OMRON

Inverter

RX2 Series

DriveProgramming

User's Manual

3G3RX2 Series
CX-Drive

I622-E1-01

Trademarks

Introduction

Thank you for purchasing the Inverter/Servo support tool CX-Drive and 3G3RX2 Series Inverter.

This manual describes the specifications and operating methods of the DriveProgramming for the inverter.

When you use this product, refer to the High-function General-purpose Inverter 3G3RX2 Series User's Manual (I620), besides the CX-Drive Operation Manual (W453).

Intended Readers

This manual is intended for the following personnel, who must also have knowledge of electrical systems (an electrical engineer or the equivalent).

- Personnel in charge of introducing FA systems.
- Personnel in charge of designing FA systems.
- Personnel in charge of installing and connecting FA systems.
- Personnel in charge of managing FA systems and facilities.

Notice

This manual contains information you need to know to use the DriveProgramming.

Before using this product, read this manual and gain a full understanding of the information provided herein.

After you finished reading this manual, keep it in a convenient place so that it can be referenced at any time.

Make sure this manual is delivered to the end user.

Manual Configuration

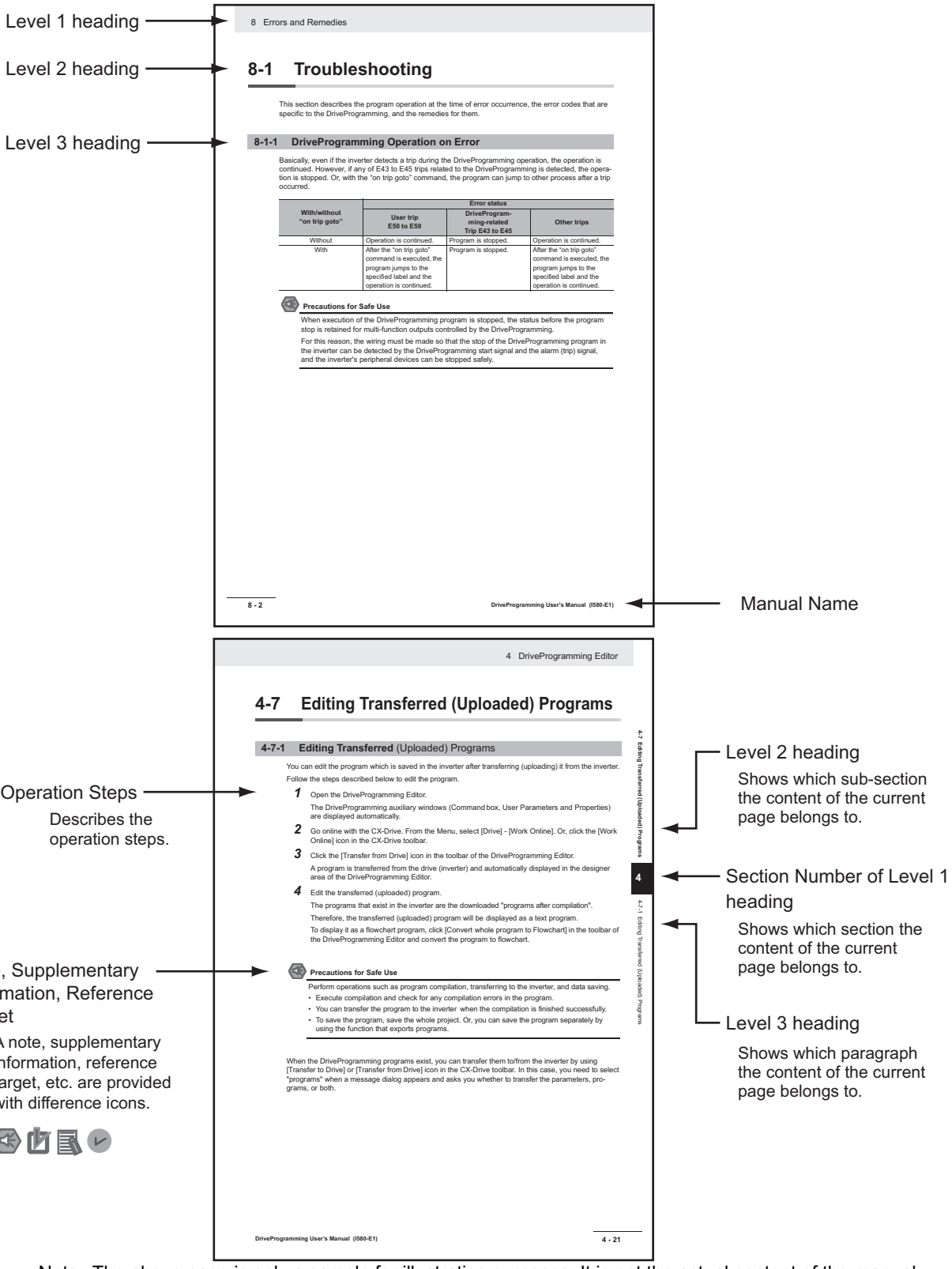
This manual is compiled section by section for user's convenience as follows.

Section		Overview
Section 1	Overview	This section describes an overview and the system configuration of the DriveProgramming.
Section 2	Specifications	This section describes the specifications of the DriveProgramming.
Section 3	Operation Procedure for DriveProgramming	This section describes the operation procedure of the DriveProgaramming, related parameters, and program structures.
Section 4	DriveProgramming Editor	This section describes how to start the DriveProgramming Editor, saving and loading data, and details on parts of the Editor.
Section 5	DriveProgramming User Variables	This section describes the user variables provided for DriveProgramming.
Section 6	DriveProgramming Commands	This section describes the commands provided for DriveProgramming.
Section 7	Precautions for Use of Parameters for DriveProgramming	This section describes the precautions for use of parameters for the DriveProgramming.
Section 8	Errors and Remedies	This section describes the program operation at the time of error occurrence, the errors that are specific to the DriveProgramming, as well as the causes and remedies.

Manual Structure

Page Structure

The following page structure and symbol icons are used in this user's manual.



Note The above page is only a sample for illustrative purposes. It is not the actual content of the manual.

Special Information

Special information in this user's manual is classified as follows:

Precautions for Safe Use

Precautions on what to do and what not to do to ensure safe usage of the product.

Precautions for Correct Use

Precautions on what to do and what not to do to ensure proper operation and performance.

Additional Information

Additional information to read as required.

This information is provided to increase understanding or make operation easier.

Sections in this Manual

CONTENTS

Section 4 DriveProgramming Editor

Section 5 DriveProgramming User Variables

Section 6 DriveProgramming Commands

Section 7 Precautions for Use of Parameters for DriveProgramming

Section 8 Errors and Remedies

Terms and Conditions Agreement

WARRANTY

- The warranty period for the Software is one year from the date of purchase, unless otherwise specifically agreed.
- If the User discovers defect of the Software (substantial non-conformity with the manual), and return it to OMRON within the above warranty period, OMRON will replace the Software without charge by offering media or download from OMRON's website. And if the User discovers defect of media which is attributable to OMRON and return it to OMRON within the above warranty period, OMRON will replace defective media without charge. If OMRON is unable to replace defective media or correct the Software, the liability of OMRON and the User's remedy shall be limited to the refund of the license fee paid to OMRON for the Software.

LIMITATION OF LIABILITY

- THE ABOVE WARRANTY SHALL CONSTITUTE THE USER'S SOLE AND EXCLUSIVE REMEDIES AGAINST OMRON AND THERE ARE NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTY OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE. IN NO EVENT, OMRON WILL BE LIABLE FOR ANY LOST PROFITS OR OTHER INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF USE OF THE SOFTWARE.
- OMRON SHALL HAVE NO LIABILITY FOR DEFECT OF THE SOFTWARE BASED ON MODIFICATION OR ALTERNATION TO THE SOFTWARE BY THE USER OR ANY THIRD PARTY.
- OMRON SHALL HAVE NO LIABILITY FOR SOFTWARE DEVELOPED BY THE USER OR ANY THIRD PARTY BASED ON THE SOFTWARE OR ANY CONSEQUENCE THEREOF.

APPLICABLE CONDITIONS

USER SHALL NOT USE THE SOFTWARE FOR THE PURPOSE THAT IS NOT PROVIDED IN THE ATTACHED USER MANUAL.

CHANGE IN SPECIFICATION

The software specifications and accessories may be changed at any time based on improvements and other reasons.

ERRORS AND OMISSIONS

The information in this manual has been carefully checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical, or proofreading errors, or omissions.

Safety Precautions

Indications and Meanings of Safety Information

In this manual, the following precautions and signal words are used to provide information to ensure the safe use of the DriveProgramming.

The information provided here is vital to safety. Strictly observe the precautions provided.

Meanings of Signal Words

⚠DANGER	Indicates an imminently hazardous situation which, if not avoided, is likely to result in serious injury or may result in death. Additionally, there may be severe property damage.
⚠WARNING	Indicates a potentially hazardous situation which, if not avoided, will result in minor or moderate injury, or may result in serious injury or death. Additionally, there may be significant property damage.
⚠CAUTION	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury or in property damage.

Alert Symbols in this Document

	⃠ This symbol indicates a prohibited item (an item you must not do). The specific instruction is indicated using an illustration or text inside or near ⃠. The symbol shown to the left indicates "disassembly prohibited."
	△ This symbol indicates danger and caution. The specific instruction is indicated using an illustration or text inside or near △. The symbol shown to the left indicates "beware of electric shock."
	△ This symbol indicates danger and caution. The specific instruction is indicated using an illustration or text inside or near △. The symbol shown to the left indicates "non-specific general danger."
	△ This symbol indicates caution (warnings included). The specific instruction is indicated using an illustration or text inside or near △. The symbol shown to the left indicates "risk of hot surface."
	● This symbol indicates a compulsory item (an item that must be done). The specific instruction is indicated using an illustration or text inside or near ●. The symbol shown to the left indicates "general compulsory items."
	● This symbol indicates a compulsory item (an item that must be done). The specific instruction is indicated using an illustration or text inside or near ●. The symbol shown to the left indicates "grounding required."

WARNING	
Turn off the power supply and implement wiring correctly. Not doing so may result in a serious injury due to an electric shock.	
Wiring work must be carried out only by qualified personnel. Not doing so may result in a serious injury due to an electric shock.	
Do not change wiring and slide switches (SW1 to SW6), put on or take off Operator and optional devices, replace cooling fans while the input power is being supplied. Doing so may result in a serious injury due to an electric shock.	
Be sure to ground the unit. Not doing so may result in a serious injury due to an electric shock or fire. (200-V class: type-D grounding, 400-V class: type-C grounding)	
Do not remove the terminal cover during the power supply and 15 minutes *a *b after the power shut off. Doing so may result in a serious injury due to an electric shock.	
Do not operate the Operator or switches with wet hands. Doing so may result in a serious injury due to an electric shock.	
Inspection of the inverter must be conducted after the power supply was turned off. Not doing so may result in a serious injury due to an electric shock. The main power supply is not necessarily shut off even if the emergency shut off function is activated.	
Do not touch the inverter fins, braking resistors and the motor, which become too hot during the power supply and for some time after the power shut off. Doing so may result in a burn.	

*a. 10 minutes: For models 3G3RX2-A2004 to A2220 and 3G3RX2-A4007 to A4220

*b. 15 minutes: For models 3G3RX2-A2300 to A2550 and 3G3RX2-A4300 to A4550, -B4750, -B4900, -B411K, -B413K

Caution	
Do not connect resistors to the terminals (PD/+1, P/+, N/-) directly. Doing so might result in a small-scale fire, heat generation, or damage to the unit.	
Install a stop motion device to ensure safety. Not doing so might result in a minor injury. (A holding brake is not a stop motion device designed to ensure safety.)	
Be sure to use a specified type of braking resistor/regenerative braking unit. In case of a braking resistor, install a thermal relay that monitors the temperature of the resistor. Not doing so might result in a moderate burn due to the heat generated in the braking resistor/regenerative braking unit. Configure a sequence that enables the inverter power to turn off when unusual over eating is detected in the braking resistor/regenerative braking unit.	
The inverter has high voltage parts inside which, if short-circuited, might cause damage to itself or other property. Place covers on the openings or take other precautions to make sure that no metal objects such as cutting bits or lead wire scraps go inside when installing and wiring.	
Take safety precautions such as setting up a molded-case circuit breaker (MCCB) that matches the inverter capacity on the power supply side. Not doing so might result in damage to property due to the short circuit of the load.	
Do not dismantle, repair or modify the product. Doing so may result in an injury.	
If a parameter is set incorrectly when starting up, adjusting, maintaining, or replacing, an unexpected operation may occur.	
If the DriveProgramming stops during multi-function output, the output status is held. Take safety precautions such as stopping peripheral devices.	

Precautions for Safe Use

Operation and Adjustment

- If the clock command is used in DriveProgramming, an unexpected operation may occur due to weak battery. Take measures such as detecting a weak battery by [E042]RTC Error and stopping the inverter or programs. When the LCD Operator is removed or disconnected, DriveProgramming is in a waiting status by the clock command.
- When using DriveProgramming, confirm that the program data is downloaded normally before starting operation.

Regulations and Standards

To export (or provide to nonresident aliens) any part of this product that falls under the category of goods (or technologies) for which an export certificate or license is mandatory according to the Foreign Exchange and Foreign Trade Control Law of Japan, an export certificate or license (or service transaction approval) according to this law is required.

Related Manuals

You need information on the devices connected for operating this product.

Please see the manuals below for related product information.

Name	Catalog number
CX-Drive Operation Manual	W453
High-function General-purpose Inverter 3G3RX2 Series User's Manual	I620

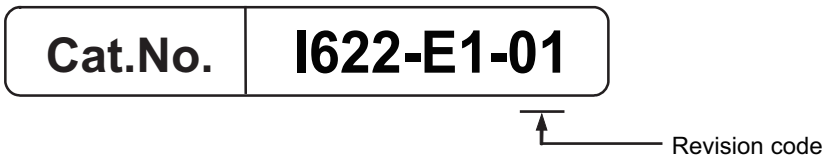
Additional Information

For the inverter operation, refer to the High-function General-purpose Inverter 3G3RX2 Series User's Manual (I620).

Revision History

The manual revision code is a number appended to the end of the catalog number found on the front and back covers.

Example



Revision code	Revision date	Revised Content
01	January 2019	Original production

1

Overview

This section describes an overview and the system configuration of the DriveProgramming.

1-1 Overview of DriveProgramming

The DriveProgramming is the simple sequence function built into the inverter.

To create sequence programs and check their status, you use the Inverter/Servo supporting tool, CX-Drive.

Transfer (download) the created programs to the 3G3RX2 Series Inverter so that the stand-alone inverter can perform simple sequence control.

Features of DriveProgramming

The DriveProgramming has the following features.

- The DriveProgramming supports both flowchart and text language method programming.
- You can create a program divided into up to five tasks.
- Five tasks can be processed in parallel.
- It is possible to execute user programs externally by settings of input terminals.
- You can use the I/O terminals by allocating them to the parameters.
- The LCD Operator enables you to change the settings of the frequency, acceleration/deceleration time, and other parameters (variables) that require on-site adjustment by specifying the user parameters (UE-10 to UE-73, UF-02 to UF-32), without connecting the computer.
- Because user programs are stored in the internal EEPROM of the inverter, you can start a program immediately after the inverter power supply is turned on.
- The LCD Operator has a built-in clock function. Therefore, you can create programs that use the LCD Operator's clock function. When a clock function is used, batteries (option: CR2032, 3V) are required.

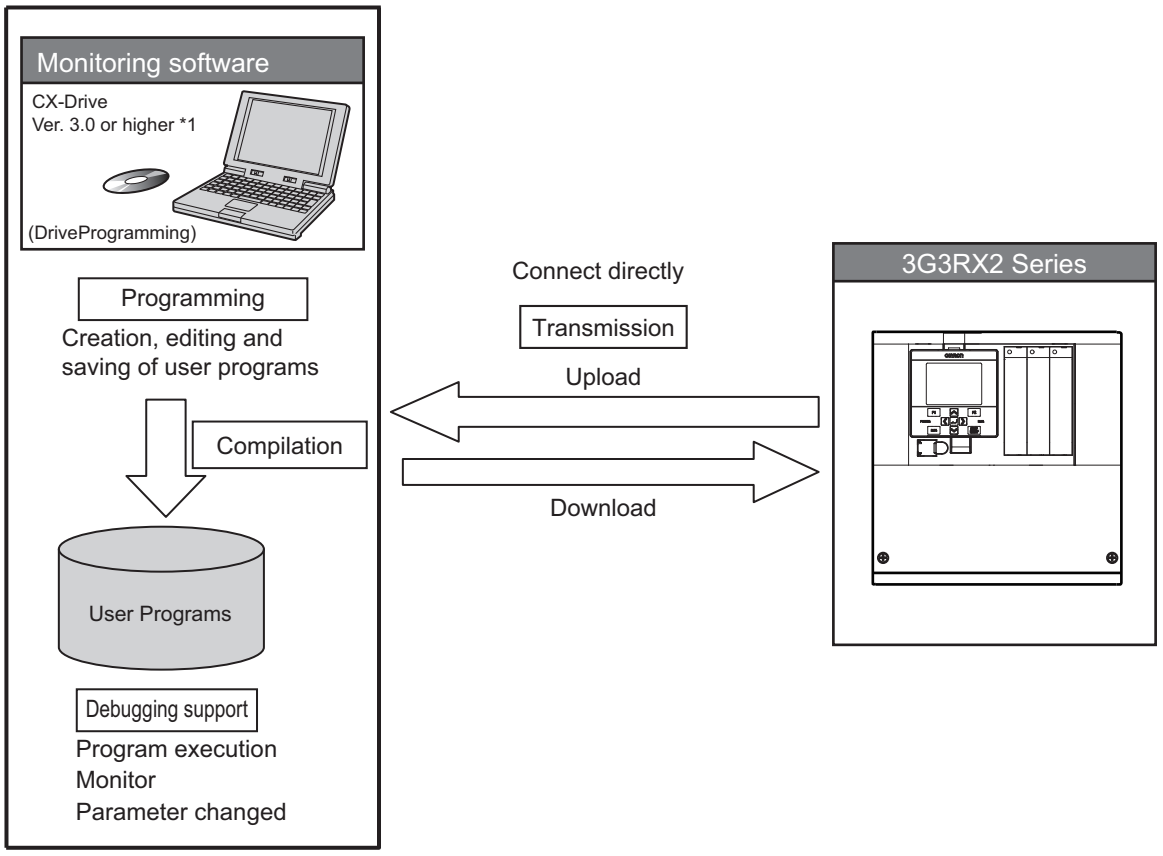
Precautions for Safe Use

If the clock command is used in DriveProgramming, an unexpected operation may occur due to weak battery. Take measures such as detecting a weak battery by [E042]RTC Error and stopping the inverter or programs. When the LCD Operator is removed or disconnected, DriveProgramming is in a waiting status by the clock command.

When the LCD Operator is removed or disconnected, DriveProgramming is in a waiting status by the clock command.

The following table shows the main functions of the DriveProgramming Editor available in CX-Drive.

Function	Description
Programming	Supports the creation, editing, saving, reading, and printing of user programs.
Compilation	Performs check of user programs and generates intermediate codes.
Transfer	Downloads a user program to the inverter. Uploads a user program from the inverter.
Debugging support	Starts and stops the execution of a program. The user can check the inverter status monitor etc.

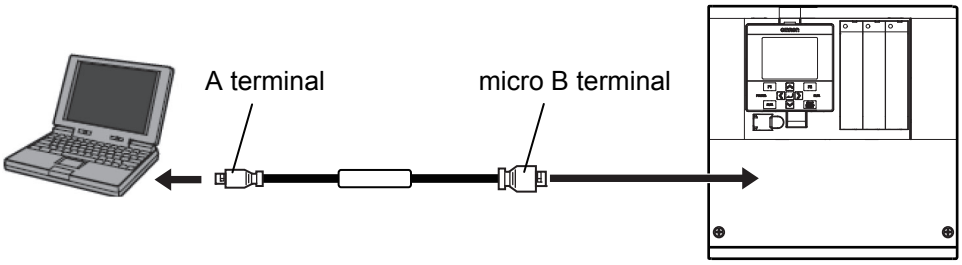


*1. Ver 3.0 or higher: Version supported for 3G3RX2

1-2 Preparation and System Configuration

You must prepare the following items to create user programs with functions of the DriveProgramming in CX-Drive and execute the programs in the 3G3RX2 Series Inverter.

- 3G3RX2 Series Inverter
- Windows personal computer
  (As for the supported OS (operation system), refer to *CX-One User's Manual* (W463).
- CX-Drive needs Ver 3.0 or higher.
  (The CX-Drive is included in the FA Integrated Tool Package, CX-One.)
- Prepare the following PC-inverter connection cable.
  Commercially-available USB cable (mini-B)



Additional Information

For how to install or upgrade the CX-Drive, refer to the CX-Drive Operation Manual (W453).

2

Specifications

This section describes the specifications of the DriveProgramming.

2-1 Specifications

The following table shows the specifications related to the DriveProgramming.

Item		Specifications	
Program specifications	Programming language	Flowchart and text language method	
	Input device	Windows personal computer (As for the supported OS (operation system), refer to CX-One User's Manual (W463).	
	Program capacity	1,024 steps max. a task Total 7,680 byte max. for 5 tasks	
	Programming support function	Functions supported in Inverter/Servo support tool CX-Drive • Program editing and display • Program compilation (Program syntax check) • Program downloading, uploading, and all clear	
	Execution format	• Execution by interpreter • Execution cycle: Select 1 ms or 2 ms /step (Parameter [UE-01]). • Subroutine call supported (Nesting in 8 levels max.) • Multi-task: 5 task max.	
Input/output functions	External input	DriveProgramming start	Select in the EzSQ function enable (UE-02) • Start/stop via PRG input terminal or DriveProgramming start
		Input Terminal	X (00) to X (10)/11 points max.
		Frequency reference input (Analog input)	XA(0): 0 to 10 V/0 to 20 mA (Ai1 terminal) XA(1): 0 to 10 V/0 to 20 mA (Ai2 terminal) XA(2): −10 to 10 V (Ai3 terminal)
	External output	Output terminal (Include relay contact two points.)	Y(00) to Y(06)/7 points
		Monitor output (Analog output)	YA(0): FM terminal YA(1): Ao1 terminal YA(2): Ao2 terminal
Commands	Program control commands	• Loop ("for") • Unconditional branch ("goto") • Time control ("wait") • Conditional branch ("if then", "ifs then", "select case", "until", "while") • Subroutine ("call", "sub") • Others ("entry", "end", "inc", "dec")	
	Arithmetic commands	• Four arithmetic operations (+, −, *, and /) • Remainder ("mod") and assignment (=) • Absolute value ("abs") • Logical operations ("or", "and", "xor", "not")	
	I/O control	• I/O terminal function (bit input, word input, bit output, and word output) • Reads inverter input terminals • Reads/writes inverter output terminals	
	Timer control	Delay operation with the timer and Control with the timer counter	
	Parameter control	Changes setting data of specified parameter number	
	Inverter control	• Executes and stops forward/reverse operation • Generates a trip by the DriveProgramming (E050 to E059/10 points) • Frequency reference and acceleration/deceleration time settings	

Item		Specifications
Function variable	User parameter variable	U(00) to U(63)/64 points
	Internal user variable	UL(00) to UL(15)/16 points
	Frequency reference variable	SET-Freq
	Acceleration time variable	ACCEL
	Deceleration time variable	DECEL
	Inverter monitor variable	The monitor functions for the inverter are available as variables. FM, Iout, Dir, PID-FB, F-CNV, Tmon, Vout, Power, RUN-Time, ON-Time, PlsCnt, POS, STATUS, DCV, ERR-CNT, ERR(1) to ERR(10)
	Input variable	The function options of the Input terminal [1] function (CA-01 to CA-11) for the inverter are available. FW, RV, CF1, CF2, CF3, CF4, SF1, SF2, SF3, SF4, SF5, SF6, SF7, ADD, SCHG, STA, STP, FR, AHD, FUP, FDN, UDC, F-OP, SET, RS, JG, DB, 2CH, FRS, EXT, USP, CS, SFT, BOK, OLR, KHC, OKHC, PID, PIDC, PID2, PIDC2, PID3, PIDC3, PID4, PIDC4, SVC1, SVC2, SVC3, SVC4, PRO, PIO1, PIO2, SLEP, WAKE, TL, TRQ1, TRQ2, PPI, CAS, SON, FOC, ATR, TBS, ORT, LAC, PCLR, STAT, PUP, PDN, CP1, CP2, CP3, CP4, ORL, ORG, FOT, ROT, SPD, PSET, Mi1, Mi2, Mi3, Mi4, Mi5, Mi6, Mi7, Mi8, Mi9, Mi10, Mi11, PCC, ECOM, PRG, HLD, REN, DISP, PLA, PLB, EMF, COK, DTR, PLZ, and TCH
Function variable	Output variable	The function options of the Output terminal [11] function (CC-01 to CC-07) for the inverter are available. RUN, FA1, FA2, FA3, FA4, FA5, IRDY, FWR, RVR, FREF, REF, SETM, OPO, AL, MJA, OTQ, IP, UV, TRQ, IPS, RNT, ONT, THM, THC, WAC, WAF, FR, OHF, LOC, LOC2, OL, OL2, BRK, BER, CON, ZS, DSE, PDD, POK, PCMP, OD, FBV, OD2, FBV2, NDc, Ai1Dc, Ai2Dc, Ai3Dc, WCAi1, WCAi2, WCAi3, LOG1, LOG2, LOG3, LOG4, LOG5, LOG6, LOG7, MO1, MO2, MO3, MO4, MO5, MO6, MO7, EMFC, EMBP, WFT, TRA, LBK, OVS, AC0, AC1, AC2, AC3, OD3, FBV3, OD4, FBV4, and SEE
	Input terminal variable	X(00) to X(10)/11 points
	Output terminal variable	Y(00) to Y(06)/7 points
	Internal user contact	UB(0) to UB(15)/16 points
	Timer output contact	TD(0) to TD(15)/16 points
	Timer counter variable	TC(0) to TC(15)/16 points
	Analog input terminal variable	XA(0): 0 to 10 V/0 to 20 mA (Ai1 terminal) XA(1): 0 to 10 V/0 to 20 mA (Ai2 terminal) XA(2): −10 to 10 V (Ai3 terminal)
	Analog output terminal variable	YA(0) to YA(2) (Allocated to FM, Ao1, Ao2 terminals)

3

Operation Procedure for DriveProgramming

This section describes the operation procedure of the DriveProgaramming, related parameters, and program structures.

3-1 Operation Procedure

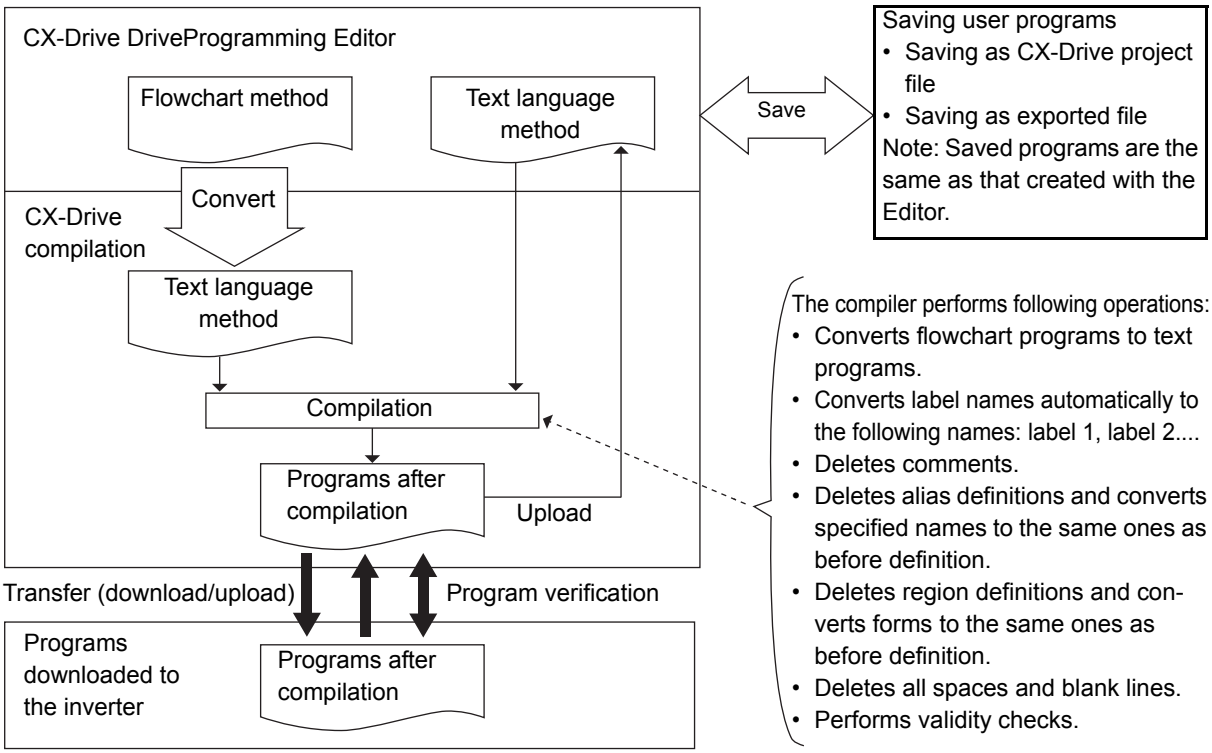
The following figure shows the flow of procedure from programming to executing programs with the DriveProgramming.

Item	Reference
Programming	P. 3-3
↓	
Compiling Programs	P. 3-3
↓	
Downloading Programs	P. 3-4
↓	
Selecting Driveprogramming Functions	P. 3-4
↓	
Starting Programs	P. 3-4

Create DriveProgramming programs by the DriveProgramming Editor in the Inverter/Servo support tool, CX-Drive.

The following figure shows the flow of procedure from programming to transferring (downloading) to the inverter.

For details on operation and other information, refer to *Section 4 DriveProgramming Editor*.



Programming

You can create user programs in the flowchart method or the text language method. It is also possible to select between two methods for each task or subroutine.

Use the DriveProgramming Editor in CX-Drive to input user programs.

DriveProgramming Editor consists of the DriveProgramming area, Toolbox window, Block Parameter window, Properties window, and Output window.

- In the DriveProgramming area, you can create programs in the flowchart method or text language method.
- The Toolbox window displays the command blocks in categories.
- In the Block Parameter window, you can set parameters that are used when the program execution is started.
- In the Properties window, you can edit the properties of a block which is currently selected in flowchart.
- The Output window displays compilation errors and warnings after a compilation is finished.

Compiling Programs

Programs created in the DriveProgramming area are compiled and converted into the final "programs after compilation". Then, the programs are transferred (downloaded) to the inverter.

The compiler performs checks for the items such as program validity, program syntax, parameter input limitation and maximum number of steps. If there is any input which is not permitted, the compilation is stopped and an error message is displayed.

The compiler also performs the operations as shown below, and creates the final "programs after compilation". Therefore, if you transfer (upload) the program once saved in the inverter to the CX-Drive, the program which is read out is the "program after compilation". While its operation is the same as before compilation, its form and contents are partially different.

- Converts flowchart programs to text programs.
- Automatically converts the label names specified in the program to the following names: label 1, label 2...
- Deletes comments entered in the program.
- Deletes alias definitions and converts specified names to the same ones as before definition.
- Deletes region definitions and converts forms to the same ones as before definition.
- Deletes all spaces and blank lines in the program.
- Performs validity checks.

Precautions for Correct Use

- The specified comments, alias definitions and region definitions are deleted when a compilation is performed for program conversion or transferring (downloading) to the inverter.

  To save those contents, save the program before you execute program conversion or you transfer (download) the program to the inverter.

  You can save the program by saving the whole project in the CX-Drive or export file of the program.
- Program verification means a comparison between "program after compilation" and the program inside the inverter. Therefore, comments, alias definitions, region definitions, etc. are not verified.

Downloading Programs

Download compiled programs to the inverter and save them in the EEPROM of the inverter.

You can start programs saved in the EEPROM after turning on the power supply, without using the tool (CX-Drive).

Selecting DriveProgramming Functions

Set EzSQ function enable (UE-02) to Enabled (01 or 02) to use the DriveProgramming function. You can change the EzSQ function enable (UE-02) even when the operation is in progress.

Parameter No.	Function name	Data	Description
UE-02	EzSQ function enable	00: Disabled	Disables the DriveProgramming function. Programs are not executed. If you change the setting to 00 (Disabled) during program execution, the program will be stopped.
		01: [PRG] terminal	Starts the DriveProgramming when the input terminal*1 which is set to 99 (PRG) is turned ON.
		02: Always	Starts the DriveProgramming automatically after the inverter power supply is turned on. If you change the setting to 02 (Enabled) while the program is stopped, the program will be started.

*1. Input terminals are for 1 to 9, A and B.

Starting Programs

When the EzSQ function enable (UE-02) is set to 01 ([PRG] terminal), set one of input 1 to 9, A and B Selection to 99 (PRG). When the EzSQ function enable (UE-02) is set to 01 ([PRG] terminal), the program is started via input terminal with this setting. The program execution continues while the PRG terminal is ON and stops when the terminal is turned OFF.

When the EzSQ function enable (UE-02) is set to 02, the program is started right after the setting is completed. The program will also be started automatically at next power on.

Once the program reaches “end” command after it was started and a series of processes was completed, the program is not executed unless it is restarted.

To repeat the program, create a loop program so that the program does not reach “end” command.

The downloaded DriveProgramming program is saved in the EEPROM of the inverter. Therefore, after downloading, you can start the program without using the support tool.

Parameter No.	Function name	Data	Description
CA-01 to CA-11	Input terminal 1 to 9, A and B	99: PRG Starting of EzSQ program	When the EzSQ function enable (UE-02) is set to 01 ([PRG] terminal: Start/stop via the input PRG terminal), the program is started via input terminal with this setting.

3-2 Parameters Related to DriveProgramming

This section describes the inverter parameters that are related to the DriveProgramming.

Initializing Programs

To initialize the DriveProgramming program downloaded to the inverter, select 04 (Trip history + parameters + DriveProgramming) for the Initialize Mode selection (Ub-01), and execute initialization by Initialize Enable (Ub-05).

Parameter No.	Function name	Data	Description
Ub-01	Initialize Mode selection	00: Disabled 01: Trip history 02: Parameter initialization 03: Trip history + parameters 04: Trip history + parameters + DriveProgramming 05: Other than terminal function 06: Other than communication function 07: Other than terminal&communication functions 08: Only DriveProgramming	Select 04 to initialize the DriveProgramming program as well.

RUN Command Selection Setting

Use FW (forward) and RV (reverse) variables to control RUN commands through the DriveProgramming program. Be sure to set the RUN Command Selection (A002/A202) to 01 (Control circuit terminal block) when you use FW or RV variable.

Parameter No.	Function name	Data	Description
AA111 /AA211	Run-command input source selection	00: [FW]/[RV] terminal 01: 3 wire 02: RUN key on operator LCD Operator 03: RS485 04: Option 1 05: Option 2 06: Option 3	Set operation commands to 00 (FW/RV terminal) when the inverter starts/stops by variables FW/RV.
CA-01 to CA-11	Input terminal 1 to 9, A and B	99: PRG Starting of EzSQ program	When the EzSQ function enable (UE-02) is set to 01 ([PRG] terminal: Start/stop via the input PRG terminal), the program is started via input terminal with this setting.

Setting Inverter I/O Functions

In the DriveProgramming, you can use the inverter's I/O functions (I/O terminal and analog I/O) as the I/O functions of the program.

To use the I/O functions, it is necessary to set each I/O function according to the purpose.

This section describes how to set I/O functions for the DriveProgramming.

By configuring the following settings, you can control I/O functions by the function variables of the DriveProgramming.

In the case that you use the functions for purpose other than DriveProgramming I/O functions, refer to the High-function General-purpose Inverter 3G3RX2 Series User's Manual (I620).

Parameter No.	Function name	Data	Description
AA101	Main speed input source selection, 1st-motor	14: Program function	Set parameters to 14 (program function) when frequency is set to variable SET-Freq.*a
CA-01 to CA-11	Input terminal 1 to 9, A and B	86 to 96: MI1 to MI11	PRG: EzSQ Function PRG Terminal*b
CC-01 to CC-07	Output terminal 11 to 15 function	69 to 75: MO1 to MO7	MO1 to MO7 General-purpose output 1 to 7
Cd-03	FM monitor output selection	Cd-03	YA(0): General-purpose analog output
Cd-04	Ao1 monitor output selection	Cd-04	YA(1): General-purpose analog output
Cd-05	Ao2 monitor output selection	Cd-05	YA(2): General-purpose analog output
AC-01	Acceleration/ Deceleration Time input selection	04: DriveProgramming	When acceleration/deceleration time is set to variables ACCEL and DECEL, set Main speed input source selection, 1st-motor [AA101] to 14 (Program function).

*a. Set main speed input source selection [AA101] and Run-command input source selection [AA111] to other than the values mentioned earlier. That enables LCD operator and analog signal to give speed/operation command.

*b. Only when EzSQ function enable is set to PRG terminal start ([UE-02=01), assign 099 [PRG] to the input terminal.

Precautions for Correct Use

- For Input terminal 1 to 9, A and B, when MI1 to MI11 (General-purpose inputs 1 to 11) is selected, Input terminal [1 to 9, A,B] active state (CA-21 to CA-31) is enabled.
- For Output terminal 11 to 15, relay output terminal 16 and relay output terminal AL selection, when MO1 to MO7 (General-purpose output 1 to 7) is selected, Output terminal [11 to 15] active state (CC-11 to CC-15), Output terminal [16] active state and Output terminal [AL] active state (CC-16, CC-17) are enabled.
- In the DriveProgramming, the analog I/O functions are allocated to XA(0) to XA(2), and YA(0) to YA(2). You can monitor the analog I/O status in the programs by using these function variables regardless of the settings for A101, Cd-03 to Cd-05.
- When MI1 to MI11 (General purpose inputs 1 to 11) is not set for Input terminal [1 to 9, A B] function on the program for DriveProgramming, states of the input signal cannot be monitored. Also, when MO1 to MO7 (General purpose outputs 1 to 7) is not set for Output terminal [11 to 15] function and Relay output terminal [16, AL] function on the program for DriveProgramming, states of the output signal cannot be monitored.

Monitor Function of DriveProgramming

The following functions are provided to monitor the status of the DriveProgramming.

Parameter No.	Function name	Data	Description
db-01	Program download monitor	00: Without a program 01: With a program	00: (Program is not downloaded.) 01: (Program is downloaded.)
db-02	Program No. monitor	0 to 9999	Transferred program No. is displayed.
db-03 to db-07	Program counter	1 to 1024	While each task (Task 1 to Task 5) is executed, line number is monitored.
db-08	User monitor 0	-2147483647 to 2147483647	Data output to Umon (00) to Umon (04) in a program is monitored. db-08←Umon (00) db-10←Umon (01) db-12←Umon (02) db-14←Umon (03) db-16←Umon (04)
db-10	User monitor 1	-2147483647 to 2147483647	
db-12	User monitor 2	-2147483647 to 2147483647	
db-14	User monitor 3	-2147483647 to 2147483647	
db-16	User monitor 4	-2147483647 to 2147483647	
db-18	Analog output monitor YA0	0 to 10000	Data output to YA (00) to YA (02) in a program is monitored. YA0 db-18←YA (00) YA1 db-19←YA (01) YA2 db-20←YA (02)
db-19	Analog output monitor YA1	0 to 10000	
db-20	Analog output monitor YA2	0 to 10000	

User Parameters of DriveProgramming

For user parameters of DriveProgramming, 64 unsigned 1-word variables and 16 signed 2-word variables are provided.

Use these parameters for various purposes such as program initial data setting, parameter for adjustment, and saving calculation results.

Parameter No.	Function name	Data	Description
UE-10	EzSQ user parameter U(00)	0 to 65535	• These user parameters corresponds to the function variables U(00) to U(63). • You can change the data by using the LCD operator. The changed data is saved in the EEPROM.
UE-11	EzSQ user parameter U(01)	0 to 65535	
UE-12	EzSQ user parameter U(02)	0 to 65535	
UE-13	EzSQ user parameter U(03)	0 to 65535	
UE-14	EzSQ user parameter U(04)	0 to 65535	
UE-15	EzSQ user parameter U(05)	0 to 65535	
UE-16	EzSQ user parameter U(06)	0 to 65535	
UE-17	EzSQ user parameter U(07)	0 to 65535	
UE-18	EzSQ user parameter U(08)	0 to 65535	
UE-19	EzSQ user parameter U(09)	0 to 65535	
UE-20	EzSQ user parameter U(10)	0 to 65535	
UE-21	EzSQ user parameter U(11)	0 to 65535	
UE-22	EzSQ user parameter U(12)	0 to 65535	
UE-23	EzSQ user parameter U(13)	0 to 65535	
UE-24	EzSQ user parameter U(14)	0 to 65535	
UE-25	EzSQ user parameter U(15)	0 to 65535	
UE-26	EzSQ user parameter U(16)	0 to 65535	
UE-27	EzSQ user parameter U(17)	0 to 65535	
UE-28	EzSQ user parameter U(18)	0 to 65535	
UE-29	EzSQ user parameter U(19)	0 to 65535	
UE-30	EzSQ user parameter U(20)	0 to 65535	
UE-31	EzSQ user parameter U(21)	0 to 65535	
UE-32	EzSQ user parameter U(22)	0 to 65535	
UE-33	EzSQ user parameter U(23)	0 to 65535	
UE-34	EzSQ user parameter U(24)	0 to 65535	
UE-35	EzSQ user parameter U(25)	0 to 65535	
UE-36	EzSQ user parameter U(26)	0 to 65535	
UE-37	EzSQ user parameter U(27)	0 to 65535	
UE-38	EzSQ user parameter U(28)	0 to 65535	
UE-39	EzSQ user parameter U(29)	0 to 65535	
UE-40	EzSQ user parameter U(30)	0 to 65535	
UE-41	EzSQ user parameter U(31)	0 to 65535	
UE-42	EzSQ user parameter U(32)	0 to 65535	
UE-43	EzSQ user parameter U(33)	0 to 65535	
UE-44	EzSQ user parameter U(34)	0 to 65535	
UE-45	EzSQ user parameter U(35)	0 to 65535	
UE-46	EzSQ user parameter U(36)	0 to 65535	
UE-47	EzSQ user parameter U(37)	0 to 65535	
UE-48	EzSQ user parameter U(38)	0 to 65535	
UE-49	EzSQ user parameter U(39)	0 to 65535	
UE-50	EzSQ user parameter U(40)	0 to 65535	
UE-51	EzSQ user parameter U(41)	0 to 65535	
UE-52	EzSQ user parameter U(42)	0 to 65535	
UE-53	EzSQ user parameter U(43)	0 to 65535	
UE-54	EzSQ user parameter U(44)	0 to 65535	

Parameter No.	Function name	Data	Description
UE-55	EzSQ user parameter U(45)	0 to 65535	
UE-56	EzSQ user parameter U(46)	0 to 65535	• These user parameters corresponds to the function variables U(00) to U(63). • You can change the data by using the LCD operator. The changed data is saved in the EEPROM.
UE-57	EzSQ user parameter U(47)	0 to 65535	
UE-58	EzSQ user parameter U(48)	0 to 65535	
UE-59	EzSQ user parameter U(49)	0 to 65535	
UE-60	EzSQ user parameter U(50)	0 to 65535	
UE-61	EzSQ user parameter U(51)	0 to 65535	
UE-62	EzSQ user parameter U(52)	0 to 65535	
UE-63	EzSQ user parameter U(53)	0 to 65535	
UE-64	EzSQ user parameter U(54)	0 to 65535	
UE-65	EzSQ user parameter U(55)	0 to 65535	
UE-66	EzSQ user parameter U(56)	0 to 65535	
UE-67	EzSQ user parameter U(57)	0 to 65535	
UE-68	EzSQ user parameter U(58)	0 to 65535	
UE-69	EzSQ user parameter U(59)	0 to 65535	
UE-70	EzSQ user parameter U(60)	0 to 65535	
UE-71	EzSQ user parameter U(61)	0 to 65535	
UE-72	EzSQ user parameter U(62)	0 to 65535	
UE-73	EzSQ user parameter U(63)	0 to 65535	
UF-02	EzSQ user parameter UL(00)	-2147483647 to 2147483647	• They are user parameters supporting function variables UL (00) to UL (15). • You can change the data by using the LCD operator. The changed data is saved in the EEPROM.
UF-03			
UF-04	EzSQ user parameter UL(01)	-2147483647 to 2147483647	
UF-05			
UF-06	EzSQ user parameter UL(02)	-2147483647 to 2147483647	
UF-07			
UF-08	EzSQ user parameter UL(03)	-2147483647 to 2147483647	
UF-09			
UF-10	EzSQ user parameter UL(04)	-2147483647 to 2147483648	
UF-11			
UF-12	EzSQ user parameter UL(05)	-2147483647 to 2147483649	
UF-13			
UF-14	EzSQ user parameter UL(06)	-2147483647 to 2147483650	
UF-15			
UF-16	EzSQ user parameter UL(07)	-2147483647 to 2147483651	
UF-17			
UF-18	EzSQ user parameter UL(08)	-2147483647 to 2147483652	
UF-19			
UF-20	EzSQ user parameter UL(09)	-2147483647 to 2147483653	
UF-21			
UF-22	EzSQ user parameter UL(10)	-2147483647 to 2147483654	
UF-23			
UF-24	EzSQ user parameter UL(11)	-2147483647 to 2147483655	
UF-25			
UF-26	EzSQ user parameter UL(12)	-2147483647 to 2147483656	
UF-27			
UF-28	EzSQ user parameter UL(13)	-2147483647 to 2147483657	
UF-29			
UF-30	EzSQ user parameter UL(14)	-2147483647 to 2147483658	
UF-31			
UF-32	EzSQ user parameter UL(15)	-2147483647 to 2147483659	
UF-33			

3-3 Program Structure

In the DriveProgramming for 3G3RX2 Series Inverter, you can create a maximum of five tasks.

The created tasks are processed in parallel.

By dividing one application into several processes and allocating them to multiple tasks, you can adjust execution condition, execution order, etc. for each process. Furthermore, parallel task processing can provide shorter processing time.

3-3-1 Tasks

Task is a unit of program executed in the DriveProgramming.

For 3G3RX2 Series Inverter, you can create a program which consists of up to five tasks.

- All tasks are started simultaneously.
- In the order of task from 1 to 5, one command of each task (one line of the "program after compilation") is executed in 1-ms or 2-ms processing time (selected by UE-01).
- All function variables such as user parameters are shared among the tasks. For transmission of information between tasks, use the user parameters, internal user contacts, etc.
- At the end of 1-ms or 2-ms processing time, the operation result of each task is reflected to the inverter operation and external output, etc. At the same time, the status of the inverter and external input terminals are read in.
- When the "end" command is executed, the task is completed and waits for the next start.

3-3-2 Subroutines

Subroutine is a separated program processing executed only when it is called.

Subroutines are useful to organize your program into parts that you can execute multiple times in the same task or reuse in other programs.

In the DriveProgramming, it is necessary to insert subroutines into each task. They cannot be shared among the tasks.

It is possible to call a subroutine from another subroutine (nesting).

However, the maximum nesting of subroutines is eight levels.

3-3-3 Task Processing

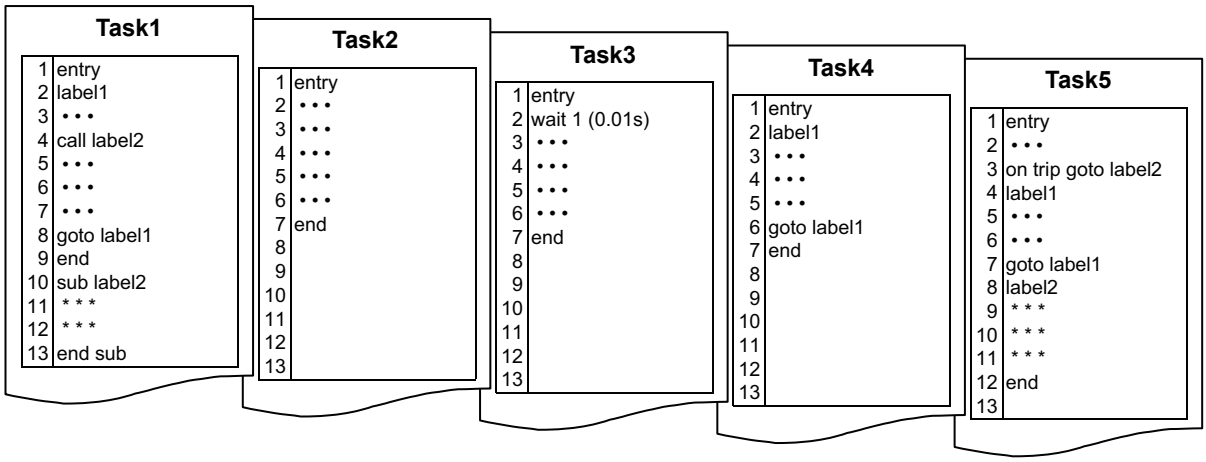
Up to five tasks are started simultaneously, and one command of each task (one line of the "program after compilation") is executed in 1-ms or 2-ms processing time (selected by UE-01).

- All tasks are started simultaneously.
- In the order of task from 1 to 5, one command of each task (one line of the "program after compilation") is executed in 1-ms or 2-ms processing time (selected by UE-01).
- All function variables such as user parameters are shared among the tasks.
- At the end of 1-ms or 2-ms processing time (selected by UE-01), the operation result of each task is reflected to the inverter operation and external output, etc. At the same time, the status of the inverter and external input terminals are read in.
- As shown in Task 1, if a subroutine is called by the “call” command, the statement of the subroutine will be executed from the next step of processing.
- As shown in Task 2, if the “end” command is executed, the task will be finished and wait for the next start.

- As shown in Task 3, if the “wait” command is executed, the task will be repeatedly executed until a certain condition is met.
- As shown in Task 4, if you create a loop by using the “goto” command, the task process will be continuously repeated.
- As shown in Task 5, if the “on trip goto” command is executed once, the processing data will be stored in the inverter.

  The program jumps to the “goto” destination immediately after a trip occurred.

● Programs After Compilation (Example)



● Program Processing Flow

Processing time [ms]	Task1 Line No.	Task1	Task2 Line No.	Task2	Task3 Line No.	Task3	Task4 Line No.	Task4	Task5 Line No.	Task5
2	1	entry	1	entry	1	entry	1	entry	1	entry
4	2	label1	2	- - -	2	wait 1 (0.01s)	2	label1	2	- - -
6	3	- - -	3	- - -	2	wait 1 (0.01s)	3	- - -	3	on trip goto label2
8	4	call label2	4	- - -	2	wait 1 (0.01s)	4	- - -	4	label1 [Trip occurred]
10	10	sub label2	5	- - -	2	wait 1 (0.01s)	5	- - -	5	label2
12	11	* * *	6	- - -	2	wait 1 (0.01s)	6	goto label1	6	* * *
14	12	* * *	7	end	3	- - -	2	label1	7	* * *
16	13	end sub			4	- - -	3	- - -	2	* * *
18	5	- - -			5	- - -	4	- - -	3	end
20	6	- - -			6	- - -	5	- - -	8	
22	7	- - -			7	end	6	goto label1	9	
24	8	goto label1					2	label1	10	
26	2	label1					3	- - -	11	
28	3	- - -					4	- - -	12	
30	4	call label2					5	- - -		
32	10	sub label2					6	goto label1		
34	11	* * *					2	label1		
36	12	* * *					3	- - -		
38	13	end sub					4	- - -		

Label2 is executed due to occurrence of an inverter trip.

Program Counter (db-03) display

3-3-4 DriveProgramming Start/Stop and Task Operation

You set the start/stop method of the DriveProgramming programs in the EzSQ function enable (UE-02).

● Start

- When the EzSQ function enable (UE-02) is set to 01 ([PRG] terminal),
  The program starts when the input terminal set to PRG is turned ON.
- When the EzSQ function enable (UE-02) is set to 02 (Always),
  The program starts when the power supply for the inverter is turned ON.

● Stop

- When the EzSQ function enable (UE-02) is set to 01 ([PRG] terminal),
  The program stops when the input terminal set to PRG is turned OFF.
- When the EzSQ function enable (UE-02) is set to 02 (Always),
  The program stops when the power supply for the inverter is turned OFF.

Also, regardless of the UE-02 setting, the DriveProgramming program stops if any of E043 to E045 trips is detected due to a program error, or when the “end” command is executed in all tasks.

Precautions for Safe Use

When the DriveProgramming program is stopped, the output terminal controlled by the DriveProgramming retains the status before the program stop.

For this reason, configure the system so that the stop of the DriveProgramming program in the inverter can be detected by the DriveProgramming start signal and the alarm (trip) signal, and the inverter's peripheral devices can be stopped safely.

Precautions for Correct Use

The following are the status of function variables when the DriveProgramming program is stopped. Take necessary measures in consideration of each status.

- For the output terminals (the inverter's actual output terminals), the status before the program stop is retained.
  However, the output terminals that are not set to MO1 to MO6 (general-purpose outputs of the DriveProgramming) operate as the inverter's normal output terminals.
- When the DriveProgramming function is selected for the frequency reference, acceleration/deceleration time or analog outputs, the set values for these functions before the program stop are retained.
- The data of the user parameter variables, internal user variables and internal user contacts before the program stop is retained.
- The status of the inverter's actual input terminals such as the input terminal and analog input terminals is not retained but always updated.
- The data of the output variables (function bits such as RUN, FA1 and AL) and inverter monitor variables is not retained but always updated according to the status of the inverter.
- Only the DriveProgramming's input variables (function bits such as FW, RV and CF1) and timer variables are cleared at the same time as the program stop, and all data are changed to zero.

When the DriveProgramming function is started/stopped through the CX-Drive, the value of the inverter parameter, the EzSQ function enable (UE-02) is changed.

Take the following measures after you started/stopped the DriveProgramming function through the CX-Drive.

- Turn ON the power supply for the inverter again and return the value of UE-02 to that saved in the EEPROM.
- Do not perform the following EEPROM saving operations before you turn ON the power supply for the inverter.
  - Transferring (downloading) a part of parameters from the CX-Drive.
  - Issuing the “enter” command via the Modbus communication or a communication option.

When the DriveProgramming program is started, five tasks are started simultaneously.

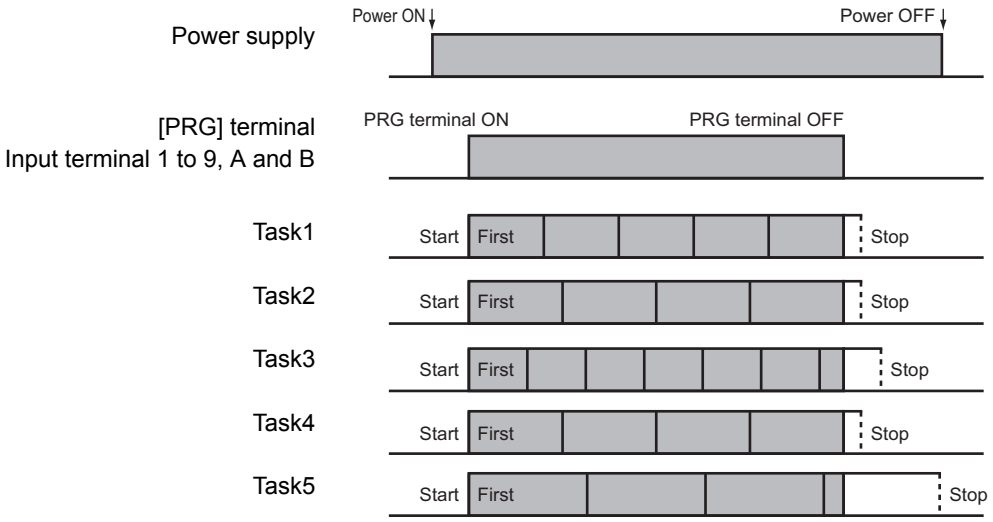
If you create a loop construction in a task by using commands such as the “goto”, the task will repeat the loop after it is started.

When the DriveProgramming program is stopped, five tasks are stopped simultaneously.

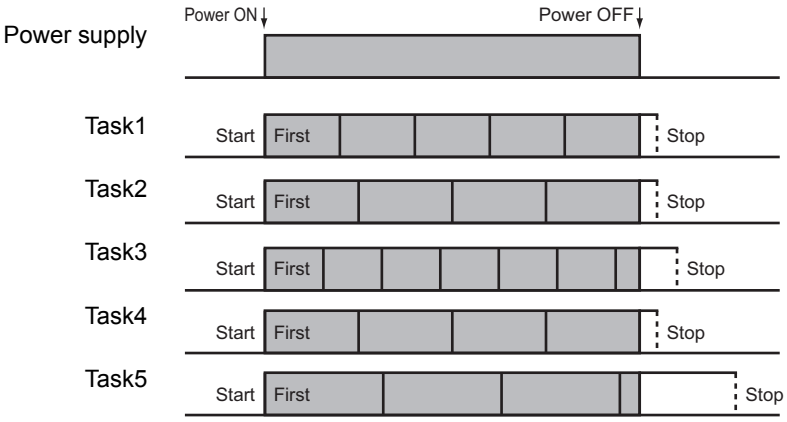
If the program is stopped while the Motor is running, the Motor will be stopped according to the setting for the STOP mode selection (AA115) (deceleration stop or free-run stop).

The task operations for different settings are shown below.

● When the EzSQ function enable (UE-02) is set to 01 ([PRG] terminal):



● When the EzSQ function enable (UE-02) is set to 02 (Always):



3-3-5 DriveProgramming Restart

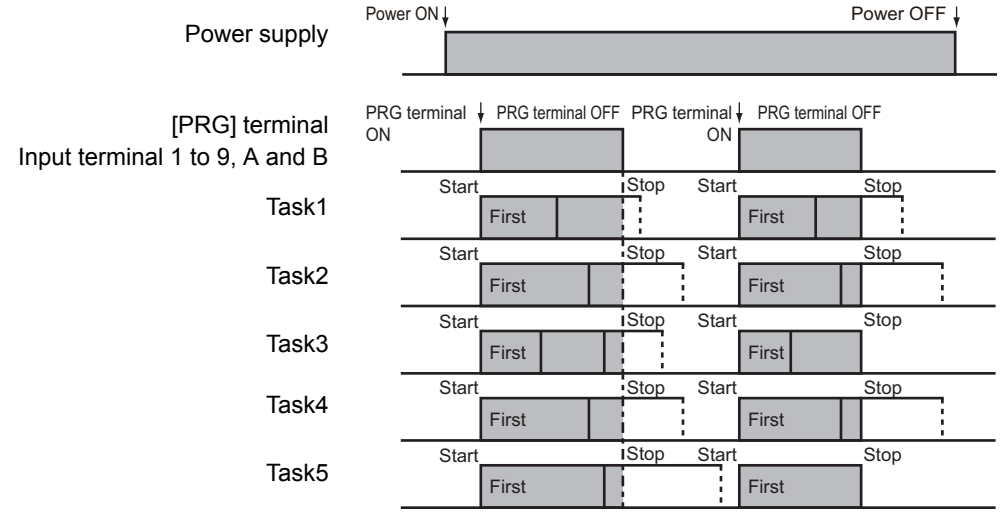
You can restart the stopped program by performing the following operations. When the program is restarted, all tasks are started simultaneously from the beginning.

- When the DriveProgramming Function Selection (UE-02) is set to 01 ([PRG] terminal): a reset input via the control circuit terminal while the PRG terminal is ON, or turning ON the PRG terminal again.
- When the DriveProgramming Function Selection (UE-02) is set to 02 (Always): a reset input via the control circuit terminal, or turning ON the power supply for the inverter again.

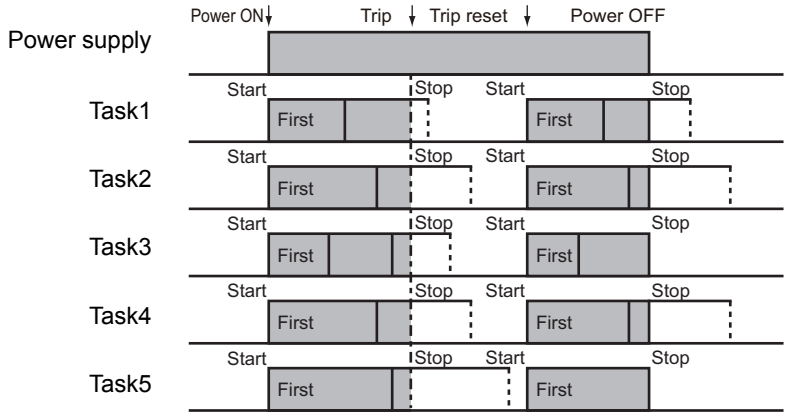
Precautions for Correct Use

- You cannot restart the DriveProgramming by pressing the STOP/RESET key of the LCD Operator. Set data 28 (RS:Reset) to input terminal 1 to 9, A and B. After that, turn ON that terminal.
- When the Reset mode selection (CA-72) is set to 00 (On to Release Trip) or 01 (Off to Release Trip), the DriveProgramming can be restarted by a reset input even when there is no trip.

● Restart when the EzSQ function enable (UE-02) is set to 01 ([PRG] terminal):



● Restart after trip when the EzSQ function enable (UE-02) is set to 02 (Always) (without “on trip goto” command):



3-3-6 Task Operation on Trip

Basically, even if the inverter detects a trip during the DriveProgramming operation, the operation is continued. However, if any of E043 to E045 trips related to the DriveProgramming is detected, the operation is stopped.

Or, with the “on trip goto” command, the program can jump to other process after a trip occurred.

With/without “on trip goto”	Error status		
	User trip E050 to E059	DriveProgramming related trip E043 to E045	Other trips
Without	Operation is continued.	Program is stopped.	Operation is continued.
With	After the “on trip goto” command is executed, the program jumps to the specified label and the operation is continued.	Program is stopped.	After the “on trip goto” command is executed, the program jumps to the specified label and the operation is continued.

For details on the trips E043 to E045, refer to *Section 8 Errors and Remedies*.

4

DriveProgramming Editor

This section describes how to start the DriveProgramming Editor, saving and loading data, and details on parts of the Editor.

4-1 Starting DriveProgramming Editor

To create DriveProgramming programs, you use the DriveProgramming Editor in the support tool for the Inverter/Servomotor, CX-Drive.

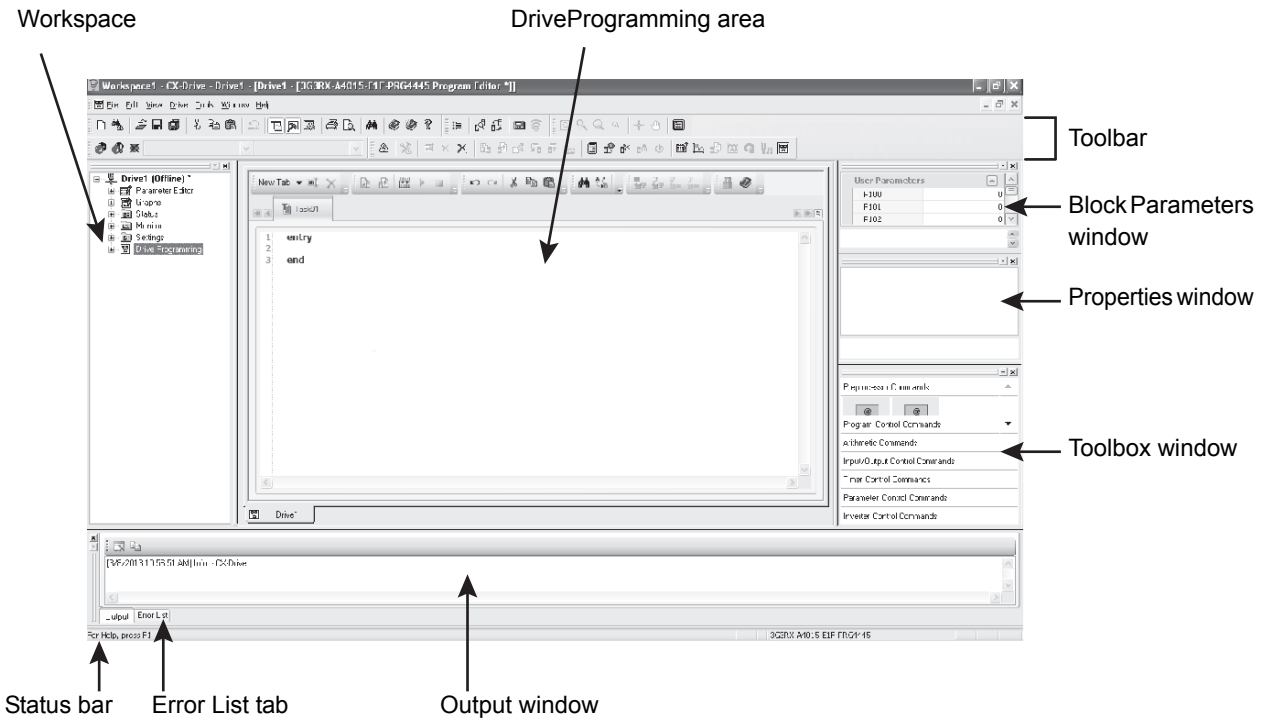
This section explains the configuration of the DriveProgramming Editor in the CX-Drive and the operations you carry out before starting the DriveProgramming Editor.

Precautions for Correct Use

- The DriveProgramming function is included in the following or higher versions of the CX-Drive.
  If the version of your CX-Drive is lower, you need to upgrade the version.
  Version 3.0 or higher (with version 2.9 or lower, operation is not possible.)
- A password is required when you start the DriveProgramming Editor in the CX-Drive for the first time.
  Please contact your OMRON representative.

CX-Drive and DriveProgramming Editor Screen Layout

The DriveProgramming Editor is included as one of the functions of the support tool for the Inverter/Servomotor, CX-Drive. When the DriveProgramming Editor is started, the screen layout of the CX-Drive will be changed as shown below.



Starting CX-Drive

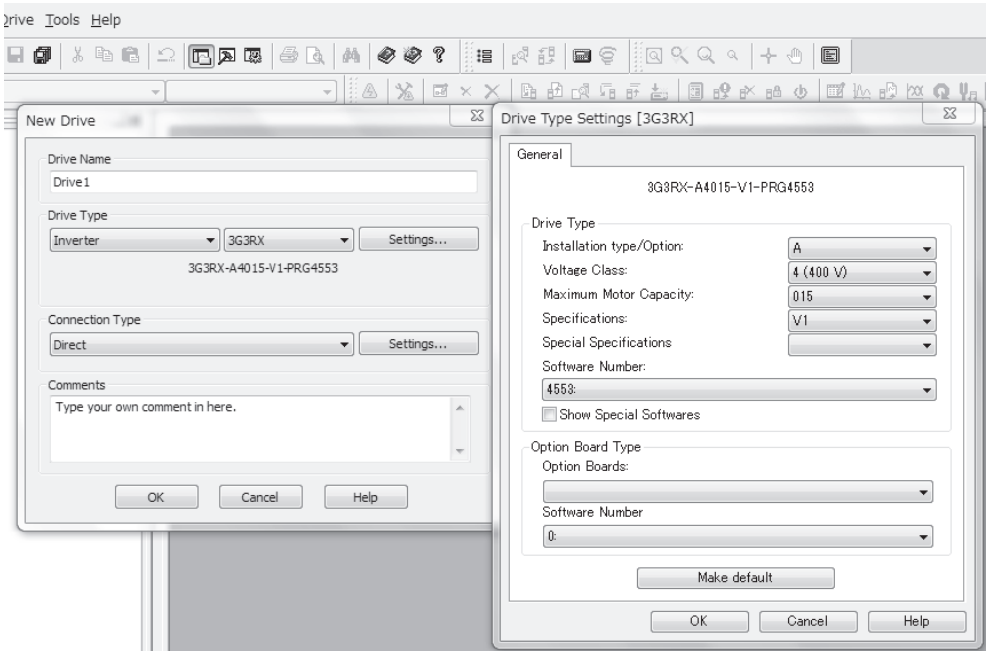
Use the following method to start the CX-Drive: from the Windows [Start] Menu, select [All Programs] - [OMRON] - [CX-One] - [CX-Drive]. Click the CX-Drive icon in the CX-Drive folder.

After the CX-Drive is started, select [File] from the CX-Drive Menu and click [New] to create a new CX-Drive file. The [New Drive] dialog box will appear.

Select 3G3RX2 series from the pull-down list under the [Drive Type]. Click the [Settings] button to the right.

In the [Drive Type Settings] dialog box, set [Installation Type/Option], [Voltage Class], and [Maximum Motor Capacity].

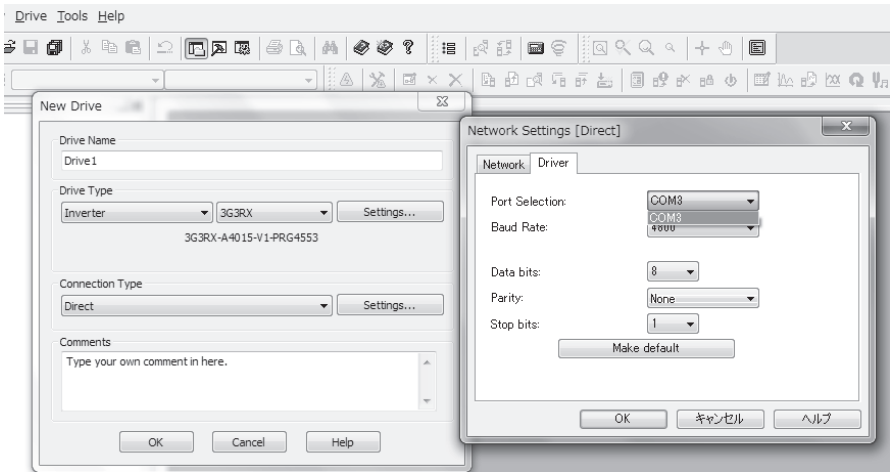
After setting these items, click the [OK] button to close the [Drive Type Settings] dialog box.



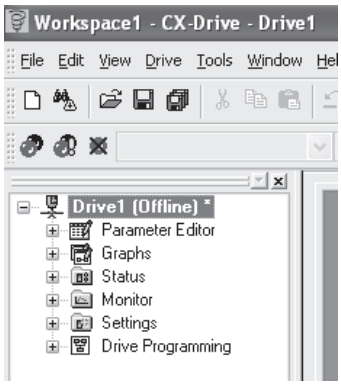
In the [New Drive] dialog box, you set the connection type for the CX-Drive and the inverter.

Under the [Connection Type], select [Direct] and click the [Settings] button to the right.

On the [Driver] tab page, set the [Port Selection] to the port name of the computer on which the CX-Drive is installed.



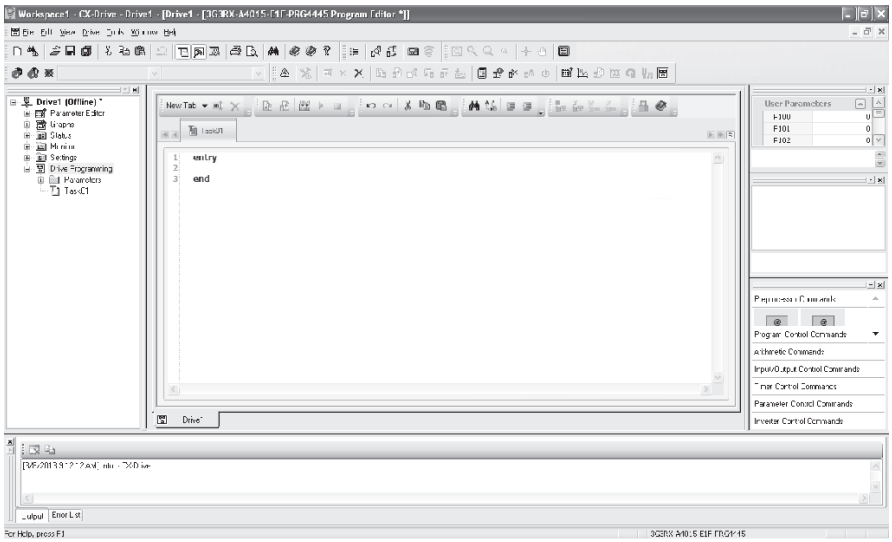
After setting these items, click the [OK] button and close all dialog boxes. The new project is registered in the workspace.



Starting DriveProgramming Editor

There are three ways to display the DriveProgramming Editor:

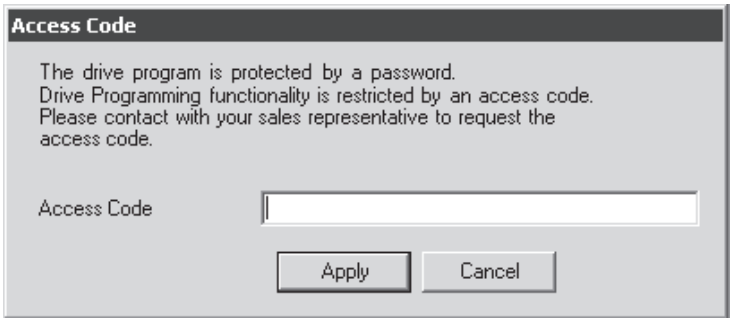
- Double click the [DriveProgramming] in the workspace.
- Click  button on the CX-Drive toolbar.
- From the [Drive] Menu, select [Program] - [Program Editor].



Precautions for Correct Use

To start the DriveProgramming Editor for the first time after you started the CX-Drive, a password is required.

Please contact with your sales representative to request the access code, and enter it in the following dialog box.

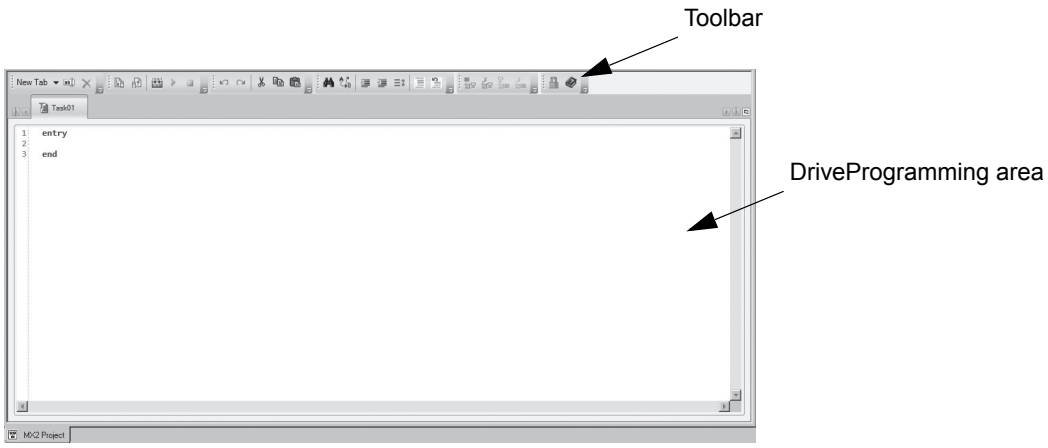


4-2 Parts of DriveProgramming Editor

This section provides the detailed information on each function of the Editor and windows related to the DriveProgramming in the CX-Drive.

4-2-1 DriveProgramming Editor

The DriveProgramming Editor is the main widow for the DriveProgramming function.



This window consists of the toolbar in which common commands are included and the DriveProgramming area in which the program is displayed as text or flowchart.

4-2-2 Toolbar

The DriveProgramming Editor provides the following commands:

● Common commands

Command	Icon	Description
New Task (flowchart)		Creates a new flowchart task for the program. Task is a unit of program executed in the DriveProgramming.
New Task (text)		Creates a new text task for the program. Task is a unit of program executed in the DriveProgramming.
New Subroutine (flowchart)		Creates a new flowchart subroutine. A subroutine is a part of the program which is executed only when it is called.
New Subroutine (text)		Creates a new text subroutine. A subroutine is a part of the program which is executed only when it is called.
Rename Current Task		Renames the current task or subroutine.
Delete Current Task		Deletes the current task or subroutine.
Undo		Undoes the latest change.
Redo		Redoes the undone operation.

Command	Icon	Description
Transfer to Drive		Compiles the program and, if there are no errors, transfers (downloads) it to the drive (inverter).
Transfer from Drive		Transfers (uploads) the program from the drive (inverter) to the DriveProgramming Editor.
Compile		Compiles the program in the DriveProgramming area. Compile errors and warnings that occurred in the program will be displayed on the Error List tab page in the Output window.
Start		Starts the program in the inverter. The CX-Drive will first compare the program in the inverter with the program in the DriveProgramming area. The program will be started only when they are the same. If they differ, the program will not be started. This starting method for the program is enabled only when the CX-Drive is connected. To start the program with a stand-alone inverter, be sure to set the EzSQ Function Selection (UE-02).
Stop		Stops the program in the inverter. This operation is performed independently of the program on the DriveProgramming area in the CX-Drive.
Set Program Number		Sets a program number from 0 to 9,999. After the program is transferred (downloaded) to the drive (inverter), the program number is updated and you can see that by the Program Number Monitor (DriveProgramming) (db-02).
Set Password		Enables to set, change or delete the program password.
Help		Displays the CX-Drive help.

● Commands for the flowchart program

Command	Icon	Description
Zoom in		Increases the zoom level.
Zoom out		Decreases the zoom level.
Zoom Reset		Restores the zoom to its initial value.
Select Mode		Selects one or more blocks[*1] of the program by dragging with the mouse cursor.
Pan Mode		By dragging, moves the field of the view in any direction while keeping the same scale.
Horizontal Align Left		Aligns horizontally the left sides of the currently selected blocks[*1].
Horizontal Align Middle		Aligns horizontally the middles of the currently selected blocks[*1].
Horizontal Align Right		Aligns horizontally the right sides of the currently selected blocks[*1].
Vertical Align Top		Aligns vertically the top sides of the currently selected blocks[*1].
Vertical Align Middle		Aligns vertically the middles of the currently selected blocks[*1].

Command	Icon	Description
Vertical Align Bottom		Aligns vertically the bottoms of the currently selected blocks*1.
Orientation		Selects an orientation for connecting the blocks*1.
Auto-arrange		Arranges the blocks*1 of the flowchart automatically in the currently selected orientation.
Show Contacts		Switches display/hide of the contacts of the blocks*1.
Show		Enables to select a display style of the program from options below. • Text only • Icon only • Icon and text • Name, icon and arguments

*1. A unit of display on the flowchart is called "Block". A block consists of a command, or a command and a label attached to the command.

● Commands for the text program

Command	Icon	Description
Find		Looks for a text in the DriveProgramming Editor.
Replace		Replaces a text in the DriveProgramming Editor.
Increment Indentation		Increases the indentation of the selected text.
Decrement Indentation		Decreases the indentation of the selected text.
Format selected text		Formats automatically the selected text.
Comment selected text		Transforms the selected lines of text to comments.
Uncomment selected text		Uncomments the selected lines of text.

● Program conversion

You can convert a flowchart program to a text program, and vice versa.

When a conversion is performed, the program is once compiled. Note that if a compilation error occurs, the conversion will not be completed.

Also, comments, alias definition, region definition, etc. are deleted when a compilation is performed for conversion. The operation does not change, however, the forms and contents of the program are partially changed.

Command	Icon	Description
Convert Text to Flowchart		Converts current text task/subroutine to flowchart.
Convert whole program to Flowchart		Converts whole program to flowchart.

Command	Icon	Description
Convert Flowchart to Text		Converts current flowchart task/subroutine to text.
Convert whole program to Text		Converts whole program to text.

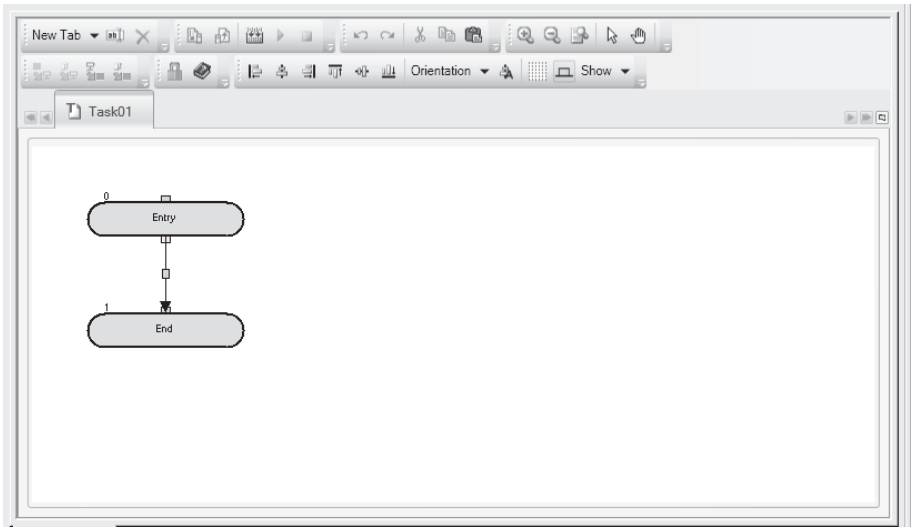
● Shortcut keys

You can use the following keyboard shortcuts in the DriveProgramming area.

Shortcut	Description
Ctrl + X	Cut
Ctrl + C	Copy
Ctrl + V	Paste
Ctrl + Z	Undo
Ctrl + Y	Redo
Ctrl + A	Select all
Ctrl + F	Find and replace
Ctrl + space	Command list
Tab	Select next (flowchart only)
Arrow Keys	Move selected block
Home	Scroll to top (flowchart only)
End	Scroll to bottom (flowchart only)
Page Up	Move the cursor to line head
Page Down	Move the cursor to end of the line
+	Zoom in
−	Zoom out

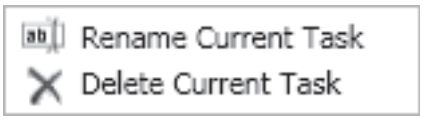
4-2-3 DriveProgramming Area

The DriveProgramming area displays the current design of the program.



This area may have different pages, organized in tabs. Each tab is either a task or a subroutine in flowchart or text.

By right-clicking the tab title, you can delete or rename a task or a subroutine.



Flowchart Program

In the flowchart program method, a unit of display is called "Block".

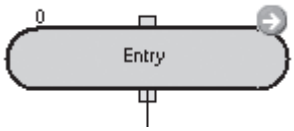
You create a program by placing more than one block in the area and setting interaction between them.

In a flowchart program, when a program is compiled successfully, an icon with a white arrow in a green circle highlights the starting point of each flowchart task.

Also, after the program is compiled successfully, the block number will appear on the upper left of each block.

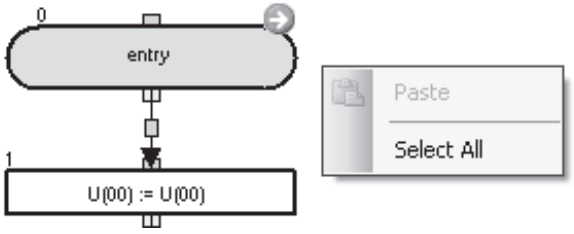
Block numbers are the consecutive numbers starting with "0". A block number is given to each block of the whole program.

The numbers do not match the line numbers of the program converted to text.

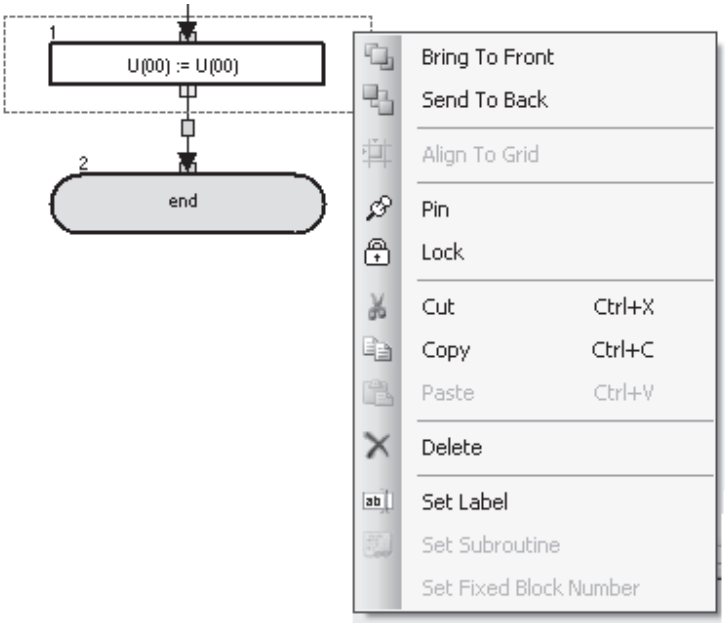


The Output window will indicate if the program is compiled successfully. For programs compiled with errors, a red icon with an exclamation mark identifies the erroneous blocks in the flowchart program. Placing the mouse on the error icon displays the compile error, which you can see in the Error List in the Output window.

If you right-click on a blank area in the flowchart, a popup menu will be displayed. It allows you to paste blocks that you copied last, or to select all the blocks.



If you right-click on a flowchart block, a popup menu with more options will appear.



The following table shows the menu commands available in the flowchart program.

Command	Description
[Bring To Front]	Places the selected block graphically in front of other blocks.
[Send To Back]	Places the selected block graphically in back of other blocks.
[Pin]	Fixes the selected block to its current position in the graph. It will not be moved in drag operations.
[Lock]	Acts like [Pin] and, besides, sets the properties of the block as read-only.
[Cut]	Deletes the selected block and saves it in the clipboard, for further pasting.
[Copy]	Saves the selected block in the clipboard, for further pasting.
[Paste]	Puts the contents previously copied in the clipboard into the design area. Note that you can also paste them as images in other applications.
[Set Label]	Sets the label name for the selected block.
[Set Subroutine]	Sets the subroutine name for the selected block. This menu command is available only for blocks for which you can set subroutines.

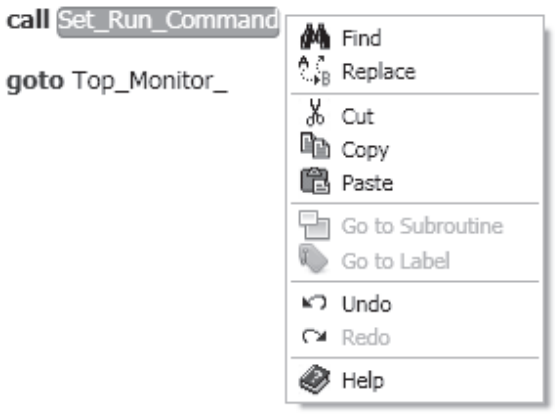
Text Program

In the text program method, you create a program by using text language.

For text programs that were not compiled successfully, the program errors will be displayed in the Output window. The statement with errors will be highlighted with a red line.

```
Dummy_UL01 := A038
```

Right-click the selected text to display a popup menu.



The following table shows the menu commands available in the text program.

Command	Description
[Find]	Looks for the selected text on the program code.
[Replace]	Replaces the selected text on the program code.
[Cut]	Deletes the selected text and saves it in the clipboard, for further pasting.
[Copy]	Saves the selected text in the clipboard, for further pasting.
[Paste]	Puts the contents previously copied in the clipboard into the design area. Note that you can also paste them as images in other applications.
[Go to Subroutine]	Jumps to the selected text subroutine.
[Go to Label]	Jumps to the selected text label.
[Undo]	Undoes the latest change.
[Redo]	Redoes the undone operation.
[Help]	Displays the CX-Drive help.

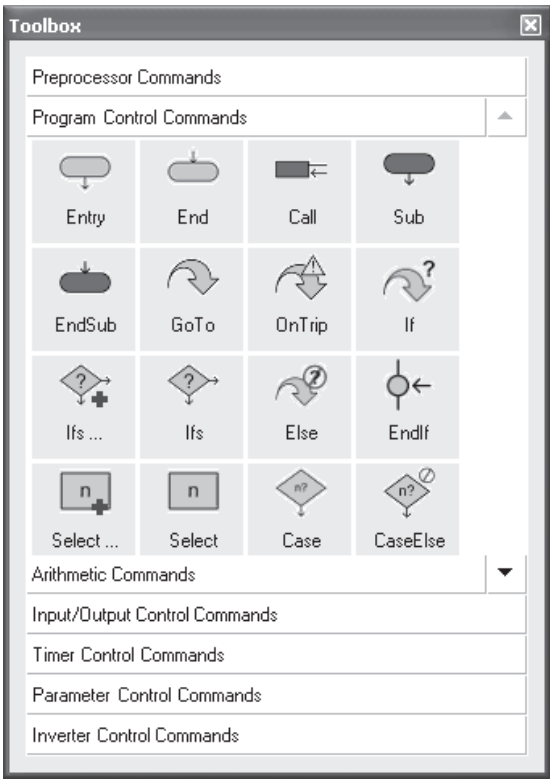
4-2-4 Toolbox Window

The Toolbox window allows you to add blocks to the DriveProgramming area by drag and drop. It displays the blocks supported for a particular command, organized in categories.

The Toolbox is displayed when the DriveProgramming Editor is started. You can also show or hide it by clicking [DriveProgramming] - [Toolbox] in the [View] Menu.

The Toolbox window is displayed by default at the right side of the CX-Drive.

You can separate the window by double-clicking the title bar (wide frame of the window).



You can also select its display style by right-clicking on it with the mouse. Three styles are available: large icons, small icons, and list. In any style, a short help text will be shown when you place the mouse cursor on a block.

Click on any category title to display the blocks which belong to that category.

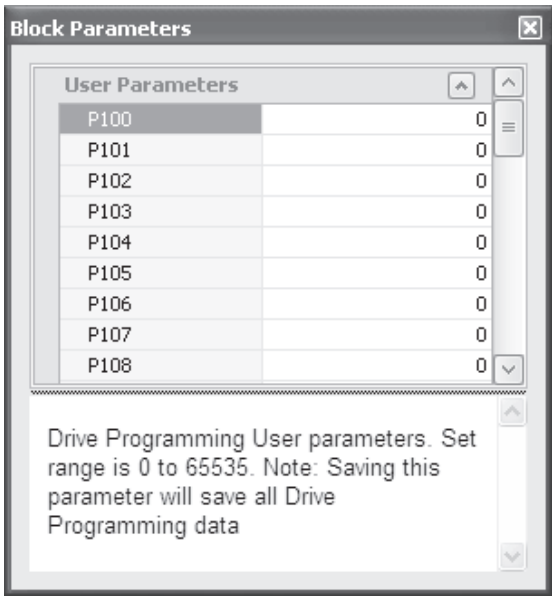
4-2-5 Block Parameters Window

The Block Parameters window allows you to edit DriveProgramming user parameters which act as variables of the program. The displayed parameters are organized in categories.

The Block Parameters is displayed when the DriveProgramming Editor is started. You can also show or hide it by clicking [DriveProgramming] - [Block Parameters] in the [View] Menu.

The Block Parameters window is displayed by default at the right side of the CX-Drive.

You can separate the window by double-clicking the title bar (wide frame of the window).



To change the value of a parameter, place the cursor at its row and click on the edition box to the right of its number. Enter the new value. A warning will be displayed if the entered value exceeds the valid range.

At the lower part of the window, a help text for the user parameters is displayed.

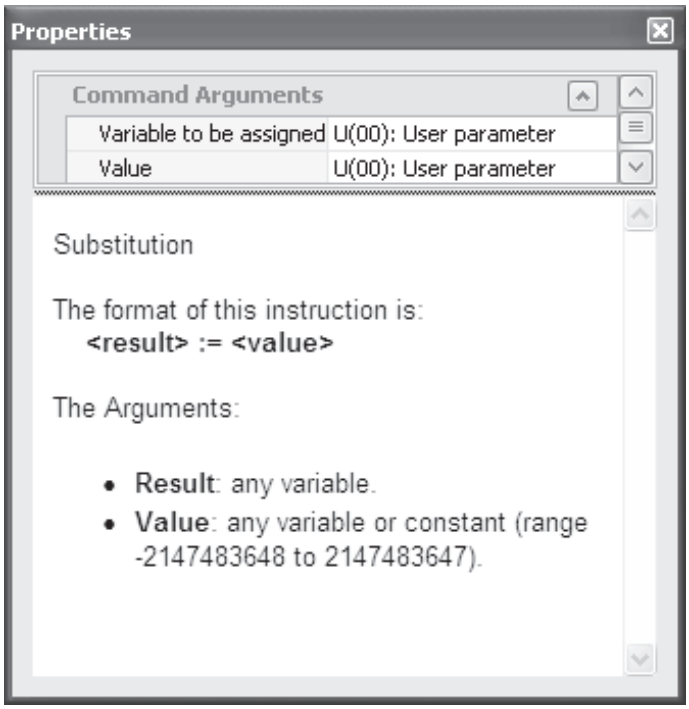
4-2-6 Properties Window

The Properties window allows you to edit the properties of the block which is currently selected in the flowchart program.

The Properties is displayed when the DriveProgramming Editor is started. You can also show or hide it by clicking [DriveProgramming] - [Properties] in the [View] Menu.

The Properties window is displayed by default at the right side of the the CX-Drive.

You can separate the window by double-clicking the title bar (wide frame of the window).



To change one block command argument, select the block in the flowchart and place the cursor on the section where you want to edit in the Properties window.

- If the block argument has options, click on the current value to display  icon at the rightmost side. Click  or double-click the current value to unfold the available options in the pulldown menu.
- If the block argument does not have options, click its current value and enter the new one to change the value. A warning will be displayed if the entered value exceeds the valid range.
- If the block argument can have both an option and a custom value, you can set a value directly by clicking the current value. Also, double-clicking the current value will unfold the available options in the pulldown menu.

4-2-7 Error List Tab in Output Window

The list of errors related to the DriveProgramming is displayed when you click the Error List tab in the Output window.

The error list shows the compilation errors and warnings of the program currently created with the DriveProgramming Editor after it is compiled.

If any compilation errors is shown, the program is not correctly compiled.

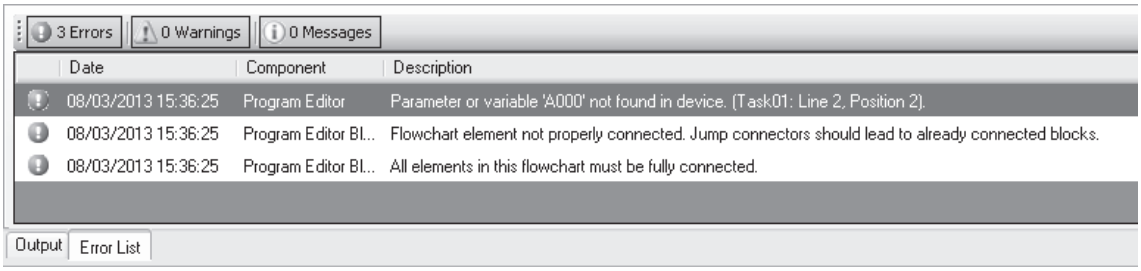
In this case, transferring to the inverter or converting between flowchart and text is not performed.

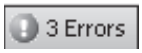
Even in the case of successful compilation, warnings may be displayed to show points to correct in the program.

The error list is updated automatically every time a compilation is completed, and fixed errors and warnings are cleared.

The Output window is displayed by default at the bottom of the CX-Drive.

You can separate the window by double-clicking the title bar (wide frame of the window).





[Errors] button switches the display of errors in the list.

[Warnings] button switches the display of warnings in the list.

[Messages] button switches the display of informative messages in the list.

The messages in the list contain the following information:

Information	Description
Date	Shows the date and time when the error occurred.
Component	Identifies the block with an error.
Name	Shows names or descriptions of the error or warning message.

4-3 Adding, Deleting and Renaming Tasks

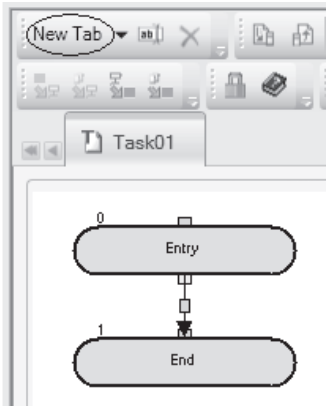
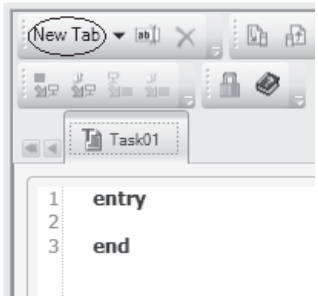
The DriveProgramming Editor will display an empty task by default when it is started from the CX-Drive.

To add a new task, select [New Tab] in the toolbar of the DriveProgramming Editor, and select [New Task (flowchart)] or [New Task (text)].

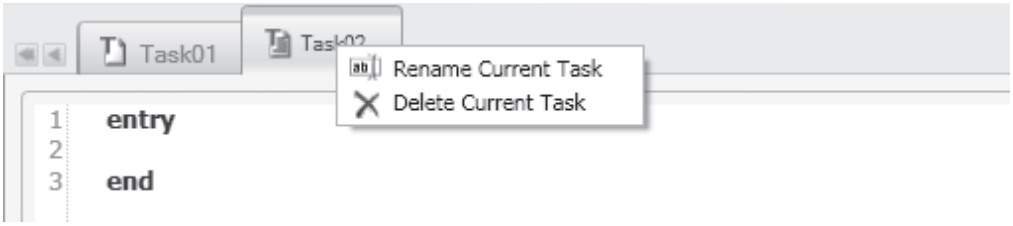
A new task appears on the DriveProgramming Editor.

The displayed tasks have the highest priority at the left side. A task at the left side in order is processed in 1-ms or 2-ms processing time (selected by UE-01).

All tasks must begin with "entry" command, and finish with "end" command.

Flowchart	Text
	

You can delete or rename the selected task by right-clicking on its tab in the DriveProgramming Editor.



4-4 Inserting, Deleting and Calling Subroutines

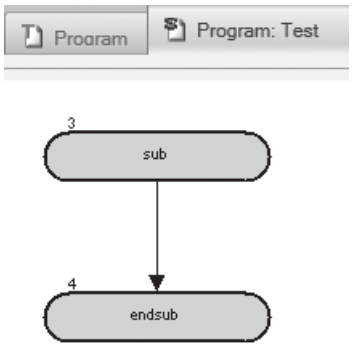
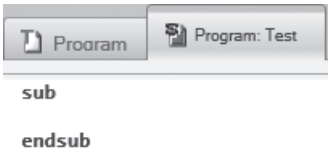
To insert a subroutine, select the tab of the task in which you want to insert a subroutine, and select [New Tab] - [New Subroutine (flowchart)] or [New Subroutine (text)] in the toolbar of the DriveProgramming Editor.

A new subroutine appears on the DriveProgramming Editor.

Similar to tasks, you can delete or rename a subroutine by right-clicking on its tab.

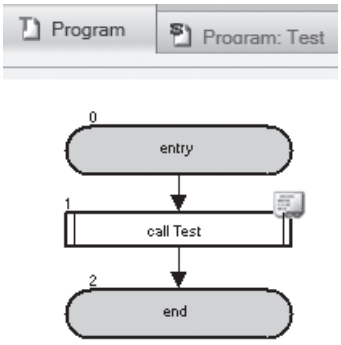
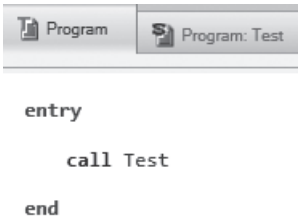
The subroutine name is displayed as "Task name: Subroutine name", next to the task name to which the subroutine belongs.

All subroutines must begin with "sub" command, and finish with "endsub" command.

Flowchart	Text
	

To execute a subroutine, specify the subroutine name in the "call" command.

It is only possible to call a subroutine that belongs to the task. Tasks cannot share the same subroutine. To use a subroutine with multiple tasks, insert the same subroutine in each task.

Flowchart	Text
	

4-5 Creating Flowchart Programs

When you create a DriveProgramming program, you can select flowchart or text for each task or subroutine.

Follow the steps described below to create a flowchart program.

1 Open the DriveProgramming Editor.

The DriveProgramming auxiliary windows (Toolbox, Block Parameters, Properties and Error List tab) are displayed automatically.

2 From [New Tab] in the toolbar, select [New Task (flowchart)] or [New Subroutine (flowchart)].

3 Select commands from the Toolbox window and move them to the DriveProgramming Editor by drag-and-drop.

On the upper left of each block placed on the Editor, the block number will appear after the compilation is finished.

Block numbers are the consecutive numbers starting with "0". A block number is given to each block of the whole program.

Block numbers do not match the line numbers of the program converted to text.

4 Click the block to edit its properties.

In the Properties window, edit arguments that are associated with the block.

5 Connect the blocks according to the program sequence.

You can connect a block with another by dragging from one's orange-colored contact to another's green-colored contact.

6 Edit user parameters in the Block Parameters window.

7 Perform operations such as program compilation, transferring to the inverter, and data saving.

Execute compilation and check for any compilation errors in the program.

You can transfer the program to the inverter when the compilation is finished successfully.

To save the program, save the whole project. Or, you can save the program separately by using the function that exports programs.

4-6 Creating Text Programs

When you create a DriveProgramming program, you can select flowchart or text for each task or subroutine.

Follow the steps described below to create a text program.

1 Open the DriveProgramming Editor.

The DriveProgramming auxiliary windows (Toolbox, Block Parameters, Properties and Error List tab) are displayed automatically.

2 From [New Tab] in the toolbar, select [New Task (text)] or [New Subroutine (text)].

3 There are three ways to edit text program codes:

- Manual typing
- Calling the text command list (Ctrl + space)
- Dragging and dropping commands from Toolbox window

4 Set arguments of each command.

The required arguments are displayed with a green background for the commands that are dragged and dropped from the Toolbox window and the commands selected from the text command list (Ctrl + space).

Set each argument to complete the command.

Each time you edit a command by manual typing, the popup support will appear to help you complete the command. For details on each command, refer to *Section 6 DriveProgramming Commands*.

You need not care about unnecessary lines and spaces when creating a program because they will be deleted by program compilation.

5 Perform operations such as program compilation, transferring to the inverter, and data saving.

Execute compilation and check for any compilation errors in the program.

You can transfer the program to the inverter when the compilation is finished successfully.

To save the program, save the whole project. Or, you can save the program separately by using the function that exports programs.

4-7 Editing Transferred (Uploaded) Programs

You can edit the program which is transferred (uploaded) from the inverter.

Follow the steps described below to edit the program.

1 Open the DriveProgramming Editor.

The DriveProgramming auxiliary windows (Toolbox, Block Parameters and Properties) are displayed automatically.

2 Go online with the CX-Drive. From the Menu, select [Drive] - [Work Online]. Or, click the [Work Online] icon in the CX-Drive toolbar.

3 Click the [Transfer from Drive] icon in the toolbar of the DriveProgramming Editor.

A program is transferred from the drive (inverter) and automatically displayed in the DriveProgramming area of the DriveProgramming Editor.

4 Edit the transferred (uploaded) program.

"Programs after compilation" are downloaded to the inverter.

Therefore, the transferred (uploaded) program will be displayed as a text program.

To display it as a flowchart program, click [Convert whole program to Flowchart] in the toolbar of the DriveProgramming Editor and convert the program to flowchart.

5 Perform operations such as program compilation, transferring to the inverter, and data saving.

Execute compilation and check for any compilation errors in the program.

You can transfer the program to the inverter when the compilation is finished successfully.

To save the program, save the whole project. Or, you can save the program separately by using the function that exports programs.

When the DriveProgramming programs exist, you can transfer them to/from the inverter by using [Transfer to Drive] or [Transfer from Drive] icon in the CX-Drive toolbar. In this case, you need to select "programs" when a message dialog appears and asks you whether to transfer the parameters, programs, or both.

4-8 Saving Programs

There are two ways to save programs created with the DriveProgramming function as described below.

Select a way suitable for your purpose.

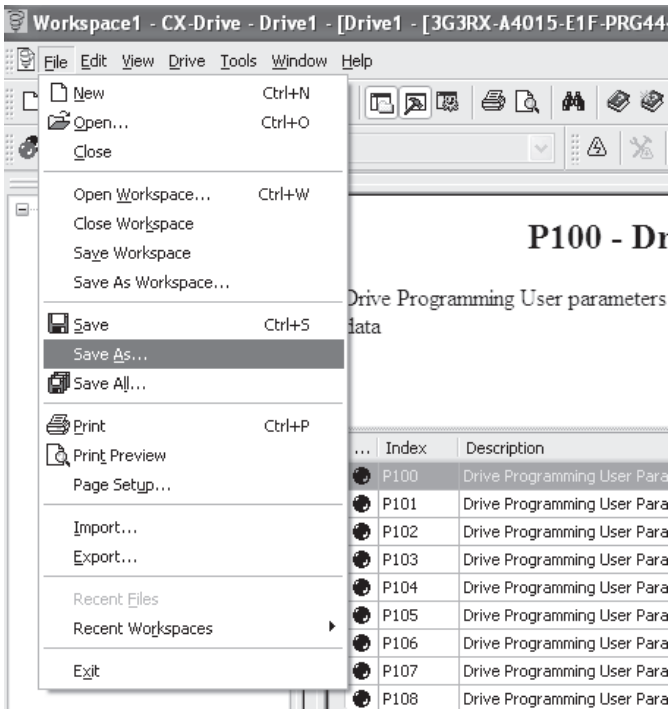
● Saving whole CX-Drive project

By saving the whole project created with the CX-Drive, you can save all drive data including the DriveProgramming program.

Click [File] in the Menu bar, select [Save As] and enter the file name.

When a saved project is opened, the DriveProgramming program included in the project is automatically loaded.

You can display the program by double-clicking the DriveProgramming in the Workspace and start the DriveProgramming Editor.

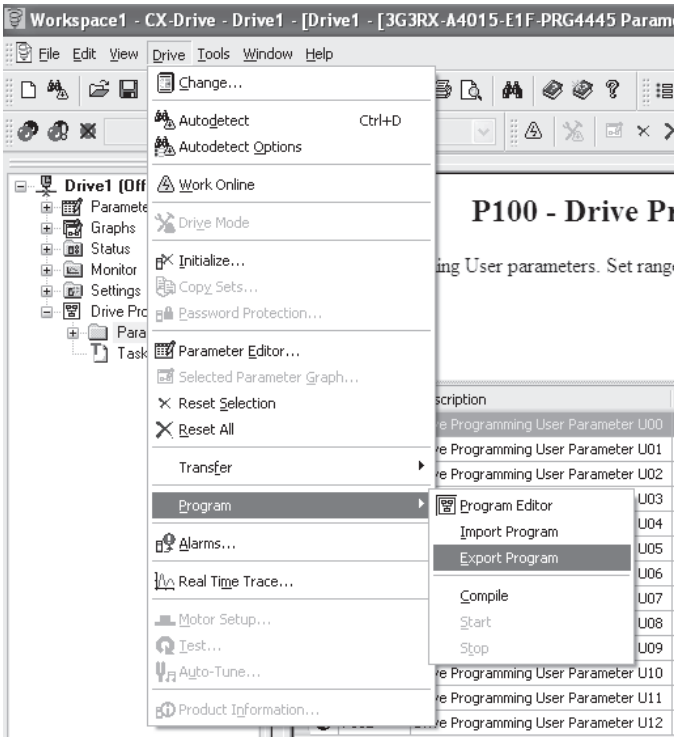


● Saving DiveProgramming program by export

You can save a DriveProgramming program separately.

Click [Drive] in the Menu bar, select [Program] - [Export Program] and enter the file name. CX-Drive separates the program from other drive information and saves the program only.

To import the exported program files into the CX-Drive, click [Drive] in the Menu bar, select [Program] - [Import Program] and select the file name.



4-9 Transferring and Verifying Programs

Program transfer and verification are possible between the inverter and the DriveProgramming function of the CX-Drive. At the same time, you can also execute parameter transfer and verification.

Transfer (from PC to Inverter)

The created programs are compiled (program-checked) and transferred (downloaded) to the inverter if there is no error.

From the [Drive] Menu, select [Transfer]- [To Drive] to transfer (download) data.

Transfer (from Inverter to PC)

The programs and parameters in the inverter are transferred (uploaded) to the DriveProgramming.

From the [Drive] Menu, select [Transfer]- [From Drive] to transfer (upload) data.

Verification (between PC and Inverter)

The DriveProgramming programs and parameters are compared with the data in the inverter.

From the [Drive] Menu, select [Transfer] - [Compare with Drive] to execute verification.

Partial Transfer (from PC to Inverter)

From the [Drive] Menu, select [Transfer] - [Selection To Drive] to transfer (download) the selected data.

Partial Transfer (from Inverter to PC)

From the [Drive] Menu, select [Transfer] - [Selection From Drive] to transfer (upload) the selected data.

4-10 Executing Programs (DriveProgramming Function Selection)

There are two ways to execute programs after you transferred (downloaded) them to the inverter as described below.

Executing Program via CX-Drive

Executing via CX-Drive is a convenient way to debug the created program.

Use  icon in the toolbar of the DriveProgramming Editor to start the program. To stop the program, use  icon. Note that executing program via the CX-Drive is enabled only when the following conditions are met.

- When the CX-Drive is connected to the inverter.
  Once the program is started, it does not stop even if the CX-Drive is disconnected. Therefore, when executing the program, make sure that you can stop the inverter immediately by turning off the power supply, etc.
- When the program in the CX-Drive matches the program in the inverter through verification after compilation.
- The program stop is executed regardless of the program in the CX-Drive.

Executing Program in Applications

You execute the program by setting the inverter parameter, the DriveProgramming Function Selection (UE-02).

Once the program reaches "end" command after a series of processes was completed, the program is not executed unless it is restarted.

To repeat the program, create a loop program so that the program does not reach “end” command.

If the program is stopped while the Motor is running, the Motor will be stopped according to the setting for the STOP mode selection (AA115) (deceleration stop or free-run stop).

Parameter No.	Function name	Data	Description
UE-02	EzSQ Function Selection	00: Disabled	Disables the DriveProgramming function. Programs are not executed. If you change the setting to 00 (Disabled) during program execution, the program will be stopped.
		01: [PRG] terminal	The DriveProgramming program is started when the input terminal[*1] set to 99 (PRG) is turned ON.
		02: Always	Starts the DriveProgramming program automatically after the inverter power supply is turned on. If you change the setting to 02 (Enabled) while the program is stopped, the program will be started.

Parameter No.	Function name	Data	Description
CA-01 to CA-11	Input terminals 1 to 9, A and B	99: PRG	When EzSQ function enable (UE-02) is set to 01 ([PRG terminal] (Start/stop via input terminal PRG), the program is started via the input terminal with this setting.

*1. Input terminals are for 1 to 9, A and B.

Precautions for Safe Use

When the DriveProgramming program is stopped, the output controlled by the DriveProgramming retains the status before the program stop.

For this reason, configure the system so that the stop of the DriveProgramming program in the inverter can be detected by the DriveProgramming start signal and the alarm (trip) signal, and the inverter's peripheral devices can be stopped safely.

Precautions for Correct Use

The following are the status of function variables when the DriveProgramming program is stopped. Take necessary measures in consideration of each status.

- For the output terminals (the inverter's actual output terminals), the status before the program stop is retained.

  However, the output terminals that are not set to MO1 to MO7 (general-purpose outputs of the DriveProgramming) operate as the inverter's normal output terminals.
- When the DriveProgramming function is selected for the frequency reference, acceleration/deceleration time or analog outputs, the set values for these functions before the program stop are retained.
- The data of the user parameter variables, internal user variables and internal user contacts before the program stop is retained.
- The status of the inverter's actual input terminals such as the input terminal and analog input terminals is not retained but always updated.
- The data of the output variables (function bits such as RUN, FA1 and AL) and inverter monitor variables is not retained but always updated according to the status of the inverter.
- Only the DriveProgramming's input variables (function bits such as FW, RV and CF1) and timer variables are cleared at the same time as the program stop, and all data are changed to zero.

When the DriveProgramming function is started/stopped through the CX-Drive, the value of the inverter parameter, EzSQ function enable (UE-02) is temporarily changed to 02 (start) or 00 (stop) only in the RAM data.

Take the following measures after you started/stopped the DriveProgramming function through the CX-Drive.

- Turn ON the power supply for the inverter again and return the value of UE-02 to that saved in the EEPROM.
- Do not perform the following EEPROM saving operations before you turn ON the power supply for the inverter.
  - Transferring (downloading) a part of parameters from the CX-Drive.
  - Issuing the "enter" command via the Modbus communication or a communication option.

When UE-02 is set to 02, stop the running program that was started at power ON before you restart the program via the CX-Drive.

If any program that was started at power ON is running, the CX-Drive cannot restart the program.

In the CX-Drive, if you click the button to start the program when inverter is already in operation, the following message will be displayed in the final step.

"The program is running, so it cannot be transferred to the drive."

If you click [OK] here, the program is started forcibly regardless of the operation status of the inverter.

Before you start the program, check the status of the equipment and ensure safety.

4-11 Other Useful Functions

Converting Flowchart to Text

There are two ways to convert flowchart programs to text programs.

Command	Icon	Description
Convert Flowchart to Text		Converts current flowchart task or subroutine to text.
Convert Whole Program to Text		Converts whole program to text.

Precautions for Correct Use

- When a flowchart program is converted to a text program, the program is once compiled. If a compilation error occurs, the conversion fails.
- On the upper left of each block in the flowchart program, the block number will appear after the compilation is finished.

  Block numbers are the consecutive numbers starting with "0". A block number is given to each block of the whole program.

  Block numbers do not match the line numbers of the program converted to text.

Converting Text to Flowchart

There are two ways to convert text programs to flowchart programs.

Command	Icon	Description
Convert Text to Flowchart		Converts current text task or subroutine to flowchart.
Convert Whole Program to Flowchart		Converts whole program to flowchart.

Precautions for Correct Use

- When a text program is converted to a flowchart program, the program is once compiled. If a compilation error occurs, the conversion fails.
- On the upper left of each block in the flowchart program, the block number will appear after the compilation is finished.

  Block numbers are the consecutive numbers starting with "0". A block number is given to each block of the whole program.

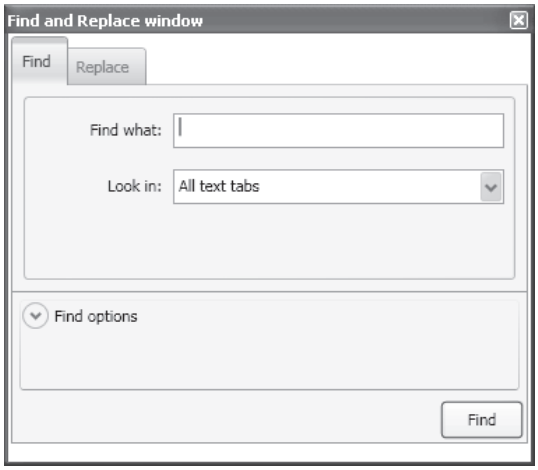
  Block numbers do not match the line numbers of the program converted to text.
- Comments, alias definition, region definition, etc. created in the text program are deleted when a compilation is performed for conversion. The operation does not change, however, the forms and contents of the program are partially changed.

Find and Replace Functions

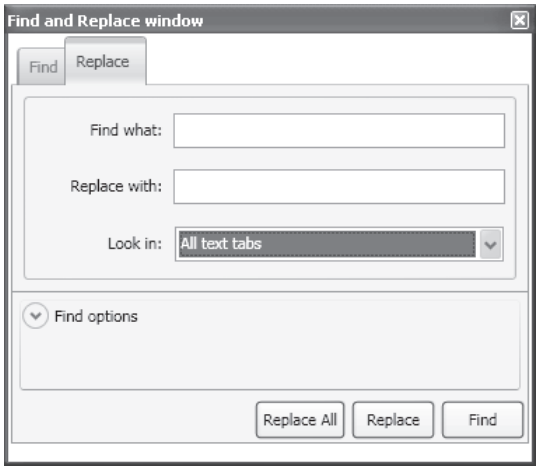
Find and replace functions are only available in text programs. You can look for or replace any character strings inside the text program by using the find and replace functions.

To use the Find function, click  icon or press the shortcut keys Ctrl + F and select [Find] tab.

To use the Replace function, click  icon or press the shortcut keys Ctrl + F and select [Replace] tab.



Find



Replace

Adding Comments (Text Program)

You can add comments to text tasks or subroutines. You cannot add comments to flowchart programs. To add a comment in a text line, type the comment after putting a ' (single quote) mark. Comments are displayed in green.

● Example

```
#alias global Time as U(10)              ' Timer time
#alias global AppTimer as TD(0)          ' Timer TD(0)
#alias global Temp as UL(05)             ' Internal use
```

Precautions for Correct Use

- The comments created in a text program are deleted when the program is transferred (downloaded) to the inverter or compiled for program conversion. To save the created comments, save the program before you execute program compilation. You can save the program by saving the whole project in the CX-Drive or export file of the program.
- In the verification process, the "program after compilation" is compared with the program inside the inverter. Therefore, the comments, alias definitions, region definitions, etc. are not verified.

Alias Definition (Text Program)

You can define aliases before the "entry" command in a task of the text program. You cannot define aliases in a subroutine or flowchart program.

Alias definition refers to specifying names for parameters, variables, commands and numeric constants.

By using the alias definition, you can handle these names specified in the program in the same way as parameters, variables, commands and numeric values. This is useful for enhancing program readability.

- **Local alias**: you can use this definition with currently selected task or subroutine. You cannot use it with other tasks or subroutines in the program. The format of the local alias definition in a task is shown below.

```
#alias local alias as replacement
```

● Example

```
#alias local ON_ as 1
#alias local OFF_ as 0
#alias local Monitor_1 as UMon(0)
#alias local MaxFrequency as A004
#alias local Count as U(00)
#alias local Dummy_1 as UL(00)

entry
```

- **Global alias**: you can use this definition with all tasks and subroutines in the program. You can set alias definitions in any of up to five tasks. The format of the global alias definition is shown below.

```
#alias global alias as replacement
```

● Example

```
#alias global const_100 as 100
#alias global Acceleration as F002
#alias global Deceleration as F003
#alias global Time as U(10)
#alias global AppTimer as TD(0)
#alias global Temp as UL(05)

entry
```

Precautions for Correct Use

- The alias definition created in the text program is deleted when the program is compiled for transferring (downloading) to the inverter or for program conversion. The specified names are converted to the normal names and numeric values. To save the created alias definition, save the program before you execute program compilation. You can save the program by saving the whole project in the CX-Drive or export file of the program.
- In the verification process, the "program after compilation" is compared with the program inside the inverter. Therefore, the comments, alias definitions, region definitions, etc. are not verified.
- For alias definitions, you cannot use the variables or commands that are already used. If you do this, a compile error will be displayed.

Region Definition (Text Program)

You can define regions in tasks or subroutines of the text program. You cannot define regions in flow-chart programs.

Region definition refers to dividing a program into groups of lines by specifying their areas.

By using the region definition, you can divide a program into parts and fold each of them in the display. This helps the program look easy to read.

● Example

```
#region Alias

#alias global const_100 as 100
#alias global Acceleration as F002
#alias global Deceleration as F003
#alias global Time as U(10)
#alias global AppTimer as TD(0)
#alias global Temp as UL(05)

#endregion

entry

#region Start

    Acceleration := const_100
    Deceleration := const_100
    Time := 500
    Temp := 10000
    set-freq := 1000
    Fw := 1

#endregion

#region Stop...
```

Precautions for Correct Use

- The region definition created in the text program is deleted when the program is compiled for transferring (downloading) to the inverter or for program conversion.

  To save the created region definition, save the program before you execute program compilation. You can save the program by saving the whole project in the CX-Drive or export file of the program.
- In the verification process, the "program after compilation" is compared with the program inside the inverter. Therefore, the comments, alias definitions, region definitions, etc. are not verified.

5

DriveProgramming User Variables

This section describes the user variables provided for DriveProgramming.

5-1 User Variables and User Parameters

The following variables are provided for creating programs: User parameter variables, internal user variables, and internal user contacts.

Use these variables for the program user interface, initial data for calculation, data saving during calculation, data saving, etc.

User Parameter Variables U(00) to U(63)

The DriveProgramming user parameter variables U(00) to U(63) are the unsigned 1-word variables.

The DriveProgramming user parameters U00 to U63 (UE-10 to UE-73) for the inverter correspond to the DriveProgramming user parameter variables U(00) to U(63), respectively. Use these variables for the following applications.

Application	Description
User interface	Use the variables as parameters to adjust calculation results. You can adjust them using the inverter parameters (UE-10 to UE-73) according to the status of application.
Initial data for calculation	You can use the user parameter variables U(00) to U(63) as the variables for calculation. The data of parameters UE-10 to UE-73 saved in the EEPROM can be used as the initial data for calculation.
Data saving during calculation	You can use the variables for saving data temporarily while the calculation is in progress.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
U(00) to U(63)	User parameter variable (corresponding to UE-10 to UE-73)	0 to 65,535	0	–	Unsigned 1 word	R/W

- The data of the parameters UE-10 to UE-73 saved in the EEPROM are automatically set to the user parameter variables U(00) to U(63) when the power is turned on.
- By monitoring the parameters UE-10 to UE-73, you can check the data of the user parameter variables U(00) to U(63) after the program execution is started. Note that the data displayed on the LCD Operator is the data at the moment when it is displayed. To update the data, once display the parameter number and then display the data again.
- When you change the parameters UE-10 to UE-73 by the LCD Operator and press the Enter key, the changed data is saved in the EEPROM and reflected in the current user parameter variables U(00) to U(63). The data saved in the EEPROM is set to the variables U(00) to U(63) automatically when the power supply is turned on again.
- Even if the data of the user parameter variables U(00) to U(63) is changed in the program, the parameters UE-10 to UE-73 is not saved in EEPROM.

Precautions for Correct Use

When the DriveProgramming program is stopped, the data of the user parameter variables before the program stop is retained. When the program execution is started again, the process begins with the retained data.

Internal User Variables UL(00) to UL(15)

The DriveProgramming's internal user variables UL(00) to UL(15) are the signed 2-word variables.

Use these variables for saving data during calculations such as four arithmetic operations.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
UL(00) to UL(15)	Internal user variable Variables that correspond to UF-02 to UF-32	-2147483648 to 2147483647	0	–	Signed 2 words	R/W

- Data saved in EEPROM of parameters UF-02 to UF-32 is set to EzSQ user parameter UL (00 to 15) when the power supply is turned ON.
- Even when data of EzSQ user parameter UL (00 to 15) is changed on a program of the DriveProgramming, parameters UF-02 to UF-32 are not in EEPROM.
- To move the upper word data of the internal user variables UL(00) to UL(15) to the 1-word size user parameter variables U(00) to U(63), use the following operation.

  Assign the positive upper word data: U(00) = UL(00)/65536

  Assign the negative upper word data: U(00) = UL(00)/65535

  The lower word data will move to U(00) when U(00) = UL(00) is executed.

Precautions for Correct Use

When the DriveProgramming program is stopped, the data of internal user variables before the program stop is retained. When the program execution is started again, the process begins with the retained data.

Internal User Contacts UB(0) to UB(15)

The DriveProgramming's internal user contacts UB(0) to UB(15) are the bit-access variables.

Use these variables for saving data during bit operations.

You can also use the variable UBw as a word-access data which contains the internal user contacts UB(0) to UB(15) in its lower byte.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
UB(0) to UB(15)	Internal user contact (bit access)	0: OFF 1: ON	0	–	bit	R/W

- You can use the internal user contacts UB(0) to UB(15) as bit-size variables.
- UB(0) to UB(15) are cleared to zero when the power supply is turned ON.
  Any data saving measure like EEPROM is not provided.
- To set initial data, create a program in which the initial data is set to UB(0) to UB(15).

Function variable	Description	Data range	Default data	Unit	Data size	R/W
Ubw	Internal user contact (word access)	0 to 255	0	–	Unsigned 1 word	R/W

- The internal user contact (word access) UBw is a function to use the internal user contacts UB(0) to UB(15) as a word-size variable.
- The internal user contacts UB(0) to UB(15) are set in the lower byte. For the upper byte data, zero is read out. If any data is written to the upper byte, the data is ignored.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	UB (15)	UB (14)	UB (13)	UB (12)	UB (11)	UB (10)	UB (9)	UB (8)	UB (7)	UB (6)	UB (5)	UB (4)	UB (3)	UB (2)	UB (1)	UB (0)

Precautions for Correct Use

When the DriveProgramming program is stopped, the data of internal user contacts before the program stop is retained. When the program execution is started again, the process begins with the retained data.

5-2 Input/Output Terminal Variables

This section describes the variables provided for using the following inverter terminals for the DriveProgramming function: Input terminals, output terminals, relay outputs, analog input terminals, and analog output terminals. Use these variables as the interface between the inverter's peripheral devices and the DriveProgramming function.

Input Terminal Variables X(00) to X(10)

You can use the inverter's input terminals as the input terminal variables X(00) to X(10) for the DriveProgramming function by setting these input terminals to the general-purpose input MI1 to MI11.

When Xw variables are used, the input terminal variables X(00) to X(10) can be used as word access data.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
X(00) to X(10)	Input terminal variable (bit access)	0: OFF 1: ON	0	–	bit	R

- The status of the input terminals 1 to 9, A and B is captured and set as bit-size variables. This variable is read-only.
- When the input terminals 1 to 9, A and B Selection (CA-01 to CA-11) are set to 86 to 96 (MI1 to MI11: General-purpose input), the status of the input terminals 1 to 9, A and B is captured and set as the input terminal variables X(00) to X(10) of the DriveProgramming.

  Numbers are given to the input terminal variables X(00) to X(10) according to the numerical order of the set general-purpose inputs MI1 to MI11, not the terminal numbers 1 to 9, A and B.

Function variable	Input Terminal (CA-01 to CA-11)
X(00)	86: MI1
X(01)	87: MI2
X(02)	88: MI3
X(03)	89: MI4
X(04)	90: MI5
X(05)	91: MI6
X(06)	92: MI7
X(07)	93: MI8
X(08)	94: MI9
X(09)	95: MI10
X(10)	95: MI11

Function variable	Description	Data range	Default data	Unit	Data size	R/W
Xw	Input terminal variable (word access)	0 to 65,535	0	–	Unsigned 1 word	R

- The input terminal variable (word access) Xw is a function to use the input terminal variables X(00) to X(10) as a word-size variable. This variable is read-only.
- The input terminal variables X(00) to X(10) are set to the bit 0 to 10. For the from bit 11 and unused input terminal variables, zero is read out.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	X (15)	X (14)	X (13)	X (12)	X (11)	X (10)	X (9)	X (8)	X (7)	X (6)	X (5)	X (4)	X (3)	X (2)	X (1)	X (0)

Precautions for Correct Use

- When the DriveProgramming program is stopped, the status of the input terminal variables is not retained but updated according to the status of actual input terminals.
- For Input terminal 1 to 9, A and B, when MI1 to MI11 (General-purpose input 1 to 11) is selected, Input terminal [21 to 31] active state (CA-21 to CA-31) is set to NO or NC.

Output Terminal Variables Y(00) to Y(06)

You can use the inverter's output terminals as the output terminals Y(00) to Y(06) for the DriveProgramming function by setting the output terminals to the general-purpose outputs MO1 to MO7.

When Yw variables are used, the output terminal variables Y(00) to Y(06) can be used as word access data.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
Y(00) to Y(06)	Output terminal variable (bit access)	0: OFF 1: ON	0	–	bit	R

- You can control the status of the output terminals 11 to 15 and the relay output terminals (16, AL) as bit-size variables.
- By setting the Output terminal 11 to 15 Selection (CC-01 to CC-05) or the Relay Output (16, AL) Function Selection (CC-06, CC-07) to 69 to 75 (MO1 to MO7: General-purpose output), you can control the output terminals 11 to 15 or the relay output terminals (16, AL) as the output terminal variables Y(00) to Y(06) of the DriveProgramming. Numbers are given to the input terminal variables Y(00) to Y(06) according to the numerical order of the set general-purpose inputs MO1 to MO7, not the terminal numbers 11 to 15, 16 and AL.

Function variable	Output Terminal (CC-01 to CC-07)
Y(00)	69:MO1
Y(01)	70:MO2
Y(02)	71:MO3
Y(03)	72:MO4
Y(04)	73:MO5
Y(05)	74:MO6
Y(06)	75:MO7

Function variable	Description	Data range	Default data	Unit	Data size	R/W
Yw	Output terminal variable (word access)	0 to 65,535	0	–	Unsigned 1 word	R/W

- The output terminal variables (word access) Yw is a function to use the output terminal variables Y(00) to Y(06) as a word-size variable.
- The output terminal variables Y(00) to Y(06) are set in the bit 0 to 6. For the from bit 7 and unused output terminal variables, zero is read out. If any data is written to the upper byte, the data is ignored.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	Y (15)	Y (14)	Y (13)	Y (12)	Y (11)	Y (10)	Y (9)	Y (8)	Y (7)	Y (6)	Y (5)	Y (4)	Y (3)	Y (2)	Y (1)	Y (0)

Precautions for Safe Use

When the DriveProgramming program is stopped, the output controlled by the DriveProgramming retains the status before the program stop.

For this reason, configure the system so that the stop of the DriveProgramming program in the inverter can be detected by the DriveProgramming start signal and the alarm (trip) signal, and the inverter's peripheral devices can be stopped safely.

Precautions for Correct Use

- When the DriveProgramming program is stopped, the data of the output terminal variables before the program stop is retained. When the program execution is started again, the process begins with the retained data.

  However, the outputs with MO1 to MO7 (General-purpose output) not set for the DriveProgramming are controlled as the inverter's output terminals independently of the program.
- If multiple tasks use the same terminal, the output status of the task which is executed last will be effective. Considering safe control, we recommend you to avoid using multiple tasks for control.
- For Output terminal 11 to 15, relay output terminal 16 and relay output terminal AL selection, when MO1 to MO7 (General-purpose output 1 to 7) is selected, Output terminal [11 to 15] active state (CC-11 to CC-15), Output terminal [16] active state and Output terminal [AL] active state (CC-06, CC-07) are set to NO or NC.

Analog Input Terminal Variables XA(0) to XA(2)

You can use the inverter's frequency reference input (analog voltage input) Ai1 terminal and the frequency reference input (analog current input) Ai2 terminal as the analog input terminal variables XA(0) and XA(1) of the DriveProgramming function.

It is also possible to use the frequency reference auxiliary input (analog voltage input) Ai3 terminal as the analog input terminal variable XA(2) of the DriveProgramming function.

You can continuously monitor the status of the analog inputs regardless of the parameter settings.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
XA(0)	0 to 10 V /0 to 20 mA (Ai1 terminals)	0 to 10,000	0	0.01%	Unsigned 1 word	R
XA(1)	10 V or less /0 to 20 mA (Ai2 terminals)					
XA(2)	−10 to 10 V (Ai3 terminals)	−10,000 to 10,000			Signed 1 word	

- The analog input terminal variables XA(0) and XA(1) are unsigned 1-word variables, and XA(2) is a signed 1-word variable. This variable is read-only.
- The variables are displayed in increments of 0.01% as a percentage of the maximum analog input 10 V or 20 mA.
- You can allocate inverter functions to the analog input terminals with the inverter parameters as shown below. Select a function in each parameter.

  If you use the analog input terminals only for the DriveProgramming, do not select the analog input terminal in the following parameters: (AA101, AH-51, AH-70, AA101, AA102, bA110, Ad-01)
- To adjust the analog inputs, use the following inverter parameters: (Ai1: Cb-03 to Cb-07, Ai2: Cb-13 to A105, Cb-13 to Cb-17 and Ai3: Cb-23 to Cb-26).

Precautions for Correct Use

When the DriveProgramming program is stopped, the status of the analog input terminal variables is not retained but updated according to the status of actual input terminals.

Analog Output Terminal Variables YA(0) to YA(2)

You can use the inverter's digital output (PWM output) FM terminal and the analog output (voltage output) Ao1 terminal as the analog output terminal variables YA(0) and YA(1) of the DriveProgramming function.

For the 3G3RX2 Series Inverters, it is also possible to use the analog output (current output) Ao2 terminal as the analog output terminal variables YA(2) of the DriveProgramming function.

You can continuously monitor the status of the analog outputs regardless of the parameter settings.

To control the analog outputs via the DriveProgramming function, select DriveProgramming for the setting of the inverter parameters, [FM] monitor output selection (Cd-03), [Ao1] monitor output selection (Cd-04) and [Ao2] monitor output selection (Cd-05).

Function variable	Description	Data range	Default data	Unit	Data size	R/W
YA(0)	FM terminal	0 to 10,000	0	0.01%	Unsigned 1 word	R/W
YA(1)	Ao1 terminals					
YA(2)	Ao2 terminals					

- The analog output terminal variables YA(0) to YA(2) are unsigned 1-word variables.

- Set the variables in increments of 0.01% as a percentage of the maximum output duty, 10 V or 20 mA.
- When you set DriveProgramming for the setting of the inverter parameters, [FM] monitor output selection (Cd-03), [Ao1] monitor output selection (Cd-04) and [Ao2] monitor output selection (Cd-05), the analog output terminal can be controlled via DriveProgramming function. Even if you do not select DriveProgramming for the parameters, it is possible to monitor the status of the analog output terminals.

Function variable	Parameter Setting
YA(00)	Set [FM] monitor output selection (Cd-03) to 0 to 65535 (register No. for d, F-code).
YA(01)	Set [Ao1] monitor output selection (Cd-04) to 0 to 65535 (register No. for d, F-code).
YA(02)	Set [Ao2] monitor output selection (Cd-05) to 0 to 65535 (register No. for d, F-code).

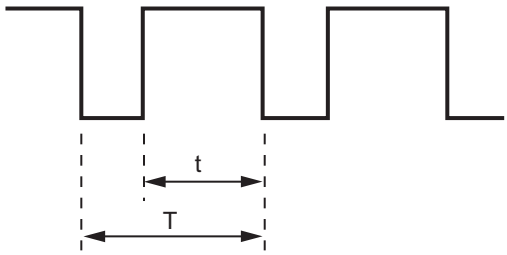
- Use the inverter parameters (Cd-14, Cd-24, Cd-34, Cd-23, Cd-33) to adjust analog output.

Precautions for Correct Use

- When the DriveProgramming program is stopped, the data of the analog output terminal variables before the program stop is retained. When the program execution is started again, the process begins with the retained data. However, the outputs with the DriveProgramming not set are controlled as the inverter's analog outputs independently of the program.
- The digital output (PWM output) FM terminal provides PWM signal outputs.

  The terminal outputs the value of 0.00 to 100.00% (variable) as the pulse width (duty ratio t/T) in a 6.4 ms cycle.

PWM output



Cycle T: Constant (6.4 ms)

Duty ratio t/T: Variable

5-3 Timer Variables

This section describes the timer variables provided for the DriveProgramming's timer control commands.

Timer Variables TD(0) to TD(15)

These are the timer counter variables and the timer output contacts used in the timer control commands of the DriveProgramming.

When the timer output contact is specified in the timer control command “delay on/off” or “timer set”, the timer counter variable corresponding to the specified contact starts counting. When the counter reaches the specified value, the timer output contact is turned ON.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
TD(0) to TD(15)	Timer output contact (bit access)	0: OFF 1: ON	0	–	bit	R

- The timer output contacts TD(0) to TD(15) are bit-size variables. This variable is read-only.
- When the program is started, the values of the timer output contacts TD(0) to TD(15) are cleared to zero.
- The timer output contacts TD(0) to TD(15) start their operation after they are specified in the “delay on/off” command or the “timer set” command. TD(0) to TD(15) are set to 0 (OFF) when the corresponding timer counter variables are cleared to zero. When the timer counter variable reaches the specified value, the contact is set to "1" (ON) and the status is retained. If the “timer off” command is executed, the status of TD(0) to TD(15) are cleared to zero.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
TDw	Timer output contact (word access)	0 to 255	0	---	Unsigned 1 word	R

- The timer output contact (word access) TDw is a function to use the timer output contacts TD(0) to TD(15) as a word-size variable. This variable is read-only.
- The timer output contacts TD(0) to TD(15) are set in the lower byte. Zero is read out for upper byte data.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	TD (15)	TD (14)	TD (13)	TD (12)	TD (11)	TD (10)	TD (9)	TD (8)	TD (7)	TD (6)	TD (5)	TD (4)	TD (3)	TD (2)	TD (1)	TD (0)

Precautions for Correct Use

If the DriveProgramming program is stopped, the data of the timer counter variables and the timer output contacts is not retained but cleared to zero.

Timer counter TC(0) to TC(15)

These are the timer counter variables used in the timer control commands of the DriveProgramming. The count data for the timer can be read.

When the variables are not used in the timer control commands, they are activated as the free running counters.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
TC(0) to TC(15)	Timer counter	0 to 2,147,483,646	0	10 ms	Unsigned 2 word	R

- Timer counter TC (0) to TC (15) is unsigned 2 word variables.
- When the variables are not used in the timer control commands, they are activated as the free running counters that count up at 10 ms cycles at the same time of the start of a program of the DriveProgramming.
- When the timer start command (timer set) and the delay command (delay on/off) are executed, the timer counter variables are activated as a timer counter for the specified timer contact outputs. When the commands are executed, the variables are cleared to zero. After a count up to the specified time, it stops.
- When the timer stop command (timer off) is executed, the variables are cleared to zero.

Precautions for Correct Use

When a program of DriveProgramming stops, the timer counter variables and the timer output contacts are not retained and the variables are cleared.

5-4 Inverter Setting Variables

This section describes the variables provided for setting inverter's frequency reference and acceleration/deceleration time.

Use these variables to control the inverter via the DriveProgramming program.

Frequency Reference Variable SET-Freq

When you directly control the frequency reference by the DriveProgramming function, set the Main speed input source selection [AA101] to 14 (Program function) to enable the frequency reference variable SET-Freq.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
SET-Freq	Frequency reference variable	0 to 59,000	0	0.01 Hz	Unsigned 1 word	R/W

- The frequency reference variable SET-Freq is an unsigned 1-word variable.
- The variable is enabled only when you set the Main speed input source selection to 14 (Program function).

Set in the 1st/2nd Main speed input source selection (AA101/AA211).
Allocate 24 (SET 2nd control) to Input 1 to 9, A and B to switch the control.

- The frequency reference variable SET-Freq is cleared to zero when the power supply is turned ON.
  Any data saving measure like EEPROM is not provided.
- To set initial data, create a program in which the initial data is set to SET-Freq.
- The range of frequency that the inverter can actually output is from the Minimum frequency adjustment, 1st-motor (Hb130) to the maximum frequency.
  If the set data is out of range, the inverter operates as follows.
  - Less than Minimum frequency adjustment, 1st-motor (Hb130)

When Control mode selection (AA121) is set to 09 (Zero-Hz range sensorless vector control) and 08 (Sensorless vector control), the Minimum frequency adjustment, 1st-motor (Hb130) is disabled and the specified frequency is output.

- More than maximum frequency

Limits the frequency reference to the value set in the 1st/2nd Maximum Frequency (Hb105/Hd105/Hb205/Hd205).

- You can monitor the frequency reference set in the frequency reference variable SET-Freq by using the inverter's parameter Main Speed reference monitor (FA-01).

Precautions for Correct Use

When the DriveProgramming program is stopped, the data of the frequency reference variable before the program stop is retained. When the program execution is started again, the process begins with the retained data.

Acceleration/deceleration Time Variables ACCEL and DECEL

When you directly control the acceleration/deceleration time of the frequency reference by the DriveProgramming function, set the inverter's parameter Acceleration/Deceleration Time Input Type (AC-01) to 04 (DriveProgramming) to enable the acceleration time variable ACCEL and the deceleration time variable DECEL.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
ACCEL	Acceleration time variable	0 to 360,000	Parameter Setting	0.01 second	Unsigned 2 words	R/W
DECEL	Deceleration time variable					

- The acceleration time variable ACCEL and the deceleration time variable DECEL are enabled only when you set the Acceleration/Deceleration Time Input Type (AC-01) to 04 (DriveProgramming).
- Only when the program is started for the first time after the power supply is turned ON, the value set in the inverter's acceleration/deceleration time parameters are set for the DriveProgramming's acceleration time variable ACCEL and deceleration time variable DECEL.

  The acceleration/deceleration time for the 1st/2nd Acceleration Time 1 (AC120/AC220) and the 1st/2nd Deceleration Time 1 (AC122/AC222) are set according to the 1st/2nd control selection.

- The data set in the acceleration time variable ACCEL and deceleration time variable DECEL are not saved in the EEPROM.
- The internal processing for the acceleration/deceleration time is performed in 40 ms cycles. Even if the value of the acceleration time variable ACCEL or the deceleration time variable DECEL is changed in the program, it takes up to 40 ms until the change is reflected in operation.
- In the following cases, the data that is set in the program is not reflected and the previously set data is used for operation: the data set for the acceleration time variable ACCEL or the deceleration time variable DECEL is out of range, or 0 is set for the 3G3RX2 Series Inverter.

Precautions for Correct Use

When the DriveProgramming program is stopped, the data of the acceleration time variable ACCEL and deceleration time variable DECEL before the program stop is retained. When the program execution is started again, the process begins with the retained data.

5-5 Inverter Monitor Variables

You can use the inverter's internal monitor function and status monitor function as the variables of the DriveProgramming function.
For details on each monitor function, refer to the High-function General-purpose Inverter 3G3RX2 Series User's Manual (I620).
Note that the data unit used for the inverter may be different from that for DriveProgramming. Be sure to use the following units.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
FM	Output Frequency Monitor (dA-01)	0 to 59,000	–	0.01 Hz	Unsigned 1 word	R

Use this function to monitor the inverter output frequency. The monitored data is equivalent to the data of the Output Frequency Monitor (dA-01). This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
Iout	Output Current Monitor (dA-02)	0 to 65,535	–	0.1%	Unsigned 1 word	R

Use this function to monitor the inverter output current. The monitored data is equivalent to the data of the Output Current Monitor (dA-02). The data is displayed in increments of 0.1% as a percentage of the rated current of the inverter. This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
Dir	RUN Direction Monitor (dA-03)	00: Stopped 01: 0 Hz output 02: Normal rotation in process 03: Reverse rotation in process	–	–	Unsigned 1 word	R

Use this function to monitor the operation direction of the inverter. The monitored data is equivalent to the data of the RUN Direction Monitor (dA-03). This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
PID-FB	PID Feedback Value Monitor (db-30)	-10,000 to 10,000	–	0.01%	Signed 2 words	R

Use this function to monitor the PID feedback value. The monitored data is equivalent to the data of the PID Feedback Value Monitor (db-30). This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
F-CNV	Output Frequency Monitor (After Conversion) (dA-06)	0 to 5,900,000	–	0.01	Unsigned 2 words	R

Use this function to monitor the output frequency after conversion. The monitored data is equivalent to the data of the Output Frequency Monitor (After Conversion) (dA-06). This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
Tmon	Output Torque Monitor (dA-17)	–10,000 to 10,000	–	%	Signed 1 word	R

Use this function to monitor the output torque. The monitored data is equivalent to the data of the Output Torque Monitor (dA-17). This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
Vout	Output Voltage Monitor (dA-18)	0 to 8,000	–	0.1 V	Unsigned 1 word	R

Use this function to monitor the output voltage. The monitored data is equivalent to the data of the Output Voltage Monitor (dA-18). This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
Power	Input Power Monitor (dA-30)	0 to 60,000 (132 kW or less) 0 to 20,000 (160 kW or more)	–	0.1 kW	Unsigned 1 word	R

Use this function to monitor the input power. The monitored data is equivalent to the data of the Input Power Monitor (dA-30). This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
RUN-Time	Total RUN Time Monitor (dC-22)	0 to 1,000,000	–	Time	Unsigned 2 words	R

Use this function to monitor the total operation time. The monitored data is equivalent to the data of the Total RUN Time Monitor (dC-22). This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
ON-Time	Cumulative power-on time (dC-24)	0 to 1,000,000	–	Time	Unsigned 2 words	R

Use this function to monitor the total power ON time. The monitored data is equivalent to the data of the Cumulative power-on time (dC-24). This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
PlsCnt	Pulse counter monitor (dA-28)	0 to 2,147,483,647	–	–	Signed 2 words	R

The DriveProgramming's user monitor variables UMon(0) to UMon(4) are the signed 2-word variables. If any data is set in the user monitor variables UMon(0) to UMon(4) in the DriveProgramming program, the data can be displayed through the inverter's user monitor parameters (db-08, db-10, db-12, db-14, db-16). Use this function to externally display the calculation status of the program.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
POS	Current Position Monitor (dA-20)	In the case of AA121=10 and AA123=03, data range -2147483648 to 2147483647. In the case of the condition mentioned above, data range -536870912 to 536870911	–	–	Signed 2 words	R

Use this function to monitor the current position. The monitored data is equivalent to the data of the Current Position Monitor (dA-20).

Function variable	Description	Data range	Default data	Unit	Data size	R/W
ERR-CNT	Fault Counter	0 to 65,535	–	Times	Unsigned 1 word	R

Use this function to monitor the inverter's total number of faults. The monitored data is equivalent to the data of the Fault Counter. This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
ERR(1) to ERR(10)	Fault Monitor 1 to 10	0 to 255	–	–	Unsigned 1 word	R

Use this function to monitor the data of the last six trips of the inverter. The monitored data is equivalent to the data of the Fault Monitor 1 to 10. This variable is read-only. Contents about trip are figures (***) in trip factor (E***) displayed on Trip History of LCD operator.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
RETRY(1) to (10)	Retry monitor 1 to 10 factor	0 to 255	–	–	Unsigned 1 word	R

The last 10 retry factors for an inverter can be monitored. The details about the monitor correspond to retry monitor 1 to 10 factor. This variable can be only read.

The contents are figures (***) in retry factor (r***) displayed on Retry History of LCD operator.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
DCV	DC Voltage Monitor (dA-40)	0 to 10,000	–	0.1 V	Unsigned 1 word	R

Use this function to monitor the inverter's internal DC voltage. The monitored data is equivalent to the data of the DC Voltage Monitor (dA-40). This variable is read-only.

Function variable	Description	Data range	Default data	Unit	Data size	R/W
STATUS	Inverter status monitor	0: Initialization 1: Ground fault selection 2: Stop 3: Run standby 4: Run preparation 5: Run 6: Stop standby 7: Retry standby 8: Retry	–	–	Unsigned 1 word	R

Use this function to monitor the inverter's status.

5-6 Input Variables

You can execute the functions allocated to the input terminals by the DriveProgramming program. The following variables correspond to the functions which can be allocated to the input terminals.

If any of the variables is set to 1 (ON), the corresponding function is enabled in the same way as when the input terminal is turned ON.

If any of the variables is set to 0 (OFF), the corresponding function is disabled. You can execute the functions by the program even if you do not allocate the functions to the Input Terminal Selection (CA-01 to CA-01). The Reference column in the following table shows the function setting data for the inverter. For details on each function, refer to the High-function General-purpose Inverter 3G3RX2 Series User's Manual (I620).

● Example

If you set FW (forward) to 1, the forward RUN command is executed.

FW: = 1 the inverter starts forward operation.

FW: = 0 the inverter stops forward operation and starts deceleration.

Precautions for Correct Use

- If the DriveProgramming program is stopped, the data of the input variables is not retained but cleared to zero.
- The variable FW (forward) and RV (reverse) are enabled only when the inverter's RUN Command Selection (AA111) is set to 00 (FW/RV terminal). The inverter does not operate with other setting options.
- Even if you set the variable FW (forward) or RV (reverse) to 1 immediately after turning on the power supply, the setting is ignored and neither forward or reverse operation is performed. Set 0, and then set 1 again. To avoid this operation, create a program that has one second of wait time after turning on the power supply, with such as “wait” command.
- The relationship between the inverter's input terminal function set in the Input Terminal Selection and the input variable is logical OR.

Function variable	Description	R/W	Reference
no	Without allocation	R/W	CA-01 to CA-11 = 0
FW	Normal rotation	R/W	CA-01 to CA-11 = 1
RV	Reverse rotation	R/W	CA-01 to CA-11 = 2
CF1	Multistage speed 1	R/W	CA-01 to CA-11 = 3
CF2	Multistage speed 2	R/W	CA-01 to CA-11 = 4
CF3	Multistage speed 3	R/W	CA-01 to CA-11 = 5
CF4	Multistage speed 4	R/W	CA-01 to CA-11 = 6
SF1 to SF7	Multistage speed bit 1 to 7	R/W	CA-01 to CA-11 = 7 to 13
ADD	Addition of frequency	R/W	CA-01 to CA-11 = 14
SCHG	Switching of command	R/W	CA-01 to CA-11 = 15
STA	3-wire starting up	R/W	CA-01 to CA-11 = 16
STP	3-wire stopping	R/W	CA-01 to CA-11 = 17
F/R	3-wire normal and reverse	R/W	CA-01 to CA-11 = 18
AHD	Retention of analog command	R/W	CA-01 to CA-11 = 19
FUP	Acceleration through remote operation	R/W	CA-01 to CA-11 = 20
FDN	Deceleration through remote operation	R/W	CA-01 to CA-11 = 21
UDC	Clearing of remote operation data	R/W	CA-01 to CA-11 = 22
F-OP	Forced switching of command	R/W	CA-01 to CA-11 = 23
SET	Second control	R/W	CA-01 to CA-11 = 24

Function variable	Description	R/W	Reference
RS	Reset	R/W	CA-01 to CA-11 = 28
JG	Jogging	R/W	CA-01 to CA-11 = 29
DB	Braking with external direct current	R/W	CA-01 to CA-11 = 30
2CH	2-step acceleration/deceleration	R/W	CA-01 to CA-11 = 31
FRS	Free-run stop	R/W	CA-01 to CA-11 = 32
EXT	External abnormality	R/W	CA-01 to CA-11 = 33
USP	Prevention of power restoration restarting	R/W	CA-01 to CA-11 = 34
CS	Commercial switch	R/W	CA-01 to CA-11 = 35
SFT	Soft-lock	R/W	CA-01 to CA-11 = 36
BOK	Brake check	R/W	CA-01 to CA-11 = 37
OLR	Switching of overload limit	R/W	CA-01 to CA-11 = 38
KHC	Clearing of integrated input power	R/W	CA-01 to CA-11 = 39
OKHC	Clearing of integrated output power	R/W	CA-01 to CA-11 = 40
PID	PID1 disabled	R/W	CA-01 to CA-11 = 41
PIDC	Resetting of PID1 integration	R/W	CA-01 to CA-11 = 42
PID2	PID2 disabled	R/W	CA-01 to CA-11 = 43
PIDC2	Resetting of PID2 integration	R/W	CA-01 to CA-11 = 44
PID3	PID3 disabled	R/W	CA-01 to CA-11 = 45
PIDC3	Resetting of PID3 integration	R/W	CA-01 to CA-11 = 46
PID4	PID4 disabled	R/W	CA-01 to CA-11 = 47
PIDC4	Resetting of PID4 integration	R/W	CA-01 to CA-11 = 48
SVC1 to SVC4	PID1 multistage target value 1 to 4	R/W	CA-01 to CA-11 = 51 to 54
PRO	Switching of PID gain	R/W	CA-01 to CA-11 = 55
PIO1	Switching of PID output 1	R/W	CA-01 to CA-11 = 56
PIO2	Switching of PID output 2	R/W	CA-01 to CA-11 = 57
SLEP	Satisfaction of SLEEP condition	R/W	CA-01 to CA-11 = 58
WAKE	Satisfaction of WAKE condition	R/W	CA-01 to CA-11 = 59
TL	Validation of torque limit	R/W	CA-01 to CA-11 = 60
TRQ1	Torque limit switchover 1	R/W	CA-01 to CA-11 = 61
TRQ2	Torque limit switchover 2	R/W	CA-01 to CA-11 = 62
PPI	PPI control switch	R/W	CA-01 to CA-11 = 63
CAS	Control gain switch	R/W	CA-01 to CA-11 = 64
SON	Servo-on	R/W	CA-01 to CA-11 = 65
FOC	Auxiliary excitation	R/W	CA-01 to CA-11 = 66
ATR	Validation of torque control	R/W	CA-01 to CA-11 = 67
TBS	Validation of torque bias	R/W	CA-01 to CA-11 = 68
ORT	Orientation	R/W	CA-01 to CA-11 = 69
LAC	Cancellation of LAD	R/W	CA-01 to CA-11 = 71
PCLR	Clearing of positional deviation	R/W	CA-01 to CA-11 = 72
STAT	Permission to inputting of Pulse string position command	R/W	CA-01 to CA-11 = 73
PUP	Addition of positional bias	R/W	CA-01 to CA-11 = 74
PDN	Subtraction of positional bias	R/W	CA-01 to CA-11 = 75
CP1 to CP4	Positional command selection 1 to 4	R/W	CA-01 to CA-11 = 76 to 79
ORL	Origin limit signal	R/W	CA-01 to CA-11 = 80
ORG	Return-to-origin start up signal	R/W	CA-01 to CA-11 = 81
FOT	Stopping of normal rotation driving	R/W	CA-01 to CA-11 = 82
ROT	Stopping of reverse rotation driving	R/W	CA-01 to CA-11 = 83
SPD	Switching of speed position	R/W	CA-01 to CA-11 = 84
PSET	Presetting of positional data	R/W	CA-01 to CA-11 = 85
MI1 to MI11	General purpose input 1 to 11	R/W	CA-01 to CA-11 = 86 to 96
PCC	Clearing of pulse counter	R/W	CA-01 to CA-11 = 97
ECOM	Starting up of EzCOM	R/W	CA-01 to CA-11 = 98
PRG	Starting of EzSQ program	R/W	CA-01 to CA-11 = 99

Function variable	Description	R/W	Reference
HLD	Stopping of acceleration/deceleration	R/W	CA-01 to CA-11 = 100
REN	Operation permission signal	R/W	CA-01 to CA-11 = 101
DISP	Fixation of display	R/W	CA-01 to CA-11 = 102
PLA	Pulse string input A	R/W	CA-01 to CA-11 = 103
PLB	Pulse string input B	R/W	CA-01 to CA-11 = 104
EMF	Emergency forced operation	R/W	CA-01 to CA-11 = 105
COK	Contactor check signal	R/W	CA-01 to CA-11 = 107
PLZ	Pulse string input Z	R/W	CA-01 to CA-11 = 109
TCH	Teaching signal	R/W	CA-01 to CA-11 = 110

5-7 Output Variables

You can execute the functions which can be allocated to the output terminals by using the DriveProgramming program. The following variables correspond to the functions which can be allocated to the output terminals.

Setting each variable to 1 (ON) or 0 (OFF) causes the same operation as when the output function turns the output terminal ON/OFF. You can monitor the status on the program even if you do not set each function in the parameters Output Terminal Selection (CC-01 to CC-05) or Relay Output (16C, AL) Function Selection (CC-06, CC-07). The Reference column in the following table shows the function setting data for the inverter. For details on each function, refer to the High-function General-purpose Inverter 3G3RX2 Series User's Manual (I620).

Precautions for Correct Use

When the DriveProgramming program is stopped, the status of the output variables is not retained but updated according to the actual function status.

Function variable	Description	R/W	Reference
no	Without allocation	R	CC-01 to CC-07 = 0
RUN	During operation	R	CC-01 to CC-07 = 1
FA1	When the constant speed is attained	R	CC-01 to CC-07 = 2
FA2	Equal to or above the set frequency	R	CC-01 to CC-07 = 3
FA3	Set frequency only	R	CC-01 to CC-07 = 4
FA4	Equal to or above the set frequency 2	R	CC-01 to CC-07 = 5
FA5	Set frequency only 2	R	CC-01 to CC-07 = 6
IRDY	Operation ready completion	R	CC-01 to CC-07 = 7
FWR	During normal rotation operation	R	CC-01 to CC-07 = 8
RVR	During reverse rotation operation	R	CC-01 to CC-07 = 9
FREF	Frequency command panel	R	CC-01 to CC-07 = 10
REF	Operation command panel	R	CC-01 to CC-07 = 11
SETM	Second control under selection	R	CC-01 to CC-07 = 12
OPO	Optional output	R	CC-01 to CC-07 = 16
AL	Alarm signal	R	CC-01 to CC-07 = 17
MJA	Severe failure signal	R	CC-01 to CC-07 = 18
OTQ	Excessive torque	R	CC-01 to CC-07 = 19
IP	During instantaneous power failure	R	CC-01 to CC-07 = 20
UV	Under insufficient voltage	R	CC-01 to CC-07 = 21
TRQ	During torque limitation	R	CC-01 to CC-07 = 22
IPS	During power failure deceleration	R	CC-01 to CC-07 = 23
RNT	RUN time elapsed	R	CC-01 to CC-07 = 24
ONT	Power ON time elapsed	R	CC-01 to CC-07 = 25
THM	Electronic thermal warning	R	CC-01 to CC-07 = 26
THC	Electronic thermal warning	R	CC-01 to CC-07 = 27
WAC	Capacitor life advance notice	R	CC-01 to CC-07 = 29
WAF	Fan life advance notice	R	CC-01 to CC-07 = 30
FR	Operation command signal	R	CC-01 to CC-07 = 31
OHF	Cooling fin heating advance notice	R	CC-01 to CC-07 = 32
LOC	Low current signal	R	CC-01 to CC-07 = 33
LOC2	Low current signal 2	R	CC-01 to CC-07 = 34
OL	Overload advance notice	R	CC-01 to CC-07 = 35
OL2	Overload advance notice 2	R	CC-01 to CC-07 = 36
BRK	Brake release	R	CC-01 to CC-07 = 37
BER	Brake abnormality	R	CC-01 to CC-07 = 38

Function variable	Description	R/W	Reference
CON	Contactor control	R	CC-01 to CC-07 = 39
ZS	0 Hz detection signal	R	CC-01 to CC-07 = 40
DSE	Excessive speed deviation	R	CC-01 to CC-07 = 41
PDD	Excessive positional deviation	R	CC-01 to CC-07 = 42
POK	Positioning completed	R	CC-01 to CC-07 = 43
PCMP	Pulse count compare-match output	R	CC-01 to CC-07 = 44
OD	PID excessive deviation	R	CC-01 to CC-07 = 45
FBV	PID feedback comparison	R	CC-01 to CC-07 = 46
OD2	PID2 excessive deviation	R	CC-01 to CC-07 = 47
FBV2	PID2 feedback comparison	R	CC-01 to CC-07 = 48
NDc	Communication disconnection	R	CC-01 to CC-07 = 49
Ai1Dc to Ai3Dc	Analog disconnection Ai1 to 3	R	CC-01 to CC-07 = 50 to 52
WCAi1 to WCAi3	Window comparator Ai1 to 3	R	CC-01 to CC-07 = 56 to 58
LOG1 to LOG7	Result of logical operation 1 to 7	R	CC-01 to CC-07 = 62 to 68
MO1 to MO7	General purpose output 1 to 7	R	CC-01 to CC-07 = 69 to 75
EMFC	Forced operation in process signal	R	CC-01 to CC-07 = 76
EMBP	During-bypass-mode signal	R	CC-01 to CC-07 = 77
LBK	Operation panel battery insufficient	R	CC-01 to CC-07 = 80
OVS	Excessive voltage of accepted power	R	CC-01 to CC-07 = 81
AC0 to AC3	Alarm code bit 0 to 3	R	CC-01 to CC-07 = 84 to 87
OD3	PID3 excessive deviation	R	CC-01 to CC-07 = 89
FBV3	PID3 feedback comparison	R	CC-01 to CC-07 = 90
OD4	PID4 excessive deviation	R	CC-01 to CC-07 = 91
FBV4	PID4 feedback comparison	R	CC-01 to CC-07 = 92
SSE	PID soft start abnormality	R	CC-01 to CC-07 = 93

6

DriveProgramming Commands

This section describes the commands provided for DriveProgramming.

6-1 Command Categories

The commands are divided into the following categories.

- Program control commands
- Arithmetic operation and logical operation commands
- I/O control commands
- Timer control commands
- Parameter control commands
- Inverter control commands

6-2 Command Format

Each command consists of the command and its arguments (0 up to 5).

An example is shown below.

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
entry						Indicates the beginning of the program.

If there is no argument, the cell is blank.

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
wait	<wait time>					<wait time> Waits for (wait time × 0.01) seconds.

There is one argument for the “wait” command. In this argument, input a wait time (variable or constant).

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
while	<condition>					Executes <command set> while <condition> is met.
<command set>						The command set that is executed while <condition> is met.
wend						Goes to the “while” (loop).

For the “while” command, set a condition in the argument. (For details, refer to *Conditions* on page 6-6.)

Between “while” and “wend”, write the command set that is executed while <condition> is met.

6-3 Command List

Program Control Commands

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
entry						Indicates the beginning of the task.
end						Indicates the end of the task.
call	<subroutine>					Jumps to <subroutine>.
sub	<subroutine>					Indicates the beginning of the subroutine.
end sub						Indicates the end of the subroutine.
goto	<label>					Jumps to <label> unconditionally.
on trip goto	<label>					Jumps to <label> when a trip occurs.
if	<condition>			goto	<label>	Jumps to <label> when the condition is met.
if	<condition>			then		Starts the “if” structure. If <condition> is met, it executes <command set 1> right after this command until the “else” command, and goes to the next step after the “endif”. If <condition> is not met, it executes <command set 2> right after the “else” command until the “endif” command, and goes to the next step after the “endif”.
<command set 1>						The command set that is executed when <condition> is met.
else						Starts the commands when <condition> is not met.
<command set 2>						The command set that is executed when <condition> is not met.
endif						Ends the “if” structure.

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
select	<condition variable>					Executes the commands after the “case” command when <condition variable> is equal to <condition value>.
case	<condition value 1>					Starts the commands assigned to the condition value.
<command set 1>						The command set assigned to <condition value 1>
[“case” condition value n] [<command set n>]						You can set one or more conditions with “case”.
case else						Starts the commands assigned to the values other than the condition value.
<command set>						The command set assigned to the values other than <condition value>
end select						Ends the “case” syntax.
for	<variable>	<start value>	<end value>	<incremental value>		Starts a loop from <start value> until <end value> with <incremental value>.
<command set>						The command set that is executed repeatedly
next						Goes to “for” (loop).
while	<condition>					Executes <command set> while the <condition> is met.
<command set>						The command set that is executed while <condition> is met.
wend						Goes to “while” (loop).
until	<condition>					Executes <command set> until <condition> is met.
<command set>						The command set that is executed while <condition> is not met.
loop						Goes to “until” (loop).
wait	<wait time>					<wait time> Waits for (wait time × 0.01) seconds.
wait	<condition>					Waits until <condition> is met.

Additional Information

For the format of <condition>, refer to *Conditions* on page 6-6.

Conditions

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
Condition	<variable 1/ constant>	=	<variable 2/ constant>			TRUE if <variable 1/constant> is equal to <variable 2/constant>.
	<variable 1/ constant>	<	<variable 2/ constant>			TRUE if <variable 1/constant> is less than <variable 2/ constant>.
	<variable 1/ constant>	<=	<variable 2/ constant>			TRUE if <variable 1/constant> is equal to or less than <variable 2/constant>.
	<variable 1/ constant>	>	<variable 2/ constant>			TRUE if <variable 1/constant> is greater than <variable 2/ constant>.
	<variable 1/ constant>	>=	<variable 2/ constant>			TRUE if <variable 1/constant> is equal to or greater than <variable 2/constant>.
	<variable 1/ constant>	<>	<variable 2/ constant>			TRUE if <variable 1/constant> is not equal to <variable 2/constant>.

Arithmetic Operation and Logical Operation Commands

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
Arithmetic operation	<variable 1>	: =	<variable 2/ constant>			Assigns <variable 2/constant> to <variable 1>.
	<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>.

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
Remainder for division	<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 2/ constant>			Assigns the absolute value of <variable 2/constant> to <variable 1>.
Logical operation	<variable 1>	: =	<variable 2/ constant>	and	<variable 3/ constant>	Performs a logical AND operation on <variable 2/constant> and <variable 3/constant> and assigns the result to <variable 1>.
	<variable 1>	: =	<variable 2/ constant>	or	<variable 3/ constant>	Performs a logical OR operation on <variable 2/constant> and <variable 3/constant> and assigns the result to <variable 1>.
	<variable 1>	: =	<variable 2/ constant>	xor	<variable 3/ constant>	Performs a logical exclusive OR operation on <variable 2/constant> and <variable 3/constant> and assigns the result to <variable 1>.
	<variable 1>	: = not	<variable 2/ constant>			Reverses the bit of <variable 2/constant> and assigns the result to <variable 1>.

Increment and Decrement Commands

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
Increment	inc	<variable>				Adds 1 to <variable>.
Decrement	Dec	<variable>				Subtracts 1 from <variable>.

I/O control commands

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
Input terminal variable assignment	<variable>	: =	X(i)			Set current state of the input terminal X(i) to <variable>. 0 = off, 1 = on
	<variable>	: =	Xw			Set the current state of the Xw to <variable> in units of words.

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
Output terminal variable output	Y(**)	: =	<variable/ constant>			Outputs data to the output terminal Y(**) in units of bits. 0 = off, 1 = on
	Yw	: =	<variable/ constant>			Outputs data to the output terminal in units of words.
Input variable output	<Input variable>	: =	<variable/ constant>			Outputs data to the input variable. 0 = off, 1 = on
Output variable assignment	<variable>	: =	<Output variable>			Assigns the status of the output variable to <variable>. 0 = off, 1 = on
Internal user contact control	<variable>	: =	UB(i)			Assigns the internal user contact information to <variable>. 0 = off, 1 = on
	<variable>	: =	UBw			Assigns the internal user contact information to <variable> in units of words.
	UB(**)	: =	<variable/ constant>			Outputs data to the internal user contact in units of bits. 0 = off, 1 = on
	UBw	: =	<variable/ constant>			Outputs data to the internal user contact in units of words.

Timer Control Commands

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
delay	on/off	<variable 1>	TD(i)	<variable 2/ constant>		Sets <variable 2/constant> for the specified timer and starts the timer. Changes <variable 1> to ON or OFF after the time of <variable 2/constant> elapsed. At the same time, it changes TD(i) to ON.
timer set	TD(*)	<variable/ constant>				Sets <variable/constant> for the specified timer and starts the timer. Changes TD(i) to ON after the time of <variable/constant> elapsed.
timer off	TD(*)					Stops the specified timer.

Parameter Control Commands

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
ChgParam	<parameter>	: =	<variable/ constant>			Replaces the content of <parameter> with <variable/constant>.
MonParam	<variable>	: =	<parameter>			Assigns the content of <parameter> to <variable>.

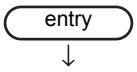
Inverter Control Commands

Command	Argument 1	Argument 2	Argument 3	Argument 4	Argument 5	Description
FW	: =	1				Operates the inverter in the forward direction.
RV	: =	1				Operates the inverter in the reverse direction.
stop						Decelerates the inverter to a stop. Note: This command acts as a reset when the inverter is in a trip state.
trip	<variable/ constant>					Issues a user trip according to <variable/ constant>.
SET-Freq	: =	<variable/ constant>				Assigns <variable/constant> (× 0.01 Hz) to the inverter frequency reference variable.
ACCEL	: =	<variable/ constant>				Assigns <variable/constant> (× 10 ms) to the inverter acceleration time variable.
DECEL	: =	<variable/ constant>				Assigns <variable/constant> (× 10 ms) to the inverter deceleration time variable.

6-4 Program Control Commands

Entry

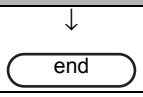
Command	Description	Argument
	Indicates the beginning of the task.	---

Format	
Flowchart method	**Text language method**
	entry

Note It is necessary to have this command at the begging of each task.

End

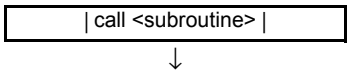
Command	Description	Argument
	Indicates the end of the task.	---

Format	
Flowchart method	**Text language method**
	end

Note It is necessary to have this command at the end of each task.

Call

Command	Description	Argument
	Jumps to <subroutine>.	**Subroutine**: subroutine is identified by a name or alias that you define.

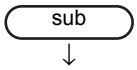
Format	
Flowchart method	**Text language method**
	call <subroutine>

Note 1. When execution of the subroutine is completed, the next command line after the “call” command is executed.

2. To set the subroutine name for the “call” command in flowchart, right-click the command and select [Set Subroutine] from the menu. Select from the displayed options.

Sub

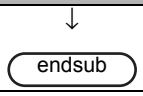
Command	Description	Argument
	Indicates the beginning of the subroutine.	---

Format	
Flowchart method	**Text language method**
	sub

Note It is necessary to have this command at the beginning of each subroutine.

End sub

Command	Description	Argument
	Indicates the end of the subroutine.	---

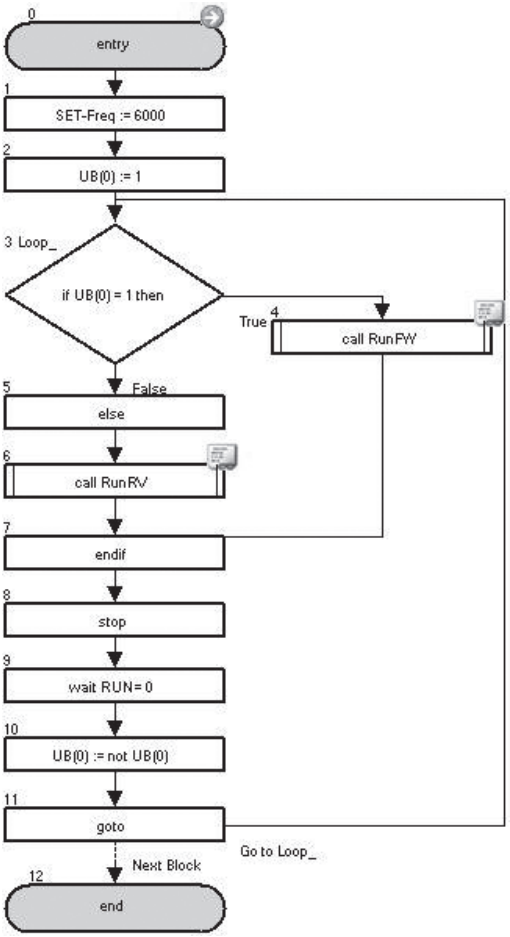
Format	
Flowchart method	**Text language method**
	endsub

Note It is necessary to have this command at the end of each subroutine.

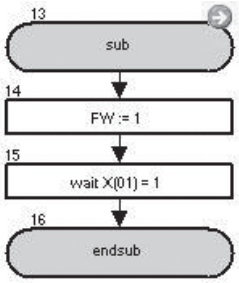
● Example

Flowchart

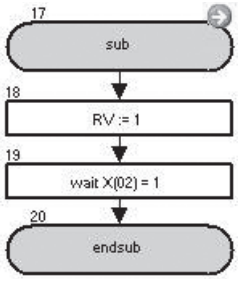
Main



Main: RunFW



Main: RunRV



Text

Main

```
entry
    set-freq := 6000
    UB(0) := 1
:Loop_
    if UB(0) = 1 then
        call RunFW
    else
        call RunRV
    endif
    stop
    wait RUN = 0
    UB(0) := not UB(0)
    goto Loop_
end
```

Main: RunFW

```
sub
    FW := 1
    wait X(01) = 1
endsub
```

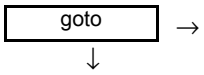
Main: RunRV

```
sub
    RV := 1
    wait X(02) = 1
endsub
```

Block number	Operation
1 and 2	Sets the inverter output frequency to 60.00 Hz and changes the internal user contact UB(0) to ON.
3 to 7	If UB(0) is ON, it executes the subroutine RunFW. If UB(0) is OFF, it executes the subroutine RunRV.
8 to 11	When execution of the subroutine is completed, it stops the inverter, waits for RUN to change to OFF, reverses UB(0), and returns to block 3.
13 to 16	Operates the inverter in the forward direction and waits with the “wait” command until X(01) changes to ON. (subroutine: RunFW)
17 to 20	Operates the inverter in the reverse direction and waits with the “wait” command until X(02) changes to ON. (subroutine: RunRV)

Go To

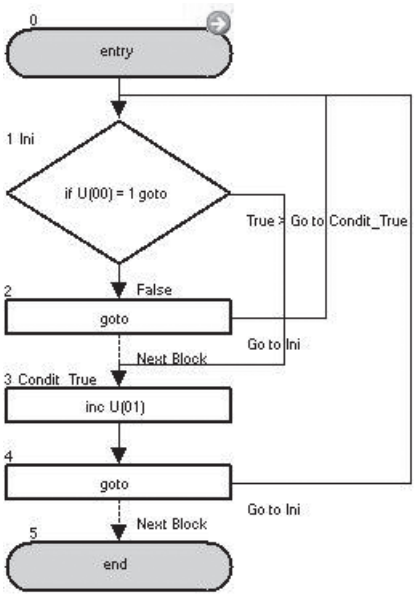
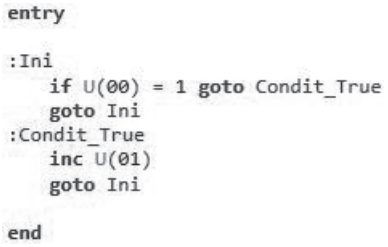
Command	Description	Argument
	Jumps to <label> unconditionally.	**Label**: a name that is used to identify a particular function block in the task.

Format	
Flowchart method	**Text language method**
	goto <label>

Note 1. This command must be connected to the command that is executed next. This is necessary to clarify the program flow.

2. To set the label name in flowchart, right-click the command and select [Set Label] from the menu. You can specify any name.

● Example

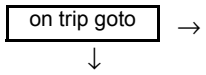
Flowchart	Text
	


```
entry

:Ini
    if U(00) = 1 goto Condit_True
    goto Ini
:Condit_True
    inc U(01)
    goto Ini

end
```

Block number	Operation
1	If U(00) is 1, it jumps to the block 3: Condit_True. If U(00) is 0, it goes to the next block 2.
2	Jumps to the block 1: Ini unconditionally.
3	Adds 1 to U(01).
4	Jumps to the block 1: Ini unconditionally.

On Trip

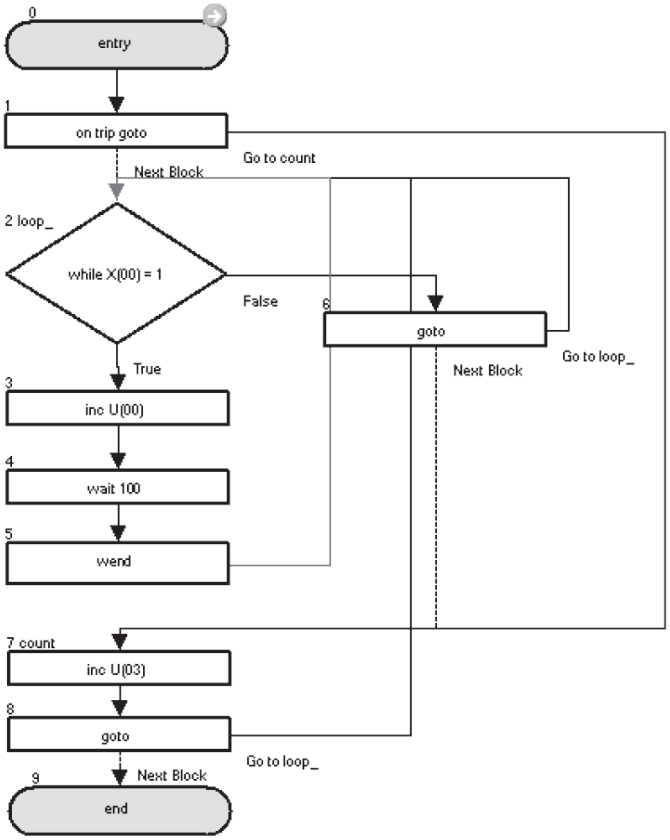
Command	Description	Argument
	Jumps to <label> when a trip occurs in the inverter.	**Label**: a name that is used to identify a particular function block in the task.

Format	
Flowchart method	**Text language method**
	on trip goto <label>

Note
1. The “on trip goto” command branches the processing when a trip occurs in the inverter.
2. Once the “on trip goto” command is executed, the execution log is saved in the inverter, and every scanned trip status is checked. When the program reaches the next step after a trip occurred, it jumps to the destination of “goto” command. If there is no trip, nothing is executed.
3. If the “end” command in the task is executed, the “on trip goto” command is canceled and the program does not branch even if a trip occurs.
4. Once the program branches with the “on trip goto” command, it will not branch again under any conditions. Be sure to restart the program after the program branches once with the “on trip goto” command.
5. To set the label name in flowchart, right-click the command and select [Set Label] from the menu. You can specify any name.

● Example

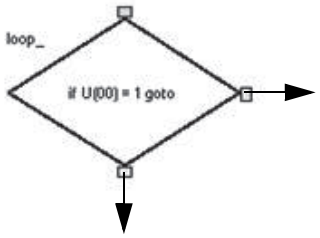
Flowchart



Text

```
entry
    on trip goto count
:loop_
    while X(00) = 1
        inc U(00)
        wait 100
    wend
    goto loop_
:count
    inc U(03)
    goto loop_
end
```

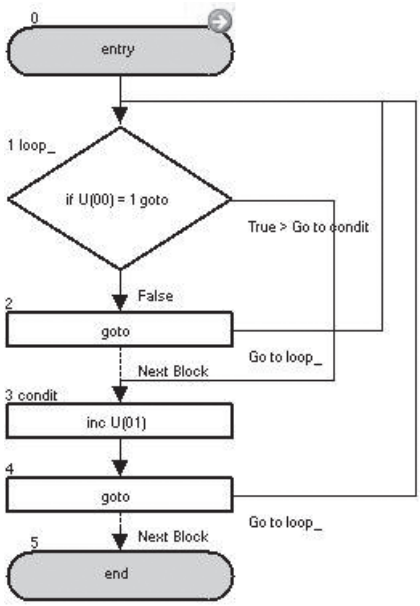
Block number	Operation
1	Executes the “on trip goto” command once so that the program jumps to the block 7: count if a trip occurs.
2 to 6	If X(00) is 1, it adds 1 to U(00), waits for 1.00 second with the “wait” command, and returns to the block 2: loop_. If X(00) is 0, jumps to the block 2: loop_ unconditionally.
7 to 8	Adds 1 to U(03) and jumps to the block 2: loop_ unconditionally.

If		
Command	**Description**	**Argument**
	Jumps to <label> when <condition> is met.	**Condition**: a comparison between two variables or constants with the format <left hand value> <comparison operator> <right hand value>. • Left hand value: any variable or constant (range −128 to 127) • Comparison operator: =, <, >, <=, >=, or <> • Right hand value: any variable or constant (range −128 to 127) **Label**: a name that is used to identify a particular function block in the task.
Format		
Flowchart method		**Text language method**
		if <condition> goto <label>

Note To set the label name in flowchart, right-click the command and select [Set Label] from the menu. You can specify any name.

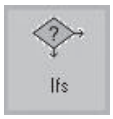
● Example

Flowchart	Text

	`entry`
	`:loop_`
	`  if U(00) = 1 goto condit`
	`  goto loop_`
	`:condit`
	`  inc U(01)`
	`  goto loop_`
	`end`

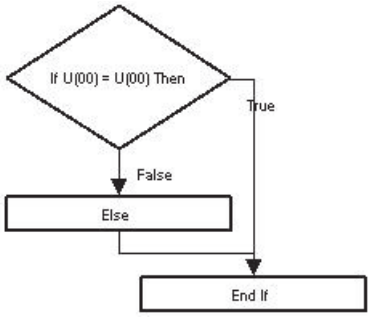
Block number	Operation
1	If U(00) is 1, it jumps to the block 3: condit. If U(00) is not 1, it goes to the next block 2.
2	Jumps to the block 1: loop_ unconditionally.
3 to 4	Adds 1 to U(01) and jumps to the block 1: loop_ unconditionally.

Ifs/Else/End If

Command	Description	Argument
   	If <condition> is met, it executes <command set 1> right after this command until the “else” command, and goes to the next step after the “endif” command. If <condition> is not met, it executes <command set 2> right after the “else” command until the “endif” command, and goes to the next step after the “endif” command.	**Condition**: a comparison between two variables or constants with the format <left hand value> <comparison operator> <right hand value>. • Left hand value: any variable or constant (range –128 to 127) • Comparison operator: =, <, >, <=, >=, or <> • Right hand value: any variable or constant (range –128 to 127) **Command set 1**: one or more commands until the “else” command. It can contain nested commands (up to eight levels). **Command set 2**: one or more commands until the “endif” command. It can contain nested commands (up to eight levels).

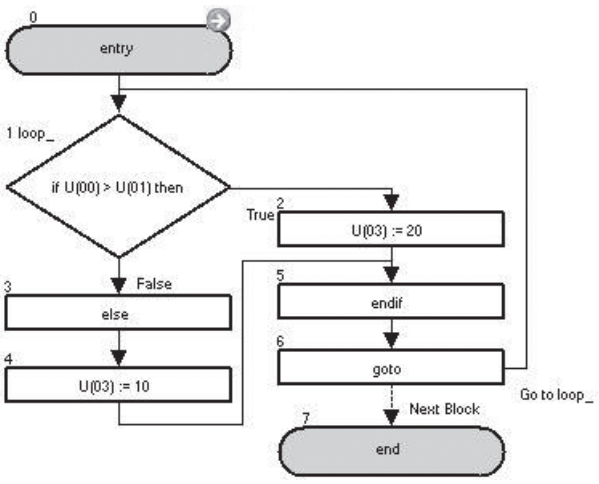
Format

Flowchart method	Text language method

	if <condition> then
	<command set 1>
	else
	<command set 2>
	endif

● Example

Flowchart	Text

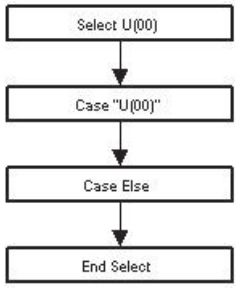
	`entry`
	`:loop_`
	`  if U(00) > U(01) then`
	`  U(03) := 20`
	`  else`
	`  U(03) := 10`
	`  endif`
	`  goto loop_`
	`end`

Block number	Operation
1 to 5	If U(00) is greater than U(01), it assigns 20 to U(03) right after the “if” and jumps to the “endif” command. If U(00) is not greater than U(01), it assigns 10 to U(03) right after the “else” command and goes to the “endif” command.
6	Jumps to the block 1: loop_ unconditionally.

Select/Case/Case Else/EndSelect

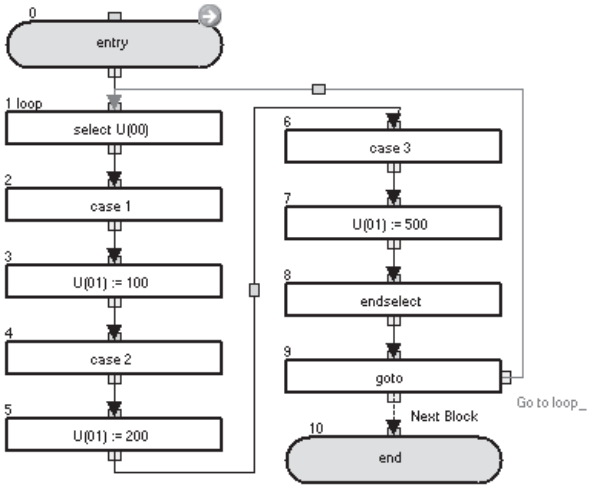
Command	Description	Argument
   	Allows the execution of multiple program sections depending on the variable value. Executes <command set n> when <condition variable> matches <condition value n> of any “case” command. Executes <command set for others> (“case else”) when <condition variable> does not match any of condition values of the “case” commands. This command is useful when you need to select among multiple options based on a <condition variable>. This command is also useful to organize programs when it is used with the “call” commands that calls subroutines.	**Condition variable**: a condition variable you use Condition value n: any variable or constant (range –128 to 127) **Command set n**: one or more commands until next “case” or “endselect” command. It can contain nested commands (up to eight levels).

Format

Flowchart method	Text language method
	select <condition variable> case <condition value 1> <command set 1> … case <condition value n> <command set n> … case else <command set for others> endselect

Note A "select” is counted as nesting level 1.
A set of "select” to “endselect” used in a subroutine is counted as nesting level 2.

● Example

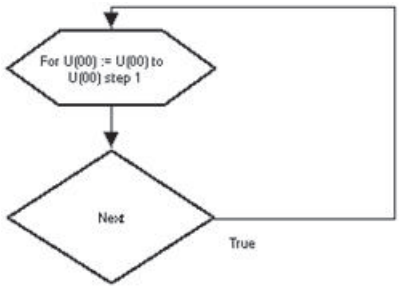
Flowchart	Text
	:loop_ select U(00) case 1 U(01) := 100 case 2 U(01) := 200 case 3 U(01) := 500 endselect goto loop_ end

Block number	Operation
1 to 8	Based on the value of U(00), 100 is assigned to U(01) if U(00) is 1, and 200 is assigned to U(01) if U(00) is 2. In the case 3, 500 is assigned to U(01) and the program jumps to the “endselect” command.
9	Jumps to the block 1: loop_ unconditionally.

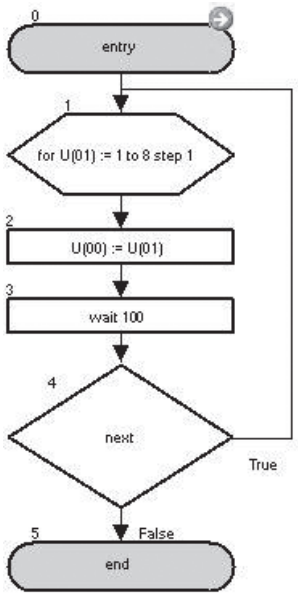
For/Next

Command	Description	Argument
	Executes <command set> repeatedly until <variable> started with <start value> reaches <end value>. In every time of execution, <incremental value> is added to <variable>. When the “next” command is executed, a judgment with <end value> is executed and <incremental value> is added to <variable>.	**Variable**: any variable **Start value**: initial value. This is the value assigned to the variable in the first loop. Constant (range −128 to 127) **End value**: a value to exit the loop. Constant (range −128 to 127) **Incremental value**: the variable is incremented by this value in each loop. Constant (range −128 to 127) **Command set**: one or more commands until the “next” command. It can contain nested commands (up to eight levels).

Format

Flowchart method	Text language method
	for <variable> : = <start value> to <end value> step <incremental value> <command set> next

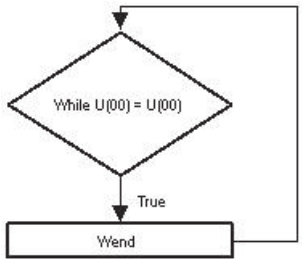
● Example

Flowchart	Text
	entry for U(01) := 1 to 8 step 1 U(00) := U(01) wait 100 next end

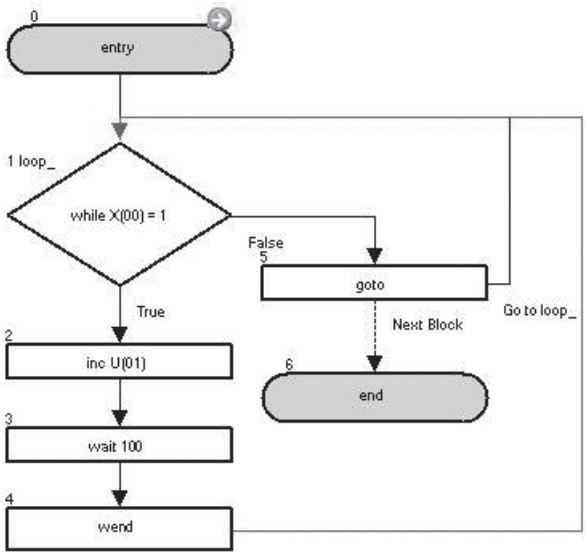
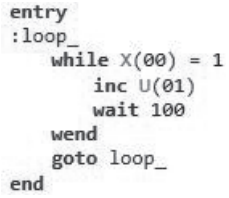
Block number	Operation
1 to 4	Assigns 1 to U(01) and the value of U(01) to U(00). Waits for 1.00 second with the “wait” command. If U(01) is less than 8, 1 is added to U(01) and the next command after “for” is executed. Goes to the next step after the “next” command if U(01) is equal to or greater than 8.

While/Wend

Command	Description	Argument
	Executes <command set> while a condition is met.	**Condition**: a comparison between two variables or constants with the format <left hand value> <comparison operator> <right hand value>. • Left hand value: any variable or constant (range −128 to 127) • Comparison operator: =, <, >, <=, >=, or <> • Right hand value: any variable or constant (range −128 to 127) **Command set**: one or more commands until the “wend” command. It can contain nested commands (up to eight levels).

Format	
Flowchart method	**Text language method**
	while <condition> <command set> wend

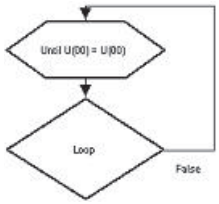
● Example

Flowchart	Text
	

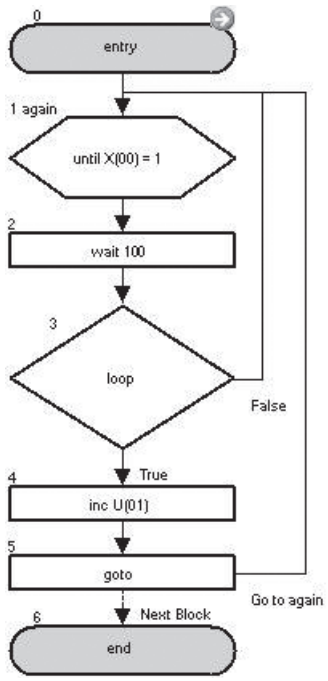
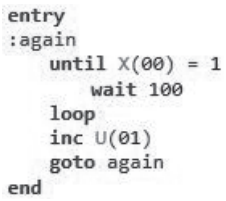
Block number	Operation
1 to 4	If X(00) is 1, it adds 1 to U(01), waits for 1.00 second with the “wait” command, and jumps to the “while” command. If X(00) is not 1, it executes the next command after the “wend” command.
5	Jumps to the block 1: loop_ unconditionally.

Until/Loop

Command	Description	Argument
	Executes <command set> until <condition> is met.	**Condition**: a comparison between two variables or constants with the format <left hand value> <comparison operator> <right hand value>. • Left hand value: any variable or constant (range –128 to 127) • Comparison operator: =, <, >, <=, >=, or <> • Right hand value: any variable or constant (range –128 to 127) **Command set**: one or more commands until the “loop” command. It can contain nested commands (up to eight levels).

Format	
Flowchart method	**Text language method**
	until <condition> <command set> loop

● Example

Flowchart	Text
	

Block number	Operation
1 to 3	If X(00) is not 1, it waits for 1.00 second with the “wait” command and jumps to the “until” command. If X(00) is 1, it executes the “loop” command and goes to the next command.
4 to 5	Adds 1 to U(00) and jumps to the block 1: again unconditionally.

Wait

Command	Description	Argument
 	Makes the program wait for specified seconds or until a condition is met.	**Value**: any variable or constant (specified time × 10 ms) Wait time value (0 to 32767 × 10 ms) **Condition**: a comparison between two variables or constants with the format <left hand value> <comparison operator> <right hand value>. • Left hand value: any variable or constant (range 0 to 127) • Comparison operator: =, <, >, <=, >=, or <> • Right hand value: any variable or constant (range 0 to 127)

Format	
Flowchart method	**Text language method**
wait <value> or <condition> ↓	wait <value> or <condition>

Note 1. With the “wait” command, the program does not go to next command and waits until the set wait time elapses or set condition is met. Therefore, the task processing is not executed during the wait time of the “wait” command. If there is any processing which requires periodic monitoring, separate it and assign it to another task so that it is executed periodically without using the “wait” command.

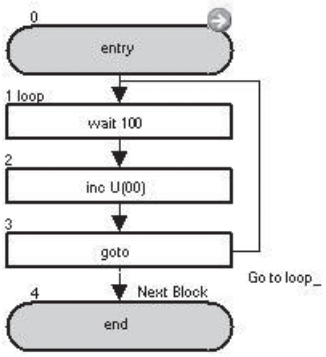
2. Do not set a negative value for the constant used in the “wait” command.

   Otherwise, the program cannot check if the wait time elapsed or the condition is met, and it continuously stops at the “wait” command without performing any operation.

3. The “wait” command is not an accurate way to measure time. To measure time accurately, use the internal timer or the clock function of the LCD Operator.

● A program example in which the wait time is set with <value>: the program waits at the “wait” command for the period of time set in <value>.

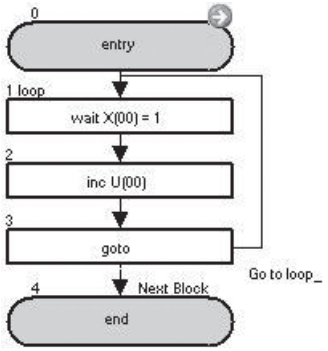
Flowchart	Text

	`entry`
	`:loop_`
	`  wait 100`
	`  inc U(00)`
	`  goto loop_`
	`end`

Block number	Operation
1	Waits for 1.00 second with the “wait” command.
2 to 3	Adds 1 to U(00) and unconditionally jumps to 1: loop_ to wait one more second.

- **A program example in which the condition to end the wait state is set in <condition>: the program waits until the condition is met.**

Flowchart	Text

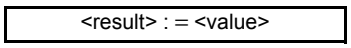
	`entry`
	`:loop_`
	`  wait X(00) = 1`
	`  inc U(00)`
	`  goto loop_`
	`end`

Block number	Operation
1	Waits until X(00) changes to 1.
2 to 3	Adds 1 to U(00) and jumps to the block 1: loop_ unconditionally.

6-5 Arithmetic Operation and Logical Operation Commands

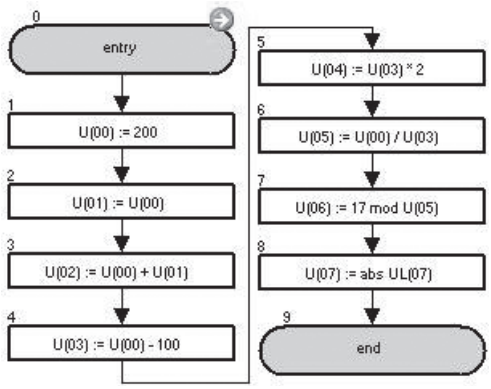
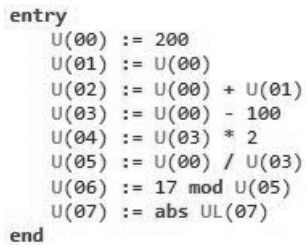
= (Substitution)

Command	Description	Argument
	Assigns <value> to <result>.	**Result**: any variable **Value**: any variable or constant (range –2,147,483,648 to 2,147,483,647)

Format	
Flowchart method	**Text language method**
	<result> : = <value>

Note If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart	Text
	


```
entry
    U(00) := 200
    U(01) := U(00)
    U(02) := U(00) + U(01)
    U(03) := U(00) - 100
    U(04) := U(03) * 2
    U(05) := U(00) / U(03)
    U(06) := 17 mod U(05)
    U(07) := abs UL(07)
end
```

Block number	Operation
1	Assigns 200 to U(00).
2	Assigns the value of U(00) to U(01).
3	Assigns the sum of U(00) and U(01) to U(02).
4	Assigns the difference between U(00) and 100 to U(03).
5	Assigns the product of U(03) and 2 to U(04).
6	Assigns the quotient of U(00) and U(03) to U(05).
7	Assigns the remainder after dividing 17 by U(05) to U(06).
8	Assigns the absolute value of UL(07) to U(07).

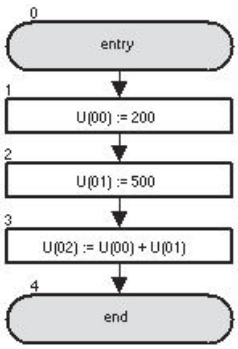
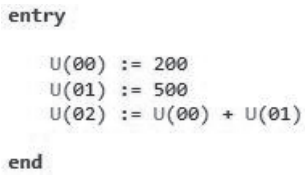
+ (Addition)

Command	Description	Argument
	Adds <value 1> and <value 2>.	**Result**: any variable **Value 1**: any variable or constant (range −128 to 127) **Value 2**: any variable or constant (range −2,147,483,648 to 2,147,483,647)

Format	
Flowchart method	**Text language method**
<result> : = <value 1> + <value 2> ↓	<result> : = <value 1> + <value 2>

Note If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart	Text
	


```
entry

    U(00) := 200
    U(01) := 500
    U(02) := U(00) + U(01)

end
```

Block number	Operation
1	Assigns 200 to U(00).
2	Assigns 500 to U(01).
3	Assigns the sum of U(00) and U(01) to U(02).

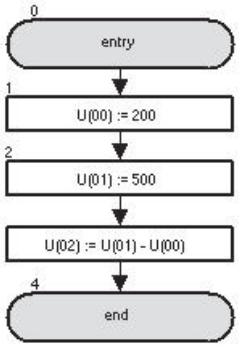
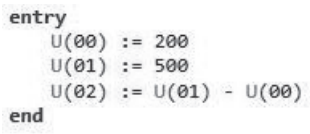
– (Subtraction)

Command	Description	Argument
	Subtracts <value 2> from <value 1>.	**Result**: any variable **Value 1**: any variable or constant (range –128 to 127) **Value 2**: any variable or constant (range –2,147,483,648 to 2,147,483,647)

Format	
Flowchart method	**Text language method**
<result> : = <value 1> – <value 2> ↓	<result> : = <value 1> – <value 2>

Note If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart	Text
	


```
entry
    U(00) := 200
    U(01) := 500
    U(02) := U(01) - U(00)
end
```

Block number	Operation
1	Assigns 200 to U(00).
2	Assigns 500 to U(01).
3	Assigns the difference between U(01) and U(00) to U(02).

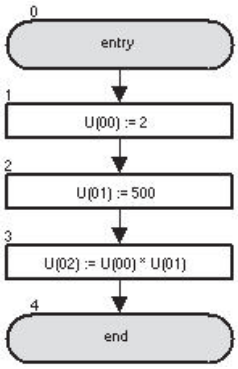
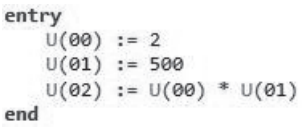
* (Multiplication)

Command	Description	Argument
	Multiplies <value 1> by <value 2>.	**Result**: any variable **Value 1**: any variable or constant (range –128 to 127) **Value 2**: any variable or constant (range –2,147,483,648 to 2,147,483,647)

Format	
Flowchart method	**Text language method**
<result> : = <value 1> * <value 2> ↓	<result> : = <value 1> * <value 2>

Note If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart	Text
	


```
entry
   U(00) := 2
   U(01) := 500
   U(02) := U(00) * U(01)
end
```

Block number	Operation
1	Assigns 2 to U(00).
2	Assigns 500 to U(01).
3	Assigns the product of U(00) and U(01) to U(02).

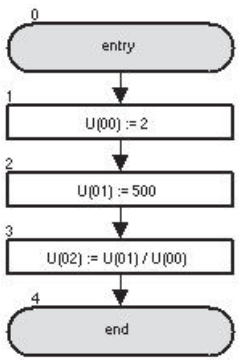
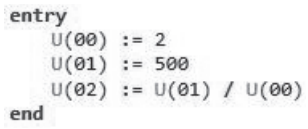
/ (Division)

Command	Description	Argument
	Divides <value 1> by <value 2>.	**Result**: any variable **Value 1**: any variable or constant (range –128 to 127) **Value 2**: any variable or constant (range –2,147,483,648 to 2,147,483,647)

Format	
Flowchart method	**Text language method**
<result> : = <value1> / <value2> ↓	<result> : = <value1> / <value2>

Note If an overflow, underflow, or division by zero occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart	Text
	


```
entry
    U(00) := 2
    U(01) := 500
    U(02) := U(01) / U(00)
end
```

Block number	Operation
1	Assigns 2 to U(00).
2	Assigns 500 to U(01).
3	Assigns the quotient of U(01) and U(00) to U(02).

Mod (Modulo-division)

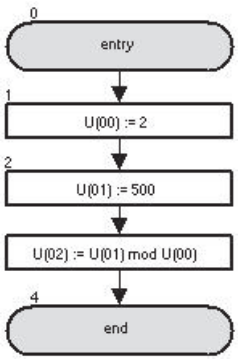
Command	Description	Argument
	Remainder for division of <value 1> by <value 2>.	**Result**: any variable **Value 1**: any variable or constant (range –128 to 127) **Value 2**: any variable or constant (range –2,147,483,648 to 2,147,483,647)

Format	
Flowchart method	**Text language method**
<result> : = <value 1> mod <value 2> ↓	<result> : = <value 1> mod <value 2>

Note If an overflow, underflow, or division by zero occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart	Text

	`entry`
	`  U(00) := 2`
	`  U(01) := 500`
	`  U(02) := U(01) mod U(00)`
	`end`

Block number	Operation
1	Assigns 2 to U(00).
2	Assigns 500 to U(01).
3	Assigns the remainder for division of U(01) and U(00) to U(02).

Abs (Absolute value)

Command	Description	Argument
	Finds the absolute value of <value>.	**Result**: any variable **Value**: any variable or constant (Range –2,147,483,647 to 2,147,483,647)

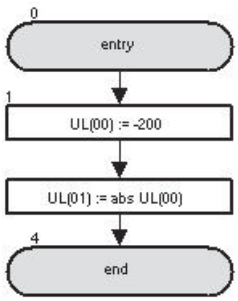
Format	
Flowchart method	**Text language method**
<result> : = abs <value> ↓	<result> : = abs <value>

Note 1. If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

2. If the variable or the constant is the maximum negative value, –2,147,483,648, it is not possible to convert it to an absolute value. The value –2,147,483,648 is recognized as a negative number as it is.

● Example

Flowchart	Text

	`entry`
	`  UL(00) := -200`
	`  UL(01) := abs UL(00)`
	`end`

Block number	Operation
1	Assigns –200 to UL(00).
2	Assigns the absolute value of UL(00) to UL(01).

And (Logical AND)

Command	Description	Argument
	Logical AND on <value 1> and <value 2> in binary format and (Logical AND)	**Result**: any variable **Value 1**: any variable or constant (range –128 to 127) **Value 2**: any variable or constant (range –2,147,483,648 to 2,147,483,647)

Value 1	Value 2	Result
0	0	0
0	1	0
1	0	0
1	1	1

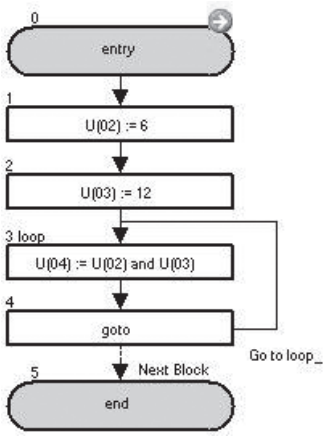
Format

Flowchart method	Text language method
<result> : = <value 1> and <value 2> ↓	<result> : = <value 1> and <value 2>

Note If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart



Text

```
entry
    U(02) := 6
    U(03) := 12
:loop_
    U(04) := U(02) and U(03)
    goto loop_
end
```

Block number	Operation
1	Assigns 6 to U(02).
2	Assigns 12 to U(03).
3	Assigns 4 (binary: 00000100), which is the result of logical AND on 6 (U(02) binary: 00000110) and 12 (U(03) binary: 00001100), to U(04).
4	Jumps to the block 3: loop_ unconditionally.

Or (Logical OR)

Command	Description	Argument
	Logical OR on <value 1> and <value 2> in binary format or (Logical OR)	**Result**: any variable **Value 1**: any variable or constant (range –128 to 127) **Value 2**: any variable or constant (range –2,147,483,648 to 2,147,483,647)

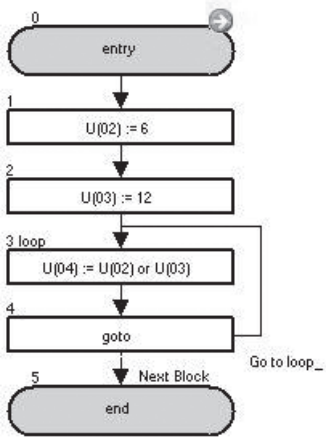
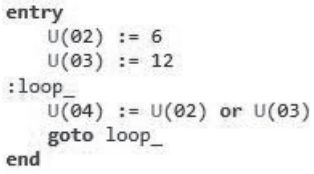
Value 1	Value 2	Result
0	0	0
0	1	1
1	0	1
1	1	1

Format

Flowchart method	Text language method
<result> : = <value 1> or <value 2> ↓	<result> : = <value 1> or <value 2>

Note If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart	Text
	


```
entry
    U(02) := 6
    U(03) := 12
:loop_
    U(04) := U(02) or U(03)
    goto loop_
end
```

Block number	Operation
1	Assigns 6 to U(02).
2	Assigns 12 to U(03).
3	Assigns 14 (binary: 00001110), which is the result of logical OR on 6 (U(02) binary: 00000110) and 12 (U(03) binary: 00001100), to U(04).
4	Jumps to the block 3: loop_ unconditionally.

XOr (Logical exclusive OR)

Command	Description	Argument
	Logical exclusive OR on <value 1> and <value 2> in binary format. XOr (Logical exclusive OR)	**Result**: any variable **Value 1**: any variable or constant (range –128 to 127) **Value 2**: any variable or constant (range –2,147,483,648 to 2,147,483,647)

Value 1	Value 2	Result
0	0	0
0	1	1
1	0	1
1	1	0

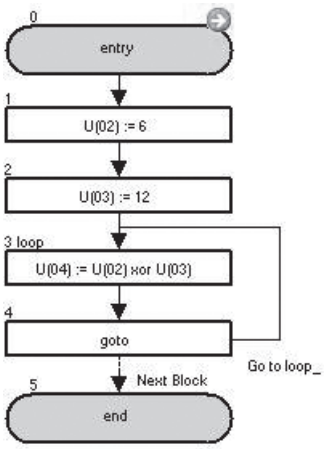
Format

Flowchart method	Text language method
<result> : = <value 1> xor <value 2> ↓	<result>: = <value 1> xor <value 2>

Note If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart

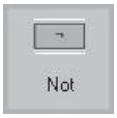


Text

```
entry
    U(02) := 6
    U(03) := 12
:loop_
    U(04) := U(02) xor U(03)
    goto loop_
end
```

Block number	Operation
1	Assigns 6 to U(02).
2	Assigns 12 to U(03).
3	Assigns 10 (binary: 00001010), which is the result of logical exclusive OR on 6 (U(02) binary: 00000110) and 12 (U(03) binary: 00001100), to U(04).
4	Jumps to the block 3: loop_ unconditionally.

Not (Negation in binary format)

Command	Description	Argument
	Negation on <value 1> in binary format (bit reversal) not (Negation)	**Result**: any variable except variables with bit data size*1 **Value**: any variable except variables and constants with bit data size*1 (range –2,147,483,648 to 2,147,483,647)

Value 1	Result
0	1
1	0

Format

Flowchart method	Text language method
<result> : = not <value> ↓	<result> : = not <value>

*1. You cannot obtain a correct result with commands such as UB(1) = not UB(0).

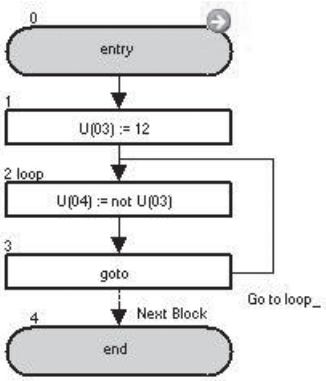
Use “xor” command to reverse variables with bit data size as shown in the following examples.

- Example 1: UB(1) = UB(0) xor 1
- Example 2: UB(2) = X(00) xor 1

Note If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart



Text

```
entry
    U(03) := 12
:loop_
    U(04) := not U(03)
    goto loop_
end
```

Block number	Operation
1	Assigns 12 to U(03).
2	Assigns 65523 (binary: 111110011), which is the result of negation (bit reversal) on 12 (U(03) binary: 00000110), to U(04).
3	Jumps to the block 3: loop_ unconditionally.

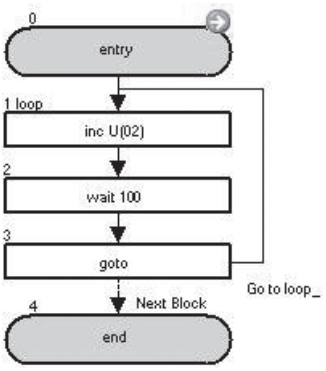
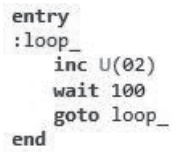
Inc (Increment by 1)

Command	Description	Argument
	Increments <value> by 1.	**Value**: any variable

Format	
Flowchart method	**Text language method**
inc <value> ↓	inc <value>

Note If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

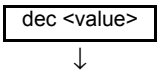
● Example

Flowchart	Text
	

Block number	Operation
1	Adds 1 to U(02).
2	Waits for 1.00 second with the “wait” command.
3	Jumps to the block 1: loop_ unconditionally.

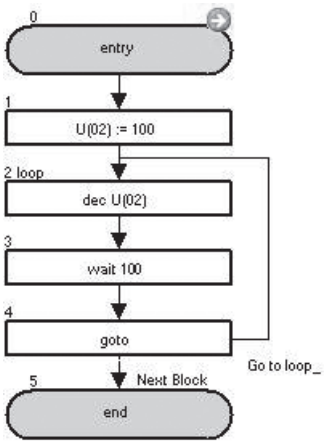
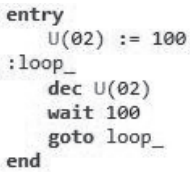
Dec (Decrement by 1)

Command	Description	Argument
	Decrements <value> by 1.	**Value**: any variable

Format	
Flowchart method	**Text language method**
	dec <value>

Note If an overflow or underflow occurs, the DriveProgramming detects it as an error. Take necessary measures in the application so that they do not occur.

● Example

Flowchart	Text
	

Block number	Operation
1	Assigns 100 to U(02).
2	Subtracts 1 from U(02).
3	Waits for 1.00 second with the “wait” command.
4	Jumps to the block 2: loop_ unconditionally.

6-6 I/O Control Commands

Use these I/O control commands to control inputs and outputs. Although you can control I/Os with the = (Assignment) command, the I/O control commands can efficiently use the program capacity because their arguments require smaller data size.

var = X(i)		
Command	**Description**	**Argument**
	Assigns one bit of the status of the input terminal variable to <variable>.	**variable**: any variable (the variable value is 0 or 1) i: input terminal variable (range 00 to 07)

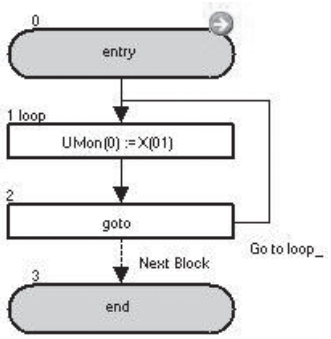
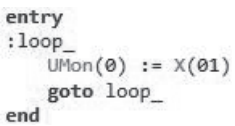
Format	
Flowchart method	**Text language method**
<variable> : = X(i) ↓	<variable> : = X(i)

Note The input terminal variable is a variable that detects the status of the inverter's input terminal. The following settings are required. The numerical order of input terminal variables follows the numerical order of the set general-input numbers.
Set the Input Terminal 1 to 9, A and B Selection (CA-01 to CA-11) to MI1 to MI11 (86 to 96).

<Assignment example>
X(00) = MI1 (function No. 86)
X(01) = MI2 (function No. 87)
X(02) = MI3 (function No. 88)
X(03) = MI4 (function No. 89)
X(04) = MI5 (function No. 90)
X(05) = MI6 (function No. 91)
X(06) = MI7 (function No. 92)
X(07) = MI8 (function No. 93)
X(08) = MI9 (function No. 94)
X(09) = MI10 (function No. 95)
X(10) = MI11 (function No. 96)

Note For details, refer to *5-2 Input/Output Terminal Variables* on page 5-5.

● Example

Flowchart	**Text**
	


```
entry
:loop_
   UMon(0) := X(01)
   goto loop_
end
```

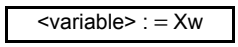
In the above example, the status of the input terminal X(01) is monitored with the parameter UMon(0) (db-08).

Block number	**Operation**
1	Assigns X(01) to UMon(0).

Block number	Operation
2	Jumps to the block 1: loop_ unconditionally.

var = Xw

Command	Description	Argument
	Assigns the status of the input terminal variable to <variable> in units of words.	**Variable**: any variable

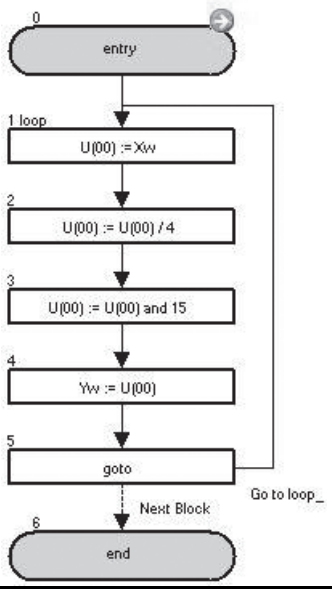
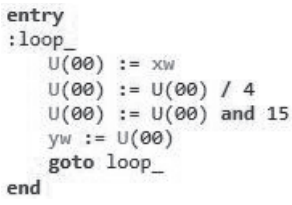
Format	
Flowchart method	**Text language method**
	<variable> : = Xw

Note The input terminal variable is a variable that detects the status of the inverter's input terminal. The following settings are required. The numerical order of input terminal variables follows the numerical order of the set general-input numbers.
Set the Input Terminal 1 to 9, A and B Selection (CA-01 to CA-11) to 86 to 96 (MI1 to MI11: General-purpose input).

When the data is assigned in units of words, zero is read out for the upper byte data and unset input terminal variables.

<Assignment example>
X(00) = MI1 (function No. 86) to Xw = 1 (bit 0)
X(01) = MI2 (function No. 87) to Xw = 2 (bit 1)
X(02) = MI3 (function No. 88) to Xw = 4 (bit 2)
X(03) = MI4 (function No. 89) to Xw = 8 (bit 3)
X(04) = MI5 (function No. 90) to Xw = 16 (bit 4)
X(05) = MI6 (function No. 91) to Xw = 32 (bit 5)
X(06) = MI7 (function No. 92) to Xw = 64 (bit 6)
X(07) = MI8 (function No. 93) to Xw = 128 (bit 7)
X(08) = MI9 (function No. 94) to Xw = 256 (bit 8)
X(09) = MI10 (function No. 95) to Xw = 518 (bit 9)
X(10) = MI11 (function No. 96) to Xw = 1036 (bit 10)
For details, refer to *5-2 Input/Output Terminal Variables* on page 5-5.

● Example

Flowchart	Text
	


```
entry
:loop_
    U(00) := xw
    U(00) := U(00) / 4
    U(00) := U(00) and 15
    yw := U(00)
    goto loop_
end
```

In the above example, the status of the input terminals X(02) to X(05) is captured and output to the output terminals Y(00) to Y(03).

Block number	Operation
1	Assigns the value of Xw (value of input terminal) to U(00).

Block number	Operation
2	Divides the value of U(00) by 4 (2-bit right shift) to assign X(02) to bit 0.
3	Performs a logical AND operation on U(00) and 15 (binary: 00001111) and changes the bits higher than X(06) to zero.
4	Assigns U(00) to Yw.
5	Jumps to the block 1: loop_ unconditionally.

Y(i) = value

Command	Description	Argument
	Outputs data to the output terminal variable in units of bits.	**i**: output terminal variable (range 0 to 6) **Value**: any variable or constant

Format	
Flowchart method	**Text language method**
Y(i) : = <value> ↓	Y(i) : = <value>

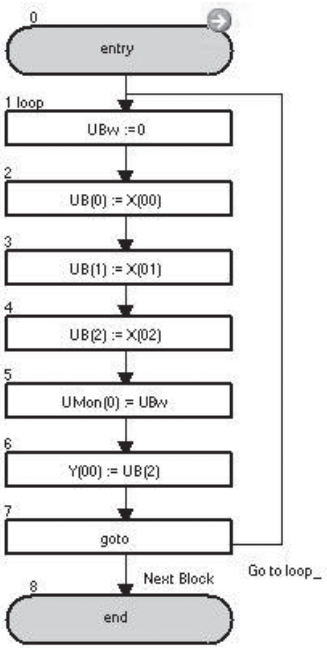
Note The output terminal variable is a variable that controls the status of the inverter's output terminal. The following settings are required. The numerical order of the output terminal variables follows the numerical order of the set general-output numbers.
Set the Output Terminal 1 to 5 Selection (CC-01 to CC-05) and the Relay Output (16, AL) Selection (CC-06, CC-07) to 69 to 75 (MO1 to MO7: General-purpose output).

<Assignment example>
Y(00) = MO1 (General-purpose output 1 No. 69)
Y(01) = MO2 (General-purpose output 2 No. 70)
Y(02) = MO3 (General-purpose output 3 No. 71)
Y(03) = MO4 (General-purpose output 4 No. 72)
Y(04) = MO5 (General-purpose output 5 No. 73)
Y(05) = MO6 (General-purpose output 6 No. 74)
Y(06) = MO7 (General-purpose output 7 No. 75)

For details, refer to *5-2 Input/Output Terminal Variables* on page 5-5.

● Example

Flowchart	Text

	`entry`
	`:loop_`
	`ubw := 0`
	`UB(0) := X(00)`
	`UB(1) := X(01)`
	`UB(2) := X(02)`
	`UMon(0) := ubw`
	`Y(00) := UB(2)`
	`goto loop_`
	`end`

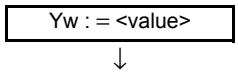
In the above example, the status of X(00) to X(02) is monitored with the parameter UMon(0) (db-08), and X(02) is output to Y(00).

Block number	Operation
1	Assigns 0 to UBw.

Block number	Operation
2 to 4	Assigns the variables X(00) to X(02) to the variables UB(0) to UB(2).
5	Assigns UBw to UMon(0).
6	Assigns UB(2) to Y(00).
7	Jumps to the block 1: loop_ unconditionally.

Yw = value

Command	Description	Argument
	Outputs data to the output terminal variable in units of words. Reflect each bit in corresponding output.	**Value**: any variable or constant

Format	
Flowchart method	**Text language method**
	Yw : = <value>

Note The output terminal variable is a variable that controls the status of the inverter's output terminal. The following settings are required. The numerical order of the output terminal variables follows the numerical order of the set general-output numbers.

Set the Output Terminal 1 to 5 Selection (CC-01 to CC-05) and the Relay Output (16, AL) Function Selection (CC-06, CC-07) to 69 to 75 (MO1 to MO7: General-purpose output).

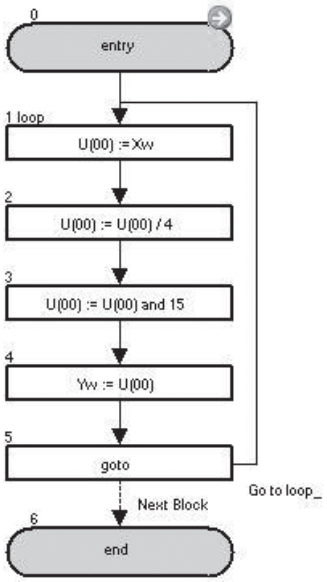
When the data is assigned in units of words, zero is read out for the upper byte data and unset input terminal variables. If there is any setting, the setting is ignored.

<Assignment example>
Yw = 1 (bit 0) to Y(00) = MO1 (General-purpose output 1 No. 69)
Yw = 2 (bit 1) to Y(01) = MO2 (General-purpose output 2 No. 70)
Yw = 4 (bit 2) to Y(02) = MO3 (General-purpose output 3 No. 71)
Yw = 8 (bit 3) to Y(03) = MO4 (General-purpose output 4 No. 72)
Yw = 16 (bit 4) to Y(04) = MO5 (General-purpose output 5 No. 73)
Yw = 32 (bit 5) to Y(05) = MO6 (General-purpose output 6 No. 74)
Yw = 64 (bit 6) to Y(06) = MO7 (General-purpose output 7 No. 75)

For details, refer to *5-2 Input/Output Terminal Variables* on page 5-5.

● Example

Flowchart	Text

	`entry`
	`:loop_`
	`U(00) := xw`
	`U(00) := U(00) / 4`
	`U(00) := U(00) and 15`
	`yw := U(00)`
	`goto loop_`
	`end`

In the above example, the status of the input terminals X(02) to X(05) is captured and output to the output terminals Y(00) to Y(03).

Block number	Operation
1	Assigns the value of Xw (value of input terminal) to U(00).

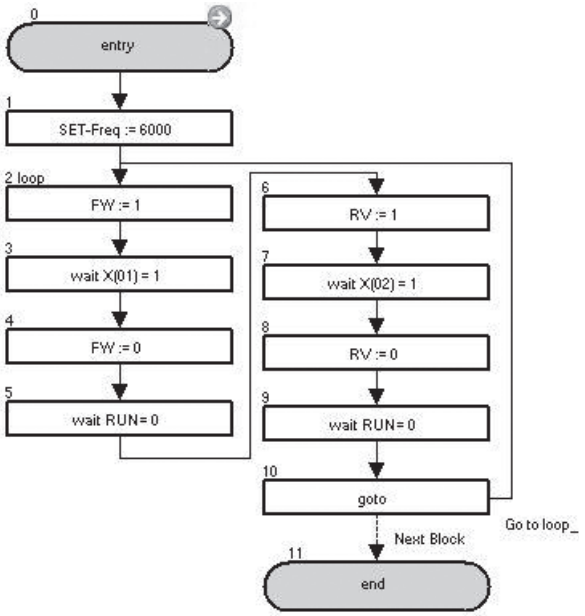
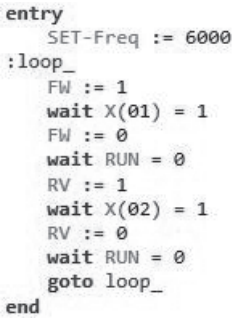
Block number	Operation
2	Divides the value of U(00) by 4 (2-bit right shift) to assign X(02) to bit 0.
3	Performs a logical AND operation on U(00) and 15 (binary: 00001111) and changes the bits higher than X(06) to zero.
4	Assigns U(00) to Yw.
5	Jumps to the block 1: loop_ unconditionally.

func = value

Command	Description	Argument
	Assigns <value> to the input variable.	**Function**: any function bit of the input variable (refer to *5-6 Input Variables* on page 5-18 for details on each bit.) **Value**: any variable or constant

Format	
Flowchart method	**Text language method**
<function> : = <value> ↓	<function> : = <value>

● Example

Flowchart	Text
	

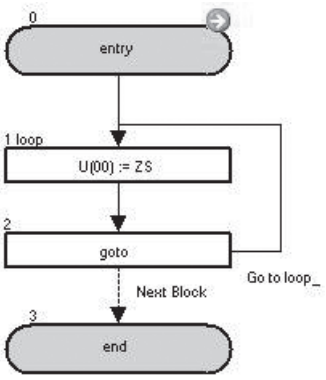
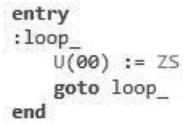

```
entry
    SET-Freq := 6000
:loop_
    FW := 1
    wait X(01) = 1
    FW := 0
    wait RUN = 0
    RV := 1
    wait X(02) = 1
    RV := 0
    wait RUN = 0
    goto loop_
end
```

In the above example, the following operation is repeated: a forward operation is performed at 60.00 Hz until X(01) changes to ON. Then, a reverse operation is performed until X(02) changes to ON.

Block number	Operation
1	Sets the output frequency to 60.00 Hz.
2	Operates the inverter in the forward direction.
3 to 5	Waits until X(01) changes to ON, stops the inverter and waits with the “wait” command until the inverter completely stops.
6	Operates the inverter in the reverse direction.
7 to 9	Waits until X(02) changes to ON, stops the inverter and waits with the “wait” command until the inverter completely stops.
10	Jumps to the block 2: loop_ unconditionally.

var = func		
Command	**Description**	**Argument**
	Assigns the status of the output variable to <variable>.	**Variable**: any variable **Function**: any bit of the output variable
Format		
Flowchart method		**Text language method**
<variable> : = <function> ↓		<variable> : = <function>
For details, refer to *5-7 Output Variables* on page 5-21.		

● Example

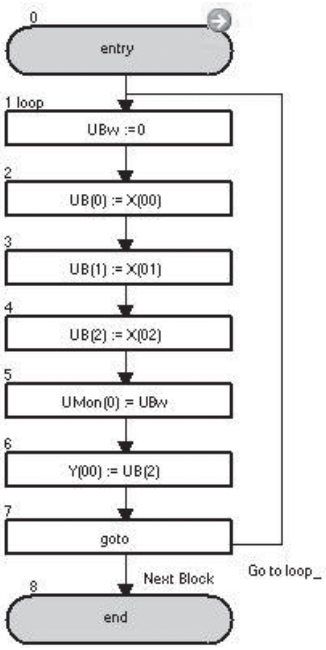
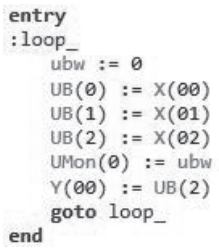
Flowchart	Text
	

In the above example, 1 is assigned to U(00) when ZS (0 Hz detection signal) is ON, and 0 is assigned to U(00) when ZS (0 Hz detection signal) is OFF.

Block number	Operation
1	Assigns the status of ZS to U(00).
2	Jumps to the block 1: loop_ unconditionally.

var = UB(i)		
Command	**Description**	**Argument**
	Assigns one bit of the value of the internal user contact to <variable>.	**Variable**: any variable (value of the variable is 0 or 1.) **i**: internal user contact number (range 0 to 7)
Format		
Flowchart method		**Text language method**
<variable> : = UB(i) ↓		<variable> : = UB(i)

● Example

Flowchart	Text
	

In the above example, the status of X(00) to X(02) is monitored with the parameter UMon(0) (db-08), and X(02) is output to Y(00).

Block number	Operation
1	Assigns 0 to UBw.
2 to 4	Assigns the variables X(00) to X(02) to the variables UB(0) to UB(2).
5	Assigns UBw to UMon(0).
6	Assigns UB(2) to Y(00).
7	Jumps to block 1: loop_ unconditionally.

var = UBw

Command	Description	Argument
	Assigns the value of the internal user contact to <variable> in units of words.	**Variable**: any variable

Format

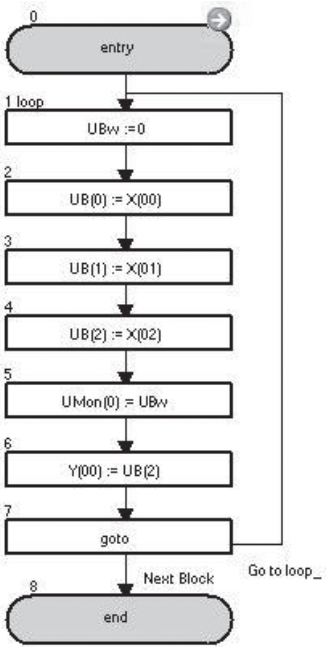
Flowchart method	Text language method
<variable> : = UBw ↓	<variable> : = UBw

Note When the data is assigned in units of words, zero is read out for the unused upper byte. If there is any setting, the setting is ignored.

UB(0) to UBw = 1 (bit 0)
UB(1) to UBw = 2 (bit 1)
UB(2) to UBw = 4 (bit 2)
UB(3) to UBw = 8 (bit 3)
UB(4) to UBw = 16 (bit 4)
UB(5) to UBw = 32 (bit 5)
UB(6) to UBw = 64 (bit 6)
UB(7) to UBw = 128 (bit 7)
UB(8) to UBw = 256 (bit 8)
UB(9) to UBw = 512 (bit 9)
UB(10) to UBw = 1024 (bit 10)
UB(11) to UBw = 2048 (bit 11)
UB(12) to UBw = 4096 (bit 12)
UB(13) to UBw = 8192 (bit 13)
UB(14) to UBw = 16384 (bit 14)
UB(15) to UBw = 32768 (bit 15)

● Example

Flowchart



Text

```
entry
:loop_
    ubw := 0
    UB(0) := X(00)
    UB(1) := X(01)
    UB(2) := X(02)
    UMon(0) := ubw
    Y(00) := UB(2)
    goto loop_
end
```

In the above example, the status of X(00) to X(02) is monitored with the parameter UMon(0) (db-08), and X(02) is output to Y(00).

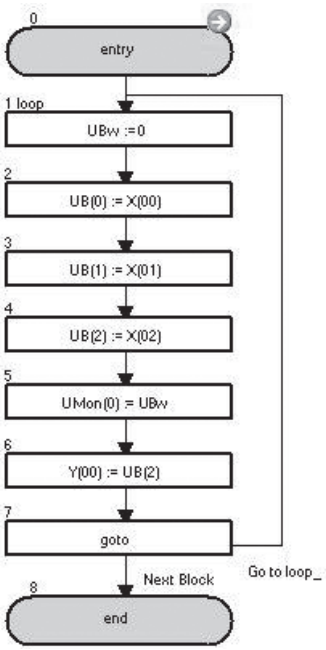
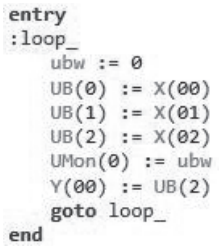
Block number	Operation
1	Assigns 0 to UBw.
2 to 4	Assigns the variables X(00) to X(02) to the variables UB(0) to UB(2).
5	Assigns UBw to UMon(0).
6	Assigns UB(2) to Y(00).
7	Jumps to the block 1: loop_ unconditionally.

UB(i) = value

Command	Description	Argument
	Assigns <value> to the internal user contact.	**i**: internal user contact number (range 0 to 7) **Value**: any variable or constant

Format	
Flowchart method	**Text language method**
UB(i) : = <value> ↓	UB(i) : = <value>

● Example

Flowchart	Text
	

In the above example, the status of X(00) to X(02) is monitored with the parameter UMon(0) (db-08), and X(02) is output to Y(00).

Block number	Operation
1	Assigns 0 to UBw.
2 to 4	Assigns the variables X(00) to X(02) to the variables UB(0) to UB(2).
5	Assigns UBw to UMon(0).
6	Assigns UB(2) to Y(00).
7	Jumps to the block 1: loop_ unconditionally.

UBw = value

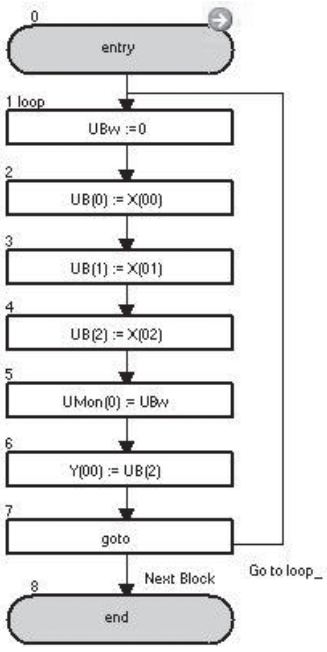
Command	Description	Argument
	Assigns <value> to the internal user contact in units of words.	**Value**: any variable or constant

Format	
Flowchart method	**Text language method**
UBw : = <value> ↓	UBw : = <value>

Note When the data is assigned in units of words, zero is read out for the unused upper byte. If there is any setting, the setting is ignored.

UB(0) to UBw = 1 (bit 0)
UB(1) to UBw = 2 (bit 1)
UB(2) to UBw = 4 (bit 2)
UB(3) to UBw = 8 (bit 3)
UB(4) to UBw = 16 (bit 4)
UB(5) to UBw = 32 (bit 5)
UB(6) to UBw = 64 (bit 6)
UB(7) to UBw = 128 (bit 7)
UB(8) to UBw = 256 (bit 8)
UB(9) to UBw = 512 (bit 9)
UB(10) to UBw = 1024 (bit 10)
UB(11) to UBw = 2048 (bit 11)
UB(12) to UBw = 4096 (bit 12)
UB(13) to UBw = 8192 (bit 13)
UB(14) to UBw = 16384 (bit 14)
UB(15) to UBw = 32768 (bit 15)

● Example

Flowchart	Text
	entry :loop_ ubw := 0 UB(0) := X(00) UB(1) := X(01) UB(2) := X(02) UMon(0) := ubw Y(00) := UB(2) goto loop_ end

In the above example, the status of X(00) to X(02) is monitored with the parameter UMon(0) (db-08), and X(02) is output to Y(00).

Block number	Operation
1	Assigns 0 to UBw.
2 to 4	Assigns the variables X(00) to X(02) to the variables UB(0) to UB(2).
5	Assigns UBw to UMon(0).
6	Assigns UB(2) to Y(00).
7	Jumps to the block 1: loop_ unconditionally.

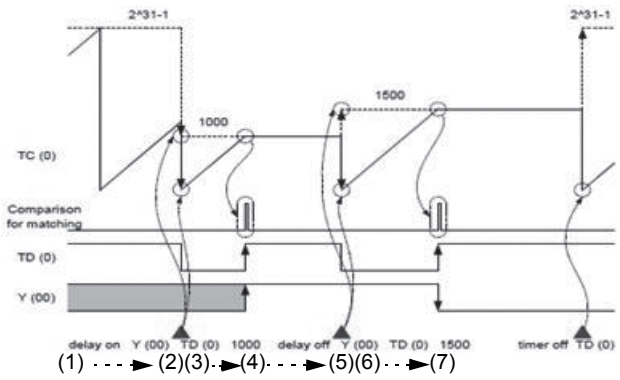
6-7 Timer Control Commands

Delay on/off		
Command	**Description**	**Argument**
	Starts the timer(k). The the timer counter TC(k) is started with 0 and incremented every 10 ms until it reaches <value 2>. When <value 2> is reached, the timer contact TD(k) changes to ON and the processing (on or off) specified for <value 1> is executed.	**on/off**: operation setting (on/off) after the delay time **Value 1**: any contact variable or variable **Value 2**: any variable or constant (specified time × 10 ms) **TD(k)**: the timer output contact of the timer that you use (range of k is 0 to 7)
Format		
Flowchart method		**Text language method**
delay on/off <value 1> TD(k) <value 2> ↓		delay on/off <value 1> TD(k) <value 2>

Note 1. The timer (k) is started when the “delay on/off” command is executed. If TD(k) is ON, it is changed to OFF when the timer (k) is started. However, the variable of <value 1> does not change from its original value.
After the timer (k) is started, the program goes to the next command.

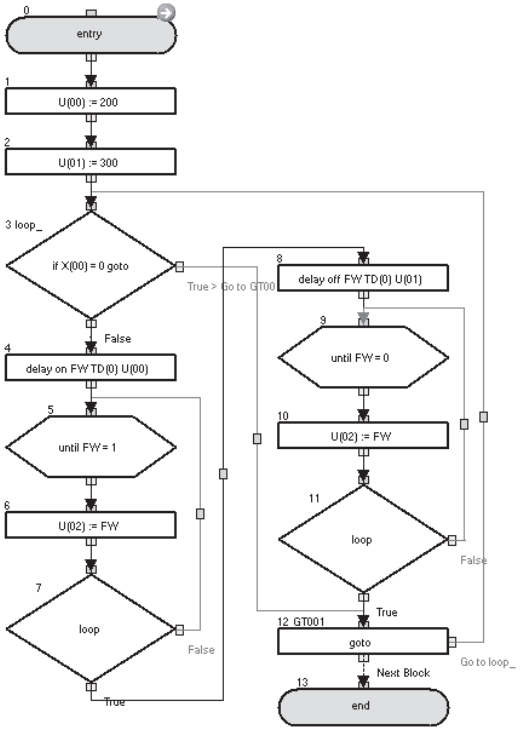
2. When the “delay on/off” command is executed, the data of <value 1>, TD(k) and <value 2> are saved internally. Even if the variable values set to <value 1> and <value 2> are changed after the “delay on/off” command is executed, the operation of the command does not change.
3. After the “delay on/off” command is executed, if the same timer (k) is restarted before its processing is completed, the ongoing processing is canceled and the timer (k) is started with the new settings. Therefore, create a program so that the timer (k) which is started once is not restarted until its processing is completed.
4. You can monitor the data of the started timer (k) with the timer counter variable TC(k). Check the completion of the timer processing with the timer output contact TD(k) (changes ON at completion).

● Timing chart



(1) The timer is in a free-run state.
(2) The “delay on” command is started.
(3) The delay operation is in progress.
(4) The time elapsed.
(5) The “delay off” command is started.
(6) The delay operation is in progress.
(7) The time elapsed.

● Example

Flowchart	Text
	entry U(00) := 200 U(01) := 300 :loop_ if X(00) = 0 goto GT001 delay on FW TD(0) U(00) until FW = 1 U(02) := FW loop delay off FW TD(0) U(01) until FW = 0 U(02) := FW loop :GT001 goto loop_ end

In the above example, when X(00) is not 0, a forward operation is started with the “delay on” command and the operation is stopped with the “delay off” command.

Block number	Operation
1	Assigns 200 to U(00).
2	Assigns 300 to U(01).
3	Jumps to the block 8: GT001 if X(00) is 0. Goes to the next step if X(00) is not 0.
4	Starts the timer (0) for the set value U(00) with the “delay on” command.
5 to 7	Assigns the status of FW to U(02) until FW changes to 1. Goes to the next step when FW changes to 1.
8	Starts the timer (0) for the set value U(01) with the “delay off” command.
9 to 11	Assigns the status of FW to U(02) until FW changes to 0. Goes to the next step when FW changes to 0.
12	Jumps to the block 3: loop_ unconditionally.

Timer set

Command	Description	Argument
	Starts the timer (k). The the timer counter TC(k) is started with 0 and incremented every 10 ms until it reaches <value>. When <value> is reached, the timer contact TD(k) changes to ON.	**TD(k)**: the timer output contact of the timer that you use (range of k is 0 to 7) **Value**: any variable or constant (specified time × 10 ms)

Format	
Flowchart method	**Text language method**
timer set TD(k) <value> ↓	timer set TD(k) <value>

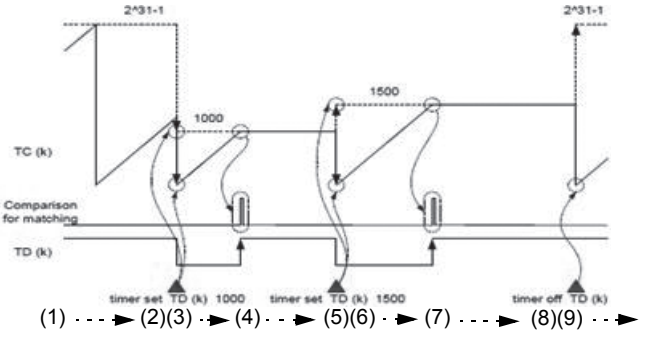
Note 1. The timer (k) is started when the “timer set” command is executed. If TD(k) is ON, it is changed to OFF when the timer (k) is started. After the timer (k) is started, the program goes to the next command.

2. When the “timer set” command is executed, the data of TD(k) and <value> are saved internally. Even if the variable values set to <value> is changed after the “timer set” command is executed, the operation of the command does not change.

3. After the “timer set” command is executed, if the same timer (k) is restarted before its processing is completed, the ongoing processing is canceled and the timer (k) is started with the new settings. Therefore, create a program so that the timer (k) which is started once is not restarted until its processing is completed.

4. You can monitor the data of the started timer (k) with the timer counter variable TC(k). Check the completion of the timer processing with the timer output contact TD(k) (changes ON at completion).

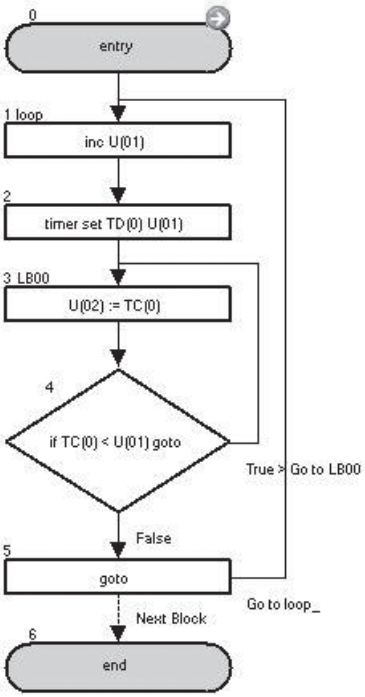
● Timing chart



(1) The timer is in a free-run state.
(2) The “timer set” command is started.
(3) The delay operation is in progress.
(4) The time elapsed.
(5) The “timer set” command is started.
(6) The delay operation is in progress.
(7) The time elapsed.
(8) The “timer off” command is started.
(9) The timer is in a free-run state.

● Example

Flowchart	Text

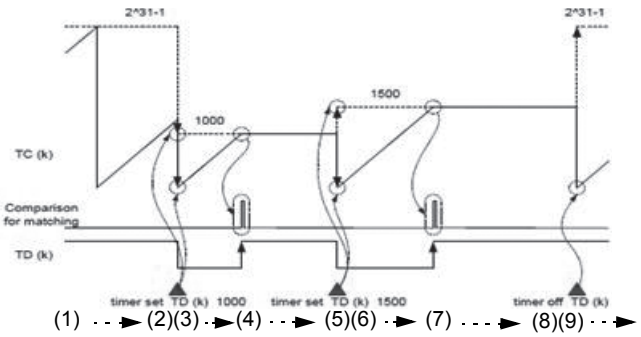
	`entry`
	`:loop_`
	`  inc U(01)`
	`  timer set TD(0) U(01)`
	`:LB00`
	`  U(02) := TC(0)`
	`  if TC(0) < U(01) goto LB00`
	`  goto loop_`
	`end`

In the above example, the set value of TD(0) is incremented by 1 every time the timer execution is completed, and the time required for each loop gets longer every execution. The current value of the timer is reflected in U(02).

Block number	Operation
1	Adds 1 to U(01).
2	Starts the timer (0) for the “timer set” command with the set value U(01).
3	Assigns the current value TC(0) of the timer (0) to U(02).
4	Jumps to the block 3: LB00 if the current value TC(0) of the timer (0) is less than the set value U(01). In other cases, the program goes to the next step.
5	Jumps to the block 1: loop_ unconditionally.

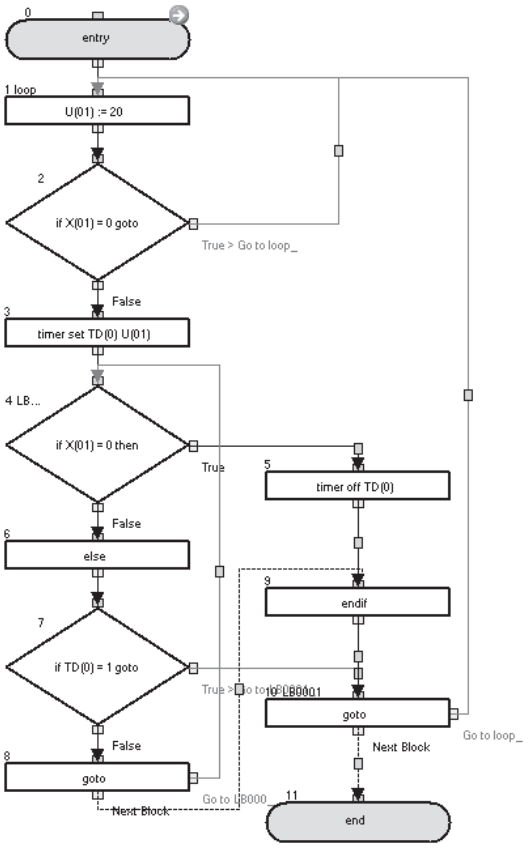
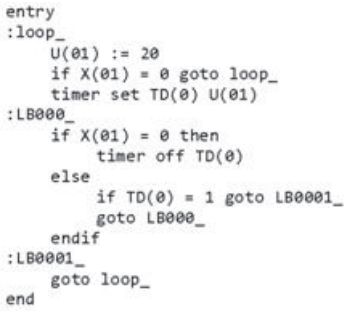
Timer Off		
Command	**Description**	**Argument**
	Resets the timer counter TC(k) to 0 and starts the timer in free-run mode.	**TD(k)**: the timer output contact of the timer that you use (range of k is 0 to 7)
Format		
Flowchart method		**Text language method**
timer off TD(k) ↓		timer off TD(k)

● Timing chart



(1) The timer is in a free-run state.
(2) The “timer set” command is started.
(3) The delay operation is in progress.
(4) The time elapsed.
(5) The “timer set” command is started.
(6) The delay operation is in progress.
(7) The time elapsed.
(8) The “timer off” command is started.
(9) The timer is in a free-run state.

● Example

Flowchart	Text
	


```
entry
:loop_
     U(01) := 20
     if X(01) = 0 goto loop_
     timer set TD(0) U(01)
:LB000_
     if X(01) = 0 then
          timer off TD(0)
     else
          if TD(0) = 1 goto LB0001_
          goto LB000_
     endif
:LB0001_
     goto loop_
end
```

In the above example, the start/end operation of the timer (0) is repeated when X(01) is not 0. If X(01) changes to 0, the timer is stopped and X(01) is monitored.

Block number	Operation
1	Assigns 20 to U(01).
2	Jumps to the block 1: loop_ if X(01) is 0. Goes to the next step if X(01) is not 0.
3	Starts the timer (0) for the “timer set” command with the set value U(01).
4 to 9	Monitors X(01) and jumps to the block 10: LB0001_ when TD(0) is 1 and X(01) is not 0. When X(01) changes to 0, the timer (0) stops and the program goes to the next step.
10	Jumps to the block 1: loop_ unconditionally.

6-8 Parameter Control Commands

ChgParam		
Command	**Description**	**Argument**
	Changes the data of the inverter parameter specified in <parameter> to <value>. You can change any inverter parameter.	**Parameter**: parameter code (Fxxx, Axxx, bXXX, Cxxx, Hxxx, or Pxxx) **Value**: any variable or constant

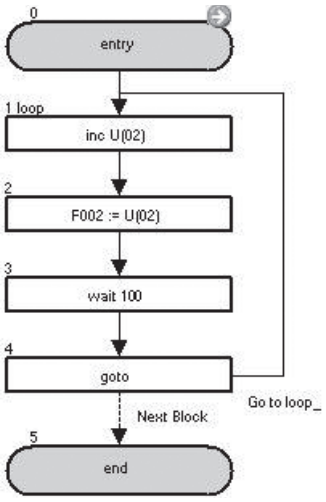
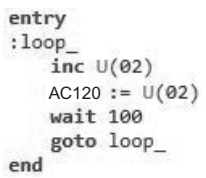
Format	
Flowchart method	**Text language method**
<parameter> : = <value> ↓	<parameter> : = <value>

Note 1. Similar to the settings with the inverter's LCD Operator, each parameter has the following restrictions. If any of the event listed in the restrictions occurs, the inverter detects Command Error (E045) and the DriveProgramming program is stopped. For details on restrictions for each parameter, refer to the High-function General-purpose Inverter 3G3RX2 Series User's Manual (I620).

- Data that exceeds the setting range was set to the parameter.
- A matching error occurred between the parameter to set and relevant parameters.
- A function which is not available with the selected mode was set.
- The parameter that cannot be changed during operation was changed during operation.
- A parameter was set while parameter data change is prohibited with the Soft Lock Selection (UA-16).

2. Even if you change the set parameter data by using the “ChgParam” command, the data of the corresponding parameter is not changed in the EEPROM.

● Example

Flowchart	Text
	

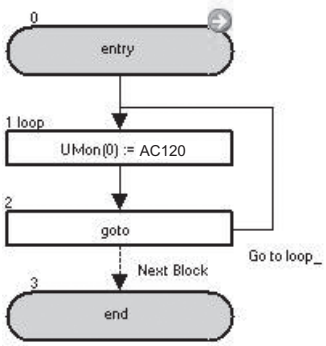
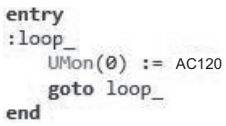

```
entry
:loop_
    inc U(02)
    AC120 := U(02)
    wait 100
    goto loop_
end
```

In the above example, the value of AC120 (1st Acceleration Time 1) is incremented by 1 every second.

Block number	Operation
1	Adds 1 to U(02).
2	Assigns U(02) to the parameter AC120.
3	Waits for 1.00 second with the “wait” command.
4	Jumps to the block 1: loop_ unconditionally.

MonParam		
Command	**Description**	**Argument**
	Assigns the content of the inverter parameter specified in <parameter> to <variable>.	**Parameter**: parameter code (Fxxx, Axxx, bxxx, Cxxx, dxxx, Hxxx, or Pxxx) **Variable**: any variable
Format		
Flowchart method		**Text language method**
<variable> : = <parameter> ↓		<variable> : = <parameter>

● Example

Flowchart	Text
	


```
entry
:loop_
    UMon(0) := AC120
    goto loop_
end
```

In the above example, the value of the parameter AC120 (Acceleration time setting 1, 1st-motor) is assigned to the parameter UMon(0) (db-08) (User Monitor) and monitored.

Block number	Operation
1	Assigns the parameter FA-01 to UMon(0).
2	Jumps to the block 1: loop_ unconditionally.

RtcSet

Command	Description	Argument
	Sets a 6-byte clock data sent from the LCD Operator in the variable. This data corresponds to year, month, day, day of the week, hour and minute. The hexadecimal variable value corresponds to year, month, day, day of the week, hour and minute (decimal). rtcset on: continuously updates the 6-byte data. rtcset off: updates the 6-byte data only once.	**On/off**: continuous/once **User variable**: any user parameter variable or internal user variable U(xx) or UL(xx)

Format

Flowchart method	Text language method
rtcset on/off <user variable> ↓	rtcset on/off <user variable>

Note 1. This function uses the clock function of the LCD Operator. If you execute the “rtcset on/off” command when the LCD Operator is not connected, its processing does not finish and the whole program enters a waiting state for the command. Therefore, we recommend you to create a program in which the “rtcset on” command is started only once right after the start of the program and the clock data is always output to the specified variable.

2. When the LCD Operator is shipped, its clock time is set to the default data (2000/1/1/SUN 00:00). Set the clock time accurately before you use the “rtcset on/off” command. Also, remember that the clock function has a time error (per month: −1.5 to 1.5 minutes) when you use this command.
3. The BCD data is output in units of bytes when the “rtcset on/off” command is executed.
   The days of the week, Sunday to Saturday are represented as 00 to 06, respectively. However, you cannot detect the BCD data using the user parameter variable U(k) or the internal user variable. When you create a program, note that the BCD data is detected as hexadecimal data and converted to the decimal data in the DriveProgramming Editor and the LCD Operator.
4. When the “rtcset on/off” U(k) is executed, the data is set in the following order from upper bytes.
   You must ensure continuous three user parameter variables including U(k).

User parameter variable	Upper byte	Lower byte
U(k)	Year BCD data	Month BCD data
U(k+1)	Day BCD data	Day of the week BCD data
U(k+2)	Hour BCD data	Minute BCD data

5. When the “rtcset on/off” UL(k) is executed, the data is set in the following order from upper bytes.
   You must ensure continuous two internal user variables including UL(k).

User parameter variable	Upper word		Lower word	
	Upper byte	Lower byte	Upper byte	Lower byte
UL(k)	Year BCD data	Month BCD data	Day BCD data	Day of the week BCD data
UL(k+1)	Hour BCD data	Minute BCD data	0 (padding)	0 (padding)

6. You must perform the installation and removal of the LCD Operator with the inverter power supply shut off. Otherwise, a fault may occur.
   If an error that causes disconnection from the LCD Operator occurs, the clock data is retained for at least 1 minute and 45 seconds. Therefore, it is possible to execute the “rtcset on/off” command within 1 minute and 45 seconds after disconnection. Although, when the time exceeds 1 minute and 45 seconds, the clock function processing does not finish and the whole program enters a waiting state for the command. When the connection with the LCD Operator is restored, the program goes to the next step. To prevent the whole program from entering a waiting state, create a program so that the “rtcset on” command is started only once when the program is started and the specified variable is always output to the clock data. Do not execute the “rtcset on/off” command for checking time in the program. Instead, use the specified variable. When the time exceeds 1 minute and 45 seconds after disconnection, an all-zero data is output to the specified variable.
7. When the battery of the LCD Operator gets weak, the Operator cannot maintain the clock data any longer. The data is maintained with the control circuit power supply while the power supply for the inverter is on. However, when the power supply is turned on again, the clock data returns to the initial data (2000/1/1 SUN 00:00).

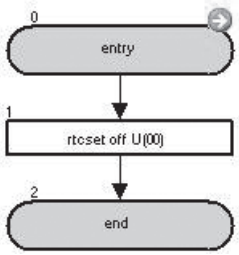
Precautions for Safe Use

If the clock command is used in DriveProgramming, an unexpected operation may occur due to weak battery. Take measures such as detecting a weak battery by [E042]RTC Error and stopping the inverter or programs. When the LCD Operator is removed or disconnected, DriveProgramming is in a waiting status by the clock command.

When the LCD Operator is removed or disconnected, DriveProgramming is in a waiting status by the clock command.

● Example

Flowchart	Text

	`entry`
	`  rtcset off U(00)`
	`end`

If you execute this program example on October 18th (Thursday) of 2012 at 2:29 P.M., then U(00), U(01) and U(02) will be displayed on the DriveProgramming Editor and the LCD Operator as follows:

User parameter variable	Clock function BCD data (actual hexadecimal data)	Display (converts hexadecimal BCD data to decimal)	Meaning of clock function BCD data
U(00)	1210	4624	"12" for year 2012, "10" for October
U(01)	1804	6148	"18" for 18th day, "04" for Thursday
U(02)	1429	5161	"14" for 2 p.m., "29" for 29 minutes

6-9 Inverter Control Commands

Run FW

Command	Description	Argument
	Makes the inverter start a forward operation. This command is a shortcut of the “func = value” command which is previously set as “FW : = 1”.	---

Format	
Flowchart method	**Text language method**
FW : = 1 ↓	FW : = 1

Run RV

Command	Description	Argument
	Makes the inverter start a reverse operation. This command is a shortcut of the “func = value” command which is previously set as “RV : = 1”.	---

Format	
Flowchart method	**Text language method**
RV : = 1 ↓	RV : = 1

Stop FW

Command	Description	Argument
	Makes the inverter decelerate to stop from a forward operation. This command is a shortcut of the “func = value” command which is previously set as “FW : = 0”.	---

Format	
Flowchart method	**Text language method**
FW : = 0 ↓	FW : = 0

Stop RV

Command	Description	Argument
	Makes the inverter decelerate to stop from a reverse operation. This command is a shortcut of the “func = value” command which is previously set as “RV : = 0”.	---

Format	
Flowchart method	**Text language method**
RV : = 0 ↓	RV : = 0

Stop (func = value)

Command	Description	Argument
	Makes the operating inverter decelerate to stop by using the “func = value” command. Refer to *"func = value" in 6-6 I/O Control Commands* on page 6-40. <value> is assigned to the selected <function> variable.	**function**: select either of the following variables. ”FW”: In a forward operation by the “Run FW” ”RV”: In a reverse operation by the “Run RV” **value**: set 0 (stop) for deceleration to stop the inverter.

Format

Flowchart method

(Forward operation) FW : = 0 or (Reverse Operation) RV : = 0

↓ ↓

Text language method

(Forward operation)	(Reverse Operation)
FW : = 0	RV : = 0

Precautions for Correct Use

- If the DriveProgramming program is stopped, the status of the variables FW (forward) and RV (reverse) is not retained but cleared to zero.
- The variable FW (forward) and RV (reverse) are enabled only when the inverter's 1st RUN Command Selection (AA111) is set to 01 ([FW]/[RV] terminal). The operation is not performed with other setting options.
- If you set the variable FW (forward) or RV (reverse) to 1 immediately after turning on the power supply, the setting is ignored and neither forward nor reverse operation is performed. Set 0 first, and then set 1 again. To avoid this operation, create a program that has one second of wait time with such as “wait” command after turning on the power supply.

Stop

Command	Description	Argument
	Makes the operating inverter decelerate and stop. When a trip is detected by the inverter, this command acts as a reset.	---

Format	
Flowchart method	**Text language method**
stop ↓	stop

Note 1. The “stop” command has the same function as the STOP/RESET key of the LCD Operator.

This command makes the operating inverter decelerate and stop. When a trip is detected by the inverter, this command acts as a reset.

2. To prevent the “stop” command from acting as a reset, set the STOP-key enable at RUN-command from terminal, 1st-motor (AA-13) to 00 (Disabled). With this setting, while the “stop” command makes the inverter decelerate and stop, a reset is not performed. However, note that the LCD Operator's STOP/RESET key will be disabled.
3. Since the “stop” command may act as a reset and cause the LCD Operator's STOP/RESET key to be disabled, we recommend you to use the previously mentioned Stop (func = value) command to stop the inverter.

Set Freq

Command	Description	Argument
	Sets the frequency of the Inverter. This command is a = (Assign) command whose left-hand side is set to the frequency reference variable SET-Freq. Unit: 0.01 Hz	**Value**: any variable or constant (range 0 to 40,000)

Format	
Flowchart method	**Text language method**
SET-Freq : = <value> ↓	SET-Freq : = <value>

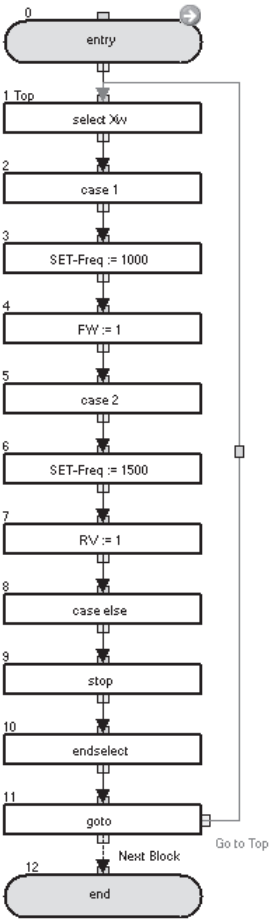
Note 1. The range of frequency that the inverter can actually output is from the Minimum frequency adjustment, 1st-motor (Hb130) to the maximum frequency.

If the set data is out of range, the inverter operates as follows.

- Minimum frequency adjustment, 1st-motor (Hb130)
  Operates at the frequency of 0.00 Hz.
  When the 1st Control Method (AA121) is set to 09 (Zero-Hz range sensorless vector control (IM)) or 10 (Vector control with sensor (IM)), the Minimum frequency adjustment, 1st-motor (Hb130) is disabled and the specified frequency is output.
- More than maximum frequency
  Limits the frequency reference to the value set in the 1st/2nd Maximum Frequency (Hb105/Hd105/Hb205/Hd205).

2. When the DriveProgramming program is stopped, the data of the frequency reference variable before the program stop is retained. When the program execution is started again, the process begins with the retained data.

● Example

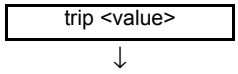
Flowchart	Text
	entry :Top select Xw case 1 SET-Freq := 1000 FW := 1 case 2 SET-Freq := 1500 RV := 1 case else stop endselect goto Top end

In the above example, if the general-input contact Xw is 1, the motor runs at 10 Hz in the forward direction. If the general-input contact Xw is 2, the motor runs at 15 Hz in the reverse direction. With other values, the motor stops.

Block number	Operation
1	After the "select" command, the program branches to the following “case” commands depending on the value of Xw. Branches to the “case 1” if Xw = 1, to the “case 2” if Xw = 2, and to the “case else” if Xw = a value other than 1 and 2.
2 to 4	Executes if Xw = 1. Sets the frequency reference variable to 10.00 Hz, operates the inverter in the forward direction, and executes the next steps after the “endselect” command.
5 to 7	Executes if Xw = 2. Sets the frequency reference variable to 15.00 Hz, operates the inverter in reverse direction, and executes the next steps after “endselect” command.
8 to 10	Executes if Xw = a value other than 1 and 2. Stops the inverter and executes the next steps after the “endselect” command.
11	Jumps to the block 1: Top unconditionally.

Trip

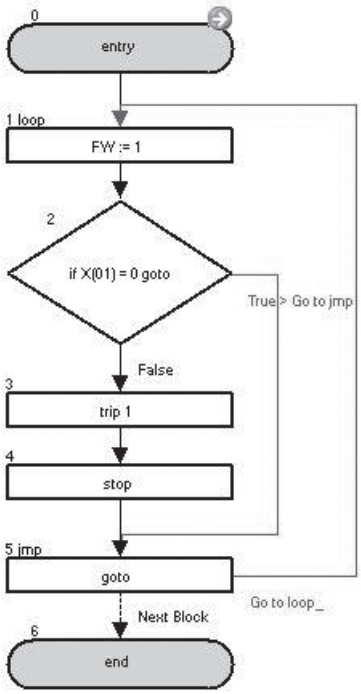
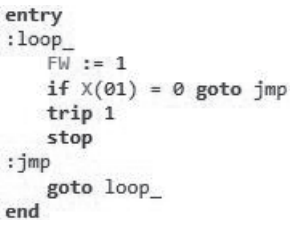
Command	Description	Argument
	Generates the inverter trip. The inverter stops when it detects a trip.	**Value**: any variable or constant (range 0 to 9) 0 to 9 correspond to the alarm code E050 to E059.

Format	
Flowchart method	**Text language method**
	trip <value>

Note 1. Even if an inverter trip occurs in one task, the DriveProgramming program does not stop the other possible tasks in the program, and the processing is continued.

2. You can set 10 or greater values by using any variable and constant (up to 127), however, even if a trip occurs, the displayed error code will always be E059.

● Example

Flowchart	Text
	

```
entry
:loop_
    FW := 1
    if X(01) = 0 goto jmp
    trip 1
    stop
:jmp
    goto loop_
end
```

In the above example, a user trip occurs in the inverter when the input terminal variable X(01) is set to ON.

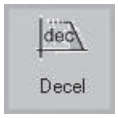
Block number	Operation
1	Operates the inverter in the forward direction.
2	If X(01) is 0, jumps to the block 5: jmp. If X(01) is not 0, the program goes to the next step.
3	Generates the user trip 1 in the inverter.
4	Stops the inverter.
5	Jumps to the block 1: loop_ unconditionally.

Accel

Command	Description	Argument
	Sets the inverter acceleration time. This command is a =(Assign) command whose left-hand side is set to the acceleration time variable ACCEL. Unit: 10 ms	**Value**: any variable or constant

Format	
Flowchart method	**Text language method**
ACCEL : = <value> ↓	ACCEL : = <value>

Decel

Command	Description	Argument
	Sets the inverter deceleration time. This command is a =(Assign) command whose left-hand side is set to the deceleration time variable DECEL. Unit: 10 ms	**Value**: any variable or constant

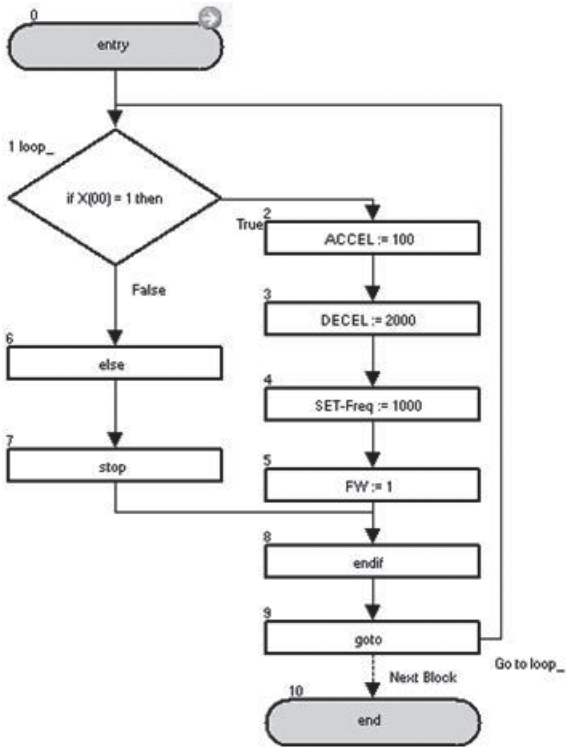
Format	
Flowchart method	**Text language method**
DECEL : = <value> ↓	DECEL : = <value>

Precautions for Correct Use

- The acceleration time variable ACCEL and the deceleration time variable DECEL are enabled only when you set the Acceleration/Deceleration Time Input Type (AC-01) to 04 (DriveProgramming).
- When the DriveProgramming program is stopped, the data of the acceleration time variable ACCEL and deceleration time variable DECEL before the program stop is retained. When the program execution is started again, the process begins with the retained data.

● Example

Flowchart	Text

	`entry`
	`:loop_`
	`  if X(00) = 1 then`
	`  accel := 100`
	`  decel := 2000`
	`  set-freq := 1000`
	`  fw := 1`
	`  else`
	`  stop`
	`  endif`
	`  goto loop_`
	`end`

In the above example, the acceleration time is set to 1.00 second and the deceleration time is set to 20.00 seconds when the input terminal variable X(00) is set to ON.

Block number	Operation
1 to 6	If X(00) is 1, the acceleration time is set to 1.00 second, the deceleration time is set to 20.00 seconds, and the output frequency is set to 10.00 Hz. The inverter is operated in the forward direction. If X(00) is not 1, the program goes to the next step after the “else” command.
7 to 8	Stops the inverter.
9	Jumps to the block 1: loop_ unconditionally.

7

Precautions for Use of Parameters for DriveProgramming

This section describes the precautions for use of parameters for the DriveProgramming.

7-1 Inverter Parameters Affected by Setting Order

In some cases, the setting range of the parameters are restricted by the setting data of other parameters. The following are the representative examples.

For details, refer to the High-function General-purpose Inverter 3G3RX2 Series User's Manual (I620).

Parameter No.	Description	Data
Hb104	Async.Motor Base frequency setting, 1st-motor	10.00 to Maximum Frequency [Hb105](Hz)
Hb105	Async.Motor Maximum frequency setting, 1st-motor	Base Frequency [Hb104] to 590.00 (Hz)
Hd104	Sync.Base frequency setting, 1st-motor	10.00 to Maximum Frequency [Hd105](Hz)
Hd105	Sync.Maximum frequency setting, 1st-motor	Base Frequency [Hd104] to 590.00 (Hz)
Hb204	Async.Motor Base frequency setting, 2nd-motor	10.00 to Maximum Frequency [Hb205](Hz)
Hb205	Async.Motor Maximum frequency setting, 2nd-motor	Base Frequency [Hb204] to 590.00 (Hz)
Hd204	Sync.Base frequency setting, 2nd-motor	10.00 to Maximum Frequency [Hd205](Hz)
Hd205	Sync.Maximum frequency setting, 2nd-motor	Base Frequency [Hd204] to 590.00 (Hz)
AA104	Sub speed setting, 1st-motor	0.00/Minimum Frequency [Hb130][Hd130] to Maximum Frequency [Hb105][Hd105](Hz)
AA204	Sub speed setting, 2nd-motor	0.00/Minimum Frequency [Hb230][Hd230] to Maximum Frequency [Hb205][Hd205](Hz)
Ab110	Multispeed-0 setting, 1st-motor	0.00/Minimum Frequency [Hb130][Hd130] to Maximum Frequency [Hb105][Hd105](Hz)
Ab210	Multispeed-0 setting, 2nd-motor	0.00/Minimum Frequency [Hb230][Hd230] to Maximum Frequency [Hb205][Hd205](Hz)
Ab-11 to Ab-25	Multispeed-1 setting to 15 setting	0.00/Minimum Frequency [Hb130][Hd130] to Maximum Frequency [Hb105][Hd105](Hz)
bA-60	Dynamic brake usage rate	0.0 to 10.0 × (Dynamic brake usage rate resistance [bA-63]/Minimum resistance)^2(%)
bA102	Upper Frequency limit, 1st-motor	Upper Frequency limit, 1st-motor [bA102] ≤ Async.Motor Maximum frequency setting, 1st-motor [Hb105] Upper Frequency limit, 1st-motor [bA102] ≤ Sync.Maximum frequency setting, 1st-motor[Hd105]
bA103	Lower Frequency limit, 1st-motor	Lower Frequency limit, 1st-motor [bA103] ≤ Async.Motor Maximum frequency setting, 1st-motor [Hb105] Lower Frequency limit, 1st-motor [bA103] ≤ Sync.Maximum frequency setting, 1st-motor[Hd105]
bA202	Upper frequency limit, 2nd motor	Upper frequency limit, 2nd motor [bA202] ≤ Async.Motor Maximum frequency setting, 2nd-motor [Hb205] Upper frequency limit, 2nd motor [bA202] ≤ Sync.Maximum frequency setting, 2nd-motor [Hd205]

Parameter No.	Description	Data
bA203	Lower frequency limit, 2nd motor	Lower frequency limit, 2nd motor [bA203] ≤ Async.Motor Maximum frequency setting, 2nd-motor [Hb205] Lower frequency limit, 2nd motor [bA203] ≤ Sync.Maximum frequency setting, 2nd-motor [Hd205]
bC120	Free electronic thermal frequency-1, 1st-motor	0.00 to Free electronic thermal frequency-2, 1st-motor [bC122](Hz)
bC122	Free electronic thermal frequency-2, 1st-motor	Free electronic thermal frequency-1, 1st-motor [bC120] to Free electronic thermal frequency-3, 1st-motor [bC124](Hz)
bC124	Free electronic thermal frequency-3, 1st-motor	Free electronic thermal frequency-2, 1st-motor [bC122] to 590.00(Hz)
bC220	Free electronic thermal frequency-1, 2nd-motor	0.00 to Free electronic thermal frequency-2, 2nd-motor [bC222](Hz)
bC222	Free electronic thermal frequency-2, 2nd-motor	Free electronic thermal frequency-1, 2nd-motor [bC220] to Free electronic thermal frequency-3, 2nd-motor [bC224](Hz)
bC224	Free electronic thermal frequency-3, 2nd-motor	Free electronic thermal frequency-2, 2nd-motor [bC222] to 590.00(Hz)
bA-31	Decel-stop at power failure starting voltage	200Vclass: 0 to 410.0(V) 400Vclass: 0 to 820.0(V) Decel-stop at power failure starting voltage [bA-31] ≤ Decel-stop at power failure control target level [bA-32]
bA-32	Decel-stop at power failure control target level	200Vclass: 0 to 410.0(V) 400Vclass: 0 to 820.0(V) Decel-stop at power failure starting voltage [bA-31] ≤ Decel-stop at power failure control target level [bA-32]
Cb-05	Start rate of Terminal [Ai1]	0.0 to End rate of Terminal [Ai1] [Cb-06](%)
Cb-06	End rate of Terminal [Ai1]	Start rate of Terminal [Ai1] [Cb-05] to 100.0(%)
Cb-15	Start rate of Terminal [Ai2]	0.0 to End rate of Terminal [Ai2] [Cb-16](%)
Cb-16	End rate of Terminal [Ai2]	Start rate of Terminal [Ai2] [Cb-15] to 100.0(%)
Cb-25	Start rate of Terminal [Ai3]	-100.0 to End rate of Terminal [Ai3] [Cb-26](%)
Cb-26	End rate of Terminal [Ai3]	Start rate of Terminal [Ai3] [Cb-25] to 100.0(%)
CE-40	Window comparator for [Ai1] higher level	Set an upper limit level. Setting range: 0. to 100. Lower limit: Window comparator for [Ai1] lower level [CE-41] + Window comparator for [Ai1] hysteresis width [CE-42] × 2
CE-41	Window comparator for [Ai1] lower level	Set a lower limit level. Setting range: 0. to 100. Upper limit: Window comparator for [Ai1] higher level [CE-40] - Window comparator for [Ai1] hysteresis width [CE-42] × 2
CE-42	Window comparator for [Ai1] hysteresis width	Set a hysteresis width for the upper and lower limit levels. Setting range: 0. to 10. Upper limit: (Window comparator for [Ai1] higher level [CE-40] - Window comparator for [Ai1] lower level [CE-41])/2

Parameter No.	Description	Data
CE-43	Window comparator for [Ai2] higher level	Set an upper limit level. Setting range: 0. to 100. Lower limit: Window comparator for [Ai2] lower level [CE-44] + Window comparator for [Ai2] hysteresis width [CE-45] × 2
CE-44	Window comparator for [Ai2] lower level	Set a lower limit level. Setting range: 0. to 100. Upper limit: Window comparator for [Ai2] higher level [CE-43] - Window comparator for [Ai2] hysteresis width [CE-45] × 2
CE-45	Window comparator for [Ai2] hysteresis width	Set a hysteresis width for the upper and lower limit levels. Setting range: 0. to 10. Upper limit: (Window comparator for [Ai2] higher level [CE-43] - Window comparator for [Ai2] lower level [CE-44])/2
CE-46	Window comparator for [Ai3] higher level	Set an upper limit level. Setting range: 0. to 100. Lower limit: Window comparator for [Ai3] lower level [CE-48] + Window comparator for [Ai3] hysteresis width [CE-47] × 2
CE-47	Window comparator for [Ai3] lower level	Set a lower limit level. Setting range: 0. to 100. Upper limit: Window comparator for [Ai3] higher level [CE-46] - Window comparator for [Ai3] hysteresis width [CE-48] × 2
CE-48	Window comparator for [Ai3] hysteresis width	Set a hysteresis width for the upper and lower limit levels. Setting range: 0. to 10. Upper limit: (Window comparator for [Ai3] higher level [CE-46] - Window comparator for [Ai3] lower level [CE-47])/2
Hb150	Free-V/f frequency 1 setting, 1st-motor	0.00 to Free-V/f frequency 2 setting, 1st-motor [Hb152](Hz)
Hb152	Free-V/f frequency 2 setting, 1st-motor	Free-V/f frequency 1 setting, 1st-motor [Hb150] to Free-V/f frequency 3 setting, 1st-motor [Hb154](Hz)
Hb154	Free-V/f frequency 3 setting, 1st-motor	Free-V/f frequency 2 setting, 1st-motor to Free-V/f frequency 4 setting, 1st-motor [Hb156](Hz)
Hb156	Free-V/f frequency 4 setting, 1st-motor	Free-V/f frequency 3 setting, 1st-motor [Hb154] to Free-V/f frequency 5 setting, 1st-motor [Hb158](Hz)
Hb158	Free-V/f frequency 5 setting, 1st-motor	Free-V/f frequency 4 setting, 1st-motor [Hb156] to Free-V/f frequency 6 setting, 1st-motor [Hb160](Hz)
Hb160	Free-V/f frequency 6 setting, 1st-motor	Free-V/f frequency 5 setting, 1st-motor to Free-V/f frequency 7 setting, 1st-motor [Hb162](Hz)
Hb162	Free-V/f frequency 7 setting, 1st-motor	Free-V/f frequency 6 setting, 1st-motor to Async.Motor Base frequency setting, 1st-motor [Hb104](Hz)
Hb250	Free-V/f frequency 1 setting, 2nd-motor	0.00 to Free-V/f frequency 2 setting, 2nd-motor [Hb252](Hz)
Hb252	Free-V/f frequency 2 setting, 2nd-motor	Free-V/f frequency 1 setting, 2nd-motor to Free-V/f frequency 3 setting, 2nd-motor [Hb254](Hz)

Parameter No.	Description	Data
Hb254	Free-V/f frequency 3 setting, 2nd-motor	Free-V/f frequency 2 setting, 2nd-motor to Free-V/f frequency 4 setting, 2nd-motor [Hb256](Hz)
Hb256	Free-V/f frequency 4 setting, 2nd-motor	Free-V/f frequency 3 setting, 2nd-motor [Hb254] to Free-V/f frequency 5 setting, 2nd-motor [Hb258](Hz)
Hb258	Free-V/f frequency 5 setting, 2nd-motor	Free-V/f frequency 4 setting, 2nd-motor [Hb256] to Free-V/f frequency 6 setting, 2nd-motor [Hb260](Hz)
Hb260	Free-V/f frequency 6 setting, 2nd-motor	Free-V/f frequency 5 setting, 2nd-motor [Hb258] to Free-V/f frequency 7 setting, 2nd-motor [Hb262](Hz)
Hb262	Free-V/f frequency 7 setting, 2nd-motor	Free-V/f frequency 6 setting, 2nd-motor [Hb260] to Async.Motor Base frequency setting, 2nd-motor [Hb204](Hz)

7-2 Parameters Affected by Rated Current [%]

In the case of parameters for which a current value is set, the parameter's default data and setting range are restricted by the rated current of the inverter.

When you set those parameters, set a percentage [0.01% unit] of the inverter rated current in the DriveProgramming program.

The following are representative examples of parameters whose rated current is included in the data.

For details, refer to the *9-5 Modbus Communication Register Number List* in the High-function General-purpose Inverter 3G3RX2 Series User's Manual (I620).

Parameter No.	Description	Data
AF136	Brake Release Current Setting, 1st-motor (Forward side)	0.00 to 200.00(%) (Inverter rated current ratio)
AF143	Brake Release Current Setting, 1st-motor (Reverse side)	0.00 to 200.00(%) (Inverter rated current ratio)
bA121	Over current suppress Level, 1st-motor	0.00 to 200.00(%) (Inverter rated current ratio)
bA123	Overload restriction 1 active level, 1st-motor	20.00 to 200.00(%) (Inverter rated current ratio)
bA127	Overload restriction 2 active level, 1st-motor	20.00 to 200.00(%) (Inverter rated current ratio)
bb-43	Restart level of Active frequency matching	0.00 to 200.00(%) (Inverter rated current ratio)
bb-46	OC-supress level of Active frequency matching	0.00 to 200.00(%) (Inverter rated current ratio)
bb160	Over current detection level, 1st-motor	20.00 to 220.00(%) (Inverter ND rated current ratio)
bC110	Electronic thermal level setting, 1st-motor	0.00 to 300.00(%) (Inverter rated current ratio)
bC121	Free electronic thermal current-1, 1st-motor	0.00 to 300.00(%) (Inverter rated current ratio)
bC123	Free electronic thermal current-2, 1st-motor	0.00 to 300.00(%) (Inverter rated current ratio)
bC125	Free electronic thermal current-3, 1st-motor	0.00 to 300.00(%) (Inverter rated current ratio)
CE102	Low current detection level 1, 1st motor	0.00 to 200.00(%) (Inverter rated current ratio)
CE103	Low current detection level 2, 1st motor	0.00 to 200.00(%) (Inverter rated current ratio)
CE106	Over current detection level 1, 1st motor	0.00 to 200.00(%) (Inverter rated current ratio)
CE107	Over current detection level 2, 1st motor	0.00 to 200.00(%) (Inverter rated current ratio)
PA-23	Output current monitor optional output value setting	0.00 to 300.00(%) (Inverter rated current ratio)
AF236	Brake Release Current Setting, 2nd-motor (Forward side)	0.00 to 200.00(%) (Inverter rated current ratio)
AF243	Brake Release Current Setting, 2nd-motor (Reverse side)	0.00 to 200.00(%) (Inverter rated current ratio)
bA221	Over current suppress Level, 2nd-motor	0.00 to 200.00(%) (Inverter rated current ratio)
bA223	Overload restriction 1 active level, 2nd-motor	0.00 to 200.00(%) (Inverter rated current ratio)
bA227	Overload restriction 2 active level, 2nd-motor	20.00 to 200.00(%) (Inverter rated current ratio)
bb260	Over current detection level, 2nd-motor	20.00 to 220.00(%) (Inverter ND rated current ratio)
bC210	Electronic thermal level setting, 2nd-motor	0.00 to 300.00(%) (Inverter rated current ratio)
bC221	Free electronic thermal current-1, 2nd-motor	0.00 to 300.00(%) (Inverter rated current ratio)

Parameter No.	Description	Data
bC223	Free electronic thermal current-2, 2nd-motor	0.00 to 300.00(%) (Inverter rated current ratio)
bC225	Free electronic thermal current-3, 2nd-motor	0.00 to 300.00(%) (Inverter rated current ratio)
CE202	Low current detection level 1, 2nd-motor	0.00 to 200.00(%) (Inverter rated current ratio)
CE203	Low current detection level 2, 2nd-motor	0.00 to 200.00(%) (Inverter rated current ratio)
CE206	Over current detection level 1, 2nd-motor	0.00 to 200.00(%) (Inverter rated current ratio)
CE207	Over current detection level 2, 2nd-motor	0.00 to 200.00(%) (Inverter rated current ratio)

7-3 Parameters Affected by PID Enabled/Disabled

The set values are scaled by the settings for AH-04/AH-05.

Parameter No.	Description	Data
		A071 = 01 or 02
db-30	PID1 feedback data 1 monitor	a) Conversion (Scale adjustment) = (PID1 scale adjustment (at 100%) [AH-05] - PID1 scale adjustment (at 0%) [AH-04]) / 100.00 * [Input (%)] + PID1 scale adjustment (at 0%) [AH-04] b) Internal conversion (%) = ([Input (*1)] - PID1 scale adjustment (at 0%) [AH-04]) / (PID1 scale adjustment (at 100%) [AH-05] - PID1 scale adjustment (at 0%) [AH-04]) * 100.00 *1. Equivalent to mon param read value (Same scale as ones of AH-04, 05)
db-32	PID1 feedback data 2 monitor	
db-34	PID1 feedback data 3 monitor	
db-42	PID1 target value monitor	
db-44	PID1 feedback data 1 monitor	
FA-30	PID1 Set Value 1	
FA-32	PID1 Set Value 2	
FA-34	PID1 Set Value 3	
AH-10	Set-point-1 setting for PID1	
AH-12	PID1 Multi stage set-point 1 setting	
AH-14	PID1 Multi stage set-point 2 setting	
AH-16	PID1 Multi stage set-point 3 setting	
AH-18	PID1 Multi stage set-point 4 setting	
AH-20	PID1 Multi stage set-point 5 setting	
AH-22	PID1 Multi stage set-point 6 setting	
AH-24	PID1 Multi stage set-point 7 setting	
AH-26	PID1 Multi stage set-point 8 setting	
AH-28	PID1 Multi stage set-point 9 setting	
AH-30	PID1 Multi stage set-point 10 setting	
AH-32	PID1 Multi stage set-point 11 setting	
AH-34	PID1 Multi stage set-point 12 setting	
AH-36	PID1 Multi stage set-point 13 setting	
AH-38	PID1 Multi stage set-point 14 setting	
AH-40	PID1 Multi stage set-point 15 setting	
AH-44	Set-point 2 setting for PID1	
AH-48	Set-point 3 setting for PID1	

The set values are scaled by the settings for AJ-04/AJ-05.

Parameter No.	Description	Data
		A071 = 01 or 02
db-36	PID2 feedback data monitor	a) Conversion (Scale adjustment) = (PID1 scale adjustment (at 100%) [AJ-05] - PID1 scale adjustment (at 0%) [AJ-04]) / 100.00 * [Input (%)] + PID1 scale adjustment (at 0%) [AJ-04] b) Internal conversion (%) = ([Input (*1)] - PID1 scale adjustment (at 0%) [AJ-04]) / (PID1 scale adjustment (at 100%) [AJ-05] - PID1 scale adjustment (at 0%) [AJ-04]) * 100.00 *1. Equivalent to mon param read value (Same scale as ones of AJ-04, 05)
FA-36	PID2 Set Value	
AJ-10	Set-point setting for PID2	
db-55	PID2 output monitor	
db-56	PID2 deviation monitor	

The set values are scaled by the settings for AJ-24/AJ-25.

Parameter No.	Description	Data
db-38	PID3 feedback data monitor	a) Conversion (Scale adjustment) = (PID1 scale adjustment (at 100%) [AJ-25] - PID1 scale adjustment (at 0%) [AJ-24]) / 100.00 * [Input (%)] + PID1 scale adjustment (at 0%) [AJ-24] b) Internal conversion (%) = ([Input (*1)] - PID1 scale adjustment (at 0%) [AJ-24]) / (PID1 scale adjustment (at 100%) [AJ-25] - PID1 scale adjustment (at 0%) [AJ-24]) * 100.00 *1. Equivalent to mon param read value (Same scale as ones of AJ-24, 25)
FA-38	PID3 Set Value	
AJ-30	Set-point setting for PID3	
db-57	PID3 output monitor	
db-58	PID3 deviation monitor	

The set values are scaled by the settings for AJ-44/AJ-45.

Parameter No.	Description	Data
db-40	PID4 feedback data monitor	a) Conversion (Scale adjustment) = (PID1 scale adjustment (at 100%) [AJ-45] - PID1 scale adjustment (at 0%) [AJ-44]) / 100.00 * [Input (%)] + PID1 scale adjustment (at 0%) [AJ-44] b) Internal conversion (%) = ([Input (*1)] - PID1 scale adjustment (at 0%) [AJ-44]) / (PID1 scale adjustment (at 100%) [AJ-45] - PID1 scale adjustment (at 0%) [AJ-44]) * 100.00 *1. Equivalent to mon param read value (Same scale as ones of AJ-44, 45)
FA-40	PID4 Set Value	
AJ-50	Set-point setting for PID4	
db-59	PID4 output monitor	
db-60	PID4 deviation monitor	

8

Errors and Remedies

This section describes the program operation at the time of error occurrence, the errors that are specific to the DriveProgramming, as well as the causes and remedies.

8-1 Troubleshooting

This section describes the program operation at the time of error occurrence, the error codes that are specific to the DriveProgramming, and the remedies for them.

8-1-1 DriveProgramming Operation on Error

Basically, even if the inverter detects a trip during the DriveProgramming operation, the operation is continued. However, if any of E043 to E045 trips related to the DriveProgramming is detected, the operation is stopped. Or, with the “on trip goto” command, the program can jump to other process after a trip occurred.

With/without “on trip goto”	Error status		
	User trip E050 to E059	DriveProgramming related trip E043 to E045	Other trips
Without	Operation is continued.	Program is stopped.	Operation is continued.
With	After the “on trip goto” command is executed, the program jumps to the specified label and the operation is continued.	Program is stopped.	After the “on trip goto” command is executed, the program jumps to the specified label and the operation is continued.

Precautions for Safe Use

When the DriveProgramming program is stopped, the status before the program stop is retained for output terminals controlled by the DriveProgramming.

For this reason, configure the system so that the stop of the DriveProgramming program in the inverter can be detected by the DriveProgramming start signal and the alarm (trip) signal, and the inverter's peripheral devices can be stopped safely.

8-1-2 DriveProgramming Operation on Error Reset

The DriveProgramming operation on error reset varies with the reset input method and the setting for the Reset Selection (CA-72). The following table shows the operation for each case.

To restart the DriveProgramming, set the Input Terminal Selection (CA-01 to CA-11) to 28 (RS (Reset)) and turn ON the corresponding input terminal.

Operation	Program status	Trip status	Reset with control terminal	
			C102 = 00 or 01	C102 = 02 or 03
DriveProgramming operation	During run	Normal status	Restart	Keep run
		During trip	Restart	Restart
	During stop	Normal status	Restart	Keep stop*1
		During trip	Restart	Restart
Inverter's reset operation		Normal status	Reset	Disabled
		During trip	Reset	Reset

*1. To restart the program, set the DriveProgramming Function Selection to 00 (Disabled) first, and then change it to 02 (Always). Also, you can restart it by turning ON the PRG terminal again, if UE-02 = 01.

8-1-3 Alarm Code List

This section describes the alarm codes that are specific to the DriveProgramming, as well as the causes and remedies.

For other errors, refer to the High-function General-purpose Inverter 3G3RX2 Series User's Manual (I620).

Alarm code	Alarm (cause of inverter trip)	Possible cause	Check	Remedy
E043	DriveProgramming Invalid Command	The terminal PRG was turned ON although no program was downloaded into the inverter.	Upload the program and check if it actually exists in the inverter.	Create the program again and download it to the inverter.
E044	DriveProgramming Nesting Count Error	A subroutine is nested over eight levels.	Upload the program and check the number of nesting levels.	Correct the program so that the number of nesting levels is eight or less.
		The “for next” command is nested over eight levels.		
		The “if” command is nested over eight levels.		
E045	DriveProgramming Command Error	The jump destination of the “goto” command points to the “next” command that terminates the “for” or other loop.	Check if the jump destination of the “goto” command is an command that terminates a loop.	Correct the jump destination of the “goto” command.
		A four arithmetic operation results in the following error: Overflow, underflow, or division by zero	Check the program and identify the command which causes overflow, underflow, or division by zero.	Correct the program so that its four arithmetic operation does not cause overflow, underflow, or division by zero.
		The “ChgParam” and “MonParam” command results in the following error: • Change to a parameter that does not exist • The written value is out of the setting range • Change of a parameter value that cannot be changed during inverter operation • Change of a parameter value that is protected against change by the Soft Lock Selection (when this setting is enabled)	• Check the parameter or the value to write to the parameter. • In the case that an error occurs during inverter operation, check if the parameter can be changed during inverter operation. • Check the setting of the Soft Lock Selection (UA-16).	• Correct the parameter or the value to write to the parameter so that it is within the setting range. • Disable the soft lock function. • If the parameter can be changed during operation, change the setting of the Soft Lock Selection (UA-16) to 10 to enable a parameter change during inverter operation.

OMRON Corporation **Industrial Automation Company**
Kyoto, JAPAN
Contact: www.ia.omron.com

Regional Headquarters
OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31)2356-81-300/Fax: (31)2356-81-388

OMRON ELECTRONICS LLC
2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A.
Tel: (1) 847-843-7900/Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

Cat. No. I622-E1-01 0119