



Cutler-Hammer

Eaton Logic Controller

System Manual

August 2005



Company Information

Eaton's electrical business is a global leader in electrical control, power distribution, and industrial automation products and services. Through advanced product development, world-class manufacturing methods, and global engineering services and support, Eaton's electrical business provides customer-driven solutions under brand names such as Cutler-Hammer®, Powerware®, Durant®, Heinemann®, Holec® and MEM®, which globally serve the changing needs of the industrial, utility, light commercial, residential, and OEM markets. For more information, visit **www.EatonElectrical.com**.

Eaton Corporation is a diversified industrial manufacturer with 2003 sales of \$8.1 billion. Eaton is a global leader in fluid power systems and services for industrial, mobile and aircraft equipment; electrical systems and components for power quality, distribution and control; automotive engine air management systems and powertrain controls for fuel economy; and intelligent drivetrain systems for fuel economy and safety in trucks. Eaton has 55,000 employees and sells products to customers in more than 100 countries. For more information, visit **www.eaton.com**.

Eaton Electrical
1000 Cherrington Parkway
Moon Township, PA 15108-4312
USA
Tel: 1-800-525-2000
www.EatonElectrical.com

EAT•N | **Cutler-Hammer**

© 2004 Eaton Corporation
All Rights Reserved
Printed in USA
Publication No. MN05003003E
August 2005



Cutler-Hammer

Eaton Logic Controller System Manual

Product Overview	1
Getting Started	2
Installation	3
ELC Concepts	4
Programming Concepts	5
Instruction Set	6
Communications	7
SFC Programming	8
Troubleshooting	9
Handheld Programmer	10

Safety Instructions

For the best results with the Eaton Logic Controller (ELC), carefully read this manual and all of the warning labels regarding the ELC before installing and operating it. Follow all instructions exactly and keep this manual handy for quick reference.

Safety notices are highlighted in this manual by a warning triangle and are marked as depending on the level of danger.



HIGH VOLTAGE: This symbol indicates high voltage. It calls your attention to items or operations that could be dangerous to you and other persons operating this equipment. Read the message and follow the instructions carefully.



WARNING: Indicates a potentially hazardous situation which, if not avoided, can result in serious injury or death.



CAUTION: Indicates a potentially hazardous situation which, if not avoided, can result in minor to moderate injury, or serious damage to the product. The situation described in the CAUTION may, if not avoided, lead to serious results. Important safety measures are described in CAUTION (as well as **WARNING**)

RESTRICTED RIGHTS

Use, duplication, or disclosure by the Government is subject to restrictions set forth in paragraph (b)(3)(B) of the Rights in Technical Data and Computer Software clause of DAR 7-104.9(a). Contractor/Manufacturer is Corporation, Operator Interface Business, 811 Green Crest Drive, Columbus, OH 43081.

TRADEMARKS

Commercial brand names (trademarks) of products of manufacturers or developers, other than Corporation or its affiliates, that appear in this manual may be registered or unregistered trademarks of those respective manufacturers or developers, which have expressed neither approval nor disapproval of Cutler-Hammer® products and services.

Preface

The Eaton Logic Controller for small applications is the “Just Right” solution to application needs. A wide range of logic controllers and modules simplify the decision process when solving applications solutions.

Information in this manual is subject to change without notice and does not represent a commitment on the part of Eaton Electrical Inc. Permission is granted to duplicate this material without modification only for your use or the internal use of other members of your company or your agents to assist you in the use and servicing of products purchased from Eaton Electrical. No permission is granted to modify this material or include this material in a compilation.

Audience

This manual provides information about installing and programming the Eaton Logic Controller and is designed for engineers, programmers, installers, and electricians who have a general knowledge of programmable logic controllers, ladder logic, Sequential Function Control (SFC), and general input and output devices.

Scope of this Manual

- ☐ ELC product family of processors
- ☐ ELC expansion modules
- ☐ ELCSOFT programming software for ELC products

Technical Assistance

The goal of Eaton Electrical is to ensure your greatest possible satisfaction with the operation of our products. We are dedicated to providing fast, friendly and accurate assistance. That is why we offer you so many ways to get the support you need. Whether it's by phone, fax or e-mail, you can access Eaton Electrical support information 24 hours a day, seven days a week. Our wide range of services is listed below.

You should contact your local distributor for product pricing, availability, ordering, expediting and repairs.

Ordering Products

Call the e-COM Support Center if you need assistance with placing an order, stock availability or proof of shipment, expediting an existing order, emergency shipments, product price information, returns other than warranty returns, and information on local distributors or sales offices.

e-COM Support Center**Voice: 800-356-1243 (8a.m.–6p.m. EST)****FAX: 800-752-8602****After-Hours Emergency: 800-543-7038
(6p.m.–8a.m. EST)**

Online Assistance

Use the Eaton Electrical website to find product information. You can also find information on local distributors or Eaton Electrical/Cutler-Hammer sales offices.

Website Addresswww.eatonelectrical.com

Phone Assistance

If you are in the US or Canada, and have ELC questions, you can take advantage of our toll-free line for technical assistance with hardware and software product selection, system design and installation, and system debugging and diagnostics. Technical support engineers are available for calls during regular business hours.

e-TRC**Technical Resource Center
(support for ELC products)****Voice:**

- 800-809-2772, selection 5 (8a.m.–5p.m. EST)
- 414-449-7100, selection 5 (8a.m.–5p.m. EST)

FAX: 614-882-0417**e-mail: CHATechSupport@eaton.com****After-Hours Emergency (Plant Down Only):**

- 800-809-2772, selection 5 (5p.m.–8a.m. EST)
- 414-449-7100, selection 5 (5p.m.–8a.m. EST)

Table of Contents

PREFACE	i
CHAPTER 1 – PRODUCT OVERVIEW	1-1
1.1 ELC Processors	1-2
1.2 ELC Digital Modules	1-4
1.3 ELC Specialty Modules	1-5
1.4 ELC Power Supplies	1-6
1.5 ELCSoft Programming Software	1-7
1.6 ELC-HHP	1-8
1.7 ELC Graphic Panels	1-8
1.8 ELCSoftGP Programming Software	1-10
CHAPTER 2 – GETTING STARTED	2-1
2.1 Connect ELC Power	2-2
2.2 Connect ELC to a PC	2-3
2.3 Connect ELC to the ELC-HHP	2-4
2.4 Current Consumed by the Control Unit	2-5
2.5 I/O Allocation	2-7
2.6 Install ELCSoft	2-9
2.7 Enter this Program	2-10
2.8 Download the Program	2-10
2.9 Monitor the Program	2-11
CHAPTER 3 – INSTALLATION	3-1
3.1 ELC Mounting and Clearance	3-2
3.2 Installing Modules	3-3
3.3 DIN Rail Installation	3-4
3.4 Wiring	3-5
3.5 Dimensions	3-15
3.6 Terminal Layouts	3-18

CHAPTER 4 – ELC CONCEPTS.....	4-1
4.1 ELC Scan Method.....	4-2
4.2 Current Flow	4-3
4.3 Contact A, Contact B	4-4
4.4 ELC Registers and Relays.....	4-4
4.5 Ladder Logic Symbols	4-5
4.6 Conversion of ELC Commands and Each Diagram Structure	4-11
4.7 ELC-HHP Programming Methods	4-12
4.8 Simplifying Ladder Logic.....	4-14
4.9 Basic Ladder Logic Examples	4-17
CHAPTER 5 – PROGRAMMING CONCEPTS	5-1
5.1 ELC Memory Map for PB model.....	5-2
5.2 ELC Memory Map for PC/PA/PH models	5-4
5.3 ELC Latched Memory Settings for PC/PA/PH Models	5-7
5.4 ELC Latched Memory Modes	5-8
5.5 ELC Bits, Nibbles, Bytes, Words, etc	5-8
5.6 Binary, Octal, Decimal, BCD, Hex	5-9
5.7 M Relay	5-11
5.8 S Relay	5-20
5.9 T (Timer)	5-20
5.10 C (Counter)	5-21
5.11 High-speed Counters.....	5-23
5.12 D (Word register)	5-28
5.13 E, F Index.....	5-38
5.14 File Register.....	5-38
5.15 Nest Level Pointer[N], Pointer[P] and Interrupt Pointer [I].....	5-39
5.16 Special M Relay and Special D Register Application	5-42
5.17 Fault Code Information	5-71

CHAPTER 6 –INSTRUCTION SET	6-1
6.1 Basic Instruction Explanations	6-2
6.2 Pointers	6-10
6.3 Interrupt Pointers	6-11
6.4 Basic Instructions (without API numbers)	6-13
6.5 Application Programming Instructions.....	6-14
6.6 Numerical List of Instructions	6-25
6.7 Detailed Instruction Explanation.....	6-33
 CHAPTER 7 – Communications	 7-1
7.1 Communication Ports	7-2
7.2 Communication Protocol ASCII mode.....	7-2
7.3 ELC Device Address	7-8
7.4 Command Code	7-9
7.5 ELC Device Addresses by Controller.....	7-17
 CHAPTER 8 – SFC Programming.....	 8-1
8.1 Step Ladder Command [STL], [RET]	8-2
8.2 Sequential Function Chart (SFC)	8-3
8.3 Step Ladder Command Explanation	8-5
8.4 Reminder of Design on the Step Ladder Program.....	8-11
8.5 Categories of Procedures.....	8-13
8.6 IST Command	8-24
 CHAPTER 9 – TROUBLESHOOTING.....	 9-1
9.1 Common Problems and Solution	9-2
9.2 Fault Code Table	9-4
9.3 Error Detection Devices	9-6
9.4 Periodic Inspection	9-6

CHAPTER 10 – HANDHELD PROGRAMMER	10-1
10.1 Introduction	10-2
10.2 ELC-HHP Standard Specification	10-4
10.3 Initial Setup	10-6
10.4 ELC-HHP Program Read/Write	10-14
10.5 Program Mode	10-26
10.6 ELC RUN/STOP Mode	10-36
10.7 MON/TEST Mode	10-37
10.8 Clear User's Program Memory	10-43
10.9 ELC and ELC-HHP Program Verifications	10-45
10.10 Parameters Settings	10-47
10.11 M_CARD Functions	10-50
10.12 File Register	10-51
10.13 Error Code Explanations	10-54
10.14 Instruction Table	10-57
10.15 Troubleshooting and Error Message	10-63
10.16 ELC-HHP Connection	10-64
10.17 ELC-HHP Operation Flow Chart	10-65

Product Overview

The Eaton Logic Controller (ELC) is programmable logic controller spanning an I/O of 10 –256 I/O points. ELC processors are so versatile they range from nano to small - PLCs I/O and application size without ever needing to change processors. ELC can control a wide variety of devices to solve your automation needs. Like most PLCs, ELC monitors inputs and modifies outputs as controlled by the user program. User program provides features like Boolean logic, counting, timing, complex math operations, and communications to other communicating products.

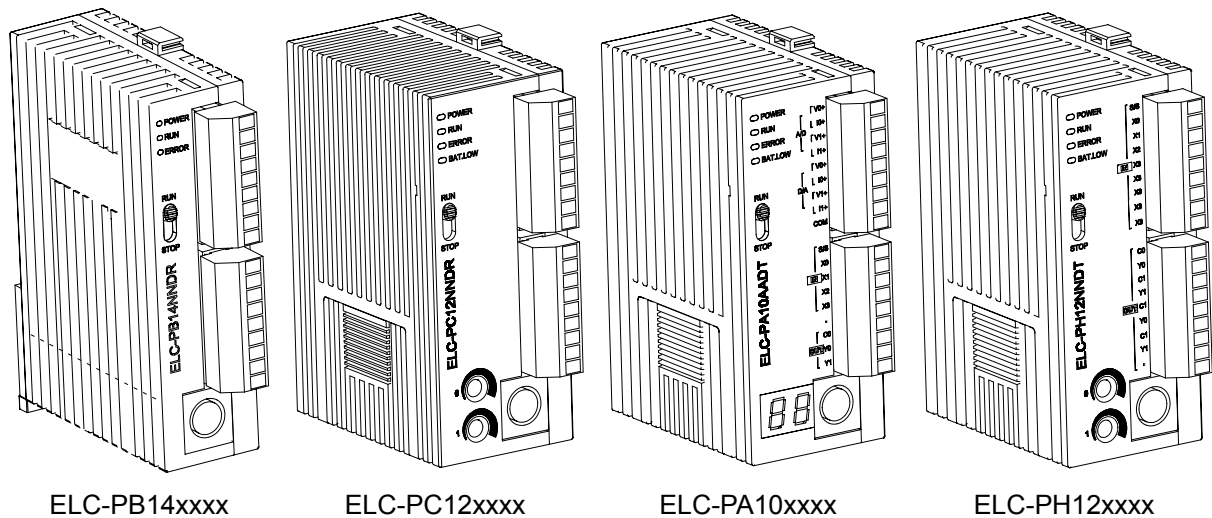
This Chapter Contains

1.1	ELC Processors	1-2
1.2	ELC Digital Modules	1-4
1.3	ELC Specialty Modules	1-5
1.4	ELC Power Supplies.....	1-6
1.5	ELCSoft Programming Software	1-7
1.6	ELC-HHP	1-8
1.7	ELC Graphic Panels	1-8
1.8	ELCSoftGP Programming Software.....	1-10

1 Product Overview

1.1 ELC Processors

The decision is easy when selecting an ELC processor. All ELC have the same internal features such as I/O capability; register size, latched memory, communications, etc. The only choice needed is choose whether the processor should have included analog in and out or all digital I/O.

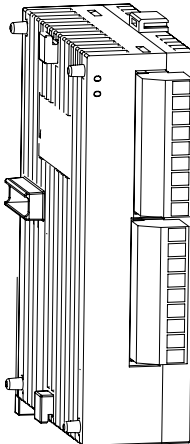
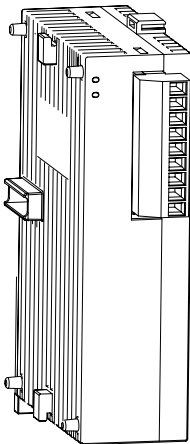


Items	ELC-PB14xxxx	ELC-PC12xxxx	ELC-PA10xxxx	ELC-PH12xxxx
Dimensions W x H x D (mm)	1.00 x 3.54 x 2.36 (25.2 x 90 x 60)	1.47 x 3.54 x 2.36 (37.4 x 90 x 60)		
Maximum I/O	256 (128 In / 128 Out) Any number of modules			
I/O points I/O Type	Total:14 points Digital Input: 8 Digital Output: 6	Total: 12 points Digital Input: 8 Digital Output: 4	Total: 10 points Digital input: 4 Digital output: 2 Analog input: 2 Analog output: 2	Total: 12 points Digital Input: 8 Digital output: 4 (Transistor)
Execution Speed	Basic commands – 2μ seconds minimum			
Program language	Commands + Ladder Logic + SFC			
Program Capacity	3792 Steps	7920 Steps		
Data Memory Capacity (bits)	1280 Bits	4096 Bits		
Data Memory Capacity (words)	744 Words	5000 Words		
Index Registers	2 Words	8 Words		
File Memory Capacity	-	1600 Words		

Items	ELC-PB14xxxx	ELC-PC12xxxx	ELC-PA10xxxx	ELC-PH12xxxx
Commands	32 Basic / 107 Advanced	32 Basic / 168 Advanced		
Floating Point	Yes	Yes		
SFC Commands	128 Steps	1024 Steps		
Timers	128 (1ms – 100ms)	256 (1ms – 100ms)		
Counters	128 (16/32 bit) (Up/Down)	235 (16/32 bit) (Up/Down)		
High Speed counters	13 1ph-1 input * 4 1ph-2 input * 1 2ph-2input * 1	15 1ph-1 input * 6 1ph-2 input * 1 2ph-2input * 1	18 1ph-1 input * 8 1ph-2 input * 2 2ph-2input * 1	
Pulse Output	2 channels 10KHz Max	2 channels, 50KHz Max		
Master Control Loop	8 Loops	8 Loops		
Subroutines	64 Subroutines	256 Subroutines		
Interrupts	6 (External / Timer / Comm.)	15 (External / Timer / HS CNTR / Comm.)		
Real-time Clock / Calendar	-	Built-in		
Specialty Expansions modules	8 (Analog In / Analog Out / PT/TC/RT) Modules do not count in total I/O			
Serial Ports	2 (RS-232 / RS-485)			
Special Features	-	2 Potentiometers	2 7-SEG displays	2 Potentiometers

1.2 ELC Digital Modules

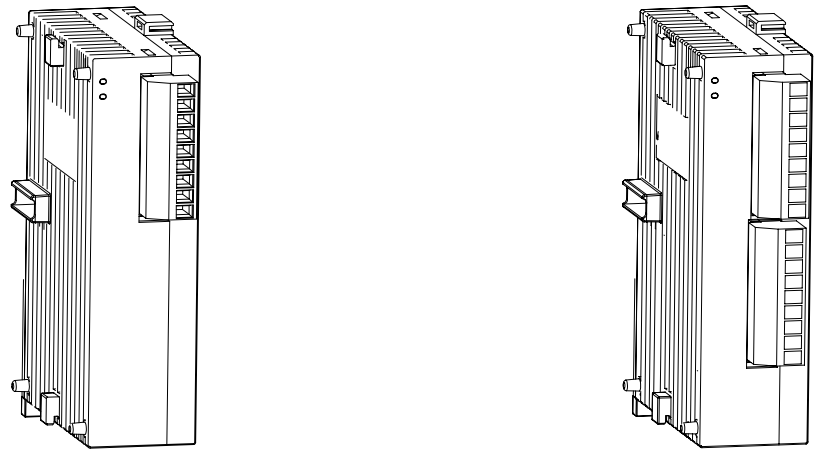
ELC expansion modules provide the correct amount of I/O for application solutions. Choose 4, 8 , or 16 I/O. Any number of expansion modules can be added to the ELC processor to create 256 I/O.



Model	Power	Input Unit		Output Unit	
		Point	Type	Point	Type
Dimensions W x H x D (mm)	0.99 x 3.54 x 2.36 (25.2 x 90 x 60)				
ELC-EX08NNAN	24VDC	8	AC	0	None
ELC-EX08NNDN		8	DC Sink or Source	0	None
ELC-EX08NNNR		0		8	Relay
ELC-EX08NNNT		0		8	Transistor
ELC-EX06NNNI		0		6	Relay
ELC-EX08NNDR		4		4	Relay
ELC-EX16NNDR		8		8	
ELC-EX08NNDT		4		4	Transistor
ELC-EX16NNDT		8		8	

1.3 ELC Specialty Modules

In addition the to expansion I/O, specialty modules like Analog In, Analog Out, Platinum Temperature, Thermocouple, etc.

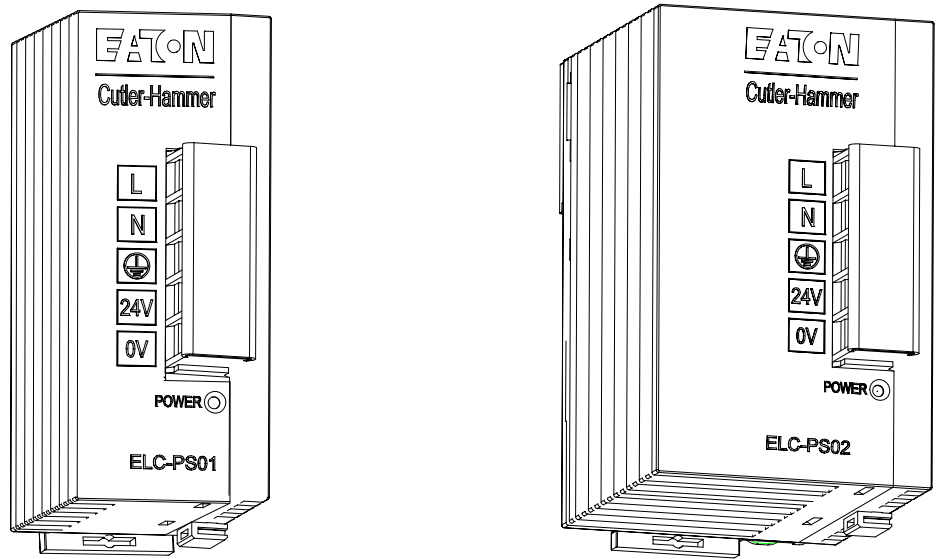


Model	Power	Input Unit		Output Unit	
		Point	Type	Point	Type
Dimensions W x H x D (mm)	0.99 x 3.54 x 2.36 (25.2 x 90 x 60)				
ELC-AN02NANN	24VDC	0	-20mA~20mA -10V ~ +10 V	2	0~20mA 0V ~ +10 V
ELC-AN04NANN		0		4	
ELC-AN06AANN		4		2	
ELC-AN04ANNN		4	Platinum Temp. Thermocouple Resistive	0	
ELC-PT04ANNN		4		0	
ELC-TC04ANNN		4		0	
ELC-RT08ANNN		8		0	

1.4 ELC Power Supplies

All ELC modules operate from 24 VDC. These power supplies provide a convenient way to provide robust DC voltage.

1



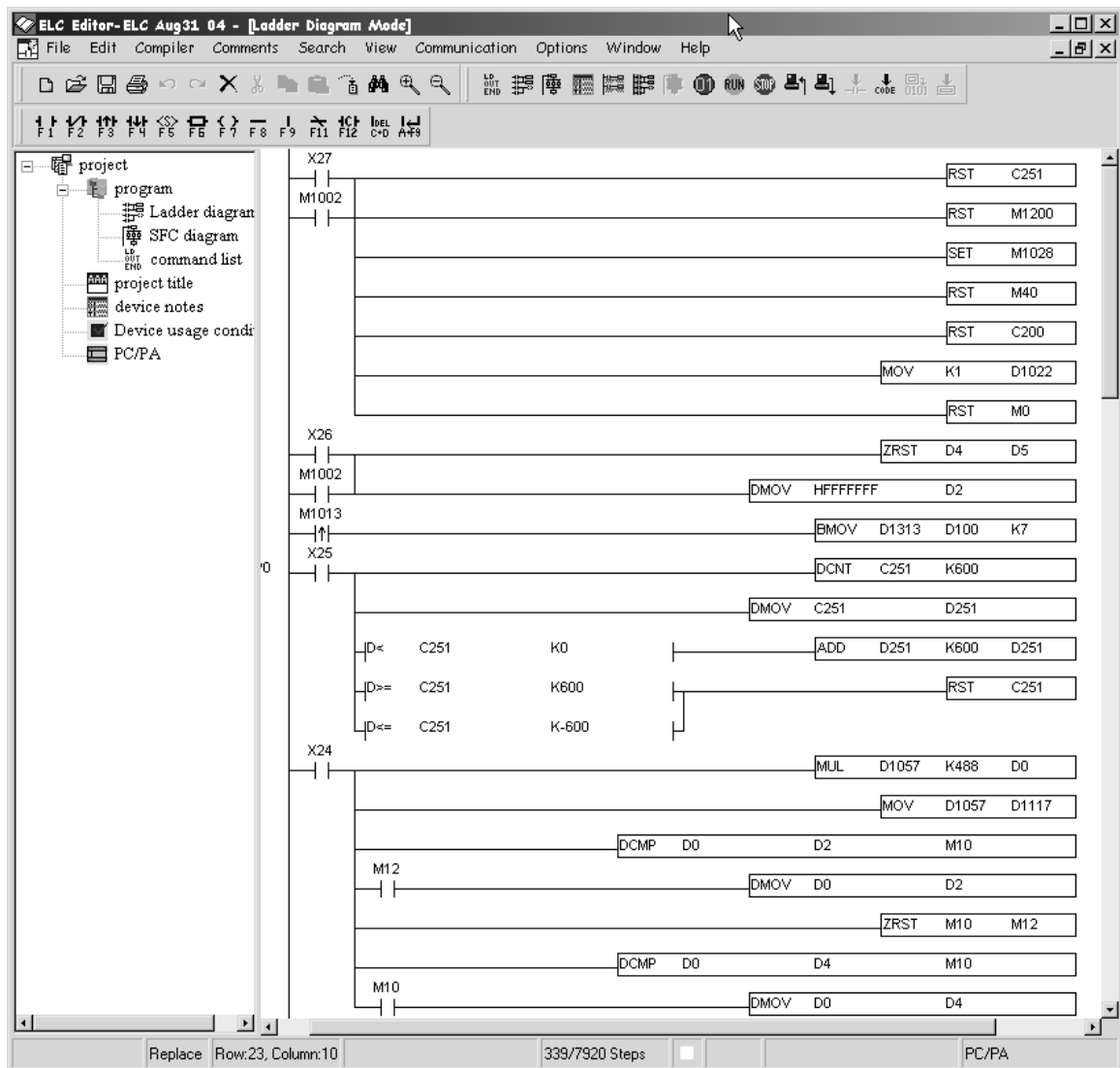
Model		
Item	ELC-PS01	ELC-PS02
Dimensions W x H x D (mm)	1.44 x 3.54 x 2.36 (36.5 x 90 x 60)	2.17 x 3.54 x 2.36 (55 x 90 x 60)
Input Power	100~240VAC 50/60Hz	
Output Volts	24VDC	
Output Current (A)	1 A	2 A

1.5 ELCSoft Programming Software

ELCSoft programming software configures all ELC controllers. With ELCSoft, applications can be created, edited, monitored, forced, etc. Move programs from one controller to a different one with ease.

Requirements:

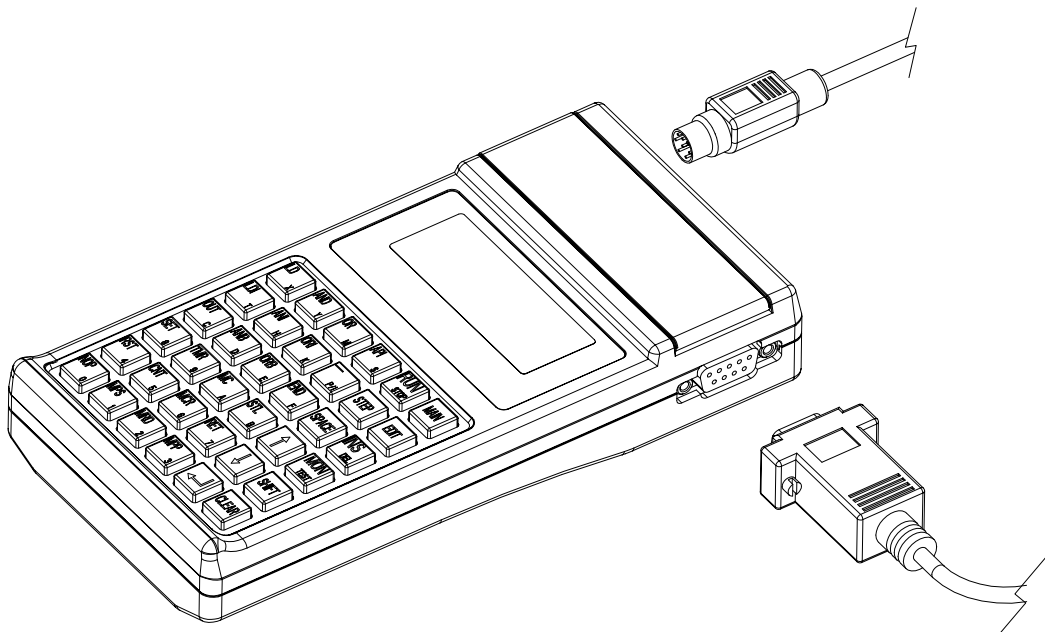
- Operating Systems – Windows 98, Windows ME, Windows 2000, Windows XP.
- Hard Drive – At least 100M bytes
- RAM – At least 256M bytes



1.6 ELC-HHP

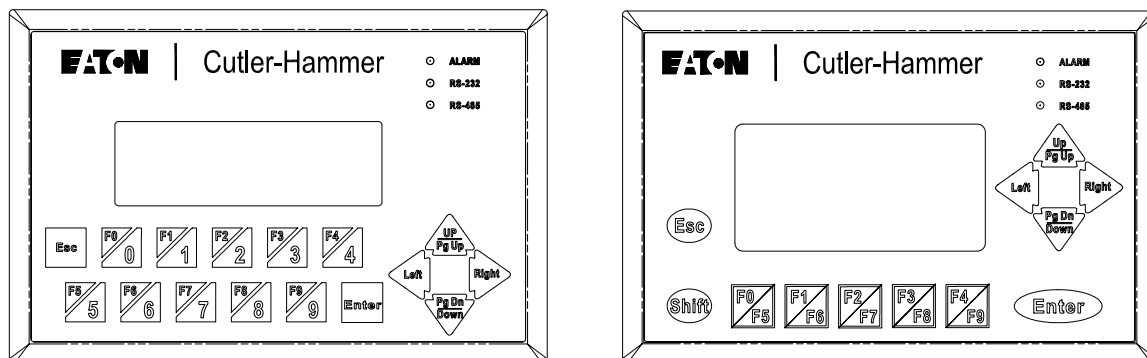
ELC-HHP is an easy to use hand held programming tool for ELC controllers when a PC is not available. With ELC-HHP, applications can be programmed directly with the attached keypad. Or uploaded from an ELC, saved, and transferred to a different ELC. Or downloaded from a PC and transferred to other ELCs. No need for outlets when using the ELC-HHP since it draws its power from either the ELC or the PC through the attached cable.

ELC-HHP with cables for ELC and PC connections



1.7 ELC Graphic Panels

ELC graphic panels allow modifying an application quick and easy. ELC graphic panels are simple to program and easily connect to ELC products. ELC graphic panels also connect to Eaton | Cutler-Hammer MVX drives.



Item		ELC-GP02	ELC-GP04
Display Screen	Screen	STN-LCD	
	Color	Monochromatic	
	Back-light	The back-light automatic turn off time is 1~99 minutes (0 = do not to turn off) (back-light life is 50 thousand hours at 25°C)	
	Resolution	160X32 dots	128X64 Points
	Display Range	72 mm (W) X 22 mm (H)	67mm (W) X 32mm (H); 3.00" (diagonal preferred)
	Contrast Adjustment	15-step contrast adjustment	10-step contrast adjustment
	Font	ASCII: characters Other: user define	
	Font Size (ASCII)	5 X 8, 8 X 8, 8 X 12, 8 X 16	
	ALARM Indication LED	1. Power on indication (Flash three times) 2. Flash for communication error or other alarm 3. Special Indication by user programming	
	RS-232 LED (Yellow)	It will be flashing when transmitting program and communicating by using RS-232.	
	RS-485/RS-422 Indication LED (Green)	It will be flashing when communicating by using RS-485/RS-422.	
Program Memory		256KB flash memory	
External Interface	Serial Communication Port RS-232 (COM1) 9 PIN D-SUB male	Data length: 7 or 8 bits Stop bits: 1 or 2 bits Parity: None/Odd/Even Baud Rate: 4800bps~115200bps RS-232: 9 PIN D-SUB male	
	Extension Communication port RS-422/RS-485 (COM2) 5-Pin Removal Terminal	Data length: 7 or 8 bits Stop bits: 1 or 2 bits Parity: None/Odd/Even Baud Rate: 4800bps~115200bps RS-485: 5-Pin removal terminal	Data length: 7 or 8 bits Stop bits: 1 or 2 bits Parity: None/Odd/Even Baud Rate: 4800bps~115200bps RS-422: 9 PIN D-SUB male RS-485: 5-Pin removal terminal
	Extension Slot	The slot for program copy card	
	Power	24V DC input	
	Battery Cover	-	DC 3V battery for HMI
	5-Pin Removable Terminal	There are DC 24V input and RS-485 input	

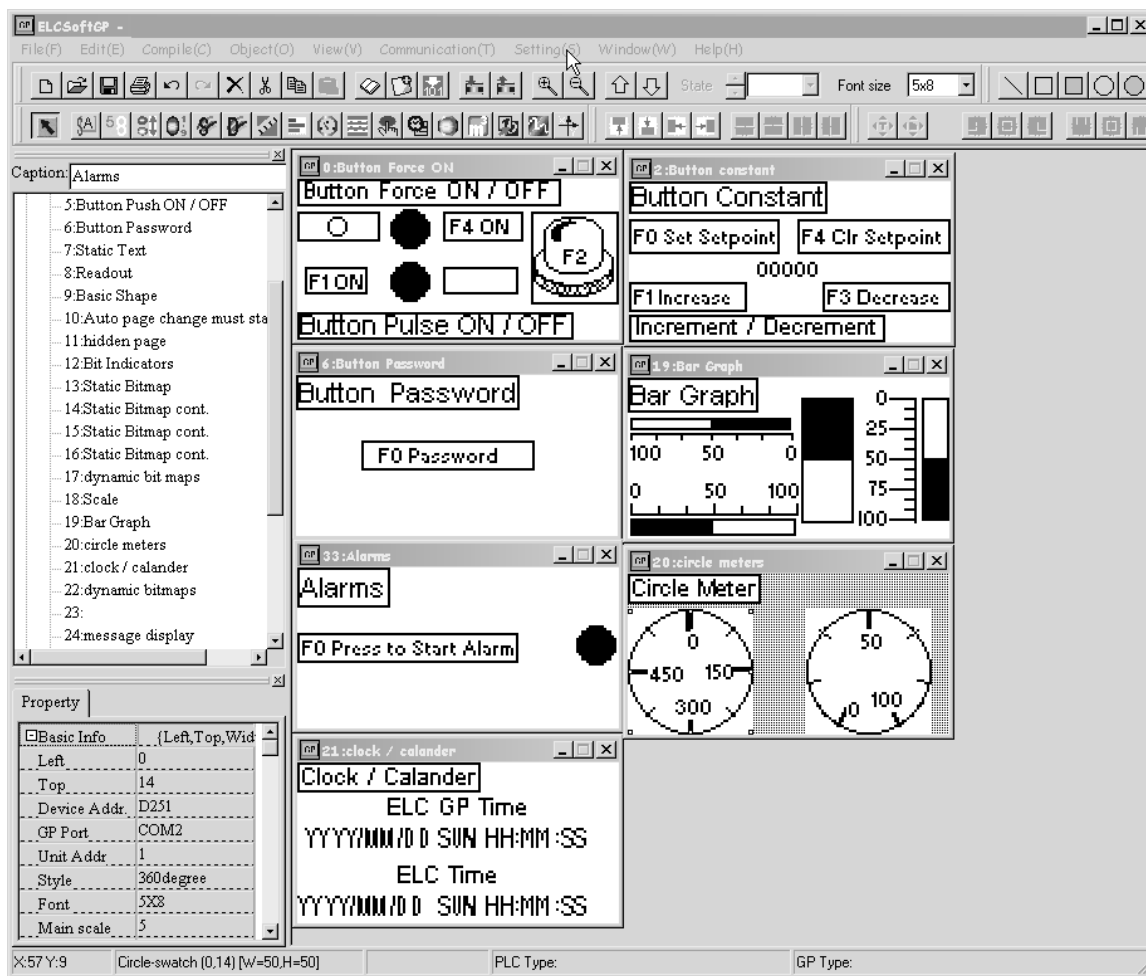
1.8 ELCSoftGP Programming Software

ELCSoftGP programming software configures all ELC graphic panels. With ELCSoftGP, applications can be created, edited, downloaded, uploaded, etc. Move programs from one controller to a different one with ease.

Requirements:

- Operating Systems – Windows 98, Windows ME, Windows 2000, Windows XP.
- Hard Drive – At least 100M bytes
- RAM – At least 256M bytes

ELCSoftGP screen shot displaying editing environment.



Getting Started

The Eaton Logic Controller (ELC) is easy to use and easy to set up. This chapter will show the quick method to connect an ELC to a PC, and upload, and download an ELC program using ELCSOFT.

This Chapter Contains

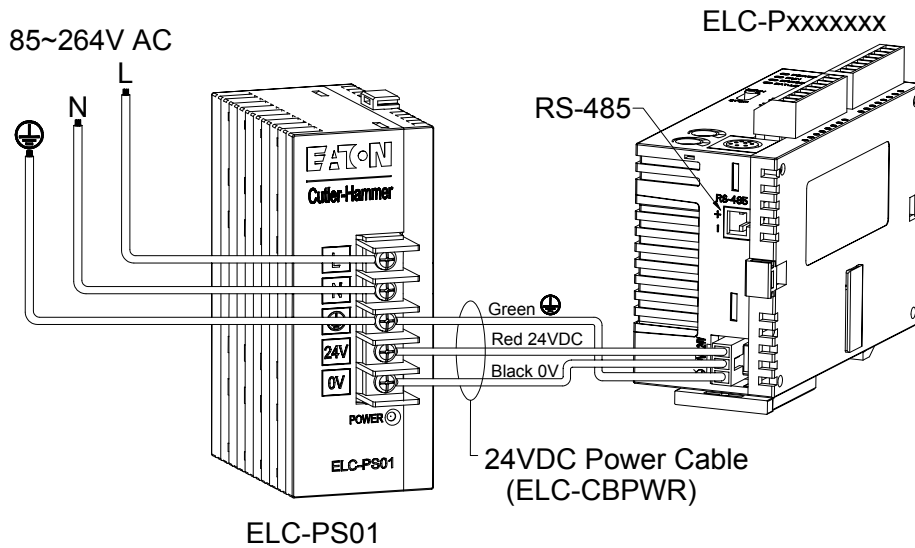
2.1	Connect ELC Power	2-2
2.2	Connect ELC to a PC.....	2-3
2.3	Connect ELC to the ELC-HHP	2-4
2.4	Current Consumed by the Control Unit	2-5
2.5	I/O Allocation.....	2-7
2.5.1	Digital Input/Output Extension Unit	2-7
2.5.2	Analog Input/Output Extension Unit	2-8
2.6	Install ELCSOFT	2-9
2.7	Enter this program	2-10
2.8	Download the Program	2-10
2.9	Monitor the Program	2-11

2 Getting Started

2.1 Connect ELC Power

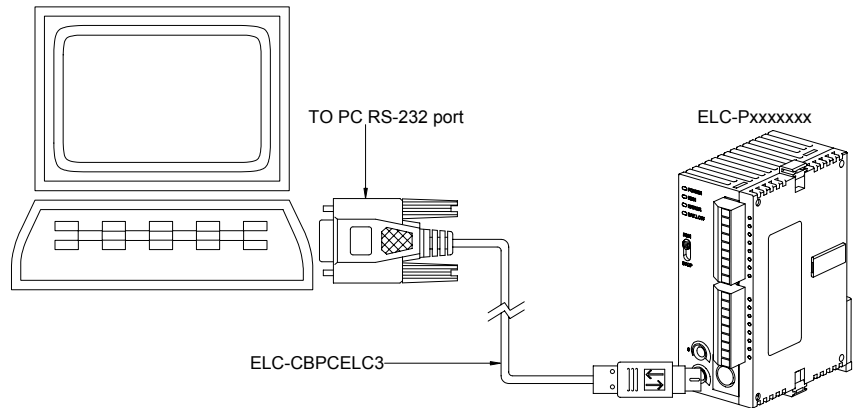
All ELC controllers operate from 24VDC. If the proper voltage is not available from the application, it can be provided from one of the ELC-PS0x power supplies. Simply wire the ELC as shown using the power cable ELC-CBPWR and plug it into the bottom of the ELC.

Once power is properly supplied to the ELC, it is ready for the next step.



2.2 Connect ELC to a PC

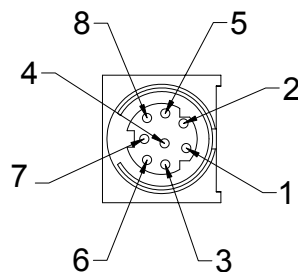
ELC is easy to connect to a PC one end of the ELC-CBPCELC3 to the 9-pin serial port of the PC RS-232 Communication port, and then connect the other end to the DIN port on the front of the ELC.



ELC communications - factory default settings (FOR ASCII):

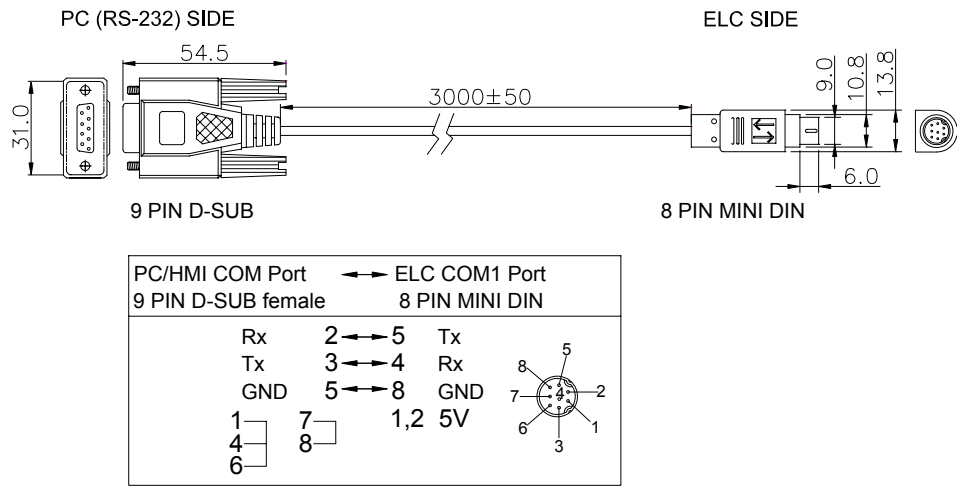
Protocol: 9600 Baud, 7 Data bits, Even Parity, 1 Stop Bit

Programming Port:

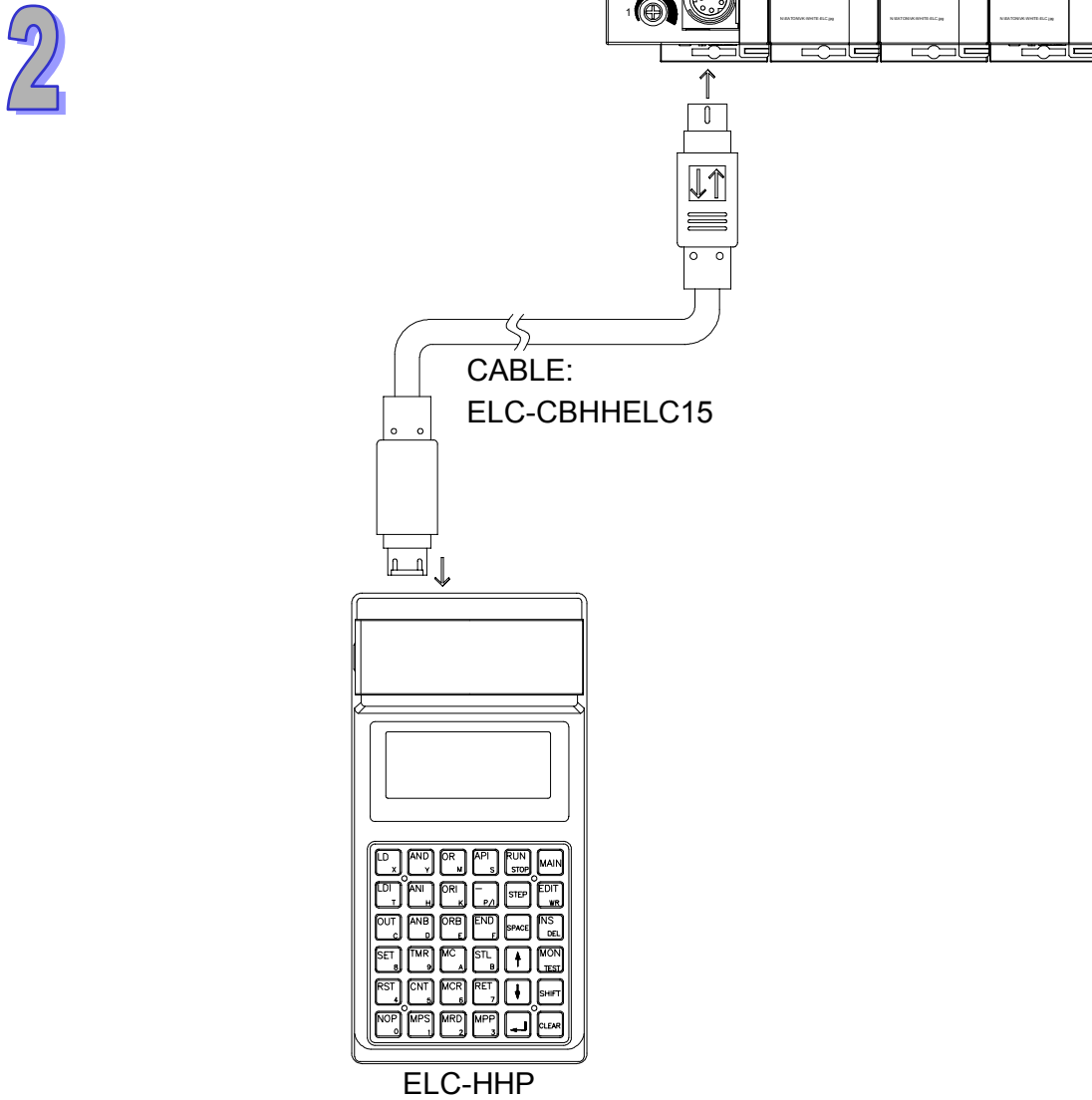


Pin no.	Abbreviation
1	+5V
2	+5V
3	GND
4	Rx
5	Tx
6	GND
7	NC
8	GND

ELC-CBPCELC3:



2-4



2.4 Current Consumed by the Control Unit

The current consumed at the power supply connector of the control unit is the sum of the current consumed by of the various units being used.

Type		Current consumption (at 24V DC)
Control unit	ELC-PA10AADR/T	210 mA or less
	ELC-PB14NNDR/T	150 mA or less
	ELC-PC12NNDR/T	250mA or less
	ELC-PC12NNAR	250mA or less
	ELC-PH12NNDT	170mA or less
Digital Input/Digital Output extension unit	ELC-EX08NNDN	50mA or less
	ELC-EX08NNAN	50mA or less
	ELC-EX08NNNR/T	70mA or less
	ELC-EX08NNDR/T	70mA or less
	ELC-EX16NNDR/T	90mA or less
	ELC-EX06NNNI	70mA or less
Handheld Programmer	ELC-HHP	70mA or less

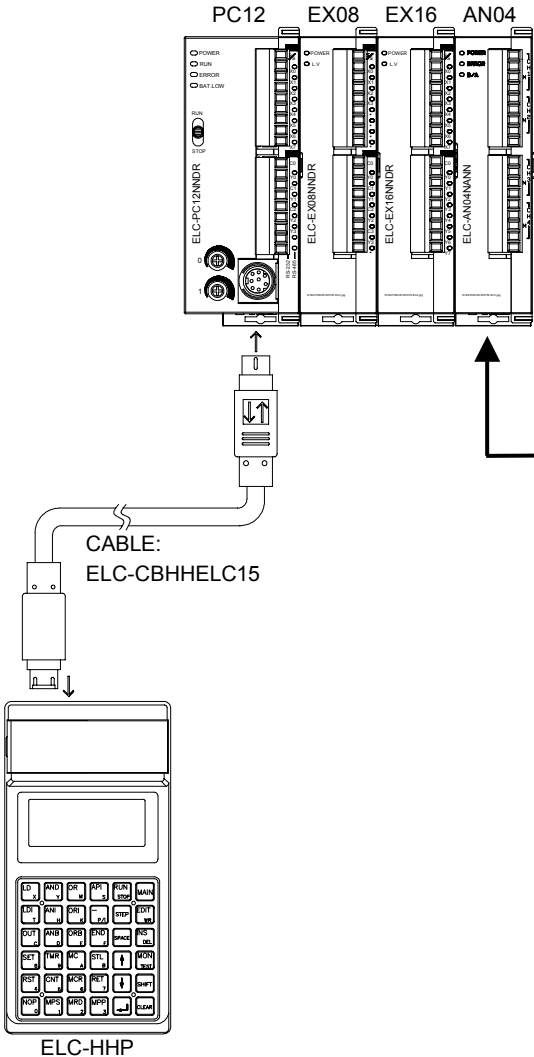
Current consumed when the unit requires an external power supply

With an analog I/O unit, it is necessary to provide a power supply to drive internal circuits.

Type		Current consumption (at 24V DC)
Analog Input/Output extension unit	ELC-AN02NANN	125mA or less
	ELC-AN04ANNN	90mA or less
	ELC-PT04ANNN	90mA or less
	ELC-TC04ANNN	90mA or less
	ELC-AN06AANN	90mA or less
	ELC-AN04NANN	170mA or less

Example: System Combination as below:

2



At power supply connector
of analog unit AN04

Type	Current consumption
ELC-AN04NANN	170mA

Power consumption calculation:

At power supply connector of control unit PC12

Type	Current consumption
ELC-PC12	250mA
ELC-EX08NNDR	70mA
ELC-EX16NNDR	90mA
ELC-AN04NANN	170mA
ELC-HHP	70mA
Total current consumption	650mA

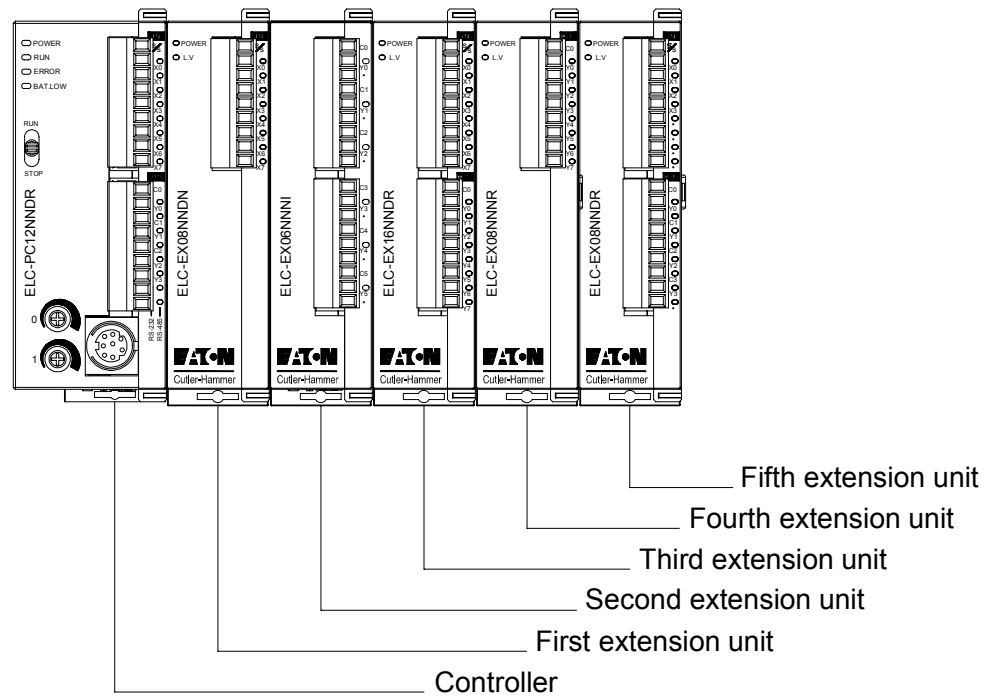
2.5 I/O Allocation

2.5.1 Digital Input/Output Extension Unit

I/O numbers automatically by the ELC control unit when an extension I/O unit is added. The I/O allocation of extension I/O unit is determined by the installation location. No matter how many points of MPU, the input of the first I/O extension unit will start from X20, output will start from Y20.

System Combination Example:

Extension unit	ELC Model	Input points	Output points	Input numbering	Output numbering
First extension	ELC-EX08NNDN	8	0	X20~X27	-
Second extension	ELC-EX06NNNI	0	6	-	Y20~Y25
Third extension	ELC-EX16NNDR	8	8	X30~X37	Y30~Y37
Fourth extension	ELC-EX08NNNR	0	8	-	Y40~Y47
Fifth extension	ELC-EX08NNDR	4	4	X40~X43	Y50~Y53



Note:

1. The second extension unit ELC-EX06NNNI will be used as 8 outputs, the higher 2 numbers of output points have no corresponding output points.
2. The fifth extension unit ELC-EX08NNDR will be used as 8 input points/8 output points, the higher part numbers of inputs points and output points have no corresponding input/output points. It is recommended to place them at the end of serial wiring, so that I/O points numbering will be continuous.

2.5.2 Analog Input/Output Extension Unit

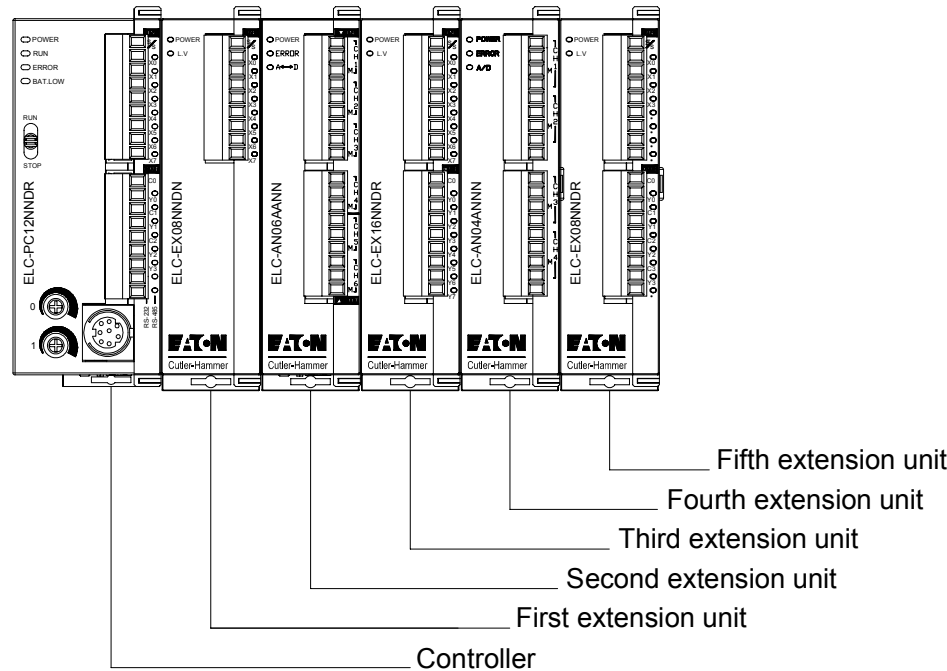
The I/O allocation of the analog I/O unit is determined by the installation location. A maximum of eight analog I/O extension units can be connected to one control unit.

There are no restrictions on the combination of different types(digital or analog) of extension units.

Extension unit	ELC Model	Input points	Output points	Input numbering	Output numbering
First extension	ELC-EX08NNDN	8	0	X20~X27	-
Second extension	ELC-AN06AANN	-	-	-	-
Third extension	ELC-EX16NNDR	8	8	X30~X37	Y20~Y27
Fourth extension	ELC-AN04ANNN	-	-	-	-
Fifth extension	ELC-EX08NNDR	4	4	X40~X43	Y30~Y33

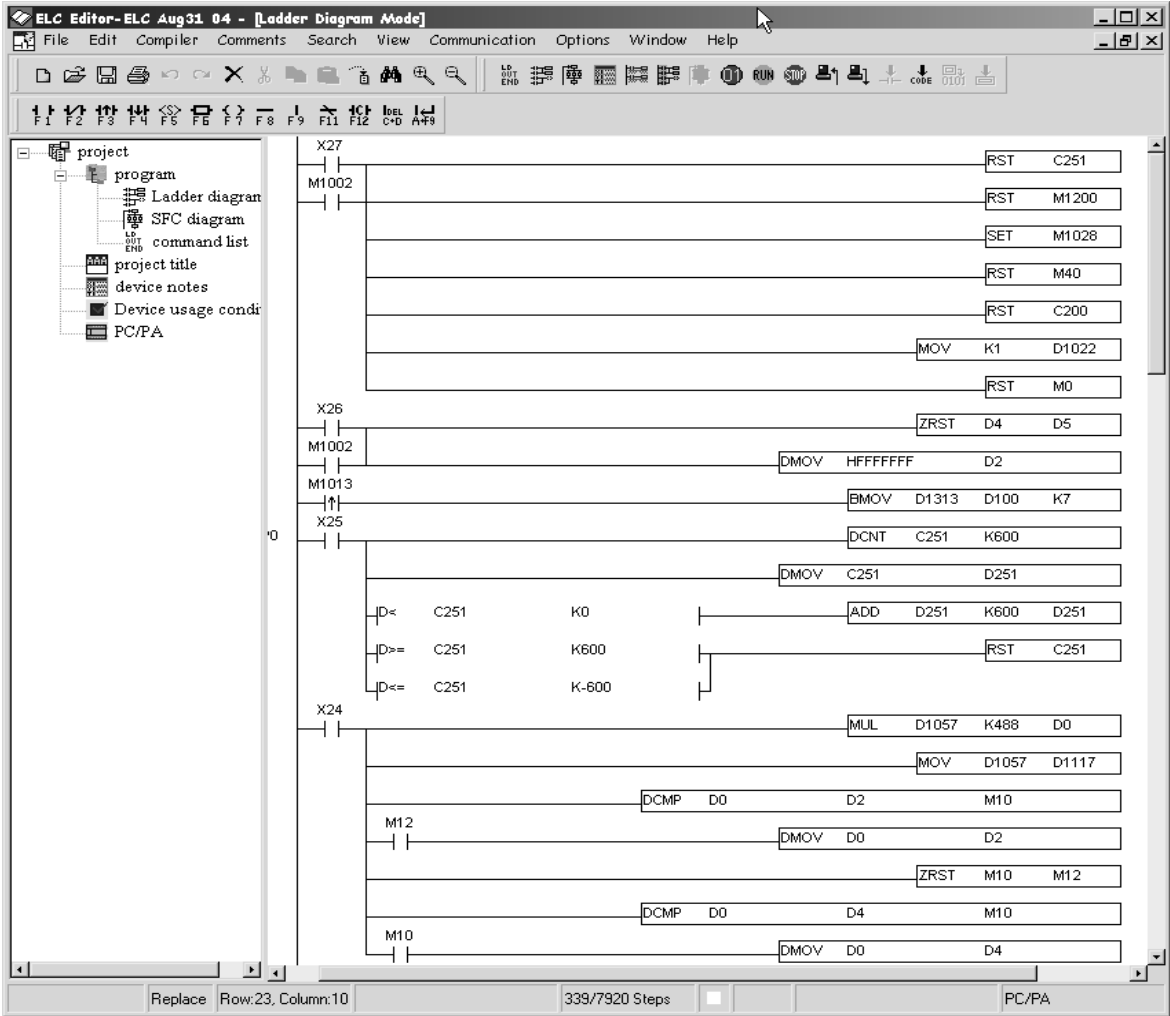
The second extension(ELC-AN06AANN) will be numbered as 0, and the fourth extension(ELC-AN04ANNN) will be numbered as 1. Please refer to chapter 6, API 78(FROM), 79(TO) for more detailed information about data accessing.

System Assembly Example:

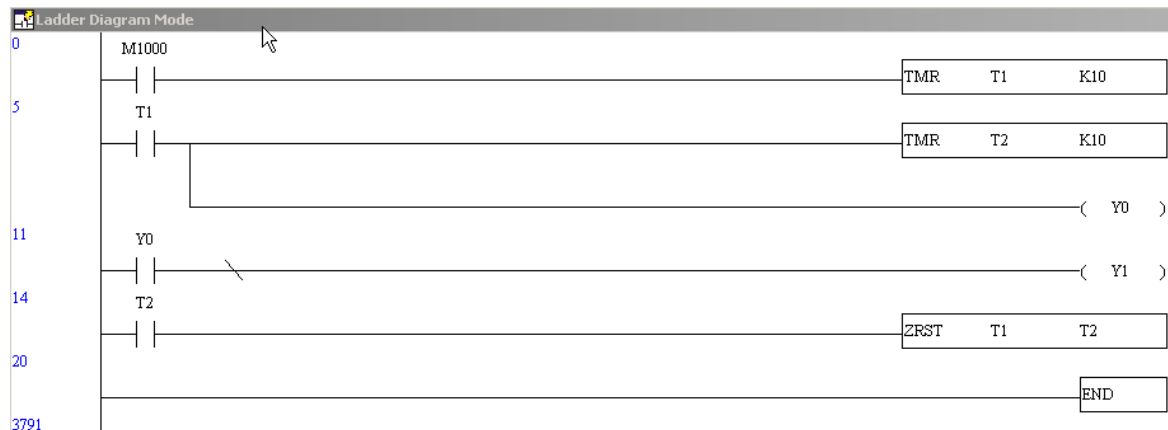


2.6 Install ELCSOft


Install ELCSOft on the PC and it is now ready to start communicating with the attached ELC.



2.7 Enter this program



2.8 Download the Program

After entering the program above, select compile . Then, select **Communication** from the menu bar, select **Transfer setup**, use the drop down button to select **Write to ELC**. ELCSoft will attempt to automatically match the settings of the ELC. Once communication is established, follow any instructions till the program is uploaded.

2.9 Monitor the Program

Make sure the ELC is in RUN mode, select **Communication** from the menu bar, and select **Ladder Start Monitoring**. You should see something like the example below.



MEMO

Installation

The ELC is easy to install. Keep in mind the safety, clearance, proper wiring, power budget, and proper grounding.

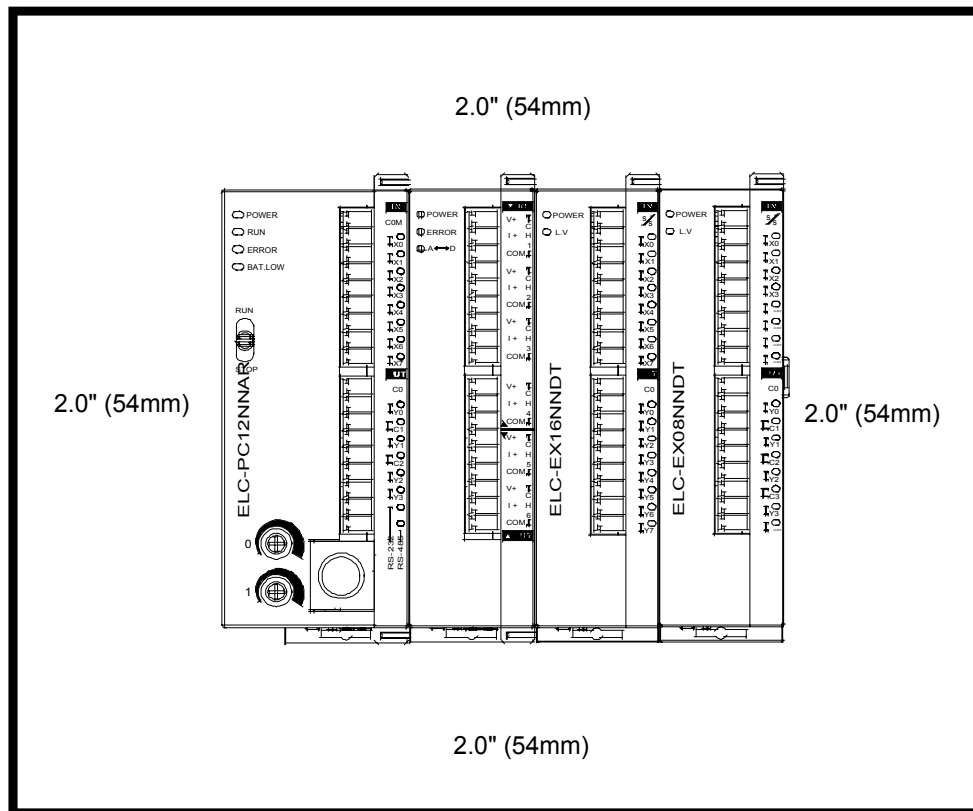
This Chapter Contains

3.1	ELC Mounting and Clearance.....	3-2
3.2	Installing Modules	3-3
3.3	DIN Rail Installation	3-4
3.4	Wiring.....	3-5
3.4.1	Safety Instructions	3-5
3.4.2	ELC Protection	3-6
3.4.3	Wiring the Power Supply/RS-485 to the control Unit	3-7
3.4.4	Grounding	3-7
3.4.5	Wiring the Terminal Block Socket.....	3-8
3.4.6	Input Point Wiring	3-9
3.4.7	Output Point Wiring	3-12
3.5	Dimensions	3-15
3.6	Terminal Layouts	3-18

3 Installation

3.1 ELC Mounting and Clearance

When installing the ELC, make sure that it is installed in an enclosure with sufficient space (as shown below) to its surroundings to allow proper heat dissipation.



Installation of the ELC products has been designed to be safe and easy. Whether the products associated with this manual are used as a system or individually, they must be installed in a suitable enclosure. The enclosure should be selected and installed in accordance to the local and national standards.



The ELC should be mounted on a vertical position. To prevent a rise in temperature, units should always be mounted on the back wall of an enclosure. Never mount ELC to the floor or ceiling of the enclosure.



Do not install units in areas with excessive or conductive dust, corrosive or flammable gas, moisture or rain, excessive heat, regular impact shocks or excessive vibration.

Do not allow debris to fall inside the unit during installation, e.g. cut wires, shavings etc.

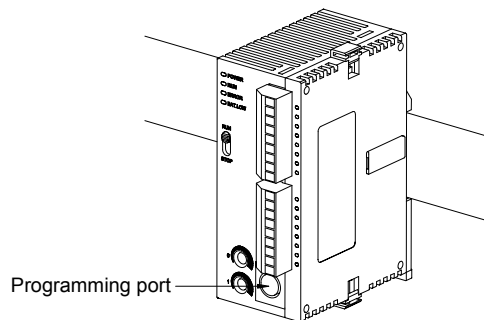
Always ensure that units are kept as far as possible from high-voltage cables and equipment.

3.2 Installing Modules

Before you install or remove any modules ensure power to the ELC and any surrounding equipment has been removed.

Installing or removing ELC modules or surrounding equipment with power applied could cause electric shock or equipment failure.

Always install the unit orientated with the programming port facing outward on the bottom in order to prevent the generation of heat.



System Assembly

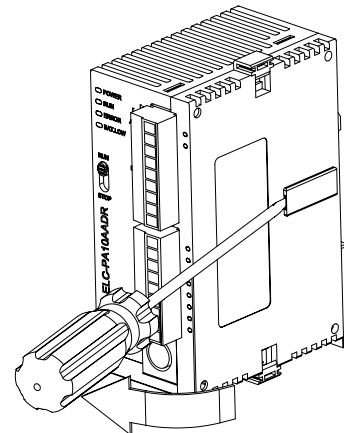
The ELC should only be installed on a proper DIN rail. Combine modules together to complete the required system I/O. Remove the expansion port cover and lift up on the top and bottom locking connectors of the module on the left.

Mate to the next desired module by lining up the alignment pins and expansion port. When the modules are fully seated, snap down the locking connectors to hold the modules together. Continue this process until the all required modules have been combined.

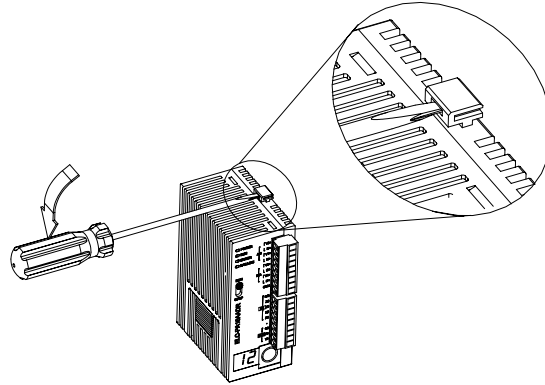
There is no order to the placement of modules. Set up the module order the preference that suite the application best.

Procedure:

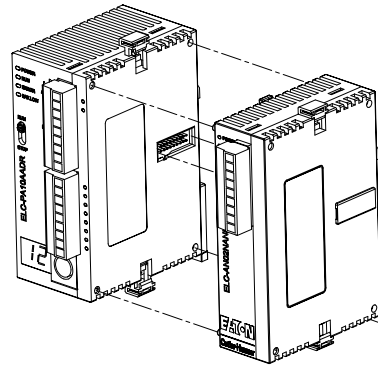
1. Open the extension cover on the side of the unit with a screw driver so that the external connector is exposed.



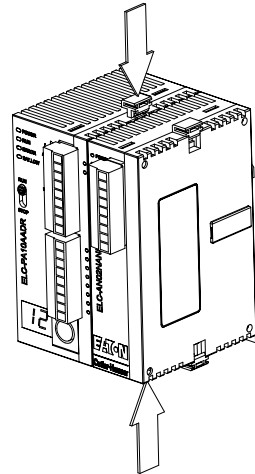
2. Raise the extension hooks on the top and bottom sides of the unit with a screwdriver.



3. Align the pins and holes in the four corners of the control unit and extension unit, and insert the pins into the holes so that there is no gap between the units.



4. Press down the expansion hooks raised in step 2 to secure the unit.



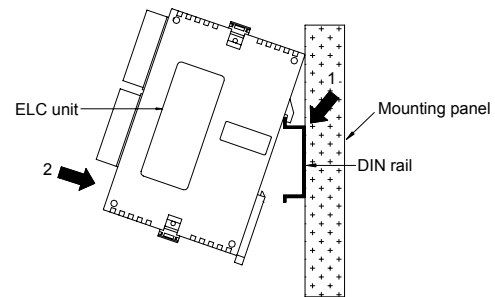
3.3 DIN Rail Installation

The ELC should be secured in a cabinet using a standard DIN rail 35mm high with a depth of 7.5mm. To reduce the chance of the wires being pulled loose use end brackets to stop any side-to-side motion of the ELC.

Secure the ELC to the DIN rail with the retaining clip located bottom of the ELC. Mount the ELC on the DIN rail and push up the clip to lock the ELC in place.

Procedure:

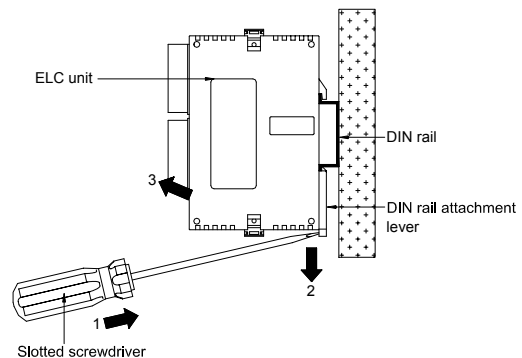
1. Fit the upper book of the ELC unit onto the DIN rail.
2. Without moving the upper hook, press the lower hook to fit the ELC unit into position



To remove, pull down the retaining clip and pull the ELC away from the DIN rail.

Procedure:

1. Insert a slotted screwdriver into the DIN rail attachment lever
2. Pull the attachment lever downwards
3. Lift up the ELC unit and remove it from the rail




3

3.4 Wiring

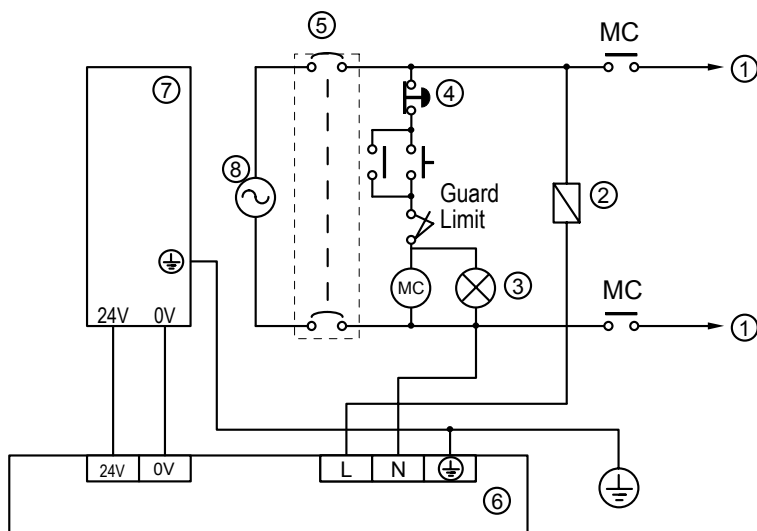
3.4.1 Safety Instructions

1. Use 22-16AWG (1.5mm) wiring (either single or multiple core) for wiring terminals.
2. ELC terminal screws should be tightened to 1.7 in-lbs (1.95 kg-cm).
3. Ensure the power at the ELC terminals is 24VDC (20.4VDC~28.8VDC). Voltage lower than 20.4VDC will cause the ELC to stop operating, all outputs will turn off and the ELC ERROR LED will flash continuously.
4. ELC has a momentary power ride-through of 10ms. A power drop less than 10ms will not have any effect. A power drop longer than 10ms, ELC will stop operating and turn all the outputs off. Once power is restored, the ELC will resume operation.
5. Use surge suppression as necessary.
6. Connect the AC input (100Vac to 240Vac) to terminals L and N. Any AC voltage connected to the +24V terminal or input point will permanently damage the ELC.
7. Avoid creating sharp bends in the wires.
8. Avoid running DC wiring in close proximity to AC wiring.
9. To minimize voltage drops on long wire runs, consider using multiple wires for the return line.

10. Avoid running input wiring close to output wiring where possible.
11. Avoid running wires near high power lines.
12. Use wire trays for routing where possible.
13. Use the shortest possible wire length.
14. Always use a continuous length of wire. Do not splice wires to attain a needed length.
15. DO NOT use the  terminal in ELC.
16. Input and output signal wires should not run through the same multi-wire cable, conduit, or near high voltage cables.
17. All low voltage wires should cross high voltage cables at 90° when possible.

3.4.2 ELC Protection

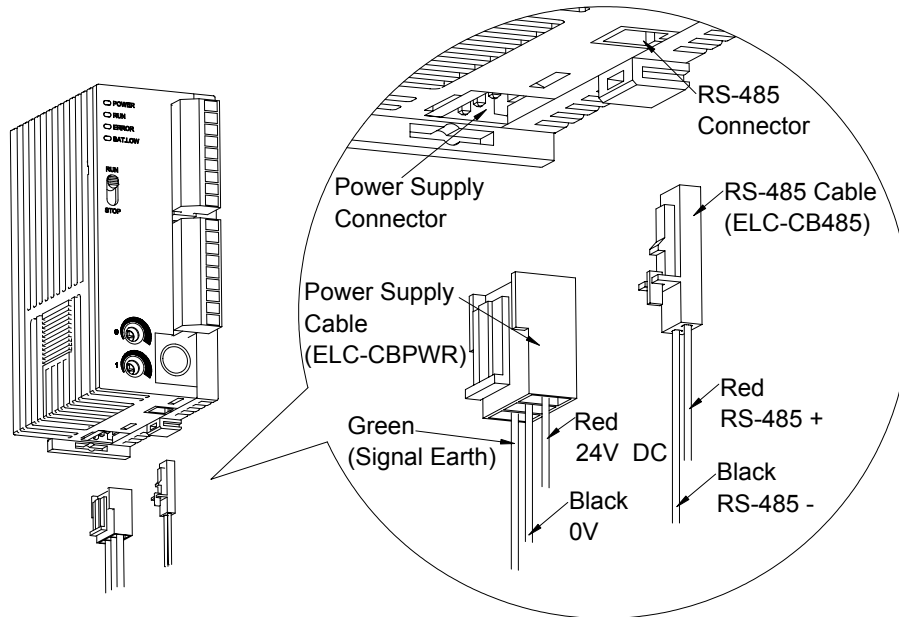
To protecting your investment, it is common to implement a protective circuit similar to the one shown here.



1	Switched AC Power for AC loads
2	Power Circuit Protection Fuse (3A)
3	Power On indicator
4	Emergency stop
5	Normally open contactor
6	DC Power supply
7	ELC
8	Power supply: AC: 100~240VAC, 50/60Hz

3.4.3 Wiring the Power Supply/RS-485 to the control Unit

Use the power supply cable(ELC-CB) and RS-485 cable(ELC-CB) that come with the unit to connect the power and communication devices.

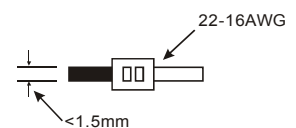


Note:

1. To minimize adverse effects from noise, twist the red and black wires of the power supply cable, also twist the RS-485 cable.
2. The regulator on the ELC unit is a non-insulated type.

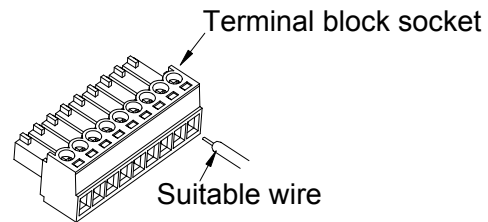
3.4.4 Grounding

1. Use wires 20-16AWG (1.6mm) or above for the grounding of the ELC. Ground wires must not be a smaller diameter than input wires.
2. Ensure good grounding by wiring all commons to the same grounding point.
3. Ground wires should be as short as possible.
4. Use Copper Conductor Only, 60 °C



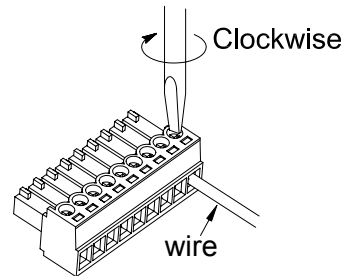
3.4.5 Wiring the Terminal Block Socket

A screw-down connection type terminal block socket for the terminal of the ELC control unit and special I/O unit is used. The terminal block socket and suitable wires are given below.



Procedure:

1. Remove a portion of the wire's insulation
2. Insert the wire into the terminal block socket until it contacts the back of the block socket, and then tighten the screw clockwise to fix the wire in place.



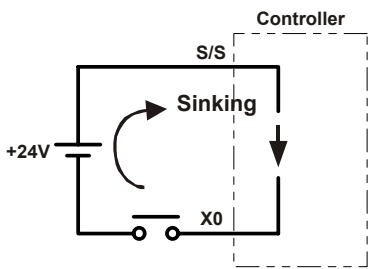
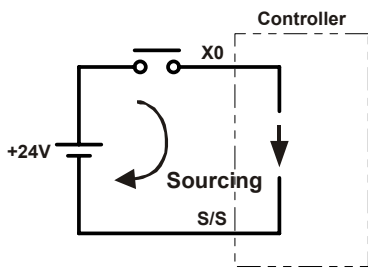
3

3.4.6 Input Point Wiring

24V DC input Specifications

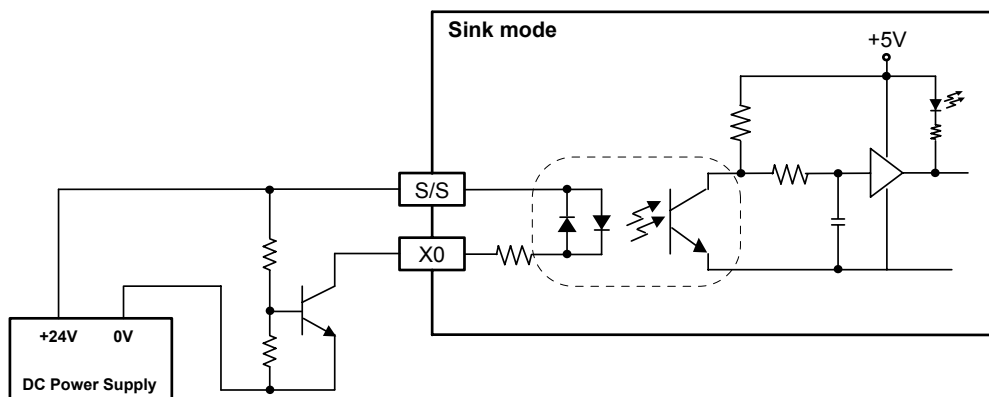
	PA/PB/PC/PH Control Unit	Digital Input Extension Unit
	X0~X7	X20 and above
Input Type	DC (SINK or SOURCE)	
Input Current	24VDC 5mA	
Motion Level	OFF→ON, above 16VDC ON→OFF, below 14.4VDC	
Responding Time	X0~X7: 10ms (factory default), 0~1,000ms adjustable. Refer to the usage of special registers D1020. X10, X11: 0 times (factory default), 0~1,000 times adjustable. Refer to the usage of special registers D1021.	10 ms

ELC input points can be configured for either sink or source operation. Each module that contains inputs will have a S/S (Sink/Source) input pin. Connect the S/S to either 24VDC or 0VDC to select the desired operation.

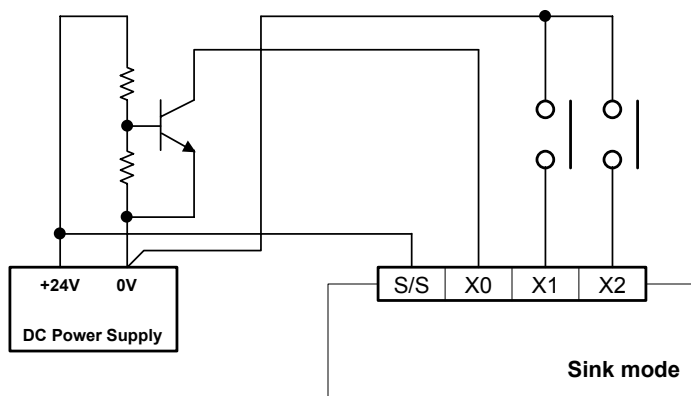
Sink = Current flows into S/S	Source = Current flows out of S/S
	

Typical Wiring

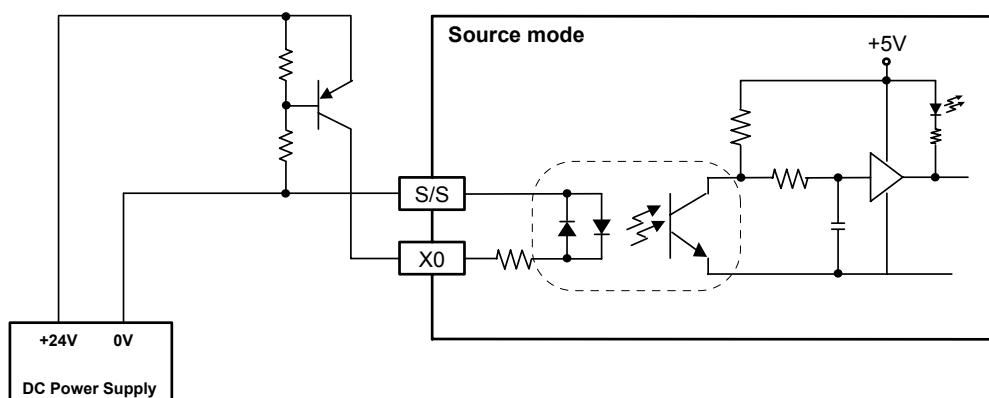
a) Sink – Equivalent Circuit



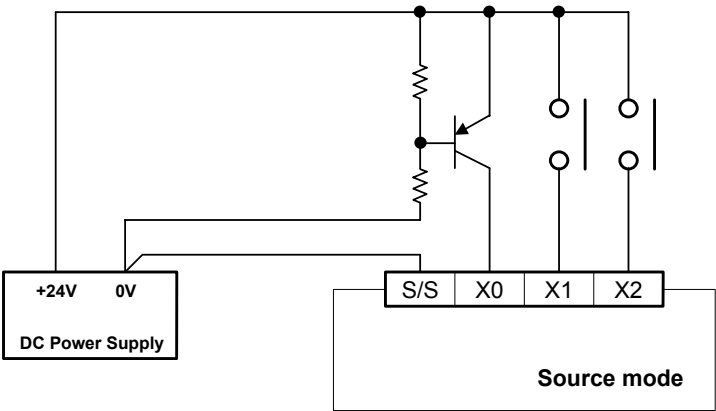
b) Sink – User wiring



c) Source – Equivalent Circuit



d) Source –User wiring



AC110V input Control Unit and Extension Units

Input Circuit Connection	110V AC Input Specifications	
<p>85~132VAC 50/60Hz</p> <p>COM X0 X1 X2</p> <p>ELC-12PCNNAR</p>	Input voltage	100~120VAC(-15%~+10%)
	Input impedance	21Kohm/50Hz 18Kohm/60Hz
	Input current	4.7mA 100VAC/50Hz 6.2mA 110VAC/60Hz
	OFF→ON/ON→OFF	80V 3.8mA/30V 1.7mA
	Response time	25ms
	Circuit isolation/Operation indication	Photocoupler/LED On

3

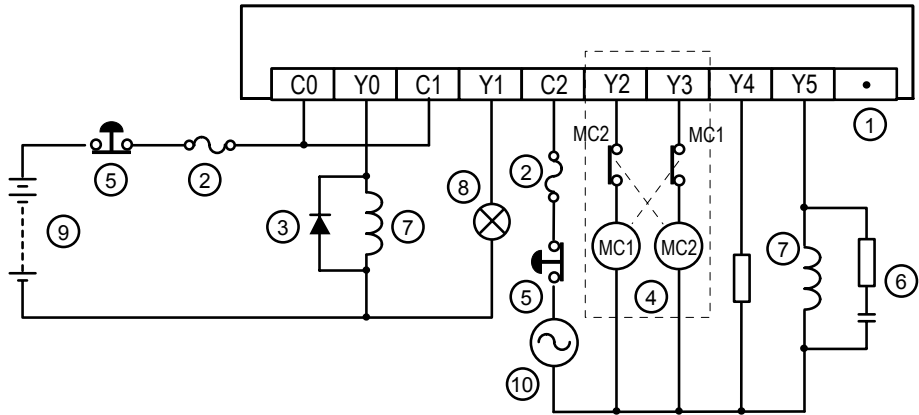
3.4.7 Output Point Wiring

Output Specifications

Output Type Item	Relay-R	Relay-R (Note 1)	Transistor-T
Current Spec.	1.5A/1 point (5A/COM)	6A/1 point	55℃ 0.1A/1 point, 40℃ 0.3A/1 point
Voltage Spec.	250VAC, below 30VDC	250VAC, below 30VDC	30VDC
Max. Loading	75VA (Inductive)	240VA (Inductive)	9W
	90 W (Resistive)	270 W (Resistive)	
Response Time	About 10 ms	About 10 ms	Off→On 15us On→Off 25us

Note 1: Only for model ELC-EX06NNNI

Relay Output Example-Typical Relay Output Wiring Diagram

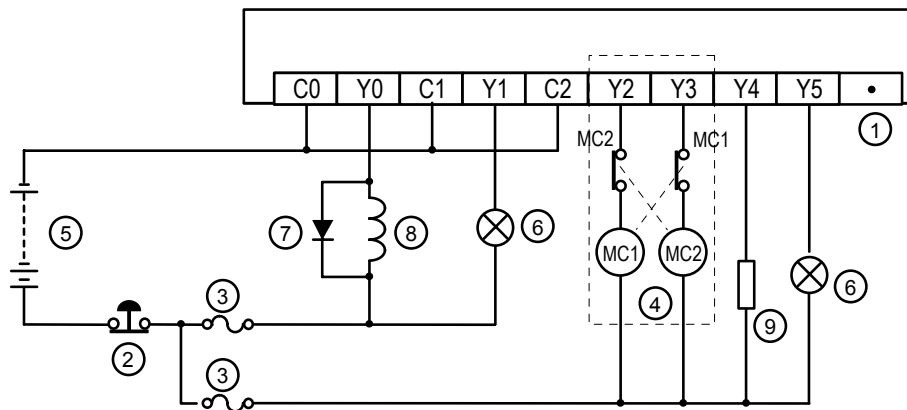


1	Do not use this terminal
2	Fuse
3	Reverse-current protection diode, Note 1
4	External Mechanical Interlock, Note 2
5	Emergency stop
6	Surge absorber(0.1uf capacitor+100~120ohm resistor, Note 3)
7	Inductive load
8	Incandescent lamp
9	DC power Supply
10	AC power Supply

Note:

1. This ELC does not have any internal protection circuitry on the relay outputs. For switching direct current on inductive loads, a reverse-current protection diode should be installed in parallel with the load. The relay contact life decreases significantly if this is not done.
The reverse-current protection diode needs to satisfy the following specifications.
 - The diode is rated for maximum reverse voltage of 5~10 times the load voltage.
 - The forward current is more that the load current
2. Ensure all loads are applied to the same side of each ELC output, see above figure. Loads which should NEVER simultaneously operate(e.g. direction control of a motor), because of a critical safety situation, should not rely on the ELC's sequencing alone. Mechanical interlocks MUST be fitted to all critical safety circuits.
3. This ELC does not have any internal protection circuitry on the relay output. For switching AC on inductive loads, a surge absorber (0.1uF + "100ohm to 120ohm") should be installed in parallel with the load. The relay contact life decreases significantly if this is not done. Besides protecting the internal circuitry of the ELC, a surge absorber decreases the noise emissions to the load.

Transistor Output Example-Typical Transistor Output Wiring Diagram



1	Do not use this terminal
2	Emergency Stop
3	Fuse
4	External Mechanical Interlock
5	DC Power Supply
6	Incandescent Lamp
7	Reverse-current protection diode, Note 1
8	Inductive load, Note 2
9	Resistive load

Note:

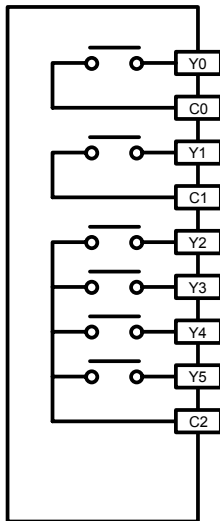
1. Ensure all loads are applied to the same side of each ELC output, see above figure. Loads which should NEVER simultaneously operate (e.g. direction control of a motor), because of a critical safety situation, should not rely on the ELC's sequencing alone. Mechanical interlocks MUST be fitted to all critical safety circuits.
2. Transistor outputs use internal zener diode(39V) as protection circuitry. When driving the inductive load with transistor output, a reverse-current protection diode can be installed in parallel with the load if necessary.

The reverse-current protection diode needs to satisfy the following specifications.

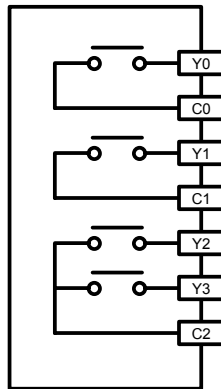
- The diode is rated for maximum reverse voltage of 5 to 10 times the load voltage.
- The forward current is more than the load current.

3. Be sure to monitor the use of common terminals when wiring outputs. The ratio of common terminal to output terminals is not the same on all modules with outputs. Some outputs share common terminals. The output terminal Y0 uses one common terminal C0, Y1 uses C1, and Y2~Y3(Y5) share C2, as shown below.

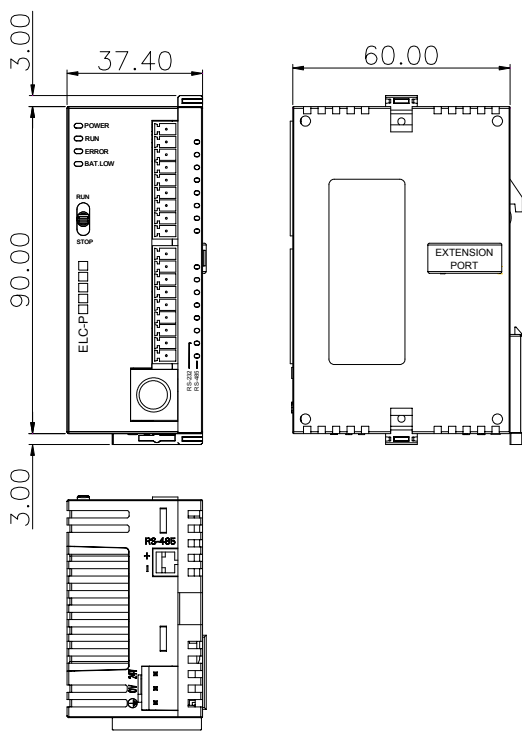
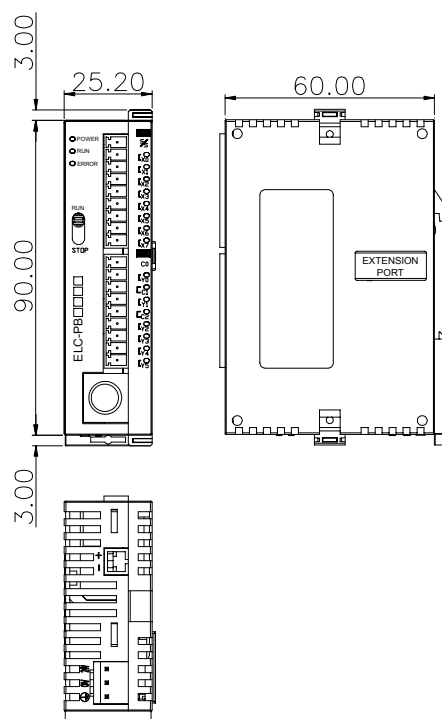
PB Series:



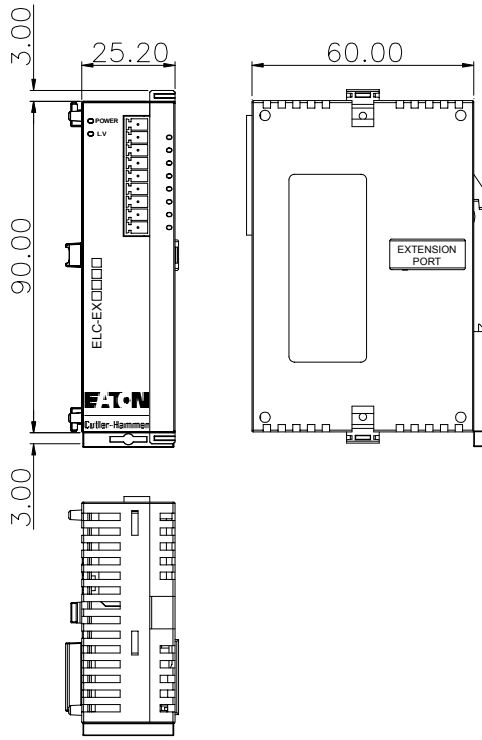
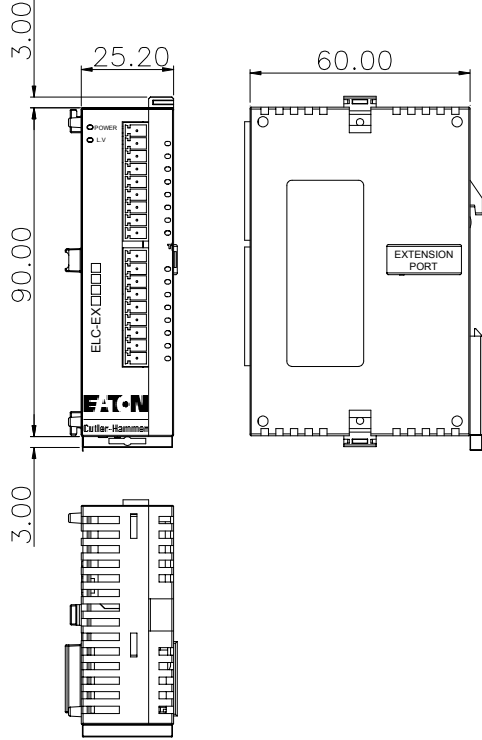
PC Series:



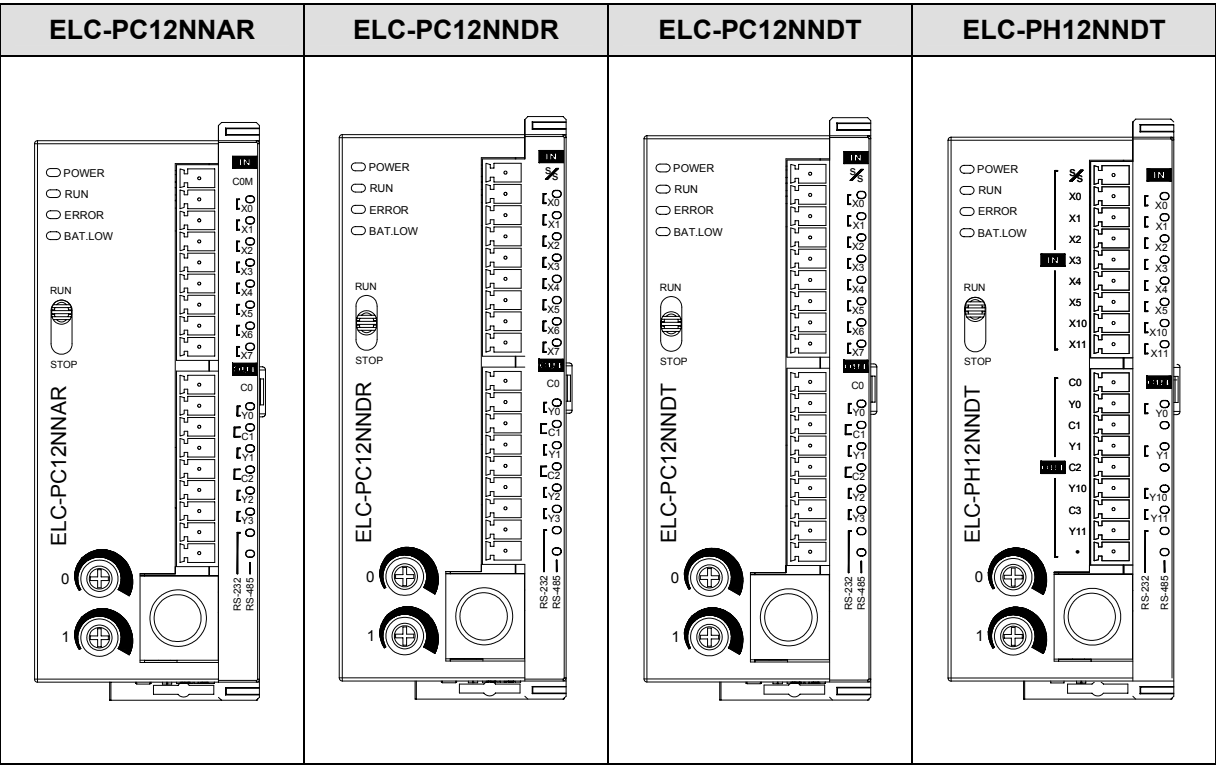
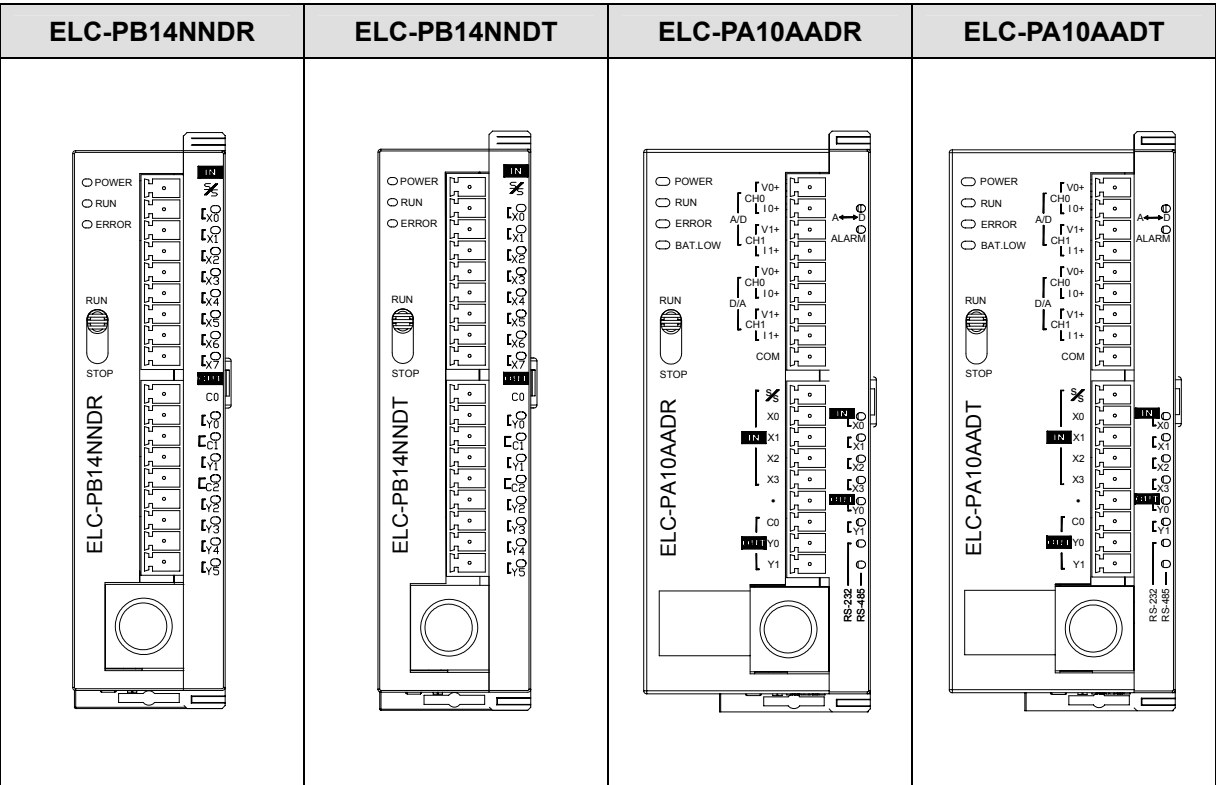
3.5 Dimensions

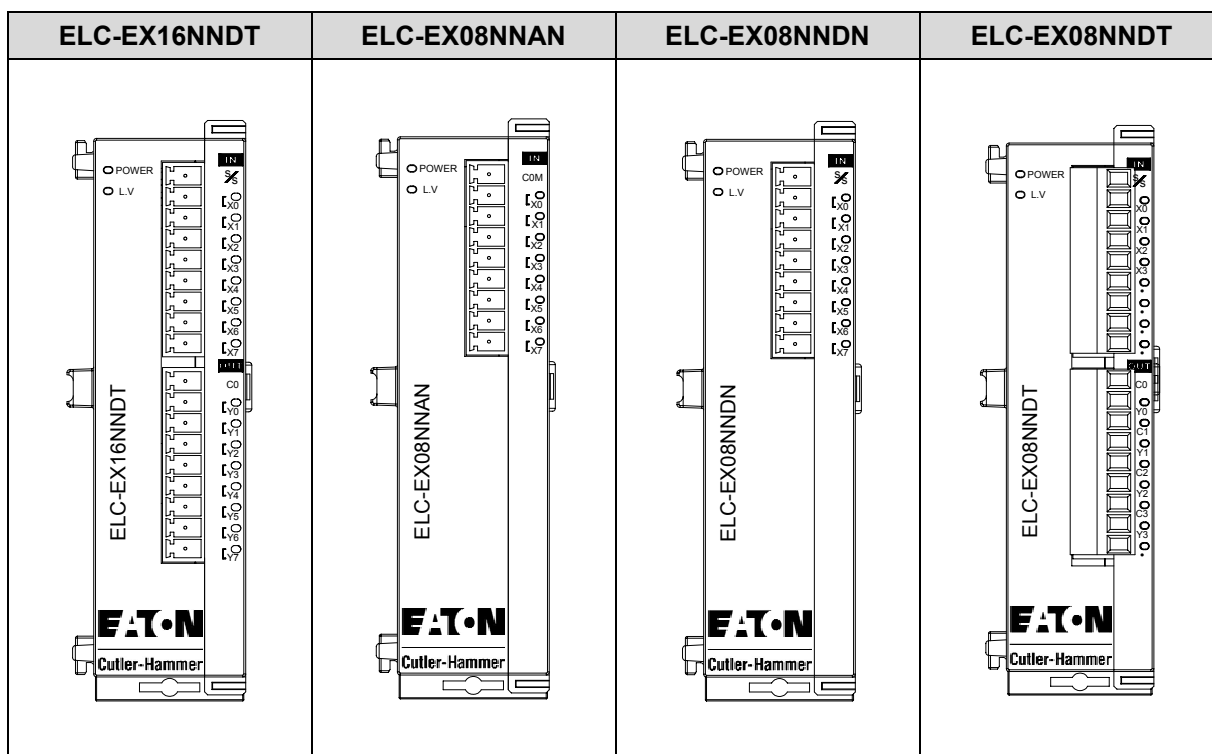
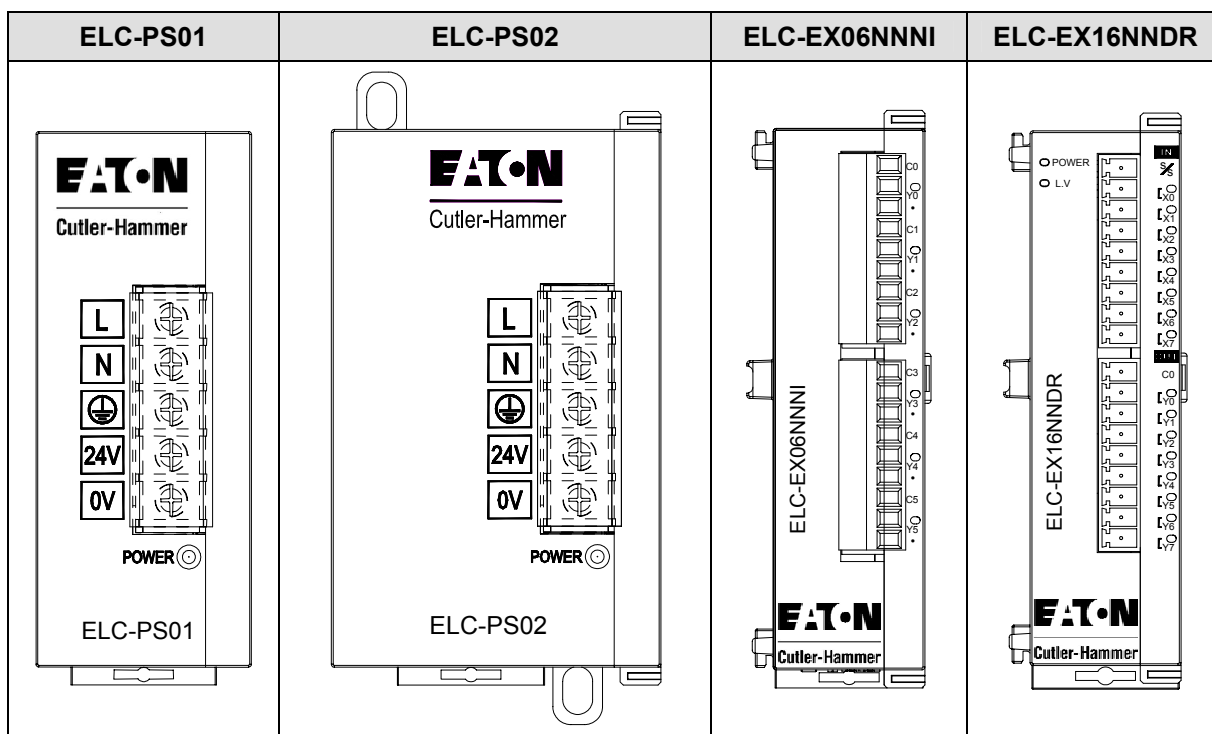
Model		Dimensions
ELC Controller	ELC-PA10AADR ELC-PA10AADT ELC-PC12NNAR ELC-PC12NNDR ELC-PC12NNDT	
	ELC-PB14NNDR ELC-PB14NNDT	

Model		Dimensions
DC Power Supply	ELC-PS01	<p>Technical drawing of the ELC-PS01 DC Power Supply. The front view shows a vertical rectangular unit with a height of 90 and a width of 36.5. It features a terminal block with terminals labeled L, N, 24V, and 0V, and a POWER symbol. The side view shows a height of 13.3 and a depth of 60. A small dimension of 3 is indicated at the bottom of the front view.</p>
	ELC-PS02	<p>Technical drawing of the ELC-PS02 DC Power Supply. The front view shows a vertical rectangular unit with a height of 100 and a width of 55. It features a terminal block with terminals labeled L, N, 24V, and 0V, and a POWER symbol. The side view shows a height of 13.3 and a depth of 60. A small dimension of 3 is indicated at the bottom of the front view. A dimension of 32.7 is also shown for the main body width.</p>

Model		Dimensions
Expansion Modules	All I/O Modules Including Specialty Modules	 <p>Technical drawings of an expansion module. The side view shows a height of 90.00 and a width of 25.20. The front view shows a width of 60.00. The rear view shows an 'EXTENSION PORT' label. The module is labeled 'ELC-EX' and 'Eaton Culler-Hamilton'.</p>
		 <p>Technical drawings of an expansion module. The side view shows a height of 90.00 and a width of 25.20. The front view shows a width of 60.00. The rear view shows an 'EXTENSION PORT' label. The module is labeled 'ELC-EX' and 'Eaton Culler-Hamilton'.</p>

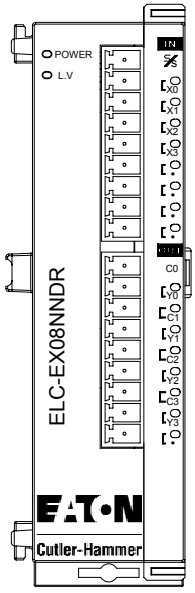
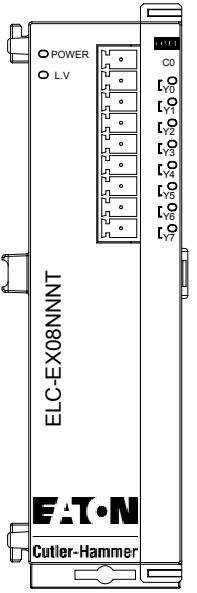
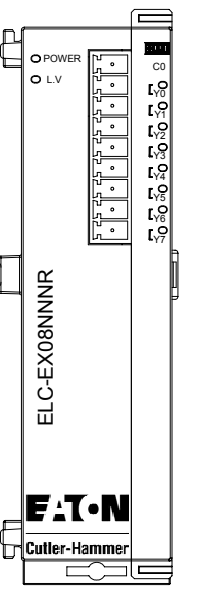
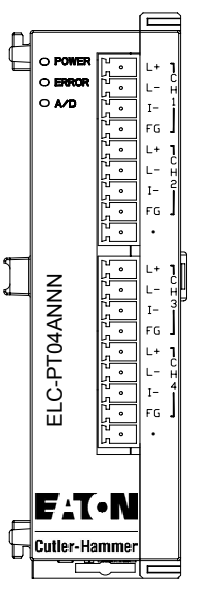
3.6 Terminal Layouts

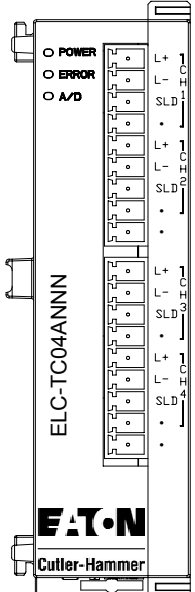
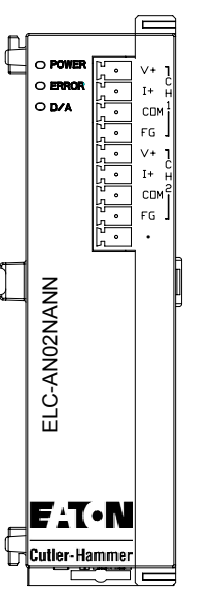
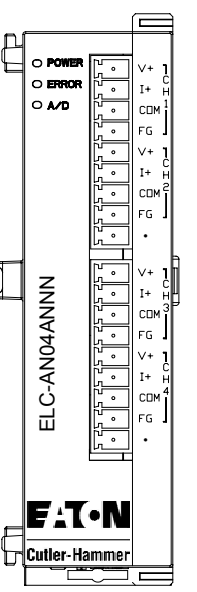
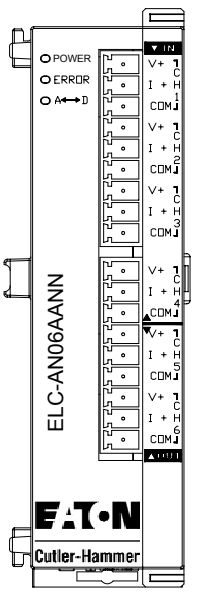




3

3

ELC-EX08NNDR	ELC-EX08NNNT	ELC-EX08NNNR	ELC-PT04ANNN
			

ELC-TC04ANNN	ELC-AN02NANN	ELC-AN04ANNN	ELC-AN06AANN
			

ELC Concepts

This chapter will cover many basic and advanced concepts of ladder logic. If you are familiar with these standard concepts, you should be able to move to the next chapter. You can always return if something is not understood.

This Chapter Contains

4.1	ELC Scan Method	4-2
4.2	Current Flow.....	4-3
4.3	Contact A, Contact B	4-4
4.4	ELC Registers and Relays	4-4
4.5	Ladder Logic Symbols	4-5
4.5.1	Creating an ELC Ladder Program.....	4-6
4.5.2	LD / LDI (Load / Load and Invert) Evaluation	4-7
4.5.3	LDP / LDF (Load Raising edge / Load Falling edge) Evaluation.....	4-8
4.5.4	AND / ANI (AND / AND and Invert) Evaluation.....	4-8
4.5.5	ANDP / ANDF (AND Raising edge / AND Falling edge) Evaluation.....	4-8
4.5.6	OR / ORI (OR / OR and Invert) Evaluation.....	4-8
4.5.7	ORP / ORF (OR Raising edge / OR Falling edge) Evaluation	4-8
4.5.8	ANB (AND Block) Evaluation (HHP Only)	4-8
4.5.9	ORB (OR Block) Evaluation (HHP Only).....	4-9
4.5.10	MPS / MRD / MPP (Branch) Evaluations (HHP Only).....	4-9
4.5.11	STL (Step Ladder) Evaluation	4-10
4.5.12	RET (Return Step) Evaluation	4-10
4.6	Conversion of ELC Commands and Each Diagram Structure	4-11
4.7	ELC-HHP Programming Methods	4-12
4.8	Simplifying Ladder Logic.....	4-14
4.9	Basic Ladder Logic Examples.....	4-17

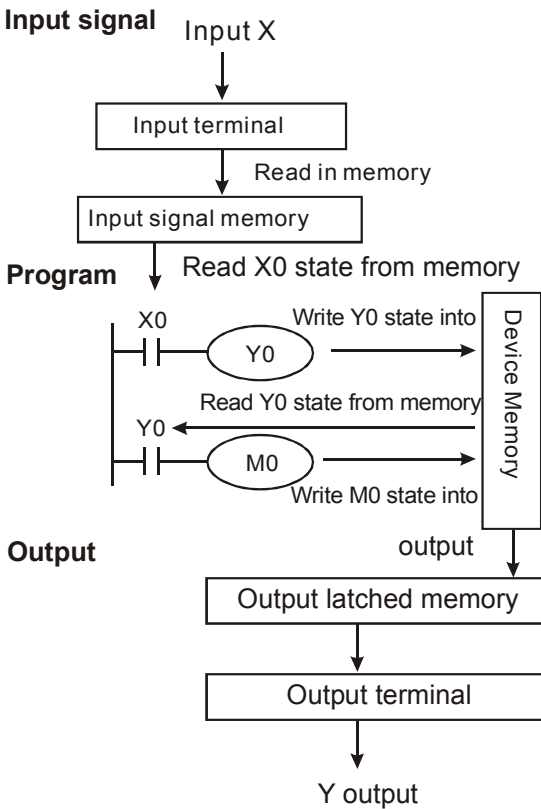
4 ELC Concepts

4.1 ELC Scan Method

ELC utilizes a standard scan method when evaluating a user application.

The steps:

Read inputs	Read the physical inputs into internal memory.
Evaluate user program	Evaluate the user program using internal memory starting with the top rung, evaluating components left to right until the reaching the bottom rung.
Write the outputs	Write the internal memory to the physical outputs



Input signal:

ELC reads the ON/OFF state of each input into memory before evaluating the user program.

Once the external inputs are read into internal memory. Any change at the external inputs will not be reflected in the internal memory until the next time (scan) the external inputs are copied into internal memory.

Program:

ELC executes each command in program left to right top to bottom placing any output ON/OFF data into internal output memory. Some of this memory is latched.

Output:

When the END command is reached the program evaluation is complete. The output memory is transferred to the external outputs.

Scan time

These three steps (read, evaluate, write) make a single scan which when timed is called scan time. Scan time in the ELC varies depending on amount of I/O and size of the user program. The general rule is the scan time increases as I/O increases and as the user program increases in size.

Reading Scan time	ELC will measure its own scan time and place the value (0.1ms increments) in register D1010, minimum scan time in register D1011, and maximum scan time in register D1012.
Measure Scan time	Scan time can also be measured by toggling an output every scan, and then measuring the pulse width on the output being toggled.
Calculate Scan time	Scan time can be calculated by add the know time required for each instruction of the user program. Timing information about each instruction is provided in the instruction chapter of this manual.

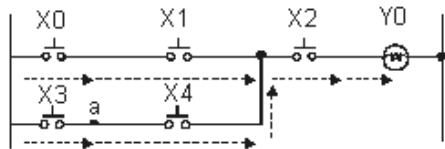
Scan time exception

ELC controllers can process certain items faster than the scan time. Some of these items interrupts, when received will halt the scan time to process the interrupt. A direct input/output command during user program evaluation allows the ELC to access I/O immediately instead of waiting until the next scan cycle.

4

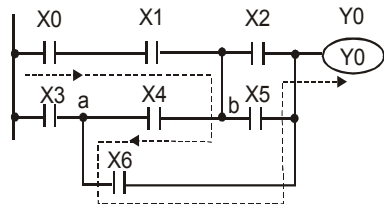
4.2 Current Flow

Ladder logic principle follows a current path that must flow from left to right. In the example below, current can flow through either the X0 or the X3 path because in both cases it flows from left to right.



Reverse Current

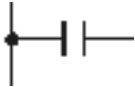
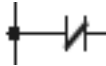
When current flow is allowed to flow from right to left, opposite of proper current flow an error will result when compiling the user program. The example below displays reverse current flow.



Following the arrowed path indicates: if X4 in on it could allow current flow from the X0 path to the X6 path, which is not the intended purpose of X4. The result is reverse current flow through X4 in this case.

4.3 Contact A, Contact B

In this manual, you will see the term contact A and contact B this is there definition.

Contact A	 Normally Open (NO) Contact
Contact B	 Normally Closed (NC) Contact

4.4 ELC Registers and Relays

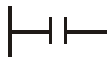






Internal to the ELC are memory locations of various sizes, functions, and names. Common names are X, Y, M, S, T, and C.

X (Relay)	X relay is the internal memory where an actual image of the external inputs is stored each scan. This memory is bit memory and counts in an octal method. i.e. X0, X1, X2...X7, X10, X11...X17, X20, X21.
Y (Relay)	Y relay is the internal memory where an actual image of the external outputs is stored each scan. This memory is bit memory and counts in an octal method. i.e. Y0, Y1...Y7, Y10, Y11...Y17, Y20, Y21.
M (Relay)	M relay is internal general working memory that can be used as needed to successfully complete the application each scan. This memory is bit memory and counts in a decimal method. i.e. M0, M1...M9, M10, M11...M19, M20, M21.
S (Relay)	S relay is internal working memory (like M) when in Step Function Control (SFC) mode instead of ladder mode. If SFC mode is not being used then these S relays can be used as general working memory when in ladder mode. This memory is bit memory and counts in a decimal method. i.e. S0, S1...S9, S10, S11...S19, S20, S21.
T (Relay) (Word) (Dword)	Timers (T) are general-purpose timers for measuring time. Depending on the timer number, its resolution ranges from 1ms to 100ms and is either 16 bit or a 32 bit timer. When the predefined timer value is reached the T relay of the same timer number will be set to ON. This memory is a bit, word, or dword value and counts in a decimal method. T0, T1...T255.

C (Relay) (Word) (Dword)	Counter (C) are general-purpose counters. Depending on the counter number, it can be either a 16 bit or a 32 bit counter. When the predefined counter value is reached the C relay of the same timer number will be set to ON. This memory is either a bit, word, or dword value and counts in a decimal method. C0, C1...C255.
D (Word)	D register is internal general working memory that can be used as needed to successfully complete the application each scan. This memory is word memory and counts in a decimal method. i.e. D0, D1...D9, D10, D11...D19, D20, D21.
E, F (Word)	E, F registers are internal index registers that point to a memory area that can be used as file storage. The E and F registers are the only way to access this storage area. If not being used for file storage, it can be used as general-purpose memory. This memory is word memory and counts in a decimal method. i.e. E0, E1..., F0, F1...
File register	File storage area. There is not a direct letter X, Y, M, D, etc access to this memory. It is accessed through read / write instructions

4.5 Ladder Logic Symbols

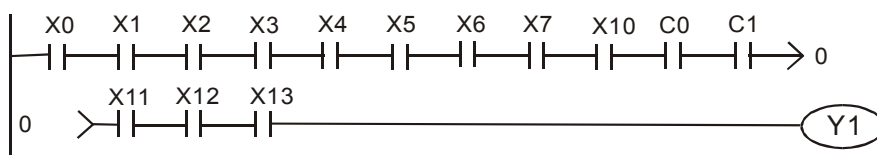
The following table displays list of ELCSoft symbols their description, command, and memory registers that are able to use the symbol.

Ladder Diagram Structure	Explanation	Command	Registers available for use
	Contact A Normally open (NO)	LD	X, Y, M, S, T, C
	Contact B Normally close (NO)	LDI	X, Y, M, S, T, C
	Serial normally open	AND	X, Y, M, S, T, C
	Parallel normally open	OR	X, Y, M, S, T, C
	Parallel normally close	ORI	X, Y, M, S, T, C
	Rising-edge trigger switch	LDP	X, Y, M, S, T, C
	Falling-edge trigger switch	LDF	X, Y, M, S, T, C

Ladder Diagram Structure	Explanation	Command	Registers available for use
	Rising-edge trigger in serial	ANDP	X, Y, M, S, T, C
	Falling-edge trigger in serial	ANDF	X, Y, M, S, T, C
	Rising-edge trigger in parallel	ORP	X, Y, M, S, T, C
	Falling-edge trigger in parallel	ORF	X, Y, M, S, T, C
	Block in serial	ANB	None
	Block in parallel	ORB	None
	Multiple output	MPS MRD MPP	None
	Output command of coil drive	OUT	Y, M, S
	Step ladder	STL	S
	Application command	Application command	Please refer chapter 6 Programming Instructions
	Inverse logic	INV	None

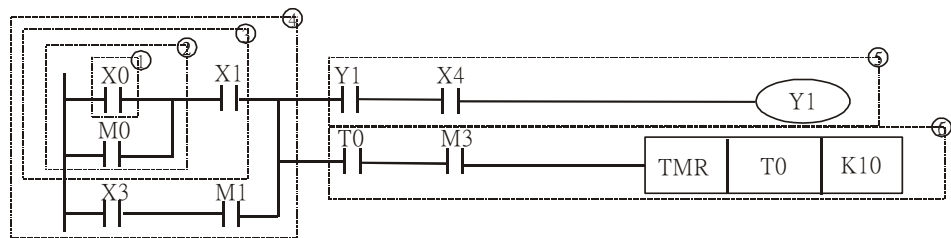
4.5.1 Creating an ELC Ladder Program

Creating an ELC ladder program is easy with ELCSoft. Most graphical editing is accomplished using a mouse. The maximum contacts allowed in a row are 11. An extension rung can be added if more contacts are needed to complete the ladder logic. The extension rung will be produced automatically when a 12th contact is added, a number will be assigned to the extension for easy tracking.



When evaluating the user program, ELC will process the ladder logic starting at the top rung evaluating each component left to right, proceed to the next rung down, and continue evaluating components left to right until ELC reaches the END statement of the user program. The following diagram for example; we analyze the process step by step. The number at the right corner is the explanation order.

Evaluating an ELC Program

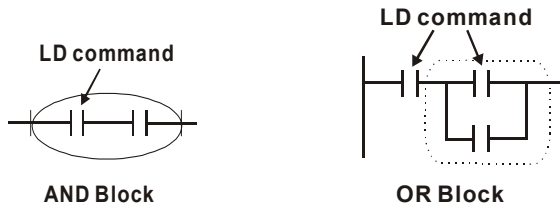


How ELC evaluates the sample program above is described below.

Execution Order	CMD	Address	Description
1	LD	X0	Load Input X0
2	OR	M0	OR internal relay M0 with input X0
3	AND	X1	AND the result of step 2 above with input X1
4	LD	X3	Load Input X3
	AND	M1	AND internal relay M1 with input X3
	ORB		OR the results of step 3 and step 4
	MPS		Push the result of OR(step3, step4) to stack
5	AND	Y1	AND the result of OR(step3, step 4) with output Y1
	AND	X4	AND input X4 with output Y1
	OUT	Y1	Output the result of above to output Y1
	MPP		Pop the result of OR(step3, step4) from stack
6	AND	T0	AND the T0 with the result of OR(step3, step4)
	AND	M3	AND T0 with M3
	TMR	T0 K10	If the result of above is ON, start timing

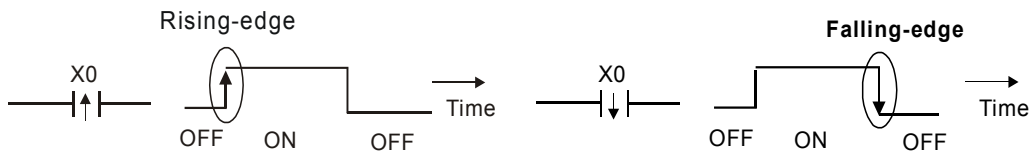
4.5.2 LD / LDI (Load / Load and Invert) Evaluation

LD or LDI start a rung or block within a rung.



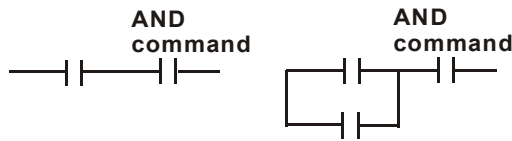
4.5.3 LDP / LDF (Load Raising edge / Load Falling edge) Evaluation

Similar to the LD command, LDP and LDF will pass an ON condition only when the device transitions from OFF → ON (LDP) or ON → OFF (LDF).



4.5.4 AND / ANI (AND / AND and Invert) Evaluation

AND (ANI) connects a device or a block in series with another device or block.

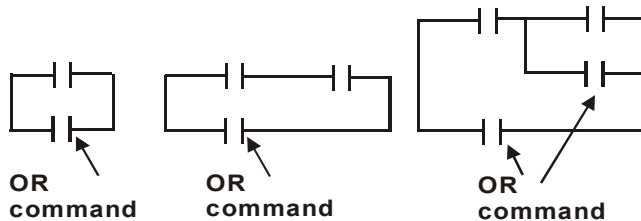


4.5.5 ANDP / ANDF (AND Raising edge / AND Falling edge) Evaluation

Similar to the AND command, ANDP and ANDF will pass an ON condition only when the device transitions from OFF → ON (ANDP) or ON → OFF (ANDF).

4.5.6 OR / ORI (OR / OR and Invert) Evaluation

OR (ORI) connects a device or a block in parallel with another device or block.



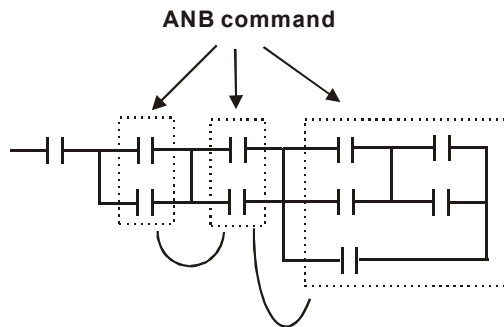
4.5.7 ORP / ORF (OR Raising edge / OR Falling edge) Evaluation

Similar to the OR command, ORP and ORF will pass an ON condition only when the device transitions from OFF → ON (ANDP) or ON → OFF (ANDF)

4.5.8 ANB (AND Block) Evaluation (HHP Only)

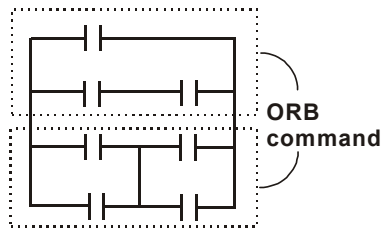
ANB will perform an AND on multiple blocks. When using ELCSOft this command is performed

automatically. When using the ELC-HHP this command must be manually inserted.



4.5.9 ORB (OR Block) Evaluation (HHP Only)


ORB will perform an OR on multiple blocks. When using ELCSoft this command is performed automatically. When using the ELC-HHP this command must be manually inserted.

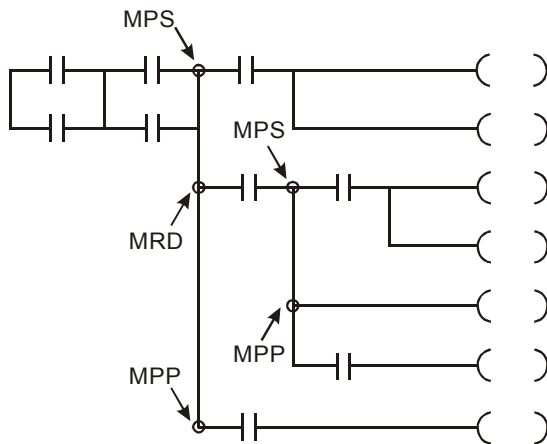


4.5.10 MPS / MRD / MPP (Branch) Evaluations (HHP Only)

These commands provide an easy method to create complex blocks within a rung of ladder logic. The three different commands provide these functions.

Branch Command	Branch Symbol	Description
MPS	┐	Starts a vertical branch. 8 of these commands are allowed each rung.
MRD	└	Starts a horizontal branch
MPP	┘	Ends branch

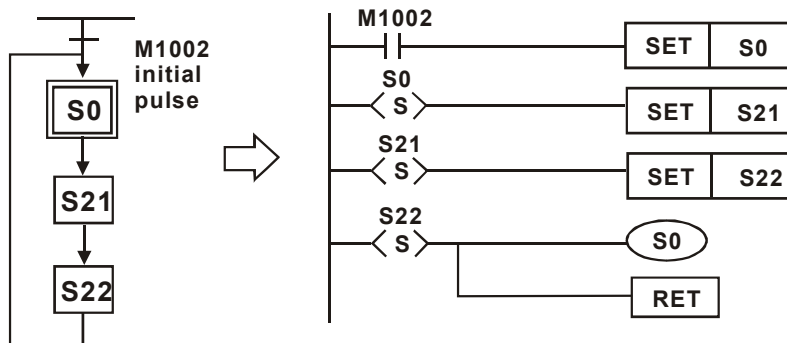
Notice an exception to the table above when reviewing the example below. There is an MPP  (looks like a MRD) starting a horizontal branch. This is due to how ELCSoft compiles the user program and is completely acceptable. If you were using ELCSoft to create this program, these commands are automatically inserted for you. If you are using the ELC-HHP, you could manually insert either an MRD or MPP command.



4.5.11 STL (Step Ladder) Evaluation

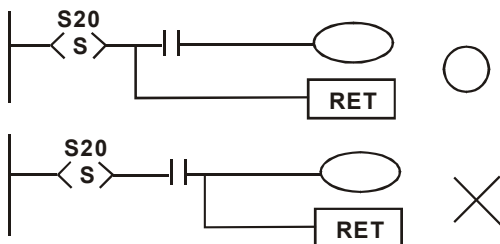
Step commands are used when using the Sequential Function Chart (SFC) mode of ELC. Programmers believe that viewing the program in this mode is easy than in ladder logic. The logic does not transition to the next step command until the current one is complete. In ladder logic this would be similar to a subroutine. Just as a subroutine is a self-contained set of instructions so is a step command.

4



4.5.12 RET (Return Step) Evaluation

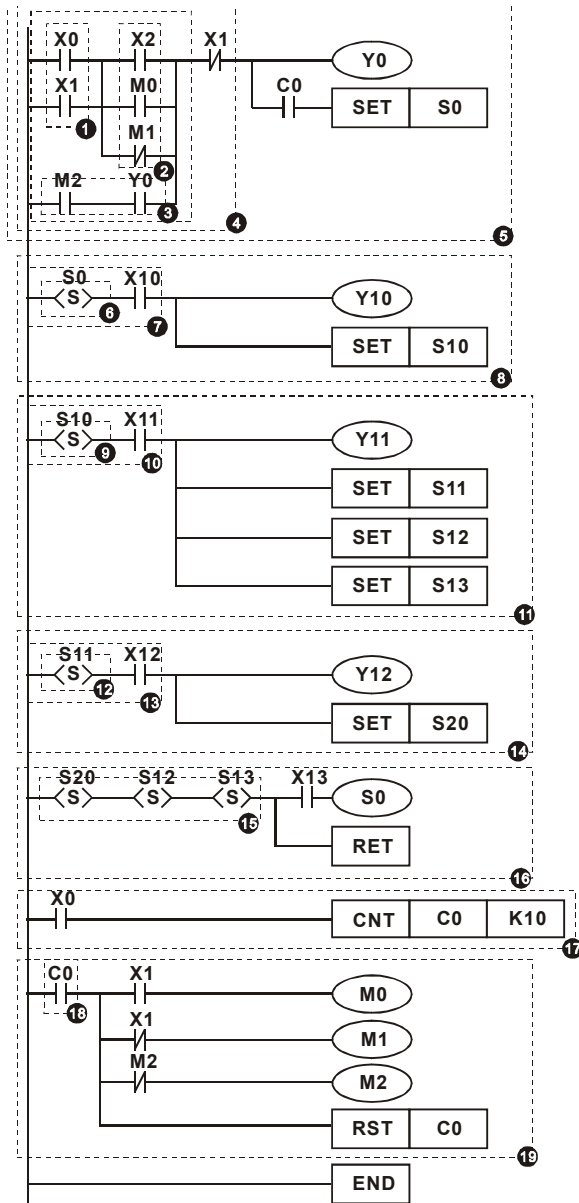
This command works in conjunction with the Step command and tells ELC that the step commands (SFC) are complete.



The RET command must be placed as in the top figure above when signaling the end of the SFC otherwise it will generate an error.

4.6 Conversion of ELC Commands and Each Diagram Structure

Ladder Diagram



Instruction

```

LD    X0
OR    X1 ① OR block
LD    X2
OR    M0 ② OR block
ORI   M1
ANB   ← Serial block
LD    M2
AND   Y0 ③ AND block
ORB   ← Serial block
ANI   X1 ④ ANI
OUT   Y0
AND   C0
SET   S0
STL   S0 ⑥ Step ladder Start
LD    X10
OUT   Y10
SET   S10 ⑧ State working item and step point transfer
STL   S10 ⑨ S10 state take out
LD    X11
OUT   Y11
SET   S11
SET   S12
SET   S13
STL   S11 ⑫ S11 state take out
LD    X12
OUT   Y12
SET   S20
STL   S20
STL   S12 ⑮ Simultaneous divergence
STL   S13
LD    X13
OUT   S0 ⑯ State working item and step point transfer
RET   S0
Return
LD    S0
CNT   C0 K10 ⑰
LD    C0 ⑱ Read C0
MPS
AND   X1
OUT   M0
MRD
ANI   X1
OUT   M1
MPP
ANI   M2
OUT   M2
RST   C0
END

```

⑤ Output state will keep on handling according to program scan state

⑩ Take out X11 state

⑬ Take out X12 state

⑰ Read C0

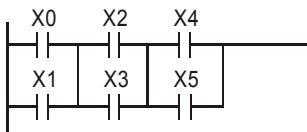
⑲ Multiple outputs

Program End

4.7 ELC-HHP Programming Methods

The analytic process of correct ladder diagram should be from left to right or from up to down. But there are some exceptions as shown in the following.

Example 1: there are two methods to enter commands for the following ladder diagram and the result is the same, but one method is better then the other.

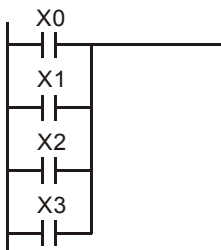


Better method		OK method	
LD	X0	LD	X0
OR	X1	OR	X1
LD	X2	LD	X2
OR	X3	OR	X3
ANB		LD	X4
LD	X4	OR	X5
OR	X5	ANB	
ANB		ANB	

4

The results for the above two programs when converted to ladder diagram are the same. One is better than the other because of how ELC operates. The operation of the program is from left side is one block merges to another one. Although the length of the program at the right side is the same as the left one, the operation of the program in the right side is merged at the last. (Command ANB is used to merge, it can't use more than 8 continuous times). In this program, it just needs to use continuous two times of command ANB and MPU allows that. But when the program needs to use more than continuous 8 times of command ANB, MPU won't allow. So the best method is to merge once the block is established and in this way the logic of programmer will be in order.

Example 2: there are two methods to use command to show the following ladder diagram but the result is the same.

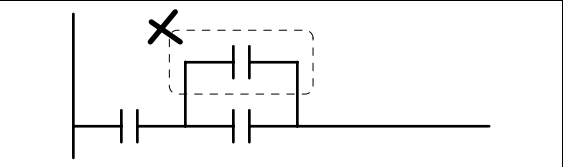
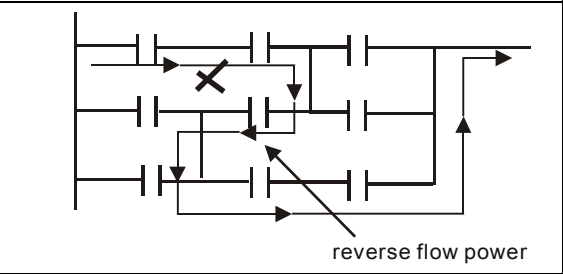
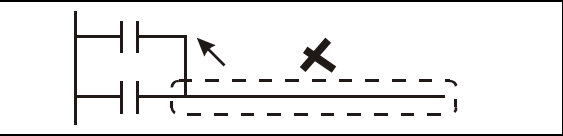
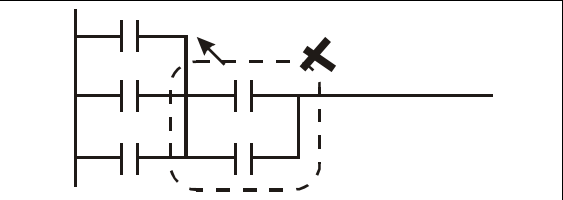
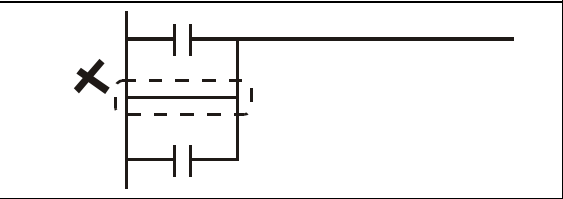
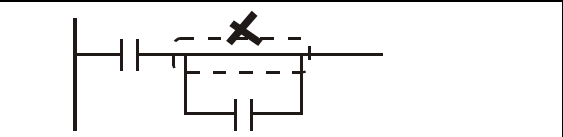
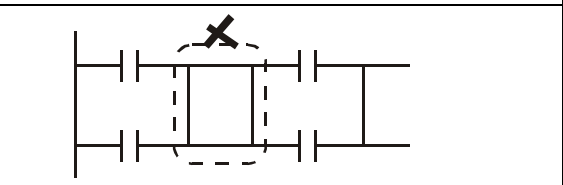
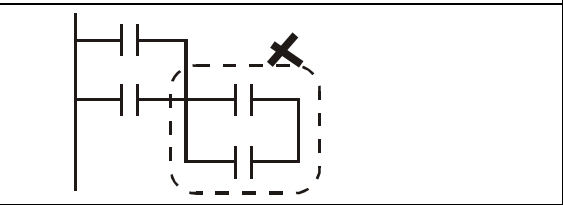


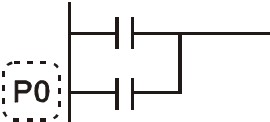
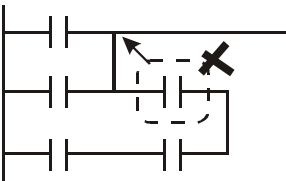
Good method		Bad method	
LD	X0	LD	X0
OR	X1	LD	X1
OR	X2	LD	X2
OR	X3	LD	X3
		ORB	
		ORB	
		ORB	

The difference is very clear in these two programs. In the bad method, the more program code it needs and the operation memory of MPU also needs to increase. So that is better to decode in the order of the definition.

Common Programming Errors

When creating ladder logic diagrams, some methods will perform better than others. Below is table listing certain methods that will cause the ELCSoft compiler to generate an error.

	OR block operation on top of rung. It should be on the bottom side of the rung.
	Reverse current flow. Current must flow right to left in a ladder circuit.
	The longest portion of the rung should be the topmost portion of the rung.
	Building rungs should have the most devices on the top of the rung with lesser amounts of device below. The dotted line block should move up.
	A blank line is not allowed since it is a short circuit.
	A blank line is not allowed since it is a short circuit
	A blank line is not allowed since it is a short circuit
	The dotted line block is positioned wrong, it should be moved up.

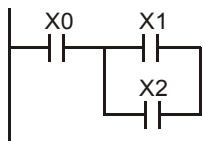
	Label P0 should be in the first row of the complete network.
	Reverse current flow. The dotted block should be moved to the top rung.

4.8 Simplifying Ladder Logic

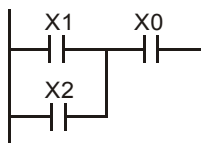
There are many ways to accomplish your ladder logic. The list below displays methods of ways to create ladder logic, some methods will not work and others could be better. For each method that will not work or could better is a suggested improvement. Review the instructions for each method. The improved method will shorten the number of instructions.

Example 1:

4

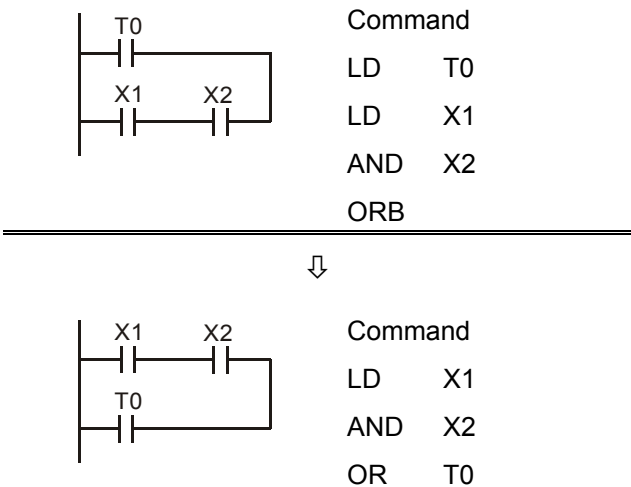


Command
LD X0
LD X1
OR X2
ANB

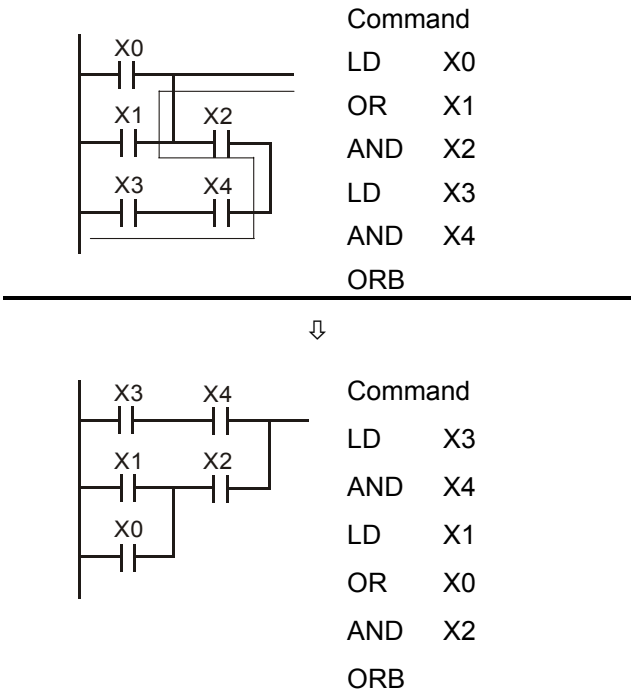


Command
LD X1
OR X2
AND X0

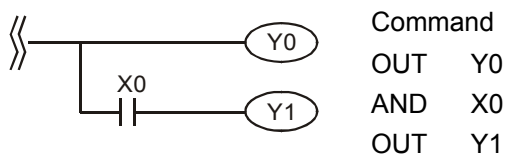
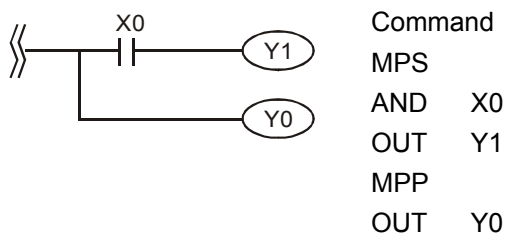
Example 2:



Example 3:

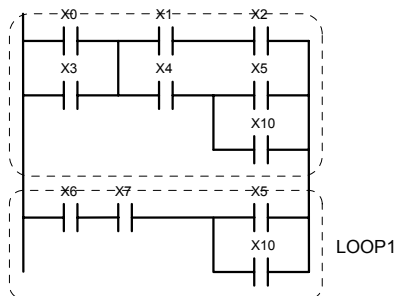
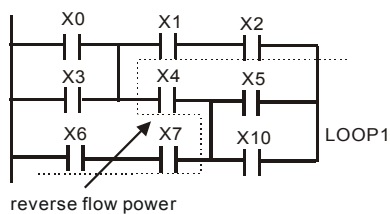


Example 4:

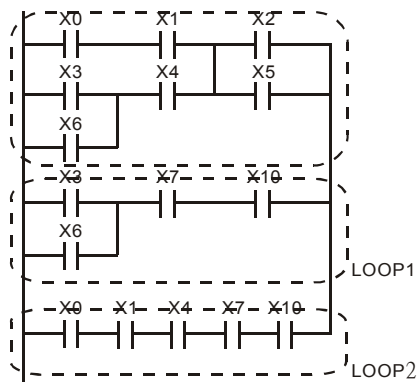
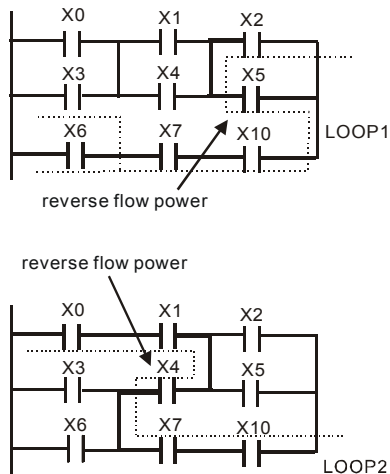


Example 5:

4



Example 6:



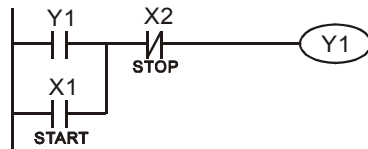
4.9 Basic Ladder Logic Examples

The examples that follow illustrate how common functions can be programmed.

Example 1 - Latch

Starting assumption: Y1 = OFF, X2 = OFF,

X1 = OFF. When Start = ON, turning Y1 = ON. Releasing Start = OFF keeps Y1 = ON until X2 = ON



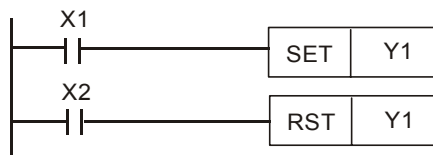
Example 2 - Latching circuit of SET and RST commands

ELC evaluate ladder logic from top rung to bottom rung and each device of each rung left to right.

Assuming X1 = Start, X2 = Stop

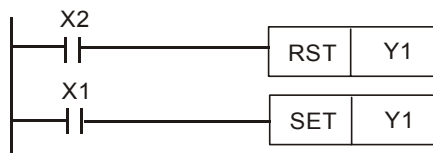
Below example is that if X1 is ON, Y1 will be ON by the SET command. If X2 = ON based Y1 = OFF by the RST command, overriding the previous SET command.

Top priority of stop



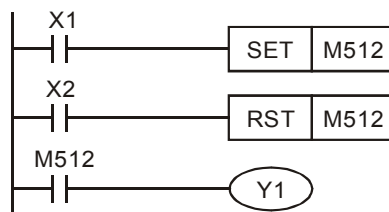
The bottom example provides the opposite logic.

Top priority of start



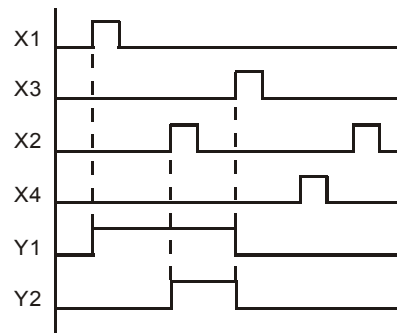
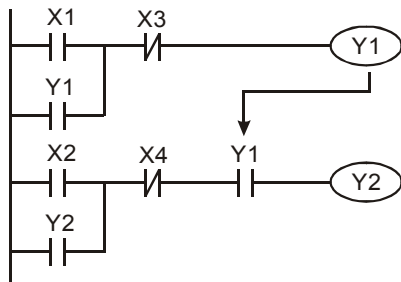
Example 3 - Latched Circuit Through Power Cycle

Once M512 is SET by X1, M512 will retain its current state even if the ELC power is cycle because M512 is a latched M relay.



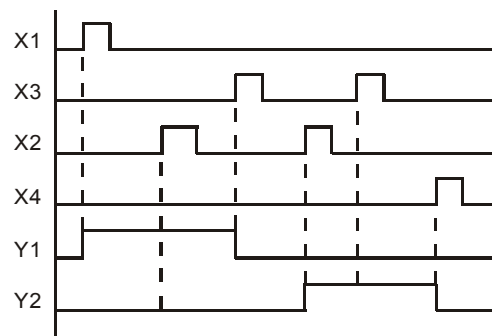
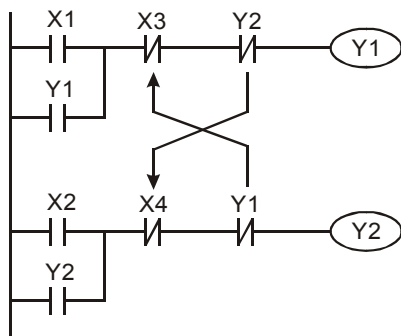
Example 4 - Conditional Control

X1 and X3 can start/stop Y1 separately, X2 and X4 can start/stop Y2 separately and they are both rungs are latched circuits. Y1 is an element for Y2 creating a conditional control circuit. Y2 will not do anything until Y1 enable it.



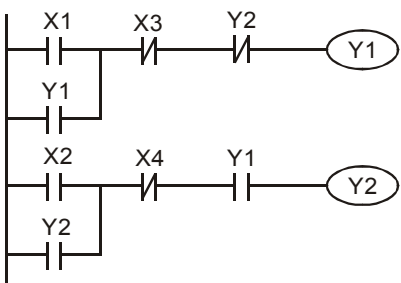
Example 5 - Interlock control

The circuit above is an interlock control circuit. Based on the ladder logic Y1 and Y2 cannot be on at the same time. When one is on, the other is locked out and vice versa (interlock). Based on how ladder logic evaluates rungs, Y1 will always have priority over Y2.



Example 6 - Sequential Control

There are times when you want control functions to happen in a sequential order. Incorporating the outputs Y1 and Y2 into each of the rung ensures a sequential order.

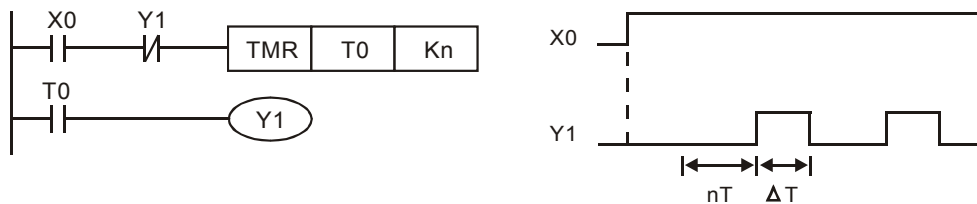


Example 7 - Oscillating Circuit

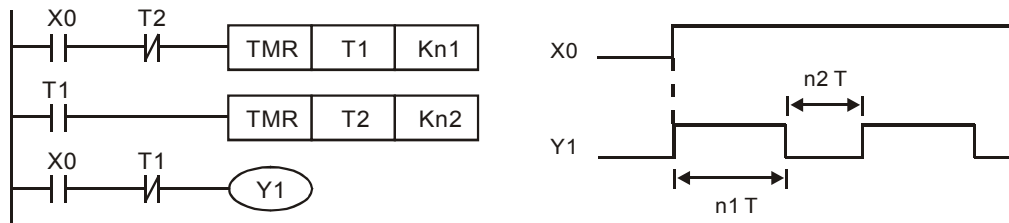
This circuit is also called a toggle circuit since it will toggle state every scan.

**Example 8 – Timer**

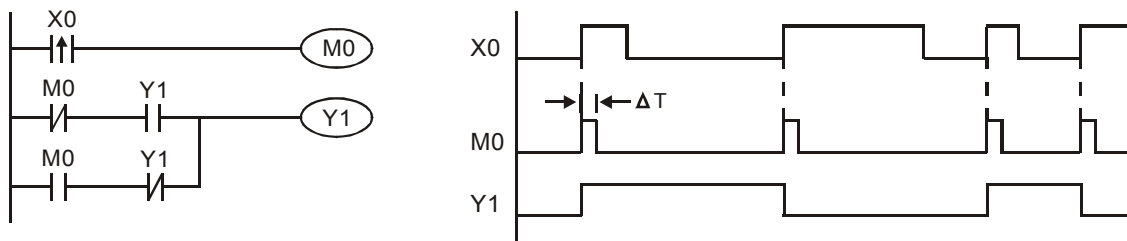
The timer above will increment when $X0 = ON$. When the timer reaches the predetermined value, Timer relay $T0 = ON$, turning $Y1 = ON$. $Y1$ will be ON for one scan then it will reset Timer 0. Starting the process over.

**Example 9 - Blinking Circuit**

Similar to a sequential circuit, this multi-timer can be programmed for almost any duty cycle desired.

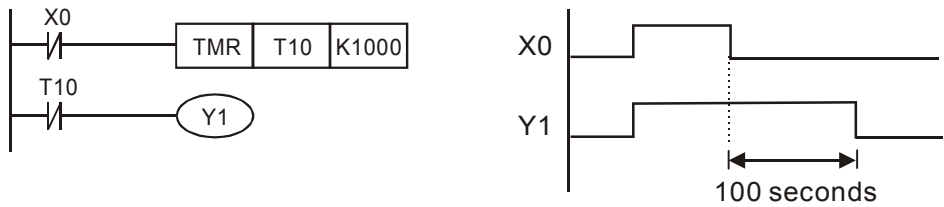
**Example 10 - Triggered Circuit**

At times, it is desired to have a rising edge pulse trigger an event. The circuit above shows the diagram how this can be done. Whenever $X0$ transitions from OFF→ON, $M0$ will pulse for one scan. This pulse can then initiate any logic desired.

**Example 11 - Delay Circuit**

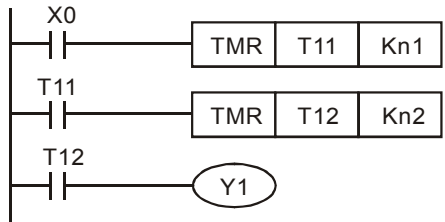
This delay circuit will hold output $Y1$ on longer than $X0$, from a start condition where $X0$ and $T10=OFF$,

when X0=OFF the timer will start timing keeping Y1=ON until timer reaches its predetermined timer value.



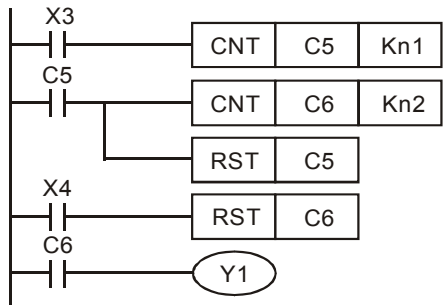
Example12 - Extended Timer Circuit

There are times when you need more time than a single timer can provide. You can easily add timers together to achieve the timing needed.



Example 13 - Extending Counter Circuit

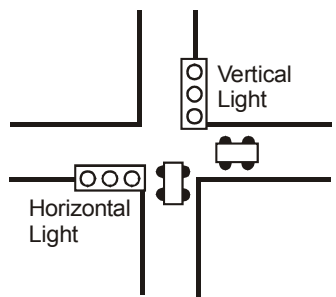
Similar to extending a timing circuit, counters can also added together to achieve a higher counter value.



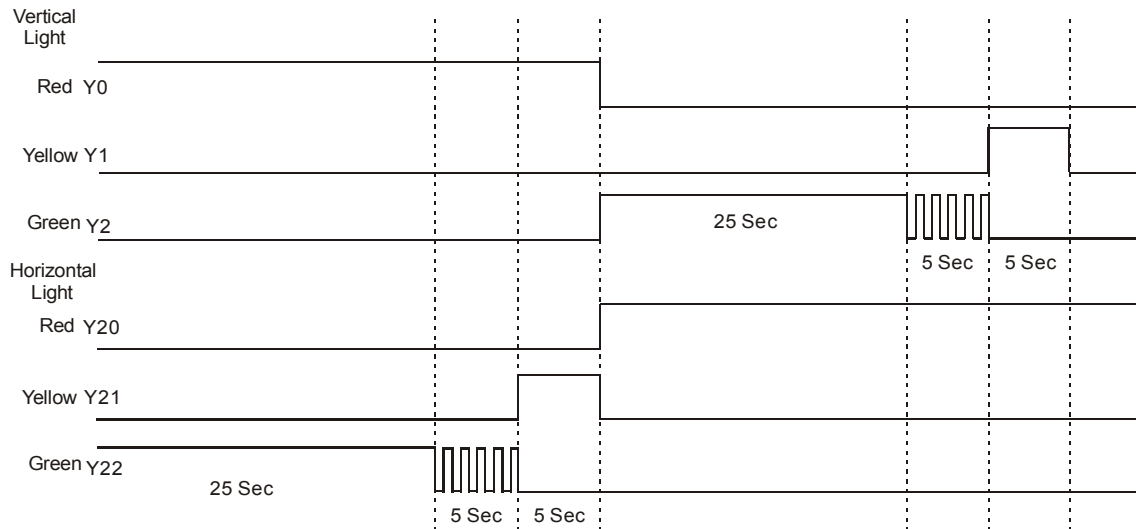
Example 14 - Traffic light control (SFC method))

Traffic light control

	Red light	Yellow light	Green light	Green blink light
Vertical light	Y0	Y1	Y2	Y2
Horizontal light	Y20	Y21	Y22	Y22
Light Time	35 Sec	5 Sec	25 Sec	5 Sec

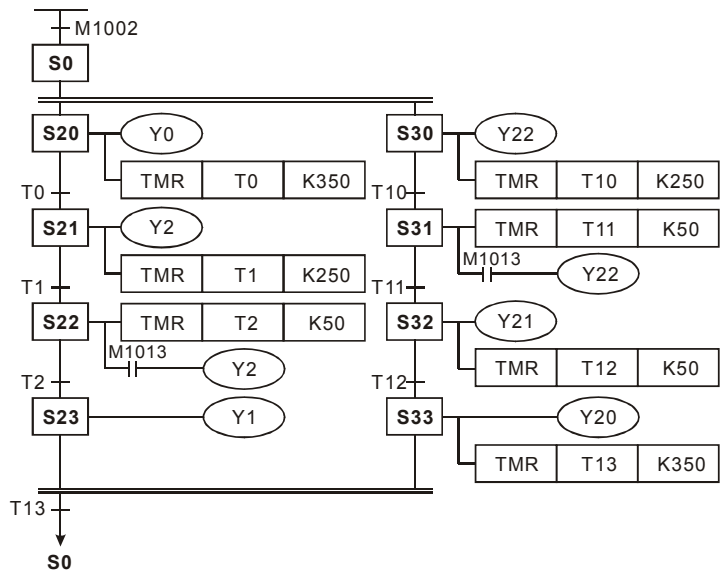


Timing chart:

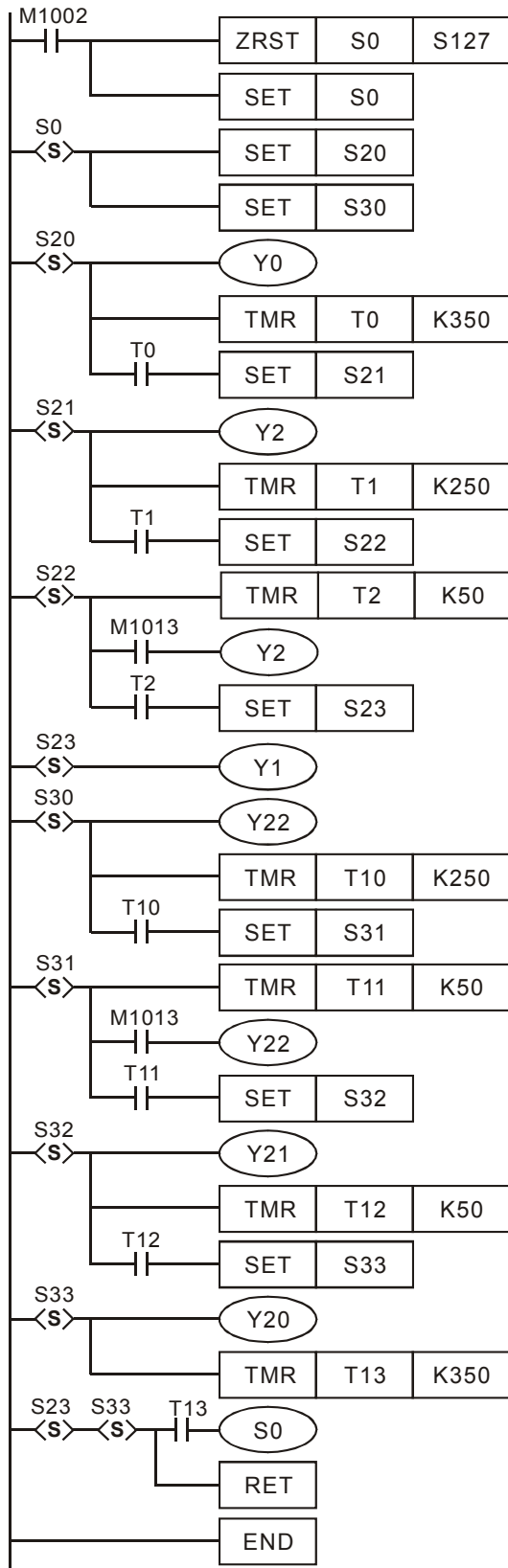


4

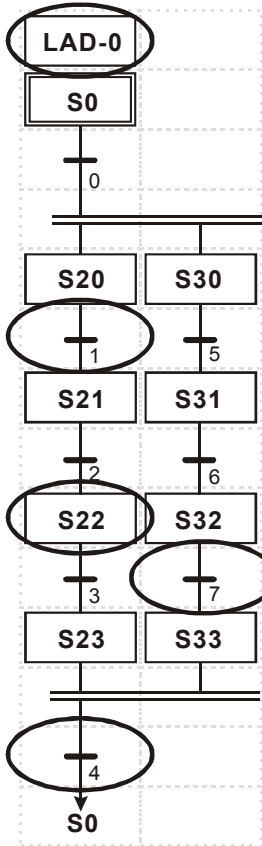
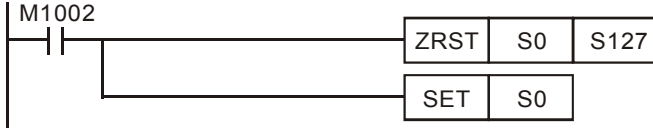

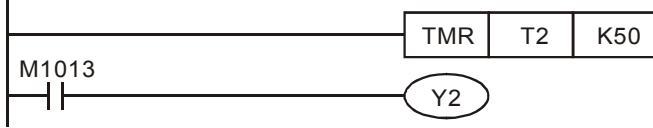


SFC Figure:



Ladder Diagram:



ELCSoft programming (SFC mode)

SFC commands	Ladder Logic View
	<p>LAD-0</p>  <p>Transferred condition 1</p>  <p>S22</p>  <p>Transferred condition 4</p>  <p>Transferred condition 7</p> 

4

MEMO

4

Programming Concepts

The Eaton Logic Controller (ELC) is a programmable logic controller spanning an I/O range of 10 –256 I/O points. ELC processors are so versatile they range from nano to small ELCs I/O and application size without ever needing to change processors. ELC can control a wide variety of devices to solve your automation needs. ELC monitors inputs and modifies outputs as controlled by the user program. User program provides features like boolean logic, counting, timing, complex math operations, and communications to other communicating products.

This Chapter Contains

5.1	ELC Memory Map for PB model	5-2
5.2	ELC Memory Map for PC/PA/PH models	5-4
5.3	ELC Latched Memory Settings for PC/PA/PH Models	5-7
5.4	ELC Latched Memory Modes	5-8
5.5	ELC Bits, Nibbles, Bytes, Words, etc	5-8
5.6	Binary, Octal, Decimal, BCD, Hex	5-9
5.7	M Relay	5-11
5.8	S Relay	5-20
5.9	T (Timer)	5-20
5.10	C (Counter)	5-21
5.11	High-speed Counters	5-23
5.12	D (Word register)	5-28
5.13	E, F Index Registers	5-38
5.14	File Register	5-38
5.15	Nest Level Pointer[N], Pointer[P], Interrupt Pointer [I]	5-39
5.16	M Relay and D Register Applications	5-42
5.17	Fault Code Information	5-71

5 Programming Concepts

5.1 ELC Memory Map for PB model

Items			Specifications		Remarks
Control Method			Stored program, cyclic scan system		
I/O Processing Method			Batch processing method (when END instruction is executed)		Fast I/O refresh instruction can override batch update
Execution Speed			Basic instructions – 3.5μ seconds minimum		Application instructions varies per instruction
Program language			Instructions + Ladder Logic + SFC		
Program Capacity			3792 Steps		Built-in EEPROM
Instructions			32 Basic instructions, Application instructions: 107		
Bit Contacts	X	External inputs	X0~X177, octal number system, 128 points max.	Total 256 I/O	Physical input points
	Y	External outputs	Y0~Y177, octal number system, 128 points max.		Physical output points
	M	Auxiliary relay	General M0~M511, M768~M999 744 points Note 1	Total 1280 bits	Main internal relay area for general use.
			Latched M512~M767, 256 points Note 3		
			Special M1000~M1279, 280 points, some are latched		
	T	Timer	100ms T0~T63, 64 points	Total 128 bits	Contact = ON when timer reaches preset value.
			10ms (M1028=ON) T64~T126, 63 points		
			1ms T127 1 points		
	C	Counter	16-bit count up C0~C111, Note 1	Total 141 bits	Contact = ON when counter reaches preset value.
			C112~C127, Note 3		
			32bit high-speed count up/down C235~C238, C241, C242, C244, 1 phase 1 input, 7 points Note 4		
			C246, C247, C249, 1 phase 2 input, 3 points Note 4		
	S	Step point	C251, C252, C254, 2 phase 2 input, 3 points Note 4	Total 128 bits	SFC usage S10~S19 is used with IST instruction
			Initial step point S0~S9, 10 points, Note 4		
			Zero return S10~S19, 10 points, Note 4		
		Latched	S20~S127, Note 4		

Items			Specifications		Remarks	
Word Register	T	Current value		T0~T127, 128 words		
	C	Current value		C0~C127, 16-bit counter,		
				C235~C254, 32-bit counter		
	D	Data register	General	D0~D407, Note 1	Total 912 words	General storage for word length data.
			Latched	D408~D599, Note 3		
Special			D1000~D1311, 312 words			
Index			E=D1028, F=D1029, Note 1			
Pointer	N	Master control loop		N0~N7, 8 points		
	P	Pointer		P0~P63, 64 points		Subroutines pointer
	I	Interrupt Service	External interrupt	I001 (X0), I101 (X1), I201 (X2), I301 (X3); 4 points (all are rising-edge trigger)		Address for interrupt subroutines
			Time interrupt	I6□□ (1ms), 1 point (□□=10~99ms)		
			Communication	I150, 1 point		
Constant	K	Decimal		K-32,768 ~ K32,767 (16-bit operation) K-2,147,483,648 ~ K2,147,483,647 (32-bit operation)		
	H	Hexadecimal		H0000 ~ HFFFF (16-bit operation), H00000000 ~ HFFFFFFFF (32-bit operation)		
Serial ports				COM1: RS-232 (Slave), COM2: RS-485 (Master/Slave), Both can be used at the same time.		
Clock/Calendar (RTC)				None		
Special Expansion Modules				Attach up to 8 modules of any type analog I/O extension modules		

Notes:

1. Data area is non-latched.
2. Default is non-latched, optionally can be set to latched.
3. Default is latched, optionally can be set to non-latched.
4. Data area is latched.

5.2 ELC Memory Map for PC/PA/PH models

Items			Specifications		Remarks	
Control Method			Stored program, cyclic scan system			
I/O Processing Method			Batch processing method (when END instruction is executed)		Fast I/O refresh instruction can override batch update	
Execution Speed			Basic instructions – 2.5μ seconds minimum		Application instructions varies per instruction	
Program language			Instructions + Ladder Logic + SFC			
Program Capacity			7920 STEPS		SRAM + Battery	
Instructions			32 Basic instructions		168 Application instructions	
Bit Contacts	X	External inputs		X0~X177, octal number system, 128 points max.	Total 256 I/O	Physical input points
	Y	External outputs		Y0~Y177, octal number system, 128 points max.		Physical output points
	M	Auxiliary relay	General	M0~M511, Note 1	Total 4096 bits	Main internal relay area for general use.
			Latched	M512~M999, Note 3		
				M2000~M4095, Note 3		
			Special	M1000~M1999 some are latched		
	T	Timer	100ms	T0~T199, Note 1	Total 256 bits	Contact = ON when timer reaches preset value.
				T192~T199 for Subroutine		
				T250~T255(accumulative), 6 points Note 4		
			10ms	T200~T239, Note 2		
				T240~T245(accumulative), 6 points, Note 4		
			1ms	T246~T249(accumulative), 4 points, Note 4		
	C	Counter	16-bit count up	C0~C95, Note 1	Total 250 bits	Contact = ON when counter reaches preset value.
				C96~C199, Note 3		
			32-bit count up/down	C200~C215, Note 1		
				C216~C234, Note 3		
32bit high-speed count up/down			C235~C244, 1 phase 1 input, 9 points, Note 3			
			C246, C247, C249, 1 phase 2 input, 3 points, Note 1			
			C251, C252, C254, 2 phase 2 input, 3 points, Note 3			

Items				Specifications		Remarks	
Bit Contacts	C	Counter		C235~C245, 1 phase 1 input, 11 points, Note 3	Total 253 bits	ELC-PH12xxxx only	
			32bit high-speed count up/down	C246, C247, C249, C250, 1 phase 2 input, 4 points Note 1			
				C251, C252, C254, 2 phase 2 input, 3 points, Note 3			
	S	Step point	Initial step point	S0~S9, 10 points Note 1	Total 1024 bits	Sequential Function Chart (SFC) usage	
			Zero point return	S10~S19, 10 points (use with IST instruction) Note 1			
			General	S20~S511, 492 points Note 1			
			Latched	S512~S895, 384 points Note 3			
			Alarm	S896~S1023, 128 points Note 3			
	Word Register	T	Current value		T0~T255, 256 words		
		C	Current value		C0~C199, 16-bit counter, 200 words		
C200~C254, 32-bit counter							
D		Data register	General	D0~D199, Note 1	Total 5000 words	General storage for word length data	
			Latched	D200~D999, Note 3			
				D2000~D4999, Note 3			
			Special	D1000~D1999, 1000 words			
			Index	E0~E3, F0~F3, Note 1			
None		File register		0~1599, 1600 words Note 4		Additional storage area to be used	
Pointer		N	Master control loop		N0~N7, 8 points		Master control nested loop
	P	Pointer		P0~P255, 256 points		Subroutine pointer	
	I	Interrupt Service	External interrupt	I001 (X0), I101 (X1), I201 (X2), I301 (X3), I401 (X4), I501 (X5); 6 points (all are rising-edge trigger)		Address for interrupt subroutines	
			Time interrupt	I6□□ (1ms), I7□□ (1ms), (□□ =1~99ms)			
			Hi-speed counter	I010, I020, I030, I040, I050, I060; 6 points			
			Communication	I150, 1 points			

Items			Specifications	Remarks
Constant	K	Decimal	K-32,768 ~ K32,767 (16-bit operation), K-2,147,483,648 ~ K2,147,483,647 (32-bit operation)	
	H	Hexadecimal	H0000 ~ HFFFF (16-bit operation), H00000000 ~ HFFFFFFFF (32-bit operation)	
Serial ports			COM1: RS-232, COM2: RS-485 (Master/Slave), Both can be used at the same time. COM1 is typically the programming port.	
Clock/Calendar (RTC)			Year, Month, Day, Hours, Minutes, Seconds	
Analog Volume dial			ELC-PC12xxxx, ELC-PH12xxxx only	
Special Expansion Modules			Attach up to 8 modules of any type analog I/O extension modules	

Notes:

1. Data area is non-latched.
2. Default is non-latched, optionally can be set to latched.
3. Default is latched, optionally can be set to non-latched.
4. Data area is latched.

5.3 ELC Latched Memory Settings for PC/PA/PH Models

M Auxiliary relay	General		Latched		Special auxiliary relay		Latched			
	M0~M511		M512~M999		M1000~M1999		M2000~M4095			
	Not latched		Start: D1200(K512) End: D1201(K999) Default is latched		Some are latched and they can't be changed.		Start: D1202(K2000) End: D1203(K4095) Default is latched			
T Timer	100 ms		10 ms		10ms		1 ms		100 ms	
	T0 ~T199		T200~T239		T240~T245		T246~T249		T250~T255	
	Factory setting is non-latched		Factory setting is non-latched		Accumulative latched					
C Counter	16-bit count up				32-bit count up/down				32-bit high-speed count up/down	
	C0~C95		C96~C199		C200~C215		C216~C234		C235~C255	
	Non-latched		Latched (default)		Non-latched		Latched (default)		Latched (default)	
			Start: D1208 (K96) End: D1209 (K199)				Start: D1210 (K216) End: D1211 (K234)		Start: D1212 (K235) End: D1213 (K255)	
S Step relay	General	Latched	Special		Latched				General	
	S0~S9	S10~S19	S20~S511		S512~S895				S896~S1023	
	Non-latched				Latched (default)				Latched	
Start: D1214 (K512) End: D1215 (K895)										
D Register	General		Latched		Special register		Latched			
	D0~D199		D200~D999		D1000~D1999		D2000~D4999			
	Non-latched		Latched (default)		Some is latched, and can't be changed		Latched (default)			
			Start: D1216 (K200) End: D1217 (K999)				Start: D1218 (K2000) End: D1219 (K4999)			
File Register	K0-1599									
	Latched									

5.4 ELC Latched Memory Modes

PB Model

Memory type	Power OFF=>ON	STOP=>RUN	RUN=>STOP	Clear all M1031 Non-latched area	Clear all M1032 latched area	Factory setting
Non-latched	Clear	When M1033=OFF, clear		Clear	Unchanged	0
		When M1033=ON, No change				
Latched	Unchanged			Unchanged	Clear	Unchanged
Special M, Special D, Index Register	Initial	Unchanged		Unchanged		Initial setting

PC/PA/PH Models

Memory type	Power OFF=>ON	STOP=>RUN	RUN=>STOP	Clear all M1031 Non-latched area	Clear all M1032 latched area	Factory setting
Non-latched	Clear	Unchanged	When M1033=OFF, clear When M1033=ON, No change	Clear	Unchanged	0
Latched	Unchanged			Unchanged	Clear	0
Special M, Special D, Index Register	Initial	Unchanged		Unchanged		Initial setting
File Register	Unchanged					0

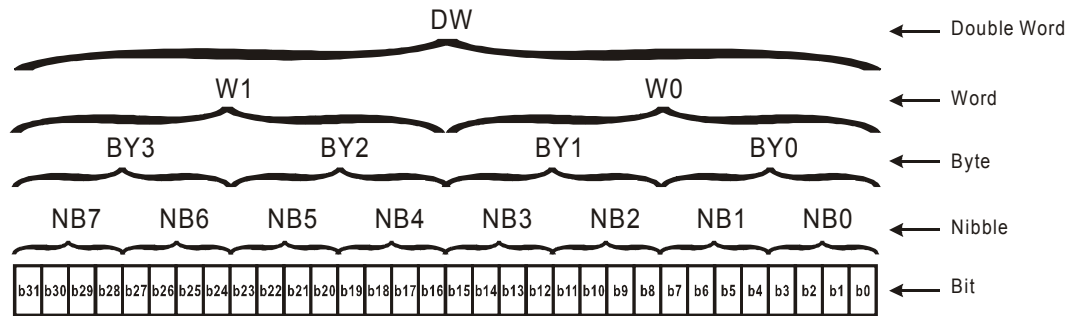
5.5 ELC Bits, Nibbles, Bytes, Words, etc

ELC utilizes five numeric types to perform different instructions. The following is the explanation of numeric types.

Numeric	Description
Bit	Bit is the basic unit of a binary number system. Range is 0 or 1
Nibble	4 consecutive bits, such as b3~b0. Range 0 – 9 (BCD) or 0~F hexadecimal.
Byte	8 consecutive bits b7~b0 Range 0 – 255 or 00 - FF hexadecimal
Word	16 consecutive bits (2 consecutive bytes) b15~b0 Range -32,768 ~ 32,767 or 0000 ~ FFFF hexadecimal

Numeric	Description
Double Word	32 consecutive bits (2 consecutive words) b31~b0 Range -2,147,483,648 ~ 2,147,483,647 or 00000000 - FFFFFFFF hexadecimal

The relationship among bit, nibble, byte, word, and double word are shown as below.



5.6 Binary, Octal, Decimal, BCD, Hex

ELC is capable of using many different numbering systems.

Number System	ELC Operation Use
Binary	Bit operations. Y, X, M, S, D, E, F, T, C
Octal	Input & Output operations. Y and X relay
Decimal (K)	Any non-bit operation. D, E, F, T, C registers can use decimal. A (K) prefix to a number in ELC represents a decimal number. i.e. K100 = Decimal 100
BCD (Binary Code Decimal)	Used mainly for sending data to 7-segement displays.
Hex (H)	Any non-bit operation. D, E, F, T, C registers can use Hexadecimal. A (H) prefix to a number in ELC represents a Hex number. i.e. K255 = Hex FF

Example: ELC Expansion Module Addressing (Octal)

ELC controllers have included I/O in them. For this reason expansion modules I/O address starts at X20 for inputs and Y20 for outputs.

Reference Chart

Binary (BIN)	Octal (OCT)	Decimal (K) (DEC)	BCD (Binary Code Decimal)	Hexadecimal (H) (HEX)
For ELC internal operation	X, Y relay	Memory registers M, S, T, C, D, E, F, P, I number	For DIP Switch and 7-segment display	Constant H
0000	0	0	0000	0
0001	1	1	0001	1
0010	2	2	0010	2
0011	3	3	0011	3
0100	4	4	0100	4
0101	5	5	0101	5
0110	6	6	0110	6
0111	7	7	0111	7
1000	10	8	1000	8
1001	11	9	1001	9
1010	12	10	0000	A
1011	13	11	0001	B
1100	14	12	0010	C
1101	15	13	0011	D
1110	16	14	0100	E
1111	17	15	0101	F
10000	20	16	0110	10
10001	21	17	0111	11

5.7 M Relay

The special auxiliary relay(special M) are as shown in the following. Please notice that some equipments with the same number will be different to the different model. In the following chart, the meaning of column "Attribute" are: "R" means can only read. "R/W" means can read/write. "-" means can do nothing. "#" means system setting, user can read the detail explanation of the setting in the manual.

Special M	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
M1000	Normally open contact (a contact). This contact is ON when running and it is ON when the status is set to RUN.	Y	Y	OFF	ON	OFF	R	NO	OFF
M1001	Normally OFF contact (b contact). This contact is OFF in running and it is OFF when the status is set to RUN.	Y	Y	ON	OFF	ON	R	NO	ON
M1002	ON only for 1 scan after RUN. Initial pulse is contact a. It will get positive pulse in the RUN moment. Pulse width=scan period.	Y	Y	OFF	ON	OFF	R	NO	OFF
M1003	OFF only for 1 scan after RUN. Initial pulse is contact a. It will get negative pulse in the RUN moment. Pulse width=scan period.	Y	Y	ON	OFF	ON	R	NO	ON
M1004	ON when error occurs	Y	Y	OFF	OFF	-	R	NO	OFF
M1008	Monitor timer flag (ON: ELC WDT time out)	Y	Y	OFF	OFF	-	R	NO	OFF
M1010	PLSY Y0 mode selection. ON = continuous output	Y	Y	OFF	-	-	R/W	NO	OFF
M1011	10ms clock pulse, 5ms ON/5ms OFF	Y	Y	OFF	-	-	R	NO	OFF
M1012	100ms clock pulse, 50ms ON / 50ms OFF	Y	Y	OFF	-	-	R	NO	OFF
M1013	1s clock pulse, 0.5s ON / 0.5s OFF	Y	Y	OFF	-	-	R	NO	OFF
M1014	1min clock pulse, 30s ON / 30s OFF	Y	Y	OFF	-	-	R	NO	OFF
M1015	High-speed connection counter	-	Y	OFF	-	-	R/W	NO	OFF
M1016	Display year bit. When OFF = display two right-most bits. When ON = display (two right-most bits + 2000).	-	Y	OFF	-	-	R/W	NO	OFF
M1017	±30 seconds adjustment	-	Y	OFF	-	-	R/W	NO	OFF
M1018	Flag for Radian/Degree, ON for degree	-	Y	OFF	-	-	R/W	NO	OFF
M1020	Zero flag	Y	Y	OFF	-	-	R	NO	OFF
M1021	Barrow flag	Y	Y	OFF	-	-	R	NO	OFF
M1022	Carry flag	Y	Y	OFF	-	-	R	NO	OFF
M1023	PLSY Y1 mode selection, ON = continuous output.	Y	Y	OFF	-	-	R/W	NO	OFF
M1025	If ELC receives an illegal communication request when HHP, PC or HMI connects to ELC, M1025 =ON and save the error code in D1025.	Y	Y	OFF	-	-	R	NO	OFF

Special M	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
M1026	Startup flag of RAMP module	-	Y	OFF	-	-	R/W	NO	OFF
M1027	PR output flag	-	Y	OFF	-	-	R/W	NO	OFF
M1028	10ms time switch flag. The base setting flag of T64~T126 is 100ms, when timer is OFF and the base setting flag is 10ms when it is ON.	Y	-	OFF	-	-	R/W	NO	OFF
M1029	Pulse output Y0 of PLSY and PLSR instruction execution completed or other relative instruction execution completed	Y	Y	OFF	-	-	R	NO	OFF
M1030	Pulse output Y1 of PLSY and PLSR instruction execution completed	Y	Y	OFF	-	-	R	NO	OFF
M1031	Clear all non-latched memory	Y	Y	OFF	-	-	R/W	NO	OFF
M1032	Clear all latched memory	Y	Y	OFF	-	-	R/W	NO	OFF
M1033	Memory latched at STOP	Y	Y	OFF	-	-	R/W	NO	OFF
M1034	All Y outputs disable	Y	Y	OFF	-	-	R/W	NO	OFF
M1035	Start X7 input point to be RUN/STOP switch and correspond to D1035	-	Y	-	-	-	R/W	YES	OFF
M1039	Constant scan mode	Y	Y	OFF	-	-	R/W	NO	OFF
M1040	Step transition inhibit	Y	Y	OFF	-	-	R/W	NO	OFF
M1041	Step transition start	Y	Y	OFF	-	OFF	R/W	NO	OFF
M1042	Start pulse	Y	Y	OFF	-	-	R/W	NO	OFF
M1043	Zero point return completed	Y	Y	OFF	-	OFF	R/W	NO	OFF
M1044	Zero point condition	Y	Y	OFF	-	OFF	R/W	NO	OFF
M1045	All outputs clear inhibit	Y	Y	OFF	-	-	R/W	NO	OFF
M1046	STL state setting (ON)	Y	Y	OFF	-	-	R	NO	OFF
M1047	STL monitor enable	Y	Y	OFF	-	-	R/W	NO	OFF
M1048	Flag for alarm point state	-	Y	OFF	-	-	R	NO	OFF
M1049	Monitor flag for alarm point	-	Y	OFF	-	-	R/W	NO	OFF
M1050	I001 masked	Y	Y	OFF	-	-	R/W	NO	OFF
M1051	I101 masked	Y	Y	OFF	-	-	R/W	NO	OFF
M1052	I201 masked	Y	Y	OFF	-	-	R/W	NO	OFF
M1053	I301 masked	Y	Y	OFF	-	-	R/W	NO	OFF
M1054	I401 masked	-	Y	OFF	-	-	R/W	NO	OFF
M1055	I501 masked	-	Y	OFF	-	-	R/W	NO	OFF
M1056	I6□□ masked	Y	Y	OFF	-	-	R/W	NO	OFF
M1057	I7□□ masked	-	Y	OFF	-	-	R/W	NO	OFF
M1059	I010~I060 masked	-	Y	OFF	-	-	R/W	NO	OFF
M1060	System error message 1	Y	Y	OFF	-	-	R	NO	OFF
M1061	System error message 2	Y	Y	OFF	-	-	R	NO	OFF
M1062	System error message 3	Y	Y	OFF	-	-	R	NO	OFF
M1063	System error message 4	Y	Y	OFF	-	-	R	NO	OFF
M1064	Operator error	Y	Y	OFF	OFF	-	R	NO	OFF
M1065	Syntax error	Y	Y	OFF	OFF	-	R	NO	OFF
M1066	Program error	Y	Y	OFF	OFF	-	R	NO	OFF

Special M	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
M1067	Program execution error	Y	Y	OFF	OFF	-	R	NO	OFF
M1068	Execution error latched (D1068)	Y	Y	OFF	-	-	R	NO	OFF
M1070	Time pulse unit switch of PWM instruction Y1. When = ON, the time pulse unit is 100us and when = OFF, the time pulse unit is 1ms.	Y	Y	OFF	-	-	R/W	NO	OFF
M1072	Execute ELC RUN instruction	Y	Y	OFF	ON	OFF	R/W	NO	OFF
M1076	Perpetual calendar error	-	Y	OFF	-	-	R	NO	OFF
M1077	Battery voltage is too low or malfunction	-	Y	OFF	-	-	R	NO	OFF
M1078	PLSY instruction Y0 pulse output stop immediately flag	Y	Y	OFF	-	-	R/W	NO	OFF
M1079	PLSY instruction Y1 pulse output stop immediately flag	Y	Y	OFF	-	-	R/W	NO	OFF
M1081	FLT instruction change direction flag	-	Y	OFF	-	-	R/W	NO	OFF
M1083	Enable/Disable interrupt in FROM/TO mode	-	Y	OFF	-	-	R/W	NO	OFF
M1088	Matrix compared flag. If the result is the same, M1088 = 1. If the result is different, M1088 = 0.	-	Y	OFF	OFF	-	R/W	NO	OFF
M1089	Matrix search start flag. Compare from the first bit and M1090=1.	-	Y	OFF	OFF	-	R	NO	OFF
M1090	Matrix search start flag. Compare from the first bit and M1090=1.	-	Y	OFF	OFF	-	R	NO	OFF
M1091	Matrix finding bit flag. When find it, it will stop comparing and M1091=1.	-	Y	OFF	OFF	-	R	NO	OFF
M1092	Matrix pointer error flag. When pointer Pr exceeds this range, M1092=1.	-	Y	OFF	OFF	-	R	NO	OFF
M1093	Matrix pointer increase flag. It will add 1 to present pointer.	-	Y	OFF	OFF	-	R/W	NO	OFF
M1094	Matrix pointer clear flag. It will clear present pointer to 0.	-	Y	OFF	OFF	-	R/W	NO	OFF
M1095	Carry flag for matrix rotate/shift output	-	Y	OFF	OFF	-	R	NO	OFF
M1096	Complement flag for matrix shift input	-	Y	OFF	OFF	-	R/W	NO	OFF
M1097	Direction flag for matrix rotate/shift	-	Y	OFF	OFF	-	R/W	NO	OFF
M1098	Matrix count bit 0 or 1 flag	-	Y	OFF	OFF	-	R/W	NO	OFF
M1099	It is ON when matrix count result 0	-	Y	OFF	OFF	-	R/W	NO	OFF
M1100	A sampling flag of SPD instruction	-	Y	OFF	OFF	-	R/W	NO	OFF
M1101	Start file register or not	-	Y	-	-	-	R/W	Yes	OFF
M1115	Start switch for accel/decel pulse output	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1116	Acceleration flag	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1117	Target attained frequency flag	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1118	Deceleration flag	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1119	Completed function flag	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1120	Communication protocol holding	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1121	Transmission ready	Y	Y	OFF	OFF	ON	R	NO	OFF
M1122	Sending request	Y	Y	OFF	OFF	OFF	R/W	NO	OFF

Special M	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
M1123	Receiving completed	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1124	Receiving wait	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1125	Communication reset	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1126	STX/ETX user/system selection	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1127	MODRD/RDST/MODRW instructions. Data receiving completed.	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1128	Transmitting/Receiving Indication	Y	Y	OFF	OFF	OFF	R/W	NO	OFF
M1129	Receiving time out	Y	Y	OFF	OFF	-	R/W	NO	OFF
M1130	STX/ETX selection	Y	Y	OFF	OFF	-	R/W	NO	OFF
M1131	MODRD/RDST/MODRW, M1131=ON when data convert to HEX	Y	Y	OFF	OFF	-	R	NO	OFF
M1132	ON= no relative communications instruction in ELC program.	Y	Y	OFF	-	-	R	NO	OFF
M1133	Special high speed pulse (50KHz) output switch (ON = start)	-	Y	OFF	OFF	OFF	R/W	NO	OFF
M1134	ON is continuous output switch	-	Y	OFF	OFF	-	R/W	NO	OFF
M1135	Output pulse numbers attained flag	-	Y	OFF	OFF	OFF	R/W	NO	OFF
M1138	COM1 (RS-232) communication protocol settings	Y	Y	OFF	-	-	R/W	NO	OFF
M1139	COM1 (RS-232) ASCII/RTU mode when ELC is slave. OFF=ASCII, ON=RTU	Y	Y	OFF	-	-	R/W	NO	OFF
M1140	MODRD/MODWR/MODRW data received error	Y	Y	OFF	OFF	-	R	NO	OFF
M1141	MODRD/MODWR/MODRW instruction error	Y	Y	OFF	OFF	-	R	NO	OFF
M1142	MVX instruction data received error	Y	Y	OFF	OFF	-	R	NO	OFF
M1143	1. COM2(RS-485) ASCII/RTU mode selections when ELC is SLAVE (it is OFF when in ASCII mode and it is ON when in RTU) 2. COM2(RS-485) ASCII/RTU mode selections when ELC is MASTER (used with MODRD/MODWR/MODRW instructions) (it is OFF when in ASCII mode and it is ON when in RTU mode)	Y	Y	OFF	OFF	-	R/W	NO	OFF
M1144	Output start switch of accel/decel pulse output function of adjustable slope	-	Y	OFF	OFF	OFF	R/W	NO	OFF
M1145	Acceleration flag of accel/decel pulse output function of adjustable slope	-	Y	OFF	OFF	-	R	NO	OFF
M1146	Target attained frequency flag of accel/decel pulse output function of adjustable slope	-	Y	OFF	OFF	-	R	NO	OFF
M1147	Deceleration flag of accel/decel pulse output function of adjustable slope	-	Y	OFF	OFF	-	R	NO	OFF
M1148	Complete function flag of accel/decel pulse output function of adjustable slope	-	Y	OFF	OFF	OFF	R/W	NO	OFF
M1149	Stop counting temporality flag of accel/decel pulse output function of adjustable slope	-	Y	OFF	OFF	-	R/W	NO	OFF
M1154	Start designated deceleration function flag of accel/decel pulse output function of adjustable	-	Y	OFF	-	-	R/W	NO	OFF

Special M	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
	slope								
M1161	8/16 bits mode (ON = 8 bit mode)	Y	Y	OFF	-	-	R/W	NO	OFF
M1167	HKY input is 16 bits mode	-	Y	OFF	-	-	R/W	NO	OFF
M1168	SMOV working mode indication	-	Y	OFF	-	-	R/W	NO	OFF
M1172	2-phase pulse output switch (on is start)	-	Y	OFF	OFF	OFF	R/W	NO	OFF
M1173	ON is continuous output switch	-	Y	OFF	-	-	R/W	NO	OFF
M1174	Output pulse number attained flag	-	Y	OFF	OFF	OFF	R/W	NO	OFF
M1178	VR0 Variable resistor enable	-	Y	OFF	-	-	R/W	NO	OFF
M1179	VR1 Variable resistor enable	-	Y	OFF	-	-	R/W	NO	OFF
M1196	7-Seg Display mode. ON=Hex, OFF=Decimal PA model only	-	Y	OFF	-	-	R/W	NO	OFF
M1197	7-Seg display. Display decimal point to the right of the LSD. PA model only	-	Y	OFF	-	-	R/W	NO	OFF
M1198	7-Seg display. Display decimal point to the right of the MSD. PA model only	-	Y	OFF	-	-	R/W	NO	OFF
M1200	C200 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1201	C201 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1202	C202 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1203	C203 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1204	C204 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1205	C205 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1206	C206 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1207	C207 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1208	C208 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1209	C209 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1210	C210 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1211	C211 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1212	C212 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1213	C213 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1214	C214 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1215	C215 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1216	C216 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1217	C217 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1218	C218 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF

Special M	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
M1219	C219 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1220	C220 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1221	C221 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1222	C222 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1223	C223 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1224	C224 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1225	C225 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1226	C226 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1227	C227 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1228	C228 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1229	C229 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1230	C230 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1231	C231 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1232	C232 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1233	C233 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1234	C234 counting mode (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1235	C235 counting mode (on: count down)	Y	Y	OFF	-	-	R/W	NO	OFF
M1236	C236 counting mode (on: count down)	Y	Y	OFF	-	-	R/W	NO	OFF
M1237	C237 counting mode (on: count down)	Y	Y	OFF	-	-	R/W	NO	OFF
M1238	C238 counting mode (on: count down)	Y	Y	OFF	-	-	R/W	NO	OFF
M1239	C239 counter mode setting (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1240	C240 counter mode setting (on: count down)	-	Y	OFF	-	-	R/W	NO	OFF
M1241	C241 counter mode setting (on: count down)	Y	Y	OFF	-	-	R/W	NO	OFF
M1242	C242 counter mode setting (on: count down)	Y	Y	OFF	-	-	R/W	NO	OFF
M1243	C243 counter mode setting (on: count down) PH model only	-	Y	OFF	-	-	R/W	NO	OFF
M1244	C244 counter mode setting (on: count down)	Y	Y	OFF	-	-	R/W	NO	OFF
M1245	C245 counter mode setting (on: count down) PH model only	-	Y	OFF	-	-	R/W	NO	OFF
M1246	C246 counter monitor (on: count down)	Y	Y	OFF	-	-	R	NO	OFF
M1247	C247 counter monitor (on: count down)	Y	Y	OFF	-	-	R	NO	OFF
M1249	C249 counter monitor (on: count down)	Y	Y	OFF	-	-	R	NO	OFF
M1250	C250 counter monitor (on: count down) PH model only	-	Y	OFF	-	-	R	NO	OFF

Special M	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
M1251	C251 counter monitor (on: count down)	Y	Y	OFF	-	-	R	NO	OFF
M1252	C252 counter monitor (on: count down)	Y	Y	OFF	-	-	R	NO	OFF
M1254	C254 counter monitor (on: count down)	Y	Y	OFF	-	-	R	NO	OFF
M1260	Let X5 be the reset input signal of all high-speed counters	-	Y	OFF	-	-	R/W	NO	OFF
M1299	I150 flag disable	-	Y	OFF	-	-	R/W	NO	OFF
M1303	Switch flag of high and low bit	-	Y	OFF	-	-	R/W	NO	OFF
M1304	X input point can decide to be ON-OFF	-	Y	OFF	-	-	R/W	NO	OFF
M1350	ELC LINK start flag	-	Y	OFF	-	-	R/W	NO	OFF
M1351	Start ELC LINK automatically	-	Y	OFF	-	-	R/W	NO	OFF
M1352	Start ELC LINK manually	-	Y	OFF	-	-	R/W	NO	OFF
M1354	Start ELC LINK synchronously read/write	-	Y	OFF	-	-	R/W	NO	OFF
M1360	ELC LINK ID 1 exists	-	Y	OFF	-	-	R	NO	OFF
M1361	ELC LINK ID 2 exists	-	Y	OFF	-	-	R	NO	OFF
M1362	ELC LINK ID 3 exists	-	Y	OFF	-	-	R	NO	OFF
M1363	ELC LINK ID 4 exists	-	Y	OFF	-	-	R	NO	OFF
M1364	ELC LINK ID 5 exists	-	Y	OFF	-	-	R	NO	OFF
M1365	ELC LINK ID 6 exists	-	Y	OFF	-	-	R	NO	OFF
M1366	ELC LINK ID 7 exists	-	Y	OFF	-	-	R	NO	OFF
M1367	ELC LINK ID 8 exists	-	Y	OFF	-	-	R	NO	OFF
M1368	ELC LINK ID 9 exists	-	Y	OFF	-	-	R	NO	OFF
M1369	ELC LINK ID 10 exists	-	Y	OFF	-	-	R	NO	OFF
M1370	ELC LINK ID 11 exists	-	Y	OFF	-	-	R	NO	OFF
M1371	ELC LINK ID 12 exists	-	Y	OFF	-	-	R	NO	OFF
M1372	ELC LINK ID 13 exists	-	Y	OFF	-	-	R	NO	OFF
M1373	ELC LINK ID 14 exists	-	Y	OFF	-	-	R	NO	OFF
M1374	ELC LINK ID 15 exists	-	Y	OFF	-	-	R	NO	OFF
M1375	ELC LINK ID 16 exists	-	Y	OFF	-	-	R	NO	OFF
M1376	ELC LINK ID 1 acts	-	Y	OFF	-	-	R	NO	OFF
M1377	ELC LINK ID 2 acts	-	Y	OFF	-	-	R	NO	OFF
M1378	ELC LINK ID 3 acts	-	Y	OFF	-	-	R	NO	OFF
M1379	ELC LINK ID 4 acts	-	Y	OFF	-	-	R	NO	OFF
M1380	ELC LINK ID 5 acts	-	Y	OFF	-	-	R	NO	OFF

Special M	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
M1381	ELC LINK ID 6 acts	-	Y	OFF	-	-	R	NO	OFF
M1382	ELC LINK ID 7 acts	-	Y	OFF	-	-	R	NO	OFF
M1383	ELC LINK ID 8 acts	-	Y	OFF	-	-	R	NO	OFF
M1384	ELC LINK ID 9 acts	-	Y	OFF	-	-	R	NO	OFF
M1385	ELC LINK ID 10 acts	-	Y	OFF	-	-	R	NO	OFF
M1386	ELC LINK ID 11 acts	-	Y	OFF	-	-	R	NO	OFF
M1387	ELC LINK ID 12 acts	-	Y	OFF	-	-	R	NO	OFF
M1388	ELC LINK ID 13 acts	-	Y	OFF	-	-	R	NO	OFF
M1389	ELC LINK ID 14 acts	-	Y	OFF	-	-	R	NO	OFF
M1390	ELC LINK ID 15 acts	-	Y	OFF	-	-	R	NO	OFF
M1391	ELC LINK ID 16 acts	-	Y	OFF	-	-	R	NO	OFF
M1392	ELC LINK ID 1 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1393	ELC LINK ID 2 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1394	ELC LINK ID 3 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1395	ELC LINK ID 4 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1396	ELC LINK ID 5 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1397	ELC LINK ID 6 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1398	ELC LINK ID 7 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1399	ELC LINK ID 8 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1400	ELC LINK ID 9 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1401	ELC LINK ID 10 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1402	ELC LINK ID 11 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1403	ELC LINK ID 12 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1404	ELC LINK ID 13 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1405	ELC LINK ID 14 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1406	ELC LINK ID 15 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1407	ELC LINK ID 16 ERROR	-	Y	OFF	-	-	R	NO	OFF
M1408	ELC LINK ID 1 read completed	-	Y	OFF	-	-	R	NO	OFF
M1409	ELC LINK ID 2 read completed	-	Y	OFF	-	-	R	NO	OFF
M1410	ELC LINK ID 3 read completed	-	Y	OFF	-	-	R	NO	OFF
M1411	ELC LINK ID 4 read completed	-	Y	OFF	-	-	R	NO	OFF
M1412	ELC LINK ID 5 read completed	-	Y	OFF	-	-	R	NO	OFF

Special M	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
M1413	ELC LINK ID 6 read completed	-	Y	OFF	-	-	R	NO	OFF
M1414	ELC LINK ID 7 read completed	-	Y	OFF	-	-	R	NO	OFF
M1415	ELC LINK ID 8 read completed	-	Y	OFF	-	-	R	NO	OFF
M1416	ELC LINK ID 9 read completed	-	Y	OFF	-	-	R	NO	OFF
M1417	ELC LINK ID 10 read completed	-	Y	OFF	-	-	R	NO	OFF
M1418	ELC LINK ID 11 read completed	-	Y	OFF	-	-	R	NO	OFF
M1419	ELC LINK ID 12 read completed	-	Y	OFF	-	-	R	NO	OFF
M1420	ELC LINK ID 13 read completed	-	Y	OFF	-	-	R	NO	OFF
M1421	ELC LINK ID 14 read completed	-	Y	OFF	-	-	R	NO	OFF
M1422	ELC LINK ID 15 read completed	-	Y	OFF	-	-	R	NO	OFF
M1423	ELC LINK ID 16 read completed	-	Y	OFF	-	-	R	NO	OFF
M1424	ELC LINK ID 1 write completed	-	Y	OFF	-	-	R	NO	OFF
M1425	ELC LINK ID 2 write completed	-	Y	OFF	-	-	R	NO	OFF
M1426	ELC LINK ID 3 write completed	-	Y	OFF	-	-	R	NO	OFF
M1427	ELC LINK ID 4 write completed	-	Y	OFF	-	-	R	NO	OFF
M1428	ELC LINK ID 5 write completed	-	Y	OFF	-	-	R	NO	OFF
M1429	ELC LINK ID 6 write completed	-	Y	OFF	-	-	R	NO	OFF
M1430	ELC LINK ID 7 write completed	-	Y	OFF	-	-	R	NO	OFF
M1431	ELC LINK ID 8 write completed	-	Y	OFF	-	-	R	NO	OFF
M1432	ELC LINK ID 9 write completed	-	Y	OFF	-	-	R	NO	OFF
M1433	ELC LINK ID 10 write completed	-	Y	OFF	-	-	R	NO	OFF
M1434	ELC LINK ID 11 write completed	-	Y	OFF	-	-	R	NO	OFF
M1435	ELC LINK ID 12 write completed	-	Y	OFF	-	-	R	NO	OFF
M1436	ELC LINK ID 13 write completed	-	Y	OFF	-	-	R	NO	OFF
M1437	ELC LINK ID 14 write completed	-	Y	OFF	-	-	R	NO	OFF
M1438	ELC LINK ID 15 write completed	-	Y	OFF	-	-	R	NO	OFF
M1439	ELC LINK ID 16 write completed	-	Y	OFF	-	-	R	NO	OFF

5.8 S Relay

Initial step relay	Starting instruction in Sequential Function Chart (SFC).
Zero point return step relay	Zero point return when using IST instruction in program. If not used for the IST instruction, they can be used as general step relays.
General step relay	General relays in sequential function chart (SFC). They will be cleared when power loss after running.
Latched step relay	In sequential function chart (SFC), latched step relay will be saved when power loss after running. The state of power on after power loss will be the same as the state before power loss.
Step relay for alarm	The step relay for alarm uses with alarm drive instruction ANS to the contact for alarm. It is used to record warning and eliminate external malfunction.

5.9 T (Timer)

Timer resolution is 1ms, 10ms or 100ms depending on the selection of the timer number and the selection of Mxxxx relay. Timers always increment up and the output coil of the timer will be ON when the present value of the timer reaches the preset value. Timer's increment by one (1) whenever the base resolution is reached, meaning timers are truly counters that increment based on the timer resolution.

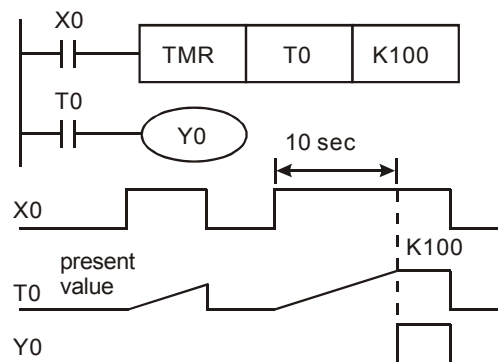
i.e. T200 set for 10ms resolution, present value = 10 (decimal)

= 10ms * 10 = 100ms time has elapsed.

General and accumulative timers function differently as described below.

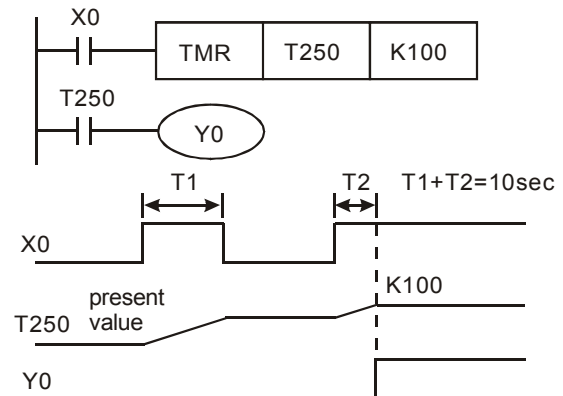
General Timer

Timer T0 will begin timing when X0=ON. If T0 has not reached its preset value by the time X0=OFF, then T0 will reset to zero (0). It will begin timing again when X0=ON.



Accumulative Timer

Timer T250 will begin timing when X0=ON. If T250 has not reached its preset value by the time X0=OFF, then T0 will pause. When X0=ON, T250 will resume timing from where it was paused.



Timers for Subroutines and Interrupts

Timers T192~T199 (PC/PA/PH Models) are the only timers that can be used in a subroutine or interrupt. Use of any other timer will not work.

5.10 C (Counter)

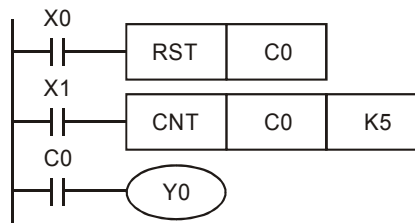
Counters will increment their present count value when the input circuit transitions from OFF→ON.

Item	16 bits counters	32 bits counters	
Type	General	General	High speed
Count direction	Count up	Count up/down	
Range	0~32,767	-2,147,483,648~+2,147,483,647	
Preset value register	Constant K or data register D (Word)	Constant K or data register D (Dword)	
Output operation	Counter will stop when preset value reached	Counter will keep on counting when preset value reached	
Output contact function	Ouptut Coil will be = ON when counter reaches preset value.	Output coil = ON when counter reaches or is above preset value. Output coil = OFF when counter is below preset value.	
Reset action	The present value will reset to 0 when RST instruction is executed, output coil = OFF.		
Update method	During every scan	During every scan	Immdiate – update is independent of scan time.

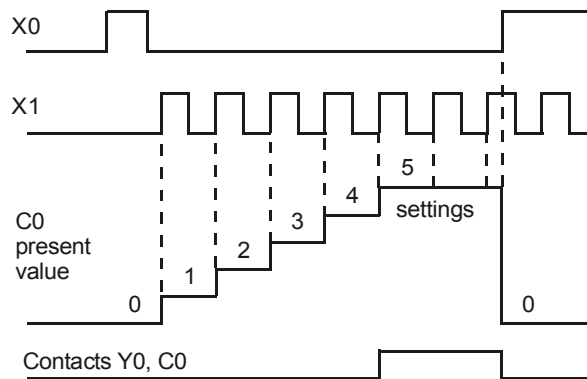
Example:

```

LD    X0
RST   C0
LD    X1
CNT   C0 K5
LD    C0
OUT   Y0
    
```



When X0=ON, RST instruction will reset C0. When X1 transitions OFF→ON, C0 will count up (add 1). When C0 reaches the preset value K5, C0 output coil Y0 will = ON and C0 will stop counting and ignore the X1 trigger signal.



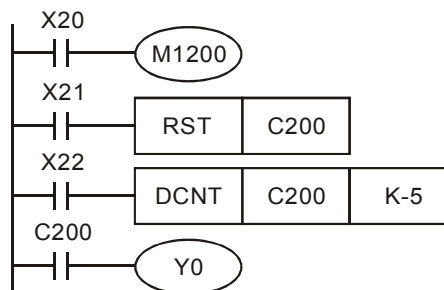
5

M relays M1200 – M1254 are used to set the up/down count direction for C200 – C254 respectively. Setting the corresponding M relay ON will set the counter to count down.

Example:

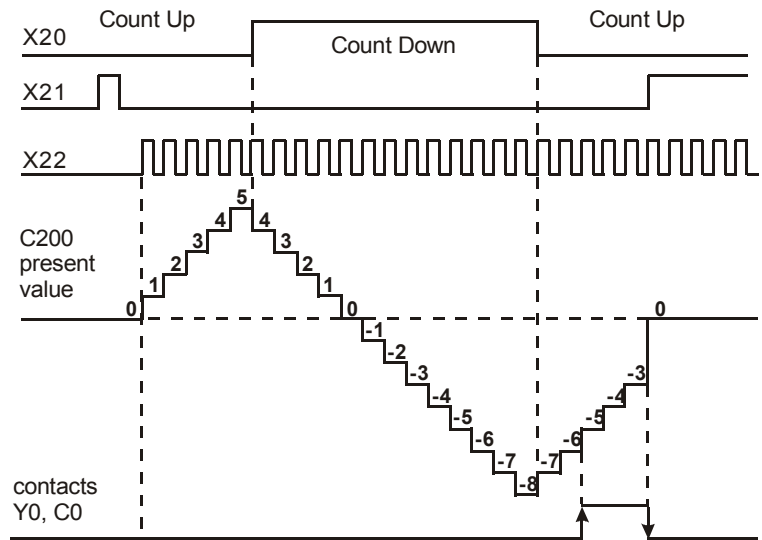
```

LD    X20
OUT   M1200
LD    X21
RST   C200
LD    X22
DCNT  C200 K-5
LD    C200
OUT   Y0
    
```



1. When X20 =ON, M1200=ON setting C200 to count down.
2. When X21 = ON, C200 will reset.
3. When X22 transitions from OFF→ON, C200 will increment or decrement (depending the M1200 ON/OFF) the present value.
4. When C200 transitions from K-6 to K-5, C200=ON.

5. When counter C200 is from K-5 to K-6, C200=OFF.



5.11 High-speed Counters

ELC High-speed counters can be 1-phase or 2 phase and can count up to a frequency of 20KHz. The table below displays the relation to inputs X0-X5, X10-X11 and counters C235-C254. (PB: X0-X3, PC/PA: X0-X5, PH: X0-X5, X10, X11)

U: Increasing A: A phase input S: Start input
D: Decreasing B: B phase input R: Clear input

	1-phase input										1-phase 2 inputs				2-phase inputs			
	C235	C236	C237	C238	C239	C240	C241	C242	C243	C244	C245	C246	C247	C249	C250	C251	C252	C254
X0	U/D						U/D			U/D		U	U	U		A	A	A
X1		U/D					R			R		D	D	D		B	B	B
X2			U/D					U/D					R	R			R	R
X3				U/D				R		S				S				S
X4					U/D													
X5						U/D												
X10									U/D						U			
X11											U/D				D			

Input X5 has two functions:

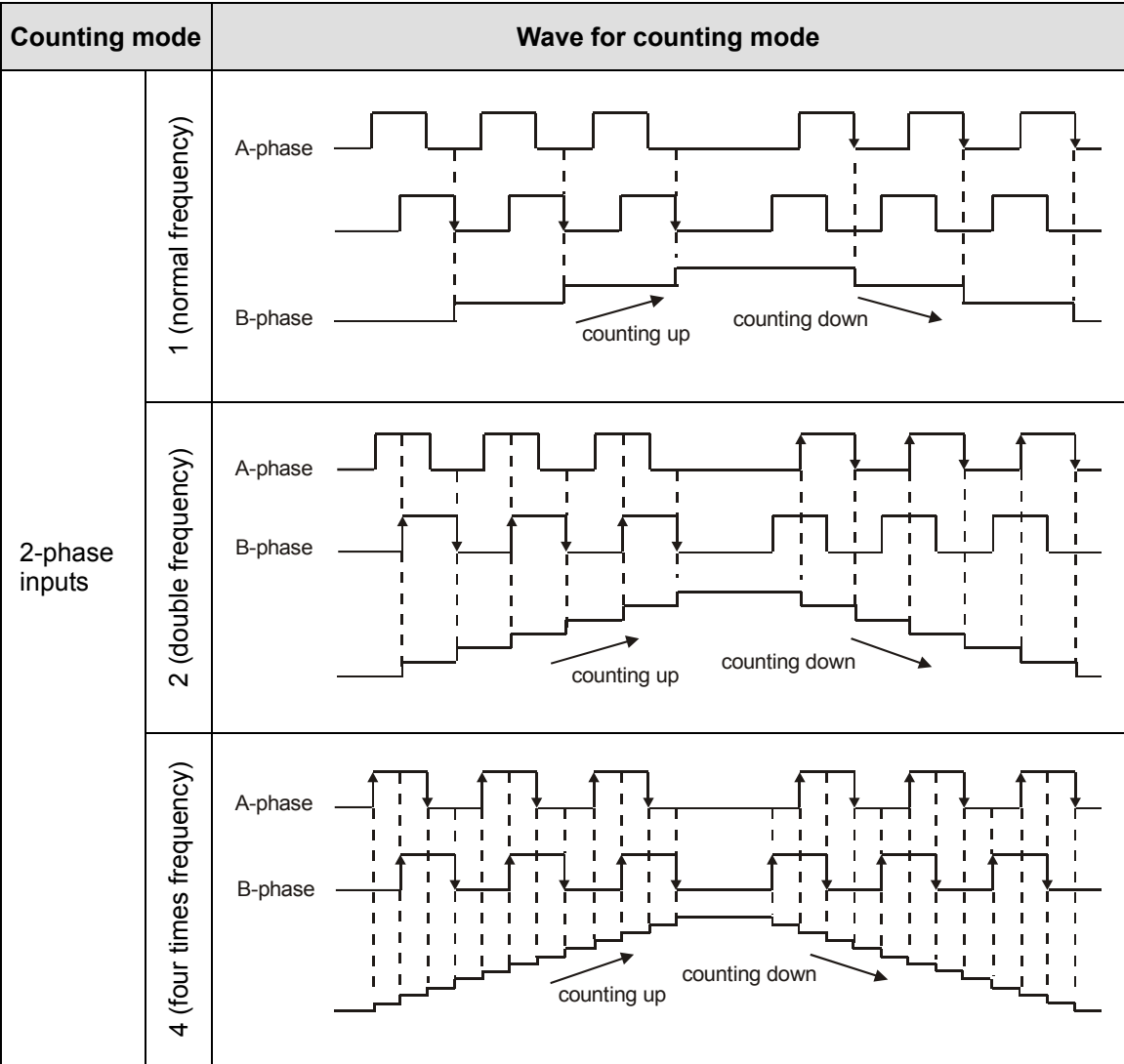
M1260=OFF C240 is general U/D high-speed counter.
M1260=ON it is Global reset for C235~C239.

Counting mode selection

The high-speed counter uses special D1022 in 2-phase inputs counting mode to select double frequency mode. D1022 content will be loaded in at the first scan time when ELC switches from STOP to RUN.

Device No.	Functions
D1022	Double frequency setting of counter counting method
D1022=K1	Normal frequency mode
D1022=K2	Double frequency mode (factory setting)
D1022=K4	Four times frequency mode

Double frequency mode (↑ , ↓ means the action of counting)



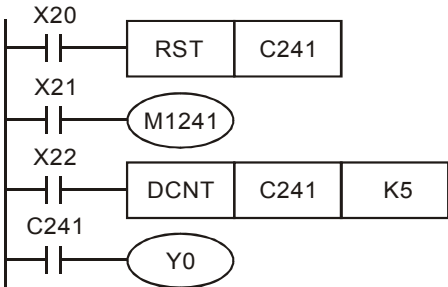
Device number and special registers for high-speed counter

Device number	Functions
M1235 ~ M1245	C235 ~ C245 are count direction of high-speed counters. When M12□□=OFF, C2□□ is count up. When M12□□=ON, C2□□ is count down.
M1246 ~ M1250 M1251 ~ M1254	C246 ~ C250, C251 ~ C254 are monitor count direction of high-speed counters. When C2□□ counts up, M12□□=OFF. When C2□□ count down, M12□□=ON.
D1022	Double frequency selection of AB phase counter

1-phase inputs high-speed counter:

Example:

LD X20
RST C241
LD X21
OUT M1241
LD X22
DCNT C241 K5
LD C241
OUT Y0

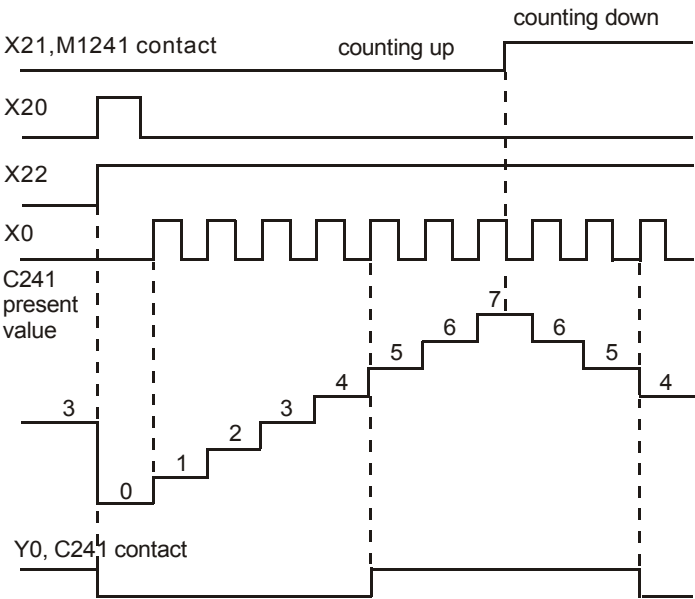


X21 drives M1241 to decide C241 is addition or subtraction.

When X20=ON and RST instruction is executed, clear C241 to 0 and reset output contact to off.

When X22=ON, C241 receives count signal from X0 and counter will count up (+1) or count down (-1).

When counter C241 attains settings K5, C241 will be ON. If there is still signal input for X0, it will keep on counting.

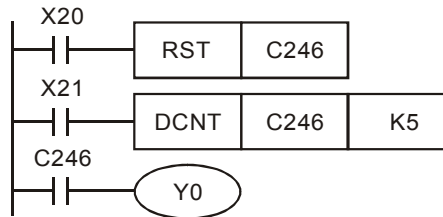


1-phase 2 inputs high-speed counters:**Example:**

```

LD      X20
RST     C246
LD      X21
DCNT    C246 K5
LD      C246
OUT     Y0

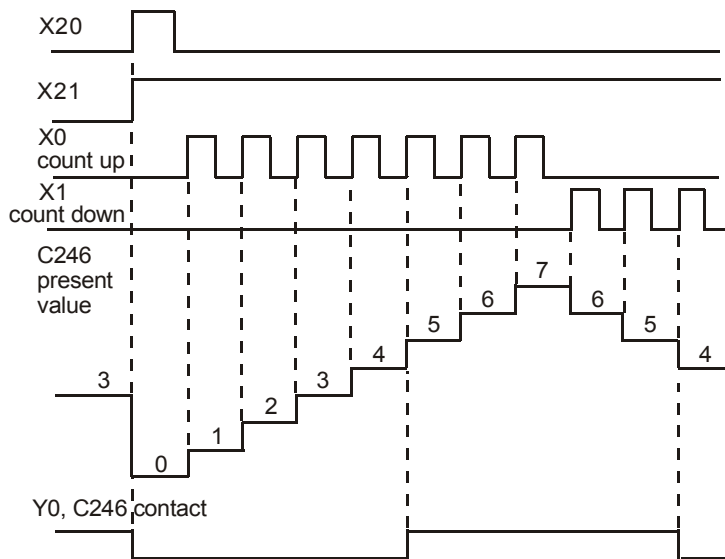
```



When X20=ON and RST instruction is executed, clear C246 to 0 and reset output contact to off.

When X21=ON, C246 receives count signal from X0 input terminal and counter will count up (+1) or receive count signal from X1 input terminal and counter will count down (-1).

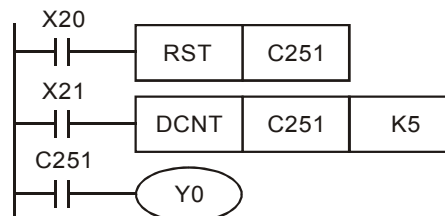
When C246 attains settings K5, C246 will be on. After C246 is ON, if there is counter pulse input, C246 will keep on counting.

**2-phase AB input high-speed counter:****Example:**

```

LD      X20
RST     C251
LD      X21
DCNT    C251 K5
LD      C251
OUT     Y0

```



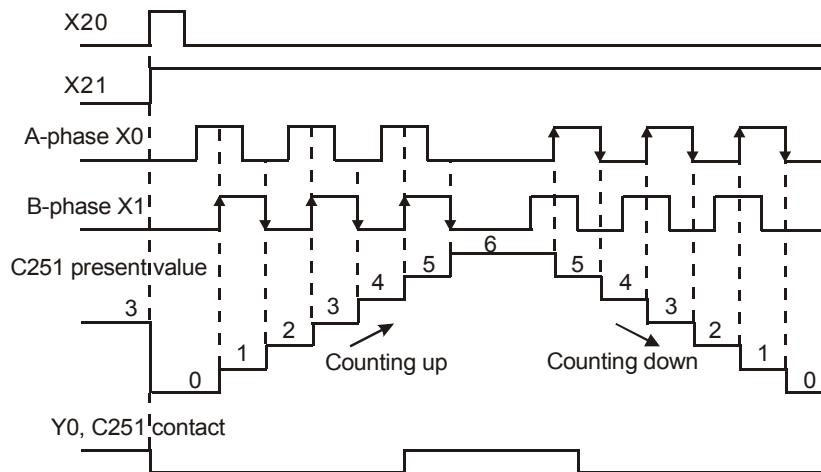
When X20=ON, RST instruction is executed and resets C251 to 0, output contact is reset to off.

C251 receives A phase counting signal of X0 input terminal and B phase counting signal of X1 input terminal to execute add 1 (count up) or subtract 1 (count down) when X21=on.

When counter C251 attains settings K5, C251 contact will be ON. After C251 is ON, if there is counter pulse input, C251 will keep on counting.

Frequency can be set to normal, double frequency or four times frequency by D1022 (counting mode setting). Factory setting is double frequency.

C251 has external input reset signal X5.



5.12 Special Data Register

The special registers (special D) are as shown in the following. Please notice that some equipments with the same number will be different to the different model. In the following chart, the meaning of column “Attribute” are: “R” means can only read. “R/W” means can read/write. “-” means can do nothing. “#” means system setting, user can read the detail explanation of the setting in the manual.

Special D	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
D1000	Watchdog timer (WDT) value (Unit: 1ms)	Y	Y	200	-	-	R/W	NO	200
D1001	ELC model number+memory capacity / type (user can read ELC program version from this register. For example, D1001 = H XX10 means version 1.0. When reading from HHP it will display Knnnnn and you can convert it to hexadecimal number by pressing <H> key.	Y	Y	-	-	-	R	NO	#
D1002	Program capacity	Y	Y	-	-	-	R	NO	#
D1003	Sum of program memory (sum of the ELC internal program memory. User can identify the content of ELC control program by this register)	Y	Y	-	-	-	R	NO	#
D1004	Grammar detective number	Y	Y	0	0	-	R	NO	0
D1008	STEP address when WDT timer is ON	Y	Y	0	-	-	R	NO	0
D1010	Present scan time (Unit: 0.1ms)	Y	Y	0	0	0	R	NO	0
D1011	Minimum scan time (Unit: 0.1ms)	Y	Y	0	0	0	R	NO	0
D1012	Maximum scan time (Unit: 0.1ms)	Y	Y	0	0	0	R	NO	0
D1015	0~32,767(unit: 0.1ms) addition type of high-speed connection timer	-	Y	0	-	-	R/W	NO	0
D1018	π PI (Low byte)	-	Y	0FDB	0FDB	0FDB	R/W	NO	0FDB
D1019	π PI(High byte)	-	Y	4049	4049	4049	R/W	NO	4049
D1020	X0~X7 input filter (unit: ms) 0~1,000ms adjustable	Y	Y	10	-	-	R/W	NO	10
D1021	X10~X17 input delay times setting (unit: times)	-	Y	0	-	-	R/W	NO	0
D1022	Double frequency selection for AB phase counter	Y	Y	0	-	-	R/W	NO	0
D1025	Communication error code	Y	Y	0	-	-	R	NO	0
D1028	Index register E0	Y	Y	0	-	-	R/W	NO	0
D1029	Index register F0	Y	Y	0	-	-	R/W	NO	0
D1030	Output numbers of Y0 pulse (Low word)	Y	Y	0	-	-	R	NO	0
D1031	Output numbers of Y0 pulse (High word)	Y	Y	0	-	-	R	NO	0
D1032	Output numbers of Y1 pulse (Low word)	Y	Y	0	-	-	R	NO	0
D1033	Output numbers of Y1 pulse (High word)	Y	Y	0	-	-	R	NO	0

Special D	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
D1036	COM1 (232) Communications protocol	Y	Y	0086	-	-	R/W	NO	0086
D1038	When ELC is master, this sets the data response delay time. Time unit is 0.1ms.	Y	Y	-	-	-	R/W	NO	0
D1039	Constant scan time (ms)	Y	Y	0	-	-	R/W	NO	0
D1040	ON state number 1 of STEP point S	Y	Y	0	-	-	R	NO	0
D1041	ON state number 2 of STEP point S	Y	Y	0	-	-	R	NO	0
D1042	ON state number 3 of STEP point S	Y	Y	0	-	-	R	NO	0
D1043	ON state number 4 of STEP point S	Y	Y	0	-	-	R	NO	0
D1044	ON state number 5 of STEP point S	Y	Y	0	-	-	R	NO	0
D1045	ON state number 6 of STEP point S	Y	Y	0	-	-	R	NO	0
D1046	ON state number 7 of STEP point S	Y	Y	0	-	-	R	NO	0
D1047	ON state number 8 of STEP point S	Y	Y	0	-	-	R	NO	0
D1049	ON number of alarm point	-	Y	0	-	-	R	NO	0
D1050 ↓ D1055	ELC will automatically convert the ASCII data saved in D1070~D1085 to HEX.	Y	Y	0	-	-	R	NO	0
D1056	Present value of PA controller analog input channel 0 (CH0)	-	Y	0	-	-	R	NO	0
D1057	Present value of PA controller analog input channel 1 (CH1)	-	Y	0	-	-	R	NO	0
D1067	Algorithm error code	Y	Y	0	0	-	R	NO	0
D1068	Lock the algorithm error address	Y	Y	0	-	-	R	NO	0
D1069	Step number of errors associated with flags M1065~M1067	Y	Y	0	-	-	R	NO	0
D1070 ↓ D1085	When the ELC's built-in RS-485 communication instruction receives feedback signals from receiver. The data will be saved in the registers D1070~D1085. User can use the contents saved in the registers to check the feedback data.	Y	Y	0	-	-	R	NO	0
D1089 ↓ D1099	When the ELC built-in RS-485 communication instruction is executed, the transmitting signals will be stored in the registers D1089~D1099. User can use the contents saved in registers to check the feedback data.	Y	Y	0	-	-	R	NO	0
D1100	System used	-	-	-	-	-	-	-	
D1101	Start address of file register	-	Y	-	-	-	R/W	Yes	0
D1102	Copy numbers of file register	-	Y	-	-	-	R/W	Yes	1600
D1103	Set start D number for file register to store (the number should be large than 2000)	-	Y	-	-	-	R/W	Yes	2000
D1104	Parameter index for Accel/Decel pulse output Y0 (corresponds to device D)	Y	Y	0	0	-	R/W	NO	0

Special D	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
D1110	Average of PA controller analog input channel 0 (CH 0)	-	Y	0	-	-	R	NO	0
D1111	Average of PA controller analog input channel 1 (CH 1)	-	Y	0	-	-	R	NO	0
D1116	PA controller analog output channel 0 (CH 0)	-	Y	0	0	0	R/W	NO	0
D1117	PA controller analog output channel 1 (CH 1)	-	Y	0	0	0	R/W	NO	0
D1118	PA controller only. It is the filter wave time setting between the A/D conversions, and with the default setting as 0 and the unit as 1ms, all will be regarded as 5ms if D1118≤5.	-	Y	5	-	-	R/W	NO	5
D1120	COM2 (RS-485) communication protocol	Y	Y	0086	-	-	R/W	NO	0086
D1121	ELC communication address (the address that save ELC communication address, it is latched)	Y	Y	-	-	-	R/W	Yes	1
D1122	Residual words of transmitting data	Y	Y	0	0	-	R	NO	0
D1123	Residual words of receiving data	Y	Y	0	0	-	R	NO	0
D1124	Start character definition (STX)	Y	Y	003A	-	-	R/W	NO	003A
D1125	First ending character definition (ETX1)	Y	Y	000D	-	-	R/W	NO	000D
D1126	Second ending character definition (ETX2)	Y	Y	000A	-	-	R/W	NO	000A
D1129	RS-485 time-out setting (ms)	Y	Y	0	-	-	R/W	NO	0
D1130	MODBUS return error code record	Y	Y	0	-	-	R	NO	0
D1133	Special high-speed pulse output Y0 (50KHz) register (D) index	-	Y	0	-	-	R/W	NO	0
D1137	Address of operator error occurs	Y	Y	0	0	-	R	NO	0
D1140	Special extension module number, maximum is 8 units	Y	Y	0	-	-	R	NO	0
D1142	Input points (X) of extension unit	Y	Y	0	-	-	R	NO	0
D1143	Output points (Y) of extension unit	Y	Y	0	-	-	R	NO	0
D1144	Parameter index for Accel/Decel pulse output Y0 of adjustable slope (corresponds to component D)	-	Y	0	-	-	R/W	NO	0
D1154	Recommended Interval of accelerated time (10~32767 ms) of Accel/Decel pulse output Y0 of adjustable slope	-	Y	200	-	-	R/W	NO	200
D1155	Recommended Interval of decelerated time (-1~ -32700 ms) of Accel/Decel pulse output Y0 of adjustable slope	-	Y	-1000	-	-	R/W	NO	-1000
D1168	RS instruction, interrupt request when receiving specified data character (I150)	Y	Y	0	-	-	R/W	NO	0
D1172	2-phase pulse output frequency (12Hz~20KHz)	-	Y	0	-	-	R/W	NO	0
D1173	2-phase pulse output mode selection (K1 and K2)	-	Y	0	-	-	R/W	NO	0

Special D	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
D1174	Target number for 2-phase pulse outputs (low 16-bit)	-	Y	0	-	-	R/W	NO	0
D1175	Target number for 2-phase pulse outputs (high 16-bit)	-	Y	0	-	-	R/W	NO	0
D1176	Present output number of 2-phase pulse (low 16-bit)	-	Y	0	-	-	R/W	NO	0
D1177	Present output number of 2-phase pulse (high 16-bit)	-	Y	0	-	-	R/W	NO	0
D1178	VR0 Variable resistor value	-	Y	0	-	-	R	NO	0
D1179	VR1 Variable resistor value	-	Y	0	-	-	R	NO	0
D1182	Pointer register E1	-	Y	0	-	-	R/W	NO	0
D1183	Pointer register F1	-	Y	0	-	-	R/W	NO	0
D1184	Pointer register E2	-	Y	0	-	-	R/W	NO	0
D1185	Pointer register F2	-	Y	0	-	-	R/W	NO	0
D1186	Pointer register E3	-	Y	0	-	-	R/W	NO	0
D1187	Pointer register F3	-	Y	0	-	-	R/W	NO	0
D1196	Number system setting for PA model 7-seg display ON=hex, OFF=decimal	-	Y	0	-	-	R/W	NO	0
D1200	Start address of M0~M999 auxiliary relay latched	-	Y	-	-	-	R/W	Yes	#
D1201	End address of M0~M999 auxiliary relay latched	-	Y	-	-	-	R/W	Yes	999
D1202	Start address of M2000~M4095 auxiliary relay latched	-	Y	-	-	-	R/W	Yes	2000
D1203	End address of M2000~M4095 auxiliary relay latched	-	Y	-	-	-	R/W	Yes	4095
D1208	Start latched address of 16-bit counter C0~C199	-	Y	-	-	-	R/W	Yes	#
D1209	End latched address of 16-bit counter C0~C199	-	Y	-	-	-	R/W	Yes	199
D1210	Start latched address of 32-bit counter C200~C234	-	Y	-	-	-	R/W	Yes	#
D1211	End latched address of 32-bit counter C200~C234	-	Y	-	-	-	R/W	Yes	234
D1212	Start latched address of 32-bit high-speed counter C235~C255	-	Y	-	-	-	R/W	Yes	235
D1213	End latched address of 32-bit high-speed counter C235~C255	-	Y	-	-	-	R/W	Yes	255
D1214	Start latched address of step point (S0~S1023)	-	Y	-	-	-	R/W	Yes	#
D1215	End latched address of step point (S0~S1023)	-	Y	-	-	-	R/W	Yes	#
D1216	Start latched address of register D0~D999	-	Y	-	-	-	R/W	Yes	200
D1217	End latched address of register D0~D999	-	Y	-	-	-	R/W	Yes	999

Special D	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
D1218	Start latched address of register D2000~D9999	-	Y	-	-	-	R/W	Yes	2000
D1219	End latched address of register D2000~D9999	-	Y	-	-	-	R/W	Yes	#
D1200	System used	-	-	-	-	-	-	-	-
D1256 ↓ D1295	MODRW instruction of RS-485 is built-in. The characters that sent during executing is saved in D1256~D1295. User can check according to the content of these registers.	Y	Y	0	-	-	R	NO	0
D1296 ↓ D1311	MODRW instruction of RS-485 is built-in. ELC system will convert ASCII in the content of the register that user indicates to HEX and save it in D1296 – D1311.	Y	Y	0	-	-	R	NO	0
D1313	Perpetual calendar (RTC) second 00~59	-	Y	0	-	-	R/W	NO	0
D1314	Perpetual calendar (RTC) minute 00~59	-	Y	0	-	-	R/W	NO	0
D1315	Perpetual calendar (RTC) hour 00~23	-	Y	0	-	-	R/W	NO	0
D1316	Perpetual calendar (RTC) day 01~31	-	Y	0	-	-	R/W	NO	1
D1317	Perpetual calendar (RTC) month 01~12	-	Y	0	-	-	R/W	NO	1
D1318	Perpetual calendar (RTC) week 1~7	-	Y	0	-	-	R/W	NO	6
D1319	Perpetual calendar (RTC) year 00 – 99	-	Y	0	-	-	R/W	NO	0
D1340	The 1 st step acceleration frequency of CH0	-	Y	-	-	-	R/W	NO	200
D1343	Acceleration /Deceleration time of CH0	-	Y	-	-	-	R/W	NO	200
D1348	Present value of CH0 pulse (low word) Y10	-	Y	0	-	-	R	NO	0
D1349	Present value of CH0 pulse (high word) Y10	-	Y	0	-	-	R	NO	0
D1350	Present value of CH1 pulse (low word) Y11	-	Y	0	-	-	R	NO	0
D1351	Present value of CH1 pulse (high word) Y11	-	Y	0	-	-	R	NO	0
D1352	The 1 st step start frequency and the step end frequency of CH1	-	Y	-	-	-	R/W	NO	200
D1353	Acceleration/Deceleration time of CH1 pulse	-	Y	-	-	-	R/W	NO	200
D1355	Communication address that read by ELC LINK ID 1	-	Y	1064	-	-	R/W	NO	1064
D1356	Communication address that read by ELC LINK ID 2	-	Y	1064	-	-	R/W	NO	1064
D1357	Communication address that read by ELC LINK ID 3	-	Y	1064	-	-	R/W	NO	1064
D1358	Communication address that read by ELC LINK ID 4	-	Y	1064	-	-	R/W	NO	1064
D1359	Communication address that read by ELC LINK ID 5	-	Y	1064	-	-	R/W	NO	1064
D1360	Communication address that read by ELC LINK ID 6	-	Y	1064	-	-	R/W	NO	1064

Special D	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
D1361	Communication address that read by ELC LINK ID 7	-	Y	1064	-	-	R/W	NO	1064
D1362	Communication address that read by ELC LINK ID 8	-	Y	1064	-	-	R/W	NO	1064
D1363	Communication address that read by ELC LINK ID 9	-	Y	1064	-	-	R/W	NO	1064
D1364	Communication address that read by ELC LINK ID 10	-	Y	1064	-	-	R/W	NO	1064
D1365	Communication address that read by ELC LINK ID 11	-	Y	1064	-	-	R/W	NO	1064
D1366	Communication address that read by ELC LINK ID 12	-	Y	1064	-	-	R/W	NO	1064
D1367	Communication address that read by ELC LINK ID 13	-	Y	1064	-	-	R/W	NO	1064
D1368	Communication address that read by ELC LINK ID 14	-	Y	1064	-	-	R/W	NO	1064
D1369	Communication address that read by ELC LINK ID 15	-	Y	1064	-	-	R/W	NO	1064
D1370	Communication address that read by ELC LINK ID 16	-	Y	1064	-	-	R/W	NO	1064
D1399	ID number of the strating slave that specified by ELC LINK	-	Y	1	-	-	R/W	YES	1
D1415	Communication address that wrote by ELC LINK ID 1	-	Y	10C8	-	-	R/W	NO	10C8
D1416	Communication address that wrote by ELC LINK ID 2	-	Y	10C8	-	-	R/W	NO	10C8
D1417	Communication address that wrote by ELC LINK ID 3	-	Y	10C8	-	-	R/W	NO	10C8
D1418	Communication address that wrote by ELC LINK ID 4	-	Y	10C8	-	-	R/W	NO	10C8
D1419	Communication address that wrote by ELC LINK ID 5	-	Y	10C8	-	-	R/W	NO	10C8
D1420	Communication address that wrote by ELC LINK ID 6	-	Y	10C8	-	-	R/W	NO	10C8
D1421	Communication address that wrote by ELC LINK ID 7	-	Y	10C8	-	-	R/W	NO	10C8
D1422	Communication address that wrote by ELC LINK ID 8	-	Y	10C8	-	-	R/W	NO	10C8
D1423	Communication address that wrote by ELC LINK ID 9	-	Y	10C8	-	-	R/W	NO	10C8
D1424	Communication address that wrote by ELC LINK ID 10	-	Y	10C8	-	-	R/W	NO	10C8
D1425	Communication address that wrote by ELC LINK ID 11	-	Y	10C8	-	-	R/W	NO	10C8
D1426	Communication address that wrote by ELC LINK ID 12	-	Y	10C8	-	-	R/W	NO	10C8

Special D	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
D1427	Communication address that wrote by ELC LINK ID 13	-	Y	10C8	-	-	R/W	NO	10C8
D1428	Communication address that wrote by ELC LINK ID 14	-	Y	10C8	-	-	R/W	NO	10C8
D1429	Communication address that wrote by ELC LINK ID 15	-	Y	10C8	-	-	R/W	NO	10C8
D1430	Communication address that wrote by ELC LINK ID 16	-	Y	10C8	-	-	R/W	NO	10C8
D1431	ELC LINK times	-	Y	0	-	-	R/W	NO	0
D1432	ELC LINK counts	-	Y	0	-	-	R/W	NO	0
D1433	ELC LINK units	-	Y	0	-	-	R/W	NO	0
D1434	Read items of ELC LINK ID 1	-	Y	16	-	-	R/W	NO	16
D1435	Read items of ELC LINK ID 2	-	Y	16	-	-	R/W	NO	16
D1436	Read items of ELC LINK ID 3	-	Y	16	-	-	R/W	NO	16
D1437	Read items of ELC LINK ID 4	-	Y	16	-	-	R/W	NO	16
D1438	Read items of ELC LINK ID 5	-	Y	16	-	-	R/W	NO	16
D1439	Read items of ELC LINK ID 6	-	Y	16	-	-	R/W	NO	16
D1440	Read items of ELC LINK ID 7	-	Y	16	-	-	R/W	NO	16
D1441	Read items of ELC LINK ID 8	-	Y	16	-	-	R/W	NO	16
D1442	Read items of ELC LINK ID 9	-	Y	16	-	-	R/W	NO	16
D1443	Read items of ELC LINK ID 10	-	Y	16	-	-	R/W	NO	16
D1444	Read items of ELC LINK ID 11	-	Y	16	-	-	R/W	NO	16
D1445	Read items of ELC LINK ID 12	-	Y	16	-	-	R/W	NO	16
D1446	Read items of ELC LINK ID 13	-	Y	16	-	-	R/W	NO	16
D1447	Read items of ELC LINK ID 14	-	Y	16	-	-	R/W	NO	16
D1448	Read items of ELC LINK ID 15	-	Y	16	-	-	R/W	NO	16
D1449	Read items of ELC LINK ID 16	-	Y	16	-	-	R/W	NO	16
D1450	Wrote items of ELC LINK ID 1	-	Y	16	-	-	R/W	NO	16
D1451	Wrote items of ELC LINK ID 2	-	Y	16	-	-	R/W	NO	16
D1452	Wrote items of ELC LINK ID 3	-	Y	16	-	-	R/W	NO	16
D1453	Wrote items of ELC LINK ID 4	-	Y	16	-	-	R/W	NO	16
D1454	Wrote items of ELC LINK ID 5	-	Y	16	-	-	R/W	NO	16
D1455	Wrote items of ELC LINK ID 6	-	Y	16	-	-	R/W	NO	16
D1456	Wrote items of ELC LINK ID 7	-	Y	16	-	-	R/W	NO	16
D1457	Wrote items of ELC LINK ID 8	-	Y	16	-	-	R/W	NO	16

Special D	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
D1458	Wrote items of ELC LINK ID 9	-	Y	16	-	-	R/W	NO	16
D1459	Wrote items of ELC LINK ID 10	-	Y	16	-	-	R/W	NO	16
D1460	Wrote items of ELC LINK ID 11	-	Y	16	-	-	R/W	NO	16
D1461	Wrote items of ELC LINK ID 12	-	Y	16	-	-	R/W	NO	16
D1462	Wrote items of ELC LINK ID 13	-	Y	16	-	-	R/W	NO	16
D1463	Wrote items of ELC LINK ID 14	-	Y	16	-	-	R/W	NO	16
D1464	Wrote items of ELC LINK ID 15	-	Y	16	-	-	R/W	NO	16
D1465	Wrote items of ELC LINK ID 16	-	Y	16	-	-	R/W	NO	16
D1466	System used	-	-	-	-	-	-	-	-
D1480 ↓ D1495	ID 1 LINK ELC reads. Communication address for ID 1 reads is in D1355. The range is D100-D115 of ID 1 ELC.	-	Y	0	-	-	R	NO	0
D1496 ↓ D1511	ID 1 LINK ELC writes. Communication address for ID 1 writes is in D1415. The range is D200-D215 of ID 1 ELC.	-	Y	0	-	-	R/W	NO	0
D1512 ↓ D1527	ID 2 LINK ELC reads. Communication address for ID 2 reads is in D1356. The range is D100-D115 of ID 2 ELC.	-	Y	0	-	-	R	NO	0
D1528 ↓ D1543	ID 2 LINK ELC writes. Communication address for ID 2 writes is in D1416. The range is D200-D215 of ID 2 ELC.	-	Y	0	-	-	R/W	NO	0
D1544 ↓ D1559	ID 3 LINK ELC reads. Communication address for ID 3 reads is in D1357. The range is D100-D115 of ID 3 ELC.	-	Y	0	-	-	R	NO	0
D1560 ↓ D1575	ID 3 LINK ELC writes. Communication address for ID 3 writes is in D1417. The range is D200-D215 of ID 3 ELC.	-	Y	0	-	-	R/W	NO	0
D1576 ↓ D1591	ID 4 LINK ELC reads. Communication address for ID 4 reads is in D1358. The range is D100-D115 of ID 4 ELC.	-	Y	0	-	-	R	NO	0
D1592 ↓ D1607	ID 4 LINK ELC writes. Communication address for ID 4 writes is in D1418. The range is D200-D215 of ID 4 ELC.	-	Y	0	-	-	R/W	NO	0
D1608 ↓ D1623	ID 5 LINK ELC reads. Communication address for ID 5 reads is in D1359. The range is D100-D115 of ID 5 ELC.	-	Y	0	-	-	R	NO	0
D1624 ↓ D1639	ID 5 LINK ELC writes. Communication address for ID 5 writes is in D1419. The range is D200-D215 of ID 5 ELC.	-	Y	0	-	-	R/W	NO	0
D1640 ↓ D1655	ID 6 LINK ELC reads. Communication address for ID 6 reads is in D1360. The range is D100-D115 of ID 6 ELC.	-	Y	0	-	-	R	NO	0

Special D	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
D1656 ↓ D1671	ID 6 LINK ELC writes. Communication address for ID 6 writes is in D1420. The range is D200-D215 of ID 6 ELC.	-	Y	0	-	-	R/W	NO	0
D1672 ↓ D1687	ID 7 LINK ELC reads. Communication address for ID 7 reads is in D1361. The range is D100-D115 of ID 7 ELC.	-	Y	0	-	-	R	NO	0
D1688 ↓ D1703	ID 7 LINK ELC writes. Communication address for ID 7 writes is in D1421. The range is D200-D215 of ID 7 ELC.	-	Y	0	-	-	R/W	NO	0
D1704 ↓ D1719	ID 8 LINK ELC reads. Communication address for ID 8 reads is in D1362. The range is D100-D115 of ID 8 ELC.	-	Y	0	-	-	R	NO	0
D1720 ↓ D1735	ID 8 LINK ELC writes. Communication address for ID 8 writes is in D1422. The range is D200-D215 of ID 8 ELC.	-	Y	0	-	-	R/W	NO	0
D1736 ↓ D1751	ID 9 LINK ELC reads. Communication address for ID 9 reads is in D1363. The range is D100-D115 of ID 9 ELC.	-	Y	0	-	-	R	NO	0
D1752 ↓ D1767	ID 9 LINK ELC writes. Communication address for ID 9 writes is in D1423. The range is D200-D215 of ID 9 ELC.	-	Y	0	-	-	R/W	NO	0
D1768 ↓ D1783	ID 10 LINK ELC reads. Communication address for ID 10 reads is in D1364. The range is D100-D115 of ID 10 ELC.	-	Y	0	-	-	R	NO	0
D1784 ↓ D1799	ID 10 LINK ELC writes. Communication address for ID 10 writes is in D1424. The range is D200-D215 of ID 10 ELC.	-	Y	0	-	-	R/W	NO	0
D1800 ↓ D1815	ID 11 LINK ELC reads. Communication address for ID 11 reads is in D1365. The range is D100-D115 of ID 11 ELC.	-	Y	0	-	-	R	NO	0
D1816 ↓ D1831	ID 11 LINK ELC writes. Communication address for ID 11 writes is in D1425. The range is D200-D215 of ID 11 ELC.	-	Y	0	-	-	R/W	NO	0
D1832 ↓ D1847	ID 12 LINK ELC reads. Communication address for ID 12 reads is in D1366. The range is D100-D115 of ID 12 ELC.	-	Y	0	-	-	R	NO	0
D1848 ↓ D1863	ID 12 LINK ELC writes. Communication address for ID 12 writes is in D1426. The range is D200-D215 of ID 12 ELC.	-	Y	0	-	-	R/W	NO	0
D1864 ↓ D1879	ID 13 LINK ELC reads. Communication address for ID 13 reads is in D1367. The range is D100-D115 of ID 13 ELC.	-	Y	0	-	-	R	NO	0
D1880 ↓ D1895	ID 13 LINK ELC writes. Communication address for ID 13 writes is in D1427. The range is D200-D215 of ID 13 ELC.	-	Y	0	-	-	R/W	NO	0
D1896 ↓ D1911	ID 14 LINK ELC reads. Communication address for ID 14 reads is in D1368. The range is D100-D115 of ID 14 ELC.	-	Y	0	-	-	R	NO	0

Special D	Function	PB	PC PA PH	OFF ↓ ON	STOP ↓ RUN	RUN ↓ STOP	Type	Latched	Factory setting
D1912 ↓ D1927	ID 14 LINK ELC writes. Communication address for ID 14 writes is in D1428. The range is D200-D215 of ID 14 ELC.	-	Y	0	-	-	R/W	NO	0
D1928 ↓ D1943	ID 15 LINK ELC reads. Communication address for ID 15 reads is in D1369. The range is D100-D115 of ID 15 ELC.	-	Y	0	-	-	R	NO	0
D1944 ↓ D1959	ID 15 LINK ELC writes. Communication address for ID 15 writes is in D1429. The range is D200-D215 of ID 15 ELC.	-	Y	0	-	-	R/W	NO	0
D1960 ↓ D1975	ID 16 LINK ELC reads. Communication address for ID 16 reads is in D1370. The range is D100-D115 of ID 16 ELC.	-	Y	0	-	-	R	NO	0
D1976 ↓ D1991	ID 16 LINK ELC writes. Communication address for ID 16 writes is in D1430. The range is D200-D215 of ID 16 ELC.	-	Y	0	-	-	R/W	NO	0

5.13 E, F Index Registers

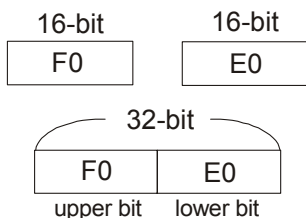
The function of Index register is the same as general operand. It can be used to move, compare or used as an index for byte device (KnX, KnY, KnM, KnS, T, C, D) and bit device (X, Y, M, S). It can't be used for constant (K, H).

Index register [E], [F]

Index registers are 16-bit registers. There are 2 points, E0 and F0, for PB models. There are 8 points, E0~E3 and F0~F3, for PC/PA/PH models. If you want to use index register to be 32-bit register, you should indicate E and at this moment F can't be used.

Example: "MOV K10 D0F0"

Index registers E, F are 16-bit data register, just the same as general data register. They are read/ write. They can be used as a 32-bit register.

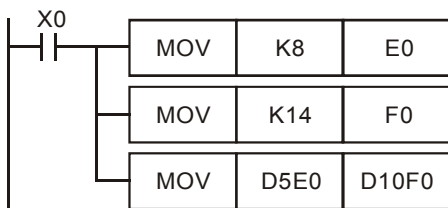


It is recommended to use instruction DMOVP K0 E and clear E and F to 0 at power on.

The combination of E and F when using as 32-bit register are:

(E0, F0), (E1, F1) (E2, F2) (E3, F3)

When X0=ON and E0=8, F0=14, D5E0=D(5+8)=D13, D10F0 =D(10+14) = D24, the content in D13 will be moved to D24.



5.14 File Register

There are 1600 file registers (K0~K1599). There is no device number (name) for file register; to execute read/writes of file register use instruction MEMR or MEMW, peripheral device HHP or ELCSoft.

ELC will check the following when ELC is powered on or change from POWER OFF→ON.

1. M1101 - starts file register function
2. D1101 - starting address of file register, K0~K1599

3. D1102 – number of item to read/write, K1~ K1600
4. D1103 - starting address of file register D register, D2000~ D4999

Note:

Reading from file register to data register D won't be executed when D1101 is greater than 1600.
When starting the action to read data from the file register to the data register, ELC will stop executing once the address of file register or data register D exceeds the viable address range.

5.15 Nest Level Pointer[N], Pointer[P], Interrupt Pointer [I]

Pointer	N	Master control nested	N0~N7, 8 points	The control point of master control nested
	P	For CJ, CALL instructions	PB Model = P0~P63 64 points PC/PA/PH Models = P0~P255, 256 points	The location point of CJ, CALL
	I	For interrupt	Insert external interrupt	The location point of interrupt subroutine.
			Insert time interrupt	
			Insert high-speed counter attained interrupt	
			Insert communication interrupt	

Nest Level Pointer N: used with instruction MC and MCR. MC is master start instruction. When the MC instruction is executed, the instructions between MC and MCR will be executed normally. MC-MCR master instruction supports nested program structure and the maximum is 8 levels, which is numbered from N0 to N7.

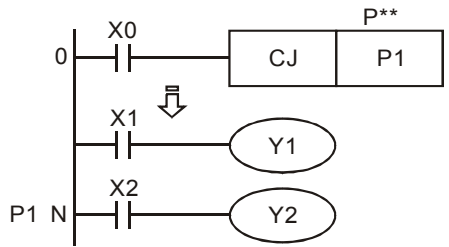
Pointer P: use with application instructions CJ, CALL, and SRET.

CJ condition jump:

When X0=ON, program will jump from 0 to N (designated label P1) and keep on executing without executing the instructions between 0 and N.

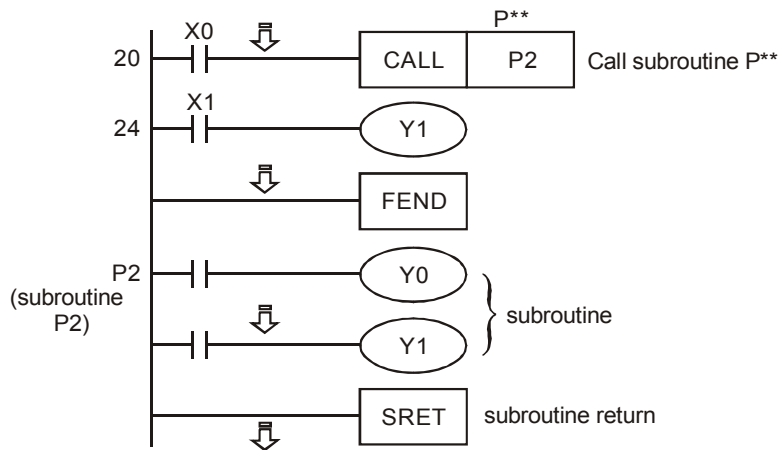
When X0=OFF, program will execute from 0 and keep on executing the followings. CJ instruction won't

be executed at this time.



CALL subroutine, SRET subroutine END:

When X0 is ON, it will jump to P2 to execute the designated subroutine as executing CALL instruction. When executing SRET instruction, return to address 24 to go on executing.



Interrupt pointer I:

It is used with application instruction EI, DI, IRET. There are five functions below. Interrupt insert should be used with EI, interrupt insert enable, interrupt insert disable and IRET interrupt insert return, etc.

1. External interrupt

When input signal of input terminal X0~X5 is triggered on rising-edge, it will interrupt the present program and jump to the designated interrupt subroutine pointer I001(X0), I101(X1), I201(X2), I301(X3), I401(X4), I501(X5) to execute and return to the previous address to execute when executing IRET instruction. Due to special hardware of ELC and is not affected by scan period.

2. Timer interrupt

ELC will stop the present program and jump to the designated interrupt subroutine. ELC will execute automatically for every time period set by 10ms~99ms

3. Counter attained interrupt

The comparison instruction DHSCS of high-speed counter can designate to interrupt present program and jump to the designated interrupt insert subroutine to execute interrupt pointer I010, I020, I030, I040, I050, I060 when the comparison is attained.

4. Communication interrupt

When using communication instruction RS, it can be set to have interrupt request when receiving specific characters. Interrupt I150 and specific characters is set to low byte of D1168.

When ELC connects to a communication device and the received data length will not the same length, setting the end character in D1168 and interrupt subroutine I150. When ELC receives this end character, it will execute interrupt subroutine I150.

5.16 M Relay and D Register Applications

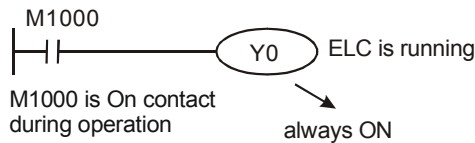
Function Group	ELC Operation Flag
Number	M1000~M1003

Contents:

These relays provide information about the ELC when switched to run mode.

M1000:

Always ON when in run mode.



M1001:

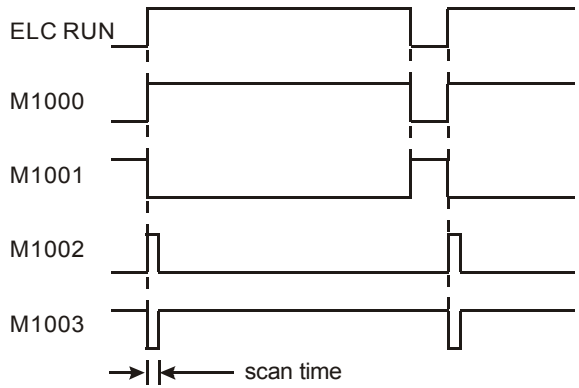
Always OFF when in run mode.

M1002:

ON for the first scan when ELC starts then OFF the rest of the time during run mode. Use to initialize registers, outputs, counters, etc when first entering run mode.

M1003:

OFF for the first scan when ELC starts then ON the rest of the time during run mode. Use to initialize registers, outputs, counters, etc when first entering run mode.

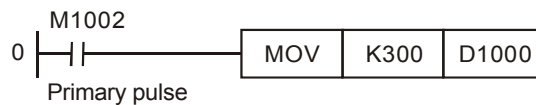


Function Group Monitor Timer

Number D1000

Contents:

1. The watchdog timer (WDT) is used to monitor ELC operation. If the ELC scan time exceeds the preset time setting of the WDT, the ELC ERROR LED will light red and all ELC outputs will be turned OFF.
2. The initial value of WDT is 200ms. You can change this value as desired using the MOV instruction. The following example will set monitor timer to 300ms.



3. The maximum setting for WDT is 32,767ms. You can also use the WDT instruction (API 07) to extend the WDT timer. When this instruction is executed it will reset the WDT's preset value to zero (0).

Function Group Program Capacity

Number D1002

Contents:

This register holds the program capacity of the ELC.

Function Group Grammar

Number M1004, D1004, D1137

Contents:

1. Syntax error, ELC will check for Syntax errors at power on, when a program transfer is taking place. If a syntax error is detected the ELC ERROR LED will blink, M relay M1004=ON.
2. The fault code of the error will be placed in D1004. The address where the fault is located is saved in D1137. If the fault is of a general error it may not have an address associated with it, in this case the value in D1137 will be invalid.

Function Group Scan Time-out Timer

Number M1008, D1008

Contents:

1. When scan time-out during executing, ELC ERROR LED will light and M1008=ON.
2. D1008 saves which address STEP the timeout occurred on.

Function Group Scan Time Monitor

Number D1010~D1012

Contents:

The present value, minimum value and maximum value of scan time are saved in these registers.

D1010: present scan time value.

D1011: minimum scan time value.

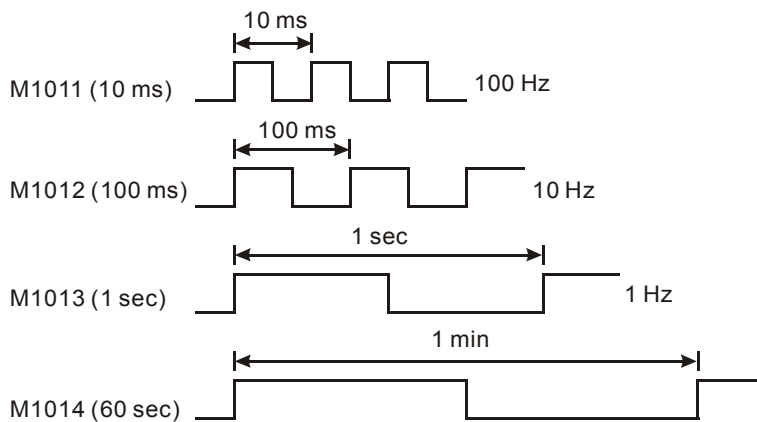
D1012: maximum scan time value.

Function Group Internal Clock Pulse

Number M1011~M1014

Contents:

ELC provides four different clock pulses to aid the application. When ELC is powered on, these four clock pulse will start automatically. All are 50% duty cycle. ELC start timing of run mode and these clock pulses are not synchronized.



Function Group High-speed Timer

Number M1015, D1015

Contents:

The steps for using special M and special D directly:

1. Only valid when ELC is RUN.
2. When M1015=ON, it will start high-speed timer D1015 once ELC finish executing END instruction of that scan period. The minimum unit of D1015 is 100us.
3. The range of D1015 is 0~32,767. When it counts to 32,767, it will start from 0.
4. When M1015=OFF, D1015 will stop counting immediately.

Example:

1. When X10 is ON, set M1015=ON to start high-speed timer and record in D1015.
2. When X10=OFF, set M1015=OFF to close high-speed timer.



Function Group M1016~M1017, M1076, D1313~D1319

Number Perpetual Calendar Clock

Contents:

1. Clock/Calendar relays and registers.

Device	Name	Function
M1016	Year Display	OFF: show the 2 right most bits ON: show the (2 right most bits + 2000)
M1017	Make ± 30 seconds adjustment to clock	When OFF→ON, adjust is triggered Clock = 0~29 seconds, second will be reset to 0 and minute will not change. Clock = 30~59 seconds, reset second to 0 and add 1 to minute.
M1076	Clock / Calendar malfunction	ON when setting exceeds range or battery has run down. Clock will reset to Jan. 1, 2000. 00:00
D1313	Second	0~59
D1314	Minute	0~59
D1315	Hour	0~23
D1316	Day	1~31
D1317	Month	1~12
D1318	Week	1~7
D1319	Year	0~99(2 right-most bit)

2. Adjust method of perpetual clock:

- a) It can use specific command TWR to adjust for PC/PAP/PH modes built-in real time clock. Refer to TWR for detail.
- b) Using peripheralELCSoft to set.

Function Group π (PI)

Number D1018~D1019

Contents:

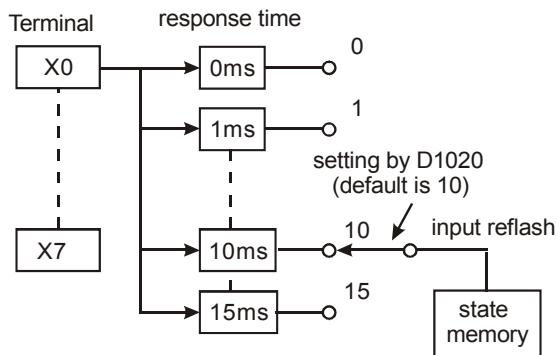
1. 32-bit data register containing floating point value of PI.
2. Floating point value = H 40490FDB

Function Group Input points time delay

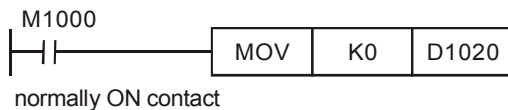
Number D1020~D1021

Contents:

1. X0~X7: 10ms (factory default), 0~1,000ms adjustable. Refer to the usage of special registers D1020. X10, X11: 0 times (factory default), 0~1,000 times adjustable. Refer to the usage of special registers D1021.
2. When ELC powers on is from OFF→ON, the content of D1020 will become to 10 and D1021 will become to 0 automatically.



3. When setting X0~X7 response time to 0ms to execute following program, the faster response time in input terminal will be 50μs due to input terminal connects to RC filter circuit in series.



4. It is not necessary to adjust response time when using high-speed counter, interrupt insert or fast pulse catch (M1056~M1059) in program.
5. It is the same to use instruction REFF or change the content of D1020 and D1021.

Function Group Execution Compelted Flag

Number M1029, M1030

Contents:

Execution Completed Flag:

MTR, HKY, DSW, SEGL, PR:

M1029=ON for a scan period once the instruction finish executing.

PLSY, PLSR:

1. M1029 will be ON after Y0 pulse finishes output and M1030 will be ON after Y1 pulse completes output. When instructions PLSY and PLSR are OFF, M1029 and M1030 will be OFF.
2. User should clear M1029 and M1030 manually.

3. INCD: M1029 will be ON for a scan period when designated group finish comparison.

RAMP, SORT:

1. M1029= ON after completing execution, M1029 must be cleared by user.
2. If this instruction is OFF, M1029 will be OFF.

DABSR, ZRN, DRVI, DRVA:

1. M1029=ON when the first output group Y0 pulses complete sending and M1030=ON when the second output group Y1 pulses complete sending.
2. M1029 or M1030 will be OFF when execute this instruction in the next time and it will be ON after completing execution.

Function Group Communication Error Code

Number D1025

Contents:

Error code when communication error:

- 01: illegal instruction.
- 02: illegal equipment address.
- 03: request data exceeds range.
- 07: checksum error

Function Group Clear Command

Number M1031, M1032

Contents:

M1031 (clear unlatched area) , M1032 (clear latched area)

Device	The component that will be cleared
M1031 Clear unlatched area	The contact state of Y, general M, general S T contact for general and time coil C contact for general, time coil reset coil D present register for general T present register for general C present register for general
M1032 Clear latched area	The contact state of M and S for latched Accumulative timer T contact and time coil Latched C and high-speed counter C contact, count coil Present register D for latched Present register of accumulative timer T Latched C and present register of high-speed counter C

Function Group M1033

Number Output Latched in STOP mode

Contents:

When M1003 = ON, ELC outputs will be latched in their current state when ELC is switched from RUN to STOP.

Function Group Turn all outputs off

Number M1034

Contents:

When M1034 = ON, all outputs will turn off.

Function Group RUN/STOP Switch

Number M1035

Contents:

When M1035 = ON, use input point X7 to be RUN/STOP switch, PB is not support.

Function Group Communication Response Delay

Number D1038

Contents:

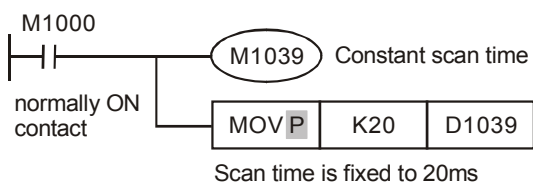
Data response delay time can be set when ELC is a Slave in RS-485 communication. Unit is 0.1ms.

Function Group Constant scan time

Number M1039, D1039

Contents:

1. When M1039 is ON, program scan time is determined by D1039. When program finishes executing, it will execute the next scan as constant scan time attained. If D1039 is less than program scan time, it will scan by program scan time.



2. The relative instructions of scan time are RAMP, HKY, SEGL, ARWS and PR. They should be used with "constant scan time" or "constant time insert interrupt".
3. Especial for instruction HKY, scan time should be set to 20ms and above when it is used 4×4 matrix to be 16 keys to operate.
4. Scan time D1010~D1012 display also include constant scan time.

Function Group Analog Function

Number D1056~D1059, D1110~D1113, D1116~D1118

Contents:

1. PA Models Only
2. If D1118 =5, it will be regarded as 5ms.
3. Resolution of analog input AD card: 12 bits ($\pm 10V$ or $\pm 20mA$)
4. Resolution of analog input DA card: 12 bits (0~10V or 0~20mA)

Device	Function
D1056	Present value of analog input channel 0 (CH0)
D1057	Present value of analog input channel 1 (CH 1)
D1110	Average value of analog input channel 0 (CH 0)
D1111	Average value of analog input channel 1 (CH 1)
D1116	Analog output channel 0 (CH 0).
D1117	Analog output channel 1 (CH 1).
D1118	Analog input filter setting (ms).

Function Group Algorithm Error Flag

Number M1067~M1068, D1067~D1068

Contents:

Algorithm error flag:

Component	Explanation	Latched	STOP→RUN	RUN→STOP
M1067	Algorithm error flag	None	Clear	Latched
M1068	Algorithm error lock flag	None	Unchanged	Latched
D1067	Algorithm error code	None	Clear	Latched
D1068	STEP value of algorithm error	None	Unchanged	Latched

Error code explanation:

D1067 error code	Function
0E18	BCD conversion error
0E19	Divisor is 0
0E1A	Usage exceeds limit (include E and F)
0E1B	It is negative number after doing radical
0E1C	FROM/TO communication error

Function Group M1101, D1101~D1103
Number File Register

Contents:

1. When ELC is powered on or from STOP to RUN, it will check start file register function from M1101, the start number of file register from D1101, read item number of file register from D1102, to determine if it should send file register data to the designated data register automatically or not.
2. Please refer to instructions MEMR and MEMW explanation.

Function Group Pulse Output with Acceleration/Deceleration
Number M1115~M1119, D1104

Contents:

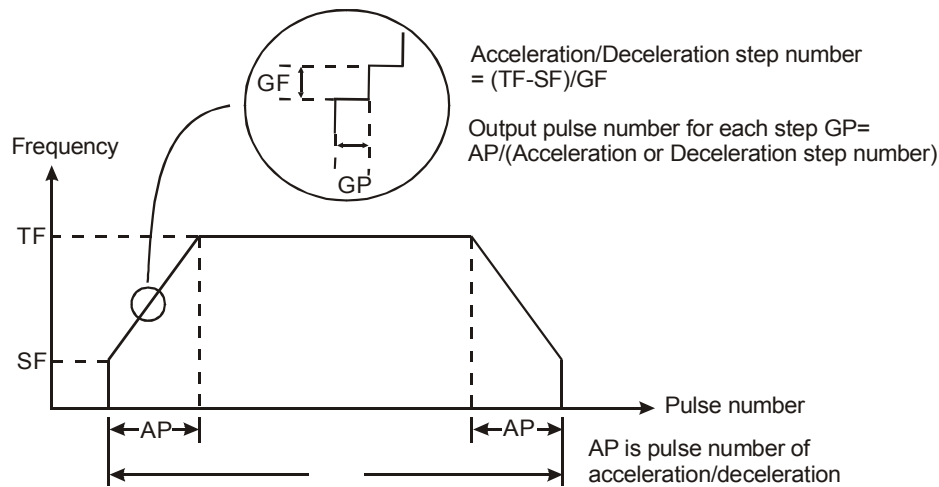
1. The definition of special D and special M which are used by pulse output with acceleration/ deceleration:

Device	Function
M1115	Start switch for accel/decel pulse output
M1116	Flag that is used in acceleration
M1117	Target frequency attained flag
M1118	Flag that is used in deceleration
M1119	Complete function flag
D1104	Using parameter index (correspond to D component)

2. Corresponding table for parameter D1104 (frequency range is 25Hz~10KHz)

Index	Function	
+0	Start frequency (SF)	
+1	Gap frequency (GF)	
+2	Target frequency (TF)	
+3	Total number of pulse output number (lower 16-bit of 32-bit)	(TP)
+4	Total number of pulse output number (upper 16-bit of 32-bit)	
+5	Output pulse number in acceleration area (lower 16-bit of 32-bit)	(AP)
+6	Output pulse number in deceleration area (upper 16-bit of 32-bit)	

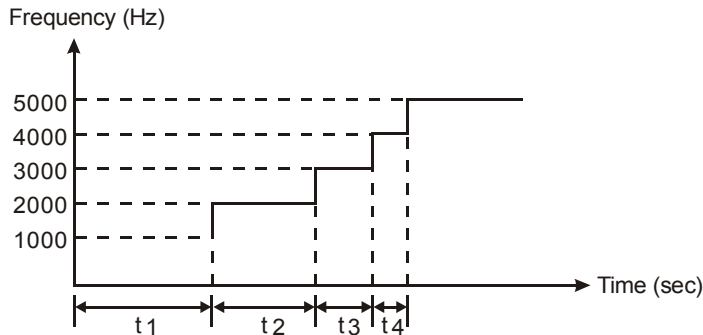
3. It doesn't need to use instruction, it just need to fill parameter chart and set M1115 to start. This function can use Y0 output only, the timing is as following.


Note:

1. This function should be executed when the following conditions all exist. Once a condition doesn't exist, this function won't execute.
2. Start frequency must be less than target frequency.
3. Gap frequency must be less than (target frequency – start frequency)
4. Total number of pulse number must be greater than (accel/decel pulse number *2)
5. Start frequency and target frequency: the minimum is 25Hz and the maximum is 10KHz.
6. Accel/decel pulse number must be more than accel/decel step number
7. When M1115 is from ON to OFF, M1119 will be cleared and M1116, M1117 and M1118 are unchanged. When ELC is from STOP→RUN or from RUN→STOP, M1115~M1119 will be cleared to OFF. And D1104 will be cleared to 0 only when it is from OFF→ON.
8. If the function "acceleration/deceleration pulse output" and instruction PLSY Y0 output exist at the same time, it will execute one action which starts Y0 output first.

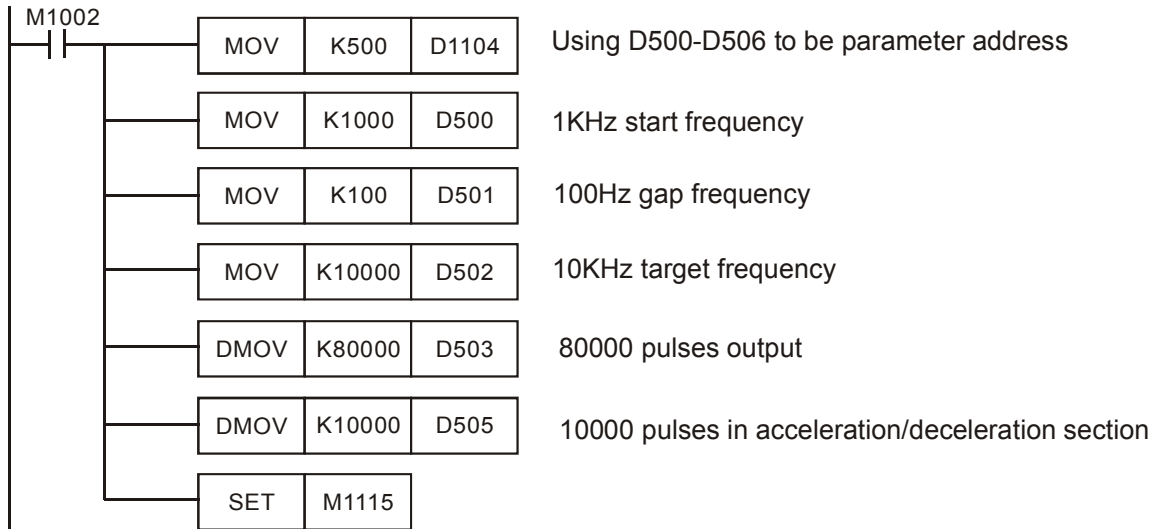
How to calculate action time of each section

1. If start frequency is set to 1KHz, gap frequency is set to 1KHz, target frequency is set to 5KHz, total pulse number is 100 and accel/decel pulse number is 40, timing chart of accel/decel area is in the following.

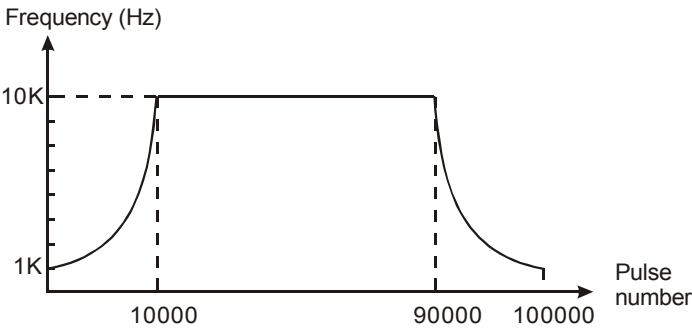


2. You can get accel/decel step = $(5K - 1K) / 1K = 4$ and output number of each pulse is $40 / 4 = 10$. Therefore, you can get $t1 = (1 / 1K) * 10 = 10ms$, $t2 = (1 / 2K) * 10 = 5ms$, $t3 = (1 / 3K) * 10 = 3.33ms$ and $t4 = (1 / 4K) * 10 = 2.5ms$ from the following figure.

Example: Forward/Reverse accel/decel step motor control



- a) When ELC is RUN, it will save each parameter setting into the register that designated by D1104.
- b) When M1115=ON, acceleration/deceleration pulse starts to output.
- c) M1116=ON during acceleration, M1117=ON when speed attained, M1118=ON in deceleration and M1119=ON after finishing executing.
- d) M1115 won't be reset automatically and it needs to be cleared by user.
- e) Actual pulse output curve is in the following:



Function Group Special High-speed pulse output

Number M1133~M1135, D1133

Contents:

1. For PC/PA/PH models, the definition of special D and special M for special high-speed pulse (50KHz) output function:

Device	Function
M1133	Output switch (ON is start executing) for special high-speed pulse (50KHz)
M1134	ON is continuous output switch for special high-speed pulse Y0 (50KHz)
M1135	Output pulse number attained flag for special high-speed pulse Y0 (50KHz)
D1133	Start number of control register (D) for special high-speed pulse Y0 (50KHz)

2. Corresponding table for D1133 parameter

Index	Function
+0	Special high-speed pulse output frequency (lower 16-bit of 32 bits)
+1	Special high-speed pulse output frequency (upper 16-bit of 32 bits)
+2	Special high-speed pulse output number (lower 16-bit of 32 bits)
+3	Special high-speed pulse output number (upper 16-bit of 32 bits)
+4	Display present special high-speed pulse output number (lower 16-bit of 32 bits)
+5	Display present special high-speed pulse output number (upper 16-bit of 32 bits)

Function explanation:

1. Output frequency and output numbers above can be modified when M1133=ON and M1135=OFF. It won't affect present output pulse once output
2. Frequency or output target number is changed. Present output pulse number will be displayed once a scan time update. It will be cleared to 0 when M1133 is from OFF→ON and it will keep that last output number when M1133 is from ON→OFF.

Note:

1. This special high-speed pulse output function can use special Y0 output point in RUN. It can exist with PLSY Y0 at the same time and PLSY (Y1) won't be affected. If instruction PLSY (Y0) is executed prior to this function, this function can't be used and vice versa. When executing this function, general Y0 output will be invalid and outputs point of Y1~Y7 can be used.
2. The difference between this function and instruction PLSY is higher than output frequency. The maximum output can up to 50KHz.

Function Group Extension Connected Detection**Number** D1140, D1142, D1143**Contents:**

D1140: Number of attached special modules (AD, DA, PT, TC, RT, etc), the maximum is 8.

D1142: Digital extension input X point number.

D1143: Digital extension input Y point number.

Function Group Adjustable Acceleration/Deceleration Pulse Output Function Explanation**Number** M1144~M1149, M1154, D1032, D1033, D1144, D1154, D1155**Contents:**

1. For PC/PA/PH Models, the definition of special D and special M of adjustable accel/decel pulse output function:

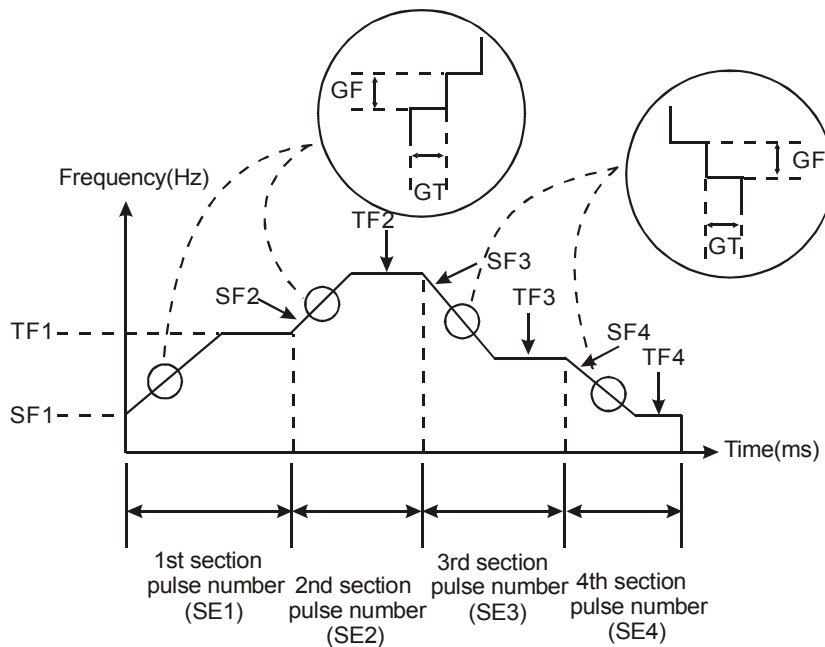
Device	Function
M1144	Start switch of accel/decel pulse output
M1145	Flag that is used in acceleration
M1146	Target frequency attained flag
M1147	Flag that is used in deceleration
M1148	Completed function flag
M1149	stop counting temporarily flag
M1154	Start designated deceleration gap time flag and frequency flag
D1030	Lower 16-bit of 32-bit of Y0 pulse accumulative output numbers
D1031	Upper 16-bit of 32-bit of Y0 pulse accumulative output numbers
D1144	Using parameter index (correspond to D component)
D1154	Recommended value of designated deceleration gap time (10~32767 ms)
D1155	Recommended value of designated acceleration gap frequency (-1~ - 32700 Hz)

2. Corresponding table of parameter D1144

Index	Function
+0	Total segment number (n) (the maximum number is 10)
+1	Present execution segment (read only)
+2	Start frequency of first segment (SF1)
+3	Interval time of first segment (GT1)
+4	Interval frequency of first segment (GF1)
+5	Target frequency of first segment (TF1)
+6	Lower 16-bit of 32-bit of target number of first segment output pulse
+7	Upper 16-bit of 32-bit of target number of first segment output pulse
+8	Start frequency of second segment (SF2)
+9	Interval time of second segment (GT2)
+10	Interval frequency of second segment (GF2)
+11	Target frequency of second segment (TF2)
+12	Lower 16-bit of 32-bit of target number of second segment output pulse
+13	Upper 16-bit of 32-bit of target number of second segment output pulse
:	:
+n*6+2	Start frequency of nth segment (SF _n)
+n*6+3	Interval time of nth segment (GT _n)
+n*6+4	Interval frequency of nth segment (GF _n)
+n*6+5	Target frequency of nth segment (TF _n)
+n*6+6	Lower 16-bit of 32-bit of target number of nth segment output pulse
+n*6+7	Upper 16-bit of 32-bit of target number of nth segment output pulse

Function Explanation:

This function can only be used for Y0 output point and the timing will be as follows. After filling parameter table, set M1144 to start (it should be used in RUN mode)



Usage rule and restriction:

1. The minimum frequency of start frequency and target frequency should be equal to or greater than 200Hz. If it is less than 200Hz, it means finish executing or not to execute.
2. The maximum frequency of start frequency of target frequency is 32700Hz. It will execute in 32700Hz as it is greater than 32700Hz.
3. The interval time range is 1~32767ms and its unit is ms.
4. The interval frequency range in acceleration segment is 1Hz~32700Hz and in deceleration segment is -1~32700Hz. If it is set to 0Hz, the executed segment can't be up to target frequency, but it will transfer to execute next segment after reaching target number.
5. Target number of segment pulse output should be greater than $((GF*GT/1000)*((TF-SF)/GF))$. Refer to example 1 for detail. Once Target number of segment pulse output isn't greater than $((GF*GT/1000)*((TF-SF)/GF))$, this function can't be used. The improve method is to add interval time or add target number of pulse output.
6. If there is Y0 output designated by high-speed instruction in RUN mode, Y0 output
7. Instruction will be started as high priority.
8. After starting to execute M1144, if M1148 outputs without attaining completed function flag and M1144 is closed, this function will start deceleration function. If designated acceleration function flag M1154 is OFF, it will reduce 200Hz per 200ms and stop output pulse till output frequency is less than 200Hz and set M1147 to deceleration flag. But if designated deceleration flag M1154 is ON, it will be executed by interval time and frequency that defined by user. And interval time can't be less than or equal to 0 (if it is less than or equal to 0, factory setting will be set to 200ms).

Interval frequency can't be greater than or equal to 0 (factory setting will be set to -1KHz when it is equal to 0 and factory setting will be added negative sign automatically when it is greater than 0.)

9. When M1148 attains completed function flag and M1144 is closed, this function won't start deceleration function and it will clear M1148 flag. Once M1144 is closed, it will clear M1149 flag.
10. The execution segment of this function will execute by total segment number. The maximum segment is 10 segments.
11. The acceleration/deceleration of this function will execute by start frequency of the next segment, i.e. when target frequency of execution segment is less than start frequency of the next segment, the next segment is acceleration and the target frequency of the next segment must be greater than start frequency of the next segment. When target frequency of execution segment is greater than the next segment frequency, the next segment is deceleration, therefore, target frequency of the next segment must be less than start frequency of the next segment. If user can't set it by this way, we can't ensure that you can get correct output pulse.
12. When STOP→RUN, M1144~M1149 will be cleared to OFF. When RUN→STOP, M1144 will be cleared and M1145~M1149 won't be cleared. D1144 will be cleared to 0 when it is from OFF→ON and unchanged in other case.
13. The valid parameter range is D0~D999 and D2000~D4999. ELC won't execute this instruction, and close M1144 if the parameter is out of range (includes all segment parameters).

Example 1:

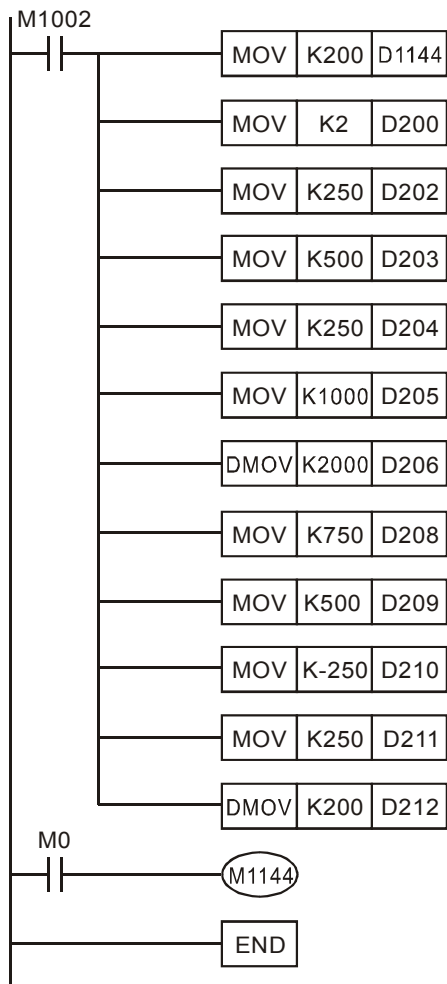
To calculate output number of acceleration/deceleration of each segment and target frequency

If setting start frequency of segment to 200Hz, segment interval time to 100ms, segment gap frequency to 100Hz, segment target frequency to 500Hz and target number of segment pulse is 1000 pulses. The calculation will be in the following:

1. Output pulse number at start acceleration/deceleration is $200 \times 100 / 1000 = 20$ pulses
2. Output pulse number of the first acceleration interval is $300 \times 100 / 1000 = 30$ pulses
3. Output pulse number of the second acceleration interval is $400 \times 100 / 1000 = 40$ pulses
4. Output pulse number of target frequency is $1000 - (40 + 30 + 20) = 910$ pulses
5. (NOTE: it is recommended to set this number to be greater than 10)
6. Output time of target frequency is $1 / 500 \times 910 = 1820$ ms
7. Total time of this segment is $1820 + 3 \times 100 = 2120$ ms

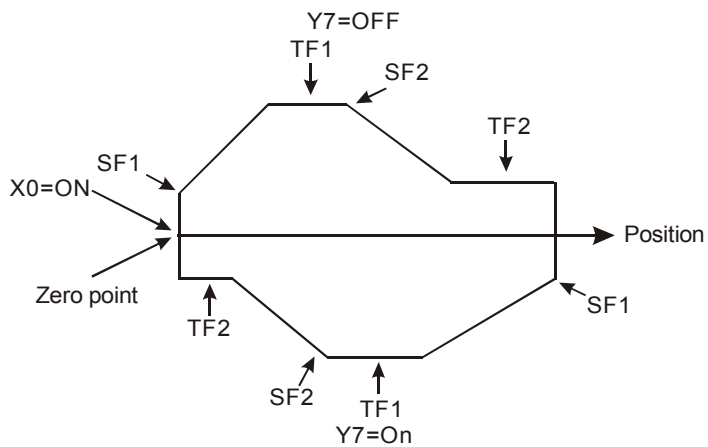
Example 2:

Simple acceleration/deceleration pulse output program of a segment acceleration and a segment deceleration



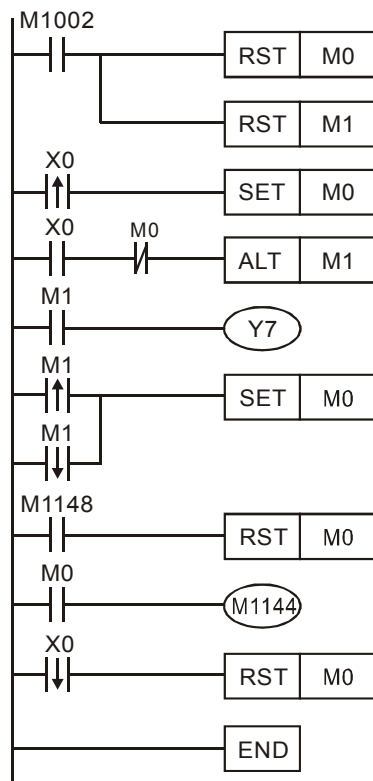
Example 3:

Pulse output program of a segment acceleration/deceleration with direction



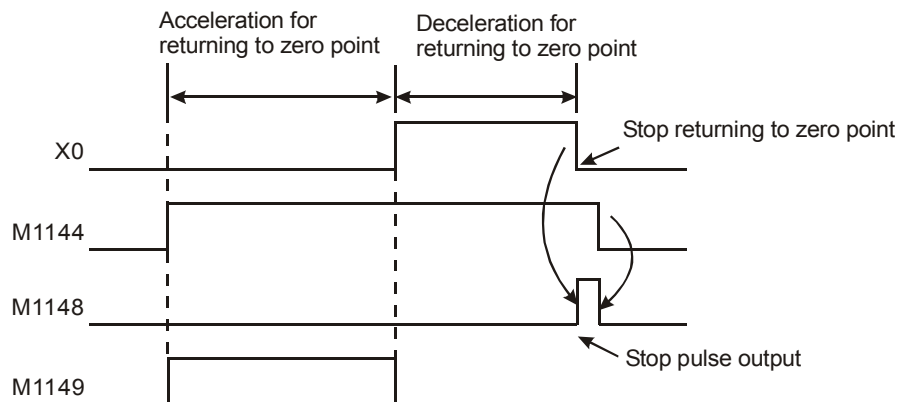
Explanation:

1. Acceleration/deceleration setting is as example 2.
2. Figure above is the example of position movement. When X0 contact is ON, it will start to move and it will stop when X0 contact is OFF. (Y7 is for direction setting)
3. Program is shown in the following.

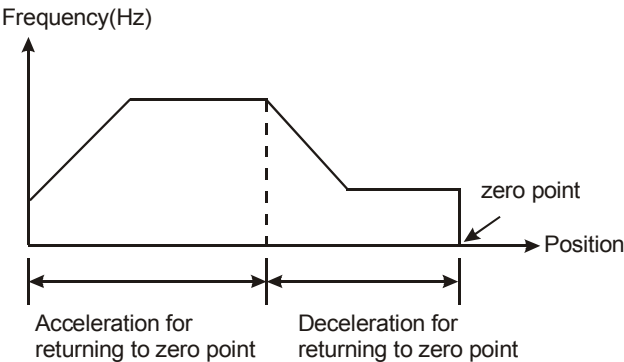


Example 4: apply acceleration and deceleration of a segment to zero point return program.

1. Relative flag timing chart is shown in the following.



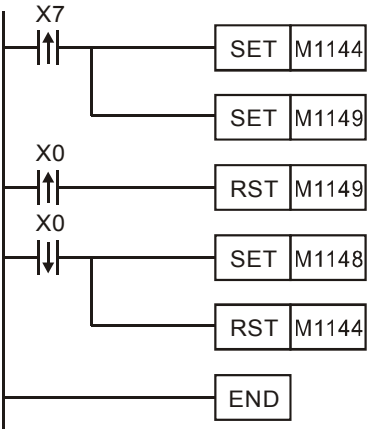
2. The relation between frequency and position are shown in the following.



3. Number setting of acceleration/deceleration, frequency and pulse are shown in the following.
(correspond to component D)

Index	Settings
+0	2
+2	250(Hz)
+3	100(ms)
+4	500(Hz)
+5	10000(Hz)
+6, +7	10(pulse)
+8	9750(Hz)
+9	50(ms)
+10	-500(Hz)
+11	250(Hz)
+12, +13	30000(pulse)

4. Program is shown in the following: (it assumes contact X7 to be start reset trigger switch)



5. Explanation:

- After contact X7 is triggered, M1144 will set to start acceleration and set M1149 not to count pulse number. And it will send 10 pulses once deceleration switch X0 is triggered and then enter deceleration segment.
- To set M1148 to end pulse output by manual and close this function once X0 is closed.
- Note: This example is just an application method that user should adjust parameters settings used in
- acceleration/deceleration segment according to actual machine characteristics and limitation.

Function Group 2-phase Pulse Output Function

Number M1172~M1174, D1172~D1177

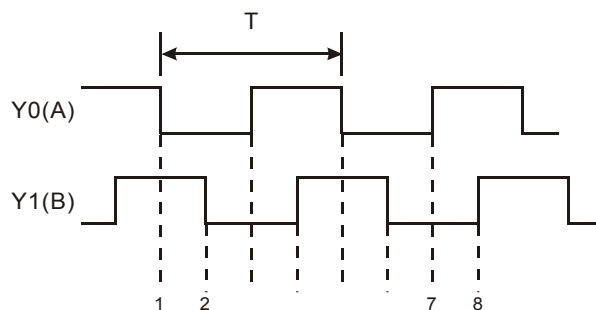
Contents:

For PC/PA/PH Models, the definition of special D and special M of 2-phase output function:

Device	Function Explanation
M1172	2-phase pulse output switch
M1173	ON is continuous output switch
M1174	Output pulse number attained flag
D1172	2-phase output frequency (12Hz~20KHz)
D1173	2-phase output mode selection (k1 and k2)
D1174	Lower bit of 32-bit of 2-phase output pulse target number
D1175	Upper bit of 32-bit of 2-phase output pulse target number
D1176	Lower bit of 32-bit of 2-phase present output pulse number
D1177	Upper bit of 32-bit of 2-phase present output pulse number

Function Explanation:

- Output frequency = $1/T$ as shown in the figure below. There are two output modes, k1 and k2, k1 means A phase gets ahead of B phase and k2 means B phase gets ahead of A phase. Output number calculation adds 1 once there is a phase difference, such as figure below, there are 8 output pulses. When output numbers attains, M1174 will be ON and if you want to clear M1174, you should close M1172.



- Output frequency, output target number and mode selection can be modified when M1172=ON and M1174=OFF. The modification of output frequency and output target number won't affect present output pulse number but mode selection modification will clear present output pulse number to 0. Present output pulse number will be updated once scan time updates and it will clear to 0 when M1172 is from Stop→Run, and keep that last output number when M1172 is from Run→Stop.

Note:

This function just can be used at RUN mode and can exist in program with PLSY instruction. But if instruction PLSY is executed first, this function can't be used, and vice versa.

Function Group VR Volume

Number M1178~M1179, D1178~D1179

Contents:

For PC/PH models, the definition of special D and special M of built-in 2 points VR Variable resistor function:

Device	Function
M1178	Start VR0
M1179	Start VR1
D1178	VR0 value
D1179	VR1 value

Function explanation:

- This function only can be used at RUN mode. When M1178=ON, the variable value of VR 0 will be converted to digit 0~255 and saved in D1178. When M1179=ON, the variable value of VR 1 will be converted to digit 0~255 and saved in D1179.
- Refer to instruction VRRD for detail.

Function Group Power Loss Latched Range Setting

Number D1200~D1219

Contents:

For PC/PA/PH Models, to set latched range. The latched range will be from start address number to end address number.

Function Group Input Point X can force to be ON/OFF

Number M1304

Contents:

For PC/PA/PH Models, When M1304=ON, input point X (X0-X7) of ELC can be forced to be ON-OFF by using peripheral ELCSOFT or HHP

Function Group Easy ELC Link

Number M1350-M1354, M1360-M1439, D1355-D1370, D1415-D1465, D1480-D1991

Contents:

Explanation of Special D and special M explanation of ELC LINK ID1-ID8:

MASTER ELC															
SLAVE ID 1		SLAVE ID 2		SLAVE ID 3		SLAVE ID 4		SLAVE ID 5		SLAVE ID 6		SLAVE ID 7		SLAVE ID 8	
Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in
D1480 D1495	D1496 D1511	D1512 D1527	D1528 D1543	D1544 D1559	D1560 D1575	D1576 D1591	D1592 D1607	D1608 D1623	D1624 D1639	D1640 D1655	D1656 D1671	D1672 D1687	D1688 D1703	D1704 D1719	D1720 D1735
Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num
D1434	D1450	D1435	D1451	D1436	D1452	D1437	D1453	D1438	D1454	D1439	D1455	D1440	D1456	D1441	D1457
Slave Device Internal Address to Read/Write															
D1355	D1415	D1356	D1416	D1357	D1417	D1358	D1418	D1359	D1419	D1360	D1420	D1361	D1421	D1362	D1422
Indicate if SLAVE ELC exists															
M1360		M1361		M1362		M1363		M1364		M1365		M1366		M1367	
Action indication flag for master ELC do to slave ELC															
M1376		M1377		M1378		M1379		M1380		M1381		M1382		M1383	
Read/write error flag															
M1392		M1393		M1394		M1395		M1396		M1397		M1398		M1399	
Read completed flag (Whenever finishing a ELC read/write, this flag will be set to OFF automatically)															
M1408		M1409		M1410		M1411		M1412		M1413		M1414		M1415	
Write completed flag (whenever finishing a ELC read/write, this flag will be OFF automatically)															
M1424		M1425		M1426		M1427		M1428		M1429		M1430		M1431	
↓		↓		↓		↓		↓		↓		↓		↓	
SLAVE ID 1		SLAVE ID 2		SLAVE ID 3		SLAVE ID 4		SLAVE ID 5		SLAVE ID 6		SLAVE ID 7		SLAVE ID 8	
Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in
D100 D115	D200 D215	D100 D115	D200 D215	D100 D115	D200 D215	D100 D115	D200 D215	D100 D115	D200 D215	D100 D115	D200 D215	D100 D115	D200 D215	D100 D115	D200 D215

Factory setting of Communication address for reading is H1064 (D100).

Factory setting of Communication address for writing is H10C8 (D200).

Explanation of Special D and special M explanation of ELC LINK ID9-ID16:

MASTER ELC															
SLAVE ID 9		SLAVE ID 10		SLAVE ID 11		SLAVE ID 12		SLAVE ID 13		SLAVE ID 14		SLAVE ID 15		SLAVE ID 16	
Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in
D1736	D1752	D1768	D1784	D1800	D1816	D1832	D1848	D1864	D1880	D1896	D1912	D1928	D1944	D1960	D1976
D1751	D1767	D1783	D1799	D1815	D1831	D1847	D1863	D1879	D1895	D1911	D1927	D1943	D1959	D1975	D1991
Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num	Item num
D1442	D1458	D1443	D1459	D1444	D1460	D1445	D1461	D1446	D1462	D1447	D1463	D1448	D1464	D1449	D1465
Slave Device Internal Address to Read/Write															
D1363	D1423	D1364	D1424	D1365	D1425	D1366	D1426	D1367	D1427	D1368	D1428	D1369	D1429	D1370	D1430
Indicate if SLAVE ELC exists															
M1368	M1369	M1370	M1371	M1372	M1373	M1374	M1375								
Action indication flag for master ELC do to slave ELC															
M1384	M1385	M1386	M1387	M1388	M1389	M1390	M1391								
Read/write error flag															
M1400	M1401	M1402	M1403	M1404	M1405	M1406	M1407								
Read completed flag (Whenever finishing a ELC read/write, this flag will be set to OFF automatically)															
M1416	M1417	M1418	M1419	M1420	M1421	M1422	M1423								
Write completed flag (whenever finishing a ELC read/write, this flag will be set to OFF automatically)															
M1432	M1433	M1434	M1435	M1436	M1437	M1438	M1439								

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

SLAVE ID 9		SLAVE ID 10		SLAVE ID 11		SLAVE ID 12		SLAVE ID 13		SLAVE ID 14		SLAVE ID 15		SLAVE ID 16	
Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in
D100	D200	D100	D200	D100	D200	D100	D200	D100	D200	D100	D200	D100	D200	D100	D200
D115	D215	D115	D215	D115	D215	D115	D215	D115	D215	D115	D215	D115	D215	D115	D215

Factory setting of Communication address for reading is H1064 (D100).

Factory setting of Communication address for writing is H10C8 (D200).

Explanation:

1. The basic communication protocol for ELC LINK is MODBUS
2. All communication format used to connect to ELC should be the same (set D1120 for ELC) and it supports ASCII and RTU mode.

3. For PC/PA/PH series, it can read/write 16 WORDs (max) from one master ELC to one slave ELC.
4. The maximum slave ELCs for a master ELC to connect is 16 slave ELCs.
5. The ID of slave ELC should be fixed to 1~16 and each slave ID can't be repeated.
6. One to one connection can connect to RS-232, RS-485 and RS-422 in series
7. One to multiple connection can connect to RS-485 in series
8. The start slave ID can be set by D1399 of master ELC and the ID for each slave and master ELC(set by D1121) can't repeat.

Operation:

1. Setting the baudrate and communication to be the same as all connected slave peripherals.
COM1_RS-232: D1036, COM2_RS-485: D1120.
2. Setting master ELC ID by D1121 and setting the start slave ID by D1399 of master ELC first.
Then, setting slave ID by each slave ELC. The ID of master ELC and slave ELC can't repeat.
3. Setting read/write items of slave (the maximum is 16 items). (refer to special D for detail)
4. Set device communication address to read/write to slave. (refer to Special D explanation above
for special D setting. Factory setting of communication address for reading is H1064 (D100) and
writing is H10C8 (D200).
5. Setting to start synchronous read/write function (M1354). It starts when M1354=ON.
6. Setting ELC LINK automatically (M1351)
7. Setting ELC LINK manually (M1352)
8. Start MASTER ELC LINK (M1350)

Master ELC action explanation:

1. When M1354=ON, it will use Modbus Function H17 (synchronous read/write function) for ELC
EASY LINK communication function. If item number for writing is set to 0, ELC will use Modbus
Function H03 (read multiple WORDs) for ELC EASY LINK communication function. In the same
way, if item number for reading is set to 0, ELC will use Modbus Function H06 (write one WORD)
or Modbus Function H10 (write multiple WORDs) for ELC EASY LINK communication function.
2. Slave ID detection: When M1350=ON, Master ELC is started and then detects the number of
active slaves and records the count in D1433.
3. You can see if there is a slave ELC by searching M1360-M1375 which saved slave ID 1-16. ON =
Slave exists.
4. If the detection of active slaves = 0, M1350 will be OFF and stop Link at the same time. ELC will
only detect the number of slaves when M1350 first turns ON.

5. Read/write of master and slave ELC: After finishing detecting slave, master ELC will read/write to each slave. The slave that master can do read/write is slave ID got after detecting slave ID. Once slave ELC is added after detecting, master can't do read/write to it till the next detection.
6. Master ELC will read first and the maximum range is 16 slave ELCs start from D100. After reading, ELC will write and the maximum range is 16 slave ELCs start from D200.
7. Master ELC will read/write to slave ELC in order, i.e. it will read/write to the next slave after finishing a slave.

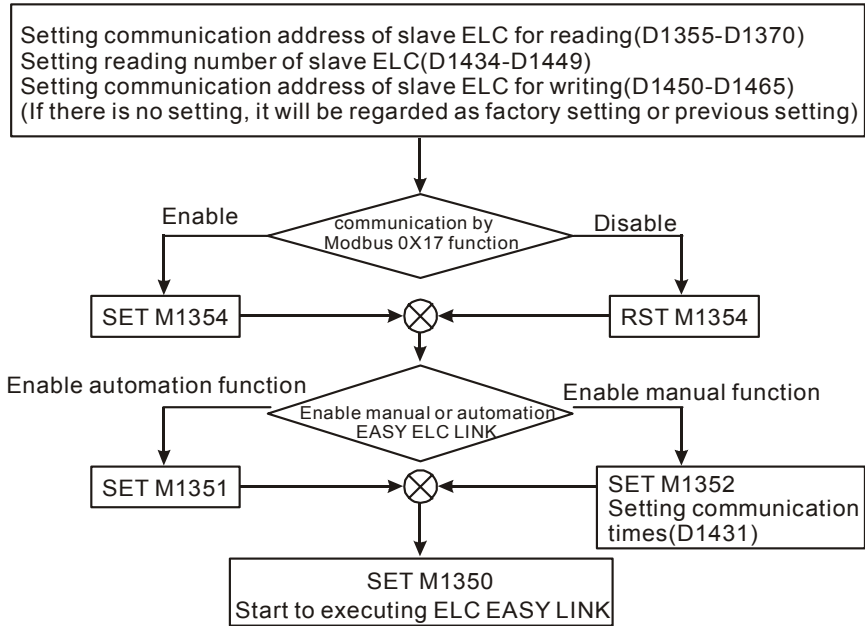
Automatic/ Manual model explanation:

1. Automatic mode: if M1351 = ON. Master ELC will read/write to slave till M1350 = OFF.
2. Manual mode: ELC needs to set times of read in D1431. One time means finish all Slave read/writes. When ELC starts Link, D1432 will start to count times of Link. When D1431 = D1432, ELC will stop Link and force M1351 = OFF at the same time. If M1351 is forced ON, ELC will start to link according to D1431 value automatically.

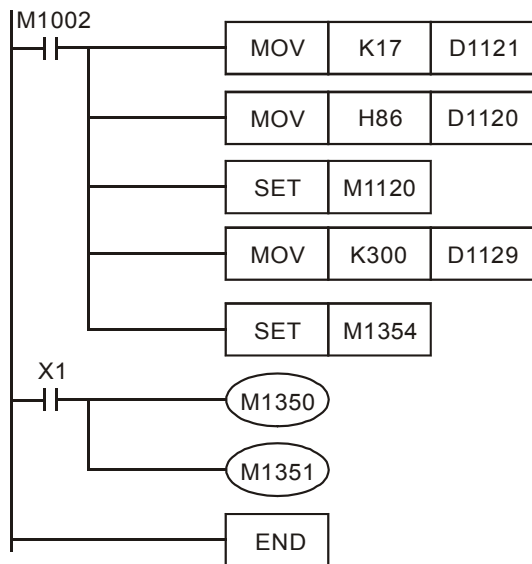
Note:

1. Automatic mode M1351 and manual mode M1352 can't be ON at the same time.
2. Please clear M1350 first before switching automation/manual mode.
3. Communication time-out can be set by D1129. The setting range is from 300 to 3000. When out of range, it will be regarded as 300 when it is less than 300 and regarded as 3000 when it is larger than 3000. ELC LINK function is only valid when baud rate is larger than 1200 bps. When baud rate is less than 9600 bps, please set communication time-out to more than 1 second.
4. It won't communicate when write/read item is 0.
5. It doesn't allow 32-bit counter read/write in

Operation flow chart:



Example: ELC EASY LINK uses with M1354



Function Group COM Port Function
Number M1120, M1138, M1139, M1143, D1036, D1120

Contents:

The additional COM ports (COM1: RS-232, COM2: RS-485) for PB/PC/PA/PH MPU support communication format of MODBUS ASCII/RTU. The speed can up to 115200 bps. COM1 and COM2 can be used simultaneously.

COM1:

It can only be used as slave and supports ASCII/RTU communication format, baudrate(115200bps max), and modify the length of data bit, including Data bits, Parity bits and Stop bits.

COM2:

It can be used as master or slave and supports ASCII/RTU communication format, baudrate (115200bps max), and modify the length of data bit, including Data bits, Parity bits and Stop bits.

Format setting:

Item \ Port	COM1	COM2
Communication format	D1036	D1120
Communication setting holding	M1138	M1120
ASCII/RTU mode	M1139	M1143

D1036:

RS-232 communication protocol of slave ELC. (b8-b15 is not used)

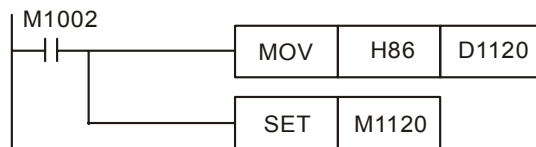
D1120:

RS-485 communication protocol of master/slave ELC. Refer table below for setting:

	Content		
b0	Data Length	0: 7 data bits, 1: 8 data bits	
b1 b2	Parity bit	00: None 01: Odd 11: Even	
b3	Stop bits	0: 1 bit, 1: 2bits	
b4 b5 b6 b7	Baud rate	0001(H1): 110 0010(H2): 150 0011(H3): 300 0100(H4): 600 0101(H5): 1200 0110(H6): 2400 0111(H7): 4800 1000(H8): 9600 1001(H9): 19200 1010(HA): 38400 1011(HB): 57600 1100(HC): 115200	
b8	Start word selection	None	D1124
b9	First end word selection	None	D1125
b10	Second end word selection	None	D1126
b11~b15	No definition		

Example 1: modification method for COM2 communication format

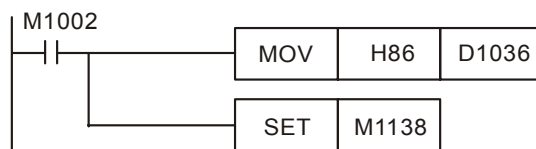
1. When modifying COM2 communication format, it needs to add program code below in the front of program. When ELC is from STOP to RUN, it will detect if M1120 is ON at ELC first scan time. If M1120 is ON, it will change COM2 setting by D1120.
2. Modifying COM2 communication format to ASCII mode, 9600bps, 7 data bits, even parity, 1 stop bits (9600, 7, E, 1).

**Note:**

1. Do NOT write any communication command in the program when COM2 is used as slave.
2. After modifying communication format, communication format setting won't be changed when ELC is from RUN to STOP.
3. After modifying communication format, communication format will be changed to factory setting after power OFF and power ON again.

Example 2: modification method for COM1 communication format

1. When modifying COM1 communication format, it needs to add program code below in the front of program. When ELC is from STOP to RUN, it will detect if M1138 is ON at ELC first scan time. If M1138 is ON, it will change COM1 setting by D1036.
2. Modifying COM1 communication format to ASCII mode, 9600bps, 7 data bits, even parity, 1 stop bits (9600, 7, E, 1).

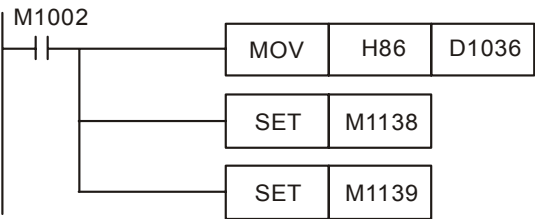
**Note:**

1. After modifying communication format, communication format setting won't be changed when ELC is from RUN to STOP.
2. After modifying communication format, communication format will be changed to factory setting after power OFF and power ON again.

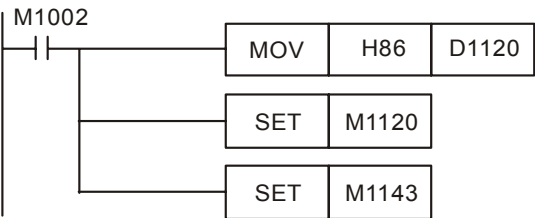
Example 3: RTU mode setting of COM1 and COM2

Both of COM1 and COM2 support ASCII/RTU mode. COM1 is used to set flag by M1139 and COM2 is used to set flag by M1143. When flag is ON, it is RTU mode. When flag is OFF, it is ASCII mode.

COM1:



COM2:



5.17 Fault Code Information

If the ELC ERROR LED is flashing or special relay M1004=ON after writing program in ELC, the problem may be an invalid operand or grammar error. You can get the fault code saved in special register D1004 and check it to the following table to get error message, error address is saved in D1137. (D1137 will be invalid if it is general loop error)

Fault Code	Description	Action
0001	Operand bit device S exceeds the valid range	Check the D1137 (Error step number) Re-enter the instruction correctly
0002	Label P exceeds the valid range or duplicated	
0003	Operand KnSm exceeds the valid range	
0102	Interrupt pointer I exceeds the valid range or duplicated	
0202	Instruction MC exceeds the valid range	
0302	Instruction MCR exceeds the valid range	
0401	Operand bit device X exceeds the valid range	
0403	Operand KnXm exceeds the valid range	
0501	Operand bit device Y exceeds the valid range	
0503	Operand KnYm exceeds the valid range	
0601	Operand bit device T exceeds the valid range	
0604	Operand word device T register exceeds limit	
0801	Operand bit device M exceeds the valid range	
0803	Operand KnMm exceeds the valid range	
0D01	DECO operand misuse	
0D02	ENCO operand misuse	
0D03	DHSCS operand misuse	
0D04	DHSCR operand misuse	
0D05	PLSY operand misuse	
0D06	PWM operand misuse	
0D07	FROM/TO operand misuse	
0D08	PID operand misuse	
0D09	SPD Misuse Operand	
0E01	Operand bit device C exceeds the valid range	
0E04	Operand word device C register exceeds limit	
0E05	DCNT operand CXXX misuse	
0E18	BCD conversion error	
0E19	Division error (divisor=0)	
0E1A	Floating Point exceeds usage range	
0E1B	Negative number after radical expression	

Fault Code	Description	Action
0E1C	FROM/TO communication error	Check the D1137 (Error step number) Re-enter the instruction correctly
0F04	Operand word device D register exceeds limit	
0F05	DCNT operand DXXX misuse	
0F06	SFTR operand misuse	
0F07	SFTL operand misuse	
0F08	REF operand misuse	
1000	ZRST operand misuse	
10EF	E and F misuse operand or exceed the usage range	
2000	Usage exceed limit (MTR, ARWS, HOUR)	

Fault Code	Description	Action
C400	An unrecognized instruction code is being used	A circuit error occurs if a combination of instructions is incorrect or badly specified. Select programming mode and correct the identified error
C401	Loop error	
C402	LD / LDI continuously use more than 9 times	
C403	MPS continuously use more than 9 times	
C404	FOR-NEXT exceeds 6 levels	
C405	STL/RET used between FOR-NEXT SRET/IRET used between FOR-NEXT	
	MC/MCR used between FOR-NEXT END / FEND used between FOR-NEXT	
C407	STL continuously use more than 9 times	
C408	Use instruction MC/MCR in STL, Use I/P in STL	
C409	Use STL/RET in subroutine or interrupt program	
C40A	Use MC/MCR in subroutine Use MC/MCR in interrupt program	
C40B	MC/MCR doesn't start from N0 or discontinuously	
C40C	MC/MCR corresponding value N is different	
C40D	Use I/P incorrectly	
C40E	IRET doesn't follow by the last FEND instruction SRET doesn't follow by the last FEND instruction	
C41C	The number of input/output points of I/O extension unit exceeds valid range	
C4EE	No END instruction in program	

Instruction Set

This chapter contains all of the instructions that are used with the ELC controllers.
Detailed information concerning the usage of the instructions.

This Chapter Contains

6.1 Basic Instruction Explanations	6-2
6.2 Pointers	6-10
6.3 Interrupt Pointers	6-11
6.4 Basic Instructions (without API numbers)	6-13
6.5 Application Programming Instructions	6-14
6.6 Numerical List of Instructions	6-25
6.7 Detailed Instruction Explanation	6-33

6 Instruction Set

6.1 Basic Instruction Explanations

Mnemonic	Operands	Function	Program steps	Controllers			
				PB	PC	PA	PH
LD	X, Y, M, S, T, C	Load A contact	1				

Explanations:

The LD command is used on the A contact that has its start from the left BUS or the A contact that is the start of a new block of program when using the ORB and ANB instructions (see later sections).

Program Example:

Ladder diagram:



Command code:

LD X0

AND X1

OUT Y1

Operation:

; Load contact A of X0

; Connect to contact A of X1 in series

; Drive Y1 coil

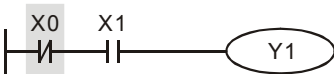
Mnemonic	Operands	Function	Program steps	Controllers			
				PB	PC	PA	PH
LDI	X, Y, M, S, T, C	Load B contact	1				

Explanations:

The LDI command is used on the B contact that has its start from the left BUS or the B contact that is the start of a new block of program when using the ORB and ANB instructions (see later sections).

Program Example:

Ladder diagram:



Command code:

LDI X0

AND X1

OUT Y1

Operation:

; Load contact B of X0

; Connect to contact A of X1 in series

; Drive Y1 coil

Mnemonic	Operands	Function	Program steps	Controllers
AND	X, Y, M, S, T, C	Series connection- A contact	1	PB PC PA PH

Explanations:

The AND command is used in the series connection of A contact.

Program Example:

Ladder diagram:



Command code:

LDI X1

Operation:

; Load contact B of X1

AND X0

; Connect to contact A of X0 in series

OUT Y1

; Drive Y1 coil

Mnemonic	Operands	Function	Program steps	Controllers
ANI	X, Y, M, S, T, C	Series connection- B contact	1	PB PC PA PH

Explanations:

The ANI command is used in the series connection of B contact.

Program Example:

Ladder diagram:



Command code:

LD X1

Operation:

; Load contact A of X1

ANI X0

; Connect to contact B of X0 in series

OUT Y1

; Drive Y1 coil

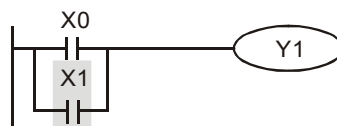
Mnemonic	Operands	Function	Program steps	Controllers
OR	X, Y, M, S, T, C	Parallel connection- A contact	1	PB PC PA PH

Explanations:

The OR command is used in the parallel connection of A contact.

Program Example:

Ladder diagram:



Command code:

LD X0

Operation:

; Load contact A of X0

OR X1

; Connect to contact A of X1 in parallel

OUT Y1

; Drive Y1 coil

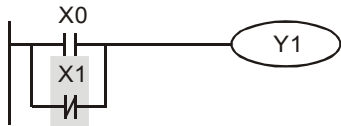
Mnemonic	Operands	Function	Program steps	Controllers			
				PB	PC	PA	PH
ORI	X, Y, M, S, T, C	Parallel connection- B contact	1				

Explanations:

The ORI command is used in the parallel connection of B contact.

Program Example:

Ladder diagram:



Command code:

Operation:

LD X0 ; Load contact A of X0
ORI X1 ; Connect to contact B of X1 in parallel
 OUT Y1 ; Drive Y1 coil

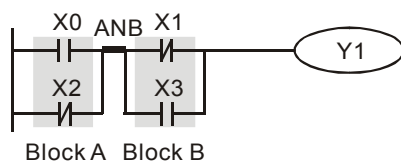
Mnemonic	Function	Program steps	Controllers			
			PB	PC	PA	PH
ANB	Series connection (Multiple Circuits)	1				

Explanations:

To perform the “AND” calculation between the previous reserved logic results and contents of the accumulative register.

Program Example:

Ladder diagram:



Command code:

Operation:

LD X0 ; Load contact A of X0
 ORI X2 ; Connect to contact B of X2 in parallel
 LDI X1 ; Load contact B of X1
 OR X3 ; Connect to contact A of X3 in parallel
ANB ; Connect circuit block in series
 OUT Y1 ; Drive Y1 coil

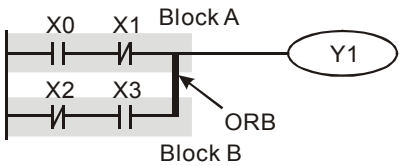
Mnemonic	Function	Program steps	Controllers			
			PB	PC	PA	PH
ORB	Parallel connection (Multiple circuits)	1				

Explanations:

To perform the “OR” calculation between the previous reserved logic results and contents of the accumulative register.

Program Example:

Ladder diagram:



Command code:	Operation:
LD X0	; Load contact A of X0
ANI X1	; Connect to contact B of X1 in series
LDI X2	; Load contact B of X2
AND X3	; Connect to contact A of X3 in series
ORB	; Connect circuit block in parallel
OUT Y1	; Drive Y1 coil

Mnemonic	Function	Program steps	Controllers				
MPS	Store the current result of the internal ELC operations	1	<table><tr><td>PB</td><td>PC</td><td>PA</td><td>PH</td></tr></table>	PB	PC	PA	PH
PB	PC	PA	PH				

Explanations:

MPS stores the connection point of the ladder circuit so that further coil branches can recall the value later.

Mnemonic	Function	Program steps	Controllers				
MRD	Reads the current result of the internal ELC operations	1	<table><tr><td>PB</td><td>PC</td><td>PA</td><td>PH</td></tr></table>	PB	PC	PA	PH
PB	PC	PA	PH				

Explanations:

MRD recalls or reads the previously stored connection point data and forces the next contact to connect to it.

Mnemonic	Function	Program steps	Controllers				
MPP	Pops (recalls and removes) the currently stored result	1	<table><tr><td>PB</td><td>PC</td><td>PA</td><td>PH</td></tr></table>	PB	PC	PA	PH
PB	PC	PA	PH				

Explanations:

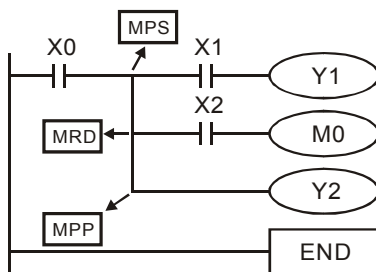
MPP pops (recalls and removes) the stored connection point. First, it connects the next contact, then it removes the point from the temporary storage area.

Basic points to remember:

1. Use these instructions to connect output coils to the left hand side of a contact. Without these instructions connections can only be made to the right hand side of the last contact.
2. For every MPS instruction there MUST be a corresponding MPP instruction.
3. The last contact or coil circuit must connect to an MPP instruction.
4. At any programming step, the number of active MPS-MPP pairs must be no greater than 8.

Program Example:

Ladder diagram:



Command code:

Operation:

```

LD      X0      ; Load contact A of X0
MPS          ; Save to stack
AND     X1      ; Connect to contact A of X1 in series
OUT     Y1      ; Drive Y1 coil
MRD          ; Read from stack
AND     X2      ; Connect to contact A of X2 in series
OUT     M0      ; Drive M0 coil
MPP          ; Read from stack and pop pointer
OUT     Y2      ; Drive Y2 coil
END      ; Program end

```

MPS, MRD and MPP usage:

1. When writing a program in ladder format, programming tools automatically add all MPS, MRD and MPP instructions at the program conversion stage. If the generated instruction program is viewed, the MPS, MRD and MPP instructions are present.
2. When writing a program in instruction format, it is entirely down to the user to enter all relevant MPS, MRD and MPP instructions as required.

Mnemonic	Operands	Function	Program steps	Controllers			
OUT	Y, M, S	Output coil	1	PB	PC	PA	PH

Explanations:

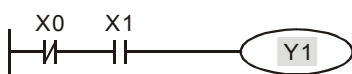
Output the logic calculation result before the OUT command to specific device.

Motion of coil contact

Operation result	OUT command		
	Coil	Contact	
		A contact (normal open)	B contact (normal close)
FALSE	OFF	Non-continuity	Continuity
TRUE	ON	Continuity	Non-continuity

Program Example:

Ladder diagram:



Command code:

Operation:

```

LDI     X0      ; Load contact B of X0
AND     X1      ; Connect to contact A of X1 in series
OUT    Y1      ; Drive Y1 coil

```

Mnemonic	Operands	Function	Program steps	Controllers			
				PB	PC	PA	PH
SET	Y, M, S	Latch (ON)	1				

Explanations:

When the SET command is driven, its specific device is set to be “ON,” which will keep “ON” whether the SET command is still driven. You can use the RST command to set the device to “OFF”.

Program Example:

Ladder Diagram:



Command Code: Operation:

LD X0 ; Load contact A of X0

ANI Y0 ; Connect to contact B of Y0 in series

SET Y1 ; Y1 latch (ON)

Mnemonic	Operands	Function	Program steps	Controllers			
				PB	PC	PA	PH
RST	Y, M, S, T, C, D, E, F	Clear the contact or the register	1				

Explanations:

When the RST command is driven, motion of its specific device is as follows:

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

If the RST command is not executed, status of specific device will not be changed.

Program Example:

Ladder diagram:



Command Code: Operation:

LD X0 Load contact A of X0

RST Y5 Clear contact Y5

Mnemonic	Operands	Function	Program steps	Controllers			
				PB	PC	PA	PH
MC/MCR	N0~N7	Master control Start/Reset	1				

Explanations:

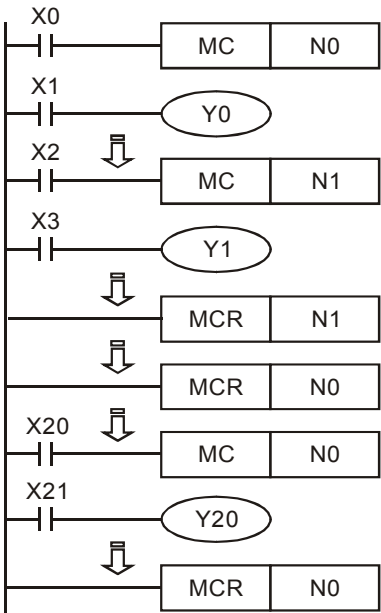
MC is the main-control start command. When the MC command is executed, the execution of instructions between MC and MCR will not be interrupted. When MC command is OFF, the motion of the instructions that between MC and MCR is described as follows:

General Timer	The timer value is set back to zero, the coil and the contact are both turned OFF
Timers for Subroutines and Interrupts	The timer value is set back to zero, the coil and the contact are both turned OFF
Accumulative timer	The timer value and the contact stay at their present condition
Counter	The counting value and the contact stay at their present condition
OUT	All turned OFF
SET/RST	Stay at present condition
Application instructions	All of them are not acted

1. MCR is the main-control ending command that is placed at the end of the main-control program.
2. An instruction of the MC-MCR main-control program supports the nest program structure, with 8 layers as its greatest. Please use the instructions in order from N0~ N7, and refer to the following:

Program Example:

Ladder Diagram:



Command Code	Operation:
LD X0 ; The control loop N0 active when X0 is	
MC N0 ; ON	
LD X1 ; Load A contact of X1	
OUT Y0 ; Drive Y0 coil	
:	
LD X2 ; The control loop N1 active when X2 is	
MC N1 ; ON	
LD X3 ; Load A contact of X3	
OUT Y1 ; Drive Y1 coil	
:	
MCR N1 ; The control loop N1 is end	
:	
MCR N0 ; The control loop N0 is end	
:	
LD X20 ; The control loop N0 active when X20 is	
MC N0 ; ON	
LD X21 ; Load A contact of X21	
OUT Y20 ; Drive Y20 coil	
:	
MCR N0 ; The control loop N0 is end	

Mnemonic	Function	Program steps	Controllers			
END	Program End	1	PB	PC	PA	PH

Explanations:

It needs to add the END command at the end of ladder diagram program or command program. ELC will scan from address 0 to END command, after executing it will return to address 0 to scan again.

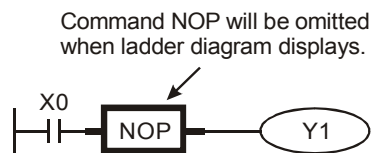
Mnemonic	Function	Program steps	Controllers			
NOP	No operation	1	PB	PC	PA	PH

Explanation:

This is a no-operation command and has no effect on the previous operation. NOP is used in the following cases: To delete a command without changing the number of steps. (Overwrite with NOP)

Program Example:

Ladder Diagram:



Command Code: Operation:

LD X0 ; Load A contact of X0

NOP ; No operation

OUT Y1 ; Drive Y1 coil

6.2 Pointers

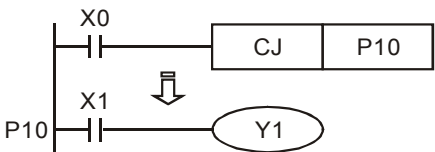
Mnemonic	Function	Program steps	Controllers			
P	Pointer (P0~P255)	1	PB	PC	PA	PH

Explanation:

Pointers are used with the jump commands (CJ, CALL) in two different ways as follows. But a number cannot be used repeatedly.

Program Example 1:

Ladder Diagram:



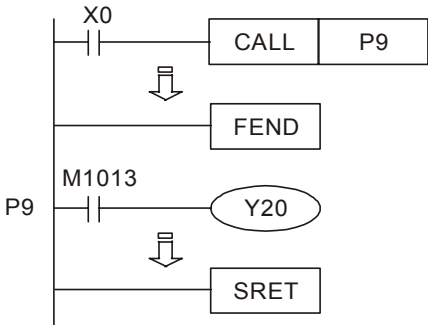
Command Code:

Operation:

```
LD      X0      ; Load A contact of X0
CJ      P10     ; Jump from command CJ to P10
:
P10
LD      X1      ; Load A contact of X1
OUT     Y1      ; Drive Y1 coil
```

Program Example 2:

Ladder Diagram:



Command Code:

Operation:

```
LD      X0      ; Call the Subroutine P9 when X0 is ON
CALL    P9
:
FEND
P9
LD      M1013
OUT     Y20
:
SRET
```

Available devices:

1. PB has 64 pointers; available from the range of P0 to P63.
2. PC, PA and PH have 256 pointers; available from the range of P0~P255.

6.3 Interrupt Pointers

Mnemonic	Function	Program steps	Controllers			
I	Interrupt program marker	1	PB	PC	PA	PH

Explanations:

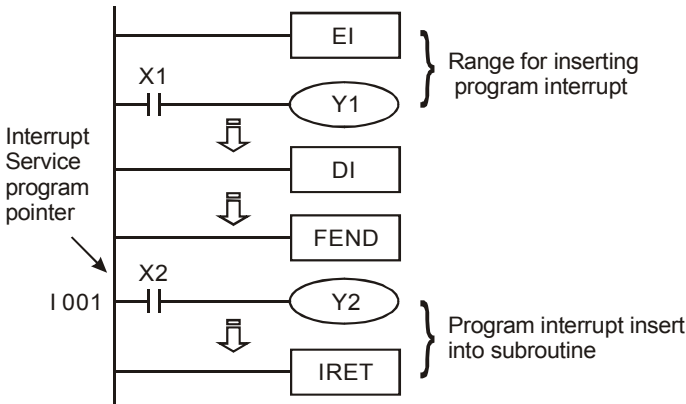
Interrupt programs should begin with interrupt pointer (I□□□) and ends with application command to be as interrupt end and return. Special numbering system based on interrupts device used and input triggering method.

Additional applied instructions:

It must use with application commands API 03 IRET, API 04 EI, API 05 DI.

Program Example:

Ladder Diagram:



Command Code: Operation:

EI ; Interrupt Enable
LD X1 ; Load A contact of X1
OUT Y1 ; Drive Y1 coil
:
DI ; Interrupt Disable
:
FEND ; Program end
I001 ; Insert interrupt point
LD X2 ; Load A contact of X2
OUT Y2 ; Drive Y2 coil
:
IRET ; Interrupt return

Input Interrupts:

- PB has 4 input interrupts: (I001, X0), (I101, X1), (I201, X2) and (I301, X3).
- PC, PA and PH have 6 input interrupts: (I001, X0), (I101, X1), (I201, X2), (I301, X3), (I401, X4) and (I501, X5).

Timer Interrupts:

- PB has 1 timer interrupt point: I6□□ (□□ = 10~99ms)
- PC, PA and PH have 2 timer interrupts: I6□□, I7□□. (□□ = 1~99ms)

Communication Interrupts:

PB/PC/PA/PH has 1 communication interrupt: I150

Counter Interrupts:

PC/PA/PH have six high-speed counter attained interrupt points:

I010 (use with C235, C241, C244, C246, C247, C249, C251, C252, C254)

I020 (use with C236, C243, C246, C247, C249, C251, C252, C254)

I030 (use with C237, C242)

I040 (use with C238, C245)

I050 (use with C239)

I060 (use with C240, C250)

Please see the following pages for more details on the interrupt functions.

Input interrupts – see page 6-41

Timer interrupts – see page 6-41

Communication interrupt – see page 6-41

Counter interrupts – see page 6-41

Enabling interrupts, API 04 EI – see page 6-40

Disabling interrupts, API 05 DI – see page 6-40

Interrupt return, API 03 IRET – see page 6-40

6.4 Basic Instructions (without API numbers)

Command Code	Function	Operands	Execution speed (us)		STEPS
			PB	PC/PA/PH	
LD	Load contact A	X, Y, M, S, T, C	5.6	4.6	1~3
LDI	Load contact B	X, Y, M, S, T, C	5.68	4.68	1~3
AND	Series connection with A contact	X, Y, M, S, T, C	4.8	3.8	1~3
ANI	Series connection with B contact	X, Y, M, S, T, C	4.88	3.88	1~3
OR	Parallel connection with A contact	X, Y, M, S, T, C	4.8	3.8	1~3
ORI	Parallel connection with B contact	X, Y, M, S, T, C	4.88	3.88	1~3
ANB	Series connects the circuit block Performed automatically by ELCSOft	None	4.4	3.4	1~3
ORB	Parallel connects the circuit block Performed automatically by ELCSOft	None	4.4	3.4	1~3
MPS	Save the operation result Performed automatically by ELCSOft	None	4.64	3.64	1~3
MRD	Read the operation result (the pointer not moving) Performed automatically by ELCSOft	None	4	3	1
MPP	Read the result Performed automatically by ELCSOft	None	4.4	3.4	1
OUT	Drive coil	Y, S, M	6.4	5.4	1~3
SET	Action latched (ON)	Y, S, M	5.04	4.04	1~3
RST	Clear the contacts or the registers	Y, M, S, T, C, D, E, F	7.6	6.6	3
MC	Connect the common series connection contacts	N0~N7	5.6	4.6	3
MCR	Disconnect the common series connection contacts	N0~N7	5.7	4.7	3
END	Program end	None	7.44	6.44	1
NOP	No function	None	3.52	2.52	1
STL	Step transition ladder start command	S	11.6	10.6	1
RET	Step transition ladder return command	None	7.04	6.04	1

6.5 Application Programming Instructions

- ELC instructions are provided a unique mnemonic name to make it easy to remember instructions. Most instructions are also given a number so the ELC knows how to execute the instruction. In the example below the numerical value given to the instruction is 00, also called the API (Application Programming Instruction). The mnemonic name is CJ and the description is Conditional Jump.

API	Mnemonic		Operands	Function	Controllers			
00		CJ	P	(S) Conditional Jump	PB	PC	PA	PH
OP	Range				Program Steps			
(S)	P0~P255				CJ, CJP: 3 steps			

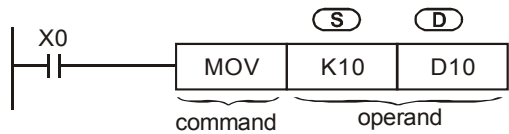
- The table will be found at the beginning of each new instruction description. The area identified as 'Operands' will list the various devices (operands) that can be used with the instruction. Various identification letters will be used to associate each operand with its function, i.e. D-destination, S-source, n, m-number of elements. Additional numeric suffixes will be attached if there are more than one operand with the same function.
- When using ELCSOft to create the application is not necessary to remember the API number of an instruction since ELCSOft uses drop down lists to select an instruction or there is a button on the toolbar for the instruction.
- When using the ELC-HHP it is important to have a API list since the ELC-HHP uses the API to insert instructions into the application.
- Not all instructions and conditions apply to all ELC's models. Applicable controllers are identified by the boxes in the bottom right hand corner of the table. For more detailed instruction variations a second indicator box is used to identify the availability of pulse, single (16 bit) word and double (32 bit (double) word format.

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

- No modification of the instruction mnemonic is required for 16 bit operation. However, pulse operation requires a 'P' to be added directly after the mnemonic while 32 bit operation requires a 'D' to be added before the mnemonic. This means that if an instruction was being used with both pulse and 32 bit operation it would look line...D***P where *** was the basic mnemonic.

Instruction Composition

Instructions consist of either just the instruction or the instruction followed by operand parameters



Command : Indicates the executive functions of the instruction

Operand : Indicates the device that calculates the operand

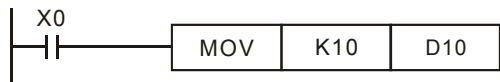
S	Source operand: if there is more than 1 source operand, then we use S ₁ , S ₂ ... to display.
D	Destination operand: if there is more than one operand, then we use D ₁ , D ₂ ... to display.
If the operand may only be represented as a constant K, H or register then we will use m, m ₁ , m ₂ , n, n ₁ , n ₂ to display.	

The Length of Operand (16-bit or 32-bit instruction)

The length of operand can be divided into two groups: 16-bit and 32-bit to process different length data.

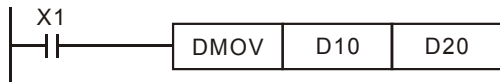
A "D" before a instruction separates 32-bit from 16-bit instructions.

16-bit MOV instruction



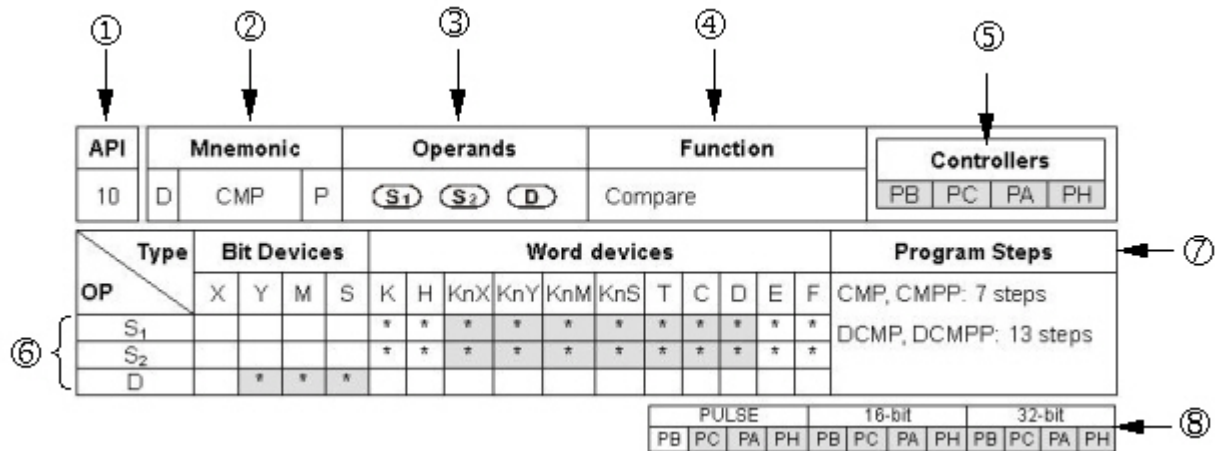
When X0=ON, K10 has been sent to D10.

32-bit DMOV instruction



When X1=ON, Data of (D11, D10) have been sent to (D21, D20)

Explanation of the format of application programming instruction

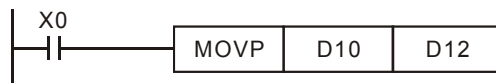


- ① API number for instruction
- ② The core mnemonic code of instruction
A "D" in this box means the instruction can be a 32 bit instruction if a "D" is added as a prefix
A "P" in this box indicates the instruction can be used as a pulse instruction
- ③ The operand format of the instruction
- ④ The description of the instruction
- ⑤ Applicable ELC models the instruction can be used with
- ⑥ A symbol "*" is the device can use the index register
A symbol "***" is given to device which can be used for this operand
- ⑦ Instruction memory steps
- ⑧ 16-bit instruction format, the name of continuous execution instruction and pulse instruction
32-bit instruction format, the name of continuous execution instruction and pulse instruction

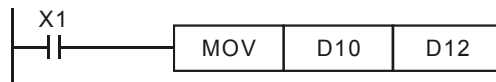
Note : it supports index E and F for those cells with shadow in above table. For example, device K of operand S₁ supports index E and F.

Continuous execution vs. Pulse execution

1. The execution type of instructions can be divided into two types: continuous execution instruction and pulse execution instruction. Execution time is shorter when instructions are not executed. Using the pulse execution instruction will reduce the scan time of the program.
2. The 'pulse' function allows the associated instruction to be activated on the rising edge of the control input. The instruction is driven ON for the duration of one program scan.
3. Thereafter, while the control input remains ON, the associated instruction is not active. To re-execute the instruction the control input must be turned from OFF to ON again.

Pulse execution instruction

When X0 goes from OFF→ON, the MOVP instruction will be executed one time and instruction cannot be re-executed again in the scan of program scan. This is pulse execution instruction.

Continuous execution instruction

When X1=ON, the MOV instruction can be re-executed again in every scan of program. This is called continuous execution instruction.

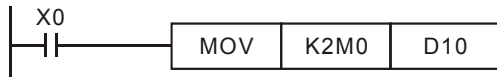
The above figures show that when X0, X1=OFF, the instruction will not be executed and the contents of the destination operand "D" will remain unchanged.

Operands

1. Bit device such as X, Y, M, S can be combined together to be defined as a WORD device. The bit device can serve as the word device (KnX, KnY, KnM, KnS) to store the numeric values to operate.
2. Data register D, Timer T, Counter C and Index Register E, F can all be assigned as operands.
3. A D register is usually a 16-bit register. It can also be assigning as a 32-bit register which consumes two D register with continuous numbers.
4. If the operand of a 32-bit instruction is assigned to D0, the 32-bit data register which is combined by D0 and D1 will be occupied. D1 is the upper 16-bit and D0 is the lower 16-bit.
5. If the 32-bit counters(C200~C255) are used as Data registers, one counter indicates a 32-bit length. Only instructions using 32-bit operands can be assigned.

Operand Data format

1. X, Y, M, S are only a single point ON/OFF, then they are defined as bit devices.
2. However, 16-bit (or 32-bit) device T, C, D, E, F are data registers and are defined as Word device.
3. If a Kn is placed in front of X, Y, M and S it will be defined as word device, whereas n=1 means 4-bit (each value of 1 is = to 4 bits). So 16-bit can be described from K1 to K4, and 32-bit can be described from K1 to K8. For example, K2M0 means use 8-bits from M0 to M7.



When X0=ON, move the contents of M0 to M7 to D10 segments 0 to 7, and segments 8 to 15 are set to 0.

Kx values

16-bit instruction		32-bit instruction	
Specified Number of Digits (16-bit instruction): K-32,768~K+32,767		Specified Number of Digits (32-bit instruction): K-2,147,483,648~K+2,147,483,647	
16-bit instruction: (K1~K4)		32-bit instruction: (K1~K8)	
K1 (4 points)	0~15	K1 (4 points)	0~15
K2 (8 points)	0~255	K2 (8 points)	0~255
K3 (12 points)	0~4,095	K3 (12 points)	0~4,095
K4 (16 points)	-32,768~+32,767	K4 (16 points)	0~65,535
		K5 (20 points)	0~1,048,575
		K6 (24 points)	0~167,772,165
		K7 (28 points)	0~268,435,455
		K8 (32 points)	-2,147,483,648~+2,147,483,647

Flags

1. General Flags

The following flags are available for the ELC:

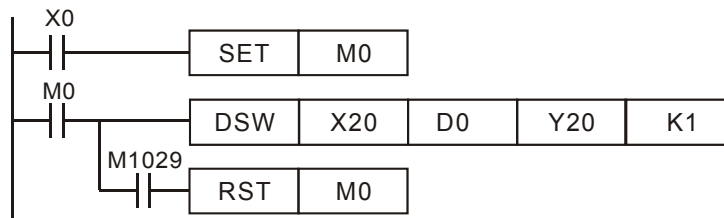
M1020 : Zero flag

M1021 : Borrow flag

M1022 : Carry flag

M1029 : Instruction execution completed flag

When executing an instruction, all flags will be turned to ON or OFF according to the operation result of the instruction. However, if the instruction is not executed, the ON/OFF state of the flags will remain unchanged.



When X0=ON, DSW instruction is activated.

When X0=OFF, wait for the program cycle of DSW instruction being completed, after M1029=ON, then M0 = OFF.

2. Error Operation Flags

If the combination of the instruction or the assigned operands result in an error, the error flags and any associated numbers in the following table will be shown during the execution of the application instructions.

M1067 D1067 D1069	When error operations occur, M1067=ON, D1067 will show the error number and D1069 will show the error address. If other errors occur, the contents of D1067 and D1069 will be refreshed. (when the error is reset, M1067=OFF)
M1068 D1068	When error operations occur, M1068=ON, D1068 will show the error address. If other errors occur, the contents of D1068 will not be refreshed, M1068 must use RST instruction to reset to OFF, otherwise the error will remain.

3. Flags to Extend Functions

Some instructions can extend their function by using some special flags.

Example: instruction RS can switch transmission mode 8-bit and 16-bit by using M1161.

Limited Use Instructions

Some instructions can be used unlimited times in the program, others can be used only a certain number of times in a program. These instructions can be modified by index registers to extend their functionality.

1. These instructions can be used once in a program:

PWM	PB Model	IST	PB/PC/PA/PH Models
SEGL	PB Model		

2. Only can be used twice in the program:

PLSY	PB Model	PLSR	PB Model
PR	PC/PA/PH Models		

3. Only can be used four times in the program:

HOUR	PC/PA/PH Models
------	-----------------

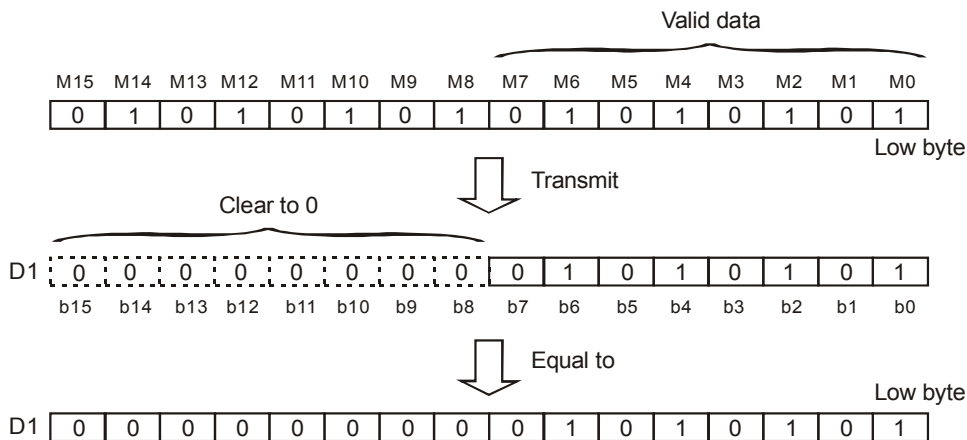
4. Only can be used eight times in the program:

TTMR	PC/PA/PH Models
------	-----------------

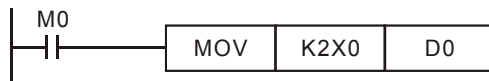
5. DHSCS and DHSCR, these instructions only can be executed simultaneously less than four times in the program for PB models.
6. DHSCS, DHSCR, DHSZ these instructions only can be executed simultaneously less than six times in the program of PC/PA/PH models.

Numeric Values

1. Devices such as X, Y, M, S are bit devices and there are only two states, ON and OFF. However, T, C, D, E, F are data registers and are word devices. Although bit device can only be a single point ON/OFF, they can also be used as numeric values (larger than bits) in the operands of instructions if the specified bit device is added front of the operand. The “specified bit device” is the “specified number of digit” and it would look like Kn where “n” can be a number from the range of 1 to 8.
2. 16-bit can be described from K1 to K4, and 32-bit can be described from K1 to K8. For example, K2M0 means there are 8-bit from M0 to M7.



3. Transmit K1M0, K2M0, K3M0 to 16-bit registers and the unused upper bit data are not transmitted. It's the same as sending K1M0, K2M0, K3M0, K4M0, K5M0, K6M0, K7M0 to 32-bit registers and the unused upper bit data are not transmitted.
4. The unused upper bit will be defined as 0 if the content of the operand assign K1 to K3 in a 16-bit operation or assign K4 to K7 in 32-bit operation. Therefore, the operation result is positive.



The BCD value combined by X0 to X7 will be converted to D0 as BIN value.

Assign Continuous Bit Numbers

As already explained, bit devices can be grouped into 4 bit units. The “n” in KnM0 defines the number of groups of 4 bits to be combined for data operation. K1 to K4 are allowed for 16-bit data operations but K1 to K8 are valid for 32-bit operations.

The bit device numbers are all of the above. To avoiding errors, please do not skip over the continuous numbers. Furthermore, if K4Y0 is used in 32-bit operation, the upper 16-bit is defined as 0. Therefore, it is recommended to use K8Y0 in 32bit operation.

Floating Point Operation

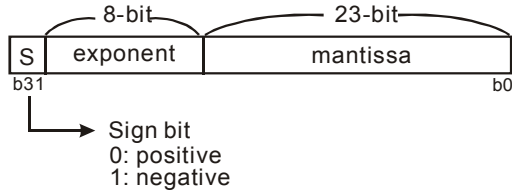
The internal operation of ELC usually is operated by “BIN integer” format. When performing integer division operation, the decimal point will be discarded. For example: $40 \div 3 = 13$, remainder is 1 and the decimal point will be discarded. But using floating point operation, the decimal point is used.

The application instructions related to floating point operation are shown in the following table.

FLT	DECOMP	DEZCP	DRAD
DDEG	DEBCD	DEBIN	DEADD
DESUB	DEMUL	DEDIV	DEXP
DLN	DLOG	DESQR	DPOW
INT	DSIN	DCOS	DTAN
DASIN	DACOS	DATAN	

Binary Floating Point

ELC uses the IEEE754 method to represent floating point numbers with 32-bit numbers:



$$\text{Equation } (-1)^S \times 2^{E-B} \times 1.M; B = 127$$

Therefore, the range of 32-bit floating is from $\pm 2^{-126}$ to $\pm 2^{+128}$, i.e. from $\pm 1.1755 \times 10^{-38}$ to $\pm 3.4028 \times 10^{+38}$.

Example 1: using 32-bit floating point to represent decimal number 23

Step 1: convert 23 to binary number: $23.0 = 10111$

Step 2: Normalizing the binary: $10111 = 1.0111 \times 2^4$, 0111 is mantissa and 4 is an exponent.

Step 3: get exponent: $\therefore E - B = 4 \rightarrow E - 127 = 4 \therefore E = 131 = 10000011_2$

Step 4: We can now combine the sign, exponent, and normalized mantissa into the binary IEEE short real representation.

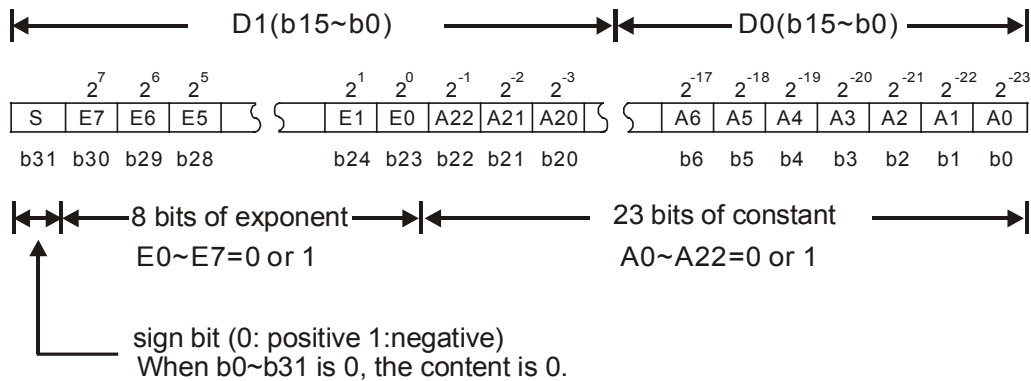
$0 \ 10000011 \ 01110000000000000000000_2 = 41B80000_{16}$

Example 2: using 32-bit floating point to represent decimal number -23

The conversion steps are the same as decimal number 23. Only need to modify sign bit from 0 to 1 to get value.

$1 \ 10000011 \ 01110000000000000000000_2 = C1B80000_{16}$

ELC also uses two registers with continuous number to store binary floating point. The following is the example that uses register (D1, D0) to store binary floating point.



Decimal Floating Point

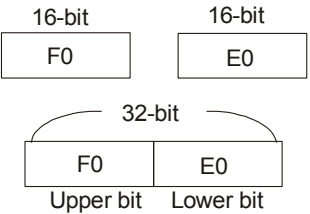
1. Binary floating point is not practical for some applications, therefore, binary floating point format can be converted to decimal floating point format for performing operation of decimal numbers. However, ELC will still use binary floating point to perform the operation of decimal numbers, it will perform much of the conversion for the user.
2. Decimal floating point is stored in the register with 2 continuous numbers. The register with small number stores constant and the register with larger number stores exponent.
For example, using register (D1, D0) to store a decimal floating point.
Decimal floating point = [constant D0] X 10^[exponent D1]
constant D0 = $\pm 1,000 \sim \pm 9,999$, exponent D1 = $-41 \sim +35$
the most significant bit of (D1, D0) is symbol bit.
Besides, constant 100 doesn't exist in D0 due to 100 will be shown with $1,000 \times 10^{-1}$.
The range of decimal number is from $\pm 1175 \times 10^{-41}$ to $\pm 3402 \times 10^{+35}$.
3. Decimal floating point can be used in the following instructions.
The conversion instruction for Binary floating point \rightarrow Decimal floating point (DEBCD)
The conversion instruction for Decimal floating point \rightarrow Binary floating point (DEBIN)
4. Zero flag (M1020), Borrow flag (M1021) and carry flag (M1022). The flags that corresponds to the floating instruction are:
 - a) Zero flag: when the result is 0, M1020=ON.
 - b) Borrow flag: when the result is least than the minimum unit, M1021=ON
 - c) Carry flag: when the absolute value of result exceeds usage range, M1022=ON

Index register E, F

The index registers are 16-bit registers. There are 2 devices for PB models (E and F), 8 devices for PC/PA models (E0~E3, F0~F3).

E and F are also 16-bit register just the same as general register.
It can be read /write.

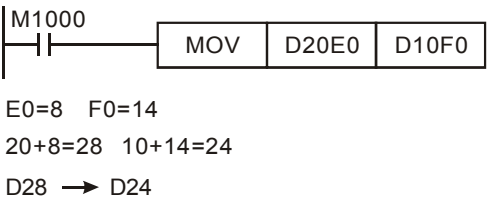
When using 32-bit index register, the combination of E, F are as follows. (E0, F0), (E1, F1), (E2, F2), (E3, F3).



If using a 32-bit register, you should specify E. In this condition, F will be covered by E and cannot be used anymore; otherwise the contents of E will become incorrect. (When ELC starts-up, it is recommended to use the MOVP instruction to clear the contents of F and reset it to 0)

As the right figure shows, the contents of operand will change according to the contents of E, F. this kind of modification is called “Index”.

For example, E0=8 and D20E0 represent constant D(20+8). If the contact is ON, register D28 will be transmitted to register D24.



6.6 Numerical List of Instructions

Loop Control

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
00	CJ	–	✓	Conditional jump	✓	✓	✓	3	–
01	CALL	–	✓	Call subroutine	✓	✓	✓	3	–
02	SRET	–	–	Subroutine return	✓	✓	✓	1	–
03	IRET	–	–	Interrupt return	✓	✓	✓	1	–
04	EI	–	–	Enable interrupt	✓	✓	✓	1	–
05	DI	–	–	Disable interrupt	✓	✓	✓	1	–
06	FEND	–	–	Terminate the main routine program	✓	✓	✓	1	–
07	WDT	–	✓	Reset the watchdog timer	✓	✓	✓	1	–
08	FOR	–	–	Nested loops begin	✓	✓	✓	3	–
09	NEXT	–	–	Nested loops end	✓	✓	✓	1	–

Transmission Comparison

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
10	CMP	DCMP	✓	Comparison output	✓	✓	✓	7	13
11	ZCP	DZCP	✓	Zone comparison	✓	✓	✓	9	17
12	MOV	DMOV	✓	Data Move	✓	✓	✓	5	9
13	SMOV	–	✓	Shift move	–	✓	✓	11	–
14	CML	DCML	✓	Counter transfer	✓	✓	✓	5	9
15	BMOV	–	✓	Block move	✓	✓	✓	7	–
16	FMOV	DFMOV	✓	Multiple devices movement	✓	✓	✓	7	13
17	XCH	DXCH	✓	Data exchange	✓	✓	✓	5	9
18	BCD	DBCD	✓	Convert BIN data into BCD	✓	✓	✓	5	9
19	BIN	DBIN	✓	Convert BCD data into BIN	✓	✓	✓	5	9

Four Fundamental Operations Arithmetic

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
20	ADD	DADD	✓	Addition of BIN data	✓	✓	✓	7	13
21	SUB	DSUB	✓	Subtraction of BIN data	✓	✓	✓	7	13
22	MUL	DMUL	✓	Multiplication of BIN data	✓	✓	✓	7	13
23	DIV	DDIV	✓	Division of BIN data	✓	✓	✓	7	13
24	INC	DINC	✓	Addition of 1	✓	✓	✓	3	5

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
25	DEC	DDEC	✓	Subtraction of 1	✓	✓	✓	3	5
26	WAND	DAND	✓	Logical product (AND) operation	✓	✓	✓	7	13
27	WOR	DOR	✓	Logical sum (OR) operation	✓	✓	✓	7	13
28	WXOR	DXOR	✓	Exclusive logical OR (XOR)	✓	✓	✓	7	13
29	NEG	DNEG	✓	Complement of 2	✓	✓	✓	3	5

Rotation and Displacement

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
30	ROR	DROR	✓	Rotate to the right	✓	✓	✓	5	9
31	ROL	DROL	✓	Rotate to the left	✓	✓	✓	5	9
32	RCR	DRCR	✓	Rotate right with carry	✓	✓	✓	5	9
33	RCL	DRCL	✓	Rotate left with carry	✓	✓	✓	5	9
34	SFTR	–	✓	Bit shift right	✓	✓	✓	9	–
35	SFTL	–	✓	Bit shift left	✓	✓	✓	9	–
36	WSFR	–	✓	Word shift right	–	✓	✓	9	–
37	WSFL	–	✓	Word shift left	–	✓	✓	9	–
38	SFWR	–	✓	Shift register write	–	✓	✓	7	–
39	SFRD	–	✓	Shift register read	–	✓	✓	7	–

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
40	ZRST	–	✓	Resets a range of device specified	✓	✓	✓	5	–
41	DECO	–	✓	8 → 256 bits decoder	✓	✓	✓	7	–

Data Operation

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
42	ENCO	–	✓	256 → 8 bits encoder	✓	✓	✓	7	–
43	SUM	DSUM	✓	Sum of ON bits	✓	✓	✓	5	9
44	BON	DBON	✓	Determine the ON bits	✓	✓	✓	7	13
45	MEAN	DMEAN	✓	Mean value	✓	✓	✓	7	13
46	ANS	–	–	Alarm device output	–	✓	✓	7	–

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
47	ANR	–	✓	Alarm device reset	–	✓	✓	1	–
48	SQR	DSQR	✓	Square root of BIN	✓	✓	✓	5	9
49	FLT	DFLT	✓	Floating point	✓	✓	✓	5	9

High Speed Processing

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
50	REF	–	✓	I/O refresh	✓	✓	✓	5	–
51	REFF	–	✓	Refresh and filter adjust	–	✓	✓	3	–
52	MTR	–	–	Matrix input	–	✓	✓	9	–
53	–	DHSCS	–	High speed counter SET	✓	✓	✓	–	13
54	–	DHSCR	–	High speed counter RESET	✓	✓	✓	–	13
55	–	DHSZ	–	HSC zone compare	–	✓	✓	–	17
56	SPD	–	–	Speed detection	✓	✓	✓	7	–
57	PLSY	DPLSY	–	Pulse output	✓	✓	✓	7	13
58	PWM	–	–	Pulse width modulation output	✓	✓	✓	7	–
59	PLSR	DPLSR	–	Ramp pulse output	✓	✓	✓	9	17

Convenience Instruction

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
60	IST	–	–	Manual/Auto control	✓	✓	✓	7	–
61	SER	DSER	✓	Multiple devices comparison	–	✓	✓	9	17
62	ABSD	DABSD	–	Absolute cam control	–	✓	✓	9	17
63	INCD	–	–	Relative cam control	–	✓	✓	9	–
64	TTMR	–	–	Alternate timer	–	✓	✓	5	–
65	STMR	–	–	Special timer	–	✓	✓	7	–
66	ALT	–	✓	ON/OFF alternate instruction	✓	✓	✓	3	–
67	RAMP	–	–	Ramp signal	–	✓	✓	9	–
69	SORT	–	–	Data sort	–	✓	✓	11	–

External I/O Display

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
70	TKY	DTKY	-	10-key keypad input	-	✓	✓	7	13
71	HKY	DHKY	-	16-key keypad input	-	✓	✓	9	17
72	DSW	-	-	Digital Switch input	-	✓	✓	9	-
73	SEGD	-	✓	Decode the 7-step display panel	✓	✓	✓	5	-
74	SEGL	-	-	7-step display scan output	✓	✓	✓	7	-
75	ARWS	-	-	Arrow keypad input	-	✓	✓	9	-
76	ASC	-	-	ASCII code conversion	-	✓	✓	11	-
77	PR	-	-	Print	-	✓	✓	5	-

Serial I/O

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
78	FROM	DFROM	✓	Read special module CR data	✓	✓	✓	9	17
79	TO	DTO	✓	Special module CR data write in	✓	✓	✓	9	17
80	RS	-	-	Serial data communication	✓	✓	✓	9	-
81	PRUN	DPRUN	✓	Octal number system transmission	-	✓	✓	5	9
82	ASCII	-	✓	Convert HEX to ASCII	✓	✓	✓	7	-
83	HEX	-	✓	Convert ASCII to HEX	✓	✓	✓	7	-
84	CCD	-	✓	Check sum	-	✓	✓	7	-
85	VRRD	-	✓	Volume read	-	✓	✓	5	-
86	VRSC	-	✓	Volume scale	-	✓	✓	5	-
87	ABS	DABS	✓	Absolute value	✓	✓	✓	3	5
88	PID	DPID	-	PID calculation	✓	✓	✓	9	17

Basic Instruction

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
89	PLS	-	-	Rising-edge output	✓	✓	✓	3	-
90	LDP	-	-	Rising-edge pulse	✓	✓	✓	3	-
91	LDF	-	-	Falling-edge pulse	✓	✓	✓	3	-
92	ANDP	-	-	Serial connection of rising-edge pulse	✓	✓	✓	3	-
93	ANDF	-	-	Serial connection falling-edge pulse	✓	✓	✓	3	-

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
94	ORP	–	–	Parallel connection of rising-edge pulse	✓	✓	✓	3	–
95	ORF	–	–	Parallel connection of falling-edge pulse	✓	✓	✓	3	–
96	TMR	–	–	Timer	✓	✓	✓	4	–
97	CNT	DCNT	–	Counter	✓	✓	✓	4	6
98	INV	–	–	Inverting operation	✓	✓	✓	1	–
99	PLF	–	–	Falling-edge output	✓	✓	✓	3	–

Communication Instruction

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
100	MODRD	–	–	MODBUS data Read	✓	✓	✓	7	–
101	MODWR	–	–	MODBUS data write in	✓	✓	✓	7	–
107	LRC	–	✓	LRC check sum	✓	✓	✓	7	–
108	CRC	–	✓	CRC check sum	✓	✓	✓	7	–

Floating Operation

API	Mnemonics		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
110	–	DECMP	✓	Floating point compare	✓	✓	✓	–	13
111	–	DEZCP	✓	Floating point zone compare	✓	✓	✓	–	17
116	–	DRAD	✓	Degree → Radian	–	✓	✓	–	9
117	–	DDEG	✓	Radian → Degree	–	✓	✓	–	9
118	–	DEBCD	✓	Float to scientific conversion	✓	✓	✓	–	9
119	–	DEBIN	✓	Scientific to float conversion	✓	✓	✓	–	9
120	–	DEADD	✓	Floating point addition	✓	✓	✓	–	13
121	–	DESUB	✓	Floating point subtraction	✓	✓	✓	–	13
122	–	DEMUL	✓	Floating point multiplication	✓	✓	✓	–	13
123	–	DEDIV	✓	Floating point division	✓	✓	✓	–	13
124	–	DEXP	✓	Floating point exponent operation	✓	✓	✓	–	9
125	–	DLN	✓	Floating natural logarithm operation	✓	✓	✓	–	9
126	–	DLOG	✓	Floating point logarithm operation	✓	✓	✓	–	13
127	–	DESQR	✓	Square root of binary floating point	✓	✓	✓	–	9
128	–	DPOW	✓	Floating point power operation	✓	✓	✓	–	13

API	Mnemonics		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
129	INT	DINT	✓	Floating point to integer	✓	✓	✓	5	9
130	–	DSIN	✓	Floating point Sine operation	✓	✓	✓	–	9
131	–	DCOS	✓	Floating point Cosine operation	✓	✓	✓	–	9
132	–	DTAN	✓	Floating point Tangent operation	✓	✓	✓	–	9
133	–	DASIN	✓	Floating point Arcsine operation	–	✓	✓	–	9
134	–	DACOS	✓	Floating point Arccosine operation	–	✓	✓	–	9
135	–	DATAN	✓	Floating point Arctangent operation	–	✓	✓	–	9

Additional Instruction

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
143	DELAY	–	✓	Delay instruction	–	✓	✓	3	–
144	GPWM	–	–	General pulse width modulation	–	✓	✓	7	–
145	FTC	–	–	Fuzzy temperature control	–	✓	✓	9	–
147	SWAP	DSWAP	✓	Swap high/low byte	✓	✓	✓	3	5
148	MEMR	DMEMR	✓	MEMORY read	–	✓	✓	7	13
149	MEMW	DMEMW	✓	MEMORY write in	–	✓	✓	7	13
150	MODRW	–	–	MODBUS data read/write in	✓	✓	✓	11	–
154	RAND	–	✓	Random value	–	✓	✓	7	–

Positioning Control

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
155	–	DABSR	–	ABS current value read	–	✓	✓	–	13
156	ZRN	DZRN	–	Zero point return	–	–	✓	9	17
158	DRVI	DDRVI	–	Relative positioning	–	–	✓	9	17
159	DRVA	DDRVA	–	Absolute positioning	–	–	✓	9	17

Perpetual Calendar

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
160	TCMP	–	✓	Calendar data comparison	–	✓	✓	11	–
161	TZCP	–	✓	Calendar data zone comparison	–	✓	✓	9	–

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
162	TADD	–	✓	Calendar data addition	–	✓	✓	7	–
163	TSUB	–	✓	Calendar data subtraction	–	✓	✓	7	–
166	TRD	–	✓	Calendar data read	–	✓	✓	3	–
167	TWR	–	✓	Calendar data write in	–	✓	✓	3	–
169	HOUR	DHOUR	–	Hour meter	–	✓	✓	7	13

Gray Code

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
170	GRY	DGRY	✓	Convert BIN to Gray code	–	✓	✓	5	9
171	GBIN	DGBIN	✓	Convert Gray code to BIN	–	✓	✓	5	9

Matrix Handling

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
180	MAND	–	✓	Matrix AND	–	✓	✓	9	–
181	MOR	–	✓	Matrix OR	–	✓	✓	9	–
182	MXOR	–	✓	Matrix XOR	–	✓	✓	9	–
183	MXNR	–	✓	Matrix XNR	–	✓	✓	9	–
184	MINV	–	✓	Matrix inverse	–	✓	✓	7	–
185	MCMP	–	✓	Matrix compare	–	✓	✓	9	–
186	MBRD	–	✓	Matrix bit read	–	✓	✓	7	–
187	MBWR	–	✓	Matrix bit write	–	✓	✓	7	–
188	MBS	–	✓	Matrix bit shift	–	✓	✓	7	–
189	MBR	–	✓	Matrix bit rotate	–	✓	✓	7	–
190	MBC	–	✓	Matrix bit state count	–	✓	✓	7	–

Contact Type Logic Operation

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
215	LD&	DLD&	–	$S_1 \& S_2$	–	✓	✓	5	9
216	LD	DLD	–	$S_1 S_2$	–	✓	✓	5	9
217	LD^	DLD^	–	$S_1 \wedge S_2$	–	✓	✓	5	9

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
218	AND&	DAND&	–	$S_1 \& S_2$	–	✓	✓	5	9
219	AND	DAND	–	$S_1 S_2$	–	✓	✓	5	9
220	AND^	DAND^	–	$S_1 \wedge S_2$	–	✓	✓	5	9
221	OR&	DOR&	–	$S_1 \& S_2$	–	✓	✓	5	9
222	OR	DOR	–	$S_1 S_2$	–	✓	✓	5	9
223	OR^	DOR^	–	$S_1 \wedge S_2$	–	✓	✓	5	9

Contact Type Compare Instruction

API	Mnemonic		PULSE	Function	Availability			STEPS	
	16 bits	32 bits			PB	PC/PA	PH	16-bit	32-bit
224	LD=	DLD=	–	$S_1 = S_2$	✓	✓	✓	5	9
225	LD>	DLD>	–	$S_1 > S_2$	✓	✓	✓	5	9
226	LD<	DLD<	–	$S_1 < S_2$	✓	✓	✓	5	9
228	LD<>	DLD<>	–	$S_1 \neq S_2$	✓	✓	✓	5	9
229	LD<=	DLD<=	–	$S_1 \leq S_2$	✓	✓	✓	5	9
230	LD>=	DLD>=	–	$S_1 \geq S_2$	✓	✓	✓	5	9
232	AND=	DAND=	–	$S_1 = S_2$	✓	✓	✓	5	9
233	AND>	DAND>	–	$S_1 > S_2$	✓	✓	✓	5	9
234	AND<	DAND<	–	$S_1 < S_2$	✓	✓	✓	5	9
236	AND<>	DAND<>	–	$S_1 \neq S_2$	✓	✓	✓	5	9
237	AND<=	DAND<=	–	$S_1 \leq S_2$	✓	✓	✓	5	9
238	AND>=	DAND>=	–	$S_1 \geq S_2$	✓	✓	✓	5	9
240	OR=	DOR=	–	$S_1 = S_2$	✓	✓	✓	5	9
241	OR>	DOR>	–	$S_1 > S_2$	✓	✓	✓	5	9
242	OR<	DOR<	–	$S_1 < S_2$	✓	✓	✓	5	9
244	OR<>	DOR<>	–	$S_1 \neq S_2$	✓	✓	✓	5	9
245	OR<=	DOR<=	–	$S_1 \leq S_2$	✓	✓	✓	5	9
246	OR>=	DOR>=	–	$S_1 \geq S_2$	✓	✓	✓	5	9

6.7 Detailed Instruction Explanation

API	Mnemonic			Operands	Function	Controllers			
00		CJ	P	<div>S</div>	Conditional Jump	PB	PC	PA	PH
OP	Range					Program Steps			
<div>S</div>	P0~P255					CJ, CJP: 3 steps			

Operands:

S: The destination pointer of the conditional jump, P can be modified by Index register E, F

1. PB has 64 pointers; available from the range of P0 to P63.
2. PC, PA and PH have 256 pointers; available from the range of P0~P255.

Explanations:

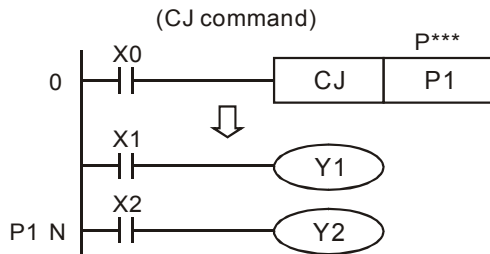
1. When the CJ instruction is active it forces the program to jump to an identified program marker. While the jump takes place the intervening program steps are skipped. This means they are not processed in any way. The resulting effect is to speed up the programs operational scan time.
2. When the destination of the pointer P is before CJ instruction, note that the WDT may cause an error if WDT exceeds its time.
3. What happens to each device type when executing the CJ instruction:
 - a) Y, M, S remains its previous state before the condition jump occurs.
 - b) General timers, accumulative timers and general counters will freeze their current values if they are skipped by a CJ instruction.
 - c) Timers for Subroutines and Interrupts are the only exception to this situation as they are processed independently of the main program.
 - d) High speed counters are the only exception to this situation as they are processed independently of the main program.
 - e) Application instructions are also skipped if they are programmed between the CJ instruction and the destination pointer. However, the SPD, PLSY, PWM, PLSR, DDRVI and DDRVA instructions will operate continuously if they were active before the CJ instruction was driven, otherwise they will be processed, i.e. skipped, as standard applied instructions.

Program Example 1:

When X0=ON the program will skip from address 0 to N (label P1) automatically and keep on executing.

Logic between address 0 and N will be skipped and will not be executed.

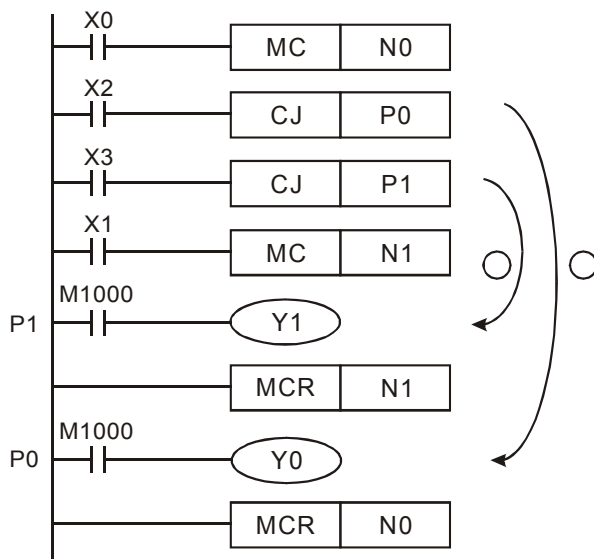
When X0=OFF, program flow will proceed with the rung immediately after the CJ instruction.



Program Example 2:

There are five conditions that the CJ instruction can be executed between the instructions MC and MCR.

1. Out of MC~MCR.
2. Valid in the loop P1 in the following chart.
3. In the same level N, inside of MC~MCR .
4. Inside of MC, out of MCR.
5. Jump from this MC~MCR to another MC~MCR.



Program Example 3:

The states of each device are shown in the following:

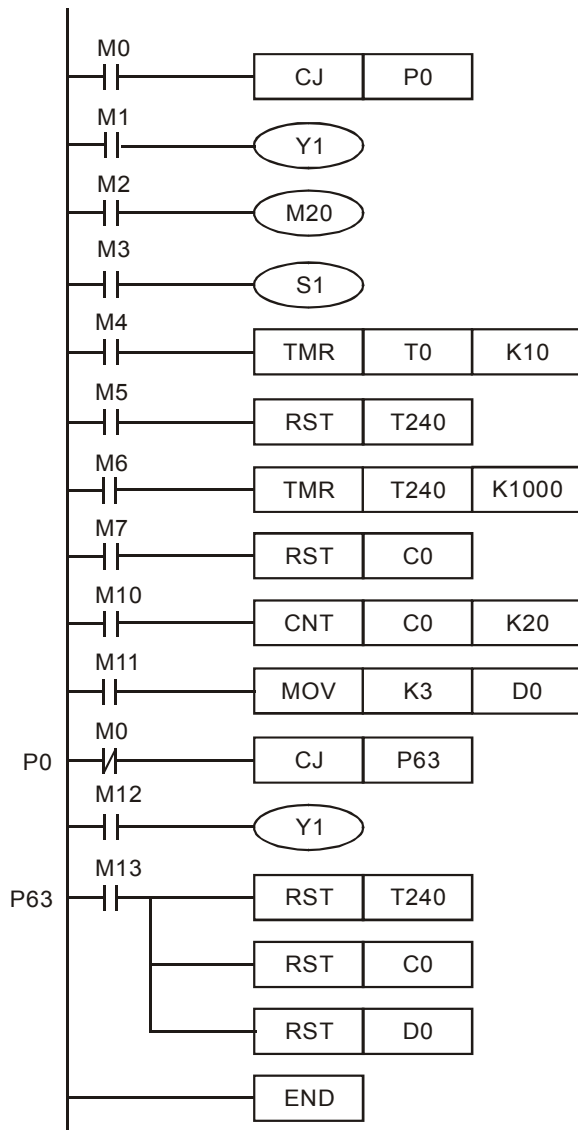
Device	Contact state before CJ execution	Contact state during CJ execution	Output coil state during CJ execution
Y, M, S	M1, M2, M3 OFF	M1, M2, M3 OFF→ON	Y1 (note1), M20, S1 OFF
	M1, M2, M3 ON	M1, M2, M3 ON→OFF	Y1 (note1), M20, S1 ON
10ms, 100ms Timer	M4 OFF	M4 OFF→ON	Timer is not activated
	M4 ON	M4 ON→OFF	Timer interrupt is latched. Keep on counting after M0 is OFF.
1ms,10ms, 100ms accumulative Timer (PC/PA/PH Series)	M6 OFF	M6 OFF→ON	Timer (T240) is not activated
	M6 ON	M6 ON→OFF	All accumulative timers will stop but latched once executing instruction CJ. When M0 is from ON→OFF, T240 will be unchanged.
C0~C234	M7, M10 OFF	M10 ON/OFF trigger	Counter does not count
	M7 OFF, M10 ON/OFF trigger	M10 ON/OFF trigger	The interrupt of counter latched. Keep on counting after M0 is OFF.
Application instruction	M11 OFF	M11 OFF→ON	Application instructions won't be executed.
	M11 ON	M11 ON→OFF	Do not execute the skipped application instruction but API 53~59, API 157~159 keep executing.

Note 1: Y1 is dual output. When M0 is OFF, it is controlled by M1. when M0 is ON, it is controlled by M12.

Note 2: When timer that subroutine used (T192~T199, for PC/PA/PH) executes CJ instruction, it will keep counting. After timer attains, output contact of timer will be ON.

Note 3: When high-speed counters (C235~C255) execute CJ instruction, it will keep counting and output point will also continue act.

Y1 is double or dual coil designation. When M0=OFF, it is controlled by M1. When M0=ON, it is controlled by M12.



API	Mnemonic			Operands	Function	Controllers			
01		CALL	P	S	Call Subroutine	PB	PC	PA	PH

OP	Valid Range	Program Steps
S	P0~P255	CALL, CALLP: 3 steps

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: The destination pointer of the call subroutine. P can be modified by Index register E, F. PB has 64 pointers; available from the range of P0 to P63. PC, PA and PH have 256 pointers; available from the range of P0~P255.

Explanations:

1. When the CALL instruction is active it forces the program to run the subroutine associated with the called pointer.
2. A CALL instruction must be used in conjunction with FEND (API 06) and SRET (API 02) instructions.
3. The program jumps to the subroutine pointer (located after an FEND instruction) and processes the contents until an SRET instruction is encountered. This forces the program flow back to the line of ladder immediately following the original CALL instruction.

Points to note:

1. Subroutine must be placed after the FEND instruction.
2. Subroutines must end with the SRET instruction.
3. CALL pointers and CJ instruction pointers are not allowed to coincide.
4. CALL instructions can call any other CALL subroutine any number of times.
5. Subroutines can be nested 5 levels including the initial CALL instruction. (If entering the six levels, the subroutine won't be executed.)

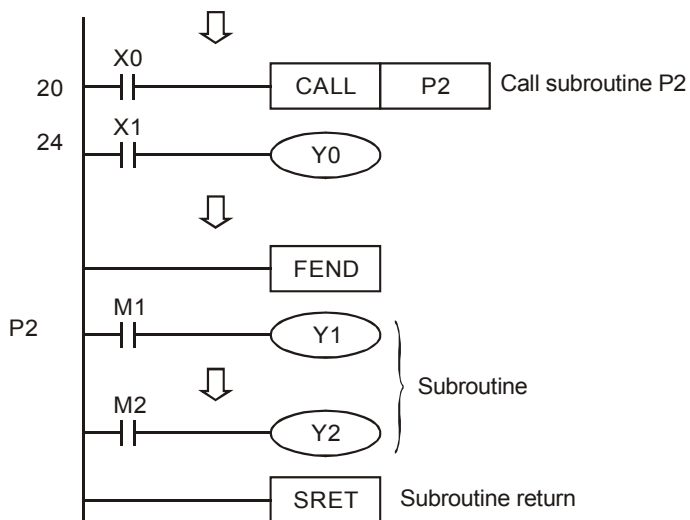
API	Mnemonic	Function	Controllers			
			PB	PC	PA	PH
02	SRET	Subroutine Return				
OP	Descriptions		Program Steps			
N/A	Automatically returns to the step immediately following the CALL instruction which activated the subroutine		SRET: 1 steps			

Explanations:

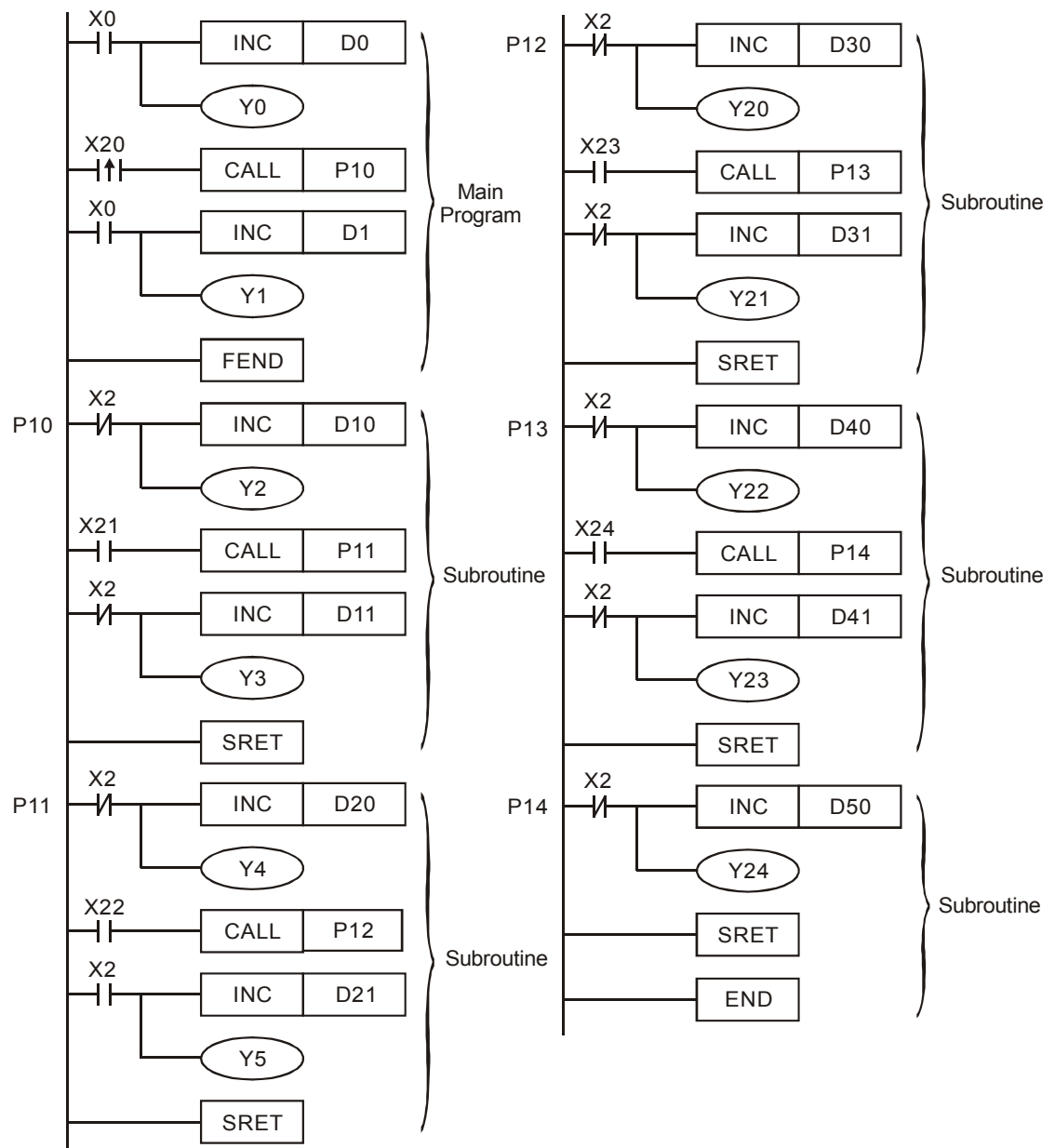
Indicates the end of subroutine program. The subroutine will return to main program and begin execution with the instruction after the CALL instruction.

Program Example 1:

When X0 = ON, the CALL instruction will jump to P2 and run the subroutine. With the execution of the SRET instruction, it will jump back to step 24 and continue execution.

**Program Example 2:**

1. When the rising-edge of X20 is triggered, CALL P10 instruction will transfer execution to subroutine P10.
2. When X21 is ON, execute CALL P11, jump to and run subroutine P11.
3. When X22 is ON, execute CALL P12, jump to and run subroutine P12.
4. When X23 is ON, execute CALL P13, jump to and run subroutine P13.
5. When X24 is ON, execute CALL P14, jump to and run subroutine P14. When the SRET instruction is reached, jump back to the last P*** subroutine and keep executing until the last SRET instruction is reached which will return execution back to the main program.



API	Mnemonic	Function	Controllers			
03	IRET	Interrupt Return	PB	PC	PA	PH
OP	Descriptions		Program Steps			
N/A	IRET ends the processing of an interrupt subroutine and returns execution back to the main program		SRET: 1 steps			

API	Mnemonic	Function	Controllers			
04	EI	Enable Interrupt	PB	PC	PA	PH
OP	Descriptions		Program Steps			
N/A	Enables Interrupts, explanation of this instruction also coincides with the explanation of the DI (disable interrupts instruction), see the DI instruction for more information. M1050~M1059, M1299		EI: 1 steps			

API	Mnemonic	Function	Controllers			
05	DI	Disable Interrupt	PB	PC	PA	PH
OP	Descriptions		Program Steps			
N/A	EI instruction enables ELC to accept interrupts; like Time interrupt or High-speed counter interrupt. Even in the interrupt allowed range when interrupting special the auxiliary relay M1050 to M1059·M1299, the corresponding interrupting request will not be activated.		DI: 1 steps			

Explanations:

1. Interrupt subroutines must be placed after the FEND instruction.
2. Other interrupts are not allowed during execution of a current interrupt routine.
3. Priority is given to the interrupt occurring first. If interrupts occur simultaneously, the interrupt with the lower pointer number will be given the higher priority.
4. Any interrupt request occurring between DI and EI instructions will not be executed immediately. The interrupt will be memorized and executed when the next EI instruction occurs.
5. Care should be used when using external interrupts and using the same inputs for high speed counter inputs.

6. During the execution of an interrupt routine, immediate I/O instruction can be performed by the REF instruction.

Points to note:

1. PB models interrupt pointers (I):

- a) External interrupts: (I001, X0), (I101, X1), (I201, X2), (I301, X3) 4 points.
- b) Time interrupts: I6□□, 1 point (□□ = 10~99, time base=1ms)
- c) Communication interrupt for specific characters received (I150)
- d) Flags:

Flag	Function
M1050	External interrupt, I 001 masked
M1051	External interrupt, I 101 masked
M1052	External interrupt, I 201 masked
M1053	External interrupt, I 301 masked

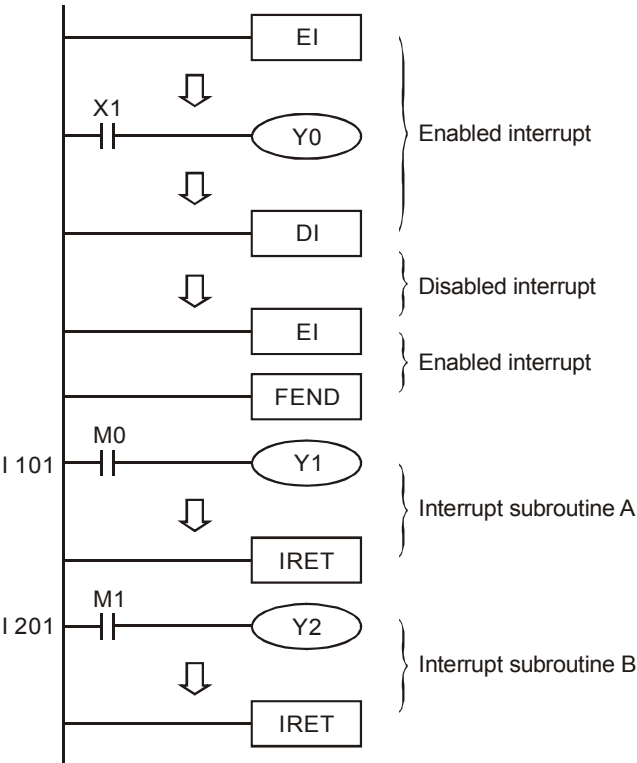
2. PC/PA/PH models Interrupt pointers (I):

- a) External interrupts: (I001, X0), (I101, X1), (I201, X2), (I301, X3), (I401, X4), (I501, X5) 6 points.
- b) Time interrupts: I6□□, I7□□ 2 points. (□□ = 1~99ms, time base=1ms)
- c) High-speed counter interrupts: I010, I020, I030, I040, I050, I060 6 points. (used with DHSCS instruction)
- d) Communication interrupt for specific characters received (I150)
- e) The priority of interrupt point I: high-speed counter interrupt, external interrupt, time interrupt and communication interrupt for specific characters received
- f) Flags:

Flag	Function
M1050	External interrupt, I 001 masked
M1051	External interrupt, I 101 masked
M1052	External interrupt, I 201 masked
M1053	External interrupt, I 301 masked
M1054	External interrupt, I 401 masked
M1055	External interrupt, I 501 masked
M1056	Timer interrupt, I6□□ masked
M1057	Timer interrupt, I7□□ masked
M1059	High-speed counter interrupt, I010~I060 masked
M1299	Communication interrupt, I150 masked

Program Example:

During the ELC operation, the program scans the instructions between EI and DI, if X1 or X2 are ON, the subroutine A or B will be interrupted. When IRET is reached, the main program will resume.



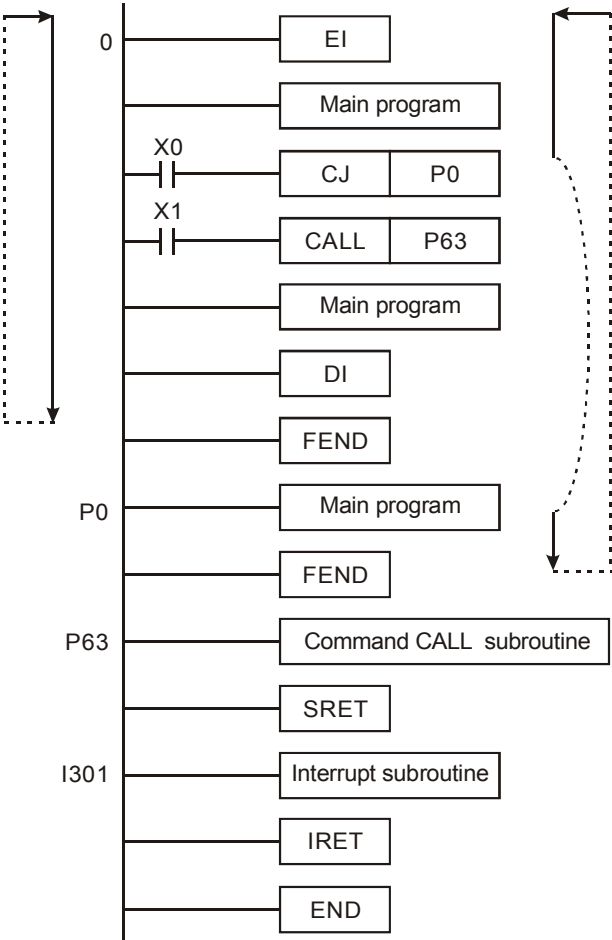
API	Mnemonic	Function	Controllers			
06	FEND	Terminate the Main Routine Program	PB	PC	PA	PH
OP	Descriptions		Program Steps			
N/A	Instruction driven by contact is not necessary.		FEND: 1 steps			

Explanations:

1. Use FEND when the application uses either CALL instructions or uses interrupts. If these CALL or interrupts are not used then use the END instruction to end the main program.
2. This instruction denotes the end of the main program. It has the same function as END instruction during ELC operation.
3. CALL subroutines must be placed after the FEND instruction. Each CALL subroutine must end with the SRET instruction.
4. Interrupt subroutines must be placed after the FEND instruction. Each interrupt subroutine must end with the IRET instruction.
5. When using the FEND instruction, an END instruction is still required, but should be placed as the last instruction after the main program and all subroutines.
6. If using several FEND instructions, place the CALL and interrupt subroutines between the last FEND and END instruction.
7. During execution of a subroutine, if an FEND instruction reached before SRET instruction, an error will occur.
8. During execution of a FOR instruction, if an FEND instruction reached before NEXT instruction, an error will occur.

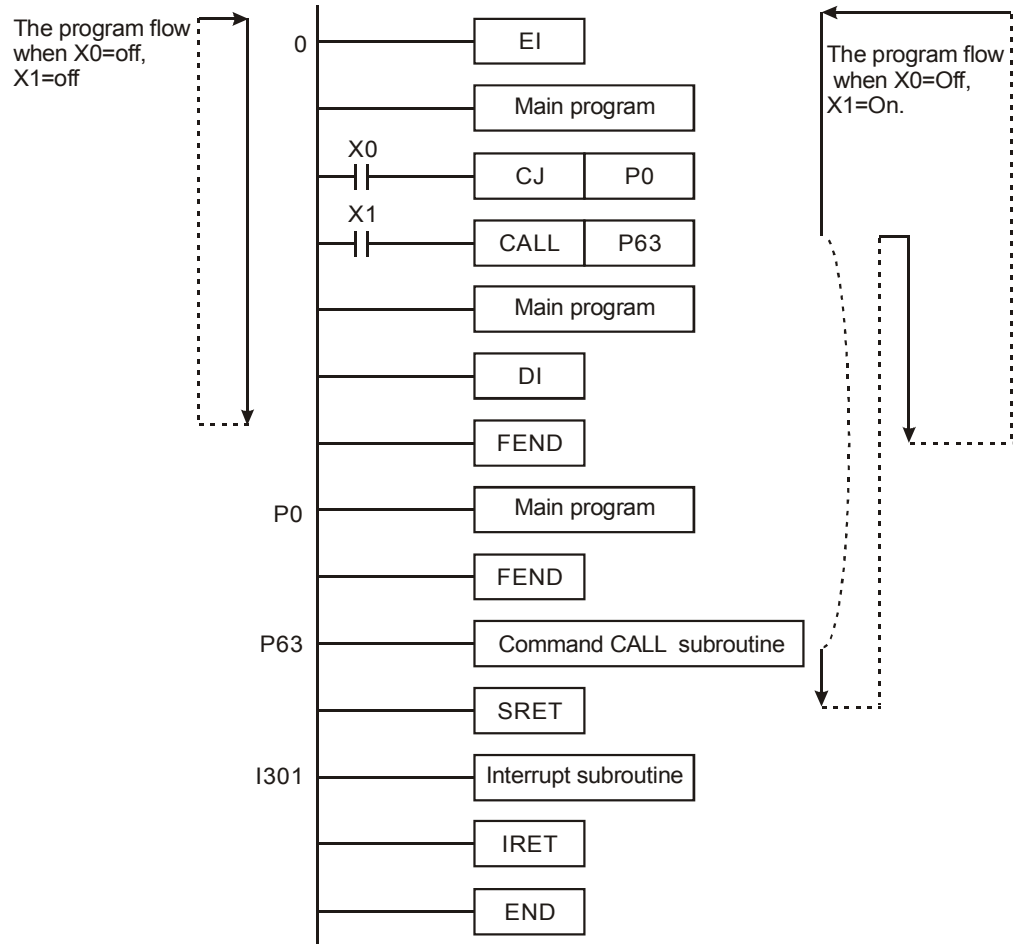
CJ Command Program Flow

The program flow when X0=off, X1=off



The program flow when X0=On
program jumps to P0

CALL Command Program Flow



API	Mnemonic			Function	Controllers			
					PB	PC	PA	PH
07		WDT	P	Reset the Watchdog Timer				

OP	Descriptions	Program Steps
N/A		WDT, WDTP: 1 steps

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Explanations:

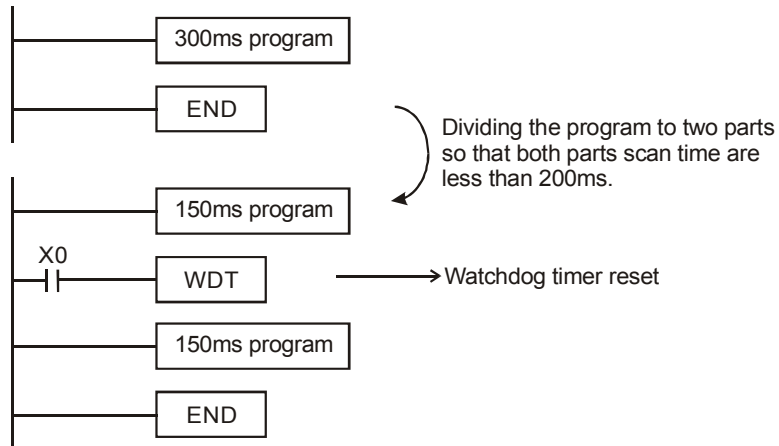
- The WDT instruction can be used to reset the Watch Dog Timer. If the ELC scan time (from step 0 to END or FEND instruction) is more than 200ms, the ERROR LED will flash. The user will have to turn the ELC OFF and then back ON to clear the fault. ELC will determine the status of RUN/STOP according to RUN/STOP switch.
- When to use WDT:
 - When error occur in ELC system.
 - When the scan time of the program exceeds the WDT value of D1000. It can be modified by using the following two methods.
 - Use WDT instruction
 - Use the set value of D1000 (default is 200ms) to change the watchdog time.

Points to note:

- When the WDT instruction is used it will operate on every program scan so long as its input condition has been made. To force the WDT instruction to operate for only ONE scan requires the user to program design. ELC users have the additional option of using the pulse(P) format of the WDT instruction, i.e. WDTP.
- The watchdog timer has a default setting of 200ms for ELC controllers. This time limit may be customized to users own requirement by editing the contents of data register D1000, the watchdog timer register.

Program Example:

If the program scan time is over 300ms, users can divide the program into 2 parts. Insert the WDT instruction in between, so both halves of the program's scan time will be less than 200ms.



API	Mnemonic		Operands		Function										Controllers				
08	FOR		<div>S</div>		Loop Begin										PB	PC	PA	PH	
<div>Type</div>		Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	FOR: 3 steps		
S						*	*	*	*	*	*	*	*	*	*	*			

Operands:

S: The number of times the loop will be repeated

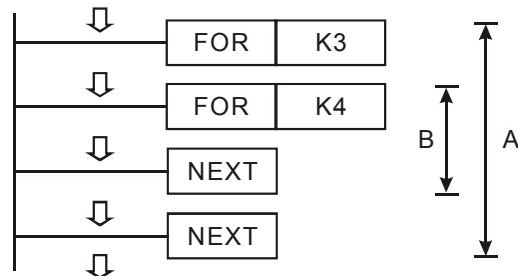
API	Mnemonic	Function	Controllers			
09	NEXT	Loop End	PB	PC	PA	PH
OP	Descriptions					Program Steps
N/A						NEXT: 1 steps

Explanations:

- FOR and NEXT instructions are used when loops are needed.
- “N” (number of times loop is repeated) may be within the range of K1 to K32767. If the range $N \leq K1$, N will always be K1.
- An error will occur in the following conditions:
 - NEXT instruction is before the FOR instruction.
 - A FOR instruction the doesn't have a NEXT instruction.
 - There is a NEXT instruction after the FEND or END instruction.
 - A mismatched number of FOR to NEXT instructions.
- The FOR to NEXT loop can be nested for five levels, if there are too many loops, the ELC scan time will increase and it may cause the watchdog timer to be activated and result in error. User can use WDT instruction to modify.

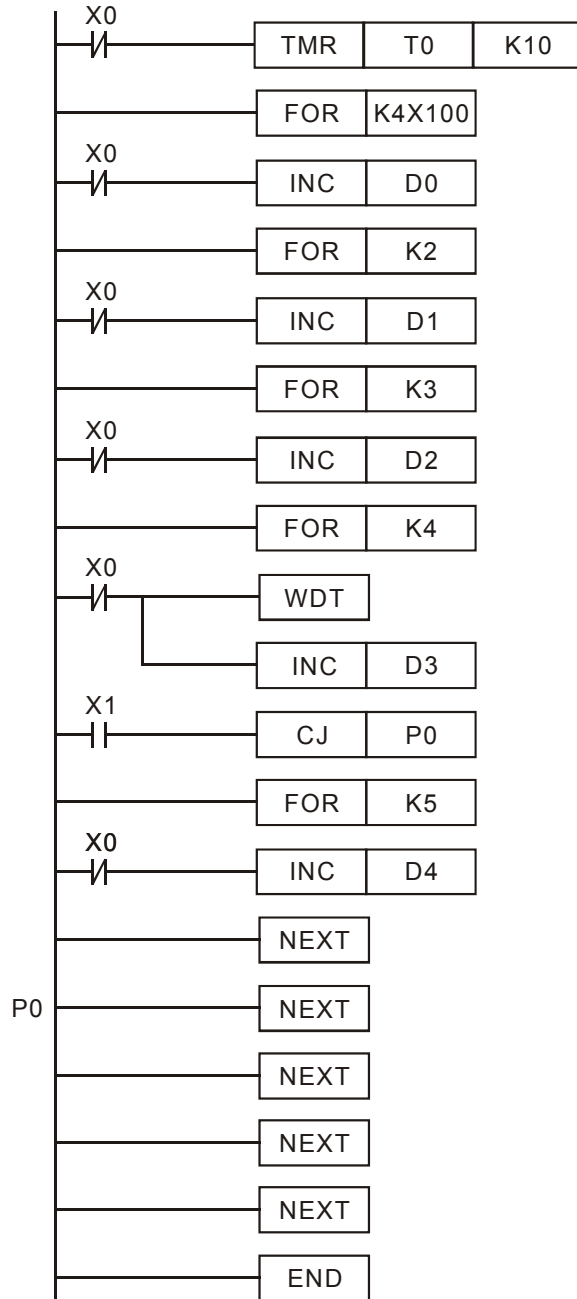
Program Example 1:

After loop A operates 3 times, the program after the NEXT instruction will resume. For every completed cycle of loop A, loop B will completely executed for 4 times, therefore, the total number of times that loop B operates will be $3 \times 4 = 12$ times.



Program Example 2:

When the FOR / NEXT instructions are not executed, a CJ instruction can be used to jump around the loop. When X1=ON, the CJ instruction will jump to P0 and not execute the most inner FOR / Next loop.



API	Mnemonic			Operands			Function			Controllers			
10	D	CMP	P	(S ₁)	(S ₂)	(D)	Compare			PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S ₁						*	*	*	*	*	*	*	*	*	*	*	CMP, CMPP: 7 steps DCMP, DCMPP: 13 steps
S ₂						*	*	*	*	*	*	*	*	*	*	*	
D			*	*	*												

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

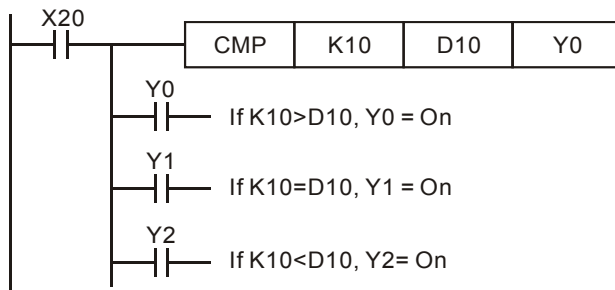
S₁: First comparison value S₂: Second comparison value D: Comparison result

Explanations:

- The contents of S₁ and S₂ are compared and D denotes the compare result.
- Operand D occupies 3 continuous devices.
- The values are binary values. If b15=1 in 16-bit instruction or b31=1 in 32-bit instruction, the comparison will regard the value as a negative binary value.
- D, D +1, D +2 hold the comparison results,
D = ON if S₁ > S₂,
D +1 = ON if S₁ = S₂
D +2 = ON if S₁ < S₂
- If operand S₁, S₂ use index register F, only a 16 bit compare is available.

Program Example:

- If D is set to Y0, then Y0, Y1, Y2 will display the results of the compare as shown below.
- When X20=ON, the CMP instruction is executed and one of Y0, Y1, Y2 will be ON. When X20=OFF, the CMP instruction is not executed and Y0, Y1, Y2 remain in their previous condition.



- Use RST or ZRST instruction to reset the comparison result.

API	Mnemonic			Operands				Function				Controllers			
11	D	ZCP	P	(S ₁)	(S ₂)	(S)	(D)	Zone Compare				PB	PC	PA	PH

Type	Bit Devices				Word devices											Program Steps			
OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ZCP, ZCPP: 9 steps DZCP, DZCPP: 17 steps			
S ₁					*	*	*	*	*	*	*	*	*	*	*				
S ₂					*	*	*	*	*	*	*	*	*	*	*				
S					*	*	*	*	*	*	*	*	*	*	*				
D		*	*	*															

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

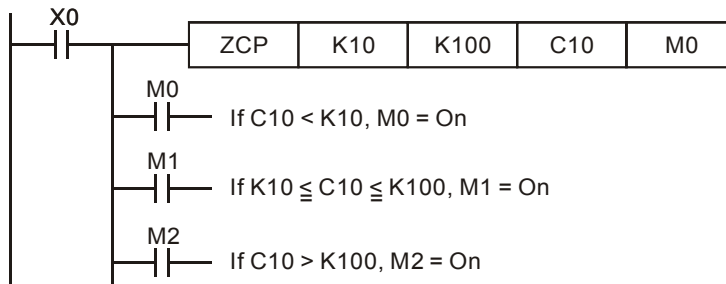
S₁: First comparison value (Minimum) **S₂**: Second comparison value (Maximum) **S**: Comparison value **D**: Comparison result

Explanations:

- S** is compared with its lower limit, **S₁** and its upper limit **S₂** and **D** denotes the compare result.
- The values are binary values. If b15=1 in 16-bit instruction or b31=1 in 32-bit instruction, the comparison will regard the value as a negative binary value.
- If operand **S₁**, **S₂**, **S** use index register F, only a 16 bit compare is available.
- Operand **S₁** should be less than Operand **S₂**,
Operand **D** occupies 3 continuous devices.

Program Example:

- If **D** is set to M0, then M0, M1, M2 will work as the program example as below.
- When X0=ON, ZCP instruction is driven and one of M0, M1, M2 is ON. When X0=OFF, ZCP instruction is not driven and M0, M1, M2 remain in the previous status.



- Use RST or ZRST instruction to reset the comparison result.

API	Mnemonic			Operands		Function										Controllers			
12	D	MOV	P	<div>S</div>	<div>D</div>	Move										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F			
S						*	*	*	*	*	*	*	*	*	*	*	MOV, MOV P: 5 steps		
D								*	*	*	*	*	*	*	*	*	DMOV, DMOV P: 9 steps		

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

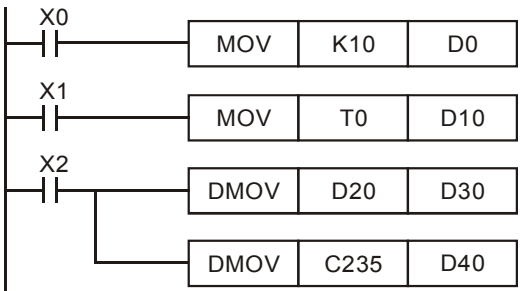
S: Data source **D:** Data destination

Explanations:

- When the MOV instruction is executed, the data in **S** is moved to **D** without any change to **S**. If the MOV instruction is not executed, the content of **D** will remain unchanged.
- If operand **S** and **D** use index register F, only a 16 bit compare is available

Program Example:

- MOV will move a 16-bit value from the source location to the destination.
- When X0=OFF, the content of D0 remains unchanged. If X0=ON, the data in K10 is moved to D0.
- When X1=OFF, the content of D10 remain unchanged. If X1=ON, the data of T0 is moved to D10 data register.
- DMOV will move a 32-bit value from the source location to the destination.
- When X2=OFF, the content of (D31, D30) and (D41, D40) remain unchanged. If X2=ON, the data of (D21, D20) is moved to (D31, D30) data register. Meanwhile, the data of C235 is moved to (D41, D40) data register.



API	Mnemonic			Operands					Function	Controllers			
13		SMOV	P	<div>S</div>	<div>m₁</div>	<div>m₂</div>	<div>D</div>	<div>n</div>	Shift Move	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

Type OP	Bit Devices				Word devices										Program Steps SMOV, SMOVP: 11 steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E		F
S							*	*	*	*	*	*	*	*		*
m ₁					*	*										
m ₂					*	*										
D								*	*	*	*	*	*	*		*
n					*	*										

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: Data source **m₁:** Source position (nibble) of the first digit to be moved **m₂:** Number of source digits (nibbles) to be moved **D:** Destination **n:** Destination position for the first digit (nibble)

Explanation:

1. BCD mode(M1168=OFF):

This mode of the SMOV operation allows BCD numbers to be manipulated in exactly the same way as the 'normal' SMOV manipulates decimal numbers, i.e. This instruction copies a specified number of digits from a 4 digit BCD source(**S**) and places them at a specified location within a destination (**D**) number (also a 4 digit BCD number).

2. BIN mode(M1168=ON) :

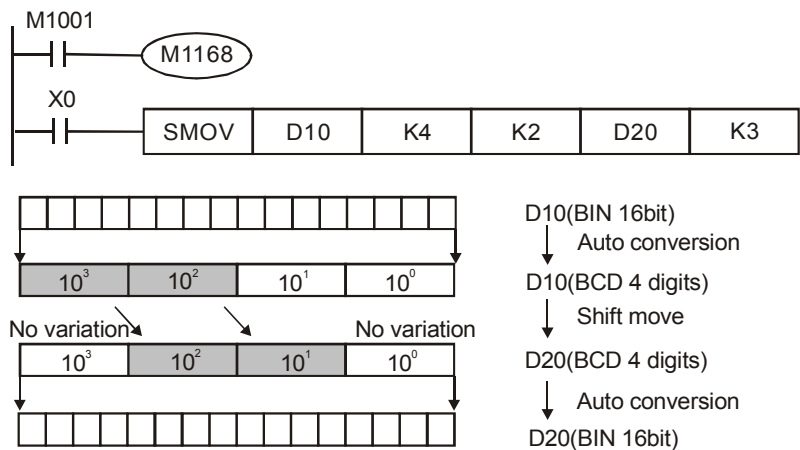
This instruction copies a specified number of digits from a 4 digit decimal source (**S**) and places them at a specified location within a destination (**D**) number (also a 4 digit decimal). The existing data in the destination is overwritten.

Points to note:

1. The range of **m₁**: 1 – 4
2. The range of **m₂**: 1 – **m₁** (cannot be great than **m₁**)
3. The range of **n**: **m₂** – 4 (cannot be less than **m₂**)

Program Example 1:

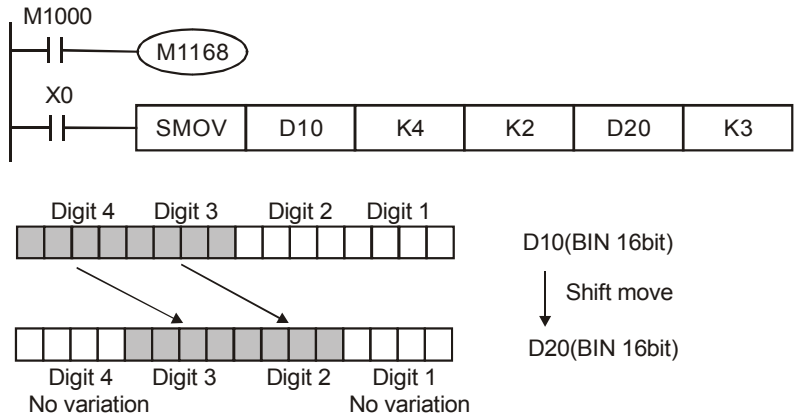
1. When M1168=OFF and X0=ON, two digits starting from the 4th digit (most significant digit MSD) of D10 (decimal number) and move them to the two digits from the 2nd digit (3rd MSD) of D20 (decimal number). The content of 103 and 100 of D20 remain unchanged after SMOV is executed.
2. If the source is not a valid BCD number an operation error will occur in ELC. The instruction will not be executed and M1067 and M1068 = ON, D1067 = error code H0E18.



3. If D10=H1234, D20=H5678 before execution, D10 remains unchange and D20=H5128 after execution.

Program Example 2:

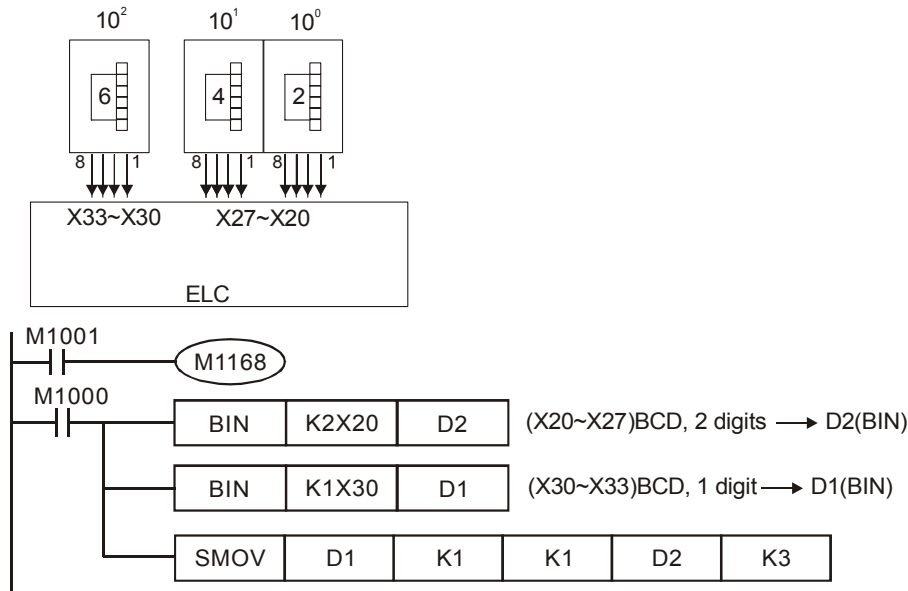
When M1168=ON and X0=ON, SMOV is executed and D10 and D20 will move nibble data in hex format.



Program Example 3:

Move the right second digit switch to the right second digit of D2 and move the left first digit switch to the right first digit of D1.

Use SMOV to move the first digit to the third digit of D2 and combine these two digit switches into one group.



API	Mnemonic			Operands		Function										Controllers			
14	D	CML	P	S	D	Compliment and Move										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	CML, CMLP: 5 steps			
	S					*	*	*	*	*	*	*	*	*	*	*	DCML, DCMLP: 9 steps		
D								*	*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

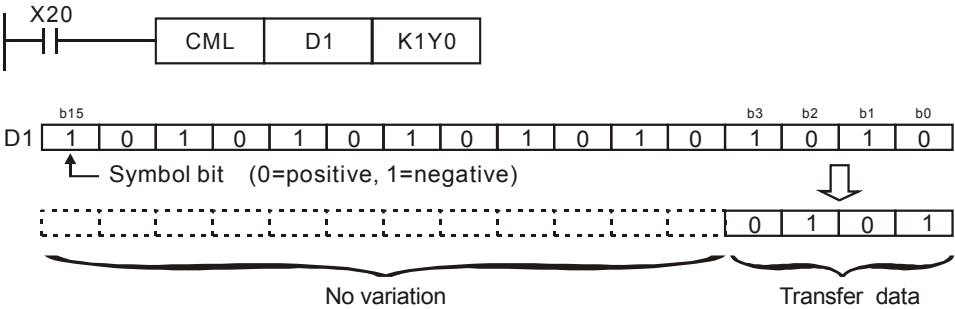
S: Data source D: Destination

Explanations:

- 1. Take the data in the source **S**, compliment (0→1, 1→0) it and move to the assigned destination **D**.
- 2. If operand **S** and **D** use index register F, only a 16 bit compare is available

Program Example 1:

When X20=ON, contents of D1, b0~b3, will be complimented and moved to K1Y0.



API	Mnemonic			Operands			Function								Controllers			
15		BMOV	P	(S)	(D)	(n)	Block Move								PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	BMOV, BMOV P: 7 steps			
S							*	*	*	*	*	*	*						
D								*	*	*	*	*	*						
n					*	*													

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

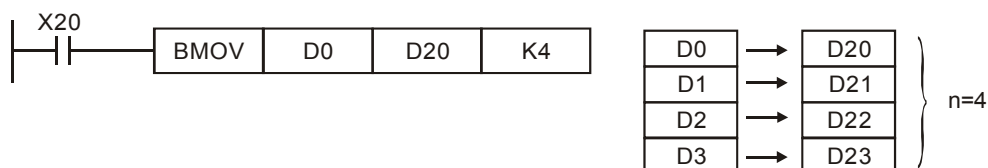
S: Source **D:** Destination **n:** Number of data to move

Explanations:

1. This instruction is used to move an assigned block of multiple data to a new destination. Move the contents of **S** through **S + n** to **D** through **D + n** registers. If the **n** assigned points exceed the range of the device, only those that are within the valid range will be moved.
2. The range of **n**=1 – 512.
3. PB models do not support KnX, KnY, KnM, KnS devices.

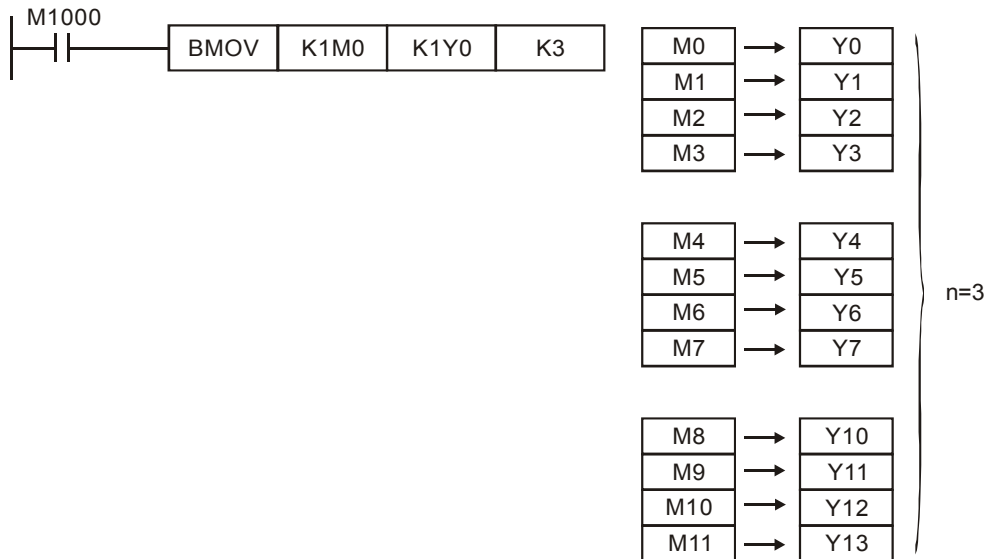
Program Example 1:

When X20=ON, move the contents of the four registers D0~D3 to their corresponding registers D20~D23.



Program Example 2:

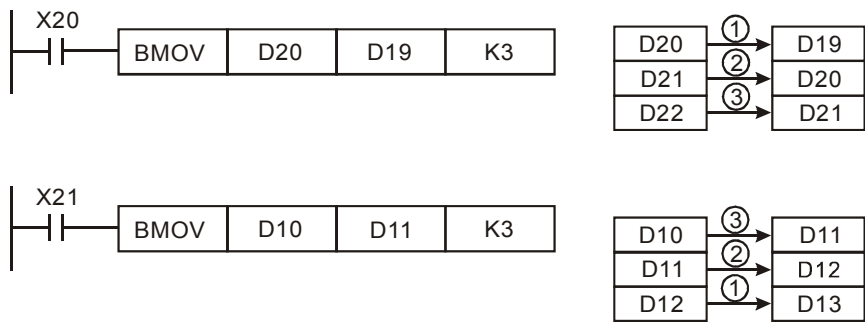
If BMOV is used to move bits, KnX, KnY, KnM, KnS, the digit numbers of **S** and **D** should be of the same data type.



Program Example 3:

The BMOV instruction will operate differently, automatically, to prevent errors when **S** and **D** coincide.

1. When **S** > **D**, the BMOV instruction is processed in the order ①→②→③.
2. When **S** < **D**, the BMOV instruction is processed in the order ③→②→①, then D11~D13 all equal to D10.



API	Mnemonic			Operands			Function								Controllers			
16	D	FMOV	P	S	D	n	Fill and Move								PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	FMOV, FMOV P: 7 steps DFMOV, DFMOV P: 13 steps			
S					*	*	*	*	*	*	*	*	*	*	*				
D								*	*	*	*	*	*						
n					*	*													

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

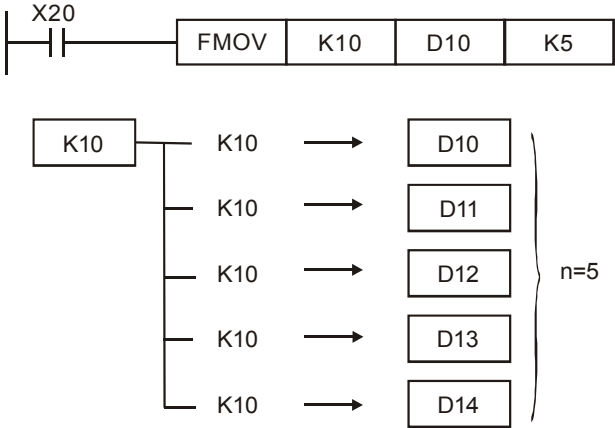
S: Source D: Destination n: Number of data to move

Explanations:

- 1. This instruction is used to move an assigned block of multiple data to a new destination. Move the contents of S through S + n to D through D + n registers. If the n -assigned points exceed the range of the device, only those that are within the valid range will be moved.
- 2. PB models do not support KnX, KnY, KnM, and KnS devices.
- 3. If operand S use index register F, only a 16 bit compare is available
- 4. The range of n: 1~ 512(16-bit instruction), 1~ 256 (32-bit instruction)

Program Example:

When X20=ON, move constant K10 to the continuous five registers (D10~D14) starting from D10.



API	Mnemonic			Operands	Function										Controllers			
17	D	XCH	P	D₁ D₂	Exchange										PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	XCH, XCHP: 5 steps		
D ₁								*	*	*	*	*	*	*	*	DXCH, DXCHP: 9 steps		
D ₂								*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

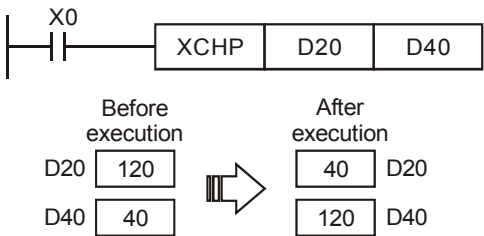
D₁: First exchange device **D₂**: Second exchange device

Explanations:

1. Exchange the contents of **D₁** and **D₂** with each other.
2. This instruction is best used as pulse execution (XCHP).
3. If operand **D1** and **D2** use index register F, only a 16-bit compare is available.

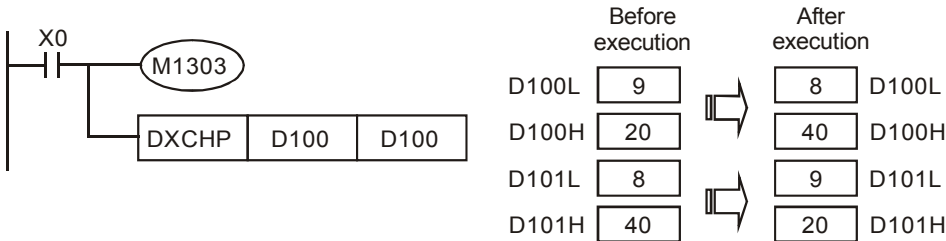
Program Example:

When X0=OFF→ON, the contents of D20 and D40 exchange with each other.



Points to note:

1. When **D₁** and **D₂** are the same, and M1303=ON, the upper and lower 16-bits will be exchanged.
PB is not support.
2. When X0=ON and M1303=ON, the upper and lower 16-bit contents of D100, D101 will exchange.



API	Mnemonic			Operands		Function										Controllers			
18	D	BCD	P	<div>S</div>	<div>D</div>	Convert BIN to BCD										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	BCD, BCDP: 5 steps			
	S						*	*	*	*	*	*	*	*	*				
D								*	*	*	*	*	*	*	*	DBCD, DBCDP: 9 steps			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

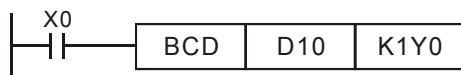
S: Source **D:** Converted result

Explanations:

- Convert BIN data (0 to 9999) of **S** into BCD and transfer the result to **D**.
- If the BCD conversion result is outside the valid range of 0 to 9999 (16-bit) or 0 to 99,999,999 (32-bit), an operation error occurs, the error flag M1067 and M1068 =ON, and D1067 will hold error code H0E18.
- If operand **S** and **D** use index register F, only a 16-bit compare is available.
- Flags: M1067 (operation error), M1068 (operation error), D1067 (error code)

Program Example:

- When X0=ON, the binary data D10 is converted into BCD number, and stored at K1Y0 (Y0~Y3).



- When D10=001E(Hex)=0030(decimal), the result will be Y0~Y3=0000(BIN).

API	Mnemonic			Operands		Function										Controllers			
19	D	BIN	P	S	D	Convert BCD to BIN										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F			
S								*	*	*	*	*	*	*	*	*	BIN, BINP: 5 steps		
D									*	*	*	*	*	*	*	*	DBIN, DBINP: 9 steps		

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

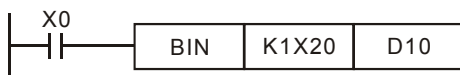
S: Source **D:** Converted result

Explanations:

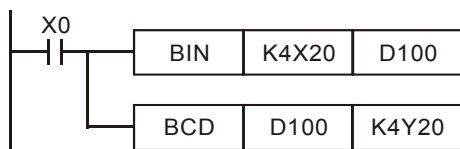
- Converts BCD data (0 to 9,999) of **S** into BIN and transfer the result to **D**.
- The valid range of source **S**: BCD (0 to 9,999), DBCD (0 to 99,999,999)
- If the content of **S** is not a valid BCD value, an operation error will occur, error flags M1067 and M1068 =ON, and D1067 holds error code H0E18.
- If operand S and D use index register F, only a 16-bit compare is available.
- Flags: M1067 (operation error), M1068 (operation error), D1067 (error code)

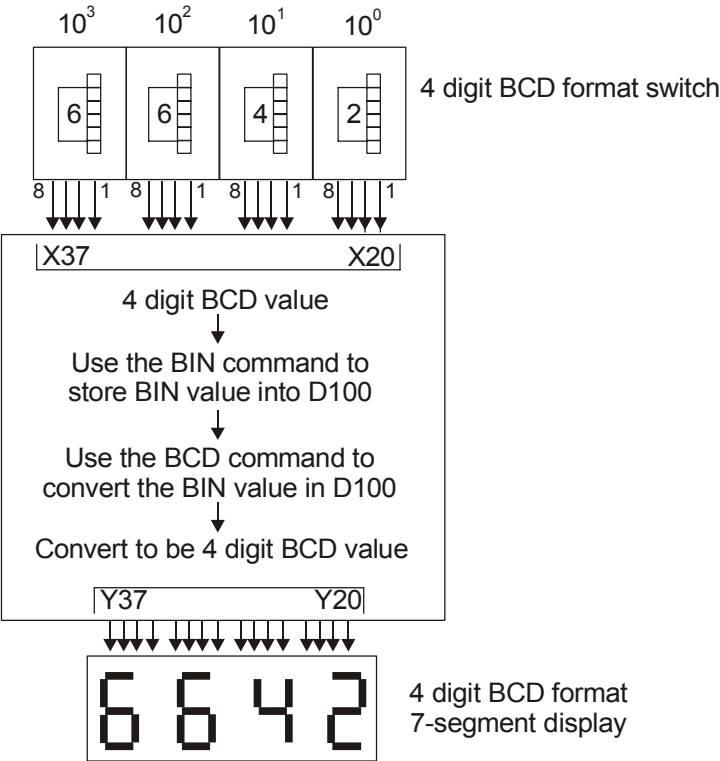
Program Example:

When X0=ON, the BCD data K1X20 is converted to BIN data, and result stored at D10.



- The BIN instruction is used to convert the source data into BIN data and store in the ELC, when ELC reads a BCD value i.e. digit switch connected externally.
- When X0=ON, convert K4X20 (BCD data) into BIN data and transmit it to D100. Then, convert BIN data of D100 into BCD data and transmit it to K4Y20.





API	Mnemonic			Operands			Function								Controllers			
20	D	ADD	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Addition								PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ADD, ADDP: 7 steps DADD, DADDP: 13 steps		
S ₁					*	*	*	*	*	*	*	*	*	*	*			
S ₂					*	*	*	*	*	*	*	*	*	*	*			
D								*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Augend **S₂**: Addend **D**: Addition result

Explanations:

1. The data contained within the source devices (**S₁**, **S₂**) is combined and the total is stored at the specified destination device (**D**).
2. The most significant bit are the sign bit. 0 indicates positive and 1 indicates negative. All calculation are algebraically processed, i.e. $3 + (-9) = -6$.
3. If operands **S₁**, **S₂**, **D** use index F, then only 16-bit instruction is available.
4. Flags: M1020 (Zero flag), M1021 (Borrow flag), M1022 (Carry flag)

Program Example 1:

16-bit instruction:

When X0 = ON, the data in D0 and data in D10 are added together and the result stored in D20. D0 and D10 are unchanged.

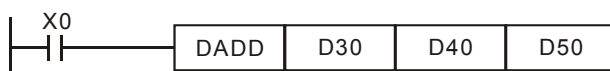


$$(D0) + (D10) = (D20)$$

Program Example 2:

32-bit instruction:

When X0 = ON, the data in (D31, D30) and data in (D41, D40) are added together and the result stored in (D51, D50). (D31, D30) and (D41, D40) are unchanged. (D30, D40, D50 is the lower 16-bit data, while D31, D41, D51 is the higher 16-bit data)



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

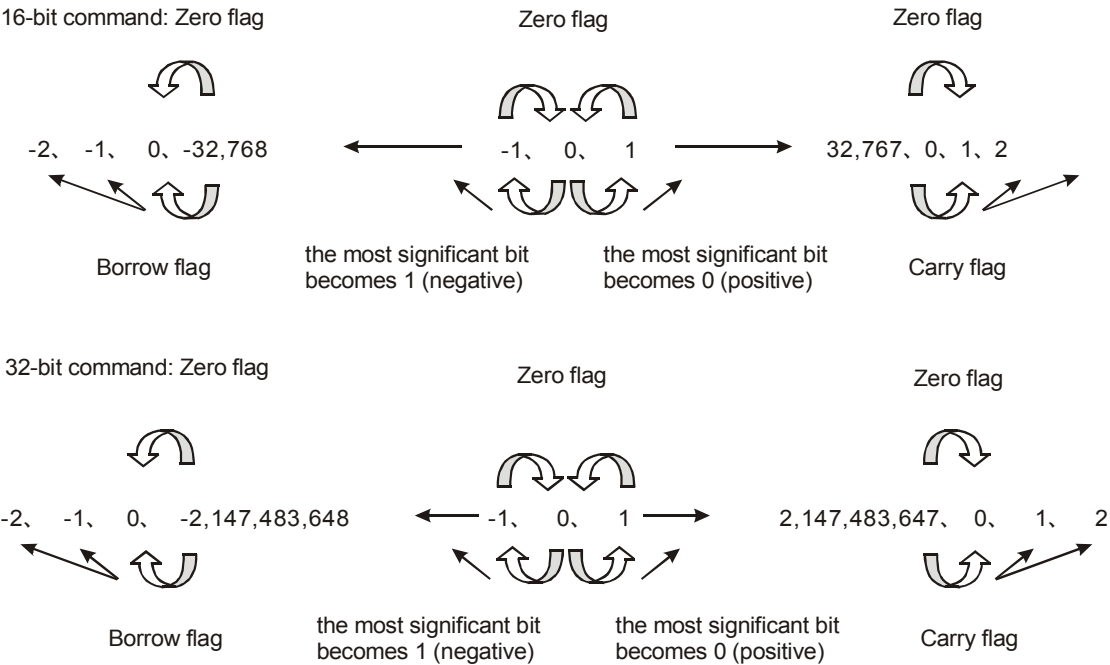
Flag operations:

16-bit instruction:

1. If the operation result is “0”, then the Zero flag, M1020 is set to ON.
2. If the operation result exceeds -32,768, the borrow flag, M1021 is set to ON.
3. If the operation result exceeds 32,767, the carry flag, M1022 is set to ON.

32-bit instruction:

1. If the operation result is “0”, then the Zero flag, M1020 is set to ON.
2. If the operation result exceeds -2,147,483,648, the borrow flag, M1021 is set to ON.
3. If the operation result exceeds 2,147,483,647, the carry flag, M1022 is set to ON



API	Mnemonic			Operands			Function								Controllers			
21	D	SUB	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Subtraction								PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F			
S ₁						*	*	*	*	*	*	*	*	*	*	*	SUB, SUBP: 7 steps DSUB, DSUBP: 13 steps		
S ₂						*	*	*	*	*	*	*	*	*	*	*			
D									*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S_1 : Minuend S_2 : Subtrahend D: Subtraction result

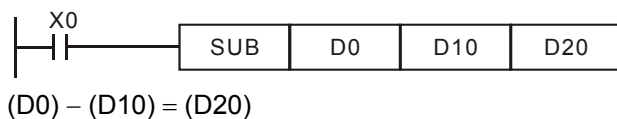
Explanations:

- The data contained within the source device, S_2 is subtracted from the contents of source device S_1 . The result or remainder of this calculation is stored in the destination device D.
- The most significant bit are the sign. 0 indicates positive and 1 indicates negative. All calculation are algebraically processed.
- If operand S_1 , S_2 , D use index F, then only 16-bit instruction is available.
- Flags: M1020 (Zero flag), M1021 (Borrow flag), M1022 (Carry flag). The flag operations ADD instruction can also be similarly applied to the subtract instruction.

Program Example 1:

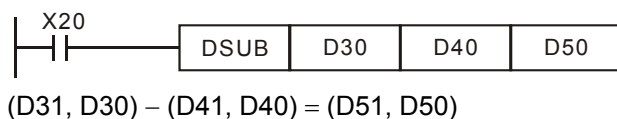
16-bit instruction:

When X0 = ON, the data in D10 is subtracted from the data in D0 and the result is placed in D20.

**Program Example 2:**

32-bit instruction:

When X20 = ON, the data in (D41, D40) is subtracted from the data in (D31, D30) and the result is placed in (D51, D50). (D30, D40, D50 is the lower 16-bit data, and D31, D41, D51 is the higher 16-bit data)



API	Mnemonic			Operands			Function								Controllers			
22	D	MUL	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Multiplication								PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MUL, DMULP: 7 steps			
	S ₁					*	*	*	*	*	*	*	*	*		DMUL, DMULP: 13 steps			
	S ₂					*	*	*	*	*	*	*	*	*					
	D							*	*	*	*	*	*	*					

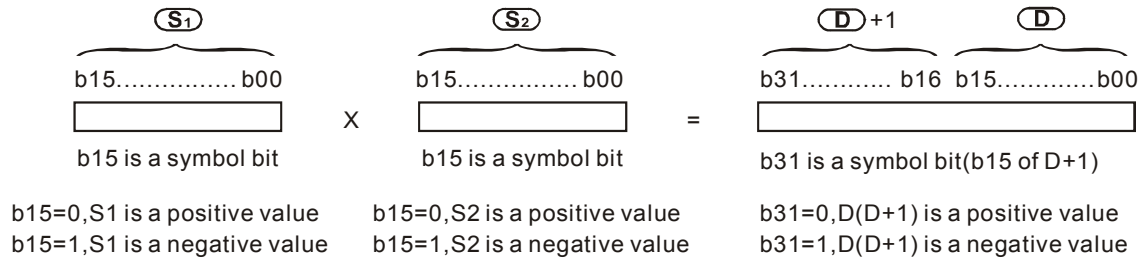
PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Multiplicand S₂: Multiplier D: Multiplication result

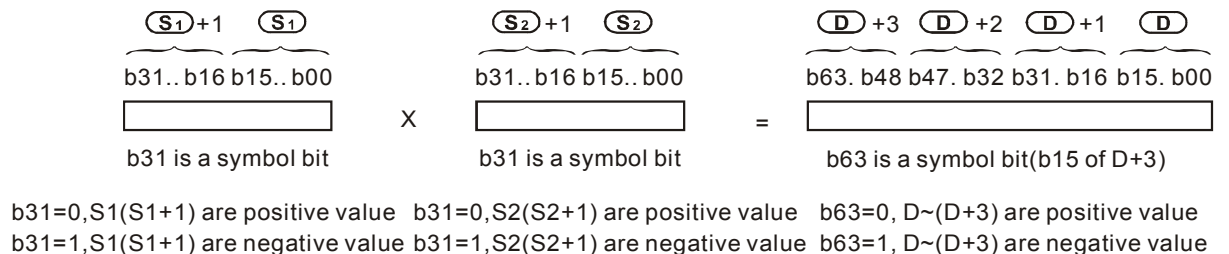
Explanations:

- The contents of the two source devices (S₁, S₂) are multiplied together and the result is stored at the destination device (D). Note the normal rules of algebra apply.
- If operands S₁, S₂ use index F, then only 16-bit instruction is available.
- If operand D use index E, then only 16-bit instruction is available.
- 16-bit instruction



When D is a bit device, it can specify K1~K4 to produce a 16-bit result and occupies 2 of continuous 16-bit.

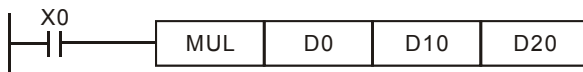
- 32-bit instruction



When D is a bit device, it can specify K1~K8 and produce a 32-bit result. The destination device in PB, is used to store low 32-bit data only. PC/PA/PH will store all 64-bit data.

Program Example:

The value in D10 is multiplied by the value in D0 and the total is a 32-bit result stored in (D21, D20). The upper 16-bit data is stored in D21 and the lower one is stored in D20. The polarity of the result is indicated by the OFF/ON of the most significant bit. OFF indicates the value of positive (0) and ON indicates the value of negative (1).



$$(D0) \times (D10) = (D21, D20)$$

$$16\text{-bit} \times 16\text{-bit} = 32\text{-bit}$$

API	Mnemonic			Operands			Function								Controllers			
23	D	DIV	P	S ₁	S ₂	D	Division								PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DIV, DIVP: 7 steps DDIV, DDIVP: 13 steps			
S ₁					*	*	*	*	*	*	*	*	*	*					
S ₂					*	*	*	*	*	*	*	*	*	*					
D								*	*	*	*	*	*	*					

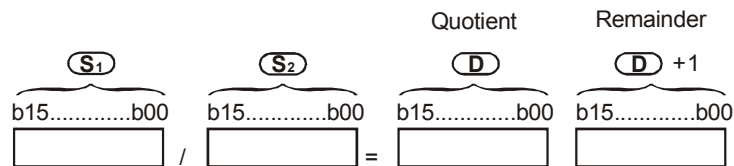
PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: dividend S₂: divisor D: Quotient and Remainder

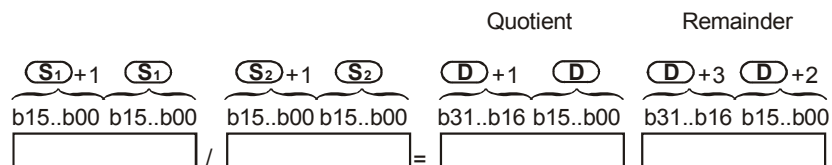
Explanation:

1. The primary source (S₁) is divided by the secondary source(S₂). The result is stored in the destination (D). Not the normal rules of algebra.
2. This instruction is not executed when the divisor is "0". Then, the flag M1067, M1068 = ON and D1067 holds error code H0E19.
3. If operands S₁, S₂, D use index F, then only 16-bit instruction is available.
4. 16-bit instruction:



When D is a bit device, it can specify K1~K4 to produce a 16-bit result and occupies 2 of continuous 16-bit for quotient and remainder.

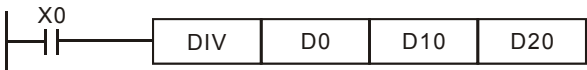
5. 32-bit instruction:



When D is a bit device, it can specify K1~K8 and produce a 32-bit quotient for PB, and occupies 2 of continuous 32-bit for quotient and remainder for PC/PA/PH.

Program Example:

When X0 = ON, the value in D0 (dividend) is divided by the value in D10 (divisor). The quotient is stored in D20 and the remainder is stored in D21. The polarity of the result is indicated by the OFF/ON of the most significant bit. OFF indicates the value of positive and ON indicates the value of negative.



API	Mnemonic			Operands			Function								Controllers								
24	D	INC		P	D			Increment								PB	PC	PA	PH				
Type	Bit Devices				Word devices										Program Steps								
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	INC, INCP: 3 steps							
D							*	*	*	*	*	*	*	*	*	DINC, DINCP: 5 steps							
												PULSE				16-bit				32-bit			
												PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

D: Destination

Explanations:

1. If the instruction is not in pulse mode, “1” is added to the value of destination **D** every execution of the instruction. Which is potentially every scan.
2. This instruction works best using pulse mode (INCP, DINCP).
3. In 16-bit instruction, when +32,767 is reached, “1” is added and it will write a value of –32,768 to the destination. In 32-bit instruction, when +2,147,483,647 is reached, “1” is added and it will write a value of -2,147,483,648 to the destination.
4. Flag M1020~M1022 won't be affected by the operation result of this instruction.
5. If operand **D** uses index F, then only 16-bit instruction is available.

Program Example:

When X0 = OFF → ON, the content of D0 will be incremented by 1.



API	Mnemonic			Operands	Function	Controllers			
25	D	DEC	P	D	Decrement	PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
D							*	*	*	*	*	*	*	*	*	DEC, DECP: 3 steps DDEC, DDECP: 5 steps

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

D: Destination

Explanation:

1. If the instruction is not in pulse mode, “1” is subtracted to the value of destination **D** on every execution of the instruction. Which is potentially every scan.
2. This instruction is usually works best using pulse mode (DECP, DDECP).
3. In 16-bit instruction, when –32,768 is reached, “1” is subtracted and it will write a value of +32,767 to the destination. In 32-bit instruction, when -2,147,483,648 is reached, “1” is subtracted and it will write a value of +2,147,483,647 to the destination.
4. Flag M1020~M1022 won't be affected by the operation result of this instruction.
5. If operand **D** use index F, then only 16-bit instruction is available.

Program Example:

When X0 = OFF → ON, the value in D0 will be decremented by 1.



API	Mnemonic			Operands			Function								Controllers			
26		WAND	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Logical AND 16-bit								<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	WAND, WANDP: 7 steps		
	S ₁					*	*	*	*	*	*	*	*	*	*	*			
	S ₂					*	*	*	*	*	*	*	*	*	*	*			
	D								*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

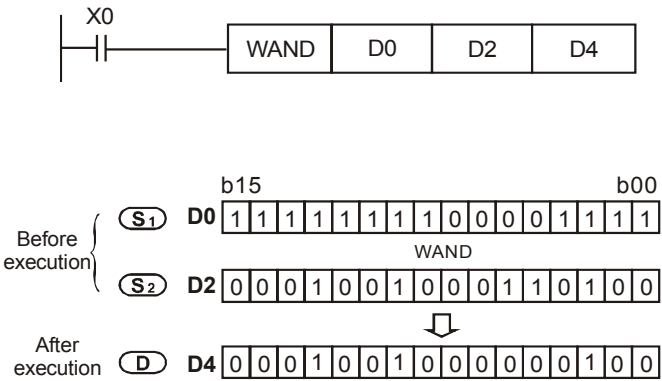
S₁: First data source S₂: Second data source D: Operation result

Explanations:

- 1. The bit patterns of the two source devices are analyzed (the contents of S₂ is compared against the contents of S₁). The result of the logical AND analysis is stored in the destination device (D).
- 2. See DAND for information about this instruction.

Program Example:

When X0 = ON, the 16-bit source D0 and D2 are analyzed and the operation result of the logical WAND is stored in D4.



API	Mnemonic			Operands			Function								Controllers												
26		DAND	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Logical AND 32-bit								<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>									
OP	Type	Bit Devices				Word devices										Program Steps											
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DAND, DANDP: 13 steps										
	S ₁					*	*	*	*	*	*	*	*	*	*												
	S ₂					*	*	*	*	*	*	*	*	*	*												
	D								*	*	*	*	*	*	*												
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

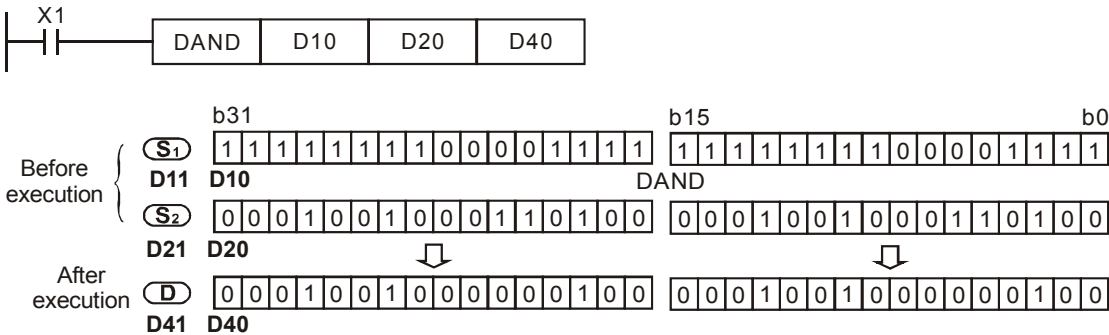
S₁: First data source S₂: Second data source D: Operation result

Explanations:

1. Logical Double word AND operation.
2. The bit patterns of the two source devices are analyzed (the contents of S₂ is compared against the contents of S₁). The result of the logical AND analysis is stored in the destination device (D).

Program Example:

When X1 = ON, the 32-bit source (D11, D10) and (D21, D20) are analyzed and the result of the logical DAND is stored in (D41, D40).



API	Mnemonic			Operands			Function								Controllers			
27		WOR	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Logical OR 16-bit								<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	WOR, WORP: 7 steps		
	S ₁					*	*	*	*	*	*	*	*	*	*	*			
	S ₂					*	*	*	*	*	*	*	*	*	*	*			
	D								*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

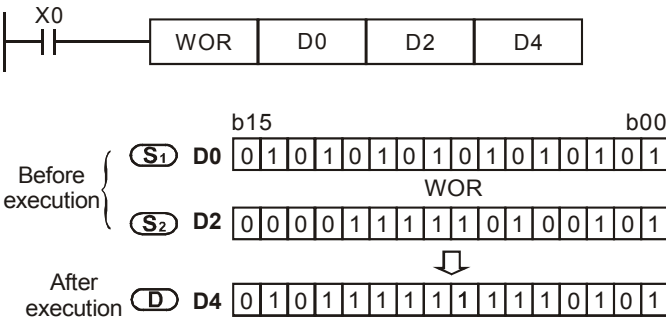
S₁: First data source S₂: Second data source D: Operation result

Explanations:

1. The bit patterns of the two source devices are analyzed (then contents of S₂ is compared against the contents of S₁). The result of the logical OR analysis is stored in the destination device (D).
2. See DOR for information about this instruction.

Program Example:

When X0 = ON, the 16-bit data source D0 and D2 are analyzed and the result of the logical WOR is stored in D4.



API	Mnemonic			Operands			Function								Controllers												
27		DOR	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Logical OR 32-bit								<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>									
OP	Type	Bit Devices				Word devices										Program Steps											
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DOR, DORP: 13 steps										
	S ₁					*	*	*	*	*	*	*	*	*	*	*											
	S ₂					*	*	*	*	*	*	*	*	*	*	*											
	D								*	*	*	*	*	*	*	*											
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

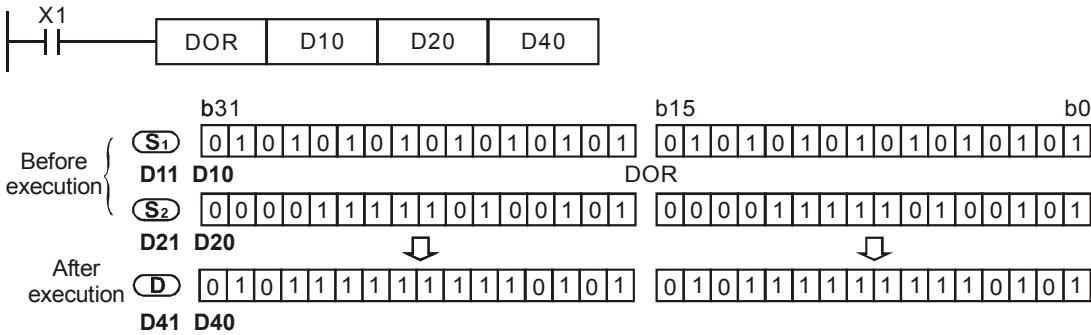
S₁: First data source S₂: Second data source D: Operation result

Explanations:

1. Logical Double word OR operation.
2. The bit patterns of the two source devices are analyzed (then contents of S₂ is compared against the contents of S₁). The result of the logical OR analysis is stored in the destination device (D)

Program Example:

When X1 is ON, the 32-bit data source (D11, D10) and (D21, D20) are analyzed and the operation result of the logical DOR is stored in (D41, D40).



API	Mnemonic			Operands			Function										Controllers			
28		WXOR	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Exclusive XOR 16-bit										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	WXOR, WXORP: 7 steps		
	S ₁					*	*	*	*	*	*	*	*	*	*	*			
	S ₂					*	*	*	*	*	*	*	*	*	*	*			
	D							*	*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

Operands:

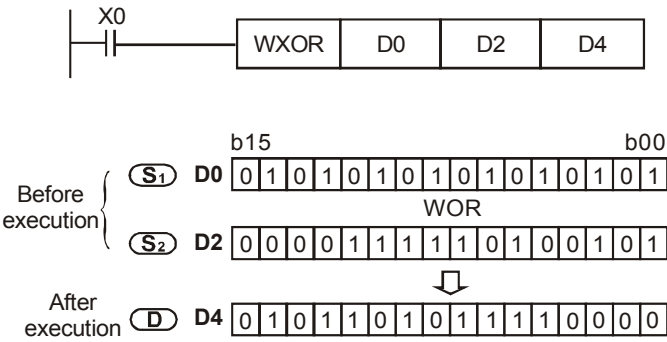
S₁: First data source S₂: Second data source D: Operation result

Explanations:

- 1. The bit patterns of the two source devices are analyzed (then contents of S₂ is compared against the contents of S₁). The result of the logical XOR analysis is stored in the destination device (D)
- 2. See DXOR for information about this instruction.

Program Example:

When X0 = ON, the 16-bit data source D0 and D2 are analyzed and the operation result of the logical WXOR is stored in D4.



API	Mnemonic			Operands			Function								Controllers			
28		DXOR	P	<div><div>S₁</div><div>S₂</div><div>D</div></div>			Exclusive XOR 32-bit								PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DXOR, DXORP: 13 steps		
	S ₁					*	*	*	*	*	*	*	*	*	*	*			
	S ₂					*	*	*	*	*	*	*	*	*	*	*			
	D								*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

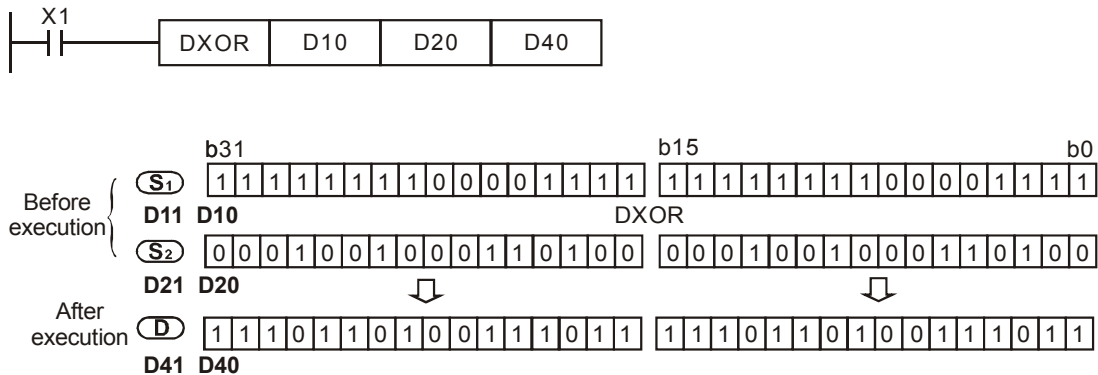
S₁: First data source S₂: Second data source D: Operation result

Explanations:

1. Logical Double word XOR operation.
2. The bit patterns of the two source devices are analyzed (then contents of S₂ is compared against the contents of S₁). The result of the logical DXOR analysis is stored in the destination device (D)

Program Example:

When X1 = ON, the 32-bit data source = (D11, D10) and (D21, D20) are analyzed and the operation result of the logical DXOR = is stored in (D41, D40).



API	Mnemonic			Operands	Function	Controllers			
29	D	NEG	P	<div>D</div>	Negative (2's Compliment)	PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	NEG, NEGP: 3 steps		
D								*	*	*	*	*	*	*	*	DNEG, DNEGP: 5 steps		

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

D: Store the operation of 2's Compliment

Explanations:

1. The bit pattern of the selected device is inverted. This means any occurrence of a '1' becomes a '0' and any occurrence of a '0' will be written as a '1'. When this is complete, a further binary 1 is added to the bit pattern. The result is the total logical sign change of the selected devices contents, e.g. a positive number will become a negative number or a negative number will become a positive.
2. This instruction is usually works best using pulse instruction(NEGP, DNEGP).
3. If operand **D** uses index F, then only 16-bit instruction is available.

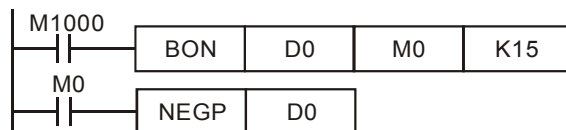
Program Example 1:

When X0 goes from OFF → ON, all bits of D10 will be negative (0→1, 1→0) and then 1 will be added and saved in the original register, D10.

**Program Example 2:**

Obtaining the absolute value of a negative value:

1. When the 15th bit of D0 is "1", M0 = ON. (D0 is a negative value).
2. When M0 = ON, the absolute value of D0 can be obtained using the NEG instruction.



Indication of the negative value and absolute value

1. The content of the most significant bit of the register indicates whether the value is positive or negative. It is a positive value when the most significant bit (MSB) = "0" and it is a negative value when the MSB = "1".
2. If it is a negative value, the absolute value can be obtained by using the NEG instruction (API 29).

(D0=2)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0=1)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0=0)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0=-1)

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 $\overline{(D0)+1=1}$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0=-2)

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 $\overline{(D0)+1=2}$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0=-3)

1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 $\overline{(D0)+1=3}$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0=-4)

1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 $\overline{(D0)+1=4}$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0=-5)

1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 $\overline{(D0)+1=5}$

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

⋮

⋮

(D0=-32,765)

1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 $\overline{(D0)+1=32,765}$

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0=-32,766)

1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 $\overline{(D0)+1=32,766}$

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0=-32,767)

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 $\overline{(D0)+1=32,767}$

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

(D0=-32,768)

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 $\overline{(D0)+1=-32,768}$

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Max. absolute value is 32,767

API	Mnemonic			Operands		Function										Controllers			
30	D	ROR	P	D	n	Rotate Right										PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ROR, RORP: 5 steps			
D								*	*	*	*	*	*	*	*	DROR, DRORP: 9 steps			
n					*	*													

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

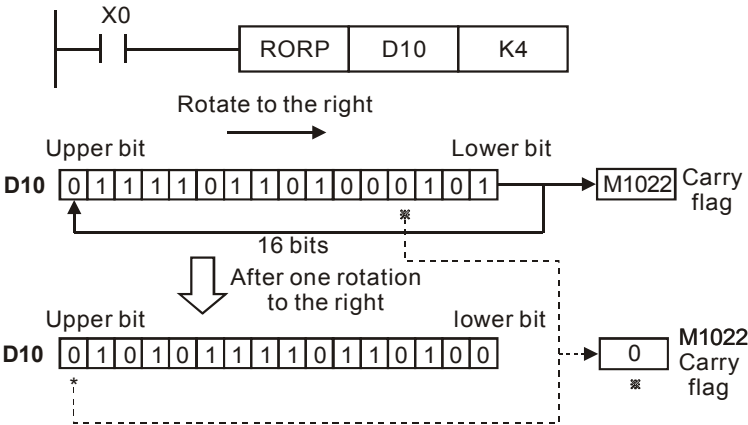
D: Destination n: Number of bit to rotate

Explanations:

- 1. All the bits of **D** are rotated **n** bit places to the right on every operation of the instruction. Which is potentially every scan.
- 2. The status of the last bit rotated is copied to the carry flag M1022 (Carry flag)
- 3. This instruction is usually works best using pulse instruction (RORP, DRORP).
- 4. If operand **D** uses index F, then only 16-bit instruction is available.
- 5. If operand **D** is specified as KnY, KnM, KnS, only K4 (16-bit) and K8 (32-bit) is valid.
- 6. Valid range of operand **n** : 1 ≤ n ≤16 (16-bit), 1 ≤ n ≤32 (32-bit)

Program Example:

When X0 goes from OFF → ON, the 16 bit data of D10 will rotate 4 bits to the right, as shown in the diagram, bit b3 (prior to rotation) will be moved to the carry flag (CY) M1022.



API	Mnemonic			Operands		Function										Controllers			
31	D	ROL	P	<div>D</div>	<div>n</div>	Rotate Left										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F			
D									*	*	*	*	*	*	*	*	ROL, ROLP: 5 steps		
n						*	*										DROL, DROLP: 9 steps		

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

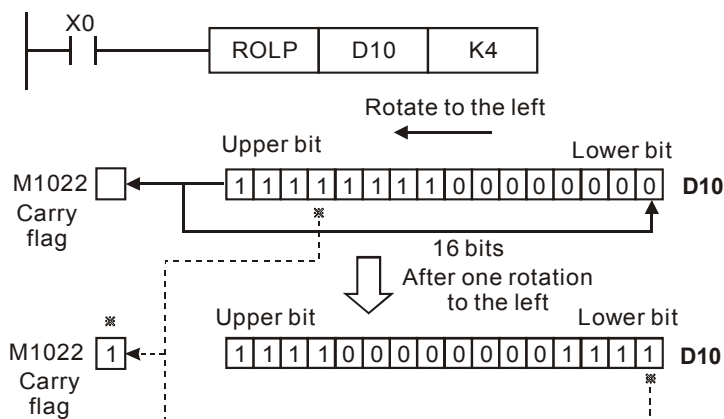
D: Destination **n:** Number of bit to rotate

Explanation:

1. All the bits of **D** are rotated **n** bit places to the left on every operation of the instruction. Which is potentially every scan.
2. The status of the last bit rotated is copied to the carry flag M1022.
3. This instruction is usually works best using pulse instruction (ROLP, DROLP).
4. If operand **D** uses index F, then only 16-bit instruction is available.
5. If operand **D** is specified as KnY, KnM, KnS, only K4 (16-bit) and K8 (32-bit) are valid.
6. Valid range of operand **n** : $1 \leq n \leq 16$ (16-bit), $1 \leq n \leq 32$ (32-bit)

Program Example:

When X0 goes from OFF → ON, all 16 bits of D10 will rotate 4 bits to the left, as shown in the diagram, and b12 (prior to rotation) will be moved to the carry flag (CY) M1022.



API	Mnemonic			Operands		Function										Controllers			
33	D	RCL	P	<div>D</div>	<div>n</div>	Rotate Left with Carry										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	RCL, RCLP: 5 steps			
D								*	*	*	*	*	*	*	*	DRCL, DRCLP: 9 steps			
n					*	*													

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

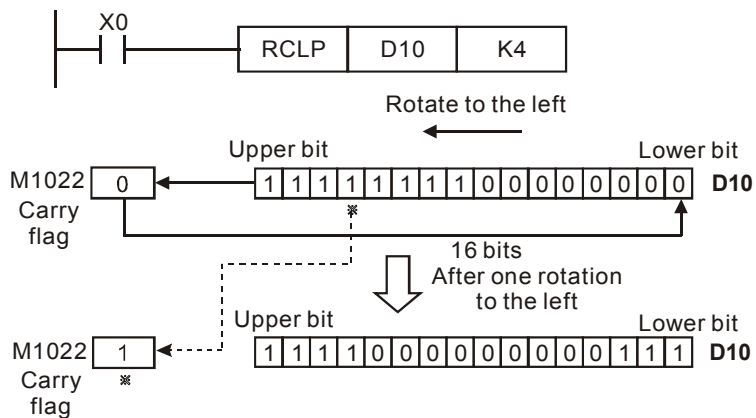
D: Destination **n:** Number of bit to rotate

Explanations:

1. All the bits of **D** are rotated **n** bit places to the left including the carry flag on every operation of the instruction. Which is potentially every scan.
2. The status of the last bit rotated is moved into the carry flag M1022. On the following operation of the instruction M1022 is the first bit to be moved back into the destination device.
3. This instruction works best with the pulse instruction(RCLP, DRCLP).
4. If operand **D** uses index F, then only 16-bit instruction is available.
5. If operand **D** is specified as KnY, KnM, KnS, only K4 (16-bit) and K8 (32-bit) is valid.
6. Valid range of operand **n** : $1 \leq n \leq 16$ (16-bit), $1 \leq n \leq 32$ (32-bit)

Program Example:

When X0 goes from OFF → ON, the 16 bit value in D10, including the carry flag (M1022), will rotate 4 bits to the left, as shown in the diagram, b12 (prior to rotation) will be moved to the carry flag M1022, and that the original contents of the carry flag M1022 will be moved to the bit of b3.



API	Mnemonic		Operands				Function	Controllers			
34	SFTR	P	(S)	(D)	(n ₁)	(n ₂)	Bit Shift Right	PB	PC	PA	PH

Type	Bit Devices				Word devices											Program Steps			
OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SFTR, SFTRP: 9 steps			
S	*	*	*	*															
D		*	*	*															
n ₁					*	*													
n ₂					*	*													

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

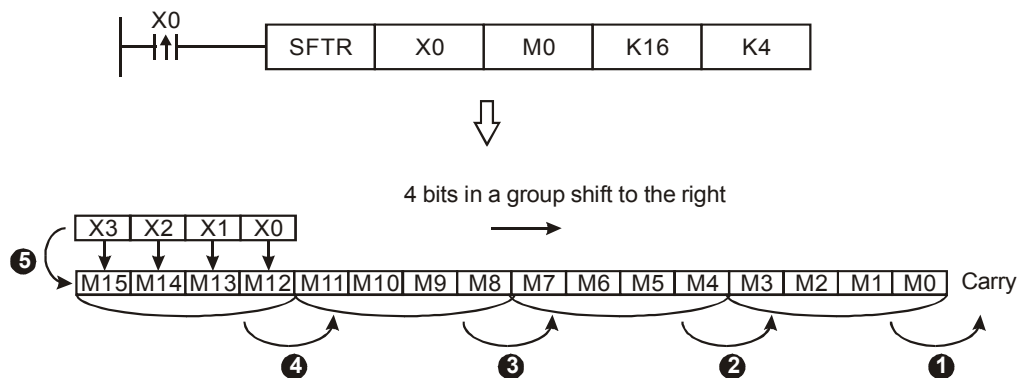
S: Source **D:** Destination **n₁:** Number of bits to shift **n₂:** Number of bits to shift at one time

Explanation:

- Shift **n₁** bits of **S** to the right by **n₁** bits. Shift **n₂** bits of **D** to the right.
- This instruction works best with the pulse instruction (SFTRP).
- Valid range of operand **n₁**, **n₂** : $1 \leq n_2 \leq n_1 \leq 1024$, In PB models: $1 \leq n_2 \leq n_1 \leq 512$

Program Example:

- When X0 OFF → ON, the 16 bit data of M0~M15 will shift 4 bits to the right. And 4 bits from X0 into the upper order bits of M0.
- Please refer to the following ❶~❺ steps to perform SFTR instruction during a single 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 complete



API	Mnemonic			Operands								Function					Controllers										
35		SFTL	P	S	D	n ₁	n ₂	Bit Shift Left								PB	PC	PA	PH								
OP	Type	Bit Devices				Word devices												Program Steps									
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SFTL, SFTLP: 9 steps											
S	*	*	*	*																							
D		*	*	*																							
n ₁					*	*																					
n ₂					*	*																					
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: Source **D:** Destination **n₁:** Number of bits to shift **n₂:** Number of bits to shift at one time

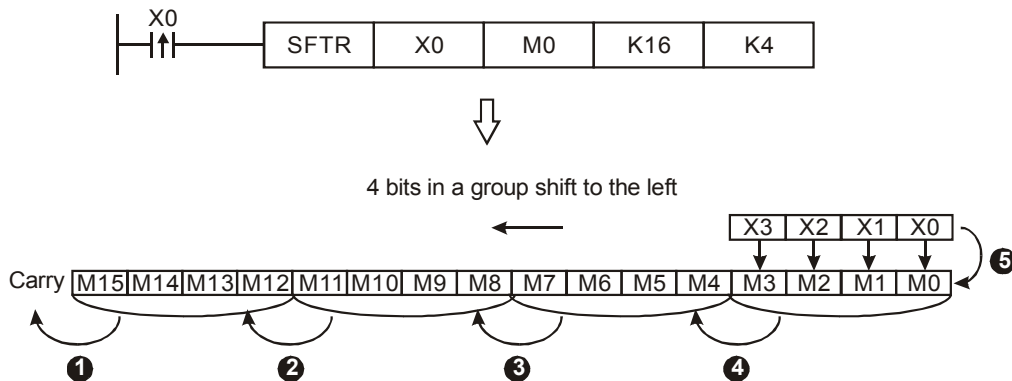
Explanations:

- Shift **n₁** bits of **S** to the left by **n₁** bits. Shift **n₂** bits of **D** to the left.
- This instruction works best with the pulse instruction(SFTLP).
- Valid range of operand **n₁**, **n₂** : $1 \leq n_2 \leq n_1 \leq 1024$, In PB models: $1 \leq n_2 \leq n_1 \leq 512$

Program Example:

- When X0 OFF → ON, the 16 bit data of M0~M15 will shift 4 bits to the left. And 4 bits from X0 into the low order bits of M0.
- Please refer to the following ❶~❺ steps to perform SFTR instruction during a single 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 complete



API	Mnemonic			Operands				Function				Controllers															
36		WSFR	P	(S)	(D)	(n ₁)	(n ₂)	Word Shift Right				PB	PC	PA	PH												
OP	Type	Bit Devices				Word devices										Program Steps											
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	WSFR, WSFRP: 9 steps										
	S							*	*	*	*	*	*	*													
	D								*	*	*	*	*	*													
	n ₁						*	*																			
n ₂						*	*																				
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: Source **D:** Destination **n₁:** Number of registers to shift **n₂:** Number of registers to shift at one time

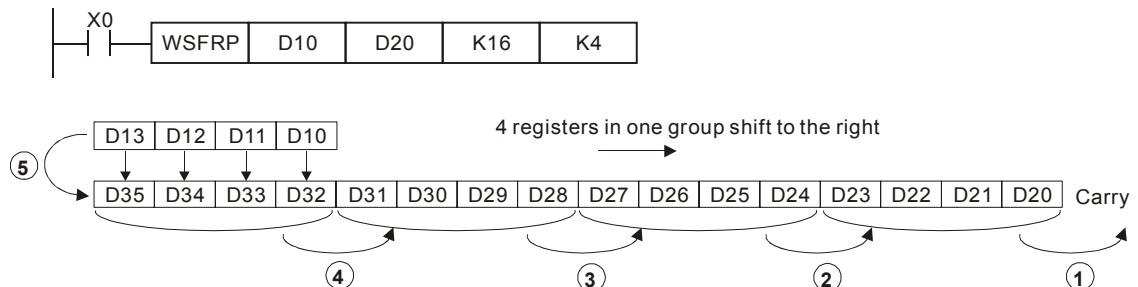
Explanations:

- Shift **n₁** registers of **S** to the right by **n₁** registers. Shift **n₂** registers of **D** to the right.
- This instruction works best with the pulse instruction (WSFRP).
- When using operand **S** and **D** for bit data types, the data type must be equal, for example, one kind is the KnX, KnY, KnM, KnS and the other kind is T, C, D.
- When using operand **S** and **D** bit data types, the Kn value must be equal.
- Valid range of operand **n₁, n₂** : $1 \leq n_2 \leq n_1 \leq 512$

Program Example 1:

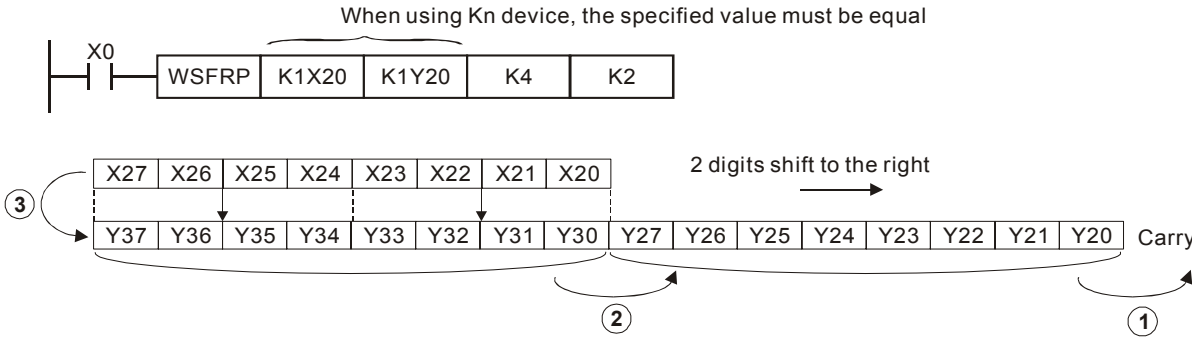
- When X0 OFF → ON, the registers starting at D20~D35 will shift 4 registers to the right. And 4 registers from D10 will shift into the upper registers of the destination.
- Please refer to the following ①~⑤ steps to perform WSFR instruction during a single scan.

- ① D23~D20 → Carry
- ② D27~D24 → D23~D20
- ③ D31~D28 → D27~D24
- ④ D35~D32 → D31~D28
- ⑤ D13~D10 → D35~D32 complete



Program Example 2:

- 1. When X0 OFF → ON, the bit registers of Y20~Y37 are series a shift area and shift 2 digits to the right.
- 2. Please refer to the following ❶~❸ steps to perform WSFR instruction of one time shift.
 - ❶ Y27~Y20 → carry
 - ❷ Y37~Y30 → Y27~Y20
 - ❸ X27~X20 → Y37~Y30 complete



API	Mnemonic			Operands						Function						Controllers			
37		WSFL	P	<div>S</div>	<div>D</div>	<div>n₁</div>	<div>n₂</div>	Word Shift Left						<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>		

Type OP		Bit Devices				Word devices										Program Steps WSFL, WSFLP: 9 steps								
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E									F
S								*	*	*	*	*	*	*										
D									*	*	*	*	*	*										
n ₁					*	*																		
n ₂					*	*																		

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: Source **D:** Destination **n₁:** Number of registers to shift **n₂:** Number of registers to shift at one time

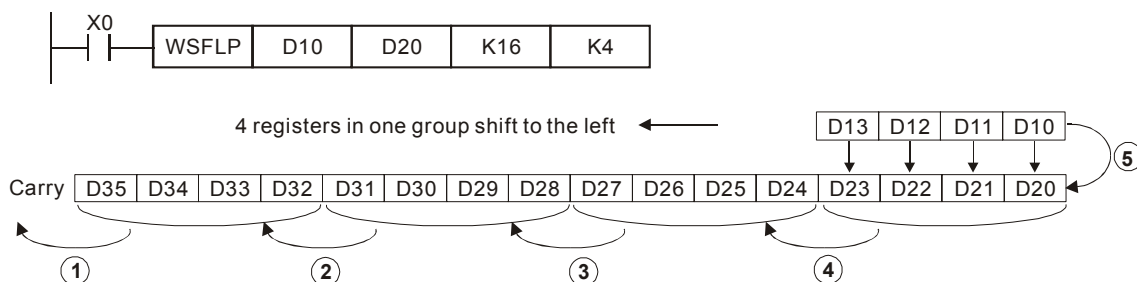
Explanations:

- Shift **n₁** registers of **S** to the left by **n₁** registers. Shift **n₂** registers of **D** to the left.
- This instruction works best with the pulse instruction (WSFLP).
- When using operand **S** and **D** for bit data types, the data type must be equal, for example, one kind is the KnX, KnY, KnM, KnS and the other kind is T, C, D.
- When using operand **S** and **D** bit data types, the Kn value must be equal.
- Valid range of operand **n₁, n₂** : $1 \leq n_2 \leq n_1 \leq 512$

Program Example:

- When X0 OFF → ON, the registers starting at D20~D35 will shift 4 registers to the left. And 4 registers from D10 will shift into the lower registers of the destination.
- Please refer to the following ①~⑤ steps to perform WSFL instruction during a single scan.

- ① D35~D32 → Carry
- ② D31~D28 → D35~D32
- ③ D27~D24 → D31~D28
- ④ D23~D20 → D27~D24
- ⑤ D13~D10 → D23~D20 complete



API	Mnemonic			Operands			Function			Controllers			
38		SFWR	P	(S)	(D)	(n)	Shift Register Write			PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SFWR, SFWRP: 7 steps
S					*	*	*	*	*	*	*	*	*	*	*	
D								*	*	*	*	*	*			
n					*	*										

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

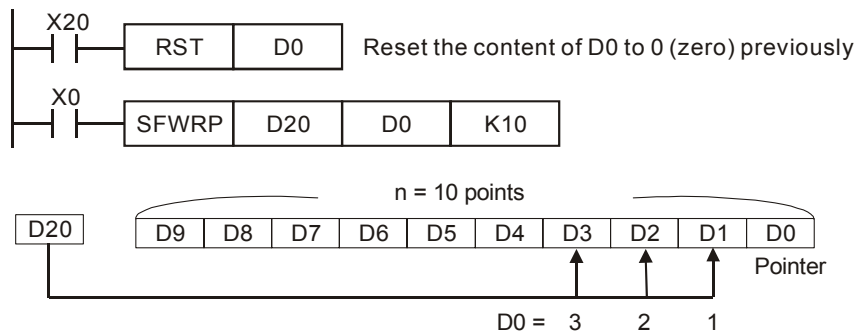
S: Source device which the data is written in **D:** Head address device **n:** Data length

Explanations:

1. **n** is the length of the First-in/First-out queue and the destination device
2. **D** is the head address device of the First-in/First-out queue. Use the first number device **D** as the pointer and add 1 to the content value of the pointer when executing this instruction. The contents of the devices specified by **S** are written into the position specified by the pointer **D** of the First-in/First-out queue.
3. If the contents of the pointer **D** exceed the value “n-1”, the insertion into the First-in/First-out queue will stop and the carry flag M1022 will be turned ON.
4. This instruction works best with the pulse instruction(SFWRP).
5. Valid range of operand **n** : $2 \leq n \leq 512$

Program Example:

1. First, X20=ON reset the content of D0 to 0. When X0 goes from OFF to ON, the content of D0 becomes 1 when the content of D20 is created and built in D1. After changing the content of D20, X0 is executed to goes from OFF to ON again, then the content of D0 becomes 2 when the content of D20 is created and built in D2.
2. Please refer to the following ❶~❷steps to perform SFWR instruction.
 - ❶ The content of D20 is created and built in D1.
 - ❷ The content of D0 becomes 1.

**Points to note:**

This API 38 SFWR instruction can be used with the API 39 SFRD instruction to execute the Write-in/Read Control of the First-in/First-out queue.

API	Mnemonic			Operands			Function			Controllers			
39		SFRD	P	S	D	n	Shift Register Read			PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SFRD, SFRDP: 7 steps		
	S								*	*	*	*	*	*					
	D								*	*	*	*	*	*	*	*			
	n					*	*												

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

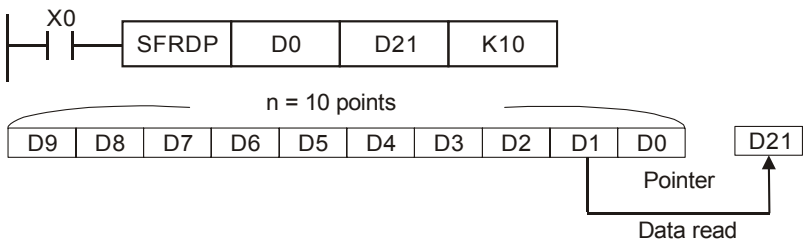
S: Head address device **D:** destination device **n:** Data length

Explanation:

- n** is the length of the First-in/First-out queue and the source device **S** is the head address device of the First-in/First-out queue. Use the first number device **S** as the pointer and subtract 1 to the content value of the pointer when executing this instruction. The contents of the devices specified by **S** are written into the position specified by the pointer of the First-in/First-out queue.
- If the contents of the pointer **S** are equal to 0 (zero), the First-in/First-out queue will be empty and the zero flag M1020 will be turned ON.
- This instruction works best with the pulse instruction(SFRDP).
- Valid range of operand **n** : $2 \leq n \leq 512$

Program Example:

- When X0 goes from OFF to ON, D9~D2 are all shifted one register to the right and the pointer content of D0 is subtracted by 1 when the content of D1 is read and moved to D21.
- Please refer to the following ❶~❸ steps to perform SFRD instruction.
 - ❶ The content of D1 is read and moved to D21.
 - ❷ D9~D2 are all shifted one register to the right.
 - ❸ The content of D0 is subtracted by 1.



API	Mnemonic			Operands		Function										Controllers			
40		ZRST	P	D₁	D₂	Zone Reset										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ZRST, ZRSTP: 5 steps			
D ₁		*	*	*							*	*	*						
D ₂		*	*	*							*	*	*						

PULSE				16-bit				32-bit							
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

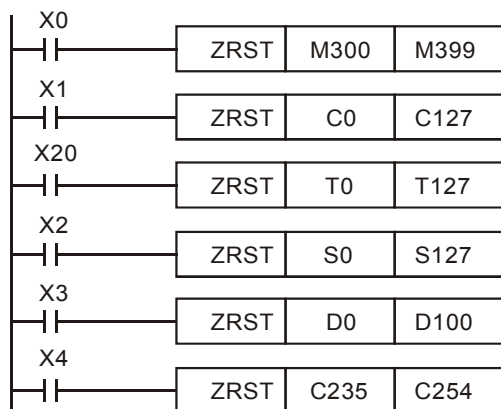
D₁: Starting destination **D₂**: Ending destination

Explanations:

- When **D₁** > **D₂**, then only device **D₂** is reset.
- PB models, standard and High speed counters cannot be mixed.
- This instruction works best with the pulse instruction (ZRSTP).
- Operand **D₁** and **D₂** must be the same data type, Valid range: **D₁** ≤ **D₂**.

Program Example:

- When X0 = ON, M300 to M399 will be reset to OFF.
- When X1 = ON, C0 to C127 will all be reset. Their present value =0 , coil output will be reset to OFF.
- When X20 = ON, T0 to T127 will all be reset. Their present value =0 , coil output will be reset to OFF.
- When X2 = ON, the status of S0 to S127 will be reset to OFF.
- When X3 = ON, the data of D0 to D100 will be reset to 0.
- When X4 = ON, C235 to C254 will all be reset. Their present value =0 , coil output will be reset to OFF.



API	Mnemonic			Operands			Function			Controllers			
41		DECO	P	(S)	(D)	(n)	Decode			PB	PC	PA	PH

OP	Type	Bit Devices				Word devices								Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DECO, DECOP: 7 steps
S		*	*	*	*	*	*					*	*	*	*	*	
D			*	*	*							*	*	*	*	*	
n						*	*										

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

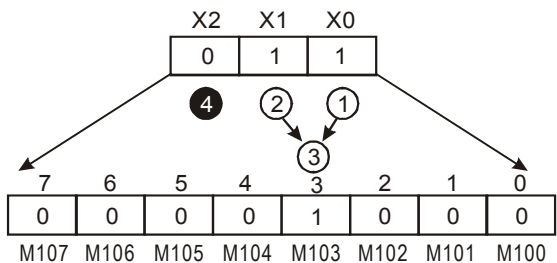
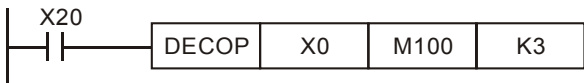
S: Source to decode **D:** Destination **n:** Number of bits to decode

Explanation:

1. Decodes the lower “n” bits of source **S** and stores the result of “2ⁿ” bit at **D**.
2. This instruction works best with the pulse instruction(DECOP).
3. When operand **D** is a bit device, **n**=1~8, when operand **D** is a word device, **n**=1~4

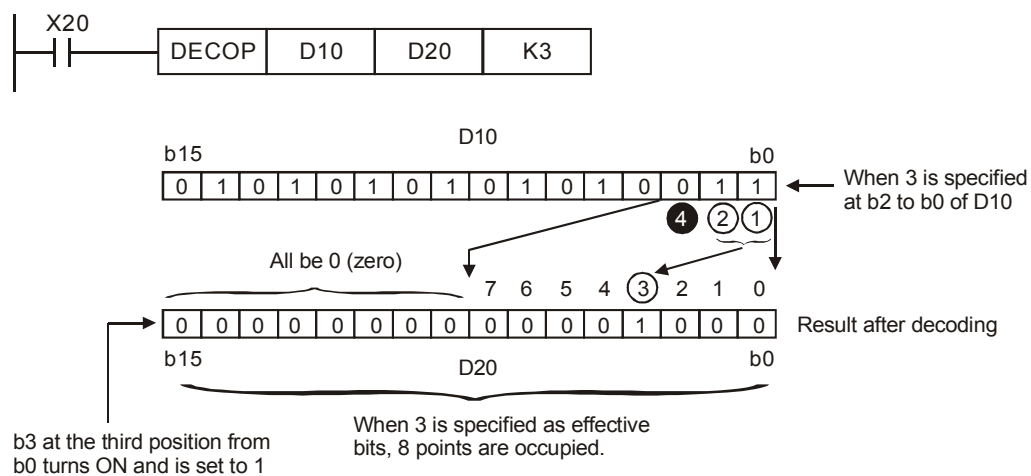
Program Example 1:

1. **n** valid range: $0 < n \leq 8$. But if **n**=0 or **n**>8, an error will occur.
2. When **n**=8, the maximum decoded data is $2^8 = 256$ points.
3. When X20 goes from OFF → ON, the data of X0~X2 will be decoded to M100~M107.
4. If **S** =3, M103 (third position from M100) = ON.
5. After the execution is completed, X20 is changed to OFF. The device which have been decoded is still action.



Program Example 2:

1. **D** valid range: $0 < n \leq 4$, if $n=0$ or $n>4$, an error will occur.
2. When $n=4$, the maximum decoded data is $2^4 = 16$ points.
3. When X20 goes from OFF \rightarrow ON, the data in D10 (b2 to b0) will be decoded and stored at D20 (b7 to b0). The unused bits in D20 (b15 to b8) will be all set to 0.
4. Decodes three lower bits in D10 and stores at eight lower bits in D20 (one bit will be 1) and the content of eight upper bits are all 0.
5. After the execution is completed, X20 is changed to OFF. The device which have been decoded is still action.



API	Mnemonic			Operands			Function			Controllers				
42		ENCO	P	<div>S</div>	<div>D</div>	<div>n</div>	Encode				PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	
	S	*	*	*	*							*	*	*	*	*
	D											*	*	*	*	*
	n					*	*									

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

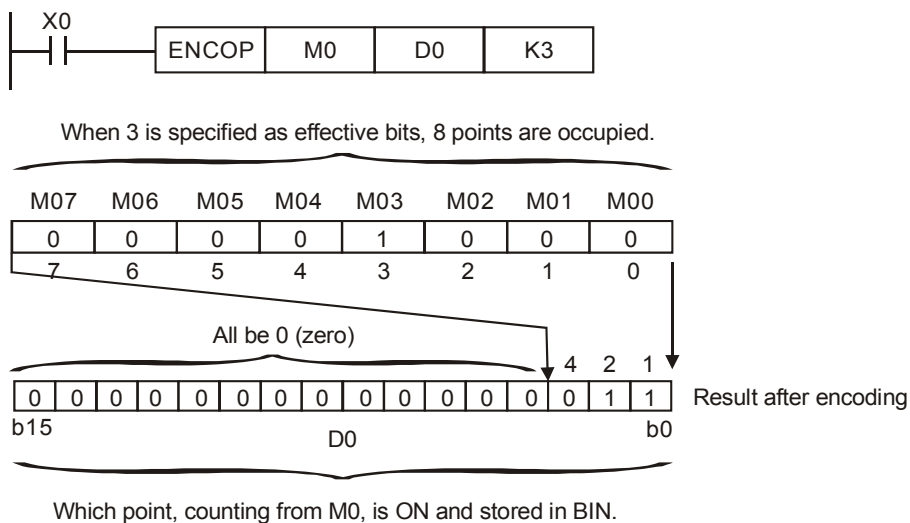
S: Source to encode **D:** Destination for storing encode data **n:** Number of bits to encode

Explanation:

1. Encodes the lower “2ⁿ” bits of source **S** and stores the result in **D**.
2. If the source device **S** multiple bit are 1, processing is performed for the last bit position.
3. This instruction works best with the pulse instruction(ENCOP).
4. When operand **S** is a bit device, **n**=1~8, when operand **S** is a word device, **n**=1~4
5. PC/PA/PH series: If there is no bit=1 in the source **S** of ENCO instruction, M1067=M1068=ON.

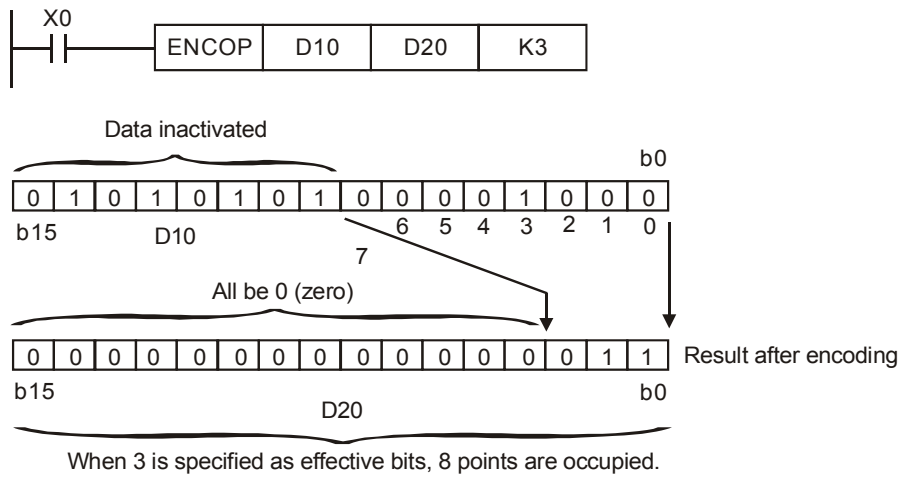
Program Example 1:

1. **S** valid range: $0 < n \leq 8$. If **n**=0 or **n**>8, an error will occur.
2. When **n**=8, the maximum decoded data is $2^8 = 256$ points.
3. When X0 goes from OFF → ON, the data in (M0 to M7) will be encoded and stored at 2³ bits of D0 (b2 to b0). The unused bits in D0 (b15 to b3) will be all set to 0.
4. After the execution is completed, X0 is changed to OFF and the data in **D** remain unchanged.



Program Example 2:

- 1. **S** Valid rang: $0 < n \leq 4$. If $n=0$ or $n>4$, an error will occur.
- 2. When $n=4$, the maximum decoded data is $2^4 = 16$ points.
- 3. When X0 goes from OFF → ON, the data in D10 will be encoded and stored at the three lower bits D20 (b2 to b0). The unused bits in D20 (b15 to b3) will be all set to 0.
- 4. After the execution is completed, X0 is changed to OFF and the data in **D** remain unchanged.



API	Mnemonic			Operands		Function										Controllers			
43	D	SUM	P	<div>S</div>	<div>D</div>	Sum of ON bits										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F				
S					*	*	*	*	*	*	*	*	*	*	*	SUM, DSUMP: 5 steps			
D											*	*	*	*	*	DSUM, DSUMP: 9 steps			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

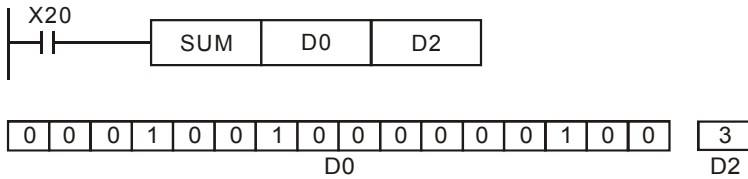
S: Source **D:** Destination stores number of ON bits

Explanation:

1. If the contents of these 16 bits are all “0”, the “Zero” flag, M1020=ON.
2. **D** will occupy two registers when using in 32-bit instruction.
3. If operand **S**, **D** uses index F, it is only available as a 16-bit instruction.
4. If no bits are ON then zero flag, M1020 is set.

Program Example:

When X20 =ON, all the bits that = “1” in D0 will be counted and the number stored in D2.



API	Mnemonic			Operands			Function								Controllers			
44	D	BON	P	(S)	(D)	(n)	Bit ON Test								PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	BON, BONP: 7 steps DBON, DBONP: 13 steps			
S					*	*	*	*	*	*	*	*	*	*	*				
D		*	*	*															
n					*	*					*	*	*	*	*				

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

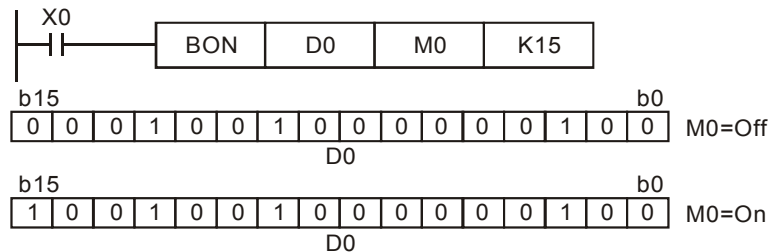
S: Source **D:** Destination for storing result **n:** Bit number to test

Explanation:

1. The status of the specified bit in the source device is indicated at the destination.
2. The destination device could be said to act as a mirror to the status of the selected bit source.
3. If operand **S, n** uses index F, then only 16-bit instruction is available.
4. Valid range of operand **n** : **n** = 0~15 (16-bit), **n** = 0~31 (32-bit)

Program Example:

1. When X0 = ON, and if the 15th bit of D0 = "1", M0 is ON. But if the 15th bit of D0 is "0", M0 is OFF.
2. Once X0 is switched to OFF, M0 will stay at its previous ON/OFF status.



API	Mnemonic			Operands			Function			Controllers			
45	D	MEAN	P	(S)	(D)	(n)	Mean Value			PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S							*	*	*	*	*	*	*			MEAN, MEANP: 7 steps
D								*	*	*	*	*	*	*	*	DMEAN, DMEANP: 13
n					*	*	*	*	*	*	*	*	*	*	*	steps

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

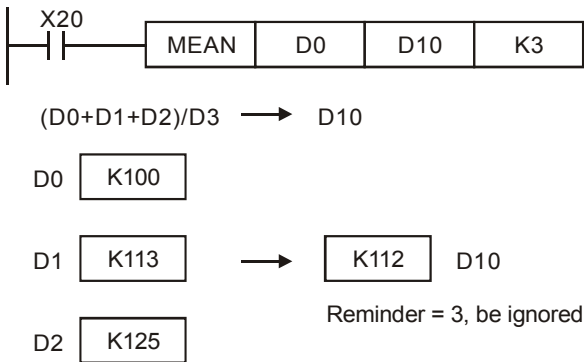
S: Starting source **D:** Destination for result **n:** Number of device to use

Explanations:

1. Add the contents of **n** registers specified by **S**, and have the sum divided by **n** to get the mean value. Save this mean value in **D**.
2. If there is remainder it will be ignored.
3. If **S** is not within the valid range, only those addresses within the range will be processed.
4. If **n** is out of the valid range (1~64), an error will be generated.
5. If operand **D**, **n** uses index F, then only 16-bit instruction is available.
6. Valid range of operand **n** : **n**=1~64

Program Example:

When X20 = ON, add up the contents of the three registers starting from D0, and divide the sum by three to get the mean value. Store this mean value in D10 and ignore the remainder.



API	Mnemonic	Operands			Function										Controllers			
46	ANS	<div>S</div>	<div>m</div>	<div>D</div>	Alarm Set										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ANS: 7 steps			
S											*								
m						*													
D				*															

Operands:

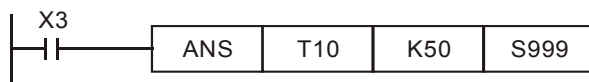
S: Alarm timer **m:** Time setting prior to alarm **D:** Alarm

Explanations:

1. ANS instruction is used to drive the output alarm device.
2. Operand **S** valid range: T0~T191
 Operand **m** valid range: K1~K32,767 in units of 100 ms
 Operand **D** valid range: S896~S1023
 See ANR for more information
3. Flag: M1048 (ON = Alarm Active), M1049 (ON = Enable Alarms)

Program Example:

If alarm device S999=ON and X3 = ON for more than 5 seconds, S999 will stay ON after X3=OFF. T10 will be reset to OFF, present value=0)



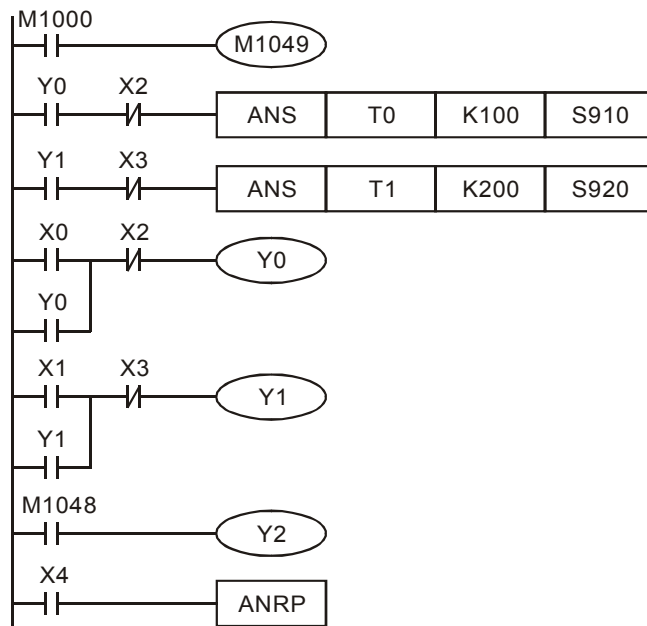
API	Mnemonic			Function	Controllers			
					PB	PC	PA	PH
47		ANR	P	Alarm Reset				

OP	Descriptions	Program Steps
N/A	Instruction driven by contact is necessary.	ANR, ANRP: 1 steps

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Application of alarm device:

X0=forward switch X1=backward switch
 X2=front location switch X3=back location switch
 X4=alarm device reset button
 Y0=forward Y1=forward
 Y2=alarm indicator
 S910=forward alarm device S920=backward alarm



1. When M1049=ON, Alarms are enabled. If M1048=ON, an alarm has occurred, D1049=lowest alarm number.
2. If Y0=ON > 10 seconds and has not reach the front location X2, S910=ON.
3. If Y1=ON > 20 seconds and not reach the back location X3, S920=ON.
4. When X1=ON, and Y1=ON, and X3=ON, Y1 = OFF.
5. If an alarm occurs, alarm indicator Y2=ON.
6. Each alarm which are activated will be reset one by one, each time the reset button X4 = ON during the execution of this instruction. The lowest numbered alarm is reset every execution of this instruction.

API	Mnemonic			Operands		Function										Controllers			
48	D	SQR	P	(S)	(D)	Square Root										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F			
S						*	*							*			SQR, SQRP: 5 steps		
D														*			DSQR, DSQRP: 9 steps		

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

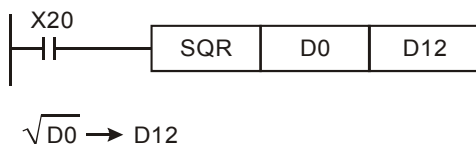
S: Source **D:** Destination to store result

Explanation:

1. Perform a square root on source **S** and store the result in **D**.
2. **S** can only be a positive value. Performing a square root operation on a negative value will result in an error and the instruction will not be executed. The error flag M1067 and M1068 = ON and D1067 records error code H0E1B.
3. SQR result **D** is calculated as an integer only, decimals are ignored. If there is a decimal ignored, the Borrow flag M1021=ON.
4. When SQR result **D** = 0, the Zero flag M1020=ON.
5. M1020 (Zero flag) is set ON when square root operation result is equal to zero, answers with rounded values will activate M1021.
6. Performing any square root operation (even on a calculator) on a negative number will result in an error. This will be identified by special M coil M1067 being activated.

Program Example:

When X20=ON, SQR of D0 will be stored in D12.



API	Mnemonic			Operands		Function										Controllers									
49	D	FLT	P	(S)	(D)	Floating Point										PB	PC	PA	PH						
<div>OP \ Type</div>	Bit Devices				Word devices										Program Steps										
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	FLT, FLTP: 5 steps									
													*			DFLT, DFLTP: 9 steps									
													*												
														PULSE				16-bit				32-bit			
														PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

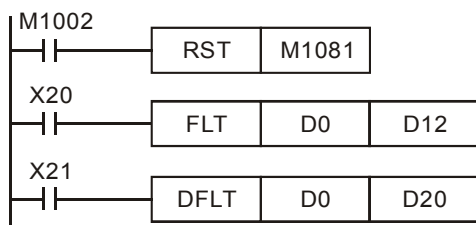
Operands:**S:** Source **D:** Destination**Explanations:**

1. When M1081 = OFF, the source is converted from BIN integer to floating point. At this time, **S** source = 16-bit and **D** occupies 32 bits.
 - a) If an absolute of conversion result is larger than max. floating value, carry flag M1022=ON.
 - b) If an absolute of conversion result is less than min. floating value, carry flag M1021=ON.
 - c) If conversion result is 0, zero flag M1020=ON.
2. When M1081 is ON, the source is converted from floating point to BIN integer. (ignore decimal). At this time, **S** source = 32-bit and **D** Destination occupies 16-bit. This is the same as instruction INT.
 - a) If conversion result exceeds BIN integer range of **D** (16-bit, -32,768~32,767 and 32-bit, -2,147,483,648~2,147,483,647), **D** will hold either max or min value, The carry flag will be set to M1022=ON.
 - b) If the decimal was ignored, borrow flag M1021=ON.
 - c) If the conversion result = 0, zero flag M1020=ON.
 - d) After conversion, **D** is saved by 16 bits.

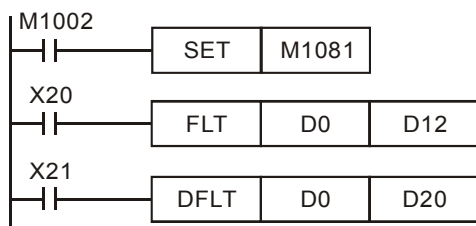
Program Example 1:

1. When M1081 = OFF, the source data is converted from BIN integer to floating point.
2. When X20 = ON, D0 is converted to D13, D12 (floating point).
3. When X21 = ON, D1, D0 are converted to D21, D20 (floating point).
4. If D0=K10, X20=ON. 32-bit of floating point after conversion will be H41200000 and it will be saved in 32-bit register D12(D13).

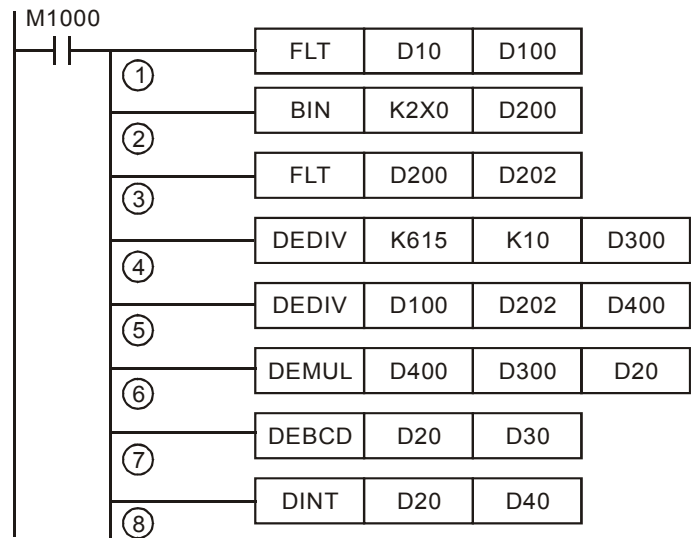
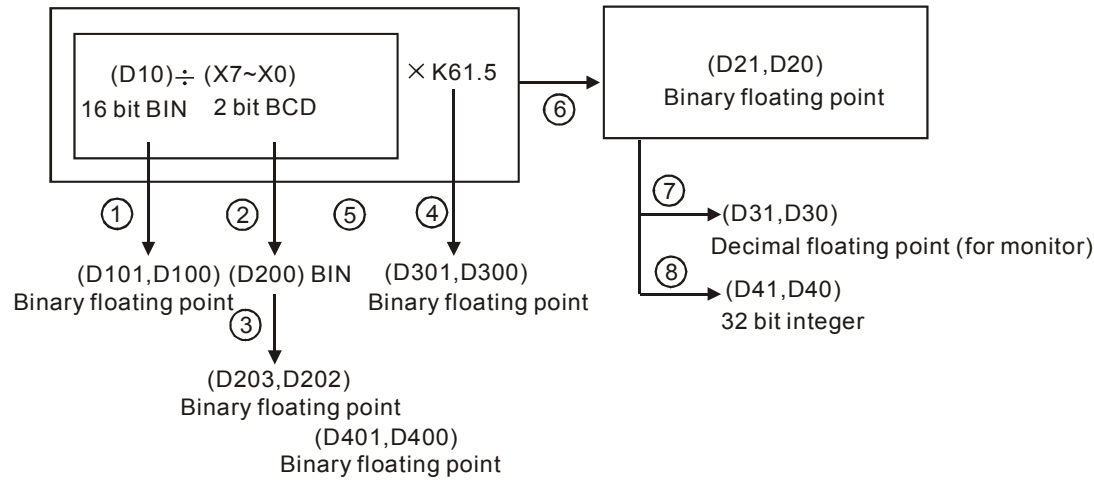
If 32-bit register D0(D1)=K100,000, X21 = ON. 32-bit of floating point after conversion will be H4735000 and it will be saved in 32-bit register D20(D21).

**Program Example 2:**

1. When M1081 = ON, the source data is converted from floating point to BIN integer. (ignore decimal)
2. When X20 = ON, D0 and D1(floating point) are converted to D12 (BIN integer). If D0(D1)=H47C35000, the floating point result is 100,000. The result will be D12=K32,767, M1022=ON, since the value exceeds the max value of the 16-bit register D12.
3. When X21 = ON, D1, D0 (floating point) are converted to D21, D20 (BIN integer). If D0(D1)=H47C35000, the floating point result is 100,000. The result will be saved in 32-bit register D20(D21).



Program Example 3:



1. Covert D10 (BIN integer) to D101, D100 (floating point).
2. Covert the value of X7~X0 (BCD value) to D200 (BIN value).
3. Covert D200 (BIN integer) to D203, D202 (floating point).
4. Save the result of $K615 \div K10$ to D301, D300 (floating point).
5. Divide the floating point:
Save the result of $(D101, D100) \div (D203, D202)$ to D401, D400 (floating point).
6. Multiply floating point:
Save the result of $(D401, D400) \times (D301, D300)$ to D21, D20 (floating point).
7. Covert floating point (D21, D20) to decimal floating point (D31, D30).
8. Covert floating point (D21, D20) to BIN integer (D41, D40).

API	Mnemonic			Operands		Function										Controllers			
50		REF	P	<div>D</div>	<div>n</div>	Refresh I/O Immediately										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F				
D	*	*														REF, REFP: 5 steps			
n					*	*													

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

D: Starting source of I/O refresh **n**: Number of I/O to refresh

Explanations:

- The state of all ELC inputs and outputs will be refreshed after scanning the main program to the END. The state of inputs is read from external inputs to saved to internal memory. Output are updates after the main program has been completed. This instruction can be used during scan time when there is a need to update an I/O point before reaching the program END.
- D** should always be a multiple of 10, i.e. 00, 10,... etc., so it should be X0, X10, Y0, Y10... etc. **n** should always be a multiple of 8, i.e. 8, 16. and its available range is 8~16. If the value of **n** is out of the stated range (8~16) or not a multiple of 8, an error will be generated.
- For all ELC controllers , the input and output points processed by this instruction are the I/O points of MPU: X0~X7, Y0~Y7 and **n**=K8

Program Example 1:

When X0 = ON, ELC will read the state of X0~X7 input points immediately and refreshed. No input delay occurs.



Program Example 2:

When X0 = ON, the output signal Y0~Y7 (8 points) are sent to output terminal immediately and refreshed.



API	Mnemonic			Operands	Function											Controllers			
51		REFF	P	<div>n</div>	Refresh and Filter Adjust											PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	REFF, REFFP: 3 steps			
n					*	*													

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

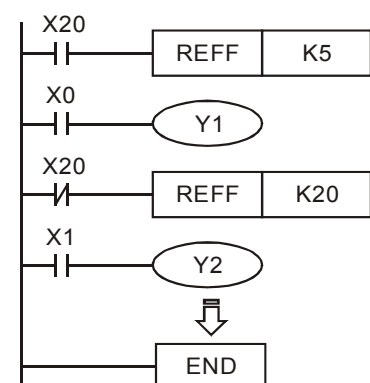
n: Response time setting, in units of ms

Explanation:

1. ELC provides an input filter to prevent noise or interference from creating a false reading. The input filters of X0~X17 inputs of ELC are digital filters, using REFF can adjust the response time of the input filters. **n** set D1020 directly and adjust the reaction time of X0~X7.
2. When power ELC turns OFF → ON or the END instruction is reached, the response time is dictated by the value of D1020.
3. During program execution, setting the value in D1020 by using MOV instruction.
4. The response time can be changed by using the REFF instruction in the execution of the program. At this time, the response time specified by REFF instruction will be moved into D1020 and will be adjusted for the next scan.
5. Valid range of operand **n** : **n**=0~1,000

Program Example:

1. When the power to ELC turns from OFF → ON, the response time of X0~X7 inputs is dictated by the value in D1020.
2. When X20=ON, REFF K5 instruction is executed, response time will change to 5 ms and will take affect the next scan.
3. When X20=OFF, the REFF instruction will not be executed, the response time will change to 20ms and will take affect the next scan.

**Points to note:**

Response time is ignored (no delay set) if using any external interrupts or high-speed counters or SPD instruction.

API	Mnemonic	Operands	Function	Controllers			
52	MTR	(S) (D ₁) (D ₂) (n)	Input Matrix	PB	PC	PA	PH

Type	Bit Devices				Word devices											Program Steps
OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MTR: 9 steps
S	*															
D ₁		*														
D ₂		*	*	*												
n					*	*										

Operands:

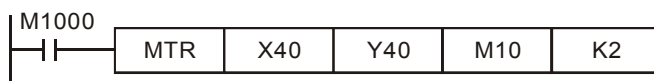
S: Head address of input matrix **D₁:** Head address of output matrix **D₂:** Corresponding head address of matrix scan **n:** Number of banks for the matrix

Explanations:

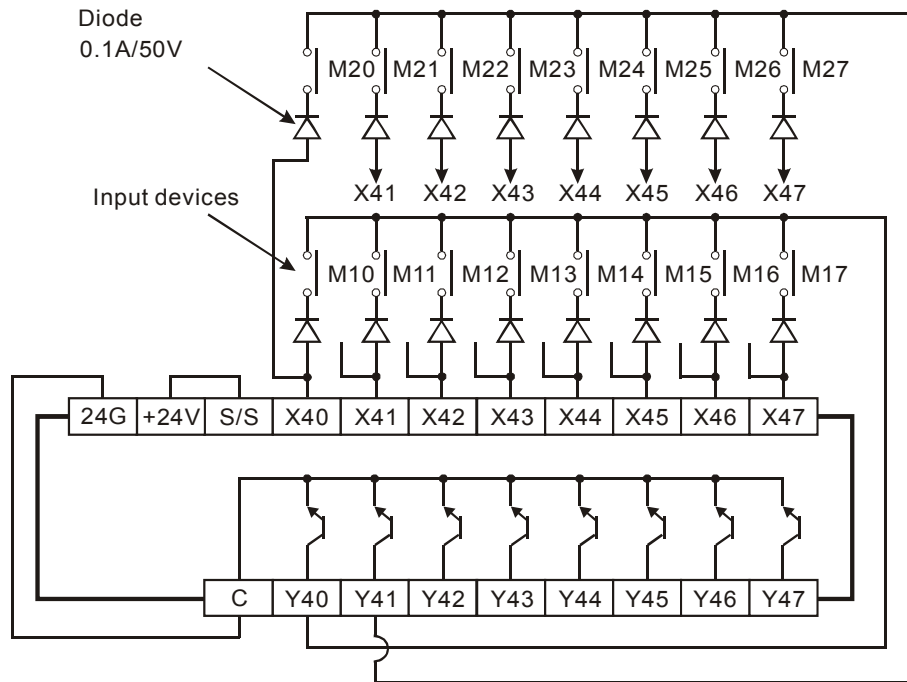
- S** is the head address that specify all inputs of the matrix. Once the input is specified, a selection of 8 continuous input devices is called as “input matrix”. **D₁** is the head address of transistor output Y of the matrix.
- This instruction allows a selection of 8 continuous input devices (head address **S**) to be used multiple (**n**) times. Each input has more than one and different **D₁** signal being processed. Each set of 8 input signals are grouped into a “rows” and there are **n** number of rows. Each row is selected by the quantity of outputs from **D₁**, used to achieve the matrix are equal to the number of rows **n**. The result is stored in a matrix-table which starts at corresponding head address **D₂**.
- The maximum inputs can achieve 64 inputs (8 inputs x 8 rows).
- When this instruction is used on an interrupt format, processing each row of inputs every 25msec. This would result in an 8 rows matrix, i.e. 64 inputs (8 inputs x 8 rows) being read in 200msec. Hence, this instruction is not available for the input signal which its ON/OFF speed is over than 200ms.
- It is recommended to use special auxiliary relay M1000(normally open contact).
- After the completion of performing MTR instruction, the instruction execution completed flag M1029 is turned ON and this flag is automatically reset when the MTR instruction is turned OFF.
- This instruction can only be used ONCE.

Program Example:

When ELC run, MTR instruction starts to execute. The external 2 rows, total 16 devices are read by order and the result are stored in the internal relay M10~M17, M20~M27.

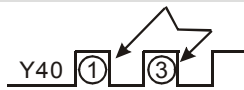


The figure below is an example wiring diagram for the operation of MTR instruction. The external 2 rows consist of X40~47 and Y40~41 and total 16 devices correspond to the internal relay M10~M17, M20~M27 are used with MTR instruction. For a general precaution to aid successful operation, diodes should be placed after each input devices. These diodes should have a rating of 0.1A, 50V.

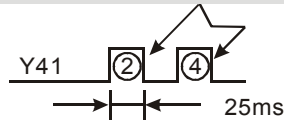


When output Y40 is ON, only those inputs in the first row are read. These results are stored in auxiliary coils M10~M17. The second step involves Y40 going OFF and Y41 coming ON and this time only inputs in the second row are read. These results are stored in M20~M27.

Read input signal in the first row



Read input signal in the second row



Processing time of each row is about 25ms

Points to note:

1. Operand **S** must be a multiple of 10, i.e. 00, 10, 20, 30... etc., so it should be X0, X10... etc. and occupies 8 continuous devices.
2. Operand **D₁** should be a multiple of 10, i.e. 00, 10, 20, 30... etc., so it should be Y0, Y10... etc. and occupies **n** continuous devices
3. Operand **D₂** should be a multiple of 10, i.e. 00, 10, 20, 30... etc., so it should be Y0, M0, S0... etc.
4. Valid range: **n=2~8**
5. Flag: M1029 Execution completed flag

API	Mnemonic			Operands			Function								Controllers				
53	D	HSCS			(S ₁)	(S ₂)	(D)	High Speed Counter Set								PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										DHSCS: 13 steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F				
S ₁					*	*	*	*	*	*	*	*	*	*	*				
S ₂												*							
D			*	*	*														

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Compare value **S₂:** Number of high-speed counter **D:** Compare result

Explanations:

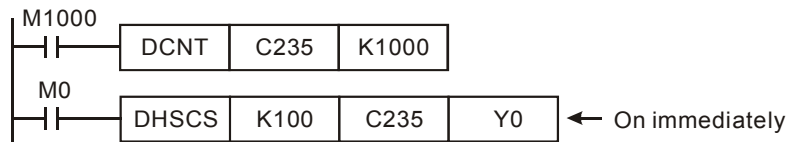
- All high-speed counters use an interrupt process, therefore, all compare results **D** are updated immediately.
- DHSCS instruction compares the current value of the selected high-speed counter **S₂** against a selected compare value **S₁**. When the counters current value equals **S₁**, then **D** = ON. Once set to ON if values **S₁** and **S₂** are not longer equal, **D** will still be ON.
- If **D** is specific as Y0~Y7, when **S₁** and **S₂** or equal, the compare result will immediately output to the external inputs Y0~Y7, and other Y devices will be affected normally by the scan cycle. However, M and S devices are immediately output and not being affected by the scan cycle.
- Operand limitation:
Operand **S₂** of PB should be C235~C238, C241~C244, C246~C249, C251~C254. Operand **S₂** of PC/PA should be C235~C244, C246~C249, C251~C254. Operand **S₂** of PH should be C235~C254.

Valid range of **D**: I010 to I060 can be set, PB models do not support interrupt function

Flag: M1150~M1333. Refer to the Remarks for details

Program Example 1:

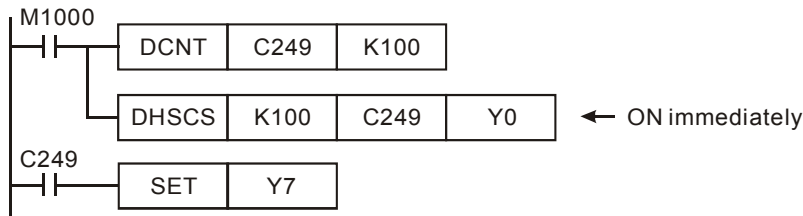
After ELC perform a RUN instruction, if M0=ON, DHSCS instruction starts to operate. Y0 = ON immediately when C235's present value stepped from 99→100 or 101→100 and be ON constantly.

**Program Example 2:**

Difference between a Y output for DHSCS instructions and general Y output:

When C249's value changes from 99→100 or 101→100, Y0 output of DHSCS is immediately output to the external Y0 using the interrupt process which is independent of the scan time. However, there will still be a delay due to the output module: relay (10ms) or transistor (10us).

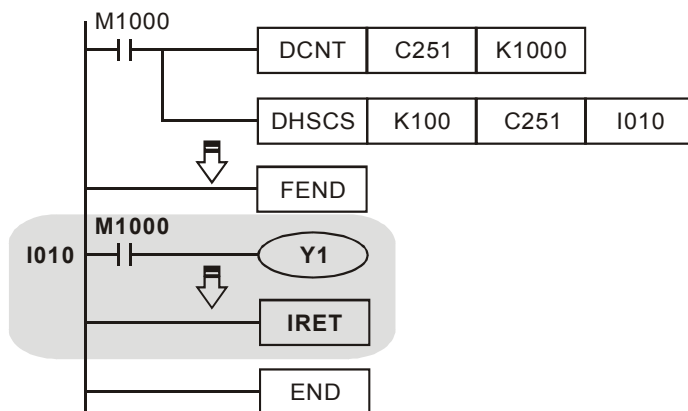
When the present value of high-speed timer C249 changes from 99 to 100, C249 will be activated, and Y7 will be ON after the END instruction based on the program scan time.



Program Example 3:

High-speed counter interrupt:

1. PC/PA/PH models using a high-speed counter interrupt
2. When using DHSCS instruction to specify I interrupt, the specified high-speed counter can not be use in other DHSCS, DHSCR, DHSZ instruction. If using it, it will result in error.
3. The interrupt pointers I010 to I060 can be used as **D** operand for DHSCS instruction and this enables the interrupt routine to be executed when the value of the specified high-speed counter reaches the value in DHSCS instruction.
4. PC/PA/PH models, there are six high-speed counter interrupts: I010, I020, I030, I040, I050, I060 (6 points) can be used. I010 is used with C235, C241, C244, C246, C247, C249, C251, C252, and C254. I020 is used with C236, C243; I030 is used with C237, C242; I040 is used with C245, C238; I050 is used with C239 and I060 is used with C240, C250.
5. When the present value of C251 changes from 99→100 or 101→100, the program will jump to the interrupt pointer I010 to execute the interrupt routine.



6. M1059 is high-speed counter interrupt disable flag

Points to note:

- The output contact of high-speed counter and the compare output of DHSCS instruction, DHSCR instruction and DHSZ instruction are all activated when there are counted inputs. If using data operation instruction, such as DADD, DMOV...etc. instructions to change the present value of high-speed counter equal to the setting value, there is comparison will be set or output because there is no counted inputs.
- PB models: total counting frequency (X0~X3) is 20 KHz.

Type	1-phase 1 input							1-phase 2 inputs			2-phase 2 inputs		
Input	C235	C236	C237	C238	C241	C242	C244	C246	C247	C249	C251	C252	C254
X0	U/D				U/D		U/D	U	U	U	A	A	A
X1		U/D			R		R	D	D	D	B	B	B
X2			U/D			U/D			R	R		R	R
X3				U/D		R	S			S			S

U: Increasing input
D: Decreasing input

A: A phase input
B: B phase input

S: Start input
R: Reset input

Input point X0 and X1 can plan to be higher speed counter and 1-phase can be up to 20KHz. But total counting frequency of these input points should be less than or equal to total frequency 20KHz. If counting input is A/B phase signal, frequency will be four times of counting frequency. Therefore, counting frequency of A/B phase is almost 5KHz.

In PB models, DHSCS and DHSCR instruction can not be used more than 4 times.

- High-speed counter provided in PC/PA/PH series models:

1-phase high-speed counter: total counting frequency is 20 KHz

Type	1-phase 1 input									1-phase 2 inputs			2-phase 2 inputs		
Input	C235	C236	C237	C238	C239	C240	C241	C242	C244	C246	C247	C249	C251	C252	C254
X0	U/D						U/D		U/D	U	U	U	A	A	A
X1		U/D					R		R	D	D	D	B	B	B
X2			U/D					U/D			R	R		R	R
X3				U/D				R	S			S			S
X4					U/D										
X5						U/D									

U: Increasing input
D: Decreasing input

A: A phase input
B: B phase input

S: Start input
R: Reset input

Input point X0 and X1 can plan to be higher speed counter and 1-phase can be up to 20KHz. But total counting frequency of these input points should be less than or equal to total frequency 20KHz. If counting input is A/B phase signal, frequency will be four times of counting frequency. Therefore, counting frequency of A/B phase is almost 5KHz.

Input X5 has two functions:

1. When M1260=OFF, C240 is general U/D high-speed counter.
2. When M1260=ON, X5 is the global reset of C235~C239.

In PC/PA series models, DHSCS, DHSCR and DHSZ instruction can not be used more than 6 times.

1. High-speed counter provided in PH series models:
 - a) For PH model, 1-phase 1 input terminals X10(C243) and X11(C245) of high-speed counter are added. The maximum input frequency can be up to 100KHz respectively. Besides, 1-phase 2 inputs terminal (X10, X11) C250 is added and the maximum input frequency can be up to 100KHz.
 - b) The available high-speed counters of PH series are X10~X11 and total bandwidth is 130KHz.

Type	1-phase 1 input											1-phase 2 inputs				2-phase 2 inputs		
Input	C235	C236	C237	C238	C239	C240	C241	C242	C243	C244	C245	C246	C247	C249	C250	C251	C252	C254
X0	U/D						U/D			U/D		U	U	U		A	A	A
X1		U/D					R			R		D	D	D		B	B	B
X2			U/D					U/D					R	R			R	R
X3				U/D				R		S				S				S
X4					U/D													
X5						U/D												
X10									U/D						U			
X11											U/D				D			

U: Increasing input

D: Decreasing input

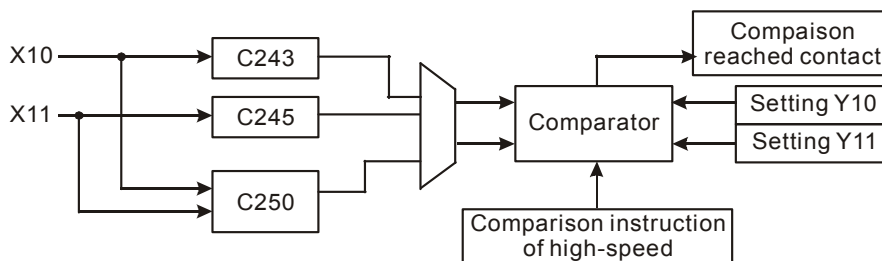
A: A phase input

B: B phase input

S: Start input

R: Reset input

- c) OFF/ON of M1243 decides count up/down of C243 and OFF/ON of M1245 decides count up/down of C245. It cannot count at rising-edge and falling-edge simultaneously.
- d) When C243 or C245 is used, C250 cannot be used. In the same way, when C250 is in use, C243 or C245 cannot be used

The relation graph of high-speed counter and high-speed comparator

- a) When DHSCS and DHSCR use additional high-speed counter, maximum high-speed instruction can use is the settings of two sets of high-speed comparison instruction. Assume that there is a set of comparison instruction of DHSCS D0 C243 Y10 is used, only a set of comparison instruction of DHSCR D2 C243 Y10 or DHSCS D4 C245 Y10 can be used.

- b) When additional high-speed counter is used by DHSZ, only a set of comparison instruction can be used.
- c) The original quantity of high-speed comparison instruction provided for PC/PA won't reduce due to above additional high-speed counter.
- d) If it needs high-speed response output for output device setting of high-speed comparison instruction (DHSCS), it is recommended to use Y10 or Y11. If using other common devices, it will delay a scan period setting or clear at most. When setting I0x0, C243 corresponds to I020, C245 corresponds to I040 and C250 corresponds to I060.
- e) The clear output device of high-speed comparison instruction (DHSCR) allows to clear counter device. But only the same counter in one instruction can be cleared.
For example: DHSCR K10 C243 C243. Besides, only three special high-speed counters (C243, C245 and C250) can use this function.

2. Select frequency mode for 2-phase 2 inputs counter mode

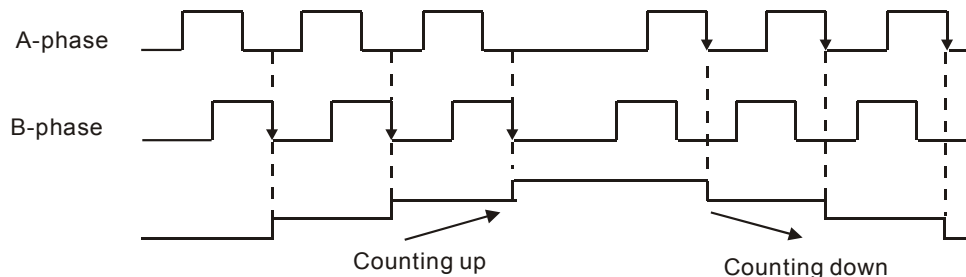
High-speed counter of PA/PB/PC/PH series is 2-phase 2 inputs counter mode and set by special device D1022 with four double frequency modes. The content value of register D1022 is loaded at the first scan time when ELC controller switch from Stop to Run status.

Device No.	Function Explanation
D1022	Double frequency mode is initial set-point when power OFF → ON
D1022=K1	Normal frequency mode
D1022=K2	Double frequency mode
D1022=K4	Four times frequency mode

Double frequency mode(2-phase 2 inputs):

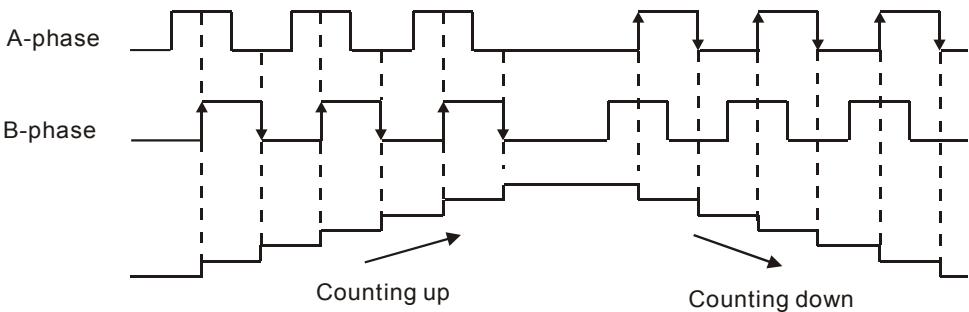
a) D1022=K1 (normal frequency)

Signal Diagram



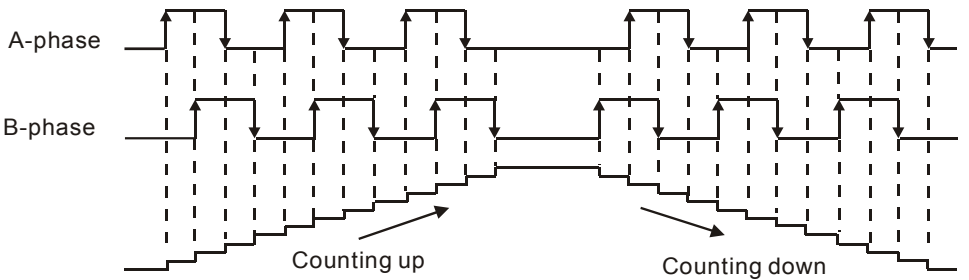
b) D1022=K2 (double frequency)

Signal Diagram



c) D1022=K4 (four times frequency)

Signal Diagram



3. Related flags of high-speed counter:

Flag	Function Explanation
M1059	High-speed counter interrupt, I010 ~ I060 masked
M1235 ~ M1245	Specify counting direction of C235 ~ C245 high-speed counter When M12□□=OFF, C2□□ counting up When M12□□=ON, C2□□ counting down
M1246 ~ M1250 M1251 ~ M1254	Monitor counting direction of C246~C250, C251~C254 high-speed counter When C2□□ counting up, M12□□=OFF. C2□□ counting down, M12□□=ON.
M1260	X5 is the reset input signal of all high-speed counters

API	Mnemonic		Operands			Function										Controllers			
54	D	HSCR	<div><div>S₁</div><div>S₂</div><div>D</div></div>			High Speed Counter Reset										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DHSCR: 13 steps		
	S ₁					*	*	*	*	*	*	*	*	*	*				
	S ₂												*						
	D		*	*	*								*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

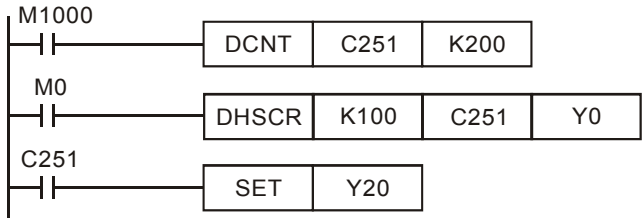
S₁: Compare value S₂: Number of high-speed counter D: Compare result

Explanation:

- HSCR compares the current value of the counter S₂ against a compare value S₁. When the counters current value changes to a value equal to S₁, then device D is reset to OFF. Once reset, even if the compare result is no longer unequal, D will still be OFF.
- If D is specified as Y0~Y7, when the compare value and the present value of high-speed counter are equal, the compare result will immediately output to the external outputs Y0~Y7 (specified Y output will be reset), and other Y devices will be affected by the normal scan cycle. However, M and S devices are also immediately output.
- Operand S₂ of PB should be C235~C238, C241~C244, C246~C249, C251~C254. Operand S₂ of PC/PA should be C235~C244, C246~C249, C251~C254. Operand S₂ of PH should be C235~C254.

Program Example 1:

- When M0=ON and C251's present value stepped from 99→100 or 101→100, Y0 will be reset to OFF.
- When C251's present value changes from 199 to 200, the contact C251 will be ON and force Y20=ON, based on normal program scan time.



API	Mnemonic		Operands				Function	Controllers			
55	D	HSZ	(S₁)	(S₂)	(S)	(D)	HSC Zone Compare	PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DHSCS: 17 steps
S ₁						*	*	*	*	*	*	*	*	*	*	*	
S ₂						*	*	*	*	*	*	*	*	*	*	*	
S													*				
D			*	*	*												

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

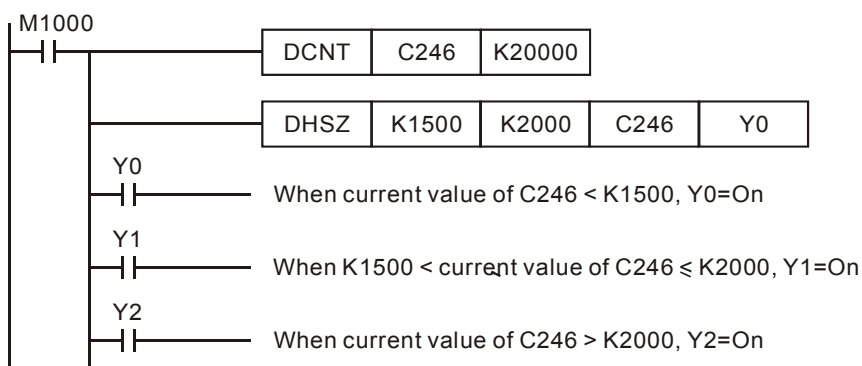
S₁: Low-limit value of zone comparison **S₂**: High-limit value of zone comparison **S**: Number of high-speed counter **D**: compared result (occupies 3 continuous bit devices)

Explanations:

1. **S₁** should be equal to or smaller than **S₂** ($S_1 \leq S_2$).
2. Output operation is not affected by the scan time.
3. All outputs and zone comparison all use interrupt operation.
4. Operand **S₂** of PC/PA should be C235~C244, C246~C249, C251~C254. Operand **S₂** of PH should be C235~C254.
5. Flag: M1059~M1260 (Refer to DHSCS for more details)

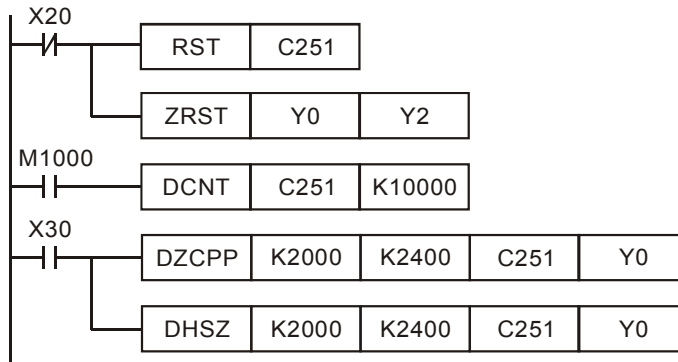
Program Example 1:

1. When **D** is specified as Y0, then Y0~Y2 will be occupied automatically.
2. When DHSZ instruction is executed and high-speed counter C246 is counting, if the high and low limit value is reached, one of Y0~Y2 will be ON.

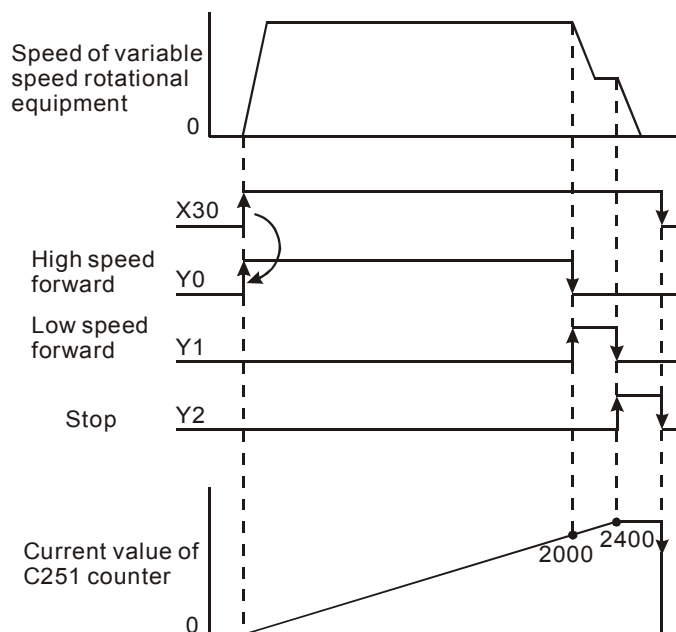
**Program Example 2:**

1. When using DHSZ instruction to control stop, high, or low speed, C251 is set as an AB phase high-speed counter.

2. When X30=ON, DHSZ will turn Y0=ON when the current value $\leq K2,000$. In order to improve this, use the DZCPP instruction to compare C251 against K2,000 when the program is RUN at the beginning. When counting the current value $\leq K2,000$, Y0=ON and DZCPP instruction is Pulse execution. Instruction DZCPP can only be executed ONCE in a program and Y0 will be still be ON.
3. When drive contact X20=OFF, Y0~Y2 will be reset to OFF.



Timing diagram



API	Mnemonic	Operands			Function											Controllers			
56	SPD	S₁	S₂	D	Speed Detection											PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F				
S ₁		*															SPD: 7 steps		
S ₂						*	*	*	*	*	*	*	*	*	*	*			
D												*	*	*					

Operands:

S₁: External pulse input **S₂**: Pulse time(ms) **D**: Result (occupies 5 continuous devices)

Explanations:

1. **S₁**: Specify the input of external pulse, it only can be X1~X2
2. PC/PA models, if X1 or X2 are used in a SPD instruction, then the related high-speed counters or external interrupts I101, I201 can not be used.
3. Count the number of pulses received at the inputs specified by **S₁** during the time specified by **S₂** (ms) and store the result in the register specified by **D**.
4. **D** occupies 5 registers, **D+1**, **D** indicate the detection value of previous pulse, **D+3**, **D+2** indicate the present accumulated count value of pulses and **D+4** indicates the remaining count time, the max. is 32767ms.
5. Maximum frequency: X1=20KHz, X2=10KHz, Total frequency is less than 20KHz
6. This instruction is mainly used to obtain a proportional value of rotation speed. The result **D** and rotation speed are in proportion. The following equation can be used to obtain the rotation speed of motor.

$$N = \frac{60(D0)}{nt} \times 10^3 (rpm)$$

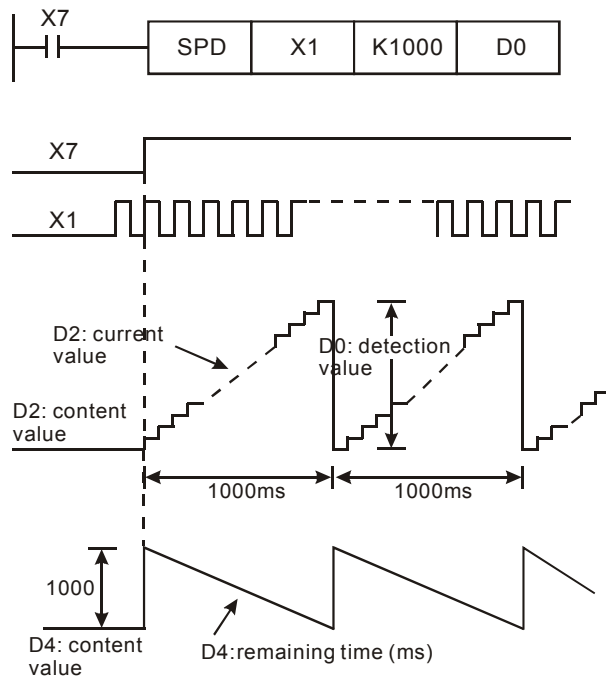
N: Rotation speed

n: The number of pulses per rotation of rotation equipment

t: Detection time specified by **S₂** (ms)

Program Example:

1. When X7=ON, D2 will count the high-speed pulse input from X1. After 1,000ms, it will stop counting automatically and store the result in D0.
2. After 1000ms of counting has completed, the content of D2 will be reset to 0. When X7 turns ON again, D2 will recount.



API	Mnemonic			Operands			Function								Controllers			
57	D	PLSY		S₁	S₂	D	Pulse Output								PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F			
S ₁						*	*	*	*	*	*	*	*	*	*	*	PLSY: 7 steps DPLSY: 13 steps		
S ₂						*	*	*	*	*	*	*	*	*	*	*			
D			*																

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Pulse output frequency **S₂**: Pulse output number **D**: External output (only Y0 and Y1 can be specified)

Explanations:

- The PLSY instruction can be used twice in the program.
- S₁** specified as the pulse output frequency

Output frequency range of each series models		
Models	PB series models	PC/PA/PH series models
Output frequency range	1~10,000Hz	1~32,000Hz

- S₂** specified as the pulse output number.

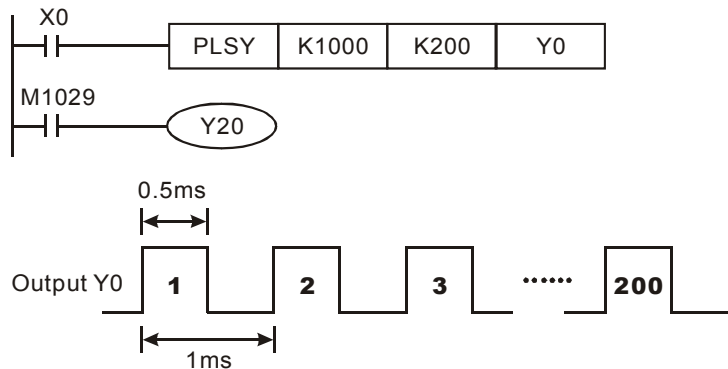
16-bit instruction: 1~32,767. 32-bit instruction: 1~2,147,483,647.

Continuous pulses	M1010 (Y0) ON, pulse output is continuous M1023 (Y1) ON pulse output is continuous
-------------------	---

- When PLSY instruction has been executed, the specified quantity of pulses **S₂** will be output through the pulse output device **D** at the specified pulse output frequency **S₁**.
- When the PLSY instruction is used in a program, the outputs used in the PLSY instruction cannot be used in the PWM instruction or PLSR instruction.
- After the Y0 pulse output is complete, M1029 = ON, after the Y1 pulse output is complete, M1030 = ON. When the PLSY instruction is OFF (not executing), M1029 or M1030 = OFF.
- The execution completed flag M1029, M1030 should be cleared by the user after the execution of the instruction has been completed.
- While the PLSY instruction is being executed, the output will not be affected if **S₂** is changed. To change the pulse output number, stop the PLSY instruction, then change the pulse number.
- S₁** can be changed while the PLSY instruction is being executed.
- The ratio of OFF time and ON time of the pulse output is 1:1.
- If operand **S₁**, **S₂** uses index F, then only 16-bit instruction is available.

Program Example:

1. When X0=ON, the pulse of 1KHz for 200 times is generated from output Y0, after the pulse has been completed, M1029=ON trigger Y20=ON.
2. When X0=OFF, pulse output Y0 will immediately stop. When X0 turns ON again, the pulse output will restart.

**Points to note:**

1. Flags description:
 - M1010: When M1010=ON, Y0 will output continuous pulses. When M1010=OFF, the pulse output numbers of Y0 are decided by **S₂**.
 - M1023: When M1023=ON, Y1 can output limitless continuous pulses. When M1023=OFF, the pulse output numbers of Y1 are decided by **S₂**.
 - M1029: M1029= ON after Y0 pulse output complete.
 - M1030: M1030= ON after Y1 pulse output complete.
 - M1078: Y0 pulse output stop immediately.
 - M1079: Y1 pulse output stop immediately.
2. Special D registers description:
 - D1030: Present total number pulses of output Y0 (LOW WORD).
 - D1031: Present total number pulses of output Y0 (HIGH WORD).
 - D1032: Present total number pulses of output Y1 (LOW WORD).
 - D1033: Present total number pulses of output Y1 (HIGH WORD).
3. When several pulse output instructions (PLSY, PWM, PLSR) use Y0 as the output pulse in the same program, and simultaneously are executed in the same scan cycle, ELC will perform the instruction which has fewest step numbers.

API	Mnemonic	Operands			Function										Controllers			
58	PWM	<div><div>S₁</div><div>S₂</div><div>D</div></div>	Pulse Width Modulation										PB	PC	PA	PH		

OP	Type	Bit Devices				Word devices										Program Steps							
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	PWM: 7 steps						
	S ₁					*	*	*	*	*	*	*	*	*	*	*							
	S ₂					*	*	*	*	*	*	*	*	*	*	*							
	D		*																				

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

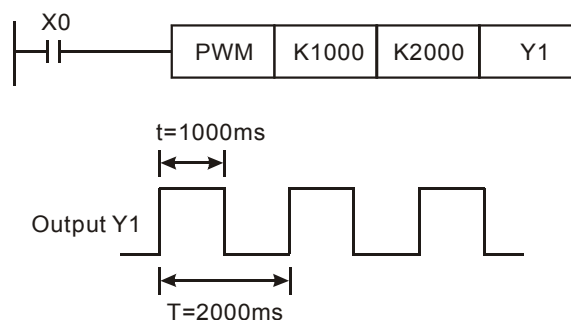
S₁: Pulse output width **S₂**: Pulse output period **D**: Pulse output device (only be specified as Y1)

Explanations:

- S₁** is specified as pulse output width as t:0~32,767ms. **S₂** is specified as pulse output period as T:1~32,767ms, **S₁ ≤ S₂**.
- PB model, The **D** output device cannot be the same as the output of PLSY or PLSR instruction when the PWM instruction is being used in a program.
- PC/PA models, When several pulse output instructions (PLSY, PWM, PLSR) use Y1 as the output pulse in the same program, and simultaneously are executed in the same scan cycle, ELC will perform the instruction which has fewest step numbers.
- If **S₁ > S₂**, an operation error will occur, M1067 =ON. When **S₁** is 0, there is no pulse output from the pulse output device. When **S₁ = S₂**, the pulse output device will be always ON.
- S₁, S₂** can be changed during the execution of PWM instruction.
- When M1070=ON, the pulse unit is 100μs, when M1070=OFF, the pulse unit is 1ms.

Program Example:

When X0=ON, Y1 output the following pulse. When X0=OFF, output Y1 will also turn OFF.

**Note:**

When several pulse output instructions (PLSY, PWM, PLSR) use Y1 as the output pulse in the same program, and simultaneously are executed in the same scan cycle, ELC will perform the instruction which has fewest step numbers.

API	Mnemonic			Operands				Function				Controllers			
59	D	PLSR		(S₁)	(S₂)	(S₃)	(D)	Pulse Ramp				PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	PLSR: 9 steps DPLSR: 17 steps			
S ₁					*	*	*	*	*	*	*	*	*	*	*				
S ₂					*	*	*	*	*	*	*	*	*	*	*				
S ₃					*	*	*	*	*	*	*	*	*	*	*				
D		*																	

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Maximum speed (HZ) **S₂:** Number of pulses **S₃:** Acceleration/Deceleration time (ms)

D: Pulse output device (only Y0 and Y1 can be specified)

Explanations:

- S₁:** the maximum frequency (Hz) for 16-bit instruction: 10 to 32,767 Hz. For 32-bit instruction: 10 to 32,767 Hz. The maximum speed must be multiples of 10, if not, the ones digit will be discarded automatically. 1/10 of the maximum speed is the max. variation per accel/decel step. Note: This condition meets acceleration requirements of the step motor and would not result in the step motor damage.
- S₂:** Number of pulses for 16-bit instruction: 110~32,767, For 32-bit instruction: 110~2,147,483,647. The minimum set-point is 110.
- S₃:** Acceleration/Deceleration time (ms). Set-points: below 5,000ms. The acceleration time and the deceleration time are the same setting.
 - The accel/decel time must be larger than 10 times the maximum scan time (contents of D1012). If the set-point is below this, the slope of the accel/decel may be inaccurate.
 - Minimum set-point of the accel/decel time could be obtained from the following equation.

$$(S_3) \geq \frac{90000}{(S_1)}$$

If the set-point is smaller than the result of the above-mentioned equation, the accel/decel time will be greater, and if the set-point is smaller than the 90000/ **S₁**, the result value of 90000/ **S₁** will be treated as its regular set-point.

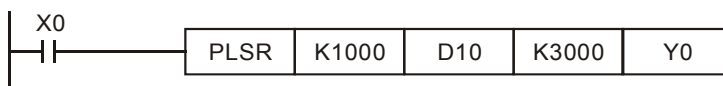
- Maximum set-point of the accel/decel time could be obtained from the following equation.

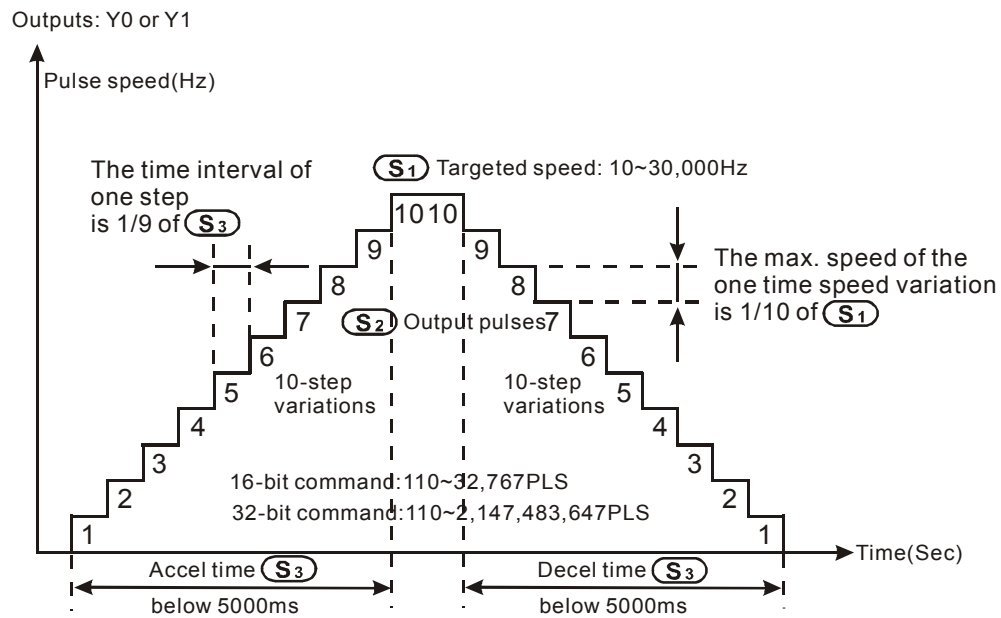
$$(S_3) \leq \frac{(S_2)}{(S_1)} \times 818$$

- d) Number of the accel/decel speed variation steps is fixed 10 steps. If the input acceleration/deceleration time is greater than the maximum set-point, the maximum set-point will be treated as its regular set-point. If the set-point is smaller than the minimum set-point, the minimum set-point will be treated as its regular set-point.
4. The acceleration is conducted when the pulse goes from the static status to reaching its targeted speed, and gets slower the closer it gets to its targeted distance. The pulse will stop its output once the targeted distance is reached.
5. When the PLSR instruction has been executed, its output frequency is first raised up in increments of 1/10 of the maximum frequency $S_1/10$ and the time of each output frequency is fixed at 1/9 of S_3 .
6. The output will not be affected if S_1 , S_2 or S_3 are changed when PLSR instruction is being executed.
7. After the first group (Y0) of pulses have been output, set by S_2 , M1029=ON. After the second group (Y1) of pulses have been output, set by S_2 , M1030=ON. When the instruction PLSR is activated again, M1029 or M1030 = OFF.
8. The output pulse numbers of the Y0/Y1 are stored in the special registers D1031, D1030 / D1033, D1032.
9. During the acceleration of each step, the pulse numbers (each frequency x time) may not all be integer values, but the output operation of ELC is conducted in whole integer number. Therefore, the time of each interval may not be the exact same and may have some deviation. The offset is determined by the frequency value and by discarding the decimal point value. In order to ensure the output pulse numbers are correct, ELC will fill in pulses as need to keep any deviation to a minimum.
10. Use transistor output as output module.
11. Flag: M1029, M1030. Execution Completed flag.

Program Example:

1. When X0=ON, the maximum frequency of instruction PLSR is 1,000Hz. The D10 is total quantity of output pulse, the accel/decel time is 3,000ms and pulses output from output Y0. The output frequency change 1,000/10 Hz every step. The frequency pulse of each is fixed as 3,000/9.
2. When X0 = OFF, the output will be interrupted, and when turned ON again, the pulses will recount by zero.



**Note:**

When several pulse output instructions (PLSY, PWM, PLSR) use Y0 as the output pulse in the same program, and simultaneously are executed in the same scan cycle, ELC will perform the instruction which has fewest step numbers.

API	Mnemonic	Operands			Function											Controllers			
60	IST	<div>S</div>	<div>D₁</div>	<div>D₂</div>	Manual/Auto Control											PB	PC	PA	PH

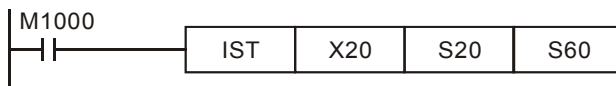
OP	Type	Bit Devices				Word devices											Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	IST: 7 steps			
S		*	*	*																
D ₁					*															
D ₂					*															

Operands:

S: The starting input number (Operand **S** will occupy 8 continuous devices). **D₁** The smallest number for the designated-status step point in auto mode. **D₂:** The greatest number for the designated-status step point in auto mode.

Explanations:

- The IST is a convenient instruction made specifically for the initial state of the step function control procedure.
- PB model, The range **D₁** and **D₂** = S20~S127 and **D₁** < **D₂**. PC/PA/PH models, The range **D₁** and **D₂** = S20~S899 and **D₁** < **D₂**.
- IST instruction can only be used one time in a program.

Program Example 1:

- S:**
- | | |
|--|-------------------------------------|
| X20: Individual operation (Manual operation) | X24: Continuous operation |
| X21: Zero point return | X25: Zero point return start switch |
| X22: Step operation | X26: Start switch |
| X23: One cycle operation | X27: Stop switch |

- When the IST instruction is executed, the following special auxiliary relay will switch automatically.

M1040: Movement inhibited	S0: Manual operation/initial state step point
M1041: Movement start	S1: Zero point return/initial state step point
M1042: Status pulse	S2: Auto operation/initial state step point
M1047: STL monitor enable	
- When IST instruction is used, S10~S19 are for zero point return operation and the step point of this state can't be used as a general step point. However, when using S0~S9 step points, S0 initiates "manual operation", S1 initiates "zero point return operation" and S2 initiates "auto

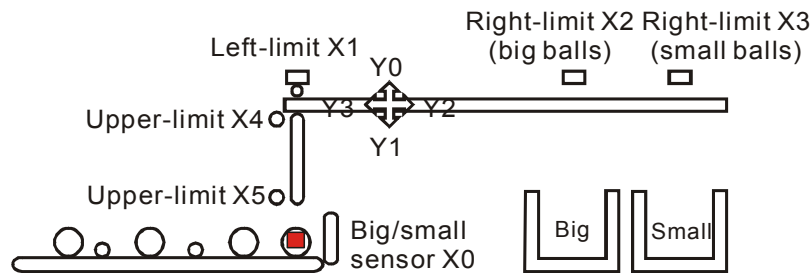
operation". Thus, there should be three circuits of these three initial state step points written first in the program.

3. When switching to S1 (zero point return mode), zero point return won't have any actions once any of S10~S19 = ON.
4. When switching to S2 (auto operation mode), auto operation won't have any actions once when S is between D_1 to D_2 = ON or if M1043=ON

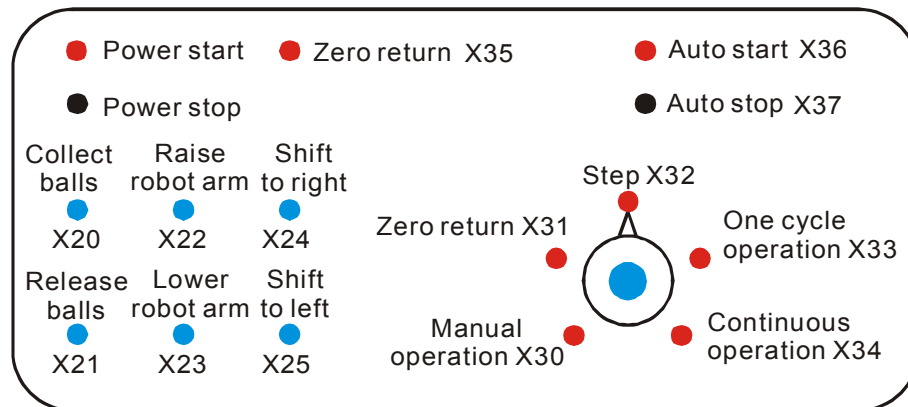
Program Example 2:

The Robot arm control (use IST instruction):

1. Motion request: In the example, two kinds of balls (big and small) are separated and moved to different boxes.
2. Motion of the Robot arm: lower robot arm, collect balls, raise robot arm, shift to right, lower robot arm, release balls, raise robot arm, shift to left to finish motion in order.
3. I/O Device

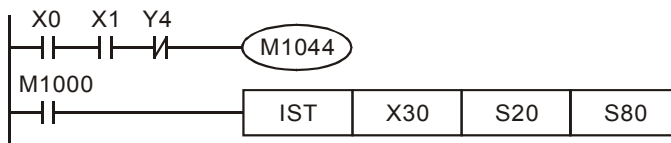


4. Control panel

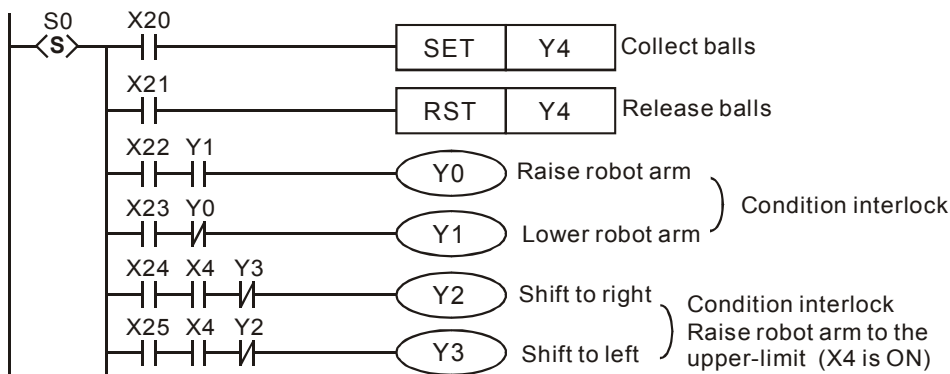


- a) Big/small sensor X0.
- b) The left-limit of the robot arm X1, the right-limit X2 (big balls), the right-limit X3 (small balls), the upper-limit X4, and the lower-limit X5.
- c) Raise robot arm Y0, lower robot arm Y1, shift to right Y2, shift to left Y3, and collect balls Y4.

5. START circuit:

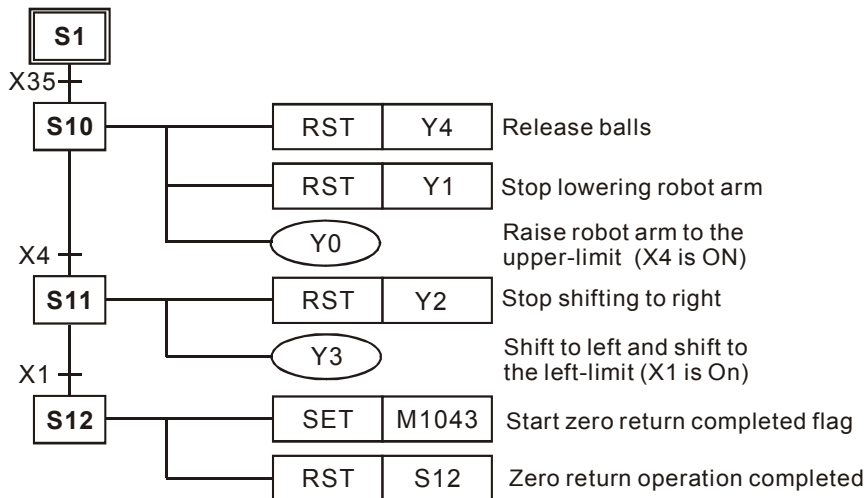


6. Manual operation mode:

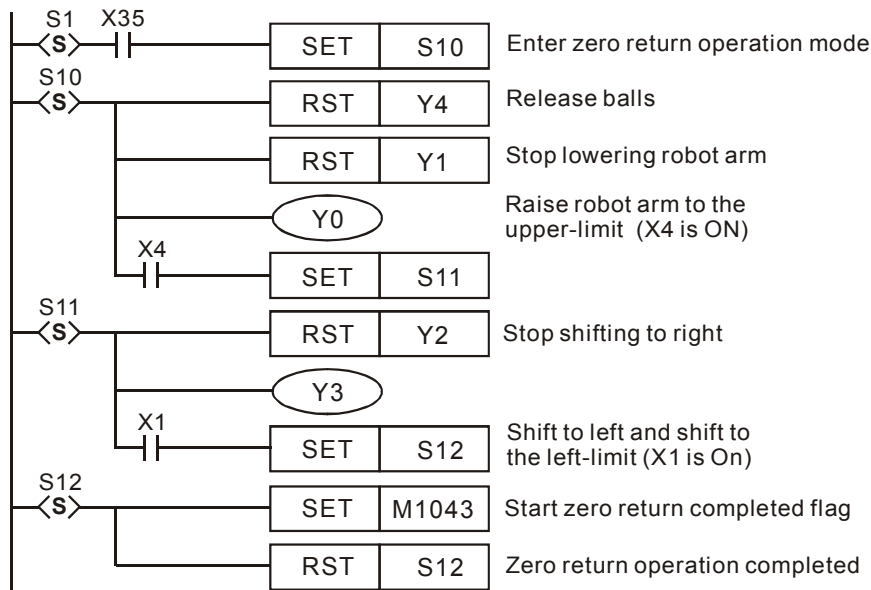


7. Zero point return mode:

a) SFC figure:

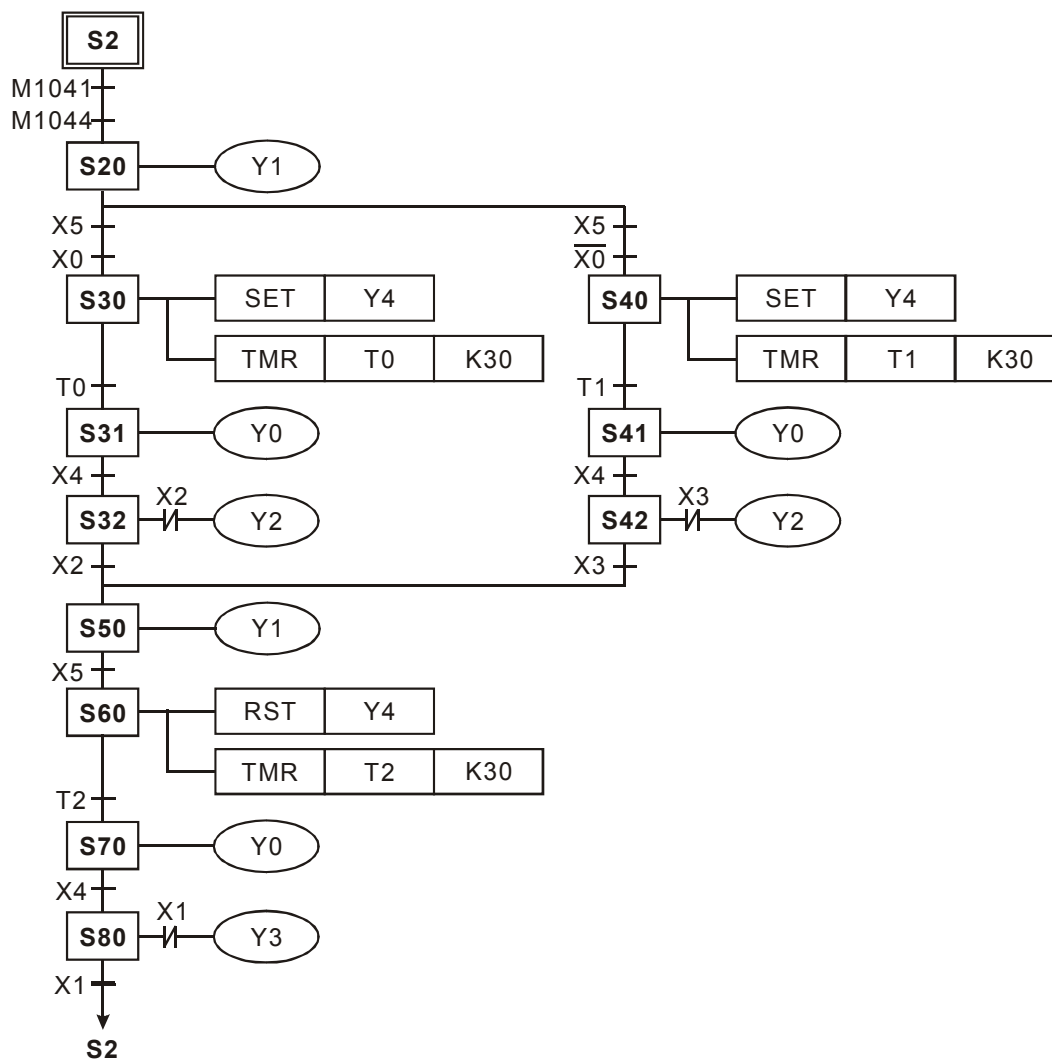


b) Ladder Diagram:

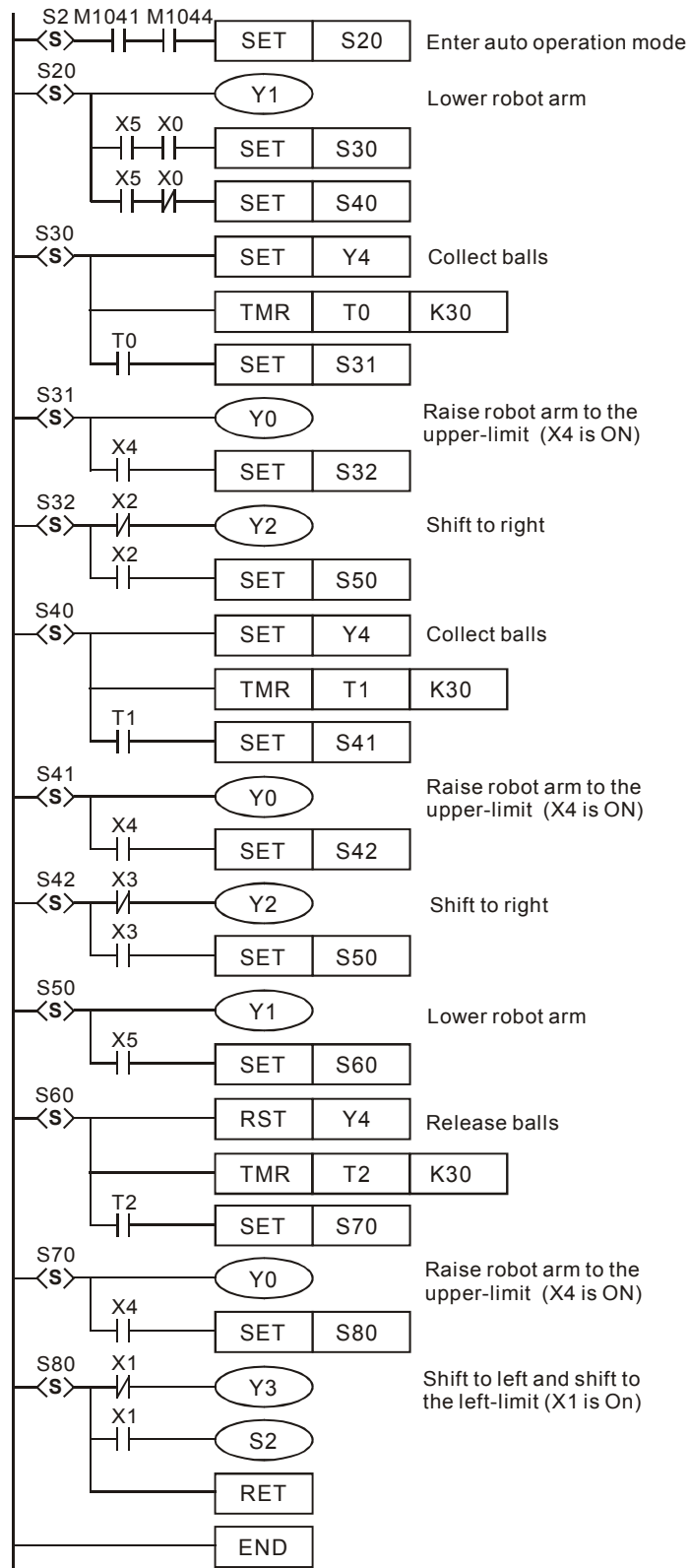


8. Auto operation (step/one-cycle/continuous operation modes):

a) SFC figure:



b) Ladder Diagram:



Flag explanation:

M1040:

Step point movement disabled. When M1040=ON, all movements of the step point are disabled.

1. **Manual operation mode:** M1040 = ON.
2. **Zero point return mode/one cycle operation mode:** Pressing the STOP button and pressing START button again, M1040 = ON.
3. **Step operation mode:** M1040 = ON, and will only be OFF when the START button is pressed.
4. **Continuous operation mode:** When ELC goes from STOP→RUN, M1040 = ON, and will be OFF when the START button is pressed.

M1041:

Step point movement start. the special auxiliary relay that reflects the movement of the primary step point (S2) to the next step point.

1. **Manual operation mode/Zero point return mode:** M1041 = OFF.
2. **Step operation mode/One cycle operation mode:** M1041 = OFF when the START button is pressed.
3. **Continuous operation mode:** Stays ON when the START button is pressed, and turns OFF when the STOP button is pressed.

M1042:

START pulse: Only once pulses will be sent out when the button is pressed.

M1043:

Zero point return complete: Once M1043 =ON, indicates that the RESET motion has been executed.

M1044:

Conditions of the origin: In continuous operation mode, conditions of the origin, M1044= ON to execute the initial step point (S2) moving to the next step point.

M1045:

All output reset inhibit. If executing conditions:

1. From manual control S0 to zero point return S1
2. From auto operation S2 to manual operation S0
3. From auto operation S2 to zero point return S1
 - a) When M1045=OFF and one of S of **D₁~D₂** is ON, step point of SET Y output and actions will be cleared to OFF.
 - b) When M1045 =ON, SET Y output will be reserved, and step point during action will be cleared to

OFF.

- c) If executing from zero point return S1 to manual operation S0, no matter if M1045=ON or M1045=OFF, SET Y output will be reserved, and step point action will be cleared to OFF.

M1046:

When STL action = ON: If one of step point S is ON, M1046=ON. After M1047 = ON, M1046 = ON once one of S is ON. Besides, 8 prior points numbers is ON of step point S will be recorded in D1040~D1047.

M1047:

STL monitor enabled. When IST instruction starts executing, M1047 will be forced to be ON and it will be forced to ON for each scan time once IST instruction is still ON. This flag is used to monitor all S.

D1040~D1047:

ON state number 1-8 of step point S.

API	Mnemonic			Operands				Function				Controllers			
61	D	SER	P	(S ₁)	(S ₂)	(D)	(n)	Search a Data Stack				PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S ₁							*	*	*	*	*	*	*			SER, SERP: 9 steps DSER, DSERP: 17 steps
S ₂					*	*	*	*	*	*	*	*	*	*	*	
D								*	*	*	*	*	*			
N					*	*							*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

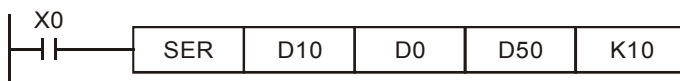
S₁: Starting source **S₂**: Compare value **D**: Starting destination for storing compared result
(occupies 5 continuous devices) **n**: Number of devices to compare

Explanations:

- S₁** specifies the starting registers to compare, **n** specifies how many registers to compare to the value specified by **S₂**, and the compared result is stored in destination registers specified by **D**.
The data is compared in algebra format.
- If operand **S₂** uses index F, it is only available in a 16-bit instruction.
- The range of operand **n**: **n**=1~256 (16-bit instruction), **n**=1~128 (32-bit instruction)

Program Example:

- When X0=ON, the data stack D10~D19 are compared against D0 and the result is stored in D50~D54. If there are no equal values, the content of D50~D52 will be 0.
- The largest value of all compared data will be record in D53 and the samllest value of all compared data will be record in D54. When there are more then one largest value and smallest value, only the number of the largest value will be recorded.



	S ₁	Content value	Compare data	Data number	Result	D	Content value	Explanation
n	D10	88	S ₂ D0=K100	0		D50	4	The total data numbers of equal value
	D11	100		1	Equal	D51	1	The number of the first equal value
	D12	110		2		D52	8	The number of the last equal value
	D13	150		3		D53	7	The number of the smallest value
	D14	100		4	Equal	D54	9	The number of the largest value
	D15	300		5				
	D16	100		6	Equal			
	D17	5		7	Smallest			
	D18	100		8	Equal			
	D19	500		9	Largest			

API	Mnemonic		Operands				Function				Controllers			
62	D	ABSD	S ₁	S ₂	D	n	Absolute Drum Sequencer				PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ABSD: 9 steps DABSD: 17 steps		
S ₁							*	*	*	*	*	*	*					
S ₂												*						
D		*	*	*														
n					*	*												

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

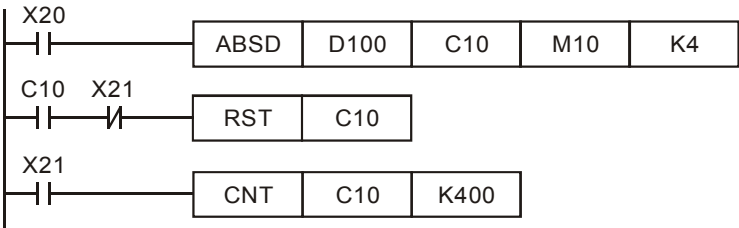
S₁: Starting device of the compared data table **S₂**: Number of counter **D**: Starting destination of compared result **n**: Groups of multi-step comparison (**n**=1~64)

Explanations:

- 1. The ABSD instruction is a multi-step comparison used in absolute cam control.
- 2. **S₂** of DABSD specifies a high-speed counter. However, when the current value of the high-speed counter is compared against the set-point value, the result can not output immediately, because it is influenced by the scan time. If immediate output is required, use the DHSZ instruction.
- 3. When operand **S₁** uses KnX, KnY, KnM, KnS, K4 only a 16-bit instruction can be used, K8 should be specified as 32-bit instruction.

Program Example:

- 1. Before executing the ABSD instruction, preload the set-point values into D100~D107. The content of the even number D register is the lower-limit value and the content of the odd number D register is the upper-limit value.
- 2. When X20=ON, the current value of counter C10 is compared against the upper and lower-limit value of D100~D107 four groups. The compared result is displayed in M10~M13.
- 3. When X20=OFF, the origin ON/OFF state of M10~M13 will be unchanged.

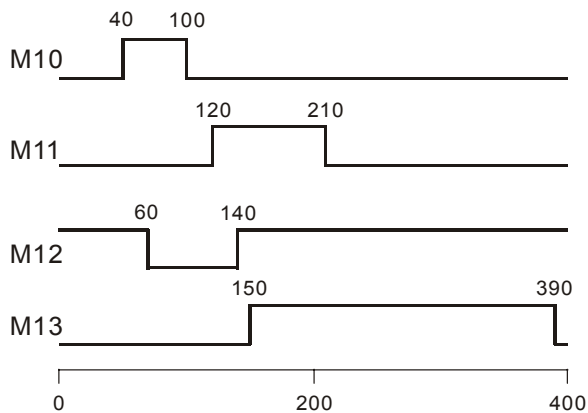


4. M10~ M13 = ON when the current value of C10 is equal to or higher than the lower-limit value and equal to or lower than the upper-limit value.

Lower-limit value	Upper-limit value	Current value of C10	Output
D100= 40	D101=100	$40 \leq C10 \leq 100$	M10=ON
D102=120	D103=210	$120 \leq C10 \leq 210$	M11=ON
D104=140	D105=170	$140 \leq C10 \leq 170$	M12=ON
D106=150	D107=390	$150 \leq C10 \leq 390$	M13=ON

5. When the lower-limit value is higher than the upper-limit value, if the current value of C10 is higher than the lower-limit value($C10 > 140$) and lower than the upper-limit value ($C10 < 60$), M12=ON.

Lower-limit value	Upper-limit value	Current value of C10	Output
D100= 40	D101=100	$40 \leq C10 \leq 100$	M10=ON
D102=120	D103=210	$120 \leq C10 \leq 210$	M11=ON
D104=140	D105= 60	$60 \leq C10 \leq 140$	M12=OFF
D106=150	D107=390	$150 \leq C10 \leq 390$	M13=ON



API	Mnemonic	Operands				Function										Controllers			
63	INCD	S₁	S₂	D	n	Incremental drum sequencer										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	INCD: 9 steps		
S ₁								*	*	*	*	*	*	*					
S ₂													*						
D			*	*	*														
n						*	*												

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Starting device of the compared data table **S₂**: Number of counter **D**: Starting number of compared result **n**: Groups of multi-step comparison (**n**=1~64)

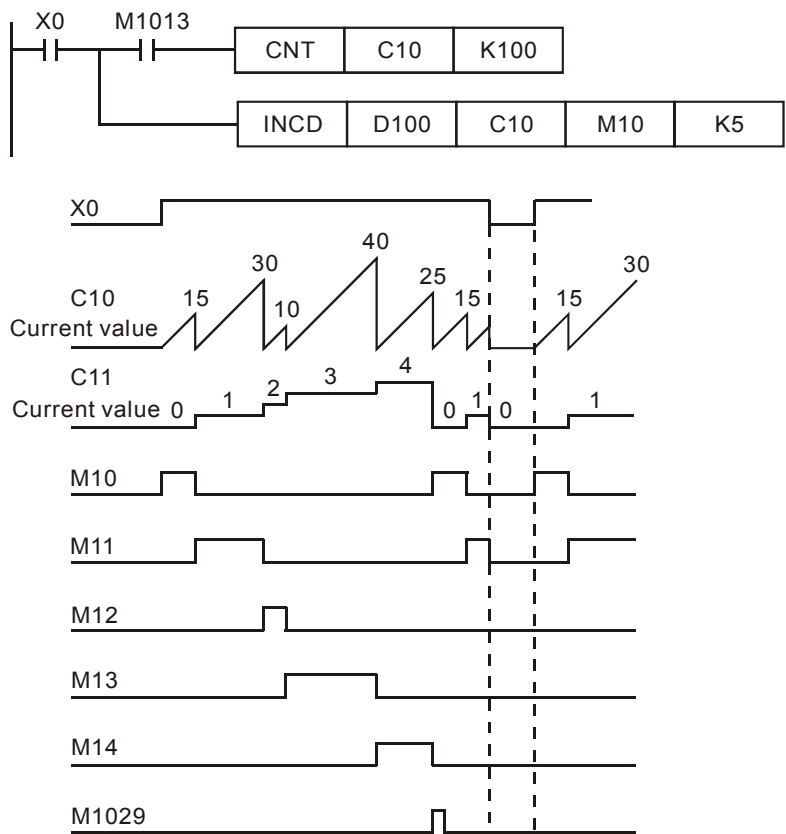
Explanations:

1. The INCD instruction is a multi-step comparison instruction and usually used in relative cam control.
2. The current value of **S₂** is compared against the set-point value of **S₁**. Once the current value is equal to the set-point value, the current value of **S₂** will be reset to 0 and be compared again. The return times will be stored in **S₂+1**.
3. When the comparison of **n** groups data has been completed, the execution completed flag M1029 = ON for one scan cycle.
4. When operand **S₁** is specified as KnX, KnY, KnM, KnS, K4 should be specified.
5. In 16-bit instructions, operand **S₂** should be C0~C198 and will occupy 2 continuous counters.
6. **Flag**: M1029 execution completed flag

Program Example:

1. Before executing the INCD instruction, preload the set-point values into D100~D104 in advance. D100=15, D101=30, D102=10, D103=40, D104=25.
2. The current value of counter C10 is compared against the set-point value of D100~D104. Once the current value is equal to the set-point value, the current value of C10 will be reset to 0 and will be compared again.
3. The current counting will be stored in C11.
4. When the content of C11 increase 1, M10~M14 will also change in response. Please refer to the following timing diagram.
5. When the comparison of 5 groups of data has been completed, the execution completed flag M1029 = ON for one scan cycle.

6. When X0 turns from ON →OFF, C10 and C11 will all be reset to 0 and M10~M14 =OFF. When X0 turns ON again, this instruction will be executed again.



API	Mnemonic	Operands		Function										Controllers				
64	TTMR	<div>D</div>	<div>n</div>	Alternate Timer										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	
Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	TTMR: 5 steps		
D													*					
n					*	*												

Operands:

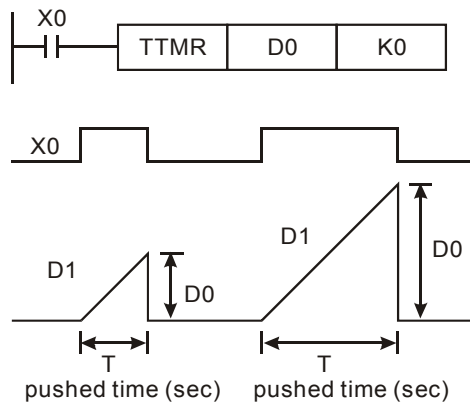
D: Device number for storing the ON time of the input **n:** Multiple set-point ($n=0\sim2$)

Explanations:

1. The ON time of the external button switch is measured and stored in **D + 1**, the measured unit is 100ms minimum increments. The content of **D + 1** in seconds is multiplied by **n** and stored in **D**.
2. When multiple set-point $n=0$, the measured unit of **D** is in seconds. When $n=1$, the measured unit of **D** is 100ms periods (is multiplied by 10). When $n=2$, the measured unit of **D** is 10ms periods (is multiplied by 100).
3. Operand **D** will occupy 2 continuous devices.
4. The TTMR instruction can only be used eight times in a program.

Program Example 1:

1. The time that the input X0 is pushed (ON duration of X0) will be stored in D1, **n** is used to specify the multiple of the time and the total bit time is stored in D0. Then the button switch can be used to adjust the set-point value of a timer.
2. When X0 = OFF, the content of D1 will be reset to 0 but the content of D0 is unchanged.

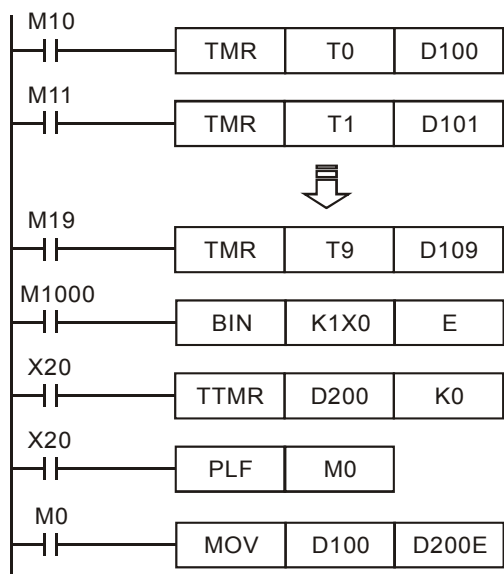


3. If ON duration of X0 is T seconds, the relation between D0, D1 and **n** are shown as the table below.

n	D0	D1(unit: 100 ms)
K0 (unit: s)	$1 \times T$	$D1 = D0 \times 10$
K1 (unit: 100 ms)	$10 \times T$	$D1 = D0$
K2 (unit: 10 ms)	$100 \times T$	$D1 = D0 / 10$

Program Example 2:

1. Using TMR instruction to write 10 groups set-point time.
2. Write the set-point value into D100~D109 in advance.
3. The measured unit of the following timers T0~T9 is 0.1 second and the measured unit of the alternate Timer is 1 second.
4. Connect one bit digital switch to X0~X3 and use BIN instruction to convert the set-point value of digital switch to BIN value and store in E.
5. The ON duration (in sec) of X20 is stored in D200.
6. M0 is a pulse for one scan cycle generated when the alternate timer button X20 is released.
7. Use the set-point number of digital switch as the pointers of index register, and then transmit the content of D100 to D200E (D200~D209).

**Note:**

PC/PA/PH models, can only use the TTMR instruction eight times in a program. If used in a CALL subroutine or interrupt subroutine, it only can be use once.

API	Mnemonic	Operands			Function										Controllers				
65	STMR	<div>S</div>	<div>m</div>	<div>D</div>	Special Timer										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	
OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	STMR: 7 steps		
	S											*							
	m					*	*												
D			*	*	*														

Operands:

S: Number of timer (T0~T191) **m:** Set-point value of timer (**m**=1~32,767), unit in 100ms
D: Starting device of output (occupies 4 continuous devices)

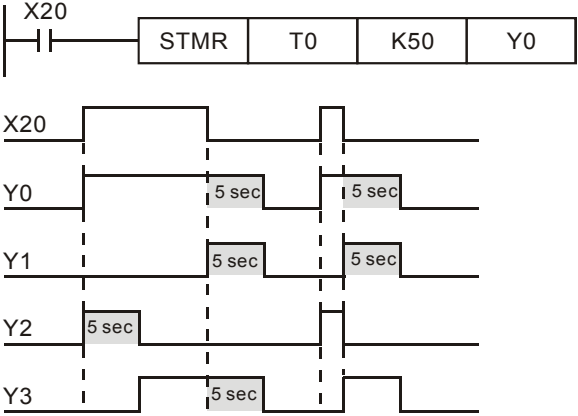
Explanations:

- 1. STMR instruction is a instruction which provides OFF-delay, one shot and flash loop.
- 2. The number of timer specified **S** by STMR instruction can not be repeated.

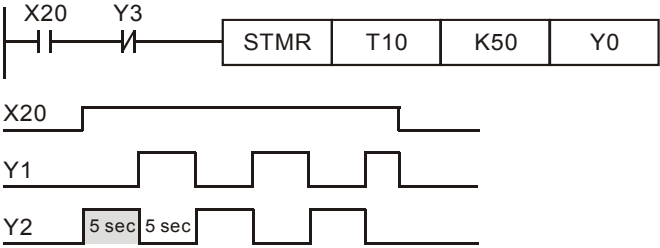
Program Example:

- 1. When X20=ON, the set-point value of the timer T0 is 5 seconds. Y0 is the OFF-delay contact :
- 2. When X20 turns from OFF →ON, Y0= ON. When X20 turns ON →OFF and delay 5 seconds, Y0=OFF. When X20 turns from ON →OFF, Y1= ON output one time for 5 seconds.
- 3. When X20 turns from OFF →ON, Y2=ON output one time for 5 seconds.

- 4. When X20 turns from OFF →ON, Y3=ON after a 5 second delay. When X20 turns from ON →OFF, Y3=OFF after a 5 second delay.



- 5. Add a b contact Y3 after the drive contact X20, and then Y1, Y2 can be used as the output of flash loop. When X20 turns OFF, Y0, Y1 and Y3 = OFF and the content of T10 will be reset to 0



API	Mnemonic			Operands	Function											Controllers			
66		ALT	P	D	Alternate ON/OFF											PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps				
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ALT, ALTP: 3 steps			
	D		*	*	*															

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

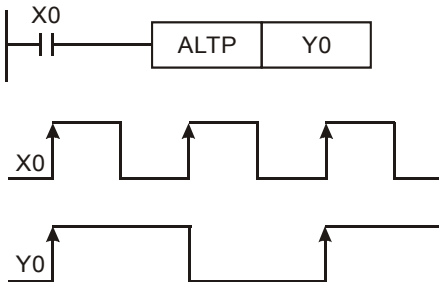
D: Destination device

Explanations:

1. The status of the destination device (**D**) is alternated on every operation of the ALT instruction.
2. This means the status of each bit device will flip-flop between ON and OFF. This will occur on every program scan unless a pulse modifier or a program interlock is used.
3. The ALT instruction is ideal for switching between two modes of operation e.g. stat and stop, ON and OFF etc.
4. This instruction usually works best as a pulse instruction (ALTP).

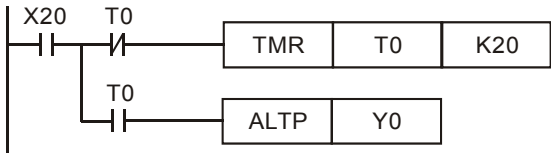
Program Example 1:

When X0 turns from OFF →ON for the first time, Y0=ON. When X0 turns from OFF →ON for the second time, Y0=OFF.



Program Example 2:

Output Y0 will flash. When X20= ON, T0 will generate a pulse every two seconds and output Y0 will be switch between ON and OFF depending on the pulse of T0.



API	Mnemonic	Operands				Function				Controllers			
67	RAMP	<div>S₁</div>	<div>S₂</div>	<div>D</div>	<div>n</div>	Ramp variable Value				<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

OP	Type	Bit Devices				Word devices										Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	RAMP: 9 steps
S ₁														*			
S ₂														*			
D														*			
n						*	*										

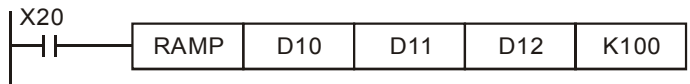
Operands:

S₁: Starting set-point of ramp signal **S₂**: Ending set-point of ramp signal **D**: Current value of ramp signal **n**: Scan times (**n**=1~32,767)

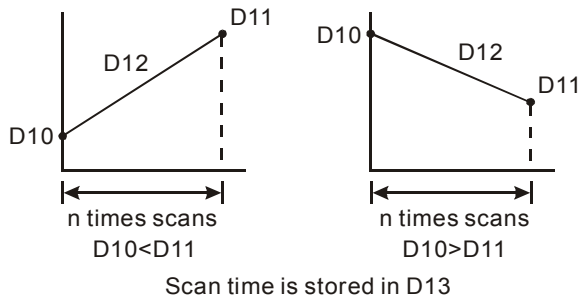
Explanations:

1. This instruction creates a ramp output. A ramp output linearity depends on a consistent scan. Therefore, set the scan time a fix time prior to using this RAMP instruction.
2. Preload the starting set-point value D10 and ending set-point value to D11. When X20 = ON, the set-point value is forwarding from D10 to D11 (set-point value in D10 will be increased) and the proceeding time (**n**= 100 times scans) is stored in D12.
3. To set a fixed scan time, set M1039=ON. Then, preload the scan time value into special register D1039. Take the program below as an example, if the set-point value is 30ms and **n**=K100, the time between D10 and D11 is 3 second (30ms×100).
4. During the execution of this instruction, when starting signal X20 turns OFF, this instruction will stop operation. When X20 turns ON again, the content of D12 will be reset to 0(zero) and calculated again.
5. When M1026=OFF, and the execution of this instruction is completed M1029= ON then the content of D12 will be reset to the set-point value of D10.
6. Using this instruction with analog signal output can execute the operation of smooth Start/Stop.
7. To ensure proper operation when switching the ELC from STOP to RUN when X20= ON, the content of D12 should be reset to 0(zero) in the beginning of the program. (If D12 is a latched area.)

Program example:

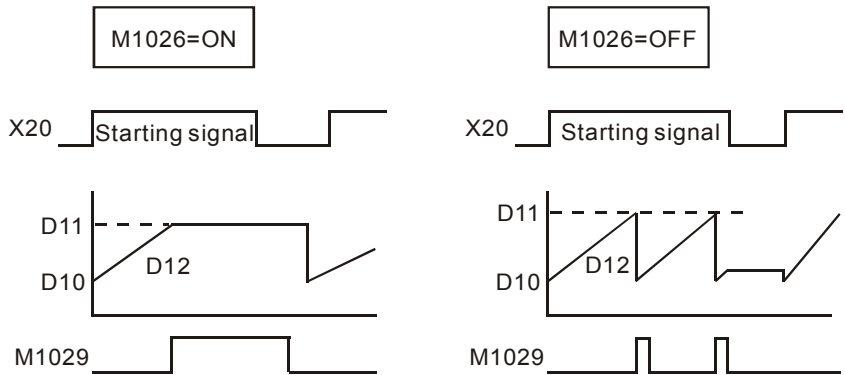


If X20=ON,



Points to note:

M1026 ON/OFF mode:



API	Mnemonic	Operands					Function					Controllers			
69	SORT	S	m₁	m₂	D	n	Data sort					PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F				
	S													*					
	m ₁					*	*												
	m ₂					*	*												
	D													*					
	n					*	*							*					

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: Starting device of source data table **m₁**: Sort data groups (**m₁** = 1~32) **m₂**: Column numbers of each data (**m₂** = 1~6) **D**: Starting device for storing sort data **n**: Reference value of sort data (**n** = 1~**m₂**)

Explanations:

1. The sorted data is stored in the **m₁ × m₂** registers counted from **D**. Therefore, if device **S** and **D** specify the same register, the resulting sorted data will be the same as the content of source device **S**.
2. An ideal number for **S** is 0.
3. Once the SORT instruction has completed, the Flag M1029 (Execution completed flag) = ON.

Program Example:

When X0 = ON, it starts to sort the specified data. After the data sort is complete, M1029 = ON. During the execution of the SORT instruction, data being sorted should not be changed. If the sort data needs to be changed the SORT instruction should be turned OFF, modify the data, and turn ON the SORT instruction.



Example table of data sort

Data numbers: m_2

Column Row	Data Column				
	1	2	3	4	5
	Students No.	English	Math.	Physics	Chemistry
1	(D0) 1	(D5) 90	(D10) 75	(D15) 66	(D20) 79
2	(D1) 2	(D6) 55	(D11) 65	(D16) 54	(D21) 63
3	(D2) 3	(D7) 80	(D12) 98	(D17) 89	(D22) 90
4	(D3) 4	(D8) 70	(D13) 60	(D18) 99	(D23) 50
5	(D4) 5	(D9) 95	(D14) 79	(D19) 75	(D24) 69

Data numbers: m_1

- Sort data table when D100=K3

Data numbers: m_2

Column Row	Data Column				
	1	2	3	4	5
	Students No.	English	Math.	Physics	Chemistry
1	(D50) 4	(D55) 70	(D60) 60	(D65) 99	(D70) 50
2	(D51) 2	(D56) 55	(D61) 65	(D66) 54	(D71) 63
3	(D52) 1	(D57) 90	(D62) 75	(D67) 66	(D72) 79
4	(D53) 5	(D58) 95	(D63) 79	(D68) 75	(D73) 69
5	(D54) 3	(D59) 80	(D64) 98	(D69) 89	(D74) 90

Data numbers: m_1

- Sort data table when D100=K5

Data numbers: m_2

Column Row	Data Column				
	1	2	3	4	5
	Students No.	English	Math.	Physics	Chemistry
1	(D50) 4	(D55) 70	(D60) 60	(D65) 99	(D70) 50
2	(D51) 2	(D56) 55	(D61) 65	(D66) 54	(D71) 63
3	(D52) 5	(D57) 95	(D62) 79	(D67) 75	(D72) 69
4	(D53) 1	(D58) 90	(D63) 75	(D68) 66	(D73) 79
5	(D54) 3	(D59) 80	(D64) 98	(D69) 89	(D74) 90

Data numbers: m_1

API	Mnemonic			Operands			Function								Controllers												
	70	D	TKY		S	D₁	D₂	Ten Key Input								PB	PC	PA	PH								
OP	Type	Bit Devices				Word devices										Program Steps											
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	TKY: 7 steps DTKY: 13 steps										
	S	*	*	*	*																						
	D ₁								*	*	*	*	*	*	*	*											
	D ₂		*	*	*																						
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

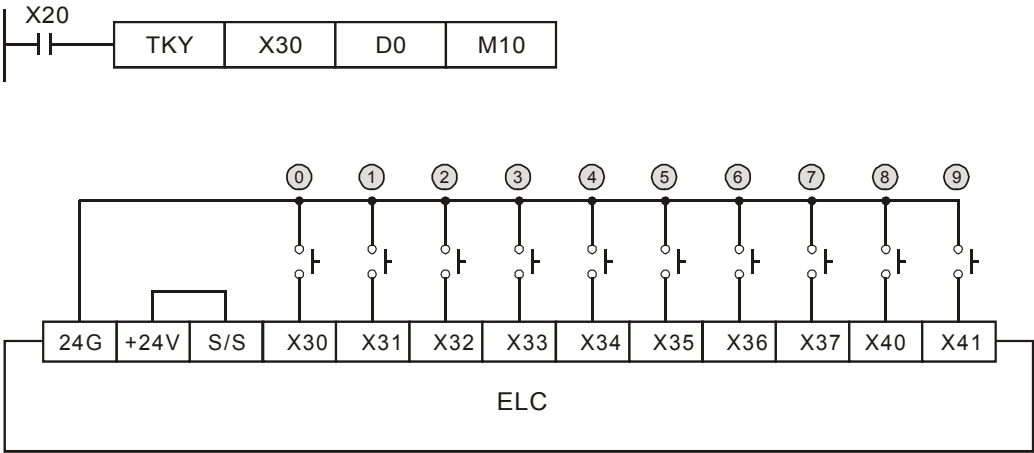
S: Start input device (occupies 10 continuous devices) **D₁**: Destination for storing key input value
D₂: Key input signal (occupies 10 continuous devices)

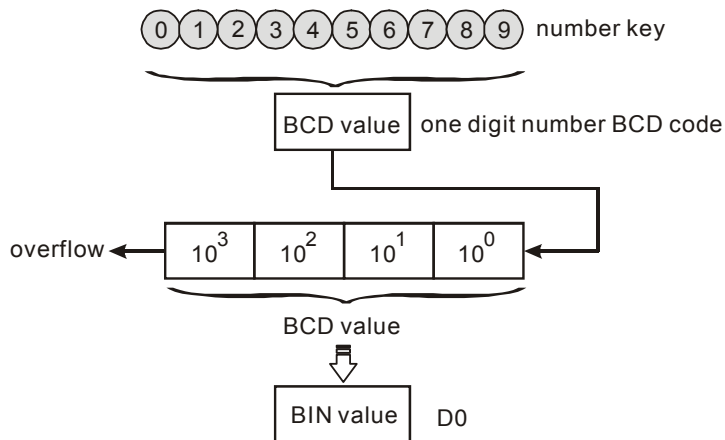
Explanations:

This instruction can specify ten external input devices from **S** and these ten external input devices is identified as decimal value of 0 to 9. These ten external input devices are connected to ten keys respectively. When keys is pressed, the value of decimal numbers from 0 to 9,999 (max. 4 digits in 16-bit instruction) or from 0 to 99,999,999 (max. 8 digits in 32-bit instruction) can be input and stored in destination **D₁**. The device **D₂** is used to store the state of that key has been pressed.

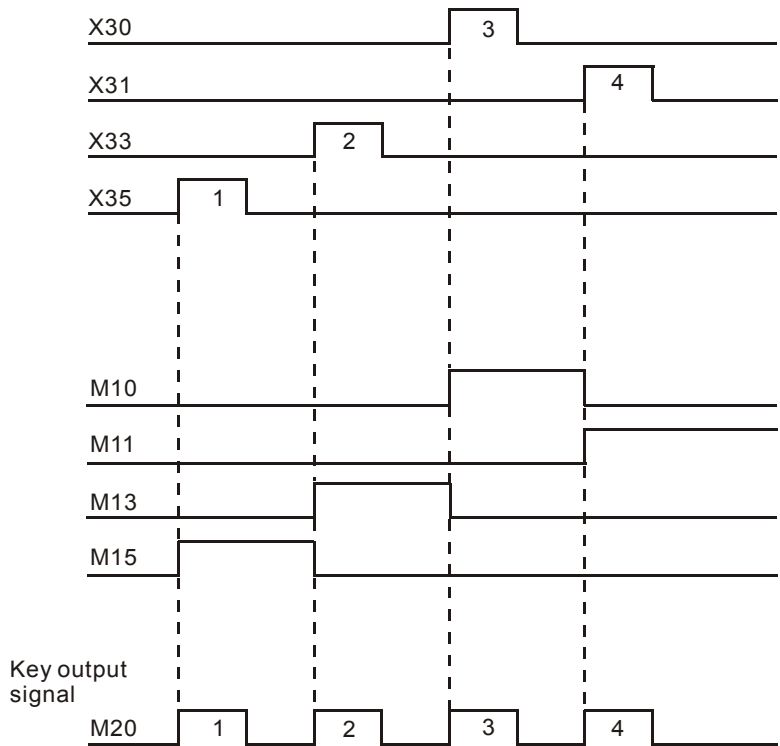
Program Example:

Using this instruction can specify ten input terminals from X30 to connect to ten keys which number is from 0 to 9. When X20=ON, the instruction is executed and it will store the BIN value which is input by keys into D0 and M10~M19 is used to store the condition of the key that has been pressed.





1. The chart below has four keys connected to X35, X33, X30 and X31 of a number keyboard. After pressing the four keys in that order of ①②③④ and the number 5,301 will be entered into D0. The max. number which can be entered in D0 is 9,999 i.e. 4 digits. If the entered number exceeds the allowable range, the highest digits will overflow.
2. After X32 is pressed, M12=ON until another key is pressed. The process is the same as other keys are pressed.
3. When any key of X30~X41 is pressed, one device of M10~M19 will be ON.
4. If any key is pressed, M20=ON.



API	Mnemonic		Operands				Function								Controllers			
71	D	HKY	(S)	(D₁)	(D₂)	(D₃)	Hexadecimal Key Input								PB	PC	PA	PH

Type	Bit Devices				Word devices										Program Steps			
OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	HKY: 9 steps DHKY: 17 steps		
S	*																	
D ₁		*																
D ₂											*	*	*	*	*			
D ₃		*	*	*														

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

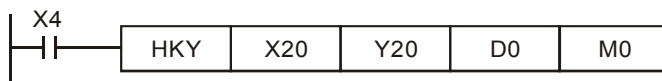
S: Start scan input device (occupies 4 continuous devices) **D₁:** Start scan output device (occupies 4 continuous devices) **D₂:** Destination for storing key input value **D₃:** Key input signal (occupies 8 continuous device)

Explanations:

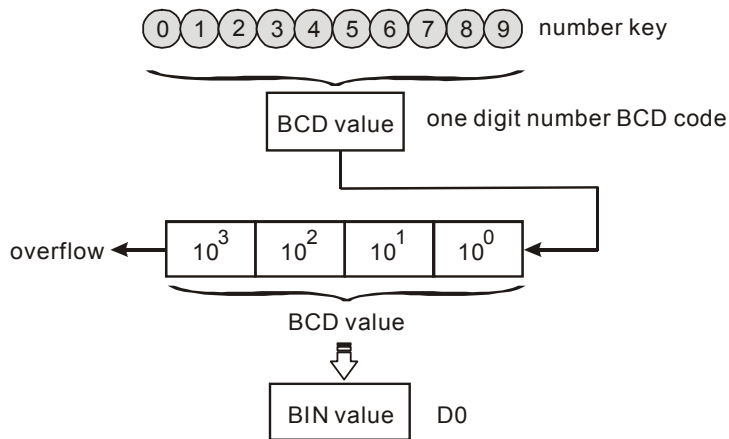
1. This instruction can create a 16-key keyboard of 4 continuous external input devices from **S** and 4 continuous external output devices from **D₁** by matrix scan. The key input value will be stored in **D₂** and **D₃** and is used to store the condition that a key has been pressed.
2. Every time this instruction is executed, the execution completed flag M1029 = ON for the duration the key is pressed (one scan cycle).
3. If two or more keys are pressed at the same time, only the key activated first will be effective.
4. When HKY instruction is used in a 16-bit instruction, **D₂** can store numbers from 0 to 9,999 (max. 4 digits). When DHKY instruction is used in a 32-bit instruction, **D₂** can store numbers from 0 to 99,999,999 (max. 8 digits). If the entered number exceeds the above allowable range, the highest digits will overflow.

Program Example:

1. Using this instruction to create a 16-key keyboard which is a multiplex of 4 continuous external input devices X20~X23 and 4 continuous external output devices Y20~Y23. When X4=ON, the instruction is executed and it will store the BIN value which is inputted by keys into D0 and M0~M7 is used to store the condition of that a key has been pressed.

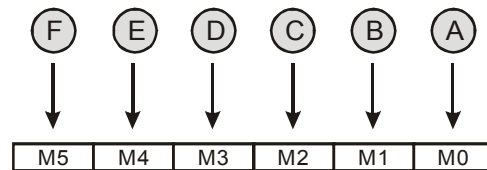


2. Number input:



3. Function key input:

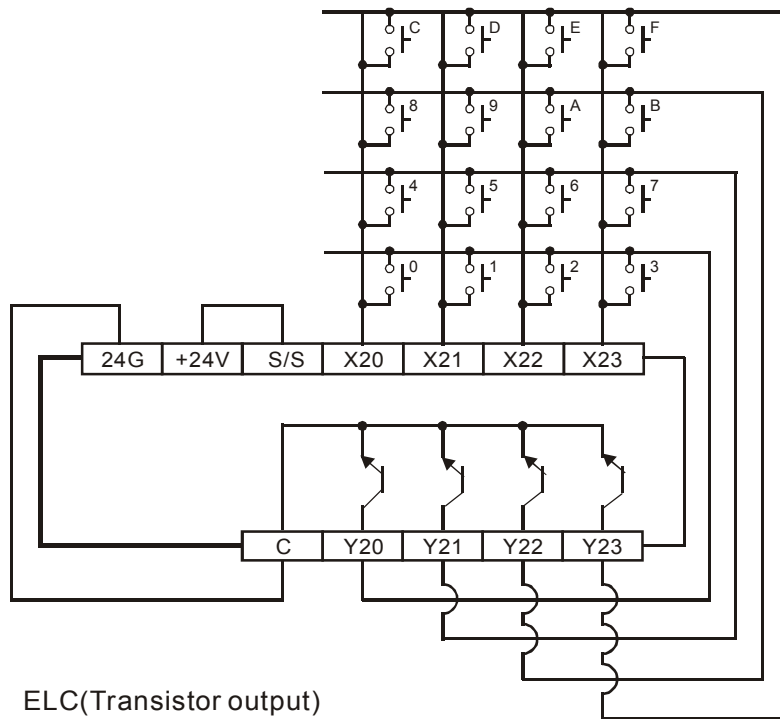
- a) When A key is pressed, M0=ON and latched. Next, press D key and then, M0=OFF, M3=ON and latch.
- b) If two or more keys are pressed at the same time, only the key activated first is effective.



4. Key output signal:

- a) When any key of A – F is pressed, M6=ON for one scan time.
 - b) When any key of 0 to 9 is pressed, M7=ON for one scan time.
5. When the drive contact X4 = OFF, the previous values do not change but M0~M7 = OFF.

6. External wiring:

**Points to note:**

1. When this instruction is executed, 8 scan time cycles are required to read the input value of keys successfully. If the scan cycle is too long or too short, it may cause the key values to be read incorrectly. Therefore, a fixed scan time is suggested.
 - a) When scan time is too short, it can not read correct I/O key input, therefore, you can fixed the scan time.
 - b) When scan time is too long, use this instruction in subroutine of time interrupt to executed.
2. The function of flag M1167:
 - a) When M1167=ON, HKY instruction can input hexadecimal value of 0~F.
 - b) When M1167=OFF, A~F of HKY instruction are used as function keys.

API	Mnemonic	Operands	Function	Controllers			
72	DSW	(S) (D₁) (D₂) (n)	Digital Switch	PB	PC	PA	PH

Type	Bit Devices				Word devices										Program Steps	
OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DSW: 9 steps
S	*															
D ₁		*														
D ₂											*	*	*			
n					*	*										

Operands:

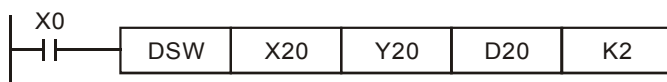
S: Starting device of switch input **D₁:** Starting device of switch output **D₂:** Destination device for storing the set-point value **n:** Number of digits ($n=1\sim 2$)

Explanations:

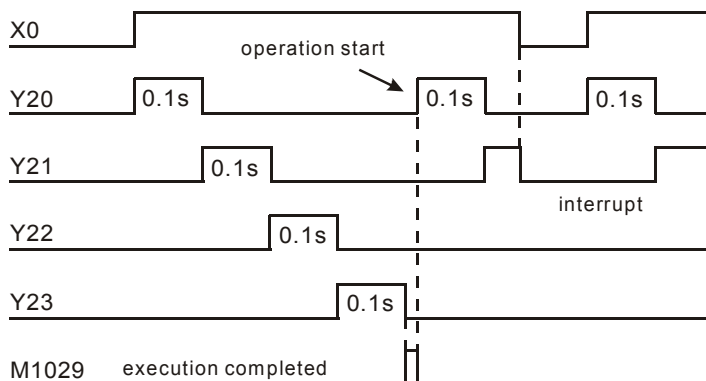
- This instruction is used to read one($n=1$) or two($n=2$) groups of 4 digits switch through 4 or 8 continuous external input devices from **S** and 4 continuous external devices from **D₁** and store the set-point value in destination device **D₂**. Flag: Operation complete M1029

Program Example:

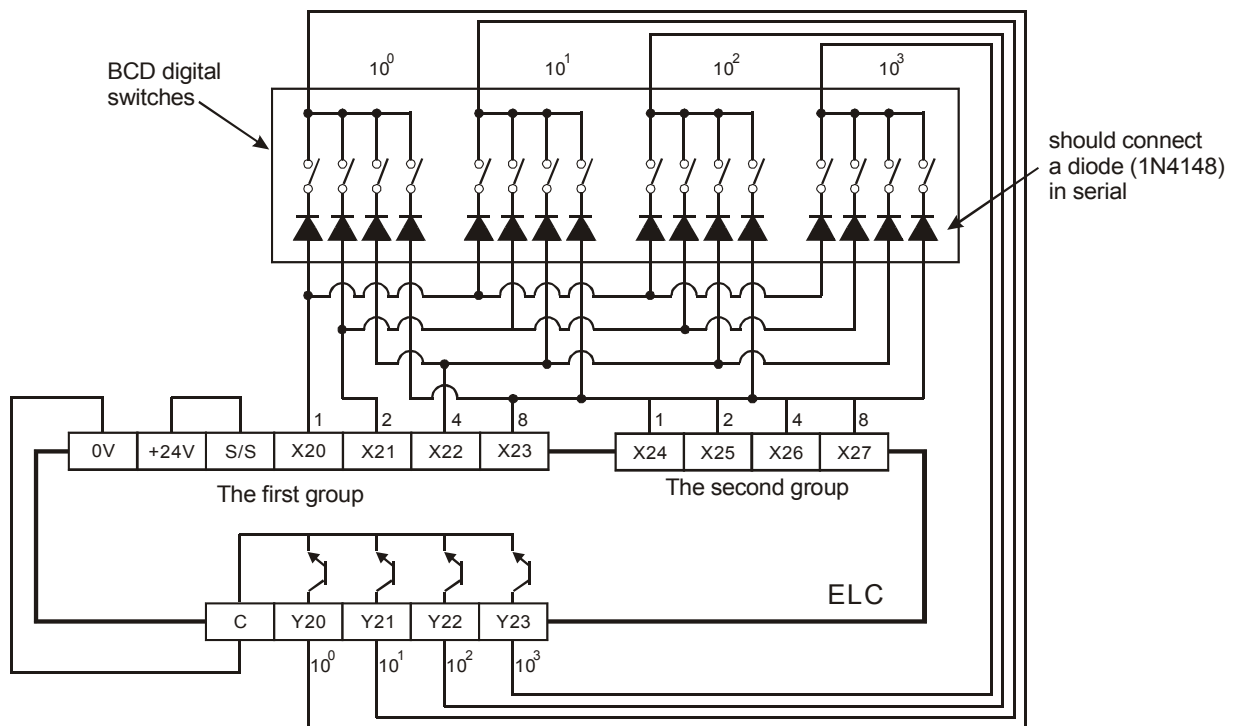
- The first group of switches consists of X20~X23 and Y20~Y23. The second group of switches consists of X24~X27 and Y20~Y23. When X0=ON, the set-point value of the first group of switches are read and converted to binary and stored in D20. The set-point value of the second group of switches are read and converted to binary and stored in D21.



- When X0=ON, Y20~Y23 will be ON and scan in circles automatically. After the completion of each circle scan, execution completed flag M1029=ON is a scan period after a circle scan.
- Outputs Y20~Y23 please use transistor output. Besides, please make sure that every 1, 2, 4, 8 terminal should connect a diode (0.1A/50V) to the inputs of ELC in serial as shown right.



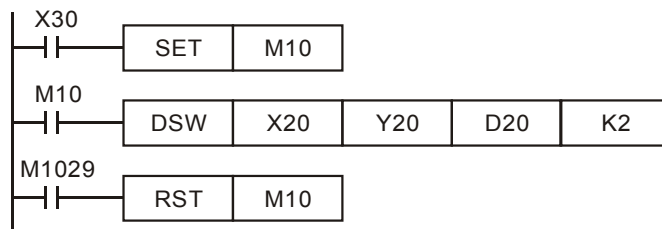
Wiring diagram of digital switch:



Points to note:

When the scan terminals are relay outputs, the following program technique is used with this instruction to operate successfully:

1. When X30=ON, DSW instruction is executed. When X30 turns OFF, M10 will be ON until the scan terminals of DSW instruction complete one output scan cycle. Then, M10 will turn OFF.
2. If the drive contact X30 use button switch, every time when X30 is pushed, M10, the scan terminals specified by DSW instruction, will be reset to OFF after the completion of one output scan cycle. Then, the instruction will stop executing, the data of digital switch will be read completely and the scan terminals will be activated while the button switch is pushed. Therefore, even relay output is used in this situation, the relay can be used for long because the operation of relay is not frequent.



API	Mnemonic			Operands		Function										Controllers			
73		SEGD	P	S	D	7-segment Decoder										PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SEGD, SEGDP: 5 steps		
S					*	*	*	*	*	*	*	*	*	*	*			
D								*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

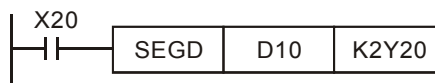
S: Source device for decoding **D:** Output device after decoding

Explanations:

A single hexadecimal digit (0 to 9, A to F) occupying the lower 4 bits of source device **S** is decoded into a data format used to drive a seven segment display. A representation of the hex digit is then displayed. The decoded data is stored in the lower 8 bits of destination device **D**. The upper 8 bits of the same device are not written to.

Program Example:

When X20=ON, contents (16 bits) of the lower 4 bits (b0~b3) of D10 will be decoded as readable in the 7-segment display panel for output. The decoding results will be stored in Y20~Y27.

**Decoding Chart of the 7-segment Display Panel**

16 bits	Bit Combination	Composition of the 7-SEG display	Status of each segment						
			B0(a)	B1(b)	B2(c)	B3(d)	B4(e)	B5(f)	B6(g)
0	0000		On	On	On	On	On	On	Off
1	0001		Off	On	On	Off	Off	Off	Off
2	0010		On	On	Off	On	On	Off	On
3	0011		On	On	On	On	Off	Off	On
4	0100		Off	On	On	Off	Off	On	On
5	0101		On	Off	On	On	Off	On	On
6	0110		On	Off	On	On	On	On	On
7	0111		On	On	On	Off	Off	Off	Off
8	1000		On	On	On	On	On	On	On
9	1001		On	On	On	On	Off	On	On
A	1010		On	On	On	Off	On	On	On
B	1011		Off	Off	On	On	On	On	On
C	1100		On	Off	Off	On	On	On	Off
D	1101		Off	On	On	On	On	Off	On
E	1110		On	Off	Off	On	On	On	On
F	1111		On	Off	Off	Off	On	On	On

API	Mnemonic	Operands			Function										Controllers				
74	SEGL	S	D	n	7-segment with Latch										PB	PC	PA	PH	
OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SEGL: 7 steps		
	S					*	*	*	*	*	*	*	*	*	*	*			
	D		*																
	n					*	*												

Operands:

S: Display source device of 7-segment display **D:** Start device of 7-segment display scan output

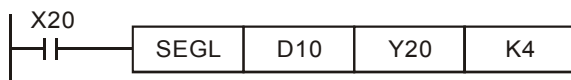
n: Polarity set-point of output signal and scan signal (**n**=0~7)

Explanations:

- 8 or 12 continuous external output points that start from this instruction operand **D** can be regarded as display and scan signal output of 1 or 2 groups of 4 digits of 7-segment display. 7-segment display module has function to convert input BCD code to 7-segment display and has control signal to latch it.
- n** will decide the numbers of groups of 4 digits of 7-segment display and also indicate the polarity of ELC output terminal and 7-segment display input terminal.
- The points number of 7-segment display output instruction that a group of 4 digits use is 8 points and 2 groups of 4 digits use are 12 points.
- Scan output terminal will circulate in sequence when this instruction executes. The drive contact will be changed from OFF to ON and scan output execute again.
- SEGL instruction can only be used twice.
- Flag: M1029 execution completed flag

Program Example:

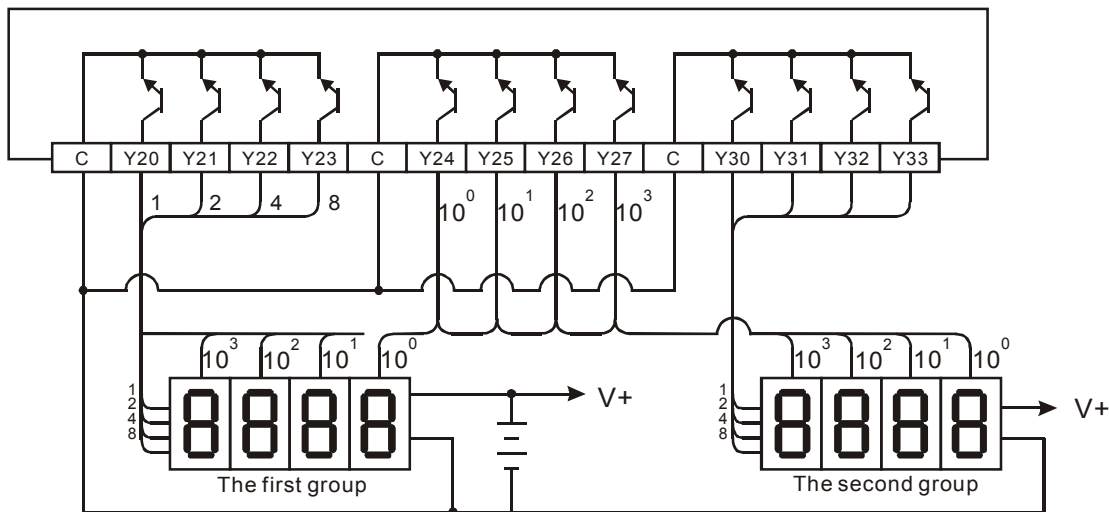
- When X20=ON, instruction will start to execute. 7-segment display scan loop is composed of Y20~Y27. The value of D10 will be converted to BCD code and send to the first group of 7-segment display to display. The value of D11 will be converted to BCD code and send to the second group of 7-segment display to display. If any value of D10 or D11 is greater than 9999, operation error will occur.



- When X20=ON, Y24~Y27 will scan in circles automatically. Each circle scan needs 12 scan cycles. M1029=ON is a scan period after a circle scan.

3. 4 digits of a group, $n=0\sim3$
 - a) After the terminal of 1, 2, 4, 8 of decoded 7-segment display connects itself in parallel, they should connect to Y20~Y23 of ELC. Latch terminal of each number connects to Y24~Y27 of ELC individually.
 - b) When X20=ON, the content of D10 will be transmitted to 7-segment display to display in sequentially according to Y24~Y27 circulates in sequence
4. 4 digits of **two** groups, $n=4\sim7$
 - a) After the terminal of 1, 2, 4, 8 of decoded 7-segment display connects itself in parallel, they should connect to Y30~Y33 of ELC. Latch terminal of each number and the first group share Y24~Y27 of ELC.
 - b) The content of D10 will be transmitted to the first group of 7-segment display and the content of D11 will be transmitted to the second group of 7-segment display to display. If D10=K1234 and D11=K4321, the first group will display 1 2 3 4 and the second group will display 4 3 2 1.

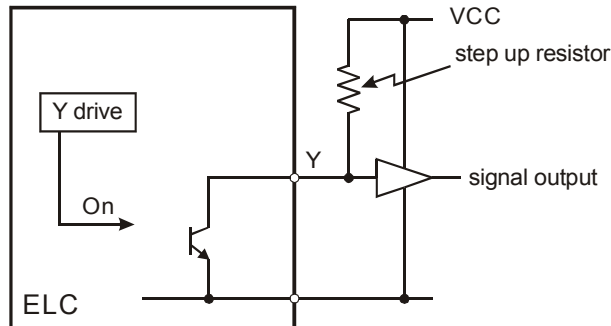
7-segment display scan output wiring



Points to note:

1. PB series only provide a group of 4 digits of 7-segment display and use 8 points to output. SEGL instruction only can be used once in the program and the usage range of operand n is 0 to 3.
2. Scan time must be longer than 10ms while this instruction is executed. If scan time is shorter than 10ms, please use fixed scan time function to fix scan time on 10ms.
3. Please use suitable 7-segment display for the transistor that ELC uses to output.
4. Set-points of n : it is used to set the polarity of transistor output loop. It can be set to positive polarity or negative polarity. What 7-segment display it connects is a group of 4 digits or two groups of 4 digits.

5. ELC transistor output is NPN type and it is open collect output. When wiring, output should connect a pull-high resistor to VCC (less than 30VDC). Therefore, when output point Y is ON, output will be low voltage.



6. Positive logic (Negative polarity) output of BCD code

BCD value				Y output (BCDcode)				Signal output			
b ₃	b ₂	b ₁	b ₀	8	4	2	1	A	B	C	D
0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	1	1	1	1	0
0	0	1	0	0	0	1	0	1	1	0	1
0	0	1	1	0	0	1	1	1	1	0	0
0	1	0	0	0	1	0	0	1	0	1	1
0	1	0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0	1	0	0	1
0	1	1	1	0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	1	0	1	1	0

7. Negative logic (Positive polarity) output of BCD code

BCD value				Y output (BCDcode)				Signal output			
b ₃	b ₂	b ₁	b ₀	8	4	2	1	A	B	C	D
0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	0	0	1	0
0	0	1	1	1	1	0	0	0	0	1	1
0	1	0	0	1	0	1	1	0	1	0	0
0	1	0	1	1	0	1	0	0	1	0	1
0	1	1	0	1	0	0	1	0	1	1	0
0	1	1	1	1	0	0	0	0	1	1	1
1	0	0	0	0	1	1	1	1	0	0	0
1	0	0	1	0	1	1	0	1	0	0	1

8. Display scan (latch) signal

Positive logic (Negative polarity) output		Negative logic (Positive polarity) output	
Y output (Latch)	Output control signal	Y output (Latch)	Output control signal
1	0	0	1

9. Parameter **n** set-points:

Groups number of 7-segment display	A group				Two groups			
Y of BCD code outputs	+		—		+		—	
Display scan latch signal	+	—	+	—	+	—	+	—
n	0	1	2	3	4	5	6	7

'+' : Positive logic (Negative polarity) output

'—' : Negative logic (Positive polarity) output

10. The matching of output polarity of ELC transistor and input polarity of 7-segment display can be set by set-points of **n**.

API	Mnemonic	Operands				Function				Controllers			
75	ARWS	S	D₁	D₂	n	Arrow Key Input				PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ARWS: 9 steps
S		*	*	*	*												
D ₁												*	*	*	*	*	
D ₂			*														
n						*	*										

Operands:

S: Start device of key input (occupies 4 continuous devices) **D₁:** Display device on 7-segment display
D₂: Scan output start device of 7-segment display **n:** Polarity set-point of output signal and scan signal (**n**=0~3)

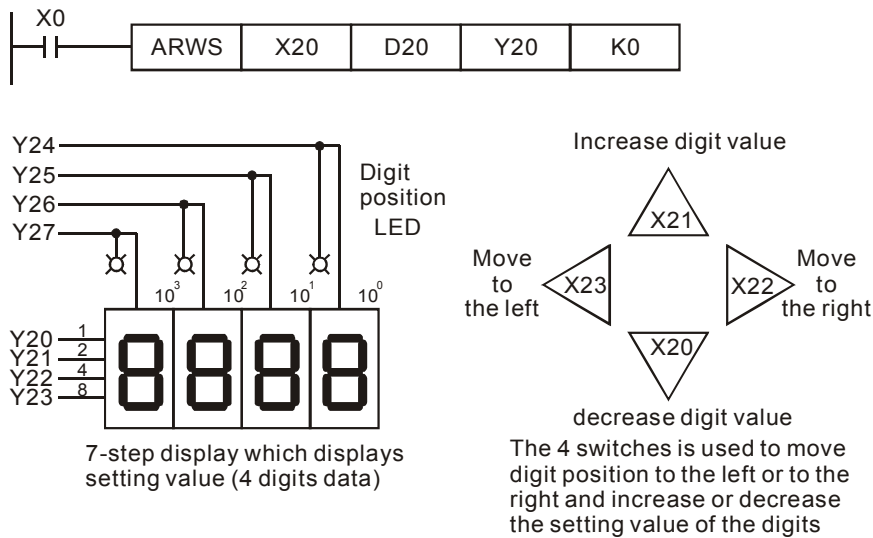
Explanations:

1. This instruction displays the contents of a single data device **D₁** on a set of 4 digit, seven segment displays. The data within **D₁** is actually in a standard decimal format but is automatically converted to BCD for display on the seven segment units. Each digit of the displayed number can be selected and edited. The editing procedure directly changes the value of the device specified as **D₁**.
2. There is no number-of-times limit when using ARWS, but this instruction only can be executed once in the program.
3. it only can be specified as a multiple of 10, e.g. Y0, Y20...etc.
4. Output point that designated by this instruction should use transistor to output.
5. When using this instruction to set the scan time to fixed or put this instruction into time interrupt subroutine (I6□□~I7□□) to execute.

Program Example:

1. When the instruction is executed, X20 is defined as the down key, X21 is defined as the up key, X22 is defined as the right key and X23 is defined as the left key. These keys are used to edit and display the external set-point value. The set-point value is stored in D20 and its set-point range is from 0 to 9,999.
2. When X0=ON, 10^3 is a effective set-point digit number. pressing left key, the effective set-point digit number will be displayed and jump by the direction from $10^3 \rightarrow 10^0 \rightarrow 10^1 \rightarrow 10^2 \rightarrow 10^3 \rightarrow 10^0$.
3. pressing the right key, the effective set-point digit number will be displayed and jump by the direction from $10^3 \rightarrow 10^2 \rightarrow 10^1 \rightarrow 10^0 \rightarrow 10^3 \rightarrow 10^2$. Meanwhile, the digit position LED connected from Y24 to Y27 will also be ON to indicate the effective set-point digit number.

4. pressing the up key to increase , the effective number will change from 0→1→2→...8→9→0→1. If pressing the down key, the effective number will change from 0→9→8→...1→0→9, meanwhile, the changed value will be displayed on the 7-segment display.



API	Mnemonic	Operands	Function	Controllers			
76	ASC	S D	ASCII code Conversion	PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ASC: 11 steps
	S																
D												*	*	*			

Operands:

S: The alphanumeric character to be converted to its ASCII code **D**: The destination for storing the ASCII code

Explanation:

1. The alphanumeric character can be used to display error message directly if connect 7-segment display when using this instruction.
2. Operand **S** is an 8 alphanumeric character string input by ELCSoft software from PC, or ASCII code input by HHP units.
3. Flag: M1161 8/16 mode exchange.

Program Example:

When X0=ON, A~H is converted to ASCII code and stored in D0~D3.



	b15	b0
D0	42H (B)	41H (A)
D1	44H (D)	43H (C)
D2	46H (F)	45H (E)
D3	48H (H)	47H (G)
	High byte	Low byte

When M1161=ON, the ASCII code converted from every character will occupy the lower 8-bit (b7~b0) of one register. The high byte will be invalid and its content is filled with 0. This also means that one register only can be used to store one character.

	b15	b0
D0	00 H	41H (A)
D1	00 H	42H (B)
D2	00 H	43H (C)
D3	00 H	44H (D)
D4	00 H	45H (E)
D5	00 H	46H (F)
D6	00 H	47H (G)
D7	00 H	48H (H)
	High byte	Low byte

API	Mnemonic	Operands		Function										Controllers			
77	PR	<div>S</div>	<div>D</div>	Print										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	PR: 5 steps			
	S											*	*	*					
D		*																	

Operands:

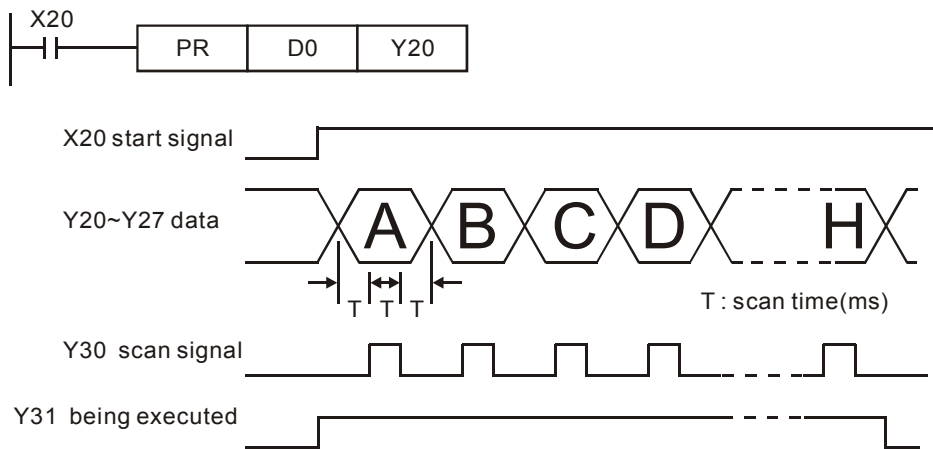
S: The device for storing ASCII code (occupies 4 continuous devices) **D:** The external output device which outputs ASCII code (occupies 10 continuous devices)

Explanations:

1. This instruction will output ASCII codes stored in 4 registers from **S** in order of the output devices specified by **D**.
2. The PR instruction only can be used twice in the program.
3. Flag: M1029 execution completed flag.

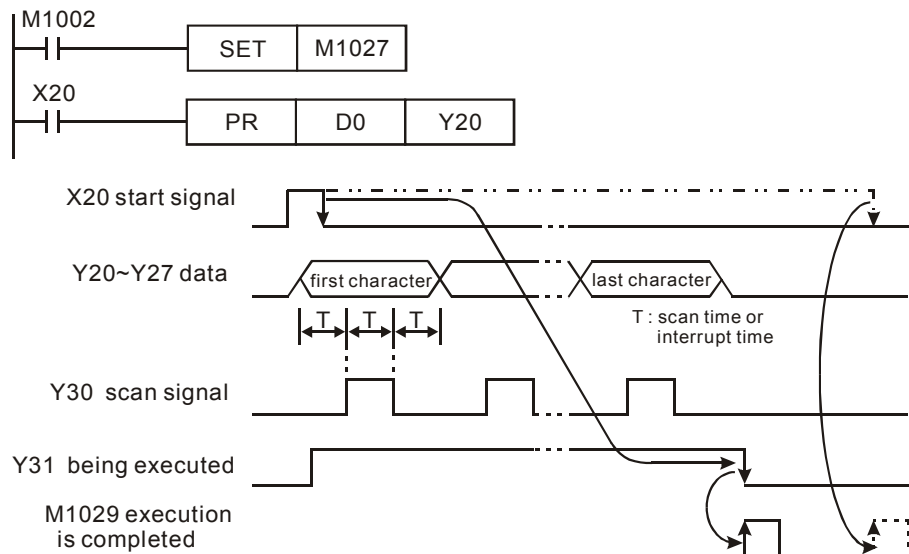
Program Example 1:

1. First, use the ASC instruction convert A~H to ASCII code and store them in D0~D3. Then, using this instruction output them in the order of A~H.
2. When M1027=OFF, X20 turns from OFF →ON, the instruction is executed, Y20 (low byte) to Y27(high byte) is specified as the data output devices, Y30 is specified as scan signal and Y31 is specified as the monitor signal while the instruction being executed. This mode can execute an 8 character string output operation.
3. If X20 turns from OFF →ON while the instruction being executed, the data output will be interrupt. When X20 = ON once more, the data will be sent again.



Program Example 2:

1. The PR instruction provides an 8 character serial string output operation. When M1027=OFF, a maximum of 8 character strings can be output serially. When M1027=ON, a 1 to 16 character string output operation can be executed.
2. When M1027=ON, X20 turns from OFF → ON, Y20(low byte) to Y27(high byte) is specified as the data output devices, Y30 is specified as a scan signal and Y31 is specified as the monitor signal while the instruction being executed. This mode can execute a 16-character string output operation.
3. If the character string 00H (NULL) has been sent, it means the end of the character string and the operation of PR instruction won't continue.
4. The drive contact X20 is always ON but it will automatically stop after one time operation of data output. However, if X20 is always ON, M1029 won't be activated.

**Points to note:**

1. This instruction should only use transistor output.
2. When using this instruction, set the scan time to fixed or execute this instruction in a time interrupt subroutine.

API	Mnemonic			Operands				Function				Controllers			
78	D	FROM	P	m₁	m₂	D	n	Read CR from Module				PB	PC	PA	PH

OP	Type	Bit Devices				Word devices											Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F		
m ₁						*	*										FROM, FROMP: 9 steps	
m ₂						*	*										DFROM, DFROMP: 17	
D									*	*	*	*	*	*	*	*	steps	
n						*	*											

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

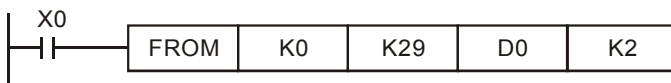
m₁: Number for special module (**m₁**=0~7) **m₂**: Number of CR (Control Register) of special module (**m₂**=0~48) that will be read **D**: Location to save read data **n**: Data words to read at one time (**n**=1~(49- **m₂**))

Explanations:

1. ELC uses this instruction to read CR data of special modules.
2. When **D** indicates a bit device operand, use K1~K4 for 16-bit instruction and K5~K8 for 32-bit instruction.
3. Flag: When M1083=ON, it enables interrupts during instruction FROM/TO. Refer to following explanation for detail.

Program Example:

1. Read the content of CR#29 of special module#0 to D0 of ELC and to read the content of CR#30 of special module#0 to D1 of ELC. The 2 words can be read at one time (**n**=2).
2. The instruction will be executed when X0=ON. The instruction won't be executed when X0=OFF and the content of previous reading data won't change.



API	Mnemonic			Operands				Function	Controllers			
79	D	TO	P	(m ₁)	(m ₂)	(S)	(n)	Write CR to Module	PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
m ₁					*	*										TO, TOP: 9 steps DTO, DTOP: 17 steps
m ₂					*	*										
S					*	*	*	*	*	*	*	*	*	*	*	
n					*	*										

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

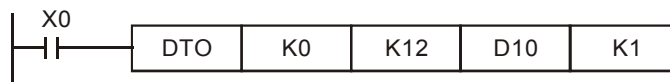
m₁: Number of special module (m₁=0~7) **m₂**: Number of CR (Control Register) of special module that will be written to (m₂=0~48) **S**: Data to write in CR **n**: number of words to write one time (n =1~(49-m₂))

Explanations:

- When **S** is a bit operand, K1~K4 can be used for 16-bit instruction and K1~K8 can be used for 32-bit instruction.
- Flag: M1083 FROM/TO mode exchange.

Program Example:

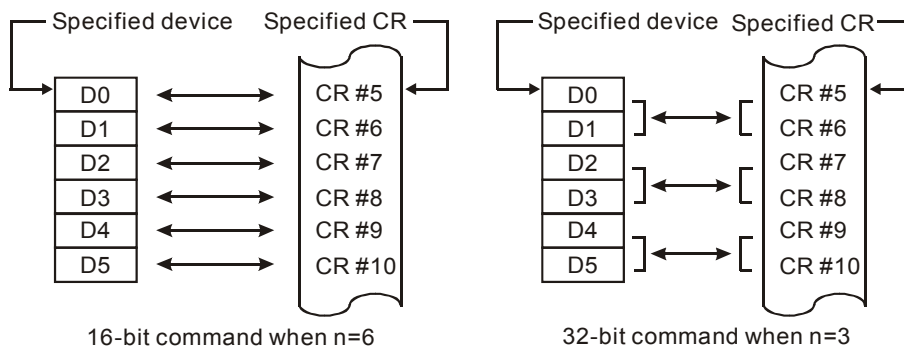
- Using the 32-bit instruction DTO, program will write D11 and D10 into CR#13 and CR#12 of special module#0. It writes a group of data at one time (n=1)
- The instruction will be executed when X0=ON and won't be executed when X0=OFF. The data that was written previously will stay unchanged.

**The rule of instruction operand:**

- m₁**: number of special module. The special module closest to the ELC controller will be assigned 0, the next closest will be assigned 1, etc. for a total of 8 modules(0~7).
- m₂**: number of CR. Built-in of 48 groups of 16-bit memory of special module are called CR (Control Register). Each CR uses decimal digits (#0~#48).

Upper 16-bit Lower 16-bit

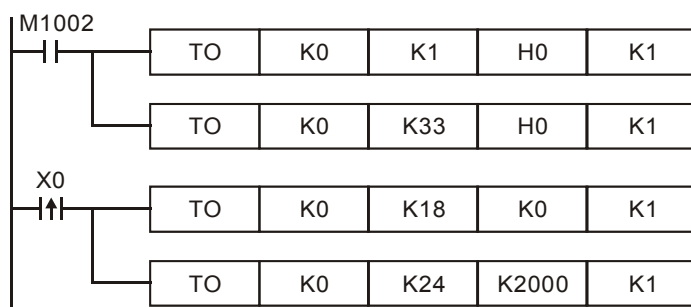




3. In PB models, flag M1083 is not provided. When the FROM/TO instruction is executed, all interrupts (including external or internal interrupt subroutines) will be disabled. All interrupts will be executed after the FROM/TO instruction is completed. The, FROM/TO instruction can also be executed in an interrupt subroutine.
4. The function of the flag M1083 (FROM/TO mode) provided in PC/PA/PH models:
 - a) When M1083=OFF, FROM/TO instruction is executed, all interrupts (including external or internal interrupt subroutines) will be disabled. All interrupts will be executed after FROM/TO instruction is completed.
 - b) When M1083=ON, if an interrupt occurs while the FROM/TO instruction has been programmed, The FROM/TO instruction will be interrupted to execute the interrupt. The FROM/TO instruction cannot be executed in the interrupt subroutine

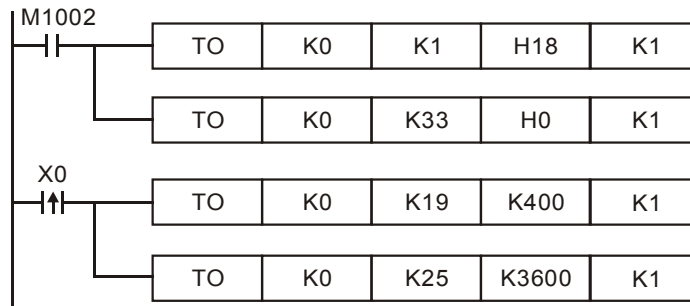
Application program example of FROM/TO instruction:

1. Example 1: Adjust A/D conversion characteristic curve of ELC-AN04ANNN by a set-point OFFSET value of CH1 to 0V(=K0_{LSB}) and GAIN value of CH1 to 2.5V(=K2000_{LSB}).

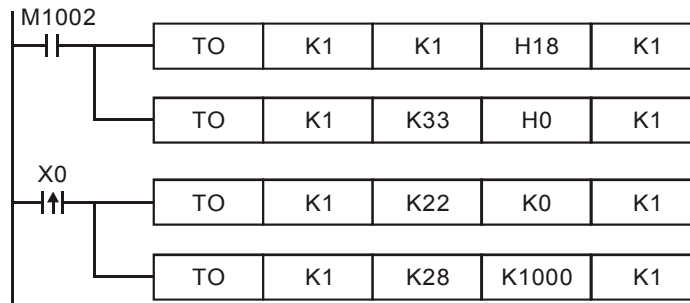


- a) Write H0 to CR#1 of the analog input module No. 0 and set CH1 to mode 0 (voltage input : -10V to +10V).
- b) Write H0 to CR#33 and allow it to adjust the characteristics of CH1 to CH4.
- c) When X0 turns from OFF →ON, K0_{LSB} OFFSET value will be written to CR#18 and K2000_{LSB} GAIN value will be written to CR#24.

2. Example 2: Adjust A/D conversion characteristic curve of ELC-AN04ANNN by the set-point OFFSET value of CH2 to 2mA(=K400_{LSB}) and GAIN value of CH2 to 18 mA(=K3600_{LSB}).

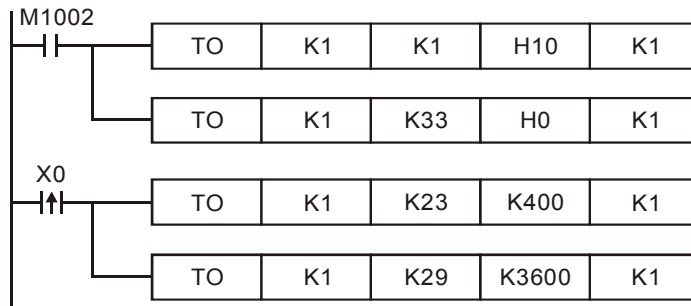


- Write H18 to CR#1 of analog input module No. 0 and set CH2 to mode 3 (current input : -20mA to +20mA).
 - Write H0 to CR#33 to adjust characteristics of CH1 to CH4.
 - When X0 turns from OFF → ON, K400_{LSB} of OFFSET value will be written to CR#19 and K3600_{LSB} of GAIN value will be written to CR#25.
3. Example 3: Adjust D/A conversion characteristic curve of ELC-AN02NANN by set-point OFFSET value of CH2 to 0mA(=K0_{LSB}) and GAIN value of CH2 to 10mA(=K1000_{LSB}).



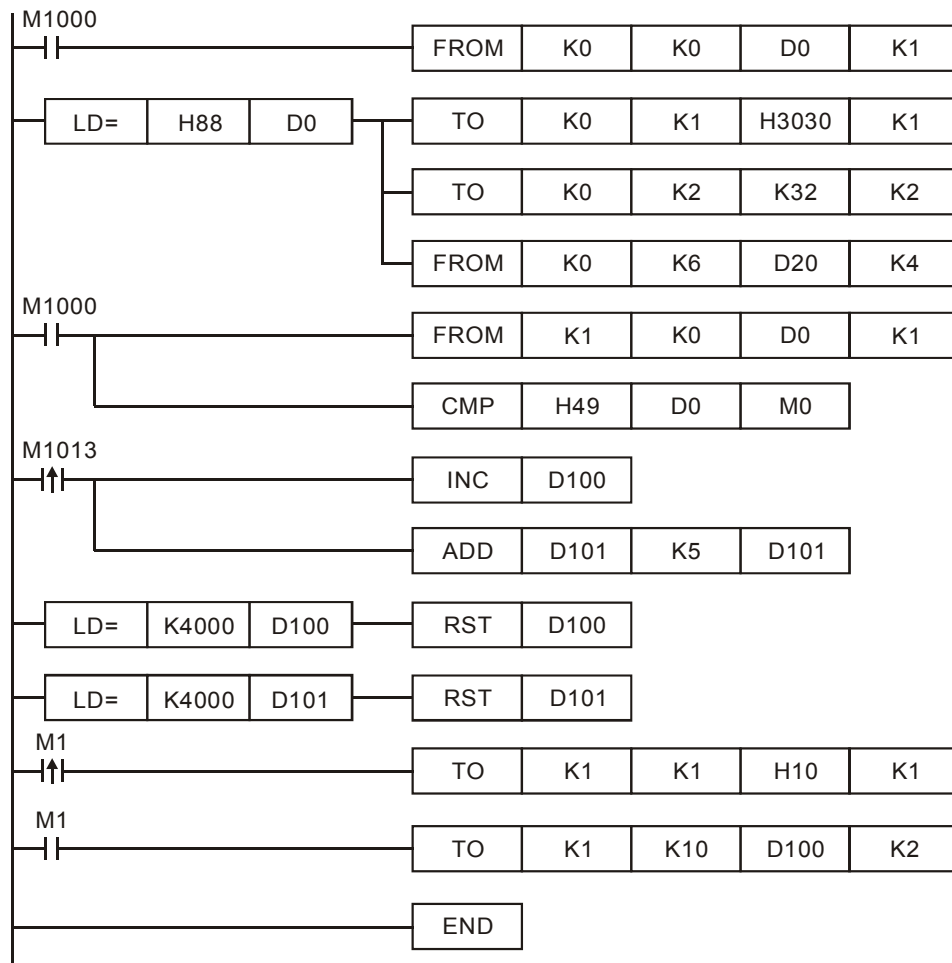
- Write H18 to CR#1 of analog input mode No. 1 and set CH2 to mode 3 (current input : 0mA to +20mA).
- Write H0 to CR#33 to adjust characteristics of CH1 and CH2.
- When X0 turns from OFF → ON, K0_{LSB} of OFFSET value will be written to CR#22 and K1000_{LSB} of GAIN value will be written to CR#28.

4. Example 4: Adjust D/A conversion characteristic curve of ELC-AN02NANN by set-point OFFSET value of CH2 to 2mA(=K400_{LSB}) and GAIN value of CH2 to 18mA(=K3600_{LSB}).



- a) Write H10 to CR#1 of analog input module No. 1 and set CH2 to mode 2 (current input : +4mA to +20mA).
 - b) Write H0 to CR#33 to adjust characteristics of CH1 and CH2.
 - c) When X0 turns from OFF → ON, K400_{LSB} of OFFSET value will be written to CR#23 and K3600_{LSB} of GAIN value will be written to CR#29.
5. Example 5: Program example when ELC-AN04ANNN and ELC-AN02NANN module are used together
- a) Read the model type from extension module K0 and determine if the data is H88 is a ELC-AN04ANNN model type.
 - b) If the model type is ELC-AN04ANNN, set input mode CR#1: (CH1, CH3)= mode 0, (CH2, CH4)= mode 3.
 - c) Set the mode of CR#2 and CR#3. The average times of CH1 and CH2 is K32.
 - d) Read the input signal average value of CH1~CH4 (4 data) from CR#6~CR#9 and store them in D20 to D23.
 - e) Read the model type from extension module K1 and determine if the data is H49 is a ELC-AN02NANN model type.
 - f) D100 will increase K1 and D101 will increase K5 every second.
 - g) When value of D100 and D101 attain to K4000, they will be reset to 0.
 - h) If the model type is ELC- AN02NANN, the drive contact M1 = ON and set input mode CR#1: CH1 mode to 0, CH2 mode to 2.
 - i) Write the output set-point D100 and D101 to CR#10 and CR#11. The analog output will change with D100 and D101 value.

Ladder Diagram:



API	Mnemonic	Operands				Function				Controllers			
80	RS	S	m	D	n	Serial Data Communication				PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	RS: 9 steps
S														*			
m						*	*							*			
D														*			
n						*	*							*			

Operands:

S: Start device for transmitting data **m**: Transmitting data group number (**m**=0~256) **D**: Start device for receiving data **n**: receiving data group number (**n**=0~256)

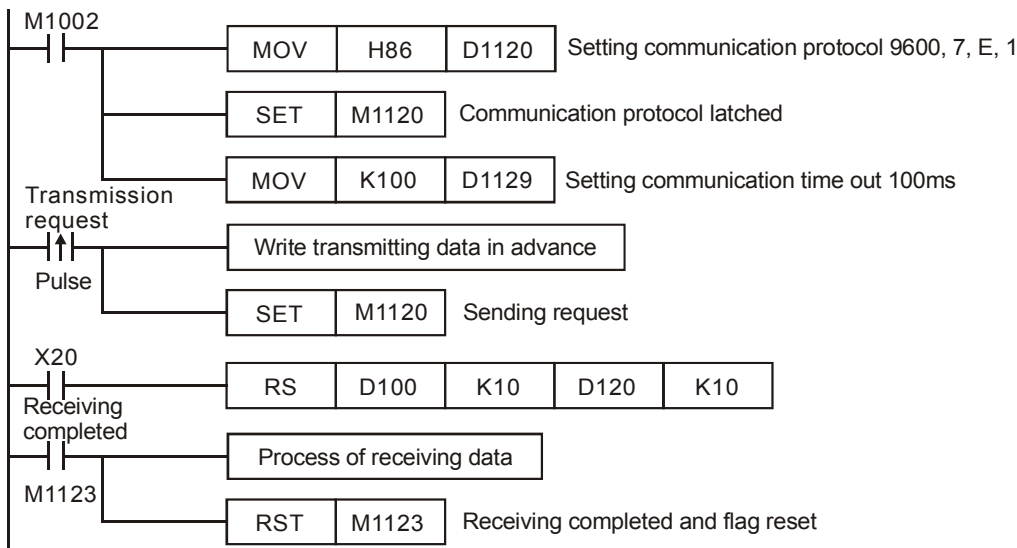
Explanations:

1. This instruction allows ELC to use RS-485 to communications. It stores word data in **S** and sets length **m**. It can set a receive data register **D** and length **n**.
2. If it doesn't need to transmit data, **m** can be indicated in K0 and if it doesn't need to receive data, **n** can be indicated in K0.
3. Two or more RS instruction cannot be executed at the same time.
4. During execution of the RS instruction, the transmitting data cannot be changed.
5. Use this RS instruction to transmit and receive data between ELC and external/peripheral equipment (AC drive, etc.).
6. If the communication format of external/peripheral equipment corresponds with the MODBUS communication format, ELC provides several communication instructions, MODRD, MODWR and MODRW.
7. Flag: M1120~M1131, M1140~M1143, M1161, see below examples.

Program Example 1:

1. Pre-write data into the register that starts from D100 and set M1122=ON (send request-flag).
2. If the RS instruction is executed when X20=ON, ELC will be in a waiting state, waiting for transmitted and received data. ELC will transmit the 10 data values that start from D100. M1122 will be set to OFF at the end of transmitting (do not use program to execute RST M1122). After 1ms delay, it will start to receive 10 external data values and store them into continuous registers that start from D120.
3. When the receive operation is complete, M1123 will be set to ON. The ELC program should set M1123 = OFF when receive is complete and in the state of transmitting data. Do not use ELC

program to execute RST M1123 continuously.

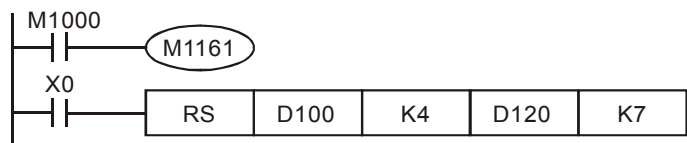


Program Example 2:

8-bit mode (M1161=ON) / 16-bit mode (M1161=OFF) switch:

8-bit mode:

1. Head code and tail code of ELC transmission data will be set by using M1126 and M1130 according to D1124~D1126. After set-point, ELC will send head code and tail code set by the user automatically when executing RS instruction.
2. When M1161=ON, the mode will be 8-bit. 16-bit data will be divided into high byte and low byte. High byte will be ignored and low byte will be received and transmitted.



Transmit data: (ELC→External equipment)

STX	D100L	D101L	D102L	D103L	ETX1	ETX2
Head code	<div><div>S</div> source data register will start from low byte of D100</div> <div><div>m</div> length = 4</div>				Tail code 1	Tail code 2

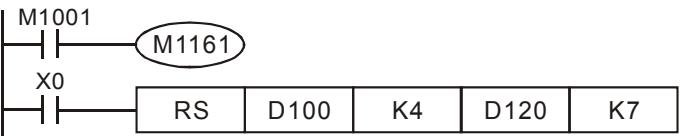
Receive data: (External equipment→ELC)

D120L	D121L	D122L	D123L	D124L	D125L	D126L
<div>Head code</div> <div> <div>S</div> receive data register will start from low byte of D120 </div> <div> <div>n</div> length = 7 </div> <div> Tail code 1 Tail code 2 </div>						

- ELC will receive all data transmitted from external equipment, including head code and tail code.
Pay attention on set-point length **n**.

16-bit mode:

- Head code and tail code of ELC transmitting data is set by using M1126 and M1130 with D1124~D1126. ELC will send head code and tail code set by the user automatically when executing RS instruction.
- When M1161=OFF, the mode will be 16-bit. 16-bit data will be divided into high byte and low byte for data transmitting and receiving.



Transmit data: (ELC→External equipment)

STX	D100L	D100H	D101L	D101H	ETX1	ETX2
<div>Head code</div> <div> <div>S</div> source data register will start from low byte of D100 </div> <div> <div>m</div> length = 4 </div> <div> Tail code 1 Tail code 2 </div>						

Receive data: (External equipment→ELC)

D120L	D120H	D121L	D121H	D122L	D122H	D123L
<div>Head code</div> <div> <div>D</div> receive data register will start from low byte of D120 </div> <div> <div>n</div> length = 7 </div> <div> Tail code 1 Tail code 2 </div>						

- ELC will receive all data transmitted from external equipment, including head code and tail code.
Pay attention on set-point length **n**.

Points to note:

1. Flags for RS-485 communication instructions RS / MODRD / MODWR / MODRW

Flag	Function Explanation	Action
M1120	Save Communication settings. If M1120 is set to ON, the communication parameters established in D1120 will be saved ELC will reset communication parameters according to special data register D1120 after the first program scan. When the second program scan starts and the RS instruction is executed, it will reset communication parameters according to special data register D1120. Once communication parameters are set, M1120 can be set to ON. At this time, communication parameters won't be reset as RS / MODRD / MODWR / MODRW is executed even if D1120 setting are changed.	User
M1121	When M1121 = OFF, RS-485 of ELC is transmitting	System
M1122	Send request. Set M1122 to ON, use the pulse instruction when using RS / MODRD / MODWR / MODRW instruction to transmit and receive data. If one of the above instructions starts to execute, ELC will transmit and receive data. M1122 will be cleared after above instructions completes transmitting.	User for setting and system clears
M1123	Receive complete. M1123 will be set to ON after RS / MODRD / MODWR / MODRW instructions complete execution. User can process the received data when M1123 is set to ON and then reset M1123 to OFF by user.	System sets ON and user clears
M1124	Receiving in process. When M1124 is set to ON, ELC is waiting for receiving data.	System
M1125	Receiving data state . When M1125 is set to ON, ELC communication state will be reset. After resetting the communications, M1125 must be reset to OFF.	User sets and clear
M1126	After executing the RS instruction, STX/ETX selection. Refer to the following table for selecting user/system definition and STX/ETX.	
M1130	After executing the RS instruction, STX/ETX selection. Refer to the following table for selecting user/system definition and STX/ETX.	

Flag	Function Explanation	Action
M1127	Communication instruction finishes transmitting and receiving. RS instruction is not included.	System sets and user clears
M1128	Transmitting/receiving indication	System
M1129	Receiving time out. This flag will be activated if D1129 is set and the process of receiving data is not completed within the established time. After resetting the communications, M1129 should be reset to OFF.	System sets and user clears
M1131	In ASCII mode, M1131=ON during MODRD / MODRW convert data to HEX. Otherwise M1131 will be OFF.	System
M1140	MODRD / MODWR / MODRW data received error	
M1141	MODRD / MODWR / MODRW instruction error	
M1143	ASCII / RTU mode selection, ON is RTU mode and OFF is ASCII mode. (Use with MODRD / MODWR / MODRW instructions)	User sets and clears
M1161	8/16-bit mode. ON = 8-bit mode and OFF =16-bit mode	

2. Special D registers related to RS-485 communication RS / MODRD / MODWR / MODRW

Special register	Function Explanation
D1038	Data response delay time setting when ELC is a slave. Time unit (0.1ms).
D1050~D1055	After executing the MODRD instruction, ELC will convert ASCII data of D1070~D1085 to HEX and store hexadecimal data to D1050~D1055.
D1070~D1085	ELC built-in RS-485 communication convenience instruction. Executing this instruction will receive feedback (return) messages from receiver. The messages will be stored at D1070~D1085. User can check return data by viewing the content of the register, not include RS instruction.
D1089~D1099	ELC built-in RS-485 communication convenience instruction. The transmitting message will be stored in D1089~D1099 when this instruction is executed. Users can check if the instruction is correct by the content of the register, not include RS instruction.
D1120	Refer to the following table for RS-485 communication protocol.
D1121	Communication address of ELC when ELC is slave.
D1122	Number of words to transmit.
D1123	Number of words received.
D1124	Start word (STX). Refer to the table above.
D1125	First end word (ETX1). Refer to the following table.
D1126	Second end word (ETX2). Refer to the following table.
D1129	Communication time out. Time unit (ms). If the value of the time is 0, it means no time out set. ELC will set M1129 = ON, if a timeout occurs after executing RS / MODRD / MODWR / MODRW instructions M1129 =ON.
D1130	MODBUS error code recorded, not include RS instruction.
D1168	Specific character communication interrupt required(I150)of RS instruction, it occurs interrupt I150 when received data character=D1168.
D1256~D1295	ELC built-in RS-485 communication convenience instruction MODRW. The characters transmitted by this instruction will be stored in

Special register	Function Explanation
	D1256~D1295 when this instruction is executed. User can check if the instruction is correct by the content of the registers.
D1296~D1311	After executing the MODRW instruction, ELC will automatically convert ASCII data in the receiving register specified by user to HEX, hexadecimal value.

3. D1120: RS-485 communication parameters. Bits b15-b11 is undefined.

	Content		
b0	Data Length	0: 7 data bits	1: 8 data bits
b1 b2	Parity bit	00: None 01: Odd 11: Even	
b3	Stop bits	0: 1 bit	1: 2bits
b4 b5 b6 b7	Baud rate	0001(H1): 110 0010(H2): 150 0011(H3): 300 0100(H4): 600 0101(H5): 1200 0110(H6): 2400 0111(H7): 4800 1000(H8): 9600 1001(H9): 19200 1010(HA): 38400 1011(HB): 57600 1100(HC): 115200	
b8	Start word selection	0: None	1: D1124
b9	First end word selection	0: None	1: D1125
b10	Second end word selection	0: None	1: D1126
b11~b15	No definition		

4. Bits b8-b10 control user defined start and end words. Setting any of these bits = ON, the value placed in the corresponding D1124-D1126 registers will be used for start and end words. When

using M1126, M1130, D1124~D1126 to set start and end word, b8~b10 of D1120 of RS485 communication protocol should be set to 1. For the set-points, refer to the following table:

		M1130	
		0	1
M1126	0	D1124: user define	D1124: H 0002
		D1125: user define	D1125: H 0003
		D1126: user define	D1126: H 0000 (no set-point)
	1	D1124: user define	D1124: H 003A (':')
		D1125: user define	D1125: H 000D (CR)
		D1126: user define	D1126: H 000A (LF)

5. Example for communication format:

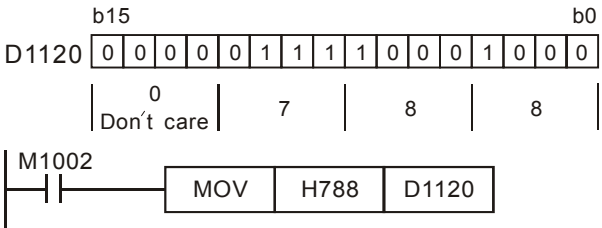
Communication format: Baud rate 9600 7, N, 2

STX : “:”

ETX1 : “CR”

ETX2 : “LF”

You can get the communication format H788 via check with table and write into D1120.



When using STX, ETX1 and ETX2 should pay attention to the ON/OFF relationship between special auxiliary relay M1126 and M1130.

6. M1143: ASCII / RTU mode selection. ON = RTU mode and OFF = ASCII mode.

Take standard MODBUS format to explanation:

ASCII mode (M1143=OFF):

Field Name	Descriptions
STX	Start word = ':' (3AH)
Address Hi	Communication address: 8-bit address consists of 2 ASCII codes
Address Lo	
Function Hi	Function code: 8-bit function code consists of 2 ASCII codes
Function Lo	
DATA (n-1)	Data content: n × 8-bit data content consists of 2n ASCII codes
.....	
DATA 0	
LRC CHK Hi	LRC check sum: 8-bit check sum consists of 2 ASCII code
LRC CHK Lo	
END Hi	End word: END Hi = CR (0DH), END Lo = LF(0AH)
END Lo	

Communication protocol is MODBUS ASCII mode. Each byte consists of 2 ASCII characters. For example: a 1-byte data 64 Hex shown as '64' in ASCII, consists of '6' (36Hex) and '4' (34Hex). The table below identifies the usable hexadecimal characters and their associated ASCII codes.

Character	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'
ASCII code	30H	31H	32H	33H	34H	35H	36H	37H

Character	'8'	'9'	'A'	'B'	'C'	'D'	'E'	'F'
ASCII code	38H	39H	41H	42H	43H	44H	45H	46H

Start word (STX): ':' (3AH)

Communication address (Address):

'0' '0': broadcast for all slaves (Broadcast)

'0' '1': toward the slave at address 01

'0' 'F': toward the slave at address 15

'1' '0': toward the slave at address 16 and consequently, the Max. address can be reached is 254 ('FE')

Function code (Function):

'0' '3': read contents from multiple registers

'0' '6': write one WORD into a single register

'1' '0': write contents to multiple registers

Data content:

The content of transmitting data send by user

LRC check:

LRC check is the added sum from “Address” to “Data contents”. For example, the 01H + 03H + 21H + 02H + 00H + 02H = 29H, then take the complementary of 2, D7H.

End word:

END Hi = CR (0DH), END Lo = LF(0AH)

For example: when the address of the drive is set as 01H, read 2 data contents that exist successively within the register, as shown follows: the address of the start register is 2102H.

Inquiry message:

STX	‘.’
Address	‘0’
	‘1’
Function	‘0’
	‘3’
Start address	‘2’
	‘1’
	‘0’
	‘2’
Number of data (count by word)	‘0’
	‘0’
	‘0’
	‘2’
LRC Check	‘D’
	‘7’
END	CR
	LF

Response message:

STX	‘.’
Address	‘0’
	‘1’
Function	‘0’
	‘3’
Number of data (count by byte)	‘0’
	‘4’
Content of start address 2102H	‘1’
	‘7’
	‘7’
	‘0’
Content of address 2103H	‘0’
	‘0’
	‘0’
	‘0’
LRC Check	‘7’
	‘1’
END	CR
	LF

RTU mode (M1143=ON):

Field Name	Descriptions
START	Please refer to following explanation
Address	Communication address: 8-bit binary
Function	Function code: 8-bit binary
DATA (n-1)	Data content: n × 8-bit data
.....	
DATA 0	
CRC CHK Low	CRC check: 16-bit CRC consists of 2 8-bit binary
CRC CHK High	
END	Please refer to following explanation

START:

RTU Timeout Timer: keep no input signal $\geq 10\text{ms}$.

Communication Address (Address):

00 H: broadcast for all driver (Broadcast)

01 H: toward the drive at the 01 address

0F H: toward the drive at the 15 address

10 H: toward the drive at the 16 address and consequently, the Max. address can be reached is 254 ('FE')

Function code (Function):

03 H: read contents from multiple registers

06 H: write one WORD into the register

10 H: write contents to multiple registers

Data content:

The content of transmitting data send by user

CRC check: CRC check starts from "Address" and ends in "Data content". Its calculation is as follows:

1. Step 1: Load the 16-bit register (the CRC register) with FFFFH.
2. Step 2: Exclusive OR the first 8-bit byte message instruction with the 16-bit CRC register of the low byte, then store the result into the CRC register.
3. Step 3: Shift the CRC register one bit to the right and fill 0 in the higher bit.
4. Step 4: Check the value that shifts to the right. If it is 0, store the new value from Step 3 into the CRC register, otherwise, Exclusive OR A001H and the CRC register, then store the result into the CRC register.
5. Step 5: Repeat Step 3 and 4 and calculates the 8-bits complete.
6. Step 6: Repeat Steps 2~5 for the next 8-bit message instruction, till all the message instructions are processed. And finally, the obtained CRC register value is the CRC check value. What should be noticed is that the CRC check must be placed interchangeably in the check sum of the message instruction.

END: RTU Timeout Timer: keep no input signal $\geq 10\text{ms}$.

For example: when the address of the drive is set as 01H, read 2 continuous data of the register shown follows: the address of the start register is 2102H

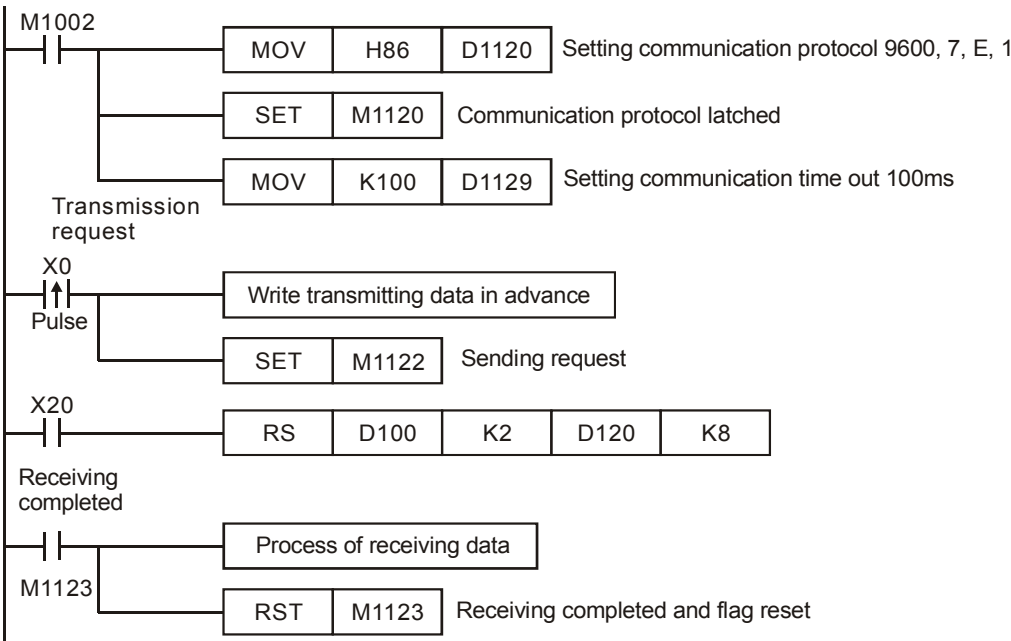
Inquiry message:

Field Name	Example(Hex)
Address	01 H
Function	03 H
Start data address	21 H
	02 H
Number of data (count by word)	00 H
	02 H
CRC CHK Low	6F H
CRC CHK High	F7 H

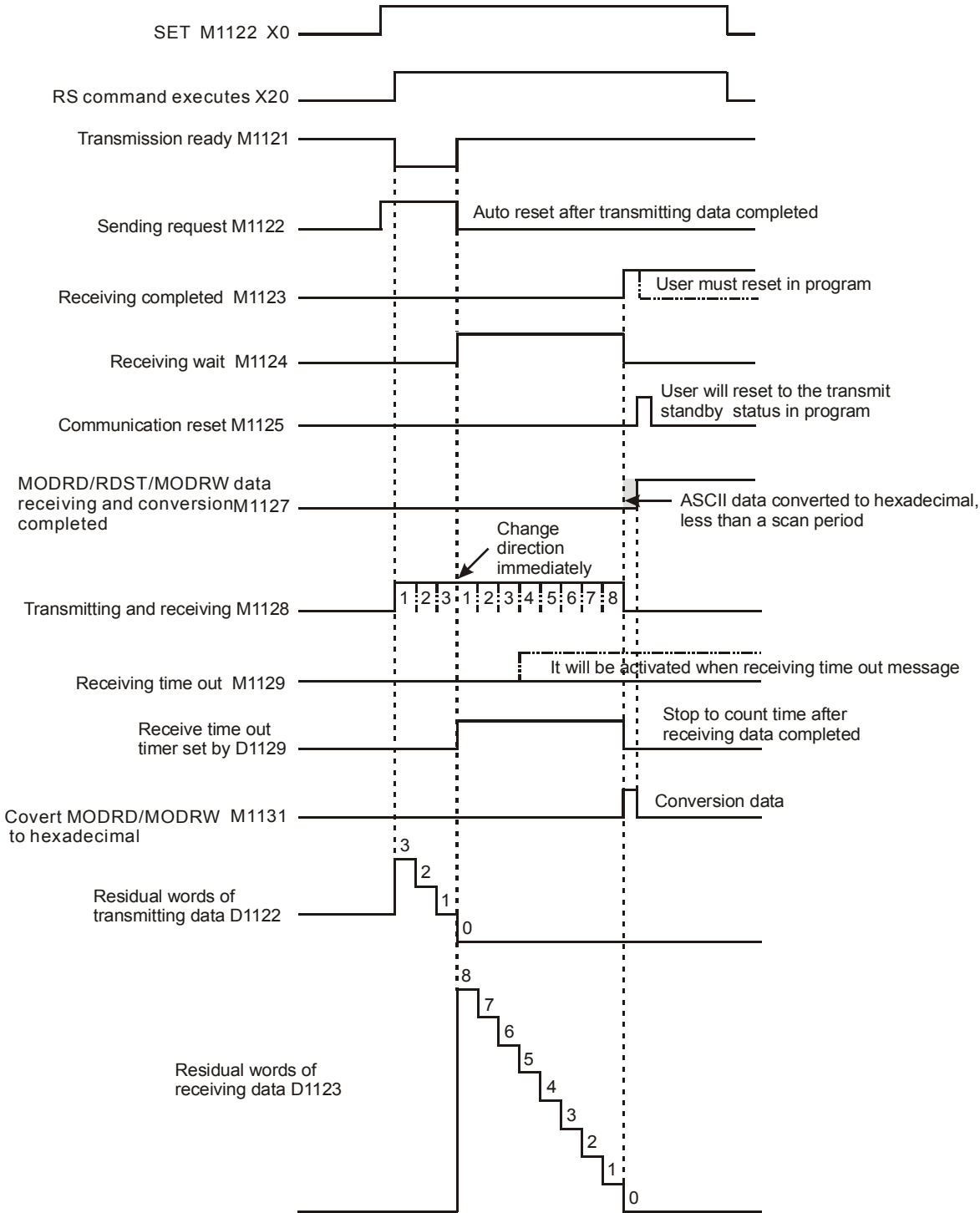
Response message:

Field Name	Example(Hex)
Address	01 H
Function	03 H
Number of data (count by byte)	04 H
Content of data address 8102H	17 H
	70 H
Content of data address 8103H	00 H
	00 H
CRC CHK Low	FE H
CRC CHK High	5C H

Timing chart of RS-485 communication program flag:



Timing chart:



API	Mnemonic			Operands		Function										Controllers			
81	D	PRUN	P	S	D	Parallel Run										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	PRUN, PRUNP: 5 steps DPRUN, DPRUNP: 9 steps			
S							*		*										
D								*	*										

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

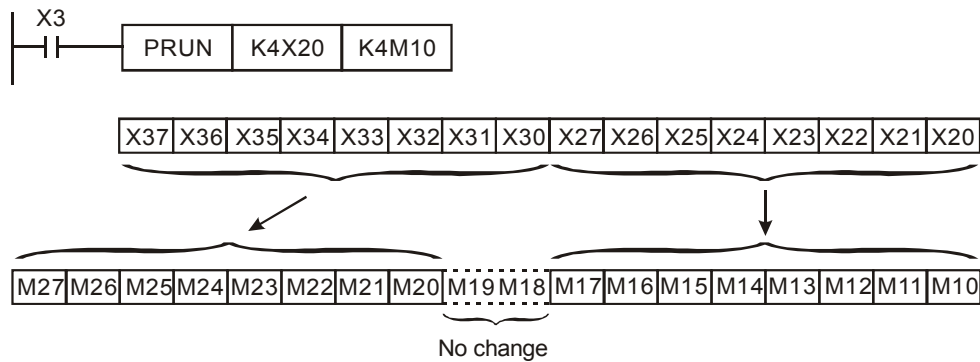
S: Transmission source device **D:** Destination device

Explanations:

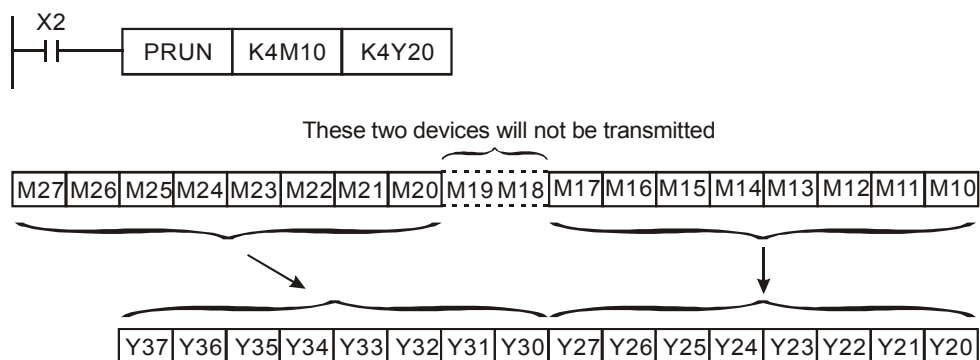
1. Transmit the content of **S** to **D** in octal number system format.
2. X, Y, M of KnX, KnY, KnM should be a multiple of 10, e.g. X20, M20, Y20.
3. When operand **S** is specified as KnX, operand **D** should be specified as KnM.
4. When operand **S** is specified as KnM, operand **D** should be specified as KnY.

Program Example 1:

When X3=ON, transmit the content of K4X20 to K4M10 in octal number system format.

**Program Example 2:**

When X2=ON, transmit the content of K4M10 to K4Y20 in octal number system format.



API	Mnemonic			Operands			Function								Controllers			
82		ASCI	P	(S)	(D)	(n)	Converts HEX to ASCII								PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ASCII, ASCIP: 7 steps			
S					*	*	*	*	*	*	*	*	*						
D								*	*	*	*	*	*						
n					*	*													

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

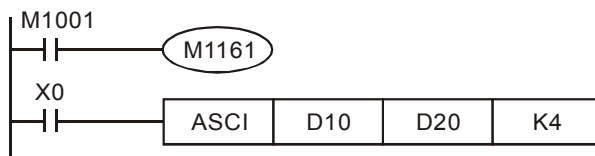
S: Source data **D:** Destination of result **n:** Number of digits to convert (**n**=1~256)

Explanations:

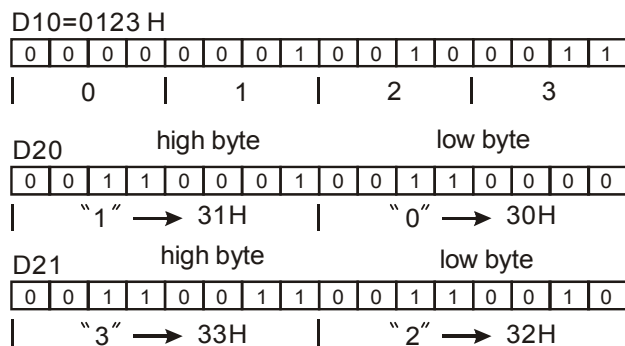
- 16-bit conversion mode: When M1161=OFF, read **n** hexadecimal characters from source **S** and convert them to ASCII. Then, store the result into high and low byte of **D**.
- 8-bit conversion mode: When M1161=ON, read **n** hexadecimal characters from source **S** and convert them to ASCII. Then, store the result into the low byte of **D** (high byte of **D** will be set to 0).

Program Example 1:

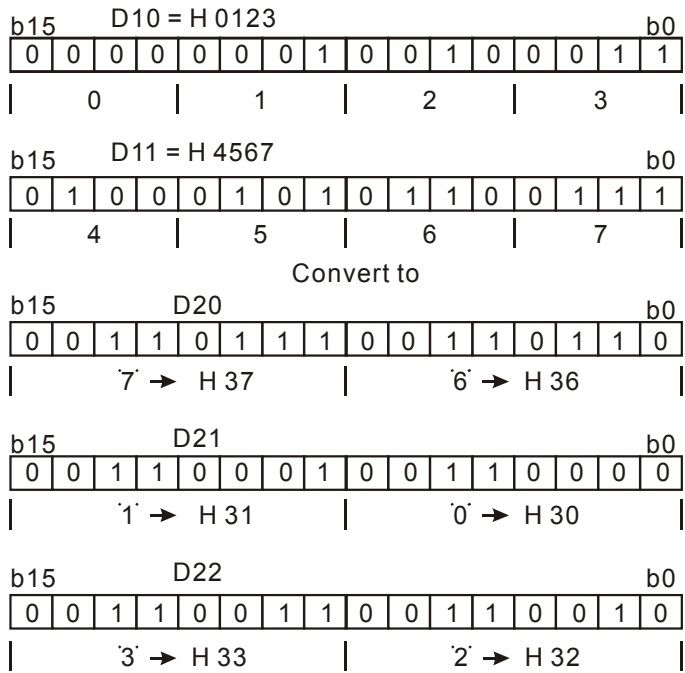
- When M1161=OFF, conversion mode is 16-bit.
- When X0=ON, read four hexadecimal characters from D10 and convert them into ASCII. Then, store the converted characters in the register started from D20.



- Supposed condition:
D10=0123 H, D11=4567H, D12=89ABH, D13=CDEFH
- When **n** is 4, the bit structure is:



5. When **n** is 6, the bit structure is:



6. When **n** = 1 to 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								
D25 low byte								
D25 high byte								
D26 low byte								
D26 high byte								
D27 low byte								
D27 high byte								

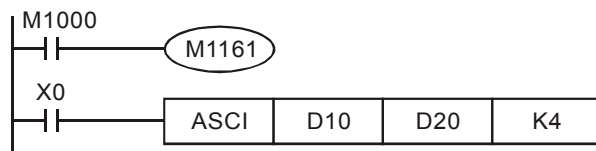
No
change

$\begin{matrix} n \\ D \end{matrix}$	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						"3"	"2"	"1"
D27 low byte							"3"	"2"
D27 high byte								"3"

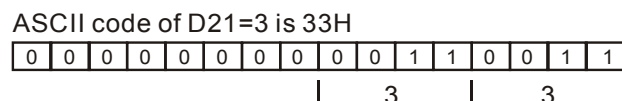
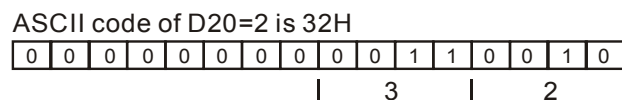
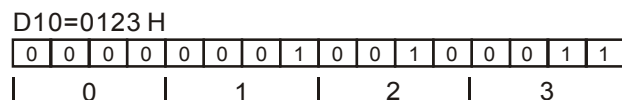
No
change

Program Example 2:

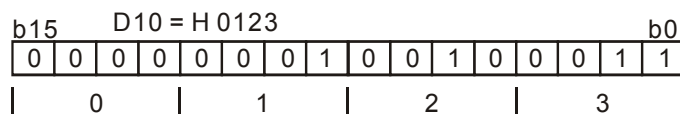
1. When M1161=ON, conversion mode is 8-bit.
2. When X0=ON, read four hexadecimal data characters from D10 and convert them into ASCII. Then, store the converted data to the register started from D20.



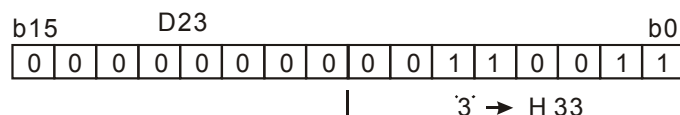
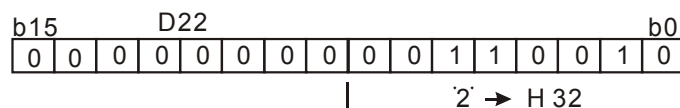
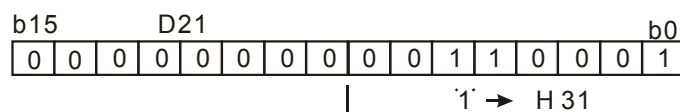
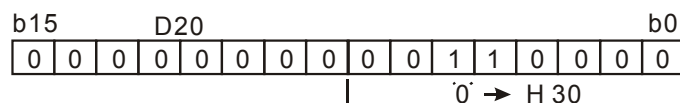
3. Supposed condition:
D10=0123H, D11=4567H, D12=89ABH, D13=CDEFH
4. When n is 2, the bit structure is:



5. When n is 4, the bit structure is:



Convert to



6. When $n = 1$ to 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								
D30								
D31								
D32								
D33								
D34								
D35								

No
change

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		"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"

No
change

API	Mnemonic			Operands			Function								Controllers			
83		HEX	P	S	D	n	Converts ASCII to HEX								PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	HEX, HEXP: 7 steps		
	S					*	*	*	*	*	*	*	*	*					
	D							*	*	*	*	*	*	*					
	n					*	*												

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

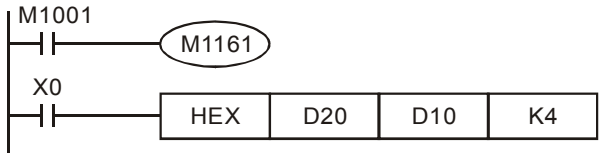
S: source data D: Destination for stored result n: number of digits to convert (n=1~256)

Explanations:

- 1. 16-bit conversion mode: When M1161=OFF, conversion mode is 16-bit. Convert 16-bit ASCII data of S (high and low byte) to hexadecimal and then store in D per 4-bit for one time. Number of converted ASCII characters is set by n.
- 2. 8-bit conversion mode: When M1161=ON, it is 16-bit conversion mode. Convert 16-bit ASCII code of S (high and low byte) to hexadecimal data characters and then transmit to low byte of D. Number of converted ASCII codes is set by n. (high byte of D are all 0)

Program Example 1:

- 1. When M1161=OFF, it is 16-bit conversion mode.
- 2. When X0=ON, read ASCII bytes of the register started from D20 and convert them to hexadecimal characters. Then, store the converted data to four registers started from D10. (The converted data is four characters converted as one segment of data)



3. Supposed condition:

S	ASCII code	HEX conversion	S	ASCII code	HEX conversion
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** is 4, the bit structure is:

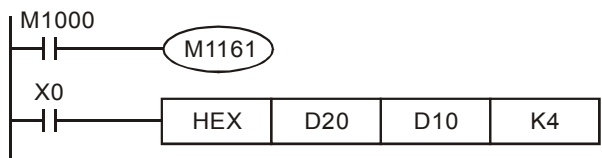
D20	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1
	44H → D								43H → C							
D21	0	1	0	0	0	1	1	0	0	1	0	0	0	1	0	1
	46H → F								45H → E							
D10	1	1	0	0	1	1	0	1	1	1	1	0	1	1	1	1
	C		D		E		F									

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

n \ D	D13	D12	D11	D10
1	The used registers which are not specified 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:

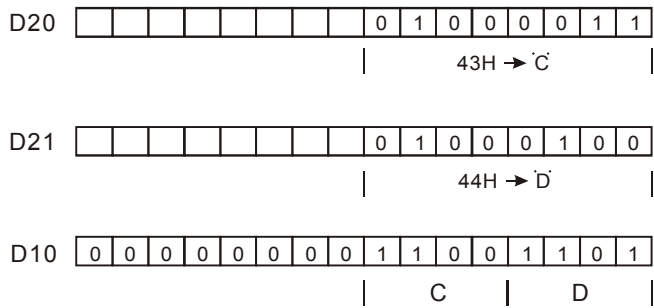
1. When M1161=ON, it is 16-bit conversion mode.



2. Supposed condition:

S	ASCII code	HEX conversion	S	ASCII code	HEX conversion
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”

3. When n is 2, the bit structure is



4. When $n = 1$ to 16:

$n \backslash D$	D13	D12	D11	D10
1	The used registers which are not specified 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

API	Mnemonic			Operands			Function								Controllers													
84		CCD	P	(S)	(D)	(n)	Check Code								PB	PC	PA	PH										
OP	Type	Bit Devices				Word devices										Program Steps												
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	CCD, CCDP: 7 steps												
S							*	*	*	*	*	*	*															
D									*	*	*	*	*															
n					*	*							*															
																	PULSE				16-bit				32-bit			
																	PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

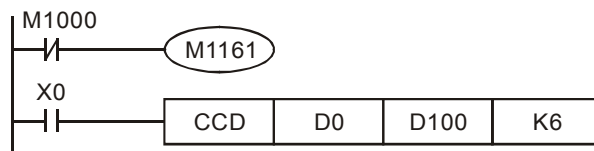
S: source data **D:** Destination for storing check sum **n:** Number of values to use in the instruction (n=1~256)

Explanations:

1. This instruction is used to create check sum of words to ensure data integrity of transmitted information.
2. 16-bit conversion mode: When M1161=OFF, conversion mode is 16-bit. Create a checksum of **n** words (8-bit in one byte) from the register specified by **S** and store the sum in the register specified **D**, the parity bits store in **D** +1.
3. 8-bit conversion mode: When M1161=ON, conversion mode is 8-bit. Create a checksum of **n** words (8-bit in one byte, only low byte are available) from the register specified by **S** and store the sum in the register specified **D**, the parity bits store in **D** +1

Program Example 1:

1. When M1161=OFF, conversion mode is 16-bit.
2. When X0=ON, checksum of 6 words from the register specified by D0 (8-bit in one byte, n=6 specifies D0~D2) and store the checksum in the register specified by D100 while the parity bits are stored in D101.



(S)	Content of data(words)	
D0 low byte	K100 = 0 1 1 0 0 1 0 0	
D0 high byte	K111 = 0 1 1 0 1 1 1 ① ←	
D1 low byte	K120 = 0 1 1 1 1 0 0 0	
D1 high byte	K202 = 1 1 0 0 1 0 1 0	
D2 low byte	K123 = 0 1 1 1 1 0 1 ① ←	
D2 high byte	K211 = 1 1 0 1 0 0 1 ① ←	
D100	K867	Total
D101	0 0 0 1 0 0 0 ①	

• An even result is indicated by the use of 0(zero)

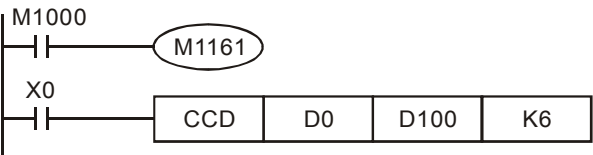
• An odd result is indicated by the use of 1(one)

D100 0 0 0 0 0 0 1 1 0 1 1 0 0 0 1 1

D101 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 ← Parity

Program Example 2:

- When M1161=ON, it is 8-bit conversion mode.
- When X0=ON, checksum of 6 words from the register specified by D0 (8-bit in one byte, n=6 means to specify D0~D5) and store the checksum in the register specified by D100 while the parity bits are stored in D101.



(S)	Content of data(words)	
D0 low byte	K100 = 0 1 1 0 0 1 0 0	
D1 low byte	K111 = 0 1 1 0 1 1 1 ① ←	
D2 low byte	K120 = 0 1 1 1 1 0 0 0	
D3 low byte	K202 = 1 1 0 0 1 0 1 0	
D4 low byte	K123 = 0 1 1 1 1 0 1 ① ←	
D5 low byte	K211 = 1 1 0 1 0 0 1 ① ←	
D100	K867	Total
D101	0 0 0 1 0 0 0 ①	

• An even result is indicated by the use of 0(zero)

• An odd result is indicated by the use of 1(one)

D100 0 0 0 0 0 0 1 1 0 1 1 0 0 0 1 1

D101 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 ← Parity

API	Mnemonic			Operands		Function											Controllers										
85		VRRD	P	S	D	Volume Read											PB	PC	PA	PH							
OP	Type	Bit Devices				Word devices										Program Steps											
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	VRRD, VRRDP: 5 steps											
	S					*	*																				
	D								*	*	*	*	*	*	*												
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

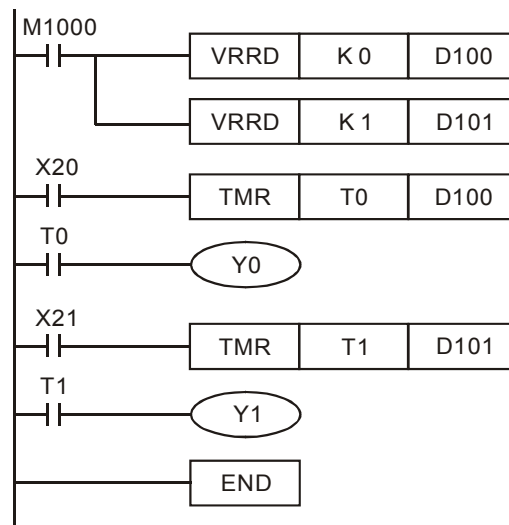
S: Variable resistor number (0~1) **D:** Destination for storing read value

Explanations:

- VRRD instruction is used to read the two variable resistors of ELC-PC models. The read value will be converted as value between 0 to 255 and stored in destination **D**.
- Flags: M1178 and M1179. (See the Note)

Program Example:

Variable resistor are read in order: S=K0 to K1 corresponding to the 2 volumes, No.0 to No.1. The timer loop converts the variable resistor value 0~255. The time unit for T0 to T1 is 0.1 second, therefore, the set-point value is 0 to 25.5 seconds

**Note:**

- VR means VARIABLE RESISTOR.
- PC/PH models, built-in 2 points VR volume can be used with special D and special M.

Device	Function
M1178	Start volume VR0
M1179	Start volume VR1
D1178	VR0 value
D1179	VR1value

API	Mnemonic			Operands		Function										Controllers			
86		VRSC	P	S	D	Volume Scale Read										PB	PC	PA	PH
OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	VRSC, VRSCP: 5 steps		
	S					*	*												
D								*	*	*	*	*	*	*	*				
				PULSE				16-bit				32-bit							
				PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH				

Operands:

S: Variable resistor number(0~1) D: Destination scaled value

Explanation:

VRSC reads the variable resistor scale value of two volumes on ELC main processing unit and the number is No.0 and No.1., (volume scale value is from 0 to 10). The read data will be stored in destination device **D** as an integer from the range 0 to 10.

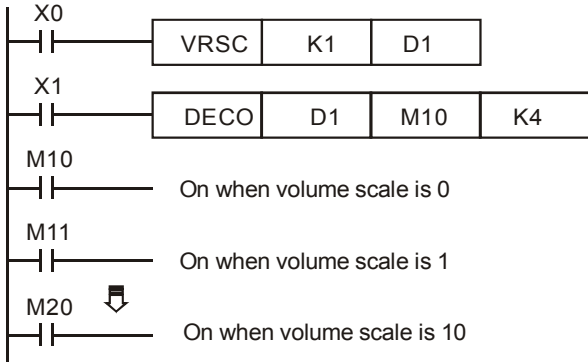
Program Example 1:


When X0=ON, the volume scale value (0 to 10) of No. 0 specified by VRSC instruction is stored in device D10.



Program Example 2:

1. Regard as digital switch: Correspond volume scale 0 to 10. Only one contact = ON between M10 to M20. Using DECO instruction (API 41) can decode the volume scale into M10~M25.
2. When X0=ON, store the volume scale value (0 to 10) of specified No. 1 volume to D1.
3. When X1=ON, use DECO instruction (API 41) to decode the volume scale into M10~M25.



API	Mnemonic			Operands	Function										Controllers			
87	D	ABS	P		Absolute Value										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ABS, ABSP: 3 steps			
	D								*	*	*	*	*	*	*	*	DABS, DABSP: 5 steps		

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

API	Mnemonic			Operands				Function	Controllers			
88	D	PID		S₁	S₂	S₃	D	PID Calculation	PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S ₁														*			PID : 9 steps DPID: 17 steps
S ₂														*			
S ₃														*			
D														*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Target value (SV) **S₂**: Present value (PV) **S₃**: Parameter (for 16-bit instruction, uses 20 continuous devices, for 32-bit instruction, uses 21 continuous devices) **D**: Output value (MV)

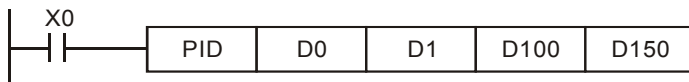
Explanations:

PID instruction will start execution after completing all parameter settings and the result will be stored in

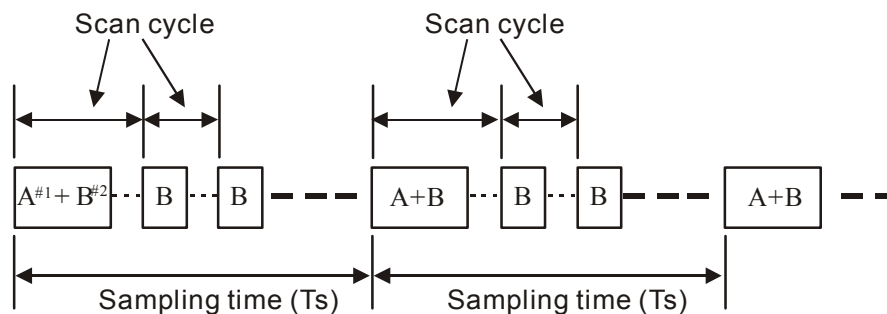
D. Do not latch **D** register area.

Program Example:

- Pre-write the PID parameters into all the registers before executing the PID instruction.
- This instruction will be executed when X0=ON and the result will be stored in D150. The instruction will not be executed when X0=OFF and the previous data will be unchanged.



- Timing chart of the PID instruction (max. operation time is 470us)



Note:#1→It is the time of equation calculation during PID operation (about 430us)

#2→It is the PID operation time without equation calculation (about 44us)

Points to note:

- There is no number-of-times limit when using the PID instruction, but the specified devices **S₃~S₃+19** cannot be repeated use.

2. For 16-bit instructions: S_3 uses 20 registers. In the example below the parameter area of the PID instruction that S_3 indicates are D100~D119.
3. Parameter table of 16-bit S_3 :

Device No.	Function	Set-point range	Explanation
S ₃ :	Sampling time (T _S)	1~2,000 (unit: 10ms)	If T _S is less than one program scan time, the PID instruction will execute one time. If T _S =0, the PID instruction won't be executed.
S ₃ +1:	Propotional gain (K _P)	0~30,000(%)	When value exceeds 30,000, the value will be set to 30,000.
S ₃ +2:	Integral gain (K _I)	0~30,000(%)	
S ₃ +3:	Differential gain (K _D)	-30,000~30,000(%)	
S ₃ +4:	Control method	0: normal control 1: Forward control (E=SV-PV) 2: Inverse control (E=PV-SV) 3: Auto-tuning for temperature control 4: Temperature control	
S ₃ +5:	The range in which Error value (E) will not work	0~32,767	For example:If the error range of value (E) is set to 5. If the error value E between -5~5, the MV will output 0.
S ₃ +6:	Upper limit of saturated output (MV)	-32,768~32,767	Example: if upper limit is set to 1000 and if output (MV) is larger than 1000, it will output 1000. (upper limit should be larger than lower limit, i.e. S ₃ +6 > S ₃ +7.)
S ₃ +7:	Lower limit of saturated output (MV)	-32,768~32,767	For example:If lower limit is set to -1000 and if the MV is less than -1000, it will output -1000.
S ₃ +8:	Upper limit of saturated integration	-32,768~32,767	For example:If upper limit is set to 1000 and if accumulated integral value is larger than 1000, it will output 1000 and won't integrate. (upper limit should be larger than lower limit, i.e. S ₃ +8 >= S ₃ +9.)
S ₃ +9:	Lower limit of saturated integration	-32,768~32,767	For example:If lower limit is set to -1000 and if accumulated integral value is less

Device No.	Function	Set-point range	Explanation
			than -1000, it will output -1000 and won't integrate. If S3+8 & S3+9 are both set to 0, the limit of integration is disable.
S ₃ +10, 11:	Save accumulated integral value temporality	32-bit floating point range	It is used usually for reference but user can clear or modify. (must be modified using 32-bit floating point)
S ₃ +12:	Save previous PV value temporality	-32,768~32,767	This previous PV is usually for reference. But user can modify.
S ₃ +13: ~ S ₃ +19:	Reserved for system usage, do not use.		

4. When parameter set-point is out of set-point range, it will be set to upper limit or lower limit. But if the operation is out of range, it will be set to 0.
5. PID instructions can be used in interrupt subroutine, Step point and CJ instruction.
6. Max. error range of sampling time T_s is $-(a \text{ scan time} + 1\text{ms}) \sim +(a \text{ scan time})$. If error value has influence on output, please keep the scan time fixed or execute PID instruction in an interrupt subroutine of timer.
7. The present value (PV) must be a stable value before the execution of the PID instruction. If using input value from special modules to perform the PID calculation, pay attention to any conversion time of the above-mentioned modules.
8. 32-bit instruction S₃ occupies 21 registers. Pre-write the parameter area of the PID instruction that designated by S₃ is D100~D120 before executing PID instruction.
9. 32-bit instruction S₃+4 device is not support k3 and k4.

10. Parameter table of 32-bit S_3 :

Device No.	Function	Set-point range	Explanation
S₃ :	Sampling time (T _S)	1~2,000 (unit: 10ms)	If T _S is less than one program scan time, the PID instruction will execute one time. If T _S =0, the PID instruction won't be executed.
S₃ +1:	Propotion gain (K _P)	0~30,000(%)	When value exceeds 30,000, the value will be set to 30,000.
S₃ +2:	Integration gain (K _I)	0~30,000(%)	
S₃ +3:	Differential gain (K _D)	-30,000~30,000(%)	
S₃ +4:	Control method	0: normal control 1: Forward control (E=SV-PV) 2: Inverse control (E=PV-SV)	
S₃ +5, 6:	The range that 32-bit Error value (E) doesn't work	0~ 2,147,483,647	For example:If the error range of value (E) is set to 5. If the error value E between -5~5, the MV will output 0.
S₃ +7, 8:	Upper limit of 32-bit saturated output (MV)	-2,147,483,648~ 2,147,483,647	For example:If upper limit is set to 1000 and if the MV is larger than 1000, it will output 1000. (upper limit should be larger than lower limit, i.e. S ₃ +7,8 > S ₃ +9,10.)
S₃ +9, 10:	Lower limit of 32-bit saturated output (MV)	-2,147,483,648~ 2,147,483,647	For example:If lower limit is set to -1000 and if the MV is less than -1000, it will output -1000
S₃ +11, 12:	Upper limit of 32-bit saturated integrator	-2,147,483,648~ 2,147,483,647	For example:If upper limit is set to 1000 and if accumulated integral value is larger than 1000, it will output 1000 and won't integrate. (upper limit should be larger than lower limit, i.e. S ₃ +11,12 >= S ₃ +13,14.)
S₃ +13, 14:	Lower limit of 32-bit saturated integrator	-2,147,483,648~ 2,147,483,647	For example:If lower limit is set to -1000 and if accumulated integral value is less than -1000, it will output -1000 and won't integrate.

Device No.	Function	Set-point range	Explanation
S₃ +15, 16:	32-bit temporary accumulation integral value	32-bit floating point range	It is usually for reference, but user can clear or modify. (needs to modified by 32-bit floating point)
S₃ +17, 18:	32-bit save previous PV temporality	-2,147,483,648~ 2,147,483,647	The previous PV is usually for reference. But user can modify.
S₃ +19, 20	Reserved for system usage, do not use.		

Explanation of 32-bit **S₃** and 16-bit **S₃** are almost the same. The difference is the capacity of **S₃+5 ~ S₃+20**.

PID Equations:

1. This instruction executes a PID calculation according to speed and differential of a measured value.
2. The PID operation has five control methods for 16 bit instruction: normal, forward inverse, auto-tuning and temperature controls. The control method is set by **S₃ +4**.
3. PID equations for control method k0~k2:

$$MV = K_p * E(t) + K_I * E(t) \frac{1}{S} + K_D * PV(t)S$$

Control Method	PID Equations
Forward control, normal control(k0,k1)	$E(t) = SV - PV$
Inverse control(k2)	$E(t) = PV - SV$

When $PV(t)S$ is the differential value of $PV(t)$, and $E(t) \frac{1}{S}$ is the integral value of $E(t)$.

Notice this instruction is different from the general PID instruction above. The difference is the change of differential usage. To avoid a too large transient differential value when executing general PID instruction for the first time, this instruction will lower the output (MV) value once the change of present value (PV) is too large by monitoring the differential value of present measured value (PV).

4. Symbols explanation:

MV : Output value

$E(t)$: Error value. Forward control $E(t) = SV - PV$, Inverse control $E(t) = PV - SV$

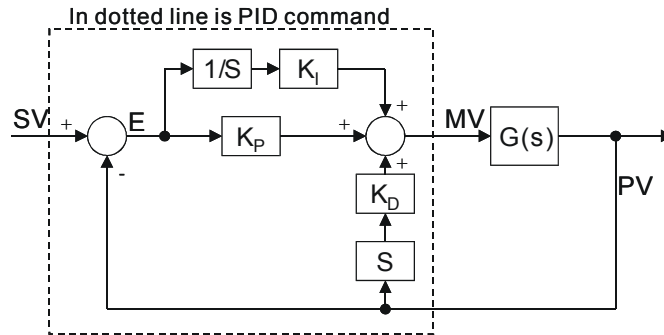
K_p : Porprootional gain

PV : Present measured value

SV : Target value

- K_D : Differential gain
 $PV(t)S$: Differential value of $PV(t)$
 K_I : Integral gain
 $E(t)\frac{1}{S}$: Integral value of $E(t)$

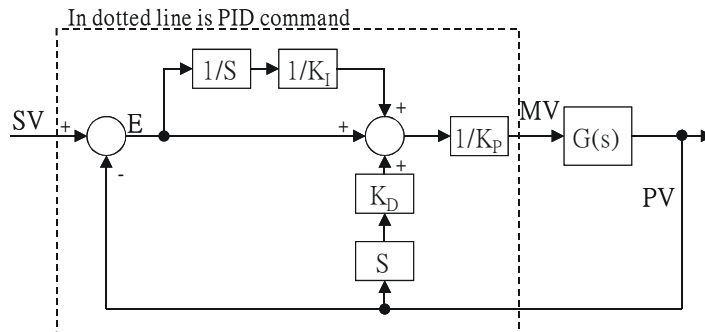
5. Control diagram for control method K0~K2:



6. PID equations for control method k3~k4:

$$MV = \frac{1}{K_P} \left[E(t) + \frac{1}{K_I} \left(E(t) \frac{1}{S} \right) + K_D * PV(t)S \right], \text{ where } E(t) = SV - PV$$

7. Control diagram for control method K3~K4::



8. Control method k3 and k4:

- The functions are specific design for temperature control. When sampling time (T_s) is set to 4 seconds (k400), indicates output range of MV will be from k0 to k4000, and cycle time of the GPWM instruction should be set to 4 seconds (k4000).
- If you don't know how to set each parameter of S_3 in temperature control environment, first you can set S_3+4 to k3 for auto-tuning. After finished auto-tuning (control method will be set to k4, and

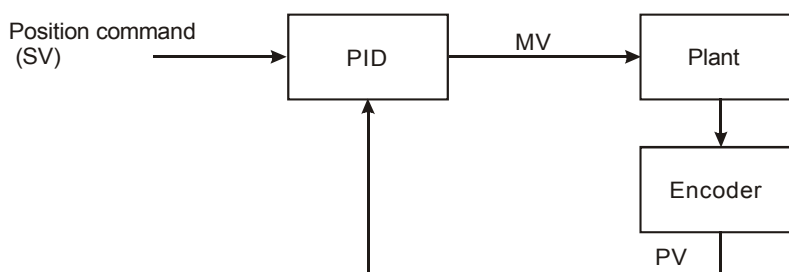
found other parameters of S_3), and you can use the parameters of S_3 to better setting depend on control result.

Notes and suggestion:

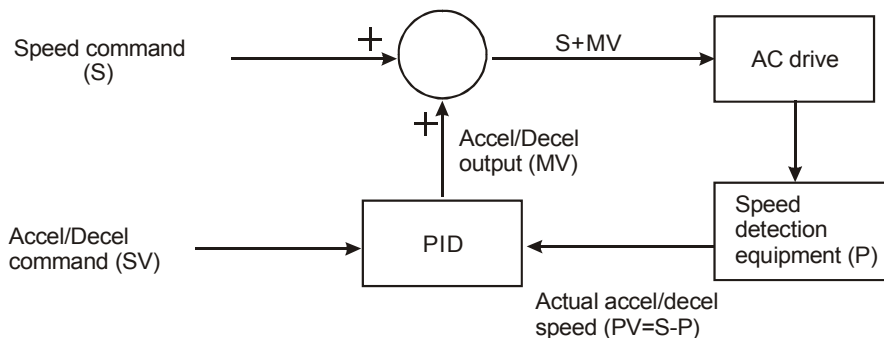
When adjusting the three major parameters, KP, KI and KD, adjust KP first and set 0 to KI and KD.

When adjusting to control, adjust KI (order from small to large) and KD (order from small to large). Refer to example 4 for adjusting. If KP=100, is 100%. When KP is less than 100%, the error value will attenuate and when KP is more than 100%, the error value will amplify.

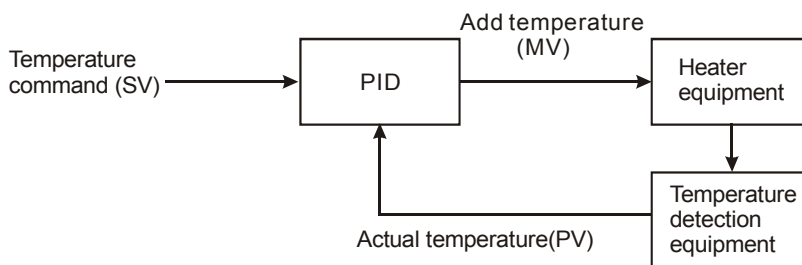
Example 1: PID instruction to control position (control method S_3+4 should be set to 0)



Example 2: PID instruction to control speed (control method S_3+4 should be set to 0)



Example 3: PID instruction to control temperature (action direction S_3+4 should be set to 1)



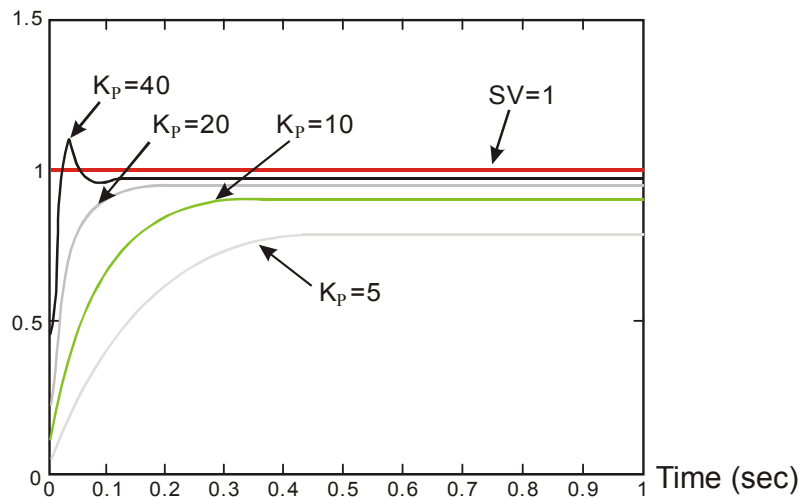
Example 4: Suggested steps of PID adjustment

Consider the transfer function of plant $G(s) = \frac{b}{s+a}$ (general AC drive function) in control system,

instruction value SV is 1 and sampling time Ts is 10ms. Suggested Steps are as follows:

Step1:

Set K_I and K_D to 0 first, then set K_P to 5, 10, 20 and 40 in order and record (SV) and (PV) state. The result will be shown as following figure.

**Step 2:**

In above figure, we will choose the situation K_P is 10. For the following:

When K_P is 40, response overshoot occurs. We won't use it.

When K_P is 20, PV response is close to SV and won't overshoot but transient MV will be to large due to a fast start-up. We also won't use it.

When K_P is 10, PV response is close to SV and is smooth. We can consider using it.

When K_P is 5, the response is too slow. So we won't use it.

Step 3: When deciding to use the curve $K_P=10$, arrange K_I in order from small to large (such as 1, 2, 4, 8) and not larger than K_P . Then arrange K_D in order from small to large (such as 0.01, 0.05, 0.1 and 0.2) and should not exceed 10% K_P . You can get following PV and SV relationship.

Application 1:

PID instruction in pressure control system. (Use block diagram of example 1)

Control destination:

Making the control system reach the pressure target value.

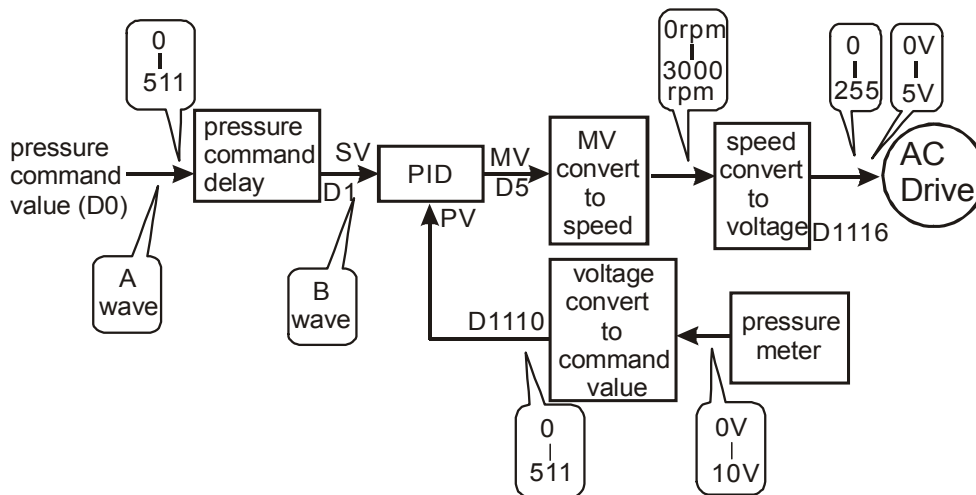
Control characteristics:

System should reach the destination. The system may go out of control or overload if it reaches the destination too fast.

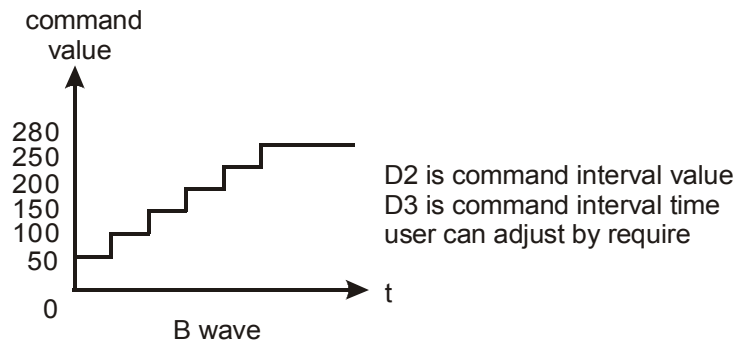
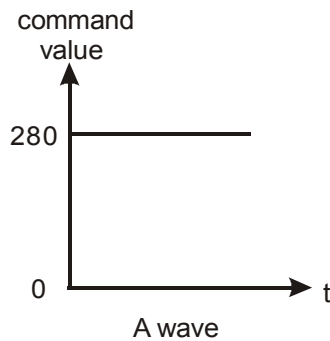
Recommend solve method:

Method 1: reach by using a long sampling time.

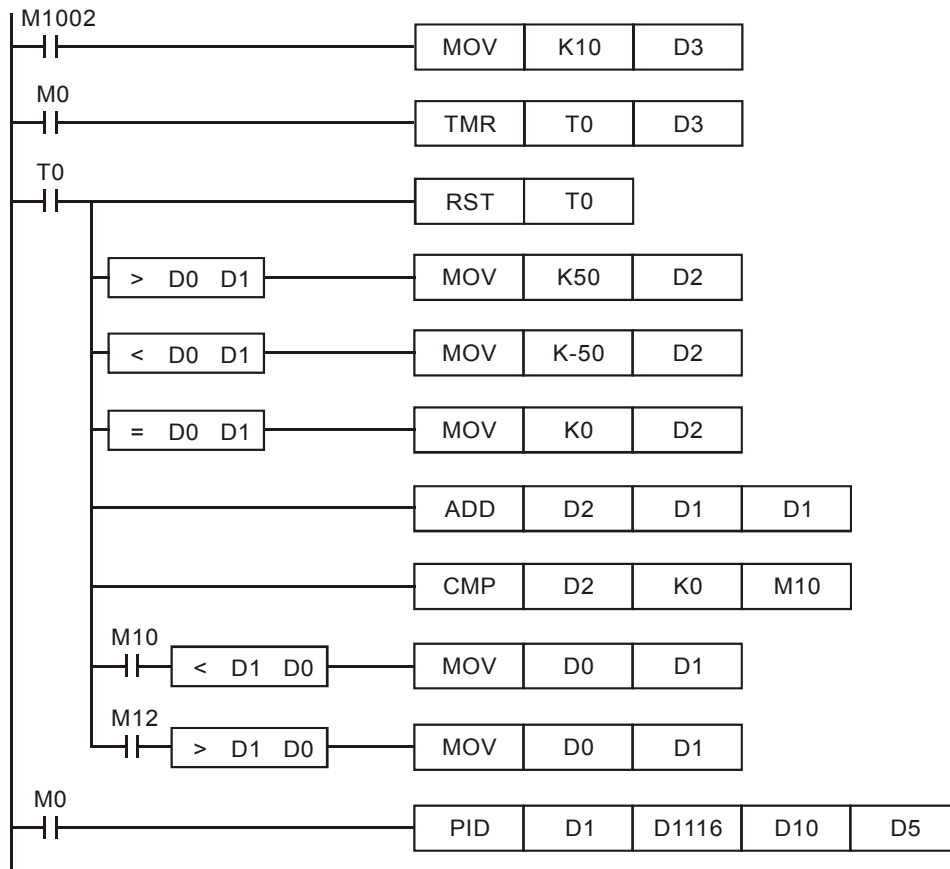
Method 2: reach by using a delay instruction and shown below



Applications:



Example for DELAY program:



Application 2:

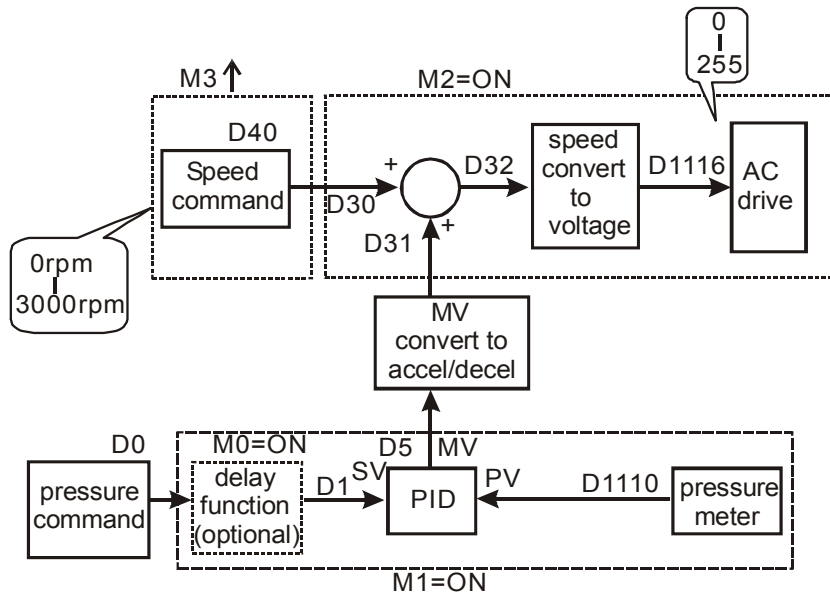
Speed control and pressure control system is controlled separately. (Use block diagram of example 2)

Control destination:

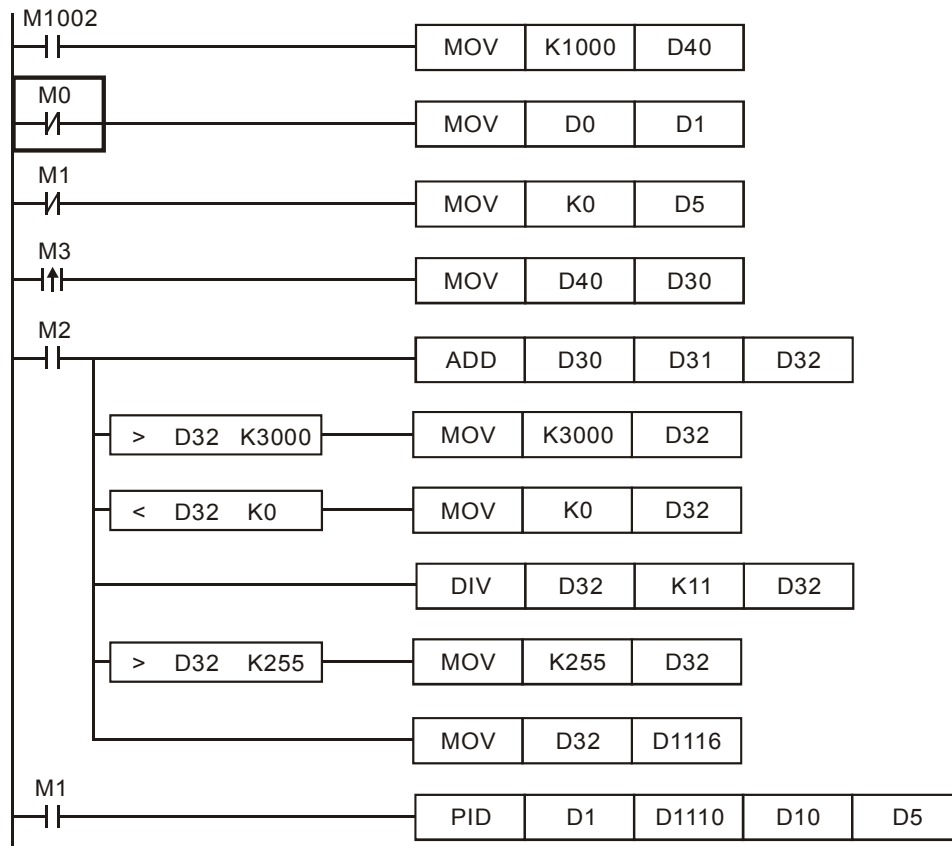
Adding pressure control system (PID instruction) after using an open loop to control speed for a period time to reach the desired pressure control.

Control characteristics:

This architecture should use open loop to reach the speed control and then reach control target by closed loop pressure control due to no relationship between speed and pressure of these two systems. You can add delay instructions to the application to avoid control instruction of pressure control system changing too fast. Control block diagram is shown in the following.



Partial program application is in the following:



Application 3:

Using auto-tuning for temperature control

Control destination:

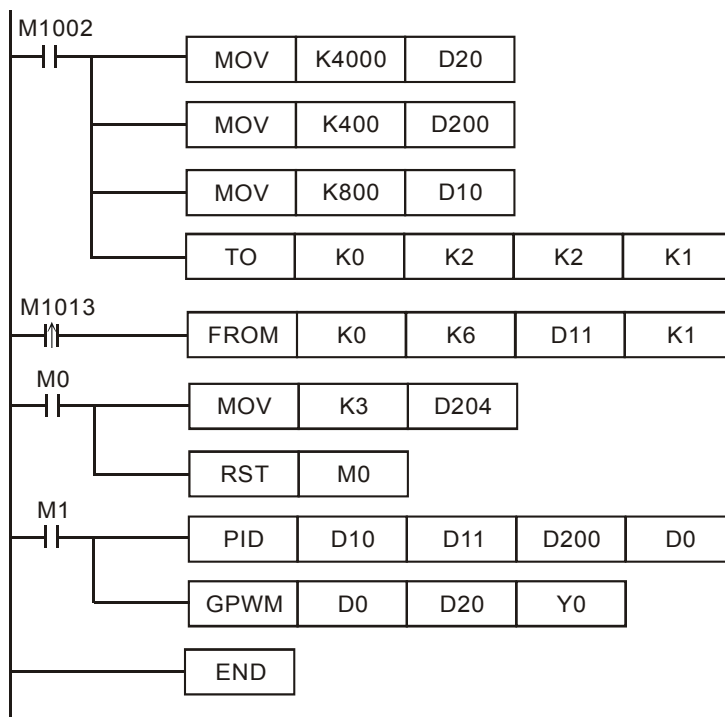
Calculating optimal parameter of PID instruction for temperature control

Control characteristics:

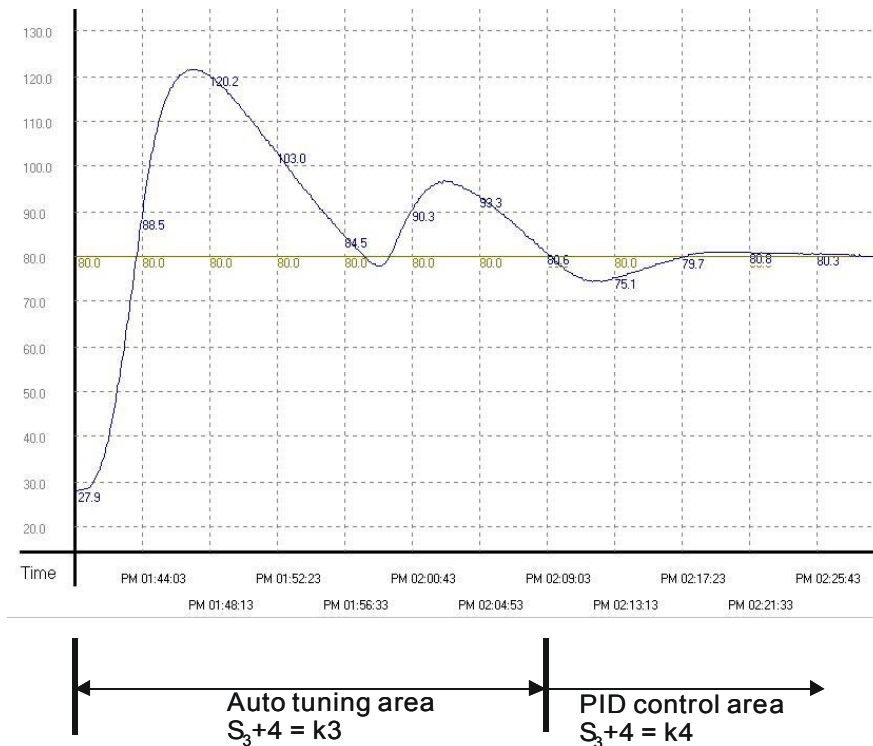
Since general users are not familiar with the environment characteristics of temperature control at the first time, user can use auto-tuning function ($S_3+4=k3$) for initial tuning. After finished tuning, this instruction will auto modify control function to the specific function of temperature control ($S_3+4=k4$).

Control environment: oven

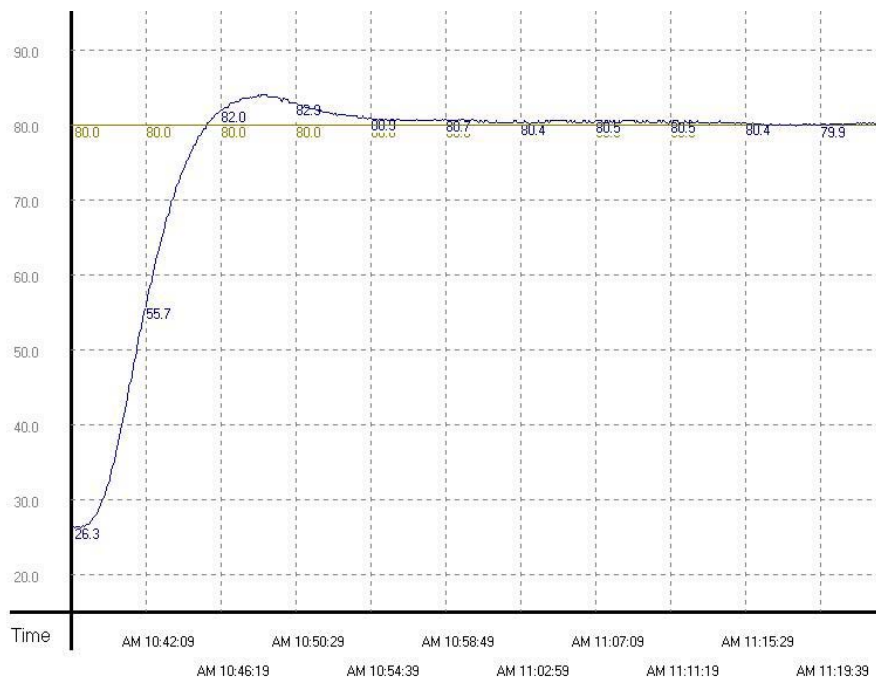
Partial program application is in the following:



The experimental result of auto-tuning is shown as follows: (M0 & M1 set ON)



The experimental result of specific temperature control method after auto-tuning is shown as follows:
(M0 set OFF, M1 set ON)



API	Mnemonic	Operands	Function												Controllers			
89	PLS	S	Rising-edge output												PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	PLS: 3 steps		
	S		*	*															

Operands:

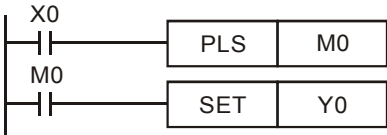
S: Rising pulse output device

Explanations:

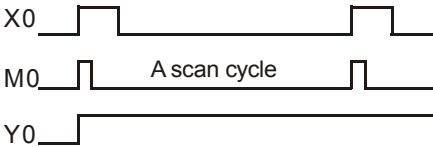
When X0=OFF→ON (rising-edge trigger), PLS command will be executed and M0 will send the pulse of one time which the width is a scan time.

Program Example:

Ladder Diagram:



Timing Diagram:



Command Code:

LD X0 ; Load A contact of X0

PLS M0 ; M0 rising-edge output

LD M0 ; Load the contact A of M0

SET Y0 ; Y0 latched (ON)

Operation:

API	Mnemonic	Operands	Function	Controllers			
90	LDP	<div>S</div>	Rising-edge detection operation	PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	LDP: 3 steps
S	*	*	*	*							*	*				

Operands:

S: The device that is detected switching from OFF to ON

Explanations:

Usage of the LDP command is the same as the LD command, but the motion is different. It is used to reserve present contents and at the same time, saving the detection status of the acquired contact rising-edge into the accumulative register.

Program Example:

Ladder Diagram:



Command Code:


LDP X0
AND X1
OUT Y1

Operation:

; Start X0 rising-edge detection
 ; Series connection A contact of X1
 ; Drive Y1 coil

Points to Note:

1. Please refer to the specification of every model for the operand usage.
2. If specific rising-edge contact state is ON before ELC is power on, rising-edge contact will be True after ELC is power on.

API	Mnemonic	Operands	Function													Controllers				
91	LDF		Falling-edge detection operation													PB	PC	PA	PH	
OP	Type	Bit Devices				Word devices										Program Steps				
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	LDF: 3 steps			
	S	*	*	*	*							*	*							

Operands:

S: The device that is detected switching from ON to OFF

Explanations:

Usage of the LDF command is the same as the LD command, but the motion is different. It is used to reserve present contents and at the same time, saving the detection status of the acquired contact falling-edge into the accumulative register.

Program Example:

Ladder Diagram:




Command Code:

LDF X0
AND X1
OUT Y1

Operation:

; Start X0 falling-edge detection
; Series connection A contact of X1
; Drive Y1 coil

API	Mnemonic	Operands	Function										Controllers			
92	ANDP		Rising-edge series connection										PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ANDP: 3 steps
S	*	*	*	*							*	*				

Operands:

S: The serial connection device that is detected switching from OFF to ON

Explanations:

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

Program Example:

Ladder Diagram:




Command Code:

LD X0
ANDP **X1**
OUT Y1

Operation:

; Load A contact of X0
; X1 rising-edge detection in series connection
; Drive Y1 coil

API	Mnemonic	Operands	Function	Controllers			
93	ANDF		Falling-edge series connection	PB	PC	PA	PH

OP \ Type	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S	*	*	*	*							*	*				ANDF: 3 steps

Operands:

S: The serial connection device that is detected switching from ON to OFF

Explanations:

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

Program Example:

Ladder Diagram:



Command Code:

LD X0

ANDF X1


OUT Y1

Operation:

; Load A contact of X0

; X1 falling-edge detection in series connection

; Drive Y1 coil

API	Mnemonic	Operands	Function										Controllers			
94	ORP		Rising-edge Parallel connection										PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ORP: 3 steps			
S	*	*	*	*							*	*							

Operands:

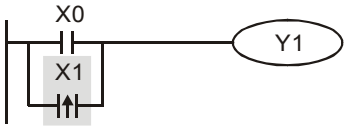
S: The parallel connection device that is detected switching from OFF to ON

Explanations:

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

Program Example:

Ladder Diagram:




Command Code:

LD X0
ORP **X1**
OUT Y1

Operation:

; Load A contact of X0
; X1 rising-edge detection in parallel connection
; Drive Y1 coil

API	Mnemonic	Operands	Function												Controllers			
95	ORF		Falling-edge Parallel connection												PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ORF: 3 steps		
	S	*	*	*	*							*	*						

Operands:

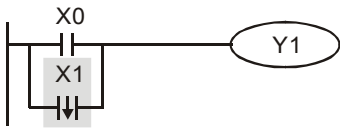
S: The parallel connection device that is detected switching from ON to OFF

Explanations:

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

Program Example:

Ladder Diagram:



Command Code:

LD X0
ORF X1
OUT Y1

Operation:

; Load A contact of X0
; X1 falling-edge detection in parallel connection
; Drive Y1 coil

API	Mnemonic	Operands	Function	Controllers			
96	TMR	(S₁) (S₂)	Timer	PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E		
	S ₁											*					
	S ₂					*								*			

Operands:

S₁: Timer number(T0~T255) **S₂**: Set value (K0~K32,767, PB:D0~D599, PC/PA/PH:D0~D4,999)

Explanations:

When TMR command is executed, the specific coil of timer is ON and timer will start to count. When the setting value of timer is attained (counting value >= setting value), the contact will be as following:

NO(Normally Open) contact	Continuity
NC(Normally Closed) contact	Non-continuity

Program example:

Ladder Diagram:



Command Code:

LD X0
TMR T5 K1000

Operation:

; Load A contact of X0
; T5 timer Setting is K1000

API	Mnemonic	Operands	Function	Controllers			
97	CNT	(S₁) (S₂)	Counter 16-bit	PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	CNT: 5 steps
S ₁												*				
S ₂					*								*			

Operands:

S₁: 16 bit counter number(C0~C199) **S₂**: Set value (K0~K32,767, PB:D0~D599, PC/PA/PH:D0~D4,999)

Explanations:

When the CNT command is executed from OFF→ON, which means that the counter coil is driven, and should thus be added to the counter's value; when the counter achieved specific set value (value of counter = the setting value), motion of the contact is as follows:

NO(Normally Open) contact	Continuity
NC(Normally Closed) contact	Non-continuity

If there is counting pulse input after counting is attained, the contacts and the counting values will be unchanged. To re-count or to conduct the CLEAR motion, please use the RST command.

Program example:

Ladder Diagram:



Command Code:

LD X0

CNT C20 K100

Operation:

; Load A contact of X0

; C20 counter Setting is K100

API	Mnemonic	Operands	Function	Controllers			
97	DCNT	(S ₁) (S ₂)	Counter 32-bit	PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DCNT: 9 steps
	S ₁												*				
S ₂						*								*			

Operands:

S₁: 32 bit counter number(PB: C235~C254, PC/PA/PH: C200~C254)

S₂: Set value (K-2,147,483,648~K2,147,483,647, PB:D0~D599, PC/PA/PH:D0~D4,999)

Explanations:

1. DCNT is the startup command for the 32-bit high-speed counter that is utilized especially in counters C235 to C254.
2. For general addition/subtraction counter C200~C234, the present value will count up (add 1) or count down (subtract 1) according to the flags M1200~M1234 set count mode when command DCNT is from OFF→ON.
3. When specific high-speed counter pulse input of high-speed addition/subtraction counters C235~C254 is from OFF→ON, it will execute counting. If counter trigger input keeps being ON or OFF, the counter value will be unchanged. See chapter 2.7 counter number and function for high-speed pulse input terminals (X0~X11) and counting (count up (add 1) and count down (subtract 1)).
4. When DCNT command is OFF, the counter will stop counting, but the counting values will not be cleared. Users can use RST C2XX command to remove the counting values and the contacts. High-speed addition/subtraction counters C235~C254 can use external specific input point to clear the counting values and the contacts.

Program Example:

Ladder Diagram:



Command Code:

LD M0

DCNT C254 K1000

Operation:

; Load A contact of M0

; C254 counter Setting is K1000

API	Mnemonic	Operands	Function	Controllers			
	INV	-	Inverse operation	PB	PC	PA	PH

OP	Descriptions	Program Steps
N/A	Invert the current result of the internal ELC operations	INV: 1 step

Explanations:

Inverting the operation result and use the new data as an operation result.

Program Example:

Ladder Diagram:



Command Code:

LD X0

INV

OUT Y1

Operation:

; Load A contact of X0

; Inverting the operation result

; Drive Y1 coil

API	Mnemonic	Operands	Function												Controllers			
99	PLF	S	Falling-edge output												PB	PC	PA	PH

Type OP	Bit Devices				Word devices												Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	PLF: 3 steps				
S		*	*																	

Operands:

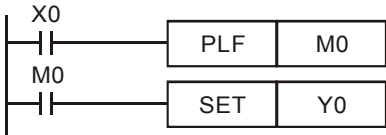
S: Falling pulse output device

Explanations:

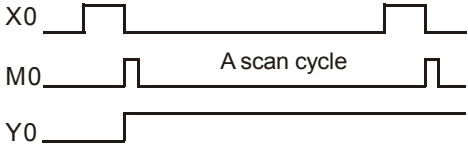
When X0= ON→OFF (falling-edge trigger), PLF command will be executed and M0 will send the pulse of one time which the width is the time for scan one time.

Program Example:

Ladder Diagram:



Timing Diagram:



Command Code:

LD	X0	; Load A contact of X0
PLF	M0	; M0 falling-edge output
LD	M0	; Load the contact A of M0
SET	Y0	; Y0 latched (ON)

Operation:

API	Mnemonic	Operands	Function	Controllers			
100	MODRD	(S₁) (S₂) (n)	Modbus Data Read	PB	PC	PA	PH

OP \ Type	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MODRD: 7 steps
S ₁					*	*							*			
S ₂					*	*							*			
n					*	*							*			

Operands:

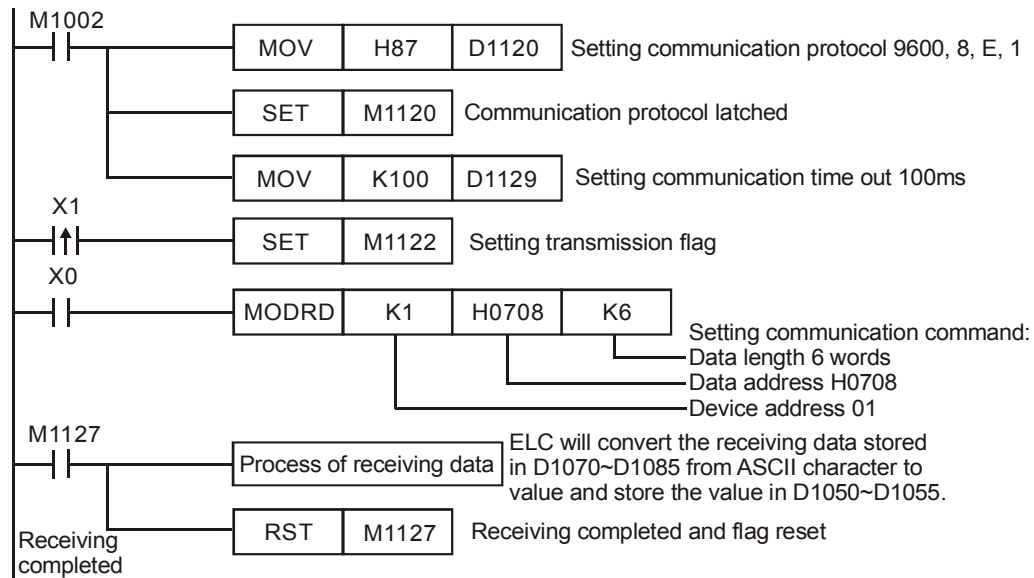
S₁: External device address (K0~K254) **S₂:** Register(s) of external device to read **n:** Number of registers to read ($K1 < n \leq K6$)

Explanations:

- MODRD is a specific instruction for the MODBUS ASCII / RTU mode communication. The MODRD instruction can be used to read MODBUS data from external devices that support MODBUS communications.
- S₂** Register(s) of external device to read. If the register value is illegal, the user will be informed by an error message. The error code will be stored in D1130, while M1141 = ON.
- The response data from external device will be stored in D1070 to D1085. After receives all of the response, ELC will check if the returned data is correct. If there is an error, then M1140 = ON.
- If using ASCII mode, ELC will automatically convert the response data to hex and store in D1050 to D1055. D1050 to D1055 will be invalid if using RTU mode.
- After M1140 or M1141 = ON, a correct data will be transmit to peripheral equipment again. If the response data are all correct, then the flag M1140, M1141 will be clear.
- Refer to RS instruction for more information

Program Example 1:

Communication between ELC and MVX AC drives (ASCII Mode, M1143= OFF)



ELC → MVX, ELC transmits: "01 03 0708 0006 E7"

MVX → ELC, ELC receives: "01 03 0C 0100 1766 0000 0000 0136 0000 3B"

ELC transmits message

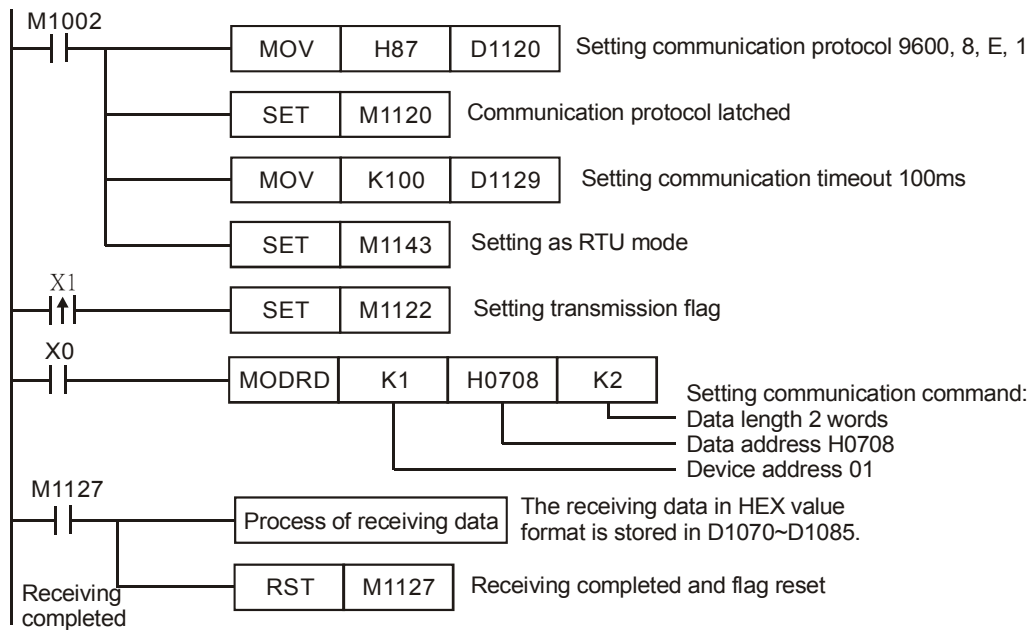
Register	Data		Descriptions	
D1089 low byte	‘0’	30 H	ADR 1	ADR (1,0) is AC drive address
D1089 high byte	‘1’	31 H	ADR 0	
D1090 low byte	‘0’	30 H	CMD 1	CMD (1,0) is instruction code
D1090 high byte	‘3’	33 H	CMD 0	
D1091 low byte	0’	30 H	Starting data address	
D1091 high byte	‘7’	37 H		
D1092 low byte	‘0’	30 H		
D1092 high byte	‘8’	38 H		
D1093 low byte	‘0’	30 H	Number of data (count by word)	
D1093 high byte	‘0’	30 H		
D1094 low byte	‘0’	30 H		
D1094 high byte	‘6’	36 H		
D1095 low byte	‘E’	45 H	LRC CHK 1	LRC CHK (0,1) is error check code
D1095 high byte	‘7’	37 H	LRC CHK 0	

ELC receives response message

Register	Data		Descriptions	
D1070 low	‘0’	30 H	ADR 1	
D1070 high	‘1’	31 H	ADR 0	
D1071 low byte	‘0’	30 H	CMD 1	
D1071 high byte	‘3’	33 H	CMD 0	
D1072 low byte	‘0’	30 H	Number of data (count by byte)	
D1072 high byte	‘C’	43 H		
D1073 low byte	‘0’	30 H	Content of address 0708 H	ELC automatically converts ASCII codes to hex and store the converted value in D1050 = 0100 H
D1073 high byte	‘1’	31 H		
D1074 low byte	‘0’	30 H		
D1074 high byte	‘0’	30 H		
D1075 low byte	‘1’	31 H	Content of address 0709 H	ELC automatically converts ASCII codes to hex and store the converted value in D1051 = 1766 H
D1075 high byte	‘7’	37 H		
D1076 low byte	‘6’	36 H		
D1076 high byte	‘6’	36 H		
D1077 low byte	‘0’	30 H	Content of address 070A H	ELC automatically converts ASCII codes to hex and store the converted value in D1052 = 0000 H
D1077 high byte	‘0’	30 H		
D1078 low byte	‘0’	30 H		
D1078 high byte	‘0’	30 H		
D1079 low byte	‘0’	30 H	Content of address 070B H	ELC automatically converts ASCII codes to hex and store the converted value in D1053 = 0000 H
D1079 high byte	‘0’	30 H		
D1080 low byte	‘0’	30 H		
D1080 high byte	‘0’	30 H		
D1081 low byte	‘0’	30 H	Content of address 070CH	ELC automatically converts ASCII codes to hex and store the converted value in D1054 = 0136 H
D1081 high byte	‘1’	31 H		
D1082 low byte	‘3’	33 H		
D1082 high byte	‘6’	36 H		
D1083 low byte	‘0’	30 H	Content of address 070D H	ELC automatically converts ASCII codes to hex and store the converted value in D1055 = 0000 H
D1083 high byte	‘0’	30 H		
D1084 low byte	‘0’	30 H		
D1084 high byte	‘0’	30 H		
D1085 low byte	‘3’	33 H	LRC CHK 1	
D1085 high byte	‘B’	42 H	LRC CHK 0	

Program Example 2:

Communication between ELC and MVX AC drive (RTU Mode, M1143= ON)



ELC → MVX, ELC transmits: 01 03 0708 0002 44 BD

MVX → ELC, ELC receives: 01 03 04 1770 0000 FE 5C

ELC transmits message

Register	Data	Descriptions
D1089 low byte	01 H	Address
D1090 low byte	03 H	Function
D1091 low byte	07 H	Starting data address
D1092 low byte	08 H	
D1093 low byte	00 H	Number of data (count by word)
D1094 low byte	02 H	
D1095 low byte	44 H	CRC CHK Low
D1096 low byte	BD H	CRC CHK High

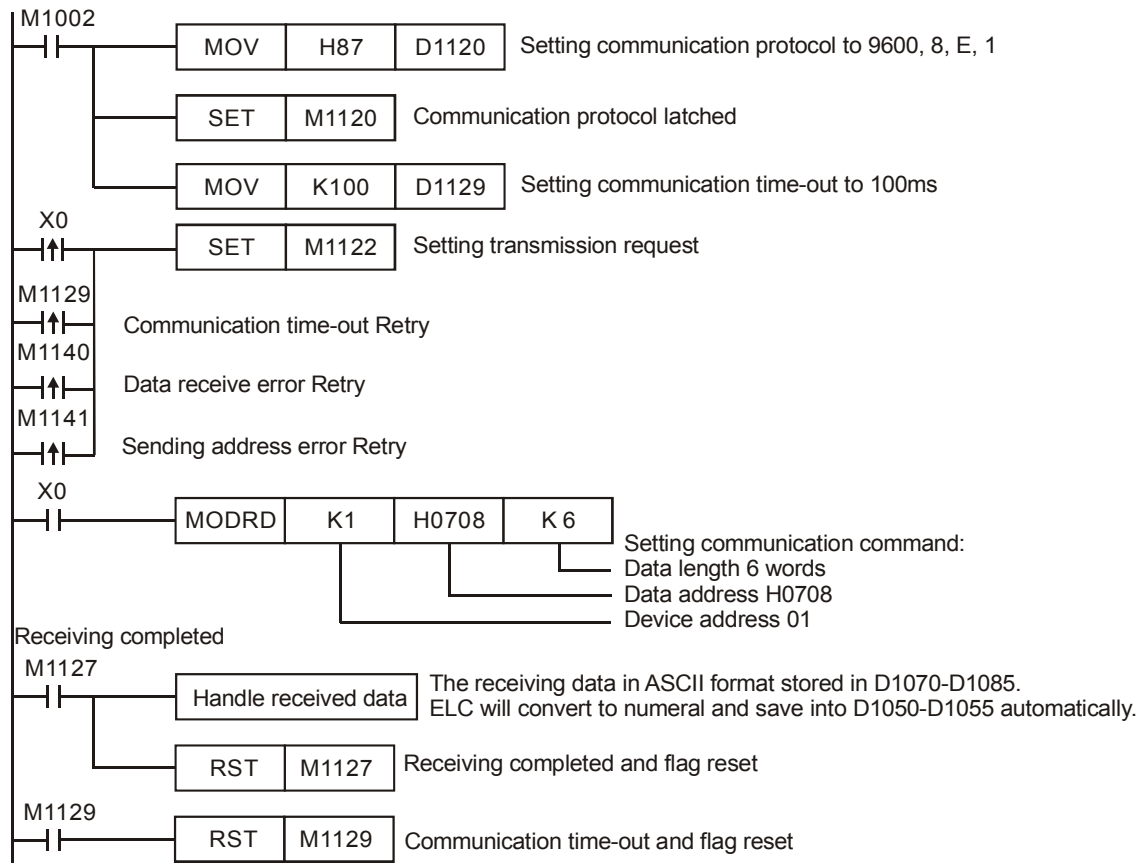
ELC receives response message

Register	Data	Descriptions
D1070 low byte	01 H	Address
D1071 low byte	03 H	Function
D1072 low byte	04 H	Number of data (count by byte)
D1073 low byte	17 H	Content of address 2102 H
D1074 low byte	70 H	
D1075 low byte	00 H	Content of address 2103 H
D1076 low byte	00 H	
D1077 low byte	FE H	CRC CHK Low
D1078 low byte	5C H	CRC CHK High

Program Example 3:

1. ELC connected to an MVX AC drive (ASCII Mode, M1143= OFF). If a communication times-out, retry when the error occurs during receives or sending data.
2. When X0= ON, read data from address H0708 of device 01 (MVX) and save in D1070~D1085 with ASCII format. ELC will auto convert its content to hex to save in D1050~D1055.
3. Flag M1129 = ON when communication times-out, the program will send a request from M1129 to ask M1122 to read again.
4. Flag M1140 = ON when a receive error occurs, the program will send a request from M1140 to ask M1122 to read again.
5. Flag M1141 = ON when a received address error occurs, the program will send a request from M1141 to ask M1122 to read again.

Program diagram



Notes:

1. For detail information of the related flags and special registers, refer to the RS instruction.
2. Rising-edge contact (LDP, ANDP, ORP) and falling-edge contact (LDF, ANDF, ORF) can not be used with MODRD, MODRW (FUNCTION CODE H03) instructions. Otherwise, the data stored in the received registers will be incorrect.

API	Mnemonic	Operands	Function	Controllers			
101	MODWR	(S₁) (S₂) (n)	Modbus Data Write	PB	PC	PA	PH

Type	Bit Devices				Word devices										Program Steps	
OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MODWR: 7 steps
S ₁					*	*							*			
S ₂					*	*							*			
n					*	*							*			

Operands:

S₁: External device address (K0~K254) **S₂:** Register(s) of external device to write to **n:** Number of values to write

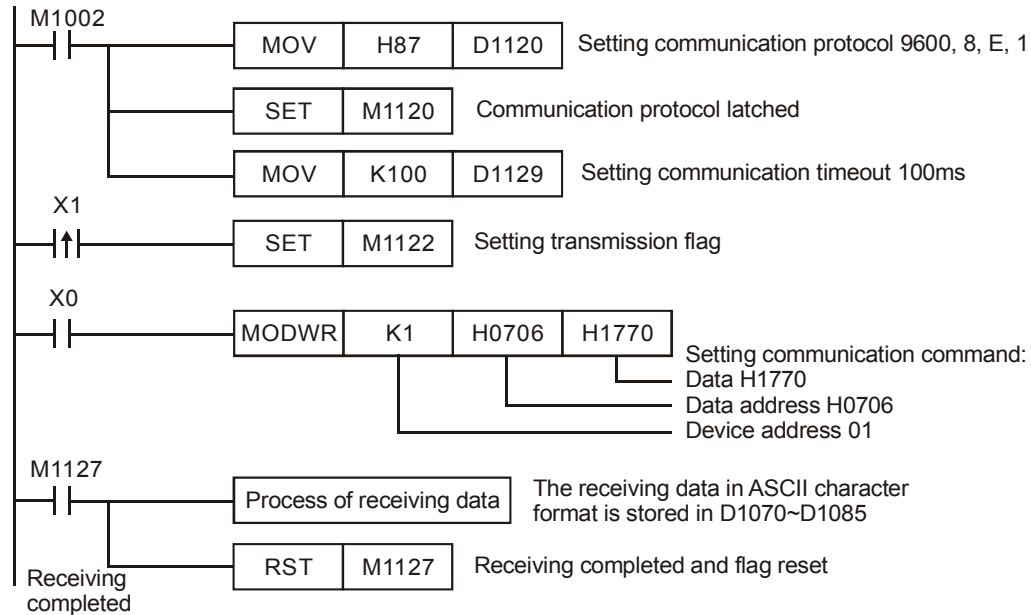
Explanations:

- MODWR is a specific instruction for the MODBUS ASCII / RTU mode communication. The MODWR instruction can be used to write MODBUS data to external devices that support MODBUS communications.
- S₂** is the register to write data to. If the register address setting is not valid, the user will be informed by an error message. The error code will be stored in D1130, while M1141 = ON. For example, 8000H is an illegal register address in the MVX drive, then M1141 = ON, D1130=2. Since the error code is generated by the external device. A user will need to refer to the external device's manual to determine the error. In this case a user would need to refer to the MVX series user manual.
- The response from the external device will be stored in D1070 to D1076. After receives all of the response, ELC will check if there are any data errors. If there is an error, then M1140 = ON.
- After M1140 or M1141 = ON, resend the correct data to the external device again. If the response is correct, then the flag M1140, M1141 will be clear.

Program Example 1:

Communication between ELC and MVX AC drives (ASCII Mode, M1143= OFF)

Program diagram



ELC → MVX, ELC transmits: "01 06 0706 1770 65 "

MVX → ELC, ELC receives: "01 06 0706 1770 65 "

ELC transmits message

Register	Data		Descriptions	
D1089 low	‘0’	30 H	ADR 1	ADR (1,0) is AC drive address
D1089 high	‘1’	31 H	ADR 0	
D1090 low	‘0’	30 H	CMD 1	CMD (1,0) is instruction code
D1090 high	‘6’	36 H	CMD 0	
D1091 low	‘0’	30 H	Data address	
D1091 high	‘7’	37 H		
D1092 low	‘0’	30 H		
D1092 high	‘6’	36 H		
D1093 low	‘1’	31 H	Data contents	
D1093 high	‘7’	37 H		
D1094 low	‘7’	37 H		
D1094 high	‘0’	30 H		
D1095 low	‘6’	36 H	LRC CHK 1	LRC CHK (0,1) is error check code
D1095 high	‘5’	35 H	LRC CHK 0	

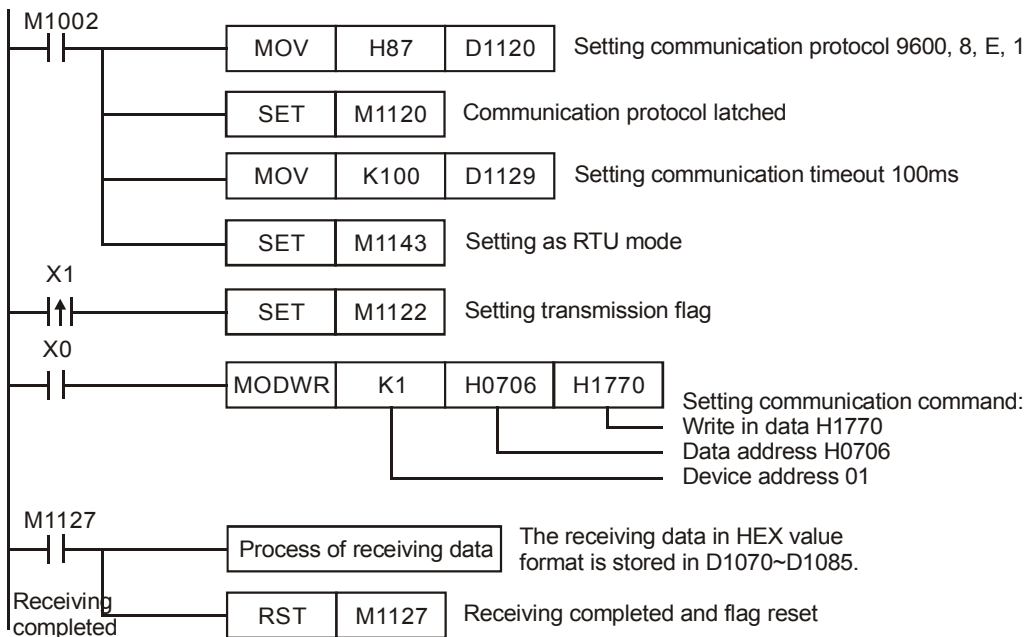
ELC receives response message

Register	Data		Descriptions
D1070 low	'0'	30 H	ADR 1
D1070 high	'1'	31 H	
D1071 low	'0'	30 H	CMD 1
D1071 high	'6'	36 H	
D1072 low	'0'	30 H	Data address
D1072 high	'7'	37 H	
D1073 low	'0'	30 H	
D1073 high	'6'	36 H	
D1074 low	'1'	31 H	Data content
D1074 high	'7'	37 H	
D1075 low	'7'	37 H	
D1075 high	'0'	30 H	
D1076 low	'6'	36 H	LRC CHK 1
D1076 high	'5'	35 H	LRC CHK 0

Program Example 2:

Communication between ELC and MVX AC drives (RTU Mode, M1143= ON)

Program diagram



ELC → MVX, ELC transmits: 01 06 0706 1770 66 AB

MVX → ELC, ELC receives: 01 06 0706 1770 66 AB

ELC transmits message

Register	Data	Descriptions
D1089 low	01 H	Address
D1090 low	06 H	Function
D1091 low	07 H	Data address
D1092 low	06 H	
D1093 low	17 H	Data content
D1094 low	70 H	
D1095 low	66 H	CRC CHK Low
D1096 low	AB H	CRC CHK High

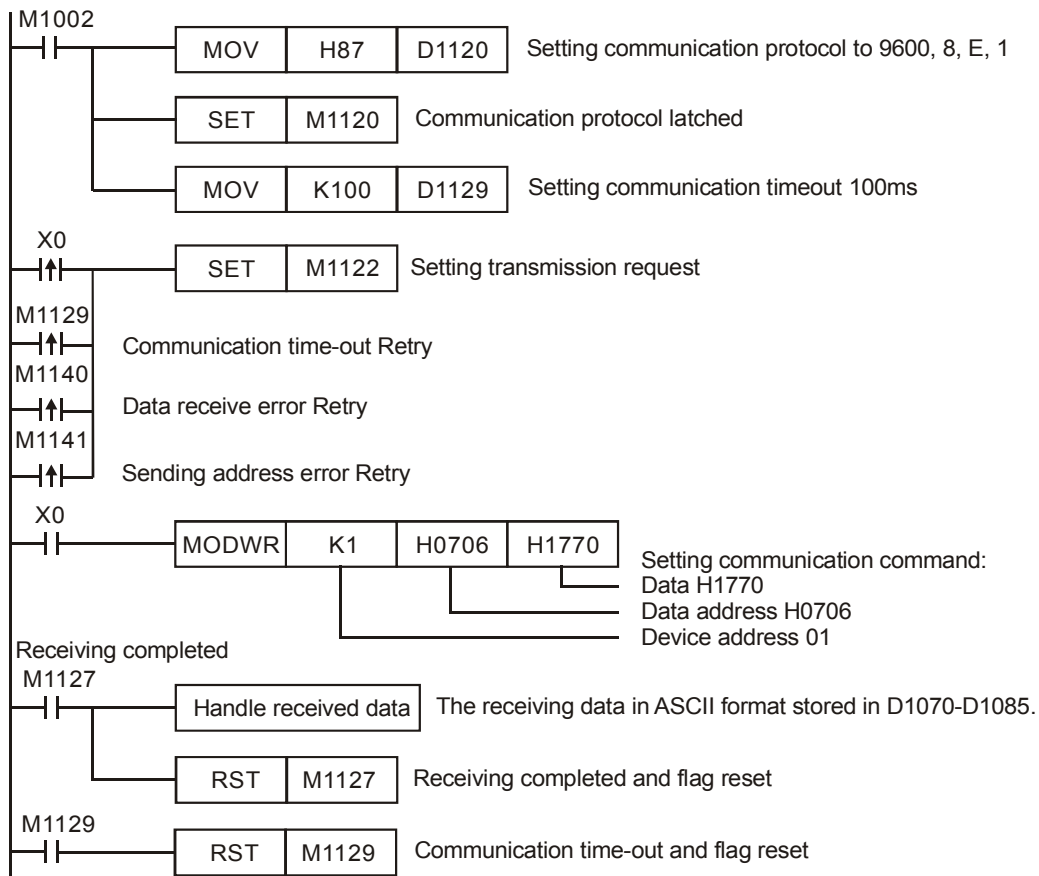
ELC receives response message

Register	Data	Descriptions
D1070 low	01 H	Address
D1071 low	06 H	Function
D1072 low	07 H	Data address
D1073 low	06 H	
D1074 low	17 H	Data content
D1075 low	70 H	
D1076 low	66 H	CRC CHK Low
D1077 low	AB H	CRC CHK High

Program Example 3:

1. ELC connects to MVX AC drive (ASCII Mode, M1143= OFF). When communication times-out, retry when the error occurs during receives or sending.
2. When X0= ON, ELC will write data H1770(K6000) into address H0706 of device 01 (MVX).
3. Flag M1129 = ON when communication times-out and the program will send a request from M1129 to ask M1122 to read again.
4. Flag M1140 = ON when response is a receive error, the program will send a request from M1140 and ask M1122 to read again.
5. Flag M1141 = ON when the response is a received address error, the program will send a request from M1141 to ask M1122 to read again.

Program diagram

**Points to note:**

1. For detail information of the related flags and special registers, refer to the RS instruction.
2. If using rising-edge (LDP, ANDP, ORP) or falling-edge (LDF, ANDF, ORF) before MODWR and MODRW (Function Code H06 and H10) instructions, the program should start the transmission request M1122 to execute properly.

API	Mnemonic			Operands			Function								Controllers			
107		LRC	P	(S)	(n)	(D)	LRC Generator								PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	LRC, LRCP: 7 steps		
S													*					
n					*	*							*					
D													*					

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: Starting device for the checksum operation (ASCII mode) **n:** Number of values to perform the operation on ($n=K1\sim K256$) **D:** Destination for storing the operation result

Explanations:

- n:** **n** must be even. If **n** is out of range, an error will occurs and the instruction will not be executed. At this time, M1067 and M1068 = ON and error code H'0E1A will be recorded in D1067.
- 16-bit conversion mode: When M1161= OFF, hexadecimal data that starts from the source **S** will be divided into upper 8-bit and lower 8-bit values and perform the checksum operation on **n** numbers. Then, store the result into upper and lower 8-bit of **D**.
- 8-bit conversion mode: When M1161= ON, divide hexadecimal data that start from the source device **S** into upper 8-bit (invalid data) and lower 8-bit and perform the checksum operation on **n** numbers. Then, store the result into lower 8-bit of **D** (upper 8-bit of **D** all be zero (0)).
- Flag: M1161 8/16-bit mode

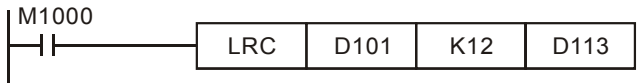
Program Example :

ASCII communication mode: Data stored as following:

Register	Data		Descriptions	
D100 low byte	‘:’	3A H	STX	
D101 low byte	‘0’	30 H	ADR 1 ADR 0	ADR (1,0) is AC drive address
D102 low byte	‘1’	31 H		
D103 low byte	‘0’	30 H	CMD 1 CMD 0	CMD (1,0) is instruction code
D104 low byte	‘3’	33 H		
D105 low byte	‘0’	30 H	Starting data address	
D106 low byte	‘7’	37 H		
D107 low byte	‘0’	30 H		
D108 low byte	‘8’	38 H		
D109 low byte	‘0’	30 H	Number of data (count by word)	

Register	Data		Descriptions	
D110 low byte	'0'	30 H		
D111 low byte	'0'	30 H		
D112 low byte	'6'	36 H		
D113 low byte	'E'	45 H	LRC CHK 0	LRC CHK (0,1) error check code
D114 low byte	'7'	37 H	LRC CHK 1	
D115 low byte	CR	D H	END	
D116 low byte	LF	A H		

The LRC CHK (0,1) above is error check code and it can be calculated by the LRC instruction (8-bit Mode, M1161= ON).



LRC check: 01 H + 03 H + 07 H + 08 H + 00 H + 06 H = 19 H, then take the complement of 2, E7 H. At that time, 'E'(45 H) is stored in the lower 8-bit of D113 and '7' (37 H) is stored in the lower 8-bits of D114.

API	Mnemonic			Operands			Function										Controllers										
108		CRC	P	S	n	D	CRC Generator										PB	PC	PA	PH							
Type	Bit Devices				Word devices										Program Steps												
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	CRC, CRCP: 7 steps											
OP													*														
S													*														
n					*	*							*														
D													*														
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: Starting device for the checksum operation (RTU mode) **n:** Number of values to perform the operation on ($n=K1\sim K256$) **D:** Destination for storing the operation result

Explanations:

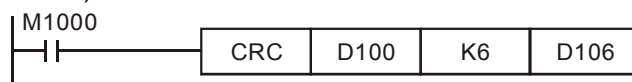
1. If **n** is out of range, an error will occurred and the instruction won't be executed. At this time, M1067 and M1068 = ON and error code H'0E1A will be recorded in D1067.
2. 8-bit conversion mode: When M1161= ON (16-bit mode, M1161=OFF) divide hexadecimal data that starts from the source **S** into high byte (invalid data) and low byte and have the CRC operation performed on **n** numbers and store the result into low byte of **D** (upper 8-bit of **D** all be zero (0)).

Program Example:

RTU communication mode: Data stored as following:

Register	Data	Descriptions
D100 low byte	01 H	Address
D101 low byte	06 H	Function
D102 low byte	07 H	Data address
D103 low byte	06 H	
D104 low byte	17 H	Data content
D105 low byte	70 H	
D106 low byte	66 H	CRC CHK 0
D107 low byte	AB H	CRC CHK 1

The CRC CHK (0,1) above is error check code and it can be calculated by CRC instruction (8-bit Mode, M1161= ON).



CRC check: At the time, 66 H is stored in the lower 8-bit of D106 and AB H is stored in the lower 8-bit of D107.

API	Mnemonic			Operands			Function								Controllers			
110	D	ECMP	P	S₁	S₂	D	Floating Point Compare								PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DECMP, DECMPP: 13 steps		
	S ₁				*	*							*					
	S ₂				*	*							*					
	D		*	*	*													

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

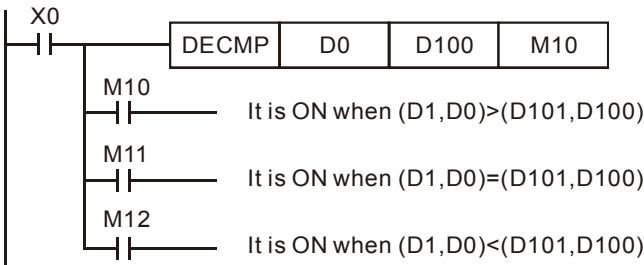
S₁: 1st comparison value **S₂**: 2nd comparison value **D**: Destination result, 3 continuous devices used (occupies 3 continuous devices)

Explanations:

1. The data of **S₁** is compared to the data of **S₂** and the result ($>$, $=$, $<$) is showed by three bit devices in **D**.
2. If the source operand **S₁** or **S₂** is indicated as constant K or H, the integer value will automatically be converted to binary floating point to compare.

Program Example:

1. If the specified device is M10, M10~M12 will automatically be used.
2. When X0= ON, one of M10~M12 = ON. When X0= OFF, DECMP is not executed, M10~M12 will retain their previous state before X0= OFF.
3. Connect M10~M12 in series or parallel to use the result of \geq , \leq , \neq .
4. Use RST or ZRST instruction to reset the result.



API	Mnemonic			Operands				Function				Controllers			
111	D	EZCP	P	S₁	S₂	S	D	Floating Zone Compare				PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DEZCP, DEZCPP: 17 steps		
S ₁					*	*							*					
S ₂					*	*							*					
S					*	*							*					
D		*	*	*														

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

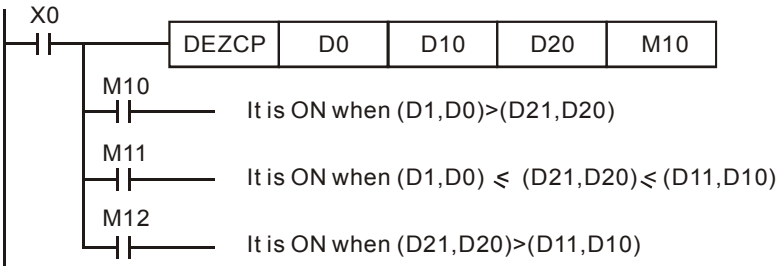
S₁: Lower limit of zone comparison **S₂**: Upper limit of zone comparison **S**: Comparison value **D**: Comparison result, 3 continuous devices used

Explanations:

- 1. The data of **S** is compared to the data range of **S₁~ S₂** and the result (>, =, <) is displayed by three bit devices in **D**.
- 2. If the source operand **S₁** or **S₂** is indicated as constant K or H, the integer value will automatically be converted to binary floating point to compare.
- 3. Operand **S₁** should be smaller than operand **S₂**, when **S₁>S₂**, **S₁** will be used as upper and lower limit for the comparison.

Program Example:

- 1. If the specified device is M10, M10~M12 will automatically be used.
- 2. When X0= ON, one of M10~M12 = ON. When X0= OFF, DEZCP instruction is not executed, M10~M12 will retain their previous state before X0= OFF.
- 3. Use RST or ZRST instruction to reset the result.



API	Mnemonic			Operands		Function										Controllers			
116	D	RAD	P	S	D	Degree → Radian										PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DRAD, DRADP: 9 steps			
S					*	*							*						
D													*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: source (degree) **D:** Destination result (radian)

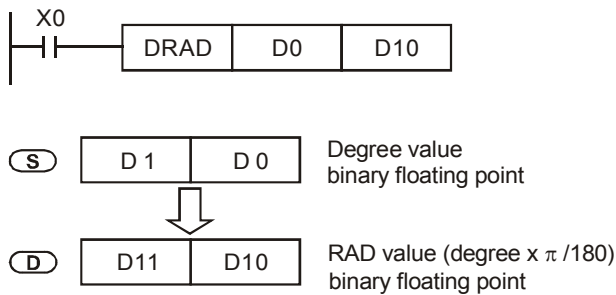
Explanation:

- Using following function to convert degree to radian:

$$\text{Radian} = \text{degree} \times (\pi / 180)$$
- Flags: M1020 Zero flag, M1021 Borrow flag, M1022 Carry flag
 If absolute value of the results is larger than max. floating point, carry flag M1022= ON.
 If absolute value of the results is less than min. floating point, borrow flag M1021= ON.
 If the conversion result is 0, zero flag M1020= ON.

Program Example:

When X0= ON, convert degree value of the binary floating point in (D1, D0) to radian and save in (D11, D10) and the result is binary floating point.



API	Mnemonic			Operands		Function										Controllers			
117	D	DEG	P	(S)	(D)	Radian → Degree										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DDEG, DDEGP: 9 steps		
	S					*	*							*					
D													*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

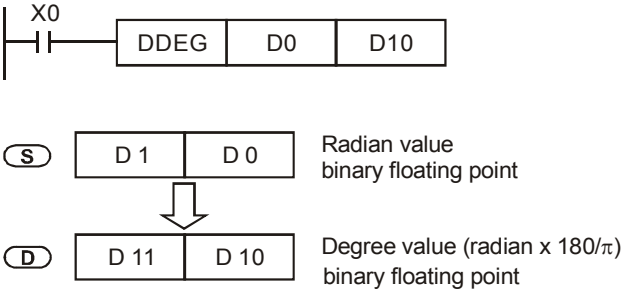
S: data source (radian) D: Destination result (degree)

Explanation

1. Using following function to convert radian to degree:
Degree = Radian × (180/π)
2. Flags: M1020 Zero flag, M1021 Borrow flag and M1022 Carry flag.
If absolute value of the result is larger than max. floating point, carry flag M1022= ON.
If absolute value of the result is less than min. floating point, borrow flag M1021= ON.
If the conversion result is 0, zero flag M1020= ON.

Program Example:

When X0= ON, convert the degree value of the binary floating point in (D1, D0) to radian and save in (D11, D10) and the result is binary floating point.



API	Mnemonic			Operands		Function										Controllers			
118	D	EBCD	P	<div>S</div>	<div>D</div>	Floating to Scientific Conversion										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DEBCD, DEBCDP: 9 steps		
	S													*					
D														*					

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

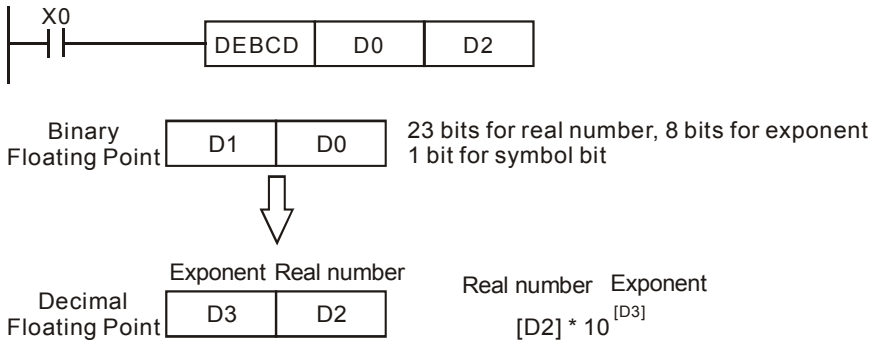
S: Data source **D:** Destination result

Explanation

- Convert the binary floating point value at the register specified by **S** to decimal floating point value and store in the register specified by **D**.
- ELC floating point is operated by the binary floating point format. DEBCD instruction is the specific instruction used to convert binary floating point to decimal floating point.
- Flag: M1020 Zero flag, M1021 Borrow flag, M1022 Carry flag
 If absolute value of the result is larger than max. floating point, carry flag M1022= ON.
 If absolute value of the result is less than min. floating point, borrow flag M1021= ON.
 If the conversion result is 0, zero flag M1020= ON.

Program Example:

When X0= ON, the binary floating point value in D1, D0 will be converted to decimal floating point stored in D3, D2.



API	Mnemonic			Operands		Function										Controllers						
119	D	EBIN	P	S	D	Scientific to Float Conversion										PB	PC	PA	PH			
Type OP	Bit Devices				Word devices										Program Steps							
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DEBIN, DEBINP: 9 steps						
													*									
													*									
											PULSE				16-bit				32-bit			
											PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

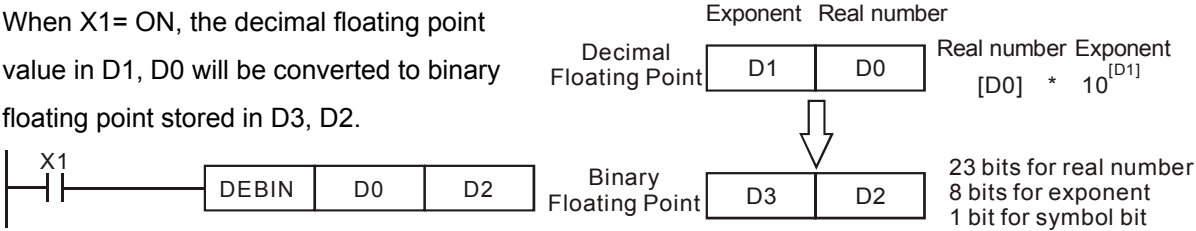
S: Data source D: Destination result

Explanation:

- Convert a decimal floating point value at the register specified by **S** to a binary floating point value and store in the register specified by **D**.
- For example, **S** =1234, **S** +1= 3 will become **S** =1.234 x 10⁶
- D** must be a binary floating point format. **S** and **S** +1 represent the real number and exponent of the floating point number respectively.

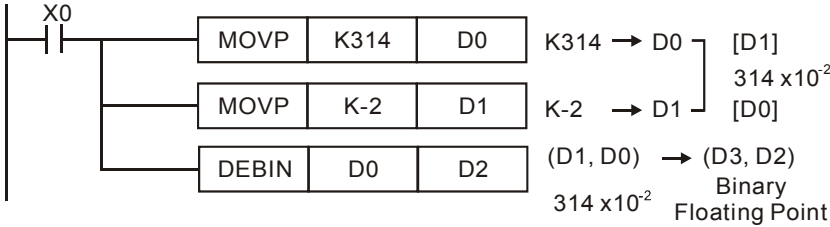
Program Example 1:

When X1= ON, the decimal floating point value in D1, D0 will be converted to binary floating point stored in D3, D2.



Program Example 2:

- Before performing this floating point operation, you must use the FLT instruction to convert BIN integer to binary floating point. The source data should be a BIN integer. However, DEBIN instruction can be used to convert floating point value to binary floating point value.
- When X0= ON, move K314 to D0 and move K-2 to D1 to generate decimal floating point format (3.14 = 314 x 10⁻²).



API	Mnemonic			Operands			Function								Controllers			
120	D	EADD	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Floating Point Addition								<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

<div>Type OP</div>	Bit Devices				Word devices										Program Steps				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E					F
	S ₁					*	*						*						
	S ₂					*	*						*						
	D												*						

PULSE				16-bit				32-bit			
<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

Operands:

S₁: Augend S₂: Addend D: Addition result

Explanations:

1. **S₁ + S₂ = D.** The floating point value in the register specified by **S₁** and **S₂** are added and the result is stored in the register specified by **D**.
2. If the source operand **S₁** or **S₂** is indicated as constant K or H, the integer value will automatically be converted to binary floating point to perform the addition operation.
3. **S₁** and **S₂** can specify the same register number (the same device can be used for **S₁** and **S₂**).
4. This instruction works best when used as the pulse instruction.
5. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag)
 If absolute value of the result is larger than max. floating point, carry flag M1022= ON.
 If absolute value of the result is less than min. floating point, borrow flag M1021= ON.
 If the conversion result is 0, zero flag M1020= ON.

Program Example 1:

When X0= ON, add the binary floating point value of (D1, D0) and binary floating point value of (D3, D2) and store the result in (D11, D10).



Program Example 2:

When X2= ON, add the binary floating point value of (D11, D10) and K1234 (automatically converted to binary floating point) and store the result in (D21, D20).



API	Mnemonic			Operands			Function								Controllers			
121	D	ESUB	P	(S ₁)	(S ₂)	(D)	Floating Point Subtraction								PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DESUB, DESUBP: 13 steps		
S ₁					*	*							*					
S ₂					*	*							*					
D													*					

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

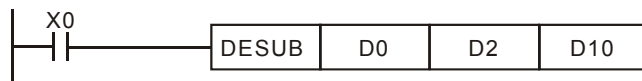
S₁: Minuend S₂: Subtrahend D: Subtraction result

Explanation:

1. **S₁ – S₂ = D.** The floating point value in the register specified by **S₂** is subtracted from the floating point value in the register specified by **S₁** and the result is stored in the register specified by **D**. All data will be operated in floating point format and the result will be also stored in floating point format.
2. If the source operand **S₁** or **S₂** is constant K or H, the integer value will automatically be converted to binary floating point to perform the subtraction operation.
3. **S₁** and **S₂** can specify the same register number (the same device can be used as **S₁** and **S₂**). If in this case and on the continuous execution of the DESUB instruction, the data in the register will be subtracted one time in every scan program during the cycle when the condition contact = ON. Therefore, the pulse instruction (DESUBP) is generally used.
4. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag)
If absolute value of the result is larger than max. floating point, carry flag M1022= ON.
If absolute value of the result is less than min. floating point, borrow flag M1021= ON.
If the conversion result is 0, zero flag M1020= ON.

Program Example:

When X0= ON, binary floating point value of (D3, D2) is subtracted from binary floating point value of (D1, D0) and the result is stored in (D11, D10).



API	Mnemonic			Operands			Function										Controllers			
122	D	EMUL	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Floating Point Multiplication										PB	PC	PA	PH

Type OP	Bit Devices				Word devices												Program Steps DEMUL, DEMULP: 13 steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F					
S ₁					*	*							*							
S ₂					*	*							*							
D													*							

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

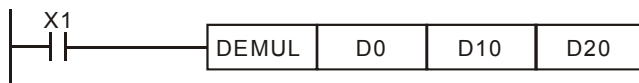
S₁: Multiplicand **S₂**: Multiplier **D**: Multiplication result

Explanations:

- S₁ × S₂ = D.** The floating point value in the register specified by **S₁** is multiplied with the floating point value in the register specified by **S₂** and the result is stored in the register specified by **D**. All data will be operated in floating point format and the result will be also stored in floating point format.
- If the source operand **S₁** or **S₂** is indicated as constant K or H, the integer value will automatically be converted to binary floating point to perform the multiplication operation.
- S₁** and **S₂** can specify the same register number (the same device can be used as **S₁** and **S₂**). If in this case and on the continuous execution of the DEMUL instruction, the data in the register will be multiplied one time in every scan program during the cycle when the condition contact = ON. Therefore, the pulse instruction (DEMULP) is generally used.
- Flags: M1020 Zero flag, M1021 Borrow flag and M1022 Carry flag
If absolute value of the result is larger than max. floating point, carry flag M1022= ON.
If absolute value of the result is less than min. floating point, borrow flag M1021= ON.
If the conversion result is 0, zero flag M1020= ON.

Program Example:

When X1= ON, binary floating point value of (D1, D0) is multiplied with binary floating point (D11, D10) and the result is stored in (D21, D20).



API	Mnemonic			Operands			Function								Controllers												
123	D	EDIV	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Floating Point Division								<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>									
<div>Type</div> <div>OP</div>	Bit Devices				Word devices										Program Steps												
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DEADD, DEADDP: 13 steps											
	S ₁				*	*							*														
	S ₂				*	*							*														
	D												*														
																PULSE				16-bit				32-bit			
																<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

Operands:

S₁: Dividend **S₂:** Divisor **D:** Quotient and Remainder

Explanation:

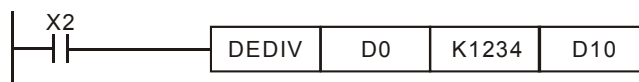
- S₁ ÷ S₂ = D.** The floating point value in the register specified by **S₁** is divided by the floating point value in the register specified by **S₂** and the result is stored in the register specified by **D**. All data will be operated in floating point format and the result will be also stored in floating point format.
- If the source operand **S₁** or **S₂** is indicated as constant K or H, the integer value will automatically be converted to binary floating point to perform the division operation.
- If **S₂** is 0 (zero), the operation will fail and will result in an “operand error”, then this instruction will not be executed.
- Flags: M1020 Zero flag, M1021 Borrow flag and M1022 Carry flag
If absolute value of the result is larger than max. floating point, carry flag M1022= ON.
If absolute value of the result is less than min. floating point, borrow flag M1021= ON.
If the conversion result is 0, zero flag M1020= ON.

Program Example 1:

When X1= ON, binary floating point value of (D1, D0) is divided by binary floating point (D11, D10) and the quotient and remainder is stored in (D21, D20).

**Program Example 2:**

When X2= ON, binary floating point value of (D1, D0) is divided by K1234 (automatically converted to binary floating point) and the result is stored in (D11, D10).



API	Mnemonic			Operands		Function										Controllers			
124	D	EXP	P	S D		Float Exponent Operation										PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps DEXP, DEXPP: 9 steps				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E					
S					*	*							*						
D													*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

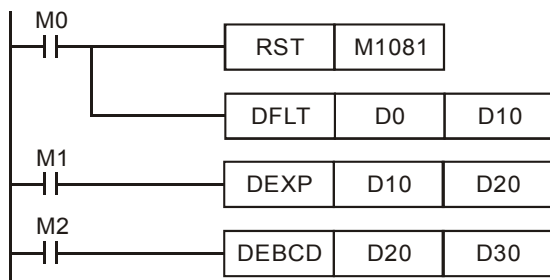
S: operand source device **D:** operand result device

Explanations:

- For example, the base $e = 2.71828$ and exponent is **S**:
- $EXP^{[S+1, S]} = [D+1, D]$
- No matter positive or negative value are valid for **S**. Specific register D needs to use 32-bit format and floating point for operating. Therefore, **S** needs to convert to floating point.
- When operand $D = e^S$, $e = 2.71828$ and **S** is specific source data.
- Error flags: M1067 and M1068, read D1067 and D1068.
- Flags: M1020 Zero flag, M1021 Borrow flag and M1022 Carry flag.
If absolute value of the result is larger than max. floating point, carry flag M1022= ON.
If absolute value of the result is less than min. floating point, borrow flag M1021= ON.
If the conversion result is 0, zero flag M1020= ON.

Program Example:

- When M0= ON, convert (D0, D1) to binary floating point and save in register (D10, D11).
- When M1= ON, use (D10, D11) as the exponent to perform exponent operation. The value is binary floating point and save in register (D20, D21).
- When M2= ON, convert (D20, D21) binary floating point to decimal floating point and save in register (D30, D31). (at this time, D31 means D30 to the power of 10)



API	Mnemonic			Operands		Function										Controllers			
125	D	LN	P	S	D	Float Natural Logarithm Operation										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DLN, DLNP: 9 steps		
	S					*	*							*					
D														*					

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

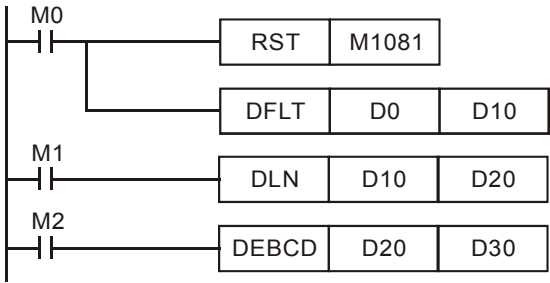
S: operand source device D: operand result device

Explanations:

- Perform natural logarithm operation to operand **S**:
 $LN[S + 1, S] = [D + 1, D]$
- Only a positive number is valid for **S**. Specific register D needs to use 32-bit format and floating point for operating. Therefore, **S** needs to be converted to floating point.
- $e^D = S$, operand **D** = $\ln S$ and **S** is specific source data.
- Flags: M1020 Zero flag, M1021 Borrow flag and M1022 Carry flag.
If absolute value of the result is larger than max. floating point, carry flag M1022 = ON.
If absolute value of the result is less than min. floating point, borrow flag M1021 = ON.
If the conversion result is 0, zero flag M1020 = ON.

Program Example:

- When M0 = ON, convert (D0, D1) to binary floating point and save in register (D10, D11).
- When M1 = ON, use (D10, D11) to be real number to perform natural logarithm operation. The value is binary floating point and save in register (D20, D21).
- When M2 = ON, convert (D20, D21) binary floating point to decimal floating point and save in register (D30, D31). (at this time, D31 means D30 to the power of 10)



API	Mnemonic			Operands			Function										Controllers			
126	D	LOG	P	(S ₁)	(S ₂)	(D)	Float Logarithm Operation										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F				
	S ₁					*	*						*			DLOG, DLOGP: 13 steps			
	S ₂					*	*						*						
	D												*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

API	Mnemonic			Operands		Function										Controllers			
127	D	ESQR	P	(S) (D)		Float point Square Root										PB	PC	PA	PH
Type OP	Bit Devices				Word devices											Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DESQR, DESQRP: 9 steps			
					*	*							*						
													*						
<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>																			
PULSE												16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH								

Operands:

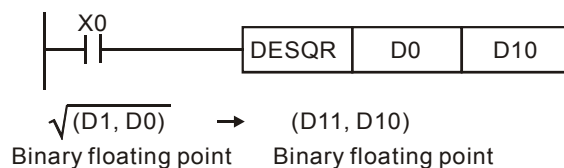
S: Source device **D :** Destination device to store the result

Explanations:

1. This instruction performs a square root operation on the floating point value of source **S** and stores the result at the destination **D**. All data will be operated in floating point format and the result will be also stored in floating point format.
2. If the source **S** is a constant K or H, the integer value will automatically be converted to binary floating point to perform the addition operation.
3. If operation result of **D** is 0 (zero), the Zero flag M1020= ON.
4. **S** can only be a positive value. Performing any square root operation on a negative value will result in an “operation error” and this instruction will not be executed. M1067 and M1068 = ON and error code “0E1B” will be recorded in D1067.
5. Flags: M1020 (Zero flag), M1067 (Program execution error)

Program Example 1:

When X0= ON, the square root of binary floating point (D1, D0) is stored in the register specified by (D11, D10) after the operation of square root.

**Program Example 2**

When X2= ON, the square root of K1234 (automatically converted to binary floating point) is stored in (D11, D10).



API	Mnemonic			Operands			Function										Controllers			
128	D	POW	P	S₁	S₂	D	Float point Power Operation										PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps DPOW, DPOWP: 13 steps						
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E						F	
	S ₁					*	*							*							
	S ₂					*	*							*							
	D													*							

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

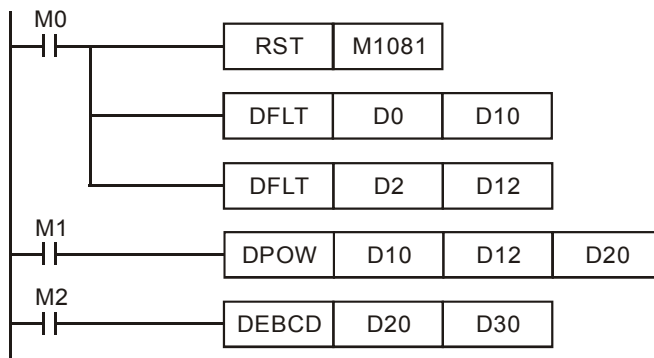
S₁: base device. **S₂**: exponent. **D**: result

Explanations:

- Perform power operation to binary floating point **S₁** and **S₂** and save the result to **D**.
 $POW [S_1+1, S_1] [S_2+1, S_2] = D$
- Only a positive number is valid for **S₁**. Specific register **D** needs to use 32-bit format and floating point for operating. Therefore, **S₁** and **S₂** need to convert to floating point.
- Error flags: M1067 and M1068, read D1067 and D1068.
 If absolute value of the result is larger than max. floating point, carry flag M1022= ON.
 If absolute value of the ion result is less than min. floating point, borrow flag M1021= ON.
 If the conversion result is 0, zero flag M1020= ON.

Program Example:

- When M0= ON, convert (D0, D1) and (D2, D3) to a binary floating point and save in register (D10, D11) and (D12, D13) individually.
- When M1= ON, use (D10, D11) and (D12, D13) binary floating point 32-bit registers to perform power operation and save the result in 32-bit register (D20, D21).
- When M2= ON, convert (D20, D21) binary floating point 32-bit registers to decimal floating point and save in register (D30, D31). (at this time, D31 means D30 to the power of 10)



API	Mnemonic			Operands		Function										Controllers			
129	D	INT	P	(S)	(D)	Floating Point to Integer										PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	INT, INTP: 5 steps DINT, DINTP: 9 steps			
S													*						
D													*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

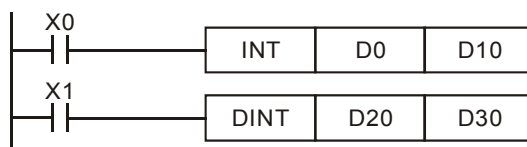
S: Source device **D:** Destination device to store the result

Explanations:

- The binary floating point value of the register **S** is converted to BIN integer and stored in the register **D**. The decimal of BIN integer will be discarded.
- This instruction is the opposite of the API 49 (FLT) instruction.
- Flags: M1020 (Zero flag), M1021 (Borrow flag), M1022 (Carry flag)
If operation result of D is 0 (zero), the Zero flag M1020= ON.
If there is any decimal discarded, the Borrow flag M1021= ON.
If the result exceeds the following range (an overflow occurs), the Carry flag M1022= ON.

Program Example:

- When X0= ON, the binary floating point value of (D1, D0) will be converted to BIN integer and the result is stored in (D10). The decimal of BIN integer will be discarded.
- When X1= ON, the binary floating point value of (D21, D20) will be converted to BIN integer and the result is stored in (D31, D30). The decimal of BIN integer will be discarded.



API	Mnemonic			Operands		Function										Controllers			
130	D	SIN	P	<div>S</div>	<div>D</div>	Floating point Sine Operation										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

Type OP	Bit Devices				Word devices										Program Steps				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DSIN, DSINP: 9 steps			
					*	*							*						
													*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

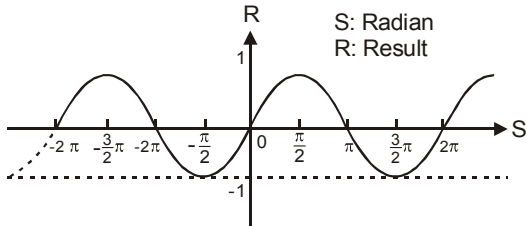
Operands:

S: Specified RAD value ($0^\circ \leq S < 360^\circ$) **D:** Area where calculated result is stored

Explanations:

- Source designated by **S** can be radian or angle by flag M1018.
- When M1018= OFF, it is set to radian mode. $RAD = angle \times \pi / 180$.
- When M1018= ON, it is set to angle mode. Angle range: $0^\circ \leq angle < 360^\circ$.
- The SIN value of an angle data specified by **S** is calculated and the calculated result is stored in the register specified by **D**.
- Flag: M1018 flag for radian/angle.

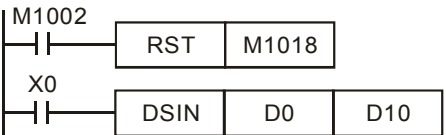
Following shows the relation between radian and result:

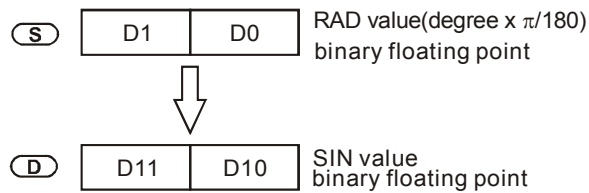


- If result of D is 0, the Zero flag M1020=ON.

Program Example 1:

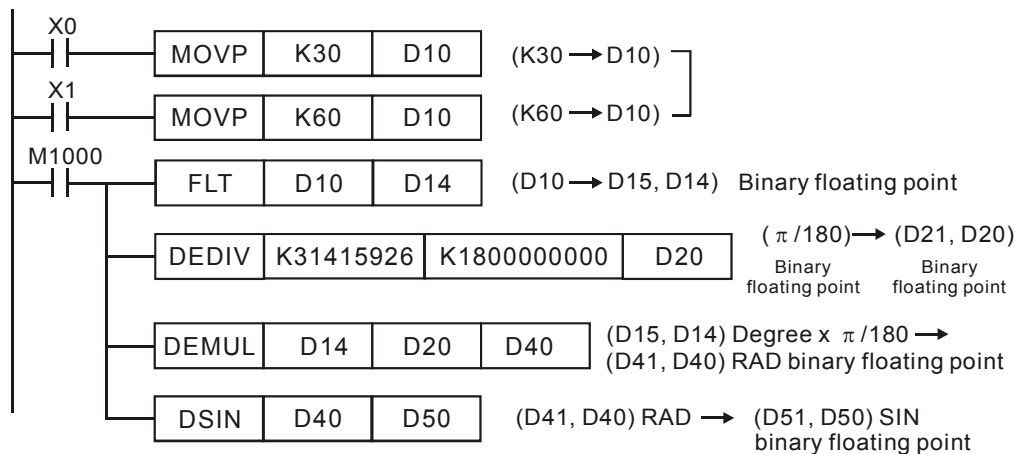
When M1018= OFF, is radian mode. When X0= ON, specify RAD value (D1, D0). Calculate SIN value of angle and store the result in (D11, D10). The result stored in (D11, D10) are all in binary floating point format.





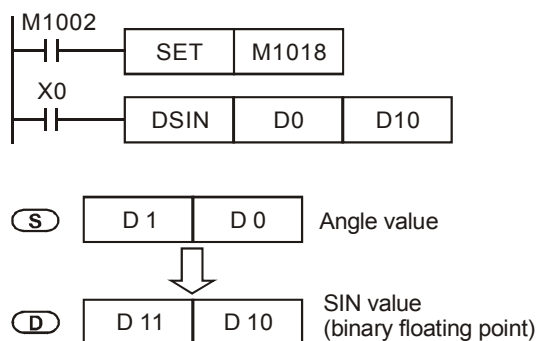
Program Example 2:

When M1018= OFF, is radian mode. Select angle from inputs X0 and X1 and convert it to RAD value to calculate SIN value.



Program Example 3:

When M1018= ON, is angle mode. When X0= ON, it designates angle value of (D1, D0). Angle range is:
 $0^{\circ} \leq \text{angle value} < 360^{\circ}$. After converting to SIN value to save in (D11, D10) with binary floating point
 number.



API	Mnemonic			Operands		Function										Controllers			
131	D	COS	P	<div>S</div>	<div>D</div>	Floating point Cosine Operation										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

Type OP	Bit Devices				Word devices										Program Steps				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DCOS, DCOSP: 9 steps			
	S					*	*						*						
D													*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

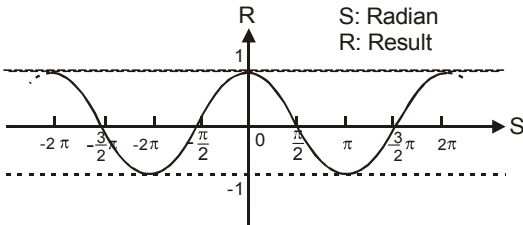
Operands:

S: Specified RAD value ($0^\circ \leq S < 360^\circ$) **D:** Area where calculated result is stored

Explanations:

- Source designated by **S** can be radian or angle by flag M1018.
- When M1018= OFF, is set to radian mode. $RAD = angle \times \pi / 180$.
- When M1018= ON, is set to angle mode. Angle range: $0^\circ \leq angle < 360^\circ$.
- The COS value of an angle data specified by **S** is calculated and the calculated result is stored in the register specified by **D**.

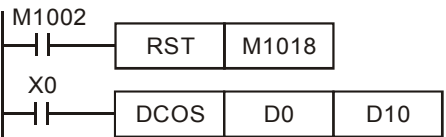
Following shows the relation between radian and result:

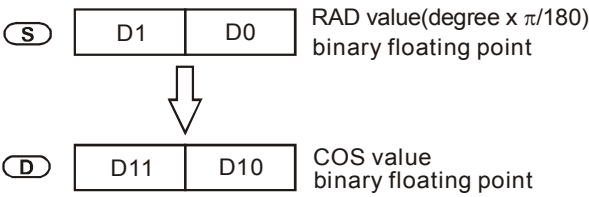


- Flag M1018 radian/angle switch: when M1018= OFF, **S** is RAD value. When M1018= ON, **S** is angle value (0~360).
- If result of D is 0, the Zero flag M1020=ON.

Program Example 1:

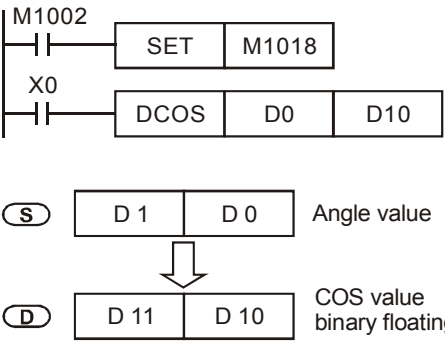
When M1018= OFF, it is radian mode. When X0= ON, specify RAD value (D1, D0). Calculate COS value of angle and store the result in (D11, D10). The value in (D1, D0) and the result stored in (D11, D10) are all in binary floating point format.





Program Example 2:

When M1018= ON, it is angle mode. When X0= ON, it is angle of specific (D1, D0). Angle range: $0^\circ \leq \text{angle} < 360^\circ$. After converting to COS value, save in (D11, D10) with binary floating point.



API	Mnemonic			Operands		Function										Controllers			
132	D	TAN	P	S	D	Floating point Tangent Operation										PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DTAN, DTANP: 9 steps			
S					*	*							*						
D													*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

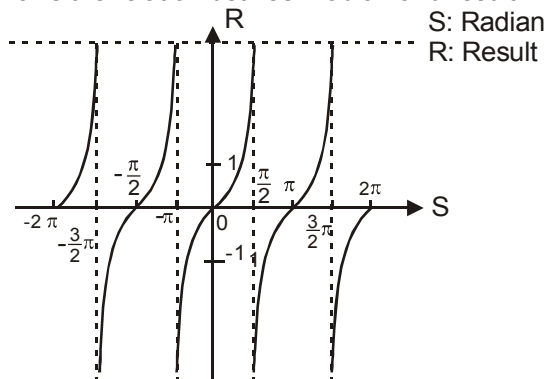
Operands:

S: Specified RAD value ($0^\circ \leq S < 360^\circ$) **D:** Area where calculated result is stored

Explanations:

- Source designated by **S** can be radian or angle by flag M1018.
- When M1018= OFF, is radian mode. $RAD = angle \times \pi / 180$.
- When M1018= ON, is angle mode. Angle range: $0^\circ \leq angle < 360^\circ$.
- The TAN value of an angle data specified by **S** is calculated and the calculated result is stored in the register specified by **D**.

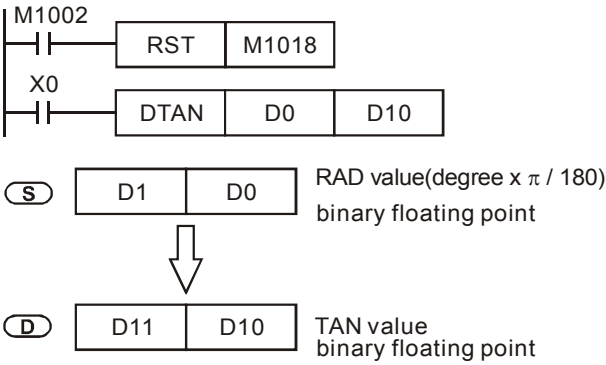
Following shows the relation between radian and result:



- Flag M1018 radian/angle switch: when M1018= OFF, **S** is RAD value. When M1018= ON, **S** is angle value (0~360).
- If result of D is 0, the zero flag M1020=ON.

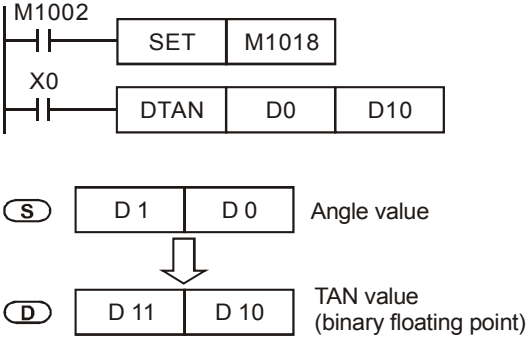
Program Example 1:

When M1018= OFF, is in radian mode. When X0= ON, specify RAD value (D1, D0). Calculate TAN value of angle and store the result in (D11, D10). The value in (D1, D0) and the result stored in (D11, D10) are all in binary floating point format.



Program Example 2:

When M1018= ON, is angle mode. When X0= ON, it is angle of specific (D1, D0). Angle range: $0^\circ \leq \text{angle} < 360^\circ$. After converting to TAN value, save in (D11, D10) with binary floating point.



API	Mnemonic			Operands		Function										Controllers			
133	D	ASIN	P	S	D	Float Arc Sine Operation										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DASIN, DASINP: 9 steps		
	S					*	*							*					
D														*					

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

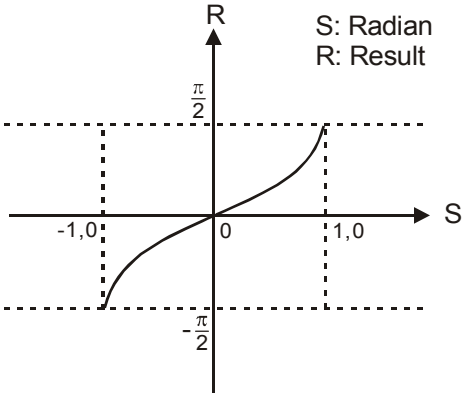
Operands:

S: Specified source (binary floating point) **D:** Area where calculated result is stored

Explanations:

1. ASIN value= SIN^{-1}

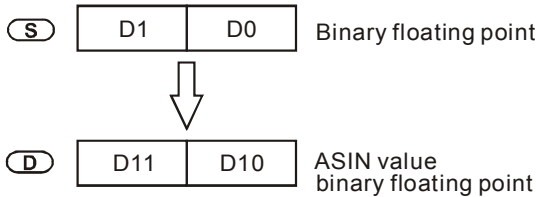
Following shows the relation between radian and result:



2. If result of D is 0, the zero flag M1020=ON.

Program Example:

When X0= ON, specify binary floating point (D1, D0). Calculate ASIN value and save the result in (D11, D10). The result stored in (D11, D10) is all in binary floating point format.



API	Mnemonic			Operands		Function										Controllers											
	D	ACOS	P	S	D	Float Arc Cosine Operation										PB	PC	PA	PH								
134																											
Type OP	Bit Devices				Word devices											Program Steps											
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DACOS, DACOSP: 9 steps											
S					*	*							*														
D													*														
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

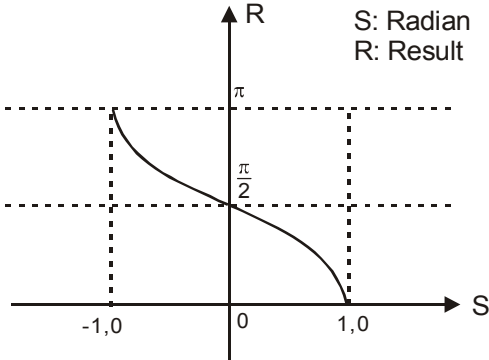
Operands:

S: Specified source (binary floating point) D: Area where calculated result is stored

Explanations:

1. ACOS value=COS⁻¹

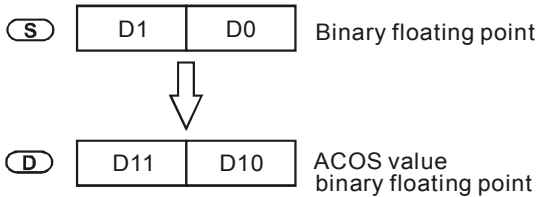
Following shows the relation between radian and result:



2. If result of D is 0, the zero flag M1020=ON.

Program Example:

When X0= ON, specify binary floating point (D1, D0). Calculate ACOS value and save the result in (D11, D10). The result stored in (D11, D10) is all in binary floating point format.



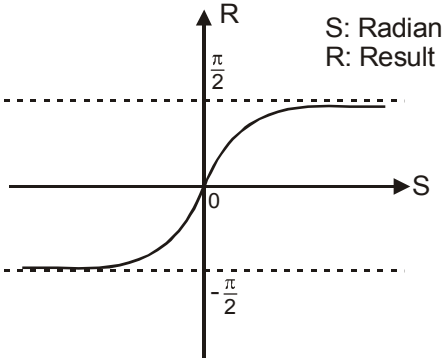
API	Mnemonic			Operands		Function										Controllers													
135	D	ATAN	P	S	D	Float Arc Tangent Operation										PB	PC	PA	PH										
OP	Type	Bit Devices				Word devices										Program Steps													
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DATAN, DATANP: 9 steps												
	S					*	*							*															
	D													*															
																		PULSE				16-bit				32-bit			
																		PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: Specified source (binary floating point) D: Area where calculated result is stored

Explanations:

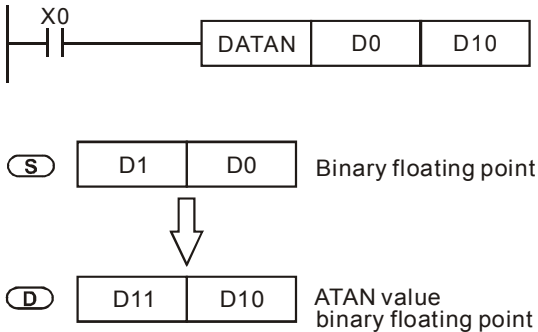
1. ATAN value=TAN⁻¹
Following shows the relation between radian and result:



2. If result of D is 0, the zero flag M1020=ON.

Program Example:

When X0= ON, specify binary floating point (D1, D0). Calculate ATAN value and save the result in (D11, D10). The result stored in (D11, D10) is all in binary floating point format.



API	Mnemonic			Operands			Function								Controllers			
143		DELAY	P	S			Delay								PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DELAY, DELAYP: 3 steps		
S					*	*							*					

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

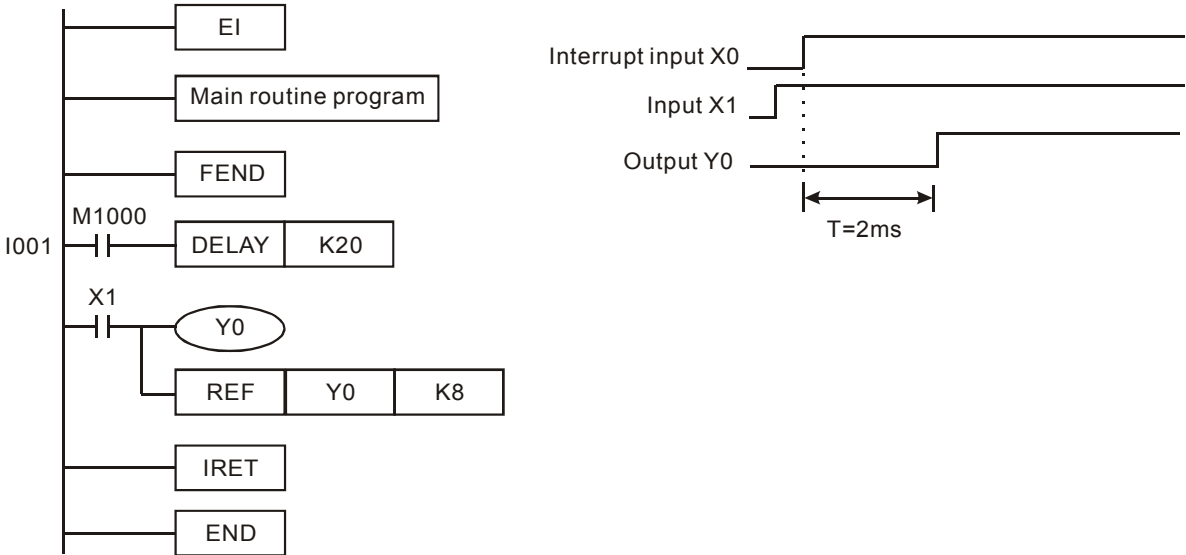
S: Delay time, unit is 0.1ms (K1~K1000)

Explanations:

After executing DELAY instruction, in every scanning cycle, the program after DELAY command will delay its execution according to the delay time that user assign.

Program Example:

If an external interrupt occurs when X0 goes from OFF to ON, the interrupt subroutine will execute the DELAY instruction and Y0=ON will delay 2 seconds after X1=ON.



Points to note:

1. User can adjust the delay time according to the actual condition.
2. When executing DELAY instruction, the delay time may increase upon the influence of communication, high-speed counter and high-speed pulse output commands.
3. If the external output (transistor output or relay output) is specified, the delay time may increase due to the transistor or relay it self.

API	Mnemonic	Operands	Function	Controllers			
144	GPWM	(S₁) (S₂) (D)	General PWM Output	PB	PC	PA	PH

Type	Bit Devices				Word devices										Program Steps	
OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	GPWM: 7 steps
S ₁													*			
S ₂													*			
D		*	*	*												

Operands:

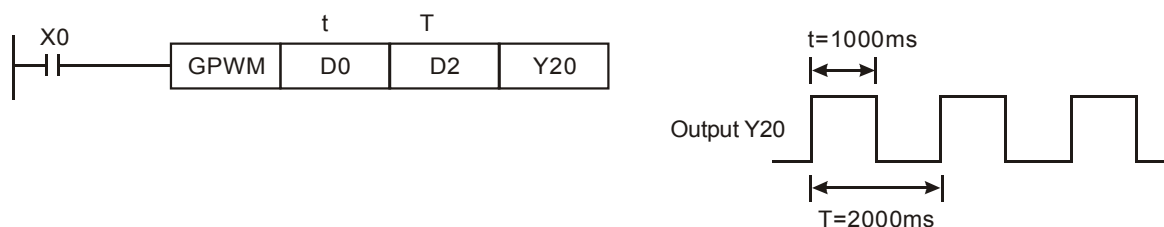
S₁: Pulse output width **S₂**: Pulse output cycle (occupies 3 devices) **D**: Pulse output device

Explanations:

1. **S₁** is specified as pulse output width as t:0~32,767ms.
2. **S₂** is specified as pulse output cycle as T:1~32,767ms, **S₁ ≤ S₂**.
3. **S₂ + 1** and **S₂ + 2** are for system, please don't use them. **D** pulse output devices: Y, M and S.
4. When the GPWM instruction has been executed, the pulse output width **S₁** and pulse output cycle **S₂** is output through pulse output device **D**.
5. When **S₁ ≤ 0**, there is no pulse output from the pulse output device. When **S₁ ≥ S₂**, the pulse output device will be always ON.
6. **S₁** and **S₂** can be modified when executing PWM instruction.

Program Example:

When D0=K1000,D2=K2000,then X0= ON, Y20 will output following pulse. When X0= OFF, Y20 output will also be OFF.

**Points to note:**

1. This instruction counts by scan cycle so the maximum offset will be a ELC scan cycle. The value of **S₁**, **S₂** and (**S₂ - S₁**) should be larger than ELC scan cycle. Otherwise, there will be error occurs for GPWM outputs.
2. If using this instruction in subroutine or interruption, GPWM output may not be accurate.

API	Mnemonic	Operands				Function										Controllers			
145	FTC	S₁	S₂	S₃	D	Fuzzy Temperature Control										PB	PC	PA	PH

Type OP	Bit Devices				Word devices												Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	FTC: 9 steps				
S ₁					*	*							*							
S ₂					*	*							*							
S ₃													*							
D													*							

Operands:

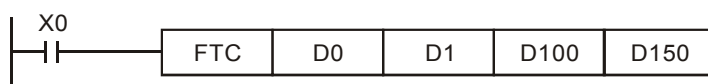
S₁: Target value (SV) **S₂**: Present measured value (PV) **S₃**: Parameter (occupies continuous 6 devices) **D**: Output value (MV)

Explanations:

- Operand **S₁** range is 1~5000 to show 0.1°C ~500°C. The unit is 0.1°C. If **S₃+1** (refer to Notes) sets to K0 to show 0.1°C~500°C.
- Operand **S₂** range is 1~5000 to show 0.1°C ~500°C. The unit is 0.1°C. If **S₃+1** (refer to Notes) sets to K0 to show 0.1°C~500°C.
- Therefore, when the result that analog converts to digital from temperature sensor, it will convert to the value during 1~5000 by using the four fundamental operations of arithmetic.
- S₃** is sampling time setting. If setting is less than K1, instruction won't act. If setting exceeds K200, it will be regarded as K200.
- The setting bit0 of **S₃+1** be K0 (means °C) and K1 (means °F). bit1=0 of **S₃+1** has filter function, bit1=1 of **S₃+1** has no filter function. The bit2~bit5 of **S₃+1** refer to Notes
- The range of output MV is from 0 to Ts. This instruction should be used with instruction GPWM. And the sampling time that used by this instruction should be the same with cycle time used by GPWM as shown in example 1. (The unit of time for this instruction is 100ms and 1ms for GPWM)
- There is no number-of-times limit when using FTC instruction, but the specify devices of **S₃** cannot be repeated use.

Program Example:

- Preset parameters before executing FTC instruction.
- When X0= ON, instruction is executed and save result in D150. When X0= OFF, instruction is not executed and previous data is unchanged.

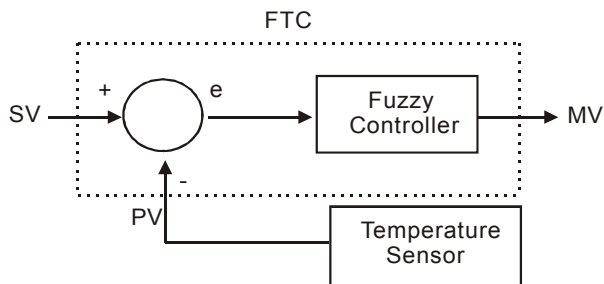


Notes:

1. The setting of S_3 is in the following:

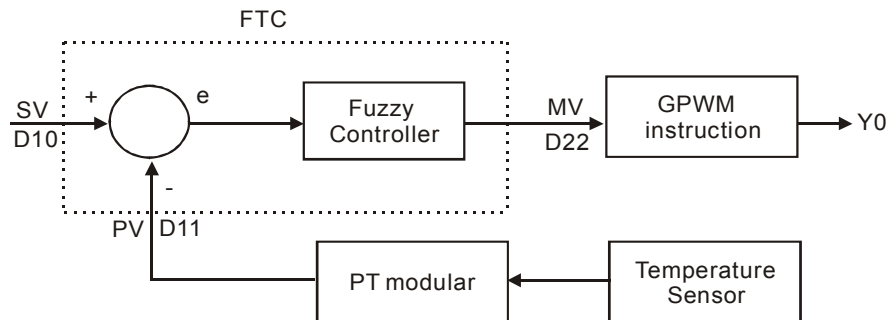
Device	Function	Usage range	Explanation
S_3 :	Sampling time (T_s)	1~200 (unit: 100ms)	When T_s is less than a scan time, PID instruction will execute for a scan time. When $T_s=0$, it won't act. Therefore, the minimum setting of T_s should be larger than program scan time. When setting exceeds 200, it will be regarded as 200.
$S_3 +1$:	b0:Temperature unit b1:filter function b2~b5:heat environment b6~b15 preserved	b0=0 : °C b0=1 : °F	Temperature unit
		b1=0:no filter b1=1:filter	When in no filter function, present value(PV)=present measure value. If in filter function, present value(PV)=(present measure value + previous measure value)/2
		b2=1	Slow heat environment
		b3=1	General heat environment
		b4=1	Rapid heat environment
		b5=1	High-speed heat environment
$S_3 +2$: } $S_3 +6$:	For system uses, please don't use.		

2. Control Diagram:

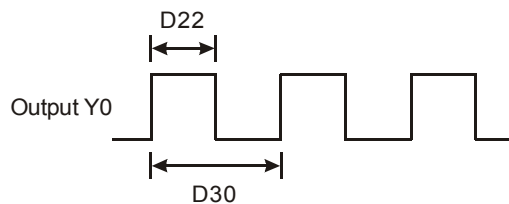


Points to notes:

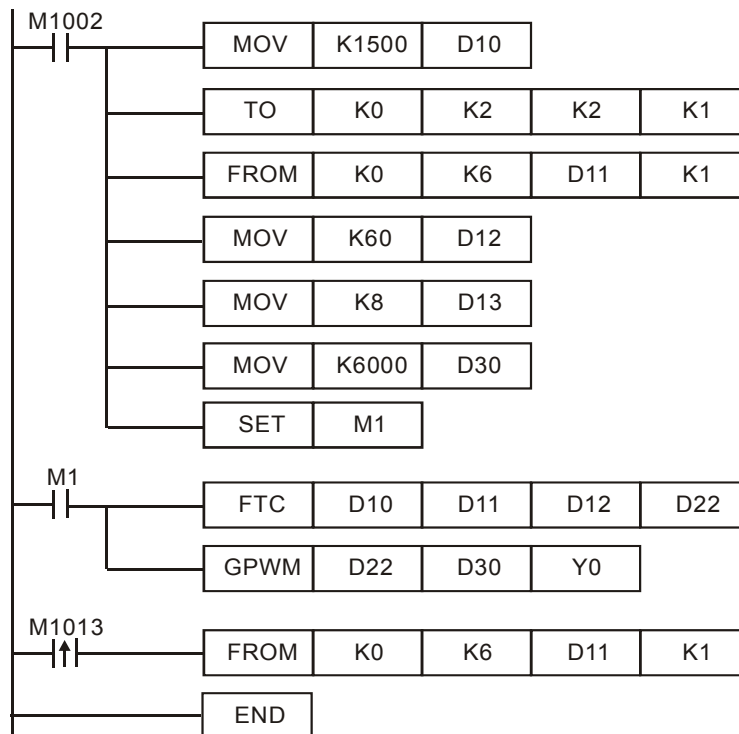
It is recommended to set sampling time to twice and above of sampling time of temperature sensor to get better temperature control.

Example 1: control diagram

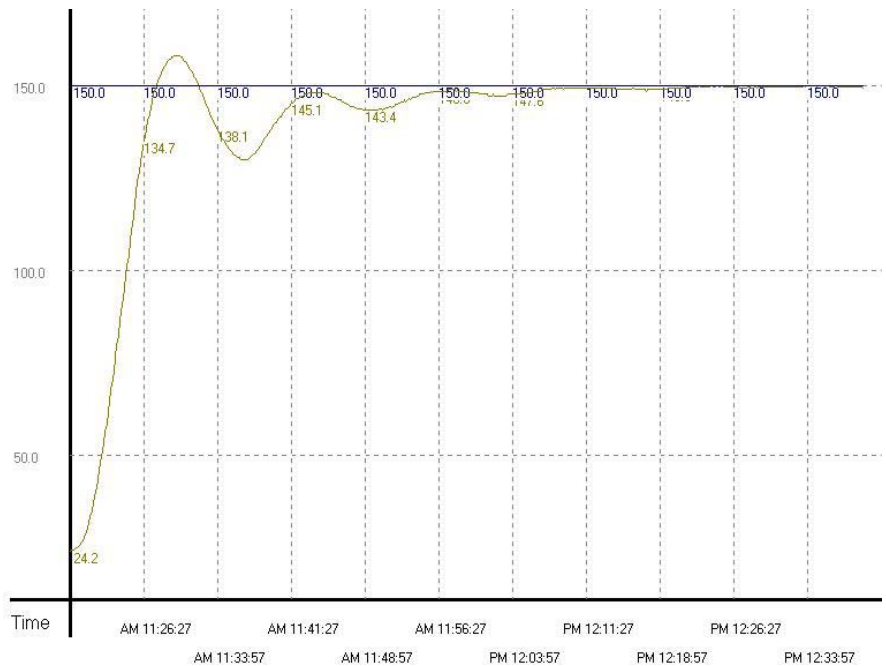
The output D22(MV) of FTC instruction is used to be the input of GPWM instruction. The function of D22 is the pulse output width and D30 is pulse output cycle. The timing chart of output Y0 is shown as follows.



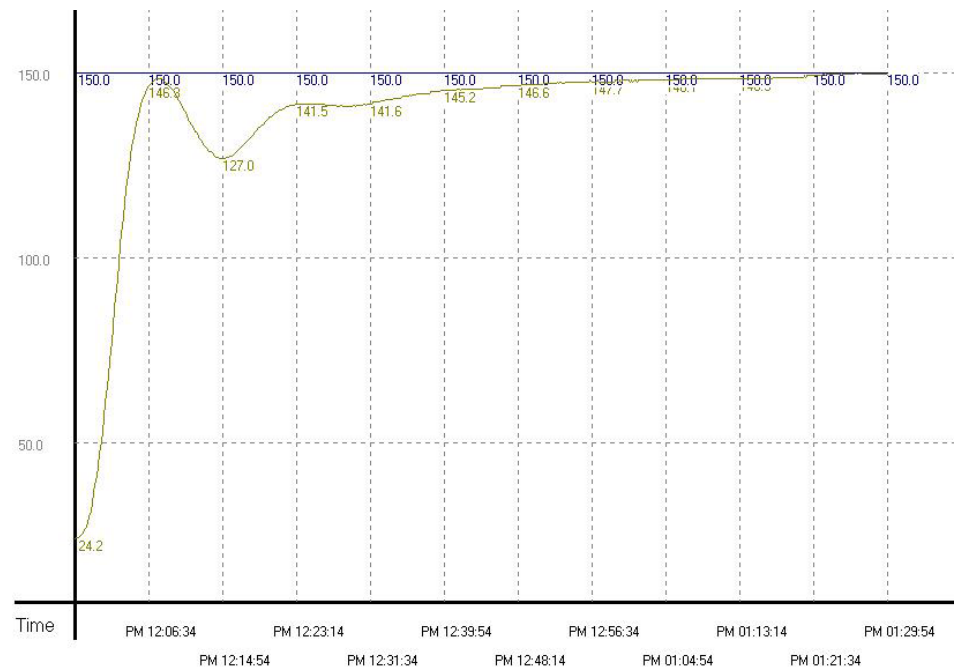
The settings of FTC instruction for this example are D10=k1500(target temperature), D12=k60(sampling time is 6 seconds), D13=k8 (Bit3=1) and D30=k6000 (=D12*100). The ladder diagram of this program is shown as follows.



The test environment is oven(The temperature it can heat is 250℃). The records of target and real temperature are shown as follows.



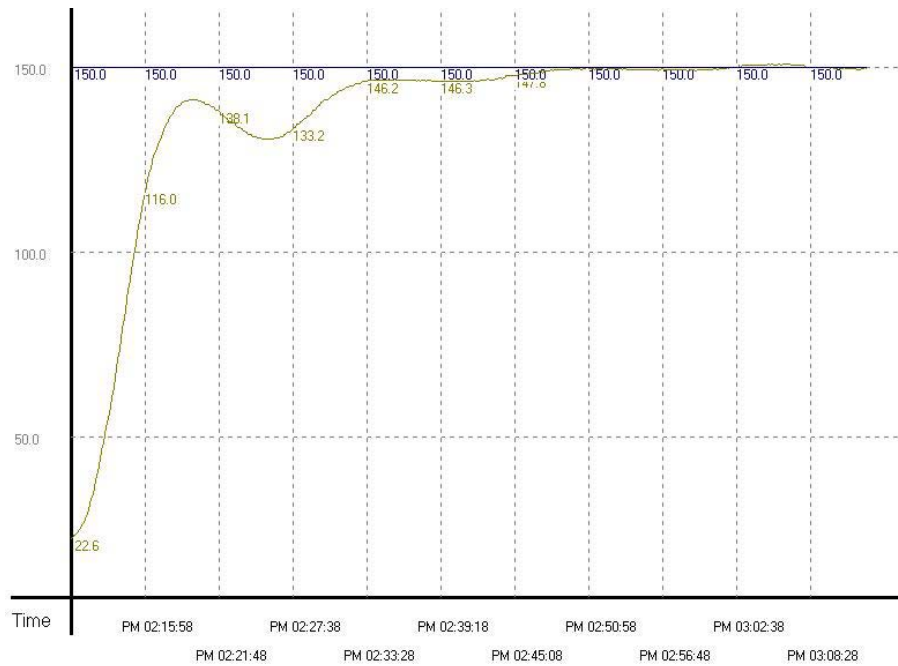
Example 2: The heat environment is modified to rapid heat environment (i.e. D13=k16) due to overshoot. The result after testing is shown as follows.



From above figure, you can find that there is no overshoot but it takes more than one hour and fifteen

minutes to make the error of target temperature within $\pm 1^{\circ}\text{C}$. Therefore, current test environment is correct but long sampling time delays whole time.

Example 3: To make example 2 reach target temperature faster, set sampling time to 4 seconds (i.e. D12=k40 and D30=k4000). The result after testing is shown as follows.



API	Mnemonic			Operands	Function										Controllers										
147	D	SWAP	P	(S)	Swap High/Low Byte										PB	PC	PA	PH							
OP	Type	Bit Devices				Word devices										Program Steps									
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SWAP, SWAPP: 3 steps									
	S							*	*	*	*	*	*	*	*	*	DSWAP, DSWAPP: 5 steps								
														PULSE				16-bit				32-bit			
														PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

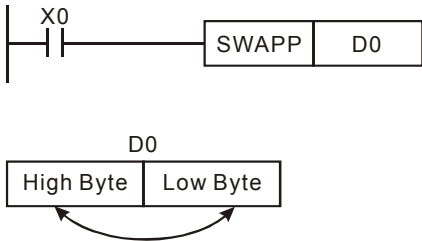
S: Device for swapping high/low byte.

Explanations:

- 1. When being 16-bit instruction, swapping the content of high/low byte.
- 2. When being 32-bit instruction, swapping the content of high/low byte of two registers separately.
- 3. This instruction works best using the pulse execution (SWAPP, DSWAPP).
- 4. If operand **D** uses with device F, it is only available in 16-bit instruction.

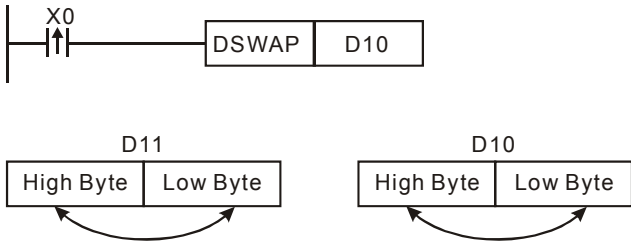
Program Example 1:

When X0=ON, swapping the content of high/low byte of D0.



Program Example 2:

When X0=ON, swapping upper 8-bit and lower 8-bit of D11 and swapping upper 8-bit and lower 8-bit of D10.



API	Mnemonic			Operands			Function								Controllers			
148	D	MEMR	P	m	D	n	File Memory Read								PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MEMR, MEMRP: 7 steps DMEMR, DMEMRP: 13 steps		
m					*	*							*					
D													*					
n					*	*							*					

PULSE				16-bit				32-bit							
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

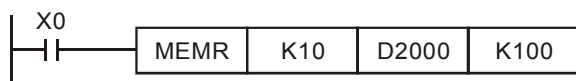
m: Address (Constant) for reading data of file register (**m**=0~1,599) **D**: Address (Constant) for storing read data (D2000~D4999) **n**: Quantity of one time reading data (16-bit instruction: **n**=K1~ K1,600, 32-bit instruction: **n**=K1~ K800)

Explanations:

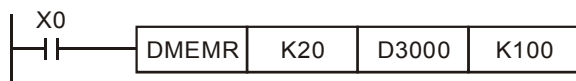
- If operands **m**, **D** and **n** are out of range, operand error will occur. M1067, M1068= ON and error code 0E1A will be recorded in D1067.
- Flag: M1101, please refer the following Note for detail.

Program Example 1:

- 16-bit instruction MEMR reads 100 items data from the 10th address of file register and store the read data in the data register started from D2000.
- When X0= ON, the instruction is executed. When X0 goes to OFF, the instruction is not executed and the content of previous read data has no change.

**Program Example 2:**

- 32-bit instruction DMEMR reads 100 items data from the 20th address of file register and store the read data in the data register started from D3000.
- When X0= ON, the instruction is executed. When X0 goes to OFF, the instruction is not executed and the content of previous read data has no change.



Refer to the note of file registers of API149 MEMW

API	Mnemonic			Operands			Function								Controllers			
149	D	MEMW	P	S	m	n	File Memory Write								PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E				
S													*			MEMW, MEMWP: 7 steps DMEMW, DMEMWP: 13 steps		
m					*	*							*					
n					*	*							*					

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

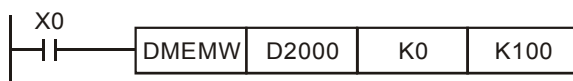
S: Address (Constant) for data writing in, (D2,000~D4,999) **m:** Address (Constant) for file register writing in, K0~K1,599 **n:** Quantity of one time reading data, (16-bit instruction: **n**= K1~ K1,600, 32-bit instruction: **n**=K1~ K800)

Explanations:

1. If operands **m**, **D** and **n** is out of range, operand error will be occurred. M1067, M1068= ON and error code 0E1A will be recorded in D1067.
2. Flag: M1101, please refer the following Note for detail.

Program Example:

1. When X0= ON, the double word instruction DMEMW is executed. Write 100 items 32-bit data started from D2001, D2000 into the file register address 0 to 199.
2. When X0= ON, the instruction is executed. When X0 goes to OFF, the instruction is not executed and the content of previous read data has no change.

**File Register:**

1. When ELC startup, PC/PA/PH series ELC will determine M1101 (if startup the function of file register), D1101 (file register starts to give number, K0~K1,599), D1102 (numbers of file registers of being read, K1~K1,600), D1103 (destination device which stores the read data of file register, specified data register D start to give number, K2,000~K4,999) and decide if automatically transfer the content of file register to the specified data register.
2. When the value of D1101 is less than 0 or more than 1,599, or the value of D1103 is less than 2,000 or more than 4,999, reading data from file register to data register is disabled.
3. When file register read data to data register D, if the address of file register or data register exceeds the limit range, ELC will stop reading.

4. As for the data read and write in of file register, in ELC program only can use API instruction 148 MEMR to read and use API instruction 149 MEMW to write in. For detailed information about file registers.
5. There are 1,600 file registers. The file registers don't have real number, therefore the read/write in function of file register should be performed by the API instruction 148 MEMR and 149 MEMW, or using a peripheral equipment HHP and ELCSoft software.
6. Related special relays and registers of file register:

Flag	Function Explanation
M1101	If startup the function of file register, Latched, Default is OFF

Special Register	Function Explanation
D1101	File register starts to give number K0~K1,599, Latched, Default is 0
D1102	Numbers of file registers of being read K1~K1,600, Latched, Default is 0
D1103	Destination device which stores the read data of file register, specified data register D start to give number K2,000~K4,999, Latched, Default is 2,000

API	Mnemonic	Operands					Function					Controllers			
150	MODRW	$\textcircled{S_1}$	$\textcircled{S_2}$	$\textcircled{S_3}$	\textcircled{S}	\textcircled{n}	MODBUS Read/ Write					PB	PC	PA	PH

Type OP	Bit Devices				Word devices										MODRW: 11 steps				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E					F
S_1					*	*							*						
S_2					*	*							*						
S_3					*	*							*						
S													*						
n					*	*							*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S_1 : External device address (K0~K254) S_2 : Function code (K3(H3), K6(H6), K16(H10)) S_3 : Data address being read from or written to within the external device S : Register of being read/write
 n : Length of read/write data.

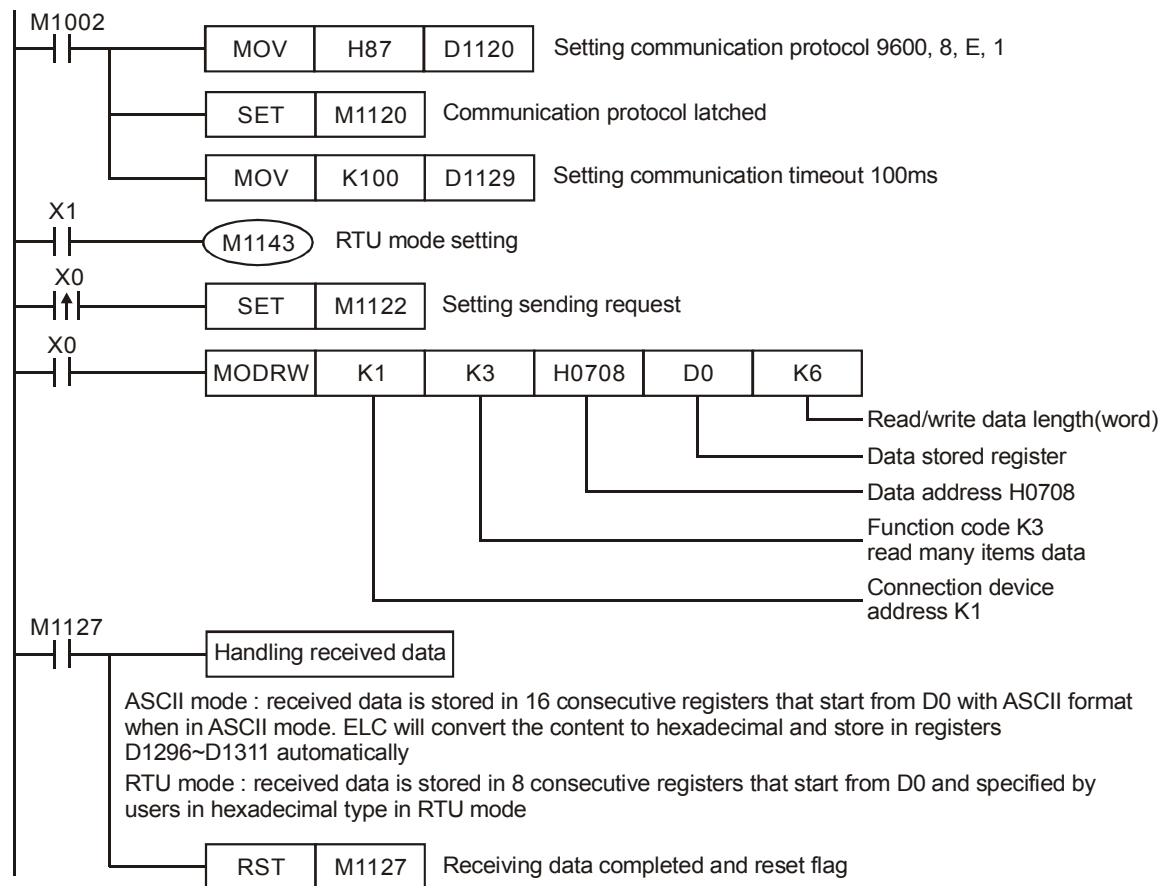
Explanations:

1. S_1 : Address of external device the ELC will communicate with. The valid range is K0~K254.
2. S_2 : Function code. For example: the instruction to read multiple items from an MVX drive is H03.
3. S_3 : Data address internal to the device that ELC is communicating with. If the address is illegal for the device, there will be fault code stored in D1130 and at the same time, M1141 = ON. For example, 8000H is illegal for MVX, M1141 = ON and D1130 = 2. Please refer to MVX user manual for the details on fault codes.
4. S : Memory area in ELC where data to be transmitted and received data is stored.
5. n : Read/Write data length (DATA LENGTH). If M1143=ON(RTU Mode), n =K1~K16. If M1143=OFF(ASCII Mode), n =K1~K8.
6. Flags: M1120~M1131, M1140~M1143, please refer to the RS (API 80) instruction for more information.

Program Example 1:

1. Function code K3(H3) : read multiple data items
 ELC connects to MVX drive. (ASCII Mode, when M1143=OFF)
 ELC connects to MVX drive. (RTU Mode, when M1143=ON)
2. Received data is stored in 16 continuous registers that start from D0 with ASCII format when in ASCII mode. ELC will convert the content to Hexadecimal and store in registers D1296~D1311 automatically. M1131=ON when ELC starts converting to hexadecimal and M1131 = OFF after conversion complete.

3. Use MOV, DMOV or BMOV instructions to move hexadecimal data from D1296~D1311 and store in general use registers.
4. While in RTU mode, received data is stored in 8 continuous registers that start from D0. At the same time, using D1296~D1311 is invalid.
5. In ASCII or RTU mode, data to be transmitted by ELC is stored in D1256~D1295. Use MOV, DMOV or BMOV instructions. Other instructions are invalid to address this area.
6. After receiving data, ELC will automatically check if the received data is correct. If there is any fault, M1140 = ON and the fault code will be stored in D1130.
7. After M1140=ON or M1141=ON, ELC will re-transmit data to a MVX drive. If received data is correct, M1140 and M1141 will be reset.



8. ASCII Mode: ELC connects to MVX drive.

ELC → MVX, ELC transmits: " 01 03 0708 0006 E7 "

MVX → ELC, ELC receives: " 01 03 0C 0100 1766 0000 0000 0136 0000 3B "

ELC transmits data register (transmit message)

Register	Data		Descriptions	
D1256 Low byte	‘0’	30 H	ADR 1	ADR (1,0) is MVX drive address
D1256 High byte	‘1’	31 H	ADR 0	
D1257 Low byte	‘0’	30 H	CMD 1	CMD (1,0) is instruction code
D1257 High byte	‘3’	33 H	CMD 0	
D1258 Low byte	‘0’	30 H	Data Address	
D1258 High byte	‘7’	37 H		
D1259 Low byte	‘0’	30 H		
D1259 High byte	‘8’	38 H		
D1260 Low byte	‘0’	30 H	Number of data (count by word)	
D1260 High byte	‘0’	30 H		
D1261 Low byte	‘0’	30 H		
D1261 High byte	‘6’	36 H		
D1262 Low byte	‘E’	45 H	LRC CHK 1	LRC CHK (0,1) error check
D1262 High byte	‘7’	37 H	LRC CHK 0	

ELC receives data register (response message)

Register	Data		Descriptions	
D0 low byte	‘0’	30 H	ADR 1	
D0 high byte	‘1’	31 H	ADR 0	
D1 low byte	‘0’	30 H	CMD 1	
D1 high byte	‘3’	33 H	CMD 0	
D2 low byte	‘0’	30 H	Number of data (count by byte)	
D2 high byte	‘C’	43 H		
D3 low byte	‘0’	30 H	Content of address 0708 H	ELC automatically converts ASCII codes to hex and store the converted value in D1296 = 0100 H
D3 high byte	‘1’	31 H		
D4 low byte	‘0’	30 H		
D4 high byte	‘0’	30 H		
D5 low byte	‘1’	31 H	Content of address 0709 H	ELC automatically converts ASCII codes to hex and store the converted value in D1297 = 1766 H
D5 high byte	‘7’	37 H		
D6 low byte	‘6’	36 H		
D6 high byte	‘6’	36 H		
D7 low byte	‘0’	30 H	Content of address 070A H	ELC automatically converts ASCII codes to hex and store the converted value in D1298 = 0000 H
D7 high byte	‘0’	30 H		
D8 low byte	‘0’	30 H		
D8 high byte	‘0’	30 H		
D9 low byte	‘0’	30 H	Content of address 070B H	ELC automatically converts ASCII codes to hex and store the converted value in D1299 = 0000 H
D9 high byte	‘0’	30 H		
D10 low byte	‘0’	30 H		
D10 high byte	‘0’	30 H		
D11 low byte	‘0’	30 H	Content of address 070C H	ELC automatically converts ASCII codes to hex and store the converted value in D1300 = 0136 H
D11 high byte	‘1’	31 H		
D12 low byte	‘3’	33 H		
D12 high byte	‘6’	36 H		
D13 low byte	‘0’	30 H	Content of address 070D H	ELC automatically converts ASCII codes to hex and store the converted value in D1301 = 0000 H
D13 high byte	‘0’	30 H		
D14 low byte	‘0’	30 H		
D14 high byte	‘0’	30 H		
D15 low byte	‘3’	33 H	LRC CHK 1	
D15 high byte	‘B’	42 H	LRC CHK 0	

9. RTU Mode: ELC connects to MVX drive

ELC → MVX, ELC transmits: 01 03 0708 0006 45 7E

MVX → ELC, ELC receives: 01 03 0C 0000 0503 0BB8 0BB8 0000 012D 8E C5

ELC transmits data register (transmit message)

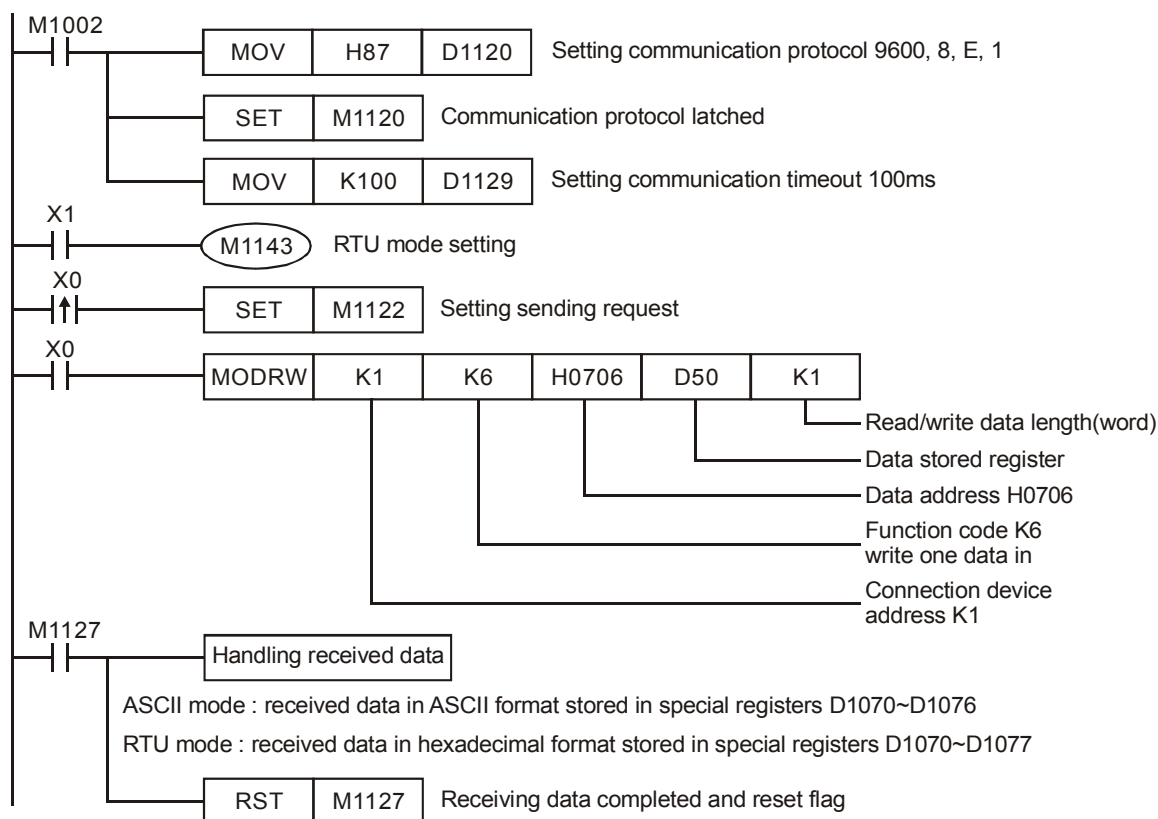
Register	Data	Descriptions
D1256 Low byte	01 H	Address
D1257 Low byte	03 H	Function
D1258 Low byte	07 H	Data Address
D1259 Low byte	08 H	
D1260 Low byte	00 H	Number of data (count by word)
D1261 Low byte	06 H	
D1262 Low byte	45 H	CRC CHK Low
D1263 Low byte	7E H	CRC CHK High

ELC receives data register (response message)

Register	Data	Descriptions	
D0 low byte	01 H	Address	
D1 low byte	03 H	Function	
D2 low byte	0C H	Number of data (count by byte)	
D3 low byte	00 H	Content of address 0708 H	ELC automatically store the value in D1296 = 0000 H
D4 low byte	00 H		
D5 low byte	05 H	Content of address 0709 H	ELC automatically store the value in D1297 = 0503 H
D6 low byte	03 H		
D7 low byte	0B H	Content of address 070A H	ELC automatically store the value in D1298 = 0BB8 H
D8 low byte	B8 H		
D9 low byte	0B H	Content of address 070B H	ELC automatically store the value in D1299 = 0BB8 H
D10 low byte	B8 H		
D11 low byte	00 H	Content of address 070C H	ELC automatically store the value in D1300 = 0000 H
D12 low byte	00 H		
D13 low byte	01 H	Content of address 070D H	ELC automatically store the value in D1301 = 012D H
D14 low byte	2D H		
D15 low byte	8E H	CRC CHK Low	
D16 low byte	C5 H	CRC CHK High	

Program Example 2:

- Function code K6(H6) : write one WORD data into register
ELC connects to MVX drive. (ASCII Mode when M1143=OFF)
ELC connects to MVX drive. (RTU Mode when M1143=ON)
- When in ASCII mode, store the data that will be written to MVX drive in ASCII format in register D50. Data received from MVX drive will be stored in registers D1070~D1076.
- When in RTU mode, store data that will be written to MVX drive in hexadecimal format in register D50. Data received from MVX drive will be stored in registers D1070~D1077.
- When in ASCII or RTU mode, ELC will transmit the data stored in registers D1256~D1295.
- After receiving a response, ELC automatically checks if the received data is correct. If there is any fault, M1140 will be set to ON.
- If the register address of the MVX drive is illegal, a fault code will be stored in D1130 and M1141 = ON. For example, 8000H is illegal for MVX and M1141=ON and D1130=2. Refer to MVX user manual to fault code.
- After M1140=ON or M1141=ON, ELC will re-transmit the same data to the MVX drive. If the received data is correct, M1140 and M1141 will be reset.



8. ASCII Mode: ELC connects to MVX drive.

ELC → MVX, ELC transmits: “ 01 06 0706 1770 65 ”

MVX → ELC, ELC receives: “ 01 06 0706 1770 65 ”

ELC transmits data register (transmit message)

Register	Data		Descriptions	
D1256 Low byte	‘0’	30 H	ADR 1	ADR (1,0) is MVX drive address
D1256 High byte	‘1’	31 H	ADR 0	
D1257 Low byte	‘0’	30 H	CMD 1	CMD (1,0) is function code
D1257 High byte	‘6’	36 H	CMD 0	
D1258 Low byte	‘0’	30 H	Data Address	
D1258 High byte	‘7’	37 H		
D1259 Low byte	‘0’	30 H		
D1259 High byte	‘6’	36 H		
D1260 Low byte	‘1’	31 H	Data contents	The content of register D50 (H1770=K6000)
D1260 High byte	‘7’	37 H		
D1261 Low byte	‘7’	37 H		
D1261 High byte	‘0’	30 H		
D1262 Low byte	‘6’	36 H	LRC CHK 1	LRC CHK (0,1) is error check
D1262 High byte	‘5’	35 H	LRC CHK 0	

ELC receives data register (response message)

Register	Data		Descriptions
D1070 Low byte	‘0’	30 H	ADR 1
D1070 High byte	‘1’	31 H	ADR 0
D1071 Low byte	‘0’	30 H	CMD 1
D1071 High byte	‘6’	36 H	CMD 0
D1072 Low byte	‘0’	30 H	Data Address
D1072 High byte	‘7’	37 H	
D1073 Low byte	‘0’	30 H	
D1073 High byte	‘6’	36 H	
D1074 Low byte	‘1’	31 H	Data content
D1074 High byte	‘7’	37 H	
D1075 Low byte	‘7’	37 H	
D1075 High byte	‘0’	30 H	
D1076 Low byte	‘6’	36 H	LRC CHK 1
D1076 High byte	‘5’	35 H	LRC CHK 0

9. RTU Mode: ELC connects to MVX drive

ELC → MVX, ELC transmits: 01 06 0706 1770 66 AB

MVX → ELC, ELC receives: 01 06 0706 1770 66 AB

ELC transmits data register (transmit message)

Register	Data	Descriptions	
D1256 Low byte	01 H	Address	
D1257 Low byte	06 H	Function	
D1258 Low byte	07 H	Data Address	
D1259 Low byte	06 H		
D1260 Low byte	17 H	Data content	The content of register D50 (H1770=K6000)
D1261 Low byte	70 H		
D1262 Low byte	66 H	CRC CHK Low	
D1263 Low byte	AB H	CRC CHK High	

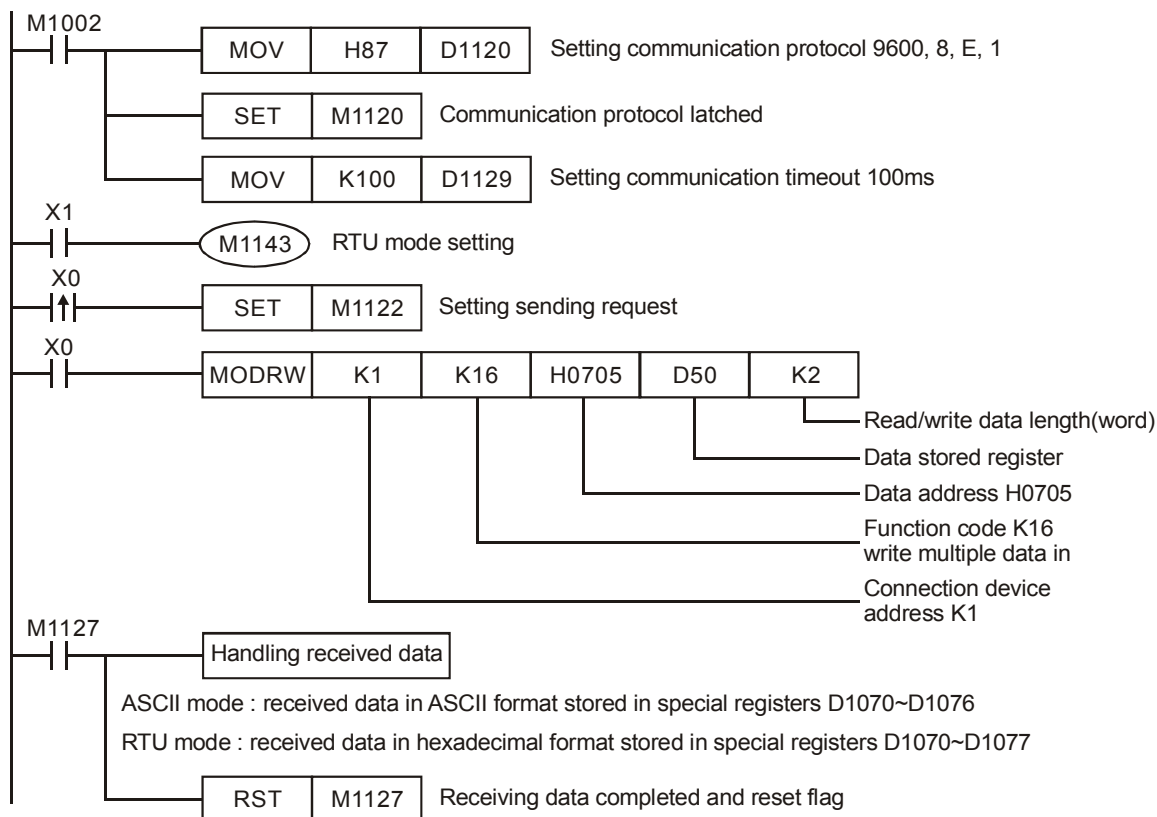
ELC receives data register (response message)

Register	Data	Descriptions	
D1070 Low byte	01 H	Address	
D1071 Low byte	06 H	Function	
D1072 Low byte	07 H	Data Address	
D1073 Low byte	06 H		
D1074 Low byte	17 H	Data content	
D1075 Low byte	70 H		
D1076 Low byte	66 H	CRC CHK Low	
D1077 Low byte	AB H	CRC CHK High	

Program Example 3:

- Function code K16(H10) : write multiple WORD items of data into register
ELC connects to MVX drive. (ASCII Mode when M1143=OFF)
ELC connects to MVX drive. (RTU Mode when M1143=ON)
- In ASCII mode, store the data that will be written to the MVX drive in ASCII format in 8 continuous specified register started from D50. Received data from the MVX drive will be stored in registers D1070~D1076.
- When in RTU mode, users store the data that will be wrote to MVX drive in hexadecimal format in 8 continuous specified register started from D50. Received data from MVX drive will be stored in registers D1070~D1077.

4. In ASCII or RTU mode, ELC will transmit data store in registers D1256~D1295. Preload these registers using MOV, DMOV or BMOV instructions. Other instructions are invalid to write to this area.
5. After receiving a response, ELC automatically checks if the received data is correct. If there is any fault, M1140 = ON.
6. If the register address of the MVX drive is illegal, a fault code will be stored in D1130 and M1141 = ON. For example, 8000H is illegal for MVX and M1141=ON and D1130=2. Refer to MVX user manual to fault code.
7. After M1140=ON or M1141=ON, ELC will re-transmit the same data to the MVX drive. If the received data is correct, M1140 and M1141 will be reset.



8. ASCII Mode: ELC connects to MVX drive.
 ELC → MVX, ELC transmits: " 01 10 0705 0002 04 1770 0012 44 "
 MVX → ELC, ELC receives: " 01 10 0705 0002 E1 "

ELC transmits data register (transmits messages)

Register	Data		Descriptions	
D1256 Low byte	‘0’	30 H	ADR 1	ADR (1,0) is MVX drive address
D1256 High byte	‘1’	31 H	ADR 0	
D1257 Low byte	‘1’	31 H	CMD 1	CMD (1,0) is instruction code
D1257 High byte	‘0’	30 H	CMD 0	
D1258 Low byte	‘0’	30 H	Data Address	
D1258 High byte	‘7’	37 H		
D1259 Low byte	‘0’	30 H		
D1259 High byte	‘5’	35 H		
D1260 Low byte	‘0’	30 H	Number of Register	
D1260 High byte	‘0’	30 H		
D1261 Low byte	‘0’	30 H		
D1261 High byte	‘2’	32 H		
D1262 Low byte	‘0’	30 H	Byte Count	
D1262 High byte	‘4’	34 H		
D1263 Low byte	‘1’	31 H	Data contents 1	The content of register D50 (H1770=K6000)
D1263 High byte	‘7’	37 H		
D1264 Low byte	‘7’	37 H		
D1264 High byte	‘0’	30 H		
D1265 Low byte	‘0’	30 H	Data contents 2	The content of register D51 (H12)
D1265 High byte	‘0’	30 H		
D1266 Low byte	‘1’	31 H		
D1266 High byte	‘2’	32 H		
D1267 Low byte	‘4’	34 H	LRC CHK 1	LRC CHK (0,1) is error check
D1267 High byte	‘4’	34 H	LRC CHK 0	

ELC receives data register (response messages)

Register	Data		Descriptions
D1070 Low byte	'0'	30 H	ADR 1
D1070 High byte	'1'	31 H	ADR 0
D1071 Low byte	'1'	31 H	CMD 1
D1071 High byte	'0'	30 H	CMD 0
D1072 Low byte	'0'	30 H	Data Address
D1072 High byte	'7'	37 H	
D1073 Low byte	'0'	30 H	
D1073 High byte	'5'	35 H	
D1074 Low byte	'0'	30 H	Number of Register
D1074 High byte	'0'	30 H	
D1075 Low byte	'0'	30 H	
D1075 High byte	'2'	32 H	
D1076 Low byte	'E'	45 H	LRC CHK 1
D1076 High byte	'1'	31 H	LRC CHK 0

9. RTU Mode: ELC connects to MVX drives

ELC → MVX, ELC transmits: 01 10 0705 0002 04 1770 0012 91 C2

MVX → ELC, ELC receives: 01 10 0705 0002 50 BD

ELC transmits data register (transmits messages)

Register	Data	Descriptions	
D1256 Low byte	01 H	Address	
D1257 Low byte	10 H	Function	
D1258 Low byte	07 H	Data Address	
D1259 Low byte	05 H		
D1260 Low byte	00 H	Number of Register	
D1261 Low byte	02 H		
D1262 Low byte	04 H	Byte Count	
D1263 Low byte	17 H	Data content 1	The content of register D50 (H1770=K6000)
D1264 Low byte	70 H		
D1265 Low byte	00 H	Data content 2	The content of register D51 (H12)
D1266 Low byte	12 H		
D1262 Low byte	91 H	CRC CHK Low	
D1263 Low byte	C2 H	CRC CHK High	

ELC receives data register (response messages)

Register	Data	Descriptions
D1070 Low byte	01 H	Address
D1071 Low byte	10 H	Function
D1072 Low byte	07 H	Data Address
D1073 Low byte	05 H	
D1074 Low byte	00 H	Number of Register
D1075 Low byte	02 H	
D1076 Low byte	50 H	CRC CHK Low
D1077 Low byte	BD H	CRC CHK High

Notes:

1. The contact before MODRD, MODRW, cannot use rising-edge or falling-edge contacts. Otherwise, the data stored in the received register will be incorrect.
2. Related flags and special registers for MODRW instruction: Refer to the RS instruction for more information.

Special M	Function Description
M1120	Latch communication settings. Any change to D1120 will not be valid after this is set.
M1121	When it is OFF, RS-485 of ELC is transmitting data.
M1122	Transmit request
M1123	Receive completed
M1124	Receiving message in process
M1125	Receive status disable
M1126	STX/ETX system definition selection
M1127	MODRD / MODWR / MODRW instructions data receive completed
M1128	Transmitting/receiving indication
M1129	Receive time out
M1130	Users/system definition STX/ETX
M1131	MODRD / MODRW data convert to HEX, M1131=ON
M1140	MODRD / MODWR / MODRW data receive error
M1141	MODRD / MODWR / MODRW instruction parameter error
M1142	MVX instruction data receive error
M1143	ASCII/RTU mode selection (OFF = ASCII, ON = RTU)

Special D	Function Description
D1070~D1085	Return response messages are saved in D1070~D1085.
D1120	RS-485 communication protocol, baud, stop bits, etc.
D1121	ELC communication address. (Latched)
D1122	Remaining characters to transmit
D1123	Remaining characters to receive
D1124	Start text definition (STX)
D1125	Definition of the first end character (ETX1)
D1126	Definition of the second end character (ETX2)
D1129	Communication time out. Time unit: (ms)
D1130	Return fault code record of MODBUS
D1256~D1295	Transmit data will be saved in D1256~D1295.
D1296~D1311	If in ASCII mode, ELC automatically converts ASCII data saved in the register specified by users to hexadecimal format.

API	Mnemonic			Operands			Function										Controllers			
154		RAND	P	(S ₁)	(S ₂)	(D)	Random Number										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	RAND, RANDP: 7 steps			
S ₁					*	*	*	*	*	*	*	*	*	*	*				
S ₂					*	*	*	*	*	*	*	*	*	*	*				
D								*	*	*	*	*	*	*	*				

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: lower limit for producing the random number **S₂**: upper limit for producing the random number

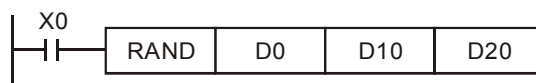
D: Random number result

Explanations:

- The range of operands **S₁**, **S₂** is: $K0 \leq S_1, S_2 \leq K32,767$, **S₁** operand \leq **S₂** operand.
- If **S₁** > **S₂**, ELC will produce an operand error and will execute the instruction, M1067 and M1068=ON, and error code 0E1A(HEX) will be recorded in D1067.

Program Example:

When X0=ON, the random number produced between lower limit D0 and upper limit D10 will be saved in D20.



API	Mnemonic			Operands			Function										Controllers			
155	D	ABSR		<div><div>S</div><div>D₁</div><div>D₂</div></div>			Absolute position read										PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps DABSR: 13 steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F				
S	*	*	*	*															
D ₁		*	*	*															
D ₂								*	*	*	*	*	*	*					

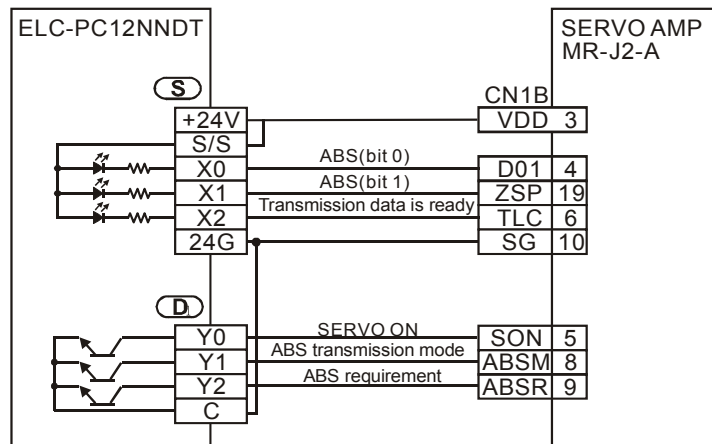
PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: Input signal from Servo (occupies 3 continuous devices) **D₁:** Control signal for controlling Servo (occupies 3 continuous devices) **D₂:** Absolute position data (32 bit) read from Servo (occupies 2 continuous devices)

Explanations:

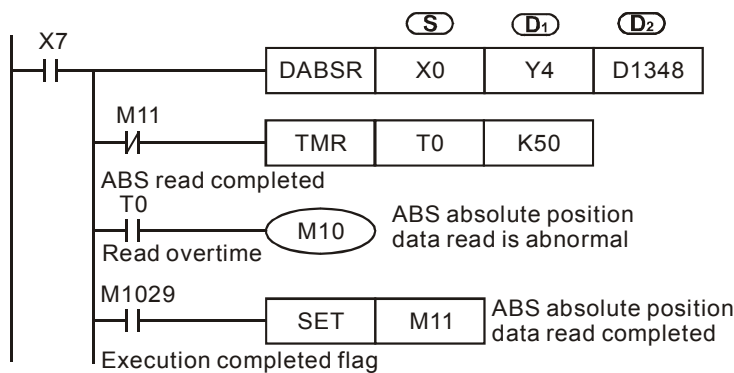
1. This command provides continuous absolute position data read function of Mitsubishi servo drive MR-J2 (with absolute position check function).
2. There is no 16-bit command for API 155, only 32-bit command, DABSR is available and it can only be used for ONCE in program.
3. Flag: For the description of M1010, M1029, M1030, M1334, M1335, M1336, M1337, M1346, please refer to the Notes.
4. **S** is the input signal from Servo and it will use 3 continuous devices **S**, **S + 1**, **S + 2**. Device **S** and **S + 1** are connected to the ABS (bit0, bit1) of Servo for data transmitting. Device **S + 2** is connected to Servo for transmitting data ready flag.
5. **D₁** is the control signal for controlling Servo and it will use 3 continuous devices **D₁**, **D₁+1**, **D₁+2**. Device **D₁** is connected to Servo ON (SON) of Servo, device **D₁+1** is connected to ABS data transmitting mode of Servo and **D₁+2** is connected to ABS data request signal.



6. D_2 is the absolute position data (32 bit) read from Servo and it will use 2 continuous devices D_2 , D_2+1 . D_2 is low word and D_2+1 is high word. The absolute position data should be stored in the current value registers (D1348, D1349) corresponding to CH0 pulse or the current value registers (D1350, D1351) corresponding to CH1 pulse, so it is recommended to specify these two registers. If specify other devices, finally, the user still have to transmit the data into the current value registers (D1348, D1349) corresponding to CH0 pulse or the current value registers (D1350, D1351) corresponding to CH1 pulse.
7. When DABSR command drive contact turns ON and read starts, the command execution completed flag M1029, M1030 will be ON. The flags must be reset by user.
8. When driving the DABSR command, please specify normally open contact. If the drive contact of DABSR command turns OFF when DABSR command read starts, the execution of absolute current value read will be interrupted and result in incorrect data. Please be careful and notice that.
9. If the drive contact of DABSR command turns OFF after the read is completed, the Servo ON (SON) signal connected to D_1 will also turn OFF and the operation will be disabled.

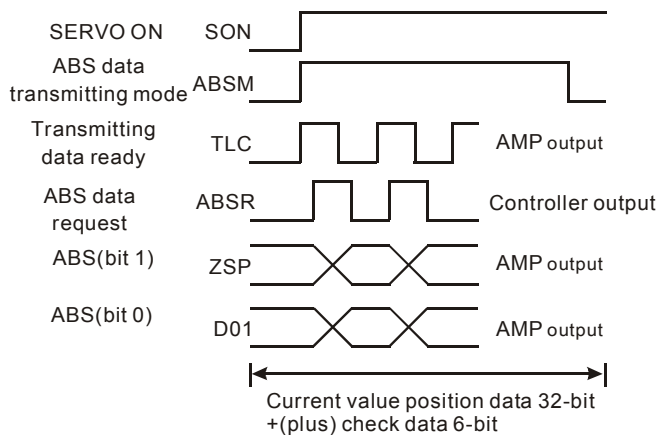
Program Example:

1. When X7= ON, the absolute position data (32 bit) read from Servo should be stored in the current value register (D1348, D1349) corresponding to CH0 pulse. At the same time, drive a timer T0 to count 5 second. If over 5 second and the absolute position data (32 bit) read not complete, it will drive M10=ON and this means the absolute position data (32 bit) read is abnormal.
2. When connecting to system, please set the power of ELC and SERVO AMP to be ON at the same time or set the SERVO AMP to be ON earlier than the power of ELC.

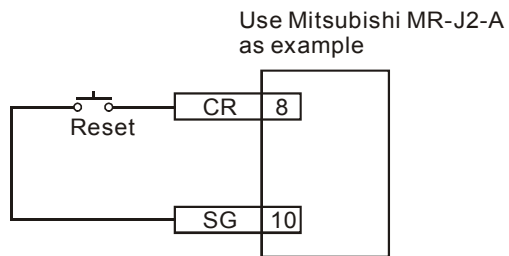


Points to notes:

1. Time chart explanation of DABSR command absolute position data read:



2. When DABSR command starts to execute, it will drive the signal of Servo ON (SON) and ABS data transmitting mode to output.
3. By the transmitting data ready flag and ABS request signal can confirm the transmission and receipt of both sides and process the data transmission of current value position data (32 bit) plus check data (6 bit).
4. Data is transmitted by ABS (bit0, bit1) two bits.
5. This command is applicable to the Servo motor equipped with absolute position detect function is connected, such as Mitsubishi MR-J2-A Servo drive.
6. The Servo motor with absolute position detect function should be started at zero point more than one degree revolution and given the reset signal before manufacturing equipments. Please use one of the following methods to proceed the first time zero point return:
7. Complete zero point return by using reset signal function to execute API 156 ZRN command.
8. After using JOG or manual operation to adjust the zero point position of the equipment, input reset signal SERVO AMP. As for the reset signal input, please refer to the external switches diagram below to see if using ELC controller to output. For the detail of the wiring between ELC and Mitsubishi MR-J2-A, please refer to the API 159 DRVA.

**Flags and Special Data registers descriptions:**

1. M1029: M1029=ON after the first group pulse (Y10) pulse output complete or other relative command complete execution.
2. M1030: M1030= ON after the second group pulse (Y11) pulse output complete.
3. D1348, D1349: D1349(HIGH WORD), D1348(LOW WORD) represents the total number of output pulse of Y10.
4. D1350, D1351: D1351(HIGH WORD), D1350(LOW WORD) represents the total number of output pulse of Y11.

API	Mnemonic			Operands				Function				Controllers			
156	D	ZRN		S₁	S₂	S₃	D	Zero Return				PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DZRN: 17 steps
S ₁					*	*	*	*	*	*	*	*	*	*	*	
S ₂					*	*	*	*	*	*	*	*	*	*	*	
S ₃	*															
D		*														

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Zero return speed **S₂**: Creep speed **S₃**: Near point signal (DOG) **D**: Pulse output device

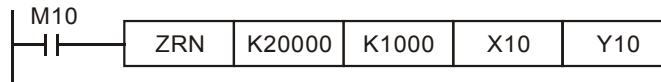
Explanations:

- S₁** is specified as the zero point return speed, the setting range of 32-bit is from 100 to 100,000Hz. When designated speed is greater(less) than max(min) setting range, max(min) setting range shall prevail.
- S₂** is specified as the creep speed(start and end frequency), the lower speed after near point signal (DOG) turns ON and its available range is 100 to 100,000Hz.
- S₃** is specified as the near point signal (DOG) input (A contact input) and only can be specified as X10 and X11.
- Pulse output device **D** only can be specified as Y10, Y11.
- D1343 (D1353) is the accel/decel time setting of Y10 (Y11) and its available range is 50 to 20000ms. If setting exceeds 20,000 ms, it will be regarded as 20,000 ms and in the same way if setting is less than 50 ms, it will be regarded as 50 ms.
- When executing API 158 DRVI and API 159 DRVA instruction, the ELC stores the current pulse value during operation in the registers (Y10: D1348, D1349, Y11: D1350, D1351) so that it can always know the machine position. However, the data may lose when the power of ELC is OFF. Therefore, the machine should execute zero point turn during initial operation to input the zero point data.
- Flag: For the description of M1010, M1029, M1030, M1334, M1335, M1336, M1337, M1346, please refer to the instruction ABSR (API 155).

Program Example:

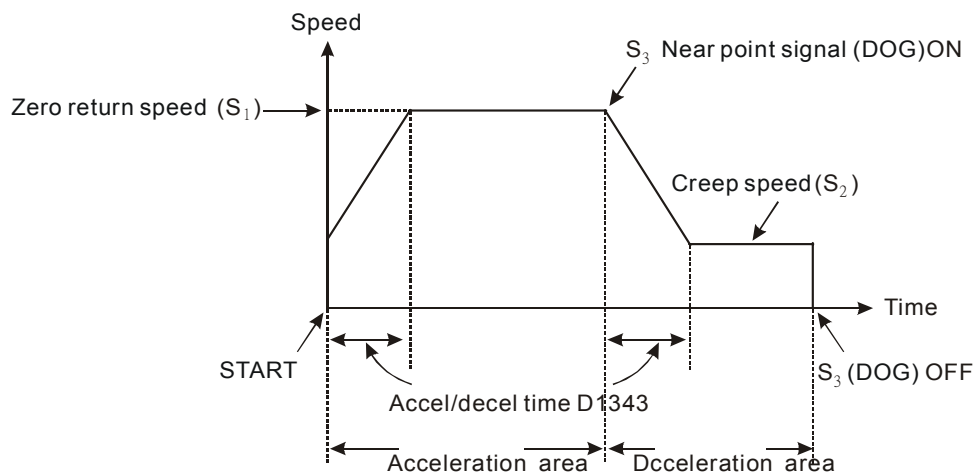
When M10=ON, a frequency of 20KHz outputs from Y10 to make motor execute the action of zero point return. When it reaches the near point signal (DOG), X10=ON and it will change to creep speed. Then, a frequency of 1KHz outputs from Y10 and the instruction will be energized. Pulses output will stop until

X10=OFF.



Explanation of zero point return operation:

1. When ZRN instruction is executed, accelerate to Zero point return speed S_1 and start to move.
(The 1st step speed and the last step speed is set by Y10(D1340) and Y11(D1352). The accel/decel time is set by Y10(D1343) and Y11(D1353)).
2. When the Near point signal (DOG) goes from OFF to ON, it will decelerate to Creep speed S_2 according to the accel/decel time setting (D1343 and D1353).
3. When the Near point signal (DOG) goes from ON to OFF and pulse output stops, 0 will be wrote into the Y10 pulse current value (D1348, D1349) or Y11 pulse current value (D1350, D1351).
4. When the operation of pulse output is completed, flag M1029, M1030 will be activated.
5. Hence, the command can not search the position of Near point signal (DOG) and the operation of ZRN command only can be processed in unidirection. In the operation of ZRN command, the pulse value register (D1348, D1349) of Y10 or the pulse value register (D1350, D1351) of Y11 pulse will decrease.



6. This command is applicable to the Servo motor equipped with absolute positioning function, such as Mitsubishi MR-J2-A Servo drive. It can record current position even the power is OFF. Besides, because the current position of the servo drive can be read by API 155 ABSR command of ELC-PH series, the ZRN command should only be executed for one time. After the power is OFF, it is unnecessary to execute the ZRN command again.
7. When Y10 and Y11 pulse execute the ZRN command, the current value of pulse output frequency will display in (D1348, D1349) and (D1350, D1351). After the operation of ZRN command is completed, 0 will be stored in (D1348, D1349) and (D1350, D1351).

8. When the drive contact of ZRN command is ON, Y10(Y11) pulse will read the content value set by D1343(D1353) as acceleration/deceleration time. After accelerating to zero point return speed, wait for the entry of the near point signal (DOG) and output the creep speed of low speed by decelerating. Immediately stop output pulse when the near point signal (DOG) turns OFF.
9. This instruction can be used many times in the program, but only one instruction will be activated when every ELC program execution. For example, if Y10 is activated, other instructions that use the same output as Y10 won't be executed. Therefore, instructions are executed in the same order as they are activated.
10. When user designates Y10 as output device, user can choose X10 or X11 as the near point signal input of the conversion from acceleration to deceleration. In the same way, when Y11 is designated as output device, X10 or X11 can also be chose as near point signal input.
11. Because no output number comparison in this instruction, the conversion condition (from OFF to ON) must be inputted from near point signal when using Y10. Otherwise, the instruction can't be converted from acceleration to deceleration. Moreover, the trigger time should be more than 10us, otherwise, it will be regarded as noise with no responses.
12. When instruction is in deceleration and output frequency reaches creep speed(end frequency), output action will stop according to the near point signal is from ON to OFF.
13. The current accumulated output number of Y10 is stored in D1348 and D1349. The current accumulated output number of Y11 is stored in D1350 and D1351. It won't be cleared to 0 when program is from STOP to RUN or from RUN to STOP.
14. When M1029=ON, it indicates Y10 pulse output ends. In the same way, if M1030=ON, it indicates that Y11 pulse output ends.
15. During instruction execution, all parameters cannot be modified until instruction execution is completed.
16. When instruction is OFF, all outputs will stop no matter what output it is.

API	Mnemonic		Operands				Function								Controllers			
158	D	DRVI	S₁	S₂	D₁	D₂	Relative Position Control								PB	PC	PA	PH

Type OP		Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DDRVI: 17 steps		
S ₁					*	*	*	*	*	*	*	*	*	*	*	*			
S ₂					*	*	*	*	*	*	*	*	*	*	*	*			
D ₁			*																
D ₂			*	*	*														

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Numbers of pulses (Target device) **S₂**: Pulse output frequency **D₁**: Pulse output device

D₂: Rotation direction signal

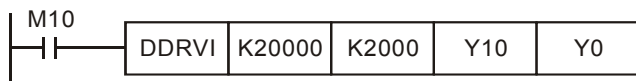
Explanations:

- S₁** is specified as the numbers of pulses (relative position). The available numbers of **S₁** is:
32-bit command: -2,147,483,648 to +2,147,483,647. The positive (+) and negative (-) symbol indicates the forward and reverse direction.
- S₂** is specified as the pulse output frequency. The available numbers of **S₂** is:
32-bit command: 100 to 100,000Hz
- D₁** is specified as pulse output designation device. In PH series models, it only can be specified as Y10 and Y11.
- D₂** is specified as rotation direction signal and it operates following the polarity of **S₁**. When **S₁** is negative (-), **D₂** is OFF. When **S₁** is positive (+), **D₂** is ON and **D₂** won't be OFF immediately after ending pulse output. **D₂** will be OFF when contact switch is OFF.
- The numbers of pulses will be stored in current value register (D1348 high byte, D1349 low byte) of CH0(Y10) pulse or current value register (D1350 high byte, D1351 low byte) of CH1(Y11) pulse. When rotation direction is negative, the content value of current value register will decrease. The content value doesn't change when program is from STOP to RUN or from RUN to STOP.
- The contents of each operand cannot be changed while the DRVI command is executed. The contents will be changed when the next execution is driven.
- If the drive contact turns OFF when the DRVI command is executed, the machine will decelerate to stop and the execution completed flag M1029 and M1030 turn ON.
- D1343 (D1353) is the accel/decel time setting of Y10 (Y11) and its available range for PH MPU is 50 to 20,000ms. If setting exceeds 20,000 ms, it will be regarded as 20,000 ms and in the same way if setting is less than 50 ms, it will be regarded as 50 ms.

9. D1340 (D1352) is Y10(Y11) start/end frequency setting. If pulse output frequency that designated by S_2 is less or equal to start/end frequency, start/end frequency will be regarded as pulse output frequency to execute.
10. Flag: For the description of M1010, M1029, M1030, M1334, M1335, M1336, M1337, M1346, please refer to the instruction ABSR (API 155).

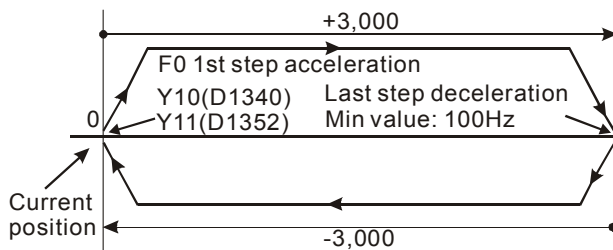
Program Example:

When M10= ON, twenty thousands (20000) of 2KHz frequency pulses outputs from Y10 (relative position). Y0= ON represents the positive direction.

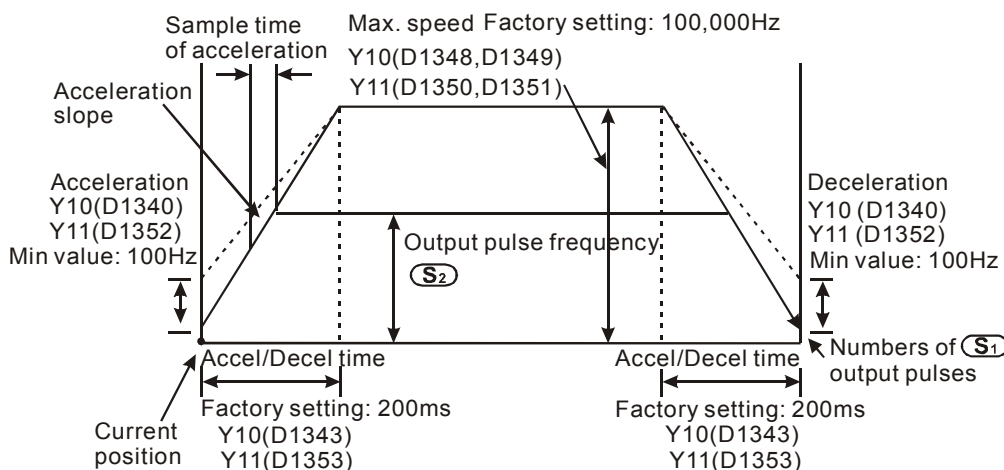


Points to note:

1. Operation explanation of relative position control: Using a positive or a negative symbol to specify travel distance from the current position is also a kind of drive method of relative position control.



2. The setting of relative position and acceleration/deceleration: D1343 (D1353) is the acceleration/deceleration setting of Y10 (Y11) first step acceleration and last step deceleration. D1340 (D1352) is Y10 (Y11) start/end frequency setting.



3. This instruction can be used many times in the program, but only one instruction will be activated when every ELC program execution. For example, if Y10 is activated, other instructions that use the same output as Y10 won't be executed. Therefore, instructions are executed in the same order as they are activated.
4. After Y10 is activated by DDRVI instruction, general Y10 output function will be disabled and so does Y11. The general output function will be enabled after instruction DDRVI is.
5. After activating instruction, all parameters cannot be modified till instruction is OFF.
6. When instruction is OFF but output is not completed, if M1334=ON, Y10 will stop output immediately. If M1334=OFF, Y10 will deceleration to the end frequency by deceleration time and then stop output.
7. Flags description:
 - M1029: For PH series, M1029 will be ON after completing the first pulse CH0 (Y10) output or other related instructions.
 - M1030: For PH series, M1030 will be ON after completing the second pulse CH1 (Y11) output.
 - M1334: For PH series, when M1334=ON, CH0 (Y10) will stop output.
 - M1335: For PH series, when M1335=ON, CH1 (Y11) will stop output.
8. Special register explanation:

D1348, D1349:

For PH series, D1349(HIGH WORD), D1348(LOW WORD) represents the current value that position control instructions (API 156 ZRN, API 158 DRVI, API 159 DRVA) output to the first output Y10. The current value increases or decreases in accordance with the direction of rotation.

D1350 , D1351:

For PH series, D1351(HIGH WORD), D1350(LOW WORD) represents the current value that position control instructions (API 156 ZRN, API 158 DRVI, API 159 DRVA) output to the second output Y11. The current value increases or decreases in accordance with the direction of rotation.

D1340:

For PH series, D1340 operates as the frequency setting of the first step acceleration and last step deceleration of first output Y10 when position control instructions (API 156 ZRN, API 158 DRVI, API 159 DRVA) are executed.

Setting range: 100~100KHz. 100/100KHz will prevail when the setting is less/exceeds 100/100KHz. The factory setting is 200Hz.

Note: When controlling stepping motor, please consider the limit of resonance and start frequency for

speed setting.

D1352:

For PH series, D1352 operates as the frequency setting of the first step acceleration and last step deceleration of second output Y11 when position control instructions (API 156 ZRN, API 158 DRVI, API 159 DRVA) are executed.

Setting range: 100~100KHz. 100/100KHz will prevail when the setting is less/exceeds 100/100KHz. The factory setting is 200Hz.

Note: When controlling stepping motor, please consider the limit of resonance and start frequency for speed setting.

D1343:

For PH series, D1343 operates as the acceleration/deceleration time setting of the first step acceleration and last step deceleration of first output Y10 when position control instructions (API 156 ZRN, API 158 DRVI, API 159 DRVA) are executed.

Setting range: 50~20,000ms. 50/20000 will prevail when the setting is less/exceeds 50/20000ms. The factory setting is 200ms.

Note: When controlling stepping motor, please consider the limit of resonance and start frequency for speed setting.

D1353:

For PH series, D1353 operates as the acceleration/deceleration time setting of the first step acceleration and last step deceleration of second output Y11 when position control instructions (API 156 ZRN, API 158 DRVI, API 159 DRVA) are executed.

Setting range: 50~20,000ms. 50/20000 will prevail when the setting is less/exceeds 50/20000ms. The factory setting is 200ms.

Note: When controlling stepping motor, please consider the limit of resonance and start frequency for speed setting.

API	Mnemonic		Operands				Function								Controllers			
159	D	DRVA	S₁	S₂	D₁	D₂	Absolute Position Control								PB	PC	PA	PH

Type OP		Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E				
	S ₁					*	*	*	*	*	*	*	*	*	*	*			
	S ₂					*	*	*	*	*	*	*	*	*	*	*			
	D ₁		*																
	D ₂		*	*	*														

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Numbers of pulses (Target device) **S₂**: pulse output frequency **D₁**: pulse output device

D₂: Rotation direction signal

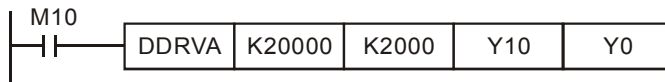
Explanations:

- S₁** is specified as the number of output pulses (absolute position). The setting range of 32-bit instruction: -2,147,483,648 ~ +2,147,483,647. The positive (+) and negative (-) symbol indicate the forward and reverse direction.
- S₂** is specified as the pulse output frequency. The setting range of 32-bit instruction is 100 ~ 100,000Hz.
- D₁** is pulse output device. For PH series, only Y10 and Y11 can be used.
- D₂** is specified as rotation direction signal.
- When **S₁** is larger than current absolute position, **D₂** is OFF.
- When **S₁** is less than current absolute position, **D₂** is ON.
- After pulse outputs, **D₂** won't be OFF immediately. **D₂** will be OFF until the contact is OFF by using instruction.
- The numbers of pulses will be stored in current value register (D1349 high byte, D1348 low byte) of CH0 (Y10) pulse or current value register (D1351 high byte, D1350 low byte) of CH1 (Y11) pulse. When rotation direction is negative, the content value of current value register will decrease.
- If the drive condition turns OFF when the DRVA command is executed, the machine will decelerates and stops and the execution completed flag M1029, M1030 will turn ON.
- D1343 (D1353) is the acceleration/deceleration time of the first step acceleration and last step deceleration of Y10 (Y11). For PH series, the setting range is 50~20,000 ms. If setting exceeds/less than 20,000 ms/50 ms, 20,000 ms/50 ms will prevail.
- D1340 (D1352) is start/end frequency setting of Y10 (Y11). If pulse output frequency that designated by **S₂** is less or equal to start/end frequency, start/end frequency will be regarded as pulse output frequency.

12. Flags: Please refer to API 158 DRVI for M1010, M1029, M1030, M1334, M1335, M1336, M1337, M1346

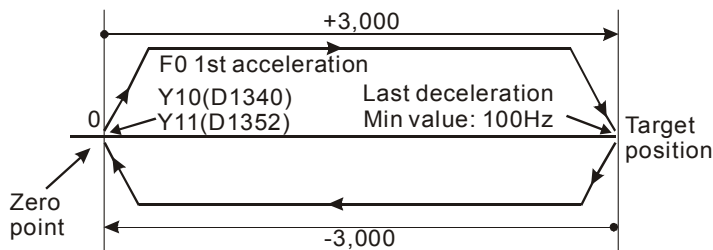
Program Example:

When M10=ON, it will output 20,000 pulses from Y10 with 2KHz frequency. Y0=ON indicates positive direction.



Points to note:

Operation explanation of absolute position control: Specifying travel distance from a zero point is also a kind of drive method of absolute position control.



Settings of absolute position and operation speed: (D1343 (D1353) is the acceleration/deceleration time setting of the first step acceleration and last step deceleration of Y10(Y11). D1340 (D1352) is start/end frequency setting of Y10 (Y11)).

When this instruction is OFF and output is not completed:

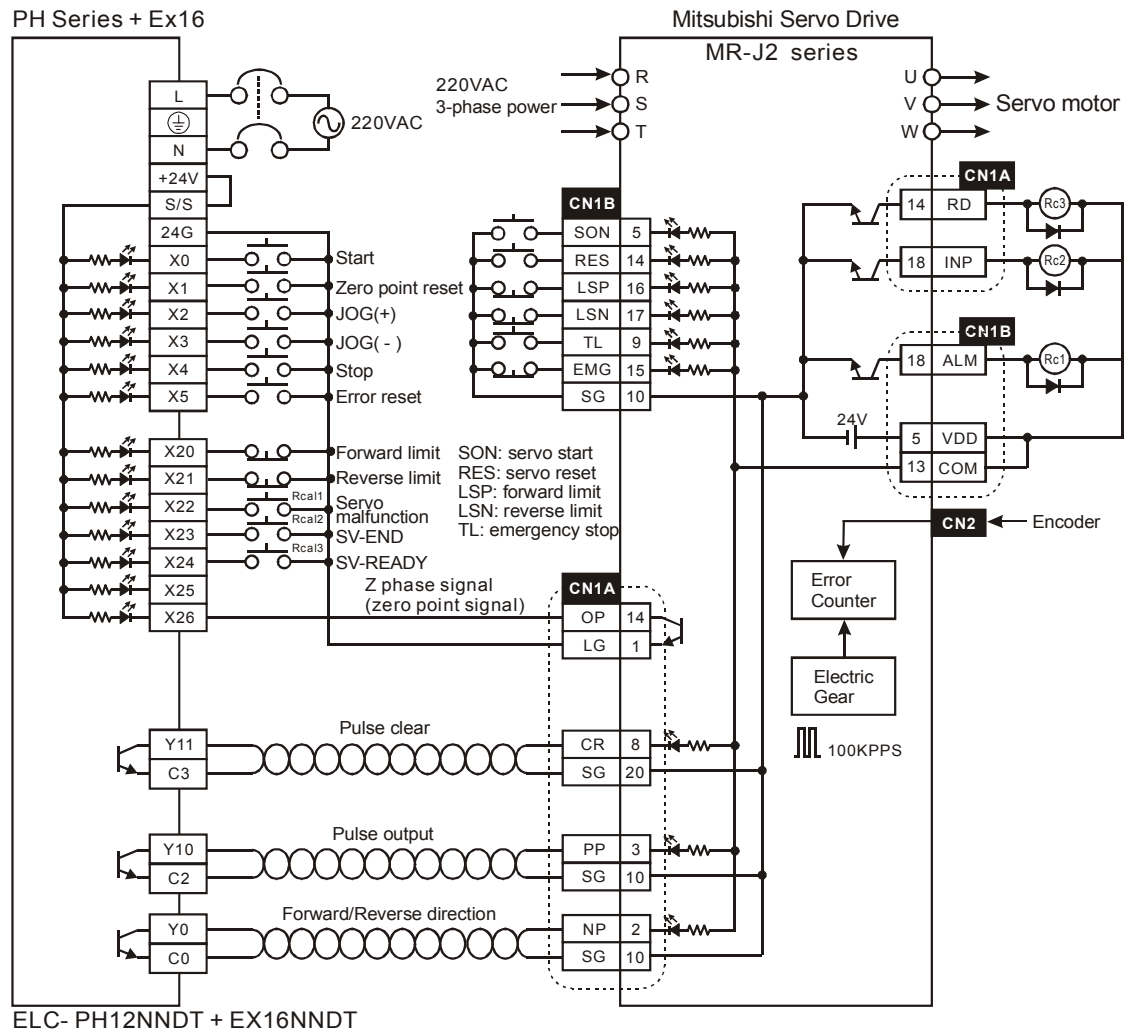
If M1334=ON, Y10 will stop output immediately.

If M1334=OFF, Y10 will decelerate from deceleration time to end frequency and then stop pulse output.

It is the same way for M1335 corresponds to Y11 output.

Please refer to instruction DDRVI Remarks for flag information.

Wiring of ELC-PH12NNDT series and Servo Drive



Notes:

1. The parameter setting of Servo drive should be set: position mode and pulse input type should be Pulse+DIR.
2. Please connect forward/reverse limit switch to SERVO AMP.

API	Mnemonic		Operands					Function					Controllers			
160	TCMP	P	(S ₁)	(S ₂)	(S ₃)	(S)	(D)	Calendar Compare					PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps	
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	TCMP, TCMPP: 11 steps
S ₁						*	*	*	*	*	*	*	*	*	*	*	
S ₂						*	*	*	*	*	*	*	*	*	*	*	
S ₃						*	*	*	*	*	*	*	*	*	*	*	
S												*	*	*			
D			*	*	*												

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: hour of comparison time, setting range is K0~K23 **S₂**: minute of comparison time, setting range is K0~K59 **S₃**: second of comparison time, setting range is K0~K59 **S**: Current time of calendar (occupies 3 continuous devices) **D**: Comparison result (occupies 3 continuous devices)

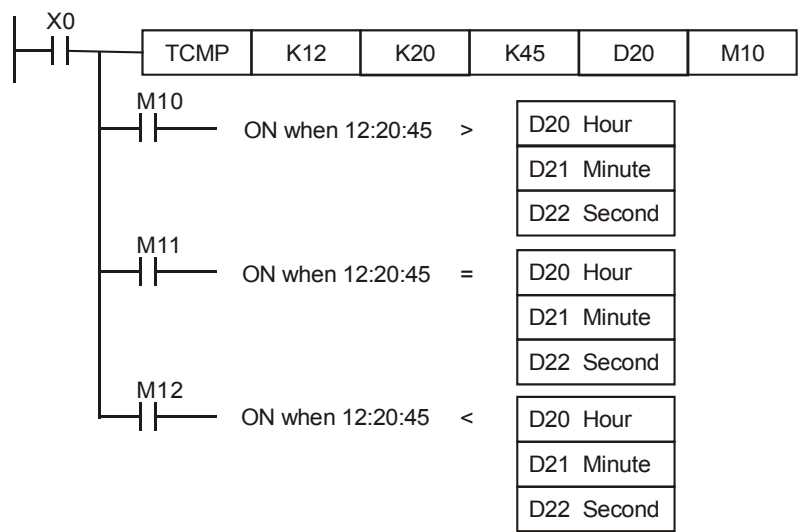
The range of operand S₁, S₂, S₃: S₁=0~23, S₂=S₃=K0~59

Explanations:

1. **S₁, S₂, S₃** is compared to the current value of the clock/calendar **S** and save the comparison result in **D**.
2. **S** is the hour of current time and the content is K0~K23. **S +1** is the minute of current time and the content is K0~K59. **S +2** is the second of current time and the content is K0~K59.
3. The current time of the calendar specified by **S** is read by using the TRD instruction previously and then compared by using TCMP instruction. If the content of **S** exceeds the range, it will result in "operation error". At this time, the instruction won't execute and M1067=ON, M1068=ON, records error code 0E1A (HEX) in D1067.

Program Example:

- 1. When X0= ON, the instruction is executed and the current time of calendar is placed in (D20~D22), is compared to the set value 12:20:45 and the result is shown at M10~M12. When X0 goes from ON→OFF, the instruction is not executed but the ON/OFF state before M10~M12 is unchanged.
- 2. Connect M10~M12 in series or in parallel and then the result of \geq , \leq , \neq are given.



API	Mnemonic		Operands				Function	Controllers			
161	TZCP	P	(S₁)	(S₂)	(S)	(D)	Calendar Zone Compare	PB	PC	PA	PH

Type	Bit Devices				Word devices										Program Steps	
OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	TZCP, TZCPP: 9 steps
S ₁											*	*	*			
S ₂											*	*	*			
S											*	*	*			
D		*	*	*												

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Lower limit time data **S₂**: Upper limit time data **S**: Current time of calendar

D: Comparison result (occupies 3 continuous devices)

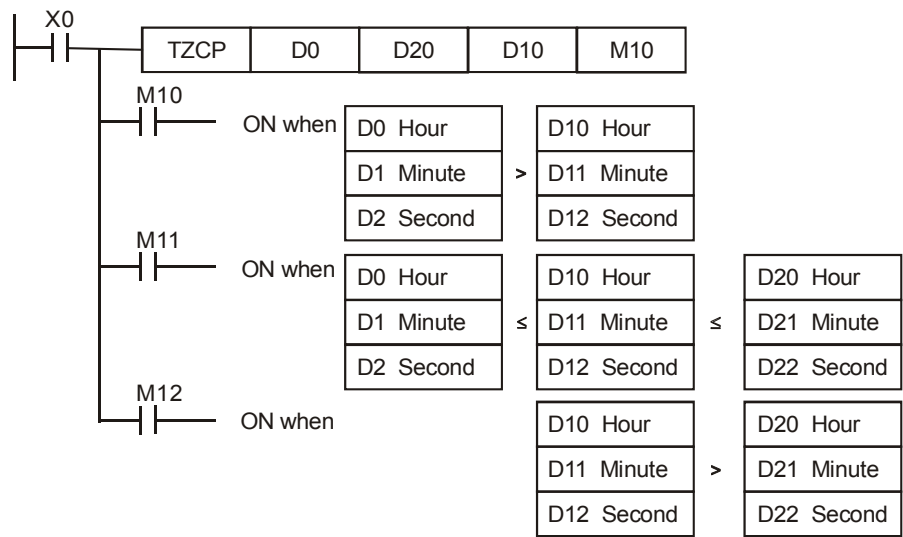
Operand **S₁**, **S₂**, **S** occupies 3 continuous devices **S₁** should be less than **S₂**, i.e. $S_1 \leq S_2$

Explanations:

- S** is compared to the time period of **S₁~ S₂** and the comparison result is stored in **D**.
- S₁**, **S₁ +1**, **S₁ +2**: respectively represent “Hour”, “Minute”, “Second” of the lower limit time data.
- S₂**, **S₂ +1**, **S₂ +2**: respectively represent “Hour”, “Minute”, “Second” of the Upper limit time data.
- S**, **S +1**, **S +2**: respectively represent “Hour”, “Minute”, “Second” of the current time of calendar.
- The current time of the calendar specified by **S** is read by using TRD instruction previously and then compared by using TZCP instruction. If the content of **S**, **S₁**, **S₂** exceeds the range, it will result in “operation error”. At this time, the instruction won't be executed and M1067=ON, M1068=ON, records error code 0E1A (HEX) in D1067.
- If **S** < **S₁** and **S** < **S₂**, **D** is ON. If **S** > **S₁** and **S** > **S₂**, **D+2** is ON. Besides these two situations, **D +1** is ON. (Lower limit **S₁** should be less than upper limit **S₂**.)

Program Example:

When X0= ON, the instruction is executed and one of M10~M12 = ON. When X0=OFF, the instruction is not executed but the state of M10~M12 before X0=OFF is unchanged.



API	Mnemonic		Operands			Function										Controllers											
162		TADD	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Calendar Data Addition										PB	PC	PA	PH							
OP	Type	Bit Devices				Word devices										Program Steps											
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	TADD, TADDP: 7 steps										
	S ₁											*	*	*													
	S ₂											*	*	*													
	D											*	*	*													
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

API	Mnemonic			Operands			Function										Controllers										
163		TSUB	P	<div>S₁</div>	<div>S₂</div>	<div>D</div>	Calendar Data Subtraction										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>							
Type		Bit Devices				Word devices										Program Steps											
OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	TSUB, TSUBP: 7 steps											
S ₁											*	*	*														
S ₂											*	*	*														
D											*	*	*														
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: Time Minuend **S₂**: Time Subtrahend **D**: Subtraction result

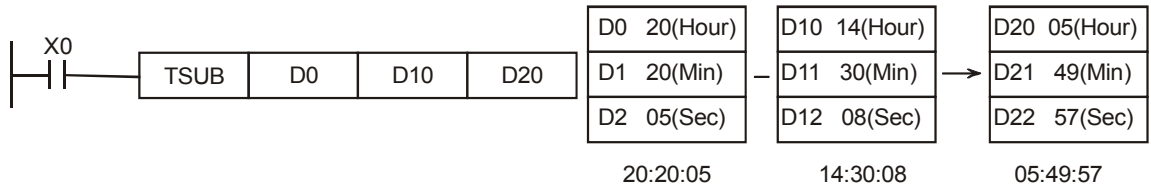
Operand **S₁**, **S₂**, **D** occupies 3 continuous devices.

Explanations:

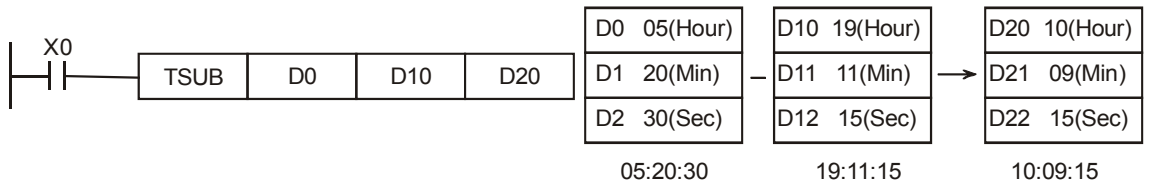
1. **S₁ – S₂ = D**. The time data in the register specified by **S₂** is subtracted from the time data in the register specified by **S₁** and the result is stored in the register specified by **D**.
2. If the time data in **S₁**, **S₂** exceeds the range, it will result in “operation error”. At this time, the instruction won't be executed and M1067=ON, M1068=ON, records error code 0E1A (HEX) in D1067.
3. If the subtraction result is a negative value (less than 0), the Zero flag M1020= ON. The value of the result shows in **D** is the time remaining below 0 (zero) hour.
4. If the subtraction result is equal to 0 (zero, 0 hour, 0 minute, 0 second), the Zero flag M1020= ON.
5. Besides using TRD instruction, MOV instruction can also be used to move the special register D1315 (Hour), D1314 (Minute), D1313 (Second) to the three registers specified to read the current time of calendar.

Program Example:

When X0= ON, the instruction is executed. The time data specified by D10~D12 is subtracted from the time data specified by D0~D2 and the result is stored in the register specified by D20~D22.



If the subtraction result is a negative value (less than 0), the borrow flag M1021= ON.



API	Mnemonic			Operands	Function											Controllers											
166		TRD	P	(D)	Calendar Data Read											PB	PC	PA	PH								
Type OP	Bit Devices				Word devices											Program Steps											
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F												
D												*	*	*													
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

D: The device stores the current time of calendar (occupies 7 continuous devices)

Explanations:

1. The calendar clock, year (A.D.), week, month, date, hours, minutes and seconds, total 7 data devices stored in D1319~D1313. The TRD instruction reads the current time of calendar and stores the values in the 7 data registers specified by **D**.
2. D1319 is read as a two right digit number of year and can be changed to a four digit number to see the notes.
3. Flags: M1016, M1017, M1076, please see the Notes.

Program Example:

When X0=ON, read the current time of calendar to the specified register D0~D6.

The content of D1318: 1 is Monday, 2 is Tuesday,..., 7 is Sunday.



Special D device	Meaning	Content		General D device	Meaning
D1319	Year (A.D.)	00~99	→	D0	Year (A.D.)
D1318	Day (Mon.~Sun.)	1~7	→	D1	Day (Mon.~Sun.)
D1317	Month	1~12	→	D2	Month
D1316	Date	1~31	→	D3	Date
D1315	Hour	0~23	→	D4	Hour
D1314	Minute	0~59	→	D5	Minute
D1313	Second	0~59	→	D6	Second

Notes:

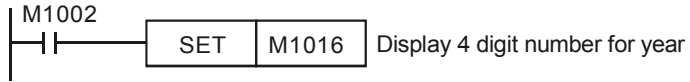
1. There are two methods to correct built-in API calendar:
 - a) Specified instruction to correct
Please refer to instruction TWR (API 167) for reference.

b) Setting by peripheral

Using ELCSoft (software to edit ladder diagram) to set

2. Display four digit number of year:

- a) It usually displays 2 digit number of year (for example: only display 03 for year 2003). If you want to display 4 digit number, please key in following program at the start of program.



- b) It will display 4 bits (two right-most digit number + 2000) to replace original 2 digit number.
- c) If you want to write new time setting in 4 digit number display mode, only 2 digit number you can write in and its range is "00-99" which corresponds to year "2000-2099". For example, 00=year 2000, 50=year 2050 and 99=year 2099.
- d) Error Flag of the calendar:

Device	Name	Function
M1016	year display of calendar	It displays 2 right-most digit number of year of D1319 when it is OFF. It displays (2000+ 2 right-most digit number of year of D1319) when it is ON.
M1017	±30 seconds correction	It will correct when it is from OFF→ON. (if it is 0-29 seconds, it will reset to 0. If it is 30-59 seconds, add 1 to minute and set 0 to second)
M1076	calendar fault	It = ON when setting is out of range or run out of battery. (it only check when power is ON)

Device	Name	Range
D1313	Second	0-59
D1314	Minue	0-59
D1315	Hour	0-23
D1316	Day	1-31
D1317	Month	1-12
D1318	Week	1-7
D1319	Year	0-99 (two right-most digit number of year)

API	Mnemonic			Operands	Function											Controllers							
167		TWR	P	D	Calendar Data Write In											PB	PC	PA	PH				
Type OP	Bit Devices				Word devices											Program Steps							
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F								
D												*	*	*									
												PULSE				16-bit				32-bit			
												PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

D: Source for new calendar time (occupies 7 continuous device)

Explanations:

1. This instruction writes the value in **S** into the calendar clock setting a new time.
2. If the time data in **S** exceeds the valid calendar range, it will result in an “operation error”. The instruction won’t execute and M1067=ON, M1068=ON, and records error code 0E1A (HEX) in D1067.
3. Flags: M1016, M1017, M1076
4. Refer to the TRD instruction for more information.

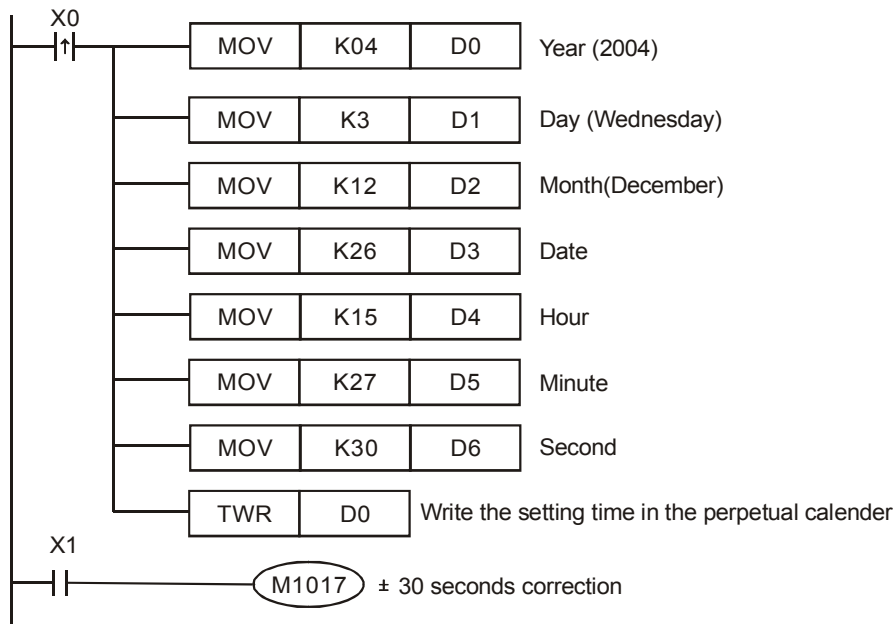
Program Example 1:

When X0= ON, write the new time into the calendar clock.



		General D device	Meaning	Content			Special D device	Meaning
New setting time		D20	Year (A.D.)	00~99	→		D1319	Year (A.D.)
		D21	Day (Mon.~Sun.)	1~7	→		D1318	Day (Mon.~Sun.)
		D22	Month	1~12	→		D1317	Month
		D23	Date	1~31	→		D1316	Date
		D24	Hour	0~23	→		D1315	Hour
		D25	Minute	0~59	→		D1314	Minute
		D26	Second	0~59	→		D1313	Second
</								

seconds of the calendar clock is between 1~29 seconds, the time will be automatically set to “0” (zero) seconds and the minute time won’t change. If the seconds of the calendar clock is between 30~59 seconds, the time will also be automatically set to “0” (zero) second but the minute time will increase by 1 minute.



API	Mnemonic			Operands			Function										Controllers				
169	D	HOUR		<div>S</div>	<div>D₁</div>	<div>D₂</div>	Hour Meter										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	
Type OP	Bit Devices				Word devices										Program Steps						
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	HOUR: 7 steps					
					*	*	*	*	*	*	*	*	*	*	*	D HOUR: 13 steps					
													*								
		*	*	*																	
										PULSE				16-bit				32-bit			
										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

Operands:

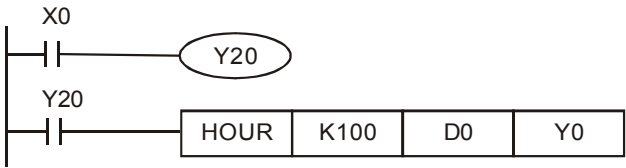
S: Hour set-point value, count hours until they reach this value **D₁:** current time during counting and unit is hour (occupies 2 continuous devices) **D₂:** output device

Explanations:

- D₂** ON contact when hours reach **S** on and unit is hour. range is K1~K32,767. **D₁** range is K0~K32,767. **D₁+1** saves current time that less than one hour and unit is second. Its setting range is K0~K3,599.
- If using input contact to be timer, output device = ON when attaining setting time (unit is hour). It can provide user a timer for managing machine operation or maintain.
- After output device is ON, timer will keep on counting.
- When the 16-bit timer counts to its max. value (32,767 hours and 3,599 seconds), it will stop. If you want to recount, **D₁** and **D₁+1** need to clear to 0. When the 32-bit timer counts to its max. value (2,147,483,647 hours and 3,599 seconds), it will stop. If you want to recount, **D₁** - **D₁+2** need to clear to 0.
- Operand S can only use a 16-bit instruction when using device F.
- Instruction HOUR can be used for four times in program.

Program Example 1:

For 16-bit instruction: When X0=ON, Y20 = ON, counting starts. When the time reaches 100 hours, Y0 = ON and D0 will record the current time (unit is hour, but if D0 is less than one hour, unit will be second and its range is 0~3599).



API	Mnemonic			Operands		Function										Controllers			
170	D	GRY	P	S	D	BIN → GRAY CODE										PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	GRY, GRYP: 5 steps		
S					*	*	*	*	*	*	*	*	*	*	*	DGRY, DGRYP: 9 steps		
D								*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

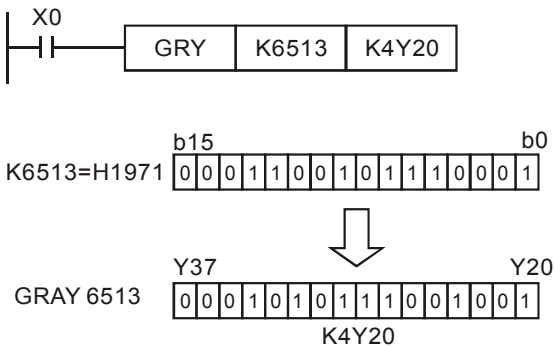
S: Source device **D:** Destination to store Gray code result

Explanations:

1. The BIN value in the specified device by **S** is converted to GRAY CODE equivalent and the converted result is stored in the device specified by **D**.
2. The range of **S** that can be converted to the GRAY CODE is shown as follows:
16-bit instruction: 0~32,767
32-bit instruction: 0~2,147,483,647
3. If the BIN value is outside the range shown above, it is determined as an “Operation Error”. At this time, the instruction won’t be executed, M1067=ON, M1068=ON, and records error code 0E1A (HEX) in D1067.
4. Operands **S** and **D** can only use 16-bit instruction when using F device.

Program Example:

When X0=ON, constant K6513 is converted to GRAY CODE and stored in the K4Y20.



API	Mnemonic			Operands		Function										Controllers			
171	D	GBIN	P	(S)	(D)	GRAY CODE → BIN										PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	GBIN, GBINP: 5 steps DGBIN, DGBINP: 9 steps		
S					*	*	*	*	*	*	*	*	*	*	*			
D							*	*	*	*	*	*	*	*	*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

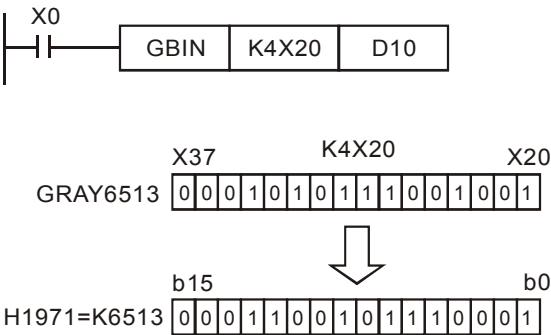
S: Source GRAY CODE D: Destination which stores converted BIN result

Explanations:

1. The GRAY CODE value in specified device **S** is converted to a BIN value equivalent and the converted result is stored in the device specified by **D**.
2. This instruction can be used to read the value from an absolute position type encoder (generally a gray code encoder) which is connected to the ELC inputs. Convert the value to a BIN value and store it in the specified register.
3. Program scan time plus input response time is equal to the output delay time specified by **S**.
4. If the source is set to inputs X20~X37, it can speed up the input response time by using REFF instruction or D1020 (adjust input response time).
5. The range of **S** that can be converted to the GRAY CODE is shown as follows:
16-bit instruction : 0~32,767
32-bit instruction : 0~2,147,483,647
6. If the GRAY CODE value is outside the range shown above, it is determined as "Operation Error".

Program Example:

When X0=ON, the GRAY CODE value in the absolute position type encoder connected to X20~X37 inputs is converted to BIN value and stored in D10.



API	Mnemonic			Operands				Function	Controllers			
180		MAND	P	(S ₁)	(S ₂)	(D)	(n)	Matrix AND	PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MAND, MANDP: 9 steps
S ₁							*	*	*	*	*	*	*			
S ₂							*	*	*	*	*	*	*			
D								*	*	*	*	*	*			
n					*	*							*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: matrix source device 1 S₂: matrix source device 2 D: Area where calculated result is stored

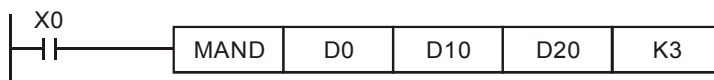
n: matrix length (n=K1~K256)

Explanations:

1. Do matrix AND operation to matrix source device 1 and 2 by length of **n** and save the result in **D**.
2. The operation rule of matrix AND is: bit is 1 when 2 bits are all 1 otherwise it is 0.
3. If use KnX, KnY, KnM, KnS bit device in **S₁**, **S₂**, **D** operands, only indicates n=4.

Program Example:

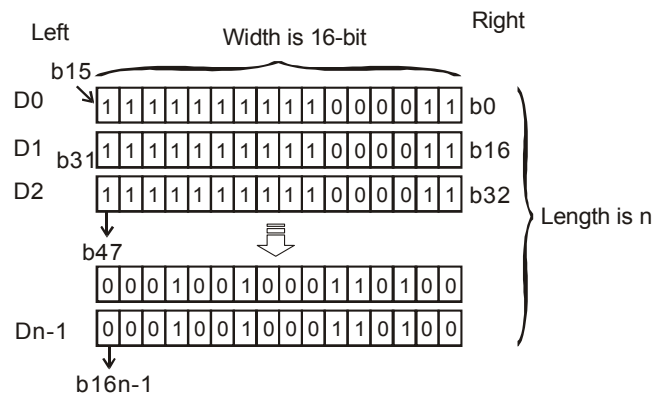
When X0=ON, do MAND and matrix AND operation to 3 rows (D0-D2) of 16-bit register and 3 rows (D10-D12) of 16-bit register. Then save the result in 3 rows (D20-D22) of 16-bit register.



Before Execution		(S ₁) D0	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1
		D1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1
		D2	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1
		MAND															
		(S ₂) D10	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0
After Execution	(D)	D20	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
		D21	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0
		D22	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0

Points to note:

1. A matrix is made up of 1 and above continuous 16-bit registers. The register number that made up matrix is called matrix length **n**. There are 16 X **n** bits (dots) for a matrix and a bit (dot) once for a operand unit.
2. 16 X **n** bits (serial number $b_0 - b_{16n-1}$) will be regarded as a set of a serial single point for matrix instruction. Thus, operate with a specific point in the set not value.
3. The matrix instruction is convenient and important application instruction for dealing with single point to multi-points or multi-point to multi-point, such as move, copy, compare, search, etc.
4. It usually needs a 16-bit register to designate one point of 16 X **n** points during matrix operation. This register calls Pr (pointer). The setting range is 0 – 16**n**-1 and correspond to $b_0 \sim b_{16n-1}$ in matrix individually.
5. There are actions: shift left, shift right or rotate during operation. Large number is defined to left and small number is defined to right as shown in the following.



Fixed width of matrix is 16-bit.

Pr: matrix pointer. If Pr is 15, it means designated point is b15.

Matrix length is **n** and **n** is 1-256.

Example: The matrix that is made up of D0 and **n**=3, D0=HAAAA, D1=H5555, D2=HAAFF

	C ₁₅	C ₁₄	C ₁₃	C ₁₂	C ₁₁	C ₁₀	C ₉	C ₈	C ₇	C ₆	C ₅	C ₄	C ₃	C ₂	C ₁	C ₀	
R ₀	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	D0
R ₁	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	D1
R ₂	1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	D2

Example: The matrix that is made up of K2X20 and **n**=3, K2X20=H37, K2X30=H68, K2X40=H45

	C ₁₅	C ₁₄	C ₁₃	C ₁₂	C ₁₁	C ₁₀	C ₉	C ₈	C ₇	C ₆	C ₅	C ₄	C ₃	C ₂	C ₁	C ₀	
R ₀	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	X ₂₀ ~X ₂₇
R ₁	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	X ₃₀ ~X ₃₇
R ₂	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	X ₄₀ ~X ₄₇

It needs to fill 0 to R0(C₁₅-C₈), R1(C₁₅-C₈), R2(C₁₅-C₈) once the value is empty.

API	Mnemonic			Operands				Function	Controllers			
181		MOR	P	(S₁)	(S₂)	(D)	(n)	Matrix OR	PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MOR, MORP: 9 steps
S ₁							*	*	*	*	*	*	*			
S ₂							*	*	*	*	*	*	*			
D								*	*	*	*	*	*			
n					*	*							*			

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: matrix source device 1 **S₂**: matrix source device 2. **D**: Area where calculated result is stored

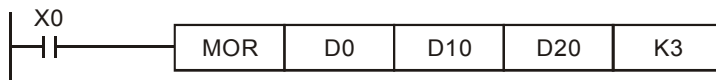
n: matrix length (**n**=K1~K256)

Explanations:

1. Do matrix OR operation to matrix source device 1 and 2 by length of **n** and save the result in **D**.
2. The operation rule of matrix OR is: bit is 1 when one of 2 bits is 1 and only 2 bits are 0 bit will be 0.
3. If use KnX, KnY, KnM, KnS bit device in **S₁**, **S₂**, **D** operands, only indicates n=4.

Program Example:

When X0=ON, do MOR and matrix OR operation to 3 rows (D0-D2) of 16-bit register and 3 rows (D10-D12) of 16-bit register. Then save the result in 3 rows (D20-D22) of 16-bit register.



<div>Before Execution</div>		<div>(S₁)</div>	D0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
			D1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
			D2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		MOR																	
		<div>(S₂)</div>	D10	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0	1
			D11	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0	1
			D12	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0	1
		<div>After Execution</div>		<div>(D)</div>	D20	0	1	0	1	1	1	1	1	1	1	0	1	0	1
D21	0				1	0	1	1	1	1	1	1	1	1	0	1	0	1	
D22	0				1	0	1	1	1	1	1	1	1	1	0	1	0	1	

API	Mnemonic			Operands				Function	Controllers			
183		MXNR	P	S₁	S₂	D	n	Matrix XNR	PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MXNR, MXNRP: 9 steps	
S ₁							*	*	*	*	*	*	*				
S ₂							*	*	*	*	*	*	*				
D								*	*	*	*	*	*				
n					*	*							*				

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: matrix source device 1 S₂: matrix source device 2 D: Area where calculated result is stored

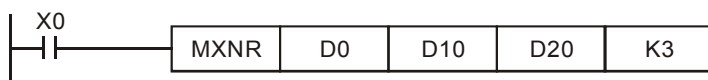
n: matrix length (K1~K256)

Explanations:

1. Do matrix XNR operation to matrix source device 1 and 2 by length of **n** and save the result in **D**.
2. The operation rule of matrix XNR is: bit is 1 when 2 bits are the same otherwise it is 0.
3. If use KnX, KnY, KnM, KnS bit device in **S₁**, **S₂**, **D** operands, only indicates n=4.

Program Example:

When X0=ON, do MXNR and matrix XNR operation to 3 rows (D0-D2) of 16-bit register and 3 rows (D10-D12) of 16-bit register. Then save the result in 3 rows (D20-D22) of 16-bit register.



Before Execution		(S ₁) D0	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		D1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		D2	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		MXNR														
		(S ₂) D10	0	0	0	0	1	1	1	1	1	0	1	0	0	1
		D11	0	0	0	0	1	1	1	1	1	0	1	0	0	1
After Execution	(D)	D20	1	0	1	0	0	1	0	1	0	0	0	1	1	1
		D21	1	0	1	0	0	1	0	1	0	0	0	1	1	1
		D22	1	0	1	0	0	1	0	1	0	0	0	1	1	1

API	Mnemonic			Operands			Function								Controllers						
184		MINV	P	S	D	n	Matrix Inverse								PB	PC	PA	PH			
Type OP	Bit Devices				Word devices										Program Steps						
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MINV, MINVP: 7 steps					
	S						*	*	*	*	*	*	*								
	D							*	*	*	*	*	*								
n					*	*						*									
										PULSE				16-bit				32-bit			
										PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

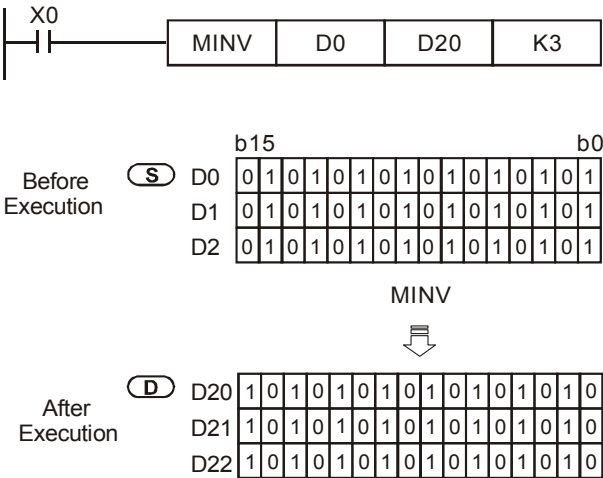
S: Matrix source device D: result n : matrix length (K1~K256)

Explanations:

- 1. Do matrix inverse operation to matrix source device 1 by length of n and save the result in D.
- 2. If use KnX, KnY, KnM, KnS bit device in S, D operands, only indicates n=4.

Program Example:

When X0=ON, do MINV operation to 3 rows (D0-D2) of 16-bit register and 3 rows (D10-D12) of 16-bit register. Then save the result in 3 rows (D20-D22) of 16-bit register.



API	Mnemonic			Operands				Function				Controllers			
185		MCMP	P	(S ₁)	(S ₂)	(n)	(D)	Matrix Compare				PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps	
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MCMP, MCMPP: 9 steps
S ₁							*	*	*	*	*	*	*			
S ₂							*	*	*	*	*	*	*			
n					*	*							*			
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S₁: matrix source device 1 S₂: matrix source device 2 n: matrix length (K1~K256)

D: pointer Pr, save target address

Explanations:

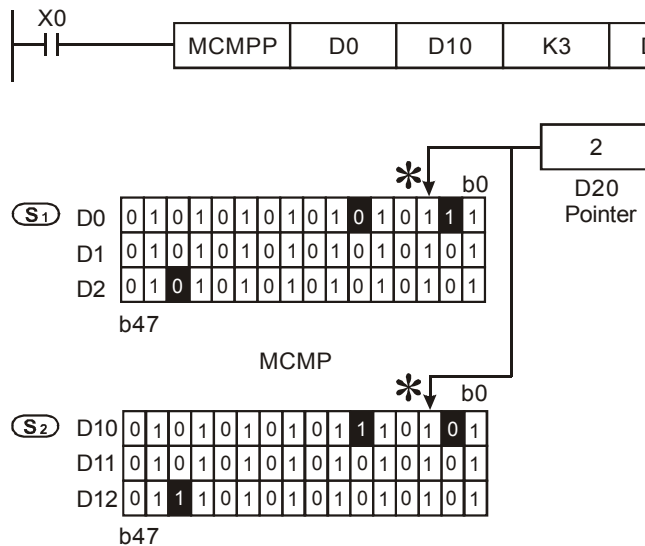
- For each comparison, it will compare each bit of S₁ with S₂ from address Pr. To find the address of different value and save the address in n to complete this comparison.
- You can find the result of comparison from comparison flag M1088. If M1088=1 compare the same value and M1088=0 for difference. Once comparison attains, it will stop comparing immediately and set bit search flag M1091=1. When comparison attains the last bit, matrix search end flag M1089 = ON and comparison attained number is saved in D. For next scan period, it will start comparing from the first bit and set matrix search start flag M1090=1. When D value exceeds the usage range, point error flag M1092 =1.
- It usually needs a 16-bit register to designate one of 16n points in matrix to operate. This register is called pointer, Pr. This is designated by user and the range is 0~16n-1 that correspond to bit b0 ~ b16n-1 individually. You should avoid to change Pr in operation to affect correct comparison search. If Pr value exceeds this range, matrix pointer error flag M1092 will be 1 and this instruction won't be executed.
- Matrix search end flag M1089 and set bit search flag M1091 will be 1 at the same time when occur at the same time.
- If use KnX, KnY, KnM, KnS bit device in S₁, S₂ operands, only indicates n=4.
- Flag: M1088-M1092

Program Example:

When X0 is from OFF→ON, matrix search start flag M1090=0 thus it will start comparing to find the different bit from the bit (* mark) that present value +1. (M1088=0 means difference)

When present value of pointer D20=2, it can get following four results (①, ②, ③, ④) when X0 is

1. D20=5, matrix bit search flag M1091=1, matrix search end flag M1089=0.
2. D20=45, matrix bit search flag M1091=1, matrix search end flag M1089=0.
3. D20=47, matrix bit search flag M1091=0, matrix search end flag M1089=1.
4. D20=1, matrix bit search flag M1091=1, matrix search end flag M1089=0.



Explanation for flag signal:

M1088: Matrix comparison flag, if M1088=1, the result of comparison is the same, M1088=0 otherwise.

M1089: Matrix search end flag, when comparing to the last bit, M1089=1.

M1090: Matrix search start flag, start comparing from the first bit, M1090=1.

M1091: Matrix bit search flag, it will stop comparing once comparison attained, M1091=1.

M1092: Matrix pointer error flag, pointer Pr exceeds that range, M1092=1.

API	Mnemonic		Operands			Function										Controllers				
186		MBRD	P	<div>S</div>	<div>n</div>	<div>D</div>	Matrix Bit Read										<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

Type OP	Bit Devices				Word devices										Program Steps MBRD, MBRDP: 7 steps				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E					F
S							*	*	*	*	*	*	*						
n					*	*							*						
D								*	*	*	*	*	*	*	*				

PULSE				16-bit				32-bit			
<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>	<div>PB</div>	<div>PC</div>	<div>PA</div>	<div>PH</div>

Operands:

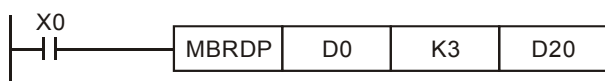
S: matrix source device **n:** matrix length (K1~K256). **D:** pointer Pr, save target address

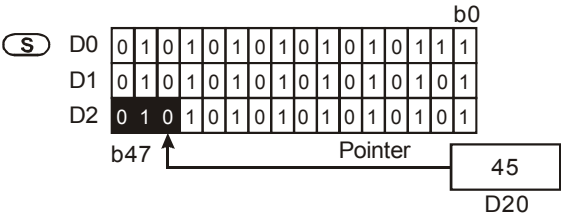
Explanations:

- When executing instruction, it will start to see if M1094 (matrix pointer clear flag) is ON. If it is ON, pointer **D** will be cleared to 0 and read **S** from the 0 bit and read ON/OFF state of each bit to M1095 (matrix rotate/shift/output/carry). It will see if M1093 (matrix pointer increase flag) is ON after reading a bit. And increase 1 to **D** if it is ON. When reading to the last bit, M1089 (matrix search end flag) =ON, pointer **D** records the number of read bit and then end executing this instruction.
- Pr (pointer) is designated by user and the range is 0~16n-1 that correspond to bit b0 ~ b16n-1 individually. If Pr value exceeds this range, matrix pointer error flag M1092 will be 1 and this instruction won't be executed.
- If use KnX, KnY, KnM, KnS bit device in **S** operands, only indicates n=4.

Program Example:

- When X0 is from OFF→ON, pointer clear flag M1094=ON, matrix pointer increase flag M1093=1, and increase 1 to pointer Pr after reading a bit.
- When present value of pointer D20=45, it can get following three results (❶, ❷, ❸) when X0 is executed from OFF→ON for three times.
 - D20=46, matrix rotate/shift/output carry flag M1095=0, matrix search end flag M1089=0.
 - D20=47, matrix rotate/shift/output carry flag M1095=1, matrix search end flag M1089=0.
 - D20=47, matrix rotate/shift/output carry flag M1095=1, matrix search end flag M1089=1.





Explanation for flag signals:

- M1089: Matrix search end flag, when comparing to the last bit, M1089=1.
- M1092: Matrix pointer error flag, pointer Pr exceeds that range, M1092=1.
- M1093: Matrix pointer increase flag, add 1 to present pointer.
- M1094: Matrix pointer clear flag, clear present pointer to 0.
- M1095: Matrix rotate/shift/output carry flag.

API	Mnemonic			Operands			Function										Controllers			
187		MBWR	P	<div>S</div>	<div>n</div>	<div>D</div>	Matrix Bit Write										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices											Program Steps MBWR, MBWRP: 7 steps				
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F					
	S							*	*	*	*	*	*	*							
	n						*	*						*							
	D								*	*	*	*	*	*	*	*					*

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

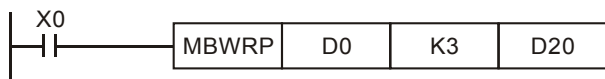
S: matrix source device **n:** matrix length (K1~K256) **D:** pointer Pr, save target address.

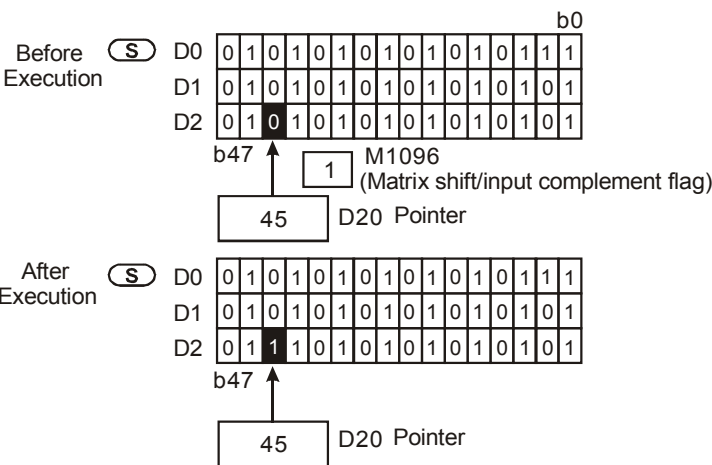
Explanations:

- When executing instruction, it will start to see if M1094 (matrix pointer clear flag) is ON. If it is ON, pointer **D** will be cleared to 0 and write M1096 (matrix shift/input complement flag) in the 0 bit of **S**. It will see if M1093 (matrix pointer increase flag) is ON after writing a bit. And increase 1 to **D** if it is ON. When writing to the last bit, M1089 (matrix search end flag) =ON, pointer **D** records the number of read bit and then end executing this instruction. If **D** exceeds range, M1092=1.
- Pr (pointer) is designated by user and the range is 0~16**n**-1 that correspond to bit b0 ~ b16**n**-1 individually. If Pr value exceeds this range, matrix pointer error flag M1092 will be 1 and this instruction won't be executed.
- If use KnX, KnY, KnM, KnS bit device in **S** operands, only indicates n=4.
- Flags: Please refer to explanation for M1089-M1096

Program Example:

- When X0 is from OFF→ON, pointer clear flag M1094=OFF, matrix pointer increase flag M1093=1, and increase 1 to pointer Pr after writing a bit.
- When present pointer is D20=45, M1096 (matrix shift/input complement flag) =1. When X0 is executed once from OFF→ON, it can get following result:





Explanation for flag signals:

M1089: Matrix search end flag, when comparing to the last bit, M1089=1.

M1092: Matrix pointer error flag, pointer Pr exceeds that range, M1092=1.

M1093: Matrix pointer increase flag, add 1 to present pointer.

M1094: Matrix pointer clear flag, clear present pointer to 0.

M1096: Matrix shift/input complement flag

API	Mnemonic			Operands			Function								Controllers			
188		MBS	P	<div>S</div>	<div>D</div>	<div>n</div>	Matrix Bit Shift								PB	PC	PA	PH

Type OP	Bit Devices				Word devices										Program Steps MBS, MBSP: 7 steps				
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E					F
S							*	*	*	*	*	*	*						
D								*	*	*	*	*	*						
n					*	*							*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

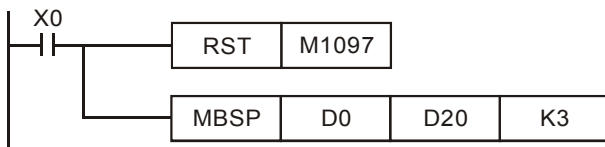
S: matrix source device **D:** result **n:** matrix length (K1~K256)

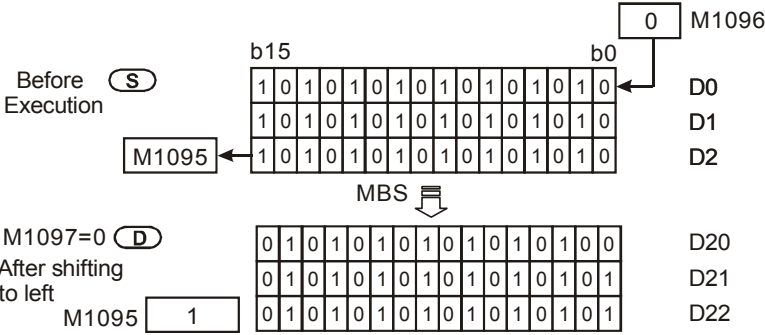
Explanations:

- This instruction is used to shift **S** to left or right by matrix length. M1097=0 moves to left and M1097=1 moves to right. It needs to use the state of M1096 (complement flag) to fill the empty bit (shift to left is b0 and shift to right is b16n-1) due to shift for each bit. If there is one more bit due to shift (shift to left is b16n-1 and shift to right is b0), it needs to send the state to M1095 (carry flag) and save the result in **D**.
- This instruction works best when used as a pulse instruction (MBSP).
- If use KnX, KnY, KnM, KnS bit device in **S**, **D** operands, only indicates n=4.
- Flags:
 - M1095: matrix rotate/shift/output carry flag
 - M1096: matrix shift/input complement flag
 - M1097: matrix rotate/shift direction flag

Program Example 1:

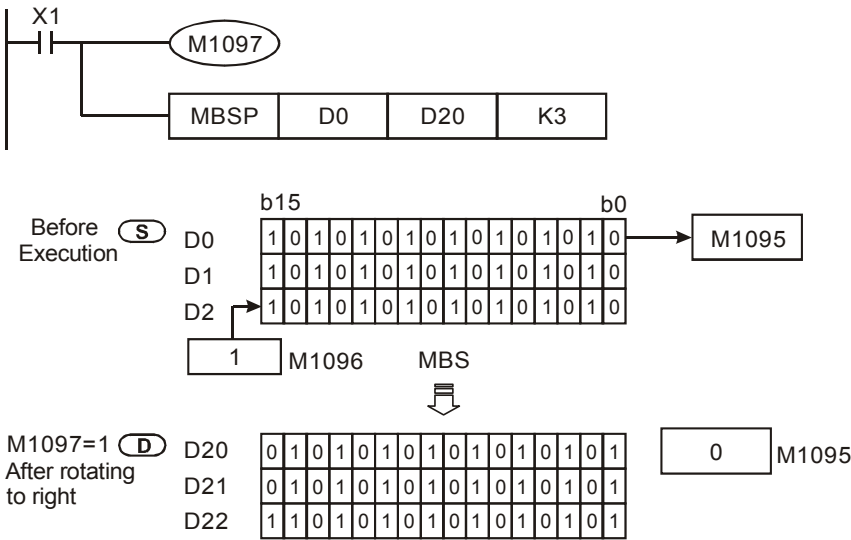
When X0=ON, M1097=OFF means shift matrix to left. Setting complement flag M1096=0, shift 16-bit registers D0~D2 to left and save the result in 16-bit register D20~D22 and carry flag M1095 will be 1.





Program Example 2:

When X1=ON, M1097=ON to shift matrix to right. Setting complement flag M1096=1, shift 16-bit registers D0~D2 to right and save the result to 16-bit registers D20~D22 and carry flag M1095 will be 0.



API	Mnemonic			Operands			Function								Controllers			
189		MBR	P	(S)	(D)	(n)	Matrix Bit Rotator								PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MBR, MBRP: 7 steps			
S							*	*	*	*	*	*	*						
D								*	*	*	*	*	*						
n					*	*							*						

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

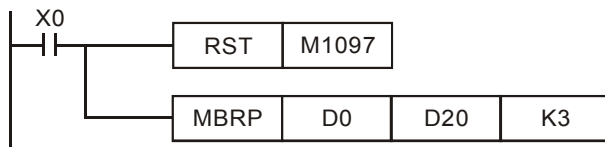
S: matrix source device **D:** result **n:** matrix length (K1~K256)

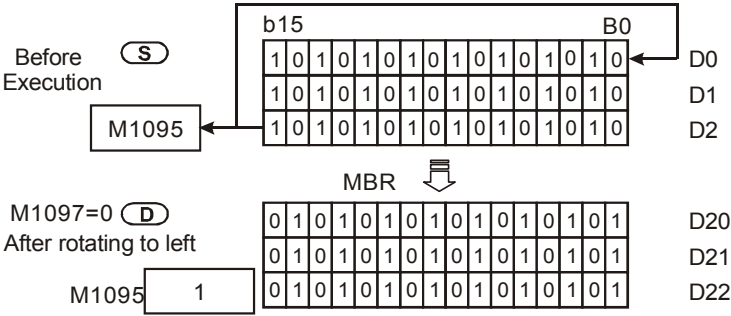
Explanations:

- This instruction is used to rotate **S** to right or left by matrix length. M1097=0 moves to left and M1097=1 moves to right. The empty bit (rotate to left is b0 and shift to right is b16n-1) due to rotation will be filled by the bit (rotate to left is b16n-1 and shift to right is b0) that rotated out and save the result in **D**. The bit that is rotated out is not only used to fill the empty bit but also send its state to carry flag M1095.
- This instruction works best when used as a pulse instruction (MBRP).
- If use KnX, KnY, KnM, KnS bit device in **S**, **D** operands, only indicates n=4.
- Flags:
M1095: matrix rotate/shift/output carry flag
M1097: matrix rotate/shift direction flag

Program Example 1:

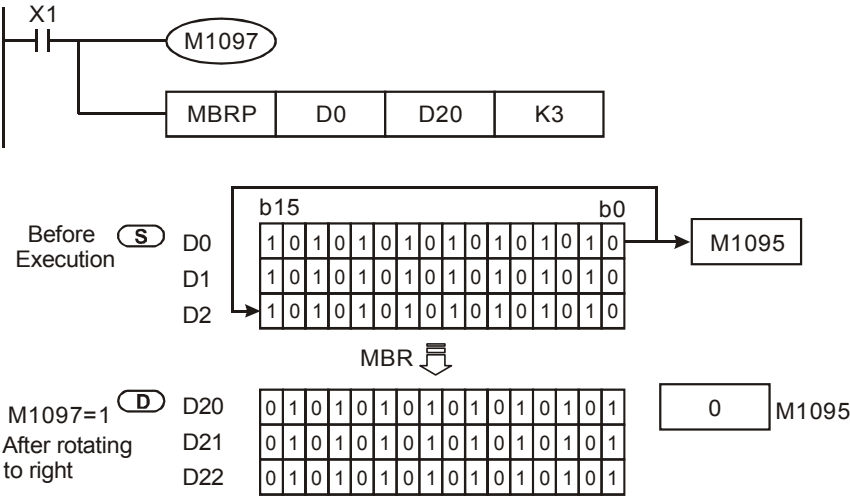
When X0=ON, M1097=OFF means rotate matrix to left. To rotate 16-bit registers D0-D2 to left and save the result in 16-bit register D20-D22. The carry flag M1095 will be 1.





Program Example 2:

When X1=ON, M1097=ON to rotate matrix to right. To rotate 16-bit registers D0~D2 to right and save the result to 16-bit registers D20~D22 . The carry flag M1095 will be 0.



API	Mnemonic			Operands			Function								Controllers			
190		MBC	P	(S)	(n)	(D)	Matrix Bit State Count								PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MBC, MBCP: 7 steps			
S							*	*	*	*	*	*	*						
n					*	*							*						
D								*	*	*	*	*	*	*	*				

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

S: Matrix source device **n:** matrix length (K1~K256) **D:** result

Explanations:

- To count number of bit 1 or bit 0 by matrix length **n** and number in **D**.
- If use KnX, KnY, KnM, KnS bit device in **S** operands, only indicates n=4.
- Flags:

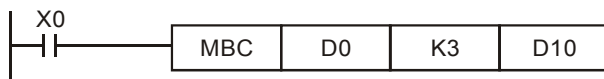
M1098: matrix count by “bit 1” or “bit 0” flag

M1099: it is ON when counted result is 0.

When M1098=1, count the number of bit 1. And count the number of bit 0 when M1098=0. If counting result is 0, M1099=1.

Program Example:

When X0=ON, it counts bit 1 number of D0~D2 and save the total number in D10. When M1098=0, it counts bit 0 number of D0~D2 and save the total number in D10.



D0	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1
D1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1
D2	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1

D10 12 M1098=0

D10 36 M1098=1

API	Mnemonic			Operands		Function										Controllers				
215~217	D	LD#			(S₁)	(S₂)	Logic Contact Operation										PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	LD#: 5 steps			
S ₁					*	*	*	*	*	*	*	*	*	*	*	DLD#: 9 steps			
S ₂					*	*	*	*	*	*	*	*	*	*	*				

												PULSE				16-bit				32-bit			
												PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

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

Explanations:

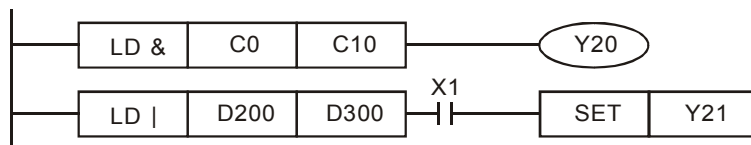
- Compare the contents of S₁ and S₂. Take “LD&” as an example, if the comparison result is NOT 0 the contact is in continuity, and if it is 0, the contact is in discontinuity.
- This instruction must be the first instruction of a rung.

API No.	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
215	LD&	DLD&	S ₁ & S ₂ ≠ 0	S ₁ & S ₂ = 0
216	LD	DLD	S ₁ S ₂ ≠ 0	S ₁ S ₂ = 0
217	LD^	DLD^	S ₁ ^ S ₂ ≠ 0	S ₁ ^ S ₂ = 0

- Operators: & : Logic “AND” operation, | : Logic “OR” operation, ^ : Logic “XOR” operation
- If the 32-bit length counter (C200~) is used with this instruction for comparison, be sure to use the 32-bit instruction (DLD#). If the 16-bit instruction (LD#) is used, a “Program Error”, will occur and the red “ERROR” indicator on the ELC panel will blink.

Program Example:

- If the result of the LD& (Logic “AND” operation) instruction to compare the content of C0 and C10. If the result is not equal to 0, Y20=ON.
- If the result of the LD| (Logic “OR” operation) instruction to compare the content of D200 and D300. If the result is not equal to 0 and X1=ON, set Y21=ON.



API	Mnemonic			Operands		Function										Controllers											
218~220	D	AND#			(S₁)	(S₂)	Series Logic Contact Operation										PB	PC	PA	PH							
OP	Type	Bit Devices				Word devices										Program Steps											
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	AND#: 5 steps											
	S ₁					*	*	*	*	*	*	*	*	*	*		*	DAND#: 9 steps									
S ₂						*	*	*	*	*	*	*	*	*	*	*											
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

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

Explanation:

- Compare the contents of S₁ and of S₂. Take “AND&” as an example, if the comparison result is NOT 0, the contact is in continuity, and if it is 0, the contact is in discontinuity.
- Instruction AND# is used to connect to contact in series.

API No.	16 -bit instruction	32 -bit instruction	Continuity condition	Discontinuity condition
218	AND&	DAND&	S ₁ & S ₂ ≠ 0	S ₁ & S ₂ = 0
219	AND	DAND	S ₁ S ₂ ≠ 0	S ₁ S ₂ = 0
220	AND^	DAND^	S ₁ ^ S ₂ ≠ 0	S ₁ ^ S ₂ = 0

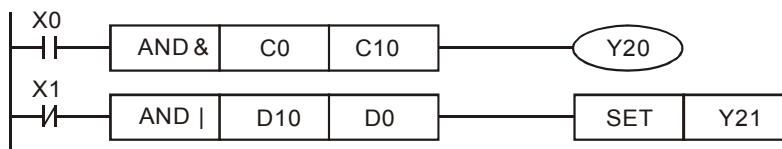
- Operators:

& : Logic “AND” operation, | : Logic “OR” operation, ^ : Logic “XOR” operation

- If the 32-bit length counter (C200~) is used with this instruction for comparison, be sure to use the 32-bit instruction (DAND#). If the 16-bit instruction (AND#) is used, a “Program Error”, will occur and the red “ERROR” indicator on the ELC panel will blink.

Program Example:

- When X0=ON, using the AND& (Logic “AND” operation) instruction to compare the content of C0 and C10. If the result is not equal to 0, Y20=ON.
- When X1=OFF, using the AND| (Logic “OR” operation) instruction to compare the content of D10 and D0. If the result is not equal to 0, set Y21=ON.



API	Mnemonic			Operands		Function										Controllers				
221~223	D	OR#			<div>S₁</div>	<div>S₂</div>	Parallel Logic Contact Operation										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F			
	S ₁					*	*	*	*	*	*	*	*	*	*	*	OR#: 5 steps		
	S ₂					*	*	*	*	*	*	*	*	*	*	*	DOR#: 9 steps		

										PULSE				16-bit				32-bit			
										PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

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

Explanation:

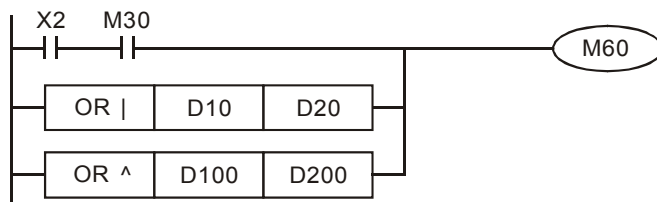
- Compare the contents of S₁ and S₂. Take "OR&" as an example, if the comparison result is NOT 0, the contact is in continuity, and if it is 0, the contact is in discontinuity.
- Instruction OR# is used to connect to contact in parallel.

API No.	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
221	OR&	DOR&	S ₁ & S ₂ ≠ 0	S ₁ & S ₂ = 0
222	OR	DOR	S ₁ S ₂ ≠ 0	S ₁ S ₂ = 0
223	OR^	DOR^	S ₁ ^ S ₂ ≠ 0	S ₁ ^ S ₂ = 0

- Operators:
& : Logic "AND" operation, | : Logic "OR" operation, ^ : Logic "XOR" operation
- If the 32-bit length counter (C200~) is used with this instruction for comparison, be sure to use the 32-bit instruction (DOR#). If the 16-bit instruction (OR#) is used, a "Program Error", will occur and the red "ERROR" indicator on the ELC panel will blink.

Program Example:

If both X2 and M30 are "ON", or when using the OR| (Logic "OR" operation) instruction to compare the content of D10 and D20 and the result is not equal to 0, or when using the OR^ (Logic "XOR" operation) instruction to compare the content of D100 and D200 and the result is not equal to 0, M60=ON.



API	Mnemonic			Operands			Function										Controllers										
224~230	D	LD*		<div><div>S₁</div><div>S₂</div></div> <div>Contact Comparison</div>			PB				PC				PA				PH								
OP	Type	Bit Devices				Word devices										Program Steps											
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	LD*: 5 steps											
S ₁					*	*	*	*	*	*	*	*	*	*	*	DLD*: 9 steps											
S ₂					*	*	*	*	*	*	*	*	*	*	*												
																PULSE				16-bit				32-bit			
																PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

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

Explanations:

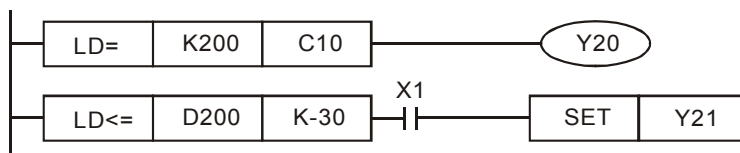
- Compare the contents of S₁ and of S₂. Take “LD=” as an example, if the comparison result is “=”, the contact is in continuity, and if it is “≠”, the contact is in discontinuity.
- This instruction must be the first instruction of a rung.

API No.	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
224	LD=	DLD=	S ₁ =S ₂	S ₁ ≠S ₂
225	LD>	DLD>	S ₁ >S ₂	S ₁ ≤S ₂
226	LD<	DLD<	S ₁ <S ₂	S ₁ ≥S ₂
228	LD<>	DLD<>	S ₁ ≠S ₂	S ₁ =S ₂
229	LD≤	DLD≤	S ₁ ≤S ₂	S ₁ >S ₂
230	LD≥	DLD≥	S ₁ ≥S ₂	S ₁ <S ₂

- When the left most bit, MSB (the 16-bit instruction: b15, the 32-bit instruction: b31), from S₁ and S₂ is 1, this comparison value will be viewed as a negative value for comparison.
- If the 32-bit length counter (C200~) is used with this instruction for comparison, be sure to use the 32-bit instruction (DLD*). If the 16-bit instruction (LD*) is used, a “Program Error”, will occur and the red “ERROR” indicator on the ELC panel will blink.

Program Example:

- If the content of counter C10 is equal to K200, Y20=ON.
- When the content of D200 is smaller or equal to K - 30, and that X1=ON, set Y21=ON.



API	Mnemonic			Operands		Function										Controllers			
232~238	D	AND*			<div>S₁</div> <div>S₂</div>	Series Contact Comparison										PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	AND*: 5 steps DAND*: 9 steps			
S ₁					*	*	*	*	*	*	*	*	*	*	*				
S ₂					*	*	*	*	*	*	*	*	*	*	*				

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

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

Explanations:

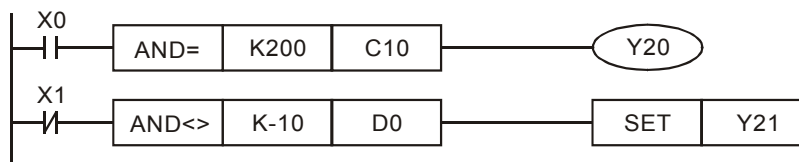
- Compare the contents of S₁ and of S₂. Take “AND=” as an example, if the comparison result is “=”, the contact is in continuity, and if it is “≠”, the contact is in discontinuity.
- Instruction AND* is the comparison instruction that connect to contact in series.

API No.	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
232	AND=	DAND=	S ₁ =S ₂	S ₁ ≠S ₂
233	AND>	DAND>	S ₁ >S ₂	S ₁ ≤S ₂
234	AND<	DAND<	S ₁ <S ₂	S ₁ ≥S ₂
236	AND<>	DAND<>	S ₁ ≠S ₂	S ₁ =S ₂
237	AND≤	DAND≤	S ₁ ≤S ₂	S ₁ >S ₂
238	AND≥	DAND≥	S ₁ ≥S ₂	S ₁ <S ₂

- When the left most bit, MSB (the 16-bit instruction: b15, the 32-bit instruction: b31), from S₁ and S₂ is 1, this comparison value will be viewed as a negative value for comparison.
- If the 32-bit length counter (C200~) is used with this instruction for comparison, be sure to use the 32-bit instruction (DAND*). If the 16-bit instruction (AND*) is used, a “Program Error”, will occur and the red “ERROR” indicator on the ELC panel will blink.

Program Example:

- If X0=ON and the current value of counter C10 equals K200, Y20=ON.
- If X1=OFF and the content of register D0 not equal to K - 10, set Y21=ON.



API	Mnemonic			Operands		Function										Controllers				
240~246	D	OR*			<div>S₁</div>	<div>S₂</div>	Parallel Contact Comparison										PB	PC	PA	PH

Type OP	Bit Devices				Word devices											Program Steps			
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	OR*: 5 steps DOR*: 9 steps			
S ₁					*	*	*	*	*	*	*	*	*	*	*				
S ₂					*	*	*	*	*	*	*	*	*	*	*				

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

Operands:

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

Explanations:

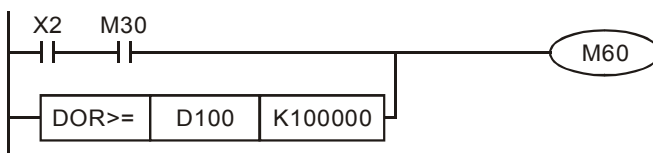
- Compare the contents of S₁ and of S₂. Take “OR=” as an example, if the comparison result is “=”, the contact is in continuity, and if it is “≠”, the contact is in discontinuity.
- Instruction OR* is the comparison instruction that connect to contact in parallel.

API No.	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
240	OR=	DOR=	S ₁ =S ₂	S ₁ ≠S ₂
241	OR>	DOR>	S ₁ >S ₂	S ₁ ≤S ₂
242	OR<	DOR<	S ₁ <S ₂	S ₁ ≥S ₂
244	OR<>	DOR<>	S ₁ ≠S ₂	S ₁ =S ₂
245	OR≤	DOR≤	S ₁ ≤S ₂	S ₁ >S ₂
246	OR≥	DOR≥	S ₁ ≥S ₂	S ₁ <S ₂

- When the left most bit, MSB (the 16-bit instruction: b15, the 32-bit instruction: b31), from S₁ and S₂ is 1, this comparison value will be viewed as a negative value for comparison.
- If the 32-bit length counter (C200~) is used with this instruction for comparison, be sure to use the 32-bit instruction (DOR*). If the 16-bit instruction (OR*) is used, a “Program Error”, will occur and the red “ERROR” indicator on the ELC panel will blink.

Program Example:

If both X2 and M30 are “ON”, or if the contents of the 32-bit registers D101 and D100 are greater or equal to K100,000, M60=ON.



Communications

This chapter contains information regarding the communications ports of the ELC.

This Chapter Contains

7.1	Communication Ports	7-2
7.2	Communication Protocol ASCII mode.....	7-2
7.2.1	ADR (Communication Address)	7-3
7.2.2	CMD (Command code) and DATA (data characters)	7-3
7.2.3	LRC CHK (check sum)	7-5
7.3	ELC Device Address.....	7-8
7.4	Command Code	7-9
7.4.1	Command Code : 01, Read Coil Status.....	7-9
7.4.2	Command Code : 02, Read Input Status	7-10
7.4.3	Command Code : 03, Read Holding Register	7-11
7.4.4	Command Code : 05, Force Single Coil	7-12
7.4.5	Command Code : 06, Preset Single Register.....	7-13
7.4.6	Command Code : 15, Force Multiple Coils	7-14
7.4.7	Command Code : 16, Preset Multiple Register	7-15
7.4.8	Command Code : 17, Report Slave ID	7-16
7.5	ELC Device Addresses by Controller	7-17

7 Communications over a network

7.1 Communication Ports

All ELC models have 2 communication ports.

Front port (DIN) is a RS-232 communication port, and is only a slave port making it ideal for HMI or slave to higher end communication networks like DeviceNet, ModbusTCP, etc.

Bottom port is a RS-485 communication port that is master/slave.

Both ports can be programmed for Modbus ASCII or RTU.

Default communication settings for both ports are:

- Modbus ASCII
- 7 data bits
- 1 stop bit
- Even parity
- 9600 baud

7.2 Communication Protocol ASCII mode

Communication Data Frame

9600 (Baud rate) , EVEN (Parity) , 1 (Start bit) , 1 (Stop bit)

STX	Start character ':' (3AH)
ADR 1	Communication address: 8-bit address consists of 2 ASCII codes
ADR 0	
CMD 1	Command code: 8-bit command consists of 2 ASCII codes
CMD 0	
DATA (0)	Contents of data: n×8-bit data consist of 2n ASCII codes. n≤37, maximum of 74 ASCII codes
DATA (1)	
.....	
DATA (n-1)	
LRC CHK 1	LRC check sum: 8-bit check sum consists of 2 ASCII codes
LRC CHK 0	
END 1	End character:
END 0	

7.2.1 ADR (Communication Address)

Valid communication addresses are in the range of 0...31. Communication address equals to 0 means broadcast to all ELCs. ELC will not respond to a broadcast message. ELC will reply to a normal message (non-broadcast) to the master device.

Example, communication to an ELC with address 16 decimal (10hex):

(ADR 1, ADR 0)='1','0'⇒'1'=31H, '0' = 30H

7.2.2 CMD (Command code) and DATA (data characters)

The data character format depends on the command code. For example, reading continuous 8 words from starting device address 0614H of ELC with address 01H.

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	03
Starting Address Hi	06
Starting Address Lo	14
Number of Points Hi	00
Number of Points Lo	08
Error Check (LRC)	DA
END 1	0D
END 0	0A

Number of Points(max)

= 18 (for 16 bit register)

= 9 (for 32 bit register)

Example : Reading Coils T20~T27 from slave device 01

PC→ELC

“: 01 03 06 14 00 08 DA CR LF”

ELC→PC

“: 01 03 10 00 01 00 02 00 03 00 04 00 05 00 06 00 07 00 08 C8 CR LF”

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	03

Field Name	Example (Hex)
Bytes Count	10
Data Hi (T20)	00
Data Lo (T20)	01
Data Hi (T21)	00
Data Lo (T21)	02
Data Hi (T22)	00
Data Lo (T22)	03
Data Hi (T23)	00
Data Lo (T23)	04
Data Hi (T24)	00
Data Lo (T24)	05
Data Hi (T25)	00
Data Lo (T25)	06
Data Hi (T26)	00
Data Lo (T26)	07
Data Hi (T27)	00
Data Lo (T27)	08
Error Check (LRC)	C8
END 1	0D
END 0	0A

7.2.3 LRC CHK (check sum)

LRC (Longitudinal Redundancy Check) is calculated by summing up, module 256, the values of the bytes from ADR1 to last data character then calculating the hexadecimal representation of the 2's-complement negation of the sum.

For example, reading 1 word form address 0401H of the ELC with address 01H

STX	' :
ADR 1	'0'
ADR 0	'1'
CMD 1	'0'
CMD 0	'3'
Starting data address	'0'
	'4'
	'0'
	'1'
Number of data	'0'
	'0'
	'0'
	'1'
LRC CHK 1	'F'
LRC CHK 0	'6'
END 1	CR
END 0	LF

$01H+03H+04H+01H+00+01H = 0AH$, the 2's-complement negation of 0AH is F6H

Exception response:

The ELC is expected to return a normal response after receiving command messages from the master device. The following depicts the conditions that either a no response or an error response is replied to the master device.

The ELC did not receive a valid message due to a communication error; thus the ELC has no response. The master device will eventually process a timeout condition.

The ELC receives a valid message without a communication error, but cannot accommodate it, an exception response will return to the master device. In the exception response, the most significant bit of the original command code is set to 1, and an exception code explaining the condition that caused the exception is returned.

An example of exception response of command code 01H and exception 02H:

Command message:

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Function	01
Starting Address Hi	04
Starting Address Lo	00
Number of Points Hi	00
Number of Points Lo	10
Error Check (LRC)	EA
END 1	0D
END 0	0A

Response message:

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Function	81
Exception Code	02
Error Check (LRC)	7C
END 1	0D
END 0	0A

Exception code:	Meaning:
01	Illegal command code: The command code received in the command message is not available for the ELC.
02	Illegal device address: The device address received in the command message is not available for the ELC.
03	Illegal device value: The device value received in the command message is not available for the ELC.
07	Check Sum Error Check if the check Sum is correct Illegal command messages The command message is too short. Command message length is out of range.

The format of data characters depends on the command. The available command codes are described as followed.

Code	Name	Description
01	Read Coil Status	S, Y, M, T, C
02	Read Input Status	S, X, Y, M, T, C
03	Read Holding Registers	T, C, D
05	Force Single Coil	S, Y, M, T, C
06	Preset Single Register	T, C, D
15	Force Multiple Coils	S, Y, M, T, C
16	Preset Multiple Register	T, C, D
17	Report Slave ID	None

7.3 ELC Device Address

Device	Range	Effective Range		Address
		PB	PC/PA/PH	
S	000~255	000~127	000~1023	0000~00FF
S	246~511			0100~01FF
S	512~767			0200~02FF
S	768~1023			0300~03FF
X	000~377 (Octal)	000~177 (Octal)	000~177 (Octal)	0400~04FF
Y	000~377 (Octal)	000~177 (Octal)	000~177 (Octal)	0500~05FF
T	000~255	000~127	000~255	0600~06FF
M	000~255	0000~1279	0000~4095	0800~08FF
M	256~511			0900~09FF
M	512~767			0A00~0AFF
M	768~1023			0B00~0BFF
M	1024~1279			0C00~0CFF
M	1280~1535			0D00~0DFF
M	1536~1791			B000~B0FF
M	1792~2047			B100~B1FF
M	2048~2303			B200~B2FF
M	2304~2559			B300~B3FF
M	2560~2815			B400~B4FF
M	2816~3071			B500~B5FF
M	3072~3327			B600~B6FF
M	3328~3583			B700~B7FF
M	3584~3839			B800~B8FF
M	3840~4095			B900~B9FF
C	000~255	000~127 235~255	000~255	0E00~0EFF
D	000~255	000~599 1000~1143	0000~4095	1000~10FF
D	256~511			1100~11FF
D	512~767			1200~12FF
D	768~1023			1300~13FF
D	1024~1279			1400~14FF
D	1280~1535	None		1500~15FF
D	1536~1791			1600~16FF
D	1792~2047			1700~17FF
D	2048~2303			1800~18FF
D	2304~2559			1900~19FF
D	2560~2815			1A00~1AFF
D	2816~3071			1B00~1BFF
D	3072~3327			1C00~1CFF
D	3328~3583			1D00~1DFF
D	3584~3839			1E00~1EFF
D	3840~4095			1F00~1FFF
D	4096~4999	4096~4999	9000~9387	

7.4 Command Code

7.4.1 Command Code : 01, Read Coil Status

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	01
Starting Address Hi	06
Starting Address Lo	14
Number of Points Hi	00
Number of Points Lo	25
Error Check (LRC)	BF
END 1	0D
END 0	0A

Number of Points (max) = 255 = 0x00FF

Example : Reading Coils T20~T56 from slave device 01

PC→ELC “:01 01 06 14 00 25 BF CR LF”

ELC→PC “:01 01 05 CD 6B B2 0E 1B D6 CR LF”

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	01
Bytes Count	05
Data (Coils T27...T20)	CD
Data (Coils T35...T38)	6B
Data (Coils T43...T36)	B2
Data (Coils T51...T44)	0E
Data (Coils T56...T52)	1B
Error Check (LRC)	E6
END 1	0D
END 0	0A

7.4.2 Command Code : 02, Read Input Status

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	02
Starting Address Hi	05
Starting Address Lo	14
Number of Points Hi	00
Number of Points Lo	25
Error Check (LRC)	BF
END 1	0D
END 0	0A

Example: Reading Contact Y024~Y070 from slave device 01

PC→ELC “: 01 02 05 14 00 25 BF CR LF”

ELC→PC “: 01 01 05 CD 6B B2 0E 1B E5 CR LF”

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	02
Bytes Count	05
Data (Coils Y033...Y024)	CD
Data (Coils Y043...Y034)	6B
Data (Coils Y053...Y044)	B2
Data (Coils Y063...Y054)	0E
Data (Coils Y070...Y064)	1B
Error Check (LRC)	E5
END 1	0D
END 0	0A

7.4.3 Command Code : 03, Read Holding Register

Holding Register : T, C, D

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	03
Starting Address Hi	06
Starting Address Lo	14
Number of Points Hi	00
Number of Points Lo	08
Error Check (LRC)	DA
END 1	0D
END 0	0A

Number of Points(max)

= 18 (for 16 bit register)

= 9 (for 32 bit register)

Example: Reading Coils T20~T27 from slave device 01

PC→ELC

“: 01 03 06 14 00 08 DA CR LF”

ELC→PC

“:01 03 10 00 01 00 02 00 03 00 04 00 05 00 06 00 07 00 08 B8 CR LF”

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	03
Bytes Count	10
Data Hi (T20)	00
Data Lo (T20)	01
Data Hi (T21)	00
Data Lo (T21)	02
Data Hi (T22)	00
Data Lo (T22)	03
Data Hi (T23)	00
Data Lo (T23)	04
Data Hi (T24)	00

Field Name	Example (Hex)
Data Lo (T24)	05
Data Hi (T25)	00
Data Lo (T25)	06
Data Hi (T26)	00
Data Lo (T26)	07
Data Hi (T27)	00
Data Lo (T27)	08
Error Check (LRC)	C8
END 1	0D
END 0	0A

7.4.4 Command Code : 05, Force Single Coil

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	05
Coil Address Hi	05
Coil Address Lo	00
Force Data Hi	FF
Force Data Lo	00
Error Check (LRC)	F6
END 1	0D
END 0	0A

MMNN = 0xFF00....Coil ON

MMNN = 0x0000....Coil OFF

Example: Forcing Coil Y000 ON

PC→ELC “: 01 05 05 00 FF 00 F6 CR LF”

ELC→PC “: 01 05 05 00 FF 00 F6 CR LF”

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	05
Coil Address Hi	05
Coil Address Lo	00

Field Name	Example (Hex)
Force Data Hi	FF
Force Data Lo	00
Error Check (LRC)	F6
END 1	0D
END 0	0A

7.4.5 Command Code : 06, Preset Single Register

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	06
Register Address Hi	06
Register Address Lo	00
Preset Data Hi	12
Preset Data Lo	34
Error Check (LRC)	AD
END 1	0D
END 0	0A

Example : Setting Register T0 to 12 34

PC→ELC “: 01 06 06 00 12 34 AD CR LF”

ELC→PC “: 01 06 06 00 12 34 AD CR LF”

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	06
Register T0 Address Hi	06
Register T0 Address Lo	00
Preset Data Hi	12
Preset Data Lo	34
Error Check (LRC)	AD
END 1	0D
END 0	0A

Switch (c)

Case 0: T0

Q →: 01 06 06 00 12 34 AD CR LF

Case 1: C0

Q →: 01 06 0E 00 12 34 AF CR LF

Case 2: C232

Q →: 01 06 0E E8 12 34 56 78 EF CR LF

Case 3: D10

Q →: 01 06 10 0A 12 34 99 CR LF

Case 4: D1000

Q →: 01 06 13 E8 12 34 BA CR LF

7.4.6 Command Code : 15, Force Multiple Coils

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	0F
Coil Address Hi	05
Coil Address Lo	00
Quantity of Coils Hi	00
Quantity of Coils Lo	0A
Byte Count	02
Force Data Hi	CD
Force Data Lo	01
Error Check (LRC)	11
END 1	0D
END 0	0A

Quantity of Coils (max) = 255

Example: Setting Coil Y007...Y000 = 1100 1101, Y011...Y010 = 01.

PC→ELC “: 01 0F 05 00 00 0A 02 CD 01 11 CR LF”

ELC→PC “: 01 0F 05 00 00 0A E1 CR LF”

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	0F
Register T0 Address Hi	05
Register T0 Address Lo	00
Preset Data Hi	00

Field Name	Example (Hex)
Preset Data Lo	0A
Error Check (LRC)	E1
END 1	0D
END 0	0A

7.4.7 Command Code : 16, Preset Multiple Register

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	10
Starting Address Hi	06
Starting Address Lo	00
Number of Register Hi	00
Number of Register Lo	02
Byte Count	04
Data Hi	00
Data Lo	0A
Data Hi	01
Data Lo	02
Error Check (LRC)	C6
END 1	0D
END 0	0A

Number of Register(max)

= 16 (for 16 bit register)

= 8 (for 8 bit register)

Example: Setting Register T0 to 00 0A, T1 to 01 02.

PC→ELC “: 01 10 06 00 00 02 04 00 0A 01 02 C6 CR LF”

ELC→PC “: 01 10 06 00 00 02 E7 CR LF”

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	10
Starting Address Hi	06
Starting Address Lo	00

Field Name	Example (Hex)
Number of Registers Hi	00
Number of Registers Lo	02
Error Check (LRC)	E7
END 1	0D
END 0	0A

7.4.8 Command Code : 17, Report Slave ID

Returns a description of controller present at the slave address, the current status of the slave Run indicator, and other information specific to the slave device.

Command message:

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	11
Error Check (LRC)	EE
END 1	0D
END 0	0A

Response message:

Field Name	Example (Hex)
Heading	3A
Slave Address	01
Command code	11
Byte Count	04
Slave ID	01
Run Indicator Status 00 = OFF FF = ON	FF
Data 0 (D1001 HI)	40
Data 1 (D1001 LOW)	10
Error Check (LRC)	9A
END 1	0D
END 0	0A

7.5 ELC Device Addresses by Controller

Device	Range		Type	Address (HEX)	Modbus address	ELC Range	
						PB	PC/PA/PH
S	000~255		bit	0000~00FF	000001~000256	0~127	0~1023
S	246~511		bit	0100~01FF	000257~000512		
S	512~767		bit	0200~02FF	000513~000768		
S	768~1023		bit	0300~03FF	000769~001024		
X	000~377 (Octal)		bit	0400~04FF	101025~101280	0~177	0~177
Y	000~377 (Octal)		bit	0500~05FF	001281~001536		
T	000~255		bit	0600~06FF	001537~001792	0~127	0~255
	000~255		word	0600~06FF	401537~401792	0~127	0~255
M	0000~1535		bit	0800~0DFF	002049~003584	0~1279	0~4095
M	1536~4095		bit	B000~B9FF	045057~047616		
C	000~199	16-bit	Bit	0E00~0EC7	003585~003784	0~127	0~199
			Word	0E00~0EC7	403585~403784	0~127	0~199
	200~255	32-bit	Bit	0EC8~0EFF	003785~003840	232~255	200~255
			Word	0700~076F	401793~401903*	232~255	200~255
D	0000~4095		Word	1000~1FFF	404097~408192	0~1311	0~4999
D	4096~4999		Word	9000~9387	436865~437768		

* Odd address valid

MEMO

7

Sequential Function Charts

This chapter contains information in programming in SFC mode.

This Chapter Contains

8.1	Step Ladder Command [STL], [RET]	8-2
8.2	Sequential Function Chart (SFC)	8-3
8.3	Step Ladder Command Explanation.....	8-5
8.4	Reminder of Design on the Step Ladder Program	8-11
8.5	Categories of Procedures	8-13
8.6	IST Command	8-24

8 Sequential Function Charts

8.1 Step Ladder Command [STL], [RET]

Mnemonic	Operands	Function	Program steps	Controllers			
				PB	PC	PA	PH
STL	S0~S1023	Step Transition Ladder Start Command	1				

Explanation:

The step ladder command, STL Sn, has constituted the stepping point, and when the STL command showed up in the program, it implies that the program is now at the step ladder diagram condition that is controlled by the step procedure. The step ladder command RET represents the end of the step ladder diagram (from S0~S9) that is to return to the BUS command. The SFC diagram is represented through the step ladder diagram composed of STL/RET. The number of step point S can't be repeated.

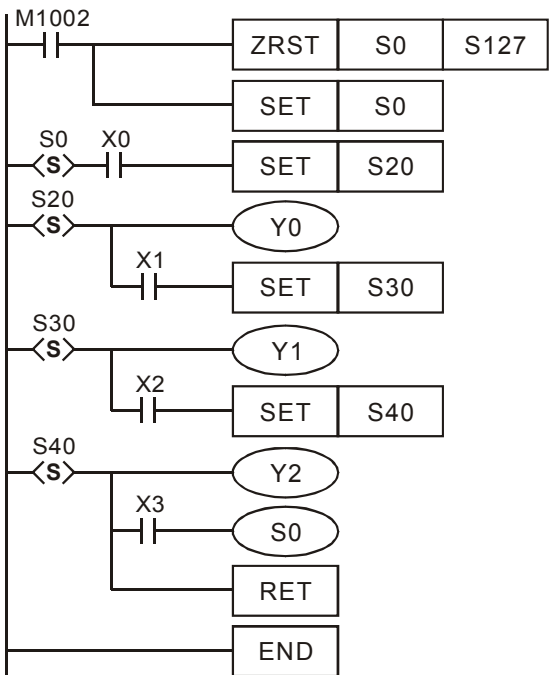
Mnemonic	Operands	Function	Program steps	Controllers			
				PB	PC	PA	PH
RET	None	Step Transition Ladder Return Command	1				

Explanation:

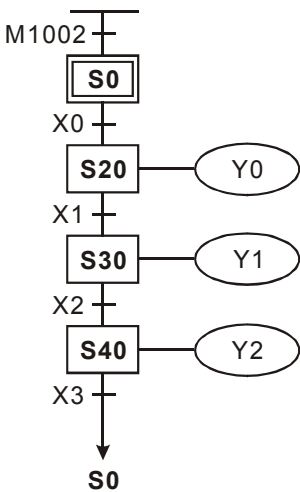
At the end of the step procedure, be sure to write in the RET command; the RET command indicates the end of the step procedure. Maximum is 10 step procedures (S0~S9) for a ELC program and it should have RET command at the end of each step procedure.

Program Example:

Step ladder diagram:



SFC:









8.2 Sequential Function Chart (SFC)

In automatic control field, it often needs to cooperate with electric control and mechanical control to reach the goal. SFC can be divided into several serial STEP (i.e. several phases). Each STEP should finish its actions. It usually has transition to transfer from each step to the next step. That is the design concept of Sequential Function Chart to have transition to end the action of the previous step and start the action of the next step (the previous step will be clear at this time).

Features:

<ol style="list-style-type: none"> 1. You don't need to do SFC design for constant state step. ELC will execute the action of interlock and double output between each state. It is only needed do simple SFC design for each state and make machine works. 2. The action is easy to understand and easy to adjust initial ELC start-up, detect and maintain. 3. SFC edition theory is made by IEC1131-3. It is figure edition mode and the structure looks like flow chart. Each ELC internal step relay S is used to be step point and also equal to each step of flow chart. After finishing present step, it will transfer to the next step, i.e. next step point S, by setting condition. By repeating this way, it can reach the result that user needs. 4. Explanation of right side SFC figure: each step has its own transition condition to move from one step to the next step. In this figure, primary step point S0 will move to step point S21 once this transition condition X0 is established, and S21 can move to S22 or S24 by transition condition X1 or X2 and S25 will move to S0 to finish a whole procedure once transition condition X6 is established. By this way, it can circulate control with repeat again and again. 	<p>SFC:</p> <pre> graph TD S0[S0] -- X0 --> S21[S21] S21 -- X1 --> S22[S22] S21 -- X2 --> S24[S24] S22 -.- X3 -.-> S24 S24 -- X4 --> S25[S25] S25 -- X5 --> S0 S0 -- X6 --> S21 </pre>
	<p>It is used for ladder step mode. This figure means internal edition program is a general step ladder diagram not step ladder program.</p>
	<p>It is for primary step point. This double-frame is used for SFC primary step point and the usage devices are S0~S9.</p>
	<p>It is used for general step point and the usage devices are S10~S1023.</p>

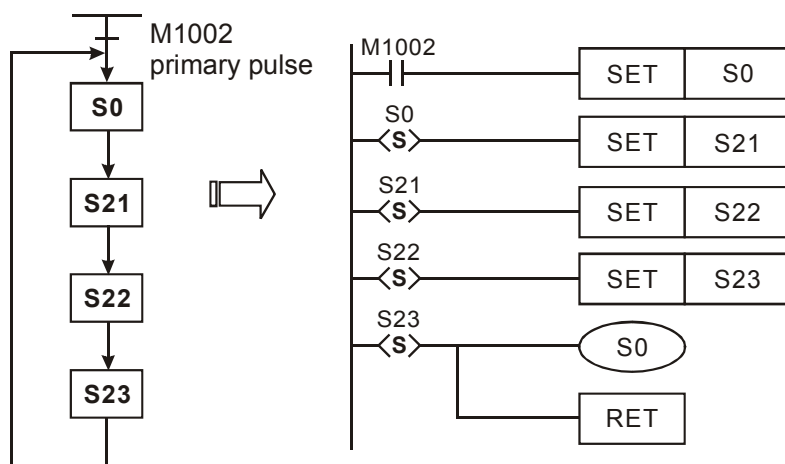
	It is JUMP step point that used to move from step point to another which is not next to it. (it can be used to disconnected jump up or jump down in the same program procedure, return to primary step point or jump between different program procedure.
	It is the transition condition of step point that used to move between each step point.
	It is alternative divergence that used for a step point to move to different corresponding step point by different transition condition.
	It is alternative convergence that used for two step points and above to move to the same step point according to transition condition.
	It is simultaneous divergence that used for a step point move to two step points and above by the same transition condition.
	It is simultaneous convergence that used for two step points and above to move to the same step point with the same transition condition when the condition is established at the same time.

8.3 Step Ladder Command Explanation

STL command: this command is used in the syntax design for the Sequential Function Chart (SFC). This command helps the program designer to have clearer ideas on the program procedure, and thus the procedure will be more readable. As shown in the following diagrams, we could switch our procedure diagram from the left diagram to the right ELC structure diagram.

At the end of the step procedure, be sure to write in the RET command; the RET command indicates the end of the step procedure. Several step procedures could be written in to the same program, just make sure to write in the RET command at the end of the step procedure. There is no limitation to the usage of the RET command, and this command should match up with the usage of the step points (S0~S9).

If the RET command is not written in at the end of the step procedure, this error will be detected by the editing device.

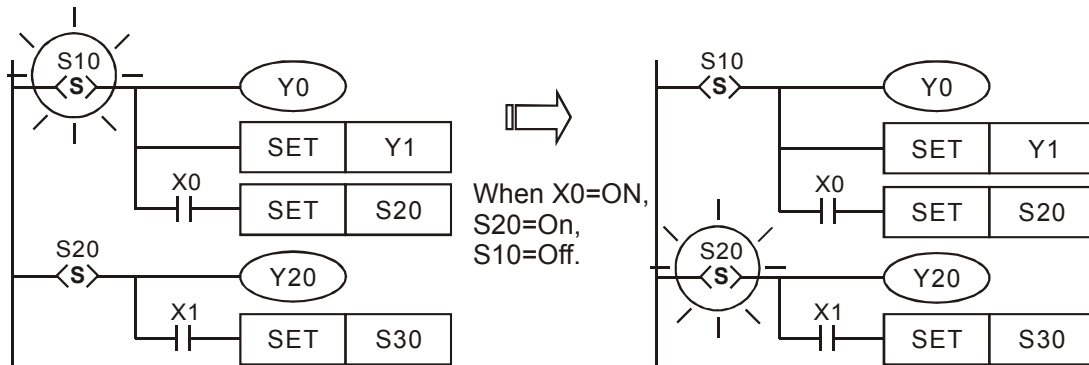


Step Ladder Action:

The step ladder is made up of numerous step points; each step point represents one control procedure action, and each step point needs to execute following three missions:

1. Drive output coil
2. Specific transition condition
3. Designate what step point is to be appointed to take over the control power of the present step point

Example:



Explanation:

When S10=ON, Y0 and Y1 are ON. When X0=ON, S20=ON and Y20 is ON, too.

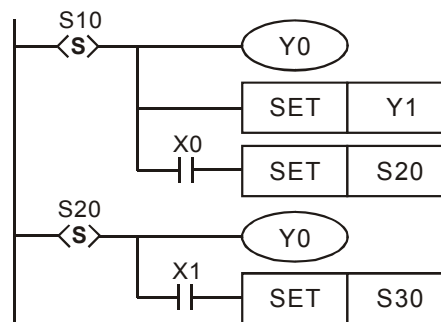
And when S10 is OFF, Y0 will be OFF, but Y1 is ON. (Since Y1 uses the SET command, it will keep in ON status)

Step ladder timing:

When state contact Sn is ON, circuit will be activated and circuit won't be activated when state contact Sn is OFF. (Above action will be executed after delaying a scan time)

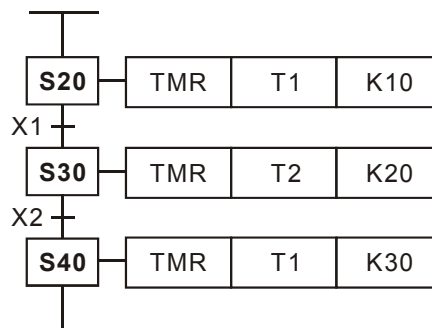
The repeated usage of the output coil:

1. Output coils of the same number could be used in different step points.
2. Such as right diagram, there is the same output device Y0 in the different state. No matter S10 or S20 is ON, Y0 will be ON.
3. Y0 will be close during the transition from S10 to S20 and output Y0 after S20 is ON. Thus in this case, Y0 will be ON no matter S10 or S20 is ON.
4. For general ladder diagrams, repeated usages of the output coils should be avoided. Output coil number used in step point should be avoided to use after returning to general ladder diagram.



Repeated usage of the timer:

Same as general output points, the timer could be used repeatedly for different step points. (this is one feature of step ladder diagram, but for general ladder diagrams, repeated usages of the output coils should be avoided. Output coil number used in step point should be avoided to use after returning to general ladder diagram.)



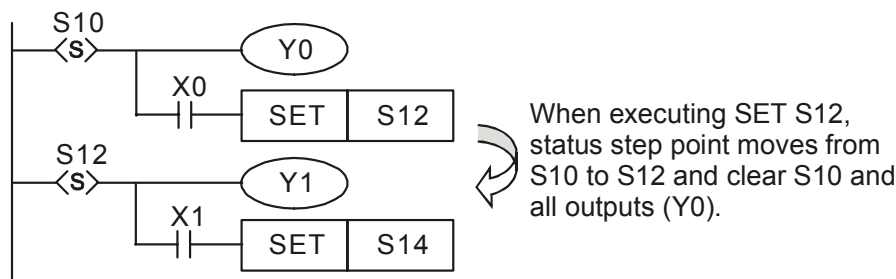
Note: as right diagram, PB/PC/PA/PH series timer only used repeatedly in disconnected step point.

Transfer of the step point:

SET Sn and OUT Sn commands are both used to start (or to transfer to) another step point, and the occasions to use these commands could be different: when the controlling power is transferred to another step point, the status of the original step point S and the action of the output point would all be erased. Due to that numerous step control procedures could exist at the same program simultaneously (take S0~S9 as the starting and ending points to lead the step ladder diagram), the transfer of steps could thus be on the same step procedure or could be transferred to different step procedures. And thus, the transfer commands, SET Sn and OUT Sn, of the step point might vary somewhat in usage; please refer to the following explanations:

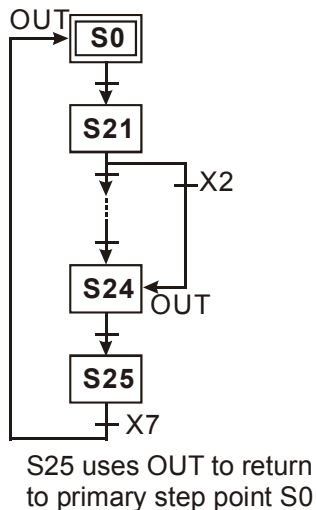
SET Sn

Within the same procedure, it is used to drive up the next status step point, and after the status is transferred, outputs of previous action status points will be clear.

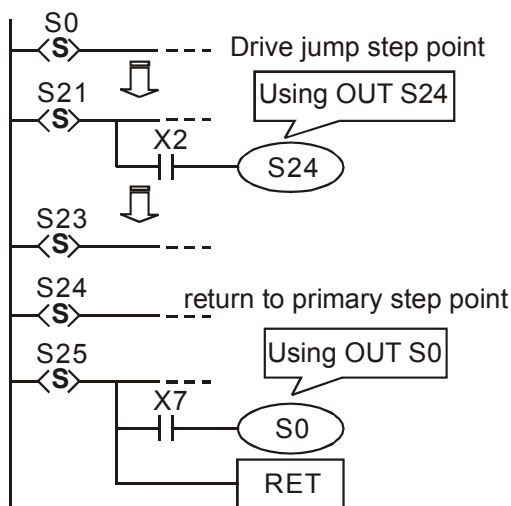
**OUT Sn**

Within the same procedure, transfer of the simultaneous convergence point and different procedures are used to drive up separate step points, and after the status is transferred, outputs of previous action status points will be clear.

SFC diagram:

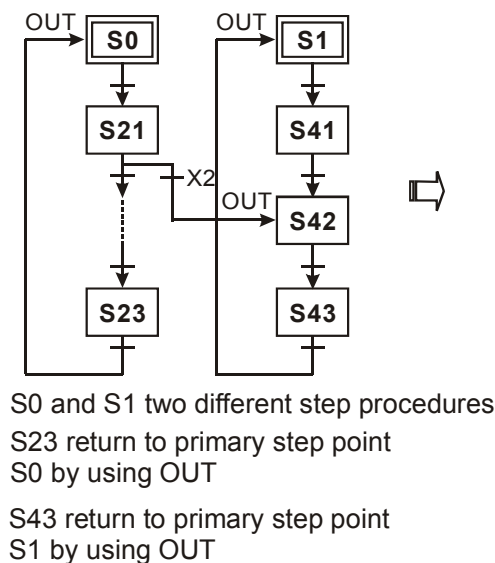


Step Ladder Diagram:

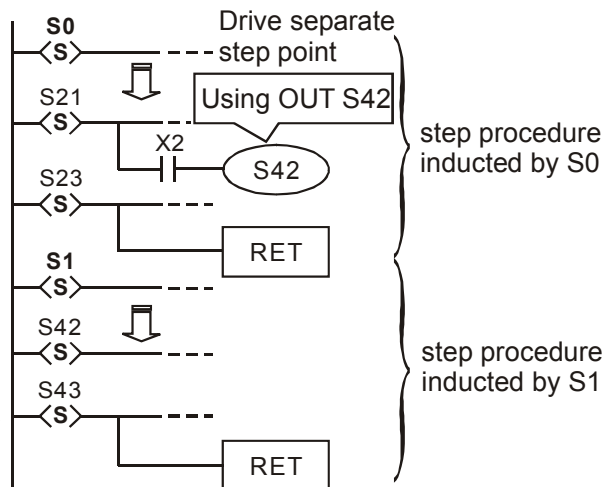


1. Within the same procedure, it is used to return to primary step point.
2. Within the same procedure, it is used for the step points to jump up or down between disconnected step point.
3. At different procedures, it is used to drive up separate step points.

SFC figure:



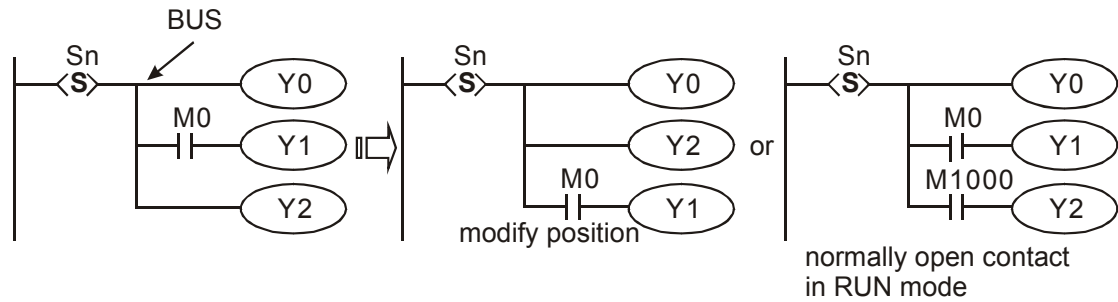
Step Ladder diagram:



Notice of Driving Output Points:

As in the following left diagram, after the LD or LDI command is written in the second line of BUS beyond the step point, output coil can't be connected from BUS directly. There will be error when

compiling. It is needed to modify to following middle and left diagram to correct diagram.



Usage restrictions for partial commands:

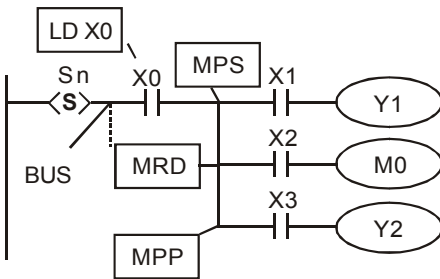
Program of every step point is identical to the general ladder diagram, and every kind of series and parallel connection circuits or application commands could all be utilized, however, part of the commands are under certain restrictions, please refer to the following descriptions:

Basic commands that are to be used within the step point

Basic command		LD/LDI/LDP/LDF AND/ANI/ANDP/ANDF OR/ORI/ORP/ORF INV/OUT/SET/RST	ANB/ORB MPS/MRD/MPP	MC/MCR
Step point				
Primary step point/ General step point		Yes	Yes	No
Diverging step point/ Converging step point	General output	Yes	Yes	No
	Step point transfer	Yes	Yes	No

1. MC/MCR commands are not to be used within the step point.
2. The STL command could not be used in general sub-programs and the interruption service sub-program.
3. Use of the CJ command is not prohibited within the STL command, however, it will complicate the action and should thus be avoided.
4. MPS/MRD/MPP command position:

Ladder Diagram:



Command code:

STL	Sn
LD	X0
MPS	
AND	X1
OUT	Y1
MRD	
AND	X2
OUT	M0
MPP	
AND	X3
OUT	Y2

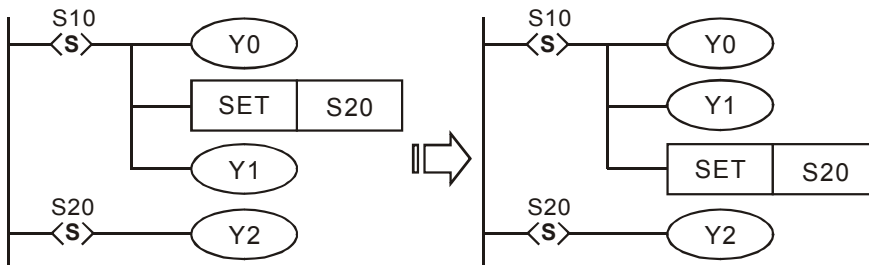
Explanation:

The BUS of step point can't use commands MPS / MRD / MPP directly. It needs to use command LD or LDI before using commands MPS / MRD / MPP.

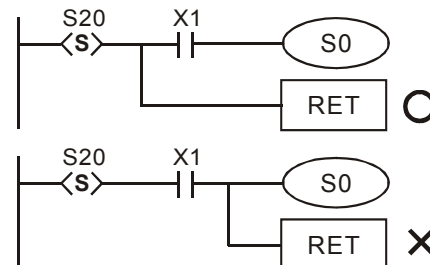
Other Notice:

For general, commands (SET S□ or OUT S□) that used to transfer to next state are better to use after finishing all relative outputs and actions.

In the following figure, they are the same after executing by ELC. If there are many conditions or actions in S10, it is recommended to execute SET S20 after modifying from left figure to right figure and finishing all relative outputs and actions. In this way, the procedure is clear and easy to maintain.



It is needed to add RET command after finishing step ladder program and RET command is also needed to add after STL as shown in right figure.



8.4 Reminder of Design on the Step Ladder Program

1. The step point up front in SFC is called the primary step point, S0~S9. Utilize the primary step point to be the start of the procedure, and use the RET command as the end to construct a complete procedure.
2. If the STL command is not in use, S could be served as general auxiliary relay.
3. The number for the step point, S, could not be used repeatedly.
4. Categories of procedures:

Single procedure: there is only a procedure in a program (the alternative diverge and converge, the simultaneous diverge and converge aren't included)

Complicated single procedure: there is only a procedure in a program and it includes alternative diverge, alternative converge procedures, Simultaneous diverge and simultaneous converge procedures.

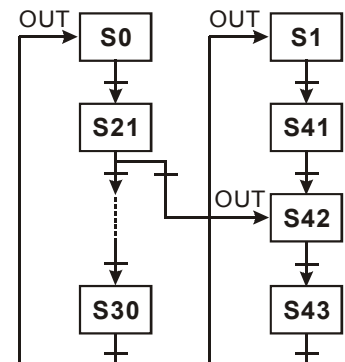
Combination procedure: there are numerous single procedures in a program and maximum is 10 (S0~S9) procedures.

5. Procedure separation: it is allowed to write in numerous procedures within one step ladder diagram

There are two single procedures S0 and S1 at the right diagram; procedure of the program is to write in S0 ~S30 first, and then S1~S43.

Either one step point on the procedure could jump to any one specified step point on other procedures.

Once the condition below S21 at the right diagram is held, it could jump to the specified S42 step point on the S1 procedure; this motion is called the separate step point.



6. Restrictions on the diverging procedure: (See following examples)
 - a) Up to 8 diverging step points could be used within a diverging procedure.
 - b) Up to 16 loops could be used in the combination of plural diverging or simultaneous converging procedures.
 - c) Either one step point on the procedure could jump to any one specified step point on other procedures.
7. Reset of the step point and the output prohibition:
 - a) Use the ZRST command to Reset a section of step points to be OFF.
 - b) Use the output Y prohibition of ELC (M1034=ON).
8. Retaining step point:

When ELC encountered power failure, the retaining step point will memorize the ON/OFF status, and go on with the execution before the power failure after the power is turned back on.

9. Special auxiliary relay and special register: refer to IST command for detail.

Device	Description
M1040	Step transition inhibits. When M1040 is ON, all movement of step point are inhibited.
M1041	Step transition start. Flag for IST command.
M1042	Start pulse. Flag for IST command.
M1043	Origin reset completed. Flag for IST command.
M1044	Origin condition. Flag for IST command.
M1045	All outputs clear inhibit. Flag for IST command.
M1046	STL state setting. Once there is a step point ON, M1046 is ON.
M1047	STL monitor enable
D1040	ON state number 1 of step point S
D1041	ON state number 2 of step point S
D1042	ON state number 3 of step point S
D1043	ON state number 4 of step point S
D1044	ON state number 5 of step point S
D1045	ON state number 6 of step point S
D1046	ON state number 7 of step point S
D1047	ON state number 8 of step point S

8.5 Categories of Procedures

Single procedure:

The basic step action is single procedure control action.

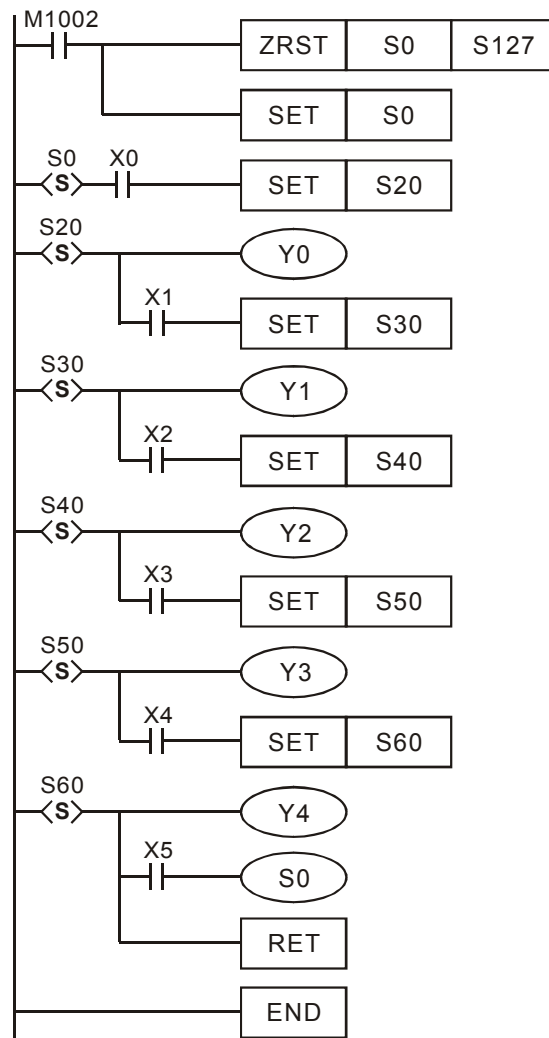
The first step point of step ladder diagram is called primary step point and the number is S0~S9. Those step points after primary step point are called general step point and the number are S10~S1023.

S10~S19 will be used as origin reset step points once use command IST.

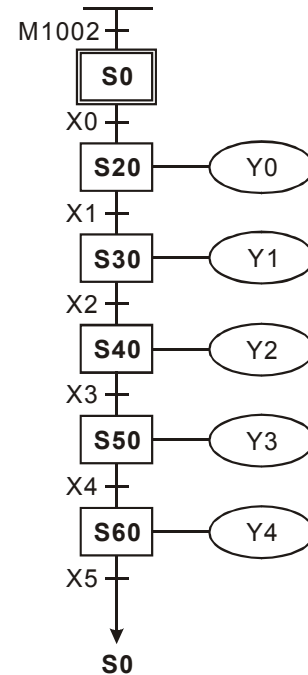
1. Single Procedure without Divergence and Convergence

After finishing a procedure, transferring control power of step point to primary step point.

Step Ladder Diagram

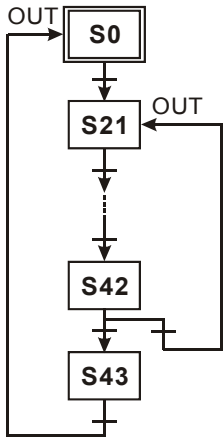


SFC diagram

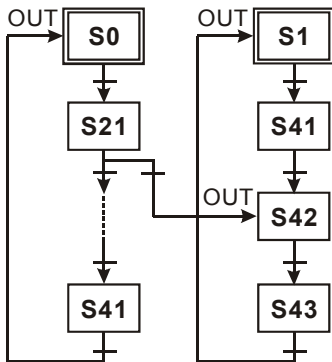


2. JUMP Procedure

- a) Transfer control power of step point to upper certain step point.

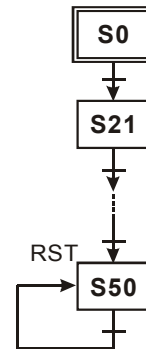


- b) Transfer control power of step point to step point of other procedure.



3. Reset Procedure

At the right diagram, S50 will Reset itself and end the procedure when condition is held.



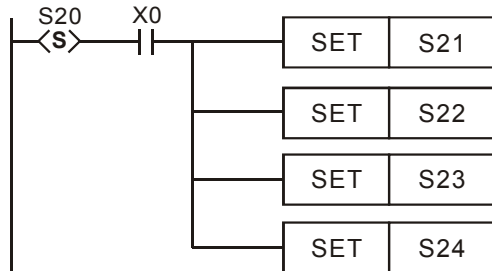
Complicated single procedure:

It includes alternative diverge, alternative converge procedures, Simultaneous diverge and simultaneous converge procedures.

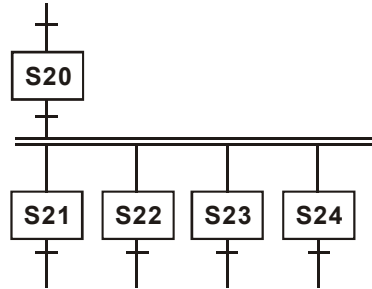
1. Structure of simultaneous divergence

The situation that transfers to many states when present condition is held is called structure of simultaneous divergence as shown in following. When X0=ON, S20 will transfer to S21, S22, S23 and S24 at the same time.

Ladder diagram of simultaneous divergence:



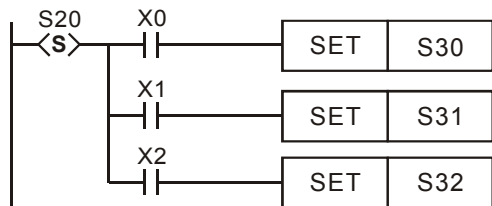
SFC diagram of simultaneous divergence:



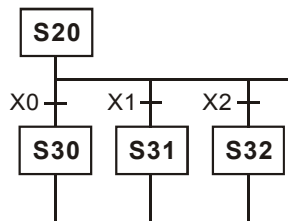
2. Structure of the alternative divergence

The situation that transfers to individual state when individual condition of present state is held is called structure of alternative divergence as shown in following. S20 will transfer to S30 when X0=ON, S20 will transfer to S31 when X1=ON and S20 will transfer to S32 when X2=ON.

Ladder diagram of alternative divergence:



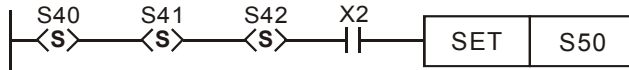
SFC diagram of alternative divergence:



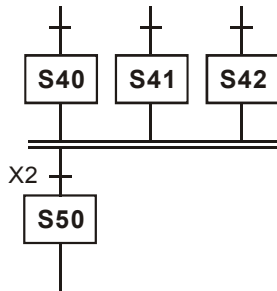
3. Structure of the simultaneous convergence

The situation that transfers to next state when continuous states are held at the same time is called simultaneous convergence.

Ladder diagram of simultaneous convergence:



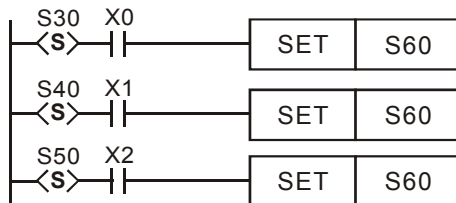
SFC diagram of simultaneous convergence:



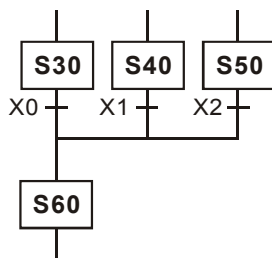
4. Structure of the alternative convergence

The following ladder diagram is alternative convergence. That means it will transfer to S60 once one of S30, S40 and S50 is held.

Ladder diagram of alternative convergence:

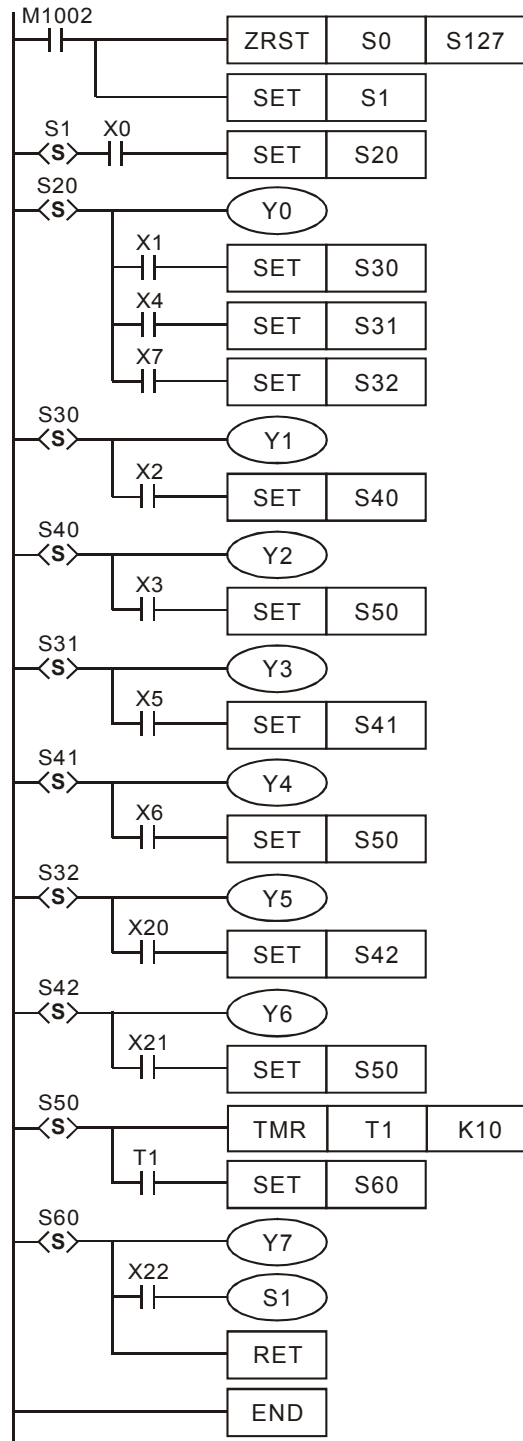


SFC diagram of alternative convergence:

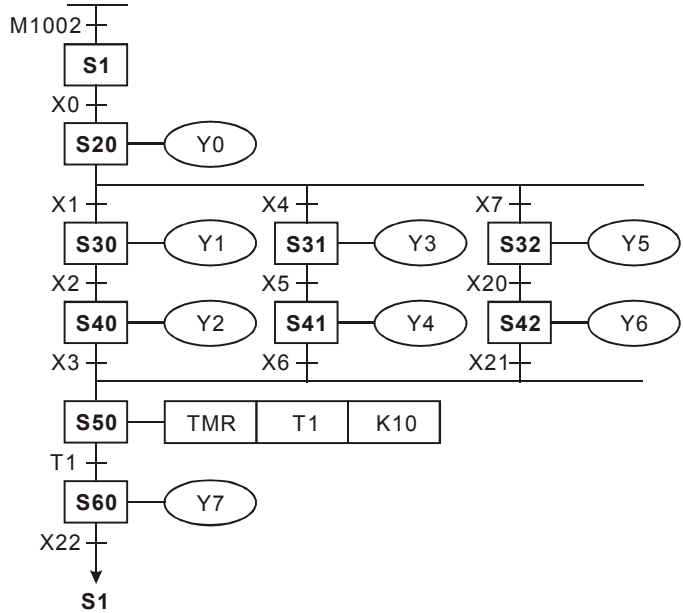


Example of the alternative divergence and alternative convergence procedures

Step Ladder Diagram:

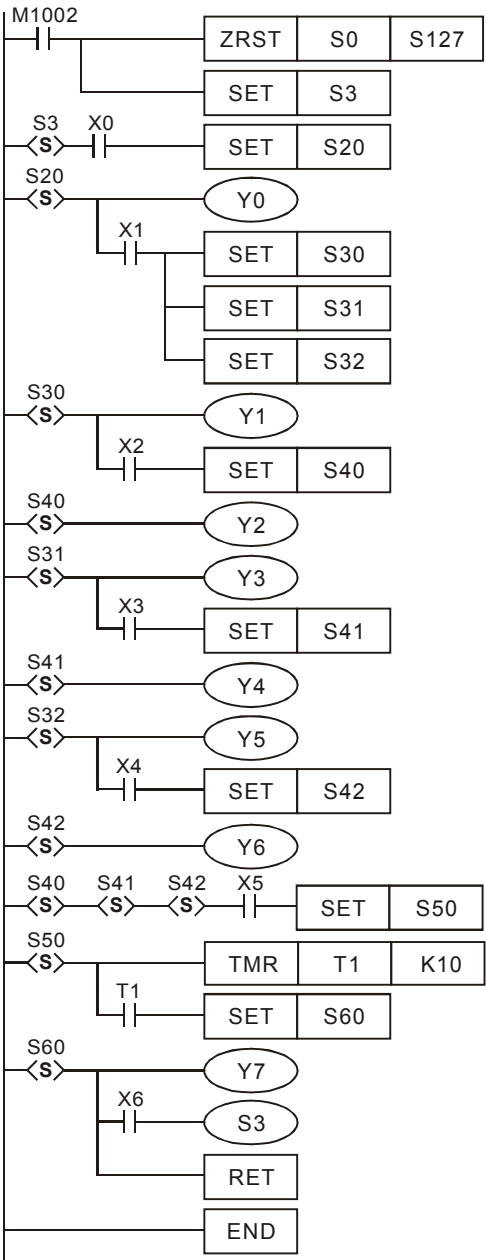


SFC Diagram:

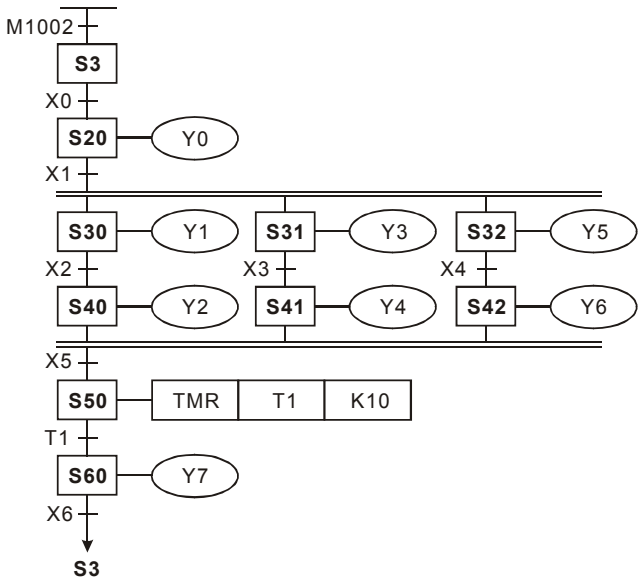


Example of the simultaneous divergence and simultaneous convergence procedures

Step Ladder Diagram:

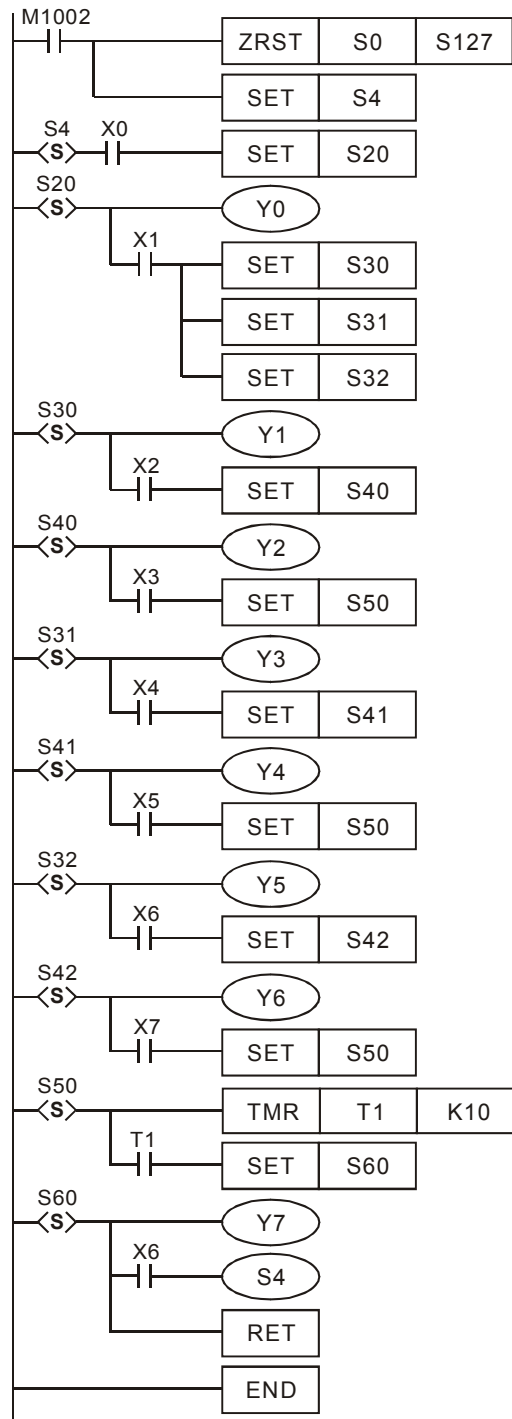


SFC Diagram:

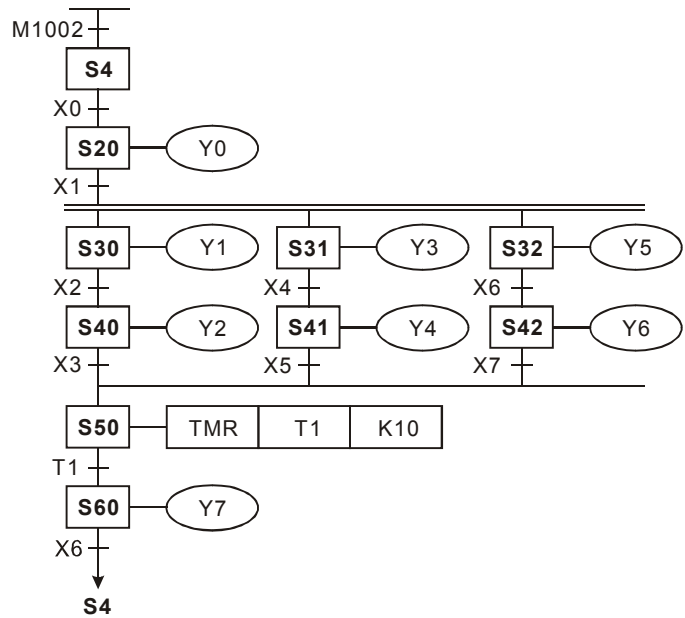


Example of the simultaneous divergence and alternative convergence procedures

Step Ladder Diagram:

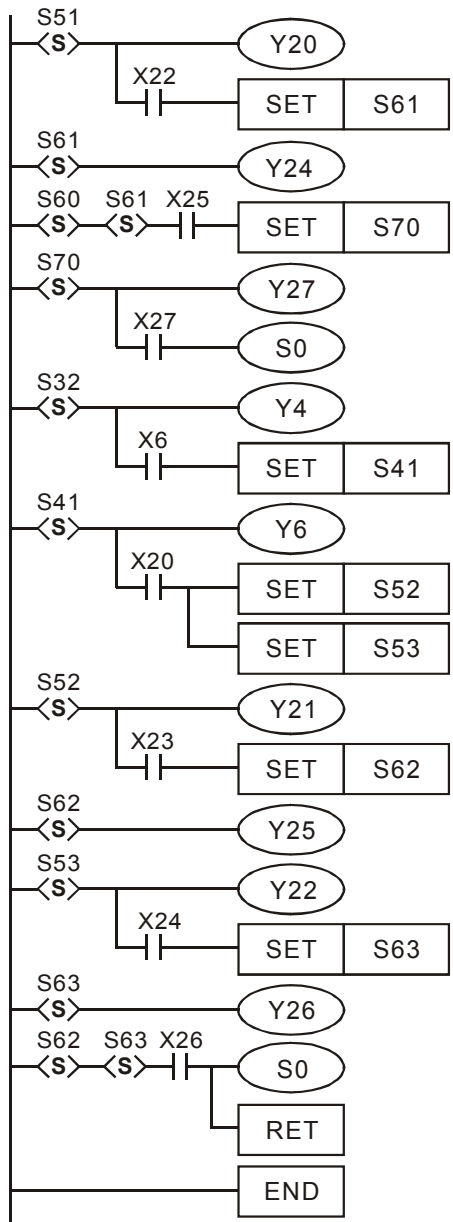
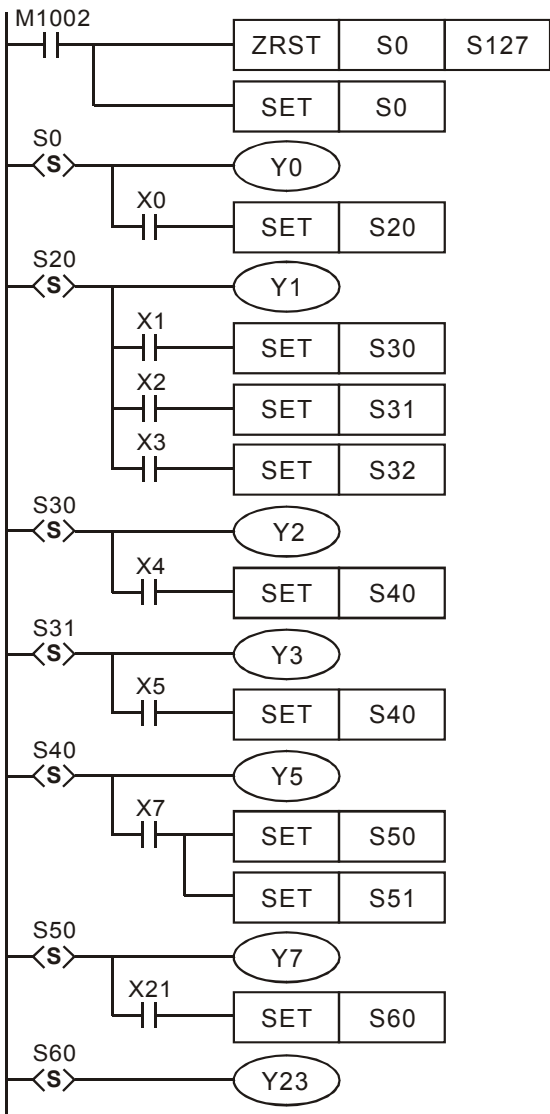


SFC Diagram:

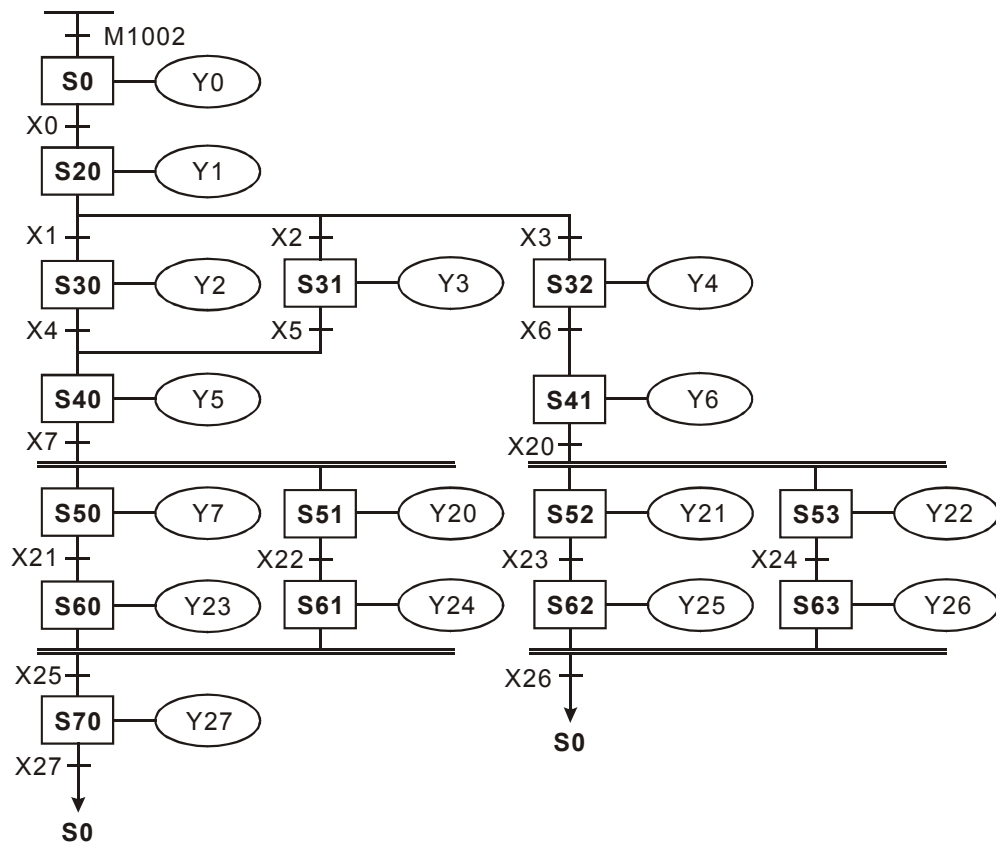


Combination example 1: (includes the alternative divergence and convergence, the simultaneous divergence and convergence)

Step Ladder Diagram:

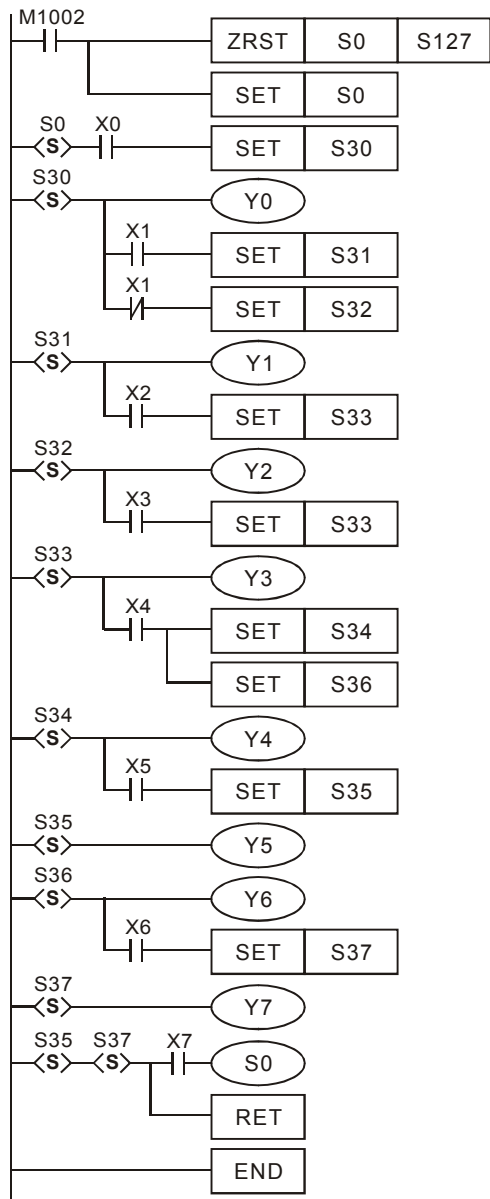


SFC Diagram:

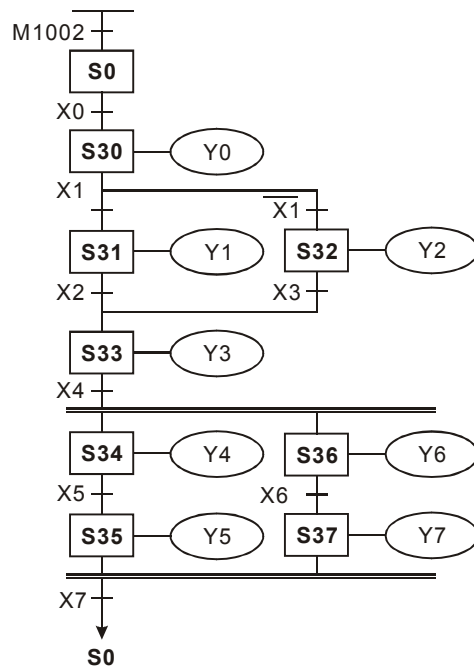


Combination example 2: (includes the alternative divergence and convergence, the simultaneous divergence and convergence)

Step Ladder Diagram:

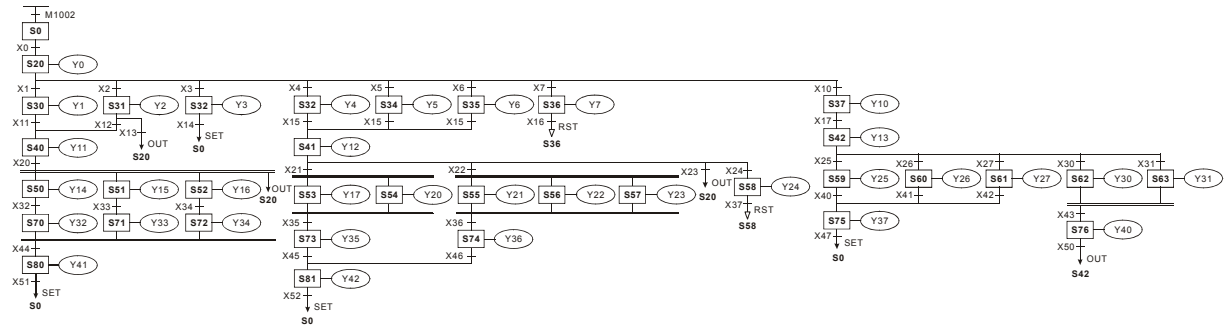


SFC Diagram:



Restrictions on the divergence procedure:

1. Up to 8 divergence step points could be used in a divergence procedure. In following diagram, maximum divergence step points after step point S20 are 8 (S30 - S37).
2. Up to 16 loops could be used in the combination of plural divergence or simultaneous convergence procedures. In following diagram, 4 step points after step point S40, 7 step points after step point S41 and 5 step points after step point S42. In this procedure, maximum is 16 loops.
3. Either one step point on the procedure could jump to any one specified step point on other procedures.

SFC Diagram:

8.6 IST Command

API	Mnemonic			Operands			Function			Controllers			
60		IST		S	D₁	D₂	Manual/Auto Control			PB	PC	PA	PH

OP	Type	Bit Devices				Word devices										Program Steps			
		X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	IST: 7 steps		
S		*	*	*															
D ₁					*														
D ₂					*														

PULSE				16-bit				32-bit			
PB	PC	PA	PH	PB	PC	PA	PH	PB	PC	PA	PH

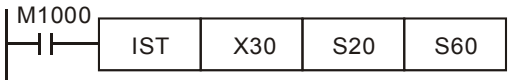
Operands:

S: The starting input number. **D₁:** The smallest number for the designated-status step point in auto mode. **D₂:** The greatest number for the designated-status step point in auto mode.

Explanations:

The IST is a convenient instruction made specifically for the initial state of the step function control procedure.

Program Example 1



- | | |
|--|-------------------------------------|
| S= X30: Individual operation (Manual operation) | X34: Continuous operation |
| X31: Zero point return | X35: Zero point return start switch |
| X32: Step operation | X36: Start switch |
| X33: One cycle operation | X37: Stop switch |

- When the IST instruction is executed, the following special auxiliary relay will switch automatically.

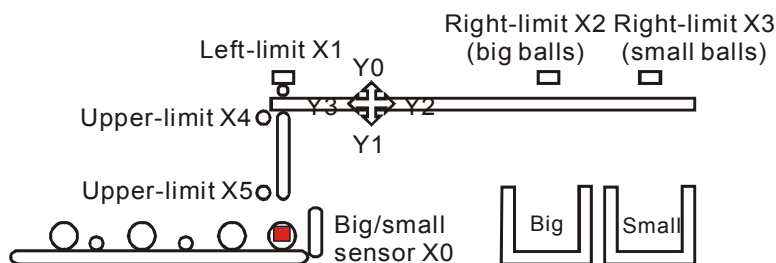
M1040: Movement inhibited	S0: Manual operation/initial state step point
M1041: Movement start	S1: Zero point return/initial state step point
M1042: Status pulse	S2: Auto operation/initial state step point
M1047: STL monitor enable	
- When IST instruction is used, S10~S19 are for zero point return operation and the step point of this state can't be used as a general step point. However, when using S0~S9 step points, S0 initiates "manual operation", S1 initiates "zero point return operation" and S2 initiates "auto operation". Thus, there should be three circuits of these three initial state step points written first in the program.
- When switching to S1 (zero point return mode), zero point return won't have any actions once any of S10~S19 = ON.

4. When switching to S2 (auto operation mode), auto operation won't have any actions once when **S** is between **D1** to **D2** = ON or if M1043=ON

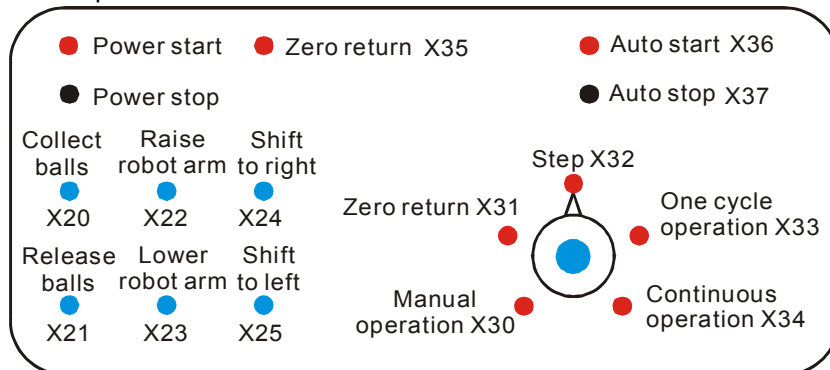
Program Example 2

The Robot arm control (use IST instruction):

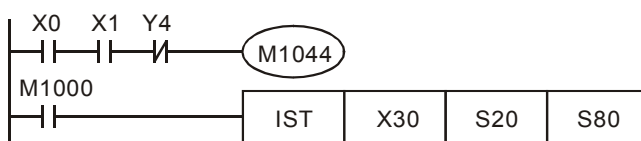
1. Motion request: In the example, two kinds of balls (big and small) are separated and moved to different boxes.
2. Motion of the Robot arm: lower robot arm, collect balls, raise robot arm, shift to right, lower robot arm, release balls, raise robot arm, shift to left to finish motion in order.
3. I/O Device:



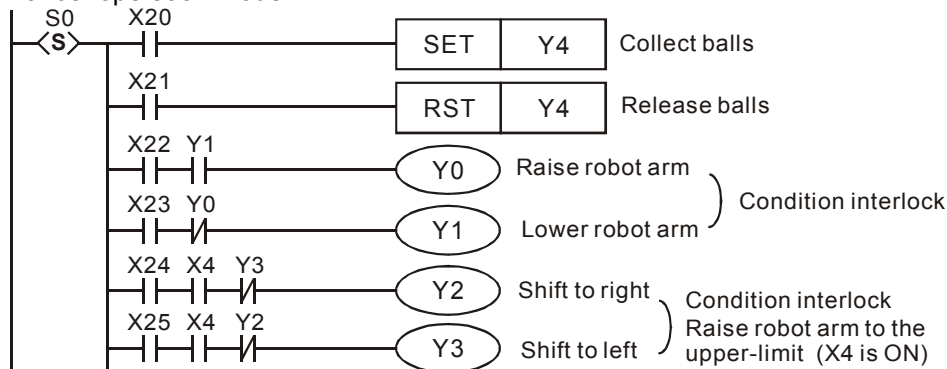
4. Control panel



- a) Big/small sensor X0.
 - b) The left-limit of the robot arm X1, the right-limit X2 (big balls), the right-limit X3 (small balls), the upper-limit X4, and the lower-limit X5.
 - c) Raise robot arm Y0, lower robot arm Y1, shift to right Y2, shift to left Y3, and collect balls Y4.
5. START circuit:

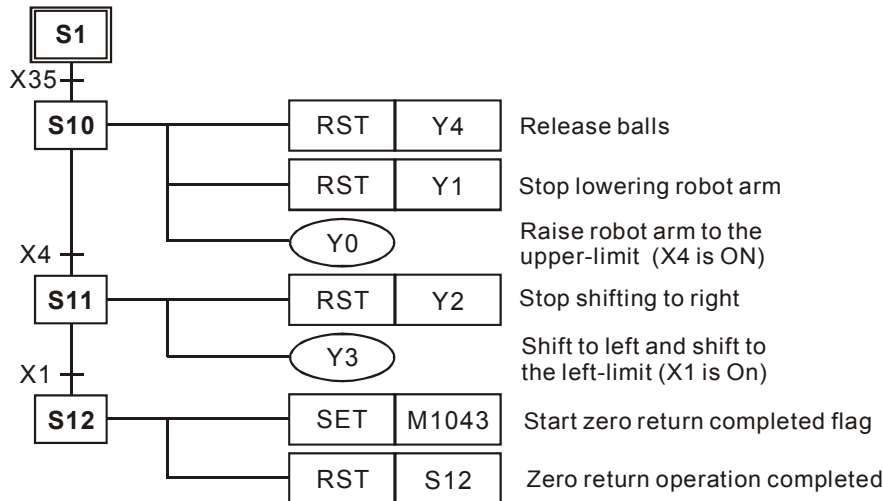


6. Manual operation mode:

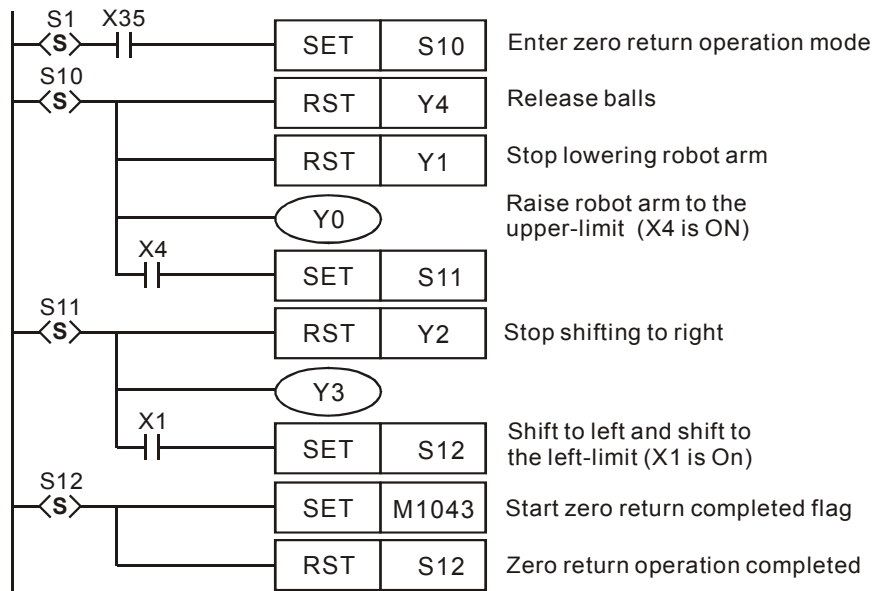


7. Zero point return mode:

a) SFC Diagram:

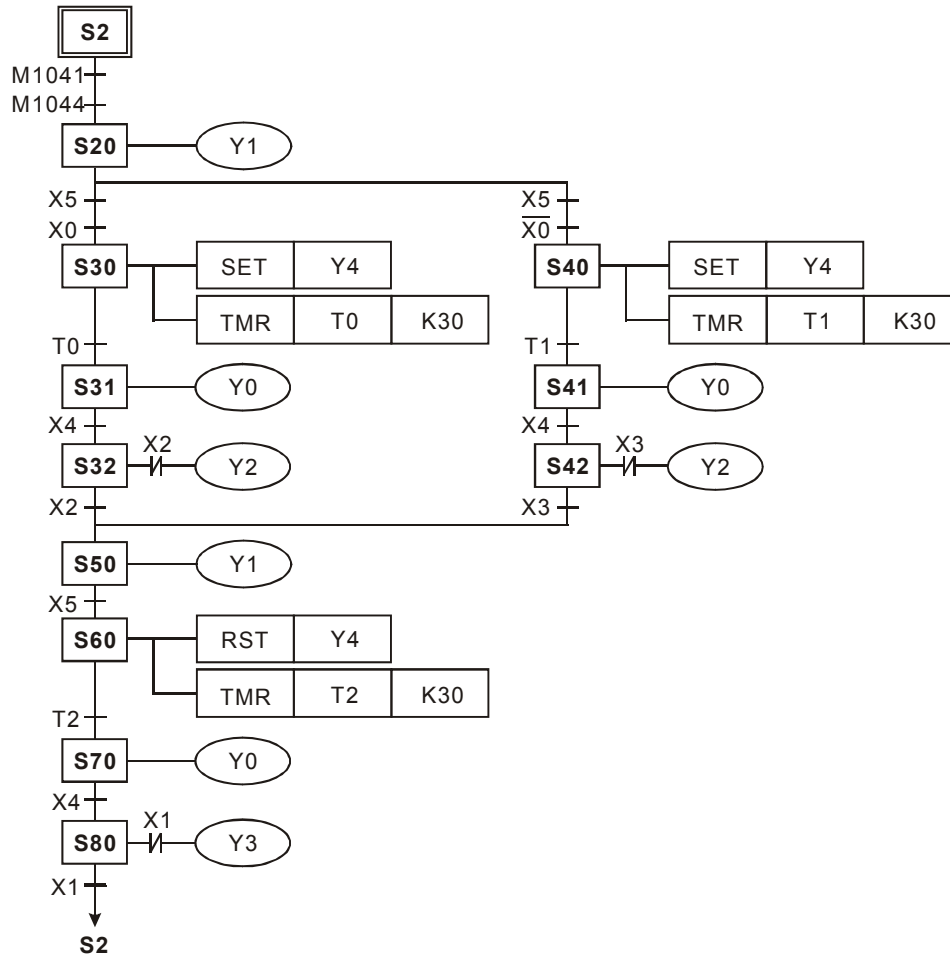


b) Step Ladder Diagram:

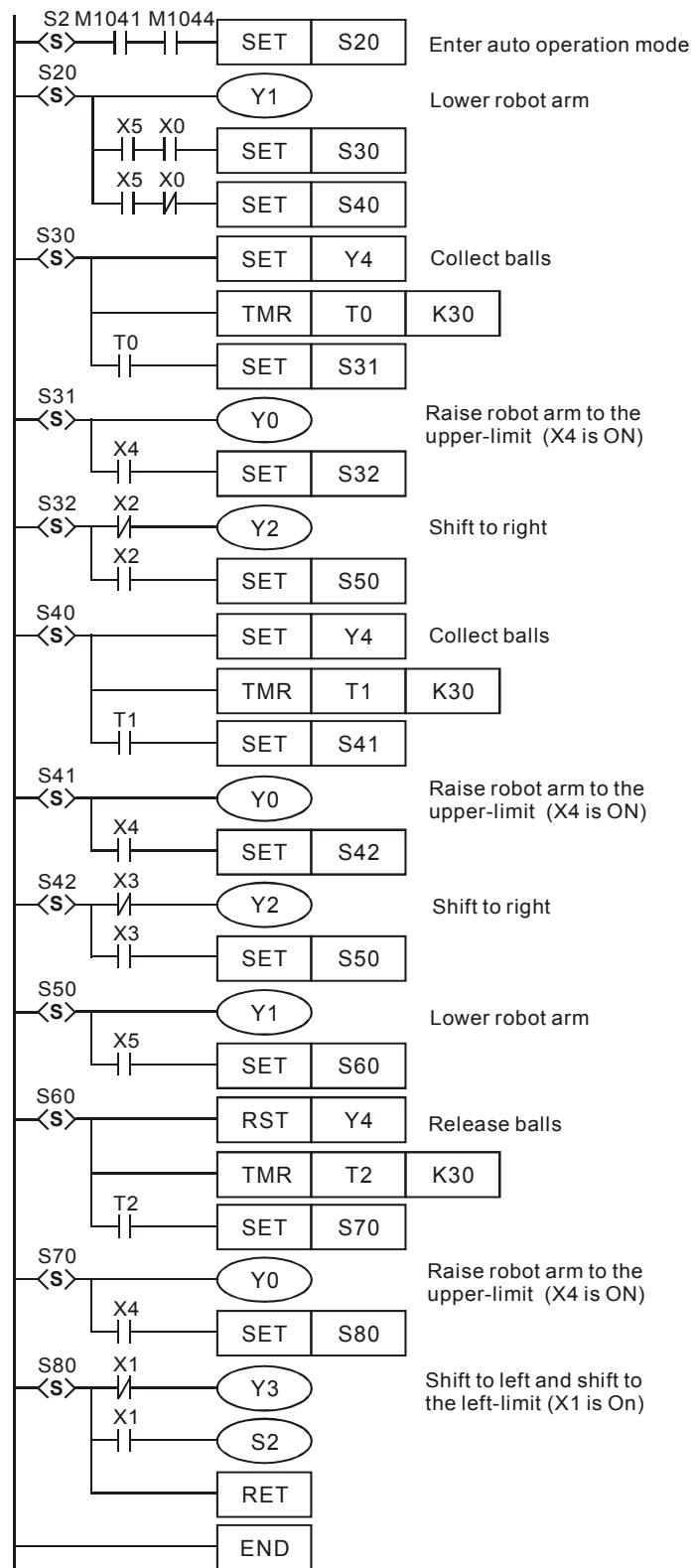


8. Auto operation (step/one-cycle/continuous operation modes):

a) SFC Diagram:



b) Step Ladder Diagram:



Flag explanation:

M1040:

Step point movement disabled. When M1040=ON, all movements of the step point are disabled.

1. Manual operation mode: M1040 = ON.
2. Zero point return mode/one cycle operation mode: Pressing the STOP button and pressing START button again, M1040 = ON.
3. Step operation mode: M1040 = ON, and will only be OFF when the START button is pressed.
4. Continuous operation mode: When ELC goes from STOP→RUN, M1040 = ON, and will be OFF when the START button is pressed.

M1041:

Step point movement start: the special auxiliary relay that reflects the movement of the primary step point (S2) to the next step point.

1. Manual operation mode/Zero point return mode: M1041 = OFF.
2. Step operation mode/One cycle operation mode: M1041 = OFF when the START button is pressed.
3. Continuous operation mode: Stays ON when the START button is pressed, and turns OFF when the STOP button is pressed.

M1042:

START pulse: Only one pulse will be sent out when the button is pressed.

M1043:

Zero point return complete: Once M1043 =ON, indicates that the RESET motion has been executed.

M1044:

Conditions of the origin: In continuous operation mode, conditions of the origin, M1044= ON to execute the initial step point (S2) moving to the next step point.

M1045:

1. All output reset inhibit. If executing conditions:
 - a) From manual control S0 to zero point return S1
 - b) From auto operation S2 to manual operation S0
 - c) From auto operation S2 to zero point return S1
2. When M1045=OFF and one of S of D1~D2 is ON, step point of SET Y output and actions will be cleared to OFF.

3. When M1045 =ON, SET Y output will be reserved, and step point during action will be cleared to OFF.
4. If executing from zero point return S1 to manual operation S0, no matter if M1045=ON or M1045=OFF, SET Y output will be reserved, and step point action will be cleared to OFF.

M1046:

Set STL = ON: If one of step point S is ON, M1046=ON. After M1047 = ON, M1046 = ON once one of S is ON. Besides, 8 points numbers before S is on will be recorded in D1040~D1047.

M1047:

STL monitor enabled. When IST instruction starts executing, M1047 will be forced to be ON and it will be forced to ON for each scan time once IST instruction is still ON. This flag is used to monitor all S.

D1040~D1047: ON state number 1-8 of step point S.

Points to Note:

1. Operand S will occupy 8 continuous devices.
2. The range D_1 and $D_2 = S20 \sim S899$ and $D_1 < D_2$.
3. IST instruction can only be used one time in a program.
4. Flag: M1040~M1047.

MEMO

8

Troubleshooting

This chapter contains information on troubleshooting the ELC.

This Chapter Contains

9.1	Common Problems and Solutions	9-2
9.2	Fault Code Table	9-4
9.3	Error Detection Devices	9-6
9.4	Periodic Inspection	9-6

9 Fault Indication From Panel

9.1 Common Problems and Solutions

The following tables list some common problems and troubleshooting procedures for the ELC system in the event of faulty operation.

System Operation

Symptom	Troubleshooting and Corrective Actions
All LEDs are OFF	<p>Check the power supply wiring. If the power supplied to the ELC control units is in the range of the rating.</p> <p>Be sure to check the fluctuation in the power supply.</p> <p>Disconnect the power supply wiring to the other devices if the power supplied to the ELC control unit is shared with them.</p> <p>If the LEDs on the ELC control unit turn ON at this moment, the capacity of the power supply is not enough to control other devices as well. Prepare another power supply for other devices or increase the capacity of the power supply.</p> <p>If the POWER LED still does not light up when the power is on after the above corrective actions, the ELC should be sent back to the dealer or the distributor whom you purchased the product from.</p>
ERROR LED is flashing	<p>If the ERROR LED is flashing, the problem may be an invalid commands, communication error, invalid operation, or missing instructions, error indication is given by self-checking function and corresponding error code and error step are stored in special registers. The corresponding error codes can be read from the ELCSoft or HHP. Error codes and error steps are stored in the following special registers.</p> <p>Error code: D1004</p> <p>Error step: D1137</p> <p>If the connections between the ELC are failed and the LED will flash rapidly, this indicates the DC24V power supply is down and please check for possible DC24V overload.</p> <p>The LED will be steady if the program loop execution time is over the preset time (D1000 preset value), check the programs or the WDT (Watch Dog Timer). When the LED lights up, switch the power ON and OFF to see if the RUN LED is off. If not, please check if there is any noise interference or any foreign object in the ELC.</p>

Symptom	Troubleshooting and Corrective Actions
Diagnosing Input Malfunction	<p>Check the wiring of the input devices(input indicator LEDs are OFF)</p> <p>Check that the power is properly supplied to the input terminals.</p> <p>If the power is properly supplied to the input terminal, there is probably an abnormality in the ELC's input circuit. Please contact your dealer.</p> <p>If the power is not properly supplied to the input terminal, there is probably an abnormality in the input device or input power supply. Check the input device and input power supply.</p> <p>Check the input condition (input indicator LEDs are ON)</p> <p>Monitor the input condition using a programming tool.</p> <p>If the input monitored is OFF, there is probably an abnormality in the ELC's input circuit. Please contact your dealer.</p> <p>If the input monitored is ON, check the program again. Also, check the leakage current at the input devices (e.g., two-wire sensor) and check for the duplicated use of output or the program flow when a control instruction such as MC or CJ is used.</p> <p>Check the settings of the I/O allocation.</p>
Diagnosing Output Malfunction	<p>Check the wiring of the loads. (output indicator LEDs are ON)</p> <p>Check if the power is properly supplied to the loads.</p> <p>If the power is properly supplied to the load, there is probably an abnormality in the load. Check the load again. If the power is not supplied to the load, there is probably an abnormality in the ELC's output circuit. Please contact your dealer.</p> <p>Check of output condition (output indicator LEDs are OFF)</p> <p>Monitor the output condition using a programming tool.</p> <p>If the output monitored is turned ON, there is probably a duplicated output error.</p> <p>Forcing ON the output using a programming tool.</p> <p>If the output indicator LED is turned ON, go to input condition check.</p> <p>If the output LED remains OFF, there is probably an abnormality in the ELC's output circuit. Please contact your dealer.</p>

9.2 Fault Code Table

Fault Code	Description	Action
0001	Operand bit device S exceeds the usage range	Check the D1137 (Error step number) Re-enter the instruction correctly
0002	Label P exceeds the usage range or duplicated	
0003	Operand KnSm exceeds the usage range	
0102	Interrupt pointer I exceeds the usage range or duplicated	
0202	Instruction MC exceeds the usage range	
0302	Instruction MCR exceeds the usage range	
0401	Operand bit device X exceeds the usage range	
0403	Operand KnXm exceeds the usage range	
0501	Operand bit device Y exceeds the usage range	
0503	Operand KnYm exceeds the usage range	
0601	Operand bit device T exceeds the usage range	
0604	Operand word device T register usage exceeds limit	
0801	Operand bit device M exceeds the usage range	
0803	Operand KnMm exceeds the usage range	
0D01	DECO Misuse operand	
0D02	ENCO Misuse Operand	
0D03	DHSCS Misuse Operand	
0D04	DHSCR Misuse Operand	
0D05	PLSY Misuse Operand	
0D06	PWM Misuse Operand	
0D07	FROM / TO Misuse Operand	
0D08	PID Misuse Operand	
0D09	SPD Misuse Operand	
0E01	Operand bit device C exceeds the usage range	
0E04	Operand word device C register usage exceeds limit	
0E05	DCNT misuse operand C	
0E18	BCD Conversion Error	
0E19	Division (divisor=0)	
0E1A	Floating Point exceeds usage range	
0E1B	It is negative number after radical expression	
0E1C	FROM/TO communication error	
0F04	Operand word device D register usage exceeds limit	
0F05	DCNT misuse operand D	

Fault Code	Description	Action
0F06	SFTR misuse operand	Check the D1137 (Error step number)
0F07	SFTL misuse operand	
0F08	REF Misuse Operand	
1000	ZRST misuse operand	Re-enter the instruction correctly
2000	Usage exceed limit (MTR, ARWS, HOUR)	

Fault Code	Description	Action
C400	An unrecognized instruction code is being used	<p>A circuit error occurs if a combination of instructions is incorrect or badly specified.</p> <p>Select programming mode and correct the identified error</p>
C401	Loop Error	
C402	LD / LDI continuously use more than 9 times	
C403	MPS continuously use more than 9 times	
C404	FOR-NEXT exceed 6 levels	
C405	STL / RET used between FOR and NEXT SRET / IRET used between FOR and NEXT MC / MCR used between FOR and NEXT END / FEND used between FOR and NEXT	
C407	STL continuously use more than 9 times	
C408	Use MC / MCR in STL, Use I / P in STL	
C409	Use STL / RET in Subroutine, Interrupt Service Routine STL / RET	
C40A	Use MC / MCR in Subroutine, Interrupt Service Routine MC / MCR	
C40B	MC / MCR does not begin from N0 or discontinuously	
C40C	MC / MCR corresponding value N is different	
C40D	Use I / P incorrectly	
C40E	IRET does not follow by the last FEND command SRET does not follow by the last FEND command	
C41C	The number of input/output points of I/O extension unit is larger than the specified limit	
C4EE	No END command in the program	

9.3 Error Detection Devices

Error Check Devices	Description	Drop Latch	STOP → RUN	RUN → STOP
M1067	Program execution error flag	None	Reset	Latch
M1068	Execution error latch flag	None	Latch	Latch
D1067	Algorithm error code	None	Reset	Latch
D1068	Step value of algorithm errors	None	Latch	Latch

Device D1067 Error Code	Description
0E18	BCD Conversion Error
0E19	DIVISION (divisor=0)
0E1A	Floating Point exceeds the usage range
0E1B	The value of square root is negative

9.4 Periodic Inspection

1. Preventive maintenance is required to operate this ELC in its optimal condition, and to ensure a long life. Be sure to observe the following precautions when selecting a mounting location. Failure to observe these precautions may void the warranty!
2. Do not mount the ELC near heat-radiating elements or in direct sunlight.
3. Do not install the ELC in a place subjected to high temperature, high humidity, excessive vibration, corrosive gasses, liquids, airborne dust or metallic particles.
4. Periodically check if the wiring and terminals are tight.

Handheld Programmer ELC-HHP



This chapter contains information regarding the handheld programmer.

This Chapter Contains

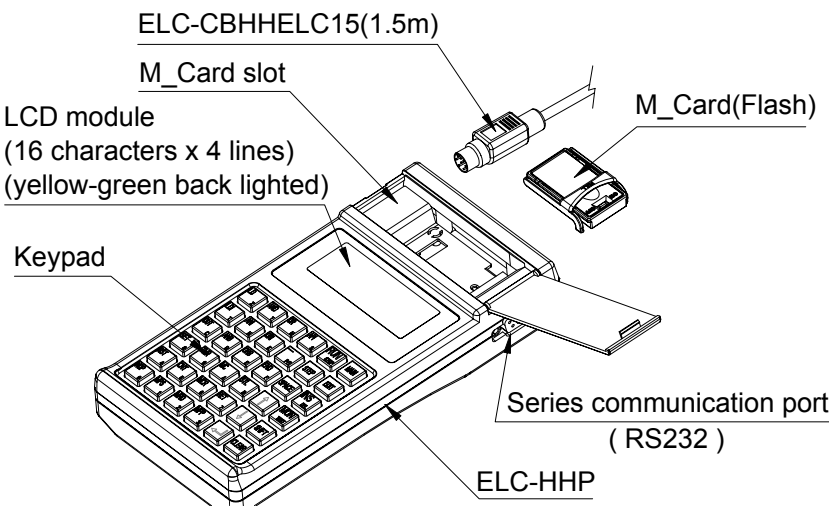
10.1	Introduction.....	10-2
10.2	ELC-HHP Standard Specification.....	10-4
10.2.1	ELC-HHP Dimensions	10-4
10.2.2	Specifications.....	10-4
10.3	Initial Setup	10-6
10.3.1	ELC ON-LINE Mode	10-7
10.3.2	PC ON-LINE Mode	10-9
10.3.3	ELC-HHP OFF-LINE Mode	10-12
10.4	ELC-HHP Program Read / Write.....	10-14
10.4.1	ELC Program Read / Write.....	10-14
10.4.2	PC Program READ / WRITE	10-18
10.4.3	M_CARD READ / WRITE and Password Settings.....	10-22
10.5	Program Mode	10-26
10.5.1	Screen Display	10-26
10.5.2	Basic Instruction Input	10-26
10.5.3	Application Instruction Input	10-27
10.5.4	Instruction INSERT / DELETE.....	10-29
10.5.5	[STEP] Function	10-29
10.5.6	REPLACE Function	10-31
10.5.7	Searching Application Instruction	10-32
10.5.8	Searching for the API Codes	10-32
10.5.9	Indirect Index Register Application	10-34
10.6	ELC RUN / STOP Mode	10-36
10.7	MON / TEST Mode	10-37
10.8	Clear User's Program Memory.....	10-43
10.9	ELC and ELC-HHP Program Verifications.....	10-45
10.10	Parameters Settings.....	10-47
10.11	M_CARD Functions.....	10-50
10.12	File Register.....	10-51
10.13	Error Code Explanations	10-54
10.14	Instruction Table.....	10-57
10.15	Troubleshooting and Error Message.....	10-63
10.16	ELC-HHP Connection.....	10-64
10.17	ELC-HHP Operation Flow Chart.....	10-65

10 Handheld Programmer

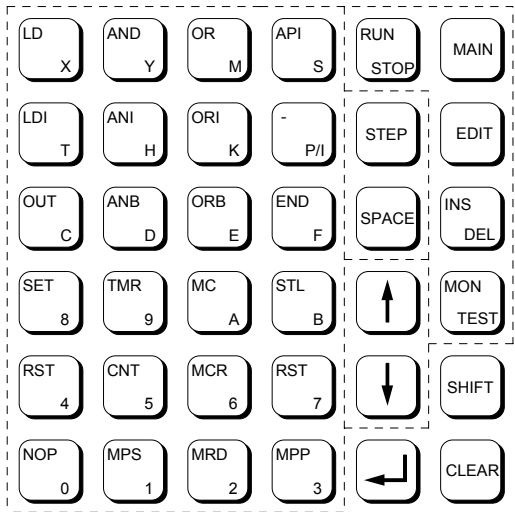
10.1 Introduction

The Handheld Programmer (ELC-HHP) is a powerful programming and monitor panel. Please refer to the following figure for detail.

The programs store in the ELC-HHP contain power loss retain function, when ELC-HHP connects to MPU (the connection time is more than 120 seconds) and then disconnect, the data can be preserved in the ELC-HHP for more than 3 days. Memory can be selected from 2000, 4000, 8000, 16000 STEPs as user defined memory (factory default is 4000 STEPs).



Keypad Arrangement





MAIN MENU: Pressing the key calls the MAIN MENU select screen, regardless of the current display mode.



EDIT: Enter programming edit mode. this key is the same as the selection 1 PGM EDIT in the MAIN MENU.



INSERT/DELETE: this is a toggle key. Pressing this key once calls the INSERT function. Pressing it again calls the DELETE function.



MONITOR/TEST: this is a toggle key. Pressing this key once calls the MONITOR function. Monitoring the content of every output points, counters, timer and data registers, yet it can not perform the forced I/O. Pressing it again, calls the TEST function, it will perform the forced I/O function.

Pressing this key will run the monitor function during editing programs.



SHIFT: In MON mode, pressing this prior to enter any monitor instruction. Pressing this key again, prior to enter any new monitored component. In EDIT mode, this key can perform as “replacement” function.



CLEAR: clear the most recent input instruction, and return to the previous instruction.



RUN/STOP: forced RUN/STOP instructions.



STEP: this key is used to input step numbers. Refer to Section 10.5.5



SPACE: this key is used whenever a device or a constant is to be input.



Cursor [↑] [↓] keys: the cursor keys are used to select the items in the EDIT, INS, DEL and MON modes or go to the previous/next pages in the MAIN MENU.



ENTER: The [↵] key is used to enter and execute instructions, to scroll displayed information, or to continue a search.

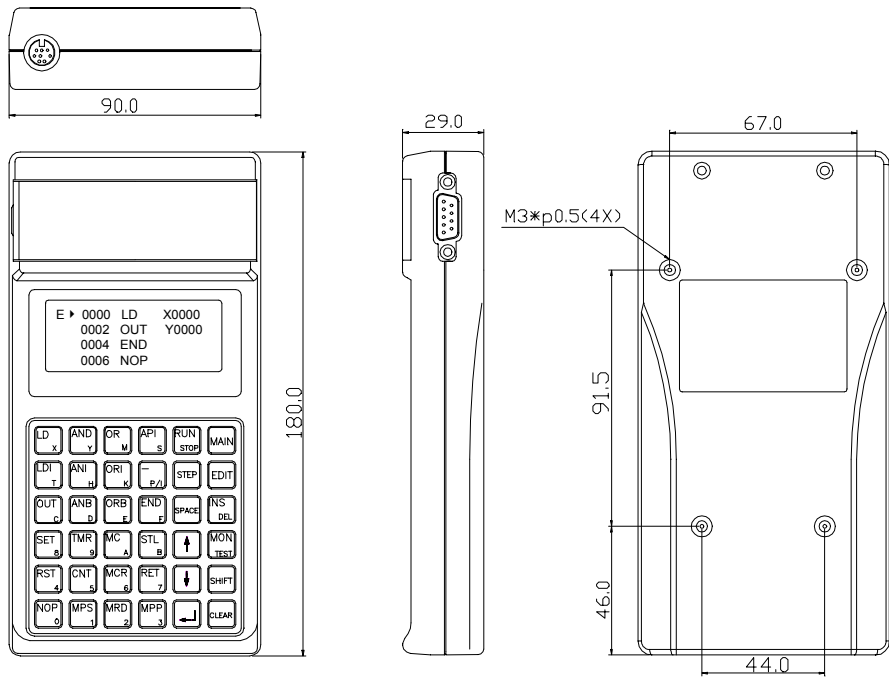
**Instruction,
Devices, and
Number keys**

Instruction keys are located at the left-upper corner, devices and number keys are located at the down-right corner.

10.2 ELC-HHP Standard Specification

10.2.1 ELC-HHP Dimensions

(Unit: mm)



10.2.2 Specifications

Items	Specification		
Ambient Temperature	0°C to 40°C		
Ambient Humidity	50~95 % RH		
Vibration	Frequency	Acceleration	Amplitude
	10 to 55Hz	1G	0.1 mm (0.004 in)
	2 hours each in X/Y/Z directions		
Shock Resistance	10G, 3 times each in X/Y/Z directions		
Environment	No corrosive gasses, liquid, or airborne dust		
Supply Voltage	5V DC $\pm 5\%$ (Supply from MPU)		
Current Consumption	300mA (normal), 400mA (momentary Max)		
Start Current	500mA MAX		
Start Setting Time	0.5 seconds		
User Memory Capacity	4000 / 8000 / 16000 STEPS		
Latched Memory	There is a flash card to preserve internal program permanently and set the function of passwords.		

Items		Specification	
Memory Backup for Power Failure		Memory backup capacitor After power is on for 120 seconds, it can retain internal device data for 3 days without externally supplied power.	
Display Unit		LCD with backlight	
Display Contents	Character Matrix	1 character: 40 dots (8 x 5) Bottom line: 5 dots (1 x 5 dots) are used for prompt.	
	Number of Characters	64 characters (16 columns x 4 lines)	
	Character Types	Alphanumeric	
File Register		For PC/PA/PH series to save register	
Keypad		36 keys	
Build-in Interface		Used ELC-CBHHELC15 cable to connect to the MPU	
Series Communication Interface		Built-in RS-232	
		RS-232	Using for PC↔HHP and ELC↔HHP transmission. Please refer to Section 10.3 for detail.
External Dimensions		180 x 90 x 29 mm	
Weight		265.5g	

Three working mode of ELC-HHP are:

- Please refer to the following for detail.

Using ELC-CBHHELC15 cable to connect ELC and ELC-HHP or using external power +5V to supply power. Because the ELC-HHP does not have its own power supply, power is supplied from the MPU through the programming cable. The initial screen will display the version (VER □.□) for 2 seconds. At this moment, the ELC-HHP is self-test.

After self-test, it will enter the next mode. If the display is shown as left, it means ELC-HHP is malfunction. Please return it to your distributor.

After finishing self-test, ELC-HHP will test the PC connection.
To make sure PC is in connection.

In PC ON-LINE mode, main menu will be shown as left.

If it can't connect to PC, ELC-HHP will test if it connects to ELC.
To make sure ELC is in connection.

Please execute the ELC ON-LINE mode to display main menu.

```

OFF-LINE MODE
EXECUTE?
YES →[↵]
NO →[CLEAR]

```

If the connection to PC and ELC is failed, it will show as left.

Pressing [↵] to enter OFF-LINE mode.

Pressing [CLEAR] to execute connection testing between PC and ELC. If the connection is successful, it will display main menu. Otherwise, it will back to the OFF-LINE MODE.

```

MAIN MENU (↑,↓)
■0.ELC↔HHP
1.PGM EDIT
2.MONITOR/TEST

```

The main menu in OFF-LINE MODE is the same as ELC connection mode, but some items are disabled in this mode.

10.3.1 ELC ON-LINE Mode

Connecting ELC-HHP to ELC in ELC ON-LINE MODE, you can read ELC program from ELC-HHP or write ELC-HHP program into ELC. Please refer to the following explanation.

```

MAIN MENU (↑,↓)
■0.ELC↔HHP
1.PGM EDIT
2.MONITOR/TEST

```

After entering to the MAIN MENU, using [↓] and [↑] keys to select the desired function, or press number keys 0 to 8. Press [↵] key when complete.

```

3.PGM CLEAR
4.PGM CHECK
5.PGM VERIFY
6.PARAMETER SET
7.HHP↔MLCARD
8.FILE REGISTER

```

Definitions of items 0 to 8:

0.ELC↔HHP:

It is program transmission between ELC-HHP and ELC (read and write). After entering, ELC will check if there is password first. If yes, you should input the correct password to enter the following items.

1. HHP←ELC:

To download the programs from ELC to ELC-HHP user programming area (Read). You can choose to read the segment program (end with the END instruction) or the whole program. And setting the parameter PGM CAPACITY of ELC-HHP to the same program capacity as MPU.

2. HHP→ELC:

Writing the program from ELC-HHP to ELC (Write). ELC-HHP will check the capacity first. If ELC-HHP setting is large than ELC, it will have the message of “the capacity is mismatch”. Otherwise, it can write and you can only write segment program (end with the END instruction) or the whole program.

1.PGM EDIT:

Entering the EDIT mode by selecting item 1 from the MAIN MENU or press EDIT key in the keypad. ELC-HHP will set edition range according to the parameter PGM CAPACITY.

- 4K: 3792 STEPS, for PB model
- 8K: 7920 STEPS, for PC/PA/PH model
- 16K: 15872 STEPS, reserved

2.MONITOR/TEST:

Entering the MON/TEST mode by selecting number 2 from the MAIN MENU or MON/TEST key in the keypad.

1. MONITOR: monitor ELC internal components status and content.
2. TEST mode: setting ELC internal components status and content.

3.PGM CLEAR:

To clear the content of user program or memory card.

1. ALL CLEAR: To clear the entire user program (16K STEPS). There is no correlation between this setting and PGM CAPACITY setting.
2. SEGMENT CLEAR: To clear partial of the user program. (You can input the start address and the end address.)
3. M_CARD CLEAR: To clear the entire memory card content. The password in M- CARD will also be clear.

4.PGM CHECK:

To execute grammar analysis and check if the user's programs are correct.

5.PGM VERIFY:

Program comparison.

1. HHP : ELC: It will compare ELC-HHP user's programs with ELC internal programs and stop at the instruction "END". After comparing, parameter PGM CAPACITY will set to the same program capacity as MPU.
2. HHP : M_CARD: It will compare ELC-HHP user's programs with M_CARD internal programs and stop at the instruction "END".

6.PARAMETER SET:

Parameter setting.

1. PGM CAPACITY: To display the ELC-HHP internal user programming capacity and modify its capacity for all ELC models. (Users can select 4000 or 8000 or 16000 STEPS).
2. OS VERSION: To display ELC-HHP operation system version.
3. PASSWORD: To set or remove the MPU password.

7.ELC↔M_CARD:

ELC-HHP will check if it needs password. If yes, you should input the correct password to enter the following item.

1. HHP←M_CARD: Reading the M_CARD internal program to ELC-HHP user programming area. It will use all memory (16K) and the setting of parameter PGM CAPACITY doesn't have any change.
2. HHP→M_CARD: Writing the program of ELC-HHP user programming area (16K) all into M_CARD. (No matter what the setting of PGM CAPACITY is.) Before writing into M_CARD, you can choose to set password or not.
3. M_CARD CHECK: Check sum of M_CARD.

8.FILE REGISTER:

The file register for transmitting files between ELC-HHP and –PC/PA/PH series.

1. HHP←ELC: To download the file from ELC to ELC-HHP (read).
2. HHP→ELC: To write the file from ELC-HHP to ELC (write).

10.3.2 PC ON-LINE Mode

User can connect ELC-HHP to PC (personal computer) in PC ON-LINE MODE and transmit data from ELC-HHP to PC (ELCSoft) and vice versa. ELC-HHP is master and PC is slave during transmission.

The connection is shown below. External power +5V is a power supplying used in PC ON-LINE MODE or in **OFF-LINE MODE**.

Connection:

Input power of external power +5V. It is used to provide the power for ELC-HHP.

Step1:

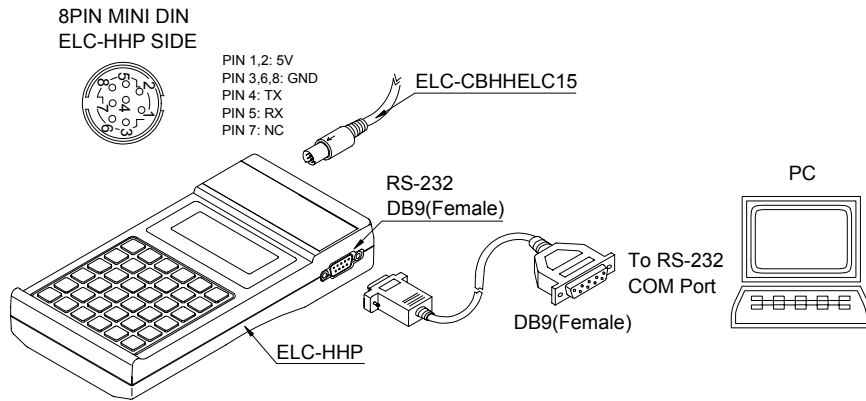
Please make sure that PC ELCSoft (for WINDOWS) is in the status of waiting to connect to ELC-HHP.

Step2:

Connecting ELC-HHP to PC with RS-232.

Step3:

Plug in CABLE as following.



After completing connection, entering main menu to set.

```
MAIN MENU (↑,↓)
0. PC<->HHP
1. PGM EDIT
2. MONITOR/TEST
```

After entering to the MAIN MENU, using [↓] and [↑] keys to select the desired item, or press number keys 0 to 8. Pressing [↵] key and then press [↓] or [↑] key to select next item.

```
3. PGM CLEAR
4. PGM CHECK
5. PGM VERIFY
6. PARAMETER SET
7. HHP<->MLOAD
8. FILE REGISTER
```

Definitions of items 0 to 8:

0.PC↔HHP:

It is program transmission between ELC-HHP and ELC (read and write).

1. HHP←PC: To download the programs from PC to ELC-HHP user programming area (Read). You can only read segment program (end with the END instruction) or the whole program.
2. HHP→PC: Entering the user programs from ELC-HHP to PC (Write). You can only write segment program (end with the END instruction) or the whole program.

1.PGM EDIT:

Entering the EDIT mode by selecting item 1 from the MAIN MENU or EDIT key in the keypad.

2.MONITOR/TEST:

It is disabled in the PC ON-LINE MODE.

3.PGM CLEAR: To clear the content of user program or memory card.

1. ALL CLEAR: To clear the entire user program.
2. SEGMENT CLEAR: To clear partial of the user program. (You can input the start address and the end address.)
3. M_CARD CLEAR: To clear the entire memory card. The passwords will also be clear.

4.PGM CHECK:

To execute grammar analysis and check if the user's programs are correct.

5.PGM VERIFY:

Program comparison.

1. HHP : ELC: It is disabled in the PC ON-LINE MODE.
2. HHP : M_CARD : It will compare ELC-HHP user's programs with M_CARD internal programs.

6.PARAMETER SET:

Parameter setting.

1. PGM CAPACITY: To display the ELC-HHP internal user programming capacity and modify ELC-HHP user's programs for all ELC models. (Users can select 4000 or 8000 or 16000 STEPS).
2. OS VERSION: Displaying ELC-HHP operation system version.
3. PASSWORD: It is disabled in PC ON-LINE MODE.

7.ELC↔ M_CARD:

HHP will check if it needs password. If yes, you should input the correct password to enter the following items.

1. HHP←M_CARD: Reading the M_CARD internal program to ELC-HHP user programming area. It will use all the memory (16K) but parameter PGM CAPACITY doesn't have any change.
2. HHP→M_CARD: Writing the program of ELC-HHP user programming area (16K) into M_CARD. (No matter what the setting of PGM CAPACITY is.) Before writing, you can choose to set password or not.
3. M_CARD CHECK: Check sum of M_CARD.

8.FILE REGISTER:

The register for transmitting file between ELC-HHP and ELCSOFT.

1. HHP←ELC: To download the file from PC (edited by ELCSOFT) to ELC-HHP. (Read)
2. HHP→ELC: To write the file from ELC-HHP to PC (ELCSOFT).

10.3.3 ELC-HHP OFF-LINE Mode

ELC-HHP OFF-LINE mode: ELC-HHP can execute partial functions in OFF-LINE mode. It doesn't connect to PC or MPU now and the power supply is from external power +5V.

```
MAIN MENU (↑,↓)
■0.ELC<->HHP
 1.PGM EDIT
 2.MONITOR/TEST
```

After entering to the MAIN MENU, press [↓] and [↑] keys to select the desired item, or press number keys 0 to 8. Then pressing [↵] key to enter the item.

```
3.PGM CLEAR
4.PGM CHECK
5.PGM VERIFY
6.PARAMETER SET
7.HMP<->MLCARD
8.FILE REGISTER
```

Definitions of items 0 to 8:

0.ELC↔HHP:

It is disabled in ELC-HHP OFF-LINE MODE.

1.PGM EDIT:

Entering the EDIT mode by selecting item 1 from the MAIN MENU or EDIT key in the keypad.

2.MONITOR/TEST:

It is disabled in ELC-HHP OFF-LINE MODE.

3.PGM CLEAR:

To clear the content of user program or memory card.

1. ALL CLEAR: To clear the entire user program content.
2. SEGMENT CLEAR: To clear partial of the user program. (You can input the start address and the end address.)
3. M_CARD CLEAR: To clear the entire memory card content. The passwords will also be cleared.

4.PGM CHECK:

To execute grammar analysis and check if the user's programs are correct.

5.PGM VERIFY:

Program comparison.

1. HHP: ELC: It is disabled in ELC-HHP OFF-LINE MODE.
2. HHP: M_CARD: To compare ELC-HHP user's programs with M_CARD internal programs.

6.PARAMETER SET:

Parameter setting.

1. PGM CAPACITY: To display the ELC-HHP internal user programming capacity and modify ELC-HHP user's programs for all ELC models. (Users can select 4000 or 8000 or 16000 STEPS).
2. OS VERSION: To display ELC-HHP operation system version.
3. PASSWORD: It is disabled in ELC-HHP OFF-LINE MODE.

7.ELC↔M_CARD:

ELC-HHP will check if it needs password first. If yes, you should input the correct password to enter the following items.

1. HHP←M_CARD: Reading internal program from the M_CARD to ELC-HHP user programming area. It will use all the memory (16K), but parameter PGM CAPACITY doesn't have any change.
2. HHP→M_CARD: Writing the program from ELC-HHP user programming area into M_CARD. (No matter what the setting of PGM CAPACITY is) Before writing, you can choose to set password or not.
3. M_CARD CHECK: Check sum of M_CARD.

8.FILE REGISTER:

It is disabled in OFF-LINE mode.

10.4 ELC-HHP Program Read / Write

There are three parts of ELC-HHP program Read / Write.

1. ELC-HHP to ELC MPU
2. ELC-HHP to PC
3. ELC-HHP to M_CARD

10.4.1 ELC Program Read / Write

Please refer to the following steps for operating in ELC connection mode.

Read segment program (ELC→ELC-HHP)

```
MAIN MENU (↑,↓)
■0.ELC<->HHP
  1.PGM EDIT
  2.MONITOR/TEST
```

To move the cursor to item 0 and press [↵] or [NOP/0].

Entering the transmission between ELC-HHP and ELC.

```
INPUT:[↑,↓]
■1.HHP<-ELC
  2.HHP->ELC
```

To move the cursor to item 1 and press [↵] or [MPS/1].

Entering the READ mode.

```
INPUT:[↑,↓]
■1.SEGMENT PGM
  2.ALL PGM
```

To move the cursor to item 1 and press [↵] or [MPS/1].

Entering the segment program mode.

```
PROGRAM READ
  SEGMENT
YES →[↵]
NO  →[CLEAR]
```

Pressing [CLEAR] key to cancel the “program read segment” function and return to the previous step.

Pressing [↵] key to execute “program read segment” function.

```
PROGRAM READ
  SEGMENT
■■■■
```

Entering program read segment mode, and it is reading.

```
PROGRAM READ
  HHP<-ELC
  COMPLETED
RETURN → [↵]
```

Reading from ELC program, which address is from 0 to END instruction, to ELC-HHP.

After reading, press [↵] key to return to main menu.

Read the whole program (ELC→ELC-HHP)

```

MAIN MENU  (↑,↓)
■0.ELC<->HHP
 1.PGM EDIT
 2.MONITOR/TEST

```

To move cursor to item 0 and press [↵] or **[NOP/0]**

Entering the program transmission between ELC-HHP and ELC.

```

INPUT:[↑,↓]
■1.HHP<-ELC
 2.HHP->ELC

```

To move cursor to item 1 and press [↵] or **[MPS/1]**

Entering the READ mode.

```

INPUT:[↑,↓]
 1.SEGMENT PGM
■2.ALL PGM

```

To move cursor to item 2 and press [↵] or **[MRD/2]**

Entering "read all program" mode.

```

PROGRAM READ
  ALL
YES →[↵]
NO  →[CLEAR]

```

Pressing **[CLEAR]** key to cancel "program read all" function and return to the previous step.

Pressing [↵] key to enter "program read all" function.

```

PROGRAM READ
  ALL

▶▶▶▶▶▶▶▶

```

Entering program read ALL mode and it is reading.

```

PROGRAM READ
  HHP<-ELC
  COMPLETED
  RETURN → [↵]

```

Reading all ELC program from ELC to ELC-HHP.

Pressing [↵] key to return to menu when completed.

Write segment ELC program (ELC-HHP→ELC)

```

MAIN MENU  (↑,↓)
■0.ELC<->HHP
 1.PGM EDIT
 2.MONITOR/TEST

```

To move cursor to item 0 and press [↵] or **[NOP/0]**.

Entering the program transmission between ELC-HHP and ELC.

```

INPUT:[↑,↓]
 1.HHP<-ELC
■2.HHP->ELC

```

To move cursor to item 2 and press [↵] or **[MRD/2]**.

Entering the WRITE mode.

```
PGM CAPACITY
** MISMATCH **
MODIFY HPP? → [↵]
EXIT → [CLEAR]
```

If the message as shown in the left is displayed, it means the capacity of ELC-HHP and ELC are mismatch.

Pressing [↵] key to modify the capacity of ELC-HHP to be the same as ELC MPU and enter “WRITE” mode.

Pressing [CLEAR] key to cancel and return to the previous page. And user also can enter item 6 “PARAMETER SET” to reset the capacity of user program of ELC-HHP. Please refer to Section 10.10 for detail.

```
INPUT:[↑, ↓]
■ 1.SEGMENT PGM
  2.ALL PGM
```

To move cursor to item 1 and press [↵] or [MPS/1].

Entering SEGMENT PROGRAM MODE.

```
PROGRAM WRITE
  SEGMENT
YES → [↵]
NO → [CLEAR]
```

Pressing [CLEAR] key to cancel “write segment program” function, and return to the previous step.

Pressing [↵] key to execute “write segment program” function.

```
PROGRAM WRITE
  SEGMENT
EXIT → [CLEAR]
▶▶▶▶▶
```

Writing ELC-HHP segment program into ELC.

Entering the WRITE mode and the data is writing.

Pressing [CLEAR] key to break the WRITE function

```
PROGRAM WRITE
  SEGMENT
  COMPLETED
RETURN → [↵]
```

Writing the ELC-HHP program, which locates from address 0 to END instruction, into ELC.

Pressing [↵] key to return to menu when completed.

```
PROGRAM WRITE
PROGRAM ERROR
CODE: C400
RETURN → [↵]
```

If the display unit shows the message as the left during the WRITE mode, which means program error (the ELC ERROR LED will flash at this moment). You can check the ERROR CODES on Section 10.13 to find out the exact error.

Pressing [↵] key to return to main menu.

Write all ELC program (ELC-HHP→ELC)

```
MAIN MENU (↑, ↓)
■ 0.ELC<->HHP
  1.PGM EDIT
  2.MONITOR/TEST
```

To move cursor to item 0 and press [↵] or [NOP/0].

Entering the program transmission between ELC-HHP and ELC.

```

INPUT:[ ↑ , ↓ ]
1.HHP<-ELC
■2.HHP->ELC

```

To move cursor to item 2 and press [↓] or [MRD/2].

Entering the WRITE mode.

```

PGM CAPACITY
** MISMATCH **
MODIFY HPP?→ [↵]
EXIT → [CLEAR]

```

If the message as shown in the left is displayed, it means the capacity of ELC-HHP and ELC are mismatch.

Pressing [↓] key to modify the capacity of ELC-HHP to be the same as ELC MPU and enter "WRITE" mode.

Pressing [CLEAR] key to cancel and return to the previous page. And user also can enter item 6 "PARAMETER SET" to reset the capacity of user program of ELC-HHP. Please refer to Section 10.10 for detail.

```

INPUT:[ ↑ , ↓ ]
1.SEGMENT PGM
■2.ALL PGM

```

To move cursor to item 2 and press [↓] or [MRD/2]

Entering the "write all program" MODE.

```

PROGRAM WRITE
ALL
YES →[↵]
NO →[CLEAR]

```

Pressing [CLEAR] key to cancel "program write all" function and return to the previous step.

Pressing [↓] key to execute "program write all" function.

```

PROGRAM WRITE
ALL
EXIT → [CLEAR]
▶▶▶▶▶▶▶▶

```

Writing all ELC-HHP program into ELC.

Entering the WRITE mode and the data is writing.

Pressing [CLEAR] key to break the WRITING.

```

PROGRAM WRITE
* PROGRAM ERROR*
CODE:C400
RETURN → [↵]

```

If the display unit shows the message as the left during the WRITE mode, which means program error (the ELC ERROR LED will flash at this moment). You can check the ERROR CODES on Section 10.13 to find out the exact error.

Pressing [↓] key to return to main menu.

```

CANNOT WRITE-IN
WHEN ELC IS
RUNNING
EXIT → [CLEAR]

```

If the display unit shows the message as the left, it indicates that MPU is running. At this moment, it can't write and you must stop the ELC first. Please refer to the Section 10.5.

Pressing [CLEAR] key to return to main menu.

```

PROGRAM VERIFY
* VERIFY ERROR *
ON LINE=0001
RETURN → [↵]

```

If the display unit shows the message as the left during the WRITE mode, which means partial data are incorrect or can't be written. Repeat the steps, if it appears again, which means the MPU is malfunction.

Pressing [↵] key to return to main menu.

```

PROGRAM WRITE
HHP→ELC
COMPLETED
RETURN → [↵]

```

To complete to Programs WRITE. Pressing [↵] key to return to main menu.

10.4.2 PC Program READ / WRITE

To set the connection item to ELC-HHP in the ELCSOFT of PC in the PC ON-LINE MODE. Following is the basic operation.

READ partial PC program (PC→ELC-HHP)

```

MAIN MENU (↑,↓)
■0.PC↔HHP
1.PGM EDIT
2.MONITOR/TEST

```

To move cursor to item 0 and press [↵] or [NOP/0]

Entering the program transmission between ELC-HHP and PC.

```

INPUT:[↑,↓]
■1.HHP↔PC
2.HHP→PC

```

To move cursor to item 1 and press [↵] or [MPS/1]

Entering the READ mode.

```

INPUT:[↑,↓]
■1.SEGMENT PGM
2.ALL PGM

```

To move cursor to item 1 and press [↵] or [MPS/1]

Entering "segment program" MODE.

```

PROGRAM READ
SEGMENT
YES →[↵]
NO →[CLEAR]

```

Pressing [CLEAR] key to cancel "program read segment" function and return to the previous step.

Pressing [↵] key to enter "program read segment" function.

```

PROGRAM READ
SEGMENT
▶▶▶▶▶▶▶▶

```

Entering the READ mode and the data is reading.


```

PROGRAM READ
  HHP<-PC
  COMPLETED
  RETURN → [↵]

```

Reading PC program, which locates from address 0 to END instruction, to ELC-HHP programming area.

After reading, pressing [↵] key to return to main menu.

READ all PC program (PC→ELC-HHP)

```

MAIN MENU (↑,↓)
■0.PC<->HHP
  1.PGM EDIT
  2.MONITOR/TEST

```

To move cursor to item 0 and press [↵] or [NOP/0]

Entering the program transmission between ELC-HHP and PC.

```

INPUT:[↑,↓]
■1.HHP<-PC
  2.HHP->PC

```

To move cursor to item 1 and press [↵] or [MPS/1]

Entering the READ mode.

```

INPUT:[↑,↓]
  1.SEGMENT PGM
■2.ALL PGM

```

To move cursor to item 2 and press [↵] or [MRD/2]

Entering “read all program” mode.

```

PROGRAM READ
  ALL
  YES →[↵]
  NO →[CLEAR]

```

Pressing [CLEAR] key to cancel “program read all” function and return to the previous step.

Pressing [↵] key to enter “program read all” function.

```

PROGRAM READ
  ALL
  ▶▶▶▶▶▶▶▶

```

Entering the “program read all” mode and the data is reading.

```

PROGRAM READ
  HHP<-PC
  COMPLETED
  RETURN → [↵]

```

Reading whole PC program (ELCSoft) to ELC-HHP programming area.

Pressing [↵] key to return to the main menu.

Write Partial PC Program (ELC-HHP→PC)

```

MAIN MENU (↑,↓)
■0.PC<->HHP
  1.PGM EDIT
  2.MONITOR/TEST

```

To move cursor to item 0 and press [↵] or [NOP/0]

Entering the program transmission between ELC-HHP and PC.

```
INPUT:[ ↑ , ↓ ]
1.HHP<-PC
■2.HHP->PC
```

To move cursor to item 2 and press [↓] or **[MRD/2]**

Entering the WRITE mode.

```
INPUT:[ ↑ , ↓ ]
■1.SEGMENT PGM
2.ALL PGM
```

To move cursor to item 1 and press [↓] or **[MPS/1]**

Entering “write segment program” mode.

```
PROGRAM WRITE
SEGMENT
YES →[↵]
NO →[CLEAR]
```

Pressing **[CLEAR]** key to cancel “program write segment” function and return to the previous step.

Pressing [↓] key to execute “program write segment” function.

```
PROGRAM WRITE
SEGMENT
RETURN→[CLEAR]
▶▶▶▶▶▶▶▶
```

Writing partial ELC-HHP program into PC (ELCSoft).

Entering the “program write segment” mode and the data is writing.

Pressing **[CLEAR]** key to break writing.

```
PROGRAM WRITE
SEGMENT
COMPLETED
RETURN → [↵]
```

Writing ELC-HHP which address is from 0 to END instruction into PC (ELCSoft).

Pressing [↓] key to return to the MAIN MENU, when completed.

Write all PC Program (ELC-HHP→PC)

```
MAIN MENU (↑,↓)
■0.PC<->HHP
1.PGM EDIT
2.MONITOR/TEST
```

To move cursor to item 0 and press [↓] or **[NOP/0]**

Entering the program transmission between ELC-HHP and PC.

```
INPUT:[ ↑ , ↓ ]
1.HHP<-PC
■2.HHP->PC
```

To move cursor to item 2 and press [↓] or **[MRD/2]**

Entering the WRITE mode.

```
INPUT:[ ↑ , ↓ ]
1.SEGMENT PGM
■2.ALL PGM
```

To move cursor to item 2 and press [↓] or **[MRD/2]**

Entering the “write all program” mode.

```
PROGRAM WRITE
  ALL
YES →[↵]
NO  →[CLEAR]
```

Pressing **[CLEAR]** key to cancel “program write all” function and return the previous step.

Pressing **[↵]** key to execute “program write all” function.

```
PROGRAM WRITE
  ALL
EXIT → [CLEAR]
▶▶▶▶▶▶▶▶
```

Writing all ELC-HHP program into PC (ELCSoft).

Entering the WRITE mode and the data is writing.

Pressing **[CLEAR]** key to break the writing.

```
PROGRAM WRITE
  HHP→PC
  COMPLETED
RETURN → [↵]
```

Completing to write whole program.

Pressing **[↵]** key to return to main menu.

10.4.3 M_CARD READ / WRITE and Password Settings

You can execute the program in the M_CARD, READ / WRITE to the ELC-HHP and check M_CARD in any mode with M_CARD. If it needs password for M_CARD and you should input the correct password to enter. Please refer to the following to operate.

Without M_CARD:

```
MAIN MENU (↑,↓)
5.PGM VERIFY
6.PARAMETER SET
■7.HHP<->M_CARD
```

To move the cursor to item 7 and press [↵] or [RET/7] .

Entering to program transmission between ELC-HHP and M_CARD and check M_CARD.

```
*NO MEMORY CARD*

EXIT → [CLEAR]
```

Pressing [CLEAR] key to return the MAIN MENU without M_CARD.

```
MAIN MENU (↑,↓)
■0.ELC<->HHP
1.PGM EDIT
2.MONITOR/TEST
```

Return to the MAIN MENU.

M_CARD is locked by password:

```
MAIN MENU (↑,↓)
5.PGM VERIFY
6.PARAMETER SET
■7.HHP<->M_CARD
```

To move cursor to item 7 and press [↵] or [RET/7] key.

Entering to program transmission between ELC-HHP and M_CARD and check M_CARD.

```
M_CARD IS LOCKED

PASSWORD:
```

Please enter four characters for password if M_CARD is locked as shown in the left. (Password can be digits and capital letters.)

If user forgets the password, it can't enter to operate the function of M_CARD.

```
M_CARD IS LOCKED

PASSWORD:****
```

When forgetting password, there is a set of universal code (four [SPACE]) to disable the password. But after disabled, the program in M_CARD will be clear. Please use it carefully.

User can use the function of “clear program” to clear the program and password in M_CARD. Please refer to Section 10.8 for detail.

```

MEMCARD CLEAR!
YES →[↵]
NO  →[CLEAR]

```

Pressing [↵] key to execute “clear M_CARD”.

Pressing [CLEAR] key to cancel the function “clear M_CARD” and return to main menu.

Times Key	1	2	3	4
NOP 0	0	0	0	0
MPS 1	1	A	B	C
MRD 2	2	D	E	F
MPP 3	3	G	H	I
RST 4	4	J	K	L
CNT 5	5	M	N	O
MCR 6	6	P	Q	R
RET 7	7	S	T	U
SET 8	8	V	W	X
TMR 9	9	Y	Z	9
SPACE	*	*	*	*

You can set password by using the left chart.

The cursor will go to the next if you don't press any key in one second after pressing.

You can press [CLEAR] key to go to the previous character to set the password.

Please refer to the following explanation for M_CARD write and password setting.

```

MLCARD IS LOCKED
*PASSWORD ERROR*
PASSWORD:_

```

If finishing password setting and password is incorrect, the “Password error” as shown in the left to enter the correct password.

Pressing [CLEAR] key to return to main menu.

```

INPUT:[↑, ↓]
■1.HHP<-MLCARD
 2.HHP->MLCARD
 3.MLCARD CHECK

```

After entering the correct password, it will enter the display as shown in the left.

If it doesn't need password for M_CARD, you can enter the display as shown in the left directly.

Read M_CARD (M_CARD→ELC-HHP)

```

INPUT:[↑, ↓]
■1.HHP<-MLCARD
 2.HHP->MLCARD
 3.MLCARD CHECK

```

To move the cursor to item 1 and press [↵] or [MPS/1]

Entering the READ mode.

```
PGM TRANSFER
MLCARD->HHP
YES →[↵]
NO →[CLEAR]
```

Pressing **[CLEAR]** key to cancel “read program” function and return to the previous step.

Pressing **[↵]** key to execute “read program” function.

```
PGM TRANSFER
MLCARD->HHP
```

Entering the READ mode and the data is reading.

```
▶▶▶▶▶▶▶▶
```

```
PGM TRANSFER
MLCARD->HHP
COMPLETED
RETURN → [↵]
```

After finishing to read program, press **[↵]** key to return to main menu.

The password still exists after reading. You need to enter the password in the next time.

Write M_CARD (ELC-HHP→M_CARD) and Password Settings

```
INPUT:[↑, ↓]
1.HHP<-MLCARD
■2.HHP->MLCARD
3.MLCARD CHECK
```

To move the cursor to item 2 and press **[↵]** or **[MRD/2]**

Entering the WRITE mode.

```
PGM TRANSFER
SET PASSWORD ?
YES→[↵] NO→[CLR]
```

System will inquire to set password or not.

Pressing **[↵]** key to execute “set password”.

Pressing **[CLEAR]** key to cancel the function of “set password” and enter the item “program write”. At this time, the password function is disabled.

```
PGM TRANSFER
PASSWORD: _
```

It needs four characters for password. (Password can be digitals and capital letters)

Please refer to the chart above to set the password.

```
PGM TRANSFER
INPUT VERIFY
PASSWORD: _
```

To input password again to confirm. After inputting, it will enter the item of “write program”.

```
PGM TRANSFER
HHP->MLCARD
YES →[↵]
NO →[CLEAR]
```

Pressing **[CLEAR]** key to cancel “write program” function and return to the previous step.

Pressing **[↵]** key to execute “write program” function.

```
PGM TRANSFER  
HWP->MLCARD  
  
▶▶▶▶▶▶▶▶
```

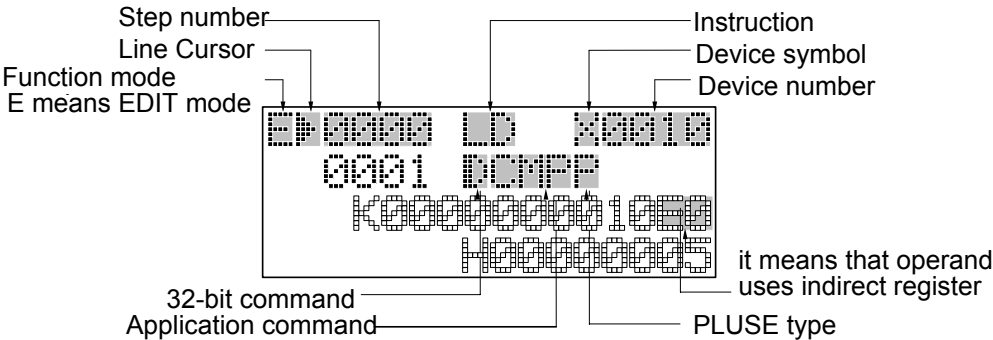
Entering the WRITE mode and the data is writing.

```
PGM TRANSFER  
HWP->MLCARD  
COMPLETED  
RETURN → [↵]
```

Finishing to write program. You can press [↵] to return to main menu.

10.5 Program Mode

10.5.1 Screen Display



10.5.2 Basic Instruction Input

After entering the basic instruction directly, input device name and device number. And press Enter key to enter the item. If you want to interrupt the instruction, please press **[CLEAR]** key.

Example: Basic Instruction LD Y10

```
E▶0000 NOP
0001 NOP
0002 NOP
0003 NOP
```

Entering the EDIT mode.

```
E▶0000 LD
0001 NOP ---
0002 NOP
0003 NOP
```

Pressing **[LD/X]** key.

Pressing **[CLEAR]**key to cancel the function of “input instruction” and return to edit mode.

```
E▶0000 LD Y0000
0001 NOP
0002 NOP
0003 NOP
```

Pressing **[AND/Y]**key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```
E▶0000 LD Y0001
0001 NOP
0002 NOP
0003 NOP
```

Pressing **[MPS/1]** key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```
E▶0000 LD Y0010
0001 NOP
0002 NOP
0003 NOP
```

Pressing **[NOP/0]**key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.


```

E 0000 LD    Y0010
▶0001 NOP
  0002 NOP
  0003 NOP

```

Pressing [↵] key.

After finishing inputting instruction, the cursor will move to next instruction.

10.5.3 Application Instruction Input

In Edit mode, the order for you to input should be **[API/S]**, API number (please refer to Section 10.14 for detail), **[SPACE]**, the first operand and then press **[SPACE]**. In this way, you can keep on entering the following instruction. After input the last instruction, you should press [↵] to finish the input. During the input, you can press **[CLEAR]** to return to the previous status. When using 32-bit instruction, you should press **[ANB/D]** after **[API/S]**. In the same way, if you want to input pulse instruction, you should press **[-/PI]** after **[API/S]**.

Example: (API = 10) CMP K10 H5 Y1

```

E▶0000 NOP
  0001 NOP
  0002 NOP
  0003 NOP

```

Entering the EDIT mode.

Pressing **[API/S]** key.

```

E▶0000 API    000
  0001 NOP
  0002 NOP
  0003 NOP

```

Pressing **[MPS/1]**, **[NOP/0]** key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```

E▶0000 API    010
  0001 NOP
  0002 NOP
  0003 NOP

```

Pressing **[SPACE]** key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```

E 0000 CMP
▶_

```

Pressing **[ORI/K]** key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```

E 0000 CMP
▶ K000000000000

```

Pressing **[MPS/1]**, **[NOP/0]** key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```

E 0000 CMP
▶ K000000000010

```

Pressing **[SPACE]** key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```
E 0000 CMP
      K0000000000010
  ▸ .....
```

Pressing **[ANI/H]** key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```
E 0000 CMP
      K0000000000010
  ▸ H0000000000005
```

Pressing **[CNT/5]** key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```
E 0000 CMP
      K0000000000010
  ▸ H0000000000005
```

Pressing **[SPACE]**, **[AND/Y]**, **[MPS/1]** key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```
E 0000 CMP
      K0000000000010
      H0000000000005
  ▸ Y00001
```

Pressing **[↵]** key.

Pressing **[CLEAR]** key to cancel the instruction input and return to the EDIT mode.

```
E          K000010
          H00005
          Y00001
  ▸00007 NOP
```

After finishing inputting instruction, cursor will move to the next instruction.

10.5.4 Instruction INSERT / DELETE

Insert Instruction

```

E▶0000 LD  X0000
   0001 AND X0001
   0002 OUT Y0000
   0003 LDI X0001

```

Entering the EDIT mode.

Pressing **[INS/DEL]** key.

```

I▶0000 LD  X0000
   0001 AND X0001
   0002 OUT Y0000
   0003 LDI X0001

```

Entering the instruction insert mode.

To move the cursor to the instruction you want to insert and input.

Instruction Delete

```

E▶0000 LD  X0000
   0001 AND X0001
   0002 OUT Y0000
   0003 LDI X0001

```

Entering the EDIT mode.

Pressing **[INS/DEL]** key one time.

```

I▶0000 LD  X0000
   0001 AND X0001
   0002 OUT Y0000
   0003 LDI X0001

```

Entering the instruction insert mode.

Pressing **[INS/DEL]** key again.

```

D▶0000 LD  X0000
   0001 AND X0001
   0002 OUT Y0000
   0003 LDI X0001

```

Entering the instruction delete mode.

To move the cursor to the instruction you want to delete and press **[CLEAR]** key.

10.5.5 [STEP] Function

There are two functions of **[STEP]** key in the EDIT or INS / DEL and MONITOR modes:

1. JUMP
2. Device SEARCH

JUMP: pressing the **[STEP]** key, enter step number and press **[↵]**. The cursor will jump to the step number.

```

E▶0000 LD  X0000
   0001 AND X0001
   0002 OUT Y0000
   0003 LDI X0001

```

Entering the EDIT or INS / DEL or MONITOR modes.

Pressing **[STEP]**key.

```

E▶0000 LD X0000
  0001 AND X0001
  0002 OUT Y0000
STEP : 00000
  
```

Pressing **[MPS/1]**, **[MRD/2]**, **[NOP/0]** keys.

Pressing **[CLEAR]** key to cancel JUMP function and return to the EDIT mode.

```

E▶0000 LD X0000
  0001 AND X0001
  0002 OUT Y0000
STEP : 00120
  
```

Pressing **[↵]** key.

Pressing **[CLEAR]** key to cancel JUMP function and return to the EDIT mode.

```

E▶0120 NOP
  0121 NOP
  0122 NOP
  0123 NOP
  
```

Cursor jumps to STEP 120.

```

E▶0000 LD X0000
  0001 AND X0001
  0002 OUT Y0000
** OVER RANGE **
  
```

If “jump” instruction exceed the range of user program, the display will show the message of “OVER RANGE”.

Device SEARCH: Pressing **[STEP]** and then input the device name and number.

```

E▶0000 LD X0000
  0001 AND X0001
  0002 OUT Y0000
  0003 LDI X0001
  
```

Entering the EDIT or INS / DEL or MONITOR modes.

Pressing **[STEP]** key.

```

E▶0000 LD X0000
  0001 AND X0001
  0002 OUT Y0000
STEP : 00000
  
```

Pressing **[LD/X]** key.

Pressing **[CLEAR]** key to cancel Device JUMP function and return to the EDIT mode.

```

E▶0000 LD X0000
  0001 AND X0001
  0002 OUT Y0000
STEP : X0000
  
```

To input the DEVICE name: X12

Pressing **[MPS/1]**, **[MRD/2]** key.

```

E▶0000 LD X0000
  0001 AND X0001
  0002 OUT Y0000
STEP : X0012
  
```

Pressing **[↵]** key.

Pressing **[CLEAR]** key to cancel Device JUMP function and return to the EDIT mode.

```

E▶0122 LD X0012
  0123 CNT 97
      C0001
      K32767
  
```

Cursor jumps to X12, press **[↵]** key to continue searching for next X12.

```

E▶0000 LD X0000
  0001 AND X0001
  0002 OUT Y0000
*:* NOT FOUND *:*

```

If it can't find the device and it will show the message of "NOT FOUND".

10.5.6 REPLACE Function

This function allows user to replace all the same device number to a new one. But the DEVICE name and indirect register doesn't have any change.

Points to note:

- This function can only be executed in the EDIT mode.
- Indirect register can't be replaced.

DEVICE REPLACE

```

E▶0000 LD X0000
  0001 AND X0010
  0002 OUT Y0000
  0003 LDI X0001

```

Entering the EDIT mode.

Pressing **[SHIFT]** key.

```

E▶0000 LD X0000
  0001 AND X0010
REPLACE: 00000
  TO:

```

Entering the REPLACE function.

```

E▶0000 LD X0000
  0001 AND X0010
REPLACE: X0010
  TO:

```

To input the replaced device number: X10.

Pressing **[LD/X]**, **[MPS/1]** and **[NOP/0]** key and then press **[SHIFT]** key or **[↵]** key.

```

E▶0000 LD X0000
  0001 AND X0010
REPLACE: X0010
  TO: X0000

```

To input the new device number: X11.

Pressing **[MPS/1]** **[MPS/1]** key.

```

E▶0000 LD X0000
  0001 AND X0010
REPLACE: X0010
  TO: X0011

```

Pressing **[↵]** key.

```

E▶0000 LD X0000
  0001 AND X0011
  0002 OUT Y0000
  0003 LDI X0001
    
```

Replacement completed.

```

E▶0000 LD X0000
  0001 AND X0010
REPLACE: X0010
** NOT FOUND **
    
```

If it can't find the device and it will show the message of "NOT FOUND".

10.5.7 Searching Application Instruction

To move the cursor to the specified API code: pressing [API] + [STEP] + API code + [↵]

```

E▶0000 NOP
  0001 NOP
  0002 NOP
  0003 NOP
    
```

Entering the EDIT or INS / DEL or MONITOR modes.

Pressing **[API/S]**, **[STEP]** keys.

```

E▶0000 NOP
  0001 NOP
  0002 NOP
SEARCH API 000
    
```

Pressing **[MPS/1]**, **[MRD/2]** keys.

```

E▶0000 NOP
  0001 NOP
  0002 NOP
SEARCH API 012
    
```

Pressing [↵] key.

```

E▶0000 NOP
  0001 NOP
  0002 NOP
** NOT FOUND **
    
```

It means there is no such code (API 12) in the line, If the display is shown as the left. Please press **[CLEAR]** to return to the previous step.

```

E▶0120 MOV
                H1234
                D0000
  0125 NOP
    
```

If there is API 12 in the line, the cursor will directly jump to the address.

10.5.8 Searching for the API Codes

User can use the following steps of Searching function to look for the API mnemonic codes.

```

E▶0000 NOP
  0001 NOP
  0002 NOP
  0003 NOP
    
```

Entering the EDIT or INS / DEL or MONITOR modes.

Pressing **[API/S]** key.

```

E▶0000 API    000
  0001 NOP
  0002 NOP
  0003 NOP

```

Pressing **[SHIFT]** key to show the searching menu.

```

  INPUT NO.: 000
  0. PROGRAM FLOW
  1. MOVE, COMPARE
  2. MATH. LOGICS

```

Using UP and DOWN key or key in directly to choose group 0 to 24 to look for the appropriate API mnemonic codes.

Example: Looking for the API code number of MOV

```

  INPUT NO.: 001
  0. PROGRAM FLOW
  1. MOVE, COMPARE
  2. MATH. LOGICS

```

Pressing **[MPS/1]**, **[↵]** keys.

To select 1, entering the MOVE/COMPARE instruction group.

```

MOVE, COMPARE:
010 CMP      DP
011 ZCP      DP
012 MOV      DP

```

From the display shown in the left:

The API code number of MOV is 12 and it has 32-bit instruction and PLUSE instruction.

```

MOVE, COMPARE:
013 SMOV     P
014 CML      DP
015 BMOV     DP

```

You can press UP and DOWN to choose other instruction.

Pressing **[CLEAR]** key to return to the previous step.

10.5.9 Indirect Index Register Application

Both indirect Index registers “E” (E0~E7) and “F” (F0~F7) are 16-bit registers. When using 32-bit, you must use indirect Index register “E” (E0~E7).

When using “K”, “H” with “E”, “F”, the order for you to input should be: input “K”, “H” first, press **[SHIFT]** key and then input the indirect Index register E and F. For other operand X, Y, M, S, KnX, KnY, KnM, KnS, T, C and D, you can input the indirect Index register directly without pressing **[SHIFT]** key.

Example: Input the instruction: API 10 CMP K200E2 K2Y2F2 Y1

To input the first operand K200E2 with indirect register

```
E▶0000 NOP
  0001 NOP
  0002 NOP
  0003 NOP
```

Entering Edit mode.

Pressing **[API/S]** key.

```
E▶0000 API 000
  0001 NOP
  0002 NOP
  0003 NOP
```

Pressing **[MPS/1]** and **[NOP/0]** keys.

Pressing **[CLEAR]** key to cancel the function of “input instruction” and return to Edit mode.

```
E▶0000 API 010
  0001 NOP
  0002 NOP
  0003 NOP
```

Pressing **[SPACE]** key.

Pressing **[CLEAR]** key to cancel the function of “input instruction” and return to Edit mode.

```
E 0000 CMP
▶...
```

Pressing **[ORI/K]** key.

Pressing **[CLEAR]** key to cancel the function of “input instruction” and return to Edit mode.

```
E 0000 CMP
▶ K00000000200
```

Pressing **[ORI/K]** **[MRD/2]** **[NOP/0]** **[NOP/0]** keys.

Pressing **[SHIFT]** Key.

```
E 0000 CMP
▶ E2K00000000000
```

Pressing **[ORB/E]** and **[MRD/2]** keys to input the indirect Index register E2.

Pressing **[SPACE]** key to complete input and press **[SPACE]** to continue to input the next operand.

To input the second operand K2Y2F2 with indirect register

```
E 0000 CMP
E2K000000000200
└─ K2Y00002
```

Pressing **[ORI/K]** **[MRD/2]** **[AND/Y]** **[MRD/2]** keys.

```
E 0000 CMP
E2K000000000200
└─ F2 K2Y00002
```

Pressing **[END/F]** **[MRD/2]** key to input the indirect Index register F2.

Pressing **[SPACE]** key to complete input and press **[SPACE]** to continue to input the next operand.

```
E 0000 CMP
E2K000000000200
F2 K2Y00002
└─ Y00001
```

Pressing **[AND/Y]** and **[MPS/1]** keys.

Pressing **[CLEAR]** key to cancel the function of “input instruction” and return to edit mode.

Pressing **[↵]** key to finish inputting instruction.

```
E          K000200E2
          K2Y00002F2
          Y00000
└─ 00007 NOP
```

To move the cursor to the address of next instruction.

10.6 ELC RUN / STOP Mode

After succeeding to connect to ELC, user can enter the RUN/STOP instructions in both MAIN MENU, program monitor and MONITOR/TEST modes to the ELC. Due to the ELC-HHP is displayed according to the users' input, the displays in the following maybe slightly different from yours.

RUN Instruction

```
MAIN MENU  (↑,↓)
0.ELC<->HHP
1.PGM EDIT
2.MONITOR/TEST
```

Entering the MAIN MENU, Program monitor and MONITOR/TEST mode.

Pressing **[RUN/STOP]** key.

```
***  RUN  ***
YES →[↵]
NO  →[CLEAR]
```

The display will be shown as left when ELC is in the STOP status.

Pressing **[↵]** key to confirm the execution.

Pressing **[CLEAR]** key to cancel and return to the previous step.

```
***  RUN  ***
COMPLETED
RETURN → [↵]
```

RUN function completed.

Pressing **[↵]** key to return to the previous step.

STOP Instruction

```
MAIN MENU  (↑,↓)
0.ELC<->HHP
1.PGM EDIT
2.MONITOR/TEST
```

Entering the MAIN MENU or MONITOR/TEST mode.

Pressing **[RUN/STOP]** key.

```
***  STOP  ***
YES →[↵]
NO  →[CLEAR]
```

The display will be shown as left when ELC is in the RUN status.

Pressing **[↵]** key to confirm the execution.

Pressing **[CLEAR]** key to cancel and return to the previous step.

```
***  STOP  ***
COMPLETED
RETURN → [↵]
```

STOP function completed.

Pressing **[↵]** key to return to the previous step.

```
**  ELC ERROR  **
CODE:C401
RETURN → [↵]
```

If ELC ERROR blinks at this moment, press **[RUN/STOP]** key to display the specified ERROR CODE as shown in the left.

Pressing **[↵]** key to return to the previous step.

10.7 MON / TEST Mode

MON/TEST mode applications:

1. MONITOR Mode: DEVICE status, VALUE READING and PROGRAM monitor.
2. TEST Mode: Bit Device force ON/OFF, DEVICE VALUE SETTING, DEVICE monitor and VALUE READING.

MONITOR Mode:

There are two functions of the MONITOR mode, DEVICE monitor and PROGRAM monitor. This function should be used in ELC ON-LINE mode. It's disabled in PC ON-LINE and OFF-LINE modes

DEVICE Monitor Mode:

Entering Monitor mode by selecting the item 2 in the main menu. The devices it can monitor are X, Y, M, S, T, C, D, E and F. X, Y, M, S are bit device and the ON/OFF status in X, Y, M, S can be got by monitor. And you can get the contact and coil status, current timer, counter, and settings (timer or counter values) from device T and C (timer and counter). Contact: when counter or timer sustained, the contact will be ON, otherwise, it will be OFF. Coil: when the device is counting or timing, the coil is ON, otherwise it is OFF.

Example:

```
MAIN MENU  (↑,↓)
■0.ELC<->HHP
1.PGM EDIT
2.MONITOR/TEST
```

Pressing **[MON/TEST]** **[MRD/2]** key or press **[↑]** **[↓]** key to select 2 to enter DEVICE MONITOR.

```
M
```

If there is a M display in the left corner that means it is in the MONITOR mode. Pressing **[SHIFT]** key each time prior to enter a new device name.

```
M X0001
```

Pressing **[SHIFT]** key, then enter **[LD/X]** **[MPS/1]** to monitor X1.

Pressing **[↵]** key to complete.

```
M ■X0001
```

The **■** symbol appears at the left side of X1. This indicates that X1 is ON.

At this moment, press UP/DOWN key to select the monitor device from the menu.

Pressing **[SHIFT]** key and key in the new device name.

Pressing **[CLEAR]** to return to the MAIN MENU.

```
*** ERROR ***
DEVICE ERROR

EXIT → [CLEAR]
```

When the device code (is keyed to be monitored) is exceeded the limit of ELC, ELC will send a wrong message to ELC-HHP to indicate that the code is illegal to ELC.

Example: Searching for timer T0 status.

```
M
```

Entering the MONITOR mode.
Pressing **[SHIFT] [LDI/T]** keys.

```
M T0000
```

Input T0.
Pressing **[↵]** key to complete.

```
M▶ T0000 COIL
K00000
NONE
```

To display the modes of timer T0. The display shown as left K00000 indicates timer T0 value. NONE indicates that the program does not use timer T0 in ELC.

You can then use UP/DOWN or **[SHIFT]** key to select next timer.

```
M▶ T0001 ■COIL
K00010
K00100
```

If display shows as the left, which means the ELC timer K100 is activated. Coil means T1 timer is activated and the value is K10.

Example:

Searching the content of D0 16-bit data register.

```
M D0000 K00000
```

Entering the DEVICE MONITOR mode.
After pressing **[SHIFT]** key, press **[ANB/D]** key.
Pressing **[↵]** key to monitor the content of 16-bit data register.

Searching the content of D0 (D1) 32-bit data register.

```
M D0000 D0001
K0000000000
```

To indicate 32-bit (Double), press **[SHIFT] [ANB/D]** and **[ANB/D]** again in order in the DEVICE MONITOR mode.

Pressing **[↵]** key to monitor the content of 32-bit data register.
Display the value of 32-bit data register D0 and D1, where as D0 is lower 16-bit and D1 is higher 16-bit.

Program Monitor Modes

In Section 10.4 and 10.5 we have discussed the program READ and EDIT functions. Therefore, using ELC-HHP to read the programs from ELC, then enter the program editing modes, press **[MON/TEST]** key, to monitor program status.

```

E▶0000 LD  M1000
   0001 TMR
           T0000
           K00100
  
```

Entering the EDIT mode.

Pressing **[MON/TEST]** key.

```

M▶0000 LD  ■M1000
   0001 TMR
           ■T0000
           K00100
  
```

Entering the program monitor mode to monitor the program status.

Pressing **[↑] [↓]** key to monitor the next program status.

Pressing **[EDIT]** key to return to the EDIT mode.

TEST Mode

This function should be used in ELC ON-LINE mode. It's disabled in PC ON-LINE and OFF-LINE modes.

ON/OFF setting of bit devices (X, Y, M, S) status:

After entering the bit device monitor mode, press MON / TEST key to enter TEST mode. It can force the bit device to be ON or OFF.

Example:

```

MAIN MENU  (↑,↓)
■0.ELC<->HHP
1.PGM EDIT
2.MONITOR/TEST
  
```

Pressing **[MON/TEST]** key to enter DEVICE MNITOR mode and then press **[MON/TEST]**key to enter TEST mode.

```

T
  
```

If the display shows "T" at the left corner, which means it is in the TEST mode. The input method of the device name is the same as the MONITOR mode.

Pressing **[SHIFT]** key each time piror to enter the new device name.

```

T  Y0001
  
```

If you want to monitor Y1, please press **[SHIFT]** key and then enter **[AND/Y][MPS/1]**.

Pressing **[↵]** key to complete.

```

T▶■Y0001
  
```

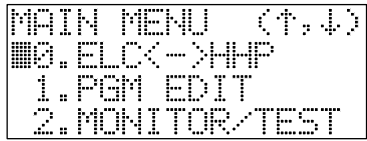
Pressing **[SET/8]** key to force Y1 ON.

Pressing **[RST/4]** key to force Y1 OFF.

Pressing UP/DOWN key to monitor the previous or the next device status.

16-bit word device T, C, D, E, F present value setting

To exchange the decimal and the hexadecimal display of Word device (T, C, D, E, F) present value setting and value monitor, please operate as the following:



```
MAIN MENU (↑,↓)
■0. ELC<->HHP
1. PGM EDIT
2. MONITOR/TEST
```

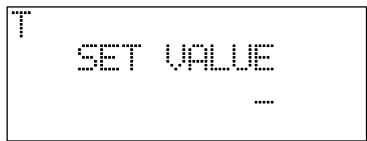
Pressing **[MON/TEST]** key to enter the device monitor mode, and then press **[MON/TEST]** key again to enter the TEST mode.



```
T
```

The input method of device name is the same as the monitor mode.

Pressing **[SPACE]** key to enter the content setting of word device.

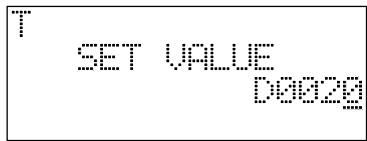


```
T
SET VALUE
---
```

To input the word device name.

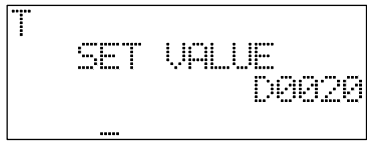
For example: D20.

Pressing **[ANB/D]** **[MRD/2]** **[NOP/0]** keys.



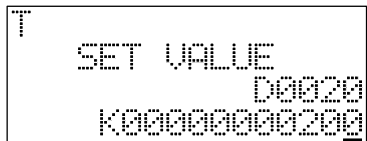
```
T
SET VALUE
D00020
```

Pressing **[↵]** key.



```
T
SET VALUE
D00020
---
```

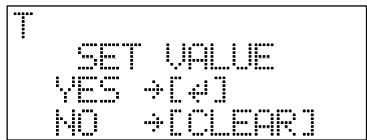
To input the setting value: Knnnnn or Hnnnn.



```
T
SET VALUE
D00020
K000000000200
```

For example: K200. Press **[ORI/K]** **[MRD/2]** **[NOP/0]** **[NOP/0]** keys.

Pressing **[↵]** key.

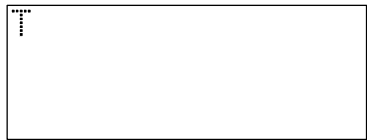


```
T
SET VALUE
YES →[↵]
NO →[CLEAR]
```

To confirm the set value.

Pressing **[↵]** key to complete the input of setting value.

Pressing **[CLEAR]** key to cancel the setting.



```
T
```

Return to the original display.

```
T  ....
```

In this mode, press **[SHIFT]** key to use the device monitor function to monitor the content of D20.

Pressing **[ANB/D]** **[MRD/2]** **[NOP/0]** keys.

```
T  D0020
```

Pressing **[↵]** key to complete the input of the device monitor.

```
T  D0020 K00200
```

Display shows the content of D20 is K200 (decimal).

```
T  D0020 H00C8
```

Pressing **[ANI/H]** key to display in hexadecimal.

To switch to hexadecimal, display as HC8.

Present setting of 32-bit word device (T, C, D, E, F)

```
T  D0020 H00C8
   SET VALUE
   ....
```

Pressing **[SPACE]** key to enter the present value setting of word device.

32-bit data setting: D10 (D11), whereas D10 is lower 16-bit and D11 is upper 16-bit.

```
T  D0020 H00C8
   SET VALUE
       D D0000
```

Pressing **[ANB/D]** key twice, the second time to press **[ANB/D]** key indicates 32-bit, "double" (two 16-bit).

```
T  D0020 H00C8
   SET VALUE
       D D0010
```

Pressing **[MPS/1]**, **[NOP/0]** keys to select D10.

```
T  D0020 H00C8
   SET VALUE
       D D0010
   ....
```

Pressing **[↵]** key.

Waiting for the input of the setting value.

```
T  D0020 H00C8
   SET VALUE
       D D0010
   K000000200000
```

Pressing **[ORI/K]** **[MRD/2]** **[NOP/0]** **[NOP/0]** **[NOP/0]** **[NOP/0]** **[NOP/0]** keys to input K200000.

```
T  D0020 H00C8
   SET VALUE
   YES →[↵]
   NO  →[CLEAR]
```

Pressing [↵] key to confirm the input of the setting value.

Pressing [CLEAR] key to cancel the setting.

```
T  D0020 H00C8
```

Pressing [↵] key.

Return to the original display.

```
T  D0020
   ....
```

In this mode, press [SHIFT] key to use device monitor function to monitor if the content of the D10 (D11) is K200000.

```
T  D0020
   D0000
```

Pressing [ANB/D] key twice, the second time indicates 32-bit (two 16-bit).

```
T  D0020
   D0010
```

To input device numbers and press [MPS/1] [NOP/0] key to select D10.

Pressing [↵] key.

```
T  D0020 H00C8
   D0010   D0011
   H00030D40
```

To display 32-bit data (composed by D10 and D11). D10 is the lower 16-bit that is HD40 and the D11 is the upper 16-bit that is H3. Therefore the value of H30D40 is K200000.

10.8 Clear User's Program Memory

In the three working modes, you can enter it by pressing the item [3] PGM CLEAR in the MAIN MENU.

```
MAIN MENU (↑,↓)
■3.PGM CLEAR
 4.PGM CHECK
 5.PGM VERIFY
```

Using [↑] [↓] key to the item or input the selected item number and then press [↵] key.

```
INPUT:[↑,↓]
■1.ALL CLEAR
 2.SEGMENT CLEAR
 3.M_CARD CLEAR
```

After entering, you can use [↑] [↓] keys to select the item or input the selected item number and then press [↵] key.

Item 1: ALL CLEAR (Clear all the programs)

```
ALL CLEAR!
YES →[↵]
NO →[CLEAR]
```

To confirm that all the programs in ELC-HHP will be cleared.

Pressing [↵] key to start clearing all the programs.

Pressing [CLEAR] key to cancel the instruction and return to the previous display.

Item 2: SEGMENT CLEAR (Clear segment programs)

```
START ADDR:000000
END ADDR:
```

To clear partial programs in ELC-HHP. After entering this mode, user will be requested to enter the range for clearance.

First, enter the start address and then press [↵] key.

Second, enter the end address and press [↵] to complete.

Pressing [CLEAR] key to cancel and return to the previous display.

```
SEGMENT CLEAR!
YES →[↵]
NO →[CLEAR]
```

Pressing [↵] key to start clearing parts of the programs.

Pressing [CLEAR] key to cancel the instruction and return to the MAIN MENU.

Item 3: M_CARD CLEAR (Clear the content of M_CARD)

```
*NO MEMORY CARD*
EXIT → [CLEAR]
```

When you don't have the M_CARD, please press [CLEAR] key to return to the MAIN MENU.

```
MEMLCARD CLEAR!
YES →[↵]
NO →[CLEAR]
```

After pressing [↵] key, it will start to clear all programs in the M_CARD.

Pressing [CLEAR] key to cancel the instruction and return to the MAIN MENU.

```
MEMLCARD CLEAR!
COMPLETED
RETURN → [↵]
```

Clear completed.

Pressing [↵] key to return to the MAIN MENU.

```
MEMLCARD CLEAR!
*** UNCOMPLETED***
EXIT → [CLEAR]
```

If M_CARD can't be clear, display will be shown as left. It means M_CARD is malfunction.

Pressing [CLEAR] key to return to main menu.

Check User's Program

After finishing editing the programs, user can select Item [0] ELC⇔ELC-HHP directly in MAIN MENU, and enter the programs directly into ELC. ELC will check if the grammar is correct. Or select Item [4] PGM CHECK to check the grammar before sending the information to ELC.

```
MAIN MENU (↑,↓)
3.PGM CLEAR
■4.PGM CHECK
5.PGM VERIFY
```

Using [↑][↓] key to the item or input the selected item number and then press [↵] key.

```
PROGRAM CHECK
WORKING!
```

Entering Check mode.

```
PROGRAM CHECK
COMPLETED
RETURN → [↵]
```

Select Item 4 from the MAIN MENU to check the edited programs. If the grammar of the programs is correct, the display will be shown as left.

Press [↵] key to return to the MAIN MENU.

```
***SYNTAX ERROR***
CODE: C401
ADDRESS: 00001
EXIT → [CLEAR]
```

If there is an error detected in the grammar, it will display (ERROR CODE). User can check with Section 10.15 to find out the error codes and the corresponding reasons.

Pressing [↵] key to return to the MAIN MENU.

10.9 ELC and ELC-HHP Program Verifications

There are two selections in this function: 1. To verify both program contents of ELC and ELC-HHP. It can be used in the ELC ONLINE mode. 2. To verify both program contents of M_CARD and ELC-HHP. It can be used in the three modes.

Item 1: To verify both program contents of ELC and ELC-HHP.

```
MAIN MENU  (↑,↓)
 3.PGM CLEAR
 4.PGM CHECK
■5.PGM VERIFY
```

Using [↑][↓] key to the item or input 5 and then press [↵] key.

```
INPUT:[↑,↓]
■1.HHP:ELC
 2.HHP:M_CARD
```

Using [↑][↓] key to select item 1 or input 1 and then press [↵] key to verify with MPU program.

```
PROGRAM VERIFY
HHP:ELC
YES →[↵]
NO →[CLEAR]
```

Pressing [↵] key to verify both program contents of ELC and ELC-HHP.

Pressing [CLEAR] key to cancel and return to the MAIN MENU.

```
PROGRAM VERIFY
HHP:ELC
COMPLETED
RETURN →[↵]
```

Program contents of ELC and ELC-HHP are the same.

Pressing [↵] key to cancel and return to the MAIN MENU.

```
PROGRAM VERIFY
* VERIFY ERROR *
ON LINE 00001
RETURN →[↵]
```

Display will show the difference line number once a mismatch is detected and stop verifying.

Pressing [↵] key to cancel and return to the MAIN MENU.

Item 2: To verify both program contents of M_CARD and ELC-HHP.

```
MAIN MENU  (↑,↓)
 3.PGM CLEAR
 4.PGM CHECK
■5.PGM VERIFY
```

Using [↑][↓] key to the item or input 5 and then press [↵] key.

```
INPUT:[↑,↓]
 1.HHP:ELC
■2.HHP:M_CARD
```

Using [↑][↓] key to select item 2 or input 2 and then press [↵] key to verify with M_CARD program.

```
*NO MEMORY CARD*  
EXIT → [CLEAR]
```

Pressing **[CLEAR]** key to return to main menu when there is no M_CARD.

```
PROGRAM VERIFY  
HHP:M_CARD  
YES → [↵]  
NO → [CLEAR]
```

Pressing **[↵]** key to verify program contents of M_CARD and ELC-HHP.

Pressing **[CLEAR]** key to cancel and return to the MAIN MENU.

```
PROGRAM VERIFY  
HHP:M_CARD  
COMPLETED  
RETURN → [↵]
```

Program contents of M_CARD and ELC-HHP are the same.

Pressing **[↵]** key to cancel and return to the MAIN MENU.

```
PROGRAM VERIFY  
* VERIFY ERROR *  
ON LINE=00001  
RETURN → [↵]
```

Display will show the difference line number once a mismatch is detected and stop verifying.

Pressing **[↵]** key to cancel and return to the MAIN MENU.

10.10 Parameters Settings

There are three items in this function:

1. Setting the program capacity of the ELC-HHP
2. Display the operation version of ELC-HHP
3. Setting the ELC password

Item 1: Setting the program capacity.

After ELC-HHP connects to ELC, executing the function of Program Read. It will display ELC program memory capacity. User can select item 6 “PARAMETER SET” to check ELC memory capacity.

You can check if memory capacity of ELC-HHP is matched with ELC by execute item “Program Write-in” in MAIN MENU. If they are mismatch, the display will show the message as following.

```
PGM CAPACITY
** MISMATCH **
MODIFY HMP? [↵]
EXIT → [CLEAR]
```

If the display is shown as left, it means the memory capacity of ELC-HHP and ELC are mismatch.

Pressing [↵] key to modify the memory capacity of ELC-HHP to be the same as ELC and enter WRITE mode.

Pressing [CLEAR] key to cancel and return to the previous page. Users can also set ELC-HHP program capacity by using item 6 “PARAMETER SET”. Please refer to Section 10.10 for detail.

Please enter this item to reset the memory capacity.

```
MAIN MENU (↑,↓)
■6.PARAMETER SET
7.HMP<->MLCARD
8.FILE REGISTER
```

You can use UP/DOWN key to select 6 in the MAIN MENU or press [MCR/6] key to enter the setting parameter function.

```
INPUT:[ ↑ , ↓ ]
■1.PGM CAPACITY
2.OS VERSION
3.PASSWORD
```

Using UP/DOWN key to select 1 or press [MPS/1] key to modify the memory capacity.

```
* 4000 STEPS *
CHANGE CAPACITY?
YES →[↵]
NO →[CLEAR]
```

The first line displays the capacity of ELC-HHP program area.

Pressing [↵] key to modify the memory capacity.

Pressing [CLEAR] key to cancel and return to the MAIN MENU.

```

INPUT:[ ↑ , ↓ ]
 1.4000 STEPS
■2.8000 STEPS
 3.16000 STEPS
    
```

Please set the memory capacity of ELC-HHP program according to the capacity of ELC program. For example: 8000 STEPS.

Using UP/DOWN key or press **[MRD/2]** key to select item 3.

```

PGM CAPACITY
=8K STEPS
YES →[↵]
NO →[CLEAR]
    
```

Pressing **[↵]** key to confirm the setting and return to the MAIN MENU.

Pressing **[CLEAR]** key to cancel and return to the MAIN MENU.

Item 2: Display the ELC-HHP operation system version.

```

INPUT:[ ↑ , ↓ ]
 1.PGM CAPACITY
■2.OS VERSION
 3.PASSWORD
    
```

Using UP/DOWN or press **[MRD/2]** key to select item 2 to enter the “OS version” function.

```

      ELC-HHP
OPERATION SYSTEM
      V1.0
RETURN → [↵]
    
```

Display the operation system version of ELC-HHP is 1.0.

Pressing **[↵]** key to return to the MAIN MENU.

Item 3: Password setting.

```

INPUT:[ ↑ , ↓ ]
 1.PGM CAPACITY
 2.OS VERSION
■3.PASSWORD
    
```

Using UP/DOWN key or press **[MPP/3]** key to select item 3 to enter the “password” function.

After entering this function, ELC-HHP will query whether you want to set password or not. After diagnosing, the first line will show the following action. The action may be to input password or not.

```

** LOCK ELC **

PASSWORD: _
    
```

If the message displayed as left, it means ELC doesn't have password. Please enter the password to set the password. (Password should be four characters and it can be digitals and capital letters)

Times Key	1	2	3	4
NOP 0	0	0	0	0
MPS 1	1	A	B	C
MRD 2	2	D	E	F
MPP 3	3	G	H	I
RST 4	4	J	K	L
CNT 5	5	M	N	O
MCR 6	6	P	Q	R
RET 7	7	S	T	U
SET 8	8	V	W	X
TMR 9	9	Y	Z	9
SPACE	*	*	*	*

```

*** LOCK ELC ***
INPUT VERIFY
PASSWORD: _

```

```

*** UNLOCK ELC ***
PASSWORD: _

```

Please refer to the left chart for the characters that each numeric button represents

After pressing the button, if you don't press any other key in one second, the cursor will move to the next position for inputting.

During inputting password, you can modify the character by pressing **[CLEAR]** to return to the previous character.

To input password again to verify the password. If it is the same as the previous password, it will lock ELC directly.

If the message shows as left, it means password of ELC is locked. ELC-HHP can't read and modify ELC program.

The locked password will be disabled if you input the correct password.

If the original password is missing, users could key in the general password by pressing **[SPACE]** key for 4 times. The ELC password lockup could thus be removed. However, the internal program memory within ELC will all be clear after executing this action. Please use carefully.

10.11 M_CARD Functions

In three modes of ELC-HHP, the functions of READ/WRITE of M_CARD and ELC-HHP program and M_CARD check can be executed. If it needs password to enter M_CARD, you should input the correct password to execute the functions. When ELC-HHP enter this function, it will check if it has M_CARD. If not, it will display the message of "NO MEMORY CARD".

```
MAIN MENU  (↑,↓)
 6.PARAMETER SET
■7.HHP<->MLCARD
 8.FILE REGISTER
```

Using UP/DOWN key or press **[RET/7]** key to select item 7 in the MAIN MENU to enter the functions of M_CARD and ELC-HHP.
(Please refer to Section 10.4.3 for M_CARD with password.)

```
INPUT:[ ↑ , ↓ ]
■1.HHP<-MLCARD
 2.HHP->MLCARD
 3.MLCARD CHECK
```

There are three items in this function:

- Read M_CARD program, M_CARD→ELC-HHP.
- Write M_CARD program, ELC-HHP→M_CARD.
- Check M_CARD.

Item 1: Read M_CARD program, M_CARD→ELC-HHP. (Please refer to Section 10.4.3)

Item 2: Write M_CARD program, M_CARD→ELC-HHP. (Please refer to Section 10.4.3)

Item 3: Check if there is data in M_CARD.

```
MEMORY CHECK
MEMORY CARD
YES →[↵]
NO →[CLEAR]
```

To display the working situation and ask for the confirmation of the next step.

Pressing **[↵]** key to confirm.

Pressing **[CLEAR]** key to cancel and return to the MAIN MENU.

```
MEMORY CHECK
MEMORY CARD
CHECK SUM:00
RETURN → [↵]
```

To list the check result and display two bits of CHECK SUM.

If there is no program in M_CARD, it will display the message of CHECK SUM : 00.

```
MAIN MENU  (↑,↓)
■0.ELC<->HHP
 1.PGM EDIT
 2.MONITOR/TEST
```

Pressing **[↵]** key to return to the MAIN MENU.

10.12 File Register

There are 1600 file registers (16-bit) in ELC MPU of PC/PA/PH series for user to save files. Users can read/write by using ELC-HHP or PC ELCSOft.

This function can be divided into two parts:

1. ELC-HHP→ELC MPU
2. ELC-HHP→PC

Read/Write of File Register for ELC MPU

Please refer to the following for basic operation in ELC connection mode.

READ for ELC MPU file register (ELC→ELC-HHP)

```
MAIN MENU (↑,↓)
 6.PARAMETER SET
 7.HHP<->MLCARD
■8.FILE REGISTER
```

To move cursor to item 8 and press [↵] or [SET/8] key.

Entering file register to transmit file between ELC-HHP and ELC.

```
INPUT:[ ↑ , ↓ ]
■1.HHP<-ELC
 2.HHP->ELC
```

To move cursor to item 1 and press [↵] or [MPS/1] key.

Entering read mode

```
READING FILE
FROM ELC
▶▶▶▶▶▶
```

It is reading...

```
COMPLETED
RETURN → [↵]
```

After reading, press [↵] key to return to main menu.

Write of file register for ELC MPU (ELC-HHP→ELC)

```
MAIN MENU (↑,↓)
 6.PARAMETER SET
 7.HHP<->MLCARD
■8.FILE REGISTER
```

To move cursor to item 8 and press [↵] or [SET/8] key.

Entering file register to transmit file between ELC-HHP and ELC.

```
INPUT:[ ↑ , ↓ ]
 1.HHP<-ELC
■2.HHP->ELC
```

To move cursor to item 2 and press [↵] or [MRD/2] key.

Enter WRITE mode.

```

  WRITING FILE
    TO ELC
  ▶▶▶▶▶▶▶▶
  
```

It is writing...

```

  COMPLETED

  RETURN → [↵]
  
```

After writing, press [↵] key to return to main menu.

Read/Write of File Register for PC

In PC ELCSOft (for WINDOWS), you should set to connect to ELC-HHP. After connecting, please refer to the Section 10.3.2 for detail. Please refer to the following for basic operation.

READ of file register for PC (ELCSOft) (PC→ELC-HHP)

```

MAIN MENU (↑,↓)
 6.PARAMETER SET
 7.HHP<->MLCARD
■8.FILE REGISTER
  
```

To move cursor to item 8 and press [↵] or [SET/8] key.

Entering file register to transmit file between ELC-HHP and PC (ELCSOft).

```

INPUT:[ ↑ , ↓ ]
■1.HHP<-PC
 2.HHP->PC
  
```

To move the cursor to item 1 and press [↵] or [MPS/1] key.

Entering READ mode.

```

  READING FILE
    FROM PC
  ▶▶▶▶▶▶▶▶
  
```

It is reading.....

```

  COMPLETED

  RETURN → [↵]
  
```

After reading, press [↵] key to return to main menu.

WRITE of file register for PC (ELCSOft) (ELC-HHP→PC)

```

MAIN MENU (↑,↓)
 6.PARAMETER SET
 7.HHP<->MLCARD
■8.FILE REGISTER
  
```

To move cursor to item 8 and press [↵] or [SET/8] key.

Entering file register to transmit file between ELC-HHP and ELC.

```
INPUT:[ ↑ , ↓ ]  
1.HHP<-PC  
■2.HHP->PC
```

To move cursor to item 2 and press [↵] or [MRD/2] key.

Entering WRITE mode.

```
WRITING FILE  
TO PC  
▶▶▶▶▶▶
```

It is writing....

```
COMPLETED  
RETURN ÷ [÷]
```

After writing, press [↵] key to return to main menu.

10.13 Error Code Explanations

After programs are entered to the ELC, ELC ERROR LED is flashing, yet the grammars are correct. The reason may be the bit devices (operand instructions) are misused. User can check the ERROR CODES, Fault Descriptions and Special Data Register Error Codes to find out what or where goes wrong. The error occurred will be stored in the data register D1137. Each bit device usage range can be found in the ELC user manual.

Device Misused

Store Error Number	Associated Meaning	Action
0001	Operand bit device S exceed the usage range	Check the D1137(Error step number)
0002	Pointer/Label P exceed the usage range or duplicated	
0003	Operand KnSm exceed the usage range	
0102	Interrupt pointer I exceed the usage range or duplicated	
0202	Instruction MC device node N exceed the usage range	
0302	Instruction MCR device node N exceed the usage range	
0401	Operand bit device X exceed the usage range	
0403	Operand KnXm exceed the usage range	
0501	Operand bit device Y exceed the usage range	
0503	Operand KnYm exceed the usage range	
0601	Operand bit device T exceed the usage range	
0604	Operand word device T register usage exceed limit	
0801	Operand bit device M exceed the usage range	
0803	Operand KnMm exceed the usage range	
0B01	Operand K/H exceed the usage range	
0D01	DECO misuse operand	
0D02	ENCO Misuse Operand	
0D03	DHSCS Misuse Operand	
0D04	DHSCR Misuse Operand	
0D05	PLSY Misuse Operand	
0D06	PWM Misuse Operand	
0D07	FROM/TO Misuse Operand	Re-enter the instruction correctly
0D08	PID Misuse Operand	
0D09	SPD Misuse Operand	
0D0A	DHSZ Misuse Operand	
0D0B	IST Misuse Operand	
0E01	CNT misuse operand C	

Store Error Number	Associated Meaning	Action
0E04	Operand word device C register usage exceed limit	<p>Check the D1137(Error step number)</p> <p>Re-enter the instruction correctly</p>
0E05	DCNT misuse operand C	
0E18	BCD Conversion Error	
0E19	DIVISION Error (divisor=0)	
0E1A	Floating Point exceeds usage range	
0E1B	It is negative number after radical expression	
0E1C	FROM/TO communication error	
0F04	Operand word device D register usage exceed limit	
0F05	DCNT misuse operand D	
0F06	SFTR misuse operand	
0F07	SFTL misuse operand	
0F08	REF misuse operand	
1000	ZRST misuse operand	
2000	Usage exceed limit (MTR, ARWS, HOUR)	

Loop Check

Store Error Number	Associated Meaning	Action
C400	An unrecognized instruction code is being used	<p>A circuit error occurs if a combination of instructions is incorrect or badly specified.</p> <p>Select programming mode and correct the identified error</p>
C401	Loop Error	
C402	LD / LDI continuously use more than 9 times	
C403	MPS continuously use more than 9 times	
C404	FOR-NEXT exceed 6 levels	
C405	STL / RET used between FOR and NEXT SRET / IRET used between FOR and NEXT MC / MCR used between FOR and NEXT END / FEND used between FOR and NEXT	
C407	STL continuously use more than 9 times	
C408	Use MC / MCR in STL, Use I / P in STL	
C409	Use STL / RET in Subroutine, Use STL / RET in Interrupt Subroutine	
C40A	Use MC / MCR in Subroutine Use MC / MCR in Interrupt Subroutine	
C40B	MC / MCR doesn't begin from N0 or discontinuously	
C40C	MC / MCR corresponding value N is different	

Store Error Number	Associated Meaning	Action
C40D	Use I / P incorrectly	<p>A circuit error occurs if a combination of instructions is incorrect or badly specified.</p> <p>Select programming mode and correct the identified error</p>
C40E	IRET does not follow by the last FEND instruction. SRET does not follow by the last FEND instruction.	
C41C	The number of input/output points of I/O extension unit is larger than the specified limit.	
C41D	Special extension module exceed the usage range	
C41E	Hardware settings of special extension module is error	
C41F	Fail to write into memory	
C420	FROM/TO communication error	
C4EE	The END instruction is not given in the main program	
C4FF	Common invalid (this common doesn't exist)	

10.14 Instruction Table

Basic INSTRUCTION and Ladder Diagram

Standard Instructions

Instruction	Function	Device	STEP
LD	Load a contact	S, X, Y, M, T, C	1
LDI	Load b contact	S, X, Y, M, T, C	1
AND	Series connection-a contact	S, X, Y, M, T, C	1
ANI	Series connection-b contact	S, X, Y, M, T, C	1
OR	Parallel connection-a contact	S, X, Y, M, T, C	1
ORI	Parallel connection-b contact	S, X, Y, M, T, C	1
ANB	Series connection (Multiple circuits)	None	1
ORB	Parallel connection (Multiple circuits)	None	1
MPS	Stores the operation result	None	1
MRD	Reads the operation result	None	1
MPP	Reads, then clears the operation result	None	1

Output Instruction

Instruction	Function	Device	STEP
OUT	Output coil	S, Y, M	1
SET	Latch (ON)	S, Y, M	1
RST	Clear the contact or the register	S, Y, M, T, C, D	3

End Instructions

Instruction	Function	Device	STEP
END	Program END	None	1

Timer and Counter

API No.	Instruction	Function	Device	STEP
96	TMR	16-bit Timer	T-K or T-D	4
97	CNT	16-bit Counter	C-K or C-D (16-bit)	4
97	DCNT	32-bit Counter	C-K or C-D (32-bit)	6

Master Control Instructions

Instruction	Function	Device	STEP
MC	Master control START instruction	N0 - N7	3
MCR	Master control RESET instruction	N0 - N7	3

Contact Rising / Falling edge Instructions

API No.	Instruction	Function	Device	STEP
90	LDP	Rising-edge detection operation	S, X, Y, M, T, C	3
91	LDI	Falling-edge detection operation	S, X, Y, M, T, C	3
92	ANDP	Series connection instruction for the rising-edge detection operation	S, X, Y, M, T, C	3
93	ANDI	Series connection instruction for the falling-edge detection operation	S, X, Y, M, T, C	3
94	ORP	Parallel connection instruction for the rising-edge detection operation	S, X, Y, M, T, C	3
95	ORI	Parallel connection instruction for the falling-edge detection operation	S, X, Y, M, T, C	3

Rising / Falling edge Output Instruction

API No.	Instruction	Function	Device	STEP
89	PLS	Rising-edge output	Y, M	3
99	PLF	Falling-edge output	Y, M	3

Other Instructions

API No.	Instruction	Function	Device	STEP
-	NOP	No operation action	None	1
98	INV	Inverting operation	None	1
-	P	Pointers	P0~P255	1
-	I	Interrupt pointers	I□□□	1

Ladder Step Instructions

Instruction	Function	Device	STEP
STL	Step transition ladder start instruction	S	1
RET	Step transition ladder return instruction	None	1

Application Instructions

API	Mnemonic		P	Functions	Operand	Steps	
	16-bit	32-bit				16bits	32bits
00	CJ	-	✓	Conditional jump	S	3	-
01	CALL	-	✓	Call subroutine	S	3	-
02	SRET	-	-	Subroutine return	None	1	-
03	IRET	-	-	Interrupt return	None	1	-
04	EI	-	-	Enable interrupts	None	1	-
05	DI	-	-	Disable interrupts	None	1	-
06	FEND	-	-	First end	None	1	-
07	WDT	-	✓	Watchdog timer refresh	None	1	-
08	FOR	-	-	Start of FOR-NEXT loop	S	3	-
09	NEXT	-	-	End of FOR-NEX loop	None	1	-
10	CMP	DCMP	✓	Compare	S1, S2, D	7	13
11	ZCP	DZCP	✓	Zone compare	S1, S2, S, D	9	17
12	MOV	DMOV	✓	Data Move	S, D	5	9
13	SMOV	-	✓	Shift Move	S, m1, m2, D, n	11	-
14	CML	DCML	✓	Compliment	S, D	5	9
15	BMOV	-	✓	Block move	S, D, n	7	-
16	FMOV	DFMOV	✓	Fill move	S, D, n	7	13
17	XCH	DXCH	✓	Data exchange	D1, D2	5	9
18	BCD	DBCD	✓	Covert BIN into BCD	S, D	5	9
19	BIN	DBIN	✓	Covert BCD into BIN	S, D	5	9
20	ADD	DADD	✓	Addition	S1, S2, D	7	13
21	SUB	DSUB	✓	Subtraction	S1, S2, D	7	13
22	MUL	DMUL	✓	Multiplication	S1, S2, D	7	13
23	DIV	DDIV	✓	Division	S1, S2, D	7	13
24	INC	DINC	✓	Increment	D	3	5
25	DEC	DDEC	✓	Decrement	D	3	5
26	WAND	DAND	✓	Logical AND	S1, S2, D	7	13
27	WOR	DOR	✓	Logical OR	S1, S2, D	7	13
28	WXOR	DXOR	✓	Logical XOR	S1, S2, D	7	13
29	NEG	DNEG	✓	Negation	D	3	5
30	ROR	DROR	✓	Rotate to the right	D, n	5	9
31	ROL	DROL	✓	Rotate to the left	D, n	5	9
32	RCR	DRCR	✓	Rotate right with carry	D, n	5	9
33	RCL	DRCL	✓	Rotate left with carry	D, n	5	9
34	SFTR	-	✓	Bit shift right	S, D, n1, n2	9	-
35	SFTL	-	✓	Bit shift left	S, D, n1, n2	9	-
36	WSFR	-	✓	Word shift right	S, D, n1, n2	9	-
37	WSFL	-	✓	Word shift left	S, D, n1, n2	9	-
38	SFWR	-	✓	Shift register write	S, D, n	7	-
39	SFRD	-	✓	Shift register read	S, D, n	7	-
40	ZRST	-	✓	Zone reset	D1, D2	5	-
41	DECO	-	✓	Decode	S, D, n	7	-
42	ENCO	-	✓	Encode	S, D, n	7	-
43	SUM	DSUM	✓	Sum of active bits	S, D	5	9
44	BON	DBON	✓	Check specified bit status	S, D, n	7	13

API	Mnemonic		P	Functions	Operand	Steps	
	16-bit	32-bit				16bits	32bits
45	MEAN	DMEAN	✓	Mean	S, D, n	7	13
46	ANS	–	–	Alarm device output	S, m, D	7	–
47	ANR	–	✓	Alarm device reset	NONE	1	–
48	SQR	DSQR	✓	Square root	S, D	5	9
49	FLT	DFLT	✓	Floating point	S, D	5	9
50	REF	–	✓	Refresh	D, n	5	–
51	REFF	–	✓	Refresh and filter adjust	N	3	–
52	MTR	–	–	Input Matrix	S,D1,D2, n	9	–
53	HSCS	DHSCS	–	High speed counter set	S1, S2, D	7	13
54	HSCR	DHSCR	–	High speed counter reset	S1, S2, D	7	13
55	HSZ	DHSZ	–	HSC zone compare	S1, S2, S, D	9	17
56	SPD	–	–	Speed detect	S1, S2, D	7	–
57	PLSY	DPLSY	–	Pulse Y output	S1, S2, D	7	13
58	PWM	–	–	Pulse width modulation	S1, S2, D	7	–
59	PLSR	DPLSR	–	Ramp Pulse output	S1, S2, S3, D	9	17
60	IST	–	–	Manual/Auto control	S, D1, D2	–	–
61	SER	DSER	✓	Search a data stack	S1, S2, D, n	9	17
62	ABSD	DABSD	–	Absolute drum sequencer	S1, S2, D, n	9	17
63	INCD	–	–	Increment drum sequencer	S1, S2, D, n	9	–
64	TTMR	–	–	Teaching timer	D, n	5	–
65	STMR	–	–	Special timer	S, m, D	7	–
66	ALT	–	–	Alternate state	D	3	–
67	RAMP	–	–	Ramp variable value	S1, S2, D, n	9	–
69	SORT	–	–	Data sort	S, m1, m2, D, n	11	–
70	TKY	DTKY	–	10-key keypad input	S, D1, D2	7	13
71	HKY	DHKY	–	16-key keypad input	S, D1, D2, D3	9	17
72	DSW	–	–	Digital switch input	S, D1, D2, n	9	–
73	SEGD	–	✓	Seven segment decoder	S, D	5	–
74	SEGL	–	–	Seven segment with latch	S, D, n	7	–
75	ARWS	–	–	Arrow switch	S, D1, D2, n	9	–
76	ASC	–	–	ASCII code	S, D	11	–
77	PR	–	–	Print	S, D	5	–
78	FROM	DFROM	✓	Read from a special module CR data	M1, m2, D, n	9	17
79	TO	DTO	✓	Write to a special module CR data	M1, m2, S, n	9	17
80	RS	–	–	Serial data communication	S, m, D, n	9	–
81	PRUN	DPRUN	✓	Octal number system transmission	S, D	5	9
82	ASCI	–	✓	Covert HEX into ASCII	S, D, n	7	–
83	HEX	–	✓	Covert ASCII into HEX	S, D, n	7	–
84	CCD	–	✓	Check code	S, D, n	7	–
85	VRRD	–	✓	Volume read	S, D	5	–
86	VRSC	–	✓	Volume scale read	S, D	5	–
87	ABS	DABS	✓	Absolute value	D	3	5

API	Mnemonic		P	Functions	Operand	Steps	
	16-bit	32-bit				16bits	32bits
88	PID	DPID	-	PID calculation	S1, S2, S3, D	9	17
89	PLS	-	-	Rising-edge output	Y, M	3	-
90	LDP	-	-	Rising edge pulse	S, X, Y, M, T, C	3	-
91	LDF	-	-	Falling edge pulse	S, X, Y, M, T, C	3	-
92	ANDP	-	-	Serial connection of rising edge pulse	S, X, Y, M, T, C	3	-
93	ANDF	-	-	Serial connection of falling edge pulse	S, X, Y, M, T, C	3	-
94	ORP	-	-	Parallel connection of rising edge pulse	S, X, Y, M, T, C	3	-
95	ORF	-	-	Parallel connection of falling edge pulse	S, X, Y, M, T, C	3	-
96	TMR	-	-	Timer	T	4	-
97	CNT	DCNT	-	Counter	C	4	6
98	INV	-	-	Inverting operation	S, X, Y, M, T, C	1	-
99	PLF	-	-	Falling edge output	Y, M	3	-
100	MODRD	-	-	Modbus data read	S1, S2, n	7	-
101	MODWR	-	-	Modbus data write	S1, S2, n	7	-
107	LRC	-	✓	LRC error check	S, n, D	7	-
108	CRC	-	✓	CRC error check	S, n, D	7	-
109	SWRD	-	✓	Digital switch read	D	3	-
110	ECMP	DECMP	✓	Floating point compare	S1, S2, D	7	13
111	EZCP	DEZCP	✓	Floating point zone compare	S1, S2, S, D	9	17
118	EBCD	DEBCD	✓	Float to scientific conversion	S, D	5	9
119	EBIN	DEBIN	✓	Scientific to float conversion	S, D	5	9
120	EADD	DEADD	✓	Floating point addition	S1, S2, D	7	13
121	ESUB	DESUB	✓	Floating point subtraction	S1, S2, D	7	13
122	EMUL	DEMUL	✓	Floating point multiplication	S1, S2, D	7	13
123	EDIV	DEDIV	✓	Floating point division	S1, S2, D	7	13
127	ESQR	DESQR	✓	Floating point square root	S, D	5	9
129	INT	DINT	✓	Float to integer	S, D	5	9
130	SIN	DSIN	✓	Sine	S, D	5	9
131	COS	DCOS	✓	Cosine	S, D	5	9
132	TAN	DTAN	✓	Tangent	S, D	5	9
147	SWAP	DSWAP	✓	Swap high/low byte	S	3	5
148	MEMR	DMEMR	✓	Data backup Memory read	m, D, n	7	13
149	MEMW	DMEMW	✓	Data backup Memory write	S, m, n	7	13
150	MODRW	-	-	Modbus data read/write	S1, S2, S3, S4, n	11	-
155	ABSR	DABSR	-	ABS current value read	S1, D1, D2	7	13
156	ZRN	DZRN	-	Zero return	S1, S2, S, D	9	17
157	PLSV	DPLSV	-	Variable speed pulse output	S1, S2, D	7	13
158	DRVI	DDRVI	-	Drive to increment	S1, S2, S, D	9	17
159	DRVA	DDRVA	-	Drive to absolute	S1, S2, S, D	9	17
160	TCMP	-	✓	Time compare	S1, S2, S3, S, D	11	-
161	TZCP	-	✓	Time zone compare	S1, S2, S, D	9	-
162	TADD	-	✓	Time addition	S1, S2, D	7	-
163	TSUB	-	✓	Time subtraction	S1, S2, D	7	-

API	Mnemonic		P	Functions	Operand	Steps	
	16-bit	32-bit				16bits	32bits
166	TRD	-	✓	Time data read	D	3	-
167	TWR	-	✓	Time data write	S	3	-
169	HOUR	DHOUR		Hour meter	S1, S2, D	7	13
170	GRY	DGRY	✓	Convert BIN into Gray code	S, D	5	9
171	GBIN	DGBIN	✓	Convert Gray code into BIN	S, D	5	9
224	LD=	DLD=	-	$S1 = S2$	S1, S2	5	9
225	LD>	DLD>	-	$S1 > S2$	S1, S2	5	9
226	LD<	DLD<	-	$S1 < S2$	S1, S2	5	9
228	LD<>	DLD<>	-	$S1 \neq S2$	S1, S2	5	9
229	LD<=	DLD<=	-	$S1 \leq S2$	S1, S2	5	9
230	LD>=	DLD>=	-	$S1 \geq S2$	S1, S2	5	9
232	AND=	DAND=	-	$S1 = S2$	S1, S2	5	9
233	AND>	DAND>	-	$S1 > S2$	S1, S2	5	9
234	AND<	DAND<	-	$S1 < S2$	S1, S2	5	9
236	AND<>	DAND<>	-	$S1 \neq S2$	S1, S2	5	9
237	AND<=	DAND<=	-	$S1 \leq S2$	S1, S2	5	9
238	AND>=	DAND>=	-	$S1 \geq S2$	S1, S2	5	9
240	OR=	DOR=	-	$S1 = S2$	S1, S2	5	9
241	OR>	DOR>	-	$S1 > S2$	S1, S2	5	9
242	OR<	DOR<	-	$S1 < S2$	S1, S2	5	9
244	OR<>	DOR<>	-	$S1 \neq S2$	S1, S2	5	9
245	OR<=	DOR<=	-	$S1 \leq S2$	S1, S2	5	9
246	OR>=	DOR>=	-	$S1 \geq S2$	S1, S2	5	9

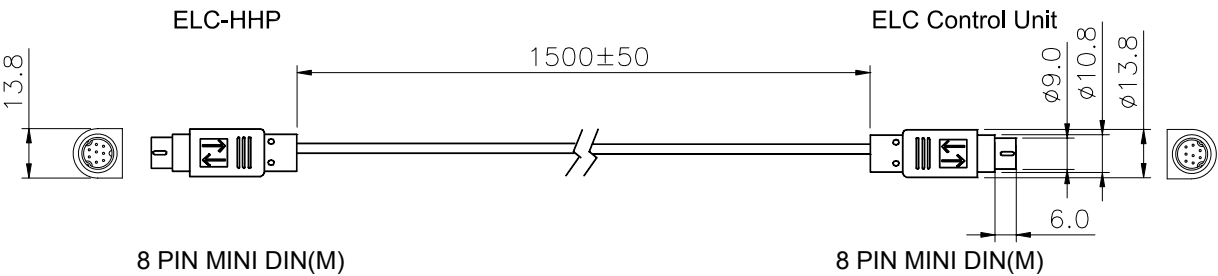
10.15 Troubleshooting and Error Message

The ELC-HHP has a comprehensive fault diagnostic system that includes several different alarms and error messages. Once a fault is detected, the corresponding protective functions will be activated to protect the ELC-HHP. Below are the fault descriptions, for a fault shown on the ELC-HHP keypad display.

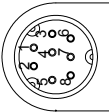
Error Message	Fault Descriptions	Corrective Actions
CPU error	CPU has been damaged.	Change CPU
RAM error	The READ/WRITE instructions of CPU external RAM is abnormal	Change CPU
SRAM error	The READ/WRITE instructions of CPU external RAM is abnormal	Change external RAM
Over range	I / P content exceeds the maximum range	Revise I / P according to ELC user manual.
COMM. ERROR	Serial communication port is abnormal	Check ELC-HHP connection cable
EXCEPTION CODE: 0002	Receive the exception code from ELC	Make sure the monitor devices are within the range
PGM DISMATCH	ELC-HHP and ELC program can not match	Check the user programs according to the return address
NOT FOUND	Instruction can not be found	Process next step
MISS END	There is no “end” instruction at the end of program	Add the “END” instruction at the end of the program
INSTRUCTION ERROR	Can not recognize the program codes	Enter the correct instructions
ONLINE FAIL	Connecting ELC and PC	Retry or operate the ELC under OFFLINE mode
CANNOT WRITE-IN WHEN ELC IS RUNNING	Can not write in during the ELC RUN mode	Set ELC in the “STOP” mode
VERIFY ERROR	Verify errors	Rephrase inconsistent address
UNEQUAL	Unequal program space.	Revise the program space
CODE ERROR	Unrecognizable instruction during the grammar checking	Recheck user’s grammar
ELC ERROR	ELC error from the content of return message D1004	Refer to the ERROR CODE chart
ERROR CODE	Grammar error	Refer to the ERROR CODE chart

10.16 ELC-HHP Connection

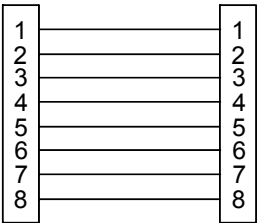
ELC-CBHHELC15:



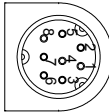
ELC-HHP SIDE



PIN 1,2: 5V
PIN 3,6,8: GND
PIN 4: TX
PIN 5: RX

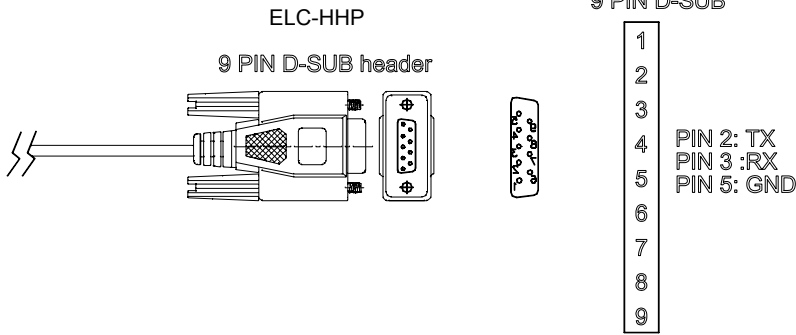


ELC SIDE

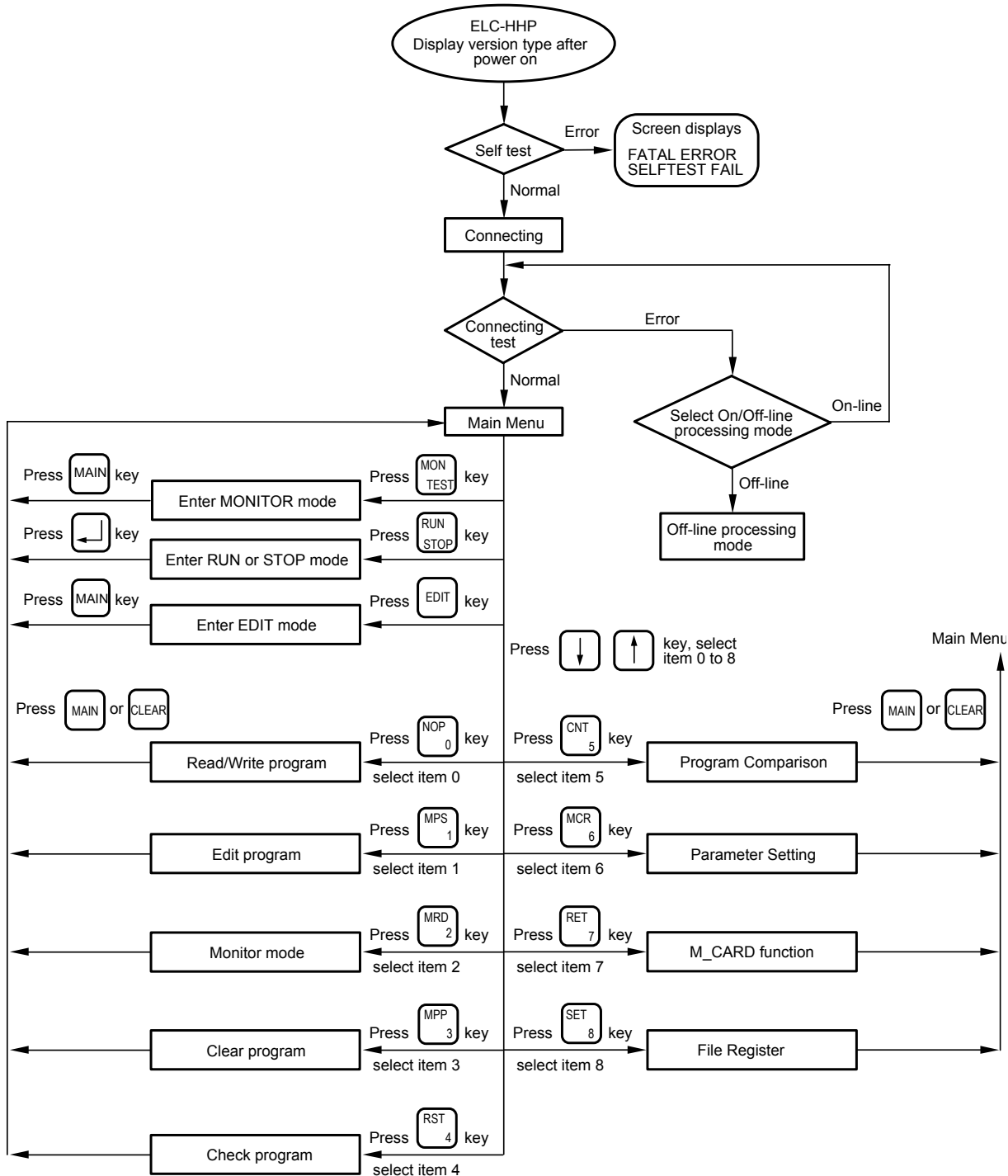


PIN 1,2: 5V
PIN 3,6,8: GND
PIN 4: RX
PIN 5: TX

9 PIN D-SUB:



10.17 ELC-HHP Operation Flow Chart



MEMO

Company Information

Eaton's electrical business is a global leader in electrical control, power distribution, and industrial automation products and services. Through advanced product development, world-class manufacturing methods, and global engineering services and support, Eaton's electrical business provides customer-driven solutions under brand names such as Cutler-Hammer®, Powerware®, Durant®, Heinemann®, Holec® and MEM®, which globally serve the changing needs of the industrial, utility, light commercial, residential, and OEM markets. For more information, visit **www.EatonElectrical.com**.

Eaton Corporation is a diversified industrial manufacturer with 2003 sales of \$8.1 billion. Eaton is a global leader in fluid power systems and services for industrial, mobile and aircraft equipment; electrical systems and components for power quality, distribution and control; automotive engine air management systems and powertrain controls for fuel economy; and intelligent drivetrain systems for fuel economy and safety in trucks. Eaton has 55,000 employees and sells products to customers in more than 100 countries. For more information, visit **www.eaton.com**.

Eaton Electrical
1000 Cherrington Parkway
Moon Township, PA 15108-4312
USA
Tel: 1-800-525-2000
www.EatonElectrical.com

EAT•N | **Cutler-Hammer**

© 2004 Eaton Corporation
All Rights Reserved
Printed in USA
Publication No. MN05003003E
August 2005