Cat. No. I170E-EN-01B

MX2/RX/LX Drive Programming

Model:
MX2 series inverter
RX series inverter
LX series inverter

USER'S MANUAL

Table of Contents

1-Introduction

This Instruction Manual explains how to use the *Drive Programming* software for the Omron MX2/RX/LX Series Inverter. Be sure to read this Instruction Manual carefully before using *Drive Programming*, and keep it on hand for future reference.

1-1 Handling of this Instruction Manual

- The contents of this Instruction Manual are subject to change without prior notice.
- No part of this Instruction Manual may be reproduced in any form without the publisher's permission.
- If you find any incorrect description, missing description or have a question concerning the contents of this Instruction Manual, please contact the publisher.

1-2 Safety Instruction

Be sure to read this Instruction Manual, Inverter Instruction Manual, and appended documents thoroughly before using *Drive Programming* and the inverter.

Before creating user programs for the inverter, also refer to the Inverter Instruction Manual and configuration software (CX-Drive) Instruction Manual for the necessary related Knowledge, and ensure you understand and follow all safety information, precautions, and operating and handling instructions for the correct use of the inverter.

Always use the inverter strictly within the range of specifications described in the Inverter Instruction Manual and correctly implement maintenance and inspection to prevent fault from occurring.

When using the inverter together with optional products, also read the manuals for those products. Note that this Instruction Manual and the manual for each optional product to be used should be delivered to the end user of the inverter.

In this instruction manual you can find WARNINGS along the instructions

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

1-3 Preparation and System configuration

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

(1) MX2, RX, LX inverter
(2) Personal computer (PC) (Windows System)
-32-bit PC: Windows XP SP3, Windows Vista (any service pack) and Windows 7.
-64-bit PC: Windows Vista (any service pack) and Windows 7.

(3) Optional programming software CX-Drive
· MX2 inverter: CX-Drive version 2.0x or higher.
· RX inverter: CX-Drive 2.3x or higher.
· LX inverter: CX-Drive 2.5x or higher.

(4) Optional PC-inverter connection cable. For MX2 it is a USB cable, For RX/LX, the converter cable USB to RJ-45 is required. Item codes:
- Item code name for MX2: **AX-CUSBM002-E**
- Item code name for RX/LX (2 option cables):
· **3G3AX-PCACN2**, or
· **USB CONVERTERCABLE**

LX:
· Inverter port: Operator-connection port RJ-45.

RX:
· Inverter port: Operator-connection port RJ-45.

MX2:
· Inverter port: USB connector.

The following figure shows the basic system configuration for programming.

Optional programming software CX-Drive	Windows personal computer	Optional PC-Inverter cable	MX2, RX or LX Inverter
 MX2: CX-Drive 2.0x or higher RX: CX-Drive 2.3x or higher LX: CX-Drive 2.5x or higher		- For MX2: · **AX-CUSBM002-E** - For RX/LX (2 options): · **3G3AX-PCACN2**, or · **USB-CONVERTERCABLE**	

Install CX-Drive on your Windows personal computer, and connect the personal computer to the inverter (MX2, RX or LX) via the PC-inverter connection cable.

After completing these preparations, you can operate Drive Programming Editor to create a user program and download it to the inverter.

The table below lists the main functions of Drive Programming Editor.

Function	Description
Programming Editor	Supports the input, editing, saving, reading, and printing of user programs
Compilation	Compile and edit a user program
Downloading and uploading	Downloads a user program to the inverter Uploads a user program from the inverter

2- Specifications

The table below lists the programming-related specifications of the Drive Programming function.

Item		Specification		
Language specification	Programming language	Flow Chart and Text language		
	Input device	Windows personal computer (OS: Windows XP-SP3**,** Windows Vista, Windows 7)		
	Max. program size	1024steps (The internal storage capacity of the inverter is 1024 steps or 6 Kilobytes.)		
	Programming support function (programming software)	-Editing (on Windows) / - Display (on Windows) -Program syntax check (on Windows) -Downloading, uploading, and full clearance of program		
	Execution format	Execution by interpreter in an execution cycle of 2ms per instruction (possible subroutine call with nesting in up to 8 layers)		
Input/output-related functions	External input	Contact Signal		24v open – collector input (using intelligent input terminals)
			Program run signal input	RX: Assign to the PRG terminal / Always run
				MX2: Assign to the PRG terminal / Always run
			Multifunction terminals	RX: Up to 8 terminals (X(00) to X(07))
				MX2: Up to 8 terminals (X(00) to X(07))
		General-purpose analog input		XA(0): 0 to 10V (O terminal)
				XA(1): 4 to 20mA (OI terminal)
				XA(2): 0 to 10V (O2 terminal) (Only RX)
	External Output	General-purpose output terminal		RX: Up to 6 terminals (Y(00) to Y(05))
				MX2: Up to 3 terminals (Y(00) to Y(02))
		General-purpose analog output		YA (0): Assignable to the **EO** terminal (FM terminal for RX)
				YA (1): Assignable to the AM terminal
				YA (2): Assignable to the AMI terminal (Only RX)
Reserved Words	Instructions	(1) Program control instructions -Loop (For) / - Unconditional branching (Goto) / -Time control (Wait) -Conditional branching (If Then, Ifs Then Else, Select Case, Until, and While) -Subroutine (call, sub) / - Others (Entry, End, Sub, End Sub, Inc, and Dec) (2) Arithmetic instructions -Arithmetic operation (+,-,*,/) / - Remainder (Mod) / -Substitution (=) -Absolute value (Abs) / - Logic operation (Or, And, Xor, and Not) (3) Input/Output control -General-purpose input/output (bit input, word input, bit output, and word output) - Reading of inverter input terminal. (4) Timer control: - Delay operation / -Timer control (5) Parameter control: - Rewriting of parameters by reselecting code on the operator's display		
	Number of variables	User-defined variable		U (00) to U (31) (32 variables)
		Internal user variable		UL (00) to UL (07) (8 variables)
		Set frequency		SET-Freq
		Acceleration time		ACCEL
		Deceleration time		DECEL
		Monitoring variable		FM, Iout, Dir, PID-FB, F-CNV, Tmon, Vout, Power, RUN-Time, ON-Time, PlsCnt (Only RX), POS, STATUS, DCV, ERR CNT, ERR(1), ERR(2), ERR(3), ERR(4), ERR(5), and ERR(6)
		Bit commands		FW, RV, CF1, CF2,CF3,CF4,JG,DB,SET,TCH,FRS,EXT,USP,CS,SFT, AT, RS, STA, STP, F/R, PID, PIDC, UP, DWN, UDC, OPE, SF1, SF2, SF3, SF4, SF5, SF6, SF7, OLR, TL, TRQ1-2, BOK, LAC, PCLR, ADD, F-TM, ATR, KHC, AHD, CP1-3, ORL, ORG, SPD, RS485, HLD, ROK, DISP.
		Output Functions		RUN, FA1, FA2, OL, OD, AL, FA3, OTQ, UV, TRQ, RNT, ONT, THM, BRK, BER, ZS, DSE, POK, FA4, FA5, OL2, ODc, OIDc, FBV, NDc, LOG1, LOG2, LOG3, WAC, WAF, FR, OHF, LOC, IRDY, FWR, RVR, MJA, WCO, WCOI, FREF, REF, SETM, EDM.
		General-purpose input contact		LX : X(00) to X(06) (7 contacts)
				RX : X (00) to X (07) (8 contacts)
				MX2 : X(00) to X(07) (8 contacts)
		Extended IO option input contact		LX : X(07) to X(11) (3G3AX-EIO-E expansion card)
				RX : -----
				MX2 : -----
		General-purpose output contact		LX : Y(00) to Y(03) (4 contacts)
				RX :Y(00) to Y(05) (6 contacts)
				MX2 : Y(00) to Y(02) (3 contacts)
		Extended output contact		LX: Y(04) to Y(06) (3G3AX-EIO-E expansion card)
				MX2 & RX : -----
		Internal user contact		UB (00) to UB (7) (8 contacts)
		Internal timer contact		TD (0) to TD (7) (8 counter contacts)
		Internal timer counter		TC (0) to TC (7) (8 counters)
		Inverter input/output		Specification by code on the remote operator's display
		User Monitor		UMon(00) to UMon(02) (3 user monitors)
		User trip		Makes the inverter trip (10 trips)

3- Drive Programming Editor

Drive Programming Editor allows the user to design drive programs in an intuitive way. CX-Drive provides a way to create drive programs, compile them, transfer them to and from the drive, start and stop their execution, and other related tasks.
You can open this function by clicking on Drive Programming in the workspace of a drive which supports it, or selecting *Program* | *Program Editor* from the Drive menu, or with the CX-Drive toolbar button .

Please create a new CX-Drive File by clicking on the menu *File* | *New*. The New Drive window will appear (Image 1). Select the Drive Type and press OK button. Then it will appear on the Workspace (Image 2).

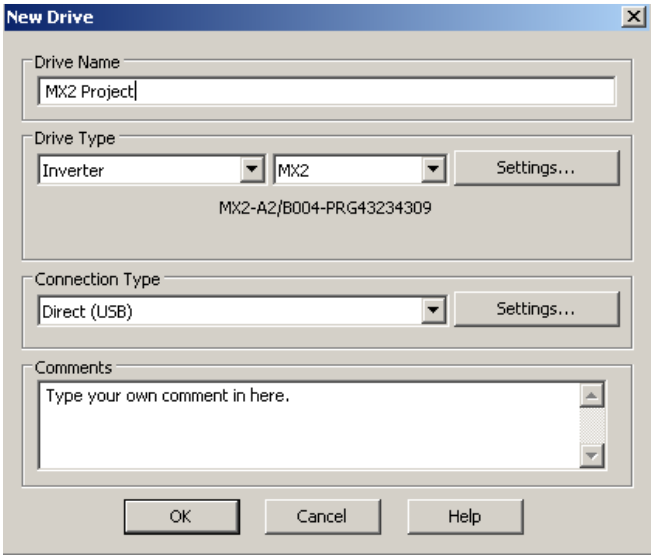


Image 1- New Drive window

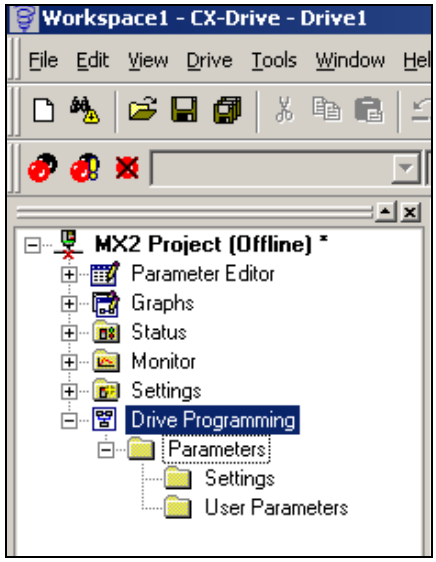
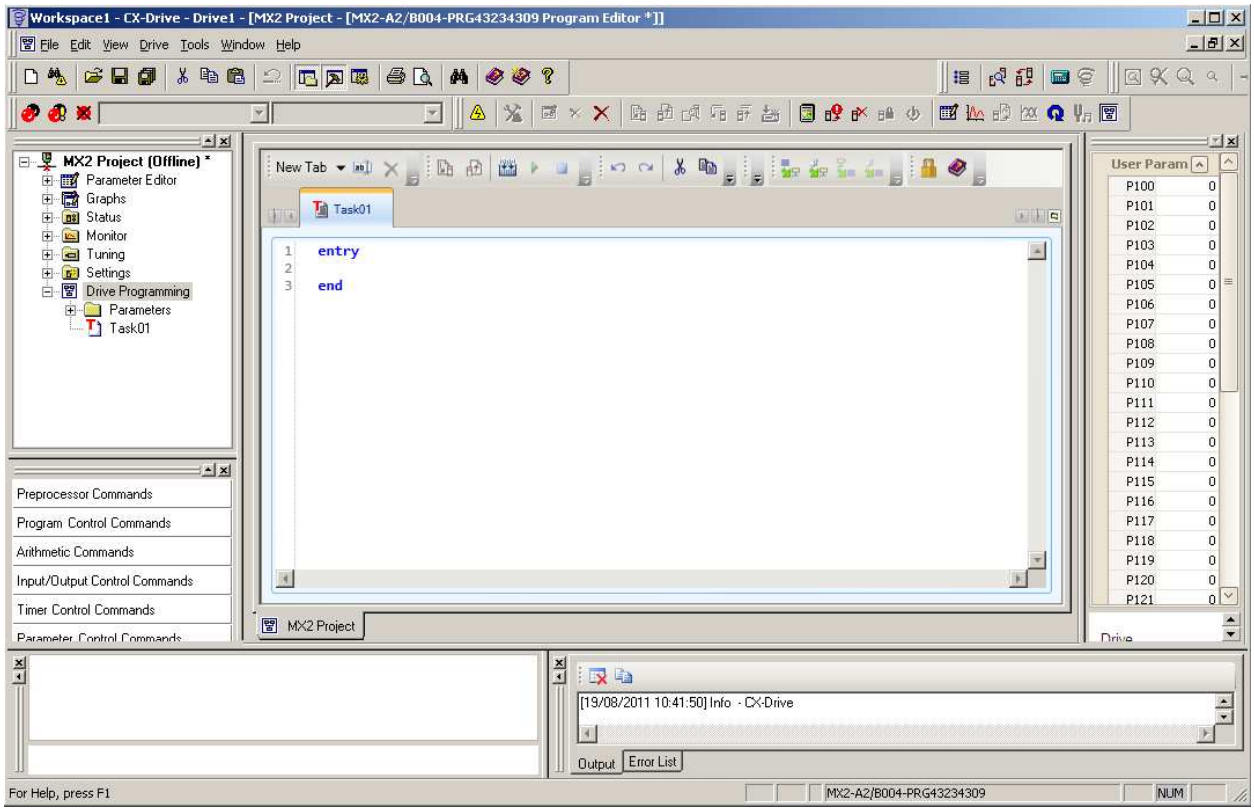


Image 2- CX-Drive Workspace

Making double-click to the Drive Programming option, the Drive Programming Editor will appear.



3-1 Saving and loading programs

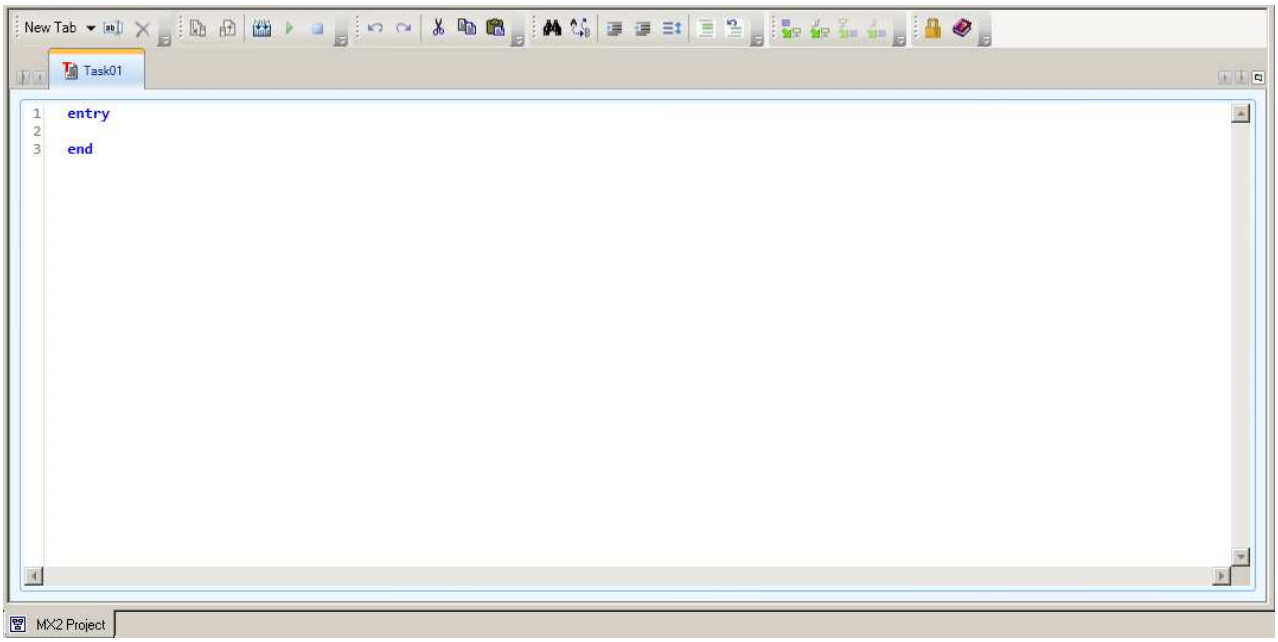
A drive program is automatically saved when the drive document which contains it is saved.
When a CX-Drive document is opened, the drive program which it contains, if any, is automatically loaded. You can display it by opening the Program Editor.

Alternatively, you can export a drive program, to save it independently of other drive information. To do so, use the *Program* | *Export Program* command in the Drive menu. Enter the name of the file to be used. The file will be saved with extension *driveprogram*.

A drive program can be imported with the *Program* | *Import Program* command in the Drive menu.

3-2 Editor

The Program Editor is the main window of the Drive Programming function.



The window area consists of a toolbar with common commands, and a designer area where the program is displayed as a text.

3-3 Toolbar

The Program Editor window contains the following commands:

3-3-1 Common Commands

Commands	Image	Description
New task (Flowchart)		It allows creating a new Flowchart task for the program, up to the maximum number of tasks allowed. Tasks are parts of the program which are executed independently of each other.
New Task (Text)		It allows creating a new Text task for the program, up to the maximum number of tasks allowed. Tasks are parts of the program which are executed independently of each other.
New Subroutine (Flowchart)		It allows creating a new Flowchart subroutine. A subroutine is a part of the program which is called from a task.
New Subroutine (Text)		It allows creating a new Text subroutine. A subroutine is a part of the program which is called from a task.
Rename Current Task/Subroutine		It allows to rename the current task/subroutine.
Delete Current Tab		It deletes the current Task or Subroutine.
Undo		It reverts the latest change.
Redo		It recovers the most recently undone change.
Transfer to Drive		It compiles the program and, if there are no errors, transfer it to the drive.
Transfer from Drive		It transfers the program from the drive to the Program Editor.
Start		It starts the program in the drive. CX-Drive will first compare it with the program currently being edited, to make sure that they are the same. If they differ, the program will not be started.
Stop		It stops the program in the drive. This action is done regardless of whether the program in the drive is the same that in the program designer.
Compile		It compiles the program which is currently being designed. Compile errors and warnings will be reported as tool tips in the blocks in the flowchart.
Password		It allows you to set, change or remove the program password.
Help		It displays the Drive Programming help.

3-3-2 Commands for the Flowchart Editor

Commands	Image	Description
Zoom in		It increases the zoom level.
Zoom out		It decreases the zoom level.
Zoom Reset		It restores the zoom to its initial value.
Select Mode		It allows the user to select one or more elements of the program, by click-and-drag with the mouse cursor. This mode is active by default.
Pan Mode		It allows the user to move the extent of the view. in any direction while keeping the same scale, by click-and-drag.
Horizontal Align Left		It aligns horizontally the left sides of the selected blocks.
Horizontal Align Middle		It aligns horizontally the middles of the selected blocks.
Horizontal Align Right		It aligns horizontally the right sides of the currently selected blocks.
Vertical Align Top		It aligns vertically the top sides of the selected blocks.
Vertical Align Middle		It aligns vertically the middles of the selected blocks.
Vertical Align Bottom		It aligns the bottom sides of the selected blocks.
Orientation		It selects a preferred orientation for connecting the blocks.
Auto-arrange		It arranges the elements of the flowchart automatically in the currently selected orientation.
Show contacts		It toggles display/hide of the contacts of the blocks, which are placeholders for the beginning and ending of arrow connections
Show		It allows you to select a display style of the program. (Text only, Icon Only, Icon and Text, or Name, Icon and Arguments).

3-3-3 Commands for the Text Editor

Commands	Image	Description
Find		It finds a text on the program code.
Replace		It replaces a text on the program code.
Increment Indentation		It increases the indentation of the selected text.
Decrement Indentation		It decreases the indentation of the selected text.
Format Selected Text		It applies the automatic formatting to the selected text.
Comment Selected Text		It transforms the selected rows of text to comments.
Uncomment Selected Text		It uncomments the selected rows of text.
Convert Text to Flowchart		It converts current text Task/Subroutine to Flowchart
Convert whole program to Flowchart		It converts whole program to Flowchart.
Convert Flowchart to Text		It converts current Flowchart Task/Subroutine to text.
Convert whole Program to Text		It converts whole program to Text.

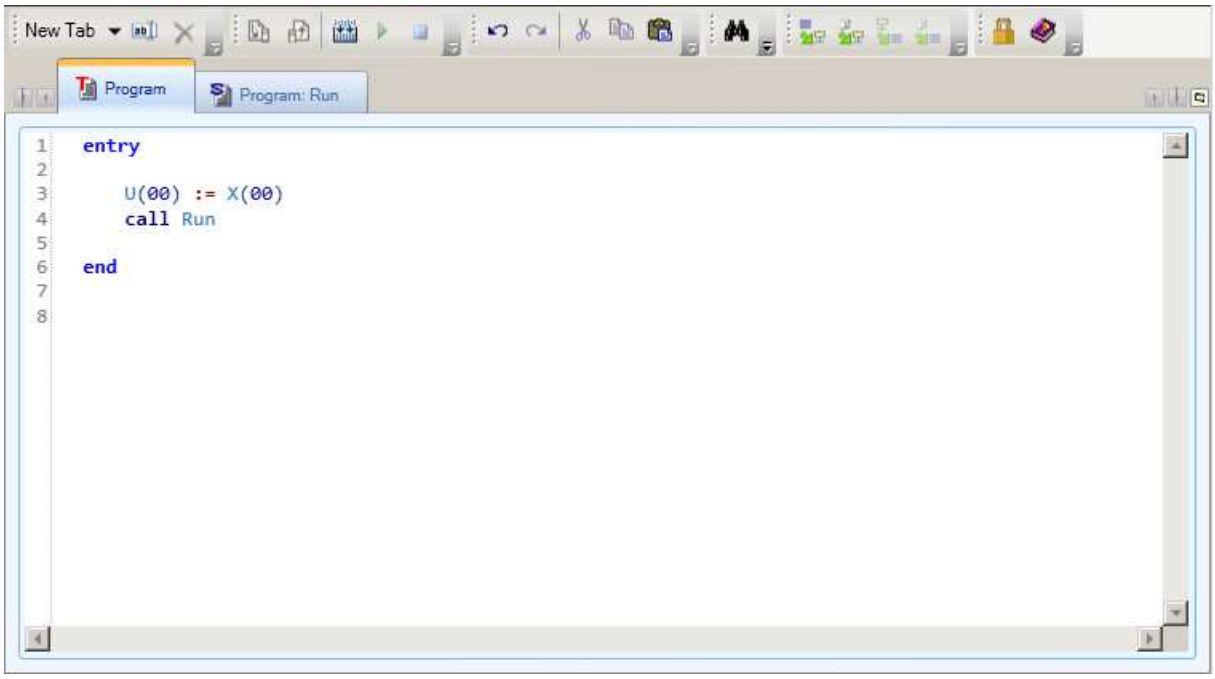
3-4 Shortcut Keys

The following Keyboard shortcuts can be applied to the designer area.

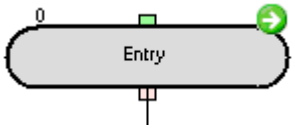
· Ctrl + X: Cut
· Ctrl + C: Copy
· Ctrl + V: Paste
· Ctrl + Z: Undo
· Ctrl + Y: Redo
· Ctrl + A: Select All
· Ctrl + F: Find function
· Ctrl + L: Lock
· Ctrl + P: Pin
· Ctrl + Space: Code Snippets
· Tab: Select Next
· Shift + Tab: Select Previous
· Arrow Keys: Move selected element
· Home, End, Page Up, Page Down: Navigate through the graph
· +: Zoom In
· -: Zoom Out

3-5 Designer Area

The designer area will display 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.
The designer is created with one default tab, which is a Text Task.
When a program is compiled without error, an icon with a circled green arrow highlights the starting point of each Flow chart task.

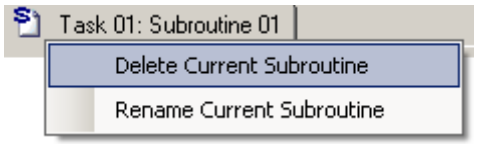


With text editor, the output window will indicate if the program is compiled successfully.

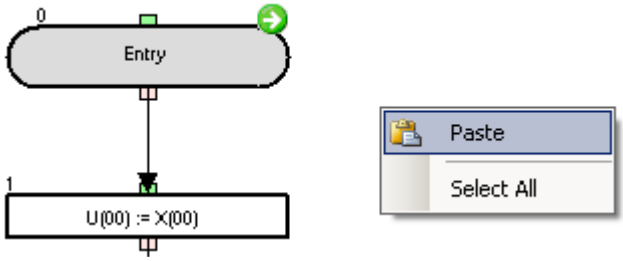
For programs compiled with errors, a red icon with an exclamation mark identifies the erroneous blocks with Flowchart Editor. Placing the mouse on the error icon displays the compile error, which can also be seen in the Error List.

With Text Editor, in the output window will appear the errors of the program. The error will be showed with a red underline.

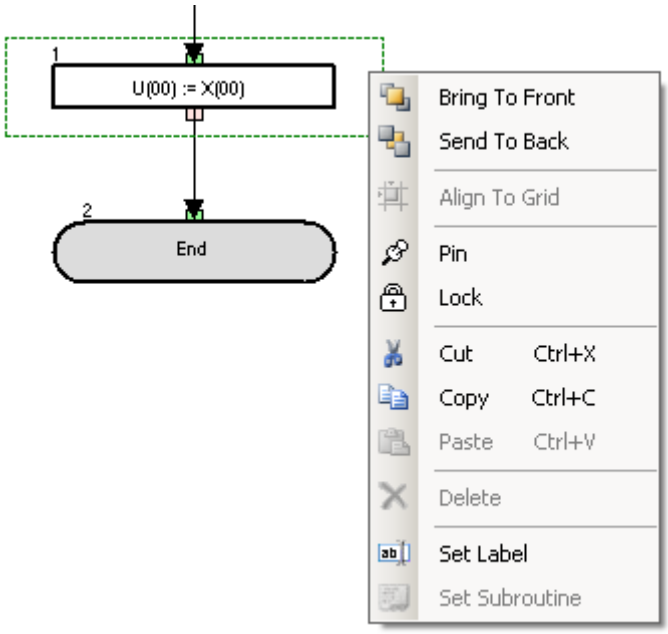
A Task or Subroutine may be deleted, or renamed, by right-clicking on the tab title.



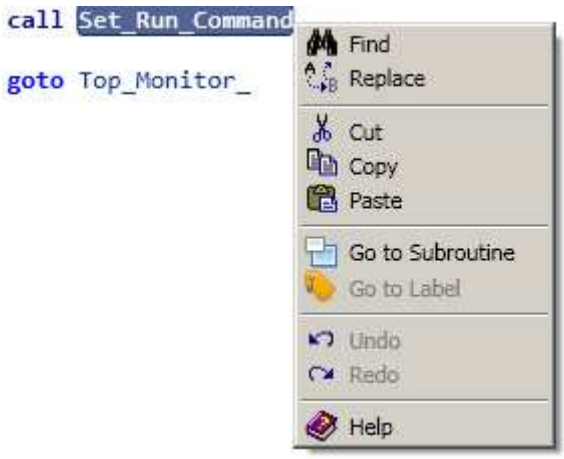
Right-clicking on an area which is not an element of the flowchart displays a popup menu which allows you to Paste elements that you have previously copied, or to select all the elements.



Right-clicking on a Flowchart block element it shows a popup menu with more options



Right-clicking on a selected Text it shows a popup menu with more options



The available menu commands with Flowchart editor are described below.

• *Bring To Front* places the element graphically in front of other elements.
• *Send To Back* places the element graphically in back of other elements.
• Pin fixes the element to its current position in the graph. It will not be moved in click-and-drag operations.
• Lock acts like Pin and, besides, sets the properties of the element as read-only.
• Cut deletes the element and saves it in the clipboard, for further pasting.
• Copy saves the element in the clipboard, for further pasting.
• Paste puts the contents previously copied in the clipboard into the design area. Note that after copying elements, you can also paste them to other contexts; for example, as images in a Microsoft Office application.

The available menu commands with Text editor are described below.

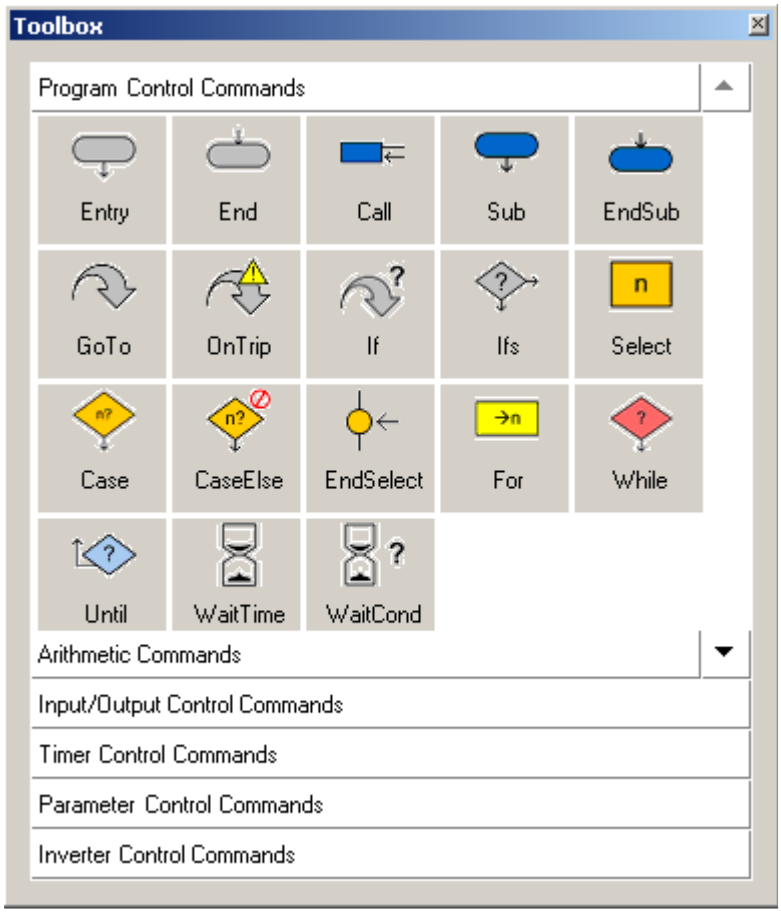
• *Find looks for* the selected text on the program code.
• *Replace* exchange the selected text on the program code.
• *Cut* deletes the element and saves it in the clipboard, for further pasting.
• *Copy* saves the element in the clipboard, for further pasting.
• *Paste* puts the contents previously copied in the clipboard into the design area. Note that after copying text elements, you can also paste them to other contexts; for example, as text in a Microsoft Office application.
• *Go to Subroutine* jumps to the selected text subroutine
• *Go to Label* jumps to the selected text label.
• *Undo* reverts the latest change.
• *Redo* recovers the most recently undone change.
• *Help* will show the CX-Drive help.

3-6 Toolbox window

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

The Toolbox is displayed when Drive Programming is entered. You can also show or hide it by clicking on *Drive Programming* | Toolbox in the View menu.

The Toolbox is displayed by default docked at the rightmost side of CX-Drive. You can resize it as needed to better display its elements. Also, you can toggle its docking by right clicking near the window's edges.



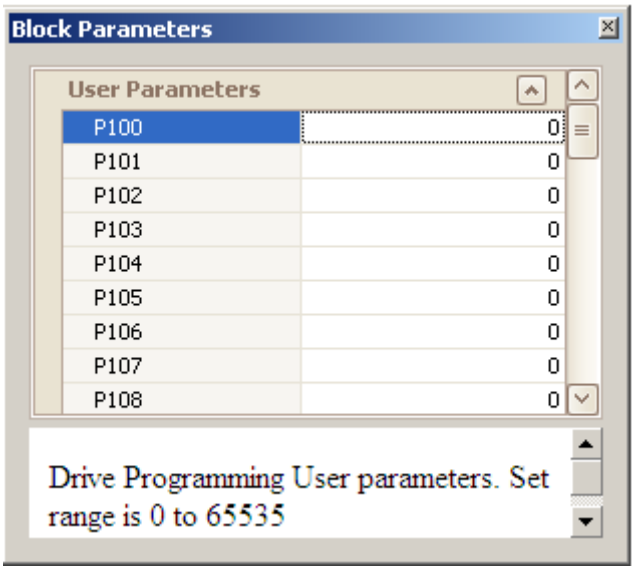
You can also choose its displays style by right-clicking on it with the mouse. Three styles are available: Large Icons, Small Icons, and List. In any style, placing the mouse cursor on a block will show a short help text for it.

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

3-7 Block Parameters window

The Block Parameters window allows the user to edit drive program parameters which act as variables of the program. The parameters are organized in categories. Block parameters is displayed when Drive Programming is entered. You can also show or hide it by clicking on Drive Programming | Block Parameters in the View menu.

Block Parameters is displayed by default docked at the rightmost side of CX-Drive. You can resize it as needed to better display its elements. Also, you can toggle its docking by right clicking near the window's edges.



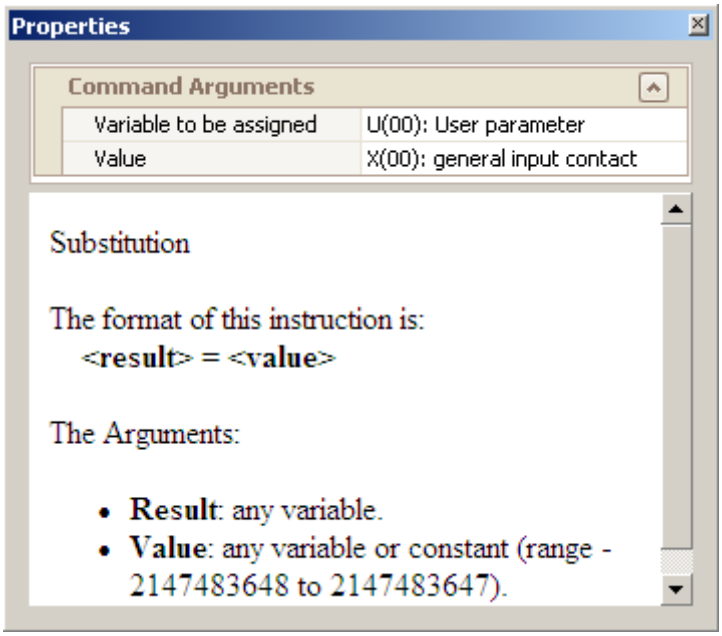
To change the value of a block parameter, place the cursor at its row and click on the edition box to the right of its name. Enter the new value. CX-Drive will warn you if the value exceeds the valid range. At the lower part of the window, a help text for the block parameters is displayed.

3-8 Properties window

The Properties window allows the user to edit the properties of the drive program block which is currently selected in the Flowchart Program Editor.

Properties are displayed when Drive Programming is entered. You can also show or hide it by clicking on Drive Programming | Properties in the View menu.

Properties are displayed by default docked at the rightmost side of CX-Drive. You can resize it as needed to better display its elements. Also, you can toggle its docking by right clicking near the window's edges.



To change one block command argument, place the cursor at its row and click on the edition box to the right of its name.

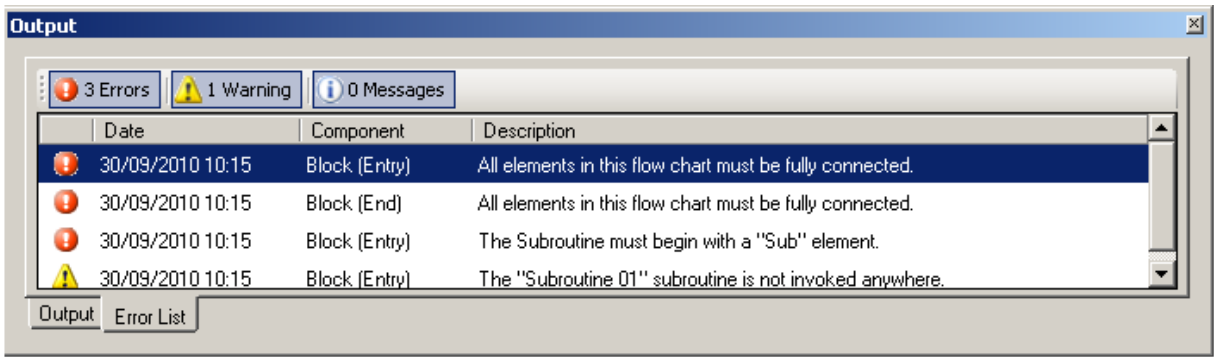
▪ If the block argument has options, a second click of the mouse will unfold the available options for you to select.

▪ If the block argument does not have options, clicking on its current value will enable you to change it by typing a new one. CX-Drive will warn you if the value exceeds the valid range.

If the block argument can have both an option and a custom value, clicking on the unfold sign at the right of the cell will unfold the available options, whereas clicking anywhere in the cell text, you will be able to edit it.

3-9 Output window

It shows the compilation errors and warnings of the currently edited drive program after it is compiled. Errors will prevent the program to be correctly compiled. Warnings will allow compilation, but advise customer of abnormal conditions.



· The 

Error(s) button toggles displaying error in the list.

· The 

Warning(s) button toggles displaying warnings in the list.

· The 

Message(s) button toggles displaying informative message in the list.

Messages in the list show the following information:

· Date: The date and time when the error was generated.
· Component: Identifies the element with an error.
· Description: The text of the error or warning message.

The list is automatically cleared every time a Compile is done.

3-10 Creating a program with Flowchart Editor

Follow the steps described below to create a drive program.

1. Open the Program Editor. The Drive Programming auxiliary windows (Toolbox, Block Parameters, Properties and Error List) will be displayed automatically.
2. Select on the menu “New Tab” New Task (flowchart) or New Subroutine(flowchart).
3. Drag each block of the program from the Toolbox window to the Flowchart Program Editor.
4. After dragging a block, edit its properties by clicking on it and edit the arguments in the Properties window.
5. Connect the blocks accordingly.
6. Edit the drive program variables in the Block Parameters window.
7. You may now compile the program, transfer it to the drive, export it, etc.

Alternatively, you can connect to a drive which has a program and transfer it, following the simple steps described below.

1. Open the Program Editor. The auxiliary Drive Programming windows (Toolbox, Block Parameters and Properties) will be displayed automatically.
2. Click the Transfer from Drive button in the program Editor Toolbar. The program will be transferred from the drive and automatically displayed in the Program Editor designer area.
3. You may now edit the program, compile it, transfer it to the drive, export it, etc.

When a drive program is present, you can also transfer it from and to the drive with the Transfer to Drive and Transfer from Drive buttons of the CX-Drive toolbar. In this case, a message dialog will ask you whether to transfer the parameters, the program or both.

3-11 Creating a program with Text Editor

Follow the steps described below to create a drive program:

1. Open the program Editor. The Drive Programming auxiliary windows (Toolbox, Block Parameters, Properties and Error List) will be displayed automatically.
2. The three ways to edit the code are:
    a. Manual typing
    b. Calling code snippets (Ctrl+Space)
    c. Drag & Drop commands from Toolbox window (like Flowchart Editor)
3. You may now compile the program, transfer it to the drive, export it, etc.

Note 1: The Text editor is supported from CX-Drive version 2.50.

Note 2: Text and Flowchart Tasks/Subroutines can be used simultaneously within same program.

3-12 Run a program

After transferring the program to the device, you can run the program with the  command or setting the next inverter parameters:

- MX2 and RX:

Parameter	Value	Description
A017 – Drive Programming Selection	0: Disabling	Drive Programming program will be stopped.
	1: PRG terminal	Drive Programming program will run by digital input. Set terminal to PRG function.
	2: Always	Drive Programming program will be always running.

- LX:

Parameter	Value	Description
F025 – Drive Programming function selection	0: Disable	Drive Programming program will be disabled.
	1: Enable	Drive Programming program will be Enabled.
F026 – Drive Programming RUN trigger selection	0: TRM('PRG' terminal)	Drive Programming program will run by terminal. Set terminal to PRG function.
	1: PARAM (setting F025=enable)	Drive Programming program will run if F025 = enable

3-13 Comments – Text Editor

Only it is possible to add comments in a Text editor task or subroutine. To add a comment in a text line press the character "'" follow by the comment. The comment will be showed in a green color format.

- **Examples**

```
#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
```

Note: if you convert a Text task or subroutine to Flowchart, the comments will be lost.

3-14 #Alias definition – Text Editor

Only it is possible to define an alias in a Text editor task and before the command '**entry**'. It's not possible to define an alias in a subroutine.

Alias definitions are user-friendly names given to parameters, variables, commands and numerical constants. There are two kinds of alias definition:

· **Local alias**: this alias definition only can be used in the current task and his subroutines, and not in the other tasks and subroutines that the program could have. This is the format for a local alias definition inside a task:

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

· **Examples**

```
#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**: this alias definition can be used in all the tasks and subroutines. This is the format for a global alias definition:

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

· **Examples**

```
#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
```

Note 1: The alias will be lost converting a text task/subroutine to flowchart. CX-Drive will show a message advising about this issue.
Note 2: reserved words cannot be used like an alias. A compilation error will appear.

3-15 #Region definition – Text Editor

A Region definition can be only defined in a text task or subroutine. It is useful to define code regions to clarify the program source code.

· Examples

```
#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...
```

3-16 Conversion from Flowchart to Text

There are two options to convert from Flowchart program to text:

Command	Image	Description
Convert Flowchart to text		It converts current Flowchart Task/Subroutine to text.
Convert whole program to text		It converts whole program to text.

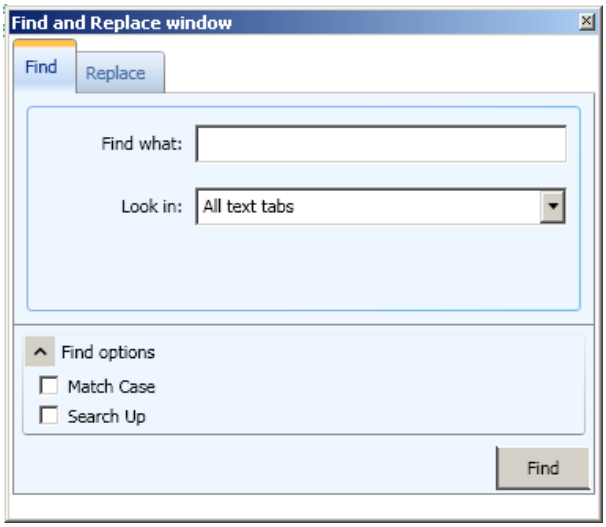
3-17 Conversion from Text to Flowchart

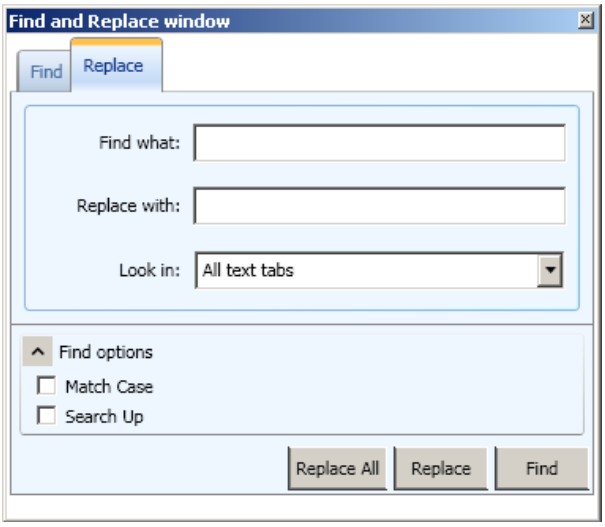
There are two option to convert from Text to Flowchart:

Command	Image	Description
Convert Text to Flowchart		It converts current Flowchart Task/Subroutine to text.
Convert whole program to Flowchart		It converts whole program to text.

3-18 Find & Replace function

Function only available in text mode. It allows look for an exchange code inside your text program.
To use Find function press the icon or the shortcut keys '**Ctrl + F**'.
To use Replace function press the icon or the shortcut keys '**Ctrl+F**'





4- Drive Program structure

The programming language is a Flowchart/Text language.
The inverter can process five parallel tasks.
The processing is as following diagram.

\+ >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> **Command Execution** >>>>>>>>>>>>>>>>>>>>>>>>>>>>> -

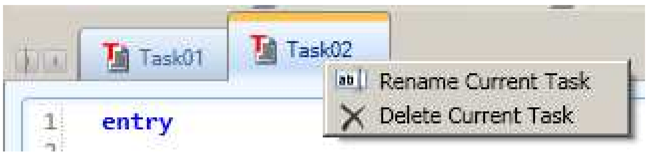
Task 1		**Task 2**		**Task 3**		**Task 4**		**Task 5**		**Elapsed time**
line	Code	line	Code	line	Code	line	Code	line	Code	
1	Entry		Entry		Entry		Entry		Entry	2 [ms]
2	| **top**	6	| **top**	11	| **top**	14	| **top**	19	| **top**	4
3	Process A	7	Process B	12	Process C	15	Process D	20	Process E	6
4	|	8	|	13	| **Goto top**	16	|	21	|	8
5	| **Goto top**	9	|	11	| **top**	17	|	22	|	10
2	| **top**	10	| **Goto top**	12	Process C	18	| **Goto top**	23	|	12
3	Process A	6	| **top**	13	| **Goto 11**	14	| **top**	24	| **Goto top**	14
4	|	7	Process B	11	| **top**	15	Process D	19	| **top**	16
5	| **Goto top**	8	|	12	Process C	16	|	20	Process E	18
2	| **top**	9	|	13	| **Goto top**	17	|	21	|	20
3	Process A	10	| **Goto top**	11	| **top**	18	| **Goto top**	22	|	22
4	|	6	| **top**	12	Process C	14	| **top**	23	|	24
5	| **Goto top**	7	Process B	13	|**Goto top**	15	Process D	24	| **Goto top**	26
2	| **top**	8	|	11	| **top**	16	|	19	| **top**	28
3	Process A	9	|	12	Process C	17	|	20	Process E	30
4	|	10	| **Goto top**	13	|**Goto top**	18	| **Goto top**	21	|	32 [ms]

Inside each task, subroutines can be associated, but maximum nesting (call inside a subroutine call) is 8 level depth.

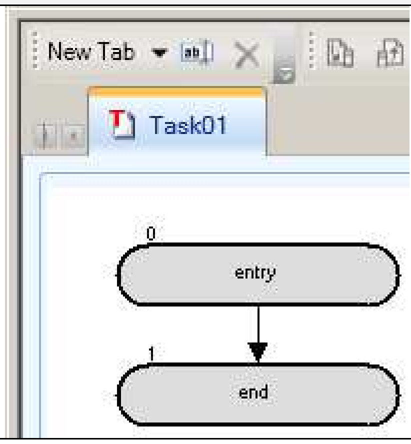
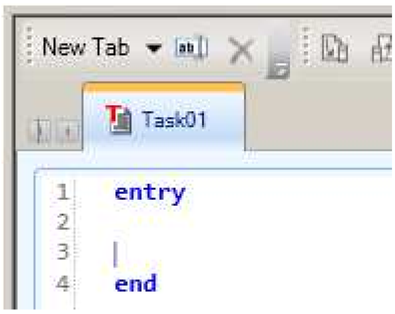
4-1 Tasks

When Drive Programming it's open, an empty task appears by default: 

.
With the right mouse click we can *Delete Current Task* or *Rename Current Task*.



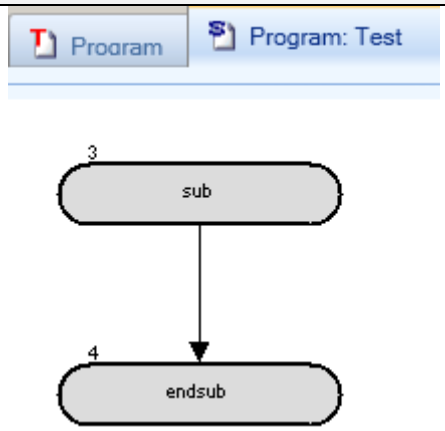
Every task must begin with **Entry** and must finish with the **End** *Control Commands*.

Flowchart	**Text**
	

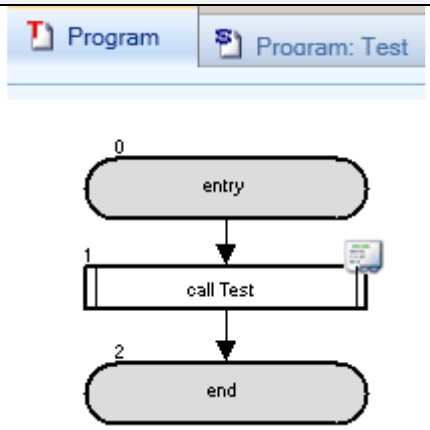
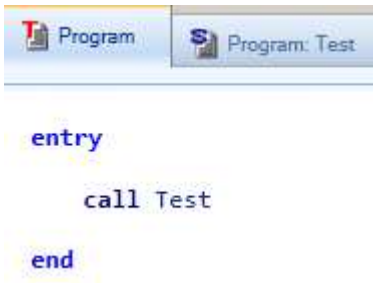
4-2 Subroutines

Subroutines are useful to organize your program into parts of code that you can reuse in other programs or in the same program. For insert a subroutine press the button or and a new subroutine will appear. Like on Tasks, you can delete or rename a subroutine.

Every subroutine must begin with the **Sub** block, and end with the **EndSub** *Control Command*.

Flowchart	Text
	

The subroutine is executed via the call command with the subroutine name.

Flowchart	Text
	

It is only possible to call a subroutine that is associated with the task. To be used with other task, a copy of the subroutine is necessary on the task.

5- Drive Programming user variables

5-1 Initial Data

U(00) to U(31) or User parameters

U(00) to U(31)	Description	Range of values	Default	Unit	Data size	Attribute
	User variable	0 to 65535	Data stored in P100 to P131	-	Unsigned 1-word	R/W

User variables are the general-purpose functions that can be used as unsigned 1-word. The data written from a drive program to the user-defined variables is not stored in the inverter's EEPROM. The variables will restore the initial settings when the inverter power is turned off. The user-defined variables correspond to inverter parameters "P100" to "P131". You can also change the settings of user-defined variables from the digital operator. The changes made from the digital operator will be stored in EEPROM. This is also possible to emulate from drive programming by using the EepWrt command.

The variables P129 to P131 (U(29) to U(31)) are saved at power down of the inverter automatically. This function may not work under heavy load (motor output current) or too small inverter (low capacity in DC bus). In case of trouble it is recommended to disable the inverter output to preserve the energy in the capacitors.

UL(00) to UL(07) or Internal User parameters.

UL(00) – UL(07)	Description	Range of values	Default	Unit	Data size	Attribute
	Internal user variable	-2^{31} to $2^{31}-1$	0	-	Signed 2-word	R/W

Internal user variables are the general-purpose functions that can be used as unsigned 2-word variables, for example, to temporarily store arithmetic operation results. The initial values can be set via the initial program data.

5-2 Setting Variables

Set-Freq	Description	Range of values	Default	Unit	Data size	Attribute
	Output frequency setting	0 to 40000	0	0.01 Hz	Unsigned 1-word	R/W

When A001=7 (Freq. ref. from Drive Programming), it becomes the frequency set point of the inverter. Always reflects the reading of parameter F001, regardless the setting of A001. This variable is not stored in the inverter EEPROM. It will be restored to initial setting after power cycle.
When the inverter receives an operation command (FW=1 or RV=1), it accelerates the motor up to the frequency that was set last.

ACCEL	Description	Range of values	Default	Unit	Data size	Attribute
	Acceleration time setting	1 to 360000	Note 1	0.01 sec	Unsigned 2-word	R/W

This variable can be used to read and write the motor acceleration time in the inverter. It is enabled only when the setting of accel/decel time input selection (P031) is "03" (PRG). (Please note that it does not correspond to the setting of inverter parameter "F002"). The data written to this variable is not stored in the inverter's EEPROM. It restores initial value after power cycle.

Note 1: By default (when the inverter power is turned on), the acceleration time follows the setting of the inverter parameter "F002", "F202", or "F302". For details, refer to the Inverter Instruction Manual.
Note 2: When a program writes a value to this variable, the value is reflected in the inverter in a 40-ms cycle, which conforms to the standard inverter specifications.

DECEL	Description	Range of values	Default	Unit	Data size	Attribute
	Deceleration time setting	1 to 360000	Note 1	0.01 sec	Unsigned 2-word	R/W

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

Note 1: By default (when the inverter power is turned on), the deceleration time follows the deceleration (1) time setting “F003”, “F203” or “F303”. For details, refer to the Inverter Instruction Manual.
Note 2: When a program writes a value to this variable, the value is reflected in the inverter in a 40-ms cycle, which conforms to the standard inverter specifications.

5-3 Inverter Monitor Variables (This units does not always corresponds with the display units)

FM	Description	Range of values	Default	Unit	Data size	Attribute
(d001)	Output frequency monitor	0 to 40000	-	0.01 Hz	Unsigned 1-word	R

The data monitored with this variable corresponds to the data monitored by the output frequency monitor (d001). This variable is read-only.

Iout	Description	Range of values	Default	Unit	Data size	Attribute
(d002)	Output current monitor	0 to 9999	-	0.01 %	Unsigned 1-word	R

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

Dir	Description	Range of values	Default	Unit	Data size	Attribute
(d003)	Rotation direction monitor	0: Stop 1: Normal rotation 2:Reverse rotation	-	-	Unsigned 1-word	R

The data monitored with this variable corresponds to the data monitored by the rotation direction monitor (d003). This variable is read-only.

PID-FB	Description	Range of values	Default	Unit	Data size	Attribute
(d004)	Process variable (PV), PID feedback monitoring	0 to 9990000	0	0.01 %	Unsigned 2-word	R

The data monitored with this variable corresponds to the data monitored by the process variable (PV), PID feedback monitor (d004). This variable is read-only.

F-CNV	Description	Range of values	Default	Unit	Data size	Attribute
(d007)	Scaled output frequency monitor	0 to 3996000	-	0.01	Unsigned 2-word	R

The data monitored with this variable corresponds to the data monitored by the scaled output frequency monitor (d007). This variable is read-only.

Tmon	Description	Range of values	Default	Unit	Data size	Attribute
(d012)	Torque monitor	-200 to 200	-	%	Unsigned 1-word	R

The data monitored with this variable corresponds to the data monitored by the torque monitor (d012). This variable is read-only.

Vout	Description	Range of values	Default	Unit	Data size	Attribute
(d013)	Output Voltage monitor	0 to 6000	-	0.1v	Unsigned 1-word	R

The data monitored with this variable corresponds to the data monitored by the output voltage monitor function (d013). This variable is read only.

Power	Description	Range of values	Default	Unit	Data size	Attribute
(d014)	Power monitor	0 to 9999	-	0.1 Kw	Unsigned 1-word	R

The data monitored whit this variable corresponds to the data monitored by the power monitor (d014). This variable is read only.

RUN-Time	Description	Range of values	Default	Unit	Data size	Attribute
(d016)	Run Time monitor	0 to 999999	-	Hour	Unsigned 2-word	R

The data monitored with this variable corresponds to the data monitored by the cumulative operation RUN time monitor (d016). This variable is read only.

On-Time	Description	Range of values	Default	Unit	Data size	Attribute
(d017)	Power-on time monitor	0 to 999999	-	Hour	Unsigned 2-word	R

The data monitored with this variable corresponds to the data monitored by the cumulative power-on time monitor (d017). This variable is read-only.

UMon(0) to Umon(2)	Description	Range of values	Default	Unit	Data size	Attribute
(d025 to d027)	User Parameter monitor 0 to 2	-2^{31} to $2^{31}-1$	0	-	Signed 2-word	R/W

The data monitored with these variables corresponds to the data monitored on d025, d026 and d027. These are monitors available for the user Drive Programming application

POS	Description	Range of values	Default	Unit	Data size	Attribute
(d030)	Current Position monitor	- (2^{28} - 1) to 2^{28} - 1 *[- (2^{30} - 1) to 2^{30} - 1]*	-	1	Signed 2-word	R

The data referenced with this variable corresponds to the data monitored by the current position monitor (d030).
With RX when "03" (high-resolution absolute position control) has been selected for control pulse setting (P012), the range in brackets "[]" applies.

ERR-CNT	Description	Range of values	Default	Unit	Data size	Attribute
(d080)	Trip counter monitor	0 to 65535	-	Nº of times	Unsigned 1-word	R

The data monitored with this variable corresponds to the data monitored by the trip counter monitor (d080).

ERR(1)-ERR(6)	Description	Range of values	Default	Unit	Data size	Attribute
(d081-d086)	Trip monitor 1 to 6	0 to 127	-	-	Unsigned 1-word	R

The data monitored with these variables correspond to the data monitored by trip monitors 1 to 6 (d081 to d086).

DCV	Description	Range of values	Default	Unit	Data size	Attribute
(d102)	DC voltage monitor	0 to 9999	-	0.1 Vdc	Unsigned 1-word	R

The data referenced with this variable corresponds to the data monitored by the DC voltage monitor (d102).

STATUS	Description	Range of values	Default	Unit	Data size	Attribute
	Inverter status monitor	-	-	-	Unsigned 1-word	R

This variable can be used to reference inverter status information.
The information is reflected with the following bit weights:

Bit 9 to 15	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserve	Under voltage	Reset	Over voltage suppression	Over current suppression	Overload suppression	Retry	Reverse	Trip	Run

5-4 Terminal Variables

Input/Output Control Instructions

X(00) - X(11)	Description	Range of Values	Data Size	Attribute
	Input terminal 0 to 11	0: Off 1: On	bit	R

See table below for each inverter function number:

Input	Inverter function number		
	MX2	RX	LX
X(00) – MI1	56	56	49
X(01) – MI2	57	57	50
X(02) – MI3	58	58	51
X(03) – MI4	59	59	52
X(04) – MI5	60	60	53
X(05) – MI6	61	61	54
X(06) – MI7	62	62	55
X(07) – MI8	63	63	56
X(08) – MI9	--	--	57
X(09) – MI10	--	--	58
X(10) – MI11	--	--	59
X(11) – MI12	--	--	60

Xw	Description	Range of Values	Data Size	Attribute
	Input terminal (word)	0 to 65535	Unsigned 1-word	R

Instruction to access contact inputs by word. Each bit reflects one of the inputs.

Y(00) - Y(06)	Description	Range of Values	Data Size	Attribute
	Output terminal 0 to 6	0: Off 1: On	bit	R/W

See table below for each inverter function number:

Output	Inverter function number		
	MX2	RX	LX
Y(00) – MO1	44	44	35
Y(01) – MO2	45	45	36
Y(02) – MO3	46	46	37
Y(03) – MO4	--	47	38
Y(04) – MO5	--	48	39
Y(05) – MO6	--	49	40
Y(06) – MO7	--	--	41

Yw	Description	Range of Values	Data Size	Attribute
	Output terminal (word)	0 to 65535	Unsigned 1-word	R/W

This variable can be used to change the digital output terminals in units of word. Each output is one bit.

XA(0) –XA(2)	Description	Range of Values	Data Size	Attribute
XA(0)	General-purpose analog input (O terminal)	0 to 10000	Unsigned 1- word (0.01%)	R
XA(1)	General-purpose analog input (OI terminal)			
XA(2)	General-purpose analog input (O2 terminal) only for RX and LX			

These variables can be used to monitor the analog input to the O and OI and O2 terminals. Terminals [O]-[L], [OI]-[L], [O2]-[L]. Associated parameters (A011 to A015, A101 to A105, A111 to A114). XA(2) is only available for Rx and LX.

YA(0) –YA(2)	Description	Range of Values	Data Size	Attribute
YA(0)	General-purpose analog output (EO terminal for MX2) (FM terminal for RX and LX)	0 to 10000	Unsigned 1-word (0.01%)	R/W
YA(1)	General-purpose analog output (AM terminal)			
YA(2)	General-purpose analog output (AMI terminal) only for RX and LX			

With this variables we can monitor the analog outputs (any multifunction assigned to them), or write analog output if YA(0) to YA(2) are assigned to analog multifunction parameters (C027, C028 and C029). Value is reflected as a data range from 0% to 100.00%. YA(2) is only available for RX and LX**.**

UB(00) – UB(07)	Description	Range of Values	Data Size	Attribute
	Internal user contact (bit access)	0: Off 1: On	bit	R/W

These variables can be used as bit variable for the user.

UBw	Description	Range of Values	Data Size	Attribute
	Internal user contact (word access)	0 to 255	Unsigned 1-word	R/W

The bit variables reflected as single word.

TC(0) - TC(7)	Description	Range of Values	Data Size	Attribute
	Timer counters (0 to 7) (Unit: 10ms)	0 to $2^{31}-1$	Unsigned 2-word	R/W

The timer counters “TC(0)” to “TC(7)” operate as 31-bit-free-running timer counters. They start with the user program startup and are incremented in a 10-ms cycle.

When a timer-start instruction (timer set) or delay operation instruction (delay on or delay off) is executed, the timer counter corresponding to the instruction operates as the counter for output to a specified timer contact. In this case, the counter is cleared to zero when the instruction is executed, start counting, and then stops counting upon reaching the specified count. When a timer-stop instruction (timer off) is executed, the timer counter corresponding to the instruction is cleared to zero and operates as a 31-bi-free-running timer counter that is incremented in a 10-ms cycle.

TD(0) - TD(7)	Description	Range of Values	Unit	Attribute
	Timer contact output 0 – 7 (bit access)	0: Off 1: On	Unsigned 1-word	R

The data in timer contact output variables “TD(0)” to “TD(7)” change only when these variables are specified in the timer-start instruction (timer set) or delay operation instruction (delay on or delay off). A timer contact output variable is set to “0”(off) when the counter corresponding to the contact output is cleared to zero, the variable is set to “1”(on) when the counter stops counting (the timing action selected finish).
While a timer counter variable “TC(k)” is being used for a free-running timer counter, timer contact output variable “TD(k)” corresponding to the timer counter variable retains its status.

TDw	Description	Range of Values	Unit	Attribute
	Timer contact output (word access)	0 to 255	Unsigned 1-word	R

It access to the timer counter outputs as word.

5-5 Digital input Functions

These variables correspond to the settings available for the digital multifunction input terminals. Setting the variable to 1 will simulate the function as if the terminal was closed in a digital input. It is interesting to note that the multifunction does not need to be configured in order to use the function.

E.g. FW := 1 will generate a RUN Forward command (as used in some examples).
Please refer to the inverter user manual for details about the individual functions.
Values: · 0: Off
· 1: On

Function	Description	MX2	RX	LX	Usage	Comment
FW	Forward	√	x	x	R/W	C001-C009 = 00
RV	Reverse	√	√	x	R/W	C001-C009 = 01
CF1-CF4	Multi-speed 1-4	√	√	x	R/W	C001-C009 = 02-05
JG	Jogging	√	√	x	R/W	C001-C009 = 06
DB	External Brake	√	√	x	R/W	C001-C009 = 07
SET	Second control	√	√	√	R/W	C001-C009 = 08
2CH	2nd acceleration/deceleration time	√	√	x	R/W	C001-C009 = 09
FRS	Free run	√	√	√	R/W	C001-C009 = 11
EXT	External trip	√	√	√	R/W	C001-C009 = 12
USP	Unattended start protection	√	√	x	R/W	C001-C009 = 13
CS	Change from commercial power	√	√	x	R/W	C001-C009 = 14
SFT	Software lock	√	√	√	R/W	C001-C009 = 15
AT	Change of analog input	√	√	x	R/W	C001-C009 = 16
SET3	3rd control	x	√	x	R/W	C001-C009 = 17
RS	System reset	√	√	√	R/W	C001-C009 = 18
STA	Start of 3 wires	√	√	x	R/W	C001-C009 = 20
STP	Stop of 3 wires	√	√	x	R/W	C001-C009 = 21
F/R	Forward/Reverse of 3 wires	√	√	x	R/W	C001-C009 = 22
PID	Switch PID	√	√	x	R/W	C001-C009 = 23
PIDC	Reset of PID integration	√	√	x	R/W	C001-C009 = 24
CAS	Control gain switching	x	√	x	R/W	C001-C009 = 26
UP	Increasing speed from remote	√	√	x	R/W	C001-C009 = 27
DWN	Decreasing speed from remote	√	√	x	R/W	C001-C009 = 28
UDC	Clear data from remote operation	√	√	x	R/W	C001-C009 = 29
OPE	Change to operator	√	√	x	R/W	C001-C009 = 31
SF1-SF7	Multi-speed bit 1-7	√	√	x	R/W	C001-C009 = 32-38
OLR	Overload protection switch	√	√	x	R/W	C001-C009 = 39
TL	Torque Limit Enable	√	√	x	R/W	C001-C009 = 40
TRQ1-2	Torque Limit Selection 1-2	√	√	x	R/W	C001-C009 = 41-42
PPI	P/PI switching	x	√	x	R/W	C001-C009 = 43
BOK	Brake Confirmation	√	√	x	R/W	C001-C009 = 44
ORT	Orientation	x	√	x	R/W	C001-C009 = 45
LAC	LAD Cancel	√	√	x	R/W	C001-C009 = 46
PCLR	Clear Position Deviation	√	√	x	R/W	C001-C009 = 47
STAT	Pulse train position command input permission	x	√	x	R/W	C001-C009 = 48
ADD	Add Setting Frequency	√	√	x	R/W	C001-C009 = 50
F-TM	Forced Terminal Block	√	√	x	R/W	C001-C009 = 51
ATR	Torque reference input permission	√	√	x	R/W	C001-C009 = 52
KHC	Integrated power clear	√	√	x	R/W	C001-C009 = 53
SON	Servo ON	x	√	x	R/W	C001-C009 = 54
FOC	Preliminary excitation	x	√	x	R/W	C001-C009 = 55
X(00) – X(07)	Drive Programming (MI1-MI8)	√	√	x	R/W	C001-C009 = 56-63
AHD	Analog command on hold	√	√	x	R/W	C001-C009 = 65
CP1-3	Position command selection 1-3	√	√	x	R/W	C001-C009 = 66-68
ORL	Origin return limit signal	√	√	x	R/W	C001-C009 = 69
ORG	Origin return start signal	√	√	x	R/W	C001-C009 = 70

FOT	Forward driving stop	x	√	x	R/W	C001-C009 = 71
ROT	Reverse driving stop	x	√	x	R/W	C001-C009 = 72
SPD	Speed/Position switching	√	√	x	R/W	C001-C009 = 73
PCNT	Pulse counter	x	√	x	R/W	C001-C009 = 74
PCC	Pulse counter clear	x	√	x	R/W	C001-C009 = 75
GS1	GS1 input	√	x	x	R/W	C001-C009 = 77
Function	**Description**	**MX2**	**RX**	**LX**	**Usage**	**Comment**
GS2	GS2 input	√	x	x	R/W	C001-C009 = 78
RS485	Inverter communication start terminal	√	x	x	R/W	C001-C009 = 81
PRG	Executing Drive Program	√	x	x	R/W	C001-C009 = 82
HLD	HOLD Acceleration / deceleration stopping	√	x	x	R/W	C001-C009 = 83
ROK	Operation OK signal	√	x	x	R/W	C001-C009 = 84
DISP	Display limitation terminal	√	x	x	R/W	C001-C009 = 86
UP	Upward RUN	x	x	√	R/W	C001-C009 = 00
DOWN	Downward RUN	x	x	√	R/W	C001-C009 = 01
SPD1	Multi-speed 1 setting	x	x	√	R/W	C001-C009 = 02
SPD2	Multi-speed 2 setting	x	x	√	R/W	C001-C009 = 03
SPD3	Multi-speed 3 setting	x	x	√	R/W	C001-C009 = 04
OLR	Change OL-level	x	x	√	R/W	C001-C009 = 32
TL	Torque Limit enable	x	x	√	R/W	C001-C009 = 33
TRQ1	Change Torque Limit 1	x	x	√	R/W	C001-C009 = 34
TRQ2	Change Torque Limit 2	x	x	√	R/W	C001-C009 = 35
PCLR	Clear the current position	x	x	√	R/W	C001-C009 = 40
KHC	Kwh clear	x	x	√	R/W	C001-C009 = 46
X(00)-X(11)	Drive Programming	x	x	√	R/W	C001-C009 =49-60
EMP	Em-Power Operation	x	x	√	R/W	C001-C009 = 61
INS1	Inspection 1	x	x	√	R/W	C001-C009 = 62
INS2	Inspection 2	x	x	√	R/W	C001-C009 = 63
COK	Contactor check signal	x	x	√	R/W	C001-C009 = 64
BOK	Brake check signal	x	x	√	R/W	C001-C009 = 65
FP1-FP6	Floor position 1 to 6	x	x	√	R/W	C001-C009 = 66-71
PAL	Auto learning data latch trigger	x	x	√	R/W	C001-C009 = 72
TCL	Torque bias latch trigger	x	x	√	R/W	C001-C009 = 73
LVS	Leveling signal	x	x	√	R/W	C001-C009 = 74
NFS	Near floor signal	x	x	√	R/W	C001-C009 = 75
PRG	Program run	x	x	√	R/W	C001-C009 = 76
CMC	Control Mode change	x	x	√	R/W	C001-C009 = 77

Note:The LX inverter functions are available for the digital multifunction input terminals P140-P144 (Multi-Input [Ex.IN1-5] → 3G3AX-EIO-E: LX extra I/O board)

5-6 Digital Output Functions

These variables correspond to the settings available for the digital multifunction output terminals. The variable can read and used as it would be for an external device connected to the digital output configured for the function.
It is interesting to note that digital outputs are not required to be assigned in order to use the function within the program (in other words, no waste of digital outputs required).

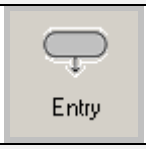
Function	Description	MX2	RX	LX	Usage	Comment
RUN	Running	√	√	√	R	C021 - C026 = 00
FA1	Reaching constant speed	√	√	√	R	C021 - C026 = 01
FA2	Greater than setting frequency	√	√	√	R	C021 - C026 = 02
OL	Overload preannounce	√	√	√	R	C021 - C026 = 03
OD	PID deviation overrate	√	√	x	R	C021 - C026 = 04
AL	Trip signal	√	√	√	R	C021 - C026 = 05
FA3	Only the setting frequency	√	√	√	R	C021 - C026 = 06
OTQ	Over torque/under torque	√	√	√	R	C021 - C026 = 07
IP	Signal during m. power interruption	x	√	√	R	C021 - C026 = 08
UV	Under voltage signal	√	√	√	R	C021 - C026 = 09
TRQ	Torque limitation signal	√	√	√	R	C021 - C026 = 10
RNT	RUN time over	√	√	√	R	C021 - C026 = 11
ONT	ON time over	√	√	√	R	C021 - C026 = 12
THM	Thermal warning	√	√	√	R	C021 - C026 = 13
ZS	0 Hz detection signal	x	x	√	R	C021 - C026 = 14
POK	Positioning complete	x	x	√	R	C021 - C026 = 16
FA4	Set frequency overreached 2	x	x	√	R	C021 - C026 = 17
FA5	Set frequency reached 2	x	x	√	R	C021 - C026 = 18
BRK	Brake open	√	√	x	R	C021 - C026 = 19
BER	Brake error	√	√	x	R	C021 - C026 = 20
ZS	Zero speed signal	√	√	x	R	C021 - C026 = 21
DSE	Speed deviation overrate	√	√	x	R	C021 - C026 = 22
POK	Positioning operation complete	√	√	x	R	C021 - C026 = 23
FA4	Greater than setting frequency 2	√	√	x	R	C021 - C026 = 24
FA5	Only the setting frequency 2	√	√	x	R	C021 - C026 = 25
OL2	Overload preannounce 2	√	√	x	R	C021 - C026 = 26
ODc	Analog O break detection	√	√	x	R	C021 - C026 = 27
OIDc	Analog OI break detection	√	√	x	R	C021 - C026 = 28
O2Dc	Analog 2 disconnection detection	x	√	x	R	C021 - C026 = 29
WAC	Capacitor life warning	x	X	√	R	C021 - C026 = 30
FBV	PID feedback comparison	√	√	x	R	C021 - C026 = 31
NDc	Communication break detection	√	√	x	R	C021 - C026 = 32
LOG1	Result of logic operation 1	√	√	x	R	C021 - C026 = 33
LOG 2	Result of logic operation 2	√	√	x	R	C021 - C026 = 34
LOG 3	Result of logic operation 3	√	√	x	R	C021 - C026 = 35
LOG 4	Result of logic operation 4	x	√	x	R	C021 - C026 = 36
LOG 5	Result of logic operation 5	x	√	x	R	C021 - C026 = 37
LOG 6	Result of logic operation 6	x	√	x	R	C021 - C026 = 38
WAC	Condenser life-span preannounce	√	√	x	R	C021 - C026 = 39
WAF	Fan life-span preannounce	√	√	x	R	C021 - C026 = 40
FR	Start contact signal	√	√	x	R	C021 - C026 = 41
OHF	Cooling fan over heat preannounce	√	√	x	R	C021 - C026 = 42
LOC	Low electricity signal	√	√	x	R	C021 - C026 = 43
Y(00)	Drive Programming (MO1)	√	√	x	R	C021 - C026 = 44
Y(01)	Drive Programming (MO2)	√	√	x	R	C021 - C026 = 45
Y(02)	Drive Programming (MO3)	√	√	x	R	C021 - C026 = 46
Y(03)	Drive Programming (MO4)	x	√	x	R	C021 - C026 = 47
Y(04)	Drive Programming (MO5)	x	√	x	R	C021 - C026 = 48
Y(05)	Drive Programming (MO6)	x	√	x	R	C021 - C026 = 49
IRDY	Operation setup complete	√	√	x	R	C021 - C026 = 50
FWR	Forward running signal	√	√	x	R	C021 - C026 = 51
RVR	Reverse running signal	√	√	x	R	C021 - C026 = 52
MJA	Serious failure signal	√	√	x	R	C021 - C026 = 53
WCO	Window comparator O	√	√	x	R	C021 - C026 = 54
WCOI	Window comparator OI	√	√	x	R	C021 - C026 = 55
WCO2	Window comparator O2	x	√	x	R	C021 - C026 = 56
MPS	Magnet pole position search	x	x	√	R	C021 - C026 = 57

Function	Description	MX2	RX	LX	Usage	Comment
FREF	Command frequency sel. mode	√	x	x	R	C021 - C026 = 58
REF	Command operation mode	√	x	x	R	C021 - C026 = 59
SETM	Setting motor	√	x	x	R	C021 - C026 = 60
EDM	STO operation monitor signal	√	x	x	R	C021 - C026 = 62
IRDY	Inverter ready	x	x	√	R	C021 - C026 = 44
FWR	Forward rotation	x	x	√	R	C021 - C026 = 45
RVR	Reverse rotation	x	x	√	R	C021 - C026 = 46
MJA	Major failure	X	x	√	R	C021 - C026 = 47
OL2	Overload advance signal 2	x	x	√	R	C021 - C026 = 19
TH-C	Thermal warning (CTL)	x	x	√	R	C021 - C026 = 20
NDc	Network disconnection	x	x	√	R	C021 - C026 = 23
WAF	Cooling-fan speed drop	x	x	√	R	C021 - C026 = 31
FR	Starting contact signal	x	x	√	R	C021 - C026 = 32
OHF	Heat sink overheat warning	x	x	√	R	C021 - C026 = 33
LOC	Low-current indication signal	x	x	√	R	C021 - C026 = 34
Y(00)	Drive Programming (MO1)	x	x	√	R	C021 - C026 = 35
Y(01)	Drive Programming (MO2)	x	x	√	R	C021 - C026 = 36
Y(02)	Drive Programming (MO3)	x	x	√	R	C021 - C026 = 37
Y(03)	Drive Programming (MO4)	x	x	√	R	C021 - C026 = 38
Y(04)	Drive Programming (MO5)	x	x	√	R	C021 - C026 = 39
Y(05)	Drive Programming (MO6)	x	x	√	R	C021 - C026 = 40
Y(06)	Drive Programming (MO7)	x	x	√	R	C021 - C026 = 41
CON	Contactor control signal	x	x	√	R	C021 - C026 = 51
BRK	Brake Control signal	x	x	√	R	C021 - C026 = 52
UPS	UPS protect direction search status	x	x	√	R	C021 - C026 = 54
UPD	UPS protect direction	x	x	√	R	C021 - C026 = 55
GMON	Gate suppress monitor	x	x	√	R	C021 - C026 = 56
SEQ	SEQ error	x	x	√	R	C021 - C026 = 58

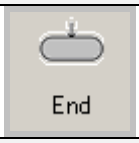
Note:The LX inverter functions are available for the digital multifunction output terminals P145-P147 (Multi-Output [Ex.OUT1-3] → 3G3AX-EIO-E: LX extra I/O board)

6- Drive Programming Instructions

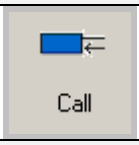
6-1 Control Commands

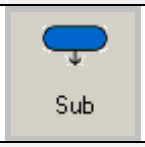
Entry		
Command	**Description**	**Arguments**
	It indicates the beginning of the task.	---
Format		

Note: It is compulsory to have Entry at the begging of each task.		

End		
Command	**Description**	**Arguments**
	It indicates the end of the task.	---
Format		

Note: It is compulsory to have End at the end of each task.		

Call		
Command	**Description**	**Arguments**
	It jumps to a subroutine	• **Subroutine**: Subroutines are identified by a name or alias defined by the user.
Format		
call <subroutine>		
Note: After the execution of the subroutine ends, the next instruction line after the call is executed.		

Sub		
Command	**Description**	**Arguments**
	It indicates the beginning of the subroutine.	---
Format		

Note: It is compulsory to have **Sub** at the beginning of each subroutine.		

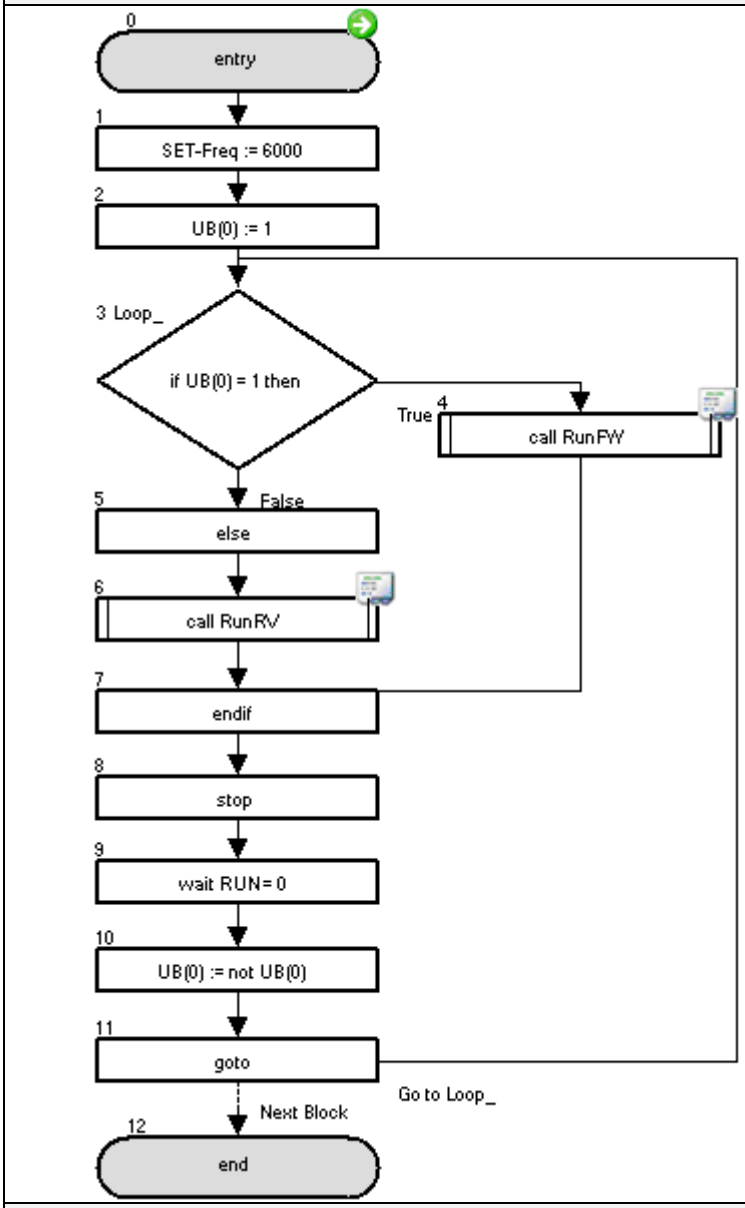
End Sub		
Command	**Description**	**Arguments**
	It indicates the end of a subroutine.	---
Format		

Note: It is compulsory to have **End Sub** at the end of each subroutine.		

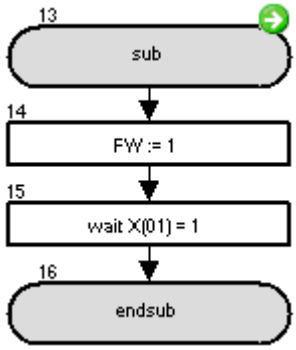
Example

Flowchart

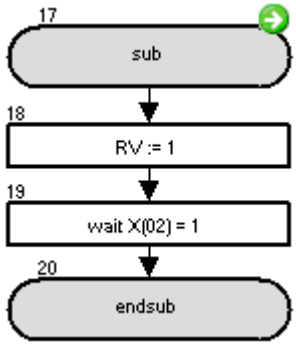
Main



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

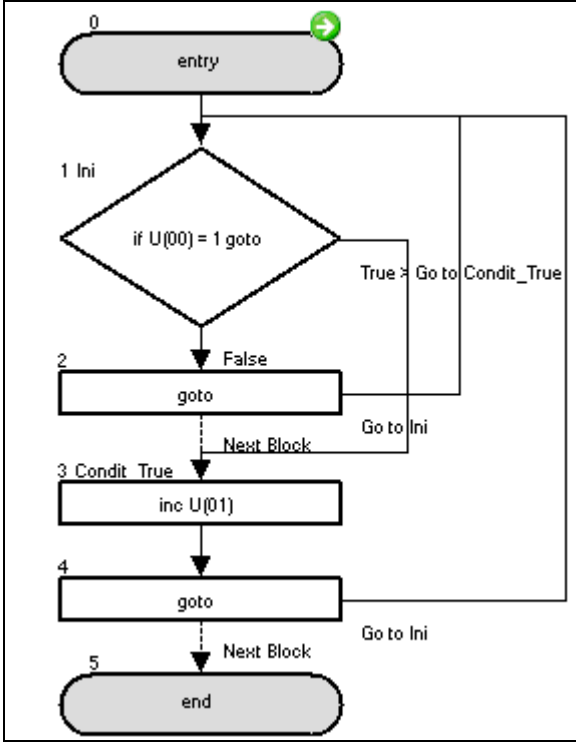
```
sub
    RV := 1
    wait X(02) = 1
endsub
```

A forward and reverse run at 60Hz is repeated continuously between two limits X(01) and X(02).

Go To		
Command	**Description**	**Arguments**
	Use this instruction to branch processing unconditionally to the step labeled with label name.	• **Label**: A name that is used to identify a particular function block in the task.
Format		
GoTo <label>		
Note: The instruction must also be connected to the next program block you want to be executed. This is necessary to make clear the flow of the program.		

Example

Flowchart	Text

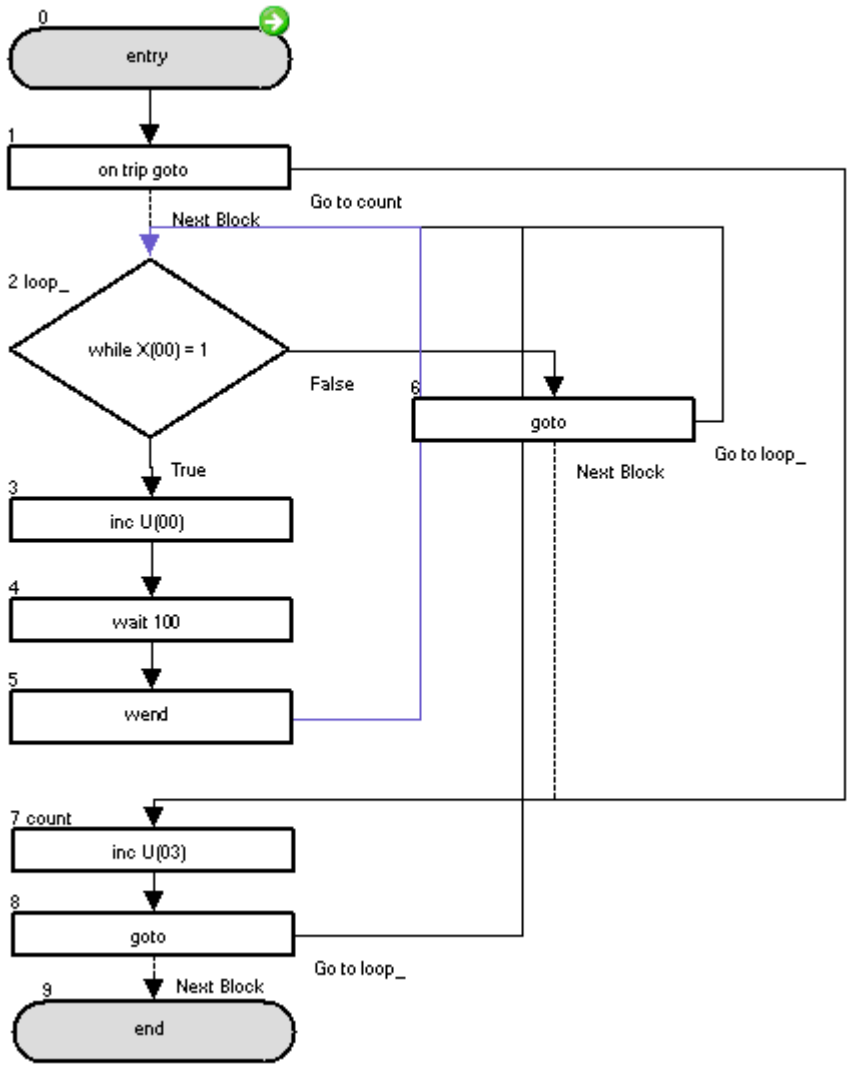
	`entry`
	
	`:Ini`
	`  if U(00) = 1 goto Condit_True`
	`  goto Ini`
	`:Condit_True`
	`  inc U(01)`
	`  goto Ini`
	
	`end`

*Change parameter P100 in order to test the **GoTo** function with this sample. When P100=1, P101 starts counting. When P100<>1 stops counting.*

On Trip		
Command	**Description**	**Arguments**
	This instruction makes conditional branching in case a trip in the inverter occurs.	• **Label**: A name that is used to identify a particular function block in the task.
Format		
On Trip goto <label>		
Note: The **On Trip** instruction works as a trigger arming. The instruction is executed once, if any trip occurs the program jumps immediately to the designated label, then the On trip trigger is disarmed.		

Example

Flowchart



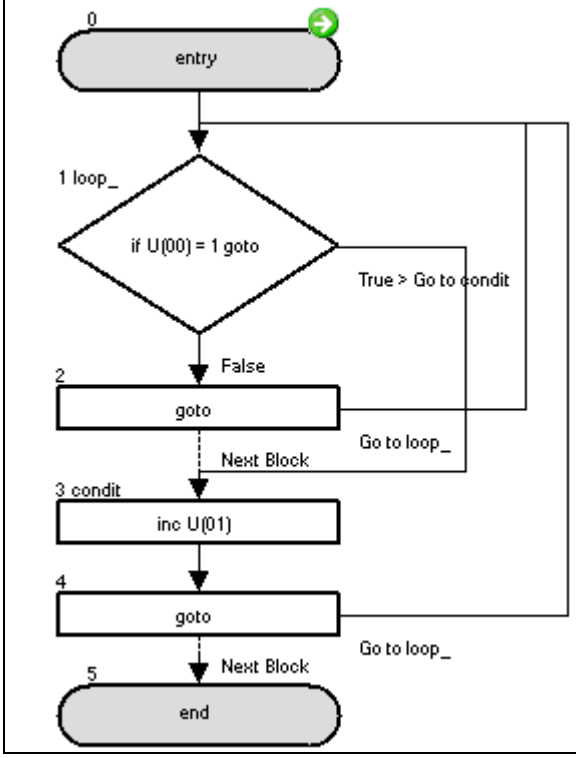
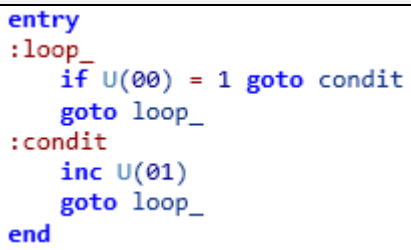
Text

```
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
```

When the digital input is set to ON value, then P100 parameter is incrementing every second. If a trip is generated (like by external trip input) then P103 increments count. And then goes to the beginning of the task.

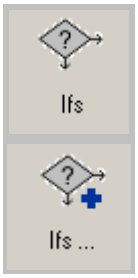
If		
Command	**Description**	**Arguments**
	Jump to a label when a condition is satisfied.	• **Condition**: A comparison between two variables or constant with the format<Left hand value><Comparison><Right Hand Value> -Left hand value: any variable or constant(range -128 to 127) -Comparison: =, <, >, <=, >=, <> -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		
If <condition> GoTo <label>		

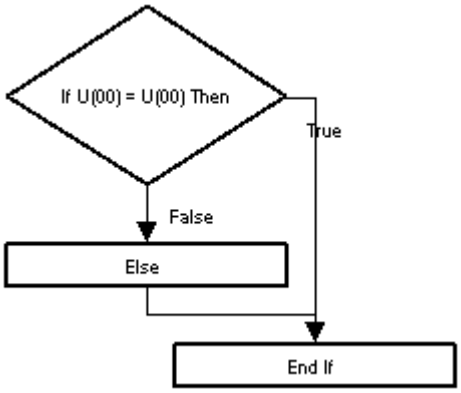
Example

Flowchart	Text
	


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

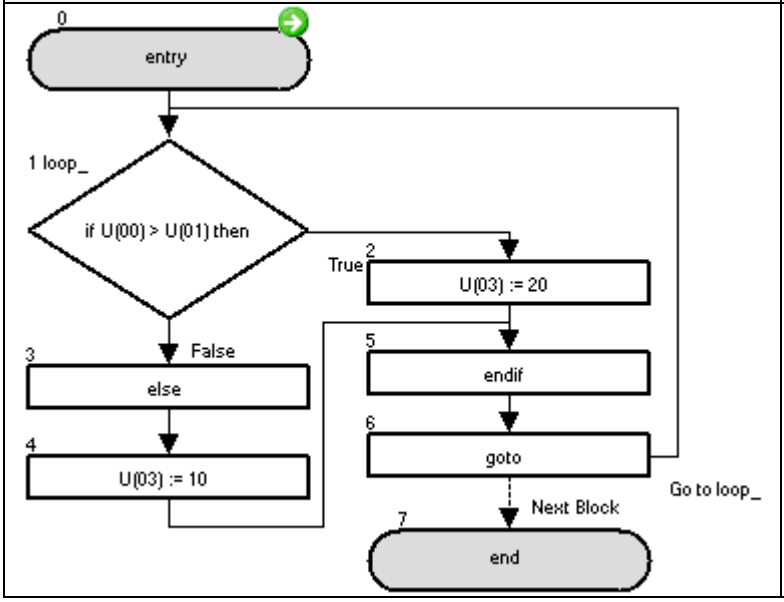
*Change parameter P100 in order to test the **GoTo** function with this sample. When P100 = 1, P101 starts counting. When P100<>1 stops counting.*

Ifs/ Else / End If		
Command	**Description**	**Arguments**
	This instruction executes different portion of code based on a condition. When the condition is met, this instruction executes <instruction set 1>. When the condition is not met, this instruction executes <instruction set 2>.	• **Condition**: A comparison between two variables or constant with the format <Left hand Value><Comparison><Right hand Value> -Left hand value: any variable or constant (range -128 to 127) -Comparison: =, <, >, <=, >=, <> -Right hand value: any variable or constant (range -128 to 127) • **Instruction set 1**: One or more instructions, until **Else** instruction. It can contain nested instructions (up to 8 levels) • **Instruction set 2**: One or more instructions, until **End If** instruction. It can contain nested instructions (up to 8 levels)
Format		

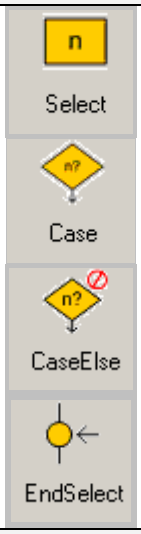
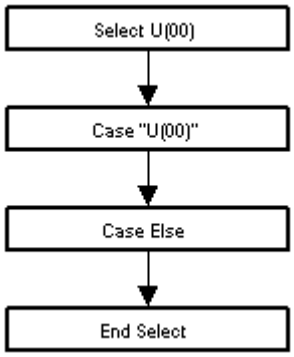
Ifs <condition> Then
  <instruction set 1>
Else
  <instruction set 2>
Endif
		

Example

Flowchart	Text

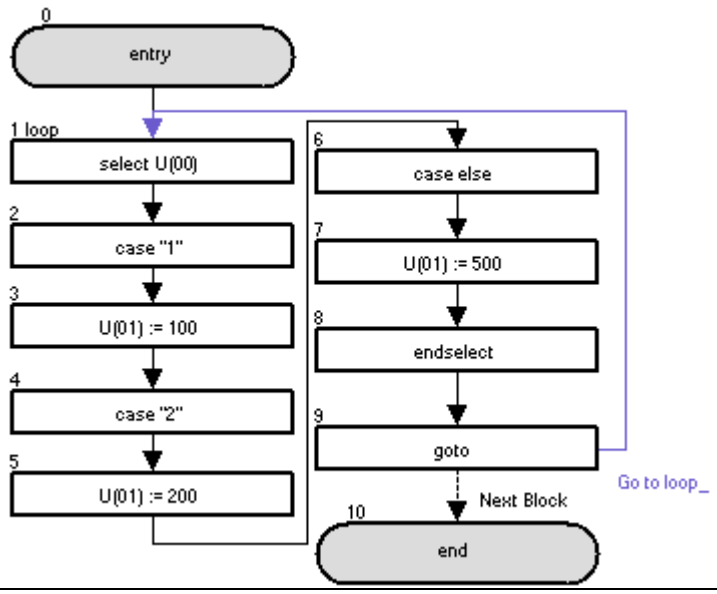
	`entry`
	`:loop_`
	`  if U(00) > U(01) then`
	`  U(03) := 20`
	`  else`
	`  U(03) := 10`
	`  endif`
	`  goto loop_`
	`end`

The example changes the value of P103 based on the value of parameter P100 and P101. If P100 is bigger than P101 then P103=10. If not P103=20.

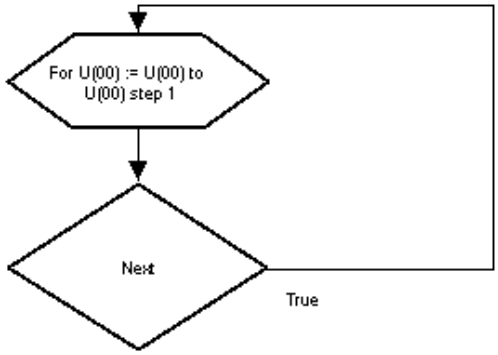
Select / Case / End Select		
Command	**Description**	**Arguments**
	This instruction allows multiple program sections to be executed depending on a variable value. For a particular CASE section it Executes <instruction set n> when <conditional variable> matches <conditional value n> If <conditional variable> doesn't match any of the CASE section the <instruction set if no other> (Case Else) is executed. This instruction is convenient when multiple choices have to be done from parameter value. It makes simple some if/then structures. This instruction is recommended to organize program by using subroutine calls as instruction set.	• **Conditional variable**: the instruction select variable. • **Conditional value x**: variable value. • **Instruction set x**: One or more instructions, until next **case** or **end select**. It can contain nested instructions (up to 8 levels).
Format		
Select <conditional variable> Case <conditional value 1> <instruction set 1> Case <conditional value 2> <instruction set 2> … Case Else <instruction set if no other> End select		

Example

Flowchart	Text

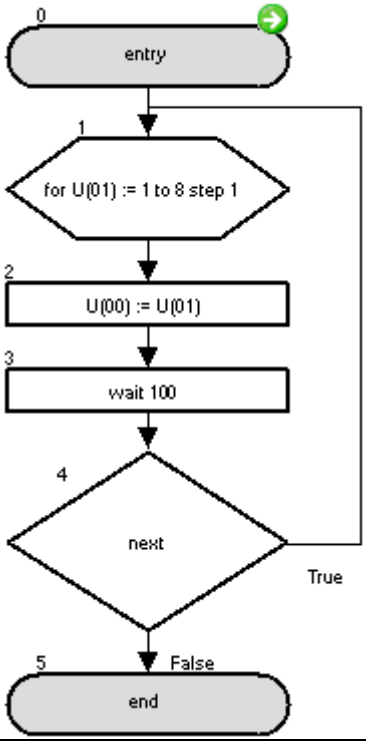
	```
	entry
	:loop_
	    select U(00)
	        case "1"
	            U(01) := 100
	        case "2"
	            U(01) := 200
	        case else
	            U(01) := 500
	    endselect
	    goto loop_
	end
	``

The P101 parameter is set to 100, 200, 300 or 500 depending on the value of the P100 parameter (1, 2, 3 or any other, respectively).

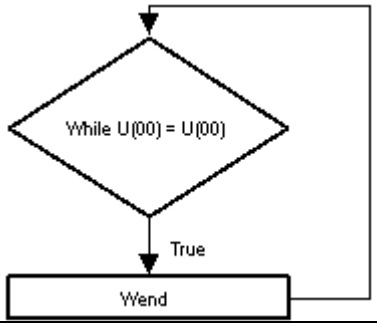
For / Next		
Command	**Description**	**Arguments**
	Executes <instruction set> repeatedly until <variable> reaches <end value>. Each cycle <incremental value> is added to <variable>.	• **Variable**: any variable • **Start value**: Initial value, it is the value assigned to the variable in the first loop. (Constant value from -128 to 127) • **End value**: Value that exits the loop. (Constant value from -128 to 127) • **Incremental value**: The variable will be incremented by this value each loop. (Constant value from -128 to 127) • **Instruction set**: One or more instructions, until **Next** instruction. It can contain nested instructions (up to 8 levels)
Format		
For <variable><start value><end value><incremental value> <instruction set> Next		

Example

Flowchart	Text

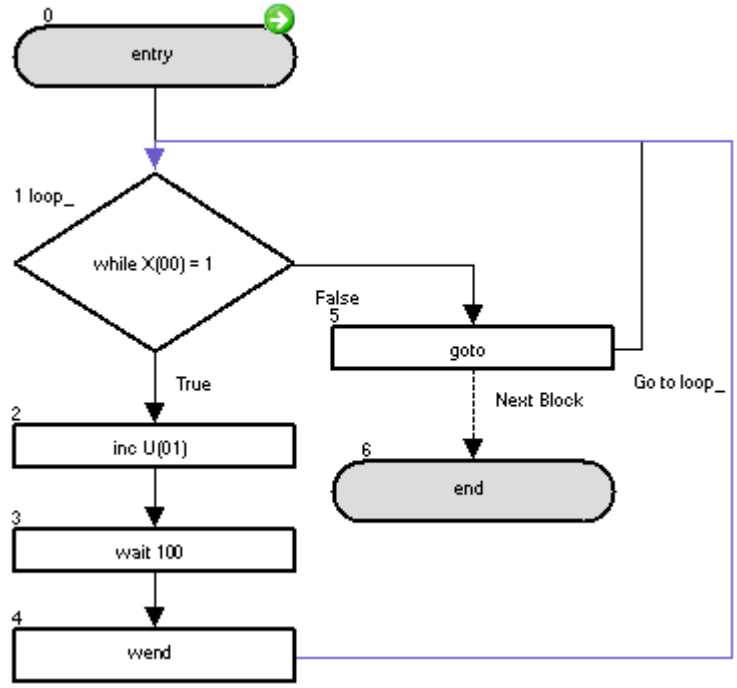
	`entry`
	`  for U(01) := 1 to 8 step 1`
	`  U(00) := U(01)`
	`  wait 100`
	`  next`
	`end`

This example make the variable U(00) P(100) count from 1 to 8 each second.

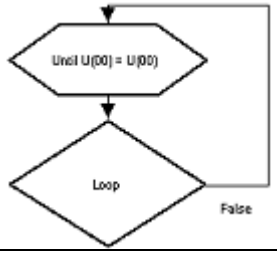
While / Wend		
Command	**Description**	**Arguments**
	Executes <instruction set> while a condition is met.	• **Condition**: A comparison between two variables or constants with the format <Left hand value><Comparison><Right Hand Value> - Left hand value: any variable or constant (range -128 to 127). - Comparison: =, <, >, <=, >=, <> - Right hand value: any variable or constant (range -128 to 127). • **Instruction set**: One or more instructions, until **Wend** instruction. It can contain nested instructions (up to 8 levels)
Format		
While <condition> <instruction set> Wend		

Example

Flowchart	Text

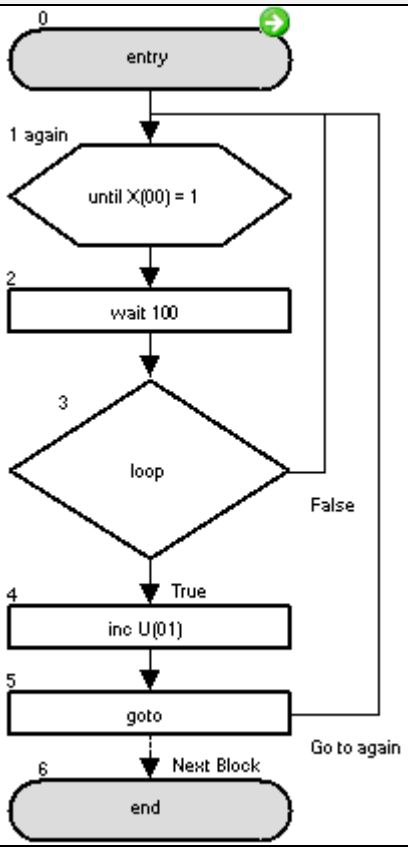
	```
	entry
	:loop_
	  while X(00) = 1
	  inc U(01)
	  wait 100
	  wend
	  goto loop_
	end
	``

*The code will increment P101 parameter every second while the digital input X(00) is closed (**while-wend** loop). If it is open, P101 is not increased (**GoTo-label loop** loop; the **while** – **wend** portion is not executed). Digital input has to be configured in the multifunction input.*

Until / Loop		
Command	**Description**	**Arguments**
	Executes <instruction set> until a <condition> is met.	• **Condition**: A comparison between two variables or constants with the format <Left hand value><Comparison><Right Hand Value> -Left hand value: any variable or constant (range -128 to 127) -Comparison: =,<,>,<=,>=,<> -Right hand value: any variable or constant (range -128 to 127) • **Instruction set**: One or more instructions, until **Loop** instruction. It can contain nested instructions (up to 8 levels)
Format		
Until <condition> <instruction set> Loop		

Example

Flowchart	Text

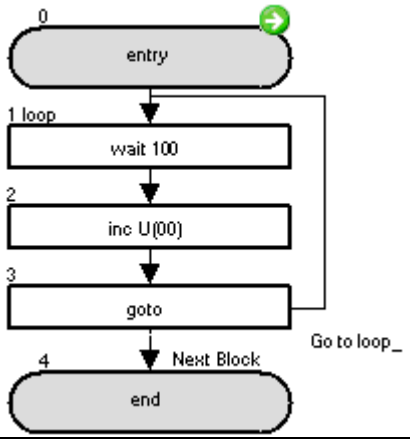
	`entry`
	`:again`
	`  until X(00) = 1`
	`  wait 100`
	`  loop`
	`  inc U(01)`
	`  goto again`
	`end`

*This code will increment while the digital input is closed. If it is open, then it will stay in the **until-loop** portion. The check of the input is every second because of this structure. Digital input has to be configured in the multifunction input.*

Wait		
Command	**Description**	**Arguments**
 	This instruction makes the program wait for a number of seconds or until a condition is met.	• **Value**: any variable or constant (time in 10 x ms). • **Condition**: A comparison between two variables or constant with the format <Left hand value><Comparison><Right Hand Value> -Wait Time value from 0 to 32767 * 10ms -Left hand value: any variable or constant (range -128 to 127) -Comparison: =, <, >, <=, >=, <>. -Right hand value: any variable or constant (range -128 to 127)
Format		
Wait <value> or <condition> **NOTE:** WaitTime is not accurate way to measure time, please use internal timers or external Real Time Clock in LCD for accurate time measurement.		

Example Wait Time: wait during a time period.

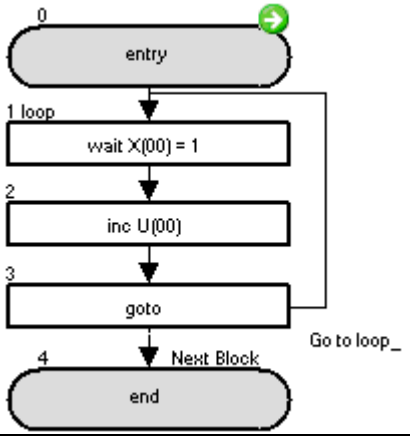
Flowchart	Text

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

The P100 parameter is increased every second.

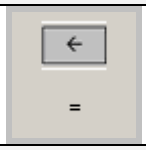
Example Wait condition: wait for condition.

Flowchart	Text

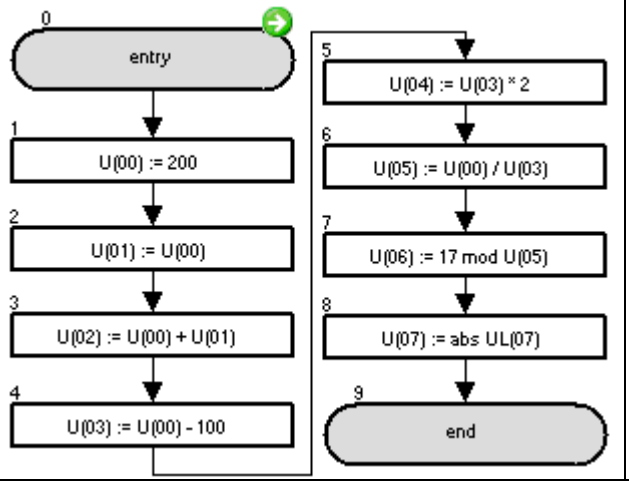
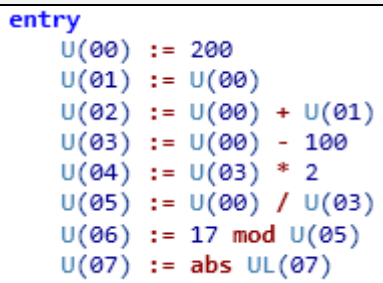
	`entry`
	`:loop_`
	`  wait X(00) = 1`
	`  inc U(00)`
	`  goto loop_`
	`end`

The program waits until the digital input is closed (you need to set one of the multifunction inputs for this), and then P100 parameter is increased.

6-2 Arithmetic and Logic Commands

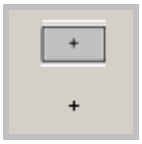
= (Substitution)		
Command	**Description**	**Arguments**
	Assigns <value> to <result>.	• **Result**: any variable. • **Value**: any variable or constant (range -2147483648 to 2147483647).
Format		
<result> = <value>		
Warning: Drive programming does not control overflow/underflow. The application should take care.		

Example

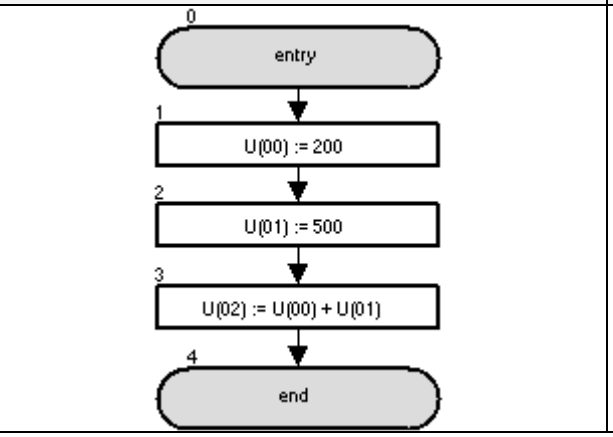
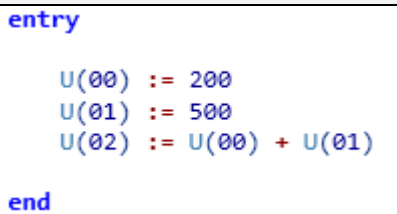
Flowchart	Text
	


```
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
```

The P100 and P101 parameters are set to 200.

+ (Addition)		
Command	**Description**	**Arguments**
	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 -2147483648 to 2147483647).
Format		
<result> = <value 1> + <value 2>		
Warning: Drive Programming does not control overflow/underflow. The application should take care.		

Example

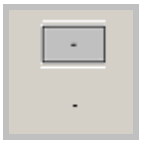
Flowchart	Text
	


```
entry

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

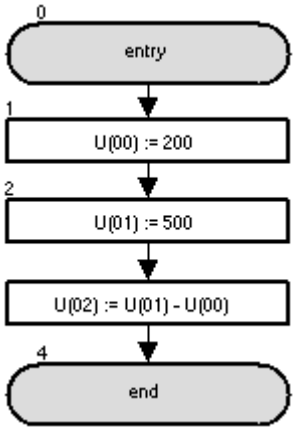
end
```

The P102 parameter calculation result is 700.

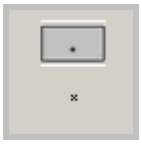
- (Subtraction)		
Command	**Description**	**Arguments**
	Subtracts <value 2> from <value 1>.	• **Result**: any variable. • **Value 1**: any variable or constant (range -128 to 127). • **Value 2**: any variable or constant (range -2147483648 to 2147483647).
Format		
<result>= <value 1> - <value 2>		
Warning: Drive Programming does not control overflow/underflow. The application should take care.		

Example

Flowchart	Text

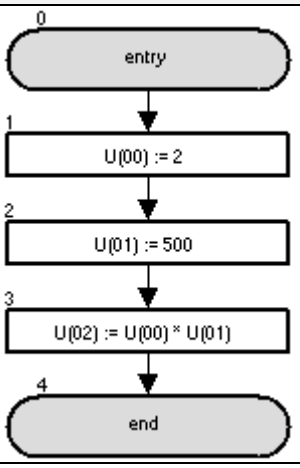
	`entry`
	`  U(00) := 200`
	`  U(01) := 500`
	`  U(02) := U(01) - U(00)`
	`end`

The P102 parameter calculation result is 300.

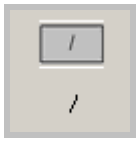
*(Multiplication)		
Command	**Description**	**Arguments**
	Multiplies <value 1> by <value 2>.	• **Result**: any variable. • **Value 1**: any variable or constant (range -128 to 127). • **Value 2**: any variable or constant (range -2147483648 to 2147483647).
Format		
<result> = <value 1> * <value 2>		
Warning: Drive Programming does not control overflow/underflow. The application should take care.		

Example

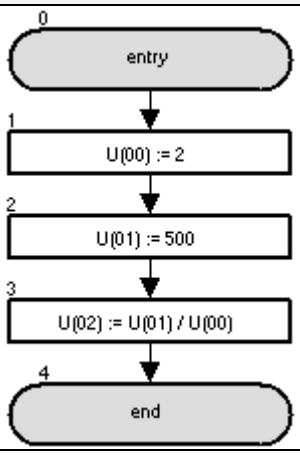
Flowchart	Text

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

The P102 parameter is set to 1000.

/(Division)		
Command	**Description**	**Arguments**
	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 -2147483648 to 2147483647).
Format		
<result> = <value 1> / <value 2>		
Warning: Drive Programming does not control overflow/underflow. The application should take care.		

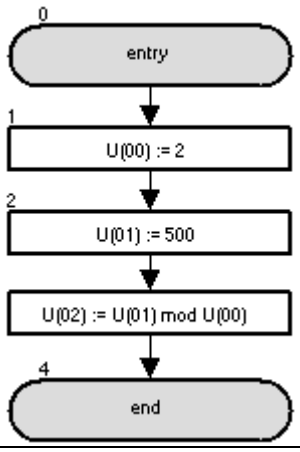
Example

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

The P102 parameter calculation result is 250.

% (Mod)		
Command	**Description**	**Arguments**
	Remainder of division.	• **Result**: Any variable. • **Value 1**: any variable or constant (range -128 to 127). • **Value 2**: any variable or constant (range -2147483648 to 2147483647)
Format		
<result> = <value 1> Mod <value 2>		
Warning: Drive Programming does not control overflow/underflow. The application should take care.		

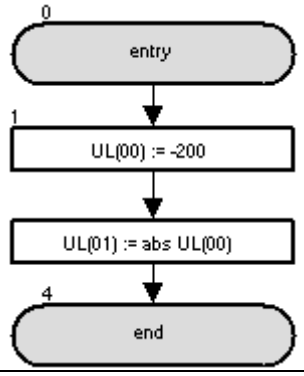
Example

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

The P102 parameter calculation result is 0.

Abs		
Command	**Description**	**Arguments**
	Absolute value.	• **Result**: any variable. • **Value**: any variable or constant (range -2147483648 to 2147483647).
Format		
<result> = Abs <value>		
Warning: Drive Programming does not control overflow/underflow. The application should take care.		

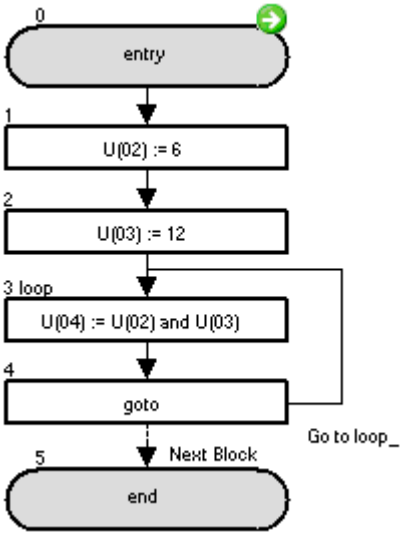
Example

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

The UL(01) variable is set to 200.

<table>
<tr><th colspan="3">And</th></tr>
<tr><th>Command</th><th>Description</th><th>Arguments</th></tr>
<tr>
<td> 
</td>
<td>And (logical product).
<table>
<tr><th>Value 1</th><th>Value 2</th><th>Result</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table></td>
<td>• Result: any variable.
• Value 1: any variable or constant (range -128 to 127).
• Value 2: any variable or constant (range -2147483648 to 2147483647).</td>
</tr>
<tr><th colspan="3">Format</th></tr>
<tr><td colspan="3"><result> = <value 1> And <Value 2></td></tr>
<tr><td colspan="3">Warning: Drive Programming does not control overflow/underflow. The application should take care.</td></tr>
</table>

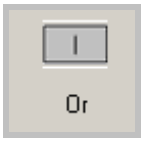
Example

<table>
<tr><th>Flowchart</th><th>Text</th></tr>
<tr>
<td> 
</td>
<td>

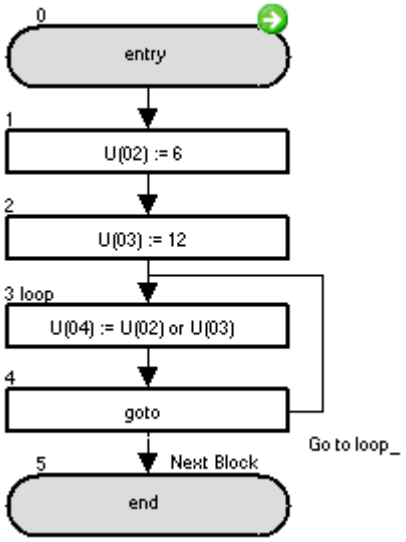
```
entry
    U(02) := 6
    U(03) := 12
:loop_
    U(04) := U(02) and U(03)
    goto loop_
end
```

</td>
</tr>
</table>

*The initial P104 parameter calculation result is 4, as 6 in binary format is 00000110 and 12 in binary format is 00001100, so the result of the **and** operation is 00000100 that is 4 in decimal format*
If P102 and P103 are changed by the user, then P104 will recalculate accordingly.

<table>
<tr><th colspan="3">Or</th></tr>
<tr><th>Command</th><th>Description</th><th>Arguments</th></tr>
<tr>
<td> 
</td>
<td>Or (logical addition).
<table>
<tr><th>Value 1</th><th>Value 2</th><th>Result</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table></td>
<td>• Result: any variable.
• Value 1: any variable or constant (range -128 to 127).
• Value 2: any variable or constant (range -2147483648 to 2147483647).</td>
</tr>
<tr><th colspan="3">Format</th></tr>
<tr><td colspan="3"><result> = <value 1> Or <value 2></td></tr>
<tr><td colspan="3">Warning: Drive programming does not control overflow/underflow. The application should take care.</td></tr>
</table>

Example

<table>
<tr><th>Flowchart</th><th>Text</th></tr>
<tr>
<td> 
</td>
<td>

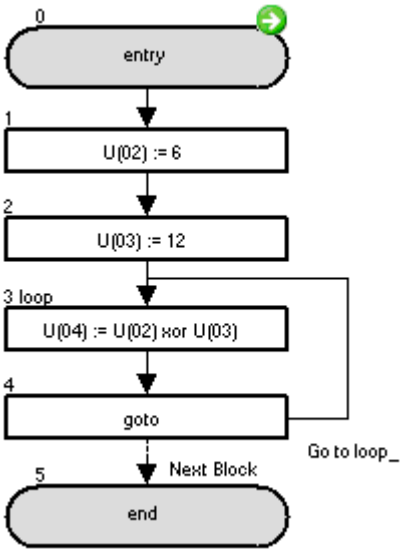
```
entry
    U(02) := 6
    U(03) := 12
:loop_
    U(04) := U(02) or U(03)
    goto loop_
end
```

</td>
</tr>
</table>

The initial P104 parameter calculation result is 14, as 6 in binary format is 00000110 and 12 in binary format is 00001100, so the result of the operation is 00001110 that is 14 in decimal format. If P102 and P103 are changed by the user, then P104 will recalculate accordingly.

<table>
<tr><th colspan="3">XOr</th></tr>
<tr><th>Command</th><th>Description</th><th>Arguments</th></tr>
<tr>
<td> 
</td>
<td>XOr(exclusive-or)
<table>
<tr><th>Value 1</th><th>Value 2</th><th>Result</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table></td>
<td>• Result: any variable.
• Value 1: any variable or constant (range -128 to 127).
• Value 2: any variable or constant (range -2147483648 to 2147483647).</td>
</tr>
<tr><th colspan="3">Format</th></tr>
<tr><td colspan="3"><result>= <value 1> XOr <value 2></td></tr>
<tr><td colspan="3">Warning: Drive Programming does not control overflow/underflow. The application should take care.</td></tr>
</table>

Example

<table>
<tr><th>Flowchart</th><th>Text</th></tr>
<tr>
<td> 
</td>
<td>

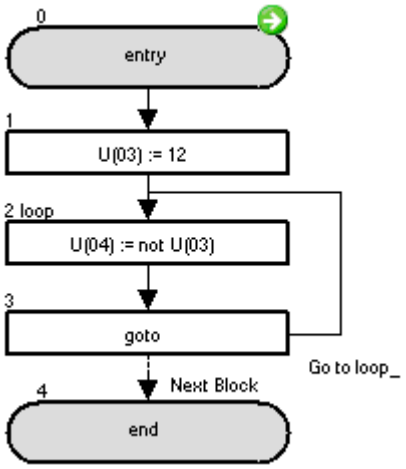
```
entry
    U(02) := 6
    U(03) := 12
:loop_
    U(04) := U(02) xor U(03)
    goto loop_
end
```

</td>
</tr>
</table>

*The initial P104 parameter calculation result is 10, as 6 in binary format is 00000110 and 12 in binary format is 00001100, so the result of the **XOr** operation is 00001010 that is 10 in decimal format. If P102 and P103 are changed by the user, then P104 will recalculate accordingly.*

<table>
<tr><th colspan="3">Not</th></tr>
<tr><th>Command</th><th>Description</th><th>Arguments</th></tr>
<tr><td> 
</td><td>Not (negation)
<table>
<tr><th>Value 1</th><th>Result</th></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td></tr>
</table></td><td>• Result: any variable, except variables with bit data size (Note 1)
• Value: any variable or constant, except variables with bit data size (Note 1) (range -2147483648 to 2147483647).</td></tr>
<tr><th colspan="3">Format</th></tr>
<tr><td colspan="3"><result> = Not<value></td></tr>
<tr><td colspan="3">Note: Unexpected result will be obtained with instructions like UB(1) = Not UB(0).
Please use XOr command to negate variables with bit data size in Drive Programming as shown on the next examples:
・Example 1: UB(1) = UB(0) Xor 1
・Example 2: UB(2) = X(00) Xor 1</td></tr>
<tr><td colspan="3">Warning: Drive Programming does not control overflow/underflow. The application should take care.</td></tr>
</table>

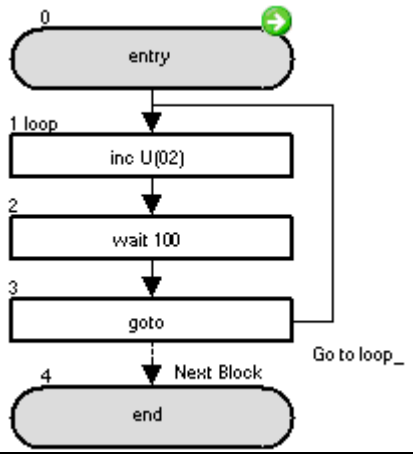
Example

<table>
<tr><th>Flowchart</th><th>Text</th></tr>
<tr><td> 
</td><td><pre>entry
    U(03) := 12
:loop_
    U(04) := not U(03)
    goto loop_
end</pre></td></tr>
</table>

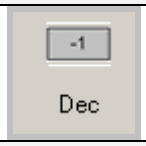
*The initial P104 parameter calculation result is 65523, as 12 in binary format is 0000000000001100, so the result of the **not** operation is 1111111111110011 that is 65523 in decimal format. If P103 is changed by the user, then P104 will recalculate accordingly.*

Inc		
Command	**Description**	**Arguments**
	Increments a value by 1.	• **Value**: any variable.
Format		
Inc<value>		
Warning: Drive Programming does not control overflow/underflow. The application should take care.		

Example

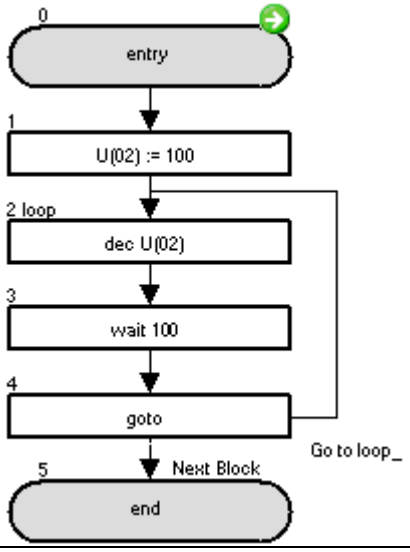
Flowchart	Text
	entry :loop_ inc U(02) wait 100 goto loop_ end

The P102 parameter is incremented by 1 every second.

Dec		
Command	**Description**	**Arguments**
	Decrements a value by 1.	• **Value**: any variable.
Format		
Dec<value>		
Warning: Drive Programming does not control overflow/underflow. The application should take care.		

Example

Flowchart	Text

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

The P102 parameter is decremented by 1 every second.

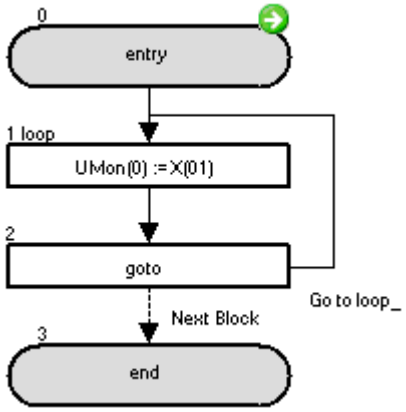
6-3 Input/Output Control Commands

For memory optimization, use Input/Output Control Commands (4 bytes) instead of the Equal Arithmetic Command "=" (8 bytes).

Var = X(i)		
Command	**Description**	**Arguments**
	Instruction to access contact inputs. Reflects the state of the input.	• **Variable**: any variable (the value of the variable will be 0 or 1). • **i**: Number of the contact input (range 0 to 11).
Format		
<variable>=X(i)		
Note: The inputs have to be assigned to digital multifunction input (by the multifunction 56 to 63). X(02) is not necessarily input 2 (depends where MF 58 is). X(00) = MI1 X(01) = MI2 X(02) = MI3 X(03) = MI4 X(04) = MI5 X(05) = MI6 X(06) = MI7 X(07) = MI8 X(08) = MI9 X(09) = MI10 X(10) = MI11 X(11) = MI12 Note: more details in chapter *5-4 Terminal Variables*.		

Example

Flowchart	Text

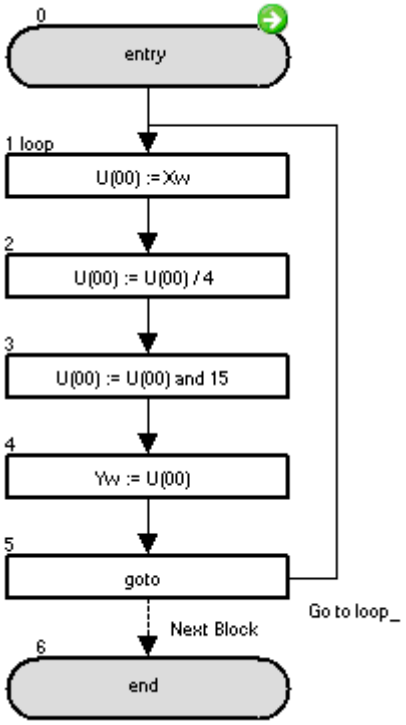
	`entry`
	`:loop_`
	`  UMon(0) := X(01)`
	`  goto loop_`
	`end`

The state of the input terminal X(01) is monitored on the d025 parameter.

Var = Xw		
Command	**Description**	**Arguments**
	Instruction to access contact inputs by word. Each bit reflects one of the inputs.	• **Variable**: any variable.
Format		
<variable> = Xw		
Note: The inputs have to be assigned to digital multifunction input (by the multifunction 56 to 63 for MX2 and RX or 49 to 60 on LX) Xw = 1 → bit 0 Xw = 2 → bit 1 Xw = 4 → bit 2 Xw = 8 → bit 3 Xw = 16→ bit 4 Xw = 32→ bit 5 Xw = 64→ bit 6 Xw = 128 → bit 7 (only for RX and LX) Xw = 256 → bit 8 (only for LX with extension I/O) Xw = 512 → bit 9 (only for LX with extension I/O) Xw = 1024 → bit 10 (only for LX with extension I/O) Xw = 2048 → bit 11 (only for LX with extension I/O)		

Example

Flowchart	Text

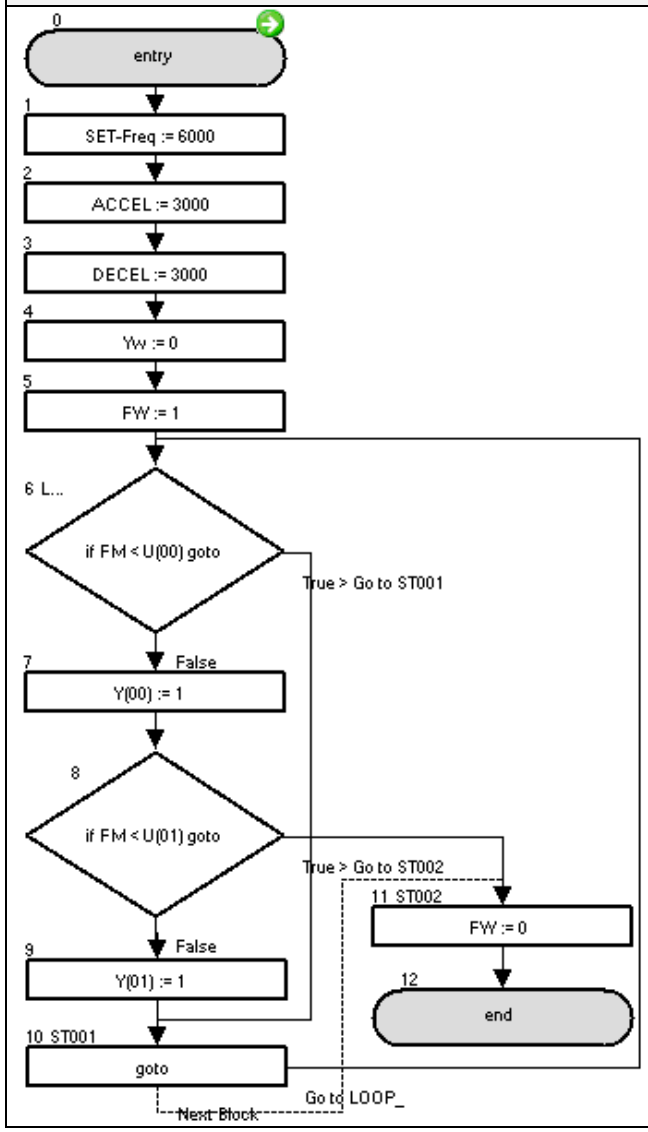
	`entry`
	`:loop_`
	`  U(00) := xw`
	`  U(00) := U(00) / 4`
	`  U(00) := U(00) and 15`
	`  yw := U(00)`
	`  goto loop_`
	`end`

This example acquires the state of the X(02) - X(05) input terminals and outputs it to Y(00) -Y(03) output terminals. To cut X(00) - X(01), the U(00) value is divided by 4. To cut X(06) - X(07), the U(00) value is masked by 15.

Y(i) = value		
Command	**Description**	**Arguments**
	Instruction to access digital outputs.	• **i**: Number of the contact output (range 0 to 6) • **Value**: any variable or constant.
Format		
Y(i)=<value>		
Note: The inputs have to be assigned to digital multifunction output (by the multifunction 44 to 49 for MX2 and RX and 35 to 41 for LX). Y(00) = MO1 Y(01) = MO2 Y(02) = MO3 Y(03) = MO4 Y(04) = MO5 Y(05) = MO6 Y(06) = MO7 Note: more details in chapter *5-4 Terminal Variables*		

Example

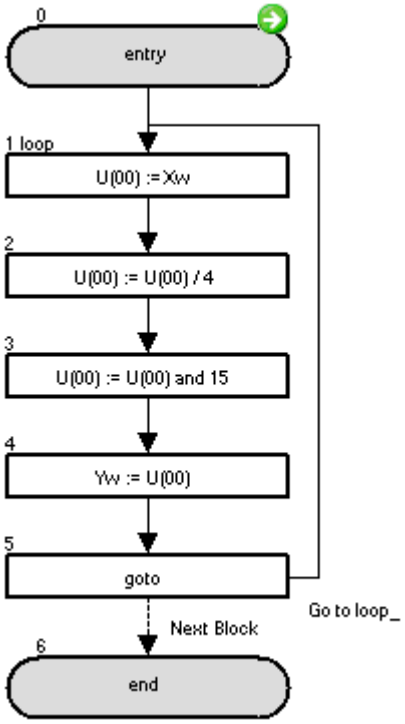
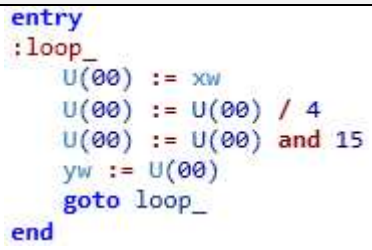
Flowchart	Text

	```
	entry
	  SET-Freq := 6000
	  ACCEL := 3000
	  DECEL := 3000
	  yw := 0
	  FW := 1
	:LOOP_
	  if FM < U(00) goto ST001
	  Y(00) := 1
	  if FM < U(01) goto ST002
	  Y(01) := 1
	:ST001
	  goto LOOP_
	:ST002
	  FW := 0
	end
	``

To test this example, initialize the user variables with the following value: U(00) = 1000, U(01) = 2000, U(02) = 3000. Y(00) – Y(01) are sequentially turned on every 10Hz step of the output frequency.

Yw = value		
Command	**Description**	**Arguments**
Yw=value	Instruction to access digital outputs by word. Each bit reflects one of the outputs.	• **Value**: any variable or constant
Format		
Yw = <value>		
Note: The inputs have to be assigned to digital multifunction input (by the multifunction 44 to 49 for MX2 and RX, 35 to 41 for LX). Yw = 1 → bit 0 Yw = 2 → bit 1 Yw = 4 → bit 2 Yw = 8 → bit 3 (only if expanded I/O board used) Yw = 16 → bit 4 (only if expanded I/O board used) Yw = 32 → bit 5 (only if expanded I/O board used, and enough outputs) Yw = 64 → bit 6 (only if expanded I/O board for LX is used)		

Example

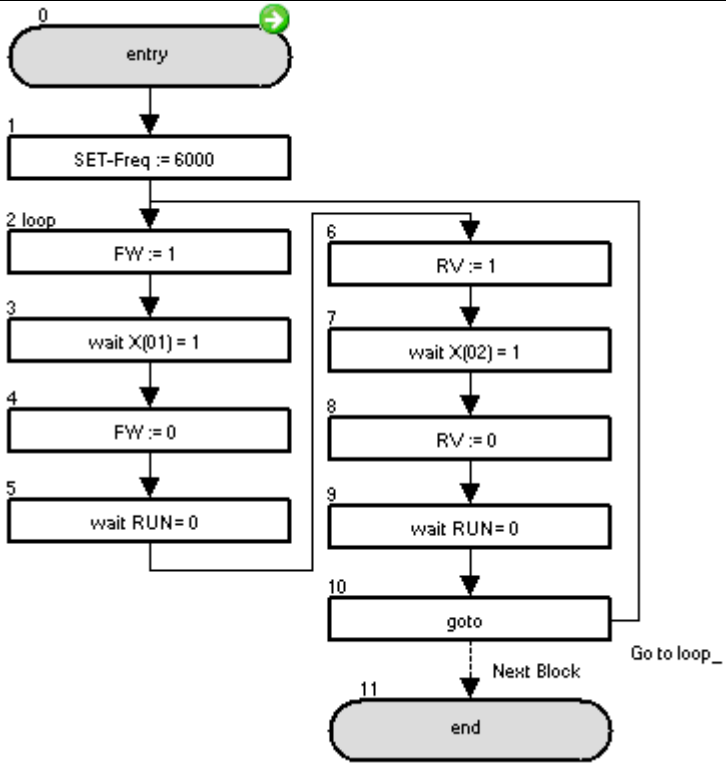
Flowchart	Text
	


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

This example acquires the state of the X(02)-X(05) input terminals and outputs it to Y(00)-Y(03) output terminals.
To cut X(00) – X(01), the U(00) value is divided by 4. To cut X(06) – X(07), the U(00) value is masked by 15.

func = value		
Command	**Description**	**Arguments**
	Assigns the value of a variable to a command of a terminal input.	• **Function**: any function of input terminal. • **Value**: any variable or constant.
Format		
<function> = <value>		

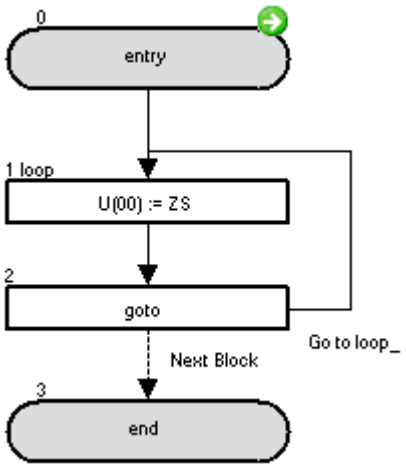
Example

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

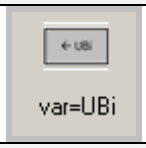
A forward and reverse run at 60Hz is repeated continuously.

Var = func		
Command	**Description**	**Arguments**
	A terminal output status is assigned to a variable.	• **Variable**: any variable. • **Function**: any function of output terminal.
Format		
<variable>=<function>		

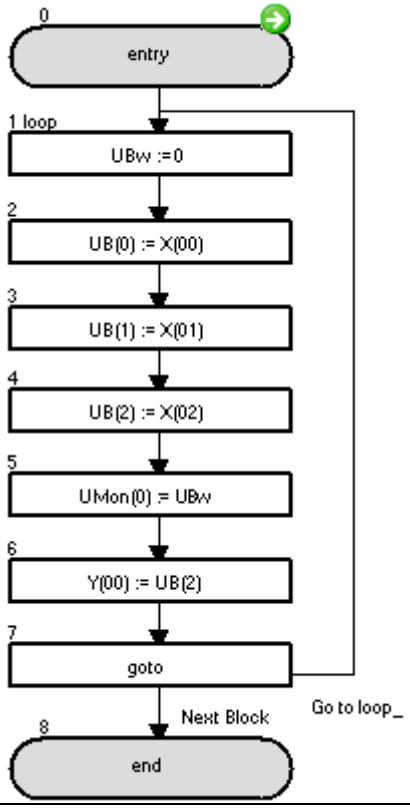
Example

Flowchart	Text
	entry :loop_ U(00) := ZS goto loop_ end

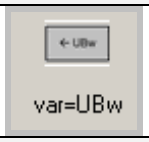
The value of P100 is set to “1” if the ZS (zero speed signal) is on, otherwise is set to “0”.

Var = UB(i)		
Command	**Description**	**Arguments**
	Assigns the value of an internal user contact to a variable.	• **Variable**: any variable (value of the variable will be 0 or 1). • **i**: Number of the user contact (range 0 to 7)
Format		
<variable> = UB(i)		

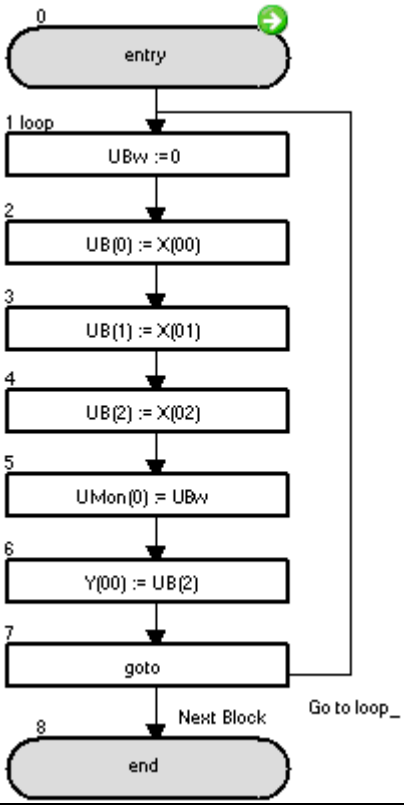
Example

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

The internal user contacts are cleared on the loop's 1st instruction.
The status of the X(00) – X(02) input terminals are stored in the UB(0) – UB(2) internal user contacts and monitored on the d025 parameter.
Finally, the status of the X(02) input terminal is set tot the Y(00) output terminal.

Var = UBw		
Command	**Description**	**Arguments**
	Assigns the value of the internal user contact as word (all together) to a word variable. .	• **Variable**: any variable.
Format		
<variable> = UBw		
Note: UBw = 1 → bit 0 UBw = 2 → bit 1 UBw = 4 → bit 2 UBw = 8 → bit 3 UBw = 16 → bit 4 UBw = 32 → bit 5 UBw = 64 → bit 6 UBw = 128 → bit 7		

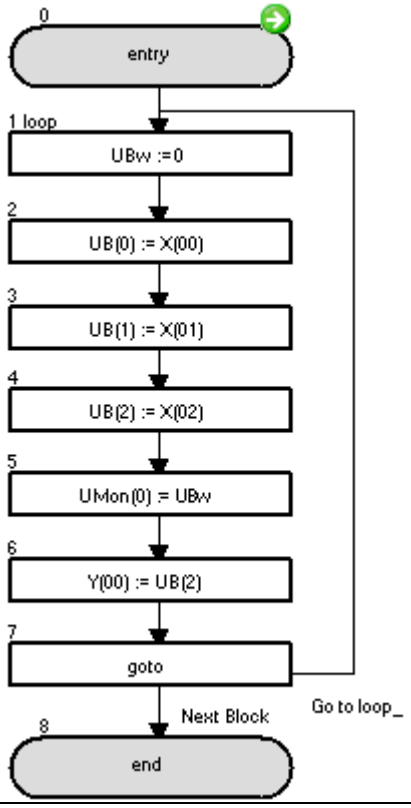
Example

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

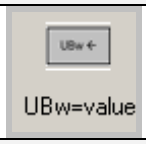
The internal user contacts are cleared on the loop's 1st instruction.
The status of the X(00) – X(02) input terminals are stored in the UB(0) – UB(2) internal user contacts and monitored on the d025 parameter.
Finally the status of the X(02) input terminal is set to the Y(00) output terminal.

UB(i) = value		
Command	**Description**	**Arguments**
	Assigns a value to an internal user contact control.	• **i**: Number of the user contact (range 0 to 7). • **Value**: any variable or constant.
Format		
UB(i) = <value>		

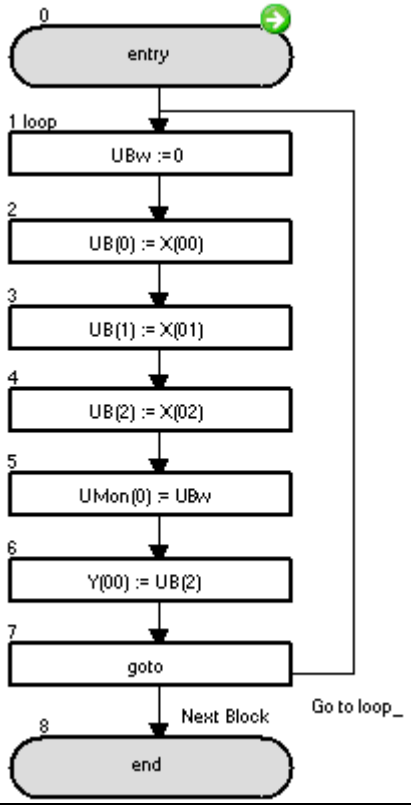
Example

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

The internal user contacts are cleared on the loop's 1st instruction.
The status of the X(00)-X(02) input terminals are stored in the UB(0)-UB(2) internal user contacts and monitored on the d025 parameter. Finally, the status of the X(02) input terminal is set to the Y(00) output terminal.

UBw = value		
Command	**Description**	**Arguments**
	Assigns a value to the internal user contact controls. Instruction to access internal user contact by word.	• **Value**: any variable or constant.
Format		
UBw = <value>		
Note: UBw = 1 → bit 0 UBw = 2 → bit 1 UBw = 4 → bit 2 UBw = 8 → bit 3 UBw = 16 → bit 4 UBw = 32 → bit 5 UBw = 64 → bit 6 UBw = 128→ bit 7		

Example

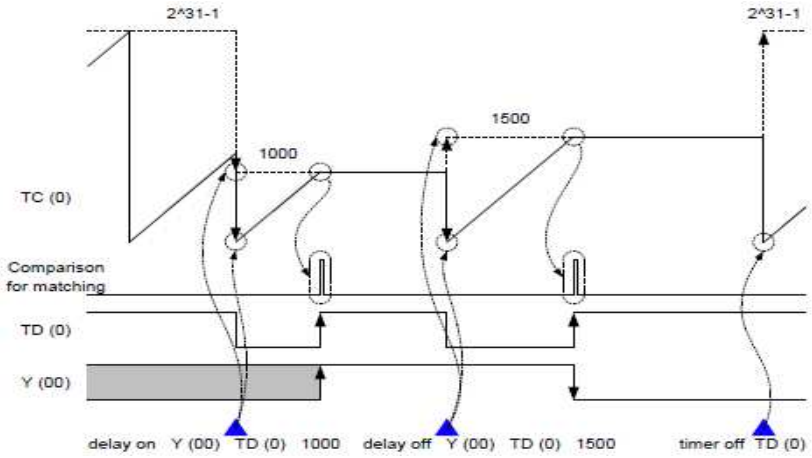
Flowchart	Text
	entry :loop_ ubw := 0 UB(0) := X(00) UB(1) := X(01) UB(2) := X(02) UMon(0) := ubw Y(00) := UB(2) goto loop_ end

The internal user contacts are cleared on the loop's 1st instruction. The status of the X(00)-X(02) input terminals are stored in the UB(0)-UB(2) internal user contacts and monitored on the d025 parameter. Finally, the status of the X(02) input terminal is set to the Y(00) output terminal.

6-4 Timer Control Commands

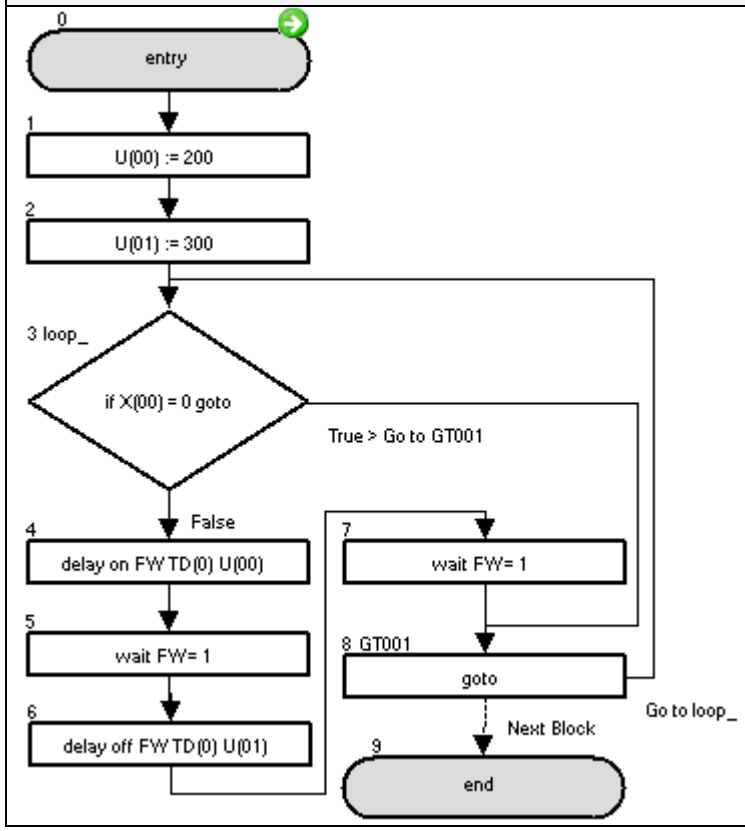
Delay		
Command	**Description**	**Arguments**
	This instruction sets the count of the timer in <value 2> and starts the timer counter. When the timer output "TD (K)" is turned on/off, <value 1> is turned on/off. It is important to note, that meantime counting proceeds, the <value 1> remains unchanged from original value.	• **Value 1**: any variable. • **Value 2**: any variable or constant (time in 10 x ms) • **K**: number of timer.
Format		
Delay on/off <value 1>TD(k)<value 2>		

Timing chart



Example

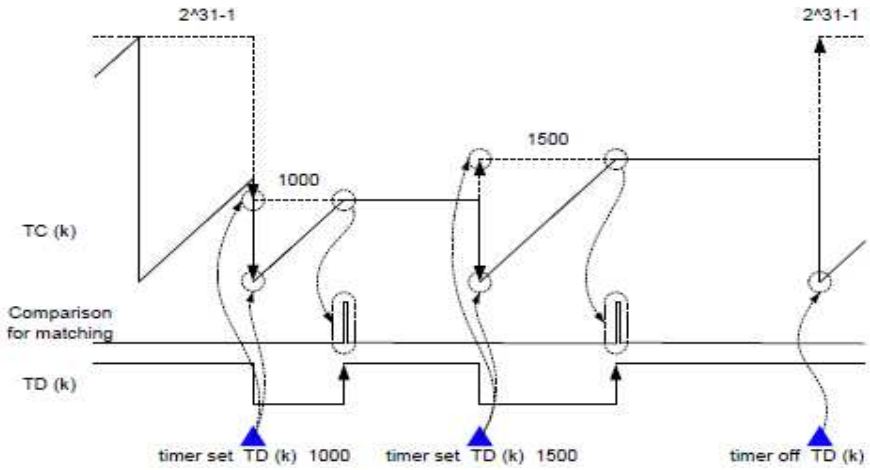
Flowchart	Text

	`entry`
	`  U(00) := 200`
	`  U(01) := 300`
	`:loop_`
	`  if X(00) = 0 goto GT001`
	`  delay on FW TD(0) U(00)`
	`  wait FW = 1`
	`  delay off FW TD(0) U(01)`
	`  wait FW = 1`
	`:GT001`
	`  goto loop_`
	`end`

Sample program that activates/deactivates the FW instruction with Delay On/Delay Off instruction.

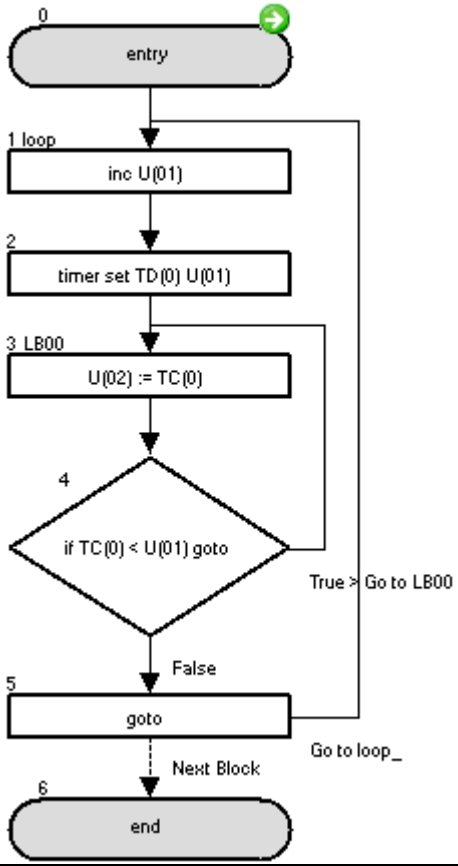
Timer Set		
Command	**Description**	**Arguments**
	Sets <value> in the timer and starts the counter. The timer starts from 0 and increments until <value>. Associated timer contact reflects status ("1" = finish timing)	• **Value**: any variable or constant (time in 10 x ms) • **K**: number of timer (range 0 to 7)
Format		
Timer set TD(k) <value>		
Note: Timer value can be check in variable TC(k). Completion of timer can be checked in variable TD(k) (when it becomes "1").		

Timing chart



Example

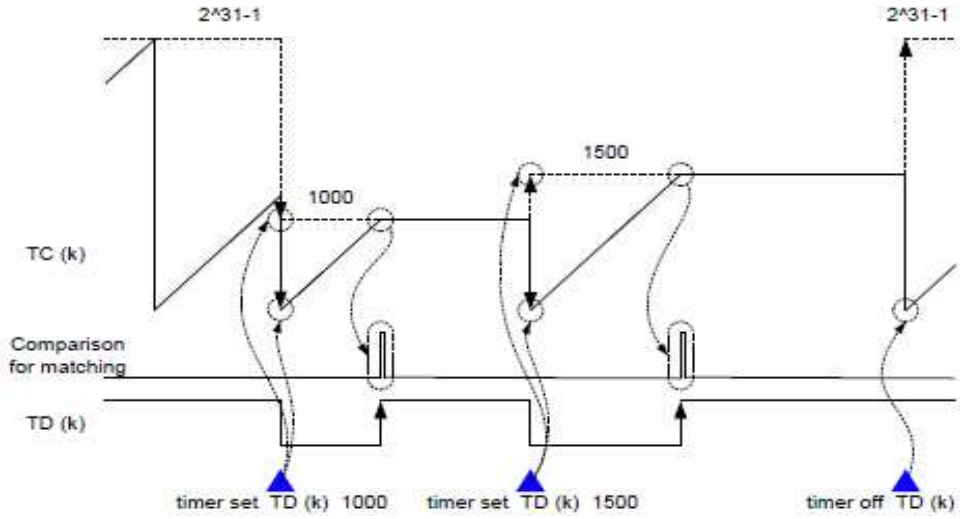
Flowchart	Text

	`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`

This program will set the timer TD(0) to an increasing value each timer execution, taking longer time on each loop.

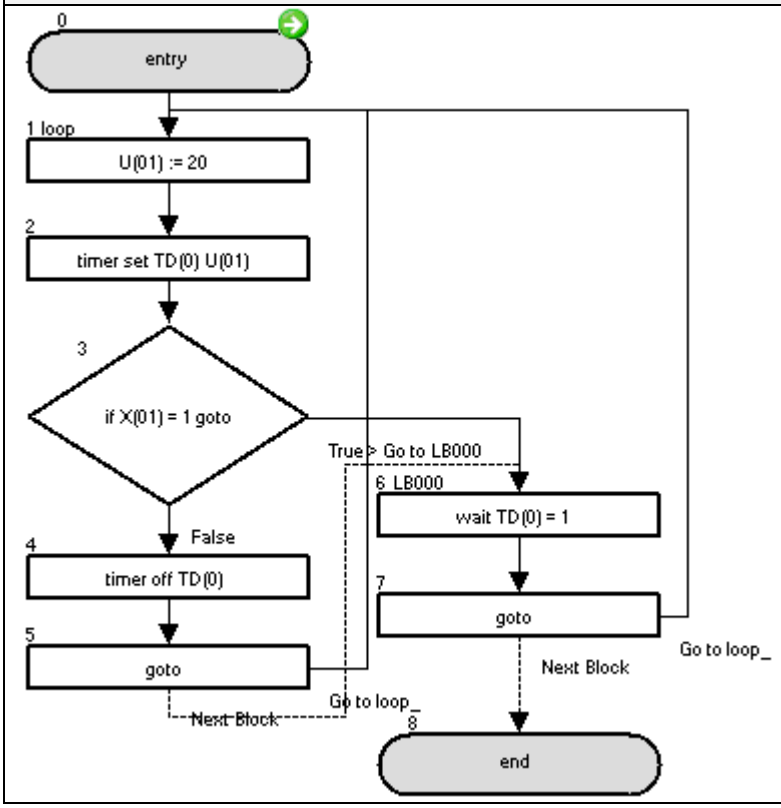
Timer Off		
Command	**Description**	**Arguments**
OFF TimerOff	Clears the timer counter (up counter) to zero, and starts it in free-running mode.	• **k**: number of timer (range 0 to 7)
Format		
Timer off TD(k)		

Timing chart



Example

Flowchart	Text

	`entry`
	`:loop_`
	`  U(01) := 20`
	`  timer set TD(0) U(01)`
	`  if X(01) = 1 goto LB000`
	`  timer off TD(0)`
	`  goto loop_`
	`:LB000`
	`  wait TD(0) = 1`
	`  goto loop_`
	`end`

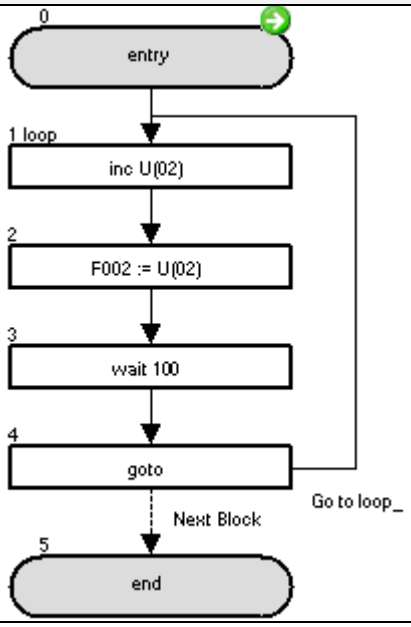
This example uses a fixed timer execution. But it is cancelled when digital input X(01) is OFF.

6-5 Parameter Control Commands

ChgParam		
Command	**Description**	**Arguments**
	Changes the parameter's inverter setting specified by display code to a value. Any inverter parameter can be changed.	• **Parameter**: parameter code (Fxxx, Axxx, bXXX, Cxxx, Hxxx, Pxxx) • **Value**: any variable or constant.
Format		
ChgParam <parameter><value>		
Note: The same rules to parameter writing from operator panel or communications apply: Some parameters can not be written in certain mode of inverter (e.g. some parameters can not be changed during RUN condition). This instruction does not fix the parameter in EEPROM (EepWrt to be used for this purpose)		

Example

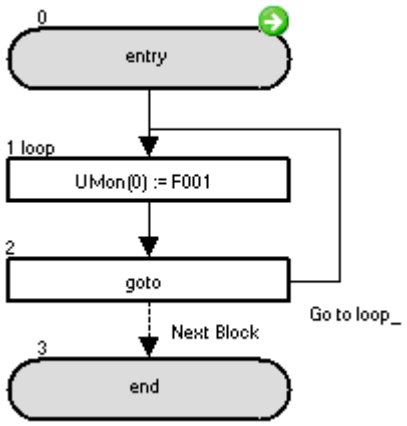
Flowchart	Text

	`entry`
	`:loop_`
	`  inc U(02)`
	`  F002 := U(02)`
	`  wait 100`
	`  goto loop_`
	`end`

The F002 (acceleration time setting 1) value is increased by 1 every second.

MonParam		
Command	**Description**	**Arguments**
	Assigns the inverter's parameter content specified by display code to a variable.	• **Parameter**: parameter code (Fxxx, Axxx, bxxx, Cxxx, dxxx, Hxxx, Pxxx). • **Variable**: any variable
Format		
MonParam<parameter><variable>		

Example

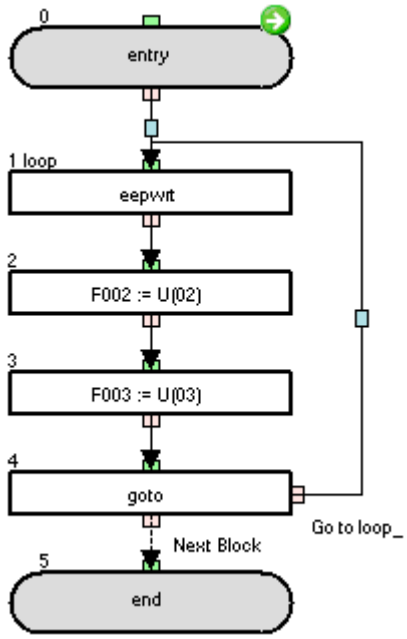
Flowchart	Text
	entry :loop_ UMon(0) := F001 goto loop_ end

The value of the F001 parameter (output frequency setting) is monitored on the d025 parameter (user parameter monitor).

EepWrt		
Command	**Description**	**Arguments**
	The command allows write into EEPROM the next ***ChgParam*** executed just after this command. (if two ChgParam follows an EepWrt, only for the first one will be saved).	---
Format		
EepWrt		
Note: Limitation of ***EepWrt***: -If this command is executed in more than one task, ***ChgParam*** is executed in the sequence it is detected. For the second invocation of the command, a waiting time of typically 10 ms will occur before each ***ChgParam*** is executed. For example, when ***ChgParam*** is detected in task 1,2 and 3 at the same time, and the one in task 1 is executed at first, is necessary to wait 10 ms for task 2 and 20 ms for task 3. But when ***Eepwrt*** is not executed, ***ChgParam*** doesn't need this waiting time.		

Example: (*only F002 is stored in EEPROM*)

Flowchart	Text

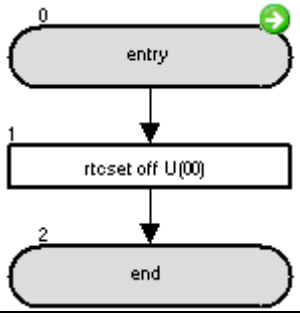
	`entry`
	`:loop_`
	`  eepwrt`
	`  F002 := U(02)`
	`  F003 := U(03)`
	`  goto loop_`
	`end`

On executing the program, only F002 parameter is saved permanently from U(02). After power off and on again, F003 will have the old value. The initial values of the U(02) and U(03) variables can be set on the program variables list or the P102, P103 parameters.

RtcSet		
Command	**Description**	**Arguments**
	This statement sets 6 bytes data of time to a variable. This data corresponds with year, month, day, day of week, hour and minute. The variable value in hexadecimal corresponds to the year, month, day, day of a week, hour and minute (in decimal). **RtcSet on**: updates the 6 bytes data continuously. **RtcSet off**: updates the 6 bytes data only once.	• **User variable**: any user or internal user variable (U(xx) or UL(xx)).
Format		
RtcSet on/off <user variable>		
Note: • **RtcSet on U(<k>)**: It will set U(<k>) with 2 bytes for year and 2 bytes for month, U(<k+1>) with 2 bytes for Month's day and 2 bytes for week's day(00 for Sunday, 06 for Saturday), and U(<k+2>) with 2 bytes for hour and 2 bytes for minutes. • **RtcSet on UL(<k>)**: It will set UL(<k>) with 2 bytes for year, 2 bytes for month, 2 bytes for month's day and 2 bytes for week's day (00 for Sunday, 06 for Saturday), and UL(<k+1>) with 2 bytes for hour, 2 bytes for minutes and 4 bytes of padding(0000). • If the watch LCD operator is not attached, **RtcSet** instruction sets 000000000000h		

Example

Flowchart	Text

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

After executing the program (with the watch LCD operator attached), the hexadecimal value of the first 2 bytes of U(00) will correspond with the current year and the hexadecimal value of the last 2 bytes of U(00) will correspond to the current month.

I.e. if the example program runs on July 5th (Monday) of 2010 at 02:29 P.M., then U(00), U(01) and U(02) will display the following values:

Parameter…	…display in decimal format…	Which converted to hexadecimal format results in…	…which means
U(00)	4103	1007	'10' for 2010 '07' for July
U(01)	1281	0501	'05 for 5th day of month '01' for Monday
U(02)	5161	1429	'14' for 2 p.m. '29' for 29 minutes

6-6 Inverter Control Commands

Run FW		
Command	**Description**	**Arguments**
	Makes the inverter run the motor in forward direction (starts the inverter output). This command is a shortcut of the ***func = value*** command.	---
Format		
FW = 1 for RX and MX2 or UP =1 for LX		
Note: The instruction is available since CX-Drive v2.10.		

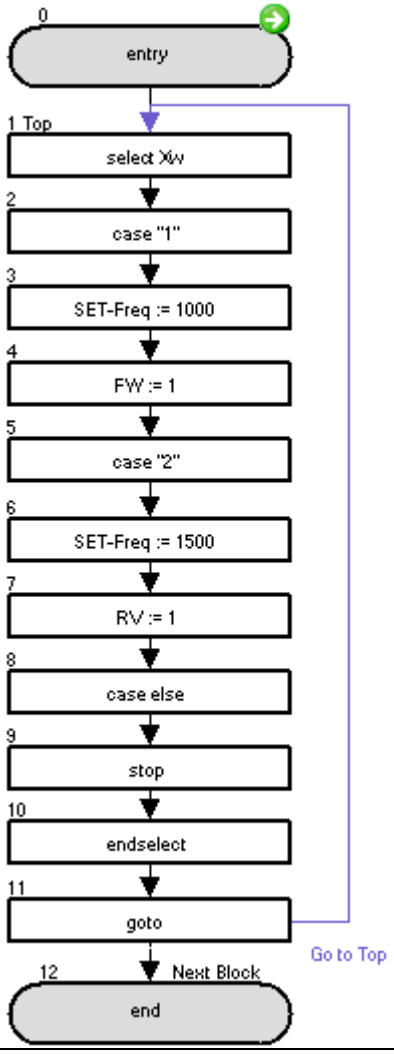
Run RV		
Command	**Description**	**Arguments**
	Makes the inverter run the motor in reverse direction (starts the inverter output). This command is a shortcut of the ***func = value*** command.	---
Format		
RV = 1 for RX and MX2 or DOWN=1 for LX		
Note: This instruction is available since CX-Drive v2.10.		

Stop		
Command	**Description**	**Arguments**
	Makes the inverter decelerate and stop the motor (stop the inverter output).	---
Format		
Stop		

Set Freq		
Command	**Description**	**Arguments**
	It sets the frequency of the inverter. This command is a shortcut of the '=' command. Units: 0.01Hz.	• **Value**: any variable or constant (range from 0 to 40000).
Format		
Set-Freq = <value>		
Note: This instruction is available since CX-Drive v2.10.		

Example

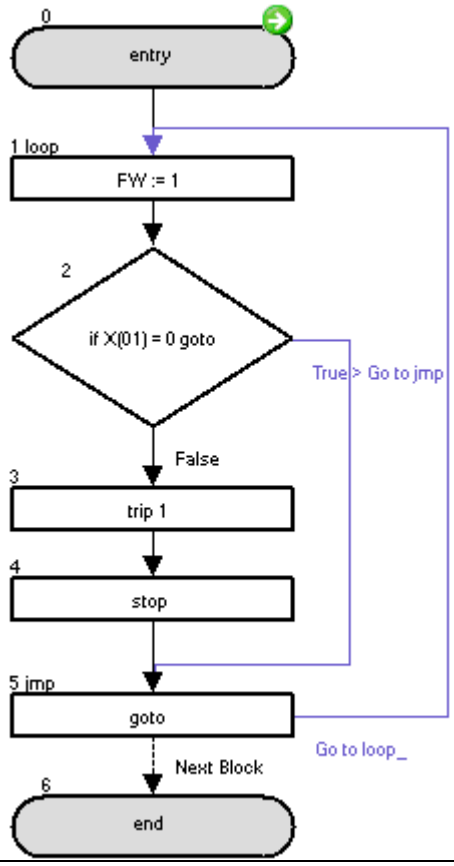
Flowchart	Text

	`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`

This program will run the motor in forward direction at 10Hz if general input contact Xw is 1. If general input contact Xw is 2, it will run in reverse direction at 15Hz. For other values the motor will stop.

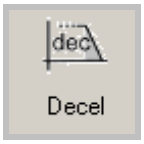
Trip		
Command	**Description**	**Arguments**
	This instruction makes inverter trip.	• **Value**: any variable or constant (range 0 to 9).
Format		
Trip<value>		

Example

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

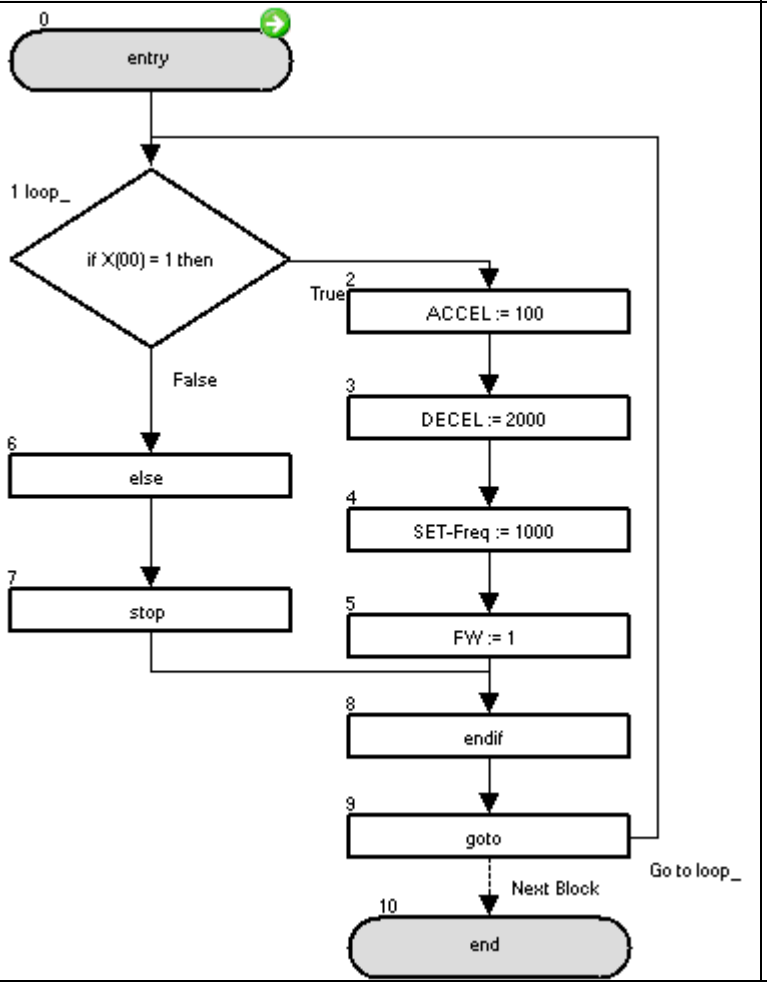
This sample program will throw a user trip on the inverter when digital input X(01) is set to ON.

Accel		
Command	**Description**	**Arguments**
	It sets the acceleration time of the inverter. This command is a shortcut of the '=' command. Units: 10 ms.	• **Value**: any variable or constant (range from 1 to 360000).
Format		
Accel =<value>		
Note: Parameters P031 for MX2 and RX or A053 on LX must be set to value 3 (Drive programming) for the command to become effective.		

Decel		
Command	**Description**	**Arguments**
	It sets the deceleration time of the inverter. This command is a shortcut of the '=' command. Units: 10ms	• **Value**: any variable or constant (range from 1 to 360000).
Format		
Decel = <value>		
Note: Parameters P031 for MX2 and RX or A053 on LX must be set to value 3 (Drive programming) for the command to become effective		

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`

This sample program will set the Acceleration to 10 seconds and deceleration to 20 seconds if digital input X(00) is set to ON.

7- Drive Programming specific trips and Troubleshooting

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

Factor code	Error (causing inverter trip)	Possible cause	Checking method	Corrective action
E43	Invalid instruction	The PRG terminal was turned on without a program downloaded to the inverter.	By uploading the program, you can check if really a program is in the inverter or not.	Recreate the program, and then download it to the inverter
E44	Nesting count error	Subroutines are nested in more than eight layers.	Read the program to check the number of nesting layers (some times difficult to recognize)…	Correct the program so that the number of layers will be eight or less.
		For-Next loop statements are nested in more than eight layers.		
		If statements are nested in more than eight layers.		
E45	Instruction error 1	The jump destination of a GoTo instruction is a next instruction to end a for or other loop.	Check whether each GoTo instruction jumps to an instruction that ends a loop.	Correct the jump destinations of GoTo instructions. As general recommendation, never jump a Goto out of the current level it is.
		The variable “U(ii)” referenced via another variable is not found.	Check the numerical value specified in “U(ii)”.	Correct the value of variable “U(ii)” or limit the range of values of variable “U(ii)”.
		An arithmetic instruction caused: -Overflow, -underflow, or -division by zero	Check the program for the instruction causing overflow, underflow, or division by zero (not in early MX2 firmware).	Correct the program so that no arithmetic instruction causes overflow, underflow, or division by zero.
		A ChgParam instruction caused: - reference to a non existing parameter. -writing of a value out of the setting range -change of a parameter value (during inverter operation) that cannot be updated during inverter operation, or Change of a parameter value of which updating is restricted by software lock (when software lock is enabled).	-Check the parameters and the values to be written. -If the error has occurred during inverter operation, check whether the parameter in question can be updated during inverter operation. -Check the setting of software lock selection (b031).	-Correct the parameters or the values to be written to parameters so that they will be within the setting range. -Disable software lock. -If the parameter to be updated is the one that cannot be updated during inverter operation, change the setting of software lock selection (b031) to “10” to switch to the mode enabling parameter updating during inverter operation.
E50 to E59	User trip 0 to 9	These trips are generated from the user application. The cause is determined by the Drive Programming logic	Check with the drive program documentation to recognize the trip conditions	Check the drive program documentation to recognize countermeasures

8- Drive Programming Parameters – General Precautions

8-1 Parameters list affected by setting order

Parameter	Description
A003	Base frequency setting
A004	Maximum frequency setting
A203	Base frequency setting, 2nd motor
A204	Maximum frequency setting, 2nd motor
B015	Free setting, electronic thermal frequency (1)
B017	Free setting, electronic thermal frequency (2)
B019	Free setting, electronic thermal frequency (3)
B049	Dual Rating Selection
B050	Controlled deceleration on power loss
B051	DC bus voltage trigger level of control deceleration
B052	Over-voltage threshold of control deceleration
B060	Maximum-limit level of window comparators O
B061	Minimum-limit level of window comparators O
B062	Hysteresis width of windows comparators O
B063	Maximum-limit level of window comparators OI
B064	Minimum-limit level of window comparators OI
B065	Hysteresis width of window comparator (OI)
B079	Watt-hour display gain setting
B082	Start frequency adjustment
B100	Free setting V/f freq. (1)
B102	Free setting V/f freq. (2)
B104	Free setting V/f freq. (3)
B106	Free setting V/f freq. (4)
B108	Free setting V/f freq. (5)
B110	Free setting V/f freq. (6)
B112	Free setting V/f freq. (7)
P070	Low-speed zero-return frequency

Note: this parameter list only affect MX2 and RX.

8-2 Parameters list affected by Rated Current (%)

Parameter	Description
B012	Level of electronic thermal setting
B016	Free setting, electronic thermal current (1)
B018	Free setting, electronic thermal current (2)
B020	Free setting, electronic thermal current (3)
B022	Overload restriction level setting
B025	Overload restriction level 2 setting
B028	Current level of active freq. matching restart setting
B126	Brake release current setting
B212	Level of electronic thermal setting, 2nd motor
B222	Overload restriction operation mode, 2nd motor
C030	Digital current monitor reference value
C039	Low load detection level
C041	Overload level setting
C111	Overload setting (2)
C241	Overload level setting, 2nd motor

Note: this parameter list affect MX2. RX and LX.

8-3 Parameters list affected by PID enabled/disabled

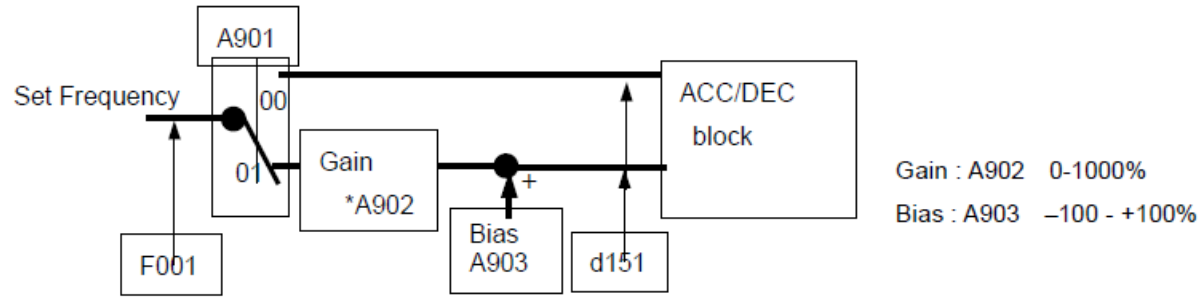
Parameter	Description
A011	Pot./O-L input active range start frequency
A012	Pot./O-L input active range end frequency
A020	Multi-speed 0 setting
A021	Multi-speed 1 setting
A022	Multi-speed 2 setting
A023	Multi-speed 3 setting
A024	Multi-speed 4 setting
A025	Multi-speed 5 setting
A026	Multi-speed 6 setting
A027	Multi-speed 7 setting
A028	Multi-speed 8 setting
A029	Multi-speed 9 setting
A030	Multi-speed 10 setting
A031	Multi-speed 11 setting
A032	Multi-speed 12 setting
A033	Multi-speed 13 setting
A034	Multi-speed 14 setting
A035	Multi-speed 15 setting
A101	[OI] input active Range start frequency
A102	[OI] input active Range end frequency
A145	ADD frequency
A220	Multi-speed 0 setting, 2nd motor
F001	Output frequency setting

These parameters are affected by A071 / A075.
Note: this parameter list only affect MX2 and RX.

9 Insertion Point (MX2 & RX)

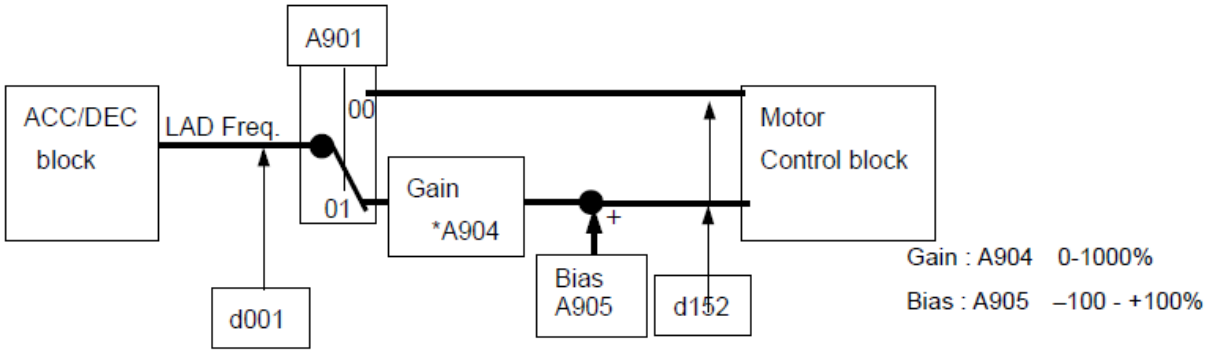
The Gain/Bias can be applied to any reference

9-1 'Frequency' before ACC/DEC



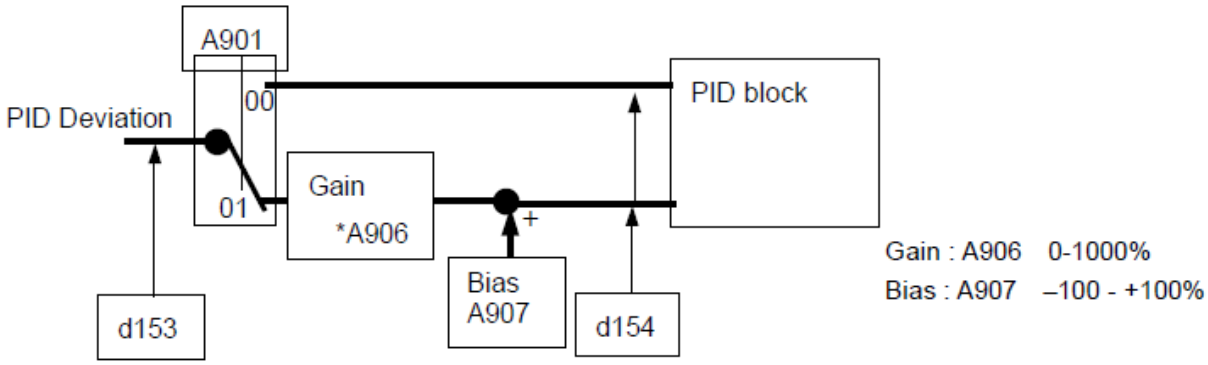
Parameter	Description	Range
A901	Insertion Point	0: Disable; 1: Enable
A902	Insertion Point 'Frequency' before ACC/Dec Gain	0 to 1000%
A903	Insertion Point 'Frequency' before ACC/DEC Bias	-100 to +100%

9-2 'Frequency' after ACC/DEC



Parameter	Description	Range
A901	Insertion Point	0: Disable; 1: Enable
A904	Insertion Point 'Frequency' after ACC/Dec Gain	0 to 1000%
A905	Insertion Point 'Frequency' after ACC/DEC Bias	-100 to +100%

9-3 'Deviation' before PID block



Parameter	Description	Range
A901	Insertion Point	0: Disable; 1: Enable
A906	Insertion Point 'Deviation' before PID block Gain	0 to 1000%
A907	Insertion Point 'Deviation' before PID block Bias	-100 to +100%

OMRON EUROPE B.V. Wegalaan 67-69, NL-2132 JD, Hoofddorp, The Netherlands.
Tel: +31 (0) 23 568 13 00 Fax: +31 (0) 23 568 13 88 www.industrial.omron.eu

Austria
Tel: +43 (0) 2236 377 800
www.industrial.omron.at

Belgium
Tel: +32 (0) 2 466 24 80
www.industrial.omron.be

Czech Republic
Tel: +420 234 602 602
www.industrial.omron.cz

Denmark
Tel: +45 43 44 00 11
www.industrial.omron.dk

Finland
Tel: +358 (0) 207 464 200
www.industrial.omron.fi

France
Tel: +33 (0) 1 56 63 70 00
www.industrial.omron.fr

Germany
Tel: +49 (0) 2173 680 00
www.industrial.omron.de

Hungary
Tel: +36 (0) 1 399 30 50
www.industrial.omron.hu

Italy
Tel: +39 02 32 681
www.industrial.omron.it

Middle East & Africa
Tel: +31 (0) 23 568 11 00
www.industrial.omron.eu

Netherlands
Tel: +31 (0) 23 568 11 00
www.industrial.omron.nl

Norway
Tel: +47 (0) 22 65 75 00
www.industrial.omron.no

Poland
Tel: +48 (0) 22 645 78 60
www.industrial.omron.com.pl

Portugal
Tel: +351 21 942 94 00
www.industrial.omron.pt

Russia
Tel: +7 495 648 94 50
www.industrial.omron.ru

Spain
Tel: +34 913 777 900
www.industrial.omron.es

Sweden
Tel: +46 (0) 8 632 35 00
www.industrial.omron.se

Switzerland
Tel: +41 41 748 13 13
www.industrial.omron.ch

Turkey
Tel: +90 (0) 216 474 00 40
www.industrial.omron.com.tr

United Kingdom
Tel: +44 (0) 870 752 08 61
www.industrial.omron.co.uk

Note: Specifications subject to change without notice.
Cat. No. I170E-EN-01B

OMRON