HITACHI INVERTER

EASY-SEQUENCE
PROGRAMMING SOFTWARE（ＥｚＳＱ）

INSTRUCTION MANUAL

Read through this Instruction Manual, and keep it handy for future reference.

NT2021XA

HITACHI

Introduction

Introduction

Thank you for purchasing the Hitachi Inverter.
This Instruction Manual explains how to use the easy-sequence programming software (EzSQ) for the Hitachi SJ700/L700/SJ700B/WJ200 Series Inverter. Be sure to read this Instruction Manual carefully before using EzSQ, and keep it on hand for future reference.
Before creating user programs for the inverter, also refer to the Inverter Instruction Manual and Configuration software (ProDriveNext) Instruction Manual for the necessary related knowledge, and ensure you understand and follow all safety information, precautions, and operating and handling instructions for the correct use of the inverter.
Always use the inverter strictly within the range of specifications described in the Inverter Instruction Manual and correctly implement maintenance and inspections to prevent faults from occurring.
When using the inverter together with optional products, also read the manuals for those products. Note that this Instruction Manual and the manual for each optional product to be used should be delivered to the end user of the inverter.

Handling of this Instruction Manual

- The contents of this Instruction Manual are subject to change without prior notice.
- Even if you lose this Instruction Manual, it will not be resupplied, so please keep it carefully.
- No part of this Instruction Manual may be reproduced in any form without the publisher's permission.
- If you find any incorrect description, missing description or have a question concerning the contents of this Instruction Manual, please contact the publisher.

Revision History

No.	Revision content	Date of issue	Manual code
1	Initial release		NT2021X
2	Added L700/SJ700B Corrected Range of values and Default of ACCEL and DECEL	2011/3	NT2021XA

- The current edition of this Instruction Manual also includes some corrections of simple misprints, missing letters, misdescriptions and certain added explanations other than those listed in the above Revision History table.

Safety Instructions

Be sure to read this Instruction Manual, Inverter Instruction Manual, and appended documents thoroughly before using EzSQ and the inverter.
In these Instruction Manuals, safety instructions are classified into two levels: WARNING and CAUTION.

WARNING : Indicates that incorrect handling may cause hazardous situations, which may result in serious personal injury or death.

CAUTION : Indicates that incorrect handling may cause hazardous situations, which may result in moderate or slight personal injury or physical damage alone.

Note that even a CAUTION level situation may lead to a serious consequence according to circumstances. Be sure to follow every safety instruction, which contains important safety information.
Also focus on and observe the items and instructions described under "Notes" in the text.

WARNING

During trial operation of the inverter with a user program, a user program error may cause the motor driven by the inverter to run uncontrollably. Be sure to implement safety measures such as the emergency stop mechanism in your system before trial operation. Otherwise, system failure or personal injury may result.

CAUTION

To debug a user program, first conduct a trial operation of the inverter with an independent motor to confirm that the motor does not run uncontrollably. After that, install the motor in your system (machine), and start system operation. Otherwise, system failure or personal injury may result.

Safety Instructions

Contents

Chapter 1 Outline of EzSQ

1.1 Outline 1-1
1.2 Corresponding Model 1-1
1.3 Specifications 1-1
1.4 Preparation and System Configuration 1-2
1.5 General Flow of Operation and Setup 1-3
1.6 Notice 1-3

Chapter 2 Syntax

2.1 Description Format 2-1
(1) Program Description Format 2-1
(2) Data Description Format 2-2
(3) Multitasking function 2-2

2.2 List of Instructions 2-3
(1) Program control instructions 2-3
(2) Conditional expressions 2-4
(3) Operational instructions 2-4
(4) Input / output control, timer control, and inverter control instructions 2-5
(5) Variables 2-6
(6) Numeric values 2-6

2.3 Program Control Instructions 2-7
entry and end statements 2-7
sub and end sub statements 2-7
goto statement 2-8
on trip goto statement 2-8
ifs-then-else-end if statements 2-9
if statement 2-10
for-next loop statements 2-11
while loop statement 2-12
until loop statement 2-13
select case syntax statement 2-14
call statement 2-15
inc statement 2-15
dec statement 2-16
Label definition statement 2-16
wait statement 2-17

2.4 Input/Output Control Instructions 2-18
X ( ) or Xw (contact input) 2-18
Y ( ) or Yw (contact output) 2-19
UB ( ) or UBw (internal user contact control) 2-22

2.5 Timer Control Instructions 2-23
timer set (timer-start instruction) 2-24
timer off (timer-stop instruction) 2-25
delay on or delay off (delay operation instruction) 2-25

Contents

2.6 Inverter Control Instructions ····· 2-27
Inverter operation command ····· 2-27
Inverter operation monitoring instruction ····· 2-28
User Monitor ····· 2-29
User Trip ····· 2-29
stop statement ····· 2-30
chg param statement ····· 2-30
mon param statement ····· 2-31
eepwrt ····· 2-32
rtcset on, rtcset off ····· 2-33

2.7 Other Reserved Variables ····· 2-34
U (00) to U (31) ····· 2-34
UL (00) to UL (07) ····· 2-34
SET-Freq ····· 2-35
ACCEL ····· 2-36
DECEL ····· 2-37
XA (0) to XA (2) ····· 2-38
YA (0) to YA (2) ····· 2-39
TD (0) to TD (7), TDw ····· 2-41

2.8 Inverter Montor Variables ····· 2-42
FM ····· 2-42
Iout ····· 2-43
Dir ····· 2-44
PID-FB ····· 2-45
F-CNV ····· 2-45
Tmon ····· 2-46
Vout ····· 2-47
Power ····· 2-47
PlsCnt ····· 2-48
POS ····· 2-48
STATUS ····· 2-49
DCV ····· 2-49
RUN-Time ····· 2-50
ON-Time ····· 2-50
ERR CNT ····· 2-51
ERR (1) to ERR (6) ····· 2-51

Chapter 3 Interface with the Inverter

3.1 Inverter Settings ····· 3-1
(1)SJ700/L700/SJ700B Series ····· 3-1
(2)WJ200 Series ····· 3-2

3.2 Switching of Operation ····· 3-3
(1) Easy sequence function selection (A017) ····· 3-3

3.3 Switching of Input / Output Terminals ····· 3-3
(1) Program run signal input terminal (PRG terminal) ····· 3-3
(2) General-purpose contact input terminals ····· 3-3
(3) General-purpose contact output terminals ····· 3-4
(4) General-purpose analog input terminal (O terminal) ····· 3-5
(5) General-purpose analog input terminal (OI terminal) ····· 3-5
(6) General-purpose analog input terminal (O2 terminal) ····· 3-5
(7) General-purpose analog output terminal
(FM terminal in SJ700/L700/SJ700B / EO terminal in WJ200) ····· 3-6
(8) General-purpose analog output terminal (AM terminal) ····· 3-6
(9) General-purpose analog output terminal (AMI terminal) ····· 3-7

3.4 Switching of Command Input Device ····· 3-8
(1) Frequency source setting (A001 / A201) ····· 3-8
(2) Run command source setting (A002 / A202) ····· 3-8
(3) Accel / decel time input selection (P031) ····· 3-8

3.5 Others ····· 3-9
(1) User-defined variables “U (00)” to “U (31)” (P100 to P131) ····· 3-9
(2) User monitor “Umon (00)” to “Umon (02)” (d025 to d027) ····· 3-9
(3) User trip “trip 0” to “trip 9” (Error code E50 to E59) ····· 3-9

Chapter 4 Errors and Troubleshooting

4.1 Errors Specific to the Easy Sequence Function ····· 4-1

4.2 Troubleshooting ····· 4-2

Contents

Chapter 1 Outline of EzSQ

This chapter explains the general procedures for creating and executing a user program.

1.1 Outline ····· 1-1

1.2 Corresponding Model ····· 1-1

1.3 Specifications ····· 1-1

1.4 Preparation and System Configuration ····· 1-2

1.5 General Flow of Operation and Setup ····· 1-3

1.6 Notice ····· 1-3

Chapter 1 Outline of EzSQ

1.1 Outline

Easy sequence function (EzSQ) can built a simple sequence function by making the program with programming software ProDriveNext. In the program, it is possible to change the I/O function and the parameter setting value.

1.2 Corresponding Model

- SJ700 series - WJ200 series - L700 series - SJ700B series

1.3 Specifications

The table below lists the programming-related specifications of the easy sequence function.

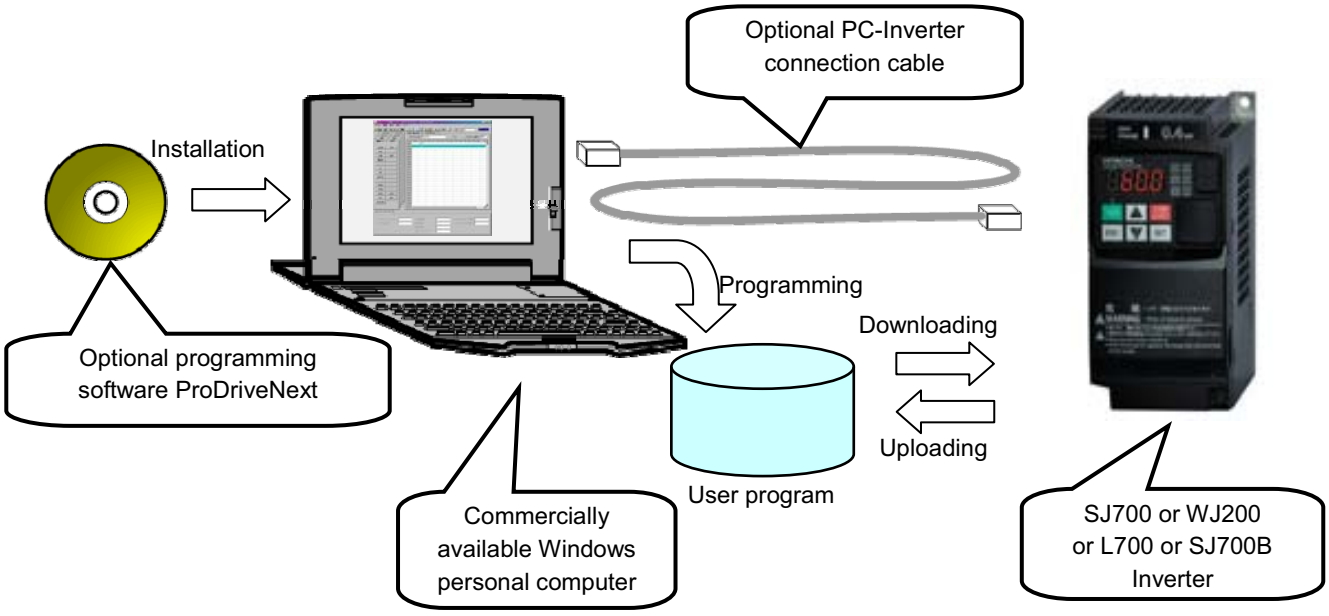
Item		Specification		
Language specification	Programming language	Basic-like language		
	Input device	Windows (DOS/V) personal computer (OS:Windows2000,WindowsXP)		
	Max. program size	1024steps (The internal storage capacity of the inverter is 1024 steps or 6144 bytes.)		
	Programming support function (programming software)	- Editing (on Windows) / - Display (on Windows) - Program syntax check (on Windows) - Downloading, uploading, and full clearance of program		
	Execution format	Execution by interpreter in an execution cycle of 2 ms per instruction (possible subroutine call with nesting in up to 8 layers)		
Input/output-related functions	External input	Contact signal		24 V open-collector input (using intelligent input terminals)
			Program run signal input	SJ700/L700/SJ700B : Always assigned to the FW terminal
				WJ200 : Assign to the PRG terminal / Always run
			Program run signal input	SJ700/L700/SJ700B : Up to 8 terminals (X (00) to X (07))
				WJ200 : Up to 8 terminals (X (00) to X (07))
		General-purpose analog input		XA (0): 0 to 10 V (O terminal)
				XA (1): 4 to 20 mA (OI terminal)
				XA (2): 0 to 10 V (O2 terminal) (Only SJ700)
	External output	General-purpose output terminal		SJ700/L700/SJ700B : Up to 6 terminals (Y (00) to Y (05))
				WJ200 : Up to 3 terminals (Y (00) to Y (02))
		General-purpose analog output		YA (0): Assignable to the FM terminal
				YA (1): Assignable to the AM terminal
				YA (2): Assignable to the AMI terminal (Only SJ700)
Reserved words	Instructions	(1) Program control instructions - Loop (for) / - Unconditional branching (goto) / - Time control (wait) - Conditional branching (if then, ifs then, select case, until, and while) - Subroutine (call, sub) / - Others (entry, end, cont, inc, and dec) (2) Arithmetic instructions - Arithmetic operation (+, -, *, /) / - Remainder (mod) / - Substitution (=) - Absolute value (abs) / - Logic operation (or, and, xor, and not) (3) Input/output control - General-purpose input/output (bit input, word input, bit output, and word output) - Reading of inverter input terminal (4) Timer control : - Delay operation / - Timer control (5) Parameter control : - Rewriting of parameters by reselecting code on the operator's display		
	Number of variables	User-defined variable		U (00) to U (31) (32 variables)
		Internal user variable		UL (00) to UL (07) (8 variables)
		Set frequency		SET-Freq
		Acceleration time		ACCEL
		Deceleration time		DECEL
		Monitoring variable		FM, Iout, Dir, PID-FB, F-CNV, Tmon, Vout, Power, RUN-Time, ON-Time, PlsCnt (Only SJ700/L700/SJ700B), POS, STATUS, DCV, ERR CNT, ERR(1), ERR(2), ERR(3), ERR(4), ERR(5), and ERR(6)
		General-purpose input contact		SJ700/L700/SJ700B : X (00) to X (07) (8 contacts)
				WJ200 : X (00) to X (07) (8 contacts)
		General-purpose output contact		SJ700/L700/SJ700B : Y (00) to Y (05) (6 contacts) (including a relay contact output)
				WJ200 : Y (00) to Y (02) (3 contacts) (including a relay contact output)
		Internal user contact		UB (00) to UB (07) (8 contacts)
		Internal timer contact		TD (0) to TD (7) (8 contacts)
		Inverter input/output		Specification by code on the remote operator's display
		User monitor		Umon (00) to Umon (02) (3 variables)
		User trip		Makes the inverter trip (10 variables)

1.4 Preparation and System Configuration

To create user programs with the easy sequence function of the inverter, you must prepare the following devices and software:

(1) SJ700 or WJ200 or L700 or SJ700B inverter
(2) Personal computer (PC) (Windows system)
(3) Optional programming software ProDriveNext
(4) Optional PC-inverter connection cable
SJ700
Inverter port: Operator-connection port
WJ200
Inverter port: USBminiB connector

The following figure shows the basic system configuration for programming.



- Install ProDriveNext on your Windows personal computer, and connect the personal computer to the inverter (SJ700 or WJ200 or L700 or SJ700B) via the PC-inverter connection cable.
- After completing these preparations, you can operate ProDriveNext to create a user program and download it to the inverter.

The table below lists the main functions of ProDriveNext. Please refer to the manual of ProDriveNext for use.

Function	Description
Programming support	Supports the input, editing, saving, reading, and printing of user programs
Compilation	Compiles an edited user program
Downloading and uploading	Downloads a user program to the inverter Uploads a user program to from the inverter
Debugging support	Monitors program execution, inverter status, and others

1.5 General Flow of Operation and Setup

A general flow of operations from programming to program execution with the easy sequence function is as follows:

No.	Description	Remarks
1	Create a user program with the ProDriveNext.	- Please refer to the manual of ProDriveNext for use. - For details on the syntax, see Chapter 4.
2	Compile and format that can be run on inverter. When a user program is compiled, the codes are checked for validity. If a syntax error is detected, ProDriveNext stops compilation and displays an error message.	
3	Download the compiled user program to the inverter, and save it in EEPROM. (*1)	
4	Configure the parameters required for the easy sequence function in the inverter.	Please refer to Chapter 3, “Interface with the Inverter”.
5	Enable the easy sequence function (set "01" or “02” in parameter "A017").	
6	When A017 = 01, turn on the PRG terminal (FW terminal in SJ700/ L700/ SJ700B) to execute the user program. When A017 = 02 in WJ200, the user program runs automatically after turning on the power. (*2)	

*1 If the downloaded user program is saved in internal EEPROM of the inverter, you can execute the user program even after resetting the inverter power. If the downloaded user program is not saved in EEPROM, the user program will be deleted when the inverter power is fully shut off. You are recommended not to save a created user program when downloading it to the inverter for debugging purposes. You should save the user program when downloading it again after debugging.

*2 After having downloaded the user program to the inverter, you can disconnect the inverter from the personal computer and execute the user program on the inverter alone.

1.6 Notice

(1) The format which can be saved by ProDriveNext is only a CSV file.
(2) The specification is different in SJ700, L700, SJ700B and WJ200.
(3) If RS terminal is turned on, a program counter is reset and the user program runs from the program head. However, the user program is restarted from the program counter before reset at C102=03.
(4) Do not shut off the power supply of the inverter while writing data in EEPROM by "eepwrt" command.

Chapter 2 Syntax

This chapter explains the syntax and definitions used for programming.

2.1 Description Format ····· 2-1

2.2 List of Instructions ····· 2-3

2.3 Program Control Instructions ····· 2-7

2.4 Input/Output Control Instructions ····· 2-18

2.5 Timer Control Instructions ····· 2-23

2.6 Inverter Control Instructions ····· 2-27

2.7 Other Reserved Variables ····· 2-34

2.8 Inverter Montor Variables ····· 2-42

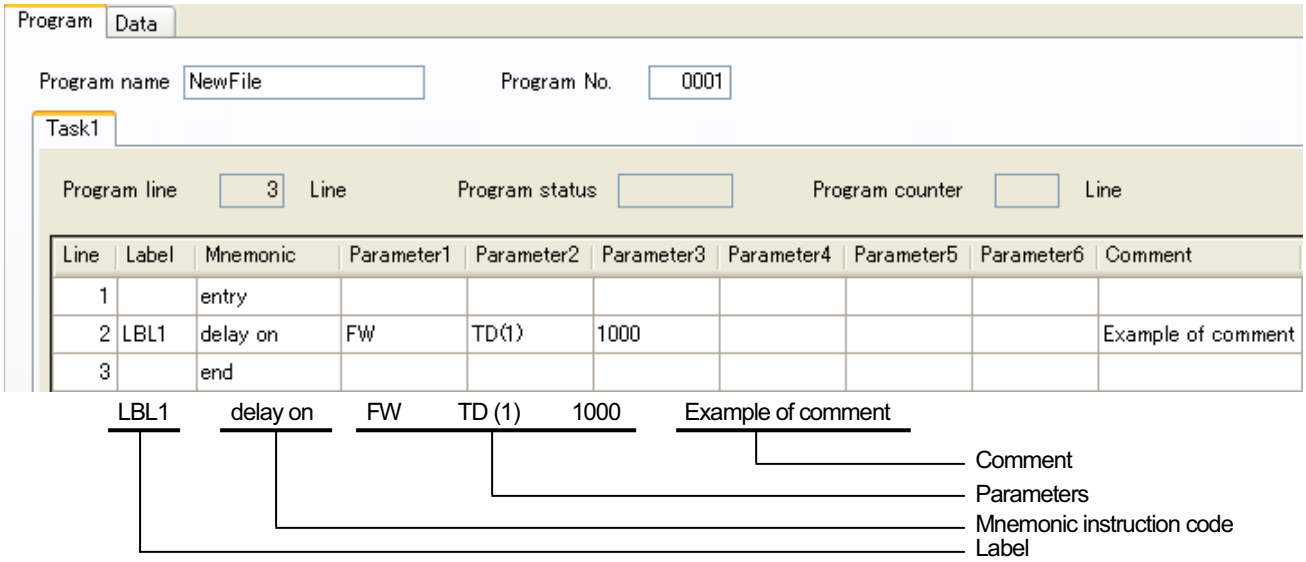
Chapter 2 Syntax

2.1 Description Format

(1) Program Description Format

Each line of a program consists of the “Label,” “Mnemonic,” “Parm 1 to 6,” and “Comment” fields. The “Mnemonic” field is used to describe an instruction word. Some instruction words do not require parameters.

(Example)



[1] Line : A line is the instruction unit of a program. You can describe one instruction word per line. It takes 2 ms to execute each line. One instruction corresponds to one step of the program.

[2] Label : Use the “Label” field to describe, for example, the branch destination for a branch instruction.

[3] Mnemonic : Use the “Mnemonic” field to describe the instruction to be executed. For details on the instructions, see Chapter 5, “Instruction Words.”

[4] Parameters : Use the “Parameter 1 to 6” fields to describe the arguments required to execute an instruction. Up to six arguments can be described as required for the instruction word described on the same line.

[5] Comment : Use the “Comment” field to describe a comment on each line.

Note 1 : Please describe the instruction in the line that describes the label. The program might not upload correctly if the instruction is not described.

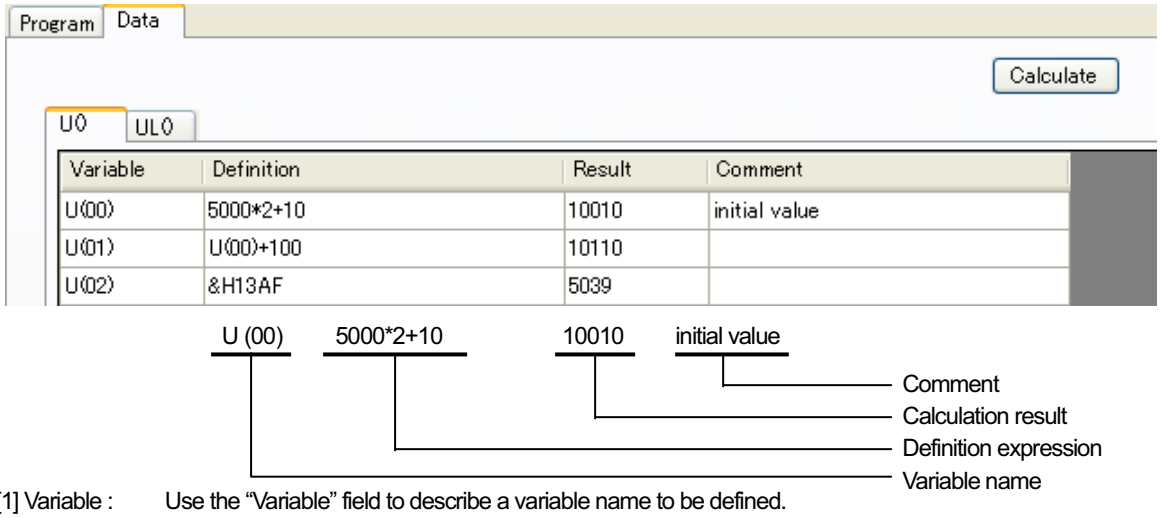
Note 2 : The item that can be uploaded are Label, Mnemonic, and Parameters. Comment cannot be uploaded. Moreover, when the program is uploaded, the label name is changed.

Note 3 : When a program including the line where nothing is described was downloaded and that program is uploaded, the empty line is deleted. In addition, because program counter monitor (d023) doesn't count the empty line, the number of Line is not corresponding to the value of the program counter.
Example) When an instruction of the fifth line is executed in a program including the empty line in the second line and the third line, a value of the program counter is 3.

(2) Data Description Format

Each variable is described on a line that consists of the “Variable,” “Define,” “Answer,” and “Comment” fields.

(Example)



[1] Variable : Use the “Variable” field to describe a variable name to be defined.

[2] Definition : Use the “Definition” field to describe a definition expression for a variable. A definition expression can be a variable name. Clicking the [Calculate] button in the “Data Window” after entering a definition expression starts calculation. (If the calculation executed for a variable by clicking the [Calculate] button results in a value that is outside the range of numeric values defined for the variable, the “Data range is invalid!!” message will appear and the “Result” field will indicate “<Range is invalid.>.”)

[3] Result : The “Result” field displays a calculation result. You cannot rewrite the calculation result.

[4] Comment : You can describe a comment about the variable described on the line.

Note : The item that can be uploaded is only Result. Definition and Comment cannot be uploaded.

(3) Multitasking function

In WJ200 series, two or more programs (a maximum of 5 tasks) can be simultaneously performed by the multitasking function. One line of each task is executed at 2ms cycle. Please refer to the manual of ProDriveNext for the method of making two or more tasks.

The limitations in multitasking are shown below.

Limitations	Description
Subroutine CALL	Subroutine CALL between tasks is not made.
Movement to the label	It cannot move between tasks to the label by the goto instruction and the if instruction, etc.
Variables	The variables such as user-defined variable U (00) become common by each task.
Timer	When plural tasks carry out timer set / timer off for the same timer number, it will not work normally.
Display of d023	Inverter function cord d023 displays program counter of task 1.

2.2 List of Instructions

This section lists the instructions that can be used in a program.

(1) Program control instructions

Instruction name	Instruction format						Description
	Mnemonic code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5	
entry statement	entry						Indicates the beginning of the main program.
	end						Indicates the end of the main program.
sub statement	sub	<subroutine name>					Indicates the beginning of a subroutine.
	end sub						Indicates the end of a subroutine.
call statement	call	<subroutine name>					Branches processing to the subroutine specified by <subroutine name>.
for loop statement	for	<variable>	<start value>	<end value>	<incremental value>		Executes <instruction set> repeatedly until <variable> reaches <end value>. Note that <variable>, which initially contains <start value>, is incremented by <incremental value> each time <instruction set> is executed.
	<instruction set>						Indicates the instructions to be executed repeatedly.
	next						Ends the "for" loop.
goto statement	goto	<label name>					Branches processing unconditionally to the step labeled with <label name>.
on trip goto statement	on	trip	goto	<label name>			Branches processing to the step labeled with <label name> when the inverter trips.
if statement	if	<condition>			then	<label name>	Branches processing to the step labeled with <label name> when the <condition> is met.
structured if	ifs	<condition>					Starts the structured if statement.
	then						Indicates the beginning of instructions to be executed when <condition> is met.
	<instruction set>						Indicates the instructions to be executed when <condition> is met.
	else						Indicates the beginning of instructions to be executed when <condition> is not met.
	<instruction set>						Indicates the instructions to be executed when <condition> is not met.
	end if						Ends the structured if statement.
select case syntax statement	select	<conditional variable>					Executes the instructions specified after "case" when the value of <conditional variable> is <conditional value>.
	case	<conditional value>					Indicates the conditional value and the beginning of instructions to be executed.
	[case else]						Indicates the beginning of instructions to be executed when the value of <conditional variable> is not <conditional value>.
	end select						Ends the select case syntax statement.
until loop statement	until	<condition>					Executes <instruction set> repeatedly until <condition> is met.
	<instruction set>						Indicates the instructions to be executed while <condition> is not met.
	loop						Ends the "until" loop.
wait loop statement	wait	***.**					Waits for "***.**" seconds.
		<variable>					Waits for <variable> × 10ms.
		<condition>					Waits until <condition> is met.
while loop statement	while	<condition>					Executes <instruction set> while <condition> is met.
	<instruction set>						Indicates the instructions to be executed while <condition> is met.
	wend						Ends the "while" loop.
inc statement	inc	<variable>					Increments the value of <variable> by 1.
dec statement	dec	<variable>					Decrements the value of <variable> by 1.

(2) Conditional expressions

The table below lists the conditional expressions that can be used for the <condition> parameters in program control instructions.

Instruction name	Instruction format						Description
	Mnemonic code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5	
Comparison		<variable 2 / constant>	=	<variable 3/ constant>			"True" when <variable 2/constant> is equal to <variable 3/constant>
		<variable 2 / constant>	<	<variable 3/ constant>			"True" when <variable 2/constant> is less than <variable 3/constant>
		<variable 2 / constant>	<=	<variable 3/ constant>			"True" when <variable 2/constant> is not greater than <variable 3/constant>
		<variable 2 / constant>	>	<variable 3/ constant>			"True" when <variable 2/constant> is greater than <variable 3/constant>
		<variable 2 / constant>	>=	<variable 3/ constant>			"True" when <variable 2/constant> is not less than <variable 3/constant>
		<variable 2 / constant>	<>	<variable 3/ constant>			"True" when <variable 2/constant> is not equal to <variable 3/constant>

Note : <variable 1> and <variable 2> can be constants ranging from 0 to 127.

(3) Operational instructions

Instruction name	Instruction format						Description
	Mnemonic code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5	
Arithmetic operation	<variable 1> =	<variable 2/ constant>	+	<variable 3/ constant>			Adds <variable 2/constant> and <variable 3/constant> and assigns the result to <variable 1>.
	<variable 1> =	<variable 2/ constant>	–	<variable 3/ constant>			Subtracts <variable 3/constant> from <variable 2/constant> and assigns the result to <variable 1>.
	<variable 1> =	<variable 2/ constant>	*	<variable 3/ constant>			Multiplies <variable 2/constant> by <variable 3/constant> and assigns the result to <variable 1>.
	<variable 1> =	<variable 2/ constant>	/	<variable 3/ constant>			Divides <variable 2/constant> by <variable 3/constant> and assigns the result to <variable 1>.
Remainder	<variable 1> =	<variable 2/ constant>	mod	<variable 3/ constant>			Divides <variable 2/constant> by <variable 3/constant> and assigns the remainder to <variable 1>.
Absolute value	<variable 1> =		abs	<variable 3/ constant>			Assigns the absolute value of <variable 3/constant> to <variable 1>.
Substitution	<variable 1> =			<variable 3/ constant>			Assigns <variable 3/constant> to <variable 1>.
Logic operation	<variable 1> =	<variable 2/ constant>	or	<variable 3/ constant>			Assigns the OR of <variable 2/constant> and <variable 3/constant> to <variable 1>.
	<variable 1> =	<variable 2/ constant>	and	<variable 3/ constant>			Assigns the AND of <variable 2/constant> and <variable 3/constant> to <variable 1>.
	<variable 1> =	<variable 2/ constant>	xor	<variable 3/ constant>			Assigns the XOR of <variable 2/constant> and <variable 3/constant> to <variable 1>.
	<variable 1> =		not	<variable 3/ constant>			Inverts the bits of <variable 3/constant> and assigns the inverted bits to <variable 1>.

Note 1 : <variable 2> can be a constant ranging from 0 to 127.

Note 2 : <variable 3> can be a constant ranging from 0 to 2^{31}-1.

(4) Input / output control, timer control, and inverter control instructions

Instruction name	Instruction format						Description
	Mnemonic code	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5	
General-purpose contact input	<variable> =	X (ii)					Fetches general-purpose contact information and stores it in <variable>. (0 = off, 1 = on)
	<variable> =	Xw					Fetches general-purpose contact information and stores it as word data in <variable>.
General-purpose contact output	Y (ii) =	<variable / constant)					Outputs bit data to a general-purpose contact. (0 = off, 1 = on)
	Yw =	<variable / constant)					Outputs word data to a general-purpose contact.
Inverter operation command	<input terminal> =	<variable / constant)					Operates an inverter input terminal. (0 = off, 1 = on)
Inverter operation monitoring	<variable> =	<output terminal>					Fetches information from an inverter output terminal.
	<variable> =	<input terminal> =					Fetches information from an inverter input terminal.
Delay operation	delay on	<variable 1>	TD (k)	<variable 2/ constant>			Turns on the terminal specified by <variable 1> after the time specified by <variable2/constant> elapses.
	delay off	<variable 1>	TD (k)	<variable 2/ constant>			Turns off the terminal specified by <variable 1> after the time specified by <variable2/constant> elapses.
Timer control	timer set	TD (k)	<variable / constant)				Sets <variable/constant> in a specified timer and starts the timer.
	timer off	TD (k)					Stops the specified timer.
Internal user contact control	<variable> =	UB (ii)					Fetches internal user contact information and stores it in <variable>. (0 = off, 1 = on)
	<variable> =	UBw					Fetches internal user contact information and stores it as word data in <variable>.
	UB (ii) =	<variable / constant)					Outputs bit data to an internal user contact. (0 = off, 1 = on)
	UBw =	<variable / constant)					Outputs word data to an internal user contact.
Parameter change	chg param	<display code>	<variable / constant)				Replaces the content of <display code> with <variable/constant>.
Parameter reading	mon param	<display code>	<variable>				Reads the content of <display code> into <variable>.
Parameter writing	eepwrt						Stores a data of only one parameter changed by chg param command that is issued immediately after the execution of eeprwrt command to EEPROM. (SJ700/ L700/ SJ700B series doesn't correspond.)
Clock command	rtcset	on	<variable>				The clock data is substituted for six bytes that make <variable> a head. Moreover, the clock data is regularly updated. (Only WJ200 Step2 corresponds.)
	rtcset	off	<variable>				The clock data is substituted for six bytes that make <variable> a head. Moreover, the update of the clock data is stopped. (Only WJ200 Step2 corresponds.)
Stop inverter	stop						The inverter decelerate and stop the motor
User monitor	Umon(ii) =			<variable>			Displays <variable> on user monitor (ii)
	Umon(ii) =	<variable1>	<operators>	<variable2>			Displays the result of operation with <variable1> and <variable2> on user monitor (ii)
	<variable>=			Umon(ii)			Value of user monitor (ii) is read out to <variable>
User trip	trip	<variable>					Makes the inverter trip

(5) Variables

Type of variable	Variable name	Range of numeric values
Bit and contact variables	FW, X (00), etc.	0, 1 (0: OFF, 1: ON)
User-defined variable	U (00) to U (31)	0 to 65535
Internal user variable	UL (00) to UL (07)	-2147483648 to 2147483647
Frequency setting variable	SET-Freq	0 to 40000
Acceleration / deceleration time setting variable	ACCEL, DECEL	0 to 360000
Monitoring and other variables	See Section 2.8, “Inverter Monitor Variables.”	

(6) Numeric values

Notation	Numeration	Remarks
(Omitted)	Decimal	Decimal number
&H	Hexadecimal	Hexadecimal number (specifiable only in the “Data Window”)
&B	Binary	Binary number (specifiable only in the “Data Window”)

2.3 Program Control Instructions

This section explains the details of program control instructions.

entry and end statements	Instructions to start and end the main program

- Format

Format	Description
entry	This instruction indicates the beginning of the main program. (This instruction must be described at the top of the main program.)
end	This instruction indicates the end of the main program.

- Explanation

The entry and end statements indicate the beginning and end of the main program, respectively. Each program always requires these instructions.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						: The main program begins.
	:						
	FW=	1					: Start forward rotation of the motor.
	wait	10.00					: Wait 10 seconds.
	stop						: Stop inverter output.
	wait	10.00					: Wait 10 seconds.
	:						
	end						: The main program ends.

sub and end sub statements	Instructions to start and end a subroutine

- Format

Format	Description
sub **<subroutine name>**	This instruction indicates the beginning of a subroutine.
end sub	This instruction indicates the end of a subroutine. Control is returned to the calling routine.

- Explanation

The sub and end sub statements indicate the beginning and end of a subroutine, respectively.

<subroutine name> : Specifies the name of a called subroutine. This subroutine name is the first argument (branch destination) of the call instruction in the calling routine.

Note : Subroutines can be nested in up to eight layers. A subroutine programmed with a structured instruction (i.e., sub, for, while, until, select, or ifs) is counted as one nesting layer. Therefore, when a for-next loop statement is described in a subroutine, there are two nesting layers.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	sub	sub1					: Subroutine "sub1" begins.
	:						
	FW=	1					: Start forward rotation of the motor.
	wait	10.00					: Wait 10 seconds.
	stop						: Stop inverter output.
	wait	10.00					: Wait 10 seconds.
	:						
	end sub						: Subroutine "sub1" ends.

goto statement	Instruction to branch processing unconditionally

- Format

Format	Description
goto **<label name>**	This instruction branches processing unconditionally to the step labeled with **<label name>**.

- Explanation

Use this instruction to branch processing unconditionally to the step labeled with **<label name>**.

<label name> : Specifies the label name of the branch-target step (line).

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	goto	LABEL1					: Unconditionally branch to step "LABEL1".
	:						
LABEL1	FW=	1					: Branch target

on trip goto statement	Instruction to branch processing upon the occurrence of an event

- Format

Format	Description
on trip goto **<label name>**	This instruction branches processing to the step labeled with **<label name>** when the inverter trips.

- Explanation

Use this instruction to branch processing to the step labeled with **<label name>** when the inverter trips.

In the SJ700 Series, when inverter trips without the description of this instruction, the program stops immediately after the occurrence of inverter trip.

In the WJ200 Series, when the user trip occurs without the description of this instruction, the program stops immediately after the occurrence of inverter trip.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	on	trip	goto	LABEL1			: Branch to step "LABEL1" when the inverter trips.
	:						
LABEL1	Y(00)=	1					: Branch target

ifs-then-else-end if statements	Structured if instruction

- Format

Format	Description
ifs **<condition>** [then] **<instruction set 1>** [else] **<instruction set 2>** end if	When **<condition>** is met, this instruction executes <instruction set 1> described between "then" and "else." When **<condition>** is not met, this instruction executes **<instruction set 2>** described between "else" and "end if."

- Explanation

This instruction executes different sets of instructions according to whether **<condition>** is met.
When **<condition>** is met, this instruction executes **<instruction set 1>.** When **<condition>** is not met, this instruction executes **<instruction set 2>**.
If neither "then **<instruction set 1>**" nor "else **<instruction set 2>**" is described, the ifs statement jumps to the end if statement.

<condition> : Specifies a conditional expression among those listed in Section 2.2, " List of Instructions (2) Conditional expressions."

<instruction set 1> : Specifies the instructions to be executed when **<condition>** is met. The instructions may be described on two or more lines. The instructions are executed in units of lines in a cycle as explained below.

<instruction set 2> : Specifies the instructions to be executed when **<condition>** is met. The instructions may be described on two or more lines. The instructions are executed in units of lines in a cycle as explained below.

- Processing cycle

Note that **<condition>** is checked in the first cycle, and the first instruction in **<instruction set 1>** or **<instruction set 2>** is executed in the second cycle. In the third cycle, the second instruction **<instruction set 1>** or **<instruction set 2>** is executed or, if no other instruction remains in the instruction set, processing jumps to the end if statement. Therefore, the routine from "ifs" to "end if" is executed in three cycles when the instruction set contains only one instruction.
Refer to the statement execution sequence indicated by parenthesized numbers in the comment fields of the sample programs below.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5		When <condition> is met	When <condition> is not met
	ifs	X(00)	=	1			:	(1)	(1)
	then								
	Y(00)=	1					:	(2)	
	Y(01)=	0					:	(3)	
	else								
	Y(00)=	0					:		(2)
	Y(01)=	1					:		(3)
	end if						:	(4)	(4)

if statement	Instruction to branch processing unconditionally

- Format

Format	Description
if **<condition>** then **<label name>**	When **<condition>** is met, processing branches to the step labeled with **<label name>**. When **<condition>** is not met, processing proceeds to the next step (line).

- Explanation

Use this instruction to branch processing conditionally.

When **<condition>** is met, processing branches to the step labeled with **<label name>** described after "then."

<condition> : Specifies a conditional expression among those listed in Section 2.2, " List of Instructions (2) Conditional expressions."

<label name> : Specifies the label name of the branch-target step (line).

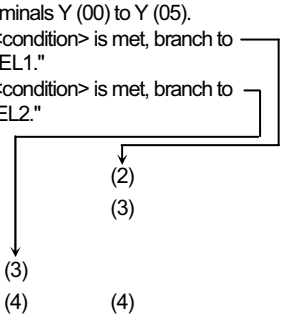
- Processing cycle

Note that **<condition>** check and branch processing are executed in the same cycle.

Refer to the statement execution sequence indicated by parenthesized numbers in the comment fields of the sample programs below.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	Yw=	0					: Turn off terminals Y (00) to Y (05).
	if	X(00)	=	1	then	LABEL1	: (1) When <condition> is met, branch to step "LABEL1."
	if	X(01)	=	1	then	LABEL2	: (2) When <condition> is met, branch to step "LABEL2."
	Y(00)=	1					: (3)
	goto	LABEL3					: (4)
LABEL1	Y(01)=	1					:
	goto	LABEL3					:
LABEL2	Y(02)=	1					:
LABEL3	Y(03)=	1					: (5)



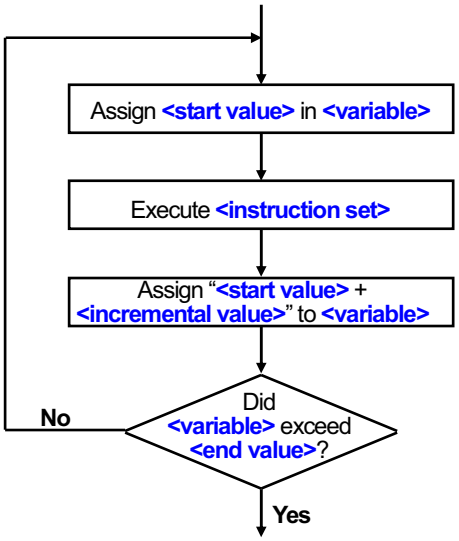
for-next loop statements	for loop instruction

- Format

Format	Description
for **<variable>** **<start value>** **<end value>** **<incremental value>** **<instruction set>** next	This loop instruction executes **<instruction set>** repeatedly until **<variable>** exceeds **<end value>**. Note that **<variable>**, which initially contains **<start value>**, is incremented by **<incremental value>** each time **<instruction set>** is executed.

- Explanation

Use the for loop statement to effectively describe a process for which the number of execution times is predetermined.
As a loop process, **<instruction set>** is executed and **<variable>** incremented by **<incremental value>** from **<start value>**.
If **<variable>** exceeds **<end value>**, processing exits the loop. Otherwise, the loop process is repeated. Therefore, **<instruction set>** is always executed at least once.
The following chart shows the flow of processing.



<variable> : Specifies the name of the variable to be used for the loop.

<start value> : Specifies the initial value of **<variable>** to be applied at the beginning of the loop. You can specify a variable name or immediate value (i.e., a value that can be entered directly). The immediate value must be an integer ranging from 0 to 127. To use a larger numerical value, preset the value in a variable and specify the variable as **<start value>**.

<end value> : Specifies the limit value at which to exit the loop. Processing exits the loop when **<variable>** exceeds **<end value>**. You can specify a variable name or immediate value (i.e., a value that can be entered directly). The immediate value must be an integer ranging from 0 to 127. To use a larger numerical value, preset the value in a variable and specify the variable as **<end value>**.

<incremental value> : Specifies the value to be added to **<variable>** each time the loop is executed. You can specify a variable name or immediate value (i.e., a value that can be entered directly). The immediate value must be an integer ranging from 0 to 127. To use a larger numerical value, preset the value in a variable and specify the variable as **<incremental value>**.

<instruction set> : Describes the set of instructions to be executed in one loop process. The instructions may be described on two or more lines. The instructions are executed in units of lines in a cycle as explained below.

- Processing cycle

Refer to the statement execution sequence indicated by parenthesized numbers in the comment fields of the sample programs below.

(1) : The "for" line is executed only once.
(2) and (3) : **<instruction set>** is executed.
(4) : **<variable>** is incremented in the cycle that follows the cycle in which the last instruction of **<instruction set>** is executed. Then, **<variable>** is checked to determine whether to exit the loop (in other words, the next statement is executed). When repeating the loop, processing returns to the first instruction of **<instruction set>** in this cycle.
(5) : This step is executed in the next cycle.
(6) to (10) : These steps are repeated in the same way as the preceding loop execution.
(11) : Processing proceeds to the following step (line).

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	Sequence of execution
	for	U(00)	0	3	1		:(1)
	Y(00)=	1					:(2) (5) (8)
	Y(00)=	0					:(3) (6) (9)
	next						:(4) (7) (10)
	Y(00)=	1					: (11)

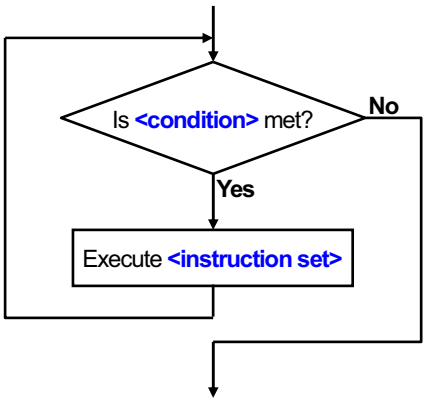
while loop statement — Instruction to conditionally execute a pre-conditioned loop

- Format

Format	Description
while **<condition>** **<instruction set>** wend	This instruction executes **<instruction set>** while **<condition>** is met. Note that **<condition>** is checked before the execution of **<instruction set>**.

- Explanation

This instruction executes **<instruction set>** repeatedly as long as **<condition>** is met. Note that **<condition>** is checked before the execution of **<instruction set>**. If **<condition>** is not met, processing proceeds to the wend statement without executing **<instruction set>**.



- Sample program (Condition "X (00) = 0" is met after the loop is executed twice.)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	Sequence of execution
	while	X(00)	=	1			:(1) (5) (9)
	Y(00)=	1					:(2) (6)
	Y(00)=	0					:(3) (7)
	wend						:(4) (8)
	Y(00)=	1					: (10)

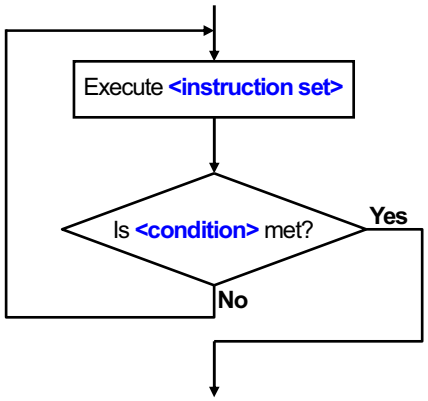
until loop statement	Instruction to conditionally execute a post-conditioned loop

- Format

Format	Description
until **<condition>** **<instruction set>** loop	This instruction executes <instruction set> until **<condition>** is met. Note that **<condition>** is checked after the execution of **<instruction set>**.

- Explanation

This instruction executes **<instruction set>** repeatedly until <condition> is met. Note that **<condition>** is checked after the execution of **<instruction set>**.



- Sample program (Condition "X (00) = 0" is met after the loop is executed twice.)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	Sequence of execution		
	until	X(00)	=	1			:(1)	(5)	(9)
	Y(00)=	1					:(2)	(6)	(10)
	Y(00)=	0					:(3)	(7)	(11)
	loop						:(4)	(8)	(12)
	Y(00)=	1					:		(13)

select case syntax statement	Instruction to branch under multiple conditions

- Format

Format	Description
select **<conditional variable>** case **<conditional value 1>** **<instruction set 1>** case **<conditional value 2>** **<instruction set 2>** ・・・ [case else] [**<instruction set n>**] end select	This instruction executes **<instruction set 1>** to **<instruction set n-1>** described in a case statement when **<conditional variable>** matches **<conditional value 1>** to **<conditional value n-1>** in the case statement, respectively. If the case else statement is described, **<instruction set n>** is executed when **<conditional variable>** does not match any of **<conditional value 1>** to **<conditional value n-1>**.

- Explanation

This instruction executes **<instruction set 1>** to **<instruction set n-1>** described in a case statement when **<conditional variable>** matches **<conditional value 1>** to **<conditional value n-1>** in the case statement, respectively. If the case else statement is described, **<instruction set n>** is executed when **<conditional variable>** does not match any of **<conditional value 1>** to **<conditional value n-1>**.

- Sample program (when Xw = 2)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	Sequence of execution
	U(00)=	Xw					: (1)
	select	U(00)					: (2)
	case	0					:
	Yw=	U(00)					:
	case	1					:
	Yw=	U(00)					:
	case	2					: (3)
	Yw=	U(00)					: (4)
	case	4					:
	Yw=	U(00)					:
	case else						:
	Yw=	0					:
	end select						: (5)
	Y(00)=	1					: (6)

call statement	Instruction to unconditionally branch to a subroutine

- Format

Format	Description
call **<subroutine name>**	This instruction branches processing unconditionally to the subroutine specified by **<subroutine name>**.

- Explanation

This instruction branches processing unconditionally to the subroutine specified by **<subroutine name>**.
After the subroutine is executed, processing proceeds to the instruction that follows the calling step.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
	:						
	call	SUB1					: Call subroutine "SUB1".
	:						
	end						
	sub	SUB1					: Called subroutine
	:						
	Y(00)=	1					
	:						
	sub end						

inc statement	Instruction to increment a variable

- Format

Format	Description
inc **<variable>**	This instruction increments **<variable>** by 1.

- Explanation

This instruction adds 1 to the value of **<variable>**.

- Sample program

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
LOOP	inc	U(00)					: Assign "U (00) + 1" to U (00).
	U(00)=	U(00)	and	U(01)			: Mask the 6 low-order bits of U (00).
	Yw=	U(00)					: Output the content of U (00) to terminals Y (00) to Y (05).
	wait	0.5					: Wait 0.5 second.
	goto	LOOP					
	end						

(Data area [Data Window])

U (00) = 255
U (01) = 63

dec statement — Instruction to decrement a variable

- Format

Format	Description
dec **<variable>**	This instruction decrements **<variable>** by 1.

- Explanation

This instruction subtracts 1 from the value of **<variable>**.

- Sample program

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
LOOP	dec	U(00)					: Assign "U (00) - 1" to U (00).
	U(00)=	U(00)	and	U(01)			: Mask the 6 low-order bits of U (00).
	Yw=	U(00)					: Output the content of U (00) to terminals Y (00) to Y (05).
	wait	0.5					: Wait 0.5 seconds.
	goto	LOOP					
	end						

(Data area [Data Window])

U (00) = 255
U (01) = 63

Label definition statement — Statement to define a label

- Format

Format	Description
<label name>	This statement defines **<label name>**.

- Explanation

Use this statement to define **<label name>** to be used in the goto or other instructions. The statement is not executed when described alone.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	goto	LABEL1					: Branch to step "LABEL1".
	:						
LABEL1	Y(00)=	1					: Branch-target step (line)

wait statement	**Instruction to make processing wait**

- Format

Format		Description
1	wait iii.ii	This instruction makes processing wait for “iii.ii” seconds.
2	wait <variable>	This instruction makes processing wait for “<variable> ×10 “ ms.
3	wait <condition>	This instruction makes processing wait until <condition> is met.

- Explanation

Format 1 : This instruction makes processing wait for “iii.ii” seconds. After “iii.ii” seconds elapse, the next instruction is executed.

Format 2 : This instruction makes processing wait for “**<variable>** ×10 “ ms. After “**<variable>** ×10 “ ms elapse, the next instruction is executed.

Format 3 : This instruction makes processing wait until **<condition>** is met. After **<condition>** is met, the next instruction is executed.

- Sample program

Sample 1 : Format 1

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	wait	1.00					: Wait 1 second.
	Y(00)=	1					
	:						

Sample 2 : Format 2

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	U(00)=			100			
	:						
	wait	U(00)					: Wait 100×10ms.
	Y(00)=	1					
	:						

Sample 3 : Format 3

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	wait	X(00)	=	1			: Wait until condition "X (00) = 1" is met.
LABEL1	Y(00)=	1					
	:						

2.4 Input/Output Control Instructions

This section describes the details of input/output control instructions.

X ( ) or Xw (contact input)	Instruction to access contact inputs

- Format

Format		Description
1	**<variable>** = X (ii) (SJ700 : ii = 00 to 07) (WJ200 : ii = 00 to 09)	This instruction assigns the ii'th bit of contact input data to **<variable>**.
2	**<variable>**= Xw	This instruction assigns contact input data as word data to **<variable>**.

- Explanation

This instruction fetches the status of contact input terminals X (ii) and stores it in **<variable>** in units of bits or words. You cannot write data to **<variable>**, which is read-only. Details of the formats are explained below.

Format 1 : With this format, the instruction assigns the status of the ii'th bit of contact input data to **<variable>**. (0 = off, 1 = on)
(Examples)
When terminal X (00) is off: UB= X (00) (UB (00) = 0)
When terminal X (00) is on: UB= X (01) (UB (00) = 1)

Format 2 : With this format, the instruction assigns the status of contact input data as word data to **<variable>**. (0 = off, 1 = on)
(Examples)
When terminals X (00) to X (03) are on and terminals X (04) to X (07) are off: Uw= Xw (Uw = 15)
When terminals X (00) to X (02) are off and terminals X (03) to X (07) are on: Uw= Xw (Uw = 248)

Note 1 : In the SJ700/L700/SJ700B Series, the setting of terminal active state (C011 to C018) is reflected in the polarity (on or off) of contact inputs X (00) to X (07) and Xw. When you create a user program, consider the on and off states of actual intelligent input terminals 1 to 8.
In the WJ200 Series, the setting of terminal active state (C011 to C017) is reflected in the polarity (on or off) of contact inputs X (00) to X (06) and Xw. When you create a user program, consider the on and off states of actual intelligent input terminals 1 to 7.

Note 2 : Since this instruction reads the internal input terminal data at least twice (in two execution cycles), storing the read data in <variable> is delayed by at least two execution cycles.

Note 3 : Wiring noise or switch chattering may cause incorrect read data to be set in <variable>. To avoid such problems, design your program so that it will verify the read data.

- Sample program

Sample 1 : Program to invert the status data of input terminal X (01) and output it to output terminal Y (05)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
LOOP	UB(00)=			X(01)			: Fetch the status of X (01) and store it as internal user contact data.
	ifs	UB(00)	=	0			: Branch according to the conditional expression.
	then						
	Y(05)=	1					: Turn Y (05) on when the condition is met.
	else						
	Y(05)=	0					: Turn Y (05) off when the condition is not met.
	end if						
	goto	LOOP					
	end						

Sample 2 : Program to acquire input terminal status as word data and output only the status of terminals X (02) to X (05) as word data to output terminals Y (00) to Y (03)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
LOOP	U(00)=	Xw					: Fetch input terminal status and store it in the user variable.
	U(00)=	U(00)	/	4			: Cut off the data of X (00) to X (01).
	U(00)=	U(00)	and	15			: Mask the data of X (06) to X (07).
	Yw=	U(00)					: Output the data of X (02) to X (05) as word data.
	goto	LOOP					
	end						

Y ( ) or Yw (contact output)	**Instruction to access contact outputs**

- Format

Format		Description
1	Y (ii)= **<variable>** or **<constant>** (ii = 00 to 05)	This instruction assigns **<variable>** or **<constant>** to the ii'th bit of contact output data.
2	Yw= **<variable>** or **<constant>**	This instruction assigns **<variable>** or **<constant>** as word data to contact outputs.

- Explanation

This instruction writes **<variable>** or **<contact>** to contact output terminals Y (00) to X (05) in units of bits or words to output the data. You can write and read data to and from **<variable>** or **<contact>**. You can also fetch and store the status data of contact output terminals Y (00) to Y (05) in **<variable>**. Details of the formats are explained below.

Format 1 : With this format, the instruction outputs **<variable>** to the ii'th bit of contact output terminal.
(0 = off, 1 = on, 2 or more = off)

(Examples)
To turn terminal Y (00) off: Y (00)= 0
To turn terminal Y (01) on: Y (01)= 1

Format 2 : With this format, the instruction outputs **<variable>** as word data to contact output terminals.

(Examples)
To turn terminal Y (00) on and turn terminals Y (01) to Y (05) off: Yw= 1
To turn terminals Y (00) to Y (04) off and turn terminal Y (05) on: Yw= U (00) (U (00) = 32)

Note : In the SJ700/L700/SJ700B Series, the setting of terminal active state (C031 to C036) is reflected in the polarity (on or off) of contact inputs Y (00) to Y (05) and Yw when the data is output to intelligent output terminals 11 to 15 and the relay output terminal. When you create a user program, consider the on and off states of actual intelligent output terminals.

In the WJ200 Series, the setting of terminal active state (C031, C032, and C036) is reflected in the polarity (on or off) of contact inputs Y (00), Y (01), Y (05) and Yw when the data is output to intelligent output terminals 11, 12 and the relay output terminal. When you create a user program, consider the on and off states of actual intelligent output terminals.

- Sample program

Sample 1 : Program to turn terminals Y (00) to Y (05) on sequentially while the output frequency is increased in 10-Hz steps.
(The inverter operation is the same as that programmed in sample 2.)

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
	SET-Freq=			6000			: Set the output frequency to 60 Hz.
	ACCEL=			3000			: Set the acceleration time to 30 seconds.
	DECEL=			3000			: Set the deceleration time to 30 seconds.
	Yw=	0					
	FW=	1					: Start forward rotation of the motor.
LOOP	if	FM	<	U(00)	then	LBL1	: When the output frequency is less than 10 Hz,
	Y(00)=	1					: turn Y (00) on.
	if	FM	<	U(01)	then	LBL1	: When the output frequency is less than 20 Hz,
	Y(01)=	1					: turn Y (01) on.
	if	FM	<	U(02)	then	LBL1	: When the output frequency is less than 30 Hz,
	Y(02)=	1					: turn Y (02) on.
	if	FM	<	U(03)	then	LBL1	: When the output frequency is less than 40 Hz,
	Y(03)=	1					: turn Y (03) on.
	if	FM	<	U(04)	then	LBL1	: When the output frequency is less than 50 Hz,
	Y(04)=	1					: turn Y (04) on.
	if	FM	<	U(05)	then	LBL2	: When the output frequency is less than 60 Hz,
	Y(05)=	1					: turn Y (05) on.
LBL1	goto	LOOP					
LBL2	FW=	0					
	Wait	RUN	=	0			: Decelerate and stop the motor.
	end						

(Data area [Data Window])

U (00) = 1000
U (01) = 2000
U (02) = 3000
U (03) = 4000
U (04) = 5000
U (05) = 6000

Sample 2 : Program to output codes sequentially to terminal Yw while the output frequency is increased in 10-Hz steps.
(The inverter operation is the same as that programmed in sample 1.)

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
	SET-Freq=			6000			: Set the output frequency to 60 Hz.
	ACCEL=			3000			: Set the acceleration time to 30 seconds.
	DECEL=			3000			: Set the deceleration time to 30 seconds.
	Yw=	0					
	FW=	1					: Start forward rotation of the motor.
LOOP	if	FM	<	U(00)	then	LBL1	: When the output frequency is less than 10 Hz
	if	FM	<	U(01)	then	LBL2	: When the output frequency is less than 20 Hz
	if	FM	<	U(02)	then	LBL3	: When the output frequency is less than 30 Hz
	if	FM	<	U(03)	then	LBL4	: When the output frequency is less than 40 Hz
	if	FM	<	U(04)	then	LBL5	: When the output frequency is less than 50 Hz
	if	FM	<	U(05)	then	LBL6	: When the output frequency is less than 60 Hz
	if	FM	=	U(05)	then	LBL8	: When the output frequency is 60 Hz
LBL1	Yw=	1					: Output "1" to Yw.
	goto	LBL7					
LBL2	Yw=	2					: Output "2" to Yw.
	goto	LBL7					
LBL3	Yw=	3					: Output "3" to Yw.
	goto	LBL7					
LBL4	Yw=	4					: Output "4" to Yw.
	goto	LBL7					
LBL5	Yw=	5					: Output "5" to Yw.
	goto	LBL7					
LBL6	Yw=	6					: Output "6" to Yw.
LBL7	goto	LOOP					
LBL2	FW=	0					: Decelerate and stop the motor.
	Wait	RUN	=	0			
	end						

(Data area [Data Window])

U (00) = 1000
U (01) = 2000
U (02) = 3000
U (03) = 4000
U (04) = 5000
U (05) = 6000

UB ( ) or UBw (internal user contact control)	Instruction to access internal user contacts

	Variable name	Range of values	Default	Unit	Data size	Attribute
UB (00) to UB (07)	Internal user contact (bit access)	0: OFF 1: ON	0	-	Unsigned 1-word data	Readable and writable
UBw	Internal user contact (word access)	0 to 255	0	-	Unsigned 1-word data	Readable and writable

- Format

Format		Description
1	**<variable>** = UB (ii) (ii = 00 to 07)	This instruction assigns the ii'th bit of internal user contact data to **<variable>**.
2	UB (ii) = **<variable>** or **<constant>** (ii = 00 to 07)	This instruction assigns **<variable>** or **<constant>** to the ii'th bit of internal user contact data.
3	**<variable>** = UBw	This instruction assigns internal user contact data as word data to **<variable>**.
4	UBw = **<variable>** or **<constant>**	This instruction assigns **<variable>** or **<constant>** as word data to internal user contact data.

- Explanation

Use this instruction to control the internal contacts that the user can use for general purposes. The inverter has eight general-purpose contacts that are writable and readable by bit access (UB (00) to UB (07)) or word access (UBw). Details of the formats are explained below.

Format 1 : With this format, the instruction reads the status of the ii'th bit of internal user contact data into **<variable>**.
(0 = off, 1 = on)

Format 2 : With this format, the instruction writes **<variable>** or **<constant>** to the ii'th bit of internal user contact data.
(0 = off, 1 = on, 2 or more = off)

Format 3 : With this format, the instruction reads internal user contact data as word data into **<variable>**.

Format 4 : With this format, the instruction writes **<variable>** or **<constant>** as word data to internal user contact data.

- Sample program

Sample 1 : Statement to read internal user contact status as bit data (format 1)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	U(00)=	UB(00)					: Assign bit data of UB (00) to U (00).
	:						

Sample 2 : Statement to turn an internal user contact on (format 2)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	UB(00)=	1					: Assign "1" (bit-on status) to UB (00).
	:						

Sample 3 : Statement to read internal user contact status as word data (format 3)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	U(00)=	UBw					: Assign word data of UBw to U (00).
	:						

Sample 4 : Statement to change internal user contact status in units of words (format 4)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	UBw(00)=	U(00)					: Write word data of U (00) to UBw.
	:						

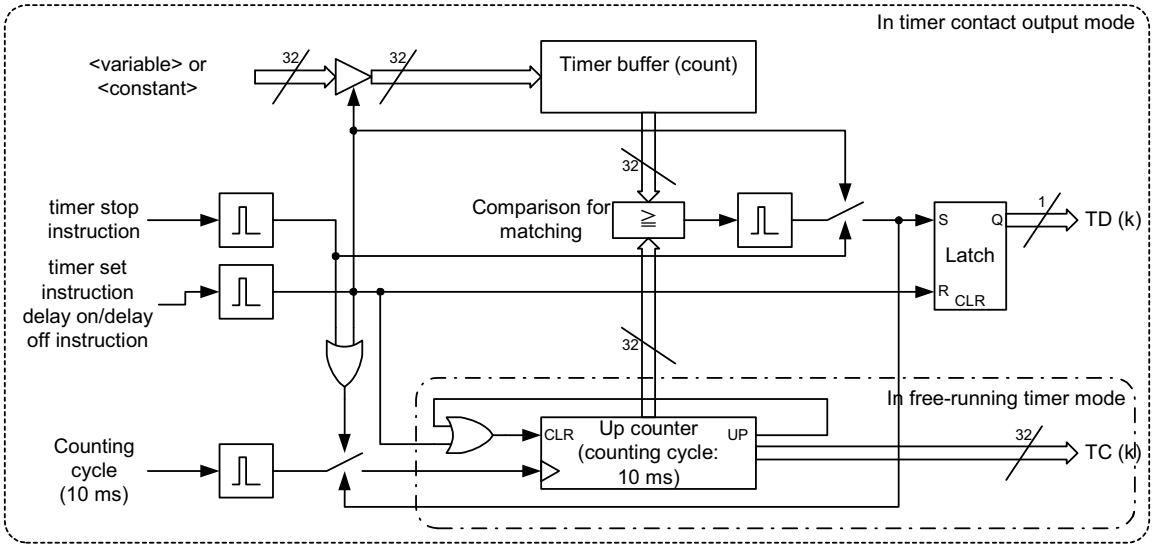
2.5 Timer Control Instructions

The easy sequence function of the inverter has a timer function that can be used in the following two modes:

(1) Free-running timer mode

(2) Timer contact output mode (timer-start, timer-stop, and delay operations)

The timer function uses eight timer counter circuits that are configured as shown in the figure below.



TC (k): Timer counter variable (up counter)

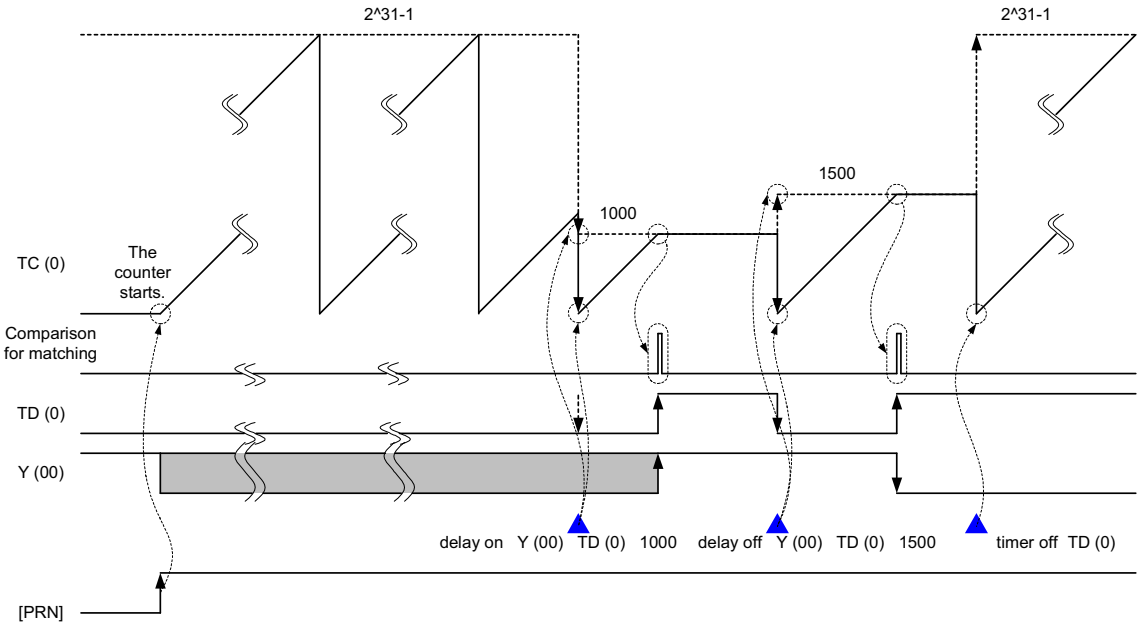
TD (k): Timer contact (one-shot timer)

Block diagram for timer function

The timer counter is a 31-bit up counter that runs in a 10-ms cycle (1 count per 10 ms), and operates as a free-running timer when the execution of a easy sequence program is started.

When the timer set, delay on, or delay off instruction is executed, the timer counter is cleared and restarted. While the timer counter is operating, its count is compared with the count specified by a variable or constant to determine whether they match. When the counts match each other, the timer counter stops counting.

When the timer off instruction is executed, the timer counter is cleared and restarted. Subsequently, the timer counter operates as a free-running timer.



Example of timer function operation

timer set (timer-start instruction)	Instruction to set and start the timer counter

- Format

Format	Description
timer set TD (k) **<variable>** or **<constant>**	This instruction sets **<variable>** or **<constant>** in the k'th timer and starts the timer counter.

- Explanation

(1) The timer set instruction sets **<variable>** or **<constant>** in the k'th timer buffer, clears the timer counter (up counter) "TC (k)" to zero, and then initiates counting by the timer counter. Then, the value of timer contact variable "TD (k)" is "0" (off).

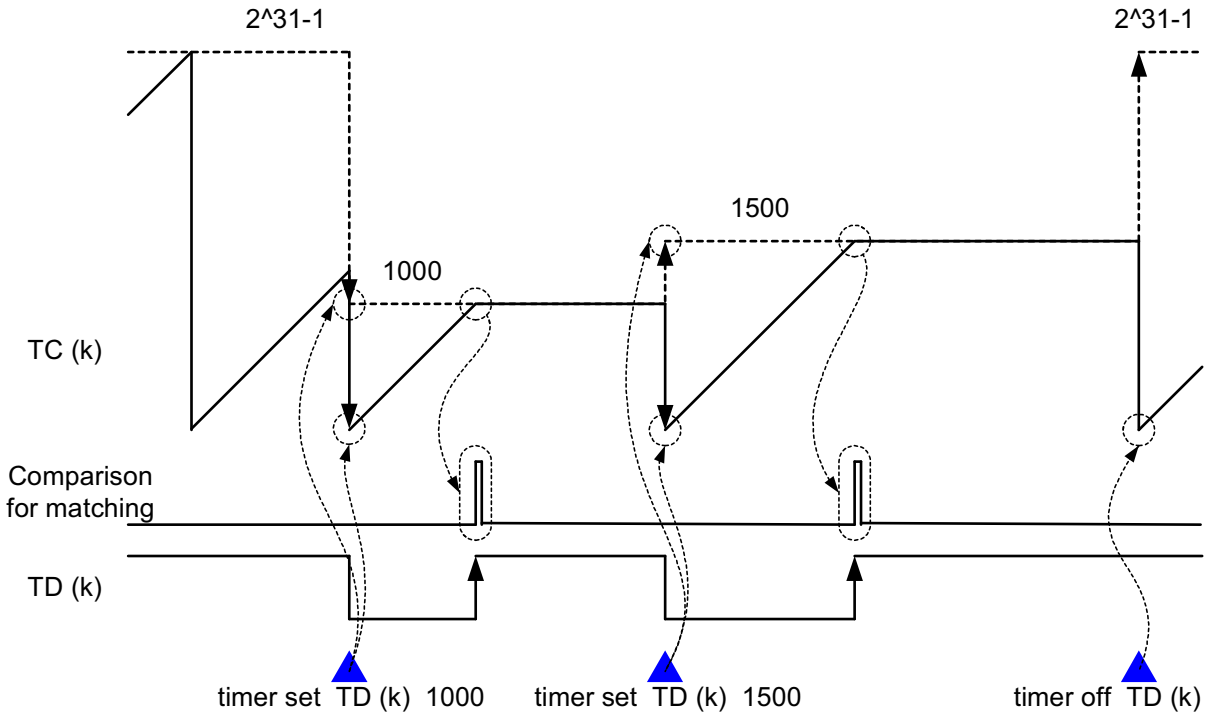
(2) Subsequently, the instructions described after the timer set instruction are executed.

(3) When the timer counter "TC (k)" reaches the specified count, the value of timer contact variable "TD (k)" changes to "1" (on) (only once). Then, the timer counter "TC (k)" stops counting.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	FW=	1					
	timer set	TD(0)	5.00				: Start the 5-second timer counter.
	wait	TD(0)	=	1			: Wait until "1" (on) is set in TD (0).
	Y(00)=	1					
	:						

- 動作例 (タイミングチャート)



Timing chart for operation using the timer set instruction

timer off (timer-stop instruction)	**Instruction to stop the timer**

- Format

Format	Description
timer off TD (k)	This instruction clears the k'th timer and operates it as a free-running timer.

- Explanation

This instruction clears the k'th timer counter (up counter) “TC (k)” to zero, and starts the timer counter in free-running timer mode. Then, the value of timer contact variable “TD (k)” is not changed. The timer counter “TC (k)” is switched from timer contact output mode to free-running timer mode.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	wait	X(00)	=	1			: Wait until terminal X (00) is turned on.
	Y(00)=	1					: Turn terminal Y (00) on.
	delay on	Y(01)	TD(1)	1000			: Turn terminal Y (01) on with a delay.
	timer set	TD(0)	15.00				: Start the 15-second timer counter.
	if	X(01)	=	0	then	LBL1	: Wait when terminal X (01) is off.
	timer off	TD(1)					: Clear timer counter TC (1).
LBL1	wait	TD(0)	=	1			: Wait until the 15-second timer counter ends counting.
	Y(02)=	1					: Turn terminal Y (02) on when the process ends.
	:						

delay on or delay off (delay operation instruction)	**Instruction to turn a variable on or off with a delay**

- Format

Format	Description
delay on **<variable 1>** TD (k) **<variable 2>** or **<constant>**	This instruction sets the count of the k'th timer in **<variable 2>** or **<constant>** and starts the timer counter. When timer output “TD (k)” is turned on, **<variable 1>** is turned on.
delay off **<variable 1>** TD (k) **<variable 2>** or **<constant>**	This instruction sets the count of the k'th timer in **<variable 2>** or **<constant>** and starts the timer counter. When timer output “TD (k)” is turned on, **<variable 1>** is turned off.

- Explanation

(1) The delay on (or delay off) instruction sets **<variable 2>** or **<constant>** in the k'th timer buffer, clears the timer counter (up counter) “TC (k)” to zero, and then initiates counting by the timer counter. Then, the value of timer output variable “TD (k)” is “0” (off).

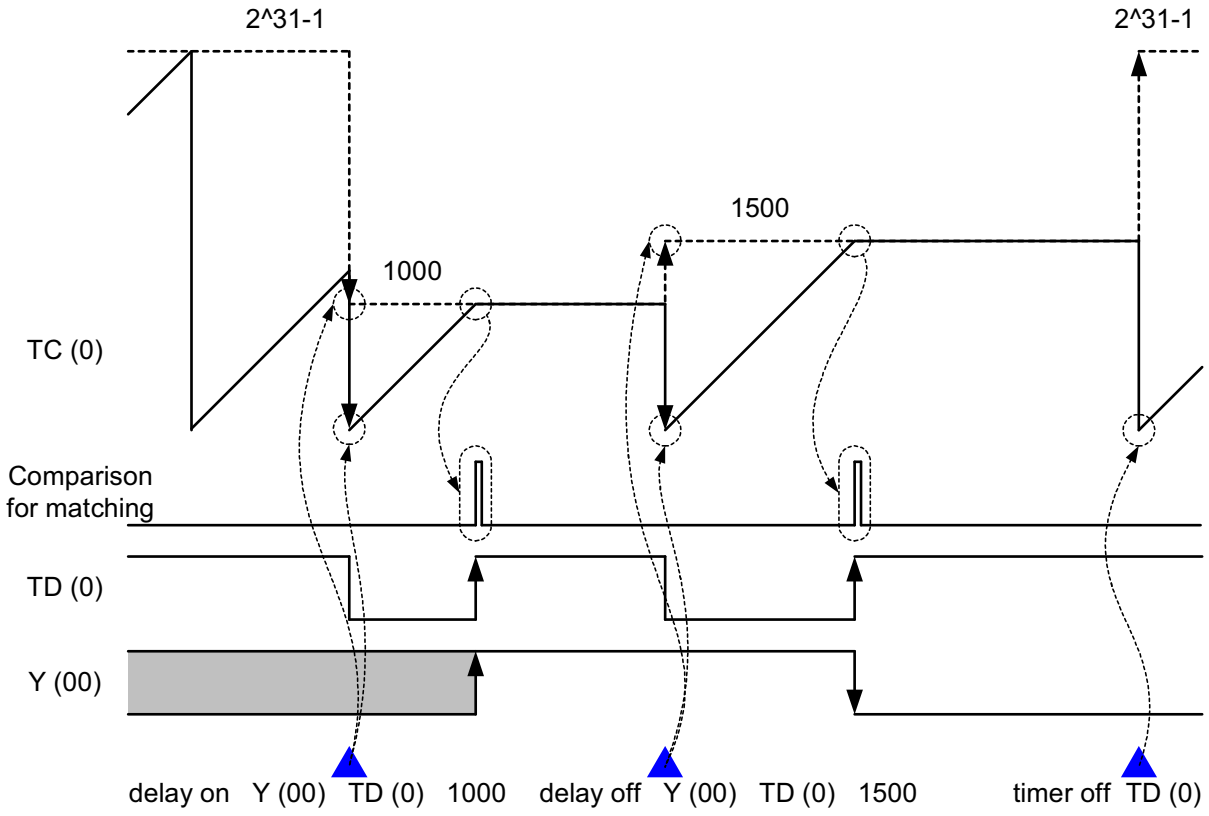
(2) Subsequently, the instructions described after the delay on (or delay off) instruction is executed.

(3) When the count of timer counter “TC (k)” matches the count preset in the timer buffer, the value of timer output variable “TD (k)” changes to “1” (on) (only once), and <variable 1> is turned on (or off). Then, the timer counter “TC (k)” stops counting.

- Sample program : Program to make the inverter alternately repeat forward rotation of the motor at 60 Hz and reverse rotation of the motor at 10 Hz

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	U(00)=			1000			: The output frequency for reverse rotation is 10 Hz.
	ACCEL=			1000			: Set the acceleration time to 10 seconds.
	DECEL=			1000			: Set the deceleration time to 10 seconds.
LOOP	SET-Freq=			6000			
	FW=	1					: Accelerate the forward rotation speed up to 60 Hz.
	wait	RUN	=	1			: Wait until the motor operates at 60 Hz.
	delay off	FW	TD(0)	15.00			: Start the 15-second timer counter.
	:						: Turn the FW terminal off after 15 seconds elapse.
	wait	RUN	=	0			: Wait until the motor stops.
	delay on	RV	TD(0)	1.00			: Start reverse rotation after 1 second elapses.
	SET-Freq=			1000			: Accelerate the reverse rotation speed up to 10 Hz.
	:						
	wait	FM	=	U(00)			: Wait until the output frequency reaches 10 Hz.
	RV=	0					: Decelerate and stop the motor.
	Wait	RUN	=	0			: Wait until the motor stops.
	goto	LOOP					
	:						

- Example of operation (timing chart)



Timing chart for operation using the delay on and delay off instructions

2.6 Inverter Control Instructions

Inverter operation command	Instruction to turn the input terminal function on or off

- Format

Format	Description
<input terminal function>= **<variable>** or **<constant>**	This instruction turns **<input terminal function>** of the inverter on or off according to the value of **<variable>** or **<constant>**.

- Explanation

This instruction turns the inverter input terminal specified by **<input terminal function>** on or off according to the value of **<variable>** or **<constant>**. When the value of **<variable>** or **<constant>** is 0, 1, or 2 or more, the input terminal specified by **<input terminal function>** is turned off, on, or off, respectively.

The function and operation of the specified input terminal are the same as those that can be specified by the terminal functions (SJ700/L700/SJ700B : C001 to C008 / WJ200 : C001 to C007) on the inverter. For details, refer to the Inverter Instruction Manual.

- Sample program: Program to make the inverter alternately repeat forward acceleration and deceleration, and reverse acceleration and deceleration of the motor at 60 Hz

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	SET-Freq=			6000			
LOOP	FW=	1					: Turn the FW terminal on.
	wait	X(01)	=	1			: Wait until X (01) is turned on.
	FW=	0					: Turn the FW terminal off.
	wait	RUN	=	0			: Wait until the motor stops.
	RV=	1					: Turn the RV terminal on.
	wait	X(02)	=	1			: Wait until X (02) is turned on.
	RV=	0					: Turn the RV terminal off.
	wait	RUN	=	0			: Wait until the motor stops.
	goto	LOOP					
	:						

Inverter operation monitoring instruction	Instruction to monitor the output terminal function

- Format

Format	Description
<variable 1> = **<output terminal function>**	This instruction fetches the on / off status of **<output terminal function>** of the inverter and stores it in **<variable 1>**.

- Explanation

This instruction fetches the on / off status of the inverter output terminal specified by **<output terminal function>** and stores it in **<variable 1>**. When the specified output terminal is off, the value of **<variable 1>** is "0"; when it is on, the value of **<variable 1>** is "1".

The function and operation of the specified output terminal are the same as those that can be specified by the terminal function (SJ700 /L700/SJ700B: C021 to C026 / WJ200 : C021, C022, and C026) on the inverter. For details, refer to the Inverter Instruction Manual.

- Sample program

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	SET-Freq=			6000			: Set the output frequency to 60 Hz.
	ACCEL=			1000			: Set the acceleration time to 10 seconds.
	DECEL=			1000			: Set the deceleration time to 10 seconds.
	FW=	1					
	wait	X(01)	=	1			: Wait until X (01) is turned on.
	ACCEL=			3000			: Increase the acceleration time to 30 seconds.
	wait	FA1	=	1			: Wait until the output frequency reaches the set frequency.
LOOP	if	X(02)	<>	1	then	LBL1	: Wait until X (02) is turned on.
	SET-Freq=			500			: Reduce the set frequency to 5 Hz.
LBL1	UB(00)=	ZS					
	if	UB(00)	<>	1	then	LOOP	
	DECEL=			3000			: Increase the deceleration time to 30 seconds when ZS is on.
	wait	10.00					: Operate the motor at 5 Hz for 10 seconds.
	FW=	0					: Decelerate and stop the motor.
	:						

(Parameter)

C063 = 5.0Hz

User Monitor	**Operator display variable**

Umon (00) to Umon (02)	Variable name	Range of values	Default	Unit	Data size	Attribute
	User monitor 0 to 2	$-2^{31} - 2^{31}-1$	0	-	Signed 2-word data	Readable and writable

- Format

Format	Description
Umon (ii) = **<variable>**	Displays **<variable>** on user monitor (ii)
Umon (ii) = **<variable1> <operator> <variable>**.	Displays the result of operation with <variable1> and <variable2> on user monitor (ii)
<variable> = Umon (ii)	Value of user monitor (ii) is read out to **<variable>**

- Explanation

This instruction displays arbitrary data to the digital operator of the inverter.
Each display variable and the correspondence of the display code are as follows.

Umon (00) : User monitor 0 (d025)
Umon (01) : User monitor 1 (d026)
Umon (02) : User monitor 2 (d027)

- Sample program : Program to display the summation of U (01) and U (02) on user monitor 2 (d027)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5
	:					
	Umon(02)=	U(01)	+	U(02)		
	:					

User Trip	**User trip issue command**

- Format

Format	Description
trip **<variable>**	Makes the inverter trip

- Explanation

This instruction makes inverter trip. Range of <variable> is 0 to 9.

Note : When the user trip occurs without the description of on trip go to instruction, the program stops immediately after the occurrence of inverter trip.

- Sample program : Program to issue the user trip 2 (E52) when the summation of variable 1 and variable2 exceeds 20

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5
	:					
	U(00)=	U(01)	+	U(02)		
	if	U(00)	>	20		
	then					
	trip	2				
	else					
	:					

stop statement	Instruction to stop motor operation by the inverter

- Format

Format	Description
stop	This instruction makes the inverter decelerate and stop the motor.

- Explanation

This instruction makes the inverter decelerate and stop the motor and reset the inverter when it is in trip situation.
When the FW terminal is on (FW = 1) or the RV terminal is on (RV = 1), this instruction turns off the FW terminal (FW = 0) or RV terminal (RV = 0).

- Sample program : Program to make the inverter operate the motor for forward or reverse rotation at a constant speed for 10 seconds

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	if	X(00)	<>	1	then	LBL1	
	FW=	1					: When X (00) is on, operate the motor for forward rotation.
	goto	LBL2					
LBL1	RV=	1					: When X (00) is on, operate the motor for reverse rotation.
LBL2	wait	FA1	=	1			: Wait until the motor rotates at a constant speed.
	wait	10.00					: Operate the motor at 5 Hz for 10 seconds.
	stop						: Decelerate and stop the motor. (FW = 0 or RV = 0)
	:						

chg param statement	Instruction to change a parameter setting

- Format

Format	Description
chg param **<display code>** **<variable>** or **<constant>**	This instruction changes the setting of the inverter parameter specified by **<display code>** to **<variable>** or **<constant>**.

- Explanation

<display code> specifies the parameter number of the inverter parameter of which the setting is to be changed. The range of parameter settings depends on the standard inverter specifications. For the inverter parameters and ranges of their settings, refer to the Inverter Instruction Manual.
Specify an integer as the desired new setting of the parameter in **<variable>** or **<constant>**. To specify a numerical value other than 0 to 127, preset the value in a variable and specify the variable as **<variable>**. The changed parameter setting is reflected in the inverter in a 40-ms cycle, which conforms to the standard inverter specifications. If, however, you directly access the inverter's EEPROM, the change is reflected in the inverter in the same cycle as that of instruction execution.

Note 1 : You cannot specify any of parameters "U001" to "U012" in <display code>.
Note 2 : When the parameter is changed, it is not memorized in EEPROM. It returns to an initial value by the power shutdown.

- Sample program : Program to change the overload restriction level according to output frequency
(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	U(00)=			2000			: Set 200% in variable "U (00)."
	U(01)=			1500			: Set 150% in variable "U (01)."
	U(02)=			1000			: Set 100% in variable "U (02)."
	U(03)=			1000			: Set 10 Hz in variable "U (03)."
	chg param	b022	U(00)				: Change the setting of "b022" to 200.0%
	FW=	1					
	wait	FM	>=	U(03)			: Wait until the output frequency reaches 10 Hz.
	chg param	b022	U(01)				: Change the setting of "b022" to 150.0%.
	wait	FA1	=	1			: Wait until acceleration ends.
	chg param	b022	U(02)				: Change the setting of "b022" to 100.0%.
	:						

(Parameter)
b031 = 10 (can be updated during operation)

mon param statement — Instruction to read a parameter

- Format

Format	Description
mon param **<display code> <variable>**	This instruction assigns the content of the inverter parameter specified by **<display code>** to **<variable>**.

- Explanation

This instruction reads the content of the inverter parameter specified by **<display code>**, and assigns the read content to **<variable>**. The range of parameter settings depends on the standard inverter specifications. For the inverter parameters and ranges of their settings, refer to the Inverter Instruction Manual.

- Sample program : Program to check whether the inputs of frequency command and acceleration/deceleration time are assigned to the easy sequence function

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
	Yw=	0					
	mon param	A001	U(00)				: Assign the content of A001 to U (00).
	mon param	P031	U(01)				: Assign the content of P031 to U (01).
	if	U(00)	<>	7	then	SKIP	: When A001 is not "07"
	if	U(01)	<>	3	then	SKIP	: When P031 is not "03"
	Y(00)=	1					: Turn Y (00) on.
	goto	LBL1					
SKIP	Y(01)=	1					: Turn Y (01) on when the process ends.
LBL1	end						

eepwrt	Instruction to issue a demand to write the parameter to EEPROM

- Format

Format	Description
eepwrt	This instruction stores a data of only one parameter changed by chg param command that is issued immediately after the execution of eeprwrt command to EEPROM.

- Explanation

This instruction is a function in the WJ200 Series.

This instruction stores the content of the parameter of the inverter in EEPROM. It combines chg param command and realizes a function. eepwrt issues a demand to write the data to EEPROM. Afterwards, a parameter changed by chg prm is stored to EEPROM. The range of parameter settings depends on the standard inverter specifications. For the inverter parameters and ranges of their settings, refer to the WJ200 Series Inverter Instruction Manual.

Note 1 : After chg param is executed, the demand to write to EEPROM is cleared. It is necessary to reissue eepwrt instruction to store the content of the parameter newly changed by chg param to EEPROM.

Note 2 : A continuous writing to EEPROM by the eepwrt instruction is a prohibition. Please create the user program so that the execution interval of the eepwrt instruction may become 30ms or more when writing the parameter in EEPROM two times or more.

- Sample program : Program to configure multispeed frequency 1 and 2 with general-purpose analog inputs

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
LOOP	UL(00)=			XA(0)			: Fetch analog input "O" data.
	UL(00)=	UL(00)	*	50			: Convert the scale.
	UL(00)=	UL(00)	/	100			
	UL(00)=	UL(00)	+	1000			
	UL(01)=			XA(1)			: Fetch analog input "OI" data.
	UL(01)=	UL(01)	*	50			: Convert the scale.
	UL(01)=	UL(01)	/	100			
	UL(01)=	UL(01)	+	1000			
	eepwrt						: Demand to write to EEPROM.
	chg param	A021	UL(00)				
	wait	0.03					: Wait 30 ms.
	eepwrt						: Demand to write to EEPROM.
	chg param	A022	UL(01)				
	wait	0.03					: Wait 30 ms.
	goto	LOOP					
	end						

rtcset on, rtcset off	Instruction to start / stop the update of clock data

- Format

Format	Description
rtcset on **<variable>**	The clock data is substituted for six bytes that make **<variable>** a head. Moreover, the clock data is regularly updated.
rtcset off **<variable>**	The clock data is substituted for six bytes that make **<variable>** a head. Moreover, the update of the clock data is stopped.

- Explanation

This instruction is a function in the WJ200 Series.

The clock data of six bytes sent from WOP is stored in six bytes that make **<variable>** a head. Clock data is 1 byte for each in "year", "month", "date", "day of the week", "hour" and "minute".

In the rtcset on instruction, the clock data is regularly updated after the clock data is substituted.

In the rtcset off instruction, the update of the clock data by rtcset on instruction is stopped after the clock data is substituted.

The data stored in “rtcset on / off U (00)” are as follows.

Example) Monday, April 19, 2010 11:15

U (00) = 1004h = 4100 (The last 2 digits at the year: 00h to 99h / Month: 01h to 12h)
U (01) = 1901h = 6401 (Date: 01h to 31h / Day of the week: Sun.(00h) to Sat.(06h))
U (02) = 1115h = 4373 (Hour: 00h to 23h / Minute: 00h to 59h)

Moreover, it is as follows when a variable is set to UL (00).

UL (00) = 10041901h = 268704001 （The last 2 digits at the year / Month / Date / Day of the week）
UL (01) = 11150000h = 286588928 （hour / Minute / 0000h）

Note : It is necessary to connect WOP with the inverter to acquire the clock data. When WOP is not connected, all clock data is set to 0.

- Sample program : Program to make the inverter operate the motor for forward rotation for 1 hour

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
	rtcset	off	U(02)				: Clock data is stored in U (02) to U (04).
	U(04)=	U(04)	+	U(00)			: Add to the present time for 1 hour.
	ifs	U(04)	>=	U(01)			: When it will be data more than in 24:00, it pulls for 24 hours.
	then						
	U(04)=	U(04)	-	U(01)			
	end if						
	rtcset	on	U(05)				: Clock data is stored in U (05) to U (07).
	FW=	1					: Start forward rotation of the motor.
LOOP	if	U(07)	<>	U(04)	then	LOOP	
	rtcset	off	U(05)				: Stop the update of the clock data.
	FW=	0					: Decelerate and stop the motor.
	end						

(Data area [Data Window])

U(00) = &H0100 : 1 hour
U(01) = &H2400 : 24:00

2.7 Other Reserved Variables

	Variable name	Range of values	Default	Unit	Data size	Attribute
U (00) to U (31)	User-defined variable	0 to 65535	Data stored in P100 to P131	-	Unsigned 1-word data	Readable and writable

- Explanation

User-designed variables are the general-purpose functions that can be used as unsigned 1-word variables regardless of format. The data written from a sequence program to the user-defined variables is not stored in the inverter's EEPROM. The variables will restore the initial settings when the inverter power is turned off. The user-defined variables correspond to inverter parameters “P100” to “P131”. You can also change the settings of user-defined variables from the digital operator. The changes made from the digital operator will be stored in EEPROM.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5
	:					
	U(00)=	U(00)	+	U(01)		
	U(02)=	U(00)	*	U(02)		
	U(03)=	U(00)	mod	U(01)		

	Variable name	Range of values	Default	Unit	Data size	Attribute
UL (00) to UL (07)	Internal user variable	-2^{31} to $2^{31}-1$	0	-	Signed 2-word data	Readable and writable

- Explanation

Internal user variables are the general-purpose functions that can be used as unsigned 2-word variables, for example, to temporarily store arithmetic operation results.

Note : If an arithmetic operation causes data overflow, an execution error (E45) will result.

- Sample program

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	UL(00)=			Tmon			: Acquire the output torque data.
	if	UL(00)	>=	0	then	SKIP	: When the output torque is a positive value
	UL(01)=			-1			
	UL(00)=	U(01)	*	UL(00)			: When the output torque is a negative value (x -1)
SKIP	U(05)=			UL(00)			
	U(05)=	U(05)	*	100			: Convert the scale.
	U(05)=	U(05)	/	300			
	YA(1)=			U(05)			: Output the data to general-purpose analog output.
	:						

SET-Freq	Variable name	Range of values	Default	Unit	Data size	Attribute
	Output frequency setting	0 to 40000	0	0.01 Hz	Unsigned 1-word data	Readable and writable

- Explanation

This variable can be used to read and write the frequency specified by the output frequency setting (F001) in the inverter. (See Note 1 and 2.) The setting of this variable corresponds to inverter parameter "F001". The data written to this variable is not stored in the inverter's EEPROM. This variable will restore the initial setting when the inverter power is turned off. When the inverter receives an operation command (FW = 1 or RV = 1), it accelerates the motor up to the frequency that was set last.

Note 1 : To reflect the frequency written in this variable as the set frequency, you must change the setting of frequency source setting (A001) to "07" (PRG).

Note 2 : This variable can be read regardless of the setting of "A001". The currently applied set frequency is read from this variable.

- Sample program : Program to alternately repeat forward rotation of the motor at 60 Hz and reverse rotation at 10 Hz
(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
LOOP	SET-Freq=			6000			: Set the output frequency to 60 Hz.
	FW=	1					: Start forward rotation of the motor.
	wait	FA1	=	1			: Wait until the output frequency reaches the set frequency.
	wait	10.00					: Operate the motor for 10 seconds.
	FW=	0					: Decelerate and stop the motor.
	wait	RUN	=	0			: Wait until the motor stops.
	SET-Freq=			1000			: Change the set frequency to 10 Hz.
	RV=	1					: Start reverse rotation of the motor.
	wait	FA1	=	1			: Wait until the output frequency reaches the set frequency.
	wait	10.00					: Operate the motor for 10 seconds.
	RV=	0					: Decelerate and stop the motor.
	wait	RUN	=	0			: Wait until the motor stops.
	goto	LOOP					
	end						

(Parameter)

A001 = 07

ACCEL	Variable name	Range of values	Default	Unit	Data size	Attribute
	Acceleration time setting	1 to 360000	Note 1	0.01 second	Unsigned 2-word data	Readable and writable

- Explanation

This variable can be used to read and write the motor acceleration time in the inverter. The acceleration time setting using this variable is enabled only when the setting of accel/decel time input selection (P031) is "03" (PRG). (The setting of this variable does not correspond to the setting of inverter parameter "F002".) The data written to this variable is not stored in the inverter's EEPROM. This variable will restore the initial setting when the inverter power is turned off.

Note 1 : Default (the inverter power is turned on) acceleration time follows the setting of inverter parameter "F002", "F202", or "F302". For details, refer to the Inverter Instruction Manual.

Note 2 : When a program writes a value to this variable, the value is reflected in the inverter in a 40-ms cycle, which conforms to the standard inverter specifications.

- Sample program : Program to change the acceleration time according to output frequency

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	SET-Freq=			6000			: Set the output frequency to 60 Hz.
	ACCEL=			1000			: Set the acceleration time to 10 seconds.
	FW=	1					: Start forward rotation of the motor.
LOOP	if	FW	<	U(00)	then	LBL1	: When the output frequency is less than 5 Hz, set the acceleration time to 10 seconds.
	if	FM	<	U(01)	then	LBL2	: When the output frequency is less than 10 Hz, change the acceleration time to 5 seconds.
	if	FM	<	U(02)	then	LBL3	: When the output frequency is less than 30 Hz, change the acceleration time to 1 second.
	if	FM	<	U(03)	then	LBL4	: When the output frequency is less than 50 Hz, change the acceleration time to 5 seconds.
	if	FM	<	U(04)	then	LBL5	: When the output frequency is less than 55 Hz, change the acceleration time to 10 seconds.
	if	FM	<	U(05)	then	LBL6	: When the output frequency is less than 60 Hz, change the acceleration time to 20 seconds.
	if	FM	>=	U(05)	then	LBL8	: When the output frequency reaches or exceeds 60 Hz, end acceleration.
	goto	LBL7					
LBL1	ACCEL=			1000			
	goto	LBL7					
LBL2	ACCEL=			500			
	goto	LBL7					
LBL3	ACCEL=			100			
	goto	LBL7					
LBL4	ACCEL=			500			
	goto	LBL7					
LBL5	ACCEL=			1000			
	goto	LBL7					
LBL6	ACCEL=			2000			
LBL7	goto	LOOP					
LBL8	Y(00)=	1					: Turn Y (00) on when acceleration ends.
	:						

(Data area [Data Window])

U (00) = 500 : Set the frequency of 5 Hz in variable "U (00)".
U (01) = 1000 : Set the frequency of 10 Hz in variable "U (01)".
U (02) = 3000 : Set the frequency of 30 Hz in variable "U (02)".
U (03) = 5000 : Set the frequency of 50 Hz in variable "U (03)".
U (04) = 5500 : Set the frequency of 55 Hz in variable "U (04)".
U (05) = 6000 : Set the frequency of 60 Hz in variable "U (05)".

(Parameters)

A001 = 07 P031 = 03

DECEL	Variable name	Range of values	Default	Unit	Data size	Attribute
	Deceleration time setting	1 to 360000	Note 1	0.01 second	Unsigned 2-word data	Readable and writable

- Explanation

This variable can be used to read and write the motor deceleration time in the inverter. The deceleration time setting using this variable is enabled only when the setting of accel/decel time input selection (P031) is “03” (PRG). (The setting of this variable does not correspond to the setting of inverter parameter “F003”.) The data written to this variable is not stored in the inverter's EEPROM. This variable will restore the initial setting when the inverter power is turned off.

Note 1 : Default (the inverter power is turned on) deceleration time follows the deceleration (1) time setting “F003”, “F203”, or “F303”. For details, refer to the Inverter Instruction Manual.

Note 2 : When a program writes a value to this variable, the value is reflected in the inverter in a 40-ms cycle, which conforms to the standard inverter specifications.

- Sample program : Program to change the deceleration time according to output frequency

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	DECEL=			3000			: Set the deceleration time to 30 seconds.
	FW=	0					: Start deceleration of the motor.
	wait	FM	<	U(04)			: Wait until the output frequency falls below 55 Hz.
	DECEL=			1000			: Change the deceleration time to 10 seconds.
	wait	FM	<	U(03)			: Wait until the output frequency falls below 50 Hz.
	DECEL=			500			: Change the deceleration time to 5 seconds.
	wait	FM	<	U(02)			: Wait until the output frequency falls below 30 Hz.
	DECEL=			1000			: Change the deceleration time to 10 seconds.
	wait	FM	<	U(01)			: Wait until the output frequency falls below 10 Hz.
	DECEL=			3000			: Change the deceleration time to 30 seconds.
	wait	FM	<	U(00)			: Wait until the output frequency falls below 5 Hz.
	DECEL=			6000			: Change the deceleration time to 60 seconds.
	wait	RUN	=	0			: Wait until the motor stops.
	Y(01)=	1					: Turn Y (00) on when acceleration ends.
	:						

(Data area [Data Window])

U (00) = 500 : Set the frequency of 5 Hz in variable “U (00)”.
U (01) = 1000 : Set the frequency of 10 Hz in variable “U (01)”.
U (02) = 3000 : Set the frequency of 30 Hz in variable “U (02)”.
U (03) = 5000 : Set the frequency of 50 Hz in variable “U (03)”.
U (04) = 5500 : Set the frequency of 55 Hz in variable “U (04)”.
U (05) = 6000 : Set the frequency of 60 Hz in variable “U (05)”.

(Parameters)

P031 = 03

	Variable name	Range of values	Default	Unit	Data size	Attribute
XA (0)	General-purpose analog input (O terminal)	0 to 10000	0	0.01 %	Unsigned 1-word data	Readable
XA (1)	General-purpose analog input (OI terminal)	0 to 10000				
XA (2)	General-purpose analog input (O2 terminal)	-10000 to 10000				

- Explanation

These variables can be used to monitor the data input to the O, OI, and O2 terminals (among the analog input terminals of the inverter) in a data range from -100.00 to +100.00. The analog inputs monitored with these variables correspond to the data set by the [O]-[L], [OI]-[L], and [O2]-[L] input functions (A011 to A015, A101 to A105, and A111 to A114).
The WJ200 Series doesn't correspond to XA (2). For details, refer to the Inverter Instruction Manual.

- Sample program : Program to configure output frequencies in steps of 10 Hz with general-purpose analog inputs
(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	FW=	1					
LOOP	UL(00)=			XA(0)			: Fetch analog input data.
	UL(00)=	UL(00)	*	60			: Convert the scale.
	UL(00)=	UL(00)	/	100			
	if	U(00)	<	U(00)	then	LBL1	: When data is less than 16.67%, set the output frequency to 10 Hz.
	if	U(00)	<	U(01)	then	LBL2	: When data is less than 33.33%, set the output frequency to 20 Hz.
	if	U(00)	<	U(02)	then	LBL3	: When data is less than 50.00%, set the output frequency to 30 Hz.
	if	U(00)	<	U(03)	then	LBL4	: When data is less than 66.67%, set the output frequency to 40 Hz.
	if	U(00)	<	U(04)	then	LBL5	: When data is less than 83.33%, set the output frequency to 50 Hz.
	if	U(00)	<	U(05)	then	LBL6	: When data is less than 100.00%, set the output frequency to 60 Hz.
	goto	LBL7					
LBL1	SET-Freq=			1000			
	goto	LBL7					
LBL2	SET-Freq=			2000			
	goto	LBL7					
LBL3	SET-Freq=			3000			
	goto	LBL7					
LBL4	SET-Freq=			4000			
	goto	LBL7					
LBL5	SET-Freq=			5000			
	goto	LBL7					
LBL6	SET-Freq=			6000			
LBL7	goto	LOOP					
	:						

(Data area [Data Window])

U (00) = 1000
U (01) = 2000
U (02) = 3000
U (03) = 4000
U (04) = 5000
U (05) = 6000

	Variable name	Range of values	Default	Unit	Data size	Attribute
YA (0)	General-purpose analog output (SJ700 : FM terminal) (WJ200 : EO terminal)	0 to 10000	0	0.01 %	Unsigned 1-word data	Readable and writable
YA (1)	General-purpose analog output (AM terminal)					
YA (2)	General-purpose analog output (AMI terminal)					

- Explanation

These variables can be used to monitor the data output to the FM or EO, AM, and AMI terminals (analog output terminals of the inverter) in a data range from 0% to 100.00%. To obtain the analog outputs, you must assign general-purpose output functions to the FM or EO, AM, and AMI terminals with inverter parameters “C027”, “C028”, and “C029”.

The WJ200 Series doesn't correspond to YA (2). For details, refer to the Inverter Instruction Manual.

YA (0) : FM or EO output terminal (C027 = 12 [YA0])
YA (1) : AM output terminal (C028 = 13 [YA1])
YA (2) : AMI output terminal (C029 = 14 [YA2])

- Sample program : Program to output inverter output frequency data to a general-purpose analog output as data that is one-half of the full-scale data

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	FW=	1					
LOOP	UL(00)=			XA(0)			: Fetch analog input data.
	UL(00)=	UL(00)	*	60			: Convert the scale.
	UL(00)=	UL(00)	/	100			
	SET-Freq=			UL(00)			: Set the output frequency.
	UL(01)=	FM	*	100			
	UL(01)=	UL(01)	/	60			: Convert the scale.
	YA(1)=	UL(01)	/	2			: Output data that is one-half of the full-scale data.
	goto	LOOP					
	:						

TC (0) to TC (7)	Variable name	Range of values	Default	Unit	Data size	Attribute
	Timer counters	0 to 2^{31}-1	0	10 ms	Unsigned 2-word data	Readable and writable

- Explanation

These variables can be used to monitor the counts of the timer counters. The timer counters "TC (0)" to "TC (7)" usually operate as 31-bit free-running timer counters that start simultaneously with user program startup and are incremented in a 10-ms cycle.

When a timer-start instruction (timer set) or delay operation instruction (delay on or delay off) is executed, the timer counter corresponding to the instruction operates as the counter for output to a specified timer contact. In this case, the counter is cleared to zero when the instruction is executed, starts counting, and then stops counting upon reaching the specified count.

When a timer-stop instruction (timer off) is executed, the timer counter corresponding to the instruction is cleared to zero and operates as a 31-bit free-running timer counter that is incremented in a 10-ms cycle.

- Sample program : Program to accelerate the motor step-by-step by using a free-running timer

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	ACCEL=			1000			: Set the acceleration time to 10 seconds.
	DECEL=			1000			: Set the deceleration time to 10 seconds.
	SET-Freq=			0			: Set the output frequency to 0 Hz.
LOOP	FW=	1					: Start forward rotation of the motor.
	if	TC(5)	<	U(00)	then	LBL1	: When TC (5) indicates less than 10 seconds
	if	TC(5)	<	U(01)	then	LBL2	: When TC (5) indicates less than 20 seconds
	if	TC(5)	<	U(02)	then	LBL3	: When TC (5) indicates less than 30 seconds
	if	TC(5)	>=	U(03)	then	LBL4	: When TC (5) indicates 40 seconds or more
LBL1	SET-Freq=			1000			: When TC (5) is less than 10 seconds, increase the output frequency to 10 Hz.
	goto	LBL5					
LBL2	SET-Freq=			3000			: When TC (5) is less than 20 seconds, increase the output frequency to 30 Hz.
	goto	LBL5					
LBL3	SET-Freq=			6000			: When TC (5) is less than 30 seconds, increase the output frequency to 60 Hz.
	goto	LBL5					
LBL4	FW=	0					: When TC (5) is 40 seconds or more, decelerate and stop the motor.
	wait	RUN	=	0			: Wait until the motor stops.
	SET-Freq=			0			: Set the output frequency to 0 Hz.
	TC(5)=			0			: Clear TC (5) to zero.
LBL5	goto	LOOP					
	:						

(Data area [Data Window])

U (00) = 1000 : Set 10 seconds in variable "U (00)".
U (01) = 2000 : Set 20 seconds in variable "U (01)".
U (02) = 3000 : Set 30 seconds in variable "U (02)".
U (03) = 4000 : Set 40 seconds in variable "U (03)".

	Variable name	Range of values	Default	Unit	Data size	Attribute
TD (0) to TD (7)	Timer contact output (bit access)	0: Off 1: On	0	-	Unsigned 1-word data	Readable
TDw	Timer contact output (word access)	0 to 255	0	-	Unsigned 1-word data	Readable

- Explanation

The data in timer contact output variables “TD (0)” to “TD (7)” is changed only when these variables are specified in the timer-start instruction (timer set) or delay operation instruction (delay on or delay off). A timer contact output variable is set to “0” (off) when the counter corresponding to the contact output is cleared to zero; the variable is set to “1” (on) when the counter stops counting.

While a timer counter variable “TC (k)” is being used for a free-running timer counter, timer contact output variable “TD (k)” corresponding to the timer counter variable retains its status.

- Sample program : Program to accelerate the motor step-by-step by using a timer contact

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	TDw=			0			
	ACCEL=			1000			: Set the acceleration time to 10 seconds.
	DECEL=			1000			: Set the deceleration time to 10 seconds.
	SET-Freq=			0			: Set the output frequency to 0 Hz.
LOOP	FW=	1					: Start forward rotation of the motor.
	timer set	TD(5)	10.00				: Start the 10-second timer counter.
	SET-Freq=			1000			: Keep the output frequency at 10 Hz for 10 seconds.
	wait	TD(5)	=	1			: Wait until the timer contact is turned on.
	timer set	TD(5)	10.00				: Start the 10-second timer counter.
	SET-Freq=			3000			: Keep the output frequency at 30 Hz for 10 seconds.
	wait	TD(5)	=	1			: Wait until the timer contact is turned on.
	timer set	TD(5)	10.00				: Start the 10-second timer counter.
	SET-Freq=			6000			: Keep the output frequency at 60 Hz for 10 seconds.
	wait	TD(5)	=	1			: Wait until the timer contact is turned on.
	FW=	0					
	wait	RUN	=	0			: Wait until the motor stops.
	SET-Freq=			0			: Set the output frequency to 0 Hz.
	goto	LOOP					
	:						

2.8 Inverter Montor Variables

FM	Variable name	Range of values	Default	Unit	Data size	Attribute
	Output frequency monitoring	0 to 40000	-	0.01 Hz	Unsigned 1-word data	Readable

- Explanation

This variable can be used to monitor the inverter output frequency. The data monitored with this variable corresponds to the data monitored by the output frequency monitoring function (d001). This variable is read-only. For details, refer to the Inverter Instruction Manual.

- Sample program : Program to turn a contact output on when output frequency exceeds 50 Hz and turn the contact output off when output frequency falls below 10 Hz

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
LOOP	if	FM	<	U(00)	then	LBL1	: When the output frequency is less than 10 Hz
	if	FM	>	U(01)	then	LBL2	: When the output frequency is more than 50 Hz
	goto	LBL3					
LBL1	UB(02)=	0					: Turn UB (02) off.
	goto	LBL3					
LBL2	UB(02)=	1					: Turn UB (02) on.
LBL3	Y(02)=	UB(02)					: Set the data of UB (02) in Y (02).
	goto	LOOP					
	:						

(Data area [Data Window])

U(00) = 1000 : Set the frequency of 10 Hz in variable “U (00)”.

U(01) = 5000 : Set the frequency of 50 Hz in variable “U (01)”.

Iout	Variable name	Range of values	Default	Unit	Data size	Attribute
	Output current monitoring	0 to 9999	-	0.01 %	Unsigned 1-word data	Readable

- Explanation

This variable can be used to monitor the inverter output current. The data monitored with this variable corresponds to the data monitored by the output current monitoring function (d002). The monitored data indicates the ratio of present output current to rated current of the inverter. This variable is read-only. For details, refer to the Inverter Instruction Manual.

- Sample program : Program to accelerate the motor while increasing the acceleration time when output current is high

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
LOOP	U(10)=			Iout			: Fetch the output current data.
	if	U(10)	<=	U(05)	then	LBL1	; When output current is 50% or less, change the acceleration time to 1 second.
	if	U(10)	<=	U(04)	then	LBL2	: When output current is 80% or less, change the acceleration time to 2 seconds.
	if	U(10)	<=	U(03)	then	LBL3	: When output current is 100% or less, change the acceleration time to 5 seconds.
	if	U(10)	<=	U(02)	then	LBL4	: When output current is 150% or less, change the acceleration time to 10 seconds.
	if	U(10)	<=	U(01)	then	LBL5	: When output current is 180% or less, change the acceleration time to 20 seconds.
	if	U(10)	<=	U(00)	then	LBL6	: When output current is 200% or less, change the acceleration time to 50 seconds.
	if	U(10)	>	U(00)	then	LBL7	: When output current exceeds 200%, change the acceleration time to 100 seconds.
	goto	LBL8					
LBL1	ACCEL=			100			
	goto	LBL8					
LBL2	ACCEL=			200			
	goto	LBL8					
LBL3	ACCEL=			500			
	goto	LBL8					
LBL4	ACCEL=			1000			
	goto	LBL8					
LBL5	ACCEL=			2000			
	goto	LBL8					
LBL6	ACCEL=			5000			
	goto	LBL8					
LBL7	ACCEL=			10000			
LBL8	if	FA1	=	1	then	LBL9	
	goto	LOOP					
LBL9	Y(00)=	1					: Turn Y (00) on when acceleration ends.
	:						

(Data area [Data Window])

U (00) = 2000 : Set output current of 200% in variable “U (00)”.
U (01) = 1800 : Set output current of 180% in variable “U (01)”.
U (02) = 1500 : Set output current of 120% in variable “U (02)”.
U (03) = 1000 : Set output current of 100% in variable “U (03)”.
U (04) = 800 : Set output current of 80% in variable “U (04)”.
U (05) = 500 : Set output current of 50% in variable “U (05)”.

(Parameter)

P031 = 03

Dir	Variable name	Range of values	Default	Unit	Data size	Attribute
	Rotation direction monitoring	0: Stop 1: Forward rotation 2: Reverse rotation	-	-	Unsigned 1-word data	Readable

- Explanation

This variable can be used to monitor the direction of motor operation by the inverter. The data monitored with this variable corresponds to the data monitored by the rotation direction monitoring function (d003). This variable is read-only.

- Sample program : Program to output the output frequency data to a general-purpose analog output while operating the motor for reverse rotation at 60 Hz and forward rotation at 60 Hz

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	U(00)=			5000			: Set the offset data.
LOOP	U(01)=			Dir			: Fetch the operation direction data.
	UL(01)=			FM			: Fetch the output frequency data.
	UL(01)=	UL(01)	*	5000			: Convert the scale to 0% to 50%.
	UL(01)=	UL(01)	/	6000			
	select	U(01)					
	case	1					: When the inverter operates the motor for forward rotation
	UL(00)=	U(00)	+	UL(01)			
	case	2					: When the inverter operates the motor for reverse rotation
	UL(00)=	U(00)	-	UL(01)			
	case else						: When the motor is stopped
	UL(00)=			U(00)			
	end select						
	YA(1)=			UL(00)			: Output the data to an analog output (AM terminal).
	goto	LOOP					
	:						

PID-FB	Variable name	Range of values	Default	Unit	Data size	Attribute
	Process variable (PV), PID feedback monitoring	0 to 9990000	0	0.01 %	Unsigned 2-word data	Readable

- Explanation

This variable can be used to monitor PID feedback data in the inverter. The data monitored with this variable corresponds to the data monitored by the process variable (PV), PID feedback monitoring function (d004). This variable is read-only.

- Sample program: Program to stop inverter output when PID feedback data falls below the sleep level (to manage sleep status)
(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
LOOP	if	PID-FB	>=	U(20)	then	LABEL1	: Compare the monitored data with the sleep level.
	stop						: Stop inverter output.
	goto	LABEL2					
LABEL1	FW=	1					
LABEL2	goto	LOOP					
	:						

(Data area [Data Window])

U(20) = 2000 : Set the PID sleep level of 20% in variable “U (20)”.

F-CNV	Variable name	Range of values	Default	Unit	Data size	Attribute
	Scaled output frequency monitoring	0 to 3996000	-	0.01	Unsigned 2-word data	Readable

- Explanation

This variable can be used to monitor the converted output frequency of the inverter. The data monitored with this variable corresponds to the data monitored by the scaled output frequency monitoring function (d007). This variable is read-only.

- Sample program : Program to output the motor speed data to a general-purpose analog output
(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	UL(00)=			F-CNV			: Fetch the converted frequency data.
	UL(00)=	UL(00)	*	10000			
	UL(00)=	UL(00)	/	1800			
	YA(0)=			UL(00)			: Output the data to an analog output.
	:						

(Parameter)

b086 = 30.0 : Assign the motor speed in Hz to frequency conversion factor variable “b086”.

Tmon	Variable name	Range of values	Default	Unit	Data size	Attribute
	Torque monitoring	-200 to 200	-	%	Unsigned 1-word data	Readable

- Explanation

This variable can be used to monitor output torque of the motor operated by the inverter. The data monitored with this variable corresponds to the data monitored by the torque monitoring function (d012). This variable is read-only.

- Sample program : Program to increase the inverter output frequency when motor output torque is low (to automatically accelerate the motor)

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
LOOP	UL(00)=			Tmon			: Fetch the motor output torque data.
	U(10)=		abs	UL(00)			: Convert the data to an absolute value.
	if	U(10)	<=	U(04)	then	LBL1	: When torque is 50% or less, change the output frequency to 100 Hz.
	if	U(10)	<=	U(03)	then	LBL2	: When torque is 60% or less, change the output frequency to 80 Hz.
	if	U(10)	<=	U(02)	then	LBL3	: When torque is 70% or less, change the output frequency to 70 Hz.
	if	U(10)	<=	U(01)	then	LBL4	: When torque is 80% or less, change the output frequency to 65 Hz.
	if	U(10)	<=	U(00)	then	LBL5	: When torque is 100% or less, change the output frequency to 60 Hz.
	if	U(10)	>	U(00)	then	LBL6	: When torque exceeds 100%, change the output frequency to 60 Hz.
	goto	LBL7					
LBL1	SET-Freq=			10000			
	goto	LBL7					
LBL2	SET-Freq=			8000			
	goto	LBL7					
LBL3	SET-Freq=			7000			
	goto	LBL7					
LBL4	SET-Freq=			6500			
	goto	LBL7					
LBL5	SET-Freq=			6000			
	goto	LBL7					
LBL6	SET-Freq=			6000			
LBL7	goto	LOOP					
	:						

(Data area [Data Window])

U (00) = 100 : Set torque of 100% in variable “U (00)”.
U (01) = 80 : Set torque of 80% in variable “U (01)”.
U (02) = 70 : Set torque of 70% in variable “U (02)”.
U (03) = 60 : Set torque of 60% in variable “U (03)”.
U (04) = 50 : Set torque of 50% in variable “U (04)”.

(Parameter)

A001 = 07
A004 = 100(Hz)

Vout	Variable name	Range of values	Default	Unit	Data size	Attribute
	Output voltage monitoring	0 to 6000	-	0.1 V	Unsigned 1-word data	Readable

- Explanation

This variable can be used to monitor the inverter output voltage. The data monitored with this variable corresponds to the data monitored by the output voltage monitoring function (d013). This variable is read-only.

- Sample program : Program to turn a contact output on when output voltage exceeds 200 V

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5
	:					
LOOP	if	Vout	>=	U(00)	then	SKIP
	Y(00)=	1				
SKIP	goto	LOOP				
	:					

(Data area [Data Window])

U (00) = 2000 : 200V

Power	Variable name	Range of values	Default	Unit	Data size	Attribute
	Power monitoring	0 to 9999	-	0.1 kW	Unsigned 1-word data	Readable

- Explanation

This variable can be used to monitor power input to the inverter. The data monitored with this variable corresponds to the data monitored by the power monitoring function (d014). This variable is read-only.

- Sample program : Program to output a signal when input power is lower than the specified minimum limit or higher than the specified maximum limit

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5
	:					
LOOP	if	Power	<	U(10)	then	LBL1
	if	Power	>	U(11)	then	LBL2
	goto	LBL3				
LBL1	Y(00)=	1				
	goto	LBL3				
LBL2	Y(02)=	1				
LBL3	goto	LOOP				
	:					

(Data area [Data Window])

U (10) = 55 : U (10) = 55: Set input power of 5.5 kW in variable “U (10)”.

U (11) = 110 : U (11) = 110: Set input power of 11 kW in variable “U (11)”.

PlsCnt	Variable name	Range of values	Default	Unit	Data size	Attribute
	Pulse count monitoring	0 to 32767	-	1	Unsigned 2-word data	Readable

- Explanation

This variable can be used to reference the pulse count when the pulse counter function is selected.
The data referenced with this variable corresponds to the data monitored by the pulse counter monitoring function (d028). This variable is read-only.
This function doesn't correspond in the WJ200 Series.

- Sample program : This program turns on contact output when the pulse count exceeds 2000 (times).
(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5
	:					
LOOP	if	PlsCnt	>=	U(00)	then	SKIP
	Y(00)=	1				
SKIP	goto	LOOP				
	:					

(Data area [Data Window])
U (00) = 2000 : 2000(pulse)

POS	Variable name	Range of values	Default	Unit	Data size	Attribute
	current position monitoring	268435455 to -268435455 (1073741823 to -1073741823)	-	1	Signed 2-word data	Readable

- Explanation

This variable can be used to reference current position information when the absolute position control function is selected.
The data referenced with this variable corresponds to the data monitored by the current position monitoring function (d030). This variable is read-only.
When “03” (high-resolution absolute position control) has been selected for control pulse setting (P012), the parenthesized range of values applies (Only SJ700).
This function doesn't correspond in the L700/SJ700B Series.

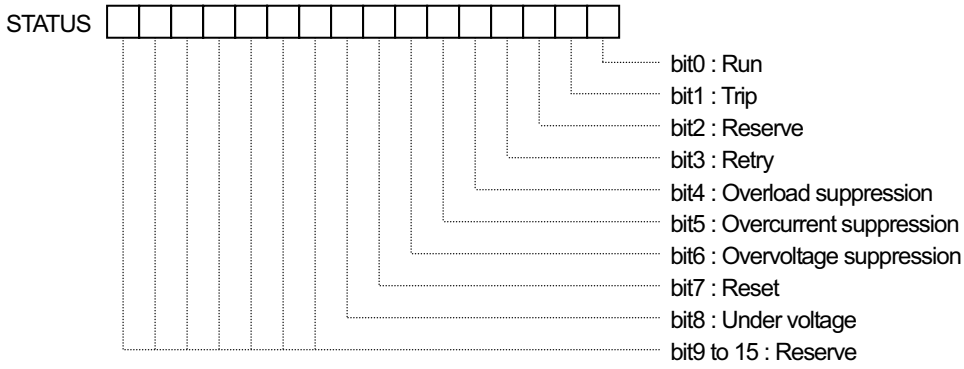
- Sample program : This program turns on contact output when the current position data exceeds 100,000 (pulses).

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5
	:					
	UL(00)=			100000		
LOOP	if	POS	>=	UL(00)	then	SKIP
	Y(00)=	1				
SKIP	goto	LOOP				
	:					

STATUS	Variable name	Range of values	Default	Unit	Data size	Attribute
	Inverter status monitoring	-	-	-	Unsigned 1-word data	Readable

- Explanation

This variable can be used to reference inverter status information.
The information to be referenced is defined as follows:



- Sample program : This program keeps turning on contact output while overvoltage restraint is applied.

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	entry						
LOOP	U(00) =	STATUS	and	U(01)			: 64 = b' 0100 0000(overvoltage restraint)
	ifs	U(00)	=	U(01)			
	then						
	Y(00)=	1					: Y(00) ON
	else						
	Y(00)=	0					: Y(00) OFF
	end if						
	goto	LOOP					
	end						

(Data area [Data Window])

U (01) = &B01000000

DCV	Variable name	Range of values	Default	Unit	Data size	Attribute
	DC voltage monitoring	0 to 9999	-	0.1V	Unsigned 1-word data	Readable

- Explanation

This variable can be used to reference the inverter DC voltage. The data referenced with this variable corresponds to the data monitored by the DC voltage monitoring function (d102). This variable is read-only.

- Sample program : This program turns on contact output when the DC voltage exceeds 350 V.

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5
	:					
LOOP	if	DCV	>=	U(00)	then	SKIP
	Y(00)=	1				
SKIP	goto	LOOP				
	:					

(Data area [Data Window])

U(00) = 3500 : 350.0V

RUN-Time	Variable name	Range of values	Default	Unit	Data size	Attribute
	Cumulative operation RUN time monitoring	0 to 999999	-	Hour	Unsigned 2-word data	Readable

- Explanation

This variable can be used to monitor the accumulated running time of the inverter. The data monitored with this variable corresponds to the data monitored by the cumulative operation RUN time monitoring function (d016). This variable is read-only.

- Sample program : Program to output a one-second pulse signal indicating the running time of the inverter to a contact output every hour

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
LOOP	UL(01)=			UL(00)			: Set the previous data in variable "UL (01)".
	UL(00)=			RUN-Time			: Fetch the running-time data.
	if	UL(00)	=	UL(01)	then	LBL1	
	Y(00)=	1					: Turn Y (00) on.
	delay off	Y(00)	TD(2)	100			: Turn Y (00) off after 1 second elapses.
LBL1	goto	LOOP					
	:						

ON-Time	Variable name	Range of values	Default	Unit	Data size	Attribute
	Cumulative power-on time monitoring	0 to 999999	-	Hour	Unsigned 2-word data	Readable

- Explanation

This variable can be used to monitor the accumulated power-on time of the inverter. The data monitored with this variable corresponds to the data monitored by the cumulative power-on time monitoring function (d017). This variable is read-only.

- Sample program : Program to convert the power-on time into a number of days and output the converted data as word data to Y (00) to Y (05)

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	UL(00)=			ON-Time			: Fetch the power-on time data.
	UL(00)=	UL(00)	/	24			: Convert the data into a number of days.
	Yw=	UL(00)	and	31			: Output the data to Yw.
	:						

ERR CNT	Variable name	Range of values	Default	Unit	Data size	Attribute
	Trip counter	0 to 65535	-	Number of times	Unsigned 1-word data	Readable

- Explanation

This variable can be used to monitor the number of times the inverter has tripped. The data monitored with this variable corresponds to the data monitored by the trip counter function (d080). This variable is read-only.

- Sample program : Program to check whether the inverter has tripped more than 10,000 times

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	Yw=	0					: Turn Y (00) to Y (05) off.
	UL(00)=			ERR CNT			: Fetch the trip count data.
	UL(01)=			100			
	If	UL(00)	<	UL(01)	then	SKIP	: When the trip count exceeds 100 (times).
	Y(00)=	1					: turn Y (00) on.
	goto	SKIP					
SKIP	Y(01)=	1					: Turn Y (01) on when the process ends.
	:						

ERR (1) to ERR (6)	Variable name	Range of values	Default	Unit	Data size	Attribute
	Trip monitoring 1 to 6	0 to 127	-	-	Unsigned 1-word data	Readable

- Explanation

These variables can be used to monitor the causes of the last six trips made by the inverter. The data monitored with this variable corresponds to the data monitored by trip monitoring functions 1 to 6 (d081 to d086). These variables are read-only.

- Sample program : Program to check whether the last six trips include one caused by overcurrent

(Code area [Code Window])

Label	Mnemonic	parameter1	parameter2	parameter3	parameter4	parameter5	
	:						
	entry						
	Yw=	0					
	if	ERR(1)	=	U(00)	then	MATCH	: Check the factor of the latest trip.
	if	ERR(2)	=	U(00)	then	MATCH	: Check the factor of the trip preceding the latest.
	if	ERR(3)	=	U(00)	then	MATCH	: Check the factor of the trip two trips before the latest.
	if	ERR(4)	=	U(00)	then	MATCH	: Check the factor of the trip three trips before the latest.
	if	ERR(5)	=	U(00)	then	MATCH	: Check the factor of the trip four trips before the latest.
	if	ERR(6)	=	U(00)	then	MATCH	: Check the factor of the trip five trips before the latest.
	Y(00)=	0					: Turn Y (00) off.
	goto	SKIP					
MATCH	Y(00)=	1					: Turn Y (00) on.
SKIP	Y(01)=	1					: Turn Y (01) on when the process ends.
	:						

(Data area [Data Window])

U (00) = 3 : Set “3” (E03) in variable “U (00)”. (Error code “E03” indicates a trip due to overcurrent.)

Chapter 3 Interface with the Inverter

This chapter explains the inverter settings to use the easy sequence function.

3.1 Inverter Settings ····· 3-1

3.2 Switching of Operation ····· 3-3

3.3 Switching of Input / Output Terminals ····· 3-3

3.4 Switching of Command Input Device ····· 3-8

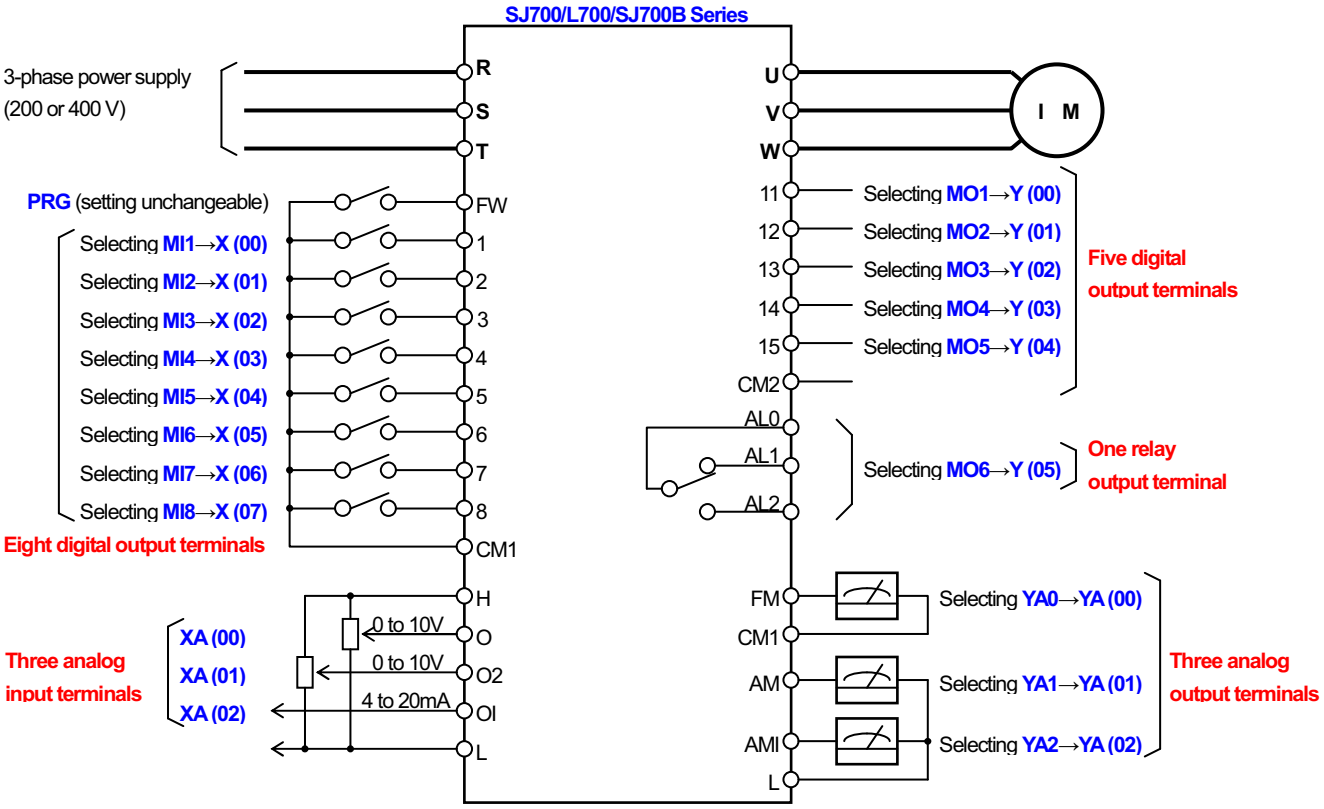
3.5 Others ····· 3-9

Chapter 3 Interface with the Inverter

3.1 Inverter Settings

The following table lists the inverter settings related to the easy sequence function.
(1)SJ700/L700/SJ700B Series

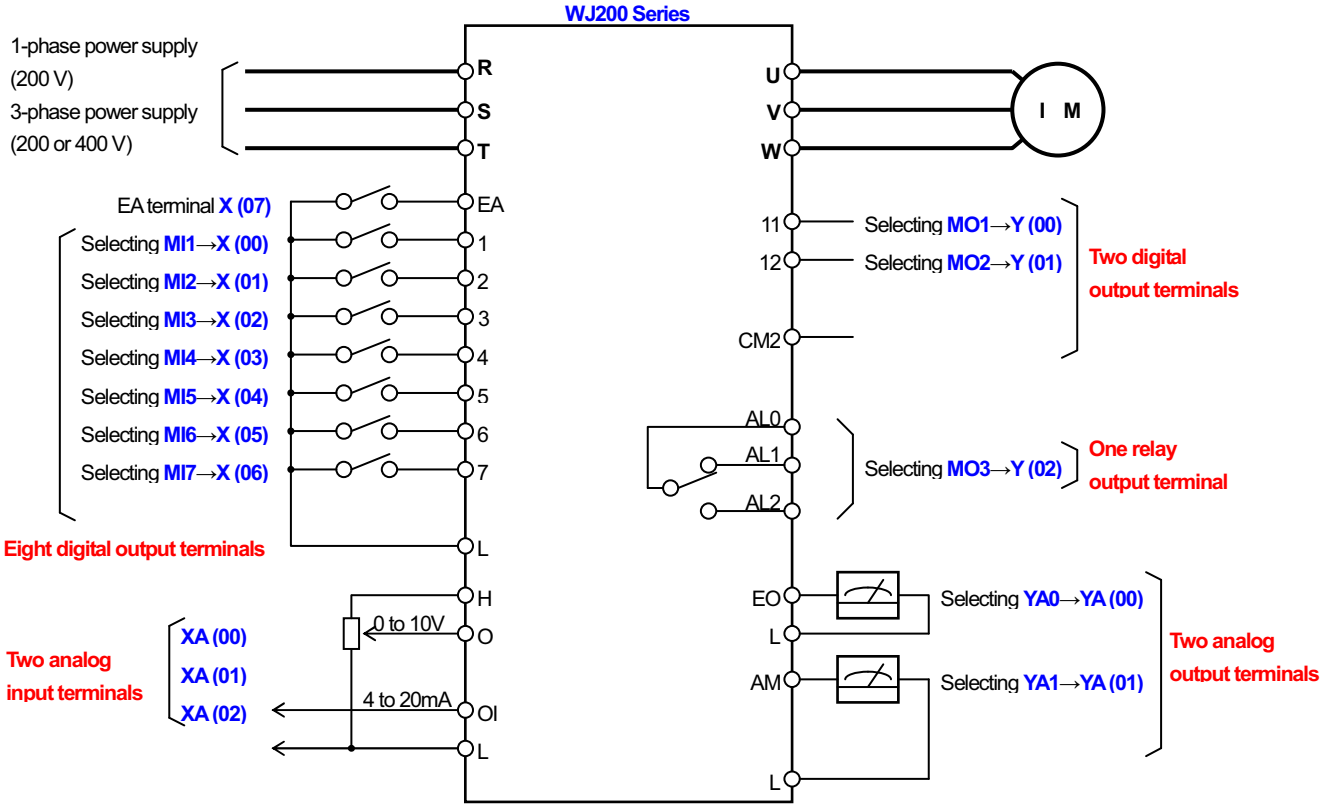
Category	Item	Terminal name	Variable notation in program	Related function code	Variable/ terminal use condition
Operation switching	Selection of easy sequence function		-	A017	-
Input/ output switching	Program run signal	PRG(FW terminal)	-	A017	The PRG terminal is enabled when A017 = 01. (The FW function is disabled.)
	General-purpose input contacts (8 contacts)	Intelligent input terminals 1 to 8	X (00) to X (07) Xw	C001 to C008	Function code settings are required.
	General-purpose output contacts (6 contacts)	Intelligent output terminals 11 to 15	Y (00) to Y (05) Yw	C021 to C025	
		Intelligent relay output terminals AL0 to AL2		C026	
	General-purpose analog inputs (3 terminals)	O terminal	XA (0)	-	No setting is required.
		OI terminal	XA (1)	-	
		O2 terminal	XA (2)	-	
	General-purpose analog outputs (3 terminals)	FM terminal	YA (0)	C027	Function code settings are required.
		AM terminal	YA (1)	C028	
		AMI terminal	YA (2)	C029	
Command switching	Frequency command selection		SET-Freq	A001	Related variables are valid only when A001 = 07.
	Operation command selection		FW, RV STA, STP, F/R	A002	Related variables are valid only when A002 = 01.
	Acceleration/deceleration input selection		ACCEL, DECEL	P031	Related variables are valid only when P031 = 03.
Other	User-defined variables (32 variables)		U(00)~U(31)	P100~P131	The variables can be redefined by using the digital operator or EzSQ.



Input and output terminals available for general-purpose input / output settings

(2)WJ200 Series

Category	Item	Terminal name	プログラム内関連変数表記	関連する機能コード	変数、端子 使用条件
Operation switching	Selection of easy sequence function		-	A017	-
Input/ output switching	Program run signal	PRG terminal	-	A017 = 01	The PRG terminal is enabled when A017 = 01.
		-		A017 = 02	The user program always runs when A017 = 02.
	General-purpose input contacts (8 contacts)	Intelligent input terminals 1 to 7 EA terminal	X (00) to X (09) Xw	C001 to C007 P003	Function code settings are required.
	General-purpose output contacts (3 contacts)	Intelligent output terminals 11 to 12	Y (00) to Y (02) Yw	C021 to C022	
		Intelligent relay output terminals AL0 to AL2		C026	
	General-purpose analog inputs (2 terminals)	O terminal	XA (0)	-	No setting is required.
		OI terminal	XA (1)	-	
	General-purpose analog outputs (2 terminals)	EO terminal	YA (0)	C027	Function code settings are required.
		AM terminal	YA (1)	C028	
Command switching	Frequency command selection		SET-Freq	A001 / A201	Related variables are valid only when A001 / A201 = 07.
	Operation command selection		FW, RV STA, STP, F/R	A002 / A202	Related variables are valid only when A002 / A202 = 01.
	Acceleration/deceleration input selection		ACCEL DECEL	P031	Related variables are valid only when P031 = 03.
Other	User-defined variables (32 variables)		U(00) to U(31)	P100 to P131	The variables can be redefined by using the digital operator or EzSQ.



Input and output terminals available for general-purpose input / output settings

3.2 Switching of Operation

(1) Easy sequence function selection (A017)

To enable the easy sequence function, specify “01” (enabling) or “02” (always on) for the easy sequence function selection (A017).
(“02” cannot be selected by the SJ700/L700/SJ700B series.)
In the SJ700/L700/SJ700B Series, when the easy sequence function is enabled, the FW terminal is switched to the PRG terminal, which is used to run the sequence program downloaded to the inverter. (The FW terminal does not function as the terminal to input the forward-rotation command while the easy sequence function is operating.)

(1) SJ700/L700/SJ700B Series

Function code	Function name	Setting	Remarks
A017	Easy sequence function selection	00 : Off (disabling) **01 : On (enabling)**	

(2) WJ200 Series

Function code	Function name	Setting	Remarks
A017	Easy sequence function selection	00 : Off (disabling) **01 : On (enabling)** **02 : On (always)**	

3.3 Switching of Input / Output Terminals

(1) Program run signal input terminal (PRG terminal)

When A017 = 01, turning on the PRG terminal (FW terminal in SJ700/L700/SJ700B) runs the sequence program downloaded to the inverter. When the PRG terminal is off, the inverter does not accept the operation command input via the RV terminal in the SJ700/L700/SJ700B Series, but waits until the sequence program runs. If the PRG terminal is turned off while the sequence program is running, the program stops. If the program is stopped while running the inverter, the inverter decelerates and stops.

(2) General-purpose contact input terminals

You can assign general-purpose input functions to the intelligent input terminals to use these terminals as general-purpose input terminals for the easy sequence function. The table below lists the inverter terminal functions and program variables corresponding to the terminal functions.
When a general-purpose input function is assigned to an intelligent input terminal, the status of the terminal is reflected in the corresponding program variables (X (**) or Xw).
You can also assign functions other than general-purpose input function to the intelligent input terminals and operate the terminals for those functions even while a sequence program is running. If both the easy sequence input and intelligent input functions have been assigned to an intelligent input terminal, the terminal functions when either input is effective (i.e., both inputs are ORed).

(1) SJ700/L700/SJ700B Series

Function code	Intelligent terminal function	Program variable	Remarks
Terminal [1] to [8] functions (C001 to C008)	56 : MI1	**X (00), Xw**	Each terminal can operate for easy sequence input and intelligent input. **(Both inputs are ORed.)**
	57 : MI2	**X (01), Xw**	
	58 : MI3	**X (02), Xw**	
	59 : MI4	**X (03), Xw**	
	60 : MI5	**X (04), Xw**	
	61 : MI6	**X (05), Xw**	
	62 : MI7	**X (06), Xw**	
	63 : MI8	**X (07), Xw**	

(2) WJ200 Series

Function code	Intelligent terminal function	Program variable	Remarks
Terminal [1] to [7] functions (C001 to C007)	56 : MI1	**X (00), Xw**	Each terminal can operate for easy sequence input and intelligent input. **(Both inputs are ORed.)**
	57 : MI2	**X (01), Xw**	
	58 : MI3	**X (02), Xw**	
	59 : MI4	**X (03), Xw**	
	60 : MI5	**X (04), Xw**	
	61 : MI6	**X (05), Xw**	
	62 : MI7	**X (06), Xw**	
EA terminal	EA terminal	**X (07), Xw**	**Related variables are valid only when P003 = 02.**
—	—	**X(08), X(09)**	**Reserve**

(3) General-purpose contact output terminals

You can assign general-purpose output functions to intelligent output terminals and the alarm relay terminal to use these terminals as general-purpose output terminals for the easy sequence function. The table below lists the inverter terminal functions and program variables corresponding to the terminal functions.
When a general-purpose output function is assigned to one of these output terminals, the data stored in variables “Y (**)” or “Yw” can be output to the terminal.
You can also assign functions other than general-purpose output function to the output terminals and operate the terminals for those functions even while a sequence program is running.

(1) SJ700/L700/SJ700B Series

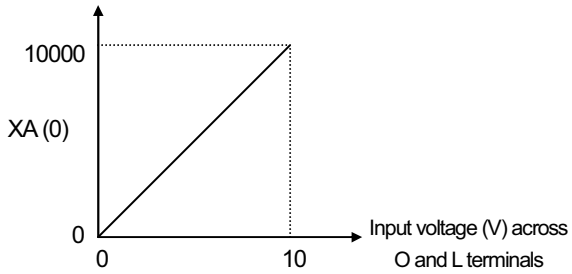
Function code	Intelligent terminal function	Program variable	Remarks
Terminal [11] to [15] functions (C021 to C025)	44 : MO1	**Y (00), Yw**	
	45 : MO2	**Y (01), Yw**	
	46 : MO3	**Y (02), Yw**	
	47 : MO4	**Y (03), Yw**	
	48 : MO5	**Y (04), Yw**	
Alarm relay terminal function (C026)	49 : MO6	**Y (05), Yw**	

(2) WJ200 Series

Function code	Intelligent terminal function	Program variable	Remarks
Terminal [11] to [12] functions (C021 to C022)	44 : MO1	**Y (00), Yw**	
	45 : MO2	**Y (01), Yw**	
Alarm relay terminal function (C026)	46 : MO3	**Y (02), Yw**	
—	—	**Y (03) to Y (05)**	**Reserve**

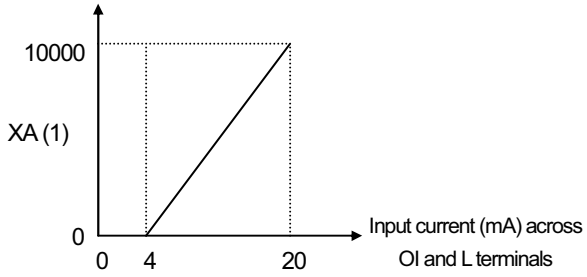
(4) General-purpose analog input terminal (O terminal)

You can use the O terminal as a general-purpose analog input terminal. By referencing the data stored in variable “XA (0)”, the data (ranging from 0 to 10000) input via the O terminal can be fetched.
Switching the O terminal to a general-purpose analog input terminal does not require any special setting. Even when the O terminal is used to input frequency commands, the O terminal can also function as a general-purpose analog input terminal. Note that the handling of data fetched via the O terminal depends on the settings made by the [O]-[L] input functions (A011 to A015).
The figure below shows the relationship between the input voltage and the value to be fetched (when the settings of functions “A011” to “A015” are the defaults).



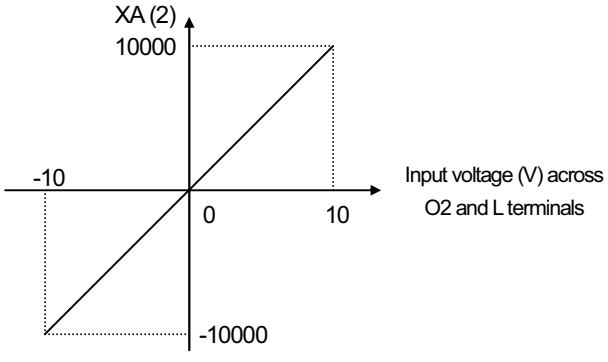
(5) General-purpose analog input terminal (OI terminal)

You can use the OI terminal as a general-purpose analog input terminal. By referencing the data stored in variable “XA (1)”, the data (ranging from 0 to 10000) input via the O terminal can be fetched.
Switching the OI terminal to a general-purpose analog input terminal does not require any special setting. Even when the OI terminal is used to input frequency commands, the OI terminal can also function as a general-purpose analog input terminal. Note that the handling of data fetched via the OI terminal depends on the settings made by the [OI]-[L] input functions (A101 to A105).
The figure below shows the relationship between the input current and the value to be fetched (when the settings of functions “A101” to “A105” are the defaults).



(6) General-purpose analog input terminal (O2 terminal)

This terminal function is available for only SJ700/L700/SJ700B series. WJ200 series does not correspond.
You can use the O2 terminal as a general-purpose analog input terminal. By referencing the data stored in variable “XA (2)”, the data (ranging from -10000 to 10000) input via the O2 terminal can be fetched.
Switching the O2 terminal to a general-purpose analog input terminal does not require any special setting. Even when the O2 terminal is used to input frequency commands, the O2 terminal can also function as a general-purpose analog input terminal. Note that the handling of data fetched via the O2 terminal depends on the settings made by the [O2]-[L] input functions (A111 to A115).
The figure below shows the relationship between the input voltage and the value to be fetched (when the settings of functions “A111” to “A115” are the defaults).

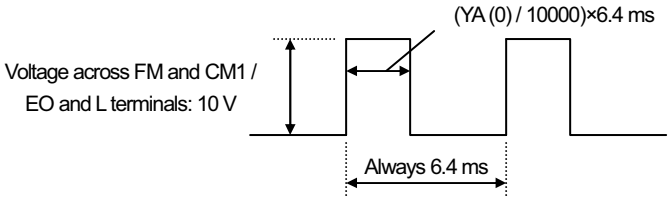


(7) General-purpose analog output terminal

(FM terminal in SJ700/L700/SJ700B / EO terminal in WJ200)

You can use the FM terminal as a general-purpose analog output terminal for the easy sequence function. For this purpose, specify "12" (YA0: general-purpose output 0) for the [FM] / [EO] signal selection (C027).
When used as a general-purpose analog output terminal, the FM / EO terminal can output the pulse signal that corresponds to the data (0 to 10000) stored in variable "YA (0)". The FM / EO output characteristics follow the FM / EO gain adjustment (C105). The figure below shows the output waveform (with "C105" set to 100%).

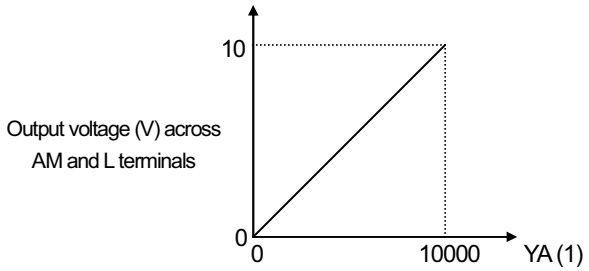
Function code	Function name	Setting	Remarks
C027	[FM] signal selection (SJ700/L700/SJ700B) / [EO] signal selection (WJ200)	**12 : General-purpose output 0**	The analog output of program variable (**YA (0)**) data is enabled only when **C027 = 12**.



(8) General-purpose analog output terminal (AM terminal)

You can use the AM terminal as a general-purpose analog output terminal for the easy sequence function. For this purpose, specify "13" (YA1: general-purpose output 1) for the [AM] signal selection (C028).
When used as a general-purpose analog output terminal, the AM terminal can output the data (0 to 10000) stored in variable "YA (1)". The AM output characteristics follow the AM gain adjustment (C106) and AM bias adjustment (C109). The figure below shows the relationship between the value of variable "YA (1)" and AM output voltage (with "C106" set to 100% and "C109" set to 0%).

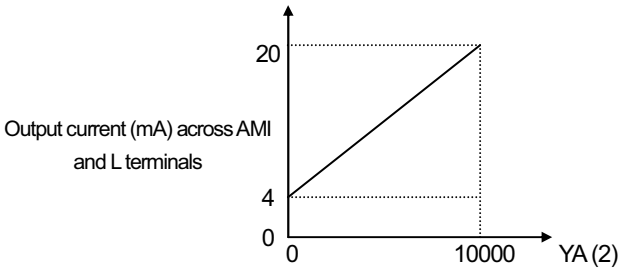
Function code	Function name	Setting	Remarks
C028	[AM] signal selection	**13 : General-purpose output 1**	The analog output of program variable (**YA (1)**) data is enabled only when **C028 = 13**.



(9) General-purpose analog output terminal (AMI terminal)

This terminal function is available for only SJ700/L700/SJ700B series. WJ200 series does not correspond.
You can use the AMI terminal as a general-purpose analog output terminal for the easy sequence function. For this purpose, specify "14" (YA2: general-purpose output 2) for the [AMI] signal selection (C029).
When used as a general-purpose analog output terminal, the AMI terminal can output the data (0 to 10000) stored in variable "YA (2)". The AMI output characteristics follow the AMI gain adjustment (C107) and AMI bias adjustment (C110). The figure below shows the relationship between the values of variable "YA (2)" and AMI output voltage (with "C107" set to 100% and "C110" set to 0%).

Function code	Function name	Setting	Remarks
C029	[AMI] signal selection	**14 : General-purpose output 2**	The analog output of program variable (**YA (2)**) data is enabled only when **C029 = 14**.



3.4 Switching of Command Input Device

(1) Frequency source setting (A001 / A201)

Selection of the device used to input frequency commands follows the frequency source setting (A001 / A201), regardless of whether the easy sequence function is enabled.
EzSQ provides variable “SET-Freq” for setting the inverter output frequency. To enable the use of this variable, specify “07” (PRG) for the frequency source setting (A001 / A201). Otherwise, the frequency setting in “SET-Freq” will not be reflected in the inverter.
A201 is available for only WJ200 series. SJ700, L700 and SJ700B series do not correspond.

Function code	Function name	Setting	Remarks
A001 / A201	Frequency source setting	**07 : PRG (easy sequence)**	The use of program variable (**SET-Freq**) data is enabled only when **A001 / A201 = 07**.

(2) Run command source setting (A002 / A202)

Selection of the device used to input operation commands follows the run command source setting (A002 / A202), regardless of whether the easy sequence function is enabled.
EzSQ provides variables “FW”, “RV”, “STA”, “STP”, and “F/R” for the inverter control related to operation commands. Since these variables are handled as terminal input data, specify “01” (TRM) for the run command source setting (A002 / A202) to enable the use of the variables.
A202 is available for only WJ200 series. SJ700, L700 and SJ700B series do not correspond.

Function code	Function name	Setting	Remarks
A002 / A202	Run command source setting	**01 : TRM (control circuit terminal block)**	The use of program variables **“FW”, “RV”, “STA”, “STP”, and “F/R”** is enabled only when **A002 / A202 = 01**.

(3) Accel / decel time input selection (P031)

Selection of the device used to input acceleration/deceleration time settings follows the setting of accel / decel time input selection (P031), regardless of whether the easy sequence function is enabled.
EzSQ provides variables “ACCEL” and “DECEL” for the inverter control related to acceleration and deceleration time. To enable the use of these variable, specify “03” (PRG) for the accel / decel time input selection (P031). Otherwise, the acceleration/deceleration time settings in “ACCEL” and “DECEL” will not be reflected in the inverter.

Function code	Function name	Setting	Remarks
P031	Accel / decel time input selection	**03 : PRG (easy sequence)**	The use of program variables "**ACCEL**" and "**DECEL**" is enabled only when **P031 = 03**.

3.5 Others

(1) User-defined variables “U (00)” to “U (31)” (P100 to P131)

The easy sequence function provides 32 user-defined variables “U (00)” to “U (31)”, which correspond to inverter parameters “P100” to “P131”. You can use the “Data Window” of EzSQ to set data in these variables, and store them as inverter parameters “P100” to “P131” by downloading the program containing the variables to the inverter. After downloading the program, you can update the parameter data by accessing parameters “P100” to “P131” from the digital operator connected to the inverter without using EzSQ.

Function code	Function name	Setting	Remarks
P100 to P131	User-defined variables U (00) to U (31)	0 to 65535 (to be defined by user)	Updateable via digital operator or EzSQ

(2) User monitor “Umon (00)” to “Umon (02)” (d025 to d027)

You can carry out the monitor display of the arbitrary data in a program to the digital operator connected to the inverter.
These variables can be used as signed 2-word variables. The data of UL (ii) can be displayed as it is.

Function code	Function name	Setting	Remarks
d025 to d027	User monitor 0 to 2	-2147483647～2147483647	

(3) User trip “trip 0” to “trip 9” (Error code E50 to E59)

This instruction makes inverter trip.
The error codes are E50 to E59. It corresponds to “trip 0” to “trip 9” respectively.

Chapter 4 Errors and Troubleshooting

This chapter explains the errors that may occur when using the easy sequence function and the methods of handling the errors.

4.1 Errors Specific to the Easy Sequence Function ····· 4-1

4.2 Troubleshooting ····· 4-2

Chapter 4 Errors and Troubleshooting

4.1 Errors Specific to the Easy Sequence Function

The table below lists the errors that are specific to the easy sequence function. For other errors in the inverter, refer to the SJ700 Series Inverter Instruction Manual.

No.	Error (causing inverter trip)	Factor code (*1)	Description
1	Invalid instruction	E43.*	The inverter assumes an error if you try to download the program that exceeds capacity, and all programs are not downloaded, or if the downloaded program includes an invalid instruction code. This error is detected if the PRG terminal is turned on when the downloaded program has been destroyed or no program has been downloaded.
2	Nesting count error	E44.*	The inverter assumes an error if subroutines, for statements, and/or next statements are nested in more than eight layers.
3	Instruction error 1	E45.*	- The inverter assumes an error if the jump destination of a goto statement is not the beginning of a nested for statement but preceded by the end of a nested next statement. - The inverter assumes an error if the variable “U (xx)” referenced via another variable is not found. - The inverter assumes an error if an arithmetic instruction results in overflow or underflow (WJ200 Step2 is excluded.), or causes a division by zero. - The inverter assumes an error if a chg param instruction causes reference to a nonexistent parameter, change of a parameter value outside the setting limits, or updating of a parameter that cannot be updated during inverter operation.
4	User trip 0 to 9	E50.* to E59.*	

*1 The asterisk (*) in the factor code represents an inverter status code.

4.2 Troubleshooting

The table below shows how to handle the errors specific to the easy sequence function. For details on other errors in the inverter, refer to the inverter instruction manual.

Factor code	Error (causing inverter trip)	Possible cause	Checking method	Corrective action
E43	Invalid instruction	The PRG terminal was turned on without a program downloaded to the inverter.	Upload the program from the inverter to the personal computer, and check whether the uploaded program matches one of the programs stored on the personal computer.	Recreate the program, and then download it to the inverter.
		The program stored in inverter memory has been destroyed.		
E44	Nesting count error	Subroutines are nested in more than eight layers.	Read the program to check the number of nesting layers.	Correct the program so that the number of layers will be eight or less.
		for-next loop statements are nested in more than eight layers.		
		if statements are nested in more than eight layers.		
E45	Instruction error 1	The jump destination of a goto instruction is a next instruction to end a for or other loop.	Check whether each goto instruction jumps to an instruction that ends a loop.	Correct the jump destinations of goto instructions.
		The variable “U (ii)” referenced via another variable is not found.	Check the numerical value specified in “U (ii)”.	Correct the value of variable “U (ii)” or limit the range of values of variable “U (ii)”.
		An arithmetic instruction caused: - overflow, - underflow, or - division by zero.	Check the program for the instruction causing overflow, underflow, or division by zero.	Correct the program so that no arithmetic instruction causes overflow, underflow, or division by zero.
		A chg param instruction caused: - reference to a nonexistent parameter, - writing of a value out of the setting range, - change of a parameter value (during inverter operation) that cannot be updated during inverter operation, or - change of a parameter value of which updating is restricted by software lock (when software lock is enabled).	- Check the parameters and the values to be written. - If the error has occurred during inverter operation, check whether the parameter in question is the one that can be updated during inverter operation. (*1) - Check the setting of software lock selection (b031). (*1)	- Correct the parameters or the values to be written to parameters so that they will be within the setting range. - Disable software lock. (*2) - If the parameter to be updated is the one that cannot be updated during inverter operation, change the setting of software lock selection (b031) to “10” to switch to the mode enabling parameter updating during inverter operation. (*2)

*1 For details, refer to the inverter instruction manual.

*2 The settings of some parameters affect inverter output and the functions of input/output terminals. Changing the settings of said parameters during inverter operation may entail the risk of abnormal operation of the motor or machine driven by the inverter. If you change the setting of a parameter after disabling the software lock or switching to the mode enabling parameter updating during inverter operation, check the influence of the update beforehand to ensure the safety of system operation.