit: %)	MH to 999999	100.0	User
	OUT3_MVPL	Public variable	REAL	MV internal operation low limit value (Unit: %)	-999999 to ML	0.0	User
	OUT3_MVREV_EN	Public variable	BOOL	MV reverse execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_FOTS_EN	Public variable	BOOL	Tight shut/full open execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	OUT3_MVFO	Public variable	REAL	Output Value for Full Open (Unit: %)	100 to 125	112.5	User
	OUT3_MVTS	Public variable	REAL	Output Value for Tight Shut (Unit: %)	-25 to 0	-16.82	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2:AUT, 3:CAS, 4:CMV, 5:CSV, 6: CASDR)	1 to 6	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB and general process FB.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
TPC function	P_TPC	Section 7.1.6
SQR function	P_SQR	Section 7.2.5
ENG function	P_ENG	Section 7.1.4
FG function	P_FG	Section 7.1.1
LLAG function	P_LLAG	Section 7.4.1
IENG function	P_IENG	Section 7.1.5
LIMIT function	LIMIT	Section 4.6.3
PHPL function	P_PHPL	Section 8.2.19
2PIDH function	P_2PIDH_T_	Section 8.2.8
OUT3 function	P_OUT3_	Section 8.1.4
MCHG function	P_MCHG	Section 8.3.1

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>PV compensation</td><td>The compensation value from the external is added to or replaces PV value.
<table>
<tr><th colspan="2">Condition</th><th>Processing result</th></tr>
<tr><td rowspan="2">PVCMP_EN = TRUE</td><td>PVCMP_MODE = 0 (Addition)</td><td>IN + PV_CMPIN</td></tr>
<tr><td>PVCMP_MODE = 1 (Replacement)</td><td>PV_CMPIN</td></tr>
<tr><td>PVCMP_EN = FALSE</td><td>—</td><td>IN</td></tr>
</table>
IN: Input value (PV value), PV_CMPIN: Compensation value, PVCMP_MODE: Compensation mode</td></tr>
<tr><td>ΔPV compensation</td><td>Add ΔPV compensation value (PVD_CMPIN) to internal addition value (Σ PVD_CMPIN) when PVDCMP_EN is valid.
Add Σ PVD_CMPIN to PV value.</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>ΔMV compensation</td><td>The compensation value from the external is added to or replaces ΔMV.
<table>
<tr><th colspan="2">Condition</th><th>Processing result</th></tr>
<tr><td rowspan="2">MVDCMP_EN = TRUE</td><td>MVDCMP_MODE = 0 (Addition)</td><td>IN + MVD_CMPIN</td></tr>
<tr><td>MVDCMP_MODE = 1 (Replacement)</td><td>MVD_CMPIN</td></tr>
<tr><td>MVDCMP_EN = FALSE</td><td>—</td><td>IN</td></tr>
</table>
IN: Input value (ΔMV value), MVD_CMPIN: Compensation value, MVDCMP_MODE: Compensation mode</td></tr>
<tr><td>ΔMV gain correction</td><td>Multiply ΔMV by gain correction value.
<table>
<tr><th>Condition</th><th>Processing result</th></tr>
<tr><td>MVDGAINCMP_EN = TRUE</td><td>IN ×MVD_GAININ</td></tr>
<tr><td>MVDGAINCMP_EN = FALSE</td><td>IN</td></tr>
</table>
IN: Input value (ΔMV value), MVD_GAININ: Gain correction value</td></tr>
</table>

Initial setting in FB property page

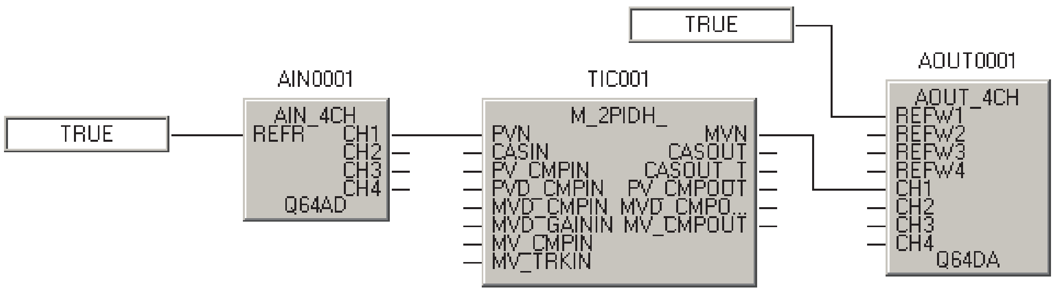
The setting is the same with the 2-degree-of-freedom Advanced PID control (with tracking to primary loop) (M_2PIDH_T_) (☞ Section 9.1.9).

Error

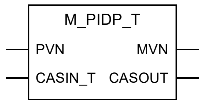
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.11 Position Type PID Control (With Tracking to primary loop, Without Tracking from secondary loop) (M_PIDP_T)

FB	FBD parts
M_PIDP_T	

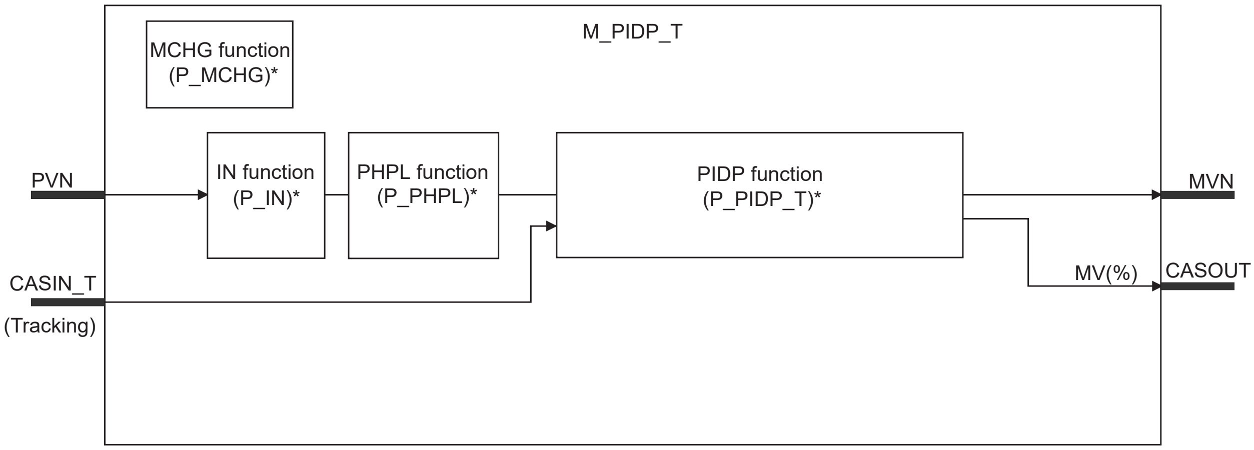
Corresponding tag type
PIDP

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute position type basic PID control taking function of P_IN+P_PHPL+P_PIDP_T as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	PIDP_NMIN to PIDP_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PIDP_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PIDP_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PIDP_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PIDP_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	PIDP_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PIDP_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	PIDP_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	PIDP_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PIDP function	P_PIDP_T	Section 8.2.9
MCHG function	P_MCHG	Section 8.3.1

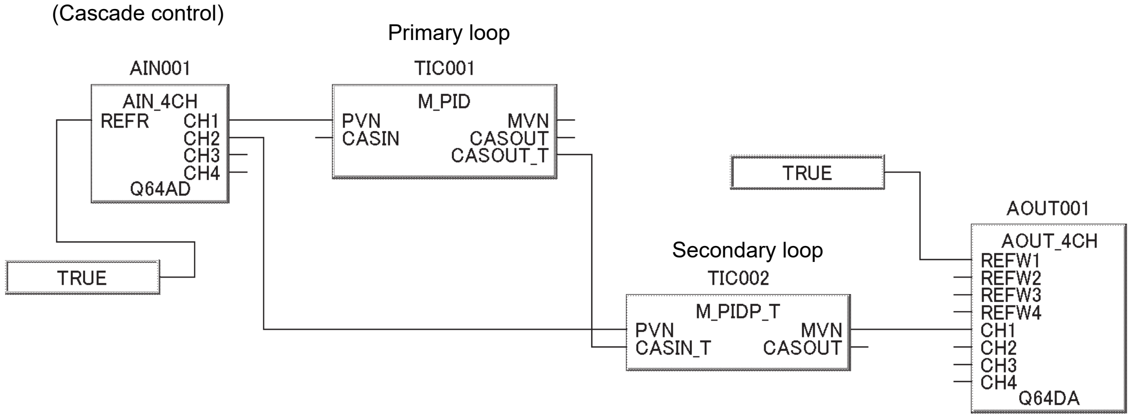
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

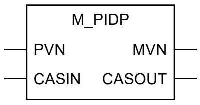
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(Cascade control)



9.1.12 Position Type PID Control (Without Tracking to primary loop, Without Tracking from secondary loop) (M_PIDP)

FB	FBD parts
M_PIDP	

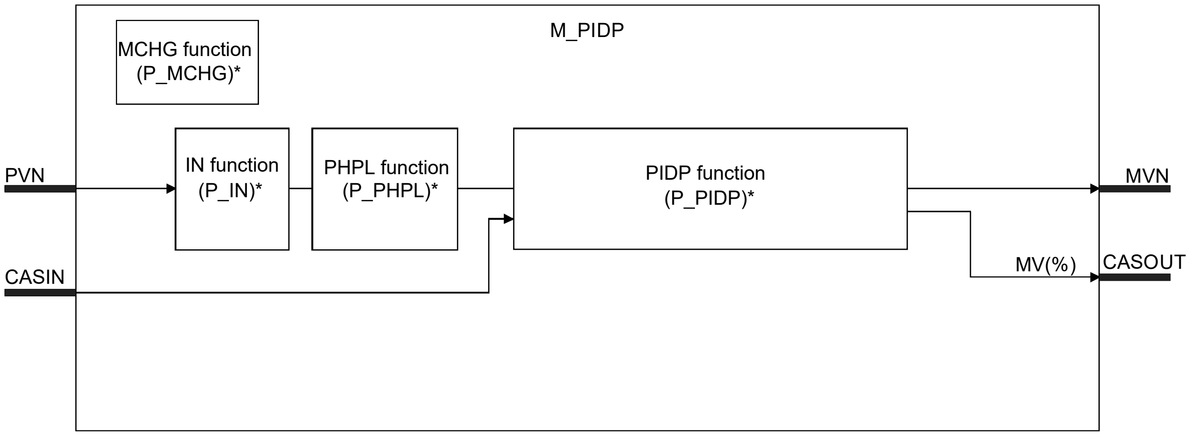
Corresponding tag type
PIDP

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute position type basic PID control taking function of P_IN+P_PHPL+P_PIDP as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	PIDP_NMIN to PIDP_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PIDP_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PIDP_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PIDP_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PIDP_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PIDP_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	PIDP_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to MAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to MAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

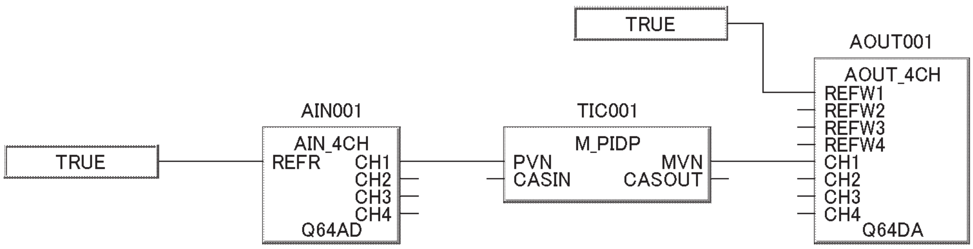
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PIDP function	P_PIDP	Section 8.2.10
MCHG function	P_MCHG	Section 8.3.1

Error

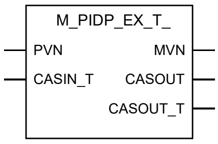
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.13 Position Type PID Control (With Tracking to primary loop, With Tracking from secondary loop) (M_PIDP_EX_T_)

FB	FBD parts
M_PIDP_EX_T_	

Corresponding tag type
PIDP

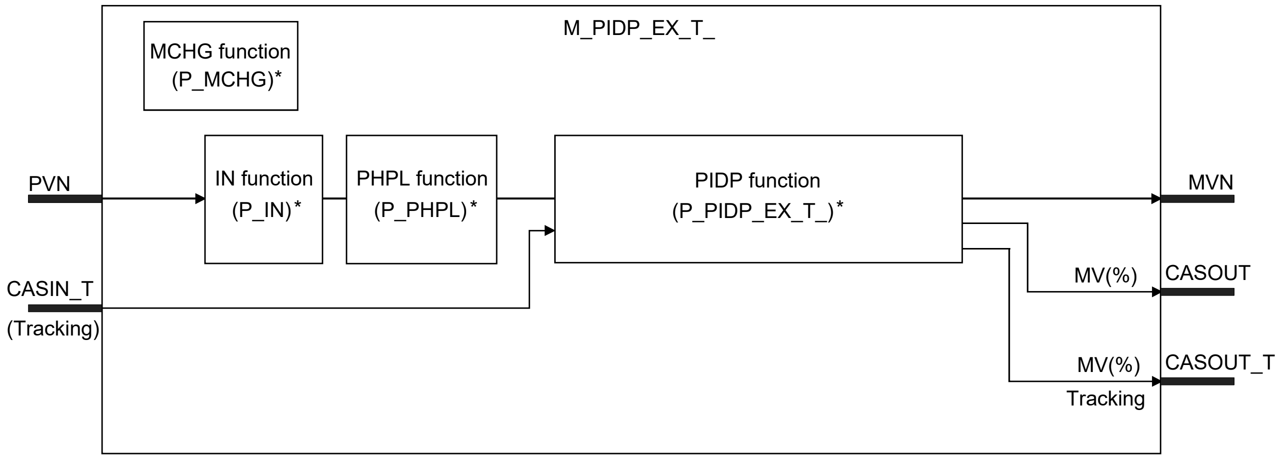
Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute position type basic PID control taking function of P_IN+P_PHPL+P_PIDP_EX_T_ as a single FB. MV value bumpless switching and tracking from the secondary loop at control mode change are also possible. *1

Function/FB classification name: Tag FB _ loop tag FB

*1 Requires Process CPU or Redundant CPU whose upper five digits of serial No. are 07032 or later.

Block Diagram



* (P_☐) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	PIDP_NMIN to PIDP_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PIDP_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PIDP_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PIDP_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PIDP_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	PIDP_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PIDP_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	PIDP_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	PIDP_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PIDP function	P_PIDP_EX_T_	Section 8.2.11
MCHG function	P_MCHG	Section 8.3.1

Restriction

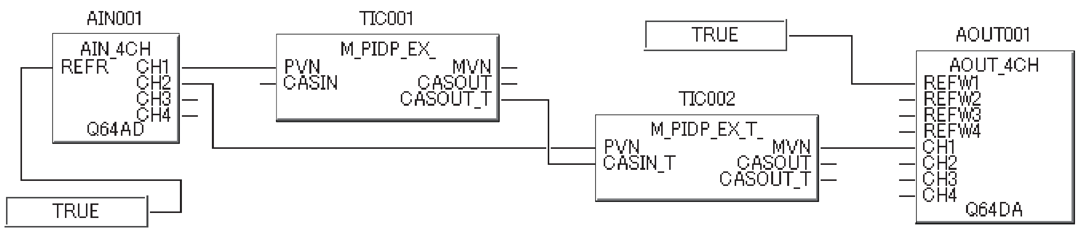
This FB supports the Process CPU and the Redundant CPU whose upper five digits of serial No.s are 07032 or later.
If either of the CPU whose upper five digits of serial No. are 07031 or earlier is used, the MV value bumpless switching and tracking processing from the secondary loop at control mode change will not be executed.

Error

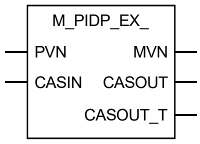
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.14 Position Type PID Control (Without Tracking to primary loop, With Tracking from secondary loop) (M_PIDP_EX_)

FB	FBD parts
M_PIDP_EX_	

Corresponding tag type
PIDP

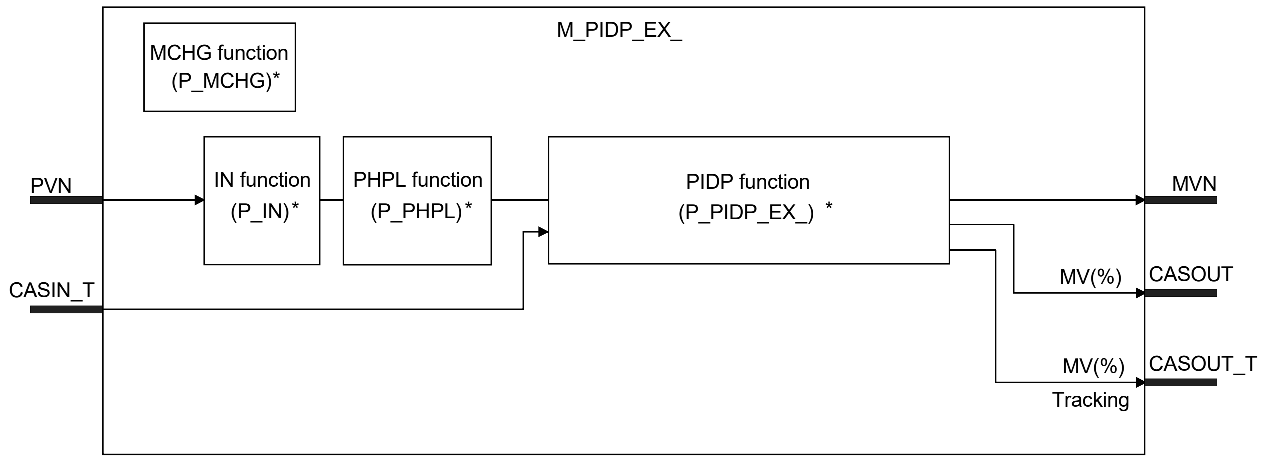
Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute position type basic PID control taking function of P_IN+P_PHPL+P_PIDP_EX_ as a single FB.
MV value bumpless switching and tracking from the secondary loop at control mode change are also possible. *1

Function/FB classification name: Tag FB_loop tag FB

*1 Requires Process CPU or Redundant CPU whose upper five digits of serial No. are 07032 or later.

Block Diagram



* (P_☐) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	PIDP_NMIN to PIDP_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	PIDP_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	PIDP_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PIDP_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PIDP_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PIDP_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	PIDP_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to MAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to MAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PIDP function	P_PIDP_EX_	Section 8.2.12
MCHG function	P_MCHG	Section 8.3.1

Restriction

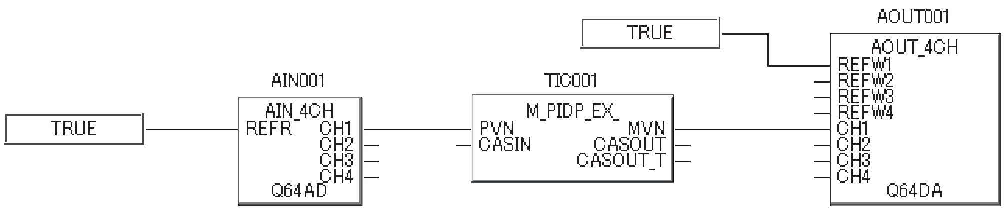
This FB supports the Process CPU and the Redundant CPU whose upper five digits of serial No.s are 07032 or later.
If either of the CPU whose upper five digits of serial No. are 07031 or earlier is used, the MV value bumpless switching and tracking processing from the secondary loop at control mode change will not be executed.

Error

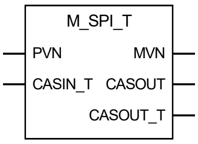
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.15 Sample PI Control (With Tracking to primary loop) (M_SPI_T)

FB	FBD parts
M_SPI_T	

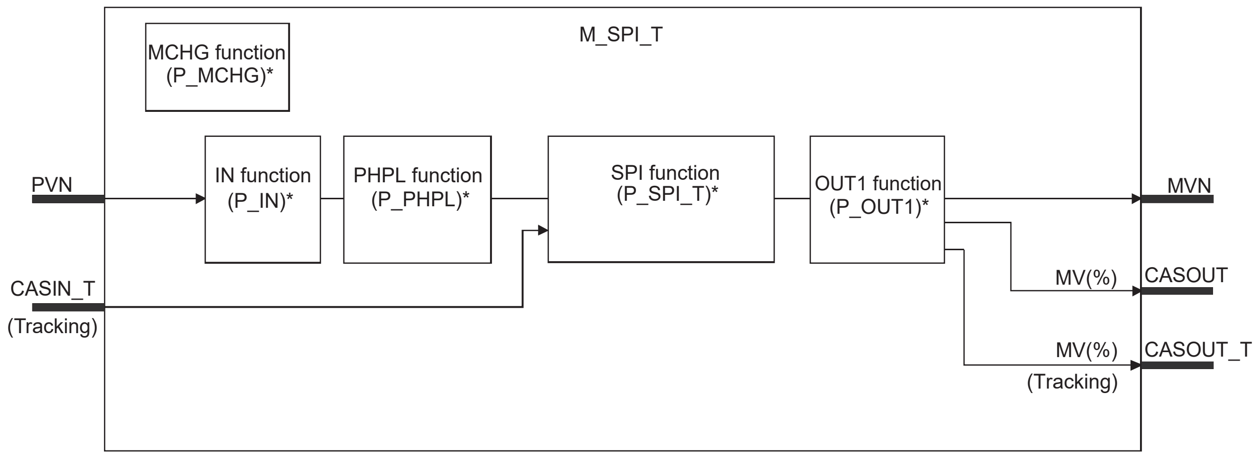
Corresponding tag type
SPI

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute sample PI control taking function of P_IN+P_PHPL+P_SPI_T+P_OUT1 as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT1_NMIN to OUT1_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	SPI_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	SPI_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SPI_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SPI_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SPI_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	OUT1_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT1_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

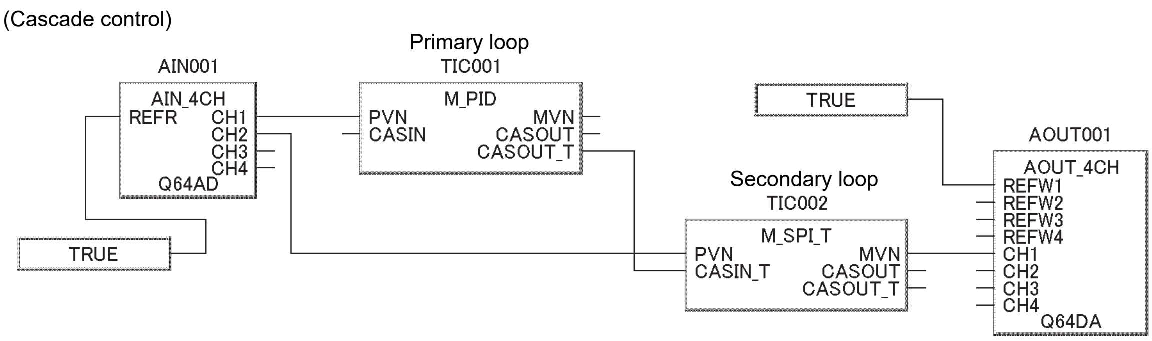
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
SPI function	P_SPI_T	Section 8.2.13
OUT1 function	P_OUT1	Section 8.1.2
MCHG function	P_MCHG	Section 8.3.1

Error

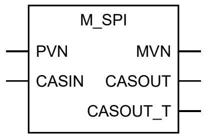
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.16 Sample PI Control (Without Tracking to primary loop) (M_SPI)

FB	FBD parts
M_SPI	

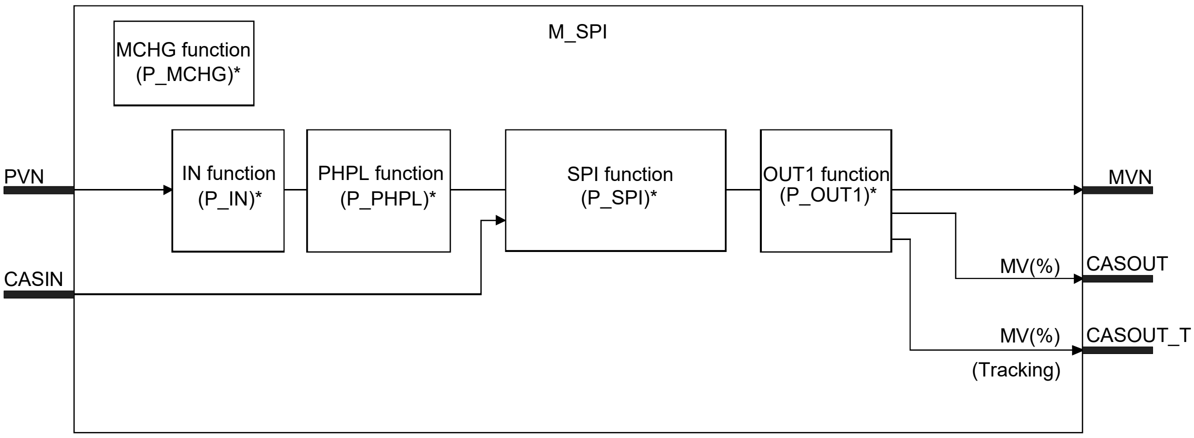
Corresponding tag type
SPI

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute sample PI control taking function of P_IN+P_PHPL+P_SPI+P_OUT1 as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT1_NMIN to OUT1_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	SPI_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	SPI_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	SPI_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	OUT1_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT1_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicate the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

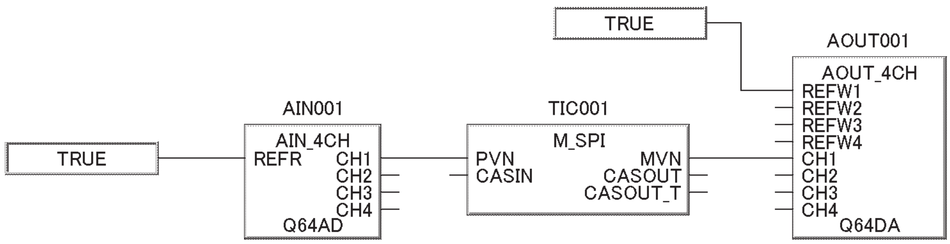
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
SPI function	P_SPI	Section 8.2.14
OUT1 function	P_OUT1	Section 8.1.2
MCHG function	P_MCHG	Section 8.3.1

Error

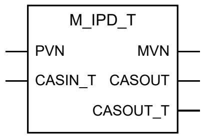
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.17 I-PD Control (With Tracking to primary loop) (M_IPD_T)

FB	FBD parts
M_IPD_T	

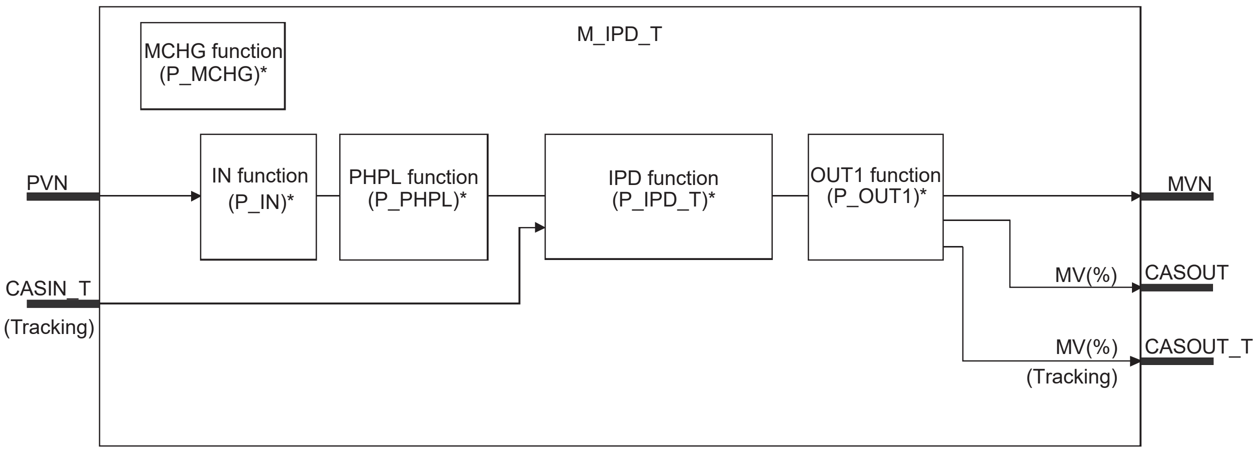
Corresponding tag type
IPD

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute I-PD control taking function of P_IN+P_PHPL+P+IPD_T+P_OUT1 as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT1_NMIN to OUT1_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	IPD_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	IPD _DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	IPD _PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	IPD _TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	IPD _SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	IPD _SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	OUT1_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT1_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

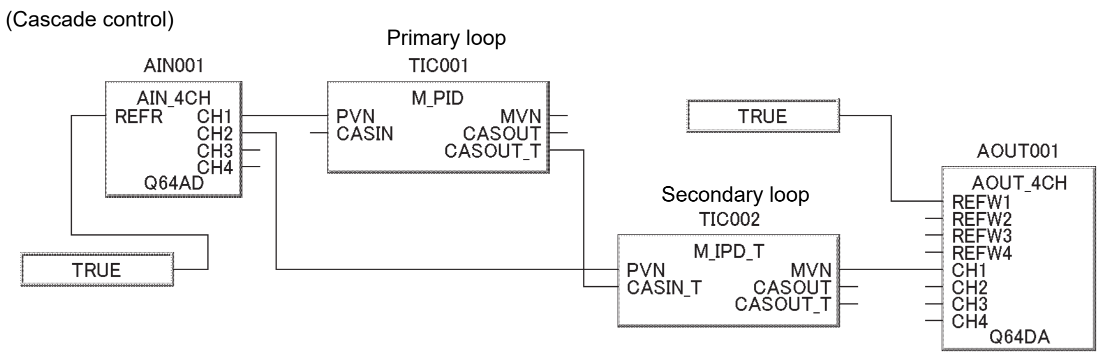
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
IPD function	P_IPD_T	Section 8.2.15
OUT1 function	P_OUT1	Section 8.1.2
MCHG function	P_MCHG	Section 8.3.1

Error

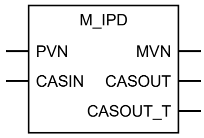
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.18 I-PD Control (Without Tracking to primary loop) (M_IPD)

FB	FBD parts
M_IPD	

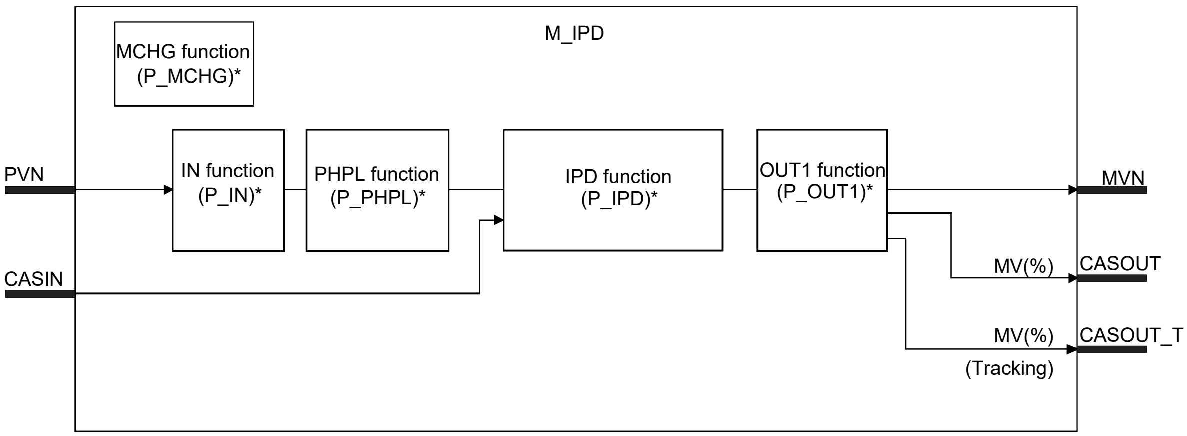
Corresponding tag type
IPD

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute I-PD control taking function of P_IN+P_PHPL+P_IPD+P_OUT1 as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT1_NMIN to OUT1_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	IPD_MTD	Public variable	REAL	Derivative gain	0 to 9999	8.0	User
	IPD_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	IPD_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	IPD_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	OUT1_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT1_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

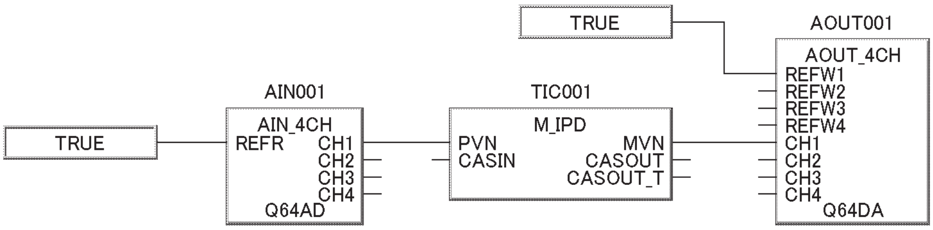
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
IPD function	P_IPD	Section 8.2.16
OUT1 function	P_OUT1	Section 8.1.2
MCHG function	P_MCHG	Section 8.3.1

Error

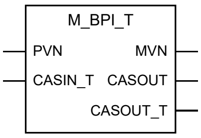
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.19 Blend PI Control (With Tracking to primary loop) (M_BPI_T)

FB	FBD parts
M_BPI_T	

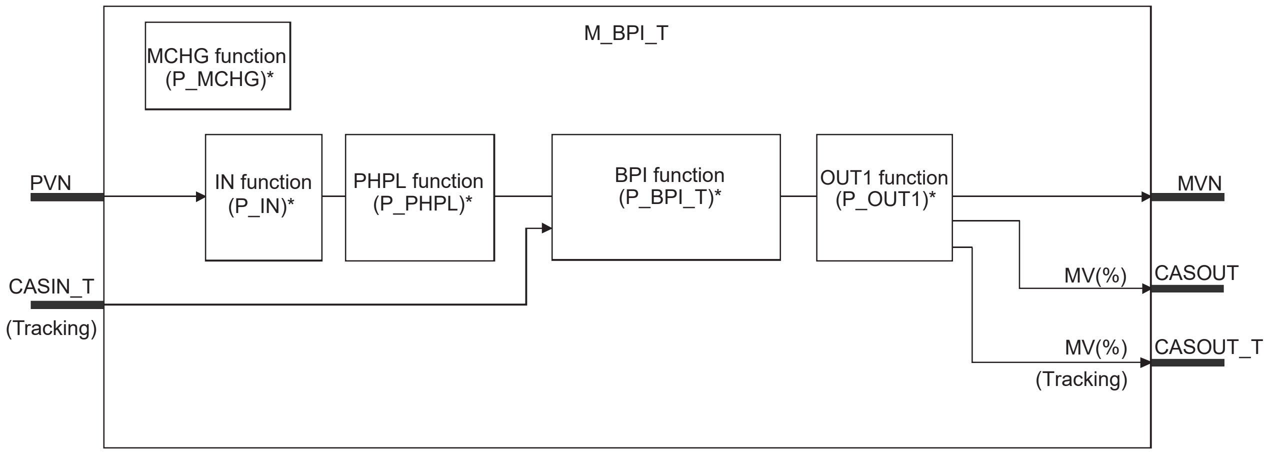
Corresponding tag type
BPI

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute blend PI control taking function of P_IN+P_PHPL+P_BPI_T+P_OUT1 as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT1_NMIN to OUT1_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	BPI_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	BPI _PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	BPI _TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	BPI_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	BPI_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	BPI_RST_SDV_ ON_CHGMODE	Public variable	BOOL	DV cumulative value reset in control mode change TRUE: Reset DV cumulative value (SDV) in control mode change (MAN/CMV → AUT/CAS/CSV) FALSE: Not reset DV cumulative value (SDV)	TRUE, FALSE	TRUE	User
	OUT1_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT1_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
Operation processing	BPI_RST_SDV	Public variable	BOOL	DV cumulative value reset FALSE → TRUE: DV cumulative value (SDV) reset	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the simulation processing.

*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

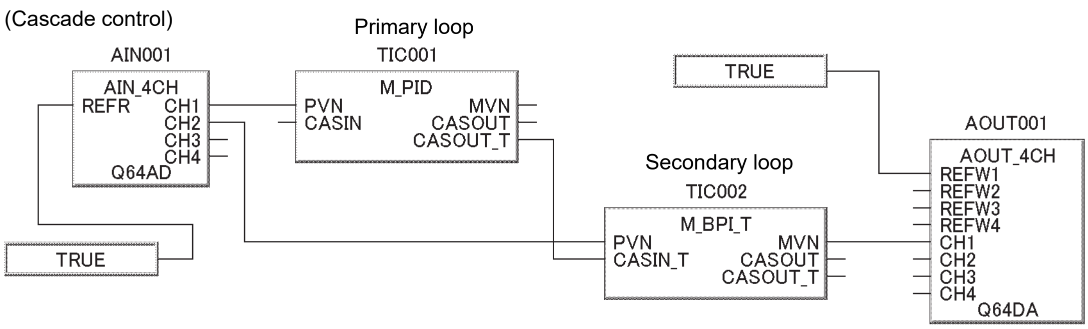
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
BPI function	P_BPI_T	Section 8.2.17
OUT1 function	P_OUT1	Section 8.1.2
MCHG function	P_MCHG	Section 8.3.1

Error

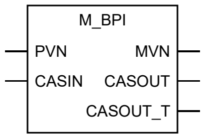
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.20 Blend PI Control (Without Tracking to primary loop) (M_BPI)

FB	FBD parts
M_BPI	

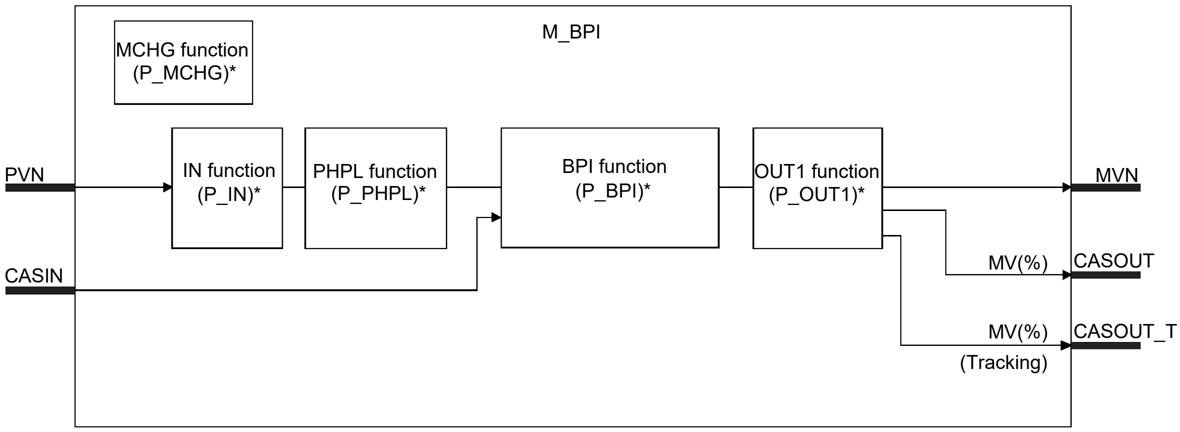
Corresponding tag type
BPI

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute blend PI control taking function of P_IN+P_PHPL+P_BPI+P_OUT1 as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT1_NMIN to OUT1_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade MV output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	BPI_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	BPI _PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	BPI_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	BPI_RST_SDV_ON_CHGMODE	Public variable	BOOL	DV cumulative value reset in control mode change TRUE: Reset DV cumulative value (SDV) in control mode change (MAN/CMV → AUT/CAS/CSV) FALSE: Not reset DV cumulative value (SDV)	TRUE, FALSE	TRUE	User
	OUT1_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT1_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
Operation processing	BPI_RST_SDV	Public variable	BOOL	DV cumulative value reset FALSE → TRUE: DV cumulative value (SDV) reset	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

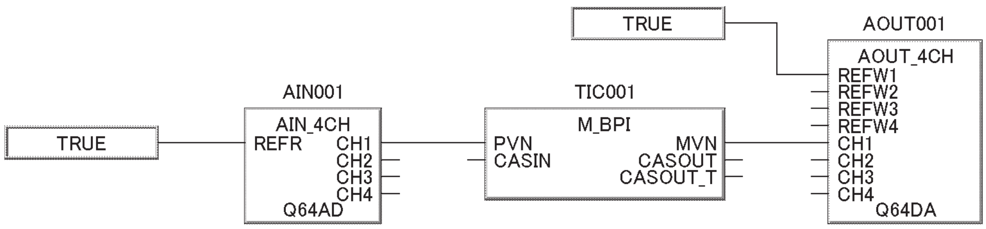
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
BPI function	P_BPI	Section 8.2.18
OUT1 function	P_OUT1	Section 8.1.2
MCHG function	P_MCHG	Section 8.3.1

Error

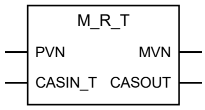
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.21 Ratio Control (With Tracking to primary loop) (M_R_T)

FB	FBD parts
M_R_T	

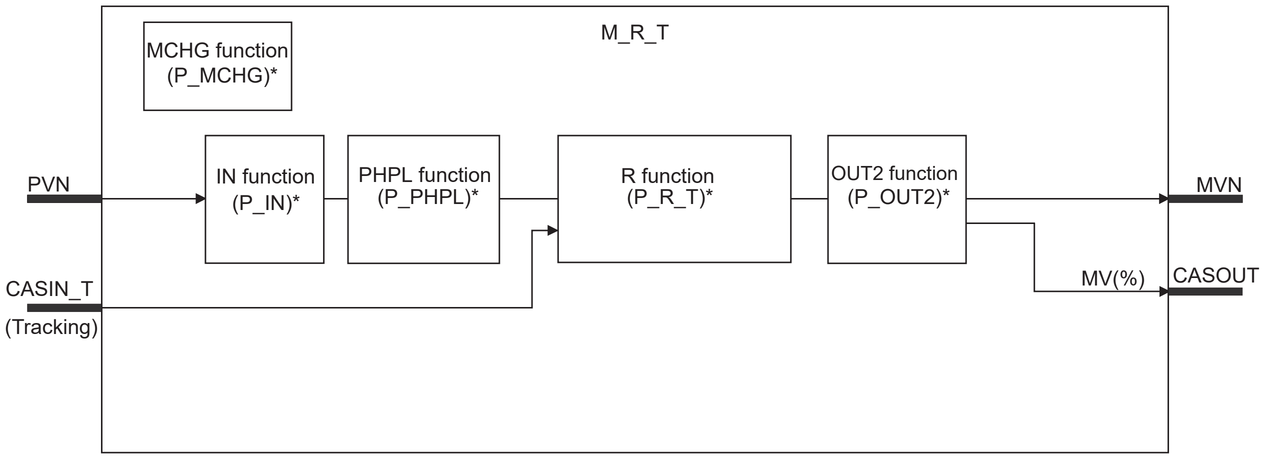
Corresponding tag type
R

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute ratio control taking function of P_IN+P_PHPL+P_R_T+P_OUT2 as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT2_NMIN to OUT2_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	R_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	R_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	R_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	OUT2_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT2_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

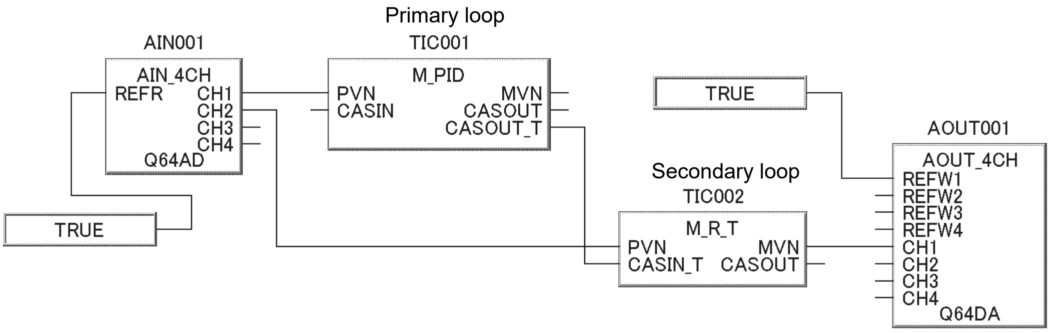
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
R function	P_R_T	Section 8.2.1
OUT2 function	P_OUT2	Section 8.1.3
MCHG function	P_MCHG	Section 8.3.1

Error

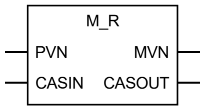
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.22 Ratio Control (Without Tracking to primary loop) (M_R)

FB	FBD parts
M_R	

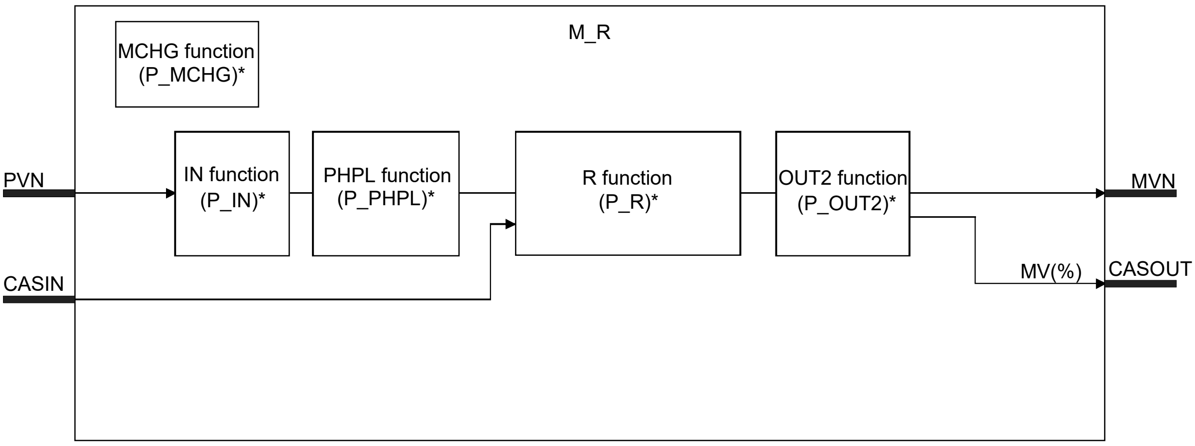
Corresponding tag type
R

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute ratio control taking function of P_IN+P_PHPL+P_R+P_OUT2 as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	OUT2_NMIN to OUT2_NMAX
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	R_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	OUT2_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	OUT2_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

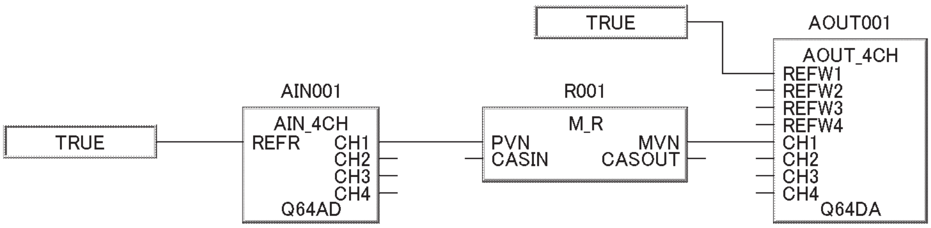
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
R function	P_R	Section 8.2.2
OUT2 function	P_OUT2	Section 8.1.3
MCHG function	P_MCHG	Section 8.3.1

Error

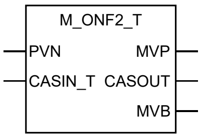
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.23 2 Position ON/OFF Control (With Tracking to primary loop) (M_ONF2_T)

FB	FBD parts
M_ONF2_T	

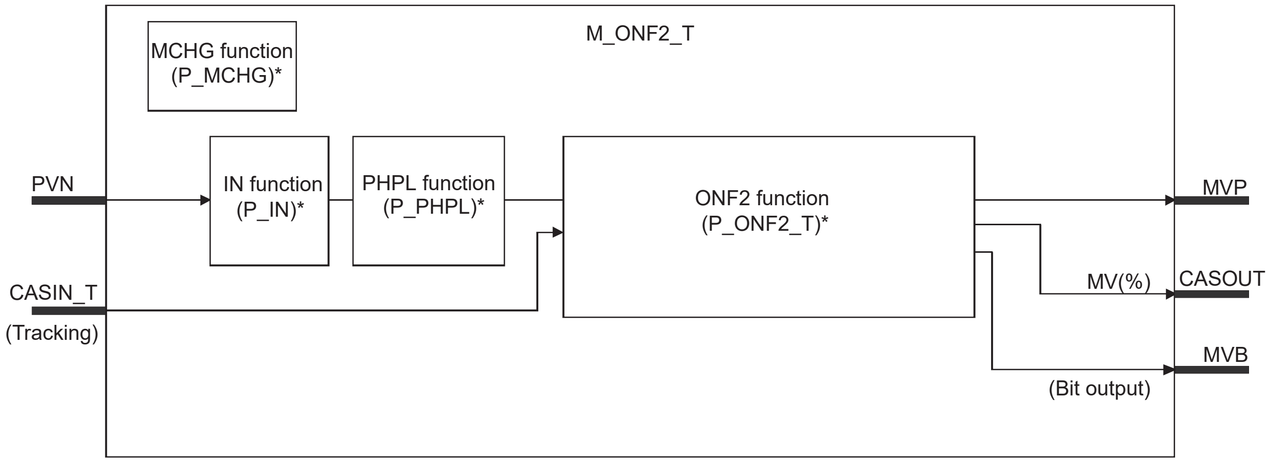
Corresponding tag type
ONF2

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 2 position ON/OFF control taking function of P_IN+P_PHPL+P_ONF2_T as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVP	Output variable	REAL	ΔMV output (unit: %)	0 to 100
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	MVB	Output variable	BOOL	ON/OFF output (ON if MV≥50%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	ONF2_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	ONF2_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	ONF2_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	ONF2_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
OUT2 function	P_ONF2_T	Section 8.2.20
MCHG function	P_MCHG	Section 8.3.1

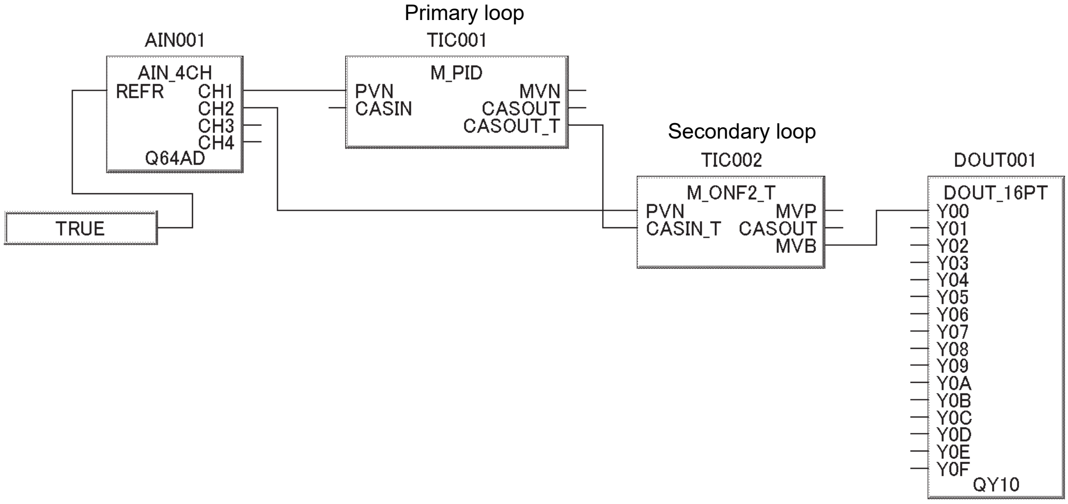
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

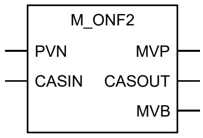
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(Cascade control)



9.1.24 2 Position ON/OFF Control (Without Tracking to primary loop) (M_ONF2)

FB	FBD parts
M_ONF2	

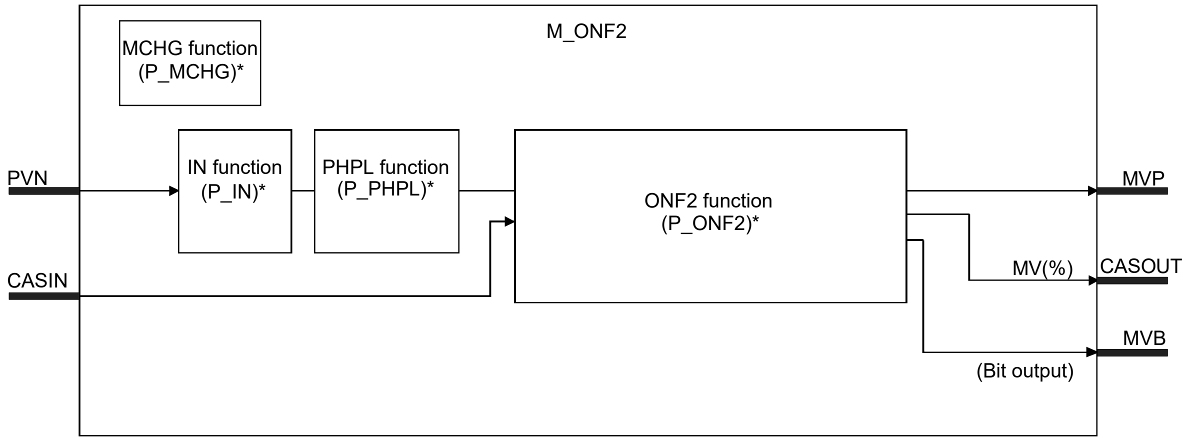
Corresponding tag type
ONF2

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 2 position ON/OFF control taking function of P_IN+P_PHPL+P_ONF2 as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVP	Output variable	REAL	∆MV output (Unit: %)	0 to 100
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	MVB	Output variable	BOOL	ON/OFF output (ON if MV≧50%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	ONF2_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	ONF2_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

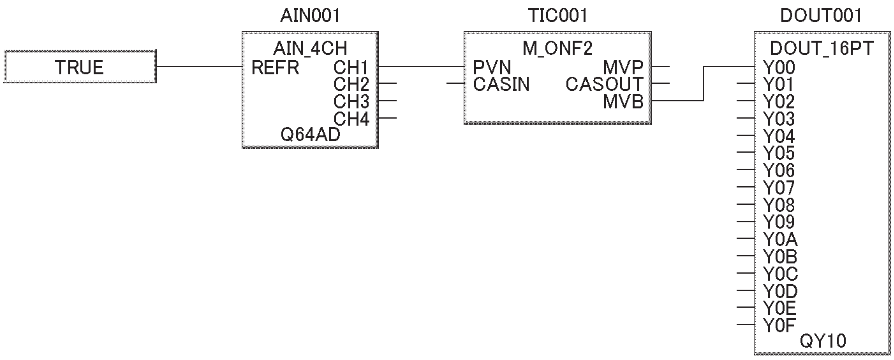
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
ONF2 function	P_ONF2	Section 8.2.21
MCHG function	P_MCHG	Section 8.3.1

Error

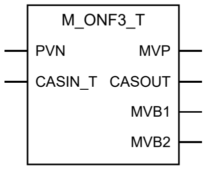
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.25 3 Position ON/OFF Control (With Tracking to primary loop) (M_ONF3_T)

FB	FBD parts
M_ONF3_T	

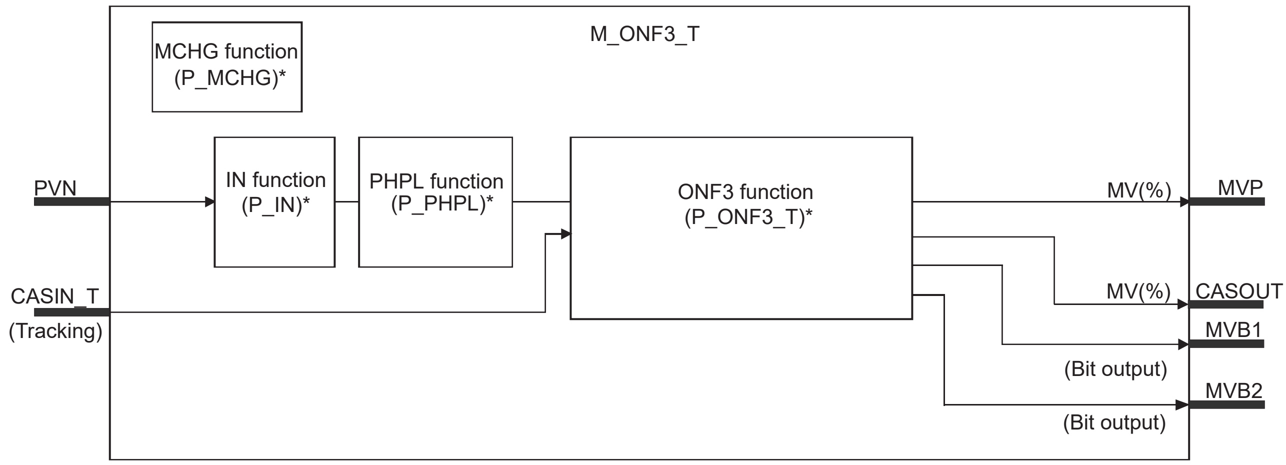
Corresponding tag type
ONF3

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 3 position ON/OFF control taking function of P_IN+P_PHPL+P_ONF3_T as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (Unit: %) (With tracking)	0 to 100
Output	MVP	Output variable	REAL	ΔMV output (Unit: %)	0 to 100
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	MVB1	Output variable	BOOL	ON/OFF output (ON if MV≧75%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE
	MVB2	Output variable	BOOL	ON/OFF output (ON if MV<25%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	ONF3_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	ONF3_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	ONF3_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	ONF3_SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
ONF3 function	P_ONF3_T	Section 8.2.22
MCHG function	P_MCHG	Section 8.3.1

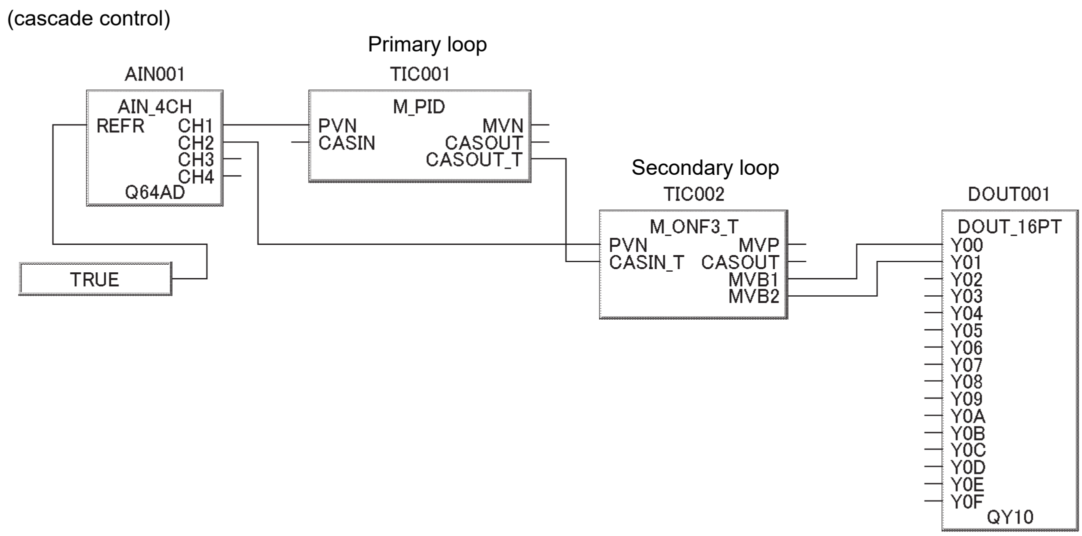
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

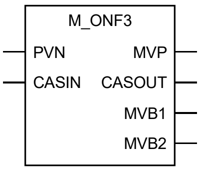
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.26 3 Position ON/OFF Control (Without Tracking to primary loop) (M_ONF3)

FB	FBD parts
M_ONF3	

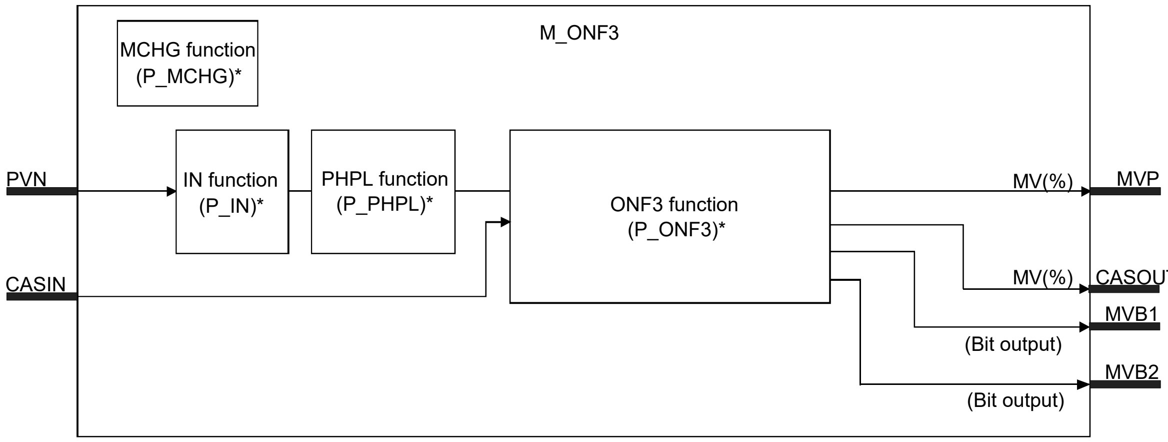
Corresponding tag type
ONF3

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute 3 position ON/OFF control taking function of P_IN+P_PHPL+P_ONF3 as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (Unit: %)	0 to 100
Output	MVP	Output variable	REAL	∆MV output (Unit: %)	0 to 100
	CASOUT	Output variable	REAL	Cascade MV output (Unit: %)	0 to 100
	MVB1	Output variable	BOOL	ON/OFF output (ON if MV≧75%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE
	MVB2	Output variable	BOOL	ON/OFF output (ON if MV<25%) (TRUE: ON, FALSE: OFF)	TRUE, FALSE

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	ONF3_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	ONF3_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3:CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the simulation processing.
*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

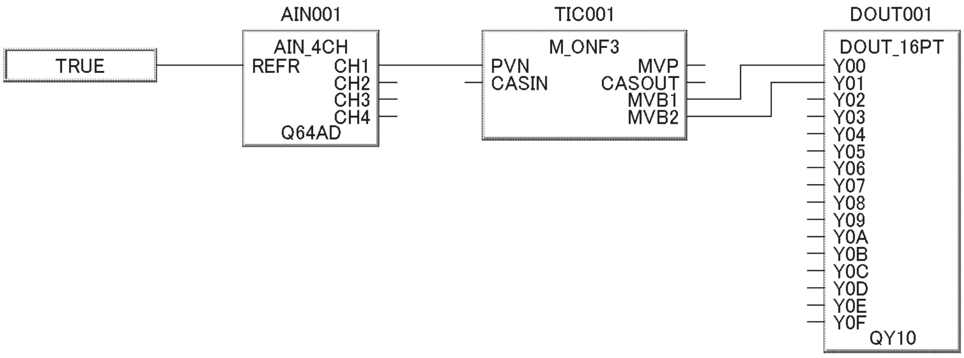
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
ONF3 function	P_ONF3	Section 8.2.23
MCHG function	P_MCHG	Section 8.3.1

Error

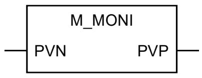
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.27 Monitor (M_MONI)

FB	FBD parts
M_MONI	

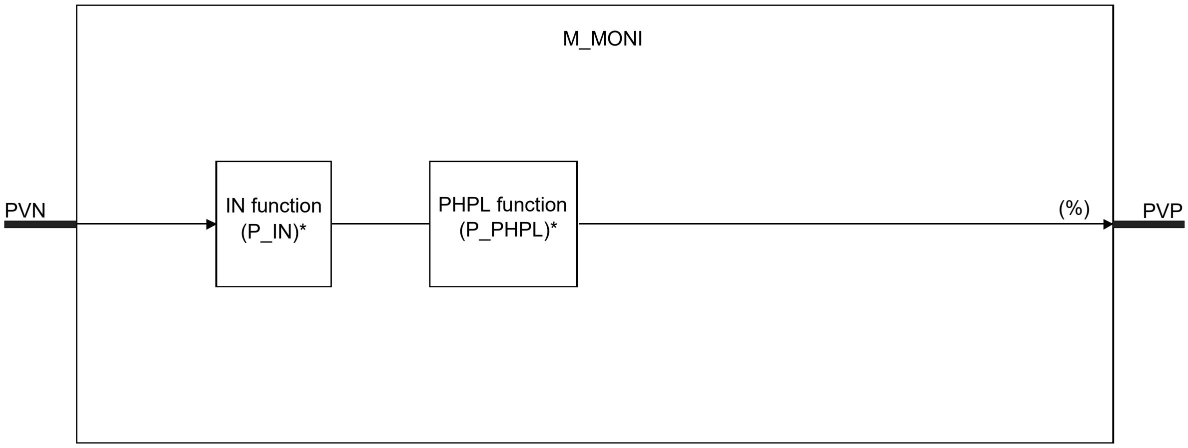
Corresponding tag type
MONI

Control mode				
MAN	AUT	CAS	CMV	CSV
—	—	—	—	—

Function overview: Execute monitoring taking function of P_IN+P_PHPL as a single FB.

Function/FB classification name: Manual output with Monitor

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
Output	PVP	Output variable	REAL	PV output (Unit: %)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

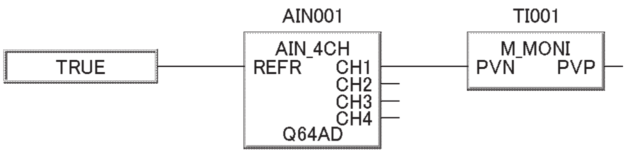
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19

Error

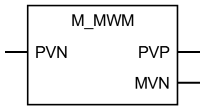
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.28 Manual Output With Monitor (M_MWM)

FB	FBD parts
M_MWM	

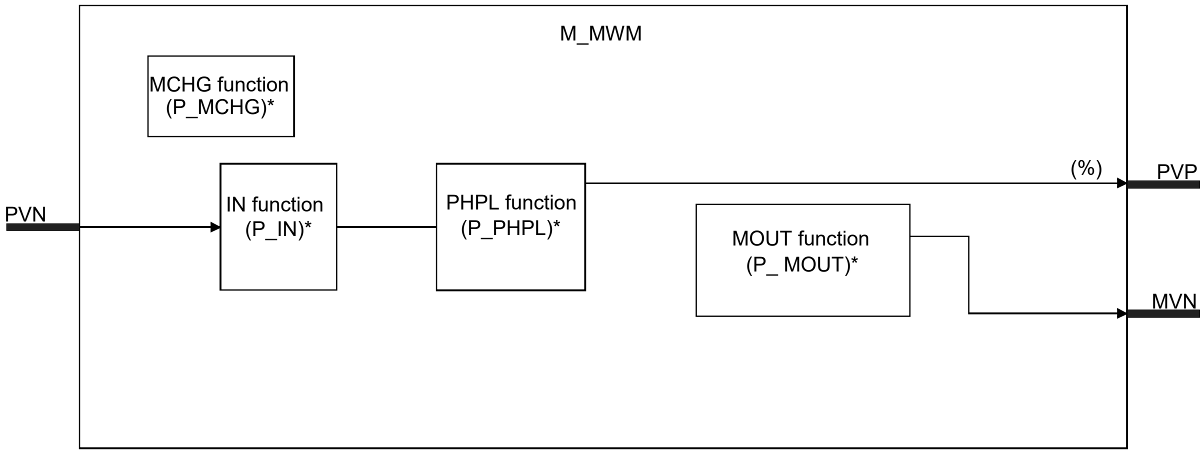
Corresponding tag type
MWM

Control mode				
MAN	AUT	CAS	CMV	CSV
○	—	—	○	—

Function overview: Execute manual output with monitor taking function of P_IN+P_PHPL+P_MOUT as function of a single FB.

Function/FB classification name: Tag access FB _ I/O control operation FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
Output	PVP	Output variable	REAL	PV output (Unit: %)	0 to 100
	MVN	Output variable	REAL	Module FB output	MOUT_NMIN to MOUT_NMAX

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	MOUT_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	MOUT_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

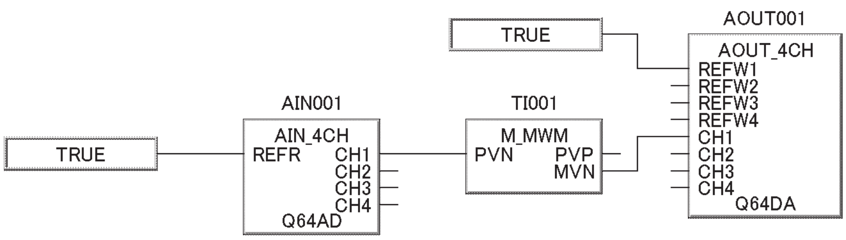
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
OUT1 function	P_MOUT	Section 8.1.5

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.29 Batch Preparation (M_BC)

FB	FBD parts
M_BC	M_BC Inputs: RUN, HOLD, STPRS, RS_START, CIN Outputs: COMP1, COMP2, COMP

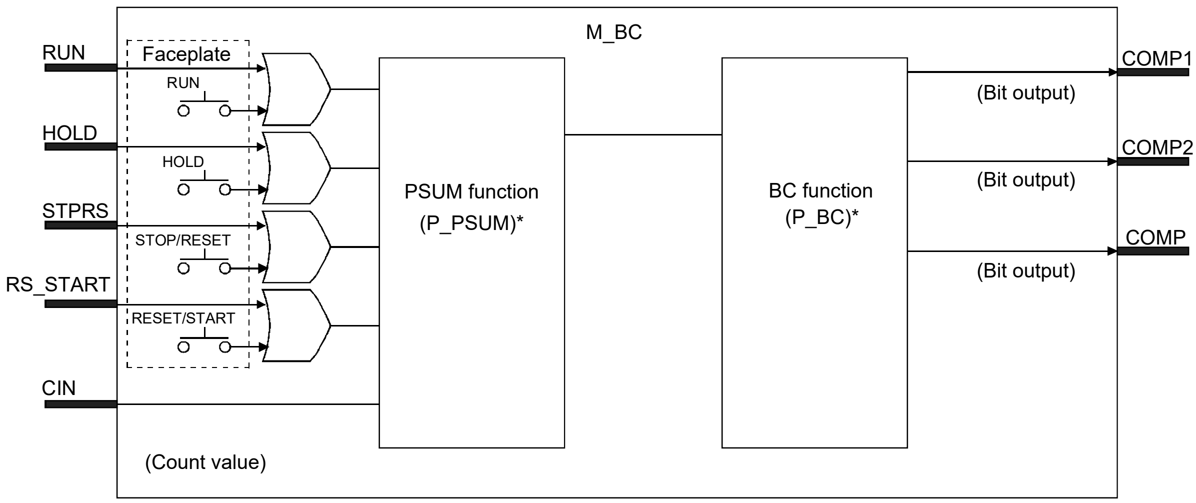
Corresponding tag type
BC

Control mode				
MAN	AUT	CAS	CMV	CSV
—	—	—	—	—

Function overview: Execute batch preparation taking function of P_PSUM+P_BC as a single FB.

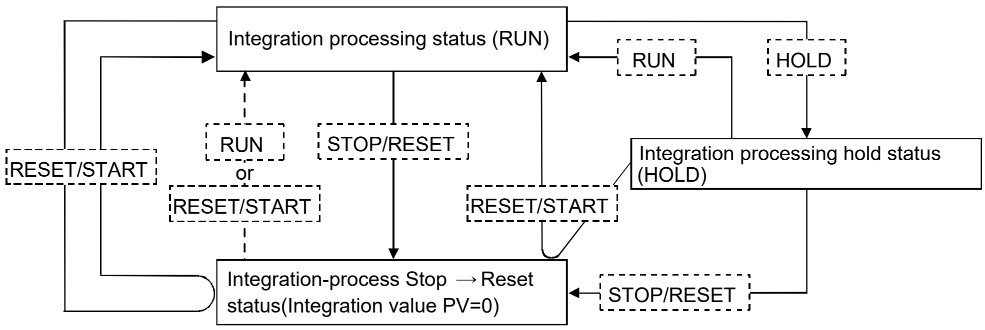
Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Indicates accumulation processing status according to input condition.



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	RUN	Input variable	BOOL	Integration start signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	HOLD	Input variable	BOOL	Integration stop signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	STPRS	Input variable	BOOL	Reset signal after integration stop (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	RS_START	Input variable	BOOL	Start signal after integration reset (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	CIN	Input variable	DINT	Count value	Ring counter of -2147483648 to 2147483647 (however, increment pulse of each time will be below 32767)
Output	COMP1	Output variable	BOOL	Setting Value 1 (SV1) completed output (TRUE: ON, FALSE: OFF)	TRUE, FALSE
	COMP2	Output variable	BOOL	Setting Value 2 (SV2) completed output (TRUE: ON, FALSE: OFF)	TRUE, FALSE
	COMP	Output variable	BOOL	Setting value (SV) completed output (TRUE: ON, FALSE: OFF) It is TRUE when Count value (CIN) and Setting value (SV) are coincidence.	TRUE, FALSE

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	PSUM_W	Public variable	INT	Weight per pulse	1 to 999	1	User
	PSUM_U	Public variable	INT	Unit conversion constant	1,10,100,1000	1	User
	PSUM_HILMT	Public variable	DINT	High limit value of integration	1 to 2147483647	2147483647	User
	PSUM_SUMPTN	Public variable	INT	Integration pattern 0: return to 0 when it is beyond integration high limit. 1: hold high limit value when it is beyond integration high limit.	0,1	0	User

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

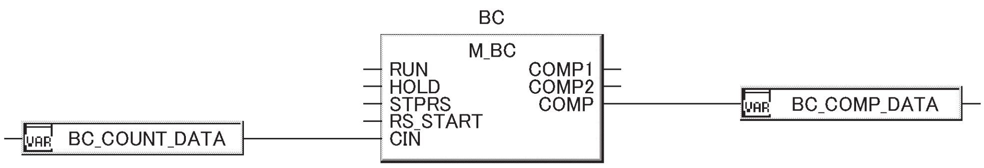
Item	Main structure member tag access FB	Reference
PSUM function	P_PSUM	Section 8.1.7
BC function	P_BC	Section 8.1.8

Error

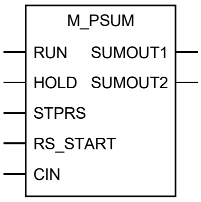
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.30 Pulse Integrator (M_PSUM)

FB	FBD parts
M_PSUM	

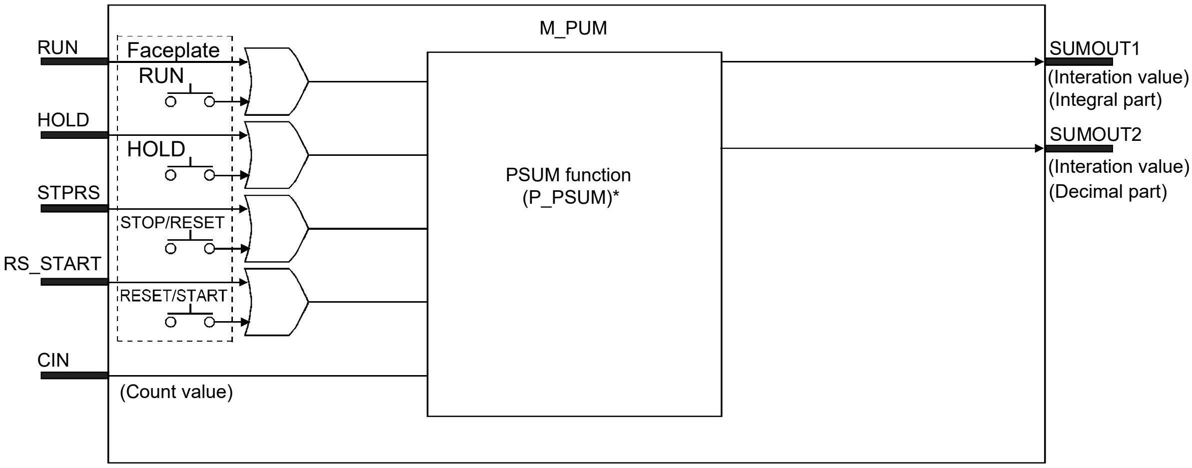
Corresponding tag type
PSUM

Control mode				
MAN	AUT	CAS	CMV	CSV
—	—	—	—	—

Function overview: Execute pulse integration taking function of P_PSUM as a single FB.

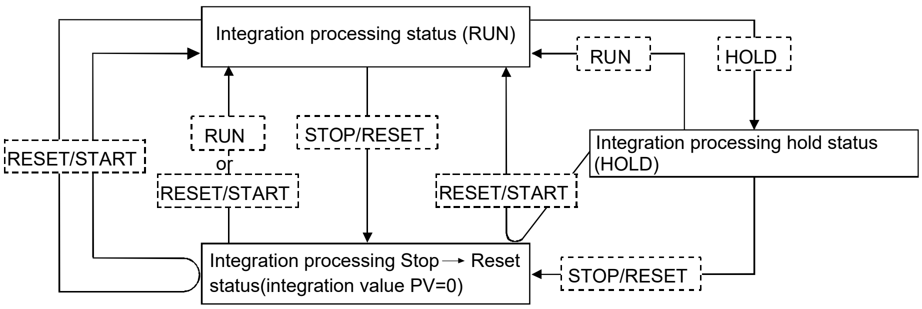
Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Indicates accumulation processing status according to input condition.



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	RUN	Input variable	BOOL	Integration start signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	HOLD	Input variable	BOOL	Integration stop signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	STPRS	Input variable	BOOL	Reset signal after integration stop (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	RS_START	Input variable	BOOL	Start signal after integration reset (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	CIN	Input variable	DINT	Count value	Ring counter of -2147483648 to 2147483647 (however, added pulse of each time will be below 32767)
Output	SIMOUT1	Output variable	DINT	Integration value (integral part) output	0 to 99999999
	SIMOUT2	Output variable	DINT	Integration value (decimal part) output	0 to 999

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	PSUM_W	Public variable	INT	Weight per pulse	1 to 999	1	User
	PSUM_U	Public variable	INT	Unit conversion constant	1,10,100,1000	1	User
	PSUM_HILMT	Public variable	DINT	High limit value of integration	1 to 2147483647	2147483647	User
	PSUM_SUMPTN	Public variable	INT	Integration pattern 0: return to 0 when it is beyond integration high limit. 1: hold high limit value when it is beyond integration high limit.	0,1	0	User

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

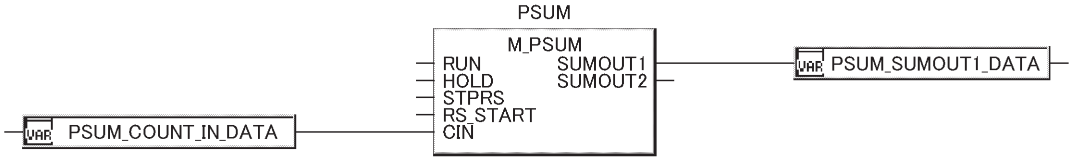
Item	Main structure member tag access FB	Reference
PSUM function	P_PSUM	Section 8.1.7

Error

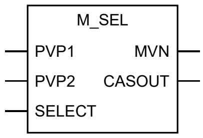
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.31 Loop Selector (Without Tracking to primary loop) (M_SEL)

FB	FBD parts
M_SEL	

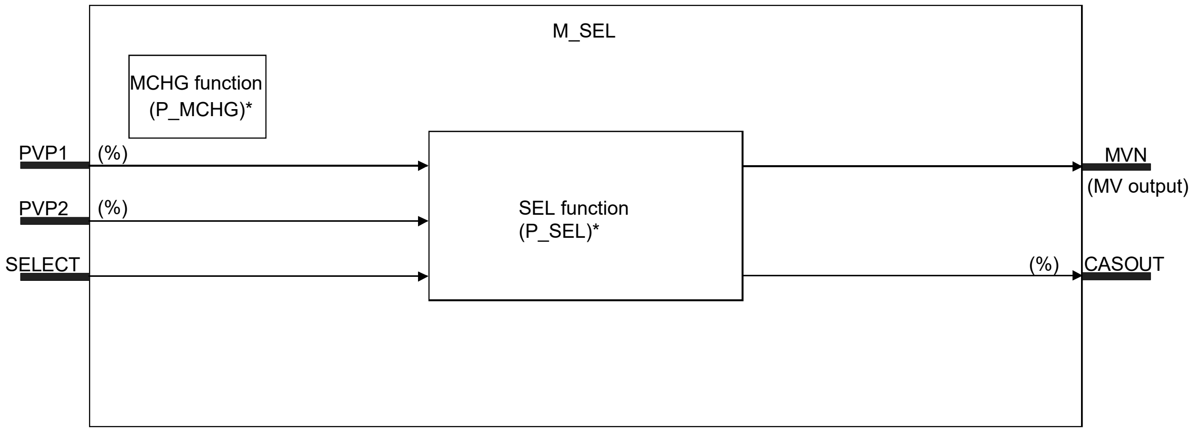
Corresponding tag type
SEL

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute loop selector taking function of P_SEL as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP1	Input variable	REAL	PV input (Unit: %)	0 to 100
	PVP2	Input variable	REAL	PV input (Unit: %)	0 to 100
	SELECT	Input variable	BOOL	Selection signal (TRUE: PVP2, FALSE: PVP1)	TRUE, FALSE
Output	MVN	Output variable	REAL	MV output to output module FB	SEL_NMIN to SEL_NMAX
	CASOUT	Output variable	REAL	Cascade output (Unit:%)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	SEL_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	SEL_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

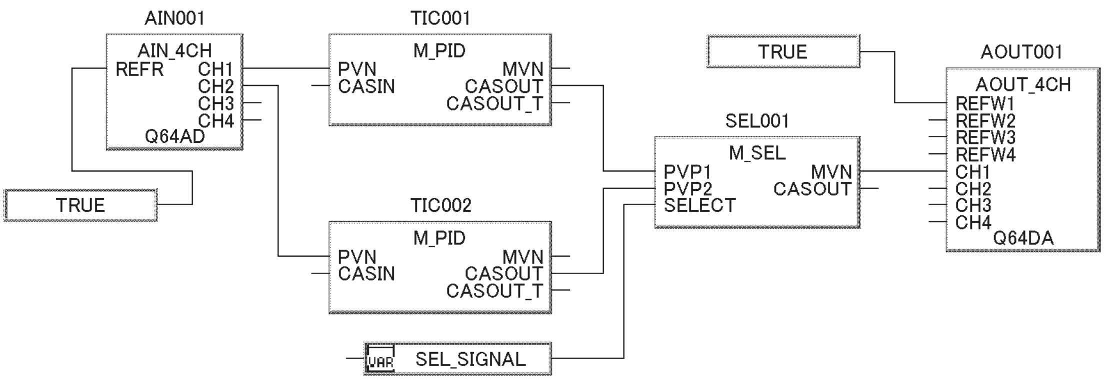
Item	Main structure member tag access FB	Reference
SEL function	P_SEL	Section 8.2.26
MCHG function	P_MCHG	Section 8.3.1

Error

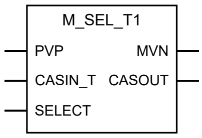
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.32 Loop Selector (With Tracking to primary loop) (M_SEL_T1)

FB	FBD parts
M_SEL_T1	

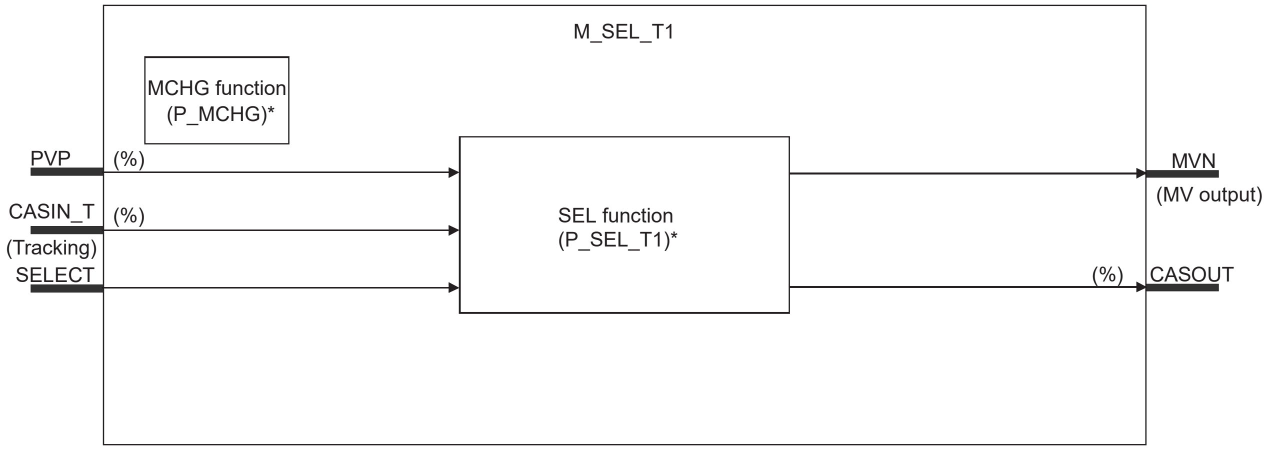
Corresponding tag type
SEL

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute loop selector taking function of P_SEL_Tt1 as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVP	Input variable	REAL	PV input (Unit: %)	0 to 100
	CASIN_T	Input variable	ADR_REAL	PV input (Unit: %) (With tracking)	0 to 100
	SELECT	Input variable	BOOL	Selection signal (TRUE: CASIN_T, FALSE: PVP)	TRUE, FALSE
Output	MVN	Output variable	REAL	MV output to output module FB	SEL_NMIN to SEL_NMAX
	CASOUT	Output variable	REAL	Cascade output (Unit:%)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	SEL_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	SEL_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	SEL_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SEL_SVPTN_B4	Public variable	BOOL	CASIN_T pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

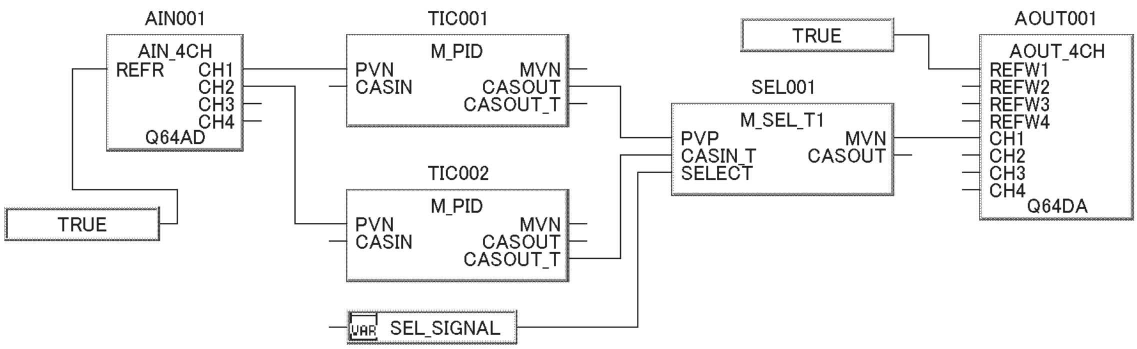
Item	Main structure member tag access FB	Reference
SEL function	P_SEL_T1	Section 8.2.27
MCHG function	P_MCHG	Section 8.3.1

Error

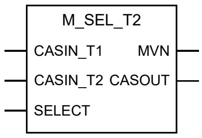
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.33 Loop Selector (With Tracking to primary loop) (M_SEL_T2)

FB	FBD parts
M_SEL_T2	

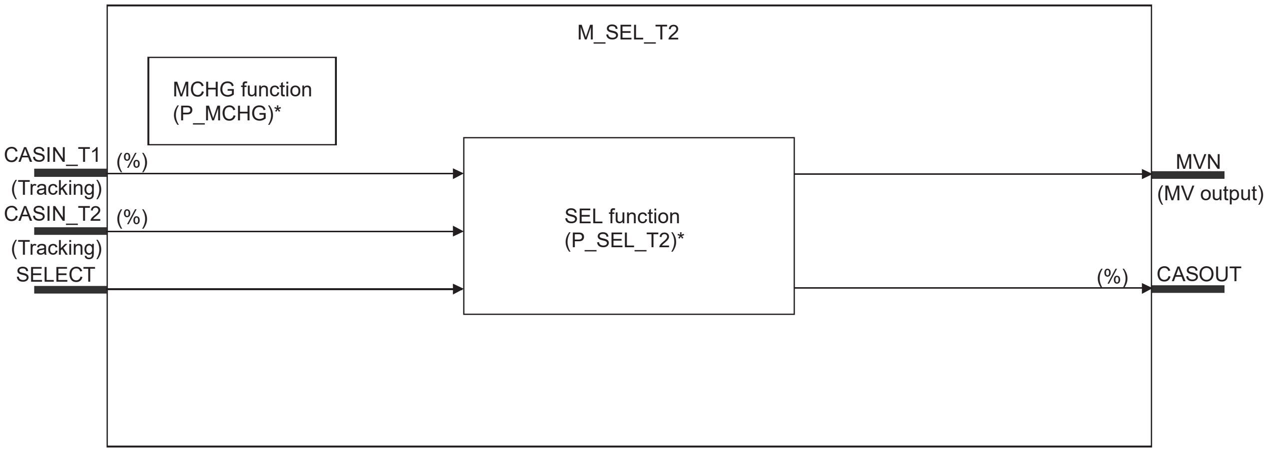
Corresponding tag type
SEL

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute loop selector taking function of P_SEL_T2 as a single FB.

Function/FB classification name: Tag access FB_I/O control operation FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	CASIN_T1	Input variable	ADR_REAL	PV input (Unit: %) (With tracking)	0 to 100
	CASIN_T2	Input variable	ADR_REAL	PV input (Unit: %) (With tracking)	0 to 100
	SELECT	Input variable	BOOL	Selection signal (TRUE: CASIN_T2, FALSE: CASIN_T1)	TRUE, FALSE
Output	MVN	Output variable	REAL	MV output to output module FB	SEL_NMIN to SEL_NMAX
	CASOUT	Output variable	REAL	Cascade output (Unit: %)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	SEL_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	SEL_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	SEL_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SEL_SVPTN_B1	Public variable	BOOL	CASIN_T1 used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SEL_SVPTN_B2	Public variable	BOOL	CASIN_T2 used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SEL_SVPTN_B3	Public variable	BOOL	CASIN_T1 pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	SEL_SVPTN_B4	Public variable	BOOL	CASIN_T2 pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

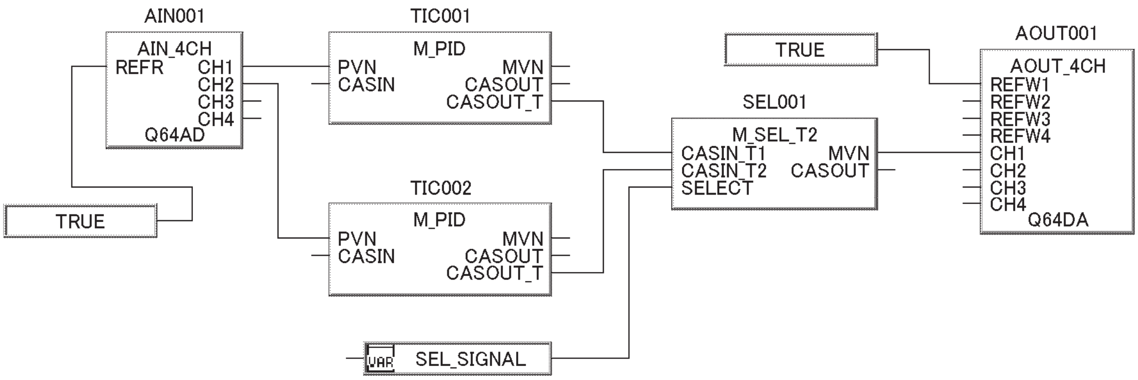
Item	Main structure member tag access FB	Reference
SEL function	P_SEL_T2	Section 8.2.28
MCHG function	P_MCHG	Section 8.3.1

Error

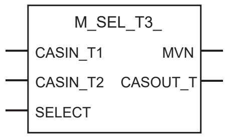
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.34 Loop Selector (With Tracking from secondary loop to primary loop) (M_SEL_T3_)

FB	FBD parts
M_SEL_T3_	

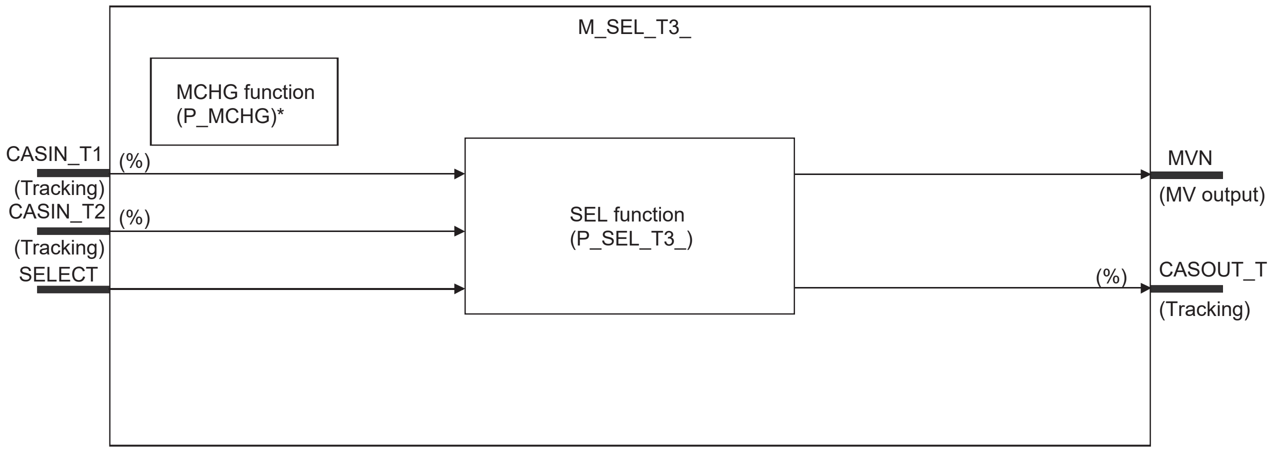
Corresponding tag type
SEL

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute loop selector taking function of P_SEL_T3_ as a single FB.

Function/FB classification name: Tag access FB_I/O control operation FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	CASIN_T1	Input variable	ADR_REAL	PV input (Unit: %) (With tracking)	0 to 100
	CASIN_T2	Input variable	ADR_REAL	PV input (Unit: %) (With tracking)	0 to 100
	SELECT	Input variable	BOOL	Selection signal (TRUE: CASIN_T2, FALSE: CASIN_T1)	TRUE, FALSE
Output	MVN	Output variable	REAL	MV output to output module FB	SEL_NMIN to SEL_NMAX
	CASOUT_T	Output variable	ADR_REAL	Cascade output (Unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	SEL_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	SEL_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	SEL_TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SEL_SVPTN_B1	Public variable	BOOL	CASIN_T1 used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SEL_SVPTN_B2	Public variable	BOOL	CASIN_T2 used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SEL_SVPTN_B3	Public variable	BOOL	CASIN_T1 pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	SEL_SVPTN_B4	Public variable	BOOL	CASIN_T2 pattern TRUE: Not upper MV FALSE: Upper MV	TRUE, FALSE	TRUE	User
	SEL_SVPTN_B5	Public variable	BOOL	Tracking to Non-selected loop (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

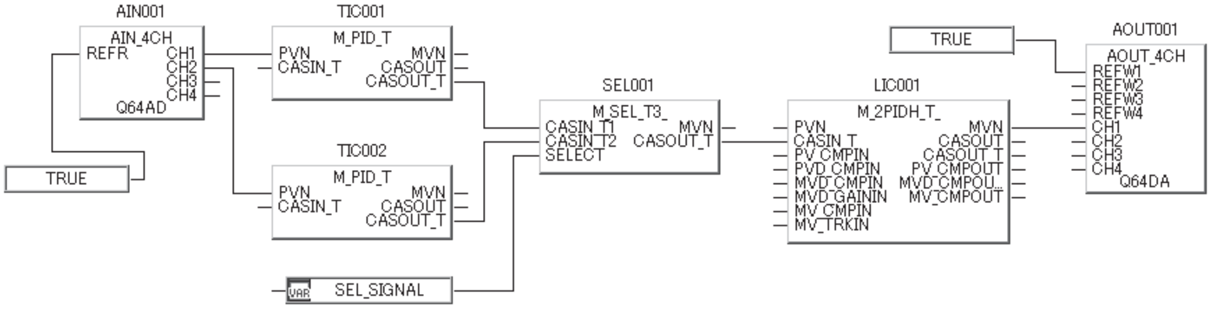
Item	Main structure member tag access FB	Reference
SEL function	P_SEL_T3_	Section 8.2.29
MCHG function	P_MCHG	Section 8.3.1

Error

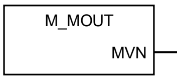
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.35 Manual Output (M_MOUT)

FB	FBD parts
M_MOUT	

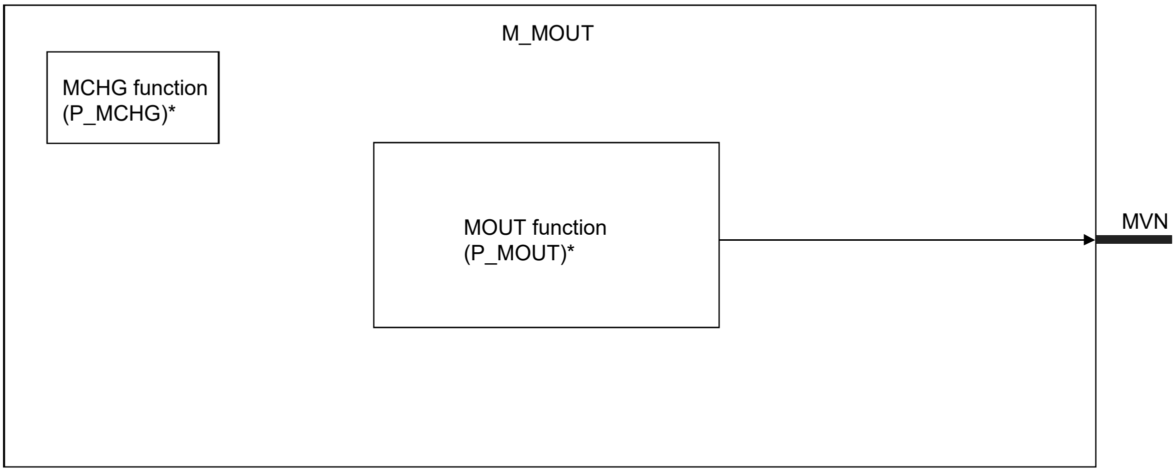
Corresponding tag type
MOUT

Control mode				
MAN	AUT	CAS	CMV	CSV
○	—	—	○	—

Function overview: Execute manual output taking function of P_MOUT as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Description	Range
Output	MVN	Output variable	REAL	MV output to output module FB	MOUT_NMIN to MOUT_NMAX

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	MOUT_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	MOUT_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

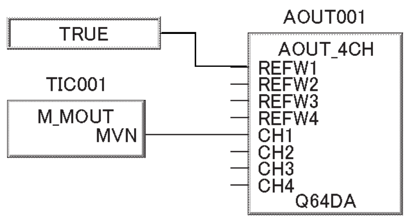
Item	Main structure member tag access FB	Reference
MOUT function	P_MOUT	Section 8.1.5
MCHG function	P_MCHG	Section 8.3.1

Error

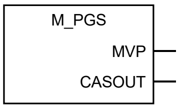
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.36 Program Setter (M_PGS)

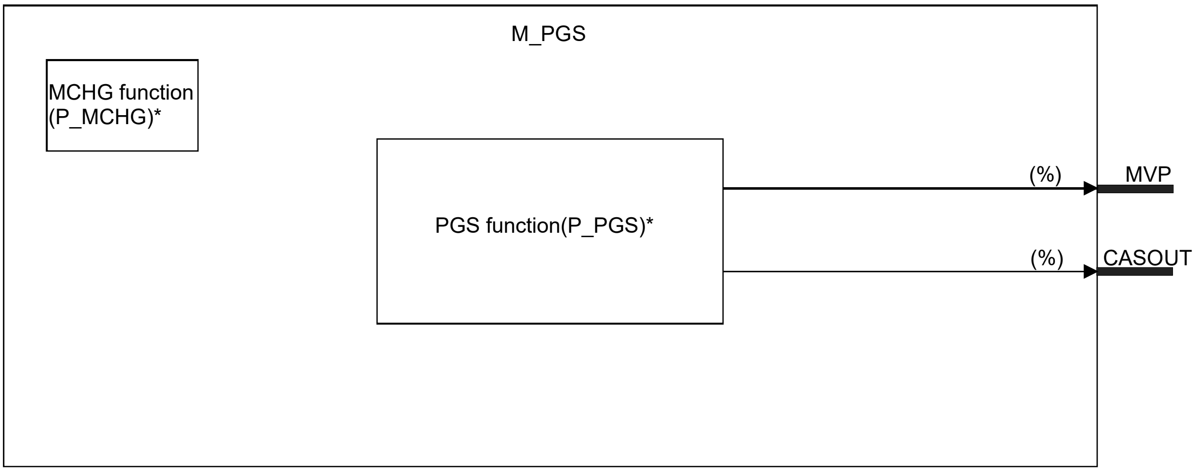
FB	FBD parts
M_PGS	

Corresponding tag type				
PGS				
Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute program setter operation taking function of P_PGS as a single FB.

Function/FB classification name: Tag FB _ loop tag FB

Block Diagram



* (P_ □) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Output	MVP	Output variable	REAL	MV output (Unit: %)	0 to 100
	CASOUT	Output variable	REAL	Cascade output (Unit: %)	0 to 100

Public Variable (Others)(*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
PGS function	P_PGS	Section 8.2.24
MCHG function	P_MCHG	Section 8.3.1

Initial value settings on the FB property page

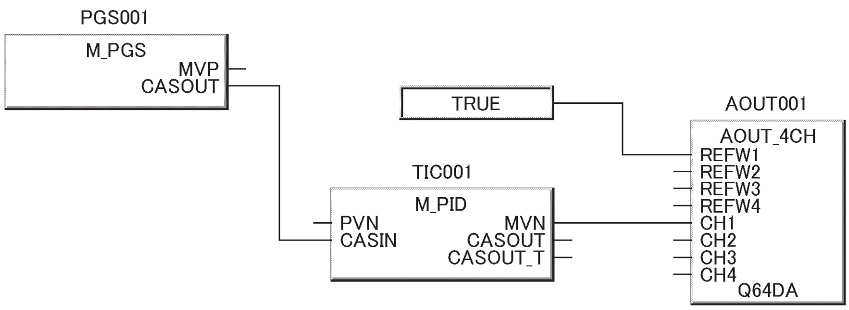
The initial values of the program setter (M_PGS) can easily be set on the FB property page of the PX Developer programming tool.

Error

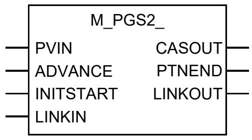
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.37 Multi-Point Program Setter (M_PGS2_)

FB	FBD parts
M_PGS2_	

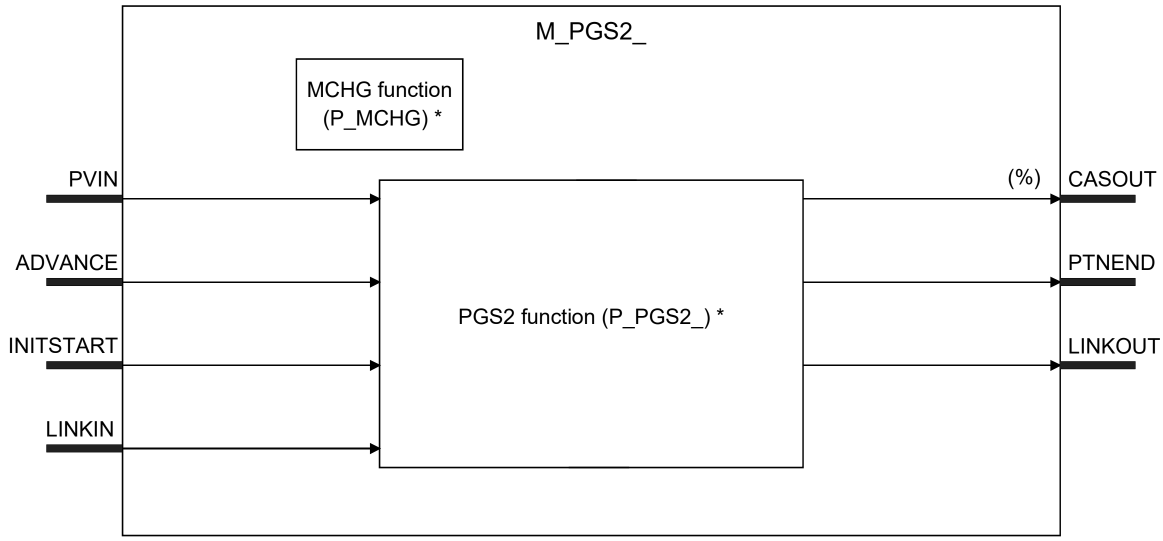
Corresponding tag type
PGS2

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	—	—	—

Function overview: Set the program using the function of P_PGS2_ as a single FB.

Function/FB classification name: Tag FB_Loop tag FB

Block Diagram



* (P_□) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVIN	Input variable	REAL	Process input (Engineering value)	-32768 to 32767
	ADVANCE	Input variable	BOOL	Advance command	TRUE, FALSE
	INITSTART	Input variable	BOOL	Initial start command	TRUE, FALSE
	LINKIN	Input variable	ADR_REAL	Link input	—
Output	CASOUT	Output variable	REAL	Cascade output (unit: %)	0 to 100
	PTNEND	Output variable	BOOL	Pattern end output	TRUE, FALSE
	LINKOUT	Output variable	ADR_REAL	Link output	—

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	PGS2_PVSTARTNO	Public variable	INT	PV start search start step	1 to 32	1	User
	PGS2_PVENDNO	Public variable	INT	PV start search end step	1 to 32	32	User
	PGS2_PRIMARY	Public variable	BOOL	Lead FB specified (TRUE: Lead, FALSE: Following)	TRUE, FALSE	TRUE	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
P_PGS2_ processing	PGS2_TCNT	Public variable	INT	Second counter for minute mode	0 to 59	0	System
	PGS2_TMCNT	Public variable	INT	Millisecond counter for second mode	0 to 999	0	System
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT)	1 to 2	0	User
	E	Public variable	BOOL	Change request (TRUE: Execution, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the control mode change processing.

Tag data

For details about the tag data that is read/written by this tag access FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
PGS2 function	P_PGS2_	Section 8.2.25
MCHG function	P_MCHG	Section 8.3.1

Initial value settings on the FB Property Page

The initial value setting of the Multi-point program setter (M_PGS2_) can easily be set on the FB Property Page of the PX Developer Programming tool.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

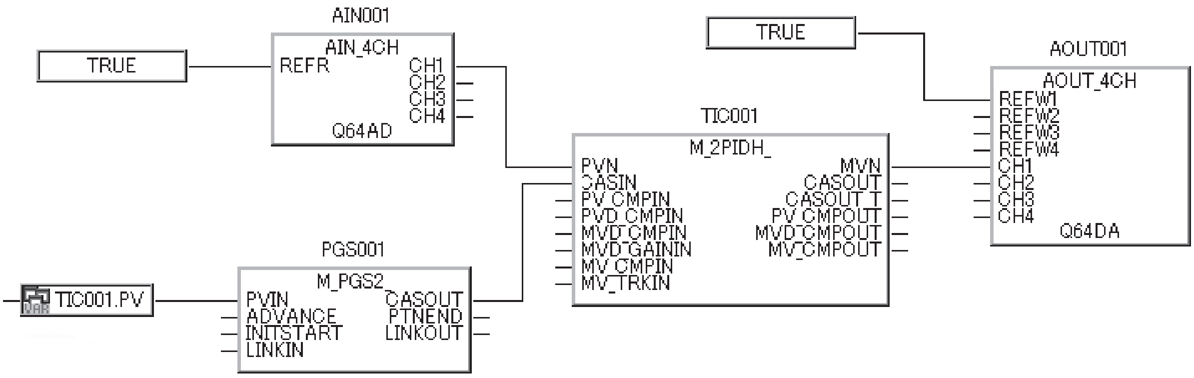
- When an overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

(1) When using M_PGS2_ output by itself

- Precautions on settings
  Set the following items.

Variable type/Pin	Variable name	Contents	Setting/connection method
Public variable	PGS2_PRIMARY	Lead FB specified	TRUE
Input pin	LINKIN	Link input	Not connected

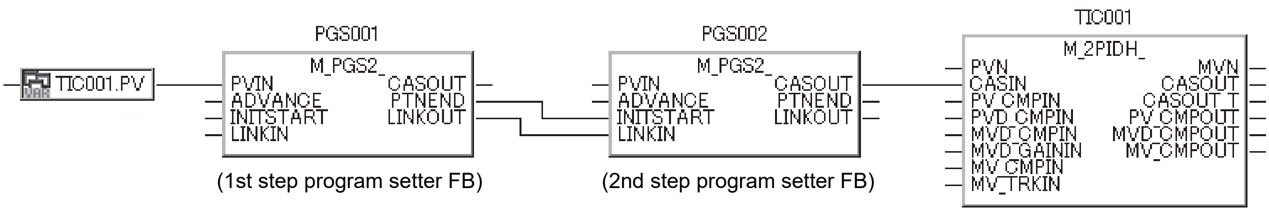


(2) When using M_PGS2_ concatenated (When using a program with over 32 steps)

- Precautions on settings
  Set the following items.

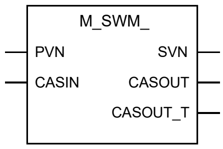
Target FB	Variable type /Pin	Variable name	Contents	Setting/connection method
Lead FB	Public variable	PGS2_PRIMARY	Lead FB specified	TRUE
	Input pin	LINKIN	Link input	Not connected
	Output pin	PTNEND	Pattern end output	Connected with INITSTART of following FB
		LINKOUT	Link output	Connected with LINKIN of following FB
Following FB	Public variable	PGS2_PRIMARY	Lead FB specified	FALSE
	Input pin	INITSTART	Initial start command	Connected with PTNEND of lead FB
		LINKIN	Link input	Connected with LINKOUT of lead FB
	Output pin	PTNEND	Pattern end output	Connected with INITSTART of following FB
		LINKOUT	Link output	Connected with LINKIN of following FB
Last FB	Public variable	PGS2_PRIMARY	Lead FB specified	FALSE
	Input pin	INITSTART	Initial start command	Connected with PTNEND of lead FB
		LINKIN	Link input	Connected with LINKOUT of lead FB
	Output pin	CASOUT	Cascade output	Connected with CASIN of tag FB such as following PID

- Operation description
  Starts PGS001 in AUT mode, and turns PTNEND output pin ON for 1 cycle at completion.
  When PGS002 receives PTNEND output, the mode is changed to AUT mode and the control is transferred.
  For details, refer to ‘FB multi-link function’ in Section 8.2.25.



(1st step program setter FB) (2nd step program setter FB)

9.1.38 Manual Setter With Monitor (M_SWM_)

FB	FBD parts
M_SWM_	

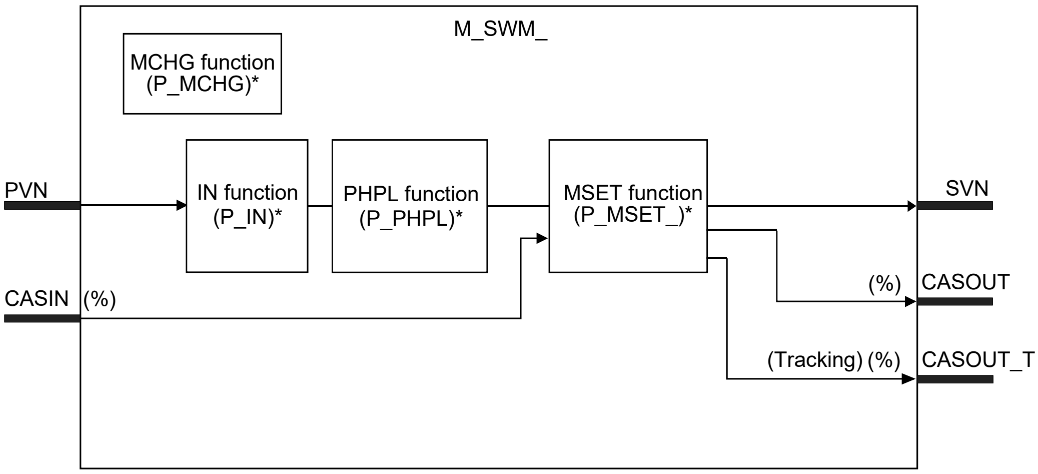
Corresponding tag type
SWM

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	–	○

Function overview: Execute manual setting with monitor taking function of P_IN+P_PHPL+P_MSET_ as function of a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_□) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Cascade SV input (unit: %)	0 to 100
Output	SVN	Output variable	REAL	SV output to module FB	-999999 to 999999
	CASOUT	Output variable	REAL	Cascade SV output (unit: %)	0 to 100
	CASOUT_T	Output variable	ADR_REAL	Cascade SV output (unit: %) (With tracking)	0 to 100

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	MSET_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	MSET_SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User
	MSET_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	MSET_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 5: CSV)	1 to 3,5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execution, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
MSET function	P_MSET_	Section 8.1.9
MCHG function	P_MCHG	Section 8.3.1

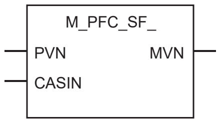
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example

9.1.39 Predictive Functional Control (Simple First Order Lag) (M_PFC_SF_)

FB	FBD parts
M_PFC_SF_	

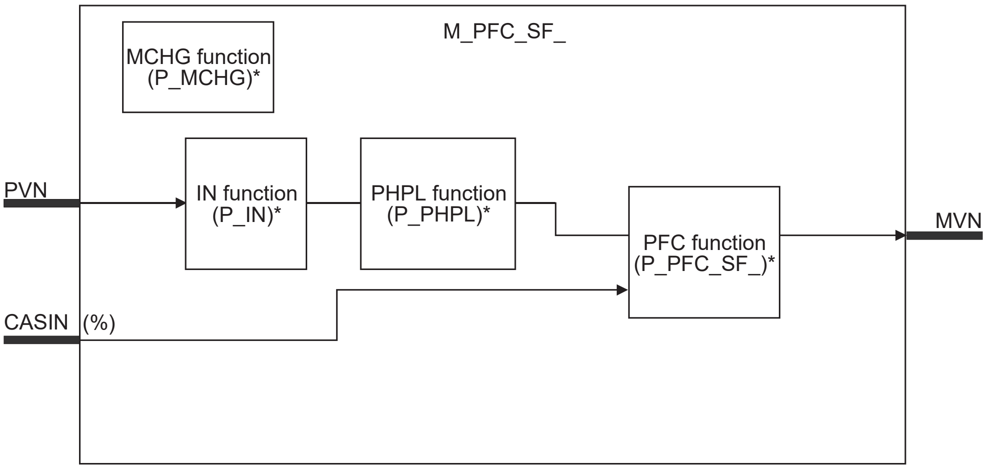
Corresponding tag type
PFC_SF

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute predictive functional controller for simple first order lag taking function of P_IN+P_PHPL+P_PFC_SF_ as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



*(P_□) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	PFC_SF_NMIN to PFC_SF_NMAX

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	110.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-10.0	User
	PFC_SF_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PFC_SF_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PFC_SF_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PFC_SF_MODEL_INIT	Public variable	BOOL	Initialize Model*1 TRUE: Initialize internal model FALSE: Do not initialize internal model	TRUE, FALSE	FALSE	User
	PFC_SF_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	PFC_SF_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	PFC_SF_SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

*1 After changing PFC parameter, set TRUE when initializing the model which is used in PFC control.
When the variable is set to TRUE, this flag turns FALSE after the initialization of internal model has been completed in the system.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execution, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the simulation processing.

*3 Indicates the control mode change processing.

POINT

- After changing the value of control cycle (CT) or changing the value of Dead time (DM), Gain (KM), or Time contrast (TM) significantly, initialize the model by turning Initialize Model (PFC_SF_MODEL_INIT) TRUE from FALSE.
  Note that, however, if the model initialization is performed during the process is not stable (MV is fluctuating), the control may not stable until the dead time is passed.
- When initializing a predictive functional control FB used in the project created with PX Developer version 1.34L or earlier after opening the project using PX Developer 1.42U or later, compilation (cold-start compile/hot-start compile/compile (online change)) is required.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

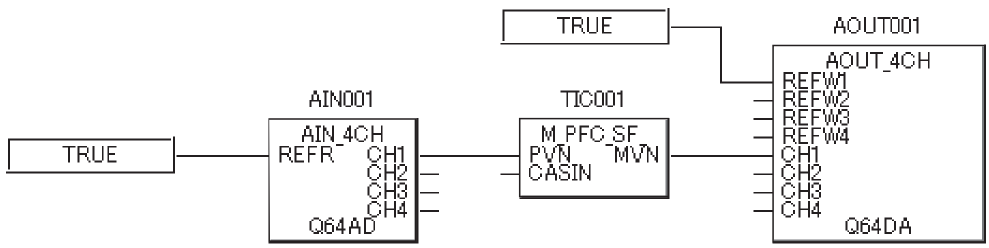
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PFC function	P_PFC_SF_	Section 8.2.30
MCHG function	P_MCHG	Section 8.3.1

Error

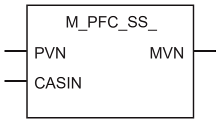
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.40 Predictive Functional Control (Simple Second Order Lag) (M_PFC_SS_)

FB	FBD parts
M_PFC_SS_	

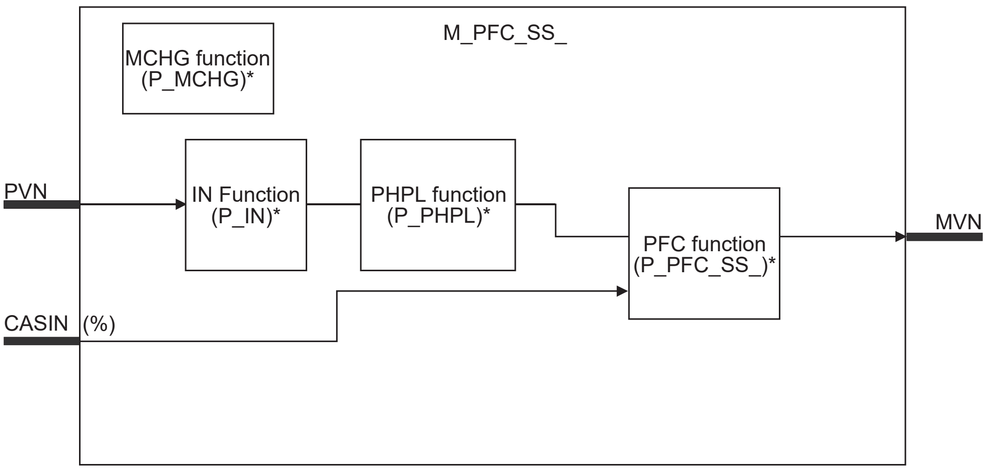
Corresponding tag type
PFC_SS

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute predictive functional controller for simple second order lag taking function of P_IN+P_PHPL+P_PFC_SS_ as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



*(P_☐) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	PFC_SS_NMIN to PFC_SS_NMAX

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	110.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-10.0	User
	PFC_SS_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PFC_SS_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PFC_SS_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PFC_SS_MODEL_INIT	Public variable	BOOL	Initialize Model*1 TRUE: Initialize internal model FALSE: Do not initialize internal model	TRUE, FALSE	FALSE	User
	PFC_SS_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	PFC_SS_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	PFC_SS_SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

*1 After changing PFC parameter, set TRUE when initializing the model which is used in PFC control.
When the variable is set to TRUE, this flag turns FALSE after the initialization of internal model has been completed in the system.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	NMIN to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execution, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the simulation processing.

*3 Indicates the control mode change processing.

POINT

- After changing the value of control cycle (CT) or changing the value of Dead time (DM), Gain (KM), Time contrast (TM1), or Time contract (TM2) significantly, initialize the model by turning Initialize Model (PFC_SS_MODEL_INIT) TRUE from FALSE.
  Note that, however, if the model initialization is performed during the process is not stable (MV is fluctuating), the control may not stable until the dead time is passed.
- When initializing a predictive functional control FB used in the project created with PX Developer version 1.34L or earlier after opening the project using PX Developer 1.42U or later, compilation (cold-start compile/hot-start compile/compile (online change)) is required.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

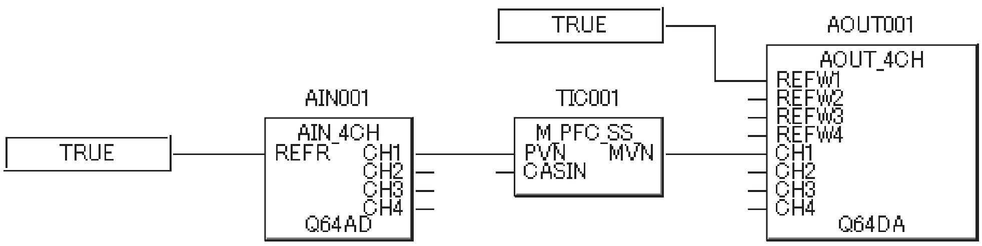
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PFC function	P_PFC_SS_	Section 8.2.31
MCHG function	P_MCHG	Section 8.3.1

Error

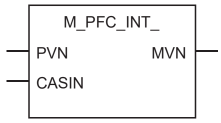
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.41 Predictive Functional Control (Integral Process) (M_PFC_INT_)

FB	FBD parts
M_PFC_INT_	

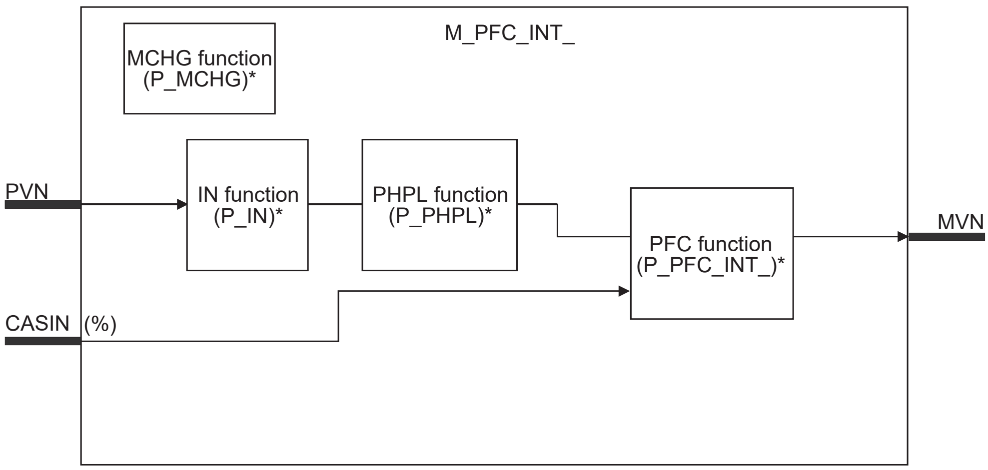
Corresponding tag type
PFC_INT

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Execute predictive functional controller for integral process taking function of P_IN+P_PHPL+P_PFC_INT_ as a single FB.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_ ☐) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Module FB input	-999999 to 999999
	CASIN	Input variable	REAL	Primary loop SV input (unit: %)	0 to 100
Output	MVN	Output variable	REAL	Module FB output	(2 × PFC_INT_NMIN – PFC_INT_NMAX) to PFC_INT_NMAX

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	110.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-10.0	User
	PFC_INT_DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PFC_INT_PN	Public variable	INT	Reverse action and direct action (0: Reverse action, 1: Direct action)	0 to 1	0	User
	PFC_INT_SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	PFC_INT_MODEL_INIT	Public variable	BOOL	Initialize Model*1 TRUE: Initialize internal model FALSE: Do not initialize internal model	TRUE, FALSE	FALSE	User
	PFC_INT_NMAX	Public variable	REAL	Output conversion high limit	-999999 to 999999	100.0	User
	PFC_INT_NMIN	Public variable	REAL	Output conversion low limit	-999999 to 999999	0.0	User
	PFC_INT_SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

*1 After changing PFC parameter, set TRUE when initializing the model which is used in PFC control.
When the variable is set to TRUE, this flag turns FALSE after the initialization of internal model has been completed in the system.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	NMIN to NMAX	0.0	User
	SIMOUT	Public variable	REAL	Simulation output	(2 × NMIN – NMAX) to NMAX	0.0	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	0	User
	E	Public variable	BOOL	Change request (TRUE: Execution, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the simulation processing.

*3 Indicates the control mode change processing.

POINT

- After changing the value of control cycle (CT) or changing the value of Dead time (DM), Gain (KM) significantly, initialize the model by turning Initialize Model (PFC_INT_MODEL_INIT) TRUE from FALSE.
  Note that, however, if the model initialization is performed during the process is not stable (MV is not 0% but output), the control may not stable until the dead time is passed.
- When initializing a predictive functional control FB used in the project created with PX Developer version 1.34L or earlier after opening the project using PX Developer 1.42U or later, compilation (cold-start compile/hot-start compile/compile (online change)) is required.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

This tag FB consists of the following tag access FB.

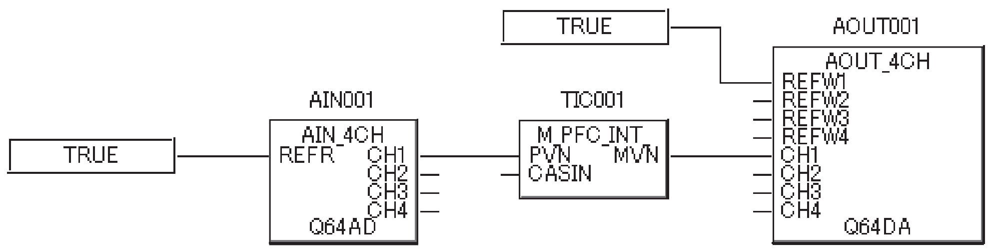
Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function	P_PHPL	Section 8.2.19
PFC function	P_PFC_INT_	Section 8.2.32
MCHG function	P_MCHG	Section 8.3.1

Error

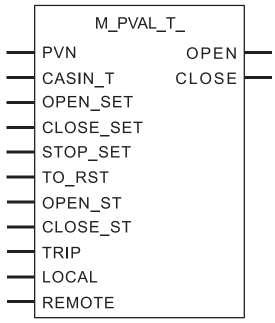
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.42 Position Proportional Output (M_PVAL_T_)

FB	FBD parts
M_PVAL_T_	

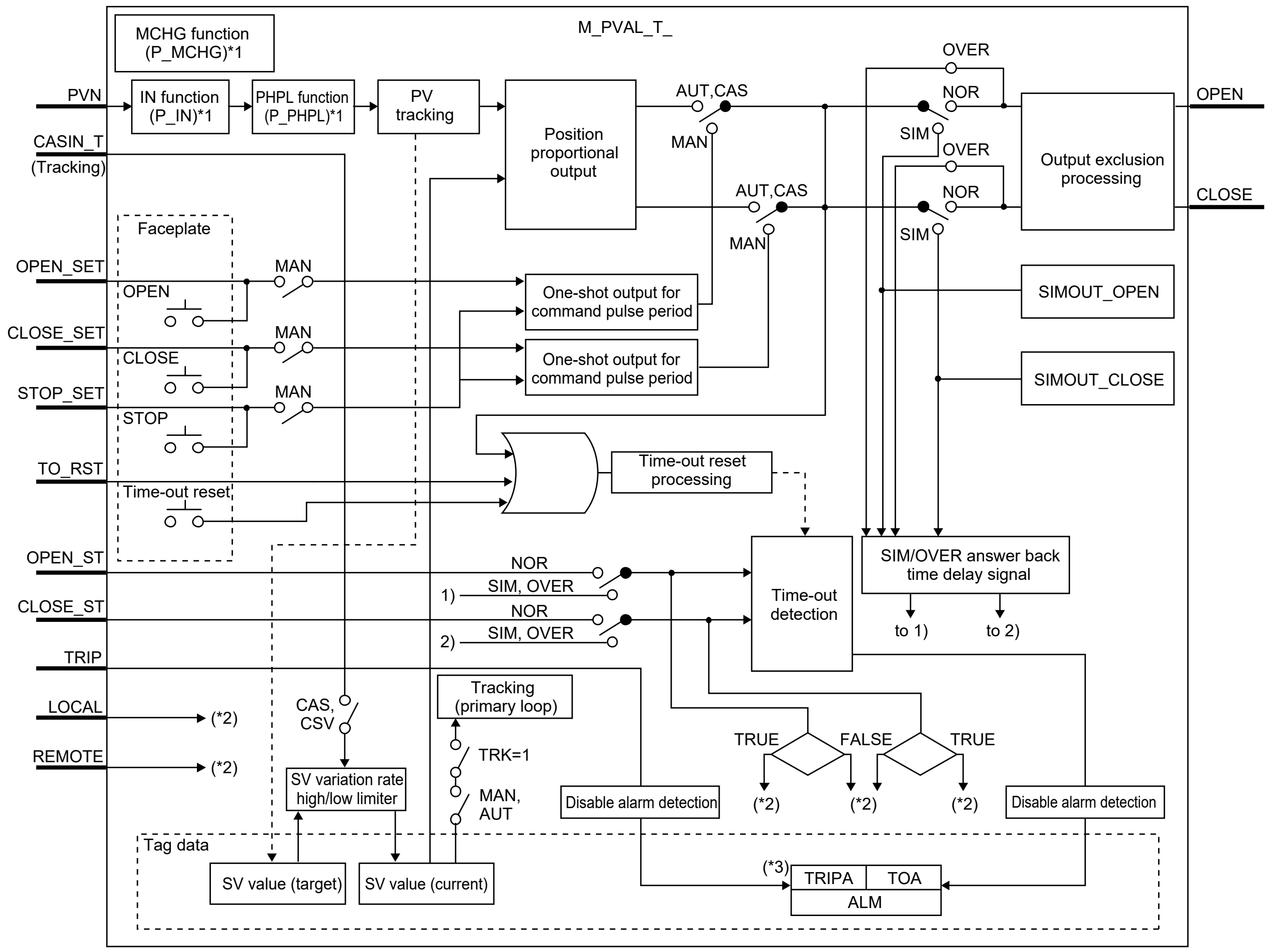
Corresponding tag type
PVAL

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	–	○

Function overview: Output OPEN/CLOSE command bits according to deviation between valve opening feedback and setting value.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



*1 (P_☐) indicates the main structure member tag access FB which is the main component of each function.
*2 For displaying the faceplate status.
*3 Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	PVN	Input variable	REAL	Valve opening feedback input	-999999 to 999999
	CASIN_T	Input variable	ADR_REAL	Primary loop SV input (unit: %) (With tracking)	0 to 100
	OPEN_SET	Input variable	BOOL	External input of OPEN operation (FALSE→TRUE: OPEN)	TRUE, FALSE
	CLOSE_SET	Input variable	BOOL	External input of CLOSE operation (FALSE→TRUE: CLOSE)	TRUE, FALSE
	STOP_SET	Input variable	BOOL	External input of STOP operation (FALSE→TRUE: set OPEN and CLOSE to FALSE)	TRUE, FALSE
	TO_RST	Input variable	BOOL	Time-out error external reset input (FALSE→TRUE: Time-out reset)	TRUE, FALSE
	OPEN_ST	Input variable	BOOL	Open status answer input (TRUE: output OPEN, FALSE: -)	TRUE, FALSE
	CLOSE_ST	Input variable	BOOL	Close status answer input (TRUE: output CLOSE, FALSE: -)	TRUE, FALSE
	TRIP	Input variable	BOOL	External failure (TRIP) input (TRUE: Occurred, FALSE: Recovered)	TRUE, FALSE
	LOCAL	Input variable	BOOL	Local operation selection signal (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE
	REMOTE	Input variable	BOOL	Remote operation selection signal (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE
Output	OPEN	Output variable	BOOL	Open command signal (OPEN) ON output (TRUE: Run, FALSE: -)	TRUE, FALSE
	CLOSE	Output variable	BOOL	Close command signal (CLOSE) ON output (TRUE: Run, FALSE: -)	TRUE, FALSE

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	IN_NMAX	Public variable	REAL	Input high limit	-999999 to 999999	100.0	User
	IN_NMIN	Public variable	REAL	Input low limit	-999999 to 999999	0.0	User
	IN_HH	Public variable	REAL	High limit range error	-999999 to 999999	102.0	User
	IN_H	Public variable	REAL	High limit range error reset	-999999 to 999999	100.0	User
	IN_L	Public variable	REAL	Low limit range error reset	-999999 to 999999	0.0	User
	IN_LL	Public variable	REAL	Low limit range error	-999999 to 999999	-2.0	User
	DVLS	Public variable	REAL	Large deviation alarm hysteresis	0 to 100	2.0	User
	PVTRK_EN	Public variable	BOOL	PV tracking execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	LMTOUT_EN	Public variable	BOOL	Output at high or low limit of valve opening execution condition TRUE : Output when PV = 0, 100% FALSE : Not output when PV = 0, 100%	TRUE, FALSE	FALSE	User
	TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern TRUE : Not upper MV FALSE : Upper MV	TRUE, FALSE	TRUE	User
	SVLMT_EN	Public variable	BOOL	SV high/low limiter (TRUE: Execute, FALSE: Not execute)	TRUE, FALSE	FALSE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
SIM processing*2	SIMIN	Public variable	REAL	Simulation input	0 to 100	0.0	User
	SIMOUT_OPEN	Public variable	BOOL	Simulation output of open command signal	TRUE, FALSE	FALSE	System
	SIMOUT_CLOSE	Public variable	BOOL	Simulation output of close command signal	TRUE, FALSE	FALSE	System
MCHG processing*3	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 5: CSV)	1, 2, 3, 5	1	User
	E	Public variable	BOOL	Change request (TRUE: Execution, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the simulation processing.

*3 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

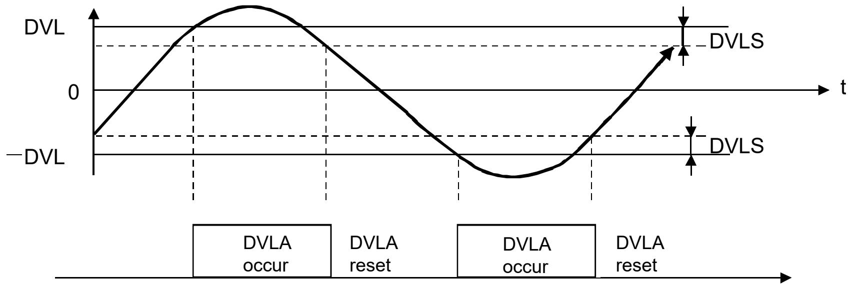
Function of components

This tag FB includes the following tag access FBs as components.

Item	Main structure member tag access FB	Reference
IN function	P_IN	Section 8.1.1
PHPL function *1	P_PHPL	Section 8.2.19
MCHG function	P_MCHG	Section 8.3.1

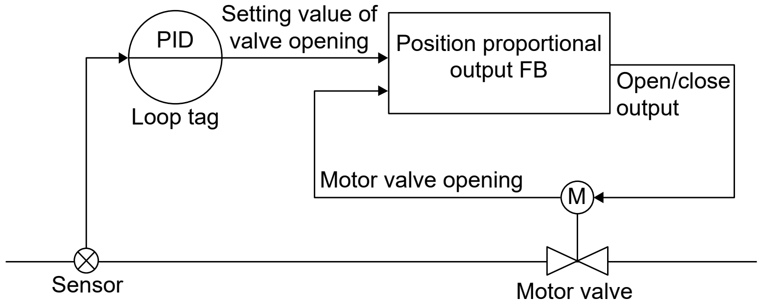
*1 Process engineering value high limit/low limit with fixed percentage of 100%, 0% respectively.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Deviation check</td><td>Execute deviation check processing


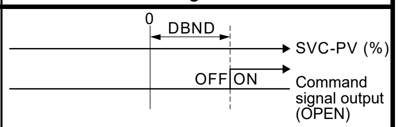
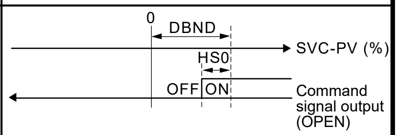
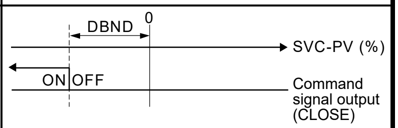
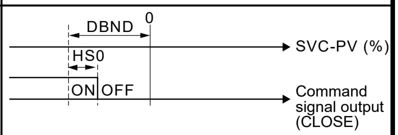
<table>
<tr><th rowspan="2">Condition</th><th>Alarm (ALM)</th></tr>
<tr><th>Large deviation (DVLA)</th></tr>
<tr><td>DVL<|DV|</td><td>TRUE (occur)</td></tr>
<tr><td>|DV|≦ (DVL−DVLS)</td><td>FALSE (reset)</td></tr>
</table>
DV: Deviation (%), DVLS: Large deviation alarm hysteresis (%), DVL: Deviation limit value (%)</td></tr>
<tr><td>PV tracking function</td><td>To avoid the sudden MV change in mode switching (MAN → AUT/CAS/CSV), matches SV value (target) with PV value when control mode is MAN and maintains the accordance.
<table>
<tr><th>Condition</th><th>PV tracking function</th></tr>
<tr><td>PVTRK_EN=TRUE and control mode = "MAN"</td><td>SV value (target) = PV value</td></tr>
<tr><td>PVTRK_EN=FALSE or control mode ≠ "MAN"</td><td>No processing</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Position proportional output</td><td>
When executing a control with motor valve, position proportional output FB is applied in combination with loop tag FB as shown below.

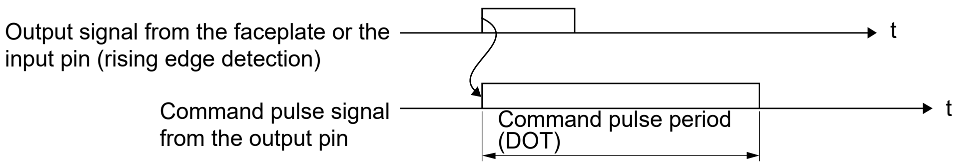


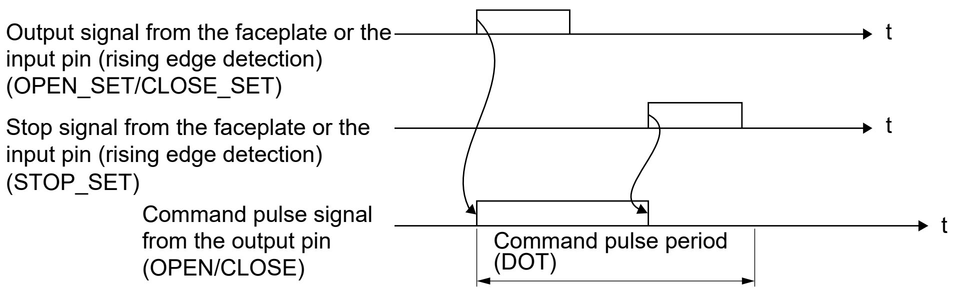
Output open/close command bits which operates the motor valve opening according to deviation (deviation of valve opening) between motor valve opening (PV) and setting value of valve opening (current) (SVC). ON/OFF the command signal in accordance with the ON/OFF condition of command signal output described below.

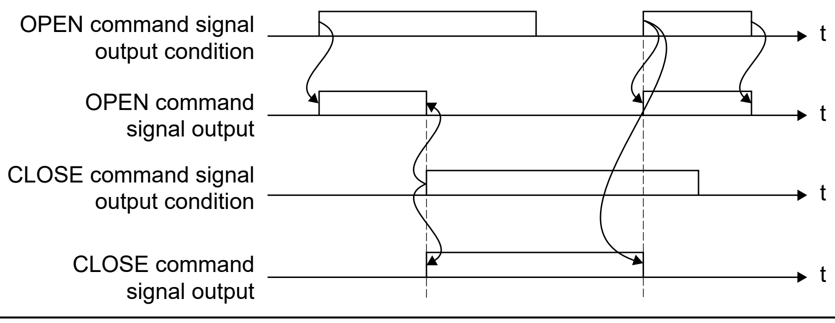
• ON/OFF condition of command signal output
The ON/OFF condition of output is calculated as shown below diagram.

<table>
<tr><th>Output</th><th>Condition</th><th>Results</th><th>Diagram</th></tr>
<tr><td rowspan="2">Output of open command signal (OPEN)</td><td>SVC – PV > DBND</td><td>Output of open command signal (OPEN)
OFF→ON</td><td>
</td></tr>
<tr><td>SVC – PV ≦ DBND – HS0</td><td>Output of open command signal (OPEN)
ON→OFF</td><td>
</td></tr>
<tr><td rowspan="2">Output of close command signal (CLOSE)</td><td>SVC – PV < –DBND</td><td>Output of close command signal (CLOSE)
OFF→ON</td><td>
</td></tr>
<tr><td>SVC – PV ≧ –(DBND – HS0)</td><td>Output of close command signal (CLOSE)
ON→OFF</td><td>
</td></tr>
</table>

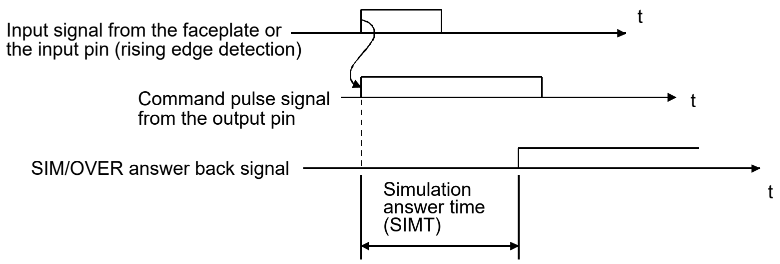
Applying dead band and hysteresis avoids switching ON/OFF of command signal output frequently.
Dead band (DBND)
When motor valve opening (PV) and setting value of valve opening (current) (SVC) change frequently, "ON/OFF" of command signal output switches frequently.
Set dead band (DBND) to avoid frequent switch of command signal output.
DBND: deviation of valve opening which starts open/close command signal output (Avoid command signal output until the deviation of valve opening is more than DBND.)
Hysteresis (HS0)
Applying hysteresis enables motor valve to operate for the amount of valve opening set in hysteresis even though deviation of valve opening is below dead band.
DBND–HS0: Command signal output is stopped when the deviation of valve opening is within DBND–HS0 range during command signal output.
</td></tr>
<tr><td>Output at high or low limit of valve opening</td><td>When PV (motor valve opening) is 0% or less, or 100% or more during command signal output, whether to output the result of position proportional output operation as it is or switch the output to OFF forcibly can be selected from Output at High Limit of Valve Opening Execution condition (LMTOUT_EN).
TRUE : when PV=0, 100%, output the result of position proportional output operation as it is.
FALSE : when PV=0%, switch CLOSE to FALSE, when PV=100%, switch OPEN to FALSE.</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>One-shot for command pulse period</td><td>
Execute the one-shot output for command pulse period according to operation from faceplate or input from input variable (OPEN_SET, CLOSE_SET)
(1) In case of operation from faceplate or the input variable (OPEN_SET) transforms from FALSE to TRUE, command pulse signal (TRUE) will be output from the output variable OPEN for the time set by command pulse period (DOT).
(2) In case of operation from faceplate or the input variable (CLOSE_SET) transforms from FALSE to TRUE, command pulse signal (TRUE) will be output from the output variable CLOSE for the time set by command pulse period (DOT).


(3) In case of stop operation from faceplate or the input variable (STOP_SET) transforms from FALSE to TRUE, switch command pulse signal being output from output variable OPEN or CLOSE to OFF.


</td></tr>
<tr><td>Output exclusion processing</td><td>
When both OPEN command output condition and CLOSE command output condition are satisfied, only a command output which satisfies the output condition later is TRUE. (The command output which output before is FALSE.)
<When command signal output condition for output variable OPEN or output variable CLOSE is TRUE in the same period>


</td></tr>
<tr><td>Operation location input</td><td>When TRUE is input to LOCAL pin, position proportional output operation is not executed, and the output from FB output pin OPEN and CLOSE is FALSE. When LOCAL pin input is switched FALSE to TRUE, the operation mode is MAN mode.</td></tr>
<tr><td>Tracking processing</td><td>
Whether execute tracking processing to the input variable CASIN_T is described in the following table:
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SVPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>SV variation rate high/low limiter</td><td>
Checks variation rate high/low limiter to setting value of valve opening (target) (SV) in execution cycle.
(1) Variation rate limiter
SV variation rate high limit value inputted in % is processed.
<table>
<tr><th>Condition</th><th>Variation rate limiter result</th><th>Target variation rate limit (DSVLA) of alarm2 (ALM2)</th></tr>
<tr><td>| SV - SVC | ≦ DSVL</td><td>SV</td><td>FALSE (reset)</td></tr>
<tr><td>SV - SVC > DSVL</td><td>SVC + DSVL</td><td>TRUE (occur)</td></tr>
<tr><td>SV - SVC < - DSVL</td><td>SVC - DSVL</td><td>TRUE (occur)</td></tr>
</table>
SV: Setting value of valve opening (target), SVC: Setting value of valve opening (current)
(2) High/low limiter
• SVLMT_EN is TRUE.
<table>
<tr><th rowspan="2">Condition</th><th rowspan="2">High/low limiter result</th><th colspan="2">Alarm2 (ALM2)</th></tr>
<tr><th>Target lower limit (SVLA)</th><th>Target upper limit (SVHA)</th></tr>
<tr><td>Variation rate limiter result > SH</td><td>SH</td><td>FALSE (reset)</td><td>TRUE (occur)</td></tr>
<tr><td>Variation rate limiter result < SL</td><td>SL</td><td>TRUE (occur)</td><td>FALSE (reset)</td></tr>
<tr><td>SL ≦ variation rate limiter result ≦ SH</td><td>Variation rate limiter result</td><td>FALSE (reset)</td><td>FALSE (reset)</td></tr>
</table>
High/low limiter result is stored to setting value of valve opening (current) (SVC).
• SVLMT_EN is FALSE.
Variation rate limiter result is stored to setting value of valve opening (current) (SVC).
</td></tr>
<tr><td>Time-out detection/ time-out reset</td><td>
(1) Time-out detection
Time-out (TOA) of alarm (ALM) will occur if TRUE is not input from the status answer input (OPEN_ST/CLOSE_ST) for more than the set time of time-out timer (TOT) after command signal (TRUE) is output from output variables OPEN/CLOSE.
<table>
<tr><th rowspan="2">Condition</th><th>Alarm</th></tr>
<tr><th>Time-out (TOA)</th></tr>
<tr><td>Time up to status answer signal input≧setting period of time-out timer (TOT)</td><td>TRUE (occur)</td></tr>
<tr><td>Time up to status answer signal input<setting period of time-out timer (TOT)</td><td>FALSE (reset)</td></tr>
</table>
(2) Time-out reset
Reset (FALSE) the time-out (TOA) of alarm (ALM) by the following operations.
(a) Output command pulse signal from output variable (OPEN, CLOSE) by the operation from faceplate or input from input variable (OPEN_SET, CLOSE_SET).
(b) Input TRUE to input variable (TO_RST).
</td></tr>
<tr><td>SIM/OVER answer back time delay signal</td><td>
In case of SIMULATION Mode or OVERRIDE Mode, status answer signal is created in CPU module after command signal output.
The delay time of status answer signal is set by simulation answer time (SIMT).


</td></tr>
</table>

Item	Contents
Disable Alarm Detection	Set whether Enable Alarm Detection or not in SV variation rate high/low limiter and time-out detection. (1) Disable Alarm Detection with the settings of "Disable Alarm Detection" and "Disable Alarm Detection 2" of tag data: If the following items of Disable Alarm Detection (INH) /Disable Alarm Detection 2 (INH2) of tag data are TRUE, the TOA, TRIPA, DSVLA, SVHA, and SVLA of alarm (ALM) and alarm 2 (ALM2) will not be detected. ● TOI, TRIPI, ERRI, DSVLI, SVHI, SVLI (2) Disable Alarm Detection by loop stop processing: Please refer to loop stop processing in the following contents.

Other Function

Item	Contents
Output processing at sensor alarm occurrence	When a sensor error (SEA) occurs in P_IN of tag access FB, both OPEN and CLOSE of command signal output are switched to FALSE.
Loop stop processing	The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE. 1) Both OPEN and CLOSE output FALSE. 2) Change the control mode automatically to MANUAL. 3) Trip (TRIPA), time-out (TOA) is not reset. 4) Reset DSVLA, SVHA, and SVLA when DSVLA, SVHA, and SVLA of alarm2 (ALM2) occur. 5) Alarm is not detected in SV variation rate high/low limiter.

Processing Operation

Processing / Control mode	Deviation check	PV tracking	Position-proportional output	Output at high or low limit of valve opening	One-shot for command pulse period	Output exclusion processing	Tracking	SV variation rate limiter high/low limiter	Time-out detection/ time-out reset	Alarm
MAN	○	○	×	×	○	○	○ (*1)	○ (*2)	○	○ (*3)
AUT	○	×	○	○	×	○	○ (*1)	○	○	○ (*4)
CAS, CSV	○	×	○	○	×	○	×	○	○	○ (*4)

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

*2 When the control mode is MAN, SV variation rate limiter processing is not executed.

*3 An alarm (ALM) whose corresponding bit is TRUE (occurred) is reset, and the alarm cannot be detected.

*4 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

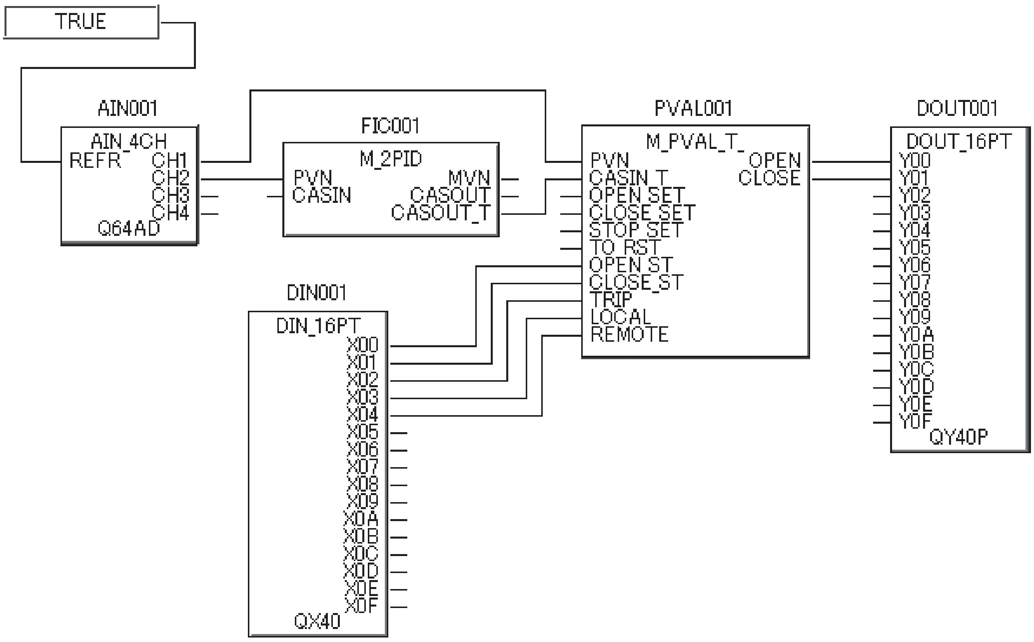
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.

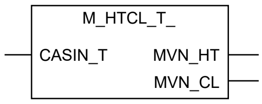
Additionally, process control error code as well as the detailed error information, will be displayed on the screen. For details of process control error code, refer to Appendix 2.

- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.1.43 Heating and Cooling Output (M_HTCL_T_)

FB	FBD parts
M_HTCL_T_	

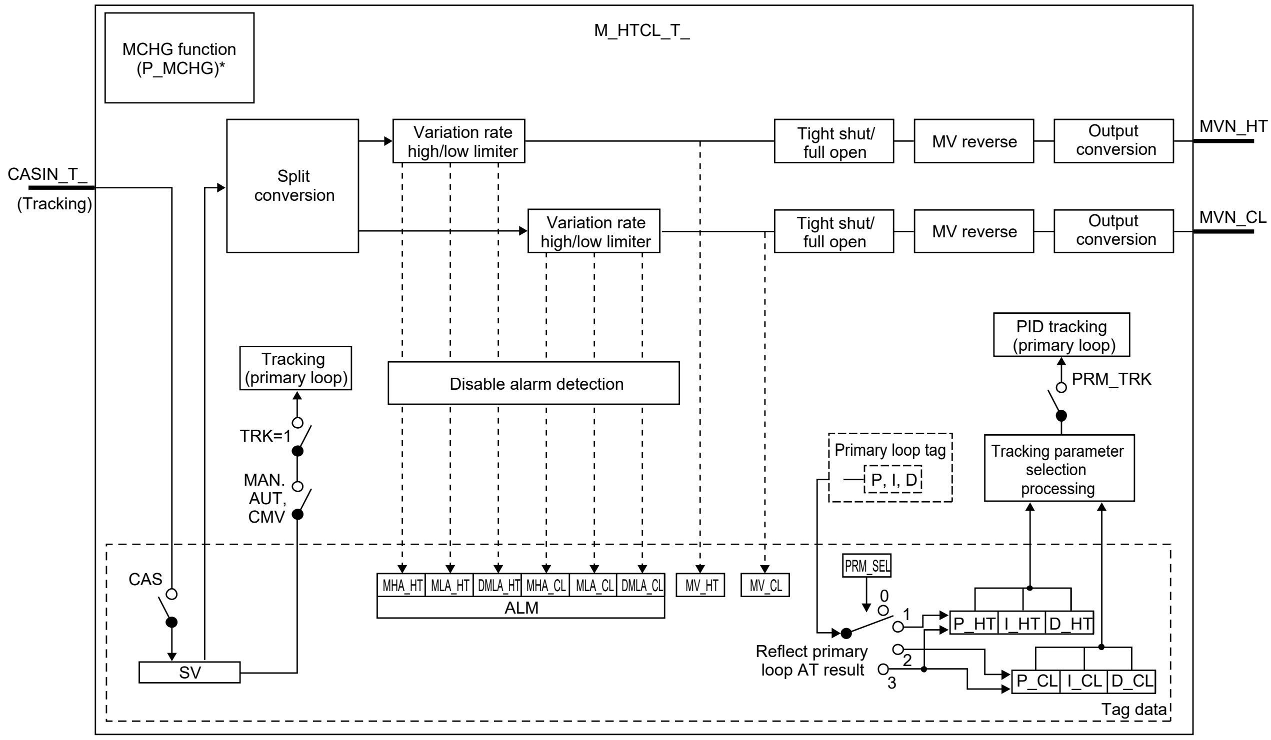
Corresponding tag type
HTCL

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	○	○	○

Function overview: Output two manipulated value after split conversion and output conversion from setting value.
Temperature control can be executed by outputting to final control elements for heating and cooling.

Function/FB classification name: Tag FB_loop tag FB

Block Diagram



* (P_☐) indicates the main structure member tag access FB which is the main component of each function.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	CASIN_T	Input variable	ADR_REAL	Cascade SV input (unit: %) (With tracking)	0 to 100
Output	MVN_HT	Output variable	REAL	Module FB output (Heat)	NMIN_HT to NMAX_HT
	MVN_CL	Output variable	REAL	Module FB output (Cool)	NMIN_CL to NMAX_CL

Public Variable (Operation Constant)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Operation processing	NMAX_HT	Public variable	REAL	Heating output conversion high limit	-999999 to 999999	100.0	User
	NMIN_HT	Public variable	REAL	Heating output conversion low limit	-999999 to 999999	0.0	User
	MVREV_HT_EN	Public variable	BOOL	Heating MV reverse execution condition (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE	FALSE	User
	NMAX_CL	Public variable	REAL	Cooling output conversion high limit	-999999 to 999999	100.0	User
	NMIN_CL	Public variable	REAL	Cooling output conversion low limit	-999999 to 999999	0.0	User
	MVREV_CL_EN	Public variable	BOOL	Cooling MV reverse execution condition (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE	FALSE	User
	FOTS_HT_EN	Public variable	BOOL	Heating tight shut/full open execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	MVFO_HT	Public variable	REAL	Heating output value for full open (unit: %)	100 to 125	112.5	User
	MVTS_HT	Public variable	REAL	Heating output value for tight shut (unit: %)	-25 to 0	-16.82	User
	FOTS_CL_EN	Public variable	BOOL	Cooling tight shut/full open execution condition (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User
	MVFO_CL	Public variable	REAL	Cooling output value for full open (unit: %)	100 to 125	112.5	User
	MVTS_CL	Public variable	REAL	Cooling output value for tight shut (unit: %)	-25 to 0	-16.82	User
	TRK(*1)	Public variable	INT	Tracking flag (0: Not execute, 1: Execute)	0 to 1	0	User
	SVPTN_B0	Public variable	BOOL	Setting value (SV) used (TRUE: Not used, FALSE: Used)	TRUE, FALSE	TRUE	User
	SVPTN_B1	Public variable	BOOL	Setting value (SV) pattern (TRUE: Not upper MV, FALSE: Upper MV)	TRUE, FALSE	TRUE	User
	HBOTIME	Public variable	DINT	Heater burnout detecting time (second) (0: Invalid, 1 to 99999999: Detecting time)	0 to 99999999	0	User
	TEMPALM_EN	Public variable	BOOL	Temperature anomaly detection execution condition (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE	FALSE	User

*1 When setting "1" (enable to execute tracking) to the tracking flag, connect the CASOUT_T of primary loop to input variable CASIN_T.

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT, 3: CAS, 4: CMV, 5: CSV)	1 to 5	1	User
	E	Public variable	BOOL	Change request (TRUE: Execution, FALSE: Stop)	TRUE, FALSE	FALSE	User

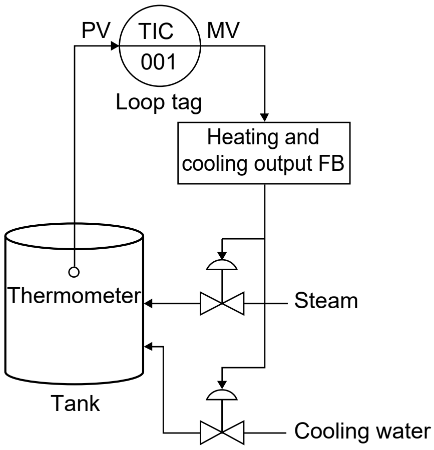
*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

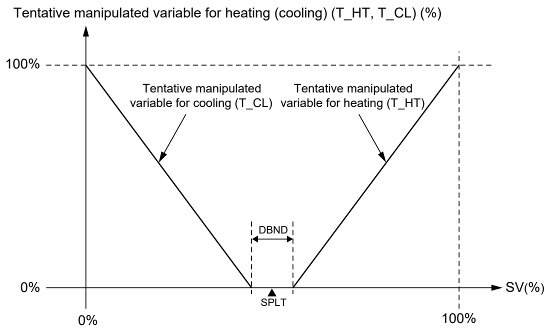
*2 Indicates the control mode change processing.

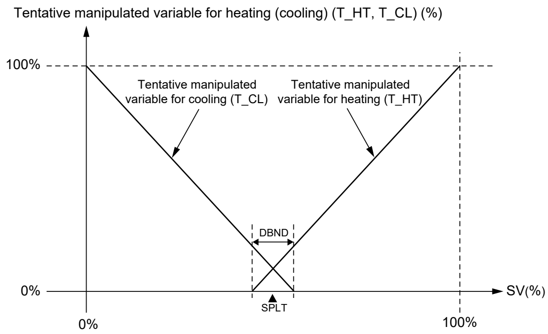
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Split conversion</td><td>
Heating and cooling output FB is applied when controlling 2 final control elements for heating and cooling with 1 loop tag FB as shown below diagram.


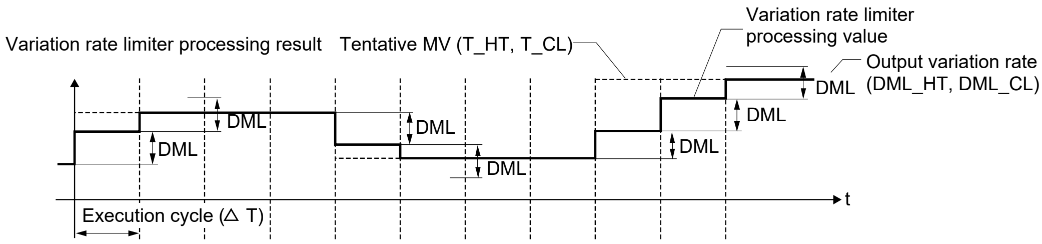
● Split conversion
Execute split conversion from setting value (SV) and calculate tentative manipulated variable (T_HT, T_CL) for heating/cooling.
Dead band (DBND) ≧ 0


Dead band (DBND) < 0


Split median (SPLT)
A value where heating output and cooling output are switched when executing split conversion.
Dead band (DBND)
When the setting value (SV) is within a half range from split median (SPLT) to value set in dead band (DBND), heating/cooling tentative manipulated variable (T_HT, T_CL) is low limit value.
When setting dead band (DBND) to negative value, and setting value (SV) exceeds a half range from split median (SPLT) to value set in dead band (DBND), heating/cooling tentative manipulated variable (T_HT, T_CL) is low limit value.
</td></tr>
</table>

Item	Contents
MV variation rate and high/low limiter	Execute variation rate limiter and high/low limit check to tentative manipulated variable (T_HT, T_CL) for heating/cooling after split conversion.

● Variation rate limiter



Heating variation rate limiter

Condition	Variation rate limiter processing result	Alarm (ALM)
		Output variation rate limit (DMLA_HT)
|T_HT - MV_HT| ≦ DML_HT	T_HT	FALSE (reset)
T_HT - MV_HT > DML_HT	MV_HT + DML_HT	TRUE (occur)
T_HT - MV_HT < -DML_HT	MV_HT - DML_HT	TRUE (occur)

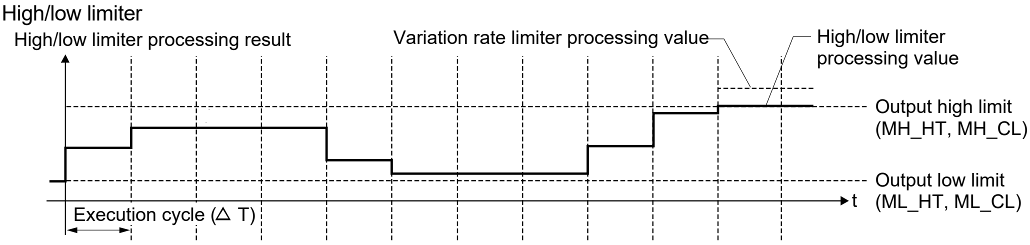
T_HT: Heating tentative MV value, MV_HT: Heating manipulated variable,
DML_HT: Heating output variation rate high limit

Cooling variation rate limiter

Condition	Variation rate limiter processing result	Alarm (ALM)
		Output variation rate limit (DMLA_CL)
|T_CL - MV_CL| ≦ DML_CL	T_CL	FALSE (reset)
T_CL - MV_CL > DML_CL	MV_CL + DML_CL	TRUE (occur)
T_CL - MV_CL < -DML_CL	MV_CL - DML_CL	TRUE (occur)

T_CL: Cooling tentative MV value, MV_CL: Cooling manipulated variable,
DML_CL: Cooling output variation rate high limit

● High/low limiter



Heating high/low limiter

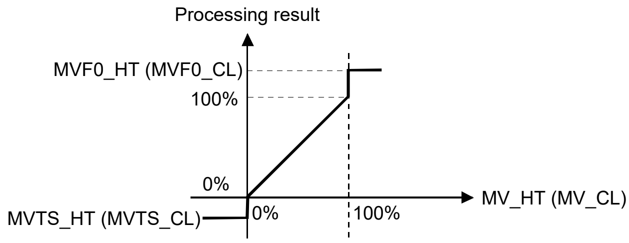
Condition	High/low limiter processing result	Alarm (ALM)	
		Output low limit (MLA_HT)	Output high limit (MHA_HT)
Heating variation rate limiter processing result > MH_HT	MH_HT	FALSE (reset)	TRUE (occur)
Heating variation rate limiter processing result < ML_HT	ML_HT	TRUE (occur)	FALSE (reset)
ML_HT ≦ Heating variation rate limiter processing result ≦ MH_HT	Heating variation rate limiter processing value	FALSE (reset)	FALSE (reset)

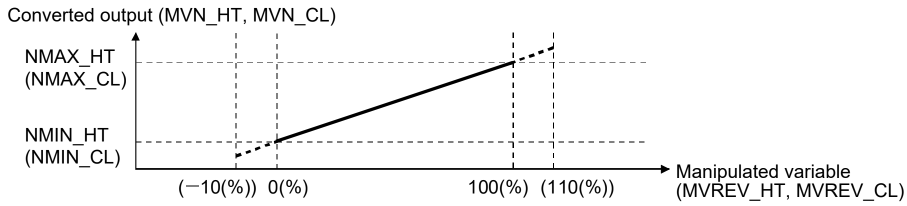
MH_HT: Heating output high limit, ML_HT: Heating output low limit

Cooling high/low limiter

Condition	High/low limiter processing result	Alarm (ALM)	
		Output low limit (MLA_CL)	Output high limit (MHA_CL)
Cooling variation rate limiter processing result > MH_CL	MH_CL	FALSE (reset)	TRUE (occur)
Cooling variation rate limiter processing result < ML_CL	ML_CL	TRUE (occur)	FALSE (reset)
ML_CL ≦ Cooling variation rate limiter processing result ≦ MH_CL	Cooling variation rate limiter processing value	FALSE (reset)	FALSE (reset)

MH_CL: Cooling output high limit, ML_CL: Cooling output low limit

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Tight shut/
full open</td><td>
Use the tight shut/full open function to open or close the control valve completely and absolutely.
Reduce the processing result to the output value for tight shut when the MV_HT or MV_CL is 0% or lower, and raise it to the output value for full open when it is 100% or higher.
● When FOTS_HT_EN/FOTS_CL_EN is TRUE


MVTS_HT: Heating output value for tight shut (%), MVFO_HT: Heating output value for full open (%)
MVTS_CL: Cooling output value for tight shut (%), MVFO_CL: Cooling output value for full open (%)
</td></tr>
<tr><td>MV
reverse</td><td>
Executes MV value inversion processing (100-MV).
● Heating MV reverse
<table>
<tr><th>Condition</th><th>Processing result</th></tr>
<tr><td>MVREV_HT_EN = TRUE</td><td>MVREV_HT = 100-MV_HT</td></tr>
<tr><td>MVREV_HT_EN = FALSE</td><td>MVREV_HT = MV_HT</td></tr>
</table>
MVREV_HT: Heating output after processing of MV reverse for internal operation (%),
MV_HT: Heating manipulated variable (%)
● Cooling MV reverse
<table>
<tr><th>Condition</th><th>Processing result</th></tr>
<tr><td>MVREV_CL_EN = TRUE</td><td>MVREV_CL = 100-MV_CL</td></tr>
<tr><td>MVREV_CL_EN = FALSE</td><td>MVREV_CL = MV_CL</td></tr>
</table>
MVREV_CL: Cooling output after processing of MV reverse for internal operation (%),
MV_CL: Cooling manipulated variable (%)
</td></tr>
<tr><td>Output
conversion</td><td>
Output conversion processing is carried out.


$$\text{Converted output (MVN\_HT)} = \left\{ (\text{NMAX\_HT} - \text{NMIN\_HT}) \times \frac{\text{MVREV\_HT}}{100} \right\} + \text{NMIN\_HT}$$

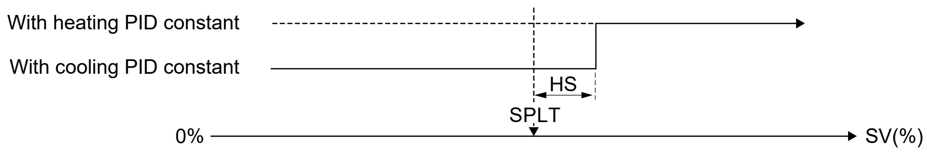
$$\text{Converted output (MVN\_CL)} = \left\{ (\text{NMAX\_CL} - \text{NMIN\_CL}) \times \frac{\text{MVREV\_CL}}{100} \right\} + \text{NMIN\_CL}$$

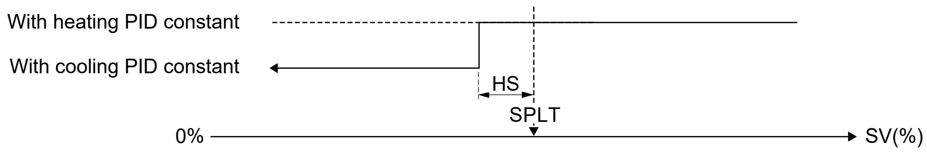
NMAX_HT: Heating output conversion high limit, NMIN_HT: Heating output conversion low limit,
MVREV_HT: Heating output after processing of MV reverse for internal operation (%),
MVN_HT: Heating converted output
NMAX_CL: Cooling output conversion high limit, NMIN_CL: Cooling output conversion low limit,
MVREV_CL: Cooling output after processing of MV reverse for internal operation (%),
MVN_CL: Cooling converted output
</td></tr>
<tr><td>Tracking
processing</td><td>
Whether execute tracking processing to the input variable CASIN_T is described in the following table:
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Result</th></tr>
<tr><th>Tracking flag (TRK)</th><th>Setting value (SV) used (SVPTN_B0)</th></tr>
<tr><td rowspan="2">1</td><td>FALSE</td><td>Input variable CASIN_T performs tracking.</td></tr>
<tr><td>TRUE</td><td rowspan="2">Input variable CASIN_T does not perform tracking.</td></tr>
<tr><td>0</td><td>FALSE or TRUE</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Auto tuning result reflection function</td><td>
Reflect the auto tuning result in primary loop to tag data of this tag FB.
The procedure when calculating heating and cooling PID parameter with auto tuning in primary loop is as follows.
1) Set the control mode of this tag to CAS mode.
2) Set heating (1) to Target to Reflect Results of Auto Tuning (PRM_SEL).
3) Execute heating auto tuning in primary loop. When auto tuning in primary loop is completed normally, the result of auto tuning is reflected to heating PID parameter of this tag FB.
4) Set cooling (2) to Target to Reflect Results of Auto Tuning (PRM_SEL).
5) Execute cooling auto tuning in primary loop. When auto tuning in primary loop is completed normally, the result of auto tuning is reflected to cooling PID parameter of this tag FB.
<table>
<tr><th colspan="2">Condition</th><th>Tag memory that stores the result of auto tuning</th></tr>
<tr><td rowspan="4">SVPTN_B0 = FALSE
and
SVPTN_B1=FALSE</td><td>PRM_SEL = 0</td><td>Not stored.</td></tr>
<tr><td>PRM_SEL = 1</td><td>Heating PID parameters (P_HT, I_HT, D_HT)</td></tr>
<tr><td>PRM_SEL = 2</td><td>Cooling PID parameters (P_CL, I_CL, D_CL)</td></tr>
<tr><td>PRM_SEL = 3</td><td>Heating/ Cooling PID parameters (P_HT, I_HT, D_HT, P_CL, I_CL, D_CL)</td></tr>
</table>
PRM_SEL: Target to reflect results of auto tuning
</td></tr>
</table>

POINT

- When using the auto tuning result reflection function, the connectable tag types as primary loop are as follows.
  PID, 2PID, 2PIDH
- When tracking PID parameter, the parameter values stored on this tag FB are not restored even though "Restore PID parameters" is executed in PX Developer Monitor Tool. Execute the following operations to restore the PID parameter to the status before executing auto tuning.
  1. Set the PID parameter tracking flag (PRM_TRK) of this tag FB to "Not execute" (0).
  2. Set the restored PID parameter of primary loop before executing auto tuning to the PID parameter of this FB.
  3. Set "Execute" (1) to the PID parameter tracking flag (PRM_TRK) of this tag FB.

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Tracking (PID parameter)</td><td>
When the following conditions of the PID parameter tracking are satisfied, track heating/cooling PID parameter to primary loop.
• PRM_TRK = 1
• SVPTN_B0 = FALSE
• SVPTN_B1 = FALSE
The following describes the description of processing.
The PID parameter to be tracked is selected with the following expressions.
<table>
<tr><th>Condition</th><th>PID parameter to be tracked</th></tr>
<tr><td>SV ≧ SPLT + HS during cooling PID parameter tracking</td><td>Heating PID parameters (P_HT, I_HT, D_HT)</td></tr>
<tr><td>SV < SPLT + HS during cooling PID parameter tracking</td><td>Cooling PID parameters (P_CL, I_CL, D_CL)</td></tr>
<tr><td>SV ≧ SPLT - HS during heating PID parameter tracking</td><td>Heating PID parameters (P_HT, I_HT, D_HT)</td></tr>
<tr><td>SV < SPLT - HS during heating PID parameter tracking</td><td>Cooling PID parameters (P_CL, I_CL, D_CL)</td></tr>
</table>
SV: Primary loop SV, SPLT: Split median, HS: Hysterisis (%)
● Switching from cooling PID parameter to heating PID parameter


● Switching from heating PID parameter to cooling PID parameter


The following table indicates the operation when the conditions for tracking PID parameters shown above are satisfied for the first time.
<table>
<tr><th>Condition</th><th>PID parameter to be tracked</th></tr>
<tr><td>SV ≧ SPLT</td><td>Heating PID parameters (P_HT, I_HT, D_HT)</td></tr>
<tr><td>SV < SPLT</td><td>Cooling PID parameters (P_CL, I_CL, D_CL)</td></tr>
</table>
</td></tr>
</table>

POINT

- When the tag type of primary loop is as follows, tracking PID parameter is executed.
  PID, 2PID, 2PIDH, PIDP, SPI*1, IPD, BPI*1
  *1 The parameters to be tracked are "P" (+52) and "I" (+56).
- When tracking PID parameter, set the initial value on the heating/cooling PID parameter of this tag FB. Do not set the PID parameter on primary loop.
- Switch timing of PID parameter can be adjusted with hysteresis.

Item	Contents
Disable Alarm Detection	Set whether enable Alarm Detection or not in variation rate high/low limiter and heater burnout detection. (1) Disable Alarm Detection with the setting of "Disable Alarm Detection" tag data: If the following items of Disable Alarm Detection (INH) of tag data are TRUE, alarm (ALM) of DMLA_HT, MHA_HT, MLA_HT, DMLA_CL, MHA_CL, MLA_CL, and HBOA will not be detected. ● ERRI, DMLI_HT, MHI_HT, MLI_HT, DMLI_CL, MHI_CL, MLI_CL, HBOI (2) Disable Alarm Detection by control mode: If the control mode is MAN and CMV, reset DMLA_HT, MHA_HT, MLA_HT, DMLA_CL, MHA_CL, MLA_CL, and HBOA of alarm (ALM) and DMLA_HT, MHA_HT, MLA_HT, DMLA_CL, MHA_CL, MLA_CL, and HBOA will not be detected. (3) Disable Alarm Detection by stop processing: Please refer to loop stop processing in the following contents.

Other Function

Item	Contents
Heater burnout detection	HBOA occurs when the status of heating manipulated variable continues being high limit value (MH_HT), and the duration is longer than that is specified in HBOTIME. Reset HBOA when a heating manipulated variable is below high limit value (MH_HT).
Temperature anomaly detection	When TEMPALM_EN is TRUE, and HHA (input high high limit alarm) occurs in primary loop, heating manipulated variable is low limit of manipulated variable (ML_HT) for heating.
Loop stop processing	The following processes are executed when either the stop alarm (SPA) of alarm (ALM) or the tag stop (TSTP) of monitor output buffer (DOM) is TRUE. 1) Hold the output (MVN_HT, MVN_CL). 2) Change the control mode automatically to MANUAL. 3) Reset DMLA_HT, MHA_HT, MLA_HT, DMLA_CL, MHA_CL, MLA_CL and HBOA when DMLA_HT, MHA_HT, MLA_HT, DMLA_CL, MHA_CL, MLA_CL and HBOA of alarm (ALM) occurs. 4) Alarm is not detected in variation rate high/low limiter and heater burnout detection.

Processing Operation

Control mode \ Processing	Split conversion	Variation rate limiter high/low limiter	Tight shut/ full open	MV reverse	Tracking	Output conversion	Auto tuning result reflection	Tracking of PID parameters	Alarm	Heater burnout detection	Tem-perature anomaly detection
MAN, CMV	×	×	○	○	○ (*1)	○	×	○	× (*2)	×	×
AUT	○	○	○	○	○ (*1)	○	×	○	○ (*3)	○	×
CAS, CSV	○	○	○	○	×	○	○ (*4)	○	○ (*3)	○	○

○: Execute ×: Not execute

*1 Tracking is executed when the tracking flag (TRK) is 1.

*2 An alarm (ALM) whose corresponding bit is TRUE (occurred) is reset, and the alarm cannot be detected.

*3 When the bit of Disable Alarm Detection (INH) which corresponds to an alarm is TRUE, the alarm is not detected.

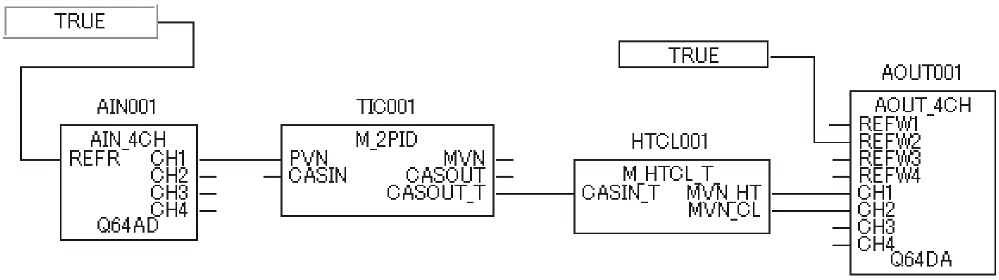
*4: Executed only for CAS mode.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics window of PX Developer programming tool.
Additionally, process control error code as well as the detailed error information, will be displayed on the screen.
For details of process control error code, refer to Appendix 2.

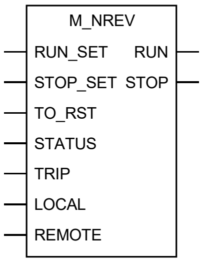
- When overflow occurs during operation. (Error code: Refer to Appendix 2)

Program Example



9.2 Tag FB_Status Tag FB

9.2.1 Motor Irreversible (2 Input, 2 Output) (M_NREV)

FB	FBD parts
M_NREV	

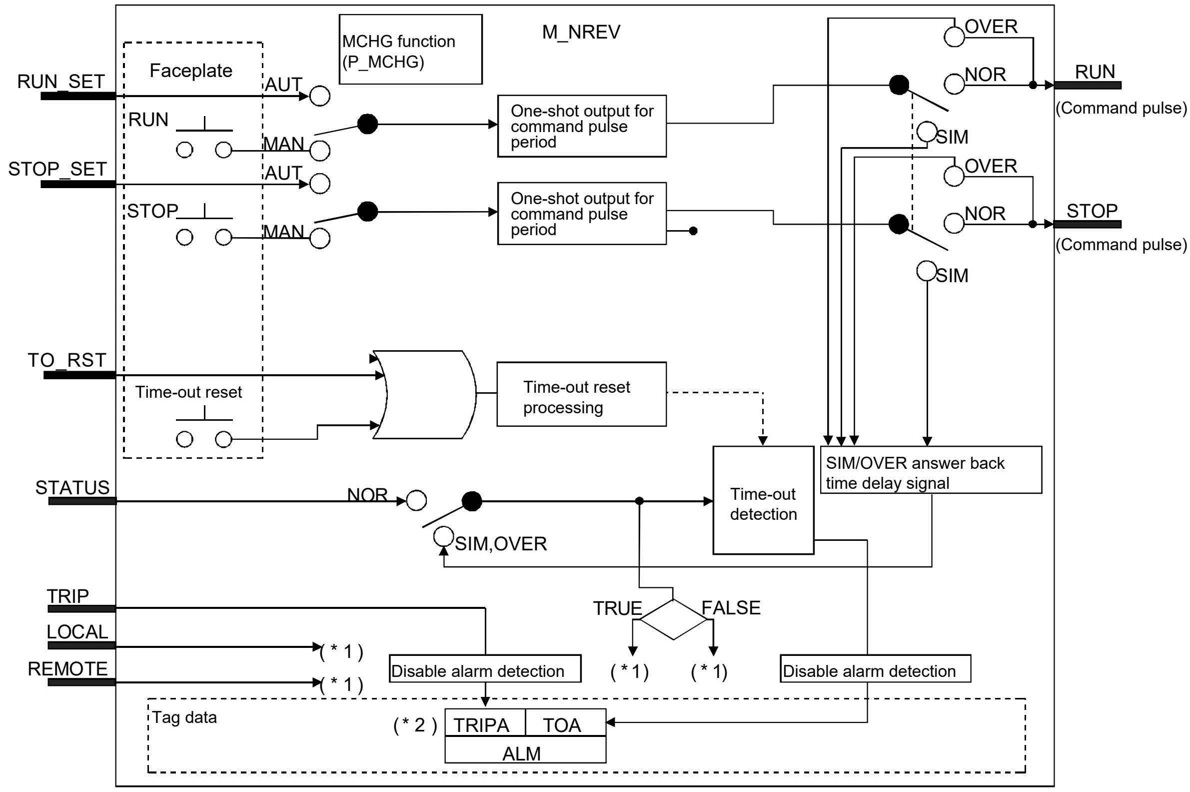
Corresponding tag type
NREV

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	—	—	—

Function overview: Execute irreversible operation and solenoid valve control.

Function/FB classification name: Tag FB_status tag FB

Block Diagram



*1 For displaying the faceplate status.
*2 Indicates bit item.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	RUN_SET	Input variable	BOOL	External input for RUN operation (FALSE →TRUE: RUN)	TRUE, FALSE
	STOP_SET	Input variable	BOOL	External input for STOP operation (FALSE→TRUE: STOP)	TRUE, FALSE
	TO_RST	Input variable	BOOL	External reset input for time-out error (FALSE→TRUE: Time-out reset)	TRUE, FALSE
	STATUS	Input variable	BOOL	Status answer input (TRUE: RUN, FALSE: STOP)	TRUE, FALSE
	TRIP	Input variable	BOOL	External failure (TRIP) input (TRUE: Occurred, FALSE: Recovered)	TRUE, FALSE
	LOCAL	Input variable	BOOL	Local operation selection signal (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE
	REMOTE	Input variable	BOOL	Remote operation selection signal (TRUE: valid, FALSE: invalid)	TRUE, FALSE
Output	RUN	Output variable	BOOL	On output for command pulse period (TRUE: Run, FALSE: -)	TRUE, FALSE
	STOP	Output variable	BOOL	On output for command pulse period (TRUE: Run, FALSE: -)	TRUE, FALSE

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN 2: AUT)	1,2	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute FALSE: Stop)	TRUE, FALSE	FALSE	User

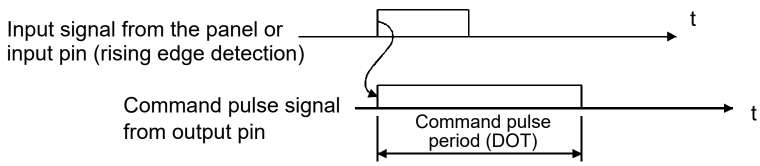
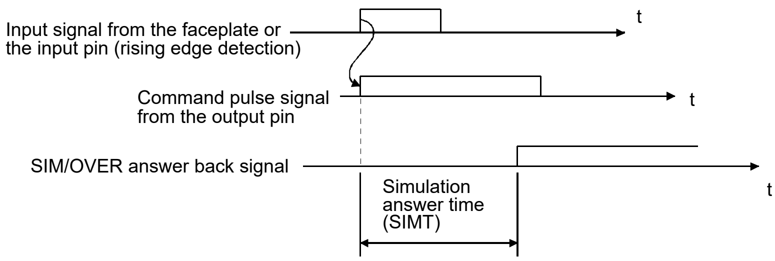
*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the control mode change processing.

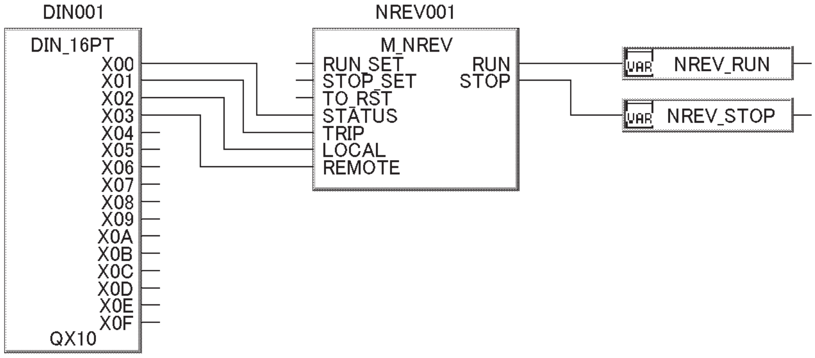
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>One-shot for command pulse period</td><td>Execute the one-shot output for command pulse period according to operation from faceplate or input from input variable (RUN_SET, STOP_SET)
(1) In case of operation from faceplate or the input variable (RUN_SET) transforms from FALSE to TRUE, command pulse signal (TRUE) will be output from the output variable RUN for the time set by command pulse period (DOT).
(2) In case of operation from faceplate or the input variable (STOP_SET) transforms from FALSE to TRUE, command pulse signal (TRUE) will be output from the output variable STOP for the time set by command pulse period (DOT).

</td></tr>
<tr><td>Time-out detection/ time-out reset</td><td>(1) Time-out detection
Time-out (TOA) of alarm (ALM) will occur if TRUE/FALSE is not input from the status answer input (STATUS) for more than the set time of time-out timer (TOT) after command pulse signal (TRUE) is output from output variables RUN/STOP.
<table><tr><th rowspan="2">Condition</th><th>Alarm</th></tr><tr><th>Time-out (TOA)</th></tr><tr><td>Time up to status answer signal input≧setting period of time-out timer (TOT)</td><td>TRUE (occur)</td></tr><tr><td>Time up to status answer signal input<setting period of time-out timer (TOT)</td><td>FALSE (reset)</td></tr></table>(2) Time-out reset
RESET (FALSE) the time-out (TOA) of alarm (ALM) by the following operations.
Output command pulse signal from output variable (RUN, STOP) by the operation from faceplate or input from input variable (RUN_SET, STOP_SET).
Input TRUE to input variable (TO_RST).</td></tr>
<tr><td>SIM/OVER answer back time delay signal</td><td>In case of SIMULATION Mode or OVERRIDE Mode, status answer signal is created in CPU module after command signal output.
The delay time of status answer signal is set by simulation answer time (SIMT).

</td></tr>
<tr><td>Disable Alarm Detection</td><td>If the following items (INH) of tag data are TRUE, TRIPA and TOA of alarm (ALM) will not be detected.
● ERRI, TRIPI, TOI</td></tr>
</table>

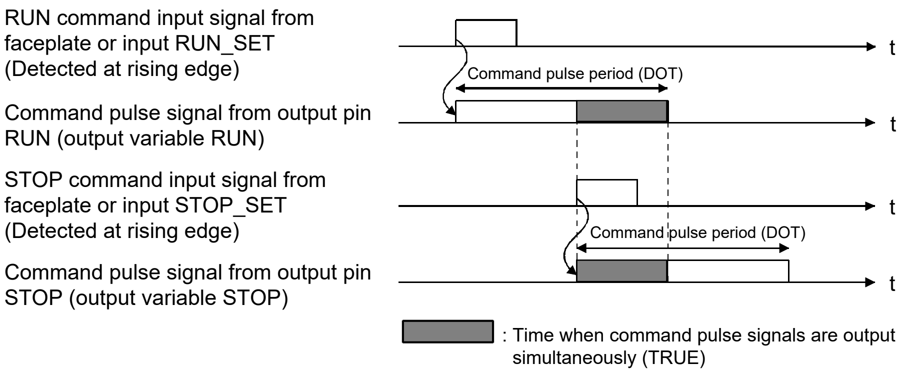
Program Example



POINT

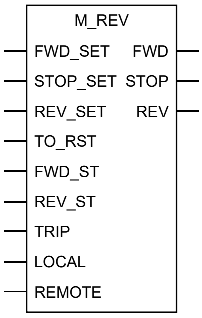
If the STOP command (RUN command) occurs during output of the command pulse signal (TRUE) from the output variable RUN (output variable STOP), the command pulse signals (TRUE) are output simultaneously from the output variable RUN and output variable STOP.

<When command pulse signals (TRUE) are output simultaneously from output variable RUN and output variable STOP>



When the output variable RUN and output variable STOP are output directly to the external device, multiple commands (RUN command and STOP command) may be output simultaneously to the external device.
When it is not desired to output multiple commands to the external device simultaneously, correct the program to output only either one of the command pulse signals to the external device.

9.2.2 Motor Reversible (2 Input, 3 Output) (M_REV)

FB	FBD parts
M_REV	

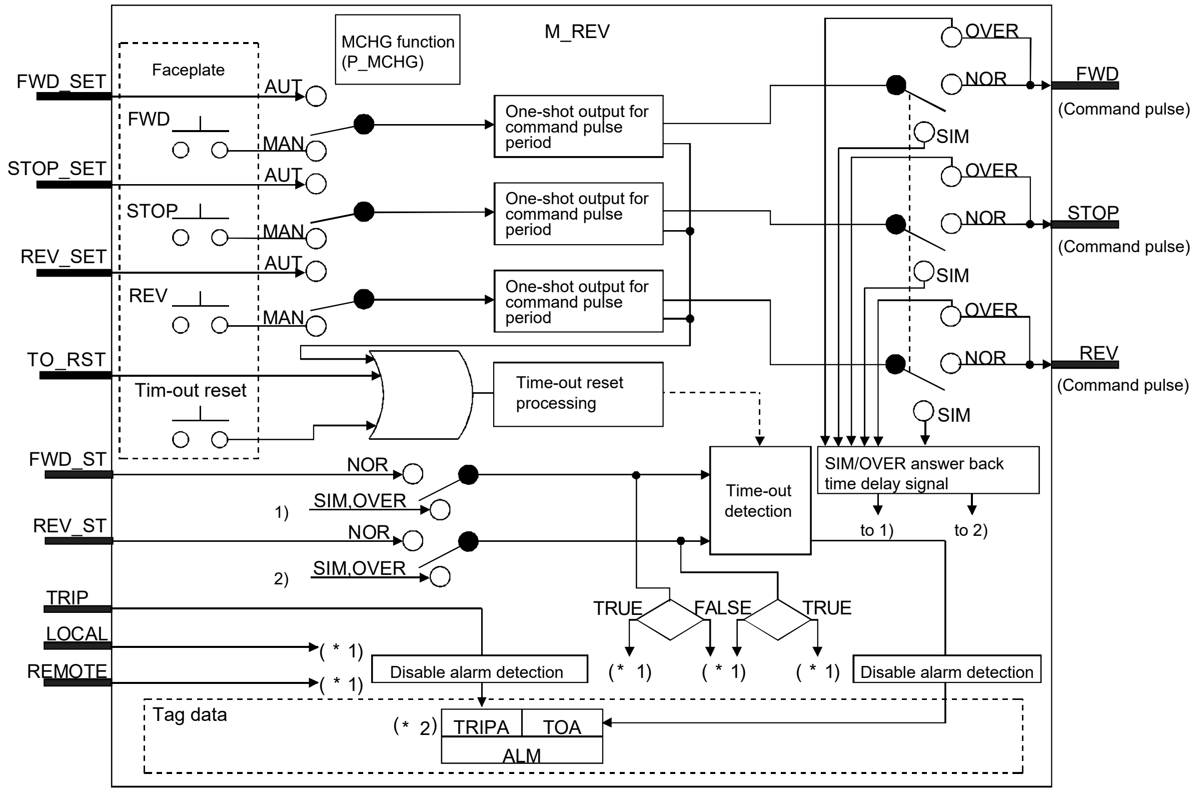
Corresponding tag type
REV

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	—	—	—

Function overview: Execute reversible operation.

Function/FB classification name: Tag FB_status tag FB

Block Diagram



*1 For displaying the faceplate status
*2 Indicates bit item

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	FWD_SET	Input variable	BOOL	External input of FWD (forward rotation) operation (FALSE→TRUE: FWD)	TRUE, FALSE
	STOP_SET	Input variable	BOOL	External input of STOP operation (FALSE→TRUE: STOP)	TRUE, FALSE
	REV_SET	Input variable	BOOL	External input of REV (reverse rotation) operation (FALSE→TRUE)	TRUE, FALSE
	TO_RST	Input variable	BOOL	Time-out error external reset input (FALSE→TRUE: Time-out reset)	TRUE, FALSE
	FWD_ST	Input variable	BOOL	Status answer input (TRUE: REV, FALSE: STOP)	TRUE, FALSE
	REV_ST	Input variable	BOOL	Status answer input (TRUE: REV, FALSE: STOP)	TRUE, FALSE
	TRIP	Input variable	BOOL	External failure (TRIP) input (TRUE: Occurred, FALSE: Recovered)	TRUE, FALSE
	LOCAL	Input variable	BOOL	Local operation selection signal (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE
	REMOTE	Input variable	BOOL	Remote operation selection signal (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE
Output	RUN	Output variable	BOOL	Command pulse period ON output (TRUE: Run, FALSE: -)	TRUE, FALSE
	STOP	Output variable	BOOL	ON output for command pulse period (TRUE: Run, FALSE: -)	TRUE, FALSE
	REV	Output variable	BOOL	ON output for command pulse period (TRUE: Run, FALSE: -)	TRUE, FALSE

Public Variable (Others) (*1)

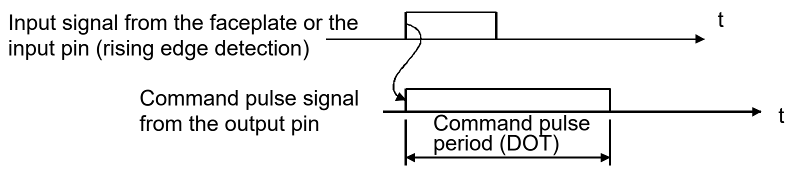
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT)	1, 2	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

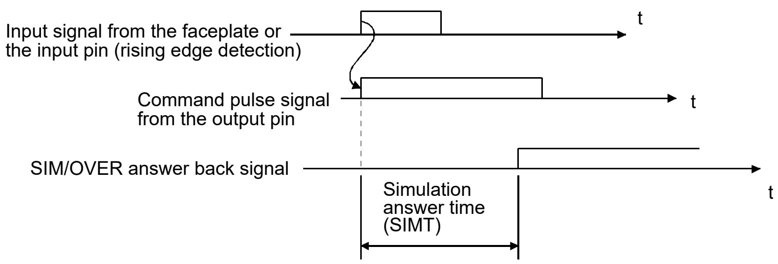
*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the control mode change processing.

Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

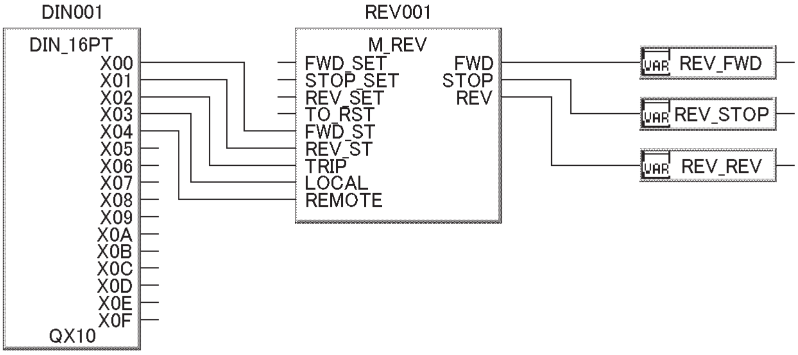
Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>One-shot output for command pulse period</td><td>
Execute the one-shot output for command pulse period according to operation from faceplate or input from input variable (RUN_SET, STOP_SET).
(1) In case of operation from faceplate or the input variable (RUN_SET) transforms from FALSE to TRUE, command pulse signal
(TRUE) will be output from the output variable RUN for the time set by command pulse period (DOT).
(2) In case of operation from faceplate or the input variable (STOP_SET) transforms from FALSE to TRUE, command pulse signal
(TRUE) will be output from the output variable STOP for the period set by command pulse period (DOT).
(3) In case of the operation from faceplate or the input variable (REV_SET) transforms from FALSE to TRUE, command pulse signal (TRUE) will be output from the output variable REV for the period set by command pulse time period (DOT).


</td></tr>
<tr><td>Time-out check/ time-out reset</td><td>
(1) Time-out detection
(a) Alarm (ALM) time-out (TOA) will occur when command pulse signal (TRUE) is output from output variable FWD/STOP and TRUE/FALSE is not input from status answer input (FWD-ST) within the set time of time-out timer (TOT).
(b) Alarm (ALM) time-out (TOA) will occur when command pulse signal (TRUE) is output from output variable REN/STOP and TRUE/FALSE is not input from status answer input (REN-ST) within the set time of time-out timer (TOT).
<table>
<tr><th rowspan="2">Condition</th><th>Alarm</th></tr>
<tr><th>Time-out (TOA)</th></tr>
<tr><td>Time up to status answer signal input≧setting period of time-out timer (TOT)</td><td>TRUE (occur)</td></tr>
<tr><td>Time up to status answer signal input<setting period of time-out timer (TOT)</td><td>FALSE (reset)</td></tr>
</table>
(2) Time-out reset
Reset (FALSE) the time-out (TOA) of alarm (ALM) by the following operations.
(a) Output command pulse signal from output variable (FWD, STOP, REN) by the operation from faceplate or input from input variable (FWD_SET, STOP_SET, REV_SET).
(b) Input TRUE to input variable (TO_RST).
</td></tr>
<tr><td>SIM/OVER answer back time delay signal</td><td>
In case of SIMULATION mode or OVERRIDE mode, status answer signal is created in CPU module after command signal output.
The delay time of status answer signal is set by simulation answer time (SIMT).


</td></tr>
<tr><td>Disable Alarm Detection</td><td>
If the following items (INH) of tag data are TRUE, the TRIPA and TOA of alarm (ALM) will not be detected.
● ERRI, TRIPI, TOI
</td></tr>
</table>

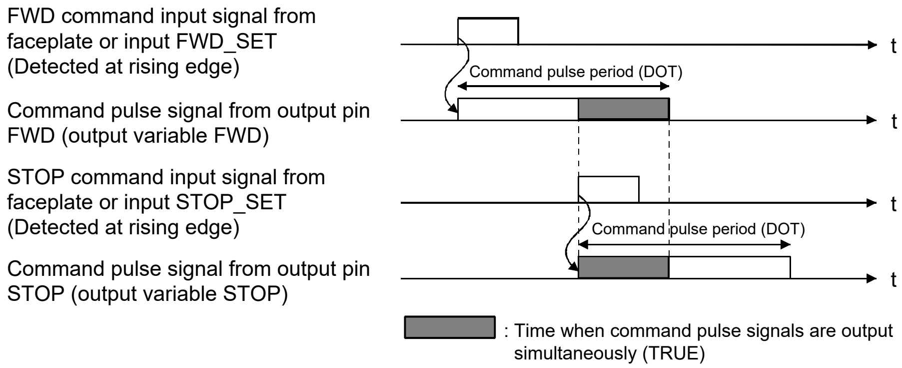
Program Example



POINT

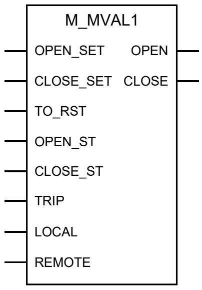
If the other command occurs during output of the command pulse signal (TRUE) from the output variable FWD, STOP or REV, multiple command pulse signals (TRUE) are output.

<When multiple command pulse signals (TRUE) are output>



When the output variables FWD, STOP and REV are output directly to the external device, multiple commands (FWD command, STOP command, REV command) may be output simultaneously to the external device.
When it is not desired to output multiple commands to the external device simultaneously, correct the program to output only any one of the command pulse signals to the external device.

9.2.3 ON/OFF Operation (2 Input, 2 Output) (M_MVAL1)

FB	FBD parts
M_ MVAL1	

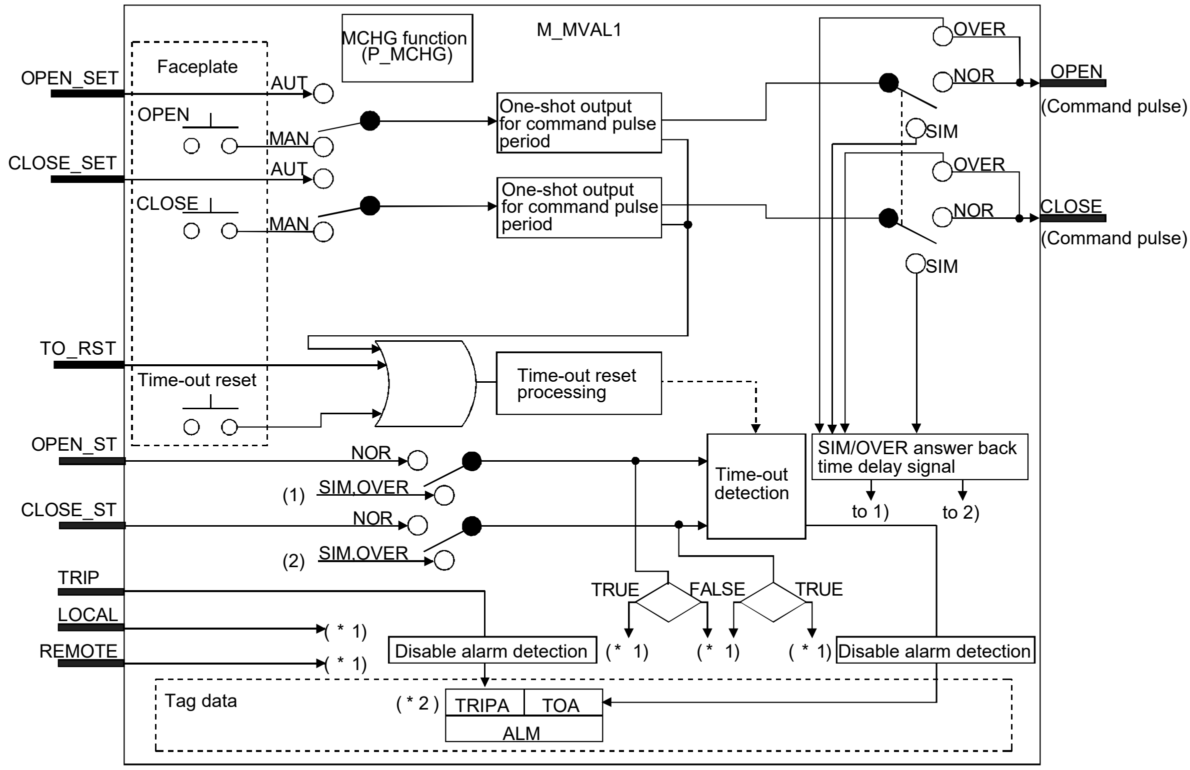
Corresponding tag type
MVAL1

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	—	—	—

Function overview: Execute ON/OFF motor valve and solenoid valve control.

Function/FB classification name: Tag FB_status tag FB

Block Diagram



*1 For displaying the faceplate status
*2 Indicates bit item

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	OPEN_SET	Input variable	BOOL	External input of OPEN operation (FALSE→TRUE: OPEN)	TRUE, FALSE
	CLOSE_SET	Input variable	BOOL	External input of CLOSE operation (FALSE→TRUE: CLOSE)	TRUE, FALSE
	TO_RST	Input variable	BOOL	Time-out error external reset input (FALSE→TRUE: Error reset)	TRUE, FALSE
	OPEN_ST	Input variable	BOOL	Status answer input (TRUE: OPEN, FALSE: SEMI_CLOSE)	TRUE, FALSE
	CLOSE_ST	Input variable	BOOL	Status answer input (TRUE: CLOSE, FALSE: SEMI_CLOSE)	TRUE, FALSE
	TRIP	Input variable	BOOL	External failure (TRIP) input (TRUE: Occurred, FALSE: Recovered)	TRUE, FALSE
	LOCAL	Input variable	BOOL	Local operation selection signal (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE
	REMOTE	Input variable	BOOL	Remote operation selection signal (TRUE: Valid, FALSE: Invalid)	TRUE, FALSE
Output	OPEN	Output variable	BOOL	ON output for command pulse period (TRUE: Run, FALSE: -)	TRUE, FALSE
	CLOSE	Output variable	BOOL	ON output for command pulse period (TRUE: Run, FALSE: -)	TRUE, FALSE

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT)	1, 2	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

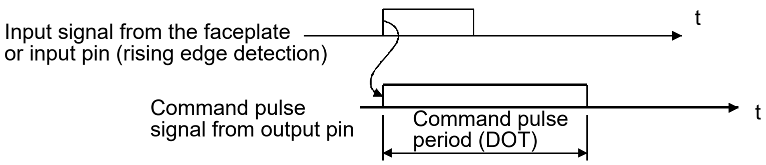
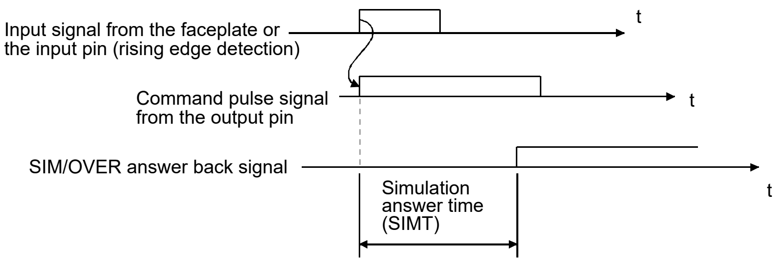
*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the control mode change processing.

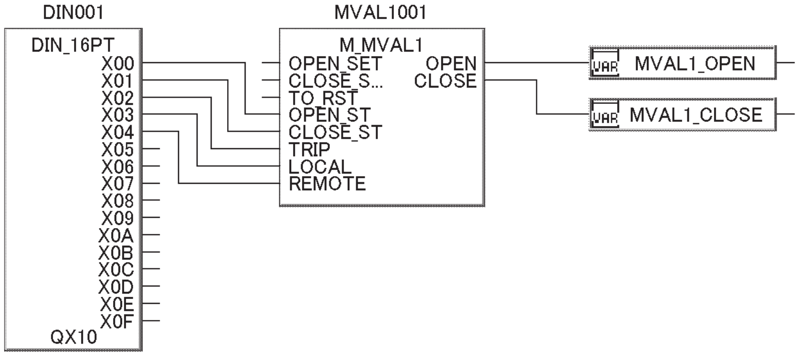
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Command pulse period one shot output</td><td>Execute the one-shot output for command pulse period according to operation from faceplate or input from input variable (OPEN_SET, CLOSE_SET).
(1) In case of operation from faceplate or the input variable (OPEN_SET) transforms from FALSE to TRUE, command pulse signal
(TRUE) will be output from the output variable RUN for the time set by command pulse period (DOT).
(2) In case of operation from faceplate or the input variable (CLOSE_SET) transforms from FALSE to TRUE, command pulse signal (TRUE) will be output from the output variable STOP for the time set by command pulse period (DOT).

</td></tr>
<tr><td>Time-out detection/ time-out reset</td><td>(1) Time-out detection
Time-out (TOA) of alarm (ALM) will occur if TRUE is not input from the status answer input (OPEN_ST/CLOSE_ST) for more than the set time of time-out timer (TOT) after command pulse signal (TRUE) is output from output variables OPEN/CLOSE.
<table>
<tr><th rowspan="2">Condition</th><th>Alarm</th></tr>
<tr><th>Time-out (TOA)</th></tr>
<tr><td>Time up to status answer signal input≧setting period of time-out timer (TOT)</td><td>TRUE (occur)</td></tr>
<tr><td>Time up to status answer signal input<setting period of time-out timer (TOT)</td><td>FALSE (reset)</td></tr>
</table>
(2) Time-out reset
Reset (FALSE) the time-out (TOA) of alarm (ALM) by the following operations.
(a) Output command pulse signal from output variable (OPEN, CLOSE) by the operation from faceplate or input from input variable (OPEN_SET, CLOSE_SET).
(b) Input TRUE to input variable (TO_RST).</td></tr>
<tr><td>SIM/OVER answer back time delay signal</td><td>In case of SIMULATION Mode or OVERRIDE Mode, status answer signal is created in CPU module after command signal output.
The delay time of status answer signal is set by analog answer time (SIMT).

</td></tr>
<tr><td>Disable Alarm Detection</td><td>If the following items (INH) of tag data are TRUE, alarm (ALM) of TRIPA and TOA will not be detected.
● ERRI, TRIPI, TOI</td></tr>
</table>

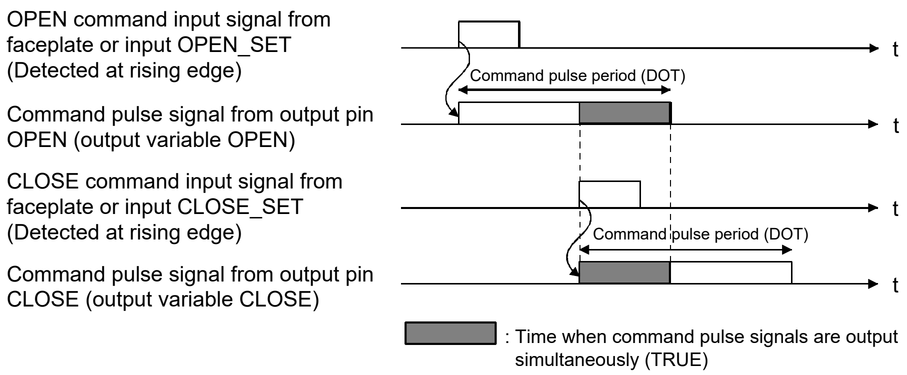
Program Example



POINT

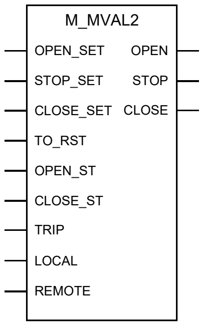
If the CLOSE command (OPEN command) occurs during output of the command pulse signal (TRUE) from the output variable OPEN (output variable CLOSE), the command pulse signals (TRUE) are output simultaneously from the output variable OPEN and output variable CLOSE.

<When command pulse signals (TRUE) are output simultaneously from output variable OPEN and output variable CLOSE>



When the output variable OPEN and output variable CLOSE are output directly to the external device, multiple commands (OPEN command and CLOSE command) may be output simultaneously to the external device.
When it is not desired to output multiple commands to the external device simultaneously, correct the program to output only either one of the command pulse signals to the external device.

9.2.4 ON/OFF Operation (2 Input, 3 Output) (M_MVAL2)

FB	FBD parts
M_MVAL2	

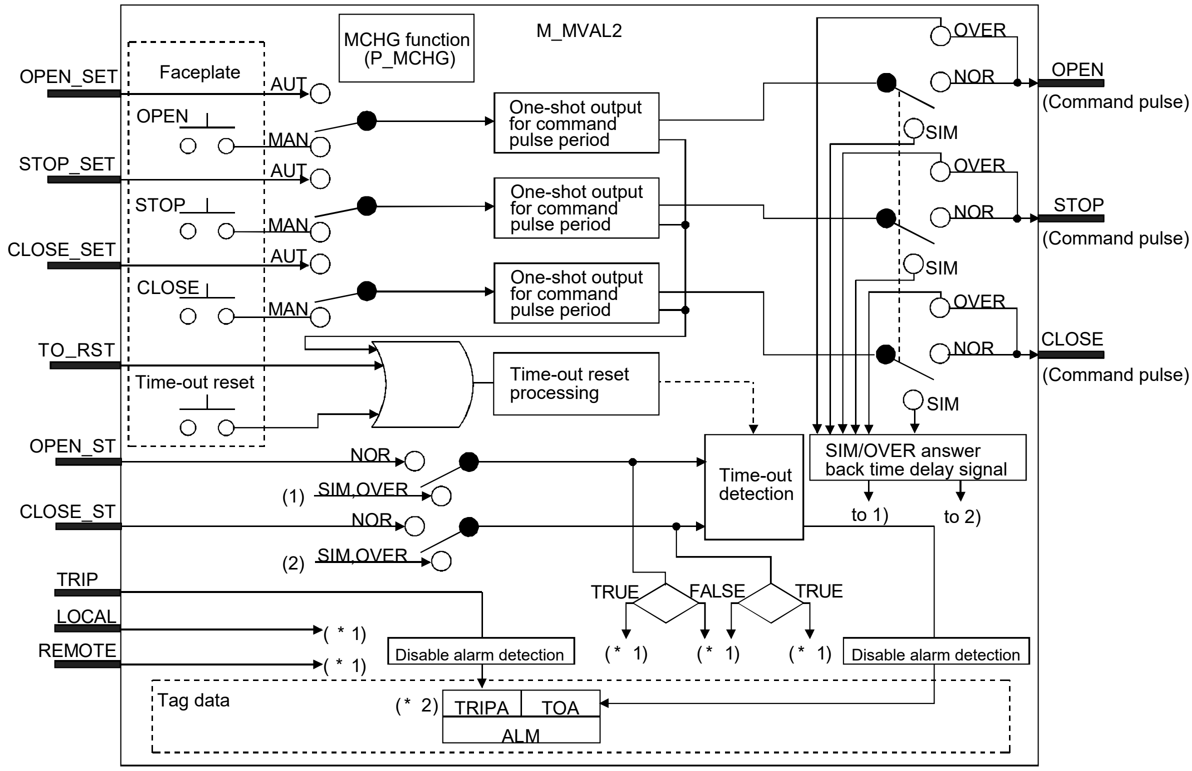
Corresponding tag type
MVAL2

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	—	—	—

Function overview: Execute ON/OFF motor valve (with intermediate status) control.

Function/FB classification name: Tag FB_status tag FB

Block Diagram



*1 For displaying the faceplate status
*2 Indicates bit item

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	OPEN_SET	Input variable	BOOL	External input of OPEN operation (FALSE→TRUE: OPEN)	TRUE, FALSE
	STOP_SET	Input variable	BOOL	External input of STOP operation (FALSE→TRUE: STOP)	TRUE, FALSE
	CLOSE_SET	Input variable	BOOL	External input of CLOSR operation (FALSE→TRUE: CLOSE)	TRUE, FALSE
	TO_RST	Input variable	BOOL	Time-out error external reset input (FALSE→TRUE: Time-out reset)	TRUE, FALSE
	OPEN_ST	Input variable	BOOL	Status answer input (TRUE: OPEN, FALSE: SEMI_CLOSE)	TRUE, FALSE
	CLOSE_ST	Input variable	BOOL	Status answer input (TRUE: CLOSE, FALSE: SEMI_CLOSE)	TRUE, FALSE
	TRIP	Input variable	BOOL	External failure (TRIP) input (TRUE: Occur, FALSE: Reset)	TRUE, FALSE
	LOCAL	Input variable	BOOL	Local operation selection signal (TRUE: Valid, FALSE: invalid)	TRUE, FALSE
	REMOTE	Input variable	BOOL	Remote operation selection signal (TRUE: valid, FALSE: invalid)	TRUE, FALSE
Output	OPEN	Output variable	BOOL	Command pulse period ON output (TRUE: Run, FALSE: -)	TRUE, FALSE
	STOP	Output variable	BOOL	Command pulse period ON output (TRUE: Run, FALSE: -)	TRUE, FALSE
	CLOSE	Output variable	BOOL	Command pulse period ON output (TRUE: Run, FALSE: -)	TRUE, FALSE

Public Variable (Others) (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT)	1,2	0	User
	E	Public variable	BOOL	Change request (TRUE: Execute, FALSE: Stop)	TRUE, FALSE	FALSE	User

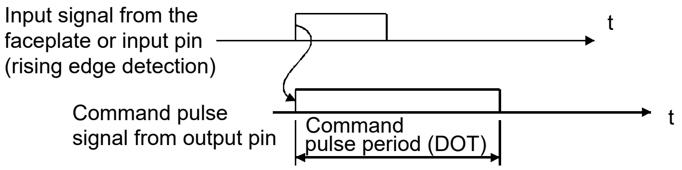
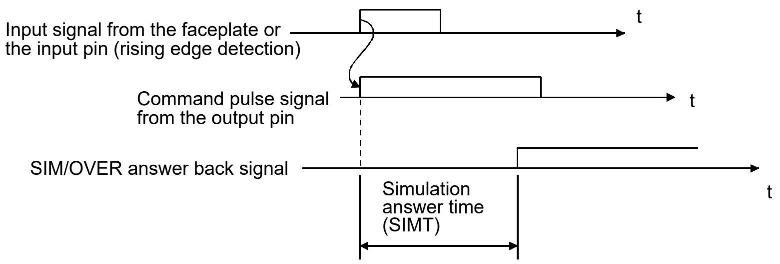
*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

*2 Indicates the control mode change processing.

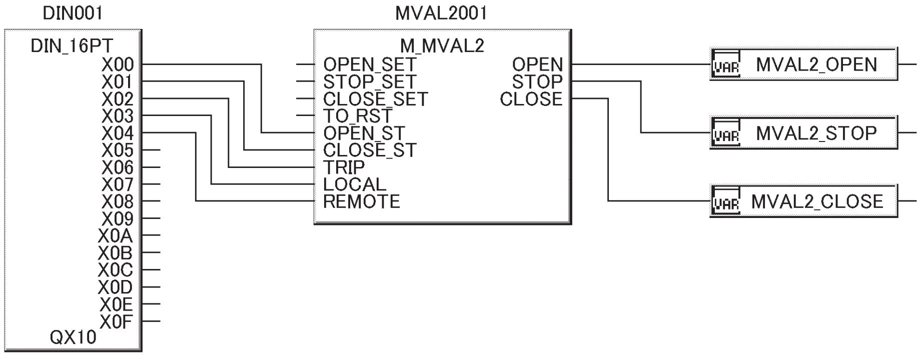
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Command pulse period one shot output</td><td>Execute the one-shot output for command pulse period according to operation from faceplate or input from input variable (OPEN_SET, STOP_SET, CLOSE_SET).
(1) In case of operation from faceplate or the input variable (OPEN_SET) transforms from FALSE to TRUE, instruction pulse signal (TRUE) will be output from the output variable RUN for the period set by command pulse period (DOT).
(2) In case of operation from faceplate or the input variable (CLOSE_SET) transforms from FALSE to TRUE, command pulse signal (TRUE) will be output from the output variable STOP for the period set by command pulse period (DOT).
(3) In case of operation from faceplate or the input variable (CLOSE_SET) transforms from FALSE to TRUE, command pulse signal (TRUE) will be output from the output variable CLOSE for the period set by command pulse period (DOT).

</td></tr>
<tr><td>Time-out check/time-out reset</td><td>(1) Time-out detection
Time-out (TOA) of alarm (ALM) will occur if TRUE is not input from the status answer input (OPEN_ST/CLOSE_ST) for more than the set time of time-out time (TOT) after command pulse signal (TRUE) is output from output variables OPEN/CLOSE.
<table>
<tr><th rowspan="2">Condition</th><th>Alarm</th></tr>
<tr><th>Time-out (TOA)</th></tr>
<tr><td>Time up to status answer signal input≧setting period of time-out timer (TOT)</td><td>TRUE (occur)</td></tr>
<tr><td>Time up to status answer signal input<setting period of time-out timer (TOT)</td><td>FALSE (reset)</td></tr>
</table>
(2) Time-out reset
Reset (FALSE) the time-out (TOA) of alarm (ALM) by the following operations.
(a) Output command pulse signal from output variable (OPEN, STOP, CLOSE) by the operation from panel or input from input variable (OPEN_SET, STOP_SET, CLOSE_SET).
(b) Input TRUE to input variable (TO_RST).</td></tr>
<tr><td>SIM/OVER answer back time delay signal</td><td>In case of SIMULATION Mode or OVERRIDE Mode, status answer signal is created in CPU module after command signal output.
The delay time of status answer signal is set by simulation answer time (SIMT).

</td></tr>
<tr><td>Disable Alarm Detection</td><td>If the following items (INH) of tag data are TRUE, alarm (ALM) of TRIPA and TOA will not be detected.
● ERRI, TRIPI, TOI</td></tr>
</table>

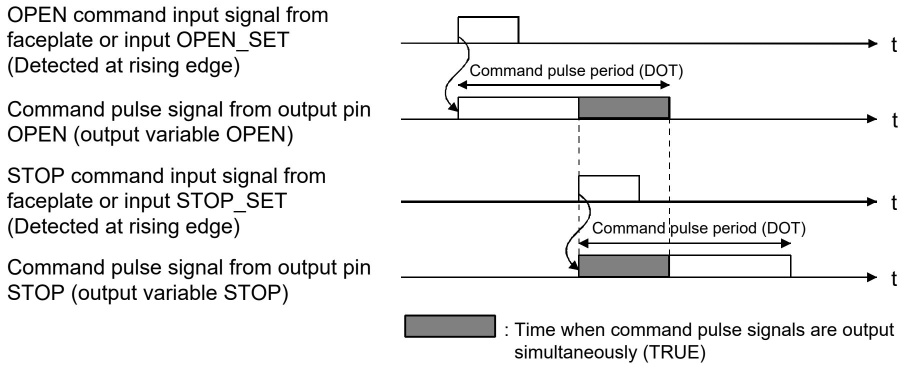
Program Example



POINT

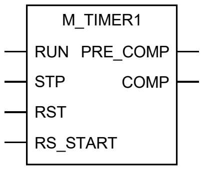
If the other command occurs during output of the command pulse signal (TRUE) from the output variable OPEN, STOP or CLOSE, multiple command pulse signals (TRUE) are output.

<When multiple command pulse signals (TRUE) are output>



When the output variables OPEN, STOP and CLOSE are output directly to the external device, multiple commands (OPEN command, STOP command, CLOSE command) may be output simultaneously to the external device.
When it is not desired to output multiple commands to the external device simultaneously, correct the program to output only any one of the command pulse signals to the external device.

9.2.5 Timer 1 (Timer Stops When COMPLETE Flag is ON) (M_TIMER1)

FB	FBD parts
M_TIMER1	

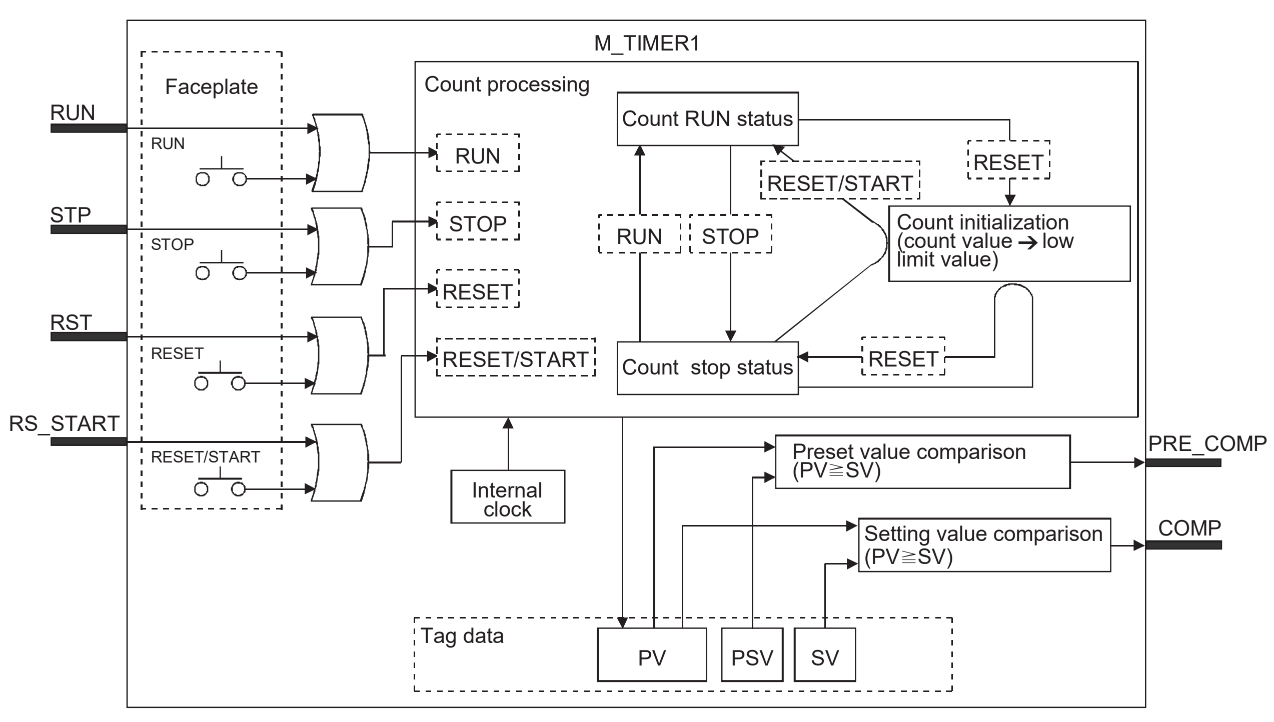
Corresponding tag type
TIMER1

Control mode				
MAN	AUT	CAS	CMV	CSV
—	—	—	—	—

Function overview: It is a clock timer. Timing stops when the count value reaches the setting value.

Function/FB classification name: Tag FB_status tag FB

Block Diagram



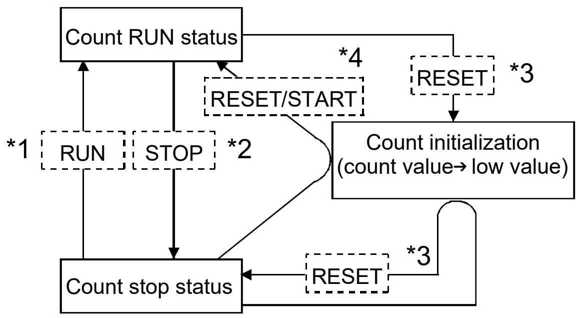
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	RUN	Input variable	BOOL	External input of RUN operation (FALSE→TRUE: RUN)	TRUE, FALSE
	STP	Input variable	BOOL	External input of STOP operation (FALSE→TRUE: STOP)	TRUE, FALSE
	RST	Input variable	BOOL	External input of RESET operation (FALSE→TRUE: RESET)	TRUE, FALSE
	RS_START	Input variable	BOOL	External input of RESET/START operation (FALSE→TRUE: RESET/START)	TRUE, FALSE
Output	PRE_COMP	Output variable	BOOL	Preset value count up completed (TRUE: Completed, FALSE: Not completed)	TRUE, FALSE
	COMP	Output variable	BOOL	Setting value count up completed (TURE: Completed, FALSE: Not completed)	TRUE, FALSE

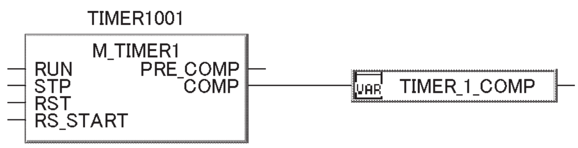
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

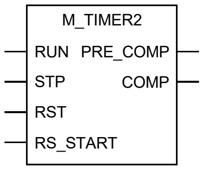
Function

Item	Contents
Count processing	(1) In case of operation from faceplate, or the input variable (RUN) transforms from FALSE to TRUE, the timer current value is stored in process variable (PV) in unit set by timer multiplying factor (MULT). (Following figure *1) When the process variable (PV) reaches the preset value (PSV) TRUE will be output from output variable PRE_COMP. (Block diagram─Preset value comparison) When the process variable (PV) reaches the setting value (SV) TRUE will be output from output variable COMP and the timer clock stops. (BLOCK diagram–Setting value comparison) When the process variable (PV) reaches timer high limit (RH), timer clock stops. (2) In case of operation from faceplate, or the input variable (STP) transforms from FALSE to TRUE, process variable (PV) measuring will be stopped. (Following figure *2) (3) In case of operation from faceplate, or the input variable (RST) transforms from FALSE to TRUE, process variable (PV) will be set as timer low limit (RL) value, and timer clock stops. (Following figure *3) (4) In case of operation from faceplate, or the input variable (RS_START) transforms from FALSE to TRUE, process variable (PV) will be set as timer low limit (RL) value, and timer clock starts. (Following figure *4) 

Program Example



9.2.6 Timer 2 (Timer Continues When COMPLETE Flag is ON) (M_TIMER2)

FB	FBD parts
M_TIMER2	

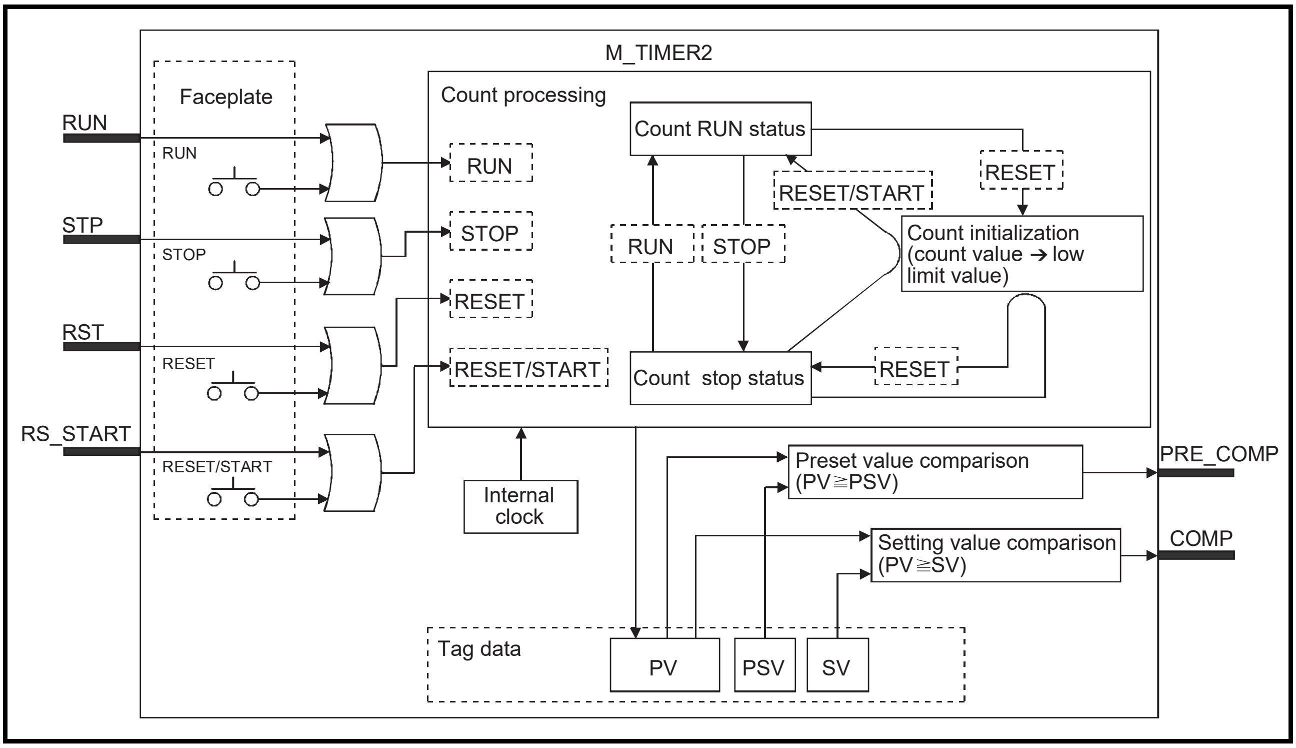
Corresponding tag type
TIMER2

Control mode				
MAN	AUT	CAS	CMV	CSV
—	—	—	—	—

Function overview: It is a clock timer. Timing continues even if the count value reaches the setting value. Timing stops when the count value reaches the high limit value.

Function/FB classification name: Tag FB_status tag FB

Block Diagram



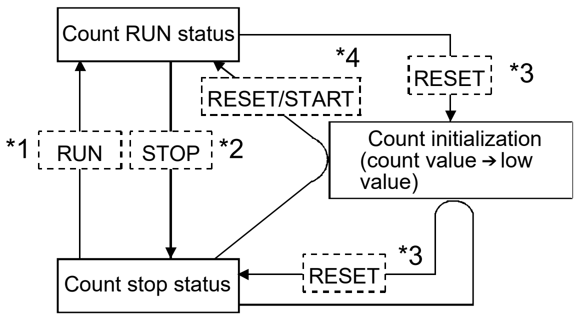
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	RUN	Input variable	BOOL	External input of RUN operation (FALSE→TRUE: RUN)	TRUE, FALSE
	STP	Input variable	BOOL	External input of STOP operation (FALSE→TRUE: STOP)	TRUE, FALSE
	RST	Input variable	BOOL	External input of RESET operation (FALSE→TRUE: RESET)	TRUE, FALSE
	RS_START	Input variable	BOOL	External input of RESET/START operation (FALSE→TRUE: RESET/START)	TRUE, FALSE
Output	PRE_COMP	Output variable	BOOL	Preset value count up completed (TRUE: Completed, FALSE: Not completed)	TRUE, FALSE
	COMP	Output variable	BOOL	Setting value count up completed (TURE: Completed, FALSE: Not completed)	TRUE, FALSE

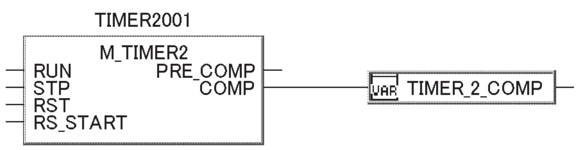
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

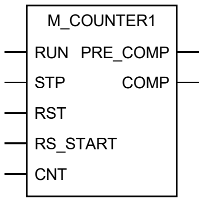
Function

Item	Contents
Count processing	(1) In case of operation for faceplate, or the input variable (RUN) transforms from FALSE to TRUE, the timer current value is stored in process variable (PV) in unit set by timer multiplying factor (MULT). When the process variable (PV) reaches the preset value (PSV) TRUE will be output from output variable PRE_COMP. (Block diagram －Preset value comparison) When the process variable (PV) reaches the setting value (SV) TRUE will be output from output variable COMP and the timer clock stops. (BLOCK diagram－Setting value comparison) When the process variable (PV) reaches timer high limit (RH), timer clock stops. (2) In case of operation from faceplate, or the input variable (STP) transforms from FALSE to TRUE, process variable (PV) measuring will be stopped. (Following figure *2) (3) In case of operation from faceplate, or the input variable (RST) transforms from FALSE to TRUE, process variable (PV) will be set as timer low limit (RL) value, and timer clock stops. (Following figure *3) (4) In case of operation from faceplate, or the input variable (RS_START) transforms from FALSE to TRUE, process variable (PV) will be set as timer low limit (RL) value, and timer clock starts. (Following figure *4) 

Program Example



9.2.7 Counter 1 (Counter Stops When COMPLETE Flag is ON) (M_COUNTER1)

FB	FBD parts
M_COUNTER1	

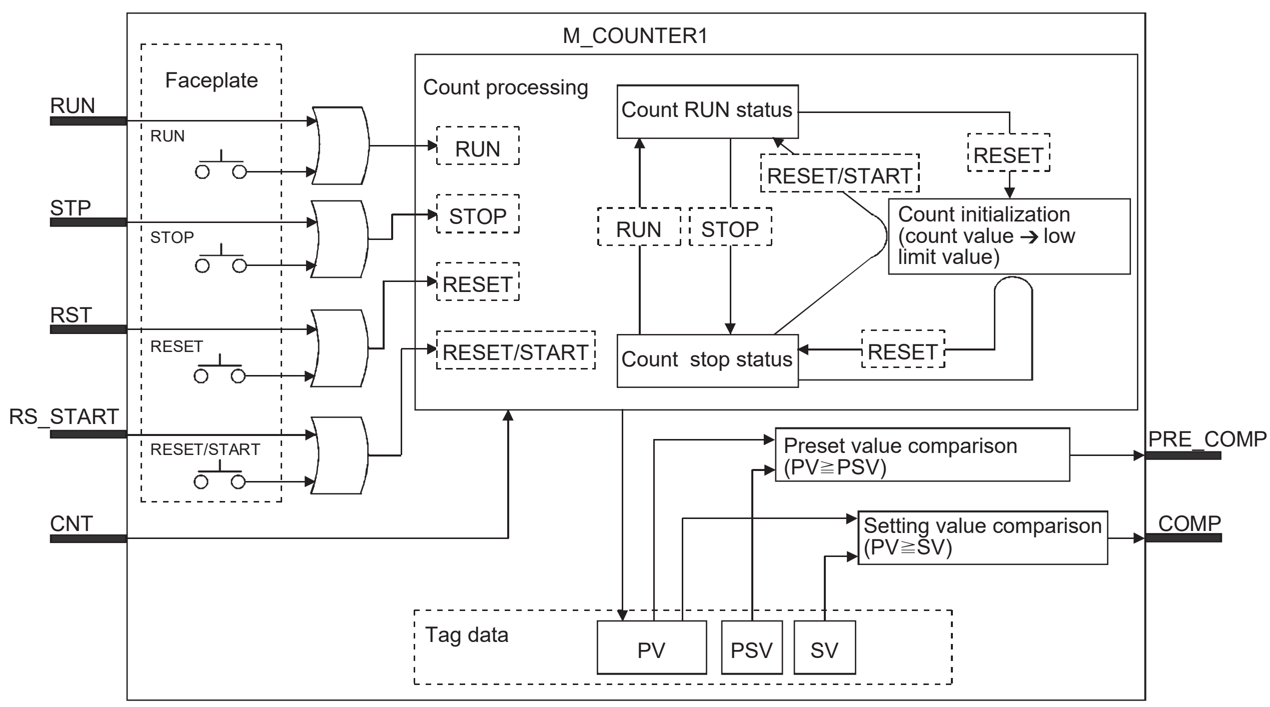
Corresponding tag type
COUNT1

Control mode				
MAN	AUT	CAS	CMV	CSV
—	—	—	—	—

Function overview: It is a counter that counts contact signal input. Count stops when count value reaches the setting value.

Function/FB classification name: Tag FB_status tag FB

Block Diagram



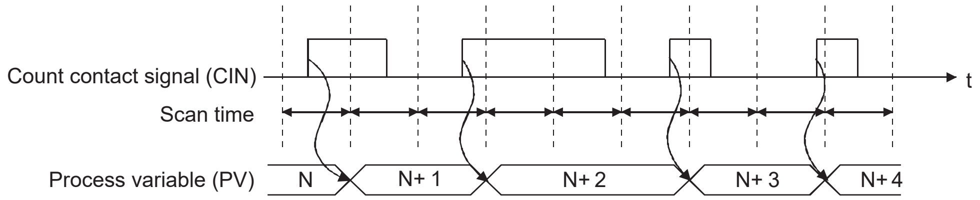
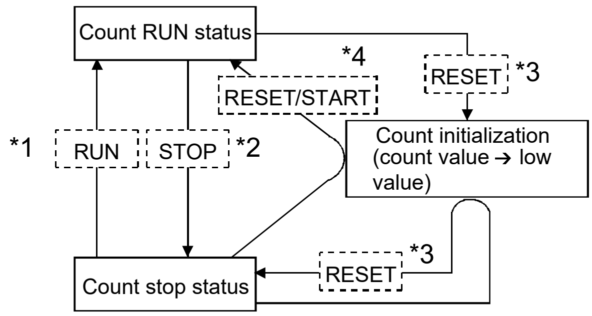
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	RUN	Input variable	BOOL	External input of RUN operation (FALSE→TRUE: RUN)	TRUE, FALSE
	STP	Input variable	BOOL	External input of STOP operation (FALSE→TRUE: STOP)	TRUE, FALSE
	RST	Input variable	BOOL	External input of RESET operation (FALSE→TRUE: RESET)	TRUE, FALSE
	RS_START	Input variable	BOOL	External input of RESET/START operation (FALSE→TRUE: RESET/START)	TRUE, FALSE
	CNT	Input variable	BOOL	Count contact signal input (FALSE→TRUE: Count)	TRUE, FALSE
Output	PRE_COMP	Output variable	BOOL	Preset value count up completed (TRUE: Completed, FALSE: Not completed)	TRUE, FALSE
	COMP	Output variable	BOOL	Setting value count up completed (TURE: Completed, FALSE: Not complete)	TRUE, FALSE

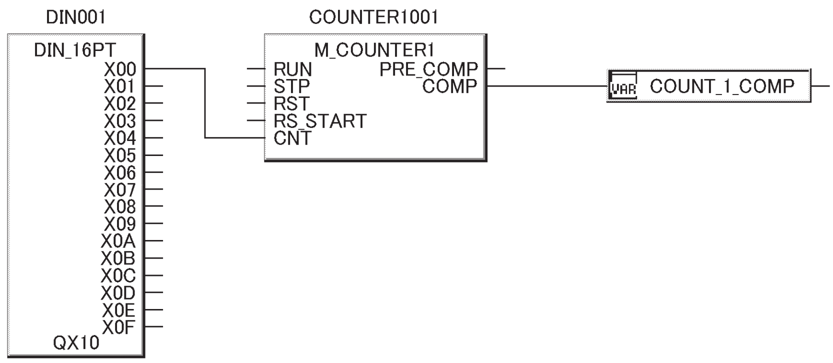
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

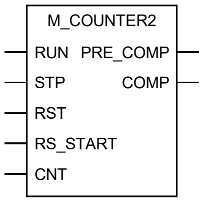
Function

Item	Contents
Count input (CNT)	Detect the rising edge (when FALSE→TRUE) of input variable (CIN) and perform count processing. (Current value +1). 
Count processing	(1) In case of operation from faceplate, or the input variable (RUN) transforms from FALSE to TRUE, the current value +1 (counter value) is stored in process variable (PV) in unit set by counter multiplying factor (MULT). (Following figure *1) When the process variable (PV) reaches the preset value (PSV) TRUE will be output from output variable PRE_COMP. (Block diagram — Preset value comparison) When the process variable (PV) reaches the setting value (SV), TRUE will be output from output variable COMP and the timer clock stops. (BLOCK diagram — Setting value comparison) When the process variable (PV) reaches timer high limit (RH), timer clock stops. (2) In case of operation from faceplate, or the input variable (STP) transforms from FALSE to TRUE, process variable (PV) measuring will be stopped. (Following figure *2) (3) In case of operation from faceplate, or the input variable (RST) transforms from FALSE to TRUE, process variable (PV) will be set as timer low limit (RL) value, and timer clock stops. (Following figure *3) (4) In case of operation in faceplate, or the input variable (RS_START) transforms from FALSE to TRUE, process variable (PV) will be set as timer low limit (RL) value, and timer clock starts. (Following figure *4) 

Program Example



9.2.8 Counter 2 (Counter Continues When COMPLETE Flag is ON) (M_COUNTER2)

FB	FBD parts
M_COUNTER2	

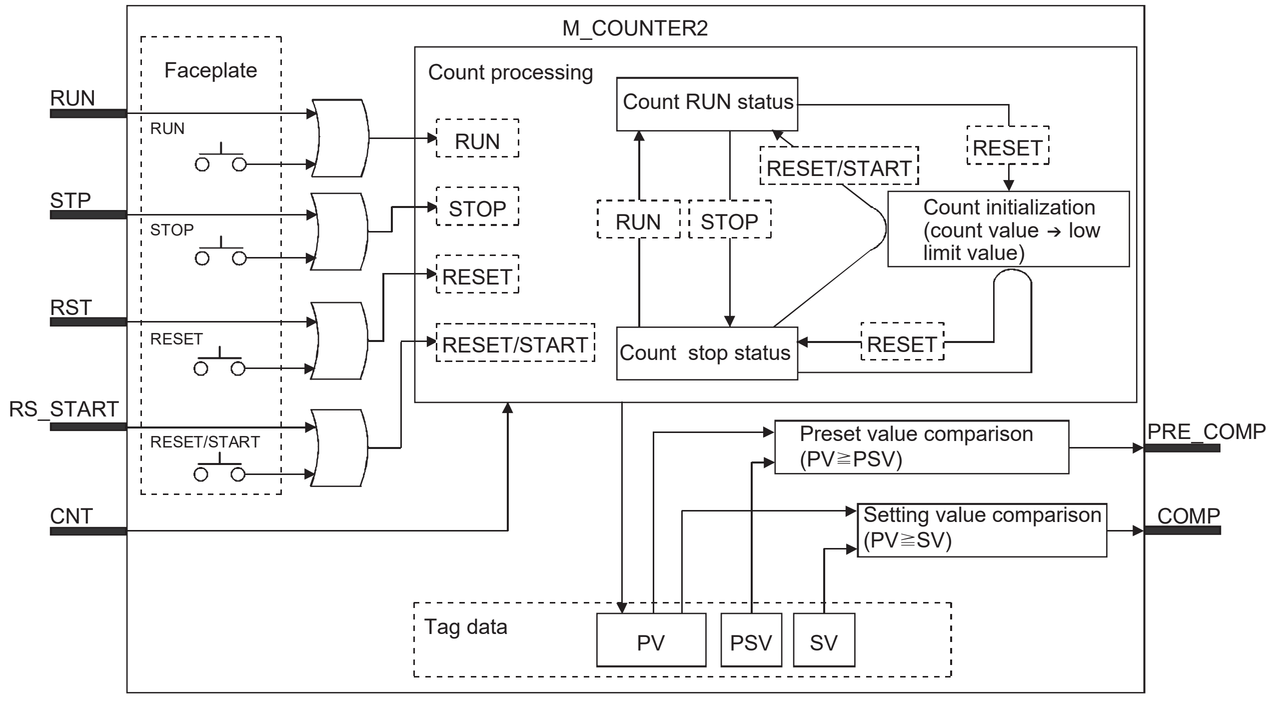
Corresponding tag type
COUNT2

Control mode				
MAN	AUT	CAS	CMV	CSV
—	—	—	—	—

Function overview: It is a counter that counts contact signal input. Count continues when count value reaches the setting value. Count stops when the count value reaches the high limit value.

Function/FB classification name: Tag FB_status tag FB

Block Diagram



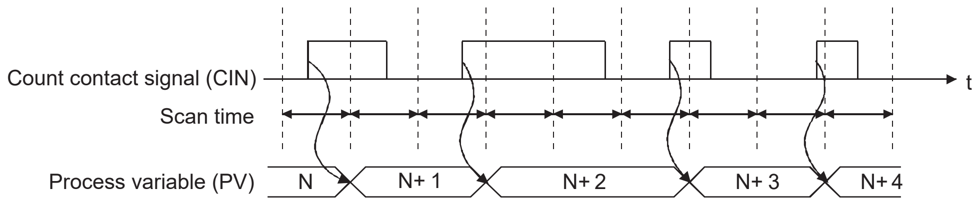
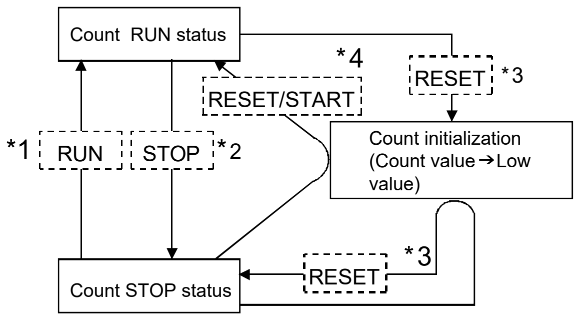
Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	RUN	Input variable	BOOL	External input of RUN operation (FALSE→TRUE: RUN)	TRUE, FALSE
	STP	Input variable	BOOL	External input of STOP operation (FALSE→TRUE: STOP)	TRUE, FALSE
	RST	Input variable	BOOL	External input of RESET operation (FALSE→TRUE: RESET)	TRUE, FALSE
	RS_START	Input variable	BOOL	External input of RESET/START operation (FALSE→TRUE: RESET/START)	TRUE, FALSE
	CNT	Input variable	BOOL	Count contact signal input (FALSE→TRUE: Count)	TRUE, FALSE
Output	PRE_COMP	Output variable	BOOL	Preset value count up completed (TRUE: Completed, FALSE: Not complete)	TRUE, FALSE
	COMP	Output variable	BOOL	Setting value count up completed (TURE: Completed, FALSE: Not complete)	TRUE, FALSE

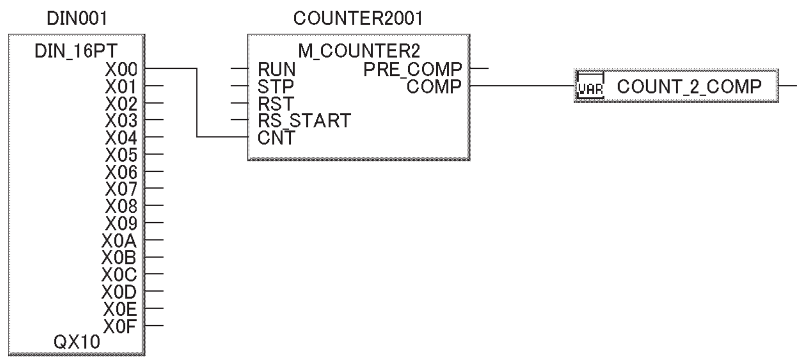
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

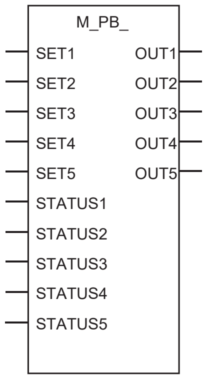
Function

Item	Contents
Count input (CNT)	Detect the rising edge (FALSE→TRUE) of input variable (CIN) and perform count processing. (Current value +1). 
Count processing	(1) In case of operation from faceplate, or the input variable (RUN) transforms from FALSE to TRUE, the current value + 1 (counter value) is stored in process variable (PV) in unit set by counter multiplying factor (MULT). When the process variable (PV) reaches the preset value (PSV) TRUE will be output from output variable PRE_COMP. (Block diagram — Preset value comparison) When the process variable (PV) reaches the setting value (SV) TRUE will be output from output variable COMP and the timer clock stops. (BLOCK figure — Setting value comparison) When the process variable (PV) reaches timer high limit (RH), timer clock stops. (2) In case of operation from faceplate, or the input variable (STP) transforms from FALSE to TRUE, process variable (PV) measuring will be stopped. (Following figure *2) (3) In case of operation from faceplate, or the input variable (RST) transforms from FALSE to TRUE, process variable (PV) will be set as timer low limit (RL) value, and timer clock stops. (Following figure *3) (4) In case of operation from faceplate, or the input variable (RS_START) transforms from FALSE to TRUE, process variable (PV) will be set as timer low limit (RL) value, and timer clock starts. (Following figure *4) 

Program Example



9.2.9 Push Button Operation (5 Input, 5 Output) (M_PB_)

FB	FBD parts
M_PB_	

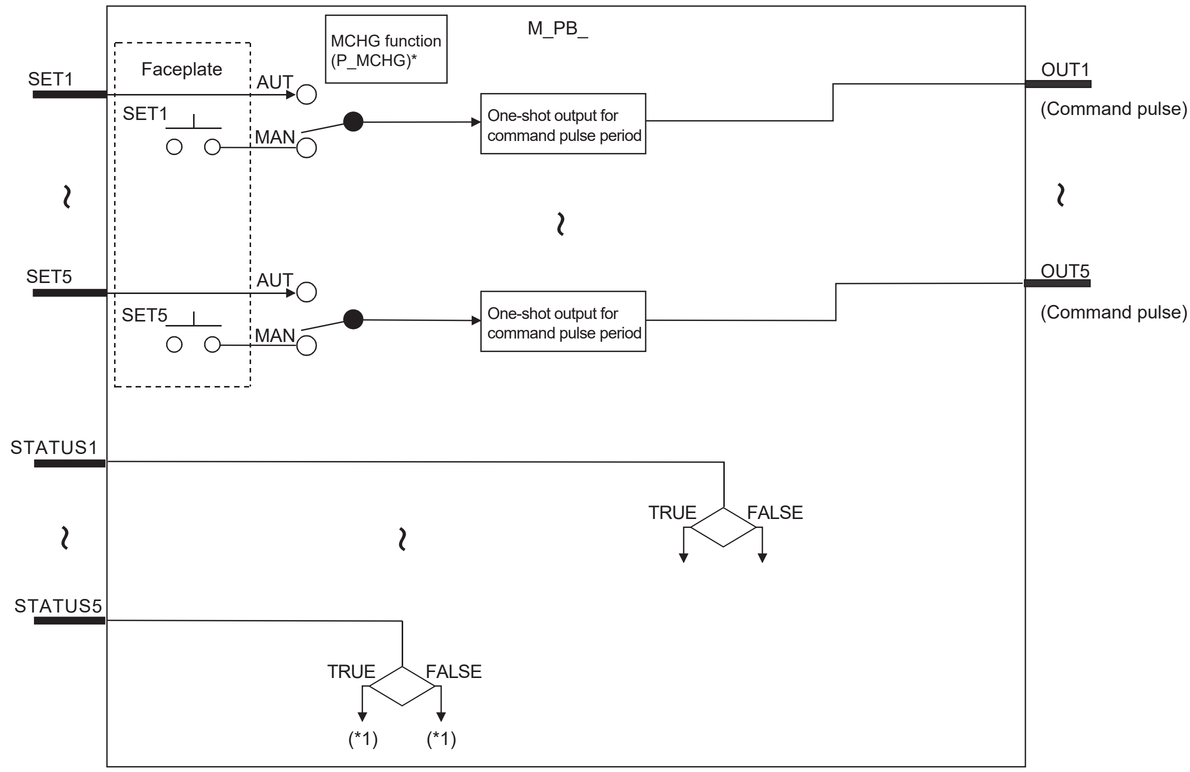
Corresponding tag type
PB

Control mode				
MAN	AUT	CAS	CMV	CSV
○	○	—	—	—

Function overview: Execute push button operation.

Function/FB classification name: Tag FB_status tag FB

Block Diagram



*1 For displaying the faceplate status.

Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	SET1	Input variable	BOOL	External input of OUT1 operation (FALSE→TRUE: ON)	TRUE, FALSE
	SET2	Input variable	BOOL	External input of OUT2 operation (FALSE→TRUE: ON)	TRUE, FALSE
	SET3	Input variable	BOOL	External input of OUT3 operation (FALSE→TRUE: ON)	TRUE, FALSE
	SET4	Input variable	BOOL	External input of OUT4 operation (FALSE→TRUE: ON)	TRUE, FALSE
	SET5	Input variable	BOOL	External input of OUT5 operation (FALSE→TRUE: ON)	TRUE, FALSE
	STATUS1	Input variable	BOOL	Status1 answer input (TRUE: ON, FALSE: OFF)	TRUE, FALSE
	STATUS2	Input variable	BOOL	Status2 answer input (TRUE: On, FALSE: OFF)	TRUE, FALSE
	STATUS3	Input variable	BOOL	Status3 answer input (TRUE: On, FALSE: OFF)	TRUE, FALSE
	STATUS4	Input variable	BOOL	Status4 answer input (TRUE: On, FALSE: OFF)	TRUE, FALSE
	STATUS5	Input variable	BOOL	Status5 answer input (TRUE: On, FALSE: OFF)	TRUE, FALSE
Output	OUT1	Output variable	BOOL	Command pulse period ON output (TRUE: Command, FALSE: -)	TRUE, FALSE
	OUT2	Output variable	BOOL	Command pulse period ON output (TRUE: Command, FALSE: -)	TRUE, FALSE
	OUT3	Output variable	BOOL	Command pulse period ON output (TRUE: Command, FALSE: -)	TRUE, FALSE
	OUT4	Output variable	BOOL	Command pulse period ON output (TRUE: Command, FALSE: -)	TRUE, FALSE
	OUT5	Output variable	BOOL	Command pulse period ON output (TRUE: Command, FALSE: -)	TRUE, FALSE

Public Variable (Others) (*1)

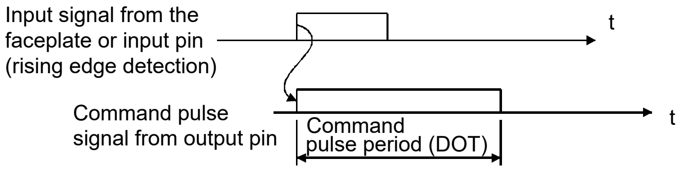
	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
MCHG processing*2	MODEIN	Public variable	INT	Mode change signal (1: MAN, 2: AUT)	1, 2	0	User
	E	Public variable	BOOL	Change request (TRUE: Execution, FALSE: Stop)	TRUE, FALSE	FALSE	User

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.
*2 Indicates the control mode change processing.

Tag Data

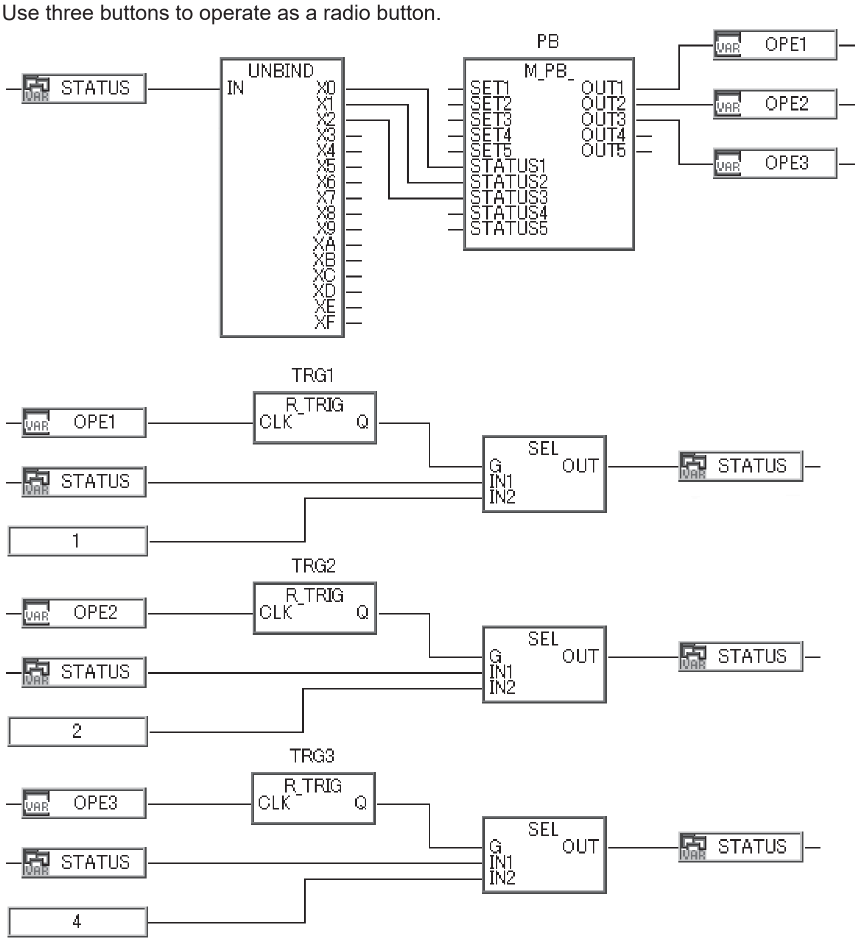
For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

Item	Contents
Command pulse period one shot output	Execute the one-shot output for command pulse period according to operation from faceplate or input from input variable (SET1 to SET5). (1) In case of operation from faceplate or the input variable (SET1 to SET5) transforms from FALSE to TRUE, instruction pulse signal (TRUE) will be output from the output variable (OUT1 to OUT5) for the period set by command pulse period (DOT). 

Program Example

Use three buttons to operate as a radio button.

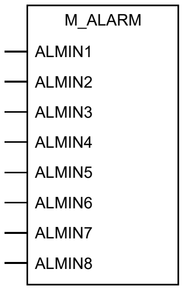


POINT

A one-shot command can be output by clicking each button from the faceplate, and names of the ON/OFF status can be displayed.
In addition, operations similar to a radio button can be achieved by combining buttons.

9.3 Tag FB_Alarm Tag FB

9.3.1 Alarm (M_ALARM)

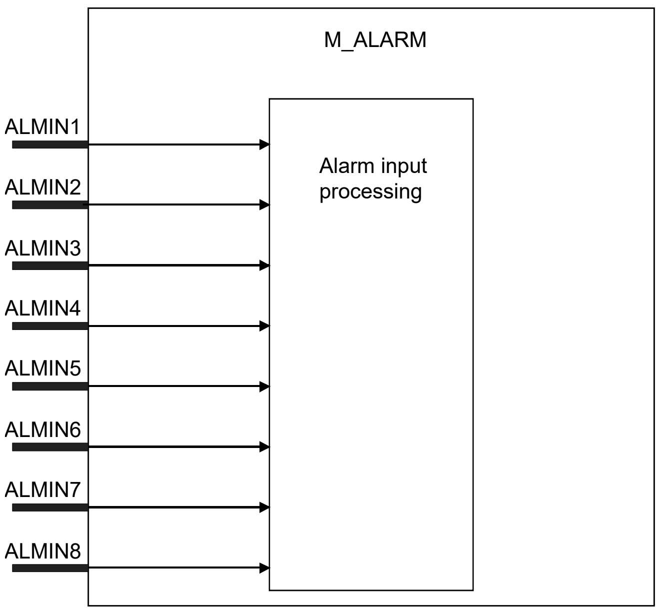
FB	FBD parts
M_ALARM	

Corresponding tag type
ALM

Function overview: The corresponding alarm of the input pins (ALMIN1 to ALMIN8) to which TRUE is input is displayed on the alarm list screen of PX Developer monitor tool.

Function/FB classification name: Tag FB_alarm tag FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	ALMIN1	Input variable	BOOL	Alarm 1 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	ALMIN2	Input variable	BOOL	Alarm 2 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	ALMIN3	Input variable	BOOL	Alarm 3 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	ALMIN4	Input variable	BOOL	Alarm 4 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	ALMIN5	Input variable	BOOL	Alarm 5 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	ALMIN6	Input variable	BOOL	Alarm 6 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	ALMIN7	Input variable	BOOL	Alarm 7 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	ALMIN8	Input variable	BOOL	Alarm 8 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE

Tag Data

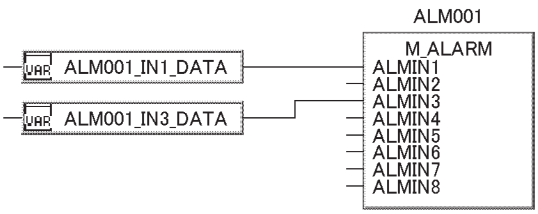
For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

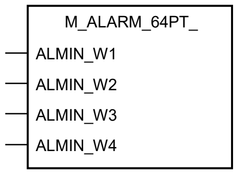
This tag FB consists of the following tag access FB.

Item	Contents
Alarm input processing	The corresponding alarm of the input pins (ALMIN1 to ALMIN8) to which TRUE is input is displayed on the alarm list screen of PX Developer monitor tool. For the operating methods of PX Developer monitor tool, refer to "PX Developer Version 1 Operating Manual (Monitor Tool)"

Program Example



9.3.2 64-points alarm (M_ALARM_64PT_)

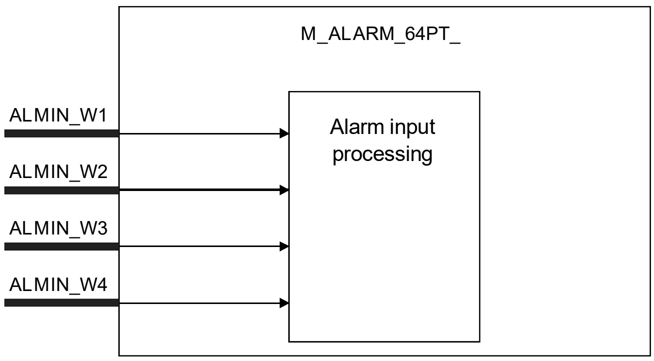
FB	FBD parts
M_ALARM_64PT_	

Corresponding tag type
ALM_64PT

Function overview: The corresponding alarm of the input pins (ALMIN_W1 to ALMIN_W4) to which TRUE is input is displayed on the alarm list screen of PX Developer monitor tool.

Function/FB classification name: Tag FB_alarm tag FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	ALMIN_W1	Input variable	WORD	Alarm 1 to 16 input signal Alarm 1 to 16 can be specified in order of bit 0 to 15. (For each bit, TRUE: Occur, FALSE: Recover)	0_H to $FFFF_H$
	ALMIN_W2	Input variable	WORD	Alarm 17 to 32 input signal Alarm 17 to 32 can be specified in order of bit 0 to 15. (For each bit, TRUE: Occur, FALSE: Recover)	0_H to $FFFF_H$
	ALMIN_W3	Input variable	WORD	Alarm 33 to 48 input signal Alarm 33 to 48 can be specified in order of bit 0 to 15. (For each bit, TRUE: Occur, FALSE: Recover)	0_H to $FFFF_H$
	ALMIN_W4	Input variable	WORD	Alarm 49 to 64 input signal Alarm 49 to 64 can be specified in order of bit 0 to 15. (For each bit, TRUE: Occur, FALSE: Recover)	0_H to $FFFF_H$

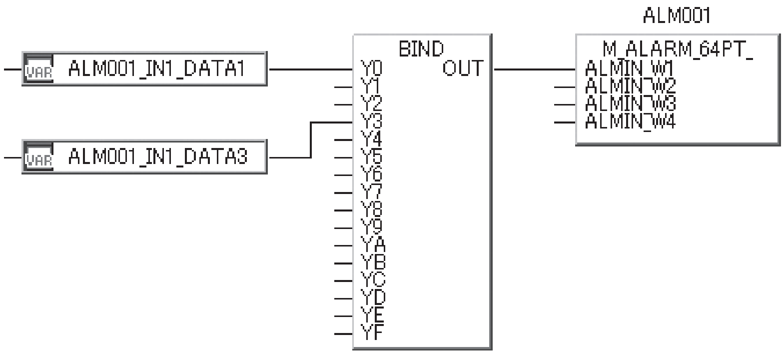
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

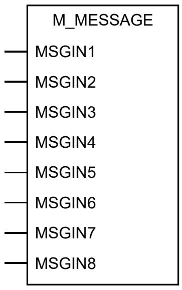
Item	Contents
Alarm input processing	The corresponding alarm of the input pins (ALMIN_W1 to ALMIN_W4) to which TRUE is input is displayed on the alarm list screen of PX Developer monitor tool. For the operating methods of PX Developer monitor tool, refer to "PX Developer Version 1 Operating Manual (Monitor Tool)"

Program Example



9.4 Tag FB_Message Tag FB

9.4.1 Message (M_MESSAGE)

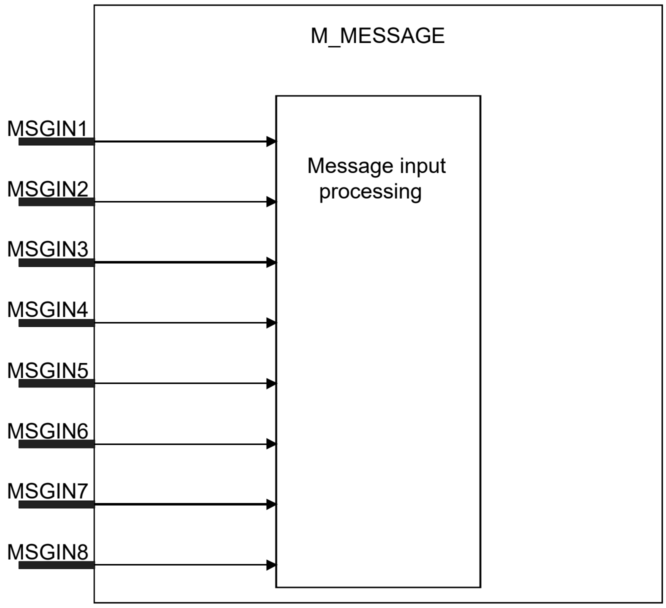
FB	FBD parts
M_MESSAGE	

Corresponding tag type
MSG

Function overview: The corresponding message of the input pins (MSGIN1 to MSGIN8) to which TRUE is input is displayed on the event list screen of PX Developer monitor tool.

Function/FB classification name: Tag FB_Message tag FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	MSGIN1	Input variable	BOOL	Message 1 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	MSGIN2	Input variable	BOOL	Message 2 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	MSGIN3	Input variable	BOOL	Message 3 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	MSGIN4	Input variable	BOOL	Message 4 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	MSGIN5	Input variable	BOOL	Message 5 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	MSGIN6	Input variable	BOOL	Message 6 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	MSGIN7	Input variable	BOOL	Message 7 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE
	MSGIN8	Input variable	BOOL	Message 8 input signal (TRUE: Occur, FALSE: Recover)	TRUE, FALSE

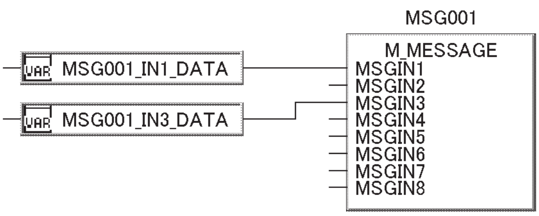
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

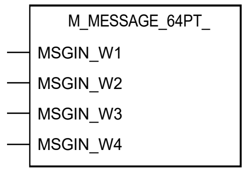
Function

Item	Contents
Message input processing	The corresponding message of the input pins (MSGIN1 to MSGIN8) to which TRUE is input is displayed on the event list screen of PX Developer monitor tool. For the operating methods of PX Developer monitor tool, refer to "PX Developer Version 1 Operating Manual (Monitor Tool)"

Program Example



9.4.2 64-points message (M_MESSAGE_64PT_)

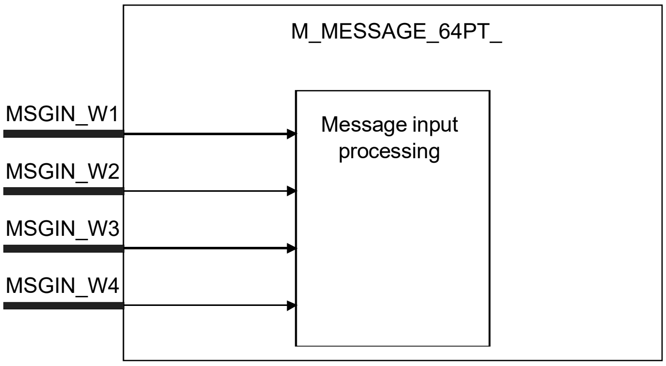
FB	FBD parts
M_MESSAGE_64PT_	

Corresponding tag type
MSG_64PT

Function overview: The corresponding message of the input pins (MSGIN_W1 to MSGIN_W4) to which TRUE is input is displayed on the event list screen of PX Developer monitor tool.

Function/FB classification name: Tag FB_Message tag FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	MSGIN_W1	Input variable	WORD	Message 1 to 16 input signal Message 1 to 16 can be specified in order of bit 0 to 15. (For each bit, TRUE: Occur, FALSE: Recover)	0_H to $FFFF_H$
	MSGIN_W2	Input variable	WORD	Message 17 to 32 input signal Message 17 to 32 can be specified in order of bit 0 to 15. (For each bit, TRUE: Occur, FALSE: Recover)	0_H to $FFFF_H$
	MSGIN_W3	Input variable	WORD	Message 33 to 48 input signal Message 33 to 48 can be specified in order of bit 0 to 15. (For each bit, TRUE: Occur, FALSE: Recover)	0_H to $FFFF_H$
	MSGIN_W4	Input variable	WORD	Message 49 to 64 input signal Message 49 to 64 can be specified in order of bit 0 to 15. (For each bit, TRUE: Occur, FALSE: Recover)	0_H to $FFFF_H$

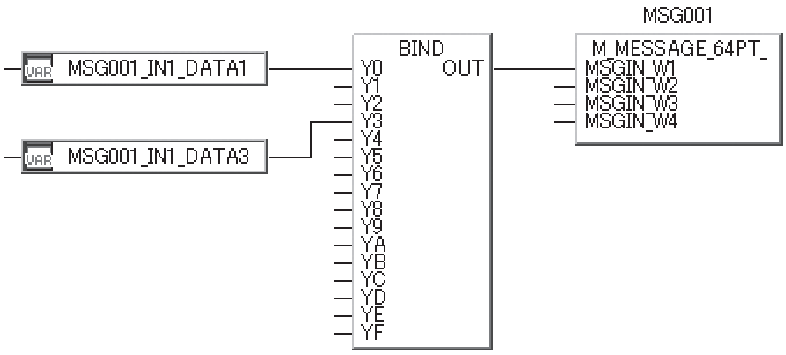
Tag Data

For details about the tag data that is read/written by this tag FB, refer to tag data list of various tag types in Appendix 1.1.

Function

Item	Contents
Message input processing	The corresponding message of the input pins (MSGIN_W1 to MSGIN_W4) to which TRUE is input is displayed on the event list screen of PX Developer monitor tool. For the operating methods of PX Developer monitor tool, refer to "PX Developer Version 1 Operating Manual (Monitor Tool)"

Program Example



MEMO

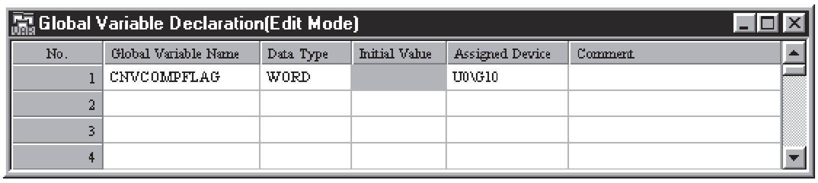
10 MODULE FB

Module FB can be classified into following types according to module buffer memory area reading/writing instructions.

Classification name	Contents	Reference
Analog module FB	● Reads A/D conversion values from A/D converter modules (4 channels, 8 channels), channel-isolated A/D converter modules (4 channels, 8 channels), channel-isolated high resolution A/D converter module with signal conditioning function (2 channels), and channel-isolated A/D converter module with signal conditioning function (6 channels). ● Writes digital values to 2 channels, 4 channels, 8 channels, 2 channel-isolated, and 6 channel-isolated D/A converter modules.	Section 10.1
Temperature input module FB	Read temperature conversion value from 4/8 channels temperature input module.	Section 10.2
Counter module FB	Read pulse count value from high-speed counter module and isolation type pulse input module	Section 10.3
Digital input/output module FB	● Read ON/OFF input value from 8/16/32/64 points input module. ● Write ON/OFF output value to 8/16/32/64 points output module. ● Read/write ON/OFF input/output value from/to 15/64 points input/output mixed module.	Section 10.4
CC-Link module FB	Read/write information of CC-Link slave stations occupying 1/2/3/4 stations.	Section 10.5

(1) Read/write of I/O signals and buffer memory data that do not exist in public variables
Some of the I/O signals and buffer memory data of the module can be read/written with the public variables of the module FB.
On the other hand, read/write the I/O signals and buffer memory data, which do not exist in the public variables of the module FB, using either of the following methods.

(a) Read/write using global variable
When declaring a global variable in the global variable declaration window of the programming tool, set the I/O signal or buffer memory address to Assigned device.



The declared global variable is placed in the FBD program and used for read/write.

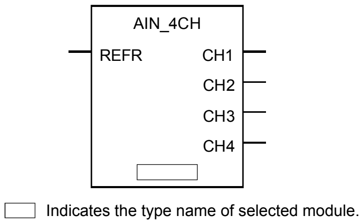
(b) Read/write using GX application
Perform read/write using Device batch or Buffer memory batch of GX application.
For details of GX application, refer to the following manuals:
- GX Works2 Version 1 Operating Manual (Common)
- GX Developer Version 8 Operating Manual

POINT

The usable I/O signals and buffer memory addresses change depending on the module.
For details, refer to the manual of the used module.

10.1 Analog Module FB

10.1.1 4 Channels Analog Input (AIN_4CH)

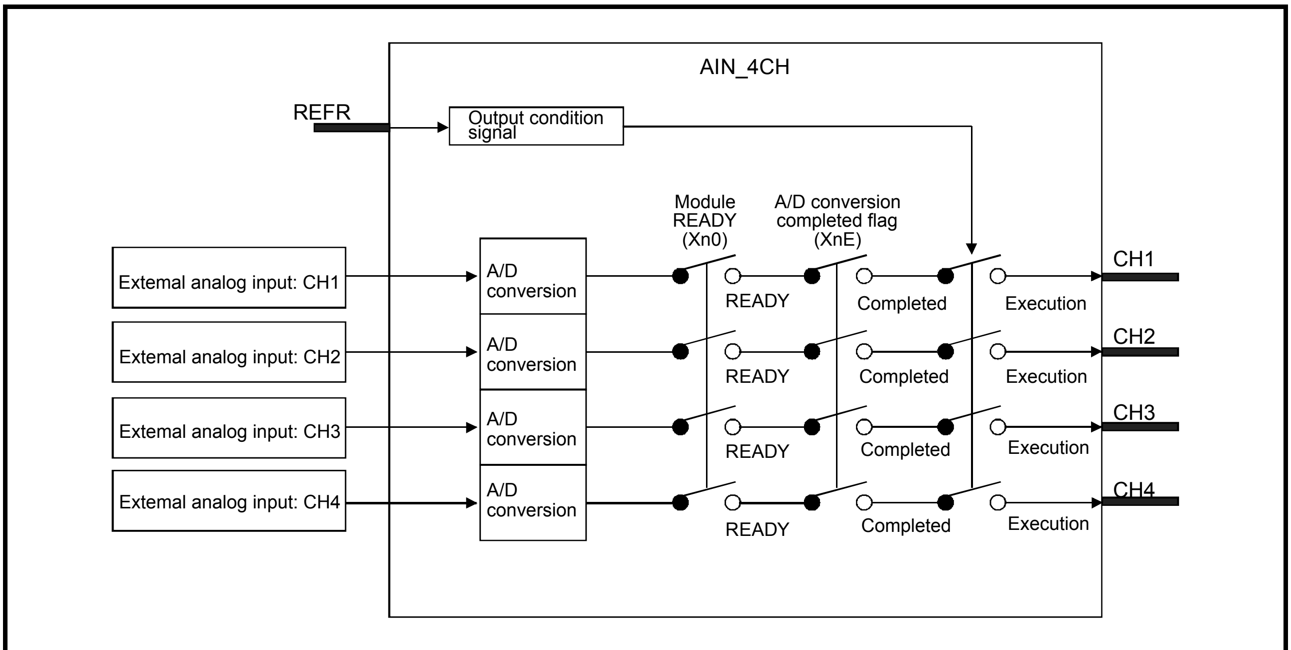
FB	FBD parts
AIN_4CH	

Corresponding module
Q64AD

Function overview: Reads the digital output value of 4 channels A/D conversion module that converts analog signal to digital value, and outputs it from output variable (CH1 to CH4).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH4	Output variable	REAL	CH1 to CH4 digital output value	Depends on the input range and resolution mode.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: -). When module error (ERR) is TRUE, module error is cleared by setting ERRC to TRUE. ERR will become FALSE accordingly.	TRUE, FALSE	FALSE	User
	CH1INH to CH4INH	Public variable	BOOL	Enable/Disable A/D conversion (CH1 to CH4) (TRUE: Enabled FALSE: Disabled). Set whether enable/disable A/D conversion value output of per channel. It is valid when STB transforms FALSE TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute A/D conversion enable/disable setting (CH1INH to CH4INH) when STB changes from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	MRES	Public variable	BOOL	Maximum/minimum value reset request. Maximum/minimum value (CH1MAX to CH4MAX, CH1MIN to CH4MIN) is cleared when MRES change from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). The status of module READY (Xn0) is stored. Execute A/D conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CNVCMPL	Public variable	BOOL	A/D conversion completed flag (TRUE: ON FALSE: OFF). Store the status of A/D conversion completed flag (XnE).	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: no error). Store TRUE when error occurs in writing.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	INT	A/D conversion enabled/disabled setting status. Store A/D conversion enabled/disabled setting status. b15 to b3 b2 b1 b0 [ ] CH 4 | CH 3 | CH 2 | CH 1 0: Enable A/D conversion 1: Disable A/D conversion	0 to 15	0	System
	ERRCOD	Public variable	INT	Error code. Store error code detected by A/D conversion module. Refer to Analog-Digital Converter Module User's Manual for details about error codes.	–	0	System
	CH1MAX to CH4MAX	Public variable	INT	Maximum value of CH1 to CH4. Store the maximum of digital values according to channels.	Depends on the input range and resolution mode.	0	System
	CH1MIN to CH4MIN	Public variable	INT	Minimum value of CH1 to CH4. Store the minimum of digital values according to channels.	Depends on the input range and resolution mode.	0	System

*1 The public variables CH1INH to CN4INH can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH4)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>A/D conversion completed flag (XnE)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read digital output value from A/D conversion module and output the digital output value of each channel from output variable CH1 to CH4.</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH4 remains the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Others</td><td>For information about the processing and setting of corresponding module of this module FB, refer to the following manual:
Analog-Digital Converter Module User's Manual.</td></tr>
</table>

POINT

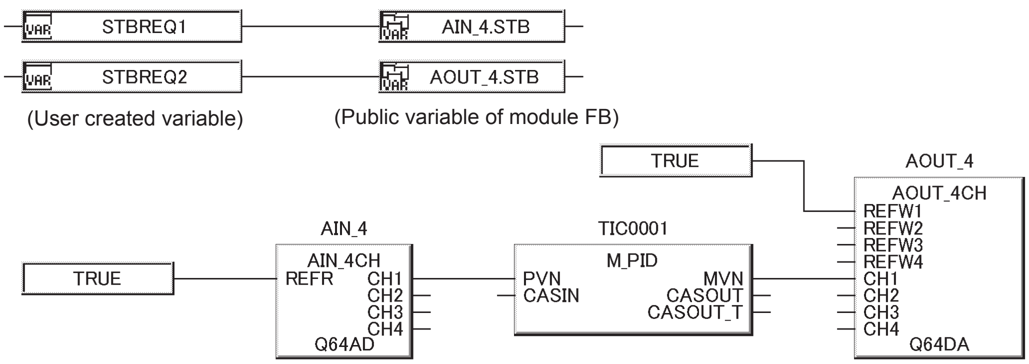
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-AD is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

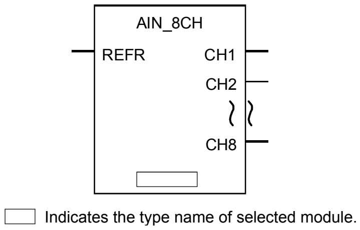
- Unable to communicate with A/D conversion module. (Error code: 1412)
- Abnormality of A/D conversion module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



When STBREQ1 is TRUE, operation condition setting request of AIN_4 (STB) will be executed.
When STBREQ2 is TRUE, operation condition setting request of AOUT_4 (STB) will be executed.

10.1.2 8 Channels Analog Input (AIN_8CH)

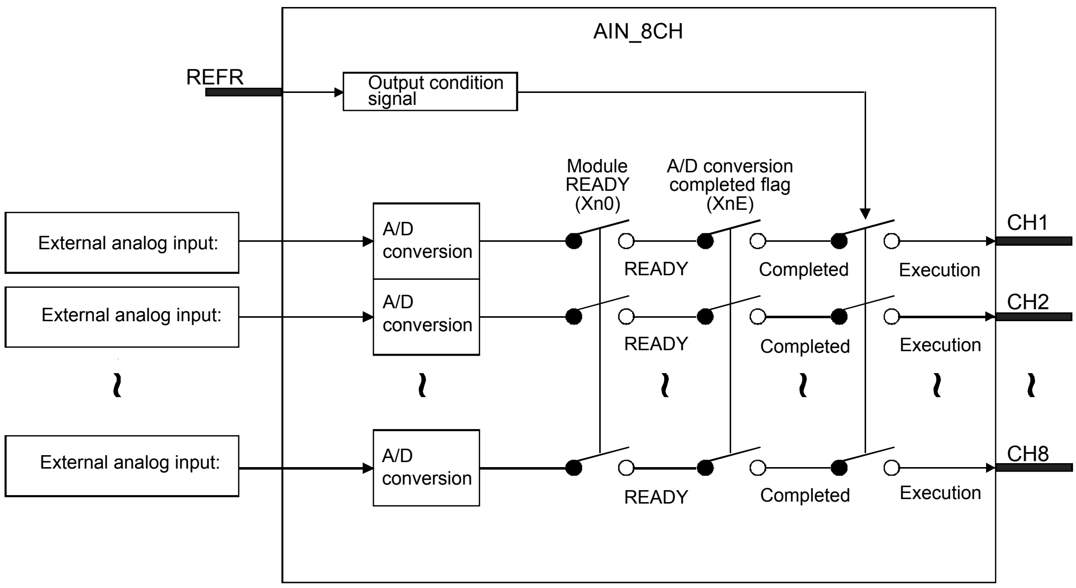
FB	FBD parts
AIN_8CH	

Corresponding module
Q68ADV, Q68ADI

Function overview: Reads the digital output value of 8 channels A/D conversion module that converts analog signal to digital value, and outputs it from output variable (CH1 to CH8).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH8	Output variable	REAL	CH1 to CH8 digital output value	Depends on the input range and resolution mode.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: -). When module error (ERR) is TRUE, module error is cleared by setting ERRC to TRUE. ERR will become FALSE accordingly.	TRUE, FALSE	FALSE	User
	CH1INH to CH8INH	Public variable	BOOL	Enable/disable A/D conversion (CH1 to CH8) (TRUE: Enabled FALSE: Disabled). Set whether enable/disable A/D conversion value output based on channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute A/D conversion enabled/disabled setting (CH1INH to CH8INH) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	MRES	Public variable	BOOL	Maximum/minimum value reset request. Maximum/minimum value (CH1MAX to CH8MAX, CH1MIN to CH8MIN) is cleared when MRES changes from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). The status of module READY (Xn0) is stored. Execute A/D conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CNVCMPL	Public variable	BOOL	A/D conversion completed flag (TRUE: ON FALSE: OFF). Store the status of A/D conversion completion flag (XnE).	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: No error). Store TRUE when error occurs in writing.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	INT	A/D conversion enabled/disabled setting status. Store A/D conversion enabled/disabled setting status. b15 to b7 / b6 / b5 / b4 / b3 / b2 / b1 / b0 (blank) / CH 8 / CH 7 / CH 6 / CH 5 / CH 4 / CH 3 / CH 2 / CH 1 0: Enable A/D conversion 1: Disable A/D conversion	0 to 255	0	System
	ERRCOD	Public variable	INT	Error code. Store error code detected by A/D conversion module. Refer to Analog-Digital Converter Module User's Manual for details about error codes.	–	0	System
	CH1MAX to CH8MAX	Public variable	INT	Maximum value of CH1 to CH8. Store the maximum digital value based on channels.	Depends on the input range and resolution mode.	0	System
	CH1MIN to CH8MIN	Public variable	INT	Minimum value of CH1 to CH8. Store the minimum digital value based on channels.	Depends on the input range and resolution mode.	0	System

*1 The public variables CH1INH to CN8INH can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH8)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>A/D conversion completed flag (XnE)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read digital output value from A/D conversion module and output digital output value of each channel from output variable CH1 to CH8.</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH8 holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Others</td><td>For information about the processing and setting of corresponding module of this module FB, refer to the following manual: Analog-Digital Converter Module User's Manual.</td></tr>
</table>

POINT

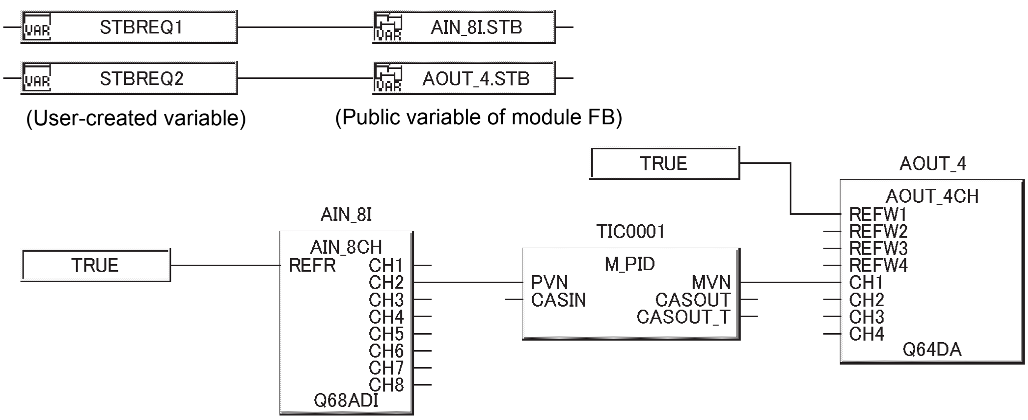
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-AD is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

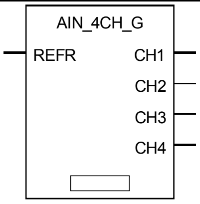
- Unable to communicate with A/D conversion module. (Error code: 1412)
- Abnormality of A/D conversion module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request of AIN_8I (STB) will be executed.
If STBREQ2 is TRUE, operation condition setting request of AOUT_4 (STB) will be executed.

10.1.3 Channel-isolated 4 Channels Analog Input (AIN_4CH_G)

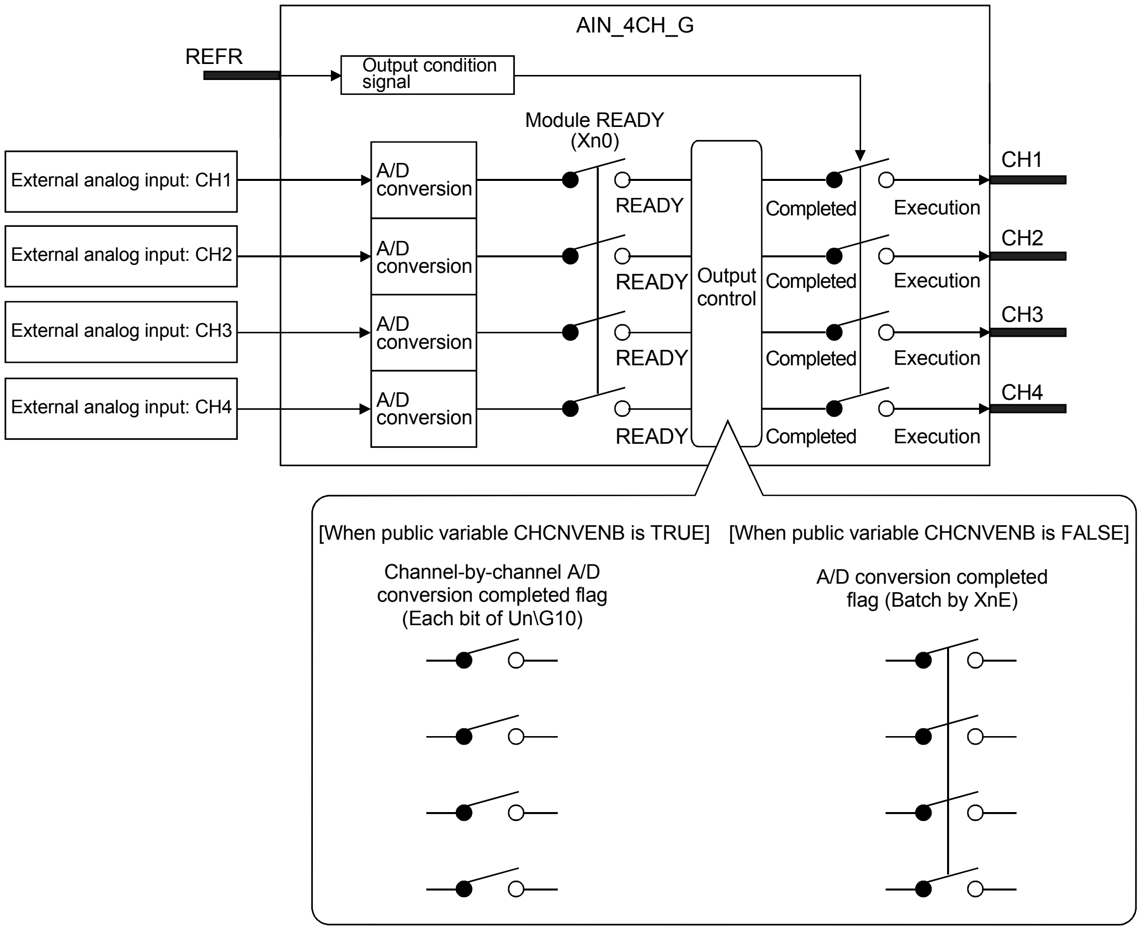
FB	FBD parts
AIN_4CH_G	 ▭ Indicates the type name of selected modul

Corresponding module
Q64AD-GH

Function overview: Reads the digital output value of channel-isolated A/D conversion module (4 channels) that converts analog signal to digital value, and outputs it from output variable (CH1 to CH4).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH4	Output variable	REAL	CH1 to CH4 digital output value	Depends on the input range and resolution mode.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: -). When occurring module error (ERR) and input signal abnormality detection signal (SYSAL) are TRUE, set ERRC to TRUE to clear the module error and the input signal abnormality, and ERR and SYSAL will become FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH4INH	Public variable	BOOL	Enable/disable A/D conversion (CH1 to CH4) (TRUE: Enabled FALSE: Disabled). Set whether enable/disable A/D conversion value output of each channel. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL4ENB	Public variable	BOOL	Enable/disable alarm output (CH1 to CH4) (TRUE: Disabled FALSE: Enabled). Set whether enable/disable alarm output of each channel. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute A/D conversion enabled/disabled setting (CH1INH to CH4INH) and alarm output enable/disable setting (AL1ENB to AL4ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	MRES	Public variable	BOOL	Maximum/minimum value reset request. Maximum/minimum value (CH1MAX to CH4MAX, CH1MIN to CH4MIN) is cleared when MRES turns from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). The status of module READY (Xn0) is stored. Execute A/D conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the digital output value. When the setting is TRUE, the channel-by-channel A/D conversion completed flag is used as the output condition. When the setting is FALSE, the A/D conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	A/D conversion completed flag (TRUE: ON FALSE: OFF). Store the status of A/D conversion completion flag (XnE).	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: no error). Store TRUE when error occurs in writing.	TRUE, FALSE	FALSE	System
	SYSAL	Public variable	BOOL	Input signal abnormality detection signal. (TRUE: abnormal FALSE: normal). Store TRUE once input signal abnormality occurs in any of CH1 to CH4 of A/D conversion module.	TRUE, FALSE	FALSE	System
	PRCAL	Public variable	BOOL	Alarm output signal (TRUE: alarm FALSE: normal). Store TRUE once process or rate alarm occurs in any of CH1 to CH4 of A/D conversion module.	TRUE, FALSE	FALSE	System
	PLAL1 to PLAL4	Public variable	BOOL	Low limit value alarm of CH1 to CH4 process alarm (TRUE: over FALSE: normal). Store TRUE of the channel if it surpasses the low limit value of process alarm.	TRUE, FALSE	FALSE	System
	PHAL1 to PHAL4	Public variable	BOOL	High limit value alarm of CH1 to CH4 process alarm (TRUE: over FALSE: normal). Store TRUE of the channel if it surpasses the high limit value of process alarm.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	RTLAL1 to RTLAL4	Public variable	BOOL	Low limit alarm of CH1 to CH4 rate alarm (TRUE: over FALSE: normal) Store TRUE of the channel if it surpasses the low limit value of rate alarm.	TRUE, FALSE	FALSE	System
	RTHAL1 to RTHAL4	Public variable	BOOL	High limit alarm of CH1 to CH4 rate alarm (TRUE: over FALSE: normal) Store TRUE of the channel if it surpasses the high limit value of rate alarm.	TRUE, FALSE	FALSE	System
	RGAL1 to RGAL4	Public variable	BOOL	CH1 to CH4 input signal abnormality detection (TRUE: abnormal FALSE: normal) Store TRUE of the channel if it surpasses the setting range of I/O signal abnormality detection setting value.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	INT	A/D conversion enabled/disabled setting status. Store A/D conversion enabled/disabled setting status b15 to b3 b2 b1 b0 CH 4 | CH 3 | CH 2 | CH 1 0: Enable A/D conversion 1: Disable A/D conversion	0 to 15	0	System
	ERRCOD	Public variable	INT	Error code. Store error code detected by A/D conversion module. Refer to Channel Isolated High Resolution Analog-Digital Converter Module, Channel Isolated High Resolution Analog-Digital Converter Module (with Signal Conditioning Function) User's Manual.	−	0	System
	ALMENB	Public variable	INT	Alarm output enabled/disabled setting status. Store alarm output enabled/disabled setting status b15 to b3 b2 b1 b0 CH 4 | CH 3 | CH 2 | CH 1 0: Enable alarm output 1: Disable alarm output	0 to 15	0	System
	CH1MAX to CH4MAX	Public variable	DINT	Maximum A/D conversion value of CH1 to CH4. Store the maximum digital value based on channels.	Depends on the input range and resolution mode.	0	System
	CH1MIN to CH4MIN	Public variable	DINT	Minimum A/D conversion value of CH1 to CH4. Store the minimum digital value based on channels.	Depends on the input range and resolution mode.	0	System

*1 The public variables CH1INH to CH4INH, AL1ENB to AL4ENB, or CHCNVENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH4)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read digital output value from A/D conversion module and output digital output value of each channel from output variable CH1 to CH4.</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH4 holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel A/D conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: A/D conversion completed flag (batch by XnE)
</td></tr>
<tr><td>Others</td><td>For information about the processing, setting, channel-by-channel A/D conversion flag and A/D conversion completed flag of corresponding module of this module FB, refer to the following manual:
● Channel Isolated High Resolution Analog-Digital Converter Module, Channel Isolated High Resolution Analog-Digital Converter Module (with Signal Conditioning Function) User's Manual.</td></tr>
</table>

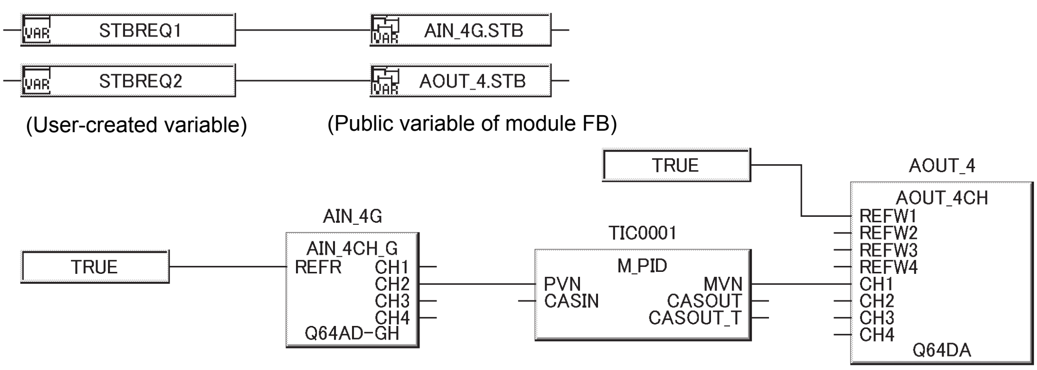
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-AD is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

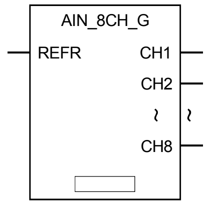
- Unable to communicate with A/D conversion module. (Error code: 1412)
- Abnormality of A/D conversion module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request of AIN_4G (STB) will be executed.
If STBREQ2 is TRUE, operation condition setting request of AOUT_4 (STB) will be executed.

10.1.4 Channel-isolated 8 Channels Analog Input (AIN_8CH_G)

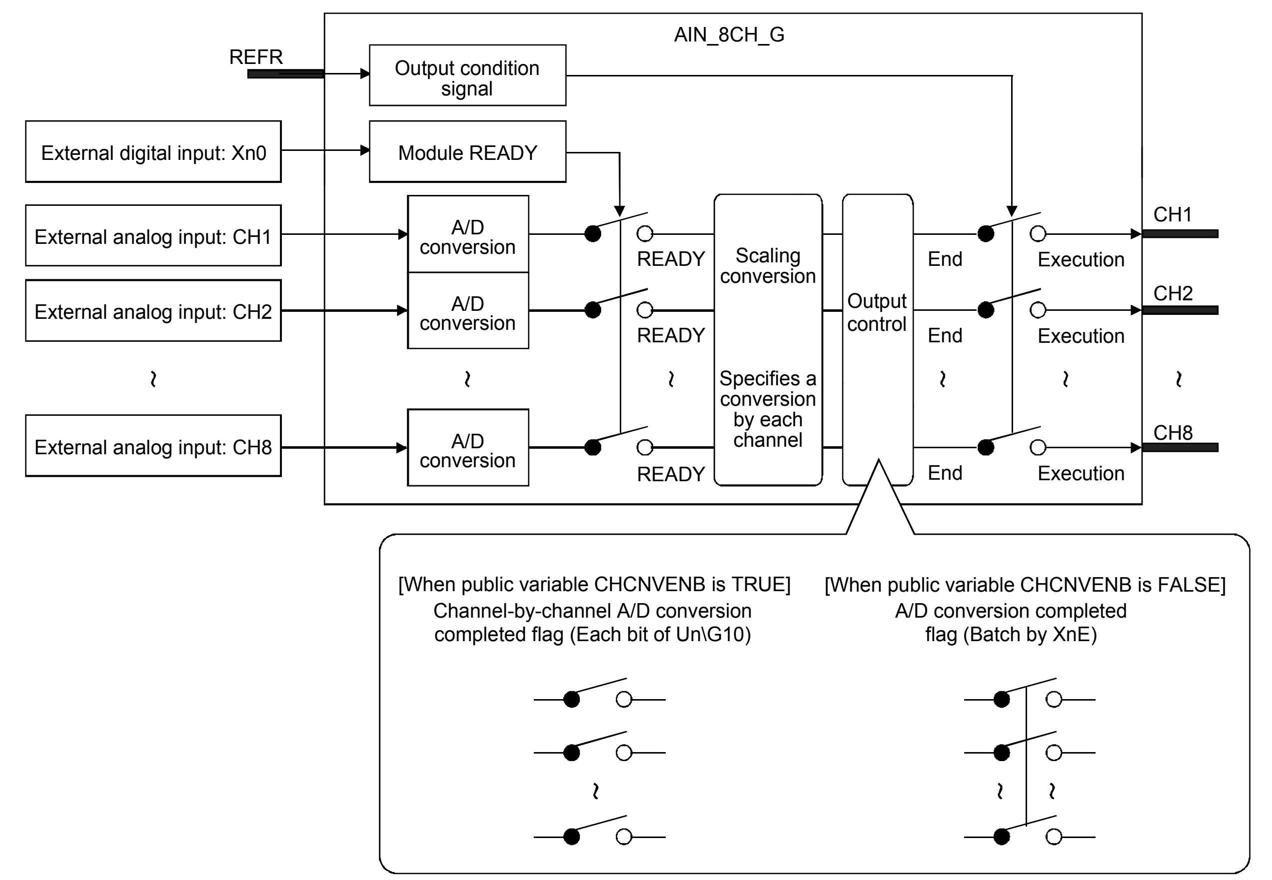
FB	FBD parts
AIN_8CH_G	 Indicates the type name of selected module.

Corresponding module
Q68AD-G

Function overview: Reads digital output values or scaling values of channel-isolated A/D converter module (8 channels), which converts analog signals to digital values, and outputs them from output variables (CH1 to CH8).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH8	Output variable	REAL	CH1 to CH8 output value (*1)	Depends on the input range, resolution mode, and scaling function of the module.

*1 With the scaling enable/disable setting, either digital output values or scaling values are selected for each channel and are output from the output values of CH1 to CH8.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: -). When occurring module error (ERR) and input signal abnormality detection signal (SYSAL) are TRUE, set ERRC to TRUE to clear the module error and the input signal abnormality, and ERR and SYSAL will become FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH8INH	Public variable	BOOL	Enable/disable A/D conversion (CH1 to CH8) (TRUE: Enabled FALSE: Disabled). Set whether enable/disable A/D conversion value output of each channel. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL8ENB	Public variable	BOOL	Enable/disable alarm output (CH1 to CH8) (TRUE: Disabled FALSE: Enabled). Set whether enable/disable alarm output of each channel. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute A/D conversion enabled/disabled setting (CH1INH to CH8INH) and alarm output enable/disable setting (AL1ENB to AL8ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	MRES	Public variable	BOOL	Maximum/minimum value reset request. Maximum/minimum value (CH1MAX to CH8MAX, CH1MIN to CH8MIN) is cleared when MRES turns from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). The status of module READY (Xn0) is stored. Execute A/D conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the digital output value. When the setting is TRUE, the channel-by-channel A/D conversion completed flag is used as the output condition. When the setting is FALSE, the A/D conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	A/D conversion completed flag (TRUE: ON FALSE: OFF). Store the status of A/D conversion completion flag (XnE).	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: no error). Store TRUE when error occurs in writing.	TRUE, FALSE	FALSE	System
	SYSAL	Public variable	BOOL	Input signal abnormality detection signal. (TRUE: abnormal FALSE: normal). Store TRUE once input signal abnormality occurs in any of CH1 to CH4 of A/D conversion module.	TRUE, FALSE	FALSE	System
	PRCAL	Public variable	BOOL	Alarm output signal (TRUE: alarm FALSE: normal). Store TRUE once process or rate alarm occurs in any of CH1 to CH8 of A/D conversion module.	TRUE, FALSE	FALSE	System
	PLAL1 to PLAL8	Public variable	BOOL	Low limit value alarm of CH1 to CH8 process alarm (TRUE: over FALSE: normal). Store TRUE of the channel if it surpasses the low limit value of process alarm.	TRUE, FALSE	FALSE	System
	PHAL1 to PHAL8	Public variable	BOOL	High limit value alarm of CH1 to CH8 process alarm (TRUE: over FALSE: normal). Store TRUE of the channel if it surpasses the high limit value of process alarm.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	RTLAL1 to RTLAL8	Public variable	BOOL	Low limit alarm of CH1 to CH8 rate alarm (TRUE: over FALSE: normal). Store TRUE of the channel if it surpasses the low limit value of rate alarm.	TRUE, FALSE	FALSE	System
	RTHAL1 to RTHAL8	Public variable	BOOL	High limit value alarm of CH1 to CH8 rate alarm (TRUE: over FALSE: normal). Store TRUE of the channel if it surpasses the high limit value of rate alarm.	TRUE, FALSE	FALSE	System
	RGAL1 to RGAL8	Public variable	BOOL	CH1 to CH8 input signal abnormality detection (TRUE: abnormal FALSE: normal). Store TRUE of the channel if it surpasses the setting range of I/O signal abnormality detection setting value.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	WORD	A/D conversion enabled/disabled setting status. Store A/D conversion enabled/disabled setting status b15 to b7 b3 b2 b1 b0 | 0 ··· 0 | CH 8 | ··· | CH 4 | CH 3 | CH 2 | CH 1 | 0: Enable A/D conversion 1: Disable A/D conversion	0 to 00FFH	0	System
	ERRCOD	Public variable	INT	Error code. Store error code detected by A/D conversion module. For detailed information about the error code, refer to Channel Isolated High Resolution Analog-Digital Converter Module/Channel Isolated High Resolution Analog-Digital Converter Module (With Signal Conditioning Function) User's Manual.	–	0	System
	ALMENB1	Public variable	WORD	Alarm output enabled/disabled setting status. Stores alarm output enabled/disabled setting status. b15 to b10 b9 b8 b7 to b2 b1 b0 | CH 8 | ··· | CH 3 | CH 2 | CH 1 | CH 8 | ··· | CH 3 | CH 2 | CH 1 | b0 to b7: Process alarm setting b8 to b15: Rate alarm setting 0: Enable alarm output 1: Disable alarm output	0 to FFFFH	FFFFH	System
	ALMENB2	Public variable	WORD	Input signal error detection enabled/disabled setting status. Stores an input signal error detection enabled/disabled setting status. b15 to b7 b3 b2 b1 b0 | 0 ··· 0 | CH 8 | ··· | CH 4 | CH 3 | CH 2 | CH 1 | 0: Enables fault detection 1: Disables fault detection	0 to 00FFH	00FFH	System

<table>
<tr><th></th><th>Variable name</th><th>Variable type</th><th>Data type</th><th>Contents</th><th>Range</th><th>Initial value</th><th>Storage</th></tr>
<tr><td rowspan="4">Variable processing</td><td>SCLENB</td><td>Public variable</td><td>WORD</td><td>Scaling enabled/disabled setting status.
Store scaling enabled/disabled setting status.
<table><tr><td>b15</td><td>to</td><td>b7</td><td></td><td>b3</td><td>b2</td><td>b1</td><td>b0</td></tr><tr><td>0</td><td>...</td><td>0</td><td>CH 8</td><td>...</td><td>CH 4</td><td>CH 3</td><td>CH 2</td><td>CH 1</td></tr></table>0: Valid
1: Invalid</td><td>0 to 00FFH</td><td>00FFH</td><td>System</td></tr>
<tr><td>CH1DOUT to CH8DOUT</td><td>Public variable</td><td>INT</td><td>CH1 to CH8 Digital output value.
Stores digital output values of each channel.</td><td>Depends on input range and resolution mode.</td><td>0</td><td>System</td></tr>
<tr><td>CH1MAX to CH8MAX</td><td>Public variable</td><td>INT</td><td>Maximum A/D conversion value of CH1 to CH8.
Stores the maximum output value of each channel. (*2)</td><td>Depends on the input range, resolution mode, and scaling function of the module.</td><td>0</td><td>System</td></tr>
<tr><td>CH1MIN to CH8MIN</td><td>Public variable</td><td>INT</td><td>Minimum A/D conversion value of CH1 to CH8.
Stores the minimum output value of each channel. (*2)</td><td>Depends on the input range, resolution mode, and scaling function of the module.</td><td>0</td><td>System</td></tr>
</table>

*1 The public variables CH1INH to CH8INH, AL1ENB to AL8ENB, or CHCNVENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

*2 With the scaling enable/disable setting, either digital output values or scaling value is selected for each channel and output from the output values of CH1 to CH8. The storage value is also switched between the maximum value and minimum value with the setting.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td><table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH4)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read digital output value from A/D conversion module and output digital output value of each channel from output variable CH1 to CH8.</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH8 holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel A/D conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: A/D conversion completed flag (batch by XnE)</td></tr>
<tr><td>Others</td><td>For information about the processing, setting, Channel-by-channel A/D conversion flag and A/D conversion completed flag of corresponding module of this module FB, refer to the following manual:
● Channel Isolated Analog-Digital Converter Module/Channel Isolated Analog-Digital Converter Module (With Signal Conditioning Function) User's Manual.</td></tr>
</table>

POINT

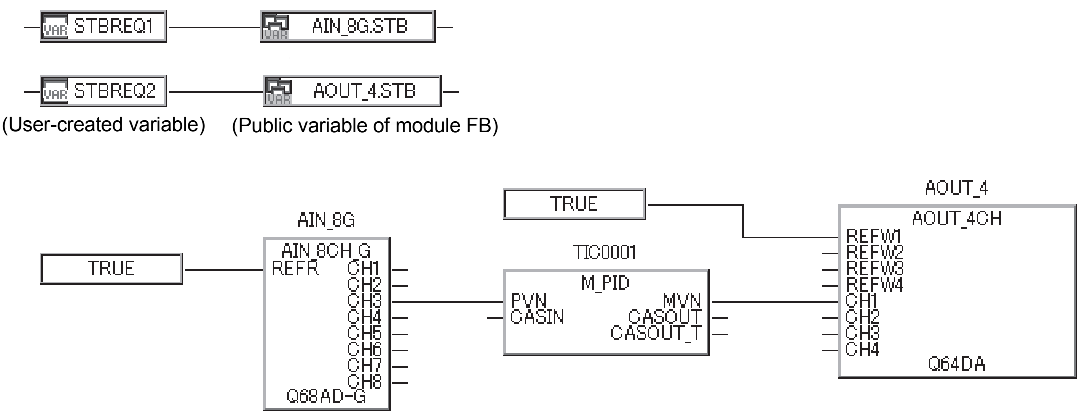
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-AD is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

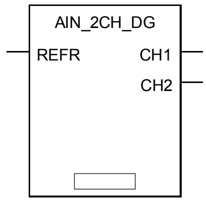
- Unable to communicate with A/D conversion module. (Error code: 1412)
- Abnormality of A/D conversion module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request of AIN_8G (STB) will be executed.
If STBREQ2 is TRUE, operation condition setting request of AOUT_4 (STB) will be executed.

10.1.5 Channel-isolated High-resolution 2 Channels Signal Condition Function (AIN_2CH_DG)

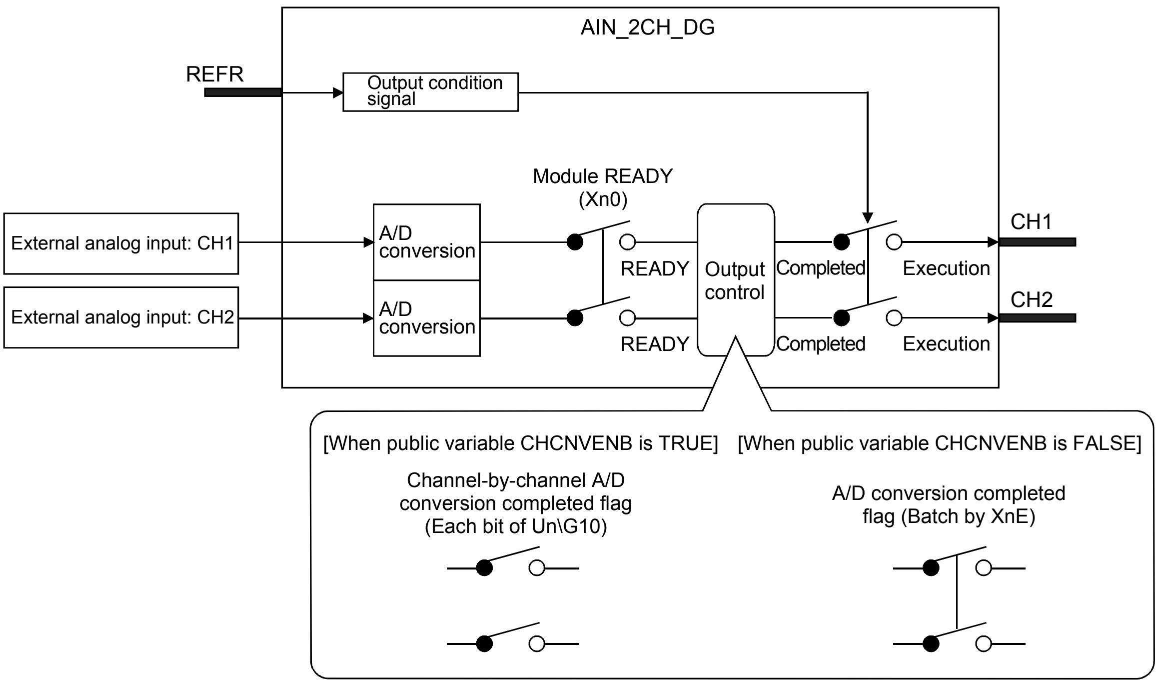
FB	FBD parts
AIN_2CH_DG	 Indicates the type name selected module.

Corresponding module
Q62AD-DGH

Function overview: Reads the digital output value of channel-isolated (2 channels) high resolution signal conditioning function that converts analog signal to digital value, and output it from output variable (CH1 to CH2)

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
Output	CH1, CH2	Output variable	REAL	CH1, CH2 digital output value	Depends on the resolution mode of module

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: −). When occurring module error (ERR) and input signal abnormality detection signal (SYSAL) are TRUE, set ERRC to TRUE to clear the module error and the input signal abnormality, and ERR and SYSAL will become FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH2INH	Public variable	BOOL	Enable/disable A/D conversion (CH1 to CH2). (TRUE: Conversion enabled FALSE: Conversion disabled) Set whether enable/disable A/D conversion based on channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL2ENB	Public variable	BOOL	Enable/disable alarm output (CH1 to CH2). (TRUE:Output enabled FALSE:Output disabled) Set alarm output enabled/disabled based on channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute A/D conversion enabled/disabled setting (CH1INH to CH2INH) and alarm output enabled/disabled setting (AL1ENB to AL2ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	MRES	Public variable	BOOL	Maximum/minimum value reset request. Reset maximum/minimum value (CH1MAX to CH2MAX, CH1MIN to CH2MIN) when MRES transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). Store the status of module READY (Xn0). Execute A/D conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the digital output value. When the setting is TRUE, the channel-by-channel A/D conversion completed flag is used as the output condition. When the setting is FALSE, the A/D conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	A/D conversion completed flag (TRUE: ON FALSE: OFF). Store the status of A/D conversion completion signal (Xn0).	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error tag (TRUE: Error FALSE: No error). Store TRUE when error occurs in writing.	TRUE, FALSE	FALSE	System
	SYSAL	Public variable	BOOL	Input signal abnormality detection signal (TRUE: Abnormal FALSE: Normal). Store TRUE when input signal abnormality occurs in either CH1 or CH2 of high-resolution signal condition function.	TRUE, FALSE	FALSE	System
	PRCAL	Public variable	BOOL	Alarm output signal (TRUE to Alarm occurs FALSE to Normal). Store TRUE when procedure alarm or rate alarm occurs on CH1 or CH2 of high-resolution signal condition function.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	PLAL1 to PLAL2	Public variable	BOOL	CH1 to CH2 low limit alarm of process alarm (TRUE: Exceed FALSE: Normal) Store TRUE in each channel when exceeding the low limit.	TRUE, FALSE	FALSE	System
	PHAL1 to PHAL2	Public variable	BOOL	CH1 to CH2 high limit alarm of process alarm (TRUE: Exceed FALSE: Normal) Store TRUE in each channel when exceeding the high limit.	TRUE, FALSE	FALSE	System
	RTLAL1 to RTLAL2	Public variable	BOOL	CH1 to CH2 low limit alarm of rate alarm (TRUE: Exceed FALSE: Normal) Store TRUE in each channel when exceeding the low limit of rate alarm.	TRUE, FALSE	FALSE	System
	RTHAL1 to RTHAL2	Public variable	BOOL	CH1 toCH2 high limit alarm of rate alarm (TRUE: Exceed FALSE: Normal) Store TRUE in each channel when exceeding the high limit of rate alarm	TRUE, FALSE	FALSE	System
	RGAL1 to RGAL2	Public variable	BOOL	CH1 toCH2 input signal error detection (TRUE: Abnormal FALSE: Normal) Store TRUE in each channel when exceeding the setting value range of input/output abnormality detection.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	INT	Setting status of A/D conversion enabled/disabled. Store the setting status of A/D conversion enabled/disabled. b15 to b1 b0 [ ] [CH 2] [CH 1] 0: Enable A/D conversion 1: Disable A/D conversion	0 to 3	3	System
	ERRCOD	Public variable	INT	Error code Store the code of error that is detected by high-resolution signal condition function. For detailed information about the error code, refer to Channel Isolated High Resolution Analog-Digital Converter Module, Channel Isolated High Resolution Analog-Digital Converter Module (with Signal Conditioning Function) User's Manual	–	0	System
	ALMENB	Public variable	INT	Setting status of alarm output enabled/disabled. Store the setting status of alarm output enabled/disabled. b15 to b9 b8 b5 b4 to b1 b0 [ ] [CH 2] [CH 1] [ ] [CH 2] [CH 1] [ ] [CH 2] [CH 1] b0, b1: Process alarm setting b4, b5: Rate alarm setting b8, b9: Input signal abnormality detection 0: Enable alarm output 1: Disable alarm output	0 to 819 (333_H)	819 (333_H)	System
	CH1MAX to CH2MAX	Public variable	DINT	Maximum value of CH1 to CH2 A/D conversion Store the maximum value of data value in each channel.	The range of module resolution.	0	System
	CH1MIN to CH2MIN	Public variable	DINT	Minimum value of CH1 to CH2 A/D conversion Store the minimum value of data value in each channel.	The range of module resolution.	0	System

*1 The public variables CH1INH to CH2INH, AL1ENB to AL2ENB, or CHCNVENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH2)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read digital output value from high-resolution signal conditioning function and output digital output value from output variable CH1 to CH2.</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH2 holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel A/D conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: A/D conversion completed flag (batch by XnE)
</td></tr>
<tr><td>Others</td><td>For information about the processing, setting, Channel-by-channel A/D conversion flag and A/D conversion completed flag of corresponding module of this module FB, refer to the following manual.
● Channel Isolated High Resolution Analog-Digital Converter Module, Channel Isolated High Resolution Analog-Digital Converter Module (with Signal Conditioning Function) User's Manual.</td></tr>
</table>

POINT

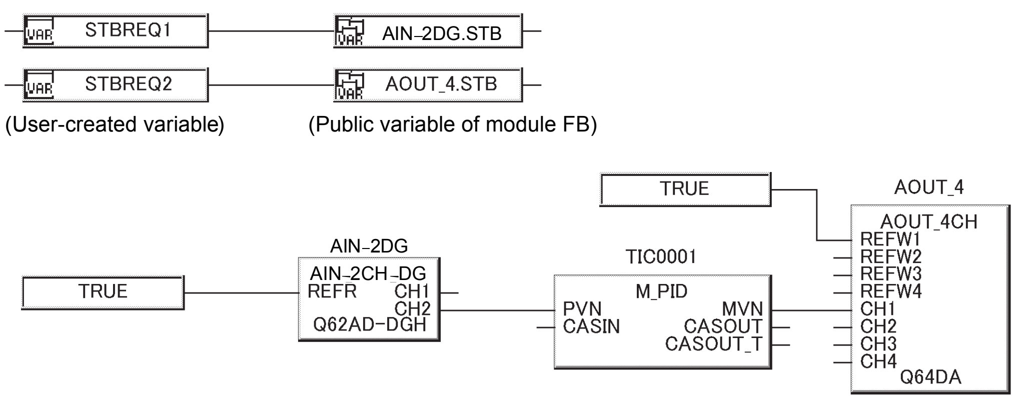
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-AD is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

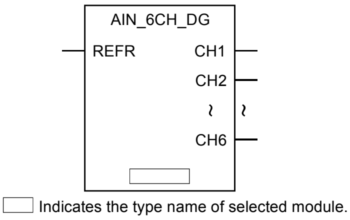
- Unable to communicate with high-resolution signal conditioning module. (Error code: 1412)
- Abnormality of high-resolution conditioning module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request of AIN_2DG (STB) will be executed.
If STBREQ2 is TRUE, operation condition setting request of AOUT_4 (STB) will be executed.

10.1.6 Channel-isolated 6 Channels A/D Converter Module with Signal Conditioning Function (AIN_6CH_DG)

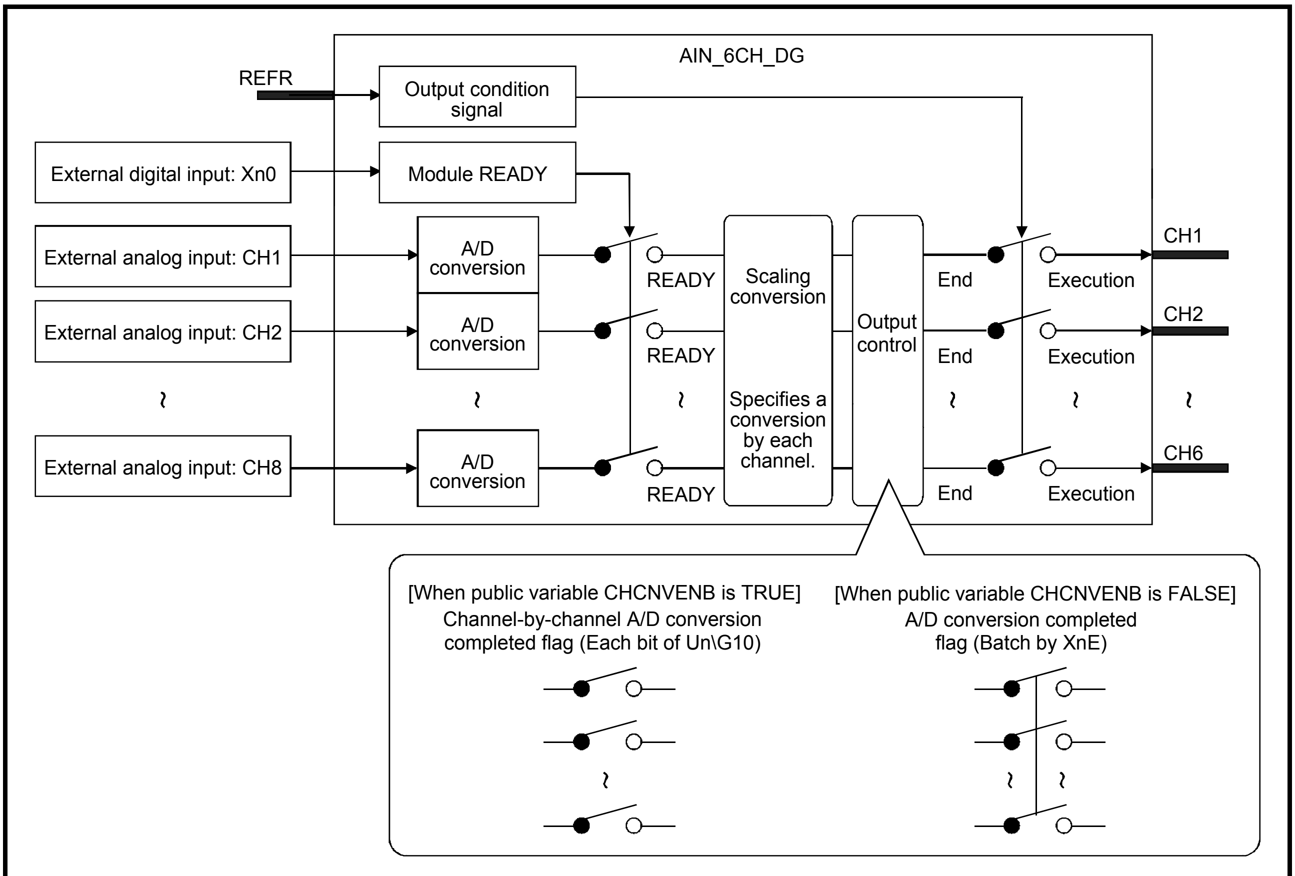
FB	FBD parts
AIN_6CH_DG	

Corresponding module
Q66AD-DG

Function overview: Reads digital output values or scaling values of channel isolated A/D converter module with signal conditioning function (6 channels), which converts analog signals to digital values, and outputs them from output variables (CH1 to CH6).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH6	Output variable	REAL	CH1 to CH8 Output value (*1)	Depends on the input range, resolution mode, and scaling function of the module.

*1 With the scaling enable/disable setting, either digital output values or scaling values are selected for each channel and are output from the output values of CH1 to CH6.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: −). When occurring module error (ERR) and input signal abnormality detection signal (SYSAL) are TRUE, set ERRC to TRUE to clear the module error and the input signal abnormality, and ERR and SYSAL will become FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH6INH	Public variable	BOOL	Enable/disable A/D conversion (CH1 to CH6). (TRUE: Conversion enabled FALSE: Conversion disabled) Set whether enable/disable A/D conversion based on channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL6ENB	Public variable	BOOL	Enable/disable alarm output (CH1 to CH6). (TRUE:Output enabled FALSE:Output disabled) Set alarm output enabled/disabled based on channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute A/D conversion enabled/disabled setting (CH1INH to CH6INH) and alarm output enabled/disabled setting (AL1ENB to AL6ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	MRES	Public variable	BOOL	Maximum/minimum value reset request. Reset maximum/minimum value (CH1MAX to CH6MAX, CH1MIN to CH6MIN) when MRES transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). Store the status of module READY (Xn0). Execute A/D conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the digital output value. When the setting is TRUE, the channel-by-channel A/D conversion completed flag is used as the output condition. When the setting is FALSE, the A/D conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	A/D conversion completed flag (TRUE: ON FALSE: OFF). Store the status of A/D conversion completion signal (Xn0).	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error tag (TRUE: Error FALSE: No error). Store TRUE when error occurs in writing.	TRUE, FALSE	FALSE	System
	SYSAL	Public variable	BOOL	Input signal abnormality detection signal (TRUE: Abnormal FALSE: Normal). Store TRUE when input signal abnormality occurs in either CH1 or CH6 of high-resolution signal condition function.	TRUE, FALSE	FALSE	System
	PRCAL	Public variable	BOOL	Alarm output signal (TRUE to Alarm occurs FALSE to Normal). Store TRUE when procedure alarm or rate alarm occurs on CH1 or CH6 of high-resolution signal condition function.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	PLAL1 to PLAL6	Public variable	BOOL	CH1 to CH6 low limit alarm of process alarm (TRUE: Exceed FALSE: Normal). Store TRUE in each channel when exceeding the low limit.	TRUE, FALSE	FALSE	System
	PHAL1 to PHAL6	Public variable	BOOL	CH1 to CH6 high limit alarm of process alarm (TRUE: Exceed FALSE: Normal). Store TRUE in each channel when exceeding the high limit.	TRUE, FALSE	FALSE	System
	RTLAL1 to RTLAL6	Public variable	BOOL	CH1 to CH6 low limit alarm of rate alarm (TRUE: Exceed FALSE: Normal). Store TRUE in each channel when exceeding the low limit of rate alarm.	TRUE, FALSE	FALSE	System
	RTHAL1 to RTHAL6	Public variable	BOOL	CH1 toCH6 high limit alarm of rate alarm (TRUE: Exceed FALSE: Normal). Store TRUE in each channel when exceeding the high limit of rate alarm	TRUE, FALSE	FALSE	System
	RGAL1 to RGAL6	Public variable	BOOL	CH1 toCH6 input signal error detection (TRUE: Abnormal FALSE: Normal). Store TRUE in each channel when exceeding the setting value range of input/output abnormality detection.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	WORD	Setting status of A/D conversion enabled/disabled. Store the setting status of A/D conversion enabled/disabled. b15 to b5 b4 b3 b2 b1 b0 0 ··· 0 CH 6 CH 5 CH 4 CH 3 CH 2 CH 1 0: Enable A/D conversion 1: Disable A/D conversion	0 to 003FH	003FH	System
	ERRCOD	Public variable	INT	Error code. Store the code of error that is detected by high-resolution signal condition function. For detailed information about the error code, refer to Channel Isolated High Resolution Analog-Digital Converter Module/Channel Isolated High Resolution Analog-Digital Converter Module (With Signal Conditioning Function) User's Manual.	–	0	System
	ALMENB1	Public variable	WORD	Setting status of alarm output enabled/disabled. Store the setting status of alarm output enabled/disabled. b15 to b13 to b9 b8 b5 to b1 b0 0 0 CH 6 ··· CH 2 CH 1 0 0 CH 6 ··· CH 2 CH 1 b0 to b5: Process alarm setting b8 to b13: Rate alarm setting 0: Enable alarm output 1: Disable alarm output	0 to 3F3FH	3F3FH	System
	ALMENB2	Public variable	WORD	Input signal error detection enabled/disabled setting status. Stores an input signal error detection enabled/disabled setting status. b15 to b5 b4 b3 b2 b1 b0 0 ··· 0 CH 6 CH 5 CH 4 CH 3 CH 2 CH 1 0: Enables fault detection. 1: Disables fault detection.	0 to 003FH	003FH	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	SCLENB	Public variable	WORD	Scaling enabled/disabled setting status. Stores scaling enabled/disabled setting status. b15 to b5 b4 b3 b2 b1 b0 0 ··· 0 | CH 6 | CH 5 | CH 4 | CH 3 | CH 2 | CH 1 0: Valid 1: Invalid	0 to 003FH	003FH	System
	CH1DOUT to CH6DOUT	Public variable	INT	CH1 to CH6 Digital output value. Stores digital output values of each channel.	Depends on the range of module resolution.	0	System
	CH1MAX to CH6MAX	Public variable	INT	Maximum A/D conversion value of CH1 to CH6. Stores the maximum output value of each channel. (*2)	Depends on the range of module resolution and scaling function.	0	System
	CH1MIN to CH6MIN	Public variable	INT	Minimum A/D conversion value of CH1 to CH6. Stores the minimum output value of each channel. (*2)	Depends on the range of module resolution and scaling function.	0	System

*1 The public variables CH1INH to CH6INH, AL1ENB to AL6ENB, or CHCNVENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

*2 With the scaling enable/disable setting, either digital output values or scaling value is selected for each channel and output from the output values of CH1 to CH6. The storage value is also switched between the maximum value and minimum value with the setting.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH6)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Reads digital output values from an A/D converter module and outputs them by channels from output variables CH1 to CH6.</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH2 holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel A/D conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: A/D conversion completed flag (batch by XnE)
</td></tr>
<tr><td>Others</td><td>For information about the processing, setting, Channel-by-channel A/D conversion flag and A/D conversion completed flag of corresponding module of this module FB, refer to the following manual.
● Channel Isolated Analog-Digital Converter Module/Channel Isolated Analog-Digital Converter Module (With Signal Conditioning Function) User's Manual.</td></tr>
</table>

POINT

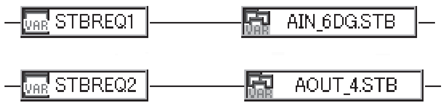
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-AD is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

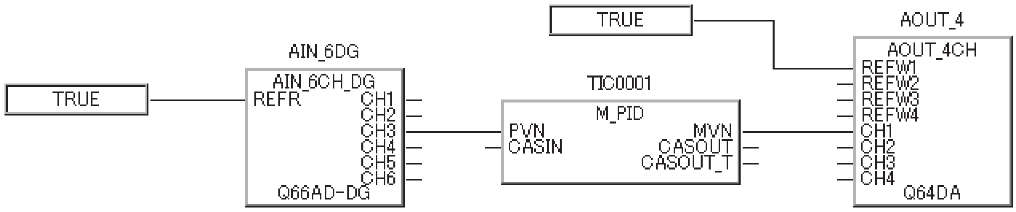
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- Unable to communicate with high-resolution signal conditioning module. (Error code: 1412)
- Abnormality of high-resolution conditioning module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example

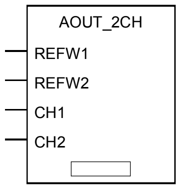


(User-created variable) (Public variable of module FB)



If STBREQ1 is TRUE, operation condition setting request of AIN_6DG (STB) will be executed.
If STBREQ2 is TRUE, operation condition setting request of AOUT_4 (STB) will be executed.

10.1.7 2 Channels Analog Output (AOUT_2CH)

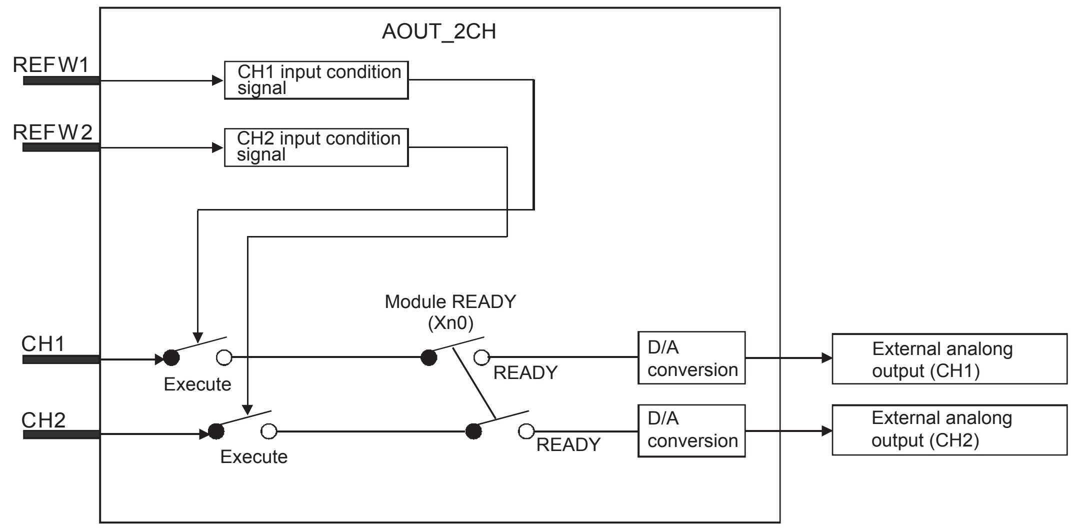
FB	FBD parts
AOUT_2CH	 ▭ Indicates the type name of selected module.

Corresponding module
Q62DA, Q62DAN

Function overview: Write the input digital value from input variable (CH1, CH2) into 2CH D/A conversion module that converts the digital value to analog signal.

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFW1	Input variable	BOOL	CH1 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW2	Input variable	BOOL	CH2 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	CH1	Input variable	REAL	CH1 digital input value	Depends on the input range and resolution mode
	CH2	Input variable	REAL	CH2 digital input value	Depends on the input range and resolution mode

POINT

The digital input to the channel that is not connected to input pin can be executed via ladder program or auto refreshing settings.
In this case, however, the CH□ input condition signal of the above-mentioned channel should be set as FALSE.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: –) When module error (ERR) is TRUE, set ERRC as TRUE to clear module error and make ERR FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH2INH	Public variable	BOOL	Enable/disable D/A conversion (CH1 to CH2). (TRUE: Disabled FALSE: Enabled) Set the D/A conversion enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	OUT1INH to OUT2INH	Public variable	BOOL	Enable/disable D/A output (CH1 to CH2). (TRUE: Offset value FALSE: D/A conversion value) Set output D/A conversion/offset value in channels.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request Execute D/A conversion enabled/disabled setting (CH1INH to CH2INH) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). Store the status of module READY (Xn0). Perform D/A conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: No error) Store TRUE when writing error occurs.	TRUE, FALSE	FALSE	System
	DAENB	Public variable	INT	D/A conversion enabled/disabled setting status Store D/A conversion enabled/disabled setting status. b15 to b1 b0 [... | CH 2 | CH 1] 0: Enable D/A conversion 1: Disabled D/A conversion	0 to 3	3	System
	ERRCOD	Public variable	INT	Error code Store the error code detected by D/A conversion module. For detailed information about the error code, refer to Digital-Analog Converter Module User's Manual.	–	0	System

*1 The public variables CH1INH to CN2INH or OUT1INH to OUT2INH can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

POINT

The default values of "Enable/Disable D/A output" for all channels are "Enable" if this module FB is used.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input condition signal (REFW1 to REFW2)</td><td>Write to D/A conversion module for every channel according to the following conditions
(Example) When the input condition of input variable (CH1) is input condition signal (REFW1)
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Input (CH1 to CH2)</th></tr>
<tr><th>Input condition signal (REFW1 to REFW2)</th><th>Module READY (Xn0)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Write the input digital value from input variable CH1 to CH2 to the D/A conversion module.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Do not write the input digital value from input variable CH1 to CH2 to the D/A conversion module.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Others</td><td>Refer to the following manual for the relative information about the processing and setting concerned with this module FB.
● Digital-Analog Converter Module User's Manual</td></tr>
</table>

POINT

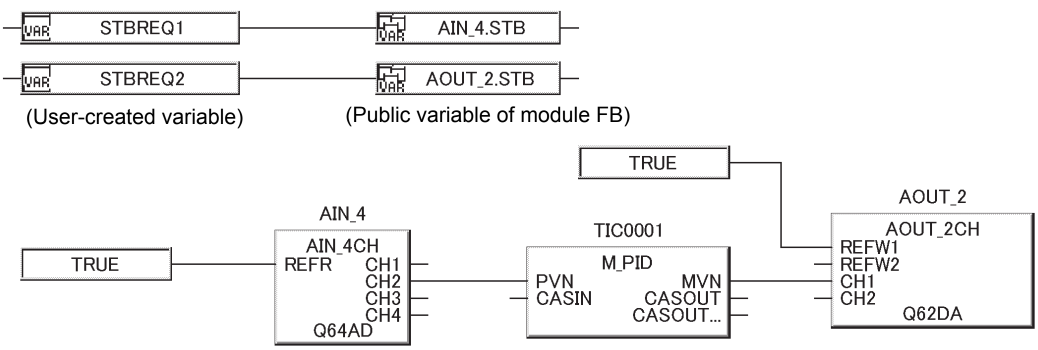
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-DA is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

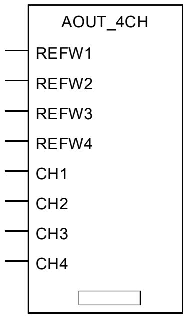
- When the input value is not within the range of –32768 to 32767. (Error code: Refer to Appendix 2)
- Unable to communicate with D/A conversion module. (Error code: 1412)
- Abnormality of D/A conversion module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request (STB) of AIN_4 will be executed.
If STBREQ2 is TRUE, operation condition setting request (STB) of AOUT_2 will be executed.

10.1.8 4 Channels Analog Output (AOUT_4CH)

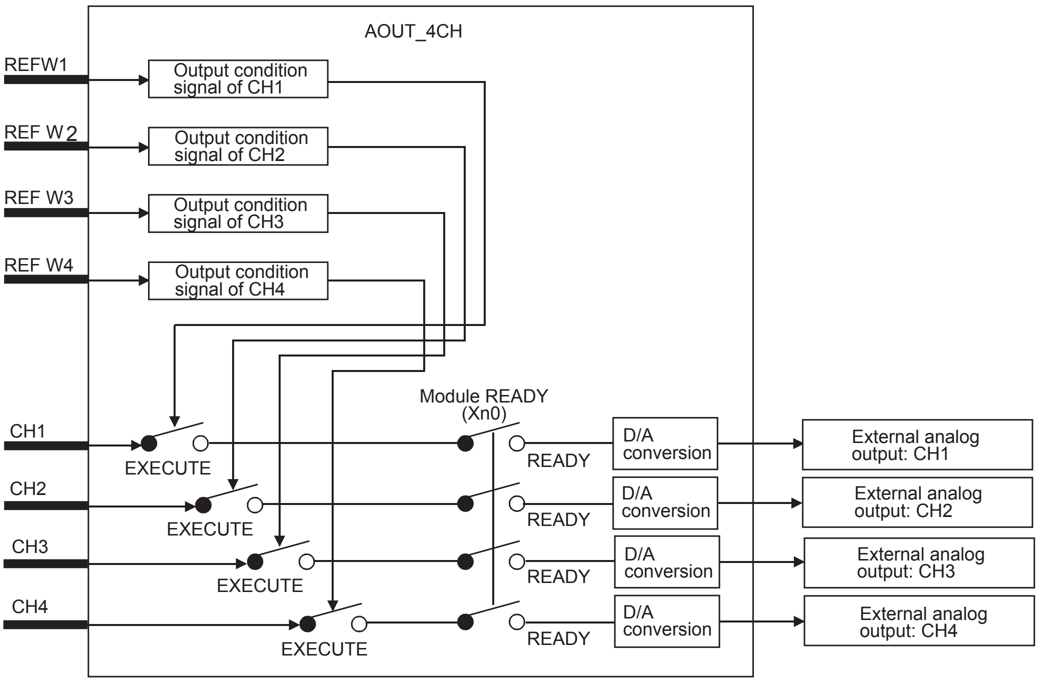
FB	FBD parts
AOUT_4CH	 Indicates the type name of selected module.

Corresponding module
Q64DA, Q64DAN

Function overview: Write the input digital value of input variable (CH1 to CH4) to 4CH D/A conversion module that converts the digital value to analog signal.

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFW1	Input variable	BOOL	CH1 input condition signal (TURE: Execute FALSE: Stop)	TRUE, FALSE
	REFW2	Input variable	BOOL	CH2 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW3	Input variable	BOOL	CH3 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW4	Input variable	BOOL	CH4 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	CH1	Input variable	REAL	CH1 digital input value	Depends on the input range and resolution mode
	CH2	Input variable	REAL	CH2 digital input value	Depends on the input range and resolution mode
	CH3	Input variable	REAL	CH3 digital input value	Depends on the input range and resolution mode
	CH4	Input variable	REAL	CH4 digital input value	Depends on the input range and resolution mode

POINT

The digital input of the channel that is not connected to input pin can be executed via ladder program or auto refresh settings.
In this case, however, the CH□ input condition signal of the above mentioned channel should be set to FALSE.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: —) When module error (ERR) is TRUE, set ERRC as TRUE to clear module error and make ERR FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH4INH	Public variable	BOOL	Enable/disable D/A conversion (CH1 to CH4). (TRUE: Disabled FALSE; Enabled) Set the D/A conversion enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	OUT1INH to OUT4INH	Public variable	BOOL	Enable/disable D/A output (CH1 to CH4). (TRUE: Offset value FALSE: D/A conversion value) Set output D/A conversion/offset value in channels.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request \Execute D/A conversion enabled/disabled setting (CH1INH to CH4INH) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). Store the status of module READY (Xn0). Perform D/A conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag. (TRUE: Error FALSE: No error) Store TRUE when writing error occurs.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	DAENB	Public variable	INT	D/A conversion enabled/disabled setting status. Store D/A conversion enabled/disabled setting status. b15 to b3 b2 b1 b0 ( ) CH 4 | CH 3 | CH 2 | CH 1 0: D/A conversion enabled 1: D/A conversion disabled	0 to F	F	System
	ERRCOD	Public variable	INT	Error code Store the error code detected by D/A conversion module. For detailed information about the error code, refer to Digital-Analog Converter Module User's Manual.	–	0	System

*1 The public variables CH1INH to CN4INH or OUT1INH to OUT4INH can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

> **POINT**
> The default values of "Enable/Disable D/A output" for all channels are "Enable" if this module FB is used.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input condition signal (REFW1 to REFW4)</td><td>Execute write-in to D/A conversion module according to the following conditions
(Example) When the input condition of input variable (CH1) is input condition signal (REFW1)
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Input (CH1 to CH4)</th></tr>
<tr><th>Input condition signal (REFW1 to REFW4)</th><th>Module READY (Xn0)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Write the input digital value of input variable CH1 to CH4 to the D/A conversion module.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Do not write the input digital value of input variable CH1 to CH4 to the D/A conversion module.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table></td></tr>
<tr><td>Others</td><td>Refer to the following manual for the relative information about the processing and setting concerned with this module FB.
● Digital-Analog Converter Module User's Manual.</td></tr>
</table>

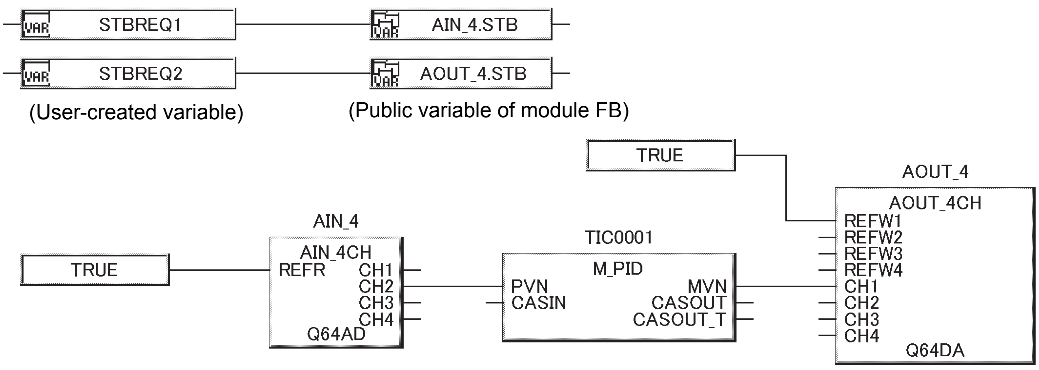
> **POINT**
> - For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-DA is recommended.
>   For details, refer to Section 2.11.1.
> - Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
>   For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

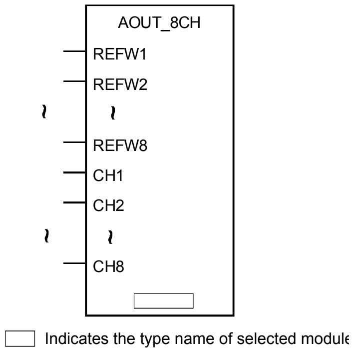
- When the input value is beyond the range of -32768 to 32767. (Error code: Refer to Appendix 2)
- Unable to communicate with D/A conversion module. (Error code: 1412)
- The error of D/A conversion module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request (STB) of AIN_4 will be executed.
If STBREQ2 is TRUE, operation condition setting request (STB) of AOUT_4 will be executed.

10.1.9 8 Channels Analog Output (AOUT_8CH)

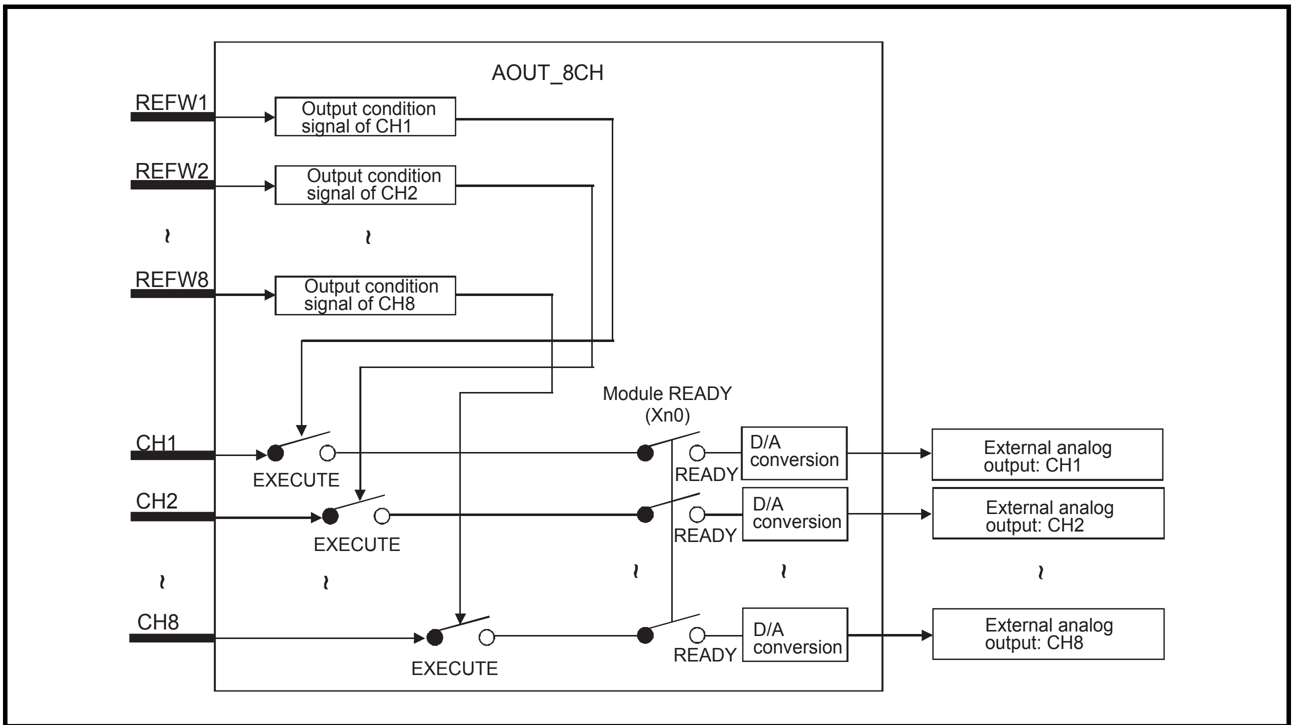
FB	FBD parts
AOUT_8CH	

Corresponding module
Q68DAV, Q68DAVN, Q68DAI, Q68DAIN

Function overview: Write the input digital value of input variable (CH1 to CH8) into 8CH D/A conversion module that converts the digital value to analog signal.

Function/FB classification name: Module FB

Block Diagram



Input and Output Pins

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFW1	Input variable	BOOL	CH1 input condition signal (TURE: Execute FALSE: Stop)	TRUE, FALSE
	REFW2	Input variable	BOOL	CH2 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW3	Input variable	BOOL	CH3 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW4	Input variable	BOOL	CH4 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW5	Input variable	BOOL	CH5 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW6	Input variable	BOOL	CH6 input condition signal (TRUE: Execute FALSE: stop)	TRUE, FALSE
	REFW7	Input variable	BOOL	CH7 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW8	Input variable	BOOL	CH8 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	CH1	Input variable	REAL	CH1 digital input value	Depends on the input range and resolution mode
	CH2	Input variable	REAL	CH2 digital input value	Depends on the input range and resolution mode
	CH3	Input variable	REAL	CH3 digital input value	Depends on the input range and resolution mode
	CH4	Input variable	REAL	CH4 digital input value	Depends on the input range and resolution mode
	CH5	Input variable	REAL	CH5 digital input value	Depends on the input range and resolution mode
	CH6	Input variable	REAL	CH6 digital input value	Depends on the input range and resolution mode
	CH7	Input variable	REAL	CH7 digital input value	Depends on the input range and resolution mode
	CH8	Input variable	REAL	CH8 digital input value	Depends on the input range and resolution mode

POINT

The digital input to the channel that is not connected to input pin can be executed via ladder program or auto refreshing settings.
In this case, however, the CH□ input condition signal of the above-mentioned channel should be set as FALSE.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: –) When module error (ERR) is TRUE, set ERRC as TRUE to clear module error and make ERR FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH8INH	Public variable	BOOL	Enable/disable D/A conversion (CH1 to CH8). (TRUE: Disabled FALSE; Enabled) Set the D/A conversion enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	OUT1INH to OUT8INH	Public variable	BOOL	Enable/disable D/A output (CH1 to CH8). (TRUE: Offset value FALSE: D/A conversion value) set output D/A conversion value or offset value in channels.	TRUE, FALSE	FALSE	User

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	STB	Public variable	BOOL	Operation condition setting request. Execute D/A conversion enabled/disabled setting (CH1INH to CH8INH) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). Store the status of module READY (Xn0). Perform D/A conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag. (TRUE: Error FALSE: No error) Store TRUE when writing error occurs.	TRUE, FALSE	FALSE	System
	DAENB	Public variable	INT	D/A conversion enabled/disabled setting status. Store D/A conversion enabled/disabled setting status. b15 to b7 b6 b5 b4 b3 b2 b1 b0 CH 8, CH 7, CH 6, CH 5, CH 4, CH 3, CH 2, CH 1 0: Enable D/A conversion 1: Disable D/A conversion	0 to FF	FF	System
	ERRCOD	Public variable	INT	Error code Store the error code detected by D/A conversion module. For detailed information about the error code, refer to Digital-Analog Converter Module User's Manual.	–	0	System

*1 The public variables CH1INH to CN8INH or OUT1INH to OUT8INH can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

POINT

The default values of "Enable/Disable D/A output" for all channels are "Enable" if this module FB is used.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input condition signal (REFW1 to REFW8HV)</td><td>Write to D/A conversion module for every channel according to the following conditions
(Example) When the input condition of input variable (CH1) is input condition signal (REFW1)
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Input (CH1 to CH8)</th></tr>
<tr><th>Input condition signal (REFW1 to REFW8)</th><th>Unit READY (Xn0)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Write the input digital value from input variable CH1 to CH8 to the D/A conversion unit.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Do not write the input digital value from input variable CH1 to CH8 to the D/A conversion unit.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Others</td><td>Refer to the following manual for the relative information about the processing and setting concerned with this module FB.
● Digital-Analog Converter Module User's Manual.</td></tr>
</table>

POINT

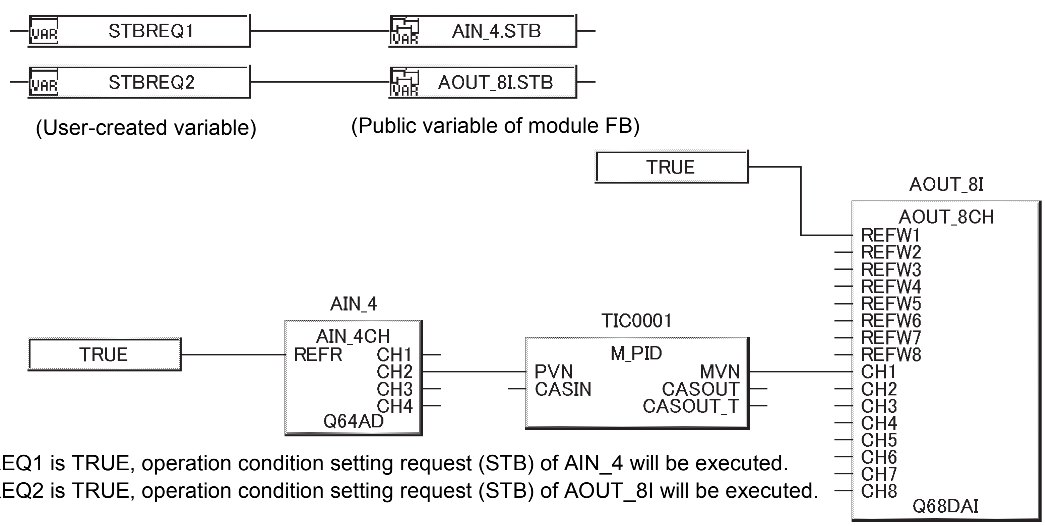
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-DA is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

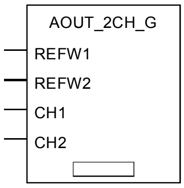
- When the input value is not within the range of –32768 to 32767. (Error code: Refer to Appendix 2)
- When it is unable to communicate with D/A conversion module. (Error code: 1412)
- The error of D/A conversion module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request (STB) of AIN_4 will be executed.
If STBREQ2 is TRUE, operation condition setting request (STB) of AOUT_8I will be executed.

10.1.10 Channel-isolated 2 Channels Analog Output (AOUT_2CH_G)

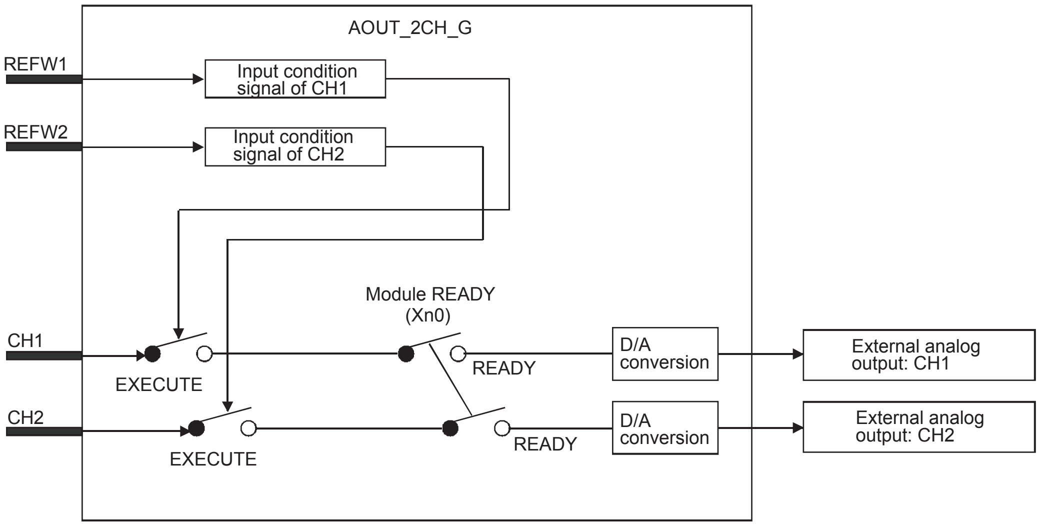
FB	FBD parts
AOUT_2CH_G	 Indicates the type name of selected module.

Corresponding module
Q62DA_FG

Function overview: Write the input digital value from input variable (CH1, CH2) into channel-isolated D/A conversion module that converts the digital value to analog signal.

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFW1	Input variable	BOOL	CH1 input condition signal (TURE: Execute FALSE: Stop)	TRUE, FALSE
	REFW2	Input variable	BOOL	CH2 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	CH1	Input variable	REAL	CH1 digital input value	Depends on the input range and resolution mode
	CH2	Input variable	REAL	CH2 digital input value	Depends on the input range and resolution mode

POINT

- The digital input to the channel that is not connected to input pin can be executed via ladder program or auto refreshing settings.
  In this case, however, the CH□ input condition signal of the above-mentioned channel should be set as FALSE.
- A wire break is detected when a value lower than the wire break detection value is output from a channel of which the input range is 4 to 20mA (extended mode) the wire break detection is enabled, and the auto configuration of wire break detection value is FALSE. In the conversion processing of the module FB, the value to be set to output value of the analog module is limited to prevent a false detection of a wire break. The approximate limiting value is shown below.
  Digital value: -2018, Percentage equivalent: -16.82%
  To input a value lower than the above-mentioned value, disable the wire break detection.
  Furthermore, for modules whose upper five digits of serial number are 11102 or later, set the auto configuration of wire break detection value to TRUE to automatically change the wire break detection value in accordance with output value of analog module. For the relation between the output value and the wire break detection value in this case, refer to Channel Isolated Digital-Analog Converter Module User's Manual.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: –) When module error (ERR) is TRUE, set ERRC as TRUE to clear module error and make ERR FALSE.	TRUE, FALSE	FALSE	User
	RGALC	Public variable	BOOL	Alarm output clear request (TRUE: Alarm output clear request FALSE: –) When alarm detection (RGAL) is TRUE, set RGALC as TRUE to clear alarm and make MHAL1, MHAL2, MLAL1, MLAL2 and RGAL FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH2INH	Public variable	BOOL	Enable/disable D/A conversion (CH1 to CH2). (TRUE: Disabled FALSE; Enabled) Set the D/A conversion enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	OUT1INH to OUT2INH	Public variable	BOOL	Enable/disable D/A output (CH1 to CH2). (TRUE: Offset value FALSE: D/A conversion value) Set output D/A conversion value or offset value in channels.	TRUE, FALSE	FALSE	User
	AL1ENB to AL2ENB	Public variable	BOOL	CH1 to CH2 alarm output disconnection enabled/ disabled setting (TRUE: Output enabled FALSE: Output disabled) Set the alarm output wire break detection enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	BNALC	Public variable	BOOL	Wire break detection clear request (TRUE: Wire break detection clear request FALSE: –) When wire break detection signal (BNAL) is TRUE, set BNALC as TRUE to clear wire break detection signal and make BNOUT1, BNOUT2 and BNAL FALSE.	TRUE, FALSE	FALSE	User
	BNAUTSET1 to BNAUTSET2	Public variable	BOOL	CH1 to CH2 auto configuration of wire break detection value valid/invalid setting (TRUE: Valid FALSE: Invalid) If invalid, detect the wire break when 1mA or less. (*2) It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	STB	Public variable	BOOL	Operation condition setting request Execute D/A conversion enabled/disabled setting (CH1INH to CH4INH), alarm output and wire break detection enable/disable setting (AL1ENB to AL2ENB), auto configuration of wire break detection value valid/invalid setting (BNAUTSET1 to BNAUTSET2) when STB transforms from FALSE to TRUE. (*2)	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF) Store the status of module READY (Xn0). Perform D/A conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: No error) Store TRUE when writing error occurs.	TRUE, FALSE	FALSE	System
	BNAL	Public variable	BOOL	Wire break detected signal (TRUE: Wire break detected FALSE: Normal) Store TRUE once either CH1 or CH2 of D/A conversion module is disconnected.	TRUE, FALSE	FALSE	System
	RGAL	Public variable	BOOL	Alarm output signal (TRUE: Alarms occur FALSE: Normal) Store TRUE once the high/low limit setting of either CH1 or CH2 of D/A conversion module is exceeded.	TRUE, FALSE	FALSE	System
	MLAL1 to MLAL2	Public variable	BOOL	CH1 to CH2 alarm output flag low limit alarm (TRUE: Over FALSE: Normal) Store TRUE in channels when the set low limit is exceeded.	TRUE, FALSE	FALSE	System
	MHAL1 to MHAL2	Public variable	BOOL	CH1 to CH2 alarm output flag high limit alarm (TRUE: Over FALSE: Normal) Store TRUE in channels when the set high limit is exceeded.	TRUE, FALSE	FALSE	System
	BNOUT1 to BNOUT2	Public variable	BOOL	CH1 to CH2 wire break detection flag (TRUE: Disconnection detected FALSE: Normal) Store TRUE in channels in disconnection.	TRUE, FALSE	FALSE	System
	MONIFLG	Public variable	BOOL	Monitor start flag (TRUE: Monitor start FALSE: -) Store TRUE when monitor starts.	TRUE, FALSE	FALSE	System
	DAENB	Public variable	INT	D/A conversion enabled/disabled setting status. Store the D/A conversion enabled/disabled setting status. b15 to b1 b0 [(b15 to b2) | CH2 | CH1] 0: Enable D/A conversion 1: Disable D/A conversion	0 to 3	3	System
	ALMENB	Public variable	INT	The setting status of Alarm output & wire break detection enabled/disabled. Store alarm the setting status of output & wire break detection enabled/disabled. b15 to b13 b12 to b1 b0 [(b15 to b14) | CH2 | CH1 | (b11 to b2) | CH2 | CH1] b0, b1 : Alarm output setting b12, b13 : Wire break detection setting 0 : Enabled 1 : Disabled	0 to 12291 (3003H)	12291 (3003H)	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRCOD	Public variable	INT	Error code Store the error code detected by D/A conversion module. For detailed information about the error code, refer to Channel Isolated Digital-Analog Converter Module User's Manual.	–	0	System
	MONI1 to MONI2	Public variable	INT	CH1 to CH2 output monitor value Store output monitor value (D/A conversion digital value &. A/D conversion digital value in module) of every channel	Range of module resolution	0	System
	BNAUTSET1ENB to BNAUTSET2ENB	Public variable	BOOL	CH1 to CH2 auto configuration of wire break detection value valid/invalid setting status (TRUE: Valid FALSE: Invalid) The status of auto configuration of wire break detection value valid/invalid is stored (*2)	TRUE, FALSE	FALSE	System

*1 The public variables CH1INH to CN2INH, OUT1INH to OUT2INH, AL1ENB to AL2ENB, or BNAUTSET1ENB to BNAUTSET2ENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

*2 The auto configuration of wire break detection value function is supported by modules whose upper five digits of serial number are 11102 or later.
For modules whose upper five digits of serial number are earlier than 11102, execute wire break detection with 1mA or less even though public variable BNAUTSET1, BNAUTSET2 are set to TRUE. (The value of BNAUTSET1ENB, BNAUTSET2ENB remains FALSE.)

POINT

- The default values of "Enable/Disable D/A output" for all channels are "Enable" if this module FB is used.
- When project files created with PX Developer Version 1.27D or later are opened and compiled with PX Developer Version 1.23Z or earlier, the auto configuration of wire break detection value will not be executed regardless the valid/invalid status of the auto configuration of wire break detection value, and output value for preventing false detection will be limited.
- Cancelling the auto configuration of wire break detection value with the digital input value of lower than -2018 may be executed the detection of wire break. For cancellation of the auto configuration of wire break detection value, set the digital input value to -2018 or more or disable alarm detection beforehand.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input condition signal (REFW1 to REFW2)</td><td>Write to D/A conversion module of channels according to the following conditions
(Example) When the input condition of input variable (CH1) is input condition signal (REFW1)
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Input (CH1 to CH2)</th></tr>
<tr><th>Input condition signal (REFW1 to REFW2)</th><th>Module READY (Xn0)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Write the input digital value from input variable CH1 to CH2 to the D/A conversion module.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Do not write the input digital value from input variable CH1 to CH2 to the D/A conversion module.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Others</td><td>Refer to the following manual for the relative information about the processing and setting concerned with this module FB.
● Channel Isolated Digital-Analog Converter Module User's Manual.</td></tr>
</table>

POINT

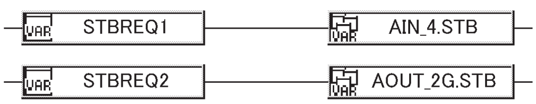
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-DA is recommended.
  For details, refer to Section 2.11.1.
  However, the auto configuration of wire break detection value function is not supported.
  When using the auto configuration of wire break detection value function, set with public variables.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

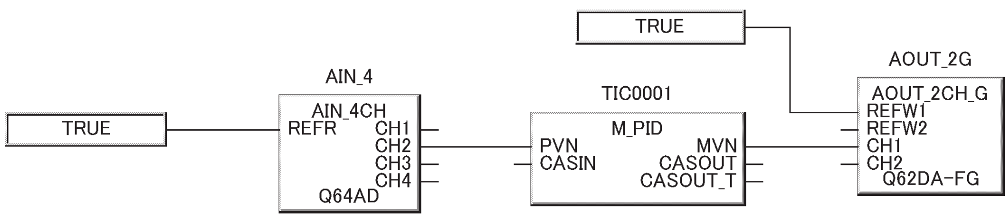
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- Unable to communicate with D/A conversion module. (Error code: 1412)
- Abnormality of D/A conversion module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example

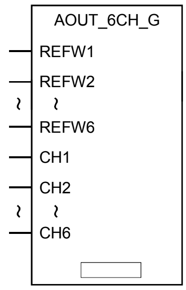


(User-defined variable) (Public variable of unit FB)



If STBREQ1 is TRUE, operation condition setting request (STB) of AIN_4 will be executed.
If STBREQ2 is TRUE, operation condition setting request (STB) of AOUT_2G will be executed.

10.1.11 Channel-isolated 6 Channels Analog Output (AOUT_6CH_G)

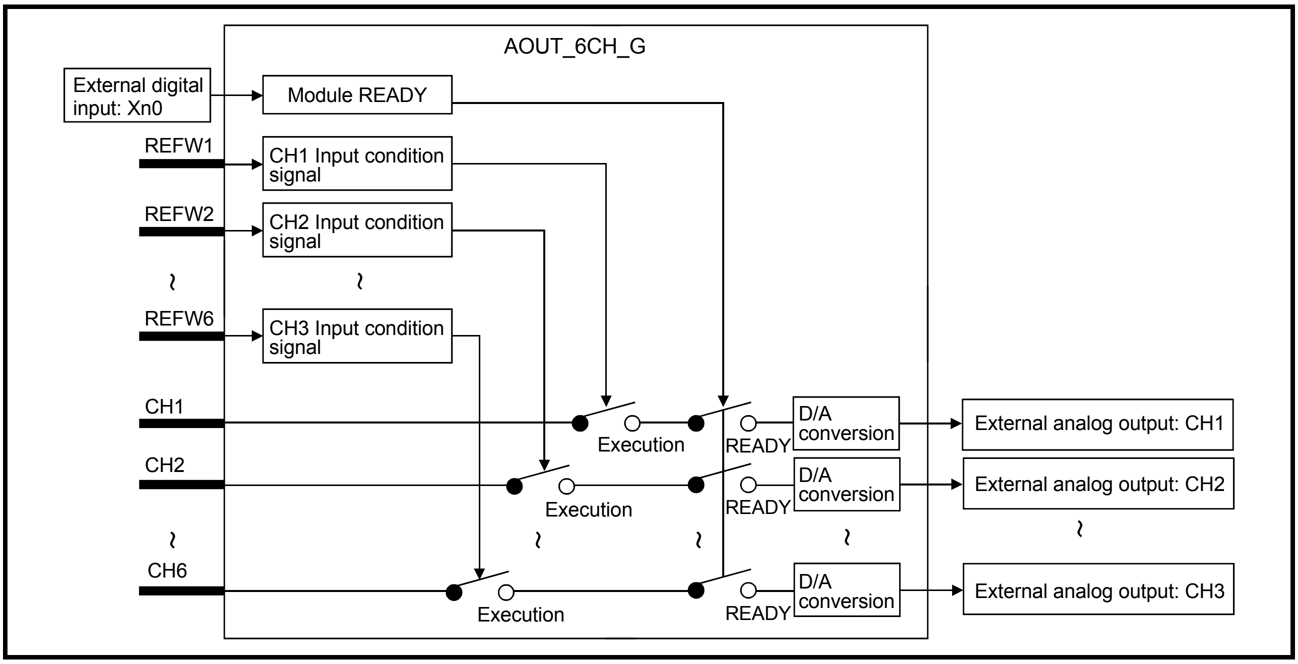
FB	FBD parts
AOUT_6CH_G	 Indicates the type name of selected module.

Corresponding module
Q66DA-G

Function overview: Writes digital values input from the input variables (CH1 to CH6) to channel-isolated D/A converter modules (6 channels), which converts digital values to analog signals.

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFW1 to REFW6	Input variable	BOOL	CH1 to CH6 Input condition signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	CH1 to CH6	Input variable	REAL	CH1 to CH6 Digital input value	Depends on the input range, resolution mode, and scaling function of the module.

POINT

The digital input to the channel that is not connected to input pin can be executed via ladder program or auto refreshing settings.
In this case, however, the CH□ input condition signal of the above-mentioned channel should be set as FALSE.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: −) When module error (ERR) is TRUE, set ERRC as TRUE to clear module error and make ERR FALSE.	TRUE, FALSE	FALSE	User
	RGALC	Public variable	BOOL	Alarm output clear request (TRUE: Alarm output clear request FALSE: −) When alarm detection (RGAL) is TRUE, set RGALC as TRUE to clear alarm and make MHAL1 to MHAL6, MLAL1 to MLAL6 and RGAL FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH6INH	Public variable	BOOL	Enable/disable D/A conversion (CH1 to CH6). (TRUE: Disabled FALSE; Enabled) Set the D/A conversion enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	OUT1INH to OUT6INH	Public variable	BOOL	Enable/disable D/A output (CH1 to CH6). (TRUE: Offset value FALSE: D/A conversion value) Set output D/A conversion value or offset value in channels.	TRUE, FALSE	FALSE	User
	AL1ENB to AL6ENB	Public variable	BOOL	CH1 to CH6 alarm output enabled/disabled setting (TRUE: Output enabled FALSE: Output disabled) Alarm output enabled/disabled setting in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request Execute D/A conversion enabled/disabled setting (CH1INH to CH6INH), alarm output and alarm output enabled/disabled setting (AL1ENB to AL6ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF) Store the status of module READY (Xn0). Perform D/A conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: No error) Store TRUE when writing error occurs.	TRUE, FALSE	FALSE	System
	RGAL	Public variable	BOOL	Alarm output signal (TRUE: Alarms occur FALSE: Normal) Store TRUE once the high/low limit setting of either CH1 or CH6 of D/A conversion module is exceeded.	TRUE, FALSE	FALSE	System
	MLAL1 to MLAL6	Public variable	BOOL	CH1 to CH6 alarm output flag low limit alarm (TRUE: Over FALSE: Normal) Store TRUE in channels when the set low limit is exceeded.	TRUE, FALSE	FALSE	System
	MHAL1 to MHAL6	Public variable	BOOL	CH1 to CH6 alarm output flag high limit alarm (TRUE: Over FALSE: Normal) Store TRUE in channels when the set high limit is exceeded.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	DAENB	Public variable	WORD	D/A conversion enabled/disabled setting status. Store the D/A conversion enabled/disabled setting status. b15 to b5 b4 b3 b2 b1 b0: 0 ··· 0, CH6, CH5, CH4, CH3, CH2, CH1 0: Enable D/A conversion 1: Disable D/A conversion	0 to 003FH	003FH	System
	ALMENB	Public variable	WORD	The setting status of Alarm output enabled/disabled. Store alarm the setting status of output enabled/ disabled. b15 to b5 b4 b3 b2 b1 b0: 0 ··· 0, CH6, CH5, CH4, CH3, CH2, CH1 b0 to b5 : Alarm output setting 0 : Enabled 1 : Disabled	0 to 003FH	003FH	System
	SCLENB	Public variable	WORD	Scaling enabled/disabled setting status Stores scaling enabled/disabled setting status. b15 to b5 b4 b3 b2 b1 b0: 0 ··· 0, CH6, CH5, CH4, CH3, CH2, CH1 0: Valid 1: Invalid	0 to 003FH	003FH	System
	ERRCOD	Public variable	INT	Error code Store the error code detected by D/A conversion module. For detailed information about the error code, refer to Channel Isolated Digital-Analog Converter Module User's Manual.	–	0	System

*1 The public variables CH1INH to CN6INH or AL1ENB to AL6ENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

POINT

When using this module FB, the default value of D/A output enabled/disabled setting is enabled for all channels of D/A output.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input condition signal (REFW1 to REFW6)</td><td>Write to D/A conversion module of channels according to the following conditions
(Example) When the input condition of input variable (CH1) is input condition signal (REFW1)
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Input (CH1 to CH6)</th></tr>
<tr><th>Input condition signal (REFW1 to REFW6)</th><th>Module READY (Xn0)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Write the input digital value from input variable CH1 to CH6 to the D/A conversion module.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Do not write the input digital value from input variable CH1 to CH6 to the D/A conversion module.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Others</td><td>Refer to the following manual for the relative information about the processing and setting concerned with this module FB.
● Channel Isolated Digital-Analog Converter Module User's Manual.</td></tr>
</table>

POINT

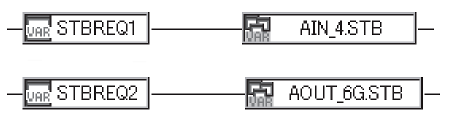
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-DA is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

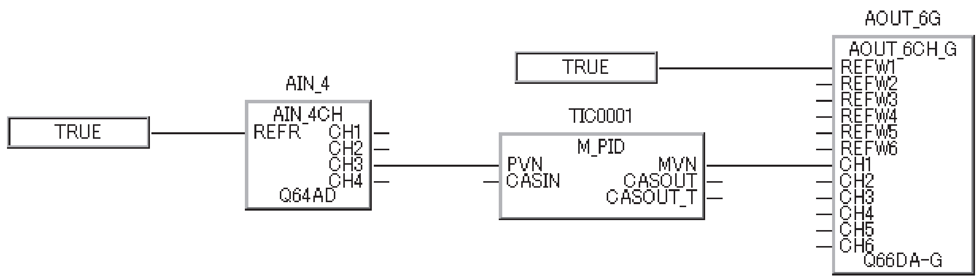
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- Unable to communicate with D/A conversion module. (Error code: 1412)
- Abnormality of D/A conversion module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example

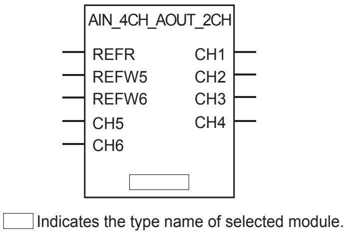


(User-created variable) (Public variable of module FB)



If STBREQ1 is TRUE, operation condition setting request of AIN_4 (STB) will be executed.
If STBREQ2 is TRUE, operation condition setting request of AOUT_6G (STB) will be executed.

10.1.12 Analog Input/Output (Input 4 channels, Output 2 channels) (AIN_4CH_AOUT_2CH)

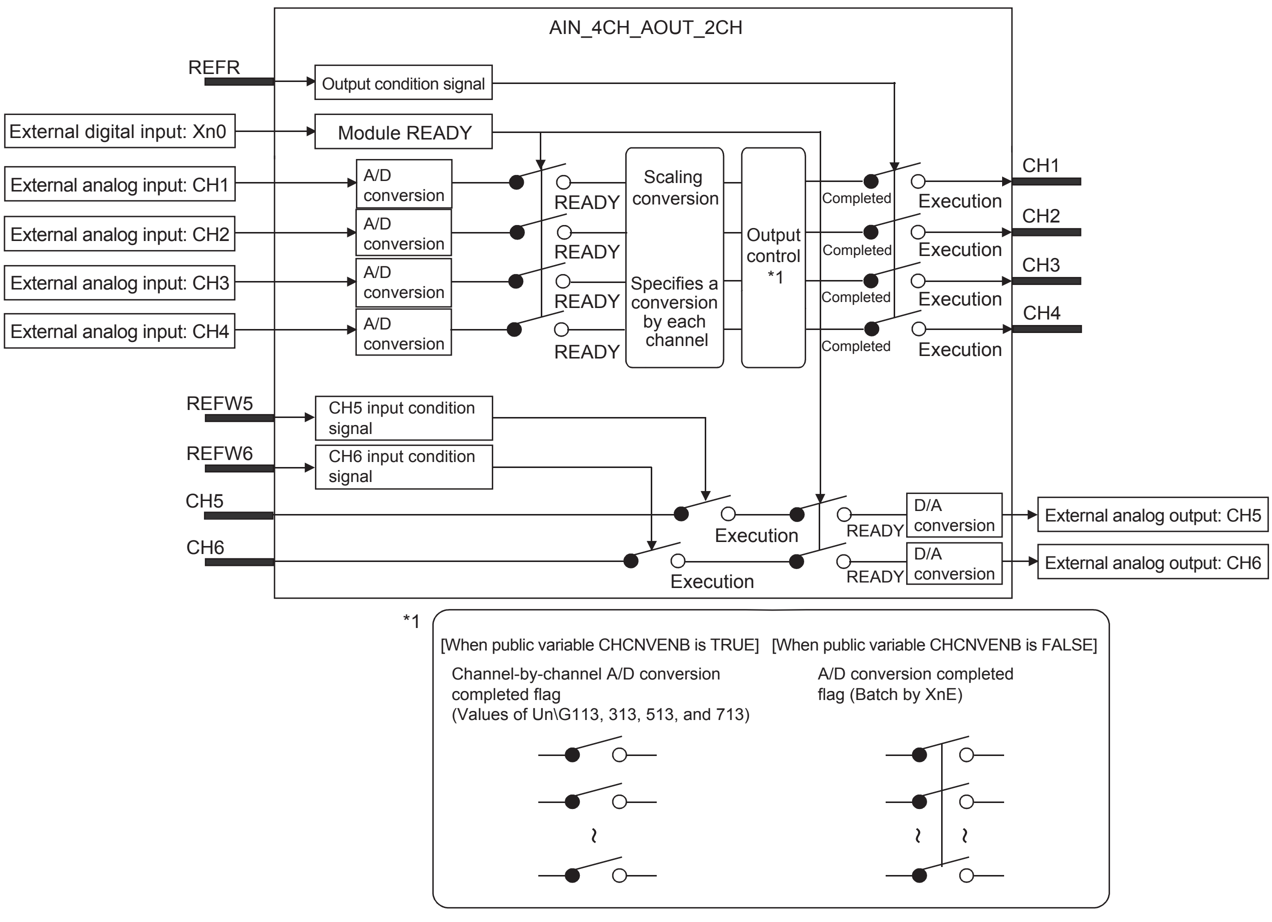
FB	FBD parts
AIN_4CH_AOUT_2CH	

Corresponding module
Q64AD2DA

Function overview: Reads the digital output value or scaling value of 4 channels A/D conversion module that converts analog signal to digital value, and outputs it from output variable (CH1 to CH4). Writes the input digital value from input variable (CH5, CH6) into 2channels D/A conversion module that converts the digital value to analog signal.

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
	REFW5	Input variable	BOOL	CH5 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW6	Input variable	BOOL	CH6 input condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	CH5	Input variable	REAL	CH5 digital input value	Depends on the output range, resolution mode, and scaling function of the module.
	CH6	Input variable	REAL	CH6 digital input value	Depends on the output range, resolution mode, and scaling function of the module.
Output	CH1 to CH4	Output variable	REAL	CH1 to CH4 Output value (*1)	Depends on the input range, resolution mode, and scaling function of the module.

*1 With the scaling enable/disable setting, either digital output values or scaling values are selected for each channel and are output from the output values of CH1 to CH4.

POINT

The digital input to the channel that is not connected to input pin can be executed via ladder program or auto refreshing settings.
In this case, however, the CH□ input condition signal of the above-mentioned channel should be set as FALSE.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: -). When occurring module error (ERR) and input signal abnormality detection signal (SYSAL) are TRUE, set ERRC to TRUE to clear the module error and the input signal abnormality, and ERR and SYSAL will become FALSE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute A/D conversion enabled/disabled setting (CH1INH to CH4INH) input signal abnormality detection enabled/disabled setting (AL1ENB to AL4ENB), input signal abnormality detection setting value (AL1SETVAL to AL4SETVAL) and D/A conversion enabled/disabled setting (CH5INH, CH6INH) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). The status of module READY (Xn0) is stored.	TRUE, FALSE	FALSE	System
	EXT_PWR_ OFF (External poweroff flag)	Public variable	BOOL	External power off flag (TRUE: ON FALSE: OFF) Store the status of External power off flag (Xn6). Store TRUE when the externally-supplied power DC24V is not supplied. When this flag is true, A/D and D/A conversion processing is not executed.	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: no error). Store TRUE when error occurs in writing.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	ERRCOD1 to ERRCOD6, ERRCODOTHER	Public variable	INT	Error code Store the error code detected by module. Store the error code related to channels and the error code unrelated to channels. For detailed information about the error code, refer to Analog Input/Output Module User's Manual.	–	0	System
	SCLENB	Public variable	WORD	Scaling enabled/disabled setting status. Stores scaling enabled/disabled setting status. b15 to b5 b4 b3 b2 b1 b0 0 ··· 0 CH6 CH5 CH4 CH3 CH2 CH1 0: Valid 1: Invalid	0 to $003F_H$	$003F_H$	System
	CH1INH to CH4INH	Public variable	BOOL	Enable/disable A/D conversion (CH1 to CH4) (TRUE: Enabled FALSE: Disabled). Set whether enable/disable A/D conversion value output of each channel. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL4ENB	Public variable	INT	CH1 to CH4 input signal error detection setting Set whether enable/disable alarm output of input signal error detection of each channel. It is valid when STB transforms from FALSE to TRUE.	0: Invalid 1: High and low limit detection 2: Low limit detection 3: High limit detection 4: Wire break detection	0	User
	AL1SETVAL to AL4SETVAL	Public variable	REAL	Input signal error detection setting value of CH1 to CH4 Set the setting value of input analog value fault detection of each channel. (unit: %) It is valid when STB transforms from FALSE to TRUE.	0 to 25.0	0	User
	MRES	Public variable	BOOL	Maximum/minimum value reset request. Maximum/minimum value (CH1MAX to CH4MAX, CH1MIN to CH4MIN) is cleared when MRES turns from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the digital output value. When the setting is TRUE, the channel-by-channel A/D conversion completed flag is used as the output condition. When the setting is FALSE, the A/D conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	A/D conversion completed flag (TRUE: ON FALSE: OFF). Store the status of A/D conversion completion flag (XnE).	TRUE, FALSE	FALSE	System
	SYSAL	Public variable	BOOL	Input signal abnormality detection signal. (TRUE: abnormal FALSE: normal). Store TRUE once input signal abnormality occurs in any of CH1 to CH4 of A/D conversion module.	TRUE, FALSE	FALSE	System
	RGAL1 to RGAL4	Public variable	BOOL	CH1 to CH4 input signal abnormality detection (TRUE: abnormal FALSE: normal) Store TRUE of the channel if it surpasses the setting range of I/O signal abnormality detection setting value.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	ADENB	Public variable	WORD	A/D conversion enabled/disabled setting status. Store A/D conversion enabled/disabled setting status b15 to b3 b2 b1 b0 CH 4 | CH 3 | CH 2 | CH 1 0: Enable A/D conversion 1: Disable A/D conversion	0 to 000FH	0	System
	AL1ENBSTAT to AL4ENBSTAT	Public variable	INT	Input signal error detection setting status of CH1 to CH4 Store the input signal error detection setting status of each channel.	0: Invalid 1: High and low limit detection 2: Low limit detection 3: High limit detection 4: Wire break detection	0	System
	CH1DOUT to CH4DOUT	Public variable	INT	Sampling value of CH1 to CH4 Store the output sampling value of each channel.	Depends on the input range and resolution mode.	0	System
	CH1MAX to CH4MAX	Public variable	DINT	Maximum A/D conversion value of CH1 to CH4. Store the maximum digital value based on channels. (*2)	Depends on the input range, resolution mode, and scaling function of the module.	0	System
	CH1MIN to CH4MIN	Public variable	DINT	Minimum A/D conversion value of CH1 to CH4. Store the minimum digital value based on channels. (*2)		0	System
	CH5INH, CH6INH	Public variable	BOOL	Enable/disable D/A conversion (CH5, CH6). (TRUE: Disabled FALSE; Enabled) Set the D/A conversion enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	OUT5INH, OUT6INH	Public variable	BOOL	Enable/disable D/A output (CH5, CH6). (TRUE: Offset value FALSE: D/A conversion value) Set output D/A conversion/offset value in channels.	TRUE, FALSE	FALSE	User
	DAENB	Public variable	WORD	D/A conversion enabled/disabled setting status. Store the D/A conversion enabled/disabled setting status. b15 to b1 b0 0 … 0 | CH 2 | CH 1 0: Enable D/A conversion 1: Disable D/A conversion	0 to 0003H	0003H	System

*1 The public variable CH1INH to CN4INH, AL1ENB to AL4ENB, AL1SETVAL to AL6SETVAL, CHCNVENB, or OUT5INH to OUT6INH can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

*2 With the scaling enable/disable setting, either digital output values or scaling value is selected for each channel and output from the output values of CH1 to CH4. The storage value is also switched between the maximum value and minimum value with the setting.

POINT

The default values of "Enable/Disable D/A output" for all channels are "Enable" if this module FB is used.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH4)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read digital output value from A/D conversion module and output digital output value of each channel from output variable CH1 to CH4.</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH4 holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel A/D conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: A/D conversion completed flag (batch by XnE)
</td></tr>
<tr><td>Input condition signal (REFW5 to REFW6)</td><td>
Write to D/A conversion module for every channel according to the following conditions
(Example) When the input condition of input variable (CH5) is input condition signal (REFW5)
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Input (CH5, CH6)</th></tr>
<tr><th>Input condition signal (REFW5, REFW6)</th><th>Module READY (Xn0)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Write the input digital value from input variable CH5, CH6 to the D/A conversion module.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Do not write the input digital value from input variable CH5, CH6 to the D/A conversion module.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Others</td><td>For information about the processing and setting of corresponding module of this module FB, refer to the following manual:
● Channel Isolated Digital-Analog Converter Module User's Manual</td></tr>
</table>

POINT

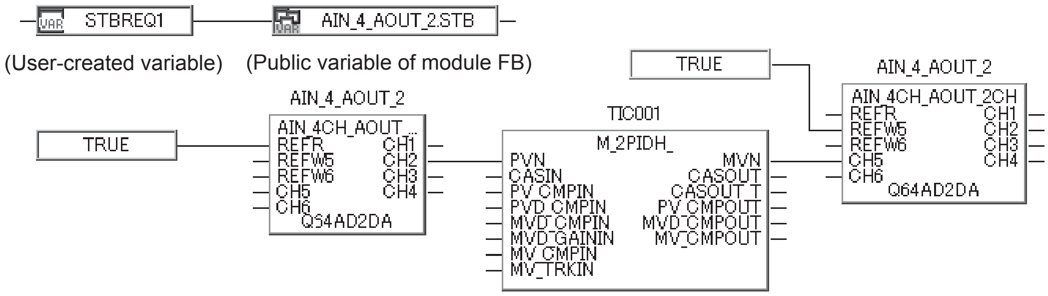
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-AD is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

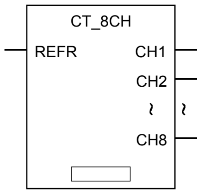
- Unable to communicate with A/D conversion module. (Error code: 1412)
- Abnormality of A/D conversion module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request of AIN_4_AOUT_2 (STB) will be executed.

10.1.13 8 Channels CT Input (CT_8CH)

FB	FBD parts
CT_8CH	

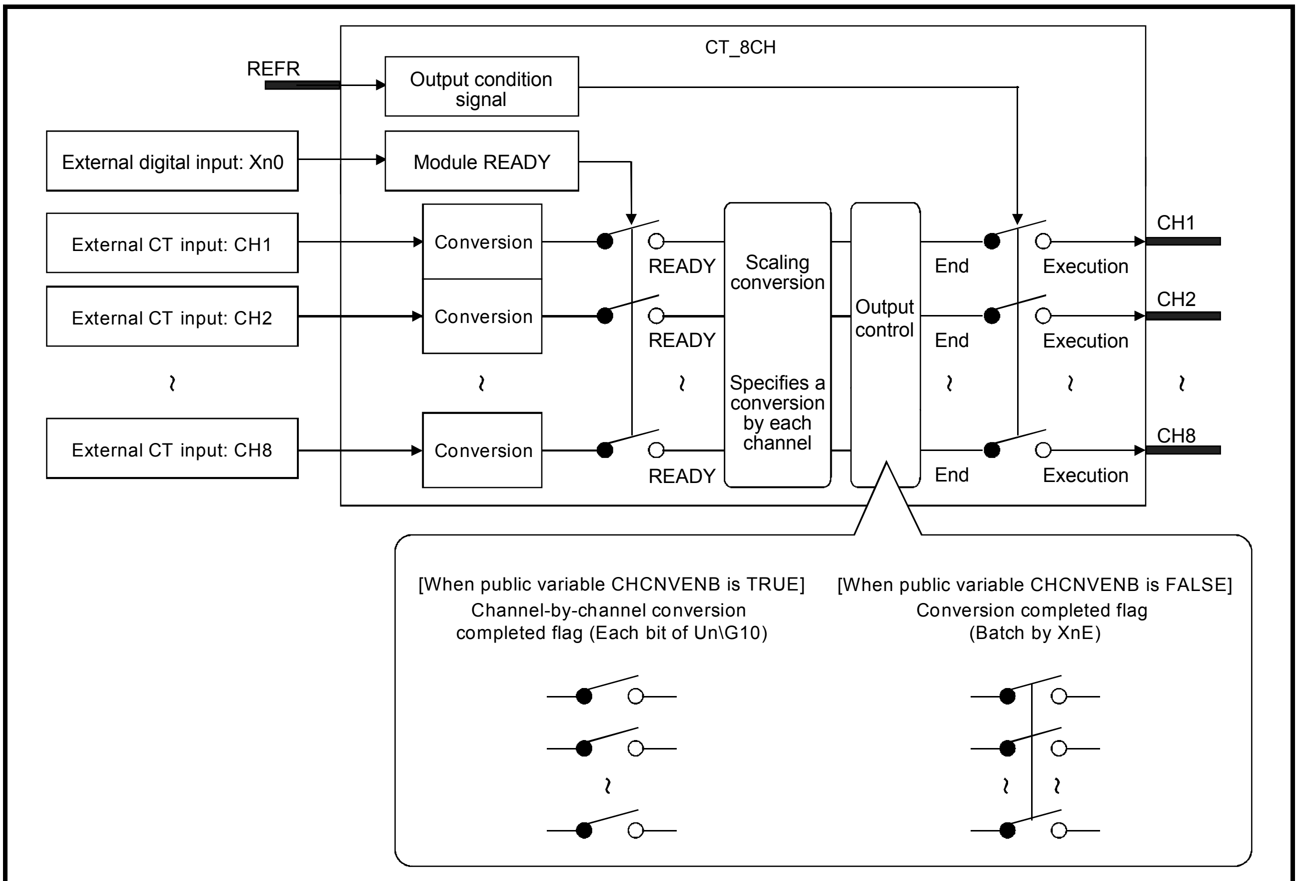
Indicates the type name of selected module.

Corresponding module
Q68CT

Function overview: Reads the digital output value of 8 channels CT Input module that converts secondary current of CT sensor to digital value, and outputs it from output variable (CH1 to CH8).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH8	Output variable	REAL	CH1 to CH8 output value (*1)	Depending on the range of digital output values (0 to 10000) of a module or scaling function

*1 With the scaling enable/disable setting, either digital output values or scaling values are selected for each channel and are output from the output values of CH1 to CH8.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: -). When occurring module error (ERR), input signal abnormality detection signal (SYSAL), and peak current detection signal (PCDAL) are TRUE, set ERRC to TRUE to clear the module error, the input signal abnormality, and peak current detection, and then ERR, SYSAL, and PCDAL will become FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH8INH	Public variable	BOOL	Enable/disable conversion (CH1 to CH8) (TRUE: Enabled FALSE: Disabled). Set whether enable/disable conversion value output of each channel. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL8ENB	Public variable	BOOL	Enable/disable alarm output (CH1 to CH8) (TRUE: Disabled FALSE: Enabled). Set whether enable/disable alarm output of input signal abnormality, process alarm, rate alarm, and peak current detection of each channel. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute conversion enabled/disabled setting (CH1INH to CH8INH) and alarm output enable/disable setting (AL1ENB to AL8ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	MRES	Public variable	BOOL	Maximum/minimum value reset request. Maximum/minimum value (CH1MAX to CH8MAX, CH1MIN to CH8MIN) is cleared when MRES turns from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	PCD1RES to PCD8RES	Public variable	BOOL	Peak current detection count reset request (CH1 to CH8) The value of peak current detection count (CH1PCDCNT to CH8PCDCNT) of the specified channel is cleared when PCD□RES turns TRUE from FALSE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). The status of module READY (Xn0) is stored. Execute conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the conversion value. When the setting is TRUE, the channel-by-channel conversion completed flag is used as the output condition. When the setting is FALSE, the conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	Conversion completed flag (TRUE: ON FALSE: OFF). Store the status of conversion completion flag (XnE).	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: no error). Store TRUE when error occurs.	TRUE, FALSE	FALSE	System
	SYSAL	Public variable	BOOL	Input signal abnormality detection signal. (TRUE: abnormal FALSE: normal). Store TRUE once input signal abnormality occurs in any of CH1 to CH4 of CT input module.	TRUE, FALSE	FALSE	System
	PRCAL	Public variable	BOOL	Alarm output signal (TRUE: alarm FALSE: normal). Store TRUE once process or rate alarm occurs in any of CH1 to CH8 of CT input module.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	PCDAL	Public variable	BOOL	Peak current detection signal (TRUE: peak current detected FALSE: normal) Store TRUE if it detects peak current in any of CH1 to CH8 of CT input module.	TRUE, FALSE	FALSE	System
	PLAL1 to PLAL8	Public variable	BOOL	Low limit value alarm of CH1 to CH8 process alarm (TRUE: over FALSE: normal). Store TRUE of the channel if it surpasses the low limit value of process alarm.	TRUE, FALSE	FALSE	System
	PHAL1 to PHAL8	Public variable	BOOL	High limit value alarm of CH1 to CH8 process alarm (TRUE: over FALSE: normal). Store TRUE of the channel if it surpasses the high limit value of process alarm.	TRUE, FALSE	FALSE	System
	RTLAL1 to RTLAL8	Public variable	BOOL	Low limit alarm of CH1 to CH8 rate alarm (TRUE: over FALSE: normal). Store TRUE of the channel if it surpasses the low limit value of rate alarm.	TRUE, FALSE	FALSE	System
	RTHAL1 to RTHAL8	Public variable	BOOL	High limit value alarm of CH1 to CH8 rate alarm (TRUE: over FALSE: normal). Store TRUE of the channel if it surpasses the high limit value of rate alarm.	TRUE, FALSE	FALSE	System
	RGAL1 to RGAL8	Public variable	BOOL	CH1 to CH8 input signal abnormality detection (TRUE: abnormal FALSE: normal). Store TRUE of the channel if it detects the CT input value which surpasses the input range.	TRUE, FALSE	FALSE	System
	PCDAL1 to PCDAL8	Public variable	BOOL	Peak current detected (CH1 to CH8) (TRUE: detected FALSE: normal). Store TRUE of the channel if it detects peak current	TRUE, FALSE	FALSE	System
	ADENB	Public variable	WORD	Enable/disable conversion setting status Store conversion enabled/disabled setting status b15 to b7 b3 b2 b1 b0 0 ··· 0 | CH 8 | ··· | CH 4 | CH 3 | CH 2 | CH 1 0: Enable conversion 1: Disable conversion	0 to 00FFH	00FFH	System
	ERRCOD	Public variable	INT	Error code. Store error code detected by CT input module. For details of error code, refer to MELSEC-Q Current Transformer Input Module User's Manual.	–	0	System
	ALMENB1	Public variable	WORD	Alarm output enabled/disabled setting status. Stores alarm output enabled/disabled setting status. b15 to b10 b9 b8 b7 to b2 b1 b0 CH 8 | ··· | CH 3 | CH 2 | CH 1 | CH 8 | ··· | CH 3 | CH 2 | CH 1 b0 to b7: Process alarm setting b8 to b15: Rate alarm setting 0: Enable alarm output 1: Disable alarm output	0 to FFFFH	FFFFH	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	ALMENB2	Public variable	WORD	Input signal abnormality detection enabled/disabled setting status. Stores an input signal abnormality detection enabled/disabled setting status. b15 to b7 b3 b2 b1 b0 0 ··· 0 | CH 8 | ··· | CH 4 | CH 3 | CH 2 | CH 1 0: Enables fault detection 1: Disables fault detection	0 to 00FFH	00FFH	System
	ALMENB3	Public variable	WORD	Peak current detection enabled/disabled setting status Stores peak current detection enabled/disabled setting status. b15 to b7 b3 b2 b1 b0 0 ··· 0 | CH 8 | ··· | CH 4 | CH 3 | CH 2 | CH 1 0: Valid 1: Invalid	0 to 00FFH	00FFH	System
	SCLENB	Public variable	WORD	Scaling enabled/disabled setting status. Store scaling enabled/disabled setting status. b15 to b7 b3 b2 b1 b0 0 ··· 0 | CH 8 | ··· | CH 4 | CH 3 | CH 2 | CH 1 0: Valid 1: Invalid	0 to 00FFH	00FFH	System
	CH1DOUT to CH8DOUT	Public variable	INT	CH1 to CH8 Digital output value. Stores digital output values of each channel.	0 to 10000	0	System
	CH1MAX to CH8MAX	Public variable	INT	Maximum conversion value of CH1 to CH8. Stores the maximum output value of each channel. (*2)	Depending on the range of digital output value (0 to 1000) of a module or scaling function	0	System
	CH1MIN to CH8MIN	Public variable	INT	Minimum conversion value of CH1 to CH8. Stores the minimum output value of each channel. (*2)	Depending on the range of digital output value (0 to 1000) of a module or scaling function	0	System
	CH1PCDCNT to CH8PCDCNT	Public variable	INT	Peak current detected count (CH1 to CH8) Stores the number of peak current detection times for each channel.	0 to 32767	0	System

*1 The public variables CH1INH to CH8INH, AL1ENB to AL8ENB, or CHCNVENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

*2 With the scaling enable/disable setting, either digital output values or scaling value is selected for each channel and output from the output values of CH1 to CH8. The storage value is also switched between the maximum value and minimum value with the setting.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH4)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read digital output value from CT input module and output conversion value of each channel from output variable CH1 to CH8.</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH8 holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: conversion completed flag (batch by XnE)
</td></tr>
<tr><td>Others</td><td>For information about the processing, setting, Channel-by-channel conversion flag and conversion completed flag of corresponding module of this module FB, refer to the following manual:
● MELSEC-Q Current Transformer Input Module User's Manual</td></tr>
</table>

POINT

- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or programming is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

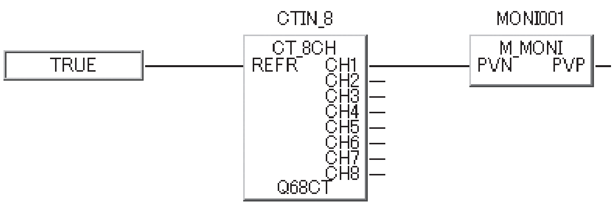
Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- Unable to communicate with CT input module. (Error code: 1412)
- Abnormality of CT input module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example

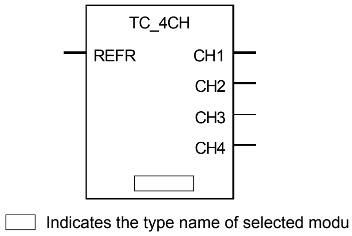
(User-created variable) (Public variable of module FB)



If STBREQ is TRUE, operation condition setting request of CTIN_8 (STB) will be executed.

10.2 Temperature Input Module FB

10.2.1 4 Channels Thermocouple Input (TC_4CH)

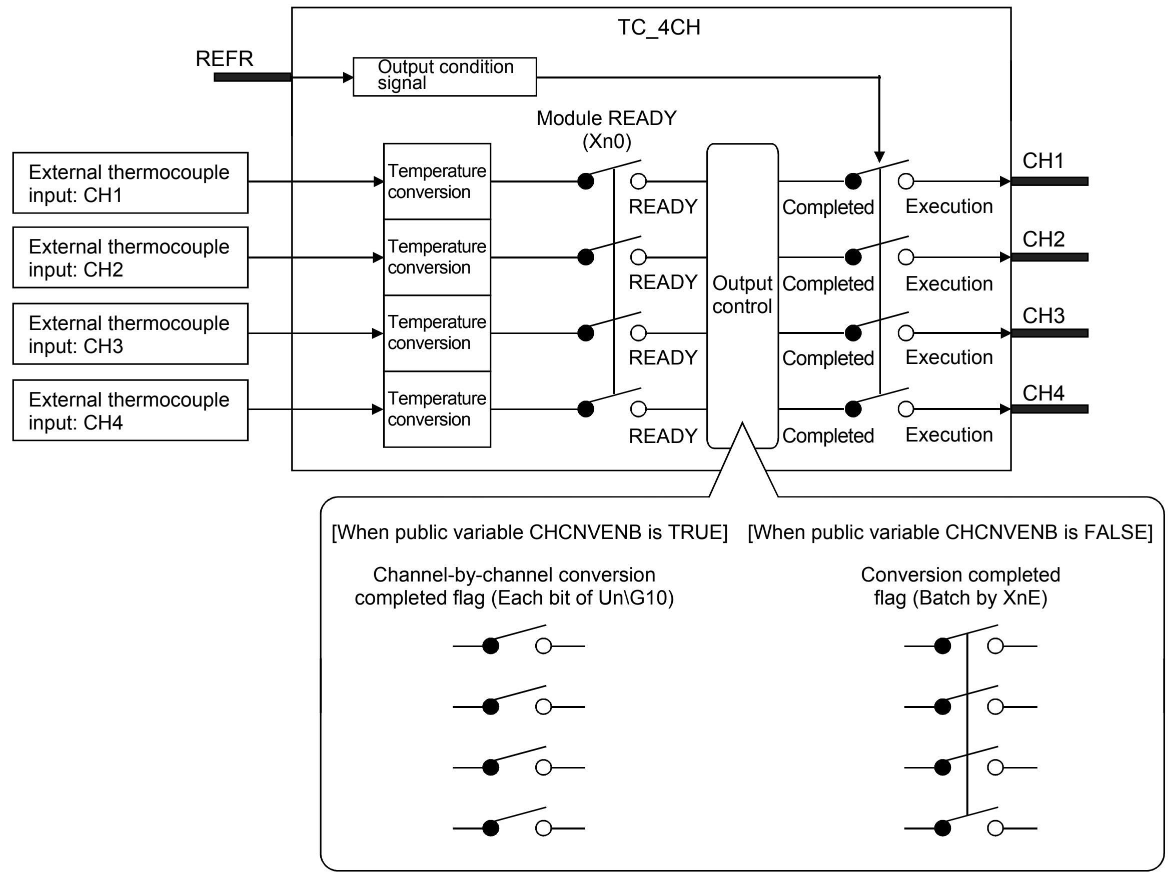
FB	FBD parts
TC_4CH	

Corresponding module
Q64TD

Function overview: Read temperature conversion value of 4 channels temperature input module that converts the thermocouple signal to digital value, and output it from output variable (CH1 to CH4).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH4	Output variable	REAL	CH1 to CH4 temperature conversion value *1	Depends on the input range and resolution mode.

*1 The buffer memory "temperature process variable (address 11 to 14)" of the module multiplies the value of the first decimal place of the measured temperature by 10, and stores as 16-bit signed binary format. The output variable "temperature conversion value" divides the value of buffer memory by 10, and outputs as 32-bit real number (single precision floating decimal).

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: –) When module error (ERR) is TRUE, set ERRC as TRUE to clear module error and make ERR FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH4INH	Public variable	BOOL	Enable/disable conversion (CH1 to CH4). (TRUE: Disabled FALSE: Enabled). Set the output of temperature conversion value enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL4ENB	Public variable	BOOL	Enable/disable alarm output (CH1 to CH4). (TRUE: Output enabled FALSE: Output disabled) Set the alarm output enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute conversion enabled/disabled setting (CH1INH to CH4INH) and alarm output enabled/disabled setting (AL1ENB to AL4ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). Store the status of module READY (Xn0). Perform temperature conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the temperature value. When the setting is TRUE, the channel-by-channel conversion completed flag is used as the output condition. When the setting is FALSE, the conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	Conversion completed flag (TRUE: ON FALSE: OFF) Store the status of conversion completion flag (XnE)	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag. (TRUE: Error FALSE: No error) Store TRUE when writing error occurs.	TRUE, FALSE	FALSE	System
	BNAL	Public variable	BOOL	Wire break detection signal. (TRUE: Disconnection detected FALSE: Normal) Store TRUE when one of CH1 to CH4 of temperature input module is disconnected.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	PRCAL	Public variable	BOOL	Alarm output signal. (TRUE: Alarm occurs FALSE: Normal) Store TRUE when one of CH1 to CH4 of temperature input module exceeds the range of high/lower limit.	TRUE, FALSE	FALSE	System
	PLAL1 to PLAL4	Public variable	BOOL	Low limit value alarm of CH1 to CH4 process alarm (TRUE: Over FALSE: Normal) Store TRUE in channel if it exceeds the low limit value set.	TRUE, FALSE	FALSE	System
	PHAL1 to PHAL4	Public variable	BOOL	High limit value alarm of CH1 to CH4 process alarm (TRUE: Over FALSE: Normal) Store TRUE in channel when it exceeds the high limit value set.	TRUE, FALSE	FALSE	System
	BNOUT1 to BNOUT4	Public variable	BOOL	CH1 to CH4 wire break detection flag (TRUE: Disconnection detected FALSE: Normal) Store TRUE in channel if disconnection occurs.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	INT	Enable/disable conversion setting status. Store the setting status of conversion enabled/disabled. b15 to b3 b2 b1 b0 [(b15 to b4) | CH 4 | CH 3 | CH 2 | CH 1] 0: Conversion enabled 1: Conversion disabled	0 to 15	0	System
	ERRCOD	Public variable	INT	Error code. Store the detected error code of temperature input module. For detailed information about the error code, refer to the Thermocouple Input Module Channel Isolated Thermocouple/Micro Voltage Input Module User's Manual.	–	0	System
	ALMENB	Public variable	INT	Alarm output enabled/disabled setting status. Store alarm output enabled/disabled setting status. b15 to b3 b2 b1 b0 [(b15 to b4) | CH 4 | CH 3 | CH 2 | CH 1] 0: Alarm output disabled 1: Alarm output enabled	0 to 15	0	System
	CH1SCAL to CH4SCAL	Public variable	INT	CH1 to CH4 scaling value (%) Store the scaled value of scaling high/low limit value through high/low limit value. (Refer to the Thermocouple Input Module Channel Isolated Thermocouple/Micro Voltage Input Module User's Manual.)	-32768 to 32767	0	System

*1 The public variables CH1INH to CN4INH, AL1ENB to AL4ENB, or CHCNVENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH4)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read temperature conversion value from temperature input module and output digital output the value in channels from output variable (CH1 to CH4).</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable (CH1 to CH4) holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: Conversion completed flag (batch by XnE)
</td></tr>
<tr><td>Others</td><td>For detailed information of all the processing, settings, channel-by-channel conversion flag and conversion completed flag of the corresponding module of this module FB, refer to the following manual.
● Thermocouple Input Module Channel Isolated Thermocouple/Micro Voltage Input Module User's Manual.</td></tr>
</table>

POINT

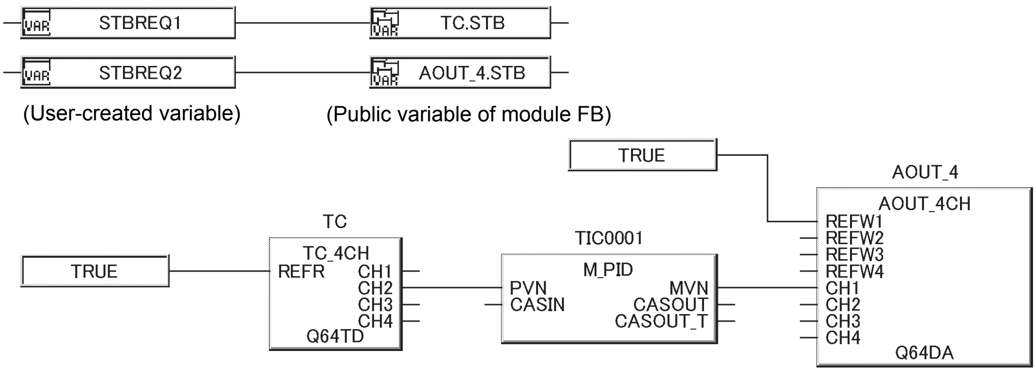
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-TI is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

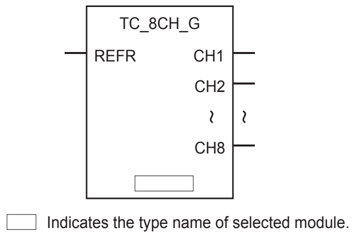
- Unable to communicate with temperature input module. (Error code: 1412)
- Abnormality of temperature input module has been detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request (STB) of TC will be executed.
If STBREQ2 is TRUE, operation condition setting request (STB) of AOUT_4 will be executed.

10.2.2 Channels-isolated 8 Channels Thermocouple Input (TC_8CH_G)

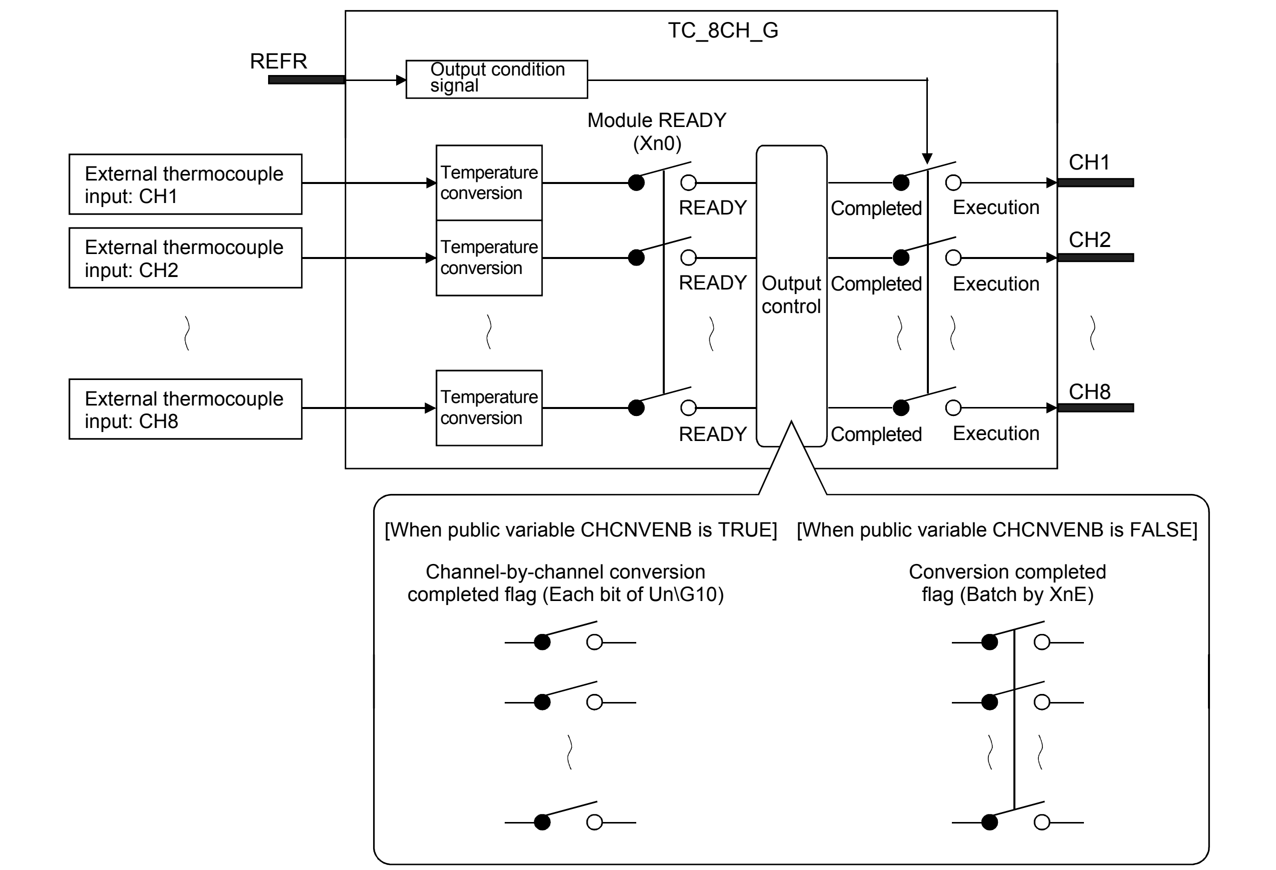
FB	FBD parts
TC_8CH_G	

Corresponding module
Q68TD-G-H01, Q68TD-G-H02

Function overview: Read temperature conversion value of 8 channels temperature input module that converts the thermocouple signal to digital value, and output it from output variable (CH1 to CH8).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH8	Output variable	REAL	CH1 to CH8 temperature conversion value *1	Depends on the input range and resolution mode.

*1 The buffer memory "temperature process variable (address 11 to 18)" of the module multiplies the value of the first decimal place of the measured temperature by 10, and stores as 16-bit signed binary format. The output variable "temperature conversion value" divides the value of buffer memory by 10, and outputs as 32-bit real number (single precision floating decimal).

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: –) When module error (ERR) is TRUE, set ERRC as TRUE to clear module error and make ERR FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH8INH	Public variable	BOOL	Enable/disable conversion (CH1 to CH8). (TRUE: Disabled FALSE: Enabled). Set the output of temperature conversion value enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL8ENB	Public variable	BOOL	Enable/disable alarm output (CH1 to CH8). (TRUE: Output enabled FALSE: Output disabled) Set the alarm output enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute conversion enabled/disabled setting (CH1INH to CH8INH) and alarm output enabled/disabled setting (AL1ENB to AL8ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). Store the status of module READY (Xn0). Perform temperature conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the temperature value. When the setting is TRUE, the channel-by-channel conversion completed flag is used as the output condition. When the setting is FALSE, the conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	Conversion completed flag (TRUE: ON FALSE: OFF) Store the status of conversion completion flag (XnE)	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag. (TRUE: Error FALSE: No error) Store TRUE when writing error occurs.	TRUE, FALSE	FALSE	System
	BNAL	Public variable	BOOL	Wire break detection signal. (TRUE: Disconnection detected FALSE: Normal) Store TRUE when one of CH1 to CH8 of temperature input module is disconnected.	TRUE, FALSE	FALSE	System
	PRCAL	Public variable	BOOL	Alarm output signal. (TRUE: Alarm occurs FALSE: Normal) Store TRUE once process or rate alarm occurs in any of CH1 to CH8 of temperature input module.	TRUE, FALSE	FALSE	System
	PLAL1 to PLAL8	Public variable	BOOL	Low limit value alarm of CH1 to CH8 process alarm (TRUE: Over FALSE: Normal) Store TRUE in channel if it exceeds the low limit value set.	TRUE, FALSE	FALSE	System
	PHAL1 to PHAL8	Public variable	BOOL	High limit value alarm of CH1 to CH8 process alarm (TRUE: Over FALSE: Normal) Store TRUE in channel when it exceeds the high limit value set.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	RTLAL1 to RTLAL8	Public variable	BOOL	Low limit alarm of CH1 to CH8 rate alarm (TRUE: over FALSE: normal) Store TRUE of the channel if it surpasses the low limit value of rate alarm.	TRUE, FALSE	FALSE	System
	RTHAL1 to RTHAL8	Public variable	BOOL	High limit alarm of CH1 to CH8 rate alarm (TRUE: over FALSE: normal) Store TRUE of the channel if it surpasses the high limit value of rate alarm.	TRUE, FALSE	FALSE	System
	BNOUT1 to BNOUT8	Public variable	BOOL	CH1 to CH8 wire break detection flag (TRUE: Disconnection detected FALSE: Normal) Store TRUE in channel if disconnection occurs.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	WORD	Enable/disable conversion setting status. Store the setting status of conversion enabled/disabled. b15 to b7 to b3 b2 b1 b0 0 ... 0 | CH 8 | ... | CH 4 | CH 3 | CH 2 | CH 1 0: Conversion enabled 1: Conversion disabled	0 to 00FFH	0H	System
	ERRCOD	Public variable	INT	Error code. Store the detected error code of temperature input module. For detailed information about the error code, refer to the Thermocouple Input Module Channel Isolated Thermocouple/Micro Voltage Input Module User's Manual.	–	0	System
	ALMENB	Public variable	WORD	Alarm output enabled/disabled setting status. Store alarm output enabled/disabled setting status. b15 to b10 b9 b8 b7 to b2 b1 b0 CH 8 | ... | CH 3 | CH 2 | CH 1 | CH 8 | ... | CH 3 | CH 2 | CH 1 b0 to b7: Process alarm setting b8 to b15: Rate alarm setting 0: Alarm output disabled 1: Alarm output enabled	0 to FFFFH	0H	System
	SCLENB	Public variable	WORD	Scaling enabled/disabled setting status. Store scaling enabled/disabled setting status. b15 to b7 to b3 b2 b1 b0 0 ... 0 | CH 8 | ... | CH 4 | CH 3 | CH 2 | CH 1 0: Valid 1: Invalid	0 to 00FFH	0H	System
	CH1SCAL to CH8SCAL	Public variable	INT	CH1 to CH8 scaling value (%) Store the scaled value of scaling high/low limit value through high/low limit value. (Refer to the Thermocouple Input Module Channel Isolated Thermocouple/Micro Voltage Input Module User's Manual.)	-32768 to 32767	0	System

*1 The public variables CH1INH to CN8INH, AL1ENB to AL8ENB, or CHCNVENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH8)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read temperature conversion value from temperature input module and output digital output the value in channels from output variable (CH1 to CH8).</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable (CH1 to CH8) holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: Conversion completed flag (batch by XnE)
</td></tr>
<tr><td>Others</td><td>For detailed information of all the processing, settings, channel-by-channel conversion flag and conversion completed flag of the corresponding module of this module FB, refer to the following manual.
● Thermocouple Input Module Channel Isolated Thermocouple/Micro Voltage Input Module User's Manual.</td></tr>
</table>

POINT

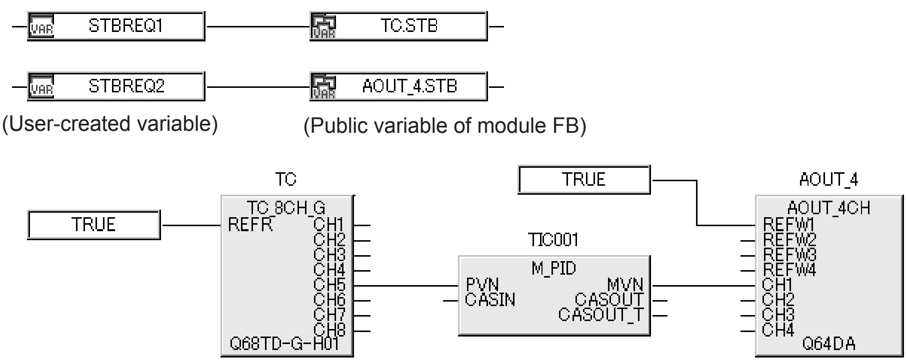
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-TI is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

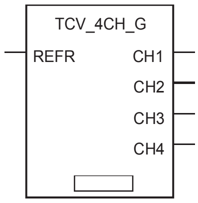
- Unable to communicate with temperature input module. (Error code: 1412)
- Abnormality of temperature input module has been detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request (STB) of TC will be executed.
If STBREQ2 is TRUE, operation condition setting request (STB) of AOUT_4 will be executed.

10.2.3 Channel-isolated 4 Channels Temperature/Micro-voltage Input (TCV_4CH_G)

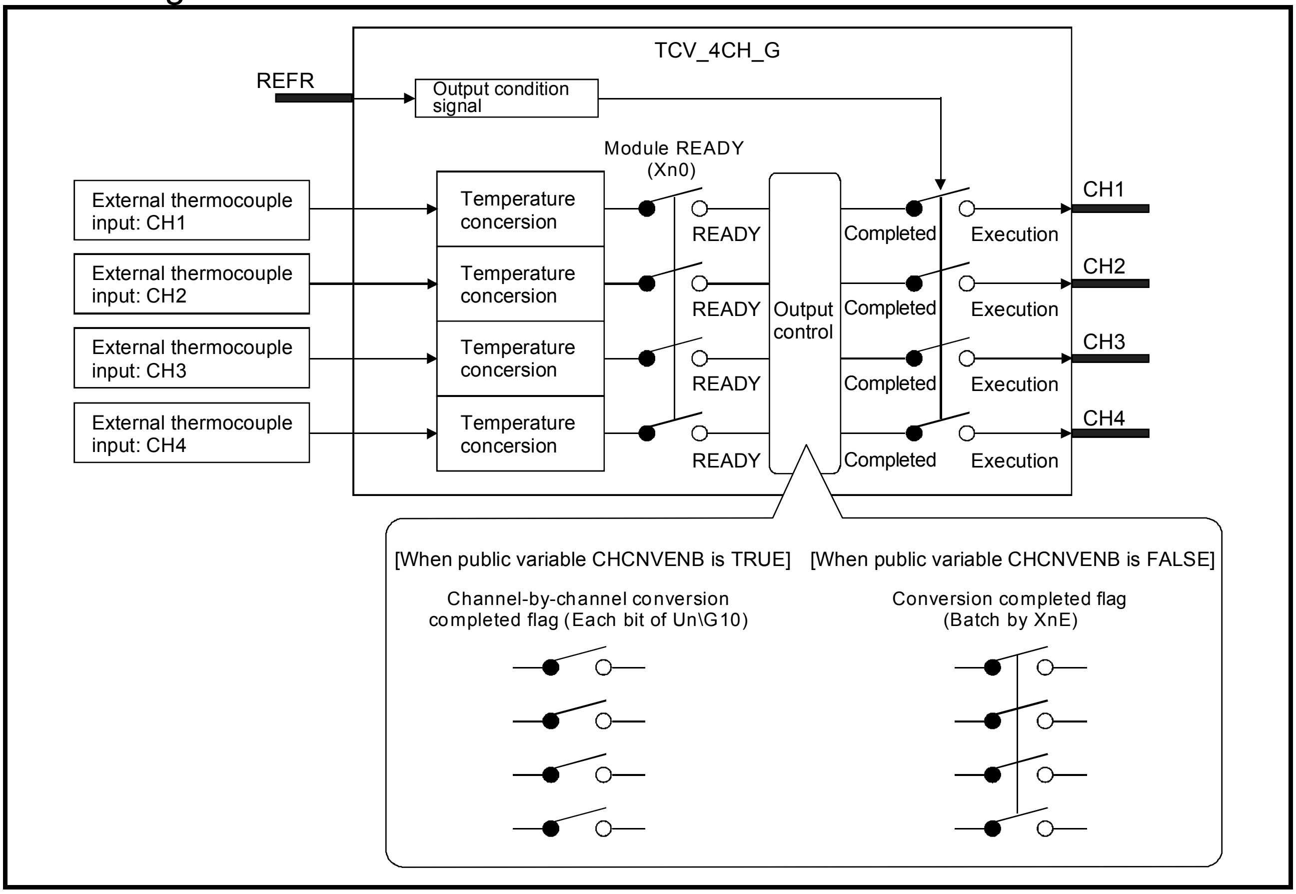
FB	FBD parts
TCV_4CH_G	 Indicates the type name of selected module.

Corresponding module
Q64TDV-GH

Function overview: Read the digital signal of 4 channels channel-isolated temperature/micro-voltage input module that converts the thermocouple temperature signal or micro-voltage signal to digital value, and output it from output variable (CH1 to CH4).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH4	Output variable	REAL	CH1 to CH4 temperature process variable/micro voltage conversion value *1	Depends on the input range and resolution mode.

*1 The buffer memory "temperature process variable (address 11 to 14)" of the module multiplies the value of the first decimal place of the measured temperature by 10, and stores as 16-bit signed binary format.
The output variable "temperature conversion value" divides the value of buffer memory by 10, and outputs as 32-bit real number (single precision floating decimal).
The micro voltage conversion value is output as 32-bit real number.

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: –) When module error (ERR) is TRUE, set ERRC as TRUE to clear module error and make ERR FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH4INH	Public variable	BOOL	Enable/disable conversion (CH1 to CH4). (TRUE: Disabled FALSE: Enabled) Set temperature process variable/micro voltage conversion value output enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL4ENB	Public variable	BOOL	Enable/disable alarm output (CH1 to CH4). (TRUE: Output enabled FALSE: Output disabled) Set alarm output enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request. Execute conversion enabled/disabled setting (CH1INH to CH4INH) and alarm output enabled/disabled setting (AL1ENB to AL4ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). Store the status of module READY (Xn0). Perform temperature conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the temperature process variable/micro voltage conversion value. When the setting is TRUE, the channel-by-channel conversion completed flag is used as the output condition. When the setting is FALSE, the conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	Conversion completion flag (TRUE: ON FALSE: OFF) Store the status of conversion completion flag (XnE)	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: No error) Store TRUE when writing error occurs.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	BNAL	Public variable	BOOL	Wire break detection signal (TRUE: Disconnection detected FALSE: Normal) Store TRUE when one of CH1 to CH4 of temperature input module is disconnected.	TRUE, FALSE	FALSE	System
	PRCAL	Public variable	BOOL	Alarm output signal (TRUE: Alarm occurs FALSE: Normal) Store TRUE when one of CH1 to CH4 of temperature input module exceeds the high/low limit set.	TRUE, FALSE	FALSE	System
	PLAL1 to PLAL4	Public variable	BOOL	Low limit value alarm of CH1 to CH4 process alarm (TRUE: Over FALSE: Normal) Store TRUE in channel if it exceeds the low limit value set.	TRUE, FALSE	FALSE	System
	PHAL1 to PHAL4	Public variable	BOOL	High limit value alarm of CH1 to CH4 process alarm (TRUE: Over FALSE: Normal) Store TRUE in channel when it exceeds the high limit value set.	TRUE, FALSE	FALSE	System
	BNOUT1 to BNOUT4	Public variable	BOOL	CH1 to CH4 wire break detection flag (TRUE: Wire break detected FALSE: Normal) Store TRUE in channel if disconnection occurs.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	INT	Enable/disable conversion setting status. Store the setting status of conversion enabled/disabled. b15 to b3 b2 b1 b0 CH4 CH3 CH2 CH1 0: Enable A/D conversion 1: Disable A/D conversion	0 to 15	0	System
	ERRCOD	Public variable	INT	Error code. Store error code detected by temperature input module. For details of error code about the error code, refer to the Thermocouple Input Module Channel Isolated Thermocouple/Micro Voltage Input Module User's Manual.	–	0	System
	ALMENB	Public variable	INT	Alarm output enabled/disabled setting status. Store alarm output enabled/disabled setting status. b15 to b3 b2 b1 b0 CH4 CH3 CH2 CH1 0: Alarm output disabled 1: Alarm output enabled	0 to 15	0	System
	CH1SCAL to CH4SCAL	Public variable	INT	CH1 to CH4 scaling value (%) Store the scaled value of scaling high/low limit value. (Refer to the Thermocouple Input Module Channel Isolated Thermocouple/Micro Voltage Input Module User's Manual.)	-32768 to 32767	0	System

*1 The public variables CH1INH to CN4INH, AL1ENB to AL4ENB, or CHCNVENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH4)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read temperature process variable/micro voltage conversion value from temperature input module and output temperature process variable/micro voltage conversion value from output variable CH1 to CH4 in channel.</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH4 holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: Conversion completed flag (batch by XnE)
</td></tr>
<tr><td>Others</td><td>For detailed information of all the processing, settings, channel-by-channel conversion flag and conversion completed flag of the corresponding module of this module FB, refer to the following manual.
● Thermocouple Input Module Channel Isolated Thermocouple/Micro Voltage Input Module User's Manual.</td></tr>
</table>

POINT

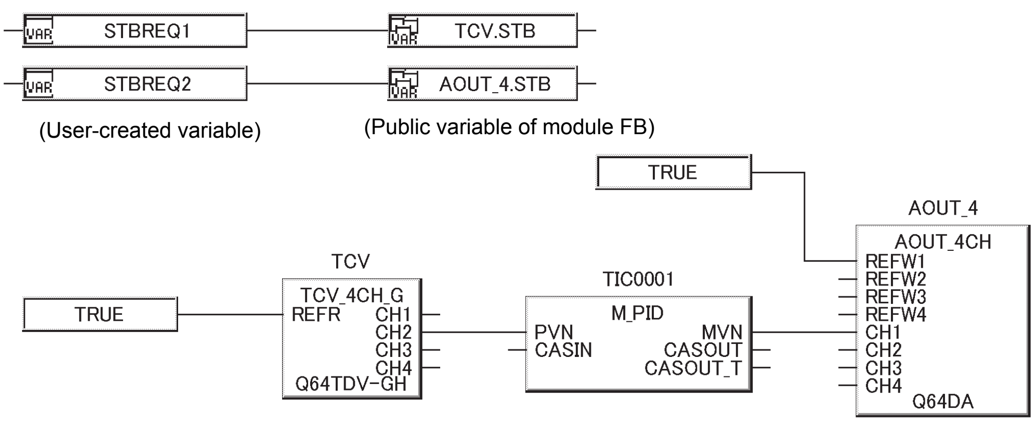
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-TI is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

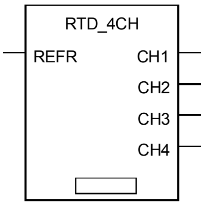
- Unable to communicate with temperature input module. (Error code: 1412)
- Abnormality of temperature input module has been detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request (STB) of TCV will be executed.
If STBREQ2 is TRUE, operation condition setting request (STB) of AOUT_4 will be executed.

10.2.4 4 Channels Temperature Input (RTD_4CH)

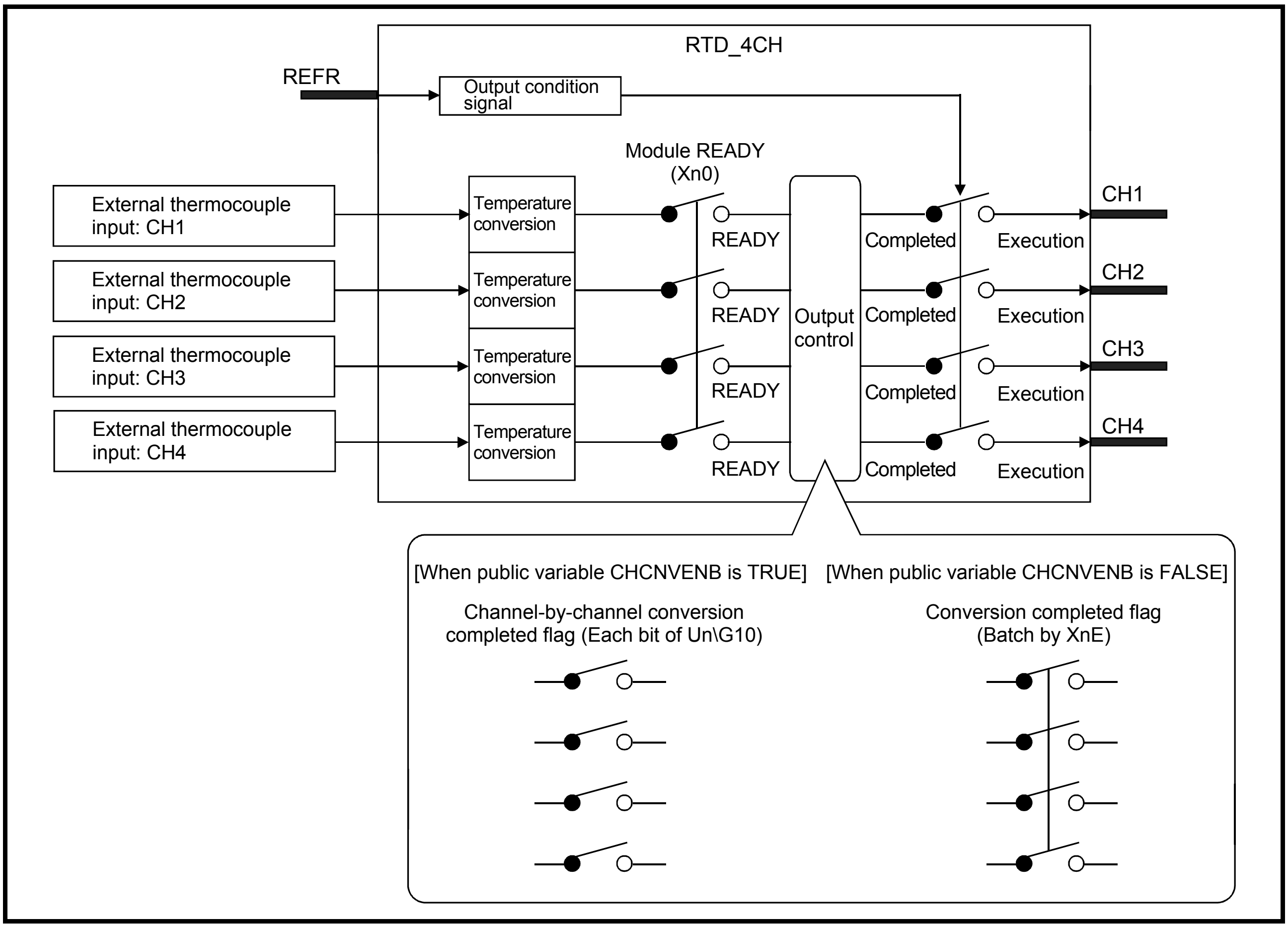
FB	FBD parts
RTD_4CH	 ▭ Indicates the type name of selected module.

Corresponding module
Q64RD, Q64RD-G

Function overview: Read the temperature conversion value of 4 channels temperature input module that converts temperature - measuring resistor temperature signal to digital value, and output it from output variable (CH1 to CH4).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH4	Output variable	REAL	CH1 to CH4 temperature conversion value *1	Depends on the input range and resolution mode.

*1 The buffer memory "temperature process variable (32-bit, address 54 to 61)" of the module multiplies the value of the third decimal place of the measured temperature by 1000, and stores as 32-bit signed binary format.
The output variable "temperature conversion value" divides the value of buffer memory by 1000, and outputs as 32-bit real number (single precision floating decimal).

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: –) When module error (ERR) is TRUE, set ERRC as TRUE to clear module error and make ERR FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH4INH	Public variable	BOOL	Enable/disable conversion (CH1 to CH4). (TRUE: Disabled FALSE: Enabled) Set the output of temperature conversion value enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL4ENB	Public variable	BOOL	Enable/disable alarm output (CH1 to CH4). (TRUE: Output enabled FALSE: Output disabled) Set the alarm output enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request Execute conversion enabled/disabled setting (CH1INH to CH4INH) and alarm output enabled/disabled setting (AL1ENB to AL4ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). Store the status of module READY (Xn0). Perform temperature conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the temperature value. When the setting is TRUE, the channel-by-channel conversion completed flag is used as the output condition. When the setting is FALSE, the conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	Conversion completed flag (TRUE: ON FALSE: OFF) Store the status of conversion completion flag (XnE)	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: No error) Store TRUE when writing error occurs.	TRUE, FALSE	FALSE	System
	BNAL	Public variable	BOOL	Wire break detection signal. (TRUE: Disconnection detected FALSE: Normal) Store TRUE when one of CH1 to CH4 of temperature input module is disconnected.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	PRCAL	Public variable	BOOL	Alarm output signal. (TRUE: Alarm occurs FALSE: Normal) Store TRUE when one of CH1 to CH4 of temperature input module exceeds the set range of high/low limit value.	TRUE, FALSE	FALSE	System
	PLAL1 to PLAL4	Public variable	BOOL	Low limit value alarm of CH1 to CH4 process alarm (TRUE: Over FALSE: Normal) Store TRUE in channel if it exceeds the set low limit value.	TRUE, FALSE	FALSE	System
	PHAL1 to PHAL4	Public variable	BOOL	High limit value alarm of CH1 to CH4 process alarm (TRUE: Over FALSE: Normal) Store TRUE in channel when it exceeds the high limit value.	TRUE, FALSE	FALSE	System
	BNOUT1 to BNOUT4	Public variable	BOOL	CH1 to CH4 wire break detection flag (TRUE: Wire break detected FALSE: Normal) Store TRUE in channel if disconnection occurs.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	INT	Enable/disable conversion setting status. Store the setting status of conversion enabled/disabled. b15 to b3 b2 b1 b0 CH4 CH3 CH2 CH1 0: Enable conversion 1: Disable conversion	0 to 15	0	System
	ERRCOD	Public variable	INT	Error code. Store error code detected by temperature input module. Refer to the RTD Input Module Channel Isolated RTD Input Module User's Manual.	-	0	System
	ALMENB	Public variable	INT	Alarm output enabled/disabled setting status. Store alarm output enabled/disabled setting status. b15 to b3 b2 b1 b0 CH4 CH3 CH2 CH1 0: Alarm output disabled 1: Alarm output enabled	0 to 15	0	System
	CH1SCAL to CH4SCAL	Public variable	INT	CH1 to CH4 scaling value (%) Store the scaled value of scaling high/low limit value. (Refer to the RTD Input Module Channel Isolated RTD Input Module User's Manual.)	-32768 to 32767	0	System

*1 The public variables CH1INH to CN4INH, AL1ENB to AL4ENB, or CHCNVENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH4)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read temperature conversion value from temperature input module and output digital output the value in channels from output variable (CH1 to CH4).</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH4 holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: Conversion completed flag (batch by XnE)
</td></tr>
<tr><td>Others</td><td>For detailed information of all the processing, settings, channel-by-channel conversion flag and conversion completed flag of the corresponding module of this module FB, refer to the following manual.
● RTD Input Module Channel Isolated RTD Input Module User's Manual.</td></tr>
</table>

POINT

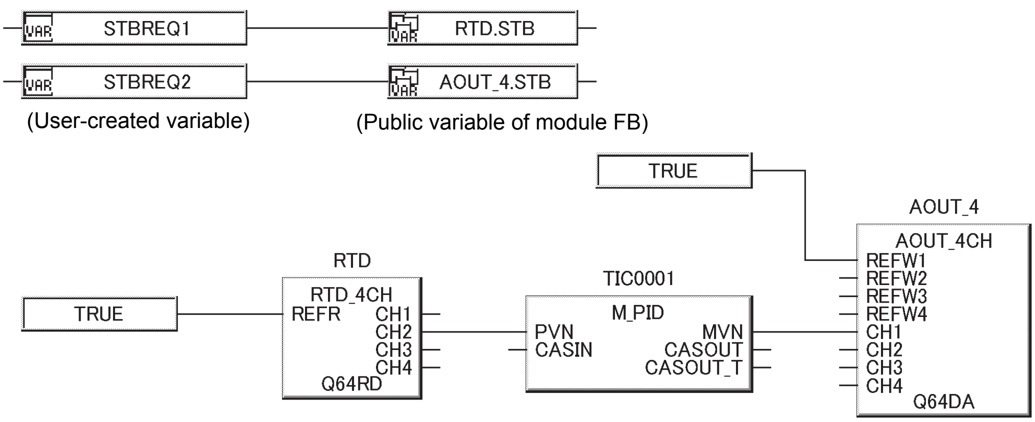
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-TI is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

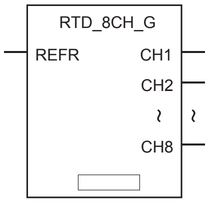
- Unable to communicate with temperature input module. (Error code: 1412)
- Abnormality of temperature input module has been detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



If STBREQ1 is TRUE, operation condition setting request (STB) of RTD will be executed.
If STBREQ2 is TRUE, operation condition setting request (STB) of AOUT_4will be executed.

10.2.5 Channel-isolated 8 Channels Temperature-Measuring Resistor Input (RTD_8CH_G)

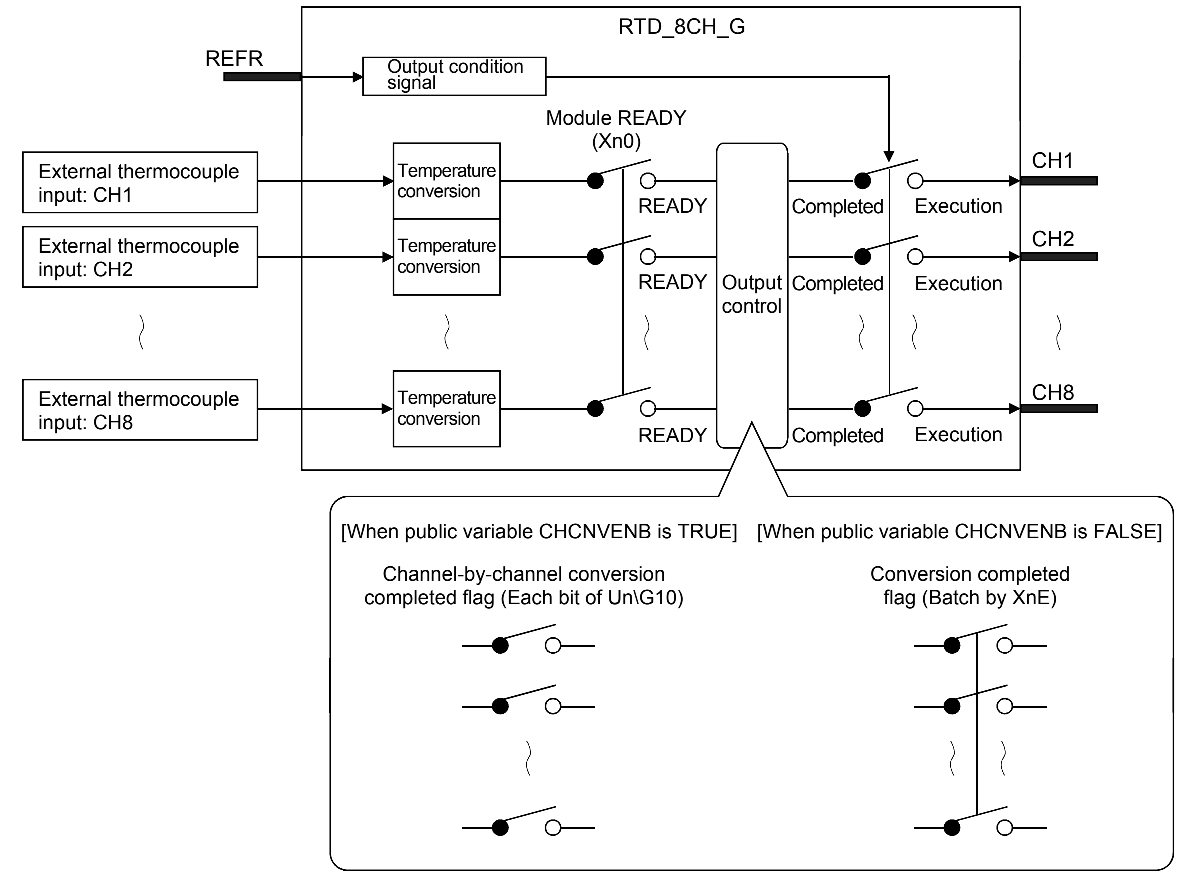
FB	FBD parts
RTD_8CH_G	 Indicates the type name of selected module.

Corresponding module
Q68RD3-G

Function overview: Read the temperature conversion value of 8 channels temperature input module that converts temperature - measuring resistor temperature signal to digital value, and output it from output variable (CH1 to CH8).

Function/FB classification name: Module FB

Block Diagram



Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Output condition signal (TRUE: Execute, FALSE: Stop)	TRUE, FALSE
Output	CH1 to CH8	Output variable	REAL	CH1 to CH8 temperature conversion value *1	Depends on the input range and resolution mode.

*1 The buffer memory "temperature process variable (address 11 to 18)" of the module multiplies the value of the first decimal place of the measured temperature by 10, and stores as 16-bit signed binary format. The output variable "temperature conversion value" divides the value of buffer memory by 10, and outputs as 32-bit real number (single precision floating decimal).

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC	Public variable	BOOL	Error clear request (TRUE: Error clear request FALSE: –) When module error (ERR) is TRUE, set ERRC as TRUE to clear module error and make ERR FALSE.	TRUE, FALSE	FALSE	User
	CH1INH to CH8INH	Public variable	BOOL	Enable/disable conversion (CH1 to CH8). (TRUE: Disabled FALSE: Enabled) Set the output of temperature conversion value enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	AL1ENB to AL8ENB	Public variable	BOOL	Enable/disable alarm output (CH1 to CH8). (TRUE: Output enabled FALSE: Output disabled) Set the alarm output enabled/disabled in channels. It is valid when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	STB	Public variable	BOOL	Operation condition setting request Execute conversion enabled/disabled setting (CH1INH to CH8INH) and alarm output enabled/disabled setting (AL1ENB to AL4ENB) when STB transforms from FALSE to TRUE.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: ON FALSE: OFF). Store the status of module READY (Xn0). Perform temperature conversion processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	CHCNVENB	Public variable	BOOL	Enable conversion completed flag for each channel. Set the output condition of the temperature value. When the setting is TRUE, the channel-by-channel conversion completed flag is used as the output condition. When the setting is FALSE, the conversion completed flag (CNVCMPL) is used as the output condition.	TRUE, FALSE	FALSE	User
	CNVCMPL	Public variable	BOOL	Conversion completed flag (TRUE: ON FALSE: OFF) Store the status of conversion completion flag (XnE)	TRUE, FALSE	FALSE	System
	ERR	Public variable	BOOL	Error flag (TRUE: Error FALSE: No error) Store TRUE when writing error occurs.	TRUE, FALSE	FALSE	System
	BNAL	Public variable	BOOL	Wire break detection signal. (TRUE: Disconnection detected FALSE: Normal) Store TRUE when one of CH1 to CH8 of temperature input module is disconnected.	TRUE, FALSE	FALSE	System
	PRCAL	Public variable	BOOL	Alarm output signal. (TRUE: Alarm occurs FALSE: Normal) Store TRUE once process or rate alarm occurs in any of CH1 to CH8 of temperature input module.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	PLAL1 to PLAL8	Public variable	BOOL	Low limit value alarm of CH1 to CH8 process alarm (TRUE: Over FALSE: Normal) Store TRUE in channel if it exceeds the low limit value set.	TRUE, FALSE	FALSE	System
	PHAL1 to PHAL8	Public variable	BOOL	High limit value alarm of CH1 to CH8 process alarm (TRUE: Over FALSE: Normal) Store TRUE in channel when it exceeds the high limit value set.	TRUE, FALSE	FALSE	System
	RTLAL1 to RTLAL8	Public variable	BOOL	Low limit alarm of CH1 to CH8 rate alarm (TRUE: over FALSE: normal) Store TRUE of the channel if it surpasses the low limit value of rate alarm.	TRUE, FALSE	FALSE	System
	RTHAL1 to RTHAL8	Public variable	BOOL	High limit alarm of CH1 to CH8 rate alarm (TRUE: over FALSE: normal) Store TRUE of the channel if it surpasses the high limit value of rate alarm.	TRUE, FALSE	FALSE	System
	BNOUT1 to BNOUT8	Public variable	BOOL	CH1 to CH8 wire break detection flag (TRUE: Wire break detected FALSE: Normal) Store TRUE in channel if disconnection occurs.	TRUE, FALSE	FALSE	System
	ADENB	Public variable	WORD	Enable/disable conversion setting status. Store the setting status of conversion enabled/disabled. b15 to b7 to b3 b2 b1 b0 [0 . . . 0] [CH 8] [. . .] [CH 4] [CH 3] [CH 2] [CH 1] 0: Enable conversion 1: Disable conversion	0 to 00FFH	0H	System
	ERRCOD	Public variable	INT	Error code. Store error code detected by temperature input module. Refer to the RTD Input Module Channel Isolated RTD Input Module User's Manual.	-	0	System
	ALMENB	Public variable	WORD	Alarm output enabled/disabled setting status. Store alarm output enabled/disabled setting status. b15 to b10 b9 b8 b7 to b2 b1 b0 [CH 8] [. . .] [CH 3] [CH 2] [CH 1] [CH 8] [. . .] [CH 3] [CH 2] [CH 1] b0 to b7: Procedure alarm setting b8 to b15: Rate alarm setting 0: Alarm output disabled 1: Alarm output enabled	0 to FFFFH	0H	System
	SCLENB	Public variable	WORD	Scaling enabled/disabled setting status. Store scaling enabled/disabled setting status. b15 to b7 to b3 b2 b1 b0 [0 . . . 0] [CH 8] [. . .] [CH 4] [CH 3] [CH 2] [CH 1] 0: Valid 1: Invalid	0 to 00FFH	0H	System
	CH1SCAL to CH8SCAL	Public variable	INT	CH1 to CH8 scaling value (%) Store the scaled value of scaling high/low limit value. (Refer to the RTD Input Module Channel Isolated RTD Input Module User's Manual.)	-32768 to 32767	0	System

*1 The public variables CH1INH to CN8INH, AL1ENB to AL8ENB, or CHCNVENB can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="3">Condition</th><th rowspan="2">Output (CH1 to CH8)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Module READY (Xn0)</th><th>(*1)</th></tr>
<tr><td rowspan="3">TRUE</td><td rowspan="2">TRUE</td><td>TRUE</td><td>Read temperature conversion value from temperature input module and output digital output the value in channels from output variable (CH1 to CH8).</td></tr>
<tr><td>FALSE</td><td rowspan="3">Output variable CH1 to CH8 holds the previous value.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td><td>TRUE or FALSE</td></tr>
</table>
*1 When the public variable CHCNVENB is TRUE : Channel-by-channel conversion flag (each bit of buffer memory address 10)
When the public variable CHCNVENB is FALSE: Conversion completed flag (batch by XnE)
</td></tr>
<tr><td>Others</td><td>For detailed information of all the processing, settings, channel-by-channel conversion flag and conversion completed flag of the corresponding module of this module FB, refer to the following manual.
● RTD Input Module Channel Isolated RTD Input Module User's Manual.</td></tr>
</table>

POINT

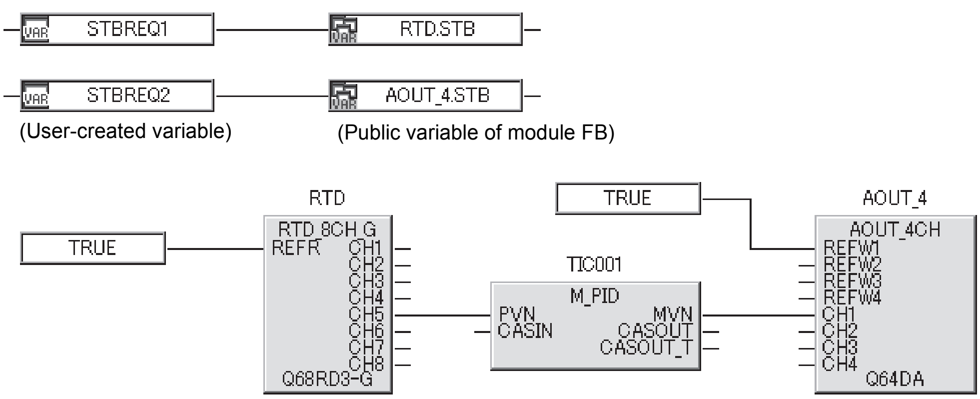
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-TI is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- Unable to communicate with temperature input module. (Error code: 1412)
- Abnormality of temperature input module has been detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

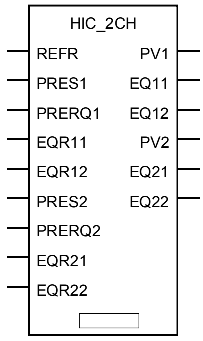
Program Example



If STBREQ1 is TRUE, operation condition setting request (STB) of RTD will be executed.
If STBREQ2 is TRUE, operation condition setting request (STB) of AOUT_4 will be executed.

10.3 Counter Module FB

10.3.1 High-speed Counter (HIC_2CH)

FB	FBD part
HIC_2CH	 Indicates the type name of selected module.

Corresponding module
QD62, QD62E, QD62D

Function overview: Read the pulse counter value and coincidence signal of high-speed counter module and output them from output variable.

Function/FB classification name: Module FB

Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	Refreshing request (TRUE: Request FALSE: No request)	TRUE, FALSE
	PRES1	Input variable	DINT	CH1 preset value	-2147483648 to 2147483647
	PRERQ1	Input variable	BOOL	CH1 preset command (TRUE: Command FALSE: No command)	TRUE, FALSE
	EQR11	Input variable	BOOL	CH1 coincidence signal No. 1 reset command (TRUE: Command FALSE: No command)	TRUE, FALSE
	EQR12	Input variable	BOOL	CH1 coincidence signal No.2 reset command (TRUE: Command FALSE: No command)	TRUE, FALSE
	PRES2	Input variable	DINT	CH2 preset value	-2147483648 to 2147483647
	PRERQ2	Input variable	BOOL	CH2 preset command (TRUE: Command FALSE: No command)	TRUE, FALSE
	EQR21	Input variable	BOOL	CH2 coincidence signal No.1 reset command (TRUE: Command FALSE: No command)	TRUE, FALSE
	EQR22	Input variable	BOOL	CH2 coincidence signal No.2 reset command (TRUE: Command FALSE: No command)	TRUE, FALSE
Output	PV1	Output variable	DINT	CH1 current value	-2147483648 to 2147483647
	EQ11	Output variable	BOOL	CH1 coincidence signal No.1 (TRUE: Coincident FALSE: Not coincident)	TRUE, FALSE
	EQ12	Output variable	BOOL	CH1 coincidence signal No.2 (TRUE: Coincident FALSE: Not coincident)	TRUE, FALSE
	PV2	Output variable	DINT	CH2 current value	-2147483648 to 2147483647
	EQ21	Output variable	BOOL	CH2 coincidence signal No.1 (TRUE: Coincident FALSE: Not coincident)	TRUE, FALSE
	EQ22	Output variable	BOOL	CH2 coincidence signal No.2 (TRUE: Coincident FALSE: Not coincident)	TRUE, FALSE

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	RDY	Public variable	BOOL	Module READY (TRUE: READY completed FALSE: READY uncompleted). Store the status of module READY (Xn0). Perform counter processing when module READY (Xn0) is TRUE.	TRUE, FALSE	FALSE	System
	FBRK	Public variable	BOOL	Fuse break detected flag (TRUE: Fuse break FALSE: normal). Store TRUE when fuse break occurs.	TRUE, FALSE	FALSE	System
	CINH1 to CINH2	Public variable	BOOL	CH1 to CH2 counter enable command (TRUE: Counter disabled FALSE: Counter enabled). Set counter enabled/disabled in channel.	TRUE, FALSE	FALSE	User
	EXENB1 to EXENB2	Public variable	BOOL	CH1 to CH2 coincidence signal enable command. TRUE: External output when counter value is coincident (EQ11, EQ12/EQ21, EQ22 TRUE) FALSE: Not external output when counter value is coincident (EQ11, EQ12/EQ21, EQ22 FALSE) Set in channel whether output counter coincidence signal when counter value is coincident.	TRUE, FALSE	FALSE	User
	DECRQ1 to DECQR2	Public variable	BOOL	CH1 to CH2 down-counter instruction. (TRUE: Execute down count FALSE: Stop down count) Set whether executing down-counter in 1-phase pulse input mode in channels.	TRUE, FALSE	FALSE	User
	EXPRER1 to EXPRER2	Public variable	BOOL	CH1 to CH2 external pre-set detection reset command. Reset EXPRE1 to 2 when FALSE→TRUE.	TRUE, FALSE	FALSE	User
	EXPRE1 to EXPRE2	Public variable	BOOL	CH1 to CH2 external pre-set request detection. ON and LATCH via pre-set command signal from pre-set input variable. Reset EXPRER1 to EXPRER2 when FALSE→TRUE.	TRUE, FALSE	FALSE	System
	FSEL1 to FSEL2	Public variable	BOOL	CH1 to CH2 counter function selection start command. Execute counter function selection when FALSE→TRUE.	TRUE, FALSE	FALSE	User
	GT11	Public variable	BOOL	CH1 counter value is bigger (point No.1) When current value>coincident output point No.1 setting, TRUE is stored.	TRUE, FALSE	FALSE	System
	GT12	Public variable	BOOL	CH1 counter value is bigger (point No.2) When current value>coincident output point No.2 setting, TRUE is stored.	TRUE, FALSE	FALSE	System
	GT21	Public variable	BOOL	CH2 counter value is bigger (point No.1) When current value>coincident output point No.1 setting, TRUE is stored.	TRUE, FALSE	FALSE	System
	GT22	Public variable	BOOL	CH2 counter value is bigger (point No.2) When current value>coincident output point No.2 setting, TRUE is stored.	TRUE, FALSE	FALSE	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	LT11	Public variable	BOOL	CH1 counter value is smaller (point No.1) When current value<coincident output point No.1 setting, TRUE is stored.	TRUE, FALSE	FALSE	System
	LT12	Public variable	BOOL	CH1 counter value is smaller (point No.2) When current value<coincident output point No.2setting, TRUE is stored.	TRUE, FALSE	FALSE	System
	LT21	Public variable	BOOL	CH2 counter value is smaller (point No.1) When current value<coincident output point No.1 setting, TRUE is stored.	TRUE, FALSE	FALSE	System
	LT22	Public variable	BOOL	CH2 counter value is smaller (point No.2) When current value<coincident output point No.2 setting, TRUE is stored.	TRUE, FALSE	FALSE	System
	PRE1 to PRE2	Public variable	DINT	CH1 to CH2 preset value It indicates the preset value set in counter.	-2147483648 to 2147483647	0	System
	OVFL1 to OVFL2	Public variable	INT	CH1 to CH2 overflow detection (1: Overflow occurs 0: Not overflow) Store the occurrence status of counter overflow when the counter type is linear counter.	0,1	0	System
	CFLG1	Public variable	INT	CH1 sampling/periodic counter flag (1: Execution 2: Stop) Store the operation status of sampling or period counter tag of CH1.	0,1	0	System
	LATCH1	Public variable	DINT	CH1 latch count value Store latch counter value.	-2147483648 to 2147483647	0	System
	SAMP1	Public variable	DINT	CH1 sampling count value Store sampling count value.	-2147483648 to 2147483647	0	System
	PRVCYC LP1	Public variable	DINT	CH1 previous value of periodic pulse count Store the previous value of period pulse count.	-2147483648 to 2147483647	0	System
	NEWCYC LP1	Public variable	DINT	CH1 current value of periodic pulse count Store the current value of periodic pulse count.	-2147483648 to 2147483647	0	System
	CFLG2	Public variable	INT	CH2 sampling/periodic counter flag (1: Execution 0: Stop) Store the operation status of sampling or periodic counter flag of CH2.	0,1	0	System
	LTACH2	Public variable	DINT	CH2 latch count value Store the latch count value.	-2147483648 to 2147483647	0	System
	SAMP2	Public variable	DINT	CH2 sampling count value Store the sampling count value.	-2147483648 to 2147483647	0	System
	PRVCYC LP2	Public variable	DINT	CH2 previous value of periodic pulse count Store the previous value of periodic pulse count.	-2147483648 to 2147483647	0	System
	NEWCYC LP2	Public variable	DINT	CH2 current value of periodic pulse count Store the current value of periodic pulse count.	-2147483648 to 2147483647	0	System

*1 The public variables CINH1 to CINH2, EXENB1 to EXENB2, or FSEL1 to FSEL2 can be set on the FB property window of PX Developer.
Execute reading/writing the public variables other than listed above by program.
It will not be displayed on the FB property window of PX Developer.

Function

Item	Contents		
	Condition		Output (CH1 to CH2)
	Output condition signal (REFR)	Module READY (Xn0)	
Output condition signal (REFR)	TRUE	TRUE	Read pulse count value and coincidence signal from high-speed counter module and output CH1 value to output variable PV1 and output CH2 value to PV2.
		FALSE	Output variable PV1, PV2 holds the previous value.
	FALSE	TRUE or FALSE	
Others	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: ● High-Speed Counter Module User's Manual.		

POINT

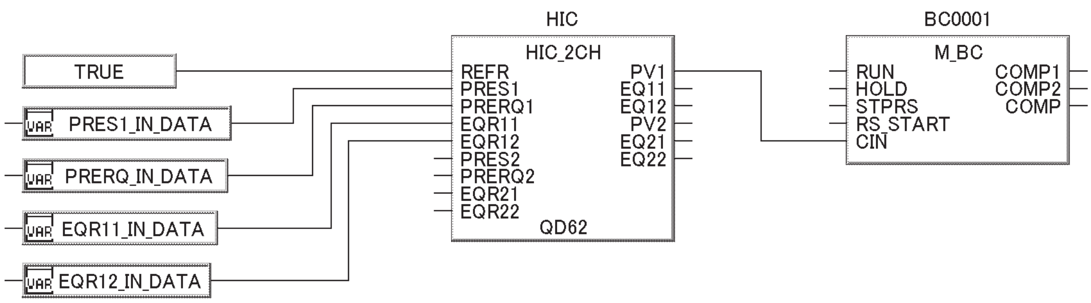
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-CT is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

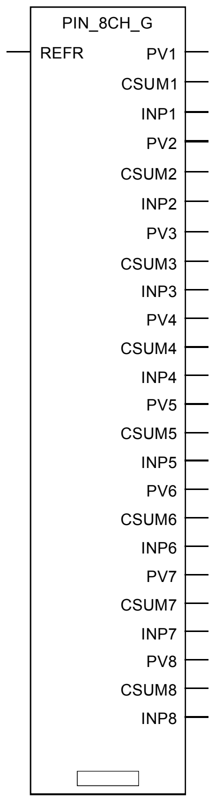
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- Unable to communicate with high-speed counter module. (Error code: 1412)
- The errors of high-speed count module have been detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



10.3.2 Channel-isolated 8 Channels Pulse Input (PIN_8CH_G)

FB	FBD part
PIN_8CH_G	

Indicates the type name of selected module.

Corresponding module
QD60P8-G

Function overview: Read the sampling pulse number of isolated pulse input module between 8 CHs channels for inputting pulse signal, accumulating count value and input pulse value, then output them from output variable.

Function/FB classification name: Module FB

Input and Output Pin

Pin	Variable name	Variable type	Data type	Content	Range
Input	REFR	Input variable	BOOL	Refreshing request (TRUE: Request FALSE: No request)	TRUE, FALSE
Output	PV1 to PV8	Output variable	INT	CH1 to CH8 sampling pulse number	0 to 32767
	CSUM1 to CSUM8	Output variable	DINT	CH1 to CH8 accumulating count value	0 to 99999999
	INP1 to INP8	Output variable	DINT	CH1 to CH8 input pulse value	0 to 2147483647

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	ERRC1 to ERRC8	Public variable	BOOL	CH1 to CH8 error reset request (TRUE: Error reset request FALSE: –) When the occurrence of CH1 to CH8 error (ERR1 to ERR8) is TRUE, ERRC1 to ERRC8 are set as TRUE, to clear errors and change ERR1 to ERRC8 setting into FALSE	TRUE, FALSE	FALSE	User
	CH1INH to CH8INH	Public variable	BOOL	Disable CH1 to CH8 count (TRUE: Count disabled, FLASE: Count enabled) Set count enabled/disabled on channels.	TRUE, FALSE	FALSE	User
	GEC1toGEC8	Public variable	BOOL	CH1 to CH8 comparison signal reset request (TRUE: Comparison signal reset request FALSE: –). When CH1 to CH8 accumulating count comparison flag (GE1 to GE8) is TRUE, clear the accumulating count comparison flag and GE1 to GE8 will become FALSE.	TRUE, FALSE	FALSE	User
	ALENBWR1 to ALENBWR8	Public variable	INT	CH1 to CH4 alarm output selection (1: alarm output enabled 0: alarm output disabled) Set alarm output enabled/disabled in each channel. STB is valid when it transforms from FLASE to TRUE.	0,1	0	User
	CRYOVRST1 to CRYOVRST8	Public variable	INT	CH1 to CH8 carry-over reset request (1: Carry-over reset request 0: –) When CH1 to CH8 carry-over occurrences (CRYOV1 to CRYOV8) are [1], clear carry-over by setting CRYOVRST1 to CRYOVRST8 as 1 and CRYOV1 to CRYOV8 will become 0 accordingly.	0,1	0	User
	CNTRST1 to CNTRST8	Public variable	INT	CH1 to CH8 count reset request (1: Count reset request 0: –) Clear the sampling pulse number accumulating count value and input pulse value of each channel by set CNTRST1 to CNTRST8 as 1.	0,1	0	User
	PRESCLSELWR1 to PRESCLSELWR8	Public variable	INT	CH1 to CH8 pre-scale function selection <setting value> 0: none 1: × 1 2: × 0.1 3: × 0.01 4: × 0.001 5: × 0.0001 Set 0 to 5 for each channel STB is valid when it transforms from FALSE to TRUE.	0 to 5	0	User
	PRESCLSVWR1 to PRESCLSVWR8	Public variable	INT	CH1 to CH8 pre-scale setting Set the pre-scale value. (When it is set as 0, sampling pulse number (PV1 to PV8) hold 0. Please pay attention to this phenomenon.) STB is valid when it transforms from FALSE to TRUE.	0 to 32767	0	User

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	STB	Public variable	BOOL	Operation condition setting request. When STB transforms form FALSE to TRUE, the setting of alarm output selection (ALENBWR1 to ALENBWR8), pre-scale function selection (PRESCLSELWR1 to PRESCLSELWR8) and pre-scale setting value (PRESCLSVWR1 to PRESCLSVWR8) is valid.	TRUE, FALSE	FALSE	User
	RDY	Public variable	BOOL	Module READY (TRUE: Ready FALSE: Not ready) Store the status of module READY (Xn0) When the module READY (Xn0) is TRUE, execute pulse input processing.	TRUE, FALSE	FALSE	System
	ERR1 to ERR8	Public variable	BOOL	CH1 to CH8 error (TRUE: Error FALSE: no error) Store TRUE when overflow or initial value setting error occurs.	TRUE, FALSE	FALSE	System
	GE1 to GE8	Public variable	BOOL	CH1 to CH8 accumulating counter comparison flag. Store TRUE when the accumulating count value ≥ comparison output value if comparison output is selected as [with comparison output function].	TRUE, FALSE	FALSE	System
	ALENBRD1 to ALENBRD8	Public variable	INT	Alarm output selection status (1: alarm output enabled 0: alarm output disabled) Store the alarm output selection status.	0,1	0	System
	AL1 to AL8	Public variable	INT	CH1 to CH8 alarm output flag (1: Range over 0: normal). Store 1 when alarm input is selected as [with alarm output function] and sampling pulse number exceeds the high high limit or low low limit of alarm output setting value.	0,1	0	System
	CRYOV1 to CRYOV8	Public variable	INT	CH1 to CH8 carry-over detection flag (1: Execute detection 0: not execute detection). Store 1 when the accumulating counter ring counter and value within the accumulating counter exceeds 99999999. Even if carry-over detection flag is 1, count operation continues.	0, 1	0	System
	OVFL1 to OVFL8	Public variable	INT	CH1 to CH8 carry-over detection flag (1: Execute detection 0: not execute detection). Store 1 when the accumulating counter linear counter and the value of accumulating counter exceeds 99999999.	0, 1	0	System
	ERRCOD1 to ERRCOD8	Public variable	INT	CH1 to CH8 error code Store the code of error that is detected by pulse input module. For detailed information about the error code, refer to Channel Isolated Pulse Input Module User's Manual.	–	0	System

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Conversion processing	PRESCLSELRD1 to PRESCLSELRD8	Public variable	INT	CH1 to CH8 pre-scale function selection status <setting value> 0: none 1: × 1 2: × 0.1 3: × 0.01 4: × 0.001 5: × 0.0001 Store the status of pre-scale function selection	0 to 5	0	System
	PRESCLSVRD1 to PRESCLVRD8	Public variable	INT	CH1 to CH8 pre-scale setting status Store the pre-scale setting status. (When it is set to 0, sampling pulse number (PV1 to PV8) hold 0. Please pay attention to this phenomenon.)	0 to 32767	0	System

*1 The public variables CHINH to CH8INH, PRESCLSELWR1 to PRESCLSELWR2, or ALENBRD1 to ALENBRD2 can be set on the FB property window of PX Developer. Execute reading/writing the public variables other than listed above by program. It will not be displayed on the FB property window of PX Developer.

Function

Item	Contents		
	Condition: Input condition signal (REFR)	Condition: Module READY (Xn0)	Output (CH1 to CH8)
Output condition signal (REFR)	TRUE	TRUE	Read sampling pulse number, accumulating count value and input pulse value from pulse input module, then output sampling pulse number for each channel from output variable PV1 to PV8, accumulating count value for each channel from output variable CSUM1 to CSUM8 and input pulse value for each channel from output variable INP1 to INP8.
	TRUE	FALSE	The previous value is kept for output variable PV1 to PV8, CSUM1 to CSUM8 and INP1 to INP8.
	FALSE	TRUE, FALSE	The previous value is kept for output variable PV1 to PV8, CSUM1 to CSUM8 and INP1 to INP8.
Others	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: Channel Isolated Pulse Input Module User's Manual.		

POINT

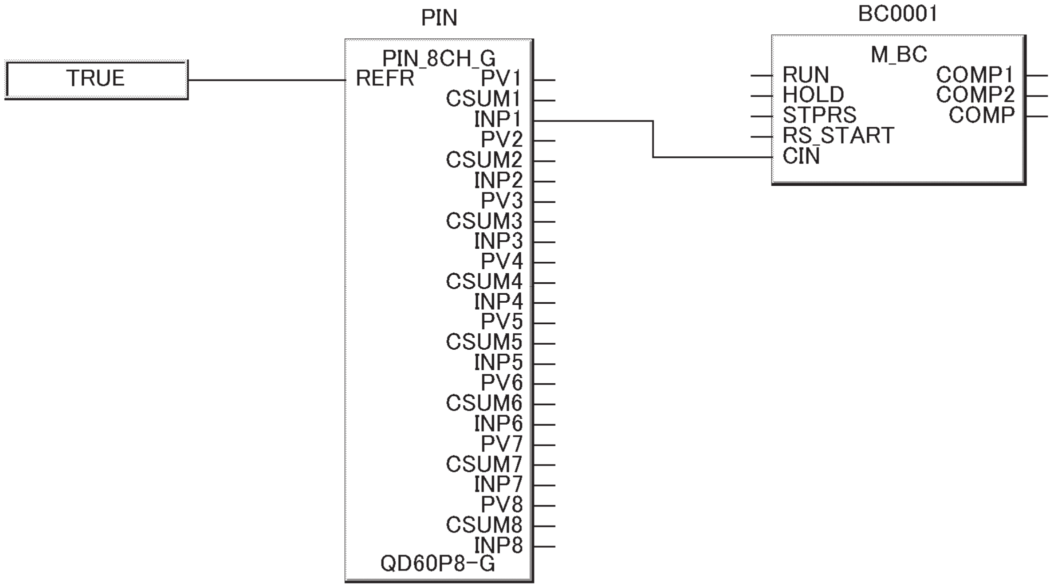
- For the settings of initial settings, etc., using the intelligent function module operation of GX Works2 or GX Configurator-CT is recommended.
  For details, refer to Section 2.11.1.
- Module FB is incompliant with the intelligent function module mounted to a MELSECNET/H remote I/O station.
  For details, refer to Section 2.11.2.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

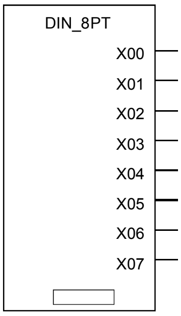
- Unable to communicate with pulse input module. (Error code: 1412)
- Abnormality of pulse input module has been detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



10.4 Digital I/O Module FB

10.4.1 8 Points Digital Input (DIN_8PT)

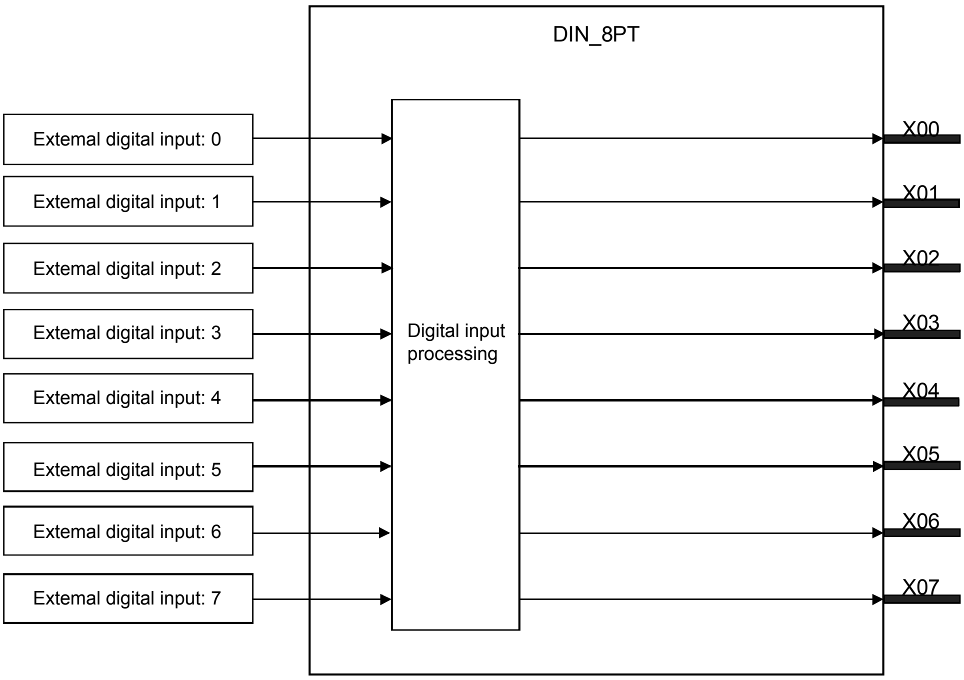
FB	FBD part
DIN_8CH_G	 Indicates the type name of the selected module.

Corresponding module
QX28

Function overview: Read the ON/OFF input value of 8 points digital input module and output it from output variable.

Function/FB classification name: Module FB

Block Diagram



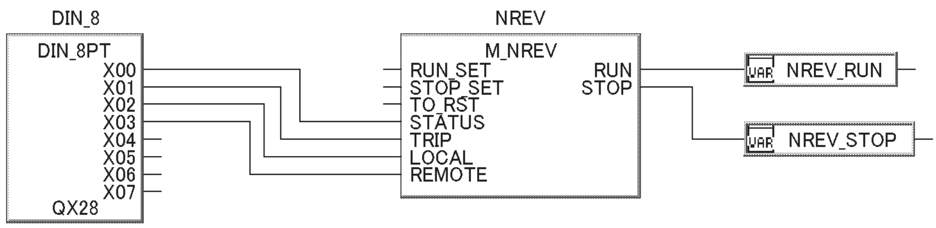
Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Output	X00 to X07	Output variable	BOOL	Output signal (TRUE: ON FALSE: OFF)	TRUE, FALSE

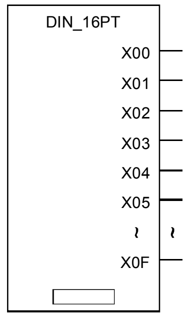
Function

Item	Contents
Digital input processing	For information about the processing and setting of corresponding module of this module FB, refer to the following manual. ● Building Block I/O Module User's Manual.

Program Example



10.4.2 16 Points Digital Input (DIN_16PT)

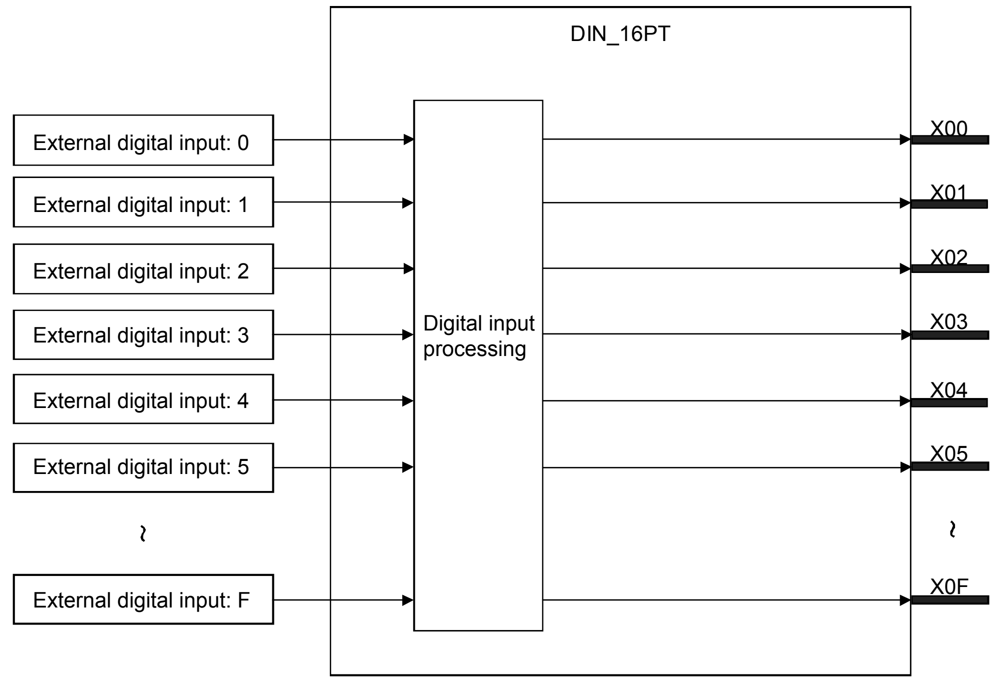
FB	FBD part
DIN_16PT	 Indicates the type name of the selected module.

Corresponding module
QX10, QX40, QX40-S1, QX50, QX70, QX80

Function overview: Read the ON/OFF input value of 16 points digital input module and output it from output variable.

Function/FB classification name: Module FB

Block Diagram



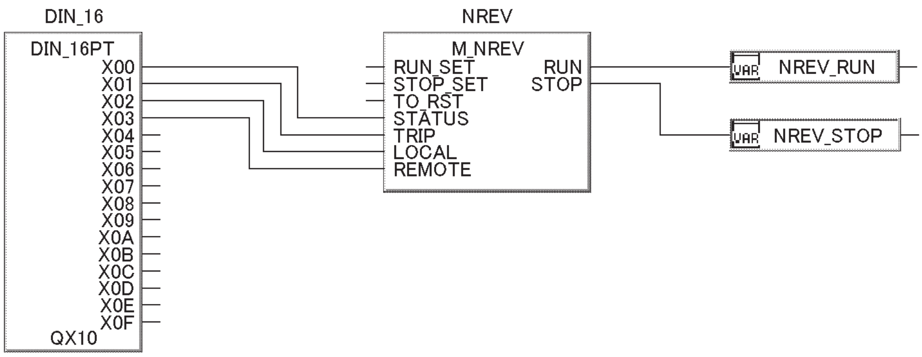
Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Output	X00 to X0F	Output variable	BOOL	Output signal (TRUE: ON FALSE: OFF)	TRUE, FALSE

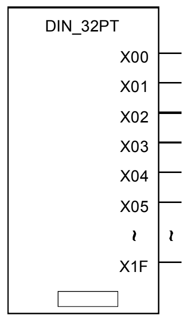
Function

Item	Contents
Digital input processing	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: ● Building Block I/O Module User's Manual.

Program Example



10.4.3 32 Points Digital Input (DIN_32PT)

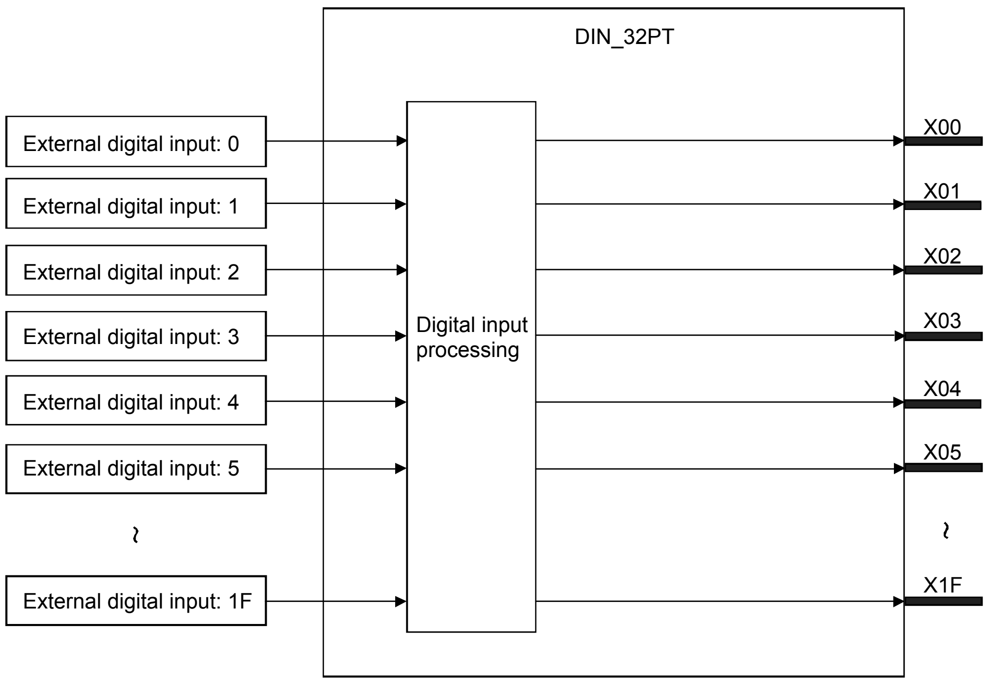
FB	FBD part
DIN_32PT	 ☐ Indicates the type name selected module.

Corresponding module
QX41, QX41-S1, QX71, QX81

Function overview: Read the ON/OFF input value of 32 points digital input module and output it from output variable.

Function/FB classification name: Module FB

Block Diagram



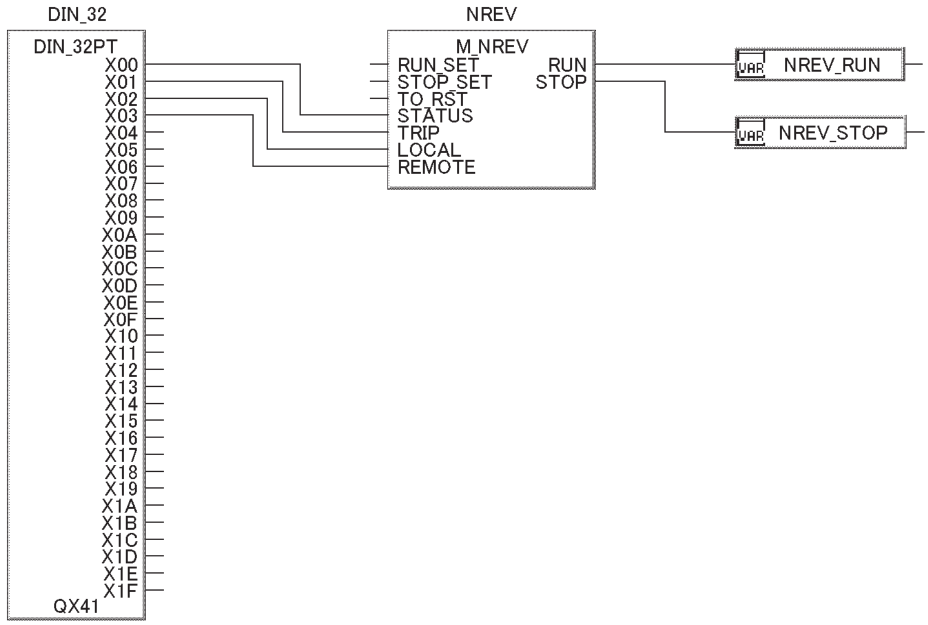
Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Output	X00 to X1F	Output variable	BOOL	Output signal (TRUE: ON FALSE: OFF)	TRUE, FALSE

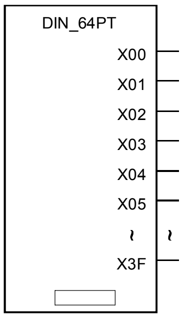
Function

Item	Contents
Digital input processing	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: ● Building Block I/O Module User's Manual.

Program Example



10.4.4 64 Points Digital Input (DIN_64PT)

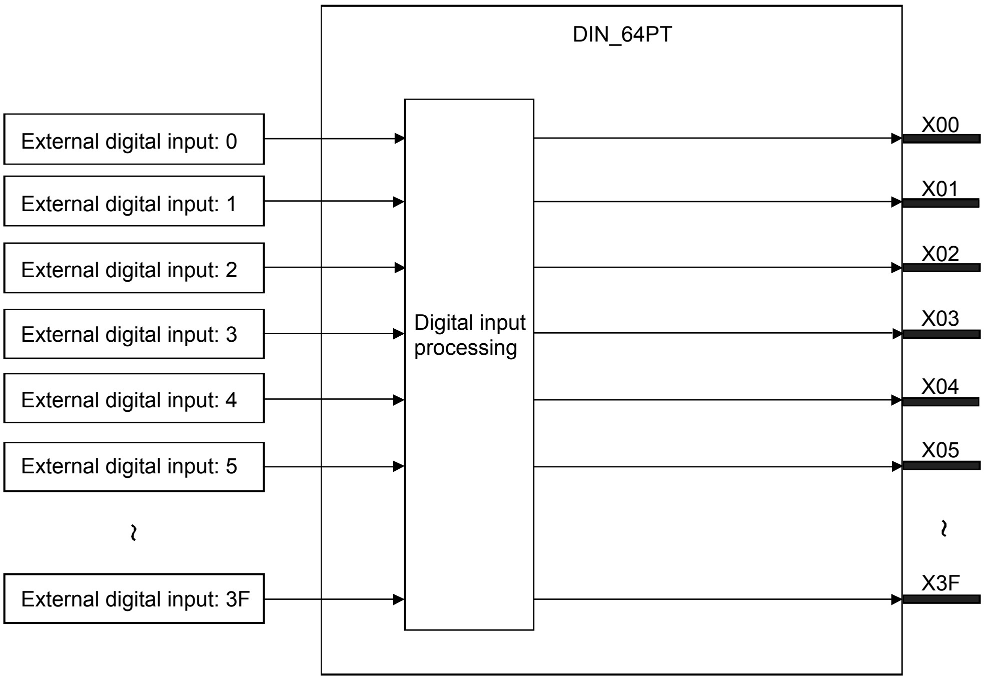
FB	FBD part
DIN_64PT	 Indicates the type name selected module.

Corresponding module
QX42, QX42-S1, QX72, QX82, QX82-S1

Function overview: Read the ON/OFF input value of 64 points digital input module and output it from output variable.

Function/FB classification name: Module FB

Block Diagram



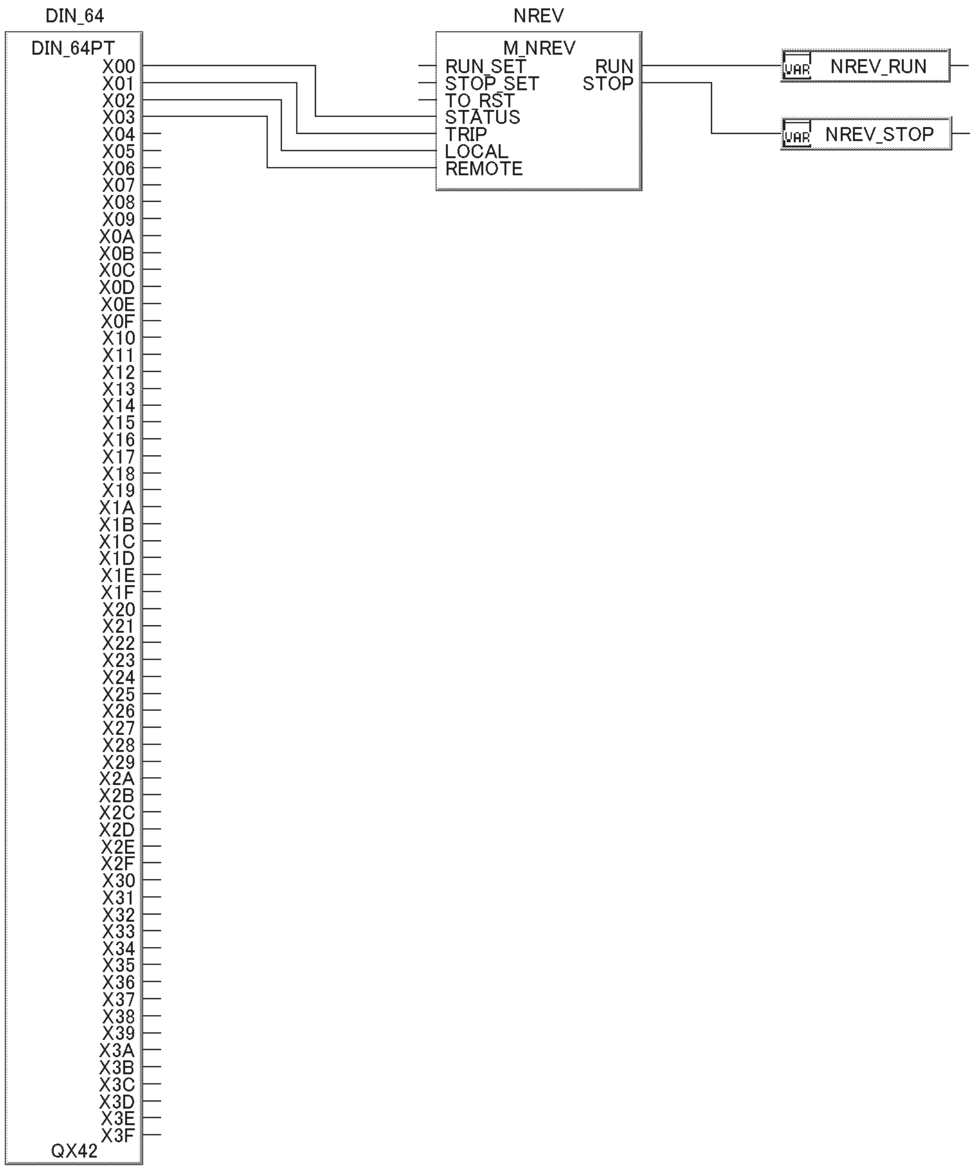
Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Output	X00 to X3F	Output variable	BOOL	Output signal (TRUE: ON FALSE: OFF)	TRUE, FALSE

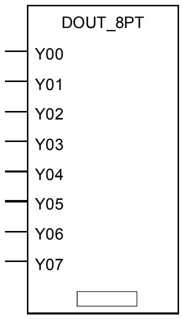
Function

Item	Contents
Digital input processing	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: ● Building Block I/O Module User's Manual.

Program Example



10.4.5 8 Points Digital Output (DOUT_8PT)

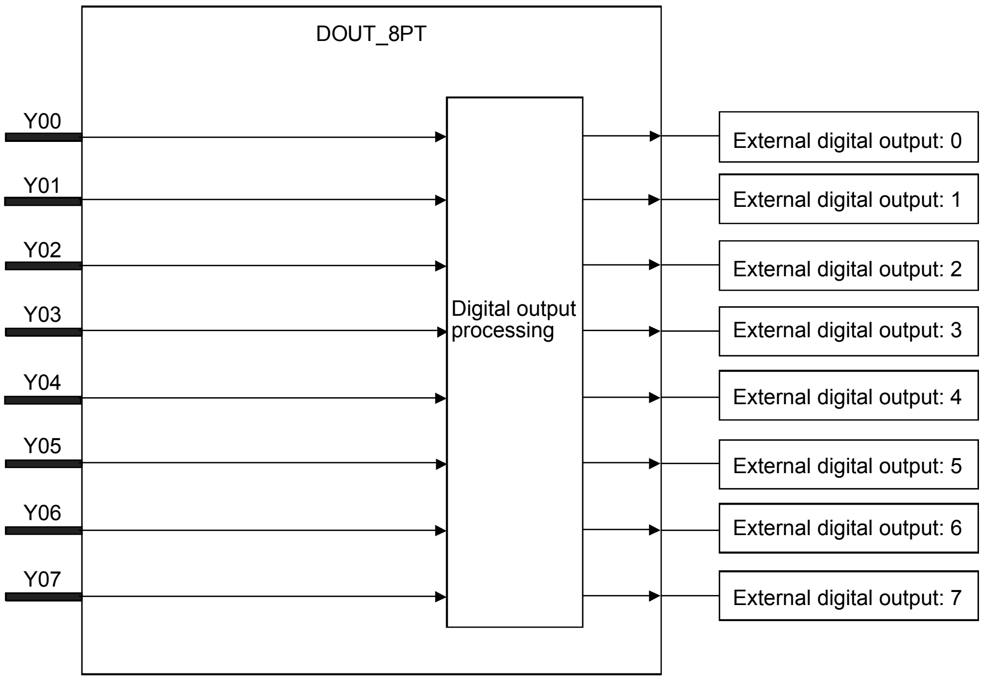
FB	FBD part
DOUT_8PT	 Indicates the type name selected module.

Corresponding module
QY18A, QY68A

Function overview: Write ON/OFF output value of input variable to 8 points digital output module.

Function/FB classification name: Module FB

Block Diagram



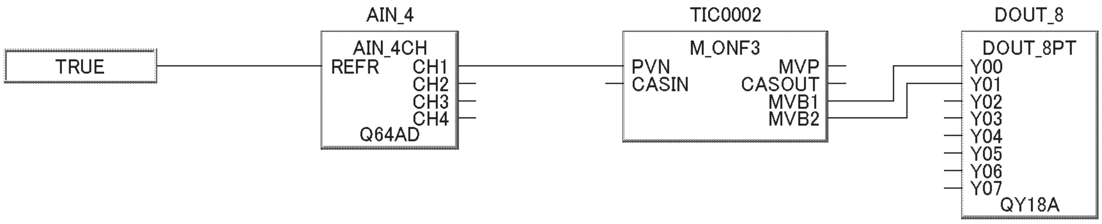
Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	Y00 to Y07	Input variable	BOOL	Input signal (TRUE: ON FALSE: OFF)	TRUE, FALSE

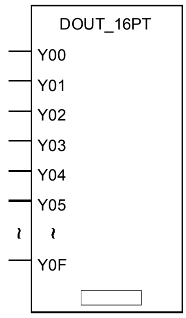
Function

Item	Contents
Digital output processing	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: ● Building Block I/O Module User's Manual.

Program Example



10.4.6 16 Points Digital Output (DOUT_16PT)

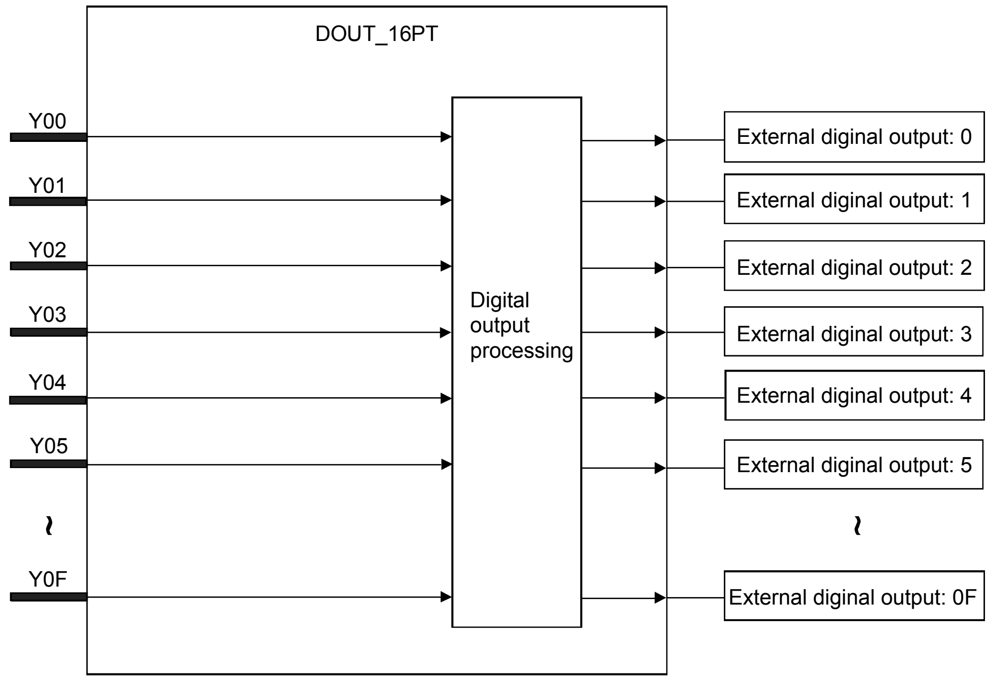
FB	FBD part
DOUT_16PT	 Indicates the type name of the selected module.

Corresponding module
QY10, QY22, QY40P, QY50, QY70, QY80

Function overview: Write ON/OFF output value from input variable to 16 points digital output module.

Function/FB classification name: Module FB

Block Diagram



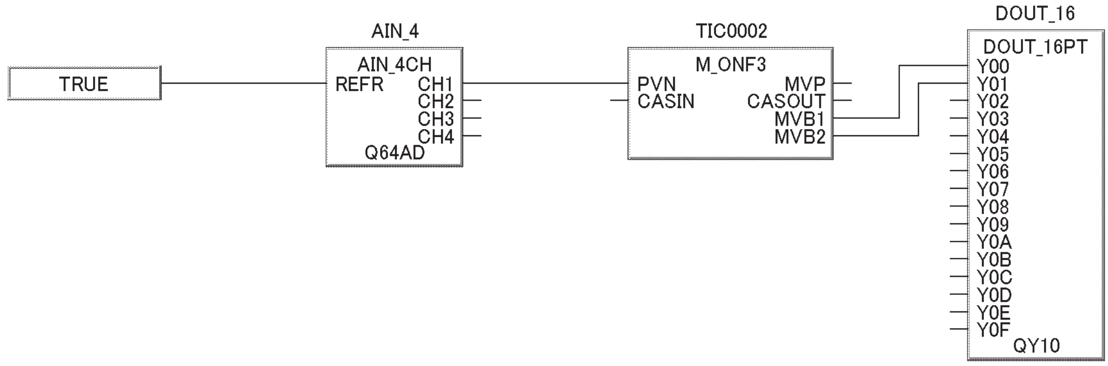
Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	Y00 to Y0F	Input variable	BOOL	Input signal (TRUE: ON FALSE: OFF)	TRUE, FALSE

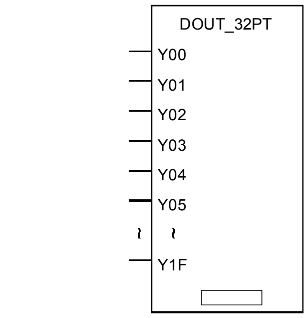
Function

Item	Contents
Digital output processing	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: ● Building Block I/O Module User's Manual.

Program Example



10.4.7 32 Points Digital Output (DOUT_32PT)

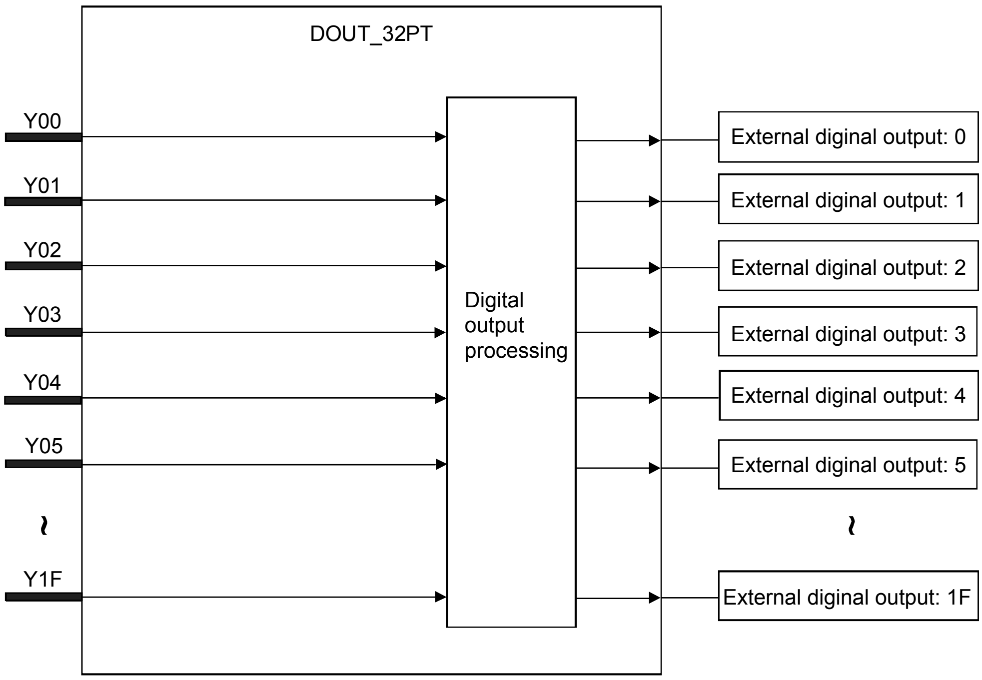
FB	FBD part
DOUT_32PT	 Indicates the type name of the selected module.

Corresponding module
QY41P, QY71, QY81P

Function overview: Write ON/OFF output value from input variable to 32 points digital output module.

Function/FB classification name: Module FB

Block Diagram



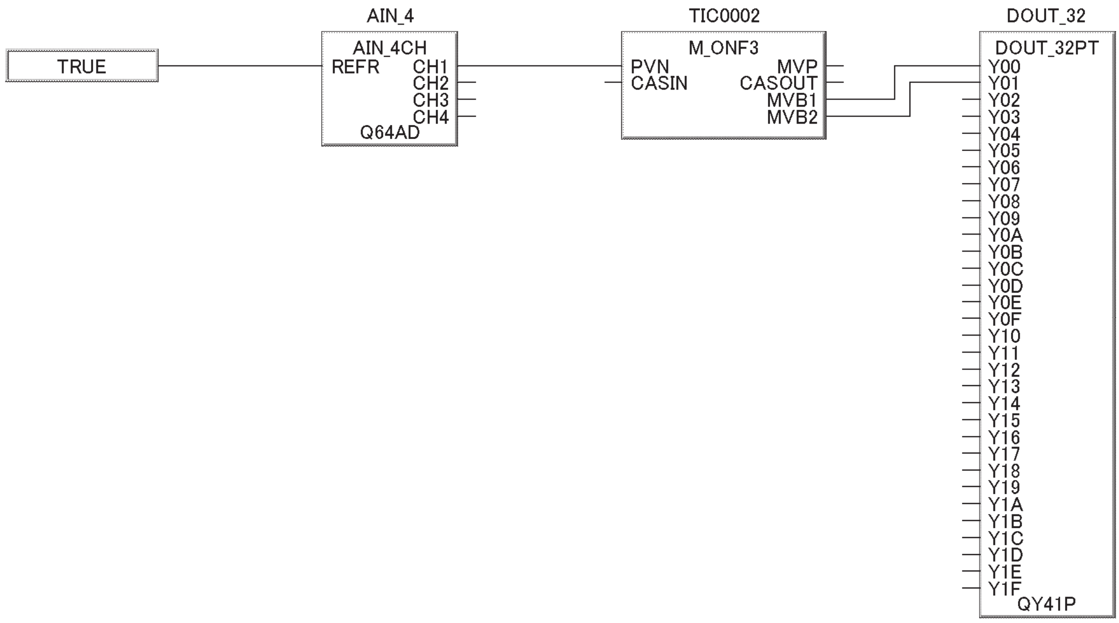
Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	Y00 to Y1F	Input variable	BOOL	Input signal (TRUE: ON FALSE: OFF)	TRUE, FALSE

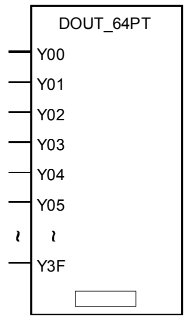
Function

Item	Contents
Digital output processing	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: ● Building Block I/O Module User's Manual.

Program Example



10.4.8 64 Points Digital Output (DOUT_64PT)

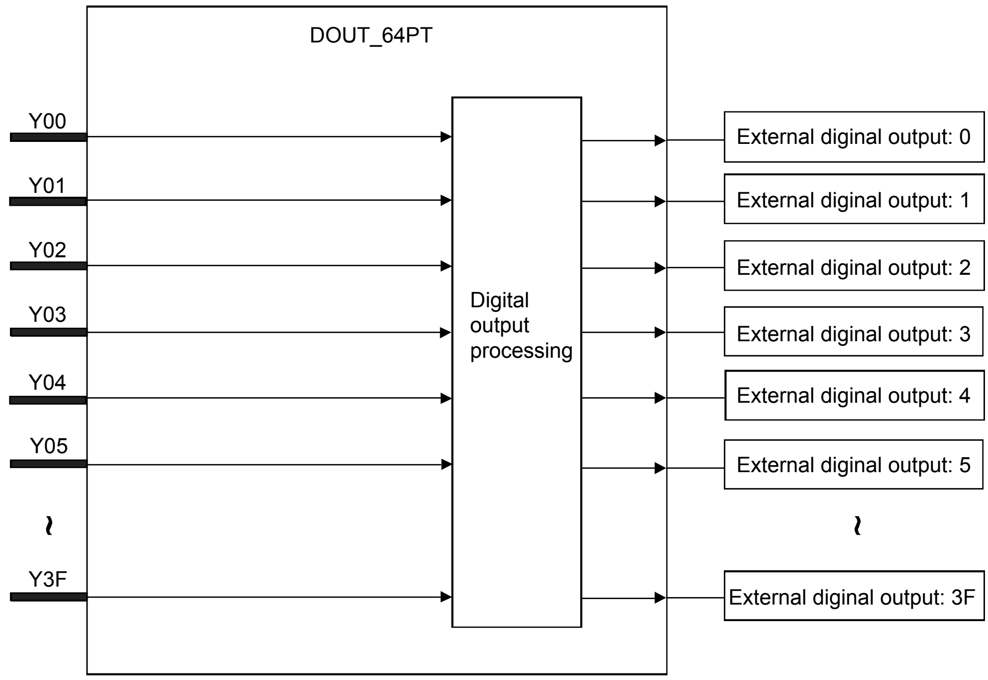
FB	FBD part
Pin_8CH_G	 ☐ Indicates the type name of the selected module.

Corresponding module
QY42P

Function overview: Write ON/OFF output value from input variable to 64 Points digital output module.

Function/FB classification name: Module FB

Block Diagram



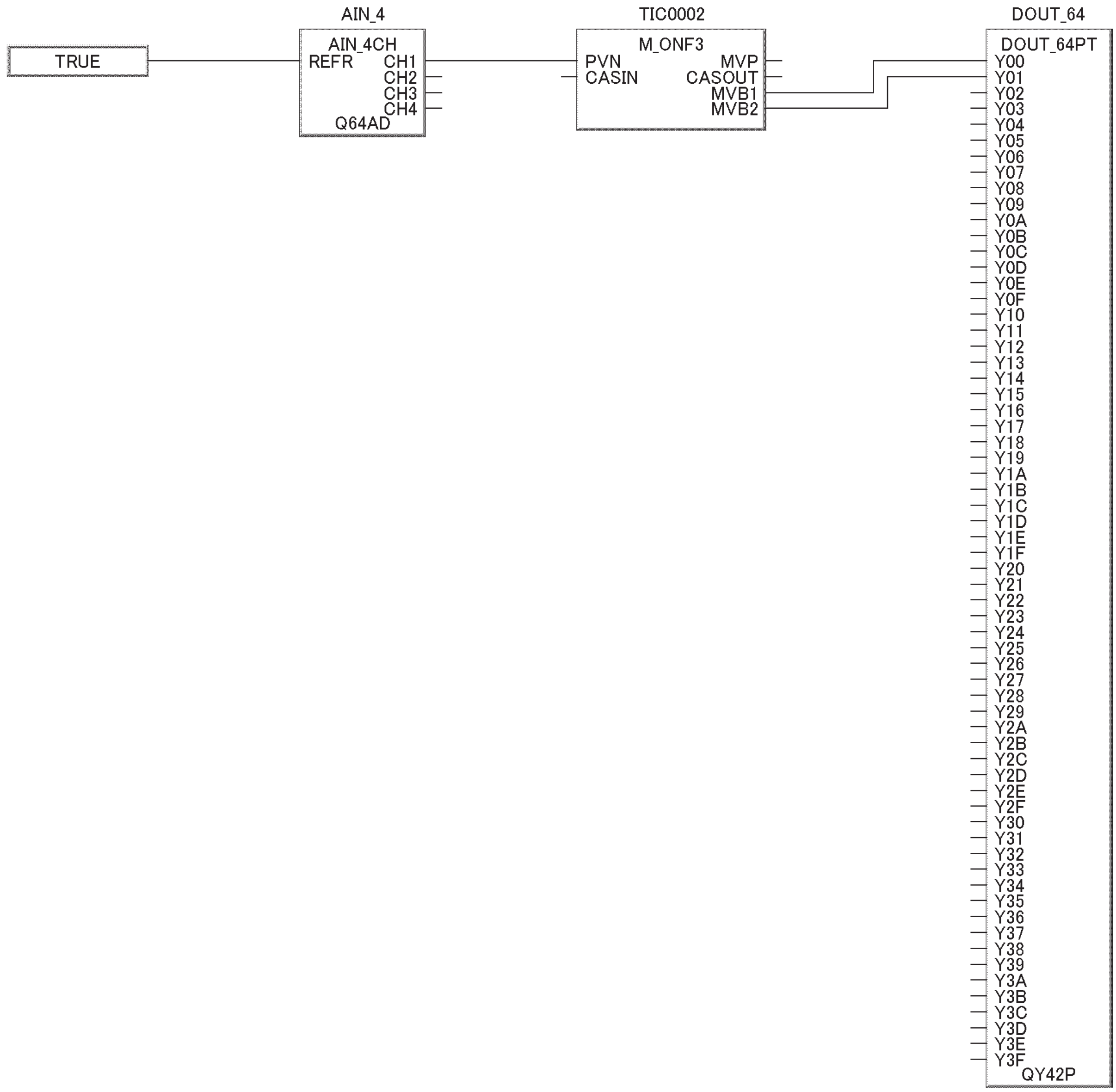
Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	Y00 to Y3F	Input variable	BOOL	Input signal (TRUE: ON FALSE: OFF)	TRUE, FALSE

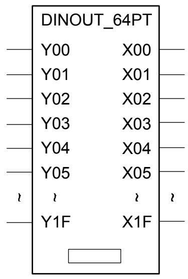
Function

Item	Contents
Digital output processing	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: ● Building Block I/O Module User's Manual.

Program Example



10.4.9 32 Points Input/32 Points Output I/O Mixed (DINOUT_64PT)

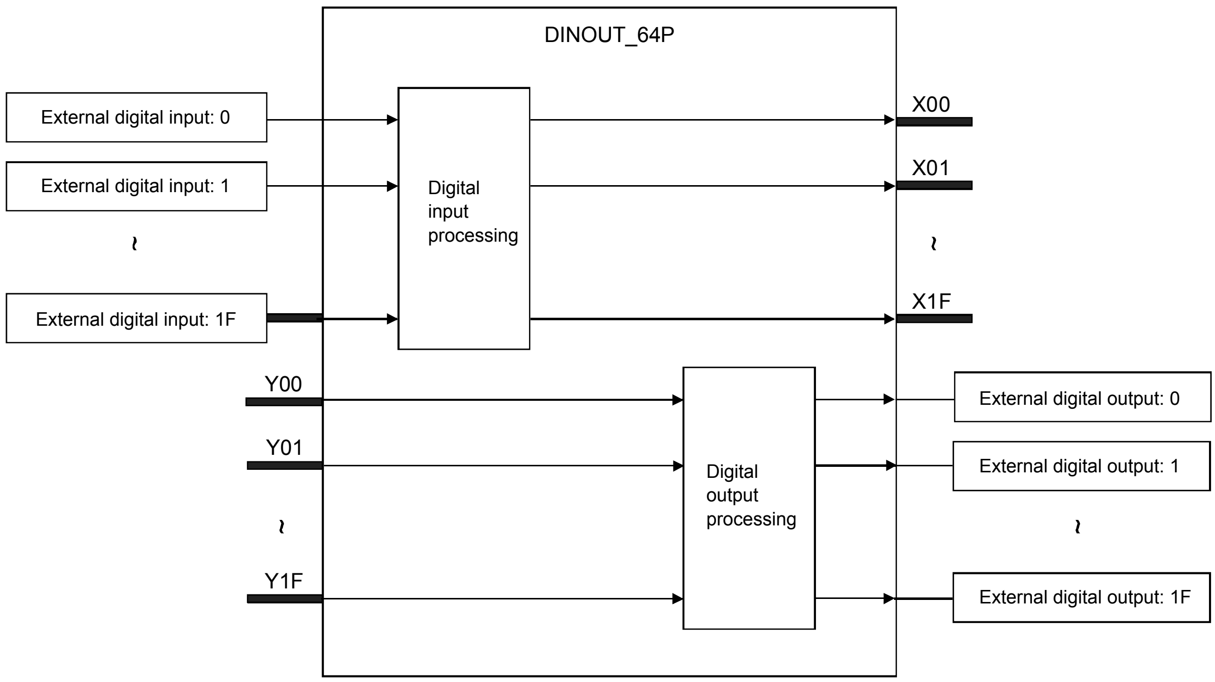
FB	FBD part
DINOUT_64PT	 Indicates the type name of the selected module

Corresponding module
QH42P

Function overview: Perform reading/writing ON/OFF input/output data on 64 points input/output mixed module (32 points digital input/32 points output).

Function/FB classification name: Module FB

Block Diagram



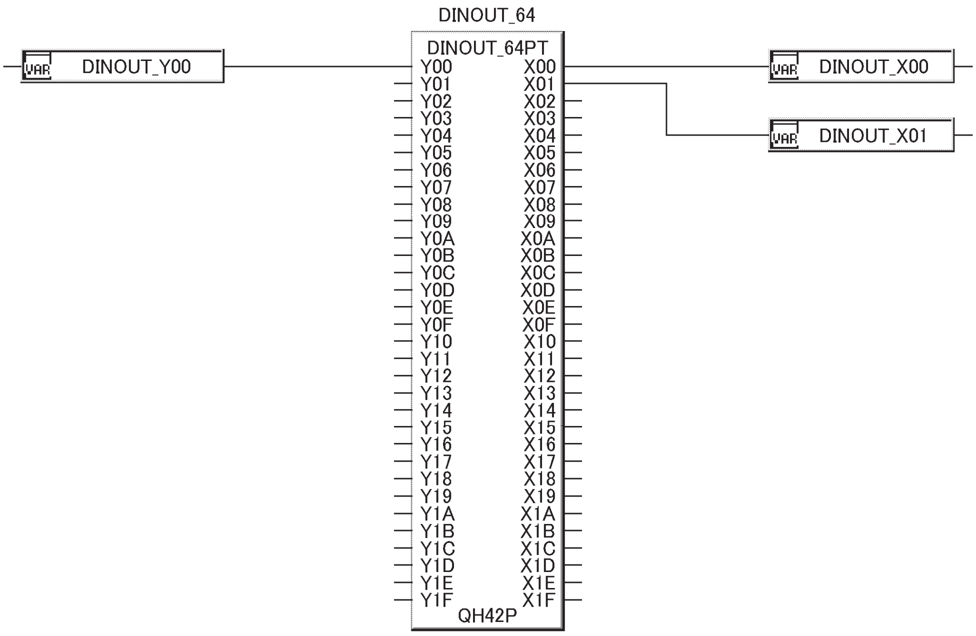
Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	Y00 to Y1F	Input variable	BOOL	Input signal (TRUE:ON FALSE:OFF)	TRUE, FALSE
Output	X00 to X1F	Output variable	BOOL	Output signal (TRUE:ON FALSE:OFF)	TRUE, FALSE

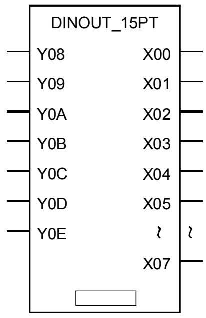
Function

Item	Contents
Digital input processing	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: ● Building Block I/O Module User's Manual.
Digital output processing	

Program Example



10.4.10 8 Points Input/7 Points Output I/O Mixed (DINOUT_15PT)

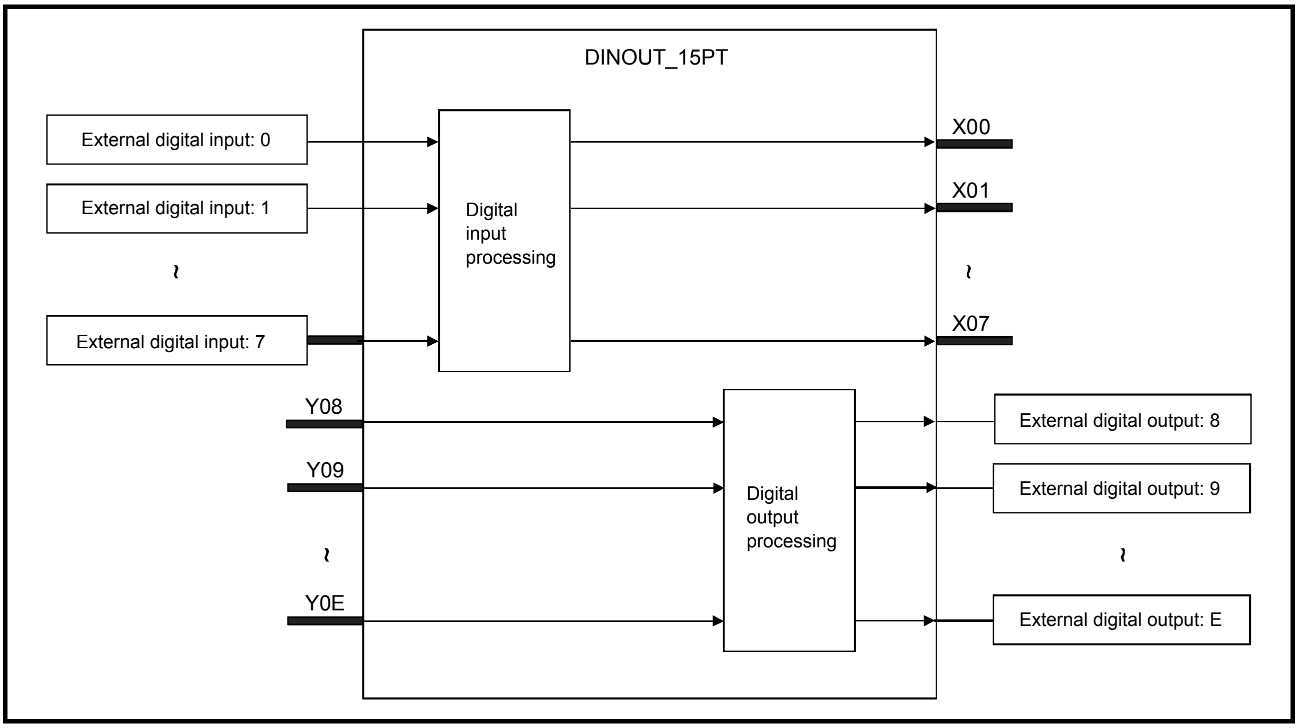
FB	FBD part
DINOUT_15PT	 Indicates the type name of the selected module.

Corresponding module
QX48Y57

Function overview: Perform reading/writing ON/OFF input/output data for 15 points input/output mixed module (8 points digital input/7 points digital output).

Function/FB classification name: Module FB

Block Diagram



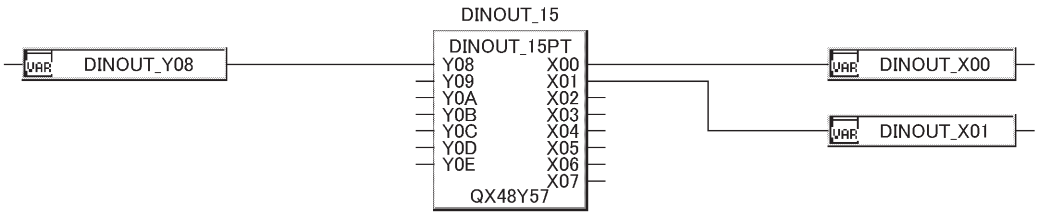
Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	Y08 to Y0E	Input variable	BOOL	Input signal (TRUE:ON FALSE:OFF)	TRUE, FALSE
Output	X00 to X07	Output variable	BOOL	Output signal (TRUE:ON FALSE:OFF)	TRUE, FALSE

Function

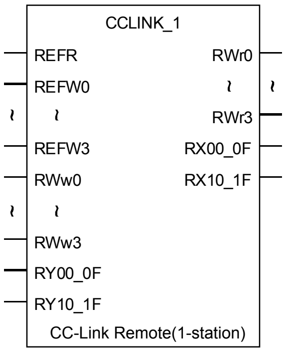
Item	Contents
Digital input processing	For information about the processing and setting of corresponding module of this module FB, refer to the following manual: ● Building Block I/O Module User's Manual.
Digital output processing	

Program Example



10.5 CC-Link Module FB

10.5.1 CC-Link Remote Station Occupying 1 Station (CCLINK_1)

FB	FBD part
CCLINK_1	

Corresponding module
CC-Link remote station occupying 1 station

Function overview: Read/write message of the remote station that occupies 1 station and is connected to CC-Link.

Function/FB classification name: Module FB

Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	RWr0 to RWr3 output condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW0 to REFW3	Input variable	BOOL	RWw0 to RWw3 input condition signal (TRUE: Enabled FALSE: Disabled)	TRUE, FALSE
	RWw0 to RWw3	Input variable	WORD	Input data of remote register RWw0 to RWw3.	0 to FFFF_H
	RY00_0F	Input variable	DWORD	Input data of remote output (RY00 to RY0F)	0 to $\mathrm{FFFFFFFF}_H$ (refer to (3) in POINT)
	RY10_1F	Input variable	DWORD	Input data of remote output (RY10 to RY1F)	0 to $\mathrm{FFFFFFFF}_H$ (refer to (3) in POINT)
Output	RWr0 to RWr3	Output variable	WORD	Output data from remote register RWr0 to RWr3	0 to FFFF_H
	RX00_0F	Output variable	WORD	Output data of remote input (RX00 to RX0F)	0 to FFFF_H (refer to (4) in POINT)
	RX10_1F	Output variable	WORD	Output data of remote input (RX10 to RX1F)	0 to FFFF_H (refer to (4) in POINT)

POINT

(1) Input of remote register or remote output pin that are not connected to input pin can be performed via ladder program.
However, please be sure to pre-set the RWw □ input condition signal used on ladder program into FALSE in making input to remote register.

(2) Remote output/input has nothing to do with REFR and REFW0 to REFW3, this module FB performs reading/writing to CC-Link master module during each execution.

(3) Please use "BIND" function in remote output.
(RY00_0F and RY10_1F are connected to OUT of BIND function)

(4) Please use "UNBIND" function in remote input.
(RX00_0F and RX10_1F are connected to IN of UNBIND function)

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	MASTERRDY	Public variable	BOOL	Module ready of master station (TRUE: ON FALSE: OFF) Store the module READY status of master station. Execute input/output when module READY of master station is TRUE.	TRUE, FALSE	FALSE	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input condition signal (REFW0 to REFW3)</td><td>
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Input (RWw0 to RWw3)</th></tr>
<tr><th>Input condition signal (REFW0 to REFW3)</th><th>Master module READY (MASTERDY)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Write the input value from input variable RWw0 to RWw3 into CC-Link module.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Not write the input value from input variable RWw0 to RWw3 into CC-Link module.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Output (RWr0 to RWr3)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Master module READY (MASTERDY)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Read the value stored in CC-Link module and output them from output variable RWr0 to RWr3.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Keep the previous value of output variable RWr0 to RWr3.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Others</td><td>For details about all the processing and settings of the corresponding module of this module FB, refer to the following manual:
● QJ61BT11 Control & Communication Link System Master/Local Module User's Manual.</td></tr>
</table>

POINT

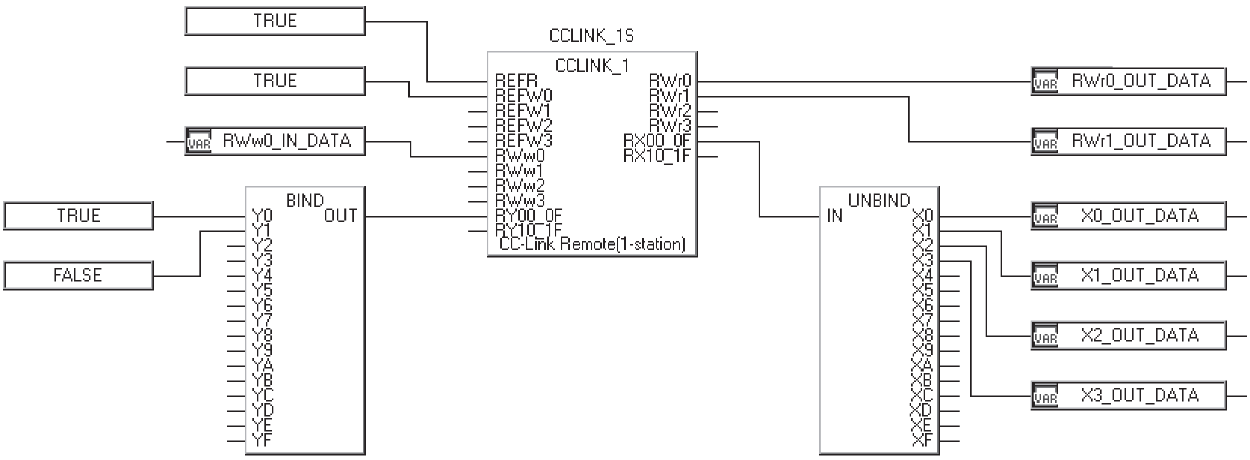
- Please do not set the auto refreshing parameter when using this module FB.
- Please set network parameter through GX application.

Error

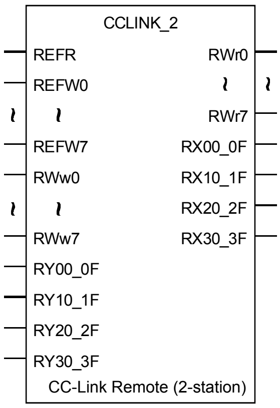
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- Unable to communicate with CC-Link master module. (Error code: 1412)
- Abnormality of CC-Link master module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



10.5.2 CC-Link Remote Station Occupying 2 Stations (CCLINK_2)

FB	FBD part
CCLINK_2	

Corresponding module
CC-Link remote station occupying 2 stations

Function overview: Perform reading/writing message of the remote station that occupies 2 stations and is connected to CC-Link.

Function/FB classification name: Module FB

Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	RWR0 to RWR7 output condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW0 to REFW7	Input variable	BOOL	RWw0 to RWw7 input condition signal (TRUE: Enabled FALSE: Disabled)	TRUE, FALSE
	RWw0 to RWw7	Input variable	WORD	Input data to remote register RWw0 to RWw7.	0 to FFFF$_H$
	RY00_0F	Input variable	DWORD	Input data of remote output (RY00 to RY0F)	0 to FFFFFFFF$_H$ (refer to (3) in POINT)
	RY10_1F	Input variable	DWORD	Input data of remote output (RY10 to RY1F)	0 to FFFFFFFF$_H$ (refer to (3) in POINT)
	RY20_2F	Input variable	DWORD	Input data of remote output (RY20 to RY2F)	0 to FFFFFFFF$_H$ (refer to (3) in POINT)
	RY30_3F	Input variable	DWORD	Input data of remote output (RY30 to RY3F)	0 to FFFFFFFF$_H$ (refer to (3) in POINT)
Output	RWr0 to RWr7	Output variable	WORD	Output data from remote register RWr0 to RWr7	0 to FFFF$_H$
	RX00_0F	Output variable	WORD	Output data of remote input (RX00 to RX0F)	0 to FFFF$_H$ (refer to (4) in POINT)
	RX10_1F	Output variable	WORD	Output data of remote input (RX10 to RX1F)	0 to FFFF$_H$ (refer to (4) in POINT)
	RX20_2F	Output variable	WORD	Output data of remote input (RX20 to RX2F)	0 to FFFF$_H$ (refer to (4) in POINT)
	RX30_3F	Output variable	WORD	Output data of remote input (RX30 to RX3F)	0 to FFFF$_H$ (refer to (4) in POINT)

POINT

(1) Input of remote register or remote output that are not connected to input pin can be performed via ladder program.
However, please be sure to pre-set the RWw □ to input condition signal used on ladder program into FALSE in making input to remote register.
(2) Remote output/input has nothing to do with REFR and REFW0 to REFW7, this module FB performs reading/writing to CC-Link master module during each execution.
(3) Please use "BIND" function in remote output.
(RY00_0F to RY30_3F are connected to OUT of BIND function)
(4) Please use "UNBIND" function in remote input.
(RX00_0F to RX30_3F are connected to IN of UNBIND function)

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	MASTERRDY	Public variable	BOOL	Module ready of master station (TRUE: ON FALSE: OFF) Store the module READY status of master station. Execute input/output when module READY of master station is TRUE.	TRUE, FALSE	FALSE	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input condition signal (REFW0 to REFW7)</td><td>
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Input (RWw0 to RWw7)</th></tr>
<tr><th>Input condition signal (REFW0 to REFW7)</th><th>Master module READY (MASTERDY)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Write the input value from input variable RWw0 to RWw7 into CC-Link module.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Not write the input value from input variable RWw0 to RWw7 into CC-Link module.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Output (RWr0 to RWr7)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Master module READY (MASTERDY)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Read the value stored on CC-Link module and output them from output variable RWr0 to RWr7.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Keep the previous value of output variable RWr0 to RWr7.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Others</td><td>For details about all the processing and settings of the corresponding module of this module FB, refer to the following manual:
● QJ61BT11 Control & Communication Link System Master/Local Module User's Manual.</td></tr>
</table>

POINT

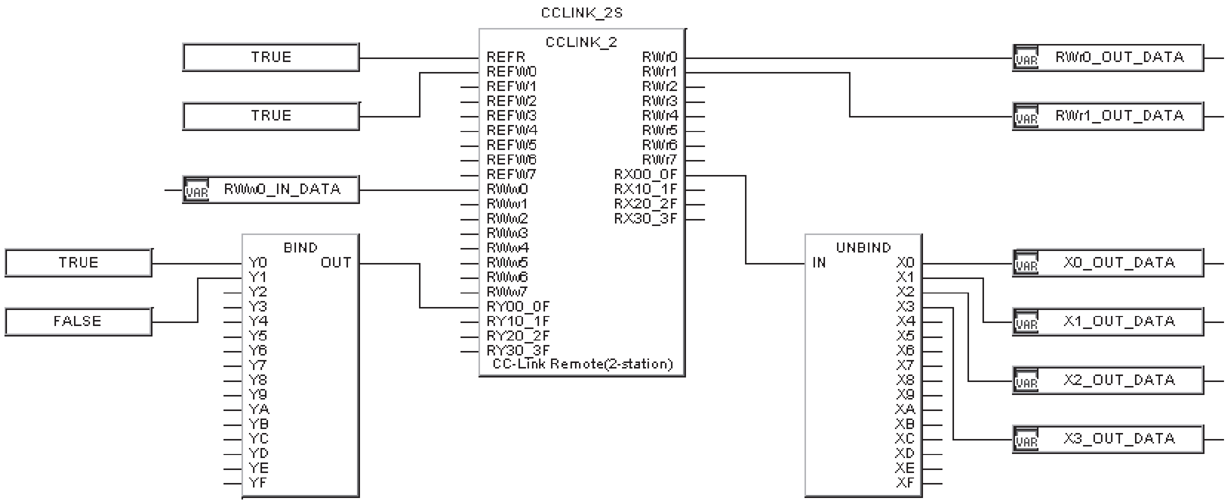
- Please do not set the auto refreshing parameter when using this module FB.
- Please set network parameter in GX application.

Error

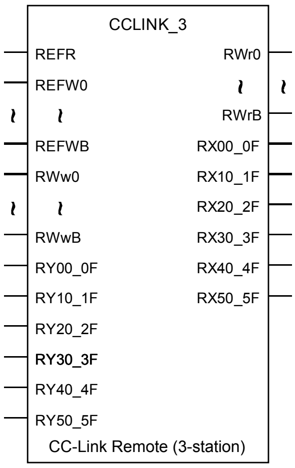
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- Unable to communicate with CC-Link master module. (Error code: 1412)
- Abnormality of CC-Link master module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



10.5.3 CC-Link Remote Station Occupying 3 Stations (CCLINK_3)

FB	FBD part
Pin_8CH_G	

Corresponding module
CC-Link remote station occupying 3 stations

Function overview: Reading/writing message of the remote station that occupies 3 stations and is connected to CC-Link.

Function/FB classification name: Module FB

Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	RWr0 to RWrB output condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW0 to REFWB	Input variable	BOOL	RWw0 to RWwB input condition signal (TRUE: Enabled FALSE: Disabled)	TRUE, FALSE
	RWw0 to RWwB	Input variable	WORD	Input data to remote register RWw0 to RWwB.	0 to FFFF$_{H}$
	RY00_0F	Input variable	DWORD	Input data of remote output (RY00 to RY0F)	0 to FFFFFFFF$_{H}$ (refer to (3) in POINT)
	RY10_1F	Input variable	DWORD	Input data of remote output (RY10 to RY1F)	0 to FFFFFFFF$_{H}$ (refer to (3) in POINT)
	RY20_2F	Input variable	DWORD	Input data of remote output (RY20 to RY2F)	0 to FFFFFFFF$_{H}$ (refer to (3) in POINT)
	RY30_3F	Input variable	DWORD	Input data of remote output (RY30 to RY3F)	0 to FFFFFFFF$_{H}$ (refer to (3) in POINT)
	RY40_4F	Input variable	DWORD	Input data of remote output (RY40 to RY4F)	0 to FFFFFFFF$_{H}$ (refer to (3) in POINT)
	RY50_5F	Input variable	DWORD	Input data of remote output (RY50 to RY5F)	0 to FFFFFFFF$_{H}$ (refer to (3) in POINT)
Output	RWr0 to RWrB	Output variable	WORD	Output data from remote register RWr0 to RWrB	0 to FFFF$_{H}$
	RX00_0F	Output variable	WORD	Output data of remote input (RX00 to RX0F)	0 to FFFF$_{H}$ (refer to (4) in POINT)
	RX10_1F	Output variable	WORD	Output data of remote input (RX10 to RX1F)	0 to FFFF$_{H}$ (refer to (4) in POINT)
	RX20_2F	Output variable	WORD	Output data of remote input (RX20 to RX2F)	0 to FFFF$_{H}$ (refer to (4) in POINT)
	RX30_3F	Output variable	WORD	Output data of remote input (RX30 to RX3F)	0 to FFFF$_{H}$ (refer to (4) in POINT)
	RX40_4F	Output variable	WORD	Output data of remote input (RX40 to RX4F)	0 to FFFF$_{H}$ (refer to (4) in POINT)
	RX50_5F	Output variable	WORD	Output data of remote input (RX50 to RX5F)	0 to FFFF$_{H}$ (refer to (4) in POINT)

POINT

(1) Input of remote register or remote output pin that are not connected to input pin can be performed via ladder program.
However, please be sure to pre-set the RWw □ input condition signal used on ladder program into FALSE in making input to remote register.

(2) Remote output/input has nothing to do with REFR and REFW0 to REFWB, this module FB performs reading/writing to CC-Link master module during each execution.

(3) Please use "BIND" function in remote output.
(RY00_0F and RY50_5F are connected to OUT of BIND function)

(4) Please use "UNBIND" function in remote input.
(RX00_0F and RX50_5F are connected to IN of UNBIND function)

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	MASTERRDY	Public variable	BOOL	Module ready signal of master station (TRUE: ON FALSE: OFF) Store the module READY status of master station. Execute input/output when module READY signal of master station is TRUE.	TRUE, FALSE	FALSE	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th>Contents</th></tr>
<tr><td>Input condition signal (REFW0 to REFWB)</td><td>
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Input (RWw0 to RWwB)</th></tr>
<tr><th>Input condition signal (REFW0 to REFWB)</th><th>Master module READY (MASTERRDY)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Write the input value from input variable RWw0 to RWwB into CC-Link module.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Not write the input value from input variable RWw0 to RWwB into CC-Link module.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Output condition signal (REFR)</td><td>
<table>
<tr><th colspan="2">Condition</th><th rowspan="2">Output (RWr0 to RWrB)</th></tr>
<tr><th>Output condition signal (REFR)</th><th>Master module READY (MASTERRDY)</th></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Read the value stored in CC-Link module and output them from output variable RWr0 to RWrB.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Keep the previous value of output variable RWr0 to RWrB.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
</table>
</td></tr>
<tr><td>Others</td><td>For detailed information of all the processing and settings of the corresponding module of this module FB, refer to the following manual:
● QJ61BT11 Control & Communication Link System Master/Local Module User's Manual.</td></tr>
</table>

POINT

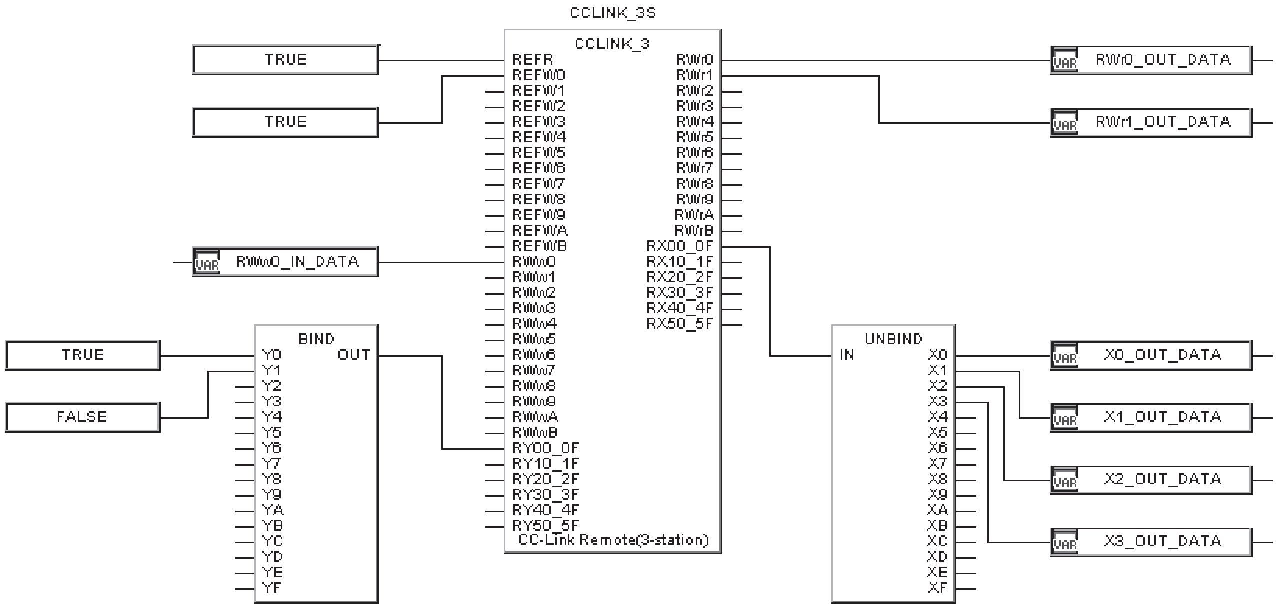
- Please do not set the auto refreshing parameter when using this module FB.
- Please set network parameter in GX application.

Error

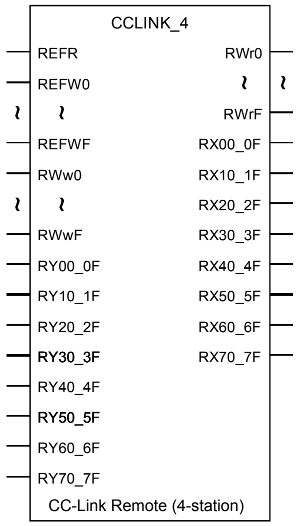
Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- Unable to communicate with CC-Link master module. (Error code: 1412)
- Abnormality of CC-Link master module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



10.5.4 CC-Link Remote Station Occupying 4 Stations (CCLINK_4)

FB	FBD part
CCLINK_4	

Corresponding module
CC-Link remote station occupying 4 stations

Function overview: Read/write message of the remote station that occupies 4 stations and is connected to CC-Link.

Function/FB classification name: Module FB

Input and Output Pin

Pin	Variable name	Variable type	Data type	Contents	Range
Input	REFR	Input variable	BOOL	RWr0 to RWrF output condition signal (TRUE: Execute FALSE: Stop)	TRUE, FALSE
	REFW0, REFWF	Input variable	BOOL	RWw0 to RWwF input condition signal (TRUE: Enabled FALSE: Disabled)	TRUE, FALSE
	RWw0, RWwF	Input variable	WORD	Input data to remote register RWw0 to RWwF.	0 to $FFFF_H$
	RY00_0F	Input variable	DWORD	Input data of remote output (RY00 to RY0F)	0 to $FFFFFFFF_H$ (refer to (3) in POINT)
	RY10_1F	Input variable	DWORD	Input data of remote output (RY10 to RY1F)	0 to $FFFFFFFF_H$ (refer to (3) in POINT)
	RY20_2F	Input variable	DWORD	Input data of remote output (RY20 to RY2F)	0 to $FFFFFFFF_H$ (refer to (3) in POINT)
	RY30_3F	Input variable	DWORD	Input data of remote output (RY30 to RY3F)	0 to $FFFFFFFF_H$ (refer to (3) in POINT)
	RY40_4F	Input variable	DWORD	Input data of remote output (RY40 to RY4F)	0 to $FFFFFFFF_H$ (refer to (3) in POINT)
	RY50_5F	Input variable	DWORD	Input data of remote output (RY50 to RY5F)	0 to $FFFFFFFF_H$ (refer to (3) in POINT)
	RY60_6F	Input variable	DWORD	Input data of remote output (RY60 to RY6F)	0 to $FFFFFFFF_H$ (refer to (3) in POINT)
	RY70_7F	Input variable	DWORD	Input data of remote output (RY70 to RY7F)	0 to $FFFFFFFF_H$ (refer to (3) in POINT)
Output	RWr0, RWrF	Output variable	WORD	Output data from remote register RWr0 to RWrF	0 to $FFFF_H$
	RX00_0F	Output variable	WORD	Output data of remote input (RX00 to RX0F)	0 to $FFFF_H$ (refer to (4) in POINT)
	RX10_1F	Output variable	WORD	Output data of remote input (RX10 to RX1F)	0 to $FFFF_H$ (refer to (4) in POINT)
	RX20_2F	Output variable	WORD	Output data of remote input (RX20 to RX2F)	0 to $FFFF_H$ (refer to (4) in POINT)
	RX30_3F	Output variable	WORD	Output data of remote input (RX30 to RX3F)	0 to $FFFF_H$ (refer to (4) in POINT)
	RX40_4F	Output variable	WORD	Output data of remote input (RX40 to RX4F)	0 to $FFFF_H$ (refer to (4) in POINT)
	RX50_5F	Output variable	WORD	Output data of remote input (RX50 to RX5F)	0 to $FFFF_H$ (refer to (4) in POINT)
	RX60_6F	Output variable	WORD	Output data of remote input (RX60 to RX6F)	0 to $FFFF_H$ (refer to (4) in POINT)
	RX70_7F	Output variable	WORD	Output data of remote input (RX70 to RX7F)	0 to $FFFF_H$ (refer to (4) in POINT)

POINT

(1) Input of remote register or remote output that are not connected to input pin can be performed via ladder program.
However, please be sure to pre-set the RWw □ input condition signal used on ladder program into FALSE in making input to remote register.
(2) Remote output/input has nothing to do with REFR and REFW0 to REFWF, this module FB performs reading/writing to CC-Link master module during each execution.
(3) Please use "BIND" function in remote output.
(RY00_0F and RY70_7F are connected to OUT of BIND function)
(4) Please use "UNBIND" function in remote input.
(RX00_0F and RX70_7F are connected to IN of UNBIND function)

Public Variable (*1)

	Variable name	Variable type	Data type	Contents	Range	Initial value	Storage
Variable processing	MASTERRDY	Public variable	BOOL	Module ready of master station (TRUE: ON FALSE: OFF) Store the module READY status of master station. Execute input/output when module READY of master station is TRUE.	TRUE, FALSE	FALSE	System

*1 Execute reading/writing them by program.
It will not be displayed on the FB property window of PX Developer.

Function

<table>
<tr><th>Item</th><th colspan="3">Contents</th></tr>
<tr><td rowspan="5">Input condition signal (REFW0 to REFWF)</td><td colspan="2">Condition</td><td rowspan="2">Input (RWw0 to RWwF)</td></tr>
<tr><td>Input condition signal (REFW0 to REFWF)</td><td>Master module READY (MASTERRDY)</td></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Write the input value from input variable RWw0 to RWwF into CC-Link module.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Not write the input value from input variable RWw0 to RWwF into CC-Link module.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td rowspan="5">Output condition signal (REFR)</td><td colspan="2">Condition</td><td rowspan="2">Output (RWr0 to RWrF)</td></tr>
<tr><td>Output condition signal (REFR)</td><td>Master module READY (MASTERRDY)</td></tr>
<tr><td rowspan="2">TRUE</td><td>TRUE</td><td>Read the value stored in CC-Link module and output them from output variable RWr0 to RWrF.</td></tr>
<tr><td>FALSE</td><td rowspan="2">Keep the previous value of output variable RWr0 to RWrF.</td></tr>
<tr><td>FALSE</td><td>TRUE or FALSE</td></tr>
<tr><td>Others</td><td colspan="3">For detailed information of all the processing and settings of the corresponding module of this module FB, refer to the following manual:
● QJ61BT11 Control & Communication Link System Master/Local Module User's Manual.</td></tr>
</table>

POINT

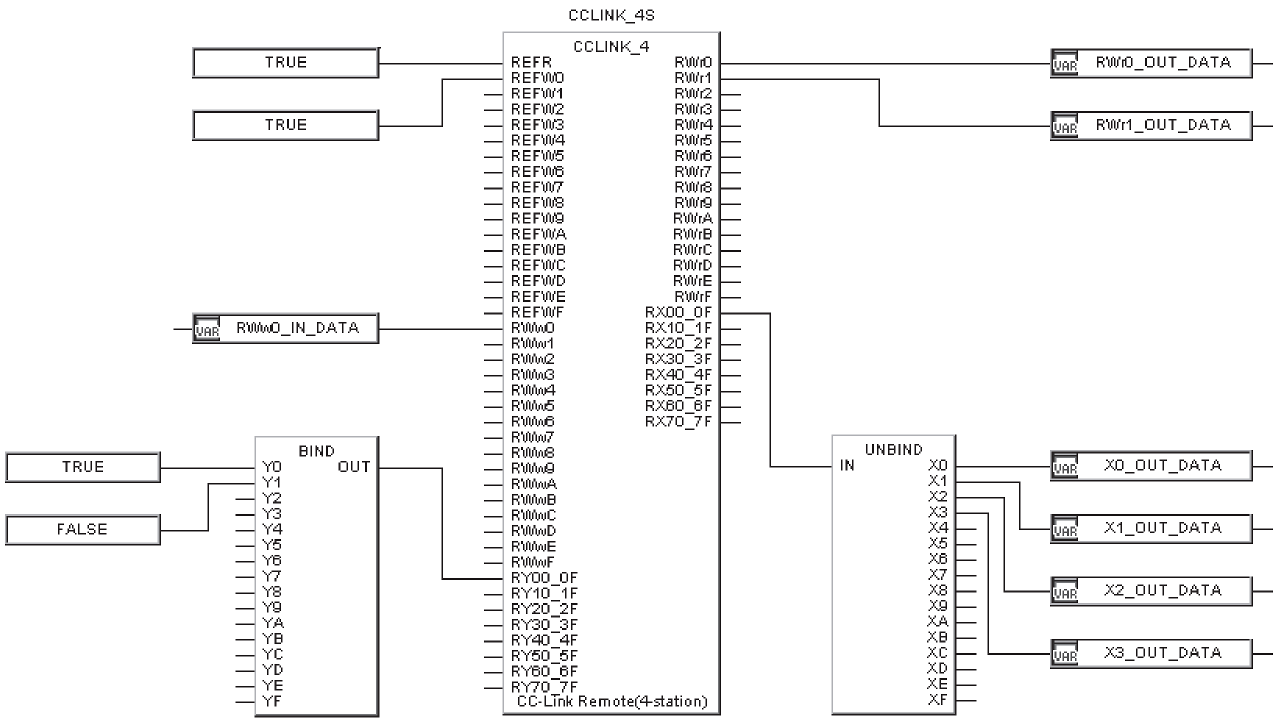
- Please do not set the auto refreshing parameter when using this module FB.
- Please set network parameter in GX application.

Error

Error may occur in the following cases, error code will be displayed on the FBD program diagnostics screen of PX Developer programming tool.

- Unable to communicate with CC-Link master module. (Error code: 1412)
- Abnormality of CC-Link master module is detected. (Error code: 1402)
- When the input and output number that is specified by the head I/O address of module FB declaration window is not intelligent function module. (Error code: 2110)

Program Example



APPENDIX

Appendix 1 List of Various Tag Type/Tag Data

The tag FB type parts hold their data area according to tag type (tag data).
The appendix covers the lists and detailed information of all tag data, and all functions of tag FB/tag access FB that can be used on all tags.
For details, refer to the following.

- Data area (tag data) list for all tag types
  Appendix 1.1
- Detailed information about data area (tag data) of all tag types.
  Appendix 1.2
- List of tag FB/tag access FB and functions that can be used on all tag types.
  Appendix 1.3

List of data area (tag data) of all tag types.

Tag type list	Tag type	Name
Loop tag	PID	PID control
	2PID	2-degree-of-freedom PID control
	2PIDH	2-degree-of-freedom advanced PID control
	PIDP	Position type PID control
	SPI	Sample PI control
	I-PD	I-PD control
	BPI	Blend PI control
	R	Ratio control
	ONF2	2 position ON/OFF control
	ONF3	3 position ON/OFF control
	PGS	Program setter
	PGS2	Multi-point program setter
	MOUT	Manual output
	MONI	Monitor
	SWM	Manual setter with monitor
	MWM	Manual output with monitor
	SEL	Loop selector
	BC	Batch counter
	PSUM	Pulse integrator
	PVAL	Position proportional output
	HTCL	Heating and cooling output
Status tag	NREV	Motor irreversible control
	REV	Motor reversible control
	MVAL1	ON/OFF control 1(without intermediate value)
	MVAL2	ON/OFF control 2(with intermediate value)
	TIMER1	Timer1(Timer stops when COMPLETE flag is on.)
	TIMER2	Timer2(Timer continues when COMPLETE flag is on.)
	COUNT1	Counter 1(Counter stops when COMPLETE flag is on.)
	COUNT2	Counter 2(Counter continues when COMPLETE flag is on.)
Alarm tag	ALM	Alarm
	ALM_64PT	64-points alarm
Message tag	MSG	Message
	MSG_64PT	64-points message

Appendix 1.1 Tag Data List of Various Tag Types

Data list for all tags is as follows.
For details, refer to following 1) to 10).
Data list explains from (1).

Table (1)-1 Tag memory table (PID) 1)

2)	3)	4)	5)		6)	7)	8)	9)	10)	11)
Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						

1) Table name: Tag types are indicated inside ( ).

2) Offset: indicates the offset word of memory data inside the tags.

3) Item: indicates tag data (tag member).
   *1 This tag data consists of multiple BOOL type variables.
   For details of BOOL type variables, refer to Appendix 1.2 (1).
   Besides, the items of this tag data are not public variables, while the BOOL type variables that form the items are public variables.
   When it is applied on FBD program, please use BOOL type variables described in Appendix 1.2 (1).
   (Example) Substitute the "AUTI" of tag FB variable name with variable "ABC"

   ○ ABC — TAG1. AUTI
   × ABC — TAG1. MDIH. AUTI
   Public variable named MDIH does not exist

4) Name: Indicates the name of tag data (tag member)

5) Setting/Storage range: Indicates the setting range of all items (*1)(*2)
   *1 Please refer to following range for PH, PL, HH, LL setting/storage range.
   PV high limit alarm value (PH) : (RL) to (RH) and (PL) < (PH)
   PV low limit alarm value (PL) : (RL) to (RH) and (PL) < (PH)
   PV high high limit alarm value (HH): (RL) to (RH) and PH) ≦ (HH)
   PV low low limit alarm value (LL) : (RL) to (RH) and (LL) ≦ (PL)
   *2 Please set the control cycle (CT) to the integral multiple of the execution cycle.
   An execution cycle can be the execution cycle set by project parameter of PX Developer, timer execution cycle set in program execution setting item and interruption interval of fixed scan interruption execution.

6) Unit: Indicate unit.

7) Initial value: Indicate default value.

8) Data type: Indicate the memory data configuration.
- INT : Integer data (1 word)
- DINT : Integer data (2 words)
- REAL : Floating point real data (2 words)
- WORD : Hexadecimal integral data (1 word)

9) Storage: To show whether it is allowed to read/write tag data via user program.
- User
  It is allowed to read/write.
  However, tag data with (condition *) can only be written only under the following conditions.
  Condition 1: When changing control mode (MODE) via user program, please do it by P_MCHG of tag access FB.
  Switch to the control mode (MODE) with TRUE as the corresponding bit item of mode inhibition (MDIH) is not allowed.
  Condition 2: Stop alarm (SPA) and output alarm (OOA) of alarm (ALM) are written via user program.
  When the user program sets stop alarm (SPA) as TRUE, stop alarm in the corresponding loop is processed.
  Additionally, the bit items except stop alarm (SPA) and output alarm (OOA) of alarm (ALM) are written by system. Please do not write by user.
  Condition 3: It is allowed to write only when the control modes are MAN and CMV.
- System
  It is allowed to read by user. Please do not write.
  The operations are not guaranteed in the case of writing.
  These are not displayed on FB property window of programming tool.
- Tag data access control
  Tag data access control can only be written from ActiveX Control application program which uses this control.
  For details about tag data access control, please refer to PX Developer Version 1 Operating Manual (Monitor Tool).

10) Number of digits after decimal point: Indicate the number of digits after the decimal point. N is indicated by number of digits after decimal point of + 9[N].

11) Tag access FB:Indicate the tag access FB which reads/writes the corresponding tag data.

(1) List of loop tag data
List of loop tag data is as follows.

Table (1)-1 Tag memory table (PID)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	1	1	—	1	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/PID(_T)/OUT1/DUTY
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Pubic
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Pubic
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Number of digits after the decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_OUT1/PID(_T)/DUTY
+14	SV	Setting value	RL	RH	UNIT	0.0	REAL	User	N	P_PID(_T)
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	P_PID(_T)
+18	MH	MV high limit	-10	110	%	100.0	REAL	User	1	P_OUT1/PID(_T)
+20	ML	MV low limit	-10	110	%	0.0	REAL	User	1	P_OUT1/PID(_T)
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	—
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	—
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low limit alarm hysterisis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.00	REAL	User	2	PID(_T)
+48	DML	Output variation rate high limit value	0	100	%	100.0	REAL	User	1	P_OUT1/DUTY
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_PID(_T)
+52	P	Gain	0	999	—	1.00	REAL	User	2	P_PID(_T)
+54	I	Integral time	0	9999	s	10.0	REAL	User	1	P_PID(_T)
+56	D	Derivative time	0	9999	s	0.0	REAL	User	1	P_PID(_T)
+58	GW	Gap width	0	100	%	0.0	REAL	User	1	P_PID(_T)
+60	GG	Gap gain	0	99	—	1.0	REAL	User	1	P_PID(_T)
+62	MVP	MV internal operation value	-999999	999999	—	0.0	REAL	System	1	—
+68	CTDUTY	Control output cycle	0	9999	s	1.0	REAL	User	2	P_DUTY
+70	AT1STEPMV	Step Manipulated variable for AT1	-100	100	%	0.0	REAL	User (*2)	1	P_PID(_T)
+72	AT1ST	Sampling interval time for AT1	0	9999	s	1.00	REAL	User (*2)	2	P_PID(_T)
+74	AT1TOUT1	Time-out period for AT1	0	9999	s	100.0	REAL	User (*2)	1	P_PID(_T)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+76	AT1TOUT2	Time-out period after Maximum slope for AT1	0	9999	s	10.0	REAL	User (*2)	1	P_PID (_T)
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	Tag data access control	—	—
+95	DIM (*1)	Monitor input buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) and Appendix 1.2 (1) (a).
*2 For Version 1.31H or later, displayed on the FB Property window in Programming Tool.

Table (1)-2 Tag storage table (2PID)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	14	14	—	14	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/2PID (_T) /OUT1/DUTY
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_OUT1/2PID(_T)/ DUTY
+14	SV	Setting value	RL	RH	UNIT	0.0	REAL	User	N	2PID(_T)
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	2PID(_T)
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_OUT1/2PID(_T)
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_OUT1/2PID(_T)
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value for	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	—
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	—
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low limit alarm hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	P_2PID(_T)
+48	DML	Output variation rate high limit value	0	100	%	100.0	REAL	User	1	P_OUT1
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_2PID(_T)
+52	P	Gain	0	999	—	1.00	REAL	User	2	P_2PID(_T)
+54	I	Integral time	0	9999	s	10.0	REAL	User	1	P_2PID(_T)
+56	D	Derivative time	0	9999	s	0.0	REAL	User	1	P_2PID(_T)
+58	GW	Gap width	0	100	%	0.0	REAL	User	1	P_2PID(_T)
+60	GG	Gap gain	0	999	—	1.0	REAL	User	1	P_2PID(_T)
+62	MVP	MV internal operation value	-999999	999999	—	0.0	REAL	System	1	—
+64	ALPHA2	2-degree-of-freedom parameter α	0	1	—	0.00	REAL	User	2	P_2PID(_T)
+66	BETA2	2-degree-of-freedom parameter β	0	1	—	1.00	REAL	User	2	P_2PID(_T)
+68	CTDUTY	Control output cycle	0	9999	s	1.0	REAL	User	2	P_DUTY
+70	AT1STEPMV	Step Manipulated variable for AT1 use	-100	100	%	0.0	REAL	User (*2)	1	P_2PID(_T)
+72	AT1ST	Sampling period for AT1 use	0	9999	s	1.00	REAL	User (*2)	2	P_2PID(_T)
+74	AT1TOUT1	Time-out period for AT1.	0	9999	s	100.0	REAL	User (*2)	1	P_2PID(_T)
+76	AT1TOUT2	Time-out period after maximal slope for AT1	0	9999	s	10.0	REAL	User (*2)	1	P_2PID(_T)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	Tag data access control	—	—
+95	DIM (*1)	Monitor input buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) and Appendix 1.2 (1) (a).
*2 For Version 1.31H or later, displayed on the FB Property window in Programming Tool.

Table (1)-3 Tag storage table (2PIDH)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	17	17	—	17	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/ P_2PIDH(_T)_/ P_OUT3_
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*2)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*2)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*2)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_OUT3_/ P_2PIDH(_T)_
+14	SVC	Setting value (current)	RL	RH	UNIT	0.0	REAL	System	N	P_2PIDH(_T)_
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	P_2PIDH(_T)_
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_OUT3_/ P_2PIDH(_T)_
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_OUT3_/ P_2PIDH(_T)_
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	P_2PIDH(_T)_
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	P_2PIDH(_T)_
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low limit alarm hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	P_2PIDH(_T)_
+48	DML	Output variation rate high limit value	0	100	%	100.0	REAL	User	1	P_OUT3_
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_2PIDH(_T)_
+52	P	Gain	0	999	—	1.00	REAL	User	2	P_2PIDH(_T)_
+54	I	Integral time	0	9999	s	10.0	REAL	User	1	P_2PIDH(_T)_
+56	D	Derivative time	0	9999	s	0.0	REAL	User	1	P_2PIDH(_T)_
+58	GW	Gap width	0	100	—	0.0	REAL	User	1	P_2PIDH(_T)_
+60	GG	Gap gain	0	99	—	1.0	REAL	User	1	P_2PIDH(_T)_
+62	MVP	MV internal operation value	-999999	999999	—	0.0	REAL	System	1	—
+64	ALPHA2	2-degree-of-freedom parameter α	0	1	—	0.00	REAL	User	2	P_2PIDH(_T)_
+66	BETA2	2-degree-of-freedom parameter β	0	1	—	1.00	REAL	User	2	P_2PIDH(_T)_
+68	CTDUTY	Control output cycle	0	9999	s	1.00	REAL	User	2	Reserved
+70	AT1STEPMV	Step Manipulated variable for AT1 use	-100	100	%	0.0	REAL	User (*4)	1	P_2PIDH(_T)_
+72	AT1ST	Sampling period for AT1 use	0	9999	s	1.00	REAL	User (*4)	2	P_2PIDH(_T)_

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+74	AT1TOUT1	Time-out period for AT1/AT2	0	9999	s	100.0	REAL	User (*4)	1	P_2PIDH(_T)_
+76	AT1TOUT2	Time-out period after maximal slope for AT1	0	9999	s	10.0	REAL	User (*4)	1	P_2PIDH(_T)_
+78	AT2HS	Hysterisis for AT2	0	10	%	1.0	REAL	User (*4)	1	P_2PIDH(_T)_
+80	AT2MVH	Output High Limit Value for AT2	0	100	%	100.0	REAL	User (*4)	1	P_2PIDH(_T)_
+82	AT2MVL	Output Low Limit Value for AT2	0	100	%	0.0	REAL	User (*4)	1	P_2PIDH(_T)_
+86	ATTYPE (*3)	Control Type for AT	0	4	—	1	INT	User (*4)	—	P_2PIDH(_T)_
+87	ALM2 (*1)	Alarm 2	0	$FFFF_H$	—	0000_H	WORD	System	—	Public
+88	INH2 (*1)	Disable alarm2 detection	0	$FFFF_H$	—	0000_H	WORD	User	—	Public
+89	ALML2 (*1)	Alarm level 2	0	$FFFF_H$	—	0000_H	WORD	User	—	Public
+90	SV	Setting value (target)	RL	RH	UNIT	0.0	REAL	User	N	—
+92	DSVL	SV variation rate high limit value	0	100	%	100.0	REAL	User	1	P_2PIDH(_T)_
+94	DOM (*1)	Monitor output buffer	0	$FFFF_H$	—	0000_H	WORD	Tag data access control	—	—
+95	DIM (*1)	Monitor input buffer	0	$FFFF_H$	—	0000_H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) (b).

*2 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

*3 Control type for AT specifies auto tuning type and control type.

ATTYPE	TYPE (Control type)
0	Step Response method
1	Limit Cycle method (Constant-value PI control)
2	Limit Cycle method (Constant-value PID control)
3	Limit Cycle method (Follow-up PI control)
4	Limit Cycle method (Follow-up PID control)

*4 For Version 1.31H or later, displayed on the FB Property window in Programming Tool.

Table (1)-4 Tag storage table (PIDP)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	2	2	—	2	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/PIDP (_T)
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG (*2)
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Number of digits after the decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_PIDP (_T) (*2)
+14	SV	Setting value	RL	RH	UNIT	0.0	REAL	User	N	P_PIDP (_T) (*2)
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	P_PIDP (_T) (*2)
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_PIDP (_T) (*2)
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_PIDP (_T) (*2)
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	—
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	—
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low limit alarm value hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	P_PIDP(_T) (*2)
+48	DML	Variation rate high limit value	0	100	%	100.0	REAL	User	1	P_PIDP(_T) (*2)
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_PIDP(_T) (*2)
+52	P	Gain	0	999	—	1.00	REAL	User	2	P_PIDP(_T) (*2)
+54	I	Integral time	0	9999	s	10.0	REAL	User	1	P_PIDP(_T) (*2)
+56	D	Derivation time	0	9999	s	0.0	REAL	User	1	P_PIDP(_T) (*2)
+58	GW	Gap width	0	100	%	0.0	REAL	User	1	P_PIDP(_T) (*2)
+60	GG	Gap gain	0	99	—	1.0	REAL	User	1	P_PIDP(_T) (*2)
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variable, refer to Appendix 1.2 (1).

*2 "P_PIDP_EX(_T)_" is included.

Table (1)-5 Tag memory table (SPI)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal points	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	3	3	—	3	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/SPI(_T) /OUT1/DUTY
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Number of digits after the decimal points	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_OUT1/DUTY
+14	SV	Setting value	RL	RH	UNIT	0.0	REAL	User	N	P_SPI (_T)
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	P_SPI (_T)
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_OUT1
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_OUT1
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	—
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	—
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low alarm value hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	ST	Operation time	0	9999	s	0.0	REAL	User	1	P_SPI (_T)
+48	DML	Output variation rate high limit value	0	100	%	100.0	REAL	User	1	P_OUT1/DUTY
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_SPI (_T)
+52	P	Gain	0	999	—	1.00	REAL	User	2	P_SPI (_T)
+54	I	Integral time	0	9999	s	10.0	REAL	User	1	P_SPI (_T)
+56	STHT	Sampling period	0	9999	s	0.0	REAL	User	1	P_SPI (_T)
+58	GW	Gap width	0	100	%	0.0	REAL	User	1	P_SPI (_T)
+60	GG	Gap gain	0	99	—	1.0	REAL	User	1	P_SPI (_T)
+62	MVP	MV internal operation value	-999999	999999	—	0.0	REAL	System	1	—
+68	CTDUTY	Control output cycle	0	9999	s	1.0	REAL	User	2	P_DUTY
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-6 Tag memory table (IPD)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	4	4	—	4	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/IPD (_T) /OUT1/DUTY
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Number of digits after the decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_OUT1/DUTY
+14	SV	Setting value	RL	RH	UNIT	0.0	REAL	User	N	P_IPD (_T)
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	P_IPD (_T)
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_OUT1
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_OUT1
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	—
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	—
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low alarm value hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	P_IPD (_T)
+48	DML	Output variation rate high limit value	0	100	%	100.0	REAL	User	1	P_OUT1/DUTY
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_IPD (_T)
+52	P	Gain	0	999	—	1.00	REAL	User	2	P_IPD (_T)
+54	I	Integral time	0	9999	s	10.0	REAL	User	1	P_IPD (_T)
+56	D	Derivation time	0	9999	s	0.0	REAL	User	1	P_IPD (_T)
+58	GW	Gap width	0	100	%	0.0	REAL	User	1	P_IPD (_T)
+60	GG	Gap gain	0	99	—	1.0	REAL	User	1	P_IPD (_T)
+62	MVP	MV internal operation value	-999999	999999	—	0.0	REAL	System	1	—
+68	CTDUTY	Control output cycle	0	9999	s	1.0	REAL	User	2	P_DUTY
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-7 Tag memory table (BPI)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Numbers of digits after the decimal points	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	5	5	—	5	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/BPI(_T) /OUT1/DUTY
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Numbers of digits after the decimal points	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_OUT1/DUTY
+14	SV	Setting value	RL	RH	UNIT	0.0	REAL	User	N	P_BPI (_T)
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	P_BPI (_T)
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_OUT1
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_OUT1
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	—
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	—
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low limit alarm hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	P_BPI(_T)
+48	DML	Output variation rate high limit value	0	100	%	100.0	REAL	User	1	P_OUT1/DUTY
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_BPI(_T)
+52	P	Gain	0	999	—	1.00	REAL	User	2	P_BPI(_T)
+54	I	Integral time	0	9999	s	10.0	REAL	User	1	P_BPI(_T)
+56	SDV	DV cumulative value	-999999	999999	—	0.0	REAL	System	1	P_BPI(_T)
+58	GW	Gap width	0	100	%	0.0	REAL	User	1	P_BPI(_T)
+60	GG	Gap gain	0	99	—	1.0	REAL	User	1	P_BPI(_T)
+62	MVP	MV internal operation value	-999999	999999	—	0.0	REAL	System	1	—
+68	CTDUTY	Control output cycle	0	9999	s	1.0	REAL	User	2	P_DUTY
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-8 Tag memory table(R)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	6	6	—	6	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFF$_H$	—	0008$_H$	WORD	User (condition 1)	—	P_MCHG/R (_T) /OUT2
+2	MDIH (*1)	Disable mode	0	FFFF$_H$	—	0600$_H$	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFF$_H$	—	0000$_H$	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFF$_H$	—	0000$_H$	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFF$_H$	—	0000$_H$	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002$_H$	—	0000$_H$	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_OUT2
+14	SV	Setting value (SPR)	RMIN	RMAX	%	0.0	REAL	User	1	P_R(_T)
+16	BIAS	Bias	-999999	999999	—	0.0	REAL	User	1	P_R(_T)
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_OUT2
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_OUT2
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RMIN	RMAX	%	100.0	REAL	User	N	—
+36	SL	SV low limit value	RMIN	RMAX	%	0.0	REAL	User	N	—
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low alarm value hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	P_R (_T)
+48	DML	Output variation rate high limit value	0	100	%	100.0	REAL	User	1	P_OUT2
+50	DR	Variation rate limit value	0	999999	%	100.0	REAL	User	1	P_R (_T)
+52	RMAX	Ratio high limit value	0	999999	%	100.0	REAL	User	1	P_R (_T)
+54	RMIN	Ratio low limit value	0	999999	%	0.0	REAL	User	1	P_R (_T)
+56	RN	Ratio current value	0	999999	%	0.0	REAL	System	1	P_R (T)
+94	DOM (*1)	Monitor output buffer	0	FFFF$_H$	—	0000$_H$	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-9 Tag storage table (ONF2)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	7	7	—	7	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/ONF2 (_T)
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	ONF2 (_T)
+14	SV	Setting value	RL	RH	UNIT	0.0	REAL	User	N	ONF2 (_T)
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	ONF2 (_T)
+18	HS0	Hysteresis	0	100	%	0.0	REAL	User	1	ONF2 (_T)
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	—
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	—
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low alarm value hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm detection time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	ONF2 (_T)
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-10 Tag memory table (ONF3)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	8	8	—	8	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/ONF3 (_T)
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	ONF3 (_T)
+14	SV	Setting value	RL	RH	UNIT	0.0	REAL	User	N	ONF3 (_T)
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	ONF3 (_T)
+18	HS0	Hysteresis	0	100	%	0.0	REAL	User	1	ONF3 (_T)
+20	HS1	Hysteresis	0	100	%	0.0	REAL	User	1	ONF3 (_T)
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high high limit value	RL	RH	UNIT	100.0	REAL	User	N	—
+36	SL	SV low low limit value	RL	RH	UNIT	0.0	REAL	User	N	—
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low alarm value hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	ONF3 (_T)
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-11 Tag memory table (MONI)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	11	11	—	11	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	System	—	—
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0000H	WORD	User	—	—
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low alarm value hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-12 Tag memory table (SWM)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	19	19	—	19	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0400H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	System	1	P_MSET_
+14	SVC	Setting value (current)	RL	RH	UNIT	0.0	REAL	System	N	P_MSET_
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	P_MSET_
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	P_MSET_
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	P_MSET_
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low alarm value hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	P_MSET_
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_MSET_
+87	ALM2 (*1)	Alarm 2	0	FFFFH	—	0000H	WORD	System	—	Public
+88	INH2 (*1)	Disable alarm2 detection	0	FFFFH	—	0000H	WORD	User	—	Public
+89	ALML2 (*1)	Alarm level 2	0	FFFFH	—	0000H	WORD	User	—	Public
+90	SV	Setting value (target)	RL	RH	UNIT	0.0	REAL	User	N	—
+92	DSVL	SV variation rate high limit value	0	100	%	100.0	REAL	User	1	P_MSET_
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) and Appendix 1.2 (1) (c).

Table (1)-13 Tag memory table (MWM)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	12	12	—	12	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/MOUT
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0630H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_MOUT
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_MOUT
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_MOUT
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low alarm value hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-14 Tag memory table (BC)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	15	15	—	15	INT	System	—	—
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+10	PV	Process variable (integral part)	0	99999999	UNIT	0	DINT	System	—	P_PSUM
+12	SUM2	Process variable (decimal part)	0	999	—	0	DINT	System	—	P_PSUM
+14	SV1	Setting value 1 (preset)	0	99999999	UNIT	0	DINT	User	—	P_BC
+16	SV2	Setting value 2 (preset)	0	99999999	UNIT	0	DINT	User	—	P_BC
+18	SV	Setting value	0	99999999	UNIT	0	DINT	User	—	—
+26	PH	PV high limit alarm value	0	99999999	UNIT	0	DINT	User	—	P_BC
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_BC
+44	DPL	Variation rate alarm value	0	99999999	UNIT	99999999	DINT	User	—	P_BC
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	Tag data access control	—	—
+95	DIM (*1)	Monitor input buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) and Appendix 1.2 (1) (d).

Table (1)-15 Tag memory table (PSUM)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	16	16	—	16	INT	System	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+10	PV	Process variable (integral part)	0	99999999	UNIT	0	DINT	System	—	P_PSUM
+12	SUM2	Process variable (decimal part)	0	999	—	0	DINT	System	—	P_PSUM
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	Tag data access control	—	—
+95	DIM (*1)	Monitor input buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) (e).

Table (1)-16 Tag memory table (SEL)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	13	13	—	13	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/SEL (_T1) (_T2) (_T3_)
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Number of digits after the decimal point	0	4	—	1	INT	User	—	—
+10	PV	Select Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_SEL (_T1) (_T2) (_T3_)
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_SEL (_T1) (_T2) (_T3_)
+14	PV1	Process variable 1	RL	RH	UNIT	0.0	REAL	System	N	P_SEL (_T1) (_T2) (_T3_)
+16	PV2	Process variable 2	RL	RH	UNIT	0.0	REAL	System	N	P_SEL (_T1) (_T2) (_T3_)
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_SEL (_T1) (_T2) (_T3_)
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_SEL (_T1) (_T2) (_T3_)
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_SEL (_T1) (_T2) (_T3_)
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_SEL (_T1) (_T2) (_T3_)
+26	SLNO	Selection No.	1	2	—	0	INT	System	—	P_SEL (_T1) (_T2) (_T3_)
+46	CT	Control cycle	0	9999	s	1.00	REAL	User	2	P_SEL_T3_
+48	DML	Output variation rate high limit	0	100	%	100.0	REAL	User	1	P_SEL (_T1) (_T2) (_T3_)
+62	MVP	MV internal operation value	-999999	999999	—	0.0	REAL	System	1	P_SEL_T3_

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-17 Tag memory table (MOUT)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	10	10	—	10	INT	System	—	—
+1	MODE(*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/ MOUT
+2	MDIH(*1)	Disable mode	0	FFFFH	—	0630H	WORD	User	—	P_MCHG
+3	ALM(*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+5	ALML(*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Number of digits after the decimal point	0	4	—	1	INT	User	—	—
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	User (condition 3)	1	P_MOUT
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_MOUT
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_MOUT

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-18 Tag memory table (PGS)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	9	9	—	9	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG/PGS
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PTNO	The number of points	0	16	—	0	INT	User	—	P_PGS
+12	MV	Manipulated volume	-10	110	%	0.0	REAL	User (condition 3)	N	P_PGS
+14	SV	Setting value	0	999999	s	0.0	REAL	User	1	P_PGS
+16	TYP	Operation type	0	1	—	0	INT	User	—	P_PGS
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	N	P_PGS
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	N	P_PGS
+22	SV1	Setting time 1	0	999999	s	0.0	REAL	User	1	P_PGS
+24	SV2	Setting time 2	0	999999	s	0.0	REAL	User	1	P_PGS
+26	SV3	Setting time 3	0	999999	s	0.0	REAL	User	1	P_PGS
+28	SV4	Setting time 4	0	999999	s	0.0	REAL	User	1	P_PGS
+30	SV5	Setting time 5	0	999999	s	0.0	REAL	User	1	P_PGS
+32	SV6	Setting time 6	0	999999	s	0.0	REAL	User	1	P_PGS
+34	SV7	Setting time 7	0	999999	s	0.0	REAL	User	1	P_PGS
+36	SV8	Setting time 8	0	999999	s	0.0	REAL	User	1	P_PGS
+38	SV9	Setting time 9	0	999999	s	0.0	REAL	User	1	P_PGS
+40	SV10	Setting time 10	0	999999	s	0.0	REAL	User	1	P_PGS
+42	SV11	Setting time 11	0	999999	s	0.0	REAL	User	1	P_PGS
+44	SV12	Setting time 12	0	999999	s	0.0	REAL	User	1	P_PGS
+46	SV13	Setting time 13	0	999999	s	0.0	REAL	User	1	P_PGS
+48	SV14	Setting time 14	0	999999	s	0.0	REAL	User	1	P_PGS
+50	SV15	Setting time 15	0	999999	s	0.0	REAL	User	1	P_PGS
+52	SV16	Setting time 16	0	999999	s	0.0	REAL	User	1	P_PGS
+54	MV1	Setting output 1	-10	110	%	0.0	REAL	User	N	P_PGS
+56	MV2	Setting output 2	-10	110	%	0.0	REAL	User	N	P_PGS
+58	MV3	Setting output 3	-10	110	%	0.0	REAL	User	N	P_PGS
+60	MV4	Setting output 4	-10	110	%	0.0	REAL	User	N	P_PGS
+62	MV5	Setting output 5	-10	110	%	0.0	REAL	User	N	P_PGS
+64	MV6	Setting output 6	-10	110	%	0.0	REAL	User	N	P_PGS
+66	MV7	Setting output 7	-10	110	%	0.0	REAL	User	N	P_PGS
+68	MV8	Setting output 8	-10	110	%	0.0	REAL	User	N	P_PGS
+70	MV9	Setting output 9	-10	110	%	0.0	REAL	User	N	P_PGS
+72	MV10	Setting output 10	-10	110	%	0.0	REAL	User	N	P_PGS
+74	MV11	Setting output 11	-10	110	%	0.0	REAL	User	N	P_PGS
+76	MV12	Setting output 12	-10	110	%	0.0	REAL	User	N	P_PGS
+78	MV13	Setting output 13	-10	110	%	0.0	REAL	User	N	P_PGS
+80	MV14	Setting output 14	-10	110	%	0.0	REAL	User	N	P_PGS
+82	MV15	Setting output 15	-10	110	%	0.0	REAL	User	N	P_PGS
+84	MV16	Setting output 16	-10	110	%	0.0	REAL	User	N	P_PGS

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1).

Table (1)-19 Tag memory table (PGS2)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	18	18	–	18	INT	System	–	–
+1	MODE(*1)	Control mode	0	FFFFH	–	0008H	WORD	User (condition 1)	–	P_MCHG/P_PGS2_
+2	MDIH(*1)	Disable mode	0	FFFFH	–	0000H	WORD	User	–	P_MCHG
+3	ALM(*1)	Alarm	0	FFFFH	–	0000H	WORD	User (condition 2)	–	Public
+4	INH(*1)	Disable alarm detection	0	FFFFH	–	0000H	WORD	User	–	Public
+5	ALML(*1)	Alarm level	0	FFFFH	–	0000H	WORD	User	–	–
+6	CTNO	Lockout tag No.	0	32	–	0	INT	System	–	–
+7	CTFN	Lockout tag function	0	0002H	–	0000H	WORD	System	–	–
+8	UNIT	Unit	0	127	–	0	INT	User	–	–
+9	N	Digit number after decimal point	0	4	–	1	INT	User	–	–
+10	STNO	Number of step setting	0	32	–	0	INT	User	–	P_PGS2_
+11	PVSTART	PV start type	0	2	–	0	INT	User	–	P_PGS2_
+12	SV	Setting value	RL	RH	UNIT	0.0	REAL	User (condition 3)	N	P_PGS2_
+14	STC	Executing step No.	0	32	–	0	INT	User	–	P_PGS2_
+15	T	Time in the step	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+16	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PGS2_
+18	SH	SV high limit value	-32768	32767	UNIT (*2)	100	INT	User	–	P_PGS2_
+19	SL	SV low limit value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+20	TYP(*1)	Operation type	0	FFFFH	-	0001H	WORD	User	–	P_PGS2_
+21	WAIT	Wait width	0	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+22	RH	Engineering value high limit	-32768	32767	UNIT (*2)	100	INT	User	–	P_PGS2_
+23	RL	Engineering value low limit	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+26	SV0	Start point	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+27	SV0C	Start point (current)	-32768	32767	UNIT (*2)	0	INT	System	–	P_PGS2_
+28	T1	Step 1 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+29	SV1	Step 1 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+30	T2	Step 2 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+31	SV2	Step 2 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+32	T3	Step 3 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+33	SV3	Step 3 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) and Appendix 1.2 (1) (f).

*2 The tag data is set in integer, ignoring the digit number after decimal point (N).
If the setting after the decimal point is required depending on the engineering value range, set the tag data as follows.
(Example) When the setting value is 1.5MPa, convert its unit to 1500kPa to fit in the range from -32768 to 32767.

*3 When the unit of time (TUNIT) is FALSE, "s" is used and "min" is used for TRUE.

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+34	T4	Step 4 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+35	SV4	Step 4 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+36	T5	Step 5 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+37	SV5	Step 5 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+38	T6	Step 6 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+39	SV6	Step 6 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+40	T7	Step 7 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+41	SV7	Step 7 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+42	T8	Step 8 time span	0	32767	S (min) (*3)	0	INT	User	–	P_PGS2_
+43	SV8	Step 8 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+44	T9	Step 9 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+45	SV9	Step 9 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+46	T10	Step 10 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+47	SV10	Step 10 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+48	T11	Step 11 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+49	SV11	Step 11 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+50	T12	Step 12 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+51	SV12	Step 12 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+52	T13	Step 13 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+53	SV13	Step 13 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+54	T14	Step 14 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+55	SV14	Step 14 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_

*2 The tag data is set in integer, ignoring the digit number after decimal point (N).
If the setting after the decimal point is required depending on the engineering value range, set the tag data as follows.
(Example) When the setting value is 1.5MPa, convert its unit to 1500kPa to fit in the range from -32768 to 32767.

*3 When the unit of time (TUNIT) is FALSE, "s" is used and "min" is used for TRUE.

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+56	T15	Step 15 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+57	SV15	Step 15 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+58	T16	Step 16 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+59	SV16	Step 16 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+60	T17	Step 17 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+61	SV17	Step 17 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+62	T18	Step 18 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+63	SV18	Step 18 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+64	T19	Step 19 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+65	SV19	Step 19 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+66	T20	Step 20 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+67	SV20	Step 20 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+68	T21	Step 21 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+69	SV21	Step 21 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+70	T22	Step 22 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+71	SV22	Step 22 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+72	T23	Step 23 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+73	SV23	Step 23 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+74	T24	Step 24 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+75	SV24	Step 24 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+76	T25	Step 25 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+77	SV25	Step 25 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_

*2 The tag data is set in integer, ignoring the digit number after decimal point (N).
If the setting after the decimal point is required depending on the engineering value range, set the tag data as follows.
(Example) When the setting value is 1.5MPa, convert its unit to 1500kPa to fit in the range from -32768 to 32767.

*3 When the unit of time (TUNIT) is FALSE, "s" is used and "min" is used for TRUE.

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+78	T26	Step 26 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+79	SV26	Step 26 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+80	T27	Step 27 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+81	SV27	Step 27 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+82	T28	Step 28 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+83	SV28	Step 28 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+84	T29	Step 29 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+85	SV29	Step 29 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+86	T30	Step 30 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+87	SV30	Step 30 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+88	T31	Step 31 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+89	SV31	Step 31 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+90	T32	Step 32 time span	0	32767	s (min) (*3)	0	INT	User	–	P_PGS2_
+91	SV32	Step 32 setting value	-32768	32767	UNIT (*2)	0	INT	User	–	P_PGS2_
+94	DOM(*1)	Monitor output buffer	0	FFFFH	–	0000H	WORD	Tag data access control	–	–
+95	DIM(*1)	Monitor input buffer	0	FFFFH	–	0000H	WORD	System	–	–

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) and Appendix 1.2 (1) (f).

*2 The tag data is set in integer, ignoring the digit number after decimal point (N).
If the setting after the decimal point is required depending on the engineering value range, set the tag data as follows.
(Example) When the setting value is 1.5MPa, convert its unit to 1500kPa to fit in the range from -32768 to 32767.

*3 When the unit of time (TUNIT) is FALSE, "s" is used and "min" is used for TRUE.

Table (1)-20 Tag storage table (PFC_SF)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	20	20	—	20	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	System	1	P_PFC_SF_
+14	SVC	Setting value (current)	RL	RH	UNIT	0.0	REAL	System	N	P_PFC_SF_
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	P_PFC_SF_
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_PFC_SF_
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_PFC_SF_
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	P_PFC_SF_
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	P_PFC_SF_
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low limit alarm hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	P_PFC_SF_
+48	DML	Output variation rate high limit value	0	100	%	100.0	REAL	User	1	P_PFC_SF_
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_PFC_SF_
+52	DM	Dead time	0	999999	s	0.0	REAL	User	1	P_PFC_SF_
+54	KM	Gain	0	999	—	1.0	REAL	User	2	P_PFC_SF_
+56	TM	Time constant	0	999999	s	10.0	REAL	User	2	P_PFC_SF_
+62	MVP	MV internal operation value	1	999999	—	0.0	REAL	System	1	—
+64	HORIZON	Coincidence horizon	0	999999	—	1.0	REAL	User	1	P_PFC_SF_
+66	TRBF	Reference model time constant	0	999999	s	1.0	REAL	User	2	P_PFC_SF_
+87	ALM2 (*1)	Alarm 2	0	FFFFH	—	0000H	WORD	System	—	Public
+88	INH2 (*1)	Disable alarm2 detection	0	FFFFH	—	0000H	WORD	User	—	Public
+89	ALML2 (*1)	Alarm level 2	0	FFFFH	—	0000H	WORD	User	—	Public
+90	SV	Setting value (target)	RL	RH	UNIT	0.0	REAL	User	N	—
+92	DSVL	SV variation rate high limit value	0	100	%	100.0	REAL	User	1	P_PFC_SF_
+94	DOM (*1)	Monitor output buffer	0	FFFFH	s	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) (g).

Table (1)-21 Tag storage table (PFC_SS)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	21	21	—	21	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-10	110	%	0.0	REAL	System	1	P_PFC_SS_
+14	SVC	Setting value (current)	RL	RH	UNIT	0.0	REAL	System	N	P_PFC_SS_
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	P_PFC_SS_
+18	MH	MV high limit value	-10	110	%	100.0	REAL	User	1	P_PFC_SS_
+20	ML	MV low limit value	-10	110	%	0.0	REAL	User	1	P_PFC_SS_
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	P_PFC_SS_
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	P_PFC_SS_
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low limit alarm hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	P_PFC_SS_
+48	DML	Output variation rate high limit value	0	100	%	100.0	REAL	User	1	P_PFC_SS_
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_PFC_SS_
+52	DM	Dead time	0	999999	s	0.0	REAL	User	1	P_PFC_SS_
+54	KM	Gain	0	999	—	1.0	REAL	User	2	P_PFC_SS_
+56	TM1	Time constant1	0	999999	s	10.0	REAL	User	2	P_PFC_SS_
+58	TM2	Time constant2	0	999999	s	1.0	REAL	User	2	P_PFC_SS_
+62	MVP	MV internal operation value	1	999999	—	0.0	REAL	System	1	—
+64	ALPHA2	2-degree-of-freedom parameter α	0	999999	—	1.0	REAL	User	1	P_PFC_SS_
+66	BETA2	2-degree-of-freedom parameter β	0	999999	s	1.0	REAL	User	2	P_PFC_SS_
+87	ALM2 (*1)	Alarm 2	0	FFFFH	—	0000H	WORD	System	—	Public
+88	INH2 (*1)	Disable alarm2 detection	0	FFFFH	—	0000H	WORD	User	—	Public
+89	ALML2 (*1)	Alarm level 2	0	FFFFH	—	0000H	WORD	User	—	Public
+90	SV	Setting value (target)	RL	RH	UNIT	0.0	REAL	User	N	—
+92	DSVL	SV variation rate high limit value	0	100	%	100.0	REAL	User	1	P_PFC_SS_
+94	DOM (*1)	Monitor output buffer	0	FFFFH	s	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) (g).

Table (1)-22 Tag storage table (PFC_INT)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	22	22	—	22	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	P_MCHG
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	UNIT	Unit	0	127	—	0	INT	User	—	—
+9	N	Digits number after decimal point	0	4	—	1	INT	User	—	—
+10	PV	Process variable	RL	RH	UNIT	0.0	REAL	System	N	P_PHPL
+12	MV	Manipulated variable	-110	110	%	0.0	REAL	System	1	P_PFC_INT_
+14	SVC	Setting value (current)	RL	RH	UNIT	0.0	REAL	System	N	P_PFC_INT_
+16	DV	Deviation	-110	110	%	0.0	REAL	System	1	P_PFC_INT_
+18	MH	MV high limit value	-110	110	%	100.0	REAL	User	1	P_PFC_INT_
+20	ML	MV low limit value	-110	110	%	-100.0	REAL	User	1	P_PFC_INT_
+22	RH	PV engineering value high limit	-999999	999999	UNIT	100.0	REAL	User	N	P_PHPL
+24	RL	PV engineering value low limit	-999999	999999	UNIT	0.0	REAL	User	N	P_PHPL
+26	PH	PV high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+28	PL	PV low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+30	HH	PV high high limit alarm value	RL	RH	UNIT	100.0	REAL	User	N	P_PHPL
+32	LL	PV low low limit alarm value	RL	RH	UNIT	0.0	REAL	User	N	P_PHPL
+34	SH	SV high limit value	RL	RH	UNIT	100.0	REAL	User	N	P_PFC_INT_
+36	SL	SV low limit value	RL	RH	UNIT	0.0	REAL	User	N	P_PFC_INT_
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low limit alarm hysteresis	0	100	%	0.0	REAL	User	1	P_PHPL
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	P_PHPL
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	P_PHPL
+46	CT	Control cycle	0	9999	s	1.0	REAL	User	2	P_PFC_INT_
+48	DML	Output variation rate high limit value	0	100	%	100.0	REAL	User	1	P_PFC_INT_
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	P_PFC_INT_
+52	DM	Dead time	0	999999	s	0.0	REAL	User	1	P_PFC_INT_
+54	KM	Gain	0	999	—	1.0	REAL	User	2	P_PFC_INT_
+62	MVP	MV internal operation value	1	999999	—	0.0	REAL	System	1	—
+64	HORIZON	Coincidence horizon	0	999999	—	1.0	REAL	User	1	P_PFC_INT_
+66	TRBF	Reference model time constant	0	999999	s	1.0	REAL	User	2	P_PFC_INT_
+87	ALM2 (*1)	Alarm 2	0	FFFFH	—	0000H	WORD	System	—	Public
+88	INH2 (*1)	Disable alarm2 detection	0	FFFFH	—	0000H	WORD	User	—	Public
+89	ALML2 (*1)	Alarm level 2	0	FFFFH	—	0000H	WORD	User	—	Public
+90	SV	Setting value (target)	RL	RH	UNIT	0.0	REAL	User	N	—
+92	DSVL	SV variation rate high limit value	0	100	%	100.0	REAL	User	1	P_PFC_INT_
+94	DOM (*1)	Monitor output buffer	0	FFFFH	s	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) (g).

Table (1)-23 Tag storage table (PVAL)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	23	23	—	23	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0400H	WORD	User	—	—
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	Public
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	FPNO	Faceplate display pattern	1	50	—	1	INT	User	—	—
+10	PV	Motor value opening	0	100	%	0.0	REAL	System	1	—
+12	VOUT (*1)	Command signal output status	0	FFFFH	—	0	WORD	System	—	—
+14	SVC	Setting value of valve opening (current)	0	100	%	0.0	REAL	System	1	—
+16	DV	Deviation of valve opening	-100	100	%	0.0	REAL	System	1	—
+18	HS0	Hysterisis	0	100	%	0.0	REAL	User	1	—
+20	DBND	Dead band	0	100	%	0.0	REAL	User	1	—
+26	PH	PV high limit alarm value	0	100	%	100.0	REAL	User	1	—
+28	PL	PV low limit alarm value	0	100	%	0.0	REAL	User	1	—
+30	HH	PV high high limit alarm value	0	100	%	100.0	REAL	User	1	—
+32	LL	PV low low limit alarm value	0	100	%	0.0	REAL	User	1	—
+34	SH	SV high limit value	0	100	%	100.0	REAL	User	1	—
+36	SL	SV low limit value	0	100	%	0.0	REAL	User	1	—
+38	ALPHA	PV filter coefficient	0	1	—	0.2	REAL	User	2	P_IN
+40	HS	PV high/low limit alarm hysteresis	0	100	%	0.0	REAL	User	1	—
+42	CTIM	Variation rate alarm check time	0	9999	s	0.0	REAL	User	2	—
+44	DPL	Variation rate alarm value	0	100	%	100.0	REAL	User	1	—
+50	DVL	Deviation limit value	0	100	%	100.0	REAL	User	1	—
+83	TOT	Time-out timer	0	99	s	5	INT	User	—	—
+84	DOT	Command pulse period	0	9.0	s	1.0	REAL	User	1	—
+86	SIMT	Simulation answer period	0	99	s	3	INT	User	—	—
+87	ALM2 (*1)	Alarm 2	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+88	INH2 (*1)	Disable alarm detection 2	0	FFFFH	—	0000H	WORD	User	—	Public
+89	ALML2 (*1)	Alarm level 2	0	FFFFH	—	0000H	WORD	User	—	Public
+90	SV	Setting value of valve opening (target)	0	100	%	0.0	REAL	User	1	—
+92	DSVL	SV variation rate high limit value	0	100	%	100.0	REAL	User	1	—
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	Tag data access control	—	—
+95	DIM (*1)	Monitor input buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) (h).

Table (1)-24 Tag storage table (HTCL)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+0	FUNC	Tag function code	24	24	—	24	INT	System	—	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—	P_MCHG
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0600H	WORD	User	—	—
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—	Public
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—	Public
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—	—
+8	FPNO	Faceplate display pattern	1	50	—	1	INT	User	—	—
+9	PRM_TRK	Tracking of PID parameters	0	1	—	0	INT	User	—	—
+10	MV_HT	Heating manipulated variable	-10	110	%	0.0	REAL	User	1	—
+12	MV_CL	Cooling manipulated variable	-10	110	%	0.0	REAL	User	1	—
+14	SV	Setting value	0	100	%	0.0	REAL	User	1	—
+26	MH_HT	Heating MV high limit value	-10	110	%	100.0	REAL	User	1	—
+28	ML_HT	Heating MV low limit value	-10	110	%	0.0	REAL	User	1	—
+30	MH_CL	Cooling MV high limit value	-10	110	%	100.0	REAL	User	1	—
+32	ML_CL	Cooling MV low limit value	-10	110	%	0.0	REAL	User	1	—
+34	SH	SV high limit value	0	100	%	100.0	REAL	User	1	—
+36	SL	SV low limit value	0	100	%	0.0	REAL	User	1	—
+48	DML_HT	Heating output variation rate high limit value	0	100	%	100.0	REAL	User	1	—
+50	DML_CL	Cooling output variation rate high limit value	0	100	%	100.0	REAL	User	1	—
+52	P_HT	Heating gain	0	999	—	1.0	REAL	User	2	—
+54	I_HT	Heating integral time	0	9999	s	10.0	REAL	User	1	—
+56	D_HT	Heating derivation time	0	9999	s	0.0	REAL	User	1	—
+58	P_CL	Cooling gain	0	999	—	1.0	REAL	User	2	—
+60	I_CL	Cooling integral time	0	9999	s	10.0	REAL	User	1	—
+62	D_CL	Cooling derivation time	0	9999	s	0.0	REAL	User	1	—
+64	DBND	Dead band	-100	100	%	0.0	REAL	User	1	—
+66	HS	Hysterisis	0	50	%	0.0	REAL	User	1	—
+68	SPLT	Split value	0	100	%	50.0	REAL	User	1	—
+86	PRM_SEL	Target to reflect results of auto tuning	0	3	—	0	INT	User	—	—
+94	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	Tag data access control	—	—
+95	DIM (*1)	Monitor input buffer	0	FFFFH	—	0000H	WORD	System	—	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (1) (i).

(2) List of status tag data
List of status tag data is as follows.

Table (2)-1 Memory table (NREV)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point
			Low limit	High limit					
+0	FUNC	Tag function code	128	128	—	128	INT	System	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0620H	WORD	User	—
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—
+7	CTFN	Lockout function	0	0002H	—	0000H	WORD	System	—
+8	FPNO	Faceplate display pattern	1	50	—	1	INT	User	—
+9	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	Tag data access control	—
+10	DIM (*1)	Monitor input buffer	0	FFFFH	—	0000H	WORD	System	—
+14	TOT	Time-out timer	0	99	s	5	INT	User	—
+15	DOT	Command pulse period	0	9	s	1	INT	User	—
+16	SIMT	Simulation answer period	0	99	s	3	INT	User	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (2) and Appendix 1.2 (2) (a).

Table (2)-2 Memory table (REV)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point
			Low limit	High limit					
+0	FUNC	Tag function code	129	129	—	129	INT	System	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0620H	WORD	User	—
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—
+7	CTFN	Lockout function	0	0002H	—	0000H	WORD	System	—
+8	FPNO	Faceplate display pattern	1	50	—	1	INT	User	—
+9	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	Tag data access control	—
+10	DIM (*1)	Monitor input buffer	0	FFFFH	—	0000H	WORD	System	—
+14	TOT	Time-out timer	0	99	s	5	INT	User	—
+15	DOT	Command pulse period	0	9	s	1	INT	User	—
+16	SIMT	Simulation answer period	0	99	s	3	INT	User	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (2) and Appendix 1.2 (2) (b).

Table (2)-3 Memory table (MVAL1)

Offset	Item	Name	Setting/Storage range Low limit	Setting/Storage range High limit	Unit	Initial value	Data type	Storage	Number of digits after the decimal point
+0	FUNC	Tag function code	130	130	—	130	INT	System	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0620H	WORD	User	—
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—
+8	FPNO	Faceplate display pattern	1	50	—	1	INT	User	—
+9	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	Tag data access control	—
+10	DIM (*1)	Monitor input buffer	0	FFFFH	—	0000H	WORD	System	—
+14	TOT	Time-out timer	0	99	s	5	INT	User	—
+15	DOT	Command pulse period	0	9	s	1	INT	User	—
+16	SIMT	Simulation answer period	0	99	s	3	INT	User	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variable, refer to Appendix 1.2 (2) and Appendix 1.2 (2) (c).

Table (2)-4 Memory table (MVAL2)

Offset	Item	Name	Setting/Storage range Low limit	Setting/Storage range High limit	Unit	Initial value	Data type	Storage	Number of digits after the decimal point
+0	FUNC	Tag function code	131	131	—	131	INT	System	—
+1	MODE (*1)	Control mode	0	FFFFH	—	0008H	WORD	User (condition 1)	—
+2	MDIH (*1)	Disable mode	0	FFFFH	—	0620H	WORD	User	—
+3	ALM (*1)	Alarm	0	FFFFH	—	0000H	WORD	User (condition 2)	—
+4	INH (*1)	Disable alarm detection	0	FFFFH	—	0000H	WORD	User	—
+5	ALML (*1)	Alarm level	0	FFFFH	—	0000H	WORD	User	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—
+7	CTFN	Lockout tag function	0	0002H	—	0000H	WORD	System	—
+8	FPNO	Faceplate display pattern	1	50	—	1	INT	User	—
+9	DOM (*1)	Monitor output buffer	0	FFFFH	—	0000H	WORD	Tag data access control	—
+10	DIM (*1)	Monitor input buffer	0	FFFFH	—	0000H	WORD	System	—
+14	TOT	Time-out timer	0	99	s	5	INT	User	—
+15	DOT	Command pulse period	0	9	s	1	INT	User	—
+16	SIMT	Simulation answer period	0	99	s	3	INT	User	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variable, refer to Appendix 1.2 (2) and Appendix 1.2 (2) (d).

Table (2)-5 Memory table (TIMER1)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point
			Low limit	High limit					
+0	FUNC	Tag function code	132	132	–	132	INT	System	–
+6	CTNO	Lockout tag No.	0	32	–	0	INT	System	–
+7	CTFN	Lockout tag function	0	0002H	–	0000H	WORD	System	–
+9	DOM (*1)	Monitor output buffer	0	FFFFH	–	0000H	WORD	Tag data access control	–
+10	DIM (*1)	Monitor input buffer	0	FFFFH	–	0000H	WORD	System	–
+14	UNIT	Unit	0	127	–	0	INT	User	–
+16	PV	Process variable	RL	RH	UNIT	0	DINT	System	–
+18	PSV	Setting value (preset)	RL	RH	UNIT	0	DINT	User	–
+20	SV	Setting value	RL	RH	UNIT	0	DINT	User	–
+22	MULT	Multiplying factor (0: second, 1: minute)	0	1	–	1	INT	User	–
+24	RH	Timer high limit	0	99999999	UNIT	99999999	DINT	User	–
+26	RL	Timer low limit	0	99999999	UNIT	0	DINT	User	–

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variable, refer to Appendix 1.2 (2) (e).

Table (2)-6 Memory table (TIMER2)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point
			Low low limit	High High limit					
+0	FUNC	Tag function code	133	133	–	133	INT	System	–
+6	CTNO	Lockout tag No.	0	32	–	0	INT	System	–
+7	CTFN	Lockout tag function	0	0002H	–	0000H	WORD	System	–
+9	DOM (*1)	Monitor output buffer	0	FFFFH	–	0000H	WORD	Tag data access control	–
+10	DIM (*1)	Monitor input buffer	0	FFFFH	–	0000H	WORD	System	–
+14	UNIT	Unit	0	127	–	0	INT	User	–
+16	PV	Process variable	RL	RH	UNIT	0	DINT	System	–
+18	PSV	Setting value (preset)	RL	RH	UNIT	0	DINT	User	–
+20	SV	Setting value	RL	RH	UNIT	0	DINT	User	–
+22	MULT	Multiplying factor (0: second, 1: minute)	0	1	–	1	INT	User	–
+24	RH	Timer high limit	0	99999999	UNIT	99999999	DINT	User	–
+26	RL	Timer low limit	0	99999999	UNIT	0	DINT	User	–

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variable, refer to Appendix 1.2 (2) (f).

Table (2)-7 Memory table (COUNT1)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point
			Low limit	High limit					
+0	FUNC	Tag function code	134	134	—	134	INT	System	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—
+7	CTFN	Lockout tag function	0	0002_H	—	0000_H	WORD	System	—
+9	DOM (*1)	Monitor output buffer	0	$FFFF_H$	—	0000_H	WORD	Tag data access control	—
+10	DIM (*1)	Monitor input buffer	0	$FFFF_H$	—	0000_H	WORD	System	—
+14	UNIT	Unit	0	127	—	0	INT	User	—
+16	PV	Process variable	RL	RH	UNIT	0	DINT	System	—
+18	PSV	Setting value (preset)	RL	RH	UNIT	0	DINT	User	—
+20	SV	Setting value	RL	RH	UNIT	0	DINT	User	—
+22	MULT	Multiplying factor	1	999	—	1	INT	User	—
+24	RH	Counter high limit	0	99999999	UNIT	99999999	DINT	User	—
+26	RL	Counter low limit	0	99999999	UNIT	0	DINT	User	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variable, refer to Appendix 1.2 (2) (g).

Table (2)-8 Memory table (COUNT2)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point
			Low limit	High limit					
+0	FUNC	Tag function code	135	135	—	135	INT	System	—
+6	CTNO	Lockout tag No.	0	32	—	0	INT	System	—
+7	CTFN	Lockout tag function	0	0002_H	—	0000_H	WORD	System	—
+9	DOM (*1)	Monitor output buffer	0	$FFFF_H$	—	0000_H	WORD	Tag data access control	—
+10	DIM (*1)	Monitor input buffer	0	$FFFF_H$	—	0000_H	WORD	System	
+14	UNIT	Unit	0	127	—	0	INT	User	
+16	PV	Process variable	RL	RH	UNIT	0	DINT	System	—
+18	PSV	Setting value (preset)	RL	RH	UNIT	0	DINT	User	—
+20	SV	Setting value	RL	RH	UNIT	0	DINT	User	—
+22	MULT	Multiplying factor	1	999	—	1	INT	User	—
+24	RH	Counter high limit	0	99999999	UNIT	99999999	DINT	User	—
+26	RL	Counter low limit	0	99999999	UNIT	0	DINT	User	—

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variable, refer to Appendix 1.2 (2) (h).

Table (2)-9 Memory table (PB)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point
			Low limit	High limit					
+0	FUNC	Tag function code	136	136	–	136	INT	System	–
+1	MODE (*1)	Control mode	0	FFFFH	–	0008H	WORD	User (condition 1)	–
+2	MDIH (*1)	Disable mode	0	FFFFH	–	0000H	WORD	User	–
+6	CTNO	Lockout tag No.	0	32	–	0	INT	System	–
+7	CTFN	Lockout tag function	0	0002H	–	0000H	WORD	System	–
+9	DOM (*1)	Monitor output buffer	0	FFFFH	–	0000H	WORD	Tag data access control	–
+10	DIM (*1)	Monitor input buffer	0	FFFFH	–	0000H	WORD	System	–
+15	DOT	Command pulse period	0	9	s	1	INT	User	–
+17	FPINH (*1)	Disable display	0	FFFFH	–	0000H	WORD	User	–
+18	BTNINH (*1)	Disable control button	0	FFFFH	–	0000H	WORD	User	–
+19	FPNO1	Faceplate display 1 pattern	1	10000	–	1	INT	User	–
+20	FPNO2	Faceplate display 2 pattern	1	10000	–	1	INT	User	–
+21	FPNO3	Faceplate display 3 pattern	1	10000	–	1	INT	User	–
+22	FPNO4	Faceplate display 4 pattern	1	10000	–	1	INT	User	–
+23	FPNO5	Faceplate display 5 pattern	1	10000	–	1	INT	User	–

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variable, refer to Appendix 1.2 (2) (i).

(3) List of alarm tag data
List of alarm tag data is as follows.

Table (3)-1 Memory table (ALM)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Storage	Data type
			Low limit	High limit				
+0	FUNC	Tag function code	256	256	—	256	System	INT
+1	ALM (*1)	Alarm	0	00FFH	—	0000H	System	WORD
+2	ALML (*1)	Alarm level	0	00FFH	—	0000H	User	WORD
+4	ALM1NO	Alarm 1 name No.	0	10000	—	0	User	INT
+5	ALM2NO	Alarm 2 name No.	0	10000	—	0	User	INT
+6	ALM3NO	Alarm 3 name No.	0	10000	—	0	User	INT
+7	ALM4NO	Alarm 4 name No.	0	10000	—	0	User	INT
+8	ALM5NO	Alarm 5 name No.	0	10000	—	0	User	INT
+9	ALM6NO	Alarm 6 name No.	0	10000	—	0	User	INT
+10	ALM7NO	Alarm 7 name No.	0	10000	—	0	User	INT
+11	ALM8NO	Alarm 8 name No.	0	10000	—	0	User	INT

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (3) (a).

Table (3)-2 Memory table (ALM_64PT)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Storage	Data type
			Low limit	High limit				
+0	FUNC	Tag function code	257	257	—	257	System	INT
+1	ALM_W1 (*1)	Alarm 1 to 16	0	FFFFH	—	0000H	System	WORD
+2	ALM_W2 (*1)	Alarm 17 to 32	0	FFFFH	—	0000H	System	WORD
+3	ALM_W3 (*1)	Alarm 33 to 48	0	FFFFH	—	0000H	System	WORD
+4	ALM_W4 (*1)	Alarm 49 to 64	0	FFFFH	—	0000H	System	WORD
+5	ALML_W1 (*1)	Alarm level 1 to 16	0	FFFFH	—	0000H	User	WORD
+6	ALML_W2 (*1)	Alarm level 17 to 32	0	FFFFH	—	0000H	User	WORD
+7	ALML_W3 (*1)	Alarm level 33 to 48	0	FFFFH	—	0000H	User	WORD
+8	ALMl_W4 (*1)	Alarm level 49 to 64	0	FFFFH	—	0000H	User	WORD
+9	ALM1NO	Alarm 1 name No.	0	10000	—	0	User	INT
+10	ALM2NO	Alarm 2 name No.	0	10000	—	0	User	INT
+11	ALM3NO	Alarm 3 name No.	0	10000	—	0	User	INT
+12	ALM4NO	Alarm 4 name No.	0	10000	—	0	User	INT
+13	ALM5NO	Alarm 5 name No.	0	10000	—	0	User	INT
+14	ALM6NO	Alarm 6 name No.	0	10000	—	0	User	INT
+15	ALM7NO	Alarm 7 name No.	0	10000	—	0	User	INT
+16	ALM8NO	Alarm 8 name No.	0	10000	—	0	User	INT
+17	ALM9NO	Alarm 9 name No.	0	10000	—	0	User	INT
+18	ALM10NO	Alarm 10 name No.	0	10000	—	0	User	INT
+19	ALM11NO	Alarm 11 name No.	0	10000	—	0	User	INT
+20	ALM12NO	Alarm 12 name No.	0	10000	—	0	User	INT
+21	ALM13NO	Alarm 13 name No.	0	10000	—	0	User	INT
+22	ALM14NO	Alarm 14 name No.	0	10000	—	0	User	INT
+23	ALM15NO	Alarm 15 name No.	0	10000	—	0	User	INT
+24	ALM16NO	Alarm 16 name No.	0	10000	—	0	User	INT
+25	ALM17NO	Alarm 17 name No.	0	10000	—	0	User	INT
+26	ALM18NO	Alarm 18 name No.	0	10000	—	0	User	INT
+27	ALM19NO	Alarm 19 name No.	0	10000	—	0	User	INT
+28	ALM20NO	Alarm 20 name No.	0	10000	—	0	User	INT
+29	ALM21NO	Alarm 21 name No.	0	10000	—	0	User	INT
+30	ALM22NO	Alarm 22 name No.	0	10000	—	0	User	INT
+31	ALM23NO	Alarm 23 name No.	0	10000	—	0	User	INT
+32	ALM24NO	Alarm 24 name No.	0	10000	—	0	User	INT
+33	ALM25NO	Alarm 25 name No.	0	10000	—	0	User	INT
+34	ALM26NO	Alarm 26 name No.	0	10000	—	0	User	INT
+35	ALM27NO	Alarm 27 name No.	0	10000	—	0	User	INT
+36	ALM28NO	Alarm 28 name No.	0	10000	—	0	User	INT
+37	ALM29NO	Alarm 29 name No.	0	10000	—	0	User	INT
+38	ALM30NO	Alarm 30 name No.	0	10000	—	0	User	INT
+39	ALM31NO	Alarm 31 name No.	0	10000	—	0	User	INT
+40	ALM32NO	Alarm 32 name No.	0	10000	—	0	User	INT
+41	ALM33NO	Alarm 33 name No.	0	10000	—	0	User	INT
+42	ALM34NO	Alarm 34 name No.	0	10000	—	0	User	INT

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Storage	Data type
			Low limit	High limit				
+43	ALM35NO	Alarm 35 name No.	0	10000	—	0	User	INT
+44	ALM36NO	Alarm 36 name No.	0	10000	—	0	User	INT
+45	ALM37NO	Alarm 37 name No.	0	10000	—	0	User	INT
+46	ALM38NO	Alarm 38 name No.	0	10000	—	0	User	INT
+47	ALM39NO	Alarm 39 name No.	0	10000	—	0	User	INT
+48	ALM40NO	Alarm 40 name No.	0	10000	—	0	User	INT
+49	ALM41NO	Alarm 41 name No.	0	10000	—	0	User	INT
+50	ALM42NO	Alarm 42 name No.	0	10000	—	0	User	INT
+51	ALM43NO	Alarm 43 name No.	0	10000	—	0	User	INT
+52	ALM44NO	Alarm 44 name No.	0	10000	—	0	User	INT
+53	ALM45NO	Alarm 45 name No.	0	10000	—	0	User	INT
+54	ALM46NO	Alarm 46 name No.	0	10000	—	0	User	INT
+55	ALM47NO	Alarm 47 name No.	0	10000	—	0	User	INT
+56	ALM48NO	Alarm 48 name No.	0	10000	—	0	User	INT
+57	ALM49NO	Alarm 49 name No.	0	10000	—	0	User	INT
+58	ALM50NO	Alarm 50 name No.	0	10000	—	0	User	INT
+59	ALM51NO	Alarm 51 name No.	0	10000	—	0	User	INT
+60	ALM52NO	Alarm 52 name No.	0	10000	—	0	User	INT
+61	ALM53NO	Alarm 53 name No.	0	10000	—	0	User	INT
+62	ALM54NO	Alarm 54 name No.	0	10000	—	0	User	INT
+63	ALM55NO	Alarm 55 name No.	0	10000	—	0	User	INT
+64	ALM56NO	Alarm 56 name No.	0	10000	—	0	User	INT
+65	ALM57NO	Alarm 57 name No.	0	10000	—	0	User	INT
+66	ALM58NO	Alarm 58 name No.	0	10000	—	0	User	INT
+67	ALM59NO	Alarm 59 name No.	0	10000	—	0	User	INT
+68	ALM60NO	Alarm 60 name No.	0	10000	—	0	User	INT
+69	ALM61NO	Alarm 61 name No.	0	10000	—	0	User	INT
+70	ALM62NO	Alarm 62 name No.	0	10000	—	0	User	INT
+71	ALM63NO	Alarm 63 name No.	0	10000	—	0	User	INT
+72	ALM64NO	Alarm 64 name No.	0	10000	—	0	User	INT
+73	ALMG1NO	Alarm group 1 name No.	0	10000	—	0	User	INT
+74	ALMG2NO	Alarm group 2 name No.	0	10000	—	0	User	INT
+75	ALMG3NO	Alarm group 3 name No.	0	10000	—	0	User	INT
+76	ALMG4NO	Alarm group 4 name No.	0	10000	—	0	User	INT
+77	ALMG5NO	Alarm group 5 name No.	0	10000	—	0	User	INT
+78	ALMG6NO	Alarm group 6 name No.	0	10000	—	0	User	INT
+79	ALMG7NO	Alarm group 7 name No.	0	10000	—	0	User	INT
+80	ALMG8NO	Alarm group 8 name No.	0	10000	—	0	User	INT

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (3) (b).

(4) List of message tag data
List of message tag data is as follows.

Table (4)-1 Memory table (MSG)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Storage	Data type
			Low limit	High limit				
+0	FUNC	Tag function code	272	272	—	272	System	INT
+1	MSG (*1)	Message	0	00FFH	—	0000H	System	WORD
+2	MSGCHK (*1)	Message check	0	00FFH	—	0000H	User	WORD
+4	MSG1NO	Message 1 name No.	0	10000	—	0	User	INT
+5	MSG2NO	Message 2 name No.	0	10000	—	0	User	INT
+6	MSG3NO	Message 3 name No.	0	10000	—	0	User	INT
+7	MSG4NO	Message 4 name No.	0	10000	—	0	User	INT
+8	MSG5NO	Message 5 name No.	0	10000	—	0	User	INT
+9	MSG6NO	Message 6 name No.	0	10000	—	0	User	INT
+10	MSG7NO	Message 7 name No.	0	10000	—	0	User	INT
+11	MSG8NO	Message 8 name No.	0	10000	—	0	User	INT

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (4) (a).

Table (4)-2 Memory table (MSG_64PT)

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Storage	Data type
			Low limit	High limit				
+0	FUNC	Tag function code	273	273	—	273	System	INT
+1	MSG_W1 (*1)	Message 1 to 16	0	FFFFH	—	0000H	System	WORD
+2	MSG_W2 (*1)	Message 17 to 32	0	FFFFH	—	0000H	System	WORD
+3	MSG_W3 (*1)	Message 33 to 48	0	FFFFH	—	0000H	System	WORD
+4	MSG_W4 (*1)	Message 49 to 64	0	FFFFH	—	0000H	System	WORD
+5	MSGCHK_W1 (*1)	Message check 1 to 16	0	FFFFH	—	0000H	User	WORD
+6	MSGCHK_W2 (*1)	Message check 17 to 32	0	FFFFH	—	0000H	User	WORD
+7	MSGCHK_W3 (*1)	Message check 33 to 48	0	FFFFH	—	0000H	User	WORD
+8	MSGCHK_W4 (*1)	Message check 49 to 64	0	FFFFH	—	0000H	User	WORD
+9	MSG1NO	Message 1 name No.	0	10000	—	0	User	INT
+10	MSG2NO	Message 2 name No.	0	10000	—	0	User	INT
+11	MSG3NO	Message 3 name No.	0	10000	—	0	User	INT
+12	MSG4NO	Message 4 name No.	0	10000	—	0	User	INT
+13	MSG5NO	Message 5 name No.	0	10000	—	0	User	INT
+14	MSG6NO	Message 6 name No.	0	10000	—	0	User	INT
+15	MSG7NO	Message 7 name No.	0	10000	—	0	User	INT
+16	MSG8NO	Message 8 name No.	0	10000	—	0	User	INT
+17	MSG9NO	Message 9 name No.	0	10000	—	0	User	INT
+18	MSG10NO	Message 10 name No.	0	10000	—	0	User	INT
+19	MSG11NO	Message 11 name No.	0	10000	—	0	User	INT
+20	MSG12NO	Message 12 name No.	0	10000	—	0	User	INT
+21	MSG13NO	Message 13 name No.	0	10000	—	0	User	INT
+22	MSG14NO	Message 14 name No.	0	10000	—	0	User	INT
+23	MSG15NO	Message 15 name No.	0	10000	—	0	User	INT
+24	MSG16NO	Message 16 name No.	0	10000	—	0	User	INT
+25	MSG17NO	Message 17 name No.	0	10000	—	0	User	INT
+26	MSG18NO	Message 18 name No.	0	10000	—	0	User	INT
+27	MSG19NO	Message 19 name No.	0	10000	—	0	User	INT
+28	MSG20NO	Message 20 name No.	0	10000	—	0	User	INT
+29	MSG21NO	Message 21 name No.	0	10000	—	0	User	INT
+30	MSG22NO	Message 22 name No.	0	10000	—	0	User	INT
+31	MSG23NO	Message 23 name No.	0	10000	—	0	User	INT
+32	MSG24NO	Message 24 name No.	0	10000	—	0	User	INT
+33	MSG25NO	Message 25 name No.	0	10000	—	0	User	INT
+34	MSG26NO	Message 26 name No.	0	10000	—	0	User	INT
+35	MSG27NO	Message 27 name No.	0	10000	—	0	User	INT
+36	MSG28NO	Message 28 name No.	0	10000	—	0	User	INT
+37	MSG29NO	Message 29 name No.	0	10000	—	0	User	INT
+38	MSG30NO	Message 30 name No.	0	10000	—	0	User	INT
+39	MSG31NO	Message 31 name No.	0	10000	—	0	User	INT

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Storage	Data type
			Low limit	High limit				
+40	MSG32NO	Message 32 name No.	0	10000	—	0	User	INT
+41	MSG33NO	Message 33 name No.	0	10000	—	0	User	INT
+42	MSG34NO	Message 34 name No.	0	10000	—	0	User	INT
+43	MSG35NO	Message 35 name No.	0	10000	—	0	User	INT
+44	MSG36NO	Message 36 name No.	0	10000	—	0	User	INT
+45	MSG37NO	Message 37 name No.	0	10000	—	0	User	INT
+46	MSG38NO	Message 38 name No.	0	10000	—	0	User	INT
+47	MSG39NO	Message 39 name No.	0	10000	—	0	User	INT
+48	MSG40NO	Message 40 name No.	0	10000	—	0	User	INT
+49	MSG41NO	Message 41 name No.	0	10000	—	0	User	INT
+50	MSG42NO	Message 42 name No.	0	10000	—	0	User	INT
+51	MSG43NO	Message 43 name No.	0	10000	—	0	User	INT
+52	MSG44NO	Message 44 name No.	0	10000	—	0	User	INT
+53	MSG45NO	Message 45 name No.	0	10000	—	0	User	INT
+54	MSG46NO	Message 46 name No.	0	10000	—	0	User	INT
+55	MSG47NO	Message 47 name No.	0	10000	—	0	User	INT
+56	MSG48NO	Message 48 name No.	0	10000	—	0	User	INT
+57	MSG49NO	Message 49 name No.	0	10000	—	0	User	INT
+58	MSG50NO	Message 50 name No.	0	10000	—	0	User	INT
+59	MSG51NO	Message 51 name No.	0	10000	—	0	User	INT
+60	MSG52NO	Message 52 name No.	0	10000	—	0	User	INT
+61	MSG53NO	Message 53 name No.	0	10000	—	0	User	INT
+62	MSG54NO	Message 54 name No.	0	10000	—	0	User	INT
+63	MSG55NO	Message 55 name No.	0	10000	—	0	User	INT
+64	MSG56NO	Message 56 name No.	0	10000	—	0	User	INT
+65	MSG57NO	Message 57 name No.	0	10000	—	0	User	INT
+66	MSG58NO	Message 58 name No.	0	10000	—	0	User	INT
+67	MSG59NO	Message 59 name No.	0	10000	—	0	User	INT
+68	MSG60NO	Message 60 name No.	0	10000	—	0	User	INT
+69	MSG61NO	Message 61 name No.	0	10000	—	0	User	INT
+70	MSG62NO	Message 62 name No.	0	10000	—	0	User	INT
+71	MSG63NO	Message 63 name No.	0	10000	—	0	User	INT
+72	MSG64NO	Message 64 name No.	0	10000	—	0	User	INT
+73	MSGG1NO	Message group 1 name No.	0	10000	—	0	User	INT
+74	MSGG2NO	Message group 2 name No.	0	10000	—	0	User	INT
+75	MSGG3NO	Message group 3 name No.	0	10000	—	0	User	INT
+76	MSGG4NO	Message group 4 name No.	0	10000	—	0	User	INT
+77	MSGG5NO	Message group 5 name No.	0	10000	—	0	User	INT
+78	MSGG6NO	Message group 6 name No.	0	10000	—	0	User	INT
+79	MSGG7NO	Message group 7 name No.	0	10000	—	0	User	INT
+80	MSGG8NO	Message group 8 name No.	0	10000	—	0	User	INT

*1 This tag data consist of multiple BOOL type variables.
For details of BOOL type variables, refer to Appendix 1.2 (4) (b).

Appendix 1.2 Detailed Information About Tag Data Of Various Tag Types

Marks in the table

User	: can be read/written by user program (Read/Write).
User *1	: write by using P_MCHG of tag access FB.
User *2	: it will not be displayed on FB property window.
System	: only can be read by user program (Read). Please do not execute writing, and the operation during writing is not guaranteed. Meanwhile, the system will not be displayed on FB.
Tag data access control	: Tag data access control can only write from ActiveX Control application program which uses this control. For details about tag data access control, refer to "PX Developer Version 1 Operating Manual (Monitor Tool)".

(1) Loop tag memory

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MODE Control Mode						CSV COMPUTER SV	CMV COMPUTER MV
							User *1*2	User *1*2
							TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
+2	MDIH Disable mode change	SIMI Disable SIMULATION	OVRI Disable OVERRIDE	ATI Disable Auto tuning			CSVI Disable COMPUTER SV	CMVI Disable COMPUTER MV
		User	User	User			User	User
		TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
+3	ALM Alarm		SPA Stop alarm			DMLA Output variation rate limit	OOA Output open	SEA Sensor error
			User *2			System	User	System
			TRUE:Occur FALSE: Reset			TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset

* For +1 and +2 of 2PIDH, refer to (b) in this section.
* For +1, +2, and +3 of SWM, refer to (c) in this section.
* For +1, +2, and +3 of PGS2, refer to (f) in this section.
* For +1, +2, and +3 of PFC_SF, PFC_SS, and PFC_INT, refer to (g) in this section.
* For +1, +2, and +3 of PVAL, refer to (h) in this section.
* For +1, +2, and +3 of HTCL, refer to (i) in this section.

__________ : The underscore indicates the initial value of each tag data. (But when the tag type is MWM or MOUT, the initial value of AUTI and CASI for "Disable mode change (MDIH)" are TRUE.)

	b8	b7	b6	b5	b4	b3	b2	b1	b0
				CAS CASCADE	AUT AUTO	MAN MANUAL			
				User *1*2	User *1*2	User *1*2			
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
				CASI Disable CASCADE	AUTI Disable AUTO	MANI Disable MANUAL			
				User	User	User			
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
	HHA Input high high limit	LLA Input low low limit	PHA Input high limit	PLA Input low limit	DPPA Positive variation rate	DPNA Negative variation rate	DVLA Large deviation	MHA Output high limit	MLA Output low limit
	System	System	System	System	System	System	System	System	System
	TRUE :Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+4	INH Disable Alarm Detection	ERRI Disable all alarms User TRUE: Valid FALSE: Invalid		TRKF Tracking flag System TRUE: Valid FALSE: Invalid		DMLI Disable output variation rate limit alarm User TRUE: Valid FALSE: Invalid		SEI Disable sensor error alarm User TRUE: Valid FALSE: Invalid
+5	ALML Alarm level		SPL Alarm level of stop Alarm User TRUE: Major alarm FALSE: Minor alarm			DMLL Alarm level of output variation rate limit alarm User TRUE: Major alarm FALSE: Minor alarm	OOL Output open level User TRUE: Major alarm FALSE: Minor alarm	SENL Alarm level of sensor error alarm User TRUE: Major alarm FALSE: Minor alarm

* For +4 and +5 of SWM, refer to (c) in this section.
* For +4 and +5 of PGS2, refer to (f) in this section.
* For +4 and +5 of PFC_SF, PFC_SS, and PFC_INT, refer to (g) in this section.
* For +4 and +5 of PVAL, refer to (h) in this section.
* For +4 and +5 of HTCL, refer to (i) in this section.

(a) Loop tag memory (PID, 2PID)*

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+94	DOM Monitor output buffer	SIM Simulation Tag data access control TRUE: Valid FALSE: Invalid	OVR Override Tag data access control TRUE: Valid FALSE: Invalid					
+95	DIM Monitor input buffer							

* The other loop data with SIM and OVR are in the same bit position.
For details about having or not having SIM and OVR, refer to Appendix 1.3 (4).

	b8	b7	b6	b5	b4	b3	b2	b1	b0
	HHI Disable input high high limit alarm	LLI Disable input low low limit alarm	PHI Disable input high limit alarm	PLI Disable input low limit alarm	DPPI Disable positive variation rate alarm	DPNI Disable negative variation rate alarm	DVLI Disable large deviation alarm	MHI Disable output high limit alarm	MLI Disable output low limit alarm
	User	User	User	User	User	User	User	User	User
	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
	HHL Alarm level of input high high limit alarm	LLL Alarm level of input low low limit alarm	PHL Alarm level of input high limit alarm	PLL Alarm level of input low limit alarm	DPPL Alarm level of positive variation rate alarm	DPNL Alarm level of negative variation rate alarm	DVLL Alarm level of large deviation alarm	MHL Alarm level of output high limit alarm	MLL Alarm level of output low limit alarm
	User	User	User	User	User	User	User	User	User
	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm

	b8	b7	b6	b5	b4	b3	b2	b1	b0
								DOM_AT_ STOP_SET Stop set	DOM_AT_ START_SET Start set
								Tag data access control	Tag data access control
								Set	Set
	DIM_AT_ID Identification	DIM_AT_MODE Operation mode	DIM_AT_TO Time-out	DIM_AT_MVL Output low limit	DIM_AT_MVH Output high limit	DIM_AT_PL Input low limit	DIM_AT_PH Input high limit		DIM_AT_RUN Tuning
	System	System	System	System	System	System	System		System
	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset		TRUE: Execute FALSE: Not execute

(b) Loop tag memory (2PIDH)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MODE Control Mode					CASDR CASCADE DIRECT *3 User TRUE: Valid FALSE: Invalid	CSV COMPUTER SV User TRUE: Valid FALSE: Invalid	CMV COMPUTER MV User TRUE: Valid FALSE: Invalid
+2	MDIH Disable mode change	SIMI Disable SIMULATION User TRUE: Valid FALSE: Invalid	OVRI Disable OVERRIDE User TRUE: Valid FALSE: Invalid	ATI Disable Auto tuning User TRUE: Valid FALSE: Invalid	TSTPI Disable Tag stop User TRUE: Valid FALSE: Invalid	CASDRI Disable CADCADE DIRECT User TRUE: Valid FALSE: Invalid	CSVI Disable COMPUTER SV User TRUE: Valid FALSE: Invalid	CMVI Disable COMPUTER MV User TRUE: Valid FALSE: Invalid

*3 In CASCADE DIRECT mode, both values of CAS (b5 of offset +1) and CASDR (b11 of offset +1) are TRUE.

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+87	ALM2 Alarm 2	DSVLA SV variation rate limit System TRUE :Occur FALSE: Reset	SVHA SV high limit System TRUE :Occur FALSE: Reset	SVLA SV low limit System TRUE :Occur FALSE: Reset				
+88	INH2 Disable alarm2 detection	DSVLI SV variation rate limit invalid User TRUE: Valid FALSE: Invalid	SVHI SV high limit invalid User TRUE: Valid FALSE: Invalid	SVLI SV low limit invalid User TRUE: Valid FALSE: Invalid				
+89	ALML2 Alarm level 2	DSVLL SV variation rate limit level User TRUE: Major alarm FALSE: Minor alarm	SVHL SV high limit level User TRUE: Major alarm FALSE: Minor alarm	SVLL SV low limit level User TRUE: Major alarm FALSE: Minor alarm				

	b8	b7	b6	b5	b4	b3	b2	b1	b0
				CAS CASCADE *3	AUT AUTO	MAN MANUAL			
				User	User	User			
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
				CASI Disable CASCADE	AUTI Disable AUTO	MANI Disable MANUAL			
				User	User	User			
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			

	b8	b7	b6	b5	b4	b3	b2	b1	b0

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+94	DOM Monitor output buffer	SIM Simulation Tag data access control TRUE: Valid FALSE: Invalid	OVR Override Tag data access control TRUE: Valid FALSE: Invalid	TSTP TAG STOP Tag data access control TRUE: Stop FALSE: Execute				
+95	DIM Monitor input buffer					DIM_MVTRK MV tracking System TRUE: Execute FALSE: Not execute	DIM_MVHLD MV hold System TRUE: Execute FALSE: Not execute	DIM_PREMV Preset MV System TRUE: Execute FALSE: Not execute

	b8	b7	b6	b5	b4	b3	b2	b1	b0
							DOM_AT_TYPE Command type	DOM_AT_ STOP_SET Stop set	DOM_AT_ START_SET Start set
							Tag data access control	Tag data access control	Tag data access control
							Set only TRUE: Limit Cycle method FALSE: Step Response method	Set	Set
	DIM_AT_ID Identification	DIM_AT_MODE Operation mode	DIM_AT_TO Time-out	DIM_AT_MVL Output low limit	DIM_AT_MVH Output high limit	DIM_AT_PL Input low limit	DIM_AT_PH Input high limit		DIM_AT_RUN Tuning
	System	System	System	System	System	System	System		System
	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset		TRUE: Execute FALSE: Not execute

(c) Loop tag memory (SWM)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MODE Control Mode						CSV COMPUTER SV User *1*2 TRUE: Valid FALSE: Invalid	
+2	MDIH Disable mode change		OVRI Disable OVERRIDE User TRUE: Valid FALSE: Invalid		TSTPI Disable Tag stop User TRUE: Valid FALSE: Invalid		CSVI Disable COMPUTER SV User TRUE: Valid FALSE: Invalid	
+3	ALM Alarm		SPA Stop alarm User *2 TRUE:Occur FALSE: Reset				OOA Output open User TRUE: Occur FALSE: Reset	SEA Sensor error System TRUE: Occur FALSE: Reset
+4	INH Disable Alarm Detection	ERRI Disable all alarms User TRUE: Valid FALSE: Invalid		TRKF Tracking flag System TRUE: Valid FALSE: Invalid				SEI Disable sensor error alarm User TRUE: Valid FALSE: Invalid
+5	ALML Alarm level		SPL Alarm level of stop Alarm User TRUE: Major alarm FALSE: Minor alarm				OOL Output open level User TRUE: Major alarm FALSE: Minor alarm	SENL Alarm level of sensor error alarm User TRUE: Major alarm FALSE: Minor alarm

	b8	b7	b6	b5	b4	b3	b2	b1	b0
				CAS CASCADE3	AUT AUTO	MAN MANUAL			
				User *1*2	User *1*2	User *1*2			
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
				CASI Disable CASCADE	AUTI Disable AUTO	MANI Disable MANUAL			
				User	User	User			
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
	HHA Input high high limit	LLA Input low low limit	PHA Input high limit	PLA Input low limit	DPPA Positive variation rate	DPNA Negative variation rate	DVLA Large deviation		
	System	System	System	System	System	System	System		
	TRUE :Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset		
	HHI Disable input high high limit alarm	LLI Disable input low low limit alarm	PHI Disable input high limit alarm	PLI Disable input low limit alarm	DPPI Disable positive variation rate alarm	DPNI Disable negative variation rate alarm	DVLI Disable large deviation alarm		
	User	User	User	User	User	User	User		
	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid		
	HHL Alarm level of input high high limit alarm	LLL Alarm level of input low low limit alarm	PHL Alarm level of input high limit alarm	PLL Alarm level of input low limit alarm	DPPL Alarm level of positive variation rate alarm	DPNL Alarm level of negative variation rate alarm	DVLL Alarm level of large deviation alarm		
	User	User	User	User	User	User	User		
	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm		

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+87	ALM2 Alarm 2	DSVLA SV variation rate limit System TRUE :Occur FALSE: Reset	SVHA SV high limit System TRUE :Occur FALSE: Reset	SVLA SV low limit System TRUE :Occur FALSE: Reset				
+88	INH2 Disable alarm2 detection	DSVLI SV variation rate limit invalid User TRUE: Valid FALSE: Invalid	SVHI SV high limit invalid User TRUE: Valid FALSE: Invalid	SVLI SV low limit invalid User TRUE: Valid FALSE: Invalid				
+89	ALML2 Alarm level 2	DSVLL SV variation rate limit level User TRUE: Major alarm FALSE: Minor alarm	SVHL SV high limit level User TRUE: Major alarm FALSE: Minor alarm	SVLL SV low limit level User TRUE: Major alarm FALSE: Minor alarm				

	b8	b7	b6	b5	b4	b3	b2	b1	b0

(d) Loop tag memory (BC)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+94	DOM Monitor output buffer							
+95	DIM Monitor input buffer							

(e) Loop tag memory (PSUM)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+94	DOM Monitor output buffer							
+95	DIM Monitor input buffer							

	b8	b7	b6	b5	b4	b3	b2	b1	b0
						DOM_RESET_ START_SET Reset/start by PC	DOM_STOP_ RESET_SET Stop/reset by PC	DOM_HOLD_ SET Hold by PC	DOM_RUN_ SET Run by PC
						Tag data access control	Tag data access control	Tag data access control	Tag data access control
						Set	Set	Set	Set
		DIM_COMP Complete	DIM_PRE_ COMP2 Pre-complete 2	DIM_PRE_ COMP1 Pre-complete 1			DIM_STOP_ RESET Stop reset	DIM_HOLD Hold	DIM_RUN Run
		System	System	System			System	System	System
		TRUE: Completed <u>FALSE: Uncompleted</u>	TRUE: Completed <u>FALSE: Uncompleted</u>	TRUE: Completed <u>FALSE: Uncompleted</u>			TRUE: Execute <u>FALSE: Not execute</u>	TRUE: Execute <u>FALSE: Not execute</u>	TRUE: Execute <u>FALSE: Not execute</u>

	b8	b7	b6	b5	b4	b3	b2	b1	b0
						DOM_RESET_ START_SET Reset/start by PC	DOM_STOP_ RESET_SET Stop/reset by PC	COM_HOLD_ SET Hold by PC	DOM_RUN_ SET Run by PC
						Tag data access control	Tag data access control	Tag data access control	Tag data access control
						Set	Set	Set	Set
							DIM_STOP_ RESET Stop reset	DIM_HOLD Hold	DIM_RUN Run
							System	System	System
							TRUE: Execute <u>FALSE: Not execute</u>	TRUE: Execute <u>FALSE: Not execute</u>	TRUE: Execute <u>FALSE: Not execute</u>

(f) Loop tag memory (PGS2)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MODE Control mode							
+2	MDIH Disable mode				TSTPI Disable TAG STOP User TRUE: Valid FALSE: Invalid			
+3	ALM Alarm		SPA Stop alarm User *2 TRUE: Occur FALSE: Reset					
+4	INH Disable alarm detection	ERRI Disable all alarms User TRUE: Valid FALSE: Invalid						
+5	ALML Alarm level		SPL Stop alarm level User TRUE: Major alarm FALSE: Minor alarm					

	b8	b7	b6	b5	b4	b3	b2	b1	b0
					AUT AUTO User *1 *2 TRUE: Valid FALSE: Invalid	MAN MANUAL User *1 *2 TRUE: Valid FALSE: Invalid			
					AUTI Disable AUTO User TRUE: Valid FALSE: Invalid	MANI Disable MANUAL User TRUE: Valid FALSE: Invalid			
								SVHA SV high limit System TRUE: Occur FALSE: Reset	SVLA SV low limit System TRUE: Occur FALSE: Reset
								SVHI Disable SV high limit User TRUE: Valid FALSE: Invalid	SVLI Disable SV low limit User TRUE: Valid FALSE: Invalid
								SVHL SV high limit level User TRUE: Major alarm FALSE: Minor alarm	SVLL SV low limit level User TRUE: Major alarm FALSE: Minor alarm

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+20	TYP Operation type							
+94	DOM Monitor output buffer			TSTP TAG STOP Tag data access control TRUE: Stop FALSE: Execute				
+95	DIM Monitor input buffer							

	b8	b7	b6	b5	b4	b3	b2	b1	b0
	TUNIT Unit of time						TYP_CYCLIC CYCLIC	TYP_RETURN RETURN	TYP_HOLD HOLD
	User						User	User	User
	TRUE: Minute FALSE: Second						TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
									DOM_ADV_START Advance command
									Tag data access control
									Set
									DIM_WAIT_MODE Waiting
									System
									TRUE: Waiting FALSE: Operating

(g) Loop tag memory (PFC_SF, PFC_SS, PFC_INT)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MODE Control Mode						CSV COMPUTER SV User *1*2 TRUE: Valid FALSE: Invalid	CMV COMPUTER MV User *1*2 TRUE: Valid FALSE: Invalid
+2	MDIH Disable mode change	SIMI Disable SIMULATION User TRUE: Valid FALSE: Invalid	OVRI Disable OVERRIDE User TRUE: Valid FALSE: Invalid		TSTPI Disable TAG STOP User TRUE: Valid FALSE: Invalid		CSVI Disable COMPUTER SV User TRUE: Valid FALSE: Invalid	CMVI Disable COMPUTER MV User TRUE: Valid FALSE: Invalid
+3	ALM Alarm		SPA Stop alarm User *2 TRUE:Occur FALSE: Reset			DMLA Output variation rate limit System TRUE: Occur FALSE: Reset	OOA Output open User TRUE: Occur FALSE: Reset	SEA Sensor error System TRUE: Occur FALSE: Reset
+4	INH Disable Alarm Detection	ERRI Disable all alarms User TRUE: Valid FALSE: Invalid				DMLI Disable output variation rate limit alarm User TRUE: Valid FALSE: Invalid		SEI Disable sensor error alarm User TRUE: Valid FALSE: Invalid
+5	ALML Alarm level		SPL Alarm level of stop Alarm User TRUE: Major alarm FALSE: Minor alarm			DMLL Alarm level of output variation rate limit alarm User TRUE: Major alarm FALSE: Minor alarm	OOL Output open level User TRUE: Major alarm FALSE: Minor alarm	SENL Alarm level of sensor error alarm User TRUE: Major alarm FALSE: Minor alarm

	b8	b7	b6	b5	b4	b3	b2	b1	b0
				CAS CASCADE	AUT AUTO	MAN MANUAL			
				User *1*2	User *1*2	User *1*2			
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
				CASI Disable CASCADE	AUTI Disable AUTO	MANI Disable MANUAL			
				User	User	User			
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
	HHA Input high high limit	LLA Input low low limit	PHA Input high limit	PLA Input low limit	DPPA Positive variation rate	DPNA Negative variation rate	DVLA Large deviation	MHA Output high limit	MLA Output low limit
	System	System	System	System	System	System	System	System	System
	TRUE :Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset
	HHI Disable input high high limit alarm	LLI Disable input low low limit alarm	PHI Disable input high limit alarm	PLI Disable input low limit alarm	DPPI Disable positive variation rate alarm	DPNI Disable negative variation rate alarm	DVLI Disable large deviation alarm	MHI Disable output high limit alarm	MLI Disable output low limit alarm
	User	User	User	User	User	User	User	User	User
	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
	HHL Alarm level of input high high limit alarm	LLL Alarm level of input low low limit alarm	PHL Alarm level of input high limit alarm	PLL Alarm level of input low limit alarm	DPPL Alarm level of positive variation rate alarm	DPNL Alarm level of negative variation rate alarm	DVLL Alarm level of large deviation alarm	MHL Alarm level of output high limit alarm	MLL Alarm level of output low limit alarm
	User	User	User	User	User	User	User	User	User
	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+87	ALM2 Alarm 2	DSVLA SV variation rate limit System TRUE :Occur FALSE: Reset	SVHA SV high limit System TRUE :Occur FALSE: Reset	SVLA SV low limit System TRUE :Occur FALSE: Reset				
+88	INH2 Disable alarm2 detection	DSVLI SV variation rate limit invalid User TRUE: Valid FALSE: Invalid	SVHI SV high limit invalid User TRUE: Valid FALSE: Invalid	SVLI SV low limit invalid User TRUE: Valid FALSE: Invalid				
+89	ALML2 Alarm level 2	DSVLL SV variation rate limit level User TRUE: Major alarm FALSE: Minor alarm	SVHL SV high limit level User TRUE: Major alarm FALSE: Minor alarm	SVLL SV low limit level User TRUE: Major alarm FALSE: Minor alarm				

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+94	DOM Monitor output buffer	SIM Simulation Tag data access control TRUE: Valid FALSE: Invalid	OVR Override Tag data access control TRUE: Valid FALSE: Invalid	TSTP TAG STOP Tag data access control TRUE: Stop FALSE: Execute				

	b8	b7	b6	b5	b4	b3	b2	b1	b0

	b8	b7	b6	b5	b4	b3	b2	b1	b0

(h) Loop tag memory (PVAL)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MODE Control Mode						CSV COMPUTER SV User *1*2 TRUE: Valid FALSE: Invalid	
+2	MDIH Disable mode change	SIMI Disable SIMULATION User TRUE: Valid FALSE: Invalid	OVRI Disable OVERRIDE User TRUE: Valid FALSE: Invalid		TSTPI Disable TAG STOP User TRUE: Valid FALSE: Invalid		CSVI Disable COMPUTER SV User TRUE: Valid FALSE: Invalid	
+3	ALM Alarm		SPA Stop alarm User *2 TRUE:Occur FALSE: Reset					SEA Sensor error System TRUE: Occur FALSE: Reset
+4	INH Disable Alarm Detection	ERRI Disable all alarms User TRUE: Valid FALSE: Invalid						SEI Disable sensor error alarm User TRUE: Valid FALSE: Invalid
+5	ALML Alarm level		SPL Alarm level of stop Alarm User TRUE: Major alarm FALSE: Minor alarm					SENL Alarm level of sensor error alarm User TRUE: Major alarm FALSE: Minor alarm

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+12	VOUT Command signal output status							

b8	b7	b6	b5	b4	b3	b2	b1	b0
			CAS CASCADE	AUT AUTO	MAN MANUAL			
			User *1*2	User *1*2	User *1*2			
			TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
			CASI Disable CASCADE	AUTI Disable AUTO	MANI Disable MANUAL			
			User	User	User			
			TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
HHA Input high high limit	LLA Input low low limit	PHA Input high limit	PLA Input low limit	DPPA Positive variation rate	DPNA Negative variation rate	DVLA Large deviation	TRIPA Trip	TOA Time-out
System	System	System	System	System	System	System	System	System
TRUE :Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset
HHI Disable input high high limit alarm	LLI Disable input low low limit alarm	PHI Disable input high limit alarm	PLI Disable input low limit alarm	DPPI Disable positive variation rate alarm	DPNI Disable negative variation rate alarm	DVLI Disable large deviation alarm	TRIPI Disable trip alarm	TOI Disable time-out alarm
User	User	User	User	User	User	User	User	User
TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
HHL Alarm level of input high high limit alarm	LLL Alarm level of input low low limit alarm	PHL Alarm level of input high limit alarm	PLL Alarm level of input low limit alarm	DPPL Alarm level of positive variation rate alarm	DPNL Alarm level of negative variation rate alarm	DVLL Alarm level of large deviation alarm	TRIPL Alarm level of Trip alarm	TOL Alarm level of Time-out alarm
User	User	User	User	User	User	User	User	User
TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm

b8	b7	b6	b5	b4	b3	b2	b1	b0
							VOUT_CLOSE Output of close command signal	VOUT_OPEN Output of open command signal
							System	System
							TRUE: Output CLOSE FALSE: —	TRUE: Output OPEN FALSE: —

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+87	ALM2 Alarm 2	DSVLA SV variation rate limit System TRUE :Occur FALSE: Reset	SVHA SV high limit System TRUE :Occur FALSE: Reset	SVLA SV low limit System TRUE :Occur FALSE: Reset				
+88	INH2 Disable alarm detection 2	DSVLI SV variation rate limit invalid User TRUE: Valid FALSE: Invalid	SVHI SV high limit invalid User TRUE: Valid FALSE: Invalid	SVLI SV low limit invalid User TRUE: Valid FALSE: Invalid				
+89	ALML2 Alarm level 2	DSVLL SV variation rate limit level User TRUE: Major alarm FALSE: Minor alarm	SVHL SV high limit level User TRUE: Major alarm FALSE: Minor alarm	SVLL SV low limit level User TRUE: Major alarm FALSE: Minor alarm				

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+94	DOM Monitor output buffer	SIM Simulation Tag data access control TRUE: Valid FALSE: Invalid	OVR Override Tag data access control TRUE: Valid FALSE: Invalid	TSTP TAG STOP Tag data access control TRUE: Stop FALSE: Execute				
+95	DIM Monitor input buffer							

	b8	b7	b6	b5	b4	b3	b2	b1	b0

	b8	b7	b6	b5	b4	b3	b2	b1	b0
				DOM_TO_ RESET Time-out reset Tag data access control Set			DOM_STOP_ SET Stop by PC Tag data access control Set	DOM_CLOSE_ SET Close by PC Tag data access control Set	DOM_OPEN_ SET Open by PC Tag data access control Set
					DIM_REMOTE Remote System TRUE: Valid FALSE: Invalid	DIM_LOCAL Local System TRUE: Valid FALSE: Invalid		DIM_CLOSE Status answer System TRUE: Close FALSE: —	DIM_OPEN Status answer System TRUE: Open FALSE: —

(i) Loop tag memory (HTCL)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MODE Control Mode						CSV COMPUTER SV User *1*2 TRUE: Valid FALSE: Invalid	CMV COMPUTER MV User *1*2 TRUE: Valid FALSE: Invalid
+2	MDIH Disable mode change				TSTPI Disable TAG STOP User TRUE: Valid FALSE: Invalid		CSVI Disable COMPUTER SV User TRUE: Valid FALSE: Invalid	CMVI Disable COMPUTER MV User TRUE: Valid FALSE: Invalid
+3	ALM Alarm		SPA Stop alarm User *2 TRUE:Occur FALSE: Reset			HBOA Heater burnout System TRUE: Occur FALSE: Reset	OOA Output open User TRUE: Occur FALSE: Reset	
+4	INH Disable Alarm Detection	ERRI Disable all alarms User TRUE: Valid FALSE: Invalid				HBOI Disable heater burnout alarm User TRUE: Valid FALSE: Invalid		
+5	ALML Alarm level		SPL Alarm level of stop Alarm User TRUE: Major alarm FALSE: Minor alarm			HBOL Alarm level of heater burnout alarm User TRUE: Major alarm FALSE: Minor alarm	OOL Output open level User TRUE: Major alarm FALSE: Minor alarm	

b8	b7	b6	b5	b4	b3	b2	b1	b0
			CAS CASCADE	AUT AUTO	MAN MANUAL			
			User *1*2	User *1*2	User *1*2			
			TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
			CASI Disable CASCADE	AUTI Disable AUTO	MANI Disable MANUAL			
			User	User	User			
			TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
			DMLA_CL Cooling output variation rate limit	MHA_CL Cooling output high limit	MLA_CL Cooling output low limit	DMLA_HT Heating output variation rate limit	MHA_HT Heating output high limit	MLA_HT Heating output low limit
			System	System	System	System	System	System
			TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset
			DMLI_CL Disable cooling output variation rate limit alarm	MHI_CL Disable cooling output high limit alarm	MLI_CL Disable cooling output low limit alarm	DMLI_HT Disable heating output variation rate limit alarm	MHI_HT Disable heating output high limit alarm	MLI_HT Disable heating output low limit alarm
			User	User	User	User	User	User
			TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
			DMLL_CL Alarm level of cooling output variation rate limit alarm	MHL_CL Alarm level of cooling output high limit alarm	MLL_CL Alarm level of cooling output low limit alarm	DMLL_HT Alarm level of heating output variation rate limit alarm	MHL_HT Alarm level of heating output high limit alarm	MLL_HT Alarm level of heating output low limit alarm
			User	User	User	User	User	User
			TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+94	DOM Monitor output buffer			TSTP TAG STOP				
				Tag data access control				
				TRUE: Stop FALSE: Execute				
+95	DIM Monitor input buffer						DIM_PID_CL Cooling PID parameters	DIM_PID_HT Heating PID parameters
							System	System
							TRUE: In use FALSE: —	TRUE: In use FALSE: —

	b8	b7	b6	b5	b4	b3	b2	b1	b0

(2) Status tag memory

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MODE Control Mode							
+2	MDIH Disable Mode	SIMI Disable SIMULATION User TRUE: Valid FALSE: Invalid	OVRI Disable OVERRIDE User TRUE: Valid FALSE: Invalid					
+3	ALM Alarm							
+4	INH Disable Alarm Detection	ERRI Disable all alarms User TRUE: Valid FALSE: Invalid						
+5	ALML Alarm level							

* For +1 and +2 of PB, refer to (c) in this section.

b8	b7	b6	b5	b4	b3	b2	b1	b0
				AUT AUTO	MAN MANUAL			
				User *1	User *1			
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
				AUTI Disable AUTO	MANI Disable MANUAL			
				User	User			
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
							TRIPA Trip	TOA Time-out
							System	System
							TRUE :Occur FALSE: reset	TRUE :Occur FALSE: reset
							TRIPI Disable trip alarm	TOI Disable time- out alarm
							User	User
							TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
							TRIPL Alarm level of Trip alarm	TOL Alarm level of Time-out alarm
							User	User
							TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm

(a) Status tag memory (NREV)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+9	DOM Monitor output buffer	SIM SIMULATION Tag data access control TRUE: Valid FALSE: Invalid	OVR OVERRIDE Tag data access control TRUE: Valid FALSE: Invalid					
+10	DIM Monitor input buffer							

(b) Status tag memory (REV)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+9	DOM Monitor output buffer	SIM SIMULATION Tag data access control TRUE: Valid FALSE: Invalid	OVR OVERRIDE Tag data access control TRUE: Valid FALSE: Invalid					
+10	DIM Monitor input buffer							

	b8	b7	b6	b5	b4	b3	b2	b1	b0
				DOM_TO_ RESET Time-out reset				DOM_STOP_ SET Stop by PC	DOM_RUN_ SET Operation by PC
				Tag data access control				Tag data access control	Tag data access control
				Set				Set	Set
					DIM_ REMOTE Remote	DIM_LOCAL Local		DIM_STOP Status answer	DIM_RUN Status answer
					System	System		System	System
					TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid		TRUE: Stop FALSE: —	TRUE: Run FALSE: —

	b8	b7	b6	b5	b4	b3	b2	b1	b0
				DOM_TO_ RESET Time-out reset			DOM_REV_ SET Reverse run by PC	DOM_STOP_ SET Stop by PC	DOM_FWD_ SET Forward run by PC
				Tag data access control			Tag data access control	Tag data access control	Tag data access control
				Set			Set	Set	Set
					DIM_ REMOTE Remote	DIM_LOCAL Local	DIM_REV Status answer	DIM_STOP Status answer	DIM_FD Status answer
					System	System	System	System	System
					TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Reverse run FALSE: —	TRUE: Stop FALSE: —	TRUE: Forward run FALSE: —

(c) Status tag memory (MVAL1)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+9	DOM Monitor output buffer	SIM SIMULATION Tag data access control TRUE: Valid FALSE: Invalid	OVR OVERRIDE Tag data access control TRUE: Valid FALSE: Invalid					
+10	DIM Monitor input buffer							

(d) Status tag memory (MVAL2)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+9	DOM Monitor output buffer	SIM SIMULATION Tag data access control TRUE: Valid FALSE: Invalid	OVR OVERRIDE Tag data access control TRUE: Valid FALSE: Invalid					
+10	DIM Monitor input buffer							

b8	b7	b6	b5	b4	b3	b2	b1	b0
			DOM_TO_ RESET Time-out reset			DOM_CLOSE_ SET Close by PC		DOM_OPEN_ SET Open by PC
			Tag data access control			Tag data access control		Tag data access control
			Set			Set		Set
				DIM_ REMOTE Remote	DIM_LOCAL Local	DIM_CLOSE Status answer	DIM_SEMI_ CLOSE Status answer	DIM_OPEN Status answer
				System	System	System	System	System
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Close FALSE: —	TRUE: Semi-open FALSE: —	TRUE: Open FALSE: —

b8	b7	b6	b5	b4	b3	b2	b1	b0
			DOM_TO_ RESET Time-out reset			DOM_CLOSE_ SET Close by PC	DOM_STOP_ SET Stop by PC	DOM_OPEN_ SET Open by PC
			Tag data access control			Tag data access control	Tag data access control	Tag data access control
			Set			Set	Set	Set
				DIM_REMOTE Remote	DIM_LOCAL Local	DIM_CLOSE Status answer	DIM_SEMI_ CLOSE Status answer	DIM_OPEN Status answer
				System	System	System	System	System
				TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Close FALSE: —	TRUE: Semi-open FALSE: —	TRUE: Open FALSE: —

(e) Status tag memory (TIMER1)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+9	DOM Monitor output buffer							
+10	DIM Monitor input buffer							

(f) Status tag memory (TIMER2)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+9	DOM Monitor output buffer							
+10	DIM Monitor input buffer							

	b8	b7	b6	b5	b4	b3	b2	b1	b0
						DOM_RESET_ START_SET Reset/start by PC	DOM_RESET_ SET Reset by PC	DOM_STOP_ SET Stop by PC	DOM_RUN_ SET Run by PC
						Tag data access control	Tag data access control	Tag data access control	Tag data access control
						Set	Set	Set	Set
		DIM_COMP Complete external output	DIM_PRE_ COMP Pre-complete external output				DIM_RESET Reset	DIM_STOP Stop	DIM_RUN Run
		System	System				System	System	System
		TRUE: Completed FALSE: Uncompleted	TRUE: Completed FALSE: Uncompleted				TRUE: Execute FALSE: Not execute	TRUE: Execute FALSE: Not execute	TRUE: Execute FALSE: Not execute

	b8	b7	b6	b5	b4	b3	b2	b1	b0
						DOM_RESET_ START_SET Reset/start by PC	DOM_RESET_ SET Reset by PC	DOM_STOP_ SET Stop by PC	DOM_RUN_ SET Run by PC
						Tag data access control	Tag data access control	Tag data access control	Tag data access control
						Set	Set	Set	Set
		DIM_COMP Complete external output	DIM_PRE_ COMP Pre-complete external output				DIM_RESET Reset	DIM_STOP Stop	DIM_RUN Run
		System	System				System	System	System
		TRUE: Completed FALSE: Uncompleted	TRUE: Completed FALSE: Uncompleted				TRUE: Execute FALSE: Not execute	TRUE: Execute FALSE: Not execute	TRUE: Execute FALSE: Not execute

(g) Status tag memory (COUNT1)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+9	DOM Monitor output buffer							
+10	DIM Monitor input buffer							

(h) Status tag memory (COUNT2)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+9	DOM Monitor output buffer							
+10	DIM Monitor input buffer							

	b8	b7	b6	b5	b4	b3	b2	b1	b0
						DOM_RESET_ START_SET Reset/Start by PC	DOM_RESET_ SET Reset by PC	DOM_STOP_ SET Stop by PC	DOM_RUN_ SET Run by PC
						Tag data access control	Tag data access control	Tag data access control	Tag data access control
						Set	Set	Set	Set
		DIM_COMP Complete external output	DIM_PRE_ COMP Pre-complete external output				DIM_RESET Reset	DIM_STOP Stop	DIM_RUN Run
		System	System				System	System	System
		TRUE: Completed FALSE: Uncompleted	TRUE: Completed FALSE: Uncompleted				TRUE: Execute FALSE: Not execute	TRUE: Execute FALSE: Not execute	TRUE: Execute FALSE: Not execute

	b8	b7	b6	b5	b4	b3	b2	b1	b0
						DOM_RESET_ START_SET Reset/Start by PC	DOM_RESET_ SET Reset by PC	DOM_STOP_ SET Stop by PC	DOM_RUN_ SET Run by PC
						Tag data access control	Tag data access control	Tag data access control	Tag data access control
						Set	Set	Set	Set
		DIM_COMP Complete external output	DIM_PRE_ COMP Pre-complete external output				DIM_RESET Reset	DIM_STOP Stop	DIM_RUN Run
		System	System				System	System	System
		TRUE: Completed FALSE: Uncompleted	TRUE: Completed FALSE: Uncompleted				TRUE: Execute FALSE: Not execute	TRUE: Execute FALSE: Not execute	TRUE: Execute FALSE: Not execute

(i) Status tag memory (PB)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MODE Control Mode							
+2	MDIH Disable Mode							

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+9	DOM Monitor output buffer							
+10	DIM Monitor input buffer							

	b8	b7	b6	b5	b4	b3	b2	b1	b0
					AUT AUTO	MAN MANUAL			
					User *1	User *1			
					TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			
					AUTI Disable AUTO	MANI Disable MANUAL			
					User	User			
					TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid			

	b8	b7	b6	b5	b4	b3	b2	b1	b0
					DOM_SET5 Command 5 by PC	DOM_SET4 Command 4 by PC	DOM_SET3 Command 3 by PC	DOM_SET2 Command 2 by PC	DOM_SET1 Command 1 by PC
					Tag data access control	Tag data access control	Tag data access control	Tag data access control	Tag data access control
					Set	Set	Set	Set	Set
					DIM_ON5 Status 5 Answer	DIM_ON4 Status 4 Answer	DIM_ON3 Status 3 Answer	DIM_ON2 Status 2 Answer	DIM_ON1 Status 1 Answer
					System	System	System	System	System
					TRUE: ON FALSE: OFF	TRUE: ON FALSE: OFF	TRUE: ON FALSE: OFF	TRUE: ON FALSE: OFF	TRUE: ON FALSE: OFF

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+17	FPINH Disable Display							
+18	BTNINH Disable Control Button							

	b8	b7	b6	b5	b4	b3	b2	b1	b0
					FPINH5 Disable display of faceplate button 5	FPINH4 Disable display of faceplate button 4	FPINH3 Disable display of faceplate button 3	FPINH2 Disable display of faceplate button 2	FPINH1 Disable display of faceplate button 1
					User	User	User	User	User
					TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
					BTNINH5Disable Control Button 5	BTNINH4Disable Control Button 4	BTNINH3Disable Control Button 3	BTNINH2Disable Control Button 2	BTNINH1Disable Control Button 1
					User	User	User	User	User
					TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid

(3) Alarm tag memory

(a) Alarm tag memory (ALM)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	ALM Alarm							
+2	ALML Alarm level							

(b) Alarm tag memory (ALM_64PT)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	ALM_W1 Alarm 1 to 16	ALM16 Alarm 16 System TRUE : Occur FALSE: Reset	ALM15 Alarm 15 System TRUE : Occur FALSE: Reset	ALM14 Alarm 14 System TRUE : Occur FALSE: Reset	ALM13 Alarm 13 System TRUE : Occur FALSE: Reset	ALM12 Alarm 12 System TRUE : Occur FALSE: Reset	ALM11 Alarm 11 System TRUE : Occur FALSE: Reset	ALM10 Alarm 10 System TRUE : Occur FALSE: Reset
+2	ALM_W2 Alarm 17 to 32	ALM32 Alarm 32 System TRUE : Occur FALSE: Reset	ALM31 Alarm 31 System TRUE : Occur FALSE: Reset	ALM30 Alarm 30 System TRUE : Occur FALSE: Reset	ALM29 Alarm 29 System TRUE : Occur FALSE: Reset	ALM28 Alarm 28 System TRUE : Occur FALSE: Reset	ALM27 Alarm 27 System TRUE : Occur FALSE: Reset	ALM26 Alarm 26 System TRUE : Occur FALSE: Reset

b8	b7	b6	b5	b4	b3	b2	b1	b0
	ALM8 Alarm 8	ALM7 Alarm 7	ALM6 Alarm 6	ALM5 Alarm 5	ALM4 Alarm 4	ALM3 Alarm 3	ALM2 Alarm 2	ALM1 Alarm 1
	System	System	System	System	System	System	System	System
	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset
	ALML8 Alarm 8 Level	ALML7 Alarm 7 Level	ALML6 Alarm 6 Level	ALML5 Alarm 5 Level	ALML4 Alarm 4 Level	ALML3 Alarm 3 Level	ALML2 Alarm 2 Level	ALML1 Alarm 1 Level
	User	User	User	User	User	User	User	User
	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm

b8	b7	b6	b5	b4	b3	b2	b1	b0
ALM9 Alarm 9	ALM8 Alarm 8	ALM7 Alarm 7	ALM6 Alarm 6	ALM5 Alarm 5	ALM4 Alarm 4	ALM3 Alarm 3	ALM2 Alarm 2	ALM1 Alarm 1
System	System	System	System	System	System	System	System	System
TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset
ALM25 Alarm 25	ALM24 Alarm 24	ALM23 Alarm 23	ALM22 Alarm 22	ALM21 Alarm 21	ALM20 Alarm 20	ALM19 Alarm 19	ALM18 Alarm 18	ALM17 Alarm 17
System	System	System	System	System	System	System	System	System
TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+3	ALM_W3 Alarm 33 to 48	ALM48 Alarm 48 System TRUE : Occur FALSE: Reset	ALM47 Alarm 47 System TRUE : Occur FALSE: Reset	ALM46 Alarm 46 System TRUE : Occur FALSE: Reset	ALM45 Alarm 45 System TRUE : Occur FALSE: Reset	ALM44 Alarm 44 System TRUE : Occur FALSE: Reset	ALM43 Alarm 43 System TRUE : Occur FALSE: Reset	ALM42 Alarm 42 System TRUE : Occur FALSE: Reset
+4	ALM_W4 Alarm 49 to 64	ALM64 Alarm 64 System TRUE : Occur FALSE: Reset	ALM63 Alarm 63 System TRUE : Occur FALSE: Reset	ALM62 Alarm 62 System TRUE : Occur FALSE: Reset	ALM61 Alarm 61 System TRUE : Occur FALSE: Reset	ALM60 Alarm 60 System TRUE : Occur FALSE: Reset	ALM59 Alarm 59 System TRUE : Occur FALSE: Reset	ALM58 Alarm 58 System TRUE : Occur FALSE: Reset
+5	ALML_W1 Alarm 1 to 16 level	ALML16 Alarm 16 level User TRUE: Major alarm FALSE: Minor alarm	ALML15 Alarm 15 level User TRUE: Major alarm FALSE: Minor alarm	ALML14 Alarm 14 level User TRUE: Major alarm FALSE: Minor alarm	ALML13 Alarm 13 level User TRUE: Major alarm FALSE: Minor alarm	ALML12 Alarm 12 level User TRUE: Major alarm FALSE: Minor alarm	ALML11 Alarm 11 level User TRUE: Major alarm FALSE: Minor alarm	ALML10 Alarm 10 level User TRUE: Major alarm FALSE: Minor alarm
+6	ALML_W2 Alarm 17 to 32 level	ALML32 Alarm 32 level User TRUE: Major alarm FALSE: Minor alarm	ALML31 Alarm 31 level User TRUE: Major alarm FALSE: Minor alarm	ALML30 Alarm 30 level User TRUE: Major alarm FALSE: Minor alarm	ALML29 Alarm 29 level User TRUE: Major alarm FALSE: Minor alarm	ALML28 Alarm 28 level User TRUE: Major alarm FALSE: Minor alarm	ALML27 Alarm 27 level User TRUE: Major alarm FALSE: Minor alarm	ALML26 Alarm 26 level User TRUE: Major alarm FALSE: Minor alarm
+7	ALML_W3 Alarm 33 to 48 level	ALML48 Alarm 48 level User TRUE: Major alarm FALSE: Minor alarm	ALML47 Alarm 47 level User TRUE: Major alarm FALSE: Minor alarm	ALML46 Alarm 46 level User TRUE: Major alarm FALSE: Minor alarm	ALML45 Alarm 45 level User TRUE: Major alarm FALSE: Minor alarm	ALML44 Alarm 44 level User TRUE: Major alarm FALSE: Minor alarm	ALML43 Alarm 43 level User TRUE: Major alarm FALSE: Minor alarm	ALML42 Alarm 42 level User TRUE: Major alarm FALSE: Minor alarm
+8	ALML_W4 Alarm 49 to 64 level	ALML64 Alarm 64 level User TRUE: Major alarm FALSE: Minor alarm	ALML63 Alarm 63 level User TRUE: Major alarm FALSE: Minor alarm	ALML62 Alarm 62 level User TRUE: Major alarm FALSE: Minor alarm	ALML61 Alarm 61 level User TRUE: Major alarm FALSE: Minor alarm	ALML60 Alarm 60 level User TRUE: Major alarm FALSE: Minor alarm	ALML59 Alarm 59 level User TRUE: Major alarm FALSE: Minor alarm	ALML58 Alarm 58 level User TRUE: Major alarm FALSE: Minor alarm

	b8	b7	b6	b5	b4	b3	b2	b1	b0
	ALM41 Alarm 41	ALM40 Alarm 40	ALM39 Alarm 39	ALM38 Alarm 38	ALM37 Alarm 37	ALM36 Alarm 36	ALM35 Alarm 35	ALM34 Alarm 34	ALM33 Alarm 33
	System	System	System	System	System	System	System	System	System
	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset
	ALM57 Alarm 57	ALM56 Alarm 56	ALM55 Alarm 55	ALM54 Alarm 54	ALM53 Alarm 53	ALM52 Alarm 52	ALM51 Alarm 51	ALM50 Alarm 50	ALM49 Alarm 49
	System	System	System	System	System	System	System	System	System
	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset
	ALML9 Alarm 9 Level	ALML8 Alarm 8 Level	ALML7 Alarm 7 Level	ALML6 Alarm 6 Level	ALML5 Alarm 5 Level	ALML4 Alarm 4 Level	ALML3 Alarm 3 Level	ALML2 Alarm 2 Level	ALML1 Alarm 1 Level
	User	User	User	User	User	User	User	User	User
	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm
	ALML25 Alarm 25 Level	ALML24 Alarm 24 Level	ALML23 Alarm 23 Level	ALML22 Alarm 22 Level	ALML21 Alarm 21 Level	ALML20 Alarm 20 Level	ALML19 Alarm 19 Level	ALML18 Alarm 18 Level	ALML17 Alarm 17 Level
	User	User	User	User	User	User	User	User	User
	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm
	ALML41 Alarm 41 Level	ALML40 Alarm 40 Level	ALML39 Alarm 39 Level	ALML38 Alarm 38 Level	ALML37 Alarm 37 Level	ALML36 Alarm 36 Level	ALML35 Alarm 35 Level	ALML34 Alarm 34 Level	ALML33 Alarm 33 Level
	User	User	User	User	User	User	User	User	User
	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm
	ALML57 Alarm 57 Level	ALML56 Alarm 56 Level	ALML55 Alarm 55 Level	ALML54 Alarm 54 Level	ALML53 Alarm 53 Level	ALML52 Alarm 52 Level	ALML51 Alarm 51 Level	ALML50 Alarm 50 Level	ALML49 Alarm 49 Level
	User	User	User	User	User	User	User	User	User
	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm	TRUE: Major alarm FALSE: Minor alarm

(4) Message tag memory

(a) Message tag memory (MSG)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MSG Message							
+2	MSGCHK Message check							

(b) Message tag memory (MSG_64PT)

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+1	MSG_W1 Message 1 to 16	MSG16 Message 16 / System / TRUE : Occur FALSE: Reset	MSG15 Message 15 / System / TRUE : Occur FALSE: Reset	MSG14 Message 14 / System / TRUE : Occur FALSE: Reset	MSG13 Message 13 / System / TRUE : Occur FALSE: Reset	MSG12 Message 12 / System / TRUE : Occur FALSE: Reset	MSG11 Message 11 / System / TRUE : Occur FALSE: Reset	MSG10 Message 10 / System / TRUE : Occur FALSE: Reset
+2	MSG_W2 Message 17 to 32	MSG32 Message 32 / System / TRUE : Occur FALSE: Reset	MSG31 Message 31 / System / TRUE : Occur FALSE: Reset	MSG30 Message 30 / System / TRUE : Occur FALSE: Reset	MSG29 Message 29 / System / TRUE : Occur FALSE: Reset	MSG28 Message 28 / System / TRUE : Occur FALSE: Reset	MSG27 Message 27 / System / TRUE : Occur FALSE: Reset	MSG26 Message 26 / System / TRUE : Occur FALSE: Reset

b8	b7	b6	b5	b4	b3	b2	b1	b0
	MSG8 Message 8	MSG7 Message 7	MSG6 Message 6	MSG5 Message 5	MSG4 Message 4	MSG3 Message 3	MSG2 Message 2	MSG1 Message 1
	System	System	System	System	System	System	System	System
	TRUE :Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset
	MSGCHK8 Message Check 8	MSGCHK7 Message Check 7	MSGCHK6 Message Check 6	MSGCHK5 Message Check 5	MSGCHK4 Message Check 4	MSGCHK3 Message Check 3	MSGCHK2 Message Check 2	MSGCHK1 Message Check 1
	User	User	User	User	User	User	User	User
	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid

b8	b7	b6	b5	b4	b3	b2	b1	b0
MSG9 Message 9	MSG8 Message 8	MSG7 Message 7	MSG6 Message 6	MSG5 Message 5	MSG4 Message 4	MSG3 Message 3	MSG2 Message 2	MSG1 Message 1
System	System	System	System	System	System	System	System	System
TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset
MSG25 Message 25	MSG24 Message 24	MSG23 Message 23	MSG22 Message 22	MSG21 Message 21	MSG20 Message 20	MSG19 Message 19	MSG18 Message 18	MSG17 Message 17
System	System	System	System	System	System	System	System	System
TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset

Offset	Item	b15	b14	b13	b12	b11	b10	b9
+3	MSG_W3 Message 33 to 48	MSG48 Message 48 System TRUE : Occur FALSE: Reset	MSG47 Message 47 System TRUE : Occur FALSE: Reset	MSG46 Message 46 System TRUE : Occur FALSE: Reset	MSG45 Message 45 System TRUE : Occur FALSE: Reset	MSG44 Message 44 System TRUE : Occur FALSE: Reset	MSG43 Message 43 System TRUE : Occur FALSE: Reset	MSG42 Message 42 System TRUE : Occur FALSE: Reset
+4	MSG_W4 Message 49 to 64	MSG64 Message 64 System TRUE : Occur FALSE: Reset	MSG63 Message 63 System TRUE : Occur FALSE: Reset	MSG62 Message 62 System TRUE : Occur FALSE: Reset	MSG61 Message 61 System TRUE : Occur FALSE: Reset	MSG60 Message 60 System TRUE : Occur FALSE: Reset	MSG59 Message 59 System TRUE : Occur FALSE: Reset	MSG58 Message 58 System TRUE : Occur FALSE: Reset
+5	MSGCHK_W1 Message check 1 to16	MSGCHK16 Message check 16 User TRUE: Valid FALSE: Invalid	MSGCHK15 Message check 15 User TRUE: Valid FALSE: Invalid	MSGCHK14 Message check 14 User TRUE: Valid FALSE: Invalid	MSGCHK13 Message check 13 User TRUE: Valid FALSE: Invalid	MSGCHK12 Message check 12 User TRUE: Valid FALSE: Invalid	MSGCHK11 Message check 11 User TRUE: Valid FALSE: Invalid	MSGCHK10 Message check 10 User TRUE: Valid FALSE: Invalid
+6	MSGCHK_W2 Message check 17 to 32	MSGCHK32 Message check 32 User TRUE: Valid FALSE: Invalid	MSGCHK31 Message check 31 User TRUE: Valid FALSE: Invalid	MSGCHK30 Message check 30 User TRUE: Valid FALSE: Invalid	MSGCHK29 Message check 29 User TRUE: Valid FALSE: Invalid	MSGCHK28 Message check 28 User TRUE: Valid FALSE: Invalid	MSGCHK27 Message check 27 User TRUE: Valid FALSE: Invalid	MSGCHK26 Message check 26 User TRUE: Valid FALSE: Invalid
+7	MSGCHK_W3 Message 33 to 48	MSGCHK48 Message check 48 User TRUE: Valid FALSE: Invalid	MSGCHK47 Message check 47 User TRUE: Valid FALSE: Invalid	MSGCHK46 Message check 46 User TRUE: Valid FALSE: Invalid	MSGCHK45 Message check 45 User TRUE: Valid FALSE: Invalid	MSGCHK44 Message check 44 User TRUE: Valid FALSE: Invalid	MSGCHK43 Message check 43 User TRUE: Valid FALSE: Invalid	MSGCHK42 Message check 42 User TRUE: Valid FALSE: Invalid
+8	MSGCHK_W4 Message check 49 to 64	MSGCHK64 Message check 64 User TRUE: Valid FALSE: Invalid	MSGCHK63 Message check 63 User TRUE: Valid FALSE: Invalid	MSGCHK62 Message check 62 User TRUE: Valid FALSE: Invalid	MSGCHK61 Message check 61 User TRUE: Valid FALSE: Invalid	MSGCHK60 Message check 60 User TRUE: Valid FALSE: Invalid	MSGCHK59 Message check 59 User TRUE: Valid FALSE: Invalid	MSGCHK58 Message check 58 User TRUE: Valid FALSE: Invalid

	b8	b7	b6	b5	b4	b3	b2	b1	b0
	MSG41 Message 41	MSG40 Message 40	MSG39 Message 39	MSG38 Message 38	MSG37 Message 37	MSG36 Message 36	MSG35 Message 35	MSG34 Message 34	MSG33 Message 33
	System	System	System	System	System	System	System	System	System
	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset	TRUE : Occur FALSE: Reset
	MSG57 Message 57	MSG56 Message 56	MSG55 Message 55	MSG54 Message 54	MSG53 Message 53	MSG52 Message 52	MSG51 Message 51	MSG50 Message 50	MSG49 Message 49
	System	System	System	System	System	System	System	System	System
	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE :Occur FALSE: Reset
	MSGCHK9 Message check 9	MSGCHK8 Message Check 8	MSGCHK7 Message Check 7	MSGCHK6 Message Check 6	MSGCHK5 Message Check 5	MSGCHK4 Message Check 4	MSGCHK3 Message Check 3	MSGCHK2 Message Check 2	MSGCHK1 Message Check 1
	User	User	User	User	User	User	User	User	User
	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
	MSGCHK25 Message check 25	MSGCHK24 Message Check 24	MSGCHK23 Message Check 23	MSGCHK22 Message Check 22	MSGCHK21 Message Check 21	MSGCHK20 Message Check 20	MSGCHK19 Message Check 19	MSGCHK18 Message Check 18	MSGCHK17 Message Check 17
	User	User	User	User	User	User	User	User	User
	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
	MSGCHK41 Message check 41	MSGCHK40 Message Check 40	MSGCHK39 Message Check 39	MSGCHK38 Message Check 38	MSGCHK37 Message Check 37	MSGCHK36 Message Check 36	MSGCHK35 Message Check 35	MSGCHK34 Message Check 34	MSGCHK33 Message Check 33
	User	User	User	User	User	User	User	User	User
	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid
	MSGCHK57 Message check 57	MSGCHK56 Message Check 56	MSGCHK55 Message Check 55	MSGCHK54 Message Check 54	MSGCHK53 Message Check 53	MSGCHK52 Message Check 52	MSGCHK51 Message Check 51	MSGCHK50 Message Check 50	MSGCHK49 Message Check 49
	User	User	User	User	User	User	User	User	User
	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid	TRUE: Valid FALSE: Invalid

Appendix 1.3 List of Applicable Tag FB/Tag Access FB/Various Functions in Various Tag Types

(1) Table of corresponding tag type and tag FB.
Tag types of all tag FB are as follows.

Classification	Tag type	Name	Manufacturer tag FB
Loop tag	PID	PID control	M_PID(_T), M_PID_DUTY(_T)
	2PID	2-degree-of-freedom PID control	M_2PID(_T), M_2PID_DUTY(_T)
	2PIDH	2-degree-of-freedom advanced PID control	M_2PIDH(_T)_
	PIDP	Position type PID control	M_PIDP(_T), M_PIDP_EX(_T)_
	SPI	Sample PI control	M_SPI(_T)
	IPD	I-PD control	M_IPD(_T)
	BPI	Blend PI control	M_BPI(_T)
	R	Ratio control	M_R(_T)
	ONF2	2 position ON/OFF control	M_ONF2(_T)
	ONF3	3 position ON/OFF control	M_ONF3(_T)
	PFC_SF	Predictive functional control (simple first order lag)	M_PFC_SF_
	PFC_SS	Predictive functional control (simple second order lag)	M_PFC_SS_
	PFC_INT	Predictive functional control (integral process)	M_PFC_INT_
	PGS	Program setter	M_PGS
	PGS2	Multi-point program setter	M_PGS2_
	MOUT	Manual output	M_MOUT
	MONI	Monitor	M_MONI
	SWM	Manual setter with monitor	M_SWM_
	MWM	Manual output with monitor	M_MWM
	SEL	Loop selector	M_SEL (_T1)(_T2)(_T3_)
	BC	Batch counter	M_BC
	PSUM	Pulse integrator	M_PSUM
	PVAL	Position proportional output	M_PVAL_T_
	HTCL	Heating and cooling output	M_HTCL_T_
Status tag	NREV	Monitor irreversible control	M_NREV
	REV	Monitor reversible control	M_REV
	MVAL1	ON/OFF control 1(without intermediate value)	M_MVAL1
	MVAL2	ON/OFF control 2(with intermediate value)	M_MVAL2
	PB	Push button operation	M_PB_
	TIMER1	Timer 1 (Timer stops when COMPLETE flag is on.)	M_TIMER1
	TIMER2	Timer 2 (Timer continues when COMPLETE flag is on.)	M_TIMER2
	COUNT1	Counter 1 (Counter stops when COMPLETE flag is on.)	M_COUNTER1
	COUNT2	Counter 2 (Counter continues when COMPLETE flag is on.)	M_COUNTER2
Alarm tag	ALM	Alarm	M_ALARM
	ALM_64PT	64-points alarm	M_ALARM_64PT_
Message tag	MSG	Message	M_MESSAGE
	MSG_64PT	64-points message	M_MESSAGE_64PT_

(2) The corresponding table of tag type/tag access FB.
The table below describes the usable tag access FBs in user-defined tag FB.
Note that the tag types not written in the table below cannot create user-defined tag FB.

Classification	Tag type / Tag access FB	PID	2PID	2PIDH	PIDP	SPI	IPD	BPI	R	ONF2	ONF3	PFC_SF	PFC_SS	PFC_INT	PGS	PGS2	MOUT	MONI	SWM	MWM	SEL	BC	PSUM
Input Output Control FB	P_IN	○	○	○	○	○	○	○	○	○	○	○	○	○	−	−	−	○	○	○	−	−	−
	P_OUT1	○	○	−	−	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_OUT2	−	−	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_OUT3_	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_MOUT	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	−	○	−	−	−	−
	P_DUTY	○	○	−	−	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_PSUM	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	○
	P_BC	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	−
	P_MSET_	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	−	−	−
Loop Control Operation FB	P_PID (_T)	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_2PID (_T)	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_2PIDH(_T)_	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_PIDP (_T)	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_PIDP_EX(_T)_	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_SPI (_T)	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_IPD (_T)	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_BPI (_T)	−	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_R (_T)	−	−	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_PHPL	○	○	○	○	○	○	○	○	○	○	○	○	○	−	−	−	○	○	○	−	−	−
	P_ONF2 (_T)	−	−	−	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−
	P_ONF3 (_T)	−	−	−	−	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−
	P_PFC_SF_	−	−	−	−	−	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−
	P_PFC_SS_	−	−	−	−	−	−	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−
	P_PFC_INT_	−	−	−	−	−	−	−	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−
	P_PGS	−	−	−	−	−	−	−	−	−	−	−	−	−	○	−	−	−	−	−	−	−	−
	P_PGS2_	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	−	−	−	−	−	−	−
	P_SEL(_T1) (_T2) (_T3_)	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	−	−
Special FB	P_MCHG	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	−	○	○	○	−	−

○: Usable −: Unusable

(3) Table of corresponding tag type and control mode.
The following table describes the corresponding relation between the tag type and control mode and it lists the tag types whose control mode can be switched. Only control modes of tag type that are listed in the following table can be switched.

Classification	Tag Type	Control mode MAN (MANUAL)	AUT (AUTO)	CAS (CASCADE)	CMV (COMPUTER MV)	CSV (COMPUTER SV)	CASDR (CASCADE DIRECT)
Loop tag	PID	○	○	○	○	○	—
	2PID	○	○	○	○	○	—
	2PIDH	○	○	○	○	○	○
	PIDP	○	○	○	○	○	—
	SPI	○	○	○	○	○	—
	IPD	○	○	○	○	○	—
	BPI	○	○	○	○	○	—
	R	○	○	○	○	○	—
	ONF2	○	○	○	○	○	—
	ONF3	○	○	○	○	○	—
	PFC_SF	○	○	○	○	○	—
	PFC_SS	○	○	○	○	○	—
	PFC_INT	○	○	○	○	○	—
	PGS	○	○	○	○	○	—
	PGS2	○	○	—	—	—	—
	MOUT	○	—	—	○	—	—
	SWM	○	○	○	—	○	—
	MWM	○	—	—	○	—	—
	SEL	○	○	○	○	○	—
	PVAL	○	○	○	—	○	—
	HTCL	○	○	○	○	○	—
Status tag	NREV	○	○	—	—	—	—
	REV	○	○	—	—	—	—
	MVAL1	○	○	—	—	—	—
	MVAL2	○	○	—	—	—	—
	PB	○	○	—	—	—	—

○: Usable —: Unusable

(4) Table of corresponding tag type and I/O mode.
Applicability of I/O mode and auto tuning for each tag type (only for the I/O mode switchable tag type) is shown below.
Only control modes of tag type that are listed in the following table can be switched.

Classification	Tag Type	I/O mode				Auto tuning	
		NOR (NORMAL)	SIM (SIMULATION)	OVR (OVERRIDE)	TSTP (TAG STOP)	AT1 (Step Response method)	AT2 (Limit Cycle method)
Loop tag	PID	○	○	○	—	○	—
	2PID	○	○	○	—	○	—
	2PIDH	○	○	○	○	○	○
	PIDP	○	○	○	—	—	—
	SPI	○	○	○	—	—	—
	IPD	○	○	○	—	—	—
	BPI	○	○	○	—	—	—
	R	○	○	○	—	—	—
	ONF2	○	—	○	—	—	—
	ONF3	○	—	○	—	—	—
	PFC_SF	○	○	○	○	○	—
	PFC_SS	○	○	○	○	○	—
	PFC_INT	○	○	○	○	○	—
	PGS2	○	—	—	○	—	—
	MONI	○	—	○	—	—	—
	SWM	○	—	○	○	—	—
	MWM	○	—	○	—	—	—
	SEL	○	—	—	—	—	—
	PVAL	○	○	○	○	—	—
	HTCL	○	—	—	○	—	—
Status tag	NREV	○	○	○	—	—	—
	REV	○	○	○	—	—	—
	MVAL1	○	○	○	—	—	—
	MVAL2	○	○	○	—	—	—

○: Usable —: Unusable

(5) Table of corresponding tag type and alarm.
Alarm (ALM) detected by tag type is listed in the following table.

Classification	Tag type	Alarm (ALM)																					
		SPA Stop alarm	HBOA Heater burnout	DMLA Output variation rate limit	OOA Output open	SEA Sensor error	HHA Input high high limit	LLA Input low low limit	PHA Input high limit	PLA Input low limit	DPPA Positive variation rate	DPNA Negative variation rate	DVLA Large deviation	MHA Output high limit	MLA Output low limit	DMLA_ HT/CL Heating/ Cooling output variation rate limit	MHA_ HT/CL Heating/ Cooling output high limit	MLA_ HT/CL Heating/ Cooling output low limit	TRIPA Trip	TOA Time-out	SVHA SV high limit	SVLA SV low limit	DSVLA SV variation rate high limit
Loop tag	PID	○	—	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—
	2PID	○	—	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—
	2PIDH	○	—	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	○	○	○
	PIDP	○	—	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—
	SPI	○	—	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—
	IPD	○	—	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—
	BPI	○	—	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—
	R	○	—	○	○	○	○	○	○	○	○	○	—	○	○	—	—	—	—	—	—	—	—
	ONF2	○	—	—	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—	—	—	—
	ONF3	○	—	—	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—	—	—	—
	PFC_ SF	○	—	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	○	○	○
	PFC_ SS	○	—	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	○	○	○
	PFC_ INT	○	—	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	○	○	○
	PGS	○	—	—	—	—	—	—	—	—	—	—	—	○	○	—	—	—	—	—	—	—	—
	PGS2	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	○	—
	MOUT	○	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	MONI	○	—	—	—	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—	—	—	—
	SWM	○	—	—	○	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	○	○	○
	MWM	○	—	—	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—	—	—	—
	SEL	○	—	○	○	—	—	—	—	—	—	—	—	○	○	—	—	—	—	—	—	—	—
	BC	—	—	—	—	—	—	—	○	—	○	—	—	—	—	—	—	—	—	—	—	—	—
	PSUM	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	PVAL	○	—	—	—	○	○	○	○	○	○	○	○	—	—	—	—	—	○	○	○	○	○
	HTCL	○	○	—	○	—	—	—	—	—	—	—	—	—	—	○	○	○	—	—	—	—	—

○: Alarm detection —: No alarm detection

Classification	Tag type	Alarm (ALM)	
		TRIPA Trip	TOA Time-out
Status tag	NREV	○	○
	REV	○	○
	MVAL1	○	○
	MVAL2	○	○
	PB	—	—
	TIMER1	—	—
	TIMER2	—	—
	COUNT1	—	—
	COUNT2	—	—

○: Alarm detection —: No alarm detection

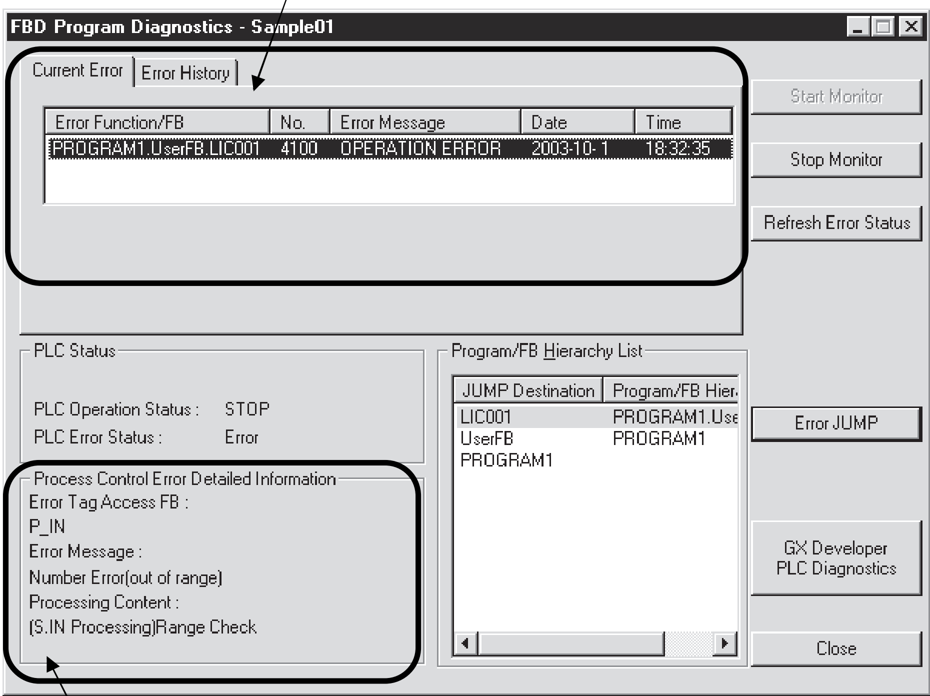
Appendix 2 Error Code List

When executing a function block or an FB, an error in the following table may occur in a CPU module.

Error code	Error contents	Process CPU	Redundant CPU	Universal model process CPU
4100	An operation error has occurred.	○	○	–
	Input data, output data, public variable data, or tag memory data is a denormalized number or a non-numeric value.	○	○	–
	Input data, output data, public variable data, or tag memory data is defective; For example, the value is outside the recommended range, or the low limit exceeds the high limit.	○	○	○
4140	Input data, output data, public variable data, or tag memory data is a denormalized number or a non-numeric value.	–	–	○
4141	An operation error has occurred.	–	–	○

When error occurs on CPU module, error code and contents will be displayed on the diagnostics screen of PX Developer programming tool.
For operation method of FBD program diagnostics dialog box, refer to "PX Developer Version 1 Operating Manual (Programming Tool)".
The contents of FBD program diagnostics dialog box are shown as below.

Error code display • • • refer to (1) in this section



Detailed information display in OPERATION ERROR occurrence • • • refer to (2) in this section

(1) Display error codes
Display error code and error information when error occurs on tag access FB/tag FB.
(Example)
OPERATION ERROR (error occurs during operation)   Error code: 4100

For details of error codes, refer to "QCPU User's Manual (Hardware Design and Maintenance and Inspection)".

(2) Display detailed information when OPERATION ERROR occurs

(a) Error tag access FB and processing contents

When OPERATION ERROR occurs, display error contents of "Error tag access FB/processing No. of tag FB" that is stored in SD1503 of error tag access FB and processing contents part.

Tag access FB/processing No. of tag FB that is stored in SD1503 are listed in the following table.

Tag access FB	CPU module Process control instruction	Process No.							
		1	2	3	4	5	6	7	8
P_IN	S.IN	Range Check	Input limiter	Inverse engineering value conversion	Digital filter	—	—	—	—
P_OUT1	S.OUT1	Input addition processing	Variation rate and high/low limiter	Reset windup	Output conversion	—	—	—	—
P_OUT2	S.OUT2	—	Variation rate and high/low limiter	—	Output conversion	—	—	—	—
P_R(_T)	S.R	Control cycle judgment	Engineering value conversion	Tracking processing	Variation rate limiter	Ratio operation	—	—	—
P_PID(_T)	S.PID	Control cycle judgment	SV setting processing	Tracking processing	Gain Kp operation	PID operation	Deviation check	—	—
	S.AT1	Input check	Time-out judgment	Time-out judgment after maximum slope	Step manipulated variable setting	Sampling interval check	Response waveform observation	Identification processing (*4)	PID constant calculation
P_2PID(_T) P_2PIDH(_T)_	S.2PID	Control cycle judgment	SV setting processing	Tracking processing	Gain Kp operation	PID operation (*1)	PID operation (*2)	PID operation (*3)	Deviation check
	S.AT1	Input Check	Time-out judgment	Time-out judgment after maximum slope	Step manipulated variable setting	Sampling period judgment	Response waveform observation	Identification processing (*4)	PID constant operation
P_PIDP(_T) P_PIDP_EX(_T)_	S.PIDP	Control cycle judgment	SV setting processing	Tracking processing	Gain Kp operation	PIDP operation	Deviation check	Variation rate limiter and high/low limiter	Output conversion
P_SPI(_T)	S.SPI	Operation time monitor	SV setting processing	Tracking processing	Gain Kp operation	SPI operation	Deviation check	—	—
P_IPD(_T)	S.IPD	Control cycle judgment	SV setting processing	Tracking processing	Gain Kp operation	IPD operation	Deviation check	—	—
P_BPI(_T)	S.BPI	Control cycle judgment	SV setting processing	Tracking processing	Gain Kp operation	BPI operation	Deviation check	—	—
P_PHPL	S.PHPL	Inverse engineering value conversion	High/ low limit check	Variation rate check	Engineering value conversion	Loop stop	—	—	—
P_ONF2(_T)	S.ONF2	Control cycle judgment	SV setting processing	Tracking processing	MV correction	MV output	2 position ON/OFF control	—	—
P_ONF3(_T)	S.ONF3	Control cycle judgment	SV setting processing	Tracking processing	MV correction	MV output	3 position ON/OFF control	—	—
P_PGS	S.PGS	Operation constant check	SV count value count up	MVPGS operation	Output processing	—	—	—	—
P_SEL (_T1)(_T2)	S.SEL	Inverse engineering value conversion	Input value selection	Inverse engineering value conversion	Variation rate limiter and high/low limiter	Output conversion	Tracking processing	—	—
P_DUTY	S.DUTY	Input addition processing	Variation rate limiter and high/low limiter	Reset windup	Output ON time Conversion	Output conversion	—	—	—
P_BC	S.BC	High limit check	Variation rate Check	Output conversion	—	—	—	—	—
P_PSUM	S.PSUM	Increment input value operation	Integrating operation	Output conversion	—	—	—	—	—

*1 Bn, Cn operation of P_2PID are included in this table.
*2 Dn operation of P_2PID is included in this table.
*3 △MV operation of P_2PID is included in this table.
*4 The identification processing is the processing that solves for process parameter (PID constant) by step response method.

(b) Error message
When OPERATION ERROR occurs, display error contents of "Tag access FB/code of error occurred on tag FB" that is stored in SD1502 of error message part.
The detailed error codes and information that stored in SD1502 are listed in the following table.

Error code	Error contents	Reason
1	A non-numeric or invalid number	Something wrong with the setting data of operation constant, loop tag memory data and execution cycle. (Perform check and correction of setting data)
2	Symbol error (the number is negative)	
3	Number error (out of range)	
4	Integer range has been exceeded.	
5	Tried to divide by 0	
6	An overflow has occurred	

Appendix 3 Related Functions of Process

Process-related functions are described in the following paragraphs.

Appendix 3.1 Auto Tuning

The auto tuning function detects dynamic characteristics of a control target and automatically tunes proportional gain (Kp), integral time (Ti), and derivative time (Td) for PID to suitable value.
The auto tuning function has two methods: the Step Response method and the Limit Cycle method.

	AT1 (Step Response method)	AT2 (Limit Cycle method)
Overview	This method calculates proportional gain (Kp), integral time (Ti), and derivative time (Td) for PID with ZN method (Step Response method by Ziegler-Nichols) and sets their initial values.	This method calculates proportional gain (Kp), integral time (Ti), and derivative time (Td) for PID with oscillation amplitude and oscillation period by repeatedly operating MV at two positions and generating process variable cycle operation.
Applicable control mode	MAN, CMV	AUT, MAN, CAS, CMV, CSV
PID constants calculation specification method	• P control tuning Execute tuning after setting Ti=0 and Td=0. • PI control tuning Execute tuning after setting Ti>0 and Td=0. • PID control tuning Execute tuning after setting Ti>0 and Td>0.	Execute tuning by selecting PI control or PID control using Monitor tool. Tuning of only P control cannot be executed.
Corresponding tag access FB	P_PID(_T), P_PID_DUTY(_T), P_2PID(_T), P_2PID_DUTY(_T), P_2PIDH(T_)_	P_2PIDH(T_)_
Corresponding tag FB	M_PID(_T), M_PID_DUTY(_T), M_2PID(_T), M_2PID_DUTY(_T), M_2PIDH(T_)_	M_2PIDH(T_)_

Appendix 3.1.1 Step Response method

(1) Operation method and processing contents

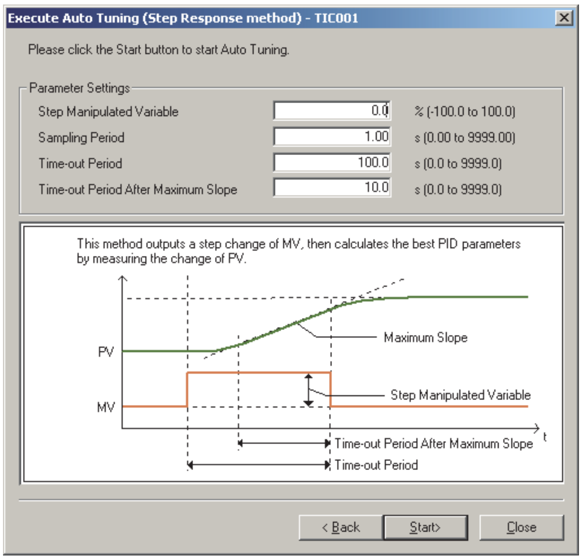
(a) Hold MV value and stabilize PV value, and then perform following procedures.

1) Display auto tuning dialog box of monitor tool.
Please refer to "PX Developer Version 1 Operating Manual (Monitor Tool)" for detailed operation methods of monitor tool.

[Startup Procedure]

[Control Panel] → [Detail button of Faceplate]
→ Auto Tuning → [Select Auto Tuning Operations] dialog box
→ Select [Executes Auto Tuning by the Step Response method].
→ Next button
→ [Execute Auto Tuning (Step Response method)] dialog box

[Execute Auto Tuning (Step Response method) dialog box]



2) Set the following items before starting.
- Step manipulated variable (AT1STEPMV)
- Sampling period (AT1ST) (seconds)
  (PV data collection period during tuning)
- Time-out period (AT1TOUT1) (seconds)
- Time-out period after maximum slope (AT1TOUT2) (seconds)

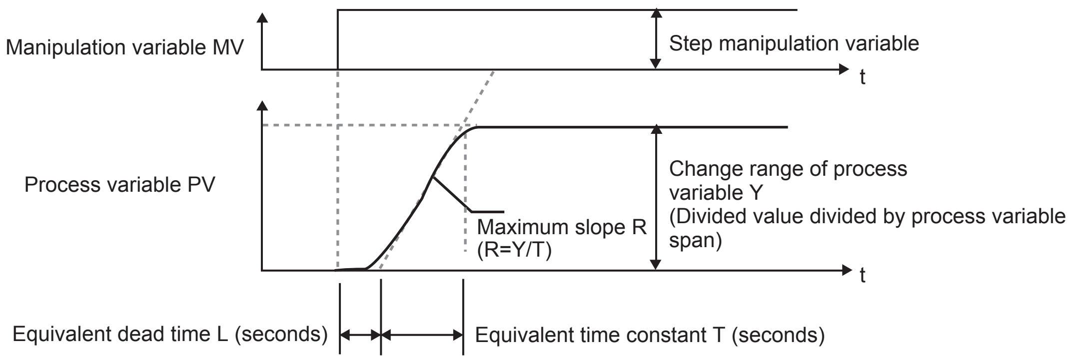
(b) Output the step manipulated variable from the current MV value in step form.

(c) Automatically return to the original MV when auto tuning is completed.
The P, I and D constant that is generated from auto tuning is automatically saved in tag memory.

POINT

- PID constants are overwritten after auto tuning automatically.
  Please save the previous PID constant in advance according to needs.
- Auto tuning will stop automatically when alarm occurs.

(2) The operation contents of step response method
For actual plant, output step MV, and determine the PID constants by maximum slope and equivalent dead time.
Following contents will start automatically after the processing of (1).
The generated value is saved in P, I and D area of tag memory.



(a) ZN method (use Ziegler Nichols's tuning method by step response) The P/PI/PID control type is decided by the I and D value (*1) of tag data before auto tuning.

Condition (*1)		Control type		Constants		
Integral time (I)	Derivative time (D)			Proportional gain (P)	Integral time (I) (seconds)	Derivative time (D) (seconds)
I=0 (Integral time∞)	D=0	P control	→	$\frac{1.0}{R \times L} \times \frac{\lvert\text{step manipulated}\rvert}{100}$	0 (Integral time∞)	0
I>0	D=0	PI control	→	$\frac{0.9}{R \times L} \times \frac{\lvert\text{step manipulated}\rvert}{100}$	3.33L	0
I>0	D>0	PID control	→	$\frac{1.2}{R \times L} \times \frac{\lvert\text{step manipulated}\rvert}{100}$	2L	0.5L

(Example) This is an example that executes auto tuning when I and D value before auto tuning satisfy I>0 and D=0. (For I>0 and D=0, calculate constants in PI control)

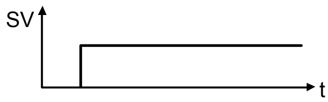
Step manipulated variable	20%
Equivalent dead time L	8 seconds
Equivalent time constant T	16 seconds
The change range of process variable Y	0.25
Maximum slope R	0.25/16=0.016

$$\text{Proportional gain (P)} = \frac{0.9}{R \times L} \times \frac{\lvert\text{step manipulated variable}\rvert}{100} = \frac{0.9}{0.016 \times 8} \times \frac{20}{100} = 1.4$$

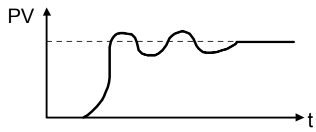
Integral time (I) = 3.33L = 3.33×8 = 26.6 seconds,
derivative time (D) = 0 second

(3) Fine tuning after auto tuning
When auto tuning is completed, observe the change of process variable (PV) in relation to the setting value (SV) by tuning setting execution screen, and trim P, I, D value to work out the optimal value.

Observe the response of PV corresponding to the change of setting value (SV).

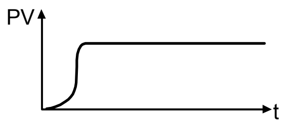


(a) Response is quick, but oscillatory.

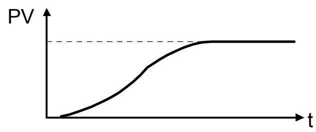


Fine tuning when respond quickly with variation.
- Proportional gain : Smaller (proportional effect become smaller)
- Integral time : Bigger (integral effect become smaller)

(b) Optimal value



(c) Response is slow



Fine tuning for slow response.
- Proportional gain : Bigger (proportional effect become bigger)
- Integral time : Smaller (integral effect become bigger)

In addition, when derivative action is applied, derivative time adjustment shall be executed with confirming stability and respond. (Derivative effect will become bigger when the derivative time is longer, and derivative effect will become smaller when derivative time is smaller.)

Appendix 3.1.2 Limit Cycle Method

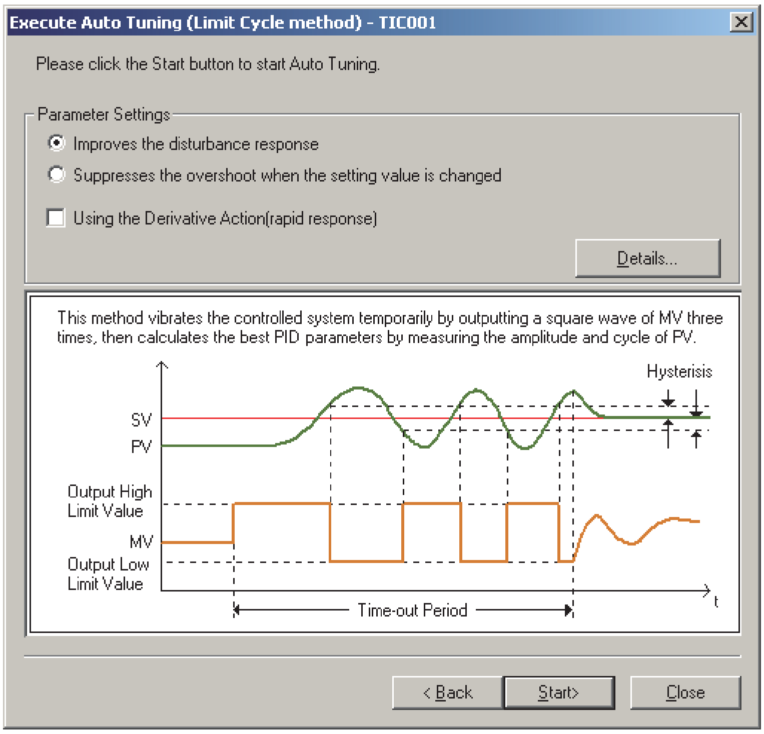
(1) Operation method and processing contents

(a) Operate the following processing.

1) Display the Execute Auto Tuning (Limit Cycle method) dialog box of the Monitor tool.
For details of the Monitor tool operation method, refer to the "PX Developer Version 1 Operating Manual (Monitor tool)".

[Startup Procedure]
[Control Panel] → [Details button of Faceplate]
→ Auto Tuning → [Auto Tuning] dialog box
→ [Select Auto Tuning Operations] dialog box
→ Select [Executes Auto Tuning by the Limit Cycle method].
→ Next button
→ [Execute Auto Tuning (Limit Cycle method)] dialog box

[Execute Auto Tuning (Limit Cycle method)] dialog box



2) Set the following and click the Start button to execute auto tuning.
Decide the control type based on the combination of either "Improves the disturbance response" or "Suppresses the overshoot when the setting value is changed" and the selection status of "Using the Derivative Action (rapid response).

Control Type	Improves the disturbance response	Suppresses the overshoot when the setting value is changed	Using the Derivative Action (rapid response)
Constant-value PI control	○	×	×
Constant-value PID control	○	×	○
Follow-up PI control	×	○	×
Follow-up PID control	×	○	○

○: Selected ×: Not selected

Set the following in the Detail Setting of Limit Cycle dialog box, displayed by clicking the Details button.

- Output high limit (AT2MVH)
- Output low limit (AT2MVL)
- Hysterisis (AT2HS)
- Time-out period (AT1TOUT1) (Seconds)

(b) MV value repeats 2-position output between output high limit and output low limit.
Even though MV values exceeding MH/ML are set to AT2MVH/AT2MVL, they are limited within the range of MH to ML.

(c) MV values return to their original values after auto tuning is completed.
Values for proportional gain (Kp), integral time (Ti), and derivative time (Td) which are calculated by auto tuning are set automatically.

POINT

- PID constants are overwritten automatically after auto tuning.
- Auto tuning stops automatically when an alarm occurs.
- MV output values return to the values at start when auto tuning is completed or interrupted.

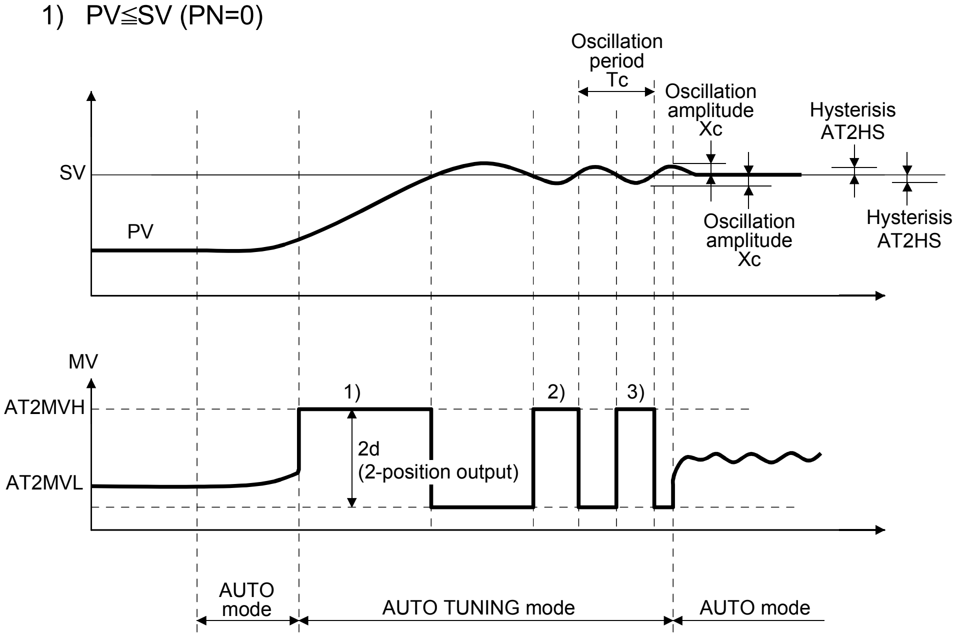
(2) Operation contents of the Limit Cycle method

(a) Generation and measurement of limit cycle waveform
In AUTO TUNING mode, generate PV limit cycle waveform by operating 2-position ON/OFF of MV output three times.
Operate 2-position ON/OFF with conditions shown in the table below.

Control operation	First MV output	2-position ON/OFF operation	Remarks
Reverse action (PN=0)	PV≦SV MV = Output high limit (AT2MVH) PV>SV MV = Output low limit (AT2MVL)	PV≧SV + Hysterisis (AT2HS) MV = Output low limit (AT2MVL) PV≦SV - Hysterisis (AT2HS) MV = Output high limit (AT2MVH)	For operation images, refer to 1) and 2).
Direct action (PN=1)	PV≦SV MV = Output low limit (AT2MVL) PV>SV MV = Output high limit (AT2MVH)	PV≧SV + Hysterisis (AT2HS) MV = Output high limit (AT2MVH) PV≦SV - Hysterisis (AT2HS) MV = Output low limit (AT2MVL)	MV high limit and low limit values at PN=0 are reversely output.

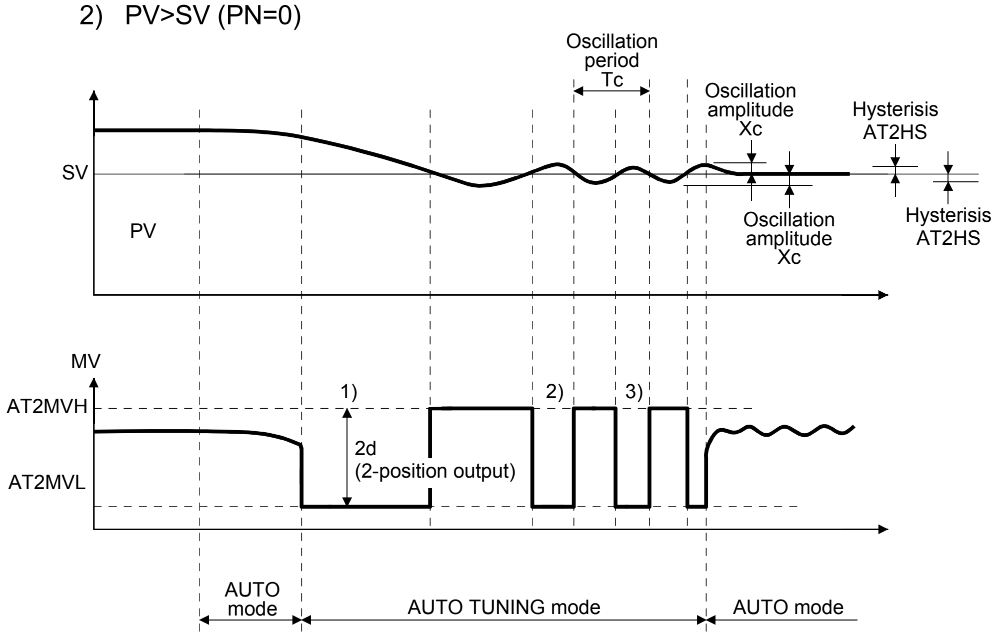
PV oscillation waveform data of the Limit Cycle method waveform by the first 2-position ON/OFF operation is ignored. Oscillation amplitude Xc and oscillation period Tc are measured using PV oscillation waveform data by the second and third 2-position ON/OFF operations.
Auto tuning ends at the apex of the third PV oscillation waveform.
SVC (setting value (current)) at auto tuning start is used to calculate SV value.
Hysterisis (AT2HS), which works as minimum required amplitude, is set in advance according to the control target so that optimum oscillation period and oscillation amplitude are measured.

1) PV≦SV (PN=0)



(2-position operation 1): 1st 2): 2nd 3): 3rd)

2) PV>SV (PN=0)



(2-position operation 1): 1st 2): 2nd 3): 3rd)

Calculate oscillation amplitude Xc by measuring and averaging out plus side and minus side maximum values of |PV - SV|.
Calculate output range d by (AT2MVH - AT2MVL) / 2.

(b) Calculation of threshold sensitivity and threshold period
Calculate threshold sensitivity (Ku) and threshold period (Tu) from measurement result of auto tuning by the Limit Cycle method.

$Ku = 4d / (\pi\sqrt{(Xc^2 - AT2HS^2)})$
$Tu = Tc$

(c) Calculation of optimum PID constant
Calculate optimum PID constant from threshold sensitivity (Ku) and threshold period (Tu).
Calculate values for proportional gain (Kp), integral time (Ti), and derivative time (Td) using coefficients specified by control type (ATTYPE) shown below.

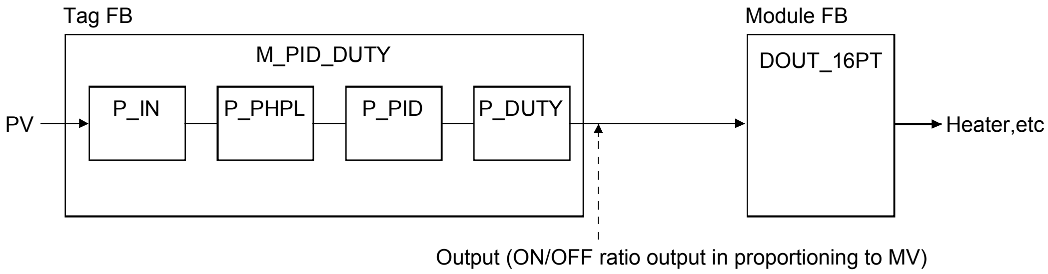
Control Type	Control operation	Control Type ATTYPE	Proportional gain (Kp)	Integral time (Ti)	Derivative time (Td)	Empirical rule
Constant-value control	PI	1	0.45Ku	0.83Tu	0	Ziegler Nichols's method
	PID	2	0.6Ku	0.5Tu	0.125 Tu	
Follow-up control	PI	3	0.3Ku	1.0Tu	0	CHR method
	PID	4	0.45Ku	0.6Tu	0.1 Tu	

Ku: Process threshold sensitivity, Tu: Process threshold period

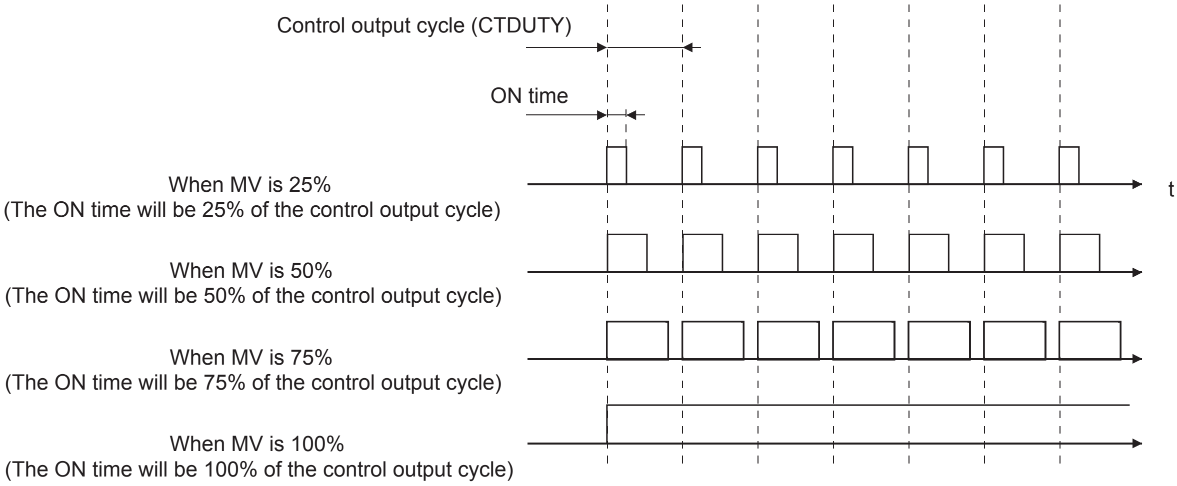
(3) Fine tuning after auto tuning
Fine tuning of PID constants is same for the Step Response method.
Refer to Appendix 3.1.1.

Appendix 3.2 Control Output Cycle (CTDUTY), Manipulated Variable (MV), and ON/OFF Output in Time Proportioning Control

(1) The relation among control output cycle (CTDUTY), manipulated variable (MV) and ON/OFF output
Change the ON/OFF ratio of output in proportion to MV and output the result. It is applicable to temperature control situations that heater is used.
It is applicable in M_□□_DUTY of tag FB and user-defined FB that use P_DUTY of tag access FB.



The relation between MV and output: Output bit ON time of each control output cycle=control output cycle (CTDUTY) × MV (%)/100



(2) Setting of control output cycle (CTDUTY)
Set the control output cycle (seconds) for CTDUTY of tag data on the FB property window of programming tool.

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+68	CTDUTY	Control output cycle	0	9999	s	1.0	REAL	User	2	P_DUTY

Appendix 3.3 I/O Mode

(1) Types of I/O mode
There are 4 types of I/O mode as follows.
For details of simulation function (SIMULATION mode) and override function (OVERRIDE mode), refer to following items
- NORMAL mode
- SIMULATION mode (☞ Appendix 3.14)
- OVERRIDE mode (☞ Appendix 3.15)
- TAG STOP mode (☞ Appendix 3.16)

(2) Operation method
I/O mode is switched with faceplate.
Moreover, it only can be switched when the control mode is manual (MANUAL).
For details of I/O mode change, refer to "PX Developer Operation Manual (Monitor Tool)".

(3) Contents of I/O mode
Contents of I/O mode are as follows.

I/O Mode	Description
NORMAL	Normal mode, the signals from I/O module are connected with the system.
SIMULATION (*1)	The signals from I/O module are separated from the system in this mode, and can perform simulated input/output.
OVERRIDE (*1)	Only signals from input module are separated from the system in this mode, and can input process variable (PV) with faceplate. It is used when error occurs on input sensor. (1) In case of loop tag When it is impossible to attain the proper PV value input signal because of detecting sensor errors, setting of input value can be executed on faceplate. External output is executed. (It is used when interlock condition is to be satisfied or batch sequence transition is to be executed.) (2) In case of status tag When it is impossible to attain the correct input status due to imperfect contact of valve open/close limit switch, etc. the input status can be set though faceplate. External output is executed. (It is used when inter-lock condition is to be satisfied or batch sequence transition is to be executed.)
TAG STOP (*1)	Any processing regarding tag is not performed in this mode. Input processing and loop control operation are stopped. Set this mode to the tags defined in advance for future use or tags being stopped. All alarms of tags are restored and unnecessary alarms do not occur.

*1 Some tags do not have this mode according to their tag types.
For the corresponding relation between tag type and I/O mode, refer to Appendix 1.3 (4).

Appendix 3.4 Execution Cycle (△T) and Control Cycle (CT) in Loop Control

(1) Execution cycle (△T)

(a) Execution cycle (△T)

Programs which consist of loop tag FB, is executed in the cycle which is set to each program.
Loop tag FB of program is executed and operated in each execution cycle.
It is named as execution cycle (△T) in loop control. In tag access FB which forms loop tag FB, P_IN, P_PHPL and P_OUT1 used in I/O control is executed in each execution cycle (△T).
Additionally, loop control operation such as P_PID and P_2PID is executed in each control cycle (CT) as described in (2).

(b) Setting of execution cycle (△T)

Set execution cycle on each program in program execution setting of programming tool.
Please refer to "PX Developer Operation Manual (Programming Tool)".
The execution cycle (△T) can be set within following ranges depending on program execution types.

1) Timer execution type
High-speed (200ms cycle).
Normal speed (400ms, 600ms, 800ms, 1s cycle).
Low-speed (1s, 2s, 4s, 5s, 10s cycle).

2) Interruption execution type
Fixed scan execution (1ms to 999ms)(*1)
Interrupt pointer execution (interruption caused by pointer I0 to I255)(*2)

*1 It is recommended to set the value over 10ms in practical use.
*2 Please do not apply it in the program that uses process-related FB.

POINT

For there is no setting for program execution cycle on interrupt pointer execution of interruption execution type, it is not applicable in program which process-related FB is used in.

(2) Control cycle (CT)

(a) Control cycle (CT)

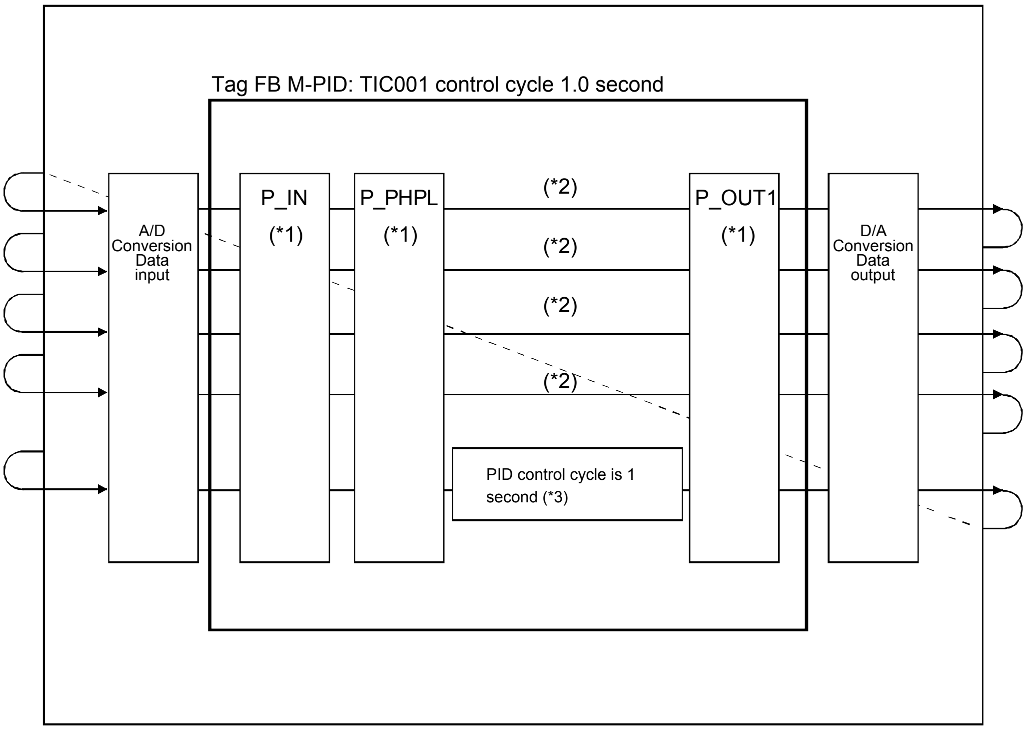
Loop control operation cycle.
The execution cycle (△T) can be set on each program, while the control cycle (CT) can be set on tag.
The control cycle shall be set as the integral multiple of the execution cycle.
The control cycle of the following tag types can be set:
PID, BPI, IPD, ONF2, ONF3, R and 2PID etc.

(b) Setting of control cycle
Set control cycle (seconds) of CT on tag data with FB property window of programming tool.

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+46	CT	Control cycle	0	9999	s	1.00	REAL	User	2	P_PID, etc.

(3) The relation between execution cycle (△T) and control cycle (CT)
The relation between execution cycle and control cycle is as following.
(Example) When the execution cycle of program "ABCD"is 0.2 seconds, and the control cycle of M_PID instruction "TIC001" is 1 second.

Program : when the ABCD execution cycle is 0.2 second.



*1 P_IN, P_PHPL, P_OUT1 is executed in every 0.2 seconds (0.2 seconds: program execution cycle).
*2 When P_PID is not executed, the output value for OUT1 shall be kept the same as the previous value.
*3 As the control cycle of PID control operation instruction is 1.0 second, P_PID is executed in every 1.0 second (please set the control cycle as integral multiple of the execution cycle.)
If the control cycle is not the integral multiple of execution cycle, round off the number after the decimal point of control cycle (CT)/execution cycle (△T) and multiply the execution cycle to calculate the control cycle.
(Example) The execution cycle (△T) has been set to 1.0 second, and the control cycle (CT) to 2.5 seconds.
2.5/1.0=3
The control cycle is 3 seconds.

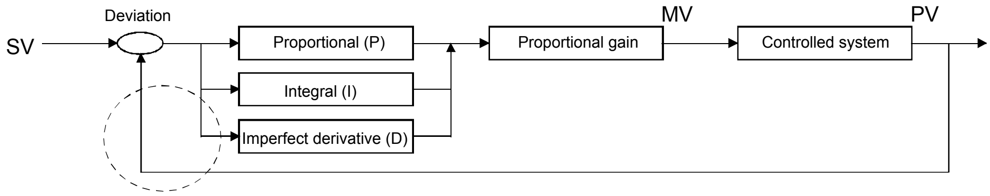
Appendix 3.5 Various PID Control

(1) PV-derivative type PID control (corresponding tag FB: M_PID)

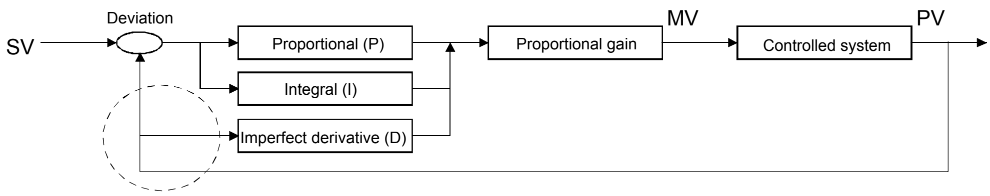
(a) Operation overview

There is a deviation derivative type in PID control that operates based on difference between SV and PV.
In the deviation derivative type, there is a problem which MV is rapidly changed when SV is rapidly changed because the influence of its derivative action is too large.
Therefore it is possible to avoid the influence of the sudden change of setting value by using PV value on the deviation.
M_PID is corresponding to PV-derivative type.
In addition, there are other control methods, in which PV is applied as proportional item and the operation will be more smooth. This method is showed in (2) (I-PD control).

Deviation derivative type



PV-derivative type (M_PID is corresponding to it)

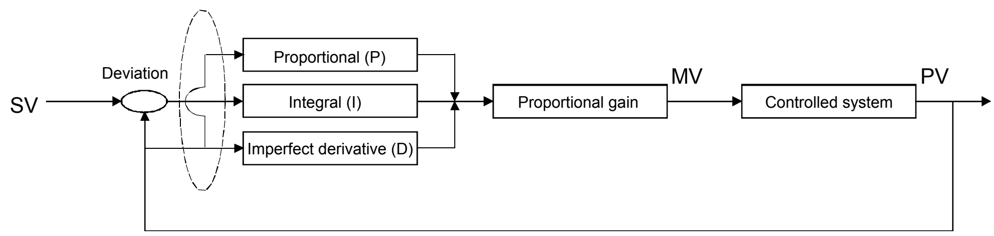


(2) PV-proportional and -derivative type (I-PD control) (corresponding tag FB:M_IPD)

(a) Operation overview

In comparison with PV-derivative type, I-PD type uses PV value on proportional item in addition to derivative item.
This control is also applicable to the situation when the setting value is changed, and rapid change to final control element and system are expected to be avoided, and also slow response is preferred.
M_IPD of tag FB is corresponding to PV-proportional and -derivative type.

PV-proportional and -derivative type CM_IPD is corresponding to it)



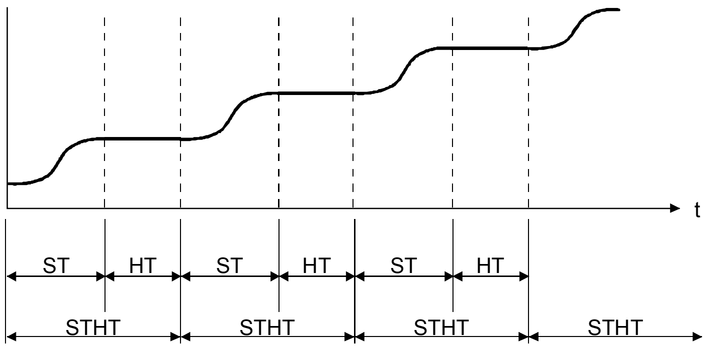
(3) Sample PI (SPI) control (corresponding tag FB: M_SPI)

(a) Operation overview

When PID control is applied on the system whose dead time is long, MV will be continuously updated before MV effect is confirmed.

Sample PI (SPI) control executes only for control cycle in every control cycle, and then holds the output after that.

M_SPI of tag FB is corresponding to sample PI (SPI) control.



ST: Operating time, HT: Hold time=STHT-ST, STHT: Sampling period

(b) Setting of operation time

Set operation time (second) for tag FB ST on FB property window of programming tool.

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+46	ST	Operating time	0	9999	s	0.0	REAL	User	1	P_SPI

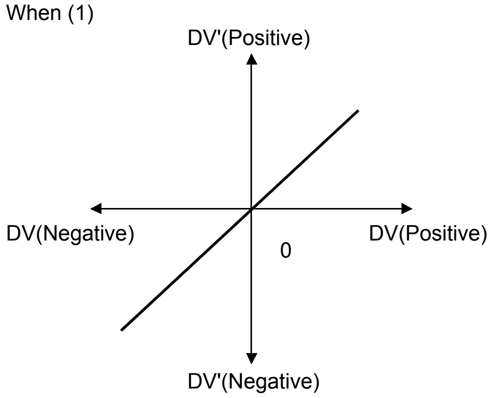
(4) PID control with gap (corresponding tag FB: M_PID, M_IPD, M_SPI, M_BPI, M_2PID, M_2PIDH_, M_PIDP)

(a) Operation overview
For PID control with gap, when deviation is within the gap width (GW) range, the gain shall be changed with gap gain (GG).
Gap width (GW) and gap gain (GG) are set by tag data.
M_PID, M_IPD, M_SPI, M_BPI, M_2PID, M_2PIDH_, M_PIDP of tag data are corresponding to gap PID control.

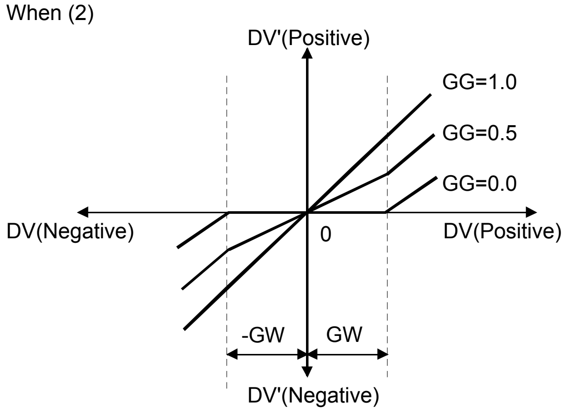
Gain Kp can be calculated as follows.
$Kp=K \times P$
K: Output gain computation

Condition		K: Output gain
(1) K value corresponding to deviation (DV) when gap width (GW)=0		K=1
(2) K value corresponding to deviation (DV) when gap width (GW)>0	$\lvert DV\rvert \leqq GW$	K=GG
	$\lvert DV\rvert > GW$	$K=1-\frac{(1-GG)\times GW}{\lvert DV\rvert}$

When (1)



When (2)



DV<-GW	DV'=-(GG×GW)+(DV+GW)
|DV|≦GW	DV'=GG×DV
DV>GW	DV'=GG×GW+(DV-GW)

Deviation of direct/reverse action is calculated as follows.

Deviation when direct action (PN=1)	DV (%)=PVP (%) – SV (%)
Deviation when reverse action (PN=0)	DV (%)=SV (%) – PVP (%)

K: Output gain, P: Gain, GW: Gap width (%)=Rate of the gap width for the deviation. GG: Gap gain DV': Deviation used for PID operation (%), DV: Deviation (%), PVP(%): PVP input value(%), SV(%)={100/(RH-RL)}×(SV-RL), RH: Engineering value high limit, RL: Engineering value low limit, SV: Setting value.

(b) Setting of gap width and gap gain
On the programming tool's FB property window, set the gap width (%) for GW of tag data, and set gap gain for GG.

Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after the decimal point	Tag access FB
			Low limit	High limit						
+58	GW	Gap width	0	100	%	0.0	REAL	User	1	P_PID, etc.
+60	GG	Gap gain	0	99	—	1.0	REAL	User	1	P_PID, etc.

(5) 2-degree-of-freedom PID control (corresponding tag FB: M_2PID, M_2PIDH_)

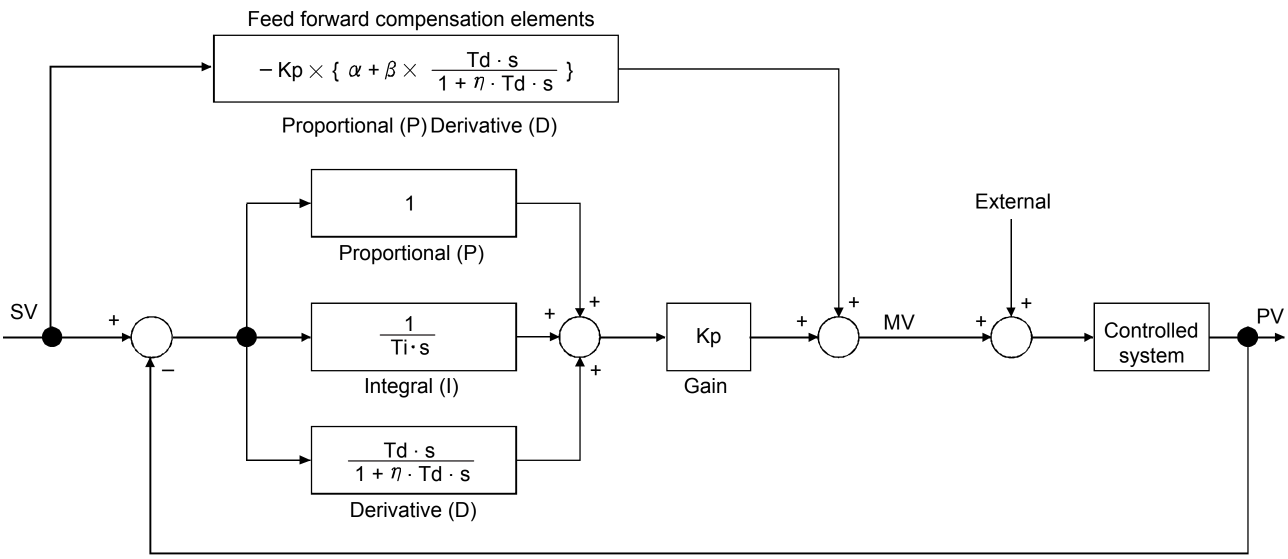
(a) Operation overview

When traditional PID control is applied, optimum PID constant for target tracking is not the same as the optimum PID constant for disturbance response in most cases. Whichever optimum value is applied, it may be the non-optimum value on the other side.

2-degree-of-freedom control is to settle this problem. It allows adjustment on both disturbance response and target tracking.

α and β is used in 2-degree-of-freedom PID control.

M_2PID and M_2PIDH_ of tag FB are compatible with the 2-degree-of-freedom PID control.



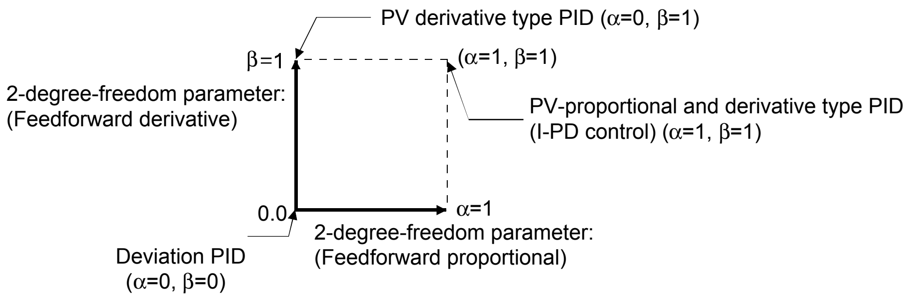
CPU module 2-degree-of-freedom PID operation expression

	Direct action (PN=1)	Reverse action (PN=0)
Deviation DVn	$DV_n = PV_n - SV_n$	$DV_n = SV_n - PV_n$
Output variation △MV	$\triangle MV = \underbrace{Kp}_{\text{Gain}} \times \{ \underbrace{(1-\alpha) \times (DV_n - DV_{n-1})}_{\text{Proportional}} + \underbrace{\frac{CT}{T_I} \times DV_n}_{\text{Integral}} + \underbrace{(1-\beta) \times B_n}_{\text{Derivative}} + \underbrace{\alpha \times C_n + \beta \times D_n}_{\text{Feedforward compensation}} \}$	
Bn	$B_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{(DV_{n-2}DV_{n-1} + DV_{n-2}) - \frac{CT \times B_{n-1}}{Td}\}$	
Cn	$PV_n - PV_{n-1}$	$-(PV_n - PV_{n-1})$
Dn	$D_n = D_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{(PV_{n-2}PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{Td}\}$	$D_n = D_{n-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{-(PV_{n-2}PV_{n-1} + PV_{n-2}) - \frac{CT \times D_{n-1}}{Td}\}$

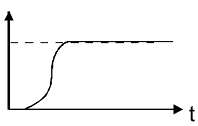
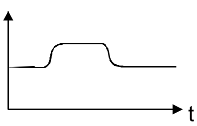
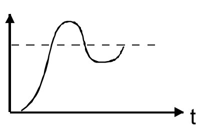
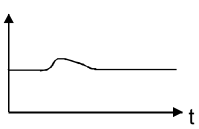
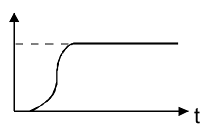
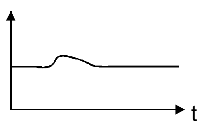
Kp: Gain, T_I: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Previous deviation value, DV_{n-2}: Deviation value before last, PVn: Process variable, PV_{n-1}: Previous process variable, PV_{n-2}: Process variable before last, α: 2-degree-of-freedom PID parameter (feedforward proportional) β: 2-degree-of-freedom PID parameter (feedforward derivative).

When 2-degree-of-freedom PID control is applied, characteristics can be changed by adjusting α and β after the constants of P, I and D are determined.

1) When α=0, β=0: deviation PID
Derivative action is effective to deviation (difference between setting value and process variable), so that the target tracking performance corresponding to the change of setting value will become better.
2) When α=0, β=1: PV-derivative type PID (corresponding tag FB: M_PID)
As derivation operation is effective to process variable, in comparison with derivative PID, disturbance response will be accelerated. On the other hand, the target tracking performance corresponding to the change of setting value will decrease.
3) When α=1, β=1: PV-proportional and derivative type PID (I-PD control) (corresponding tag FB: M_IPD)
As both proportional and derivative action is effective to process variable and integral control action is effective to deviation (difference between setting value and process variable), the target tracking performance, in comparison with PV-derivative PID, will decrease corresponding to the change of setting value.
This is effective for the following cases. Because the manipulated variable does not change suddenly when the setting value is changed.
   - Over shoot is not permitted.
   - Make it respond slowly not to give a certain shock to the final control element and the system.



(b) The response of traditional PID control and 2-degree-of-freedom PID control
The response of traditional PID control and 2-degree-of-freedom PID control are as follows.

Traditional PID control	2-degree-of-freedom PID control
(1) When target tracking performance of setting value changing is better, disturbance response will become worse.  Target tracking performance  Disturbance response (2) When disturbance response performance is better, the target tracking performance of setting value changing will become worse.  Target tracking performance  Disturbance response	The target tracking performance of setting value changing and the disturbance response will both become better.  Target tracking performance  Disturbance response

(c) Adjustment method of 2-degree-of-freedom PID control

1) Calculate PID constant by using auto tuning.

2) Fine tune PID constants (basic parameters of PID: Kp, T_I, T_D) to optimize the response performance for disturbance if necessary.

(Proportional gain K_P)
- If K_P is tuned down, the manipulated variable will become smaller, and it will take a longer time to be stable.
- If K_P is tuned up, the manipulated variable will become bigger, there may be oscillation in response due to the enhancement of compensation operation.

(Integral time T_I)
- If T_I is tuned down, integral control action will be enhanced, and the response will sometimes become oscillation. (Oscillation period becomes longer.)
- If T_I is tuned up, integral effect will be come smaller, and it will take a long time to be stable.

(Derivative time T_D)
- If T_D is tuned down, derivative effect will become smaller, and derivative will only effect for a short period of time.
- If T_D is tuned up, derivative effect will become bigger, short-period oscillation will occurs, and sometimes the system will be quite unstable.

(3) Hold the optimum disturbance response, while adjusting 2-degree-of-freedom parameter (α, β) to optimize the target tracking response.
- If α is tuned up, the manipulated variable in relation to setting value changing will become smaller, and it will take a longer time to be stable.
- If α is tuned down, the manipulated variable in relation to setting value changing will become bigger, and response will sometimes become oscillatory due to the enhancement of compensation operation.
- If β is tuned up, the derivative effect corresponding to setting value changing will become smaller, and derivation will only effect for la short period of time.
- If β is tuned down, the derivative effect in relation to setting value changing will become bigger, and short-time period oscillation will occur, sometimes the system will be unstable.

The response performance corresponding to setting value changing when α is changed is as follows.
Quick: α =0, Medium: α =0.65, Slow: α =1
(Here β =1. the derivative action corresponding to setting value changing makes manipulated variable change sharply (kick), and shock on the final control element and system. Therefore when β =1, usually treat the derivative action in relation to setting value changing as invalid.)

(d) Setting of 2-degree-of-freedom parameter α, β.
On the programming tool FB property window, set the tag data ALPHA2 as α and set BETA2 as β

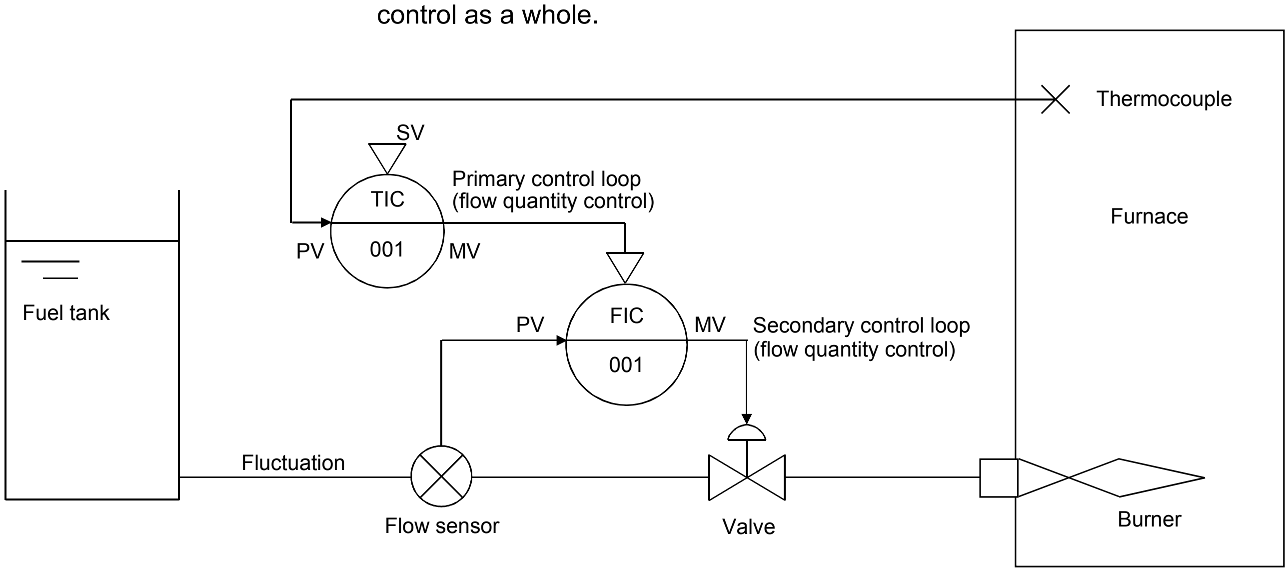
Offset	Item	Name	Setting/Storage range		Unit	Initial value	Data type	Storage	Number of digits after decimal point	Tag access FB
			Low limit	High limit						
+64	ALPHA2	2-degree-of-freedom parameter α	0	1	—	0.00	REAL	User	2	P_2PID
+66	BETA2	2-degree-of-freedom parameter β	0	1	—	1.00	REAL	User	2	P_2PID

Appendix 3.6 Various Control

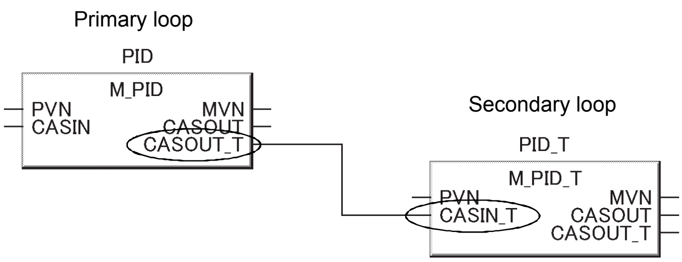
(1) Cascade control

(a) Operation overview

Cascade control consists of primary loop and secondary loop. It is the control that removes the effect on the process and improves the whole control performance by checking out disturbance entering secondary loop in an early stage as well as absorbing them into secondary loop.
Usually, the response of secondary loop is over 3 times faster than primary loop.
Following diagram is an example of controlling the furnace temperature in a certain value. It absorbs fuel supply variation by flow quantity control of secondary control and can improve response characteristics of temperature control as a whole.



(b) Example of applying cascade connection with FB when tracking is needed.
Connect the CASOUT_T of primary loop and the CASIN_T of secondary loop.
For secondary loop, the tag FB shall be the tag FB (M_PID_T, M_2PID_T, etc.) with CASIN_T pins attached.



When tracking is applied, the operation constants of secondary loop tag FB shall be set for tracking.

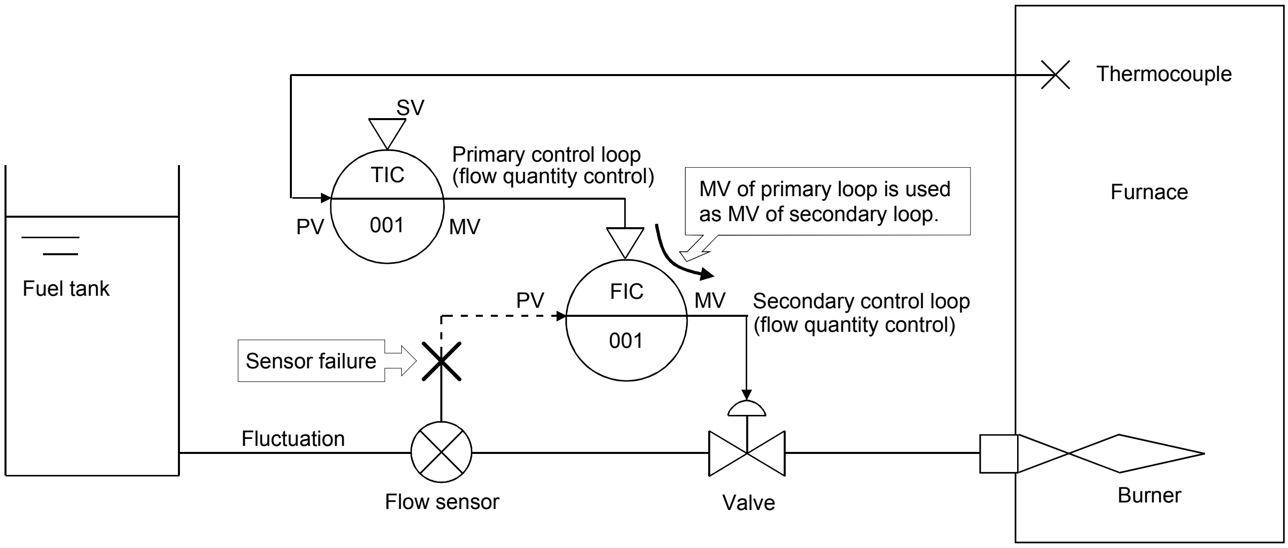
Operation constant item	Contents	Settings for tracking
PID_TRK	0: Without tracking 1: With tracking	1
PID_SVPTN_B0	TRUE : Without primary loop connection FALSE : With primary loop connection	FALSE
PID_SVPTN_B1	TRUE : SV is not of MV of upper loop FALSE : SV is of MV of primary loop	When primary loop is tag FB: FALSE (usually FALSE) When primary loop is not tag FB: TRUE

(c) Cascade direct

For 2PIDH tags, Cascade Direct (CASCADE DIRECT) mode can be selected as a control mode. In Cascade Direct control, output value of the primary loop is directly output as output value of the secondary loop in the cascade connection.

In the case of input sensor failure, the output result of the primary loop is substituted for and directly output as the output value of the secondary loop since the PID operation result of the secondary loop will be illegal.

The CASCADE DIRECT mode can be set with the tag of secondary loop.

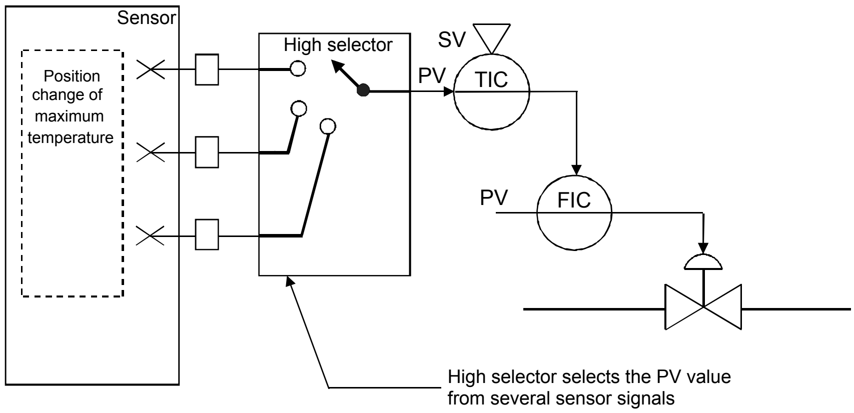


(2) Selection control

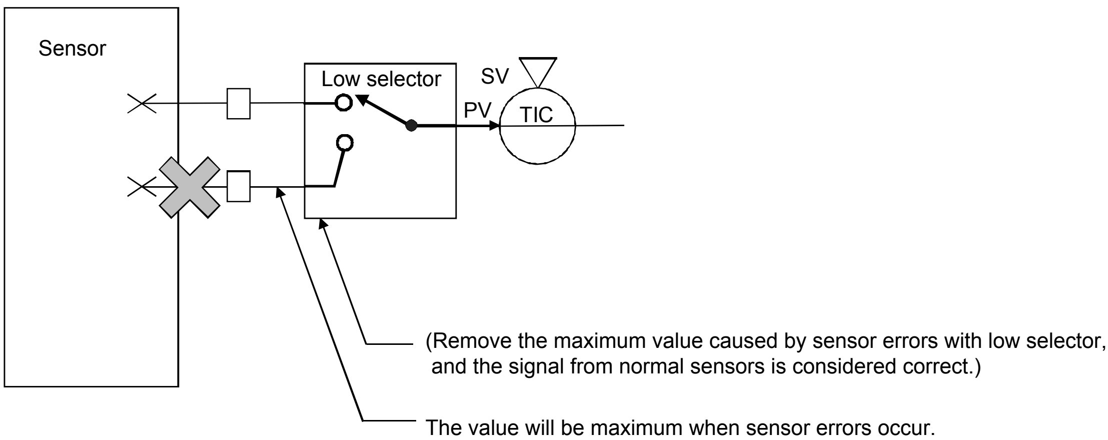
(a) Operation overview

By this method, users can select the necessary signals among various sensor signals or operation signals (high selection, low selection, Middle value selection) for control.

(Example 1) When the highest temperature position changes, the control is performed by selecting the highest temperature among two or more measurement points.



(Example 2) In case that the input signal from the sensor becomes maximum when sensor errors such as wire break occur.
The redundancy of the system is realized by installing two or more sensors for sensor wire break and trouble and selecting the normal one.
(Connect multiple sensors, combine the low, high and intermediate selectors according to the status when burnout occurs and obtain normal sensor signals. (When burnout occurs, the signal of sensor is the maximum or the minimum.))



Appendix 3.7 PID Operation

(1) Proportional (P) Control action

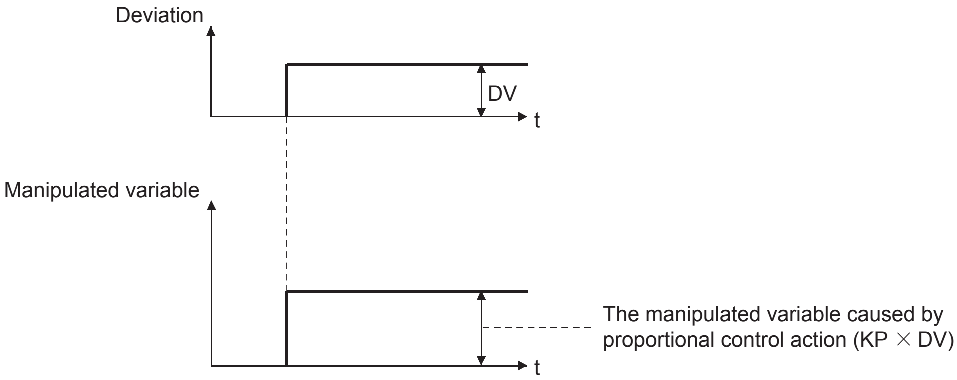
(a) Proportional (P) Control action

The proportional (P) control action is the operation that obtains the manipulated variable in proportion to deviation DV (difference between process variable and setting value)

$$\boxed{\text{Manipulated variable = Proportional gain Kp} \times \text{Deviation DV}}$$

Proportional gain Kp: Not proportional band but proportional gain = 100 / proportional band (%)

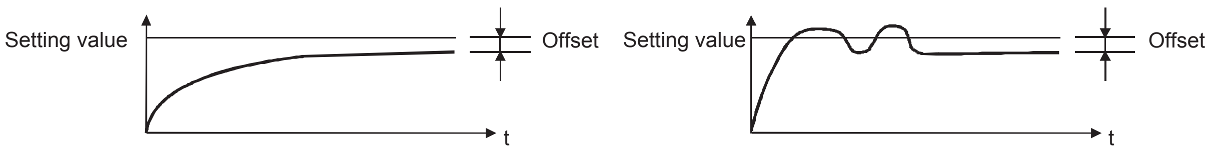
The proportional action of step response whose deviation is a certain value is as follows.



Condition	Proportional control action
When proportional gain Kp is relatively smaller	Control operation become slow
When proportional gain Kp is relatively bigger	Control operation become faster and easy to cause hunting

(b) Offset

The error to setting value is named offset. Offset will occur in proportional control action.



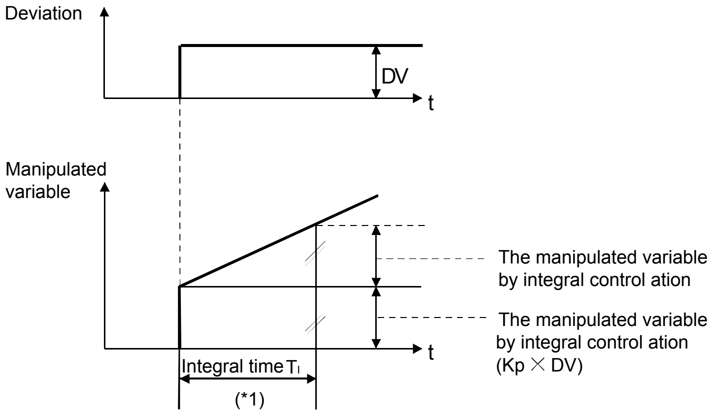
(2) Integral (I) operation

(a) Integral (I) operation

Integral control action is the operation that continuously changes the manipulated variable, in order to eliminate deviation DV (difference between process variable and setting value).

It can eliminate offset caused in proportional action.

The time interval from the moment when deviation occurs until the manipulated variable determined by integral action equals the manipulated variable determined by proportional control action is called Integral time "Ti".



(*1) The time interval in which the manipulated variable due to integral control ation equal to the manipulated variable due to the proportional control ation is called integral time (T_I).

Condition	Integral control action
When integral time T_I is relatively smaller	The integral effect becomes stronger, and the time for eliminating offset becomes shorter. But hunting may easily occur.
When integral time T_I is relatively bigger	The integral effect becomes lighter, and the time for eliminating offset becomes longer.

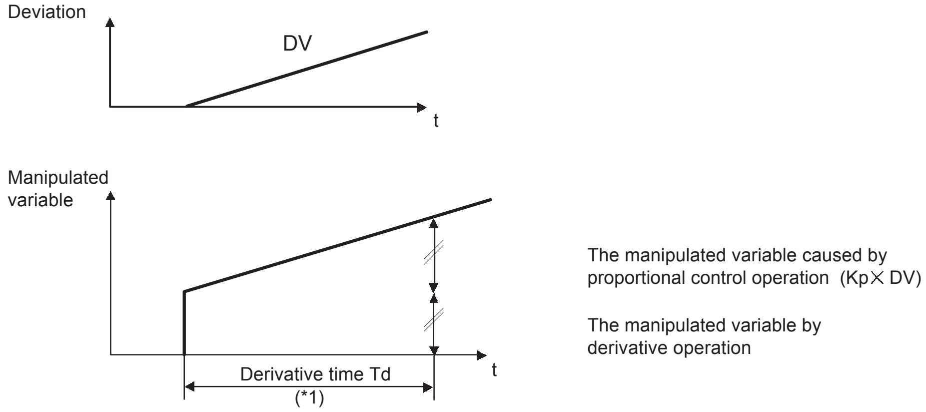
(3) Derivative (D) action

(a) Derivative (D) action

This is the operation that imposed on the manipulated variable that is in proportion to the variation rate (difference between the current value and the last value) of deviation DV (the difference between process variable and setting value).

The time interval from the moment when deviation occurs until the manipulated variable determined by derivative action equals the manipulated variable determined by proportional control action is called Derivative time "Td".

When deviation is changing at a constant rate



(*1) The time interval in which the manipulated variable due to proportional operation equal to the manipulated variable due to derivative operation is called Derivative Time (Td).

Condition	Differential action
When derivation time T_D is relatively smaller	Derivation effect becomes lighter.
When derivation time T_D is relatively bigger	Derivation effect becomes stronger. Cause short-period hunting, and the system may become unstable.

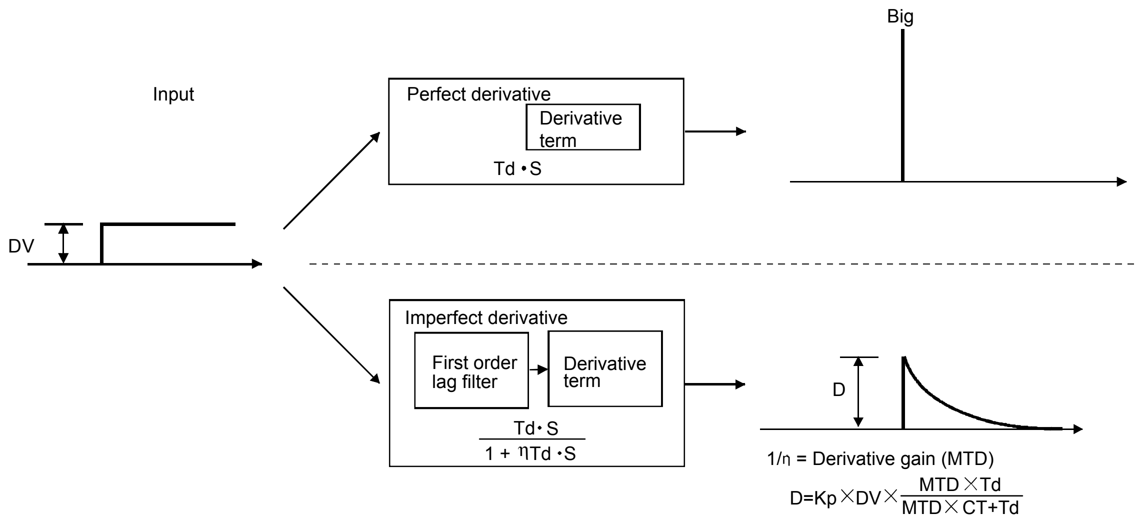
(b) Imperfect derivative

If derivative is applied as it is, it may be effected by increase of high-frequency noise, and because the time range of MV is narrow (in case of step-shaped change, it will be output only at the moment like pulse shape.). There may be the bad influence that the energy which outputs final control element fully is not given.

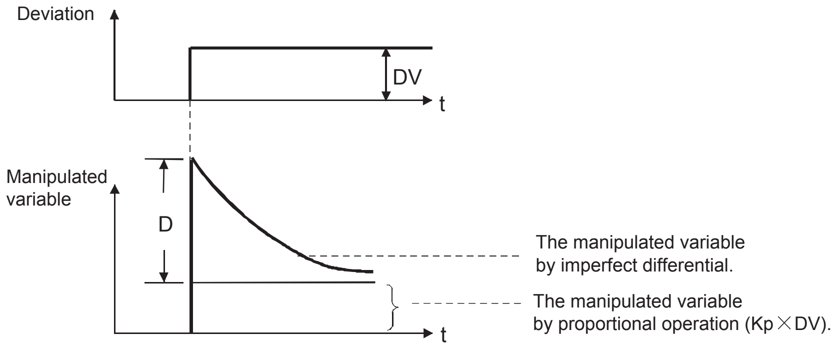
Therefore, normally the derivative term input with imperfect differentiation for which filter shall be applied once.

M_PID, M_IPD, M_2PID of tag FB and P_PID, P_IPD, P_2PID of tag access FB have applied imperfect differentiation.

Derivative gain (MTD) can be set by operation constant.



Step response operation with a constant deviation



(4) PID operation

(a) PID operation

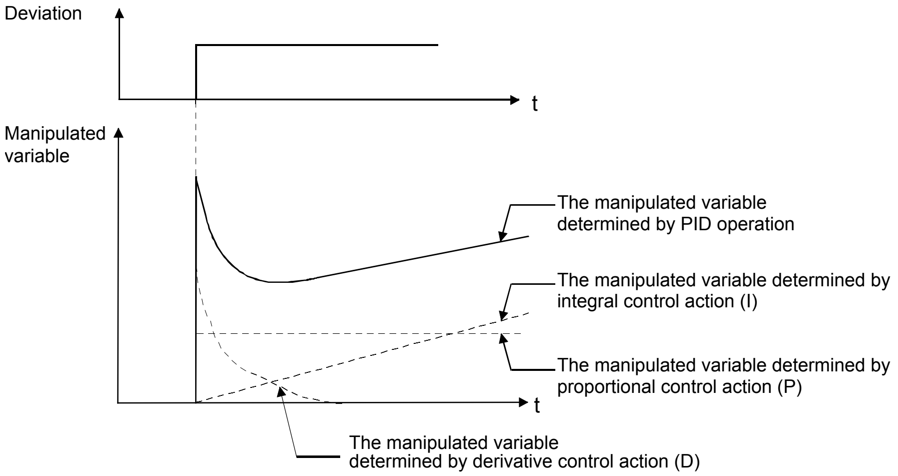
This is the control operation which output the manipulated variable (MV), so that process variable (PV) can approach setting value (SV) rapidly and correctly by combining P control action, I control action and D control action. Besides, if P, I, D operation are not all included in the control, it will be named P control or PI control according to the control action included.

CPU module velocity type process variable derivation PID expression

	Direct action (PN =1)	Reverse action (PN =0)
Deviation DVn	DVn=PVn-SVn	DVn=SVn-PVn
Output variation △MV	$\Delta MV = \underbrace{Kp}_{\text{Gain}} \times \{\underbrace{(DV_n - DV_{n-1})}_{\text{Proportional}} + \underbrace{\frac{CT}{T_I} \times DV_n}_{\text{Integral}} + \underbrace{B_n}_{\text{Derivative (imperfect derivative)}}\}$ Proportional, integral and derivative term of △MV are as follows. Proportional term: $\Delta MV = Kp \times (DV_n - DV_{n-1})$ Integral term: $\Delta MV = Kp \times \frac{CT}{T_i} \times DV_n$ Derivative term: $\Delta MV = Kp \times Bn$ (Bn as follows)	
Bn	$Bn = Bn_{-1} + \frac{Md \times Td}{Md \times CT + Td} \times \{(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times B_{n-1}}{Td}\}$	$Bn = Bn\text{-}1 + \frac{Md \quad Td}{Md \times CT + Td} \times \{-(PV_n - 2PV_{n-1} + PV_{n-2}) - \frac{CT \times B_{n-1}}{Td}\}$

Kp: Gain, T_I: Integral time, Td: Derivative time, Md: Derivative gain, CT: Control cycle, DVn: Deviation, DV_{n-1}: Previous deviation value, PVn: Process variable, PV_{n-1}: Previous process variable, PV_{n-2}: Process variable before last

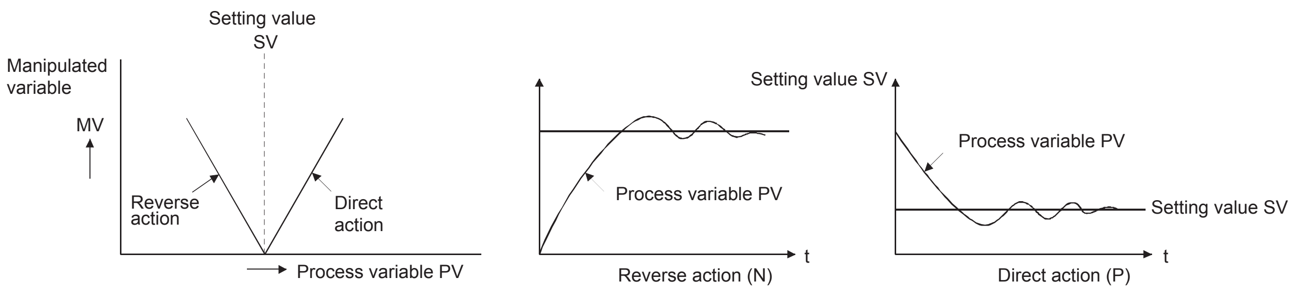
1) The step response operation with constant deviation



2) Direct action and reverse action (PN)

- Direct action (P): Decreases manipulated variable when the process variable is bigger than the setting value (like cooling)
- Reverse action (N): Increases manipulated variable when the process variable is smaller than the setting value (like heating)

Reverse action and direct action (PN) can be set by operation constant.



Appendix 3.8 Control Mode

(1) Types of control mode
Control mode types of tag FB are as follows.
Valid control modes are different due to different tag types.
For the corresponding relation between tag type and control mode, refer to Appendix 1.3 (3).

(a) Manual Mode (MANUAL)
This is the mode of manual operation. Output MV.

(b) Automatic Mode (AUTO)
This is the mode for automatic operation. Control MV according to SV.

(c) Cascade Mode (CASCADE)
This is the mode for cascade operation. Set the output value (MV) of primary loop as setting value.

(d) Computer MV (COMPUTER MV) Mode
This is the mode of manual operation with upper computer. Output the upper MV from upper computer.

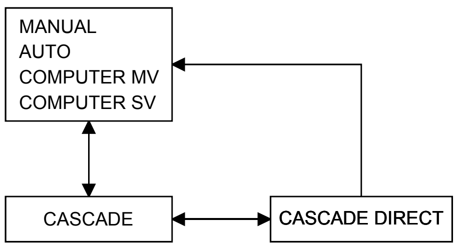
(e) Computer SV (COMPUTER SV) Mode
This is the mode for automatic operation with upper computer. Control MV according to SV which comes from upper computer.

(f) Cascade Direct (CASCADE DIRECT) Mode
This is the mode for directly outputting the output value of primary loop as the output value of secondary loop in the cascade connection.

(2) Control mode transition

(a) Control mode transition when the tag type is other than 2PIDH
The control mode transition has no restriction.

(b) Control mode transition when the tag type is 2PIDH
Mode change to CASCADE DIRECT mode can be performed only from CASCADE mode.
Other control mode changes have no restriction.

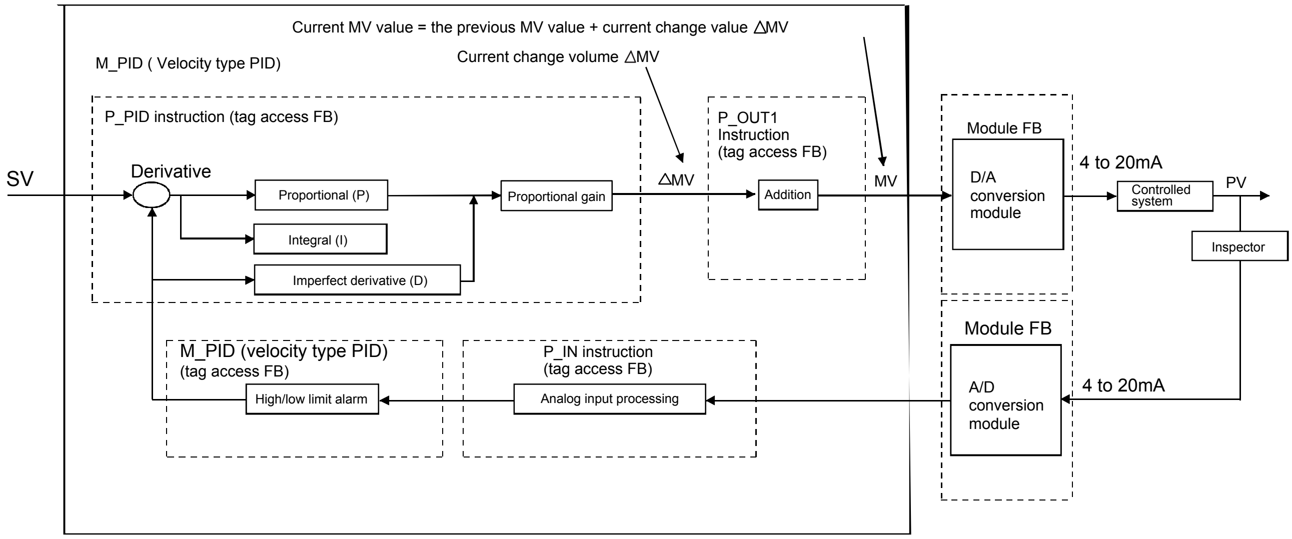


POINT

When loop stop alarm (SPA) occurs, all the current control modes will be converted to MANUAL forcibly and automatically.
For loop stop alarm (SPA) occurrence, refer to Appendix 3.10.

Appendix 3.9 Velocity Type PID and Position Type PID

(1) Velocity type PID
Velocity type PID is an operation method for calculating the difference (△MV) of current and previous MV.
In the following chart, velocity type PID operation output △MV by using tag access FB's P_PID. Output △MV is added to previous MV by P_OUT1, and the manipulated variable MV is output to the controlled system.
Compared to PID position type, velocity type is more convenient in operation of bumpless manual-auto switching, prevention of reset wind-up, complicated control and slow change when gain is changed. Hereby the velocity type has become the mainstream choice.
Perform velocity type operation for the P_PID, P_SPI, P_IPD, P_BPI, P_2PID, P_2PIDH_ of tag access FB, and output △MV.
Output △MV is added to previous MV by P_OUT1 of tag access FB, and the result is manipulated variable MV.
M_PID, M_SPI, M_IPD, M_BPI, M_2PID, M_2PIDH_ of tag FB are corresponding to this.



(2) Position type PID
Position type PID is a PID operation method which calculates the manipulated variable MV.
M_PIDP of tag FB and P_PIDP of tag access FB are corresponding to this.

Appendix 3.10 Stop Alarm Processing in Loop Control

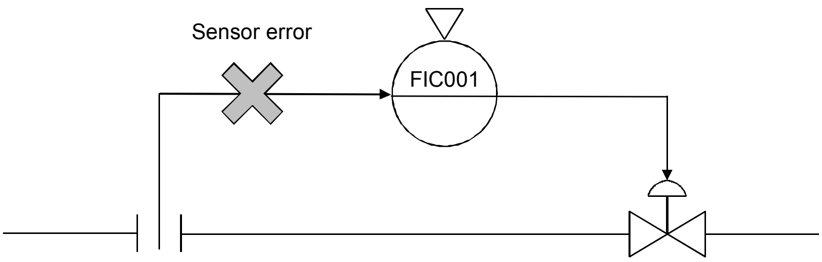
(1) Stop alarm (SPA) overview
When loop tag memory alarm (sensor error (SEA) etc.) occurs, the control mode can be changed to Manual (MAN) by setting stop alarm (SPA) FALSE → TRUE.
The operation of stop alarm (SPA) from FALSE → TRUE can be executed by user's program according to needs.
Besides, when stop alarm (SPA) is set as TRUE, the alarm which has occurred. (MLA, MHA, DVLA, DPPA, PLA, PHA, LLA, HHA, SEA, DMLA) will be automatically reset (TRUE → FALSE).

(2) The alarm (ALM) items of loop tag memory
The alarm (ALM) items of loop tag memory is as follows.
For the list of loop tag memory, refer to Appendix 1.2 (1).

Offset	Item	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
+3	ALM Alarm		SPA Stop alarm			DMLA Output variation rate limit	OOA Output open	SEA Sensor error	HHA Input high high limit	LLA Input low low limit	PHA Input high limit	PLA Input low limit	DPPA Positive variation rate	DPNA Negative variation rate	DVLA Large deviation	MHA Output high limit	MLA Output low limit
			User *2			System	User	System	System	System	System	System	System	System	System	System	System
			TRUE:Occur FALSE: Reset			TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE :Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset	TRUE: Occur FALSE: Reset

Alarm (ALM) consists of multiple BOOL variables. (refer to the table above)
All the BOOL variables configuring alarm are global variables.
It is used in FBD program as follows.

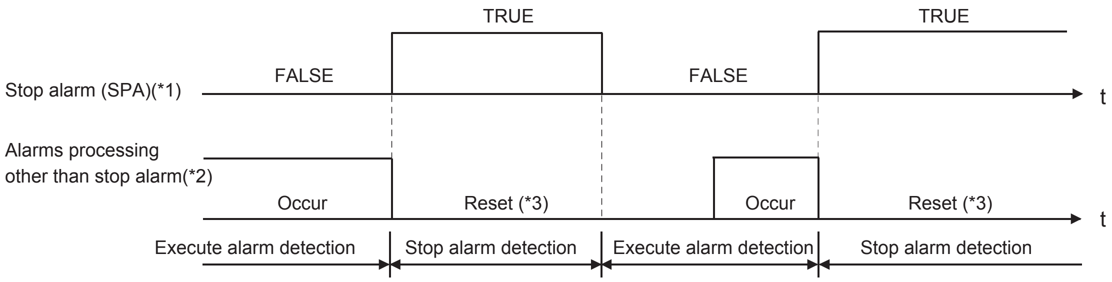
(Example) When loop sensor error (SEA) of PID1 occurs:



In above example, sensor error (SEA) can be acquired through external variables parts.

(3) The relation between stop alarm (SPA) and the other alarms of loop tag memories.
The relation between stop alarm (SPA) and the other alarms of loop tag memories is as follows.

Condition Stop alarm (SPA)	Results
TRUE	Stop alarms processing, except "stop alarm" (MLA, MHA, DVLA, DPPA, PLA, PHA, LLA, HHA, SEA, DMLA), become to FALSE automatically.
FALSE	Execute alarms processing, except "stop alarm" (MLA, MHA, DVLA, DPPA, PLA, PHA, LLA, HHA, SEA, DMLA)



*1 Stop alarm (SPA) can be set to TRUE or FALSE by user's program.
*2 Output open alarm (OOA) follows the processing in the user's program.
*3 When stop alarm (SPA) is TRUE: alarm detection processing stops.
Alarm in occurrence will be reset automatically.

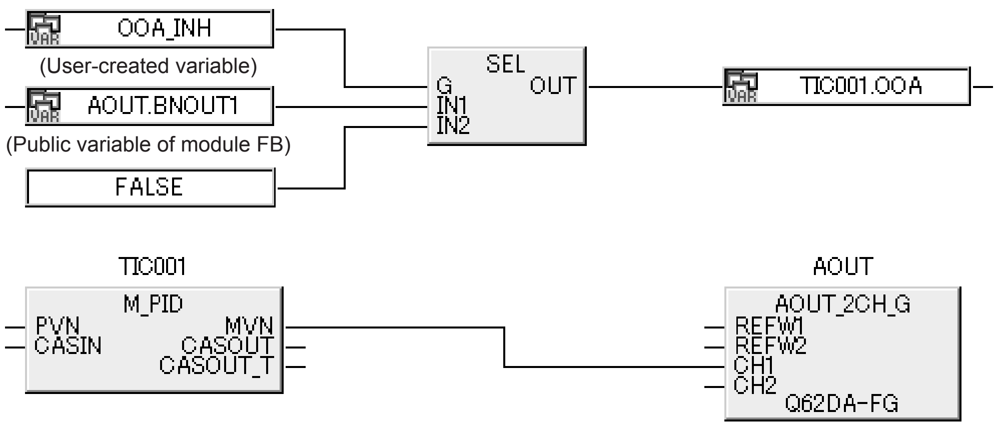
Appendix 3.11 How to Use Output Open Alarm

The output open alarm (OOA) of loop tag memory is designed for controlling to display the wire break detection signal as an alarm on the loop tag FB of the output source when a disconnection is detected on the module FB on the output side.
The following shows a programming method and program example for giving a signal feedback from the module FB that is to detect a disconnection to the loop tag FB that is to display it as an alarm.

<Programming method>

Input the public variable (BNOUT1) of module FB on the output side to the tag item (OOA) of loop tag FB.
For the output open alarm (OOA), the disable alarm detection flag does not exist. Therefore, program separately to enable or disable this alarm.

<Program example>

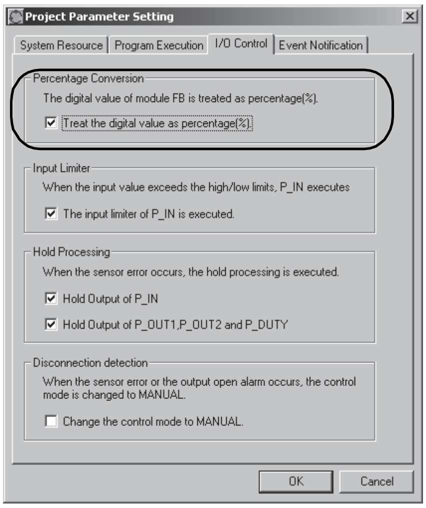


Appendix 3.12 Converting Digital Value of Analog Module FB to Percentage

On the <<I/O Control>> tab of the Project Parameter Setting dialog box, whether to treat a digital value of an analog module FB as percentage (%) can be set. When this item is selected, analog I/O values are treated as percentage (%) without regard to their module types.
The following shows the module FBs that support this function.

Analog module FB
AIN_4CH, AIN_8CH, AIN_4CH_G, AIN_8CH_G, AIN_2CH_DG, AIN_6CH_DG, AOUT_2CH, AOUT_4CH, AOUT_8CH, AOUT_2CH_G, AOUT_6CH_G, AIN_4CH_AOUT_2CH, CT_8CH

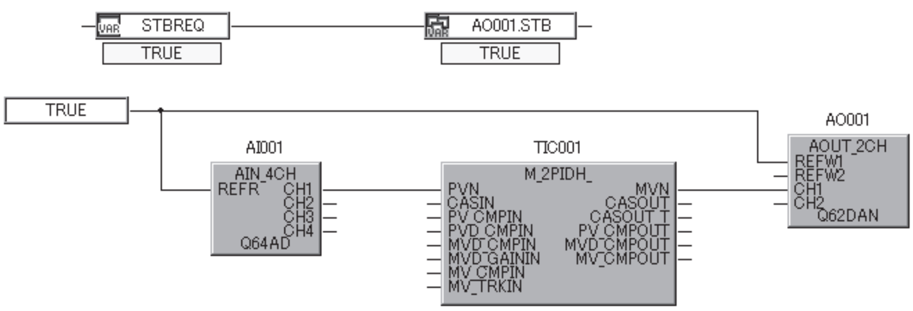
The following shows the program example and the setting method of variables of the tag FB connected to the analog module FB.



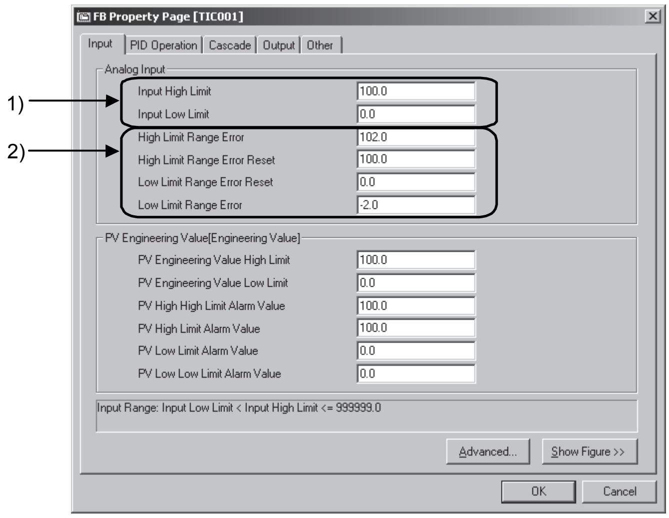
POINT

- The percentage conversion is based on the digital values of the analog I/O range (off set value to gain value).
- When this item is selected, and the input range of 0 to 10V is used in the high resolution mode with Q68ADV, the input range setting value of the switch setting item must be set to 5H. If it is set to 0H, the digital value percentage conversion of module FB (AIN_8CH) does not operate properly.
- For the channels of analog modules in which the scaling function is valid, the percentage conversion is not performed even when this setting is selected, and values converted by the scaling function are used.

<Program example>



(1) Analog input setting
Set values of input high/low limit, range error, and range error reset.



1) Input high/low limit
Set values as shown below since the percentage conversion is based on the digital output values in the range from off set value to gain value.

- If the digital output value for the analog input range is 0 to XXXX, the default values can be used.

Item	Setting value
Input High Limit (IN_NMAX)	100.0
Input Low Limit (IN_NMIN)	0.0

- If the digital output value for the analog input range is -XXXX to XXXX, set the input high/low limit in the range from -100.0 to 100.0 in accordance with the usage range of analog input.

< Setting example when using the input range of -10 to 10V (digital value -4000 to 4000) as -5 to 7V >

Item	Setting value
Input High Limit (IN_NMAX)	70.0
Input Low Limit (IN_NMIN)	-50.0

2) Range error, range error reset
The default values are based on the default input range of an analog module. When the input range of the analog module is changed, change the values as required.

- When the extended mode is not used for the input range
If the maximum/minimum digital output value of the analog input module is ±2% or more for the input range, the default values can be used.

< Setting example when the input range is 4 to 20mA, and digital output value is 0 to 4000 (minimum: -96, maximum: 4095) >

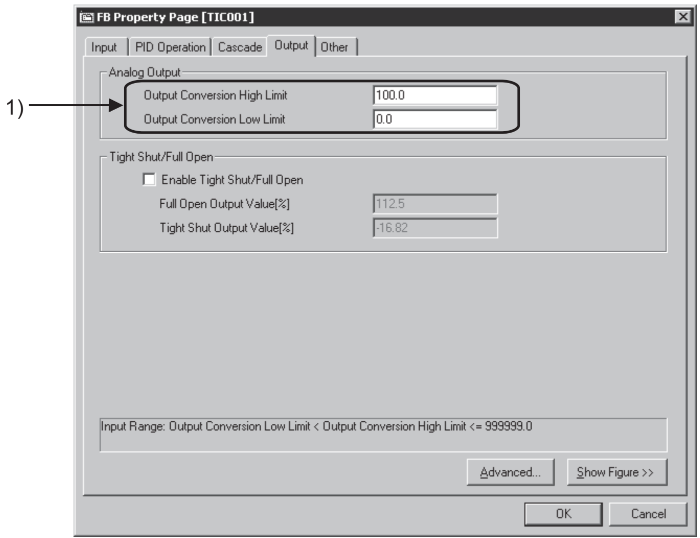
Item	Setting value	Remarks
High Limit Range Error (IN_HH)	102.0	Equivalent to digital output value 4080
High Limit Range Error Reset (IN_H)	100.0	Equivalent to digital output value 4000
Low Limit Range Error Reset (IN_L)	0.0	Equivalent to digital output value 0
Low Limit Range Error (IN_LL)	-2.0	Equivalent to digital output value -80

- When the extended mode is used for the input range
  Range errors can be detected with a value larger than the default value (±2%).

< Setting example when the input range is 4 to 20mA (extended mode), and digital output value is 0 to 4000 (minimum: -1096, maximum: 4595) >

Item	Setting value	Remarks
High Limit Range Error (IN_HH)	110.0	Equivalent to digital output value 4400
High Limit Range Error Reset (IN_H)	108.0	Equivalent to digital output value 4320
Low Limit Range Error Reset (IN_L)	-8.0	Equivalent to digital output value -320
Low Limit Range Error (IN_LL)	-10.0	Equivalent to digital output value -400

(2) Analog output setting
Set values of output conversion high/low limit.



1) Output conversion high/low limit
Set values as shown below since the percentage conversion is based on the digital input values in the range from off set value to gain value.

- If the digital input value for the analog output range is 0 to XXXX, the default values can be used.

Item	Setting value
Output Conversion High Limit (OUT3_NMAX)	100.0
Output Conversion Low Limit (OUT3_NMIN)	0.0

- If the digital input value for the analog output range is - XXXX to XXXX, set the output conversion high/low limit in the range from -100.0 to 100.0 in accordance with the usage range of analog output.

< Setting example when using the output range of -10 to 10V (digital value -4000 to 4000) as -5 to 7V >

Item	Setting value
Output Conversion High Limit (OUT3_NMAX)	70.0
Output Conversion Low Limit (OUT3_NMIN)	-50.0

Appendix 3.13 Tracking

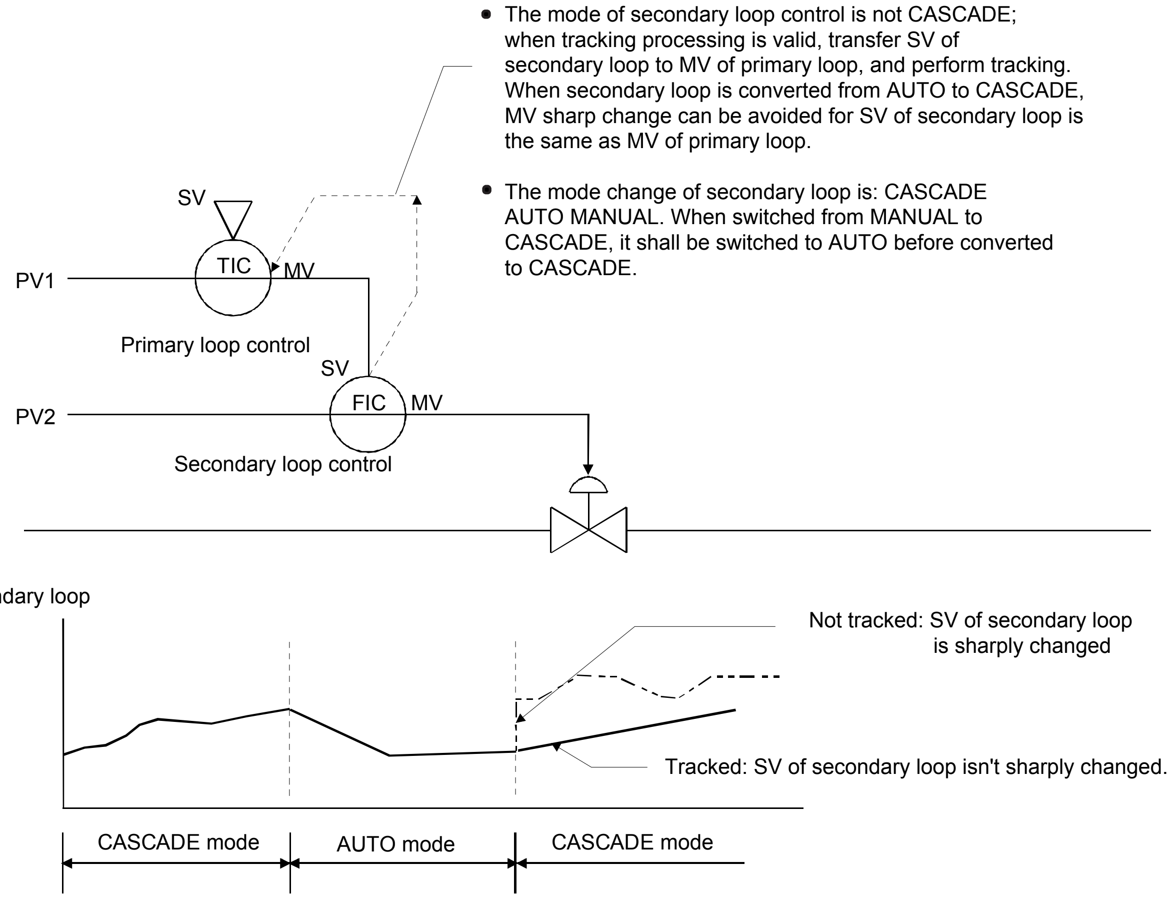
(1) Operation
Tracking has two kinds of functions.

(a) Bumpless function
At the time of Auto ↔ Manual mode switching, this function prevents step-shaped changes caused by sharp change of manipulated variable (MV) output, and ensures MV value to be converted smoothly and bumplessly.

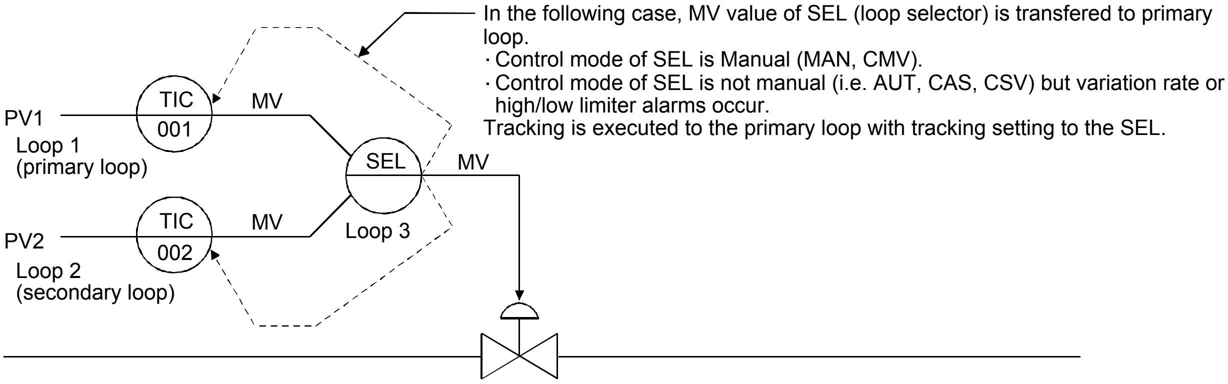
(b) Output limiter processing function
It can limit manipulated variable (MV) within the high/low limit which is output by PID operation in Auto Mode.
This function is only valid in Auto Mode, and cannot be executed in Manual Mode.
Additionally, when the primary loop is Auto Mode and tracking from the secondary loop is executed, as the tracking data will be stored as MV value, output limiter processing function will not be executed in this case.

(2) Application example

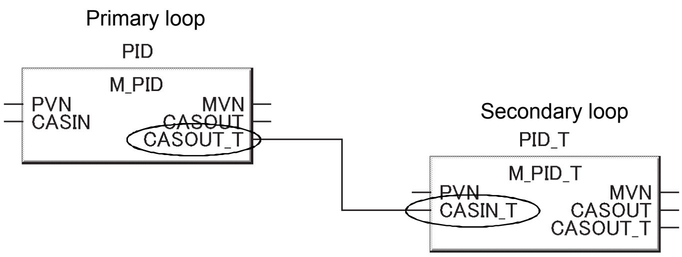
(a) Tracking example of cascade loop
For the control loop which composes cascade loop, if control mode switching of secondary loop is executed, the SV value of secondary loop shall be transmit to MV value of primary loop, in order to prevent sharp changes of SV value.



(b) Tracking examples of loop selector



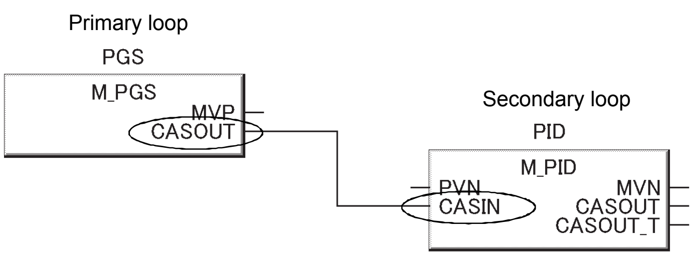
(c) Illustration of cascade connection by using FB when tracking is necessary
Connect CASOUT_T of primary loop with CASIN_T of secondary loop.
The tag FB of secondary loop should be the tag FB (M_PID_T, M_2PID_T, etc.) which has CASIN_T pin.



When tracking is executed, it is necessary to set the tracking of operation constant of secondary loop tag FB.

Operation constant item	Contents	Settings for tracking
PID_TRK	0: Without tracking 1: With tracking	1
PID_SVPTN_B0	TRUE: Not connected with primary loop FALSE: Connected with primary loop	FALSE
PID_SVPTN_B1	TRUE: SV is not MV of primary loop FALSE: SV is MV of primary loop	When primary loop is tag FB: FALSE (normally FALSE) When primary loop isn't tag FB: TRUE

(d) Illustration of cascade connection by using FB when tracking is not necessary
Connect CASOUT of primary loop with CASIN of secondary loop. The tag FB of secondary loop should be the tag FB which has CASIN pins.



When tracking is not necessary, the operation constants of secondary loop tag FB are as follows.

Operation constant item	Contents	Settings for tracking
PID_SVPTN_B0	TRUE: Not connected with primary loop FALSE: Connected with primary loop	FALSE

Appendix 3.14 Simulation Function in I/O mode (SIMULATION mode)

(1) Overview

The simulation function (SIMULATION mode) is the function that does not actually input/output for I/O module, but performs simulation.
Simulation function (SIMULATION mode) is executed after changing the mode to SIMULATION with faceplate.
For details of I/O mode change, refer to the "PX Developer Version 1 Operating Manual (Monitor Tool)".

(2) Function contents

(a) For loop tag FB

Execute the loop control using MV output as feedback input while not executing PV external output and MV external input (separate input and output from the external).
By using it, it is possible to execute loop test separated with the actual plant.

(b) For status tag FB

Separate the input and output from the external and substitute the input signal a certain period after receiving the output instruction.
It can simulate the actual valve ON/OFF instructions and response of answer signal to confirm the control operation.

(3) Creation of simulation data

(a) Creation of simulation data

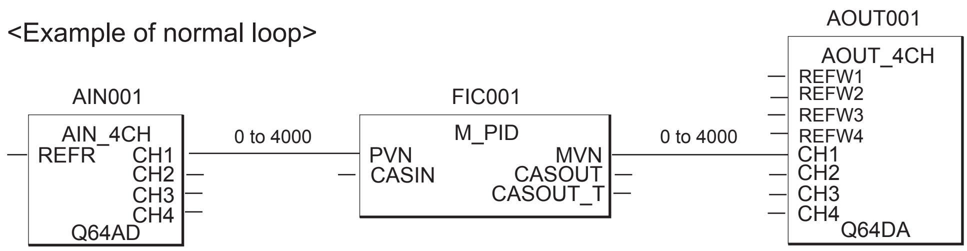
1) For loop tag FB
The simulation data of loop tag FB can be made by returning the simulation output (SIMOUT) to simulation input (SIMIN) in tag FB.

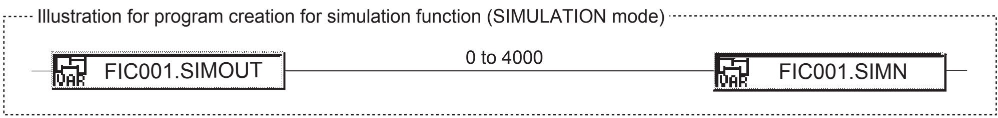
2) For status tag FB
Creation of program which returns input to output is not required for status tag FB.
Set the simulation answer back period (SIMT) of tag data.

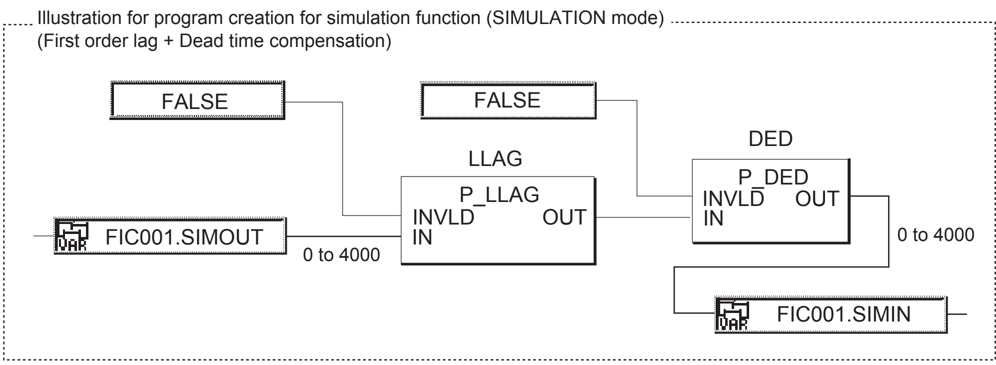
(b) Program example of loop tag FB simulation function (SIMULATION mode)
Following are the program examples of simulation function (SIMULATION mode) that uses loop tag FB.

1) When the range of PVN and MVN are the same
Following is an example of normal loop and cascade.

<Example of normal loop>







<Cascade example>

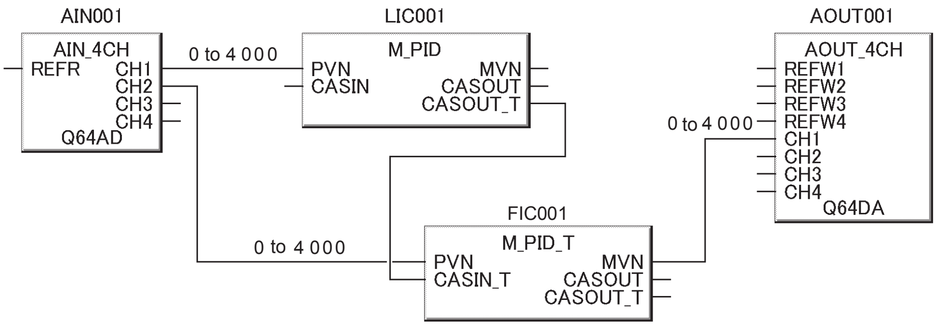


Illustration for program creation for simulation function (SIMULATION mode)

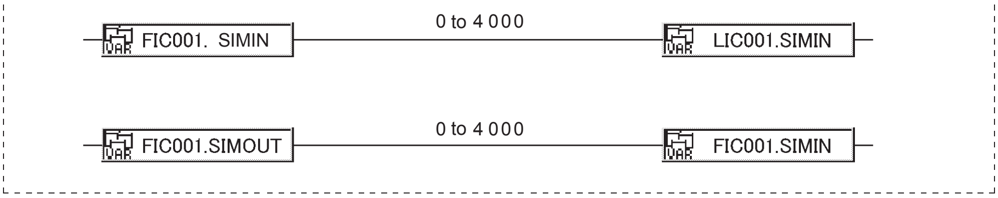
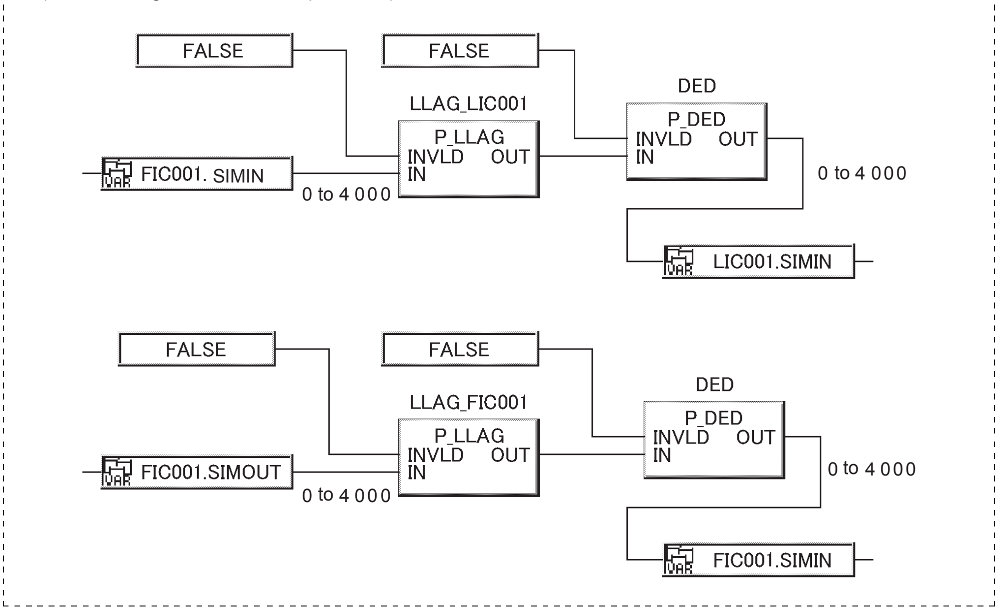


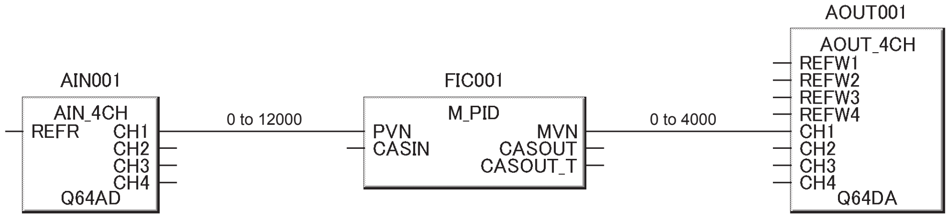
Illustration for program creation for simulation function (SIMULATION mode)
(First order lag + Dead time compensation)

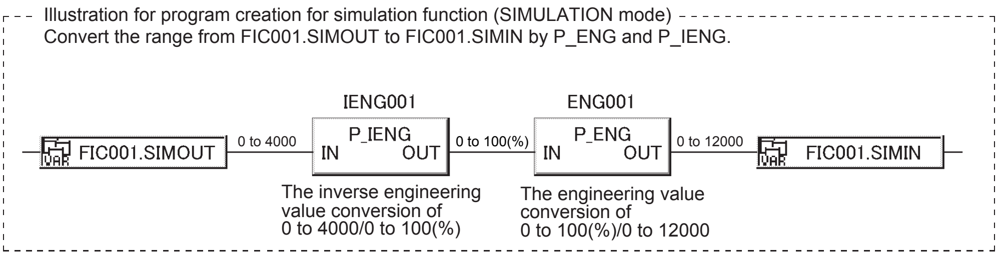


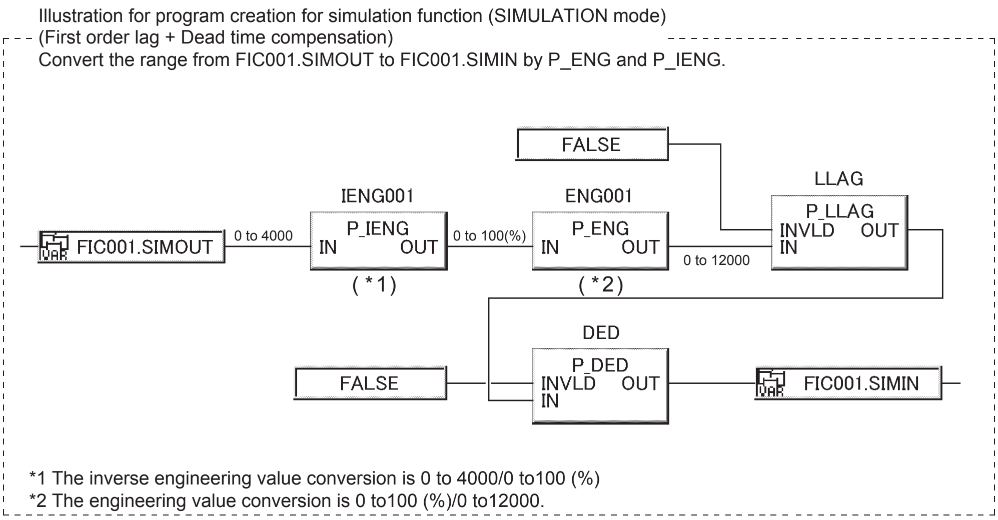
POINT

In the case of cascade connection, use simulation loopback input data of secondary loop (such as FIC001.SIMIN as mentioned above) as simulation input data of primary loop (such as LIC001.SIMIN as mentioned above).

2) When the ranges of PVN and MVN are not the same.
The program example when the range of A/D conversion module (AIN_4CH) is 0 to 12000, and the range of D/A conversion module (AOUT_4CH) is 0 to 4000.







Appendix 3.15 Override Function

(1) Overview
This function enables setting of PV value on the pop-up tuning screen of monitor tool when it is unable to attain the correct input signal that results from the faults of sensor, limit switch and A/D conversion module.
However, the external output such as manual MV and ON/OFF signal is carried out. It is necessary to change the mode to OVERRIDE mode by I/O mode change on faceplate in order to use Override function.
For details of I/O mode change, refer to the "PX Developer Version 1 Operating Manual (Monitor Tool)".

(2) Function contents
(a) For loop tag FB
It enables setting of PV value on the pop-up tuning screen of monitor tool when it is unable to attain the correct input signal that results from the faults of sensor and A/D conversion module. However, external output is carried out. In this case, MV output shall be carried out in MANUAL Mode.
It is used when applying input signal under inter-lock conditions or transition conditions of batch sequence.
The setting of PV value shall be input from the tag monitor column on pop-up tuning screen of monitor tool.

(b) For status tag FB
It enables setting input signal through the pop-up tuning screen of monitor tool when it is unable to get the correct input signal due to reasons as the contact failure of valve ON/OFF limit switch.
However, external output is carried out.
It is used in applying the input signal under interlock conditions or transition conditions of batch sequence.

POINT

For loop tag FB, the override function can be operated in MANUAL mode only. Operations in other than MANUAL mode are disabled due to incorrect sensor inputs.

Appendix 3.16 Tag Stop Function

(1) Overview
The tag stop function stops the input processing and loop control. This is set for tags reserved for future use.
The tag stop function can be set by changing the I/O mode with faceplate. For details of I/O mode change, refer to the "PX Developer Version 1 Operating Manual (Monitor Tool)".

(2) Function contents
The tag stop function can be used for 2PIDH, PGS2 tags.
This function is used for the tags defined by a programming tool in advance or tags being stopped in order to use them in the future.
For details, refer to corresponding FB section in Chapter 7, Chapter 8 and Chapter 9.

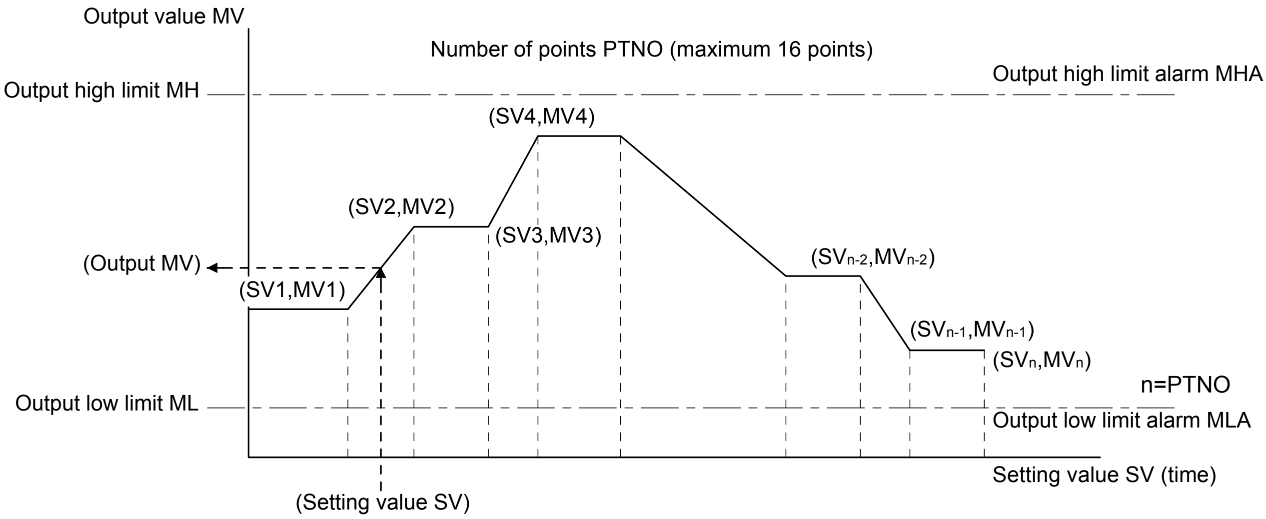
Appendix 3.17 Program Setter Setting Method

The following shows the setting method of Program setter (PGS) and Multi-point program setter (PGS2).

(1) Program setter (PGS)

1) Operation method
An operation uses the X-Y graph method.
Output MV is calculated using the X-Y graph function depending on the SV (time) through the program.

2) Relation between the program setting method and each variable
For details of each variable, refer to Appendix 1.1.
Register a program using the X-Y graph method shown below.



3) Registration format
Point data are registered up to 16 points in real number (REAL).

4) Time management
Time is set by seconds.

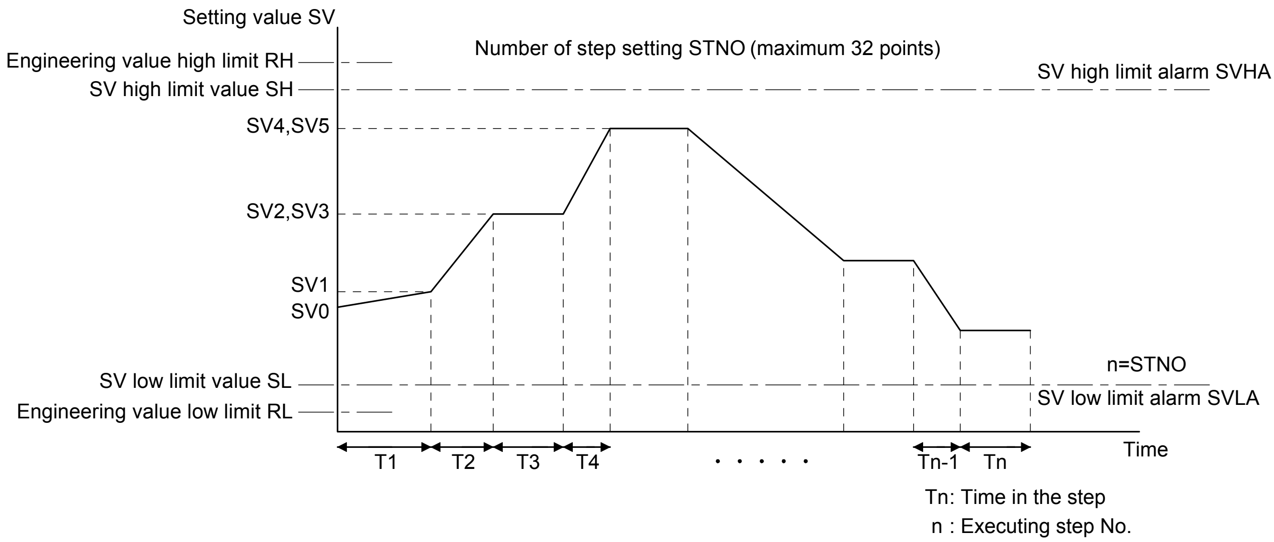
5) Output high/low limit alarm
Output low limit alarm MLA and output high limit alarm MHA are assigned to bit 0 and bit 1 (standard locations) of the loop tag memory +3 (ALM).

6) Mode and operation type
Five control modes are available; MAN, CMV, AUT, CAS, and CSV.
The operation type is CYCLIC in CAS mode.
The operation type is selectable either HOLD or RETURN in AUT mode.

(2) Multi-point program setter (PGS2)

1) Operation method
An operation is performed by registering steps (time span and setting value) and managing the process by each step.
Each step calculates the setting value SV according to the time in the step (T).

2) Relation between the program setting method and each variable
For details of each variable, refer to Appendix 1.1.
Register each step data as shown in the figure below.
Set the program start point to SV0.
Note that the following main parameters are changed from the program setter (PGS).
MV (Output value) → SV (Setting value)
SV (Setting value) → STC (Executing step number) + T (Time in the step)



3) Registration format
Step data are registered up to 32 steps in integer (INT).
The set range is from -32768 to 32767.

4) Time management
Time is set by either seconds or minutes. (Set at TUNIT in the loop tag item.)

5) Output high/low limit alarm
SV low limit alarm SVLA and SV high limit alarm SVHA are assigned to bit 0 and bit 1 of the loop tag memory +3 (ALM).
Note that variable names differ from those for the program setter (PGS).

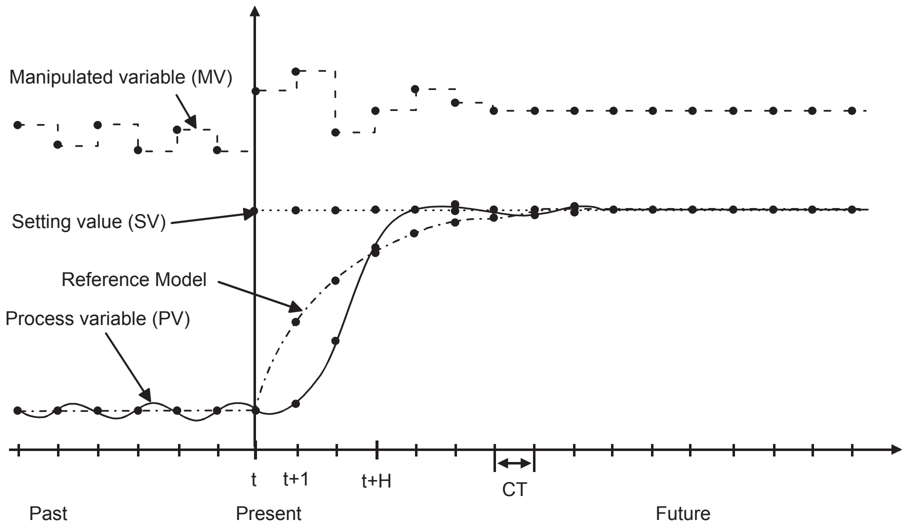
6) Mode and operation type
Two control modes are available; MAN and AUT.
The operation type is selectable either HOLD, RETURN or CYCLIC in AUT mode.

Appendix 3.18 Predictive Functional Control

The predictive functional control is one of advanced control methods and has been applied to a variety of fields. The predictive functional control has internal process models, and this makes the predictive functional control characteristically different from the PID control.
The predictive functional control is suited to control a difficult process with a long dead time and a large time constant.

(1) Operation overview

The predictive functional control predicts the change in the process variable based on an internal model, and outputs the manipulated variable so that the process variable corresponds to the setting value.
In the following chart, the manipulated variable (MV) at the present time (t) is calculated using parameters (Dead time (DM), Gain (KM), and Time contrast (TM)) which indicate the internal process so that the process variable (PV) is to be the value of Reference model at the time (t+H) of Coincidence horizon (H).
This calculation is performed in every control cycle (CT).



PV: Process variable (%), MV: Manipulated variable (%), H: HORIZON (Coincidence Horizon), CT Control cycle

(2) Operation overview

FBs which perform the predictive functional control are subdivided into the following three types.

(a) Simple first order lag (PFC_SF)
Apply this FB when the process has a characteristic of the simple first order lag.

(b) Simple second order lag (PFC_SS)
Apply this FB when the process has a characteristic of the simple second order lag. However, it is limited to processes whose "Tza / Tiz" is 0.1 or less in the identification method of the PFC parameters be explained below.
When it is more than 0.1, it is recommended to use the simple first order lag (PFC_SF).

(c) Integral process (PFC_INT)
Apply this FB when the process has a characteristic of the integral process.

(3) Identification methods of the PFC parameters

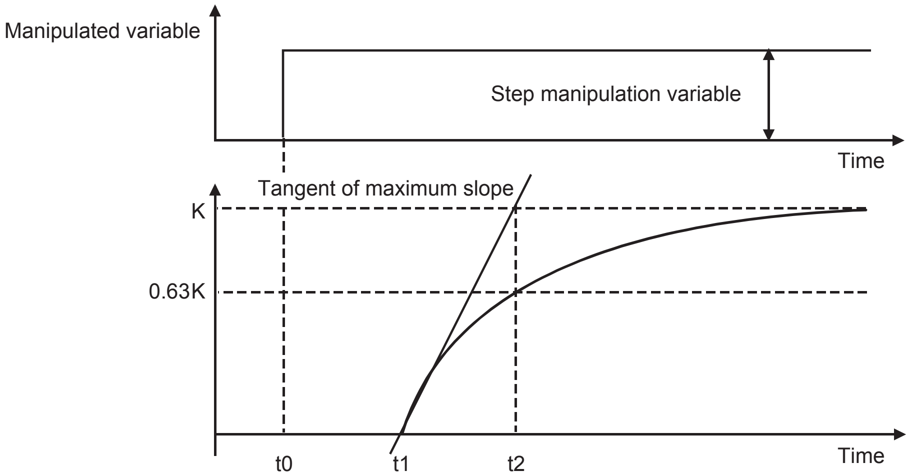
(a) Simple first order lag

Performs the step response operation to the process and calculate the parameters.

The parameters of the internal process are as follows when the step response shown below is achieved

- DM: Dead time
  DM = t1 - t0
  Set the control cycle (CT) to DM/128 or longer.
- KM: Gain
  KM = K
- TM: Time constant
  TM = t2 – t1

Supplement: If it is difficult to specify the maximum slope, assume the position of 0.63K as t2.



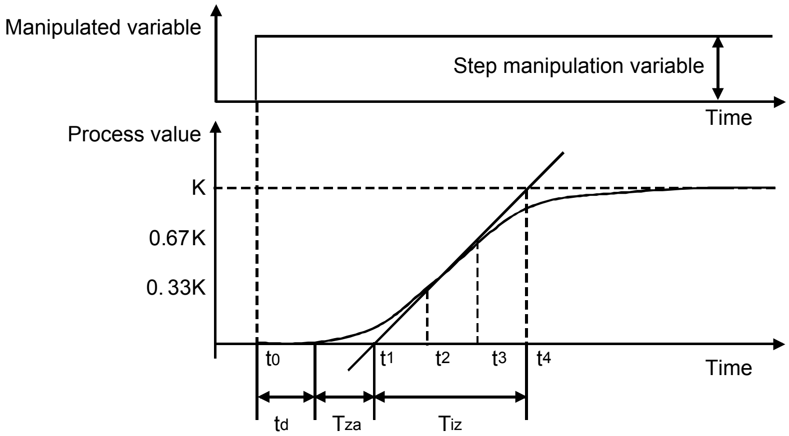
Fix the following parameters indicating points which are to be targets of the predictive value.

- TRBF: Reference model time constant
  Process with a short dead time (TM/DM>10):
  Approximately 1/10 of time constant
  Process with a long dead time (TM/DM<3):
  Approximately 1/5 of time constant
- H : Coincidence horizon
  Process with a short dead time (TM/DM>10):
  Approximately "Reference model time constant (TRBF) / Control cycle (CT) × 3/2"
  Process with a long dead time (TM/DM<3):
  Approximately "Reference model time constant (TRBF) / Control cycle (CT) × 4"

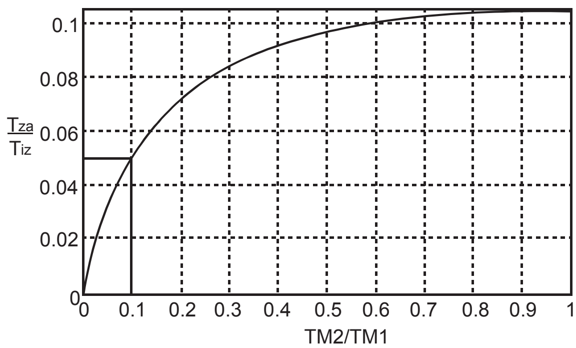
(b) Simple second order lag
Performs the step response operation to the process and calculate the parameters.
The parameters of the internal process are as follows when the step response shown below is achieved. For Reference model time constant (TRBF)*1 and Coincidence horizon (H), refer to (3) (a) in this section.
*1 Let TM1 + TM2 = TM to fix the parameter.

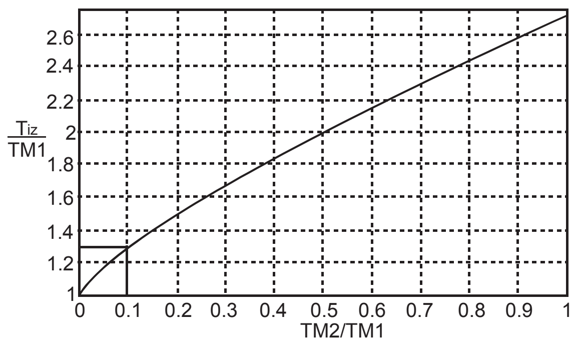
- DM: Dead time
  DM = td
  Set the control cycle (CT) to DM/128 or longer.
- KM: Gain
  KM = K
- TM: Time constant
  Calculate time constant TM1 and TM2 in the following method (Strejc method).
  1. Calculate a difference between K and the start point of the step response (in this case, 0). Then let t2, t3 respectively be the position at 33, 67 percent of the value, assuming the start point of the step response is t0.
  2. Calculate the slope of the tangent between t2 and t3.
  3. Let t1, t4 respectively be the point at the intersection of the tangent between t2 and t3 with the X axis, with the steady-state value K.
  4. Calculate Tza, Tiz respectively by Tza = t1 - (t0 + td), Tiz = t4 - t1.

  Derive time constant TM1 and TM2.



(Example) When t0 = 0, t1 = 28, td = 5, and t4 = 459
Tza = t1 - (t0 + td) = 23 Tiz = t4 - t1 = 431
Calculate TM1 and TM2 using the following charts.
Tza / Tiz ≒ 0.053 TM2 / TM1 ≒ 0.1 Tiz / TM1 ≒ 1.3
TM1 ≒ Tiz / 1.3 ≒ 331 TM2 = 0.1 × TM1 ≒ 33



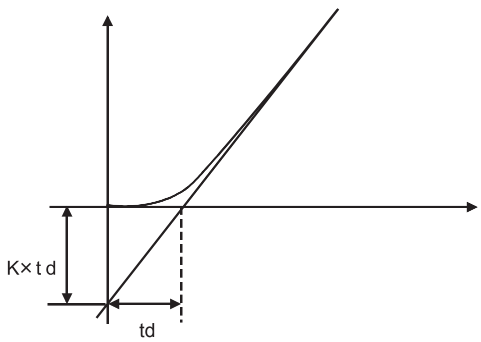


(c) Integral process

Performs the step response operation to the process and calculate the parameters.

The parameters of the internal process are as follows when the step response shown below is achieved.

- DM: Dead time
  DM = td
  Set the control cycle (CT) to DM/128 or longer.
- KM: Gain
  KM = K



Fix the following parameters indicating points which are to be targets of the predictive value.

- TRBF: Reference model time constant
  Quick response in which overshoot is permitted: 3 to 10 times the dead time
  Slow response in which overshoot is permitted: More than 10 times the dead time
- H : Coincidence horizon
  2 or more

(4) Tuning

In tuning the parameters, the step response firstly calculates the model values of the process (Dead time (DM), Gain (KM), and Time constant (TM)). For Reference model time constant (TRBF) and Coincidence horizon (H), refer to (3) in this section.

The following behaviors occur when the values show deviations from the model, so adjust the model values depending on the response of the process.

	Value is smaller than model value	Value is larger than model value	Influence
DM	•Time for reaching the setting value is faster. •Overshoots occur more commonly.	•Time for reaching the setting value is slower. •Overshoots occur less commonly.	Middle
KM	•Time for reaching the setting value is faster. •Overshoots occur more commonly. •The oscillation amplitude of the MV value is larger.	•Time for reaching the setting value is slower. •Overshoots occur less commonly. •The oscillation amplitude of the MV value is smaller.	Large
TM	•Time for reaching the setting value is slower. •The oscillation amplitude of the MV value is smaller.	•Time for reaching the setting value is faster. •The oscillation amplitude of the MV value is larger.	Small

For optimal tuning, adjust Reference model time constant (TRBF) and Coincidence horizon (H).

The following table shows behaviors of processes when the parameters are changed.

	Increase value	Decrease value	Influence
TRBF	•Time for reaching the setting value becomes slower. •Overshoots become to occur less commonly.	•Time for reaching the setting value becomes faster. •Overshoots become to occur more commonly.	Large
H	•The oscillation amplitude of the MV value becomes smaller.	•The oscillation amplitude of the MV value becomes larger.	Small

Appendix 3.19 Method for Using Tight Shut/Full Open Function (for module without extended mode in range setting)

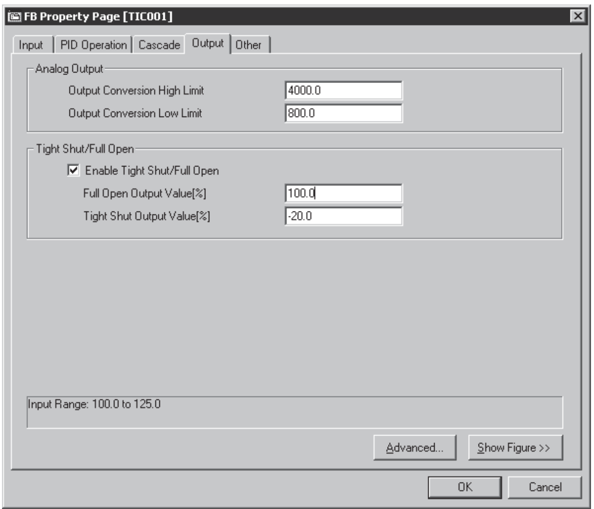
For module without extended mode in range setting, set to 0 to 20mA, 0 to 5V in range setting, and reset output conversion high/low limit value of 2-degree-of-freedom advanced PID control FB to enable tight shut/full open function.

POINT

When 4 to 20mA, 1 to 5V are regarded as a standard, only tight shut (full open when MV reverse is valid) is applied since upper limit side can output up to 100% (20mA, 5V) with output range setting of 0 to 20mA, 0 to 5V.

The following shows a setting example of 2-degree-of-freedom advanced PID control FB when the signal of 4 to 20mA is regarded as a standard, tight shut output value is set to -20% (0.8mA).

Output range setting: 0 to 20mA, resolution: 0 to 4000



Item	Set value	Remarks
Output Conversion High Limit	4000.0 (When setting to regard a digital value of analog module FB as percentage (%) in the Project Parameter Setting, set to 100.0, not depending on resolution)	Equivalent to 20mA
Output Conversion Low Limit	800.0 (When setting to regard a digital value of analog module FB as percentage (%) in the Project Parameter Setting, set to 20.0, not depending on resolution)	Equivalent to 4mA
Enable Tight Shut/Full Open	Checked	–
Full Open Output Value [%]	100.0	100%, since cannot output more than 20mA
Tight Shut Output Value [%]	-20.0	Equivalent to 0.8mA

Appendix 4 Approximate number of steps

The following shows the approximate number of steps of ladder program created by compiling FBD program.

- System control processing
  Such as initialization processing, tag data communication processing, program start scheduler processing
- Manufacturer library
  Function, manufacturer FB/tag FB, module FB
- Connector
  Value substitution processing by connecting output pins and input pins of FBD parts

Note that the approximate number of steps of manufacturer library includes the number of argument substitution and error processing.

> **POINT**
>
> When changing PLC type from Process CPU or Redundant CPU to Universal model process CPU, a program size may exceed its capacity if the program includes a function or FB whose number of steps is increased by the change.
>
> Therefore, check the program size by calculating its memory size with a GX application after changing PLC type.
>
> For memory size calculation, refer to the following manuals.
> - GX Works2 Version 1 Operating Manual (Common)
> - GX Developer Version 8 Operating Manual
>
> If the program size exceeds its capacity, consider taking the following corrective actions.
> - Review the program.
> - Select a model which has a larger program memory.
> - Use the file register of the system resource within the range less than 64K words.
>   (Using a file register whose size is less than 64K words can save one step in a program compared with using the one whose size has exceeded 64K words.)

(1) System control processing
Process CPU/Redundant CPU : Approximately 1100 steps
Universal model process CPU : Approximately 1400 steps

(2) General function

Classification	Function	Function name	Approximate number of steps *1			
			QnPH/ QnPRHCPU		QnUDPV CPU	
			MIN	MAX	MIN	MAX
Type conversion funcion	INT_TO_REAL	INT type → REAL type conversion	20	–	20	–
	INT_TO_REAL_E	INT type → REAL type conversion (with EN/ENO)	30	–	30	–
	DINT_TO_REAL	DINT type → REAL type conversion	20	–	20	–
	DINT_TO_REAL_E	DINT type → REAL type conversion (with EN/ENO)	30	–	30	–
	INT_TO_DINT	INT type → DINT type conversion	20	–	20	–
	INT_TO_DINT_E	INT type → DINT type conversion (with EN/ENO)	30	–	30	–
	DINT_TO_INT	DINT type → INT type conversion	30	–	30	–
	DINT_TO_INT_E	DINT type → INT type conversion (with EN/ENO)	40	–	40	–
	INT_TO_BCD	INT type → BCD type conversion	30	–	30	–
	INT_TO_BCD_E	INT type → BCD type conversion (with EN/ENO)	40	–	40	–
	DINT_TO_BCD	DINT type → BCD type conversion	30	–	30	–
	DINT_TO_BCD_E	DINT type → BCD type conversion (with EN/ENO)	40	–	40	–
	INT_TO_WORD	INT type → WORD type conversion	20	–	20	–
	INT_TO_WORD_E	INT type → WORD type conversion (with EN/ENO)	30	–	30	–
	DINT_TO_WORD	DINT type → WORD type conversion	20	–	20	–
	DINT_TO_WORD_E	DINT type → WORD type conversion (with EN/ENO)	30	–	40	–
	INT_TO_DWORD	INT type → DWORD type conversion	20	–	20	–
	INT_TO_DWORD_E	INT type → DWORD type conversion (with EN/ENO)	30	–	40	–
	DINT_TO_DWORD	DINT type → DWORD type conversion	20	–	20	–
	DINT_TO_DWORD_E	DINT type → DWORD type conversion (with EN/ENO)	30	–	30	–
	INT_TO_BOOL	INT type → BOOL type conversion	20	–	30	–
	INT_TO_BOOL_E	INT type → BOOL type conversion (with EN/ENO)	40	–	40	–
	DINT_TO_BOOL	DINT type → BOOL type conversion	30	–	30	–
	DINT_TO_BOOL_E	DINT type → BOOL type conversion (with EN/ENO)	40	–	40	–
	REAL_TO_INT	REAL type → INT type conversion	30	–	30	–
	REAL_TO_INT_E	REAL type → INT type conversion (with EN/ENO)	40	–	40	–
	REAL_TO_DINT	REAL type → DINT type conversion	30	–	30	–
	REAL_TO_DINT_E	REAL type → DINT type conversion (with EN/ENO)	40	–	40	–
	BCD_TO_INT	BCD type → INT type conversion	40	–	40	–
	BCD_TO_INT_E	BCD type → INT type conversion (with EN/ENO)	50	–	50	–
	BCD_TO_DINT	BCD type → DINT type conversion	40	–	40	–
	BCD_TO_DINT_E	BCD type → DINT type conversion (with EN/ENO)	50	–	50	–
	WORD_TO_INT	WORD type → INT type conversion	20	–	20	–
	WORD_TO_INT_E	WORD type → INT type conversion (with EN/ENO)	30	–	30	–
	WORD_TO_DINT	WORD type → DINT type conversion	20	–	20	–
	WORD_TO_DINT_E	WORD type → DINT type conversion (with EN/ENO)	30	–	40	–
	WORD_TO_BOOL	WORD type → BOOL type conversion	20	–	30	–
	WORD_TO_BOOL_E	WORD type → BOOL type conversion (with EN/ENO)	40	–	40	–
	WORD_TO_BOOL	WORD type → BOOL type conversion	20	–	30	–
	WORD_TO_BOOL_E	WORD type → BOOL type conversion (with EN/ENO)	40	–	40	–

*1 The approximate number of steps is indicated when all input/output pins are connected
When the number of input pins is changeable function, the approximate number of steps for the minimum and maximum number of input pins is indicated.
For functions whose number of input pins is fixed, the approximate number of steps is indicated on the column of Minimum.

Classification	Function	Function name	Approximate number of steps *1			
			QnPH/ QnPRHCPU		QnUDPV CPU	
			MIN	MAX	MIN	MAX
Type conversion function (continued)	WORD_TO_BOOL	WORD type → BOOL type conversion	20	–	30	–
	WORD_TO_BOOL_E	WORD type → BOOL type conversion (with EN/ENO)	40	–	40	–
	DWORD_TO_BOOL	DWORD type → BOOL type conversion	30	–	30	–
	DWORD_TO_BOOL_E	DWORD type → BOOL type conversion (with EN/ENO)	40	–	40	–
	DWORD_TO_INT	DWORD type → INT type conversion	20	–	20	–
	DWORD_TO_INT_E	DWORD type → INT type conversion (with EN/ENO)	30	–	40	–
	DWORD_TO_DINT	DWORD type → DINT type conversion	20	–	20	–
	DWORD_TO_DINT_E	DWORD type → DINT type conversion (with EN/ENO)	30	–	30	–
	WORD_TO_DWORD	WORD type → DWORD type conversion	20	–	20	–
	WORD_TO_DWORD_E	WORD type → DWORD type conversion (with EN/ENO)	30	–	40	–
	DWORD_TO_WORD	DWORD type → WORD type conversion	20	–	20	–
	DWORD_TO_WORD_E	DWORD type → WORD type conversion (with EN/ENO)	30	–	40	–
	INT_TO_STRING	INT type → STRING type conversion	40	–	50	–
	INT_TO_STRING_E	INT type → STRING type conversion (with EN/ENO)	50	–	60	–
	DINT_TO_STRING	DINT type → STRING type conversion	40	–	50	–
	DINT_TO_STRING_E	DINT type → STRING type conversion (with EN/ENO)	50	–	60	–
	REAL_TO_STRING	REAL type → STRING type (exponent form) conversion	60	–	70	–
	REAL_TO_STRING_E	REAL type → STRING type (exponent form) conversion (with EN/ENO)	70	–	80	–
	REAL_TO_STRING_EX	REAL type → STRING type (decimal point form) conversion	60	–	70	–
	REAL_TO_STRING_EX_E	REAL type → STRING type (decimal point form) conversion (with EN/ENO)	70	–	80	–
	STRING_TO_INT	STRING type → INT type conversion	30	–	30	–
	STRING_TO_INT_E	STRING type → INT type conversion (with EN/ENO)	40	–	40	–
	STRING_TO_DINT	STRING type → DINT type conversion	30	–	30	–
	STRING_TO_DINT_E	STRING type → DINT type conversion (with EN/ENO)	40	–	40	–
	STRING_TO_REAL	STRING type → REAL type conversion	30	–	30	–
	STRING_TO_REAL_E	STRING type → REAL type conversion (with EN/ENO)	40	–	40	–
	BOOL_TO_INT	BOOL type → INT type conversion	20	–	30	–
	BOOL_TO_INT_E	BOOL type → INT type conversion (with EN/ENO)	40	–	40	–

*1 The approximate number of steps is indicated when all input/output pins are connected
When the number of input pins is changeable function, the approximate number of steps for the minimum and maximum number of input pins is indicated.
For functions whose number of input pins is fixed, the approximate number of steps is indicated on the column of Minimum.

Classification	Function	Function name	Approximate number of steps *1			
			QnPH/ QnPRHCPU		QnUDPV CPU	
			MIN	MAX	MIN	MAX
Type conversion function (continued)	BOOL_TO_DINT	BOOL type → DINT type conversion	30	–	30	–
	BOOL_TO_DINT_E	BOOL type → DINT type conversion (with EN/ENO)	40	–	40	–
	BOOL_TO_WORD	BOOL type → WORD type conversion	20	–	30	–
	BOOL_TO_WORD_E	BOOL type → WORD type conversion (with EN/ENO)	40	–	40	–
	BOOL_TO_DWORD	BOOL type → DWORD type conversion	30	–	30	–
	BOOL_TO_DWORD_E	BOOL type → DWORD type conversion (with EN/ENO)	40	–	40	–
Numerical operation function	ABS	Absolute value	40	–	40	–
	ABS_E	Absolute value (with EN/ENO)	50	–	50	–
	SQRT	Square root	30	–	30	–
	SQRT_E	Square root (with EN/ENO)	40	–	40	–
	LN	Natural logarithm	30	–	30	–
	LN_E	Natural logarithm (with EN/ENO)	40	–	40	–
	LOG	Common logarithm	30	–	40	–
	LOG_E	Common logarithm (with EN/ENO)	50	–	50	–
	EXP	Natural exponential	30	–	30	–
	EXP_E	Natural exponential (with EN/ENO)	40	–	40	–
	SIN	SIN operation	30	–	30	–
	SIN_E	SIN operation (with EN/ENO)	40	–	40	–
	COS	COS operation	30	–	30	–
	COS_E	COS operation (with EN/ENO)	40	–	40	–
	TAN	TAN operation	30	–	30	–
	TAN_E	TAN operation (with EN/ENO)	40	–	40	–
	ASIN	ASIN operation	30	–	30	–
	ASIN_E	ASIN operation (with EN/ENO)	40	–	40	–
	ACOS	ACOS operation	30	–	30	–
	ACOS_E	ACOS operation (with EN/ENO)	40	–	40	–
	ATAN	ATAN operation	30	–	30	–
	ATAN_E	ATAN operation (with EN/ENO)	40	–	40	–
	NEG_	Sign reversal	30	–	30	–
	NEG_E_	Sign reversal (with EN/ENO)	40	–	40	–
Arithmetic operation function	ADD	Addition	30	50	30	60
	ADD_E	Addition (with EN/ENO)	40	60	40	70
	MUL	Multiplication	30	50	30	60
	MUL_E	Multiplication (with EN/ENO)	40	70	40	70
	SUB	Subtraction	30	–	30	–
	SUB_E	Subtraction (with EN/ENO)	40	–	40	–
	DIV	Division	30	–	30	–
	DIV_E	Division (with EN/ENO)	40	–	40	–
	MOD	Modulus operation	30	–	30	–
	MOD_E	Modulus operation (with EN/ENO)	50	–	50	–

*1 The approximate number of steps is indicated when all input/output pins are connected
When the number of input pins is changeable function, the approximate number of steps for the minimum and maximum number of input pins is indicated.
For functions whose number of input pins is fixed, the approximate number of steps is indicated on the column of Minimum.

Classification	Function	Function name	Approximate number of steps *1			
			QnPH/ QnPRHCPU		QnUDPV CPU	
			MIN	MAX	MIN	MAX
Arithmetic operation function (continued)	POW_	Exponentiation	60	－	70	－
	POW_E_	Exponentiation (with EN/ENO)	70	－	80	－
	MOVE_E_	Transfer (with EN/ENO)	30	－	30	－
Bit-string function	SHL	Shift left	90	－	100	－
	SHL_E	Shift left (with EN/ENO)	100	－	110	－
	SHR	Shift right	60	－	70	－
	SHR_E	Shift right (with EN/ENO)	70	－	80	－
	ROL	Rotate left	20	－	20	－
	ROL_E	Rotate left (with EN/ENO)	30	－	40	－
	ROR	Rotate right	20	－	20	－
	ROR_E	Rotate right (with EN/ENO)	30	－	40	－
Logical operation function	AND	AND	20	50	20	40
	AND_E	AND (with EN/ENO)	30	60	30	50
	OR	OR	20	50	20	40
	OR_E	OR (with EN/ENO)	30	60	30	50
	XOR	XOR	20	50	20	40
	XOR_E	XOR (with EN/ENO)	30	60	30	50
	NOT	NOT	20	－	20	－
	NOT_E	NOT (with EN/ENO)	30	－	30	－
Selection function	SEL	Input value selection	30	－	30	－
	SEL_E	Input value selection (with EN/ENO)	40	－	40	－
	MAX	Maximum value selection	40	80	30	70
	MAX_E	Maximum value selection (with EN/ENO)	50	90	50	80
	MIN	Minimum value selection	40	80	30	70
	MIN_E	Minimum value selection (with EN/ENO)	50	90	50	80
	LIMIT	High/Low limit control	50	－	50	－
	LIMIT_E	High/Low limit control (with EN/ENO)	70	－	60	－
	MUX	Multiplexer	40	80	40	80
	MUX_E	Multiplexer (with EN/ENO)	60	100	60	90
Comparison function	>	Comparison	30	50	40	50
	>_E	Comparison (with EN/ENO)	50	60	50	70
	>=	Comparison	30	50	40	50
	>=_E	Comparison (with EN/ENO)	50	60	50	70
	=	Comparison	30	50	40	50
	=_E	Comparison (with EN/ENO)	50	60	50	70
	<=	Comparison	30	50	40	50
	<=_E	Comparison (with EN/ENO)	50	60	50	70
	<	Comparison	30	50	40	50
	<_E	Comparison (with EN/ENO)	50	60	50	70
	<>	Comparison	30	－	40	－
	<>_E	Comparison (with EN/ENO)	50	－	50	－

*1 The approximate number of steps is indicated when all input/output pins are connected
When the number of input pins is changeable function, the approximate number of steps for the minimum and maximum number of input pins is indicated.
For functions whose number of input pins is fixed, the approximate number of steps is indicated on the column of Minimum.

Classification	Function	Function name	Approximate number of steps *1			
			QnPH/ QnPRHCPU		QnUDPV CPU	
			MIN	MAX	MIN	MAX
Character string function	LEN	String length	30	–	30	–
	LEN_E	String length (with EN/ENO)	40	–	40	–
	LEFT	Leftmost characters	50	–	60	–
	LEFT_E	Leftmost characters (with EN/ENO)	60	–	70	–
	RIGHT	Rightmost characters	50	–	60	–
	RIGHT_E	Rightmost characters (with EN/ENO)	60	–	70	–
	MID	Middle characters	50	–	60	–
	MID_E	Middle characters (with EN/ENO)	70	–	80	–
	CONCAT	Concatenation	70	–	80	–
	CONCAT_E	Concatenation (with EN/ENO)	80	–	100	–
	INSERT	Inserting characters	90	–	110	–
	INSERT_E	Inserting characters (with EN/ENO)	110	–	130	–
	DELETE	Deleting substring	70	–	80	–
	DELETE_E	Deleting substring (with EN/ENO)	80	–	100	–
	REPLACE	Replacing characters	60	–	70	–
	REPLACE_E	Replacing characters (with EN/ENO)	70	–	80	–
	FIND	Finding characters	30	–	30	–
	FIND_E	Finding characters (with EN/ENO)	40	–	50	–
Helper function	UNBIND	WORD → 16BOOL unbinding	180	–	210	–
	UNBIND_E	WORD → 16BOOL unbinding (with EN/ENO)	200	–	240	–
	BIND	16 BOOL → WORD/DWORD	70	–	90	–
	BIND_E	16 BOOL → WORD/DWORD (with EN/ENO)	90	–	110	–
	MAKE_DWORD	2WORD → DWORD	20	–	30	–
	MAKE_DWORD_E	2WORD → DWORD (with EN/ENO)	40	–	40	–
	HI_WORD	High-order output of DWORD type data	20	–	20	–
	HI_WORD_E	High-order output of DWORD type data (with EN/ENO)	30	–	40	–
	LO_WORD	Low-order output of DWORD type data	20	–	20	–
	LO_WORD_E	Low-order output of DWORD type data (with EN/ENO)	30	–	40	–
	IS_CONNECTED_	Input pins connection status acquisition	70	–	80	–
	IS_CONNECTED_E_	Input pins connection status acquisition (with EN/ENO)	90	–	100	–
Ladder program control function	CALL_DINT	Sub-routine program call (DINT type argument)	90	–	90	–
	CALL_DINT_E	Sub-routine program call (DINT type argument) (with EN/ENO)	110	–	110	–
	CALL_REAL	Sub-routine program call (REAL type argument)	90	–	90	–
	CALL_REAL_E	Sub-routine program call (REAL type argument) (with EN/ENO)	110	–	110	–
	PSCAN	Program scan execution registration	20	–	30	–
	PSCAN_E	Program scan execution registration (with EN/ENO)	40	–	30	–

*1 The approximate number of steps is indicated when all input/output pins are connected
When the number of input pins is changeable function, the approximate number of steps for the minimum and maximum number of input pins is indicated.
For functions whose number of input pins is fixed, the approximate number of steps is indicated on the column of Minimum.

Classification	Function	Function name	Approximate number of steps *1			
			QnPH/ QnPRHCPU		QnUDPV CPU	
			MIN	MAX	MIN	MAX
Ladder program control function (continued)	PSTOP	Program standby instruction	20	–	30	–
	PSTOP_E	Program standby instruction (with EN/ENO)	40	–	30	–
	POFF	Program output standby instruction	20	–	30	–
	POFF_E	Program output standby instruction (with EN/ENO)	40	–	30	–
	PLOW	Program low-speed execution registration	20	–	–	–
	PLOW_E	Program low-speed execution registration (with EN/ENO)	40	–	–	–

*1 The approximate number of steps is indicated when all input/output pins are connected
When the number of input pins is changeable function, the approximate number of steps for the minimum and maximum number of input pins is indicated.
For functions whose number of input pins is fixed, the approximate number of steps is indicated on the column of Minimum.

(3) General FB

Classification	FB	FB name	Approximate number of steps *1	
			QnPH/ QnPRHCPU	QnUDPV CPU
Bistable FB	SR	Set-dominant flip-flop	70	80
	RS	Reset-dominant flip-flop	70	80
	LATCH_BOOL	Latch FB (BOOL type)	70	80
	LATCH_REAL	Latch FB (REAL type)	70	80
	LATCH_WORD	Latch FB (WORD type)	70	80
	LATCH_DWORD	Latch FB (DWORD type)	70	80
Edge detection FB	R_TRIG	Rising edge detector	60	70
	F_TRIG	Falling edge detector	60	80
	EDGE_CHECK	Edge detection input	80	90
Counter FB	CTU	Up-counter	110	140
	CTD	Down-counter	110	140
	CTUD	Up-down-counter	160	220
Timer FB	TP_HIGH	Pulse timer (high-speed timer)	100	120
	TP_LOW	Pulse timer (low-speed timer)	100	120
	TON_HIGH	ON delay timer (high-speed timer)	90	110
	TON_LOW	ON delay timer (low-speed timer)	90	110
	TOF_HIGH	OFF delay timer (high-speed timer)	90	120
	TOF_LOW	OFF delay timer (low-speed timer)	90	120
Communication control FB	SEND	Sending data to PLC CPUs of other stations	160	190
	RECV	Receiving data from PLC CPUs of other stations	140	170

*1 The approximate number of steps is indicated when all input/output pins are connected
Indicate the number of steps when the first FB is pasted. When the same type of FB is pasted 2 or more, the approximate number of steps for the second and later is increased for the amount of common subroutine processing calls (approximately 30 to 80 steps).

(4) Process function

Classification	Function	Function name	Approximate number of steps *1			
			QnPH/QnPRHCPU		QnUDPV CPU	
			MIN	MAX	MIN	MAX
Analog value selection and average value function	P_HS	High selector	50	90	60	100
	P_HS_E	High selector (with EN/ENO)	60	100	70	110
	P_LS	Low selector	50	90	60	100
	P_LS_E	Low selector (with EN/ENO)	60	100	70	110
	P_MID	Middle value selection	50	90	60	100
	P_MID_E	Middle value selection (with EN/ENO)	60	110	70	110
	P_AVE	Average value	50	90	50	90
	P_AVE_E	Average value (with EN/ENO)	60	100	60	100
	P_ABS	Absolute value	50	—	50	—
	P_ABS_E	Absolute value (with EN/ENO)	60	—	60	—

*1 The approximate number of steps is indicated when all input/output pins are connected
When the number of input pins is changeable function, the approximate number of steps for the minimum and maximum number of input pins is indicated.
For functions whose number of input pins is fixed, the approximate number of steps is indicated on the column of Minimum.

(5) Process FB_General process FB

Classification	FB	FB name	Approximate number of steps *1	
			QnPH/QnPRHCPU	QnUDPV CPU
Correction operation FB	P_FG	Function generator	80	80
	P_IFG	Inverse function generator	80	80
	P_FLT	Standard filter (Moving average)	80	80
	P_ENG	Engineering value conversion	70	80
	P_IENG	Inverse engineering value conversion	70	80
	P_TPC	Temperature/Pressure correction	90	100
	P_SUM	Summation	80	90
	P_SUM2_	Summation (internal integer integration)	430	580
	P_RANGE_	Range conversion	160	200
Arithmetic operation FB	P_ADD	Addition (with coefficient)	100	100
	P_SUB	Subtraction (with coefficient)	100	100
	P_MUL	Multiplication (with coefficient)	100	100
	P_DIV	Division (with coefficient)	80	80
	P_SQR	Square root (with coefficient)	70	80
Comparison operation FB	P_>	Compare greater than (with setting value)	80	90
	P_<	Compare less than (with setting value)	80	90
	P_=	Compare equal than (with setting value)	80	90
	P_>=	Compare greater or equal (with setting value)	80	90
	P_<=	Compare less or equal (with setting value)	80	90

*1 The approximate number of steps is indicated when all input/output pins are connected
Indicate the number of steps when the first FB is pasted. When the same type of FB is pasted 2 or more, the approximate number of steps for the second and later is increased for the amount of common subroutine processing calls (approximately 30 to 80 steps).

Classification	FB	FB name	Approximate number of steps *1	
			QnPH/ QnPRHCPU	QnUDPV CPU
Control operation FB	P_LLAG	Lead-lag	90	90
	P_I	Integral	80	80
	P_D	Derivative	90	90
	P_DED	Dead time	160	180
	P_LIMT	High/Low limiter	90	90
	P_VLMT1	Variation rate limiter1	90	90
	P_VLMT2	Variation rate limiter2	90	90
	P_DBND	Dead band	80	90
	P_BUMP	Bumpless transfer	90	100
	P_AMR	Analog memory	100	110
	P_DUTY_8PT_	8-points time proportioning output	720	860

*1 The approximate number of steps is indicated when all input/output pins are connected
Indicate the number of steps when the first FB is pasted. When the same type of FB is pasted 2 or more, the approximate number of steps for the second and later is increased for the amount of common subroutine processing calls (approximately 30 to 80 steps).

(6) Process FB_Tag access FB

Classification	FB	FB name	Approximate number of steps *1	
			QnPH/ QnPRHCPU	QnUDPV CPU
I/O control operation FB	P_IN	Analog input processing	180	210
	P_OUT1	Output processing-1 with mode switching (with input addition)	180	200
	P_OUT2	Output processing-2 with mode switching (without input addition)	160	180
	P_OUT3_	Output processing-3 with mode switching (with input addition and compensation)	1250	1740
	P_MOUT	Manual output	90	90
	P_DUTY	Time proportioning output	170	190
	P_PSUM	Pulse integration	270	350
	P_BC	Batch counter	140	160
	P_MSET_	Manual setter	770	1010
Loop control operation FB	P_R_T	Ratio control (with tracking to primary loop)	150	160
	P_R	Ratio control (without tracking to primary loop)	150	160
	P_PID_T	Velocity type PID control (with tracking to primary loop)	310	370
	P_PID	Velocity type PID control (without tracking to primary loop)	310	370
	P_2PID_T	2-degree-of-freedom PID control (with tracking to primary loop)	310	370
	P_2PID	2-degree-of-freedom PID control (without tracking to primary loop)	310	370
	P_2PIDH_T_	2-degree-of-freedom advanced PID control (with tracking to primary loop)	1930	2670
	P_2PIDH_	2-degree-of-freedom advanced PID control (without tracking to primary loop)	1900	2630

*1 The approximate number of steps is indicated when all input/output pins are connected
Indicate the number of steps when the first FB is pasted. When the same type of FB is pasted 2 or more, the approximate number of steps for the second and later is increased for the amount of common subroutine processing calls (approximately 30 to 80 steps).

Classification	FB	FB name	Approximate number of steps *1	
			QnPH/ QnPRHCPU	QnUDPV CPU
Loop control operation FB	P_PIDP_T	Position type PID control (with tracking to primary loop, without tracking from secondary loop)	230	250
	P_PIDP	Position type PID control (without tracking to primary loop, without tracking from secondary loop)	230	250
	P_PIDP_EX_T_	Position type PID control (with tracking to primary loop, with tracking from secondary loop)	260	290
	P_PIDP_EX_	Position type PID control (without tracking to primary loop, with tracking from secondary loop)	260	290
	P_SPI_T	Sample PI control (with tracking to primary loop)	150	160
	P_SPI	Sample PI control (without tracking to primary loop)	160	170
	P_IPD_T	I-PD control (with tracking to primary loop)	150	160
	P_IPD	I-PD control (without tracking to primary loop)	160	170
	P_BPI_T	Blend PI control (with tracking to primary loop)	210	250
	P_BPI	Blend PI control (without tracking to primary loop)	220	250
	P_PHPL	High/Low limit alarm check	150	190
	P_ONF2_T	2 position ON/OFF (with tracking to primary loop)	190	210
	P_ONF2	2 position ON/OFF (without tracking to primary loop)	200	210
	P_ONF3_T	3 position ON/OFF (with tracking to primary loop)	200	220
	P_ONF3	3 position ON/OFF (without tracking to primary loop)	210	230
	P_PGS	Program setter	110	120
	P_PGS2_	Multi-point program setter	2000	2720
	P_SEL	Loop selector (without tracking to primary loop)	170	180
	P_SEL_T1	Loop selector (with Tracking to primary loop)	190	210
	P_SEL_T2	Loop selector (with Tracking to primary loop)	170	180
	P_SEL_T3_	Loop selector (with tracking from secondary loop to primary loop)	670	880
	P_PFC_SF_	Predictive Functional Control (Simple First Order Lag)	1660	2240
	P_PFC_SS_	Predictive Functional Control (Simple Second Order Lag)	2020	2740
	P_PFC_INT_	Predictive Functional Control (Integral Process)	1660	2270
Tag special FB	P_MCHG	Control mode change	190	240

*1 The approximate number of steps is indicated when all input/output pins are connected
Indicate the number of steps when the first FB is pasted. When the same type of FB is pasted 2 or more, the approximate number of steps for the second and later is increased for the amount of common subroutine processing calls (approximately 30 to 80 steps).

(7) Process FB_Tag FB

Classification	FB	FB name	Approximate number of steps *1	
			QnPH/ QnPRHCPU	QnUDPV CPU
Loop tag FB	M_PID_T	Velocity type PID control (with tracking to primary loop)	890	1090
	M_PID	Velocity type PID control (without tracking to primary loop)	900	1100
	M_PID_DUTY_T	Velocity type PID control and duty output (with tracking to primary loop)	880	1080
	M_PID_DUTY	Velocity type PID control and duty output (without tracking to primary loop)	880	1090
	M_2PID_T	2-degree-of-freedom PID control (with tracking to primary loop)	890	1090
	M_2PID	2-degree-of-freedom PID control (without tracking to primary loop)	900	1100
	M_2PID_DUTY_T	2-degree-of-freedom PID control and duty output (with tracking to primary loop)	880	1080
	M_2PID_DUTY	2-degree-of-freedom PID control and duty output (without tracking to primary loop)	880	1090
	M_2PIDH_T_	2-degree-of-freedom advanced PID control (with tracking to primary loop)	4520	6060
	M_2PIDH_	2-degree-of-freedom advanced PID control (without tracking to primary loop)	4490	6020
	M_PIDP_T	Position type PID control (with tracking to primary loop, without tracking from secondary loop)	660	800
	M_PIDP	Position type PID control (without tracking to primary loop, without tracking from secondary loop)	660	800
	M_PIDP_EX_T_	Position type PID control (with tracking to primary loop, with tracking from secondary loop)	680	830
	M_PIDP_EX_	Position type PID control (without tracking to primary loop, with tracking from secondary loop)	690	840
	M_SPI_T	Sample PI control (with tracking to primary loop)	730	890
	M_SPI	Sample PI control (without tracking to primary loop)	740	890
	M_IPD_T	I-PD control (with tracking to primary loop)	730	890
	M_IPD	I-PD control (without tracking to primary loop)	740	890
	M_BPI_T	Blend PI control (with tracking to primary loop)	790	970
	M_BPI	Blend PI control (without tracking to primary loop)	800	980
	M_R_T	Ratio control (with tracking to primary loop)	720	860
	M_R	Ratio control (without tracking to primary loop)	720	860
	M_ONF2_T	2 position ON/OFF control (with tracking to primary loop)	610	750
	M_ONF2	2 position ON/OFF control (without tracking to primary loop)	620	750
	M_ONF3_T	3 position ON/OFF control (with tracking to primary loop)	620	760
	M_ONF3	3 position ON/OFF control (without tracking to primary loop)	620	760
	M_MONI	Monitor	310	380
	M_MWM	Manual output with monitor	520	640
	M_BC	Batch preparation	390	500
	M_PSUM	Pulse integrator	240	310

*1 The approximate number of steps is indicated when all input/output pins are connected
Indicate the number of steps when the first FB is pasted. When the same type of FB is pasted 2 or more, the approximate number of steps for the second and later is increased for the amount of common subroutine processing calls (approximately 30 to 80 steps).

Classification	FB	FB name	Approximate number of steps *1	
			QnPH/ QnPRHCPU	QnUDPV CPU
Loop tag FB (continued)	M_SEL	Loop selector (without tracking to primary loop)	300	370
	M_SEL_T1	Loop selector (with tracking to primary loop)	320	400
	M_SEL_T2	Loop selector (with tracking to primary loop)	300	370
	M_SEL_T3_	Loop selector (with tracking from secondary loop to primary loop)	800	1070
	M_MOUT	Manual output	240	300
	M_PGS	Program setter	260	320
	M_PGS2_	Multi-point program setter	2120	2880
	M_SWM_	Manual setter with monitor	1190	1560
	M_PFC_SF_	Predictive functional control (simple first order Lag)	2090	2800
	M_PFC_SS_	Predictive functional control (simple second order lag)	2450	3290
	M_PFC_INT_	Predictive functional control (integral process)	2090	2820
	M_PVAL_T_	Position-proportional output	2090	2910
	M_HTCL_T_	Heating and cooling output	1630	2250
Status tag FB	M_NREV	Motor irreversible (2 input, 2 output)	670	930
	M_REV	Motor reversible (2 input, 3 output)	820	1160
	M_MVAL1	ON/OFF operation (2 input, 2 output)	700	980
	M_MVAL2	ON/OFF operation (2 input, 3 output)	810	1150
	M_TIMER1	Timer 1 (timer stops when COMPLETE flag is ON)	420	580
	M_TIMER2	Timer 2 (timer continues when COMPLETE flag is ON)	410	570
	M_COUNTER1	Counter 1 (counter stops when COMPLETE flag is ON)	370	520
	M_COUNTER2	Counter 2 (counter continues when COMPLETE flag is ON)	370	510
	M_PB_	Push button operation (5 input, 5 output)	590	800
Alarm tag FB	M_ALARM	Alarm	110	120
	M_ALARM_64PT_	64-points alarm	60	70
Message tag FB	M_MESSAGE	Message	110	120
	M_MESSAGE_64PT_	64-points message	60	70

*1 The approximate number of steps is indicated when all input/output pins are connected
Indicate the number of steps when the first FB is pasted. When the same type of FB is pasted 2 or more, the approximate number of steps for the second and later is increased for the amount of common subroutine processing calls (approximately 30 to 80 steps).

(8) Module FB

Classification	FB	FB name	Approximate number of steps *1	
			QnPH/ QnPRH CPU	QnUDPV CPU
Module FB	AIN_4CH	4 channels analog input	380	520
	AIN_8CH	8 channels analog input	420	570
	AIN_4CH_G	Channel-isolated 4 channels analog input	460	650
	AIN_8CH_G	Channel-isolated 8 channels analog input	730	1030
	AIN_2CH_DG	Channel-isolated high-resolution 2 channels signal condition function	390	550
	AIN_6CH_DG	Channel-isolated 6 channels A/D converter module with signal conditioning function	660	940
	AOUT_2CH	2 channels analog output	330	470
	AOUT_4CH	4 channels analog output	380	530
	AOUT_8CH	8 channels analog output	520	700
	AOUT_2CH_G	Channel-isolated 2 channels analog output	560	800
	AOUT_6CH_G	Channel-isolated 6 channels analog output	590	820
	AIN_4CH_AOUT_2CH	Analog input/output (input 4 channels, output 2 channels)	1420	2070
	CT_8CH	8 channels CT Input	670	940
	TC_4CH	4 channels thermocouple input	340	460
	TC_8CH_G	Channel-isolated 8 channels thermocouple input	490	670
	TCV_4CH_G	Channel-isolated 4 channels temperature/micro-voltage input	410	550
	RTD_4CH	4 channels temperature-measuring resistor input	340	480
	RTD_8CH_G	Channel-isolated 8 channels temperature-measuring resistor input	490	670
	HIC_2CH	High-speed counter	490	670
	PIN_8CH_G	Channel-isolated 8 channels pulse input	1670	2210
	DIN_8PT	8 points digital input	100	110
	DIN_16PT	16 points digital input	160	180
	DIN_32PT	32 points digital input	270	310
	DIN_64PT	64 points digital input	510	580
	DOUT_8PT	8 points digital output	130	170
	DOUT_16PT	16 points digital output	190	230
	DOUT_32PT	32 points digital output	300	360
	DOUT_64PT	64 points digital output	570	680
	DINOUT_64PT	32 points input/32 points output I/O mixed	540	630
	DINOUT_15PT	8 points input/7 points output I/O mixed	190	230
	CCLINK_1	CC-Link remote station occupying 1 station	300	380
	CCLINK_2	CC-Link remote station occupying 2 station	500	640
	CCLINK_3	CC-Link remote station occupying 3 station	690	890
	CCLINK_4	CC-Link remote station occupying 4 station	890	1140

*1 The approximate number of steps is indicated when all input/output pins are connected
Indicate the number of steps when the first FB is pasted. When the same type of FB is pasted 2 or more, the approximate number of steps for the second and later is increased for the amount of common subroutine processing calls (approximately 30 to 80 steps).

(9) Connector

Data type	Approximate number of steps *1	
	QnPH/ QnPRHCPU	QnUDPVCPU
INT	6	6
DINT	6	6
REAL	6	6
STRING (20)	28	34
BOOL	15	12
WORD	6	6
DWORD	6	6
ADR_REAL	6	6

*1 The number of steps differs depending on the type of assigned devices of variables. Approximate number of steps indicates the maximum number of steps.

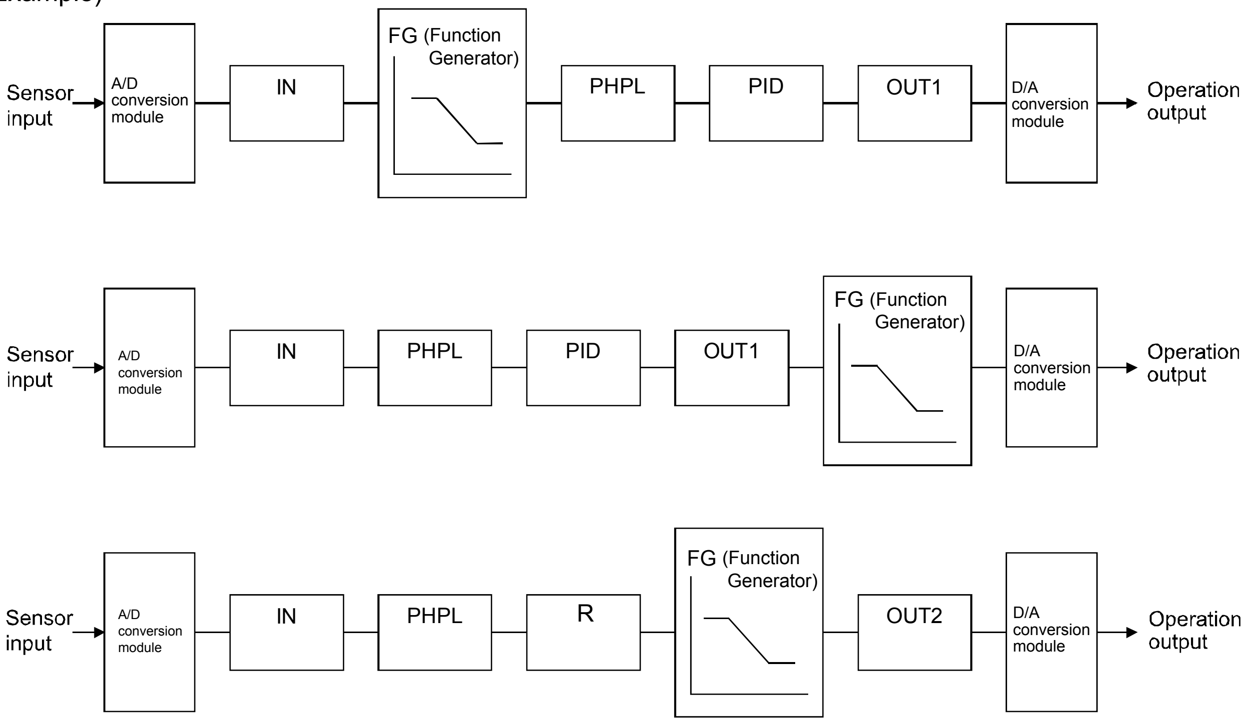
Appendix 5 Terms

The contents explain the process-related technical words.

Broken line correction

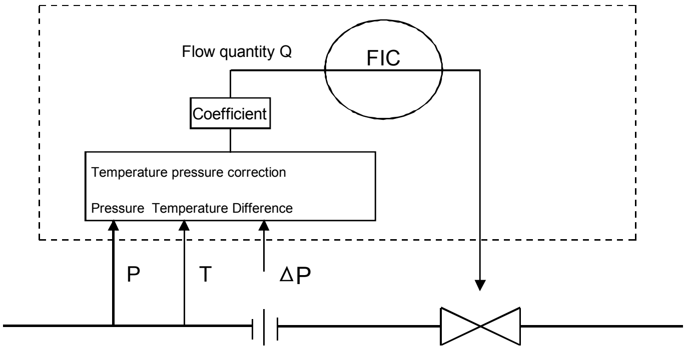
It is used when the value from the process target is not in proportion to process variable from sensor. Input value is approximated and corrected by broken line.

(Example)



Temperature/Pressure Correction

When the fluid conditions (temperature, pressure), of which the differential pressure measured by equipments which has diagram such as orifice, are not the same as the design conditions, it shall be corrected.
Correction shall be performed by process variable to multiply the temperature/pressure correction coefficient.
In addition, when equipments with diaphragm such as orifice are used, the obtained value is square of the flow quantity. So that extraction of square root shall be applied.



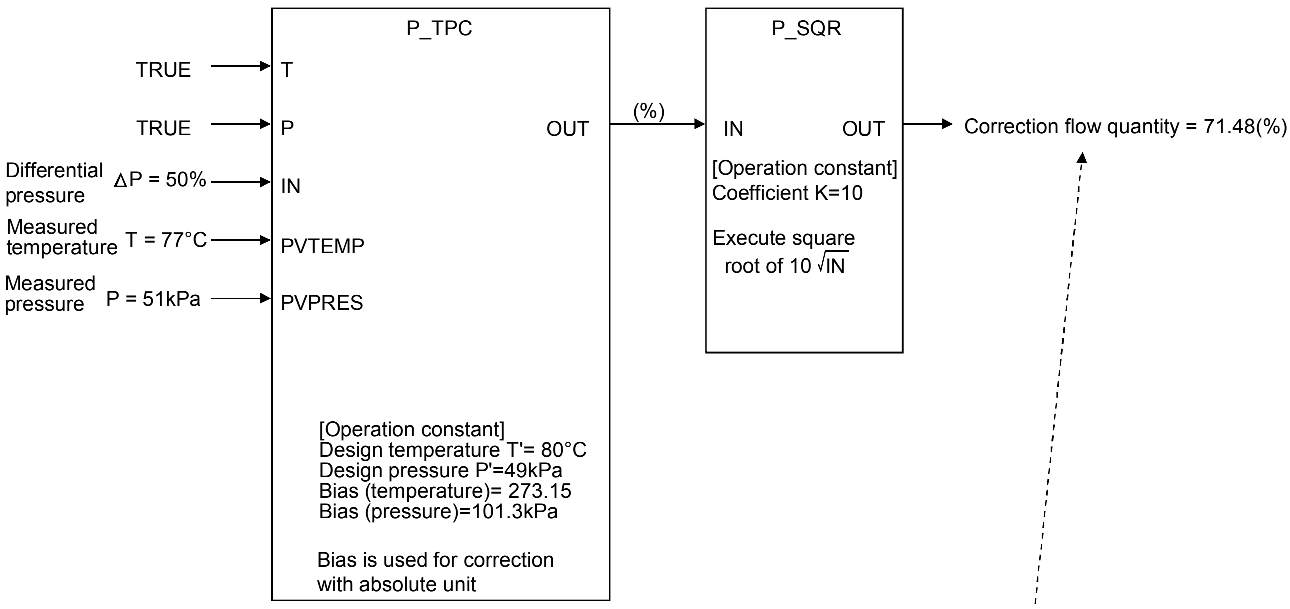
$$\text{Flow quantity Q} = \text{Temperature correction} \times \text{Pressure correction} \times \text{Coefficient}\sqrt{\text{Differential pressure}}$$

$$= \sqrt{\frac{\text{Design temperature}}{\text{Measured temperature}}} \times \sqrt{\frac{\text{Measured pressure}}{\text{Design pressure}}} \times \text{Coefficient}\sqrt{\text{Differential pressure}}$$

$$= \sqrt{\frac{T'}{T}} \times \sqrt{\frac{P}{P'}} \times \text{Coefficient}\sqrt{\Delta P}$$

T,T': Absolute temperature
P,P': Absolute pressure

Operation Example

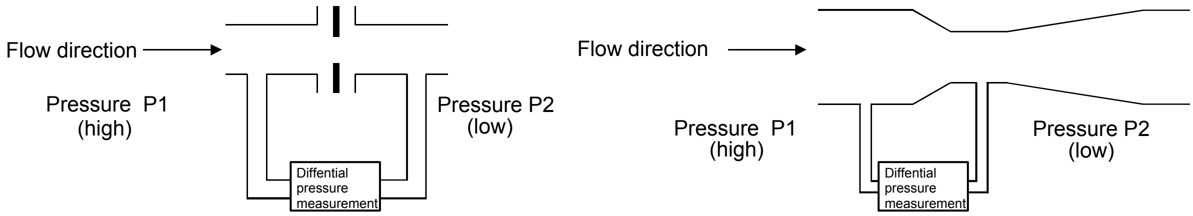


$$\text{Correction flow quantity} = 10 \times \sqrt{50\left(\frac{80+273.15}{77+273.15}\right)\left(\frac{51000+101300}{49000+101300}\right)} = 71.48\ (\%)$$

- Differential pressure △P is the proportional assuming the measuring range of differential pressure transmitter is100%.
- The corrected flow quantity is the flow quantity value corresponding to the measurement differential pressure range of differential pressure transmitter.

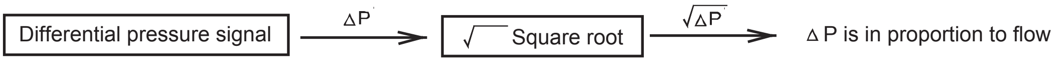
Square Root Extraction

When measuring flow quantity through differential pressure of orifice or venturi tube, the signal which is obtained from sensor has square characteristics. This control linearize the signals.



Flow quantity $Q = k \cdot \sqrt{\frac{\Delta P}{3}}$
Differential pressure ΔP: (P_1-P_2), K: Proportional constant, Y: Density

When measure flow quantity by using differential pressure, the proportional characteristics will be obtained through square root of differential pressure data.
Temperature/pressure correction (P_TPC) is used according to the needs.



Sample PI (SPI) Control

Sample PI (SPI) control executes PI control for the execution cycle and then holds the output in every control cycle.
For details, refer to Appendix 3.5.

Time Proportioning Control

Time proportioning control changes the ON/OFF ratio proportionally with the PID operation result.
For details, refer to Appendix 3.2.

High/low Limiter

It is the function that limits the output MV by PID operation in Auto Mode within the high/low limit.
High/low limiter processing function is only applicable in Auto Mode. (It is not processed in Manual Mode.)

Selection Control

This is the control method that selects the necessary signals (high selector, low selector, intermediate value selector, etc.) among multiple sensor signals or operation signals to control the system.
For details, refer to Appendix 3.6

PV-proportional and -derivative Type (I-PD Control)

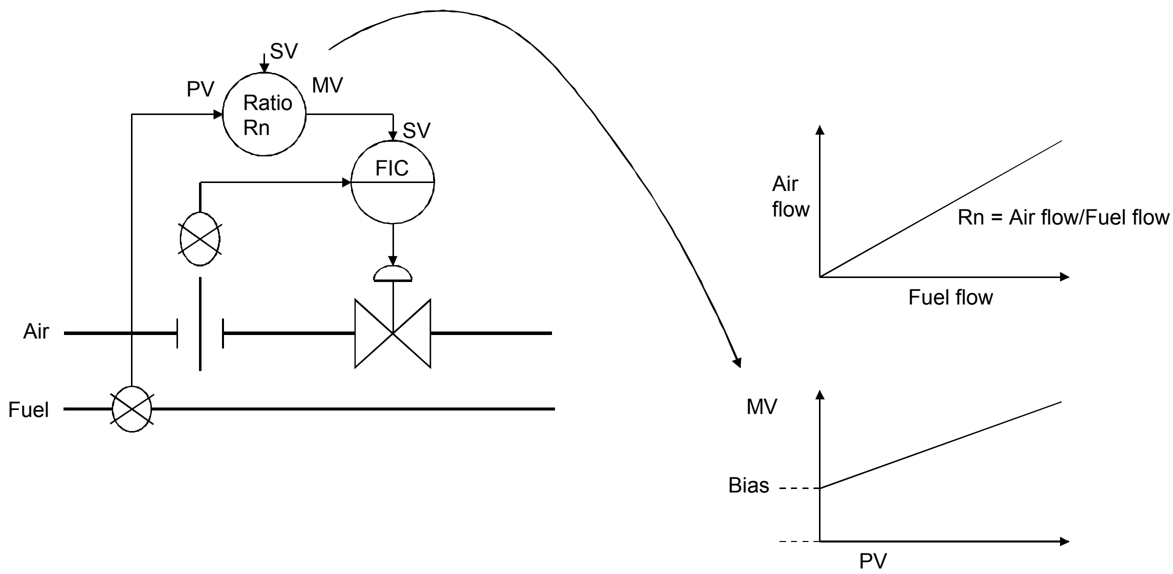
I-PD type control is the control that applies PV value on not only derivative term but also proportional term compared to PV-derivative type.
For details, refer to Appendix 3.5.

Velocity type PID Control

(1) Velocity type control
Velocity type PID control is the operation method for calculating differential manipulated variable △MV between the current value and the previous value.
For details, refer to Appendix 3.9.
(2) PID control
This is the control method that outputs the manipulated variable by combining P, I, D operations, so as to make the manipulated variable reach the same value as the setting value rapidly and correctly.
For details, refer to Appendix 3.7.

Ratio Control

This control holds the proportional relation between more than 2 variables. For example, SV changes in a constant ratio to other variables.
(Example) Control the proportional of air flow to fuel flow.

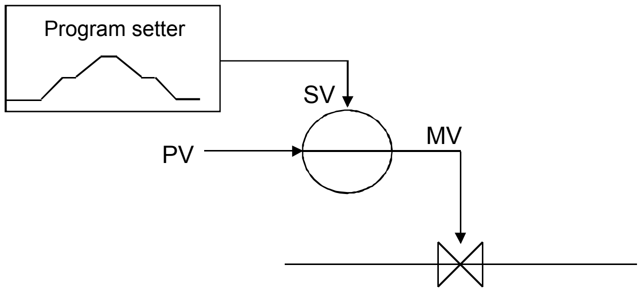


Blend PI Control

This process control method is applicable for the system in which it is good if the control volume is held in a longer period despite short-period vibration.

Program Control

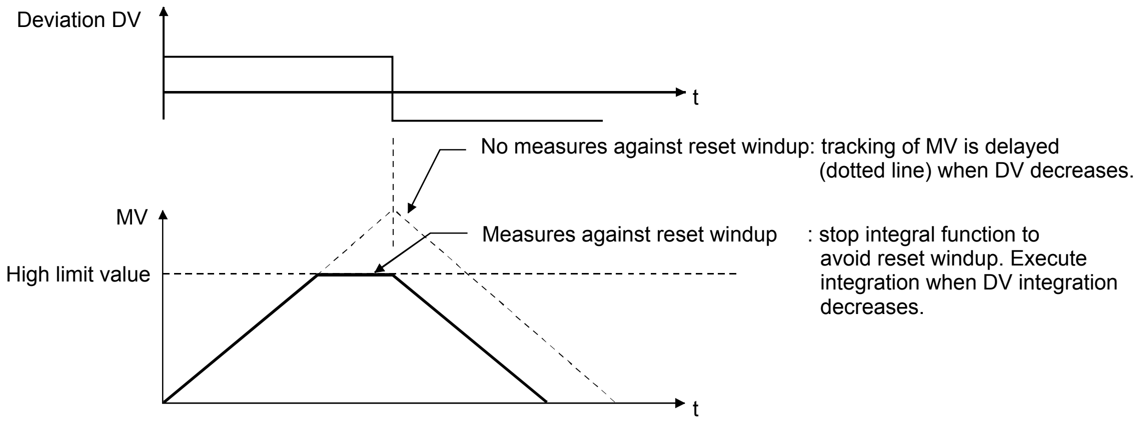
It is the control method to change the setting value by the pre-set program.
It is used for temperature control, etc.
It needs to combine the program setter and PID control for using.



Reset Windup

Reset windup is the problem that deviation is accumulated continuously when integral element exceed saturation limit in the case of excessive deviation.
When reset wind-up occurs, following measures should be taken to enable prompt response when the deviation is inverted.
Measures against reset windup have been applied on CPU module.

1) When integral element of MV exceeds high/low limit, it shall return to the high/low limit value.
2) When integral element of MV exceeds high/low limit, integration operation to the exceeding direction shall be stopped.



First Order Lag Filter

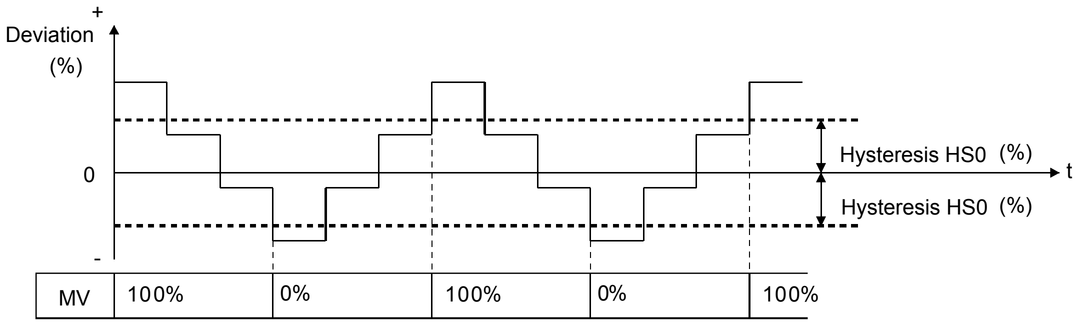
This is used as filter for eliminating noise of measured value PV.
Execute the first order lag operation by the following expression.

$$PVf = \frac{T1 \times PVf_{n-1}}{T1 + \triangle T} + \frac{\triangle T \times PV}{T1 + \triangle T}$$

T1: Time constant(s), $\triangle$T: Execution cycle, PV: Present input value, PVf_{n-1}: Previous filter value

2 position ON/OFF Control

This is the method that outputs 2 steps of MV signals for deviation to control the system.
It is used when it is good if the process variable is within a certain range.
The output is BOOL type.



Condition	Deviation (DV)
Direct action	DV (%)=PV (%)-SV (%)
Reverse action	DV (%)=SV (%)-PV (%)

DV: Deviation (%), HS0: Hysteresis (%), MV: MV output

$$SV (\%) = \frac{\text{SV - low limit of engineering value}}{\text{high limit of engineering value - low limit of engineering value}} \times 100$$

$$PV (\%) = \frac{\text{PV - low limit of engineering value}}{\text{high limit of engineering value - low limit of engineering value}} \times 100$$

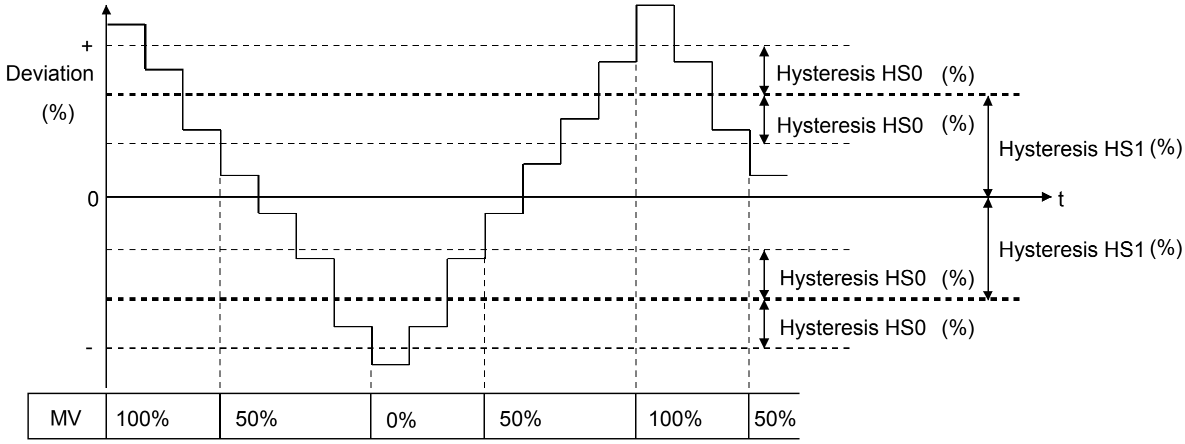
Hysteresis (%) is the percentage to (High limit of engineering value- Low limit of engineering value).

2-degree-of-freedom PID Control

In former PID control, the optimum PID constants for SV value changing and disturbance response are not identical. Whichever optimum value is applied, it may not be the optimum value for the other side.
2-degree-of-freedom PID control is a method for optimizing both disturbance response and target tracking.
For details, refer to Appendix 3.5.

3 position ON/OFF Control

This is the control method that outputs 3 steps of MV signals for deviation to control the system.
It can be applied when it is good if the process variable is within a certain range.
The output is BOOL type.



Condition	Deviation (DV)
Direct action	DV (%)=PV (%)-SV (%)
Reverse action	DV (%)=SV (%)-PV (%)

DV: Deviation (%), HS0: Hysteresis (%), HS1: Hysteresis (%), MV: MV output

$$SV(\%) = \frac{\text{SV - low limit of engineering value}}{\text{high limit of engineering value - low limit of engineering value}} \times 100$$

$$PV(\%) = \frac{\text{PV - low limit of engineering value}}{\text{high limit of engineering value - low limit of engineering value}} \times 100$$

Hysteresis (%) is expressed by percentage corresponding to (High limit value of engineering value - Low limit value of engineering value).

Appendix 6 Instructions Added to and Changed from Old Version

The following instructions are newly included and changed with the upgrade.

Compatible version*1	Added/Changed Instruction	Description	Reference
Version 1.04E	SR	Public variable (variable name: IR) included	Section 5.1.1
	RS		Section 5.1.2
	AIN_4CH_G	Public variable (variable name: CHCNVENB) included	Section 10.1.3
	AIN_2CH_DG		Section 10.1.5
	TC_4CH		Section 10.2.1
	TCV_4CH_G		Section 10.2.3
	RTD_4CH	● Public variable (variable name: CHCNVENB) included ● Q64RD-G included as compatible modules	Section 10.2.4
	DIN_64PT	QX82, QX82-S1 included as compatible modules	Section 10.4.4
Version 1.06G	SEND	Public variable (variable name: CHGSYS) included	Section 5.5.1
	RECV		Section 5.5.2
Version 1.08J	IS_CONNECTED(_E)_	New addition	Section 4.9.5
Version 1.10L	P_PIDP_EX_T_	New addition	Section 8.2.11
	P_PIDP_EX_		Section 8.2.12
	M_PIDP_EX_T_		Section 9.1.13
	M_PIDP_EX_		Section 9.1.14
	P_OUT3_	New addition ● Tag type 2PIDH is added. ● CASCADE DIRECT (CASDR) is added to control modes. ● Tag Stop (TSTP) is added to I/O modes.	Section 8.1.4
	P_2PIDH_T_		Section 8.2.7
	P_2PIDH_		Section 8.2.8
	M_2PIDH_T_		Section 9.1.9
	M_2PIDH_		Section 9.1.10
Version 1.14Q	P_PGS2_	New addition ● Tag type PGS2 is added.	Section 8.2.25
	M_PGS2_		Section 9.1.37
	P_MCHG	PGS2 is added to the compatible tag type.	Section 8.3.1
	P_OUT3_	The Limit Cycle method of auto tuning is added.	Section 8.1.4
	P_2PIDH_T_		Section 8.2.7
	P_2PIDH		Section 8.2.8
	P_PGS2_	Processing when the Number of steps setting is 0 is added to the PV start function.	Section 8.2.25
	AIN_8CH_G	New addition	Section 10.1.4
	AIN_6CH_DG		Section 10.1.6
	AOUT_6CH_G		Section 10.1.11
	AOUT_2CH	Q62DAN is included as compatible module.	Section 10.1.7
	AOUT_4CH	Q64DAN is included as compatible module.	Section 10.1.8
	AOUT_8CH	Q68DAVN and Q68DAIN are included as compatible module.	Section 10.1.9
	DIN_32PT	QX41-S1 is included as compatible module.	Section 10.4.3
	DIN_64PT	QX42-S1 is included as compatible module.	Section 10.4.4
	SEND	Operational restrictions for an Ethernet module mounted on the redundant type extension base unit of Redundant CPU are added.	Section 5.5.1
	RECV		Section 5.5.2

*1 The compatible version can be confirmed in Product Information. For details, refer to PX Developer Version 1 Operating Manual (Programming Tool) or (Monitor Tool).

Compatible version*1	Added/Changed Instruction	Description	Reference
Version 1.18U	TC_8CH_G	New addition	Section 10.2.2
	P_SUM2_		Section 7.1.8
	P_IN	Enabling/Disabling the input limiter processing is added.	Section 8.1.1
	RTD_8CH_G	New addition	Section 10.2.5
	Loop tag FB	Usage of output open alarm is added.	Appendix 3.11
Version 1.20W	P_IN	Initial values of the public variable are changed.	Section 8.1.1
	P_MSET_	New addition	Section 8.1.9
	M_SWM_		Section 9.1.38
	TC_8CH_G	Q68TD-G-H02 is included as compatible module.	Section 10.2.2
	Analog module FB	Explanation on the percentage conversion of digital value is added.	Appendix 3.12
Version 1.23Z	P_RANGE_	New addition	Section 7.1.9
	P_PFC_SF_		Section 8.2.30
	P_PFC_SS_		Section 8.2.31
	P_PFC_INT_		Section 8.2.32
	M_PFC_SF_		Section 9.1.39
	M_PFC_SS_		Section 9.1.40
	M_PFC_INT_		Section 9.1.41
	M_PB_		Section 9.2.9
	P_OUT3_	The tight shut/full open function is added.	Section 8.1.4
	M_2PIDH_T_	Public variable (variable name: OUT3_FOTS_EN, OUT3_MVFO, OUT3_MVTS) included	Section 9.1.9
	M_2PIDH_		Section 9.1.10
	AIN_4CH_AOUT_2CH	New addition	Section 10.1.12
Version 1.28E	Inline ST	New addition	Section 2.9
	NEG(_E)_		Section 4.2.7
	POW(_E)_		Section 4.3.6
	P_SEL_T3_		Section 8.2.29
	M_SEL_T3_		Section 9.1.34
	AOUT_2CH_G	Public variable (variable name: BNAUTSET1 to BNAUTSET2, BNAUTSET1ENB to BNAUTSET2ENB) included	Section 10.1.10
Version 1.31H	P_DUTY_8PT_	New addition	Section 7.4.11
	M_PVAL_T_		Section 9.1.42
	M_HTCL_T_		Section 9.1.43
Version 1.42U	MOVE_E_	New addition	Section 4.3.7
	P_PFC_SF_	Public variable (variable name: MODEL_INIT) included	Section 8.2.30
	P_PFC_SS_		Section 8.2.31
	P_PFC_INT_		Section 8.2.32
	M_PFC_SF_	Public variable (variable name: PFC_SF_MODEL_INIT) included	Section 9.1.39
	M_PFC_SS_	Public variable (variable name: PFC_SS_MODEL_INIT) included	Section 9.1.40
	M_PFC_INT_	Public variable (variable name: PFC_INT_MODEL_INIT) included	Section 9.1.41
	M_ALARM_64PT_	New addition	Section 9.3.2
	M_MESSAGE_64PT_		Section 9.4.2
	CT_8CH		Section 10.1.13

*1 The compatible version can be confirmed in Product Information. For details, refer to PX Developer Version 1 Operating Manual (Programming Tool) or (Monitor Tool).

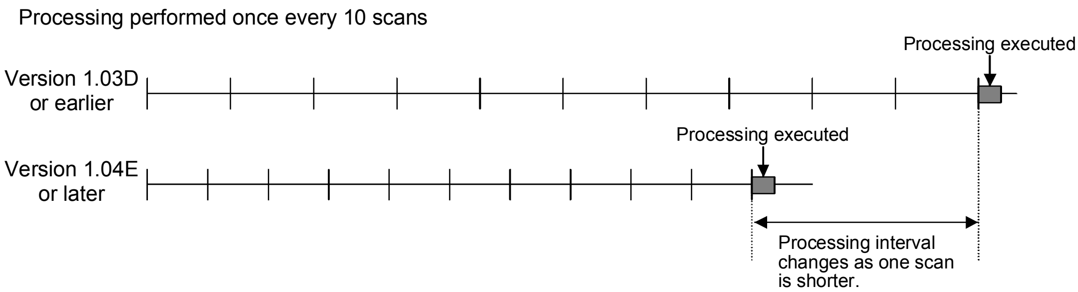
Appendix 6.1 Precautions an the compile function improvement

The programming tool includes the improved compile function from the new version (Version 1.04E or later).
Therefore, the FBD programs compiled by the new version (Version 1.04E or later) outperform those compiled by the old version (Version 1.03D or earlier) as follows;

- Reduce the number of ladder program steps generated by compile.
- Reduce the scan time of the FBD program.
- Program execution timing when the CPU module is reset, switched from STOP to RUN or power ON.

Note the following when utilizing the program created by the Version 1.03D or earlier

(1) Precautions on reduced scan time of FBD program
As the FBD programs compiled by the new version (Version 1.04E or later) are executed faster, they require less scan time as compared with those compiled by the old version (Version 1.03D or earlier).
Therefore, if the scan time-dependent processing is executed for scan execution FBD programs or the user-created ladder programs, the processing interval differs between the old version (Version 1.03D or earlier) and new version (Version 1.04E or later).



POINT

The scan time can be confirmed on GX application.
For the confirmation method, refer to the following manuals:

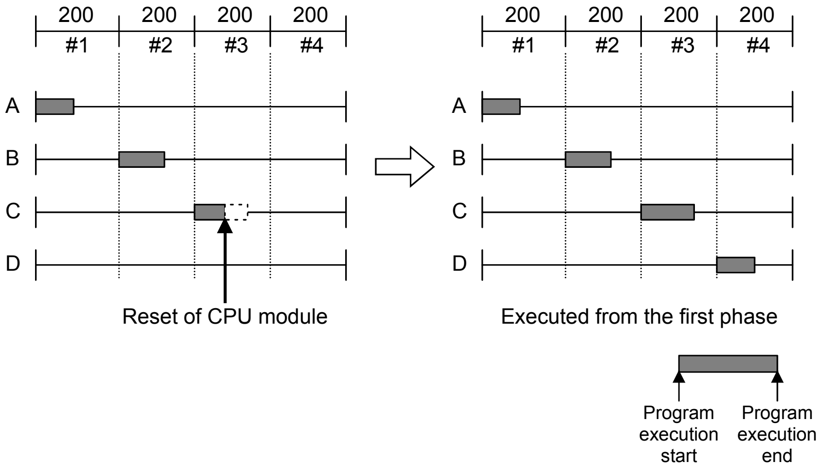
- GX Works2 Version 1 Operating Manual (Common)
- GX Developer Version 8 Operating Manual".

(2) Precautions on program execution timing at CPU module reset, STOP → RUN or power OFF → ON

The FBD programs compiled by the new version (Version 1.04E or later) differ from those compiled by the old version (Version 1.03D or earlier) in the program execution timing when the CPU module is reset, switched from STOP to RUN or power OFF to ON.

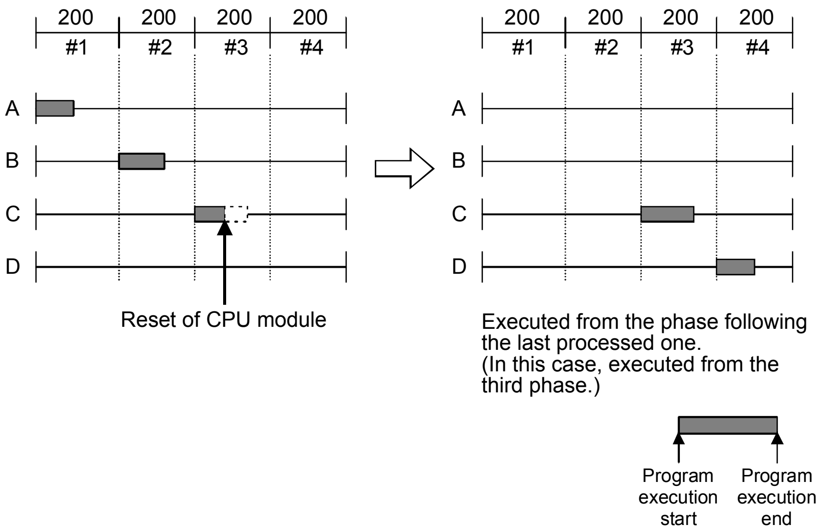
[Version 1.04E or later]

The program is executed from the first phase when the CPU module is reset, switched from STOP to RUN or power OFF to ON.



[Version 1.02C or earlier]

The program is executed from the phase following the last processed one when the CPU module is reset, switched from STOP to RUN or power OFF to ON.



MEMO

INDEX

C-1

C-2

C-3

C-4

C-5

C-7

C-10

WARRANTY

Please confirm the following product warranty details before using this product.

1. Gratis Warranty Term and Gratis Warranty Range

If any faults or defects (hereinafter "Failure") found to be the responsibility of Mitsubishi occurs during use of the product within the gratis warranty term, the product shall be repaired at no cost via the sales representative or Mitsubishi Service Company.
However, if repairs are required onsite at domestic or overseas location, expenses to send an engineer will be solely at the customer's discretion. Mitsubishi shall not be held responsible for any re-commissioning, maintenance, or testing on-site that involves replacement of the failed module.
[Gratis Warranty Term]
The gratis warranty term of the product shall be for one year after the date of purchase or delivery to a designated place. Note that after manufacture and shipment from Mitsubishi, the maximum distribution period shall be six (6) months, and the longest gratis warranty term after manufacturing shall be eighteen (18) months. The gratis warranty term of repair parts shall not exceed the gratis warranty term before repairs.
[Gratis Warranty Range]

(1) The range shall be limited to normal use within the usage state, usage methods and usage environment, etc., which follow the conditions and precautions, etc., given in the instruction manual, user's manual and caution labels on the product.

(2) Even within the gratis warranty term, repairs shall be charged for in the following cases.

1. Failure occurring from inappropriate storage or handling, carelessness or negligence by the user. Failure caused by the user's hardware or software design.
2. Failure caused by unapproved modifications, etc., to the product by the user.
3. When the Mitsubishi product is assembled into a user's device, Failure that could have been avoided if functions or structures, judged as necessary in the legal safety measures the user's device is subject to or as necessary by industry standards, had been provided.
4. Failure that could have been avoided if consumable parts (battery, backlight, fuse, etc.) designated in the instruction manual had been correctly serviced or replaced.
5. Failure caused by external irresistible forces such as fires or abnormal voltages, and Failure caused by force majeure such as earthquakes, lightning, wind and water damage.
6. Failure caused by reasons unpredictable by scientific technology standards at time of shipment from Mitsubishi.
7. Any other failure found not to be the responsibility of Mitsubishi or that admitted not to be so by the user.

2. Onerous repair term after discontinuation of production

(1) Mitsubishi shall accept onerous product repairs for seven (7) years after production of the product is discontinued. Discontinuation of production shall be notified with Mitsubishi Technical Bulletins, etc.

(2) Product supply (including repair parts) is not available after production is discontinued.

3. Overseas service

Overseas, repairs shall be accepted by Mitsubishi's local overseas FA Center. Note that the repair conditions at each FA Center may differ.

4. Exclusion of loss in opportunity and secondary loss from warranty liability

Regardless of the gratis warranty term, Mitsubishi shall not be liable for compensation to:

(1) Damages caused by any cause found not to be the responsibility of Mitsubishi.

(2) Loss in opportunity, lost profits incurred to the user by Failures of Mitsubishi products.

(3) Special damages and secondary damages whether foreseeable or not, compensation for accidents, and compensation for damages to products other than Mitsubishi products.

(4) Replacement by the user, maintenance of on-site equipment, start-up test run and other tasks.

5. Changes in product specifications

The specifications given in the catalogs, manuals or technical documents are subject to change without prior notice.

SH(NA)-080371E(2/2)-S(2110)KWIX
MODEL:SW1D5C-FBDQ-P-E
MODEL CODE: 13JW00

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : TOKYO BUILDING, 2-7-3 MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS : 1-14 , YADA-MINAMI 5-CHOME , HIGASHI-KU, NAGOYA , JAPAN