μGPCsH series

Instruction Words

TOYO DENKI

Introduction

Thank you for purchasing the Toyo Denki FA *μ*GPCsH digital controller.

This instruction word programming manual explains the theory of programming, the relays and registers, and each of the instruction words. In order to use the *μ*GPCsH correctly, please read this manual carefully.

Please also read the related manuals below.

Name	Manual number	Content
*μ*GPCsH Series TDFlowEditor Manual: Operation	QG18721	This manual explains the interface of TDFlowEditor and how to use the program.
*μ*GPCsH Series User's Manual (Hardware)	QG18720	*μ*GPCsH Series system configuration, hardware specifications of each module, etc.

Caution

(1) No part of this manual may be reproduced or duplicated without permission.

(2) The content of this manual is subject to change without prior notice.

(3) We have endeavored to make this manual as complete and accurate as possible. However, if you notice any errors or ambiguities, please report them to the sales office shown on the back of this manual, stating the manual number indicated on the front cover.

Safety Notice

Read this “Safety Notice” carefully before using the product and use the product accordingly. In this manual, safety-related items are divided into “Danger” and “Caution” as follows.

⚠Danger: Mishandling may cause death or serious injury.

⚠Caution: Mishandling may cause moderate bodily injury, minor injury or damage to property.

Note that items marked ⚠ Caution may also result in other serious consequences depending on the circumstances.

All safety notices contain important information which should be strictly observed. Matters requiring special attention are shown below, which are also indicated with the marks shown above.

⚠Danger
• Emergency stop circuits and interlock circuits should be implemented outside the PC. Malfunction of the PC may result in damage or accidents involving the machinery.

⚠Caution
• Only perform operations such as changing programs, forced output, start, stop, etc., after ensuring safety. Incorrect operation may cause the machine to function, resulting in accidents or damage to the machinery.

Revision History

* The manual number is shown at the bottom right of the cover sheet.

Date printed	* Manual number	Details of revision
2010.10	QG18719	First edition issued

Contents

Chapter 1 Outline

In the *μ*GPCsH series, we have developed a new language for the *μ*-GPC as a control language for application programs, without using computer languages (assembly language, C-language, etc.).

The *μ*-GPC language employs the ladder network that has been conventionally used in sequencers, etc., for logic operations, and the DFS (data flow symbol) that has been used in analog computers, etc., for numerical operations. This is a new programming technique that enables visual programming using programming tools that run on personal computers.

The *μ*-GPC language has the following features.

(1) It has an optimized language system that has revolutionized the concept of computer languages. Rather than defining the processing procedures of the microprocessor, it defines the processing procedures for the data.

(2) It is a graphic display language which makes a program very easy to understand, thus enabling programming with minimum errors.
It is possible to program both logic operations and data processing on the same screen.

(3) Since it automatically converts the types of data handled (integers, BCD types, real numbers, etc.), there is no need to use type conversion instructions in a program. If data is divided, conversion instructions can be used.

(4) Since you can use a wide range of time series functions for control such as S-letter operations, a function created using a number of ladder symbols can be defined with a single symbol, making it easy for anyone to create programs.
Because it automatically measures and adjusts the execution time of a program, you do not need to pay attention to the time at all.

(5) With three index registers (X, Y and Z), you can make index decorations and you can also create flexible programs typical of computers. Program loops using jump instructions also help to reduce the number of steps.

(6) You can easily prepare structured programs using subprograms.
This is ideal for the reuse and standardization of application programs.

(7) You can create four multi-task programs to achieve efficient systems. Since you can set execution cycle times independently, the execution cycle can be divided into four.

(8) Since all the program data is stored in the CPU, even if the computer used for development fails, you can perform maintenance using another computer. The comments in programs can also be recovered, so programs, comments and execution data can be maintained as a set.

(9) Using the many convenient functions of the TDFlowEditor programming tool, you can perform the work required to changeover the system accurately, in a very short time, with few errors. For the details of functions such as loader, monitor, debugger, trend, and trace back that can be used when the equipment is running, refer to the *μGPCsH Series TDFlowEditor Manual: Operation*.

Chapter 2 Programming Method Using the μ-GPC Language

With the μGPCsH, the programs loaded on a single CPU are constructed using the concept of a project. A project is given a name that can be changed freely. Project names should be set appropriately. A single project can be divided into six parts: IO allocation, Task 1, Task 2, Task 3, Task 4 and subroutine.

(1) IO Allocation

This defines the hardware related conditions of the CPU; therefore it defines the system configuration.

(2) Task 1, Task 2, Task 3, and Task 4

The task with the highest priority is made Task 1, which consists of scan time and a number of subprograms. Each subprogram is given a program name which can be changed to indicate the process that it handles within a program. If you do not specify a name, it is called NoName by default.

A subprogram is written on a programming sheet comprising 12 horizontal columns and 19 vertical lines. A single programming sheet is one page, and more pages can be added successively.

Local symbols can be used within a subprogram, but handover between subprograms can only be effected by the global memory.

(3) Subroutine

This is a subroutine that is used in common, similarly to the subprograms in Task 1, Task 2, Task 3, and Task 4. Subroutines are given names using six-digit English alphanumeric codes.

(4) Programming sheets

Each of the twelve horizontal columns has a part for inserting a symbol and a part for a crosspoint. A program is made by placing symbols in these parts and entering label names for them.

No END instructions or compilation operations are required and the program is compiled automatically when the editor is closed.

Contacts using ladder symbols and data flow symbols can be placed in columns 1 to 11.

Only coils using ladder symbols can be placed in Column 12.

There is no crosspoint in column 11, and therefore no intersection of addition instructions or ladder symbols can be inserted.

Usually, binary operators (addition, subtraction, multiplication, etc.) are placed at a cross point, but C-contacts are placed as symbols since they alone can be given a contact name.

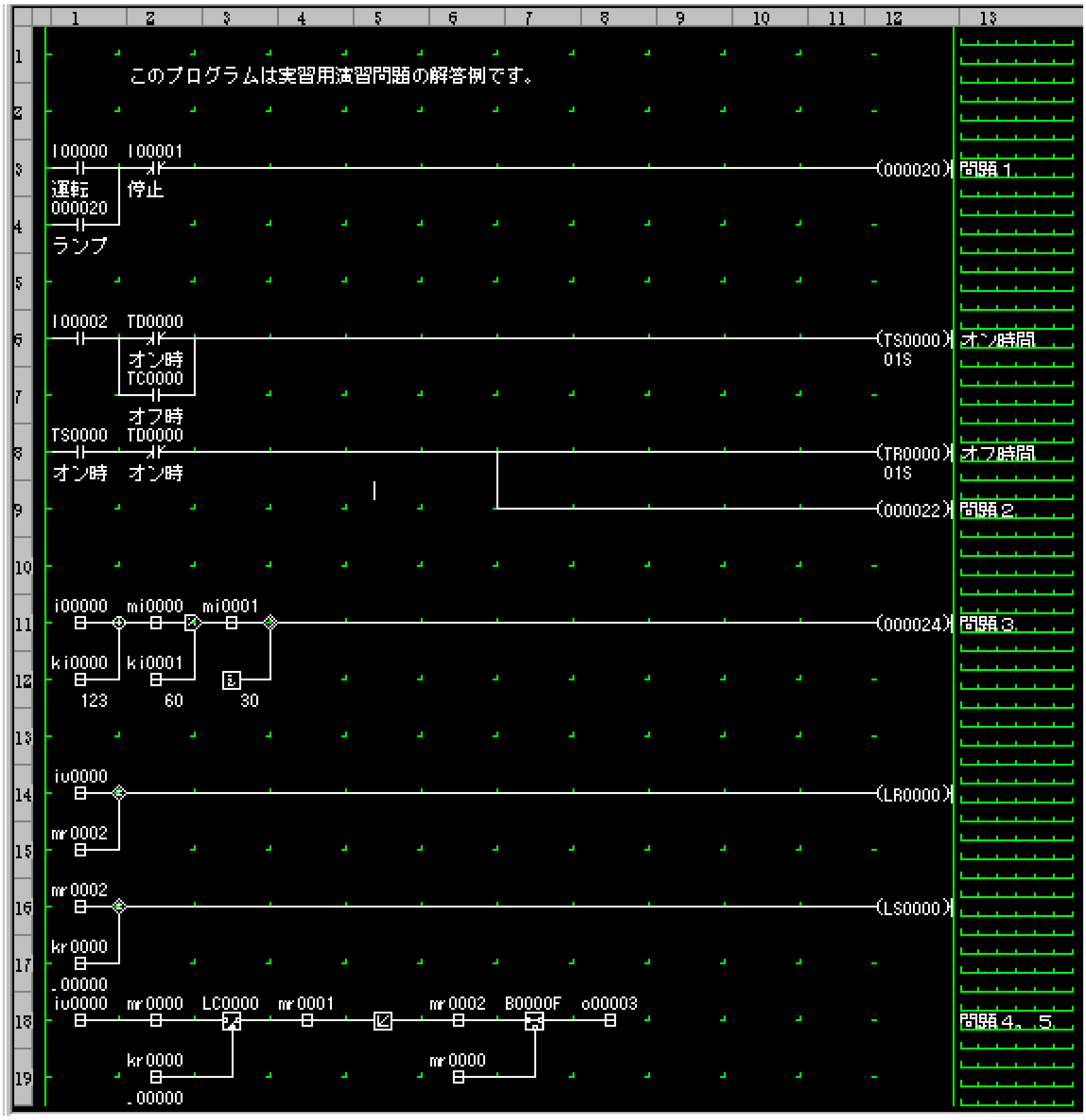
Each of the nineteen vertical lines is divided into parts for inserting a label name, symbol and data comment.

Programs that use crosspoints can extend over multiple rows, but programs exceeding nineteen lines are divided across multiple pages using a temporary label.

(5) Program comment

In the programming sheet, column 13 can be used for comments as shown in the programming example in the figure on the next page. If a coil is placed with a ladder symbol, it is applied to the position of comment at the relevant contact point. The comment is displayed automatically, unless it is input at the contact side.

However, you can only enter a maximum of three 2-byte characters (six 1-byte characters) so the characters should be chosen carefully with easy identification in mind. Also, as in the first line, positions for comments with no symbols entered can be used in their entirety for comments.



(6) Explanation of the sample program

The sample problem shown above is explained for reference.

- The first line is a comment line. As shown in this example, the content of the program is written beforehand.
- The second line is a blank line. Blank lines are inserted where necessary to make the program list easier to read.
- The third and fourth lines are ladder symbols for a HOLD circuit that uses a typical 2-operation switch.
- By turning the input switch I00000 ON, the lamp circuit O00020 is set to light up, and the status is set to HOLD.
- I00001 is a B-contact input switch that releases the HOLD status. If it is ON, the lamp is turned off.
- The fifth line is a blank line.
- The sixth to ninth lines are the flash circuit of a lamp combining an on-delay timer and an off-delay timer. The on-time and off-time can be changed independently.
- The setting time of each timer is specified at the bottom of the coil in column 12. In the example above it is set at 1.0 S (second), but it can be set up to 2 hours, with H representing the hour, M the minutes, and S the seconds. The minimum unit is 10 mS, which is written as 0.01 S.
- The tenth line is a blank line.
- The eleventh to twelfth lines are a circuit that reads numerical data from the 16-bit input module, adds the constant 123 to it, divides the resulting value by 60 to obtain the remainder, and turns the lamp on if the remainder exceeds 30.
- Since the results of operations in the process are stored in registers, you can check them while monitoring the results during debugging. The logic operation symbol is placed to the right of the comparison instruction symbol.
- The thirteenth line is a blank line.
- The fourteenth to nineteenth lines show an example of a pattern generation circuit that uses a latch relay and a change ratio limitation function (we call it ARC). It continuously generates triangular waves. The wave height value can be set from the input module using BCD numerical values. The cycle can be changed indirectly by changing the alteration ratio parameters of the ARC function. Real number operations, integer operations and BCD operations are all present in the eighteenth and nineteenth lines and the patterns are continuously generated by switching the input value of ARC using the C-contact.
- The C-contact at B0000F is for test use, and it directly outputs the input value by turning it on using debugger.

Chapter 3 Data Types and Range That Can Be Handled

The data handled in the μGPCsH is represented by a label name consisting of a 2-digit type plus 4-digit hexadecimal number. Also, the first digit of the hexadecimal number can be replaced by the index label X, Y, or Z.

Label examples: IOX123, b0y234, mr02AF

3.1 Kinds of Data

The data handled in the μGPCsH can roughly be divided into two kinds, “logic data” and “numerical data.”

3.1.1 Logic Data

- Logic data is a data that represents logic of one bit, namely “1” or “0.”
- Logic data is processed using logic operations, etc.
- Logic data is stored in a “relay,” and it can be accessed in a program by specifying a “relay number.”
- The result of the operation of the comparison operation symbol is logic data.

Points

- In the μGPCsH, “relays” are what store logic data.
- “1” in logic data corresponds to a relay being “on” while “0” corresponds to “off.”

3.1.2 Numerical Data

- Numerical data is data that represents 16 bits (1 word) or 32 bits (2 words) as a single unit.
- Numerical data is stored in a “register,” and it can be accessed in a program by specifying a “register number.”
- Logic data is the input condition of the comparison operation symbol.

Points

In the μGPCsH, “registers” are what store numeric data.

An uppercase character is used as the initial letter of the relay number of logic data.
Example: I00000

A lowercase character is used as the initial letter of the register number of numerical data.
Example: i00000

3.2 Kinds of Data Types

3.2.1 Types of Logic Data

There is no particular distinction between types. The data that can be handled is 1 (on) and 0 (off).

3.2.2 Types of Numerical Data

There are the following types of numerical data, which are explained in 3-3 and thereafter.

(1) 16-bit integer type (i-form)

(2) 16-bit BCD type (u-form)

(3) 32-bit integer type (w-form)

(4) 32-bit BCD type (v-form)

(5) 32-bit real number type (r-form)

3.3 16-Bit Integer Type (i-Form)

Represents 16-bit integer value signed data as a single unit (one word).

The range of data that is handled internally is: -32,768 to 32,767 (8000H to 7FFFH).

This kind of numerical data is called “16-bit integer data.”

3.4 16-Bit BCD Type (u-Form)

Represents 16-bit BCD (binary coded decimal) 4-digit data as a single unit (one word).

The range of data that is handled internally is: 0000 to 9999 (0000H to 270FH)

This kind of numerical data is called “16-bit BCD data.”

Note: The 16-bit BCD data can only be used for data exchanged with an input and output (I/O) unit (I/O data).

3.5 32-Bit Integer Type (w-Form)

Represents 32-bit integer value signed data as a single unit (two words).

The range of data that is handled internally is: -2147483648 to 2147483647 (80000000H to 7FFFFFFFH)

This kind of numerical data is called “32-bit integer data.”

Note: The 32-bit integer data can only be used for data exchanged with an input and output (I/O) unit (I/O data).

3.6 32-Bit BCD Type (v-Form)

Represents 32-bit BCD (binary coded decimal) 8-digit data as a single unit (two words).

The range of data that is handled internally is: 00000000 to 99999999 (00000000H to 05F5EOFFH)

This kind of numerical data is called “32-bit BCD data.”

Note: The 32-bit BCD data can only be used for data exchanged with an input and output (I/O) unit (I/O data).

3.7 32-Bit Real Number Type (r-Form)

Represents 32-bit floating-point format data as a single unit (two words).

The range of data that is handled internally is: $-6.2573187 \times 10^{38}$ to 6.2573187×10^{38}

This kind of numerical data is called “32-bit real number data.”

Reference: The 32-bit real number data is handled internally as follows.
The user can ignore it.

$(-1)^{S} \times 2^{e-127} \times 1.f$

s: Value of the code part

e: Value of the exponent part

f: Value of the mantissa part (normalized to a 23-bit binary number)

31	30 23	22 0
S	Exponent part	Mantissa part

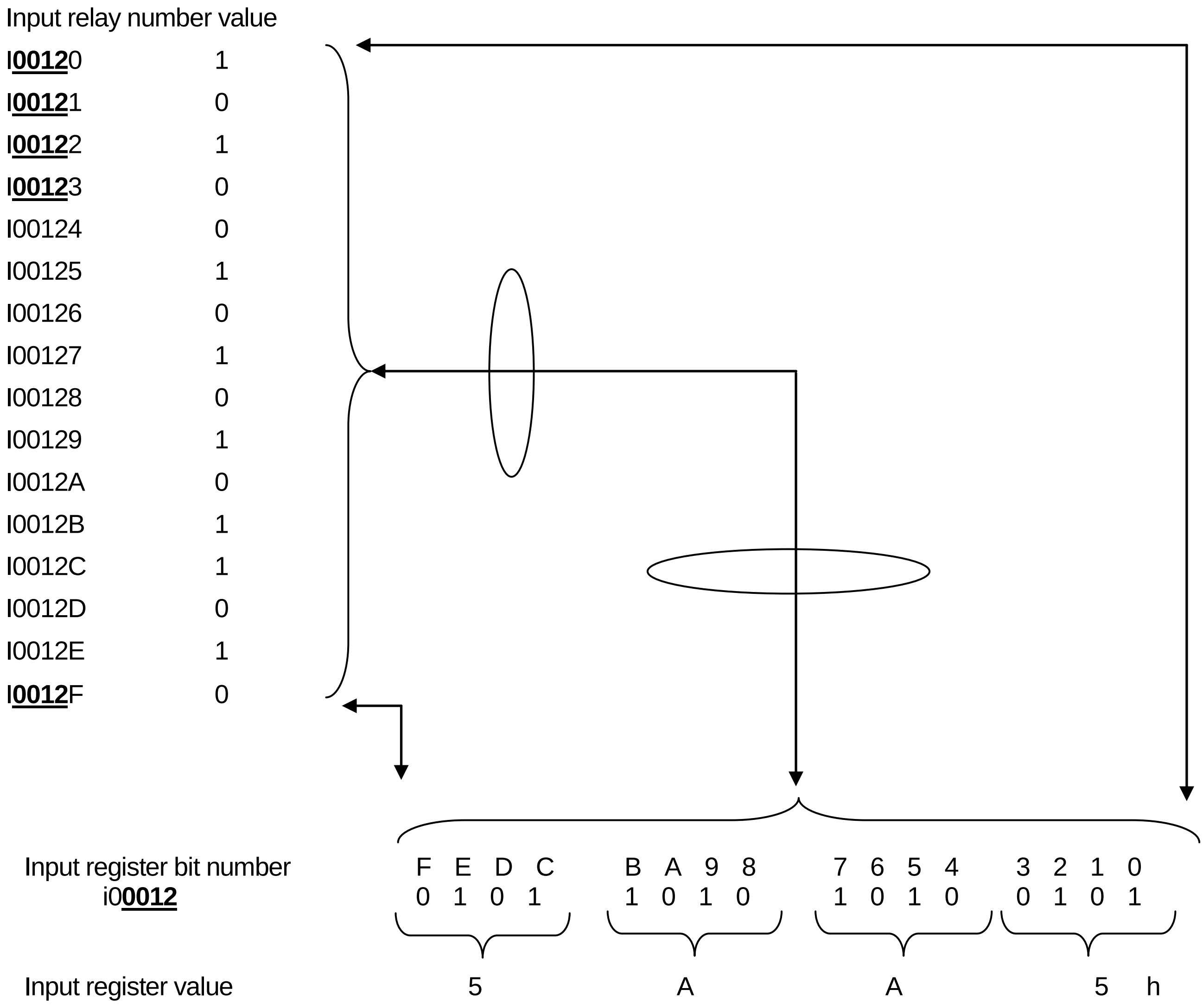
1-bit 8-bit 23-bit

3.8 Relation between the Logic Data and the 16-Bit Integer Data (i-Form)

The “logic data” handled in the μGPCsH can be assembled into a group of 16 bits that is associated with a single unit of “16-bit integer (i-form) data.”

In this case, the logic data and 16-bit integer data, the relay and register that store these data, and the relay number and register number have the following relationship.

Example: Continuous relay numbers (from I00120 to I0012F in the figure below) correspond to the input relays that contain 16 units of logic data. Meanwhile, register number i00012 corresponds to the input register that contains 1 unit of 16-bit integer data. The relation between both can be illustrated as in the figure below. This figure represents how the content of input register i00012: 5AA5 (hexadecimal) is expanded in input registers from I00120 to I0012F.



Likewise, the correspondence between the input relays that are put into groups of 16 bits and the input register is as follows.

Input relay number			Input register number
I00000, I00001,	to	I0000F	i00000
I00010, I00011,	to	I0001F	i00001
I00020, I00021,	to	I0002F	i00002

Aside from these, each kind of relay such as output relays, link relays, auxiliary relays, etc., can likewise be associated with the output register, link register, auxiliary register, etc.

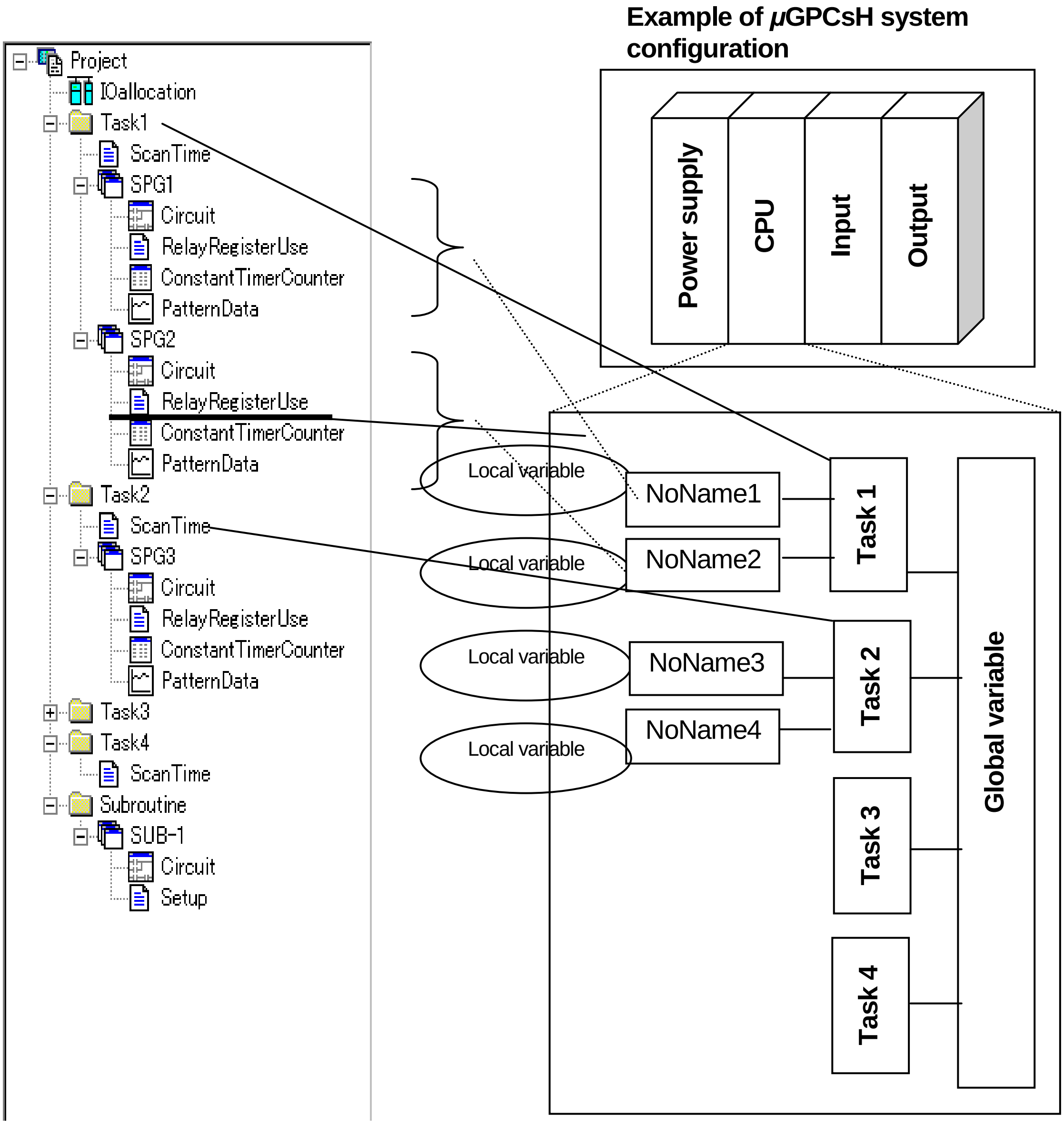
Points Correspondence between the relay number and the register number

Example: Relay number I00123 represents bit number 3 of register number i00012.

Note: The range of relay numbers and register numbers depends on the kinds of relays and registers. Some registers are meaningless when expanded in relays, and hence they cannot be expanded (kr, mr, mi, etc.)

Chapter 4 Kinds of Relays and Registers

4.1 Relation between the Local Variable, Global Variable and Subprogram



- Local variable: A variable that can be accessed within a single subprogram only (it cannot be accessed from other subprograms). The number used should be set with "**RelayRegisterUse**" in each subprogram. It is prepared by dividing it depending on the processing function.
  Example: mi, B0, etc.
- Global variable: A variable that can be accessed from any subprogram within a single project.
  Example: G0, fi, RI, etc.

4.2 Number of Relays and Registers That Can Be Used

(1) Global variable

The maximum number of variables that can be used in any subprogram within a project is shown in the table below.

Name	Number used (Maximum)	Kind	Data number	Data direction	Remarks
Input relay	8,192	Contact	I00000 to I01FFF	Load	*1
Input register	512	Input data	ix0000 to ix01FF		*3
Output relay	8,192	Coil, contact	O00000 to O01FFF	Store	*1
Output register	(512)	Output data	ox0000 to ox01FF		*3
Announce relay	32,768	System information	Z00000 to Z07FFF	Load	
Announce register	2,048		z00000 to z007FF		
Global relay Global register	131,072	Coil, contact	G00000 to G1FFFF	Load Store	
	1,048,576	Global data	g00000 to gFFFFF		
	32,768		gr0000 to grFFFE		*2
Retain relay Retain register	65,536	Coil, contact	RI0000 to RIFFFF	Load Store	
	65,536	Retain data	ri0000 to riFFFF		
	32,768		rr0000 to rrFFFE		*2
Network relay Network register	65,536	Coil, contact	FI0000 to FIFFFF	Load Store	
	4,096	Network data	fi0000 to fi0FFF		
	2,048		fr0000 to fr0FFE		*2
	4,096	Network data	ei0000 to ei0FFF	Load Store	
	2,048		er0000 to er0FFE		*2

*1: The total number of inputs and outputs.

*2: Odd numbers cannot be used.

*3: X is replaced with u (BCD 4-digit), v (BCD 8-digit) or w (32-bit integer) representing the type of I/O register.

(2) Local variable

The maximum number that can be used in each subprogram is shown in the table below.

Name	Number used (Maximum)	Kind	Data number	Data direction	Remarks
Auxiliary relay	6144	Coil, contact	B00000 to B017FF	Load	
Auxiliary register	384	Auxiliary data	b00000 to b0017F	Store	
Latch relay Latch register	512	Set coil	LS0000 to LS01FF	Load	
			ls0000 to ls001F	Store	
		Reset coil	LR0000 to LR01FF	Load	
			lr0000 to lr001F	Store	
	32	Latch contact	LC0000 to LC01FF	Load	
			lc0000 to lc001F		
On differential relay On differential register	512	Coil	US0000 to US01FF	Load	
			us0000 to us001F	Store	
	32	Differential contact	UC0000 to UC01FF	Load	
			uc0000 to uc001F		
Off differential relay Off differential register	512	Coil	DS0000 to DS01FF	Load	
			ds0000 to ds001F	Store	
	32	Differential contact	DC0000 to DC01FF	Load	
			dc0000 to dc001F		
On timer On timer register	512	Coil, instantaneous contact	TS0000 to TS01FF	Load	
			ts0000 to ts001F	Store	
	32	Timing contact	TD0000 to TD01FF	Load	
			td0000 to td001F		
	512	Elapsed time	tn0000 to tn01FF	Load	
Off timer Off timer register	512	Coil, instantaneous contact	TR0000 to TR01FF	Load	
			tr0000 to tr001F	Store	
	32	Timing contact	TC0000 to TC01FF	Load	
			tc0000 to tc001F		
	512	Elapsed time	tf0000 to tf01FF	Load	

Counter	256	Reset Coil	NR0000 to NR00FF	Load	
			nr0000 to nr000F	Store	
		Preset Coil	NP0000 to NP00FF	Load	
			np0000 to np000F	Store	
		Up coil	NU0000 to NU00FF	Load	
			nu0000 to nu000F	Store	
		Down coil	ND0000 to ND00FF	Load	
			nd0000 to nd000F	Store	
Counter register	16	Zero detection contact	NZ0000 to NZ00FF	Load	
			nz0000 to nz000F		
	256	Current count value	N00000 to n000FF	Load	
Operation data	8192	Integer	mi0000 to mi1FFF	Load	
	4096	Real number	mr0000 to mr0FFF	Store	
Constant data	8192	Integer	ki0000 to ki1FFF	Load	
	4096	Real number	kr0000 to kr0FFF		
Pattern data	10	Integer	pi0000 to pi0009	Load	*1
	10	Real number	pr0000 to pr0009		*1
Stack register	4096	Coil, contact	SI0000 to SIFFFF	Load Store	
	256	Integer	si0000 to si00FF		
	128	Real number	sr0000 to sr00FF		*2
Index register	3	Integer	indx_x, indx_y, indx_z	Load Store	

*1: The number of patterns that can be used varies depending on the setting of the number of points of pattern data.

*2: Odd numbers cannot be used.

(3) Shared structure of registers

For ease of handling, the global register and stack register have a shared object relationship. The shared object relationship between the relays, integer registers and real number registers of the global memory is shown in the table below. sr0000 represents live line data, and sr0002 represents the first argument.

Relay name	Integer register	Real number register
G00000	g00000	gr0000
G00001		
G00002 ~ G0000F		
G00010	g00001	
G00011		
G00012 ~ G0001F		
G00020 ~ G0002F	g00002	gr0002
G00030 ~ G0003F	g00003	

Relay name	Integer register	Real number register
SI0000	si0000	sr0000
SI0001		
SI0002 ~ SI000F		
SI0010	si0001	
SI0011		
SI0012 ~ SI001F		
SI0020 ~ SI002F	si0002	sr0002
SI0030 ~ SI003F	si0003	

Note: Since the shared object relationship allows operations from any register, special care should be taken when using it.

(4) CPU announce register

Register	Relay	Name	Content
z00000	Z00000	CPU RUN	A relay that turns on when the CPU is running
	Z00001	Serious failure	A relay that turns on when the CPU is experiencing serious failure
	Z00002	Minor failure	A relay that turns on when the CPU is experiencing minor failure
z00003	—	Scan time 1	Task 1 scan time register (BCD) msec
z00004	—	Scan time 2	Task 3 scan time register (BCD) msec
z00005	—	Clock register (year, month)	Year (H) and month (L) display (BCD)
z00006	—	Clock register (day, time)	Day (H) and time (L) display (BCD)
z00007	—	Clock register (minutes, seconds)	Minutes (H) and seconds (L) display (BCD)
z00008	—	Unused	Normally 0
z00009	—	0.25 ms counter	A counter that increases in 0.25 ms increments
z0000A	—	1 sec counter	A counter that increases in 1 s increments
z0000B	—	System task counter	A counter that increases whenever a system task starts
z0000C	—	Code switch information	FL-net code switch value (00h to FFh “255”)
z0000D	Z000D0	CPU implementation information	CPU slot implementation information (Normally 0)
	Z000D1	IO1 implementation information	IO1 slot implementation information (With implementation: 0, With no implementation: 1)
	Z000D2	IO2 implementation information	IO2 slot implementation information (With implementation: 0, With no implementation: 1)
	Z000D3	IO3 implementation information	IO3 slot implementation information (With implementation: 0, With no implementation: 1)
	Z000D4	IO4 implementation information	IO4 slot implementation information (With implementation: 0, With no implementation: 1)
	Z000D5	IO5 implementation information	IO5 slot implementation information (With implementation: 0, With no implementation: 1)
	Z000D6	IO6 implementation information	IO6 slot implementation information (With implementation: 0, With no implementation: 1)
	Z000D7	IO7 implementation information	IO7 slot implementation information (With implementation: 0, With no implementation: 1)
	Z000D8	IO8 implementation information	IO8 slot implementation information (With implementation: 0, With no implementation: 1)
	Z000D9	IO9 implementation information	IO9 slot implementation information (With implementation: 0, With no implementation: 1)
	Z000DA	Unused	Normally 1
	Z000DB	Unused	Normally 1
	Z000DC	USB connection	Tool I/F USB connection: 0, USB not connected: 1
	Z000DD	CPU implementation	CPU implementation information (Normally 1)
	Z000DE	Cell voltage	Cell voltage normal or no battery: 2, Cell voltage down: 0
	Z000DF	RUN/STOP lever	RUN: 1, STOP: 0
z0000E	Z000E0 to Z000E7	Unused	Normally 0
	Z000E8	Operation switch ENT	ENT button press: 1, ENT button release: 0
	Z000E9	Operation switch D	DU lever D: 1, Neutral or U: 0
	Z000EA	Operation switch U	DU lever U: 1, Neutral or D: 0
	Z000EB	Operation switch L	LR lever L: 1, Neutral or R: 0
	Z000EC	Operation switch R	LR lever R: 1, Neutral or L: 0
z0000E	Z000ED to Z000EF	Unused	Normally 0
z0000F	—	CPU version	CPU version register 1.00 = 100

(4) CPU announce register (Continued 1)

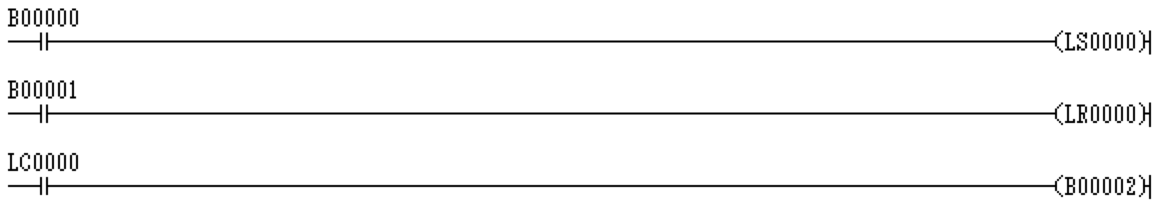
Register	Relay	Name	Content
z00010 to z00017	—	IO initialization error	IO initialization error or blown Tr output module fuse Z001X0 to Z001XF: Slot number Z0010X to Z0017X: Unit number (Basic unit: 0)
z00018 to z0001F	—	IO online check error	IO online check error or loss of external power to Tr output module Z001X0 to Z001XF: Slot number Z0018X to Z0017X: Unit number (Basic unit: 0)
z00020 to z00027	—	IO configuration change	Changed the IO module information Z002X0 to Z002XF: Slot number Z0020X to Z0027X: Unit number (Basic unit: 0)
z00030 to z00037	—	IO setting malfunction	IO allocation and actual configuration differ Z003X0 to Z003XF: Slot number Z0030X to Z0037X: Unit number (Basic unit: 0)
z00038 to z0004F	—	Unused	Unused
z00050 to z000FF	—	Unused	Unused (Can be used as control relay for functions when transplanting past applications)
z00100 to z0012F	—	Unused	Unused
z00130	—	Amount of local memory used	Local memory words used (Variable part: b0, mi, mr etc.) Maximum131,072 words (Displayed only in L)
z00131	—	Amount of local memory used	Local memory words used (Parameter part: ki, kr etc.) Maximum 65,536 words
z00132	—	Number of codes used (L)	Code words used (L) Maximum 327,680 words
z00133	—	Number of codes used (H)	Code words used (H) Maximum 327,680 words
z00134	—	Number of system definitions used	System definition words used Maximum 8,192 words
z00135	—	Unused	Unused
z00136	—	Unused	Unused
z00137	—	Number of general-purpose files used	General-purpose information words used Maximum 131,072 words
z00138	—	IP address	Local module IP address (LL)
z00139	—	IP address	Local module IP address (LH)
z0013A	—	IP address	Local module IP address (HL)
z0013B	—	IP address	Local module IP address (HH)
z0013C to z0013F	—	Unused	Unused
z00140	—	For self-diagnosis	Self-diagnosis register (use prohibited)
z00141 to z0014F	—	Unused	Unused

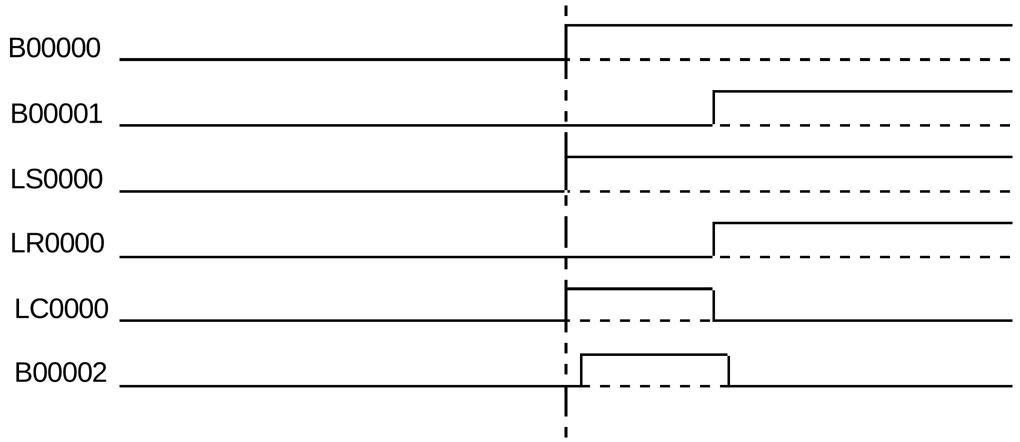
(4) CPU announce register (Continued 2)

Register	Relay	Name	Content
z00150	—	Execution time register	IO refresh execution time (Unit ms) (BCD)
z00151	—	Scan time register	IO refresh startup period (Unit ms) (BCD)
z00152	—	Execution time register	Task 1 execution time (Unit ms) (BCD)
z00153	—	Scan time register	Task 1 startup time (Unit ms) (BCD)
z00154	—	Execution time register	Task 2 execution time (Unit ms) (BCD)
z00155	—	Scan time register	Task 2 startup time (Unit ms) (BCD)
z00156	—	Execution time register	Task 3 execution time (Unit ms) (BCD)
z00157	—	Scan time register	Task 3 startup time (Unit ms) (BCD)
z00158	—	Execution time register	Task 4 execution time (Unit ms) (BCD)
z00159	—	Scan time register	Task 4 startup time (Unit ms) (BCD)
z0015A	—	Priority register	IO refresh task priority in RTOS
z0015B	—	Priority register	Task 1 task priority in RTOS
z0015C	—	Priority register	Task 2 task priority in RTOS
z0015D	—	Priority register	Task 3 task priority in RTOS
z0015E	—	Priority register	Task 4 task priority in RTOS
z0015F	—	Bank register	Bank register of the currently used program 1 or 2
zr0160	—	Scan time register	IO refresh startup time (Real number: Unit s)
zr0162	—	Scan time register	Task 1 startup time (Real number: Unit s)
zr0164	—	Scan time register	Task 2 startup time (Real number: Unit s)
zr0166	—	Scan time register	Task 3 startup time (Real number: Unit s)
zr0168	—	Scan time register	Task 4 startup time (Real number: Unit s)
zr016A to zr016E	—	Unused	Unused
zr016F	—	Program switching register	When switching programs: 1
zr0170	—	Execution time register	IO refresh execution time (Real number: Unit s)
zr0172	—	Execution time register	Task 1 execution time (Real number: Unit s)
zr0174	—	Execution time register	Task 2 execution time (Real number: Unit s)
zr0176	—	Execution time register	Task 3 execution time (Real number: Unit s)
zr0178	—	Execution time register	Task 4 execution time (Real number: Unit s)
zr017A to zr017F	—	Unused	Unused
z00180	—	Location of IO error	"00US" system configuration definition abnormality (No definition, implementation)
z00181	—	Location of IO error	"00US" system configuration definition abnormality (With definition, with no implementation)
z00182	—	Location of IO error	"00US" I/O module malfunction (IO ID Er)
z00183	—	Location of IO error	"00US" I/O module malfunction (IODef Er)
z00184	—	Location of IO error	"00US" common module malfunction (IOFaltEr)
z00185	—	Location of IO error	"00US" memory bus access malfunction (BusAccEr)
z00186	—	System count register	Flnet data transfer task start count
z00187	—	System count register	Flnet data transfer task startup period (μs)
z00188	—	System count register	NULL task start count
z00189	—	System count register	NULL task startup period (μs)
z0018A	—	System count register	IO refresh OS period

4.3 Outline of the Special Relay

(1) Latch relay/register





When set coil LS0000 is turned on, latch contact LC0000 is turned on, and O00020 remains on.

When reset coil LR0000 is turned on, latch contact LC0000 is turned off, and O00020 remains off.

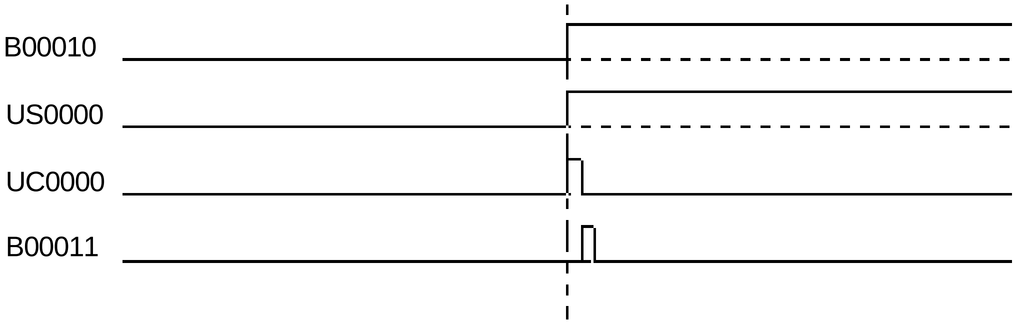
The latch contact LC0000 is delayed for one scan from latch coil.

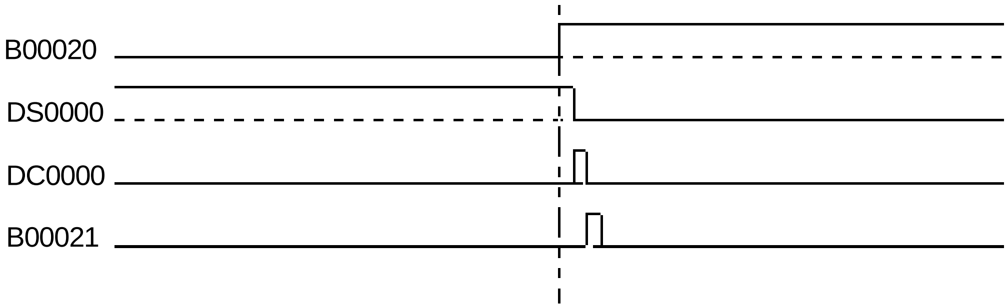
The latch coil is usually turned off when the power supply is opened.

To retain the latch coil even when the power supply is open, transfer it with the memory transfer definition using the retain memory, or use the SET RESET function (set the retain relay as the parameter).

To achieve the same functions within the subroutine, use the SET RESET function using SI0000 in the subroutine.

(2) On/off differential relay/register



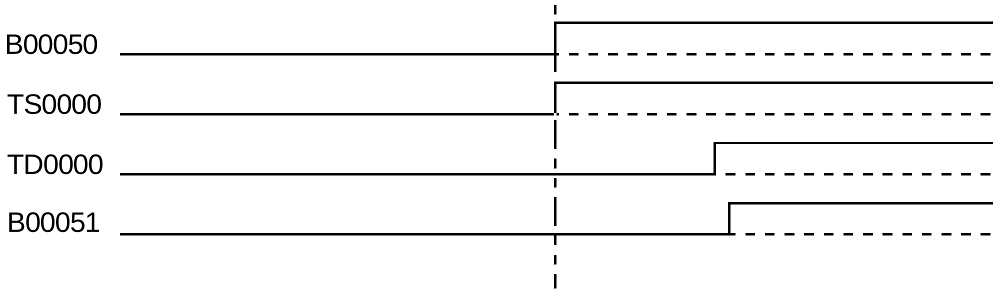


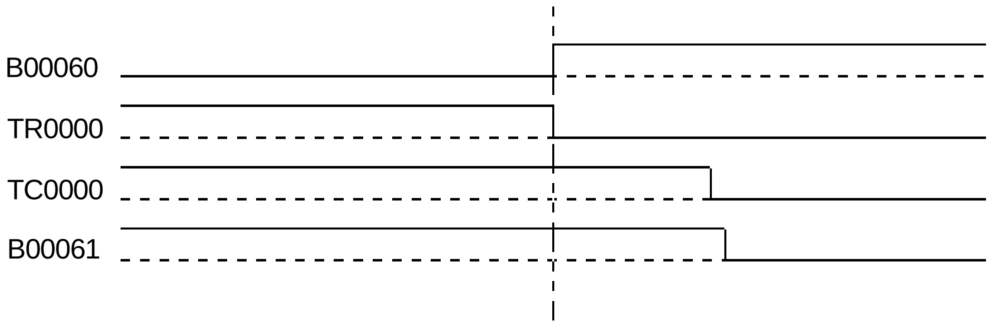
When coil US0000 is turned on, after a delay of one scan, differential contact UC0000 is turned on for one scan.

When coil DS0000 is turned off, after a delay of one scan, differential contact DC0000 is turned on for one scan.

There is also a USUC function and DSDC function to achieve the same functions.

(3) On/off timer relay/register





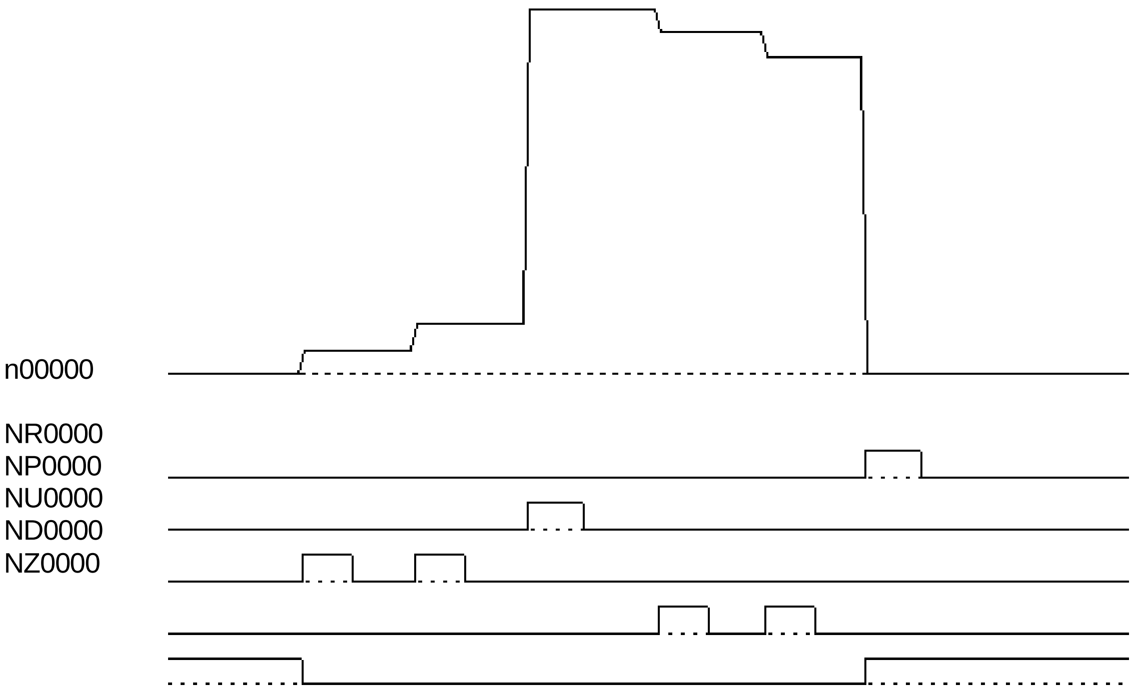
When coil TS0000 is turned on, after the set time has lapsed, timing contact TD0000 is turned on. TD0000 is turned off within one scan after TS0000 is turned off. (The timer setting value is input at the bottom of the TS coil.)

Here, S stands for second, M for minute and H for hour, and the setting range is from 0.01 seconds to 2 hours.

When coil TR0000 is turned on, timing contact TC0000 is turned on within one scan after TR0000 is turned ON. It is turned off after the set time has lapsed. (The timer setting value is input at the bottom of the TR coil.)

Here, S stands for second, M for minute and H for hour, and the setting range is from 0.01 seconds to 2 hours.

(4) Counter relay/register



The initial value of the counter is 0. Next, the up coil is turned on and the counter value increases by one. The zero detection contact is turned off with 0 initially, but since 1 has been added, it is no longer 0, so it is turned off.

In addition, the up coil is turned on and the counter value increases by 1, to 2.

The preset coil is turned on and the counter value becomes 15.

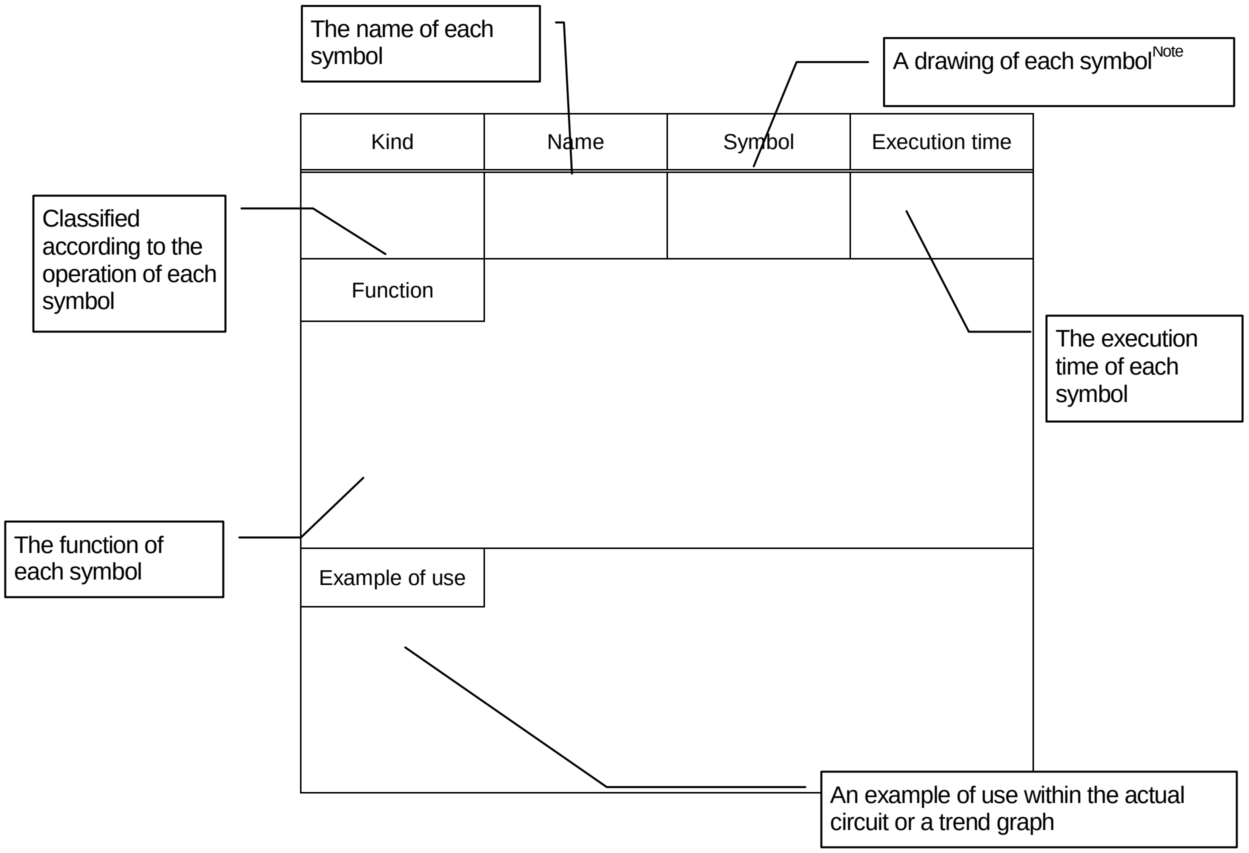
The preset value is set at the bottom of the NP coil.

The down coil is turned on, and the counter value decreases by 1.

The reset coil is turned on, the counter value becomes 0, and the zero detection contact is turned on.

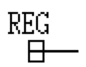
Chapter 5 Explanation of Instruction Words

How to read the table



Note: Relay and Reg that are displayed in the symbol column hereafter are explained below.

The figure on the left shows a relay. Here it is represented by the word RELAY for the sake of simplicity. All relays can be set as G0, I0, B0, etc.



The figure on the left shows a register. Here it is represented by the word REG for the sake of simplicity. All registers can be set as g0, mi, kr, etc.

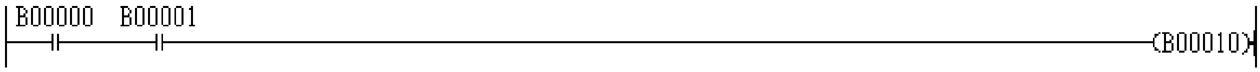
Kind	Name	Symbol	Execution time
LD language	A-contact		0.10 [μs]

Function	If RELAY is on, the input logic value is output. If it is off, the output logic value is turned off.

RELAY	A	B
On	On	On
On	Off	Off
Off	X	Off

X: Not applicable

Example of use



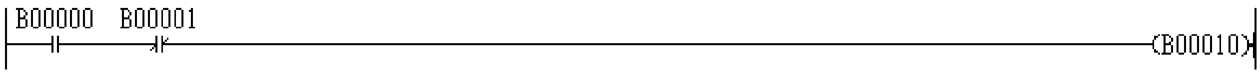
When both relay B00000 and relay B00001 are on, relay B00010 is turned on. Otherwise, relay B00010 is turned off.

Kind	Name	Symbol	Execution time
LD language	B-contact	 RELAY	0.12 [μs]
Function	If RELAY is off, the input logic value is output. If it is on, the output logic value is turned off.		

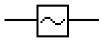
RELAY	A	B
Off	On	On
Off	Off	Off
On	X	Off

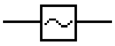
X: Not applicable

Example of use



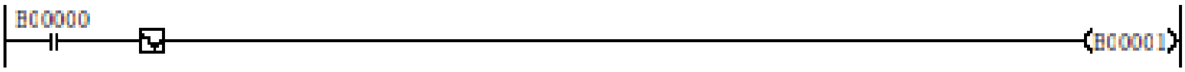
When relay B00000 is off and relay B00001 is off, relay B00010 is turned on. Otherwise, relay B00010 is turned off.

Kind	Name	Symbol	Execution time
LD language	Logical reversal		0.10 [μs]
Function	Logically reverses the input value.		

A  B

A	B
On	Off
Off	On

Example of use



When relay B00000 is on, relay B00001 is turned off.

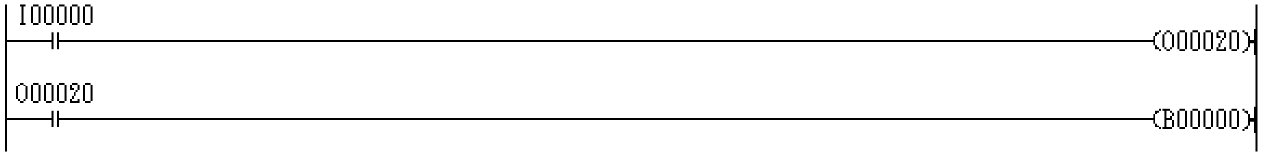
When relay B00000 is off, relay B00001 is turned on.

Kind	Name	Symbol	Execution time
LD language	Coil	—(RELAY)—	0.22 [μs]
Function	Outputs the input logic value to RELAY.		

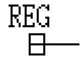
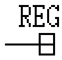
A —(RELAY)—

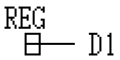
A	RELAY
On	On
Off	Off

Example of use

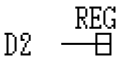


When relay I00000 is on, both relay O00020 and relay B00000 are turned on.
When relay I00000 is off, both relay O00020 and relay B00000 are turned off.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Load		Integer 0.16 [μs] Real number 0.20 [μs]
	Store		
Function	Load: The data in REG is set to the output numerical value. Store: The input numerical value is output to REG.		

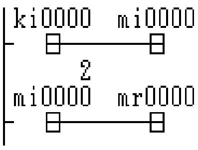


D1 = REG



REG = D2

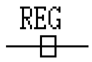
Example of use

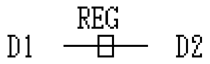


The data in register ki0000 (2) is loaded and stored in register mi0000.

Next, the data in register mi0000 is loaded and stored in register mr0000.

Since register mr0000 is a register of the real number type, it is converted from an integer to a real number and the data (2.0) is stored.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Store & load Store		Integer 0.19 [μs] Real number 0.14 [μs]
Function	The input numerical value is output to REG, and the data of REG is set as the output numerical value. It is used when data during an operation must be retained in REG.		



REG = D1

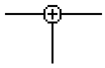
D2 = REG

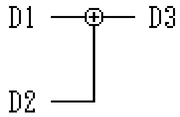
Example of use

The data in register mi0000 and the data in register mi0001 are added and the result is stored in register mi0002.

Next, the data in register mi0003 is subtracted from the data in register mi0002 and the result is stored in register mi0004.

The addition data during an operation is stored in register mi0002.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Addition		Integer 0.24 [μs] Real number 0.15 [μs]
Function	Two input numerical values are added and the result is output. The operation can be performed even if the types of value are different. However, an integer is converted to a real number, which is then subject to a real number operation.		



D3 = D1 + D2

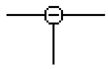
Type conversion

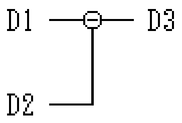
If the type of the register being used in one operation block is the integer type or the 16-bit BCD type, the data is converted to the 16-bit integer type before the operation, whereas if a register of the real number type, 32-bit integer type or 32-bit BCD type is used, it is converted to the real number type before the operation. (After this, type conversion is also carried out for subtraction, multiplication, division, remainder, priority given to a higher-level, and priority given to a lower-level.)

Example of use

The data in register mi0000 and the data in register mr0000 are added and the result is stored in register mr0001.

Although the data in register mi0000 is an integer, since the data in register mr0000 is a real number, addition is performed after type conversion of the integer/real number.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Subtraction		Integer 0.28 [μs] Real number 0.18 [μs]
Function	Subtraction is performed using two input numerical values and the result is output. The operation can be performed even if the types of value are different. However, an integer is converted to a real number, which is then subject to a real number operation.		

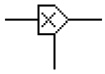


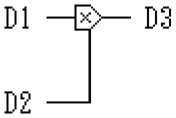
D3 = D1 - D2

Example of use

The data in register mr0000 is subtracted from the data in register mi0000 and the result is stored in register mr0001.

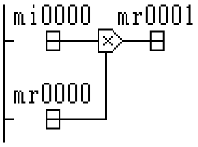
Although the data in register mi0000 is an integer, since the data in register mr0000 is a real number, subtraction is performed after type conversion of the integer/real number.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Multiplication		Integer 0.26 [μs] Real number 0.23 [μs]
Function	Two input numerical values are multiplied and the result is output. The operation can be performed even if the types of value are different. However, an integer is converted to a real number, which is then subject to a real number operation.		



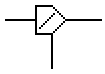
D3 = D1 * D2

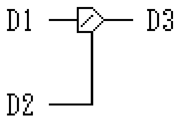
Example of use



The data in register mi0000 and the data in register mr0000 are multiplied and the result is stored in register mr0001.

Although the data in register mi0000 is an integer, since the data in register mr0000 is a real number, multiplication is performed after type conversion of the integer/real number.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Division		Integer 0.69 [μs] Real number 0.38 [μs]
Function	Division is performed using two input numerical values and the result is output. The operation can be performed even if the types of value are different. However, an integer is converted to a real number, which is then subject to a real number operation.		

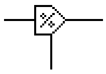


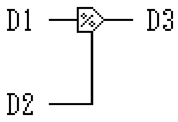
D3 = D1 / D2

Example of use

The data in register mi0000 and the data in register mr0000 are divided and the result is stored in register mr0001.

Although the data in register mi0000 is an integer, since the data in register mr0000 is a real number, division is performed after type conversion of the integer/real number.

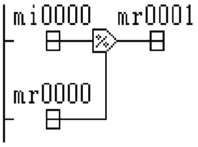
Kind	Name	Symbol	Execution time
Data flow language (Basic)	Remainder		0.64 [μs]
Function	Division is performed using two input numerical values and the result (remainder) is output.		



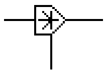
D3 = D1 % D2

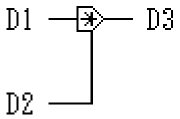
Note: Only operations with integers are valid.

Example of use



The data in register mi0000 is divided by the data in register mi0001 and the result (remainder) is stored in register mi0002.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Priority given to a upper-level		Integer 0.33 [μs] Real number 0.40 [μs]
Function	Two input numerical values are compared and the larger numerical value is output. The operation can be performed even if the types of value are different. However, an integer is converted to a real number, which is then subject to a real number operation.		



If D1 > D2 then D3 = D1

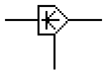
If D1 ≤ D2 then D3 = D2

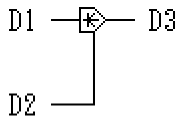
Example of use

The data in register mi0000 and the data 100.0 in register kr0000 are compared and the larger data is stored in register mr0001.

Although the data in register mi0000 is an integer, since the data in register kr0000 is a real number, comparison is performed after type conversion of the integer/real number.

It serves as a limiter for which the lower limit value is the data in register kr0000 (100.0).

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Priority given to a lower-level		Integer 0.40 [μs] Real number 0.28 [μs]
Function	Two input numerical values are compared and the smaller numerical value is output. The operation can be performed even if the types of value are different. However, an integer is converted to a real number, which is then subject to a real number operation.		



If D1 > D2 then D3 = D2

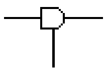
If D1 ≤ D2 then D3 = D1

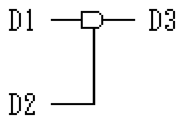
Example of use

The data in register mi0000 and the data 100.0 in register kr0000 are compared and the smaller data is stored in register mr0001.

Although the data in register mi0000 is an integer, since the data in register kr0000 is a real number, comparison is performed after type conversion of the integer/real number.

It serves as a limiter for which the upper limit value is the data in register kr0000 (100.0).

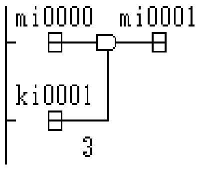
Kind	Name	Symbol	Execution time
Data flow language (Basic)	Product of numerical values		0.33 [μs]
Function	A logical multiplication operation is performed using two input numerical values and the result is output.		



D3 = D1 & D2

Note: Only operations with integers are valid.

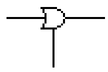
Example of use



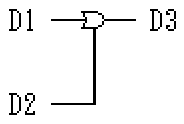
A logical multiplication operation is performed using the data in register mi0000 and the data in register ki0001 (3) and the result is stored in register mi0001.

If the data in register mi0000 is (10), then (2) is stored in register mi0001.

mi0000	0000	0000	0000	1010	(10)
ki0000	0000	0000	0000	0011	(3)
mi0001	0000	0000	0000	0010	(2)

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Sum of numerical values		0.32 [μs]

Function	A logical sum operation is performed using two input numerical values and the result is output.



D3 = D1 | D2

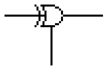
Note: Only operations with integers are valid.

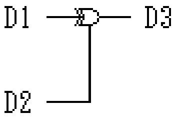
Example of use

A logical sum operation is performed using the data in register mi0000 and the data in register ki0001 (3) and the result is stored in register mi0001.

If the data in register mi0000 is (10), then (11) is stored in register mi0001.

mi0000	0000	0000	0000	1010	(10)
ki0000	0000	0000	0000	0011	(3)
mi0001	0000	0000	0000	1011	(11)

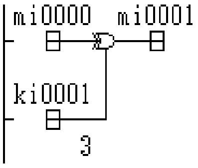
Kind	Name	Symbol	Execution time
Data flow language (Basic)	Exclusive OR of numerical values		0.31 [μs]
Function	An exclusive OR operation is performed using two input numerical values and the result is output.		



D3 = D1 ˆ D2

Note: Only operations with integers are valid.

Example of use



An exclusive OR operation is performed using the data in register mi0000 and the data in register ki0001 (3) and the result is stored in register mi0001.

If the data in register mi0000 is (10), then (9) is stored in register mi0001.

mi0000	0000	0000	0000	1010	(10)
ki0000	0000	0000	0000	0011	(3)
mi0001	0000	0000	0000	1001	(9)

Kind	Name	Symbol	Execution time
Data flow language (Basic)	a-contact		Integer 0.34 [μs] Real number 0.36 [μs]
Function	If RELAY is on, the input numerical value is output. If it is off, the output numerical value is 0.		

If RELAY = on then D2 = D1

If RELAY = off then D2 = 0

Example of use

When relay I00000 is on, the data in register mi0000 is stored in register mi0001.

When relay I00000 is off, (0) is stored in register mi0001.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	b-contact		Integer 0.50 [μs] Real number 0.38 [μs]
Function	If RELAY is off, the input numerical value is output. If it is on, the output numerical value is 0.		

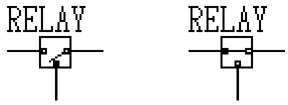
If RELAY = on then D2 = 0

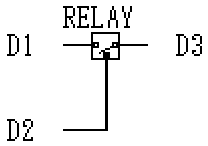
If RELAY = off then D2 = D1

Example of use

When relay I00000 is off, the data in register mi0000 is stored in register mi0001.

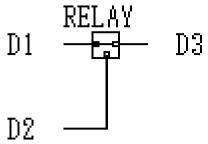
When relay I00000 is on, (0) is stored in register mi0001.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	c-contact		Integer 0.39 [μs] Real number 0.33 [μs]
Function	Depending on the logical value of RELAY, either of the two input numerical values is selected and output.		



If RELAY = on then D3 = D1

If RELAY = off then D3 = D2



If RELAY = on then D3 = D2

If RELAY = off then D3 = D1

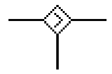
Example of use

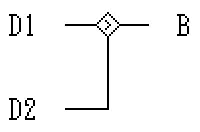
When relay I00000 is off, the data in register mi0001 is selected and stored in register mi0002.

When relay I00000 is on, the data in register mi0000 is selected and stored in register mi0002.

When relay I00000 is off, the data (3) in register ki0000 is selected and stored in register mi0003.

When relay I00000 is on, the data (6) in register ki0001 is selected and stored in register mi0003.

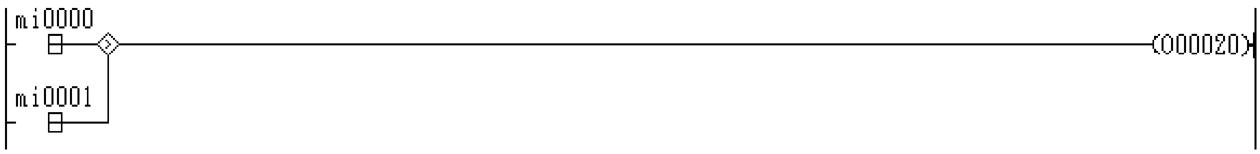
Kind	Name	Symbol	Execution time
Data flow language (Basic)	Compare high		Integer 0.17 [μs] Real number 0.13 [μs]
Function	A comparison of two input numerical values is performed and the result is output as a logical value.		



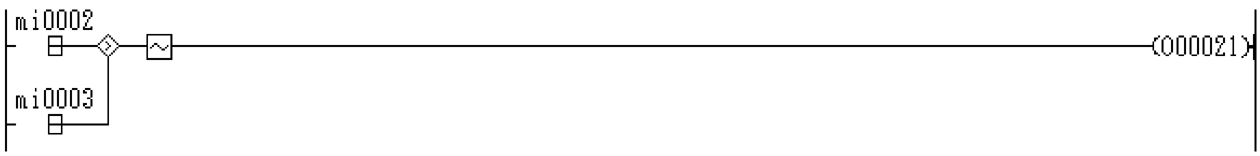
If D1 > D2 then B = on

If D1 ≤ D2 then B = off

Example of use

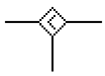


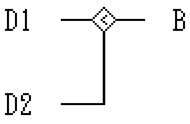
If the data in register mi0000 is greater than the data in mi0001, relay O00020 is turned on. Otherwise, relay O0020 is turned off.



It can change the logic in combination with logical reversal.

If the data in register mi0002 is equal to the data in mi0003 or smaller than the data in mi0003, then relay O00021 is turned on. Otherwise, relay O00021 is turned off.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Compare low		Integer 0.17 [μs] Real number 0.13 [μs]
Function	A comparison of two input numerical values is performed and the result is output as a logical value.		

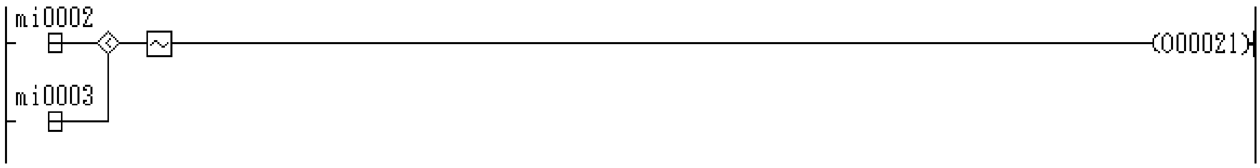


If D1 < D2 then B = on

If D1 ≥ D2 then B = off

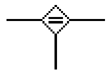
Example of use

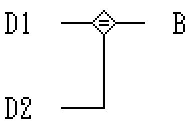
If the data in register mi0000 is smaller than the data in mi0001, relay O00020 is turned on.
Otherwise, relay O0020 is turned off.



It can change the logic in combination with logical reversal.

If the data in register mi0002 is equal to the data in mi0003 or greater than the data in mi0003, then relay O00021 is turned on. Otherwise, relay O00021 is turned off.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Compare equal		Integer 0.18 [μs] Real number 0.11 [μs]
Function	A comparison of two input numerical values is performed and the result is output as a logical value.		



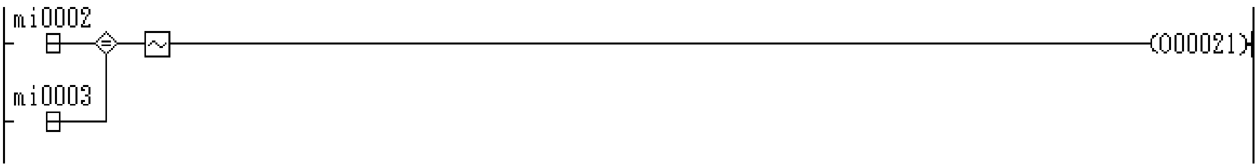
If D1 = D2 then B = on

If D1 ≠ D2 then B = off

Note: If a real number is in the register used, then in some cases the relay may not be turned on since the numerical value is too small to register.

Example of use

If the data in register mi0000 is equal to the data in mi0001, relay O00020 is turned on.
Otherwise, relay O0020 is turned off.



It can change the logic in combination with logical reversal.

If the data in register mi0002 is not equal to the data in mi0001, relay O00020 is turned on.
Otherwise, relay O00021 is turned off.

Kind	Name	Symbol	Execution time
Data flow language (Basic)	Load local constant (Integer, real number)		Integer 0.24 [μs] Real number 0.21 [μs]

Function	A local constant (integer or real number) is loaded.

The constant is held in the program (instead of the parameter).

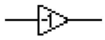
The load local constant (integer) can only be used within the operation block of i-form.

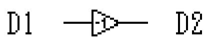
An integer and real number cannot both be present within a single operation block.

Example of use

The integer value (10) is loaded in register mi0000.

The real number value (5.0000) is loaded in register mr0000.

Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Code conversion		Integer 0.18 [μs] Real number 0.12 [μs]
Function	The positive/negative sign of input numerical values is reversed and output.		

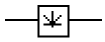


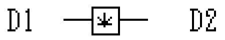
D2 = - (D1)

Example of use

The sign of the data (-10) in register ki0000 is converted to positive and (10) is stored in register mi0000.

The sign of the data (5.0000) in register kr0000 is converted to negative and (-5.0000) is stored in register mr0000.

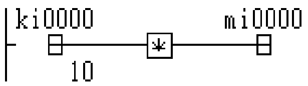
Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Absolute value conversion		Integer 0.30 [μs] Real number 0.20 [μs]
Function	Obtains the absolute value of the input numerical value and outputs it.		



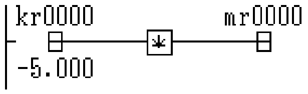
If D1 < 0 then D2 = -(D1)

If D1 ≥ 0 then D2 = D1

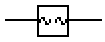
Example of use

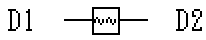


The sign of the data (10) in register ki0000 is converted to an absolute value and (10) is stored in register mi0000.



The sign of the data (-5.0000) in register kr0000 is converted to an absolute value and (5.0000) is stored in register mr0000.

Kind	Name	Symbol	Execution time
Data flow language (Function 1)	1’s complement		0.19 [μs]
Function	A complement operation is performed using the input numerical value and the result is output.		



D2 = NOT (D1)

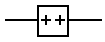
Note: Only operations with integers are valid.

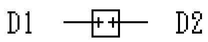
Example of use

A one’s complement operation is performed using the data in register mi0000 and the result is stored in register mi0001.

If the data in register mi0000 is (10), then (-11) is stored in register mi0002.

mi0000	0000	0000	0000	1010	(10)
mi0001	1111	1111	1111	0101	(-11)

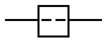
Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Increment		Integer 0.19 [μs] Real number 0.14 [μs]
Function	1 is added to the input numerical value and the result is output.		

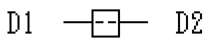


D2 = D1 + 1
(D2 = D1 + +)

Example of use

(1) is added to the data (10) in register ki0000 and the result (11) is stored in register mi0000.

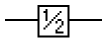
Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Decrement		Integer 0.23 [μs] Real number 0.16 [μs]
Function	1 is subtracted from the input numerical value and the result is output.		

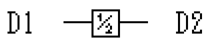


D2 = D1-1
(D2 = D1 - -)

Example of use

(1) is added to the data (10) in register ki0000 and the result (9) is stored in register mi0000.

Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Half		0.20 [μs]
Function	The result of multiplying the input numerical value by one half is output.		



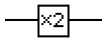
D2 = D1 / 2

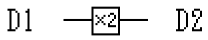
Note: Only operations with integers are valid.

Example of use

The data (10) in register ki0000 is halved and the result (5) is stored in register mi0000.

This instruction is used when the data in an integer register is signed and multiplied by one half.

Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Two times		0.17 [μs]
Function	The result of multiplying the input numerical value by two is output.		



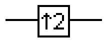
D2 = D1 * 2

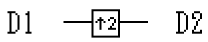
Note: Only operations with integers are valid.

Example of use

The data (10) in register ki0000 is multiplied by 2 and the result (20) is stored in register mi0000.

This instruction is used when the data in an integer register is signed and multiplied by 2.

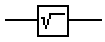
Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Second power		Integer 0.25 [μs] Real number 0.14 [μs]
Function	The result of obtaining the second power of the input numerical value is output.		



D2 = D1 * * 2
(D2 = D1^2)

Example of use

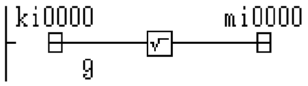
The data (10) in register ki0000 is multiplied by itself and the result (100) is stored in register mi0000.

Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Square root		Integer 0.36 [μs] Real number 0.33 [μs]
Function	The square root of the input numerical value is output.		

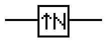
D2 = SQRT (D1)

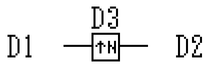
Note: When the input value is a negative value, the output also takes a negative value.

Example of use



The square root of the data (9) in register ki0000 is obtained and the result (3) is stored in register mi0000.

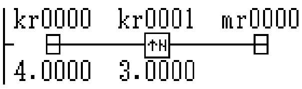
Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Exponential function		3.60 [μs]
Function	An exponential operation is performed using the input numerical value and the result is output.		



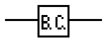
D2 = D3 * * D1
($D2 = D3^{D1}$)

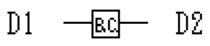
Note: Only operations with real numbers are valid.

Example of use



An exponential operation is performed using the data (4.0000) in register kr0000 with the data (3.0000) in register kr0001 as its exponent and the result (64) is stored in register mr0000.

Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Bit count		0.73 [μs]
Function	Reads the input numerical value as a 16-bit binary number, and outputs the number of bits that are on.		



Note: Only operations with integers are valid.

Example of use

The data (1234) in register ki0000 is read as a 16-bit binary number, the number of bits that are on (bits that are 1) is calculated, and the result (5) is stored in register mi0000.

ki0000	0000	0001	1010	1010	(1234)
mi0001	0+ 1	+ 2	+ 2		= 5

Kind	Name	Symbol	Execution time
Data flow language (Function 1)	Gray code binary	—[G.B]—	16.1 [μs]

Function	The input numerical value (Gray code) is converted and the result is output as a binary number.

Since in the Gray code, only one bit changes as the numerical value changes, it is used in positioning control, etc.

D1 —[G.B]— D2

The bit pattern of 0 to 15 is as follows.

D2	D1	D2	D1	D2	D1	D2	D1
Integer	Gray	Integer	Gray	Integer	Gray	Integer	Gray
0000	0000	0100	0110	1000	1100	1100	1010
0001	0001	0101	0111	1001	1101	1101	1011
0010	0011	0110	0101	1010	1111	1110	1001
0011	0010	0111	0100	1011	1110	1111	1000

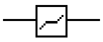
Note: Only operations with integers are valid.

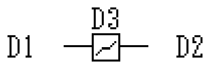
Example of use

Gray code conversion is performed using the data in register mi0000 and the result is stored in mi0001.

If the data in register mi0000 is (10), then (12) is stored in register mi0001.

10	→	1010	→	1100	→	12
↑ Input		↑ Gray code		↑ Integer		↑ Output

Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Insensitive band		Integer 0.27 [μs] Real number 0.21 [μs]
Function	If the input numerical value is within the range of the insensitive band, 0 is output. If the input numerical value is out of the range of the insensitive band, then the insensitive value (absolute value) is subtracted from it and the result is output.		



If -D3 < D1 < +D3 then D2 = 0

If +D3 ≤ D1 then D2 = D1 – D3

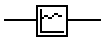
If -D3 ≥ D1 then D2 = D1 + D3

Example of use

If the data in register mi0000 is greater than the data obtained by sign conversion from the data (-10) in register ki0000, and is smaller than the positive data (10), then (0) is stored in register mi0001.

If the data in register mi0000 is equal to, or greater than the data (10) in register ki0000, then the result of subtracting the data (10) in register ki0000 from the data in register mi0000 is stored in register mi0001.

If the data in register mi0000 is equal to, or smaller than the data obtained by sign conversion from the data (-10) in register ki0000, then the result of adding the data (-10) in register ki0000 and the data in register mi0000 is stored in register mi0001.

Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Pattern		Integer 1.70 [μs] Real number 1.50 [μs]

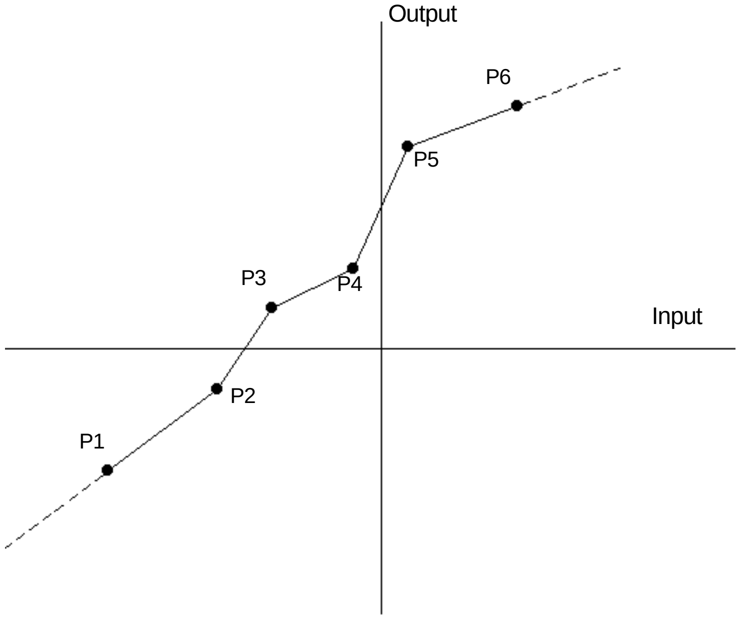
Function	Approximation conversion is performed using the input numerical value by line segmentation with pattern memory and the result is output.

The pattern data is set beforehand by the pattern data in the tool. The horizontal axis data must be arranged from the smaller data to the larger data.

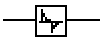
The horizontal axis corresponds to the input value of a function. Even if data that deviates from the pattern data is input, it is converted by extending the line of the pattern data, and is then output.

Graph

If the input is smaller than P1, it is converted to an approximation straight line obtained by extending straight line P1-P2 and the result is output. If it is greater than P6, it is likewise converted to an approximation straight line obtained by extending straight line P5-P6 and the result is output.



	Input	Output
P1/Q1	-10	-3
P2/Q2	-6	-1
P3/Q3	-4	1
P4/Q4	-1	2
P5/Q5	1	5
P6/Q6	5	6

Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Differential compensation		1.40 [μs]
Function	Three averages are taken of the time differential values of the input numerical value and the result is output.		

Settings of the function argument

(1) Differential gain: Differential coefficient in second units (When the change in input is 1.0 per second, 1.0 is output.)

For the sake of safety, averaging is performed to prevent rapid changes.

In addition to krxxxx, mrxxxx can also be used as the operation parameter, in which case each parameter should be set in the user program.

Note: Only operations with real numbers are valid.

Graph

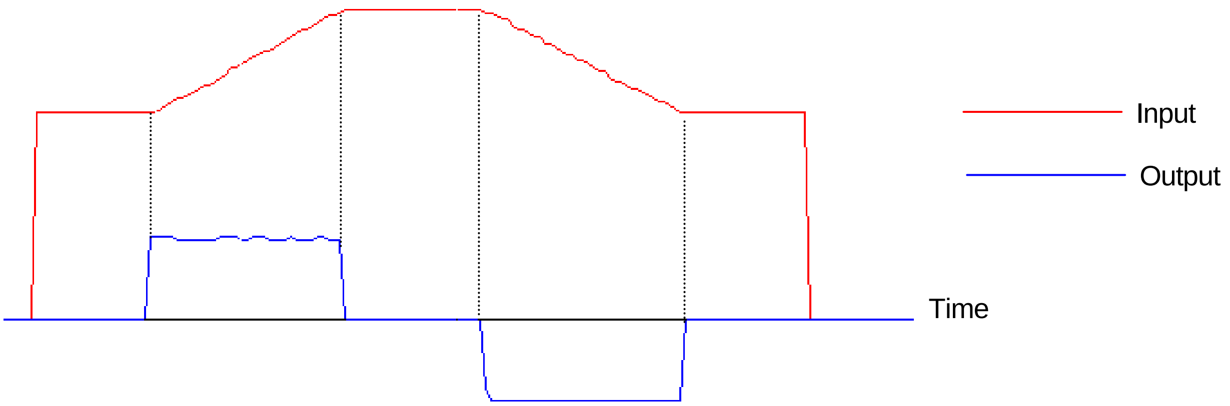
When the function argument is set as shown at right, the resulting trend graph is as shown below.

Differential compensation

Differential gain	kr0000	10.000

Where the input value is constant (gradient = 0), the differential value is also 0, and so the output is 0. The output value changes only where the input value is always changing.

Note: In the trend graph below, the rapidly changing part is not shown.



Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Phase compensation	θ	1.47 [μs]
Function	Phase compensation is performed using the input numerical value and the result is output.		

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) Phase gain (A): Advanced phase or lagged phase is set depending on whether or not the value is greater than 1.0.

(3) Time gain (T): Time coefficient in seconds

In addition to krxxxx, mrxxxx can also be used as the operation parameter, in which case each parameter should be set in the user program.

When reset is on, the input and output are short-circuited so that an arbitrary value can be preset.

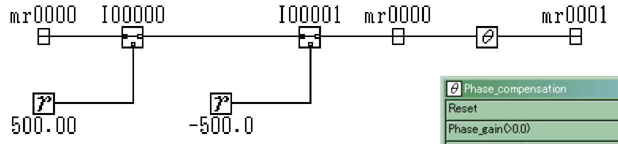
Note: Only operations with real numbers are valid.

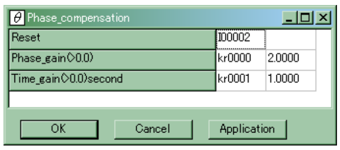
Graph

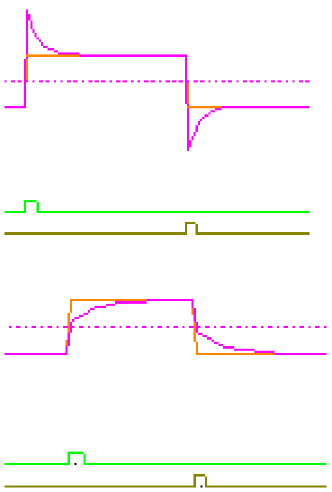
When the function argument is set as shown at right, the resulting trend graph is as shown below. Depending on the time gain, the output value approaches the input value so that the size of the curve changes. When the gain is small, a small arc is drawn, and when it is large, a large arc is drawn.

Scan time: 10 ms

Trend sampling time: Example at 100 ms





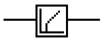


Phase gain (A) < Time gain (T)

Reset	G00000	
Phase gain (A1)	kr0000	0.5000
Time gain	kr0001	1.0000

Phase gain (A) > Time gain (T)

Reset	G00000	
Phase gain (A1)	kr0000	2.0000
Time gain	kr0001	0.5000

Kind	Name	Symbol	Execution time
Data flow language (Function 2)	PI compensation		2.53 [μs]

Function

PI compensation (proportioning, integration) is performed using the input numerical value and the result is output.

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) Hold: Integration hold switch (stops integration)

(3) Proportioning gain:

(4) Integral gain: Integral coefficient in second units

(5) Upper limit value: Designate the upper limit value to be output

(6) Lower limit value: Designate the lower limit value to be output.

In addition to krxxxx, mrxxxx can also be used as the operation parameter, in which case each parameter should be set in the user program.

When reset is on, the input and output are short-circuited so that an arbitrary value can be preset.

Note: Only operations with real numbers are valid.

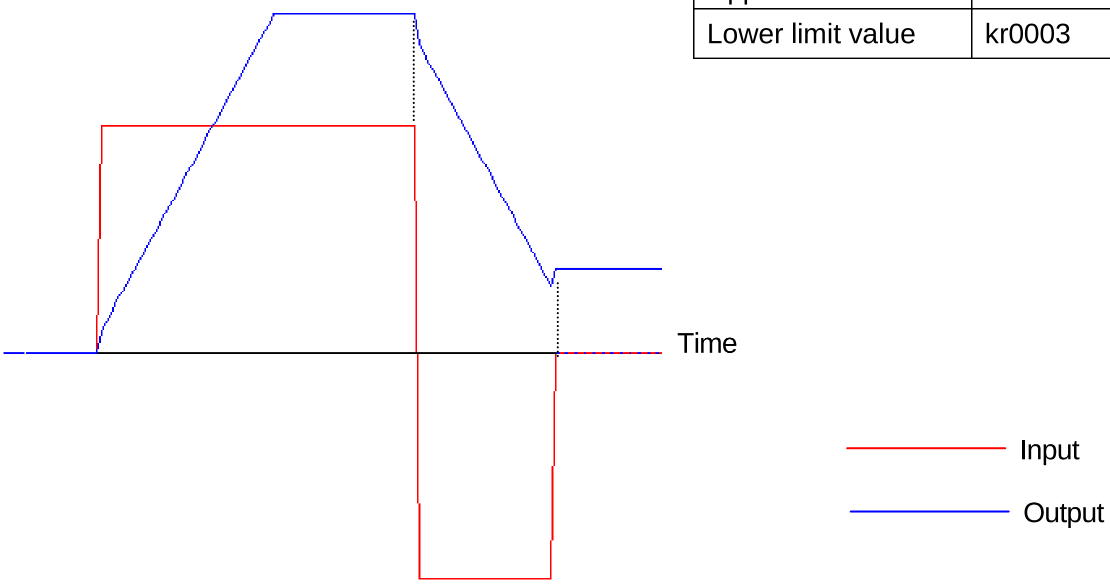
Graph

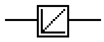
When the function argument is set as shown at right, the resulting trend graph is as shown below.

Depending on the proportioning gain, the output value at the start changes, and depending on the integral gain, the gradient of the output value changes.

PI compensation

Reset	G00000	
Hold	G00001	
Proportioning gain	kr0000	0.1000
Integral gain	kr0001	3.0000
Upper limit value	kr0002	30.000
Lower limit value	kr0003	-30.000



Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Limitation on the change ratio in a straight line form		1.08 [μs]

Function

Time rate of change limitation is performed on the input numerical value and the result is output.

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) Maximum rising ratio: (> 0.0: Positive value): Limitation value of the rising ratio of output per second
(Example: 10.0 = Permits a rise of 10 or less per second)

(3) Maximum falling ratio: (< 0.0: Negative value): Limitation value of the falling ratio of output per second
(Example: -10.0 = Permits a fall of 10 or less per second)

In addition to krxxxx, mrxxxx can also be used as the operation parameter, in which case each parameter should be set in the user program.

When reset is on, the input and output are short-circuited so that an arbitrary value can be preset.

Note: Only operations with real numbers are valid.

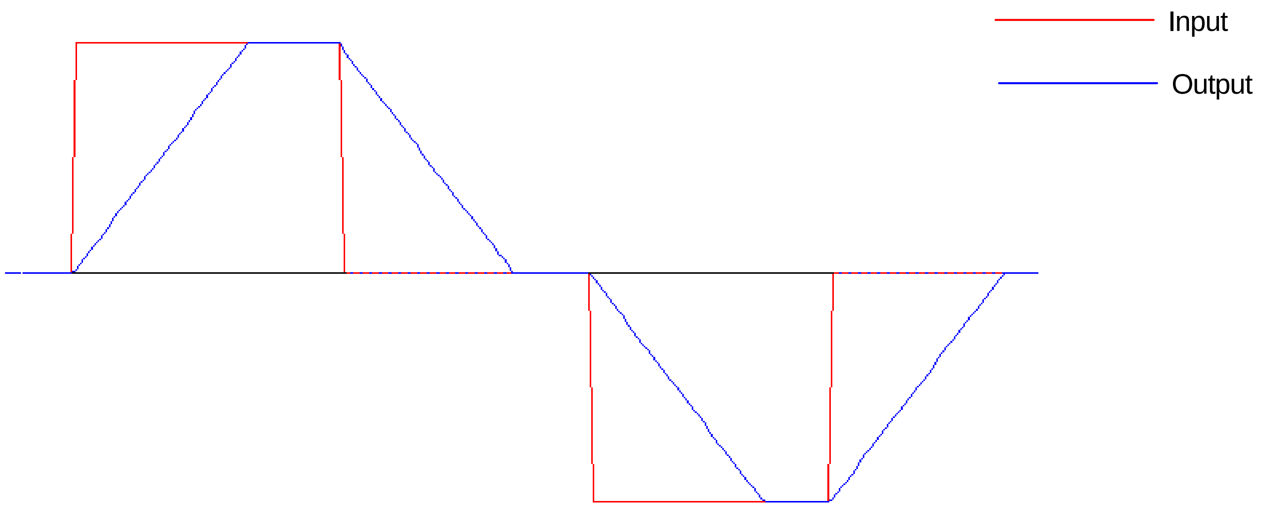
Graph

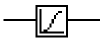
When the function argument is set as shown at right, the resulting trend graph is as shown below.

Depending on the rising or falling ratio, the gradient of the output value can be set (if the step input has been added).

Limitation on the change ratio in a straight line form

Reset	G00000	
Maximum rising ratio	kr0000	0.1000
Maximum falling ratio	kr0001	-0.1000



Kind	Name	Symbol	Execution time
Data flow language (Function 2)	S-form change ratio limitation (S-ARC)		4.01 [μs]

Function	S-form change ratio limitation is performed on the input numerical value and the result is output.

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) Maximum rising ratio: (> 0.0): Limitation value of the rising ratio of output per second

(3) Maximum falling ratio: (< 0.0): Limitation value of the falling ratio of output per second

(4) Increasing-rising ratio (> 0.0): Acceleration increasing value per second when acceleration starts

(5) Decreasing-rising ratio (> 0.0): Acceleration decreasing value per second when acceleration ceases

(6) Decreasing-decreasing ratio (> 0.0): Deceleration value per second when acceleration finishes

(7) Increasing-decreasing ratio (< 0.0): Deceleration increasing value per second when deceleration starts

(8) S-form acceleration/deceleration ceasing coefficient (> 0.0): Change ratio limitation value when the acceleration/deceleration has ceased

Usually, it should be set at twice the value obtained by choosing the largest of the absolute values of (4) to (7).

When reset is on, the input and output are short-circuited so that an arbitrary value can be preset.

Note: Only operations with real numbers are valid.

Graph

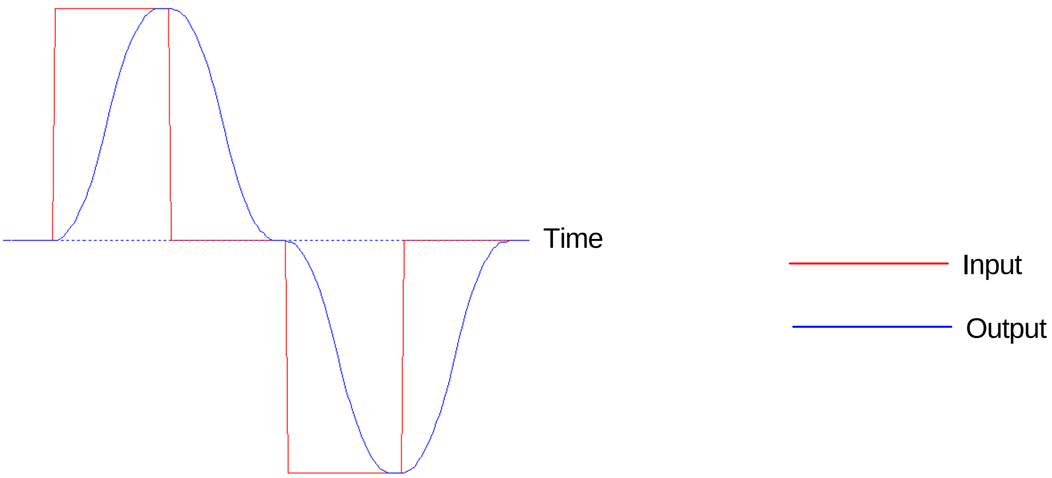
When the function argument is set as shown at right, the resulting trend graph is as shown below.

Although the graph is the same as ARC, since the curve before the straight line (B1 to 4) is also set, an S-shaped waveform is output.

Note: If the input value is changed while accelerating or decelerating, overshooting may occur.

S-form change ratio limitation

Reset	G00000	
Maximum rising ratio	kr0000	10.000
Maximum falling ratio	kr0001	-10.000
Increasing-rising ratio	kr0002	0.020
Decreasing-rising ratio	kr0003	-0.020
Decreasing-decreasing ratio	kr0004	0.0020
Increasing-decreasing ratio	kr0005	-0.0020
S-form acceleration/ deceleration ceasing coefficient	kr0006	0.0040



Kind	Name	Symbol	Execution time
Data flow language (Function 3)	Trigonometric function Inverse trigonometric function		SIN 7.4 [μs] COS 7.1 [μs] TAN 7.3 [μs] ASIN 7.2 [μs] ACOS 7.2 [μs] ATAN 9.3 [μs]
Function	A trigonometric function (inverse trigonometric function) operation is performed using the input numerical value and the result is output.		

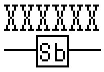
sin function	D1 —[f]— D2 (SIN)	D2 = sin (D1)
cos function	D1 —[f]— D2 (COS)	D2 = cos (D1)
tan function	D1 —[f]— D2 (TAN)	D2 = tan (D1)
asin function	D1 —[f]— D2 (ASIN)	D2 = $\sin^{-1}$ (D1)
acos function	D1 —[f]— D2 (ACOS)	D2 = $\cos^{-1}$ (D1)
atan function	D1 —[f]— D2 (ATAN)	D2 = $\tan^{-1}$ (D1)

Note: Only operations with real numbers are valid.

Example of use

mr0001 = SIN (mr0000)

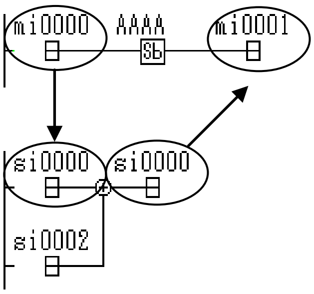
The sine of the data in register mr0000 is obtained and the result is stored in register mr0001.

Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Unconditional subroutine		6.64 [μs]
Function	Executes subroutines unconditionally.		

By double clicking on the symbol, a window for setting an argument appears and you can set an argument for the subroutine. In the subroutine, exchange of data is performed by the stack registers (sr0000, si0000, SI0000). The stack registers are set in the window for setting an argument. The actual data flow is as follows.

Input data		Stack register		Output data
Integer data	→	si0000	→	Integer data
Real data	→	sr0000	→	Real data
Relay/coil	→	SI0000	→	Relay/coil
Invoker		Subroutine		Invoker

Example of use



Subroutine AAAA is executed unconditionally. Registers mi0000 and mi0001 are conventionally used, and to use these data as well, the data in register mi0000 is passed to stack register si0000 of subroutine AAAA. When the data calculated in subroutine AAAA is stored in stack register si0000, the data is stored in register mi0001.

However, if they are not used in subroutine AAAA, the data in mi0000 is stored in mi0001.

Kind	Name	Symbol	Execution time
LD language	Jump instruction	—(JPXXXX)┤	—
	Label instruction	XXXXXX [L]—	
Function	Jump: Jumps to the designated circuit or designated label Label: Used to label the destination of a jump.		

This is regarded as one of the logic circuits.

XXXX stands for the circuit number or label name (4 digits).

Note 1: A jump cannot be performed between subprograms or subroutines.

Note 2: You can also create a program that loops at one point, but it must not be a permanent loop.

Note 3: Place a register store to the right of the label.

Example of use

When relay B00000 is ON, a jump is made to the line of label ABCD, and the programs between it and label ABCD are not executed.

When relay B00000 is on, the data (10.000) in register kr0000 is stored in register mr0000 and 1 is stored in register mi0000.

When relay B00000 is off, the data in register kr0000 (10.000) is not stored in register mr0000 and 0 is stored in register mi0000.

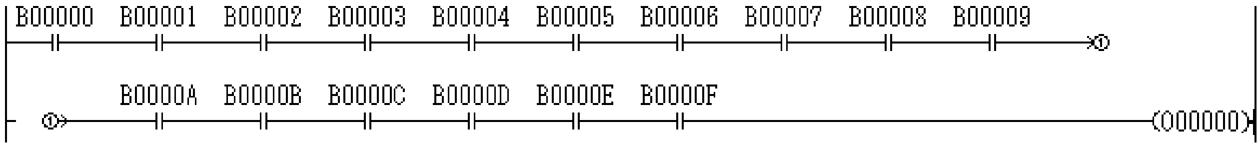
Kind	Name	Symbol	Execution time
LD language	Connective (Store)	─→①	0.16 [μs]
	Connective (Load)	①→─	0.16 [μs]
Function	Performs storing and loading of the result of logical and numerical operations, to and from the intermediate memory.		

It is used when there are twelve or more logical codes or numerical codes arranged in series.

It must be placed between networks.

Ten sets of symbols can be inserted into a single circuit. Loading must be performed after storing.

Example of use

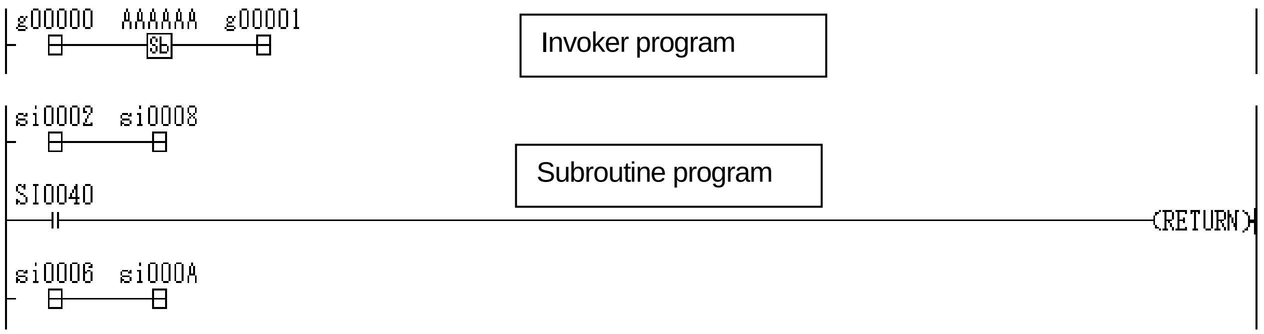


Kind	Name	Symbol	Execution time
LD language	Termination of processing of a subroutine program	—(RETURN)—	2.00 [μs]

Function	Terminates processing of the subroutine program.

This is used to terminate a subroutine program under a certain condition.

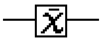
Example of use



When relay I00000 is off, the data ki0000 (5) in stack register si0002 is stored in stack register si0008 and data (5) is loaded in register mi0000. The data z0009 in stack register si0006 is stored in stack register si000A and loaded in register mi0001.

However, when relay I00000 is on, although the data (5) in stack register si0002 is stored as it is in stack register si0008, the data in stack register si0006 when I0000 is turned on is stored in si000A and remains there. (Since z00009 is a millisecond counter, if the relay is turned on when it is 100, then 100 is stored in si000A. If I0000 is turned on, then the data in si0006 is stored there.)

Argument	Label	Value
si0002	ki0000	5
SI0040	I00000	
si0006	z00009	
si0008	mi0000	
si000A	mi0001	

Kind	Name	Symbol	Execution time
Data flow language (Function 2)	(Arithmetic) average		————————
Function	The arithmetic average value of the data of the input numerical value is obtained from the start address set by the argument and the result is output.		

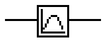
Settings of the function argument

(1) Start of buffer addresses (mrXXXX): If the input is smaller than 1, it is regarded as 1 and the value of the first data is returned.

Example of use

Argument of arithmetic average Buffer start address: mr0000

If the setting above is made, the arithmetic average reads the data (5.0000) in register kr0000 and the argument, and the result (12,000) of the operation (mr0000 + mr0001 + mr0002 + mr0003 + mr0004)/5 is stored in register gr0000.

Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Filter		2.42 [μs]
Function	Frequency limitation is performed using the input numerical value and the result is output.		

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) Lower limit frequency (>0.0: Positive value): Lower limit frequency of 3 db decrease

(3) Upper limit frequency (>0.0: Positive value): Upper limit frequency of 3 db decrease

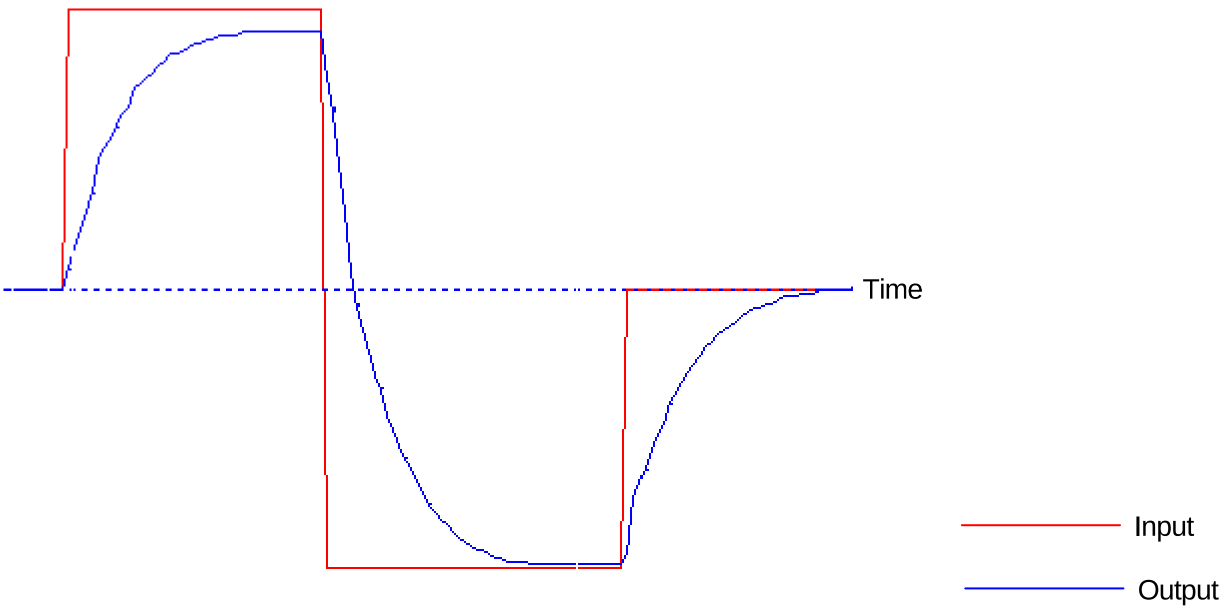
Note: Only operations with real numbers are valid.

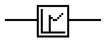
Graph

When the function argument is set as shown at right, the resulting trend graph is as shown below.

Filter

Reset	G00000	
Lower limit frequency	kr0000	0.0001
Upper limit frequency	kr0001	0.0500



Kind	Name	Symbol	Execution time
Data flow language (Function 2)	PID compensation		2.36 [μs]
Function	PID compensation is performed using the input numerical value and the result is output.		

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) Hold: Integration stop switch

(3) Zero clear: Designates a relay that commands the zero reset.

(4) Proportioning gain:

(5) Integral gain: Integral coefficient in second units system (the time until the output value reaches the input value: Seconds)

(6) Differential gain: Differential coefficient in second units (When the change in input is 1.0 per second, 1.0 is output.)

(7) MAX limit value: Designate the upper limit value to be output

(8) MIN limit value: Designate the lower limit value to be output

When reset is on, the input and output are short-circuited so that an arbitrary value can be preset.

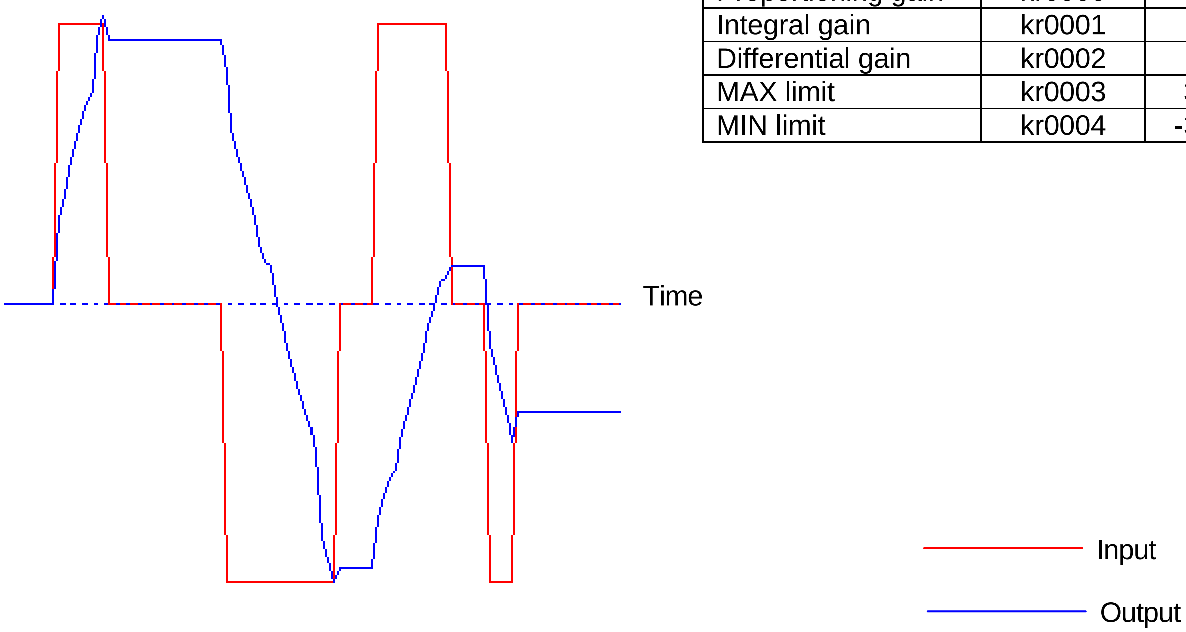
Note: Only operations with real numbers are valid.

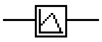
Graph

When the function argument is set as shown at right, the resulting trend graph is as shown below.

Filter

Reset	G00000	
Hold	G00001	
Zero clear	G00002	
Proportioning gain	kr0000	0.1000
Integral gain	kr0001	3.0000
Differential gain	kr0002	0.0100
MAX limit	kr0003	30.000
MIN limit	kr0004	-30.000



Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Temporary delay		1.70 [μs]

Function	Outputs a temporary delay response for the input numerical value.

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) Time constant: T seconds

The reset switch must be set to on when starting the operation.

Note: Only operations with real numbers are valid.

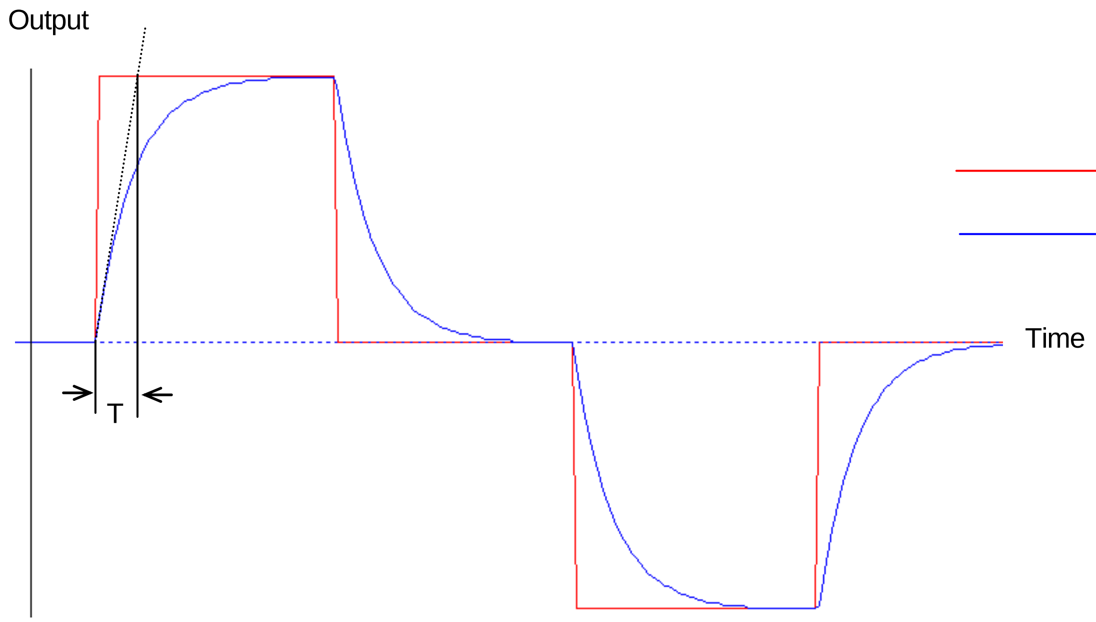
Graph

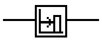
When the function argument is set as shown at right, the resulting trend graph is as shown below.

During the period when the input is changed by the time constant, the output values are plotted to draw an arc.

Filter

Reset	G00000	
Time constant	kr0000	1.0000



Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Delay (Time delay)		1.57 [μs]

Function The delay time set is added to the input numerical value and the result is output.

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) Delay time: T (seconds)

(3) Sampling time: ΔT (seconds) The number of samples (T/ΔT) is valid when it is 1000 or less.

The delay is canceled when the reset switch is turned on.

Note: Only operations with real numbers are valid.

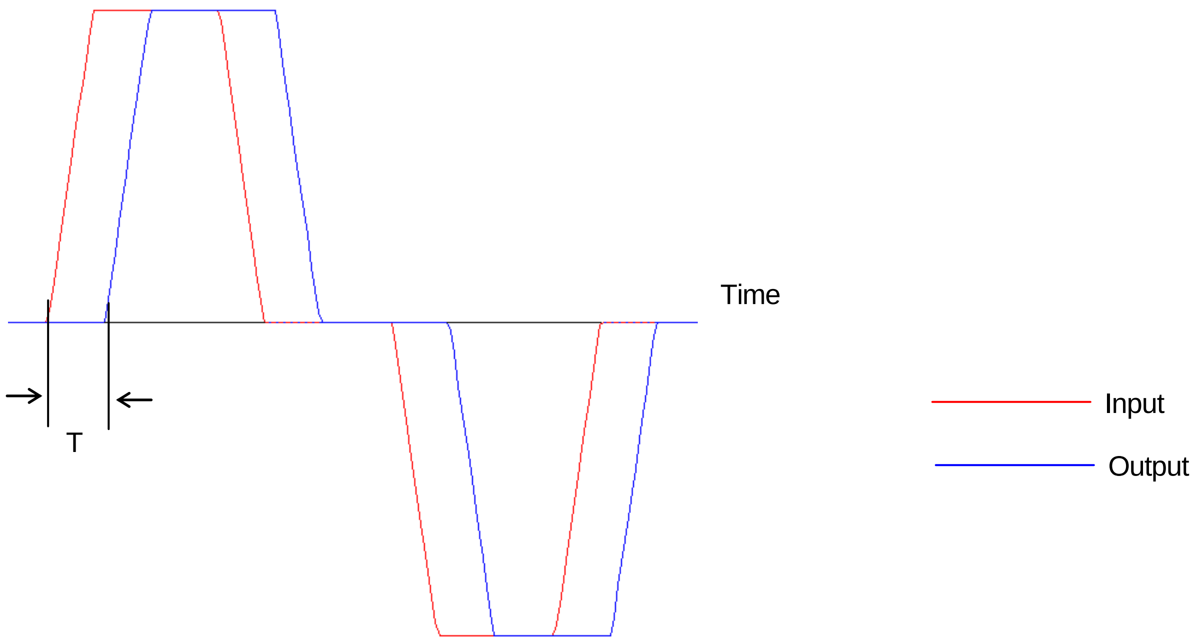
Graph

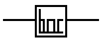
When the function argument is set as shown at right, the resulting trend graph is as shown below.

The input waveform is delayed by T (seconds) according to the delay time and is then output.

Delay

Reset	G00000	
Delay time	kr0000	5.0000
Sampling time	kr0001	1.0000



Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Constant frequency pulse		1.39 [μs]

Function: The input numerical value is turned on/off at the intervals set and is then output.

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) On time (seconds): Specifies the time that output is on.

(3) Off time (seconds): Specifies the time that output is off.

Note: Only operations with real numbers are valid.

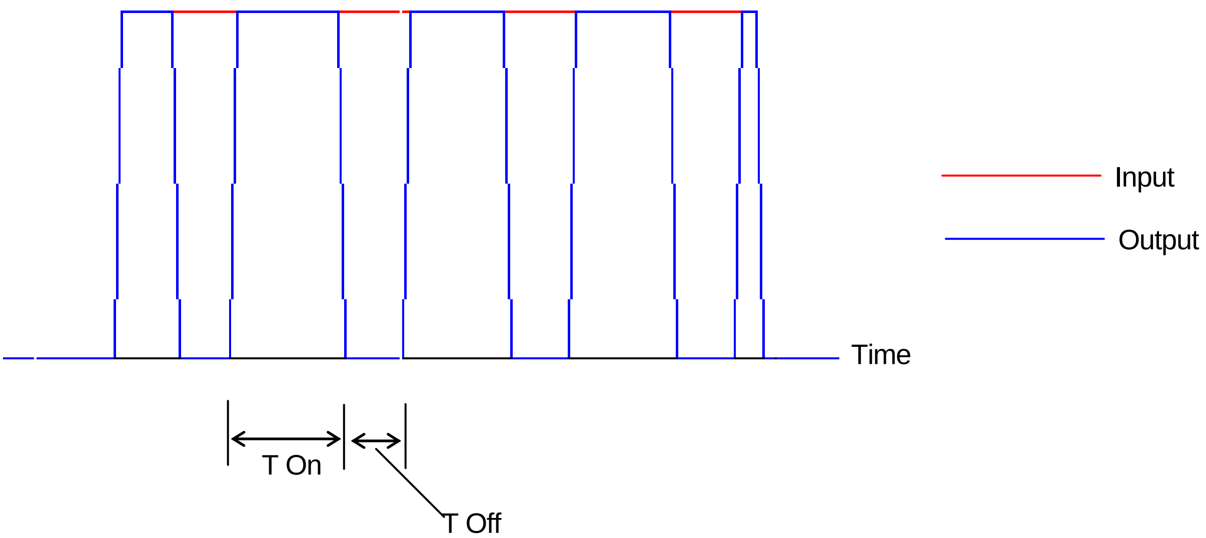
Graph

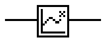
When the function argument is set as shown at right, the resulting trend graph is as shown below.

The input waveform is output according to the on/off time.

Constant frequency pulse

Reset	G00000	
On time	kr0000	5.0000
Off time	kr0001	3.0000



Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Variable setting pattern		2.00 [μs]
Function	Approximation conversion is performed using the input numerical value by line segmentation with pattern memory and the result is output.		

Settings of the function argument

(1) Number of points (≥2: integer): Number of input patterns

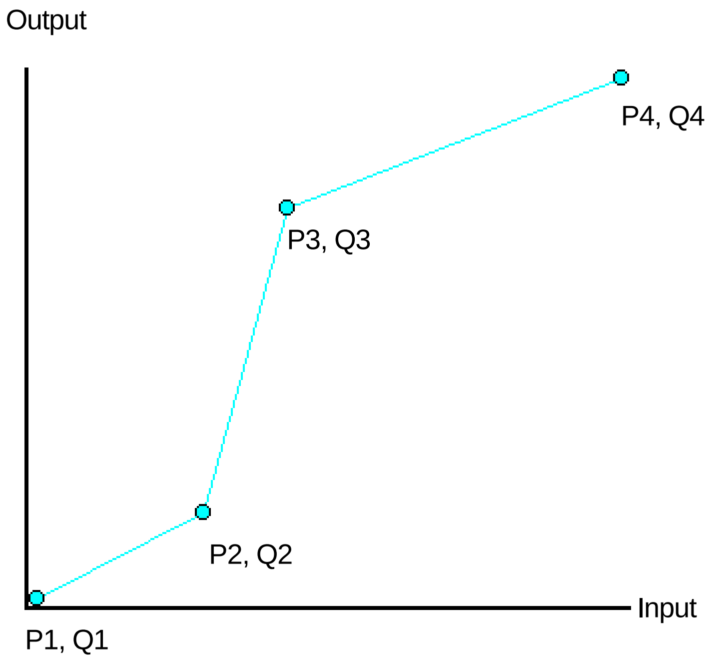
(2) Start of the pattern buffer (mrXXXX): Start address of the input buffer

In the pattern, an initial value was set beforehand using the pattern data, but here the real number value in a circuit can be changed.

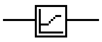
By accumulating the data obtained through process control, it can be applied to learning control.

Note: Only operations with real numbers are valid.

Graph



P1/Q1	mr0000	mr0001
P2/Q2	mr0002	mr0003
P3/Q3	mr0004	mr0005
P4/Q4	mr0006	mr0007

Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Upper and lower limiters		0.72 [μs]
Function	Upper and lower limiters are added to the input numerical value and it is then output.		

Settings of the function argument

(1) Upper limit value: Designate the upper limit value to be output

(2) Lower limit value: Designate the lower limit value to be output.

Note: Only operations with real numbers are valid.

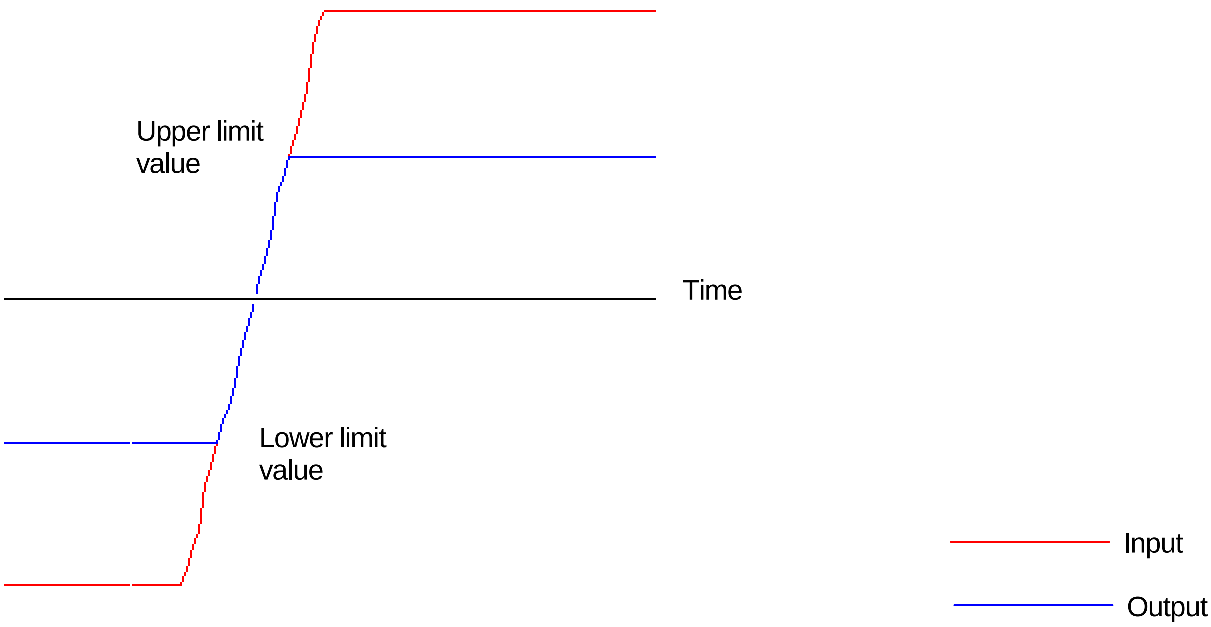
Graph

When the function argument is set as shown at right, the resulting trend graph is as shown below.

The input waveform is output according to the upper and lower limit values.

Upper and lower limiters

Upper limit value	kr0000	10.000
Lower limit value	kr0001	-10.000



Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Hysteresis		1.27 [μs]

Function

Hysteresis (2-gain amplifier during rising and falling) is added to the input numerical value and it is then output.

Settings of the function argument:

(1) Reset: Output value = Input value × G1

(2) Low side gain: G1 (0.0 < G1 < G2)

(3) High side gain: G2 (0.0 < G1 < G2)

When the input data is rising, G1 is valid, and when it is falling G2 is valid.

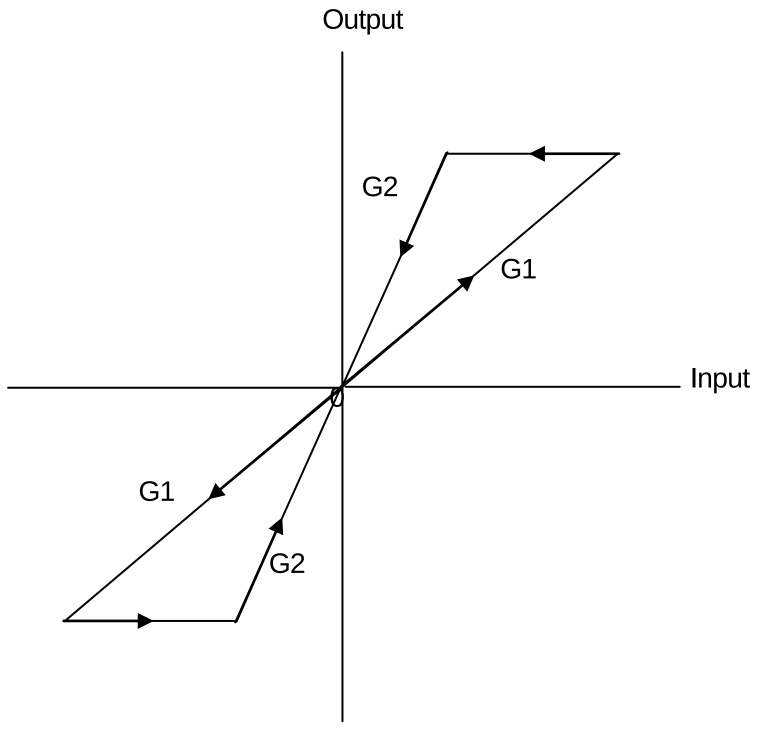
The output remains at a certain value when switching from rising to falling, or from falling to rising.

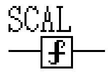
The reset switch must be set to on when starting the operation.

Note: Only operations with real numbers are valid.

Graph

The output data is plotted as the curve shown in the figure below according to the history of changes in the input data.



Kind	Name	Symbol	Execution time
Data flow language (Function 3)	Scaling		0.92 [μs]

Function

Scaling (sum of product operation) is added to the input numerical value and it is then output.

Settings of the function argument:

(1) Gain: Multiplication coefficient of the sum of product operation

(2) Offset: Addition coefficient of the sum of product operation

Output = Input * Gain + Offset

Note: Only operations with real numbers are valid.

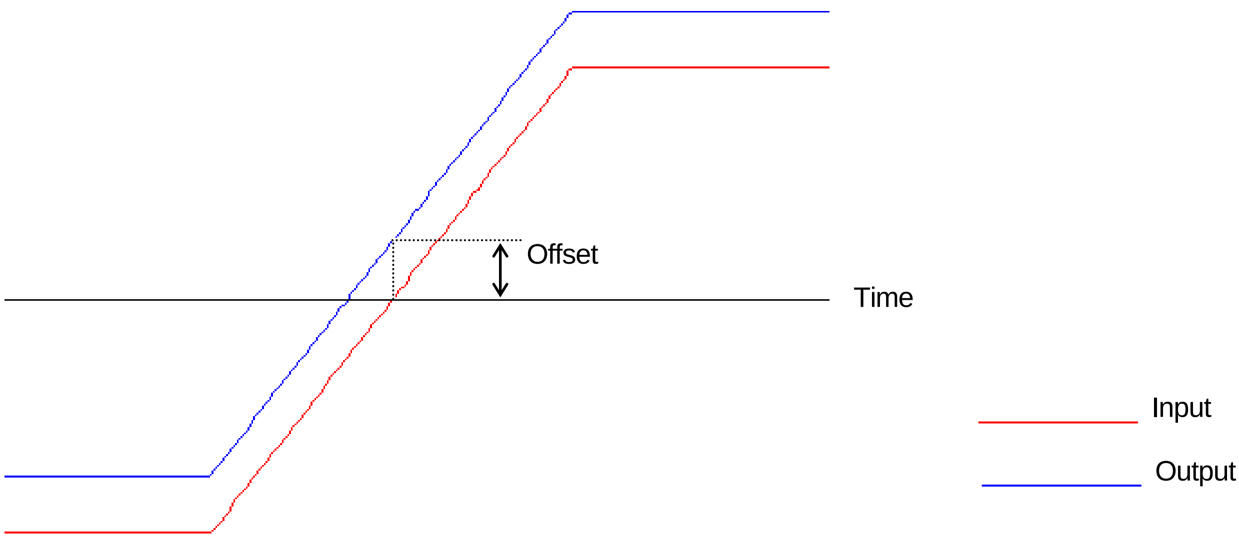
Graph

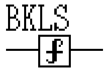
When the function argument is set as shown at right, the resulting trend graph is as shown below.

The input waveform is output according to the gain offset.

Scaling

Gain	kr0000	1.0000
Offset	kr0001	5.0000



Kind	Name	Symbol	Execution time
Data flow language (Function 3)	Backlash		0.81 [μs]

Function	Backlash (a kind of integral compensation) is added to the input numerical value and it is then output.

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) Width of backlash: W

The reset switch must be set to on when starting the operation.

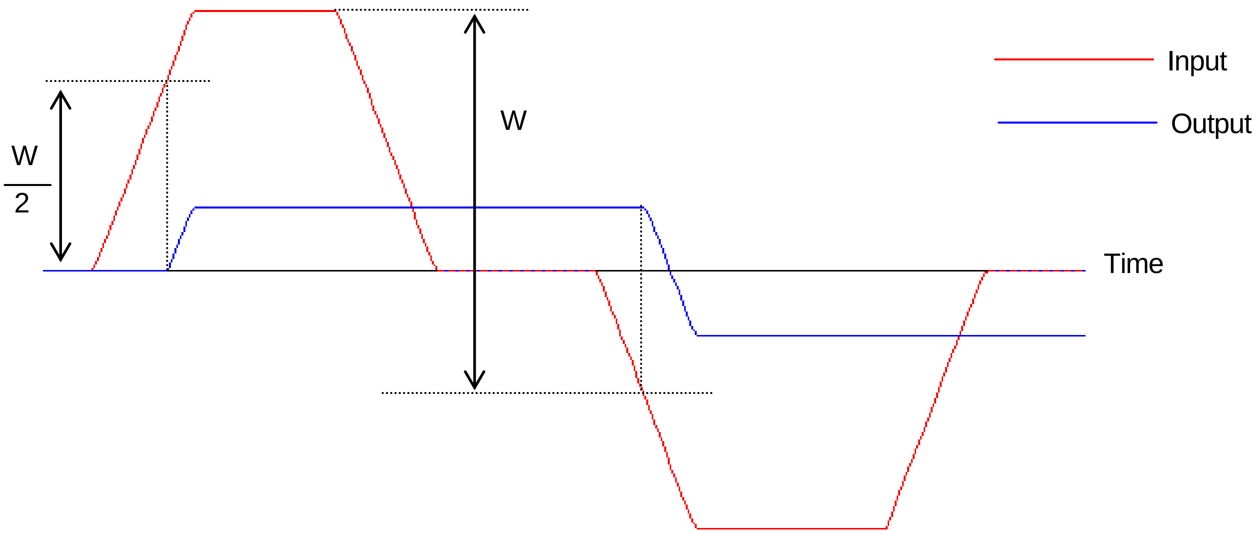
Note: Only operations with real numbers are valid.

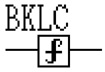
Graph

When the function argument is set as shown at right, the resulting trend graph is as shown below.

Backlash

Reset	G00001	
Width of backlash	kr0000	20.000



Kind	Name	Symbol	Execution time
Data flow language (Function 3)	Backlash compensation		0.95 [μs]

Function	Backlash compensation (a kind of differential compensation) is performed using the input numerical value and it is then output.

Settings of the function argument

(1) Reset: Input and output short-circuit reset command.

(2) Width of backlash: W

The reset switch must be set to on when starting the operation.

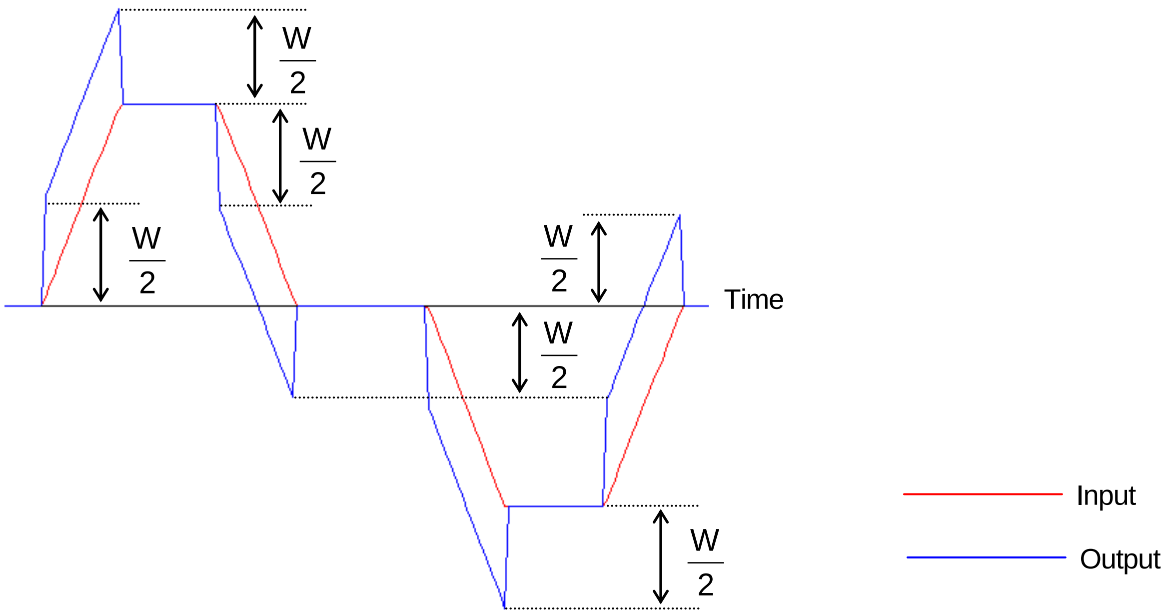
Note: Only operations with real numbers are valid.

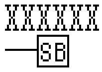
Graph

When the function argument is set as shown at right, the resulting trend graph is as shown below.

Backlash compensation

Reset	G00001	
Width of backlash	kr0000	20.000

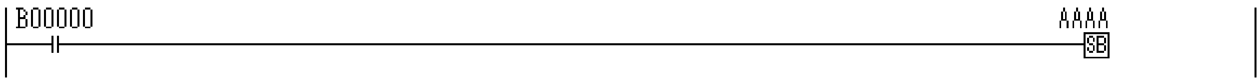


Kind	Name	Symbol	Execution time
Data flow language (Function 2)	Conditional subroutine		――――――――
Function	A subroutine is executed according to the logical condition of the input.		

The subroutine is executed when the input is on and not executed when off.

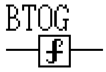
The other content is the same as that of the unconditional subroutine.

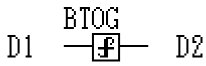
Example of use

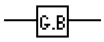


When relay B00000 is on, subroutine AAAA is executed.

When relay B00000 is off, subroutine AAAA is not executed.

Kind	Name	Symbol	Execution time
Data flow language (Function 3)	Binary Gray code		————
Function	The input numerical value is read as integer data, converted to a Gray code and is then output.		



Note: This performs the reverse operation of the Gray code conversion.  Be careful not to confuse them.

Example of use

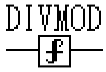
The data in register mi0000 is read as a 16-bit integer, converted to a Gray code and is then output.

If the data in register mi0000 is (10), then (15) is stored in register mi0001.

D1	D2	D1	D2	D1	D2	D1	D2
Integer	Gray	Integer	Gray	Integer	Gray	Integer	Gray
0000	0000	0100	0110	1000	1100	1100	1010
0001	0001	0101	0111	1001	1101	1101	1011
0010	0011	0110	0101	1010	1111	1110	1001
0011	0010	0111	0100	1011	1110	1111	1000

10 → 1010 → 1111 → 15

Input / Gray code / Integer / Output

Kind	Name	Symbol	Execution time
Data flow language (Function 3)	Division and remainder		————————

Function	The division and remainder of the input numerical value is output.

Settings of the function argument

(1) Divisor (integer): Number that divides the input numerical value

(2) Remainder (integer): Register that stores the remainder

Example of use

If the argument of DIVMOD is set as shown on the right, the remainder when the data in register mi0000 is divided by the divisor ki0000 (7) is stored in register mi0002. Also, the quotient is stored in register mi0001.

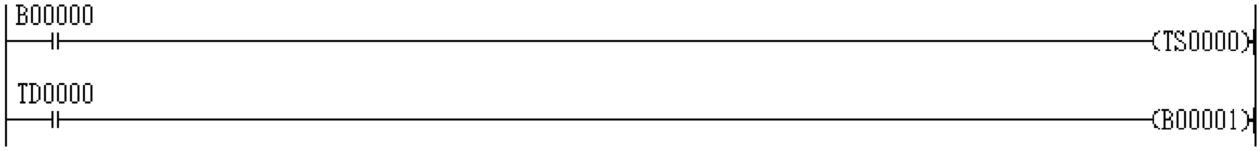
DIVMOD

Parameter	Label	Value
Divisor (integer)	ki0000	7
Remainder (integer)	mi0002	

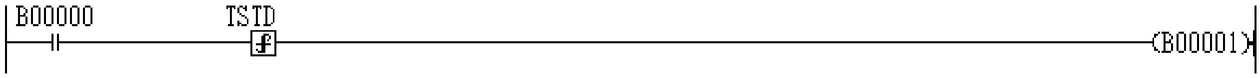
If the data in register mi0000 is (10), then (1) is stored in register mi0001 as the quotient, and (3) is stored in register mi0002 as the remainder.

Kind	Name	Symbol	Execution time
Data flow language (Function 3)	On timer (TSTD)	TSTD	――――
	Off timer (TRTC)	TRTC	
Function	Combines the on timer relay (TS, TD) and the off timer relay (TR, TC) in one line, and performs the same operation.		

TSTD: If the input bit is turned on, the coil is turned on after the time set by the argument elapses.



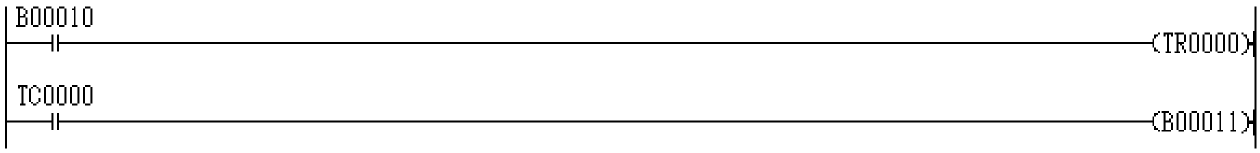
↓ Allows the content of two lines to be condensed into one.



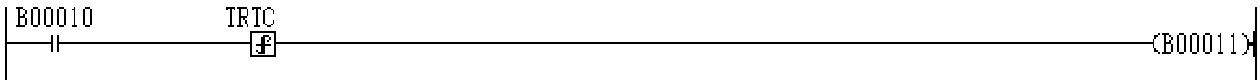
Settings of the function argument

(1) Timer value (real number): Sets the time for turning the coil on after the designated time elapses.

TRTC: If the input bit is turned off, the coil is turned off after the time set by the argument elapses.



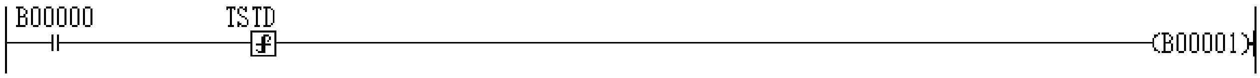
↓ Allows the content of two lines to be condensed into one.



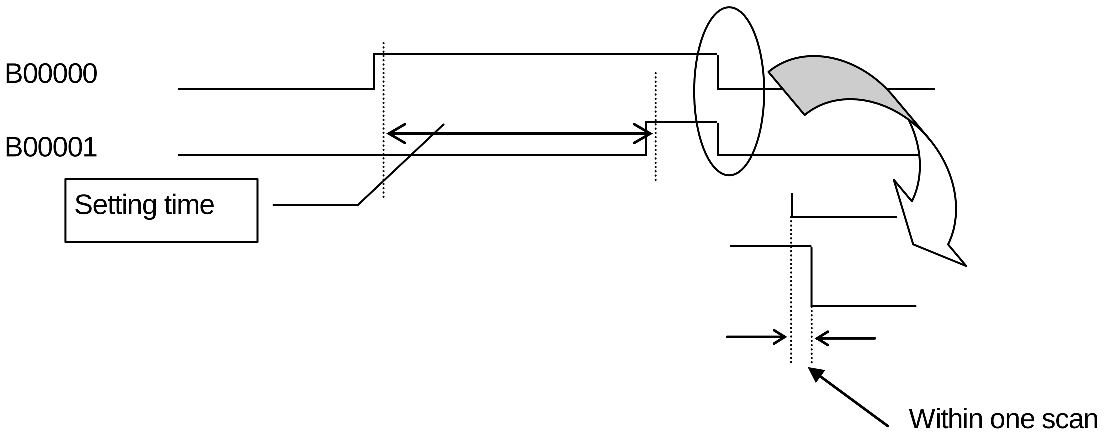
Settings of the function argument

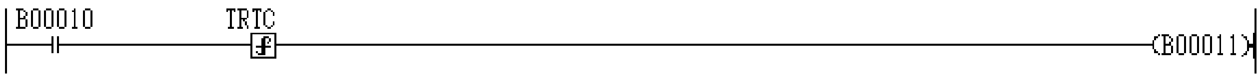
(1) Timer value (real number): Sets the time for turning the coil off after the designated time elapses.

Example of use

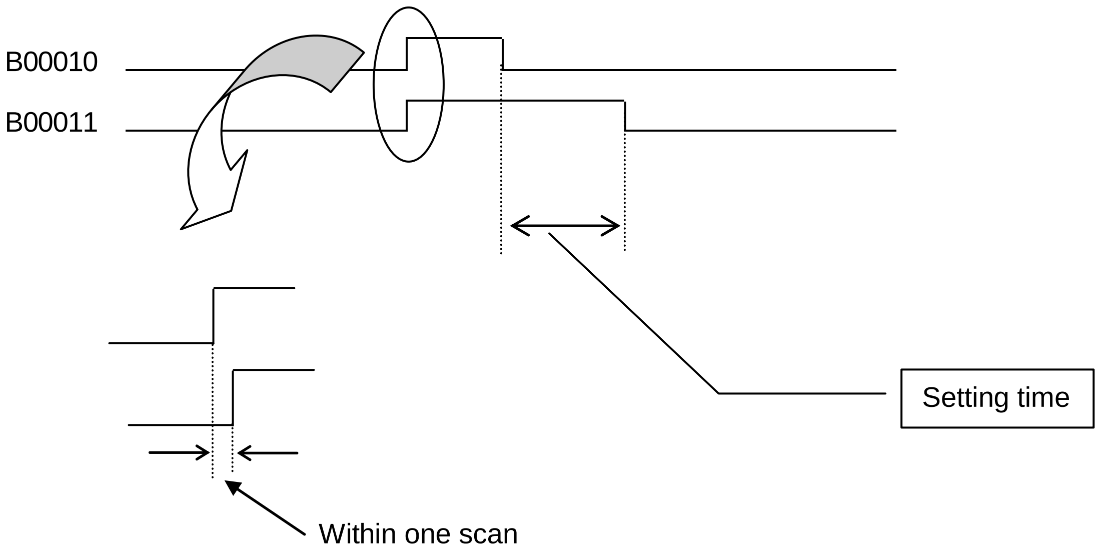


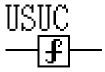
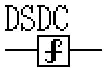
After relay B00000 is turned on, relay B00001 is turned on after the time set by TSTD elapses.



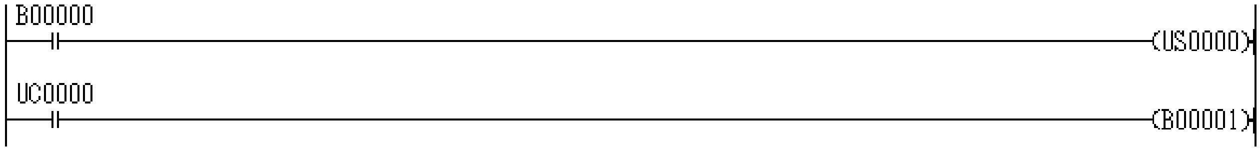


After relay B00010 is turned off, relay B00011 is turned off after the time set by TRTC elapses.

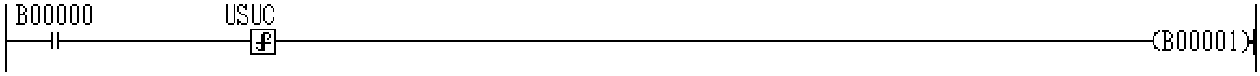


Kind	Name	Symbol	Execution time
Data flow language (Function 3)	On differential (USUC)		――――
	Off differential (DSDC)		
Function	Combines the on differential relay (US, UC) and the off differential relay (DS, DC) in one line, and performs the same operation but without a one scan delay.		

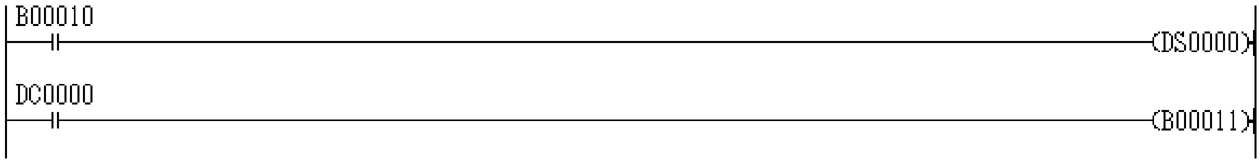
USUC: If the input bit is turned on, one scan is turned on without a one scan delay.



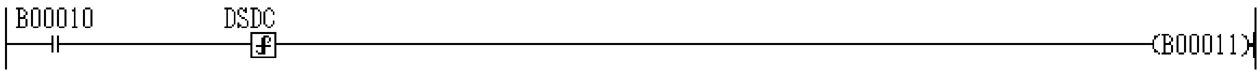
↓ Allows the content of two lines to be condensed into one.



DSDC: If the input bit is turned off, one scan is turned on without a one scan delay.

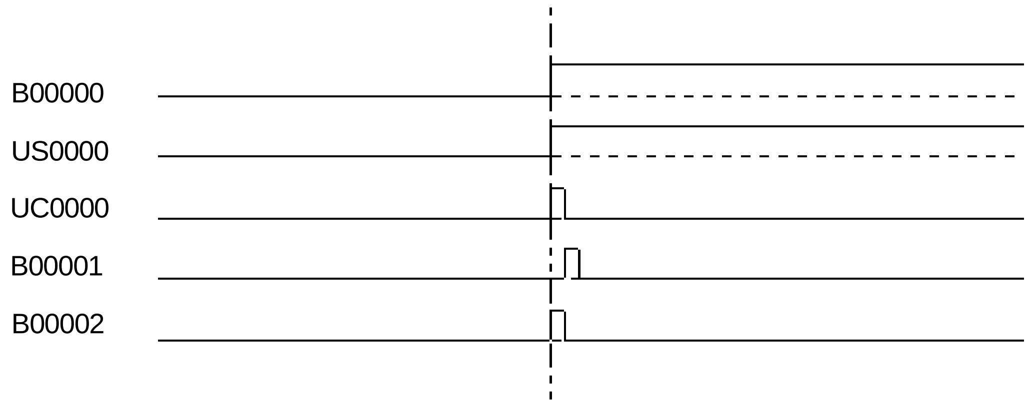


↓ Allows the content of two lines to be condensed into one.

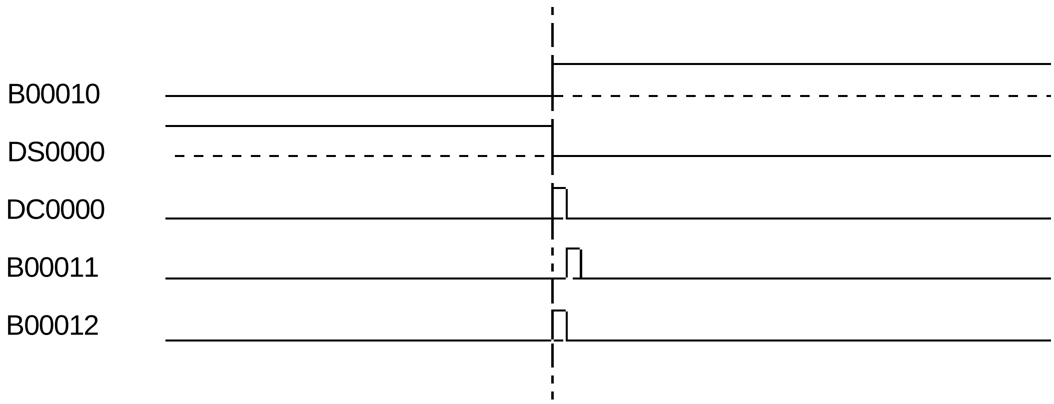


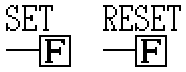
Example of use

When B00000 is turned on, after a delay for one scan, B00001 is turned off for one scan, but B00002 is turned on for one scan immediately after B00000 is turned on without a one scan delay.



When B00010 is turned on, after a delay for one scan, B00011 is turned on for one scan, but B00012 is turned on for one scan immediately after B00010 is turned on without a one scan delay. When B00010 is turned on, B00012 is turned on for one scan without a one scan delay.



Kind	Name	Symbol	Execution time
Data flow language (Function 4)	Set Reset		————————
Function	Set: When the input bit is turned on, the designated output bit remains on. Reset: When the input bit is turned off, the designated output bit remains off.		

Set:
Note: When set is on, the contact set by the argument is turned off when reset is turned on.

Settings of the function argument

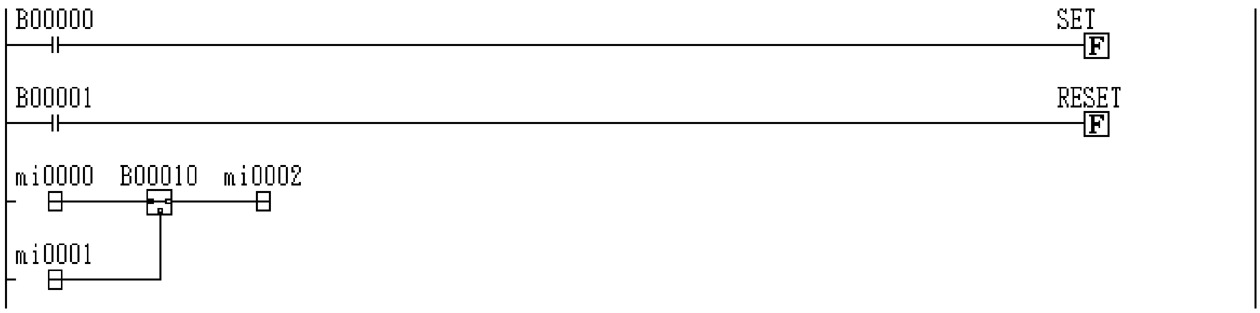
(1) Set coil: Designates the relay to remain on.

Reset:
Note: When reset is on, the contact set by the argument is not turned on, even when set is turned on.

Settings of the function argument

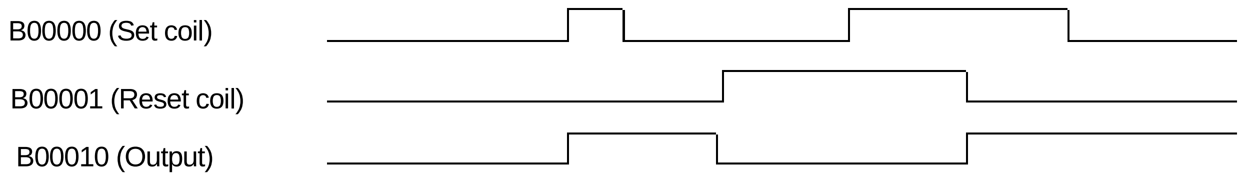
(1) Reset coil: Designates the relay to remain off.

Example of use



If B00000 = on, then B00010 = on, and the value in mi0001 is stored in mi0002.

If B0001 = on, then B00010 = off, and the value in mi0000 is stored in mi0002.



If B00000 = on, then B00010 = on. (Even when B00000 = off, B00010 is not turned off.)

If B00001 = on, then B00010 = off. (Even when B00000 = on, B00010 is not turned on.)

If B00001 = off, then B00000 = on, therefore B00010 = on.

Kind	Name	Symbol	Execution time
Data flow language (Function 4)	Counter (UPDOWN)		————————
Function	Combines the counters (NR, NP, NU, ND, NZ, n0) in one line, and performs the same operation.		

Settings of the function argument

(1) Reset coil: Sets the relay so that the current count value is 0.

(2) Preset coil: Sets the relay that makes the current count the value set by the count preset value.

(3) Upcoil: Sets the current count value to be incremental.

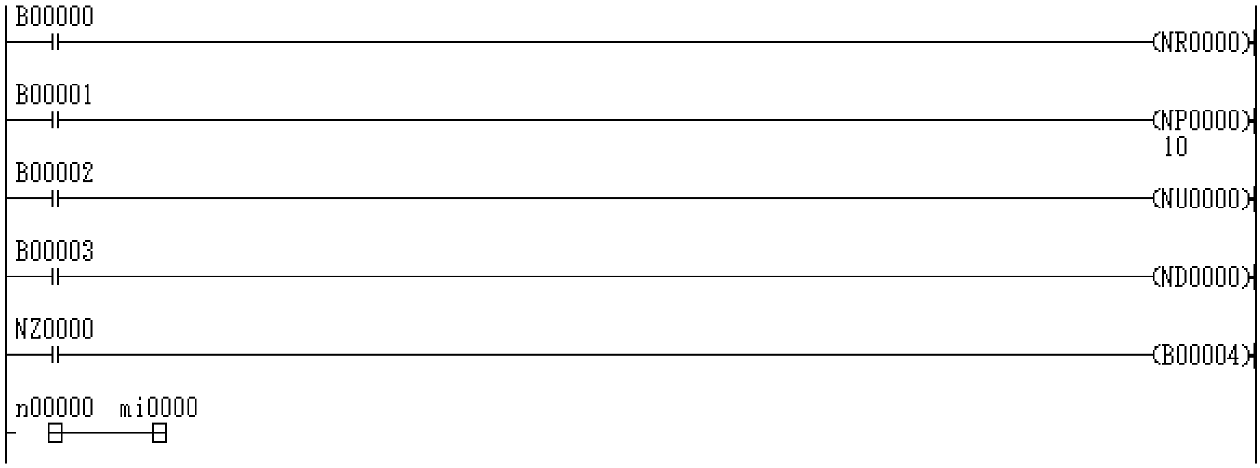
(4) Downcoil: Sets the current count value to be decremental.

(5) Zero detection contact: Sets the relay that communicates that the current count value is zero.

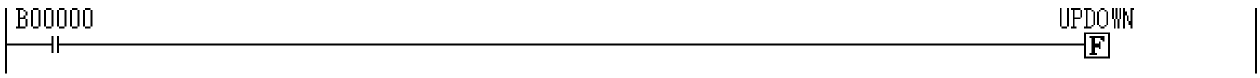
(6) Present value of count: Sets the register to store the current value.

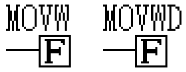
(7) Count preset value: Sets the value to be set to the current count value when the preset coil is turned on.

Example of use



↓ Allows the content of five lines to be condensed into one.



Kind	Name	Symbol	Execution time
Data flow language (Function 4)	Data transfer (MOVW/MOVWD)		————————
Function	Transfers the designated data to the designated label in units of words.		

Settings of the function argument

(1) Label of transferor: Designates the start address from which the data is transmitted.

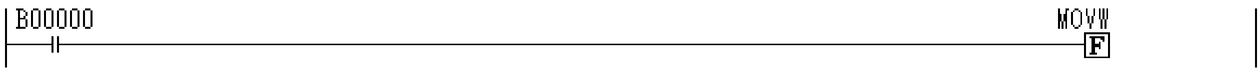
(2) Label of transferee: Designates the start address where the data is received.

(3) Offset of transferor: Designates the number of the label of the transferor from which the data is transmitted (for MOVW only).

(4) Offset of transferee: Designates the number of label of the transferee where the data is received (for MOVW only).

(5) Number to be transferred: Designates the number of data to be transferred.

Example of use



With the setting as shown at right, five-word data is transferred from mi000A to b00004.

mi000A → b00004

mi000B → b00005

mi000C → b00006

mi000D → b00007

mi000E → b00008

MOVW

Parameter	Label	Value
Transferor label	mi0000	
Transferee label	b00000	
Transferor offset	ki0000	10
Offset of transferee	ki0001	4
Number to be transferred	ki0002	5

Kind	Name	Symbol	Execution time
Data flow language (Function 3)	Integer conversion	TODINT	―――
	Real number conversion	TOREAL	
Function	Converts the designated data to the designated type and outputs the result.		

TODINT (The real number input is converted to a 32-bit integer.)

Settings of the function argument

(1) Transferor (2 points used: even address): Designates the address where the input real number data is converted to a 32-bit integer and is output.

(2) Transferee (2 points used: even address +1): Designates the address where the sign is output when the input real number data is converted to a 32-bit integer.

TOREAL (The 32-bit integer input is converted to a real number.)

Settings of the function argument

(1) Transferor (2 points used: even address): Designates the address where the input 32-bit integer data is converted to a real number and is output.

(2) Transferee (2 points used: even address +1): Designates the address where the sign is output when the input 32-bit integer data is converted to a real number.

Example of use

If TODINT is set as shown at right and the data in the input real number register mr0000 is (-12.5600), then:
mi0010 = -13 and mi0011 = -1

TODINT

Parameter	Label	Value
Transferee (even address)	mi0010	
Transferee (even address +1)	mi0011	

When TOREAL is set as shown at right, then:
mr0011 = 131082

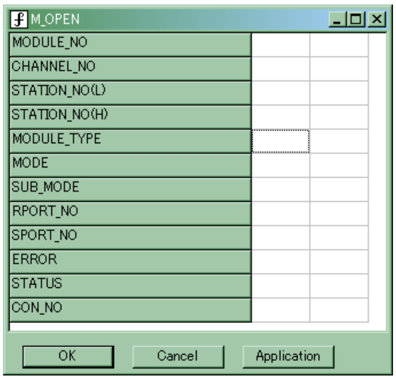
mr0011 = ki0000 + ki0001 * 65536
= 10 + 2 * 65536
= 10 + 131072
= 131082

TOREAL

Parameter	Label	Value
Transferor (even address)	ki0000	10
Transferor (even address +1)	ki0001	2

Kind	Name	Symbol	Execution time
Data flow language (Function 4)	Channel open	M_OPEN	————
Function	A function for setting the destination of message communications. This setting is used in M_SEND (transmitting messages) and M_RECV (receiving messages), which are explained later in this document.		

Settings of the function argument



(1) Communication station number (slot number): Designates the slot number (0 to 9) of the Ethernet module (CPU module) used for communication. Set to 0 for communication with the local CPU module.

(2) Channel number: Designates the channel number in the communications module. (Connection number: 1 to 9)

(3) Station number (L): The IP address of the communication target (lower 16 bits)

(4) Station number (H): The IP address of the communication target (upper 16 bits)

(5) Module type number: 0 (Not used)

(6) Communications mode: Sets the communication conditions of the connection.

(7) Communications submode: (Not used in the μGPCsH)

(8) Communication target port number: Sets the port number of the communication target.

(9) Local port number: Sets the local port number.

(10) Error flag: Turned on for one scan if open processing finishes abnormally.

(11) Status: Displays details of the error.

(12) Connection number: After open request, H: slot number, L: channel number is entered.

Operation of the instructions

(1) As a result of the input relay (B00000) starting up, open processing of the module designated by the station number (slot number) is started. (Open processing is not completed within one scan.)

(2) If open processing is completed normally, the normal flag is turned on and the connection number is output to the connection number. In this state, M_SEND and M_RECV can now be used.

(3) If open processing is not completed normally, the error flag is turned on for one scan, and the error code is output to status.

(4) When the input relay is turned off, close processing is performed. (Close processing is also not completed within one scan.)

(5) When close processing is completed, the normal flag is turned off. (Close processing is not completed abnormally.)

Precautions

(1) There are two open methods for receiving, the passive method and active method. For communicating, open processing is used for receiving and open processing is used for sending.

(2) In order to send data, the communication target must be ready for receiving, therefore open processing for the passive method must be completed first.

(3) If the input relay is turned on (off in the open state, close processing is performed.

(4) When reopen is performed after close processing has finished, the communication target must be closed first, before performing the reopen process.

Function details

(1) Station number (L), (H)

Sets the IP address of the communication target. The IP address is set as a hexadecimal or decimal number. The station number (L) is set in the lower 16 bits while the station number (H) is set in the upper 16 bits.

Example: Set as follows when the IP address is 172.16.0.1.

ACh	10h	00h	01h
172	16	0	1

Station number (L) = 0001(h) or 1

Station number (H) = AC10(h) or -21488

(2) Communication mode

The communication conditions of the connection to be set to channel open are set respectively with one word data representing bit information. The one word content is as follows.

0082: UDP/IP
0002: TCP/IP active open
C002: TCP/IP passive open
8002: TCP/IP passive open

Bit details

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*													1	0

Open method (bits 15, 14)
00: Active
11: Fullpassive
10: Unpassive node

Communication protocol (bit 7)
0: TCP/IP
1: UDP/IP

Bit 1
1: Fixed

(a) Communication protocol
Sets whether to use TCP/IP or UDP/IP as the communication protocol for each connection.

(b) Open method
When TCP/IP is used for open, after open processing of the node that performs the Fullpassive/Unpassive open (passive open) is completed, the node that performs Active open is opened.

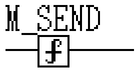
(i) Active open method
Performs active open processing of other nodes that are open for passive TCP connection.

(ii) Fullpassive open method
Opens passive TCP connections only for the specific nodes set in the communication address setting area. The nodes stand by for active open requests from other nodes set in the communication address setting area.

(iii) Unpassive open method
Opens passive TCP connections for all other nodes connected to the network. The nodes stand by for active open requests from all other nodes in the network.

(3) Error status

Name	Code	Content
Abnormal parameters	177 (B1h)	When there is no Ethernet module in the slot specified with the communication station number (slot number)
Abnormal channel open	193 (C1H)	When an inconsistent value is set in the communication mode
Abnormal port designation	200 (C8h)	When an inconsistent value is set for the IP address, local port number or communication target port number

Kind	Name	Symbol	Execution time
Data flow language (Function 4)	Message transmittal		—
Function	Performs message transmittal with the communication target set with M_OPEN.		

Settings of the function argument

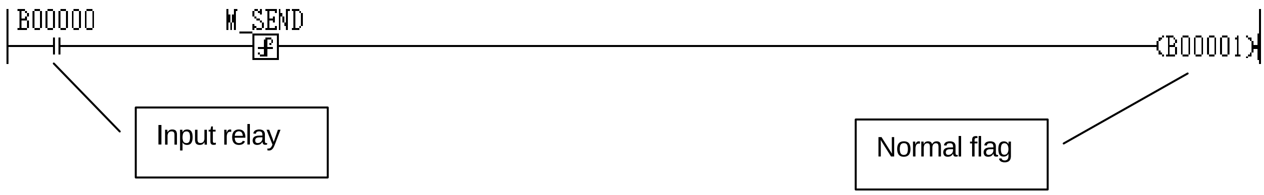
(1) Connection number: Sets the connection number opened with M_OPEN.

(2) Transmittal data storage variable: Sets the size of the data where the transmission data is stored.

(3) Transmittal data storage variable size: Sets the size of the data where the transmission data is stored. (In word units)

(4) Error flag: Turned on for one scan if message transmission is not performed normally.

(5) Status: Outputs the status if message transmission is not performed normally.



Operation of the instructions

(1) When the input relay starts up (off → on), a message is sent to stations with connection numbers set in connection number. (The transmission process is not completed within one scan.)

(2) If message transmission is performed normally, the normal flag is turned on for one scan.

(3) If message transmission is not completed normally, the error flag is turned on for one scan, and the error code is output to status.

Precautions

(1) The amount of data that can be transmitted in a single message is 512 words.

(2) The input relay is disabled while messages are being transmitted (from the startup of the input relay to the startup of the normal flag or error flag.)

(3) Do not change the transmittal data storage variable while messages are being transmitted. If it is changed, the sent data is not guaranteed.

(4) When the number of data designated by the transmittal data storage variable size exceeds the variable size designated by the transmittal data storage variable, the data in excess of the latter may be indefinite. You must input the designated variable size as the transmittal data storage variable size.

(5) The program should be created so that the ON flag is input to the input relay after the normal flag of M_OPEN has been turned on.

Precautions when using M_SEND

(1) In the versatile communications mode of UDP/IP, no delivery confirmation or flow control is performed. When the receive procedure cannot keep pace, the receive buffer becomes full and the subsequent data is lost. Therefore, the amount of sent data at the transmitting side does not match the amount of received on the receiving side. Also, when the receive buffer is full, about 10 seconds are required for releasing the buffer, and hence receiving may stop during this time.

(2) In Full Passive open, if an open request is received from a target where the IP address and port number do not match, once a connection is established, the Full Passive side sends a close request to the Active side. Consequently, at the Active side, when opening is completed normally and the data has been sent, Error Status C7h (compulsory close) occurs.

(3) When the port number of the transmitting side does not match that of the receiving side, a transmittal error occurs, and the transmitting side performs compulsory close. The Error Status C7h: (compulsory close) also occurs.

Error status

Name	Code	Content
Abnormal parameters	177 (B1h)	When there is no Ethernet module in the slot specified with the communication station number (slot number)
Abnormal channel open	193 (C1H)	When an inconsistent value is set in the communication mode
Abnormal port designation	200 (C8h)	When an inconsistent value is set for the IP address, local port number or communication target port number

Kind	Name	Symbol	Execution time
Data flow language (Function 4)	Message receiving	M_RECV	————
Function	Performs message receiving with the communication target set with M_OPEN.		

Settings of the function argument

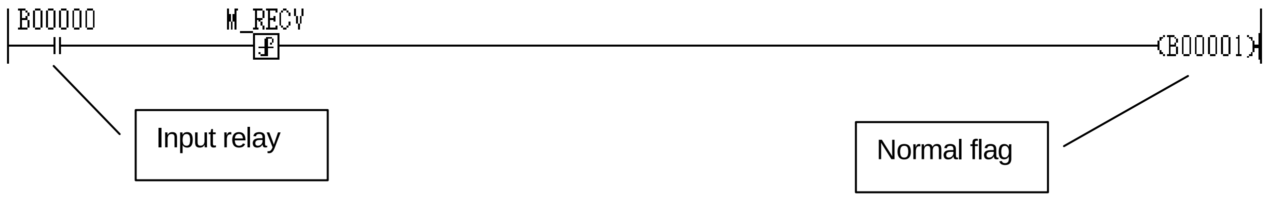
(1) Connection number: Sets the connection number opened with M_OPEN.

(2) Transmittal data storage variable: Sets the size of the data where the transmission data is stored.

(3) Transmittal data storage variable size: Sets the size of the data where the transmission data is stored. (In word units)

(4) Error flag: Turned on for one scan if message transmission is not performed normally.

(5) Status: Outputs the status if message transmission is not performed normally.



Operation of the instructions

(1) When the input relay starts up (off → on), a message is received from stations with connection numbers set in connection number. (The receiving process is not completed within one scan.)

(2) If message receiving is performed normally, the normal flag is turned on.

(3) If message receiving is not completed normally, the error flag is turned on for one scan, and the error code is output to status.

Precautions

(1) The amount of data that can be transmitted in a single message is 512 words.

(2) Keep the input relay on while messages are being received (from the startup of the input relay to the startup of the normal flag or error flag.) Turning the input relay off means that receiving is paused.

(3) After receiving is paused, starting up the input relay (off → on) restarts receiving. Even if the connection number, receiving data storage variable, and receiving data storage variable size have changed, receiving restarts with input values from before the pause. Changes are not reflected in the process of receiving messages.

(4) When the message receiving process is finished, if the input relay remains on in the next scan, a new message receiving process starts.

(5) Maintain the receiving data storage variable throughout the message receiving process. If it is overwritten, the received data is not guaranteed.

(6) When the number of data designated by the receiving data storage variable size exceeds the variable size designated by the receiving data storage variable, it may overwrite other variable areas. You must input the designated variable size as the receiving data storage variable size.

(7) The program should be created so that any input to the input relay is made after the normal flag of M_OPEN is turned on.

Precautions when using M_RECV

The precautions are the same as for M_SEND. Refer to “Precautions when using M_SEND.”

Error status

Name	Code	Content
Abnormal parameters	177 (B1h)	When there is no Ethernet module in the slot specified with the communication station number (slot number)
Abnormal channel open	193 (C1H)	When an inconsistent value is set in the communication mode
Abnormal port designation	200 (C8h)	When an inconsistent value is set for the IP address, local port number or communication target port number
Channel close	199 (C7H)	When the communication target is closed

Kind	Name	Symbol	Execution time
Data flow language (Function 4)	Matrix	MATRIX	――――――――

Function	A function for inputting a matrix.

Settings of the function argument

(1) Input register: Connects external equipment which switches output data using a strobe.

(2) Output register: Strobe output (connected to the strobe input of external equipment.)

(3) Name of the foremost matrix input register: Designates the start of the register name where the data input by the strobe output is stored sequentially.

Example of use

Input register: i00000 (Register name for data input of one word)

Output register: o00001 (Output register name for generating strobe pulses)

Name of the first matrix input register: mi0010

i00000 data input by the strobe output of o00001 (000010 to 00001F) is stored sequentially from mi0010 to mi001F.

i00000 = 1　O00010 = ON　mi0010 = 1

i00000 = 2　O00011 = ON　mi0011 = 2

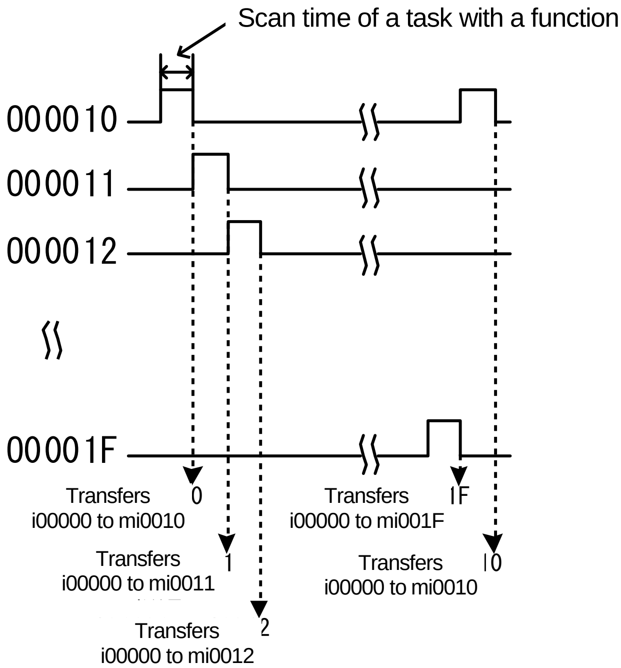
i00000 = 3　O00012 = ON　mi0012 = 3

↓

i00000 = 16　O0001F = ON　mi001F = 16

i00000 = 17　O00010 = ON　mi0010 = 17

i00000 = 18　O00011 = ON　mi0011 = 18



Kind	Name	Symbol	Execution time
Data flow language (Function 4)	FREAD	FREAD –[f]–	————

Function	Reads files saved in a CompactFlash card.

Settings of the function argument

(1) Attribute (CSV digit)
0: Reads the file as a binary file.
Other than 0: Reads the file as a CSV file and specifies number of digits of the first row of the CSV file.

(2) File name storage variable
Specifies the variable name where the file name is stored. The data is ASCII code.

(3) Read data storage variable
Specifies the variable name where the read data is stored.

(4) Read data storage variable size
Specifies the available area for the variable that stores the read data.

(5) Error flag
Turns on when an error occurs.

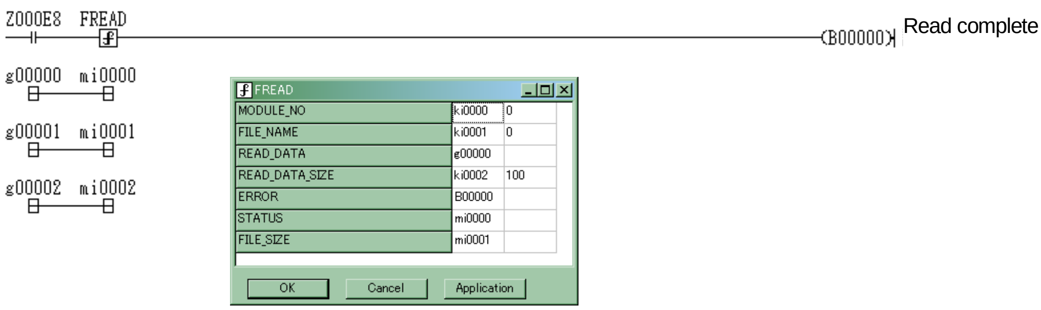
(6) Status
Stores an error code when an error occurs.

(7) Read file size
The size of the read file is stored (in word units).

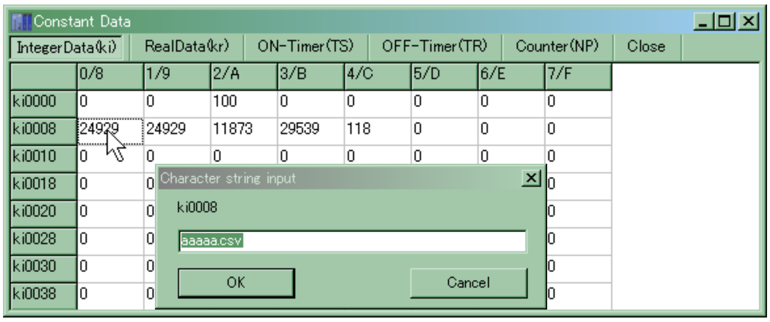
Example of use

By switching Z000E8 on, a value is stored in g00000~.

The actual process is performed in the background so when reading finishes, B00000 is turned on.



You can enter a file name in ASCII code by double-clicking the constant data input ki variable area.



Status list

(1) Abnormal file name (Code: 65)
The file name storage variable includes characters that are not allowed in a file name.

(2) File being processed (Code: 35)
Another program (another place) is currently executing the file function.

Kind	Name	Symbol	Execution time
Data flow language (Function 4)	FWRITE	FWRITE	――――――――

Function	Saves data in the PLC as files in a CompactFlash card.

Settings of the function argument

(1) Attribute (CSV digit)
0: Reads the file as a binary file.
Other than 0: Reads the file as a CSV file and specifies number of digits of the first row of the CSV file.

(2) File name storage variable
Specifies the variable name where the file name is stored. The data is ASCII code.

(3) Write data storage variable
Specifies the variable name where the written data is stored.

(4) Write data storage variable size
The size of the write file is stored (in word units).

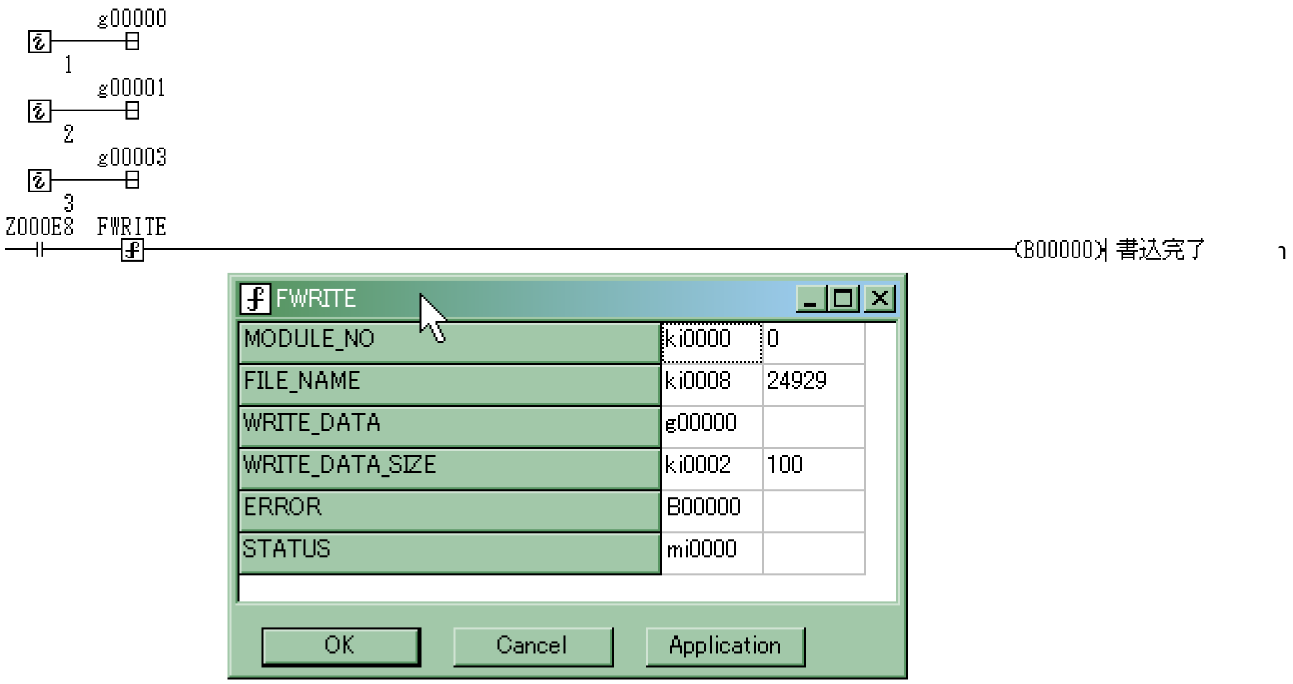
(5) Error flag
Turns on when an error occurs.

(6) Status
Stores an error code when an error occurs.

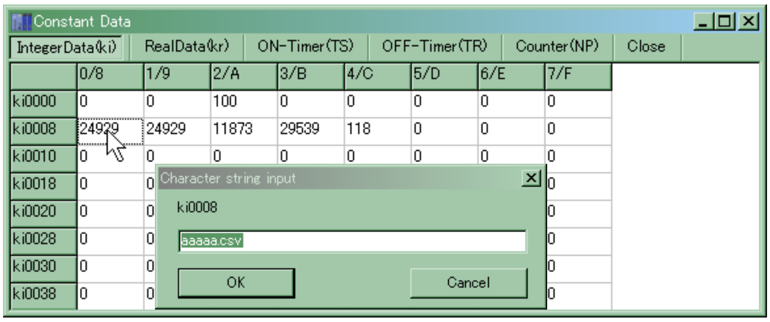
Example of use

By switching Z000E8 on, the g00000~ data generates the stored file.

The actual process is performed in the background so when writing finishes, B00000 is turned on.



You can enter a file name in ASCII code by double-clicking the constant data input ki variable area.

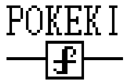


Status list

(1) Abnormal file name (Code: 65)
The file name storage variable includes characters that are not allowed in a file name.

(2) File being processed (Code: 35)
Another program (another place) is currently executing the file function.

(3) Abnormal file access (Code: 66)
An abnormality occurred during file access (Code: 66)

Kind	Name	Symbol	Execution time
Data flow language (Function 4)	POKEKI		――――――――
Function	Saves the ki (constant data) value as an application program. (When reset is performed, it becomes written data.)		

Settings of the function argument

(1) ki start offset (Integer)
Specifies the start of the ki area to write.

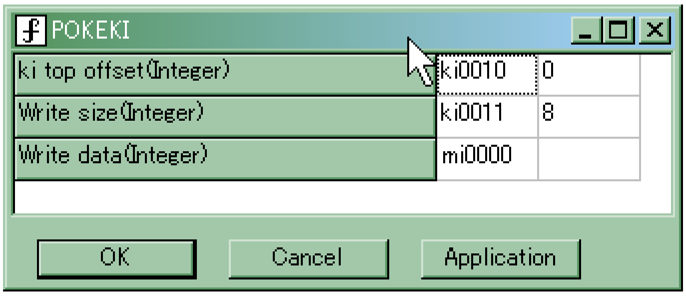
(2) Write size (Integer)
Specifies the size of ki to write file (in word units).

(3) Write data (Integer)
Specifies the variable where the written data for writing ki is stored.

Example of use

By switching Z000E8 on, ki0000~ changes to 1, 2, 3.

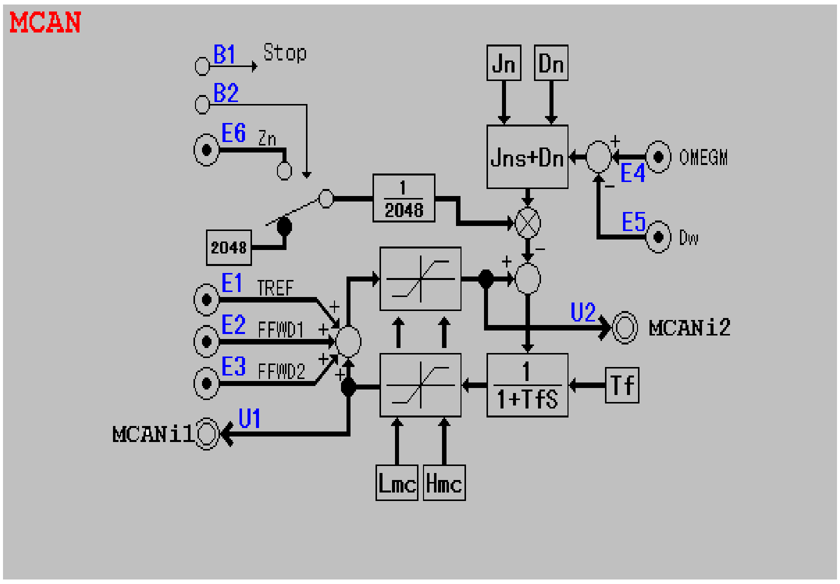
When writing finishes, B00000 is turned on.



Kind	Name	Symbol	Execution time
Data flow language (Function 4)	Versatile communications	C_FREE —[F]	――――――――
Function	A function for versatile communications.		

(1)Settings of the function argument	
0000	Transmittal request: Starts sending data. When sending finishes, this must turned off by the application.
0001	Transmittal data length: Designates the length of the data sent in bytes.
0002	Transmittal data address: Designates the start address of the data sent.
0003	Receiving data address: Designates the start address of the receiving data.
0004	Parameter address: Designates the start address of the parameters for port initialization.
0005(Not used) -0006(Not used)	
0007	Transmittal completed: Turned on when the data has been sent. (1 scan)
0008(Not used) -0009(Not used)	
000A	Receiving completed: Turned on when the data has been received. (1 scan)
000B: (Not used) 000C: (Not used)	
000D 000E	Receiving data length: Stores the received data length. RS-485 post number: Stores the circuit number of the versatile communications module.
(2)Port initialization parameter details	
0000	Post number of versatile communications module number (unit number, slot number) Example of unit 1, slot 2: 102h
0001	Port no (1:CH1 2:CH2 3:CH3)
0002(Not used) to 0002: (Not used)	
000D	Frame detection 0:No (Receiving completed if the data is received.) 1:Variable-length (Receiving completed when data enclosed in a start code and end code is detected.) 2:Fixed (Receiving completed when the received data reaches the number of bytes received.)
000E	Designates the number of receiving bytes when the number of receiving bytes is fixed. “0” is specified when the length is variable.
000F	Designates the number of bytes in the start code when the number of bytes in the start code is variable.
0010	Designates the start code when start code 1 is variable length.
0011:Start code 2, 0012:Start code 3, 0013:Start code 4, 0014:Start code 5	
0015	Designates the number of bytes in the end code when the number of bytes in the end code is variable.
0016	Designates the end code when end code 1 is variable length.
0017:End code 2,0018:End code 3,0019:End code 4,001A:End code 5	
001B: …(Not used) to 001F: (Not used)	

Kind	Name	Symbol	Execution time
Data flow language (Function4) SHPC-115-Z Only	Motor side cancellation (Modern control)	MCAN	————————
Function	I consider inertia coefficient Jn and dumping coefficient Dn from the torque order of the motor and real motor turn speed and control it to minimize a difference with the value that I found.		



Input signal

Variable	Type	Contents	Range/Unit	Remarks
B1	Relay	Stop switch		
B2	Relay	Power Con reshuffling switch		
E1	Real	Torque speed order Tref	%	
E2	Real	Feed forward Ffwd1	%	
E3	Real	Feed forward Ffwd2	%	
E4	Real	Motor angular velocity Omegam	%	
E5	Real	Speed deviation Dw for FCAN output	%	
E6	Real	Power Con coefficient Zn	0.0~16.0	
Jn	Real	Inertia set point	0.0~31.999	(Note 1)
Dn	Real	Dumping set point	0.0~0.999	(Note 2)
Tf	Real	Time constant filter	ms	Default=10.0ms(Note 3)

Output signal

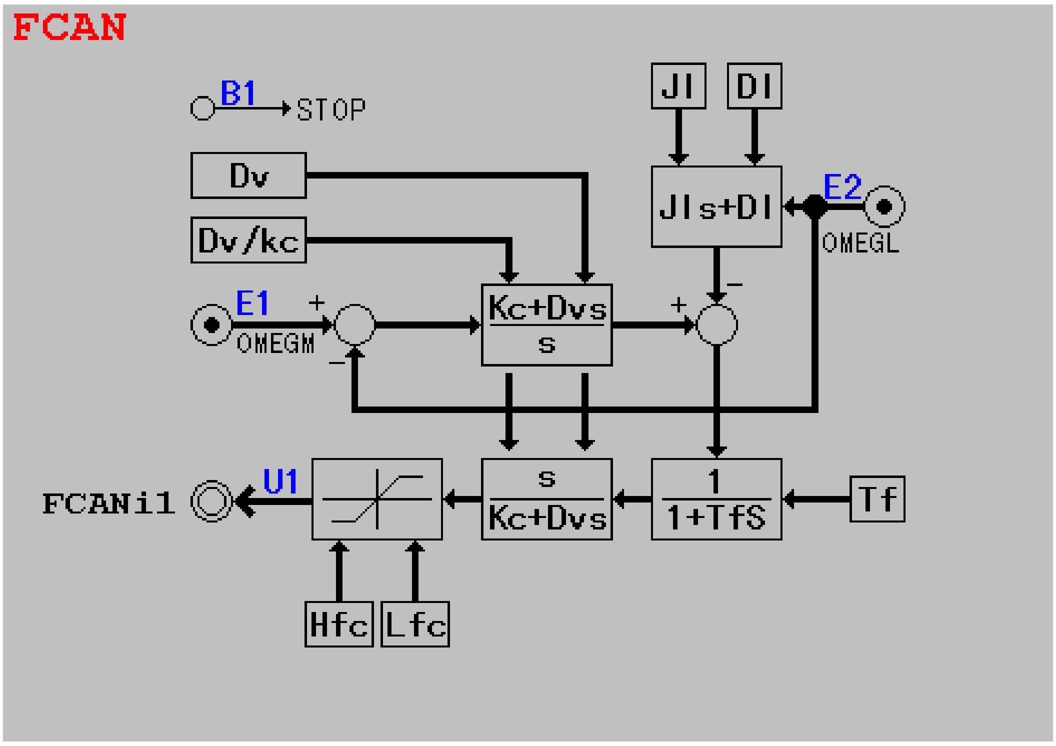
Variable	Type	Contents	Range/Unit	Remarks
U1	Real	MCAN Output1	%	
U2	Real	MCAN Output2	%	

(Note 1) A inertia set point (inertia [kgm2] X rating speed [rad/s] / rating torque of the Jn)= motor axis conversion)[Nm]

(Note 2) A dumping set point (dumping of the Dn)= motor axis conversion)[Nm / s/rad] X rating speed [rad/s] / rating torque[Nm]

(Note 3) When Tf set the value that is shorter than the double of the practice (operation) period, I do not become the right operation result. Please set it so that Tf becomes the value that is longer than double of sample time.

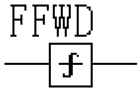
Kind	Name	Symbol	Execution time
Data flow language (Function4) SHPC-115-Z Only	flexible side cancellation (Modem control)	FCAN	————
Function	From the torque order of the motor, real motor turn speed and rotary speed of the load side, I twist Inertia coefficient Jln and dumping coefficient Dln of the load side and consider various set points of the axis and control it to minimize a difference with the value that demanded it.		

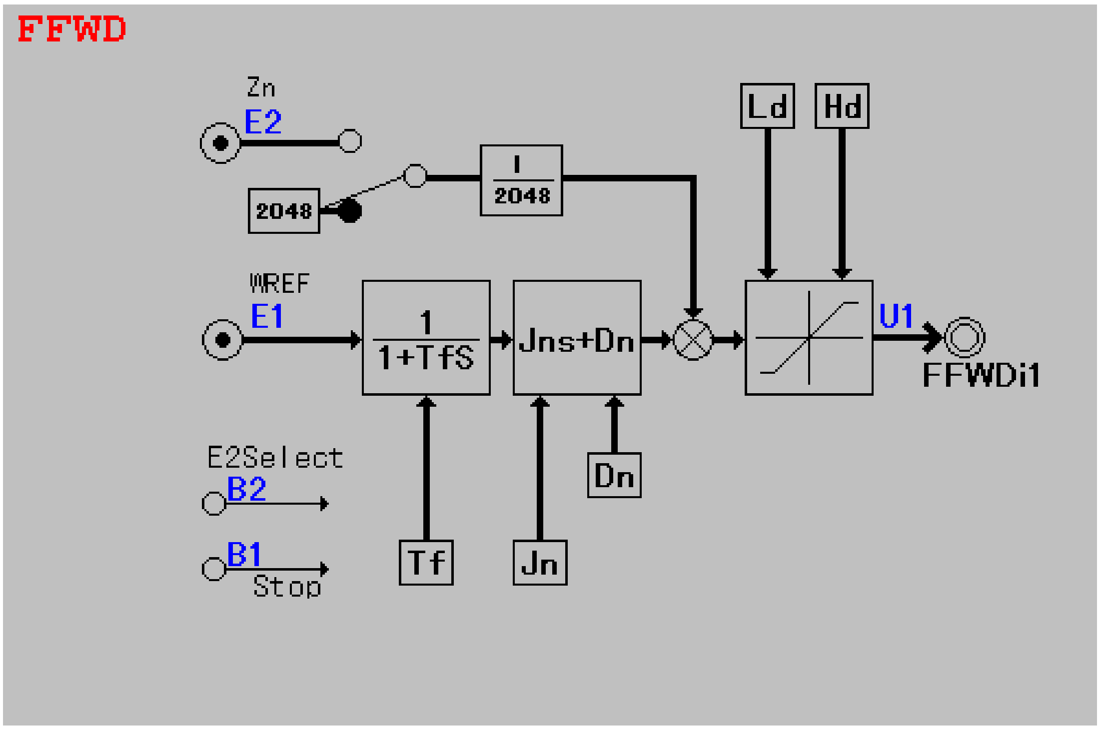


Input signal

Variable	Type	Contents	Range/Unit	Remarks
B1	Relay	A stop switch		
E1	Real	Motor angular velocity Omegam	—	
E2	Real	Load angular velocity Omegal	—	
Dv	Real	Torsion axis dumping set point	Dv:0.13~300.0	Default=1.00 (Note 1)
Dv/Kc	Real	Time constant torsion axis delay	1~1000ms	Default=10ms(Note 1)
Jl	Real	A load Inertia set point	0.0~0.999	Default=1.0(Note 2)
Dl	Real	A load dumping set point	0.0~0.999	Default=0.0(Note 3)
Kf(=1/Tf)	Real	Tf: Time constant filter	Tf:1~1000[ms]	Default=10.0ms(Note 4)
Lfc	Real	FCAN output bottom value	%	
Hfc	Real	FCANoutput upper limit value	%	

(Note 1)Dumping [Nm / the s/rad] X rating speed [rad/s]/ rating torque of the torsion axis dumping set point Dv= torsion axis[Nm]
The spring constant [Nm/rad] X rating speed [rad/s]/ rating torque of the torsion axis spring constant set point Kc= torsion axis[Nm]
(Note 2) Inertia [kgm2] X rating speed [rad/s]/ rating torque of the load Inertia set point (Jln)= load[Nm]
(Note 3)Dumping of the load dumping set point (Dln)= load[Nm / s/rad] X rating speed [rad/s]/ rating torque[Nm]
(Note 4)When Tf which is a reciprocal number of Kf set the value that is shorter than the double of the practice (operation) period, I do not become a right operation result. Please set it so that Tf which is a reciprocal number of Kf becomes the value that is longer than double of sample time.

Kind	Name	Symbol	Execution time
Data flow language (Function4) SHPC-115-Z Only	Feed forward (Modern control)		―――――――
Function	I compensate you for delay (a difference) of the real rotary speed in consideration of Inertia coefficient Jn and dumping coefficient Dn for a motor rotary speed order value.		



Input signal

Variable	Type	Contents	Range/Unit	Remarks
B1	Relay	A stop switch		
B2	Relay	A Power Con reshuffling switch		
E1	Real	Motor speed order Wref	%	
E2	Real	A Power Con coefficient Zn	%	
Jn	Real	Inertia set point	0.000～31.999	Default=1.0(Note1)
Dn	Real	Dumping set point	0.0～0.999	Default=0.0(Note2)
Tf	Real	Time constant filter	(Tf:1～1000[ms])	(Note3)
Hd	Real	The output upper limit value	-163.0%～163.0%	
Ld	Real	The output bottom value	-163.0%～163.0%	

Output signal

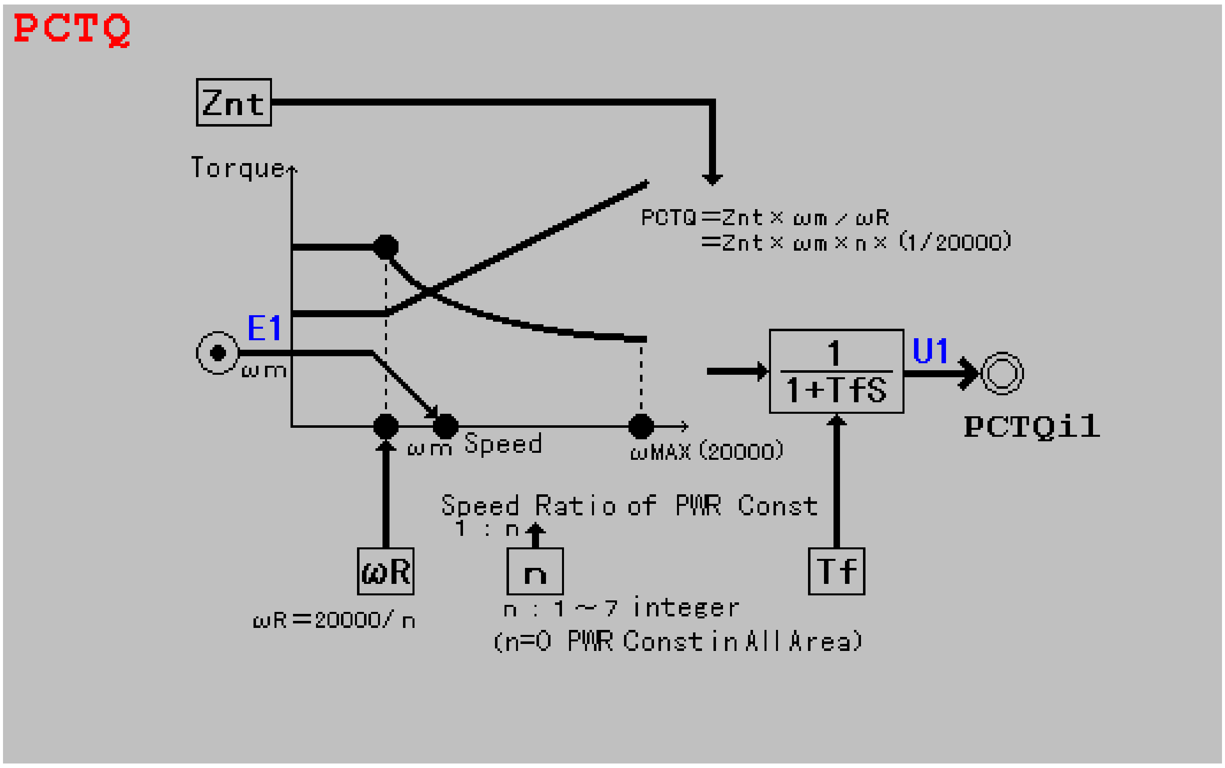
Variable	Type	Contents	Range/Unit	Remarks
U1	Real	FFW Doutput	%	

(Note 1) A Inertia set point (Inertia [kgm2] X rating speed [rad/s] / rating torque of the Jn)= motor axis conversion)[Nm]

(Note 2) A dumping set point (dumping of the Dn)= motor axis conversion)[Nm / s/rad] X rating speed [rad/s] / rating torque[Nm]

(Note 3) When Tf which is a reciprocal number of Kf set the value that is shorter than the double of the practice (operation) period, I do not become a right operation result. Please set it so that Tf which is a reciprocal number of Kf becomes the value that is longer than double of sample time.

Kind	Name	Symbol	Execution time
Data flow language (Function4) SHPC-115-Z Only	Power Con coefficient （Reciprocal number of the torque outbreak coefficient）block	PCTQ ƒ	――――
Function	I output electricity uniformity and the torque order that I converted so that it is it at the turn speed that is higher than base turn speed by inputting a torque order and real motor turn speed.		



Input signal

Variable	Type	Contents	Range/Unit	Remarks
E1	Real	Input Wm	%	
wR	Real	Rating (a base) speed	100.0～40000.0	
Znt	Real	A gain（The reciprocal number of the torque outbreak coefficient）	0.001～15.999	
Tf	Real	A delay gain (Tf)： Time constant delay)	1～10000[ms]	（Note1）

Output signal

Variable	Type	Contents	Range/Unit	Remarks
U1	Real	PCTQ Output	―	

（Note 1）When Tf which is a reciprocal number of Kf set the value that is shorter than the double of the practice (operation) period, I do not become a right operation result.
Please set it so that Tf which is a reciprocal number of Kf becomes the value that is longer than double of sample time.

Chapter 6 Appendix

(Appendix 1) Symbols and their Names

(1) LD language

Table 6.1

A-contact	B-contact	Logical reversal	Coil	Connector load	Connector store
			( )	①	①
Label	Jump	Return			
XXXXXX L	(JPXXXX)	(RETURN)			

(2) Data flow language (Basic)

Table 6.2

Load	Store & load	Store	a-contact	b-contact	c-contact
c-contact	Compare high	Compare low	Compare equal	Priority given to a upper-level	Priority given to a lower-level
	>	<	=		
Logical multiplication	Logical sum	Exclusive OR	Addition	Subtraction	Multiplication
			+	−	×
Division	Remainder	Local constant: integer	Local constant: real number		
/	%	i	r		

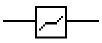
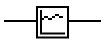
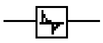
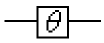
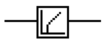
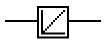
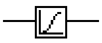
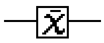
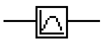
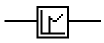
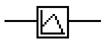
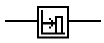
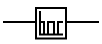
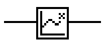
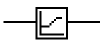
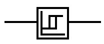
(3) Data flow language (Function 1)

Table 6.3

Code conversion	1's complement	Absolute value conversion	Increment	Decrement	Half
			++	−−	½
Two times	Square	Exponent	Square root	Bit count	Gray code binary
×2	↑2	↑N	√	B.C	G.B

(4) Data flow language (Function 2)

Table 6.4

Insensitive band	Pattern	Differential compensation	Phase compensation	PI compensation	ARC
					
S-ARC	Arithmetic average	Filter	PID compensation	Temporary delay	Delay
					
Constant frequency pulse	Variable setting pattern	Upper and lower limiters	Hysteresis	Unconditional subroutine	Conditional subroutine
				XXXXXX Sb	XXXXXX SB

(5) Data flow language (Function 3)

Table 6.5

Sine	Cosine	Tangent	Cosecant	Secant	Cotangent
SIN f	COS f	TAN f	ASIN f	ACOS f	ATAN f
On timer	Off timer	On differential	Off differential	Backlash	Backlash correction
TSTD f	TRTC f	USUC f	DSDC f	BKLS f	BKLC f
Scaling	Binary Gray conversion	Division and remainder	Integer conversion	Real number conversion	
SCAL f	BTOG f	DIVMOD f	TODINT f	TOREAL f	

(6) Data flow language (Function 6)

Table 6.6

Channel open	Message transmittal	Message receiving	Matrix	Set	Reset
M_OPEN [f]	M_SEND [f]	M_RECV [f]	MATRIX [f]	SET [F]	RESET [F]
Data transfer	Data transfer	Counter	Versatile communications		
MOVW [F]	MOVWD [F]	UPDOWN [F]	C_FREE [F]		
Motor side cancellation	Flexible side cancellation	Feed forward	Power Con coefficient		
MCAN [f]	FCAN [f]	FFWD [f]	PCTQ [f]		

TOYODENKI SEIZO K.K.
http://www.toyodenki.co.jp/
HEAD OFFICE: Tokyo Tatemono Yaesu Bldg, 1-4-16 Yaesu, Chuoh-ku,
Tokyo, Japan ZIP CODE 103-0028
TEL : +81-3-5202-8132 - 6
FAX : +81-3-5202-8150