DELTA ELECTRONICS, INC.

www.delta.com.tw/industrialautomation

ASIA

Delta Electronics, Inc.
Taoyuan1
31-1, Xingbang Road, Guishan Industrial Zone,
Taoyuan County 33370, Taiwan, R.O.C.
TEL: 886-3-362-6301 / FAX: 886-3-362-7267

Delta Electronics (Jiang Su) Ltd.
Wujiang Plant3
1688 Jiangxing East Road,
Wujiang Economy Development Zone,
Wujiang City, Jiang Su Province,
People's Republic of China (Post code: 215200)
TEL: 86-512-6340-3008 / FAX: 86-512-6340-7290

Delta Electronics (Japan), Inc.
Tokyo Office
Delta Shibadaimon Building, 2-1-14 Shibadaimon,
Minato-Ku, Tokyo, 105-0012, Japan
TEL: 81-3-5733-1111 / FAX: 81-3-5733-1211

Delta Electronics (Korea), Inc.
Donghwa B/D 3F, 235-6, Nonhyun-dong,
Kangnam-gu, Seoul 135-010, Korea
TEL: 82-2-515-5303/5 / FAX: 82-2-515-5302

Delta Electronics (Singapore) Pte. Ltd.
8 Kaki Bukit Road 2, #04-18 Ruby Warehouse Complex,
Singapore 417841
TEL: 65-747-5155 / FAX: 65-744-9228

Delta Energy Systems (India) Pvt. Ltd.
Plot No. 27 & 31, Sector-34, EHTP,
Gurgaon-122001 Haryana, India
TEL: 91-124-4169040 / FAX: 91-124-4036045

AMERICA

Delta Products Corporation (USA)
Raleigh Office
P.O. Box 12173,5101 Davis Drive,
Research Triangle Park, NC 27709, U.S.A.
TEL: 1-919-767-3813 / FAX: 1-919-767-3939

EUROPE

Deltronics (The Netherlands) B.V.
Eindhoven Office
De Witbogt 15, 5652 AG Eindhoven, The Netherlands
TEL: 31-40-2592850 / FAX: 31-40-2592851

*We reserve the right to change the information in this manual without prior notice

20080310

DELTA DOP Series HMI-WPLSoft Instruction Manual

DELTA ELECTRONICS, INC.

DOP Series

HMI-WPLSoft Instruction Manual

www.delta.com.tw/industrialautomation

Table of Contents

This page intentionally left blank.

Chapter 1 Getting Started

Delta Extension Digital I/O Module, DOP-EXIO14RAE and DOP-EXIO28RAE (hereinafter called “DOP-EXIO series”) provided for DOP-AE series HMI only. Therefore, before using Delta Extension Digital I/O Module, the user has to open the ScrEdit (Screen Editor) programming software, click “File” > “New” to open a new project, and select the type of DOP-AE series HMI being used (see Fig. 1.1).

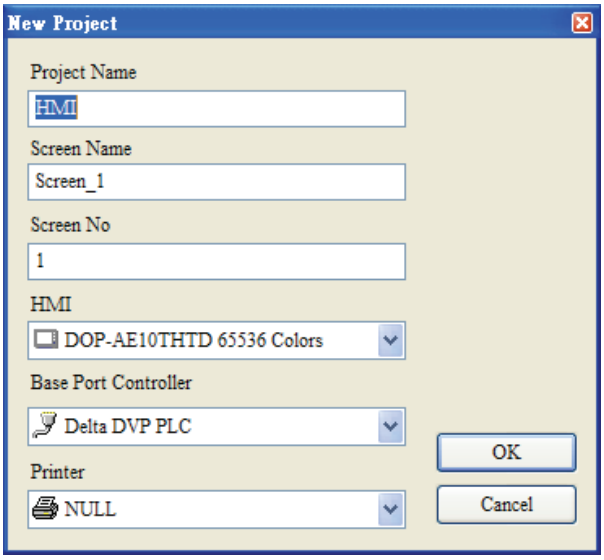


Fig. 1.1 New project dialog box

After selecting the type of DOP-AE series HMI, press OK button to complet the setting. Then, click “Options > “Configuration” to enter into “Other” tab in Configuration option (see Fig. 1.2). Check the box next to “Enable EXIO (Compile Ladder)” to activate the function of Delta Extension Digital I/O Module. The user can also select the digital input and output points here by using the drop down list right below the “Enable EXIO (Compile Ladder)” option (see Fig. 1.3).

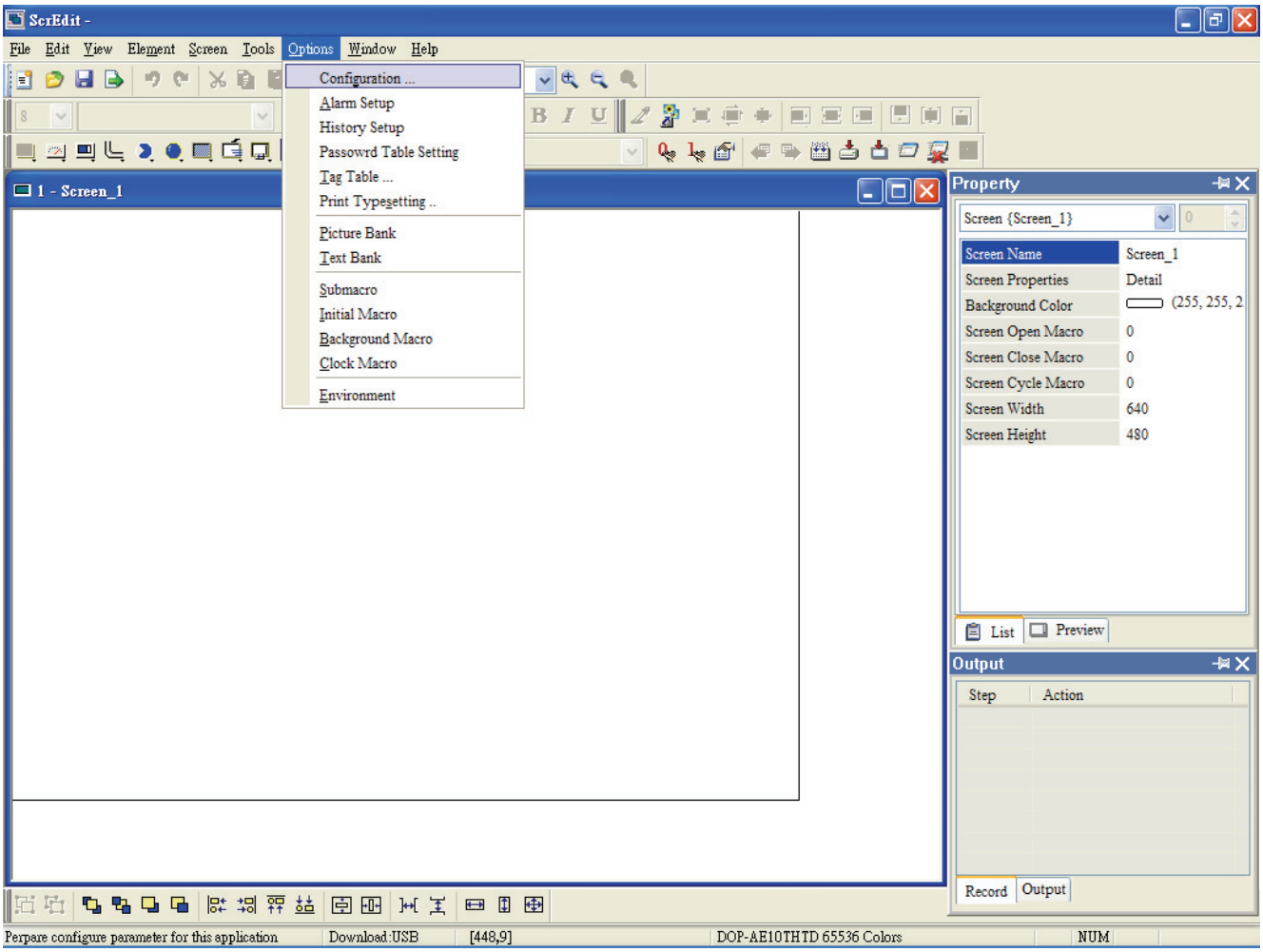


Fig. 1.2 Configuration option

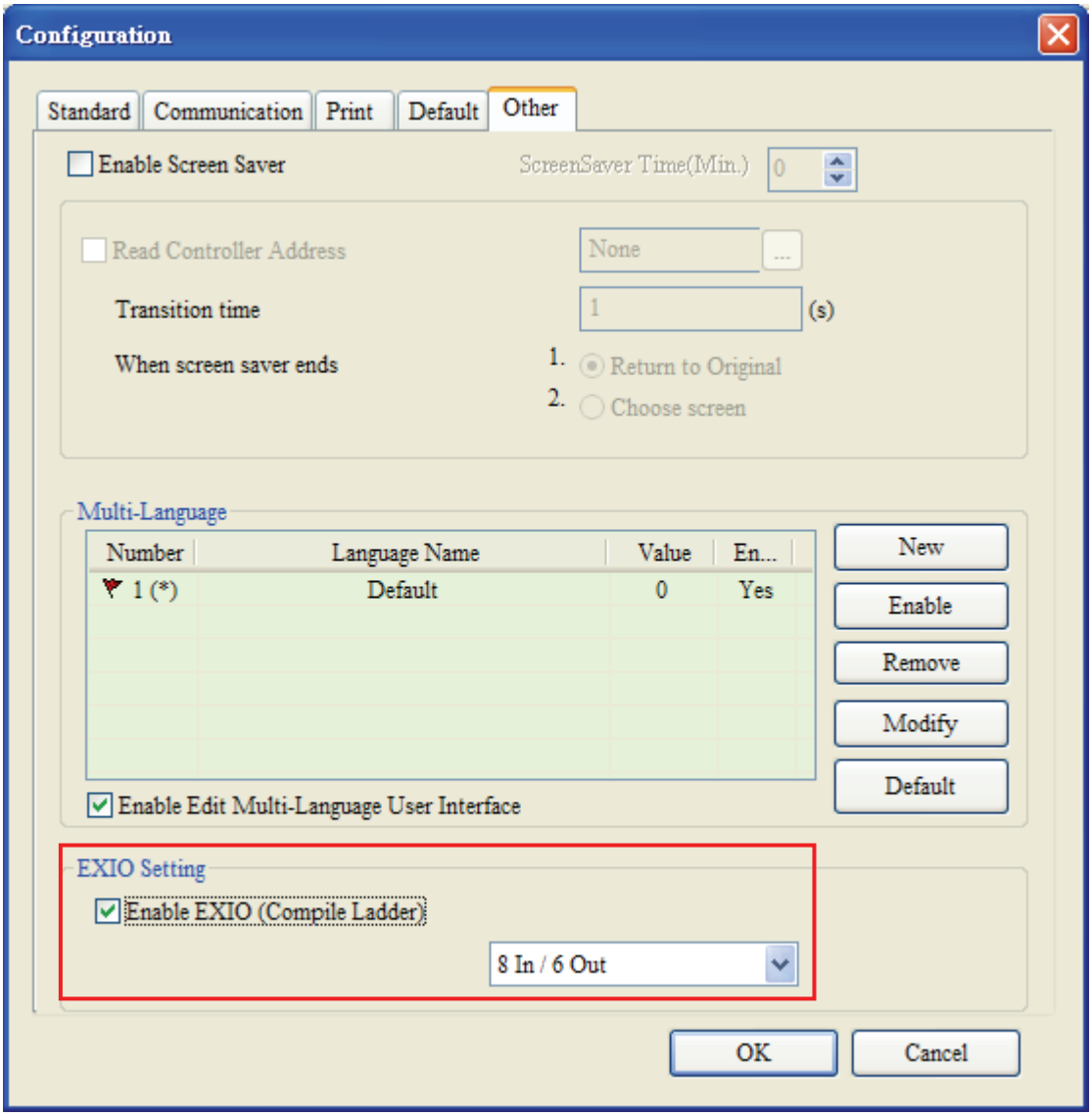


Fig. 1.3 Other tab

When “Enable EXIO (Compile Ladder)” option is selected, the “Ladder Editor” icon will appear and be available for use on the toolbar (See Fig. 1.4 and 1.5). The user can click this icon and start ladder diagram editing directly or click “Tool” > “Ladder Editor” command from the menu (See Fig. 1.6).

Fig. 1.4 Toolbar before “Enable EXIO (Compile Ladder)” option is selected

Fig. 1.5 Toolbar after “Enable EXIO (Compile Ladder)” option is selected

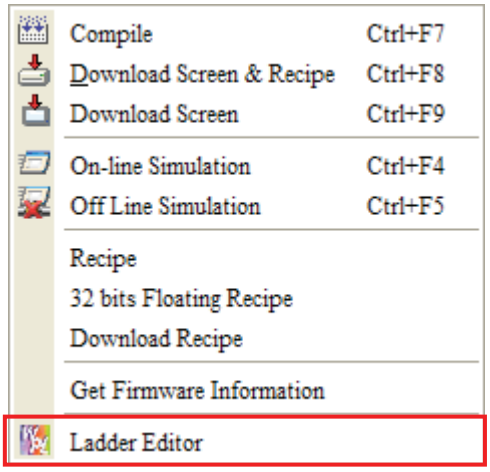


Fig. 1.6 Ladder Editor option

Chapter 2 HMI-WPLSoft Introduction

Clicking the “Ladder Editor”  icon can open HMI-WPLSoft editing window immediately (see Fig. 2.1). At the same time, the window of ScrEdit (Screen Editor) will zoom out and hide automatically. Please note that HMI-WPLSoft and ScrEdit programming software cannot be used simultaneously. When the user is editing a ladder diagram and in the meantime the user wants to edit a HMI program, the user must close the window of HMI-WPLSoft and then it is possible for the user to edit a HMI program in the environment of ScrEdit programming software successfully. There is no Open and Save option provided in the ladder diagrm editing window. When the ladder diagrm editing window is closed, the ladder diagrm editing program is saved automatically.

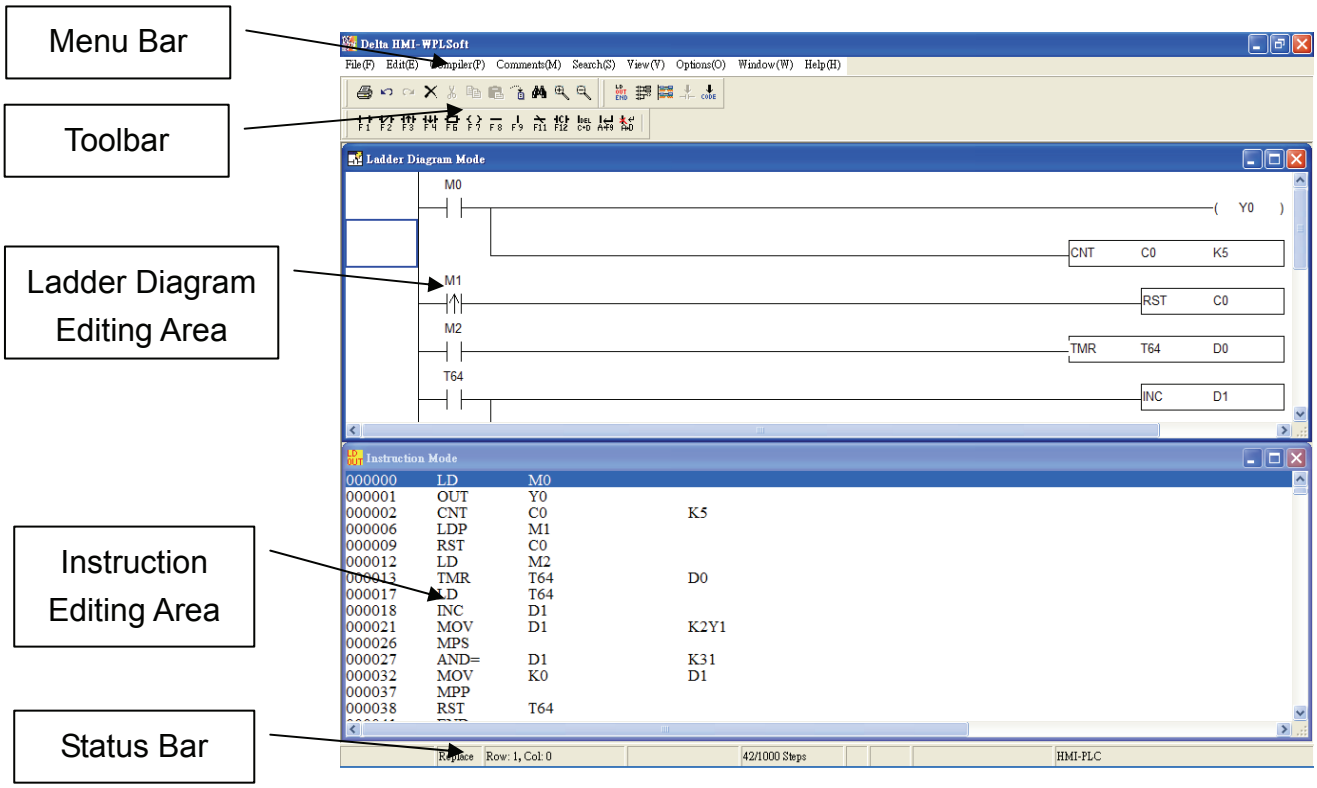
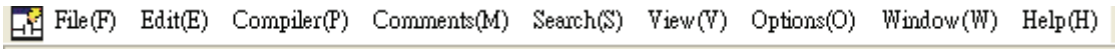


Fig. 2.1 HMI-WPLSoft editing window

There are five parts in the following for the window of HMI-WPLSoft.

- **Menu bar**



There are nine functions for selection: File(F), Edit(E), Compiler(P), Comments(M), Search(S), View(V), Options(O), Window(W), and Help(H). Each option has a pull-down menu.

■ **Toolbar**

There are many icons provided for the user to execute functions by clicking the mouse directly. The followings are the available toolbar on HMI-WPLSoft.

1. Standard Toolbar

2. HMI-WPLSoft Toolbar

3. Ladder diagram Toolbar (display in Ladder Diagram Mode only)

■ **Ladder Diagram Editing Area**

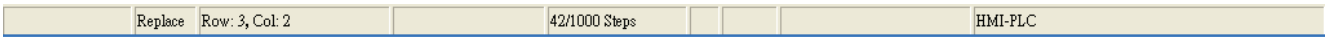
This is the area for designing the editing the ladder diagram by requirement.

■ **Instruction Editing Area**

This is the area for designing the editing the instructions by requirement.

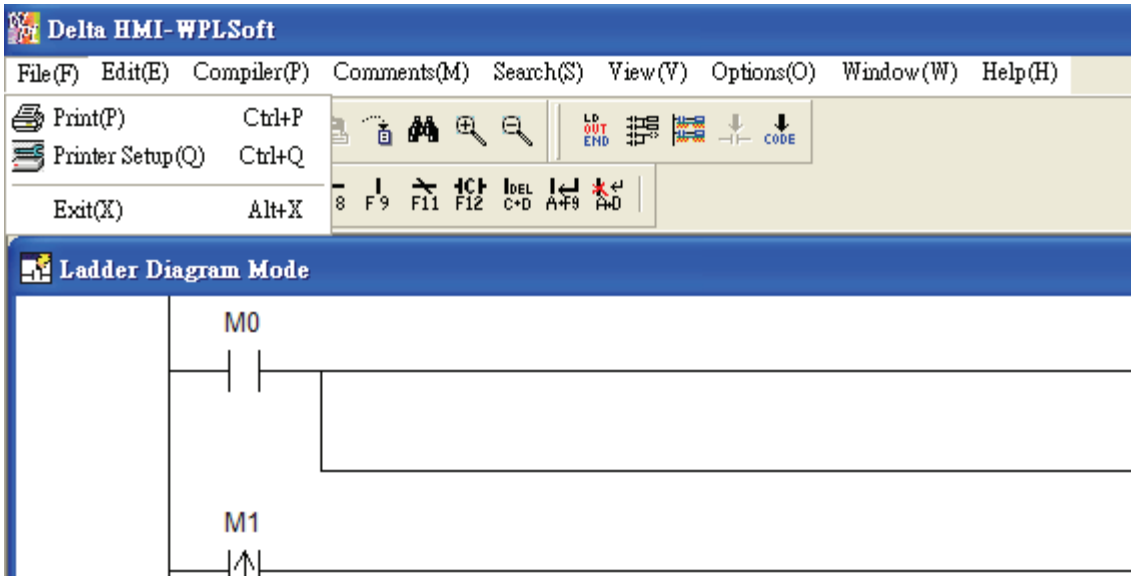
■ **Status Bar**

It is used to display messages, including replace/insert mode, the coordinate of the editing diagram or object, etc.



2.1 File

The “File” function is shown as follows, including pull-down menu options:



■ Print(P) ⇨ Print current file (only print current window, i.e. one of ladder diagram or instruction mode).

◈ Method 1: Click “File(F)” > “Print (P)”.

◈ Method 2: Click the icon  from the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (P).

After the editing is completed, the user can use the icon  or click “File” > “Print (P)” to print the editing program or instrucation and relevant data. In the different editing window, the user can use Print(P) function to print the ladder diagram or instruction data. Please refer to the following descriptions.

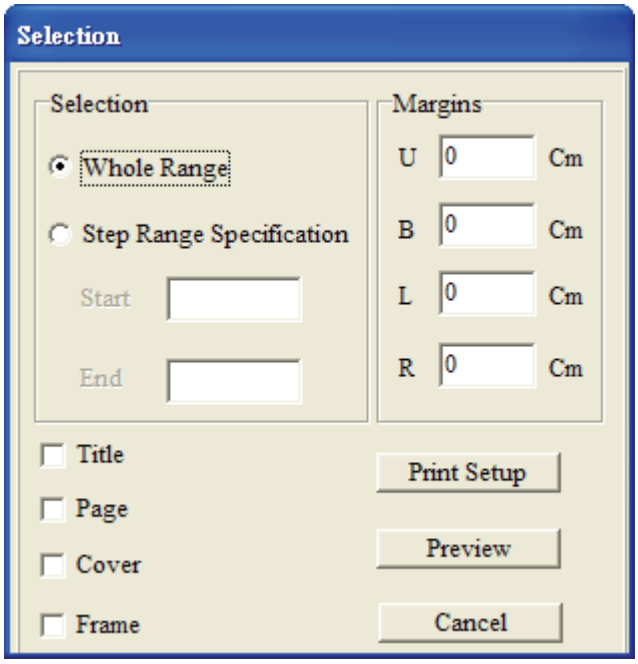
■ Print Ladder Diagram

In Ladder Diagram Mode (when the ladder diagram editing window is opened), click the icon  on toolbar or choose “Print(P)” command from the “File” menu, the print selection dialog box will open allowing the user to set the print options, configure printed diagrams layout and print the ladder diagrams shown on the screen. When the print selection dialog box is opened, the user can choose “Whole Range” to print all ladder diagrams displayed on the screen or choose “Step Range Specification” to print the range specified by the user (Start and End). Also, the user can determine if the title, page numbers and cover are printed or not. Click “Preview” button is to show the ladder diagrams as they would look if printed. Click “Printer setup” button is to setup the printer and configure the layout of the printed ladder diagrams.

The ladder diagrams displayed in the ladder diagram editing window is the same as the printed file. It indicates that the comments will be printed also if there are comments displayed on the ladder diagrams.

■ Print Instruction

In Instruction Mode (when the instruction editing window is opened), click the icon  on toolbar or choose “Print(P)” command from the “File” menu, the print selection dialog box will open allowing the user to set the print options, configure printed instruction layout and print the instructions. When the print selection dialog box is opened, the user can choose “Whole Range” to print all instructions displayed on the screen or choose “Step Range Specification” to print the range specified by the user (Start and End). Also, the user can determine if the title, page numbers and cover are printed or not. Click “Preview” button is to show the instructions as they would look if printed. Click “Printer setup” button is to setup the printer and configure the layout of the printed instructions.



- Printer Setup(Q) ⇨ Select and set printer.
  - ◈ Method 1: Click “File” > “Printer Setup(Q)”.
  - ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Q).
- Exit(X) ⇨ End HMI-WPLSoft
  - ◈ Method 1: Click “File(F)” > “Exit(X)”.
  - ◈ Method 2: Click the icon ✕ at the right upper corner of the window.
  - ◈ Method 3: Use keyboard shortcuts by pressing keys (Alt) + (X).

File Explanation:

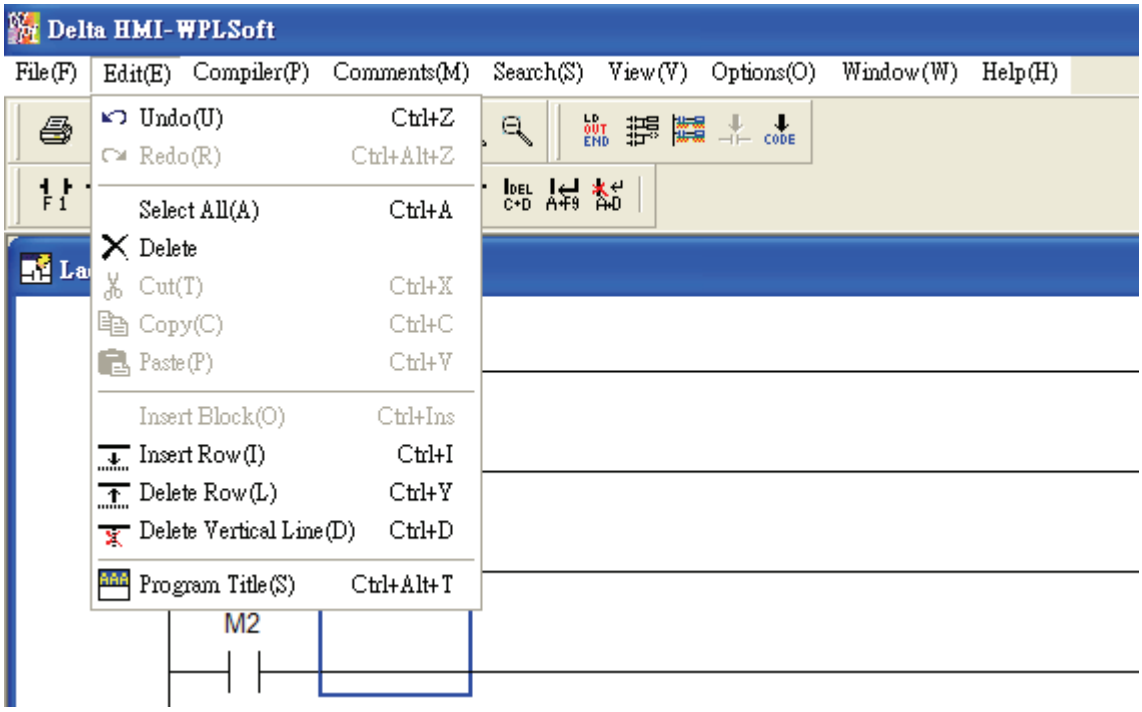
There are six saved files which each one of them has different extension names created simutaneously after finishing program editing and compiler. If the user wants to copy a complete program (including all comments and settings in the program) to other disk or another directory, it is recommended to copy all six saved files with different extension names. If the user wants to make a complete backup copy of the program file, the following six different files should be saved all together.

	Extension Name		Explanation
1	* .**DLP**	⇨	The instruction file for DOP-EXIO series.
2	* .**LAD**	⇨	Ladder diagram file
3	* .**LMT**	⇨	The file used to record ladder diagram segment comments.
4	* .**LAB**	⇨	The file used to record label P and I.

	Extension Name		Explanation
5	* .**RCM**	⇨	The default comment file for special D/special M.
6	* .**DOP***	⇨	HMI ScrEdit (Screen Editor) file.

2.2 Edit

The “Exit” function is shown as follows, including pull-down menu options:



- Undo(U) ⇨ Undo the most recent actions (the system allows the user to perform undo action for max. 10 times)

  ◈ Method 1: Click “Edit(E)” > “Undo(U)”.

  ◈ Method 2: Click the icon on the toolbar.

  ◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (Z).

  ◈ Method 4: Right click the mouse to get a pop-up menu and select “Undo” in the pop-up menu.

- Redo(R) ⇨ Redo the undo action.

  ◈ Method 1: Click “Edit(E)” > “Redo(R)”.

  ◈ Method 2: Click the icon on the toolbar.

  ◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (Alt) + (Z).

◈ Method 4: Right click the mouse to get a pop-up menu and select “Redo” in the pop-up menu.

■ Select All(A) ⇨ Select everything in a program file.

◈ Method 1: Click “Edit(E)” > “Select All (A)”.

◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (A).

■ Delete ⇨ Delete a selection (selected block or data) where the cursor is.

◈ Method 1: Click “Edit(E)” > “Delete”.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing key (Delete).

◈ Method 4: Right click the mouse to get a pop-up menu and select “Delete” in the pop-up menu.

■ Cut(T) ⇨ Cut a selection (selected block or data) in a program file.

◈ Method 1: Click “Edit(E)” > “Cut(T)”.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (X).

◈ Method 4: Right click the mouse to get a pop-up menu and select “Cut” in the pop-up menu.

■ Copy(C) ⇨ Copy a selection (selected block or data) from a program file.

◈ Method 1: Click “Edit(E)” > “Copy(C)”.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (C).

◈ Method 4: Right click the mouse to get a pop-up menu and select “Copy” in the pop-up menu.

■ Paste(P) ⇨ Paste a selection (selected block or data) on a program file.

◈ Method 1: Click “Edit(E)” > “Paste(P)”.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (V).

◈ Method 4: Right click the mouse to get a pop-up menu and select “Paste” in the pop-up menu.

- Insert Block(O) ⇨ Insert a selection (selected block or data) into a program file (This function is valid for Ladder Diagram Mode only.).

  ◈ Method 1: Click “Edit(E)” > “Insert Bock(O)”.

  ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Ins).

  ◈ Method 3: Right click the mouse to get a pop-up menu and select “Insert Block” in the pop-up menu.

- Insert Row(I) ⇨ Insert a blank row into a program file.

  ◈ Method 1: Click “Edit(E)” > “Insert Row(I)”.

  ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (I).

  ◈ Method 3: Right click the mouse to get a pop-up menu and select “Insert Row” in the pop-up menu.

- Delete Row(L) ⇨ Delete a row from a program file.

  ◈ Method 1: Click “Edit(E)” > “Delete Row(L)”.

  ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Y).

  ◈ Method 3: Right click the mouse to get a pop-up menu and select “Delete Row” in the pop-up menu.

- Delete Vertical Line(D) ⇨ Delete the vertical lines from a program file(This function is valid for Ladder Diagram Mode only.).

  ◈ Method 1: Click “Edit(E)” > “Delete Vertical Line(D)”.

  ◈ Method 2: Click the icon  on the toolbar.

  ◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (D).

  ◈ Method 4: Right click the mouse to get a pop-up menu and select “Delete Vertical Line” in the pop-up menu.

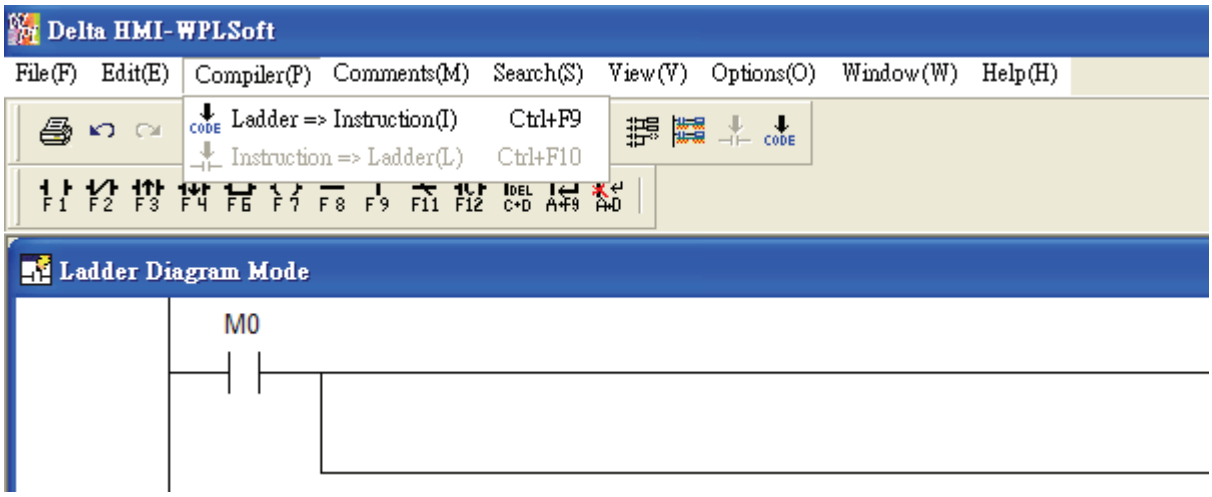
- Program Title(S) ⇨ The information of program title, file name, company name and designer are shown here and can be printed as an easy cover.

  ◈ Method 1: Click “Edit(E)” > “Program Title(S)”.

  ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Alt) + (T).

2.3 Compiler

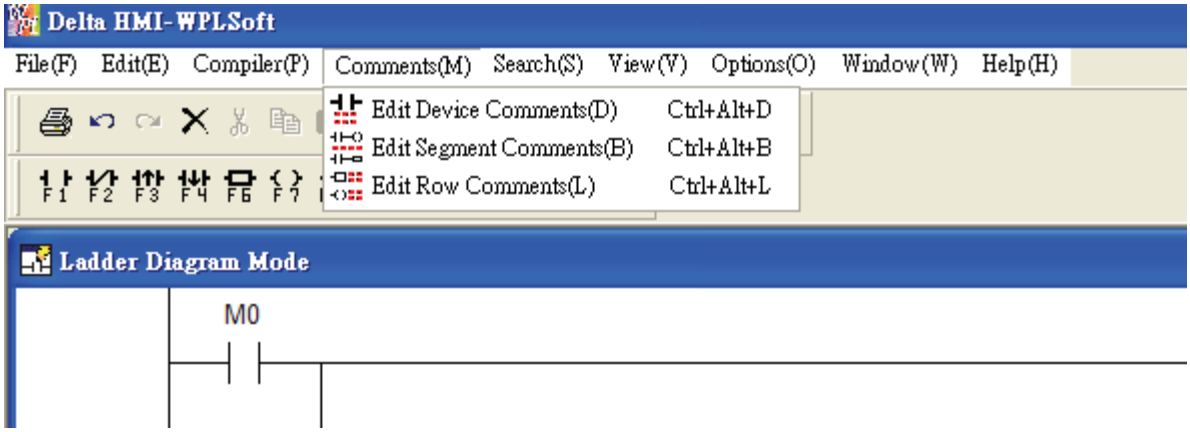
The “Compiler” function is shown as follows, including pull-down menu options:



- Ladder => Instruction(I) ⇨ Convert ladder diagrams to instruction codes.
  - ◈ Method 1: Click “Compiler(P)” > “Ladder => Instruction(I)”.
  - ◈ Method 2: Click the icon on the toolbar.
  - ◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (F9).
- Instruction => Ladder(L) ⇨ Convert instruction codes to ladder diagrams.
  - ◈ Method 1: Click “Compiler(P)” > “Instruction => Ladder(L)”.
  - ◈ Method 2: Click the icon on the toolbar.
  - ◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (F10).

2.4 Comments

The “Comments” function is shown as follows, including pull-down menu options:



- Edit Device Comments(D) ⇨ Insert a comment for every operand of the device where the cursor is positioned.

  ◈ Method 1: Click “Comment(M)” > ” Edit Device Comments(D)”.

  ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Alt) + (D).

  ◈ Method 3: Right click the mouse to get a pop-up menu and select “Edit Device comments” in the pop-up menu.

- Edit Segment Comments(B) ⇨ Insert a segment comment in the blank row (This function is valid for Ladder Diagram Mode only.).

  ◈ Method 1: Click “Comment(M)” > "Edit Segment Comments(B)”.

  ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Alt) + (B).

  ◈ Method 3: Right click the mouse to get a pop-up menu and select “Edit Segment Comments” in the pop-up menu.

- Edit Row Comments(L) ⇨ Insert a row comment after output coil or instruction of each row (This function is valid for Ladder Diagram Mode only.).

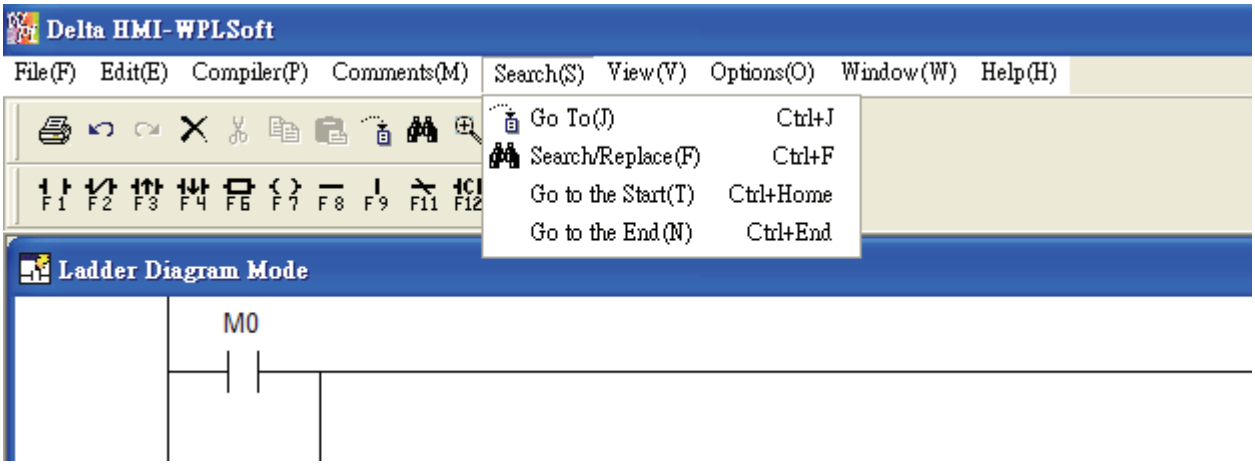
  ◈ Method 1: Click “Comment(M)” > ” Edit Row Comments(L)”.

  ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Alt) + (L).

  ◈ Method 3: Right click the mouse to get a pop-up menu and select “Edit Row comments” in the pop-up menu.

2.5 Search

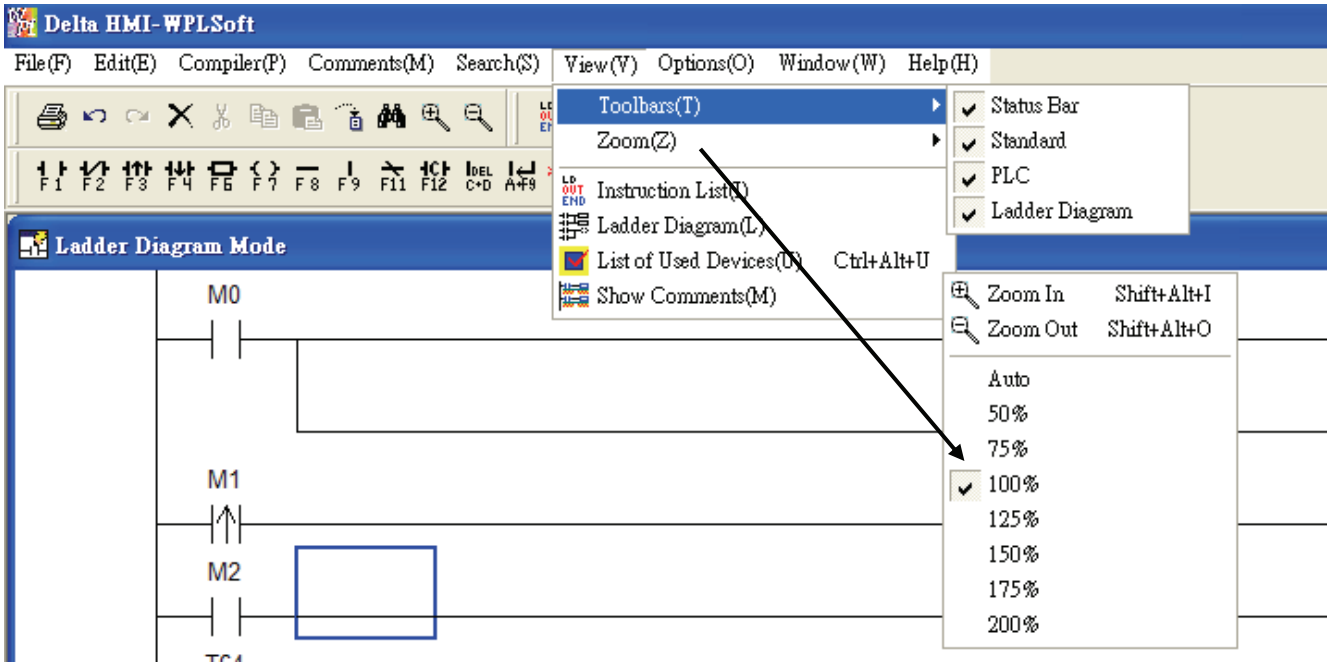
The “Search” function is shown as follows, including pull-down menu options:



- Go to(J) ⇨ Jump to the designated location (unit: Step).
  - ◈ Method 1: Click “Search(S)” > ”Go to(J)”.
  - ◈ Method 2: Click the icon  on the toolbar.
  - ◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (J).
- Search/Replace(F) ⇨ Search or replace the device name and instruction of the designated device.
  - ◈ Method 1: Click “Search(S)” > ”Search/Replace(F)”.
  - ◈ Method 2: Click the icon  on the toolbar.
  - ◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (F).
- Go to the Start(T) ⇨ Jump to the start of the program.
  - ◈ Method 1: Click “Search(S)” > ”Go to the Start(T)”.
  - ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Home).
- Go to the End(N) ⇨ Jump to the end of the program.
  - ◈ Method 1: Click “Search(S)” > ”Go to the End(N)”.
  - ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (End).

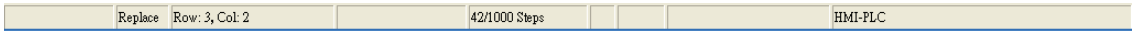
2.6 View

The “View” function is shown as follows, including pull-down menu options:



■ Toolbars(T) ⇨ Display a list of the toolbars available in HMI-WPLSoft, including Status Bar, Standard, PLC and Ladder Diagram toolbars.

⊙ Status Bar: display or hide status bar.



◈ Method: Click “View(V)” > “Toolbars(T)” > “Status Bar”.

⊙ Standard: display or hide standard toolbar.

◈ Method: Click “View(V)” > “Toolbars(T)” > “Standard”.

⊙ PLC: display or hide HMI-WPLSoft toolbar.

◈ Method: Click “View(V)” > “Toolbars(T)” > “PLC”.

⊙ Ladder Diagram toolbar: display or hide Ladder Diagram toolbar (display in Ladder Diagram Mode only).

◈ Method: Click “View(V)” > “Toolbars(T)” > “Ladder Diagram”.

■ Zoom(Z) ⇨ Let the user change and reduce the magnification level of the program.

Zoom In  function is used to get a closer look of the program and Zoom Out  function is used to see more of the program. The default settings for zooming provided by the system are Auto, 50 %, 75 %, 100 %, 125 %, 150 %, 175 % and 200 %.

◈ Method 1: Click “View(V)” > “Zoom (Z)”.

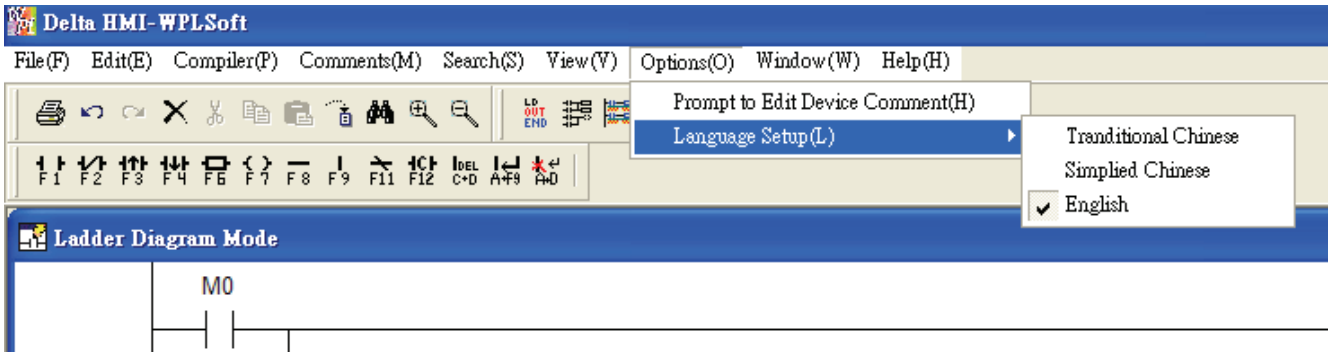
◈ Method 2: Zoom In. Use keyboard shortcuts by pressing keys (Shift) + (Alt) + (I) or click the icon  on the toolbar to zoom in.

◈ Method 3: Zoom Out. Use keyboard shortcuts by pressing keys (Shift) + (Alt) + (O) or click the icon  on the toolbar to zoom out.

- Instruction List(I) ⇨ Change to Instruction Mode.
  - ◈ Method 1: Click “View(V)” > “Instruction List(I)”.
  - ◈ Method 2: Click the icon  on the toolbar.
- Ladder Diagram(L) ⇨ Change to Ladder Diagram Mode.
  - ◈ Method 1: Click “View(V)” > “Ladder Diagram(L)”.
  - ◈ Click the icon  on the toolbar.
- List of Used Device(U) ⇨ Display all device usage status.
  - ◈ Method 1: Click “View(V)” > “List of Used Device(U)”.
  - ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl)+ (Alt) + (U).
- Show Comments(M) ⇨ Display or hide device comments.
  - ◈ Method 1: Click “View(V)” > “Show Comments(M)”.
  - ◈ Method 2: Click the icon  on the toolbar.

2.7 Options

The “Options” function is shown as follows, including pull-down menu options:



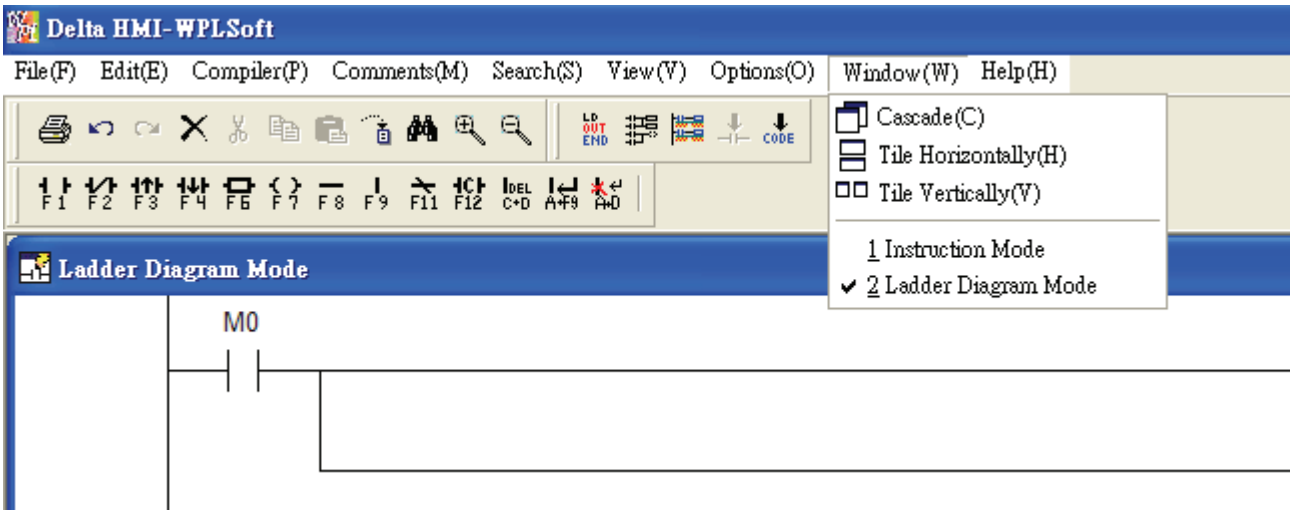
- Prompt to Edit Device Comment(H) ⇨ If this option is selected, in Instruction Mode or Ladder Diagram Mode, the system will ask the user to enter the corresponding device comment at the same time when the user uses the instruction code to edit a DOP-EXIO series program.
  - ◈ Method: Click “Options(O)” > “Prompt to Edit Device Comment(H)”.

- Language Setup(L) ⇨ Allow the user to change the display language of HMI-WPLSoft by requirement. There are three available languages for selection, Tranditional Chinese, Simplied Chinese and English.

  ◈ Method: Click “Options(O)” > “Language Setup(L)”.

2.8 Window

The “Window” function is shown as follows, including pull-down menu options:



- Cascade(C) ⇨ Arrange windows in an overlapping way.

  ◈ Method: Click “Window(W)” > “Cascade(C)”.

- Title Horizontally(H) ⇨ Arrange the file in a horizontal way.

  ◈ Method: Click “Window(W)” > “Title Horizontally(H)”.

- Title Vertically(V) ⇨ Arrange files in a vertical way.

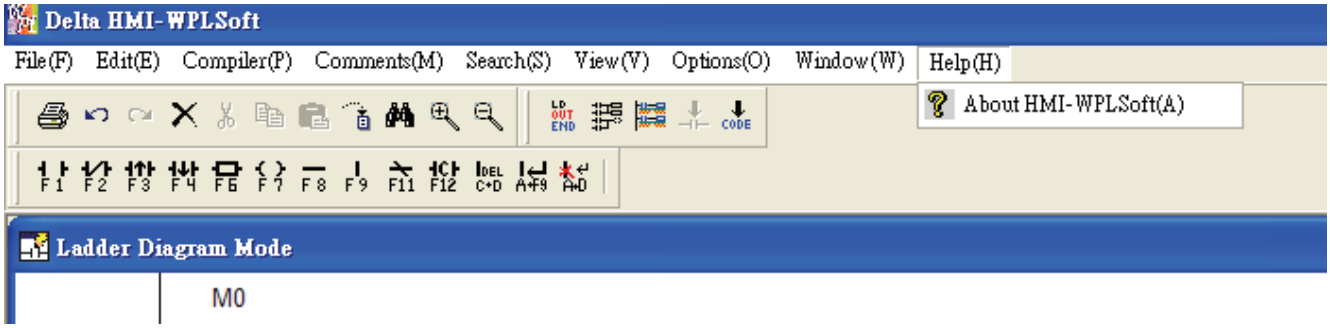
  ◈ Method: Click “Window(W)” > “Title Vertically(V)”.

- The current files list ⇨ e.g. Instruction Mode and Ladder Diagram Mode.

  ◈ Method: In HMI-WPLSoft editing window, activate Instruction Mode and Ladder Diagram Mode and click “Window(W)”, and then the user can see them show in Window drop-down menu.

2.9 Help

The “Help” function is shown as follows, including pull-down menu options:



- About HMI-WPLSoft(A) ⇨ This command shows the version information of HMI-WPLSoft.

  ◈ Method: Click “Help(H)” > “About HMI-WPLSoft(A)”.

Chapter 3 Creating and Editing Programs

Activate HMI-WPLSoft, and then the system will enter into Ladder Diagram Mode as shown as the Fig. 3.1 below.

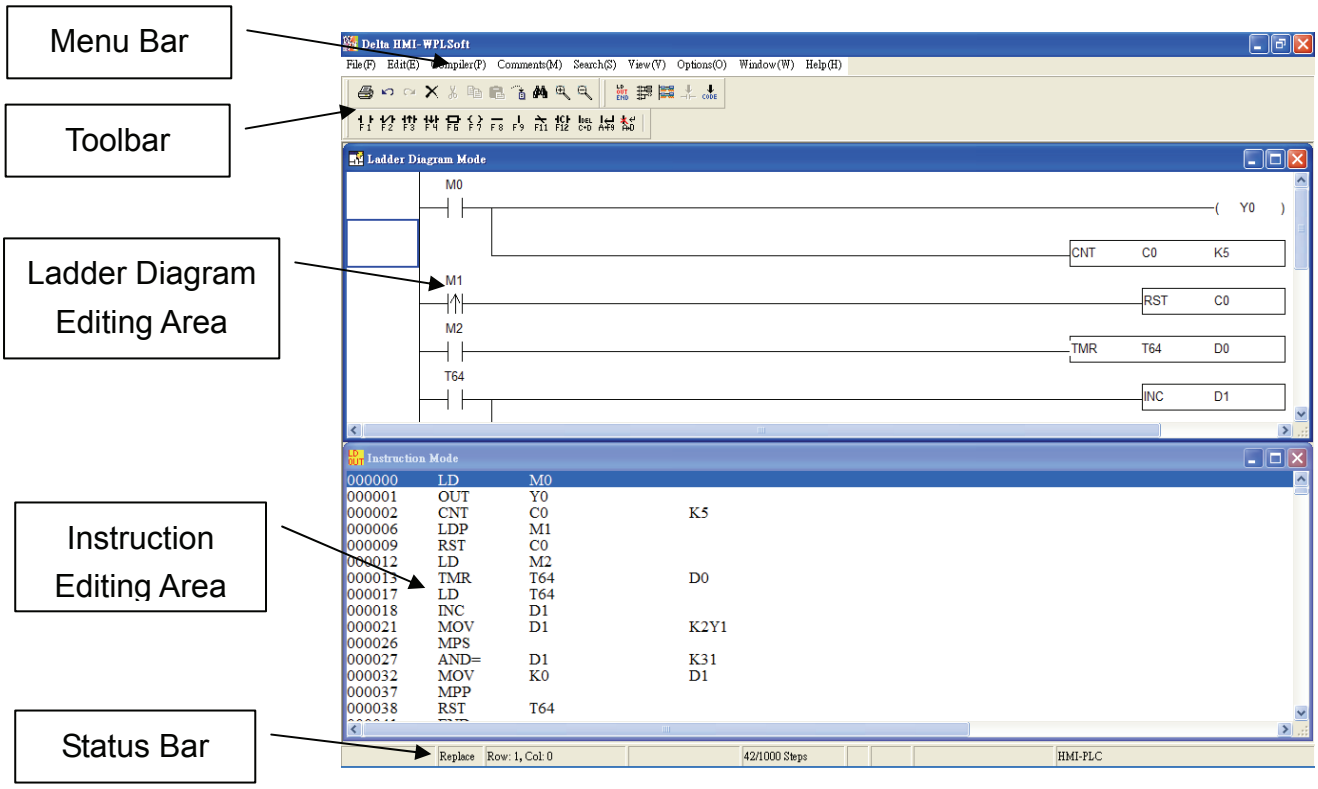
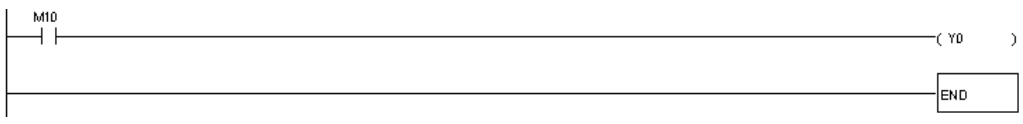


Fig. 3.1 Ladder Diagram Mode

There is a ladder diagram toolbar shown on the top of the Ladder Diagram Mode window. To create and edit a ladder diagram, the user can click the icon on toolbar directly by the mouse or move the editing block to the proper position and enter instructions. Besides, the user also can press F1 ~ F12 function keys on the keyboard to create and edit the ladder diagram. Please refer to the following sections for how to create and edit ladder diagram.

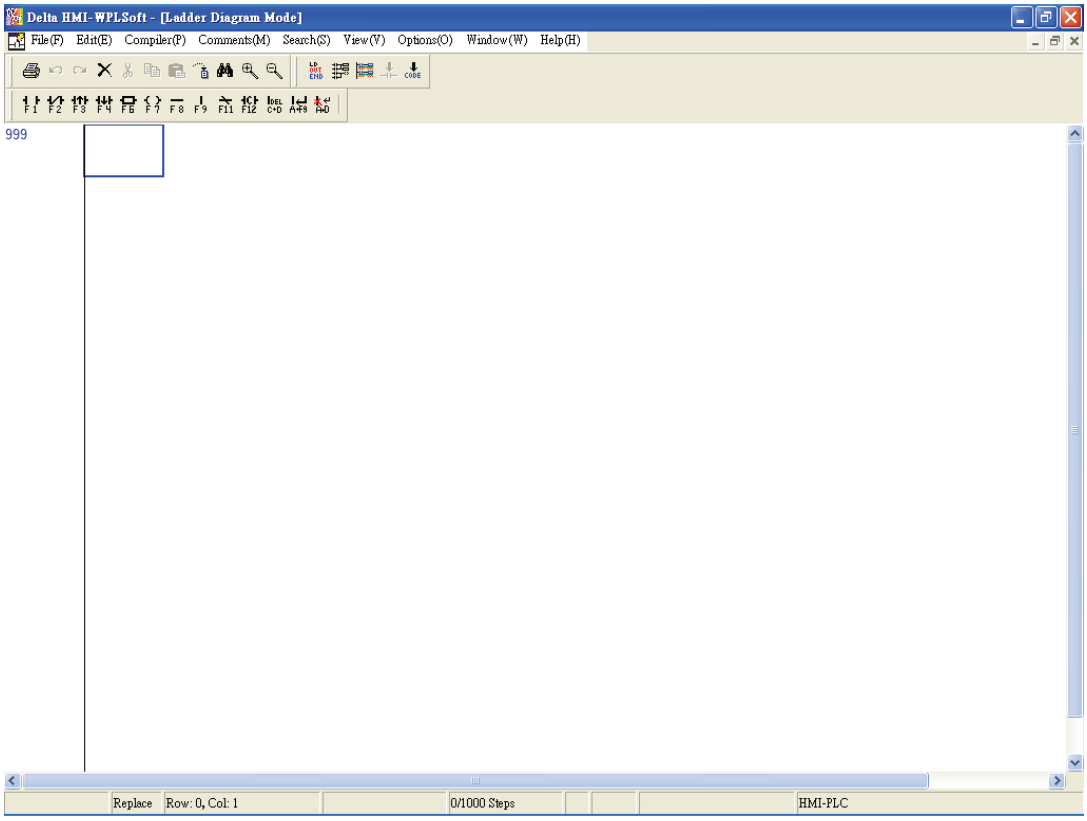
3.1 Basic Operation

Example: Create the diagram shown below.

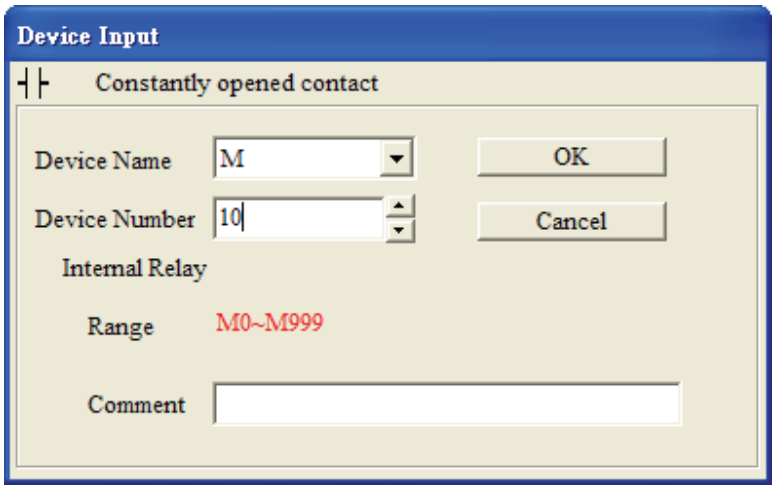


- Using the mouse and F1 ~ F12 function keys on the keyboard.

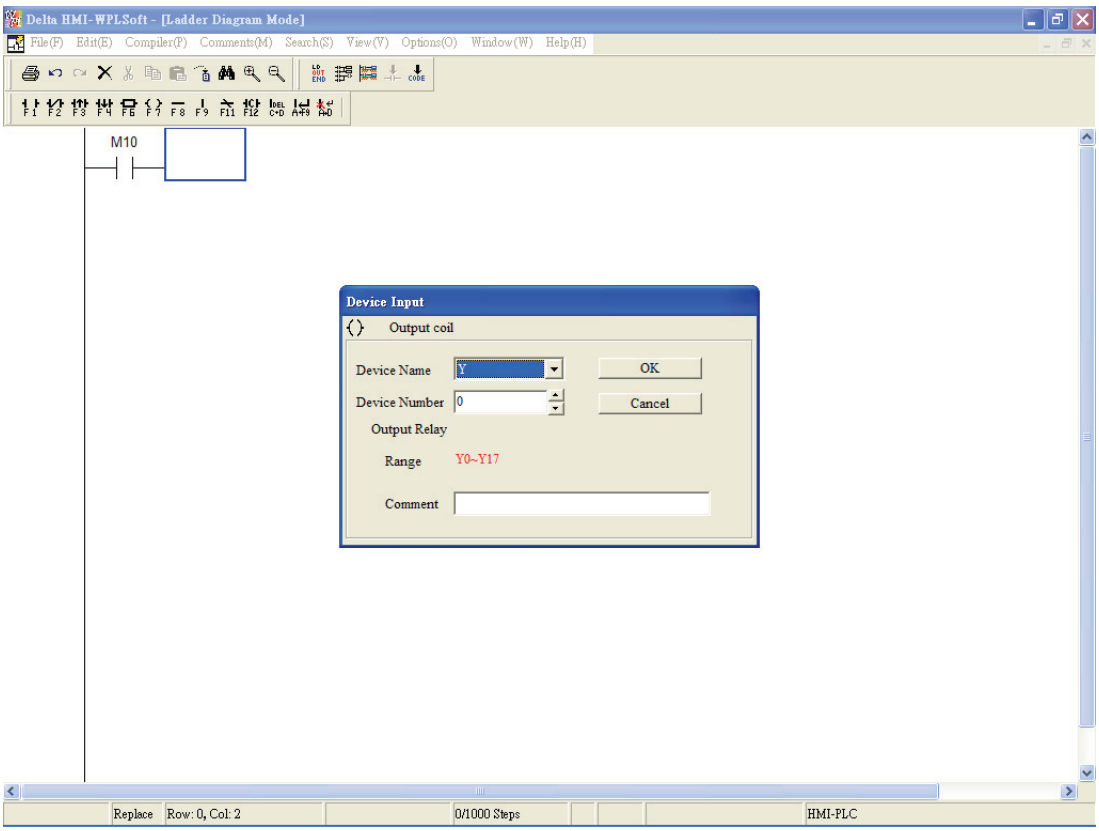
1. Click the Normally Open Contact icon  on the toolbar or press F1 function key.



2. The “Device Input” dialog box will appear. The user can select device name (e.g. M) , device number (e.g. 10), and enter comments (e.g. Internal Relay). Then, press the button “OK” to save the settings.

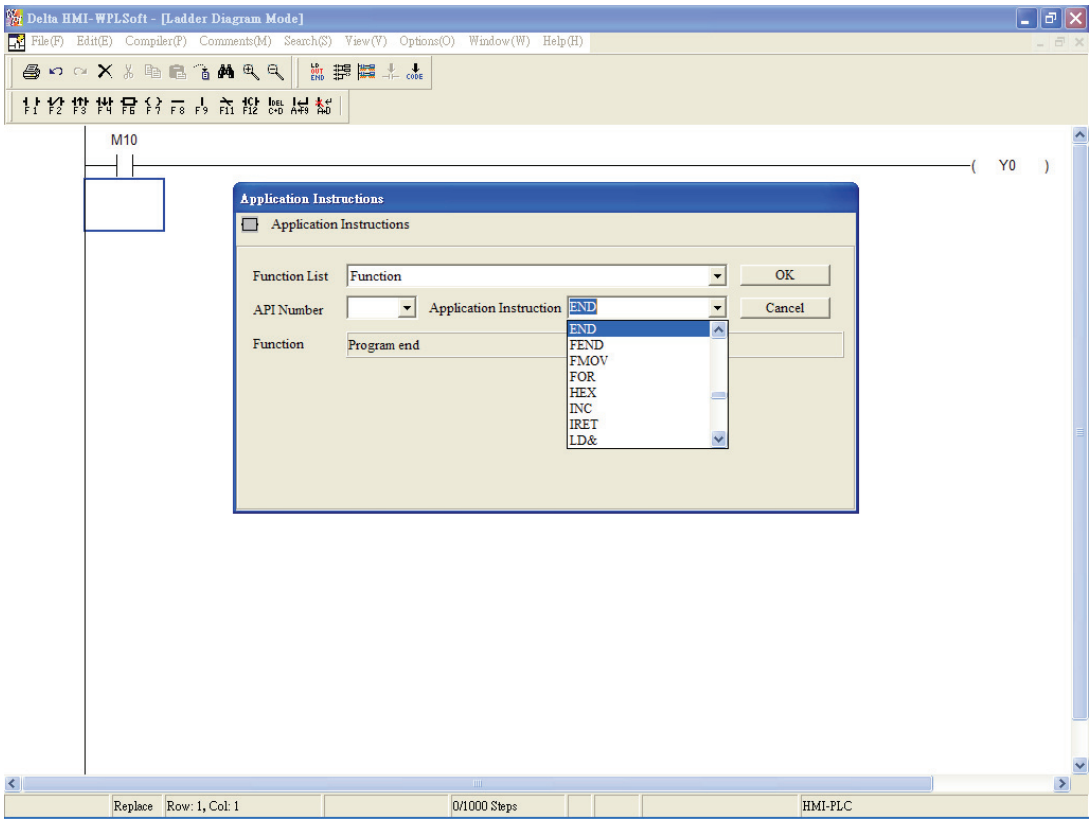


3. Click the Output Coil icon  on the toolbar or press F7 function key. The “Device Input” dialog box will appear next. The user can select device name (e.g. Y), device number (e.g. 0), and enter comments (e.g. Output Relay). Then, press the button “OK” to save the settings.



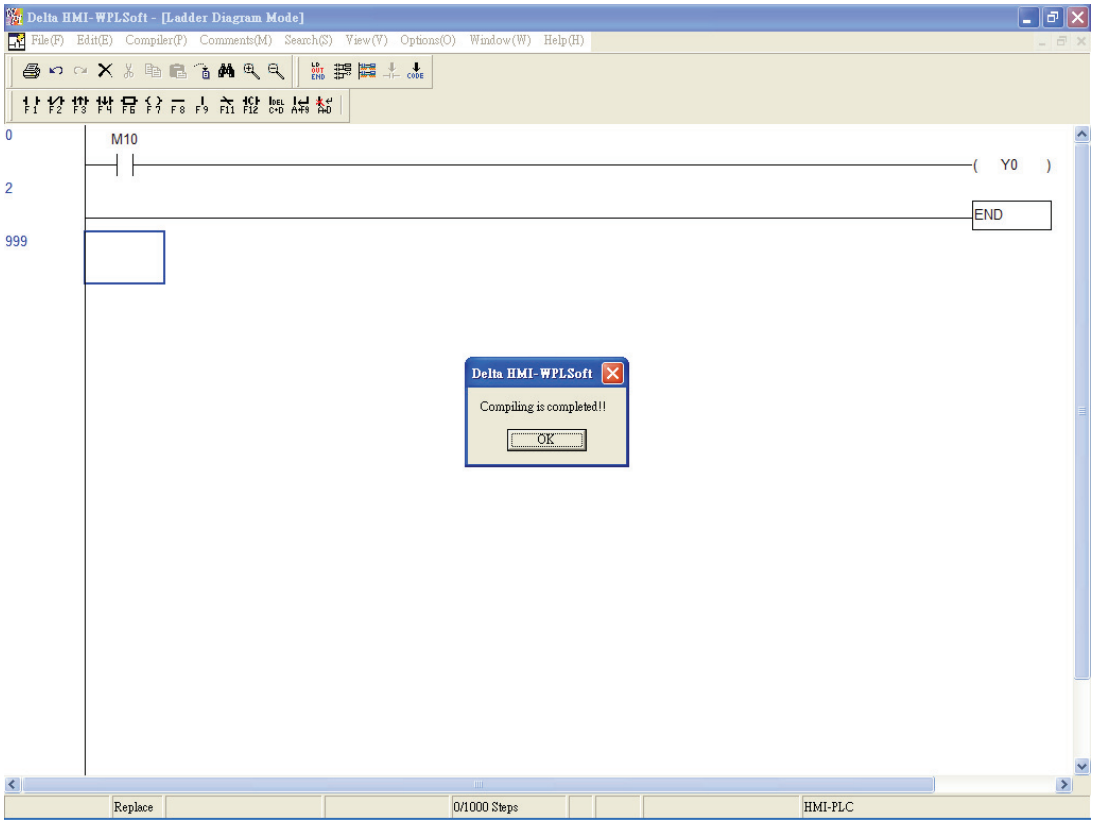
4. Click Application Instruction icon 

or press F6 function key. Choose “Function” from the “Function” drop-down menu and select “END” instruction from the “Application Instruction” drop-down menu. The user can also type in “END” instruction directly in the field of "Application Instruction". Then, press the button “OK” to save the settings.

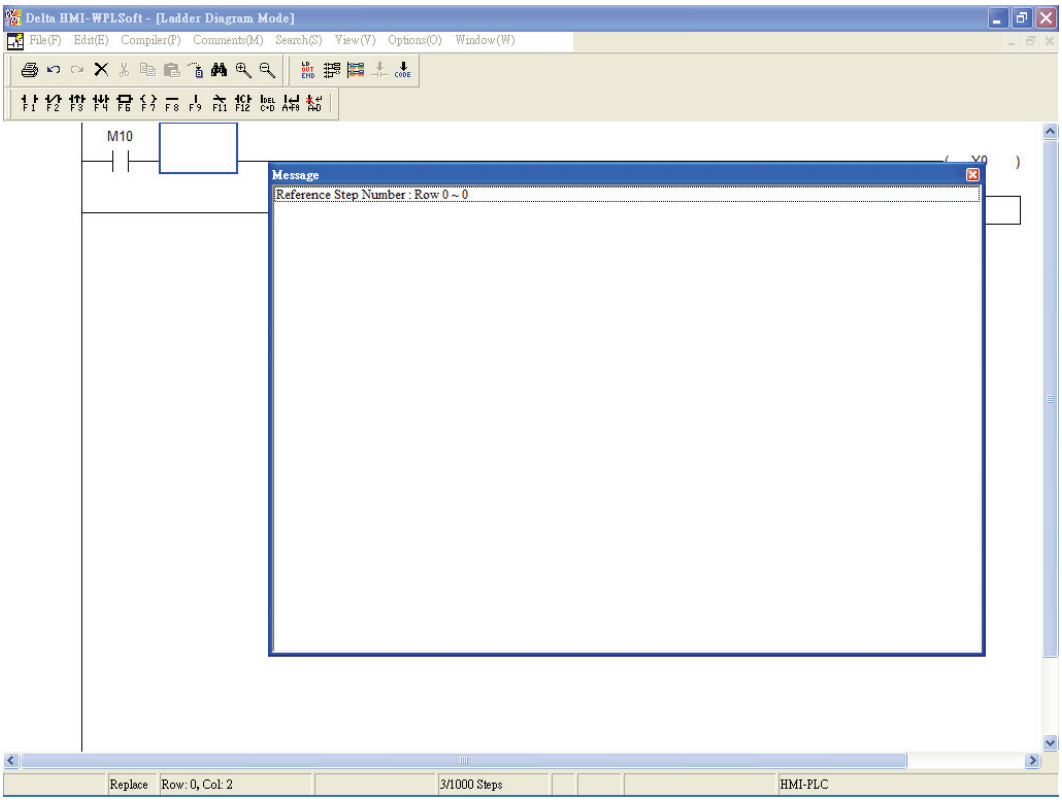


5. Click the icon  to compile the ladder diagram and convert it to instruction codes.

   After compiler action is completed, the numbers of steps will show on the left-hand side of the start of the ladder diagram.

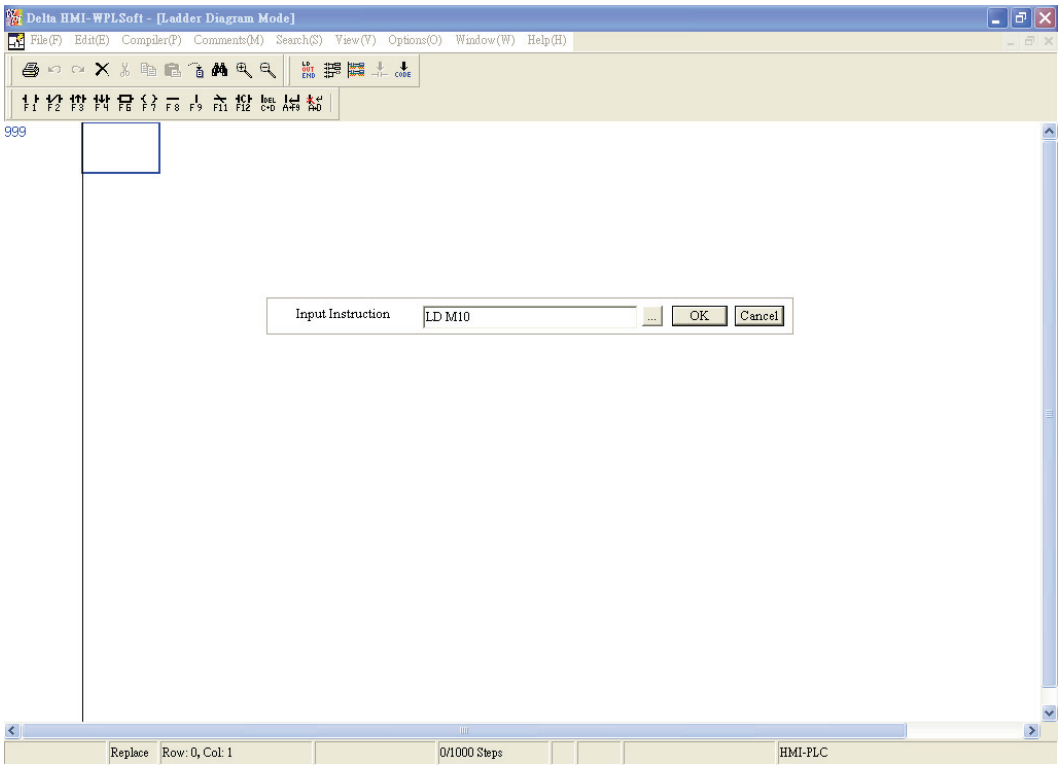


6. If the ladder diagram is not correct, an error message dialog box will appear and point out the exact erroneous rows and addresses after the compiler action is completed.



■ Keyboard Operation

1. Place the editing block at the start of the program (Row: 0, Col: 1), and type in “LD M10” by using the keyboard. Then, press the Enter key on the keyboard, or click the “OK” button to complete the settings.



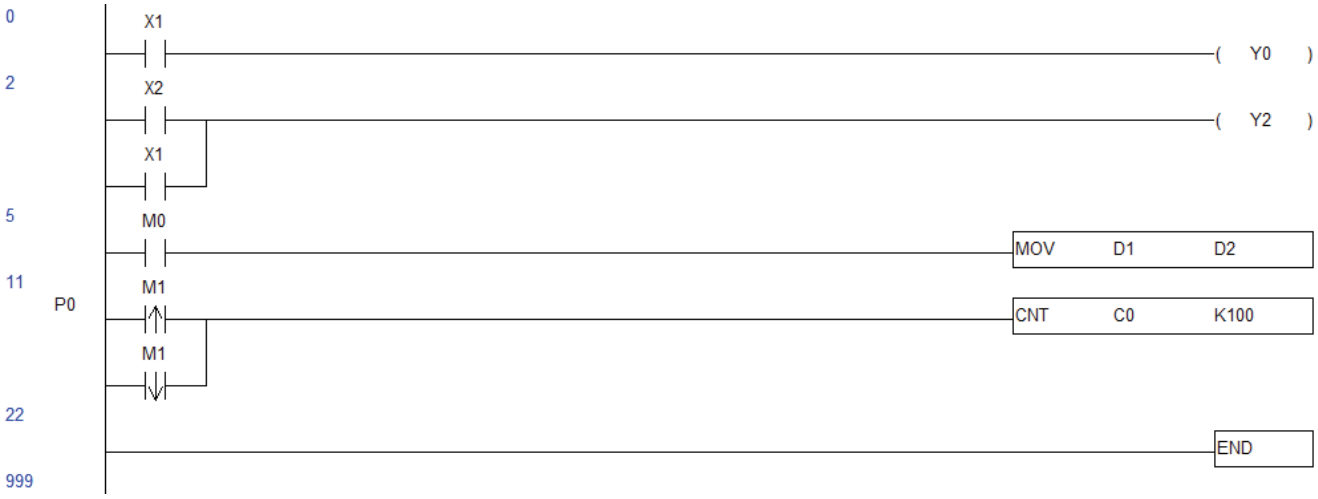
2. Type in “OUT Y0” by using the keyboard and press Enter key on the keyboard. Then, type in “END” by using the keyboard and press Enter key on the keyboard. Finally, click the icon 

to compile the completed ladder diagram.

If the user wants to edit the comments at the same time when input an instruction by using keyboard, the user can click the “Prompt to Edit Device Comment(H)” under the “Options” menu. Then, the “Comment” dialog box (see the figure below) will appear for the user to enter and edit the corresponding comments after an instruction is input correctly.

3.2 Editing Example

Ladder Diagram

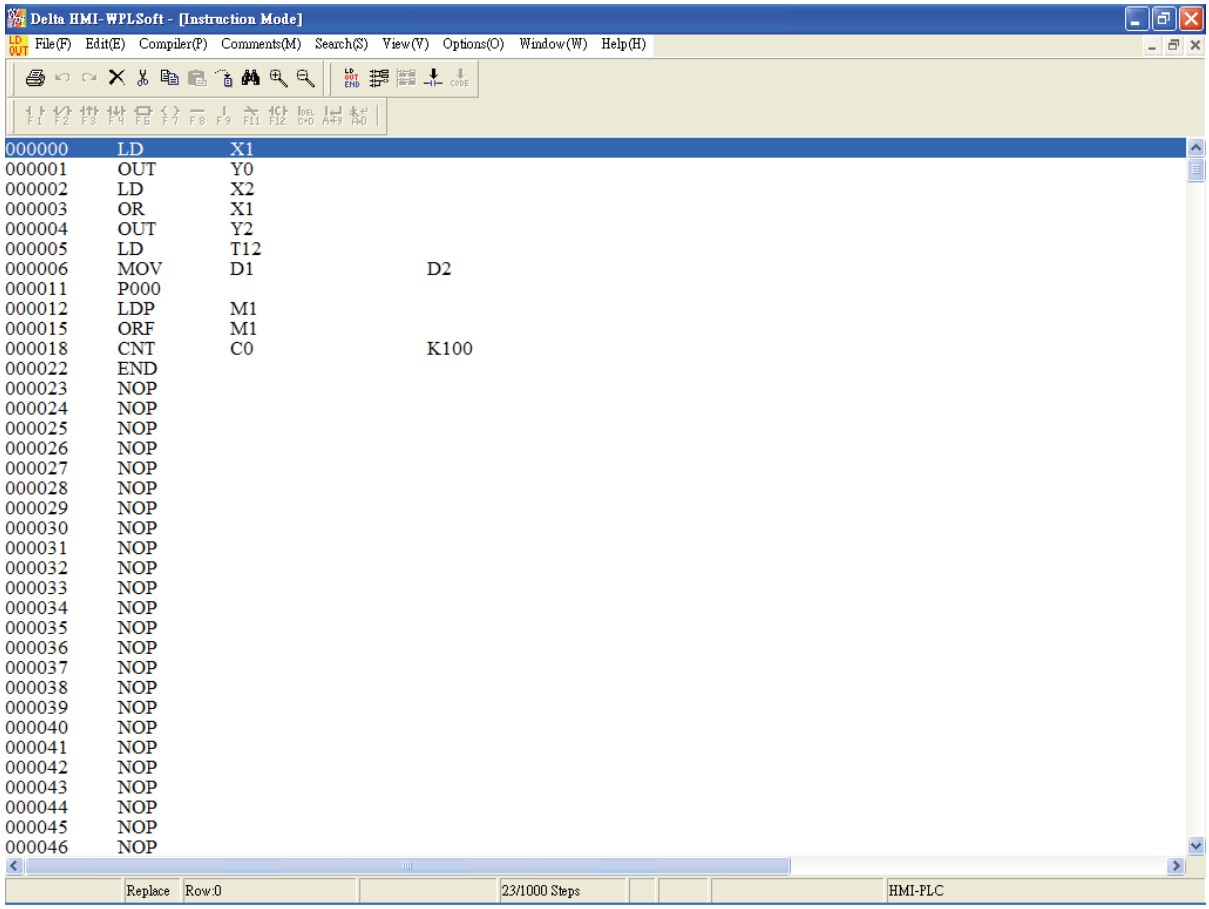


■ Operation steps for editing the Ladder Diagram

Step	Ladder Symbol	Cursor Location	Input by clicking the Icon on the toolbar		Input by using the Keyboard
1	⊣ ⊢	Row: 0, Column: 1	F1 *Footnote 1	Device Name: X Device Number: 1	LD X1 ↵ or A X1 ↵
2	—( )	Row: 0, Column:2	F7 *Footnote 2	Device Name: Y Device Number:0	OUT Y0↵ or O Y0
3	⊣ ⊢	Row: 1, Column: 1	F1	Device Name: X Device Number:2	LD X2 ↵ or A X2 ↵
4	|	Row: 1, Column: 2	F9		F9
5	—( )	Row: 1, Column: 2	F7	Device Name: Y Device Number:2	OUT Y2 ↵ or O Y2 ↵
6	⊣ ⊢	Row: 2, Column: 1	F1	Device Name: X Device Number:1	LD X1 ↵ or A X1 ↵
7	⊣ ⊢	Row: 3, Column: 1	F1	Device Name: M Device Number:0	LD M0 ↵ or A M0 ↵
8	—▭	Row: 3, Column: 2	F6 *Footnote 3	MOV Instruction Operand 1: D Device Number:1 Operand 2: D Device Number:2	MOV D1 D2 ↵
9		Row: 4, Column: 0		Double click the mouse and enter P0	P0 ↵

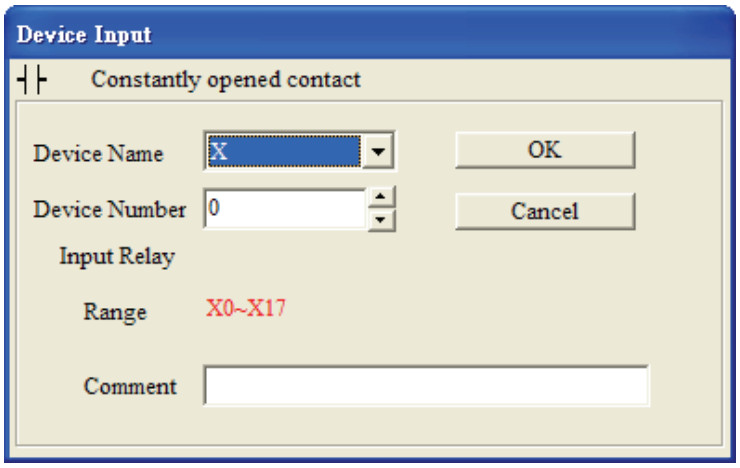
Step	Ladder Symbol	Cursor Location	Input by clicking the Icon on the toolbar		Input by using the Keyboard
10	⊣↑⊢	Row: 4, Column: 1	F3	Device Name: M Device Number:1	LDP M1 ↵ or + M1 ↵
11	|	Row: 4, Column: 2	F9		F9
12	⊣ ▭	Row: 4, Column: 2	F6	CNT Instruction Operand 1: C Device Number: 0 Operand 2: K Device Number: 100	CNT C0 K100 ↵
13	⊣↓⊢	Row: 5, Column: 1	F4	Device Name: M Device Number: 1	LDF M1 ↵ or – M1 ↵
14	⊣ ▭	Row: 6, Column: 1	F6	END Instruction	END ↵

After the ladder diagram is completed, the user can compile and convert the completed ladder diagram to instruction codes. The ladder diagram which has been converted to instruction codes is shown as the figure below.

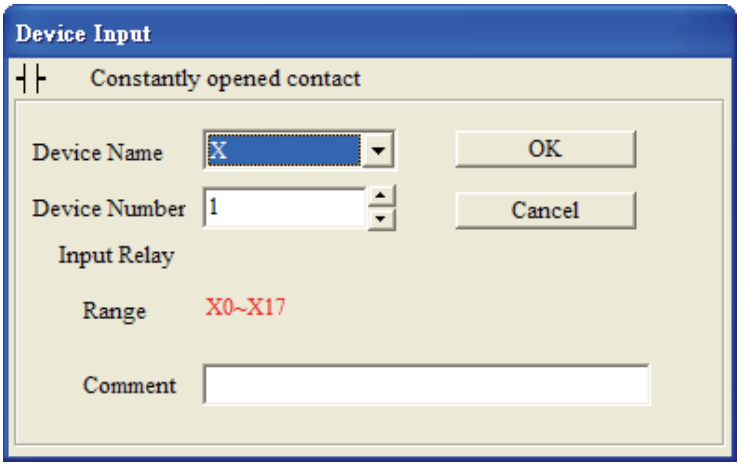


*Footnote 1: Input Basic Instruction

1. Click the icon  on the toolbar or press the F1 function key on the keyboard and the “Device input” dialog box will appear. Then, the user can enter device name, device number, and edit comments in this dialog box.

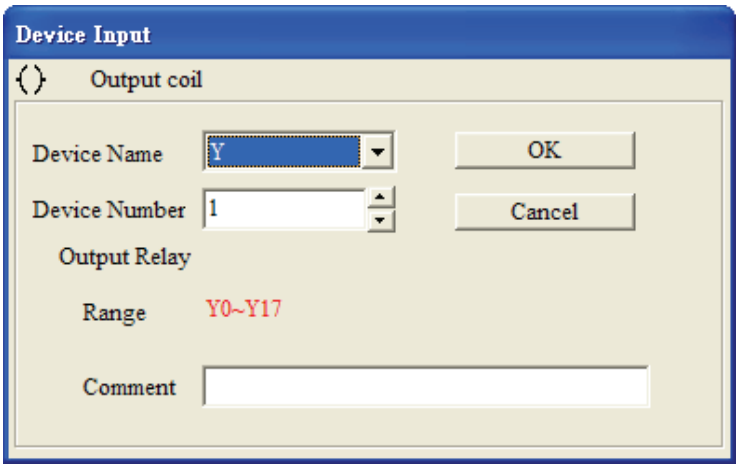


2. For example, select the device name “X” and device number “1” from the drop-down menu or type in the device name “X” and device number “1” by using the keyboard. Then, press Enter key on the keyboard or click the “OK” button to save the settings.



*Footnote 2: Input Output Coil

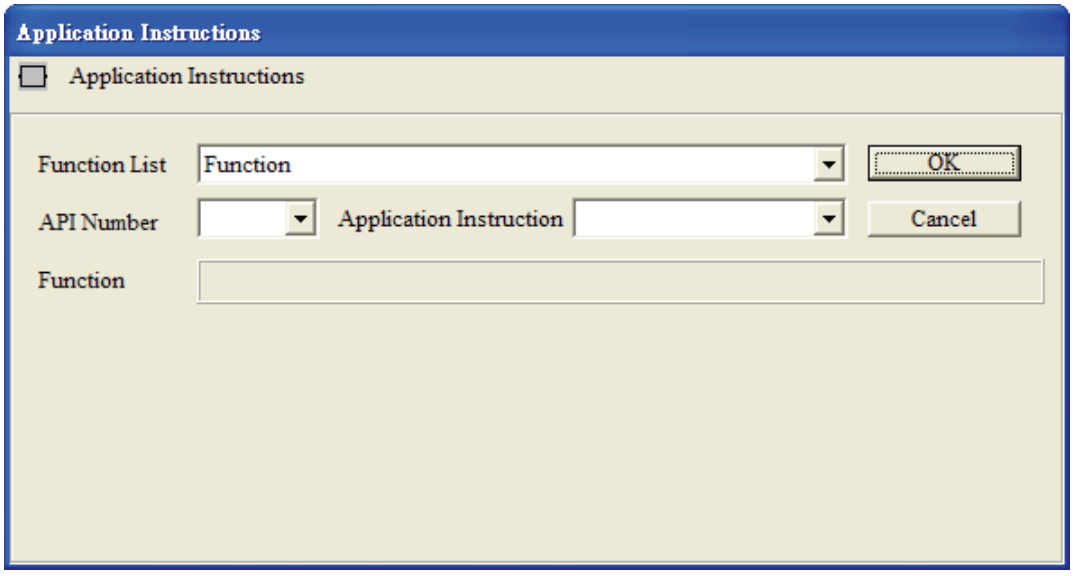
1. Click the icon  on the toolbar or press the F7 function key on the keyboard and the “Device input” dialog box will appear. Then, the user can enter device name, device number, and edit comments in this dialog box.



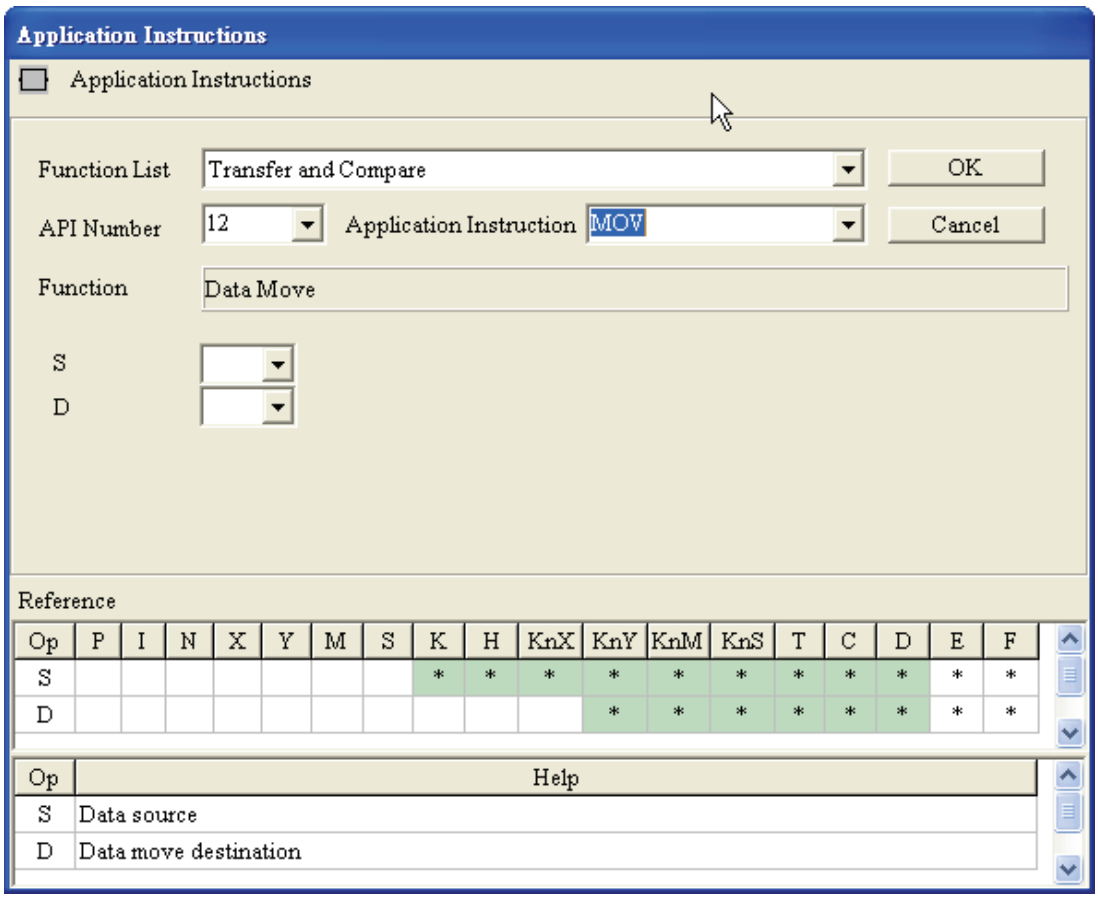
2. For example, select the device name “Y” and device number “1” from the drop-down menu or type in the device name “Y” and device number “1” by using the keyboard. Then, press Enter key on the keyboard or click the “OK” button to save the settings.

*Footnote 3: Input Application Instruction

1. Click the icon  on the toolbar or press the F6 function key on the keyboard and the “Application Instructions” dialog box will appear.



2. First, choose one selection from the “Function List” drop-down menu (including all application instructions and output commands, etc.). Then, select the “API Number” and “Application Instruction”. The user can also type in the desired instruction, e.g. MOV in the "Application Instruction" drop-down menu directly. After all settings are completed, press Enter key on the keyboard.
3. Select “Transfer and Compare” from the “Function List” drop-down menu and type in “MOV” in the field of "Application Instruction" directly (or choose “MOV” instruction from the “Application Instruction” drop-down menu). Then, press Enter key on the keyboard, and the user can see the figure below on the screen.



4. Input device name in the field of “S” (Operand 1) and “D” (Operand 2), and input device number in the field of “Device Number” in order. Select index register E or F if it exists. Then, press the “OK” button to save the settings.
5. The user can also double click the mouse on the “@” or “*” symbol in the device reference table (refer to the figure above) to designate the device name (The symbol @ indicates this device can be modified by index register E or F and the symbol * indicates this device can not be modified by index register E or F).

3.3 Ladder Diagram Editing Explanation

☞ **Keyboard Entry**

HMI-WPLSoft provides several brevity codes for the user to input Instructions more quickly and conveniently when editing a ladder diagram. Please refer to the following table.

Explanation	Instruction Icon	Instruction Code (Mnemonic Code)	Brevity Code	Example
Normally open contact		LD	A	LD M0 or A M0
Normally closed		LDI	B	LDI M0 or B M0

Explanation	Instruction Icon	Instruction Code (Mnemonic Code)	Brevity Code	Example
contact				
Rising pulse		LDP	+	LDP M0 or + M0
Falling pulse		LDF	–	LDF M0 or – M0
Output coil		OUT	O	OUT M0 or O M0

☞ **Insert / Replace Mode**

Using the “Insert” key on the keyboard can switch to the Insert Mode or the Replace Mode when editing a ladder diagram.

◈ If the “Replace” word is displayed on the status bar, pressing the Insert key on the keyboard is to switch to the Insert Mode. In the Insert Mode, insert a new ladder diagram to where the editing block is located, and the original ladder diagrams following the new diagram will shift one space to the right.

◈ If the “Insert” word is displayed on the status bar, pressing the Insert key on the keyboard is to switch to the Replace Mode. In Replace Mode, inserting a new ladder diagram can replace the original ladder diagram located in the editing block, and the following other ladder diagrams will not be changed.

☞ **Edit(E)**

■ Undo(U) ⇨ Undo the most recent actions (the system allows the user to perform undo action for max. 10 times)

◈ Method 1: Click “Edit(E)” > “Undo(U)”.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (Z).

◈ Method 4: Right click the mouse to get a pop-up menu and select “Undo” in the pop-up menu.

■ Redo(R) ⇨ Redo the undo action.

◈ Method 1: Click “Edit(E)” > “Redo(R)”.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (Alt) + (Z).

◈ Method 4: Right click the mouse to get a pop-up menu and select “Redo” in the pop-up menu.

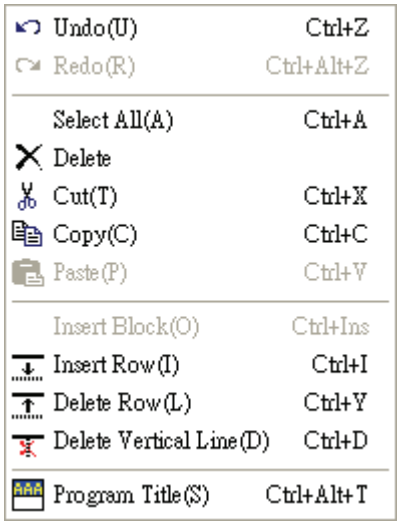
■ Delete ⇨ Delete a selection (selected block or data) where the cursor is.

◈ Method 1: Click “Edit(E)” > “Delete”.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing key (Delete).

◈ Method 4: Move the cursor to the diagram block that the user wants to delete and right click the mouse to get a pop-up menu and select “Delete” in the pop-up menu to delete the selected diagram block.

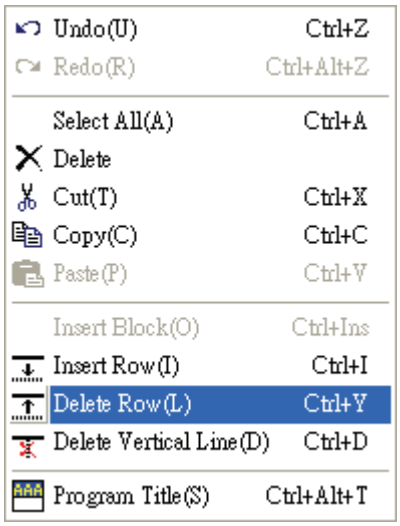


■ Delete Row(L) ⇨ Delete a row or several rows in the ladder diagram

◈ Method 1: Click “Edit(E)” > “Delete Row(L)”. Then, the row where the cursor is will be deleted and the rows below the deleted row will move up.

◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Y).

◈ Method 3: Move the cursor to the row that the user wants to delete and right click the mouse to get a pop-up menu and select “Delete Row(L)” in the pop-up menu to delete the row.



◈ Method 4: Select the row that the user wants to delete. Right clicking the mouse to select the “Delete” command in the pop-up menu can delete the selected row immediately. Pressing the Delete key on the keyboard or clicking the icon  on the toolbar can also delete the selected row.

■ Delete Vertical Line(D) ⇨ Delete the vertical lines in the ladder diagram.

◈ Method 1: Click “Edit(E)” > “Delete Vertical Line(D)”. Then, the vertical line on the left-hand side of the editing block will be deleted.

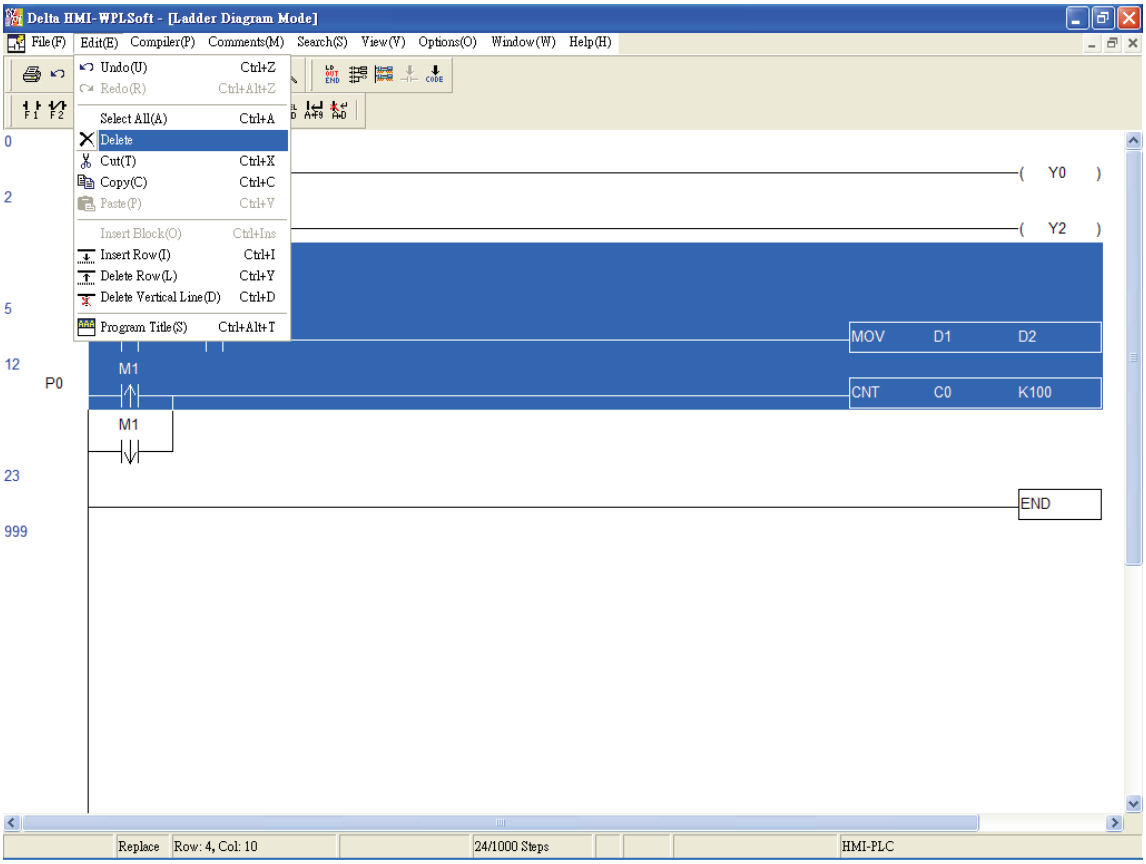
◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (D).

◈ Method 3: Move the editing block to the right-hand side of the vertical line that the user wants to delete and click the icon  on the toolbar. Then, the vertical line on the left-hand side of the editing block will be deleted.

◈ Method 4: Move the editing block to the right-hand side of the vertical line that the user wants to delete. Right click the mouse to get a pop-up menu and select “Delete Vertical Line(D)” in the pop-up menu. Then, the vertical line on the left-hand side of the editing block will be deleted.

■ Delete Block ⇨ Delete the selected block.

◈ Method 1: Click “Edit(E)” > “Delete”. Then, the selected block in the ladder diagram that the user wants to delete will be deleted immediately.



◈ Method 2: Select the block that the user wants to delete and click the icon on the toolbar.

◈ Method 3: Select the block that the user wants to delete and right click the mouse to select the “Delete” command in the pop-up menu.

◈ Method 4: Select the block that the user wants to delete and press the Delete key on the keyboard.

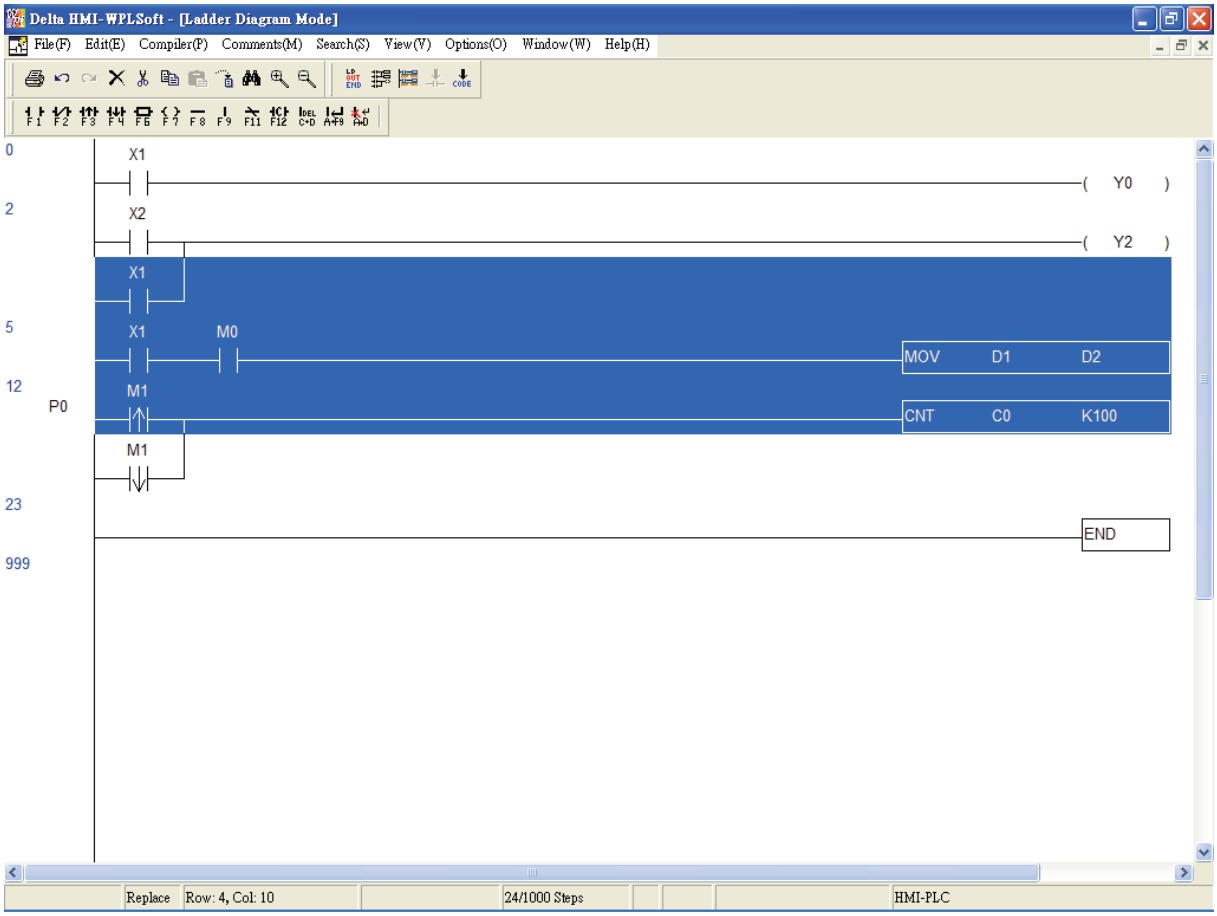
■ Copy Block ⇨ Copy the selected block.

◈ Method 1: Click “Edit(E)” > “Copy(C)”.

◈ Method 2: Click the icon on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (C).

◈ Method 4: Right click the mouse to select the “Copy” command in the pop-up menu.



■ Cut Block ⇨ Cut the selected block.

◈ Method 1: Click “Edit(E)” > “Cut(T)”.

◈ Method 2: Click the icon on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (X).

◈ Method 4: Right click the mouse to select the “Cut” command in the pop-up menu.

■ Paste Block ⇨ Paste the selected block.

◈ Method 1: Click “Edit(E)” > “Paste(P)”.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (V).

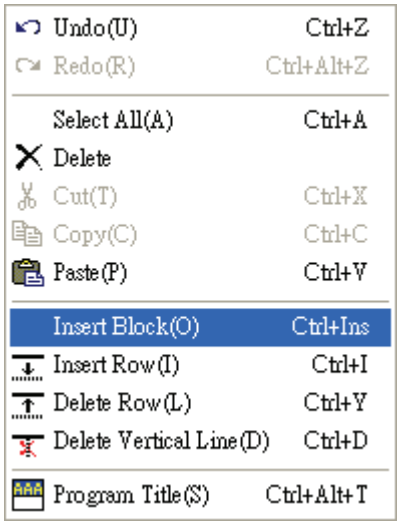
◈ Method 4: Right click the mouse to select the “Paste” command in the pop-up menu.

■ Insert Block ⇨ Insert the selected block (This function is valid after the “Copy Block” function is executed. Therefore, before inserting the selected block, perform “Copy Block” action first).

◈ Method 1: Click “Edit(E)” > “Insert Block(O)”.

◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Ins).

◈ Method 3: Right click the mouse to select the “Insert Block” command in the pop-up menu.



☞ **Compiler(P)**

This function is used to compile current HMI-WPLSoft programs for DOP-EXIO series product. If the user completes the editing of the ladder diagram in the ladder diagram mode, performing this function will check whether the ladder diagram is valid or not. If there is no error occurred when converting the program, the ladder diagram can be converted to the instruction program successfully; meanwhile, the program memory addresses (numbers of steps) for each editing block will appear on the left-hand side of the start of the ladder diagram. However, if there is any error occurred, a ladder diagram error message dialog box will appear to display the error code and point out the exact erroneous addresses (exact row and column where the error occurred) after the compiler action is completed.

If the user completes program editing in the instruction mode when performing this function, the system will start to check if there is any error occurred or not. If there is no error, the

instruction program will be converted to the ladder diagram successfully. However, if there is any error occurred, an error message dialog box will appear to display the error code and point out the exact erroneous steps (where the error occurred) after the compiler action is completed.

■ Ladder => Instruction(I) ⇨ This function is valid in Ladder Diagram Mode only.

◈ Method 1: Click “Compiler(P)” > “Ladder => Instruction(I)”.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (F9).

■ Instruction => Ladder(L) ⇨ This function is valid in Instruction Mode only.

◈ Method 1: Click “Compiler(P)” > “Instruction => Ladder(L)”.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (F10).

☞ **Search(S)**

■ Go to(J) ⇨ Jump to the designated location (unit: Step). This command is used to specify the program to jump to a designated location. If the designated step already exists, the program will jump to this existing designated step and put it in the first line.

◈ Method 1: Click “Search(S)” > ”Go to(J)”. Enter the designated step where the user want to jump to, and then the ladder diagram will put this designated step in the first line.

◈ Method 2: Click the icon  on the toolbar.

◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (J).

■ Search/Replace(F)

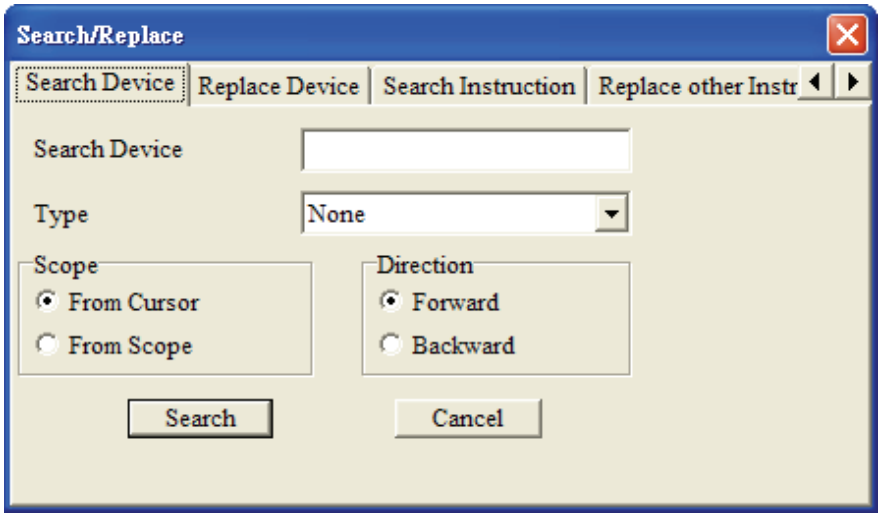
The “Search/Replace(F)” command is used to search and replace the device and instruction within the program (if only the “Search” command is used, just enter the device name to be searched in the dialog box). If the device or the command is found, the view will be scrolled to the device or the command. Also, the user can search and replace the device and instruction by specifying the type of the device and instruction.

◈ Method 1: Click “Search(S)” > ”Search/Replace(F)”.

◈ Method 2: Click the icon  on the toolbar.

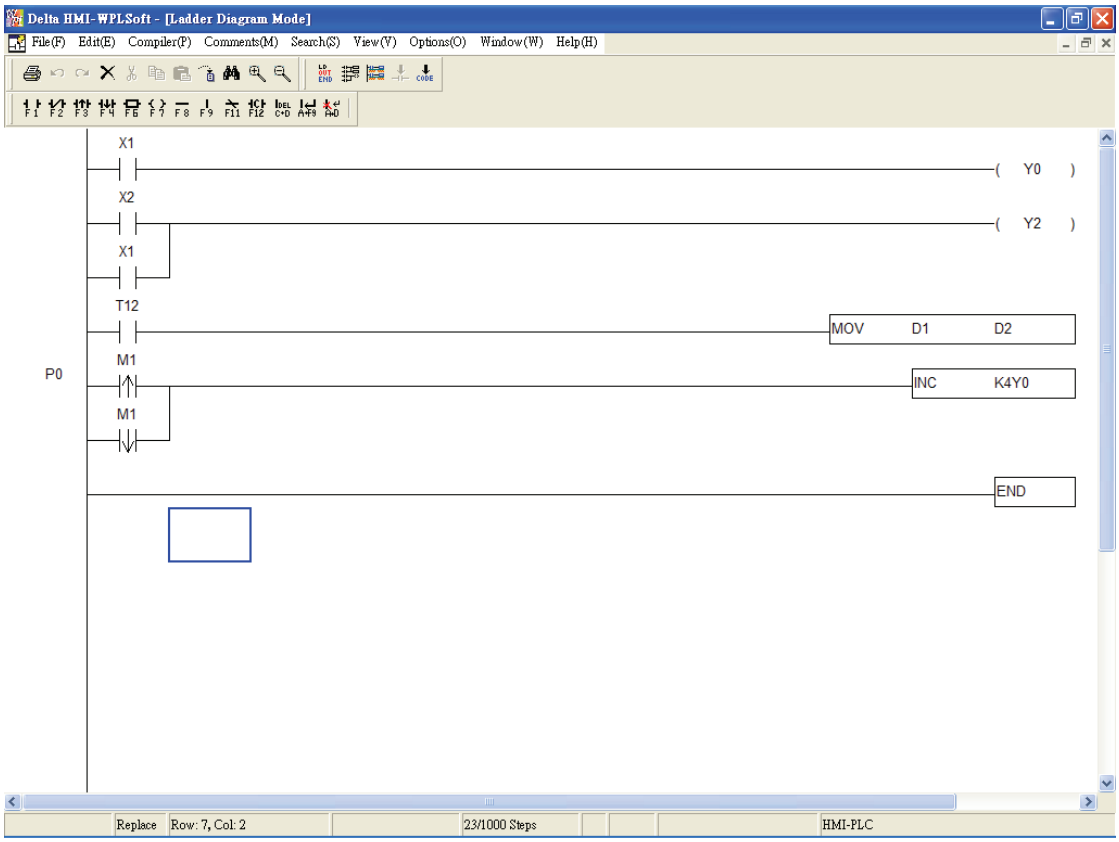
◈ Method 3: Use keyboard shortcuts by pressing keys (Ctrl) + (F).

When the user selects the “Search/Replace” command, the following “Search/Replace” dialog box will appear. There are “Search Device”, “Replace Device”, “Search Instruction” and “Replace other Instruction” four functions in this dialog box for the user to use.

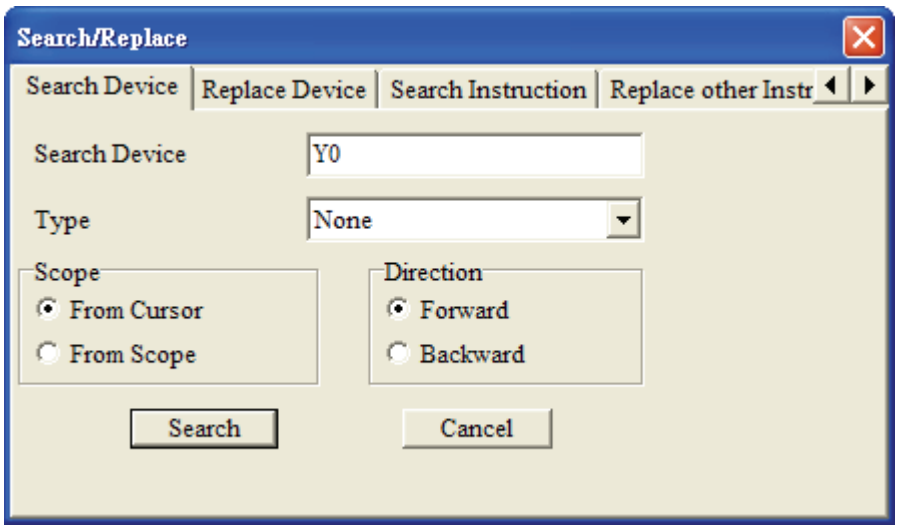


◈ Search Device

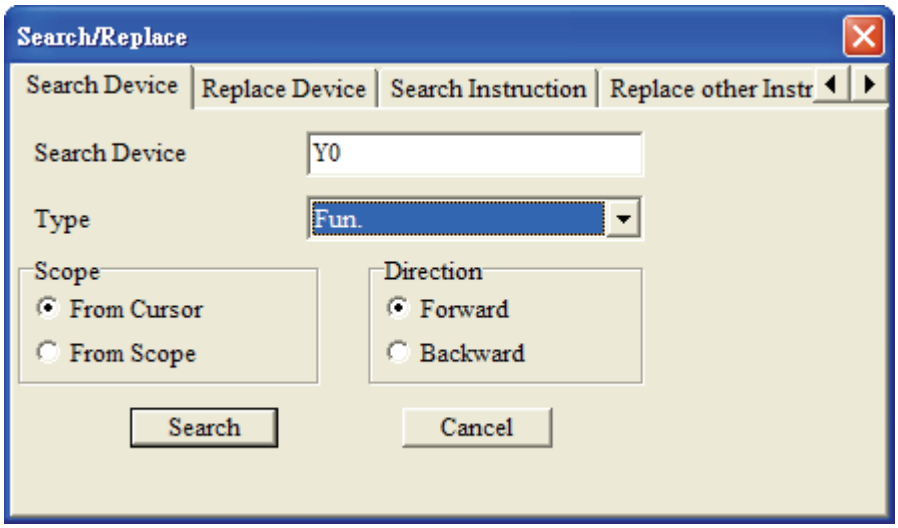
Use this command to search the specified device name match the search criteria in the program. For example, in the ladder diagram shown below, the instructions that contain device name Y0 are OUT Y0 and INC K4Y0.



Activate “Search/Replace” fuction to open “Search/Replace” dialog box and choose the “Search Device” tab. Enter device name “Y0” in the field of “Search Device” and select the “None” in the field of “Type”. Then, press the “Search” button and the system will find the instructions “OUT Y0” and “INC K4Y0”.



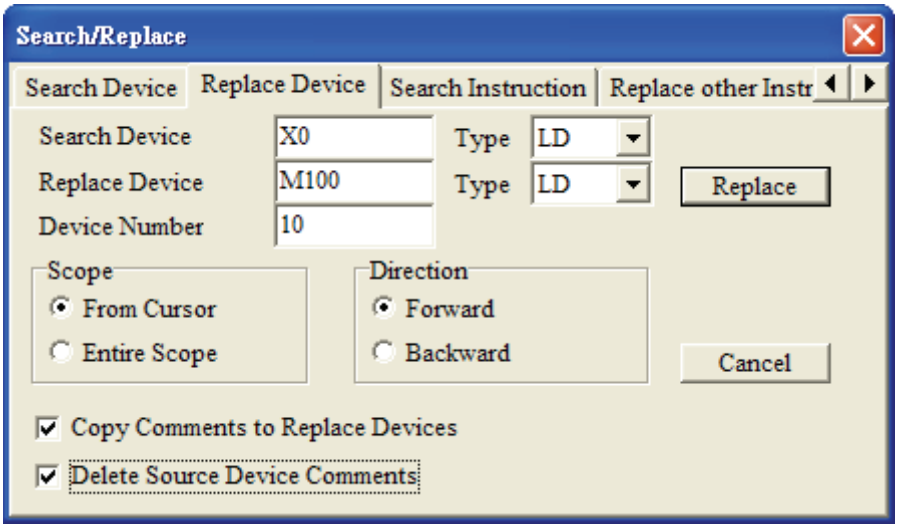
However, if enter the device name “Y0” in the field of “Search Device” still, but change the “None” to “Fun.” in the field of “Type”, only the instruction “INC K4Y0” will be found when “Search” button is pressed.



◈ Replace Device

Use this command to replace the specified device name match the search criteria in the program. For example, activate “Search/Replace” fuction and choose the “Replace Device” tab. In “Replace Device” tab, enter the “X0” in the field of “Search Device” and select the type of search device as “LD”. Then, enter the “M100” in the field of “Replace Device” and select the type of replace device as “LD”. Next, type in “10” in the field of “Device Number”. Finally, press the “Replace” button, and the instructions which match the criteria will be changed to LD M100~M109.

Original Command	Criteria	Replaced Command
LD X0~X7 LD X10~X11	Type LD + Device X0 → Type LD + Device M100 Device Number: 10	LD M100~M107 LD M108~M109



If the user choose the device type as None, Out and Fun these three types, only the same type of the device which the name match the replace criteria can be replaced.

When None, Out and Fun these three types are selected, if the user tries to replace the different type of the device, a warning message dialog box looks like the figure below will appear.

Besides, the user can use “Copy Comments to Replace Devices” this option to copy the comments into the replace device. If “Delete Source Device Comments” this option is also selected, the comments of the search device will be deleted after the comments of the search device has been copied to the replace device. In this case, the boxes next to “Copy Comments to Replace Devices” and “Delete Source Device Comments” are checked both. It indicates that when the device name is replaced, i.e. the “Replace” button is pressed, the comment of the search device “X0” will be copied to the replace device “M100” and the comment of the search device “X0” will be deleted immediately at the same time.

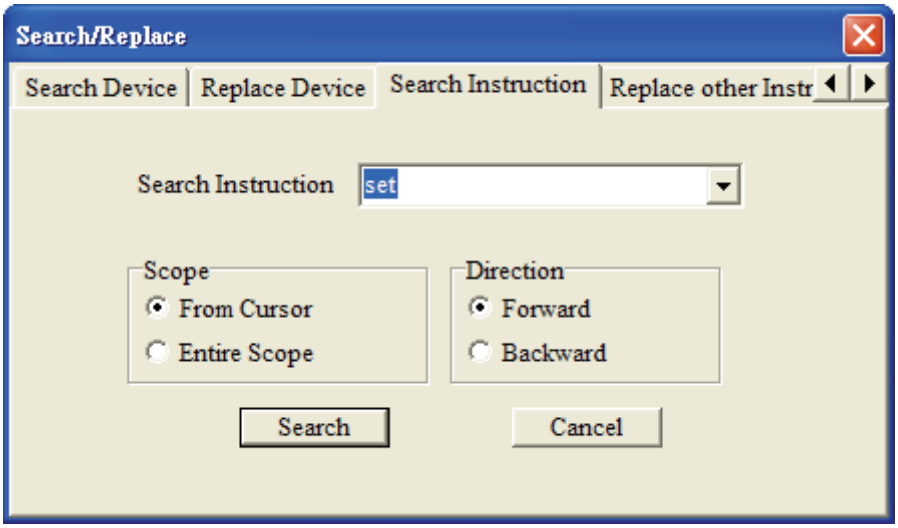
✍ Limits

In “Replace Device” dialog box, only the devices of the same type can be replaced. For example, if D1 is replaced by D11, it is thus viewed as successful replacement; but if it

is replaced by C100, it is then a failure.

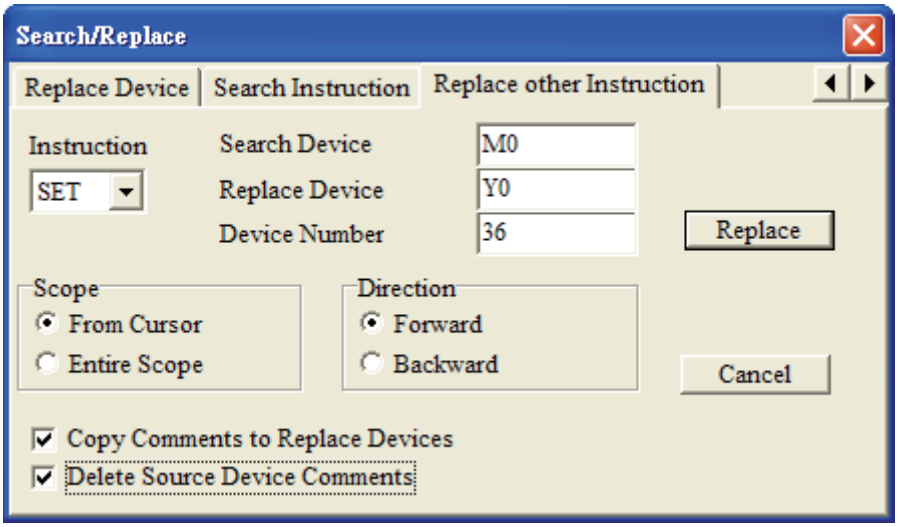
◈ Search Instruction

Use this command to search the specified instruction name match the search criteria in the program. Click “Search Instruction” tab after the “Seach/Replace” function is activated, and enter the instruction name that the user is looking for in the field of “Search Instruction”. Then, press the “Search” button to start the search. The system will memorize and record all the searched instruction names in the “Search Instruction” drop-down menu. This is a useful function for the user to search more quickly and conveniently next time.



◈ Replace other Instruction

In “Replace other Instruction” tab, the system provides the replace criteria for SET, RST, PLS and PLF, these four kinds of instructions and allows the user to replace the devices match the criteria of these instructions in the program. For example, if the user wants to replace SET M0 ~ M35 with SET Y0 ~ Y43, in order to complete the replacement, the user can set the settings as shown as the figure below.



Besides, as the boxes next to “Copy Comments to Replace Devices” and “Delete Source Device Comments” are checked both, it indicates that when the device name is replaced, the comments of the search device “M0 ~ M35” will be copied to the replace device “Y0 ~ Y43” and the comments of the search device “M0 ~ M35” will be deleted immediately at the same time.

- Go to the Start(T) ⇨ Jump to the start of the program.

  ◈ Method 1: Click “Search(S)” > ”Go to the Start(T)”.

  ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (Home).

- Go to the End(N) ⇨ Jump to the end of the program.

  ◈ Method 1: Click “Search(S)” > ”Go to the End(N)”.

  ◈ Method 2: Use keyboard shortcuts by pressing keys (Ctrl) + (End).

3.4 Editng Instructions

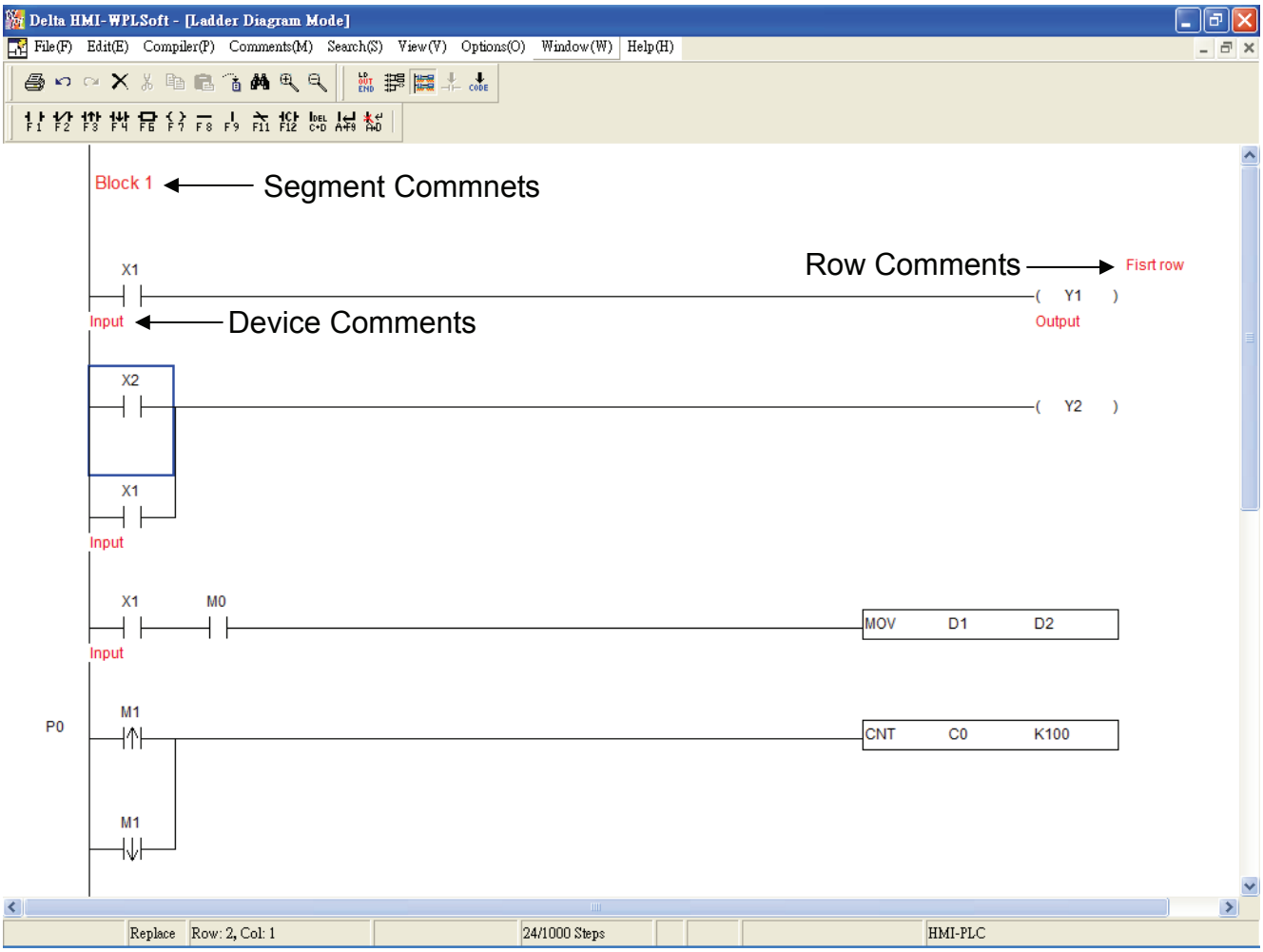
- Input DOP-EXIO Series Instructions

  After entering the instruction mode, the user may type an instruction directly. If the instruction format is valid, press the Enter key on the keyboard to complete the settings. The input instructions will be located in the editing area and the program memory address of DOP-EXIO series will appear on the left-hand side of the program. Thus, the user can get the corresponding program memory addresses of the instructions clearly. For the introductions of the formats of all instructions, please refer to Appendix A and Appendix B in this manual.

3.5 Editing Comments

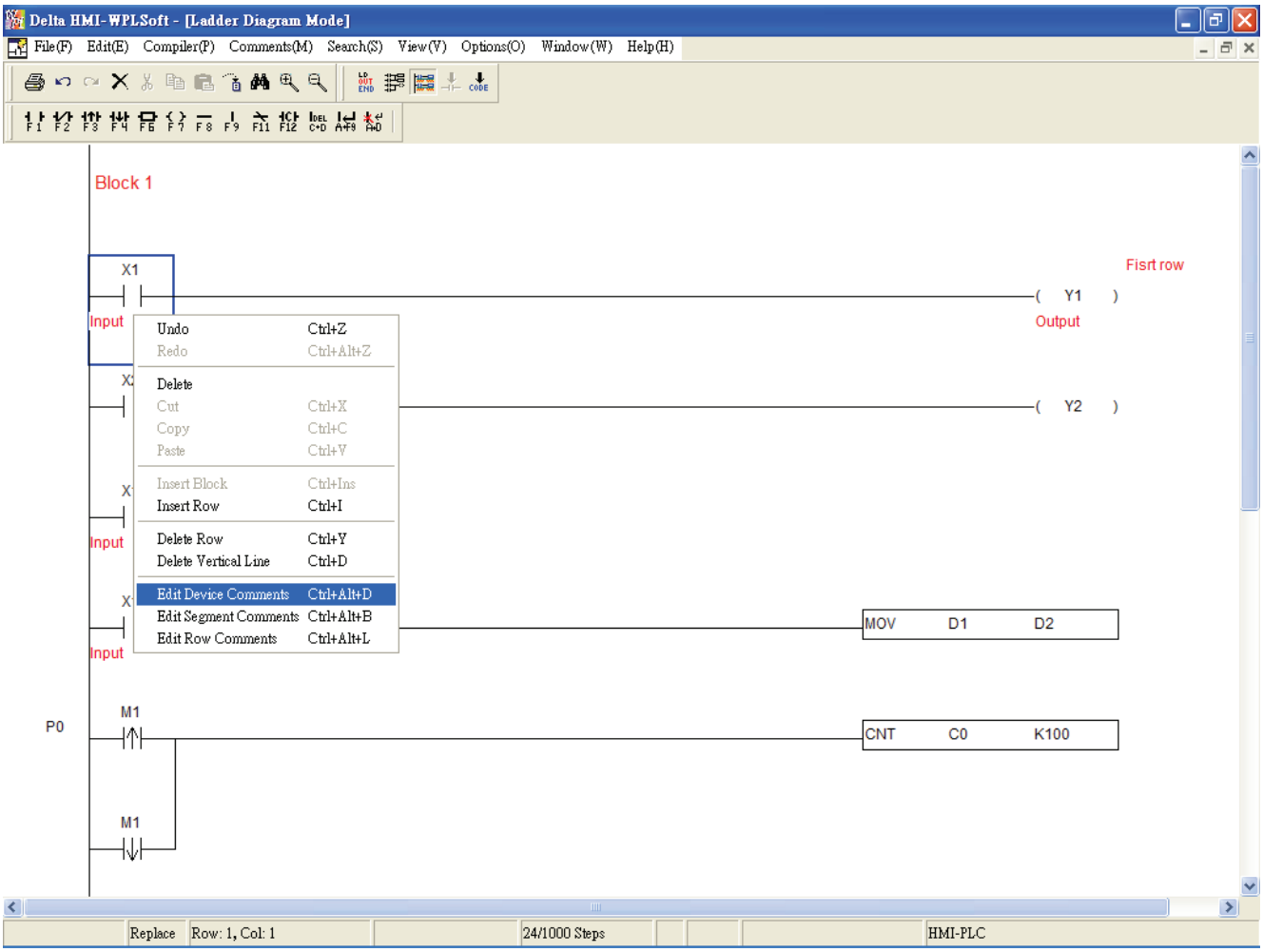
In the ladder diagram mode, there are three operating modes for editing comments: Device comments, Segment comments and Row comments. Please refer to the following sections for more introductions on editing comments.

Ladder Diagram Mode:



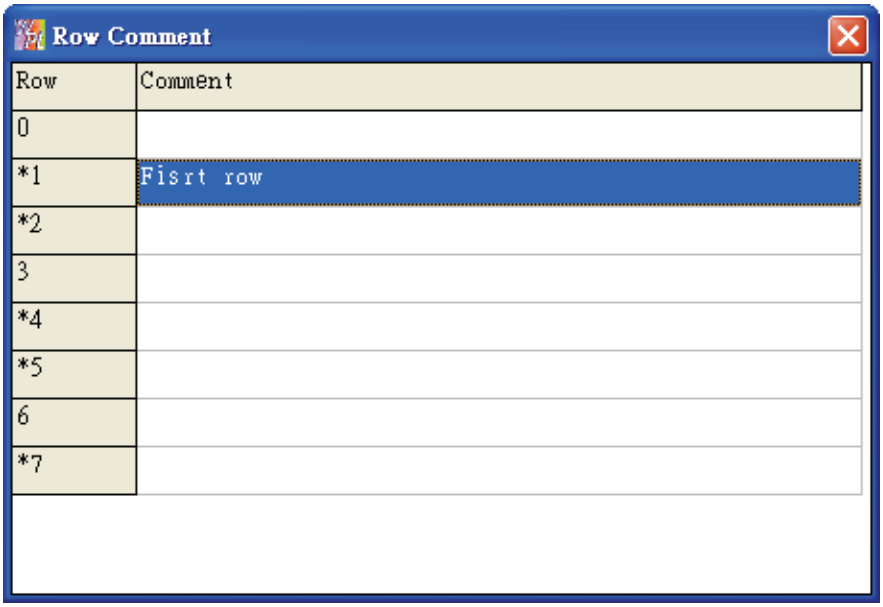
■ Edit Device Comments

Move the editing block on the desired device and right click the mouse. The pop-up menu box shown on the following figure will appear. From this pop-up menu, choosing “Edit Device Comments” can enter and edit device comments. After editing the comments is completed, press “Enter” key on the keyboard or click the “OK” button by using the mouse to have the record saved.



■ Edit Row Comments: (Only for ladder diagram mode)

Enable this function, and then the user can edit all row comments at the same time.



■ Edit Segment Comments: (Only for ladder diagram mode)

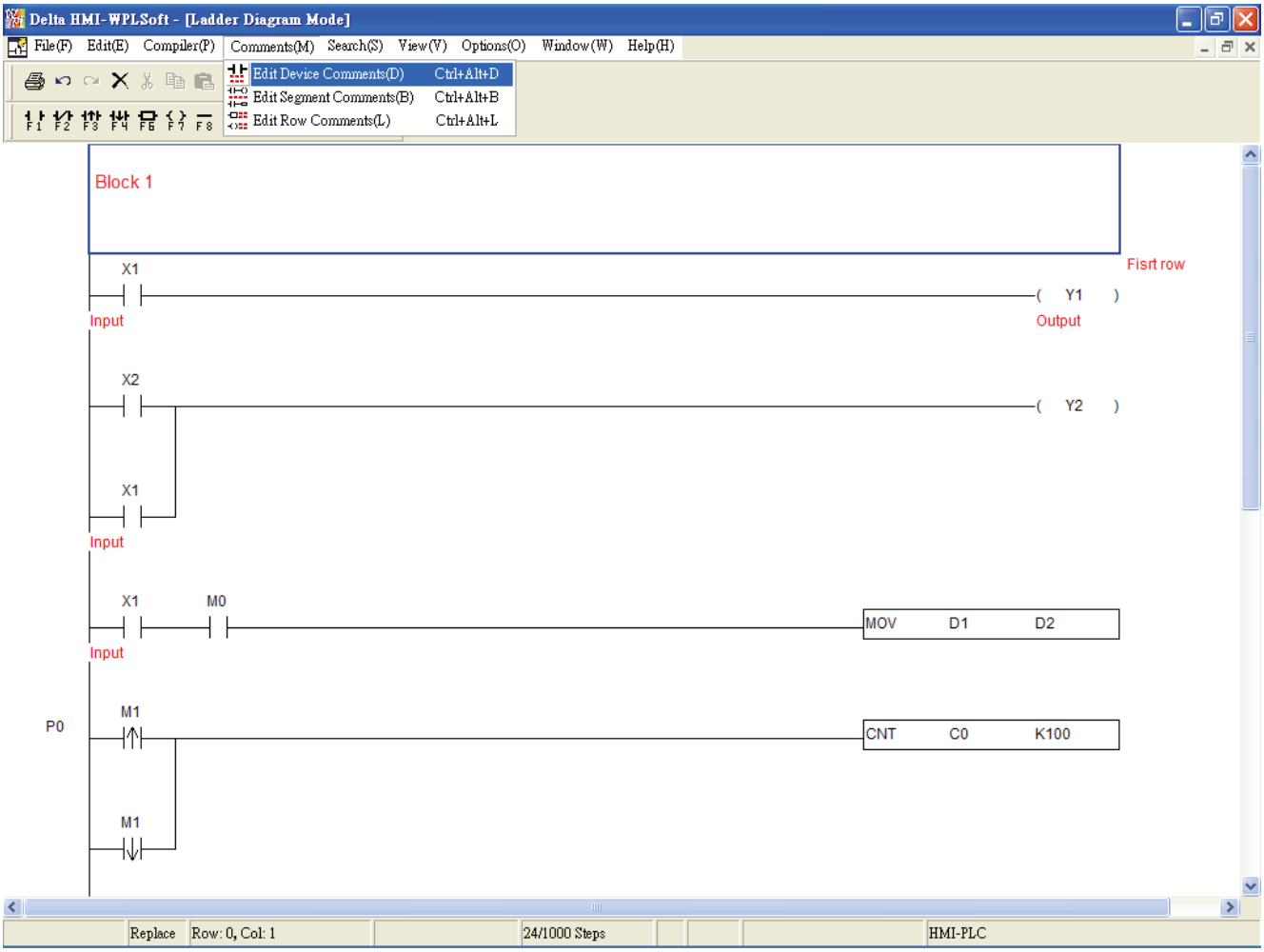
After editing the segment comments is completed, press the “OK” button to save the settings.

3.6 Edit Device Comments

In the Ladder Diagram Mode and Instruction Mode, the user can set the comments to be displayed in the device.

◈ Method 1:

1. First, choose to enter the Ladder Diagram Mode (or Instruction Mode). Move the editing block on the desired device. From the “Comments” menu, choose “Edit Device Comments(D)” or use the keyboard shortcuts by pressing keys (Ctrl) + (Alt) + (D).



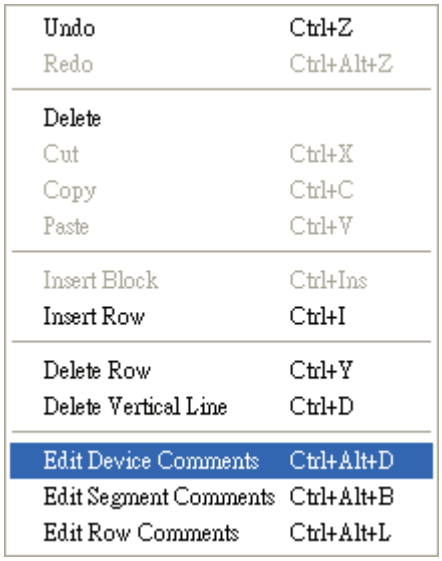
2. The Comment dialog box will appear and the user can edit comments for the desired device that the user chooses, e.g. M0 (If the chose device is the special M and D device, the user will see the preset comments shown in the Comment dialog box). After the device comments editing is completed, press “Enter” key on the keyboard or click the “OK” button by using the mouse.

3. If the user wants to display or hide device comments in the Ladder Diagram Mode, click the icon  on the toolbar or choose “Show Comments(M)” from “View(V)” menu.

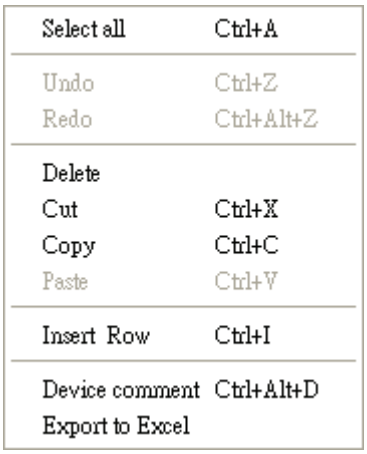
◈ Method 2:

1. Enter the Ladder Diagram Mode (or Instruction Mode). Move the editing block on the desired device (such as T64). Right click the mouse and then the following pop-up menu will appear on the screen.

In Ladder Diagram Mode



In Instruction Mode

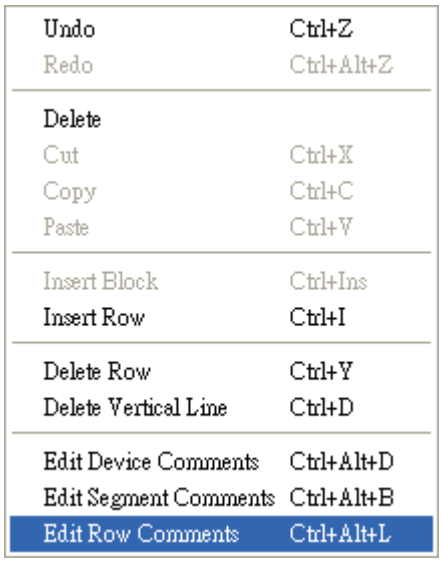


2. Choose “Edit Device Comments” from the pop-up menu, and the Comment dialog box will appear (see the figure below). Select the desired device, e.g. T64 and enter the comments for device T64. After the comments editing is completed, press “Enter” key on the keyboard or click the “OK” button by using the mouse.

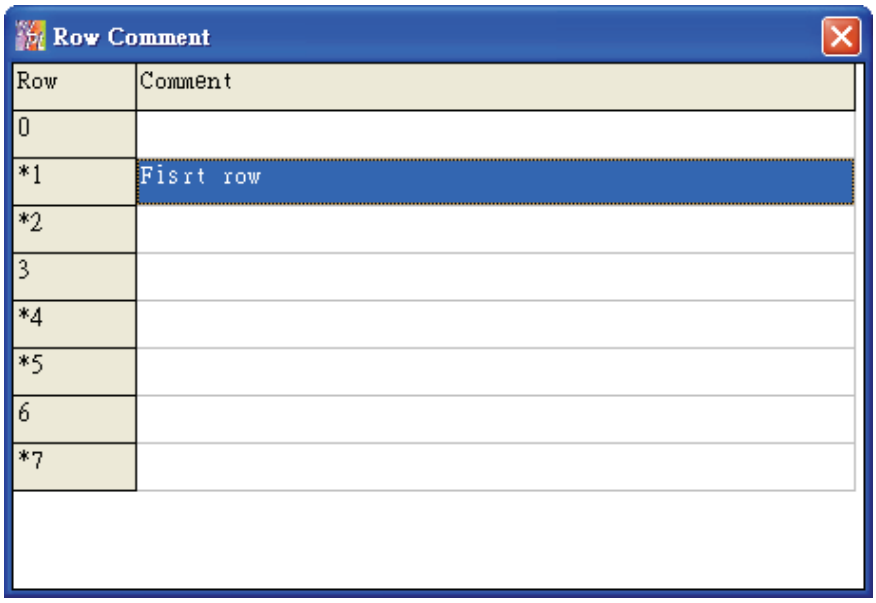
3.7 Edit Row Comments

◈ Method 1:

1. Move the editing block to the desired row. Right click the mouse and the pop-up menu below will appear. Select “Edit Row Comments” to add and edit comments into the row.

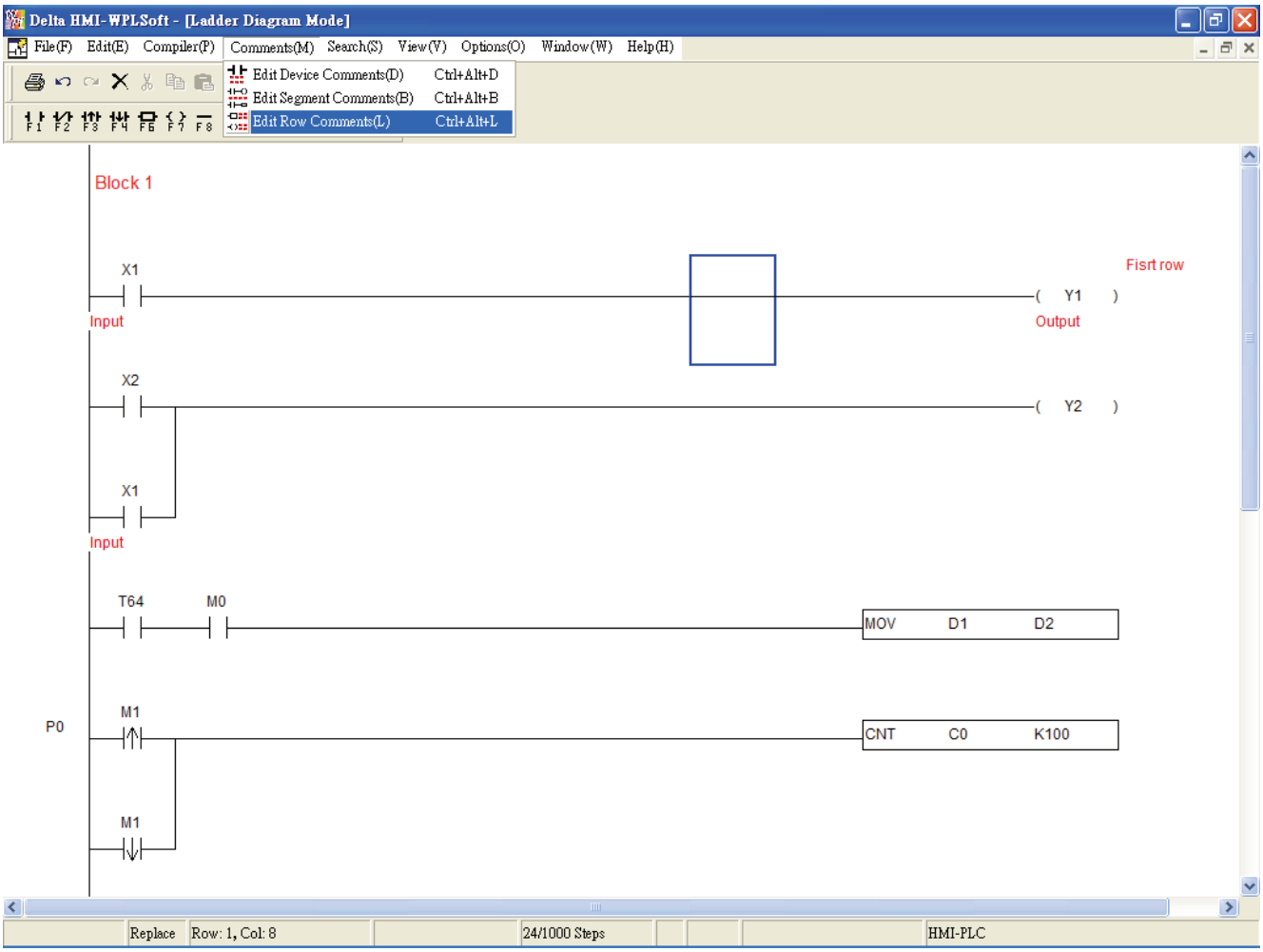


2. After clicking on “Edit Row Comments”, the following dialog box will appear. Then, the user can add and edit several row comments at the same time. After the comments editing is completed, close this dialog box to save the edited comments.



◈ Method 2:

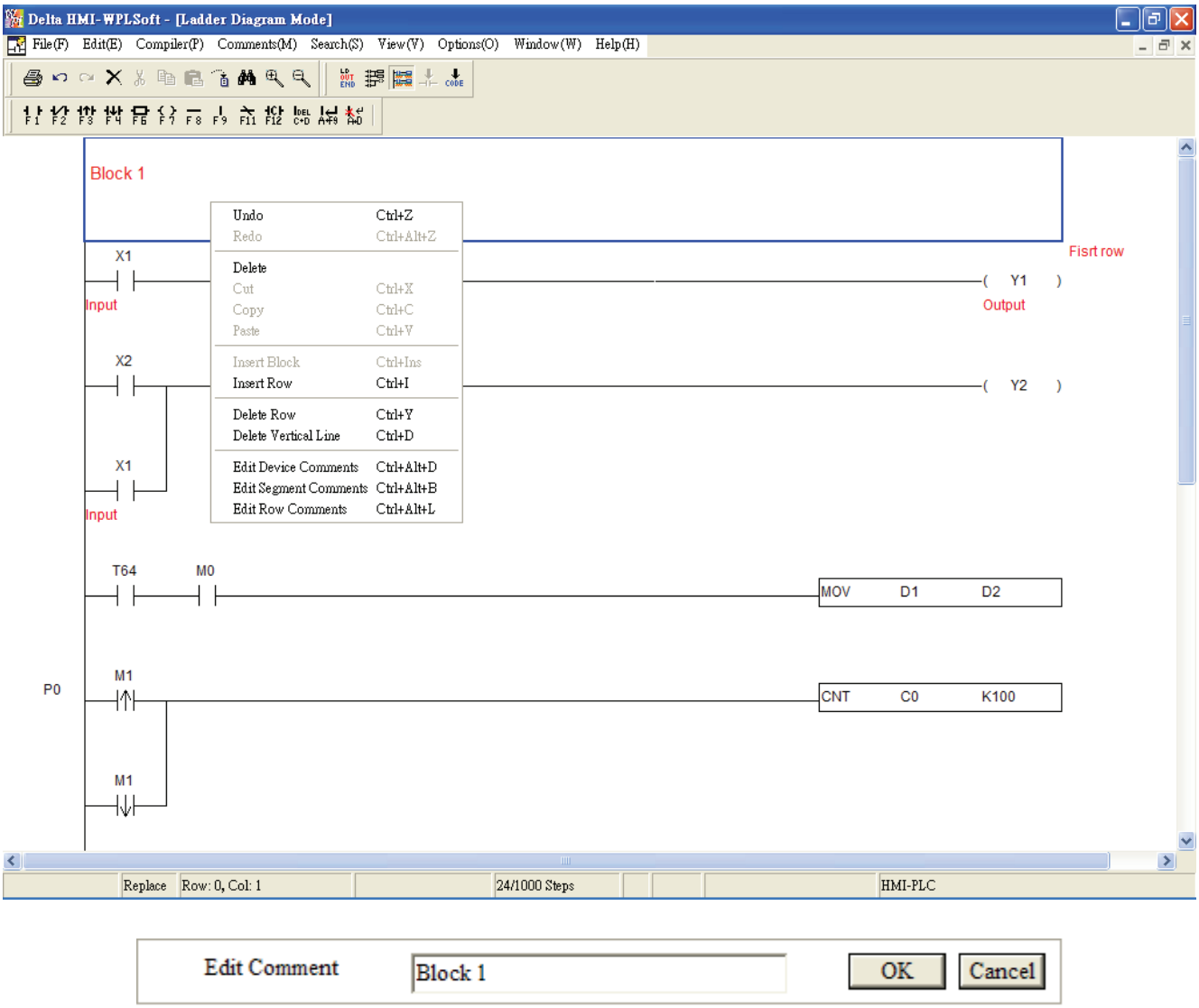
Move the editing block on the desired device. From “Comments” menu, choose “Edit Row comments(L)” or use the keyboard shortcuts by pressing keys (Ctrl) + (Alt) + (L). The “Row comment” dialog box will appear. Then, the user can enter the comments in each row. After the row comments editing is completed, close this dialog box to save the edited comments.



3.8 Segment Comments

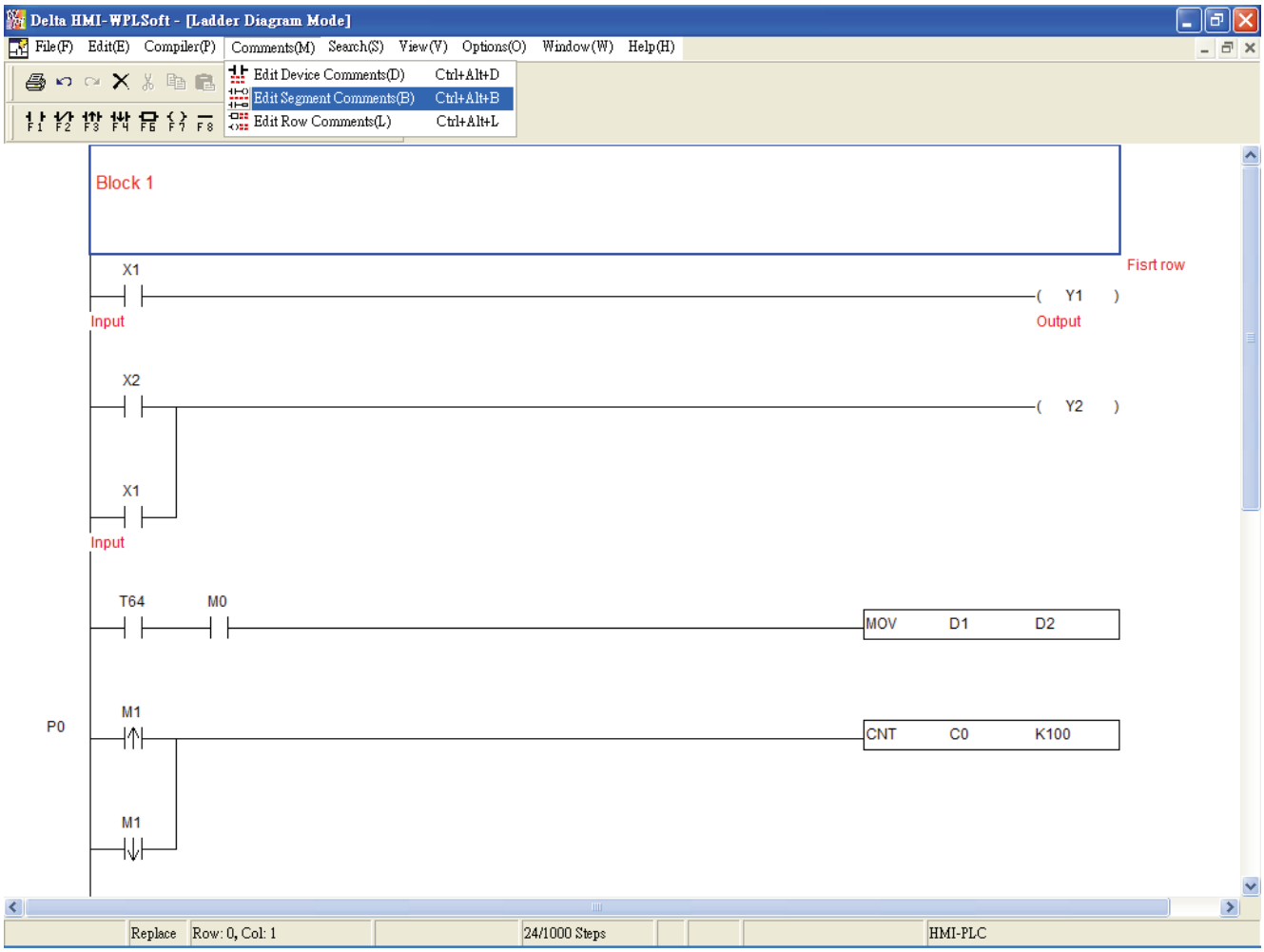
◈ Method 1:

Move the editing block to the blank area that the user wants to enter the segment comments (the user can also use the keyboard shortcuts by pressing keys (Ctrl) + (I) to insert a new row). Right click the mouse, and the pop-up menu in the following figure will appear. Then, choose “Edit Segment Comments” to enter the segment comments (60 characters maximum). Finally, press the “OK” button to complete the editing.



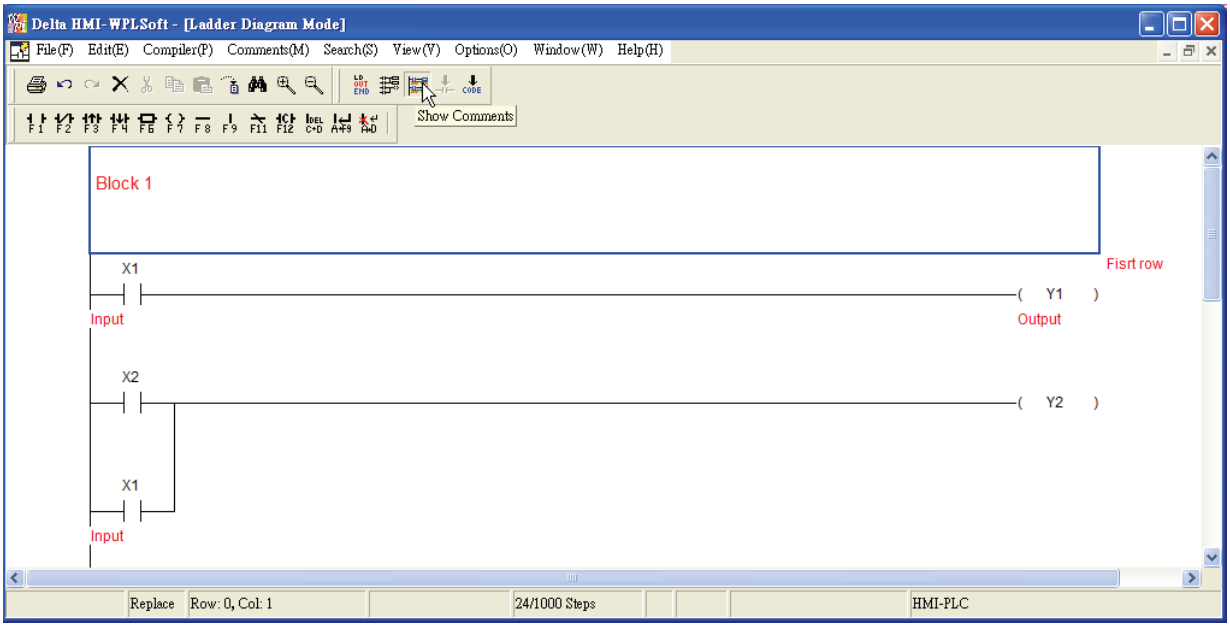
◈ Method 2:

Click “Comment” from the menu bar, and choose “Edit Segment Comments(B)” or use the keyboard shortcuts by pressing keys (Ctrl) + (Alt) + (B) to enter and edit the segment comments.



■ Show or Hide Comments

The user can show and hide the comment by clicking “View(V)” > “Show Comments(M)” or clicking the icon  on the toolbar. However, this function is provided for device comments and row comments only. The user cannot show and hide segment comments by using this function. When this function is enabled, the height of the ladder diagram will become higher in order to display the comments.


Delta HMI- WPLSoft - [Ladder Diagram Mode]
File(F) Edit(E) Compiler(P) Comments(M) Search(S) View(V) Options(O) Window(W) Help(H)
Show Comments
Block 1
X1
Input
Y1
Output
Fisrt row
X2
Y2
X1
Input
Replace
Row: 0, Col: 1
24/1000 Steps
HMI-PLC

Chapter 4 I/O Point Indicators

In the editing environment of ScrEdit (Screen Editor) programming software, the user can use digital input/output point indicators (hereinafter called “I/O point indicators”) to display the status of the input and output points and monitor the operation of DOP-EXIO series. Please refer to the Fig. 4.1 below.

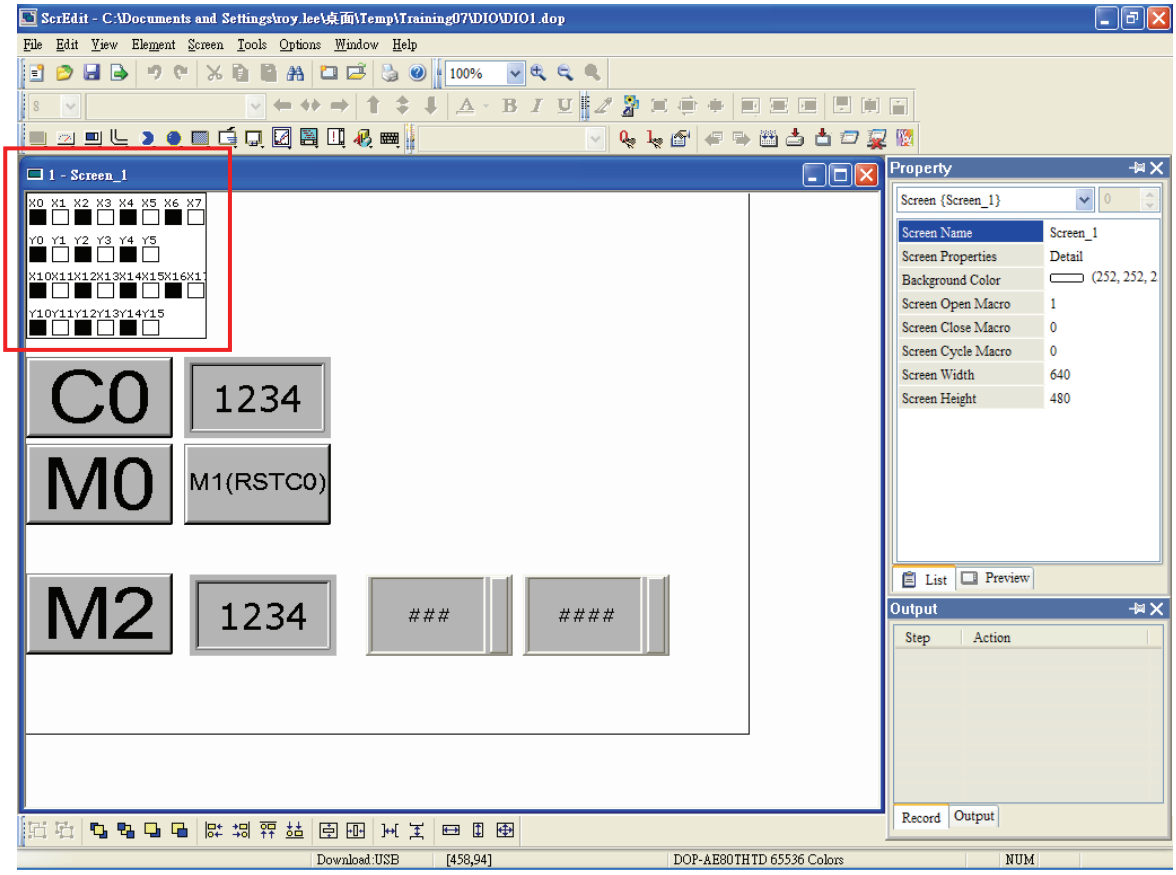


Fig. 4.1 Status of I/O point indicators

After opening the “Screen Properties” dialog box, which provides screen property settings for each screen, the user can set the settings of the I/O point indicators. Please refer to Fig. 4.2 in the following page.

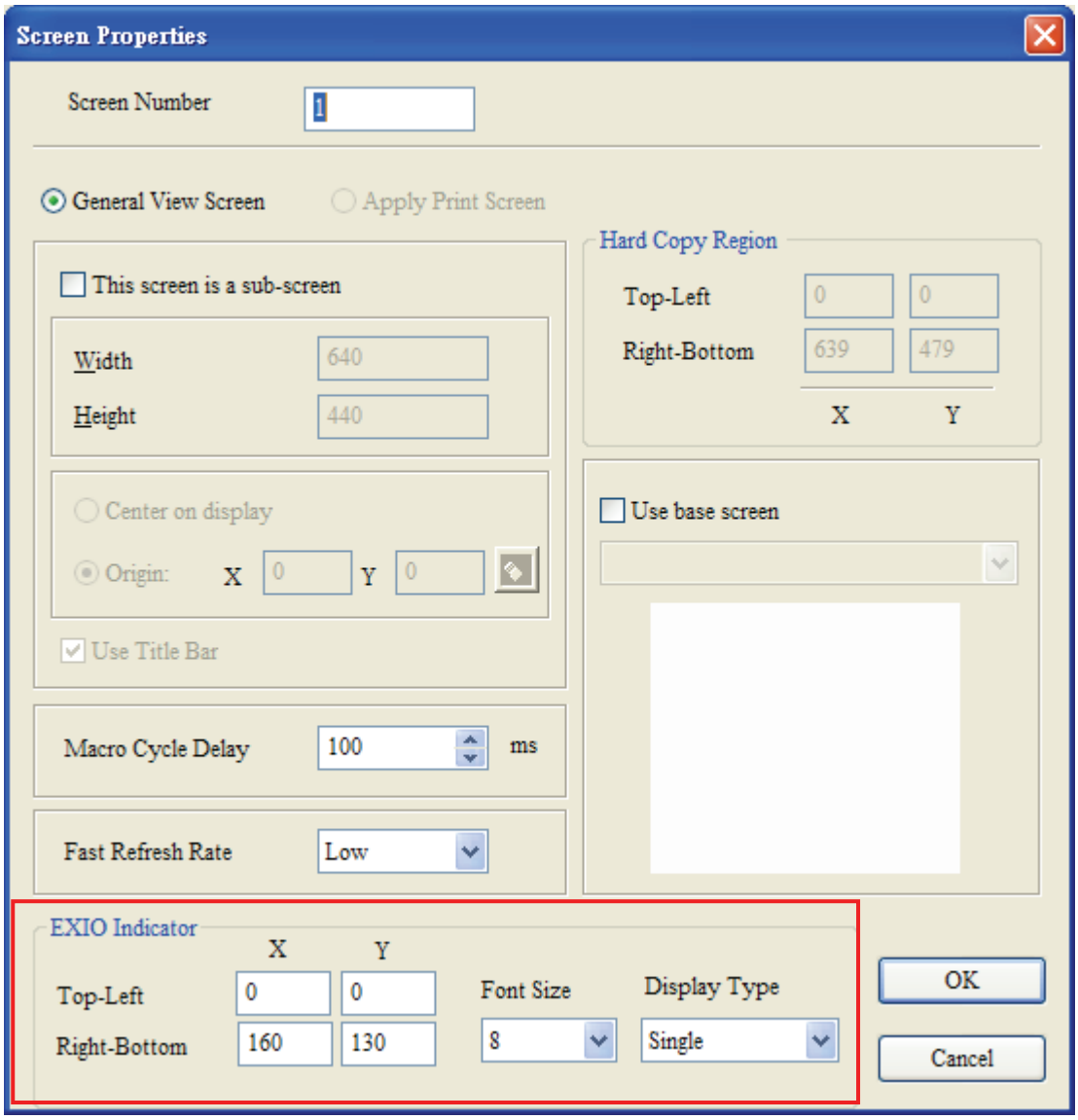


Fig. 4.2 “Screen Properties” dialog box

1. Display Position:
   Determine the position of the I/O point indicators by setting the coordinates of the Top-Left and Right-Bottom points.

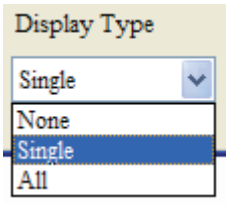
2. Font Size:
   Determine the font size of the font which displays in the I/O point indicators. (The available selection includes 8, 10, 12, 14, 16, 18, 20, 24, 28, 32, 40, 48, 64.)

3. Display Type:
   There are three kinds of display types: None, Single and All.
   The display of the indicators will change depending on the settings of the screen properties.

None: When the user selects this option, the indicators will not show on the screen.

Single: When the user selects this option, the indicators will display on a certain screen only.

All: When the user selects this option, the indicators will display on all screens.

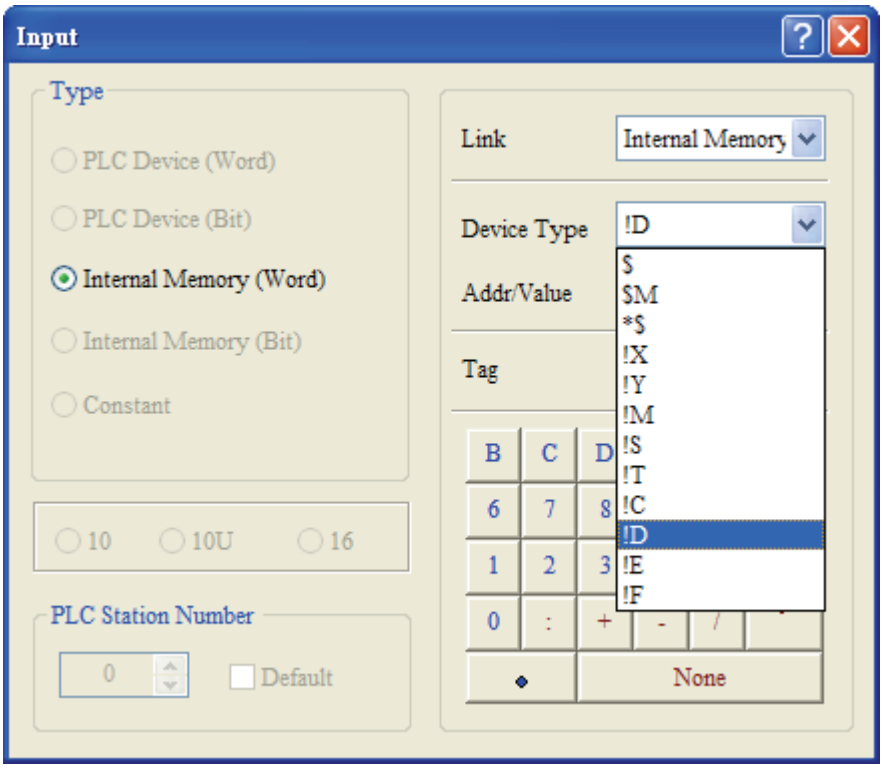


This page intentionally left blank.

Chapter 5 Internal Memory Address

After enabling the DOP-EXIO function, all the HMI elements can use DOP-EXIO series as internal memory addresses. The usage is the same as the usage of HMI internal memory $0 ~ $65535. Some of the internal memory addresses are “For latched”. For more introductions and the setting range of each device, please refer to Appendix A in this manual.

When the function for DOP-EXIO series is activated, the devices for DOP-EXIO series will appear in the “Device Type” drop-down menu shown in the “Internal Memory” selection.Some of the devices have special definitions and will be set or referred within some specific instructions. For more descriptions of the devices and instructions for DOP-EXIO series, please refer to Appendixes in this manual.



This page intentionally left blank.

Appendix A List of Devices

Type	Device	Item		Range		Function
Relay (bit)	X	Input relay		X0~X7, 8 points, octal	DOP-EXIO14RAE	Corresponds to external input point.
				X0~X17, 16 points, octal	DOP-EXIO28RAE	Corresponds to external input point.
	Y	Output relay		Y0~Y5, 6 points, octal	DOP-EXIO14RAE	Internal output point.
				Y0~Y5, Y10~Y15, 12 points, octal	DOP-EXIO28RAE	Internal output point.
	M	Auxiliary Relay	General puspose	M0~M511, M768~M999, 744 points; M1000~ M1279, 280 points[*2]	Total is 1,280 points	The contacts can be ON/OFF in the program.
			Latched[*1]	M512~M767, 256 points		
	T	Timer	100ms	T0~T63, 64 points	Total is 128 points	Timer indicated by TMR instruction. If timing reaches its target, the T contact of the same number will be On.
			10ms	T64~T126, 63 points		
			1ms	T127, 1 point		
	C	Counter	16-bit counting up	C0~C111, 112 points	Total is 128 points	Counter indicated by CNT (DCNT) instruction. If counting reaches its target, the C contact of the same number will be On.
				C112~C127, 16 points		
			32-bit counting up/down (Latched[*1])	C235,C236,C237,C238, C241,C242,C244,C246, C247,C249,C251,C252, C254, 13 points	Total is 13 points	
	S	Step point	Latched[*1]	S0~S127, 128 points	Total is 128 points	Used for step ladder diagram
Register (word data)	T	Present value of timer		T0~T127, 128 points		When the timing reaches the target, the contact of the timer will be On.

Type	Device	Item		Range		Function
Register (word data)	C	Present value of counter		C0~C127, 16-bit counter, 128 points C235,C236,C237,C238, C241,C242, C244, C246, C247,C249,C251,C252, C254, 32-bit counter, 13 points		When the ounting reaches the arget, the contact of the counter will be On.
	D	Data register	General purpose	D0~D407, 408 points	Total is 600 points	Memory area for data storage; E, F can be used for index indication.
			Latched*1	D408~D599, 192 points		
			Index indication	E, F, 2 points	Total is 2 points	
Pointer	N	For master control nested loop		N0~N7, 8 points		Control point for main control loop.
	P	For CJ, CALL instructions		P0~P63, 64 points		Position index for CJ and CALL.
Constant	K	Decimal form		K-32,768 ~ K32,767 (16-bit operation) K-2,147,483,648 ~ K2,147,483,647 (32-bit operation)		
	H	Hexadecimal form		H0000 ~ HFFFF (16-bit operation) H00000000 ~ HFFFFFFFF (32-bit operation)		

*1: The latched area is fixed and cannot be changed.

*2: M1000, M1001, M1002, M1003, M1020, M1021, M1022, M1067, M10068, and M1161 are the special auxiliary relays (special M).

Special Auxiliary Relay

The types and functions of special auxiliary relays (special M) are listed in the table below. Please be noted that the columns marked with “R” refers to “read only”, “and “R/W” refers to “read and write” and “-“ refers to the status remains unchanged.

Special M	Function	Power Off ⇩ Power On	STOP ⇩ RUN	RUN ⇩ STOP	Attribute	Latched	Default	Applicable Model
M1000	Monitoring normally open contact (A) *1	Off	On	Off	R	No	Off	DOP-EXIO14RAE DOP-EXIO28RAE
M1001	Monitoring normally closed contact (B) *2	On	Off	On	R	No	On	
M1002	Enabling positive pulses *3	Off	On	Off	R	No	Off	
M1003	Enabling negative pulses *4	On	Off	On	R	No	On	
M1020	Zero flag	Off	-	-	R	No	Off	
M1021	Borrow flag	Off	-	-	R	No	Off	
M1022	Carry flag	Off	-	-	R	No	Off	
M1067	Calculation error	Off	Off	-	R	No	Off	

Special M	Function	Power Off ⇩ Power On	STOP ⇩ RUN	RUN ⇩ STOP	Attribute	Latched	Default	Applicable Model
M1068	Calculation error locked	Off	-	-	R	No	Off	DOP-EXIO14RAE DOP-EXIO28RAE
M1161	8/16 bit mode switch On: in 8-bit mode	Off	-	-	R/W	No	Off	

*1: M1000 (A contact) is constantly “On” during operation and detection. When ELC is in RUN status, M1000 remains “On”.

*2: M1001 (B contact) is constantly “Off” during operation and detection. When ELC is in RUN status, M1001 remains “Off”

*3: M1002 is “On” during the first scan when ELC starts to RUN and remains “Off” afterward. The pulse width = 1 scan time. Use this contact for all kinds of initial settings. (On immediately when RUN).

*4: M1003 is “Off” during the first scan when ELC starts to RUN and remains “On” afterward. M1003 enables negative-direction pulses. (“Off” immediately when RUN)

This page intentionally left blank.

Appendix B List of Instructions

Available Instructions		
16-bit Instruction	32-bit Instruction	Function
LD	-	Loading in A contact
LDI	-	Loading in B contact
AND	-	Series Connection- A Contact
ANI	-	Series Connection- B Contact
OR	-	Parallel Connection- A Contact
ORI	-	Parallel Connection- B Contact
ANB	-	Series connection- loop blocks
ORB	-	Parallel connection- loop blocks
MPS	DMOV	Store the current result of the internal EXIO operations
MRD	DCML	Read the current result of the internal EXIO operations
ANDP	-	Rising-edge Series Connection
ANDF	DFMOV	Falling-edge Series Connection
ORP	DXCH	Rising-edge Parallel Connection
ORF	DBCD	Falling-edge Parallel Connection
PLS	DBIN	Rising-edge Output
PLF	DADD	Falling-edge Output
END	DSUB	Program End
NOP	DMUL	No Operation
INV	DRCL	Inverting Operation
P	-	Pointer
MOV	-	Move
CML	-	Compliment
BMOV	-	Block Move
FMOV	-	Fill Move
XCH	-	Exchange
BCD	-	Binary Coded Decimal
BIN	-	Binary
ADD	-	Addition
SUB	-	Subtraction
MUL	-	Multiplication
RCL	-	Rotation Left with Carry
SFTR	-	Bit Shift Right
SFTL	-	Bit Shift Left
ZRST	-	Zero Reset

Available Instructions		
16-bit Instruction	32-bit Instruction	Function
SUM	DSUM	Sum of Active Bits
BON	DBON	Check Specified Bit Status
MEAN	DMEAN	Mean
REF	-	Refresh
ALT	-	Alternate State
ASCI	-	Convert Hex to ASCII
AND=	DAND=	Series Connection Contact Compare =
AND>	DAND>	Series Connection Contact Compare >
AND<	DAND<	Series Connection Contact Compare <
AND<>	DAND<>	Series Connection Contact Compare <>
AND<=	DAND<=	Series Connection Contact Compare <=
AND>=	DAND>=	Series Connection Contact Compare >=
MPP	-	Pop (recall and remove) the currently stored result
OUT	-	Output Coil
SET	-	Latch（ON）
RST	-	Clear the contacts or the registers
TMR	-	16-bit Timer
CNT	DCNT	16-bit / 32-bit Counter
MC	-	Master Control Start
MCR	-	Master Control Reset
LDP	-	Rising-edge Detection Operation
LDF	-	Falling-edge Detection Operation
STL	-	Step Transition Ladder Start Command
RET	-	Step Transition Ladder Return Command
CJ	-	Conditional Jump
CALL	-	Call Subroutine
SRET	-	Subroutine Return
FEND	-	The End of the Main Program (First End)
FOR	-	Start of a FOR-NEXT Loop
NEXT	-	End of a FOR-NEXT Loop
CMP	DCMP	Compare
ZCP	DZCP	Zone Compare
DIV	DDIV	Division
INC	DINC	Increment
DEC	DDEC	Decrement
WAND	DAND	Logical Word AND
WOR	DOR	Logical Word OR

Available Instructions		
16-bit Instruction	32-bit Instruction	Function
WXOR	DXOR	Logical Exclusive OR
NEG	DNEG	2's Complement (Negative)
ROR	DROR	Rotation Right
ROL	DROL	Rotation Left
RCR	DRCR	Rotation Right with Carry
HEX	-	Convert ASCII to Hex
ABS	DABS	Absolute Value
SWAP	DSWAP	Byte Swap
LD=	DLD=	Load Contact Compare =
LD>	DLD>	Load Contact Compare >
LD<	DLD<	Load Contact Compare <
LD<>	DLD<>	Load Contact Compare <>
LD<=	DLD<=	Load Contact Compare <=
LD>=	DLD>=	Load Contact Compare >=
OR=	DOR=	Parallel Connection Contact Compare =
OR>	DOR>	Parallel Connection Contact Compare >
OR<	DOR<	Parallel Connection Contact Compare <
OR<>	DOR<>	Parallel Connection Contact Compare <>
OR<=	DOR<=	Parallel Connection Contact Compare <=
OR>=	DOR>=	Parallel Connection Contact Compare >=

This page intentionally left blank.

Appendix C Use of Basic Instructions

Mnemonic	Functions
LD	Loading in A contact

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanation:

The LD instruction is used on the A contact that has its start from the left BUS or the A contact that is the start of a contact circuit. The functions are to save the present contents and store the acquired contact status into the accumulative register.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LD	**X0**	Loading in contact A of X0
AND	X1	Connecting to contact A of X1 in series
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
LDI	Loading in B contact

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanation:

The LDI instruction is used on the B contact that has its start from the left BUS or the B contact that is the start of a contact circuit. The functions are to save the present contents and store the acquired contact status into the accumulative register.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LDI	**X0**	Loading in contact B of X0
AND	X1	Connecting to contact A of X1 in series
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
AND	Series Connection- A Contact

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanation:

The AND instruction is used in the series connection of A contact. The functions are to read out the status of present specific series connection contacts and perform the “AND” operation with the logical operation result obtained. The final result will be store in the accumulative register.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LDI	X1	Loading in contact B of X1
AND	**X0**	Connecting to contact A of X0 in series
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
ANI	Series Connection- B Contact

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanation:

The ANI instruction is used in the series connection of B contact. The functions are to read out the status of present designated series connection contacts and perform the “AND” operation with the logical operation result obtained. The final result will be store in the accumulative register.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LD	X1	Loading in contact A of X1
ANI	**X0**	Connecting to contact B of X0 in series
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
OR	Parallel Connection- A Contact

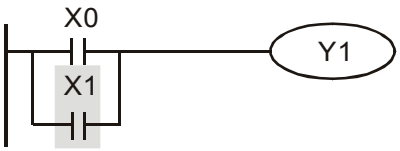
Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanation:

The OR instruction is used in the parallel connection of A contact. The functions are to read out the status of present designated parallel connection contacts and perform the “OR” operation with the logical operation result obtained. The final result will be store in the accumulative register.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in contact A of X0
OR	**X1**	Connecting to contact A of X1 in parallel
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
ORI	Parallel Connection- B Contact

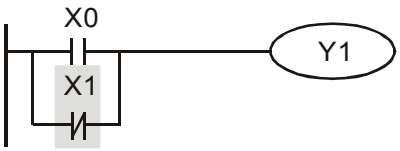
Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanation:

The ORI instruction is used in the parallel connection of B contact. The functions are to read out the status of present designated parallel connection contacts and perform the “ORI” operation with the logical operation result obtained. The final result will be store in the accumulative register.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in contact A of X0
ORI	**X1**	Connecting to contact B of X1 in parallel
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
ANB	Series connection- loop blocks

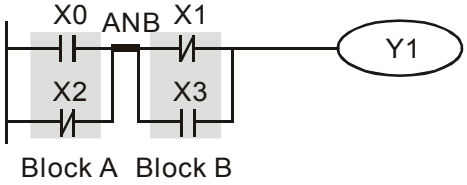
Operand	none

Explanation:

To perform the “AND” operation of the preserved logic results and content in the accumulative register.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in contact A of X0
ORI	X2	Connecting to contact B of X2 in parallel
LDI	X1	Loading in contact B of X1
OR	X3	Connecting to contact A of X3 in parallel
ANB		Connecting circuit block in series
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
ORB	Parallel connection- loop blocks

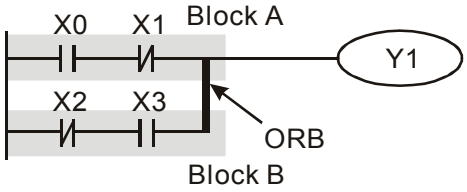
Operand	None

Explanation:

To perform the “OR” operation of the preserved logic results and content in the accumulative register.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in contact A of X0
ANI	X1	Connecting to contact B of X1 in series
LDI	X2	Loading in contact B of X2
AND	X3	Connecting to contact A of X3 in series
ORB		Connecting circuit block in parallel
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
MPS	Store the current result of the internal EXIO operations

Operand	None

Explanation:

To save the content in the accumulative register into the operational result (the pointer of operational result will plus 1).

Mnemonic	Functions
MRD	Read the current result of the internal EXIO operations

Operand	None

Explanation:

To read the operational result and store it into the accumulative register (the pointer of operational result stays intact).

Mnemonic	Functions
MPP	Pop (recall and remove) the currently stored result

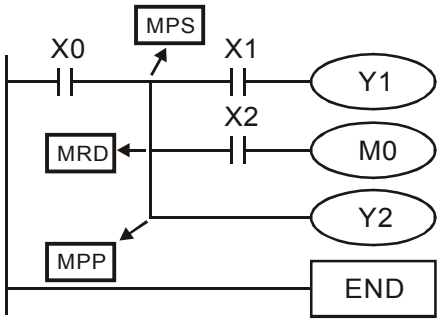
Operand	None

Explanation:

To retrieve the previous preserved logical operation result and store it into the accumulative register (the pointer of operational result will minus 1).

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in contact A of X0
MPS		Saving into stack
AND	X1	Connecting to contact A of X1 in series
OUT	Y1	Driving Y1 coil
MRD		Reading from stack
AND	X2	Connecting to contact A of X2 in series
OUT	M0	Driving M0 coil
MPP		Reading from stack and pop pointer
OUT	Y2	Driving Y2 coil
END		Program ends

Mnemonic	Functions
OUT	Output Coil

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	-	✓	✓	✓	-	-	-

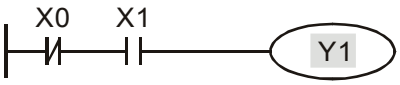
Explanations:

1. To output the logical operation result before OUT instruction into a designated device.
2. Actions of coil contact:

Operational result	OUT instruction		
	Coil	Contact	
		A contact (normally open)	B contact (normally closed)
FALSE	Off	Off	On
TRUE	On	On	Off

Program Example:

Ladder diagram:



Instruction code:		Operation:
LDI	X0	Loading in contact B of X0
AND	X1	Connecting to contact A of X1 in series
OUT	**Y1**	Driving Y1 coil

Mnemonic	Functions
SET	Latch (ON)

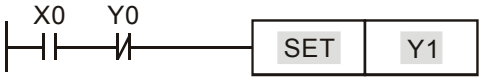
Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	-	✓	✓	✓	-	-	-

Explanations:

When the SET instruction is driven, its designated device will be “On” and keep being On both when SET instruction is still being driven or not driven. Use RST instruction to set “Off” the device.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in contact A of X0
ANI	Y0	Connecting to contact B of Y0 in series
SET	**Y1**	Y1 latched (On)

Mnemonic	Functions
RST	Clear the contact or the registers

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599	E, F
	-	✓	✓	✓	✓	✓	✓	✓

Explanations:

1. When the RST instruction is driven, the actions of the designated devices are:

Device	Status
Y, M, S,	Coil and contact will be set to “Off”
T, C	Present values of the timer or counter will be set to “0”, and the coil and contact will be set to “Off”
D, E, F	The content will be set to “0”.

2. If RST instruction is not being executed, the status of the designated device will stay intact.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LD	X0	Loading in contact A of X0
RST	**Y5**	Resetting contact Y5

Mnemonic	Functions
TMR	16-bit Timer

Operand		
	T-K	T0~T127, K0~K32,767
	T-D	T0~T127, D0~D599

Explanations:

When TMR instruction is executed, the designated coil of the timer will be On and the timer will start to time. When the set value in the timer is reached (present ≥ set value), the contact will be:

NO (Normally Open) contact	Open collector
NC (Normally Closed) contact	Close collector

Program Example:

Ladder diagram:

Instruction code:		Operation:
LD	X0	Loading in contact A of X0 T5 timer
TMR	**T5 K1000**	Set value in timer T5 as K1,000

Mnemonic	Functions
CNT	16-bit Counter

Operand	C-K	C0~C127, K0~K32,767
	C-D	C0~C127, D0~D599

Explanations:

1. When the CNT instruction goes from Off to On, the designated counter coil will be driven, and the present value in the counter will plus 1. When the counting reaches the set value (present value = set value), the contact will be:

NO (Normally Open) contact	Open collector
NC (Normally Closed) contact	Close collector

2. If there are other counting pulse input after the counting reaches its target, the contact and present value will stay intact. Use RST instruction to restart or reset the counting.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LD	X0	Loading in contact A of X0
CNT	**C20 K100**	Set value in counter C20 as K100

Mnemonic	Functions
DCNT	32-bit Counter

Operand	C-K	C235~C254
	C-D	C235~C254, D0~D598

Explanations:

1. DCNT is the instruction for enabling the 32-bit high-speed counters C235 ~ C254. The method of

using DCNT instruction is the same as using CNT instruction to enabling C0~C127.

2. When DCNT is Off, the counting will stop, but the existing present value in the counter will not be cleared. To clear the present value and the contact, the user has to use the instruction RST C2XX.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LD	M0	Loading in contact A of M0
DCNT	**C254**	Set value of counter C254 as K1,000
	K1000	

Mnemonic	Functions
MC / MCR	Master Control Start / Reset

Operand	N0~N7

Explanations:

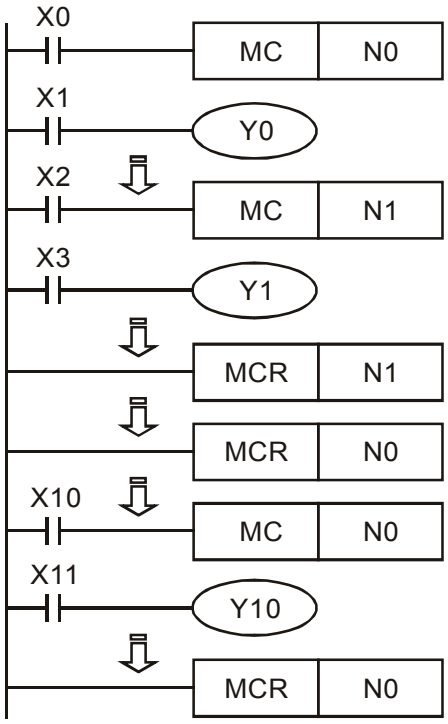
1. MC is the main-control start instruction. When MC instruction is executed, the execution of instructions between MC and MCR will not be interrupted. When MC instruction is Off, the actions of the instructions between MC and MCR are:

Instruction type	Explanation
General purpose timer	Present value = 0 Coil is Off, No action for the contact
Accumulative timer	Coil is Off, present value and contact stay intact
Subroutine timer	Present value = 0 Coil is Off, No action for the contact
Counter	Coil is Off, present value and contact stay intact
Coils driven by OUT instruction	All Off
Devices driven by SET and RST instructions	Stay intact
Application instructions	All disabled. The FOR-NEXT nested loop will still execute back and forth for N times. Instructions between FOR-NEXT will act as the instructions between MC and MCR.

2. MCR is the main-control end instruction that is placed in the end of the main-control program. There should not be any contact instructions prior to MCR instruction.
3. MC-MCR main-control program instructions support the nested program structure (max. 8 layers) and please use the instruction in the order N0 ~ N7.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in A contact of X0
MC	**N0**	Enabling N0 common series connection contact
LD	X1	Loading in A contact of X1
OUT	Y0	Driving Y0 coil
:		
LD	X2	Loading in A contact of X2
MC	**N1**	Enabling N1 common series connection contact
LD	X3	Loading in A contact of X3
OUT	Y1	Driving Y1 coil
:		
MCR	**N1**	Disabling N1 common series connection contact
:		
MCR	**N0**	Disabling N0 common series connection contact
:		
LD	X10	Loading in A contact of X10
MC	**N0**	Enabling N0 common series connection contact
LD	X11	Loading in A contact of X11
OUT	Y10	Driving Y10 coil
:		
MCR	**N0**	Disabling N0 common series connection contact

Mnemonic	Functions
LDP	Rising-edge Detection Operation

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanations:

The method of using LDP is the same as using LD, but the actions of the two instructions differ. LDP saves the current content and store the detected status of rising-edge to the accumulative register.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LDP	**X0**	Starting X0 rising-edge detection
AND	X1	Series connecting A contact of X1
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
LDF	Falling-edge Detection Operation

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanations:

The method of using LDF is the same as using LD, but the actions of the two instructions differ. LDF saves the current content and store the detected status of falling-edge to the accumulative register.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LDF	**X0**	Starting X0 falling-edge detection
AND	X1	Series connecting A contact of X1
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
ANDP	Riding-edge Series Connection

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanations:

ANDP instruction is used in the series connection of the contacts' rising-edge detection.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LD	X0	Loading in A contact of X0
ANDP	**X1**	X1 rising-edge detection in series connection
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
ANDF	Falling-edge Series Connection

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanations:

ANDF instruction is used in the series connection of the contacts' falling-edge detection.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LD	X0	Loading in A contact of X0
ANDF	**X1**	X1 falling-edge detection in series connection
OUT	Y1	Drive Y1 coil

Mnemonic	Functions
ORP	Rising-edge Parallel Connection

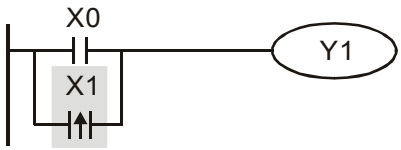
Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanations:

The ORP instructions are used in the parallel connection of the contact's rising-edge detection.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in A contact of X0
ORP	**X1**	X1 rising-edge detection in parallel connection
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
ORF	Falling-edge Parallel Connection

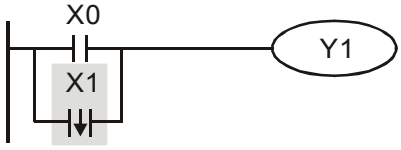
Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	✓	✓	✓	✓	✓	✓	-

Explanations:

The ORF instructions are used in the parallel connection of the contact's falling-edge detection.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in A contact of X0
ORF	**X1**	X1 falling-edge detection in parallel connection
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
PLS	Rising-edge Output

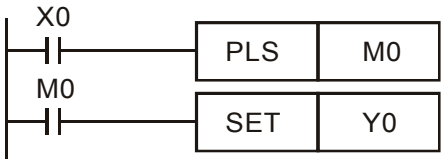
Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	-	✓	✓	-	-	-	-

Explanations:

When X0 goes from Off to On (rising-edge trigger), PLS instruction will be executed and M0 will send out pulses for once of 1 scan time.

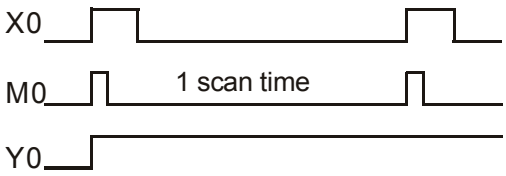
Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in A contact of X0
PLS	**M0**	M0 rising-edge output
LD	M0	Loading in contact A of M0
SET	Y0	Y0 latched (On)

Timing Diagram:



Mnemonic	Functions
PLF	Falling-edge Output

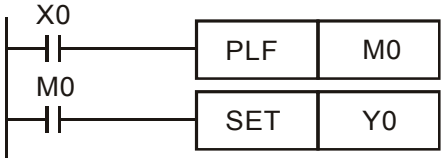
Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	-	✓	✓	-	-	-	-

Explanations:

When X0 goes from On to Off (falling-edge trigger), PLF instruction will be executed and M0 will send out pulses for once of 1 scan time.

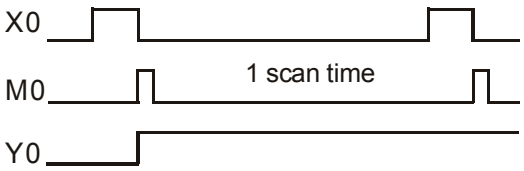
Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in A contact of X0
PLF	**M0**	M0 falling-edge output
LD	M0	Loading in contact A of M0
SET	Y0	Y0 latched (On)

Timing Diagram:



Mnemonic	Functions
END	Program End

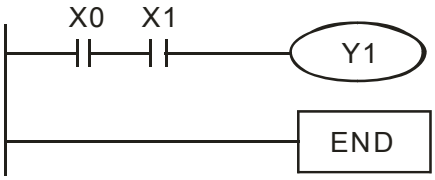
Operand	None

Explanations:

END instruction has to be placed in the end of a ladder diagram or instruction program. DOP-EXIO series will start to scan from address 0 to END instruction and return to address 0 to restart the scan.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in B contact of X0
AND	X1	Series connecting A contact of X1
OUT	Y1	Driving Y1 coil
END		Program end

Mnemonic	Functions
NOP	No Operation

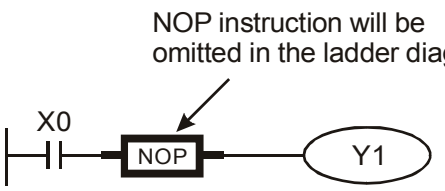
Operand	None

Explanations:

NOP instruction does not conduct any operations in the program; therefore, after the execution of NOP, the existing logical operation result will be kept. If the user wants to delete a certain instruction without altering the length of the program, the user can use NOP instruction. If the user wants to delete a certain instruction temporarily, the user can also use NOP instruction.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in B contact of X0
NOP		No operation
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
INV	Inverting Operation

Operand	None

Explanations:

The logical operation result before INV instruction will be inverted and stored in the accumulative register.

Program Example:

Ladder diagram:

Instruction code:		Operation:
LD	X0	Loading in A contact of X0
INV		Inverting the operation result
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
P	Pointer

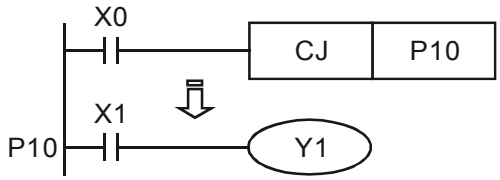
Operand	P0~P63

Explanations:

Pointer P is used in 00 CJ and 01 CALL instructions. The use of P does not need to start from No. 0, and the No. of P cannot be repeated; otherwise, unexpected errors may occur.

Program Example:

Ladder diagram:



Instruction code:		Operation:
LD	X0	Loading in A contact of X0
CJ	P10	From instruction CJ to P10
:		
P10		Pointer P10
LD	X1	Loading in A contact of X1
OUT	Y1	Driving Y1 coil

Mnemonic	Functions
STL	Step Transition Ladder Start Command

Operand	X0~X17	Y0~Y17	M0~M1279	S0~S127	T0~T127	C0~C254	D0~D599
	-	-	-	✓	-	-	-

Explanations:

STL Sn constructs a step. When STL instruction appears in the program, the program will enter a step ladder diagram status controlled by steps. The initial status has to start from S0 ~ S9. RET instruction

indicates the end of a step ladder diagram starting from S0 ~ S9 and the bus returns to a normal ladder diagram instruction. The No. of S cannot be repeated.

Mnemonic	Functions
RET	Step Transition Ladder Return Command

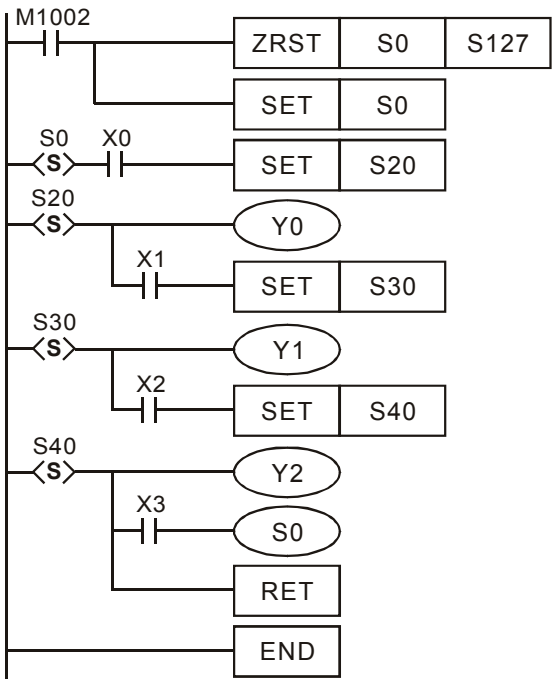
Operand	None

Explanations:

RET indicates the end of a step. There has to be a RET instruction in the end of a series of steps. One EXIO program can be written in maximum 10 steps (S0 ~ S9) and every step should end with a RET.

Program Example:

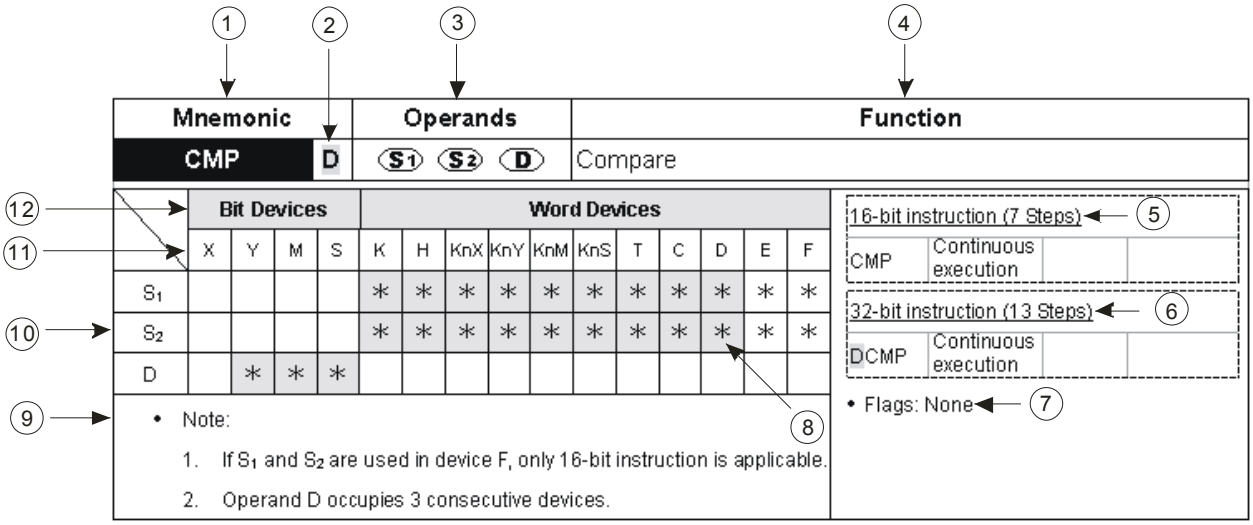
Ladder diagram:



This page intentionally left blank.

Appendix D Use of Application Instructions

■ Format of an application instruction:



① Mnemonic of an application instruction.

② Indication of if there is a 16-bit or 32-bit instruction. If there is a 32-bit instruction, the column will be marked with "**D**".

③ Operands

④ Function of the application instruction

⑤ Steps occupied by the 16-bit execution instruction

⑥ Steps occupied by the 32-bit execution instruction

⑦ Related flags for the application instruction

⑧ Column marked with * and in grey refers to E, F index register modification is applicable.

⑨ Note

⑩ Column marked with * is the device applicable for the operand

⑪ Device name

⑫ Device type

Mnemonic	Operands	Function
CJ	(S)	Conditional Jump

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F

- Note:
  1. Operand S can designate P.
  2. P can be modified by index register E, F.

16-bit instruction (3 Steps)

CJ	Continuous execution

32-bit instruction

- - - -

- Flags: None

Operands:

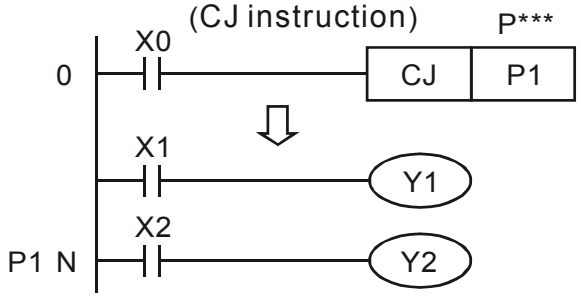
(S): The destination pointer of conditional jump

Explanations:

1. When the user does not wish a particular part of DOP-EXIO program in order to shorten the scan time and execute dual outputs, CJ instruction or CJP instruction can be adopted.
2. When the program designated by pointer P is prior to CJ instruction, WDT timeout will occur and DOP-EXIO will stop running. Please use it carefully.
3. CJ instruction can designate the same pointer P repeatedly. However, CJ and CALL cannot designate the same pointer P; otherwise an error will occur.

Program Example 1:

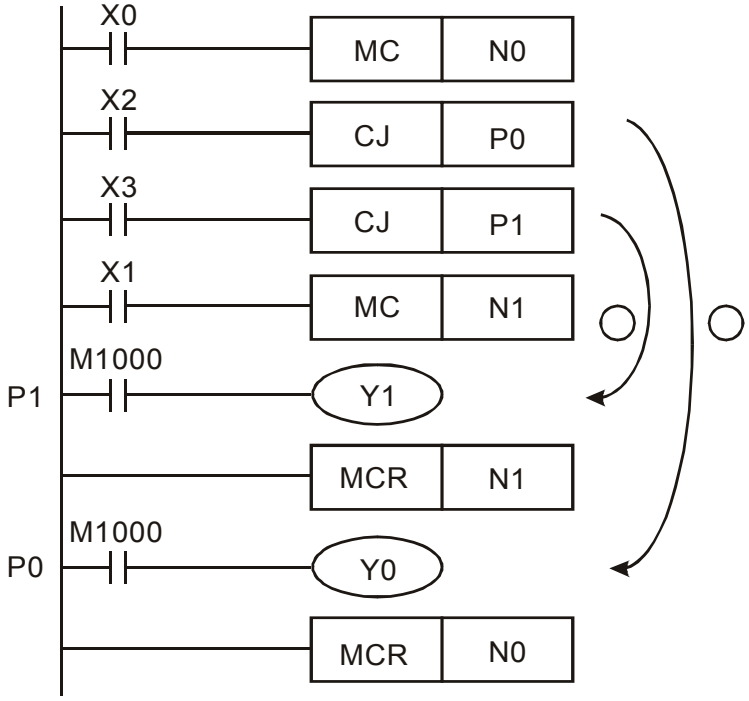
1. When X0 = On, the program automatically jumps from address 0 to N (the designated label P1) and keeps its execution. The addresses between 0 and N will not be executed.
2. When X0 = Off, as an ordinary program, the program keeps on executing from address 0. CJ instruction will not be executed at this time.



Program Example 2:

1. CJ instruction can be used in the following 5 conditions between MC and MCR instructions.
   a) Without MC ~ MCR.
   b) From without MC to within MC. Valid in the loop P1 as shown in the figure below.
   c) In the same level N, inside of MC~MCR.
   d) From within MC to without MCR.
   e) Jumping from this MC ~ MCR to another MC ~ MCR

2. When CJ instruction is used between MC and MCR, it can only be applied without MC ~ MCR or in the same N layer of MC ~ MCR. Jumping from this MC ~ MCR to another MC ~ MCR will result in errors, i.e. a) and c) as stated above can ensure correct actions; others will cause errors.

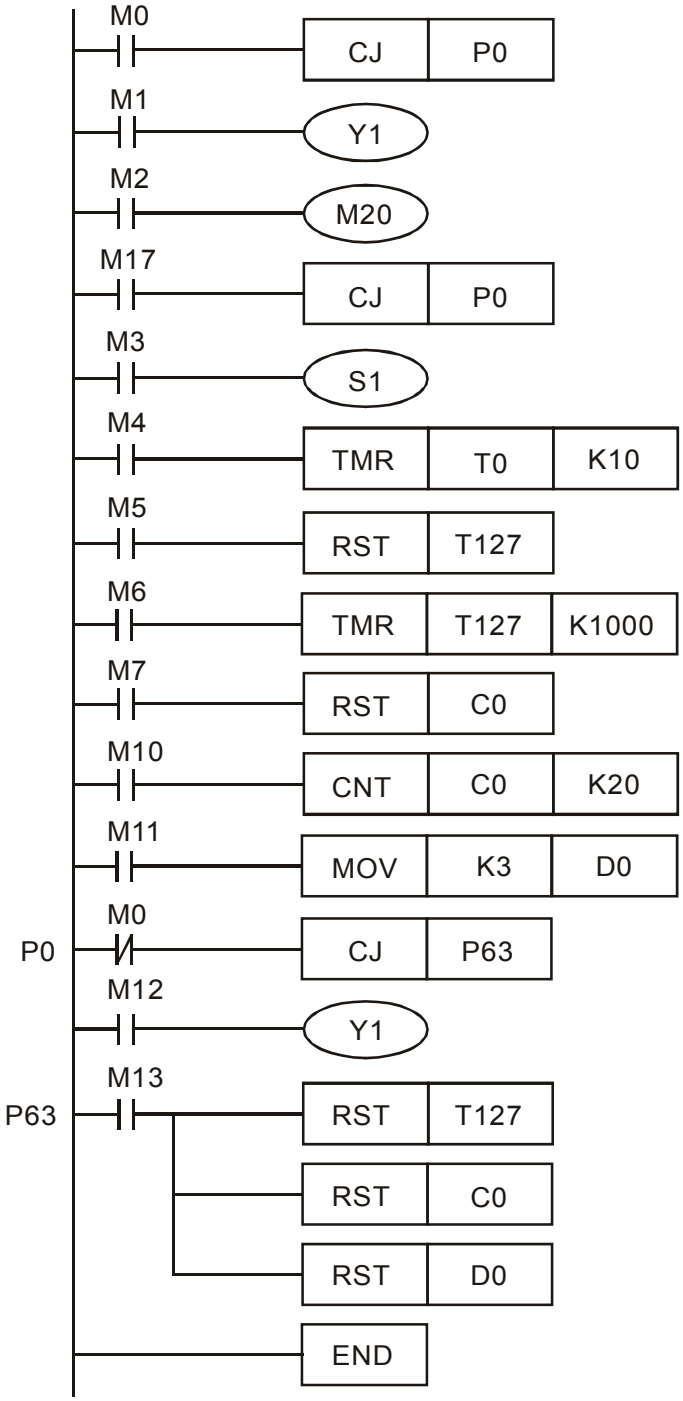


Program Example 3:

1. The status of each device when executing CJ instruction:

   - The method of using this CJ instruction is similar to the method of using goto instruction of C-language. When executing CJ instruction, the status of each device will not be changed.
   - When the timers are driven and encounter the execution of CJ instruction, the timing will resume. After the timing target is reached, the output contact of the timer will be On.
   - The counter will stop counting (This is because the counter is activated to count via the software).
   - All the instructions which have encounterd the execution of CJ instruction will not be activated.

2. Y1 is a dual output. When M0 = Off, Y1 is controlled by M1. When M0 = On, Y1 is controlled by M12.


M0
CJ
P0
M1
Y1
M2
M20
M17
CJ
P0
M3
S1
M4
TMR
T0
K10
M5
RST
T127
M6
TMR
T127
K1000
M7
RST
C0
M10
CNT
C0
K20
M11
MOV
K3
D0
M0
P0
CJ
P63
M12
Y1
M13
P63
RST
T127
RST
C0
RST
D0
END

Mnemonic	Operands	Function
CALL	(S)	Call Subroutine

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F

- Note:
  1. Operand S can designate P.
  2. P can be modified by index register E, F.

16-bit instruction (3 Steps)

CALL	Continuous execution

32-bit instruction

-	-	-	-

- Flags: None

Operands:

(S): The pointer of call subroutine.

Explanations:

1. Edit the subroutine designated by the pointer after FEND instruction.
2. The number of pointer P, when used by CALL, cannot be the same as the number designated by CJ instruction.
3. If only CALL instruction is in use, it can call subroutines of the same pointer number with no limit on times.
4. Subroutine can be nested for 5 levels including the initial CALL instruction. (If entering the sixth level, the subroutine won't be executed.)

Mnemonic	Operands	Function
SRET	None	Subroutine Return

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F

- Note:
  1. No operand.
  2. No contact to drive the instruction is required.

16-bit instruction (1 Step)

SRET	Continuous execution	-	-

32-bit instruction

-	-	-	-

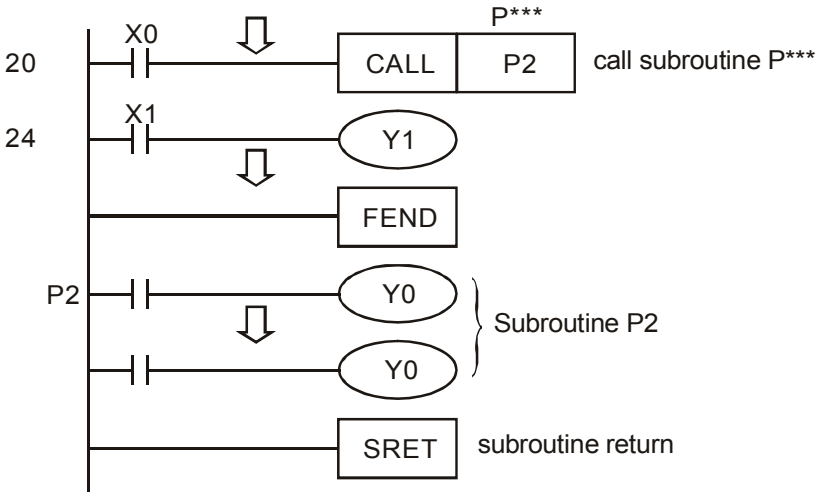
- Flags: None

Explanations:

1. This instruction denotes the end of the subroutine program.
2. The subroutine will return to main program by SRET after the termination of subroutine and execute the sequence program located at the next step to the CALL instruction.

Program Example 1:

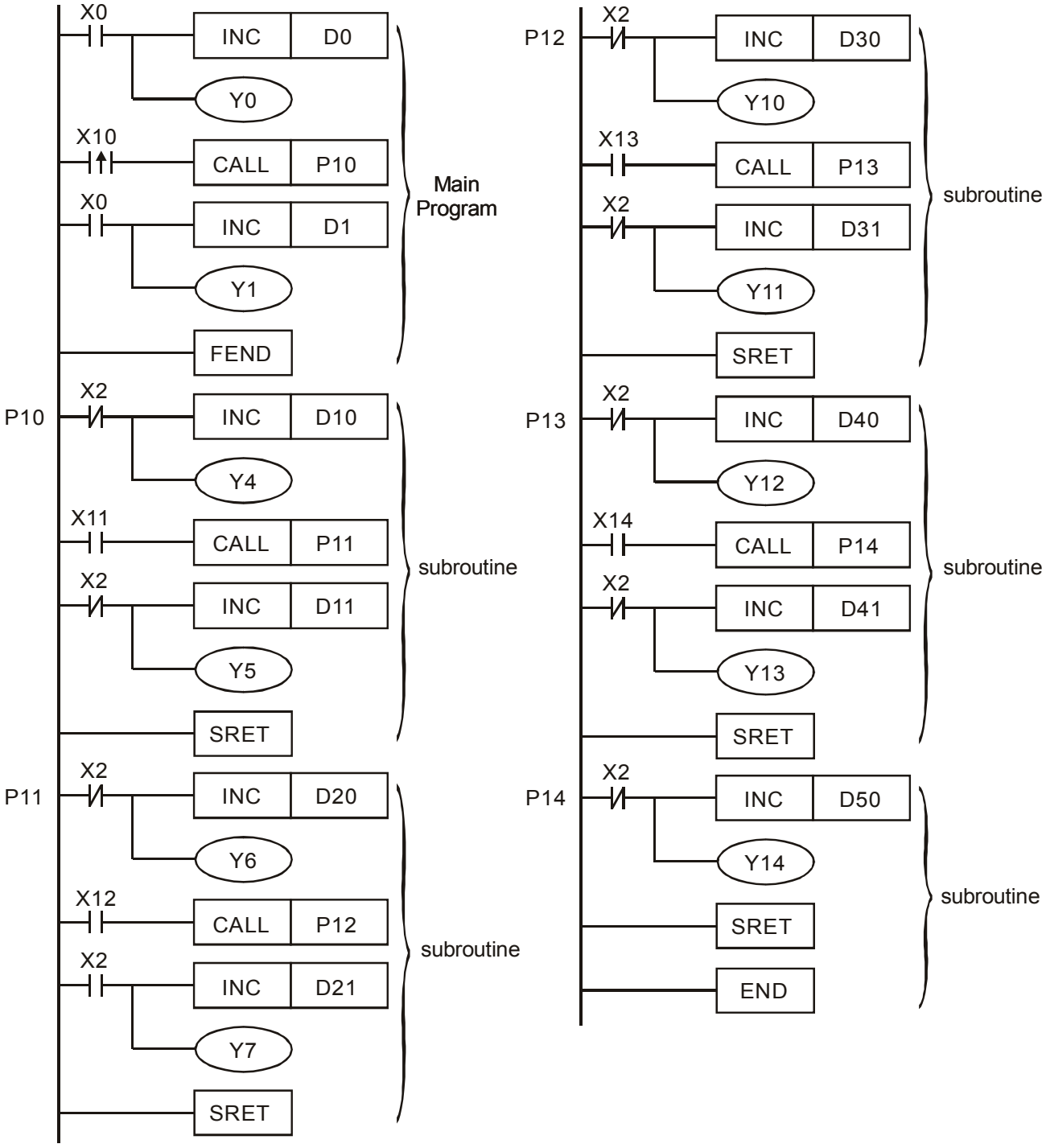
When X0 = On, CALL instruction is executed and the program jumps to the subroutine designated by P2. When SRET instruction is executed, the program returns to address 24 and continues its execution.



Program Example 2:

1. When X10 goes from Off to On, its rising-edge trigger executes CALL P10 instruction and the program jumps to the subroutine designated by P10.
2. When X11 is On, CALL P11 is executed and the program jumps to the subroutine designated by P11.
3. When X12 is On, CALL P12 is executed and the program jumps to the subroutine designated by P12.
4. When X13 is On, CALL P13 is executed and the program jumps to the subroutine designated by P13.

5. When X14 is On, CALL P14 is executed and the program jumps to the subroutine designated by P14. When SRET is executed, the program returns to the previous P** subroutine and continues its execution.
6. After SRET instruction is executed in P10 subroutine, returning to the main program.



Mnemonic	Operands	Function
FEND	None	The End of The Main Program (First End)

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F

- Note:
  1. No operand.
  2. No contact to drive the instruction is required.

16-bit instruction (1 Step)

FEND	Continuous execution	-	-

32-bit instruction

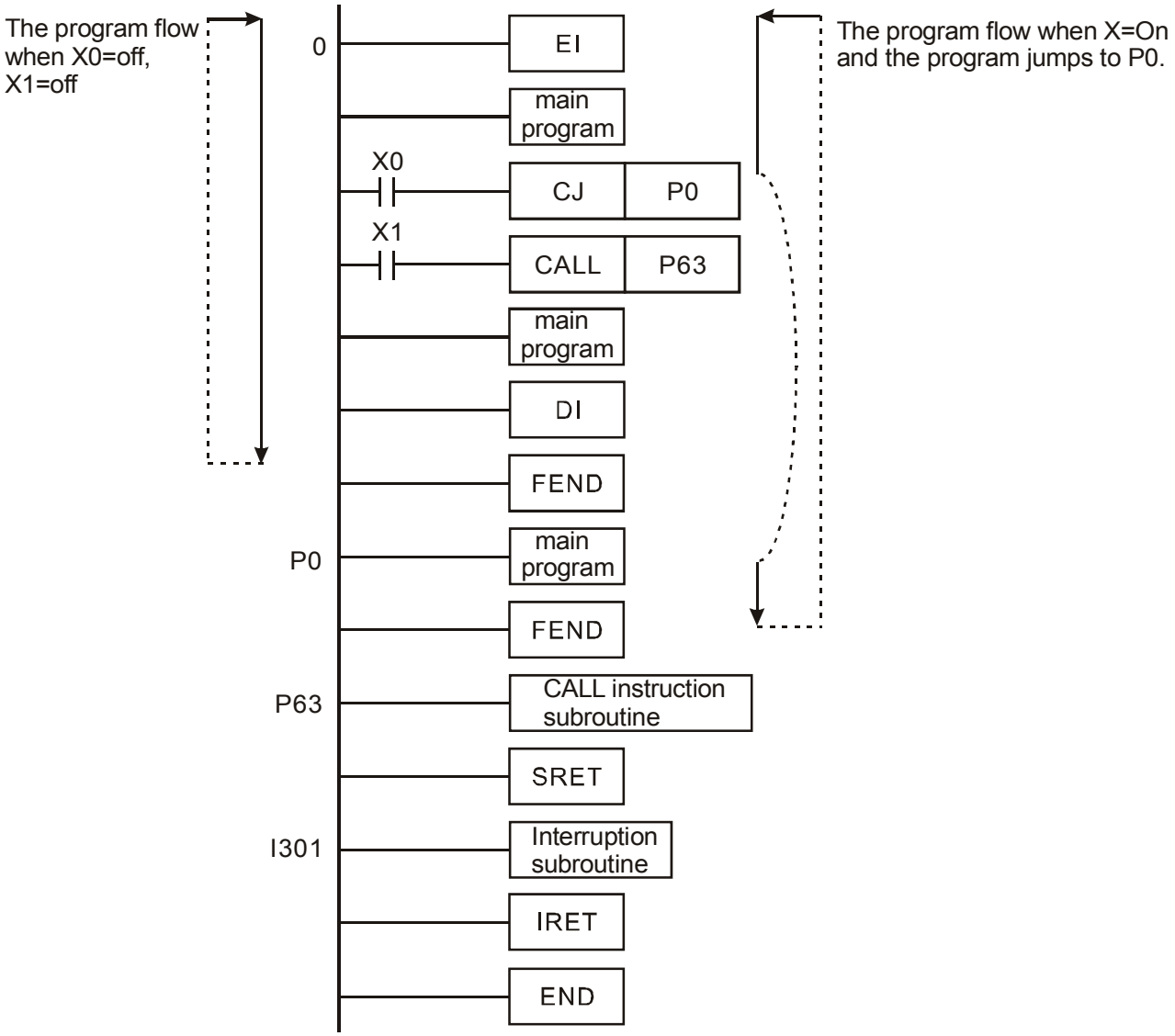
-	-	-	-

- Flags: None

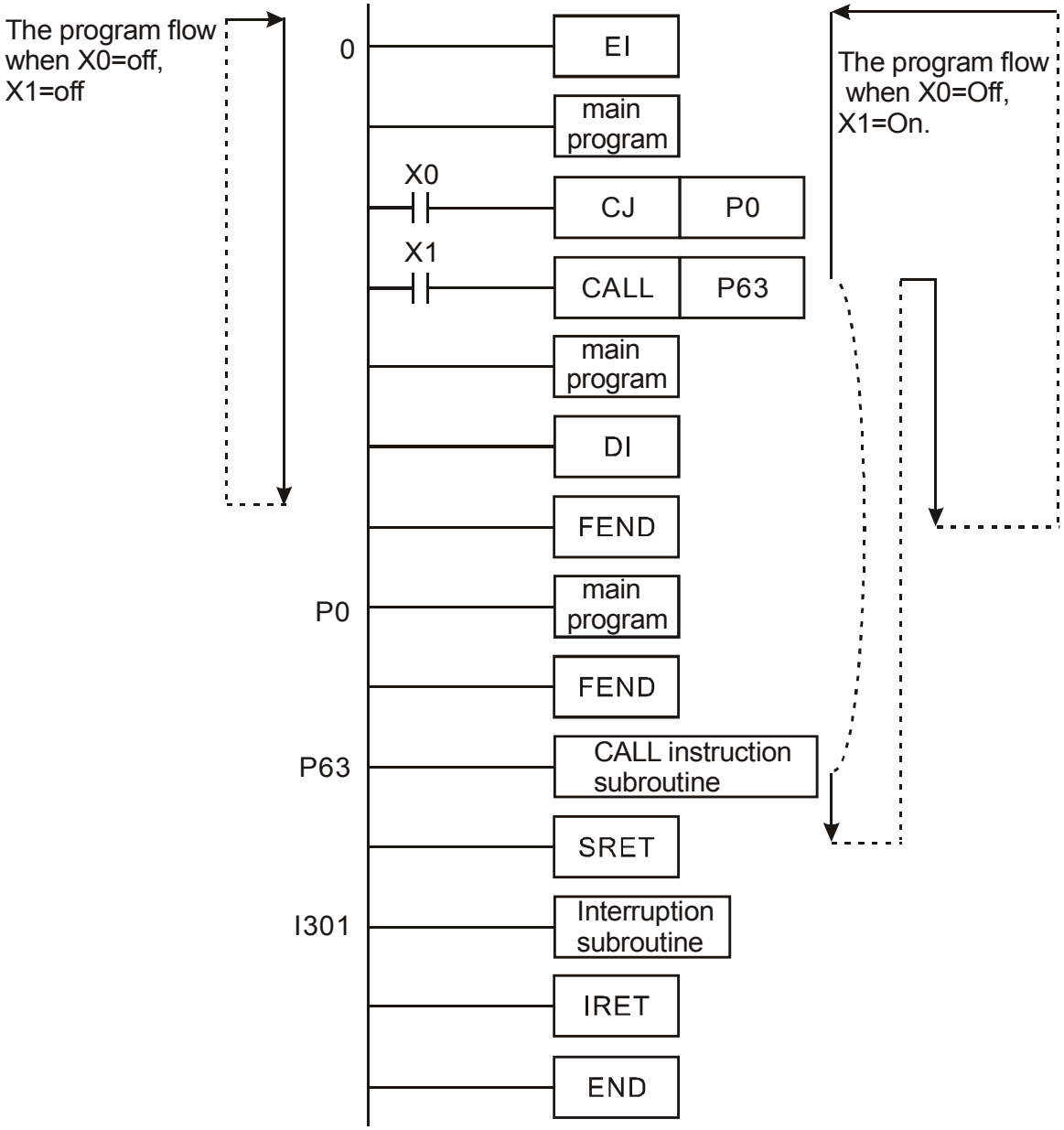
Explanations:

1. This instruction denotes the end of the main program. It has the same function as that of END instruction when being executed by DOP-EXIO series.
2. CALL must be written after FEND instruction and add SRET instruction in the end of its subroutine. Interruption program has to be written after FEND instruction and IRET must be added in the end of the service program.
3. If several FEND instructions are in use, place the subroutine and interruption service programs between the final FEND and END instruction.
4. After CALL instruction is executed, executing FEND before SRET will result in errors in the program.
5. After FOR instruction is executed, executing FEND before NEXT will result in errors in the program.

CJ Instruction Program Flow:



CALL Instruction Program Flow:



Mnemonic	Operands	Function
FOR	(S)	Start of a FOR-NEXT Loop

	Bit Devices				Word Devices									
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S					*	*	*	*	*	*	*	*	*	*	*

- Note:
  1. No contact to drive the instruction is required.

16-bit instruction (3 Steps)

FOR	Continuous execution	-	-

32-bit instruction

-	-	-	-

- Flags: None

Operands:

(S): The number of repeated nested loops

Mnemonic	Operands	Function
NEXT	None	End of a FOR-NEXT Loop

	Bit Devices				Word Devices									
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F

- Note:
  1. No operand.
  2. No contact to drive the instruction is required.

16-bit instruction (1 Step)

NEXT	Continuous execution	-	-

32-bit instruction

-	-	-	-

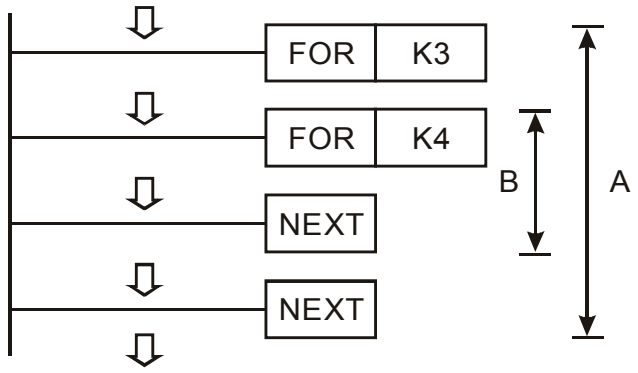
- Flags: None

Explanations:

1. FOR instruction indicates FOR ~ NEXT loops executing back and forth N times before escaping for the next execution.
2. N = K1 ~ K32,767. N is regarded as K1 when N ≤ 1.
3. When FOR~NEXT loops are not executed, the user can use the CJ instruction to escape the loops.
4. Error will occur when
   a) NEXT instruction is before FOR instruction.
   b) FOR instruction exists but NEXT instruction does not exist.
   c) There is NEXT instruction after FEND or END instruction.
   d) The number of instructions between FOR ~ NEXT differs.
5. FOR~NEXT loops can be nested for maximum five levels. Be careful that if there are too many loops, the increased PLC scan time may cause timeout of watchdog timer and error. Users can use WDT instruction to modify this problem.

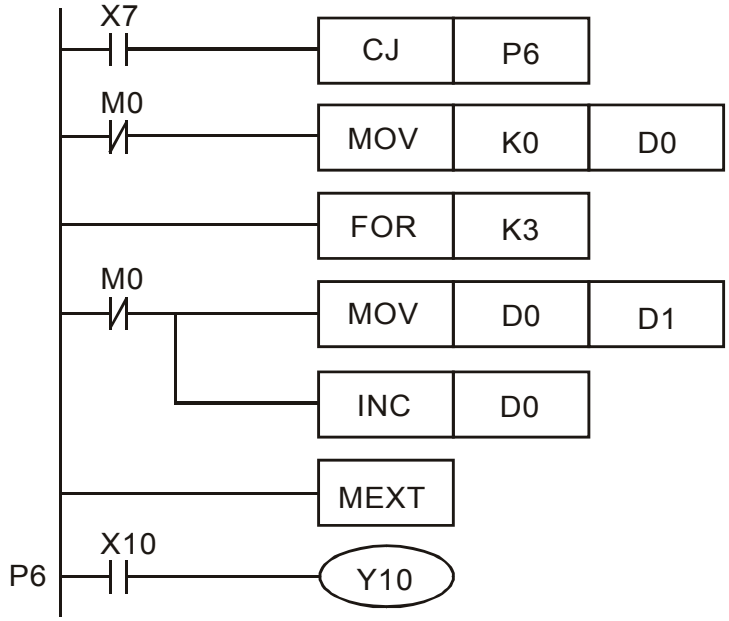
Program Example 1:

After program A has been executed for 3 times, it will resume its execution after NEXT instruction. Program B will be executed for 4 times whenever program A is executed once. Therefore, program B will be executed 3 × 4 = 12 times in total.



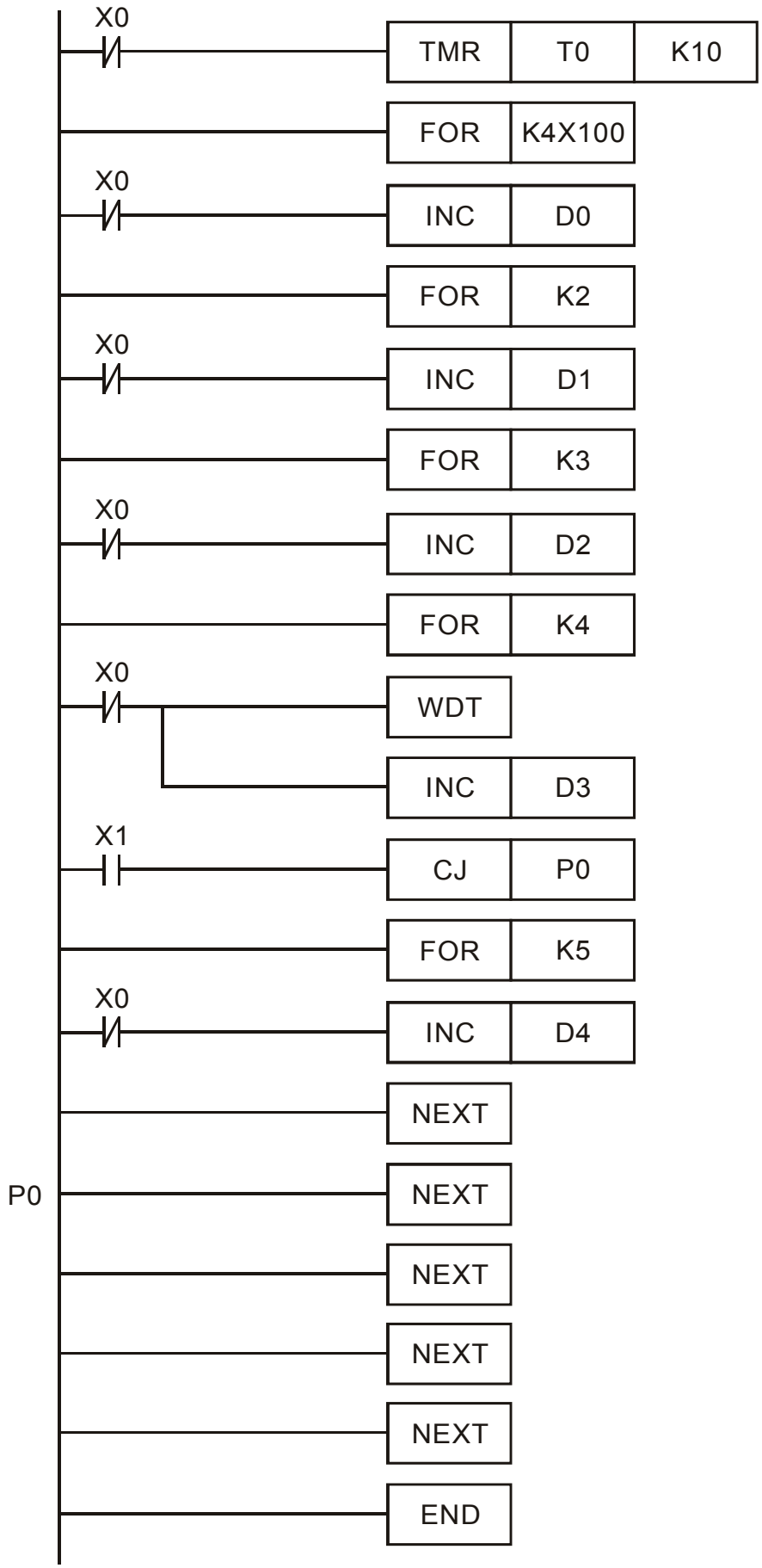
Program Example 2:

When X7 = Off, DOP-EXIO series will execute the program between FOR ~ NEXT. When X7 = On, CJ instruction jumps to P6 and avoids executing the programs between FOR ~ NEXT.



Program Example 3:

When the programs between FOR ~ NEXT are not to be executed, the user can adopt CJ instruction for a jumping. When the most inner FOR ~ NEXT loop is in the status of X1 = On, CJ instruction executes jumping to P0 and skips the execution on P0.


X0
TMR T0 K10
FOR K4X100
X0
INC D0
FOR K2
X0
INC D1
FOR K3
X0
INC D2
FOR K4
X0
WDT
INC D3
X1
CJ P0
FOR K5
X0
INC D4
NEXT
P0
NEXT
NEXT
NEXT
NEXT
END

Mnemonic		Operands	Function
CMP	**D**	(S1) (S2) (D)	Compare

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					*	*	*	*	*	*	*	*	*	*	*
S_2					*	*	*	*	*	*	*	*	*	*	*
D		*	*	*											

- Note:
  1. If S_1 and S_2 are used in device F, only 16-bit instruction is applicable.
  2. Operand D occupies 3 consecutive devices.

16-bit instruction (7 Steps)

CMP Continuous execution

32-bit instruction (13 Steps)

DCMP Continuous execution

- Flags: None

Operands:

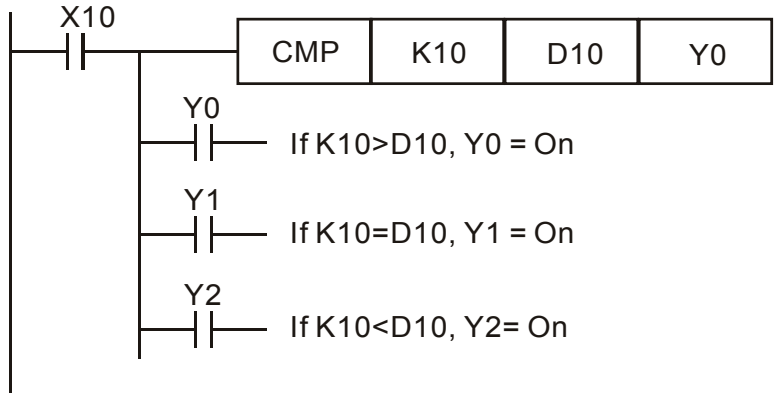
(S1): Comparison Value 1 (S2): Comparison Value 2 (D): Comparison result

Explanations:

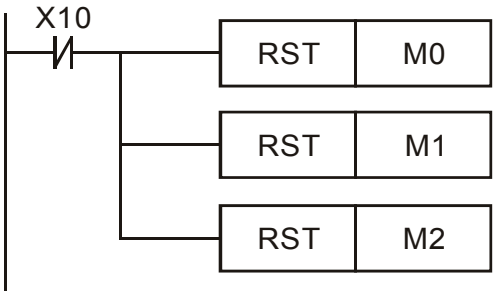
1. The contents in **S_1** and **S_2** are compared and the result will be stored in **D**.
2. The two comparison values are compared algebraically and the two values are signed binary values. When b15 = 1 in 16-bit instruction or b31 = 1 in 32-bit instruction, the comparison will regard the value as negative binary values.

Program Example:

1. Designate device Y0, and operand D automatically occupies Y0, Y1, and Y2.
2. When X10 = On, CMP instruction will be executed and one of Y0, Y1, and Y2 will be On. When X10 = Off, CMP instruction will not be executed and Y0, Y1, and Y2 remain their status before X10 = Off.
3. If the user needs to obtain a comparison result with ≥ ≤, and ≠, make a series parallel connection between Y0 ~ Y2.



4. To clear the comparison result, use RST or ZRST instruction.


X10
RST
M0
RST
M1
RST
M2


X10
ZRST
M0
M2

Mnemonic	Operands	Function
ZCP	(S1) (S2) (S) (D)	Zone Compare

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
S_2					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
S					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
D		✳	✳	✳											

- Note:
  1. If S_1, S_2 and S are used in device F, only 16-bit instruction is applicable.
  2. The content in S_1 should be smaller than the content in S_2.
  3. Operand D occupies 3 consecutive devices.

16-bit instruction (9 Steps)

ZCP Continuous execution

32-bit instruction (17 Steps)

DZCP Continuous execution

- Flags: None

Operands:

(S1): Lower bound of zone comparison (S2): Upper bound of zone comparison

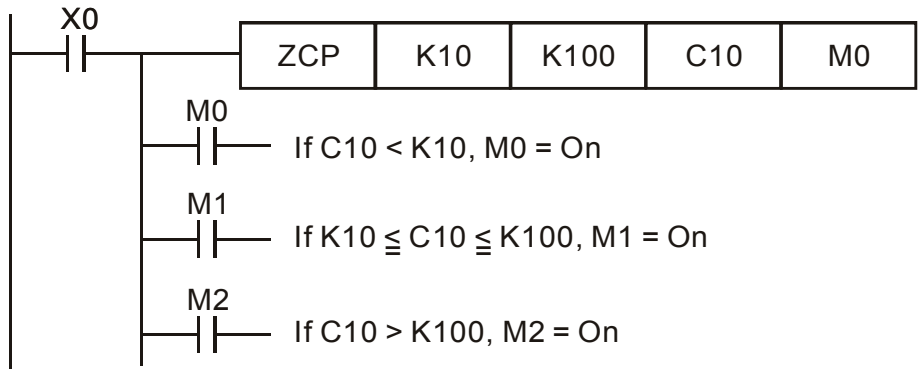
(S): Comparison value (D): Comparison result

Explanations:

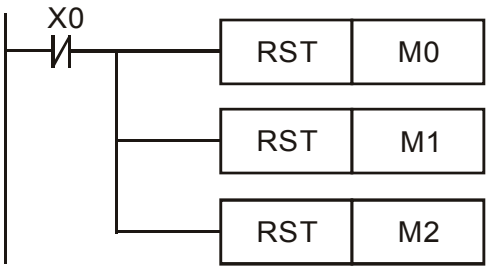
1. **S** is compared with its **S_1**, **S_2** and the result is stored in **D**.
2. When **S_1** > **S_2**, the instruction performs comparison by using **S_1** as the lower/upper bound.
3. The two comparison values are compared algebraically and the two values are signed binary values. When b15 = 1 in 16-bit instruction or b31 = 1 in 32-bit instruction, the comparison will regard the value as negative binary values.

Program Example:

1. Designate device M0, and operand D automatically occupies M0, M1 and M2.
2. When X0 = On, ZCP instruction will be executed and one of M0, M1, and M2 will be On. When X0 = Off, ZCP instruction will not be executed and M0, M1, and M2 remain their status before X0 = Off.



3. To clear the comparison result, use RST or ZRST instruction.


X0
RST
M0
RST
M1
RST
M2


X0
ZRST
M0
M2

Mnemonic	Operands	Function
MOV **D**	(S) (D)	Move

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
D								✳	✳	✳	✳	✳	✳	✳	✳

- Note:
  1. If S, and D are used in device F, only 16-bit instruction is applicable.

16-bit instruction (5 Steps)

MOV Continuous execution

32-bit instruction (9 Steps)

DMOV Continuous execution

- Flags: None

Operands:

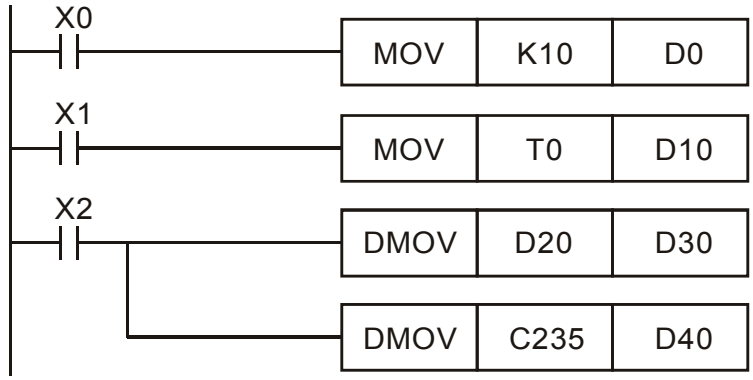
(S): Source of data (D): Destination of data

Explanations:

1. When this instruction is executed, the content of **S** will be moved directly to **D**. When this instruction is not executed, the content of **D** remains unchanged.
2. If the operation result refers to a 32-bit output, (i.e. application instruction MUL and so on), and the user needs to move the present value in the 32-bit high-speed counter, DMOV instruction has to be adopted.

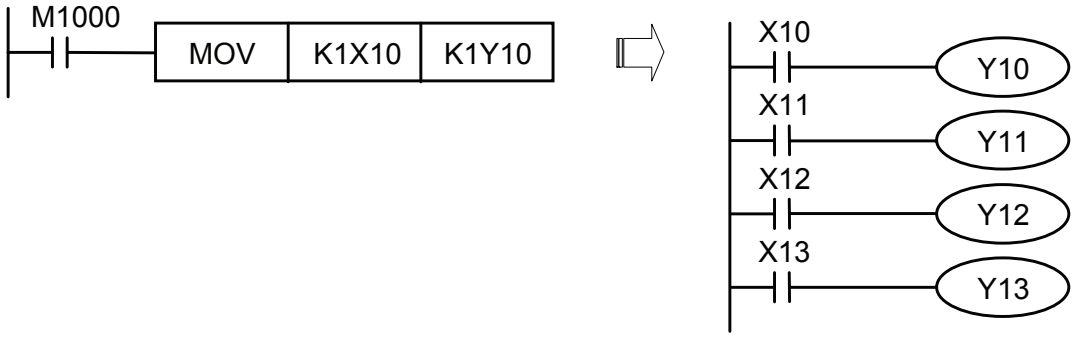
Program Example:

1. MOV instruction has to be adopted in the moving of 16-bit data.
   a) When X0 = Off, the content in D10 will remain unchanged. If X0 = On, the value K10 will be moved to D10 data register.
   b) When X1 = Off, the content in D10 will remain unchanged. If X1 = On, the present value T0 will be moved to D10 data register.
2. DMOV instruction has to be adopted in the moving of 32-bit data.
   When X2 = Off, the content in (D31, D30) and (D41, D40) will remain unchanged. If X2 = On, the present value of (D21, D20) will be sent to (D31, D30) data register. Meanwhile, the present value of C235 will be moved to (D41, D40) data register.



3. Move bit data:

When the program is driven, the data of X10~X13 is moved to the Y10~Y13. Please refer to the figure below. The left program has the same function as the right.



Mnemonic		Operands	Function
CML	**D**	(S) (D)	Compliment

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S					*	*	*	*	*	*	*	*	*	*	*
D								*	*	*	*	*	*	*	*

- Note:
  1. If S, and D are used in device F, only 16-bit instruction is applicable.

16-bit instruction (5 Steps)

CML	Continuous execution

32-bit instruction (9 Steps)

DCML	Continuous execution

- Flags: None

Operands:

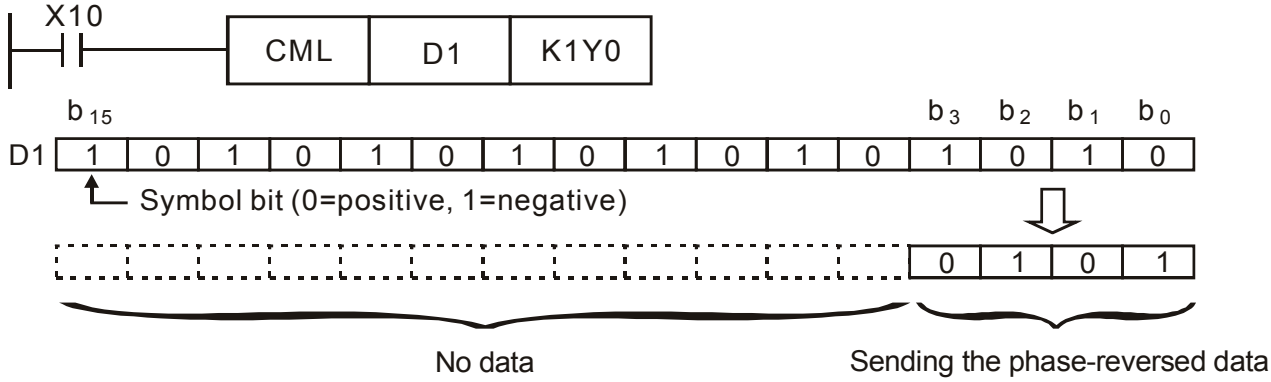
(S): Source of data (D): Destination device

Explanations:

1. This instruction can be used for phase-reversed output.
2. Reverse the phase (0→1, 1→0) of all the contents in **S** and send the contents to **D**. Given that the content is a constant K, K will be automatically converted into a BIN value.

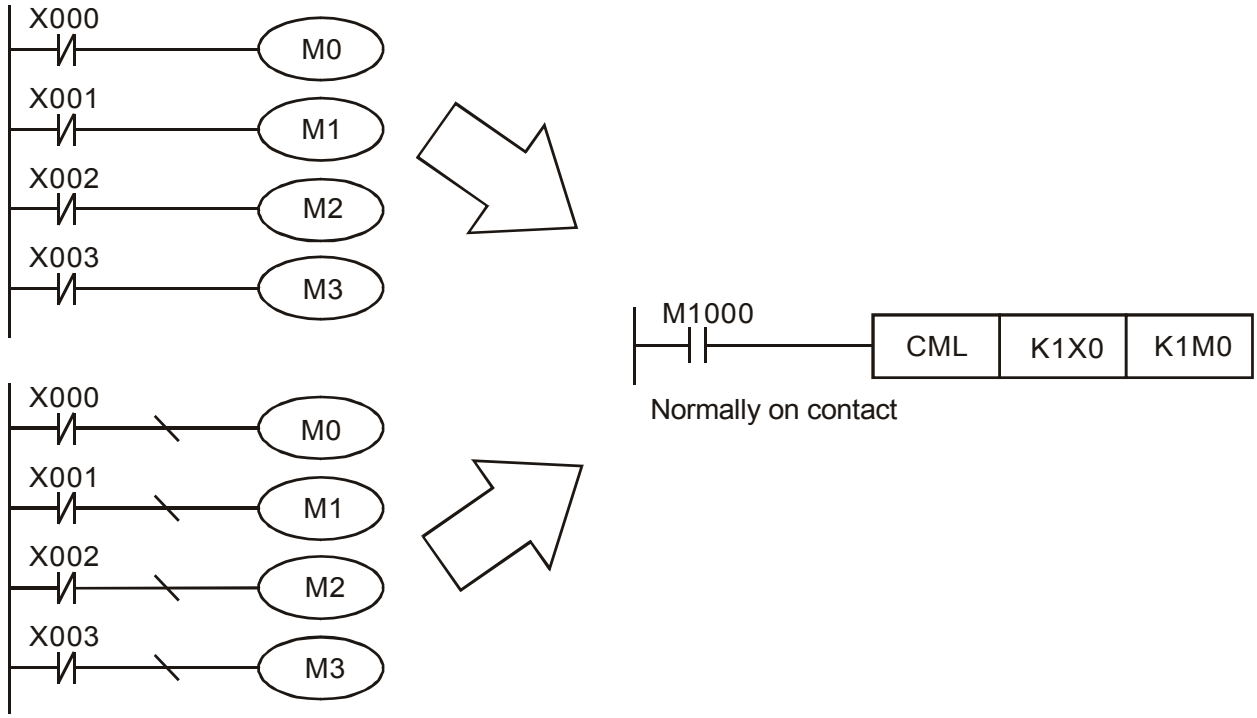
Program Example 1:

1. When the user wants to perform the phase-reversed output operation, please use this instruction.
2. When X10 = On, b0 ~ b3 in D1 will be phase-reversed and send to Y0 ~ Y3.



Program Example 2:

The loop below can also adopt CML instruction (see the right side program of the figure below).


X000
M0
X001
M1
X002
M2
X003
M3
M1000
CML
K1X0
K1M0
Normally on contact
X000
M0
X001
M1
X002
M2
X003
M3

Mnemonic		Operands	Function
BMOV	**D**	(S) (D) (n)	Block Move

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S							*	*	*	*	*	*	*		
D								*	*	*	*	*	*		
n					*	*									

- Note:
  1. Range of n: 1 ~ 512

16-bit instruction (7 Steps)

BMOV Continuous execution

32-bit instruction

- - - -

- Flags: None

Operands:

(S): Start of source devices (D): Start of destination devices (n): Number of data to be moved

Explanations:

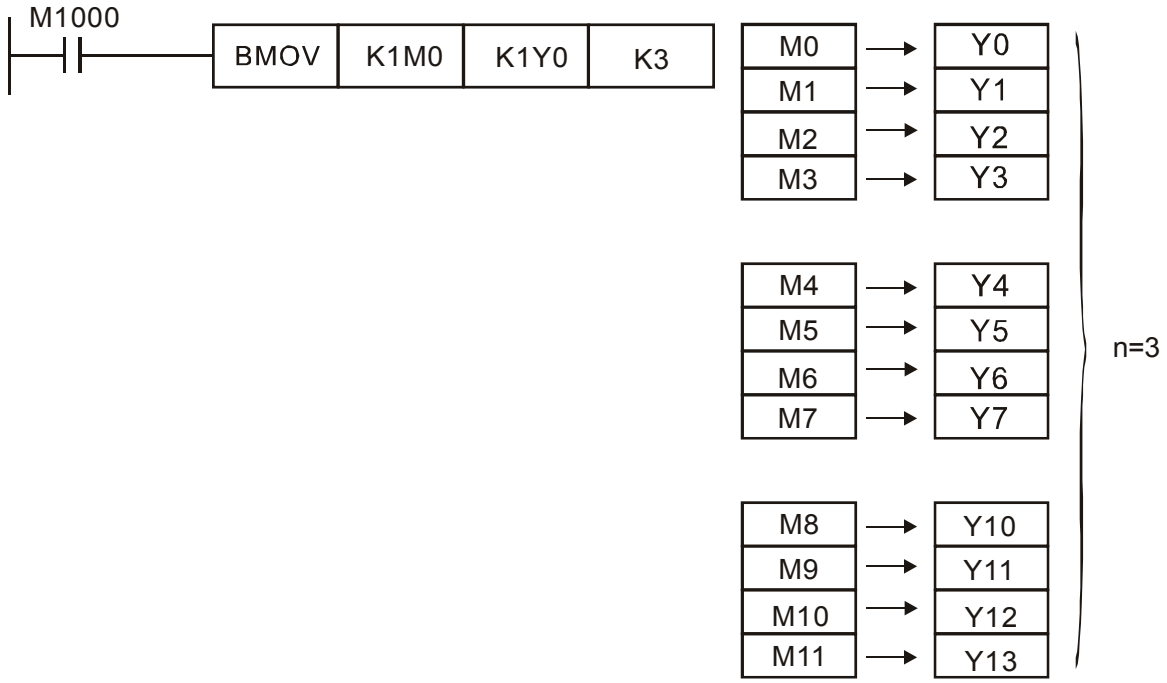
The contents in n registers starting from the device designated by **S** will be moved to n registers starting from the device designated by **D**. If n exceeds the actual number of available source devices, only the devices that fall within the valid range will be used.

Program Example 1:

When X10 = On, the contents in registers D0 ~ D3 will be moved to the 4 registers D20 ~ D23.

Program Example 2:

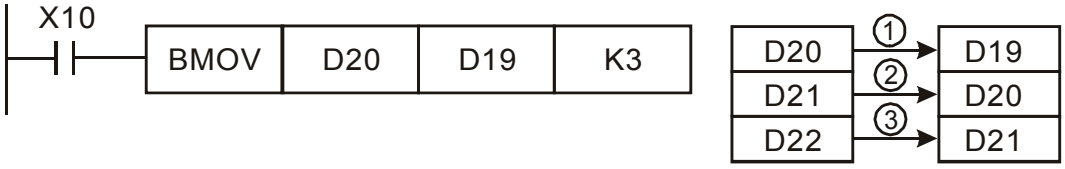
Assume the bit devices KnX, KnY, KnM and KnS are designated for moving, the number of digits of **S** and **D** has to be the same, i.e. their n has to be the same.



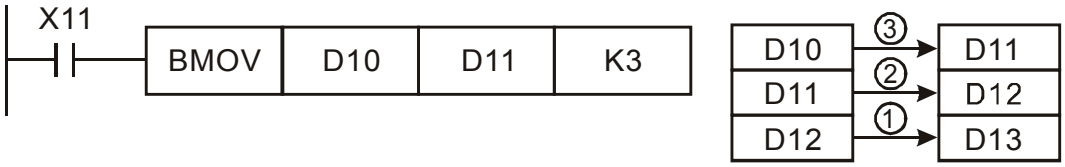
Program Example 3:

To avoid coincidence of the device numbers to be moved designated by the two operands and cause confusion, please be aware of the arrangement on the designated device numbers.

1. When **S** > **D**, the instruction is processed following the order ①→②→③



2. When **S** < **D**, the instruction is processed following the order ③→②→①



Mnemonic		Operands	Function
FMOV	**D**	(S) (D) (n)	Fill Move

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
D								✳	✳	✳	✳	✳	✳		
n					✳	✳									

- Note:
  1. If S is used in device F, only 16-bit instruction is applicable.
  2. Range of n: 1~ 512

16-bit instruction (7 Steps)

FMOV Continuous execution

32-bit instruction (13 Steps)

DFMOV Continuous execution

- Flags: None

Operands:

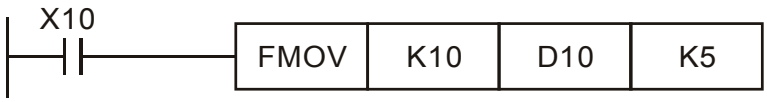
(S): Source of data (D): Destination of data (n): Number of data to be moved

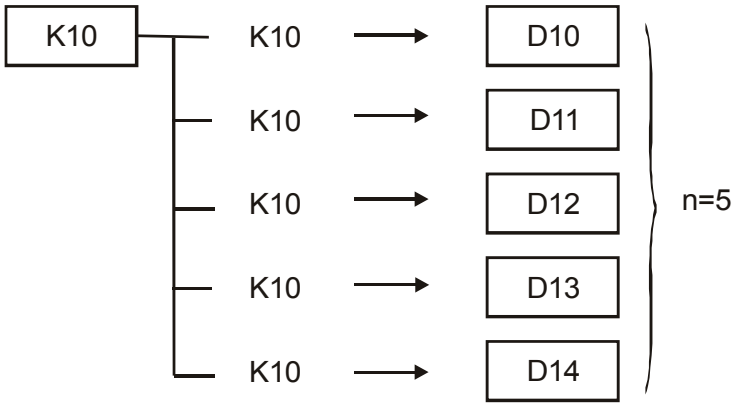
Explanations:

The contents in n registers starting from the device designated by **S** will be moved to n registers starting from the device designated by **D**. If n exceeds the actual number of available source devices, only the devices that fall within the valid range will be used.

Program Example:

When X10 = On, K10 will be moved to the 5 consecutive registers starting from D10.





Mnemonic	Operands	Function
XCH **D**	(D1) (D2)	Exchange

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D_1								*	*	*	*	*	*	*	*
D_2								*	*	*	*	*	*	*	*

- Note:
  1. If D_1 and D_2 are used in device F, only 16-bit instruction is applicable.

16-bit instruction (5 Steps)

XCH Continuous execution

32-bit instruction (9 Steps)

DXCH Continuous execution

- Flags: None

Operands:

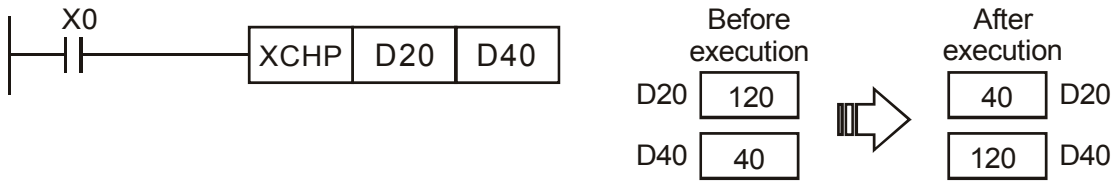
(D1): Data to be exchanged 1 (D2): Data to be exchanged 2

Explanations:

The contents in the devices designated by **D_1** and **D_2** will exchange.

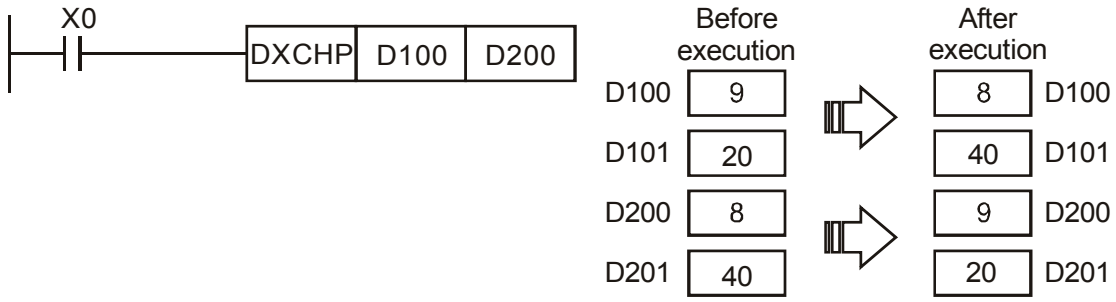
Program Example 1:

When X0 = Off→On, the contents in D20 and D40 exchange with each other.



Program Example 2:

When X0 = Off → On, the contents in D100 and D200 exchange with each other.



Mnemonic		Operands	Function
BCD	**D**	(S) (D)	Binary Coded Decimal

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S							*	*	*	*	*	*	*	*	*
D								*	*	*	*	*	*	*	*

- Note:
  1. If S and D are used in device F, only 16-bit instruction is applicable.

16-bit instruction (5 Steps)

BCD Continuous execution

32-bit instruction (9 Steps)

DBCD Continuous execution

- Flags: M1067 (Calculation error)
  M1068 (Calculation error locked)

Operands:

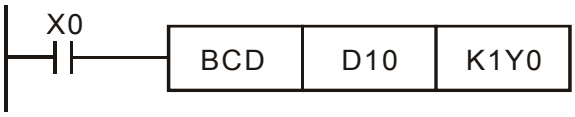
(S): Source of data (D): Conversion result

Explanations:

1. The content in **S** (BIN value) is converted into BCD value and stored in **D**.
2. As a 16-bit instruction, when the conversion result exceeds the range of 0 ~ 9,999, M1067 and M1068 will be On.
3. As a 32-bit instruction, when the conversion result exceeds the range of 0 ~ 99,999,999, M1067 and M1068 will be On.
4. The four arithmetic operations and applications in DOP-EXIO series and the execution of INC and DEC instructions are performed in BIN format. Therefore, if the user needs to see the decimal value display, simply use this instruction to convert the BIN value into BCD value.

Program Example:

When X0 = On, the binary value of D10 will be converted into BCD value, and the 1s digit of the conversion result will be stored in K1Y0 (Y0 ~ Y3, the 4 bit devices).



Mnemonic		Operands	Function
BIN	**D**	(S) (D)	Binary

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S							*	*	*	*	*	*	*	*	*
D								*	*	*	*	*	*	*	*

- Note:
  1. If S and D are used in device F, only 16-bit instruction is applicable.

16-bit instruction (5 Steps)

BIN Continuous execution

32-bit instruction (9 Steps)

DBIN Continuous execution

- Flags: M1067 (Calculation error)

  M1068 (Calculation error locked)

Operands:

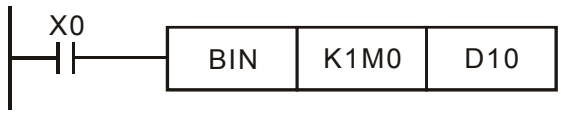
(S): Source of data (D): Conversion result

Explanations:

1. The content in **S** (BCD value) is converted into BIN value and stored in **D**.
2. Valid range of **S** : BCD (0 ~ 9,999), DBCD (0 ~ 99,999,999)
3. Provided the content in S is not a BCD value (in hex and any one of its digits does not fall in the range of 0 ~ 9), an operation error will occur, and M1067 and M1068 will be On.
4. Constant K and H will automatically be converted into BIN format. Thus, they do not need to adopt this instruction.

Program Example:

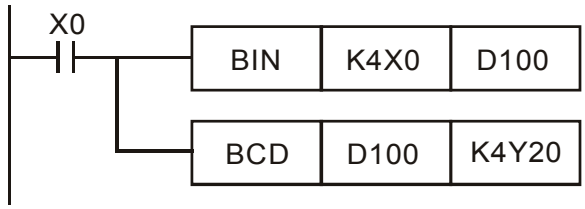
When X0 = On, the BCD value of K1M0 will be converted to BIN value and stored in D10.

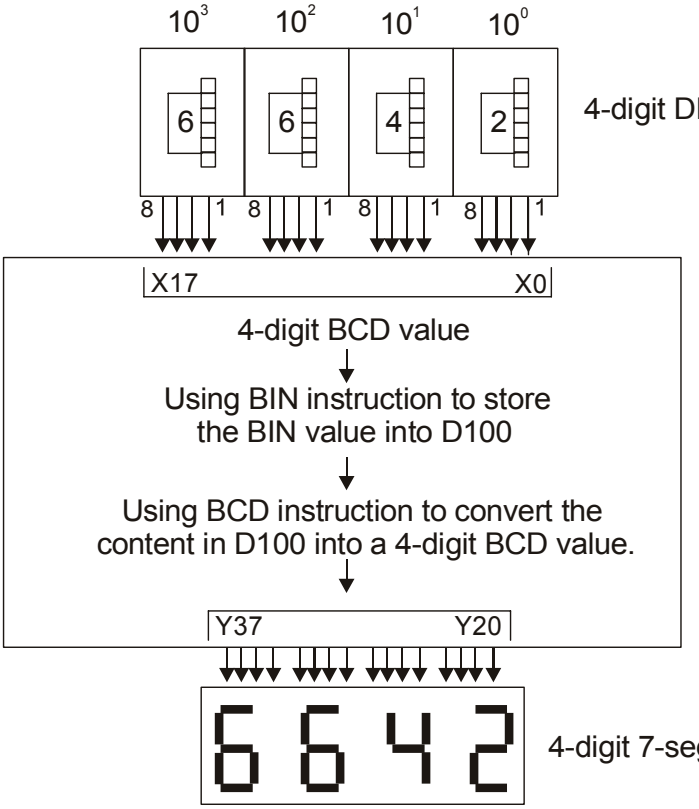


Remarks:

Explanations on BCD and BIN instructions:

1. When DOP-EXIO series needs to read an external DIP switch in BCD format, BIN instruction has to be first adopted to convert the read data into BIN value and store the data in DOP-EXIO series.
2. When DOP-EXIO series needs to display its stored data by a 7-segment display in BCD format, BCD instruction has to be first adopted to convert the data into BCD value and send the data to the 7-segment display.
3. When X0 = On, the BCD value of K4X0 is converted into BIN value and sent it to D100. The BIN value of D100 will then be converted into BCD value and sent to K4Y20.





4-digit DIP switch in BCD format

4-digit 7-segment display in BCD format

Mnemonic		Operands	Function
ADD	**D**	(S1) (S2) (D)	Addition

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					*	*	*	*	*	*	*	*	*	*	*
S_2					*	*	*	*	*	*	*	*	*	*	*
D								*	*	*	*	*	*	*	*

- Note:
  1. If S_1, S_2 and D are used in device F, only 16-bit instruction is applicable.

16-bit instruction (7 Steps)

ADD Continuous execution

32-bit instruction (13 Steps)

DADD Continuous execution

- Flags: M1020 (Zero flag)
  M1021 (Barrow flag)
  M1022 (Carry flag)

Operands:

(S1): Summand (S2): Addend (D): Sum

Explanations:

1. This instruction adds **S_1** and **S_2** in BIN format and store the result in **D**.
2. The highest bit is symbolic bit 0 (+) and 1 (-), which is suitable for algebraic addition, e.g. 3 + (-9) = -6.
3. Flag changes in binary addition

   In 16-bit BIN addition,

   a) If the operation result = 0, zero flag M1020 = On.
   b) If the operation result < -32,768, borrow flag M1021 = On.
   c) If the operation result > 32,767, carry flag M1022 = On.

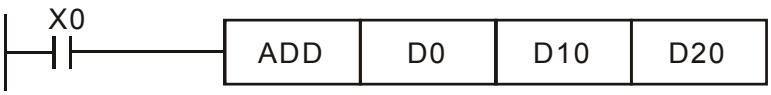
   In 32-bit BIN addition,

   a) If the operation result = 0, zero flag M1020 = On.
   b) If the operation result < -2,147,483,648, borrow flag M1021 = On.
   c) If the operation result > 2,147,483,647, carry flag M1022 = On.

Program Example 1:

In 16-bit BIN addition:

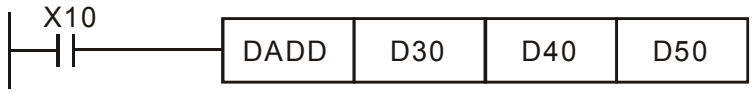
When X0 = On, the content in D0 will plus the content in D10 and the sum will be stored in D20.



Program Example 2:

In 32-bit BIN addition:

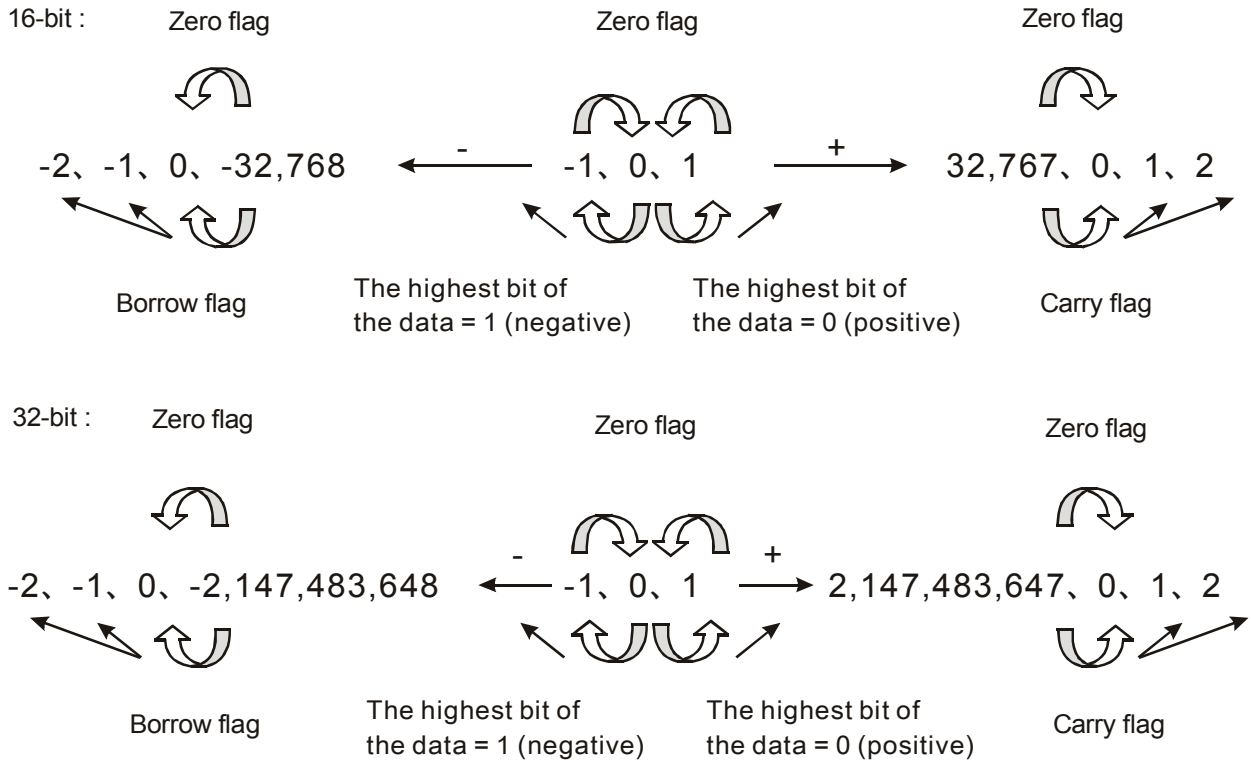
When X0 = On, the content in (D31, D30) will plus the content in (D41, D40) and the sum will be stored in (D51, D50). D30, D40 and D50 are low 16-bit data; D31, D41 and D51 are high 16-bit data.



(D31, D30) + (D41, D40) = (D51, D50)

Remarks:

Flags and the positive/negative sign of the values:



Mnemonic		Operands	Function
SUB	**D**	(S1) (S2) (D)	Subtraction

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
S_2					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
D								✳	✳	✳	✳	✳	✳	✳	✳

- Note:
  1. If S_1, S_2 and D are used in device F, only 16-bit instruction is applicable.

16-bit instruction (7 Steps)

SUB Continuous execution

32-bit instruction (13 Steps)

DSUB Continuous execution

- Flags: M1020 (Zero flag)
  M1021 (Barrow flag)
  M1022 (Carry flag)

Operands:

(S1): Minuend (S2): Subtrahend (D): Remainder

Explanations:

1. This instruction subtracts $\mathbf{S}_1$ and $\mathbf{S}_2$ in BIN format and stores the result in **D**.
2. The highest bit is symbolic bit 0 (+) and 1 (-), which is suitable for algebraic subtraction.
3. Flag changes in binary subtraction

   In 16-bit instruction:

   a) If the operation result = 0, zero flag M1020 = On.
   b) If the operation result < –32,768, borrow flag M1021 = On.
   c) If the operation result > 32,767, carry flag M1022 = On.

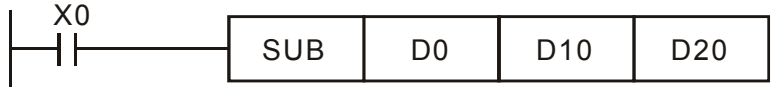
   In 32-bit instruction:

   a) If the operation result = 0, zero flag M1020 = On.
   b) If the operation result < –2,147,483,648, borrow flag M1021 = On.
   c) If the operation result > 2,147,483,647, carry flag M1022 = On.
4. For flag operations of SUB instruction and the positive/negative sign of the value, see the explanations in ADD instruction on the previous page.

Program Example 1:

In 16-bit BIN subtraction:

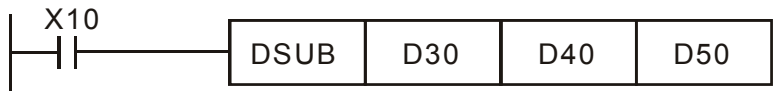
When X0 = On, the content in D0 will minus the content in D10 and the remainder will be stored in D20.



Program Example 2:

In 32-bit BIN subtraction:

When X10 = On, the content in (D31, D30) will minus the content in (D41, D40) and the remainder will be stored in (D51, D50). D30, D40 and D50 are low 16-bit data; D31, D41 and D51 are high 16-bit data.



(D31, D30) − (D41, D40) = (D51, D50)

Mnemonic		Operands	Function
MUL	**D**	(S1) (S2) (D)	Multiplication

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					*	*	*	*	*	*	*	*	*	*	*
S_2					*	*	*	*	*	*	*	*	*	*	*
D								*	*	*	*	*	*	*	

- Note:
  1. If S_1 and S_2 are used in device F, only 16-bit instruction is applicable.
  2. If D is used in device E, only 16-bit instruction is applicable.

16-bit instruction (7 Steps)

MUL Continuous execution

32-bit instruction (13 Steps)

DMUL Continuous execution

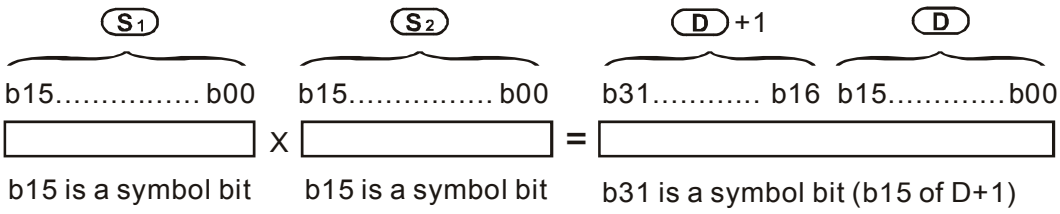
- Flags: None

Operands:

(S1): Multiplicand (S2): Multiplicator (D): Product

Explanations:

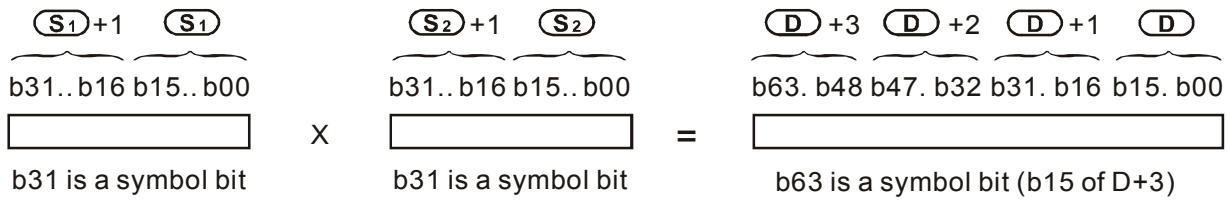
1. This instruction multiplies **S_1** by **S_2** in BIN format and stores the result in **D**. Be careful with the positive/negative signs of **S_1**, **S_2** and **D** when doing 16-bit and 32-bit operations.
2. In 16-bit BIN multiplication,



Symbol bit = 0 refers to a positive value.
Symbol bit = 1 refers to a negative value.

When D serves as a bit device, it can designate K1 ~ K8 and construct a 32-bit result.

3. 32-bit BIN multiplication,

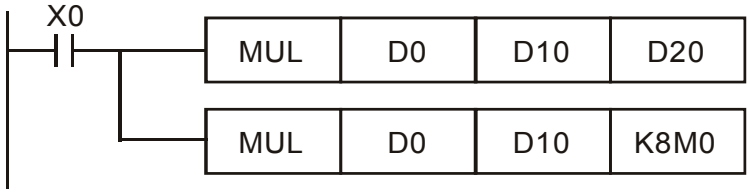


Symbol bit = 0 refers to a positive value.
Symbol bit = 1 refers to a negative value.

When D serves as a bit device, it can designate K1 ~ K8 and construct a 32-bit result, but only stores low 32-bit data.

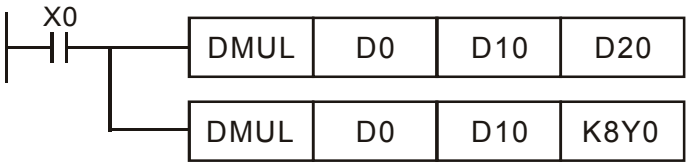
Program Example 1:

The 16-bit D0 is multiplied by the 16-bit D10 and stores the result in a 32-bit data(D21, D20). The higher 16-bit data is stored in D21 and the lower 16-bit data is stored in D20. On/Off of the most left bit indicates the positive/negative status of the result value.



Program Example 2:

The 32-bit (D1, D0) is multiplied by the 32-bit (D11, D10) and stores the result in a 64-bit data (D23, D22, D21, D20). On/Off of the most left bit indicates the positive/negative status of the result value.



Mnemonic		Operands	Function
DIV	**D**	(S1) (S2) (D)	Division

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
S_2					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
D								✳	✳	✳	✳	✳	✳	✳	

16-bit instruction (7 Steps)

DIV Continuous execution

32-bit instruction (13 Steps)

DDIV Continuous execution

- Flags: None

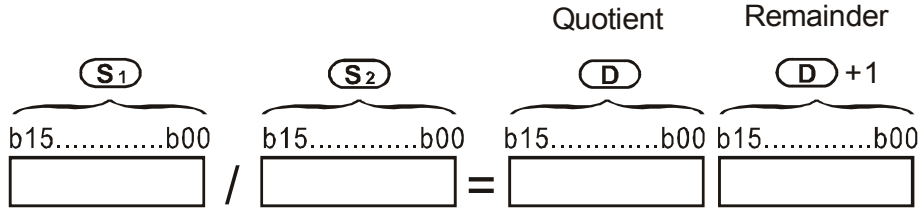
- Note:
  1. If S_1 and S_2 are used in device F, only 16-bit instruction is applicable.
  2. If D is used in device E, only 16-bit instruction is applicable.

Operands:

(S1): Dividend (S2): Divisor (D): Quotient and remainder

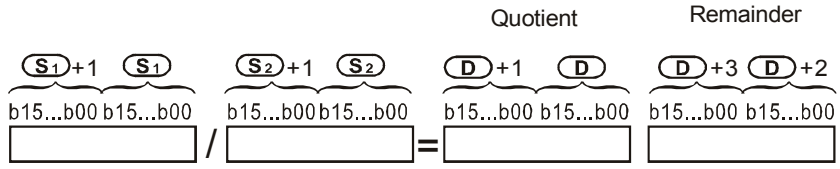
Explanations:

1. This instruction divides **S_1** and **S_2** in BIN format and stores the result in **D**. Be careful with the positive/negative signs of **S_1**, **S_2** and **D** when doing 16-bit and 32-bit operations.
2. This instruction will not be executed when the divisor is 0.
3. In 16-bit BIN division,



When D serves as a bit device, it can designate K1 ~ K8 and construct a 32-bit result, and bringing forth the quotient and remainder.

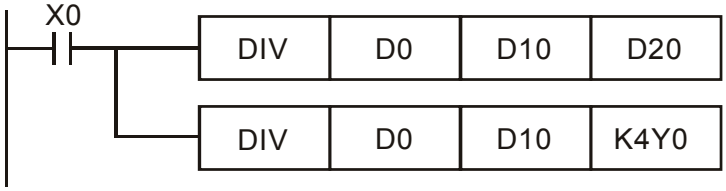
4. In 32-bit BIN division,



When D serves as a bit device, it can designate K1 ~ K8 and construct a 32-bit result, and bringing forth only quotient without the remainder.

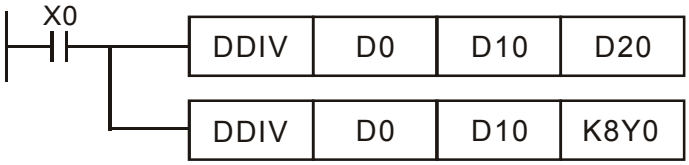
Program Example 1:

When X0 = On, D0 will be divided by D10 and the quotient will be stored in D20 and remainder in D21. On/Off of the highest bit indicates the positive/negative status of the result value.



Program Example 2:

When X0 = On, (D1, D0) will be divided by (D11, D10) and the quotient will be stored in (D21, D20) and remainder in (D23, D22). On/Off of the highest bit indicates the positive/negative status of the result value.



Mnemonic	Operands	Function
INC **D**	(D)	Increment

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D								*	*	*	*	*	*	*	*

- Note:
  1. If D is used in device F, only 16-bit instruction is applicable.

16-bit instruction (3 Steps)

INC Continuous execution

32-bit instruction (5 Steps)

DINC Continuous execution

- Flags: None

Operands:

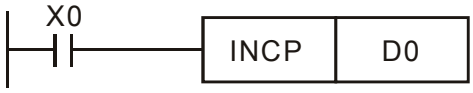
(D): Destination device

Explanations:

1. If the instruction is not a pulse execution one, the content in the designated device D will plus “1” in every scan period whenever the instruction is executed.
2. In 16-bit operation, 32,767 pluses 1 and obtains -32,768. In 32-bit operation, 2,147,483,647 pluses 1 and obtains -2,147,483,648.
3. The operation results will not affect any flags.

Program Example:

When X0 = Off→On, the content in D0 pluses 1 automatically.



Mnemonic	Operands	Function
DEC D	(D)	BIN 減一

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D								*	*	*	*	*	*	*	*

- Note:
  1. If D is used in device F, only 16-bit instruction is applicable.

16-bit instruction (3 Steps)

DEC Continuous execution

32-bit instruction (5 Steps)

DDEC Continuous execution

- Flags: None

Operands:

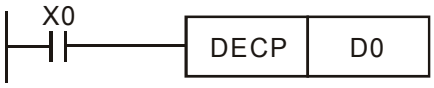
(D): Destination device

Explanations:

1. If the instruction is not a pulse execution one, the content in the designated device D will minus "1" in every scan period whenever the instruction is executed.
2. In 16-bit operation, -32,768 minuses 1 and obtains 32,767. In 32-bit operation, -2,147,483,648 minuses 1 and obtains 2,147,483,647.
3. The operation results will not affect any flags.

Program Example:

When X0 = Off→On, the content in D0 minuses 1 automatically.



Mnemonic		Operands	Function
AND	**D**	(S1) (S2) (D)	Logical Word AND

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
S_2					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
D								✳	✳	✳	✳	✳	✳	✳	✳

- Note:
  1. If S_1, S_2 and D are used in device F, only 16-bit instruction is applicable.

16-bit instruction (7 Steps)

WAND Continuous execution

32-bit instruction (13 Steps)

DAND Continuous execution

- Flags: None

Operands:

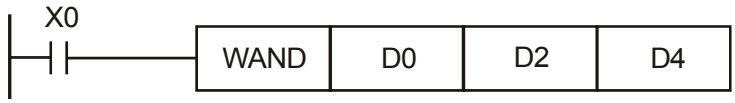
(S1): Source data device 1 (S2): Source data device 2 (D): Operation result

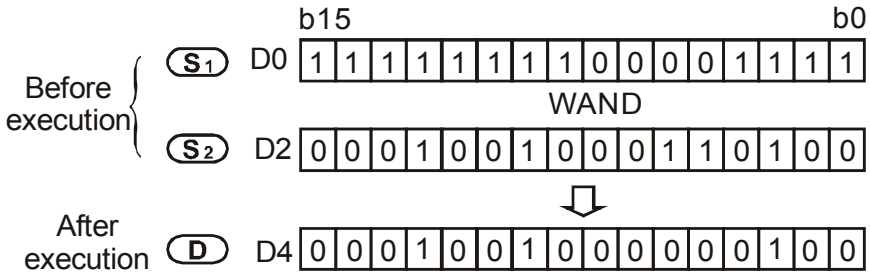
Explanations:

1. This instruction conducts logical AND operation of **S_1** and **S_2** and stores the result in **D**.
2. Operation rule: The corresponding bit of the operation result in **D** will be "0" if any of the bits in **S_1** or **S_2** is "0".

Program Example 1:

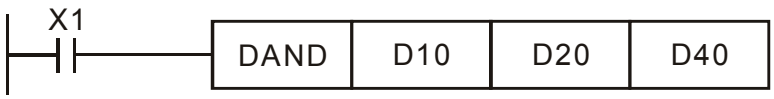
When X0 = On, the 16-bit D0 and D2 will perform WAND, logical AND operation, and the result will be stored in D4.

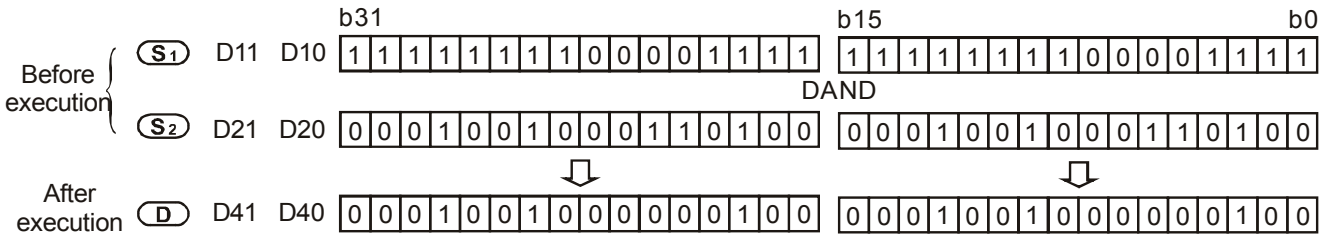




Program Example 2:

When X1 = On, the 32-bit (D11, D10) and (D21, D20) will perform DAND, logical AND operation, and the result will be stored in (D41, D40).




b31
b15
b0
Before execution
S1
D11
D10
1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1
DAND
S2
D21
D20
0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0
0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0
After execution
D
D41
D40
0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0

Mnemonic		Operands	Function
OR	**D**	(S1) (S2) (D)	Logical Word OR

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
S_2					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
D								✳	✳	✳	✳	✳	✳	✳	✳

- Note:
  1. If S_1, S_2 and D are used in device F, only 16-bit instruction is applicable.

16-bit instruction (7 Steps)

WOR Continuous execution

32-bit instruction (13 Steps)

DOR Continuous execution

- Flags: None

Operands:

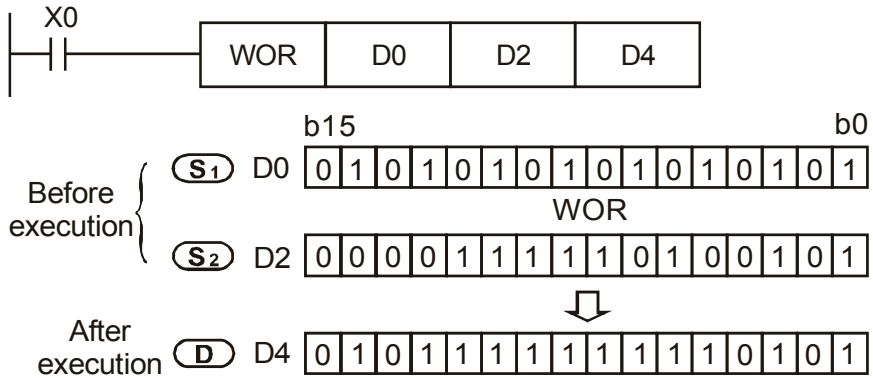
(S1): Source data device 1 (S2): Source data device 2 (D): Operation result

Explanations:

1. This instruction conducts logical OR operation of **S_1** and **S_2** and stores the result in **D**.
2. Operation rule: The corresponding bit of the operation result in **D** will be "1" if any of the bits in **S_1** or **S_2** is "1".

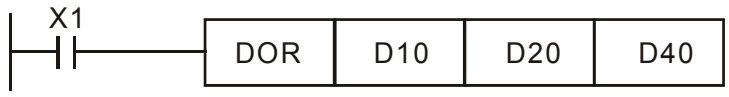
Program Example 1:

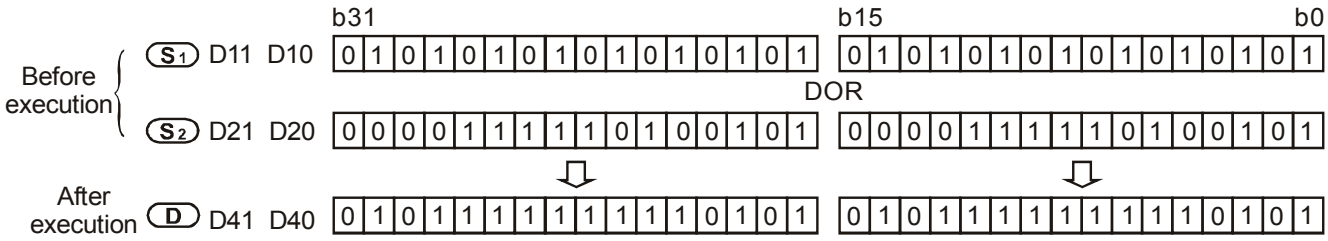
When X0 = On, the 16-bit D0 and D2 will perform WOR, logical OR operation, and the result will be stored in D4.



Program Example 2:

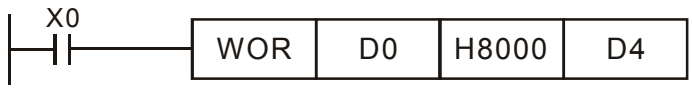
When X1 = On, the 32-bit (D11, D10) and (D21, D20) will perform DOR, logical OR operation, and the result will be stored in (D41, D40).

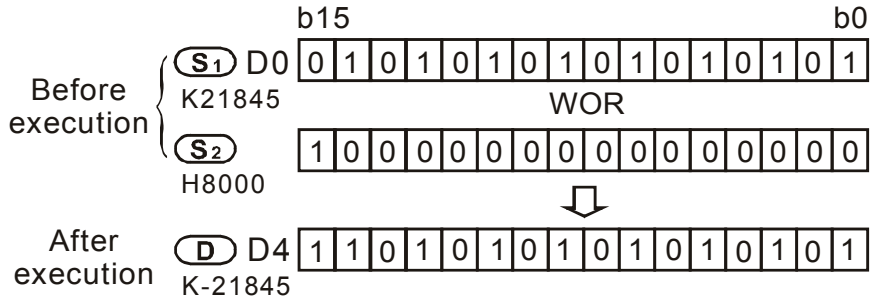




Program Example 3:

Positive value becomes negative value.





Revision March, 2008, Doc. Name: 2007PDD23000014

Mnemonic	Operands	Function
XOR D	(S1) (S2) (D)	Logical Exclusive OR

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
S_2					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
D								✳	✳	✳	✳	✳	✳	✳	✳

- Note:
  1. If S_1, S_2 and D are used in device F, only 16-bit instruction is applicable.

16-bit instruction (7 Steps)

WXOR Continuous execution

32-bit instruction (13 Steps)

DXOR Continuous execution

- Flags: None

Operands:

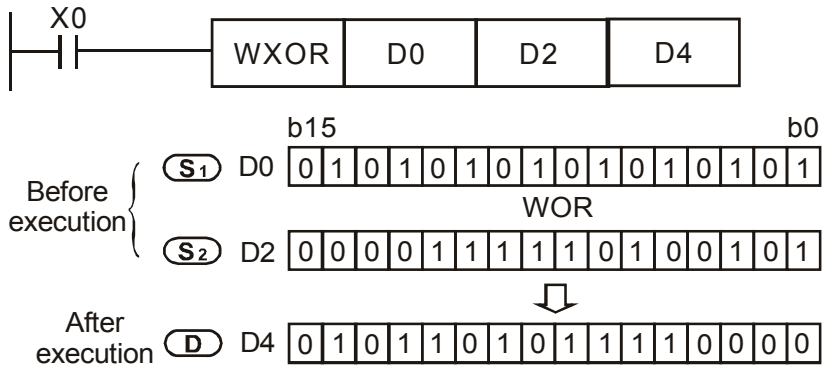
(S1): Source data device 1 (S2): Source data device 2 (D): Operation result

Explanations:

1. This instruction conducts logical XOR operation of **S_1** and **S_2** and stores the result in **D**.
2. Operation rule: If the bits in **S_1** and **S_2** are the same, the corresponding bit of the operation result in **D** will be "0"; if the bits in **S_1** and **S_2** are different, the corresponding bit of the operation result in **D** will be "1".

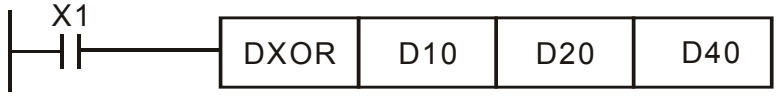
Program Example 1:

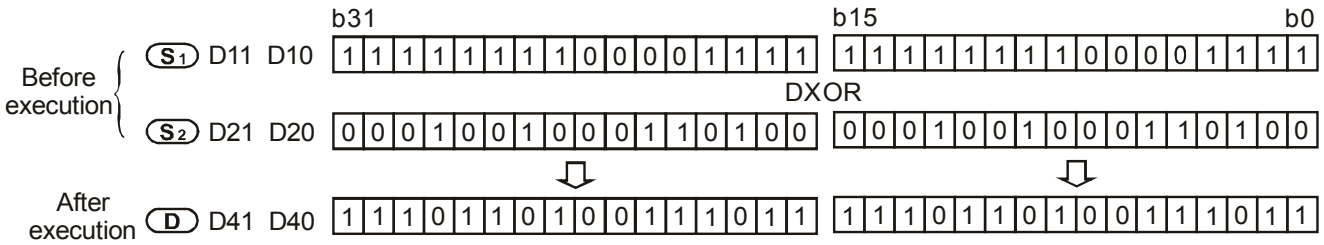
When X0 = On, the 16-bit D0 and D2 will perform WXOR, logical XOR operation, and the result will be stored in D4.



Program Example 2:

When X1 = On, the 32-bit (D11, D10) and (D21, D20) will perform DXOR, logical XOR operation, and the result will be stored in (D41, D40).




b31
b15
b0
Before execution
S1 D11 D10
1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1
1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1
DXOR
S2 D21 D20
0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0
0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0
After execution
D D41 D40
1 1 1 0 1 1 0 1 0 0 1 1 1 0 1 1
1 1 1 0 1 1 0 1 0 0 1 1 1 0 1 1

Mnemonic	Operands	Function
NEG D	(D)	2's Complement (Negative)

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D								*	*	*	*	*	*	*	*

- Note:
  1. If D is used in device F, only 16-bit instruction is applicable.

16-bit instruction (3 Steps)

NEG Continuous execution

32-bit instruction (5 Steps)

DNEG Continuous execution

- Flags: None

Operands:

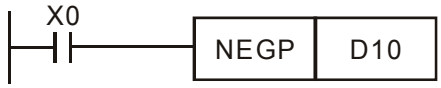
(D): Device to store 2's complement

Explanations:

1. This instruction converts a negative BIN value into an absolute value.
2. This instruction can convert a negative binary value into its absolute value.

Program Example 1:

When X0 = Off→On, the phase of every bit of the content in D10 will be reversed (0→1, 1→0) and pluses 1. The result will then be stored in D10.

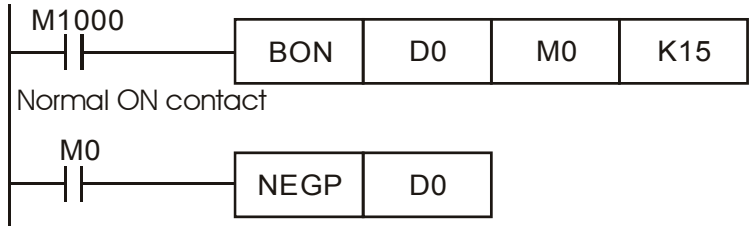


Program Example 2:

Obtaining the absolute value of a negative value:

a) When the 15th bit of D0 is "1", M0 = On. (D0 is a negative value).

b) When M0 = Off→On, NEG instruction will obtain 2's complement of D0 and further its absolute value.



Program Example 3:

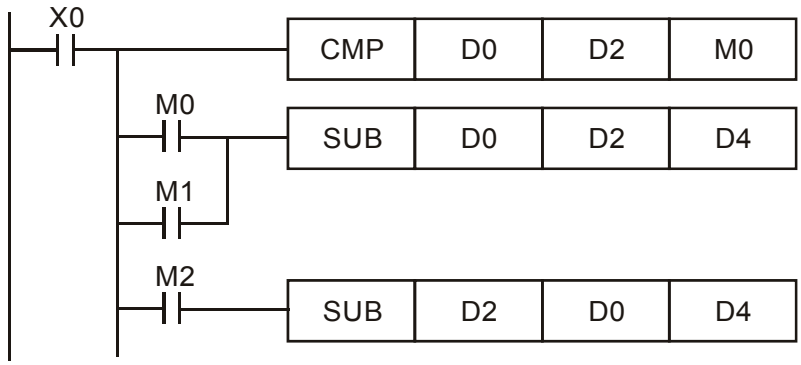
Obtaining the absolute value by the remainder of the subtraction. When X0 = On,

a) If D0 > D2, M0 = On.

b) If D0 = D2, M1 = On.

c) If D0 < D2, M2 = On.

d) D4 is then able to remain positive.

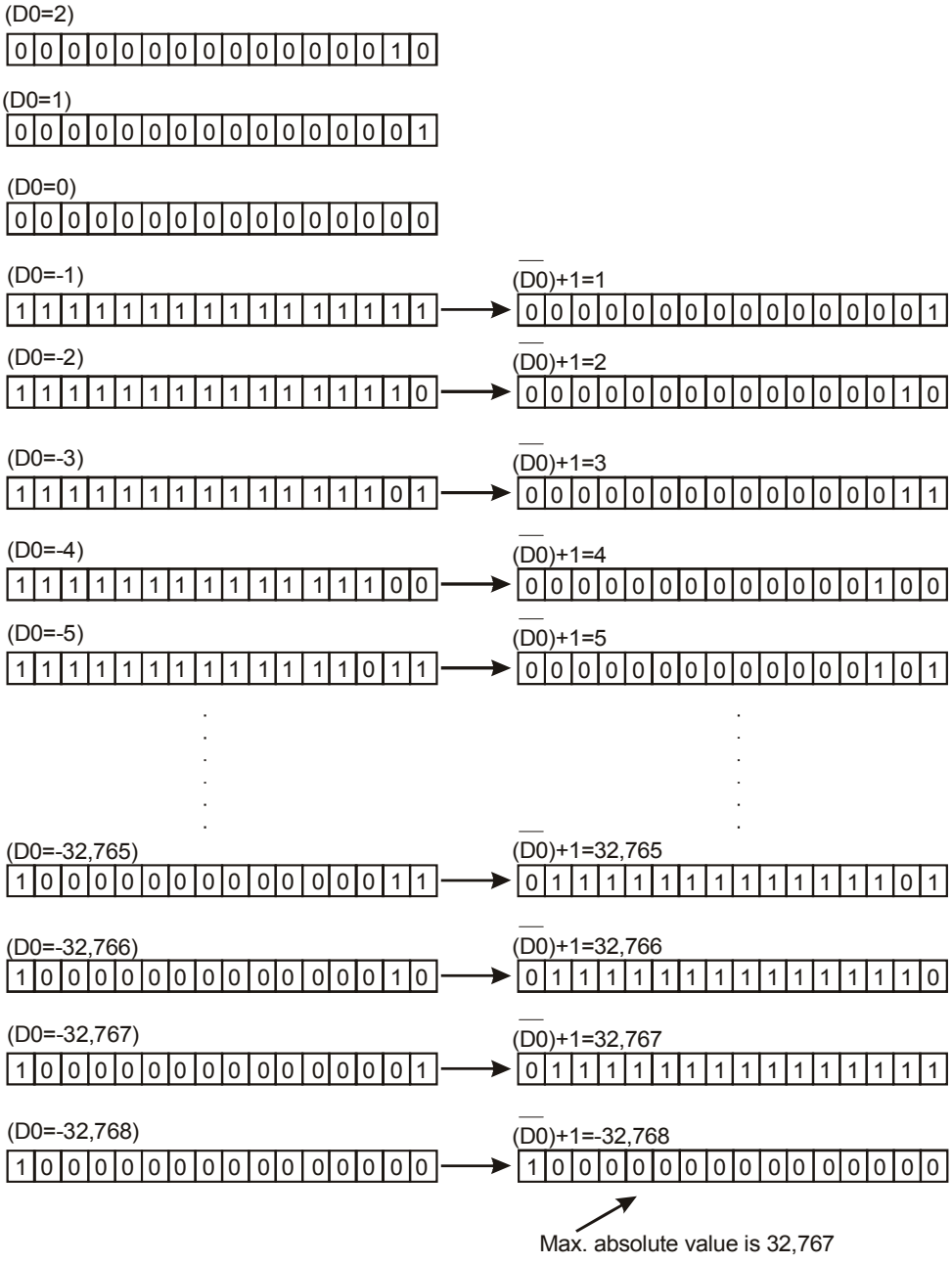


Remarks:

Negative value and its absolute value

a) The sign of a value is indicated by the highest (most left) bit in the register. 0 indicates that the value is a positive one and 1 indicates that the value is a negative one.

b) NEG instruction is able to convert a negative value into its absolute value.



Mnemonic	Operands	Function
ROR D	(D) (n)	Rotation Right

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D								*	*	*	*	*	*	*	*
n					*	*									

- Note:
  1. If D is used in device F, only 16-bit instruction is applicable.
  2. If D is designated as KnY, KnM, and KnS, only K4 (16-bit) and K8 (32-bit) are valid.
  3. Range of n: 1 ~ 16 (16-bit); 1 ~ 32 (32-bit)

16-bit instruction (5 Steps)

ROR Continuous execution

32-bit instruction (9 Steps)

DROR Continuous execution

- Flags: M1022 (Carry flag)

Operands:

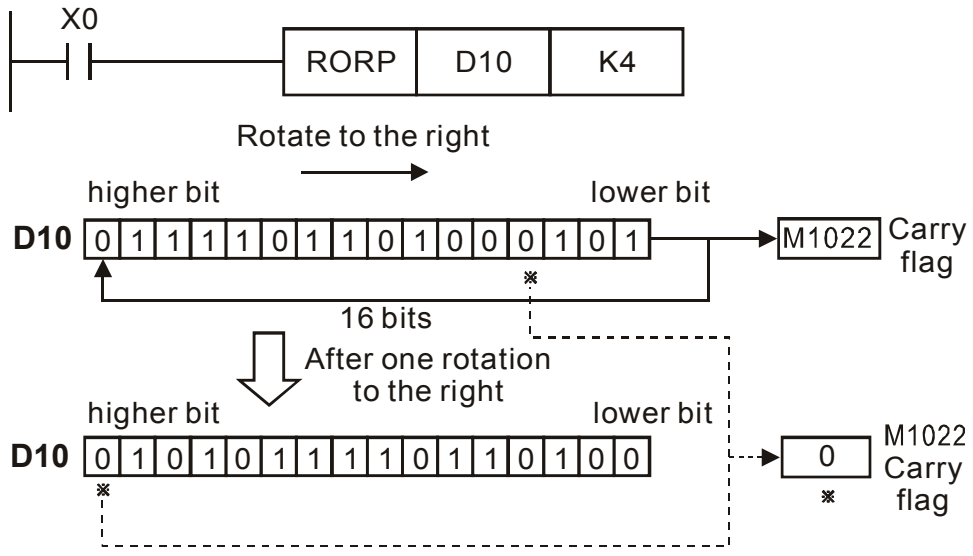
(D): Device to be rotated (n): Number of bits to be rotated in 1 rotation

Explanations:

This instruction rotates the device content designated by **D** to the right for **n** bits.

Program Example:

When X0 = Off→On, the 16 bits (4 bits as a group) in D10 will rotate to the right, as shown in the figure below. The bit marked with ※ will be sent to carry flag M1022.



Mnemonic		Operands	Function
ROL	**D**	(D) (n)	Rotation Left

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D								*	*	*	*	*	*	*	*
n					*	*									

- Note:
  1. If D is used in device F, only 16-bit instruction is applicable.
  2. If D is designated as KnY, KnM, and KnS, only K4 (16-bit) and K8 (32-bit) are valid.
  3. Range of n: 1 ~ 16 (16-bit); 1 ~ 32 (32-bit)

16-bit instruction (5 Steps)

ROL Continuous execution

32-bit instruction (9 Steps)

DROL Continuous execution

- Flags: M1022 (Carry flag)

Operands:

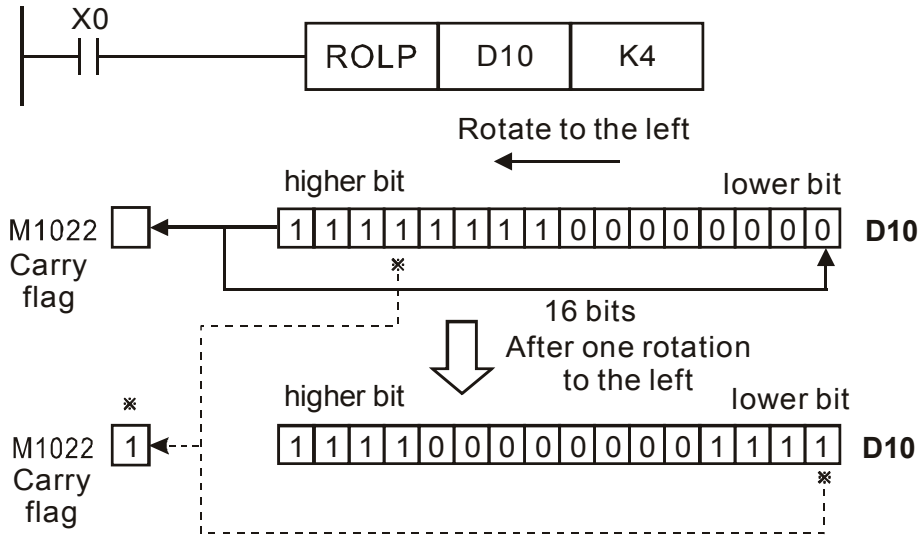
(D): Device to be rotated (n): Number of bits to be rotated in 1 rotation

Explanations:

This instruction rotates the device content designated by **D** to the left for **n** bits.

Program Example:

When X0 = Off→On, the 16 bits (4 bits as a group) in D10 will rotate to the left, as shown in the figure below. The bit marked with ※ will be sent to carry flag M1022.



Mnemonic	Operands	Function
RCR **D**	(D) (n)	Rotation Right with Carry

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D								*	*	*	*	*	*	*	*
n					*	*									

- Note:
  1. If D is used in device F, only 16-bit instruction is applicable.
  2. If D is designated as KnY, KnM, and KnS, only K4 (16-bit) and K8 (32-bit) are valid.
  3. Range of n: 1 ~ 16 (16-bit); 1 ~ 32 (32-bit)

16-bit instruction (5 Steps)

RCR Continuous execution

32-bit instruction (9 Steps)

DRCR Continuous execution

- Flags: M1022 (Carry flag)

Operands:

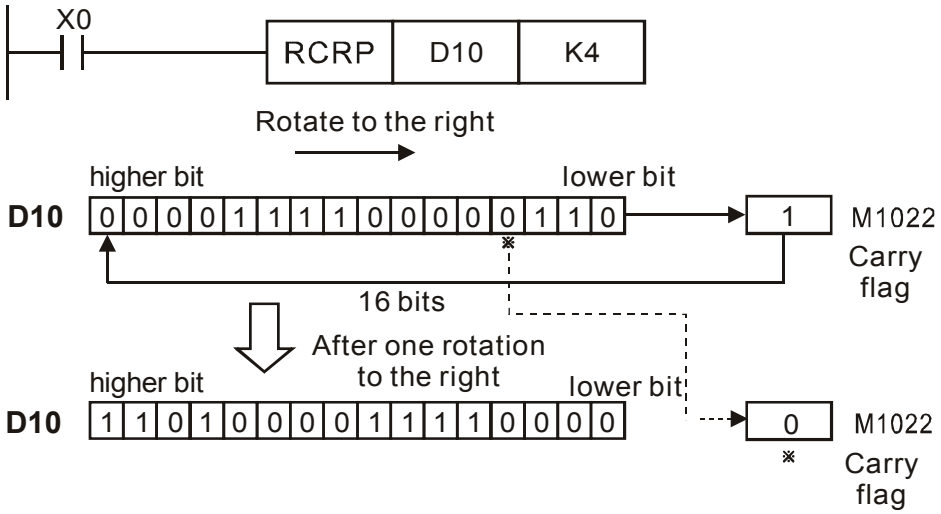
(D): Device to be rotated (n): Number of bits to be rotated in 1 rotation

Explanations:

This instruction rotates the device content designated by **D** together with carry flag M1022 to the right for **n** bits.

Program Example:

When X0 = Off→On, the 16 bits (4 bits as a group) in D10 together with carry flag M1022 (total 17 bits) will rotate to the right, as shown in the figure below. The bit marked with ※ will be sent to carry flag M1022.



Mnemonic	Operands	Function
RCL **D**	(D) (n)	Rotation Left with Carry

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D								*	*	*	*	*	*	*	*
n					*	*									

- Note:
  1. If D is used in device F, only 16-bit instruction is applicable.
  2. If D is designated as KnY, KnM, and KnS, only K4 (16-bit) and K8 (32-bit) are valid.
  3. Range of n: 1 ~ 16 (16-bit); 1 ~ 32 (32-bit)

16-bit instruction (5 Steps)

RCL Continuous execution

32-bit instruction (9 Steps)

DRCL Continuous execution

- Flags: M1022 (Carry flag)

Operands:

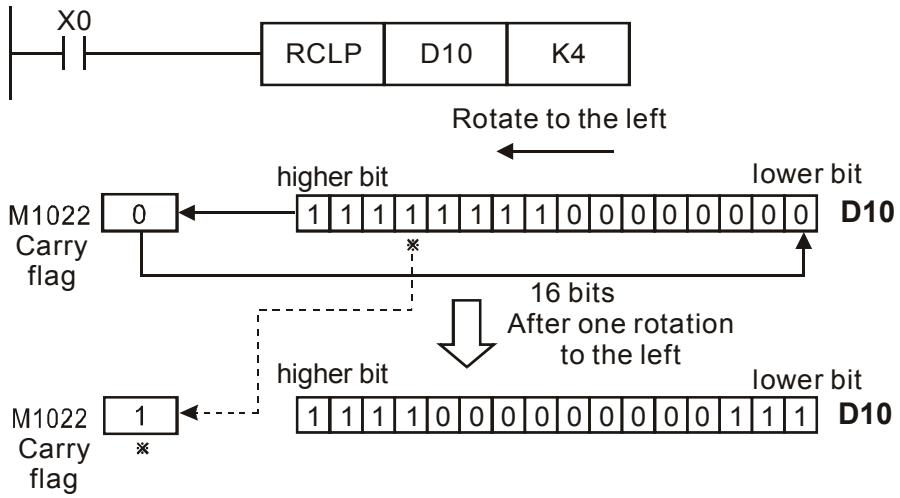
(D): Device to be rotated (n): Number of bits to be rotated in 1 rotation

Explanations:

This instruction rotates the device content designated by **D** together with carry flag M1022 to the left for **n** bits.

Program Example:

When X0 = Off→On, the 16 bits (4 bits as a group) in D10 together with carry flag M1022 (total 17 bits) will rotate to the left, as shown in the figure below. The bit marked with ※ will be sent to carry flag M1022.



Mnemonic	Operands	Function
SFTR	(S) (D) (n1) (n2)	Bit Shift Right

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S	✳	✳	✳	✳											
D		✳	✳	✳											
n_1					✳	✳									
n_2					✳	✳									

- Note:
  1. Range of n_1: 1~ 1,024
  2. Range of n_2: 1~ n_1

16-bit instruction (9 Steps)

SFTR Continuous execution

32-bit instruction

- - - -

- Flags: None

Operands:

(S): Start No. of the shifted device (D): Start No. of the device to be shifted

(n1): Length of data to be shifted (n2): Number of bits to be shifted in 1 shift

Explanations:

This instruction shifts the bit device of **n_1** bits (desired length for shifted register) starting from **D** to the right for **n_2** bits. **S** is shifted into **D** for **n_2** bits to supplement empty bits.

Program Example:

When X0 = Off→On, M0 ~M15 will form 16 bits and shifts to the right (4 bits as a group). The figure below illustrates the right shift of the bits in one scan.

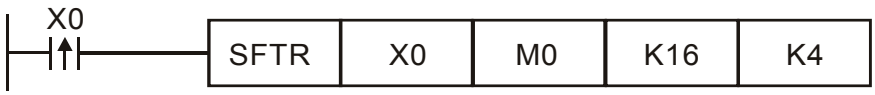
❶ M3 ~ M0 → carry

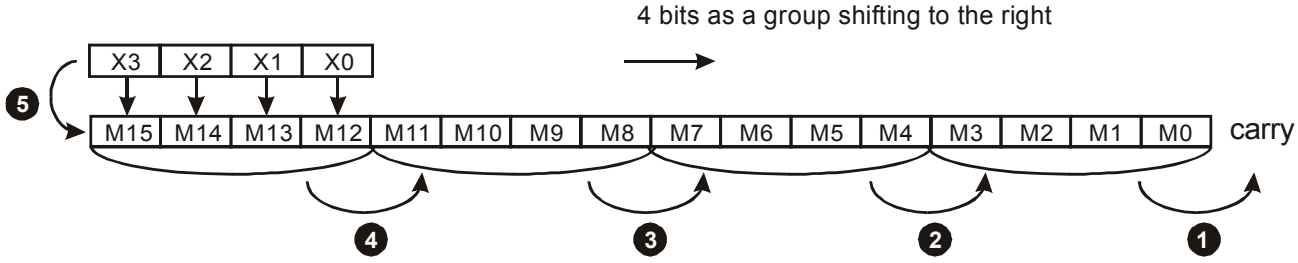
❷ M7 ~ M4 → M3 ~ M0

❸ M11 ~ M8 → M7 ~ M4

❹ M15 ~ M12 → M11 ~ M8

❺ X3 ~ X0 → M15 ~ M12 completed





Mnemonic	Operands	Function
SFTL	(S) (D) (n1) (n2)	Bit Shift Left

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S	✳	✳	✳	✳											
D		✳	✳	✳											
n_1					✳	✳									
n_2					✳	✳									

- Note:
  1. Range of n_1: 1~ 1,024
  2. Range of n_2: 1~ n_1

16-bit instruction (9 Steps)

SFTL	Continuous execution

32-bit instruction

- - - -

- Flags: None

Operands:

(S): Start No. of the shifted device (D): Start No. of the device to be shifted

(n1): Length of data to be shifted (n2): Number of bits to be shifted in 1 shift

Explanations:

This instruction shifts the bit device of **n_1** bits (desired length for shifted register) starting from **D** to the left for **n_2** bits. **S** is shifted into **D** for **n_2** bits to supplement empty bits.

Program Example:

When X0 = Off→On, M0 ~M15 will form 16 bits and shifts to the left (4 bits as a group). The figure below illustrates the left shift of the bits in one scan.

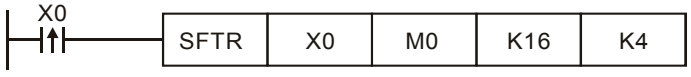
❶ M15 ~ M12 → carry

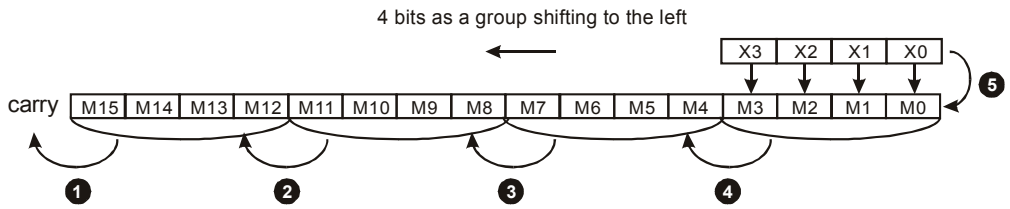
❷ M11 ~ M8 → M15 ~ M12

❸ M7 ~ M4 → M11 ~ M8

❹ M3 ~ M0 → M7 ~ M4

❺ X3 ~ X0 → M3 ~ M0 completed





Mnemonic	Operands	Function
ZRST	(D1) (D2)	Zero Reset

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D_1		✳	✳	✳							✳	✳	✳	✳	✳
D_2		✳	✳	✳							✳	✳	✳	✳	✳

- Note:
  1. Number of operand D1 ≤ Number of operand D2.
  2. D_1 and D_2 have to designate devices of the same type.

16-bit instruction (5 Steps)

ZRST Continuous execution

32-bit instruction

\- - - -

- Flags: None

Operands:

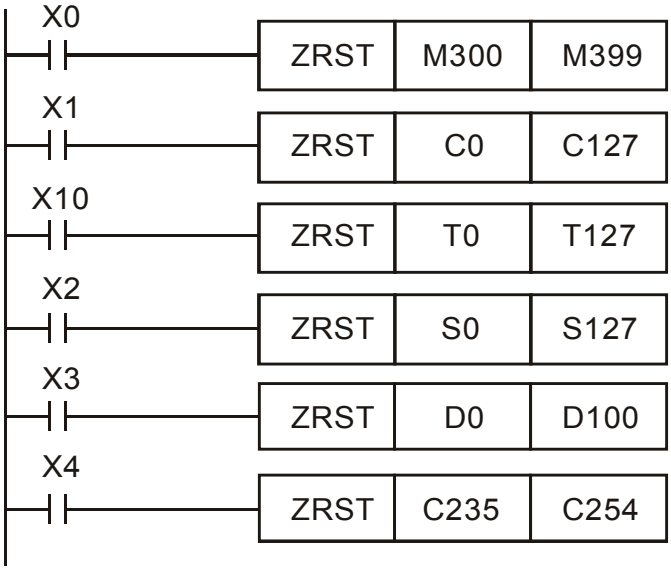
(D1): Start device of the range to be reset (D2): End device of the range to be reset

Explanations:

1. When the instruction is executed, area from **D_1** to **D_2** will be cleared.
2. 16-bit counter and 32-bit counter cannot use ZRST instruction together.
3. When **D_1** > **D_2**, only operands designated by **D_2** will be reset.

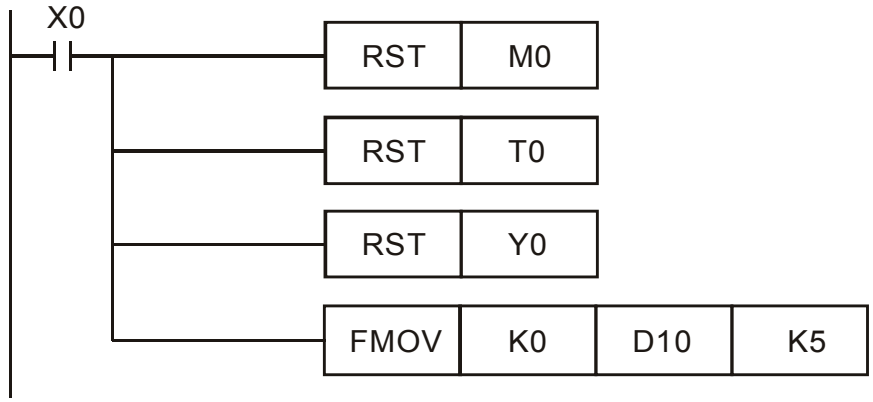
Program Example:

1. When X0 = On, auxiliary relays M300 ~ M399 will be reset to Off.
2. When X1 = On, 16 counters C0 ~ C127 will all be reset (writing in 0; contact and coil being reset to Off).
3. When X2 = On, steps S0 ~ S127 will be reset to Off.
4. When X3 = On, data registers D0 ~ D100 will be reset to 0.
5. When X4 = On, 32-bit counters C235 ~ C254 will all be reset. (writing in 0; contact and coil being reset to Off)



Remarks:

1. Devices, e.g. bit devices Y, M, S and word devices T, C, D, can use RST instruction.
2. FMOV instruction can be also used to send K0 to word devices T, C, D or bit registers KnY, KnM, KnS for reset.



Mnemonic		Operands	Function
SUM	**D**	(S) (D)	Sum of Active Bits

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
D								✳	✳	✳	✳	✳	✳	✳	✳

- Note:
  1. If S and D is used in device F, only 16-bit instruction is applicable.

16-bit instruction (5 Steps)

SUM	Continuous execution

32-bit instruction (9 Steps)

DSUM	Continuous execution

- Flags: M1020 (Zero flag)

Operands:

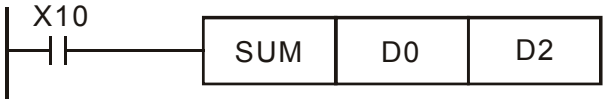
(S): Source device (D): Destination device for storing counted value

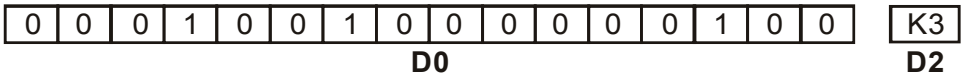
Explanations:

1. Among the bits of **S**, the total of bits whose content is “1” will be stored in **D.**
2. When all the 16 bits of **S** are “0”, zero flag M1020 = On.
3. When 32- instruction is in use, **D** will occupy 2 registers.

Program Example:

When X10 = On, among the 16 bits of D0, the total of bits whose content is “1” will be stored in D2.





Mnemonic		Operands	Function
BON	**D**	(S) (D) (n)	Check Specified Bit Status

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S					✳	✳	✳	✳	✳	✳	✳	✳	✳	✳	✳
D		✳	✳	✳											
n					✳	✳									

- Note:
  1. If S is used in device F, only 16-bit instruction is applicable.
  2. Range of n: 0 ~ 15 (16-bit instruction); 0 ~ 31 (32-bit instruction)

16-bit instruction (7 Steps)

BON Continuous execution

32-bit instruction (13 Steps)

DBON Continuous execution

- Flags: None

Operands:

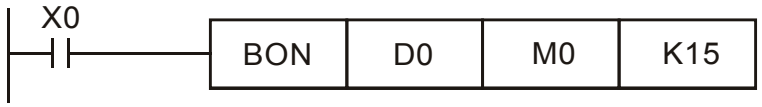
(S): Source device (D): Device for storing check result (n): Bits specified for check

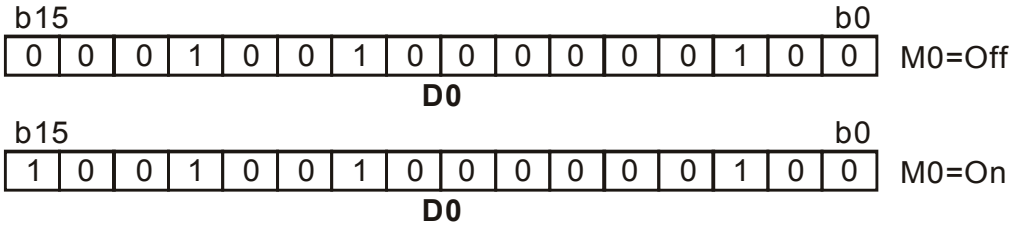
Explanations:

When the **n**th bit of **S** is "1", D = On; when the **n**th bit of **S** is "0", D = Off.

Program Example:

1. When X0 = On, assume the 15th bit of D0 is "1", and M0 = On. Assume the 15th bit of D0 is "0", and M0 = Off.
2. When X0 goes Off, M0 will remains in its previous status.





Mnemonic		Operands	Function
MEAN	**D**	(S) (D) (n)	Mean

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S							*	*	*	*	*	*	*		
D								*	*	*	*	*	*	*	*
n					*	*									

- Note:
  1. If D is used in device F, only 16-bit instruction is applicable.
  2. Range of n: 1 ~ 64

16-bit instruction (7 Steps)

MEAN Continuous execution

32-bit instruction (13 Steps)

DMEAN Continuous execution

- Flags: None

Operands:

(S): Start device to obtain mean value (D): Destination device for storing mean value

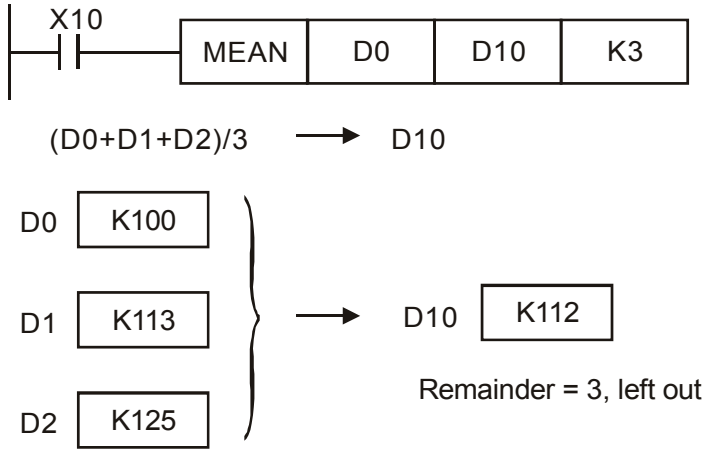
(n): The number of consecutive source devices used

Explanations:

1. After the content of **n** devices starting from **S** are added up, the mean value of the result will be stored in **D**.
2. Remainders in the operation will be left out.
3. Provided the No. of designated device exceeds its normal range, only the No. within the normal range can be processed.
4. If **n** falls without the range of 1 ~ 64, DOP-EXIO series will determine it as an “instruction operation error”.

Program Example:

When X10 = On, the contents in 3 (n = 3) registers starting from D0 will be summed and then divided by 3. The obtained mean value will be stored in D10 and the remainder will be left out.



Mnemonic	Operands	Function
REF	(D) (n)	Refresh

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D	✳	✳													
n					✳	✳									

- Note:
  1. D must designate X0, X10, Y0, Y10…the points whose 1s digit is "0". See remarks for more details.
  2. Range of n: 8 ~ 256 (has to be the multiple of 8).

16-bit instruction (5 Steps)

REF Continuous execution

32-bit instruction

\- - - -

- Flags: None

Operands:

(D): Start device to be I/O refreshed (n): Number of items to be I/O refreshed

Explanations:

1. The status of all input/output terminals of DOP-EXIO series will be updated after the program scans to END. When the program starts to scan, the status of the external input terminal is read and stored into the memory of the input point. The output terminal will send the content in the output memory to the output device after END instruction is executed. Therefore, this instruction is applicable when the latest input/output data are needed for the operation.
2. REF command can be used between FOR and NEXT instruction, and between CJ instructions. If there is an interrupt occurs in the period of time when input/output terminals is working, REF command can also be used. It can be also used to interrupt the subroutine program.
3. The operand **D** should always be a multiple of 10, i.e. 00, 10, 20, 30… etc., so it should be X0, X10, Y0, Y10… etc. The operand **n** hould always be a multiple of 8, i.e. 8, 16, 24, 32…etc. and its available range is 8~256. If the value of **n** is out of the stated range (8~256) or not a multiple of 8, an "operation error" will be generated.

Program Example 1:

When X0 = On, DOP-EXIO series will read the status of input points X0 ~ X7 immediately and refresh the input signals without any input delay.

Program Example 2:

When X0 = On, the 8 output signal from Y0 ~ Y7 will be sent to output terminals and refreshed. But there is 10ms input delay occurred on the input signals.

Mnemonic	Operands	Function
ALT	(D)	Alternate State

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D		*	*	*											

16-bit instruction (3 Steps)

ALT Continuous execution

32-bit instruction

- - - -

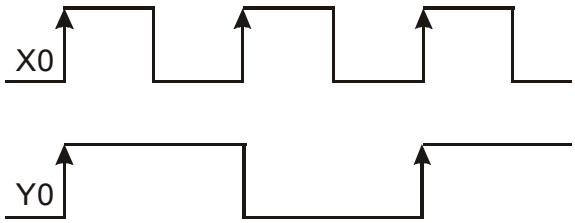
- Flags: None

Operands:

(D): Destination device

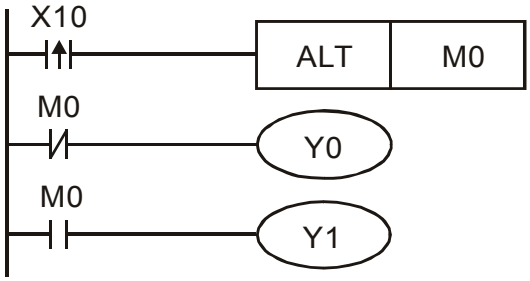
Program Example 1:

When X0 goes from Off to On for the first time, Y0 will be On. When X0 goes from Off to On for the second time, Y0 will be Off.



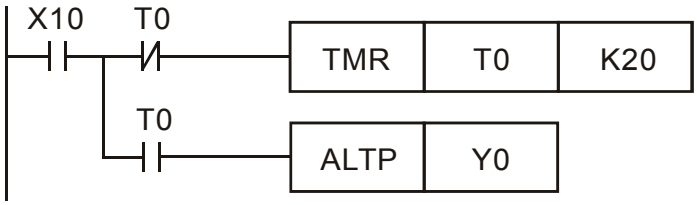
Program Example 2:

Using a single switch to enable and disable control. At the beginning, M0 = Off, so Y0 = On and Y1 = Off. When X10 switches between On/Off for the first time, M0 will be On, so Y1 = On and Y0 = Off. For the second time of On/Off switching, M0 will be Off, so Y0 = On and Y1 = Off.



Program Example 3:

Generating flashing. When X10 = On, T0 will generate a pulse every 2 seconds and Y0 output will switch between On and Off following the T0 pulses.


X10
T0
TMR
T0
K20
T0
ALTP
Y0

Mnemonic	Operands	Function
ASCI	(S) (D) (n)	Convert Hex to ASCII

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S					*	*	*	*	*	*	*	*	*	*	*
D								*	*	*	*	*	*		
n					*	*									

- Note:
  1. Range of n: 1 ~ 256

16-bit instruction (7 Steps)

ASCI	Continuous execution

32-bit instruction

- - - -

- Flags: M1161 (8/16 bit mode switch)

Operands:

(S): Start device for source data (D): Start device for storing the converted result

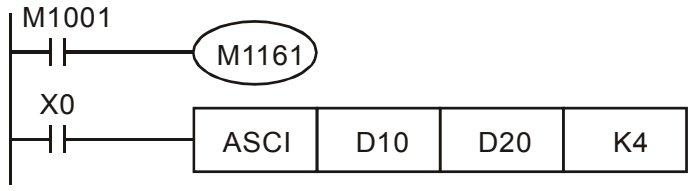
(n): Number of bits to be converted

Explanations:

1. 16-bit conversion mode: When M1161 = Off, the instruction converts every bit of the hex data in **S** into ASCII codes and send them to the 8 high bits and 8 low bits of **D**. **n** = the converted number of bits.
2. 8-bit conversion mode: When M1161 = On, the instruction converts every bit of the hex data in **S** into ASCII codes and send them to the 8 low bits of **D**. **n** = the number of converted bits. (All 8 high bits of **D** = 0)

Program Example 1:

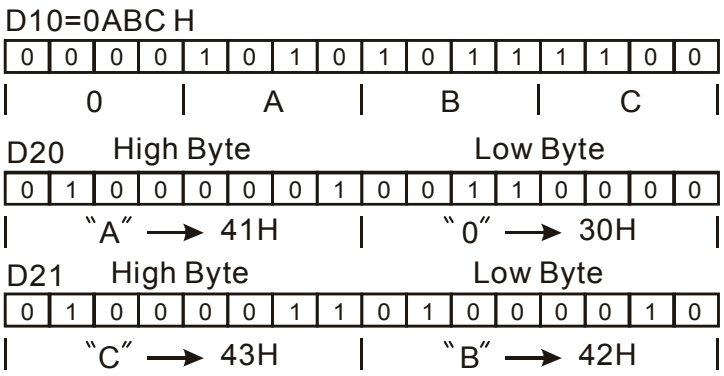
1. M1161 = Off: The 16-bit conversion mode
2. When X0 = On, convert the 4 hex values in D10 into ASCII codes and send the result to registers starting from D20.



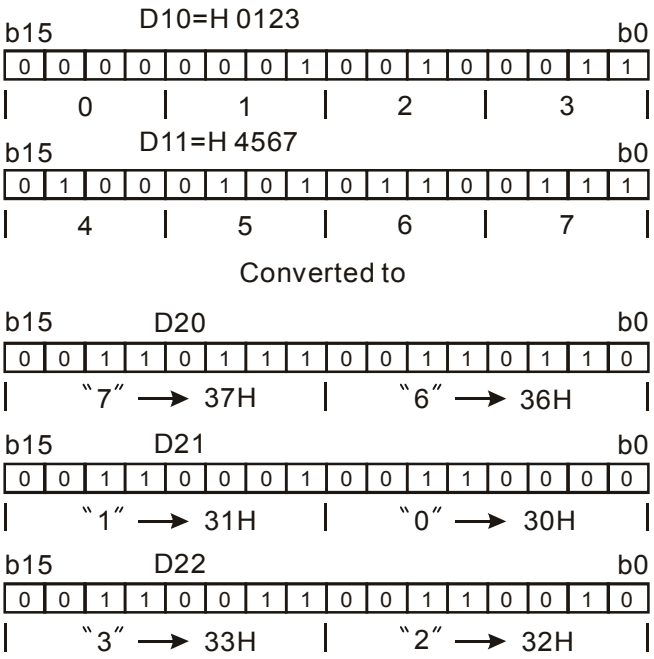
3. Assume

(D10) = 0ABC H	'0' = 30H	'1' = 31H	'5' = 35H
(D11) = 1234 H	'A' = 41H	'2' = 32H	'6' = 36H
(D12) = 5678 H	'B' = 42H	'3' = 33H	'7' = 37H
	'C' = 43H	'4' = 34H	'8' = 38H

4. When **n** = 4, the bit structure will be as:



5. When **n** = 6, the bit structure will be as:



6. When **n** = 1 ~ 16:

D \ n	K1	K2	K3	K4	K5	K6	K7	K8
D20 Low Byte	“3”	“2”	“1”	“0”	“7”	“6”	“5”	“4”
D20 High Byte		“3”	“2”	“1”	“0”	“7”	“6”	“5”
D21 Low Byte			“3”	“2”	“1”	“0”	“7”	“6”
D21 High Byte				“3”	“2”	“1”	“0”	“7”
D22 Low Byte					“3”	“2”	“1”	“0”
D22 High Byte						“3”	“2”	“1”
D23 Low Byte							“3”	“2”
D23 High Byte								“3”
D24 Low Byte								
D24 High Byte				No Change				
D25 Low Byte								
D25 High Byte								
D26 Low Byte								
D26 High Byte								
D27 Low Byte								
D27 High Byte								

D \ n	K9	K10	K11	K12	K13	K14	K15	K16
D20 Low Byte	“B”	“A”	“9”	“8”	“F”	“E”	“D”	“C”
D20 High Byte	“4”	“B”	“A”	“9”	“8”	“F”	“E”	“D”
D21 Low Byte	“5”	“4”	“B”	“A”	“9”	“8”	“F”	“E”
D21 High Byte	“6”	“5”	“4”	“B”	“A”	“9”	“8”	“F”
D22 Low Byte	“7”	“6”	“5”	“4”	“B”	“A”	“9”	“8”
D22 High Byte	“0”	“7”	“6”	“5”	“4”	“B”	“A”	“9”
D23 Low Byte	“1”	“0”	“7”	“6”	“5”	“4”	“B”	“A”
D23 High Byte	“2”	“1”	“0”	“7”	“6”	“5”	“4”	“B”
D24 Low Byte	“3”	“2”	“1”	“0”	“7”	“6”	“5”	“4”
D24 High Byte		“3”	“2”	“1”	“0”	“7”	“6”	“5”
D25 Low Byte			“3”	“2”	“1”	“0”	“7”	“6”
D25 High Byte				“3”	“2”	“1”	“0”	“7”
D26 Low Byte					“3”	“2”	“1”	“0”
D26 High Byte			No Change			“3”	“2”	“1”
D27 Low Byte							“3”	“2”
D27 High Byte								“3”

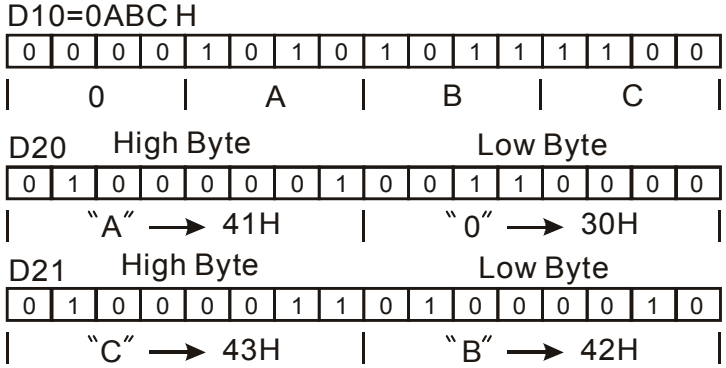
Program Example 2:

1. M1161 = On: The 8-bit conversion mode
2. When X0 = On, convert the 4 hex values in D10 into ASCII codes and send the result to registers starting from D20.

3. Assume

(D10) = 0ABC H	‘0’ = 30H	‘1’ = 31H	‘5’ = 35H
(D11) = 1234 H	‘A’ = 41H	‘2’ = 32H	‘6’ = 36H
(D12) = 5678 H	‘B’ = 42H	‘3’ = 33H	‘7’ = 37H
	‘C’ = 43H	‘4’ = 34H	‘8’ = 38H

4. When **n** = 2, the bit structure will be as:



D10=0ABC H

0	0	0	0	1	0	1	0	1	0	1	1	1	1	0	0
0				A				B				C			

ASCII code of D20=B is 42H

0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
								4				2			

ASCII code of D21=C is 43H

0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
								4				3			

5. When **n** = 4, the bit structure will be as:

D10= H 0123

b15 b0

0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1
0				1				2				3			

Converted to

b15 D20 b0

0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
										"0" → 30H					

b15 D21 b0

0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
										"1" → 31H					

b15 D22 b0

0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
										"2" → 32H					

b15 D23 b0

0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
										"3" → 33H					

6. When **n** = 1 ~ 16:

D \ n	K1	K2	K3	K4	K5	K6	K7	K8
D20	"3"	"2"	"1"	"0"	"7"	"6"	"5"	"4"
D21		"3"	"2"	"1"	"0"	"7"	"6"	"5"
D22			"3"	"2"	"1"	"0"	"7"	"6"
D23				"3"	"2"	"1"	"0"	"7"
D24					"3"	"2"	"1"	"0"
D25						"3"	"2"	"1"
D26							"3"	"2"
D27								"3"
D28								
D29				No Change				
D30								
D31								
D32								
D33								
D34								
D35								

D \ n	K9	K10	K11	K12	K13	K14	K15	K16
D20	“B”	“A”	“9”	“8”	“F”	“E”	“D”	“C”
D21	“4”	“B”	“A”	“9”	“8”	“F”	“E”	“D”
D22	“5”	“4”	“B”	“A”	“9”	“8”	“F”	“E”
D23	“6”	“5”	“4”	“B”	“A”	“9”	“8”	“F”
D24	“7”	“6”	“5”	“4”	“B”	“A”	“9”	“8”
D25	“0”	“7”	“6”	“5”	“4”	“B”	“A”	“9”
D26	“1”	“0”	“7”	“6”	“5”	“4”	“B”	“A”
D27	“2”	“1”	“0”	“7”	“6”	“5”	“4”	“B”
D28	“3”	“2”	“1”	“0”	“7”	“6”	“5”	“4”
D29	No Change	“3”	“2”	“1”	“0”	“7”	“6”	“5”
D30			“3”	“2”	“1”	“0”	“7”	“6”
D31				“3”	“2”	“1”	“0”	“7”
D32					“3”	“2”	“1”	“0”
D33						“3”	“2”	“1”
D34							“3”	“2”
D35								“3”

Mnemonic	Operands	Function
HEX	(S) (D) (n)	Convert ASCII to Hex

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S					*	*	*	*	*	*	*	*	*		
D								*	*	*	*	*	*	*	*
n					*	*									

- Note:
  1. Range of n: 1 ~ 256

16-bit instruction (7 Steps)

HEX Continuous execution

32-bit instruction

- - - -

- Flags: M1161 (8/16 bit mode switch)

Operands:

(S): Start device for source data (D): Start device for storing the converted result

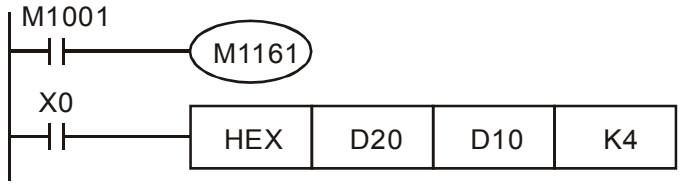
(n): Number of bits to be converted

Explanations:

1. 16-bit conversion mode: When M1161 = Off, the instruction is in 16-bit conversion mode. ASCII codes of the 8 high bits and 8 low bits of the hex data in **S** are converted into hex value and sent to **D** (every 4 bits as a group). **n** = the number of bits converted into ASCII codes.
2. 8-bit conversion mode: When M1161 = On, the instruction is in 8-bit conversion mode. Every bit of the hex data in **S** are converted into ASCII codes and sent to the 8 low bits of **D**. **n** = the number of converted bits. (All 8 high bits of **D** = 0)

Program Example 1:

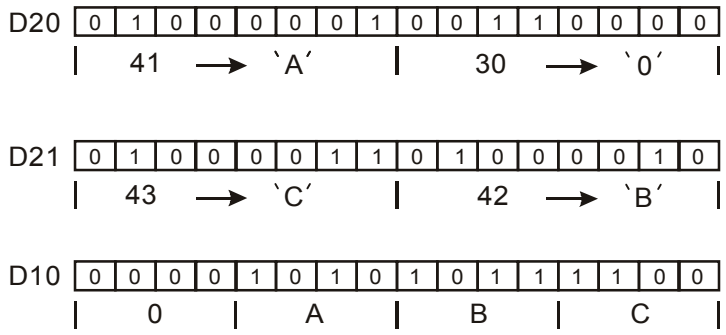
1. M1161 = Off: The 16-bit conversion mode
2. When X0 = On, convert the ASCII codes stored in the registers starting from D20 into hex value and send the result (every 4 bits as a group) to registers starting from D10. **n** = 4.



3. Assume

S	ASCII code	Converted to hex	S	ASCII code	Converted to hex
D20 low byte	H 43	"C"	D24 low byte	H 34	"4"
D20 high byte	H 44	"D"	D24 high byte	H 35	"5"
D21 low byte	H 45	"E"	D25 low byte	H 36	"6"
D21 high byte	H 46	"F"	D25 high byte	H 37	"7"
D22 low byte	H 38	"8"	D26 low byte	H 30	"0"
D22 high byte	H 39	"9"	D26 high byte	H 31	"1"
D23 low byte	H 41	"A"	D27 low byte	H 32	"2"
D23 high byte	H 42	"B"	D27 high byte	H 33	"3"

4. When **n** = 4, the bit structure will be as:

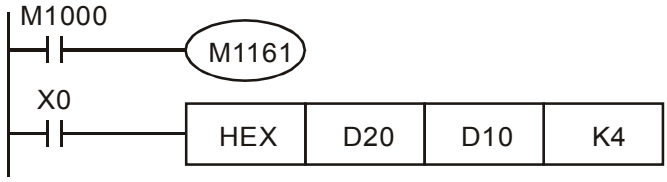


5. When **n** = 1 ~ 16:

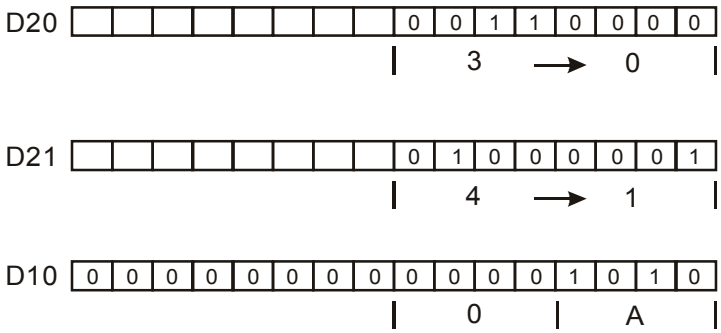
n \ D	D13	D12	D11	D10
1	The undesignated parts in the registers in use are all 0.			***C H
2				**CD H
3				*CDE H
4				CDEF H
5			***C H	DEF8 H
6			**CD H	EF89 H
7			*CDE H	F89A H
8			CDEF H	89AB H
9		***C H	DEF8 H	9AB4 H
10		**CD H	EF89 H	AB45 H
11		*CDE H	F89A H	B456 H
12		CDEF H	89AB H	4567 H
13	***C H	DEF8 H	9AB4 H	5670 H
14	**CD H	EF89 H	AB45 H	6701 H
15	*CDE H	F89A H	B456 H	7012 H
16	CDEF H	89AB H	4567 H	0123 H

Program Example 2:

1. M1161 = On: The 8-bit converstion mode



2. When **n** = 4, the bit structure will be as:



3. Assume

S	ASCII code	Converted to hex	S	ASCII code	Converted to hex
D20	H 43	“C”	D28	H 34	“4”
D21	H 44	“D”	D29	H 35	“5”
D22	H 45	“E”	D30	H 36	“6”
D23	H 46	“F”	D31	H 37	“7”
D24	H 38	“8”	D32	H 30	“0”
D25	H 39	“9”	D33	H 31	“1”
D26	H 41	“A”	D34	H 32	“2”
D27	H 42	“B”	D35	H 33	“3”

4. When **n** = 1 ~ 16:

D / n	D13	D12	D11	D10
1	The undesignated parts in the registers in use are all 0.			***C H
2				**CD H
3				*CDE H
4				CDEF H
5			***C H	DEF8 H
6			**CD H	EF89 H
7			*CDE H	F89A H
8			CDEF H	89AB H
9		***C H	DEF8 H	9AB4 H
10		**CD H	EF89 H	AB45 H
11		*CDE H	F89A H	B456 H
12		CDEF H	89AB H	4567 H
13	***C H	DEF8 H	9AB4 H	5670 H
14	**CD H	EF89 H	AB45 H	6701 H
15	*CDE H	F89A H	B456 H	7012 H
16	CDEF H	89AB H	4567 H	0123 H

Mnemonic	Operands	Function
ABS **D**	(D)	Absolute Value

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
D					*	*	*	*	*	*	*	*	*	*	*

16-bit instruction (3 Steps)

ABS Continuous execution

32-bit instruction (5 Steps)

DABS Continuous execution

- Flags: None

Operands:

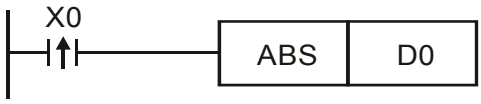
(D): Device of the absolute value

Explanations:

This instruction obtains the absolute value of the content in the designated in **D**.

Program Example:

When X0 = Off→On, obtain the absolute value of the content in D0.



Mnemonic		Operands	Function
SWAP	**D**	(S)	Byte Swap

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S								*	*	*	*	*	*	*	*

- Note:
  1. If D is used in device F, only 16-bit instruction is applicable.

16-bit instruction（5 Steps）

SWAP Continuous execution

32-bit instruction (9 Steps)

DSWAP Continuous execution

- Flags: None

Operands:

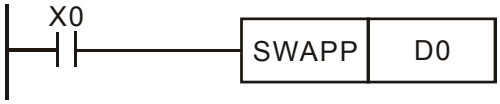
(S): Device for swapping 8 high/low byte.

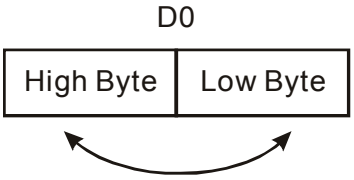
Explanations:

1. As 16-bit instruction: the contents in the 8 high bytes and 8 low bytes are swapped.
2. As 32-bit instruction: the 8 high bytes and 8 low bytes in the two registers swap with each other respectively.

Program Example 1:

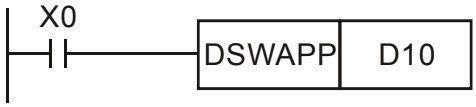
When X0 = On, the high 8 bytes and low 8 bytes in D0 will swap with each other.





Program Example 2:

When X0 = On, the high 8 bytes and low 8 bytes in D11 will swap with each other and the high 8 bytes and low 8 bytes in D10 will swap with each other.



Mnemonic		Operands	Function
LD※	**D**	(S1) (S2)	Contact Logical Operation LD※

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					*	*	*	*	*	*	*	*	*	*	*
S_2					*	*	*	*	*	*	*	*	*	*	*

• Note: ※: =, >, <, <>, ≦, ≧

16-bit instruction (5 Steps)

LD※	Continuous execution	-	-

32-bit instruction (9 Steps)

DLD※	Continuous execution	-	-

• Flags: None

Operands:

(S1): Data source device 1 (S2): Data source device 2

Explanations:

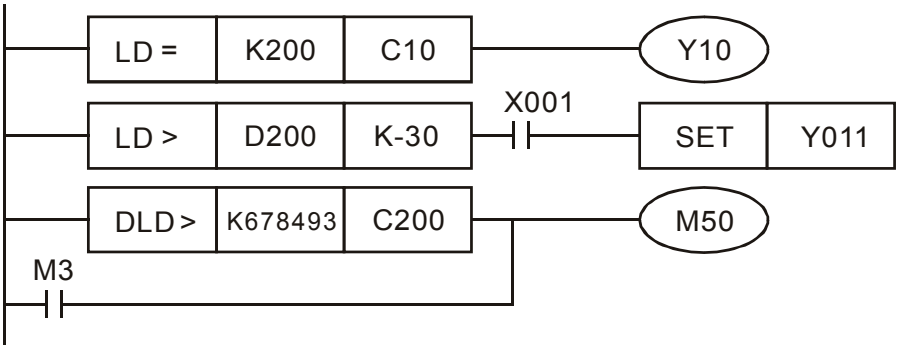
1. This instruction compares the content in **S_1** and **S_2**. If the result is not "equal", the continuity of the instruction is enabled. If the result is "equal", the continuity of the instruction is disabled.
2. LD※ (=, >, <, <>, ≦, ≧) instruction is used for direct connection with BUS.

16 -bit instruction	32 -bit instruction	Continuity condition	No-continuity condition
LD=	**D**LD=	(S1) = (S2)	(S1) ≠ (S2)
LD>	**D**LD>	(S1) > (S2)	(S1) ≦ (S2)
LD<	**D**LD<	(S1) < (S2)	(S1) ≧ (S2)
LD<>	**D**LD<>	(S1) ≠ (S2)	(S1) = (S2)
LD<=	**D**LD<=	(S1) ≦ (S2)	(S1) > (S2)
LD>=	**D**LD>=	(S1) ≧ (S2)	(S1) < (S2)

3. If the most left bit of **S_1** and **S_2** (16-bit instruction: b15、32-bit instruction: b31) is "1", the compare value will be regarded as the negative value for comparison.
4. When 32-bit counters (C200 ~) are used in this instruction for comparison, make sure to adopt 32-bit instruction (DLD※). If 16-bit instructions (LD※) is adopted, a "program error" will occur and the ERROR indicator on the panel will flash and the connecting controller can not run..

Program Example:

1. When the value of C0 is equal to the value of K200, Y10 = On.
2. When the value of D200 is higher than -29 and X1 = On, Y11 = On will be retained.
3. When the value of C200 is lower than 678,493 and M3 = On, M50 = On.


LD =
K200
C10
Y10
X001
LD >
D200
K-30
SET
Y011
DLD >
K678493
C200
M50
M3

Mnemonic		Operands	Function
AND※	**D**	(S1) (S2)	Contact Logical Operation AND※

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					＊	＊	＊	＊	＊	＊	＊	＊	＊	＊	＊
S_2					＊	＊	＊	＊	＊	＊	＊	＊	＊	＊	＊

• Note: ※: =, >, <, <>, ≦, ≧

16-bit instruction (5 Steps)

AND※	Continuous execution	-	-

32-bit instruction (9 Steps)

DAND※	Continuous execution	-	-

• Flags: None

Operands:

(S1): Data source device 1 (S2): Data source device 2

Explanations:

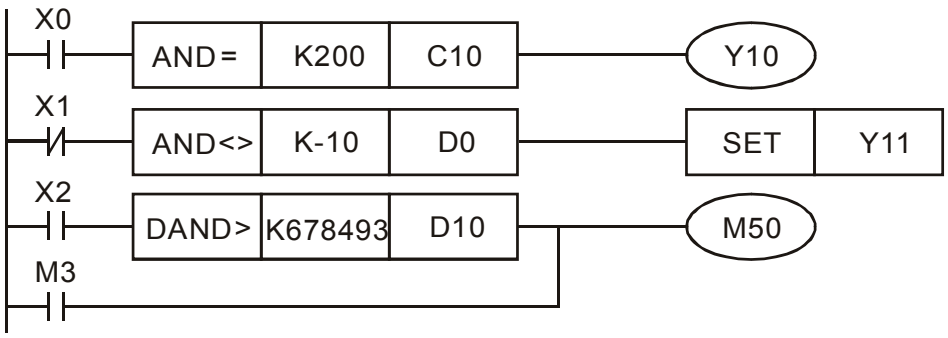
1. This instruction compares the content in **S_1** and **S_2**. If the result is not “equal”, the continuity of the instruction is enabled. If the result is “equal”, the continuity of the instruction is disabled.
2. AND※ (=, >, <, <>, ≦, ≧) instruction is used for direct connection with BUS.

16 -bit instruction	32 -bit instruction	Continuity condition	No-continuity condition
AND＝	**D**AND＝	(S1)＝(S2)	(S1)≠(S2)
AND＞	**D**AND＞	(S1)＞(S2)	(S1)≦(S2)
AND＜	**D**AND＜	(S1)＜(S2)	(S1)≧(S2)
AND＜＞	**D**AND＜＞	(S1)≠(S2)	(S1)＝(S2)
AND＜＝	**D**AND＜＝	(S1)≦(S2)	(S1)＞(S2)
AND＞＝	**D**AND＞＝	(S1)≧(S2)	(S1)＜(S2)

3. If the most left bit of **S_1** and **S_2** (16-bit instruction: b15、32-bit instruction: b31) is “1”, the compare value will be regarded as the negative value for comparison.
4. When 32-bit counters (C200 ~) are used in this instruction for comparison, make sure to adopt 32-bit instruction (DAND※). If 16-bit instructions (AND※) is adopted, a “program error” will occur and the ERROR indicator on the panel will flash and the connecting controller can not run..

Program Example:

1. When X0 = On and the value of C0 is equal to the value of K200, Y10 = On.
2. When X0 = Off and the value of D0 is not equal to -10 and X1 = On, Y11 = On will be retained.
3. When X2 = On and the value of (D11, D10) is lower than 678,493 and M3 = On, M50 = On.


X0
AND=
K200
C10
Y10
X1
AND<>
K-10
D0
SET
Y11
X2
DAND>
K678493
D10
M50
M3

Mnemonic		Operands	Function
OR※	**D**	(S1) (S2)	Contact Logical operation OR※

	Bit Devices				Word Devices										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1					＊	＊	＊	＊	＊	＊	＊	＊	＊	＊	＊
S_2					＊	＊	＊	＊	＊	＊	＊	＊	＊	＊	＊

• Note: ※: =, >, <, <>, ≦, ≧

16-bit instruction (5 Steps)

OR※	Continuous execution	-	-

32-bit instruction (9 Steps)

DOR※	Continuous execution	-	-

• Flags: None

Operands:

(S1): Data source device 1 (S2): Data source device 2

Explanations:

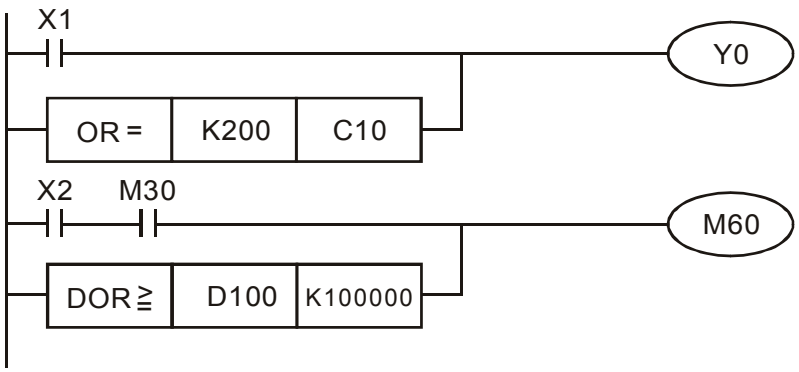
1. This instruction compares the content in **S_1** and **S_2**. If the result is not “equal”, the continuity of the instruction is enabled. If the result is “equal”, the continuity of the instruction is disabled.
2. OR※ (=, >, <, <>, ≦, ≧) instruction is used for direct connection with BUS.

16 -bit instruction	32 -bit instruction	Continuity condition	No-continuity condition
OR＝	**D**OR＝	(S1)＝(S2)	(S1)≠(S2)
OR＞	**D**OR＞	(S1)＞(S2)	(S1)≦(S2)
OR＜	**D**OR＜	(S1)＜(S2)	(S1)≧(S2)
OR＜＞	**D**OR＜＞	(S1)≠(S2)	(S1)＝(S2)
OR＜＝	**D**OR＜＝	(S1)≦(S2)	(S1)＞(S2)
OR＞＝	**D**OR＞＝	(S1)≧(S2)	(S1)＜(S2)

3. If the most left bit of **S_1** and **S_2** (16-bit instruction: b15、32-bit instruction: b31) is “1”, the compare value will be regarded as the negative value for comparison.
4. When 32-bit counters (C200 ~) are used in this instruction for comparison, make sure to adopt 32-bit instruction (DOR※). If 16-bit instructions (OR※) is adopted, a “program error” will occur and the ERROR indicator on the panel will flash and the connecting controller can not run..

Program Example:

1. When X1 = On, or the value of C0 is equal to the value of K200, Y10 = On.
2. When X2 and M30 are both, or the value of 32-bit data (D101, D100) is equal to or higher than K100,000, M60 = On.


X1
Y0
OR =
K200
C10
X2
M30
M60
DOR ≧
D100
K100000