IABU Headquarters
Delta Electronics, Inc.
Taoyuan3
No.18, Xinglong Rd., Taoyuan City,
Taoyuan County 330, Taiwan, R.O.C.
TEL: 886-3-362-6301 / FAX: 886-3-371-6301

Asia
Delta Electronics (Jiangsu) Ltd.
Wujiang Plant3
1688 Jiangxing East Road,
Wujiang Economic Development Zone
Wujiang City, Jiang Su Province,
People's Republic of China (Post code: 215200)
TEL: 86-512-6340-3008 / FAX: 86-769-6340-7290

Delta Greentech (China) Co., Ltd.
238 Min-Xia Road, Cao-Lu Industry Zone,Pudong, Shanghai,
People's Republic of China
Post code : 201209
TEL: 021-58635678 / FAX: 021-58630003

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

Delta Electronics (Korea), Inc.
234-9, Duck Soo Building 7F, Nonhyun-Dong,
Kangnam-Gu, Seoul, Korea 135-010
TEL: 82-2-515-5305 / FAX: 82-2-515-5302

Delta Electronics Int'l (S) Pte Ltd
4 Kaki Bukit Ave 1, #05-05, Singapore 417939
TEL: 65-6747-5155 / FAX: 65-6744-9228

Delta Electronics (India) Pvt. Ltd.
Plot No. 43, Sector – 35, HSIIDC,
Gurgaon122001, Haryana, India
TEL: 1-919-767-3800 / FAX: 91-124-403-6045

Americas
Delta Products Corporation (USA)
Raleigh Office
P.O. Box 12173,5101 Davis Drive,
Research Triangle Park, NC 27709, U.S.A.
TEL: 1-919-767-3813 / FAX: 1-919-767-3969

Delta Greentech (Brasil) S.A
Sao Paulo Office
Rua Itapeva, 26-30 Andar Edificio Itapeva One-Bela Vista
01332-000-Sao Paulo-SP-Brazil
TEL: +55 11 3568-3850/FAX: +55 11 3568-3865

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

DVP-0139720-03

20110920

DVP-ES2/EX2/SS2/SA2/SX2 Operation Manual (Programming)

DVP-ES2/EX2/SS2/ SA2/SX2

Operation Manual (Programming)

www.delta.com.tw/industrialautomation

DVP-ES2/EX2/SS2/SA2/SX2

Operation Manual

Programming

Publication History

Issue	Description of Changes	Date
First Edition	Issued the first edition,	2010/08/04
Second Edition	1. Chapter 2.8 M Relay: Add M1037, M1119, M1182, M1308, M1346, and M1356, and update the description of the functions of M1055~M1057and M1183. 2. Chapter 2.13 Special Data Register: Add D1037, D1312, D1354, and D1900~D1931, and modify the attributes of the latched functions of D1062, D1114, D1115, and D1118. 3. Chapter 2.16 Applications of Special M Relays and D Registers: Update the description of the functions of RTCs; add M1037, D1037(Enable SPD function）, M1119（Enable 2-speed output function of DDRVI instruction）, M1308, D1312（Output specified pulses or seek Z phase signal when zero point is achieved）, and M1346（Output clear signals when ZRN is completed）; Easy PLC Link is changed to PLC Link, and the description is added. 4. Chapter 3.1 Basic Instructions (without API numbers) and Chapter 3.2 Explanations to Basic Instructions: Add NP and PN instructions, and add Chapter 3.7 Numerical List of Instructions (in alphabetic order) 5. Chapter 3.6 Numerical List of Instructions and Chapter 3.8 Detailed Instruction Explanation: Increase explanations of DSPA instruction, and add floating-point contact type comparison instructions FLD=, FLD>, FLD<, FLD<>, FLD<=, FLD>=, FAND=, FAND>, FAND<, FAND<>, FAND<=, FAND>=, FOR=, FOR>, FOR<, FOR<>, FOR<=, FOR>=; add the supplementary description of PLSR instruction and the description of K11~K19 in DTM instruction mode; update the description of API166 instruction.	2011/09/15

PLC Concepts

1

This chapter introduces basic and advanced concepts of ladder logic, which is the mostly adopted programming language of PLC. Users familiar with the PLC concepts can move to the next chapter for further programming concepts. However, for users not familiar with the operating principles of PLC, please refer to this chapter to get a full understanding of PLC concepts.

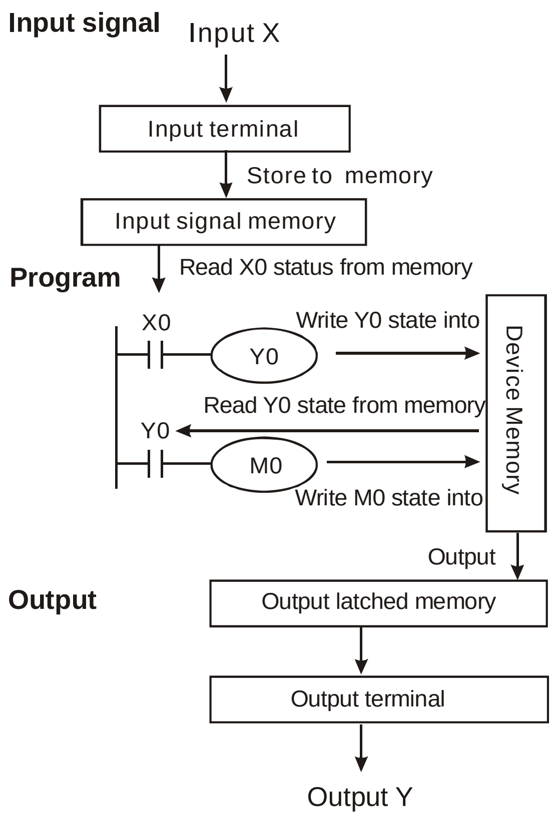
Chapter Contents

1.1 PLC Scan Method

PLC utilizes a standard scan method when evaluating user program.

Scanning process:

Scan input status	Read the physical input status and store the data in internal memory.
Evaluate user program	Evaluate the user program with data stored in internal memory. Program scanning starts from up to down and left to right until reaching the end of the program.
Refresh the outputs	Write the evaluated data to the physical outputs



Input signal:

PLC reads the ON/OFF status of each input and stores the status into memory before evaluating the user program.

Once the external input status is stored into internal memory, any change at the external inputs will not be updated until next scan cycle starts.

Program:

PLC executes instructions in user program from top to down and left to right then stores the evaluated data into internal memory. Some of this memory is latched.

Output:

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

Scan time

The duration of the full scan cycle (read, evaluate, write) is called “scan time.” With more I/O or longer program, scan time becomes longer.

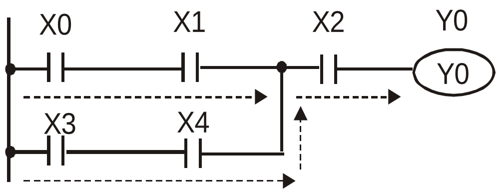
Read scan time	PLC measures its own scan time and stores the value (0.1ms) in register D1010, minimum scan time in register D1011, and maximum scan time in register D1012.
Measure scan time	Scan time can also be measured by toggling an output every scan and then measuring the pulse width on the output being toggled.
Calculate scan time	Scan time can be calculated by adding the known time required for each instruction in the user program. For scan time information of individual instruction please refer to Ch3 in this manual.

Scan time exception

PLC can process certain items faster than the scan time. Some of these items interrupts and halt the scan time to process the interrupt subroutine program. A direct I/O refresh instruction REF allows the PLC to access I/O immediately during user program evaluation instead of waiting until the next scan cycle.

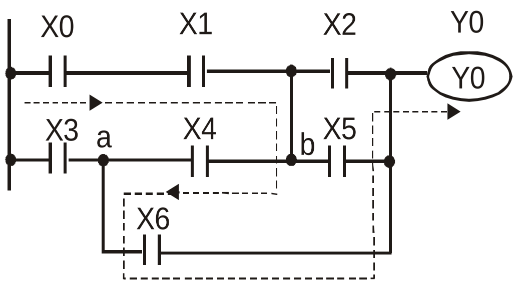
1.2 Current Flow

Ladder logic follows a left to right principle. In the example below, the current flows through paths started from either X0 or X3.

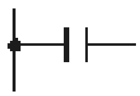
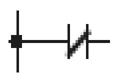


Reverse Current

When a current flows from right to left, which makes a reverse current logic, an error will be detected when compiling the program. The example below shows the reverse current flow.



1.3 NO Contact, NC Contact

NO contact	 Normally Open Contact, A contact
NC Contact	 Normally Closed Contact, B contact

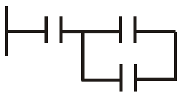
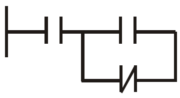
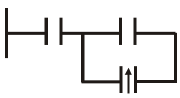
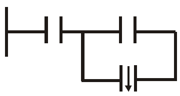
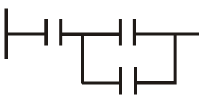
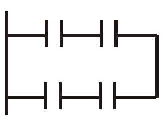
1.4 PLC Registers and Relays

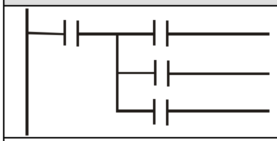
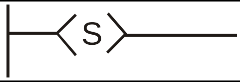
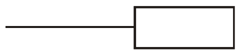
Introduction to the basic internal devices in a PLC

X (Input Relay)	Bit memory represents the physical input points and receives external input signals. ■ Device indication: Indicated as **X** and numbered in octal, e.g. X0~X7, X10~X17…X377
Y (Output Relay)	Bit memory represents the physical output points and saves the status to be refreshed to physical output devices. ■ Device indication: Indicated as **Y** and numbered in octal, e.g. Y0~Y7, Y10~Y17. ..Y377
M (Internal Relay)	Bit memory indicates PLC status. ■ Device indication: Indicated as **M** and numbered in decimal, e.g. M0, M1, M2…M4095
S (Step Relay)	Bit memory indicates PLC status in Step Function Control (SFC) mode. If no STL instruction is applied in program, step point S can be used as an internal relay M as well as an annunciator. ■ Device indication: Indicated as **S** and numbered in decimal, e.g. S0, S1, S2…S1023
T (Relay) (Word) (Dword)	Bit, word or double word memory used for timing and has coil, contact and register in it. When its coil is ON and the set time is reached, the associated contact will be energized. Every timer has its resolution (unit: 1ms/10ms/100ms). ■ Device indication: Indicated as **T** and numbered in decimal, e.g. T0, T1, T2…T255
C (Counter) (Relay) (Word) (Dword)	Bit, word or double word memory used for counting and has coil, contact and register in it. The counter count once (1 pulse) when the coil goes from OFF to ON. When the predefined counter value is reached, the associated contact will be energized. There are 16-bit and 32-bit high-speed counters available for users. ■ Device indication: Indicated as **C** and numbered in decimal, e.g. C0, C1, C2…C255
D (Data register) (Word)	Word memory stores values and parameters for data operations. Every register is able to store a word (16-bit binary value). A double word will occupy 2 consecutive data registers. ■ Device indication: Indicated as **D** and numbered in decimal, e.g. D0, D1, D2…D4999
E, F (Index register) (Word)	Word memory used as a modifier to indicate a specified device (word and double word) by defining an offset. Index registers not used as a modifier can be used as general purpose register. ■ Device indication: indicated as E0 ~ E7 and F0 ~ F7.

1.5 Ladder Logic Symbols

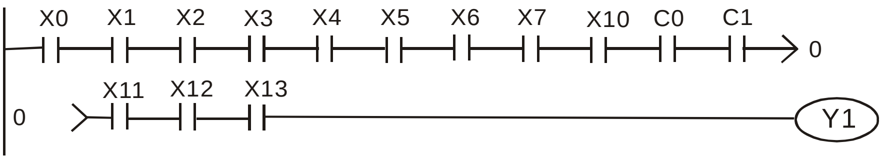
The following table displays list of WPLSoft symbols their description, command, and memory registers that are able to use the symbol.

Ladder Diagram Structure	Explanation	Instruction	Available Devices
	NO (Normally Open) contact / A contact	LD	X, Y, M, S, T, C
	NC (Normally Closed) contact / B contact	LDI	X, Y, M, S, T, C
	NO contact in series	AND	X, Y, M, S, T, C
	NC contact in series	ANI	X, Y, M, S, T, C
	NO contact in parallel	OR	X, Y, M, S, T, C
	NC contact in parallel	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
	Rising-edge trigger in series	ANDP	X, Y, M, S, T, C
	Falling-edge trigger in series	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 series	ANB	None
	Block in parallel	ORB	None

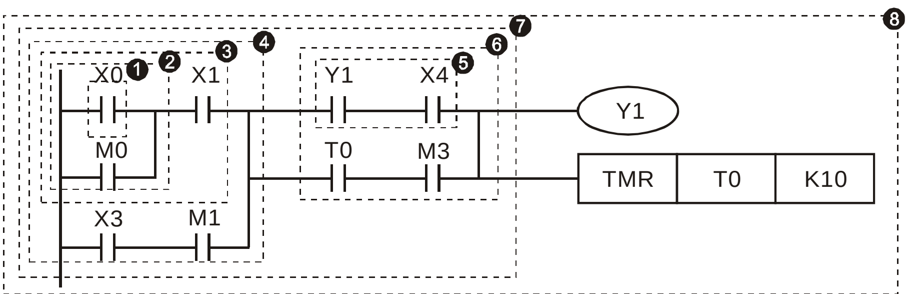
Ladder Diagram Structure	Explanation	Instruction	Available Devices
	Multiple output branches	MPS MRD MPP	None
	Output coil	OUT	Y, M, S
	Step ladder	STL	S
	Basic / Application instruction	-	Basic instructions and API instructions. Please refer to chapter 3 Instruction Set
	Inverse logic	INV	None

1.5.1 Creating a PLC Ladder Program

The editing of the program should start from the left side bus line to the right side bus line, and from up to down. However, the right side bus line is omitted when editing in WPLSoft. A single row can have maximum 11 contacts on it. If more than 11 contacts are connected, a continuous symbol "0" will be generated automatically and the 12th contact will be placed at the start of next row. The same input points can be used repeatedly. See the figure below:



When evaluating the user program, PLC scan starts from left to right and proceeds to next row down until the PLC reaches END instruction. Output coils and basic / application instructions belong to the output process and are placed at the right of ladder diagram. The sample program below explains the execution order of a ladder diagram. The numbers in the black circles indicate the execution order.

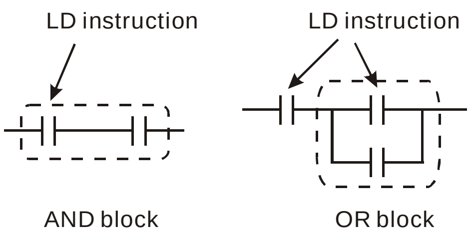


Execution order of the sample program:

1	LD	X0
2	OR	M0
3	AND	X1
4	LD	X3
	AND	M1
	ORB	
5	LD	Y1
	AND	X4
6	LD	T0
	AND	M3
	ORB	
7	ANB	
8	OUT	Y1
	TMR	T0 K10

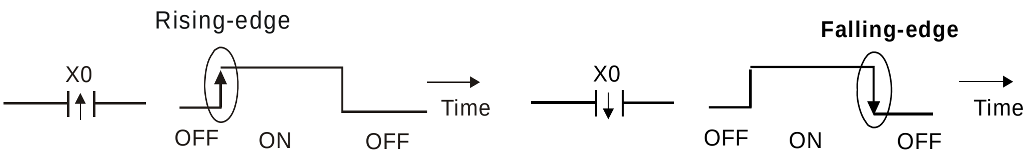
1.5.2 LD / LDI (Load NO contact / Load NC contact)

LD or LDI starts a row or block



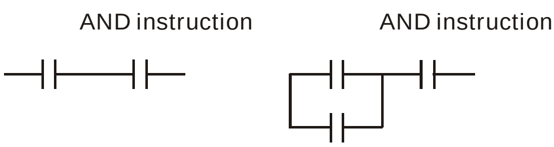
1.5.3 LDP / LDF (Load Rising edge trigger/ Load Falling edge trigger)

Similar to LD instruction, LDP and LDF instructions only act at the rising edge or falling edge when the contact is ON, as shown in the figure below.



1.5.4 AND / ANI (Connect NO contact in series / Connect NC contact in series)

AND (ANI) instruction connects a NO (NC) contact in series with another device or block.

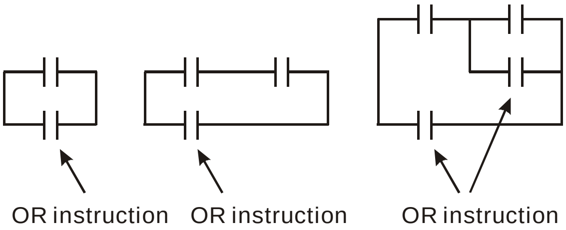


1.5.5 ANDP / ANDF (Connect Rising edge in series/ Connect Falling edge in series)

Similar to AND instruction, ANDP (ANDF) instruction connects rising (falling) edge triggers in series with another device or block.

1.5.6 OR / ORI (Connect NO contact in parallel / Connect NC contact in parallel)

OR (ORI) instruction connects a NO (NC) in parallel with another device or block.

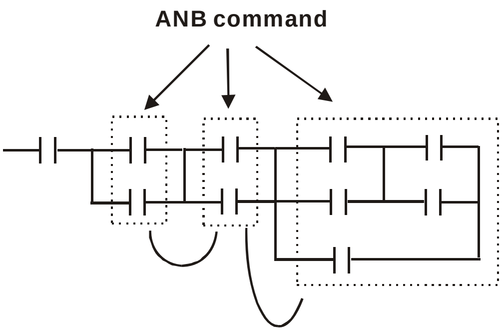


1.5.7 ORP / ORF (Connect Rising edge in parallel/ Connect Falling edge in parallel)

Similar to OR instruction, ORP (ORF) instruction connects rising (falling) edge triggers in parallel with another device or block

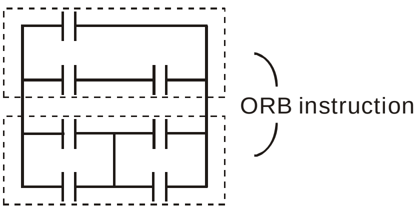
1.5.8 ANB (Connect block in series)

ANB instruction connects a block in series with another block



1.5.9 ORB (Connect block in parallel)

ORB instruction connects a block in parallel with another block



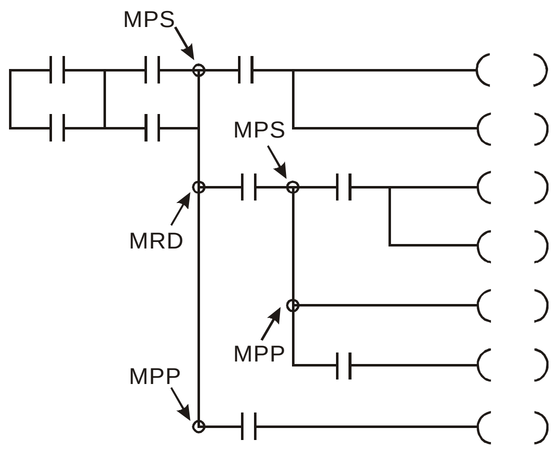
1.5.10 MPS / MRD / MPP (Branch instructions)

These instructions provide a method to create multiplexed output branches based on current result stored by MPS instruction.

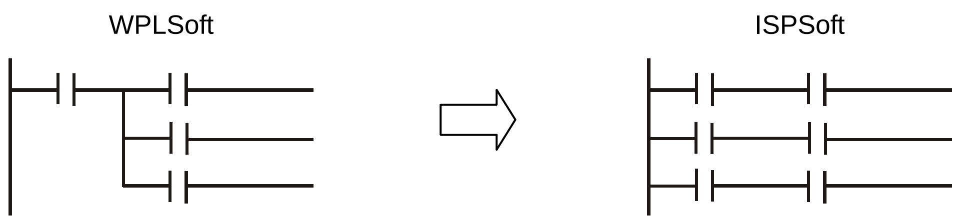
Branch instruction	Branch Symbol	Description
MPS	┬	Start of branches. Stores current result of program evaluation. Max. 8 MPS-MPP pairs can be applied
MRD	├	Reads the stored current result from previous MPS
MPP	└	End of branches. Pops (reads then resets) the stored result in previous MPS

Note: When compiling ladder diagram with WPLSoft, MPS, MRD and MPP could be automatically added to the compiled results in instruction format. However, sometimes the branch instructions are ignored by WPLSoft if not necessary. Users programming in instruction format can enter branch instructions as required.

Connection points of MPS, MRD and MPP:

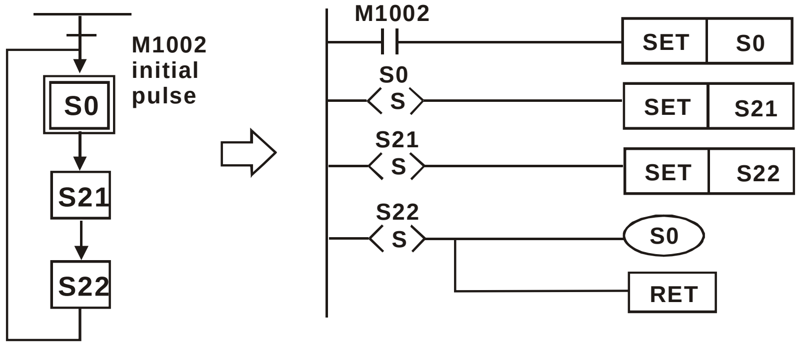


Note: Ladder diagram editor in ISPSoft does not support MPS, MRD and MPP instructions. To achieve the same results as branch instructions, users have to connect all branches to the left hand bus bar.



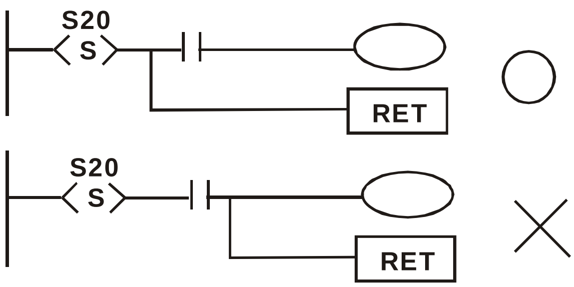
1.5.11 STL (Step Ladder Programming)

STL programming uses step points, e.g. S0 S21, S22, which allow users to program in a clearer and understandable way as drawing a flow chart. The program will proceed to next step only if the previous step is completed, therefore it forms a sequential control process similar to SFC (Sequential Function Chart) mode. The STL sequence can be converted into a PLC ladder diagram which is called “step ladder diagram” as below.



1.5.12 RET (Return)

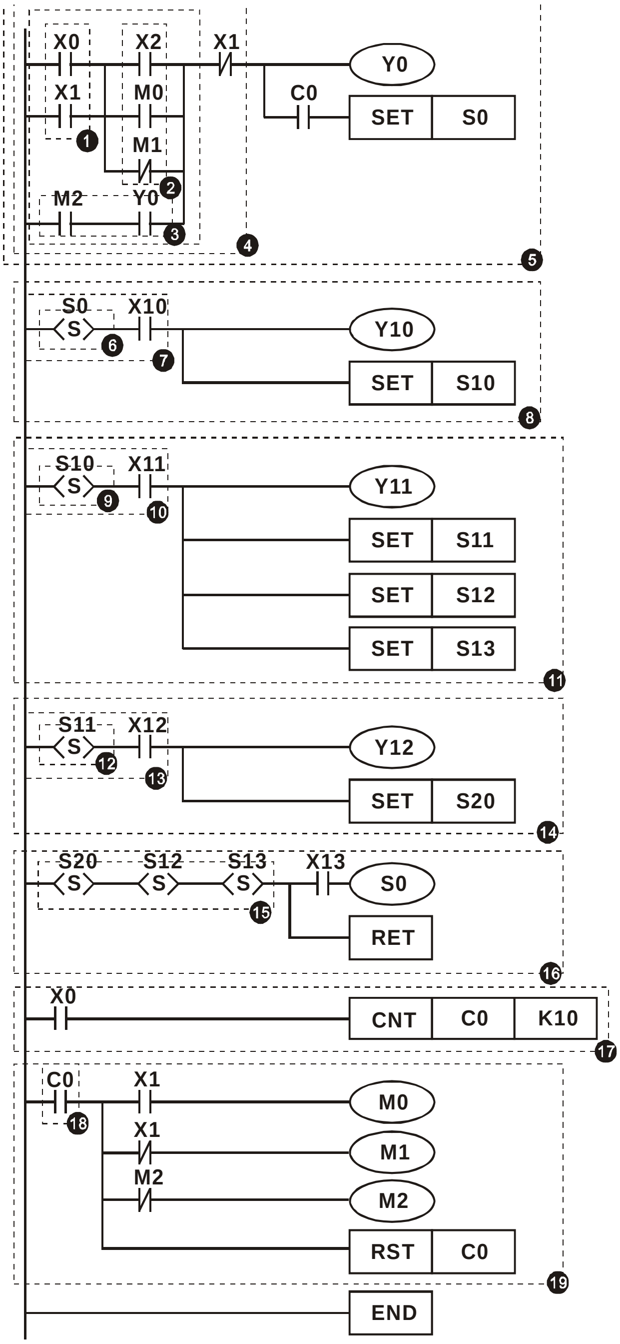
RET instruction has to be placed at the end of sequential control process to indicate the completion of STL flow.



Note: Always connect RET instruction immediately after the last step point indicated as the above diagram otherwise program error may occur.

1.6 Conversion between Ladder Diagram and Instruction List Mode

Ladder Diagram



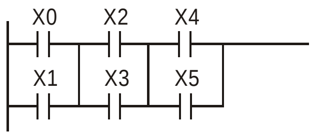
Instruction

LD	X0	❶ OR block
OR	X1	
LD	X2	❷ OR block
OR	M0	
ORI	M1	
ANB		Block in series
LD	M2	❸ AND block
AND	Y0	
ORB		Block in parallel
ANI	X1	❹ ANI
OUT	Y0	Multiple outputs; ❺ The output continues based on status of ❹
AND	C0	
SET	S0	
STL	S0	Start of step ladder ❻; ❼; ❽
LD	X10	S0 status operates with X10
OUT	Y10	Output Y10 and transfer of step point
SET	S10	
STL	S10	❾ Read S10 status
LD	X11	❿ S10 operates with X11; ⓫
OUT	Y11	
SET	S11	Output Y11 and transfer of step points
SET	S12	
SET	S13	
STL	S11	⓬ Read S11 status
LD	X12	⓭ S11 operates with X12; ⓮
OUT	Y12	Output Y12 and transfer of step points
SET	S20	
STL	S20	
STL	S12	⓯ Convergence of multiple status
STL	S13	
LD	X13	Read X13 status and transfer of step point; ⓰ End of step ladder
OUT	S0	
RET		Return
LD	X0	⓱
CNT	C0 K10	
LD	C0	⓲ Read C0
MPS		
AND	X1	⓳
OUT	M0	
MRD		
ANI	X1	Multiple outputs
OUT	M1	
MPP		
ANI	M2	
OUT	M2	
RST	C0	
END		End of program

1.7 Fuzzy Syntax

Generally, the ladder diagram programming is conducted according to the “up to down and left to right” principle. However, some programming methods not following this principle still perform the same control results. Here are some examples explaining this kind of “fuzzy syntax.”

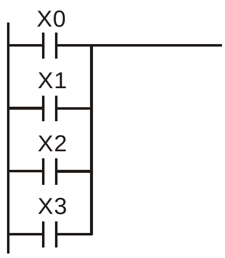
Example 1:



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	

The two instruction programs can be converted into the same ladder diagram. The difference between Better and OK method is the ANB operation conducted by MPU. ANB instruction cannot be used continuously for more than 8 times. If more than 8 ANB instructions are used continuously, program error will occur. Therefore, apply ANB instruction after a block is made is the better method to prevent the possible errors. In addition, it’s also the more logical and clearer programming method for general users.

Example 2:

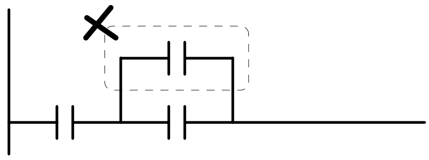
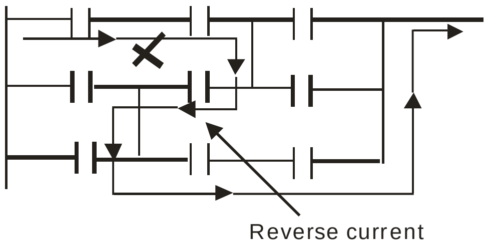
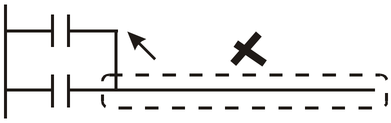
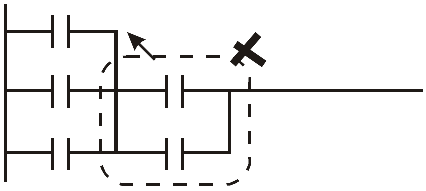
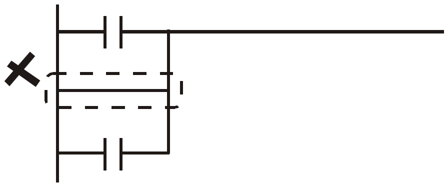
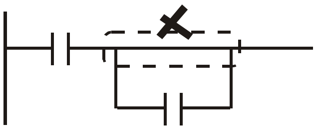
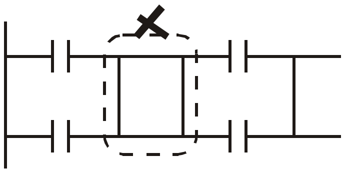
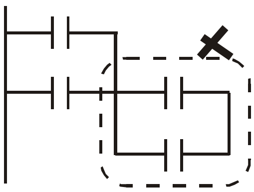
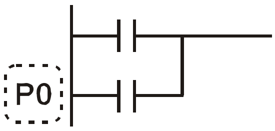


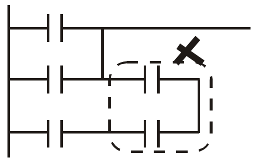
Good method		Bad method	
LD	X0	LD	X0
OR	X1	LD	X1
OR	X2	LD	X2
OR	X3	LD	X3
		ORB	
		ORB	
		ORB	

The difference between Good and Bad method is very clear. With longer program code, the required MPU operation memory increases in the Bad method. To sum up, following the general principle and applying good / better method when editing programs prevents possible errors and improves program execution speed as well.

Common Programming Errors

PLC processes the diagram program from up to down and left to right. When editing ladder diagram users should adopt this principle as well otherwise an error would be detected by WPLSoft when compiling user program. Common program errors are listed below:

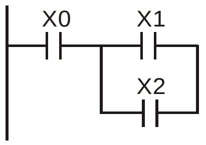
	OR operation upward is not allowed.
	“Reverse current” exists.
	Output should be connected on top of the circuit..
	Block combination should be made on top of the circuit..
	Parallel connection with empty device is not allowed..
	Parallel connection with empty device is not allowed.
	No device in the middle block.
	Devices and blocks in series should be horizontally aligned
	Label P0 should be at the first row of the complete network.

	“Reverse current” exists

1.8 Correcting Ladder Diagram

Example 1:

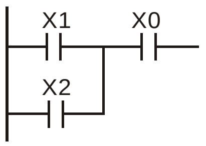
Connect the block to the front for omitting ANB instruction because simplified program improves processing speed



Instruction List

LD	X0
LD	X1
OR	X2
ANB	

⇩

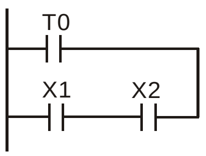


Instruction List

LD	X1
OR	X2
AND	X0

Example 2:

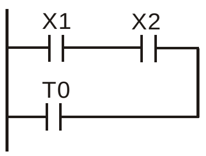
When a device is to be connected to a block, connect the device to upper row for omitting ORB instruction



Instruction List

LD	T0
LD	X1
AND	X2
ORB	

⇩

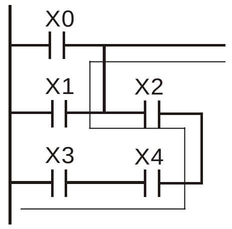


Instruction List

LD	X1
AND	X2
OR	T0

Example 3:

“Reverse current” existed in diagram (a) is not allowed for PLC processing principle.

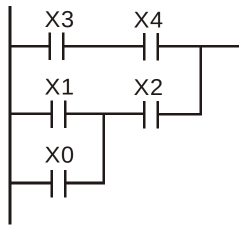


(a)

Instruction List

LD	X0
OR	X1
AND	X2
LD	X3
AND	X4
ORB	

⇩



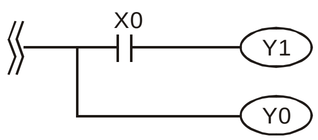
(b)

Instruction List

LD	X3
AND	X4
LD	X1
OR	X0
AND	X2
ORB	

Example 4:

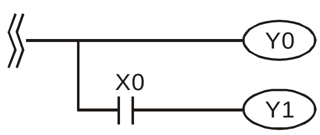
For multiple outputs, connect the output without additional input devices to the top of the circuit for omitting MPS and MPP instructions.



Instruction List

MPS	
AND	X0
OUT	Y1
MPP	
OUT	Y0

⇩

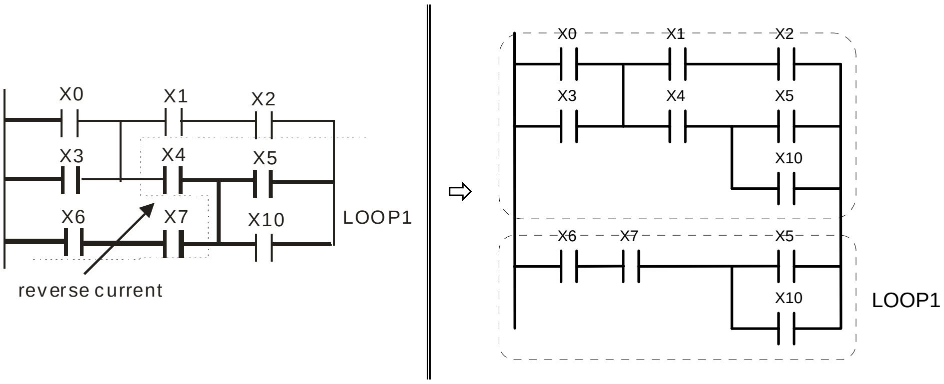


Instruction List

OUT	Y0
AND	X0
OUT	Y1

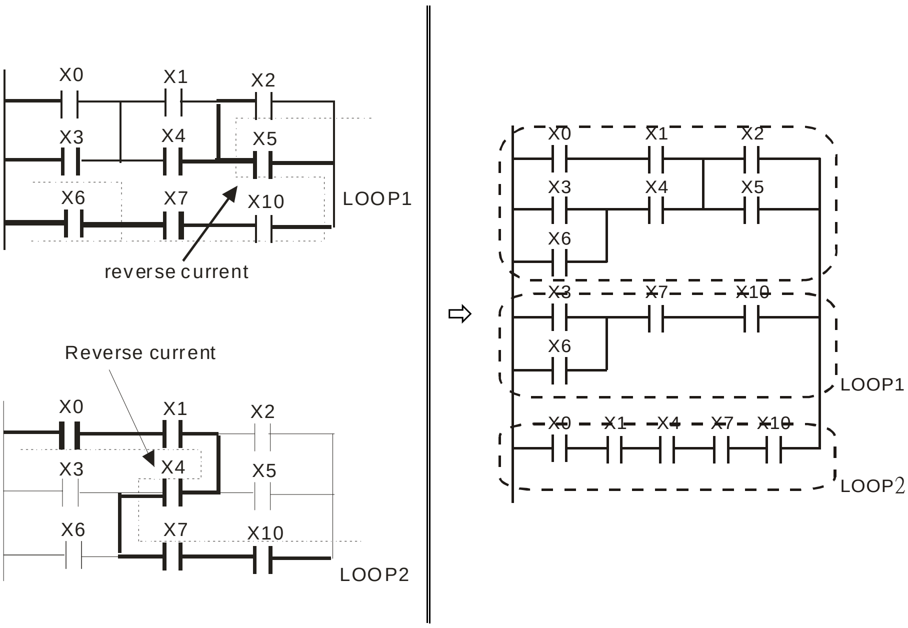
Example 5:

Correct the circuit of reverse current. The pointed reverse current loops are modified on the right.



Example 6:

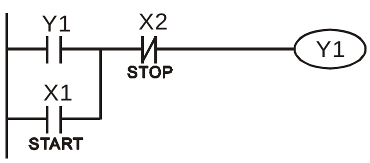
Correct the circuit of reverse current. The pointed reverse current loops are modified on the right.



1.9 Basic Program Design Examples

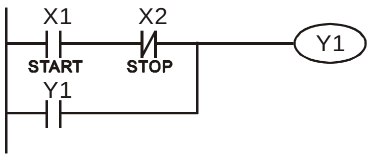
Example 1 - Stop First latched circuit

When X1 (START) = ON and X2 (STOP) = OFF, Y1 will be ON. If X2 is turned on, Y1 will be OFF. This is a Stop First circuit because STOP button has the control priority than START



Example 2 - Start First latched circuit

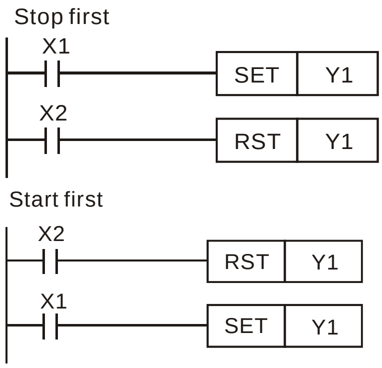
When X1 (START) = ON and X2 (STOP) = OFF, Y1 will be ON and latched. If X2 is turned ON, Y1 remains ON. This is a Start First circuit because START button has the control priority than STOP



Example 3 - Latched circuit of SET and RST

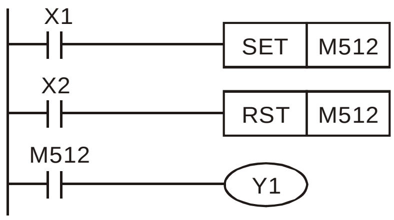
The diagram opposite are latched circuits consist of RST and SET instructions.

In PLC processing principle, the instruction close to the end of the program determines the final output status of Y1. Therefore, if both X1 and X2 are ON, RST which is lower than SET forms a Stop First circuit while SET which is lower than RST forms a Start First circuit.

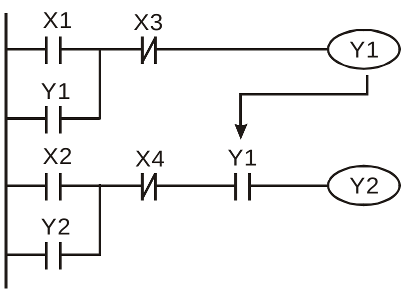


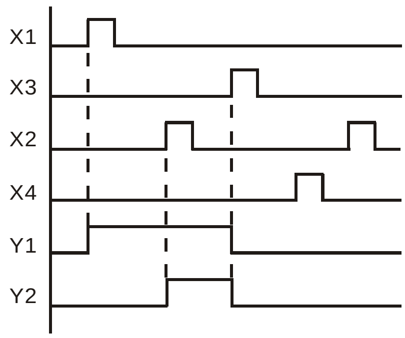
Example 4 - Power down latched circuit

The auxiliary relay M512 is a latched relay. Once X1 is ON, Y1 retains its status before power down and resumes after power up.



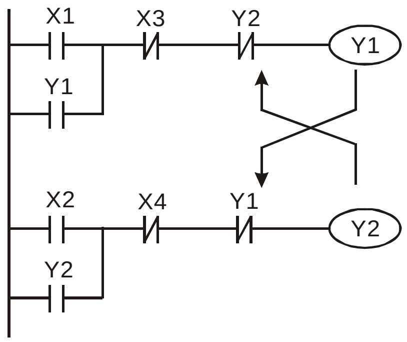
Example 5 - Conditional Control

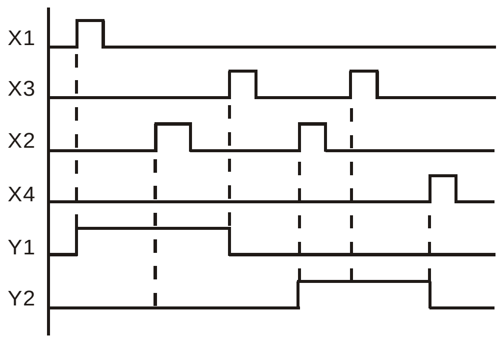




Because NO contact Y1 is connected to the circuit of Y2 output, Y1 becomes one of the conditions for enabling Y2, i.e. for turning on Y2, Y1 has to be ON

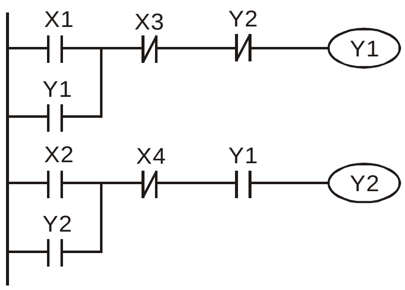
Example 6- Interlock control





NC contact Y1 is connected to Y2 output circuit and NC contact Y2 is connected Y1 output circuit. If Y1 is ON, Y2 will definitely be OFF and vice versa. This forms an Interlock circuit which prevents both outputs to be ON at the same time. Even if both X1 and X2 are ON, in this case only Y1 will be enabled.

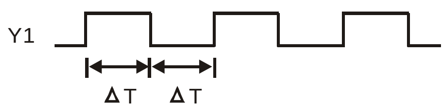
Example 7 - Sequential Control



Connect NC contact Y2 to Y1 output circuit and NO contact Y1 to Y2 output circuit. Y1 becomes one of the conditions to turn on Y2. In addition, Y1 will be OFF when Y2 is ON, which forms an sequential control process.

Example 8 - Oscillating Circuit

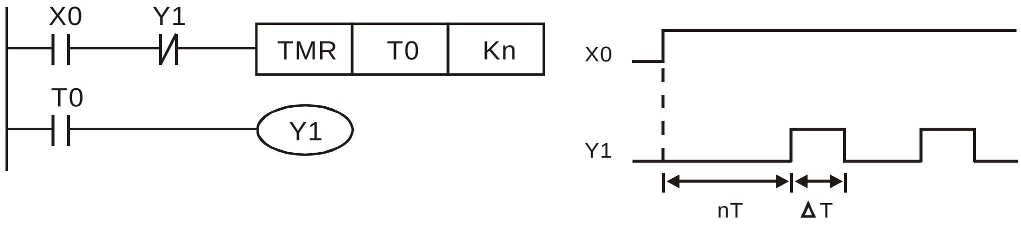
An oscillating circuit with cycle △T+△T



In the first scan, Y1 turns on. In the second scan, Y1 turns off due to the reversed state of contact Y1. Y1 output status changes in every scan and forms an oscillating circuit with output cycle△T(ON)+△T(OFF)

Example 9 – Oscillating Circuit with Timer

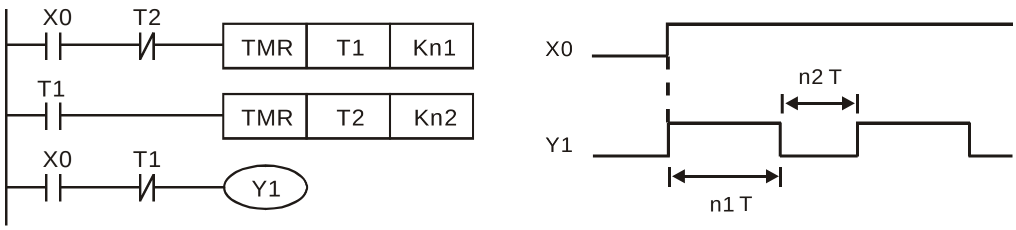
An oscillating circuit with cycle nT+ΔT



When X0 = ON, T0 starts timing (nT). Once the set time is reached, contact T0 = ON to enable Y1(ΔT). In next scan, Timer T0 is reset due to the reversed status of contact Y1. Therefore contact T0 is reset and Y1 = OFF. In next scan, T0 starts timing again. The process forms an oscillating circuit with output cycle nT+ΔT.

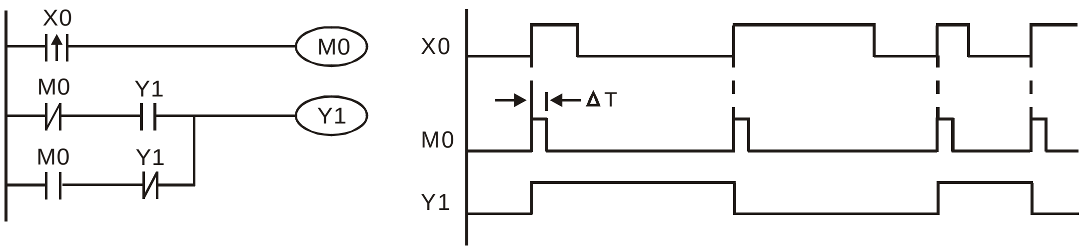
Example 10 - Flashing Circuit

The ladder diagram uses two timers to form an oscillating circuit which enables a flashing indicator or a buzzing alarm. n1 and n2 refer to the set values in T1 and T2 and T refers to timer resolution.



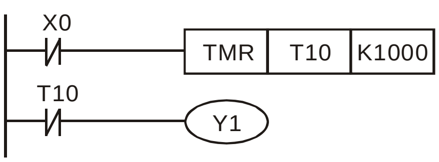
Example 11 - Trigger Circuit

In this diagram, rising-edge contact X0 generates trigger pulses to control two actions executing interchangeably.

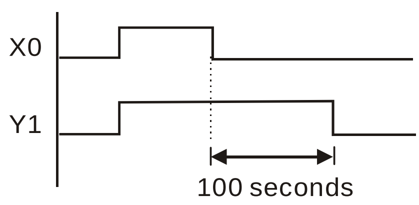


Example 12 - Delay OFF Circuit

If X0 = ON, timer T10 is not energized but coil Y1 is ON. When X0 is OFF, T10 is activated. After 100 seconds (K1000 × 0.1 sec = 100 sec), NC contact T10 is ON to turn off Y1. Turn-off action is delayed for 100 seconds by this delay OFF circuit.

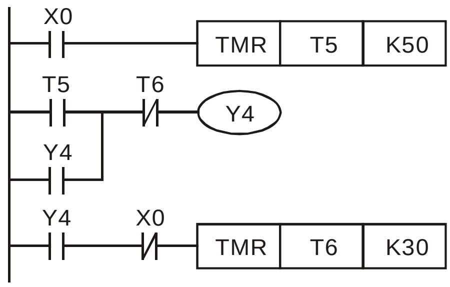


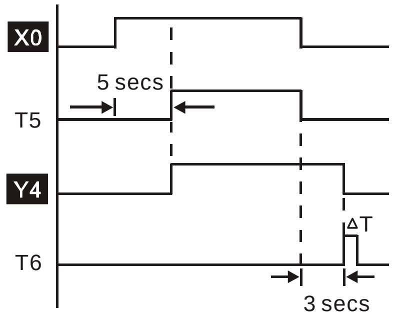
Timer Resolution: 0.1 sec



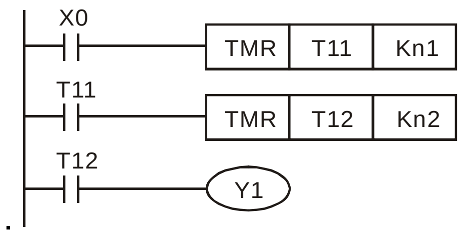
Example 13 - Output delay circuit

The output delay circuit is composed of two timers executing delay actions. No matter input X0 is ON or OFF, output Y4 will be delayed.



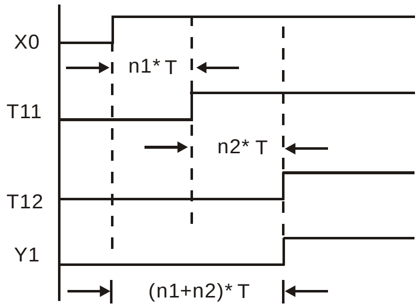


Example 14 - Timing extension circuit

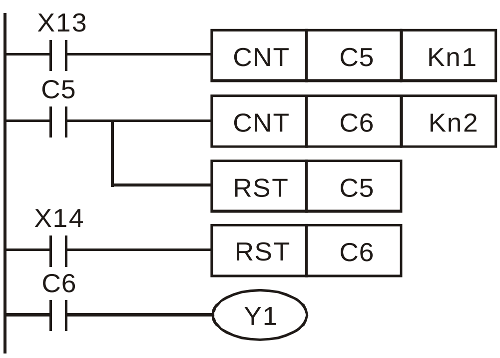


Timer = T11, T12
Timer resolution: T

The total delay time: (n1+n2)* T. T refers to the timer resolution.



Example 15 – Counting Range Extension Circuit

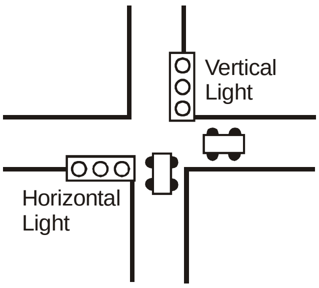


The counting range of a 16-bit counter is 0 ~ 32,767. The opposite circuit uses two counters to increase the counting range as n1*n2. When value in counter C6 reaches n2, The pulses counted from X13 will be n1*n2.

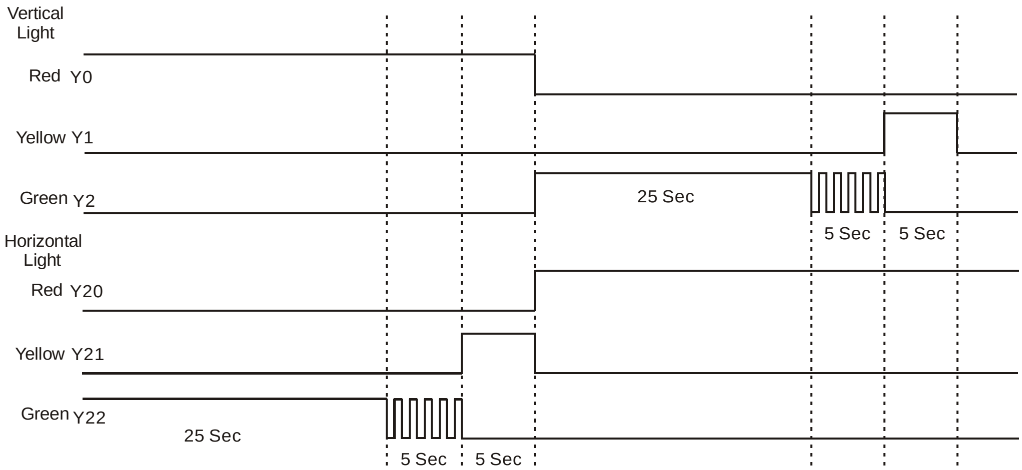
Example 16 - Traffic light control (Step Ladder Logic)

Traffic light control

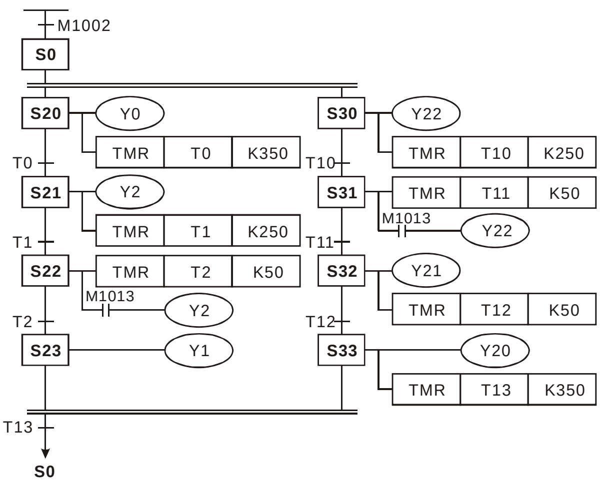
	Red light	Yellow light	Green light	Green light blinking
Vertical light	Y0	Y1	Y2	Y2
Horizontal light	Y20	Y21	Y22	Y22
Light Time	35 Sec	5 Sec	25 Sec	5 Sec



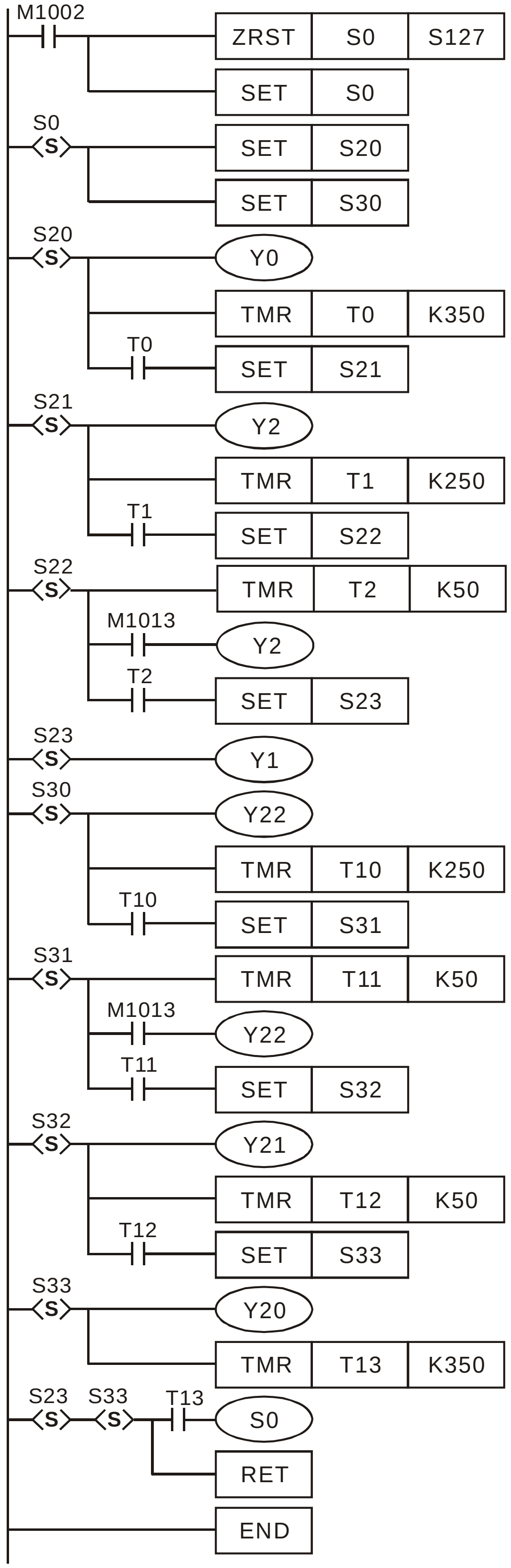
Timing Diagram:



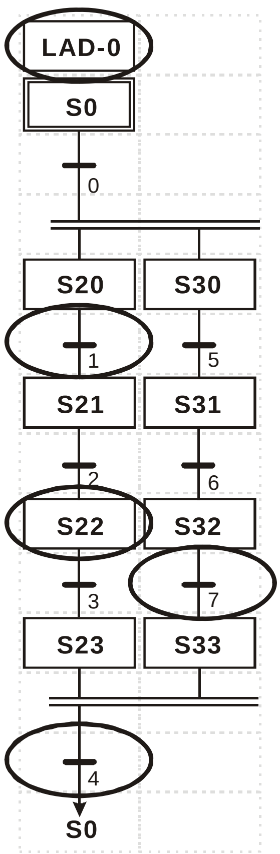
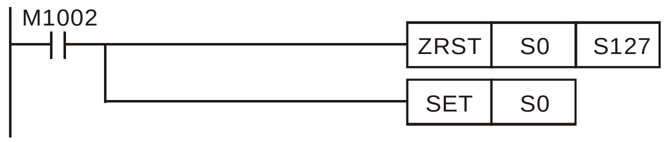
SFC Figure:

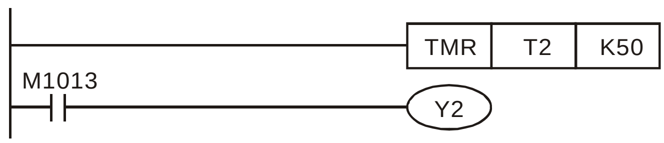


Ladder Diagram:



WPLSoft programming (SFC mode)

<table>
<tr><th>SFC logic</th><th>Internal Ladder Logic</th></tr>
<tr><td rowspan="5">
</td>
<td>LAD-0

</td></tr>
<tr><td>Transfer condition 1

</td></tr>
<tr><td>S22

</td></tr>
<tr><td>Transfer condition 4

</td></tr>
<tr><td>Transfer condition 7

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

MEMO

Programming Concepts 2

DVP-ES2/EX2/SS/SA2/SX2 is a programmable logic controller spanning an I/O range of 10–256 I/O points (SS2/SA2/SX2: 512 points). PLC can control a wide variety of devices to solve your automation needs. PLC monitors inputs and modifies outputs as controlled by the user program. User program provides features such as boolean logic, counting, timing, complex math operations, and communications to other communicating products.

Chapter Contents

2.1 ES2/EX2 Memory Map

<table>
<tr><th colspan="6">Specifications</th></tr>
<tr><td colspan="4">Control Method</td><td colspan="2">Stored program, cyclic scan system</td></tr>
<tr><td colspan="4">I/O Processing Method</td><td colspan="2">Batch processing method (when END instruction is executed)</td></tr>
<tr><td colspan="4">Execution Speed</td><td colspan="2">LD instructions – 0.54μs, MOV instructions – 3.4μs</td></tr>
<tr><td colspan="4">Program language</td><td colspan="2">Instruction List + Ladder + SFC</td></tr>
<tr><td colspan="4">Program Capacity</td><td colspan="2">15872 steps</td></tr>
<tr><td rowspan="17">Bit Contacts</td><td>X</td><td colspan="2">External inputs</td><td>X0~X377, octal number system, 256 points max, (*4)</td><td rowspan="2">Total 256+16 I/O</td></tr>
<tr><td>Y</td><td colspan="2">External outputs</td><td>Y0~Y377, octal number system, 256 points max, (*4)</td></tr>
<tr><td rowspan="3">M</td><td rowspan="3">Auxiliary relay</td><td>General</td><td>M0~M511, 512 points, (*1)
M768~M999, 232 points, (*1)
M2000~M2047, 48 points, (*1)</td><td rowspan="3">Total 4096 points</td></tr>
<tr><td>Latched</td><td>M512~M767, 256 points, (*2)
M2048~M4095, 2048 points, (*2)</td></tr>
<tr><td>Special</td><td>M1000~M1999, 1000 points, some are latched</td></tr>
<tr><td rowspan="6">T</td><td rowspan="6">Timer</td><td rowspan="3">100ms (M1028=ON, T64~T126: 10ms)</td><td>T0~T126, 127 points, (*1)
T128~T183, 56 points, (*1)</td><td rowspan="6">Total 256 points</td></tr>
<tr><td>T184~T199 for Subroutines, 16 points, (*1)</td></tr>
<tr><td>T250~T255(accumulative), 6 points (*1)</td></tr>
<tr><td rowspan="2">10ms (M1038=ON, T200~T245: 1ms)</td><td>T200~T239, 40 points, (*1)</td></tr>
<tr><td>T240~T245(accumulative), 6 points, (*1)</td></tr>
<tr><td>1ms</td><td>T127, 1 points, (*1)
T246~T249(accumulative), 4 points, (*1)</td></tr>
<tr><td rowspan="4">C</td><td rowspan="4">Counter</td><td rowspan="2">16-bit count up</td><td>C0~C111, 112 points, (*1)
C128~C199,72 points, (*1)</td><td rowspan="4">Total 232 points</td></tr>
<tr><td>C112~C127,16 points, (*2)</td></tr>
<tr><td rowspan="2">32-bit count up/down</td><td>C200~C223, 24 points, (*1)</td></tr>
<tr><td>C224~C231, 8 points, (*2)</td></tr>
</table>

Specifications						
			32bit high-speed count up/down	Soft-ware	C235~C242, 1 phase 1 input, 8 points, (*2)	Total 23 points
					C232~C234, 2 phase 2 input, 3 points, (*2)	
				Hard-ware	C243~C244, 1 phase 1 input, 2 points, (*2)	
					C245~C250, 1 phase 2 input, 6 points, (*2)	
					C251~C254 2 phase 2 input, 4 points, (*2)	
	S	Step point	Initial step point		S0~S9, 10 points, (*2)	Total 1024 points
			Zero point return		S10~S19, 10 points (use with IST instruction), (*2)	
			Latched		S20~S127, 108 points, (*2)	
			General		S128~S911, 784 points, (*1)	
			Alarm		S912~S1023, 112 points, (*2)	
Word Register	T	Current value			T0~T255, 256 words	
	C	Current value			C0~C199, 16-bit counter, 200 words	
					C200~C254, 32-bit counter, 55 words	
	D	Data register	General		D0~D407, 408 words, (*1) D600~D999, 400 words, (*1) D3920~D9999, 6080 words, (*1)	Total 10000 points
			Latched		D408~D599, 192 words, (*2) D2000~D3919, 1920 words, (*2)	
			Special		D1000~D1999, 1000 words, some are latched	
			For Special mudules		D9900~D9999，100 words， (*1), (*5)	
			Index		E0~E7, F0~F7, 16 words, (*1)	
Pointer	N	Master control loop			N0~N7, 8 points	
	P	Pointer			P0~P255, 256 points	
	I	Interrupt Service	External interrupt		I000/I001(X0), I100/I101(X1), I200/I201(X2), I300/I301(X3), I400/I401(X4), I500/I501(X5), I600/I601(X6), I700/I701(X7), 8 points (01: rising-edge trigger, 00: falling-edge trigger)	

Specifications				
			Timer interrupt	I602~I699, I702~I799, 2 points (Timer resolution = 1ms)
			High-speed counter interrupt	I010, I020, I030, I040, I050, I060, I070, I080,8 points
			Communication interrupt	I140(COM1), I150(COM2), I160(COM3), 3 points, (*3)
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: built-in RS-232 ((Master/Slave) COM2: built-in RS-485 (Master/Slave) COM3: built-in RS-485 (Master/Slave) COM1 is typically the programming port.
Real Time Clock				Year, Month, Day, Week, Hours, Minutes, Seconds
Special I/O Modules				Up to 8 special I/O modules can be connected

Notes:

1. Non-latched area cannot be modified
2. Latched area cannot be modified
3. COM1: built-in RS232 port. COM2: built-in RS485 port. COM3: built-in RS485 port.
4. When input points(X) are expanded to 256 points, only 16 output points(Y) are applicable. Also, when ouput points(Y) are expanded to 256 points, only 16 input points(X) are applicable.
5. This area is applicable only when the ES2/EX2 MPU is connected with special I/O modules. Every special I/O module occupies 10 points.

2.2 SS2 Memory Map

Specifications					
Control Method				Stored program, cyclic scan system	
I/O Processing Method				Batch processing method (when END instruction is executed)	
Execution Speed				LD instructions – 0.54μs, MOV instructions – 3.4μs	
Program language				Instruction List + Ladder + SFC	
Program Capacity				7920 steps	
Bit Contacts	X	External inputs		X0~X377, octal number system, 256 points max.	Total 480+14 I/O(*4)
Bit Contacts	Y	External outputs		Y0~Y377, octal number system, 256 points max.	Total 480+14 I/O(*4)
Bit Contacts	M	Auxiliary relay	General	M0~M511, 512 points, (*1) M768~M999, 232 points, (*1) M2000~M2047, 48 points, (*1)	Total 4096 points
Bit Contacts	M	Auxiliary relay	Latched	M512~M767, 256 points, (*2) M2048~M4095, 2048 points, (*2)	Total 4096 points
Bit Contacts	M	Auxiliary relay	Special	M1000~M1999, 1000 points, some are latched	Total 4096 points
Bit Contacts	T	Timer	100ms (M1028=ON, T64~T126: 10ms)	T0~T126, 127 points, (*1) T128~T183, 56 points, (*1)	Total 256 points
Bit Contacts	T	Timer	100ms (M1028=ON, T64~T126: 10ms)	T184~T199 for Subroutines, 16 points, (*1)	Total 256 points
Bit Contacts	T	Timer	100ms (M1028=ON, T64~T126: 10ms)	T250~T255(accumulative), 6 points (*1)	Total 256 points
Bit Contacts	T	Timer	10ms (M1038=ON, T200~T245: 1ms)	T200~T239, 40 points, (*1)	Total 256 points
Bit Contacts	T	Timer	10ms (M1038=ON, T200~T245: 1ms)	T240~T245(accumulative), 6 points, (*1)	Total 256 points
Bit Contacts	T	Timer	1ms	T127, 1 points, (*1) T246~T249(accumulative), 4 points, (*1)	Total 256 points
Bit Contacts	C	Counter	16-bit count up	C0~C111, 112 points, (*1) C128~C199, 72 points, (*1)	Total 233 points
Bit Contacts	C	Counter	16-bit count up	C112~C127, 16 points, (*2)	Total 233 points
Bit Contacts	C	Counter	32-bit count up/down	C200~C223, 24 points, (*1)	Total 233 points
Bit Contacts	C	Counter	32-bit count up/down	C224~C232, 9 points, (*2)	Total 233 points

Specifications						
			32bit high-speed count up/down	Soft-ware	C235~C242, 1 phase 1 input, 8 points, (*2)	Total 22 points
					C233~C234, 2 phase 2 input, 2 points, (*2)	
				Hard-ware	C243~C244, 1 phase 1 input, 2 points, (*2)	
					C245~C250, 1 phase 2 input, 6 points, (*2)	
					C251~C254 2 phase 2 input, 4 points, (*2)	
	S	Step point	Initial step point		S0~S9, 10 points, (*2)	Total 1024 points
			Zero point return		S10~S19, 10 points (use with IST instruction), (*2)	
			Latched		S20~S127, 108 points, (*2)	
			General		S128~S911, 784 points, (*1)	
			Alarm		S912~S1023, 112 points, (*2)	
Word Register	T	Current value			T0~T255, 256 words	
	C	Current value			C0~C199, 16-bit counter, 200 words	
					C200~C254, 32-bit counter, 55 words	
	D	Data register	General		D0~D407, 408 words, (*1) D600~D999, 400 words, (*1) D3920~D4999, 1080 words, (*1)	Total 5016 points
			Latched		D408~D599, 192 words, (*2) D2000~D3919, 1920 words, (*2)	
			Special		D1000~D1999, 1000 words, some are latched	
			Index		E0~E7, F0~F7, 16 words, (*1)	
Pointer	N	Master control loop			N0~N7, 8 points	
	P	Pointer			P0~P255, 256 points	
	I	Interrupt Service	External interrupt		I000/I001(X0), I100/I101(X1), I200/I201(X2), I300/I301(X3), I400/I401(X4), I500/I501(X5), I600/I601(X6), I700/I701(X7), 8 points (01: rising-edge trigger, 00: falling-edge trigger)	
			Timer interrupt		I602~I699, I702~I799, 2 points (Timer resolution = 1ms)	

Specifications				
			High-speed counter interrupt	I010, I020, I030, I040, I050, I060, I070, I080, 8 points
			Communication interrupt	I140(COM1), I150(COM2), 2 points, (*3)
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: built-in RS-232 ((Master/Slave) COM2: built-in RS-485 (Master/Slave) COM1 is typically the programming port.
Real Time Clock				Year, Month, Day, Week, Hours, Minutes, Seconds
Special I/O Modules				Up to 8 special I/O modules can be connected

Notes:

1. Non-latched area cannot be modified
2. Latched area cannot be modified
3. COM1: built-in RS232 port. COM2: built-in RS485 port.
4. SS2 MPU occupies 16 input points (X0~X17) and 16 output points (Y0~Y17).

2.3 SA2 Memory Map

Specifications					
Control Method				Stored program, cyclic scan system	
I/O Processing Method				Batch processing method (when END instruction is executed)	
Execution Speed				LD instructions – 0.54μs, MOV instructions – 3.4μs	
Program language				Instruction List + Ladder + SFC	
Program Capacity				15872 steps	
Bit Contacts	X	External inputs		X0~X377, octal number system, 256 points max.	Total 480+14 I/O(*4)
	Y	External outputs		Y0~Y377, octal number system, 256 points max.	
	M	Auxiliary relay	General	M0~M511, 512 points, (*1) M768~M999, 232 points, (*1) M2000~M2047, 48 points, (*1)	Total 4096 points
			Latched	M512~M767, 256 points, (*2) M2048~M4095, 2048 points, (*2)	
			Special	M1000~M1999, 1000 points, some are latched	
	T	Timer	100ms (M1028=ON, T64~T126: 10ms)	T0~T126, 127 points, (*1) T128~T183, 56 points, (*1)	Total 256 points
				T184~T199 for Subroutines, 16 points, (*1)	
				T250~T255(accumulative), 6 points (*1)	
			10ms (M1038=ON, T200~T245: 1ms)	T200~T239, 40 points, (*1)	
				T240~T245(accumulative), 6 points, (*1)	
			1ms	T127, 1 points, (*1) T246~T249(accumulative), 4 points, (*1)	
	C	Counter	16-bit count up	C0~C111, 112 points, (*1) C128~C199, 72 points, (*1)	Total 233 points
				C112~C127, 16 points, (*2)	
			32-bit count up/down	C200~C223, 24 points, (*1)	
				C224~C232, 9 points, (*2)	
			32bit high- / Soft-ware	C235~C242, 1 phase 1 input, 8 points, (*2)	Total 22 points

Specifications						
			speed count up/down	Hard-ware	C233~C234, 2 phase 2 input, 2 points, (*2)	
					C243~C244, 1 phase 1 input, 2 points, (*2)	
					C245~C250, 1 phase 2 input, 6 points, (*2)	
					C251~C254 2 phase 2 input, 4 points, (*2)	
	S	Step point	Initial step point		S0~S9, 10 points, (*2)	Total 1024 points
			Zero point return		S10~S19, 10 points (use with IST instruction), (*2)	
			Latched		S20~S127, 108 points, (*2)	
			General		S128~S911, 784 points, (*1)	
			Alarm		S912~S1023, 112 points, (*2)	
Word Register	T	Current value			T0~T255, 256 words	
	C	Current value			C0~C199, 16-bit counter, 200 words	
					C200~C254, 32-bit counter, 55 words	
	D	Data register	General		D0~D407, 408 words, (*1) D600~D999, 400 words, (*1) D3920~D9999, 6080 words, (*1)	Total 10000 points
			Latched		D408~D599, 192 words, (*2) D2000~D3919, 1920 words, (*2)	
			Special		D1000~D1999, 1000 words, some are latched	
			Index		E0~E7, F0~F7, 16 words, (*1)	
Pointer	N	Master control loop			N0~N7, 8 points	
	P	Pointer			P0~P255, 256 points	
	I	Interrupt Service	External interrupt		I000/I001(X0), I100/I101(X1), I200/I201(X2), I300/I301(X3), I400/I401(X4), I500/I501(X5), I600/I601(X6), I700/I701(X7), 8 points (01: rising-edge trigger, 00: falling-edge trigger)	
			Timer interrupt		I602~I699, I702~I799, 2 points (Timer resolution = 1ms)	

Specifications				
			High-speed counter interrupt	I010, I020, I030, I040, I050, I060, I070, I080, 8 points
			Communication interrupt	I140(COM1), I150(COM2), I160(COM3), 3 points, (*3)
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: built-in RS-232 ((Master/Slave) COM2: built-in RS-485 (Master/Slave) COM3: built-in RS-485 (Master/Slave) COM1 is typically the programming port.
Real Time Clock				Year, Month, Day, Week, Hours, Minutes, Seconds
Special I/O Modules				Up to 8 special I/O modules can be connected

Notes:

1. Non-latched area cannot be modified
2. Latched area cannot be modified
3. COM1: built-in RS232 port. COM2: built-in RS485 port. COM3: built-in RS-485 port
4. SA2 MPU occupies 16 input points (X0~X17) and 16 output points (Y0~Y17).

2.4 SX2 Memory Map

<table>
<tr><th colspan="7">Specifications</th></tr>
<tr><td colspan="5">Control Method</td><td colspan="2">Stored program, cyclic scan system</td></tr>
<tr><td colspan="5">I/O Processing Method</td><td colspan="2">Batch processing method (when END instruction is executed)</td></tr>
<tr><td colspan="5">Execution Speed</td><td colspan="2">LD instructions – 0.54μs, MOV instructions – 3.4μs</td></tr>
<tr><td colspan="5">Program language</td><td colspan="2">Instruction List + Ladder + SFC</td></tr>
<tr><td colspan="5">Program Capacity</td><td colspan="2">15872 steps</td></tr>
<tr><td rowspan="16">Bit Contacts</td><td>X</td><td colspan="3">External inputs</td><td>X0~X377, octal number system, 256 points max.</td><td rowspan="2">Total 480+14 I/O(*4)</td></tr>
<tr><td>Y</td><td colspan="3">External outputs</td><td>Y0~Y377, octal number system, 256 points max.</td></tr>
<tr><td rowspan="3">M</td><td rowspan="3">Auxiliary relay</td><td colspan="2">General</td><td>M0~M511, 512 points, (*1)
M768~M999, 232 points, (*1)
M2000~M2047, 48 points, (*1)</td><td rowspan="3">Total 4096 points</td></tr>
<tr><td colspan="2">Latched</td><td>M512~M767, 256 points, (*2)
M2048~M4095, 2048 points, (*2)</td></tr>
<tr><td colspan="2">Special</td><td>M1000~M1999, 1000 points, some are latched</td></tr>
<tr><td rowspan="6">T</td><td rowspan="6">Timer</td><td rowspan="3" colspan="2">100ms
(M1028=ON, T64~T126: 10ms)</td><td>T0~T126, 127 points, (*1)
T128~T183, 56 points, (*1)</td><td rowspan="6">Total 256 points</td></tr>
<tr><td>T184~T199 for Subroutines, 16 points, (*1)</td></tr>
<tr><td>T250~T255(accumulative), 6 points (*1)</td></tr>
<tr><td rowspan="2" colspan="2">10ms
(M1038=ON, T200~T245: 1ms)</td><td>T200~T239, 40 points, (*1)</td></tr>
<tr><td>T240~T245(accumulative), 6 points, (*1)</td></tr>
<tr><td colspan="2">1ms</td><td>T127, 1 points, (*1)
T246~T249(accumulative), 4 points, (*1)</td></tr>
<tr><td rowspan="5">C</td><td rowspan="5">Counter</td><td rowspan="2" colspan="2">16-bit count up</td><td>C0~C111, 112 points, (*1)
C128~C199, 72 points, (*1)</td><td rowspan="4">Total 232 points</td></tr>
<tr><td>C112~C127, 16 points, (*2)</td></tr>
<tr><td rowspan="2" colspan="2">32-bit count up/down</td><td>C200~C223, 24 points, (*1)</td></tr>
<tr><td>C224~C231, 8 points, (*2)</td></tr>
<tr><td>32bit high-</td><td>Soft- ware</td><td>C235~C242, 1 phase 1 input, 8 points, (*2)</td><td>Total 23 points</td></tr>
</table>

Specifications						
			speed count up/down	Hard-ware	C232~C234, 2 phase 2 input, 2 points, (*2)	
					C243~C244, 1 phase 1 input, 2 points, (*2)	
					C245~C250, 1 phase 2 input, 6 points, (*2)	
					C251~C254 2 phase 2 input, 4 points, (*2)	
	S	Step point	Initial step point		S0~S9, 10 points, (*2)	Total 1024 points
			Zero point return		S10~S19, 10 points (use with IST instruction), (*2)	
			Latched		S20~S127, 108 points, (*2)	
			General		S128~S911, 784 points, (*1)	
			Alarm		S912~S1023, 112 points, (*2)	
Word Register	T	Current value			T0~T255, 256 words	
	C	Current value			C0~C199, 16-bit counter, 200 words	
					C200~C254, 32-bit counter, 55 words	
	D	Data register	General		D0~D407, 408 words, (*1) D600~D999, 400 words, (*1) D3920~D9999, 6080 words, (*1)	Total 10000 points
			Latched		D408~D599, 192 words, (*2) D2000~D3919, 1920 words, (*2)	
			Special		D1000~D1999, 1000 words, some are latched	
			Index		E0~E7, F0~F7, 16 words, (*1)	
Pointer	N	Master control loop			N0~N7, 8 points	
	P	Pointer			P0~P255, 256 points	
	I	Interrupt Service	External interrupt		I000/I001(X0), I100/I101(X1), I200/I201(X2), I300/I301(X3), I400/I401(X4), I500/I501(X5), I600/I601(X6), I700/I701(X7), 8 points (01: rising-edge trigger ⎍, 00: falling-edge trigger ⎍)	
			Timer interrupt		I602~I699, I702~I799, 2 points (Timer resolution = 1ms)	

Specifications				
			High-speed counter interrupt	I010, I020, I030, I040, I050, I060, I070, I080, 8 points
			Communication interrupt	I140(COM1), I150(COM2), 2 points, (*3)
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: built-in RS-232 ((Master/Slave) COM2: built-in RS-485 (Master/Slave) COM3: built-in USB port (Slave) COM1 is typically the programming port.
Real Time Clock				Year, Month, Day, Week, Hours, Minutes, Seconds
Special I/O Modules				Right side: Up to 8 special I/O modules can be connected Left side: Up to 8 high-speed I/O modules can be connected

Notes:

1. Non-latched area cannot be modified
2. Latched area cannot be modified
3. COM1: built-in RS232 port. COM2: built-in RS485 port.
4. SX2 MPU occupies 16 input points (X0~X17) and 16 output points (Y0~Y17).

2.5 Status and Allocation of Latched Memory

Memory type	Power OFF=>ON	STOP=>RUN	RUN=>STOP	Clear all non-latched area (M1031=ON)	Clear all latched area (M1032=ON)	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

M Auxiliary relay	General	Latched	Special auxiliary relay
	M0~M511 M768~M999 M2000~M2047	M512~M999 M2048~M4095	M1000~M1999
	Not latched	Latched	Some are latched and can't be changed.

T Timer							
	100 ms	100 ms	1 ms	10 ms	10ms	1 ms	100 ms
	T0 ~T126 T128~T183	T184~T199	T127	T200~T239	T240~T245	T246~T249	T250~T255
	M1028=1,T64~T126:10ms	For subroutine	-	M1038=1,T200~T245: 1ms		-	
	non-latched	non-latched			Accumulative non-latched		

C Counter	16-bit count up		32-bit count up/down		32-bit high-speed count up/down
	C0~C111 C128~C199	C112~C127	C200~C223	C224~C231	C232~C254
	Non-latched	Latched	Non-latched	Latched	Latched

S Step relay	Initial	Zero return	Latched	General	Step alarm
	S0~S9	S10~S19	S20~S127	S128~S911	S912~S1023
	Latched			Non-latched	Latched

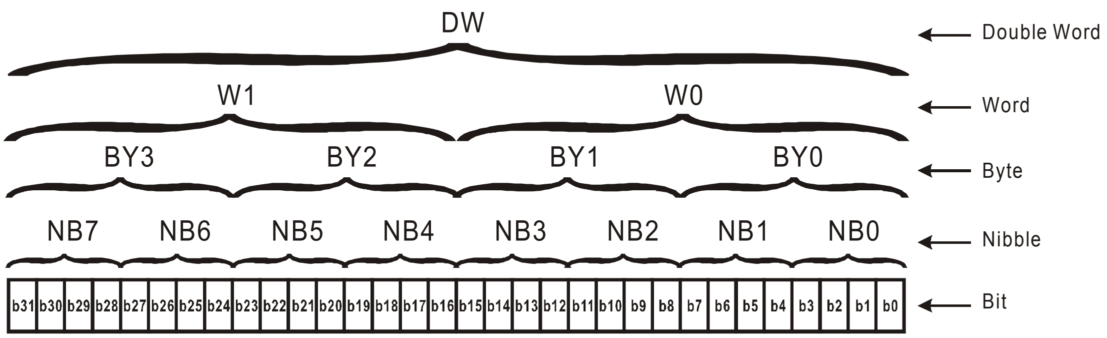
D Register	General	Latched	Special register	For AIO
	D0~D407 D600~D999 D3920~D9899	D408~D599 D2000~D3919	D1000~D1999	D9900~D9999
	Non-latched	Latched	Some are latched, and can't be changed	Non-latched

2.6 PLC Bits, Nibbles, Bytes, Words, etc

For different control purposes, there are five types of values inside DVP-PLC for executing the operations.

Numeric	Description
Bit	Bit is the basic unit of a binary number system. Range is 0 or 1
Nibble	Consists of 4 consecutive bits, e.g. b3~b0. Range 0 ~ 9 in Decimal or 0~F in Hex
Byte	Consists of 2 consecutive nibbles, e.g. b7~b0. Range 00 ~ FF in Hex
Word	Consists of 2 consecutive bytes, e.g. b15~b0. Range 0000 ~ FFFF in Hex
Double Word	Consists of 2 consecutive words, e.g. b31~b1. Range 00000000 - FFFFFFFF in Hex

Bit, nibble, byte, word, and double word in a binary system:



2.7 Binary, Octal, Decimal, BCD, Hex

For fulllfilling different kinds of internal manipulation, DVP-PLC appies 5 foramts of number systems. Each number system has its specific purpose and function described as below.

1. Binary Number, (BIN)

   PLC internally calculates, operates, and stores the value in Binary format.

2. Octal Number, (OCT)

   The external I/O points of DVP-PLC are numbered in octal format.

   e.g.
   External inputs: X0～X7, X10～X17, …, X377. (No. of device)
   External outputs: Y0～Y7, Y10～Y17, …, Y377. (No. of device)

3. Decimal Number, (DEC)

   DVP-PLC appies decimal operation in situations below:

- Set value for timers and counters, e.g. TMR C0 K50. (K value)
- No. of S, M, T, C, D, E, F, P, I devices, e.g. M10, T30. (No. of device)
- For use of operand in API instructions, e.g. MOV K123 D0. (K value)

- Constant K:

Decimal value in PLC operation is attached with an “K”, e.g. K100 indicates the value 100 in Decimal format.

Exception:
When constant K is used with bit devices X, Y, M, S, the value specifed after K indicates the groups of 4-bit unit, which forms a digit(4-bit), byte(8 bit), word(16bit), or double word(32-bit) data, e.g. K2Y10, K4M100, representing Y10 ~ Y17 and M100~M115.

4. BCD (Binary Coded Decimal)

BCD format takes 1 digit or 4 bits to indicate a Decimal value, so that data of consecutive 16 bits indicates a 4-digit decimal value. Used mainly for reading values from DIP switches or sending data to 7-segement displays

5. Hexadecimal Number, HEX

DVP-PLC appies Hexadecimal operation in situations below:

2

- For use of operand in API instructions, e.g. MOV H1A2B D0。(H value)
- Constant H:

Hexadecimal value in PLC operation is attached with an “H”, e.g. H100 indicates the value 100 in Hex format.

Reference Table:

Binary (BIN)	**Octal (OCT)**	**Decimal (K) (DEC)**	**BCD (Binary Code Decimal)**	**Hexadecimal (H) (HEX)**
For PLC internal operation	No. of X, Y relay	Costant K, No. of registers M, S, T, C, D, E, F, P, I devices	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

Binary (BIN)	Octal (OCT)	Decimal (K) (DEC)	BCD (Binary Code Decimal)	Hexadecimal (H) (HEX)
For PLC internal operation	No. of X, Y relay	Costant K, No. of registers M, S, T, C, D, E, F, P, I devices	For DIP Switch and 7-segment display	Constant H
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

2.8 M Relay

The types and functions of special auxiliary relays (special M) are listed in the table below. Care should be taken that some devices of the same No. may bear different meanings in different series MPUs. Special M and special D marked with "*" will be further illustrated in 2.13. Columns marked with "R" refers to "read only", "R/W" refers to "read and write", "-" refers to the status remains unchanged and "#" refers to that system will set it up according to the status of the PLC.

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch-ed	Default
M1000*	Monitor normally open contact	○	○	○	○	OFF	ON	OFF	R	NO	OFF
M1001*	Monitor normally closed contact	○	○	○	○	ON	OFF	ON	R	NO	ON
M1002*	Enable single positive pulse at the moment when RUN is activate (Normally OFF)	○	○	○	○	OFF	ON	OFF	R	NO	OFF
M1003*	Enable single negative pulse at the moment when RUN is activate (Normally ON)	○	○	○	○	ON	OFF	ON	R	NO	ON
M1004*	ON when syntax errors occur	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1008*	Watchdog timer (ON: PLC WDT time out)	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1009	Indicate LV signal due to 24VDC insufficiency	○	○	○	○	OFF	-	-	R	NO	OFF
M1011*	10ms clock pulse, 5ms ON/5ms OFF	○	○	○	○	OFF	-	-	R	NO	OFF
M1012*	100ms clock pulse, 50ms ON / 50ms OFF	○	○	○	○	OFF	-	-	R	NO	OFF
M1013*	1s clock pulse, 0.5s ON / 0.5s OFF	○	○	○	○	OFF	-	-	R	NO	OFF
M1014*	1 min clock pulse, 30s ON / 30s OFF	○	○	○	○	OFF	-	-	R	NO	OFF
M1015*	Enable high-speed timer	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1016*	Indicate Year display mode of RTC.	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1017*	±30 seconds correction on real time clock	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1018	Flag for Radian/Degree, ON for degree	○	○	○	○	OFF	-	-	R/W	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1020	Zero flag	○	○	○	○	OFF	-	-	R	NO	OFF
M1021	Borrow flag	○	○	○	○	OFF	-	-	R	NO	OFF
M1022	Carry flag	○	○	○	○	OFF	-	-	R	NO	OFF
M1024	COM1 monitor request	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1025*	Indicate incorrect request for communication	○	○	○	○	OFF	-	-	R	NO	OFF
M1026	RAMP mode selection	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1027	PR output mode selection (8/16 bytes)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1028	Switch T64~T126 timer resulotion (10ms/100ms). ON =10ms	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1029*	CH0 (Y0, Y1) pulse output execution completed.	○	○	○	○	OFF	-	-	R	NO	OFF
M1030*	Pulse output Y1 execution completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1031*	Clear all non-latched memory	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1032*	Clear all latched memory	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1033*	Output state latched at STOP	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1034*	Disable all Y outputs	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1035*	Enable X7 input point as RUN/STOP switch	○	○	○	○	-	-	-	R/W	YES	OFF
M1037*	Enable 8-sets SPD function (Has to be used with D1037)	×	×	○	○	OFF	OFF	OFF	R/W	NO	OFF
M1038	Switch T200~T255 timer resulotion (10ms/1ms). ON = 1ms	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1039*	Fix scan time	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1040	Disable step transition	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1041	Step transition start	○	○	○	○	OFF	-	OFF	R/W	NO	OFF
M1042	Enable pulse operation	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1043	Zero return completed	○	○	○	○	OFF	-	OFF	R/W	NO	OFF
M1044	Zero point condition	○	○	○	○	OFF	-	OFF	R/W	NO	OFF
M1045	Disable “all output reset” function	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1046	Indicate STL status	○	○	○	○	OFF	-	-	R	NO	OFF
M1047	Enable STL monitoring	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1048	Indicate alarm status	○	○	○	○	OFF	-	-	R	NO	OFF
M1049	Enable alarm monitoring	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1050	Disable interruption I000 / I001	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1051	Disable interruption I100 / I101	○	○	○	○	OFF	-	-	R/W	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1052	Disable interruption I200 / I201	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1053	Disable interruption I300 / I301	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1054	Disable interruption I400 / I401	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1055	Disable interruption I500 / I501	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1056	Disable interruption I600~I699	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1057	Disable interruption I700~I799	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1058	COM3 monitor request	○	X	○	○	OFF	-	-	R/W	NO	OFF
M1059	Disable high-speed counter interruptions I010~I080	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1060	System error message 1	○	○	○	○	OFF	-	-	R	NO	OFF
M1061	System error message 2	○	○	○	○	OFF	-	-	R	NO	OFF
M1062	System error message 3	○	○	○	○	OFF	-	-	R	NO	OFF
M1063	System error message 4	○	○	○	○	OFF	-	-	R	NO	OFF
M1064	Incorrect use of operands	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1065	Syntax error	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1066	Loop error	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1067*	Program execution error	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1068*	Execution error locked (D1068)	○	○	○	○	OFF	-	-	R	NO	OFF
M1070	Switching clock pulse of Y1 for PWM instruction (ON: 100us; OFF: 1ms)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1071	Switching clock pulse of Y3 for PWM instruction (ON: 100us; OFF: 1ms)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1072	PLC status (RUN/STOP), ON = RUN	○	○	○	○	OFF	ON	OFF	R/W	NO	OFF
M1075	Error occurring when write in Flash ROM	○	○	○	○	OFF	-	-	R	NO	OFF
M1078	Y0/CH0(Y0, Y1) pulse output pause (immediate)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1079	Y1 pulse output pause (immediate)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1080	COM2 monitor request	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1081	Changing conversion mode for FLT instruction	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1083*	Selecting X6 pulse-width detecting mode. M1083 = ON, detecting pulse-width when X6 = ON; M1083 = OFF, detecting pulse-width when X6 = OFF.	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1084*	Enabling X6 Pulse width detecting function. (has to be used with M1183 and D1023)	○	○	○	○	OFF	OFF	OFF	R/W	NO	OFF
M1085	Selecting DVP-PCC01 duplicating function	○	○	○	○	OFF	-	-	R/W	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1086	Enabling password function for DVP-PCC01	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1088	Matrix comparison. Comparing between equivalent values (M1088 = ON) or different values (M1088 = OFF).	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1089	Indicating the end of matrix comparison. When the comparison reaches the last bit, M1089 = ON.	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1090	Indicating start of matrix comparison. When the comparison starts from the first bit, M1090 = ON.	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1091	Indicating matrix searching results. When the comparison has matched results, comparison will stop immediately and M1091 = ON.	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1092	Indicating pointer error. When the pointer Pr exceeds the comparison range, M1092 = ON	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1093	Matrix pointer increasing flag. Adding 1 to the current value of the Pr.	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1094	Matrix pointer clear flag. Clear the current value of the Pr to 0	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1095	Carry flag for matrix rotation / shift / output.	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1096	Borrow flag for matrix rotation/shift/input	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1097	Direction flag for matrix rotation/displacement	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1098	Counting the number of bits which are “1” or “0”	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1099	ON when the bits counting result is “0”	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1102*	Y2/CH1 (Y2, Y3) pulse output execution completed	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1103*	Y3 pulse output completed	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1104	Y2/CH1 (Y2, Y3) pulse output pause (immediate)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1105	Y3 pulse output pause (immediate)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1106	Zero point selection. M1106=ON, change the zero point to the right of DOG switch for zero return on CH0.	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1107	Zero point selection. M1107=ON, change the zero point to the right of DOG switch for zero return on CH1.	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1108	Y0/CH0 (Y0, Y1) pulse output pause (ramp down)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1109	Y1 pulse output pause (ramp down)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1110	Y2/CH1 (Y2, Y3) pulse output pause (ramp down)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1111	Y3 pulse output pause (ramp down)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1112	Switching clock pulse of Y0 for PWM instruction (ON: 100us; OFF: 1ms)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1113	Switching clock pulse of Y2 for PWM instruction (ON: 100us; OFF: 1ms)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1119*	Enable 2-speed output function of DDRVI instruction	○	╳	○	○	OFF	OFF	OFF	R/W	NO	OFF
M1120*	Retaining the communication setting of COM2 (RS-485), modifying D1120 will be invalid when M1120 is set.	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1121	For COM2(RS-485), data transmission ready	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1122	For COM2(RS-485), sending request	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1123	For COM2(RS-485), data receiving completed	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1124	For COM2(RS-485), data receiving ready	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1125	For COM2(RS-485), communication ready status reset	○	○	○	○	OFF	OFF	OFF	R/W	NO	OFF
M1126	For COM2(RS-485), set STX/ETX as user defined or system defined	○	○	○	○	OFF	OFF	OFF	R/W	NO	OFF
M1127	For COM2(RS-485), data sending / receiving / converting completed. (RS instruction is not supported)	○	○	○	○	OFF	OFF	OFF	R/W	NO	OFF
M1128	For COM2(RS-485), Transmitting/Receiving status Indication	○	○	○	○	OFF	OFF	OFF	R/W	NO	OFF
M1129	For COM2(RS-485), receiving time out	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1130	For COM2(RS-485), STX/ETX selection	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1131	For COM2(RS-485), ON when MODRD/RDST/MODRW data is being converted from ASCII to Hex	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1132	ON when there are no communication related instructions in the program	○	○	○	○	OFF	-	-	R	NO	OFF
M1136*	For COM3(RS-485/USB), retaining communication setting	○	╳	○	○	OFF	-	-	R/W	NO	OFF
M1137	Retain DNET mapping data during non-executing period	╳	╳	○	○	-	-	-	R/W	NO	OFF
M1138*	For COM1 (RS-232), retaining communication setting. Modifying D1036 will be invalid when M1138 is set.	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1139*	For COM1(RS-232), ASCII/RTU mode selection (OFF: ASCII; ON: RTU)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1140	For COM2 (RS-485), MODRD / MODWR / MODRW data receiving error	○	○	○	○	OFF	OFF	-	R	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1141	For COM2 (RS-485), MODRD / MODWR / MODRW parameter error	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1142	Data receiving error of VFD-A handy instructions	○	○	○	○	OFF	OFF	-	R	NO	OFF
M1143*	For COM2(RS-485), ASCII/RTU mode selection (OFF: ASCII; ON: RTU)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1156*	Enabling the mask and alignment mark function on I400/I401(X4) corresponding to Y0	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1158*	Enabling the mask and alignment mark function on I600/I601(X6) corresponding to Y2	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1161	8/16 bit mode (ON = 8 bit mode)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1162	Switching between decimal integer and binary floating point for SCLP instruction. ON: binary floating point; OFF: decimal integer	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1167	16-bit mode for HKY input	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1168	Designating work mode of SMOV	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1177	Enable the communication instruction for Delta VFD series inverter. ON: VFD-A (Default), OFF: other models of VFD	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1178	Enable knob VR0	╳	╳	○	○	OFF	-	-	R/W	NO	OFF
M1179	Enable knob VR1	╳	╳	○	○	OFF	-	-	R/W	NO	OFF
M1182	M1182 = ON, disable auto-mapping function when connected with left-side modules. ■ For SA2 /SX2 models, values of AIO modules will be auto-mapped to D9800 and above. ■ If the left side is connected with a communication module, additional 10 words will be occupied. Ex: 04AD-SL + EN01-SL + SA2, average value of Ch1~Ch4 of 04AD-SL maps to D9810~D9813.	╳	╳	○	○	OFF	-	-	R/W	NO	OFF
M1183	M1183 = ON, disable auto mapping function when connected with special modules #: ES2/EX2: OFF; SS2/SA2/SX2: ON (maps to D9900 and above)	○	○	○	○	#	-	-	R/W	NO	#
M1190	Set Y0 high speed output as 0.01 ~ 100Hz	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1191	Set Y1 high speed output as 0.01 ~ 100Hz	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1192	Set Y2 high speed output as 0.01 ~ 100Hz	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1193	Set Y3 high speed output as 0.01 ~ 100Hz	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1200	C200 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1201	C201 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1202	C202 counting mode ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1203	C203 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1204	C204 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1205	C205 counting mode (ON :count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1206	C206 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1207	C207 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1208	C208 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1209	C209 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1210	C210 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1211	C211 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1212	C212 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1213	C213 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1214	C214 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1215	C215 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1216	C216 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1217	C217 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1218	C218 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1219	C219 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1220	C220 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1221	C221 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1222	C222 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1223	C223 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1224	C224 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1225	C225 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1226	C226 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1227	C227 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1228	C228 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1229	C229 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1230	C230 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1231	C231 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1232	C232 counting mode (ON: count down)	×	○	×	×	OFF	-	-	R/W	NO	OFF
	C232 counter monitor (ON: count down)	○	×	○	○	OFF	-	-	R	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1233	C233 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1234	C234 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1235	C235 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1236	C236 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1237	C237 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1238	C238 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1239	C239 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1240	C240 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1241	C241 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1242	C242 counting mode (ON: count down)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1243	C243 Reset function control. ON = R function disabled	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1244	C244 Reset function control. ON = R function disabled	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1245	C245 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1246	C246 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1247	C247 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1248	C248 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1249	C249 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1250	C250 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1251	C251 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1252	C252 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1253	C253 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1254	C254 counter monitor (ON: count down)	○	○	○	○	OFF	-	-	R	NO	OFF
M1257	Set the ramp up/down of Y0, Y2 to be "S curve." ON = S curve.	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1260	Set up X7 as the reset signal for software counters C235 ~ C241	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1262	Enable cyclic output for table output function of DPTPO instruction. ON = enable.	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1270	C235 counting mode (ON: falling-edge count)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1271	C236 counting mode ON: falling-edge count)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1272	C237 counting mode (ON: falling-edge count)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1273	C238 counting mode (ON: falling-edge count)	○	○	○	○	OFF	-	-	R/W	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1274	C239 counting mode (ON: falling-edge count)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1275	C240 counting mode (ON: falling-edge count)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1276	C241 counting mode (ON: falling-edge count)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1277	C242 counting mode (ON: falling-edge count)	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1280*	For I000 / I001, reverse interrupt trigger pulse direction (Rising/Falling)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1284*	For I400 / I401, reverse interrupt trigger pulse direction (Rising/Falling)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1286*	For I600 / I601, reverse interrupt trigger pulse direction (Rising/Falling)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1303	High / low bits exchange for XCH instruction	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1304*	Enable force-ON/OFF of input point X	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1305	Reverse Y1 pulse output direction in high speed pulse output instructions	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1306	Reverse Y3 pulse output direction in high speed pulse output instructions	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1307	For ZRN instruction, enable left limit switch	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1308*	Output specified pulses or seek Z phase signal when zero point is achieved.	○	○	○	○	OFF	OFF	OFF	R/W	NO	OFF
M1312	For COM1(RS-232), sending request (Only applicable for MODRW and RS instruction)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1313	For COM1(RS-232), ready for data receiving (Only applicable for MODRW and RS instruction)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1314	For COM1(RS-232), data receiving completed (Only applicable for MODRW and RS instruction)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1315	For COM1(RS-232), data receiving error (Only applicable for MODRW and RS instruction)	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1316	For COM3(RS-485), sending request (Only applicable for MODRW and RS instruction)	○	×	○	×	OFF	OFF	-	R/W	NO	OFF
M1317	For COM3(RS-485), ready for data receiving (Only applicable for MODRW and RS instruction)	○	×	○	×	OFF	OFF	-	R/W	NO	OFF
M1318	For COM3(RS-485), data receiving completed (Only applicable for MODRW and RS instruction)	○	×	○	×	OFF	OFF	-	R/W	NO	OFF
M1319	For COM3(RS-485), data receiving error (Only applicable for MODRW and RS instruction)	○	×	○	×	OFF	OFF	-	R/W	NO	OFF
M1320*	For COM3 (RS-485), ASCII/RTU mode selection. (OFF: ASCII; ON: RTU)	○	×	○	×	OFF	-	-	R/W	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1346*	Output clear signals when ZRN is completed	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1347	Auto-reset Y0 when high speed pulse output is completed	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1348	Auto-reset Y1 when high speed pulse output is completed	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1350*	Enable PLC LINK	○	○	○	○	Off	-	OFF	R/W	NO	OFF
M1351*	Enable auto mode on PLC LINK	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1352*	Enable manual mode on PLC LINK	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1353*	Enable access up to 50 words through PLC LINK	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1354*	Enable simultaneous data read/write in a polling of PLC LINK	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1355*	Select Slave linking mode in PLC LINK (ON: manual; OFF: auto-detection)	○	○	○	○	-	-	-	R/W	YES	OFF
M1356*	Enable station number selection function. When both M1353 and M1356 are ON, the user can specify the station number in D1900~D1931	○	╳	○	○	-	-	-	R/W	YES	OFF
M1360*	Slave ID#1 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1361*	Slave ID#2 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1362*	Slave ID#3 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1363*	Slave ID#4 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1364*	Slave ID#5 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1365*	Slave ID#6 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1366*	Slave ID#7 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1367*	Slave ID#8 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1368*	Slave ID#9 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1369*	Slave ID#10 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1370*	Slave ID#11 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1371*	Slave ID#12 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1372*	Slave ID#13 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1373*	Slave ID#14 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1374*	Slave ID#15 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1375*	Slave ID#16 status on PLC LINK network	○	○	○	○	-	-	-	R/W	YES	OFF
M1376*	Indicate Slave ID#1 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1377*	Indicate Slave ID#2 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1378*	Indicate Slave ID#3 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1379*	Indicate Slave ID#4 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1380*	Indicate Slave ID#5 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1381*	Indicate Slave ID#6 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1382*	Indicate Slave ID#7 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1383*	Indicate Slave ID#8 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1384*	Indicate Slave ID#9 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1385*	Indicate Slave ID#10 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1386*	Indicate Slave ID#11 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1387*	Indicate Slave ID#12 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1388*	Indicate Slave ID#13 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1389*	Indicate Slave ID#14 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1390*	Indicate Slave ID#15 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1391*	Indicate Slave ID#16 data interchange status on PLC LINK	○	○	○	○	OFF	-	-	R	NO	OFF
M1392*	Slave ID#1 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1393*	Slave ID#2 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1394*	Slave ID#3 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1395*	Slave ID#4 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1396*	Slave ID#5 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1397*	Slave ID#6 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1398*	Slave ID#7 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1399*	Slave ID#8 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1400*	Slave ID#9 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1401*	Slave ID#10 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1402*	Slave ID#11 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1403*	Slave ID#12 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1404*	Slave ID#13 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1405*	Slave ID#14 linking error	○	○	○	○	OFF	-	-	R	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1406*	Slave ID#15 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1407*	Slave ID#16 linking error	○	○	○	○	OFF	-	-	R	NO	OFF
M1408*	Indicate that reading from Slave ID#1 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1409*	Indicate that reading from Slave ID#2 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1410*	Indicate that reading from Slave ID#3 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1411*	Indicate that reading from Slave ID#4 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1412*	Indicate that reading from Slave ID#5 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1413*	Indicate that reading from Slave ID#6 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1414*	Indicate that reading from Slave ID#7 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1415*	Indicate that reading from Slave ID#8 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1416*	Indicate that reading from Slave ID#9 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1417*	Indicate that reading from Slave ID#10 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1418*	Indicate that reading from Slave ID#11 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1419*	Indicate that reading from Slave ID#12 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1420*	Indicate that reading from Slave ID#13 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1421*	Indicate that reading from Slave ID#14 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1422*	Indicate that reading from Slave ID#15 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1423*	Indicate that reading from Slave ID#16 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1424*	Indicate that writing to Slave ID#1 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1425*	Indicate that writing to Slave ID#2 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1426*	Indicate that writing to Slave ID#3 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1427*	Indicate that writing to Slave ID#4 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1428*	Indicate that writing to Slave ID#5 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1429*	Indicate that writing to Slave ID#6 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1430*	Indicate that writing to Slave ID#7 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1431*	Indicate that writing to Slave ID#8 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1432*	Indicate that writing to Slave ID#9 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1433*	Indicate that writing to Slave ID#10 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1434*	Indicate that writing to Slave ID#11 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1435*	Indicate that writing to Slave ID#12 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1436*	Indicate that writing to Slave ID#13 is completed	○	○	○	○	OFF	-	-	R	NO	OFF

Special M	Function	ES2 EX2	SS2	SA2	SX2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
M1437*	Indicate that writing to Slave ID#14 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1438*	Indicate that writing to Slave ID#15 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1439*	Indicate that writing to Slave ID#16 is completed	○	○	○	○	OFF	-	-	R	NO	OFF
M1524	Auto-reset Y2 when high speed pulse output is completed	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1525	Auto-reset Y3 when high speed pulse output is completed	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1534	Enable ramp-down time setting on Y0. Has to be used with D1348.	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1535	Enable ramp-down time setting on Y2. Has to be used with D1349.	○	○	○	○	OFF	-	-	R/W	NO	OFF
M1538	Indicate pause status of Y0	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1539	Indicate pause status of Y1	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1540	Indicate pause status of Y2	○	○	○	○	OFF	OFF	-	R/W	NO	OFF
M1541	Indicate pause status of Y3	○	○	○	○	OFF	OFF	-	R/W	NO	OFF

2.9 S Relay

Initial step relay	Starting instruction in Sequential Function Chart (SFC). S0~S9, total 10 points.
Zero return step relay	Returns to zero point when using IST instruction in program. Zero return step relays not used for IST instruction can be used as general step relays. S10~S19, total 10 ponits.
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 sate before power loss. S20 ~ S127, total 108 points.
General purpose step relay	General relays in sequential function chart (SFC). They will be cleared when power loss after running. S128 ~ S911, total 784 points.
Alarm step relay	Used with alarm driving instruction API 46 ANS as an alarm contact for recording the alarm messages or eliminating external malfunctions. S912 ~ S1023, total 112 points.

2.10 T (Timer)

The units of the timer are 1ms, 10ms and 100ms and the counting method is counting up. When the present value in the timer equals the set value, the associated output coil will be ON. The set value should be a K value in decimal and can be specified by the content of data register D.

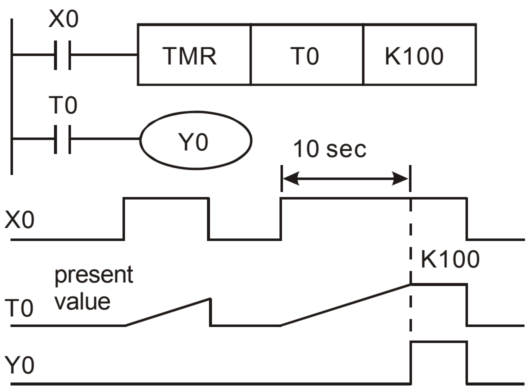
The actual set time in the timer = timer resolution× set value

Ex: If set value is K200 and timer resolution is 10ms, the actual set time in timer will be 10ms*200 = 2000ms = 2 sec.

General Timer

The timer executes once when the program reaches END instruction. When TMR instruction is executed, the timer coil will be ON when the current value reaches its preset value.

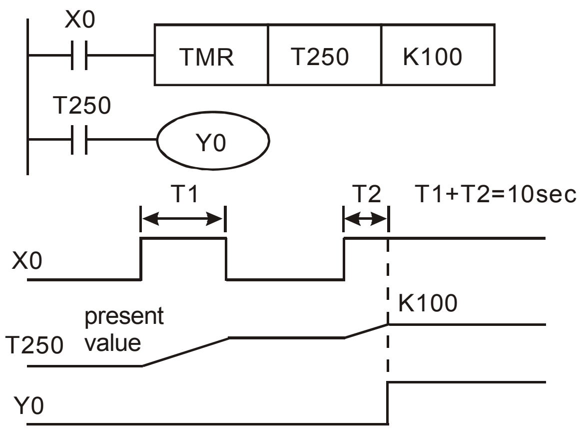
When X0 = ON, TMR instruction is driven. When current value achieves K100, the assocailte timer contact T0 is ON to drive Y0. If X0 = OFFor the power is off, the current value in T0 will be cleared as 0 and output Y0 driven by contact T0 will be OFF.



Accumulative Timer

The timer executes once when the program reaches END instruction. When TMR instruction is executed, the timer coil will be ON when the current value reaches its preset value. For accumulative timers, current value will not be cleared when timing is interrupted.

Timer T250 will be driven when X0 = ON. When X0 = OFFor the power is off, timer T250 will pause and retain the current value. When X0 is ON again, T250 resumes timing from where it was paused.



Timers for Subroutines and Interrupts

Timers for subroutines and interrupts count once when END instruction is met. The associated output coils will be ON if the set value is achieved when End instruction executes. T184~T199 are the only timers that can be used in subroutines or interrupts. Generals timers used in subroutines and interrupts will not work if the subroutines or interrupts are not executing.

2.11 C (Counter)

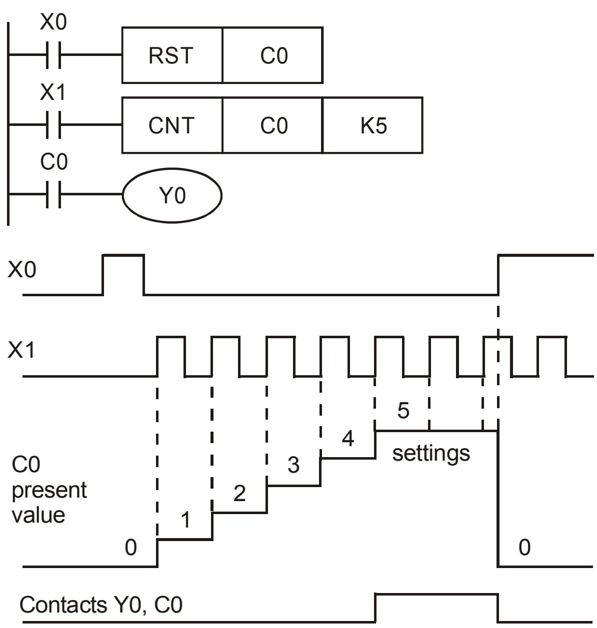
Counters will increment their present count value when input signals are triggered from OFF→ON.

	16 bits counters	32 bits counters		
Type	General	General	High speed	
Counters	C0~C199	C200~C231(C232)	C232(C233)~C242, C245~C254	C243, C244
Count direction	Count up	Count up/down		Count up
Range	0~32,767	-2,147,483,648~+2,147,483,647		0~2,147,483,647
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. The count value will become -2,147,483,648 if one more count is added to +2,147,483,647		Counter will keep on counting when preset value is reached. The count value will become 0 if one more count is added to +2,147,483,647

	16 bits counters	32 bits counters	
Output contact function	Ouptut Coil will be ON when counter reaches preset value.	Output coil is ON when counter reaches or is above preset value. Output coil is OFF when counter is below preset value.	Output coil is ON when counter reaches or is above preset value
High speed conparison	-	Associated devices are activated immediately when preset value is reached, i.e. independant of scan time.	-
Reset action	The present value will reset to 0 when RST instruction is executed, output coil will be OFF.		

Example:

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



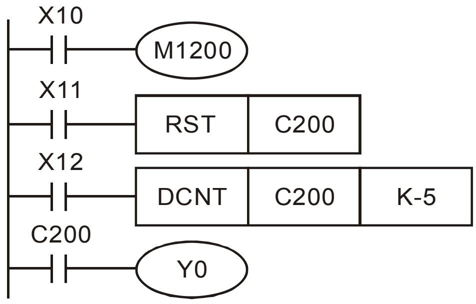
When X0 = ON, RST instruction resets C0. Every time When X1 is driven, C0 will count up (add 1).

When C0 reaches the preset value K5, output coil Y0 will be ON and C0 will stop counting and ignore the signals from input X1.

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

Example:

```
LD    X10
OUT   M1200
LD    X11
RST   C200
LD    X12
CNT   C200  K-5
LD    C200
OUT   Y0
```



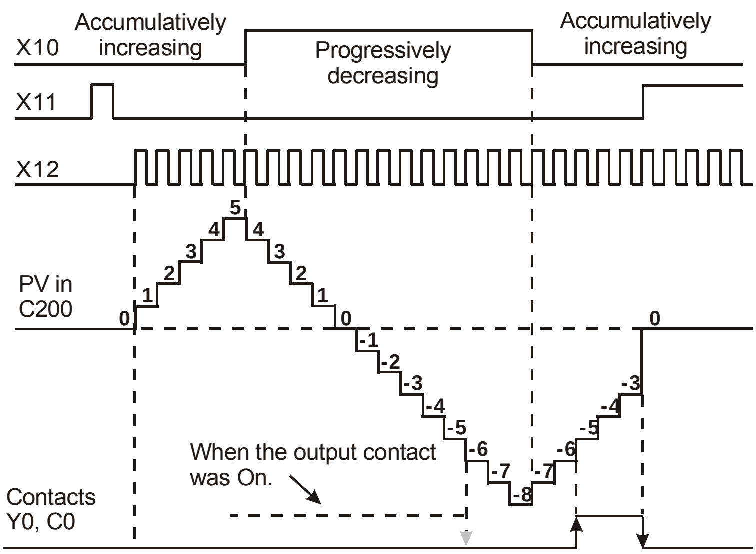
a) X10 drives M1200 to determine counting direction (up / down) of C200

b) When X11 goes from OFF to ON, RST instsruction will be executed and the PV (present value) in C200 will be cleared and contact C200 is OFF.

c) When X12 goes from Off to On, PV of C200 will count up (plus 1) or count down (minus 1).

d) When PV in C200 changes from K-6 to K-5, the contact C200 will be energized. When PV in C200 changes from K-5 to K-6, the contact of C200 will be reset.

e) If MOV instruction is applied through WPLSoft or HPP to designate a value bigger than SV to the PV register of C0, next time when X1 goes from OFF to ON, the contact C0 will be ON and PV of C0 will equal SV.



2.12 High-speed Counters

There are two types of high speed counters provided including Software High Speed Counter (SHSC) and Hardware High Speed Counter (HHSC). The same Input point (X) can be designated with only one high speed counter. Double designation on the same input or the same counter will result in syntax error when executing DCNT instruction.

Applicable Software High Speed Counters:

C / X	1-phase input								2 phase 2 input		
	C235	C236	C237	C238	C239	C240	C241	C242	C232	C233	C234
X0	U/D								A		
X1		U/D									
X2			U/D						B		
X3				U/D							
X4					U/D					A	
X5						U/D				B	
X6							U/D				A
X7								U/D			B
R/F	M1270	M1271	M1272	M1273	M1274	M1275	M1276	M1277	-	-	-
U/D	M1235	M1236	M1237	M1238	M1239	M1240	M1241	M1242	-	-	-

U: Count up D: Count down A: Phase A input B: Phase B input

Note:

1. U/D (Count up/Count down) can be specified by special M. OFF = count up; ON = count down.
2. R/F (Rising edge trigger/ Falling edge trigger) can also be specified by special M. OFF = Rising; ON = Falling.
3. SHSC supports max 10kHz input pulse on single point. Max 8 counters are applicable in the same time.
4. SS2 model does not support 2-phase 2-input conuting by (X0,X2) (C232).
5. For 2-phase 2-input conuting, (X4, X5) (C233) and (X6, X7) (C234), max 5kHz. (X0,X2) (C232), max 15kHz.
6. 2-phase 2-input counting supports double and quadruple frequency, which is selected in D1022 as the table in next page

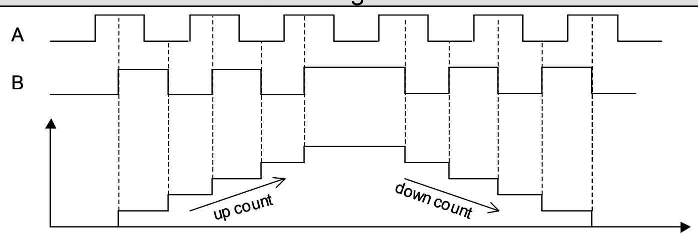
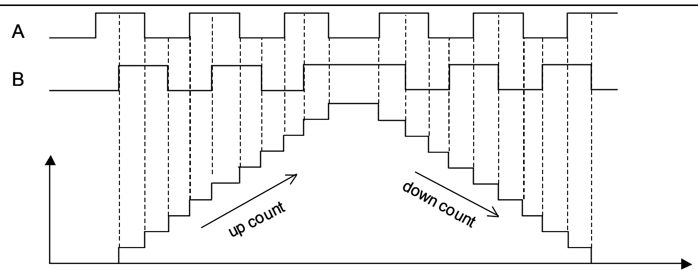
Applicable Hardware High Speed Counters:

C / X	1-phase input		1-phase 2-input						2-phase 2-input			
	C243	C244	C245	C246	C247	C248	C249	C250	C251	C252	C253	C254
X0	U		U/D	U/D	U	U			A	A		
X1	R		Dir	Dir	D	D			B	B		
X2		U					U/D	U/D			A	A
X3		R					Dir	Dir			B	B
X4				R		R				R		
X5								R				R

U: Count up A: Phase A input Dir: Directoin signal input
D: Count down B: Phase B input R: Reset signal input

Note:

1. The max frequency of the 1-phase input counters X0 (C243) and X2(C244) is 100kHz on ES2/EX2/SA2/SX2 model and 20kHz on SS2 model.
2. The max frequency of the 1-phase 2-input counters (X0, X1)(C245, C246) and (X2, X3)(C249, C250) is 100kHz on ES2/EX2/SA2/SX2 model and 20kHz on SS2 model.
3. The max frequency of the 1-phase 2-input counters (X0, X1)(C247, C248) is 10kHz on ES2/EX2/SS2/SX2 model and 100kHz on 32ES211T and SA2 model..
4. The max frequency of the 2-phase 2-input counter (X0, X1)(C251, C252) is 5kHz on ES2/EX2 model, 10kHz on SS2/SX2 model and 50kHz on 32ES211T and SA2 model.
5. The max frequency of the 2-phase 2-input counter (X2, X3)(C253, C254) is 5kHz on ES2/EX2/SA2 model, 10 kHz on SS2/SX2 model and 50kHz on 32ES211T.
6. 2-phase 2-input counting supports double and 4 times frequency, which is selected in D1022 as the table in next page. Please refer to the below table for detailed counting wave form.

D1022	Counting mode
K2 (Double Frequency)	
K4 or other value (Quadruple frequency) (Default)	

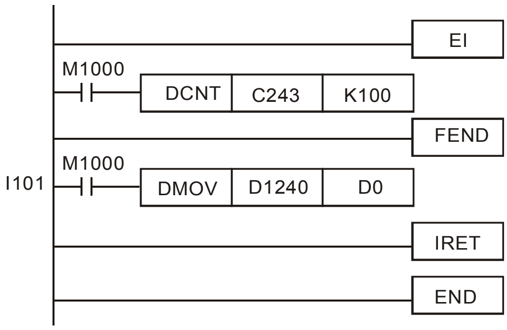
7. C243 and C244 support count-up mode only and occupy the associate input points X1 and X3 as reset ("R") function. If users do not need to apply reset function, set ON the associated special M relays (M1243 and M1244) to disable the reset function.
8. "Dir" refers to direction control function. OFF indicates counting up; ON indicates counting down.
9. When X1, X3, X4 and X5 is applied for reset function and associated external interrupts are disabled, users can define the reset function as Rising/Falling-edge triggered by special M relays

Reset Function	X1	X3	X4	X5
R/F	M1271	M1273	M1274	M1275

10. When X1, X3, X4 and X5 is applied for reset function and external interrupts are applied, the interrupt instructions have the priority in using the input points. In addition, PLC will move the current data in the counters to the associated data registers below then reset the counters.

Special D	D1241, D1240				D1243, D1242		
Counter	C243	C246	C248	C252	C244	C250	C254
External Interrupt	X1 (I100/I101)	X4(I400/I401)			X3 (I300/I301)	X5(I500/I501)	

Example:

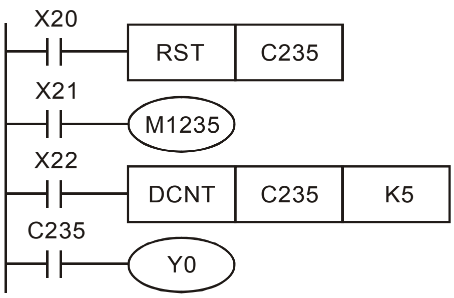


When C243 is counting and external interrupt is triggerred from X1(I101), counted value in C243 will be move to (D1241, D1240) immediately then C243 is reset. After this interrupt I101 executes.

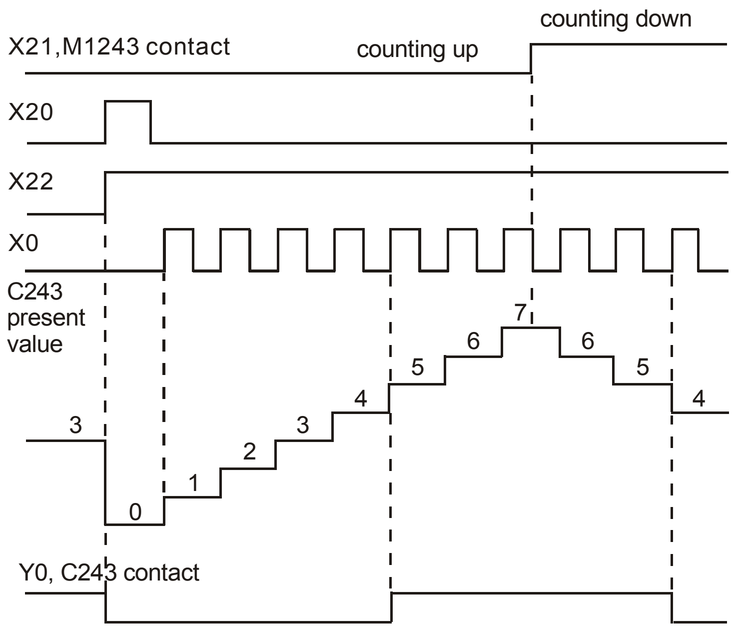
1-phase 1 input high-speed counter:

Example:

```
LD      X20
RST     C235
LD      X21
OUT     M1235
LD      X22
DCNT    C235  K5
LD      C235
OUT     Y0
```



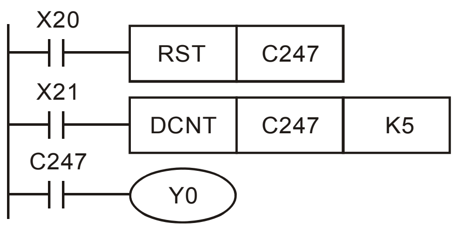
1. X21 drives M1235 to determine counting direction (Up/Down) of C235.
2. When X20 = ON, RST instsruction executes and the current value in C235 will be cleared. Contact C235 will be OFF
3. When X22 = ON, C235 receives signals from X0 and counter will count up (+1) or count down (-1).
4. When counter C235 reaches K5, contact C235 will be ON. If there is still input signal input for X0, it will keep on counting.



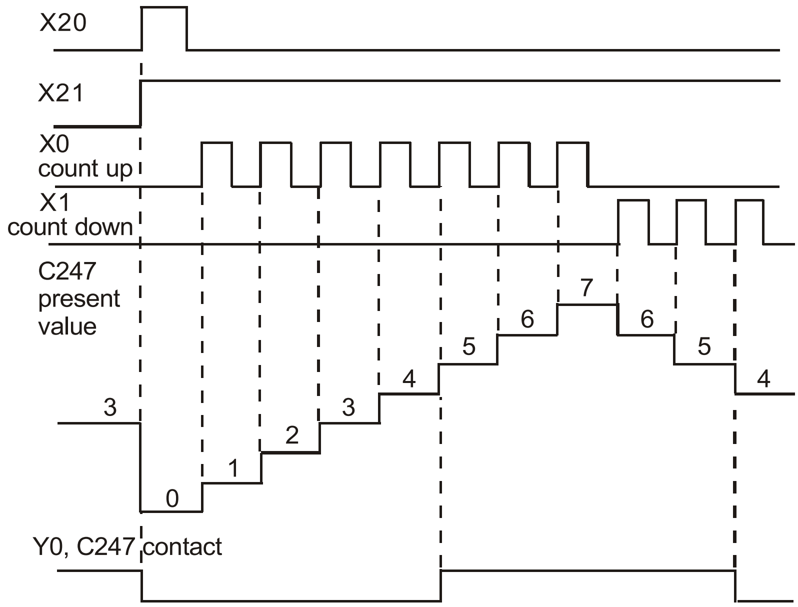
1-phase 2 inputs high-speed counter:

Example:

LD	X20
RST	C247
LD	X21
DCNT	C247 K5
LD	C247
OUT	Y0



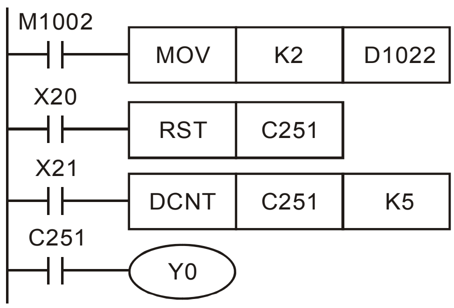
1. When X20 is ON, RST instsruction executes and the current value in C247 will be cleared. Contact C247 will be OFF.
2. When X21=ON, C247 receives count signals from X0 and counter counts up (+1), or C247 receives count signal from X1 and counter counts down (-1)
3. When C247 reaches K5, contact C247 will be ON. If there is still input signal from X0 or X1, C247 will keep on counting



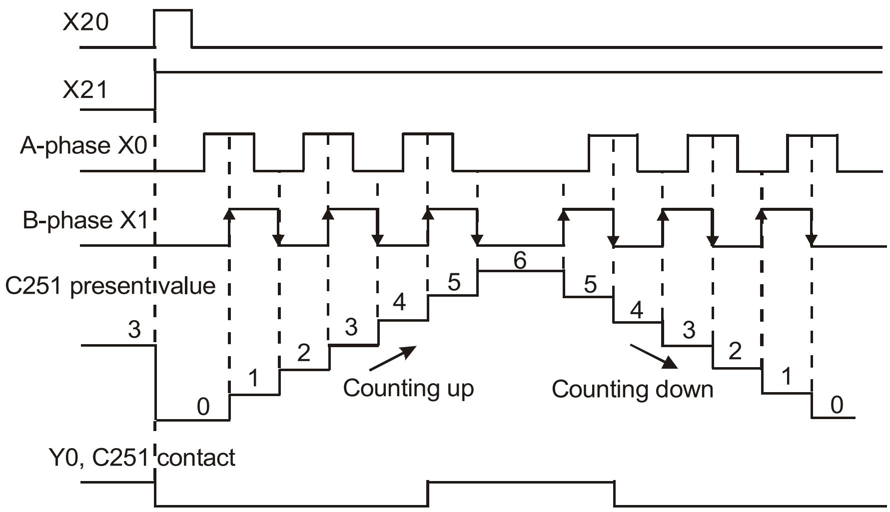
AB-phase input high-speed counter:

Example:

LD	M1002
MOV	K2 D1022
LD	X20
RST	C251
LD	X21
DCNT	C251 K5
LD	C251
OUT	Y0



1. When X20 is ON, RST instsruction executes and the current value in C251 will be cleared. Contact C251 will be OFF.
2. When X21 is ON, C251 receives A phase counting signal of X0 input terminal and B phase counting signal of X1 input terminal and executes count up or count down
3. When counter C251 reaches K5, contact C251 will be ON. If there is still input signal from X0 or X1, C251 will keep on counting
4. Counting mode can be specified as double frequency or 4-times frequency by D1022. Default: quadruple frequency.



2.13 Special Data Register

The types and functions of special registers (special D) are listed in the table below. Care should be taken that some registers of the same No. may bear different meanings in different series MPUs. Special M and special D marked with "*" will be further illustrated in 2.13. Columns marked with "R" refers to "read only", "R/W" refers to "read and write", "-" refers to the status remains unchanged and "#" refers to that system will set it up according to the status of the PLC. For detailed explanation please also refer to 2.13 in this chapter.

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1000*	SV of program scan ning WDT (Unit: 1ms)	○	○	○	○	200	-	-	R/W	NO	200
D1001	Displaying the firmware version of DVP-PLC (initial factory setting)	○	○	○	○	-	-	-	R	NO	#
D1002*	Program capacity	○	○	○	○	-	-	-	R	NO	#
D1003	Sum of program memory (sum of the PLC internal program memory.	○	○	○	○	#	-	-	R	YES	15872
D1004*	Syntax check error code	○	○	○	○	0	0	-	R	NO	0
D1008*	Step address when WDT is ON	○	○	○	○	0	-	-	R	NO	0
D1009	Number of LV (Low voltage) signal occurrence	○	○	○	○	-	-	-	R	YES	0
D1010*	Current scan time (Unit: 0.1ms)	○	○	○	○	#	#	#	R	NO	0
D1011*	Minimum scan time (Unit: 0.1ms)	○	○	○	○	#	#	#	R	NO	0
D1012*	Maximum scan time (Unit: 0.1ms)	○	○	○	○	#	#	#	R	NO	0
D1015*	Value of accumulative high-speed timer (0~32,767, unit: 0.1ms)	○	○	○	○	0	-	-	R/W	NO	0
D1018*	π PI (Low byte)	○	○	○	○	H' 0FDB	H' 0FDB	H' 0FDB	R/W	NO	H' 0FDB
D1019*	π PI(High byte)	○	○	○	○	H' 4049	H' 4049	H' 4049	R/W	NO	H' 4049
D1020*	X0~X7 input filter (unit: ms) 0~20ms adjustable	○	○	○	○	10	-	-	R/W	NO	10
D1022	Counting mode selection (Double frequency/ 4 times frequency) for AB phase counter (From X0, X1 input)	○	○	○	○	4	-	-	R/W	NO	4

2

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1023*	Register for Storing detected pulse width (unit: 0.1ms)	○	○	○	○	0	-	-	R/W	NO	0
D1025*	Code for communication request error	○	○	○	○	0	-	-	R	NO	0
D1026*	Pulse number for masking Y0 when M1156 = ON (Low word)	○	○	○	○	0	0	-	R/W	NO	0
D1027*	Pulse number for masking Y0 when M1156 = ON (High word)	○	○	○	○	0	0	-	R/W	NO	0
D1028	Index register E0	○	○	○	○	0	-	-	R/W	NO	0
D1029	Index register F0	○	○	○	○	0	-	-	R/W	NO	0
D1030	PV of Y0 pulse output (Low word)	○	○	○	○	-	-	-	R/W	YES	0
D1031	PV of Y0 pulse output (High word)	○	○	○	○	-	-	-	R/W	YES	0
D1032	PV of Y1 pulse output (Low word)	○	○	○	○	0	-	-	R/W	NO	0
D1033	PV of Y1 pulse output (High word)	○	○	○	○	0	-	-	R/W	NO	0
D1036*	COM1 (RS-232) communication protocol	○	○	○	○	H'86	-	-	R/W	NO	H'86
D1037*	Register for setting 8-sets SPD function (has to be used with M1037)	○	○	○	○	0	-	-	R/W	NO	0
D1038	1. Delay time setting for data response when PLC is SLAVE in COM2 / COM3 RS-485 communication. Range: 0 ~ 10,000 (unit: 0.1ms). 2. By using PLC LINK in COM2 (RS-485), D1038 can be set to send next communication data with delay. Range: 0 ~ 10,000 (Unit: one scan cycle)	○	○	○	○	-	-	-	R/W	NO	0
D1039*	Fixed scan time (ms)	○	○	○	○	0	-	-	R/W	NO	0
D1040	No. of the 1st step point which is ON.	○	○	○	○	0	-	-	R	NO	0
D1041	No. of the 2nd step point which is ON	○	○	○	○	0	-	-	R	NO	0
D1042	No. of the 3rd step point which is ON.	○	○	○	○	0	-	-	R	NO	0
D1043	No. of the 4th step point which is ON	○	○	○	○	0	-	-	R	NO	0
D1044	No. of the 5th step point which is ON.	○	○	○	○	0	-	-	R	NO	0
D1045	No. of the 6th step point which is ON	○	○	○	○	0	-	-	R	NO	0
D1046	No. of the 7th step point which is ON.	○	○	○	○	0	-	-	R	NO	0
D1047	No. of the 8th step point which is ON	○	○	○	○	0	-	-	R	NO	0
D1049	No. of alarm which is ON	○	○	○	○	0	-	-	R	NO	0
D1050 ↓ D1055	Converted data for Modbus communication data processing. PLC automatically converts the ASCII data in D1070~D1085 into Hex data and stores the 16-bit Hex data into D1050~D1055	○	○	○	○	0	-	-	R	NO	0
D1062*	Average times of analog input channels (CH0~CH3): 1~20. (For EX2/SX2)	○	×	×	○	2	-	-	R/W	YES	2
D1067*	Error code for program execution error	○	○	○	○	0	0	-	R	NO	0
D1068*	Address of program execution error	○	○	○	○	0	-	-	R	NO	0
D1070 ↓ D1085	Feedback data (ASCII) of Modbus communication. When PLC's RS-485 communication instruction receives feedback signals, the data will be saved in the registers D1070~D1085. Usres can check the received data in these registers.	○	○	○	○	0	-	-	R	NO	0

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1086	High word of the password in DVP-PCC01 (displayed in hex according to its ASCII codes)	○	○	○	○	0	-	-	R/W	NO	0
D1087	Low word of the password in DVP-PCC01 (displayed in hex according to its ASCII codes)	○	○	○	○	0	-	-	R/W	NO	0
D1089 ↓ D1099	Sent data of Modbus communication. When PLC's RS-485 communication instruction sends out data, the data will be stored in D1089~D1099. Users can check the sent data in these registers.	○	○	○	○	0	-	-	R	NO	0
D1109*	COM3 (RS-485) Communication protocol	○	×	○	○	H'86	-	-	R/W	NO	H'86
D1110*	Average value of EX2/SX2 analog input channel 0 (AD 0) When average times in D1062 is set to 1, D1110 indicates present value.	○	×	×	○	0	-	-	R	NO	0
D1111*	Average value of EX2/SX2 analog input channel 1 (AD 1) When average times in D1062 is set to 1, D1111 indicates present value	○	×	×	○	0	-	-	R	NO	0
D1112*	Average value of EX2/SX2 analog input channel 2 (AD 2) Whenaverage times in D1062 is set to 1, D1112 indicates present value	○	×	×	○	0	-	-	R	NO	0
D1113*	Average value of EX2/SX2 analog input channel 3 (AD 3) Whenaverage times in D1062 is set to 1, D1113 indicates present value	○	×	×	○	0	-	-	R	NO	0
D1114*	Enable/disable EX2/SX2 AD channels (0: enable (default) / 1: disable) bit0~bit3 sets AD0~AD3	○	×	×	○	0	-	-	R/W	YES	0
D1115*	EX2/SX2 analog mode selection (0: Voltage / 1: Current) bit0~bit3 sets AD0~AD3, bit4~bit5 sets DA0~DA1 bit8~bit13 : range of current bit8~bit11 sets AD0~AD3 (0: -20mA~20mA, 1: 4~20mA) Bit12~bit13 sets DA0~DA1 (0: 0~20mA, 1: 4~20mA)	○	×	×	○	0	0	0	R/W	YES	0
D1116*	Output value of analog output channel 0 (DA 0)	○	×	×	○	0	0	0	R/W	NO	0
D1117*	Output value of analog output channel 1 (DA 0)	○	×	×	○	0	0	0	R/W	NO	0
D1118*	EX2/SX2 sampling time of analog/digital converstion. Default: 2. Unit: 1ms. Sampling time will be regarded as 2ms if D1118≦2	○	×	×	○	2	-	-	R/W	YES	2
D1120*	COM2 (RS-485) communication protocol	○	○	○	○	H'86	-	-	R/W	NO	H'86
D1121*	COM1(RS-232) and COM2(RS-485) PLC communication address	○	○	○	○	-	-	-	R/W	Yes	1
D1122	COM2(RS-485) Residual number of words of transmitting data	○	○	○	○	0	0	-	R	NO	0
D1123	COM2(RS-485) Residual number of words of the receiving data	○	○	○	○	0	0	-	R	NO	0
D1124	COM2(RS-485) Definition of start character (STX)	○	○	○	○	H'3A	-	-	R/W	NO	H'3A
D1125	COM2(RS-485) Definition of first ending character (ETX1)	○	○	○	○	H'0D	-	-	R/W	NO	H'0D
D1126	COM2(RS-485) Definition of second ending character (ETX2)	○	○	○	○	H'0A	-	-	R/W	NO	H'0A

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch-ed	Default
D1127	Number of pulses for ramp-up operation of positioning instruction (Low word)	○	○	○	○	0	-	-	R/W	NO	0
D1128	Number of pulses for ramp-up operation of positioning instruction (High word)	○	○	○	○						
D1129	COM2 (RS-485) Communication time-out setting (ms)	○	○	○	○	0	-	-	R/W	NO	0
D1130	COM2 (RS-485) Error code returning from Modbus	○	○	○	○	0	-	-	R	NO	0
D1131	Input/output percentage value of CH0(Y0,Y1) close loop control	○	○	○	○	100	-	-	R/W	NO	100
D1132	Input/output percentage value of CH1(Y2,Y3) close loop control	○	○	○	○	100	-	-	R/W	NO	100
D1133	Number of pulses for ramp-down operation of positioning instruction (Low word)	○	○	○	○	0	-	-	R	NO	0
D1134	Number of pulses for ramp-down operation of positioning instruction (High word)	○	○	○	○	0	-	-	R	NO	0
D1135*	Pulse number for masking Y2 when M1158 = ON (Low word)	○	○	○	○	0	0	-	R/W	NO	0
D1136*	Pulse number for masking Y2 when M1158 = ON (High word)	○	○	○	○	0	0	-	R/W	NO	0
D1137*	Address where incorrect use of operand occurs	○	○	○	○	0	0	-	R	NO	0
D1140*	Number of I/O modules (max. 8)	○	○	○	○	0	-	-	R	NO	0
D1142*	Number of input points (X) on DIO modules	○	○	○	○	0	-	-	R	NO	0
D1143*	Number of output points (Y) on DIO modules	○	○	○	○	0	-	-	R	NO	0
D1145*	Number of the connected let-side modules	×	×	○	○	0	-	-	R	NO	0
D1167	The specific end word to be detected for RS instruction to execute an interruption request (I140) on COM1 (RS-232).	○	○	○	○	0	-	-	R/W	NO	0
D1168	The specific end word to be detected for RS instruction to execute an interruption request (I150) on COM2 (RS-485)	○	○	○	○	0	-	-	R/W	NO	0
D1169	The specific end word to be detected for RS instruction to execute an interruption request (I160) on COM3 (RS-485)	○	×	○	×	0	-	-	R/W	NO	0
D1178	VR0 value	×	×	○	○	0	-	-	R	NO	0
D1179	VR1 value	×	×	○	○	0	-	-	R	NO	0
D1182	Index register E1	○	○	○	○	0	-	-	R/W	NO	0
D1183	Index register F1	○	○	○	○	0	-	-	R/W	NO	0
D1184	Index register E2	○	○	○	○	0	-	-	R/W	NO	0
D1185	Index register F2	○	○	○	○	0	-	-	R/W	NO	0
D1186	Index register E3	○	○	○	○	0	-	-	R/W	NO	0
D1187	Index register F3	○	○	○	○	0	-	-	R/W	NO	0
D1188	Index register E4	○	○	○	○	0	-	-	R/W	NO	0
D1189	Index register F4	○	○	○	○	0	-	-	R/W	NO	0

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1190	Index register E5	○	○	○	○	0	-	-	R/W	NO	0
D1191	Index register F5	○	○	○	○	0	-	-	R/W	NO	0
D1192	Index register E6	○	○	○	○	0	-	-	R/W	NO	0
D1193	Index register F6	○	○	○	○	0	-	-	R/W	NO	0
D1194	Index register E7	○	○	○	○	0	-	-	R/W	NO	0
D1195	Index register F7	○	○	○	○	0	-	-	R/W	NO	0
D1220	Pulse output mode setting of CH0 (Y0, Y1)	○	○	○	○	0	-	-	R/W	NO	0
D1221	Pulse output mode setting of CH1 (Y2, Y3)	○	○	○	○	0	-	-	R/W	NO	0
D1232*	Number of output pulses for CH0 (Y0, Y1) ramp-down stop when mark sensor receives signals. (Low word).	○	○	○	○	0	0	--	R/W	NO	0
D1233*	Number of output pulses for CH0 (Y0, Y1) ramp-down stop when mark sensor receives signals. (High word).	○	○	○	○	0	0	--	R/W	NO	0
D1234*	Number of output pulses for CH1 (Y2, Y3) ramp-down stop when mark sensor receives signals. (Low word).	○	○	○	○	0	0	--	R/W	NO	0
D1235*	Number of output pulses for CH2 (Y2, Y3) ramp-down stop when mark sensor receives signals. (High word).	○	○	○	○	0	0	--	R/W	NO	0
D1240*	When interupt I400/I401/I100/I101 occurs, D1240 stores the low word of high-speed counter.	○	○	○	○	0	0	-	R	NO	0
D1241*	When interupt I400/I401/I100/I101 occurs, D1241 stores the high Word of high-speed counter.	○	○	○	○	0	0	-	R	NO	0
D1242*	When interupt I500/I501/I300/I301 occurs, D1242 stores the low Wordof high-speed counter.	○	○	○	○	0	0	-	R	NO	0
D1243*	When interupt I500/I501/I300/I301 occurs, D1243 stores the high Word of high-speed counter.	○	○	○	○	0	0	-	R	NO	0
D1244	Idle time (pulse number) setting of CH0 (Y0, Y1) The function is disabled if set value≦0.	○	○	○	○	0	-	-	R/W	NO	0
D1245	Idle time (pulse number) setting of CH1 (Y2, Y3) The function is disabled if set value≦0.	○	○	○	○	0	-	-	R/W	NO	0
D1249	Set value for COM1 (RS-232) data receiving time-out (Unit: 1ms, min. 50ms, value smaller than 50ms will be regarded as 50ms) (only applicable for MODRW/RS instruction) In RS instruction, no time-out setting if “0” is specified.	○	○	○	○	0	-	-	R/W	NO	0
D1250	COM1 (RS-232) communication error code (only applicable for MODRW/RS instruction)	○	○	○	○	0	-	-	R/W	NO	0
D1252	Set value for COM3 (RS-485) data receiving time-out (Unit: 1ms, min. 50ms, value smaller than 50ms will be regarded as 50ms) (only applicable for MODRW/RS instruction) In RS instruction, no time-out setting if “0” is specified	○	×	○	×	50	-	-	R/W	NO	50
D1253	COM3 (RS-485) communication error code (only applicable for MODRW/RS instruction)	○	×	○	×	0	-	-	R/W	NO	0
D1255*	COM3 (RS-485) PLC communication address	○	×	○	○	50	-	-	R/W	YES	1

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1256 ↓ D1295	For COM2 RS-485 MODRW instruction. D1256~D1295 store the sent data of MODRW instruction. When MODRW instruction sends out data, the data will be stored in D1256~D1295. Users can check the sent data in these registers.	○	○	○	○	0	-	-	R	NO	0
D1296 ↓ D1311	For COM2 RS-485 MODRW instruction. D1296~D1311 store the converted hex data from D1070 ~ D1085 (ASCII). PLC automatically converts the received ASCII data in D1070 ~ D1085 into hex data.	○	○	○	○	0	-	-	R	NO	0
D1312*	Specify the number of additional pulses for additional pulses output and Z-phase seeking function of ZRN instruction (Has to be used with M1308)	○	×	○	○	0	0	-	R/W	NO	0
D1313*	Second of RTC: 00 ~ 59	○	○	○	○	-	-	-	R/W	YES	0
D1314*	Minute of RTC: 00 ~ 59	○	○	○	○	-	-	-	R/W	YES	0
D1315*	Hour of RTC: 00 ~ 23	○	○	○	○	-	-	-	R/W	YES	0
D1316*	Day of RTC: 01 ~ 31	○	○	○	○	-	-	-	R/W	YES	1
D1317*	Month of RTC: 01 ~ 12	○	○	○	○	-	-	-	R/W	YES	1
D1318*	Week of RTC: 1 ~ 7	○	○	○	○	-	-	-	R/W	YES	2
D1319*	Year of RTC: 00 ~ 99 (A.D.)	○	○	○	○	-	-	-	R/W	YES	8
D1320*	ID of the 1st right side module	○	×	×	×	0	-	-	R	NO	0
D1321*	ID of the 2nd right side module	○	×	×	×	0	-	-	R	NO	0
D1322*	ID of the 3rd right side module	○	×	×	×	0	-	-	R	NO	0
D1323*	ID of the 4th right side module	○	×	×	×	0	-	-	R	NO	0
D1324*	ID of the 5th right side module	○	×	×	×	0	-	-	R	NO	0
D1325*	ID of the 6th right side module	○	×	×	×	0	-	-	R	NO	0
D1326*	ID of the 7th right side module	○	×	×	×	0	-	-	R	NO	0
D1327*	ID of the 8th right side module	○	×	×	×	0	-	-	R	NO	0
D1336	PV of Y2 pulse output (Low word)	○	○	○	○	-	-	-	R/W	YES	0
D1337	PV of Y2 pulse output (High word)	○	○	○	○	-	-	-	R/W	YES	0
D1338	PV of Y3 pulse output (Low word)	○	○	○	○	-	-	-	R/W	NO	0
D1339	PV of Y3 pulse output (High word)	○	○	○	○	-	-	-	R/W	NO	0
D1340	Start/end frequency of the 1st group pulse output CH0 (Y0, Y1)	○	○	○	○	100	-	-	R/W	NO	100
D1343	Ramp up/down time of the 1st group pulse output CH0 (Y0, Y1)	○	○	○	○	100	-	-	R/W	NO	100
D1348*	When M1534 = ON, D1348 stores the ramp-down time of CH0(Y0, Y1) pulse output.	○	○	○	○	100	-	-	R/W	NO	100
D1349*	When M1535 = ON, D1349 stores the ramp-down time of CH1(Y2, Y3) pulse output.	○	○	○	○	100	-	-	R/W	NO	100
D1352	Start/end frequency of the 2nd group pulse output CH1 (Y2, Y3)	○	○	○	○	100	-	-	R/W	NO	100

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1353	Ramp up/down time of the 2nd group pulse output CH1 (Y2, Y3)	○	○	○	○	100	-	-	R/W	NO	100
D1354	PLC Link scan cycle (Unit: 1ms) ■ Max: K32000 ■ D1354 = K0 when PLC Link stops or when the first scan is completed	○	○	○	○	0	0	0	R	NO	0
D1355*	Starting reference for Master to read from Slave ID#1	○	○	○	○	-	-	-	R/W	YES	H'1064
D1356*	Starting reference for Master to read from Slave ID#2	○	○	○	○	-	-	-	R/W	YES	H'1064
D1357*	Starting reference for Master to read from Slave ID#3	○	○	○	○	-	-	-	R/W	YES	H'1064
D1358*	Starting reference for Master to read from Slave ID#4	○	○	○	○	-	-	-	R/W	YES	H'1064
D1359*	Starting reference for Master to read from Slave ID#5	○	○	○	○	-	-	-	R/W	YES	H'1064
D1360*	Starting reference for Master to read from Slave ID#6	○	○	○	○	-	-	-	R/W	YES	H'1064
D1361*	Starting reference for Master to read from Slave ID#7	○	○	○	○	-	-	-	R/W	YES	H'1064
D1362*	Starting reference for Master to read from Slave ID#8	○	○	○	○	-	-	-	R/W	YES	H'1064
D1363*	Starting reference for Master to read from Slave ID#9	○	○	○	○	-	-	-	R/W	YES	H'1064
D1364*	Starting reference for Master to read from Slave ID#10	○	○	○	○	-	-	-	R/W	YES	H'1064
D1365*	Starting reference for Master to read from Slave ID#11	○	○	○	○	-	-	-	R/W	YES	H'1064
D1366*	Starting reference for Master to read from Slave ID#12	○	○	○	○	-	-	-	R/W	YES	H'1064
D1367*	Starting reference for Master to read from Slave ID#13	○	○	○	○	-	-	-	R/W	YES	H'1064
D1368*	Starting reference for Master to read from Slave ID#14	○	○	○	○	-	-	-	R/W	YES	H'1064
D1369*	Starting reference for Master to read from Slave ID#15	○	○	○	○	-	-	-	R/W	YES	H'1064
D1370*	Starting reference for Master to read from Slave ID#16	○	○	○	○	-	-	-	R/W	YES	H'1064
D1386	ID of the 1st left side module	╳	╳	○	○	0	-	-	R	NO	0
D1387	ID of the 2nd left side module	╳	╳	○	○	0	-	-	R	NO	0
D1388	ID of the 3rd left side module	╳	╳	○	○	0	-	-	R	NO	0
D1389	ID of the 4th left side module	╳	╳	○	○	0	-	-	R	NO	0
D1390	ID of the 5th left side module	╳	╳	○	○	0	-	-	R	NO	0
D1391	ID of the 6th left side module	╳	╳	○	○	0	-	-	R	NO	0
D1392	ID of the 7th left side module	╳	╳	○	○	0	-	-	R	NO	0
D1393	ID of the 8th rleft side module	╳	╳	○	○	0	-	-	R	NO	0
D1399*	Starting ID of Slave designated by PLC LINK	○	○	○	○	-	-	-	R/W	YES	1

2

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1415*	Starting reference for Master to write in Slave ID#1	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1416*	Starting reference for Master to write in Slave ID#2	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1417*	Starting reference for Master to write in Slave ID#3	○	○	○	○	-	-	-	R/W	YES	10C8
D1418*	Starting reference for Master to write in Slave ID#4	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1419*	Starting reference for Master to write in Slave ID#5	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1420*	Starting reference for Master to write in Slave ID#6	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1421*	Starting reference for Master to write in Slave ID#7	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1422*	Starting reference for Master to write in Slave ID#8	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1423*	Starting reference for Master to write in Slave ID#9	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1424*	Starting reference for Master to write in Slave ID#10	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1425*	Starting reference for Master to write in Slave ID#11	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1426*	Starting reference for Master to write in Slave ID#12	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1427*	Starting reference for Master to write in Slave ID#13	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1428*	Starting reference for Master to write in Slave ID#14	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1429*	Starting reference for Master to write in Slave ID#15	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1430*	Starting reference for Master to write in Slave ID#16	○	○	○	○	-	-	-	R/W	YES	H'10C8
D1431*	Times of PLC LINK polling cycle	○	○	○	○	0	-	-	R/W	NO	0
D1432*	Current times of PLC LINK polling cycle	○	○	○	○	0	-	-	R/W	NO	0
D1433*	Number of slave units linked to EASY PLC LINK	○	○	○	○	0	-	-	R/W	NO	0
D1434*	Data length to be read on Slave ID#1	○	○	○	○	-	-	-	R/W	YES	16
D1435*	Data length to be read on Slave ID#2	○	○	○	○	-	-	-	R/W	YES	16
D1436*	Data length to be read on Slave ID#3	○	○	○	○	-	-	-	R/W	YES	16
D1437*	Data length to be read on Slave ID#4	○	○	○	○	-	-	-	R/W	YES	16
D1438*	Data length to be read on Slave ID#5	○	○	○	○	-	-	-	R/W	YES	16
D1439*	Data length to be read on Slave ID#6	○	○	○	○	-	-	-	R/W	YES	16
D1440*	Data length to be read on Slave ID#7	○	○	○	○	-	-	-	R/W	YES	16
D1441*	Data length to be read on Slave ID#8	○	○	○	○	-	-	-	R/W	YES	16
D1442*	Data length to be read on Slave ID#9	○	○	○	○	-	-	-	R/W	YES	16
D1443*	Data length to be read on Slave ID#10	○	○	○	○	-	-	-	R/W	YES	16

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1444*	Data length to be read on Slave ID#11	○	○	○	○	-	-	-	R/W	YES	16
D1445*	Data length to be read on Slave ID#12	○	○	○	○	-	-	-	R/W	YES	16
D1446*	Data length to be read on Slave ID#13	○	○	○	○	-	-	-	R/W	YES	16
D1447*	Data length to be read on Slave ID#14	○	○	○	○	-	-	-	R/W	YES	16
D1448*	Data length to be read on Slave ID#15	○	○	○	○	-	-	-	R/W	YES	16
D1449*	Data length to be read on Slave ID#16	○	○	○	○	-	-	-	R/W	YES	16
D1450*	Data length to be written on Slave ID#1	○	○	○	○	-	-	-	R/W	YES	16
D1451*	Data length to be written on Slave ID#2	○	○	○	○	-	-	-	R/W	YES	16
D1452*	Data length to be written on Slave ID#3	○	○	○	○	-	-	-	R/W	YES	16
D1453*	Data length to be written on Slave ID#4	○	○	○	○	-	-	-	R/W	YES	16
D1454*	Data length to be written on Slave ID#5	○	○	○	○	-	-	-	R/W	YES	16
D1455*	Data length to be written on Slave ID#6	○	○	○	○	-	-	-	R/W	YES	16
D1456*	Data length to be written on Slave ID#7	○	○	○	○	-	-	-	R/W	YES	16
D1457*	Data length to be written on Slave ID#8	○	○	○	○	-	-	-	R/W	YES	16
D1458*	Data length to be written on Slave ID#9	○	○	○	○	-	-	-	R/W	YES	16
D1459*	Data length to be written on Slave ID#10	○	○	○	○	-	-	-	R/W	YES	16
D1460*	Data length to be written on Slave ID#11	○	○	○	○	-	-	-	R/W	YES	16
D1461*	Data length to be written on Slave ID#12	○	○	○	○	-	-	-	R/W	YES	16
D1462*	Data length to be written on Slave ID#13	○	○	○	○	-	-	-	R/W	YES	16
D1463*	Data length to be written on Slave ID#14	○	○	○	○	-	-	-	R/W	YES	16
D1464*	Data length to be written on Slave ID#15	○	○	○	○	-	-	-	R/W	YES	16
D1465*	Data length to be written on Slave ID#16	○	○	○	○	-	-	-	R/W	YES	16
D1480* ↓ D1495*	Data buffer to store the data read from Slave ID#1. PLC reads 16 data from the starting reference set in D1355. (Default of D1355: D100)	○	○	○	○	0	-	-	R	NO	0
D1496* ↓ D1511*	Data buffer to store the data to be written on Slave ID#1. PLC writes 16 data into the starting reference set in D1415. (Default of D1415: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1512* ↓ D1527*	Data buffer to store the data read from Slave ID#2 PLC reads 16 data from the starting reference set in D1356. (Default of D1356: D100)	○	○	○	○	0	-	-	R	NO	0
D1528* ↓ D1543*	Data buffer to store the data to be written on Slave ID#2. PLC writes 16 data into the starting reference set in D1416. (Default of D1416: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1544* ↓ D1559*	Data buffer to store the data read from Slave ID#3. PLC reads 16 data from the starting reference set in D1357. (Default of D1357: D100)	○	○	○	○	0	-	-	R	NO	0

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1560* ↓ D1575*	Data buffer to store the data to be written on Slave ID#3. PLC writes 16 data into the starting reference set in D1417. (Default of D1417: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1576* ↓ D1591*	Data buffer to store the data read from Slave ID#4. PLC reads 16 data from the starting reference set in D1358. (Default of D1358: D100)	○	○	○	○	0	-	-	R	NO	0
D1592* ↓ D1607*	Data buffer to store the data to be written on Slave ID#4. PLC writes 16 data into the starting reference set in D1418. (Default of D1418: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1608* ↓ D1623*	Data buffer to store the data read from Slave ID#5. PLC reads 16 data from the starting reference set in D1359. (Default of D1359: D100)	○	○	○	○	0	-	-	R	NO	0
D1624* ↓ D1639*	Data buffer to store the data to be written on Slave ID#5. PLC writes 16 data into the starting reference set in D1419. (Default of D1419: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1640* ↓ D1655*	Data buffer to store the data read from Slave ID#6. PLC reads 16 data from the starting reference set in D1360. (Default of D1360: D100)	○	○	○	○	0	-	-	R	NO	0
D1656* ↓ D1671*	Data buffer to store the data to be written on Slave ID#6. PLC writes 16 data into the starting reference set in D1420. (Default of D1420: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1672* ↓ D1687*	Data buffer to store the data read from Slave ID#7. PLC reads 16 data from the starting reference set in D1361. (Default of D1361: D100)	○	○	○	○	0	-	-	R	NO	0
D1688* ↓ D1703*	Data buffer to store the data to be written on Slave ID#7. PLC writes 16 data into the starting reference set in D1421. (Default of D1421: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1704* ↓ D1719*	Data buffer to store the data read from Slave ID#8. PLC reads 16 data from the starting reference set in D1362. (Default of D1362: D100)	○	○	○	○	0	-	-	R	NO	0
D1720* ↓ D1735*	Data buffer to store the data to be written on Slave ID#8. PLC writes 16 data into the starting reference set in D1422. (Default of D1422: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1736* ↓ D1751*	Data buffer to store the data read from Slave ID#9. PLC reads 16 data from the starting reference set in D1363. (Default of D1363: D100)	○	○	○	○	0	-	-	R	NO	0
D1752* ↓ D1767*	Data buffer to store the data to be written on Slave ID#9. PLC writes 16 data into the starting reference set in D1423. (Default of D1423: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1768* ↓ D1783*	Data buffer to store the data read from Slave ID#10. PLC reads 16 data from the starting reference set in D1364. (Default of D1364: D100)	○	○	○	○	0	-	-	R	NO	0

Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1784* ↓ D1799*	Data buffer to store the data to be written on Slave ID#10. PLC writes 16 data into the starting reference set in D1424. (Default of D1424: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1800* ↓ D1815*	Data buffer to store the data read from Slave ID#11. PLC reads 16 data from the starting reference set in D1365. (Default of D1365: D100)	○	○	○	○	0	-	-	R	NO	0
D1816* ↓ D1831*	Data buffer to store the data to be written on Slave ID#11. PLC writes 16 data into the starting reference set in D1425. (Default of D1425: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1832* ↓ D1847*	Data buffer to store the data read from Slave ID#12. PLC reads 16 data from the starting reference set in D1366. (Default of D1366: D100)	○	○	○	○	0	-	-	R	NO	0
D1848* ↓ D1863*	Data buffer to store the data to be written on Slave ID#12. PLC writes 16 data into the starting reference set in D1426. (Default of D1426: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1864* ↓ D1879*	Data buffer to store the data read from Slave ID#13. PLC reads 16 data from the starting reference set in D1367. (Default of D1367: D100)	○	○	○	○	0	-	-	R	NO	0
D1880* ↓ D1895*	Data buffer to store the data to be written on Slave ID#13. PLC writes 16 data into the starting reference set in D1427. (Default of D1427: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1896* ↓ D1911*	Data buffer to store the data read from Slave ID#14. PLC reads 16 data from the starting reference set in D1368. (Default of D1368: D100)	○	○	○	○	0	-	-	R	NO	0
D1900* ↓ D1931*	Specify the station number of Slaves for PLC-Link when M1356 is ON. Consecutive station numbers set by D1399 will be invalid in this case. Note that the registers are latched only when M1356 is ON.	○	×	○	○	0	-	-	R/W	NO	
D1912* ↓ D1927*	Data buffer to store the data to be written on Slave ID#14. PLC writes 16 data into the starting reference set in D1428. (Default of D1428: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1928* ↓ D1943*	Data buffer to store the data read from Slave ID#15. PLC reads 16 data from the starting reference set in D1369. (Default of D1369: D100)	○	○	○	○	0	-	-	R	NO	0
D1944* ↓ D1959*	Data buffer to store the data to be written on Slave ID#15. PLC writes 16 data into the starting reference set in D1429. (Default of D1429: D200)	○	○	○	○	0	-	-	R/W	NO	0
D1960* ↓ D1975*	Data buffer to store the data read from Slave ID#16. PLC reads 16 data from the starting reference set in D1370. (Default of D1370: D100)	○	○	○	○	0	-	-	R	NO	0
D1976* ↓ D1991*	Data buffer to store the data to be written on Slave ID#16. PLC writes 16 data into the starting reference set in D1430. (Default of D1430: D200)	○	○	○	○	0	-	-	R/W	NO	0

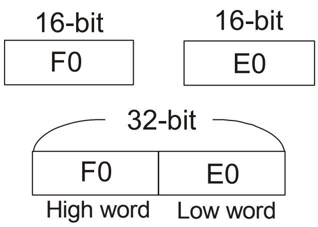
Special D	Content	ES2 EX2	SS 2	SA 2	SX 2	OFF ⇩ ON	STOP ⇩ RUN	RUN ⇩ STOP	Attrib.	Latch -ed	Default
D1994	Remaining times for PLC password setting on DVP-PCC01	○	○	○	○	0					
D1995	Data length for PLC ID Setting on DVP-PCC01	○	○	○	○	0	-	-	R/W	NO	0
D1996	1st Word of PLC ID Setting for DVP-PCC01 (Indicated by Hex format corresponding to ASCII codes)	○	○	○	○	0	-	-	R/W	NO	0
D1997	2nd Word of PLC ID Setting for DVP-PCC01 (Indicated by Hex format corresponding to ASCII codes)	○	○	○	○	0	-	-	R/W	NO	0
D1998	3rd Word of PLC ID Setting for DVP-PCC01 (Indicated by Hex format corresponding to ASCII codes)	○	○	○	○	0	-	-	R/W	NO	0
D1999	4th word of PLC ID Setting for DVP-PCC01 (Indicated by Hex format corresponding to ASCII codes)	○	○	○	○	0	-	-	R/W	NO	0
D9900~ D9999	For AIO modules only. (Please refer to **DVP-PLC Operation Manual – Modules** for more information)	○	×	×	×	-	-	-	R/W	NO	0

2.14 E, F Index Registers

Index registers are used as modifiers to indicate a specified device (word, double word) by defining an offset. Devices can be modified includes byte device (KnX, KnY, KnM, KnS, T, C, D) and bit device (X, Y, M, S). E, F registers cannot be used for modifying constant (K, H) Index registers not used as a modifier can be used as general purpose register.

Index register [E], [F]

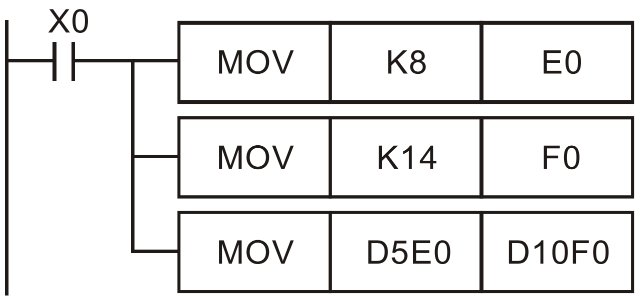
Index registers are 16-bit registers which can be read and written. There are 16 points indicated as E0~E7 and F0~F7. If you need a 32-bit register, you have to designate E. In this case, F will be covered up by E and cannot be used. It is recommended to use instruction DMOVP K0 E to reset E (including F) at power-on.



The combinations of E and F when designating a 32-bit register are:
(E0, F0) , (E1, F1) (E2, F2) (E3, F3) (E4, F4) , (E5, F5) (E6, F6) (E7, F7)

Example:

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.



2.15 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	P0~P255, 256 points	The location point of CJ, CALL

Pointer	I	For interrupt	External interrupt	I000/I001(X0), I100/I101(X1), I200/I201(X2), I300/I301(X3), I400/I401(X4), I500/I501(X5), I600/I601(X6), I700/I701(X7), 8 points (01, rising-edge trigger , 00, falling-edge trigger)	The location point of interrupt subroutine.
			Timer interrupt	I602/I699, I702/I799, 2 points (Timer resolution=1ms)	
			High-speed counter interrupt	I010, I020, I030, I040, I050, I060, I070, I080, 8 points	
			Communication interrupt	I140(COM1: RS232), I150(COM2: RS-485), I160(COM3: RS-485), 3 points	

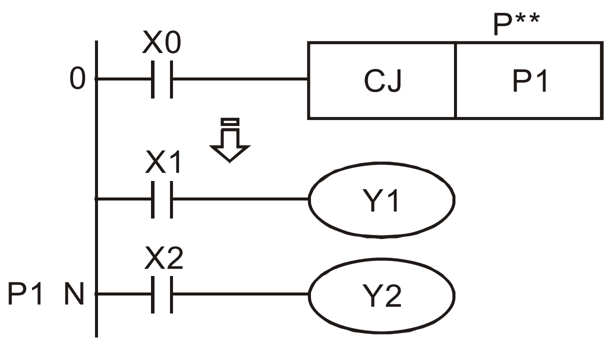
Nest Level Pointer N: used with instruction MC and MCR. MC is master start instruction. When the MC instruction is executed, the instructions between MC and MCR will be executed normally. MC-MCR master control instruction is nested level structure and max. 8 levels can be applicable, which is numbered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

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

<u>CJ condition jump:</u>

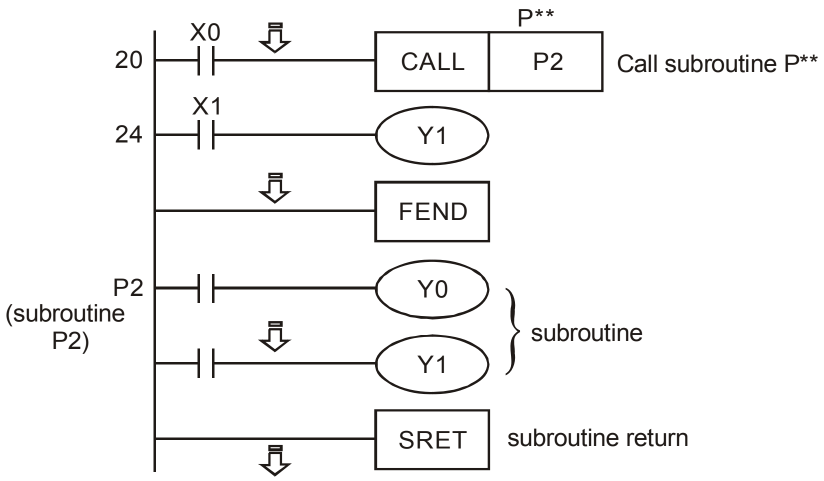
When X0 = ON, program will jump from address 0 to N (designated label P1) and keep on the execution. Instructions between 0 and N will be ignored.

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



<u>CALL subroutine, SRET subroutine END:</u>

When X0 is ON, program will jump to P2 to execute the designated subroutine. When SRET instruction is executed, it returns to address 24 to go on executing.



Interrupt pointer I: used with application instruction API 04 EI, API 05 DI, API 03 IRET. There are four types of interruption pointers. To insert an interruption, users need to combine EI (enable interruption), DI (disable interruption) and IRET (interruption return) instructions

1. External interrupt

- When input signal of input terminal X0~X7 is triggered on rising-edge or falling-edge, it will interrupt current program execution and jump to the designated interrupt subroutine pointer I000/I001(X0), I100/I101(X1), I200/I201(X2), I300/I301(X3), I400/I401(X4), I500/I501(X5), I600/I601(X6), I700/I701(X7). When IRET instruction is executed, program execution returns to the address before interrupt occurs.
- When X0 (C243) works with I100/I101 (X1), X0/X1 (C246, C248, C252) works with I400/I401, the value of C243, C246, C248, C252 will be stored in (D1240, D1241)
- When X2 (C244) works with I300/I301 (X3), X2/X3 (C250, C254) works with I500/I501, the value of C244, C250, C254 will be stored in (D1242, D1243).

2. Timer interrupt

PLC automatically interrupts the currently executed program every a fixed period of time (2ms~99ms) and jumps to the execution of a designated interruption subroutine

3. Counter interrupt

The high-speed counter comparison instruction API 53 DHSCS can designate that when the comparison reaches the target, the currently executed program will be interrupted and jump to the designated interruption subrountine executing the interruption pointers I010, I020, I030, I040, I050 ,I060, I070, I080..

4. Communication interrupt

I140:
Communication instruction RS (COM1 RS-232) can be designated to send interrupt request when specific charcters are received. Interrupt I140 and specific characters is set to low byte of D1167.

This function can be adopted when the PLC receives data of different length during the communication. Set up the specific end word in D1167 and write the interruption subroutine I140. When PLC receives the end word, the program will execute I140.

I150:

Communication instruction RS (COM2 RS-485) can be designated to send interrupt request when specific charcters are received. Interrupt I150 and specific characters is set to low byte of D1168. This function can be adopted when the PLC receives data of different length during the communication. Set up the specific end word in D1168 and write the interruption subroutine I150. When PLC receives the end word, the program will execute I150..

I160:

Communication instruction RS (COM3 RS-485) can be designated to send interrupt request when specific charcters are received. Interrupt I160 and specific characters is set to low byte of D1169 This function can be adopted when the PLC receives data of different length during the communication. Set up the specific end word in D1169 and write the interruption subroutine I160. When PLC receives the end word, the program will execute I160

2.16 Applications of Special M Relays and D Registers

Function Group PLC Operation Flag

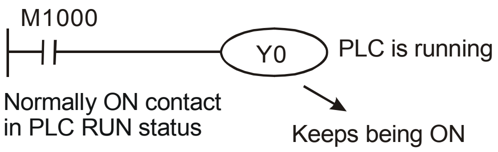
Number M1000~M1003

Contents:

These relays provide information of PLC operation in RUN status.

M1000:

NO contact for monitoring PLC status. M1000 remains “ON” when PLC is running.



M1001:

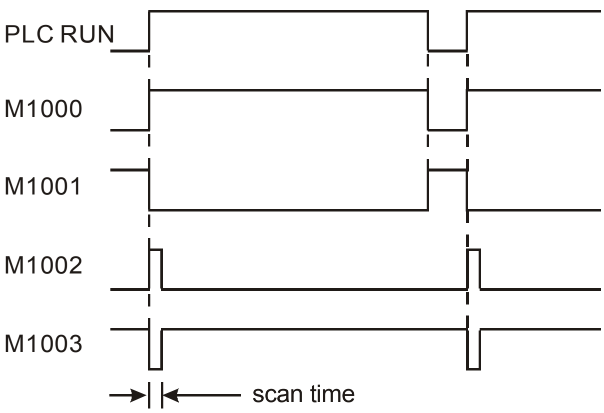
NC contact for monitoring PLC status. M1001 remains “OFF” when PLC is running.

M1002:

Enables single positive pulse for the first scan when PLC RUN is activated. Used to initialize registers, ouptuts, or counters when RUN is executed..

M1003:

Enables single negative pulse for the first scan when PLC RUN is activated. Used to initialize registers, ouptuts, or counters when RUN is executed.



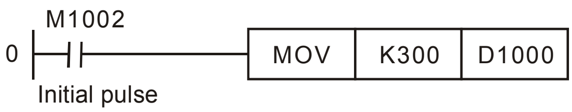
Function Group Monitor Timer

Number D1000

Contents:

1. Monitor timer is used for moitoring PLC scan time. When the scan time exceeds the set value (SV) in the monitor timer, the red ERROR LED will be ON and all outputs will be “OFF”.

2. The default in the monitor timer is 200ms. If the program is long or the operation is too complicated, MOV instruction can be used to modify SV. See the example below for SV = 300ms.



3. The maximum SV in the monitor timer is 32,767ms. However, care should be taken when adjusting SV. If SV in D1000 is too big, it cost much longer for operation errors to be detected. Therefore, SV is suggested to be shorter than 200ms.

4. Scan time could be prolonged due to complicated instruction operations or too many I/O modules being connected. Check D1010 ~ D1012 to see if the scan time exceeds the SV in D1000. Besides modifying the SV in D1000, users can also apply WDT instruction (API 07). When program execution progresses to WDT instruction, the internal monitor timer will be reset and therefore the scan time will not exceed the set value in the monitor timer.

Function Group Program Capacity

Number D1002

Contents:

This register holds the program capacity of the PLC.

SS2: 7,920 steps (Word)

ES2 / EX2 / SA2 / SX2 series: 15,872 steps (Word)

Function Group Syntax Check

Number M1004, D1004, D1137

Contents:

1. When errors occur in syntax check, ERROR LED indicator will flash and special relay M1004 = ON.
2. Timings for PLC syntax check:
   a) When the power goes from “OFF” to “ON”.
   b) When WPLSoft writes the program into PLC.
   c) When on-line editing is being conducted on WPLSoft.
3. Errors might result from parameter error or grammar error. The error code of the error will be placed in D1004. The address where the fault is located is saved in D1137. If the error belongs to loop error it may not have an address associated with it. In this case the value in D1137 is invalid.
4. For syntax error codes pease refer to section 6.2 Error Code table.

Function Group Watchdog Timer

Number M1008, D1008

Contents:

1. When the scan is time-out during execution, ERROR LED will be ON and M1008 = ON.

2. D1008 saves the STEP address where the timeout occurred

Function Group Scan Time Monitor

Number D1010~D1012

Contents:

The present value, minimum value and maximum value of scan time are stored in D1010 ~ D1012.

D1010: current scan time

D1011: minimum scan time

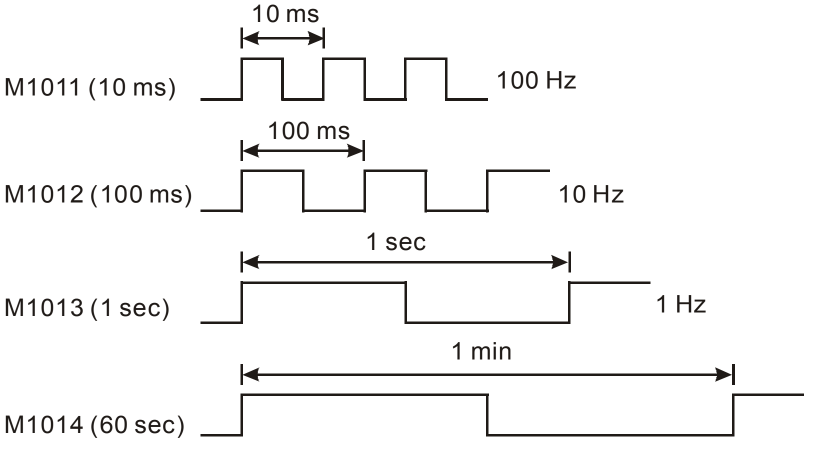
D1012: maximum scan time

Function Group Internal Clock Pulse

Number M1011~M1014

Contents:

1. PLC provides four different clock pulses to aid the application. When PLC is power-on, the four clock pulses will start automatically.



2. Clock pulse works even when PLC stops, i.e. activation of clock pulse is not synchronized with PLC RUN execution.

Function Group High-speed Timer

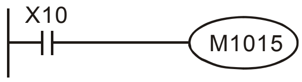
Number M1015, D1015

Contents:

1. When M1015 = ON, high-speed timer D1015 will be activated when the current scan proceeds to END instruction. The minimum resolution of D1015 is 100us.
2. The range of D1015 is 0~32,767. When it counts to 32,767, it will start from 0 again.
3. When M1015 = OFF, D1015 will stop timing immediately.

Example:

1. When X10 = ON, M1015 = ON to start high-speed timer and record the present value in D1015.
2. When X10 = OFF, M1015 = OFF. High-speed timer is disabled.



Function Group M1016~M1017, D1313~D1319

Number Real Time Clock

Contents:

1. Special M and special D relevant to RTC

Device	Name	Function
M1016	Year Display	OFF: display the last 2 digits of year in A.D ON: display the last 2 digits of year in A.D. plus 2,000
M1017	±30 seconds correction	When triggered from “Off” to “On”, the correction is enabled. 0 ~ 29 second: minute intact; second reset to 0 30~ 59 second: mimute + 1; second reset to 0
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 (last 2 digits of Year in A.D.)

2. If set value for RTC is invalid. RTC will display the time as Second→0, Minute→0, Hour→0, Day→1, Month→1, Week→1, Year→0.

3. Only when power is on can RTCs of SS2 series perform the fuction of timing. Memory of RTC is latched. RTC will resume the time when power is down. For higher accuracy of RTC, please conduction calibratoin on RTC when power resumes.

4. RTCs of SA2 V1.0 及 ES2/EX2/SX2 V2.0 series can still operate for one or two weeks after the power is off (they vary with the ambient temperature). Therefore, if the machine has not operated since one or two weeks ago, please reset RTC.

5. Methods of modifying RTC:
   a) Apply TWR instruction to modify the built-in real time clock. Please refer to TWR instruction for detail.
   b) Use peripheral devices or WPLSoft to set the RTC value.

Function Group π (PI)

Number D1018~D1019

Contents:

6. D1018 and D1019 are combined as 32-bit data register for storing the floating point value of π

7. Floating point value = H 40490FDB

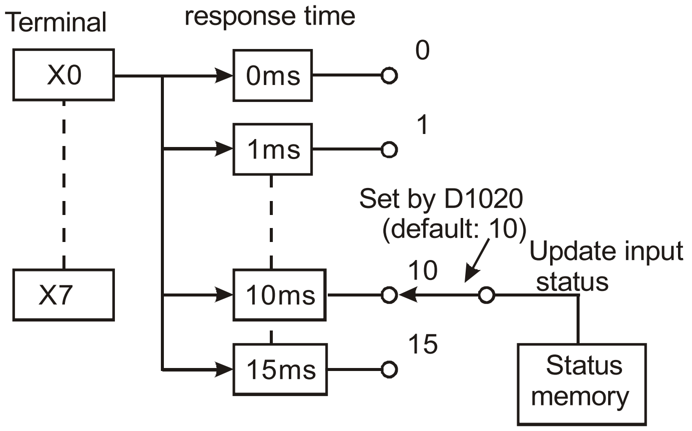
Function Group Adjustment on Input Terminal Response Time

Number D1020

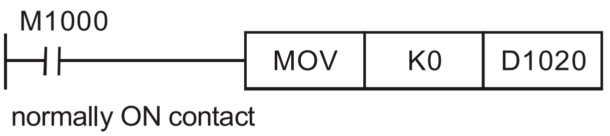
Contents:

1. D1020 can be used for setting up the response time of receiving pulses at X0 ~X7 for ES2 series MPU. Default: 10ms, 0~20ms adjustable.

2. When the power of PLC goes from "OFF" to "ON", the content of D1020 is set to 10 automatically.



3. If the following programs are executed, the response time of X0 ~ X7 will be set to 0ms. However, the fastest response time of input terminals will be 50μs due to that all terminals are connected with RC filters..



4. It is not necessary to adjust response time when using high-speed counters or interrupts

5. Using API 51 REFF instruction has the same effect as modifying D1020.

Function Group X6 pulse width detecting function

Number M1083,M1084, D1023

Contents:

When M1084 = ON, X6 pulse width detecting function is enabled and the detected pulse width is stored in D1023 (unit: 0.1ms)

M1083 On：detecting width of negative half cycle (OFF→ON)

M1083 Off：detecting width of positive half cycle (ON→OFF)

Function Group Communication Error Code

Number M1025, D1025

Contents:

In the connection between PLC and PC/HMI, M1025 will be ON when PLC receives illegal communication request during the data transmission process. The error code will be stored in D1025.

01: illegal instruction code
02: illegal device address.
03: requested data exceeds the range.
07: checksum error

Function Group Pulse output Mark and Mask function

Number M1108, M1110, M1156, M1158, M1538, M1540, D1026, D1027, D1135, D1136, D1232, D1233, D1234, D1235, D1348, D1349

Contents:

Please refer to explanations of API 59 PLSR / API 158 DDRVI / API 197 DCLLM instructions.

Function Group Execution Completed Flag

Number M1029, M1030, M1102, M1103

Contents:

Execution Completed Flag:

MTR, HKY, DSW, SEGL, PR:

M1029 = ON for a scan cycle whenever the above instructions complete the execution.

PLSY, PLSR:

1. M1029 = ON when Y0 pulse output completes.
2. M1030 = ON when Y1 pulse output completes
3. M1102 = ON when Y2 pulse output completes.
4. M1103 = ON when Y3 pulse output completes.
5. When PLSY, PLSR instruction are OFF, M1029, M1030, M1102, M1103 will be OFF as well. When pulse output instructions executes again, M1029, M1030, M1102, M1103 will be OFF and turn ON when execution completes.
6. Users have to clear M1029 and M1030 manually.

INCD:

M1029 will be “ON” for a scan period when the assigned groups of data comparison is completed

RAMP, SORT:

1. M1029= ON when instruction is completed. M1029 must be cleared by user manually.
2. If this instruction is OFF, M1029 will be OFF.

DABSR:

1. M1029= ON when instruction is completed.
2. When the instruction is re-executed for the next time, M1029 will turn off first then ON again when the instruction is completed

ZRN, DRVI, DRVA:

1. M1029 will be "ON" after Y0 and Y1 pulse output is completed. M1102 will be "ON" after Y2 and Y3 pulse output is compeleted.
2. When the instruction is re-executed for the next time, M1029 / M1102 will turn off first then ON again when the instruction is completed.

Function Group Clear Instruction

Number M1031, M1032

Contents:

M1031 (clear non-latched memory) , M1032 (clear latched memory)

Device	Devices will be cleared
M1031 Clear non-latched area	Contact status of Y, general-purpose M and general-purpose S ■ General-purpose contact and timing coil of T ■ General-purpose contact, counting coil reset coil of C ■ General-purpose present value register of D ■ General-purpose present value register of T ■ General-purpose present value register of C
M1032 Clear latched area	Contact status of M and S for latched ■ Contact and timing coil of accumulative timer T ■ Contact and timing coil of high-speed counter C for latched ■ Present value register of D for latched ■ Present value register of accumulative timer T ■ Present value register of high-speed counter C for latched

Function Group Output State Latched in STOP mode

Number M1033

Contents:

When M1033 = ON, PLC outputs will be latched when PLC is switched from RUN to STOP.

Function Group Disabling all Y outputs

Number M1034

Contents:

When M1034 = ON, all outputs will turn off.

Function Group RUN/STOP Switch

Number M1035

Contents:

When M1035 = ON, PLC uses input point X7 as the switch of RUN/STOP.

Function Group COM Port Function

Number

Item \ Port	COM1	COM2	COM3
Communication format	D1036	D1120	D1109
Communication setting holding	M1138	M1120	M1136
ASCII/RTU mode	M1139	M1143	M1320
Slave communication address	D1121		D1255

Contents:

COM ports (COM1: RS-232, COM2: RS-485, COM3: RS-485) support communication format of MODBUS ASCII/RTU modes. When RTU format is selected, the data length should be set as 8. COM2 and COM3 support transmission speed up to 921kbps. COM1, COM2 and COM3 can be used at the same time.

COM1:

Can be used in master or slave mode. Supports ASCII/RTU communication format, baudrate (115200bps max), and modification on data length (data bits, parity bits, stop bits). **D1036:** COM1 (RS-232) communication protocol of master/slave PLC. (b8 - b15 are not used) Please refer to table below for setting.

COM2:

Can be used in master or slave mode. Supports ASCII/RTU communication format, baudrate (921kbps max), and modification on data length (data bits, parity bits, stop bits). **D1120:** COM2 (RS-485) communication protocol of master/slave PLC. Please refer to table below for setting.

COM3:

Can be used in master or slave mode. Supports ASCII/RTU communication format, baudrate (921kbps max), and modification on data length (data bits, parity bits, stop bits). **D1109:** COM3 (RS-485) communication protocol of master/slave PLC. (b8 - b15 are not used) Please refer to table below for setting.

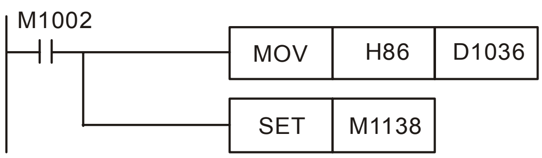
	Content	
b0	Data Length	0: 7 data bits, 1: 8 data bits (RTU supports 8 data bits only)
b1 b2	Parity bit	00: None 01: Odd 11: Even
b3	Stop bits	0: 1 bit, 1: 2bits
b4 b5 b6	Baud rate	0001(H1): 110 0010(H2): 150 0011(H3): 300

	Content		
b7		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
		1101(HD):	500000 (COM2 / COM3)
		1110(HE):	31250 (COM2 / COM3)
		1111(HF):	921000 (COM2 / COM3)
b8	Select start bit	0: None	1: D1124
b9	Select the 1st end bit	0: None	1: D1125
b10	Select the 2nd end bit	0: None	1: D1126
b11~b15	Undefined		

Example 1: Modifying COM1 communication format

1. Add the below instructions on top of the program to modify the communication format of COM1. When PLC switches from STOP to RUN, the program will detect whether M1138 is ON in the first scan. If M1138 is ON, the program will modify the communication settings of COM1 according to the value set in D1036

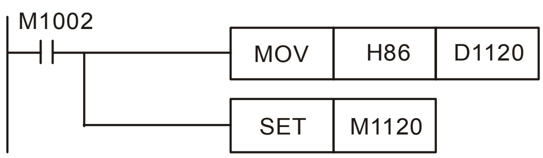
2. Modify COM1 communication format to ASCII mode, 9600bps, 7 data bits, even parity, 1 stop bits (9600, 7, E, 1).



Example 2: Modiying COM2 communication format

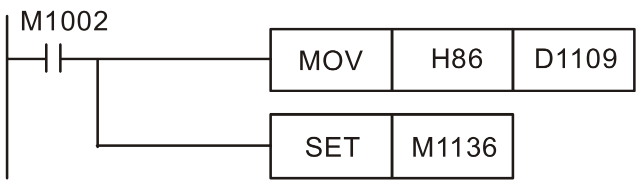
1. Add the below instructions on top of the program to modify the communication format of COM2. When PLC switches from STOP to RUN, the program will detect whether M1120 is ON in the first scan. If M1120 is ON, the program will modify the communication settings of COM2 according to the value set in D1120

2. Modify COM2 communication format to ASCII mode, 9600bps, 7 data bits, even parity, 1 stop bits (9600, 7, E, 1)



Example 3: Modifying COM3 communication format

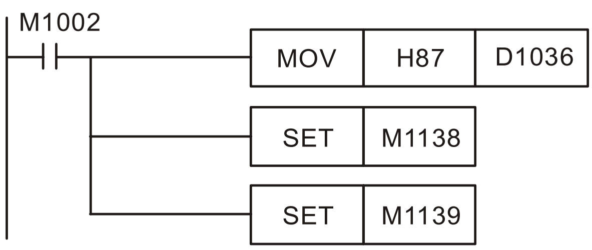
1. Add the below instructions on top of the program to modify the communication format of COM3. When PLC switches from STOP to RUN, the program will detect whether M1136 is ON in the first scan. If M1136 is ON, the program will modify the communication settings of COM3 according to the value set in D1109
2. Modify COM3 communication format to ASCII mode, 9600bps, 7 data bits, even parity, 1 stop bits (9600, 7, E, 1).



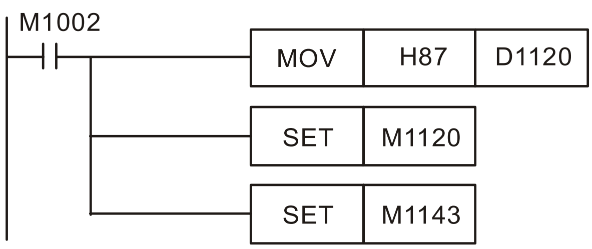
Example 4: RTU mode setting of COM1、COM2、COM3

1. COM1, COM2 and COM3 support ASCII/RTU mode. COM1 is set by M1139, COM2 is set by M1143 and COM3 is set by M1320. Set the flags ON to enable RTU mode or OFF to enable ASCII mode.
2. Modify COM1/COM2/COM3 communication format to RTU mode, 9600bps, 8 data bits, even parity, 1 stop bits (9600, 8, E, 1).

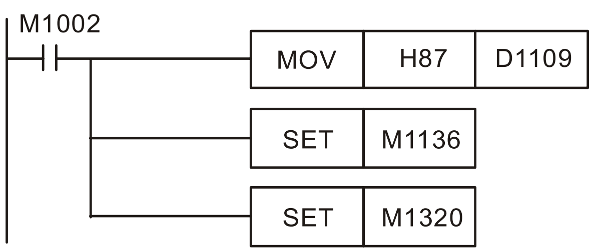
COM1:



COM2:



COM3:



Note:

1. The modified communication format will not be changed when PLC state turns from RUN to STOP.
2. If the PLC is powered OFF then ON again in STOP status, the modified communication format on COM1~COM3 will be reset to default communication format (9600, 7, E, 1).

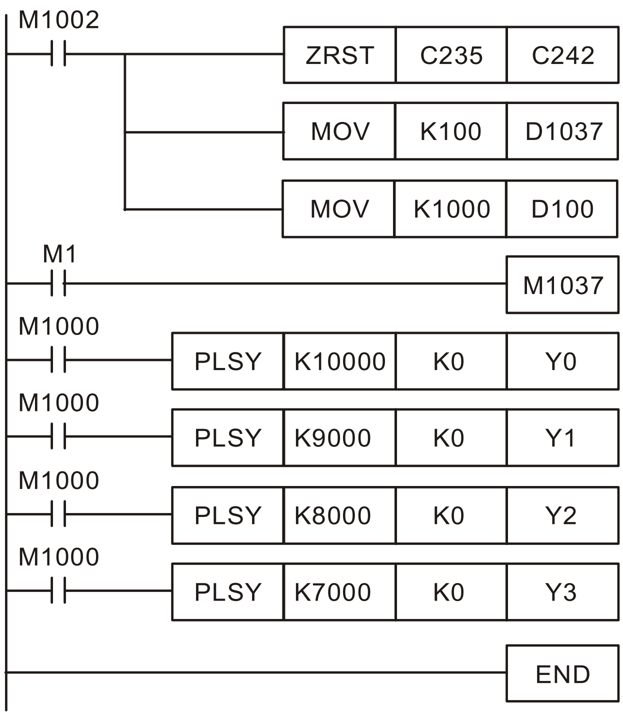
Function Group Enable SPD function

Number M1037, D1037

Contents:

1. M1037 and D1037 can be used to enable 8 sets of SPD instructions. When M1037 is ON, 8 sets of SPD instructions will be enabled. When M1037 is OFF, the function will be disabled.
2. The detected speed will be stored in the registers designated by D1037, e.g. if D1037 = K100, the user has to set up the value in D100, indicating the interval for capturing the speed value (unit: ms). In addition, the captured speed value will be stored in D101 ~ D108 in order.

   ※ When the function is enabled, C235~C242 will be occupied and unavailable in PLC execution process program.



Function Group Communication Response Delay

Number D1038

Contents:

1. Data response delay time can be set when PLC is a Slave in COM2, COM3 RS-485 communication. Unit: 0.1ms. 0~10,000 adjustable.

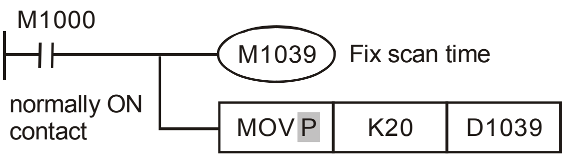
2. By using PLC-Link, D1038 can be set to send next communication data with delay. Unit: 1 scan cycle. 0~10,000 adjustable

Function Group Fixed scan time

Number M1039, D1039

Contents:

1. When M1039 is ON, program scan time is determined by D1039. When program execution is completed, next scan will be activated only when the fixed scan time is reached. If D1039 is less than actual scan time, it will scan by the actual program scan time.



Scan time is fixed to 20ms

2. Instructions related to scan time, RAMP, HKY, SEGL, ARWS and PR should be used with “fixed scan time” or “timed interrupt”.

3. Particularly for instruction HKY, which is applied for 16-keys input operated by 4x4 matrix, scan time should be set to 20ms or above.

4. Scan time displayed in D1010~D1012 also includes fixed scan time.

Function Group Analog Function

Number D1062, D1110~D1113, D1116~D1118

Contents:

1. The function is for EX2/SX2 Only

2. Resolution of AD (analog input) channels: 12 bits.

   Voltage: -10V~10V ⇔ Value: -2000~2000.

   Current: -20mA~20mA⇔ Value: -2000~2000

   Current: 4mA~20mA ⇔ Value: 0~2000

3. Resolution of DA (analog output) channels: 12 bits

   Voltage: -10V~10V ⇔ Value: -2000~2000

   Current: 0~20mA ⇔ Value: 0~4000

   Current: 4mA~20mA ⇔ Value: 0~2000

4. D1118: EX2/SX2 sampling time of analog/digital converstion. Default: 2. Unit: 1ms. If D1118 ≤ 2, it will be regarded as 2ms.

5. Default of average times in analog input channels: (K2). If set value = K1, PLC takes the present value.

Device	Function
D1062	Average times of EX2/SX2 analog input channels (CH0~CH3): 1~20, Default = K2
D1110	Average value of EX2/SX2 analog input channel 0 (AD 0)
D1111	Average value of EX2/SX2 analog input channel 1 (AD 1)
D1112	Average value of EX2/SX2 analog input channel 2 (AD 2)
D1113	Average value of EX2/SX2 analog input channel 3 (AD 3)
D1115	EX2/SX2 analog mode selection (0: Voltage / 1: Current) bit0~bit3 sets AD0~AD3, bit4~bit5 sets DA0~DA1 bit8~bit13 : range of current bit8~bit11 sets AD0~AD3 (0: -20mA~20mA, 1: 4~20mA) Bit12~bit13 sets DA0~DA1 (0: 0~20mA, 1: 4~20mA)
D1116	Output value of analog output channel 0 (DA 0)
D1117	Output value of analog output channel 1 (DA 1)
D1118	For EX2/SX2 series, sampling time of analog/digital conversion. Sampling time will be regarded as 2ms If D1118≦2.

2

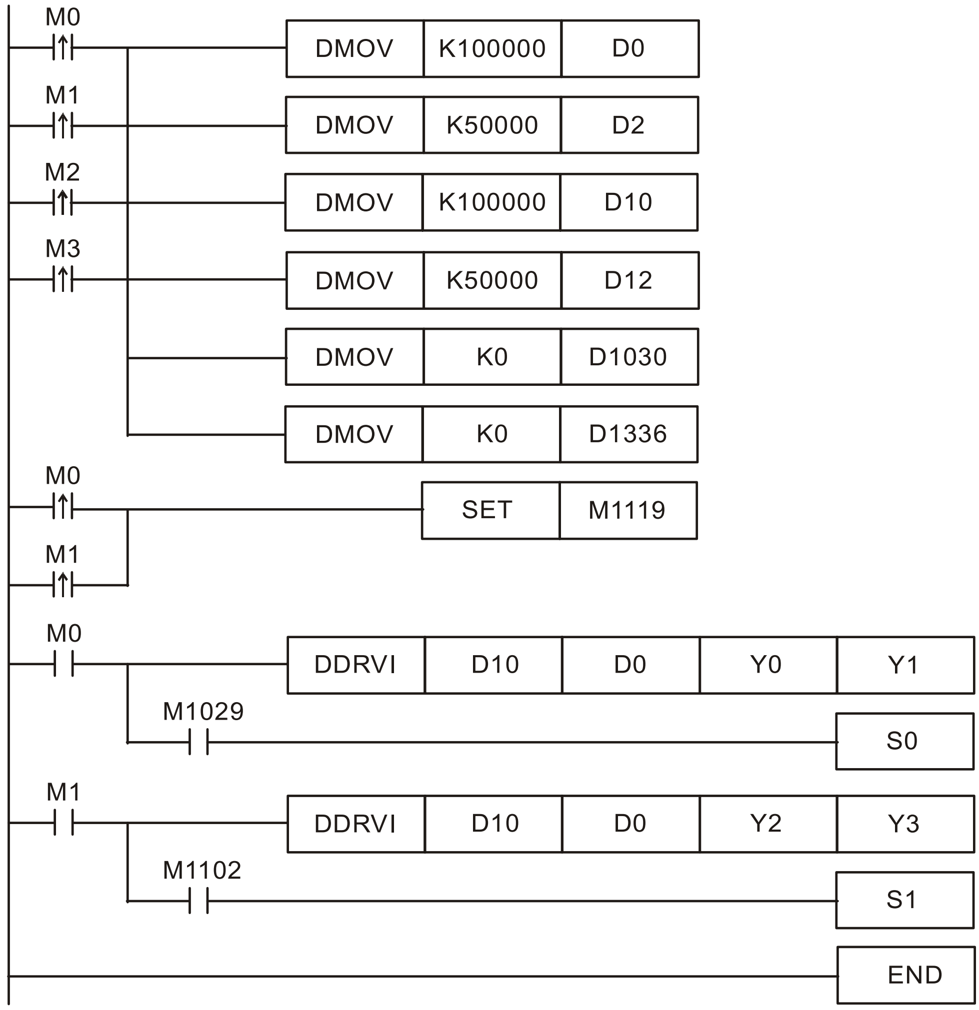
Function Group Enable 2-speed output function of DDRVI instruction

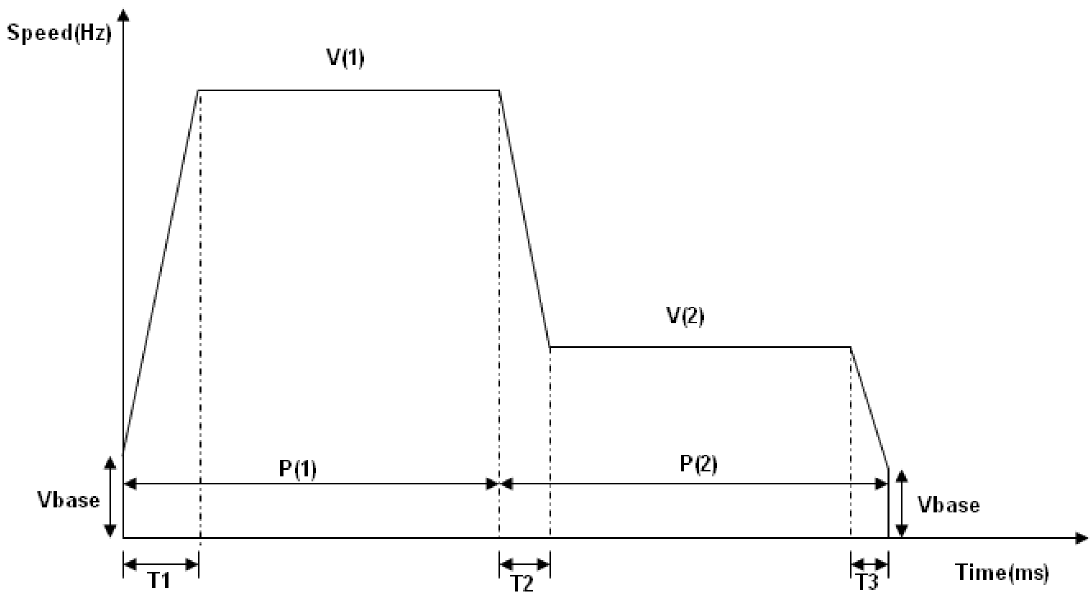
Number M1119

Contents:

When M1119 is ON, 2-speed output function of DDRVI will be enabled.

Example: Assume that D0(D1) is the first speed and D2(D3) is the second speed. D10(D11) is the output pulse number of the first speed and D12(D13) is the output pulse number of the second speed.





Vbase	T1	T2+T3	P(1)	V(1)	P(2)	V(2)
Initial frequency	Ramp-up time	Ramp-down time	Position of the first speed	The first speed	Position of the second speed	The second speed

Function Group Program Execution Error

Number M1067~M1068, D1067~D1068

Contents:

Device	Explanation	Latched	STOP→RUN	RUN→STOP
M1067	Program execution error	None	Clear	Unchanged
M1068	Execution error locked	None	Unchanged	Unchanged
D1067	Error code for program execution	None	Clear	Unchanged
D1068	Address of program execution error	None	Unchanged	Unchanged

Error code explanation:

D1067 error code	Function
0E18	BCD conversion error
0E19	Divisor is 0
0E1A	Use of device exceeds the range (including E, F index register modification)
0E1B	Square root value is negative
0E1C	FROM/TO instruction communication error

2

Function Group I/O Modules Detection

Number D1140, D1142, D1143, D1145

Contents:

D1140: Number of right-side modules (AIO, PT, TC, etc.), max. 8 modules can be connected.

D1142: Number of input points (X) on DIO modules.

D1143: Number of output points (Y) on DIO modules.

D1145: Number of left-side modules (AIO, PT, TC, etc.), max. 8 modules can be connected.

(Only applicable for SA2/SX2).

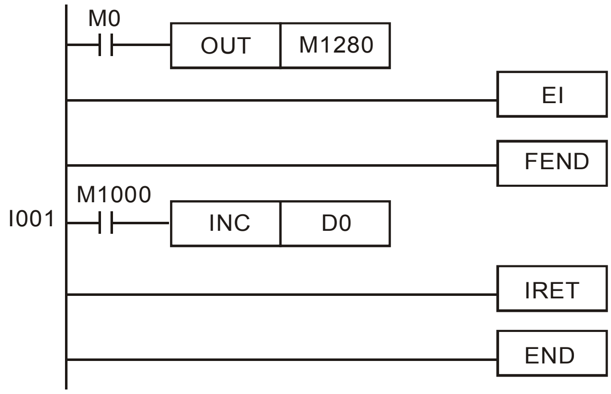
Function Group Reverse Interrupt Trigger Pulse Direction

Number M1280, M1284, M1286

Contents:

1. The falgs should be used with EI instruction and should be inserted before EI instruction
2. The default setting of interrupt I101 (X0) is rising-edge triggered. If M1280 is ON and EI instruction is executed, PLC will reverse the trigger direction as falling-edge triggered. The trigger pulse direction of X1 will be set as rising-edge again by resetting M1280.
3. When M0 = OFF, M1280 = OFF. X0 external interrupt will be triggered by rising-edge pulse.
4. When M0 = ON, M1280 = ON. X0 external interrupt will be triggered by falling-edge pulse.

Users do not have to change I101 to I000.



Function Group Stores Value of High-speed Counter when Interrupt Occurs

Number D1240~D1243

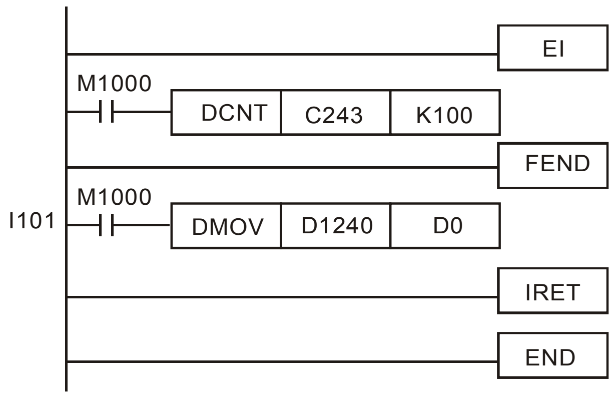
Contents:

1. If extertal interrupts are applied on input points for Reset, the interrupt instructions have the priority in using the input points. In addition, PLC will move the current data in the counters to the associated data registers below then reset the counters.

Special D	D1241, D1240				D1243, D1242		
Counter	C243	C246	C248	C252	C244	C250	C254
Interrupt signal	X1(I100/I101)	X4(I400/I401)			X3(I300/I301)	X5(I500/I501)	

2. Function:
   a) When X0 (counter input) and X1 (external Interrupt) correspondingly work together with C243, and I100/I101, PLC will move the count value to D1241 and D1240.
   b) When X0 (counter input) and X4 (external Interrupt) correspondingly work together with C246, C248, C252 and I400/I401, PLC will move the count value to D1241 and D1240
   c) When X2 (counter input) and X3 (external Interrupt) correspondingly work together with C244, and I300/I301, PLC will move the count value to D1243 and D1242.
   d) When X2 (counter input) and X5 (external Interrupt) correspondingly work together with C250, C254 and I500/I501, PLC will move the count value to D1243 and D1242.

Example:



When external interrupt (X1, I101) occurs during counting process of C243, the count value in C243 will be stored in (D1241, D1240) and C243 is reset. After this, the interrupt subroutine I101 will be executed

Function Group Enabling force-ON/OFF of input point X

Number M1304

Contents:

When M1304 = ON, WPLSoft or ISPSoft can set ON/OFF of input pont X, but the associated hardware LED will not respond to it.

Function Group Output specified pulses or seek Z phase signal when zero point is achieved.

Number M1308, D1312

Contents:

When zero point is achieved, PLC can output specified pulses or seek Z phase signal by this function. Input terminals X2, X3 are the Z-phase signal input point of CH1, CH2. When M1308= ON, D1312 is the setting register to specify the additional pulses within the range -30,000~30,000. Specified value exceeds the range will be changed as the max/min value automatically. When D1312 is set to 0, the additional pulses output function will be disabled.

Functions of other input terminals:

X4 → CH1 DOG signal input

X5 → CH1 LSN signal input

X6 → CH2 DOG signal input

X7 → CH2 LSN signal input

Function Group ID of right side modules on ES2/EX2

Number D1320~ D1327

Contents:

When right side modules are connected on ES2/EX2, the ID of each I/O module will be stored in D1320~D1327 in connection order.

ID of each special module:

Name	ID (HEX)	Name	ID (HEX)
DVP04AD-E2	H’0080	DVP06XA-E2	H’00C4
DVP02DA-E2	H’0041	DVP04PT-E2	H’0082
DVP04DA-E2	H’0081	DVP04TC-E2	H’0083

Function Group ID of left side modules on SA2/SX2

Number D1386~D1393

Contents:

When left side modules are connected on SA2/SX2, the ID of each I/O module will be stored in D1386~D1393 in connection order.

ID of each special module:

Name	ID (HEX)	Name	ID (HEX)
DVP04AD-SL	H’4480	DVP01HC-SL	H’4120
DVP04DA-SL	H’4441	DVP02HC-SL	H’4220
DVP04PT-SL	H’4402	DVPDNET-SL	H’4131
DVP04TC-SL	H’4403	DVPEN01-SL	H’4050
DVP06XA-SL	H’6404	DVPMDM-SL	H’4040
DVP01PU-SL	H’4110	DVPCOPM-SL	H’4133

Function Group Output clear signals when ZRN is completed

Number M1346

Contents:

When M1346 = ON, PLC will output clear signals when ZRN is completed. The clear signals to Y0, Y1 will be sent by Y4 for 20ms, and the clear signals to Y2, Y3 will be sent by Y5 for 20ms.

Function Group PLC LINK

Number M1350-M1356, M1360-M1439, D1355-D1370, D1399, D1415-D1465, D1480-D1991

Contents:

1. PLC LINK supports COM2 (RS-485) with communication of up to 16 slaves and access of up to 50 words.
2. Special D and special M corresponding to Slave ID1~ Slave ID8: (M1353 = OFF, access available for only 16 words)

MASTER PLC															
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
Special D registers for storing the read/written 16 data (Auto-assigned)															
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
Data length for accessing the Slave (Max 16 pieces of data, no access is performed when SV = 0)															
D1434	D1450	D1435	D1451	D1436	D1452	D1437	D1453	D1438	D1454	D1439	D1455	D1440	D1456	D1441	D1457
Starting reference of the Slave to be accessed*															
D1355	D1415	D1356	D1416	D1357	D1417	D1358	D1418	D1359	D1419	D1360	D1420	D1361	D1421	D1362	D1422
M1355 = ON, Slave status is user-defined. Set the linking status of Slave manually by M1360~M1375. M1355 = OFF, Slave status is auto-detected. Linking status of Slave can be monitored by M1360~M1375															
M1360		M1361		M1362		M1363		M1364		M1365		M1366		M1367	
Data interchange status of Slaves.															
M1376		M1377		M1378		M1379		M1380		M1381		M1382		M1383	
Access error flag (ON = normal; OFF = error)															
M1392		M1393		M1394		M1395		M1396		M1397		M1398		M1399	

"Reading completed" flag (turns "Off" whenever access of a Slave is completed)							
M1408	M1409	M1410	M1411	M1412	M1413	M1414	M1415
"Writing completed" flag (turns "Off" whenever access of a Slave is completed)							
M1424	M1425	M1426	M1427	M1428	M1429	M1430	M1431
↓	↓	↓	↓	↓	↓	↓	↓

Slave PLC*															
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

3. Special D and special M corresponding to Slave ID9~ Slave ID16: (M1353 = OFF, access available for only 16 words)

MASTER PLC															
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	Reado ut	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in
Special D registers for storing the read/written 16 pieces of data (Auto-assigned)															
D1736–D1751	D1752–D1767	D1768–D1783	D1784–D1799	D1800–D1815	D1816–D1831	D1832–D1847	D1848–D1863	D1864–D1879	D1880–D1895	D1896–D1911	D1912–D1927	D1928–D1943	D1944–D1959	D1960–D1975	D1976–D1991
Data length for accessing the Slave (Max 16 pieces of data, no access is performed when SV = 0)															
D1442	D1458	D1443	D1459	D1444	D1460	D1445	D1461	D1446	D1462	D1447	D1463	D1448	D1464	D1449	D1465
Starting reference of the Slave to be accessed*															
D1363	D1423	D1364	D1424	D1365	D1425	D1366	D1426	D1367	D1427	D1368	D1428	D1369	D1429	D1370	D1430
M1355 = ON, Slave status is user-defined. Set the linking status of Slave manually by M1360~M1375. M1355 = OFF, Slave status is auto-detected. Linking status of Slave can be monitored by M1360~M1375															
M1368		M1369		M1370		M1371		M1372		M1373		M1374		M1375	
Data interchange status of Slaves															
M1384		M1385		M1386		M1387		M1388		M1389		M1390		M1391	
Access error flag (ON = normal; OFF = error)															
M1400		M1401		M1402		M1403		M1404		M1405		M1406		M1407	
"Reading completed" flag (turns "Off" whenever access of a Slave is completed)															
M1416		M1417		M1418		M1419		M1420		M1421		M1422		M1423	
"Writing completed" flag (turns "Off" whenever access of a Slave is completed)															
M1432		M1433		M1434		M1435		M1436		M1437		M1438		M1439	
↓		↓		↓		↓		↓		↓		↓		↓	

Slave PLC*							
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	Readout	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

4. Special D and special M corresponding to Slave ID1~ID8: (M1353 = ON, access available for up to 50 words)

MASTER PLC															
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	Readout	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in
M1353 = ON, enable access up to 50 words. The user can specify the starting register for storing the read/written data in registers below															
D1480	D1496	D1481	D1497	D1482	D1498	D1483	D1499	D1484	D1500	D1485	D1501	D1486	D1502	D1487	D1503
M1356 = ON, the user can specify the station number of Slave ID1~ID8 in D1900~D1907															
D1900		D1901		D1902		D1903		D1904		D1905		D1906		D1907	
Data length for accessing the Slave (Max 50 pieces of data, no access is performed when SV = 0)															
D1434	D1450	D1435	D1451	D1436	D1452	D1437	D1453	D1438	D1454	D1439	D1455	D1440	D1456	D1441	D1457
Starting reference of the Slave to be accessed*															
D1355	D1415	D1356	D1416	D1357	D1417	D1358	D1418	D1359	D1419	D1360	D1420	D1361	D1421	D1362	D1422
M1355 = ON, Slave status is user-defined. Set the linking status of Slave manually by M1360~M1375. M1355 = OFF, Slave status is auto-detected. Linking status of Slave can be monitored by M1360~M1375															
M1368		M1369		M1370		M1371		M1372		M1373		M1374		M1375	
Data interchange status of Slaves															
M1376		M1377		M1378		M1379		M1380		M1381		M1382		M1383	
Access error flag (ON = normal; OFF = error)															
M1392		M1393		M1394		M1395		M1396		M1397		M1398		M1399	
"Reading completed" flag (turns "Off" whenever access of a Slave is completed)															
M1408		M1409		M1410		M1411		M1412		M1413		M1414		M1415	
"Writing completed" flag (turns "Off" whenever access of a Slave is completed)															
M1424		M1425		M1426		M1427		M1428		M1429		M1430		M1431	

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

Slave PLC*															
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	Readout	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

5. Special D and special M corresponding to Slave ID9~ID16: (M1353 = ON, access available for up to 50 words)

MASTER PLC															
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	Readout	Write in	Read out	Write in	Read out	Write in	Read out	Write in	Read out	Write in
M1353 = ON, enable access up to 50 words. The user can specify the starting register for storing the read/written data in registers below															
D1488	D1504	D1489	D1505	D1490	D1506	D1491	D1507	D1492	D1508	D1493	D1509	D1494	D1510	D1495	D1511
M1356 = ON, the user can specify the station number of Slave ID9~ID16 in D1908~D1915															
D1908		D1909		D1910		D1911		D1912		D1913		D1914		D1915	
Data length for accessing the Slave (Max 50 pieces of data, no access is performed when SV = 0)															
D1442	D1458	D1443	D1459	D1444	D1460	D1445	D1461	D1446	D1462	D1447	D1463	D1448	D1464	D1449	D1465
Starting reference of the Slave to be accessed*															
D1363	D1423	D1364	D1424	D1365	D1425	D1366	D1426	D1367	D1427	D1368	D1428	D1369	D1429	D1370	D1430
M1355 = ON, Slave status is user-defined. Set the linking status of Slave manually by M1368~M1375. M1355 = OFF, Slave status is auto-detected. Linking status of Slave can be monitored by M1368~M1375															
M1368		M1369		M1370		M1371		M1372		M1373		M1374		M1375	
Data interchange status of Slaves															
M1384		M1385		M1386		M1387		M1388		M1389		M1390		M1391	
Access error flag (ON = normal; OFF = error)															
M1400		M1401		M1402		M1403		M1404		M1405		M1406		M1407	
“Reading completed” flag (turns “Off” whenever access of a Slave is completed)															
M1416		M1417		M1418		M1419		M1420		M1421		M1422		M1423	
“Writing completed” flag (turns “Off” whenever access of a Slave is completed)															
M1432		M1433		M1434		M1435		M1436		M1437		M1438		M1439	

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

Slave PLC*															
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	Readout	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

*Note:

- Default setting for starting reference of the Slave (DVP-PLC) to be read: H1064 (D100)
- Default setting for starting reference of the Slave (DVP-PLC) to be written: H10C8 (D200)

6. Explanation:

a) PLC LINK is based on MODBUS communication protocol

b) Baud rate and communication format of all phariferal devices connected to the Slave PLC should be the same as the communication format of Master PLC, no matter which COM port of Slave PLC is used.

c) When M1356 = OFF(Default), the station number of the starting Slave (ID1) can be designated by D1399 of Master PLC through PLC LINK, and PLC will automatically assign ID2~ID16 with consecutive station numbers according to the station number of ID1. For example, if D1399 = K3, Master PLC will send out communication commands to ID1~ID16 which carry station number K3~K18. In addition, care should be taken when setting the station number of Slaves. All station numbers of slaves should not be the same as the station number of the Master PLC, which is set up in D1121/D1255.

d) When both M1353 and M1356 are ON, the station number of ID1~ID16 can be specified by the user in D1900~D1915 of Master PLC. For example, when D1900~D1903 = K3, K3, K5, K5, Master PLC will access the Slave with station number K3 for 2 times, then the slave with station number K5 for 2 times as well. Note that all station numbers of slaves should not be the same as the station number of the Master PLC, and M1353 must be set ON for this function.

e) Station number selection function (M1356 = ON) is supported by versions of ES2/EX2 v1.4.2 or later, SS2/SX2 v1.2 or later, and SA2 v1.0 or later.

2

7. Operation:

a) Set up the baud rates and communication formats. Master PLC and all connected Slave PLCs should have the same communication settings. COM1_RS-232: D1036, COM2_RS-485: D1120, COM3_RS-485: D1109.

b) Set up Master PLC ID by D1121 and the starting slave ID by D1399. Then, set slave ID of each slave PLC. The ID of master PLC and slave PLC cannot be the same.

c) Set data length for accessing. (If data length is not specified, PLC will take default setting or the previous value as the set value. For details of data length registers, please refer to the tables above)

d) Set starting reference of the Slave to be accessed. (Default setting for starting reference to be read: H1064 (D100); default setting for starting reference to be written: H10C8 (D200). For details of starting reference registers, please refer to the tables above)

e) Steps to start PLC LINK:

- Set ON M1354 to enable simultabeous data read/write in a polling of PLC LINK..
- M1355 = ON, Slave status is user-defined. Set the linking status of Slave manually by M1360~M1375. M1355 = OFF, Slave status is auto-detected. Linking status of Slave can be monitored by M1360~M1375
- Select auto mode on PLC LINK by M1351 or manual mode by M1352 (Note that the 2 flags should not be set ON at the same time.) After this, set up the times of polling cycle by D1431.
- Finally, enable PLC LINK (M1350)

8. The Operation of Master PLC:

a) M1355 = ON indicates that Slave status is user-defined. Set the linking status of Slave manually by M1360~M1375.

b) M1355 = OFF indicates that Slave status is auto-detected. Linking status of Slave can be monitored by M1360~M1375.

- Enable PLC LINK (M1350). Master PLC will detect the connected Slaves and store the number of connected PLCs in D1433. The time for detection differs by number of connected Slaves and time-out setting in D1129.
- M1360~M1375 indicate the linking status of Slave ID 1~16
- If no slave is detected, M1350 will be OFF and PLC LINK will be stopped.
- PLC will only detect the number of slaves at the first time when M1350 turns ON.
- After auto-detection is completed, master PLC starts to access each connected slave. Once slave PLC is added after auto-detection, master PLC cannot access it unless auto-detection is conducted again.

c) Simultaneous read/write function (M1354) has to be set up before enabling PLC LINK. Setting up this flag during PLC LINK execution will not take effect.

d) When M1354 = ON, PLC takes Modbus Function H17 (simultaneous read/write function) for PLC LINK communication function. If the data length to be written is set to 0, PLC will select Modbus Function H03 (read multiple WORDs) automatically. In the same way, if data length to be read is set to 0, PLC will select Modbus Function H06 (write single WORD) or Modbus Function H10 (write multiple WORDs) for PLC LINK communication function.

e) When M1353 = OFF, PLC LINK accesses the Slave with max 16 words, and the data is automatically stored in the corresponding registers. When M1353 = ON, up to 50 words are accessible and the user can specify the starting register for storing the read/written data. For example, if the register for storing the read/written data on Slave ID1 is specified as D1480 = K500, D1496 = K800, access data length D1434 = K50, D1450 = K50, registers of Master PLC D500~D549 will store the data read from Slave ID1, and the data stored in D800~D849 will be written into Slave ID1.

f) Master PLC conducts reading before writing. Both reading and writing is executed according to the range specified by user.

g) Master PLC accesses slave PLCs in order, i.e. data access moves to next slave only when access on previous slave is completed.

9. Auto mode and Manual mode:

a) Auto mode (M1351): when M1351 = ON, Master PLC will access slave PLCs as the operation described above, and stop the polling till M1350 or M1351 is OFF.

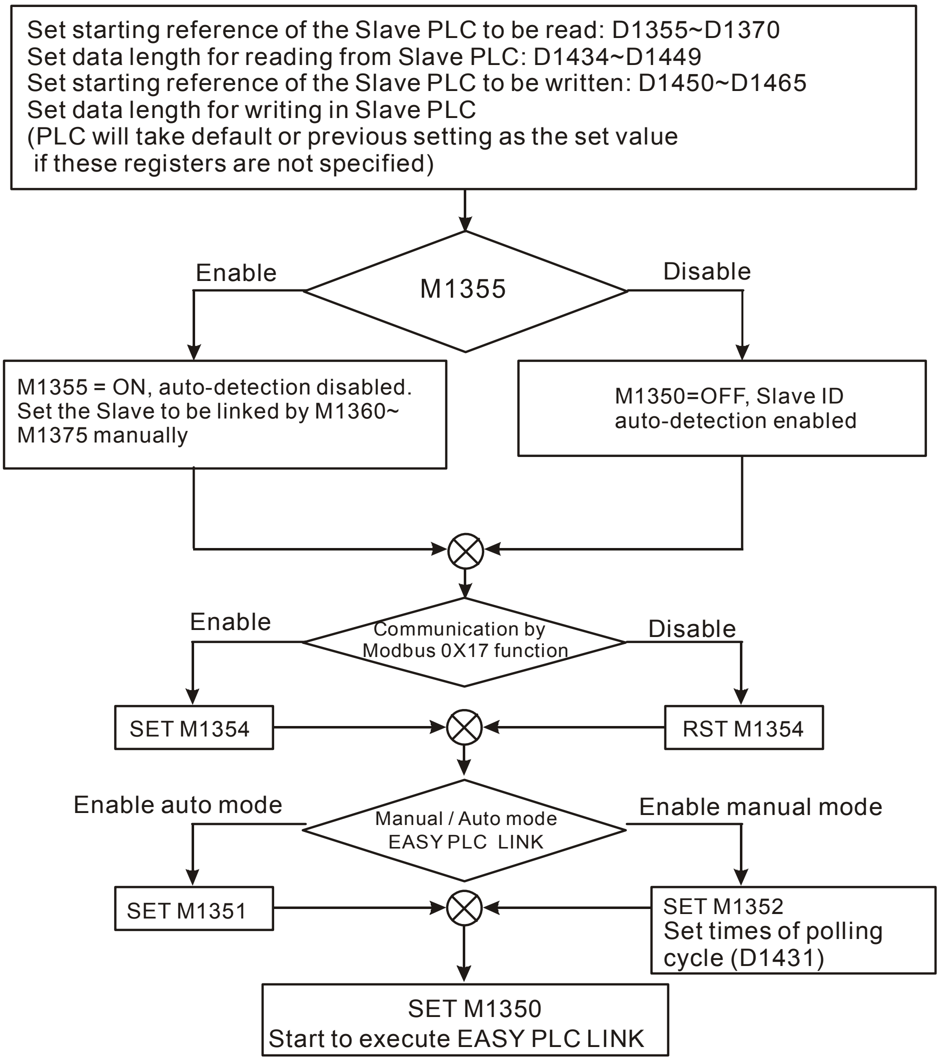
b) Manual mode (M1352): When manual mode is selected, times of polling cycle in D1431 has to be set up. A full polling cycle refers to the completion of accessing all Slaves. When PLC LINK is enabled, D1432 starts to store the times of polling. When D1431 = D1432, PLC LINK stops and M1352 is reset. When M1352 is set ON again, PLC will start the polling according to times set in D1431 automatically.

c) Note:

- Auto mode M1351 and manual mode M1352 cannot be enabled at the same time. If M1351 is enabled after M1352 is ON, PLC LINK will stop and M1350 will be reset.
- Communication timeout setting can be modified by D1129 with available range 200 ≦ D1129 ≦ 3000. PLC will take the upper / lower bound value as the set value if the specified value is out of the available range. D1129 has to be set up before M1350 = ON.
- PLC LINK function is only valid when baud rate is higher than 1200 bps. When baud rate is less than 9600 bps, please set communication time-out to more than 1 second.
- The communication is invalid when data length to be accessed is set to 0.
- Access on 32-bit high speed counters (C200~C255) is not supported.
- Available range for D1399: 1 ~ 230. PLC will take the upper / lower bound value as the set value if the specified value exceeds the availanle range.
- D1399 has to be set up before enabling PLC LINK. Setting up this register during PLC LINK execution will not take effect.
- Advantage of using D1399 (Designating the ID of starting Slave):
  In old version PLC LINK, PLC detects Slaves from ID1 to ID16. Therefore, when PLC LINK is applied in multi-layer networks, e.g. 3 layers of networks, the Slave ID of 2nd and 3rd layer will be repeated. When Slave ID is repeated, i.e. the same as Master ID, the Slave will be passed. In this case, only 15 Slaves can be connected in 3rd layer. To solve this problem, D1399 can be applied for increasing the connectable Slaves in multi-layer network structure.

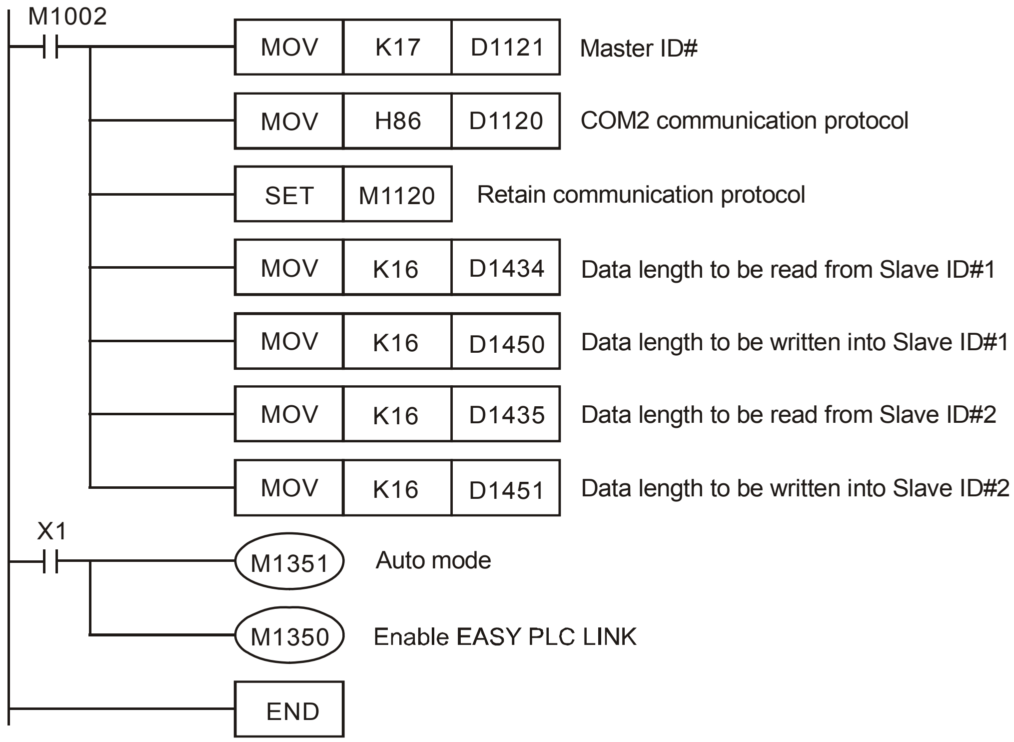
2

10. Operation flow chart:

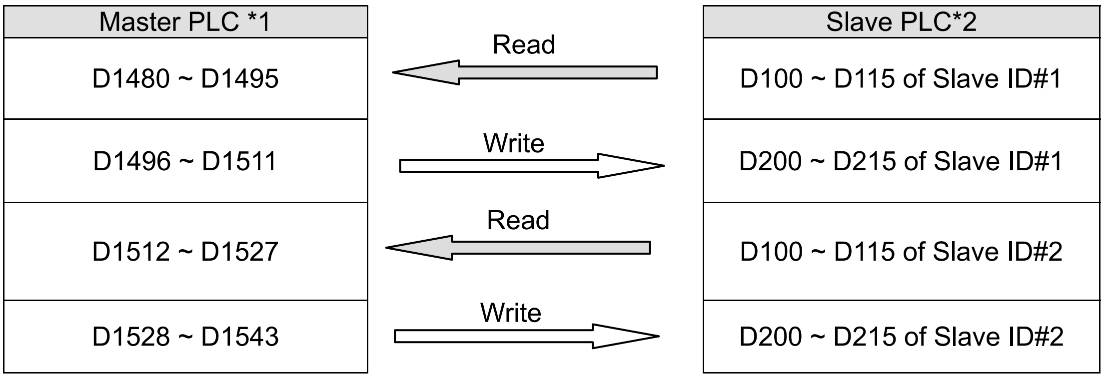


11. Example 1: Connect 1 Master and 2 Slaves by RS-485 and exchange 16 data between Master and Slaves through PLC LINK

a) Write the ladder diagram program into Master PLC (ID#17)



b) When X1 = On, the data exchange between Master and the two Slaves will be automatically executed by PLC LINK. The data in D100 ~ D115 in the two Slaves will be read into D1480 ~ D1495 and D1512 ~ D1527 of the Master, and the data in D1496 ~ D1511 and D1528 ~ D1543 will be written into D200 ~ D215 of the two Slaves.



c) Assume the data in registers for data exchange before enabling PLC LINK (M1350 = OFF) is as below:

Master PLC	Preset value	Slave PLC	Preset value
D1480 ~ D1495	K0	D100 ~ D115 of Slave ID#1	K5,000
D1496 ~ D1511	K1,000	D200 ~ D215 of Slave ID#1	K0
D1512 ~ D1527	K0	D100 ~ D115 of Slave ID#2	K6,000
D1528 ~ D1543	K2,000	D200 ~ D215 of Slave ID#2	K0

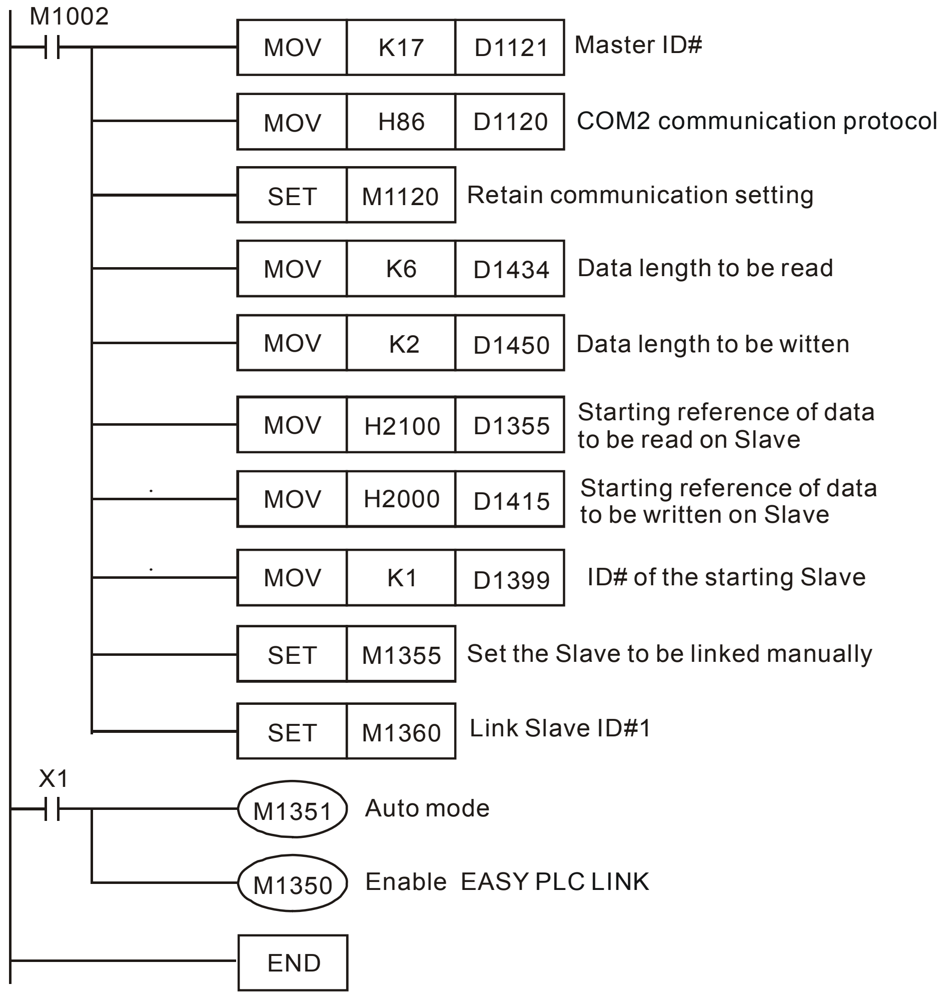
After PLC LINK is enabled (M1350 = ON), the data in registers for data exchange becomes:

Master PLC	Preset value	Slave PLC	Preset value
D1480 ~ D1495	K5,000	D100 ~ D115 of Slave ID#1	K5,000
D1496 ~ D1511	K1,000	D200 ~ D215 of Slave ID#1	K1,000
D1512 ~ D1527	K6,000	D100 ~ D115 of Slave ID#2	K6,000
D1528 ~ D1543	K2,000	D200 ~ D215 of Slave ID#2	K2,000

d) Up to16 Slaves can be accessed through PLC LINK. For allocation of D100 ~ D115 and D200 ~ D215 in each Slave PLC, please refer to the tables of Special M and Special D of this function in previous pages.

12. Example 2: Conncet DVP-PLC with VFD-M inverter and control the RUN, STOP, Forward operation, Reverse operation through PLC LINK.

a) Write the ladder diagram program into Master PLC (ID#17)



b) M1355 = ON. Set the Slave to be linked manually by M1360~M1375. Set ON M1360 to link Slave ID#1.

c) Address H2100-H2105 maps to registers D1480-D1485 of PLC. When X1 = ON, PLC LINK executes, and the data in H2100-H2105 will be displayed in D1480-D1485.

d) Address H2000-H2001 maps to registers D1496-D1497 of PLC. When X1 = ON, PLC LINK executes, and the parameter in H2000-H2001 will be specified by D1496-D1497.

e) Commands of VFD can be specified by changing the value in D1496, e.g. D1496 = H12=>VFD forward operation; D1496 = H1=> VFD stops）

f) Frequency of VFD can be specified by changing the value in D1497, e.g. D1497 = K5000, set VFD frequency as 50kHz.

g) In addition to VFD AC motor drives, devices support MODBUS protocol such as DTA/DTB temperature controllers and ASDA servo drives can also be connected as Slaves. Up to 16 Slaves can be connected.

13. D1354 is PLC link scan cycle with unit is 1ms and max. display value is K32000. D1354 = K0 when PLC Link stops or when the first scan is completed.

Instruction Set

3

This chapter explains all of the instructions that are used with DVP-ES2/EX2/SS2/ SA2/SX2 as well as detailed information concerning the usage of the instructions.

Chapter Contents

3.1 Basic Instructions (without API numbers)

Instruction	Function	Operand	Execution speed (us)	Steps
LD	Load NO contact	X, Y, M, S, T, C	0.76	1~3
LDI	Load NC contact	X, Y, M, S, T, C	0.78	1~3
AND	Connect NO contact in series	X, Y, M, S, T, C	0.54	1~3
ANI	Connect NC contact in series	X, Y, M, S, T, C	0.56	1~3
OR	Connect NO contact in parallel	X, Y, M, S, T, C	0.54	1~3
ORI	Connect NC contact in parallel	X, Y, M, S, T, C	0.56	1~3
ANB	Connect a block in series	N/A	0.68	1
ORB	Connect a block in parallel	N/A	0.76	1
MPS	Start of branches. Stores current result of program evaluation	N/A	0.74	1
MRD	Reads the stored current result from previous MPS	N/A	0.64	1
MPP	End of branches. Pops (reads and resets) the stored result in previous MPS	N/A	0.64	1
OUT	Output coil	Y, S, M	0.88	1~3
SET	Latches the ON status	Y, S, M	0.76	1~3
RST	Resets contacts, registers or coils	Y, M, S, T, C, D, E, F	2.2	3
MC	Master control Start	N0~N7	1	3
MCR	Master control Reset	N0~N7	1	3
END	Program End	N/A	1	1
NOP	No operation	N/A	0.4	1
P	Pointer	P0~P255	0.4	1
I	Interrupt program pointer	I□□□	0.4	1
STL	Step ladder start instruction	S	2.2	1
RET	Step ladder return instruction	N/A	1.6	1
NP	Negative contact to Positive contact	N/A	1.66	1
PN	Positive contact to Negative contact	N/A	1.62	1

Note: The execution speed is obtained by basic test programs, therefore the actual instruction execution time could be longer due to a more complicated program, e.g. program contains multiple interruptions or high speed input/output.

3.2 Explanations to Basic Instructions

Mnemonic	Operands	Function	Program steps	Controllers
LD	X, Y, M, S, T, C	Load NO contact	1~3	ES2/EX2 | SS2 | SA2 | SX2

Explanations:

The LD instruction is used to load NO contact which connects to left side bus line or starts a new block of program connecting in series or parallel connection.

Program example:

Ladder diagram:

Instruction:		Operation:
LD	**X0**	Load NO contact X0
AND	X1	Connect NO contact X1 in series
OUT	Y1	Drive coil Y1

Mnemonic	Operands	Function	Program steps	Controllers
LDI	X, Y, M, S, T, C	Load NC contact	1~3	ES2/EX2 | SS2 | SA2 | SX2

Explanations:

The LDI instruction is used to load NC contact which connects to left side bus line or starts a new block of program connecting in series or parallel connection.

Program example:

Ladder diagram:

Instruction:		Operation:
LDI	**X0**	Load NC contact X0
AND	X1	Connect NO contact X1 in series
OUT	Y1	Drive coil Y1

Mnemonic	Operands	Function	Program steps	Controllers
AND	X, Y, M, S, T, C	Connect NO contact in series	1~3	ES2/EX2 | SS2 | SA2 | SX2

Explanations:

The AND instruction is used to connect NO contact in series.

Program example:

Ladder diagram:

Instruction:		Operation:
LDI	X1	Load NC contact X1
AND	**X0**	Connect NO contact X0 in series
OUT	Y1	Drive Y1 coil

Mnemonic	Operands	Function	Program steps	Controllers ES2/EX2 | SS2 | SA2 | SX2
ANI	X, Y, M, S, T, C	Connect NC contact in series	1~3	

Explanations:

The ANI instruction is used to connect NC contact in series.

Program example:

Ladder diagram:

Instruction:		Operation:
LD	X1	Load NO contact X1
ANI	**X0**	Connect NC contact X0 in series
OUT	Y1	Drive Y1 coil

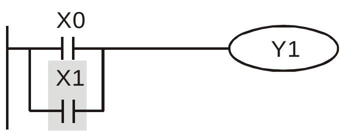
Mnemonic	Operands	Function	Program steps	Controllers ES2/EX2 | SS2 | SA2 | SX2
OR	X, Y, M, S, T, C	Connect NO contact in parallel	1~3	

Explanations:

The OR instruction is used to connect NO contact in parallel.

Program example:

Ladder diagram:



Instruction:		Operation:
LD	X0	Load NO contact X0
OR	**X1**	Connect NO contact X1 in parallel
OUT	Y1	Drive Y1 coil

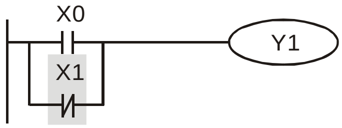
Mnemonic	Operands	Function	Program steps	Controllers ES2/EX2 | SS2 | SA2 | SX2
ORI	X, Y, M, S, T, C	Connect NC contact in parallel	1~3	

Explanations:

The ORI instruction is used to connect NC contact in parallel.

Program example:

Ladder diagram:



Instruction:		Operation:
LD	X0	Load NO contact X0
ORI	**X1**	Connect NC contact X1 in parallel
OUT	Y1	Drive Y1 coil

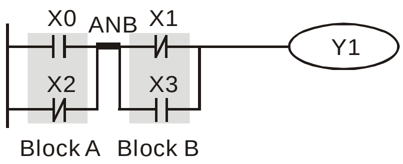
Mnemonic	Function	Program steps	Controllers
ANB	Connect a block in series	1	ES2/EX2 | SS2 | SA2 | SX2

Explanations:

The ANB instruction is used to connect a circuit block to the preceding block in series. Generally, the circuit block to be connected in series consists of several contacts which form a parallel connection structure.

Program example:

Ladder diagram:



Instruction:		Operation:
LD	X0	Load NO contact X0
ORI	X2	Connect NC contact X2 in parallel
LDI	X1	Load NC contact X1
OR	X3	Connect NO contact X3 in parallel
ANB		Connect circuit block in series
OUT	Y1	Drive Y1 coil

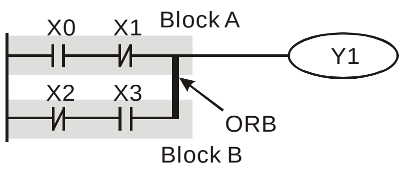
Mnemonic	Function	Program steps	Controllers
ORB	Connect a block in parallel	1	ES2/EX2 | SS2 | SA2 | SX2

Explanations:

The ORB instruction is used to connect a circuit block to the preceding block in parallel. Generally, the circuit block to be connected in parallel consists of several contacts which form a serial connection structure.

Program example:

Ladder diagram:



Instruction:		Operation:
LD	X0	Load NO contact X0
ANI	X1	Connect NC contact X1 in series
LDI	X2	Load NC contact X2
AND	X3	Connect NO contact X3 in series
ORB		Connect circuit block in parallel
OUT	Y1	Drive Y1 coil

Mnemonic	Function	Program steps	Controllers
MPS	Start of branches. Stores current result of program evaluation	1	ES2/EX2 SS2 SA2 SX2

Explanations:

As the start of branches, MPS stores current result of program evaluation at the point of divergence.

Mnemonic	Function	Program steps	Controllers
MRD	Reads the stored current result from previous MPS	1	ES2/EX2 SS2 SA2 SX2

Explanations:

MRD reads the stored current result from previous MPS and operates with the contact connected after MRD.

Mnemonic	Function	Program steps	Controllers
MPP	End of branches. Pops (reads and resets) the stored result in previous MPS.	1	ES2/EX2 SS2 SA2 SX2

Explanations:

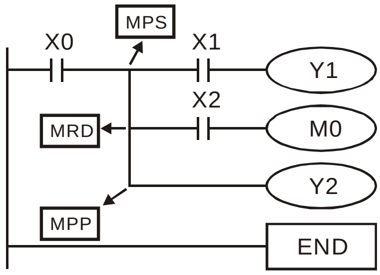
As the end of branches, MPP pops the stored result in previous MPP, which means it operates with the contact connected first then resets the storage memory.

Points to note:

1. Every MPS can not be applied without a corresponding MPP
2. Max. 8 MPS-MPP pairs can be applied..

Program example:

Ladder diagram:



Instruction:		Operation:
LD	X0	Load NO contact X0
MPS		Store current status
AND	X1	Connect NO contact X1 in series
OUT	Y1	Drive Y1 coil
MRD		Read the stored status
AND	X2	Connect NO contact X2 in series
OUT	M0	Drive M0 coil
MPP		Read the stored status and reset
OUT	Y2	Drive Y2 coil
END		End of program

Note: When compiling ladder diagram with WPLSoft, MPS, MRD and MPP will be automatically added to the compiled results in instruction format. However, users programming in instruction mode have to enter branch instructions as required.

Mnemonic	Operands	Function	Program steps	Controllers			
				ES2/EX2	SS2	SA2	SA2
OUT	Y, M, S	Output coil	1~3				

Explanations:

Output the program evaluation results before OUT instruction to the designated device.

Status of coil contact

Evaluation result	OUT instruction		
	Coil	Associated Contacts	
		NO contact（normal open）	NC contact（normal close）
FALSE	OFF	Current blocked	Current flows
TRUE	ON	Current flows	Current blocked

Program example:

Ladder diagram:

Instruction:		Operation:
LDI	X0	Load NC contact X0
AND	X1	Connect NO contact X1 in series
OUT	**Y1**	Drive Y1 coil

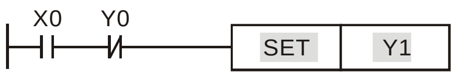
Mnemonic	Operands	Function	Program steps	Controllers
SET	Y, M, S	Latches the ON status	1~3	ES2/EX2 SS2 SA2 SX2

Explanations:

When the SET instruction is driven, its designated device will be ON and latched whether the SET instruction is still driven. In this case, RST instruction can be applied to turn off the device.

Program example:

Ladder Diagram:



Instruction:		Operation:
LD	X0	Load NO contact X0
ANI	Y0	Connect NC contact Y0 in series
SET	**Y1**	Drive Y1 and latch the status

Mnemonic	Operands	Function	Program steps	Controllers
RST	Y, M, S, T, C, D, E, F	Resets contacts, registers or coils	3	ES2/EX2 SS2 SA2 SX2

Explanations:

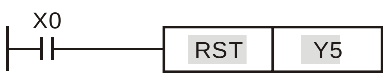
Device status when RST instruction is driven:

Device	Status
S, Y, M	Coil and contact are set to OFF.
T, C	Current value is cleared. Associated contacts or coils are reset .
D, E, F	The content is set to 0.

Status of designated devices remains the same when RST instruction is not executed.

Program example:

Ladder diagram:



Instruction:		Operation:
LD	X0	Load NO contact X0
RST	**Y5**	Reset contact Y5

Mnemonic	Operands	Function	Program steps	Controllers
MC/MCR	N0~N7	Master control Start/Reset	3	ES2/EX2 SS2 SA2 SX2

Explanations:

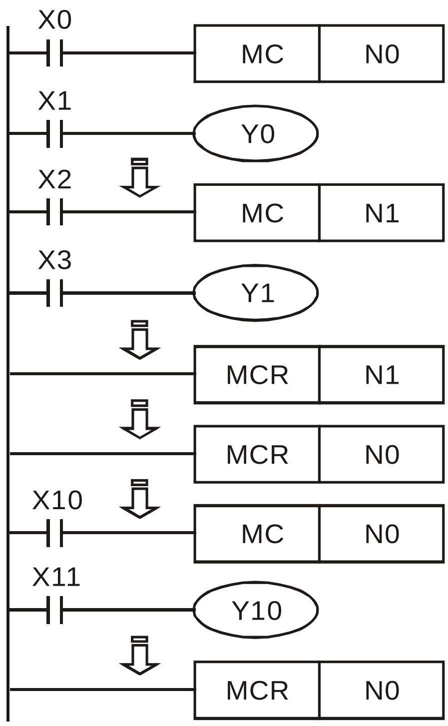
MC is the master-control start instruction. When MC instruction executes, the program execution turns to the designated nest level and executes the instructions between MC and MCR. However, MCR is the master-control reset instruction placed at the end of the designated nest level and no drive contact is required before MCR. When MC/MCR is not active, devices and instructions between MC/MCR will operate as the following table.

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

Note: MC-MCR master-control instruction supports max 8 layers of nest levels. Please use the instructions in order from N0~ N7.

Program example:

Ladder diagram:



Instruction:		Operation:
LD	X0	Load NO contact X0
MC	**N0**	Enable N0 nest level
LD	X1	Load NO contact X1
OUT	Y0	Drive coil Y1
:		
LD	X2	Load NO contact X2
MC	**N1**	Enable N1 nest level
LD	X3	Load NO contact X3
OUT	Y1	Drive coil Y1
:		
MCR	**N1**	Reset N1 nest level
:		
MCR	**N0**	Reset N0 nest level
:		
LD	X10	Load NO contact X10
MC	**N0**	Enable N0 nest level
LD	X11	Load NO contact X11
OUT	Y10	Drive coil Y10
:		
MCR	**N0**	Reset N0 nest level

Mnemonic	Function	Program steps	Controllers			
			ES2/EX2	SS2	SA2	SX2
END	Program End	1				

Explanations:

END instruction needs to be connected at the end of program. PLC will scan from address 0 to END instruction and return to address 0 to scan again.

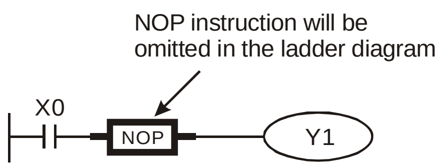
Mnemonic	Function	Program steps	Controllers			
			ES2/EX2	SS2	SA2	SX2
NOP	No operation	1				

Explanation:

NOP instruction does not conduct any operations in the program, i.e. the operation result remains the same after NOP is executed. Generally NOP is used for replacing certain instruction without altering original program length.

Program example:

Ladder Diagram:



Instruction:		Operation:
LD	X0	Load NO contact X0
NOP		No operation
OUT	Y1	Drive coil Y1

Mnemonic	Function	Program steps	Controllers			
			ES2/EX2	SS2	SA2	SX2
NP	Negative contact to Positive contact	1				

Explanation:

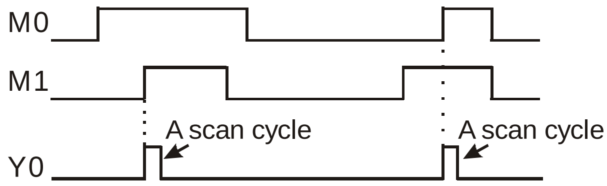
When the conditions preceding NP command change from false to true, NP command (works as contact A) will be ON for a scan cycle. In the next scan cycle it turns OFF.

Program Example:

Ladder Diagram:

Instruction:		Operation:
LD	M0	Load NO contact M0
AND	M1	Connect NO contact M1 in series
NP		Negative contact to Positive contact
OUT	Y0	Drive coil Y0

Timing Diagram:



Mnemonic	Function	Program steps	Controllers ES2/EX2 | SS2 | SA2 | SX2
PN	Positive contact to Negative contact	1	

Explanation:

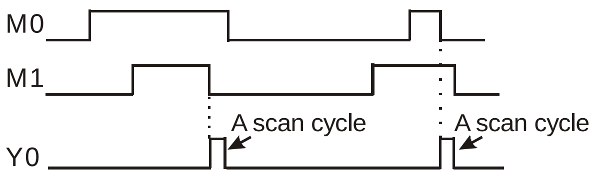
When the conditions preceding PN command change from true to false, PN command (works as contact A) will be ON for a scan cycle. In the next scan cycle it turns OFF.

Program Example:

Ladder Diagram:

Instruction:		Operation:
LD	M0	Load NO contact M0
AND	M1	Connect NO contact M1 in series
PN		Negative contact to Positive contact
OUT	Y0	Drive coil Y0

Timing Diagram:



3.3 Pointers

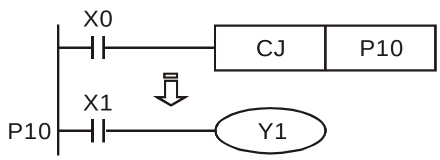
Mnemonic	Operands	Function	Program steps	Controllers ES2/EX2 | SS2 | SA2 | SX2
P	P0~P255	Pointer	1	

Explanation:

Pointer P is used with API 00 CJ and API 01 CALL instructions. The use of P does not need to start from P0, and the No. of P cannot be repeated; otherwise, unexpected errors may occur. For other information on P pointers, please refer to section 2.12 in this manual

Program example 1:

Ladder Diagram:



Instruction:		Operation:
LD	X0	Load NO contact X0
CJ	P10	Jump to P10
:		
P10		Pointer P10
LD	X1	Load NO contact X1
OUT	Y1	Drive coil Y1

3.4 Interrupt Pointers

Mnemonic	Function	Program steps	Controllers
I	Interrupt program pointer	1	ES2/EX2 SS2 SA2 SX2

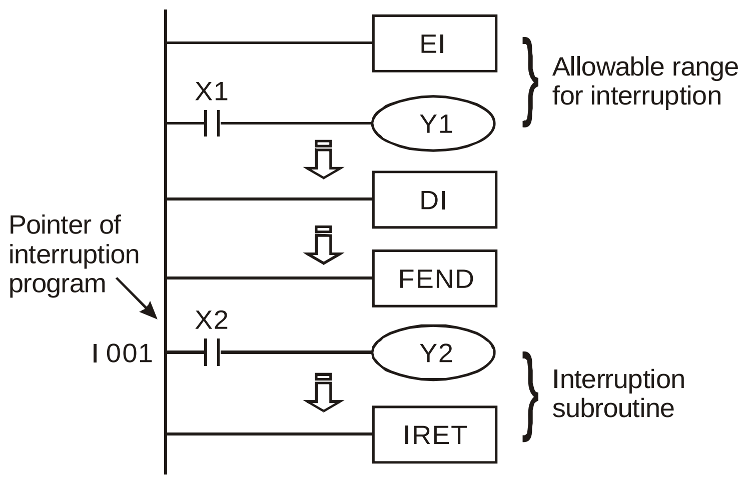
3

Explanations:

A interruption program has to start with a interruption pointer (I□□□) and ends with API 03 IRET. I instruction has to be used with API 03 IRET, API 04 EI, and API 05 DI. For detailed information on interrupt pointes, please refer to section 2.12 in this manual

Program example:

Ladder diagram:



Instruction code:		Operation:
EI		Enable interruption
LD	X1	Load NO contact X1
OUT	Y1	Drive Y1 coil
:		
DI		Disable interruption
:		
FEND		Main program ends
I001		Interruption pointer
LD	X2	Load NO contact X2
OUT	Y2	Drive Y2 coil
:		
IRET		Interruption return

External interrupt:

ES2 supports 8 external input interrupts: (I000/I001, X0), (I100/I101, X1), (I200/I201, X2), (I300/I301, X3), (I400/I401, X4), (I500/I501, X5), (I600/I601, X6) and (I700/I701, X7). (01, rising-edge trigger, 00, falling-edge trigger)

Timer Interrupts:

ES2 supports 2 timer interrupts: I602~I699, I702~I799, (Timer resolution: 1ms)

Communication Interrupts:

ES2 supports 3 communication interrupts: I140, I150 and I160.

Counter Interrupts:

ES2 supports 8 high-speed counter interrupts: I010, I020, I030, I040, I050, I060, I070 and I080.

3.5 Application Programming Instructions

1. PLC instructions are provided with a unique mnemonic name to make it easy to remember instructions. In the example below the API number given to the instruction is 12, the mnemonic name is MOV and the function description is Move.

API	Mnemonic			Operands	Function	Controllers			
12	D	MOV	P	S D	Move	ES2/EX2	SS2	SA2	SX2

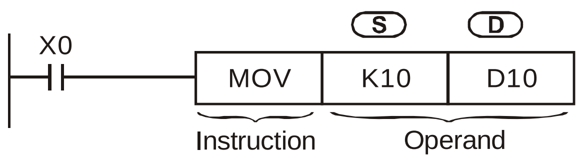
Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MOV, MOVP: 5 steps
S					*	*	*	*	*	*	*	*	*	*	*	DMOV, DMOVP: 9 steps
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

2. The area of ‘Operands’ lists the devices (operands) required for the instruction. Identification letters are used to associate each operand with its function, e.g. D-destination, S-source, n, m-number of devices. Additional numeric suffixes will be attached if there are more than one operand with the same function, e.g. S_1, S_2.
3. When using WPLSoft for programming user program, it is not necessary to remember the API number of an instruction since WPLSoft offers drop down list to select an instruction.
4. Applicable controllers are identified by the boxes at the right of the table. For individual instruction properties of Pulse, 16-bit or 32-bit, please refer to the box down the table.
5. 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, i.e. if an instruction was being used with both pulse and 32 bit operation it appears as “D***P” where *** is the basic mnemonic.

Instruction Composition

The application instructions are specified by API numbers 0~--- and each has its mnemonic. When designing the user program with ladder editing program (WPLSoft), users only need to key in the mnemonic, e.g. MOV, and the instruction will be inserted. Instructions consist of either just the instruction or the instruction followed by operands for parameter settings. Take MOV instruction for example:



Mnemonic : Indicates the name and the function of the instruction

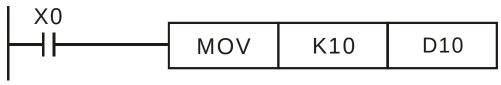
Operand : The parameter setting for the instruction

S	Source: if there are more than one source is required, it will be indicated as S_1, S_2...etc.
D	Destination: if there are more than one destination is required, it will be indicated as D_1, D_2...etc.
If the operand can only be constant K/H or a register, it will be represented as **m**, **m_1**, **m_2**, **n**, **n_1**, **n_2**…etc.	

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

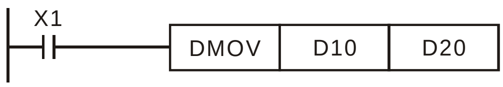
The length of operand can be divided into two groups: 16-bit and 32-bit for processing data of different length. A prefix "D" indicates 32-bit instructions.

16-bit MOV instruction



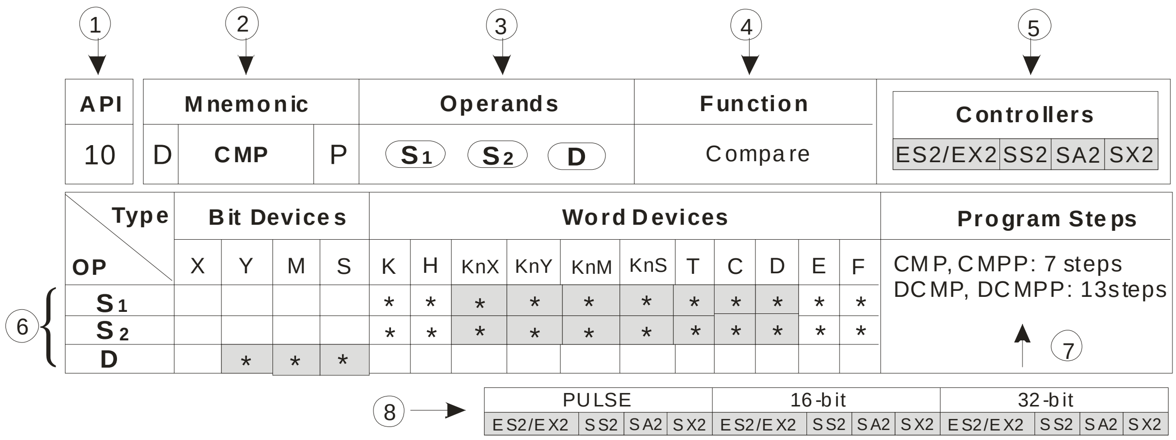
When X0 = ON, K10 will be sent to D10.

32-bit DMOV instruction



When X1 = ON, the content in (D11, D10) will be sent to (D21, D20).

Explanation of the format of application instruction



① API number for instruction

② The core mnemonic code of instruction

A prefix "D" indicates a 32 bit instruction

A suffix "P" in this box indicates a pulse instruction

③ Operand format of the instruction

④ Function of the instruction

⑤ Applicable PLC models for this instruction

⑥ A symbol **"*"** is the device can use the index register. For example, device D of operand S1 supports index E and F.

A symbol **"*"** is given to device which can be used for this operand

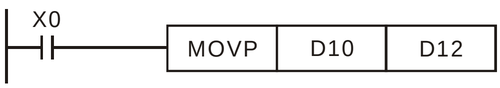
⑦ Steps occupied by the 16-bit/32-bit/pulse instruction

⑧ Applicable PLC models for 16-bit/32-bit/pulse execution instruction.

Continuous execution vs. Pulse execution

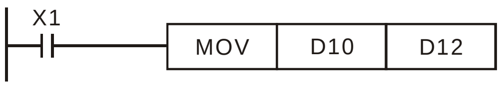
1. There are two execution types for instructions: continuous execution instruction and pulse instruction. Program scan time is shorter when instructions are not executed. Therefore, using the pulse execution instruction can reduce the scan time of the program.
2. The ‘pulse’ function allows the associated instruction to be activated on the rising edge of the drive contact. The instruction is driven ON for the duration of one program scan.
3. In addition, while the control input remains ON, the associate instruction will not be executed for the second time. To re-execute the instruction the control input must be turned from OFF to ON again.

Pulse execution instruction



When X0 goes from OFF to ON, MOVP instruction will be executed once and the instruction will not be executed again in the scan period

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.

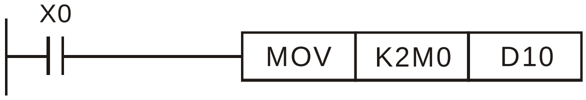
Operands

1. Bit devices X, Y, M, and S can be combined into word device, storing values and data for operations in the form of KnX, KnY, KnM and KnS in an application instruction.
2. Data register D, timer T, counter C and index register E, F are designated by general operands.
3. A data register D consists of 16 bits, i.e. a 32-bit data register consists of 2 consecutive D registers.
4. If an operand of a 32-bit instruction designates D0, 2 consecutive registers D1 and D0 will be occupied. D1 is thehigh word and D0 is the low word. This proncipal also applys to timer T and 16-bit counters C0 ~ C199.
5. When the 32-bit counters C200 ~ C255 are used as data registers, they can only be designataed by the operands of 32-bit instructions.

Operand Data format

1. X, Y, M, and S are defined as bit devices which indicate ON/OFF status.
2. 16-bit (or 32-bit) devices T, C, D, and registers E, F are defined as word devices.

3. "Kn" can be placed before bit devices X, Y, M and S to make it a word device for performing word-device operations. (n = 1 refers to 4 bits. For 16-bit instruction, n = K1 ~ K4; for 32-bit instruction, n = K1 ~ K8). For example, K2M0 refers to 8 bits, M0 ~ M7.



When X0 = ON, the contents in M0 ~ M7 will be moved to b0 ~b7 in D10 and b8 ~b15 will be set to "0".

Kn values

16-bit instruction	
Designated value: K-32,768 ~ K32,767	
16-bit instruction: (K1~K4)	
K1 (4 bits)	0~15
K2 (8 bits)	0~255
K3 (12 bits)	0~4,095
K4 (16 bits)	-32,768~+32,767

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

Flags

1. General Flags

   The flags listed below are used for indicating the operation result of the application instruction:

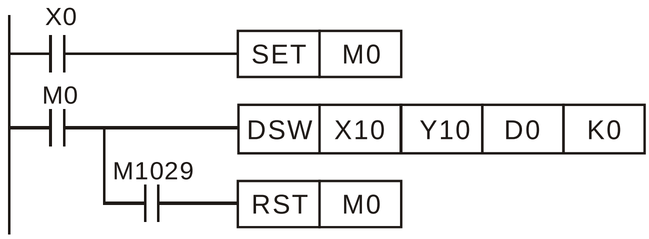
   M1020: Zero flag

   M1021: Borrow flag

   M1022: Carry flag

   M1029: Execution of instruction is completed

   All flags will turn ON or OFF according to the operation result of an instruction. For example, the execution result of instructions ADD/SUB/MUL/DVI will affect the status of M1020 ~ M1022. When the instruction is not executed, the ON/OFF status of the flag will be held. The status of the four flags relates to many instructions. See relevant instructions for more details.



When X0 = ON, DSW will be enabled.
When X0 = OFF, M0 is latched. M0 will be reset only when DSW instruction is completed to activate M1029.

2. Error Operation Flags

   Errors occur during the execution of the instruction when the combination of application instructions is incorrect or the devices designated by the operand exceed their range. Other than errors, the flags listed in the table below will be On, and error codes will also appear.

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.

Device	Explanation
M1067 D1067 D1069	When operational errors occur, M1067 = ON. D1067 displays the error code. D1069 displays the address where the error occurs. Other errors occurring will update the contents in D1067 and D1069. M1067 will be OFF when the error is cleared.
M1068 D1068	When operational errors occur, M1068 = ON. D1068 displays the address where the error occurs. Other errors occurring wil not update the content in D1068. RST instruction is required to reset M1068 otherwise M1068 is latched.

Limitations for times of using instructions

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

1. Instructions can be used once in a program:

API 60 (IST)	API 155 (DABSR)

2. Instruction can be used twice in a program:

   API 77 (PR)

3. Instruction can be used 8 times in a program:

   API 64 (TTMR)

4. For counters C232~C242, the total max times for using DHSCS, DHSCR and DHSZ instructions: 6. DHSZ can only be used less than 6 times.

5. For counters C243, C245~C248, C251, C252, the total max times for using DHSCS, DHSCR and DHSZ instructions: 4. DHSZ takes up 2 times of the total available times.

6. For counters C244, C249, C250, C253, C254, the total max times for using DHSCS, DHSCR and DHSZ instructions: 4. DHSZ takes up 2 times of the total available times.

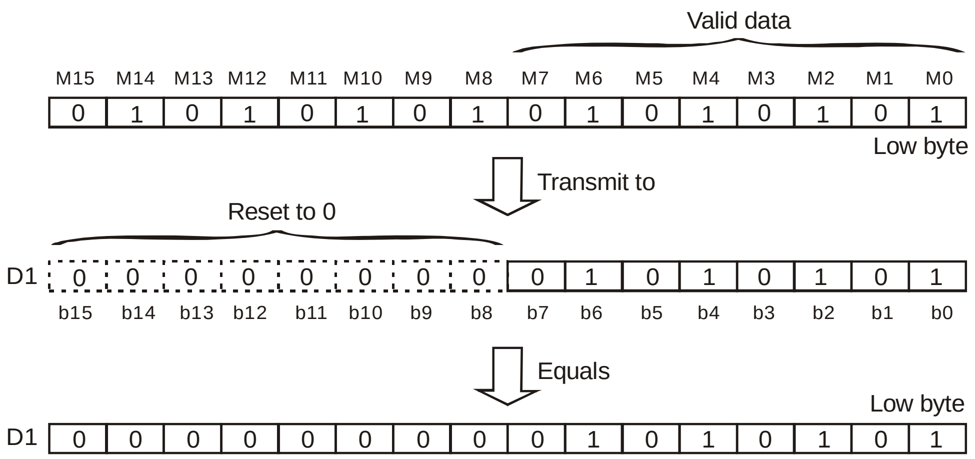
Limitation of synchronized execution

Most instructions have no limitation on the times to be used in a program, but there are limitations on the number of instruction to be executed in the same scan cycle.

1. Only 1 instruction can be executed at the same scan cycle: API 52 MTR, API 69 SORT, API 70 TKY, API 71 HKY, API 72 DSW, API 74 SEGL, API 75 ARWS.
2. Only 4 instruction can be executed at the same scan cycle: API 56 SPD, API 169 HOUR.
3. There is no limitation on the times of using the high-speed output instructions API 57 PLSY, API 58 PWM, API 59 PLSR, API 156DZRN, API 158 DDRVI, API 159 DDRVA and API 195 DPTPO, but only one high-speed output instruction will be executed in the same scan time.
4. There is no limitation on the times of using the communication instructions API 80 RS, API 100 MODRD, API 101 MODWR, API 102 FWD, API 103 REV, API 104 STOP, API 105 RDST, API 106 RSTEF , API 150 MODRW, but only one communication instruction will be executed on single COM port during the same scan cycle.

Numeric Values

1. Devices indicates ON/OFF status are called bit devices, e.g. X, Y, M and S. Devices used for storing values are called word devices, e.g. T, C, D, E and F. Although bit device can only be ON/OFF for a single point, they can also be used as numeric values in the operands of instructions if the data type declaration device Kn is added in front of the bit device.
2. For 16-bit data, K1~K4 are applicable. For 32-bit data, K1~K8 are applicable. For example, K2M0 refers to a 8-bit value composed of M0 ~ M7.



3. Transmit K1M0, K2M0, K3M0 to 16-bit registers. Only the valid bit data will be transmitted and the upper bits in the 16-bit register will all be filled with 0. The same rule applies when sending K1M0, K2M0, K3M0, K4M0, K5M0, K6M0, K7M0 to 32-bit registers.
4. When the Kn value is specified as K1~K3 (K4~K7) for a 16-bit (32-bit) operation, the empty upper bits of the target register will be filled with “0.” Therefore, the operation result in this case is positive since the MSB(Most significant bit) is 0.

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 Kn defines the number of groups of 4 bits to be combined for data operation. For data register D, consecutive D refers to D0, D1, D2, D3, D4…; For bit devices with Kn, consecutive No. refers to:

K1X0	K1X4	K1X10	K1X14…
K2Y0	K2Y10	K2Y20	Y2X30…
K3M0	K3M12	K3M24	K3M36…
K4S0	K4S16	K4S32	K4S48…

Note: To avoid errors, please do not skip over the continuous numbers. In additoin, when 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

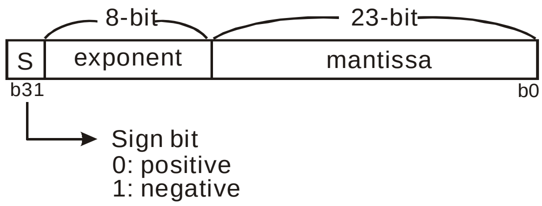
The operations in DVP-PLC are conducted in BIN integers. When the integer performs division, e.g. 40 ÷ 3 = 13, the remainder will be 1. When the integer performs square root operations, the decimal point will be left out. To obtain the operation result with decimal point, please use floating point instructions.

Application instructions revelant to floating point:

FLT	DECMP	DEZCP	DMOVR	DRAD
DDEG	DEBCD	DEBIN	DEADD	DESUB
DEMUL	DEDIV	DEXP	DLN	DLOG
DESQR	DPOW	INT	DSIN	DCOS
DTAN	DASIN	DACOS	DATAN	DADDR
DSUBR	DMULR	DDIVR		

Binary Floating Point

DVP-PLC represents floating point value in 32 bits, following the IEEE754 standard:



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

Therefore, the range of 32-bit floating point value 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: Represent "23" in 32-bit floating point value

Step 1: Convert "23" into a binary value: 23.0 = 10111

Step 2: Normalize the binary value: 10111 = 1.0111 × 2^4, in which 0111 is mantissa and 4 is exponent.

Step 3: Obtain the exponent: ∵ E – B = 4 → E – 127 = 4 ∴ E = 131 = 10000011_2

Step 4: Combine the sign bit, exponent and mantissa into a floating point

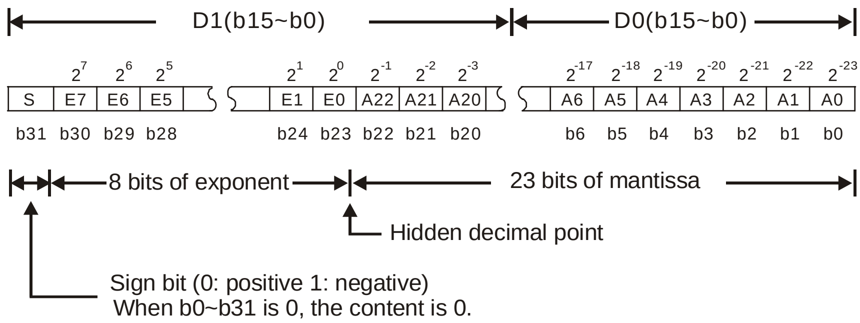
0 100**0001**1 011**1000**0000**0000**0000**0000**$_2$ = $41B80000_{16}$

Example 2: Represent "-23.0" in 32-bit floating point value

The steps required are the same as those in Example 1 and only differs in modifying the sign bit into "1".

1 100**0001**1 011**1000**0000**0000**0000**0000**$_2$=$C1B80000_{16}$

DVP-PLC uses registers of 2 continuous No. to store a 32-bit floating point value. For example, we use registers (D1, D0) for storing a binary floating point value as below:



Decimal Floating Point

- Since the binary floating point value is not very user-friendly, we can convert it into a decimal floating point value for use. However, please note that the floating point operation in DVP-PLC is still operated in binary floating point format.

- The decimal floating point is represented by 2 continuous registers. The register of smaller number is for the constant while the register of bigger number is for the exponent.

  Example: Store a decimal floating point in registers (D1, D0)

  Decimal floating point = [constant D0] $\times 10^{[\text{exponent D1}]}$

  Constant D0 = ±1,000 ~ ±9,999

  Exponent D1 = -41 ~ +35

  The constant 100 does not exist in D0 because 100 is represented as $1{,}000 \times 10^{-1}$. The range of decimal floating point is $\pm 1175 \times 10^{-41} \sim \pm 3402 \times 10^{+35}$.

- The decimal floating point can be used in the following instructions:

  D EBCD: Convert binary floating point to decimal floating point

  D EBIN: Convert decimal floating point to binary floating point

- Zero flag (M1020), borrow flag (M1021), carry flag (M1022) and the floating point operation instruction

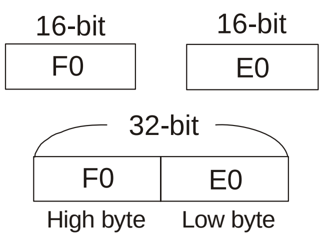
  Zero flag: M1020 = On if the operational result is “0”.

  Borrow flag: M1021 = On if the operational result exceeds the minimum unit.

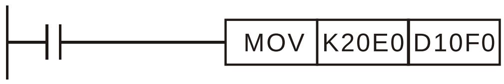
  Carry flag: M1022 = On if the absolute value of the operational result exceeds the range of use.

Index register E, F

The index registers are 16-bit registers. There are 16 devices including E0 ~ E7 and F0 ~ F7.



- E and F index registers are 16-bit data registers which can be read and written.
- If you need a 32-bit register, you have to designate E. In this case, F will be covered up by E and cannot be used; otherwise, the contents in E may become incorrect. (We recommend you use MOVP instruction to reset the contents in D to 0 when the PLC is switched on.)
- Combination of E and F when you designate a 32-bit index register: (E0, F0), (E1, F1), (E2, F2), … (E7, F7)



E0 = 8   F0 = 14

20 + 8 = 28   10 + 14 = 24

Transmission   K28 → D24

The opposite diagram E, F index register modification refers to the content in the operand changes with the contents in E and F.

For example, E0 = 8 and K20E0 represents constant K28 (20 + 8). When the condition is true, constant K28 will be transmitted to register D24.

Devices modifiable: P, X, Y, M, S, KnX, KnY, KnM, KnS, T, C, D.
E and F can modify the devices listed above but cannot modify themselves and Kn., e.g. K4M0E0 is valid and K0E0M0 is invalid. Grey columns in the table of operand at the beginning page of each application instruction indicate the operands modifiable by E and F.

If you need to modify device P, I, X, Y, M, S, KnX, KnY, KnM, KnS, T, C and D by applying E, F, you have to select a 16-bit register, i.e. you can designate E or F.

3.6 Numerical List of Instructions (classified according to the function)

Loop Control

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	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	-	-	The end of the main program (First end)	✓	✓	✓	✓	1	-
07	WDT	-	✓	Watchdog timer refresh	✓	✓	✓	✓	1	-
08	FOR	-	-	Start of a For-Next Loop	✓	✓	✓	✓	3	-
09	NEXT	-	-	End of a For-Next Loop	✓	✓	✓	✓	1	-

Transmission Comparison

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
10	CMP	DCMP	✓	Compare	✓	✓	✓	✓	7	13
11	ZCP	DZCP	✓	Zone compare	✓	✓	✓	✓	9	17
12	MOV	DMOV	✓	Move	✓	✓	✓	✓	5	9
13	SMOV	-	✓	Shift move	✓	✓	✓	✓	11	-
14	CML	DCML	✓	Complement	✓	✓	✓	✓	5	9
15	BMOV	-	✓	Block move	✓	✓	✓	✓	7	-
16	FMOV	DFMOV	✓	Fill move	✓	✓	✓	✓	7	13
17	XCH	DXCH	✓	Exchange	✓	✓	✓	✓	5	9
18	BCD	DBCD	✓	Convert BIN to BCD	✓	✓	✓	✓	5	9
19	BIN	DBIN	✓	Convert BCD to BIN	✓	✓	✓	✓	5	9

Four Arithmetic Operations

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
20	ADD	DADD	✓	Addition	✓	✓	✓	✓	7	13
21	SUB	DSUB	✓	Subtraction	✓	✓	✓	✓	7	13
22	MUL	DMUL	✓	Multiplication	✓	✓	✓	✓	7	13

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
23	DIV	DDIV	✓	Division	✓	✓	✓	✓	7	13
24	INC	DINC	✓	Increment	✓	✓	✓	✓	3	5
25	DEC	DDEC	✓	Decrement	✓	✓	✓	✓	3	5
26	WAND	DAND	✓	Logical Word AND	✓	✓	✓	✓	7	13
27	WOR	DOR	✓	Logical Word OR	✓	✓	✓	✓	7	13
28	WXOR	DXOR	✓	Logical XOR	✓	✓	✓	✓	7	13
29	NEG	DNEG	✓	2's Complement (Negation)	✓	✓	✓	✓	3	5

Rotation and Displacement

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
30	ROR	DROR	✓	Rotate right	✓	✓	✓	✓	5	9
31	ROL	DROL	✓	Rotate 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	-

Data Processing

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
40	ZRST	-	✓	Zone reset	✓	✓	✓	✓	5	-
41	DECO	-	✓	Decode	✓	✓	✓	✓	7	-
42	ENCO	-	✓	Encode	✓	✓	✓	✓	7	-
43	SUM	DSUM	✓	Sum of Active bits	✓	✓	✓	✓	5	9
44	BON	DBON	✓	Check specified bit status	✓	✓	✓	✓	7	13
45	MEAN	DMEAN	✓	Mean	✓	✓	✓	✓	7	13
46	ANS	-	-	Timed Annunciator Set	✓	✓	✓	✓	7	-
47	ANR	-	✓	Annunciator Reset	✓	✓	✓	✓	1	-
48	SQR	DSQR	✓	Square Root	✓	✓	✓	✓	5	9
49	FLT	DFLT	✓	Floating point	✓	✓	✓	✓	5	9

High Speed Processing

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
50	REF	-	✓	Refresh	✓	✓	✓	✓	5	-
51	REFF	-	✓	Refresh and filter adjust	✓	✓	✓	✓	3	-
52	MTR	-	-	Input Matrix	✓	✓	✓	✓	9	-
53	-	DHSCS	-	High speed counter SET	✓	✓	✓	✓	-	13
54	-	DHSCR	-	High speed counter RESET	✓	✓	✓	✓	-	13
55	-	DHSZ	-	High speed zone compare	✓	✓	✓	✓	-	17
56	SPD	-	-	Speed detection	✓	✓	✓	✓	7	-
57	PLSY	DPLSY	-	Pulse output	✓	✓	✓	✓	7	13
58	PWM	-	-	Pulse width modulation	✓	✓	✓	✓	7	-
59	PLSR	DPLSR	-	Pulse ramp	✓	✓	✓	✓	9	17

Handy Instructions

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
60	IST	-	-	Initial state	✓	✓	✓	✓	7	-
61	SER	DSER	✓	Search a data stack	-	✓	✓	✓	9	17
62	ABSD	DABSD	-	Absolute drum sequencer	-	✓	✓	✓	9	17
63	INCD	-	-	Incremental drum sequencer	-	✓	✓	✓	9	-
64	TTMR	-	-	Teaching timer	-	✓	✓	✓	5	-
65	STMR	-	-	Special timer	-	✓	✓	✓	7	-
66	ALT	-	✓	Alternate state	✓	✓	✓	✓	3	-
67	RAMP	DRAMP	-	Ramp variable value	-	✓	✓	✓	9	17
68	DTM	-	✓	Data transform and move	-	✓	✓	✓	9	-
69	SORT	DSORT	-	Data sort	-	✓	✓	✓	11	21

External I/O Display

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
70	TKY	DTKY	-	10-key input	-	✓	✓	✓	7	13
71	HKY	DHKY	-	Hexadecimal key input	-	✓	✓	✓	9	17
72	DSW	-	-	DIP Switch	-	✓	✓	✓	9	-
73	SEGD	-	✓	7-segment decoder	✓	✓	✓	✓	5	-
74	SEGL	-	-	7-segment with latch	✓	✓	✓	✓	7	-
75	ARWS	-	-	Arrow switch	-	✓	✓	✓	9	-

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
76	ASC	-	-	ASCII code conversion	-	✓	✓	✓	11	-
77	PR	-	-	Print (ASCII code output)	-	✓	✓	✓	5	-

Serial I/O

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
78	FROM	DFROM	✓	Read CR data from special modules	✓	✓	✓	✓	9	17
79	TO	DTO	✓	Write CR data into special modules	✓	✓	✓	✓	9	17
80	RS	-	-	Serial communication	✓	✓	✓	✓	9	-
81	PRUN	DPRUN	✓	Parallel run	-	✓	✓	✓	5	9
82	ASCII	-	✓	Convert HEX to ASCII	✓	✓	✓	✓	7	-
83	HEX	-	✓	Convert ASCII to HEX	✓	✓	✓	✓	7	-
84	CCD	-	✓	Check code	-	✓	✓	✓	7	-
85	VRRD	-	✓	Volume read	-	-	✓	✓	5	-
86	VRSC	-	✓	Volume scale read	-	-	✓	✓	5	-
87	ABS	DABS	✓	Absolute value	✓	✓	✓	✓	3	5
88	PID	DPID	-	PID control	✓	✓	✓	✓	9	17

Basic Instructions

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
89	PLS	-	-	Rising-edge output	✓	✓	✓	✓	3	-
90	LDP	-	-	Rising–edge detection operation	✓	✓	✓	✓	3	-
91	LDF	-	-	Falling–edge detection operation	✓	✓	✓	✓	3	-
92	ANDP	-	-	Rising-edge series connection	✓	✓	✓	✓	3	-
93	ANDF	-	-	Falling-edge series connection	✓	✓	✓	✓	3	-
94	ORP	-	-	Rising-edge parallel connection	✓	✓	✓	✓	3	-
95	ORF	-	-	Falling-edge parallel connection	✓	✓	✓	✓	3	-
96	TMR	-	-	Timer	✓	✓	✓	✓	4	-
97	CNT	DCNT	-	Counter	✓	✓	✓	✓	4	6
98	INV	-	-	Inverse operation	✓	✓	✓	✓	1	-

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
99	PLF	-	-	Falling-edge output	✓	✓	✓	✓	3	-

Communication Instructions

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
100	MODRD	-	-	Read Modbus data	✓	✓	✓	✓	7	-
101	MODWR	-	-	Write Modbus Data	✓	✓	✓	✓	7	-
102	FWD	-	-	Forward Operation of VFD	✓	✓	✓	✓	7	–
103	REV	-	-	Reverse Operation of VFD	✓	✓	✓	✓	7	–
104	STOP	-	-	Stop VFD	✓	✓	✓	✓	7	–
105	RDST	-	-	Read VFD Status	✓	✓	✓	✓	5	–
106	RSTEF	-	-	Reset Abnormal VFD	✓	✓	✓	✓	5	–
107	LRC	-	✓	LRC checksum	✓	✓	✓	✓	7	-
108	CRC	-	✓	CRC checksum	✓	✓	✓	✓	7	-
150	MODRW	-	-	MODBUS Read/ Write	✓	✓	✓	✓	11	-
206	ASDRW	-	-	ASDA servo drive R/W	-	✓	✓	✓	7	-

Floating Point Operation

API	Mnemonics		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
110	-	DECMP	✓	Floating point compare	✓	✓	✓	✓	-	13
111	-	DEZCP	✓	Floating point zone compare	✓	✓	✓	✓	-	17
112		DMOVR	✓	Move floating point data	✓	✓	✓	✓		9
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	✓	Float exponent operation	✓	✓	✓	✓	-	9
125	-	DLN	✓	Float natural logarithm operation	✓	✓	✓	✓	-	9
126	-	DLOG	✓	Float logarithm operation	✓	✓	✓	✓	-	13
127	-	DESQR	✓	Floating point square root	✓	✓	✓	✓	-	9

API	Mnemonics		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
128	-	DPOW	✓	Floating point power operation	✓	✓	✓	✓	-	13
129	INT	DINT	✓	Float to integer	✓	✓	✓	✓	5	9
130	-	DSIN	✓	Sine	✓	✓	✓	✓	-	9
131	-	DCOS	✓	Cosine	✓	✓	✓	✓	-	9
132	-	DTAN	✓	Tangent	✓	✓	✓	✓	-	9
133	-	DASIN	✓	Arc Sine	✓	✓	✓	✓	-	9
134	-	DACOS	✓	Arc Cosine	✓	✓	✓	✓	-	9
135	-	DATAN	✓	Arc Tangent	✓	✓	✓	✓	-	9
172	-	DADDR	✓	Floating point addition	✓	✓	✓	✓	-	13
173	-	DSUBR	✓	Floating point subtraction	✓	✓	✓	✓	-	13
174	-	DMULR	✓	Floating point multiplication	✓	✓	✓	✓	-	13
175	-	DDIVR	✓	Floating point division	✓	✓	✓	✓	-	13

Additional Instruction

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
143	DELAY	-	✓	Delay	✓	✓	✓	✓	3	-
144	GPWM	-	-	General PWM output	✓	✓	✓	✓	7	-
147	SWAP	DSWAP	✓	Byte swap	✓	✓	✓	✓	3	5
154	RAND	DRAND	✓	Random number	✓	✓	✓	✓	7	13
168	MVM	DMVM	✓	Mask and combine designated Bits	✓	✓	✓	✓	7	13
176	MMOV	–	✓	16-bit→32-bit Conversion	✓	✓	✓	✓	5	–
177	GPS	-	-	GPS data receiving	✓	✓	✓	✓	5	-
178	-	DSPA	-	Solar cell positioning	✓	✓	✓	✓	–	9
179	WSUM	DWSUM	✓	Sum of multiple devices	✓	✓	✓	✓	7	13
202	SCAL	-	✓	Proportional value calculation	✓	✓	✓	✓	9	-
203	SCLP	DSCLP	✓	Parameter proportional value calculation	✓	✓	✓	✓	9	13
205	CMPT	-	✓	Compare table	✓	✓	✓	✓	9	-
207	CSFO	-	-	Catch speed and proportional output	✓	✓	✓	✓	7	-

Positioning Control

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
155	-	DABSR	-	Absolute position read	✓	✓	✓	✓	-	13
156	-	DZRN	-	Zero return	✓	✓	✓	✓	-	17
157	-	DPLSV		Adjustable speed pulse output	✓	✓	✓	✓	-	13
158	-	DDRVI	-	Relative position control	✓	✓	✓	✓	-	17
159	-	DDRVA	-	Absolute position control	✓	✓	✓	✓	-	17
191	-	DPPMR	-	2-Axis Relative Point to Point Motion	✓	-	✓	✓	-	17
192	-	DPPMA	-	2-Axis Absolute Point to Point Motion	✓	-	✓	✓	-	17
193	-	DCIMR	-	2-Axis Relative Position Arc Interpolation	✓	-	✓	✓	-	17
194	-	DCIMA	-	2-Axis Absolute Position Arc Interpolation	✓	-	✓	✓	-	17
195	-	DPTPO	-	Single-Axis pulse output by table	✓	✓	✓	✓	-	13
197	-	DCLLM	-	Close loop position control	✓	✓	✓	✓	-	17
198	-	DVSPO	-	Variable speed pulse output	✓	✓	✓	✓	-	17
199	-	DICF	✓	Immediately change frequency	✓	✓	✓	✓	-	13

Real Time Calendar

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
160	TCMP	-	✓	Time compare	✓	✓	✓	✓	11	-
161	TZCP	-	✓	Time Zone Compare	✓	✓	✓	✓	9	-
162	TADD	-	✓	Time addition	✓	✓	✓	✓	7	-
163	TSUB	-	✓	Time subtraction	✓	✓	✓	✓	7	-
166	TRD	-	✓	Time read	✓	✓	✓	✓	3	-
167	TWR	-	✓	Time write	✓	✓	✓	✓	3	-
169	HOUR	DHOUR	-	Hour meter	✓	✓	✓	✓	7	13

Gray Code

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
170	GRY	DGRY	✓	BIN → Gray Code	✓	✓	✓	✓	5	9

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
171	GBIN	DGBIN	✓	Gray Code → BIN	✓	✓	✓	✓	5	9

Matrix Operation

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	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 status count	✓	✓	✓	✓	7	-

Contact Type Logic Operation

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
215	LD&	DLD&	-	S_1 & S_2	✓	✓	✓	✓	5	9
216	LD|	DLD|	-	$S_1 \vert S_2$	✓	✓	✓	✓	5	9
217	LD^	DLD^	-	S_1 ^ S_2	✓	✓	✓	✓	5	9
218	AND&	DAND&	-	S_1 & S_2	✓	✓	✓	✓	5	9
219	AND|	DAND|	-	$S_1 \vert S_2$	✓	✓	✓	✓	5	9
220	AND^	DAND^	-	S_1 ^ S_2	✓	✓	✓	✓	5	9
221	OR&	DOR&	-	S_1 & S_2	✓	✓	✓	✓	5	9
222	OR|	DOR|	-	$S_1 \vert S_2$	✓	✓	✓	✓	5	9
223	OR^	DOR^	-	S_1 ^ S_2	✓	✓	✓	✓	5	9

Contact Type Comparison

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
224	LD=	DLD=	-	$S_1 = S_2$	✓	✓	✓	✓	5	9
225	LD>	DLD>	-	$S_1 > S_2$	✓	✓	✓	✓	5	9

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
226	LD<	DLD<	-	$S_1 < S_2$	✓	✓	✓	✓	5	9
228	LD<>	DLD<>	-	$S_1 \neq S_2$	✓	✓	✓	✓	5	9
229	LD<=	DLD<=	-	$S_1 \leqq S_2$	✓	✓	✓	✓	5	9
230	LD>=	DLD>=	-	$S_1 \geqq S_2$	✓	✓	✓	✓	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<=	-	$S_1 \leqq S_2$	✓	✓	✓	✓	5	9
238	AND>=	DAND>=	-	$S_1 \geqq S_2$	✓	✓	✓	✓	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<=	-	$S_1 \leqq S_2$	✓	✓	✓	✓	5	9
246	OR>=	DOR>=	-	$S_1 \geqq S_2$	✓	✓	✓	✓	5	9

Specific Bit Control

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
266	BOUT	DBOUT	-	Output specified bit of a word	✓	✓	✓	✓	5	9
267	BSET	DBSET	-	Set ON specified bit of a word	✓	✓	✓	✓	5	9
268	BRST	DBRST	-	Reset specified bit of a word	✓	✓	✓	✓	5	9
269	BLD	DBLD	-	Load NO contact by specified bit	✓	✓	✓	✓	5	9
270	BLDI	DBLDI	-	Load NC contact by specified bit	✓	✓	✓	✓	5	9
271	BAND	DBAND	-	Connect NO contact in series by specified bit	✓	✓	✓	✓	5	9
272	BANI	DBANI	-	Connect NC contact in series by specified bit	✓	✓	✓	✓	5	9
273	BOR	DBOR	-	Connect NO contact in parallel by specified bit	✓	✓	✓	✓	5	9
274	BORI	DBORI	-	Connect NC contact in parallel by specified bit	✓	✓	✓	✓	5	9

Floating-Point Contact Type Comparison

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
275	-	FLD=	-	$S_1 = S_2$	✓	✓	✓	✓	-	9
276	-	FLD>	-	$S_1 > S_2$	✓	✓	✓	✓	-	9
277	-	FLD<	-	$S_1 < S_2$	✓	✓	✓	✓	-	9
278	-	FLD<>	-	$S_1 \neq S_2$	✓	✓	✓	✓	-	9
279	-	FLD<=	-	$S_1 \leqq S_2$	✓	✓	✓	✓	-	9
280	-	FLD>=	-	$S_1 \geqq S_2$	✓	✓	✓	✓	-	9
280	-	FAND=	-	$S_1 = S_2$	✓	✓	✓	✓	-	9
282	-	FAND>	-	$S_1 > S_2$	✓	✓	✓	✓	-	9
283	-	FAND<	-	$S_1 < S_2$	✓	✓	✓	✓	-	9
284	-	FAND<>	-	$S_1 \neq S_2$	✓	✓	✓	✓	-	9
285	-	FAND<=	-	$S_1 \leqq S_2$	✓	✓	✓	✓	-	9
286	-	FAND>=	-	$S_1 \geqq S_2$	✓	✓	✓	✓	-	9
287	-	FOR=	-	$S_1 = S_2$	✓	✓	✓	✓	-	9
288	-	FOR>	-	$S_1 > S_2$	✓	✓	✓	✓	-	9
289	-	FOR<	-	$S_1 < S_2$	✓	✓	✓	✓	-	9
290	-	FOR<>	-	$S_1 \neq S_2$	✓	✓	✓	✓	-	9
291	-	FOR<=	-	$S_1 \leqq S_2$	✓	✓	✓	✓	-	9
292	-	FOR>=	-	$S_1 \geqq S_2$	✓	✓	✓	✓	-	9

3.7 Numerical List of Instructions (in alphabetic order)

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
87	ABS	DABS	✓	Absolute value	✓	✓	✓	✓	3	5
62	ABSD	DABSD	-	Absolute drum sequencer	-	✓	✓	✓	9	17
20	ADD	DADD	✓	Addition	✓	✓	✓	✓	7	13
66	ALT	-	✓	Alternate state	✓	✓	✓	✓	3	-
218	AND&	DAND&	-	S_1 & S_2	✓	✓	✓	✓	5	9
220	AND^	DAND^	-	S_1 ^ S_2	✓	✓	✓	✓	5	9
219	AND|	DAND|	-	S1 | S2	✓	✓	✓	✓	5	9
234	AND<	DAND<	-	S1 < S2	✓	✓	✓	✓	5	9
237	AND<=	DAND<=	-	S1 ≦ S2	✓	✓	✓	✓	5	9
236	AND<>	DAND<>	-	S1 ≠ S2	✓	✓	✓	✓	5	9

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
232	AND=	DAND=	-	S1 = S2	✓	✓	✓	✓	5	9
233	AND>	DAND>	-	S1 > S2	✓	✓	✓	✓	5	9
238	AND>=	DAND>=	-	S1 ≧ S2	✓	✓	✓	✓	5	9
93	ANDF	-	-	Falling-edge series connection	✓	✓	✓	✓	3	-
92	ANDP	-	-	Rising-edge series connection	✓	✓	✓	✓	3	-
47	ANR	-	✓	Annunciator Reset	✓	✓	✓	✓	1	-
46	ANS	-	-	Timed Annunciator Set	✓	✓	✓	✓	7	-
75	ARWS	-	-	Arrow switch	-	✓	✓	✓	9	-
76	ASC	-	-	ASCII code conversion	-	✓	✓	✓	11	-
82	ASCII	-	✓	Convert HEX to ASCII	✓	✓	✓	✓	7	-
206	ASDRW	-	-	ASDA servo drive R/W	✓	✓	✓	✓	7	-
271	BAND	DBAND	-	Connect NO contact in series by specified bit	✓	✓	✓	✓	5	9
272	BANI	DBANI	-	Connect NC contact in series by specified bit	✓	✓	✓	✓	5	9
18	BCD	DBCD	✓	Convert BIN to BCD	✓	✓	✓	✓	5	9
19	BIN	DBIN	✓	Convert BCD to BIN	✓	✓	✓	✓	5	9
269	BLD	DBLD	-	Load NO contact by specified bit	✓	✓	✓	✓	5	9
270	BLDI	DBLDI	-	Load NC contact by specified bit	✓	✓	✓	✓	5	9
15	BMOV	-	✓	Block move	✓	✓	✓	✓	7	-
44	BON	DBON	✓	Check specified bit status	✓	✓	✓	✓	7	13
273	BOR	DBOR	-	Connect NO contact in parallel by specified bit	✓	✓	✓	✓	5	9
274	BORI	DBORI	-	Connect NC contact in parallel by specified bit	✓	✓	✓	✓	5	9
266	BOUT	DBOUT	-	Output specified bit of a word	✓	✓	✓	✓	5	9
268	BRST	DBRST	-	Reset specified bit of a word	✓	✓	✓	✓	5	9
267	BSET	DBSET	-	Set ON specified bit of a word	✓	✓	✓	✓	5	9
01	CALL	-	✓	Call subroutine	✓	✓	✓	✓	3	-
84	CCD	-	✓	Check code	-	✓	✓	✓	7	-
00	CJ	-	✓	Conditional jump	✓	✓	✓	✓	3	-
14	CML	DCML	✓	Complement	✓	✓	✓	✓	5	9
10	CMP	DCMP	✓	Compare	✓	✓	✓	✓	7	13
205	CMPT	-	✓	Compare table	✓	✓	✓	✓	9	-

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
97	CNT	DCNT	-	Counter	✓	✓	✓	✓	4	6
108	CRC	-	✓	CRC checksum	✓	✓	✓	✓	7	-
207	CSFO	-	-	Catch speed and proportional output	✓	✓	✓	✓	7	-
25	DEC	DDEC	✓	Decrement	✓	✓	✓	✓	3	5
41	DECO	-	✓	Decode	✓	✓	✓	✓	7	-
143	DELAY	-	✓	Delay	✓	✓	✓	✓	3	-
05	DI	-	-	Disable interrupt	✓	✓	✓	✓	1	-
23	DIV	DDIV	✓	Division	✓	✓	✓	✓	7	13
72	DSW	-	-	DIP Switch	-	✓	✓	✓	9	-
68	DTM	-	✓	Data transform and move	-	✓	✓	✓	9	-
04	EI	-	-	Enable interrupt	✓	✓	✓	✓	1	-
42	ENCO	-	✓	Encode	✓	✓	✓	✓	7	-
06	FEND	-	-	The end of the main program (First end)	✓	✓	✓	✓	1	-
49	FLT	DFLT	✓	Floating point	✓	✓	✓	✓	5	9
16	FMOV	DFMOV	✓	Fill move	✓	✓	✓	✓	7	13
08	FOR	-	-	Start of a For-Next Loop	✓	✓	✓	✓	3	-
78	FROM	DFROM	✓	Read CR data from special modules	✓	✓	✓	✓	9	17
102	FWD	-	-	Forward Operation of VFD	✓	✓	✓	✓	7	–
171	GBIN	DGBIN	✓	Gray Code → BIN	✓	✓	✓	✓	5	9
177	GPS	-	-	GPS data receiving	✓	✓	✓	✓	5	-
144	GPWM	-	-	General PWM output	✓	✓	✓	✓	7	-
170	GRY	DGRY	✓	BIN → Gray Code	✓	✓	✓	✓	5	9
83	HEX	-	✓	Convert ASCII to HEX	✓	✓	✓	✓	7	-
71	HKY	DHKY	-	Hexadecimal key input	-	✓	✓	✓	9	17
169	HOUR	DHOUR	-	Hour meter	✓	✓	✓	✓	7	13
24	INC	DINC	✓	Increment	✓	✓	✓	✓	3	5
63	INCD	-	-	Incremental drum sequencer	-	✓	✓	✓	9	-
129	INT	DINT	✓	Float to integer	✓	✓	✓	✓	5	9
98	INV	-	-	Inverse operation	✓	✓	✓	✓	1	-
03	IRET	-	-	Interrupt return	✓	✓	✓	✓	1	-
60	IST	-	-	Initial state	✓	✓	✓	✓	7	-

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
215	LD&	DLD&	-	S1 & S2	✓	✓	✓	✓	5	9
217	LD^	DLD^	-	S1 ^ S2	✓	✓	✓	✓	5	9
216	LD|	DLD|	-	S1 | S2	✓	✓	✓	✓	5	9
226	LD<	DLD<	-	S1 < S2	✓	✓	✓	✓	5	9
229	LD<=	DLD<=	-	S1 ≦ S2	✓	✓	✓	✓	5	9
228	LD<>	DLD<>	-	S1 ≠ S2	✓	✓	✓	✓	5	9
224	LD=	DLD=	-	S1 = S2	✓	✓	✓	✓	5	9
225	LD>	DLD>	-	S1 > S2	✓	✓	✓	✓	5	9
230	LD>=	DLD>=	-	S1 ≧ S2	✓	✓	✓	✓	5	9
91	LDF	-	-	Falling–edge detection operation	✓	✓	✓	✓	3	-
90	LDP	-	-	Rising–edge detection operation	✓	✓	✓	✓	3	-
107	LRC	-	✓	LRC checksum	✓	✓	✓	✓	7	-
180	MAND	-	✓	Matrix AND	✓	✓	✓	✓	9	-
190	MBC	-	✓	Matrix bit status count	✓	✓	✓	✓	7	-
189	MBR	-	✓	Matrix bit rotate	✓	✓	✓	✓	7	-
186	MBRD	-	✓	Matrix bit read	✓	✓	✓	✓	7	-
188	MBS	-	✓	Matrix bit shift	✓	✓	✓	✓	7	-
187	MBWR	-	✓	Matrix bit write	✓	✓	✓	✓	7	-
185	MCMP	-	✓	Matrix compare	✓	✓	✓	✓	9	-
45	MEAN	DMEAN	✓	Mean	✓	✓	✓	✓	7	13
184	MINV	-	✓	Matrix inverse	✓	✓	✓	✓	7	-
176	MMOV	-	✓	16-bit→32-bit Conversion	✓	✓	✓	✓	5	-
100	MODRD	-	-	Read Modbus data	✓	✓	✓	✓	7	-
150	MODRW	-	-	MODBUS Read/ Write	✓	✓	✓	✓	11	-
101	MODWR	-	-	Write Modbus Data	✓	✓	✓	✓	7	-
181	MOR	-	✓	Matrix OR	✓	✓	✓	✓	9	-
12	MOV	DMOV	✓	Move	✓	✓	✓	✓	5	9
52	MTR	-	-	Input Matrix	✓	✓	✓	✓	9	-
22	MUL	DMUL	✓	Multiplication	✓	✓	✓	✓	7	13
168	MVM	DMVM	✓	Mask and combine designated Bits	✓	✓	✓	✓	7	13
183	MXNR	-	✓	Matrix XNR	✓	✓	✓	✓	9	-

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
182	MXOR	-	✓	Matrix XOR	✓	✓	✓	✓	9	-
29	NEG	DNEG	✓	2's Complement (Negation)	✓	✓	✓	✓	3	5
09	NEXT	-	-	End of a For-Next Loop	✓	✓	✓	✓	1	-
221	OR&	DOR&	-	S1 & S2	✓	✓	✓	✓	5	9
223	OR^	DOR^	-	S1 ^ S2	✓	✓	✓	✓	5	9
222	OR|	DOR|	-	S1 | S2	✓	✓	✓	✓	5	9
242	OR<	DOR<	-	S1 < S2	✓	✓	✓	✓	5	9
245	OR<=	DOR<=	-	S1 ≦ S2	✓	✓	✓	✓	5	9
244	OR<>	DOR<>	-	S1 ≠ S2	✓	✓	✓	✓	5	9
240	OR=	DOR=	-	S1 = S2	✓	✓	✓	✓	5	9
241	OR>	DOR>	-	S1 > S2	✓	✓	✓	✓	5	9
246	OR>=	DOR>=	-	S1 ≧ S2	✓	✓	✓	✓	5	9
95	ORF	-	-	Falling-edge parallel connection	✓	✓	✓	✓	3	-
94	ORP	-	-	Rising-edge parallel connection	✓	✓	✓	✓	3	-
88	PID	DPID	-	PID control	✓	✓	✓	✓	9	17
99	PLF	-	-	Falling-edge output	✓	✓	✓	✓	3	-
89	PLS	-	-	Rising-edge output	✓	✓	✓	✓	3	-
59	PLSR	DPLSR	-	Pulse ramp	✓	✓	✓	✓	9	17
57	PLSY	DPLSY	-	Pulse output	✓	✓	✓	✓	7	13
77	PR	-	-	Print (ASCII code output)	-	✓	✓	✓	5	-
81	PRUN	DPRUN	✓	Parallel run	-	✓	✓	✓	5	9
58	PWM	-	-	Pulse width modulation	✓	✓	✓	✓	7	-
67	RAMP	DRAMP	-	Ramp variable value	-	✓	✓	✓	9	17
154	RAND	DRAND	✓	Random number	✓	✓	✓	✓	7	13
33	RCL	DRCL	✓	Rotate left with carry	✓	✓	✓	✓	5	9
32	RCR	DRCR	✓	Rotate right with carry	✓	✓	✓	✓	5	9
105	RDST	-	-	Read VFD Status	✓	✓	✓	✓	5	–
50	REF	-	✓	Refresh	✓	✓	✓	✓	5	-
51	REFF	-	✓	Refresh and filter adjust	✓	✓	✓	✓	3	-
103	REV	-	-	Reverse Operation of VFD	✓	✓	✓	✓	7	–
31	ROL	DROL	✓	Rotate left	✓	✓	✓	✓	5	9
30	ROR	DROR	✓	Rotate right	✓	✓	✓	✓	5	9
80	RS	-	-	Serial communication	✓	✓	✓	✓	9	-
106	RSTEF	-	-	Reset Abnormal VFD	✓	✓	✓	✓	5	–

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
202	SCAL	-	✓	Proportional value calculation	✓	✓	✓	✓	9	-
203	SCLP	DSCLP	✓	Parameter proportional value calculation	✓	✓	✓	✓	7	13
73	SEGD	-	✓	7-segment decoder	✓	✓	✓	✓	5	-
74	SEGL	-	-	7-segment with latch	✓	✓	✓	✓	7	-
61	SER	DSER	✓	Search a data stack	-	✓	✓	✓	9	17
39	SFRD	-	✓	Shift register read	✓	✓	✓	✓	7	-
35	SFTL	-	✓	Bit shift left	✓	✓	✓	✓	9	-
34	SFTR	-	✓	Bit shift right	✓	✓	✓	✓	9	-
38	SFWR	-	✓	Shift register write	✓	✓	✓	✓	7	-
13	SMOV	-	✓	Shift move	✓	✓	✓	✓	11	-
69	SORT	DSORT	-	Data sort	-	✓	✓	✓	11	21
56	SPD	-	-	Speed detection	✓	✓	✓	✓	7	-
48	SQR	DSQR	✓	Square Root	✓	✓	✓	✓	5	9
02	SRET	-	-	Subroutine return	✓	✓	✓	✓	1	-
65	STMR	-	-	Special timer	-	✓	✓	✓	7	-
104	STOP	-	-	Stop VFD	✓	✓	✓	✓	7	–
21	SUB	DSUB	✓	Subtraction	✓	✓	✓	✓	7	13
43	SUM	DSUM	✓	Sum of Active bits	✓	✓	✓	✓	5	9
147	SWAP	DSWAP	✓	Byte swap	✓	✓	✓	✓	3	5
162	TADD	-	✓	Time addition	✓	✓	✓	✓	7	-
160	TCMP	-	✓	Time compare	✓	✓	✓	✓	11	-
70	TKY	DTKY	-	10-key input	-	✓	✓	✓	7	13
96	TMR	-	-	Timer	✓	✓	✓	✓	4	-
79	TO	DTO	✓	Write CR data into special modules	✓	✓	✓	✓	9	17
166	TRD	-	✓	Time read	✓	✓	✓	✓	3	-
163	TSUB	-	✓	Time subtraction	✓	✓	✓	✓	7	-
64	TTMR	-	-	Teaching timer	-	✓	✓	✓	5	-
167	TWR	-	✓	Time write	✓	✓	✓	✓	3	-
161	TZCP	-	✓	Time Zone Compare	✓	✓	✓	✓	9	-
85	VRRD	-	✓	Volume read	-	-	✓	✓	5	-
86	VRSC	-	✓	Volume scale read	-	-	✓	✓	5	-
26	WAND	DAND	✓	Logical Word AND	✓	✓	✓	✓	7	13

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
07	WDT	-	✓	Watchdog timer refresh	✓	✓	✓	✓	1	-
27	WOR	DOR	✓	Logical Word OR	✓	✓	✓	✓	7	13
37	WSFL	-	✓	Word shift left	✓	✓	✓	✓	9	-
36	WSFR	-	✓	Word shift right	✓	✓	✓	✓	9	-
179	WSUM	DWSUM	✓	Sum of multiple devices	✓	✓	✓	✓	7	13
28	WXOR	DXOR	✓	Logical XOR	✓	✓	✓	✓	7	13
17	XCH	DXCH	✓	Exchange	✓	✓	✓	✓	5	9
11	ZCP	DZCP	✓	Zone compare	✓	✓	✓	✓	9	17
40	ZRST	-	✓	Zone reset	✓	✓	✓	✓	5	-
155	-	DABSR	-	Absolute position read	✓	✓	✓	✓	-	13
134	-	DACOS	✓	Arc Cosine	✓	✓	✓	✓	-	9
172	-	DADDR	✓	Floating point addition	✓	✓	✓	✓	-	13
133	-	DASIN	✓	Arc Cosine	✓	✓	✓	✓	-	9
135	-	DATAN	✓	Arc Tangent	✓	✓	✓	✓	-	9
194	-	DCIMA	-	2-Axis Absolute Position Arc Interpolation	✓	-	✓	✓	-	17
193	-	DCIMR	-	2-Axis Relative Position Arc Interpolation	✓	-	✓	✓	-	17
197	-	DCLLM	-	Close loop position control	✓	✓	✓	✓	-	17
131	-	DCOS	✓	Cosine	✓	✓	✓	✓	-	9
117	-	DDEG	✓	Radian → Degree	✓	✓	✓	✓	-	9
175	-	DDIVR	✓	Floating point division	✓	✓	✓	✓	-	13
159	-	DDRVA	-	Absolute position control	✓	✓	✓	✓	-	17
158	-	DDRVI	-	Relative position control	✓	✓	✓	✓	-	17
120	-	DEADD	✓	Floating point addition	✓	✓	✓	✓	-	13
118	-	DEBCD	✓	Float to scientific conversion	✓	✓	✓	✓	-	9
119	-	DEBIN	✓	Scientific to float conversion	✓	✓	✓	✓	-	9
110	-	DECMP	✓	Floating point compare	✓	✓	✓	✓	-	13
123	-	DEDIV	✓	Floating point division	✓	✓	✓	✓	-	13
122	-	DEMUL	✓	Floating point multiplication	✓	✓	✓	✓	-	13
127	-	DESQR	✓	Floating point square root	✓	✓	✓	✓	-	9
121	-	DESUB	✓	Floating point subtraction	✓	✓	✓	✓	-	13
124	-	DEXP	✓	Float exponent operation	✓	✓	✓	✓	-	9
111	-	DEZCP	✓	Floating point zone compare	✓	✓	✓	✓	-	17

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
54	-	DHSCR	-	High speed counter RESET	✓	✓	✓	✓	-	13
53	-	DHSCS	-	High speed counter SET	✓	✓	✓	✓	-	13
55	-	DHSZ	-	High speed zone compare	✓	✓	✓	✓	-	17
199	-	DICF	✓	Immediately change frequency	✓	✓	✓	✓	-	13
125	-	DLN	✓	Float natural logarithm operation	✓	✓	✓	✓	-	9
126	-	DLOG	✓	Float logarithm operation	✓	✓	✓	✓	-	13
112	-	DMOVR	✓	Move floating point data	✓	✓	✓	✓	-	9
174	-	DMULR	✓	Floating point multiplication	✓	✓	✓	✓	-	13
157	-	DPLSV	-	Adjustable speed pulse output	✓	✓	✓	✓	-	13
128	-	DPOW	✓	Floating point power operation	✓	✓	✓	✓	-	13
192	-	DPPMA	-	2-Axis Absolute Point to Point Motion	✓	-	✓	✓	-	17
191	-	DPPMR	-	2-Axis Relative Point to Point Motion	✓	-	✓	✓	-	17
195	-	DPTPO	-	Single-Axis pulse output by table	✓	✓	✓	✓	-	13
116	-	DRAD	✓	Degree → Radian	✓	✓	✓	✓	-	9
130	-	DSIN	✓	Sine	✓	✓	✓	✓	-	9
178	-	DSPA	-	Solar cell positioning	✓	✓	✓	✓	–	9
173	-	DSUBR	✓	Floating point subtraction	✓	✓	✓	✓	-	13
132	-	DTAN	✓	Tangent	✓	✓	✓	✓	-	9
198	-	DVSPO	-	Variable speed pulse output	✓	✓	✓	✓	-	17
156	-	DZRN	-	Zero return	✓	✓	✓	✓	-	17
283	-	FAND<	-	S1 < S2	✓	✓	✓	✓	-	9
285	-	FAND<=	-	S1 ≦ S2	✓	✓	✓	✓	-	9
284	-	FAND<>	-	S1 ≠ S2	✓	✓	✓	✓	-	9
280	-	FAND=	-	S1 = S2	✓	✓	✓	✓	-	9
282	-	FAND>	-	S1 > S2	✓	✓	✓	✓	-	9
286	-	FAND>=	-	S1 ≧ S2	✓	✓	✓	✓	-	9
277	-	FLD<	-	S1 < S2	✓	✓	✓	✓	-	9
279	-	FLD<=	-	S1 ≦ S2	✓	✓	✓	✓	-	9
278	-	FLD<>	-	S1 ≠ S2	✓	✓	✓	✓	-	9

API	Mnemonic		PULSE	Function	Applicable to				STEPS	
	16 bits	32 bits			ES2 EX2	SS2	SA2	SX2	16-bit	32-bit
275	-	FLD=	-	S1 = S2	✓	✓	✓	✓	-	9
276	-	FLD>	-	S1 > S2	✓	✓	✓	✓	-	9
280	-	FLD>=	-	S1 ≧ S2	✓	✓	✓	✓	-	9
289	-	FOR<	-	S1 < S2	✓	✓	✓	✓	-	9
291	-	FOR<=	-	S1 ≦ S2	✓	✓	✓	✓	-	9
290	-	FOR<>	-	S1 ≠ S2	✓	✓	✓	✓	-	9
287	-	FOR=	-	S1 = S2	✓	✓	✓	✓	-	9
288	-	FOR>	-	S1 > S2	✓	✓	✓	✓	-	9
292	-	FOR>=	-	S1 ≧ S2	✓	✓	✓	✓	-	9

3.8 Detailed Instruction Explanation

API	Mnemonic			Operands	Function	Controllers
00		CJ	P	S	Conditional Jump	ES2/EX2 SS2 SA2 SX2

OP	Range	Program Steps
S	P0~P255	CJ, CJP: 3 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: The destination pointer P of the conditional jump.

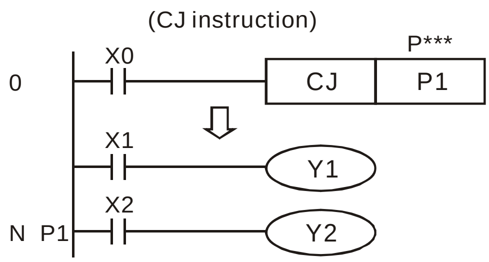
Explanations:

1. If users need to skip a particular part of PLC program in order to shorten the scan time and execute dual outputs, CJ instruction or CJP instruction can be adopted.
2. When the program designated by pointer P is prior to CJ instruction, WDT timeout will occur and PLC will stop running. Please use it carefully.
3. CJ instruction can designate the same pointer P repeatedly. However, CJ and CALL cannot designate the same pointer P; otherwise operation error will occur
4. Actions of all devices while conditional jump is being executed:
   a) Y, M and S remain their previous status before the conditional jump takes place.
   b) 10ms and 100ms timer that is executing stops.
   c) Timer T192 ~ T199 that execute the subroutine program will continue and the output contact executes normally.
   d) The high-speed counter that is executing the counting continues counting and the output contact executes normally.
   e) General counters stop executing.
   f) If timer is reset before CJ instruction executes, the timer will still be in the reset status while CJ instruction is being executed.
   g) General application instructions are not executed.
   h) The application instructions that are being executed, i.e. DHSCS, DHSCR, DHSZ, SPD, PLSY, PWM, PLSR, PLSV, DRVI, DRVA, continue being executed.

Program example 1:

When X0 = ON, the program will skip from address 0 to N (Pointer P1) automatically and keep on executing. Instructions between address 0 and N will be skipped..

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



Program example 2:

The table explains the device status in the ladder diagram below.

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 [*1], M20, S1 OFF
	M1, M2, M3 ON	M1, M2, M3 ON→OFF	Y1 [*1], M20, S1 ON
10ms, 100ms Timer[*2]	M4 OFF	M4 OFF→ON	Timer is not activated
	M4 ON	M4 ON→OFF	Timer T0 immediately stops and is latched. When M0 ON → OFF, T0 will be reset.
1ms,10ms, 100ms accumulative Timer	M6 OFF	M6 OFF→ON	Timer T240 is not activated
	M6 ON	M6 ON→OFF	Timer T240 immediately stops and is latched. When M0 ON → OFF, T240 will still be latched.
C0~C234 [*3]	M7, M10 OFF	M10 is ON/OFF triggered	Counter C0 stops
	M7 OFF, M10 is ON/OFF triggered	M10 is ON/OFF triggered	Counter C0 stops and latched. When M0 is OFF, C0 resumes counting.
Application instruction	M11 OFF	M11 OFF→ON	Application instructions will not be executed.

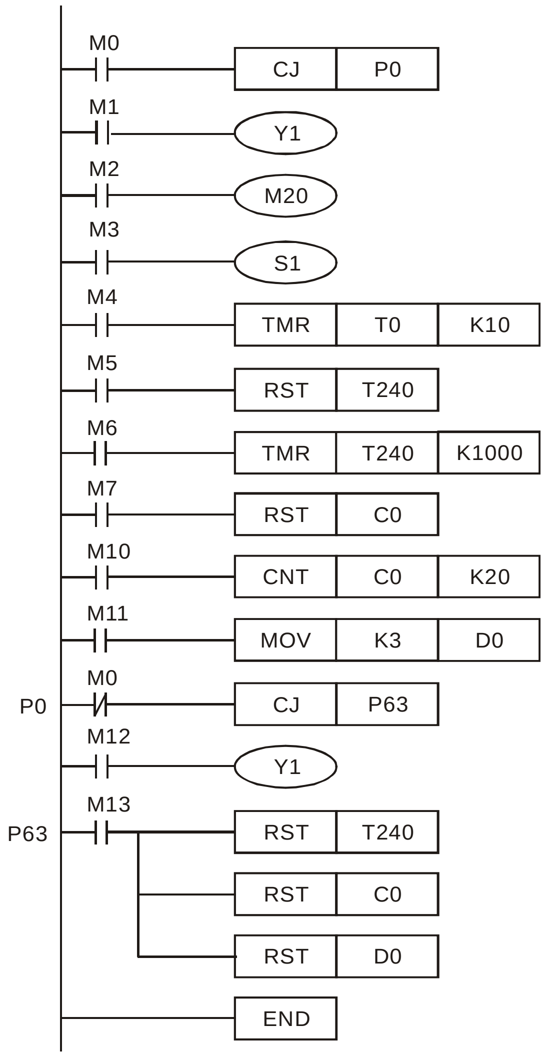
Device	Contact state before CJ execution	Contact state during CJ execution	Output coil state during CJ execution
	M11 ON	M11 ON→OFF	The skipped application instruction will not be executed but API 53~59, API 157~159 keep executing.

*1: Y1 is dual output. When M0 is OFF, it is controlled by M1. When M0 is ON, M12 will control Y1

*2: When timer that subroutine used (T184~T199) executes first and then CJ instruction is executed, the timer will keep counting. After the timer reaches the set value, output contact of timer will be ON.

*3: When high-speed counters (C235~C254) executes first and then CJ instruction is executed, he counter will keep counting and its associated output status remains.

Y1 is a dual output. When M0 = OFF, Y1 is controlled by M1. M0 = ON, Y1 is controlled by M12.



API	Mnemonic			Operands	Function	Controllers			
						ES2/EX2	SS2	SA2	SX2
01		CALL	P	S	Call Subroutine				

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: The destination pointer P of the call subroutine.

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. Subroutines must be placed after FEND instruction.
2. Subroutines must end with SRET instruction.
3. CALL pointers and CJ instruction pointers are not allowed to coincide.
4. CALL instructions can call the same 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
02	SRET	Subroutine Return	ES2/EX2 SS2 SA2 SX2

OP	Descriptions	Program Steps
N/A	No contact to drive the instruction is required Automatically returns program execution to the address after CALL instruction in O100.	SRET: 1 step

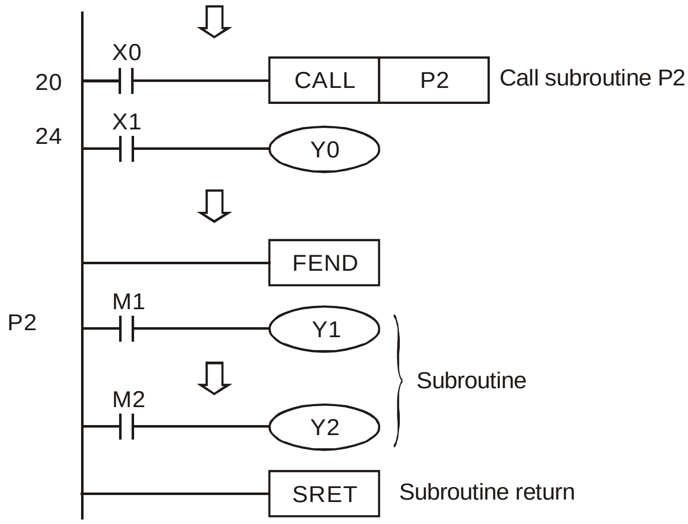
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Explanations:

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

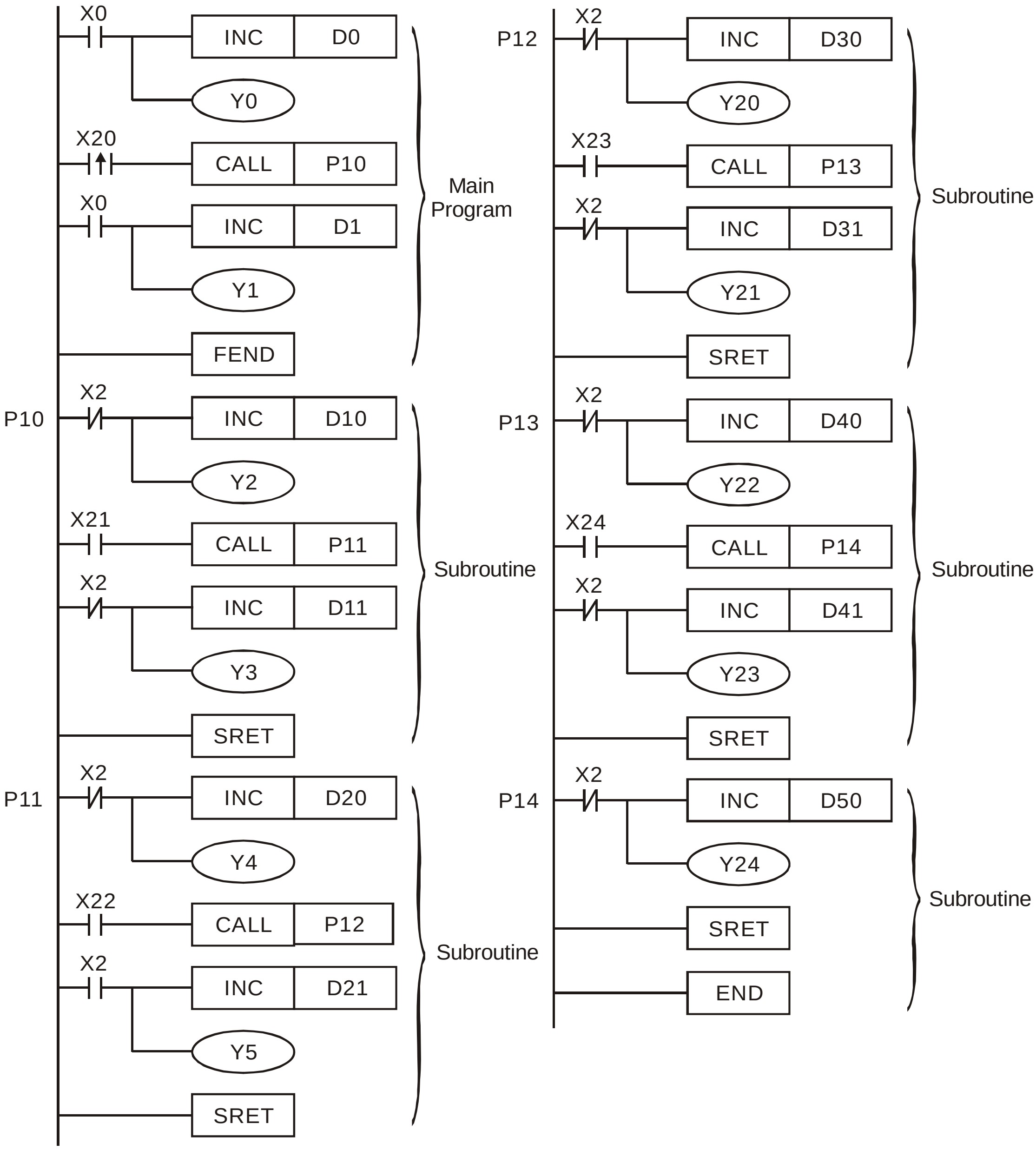
Program example 1:

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 address 24 and continue the 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 to finish the remaining instructions.
6. The execution of subroutines will go backwards to the subroutine of upper level until SRET instruction in P10 subroutine is executed. After this program execution will return to the main program.



API	Mnemonic	Function	Controllers
03	IRET	Interrupt Return	ES2/EX2 SS2 SA2 SX2

OP	Descriptions	Program Steps
N/A	No contact to drive the instruction is required. IRET ends the processing of an interrupt subroutine and returns execution back to the main program	IRET: 1 step

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

API	Mnemonic	Function	Controllers
04	EI	Enable Interrupt	ES2/EX2 SS2 SA2 SX2

OP	Descriptions	Program Steps
N/A	No contact to drive the instruction is required. 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	EI: 1 step

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

API	Mnemonic	Function	Controllers
05	DI	Disable Interrupt	ES2/EX2 SS2 SA2 SX2

OP	Descriptions	Program Steps
N/A	No contact to drive the instruction is required. DI instruction disables PLC to accept interrupts. When the special auxiliary relay M1050 ~ M1059 for disabling interruption is driven, the corresponding interruption request will not be executed even in the range allowed for interruptions.	DI: 1 step

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Explanations:

1. EI instruction allows interrupting subroutine in the program, e.g. external interruption, timer interruption, and high-speed counter interruption.
2. In the program, interruption subroutines are enabled between EI and DI instructions. If there is no section requires to be interrupt-disabled, DI instruction can be omitted.
3. Interrupt subroutines must be placed after the FEND instruction.

4. Other interrupts are not allowed during execution of a current interrupt routine.
5. When many interruptions occur, the priority is given to the firstly executed interruption. If several interruptions occur at the same time, the priority is given to the interruption with the smaller pointer No.
6. 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 occurs.
7. When using the interruption pointer, DO NOT repeatedly use the high-speed counter driven by the same X input contact.
8. When immediate I/O is required during the interruption, write REF instruction in the program to update the status of I/O

Points to note:

Interrupt pointers (I):

a) External interrupts: 8 points including (I000/I001, X0), (I100/I101, X1), (I200/I201, X2), (I300/I301, X3), (I400/I401, X4), (I500/I501, X5), (I600/I601, X6) and (I700/I701, X7) (00 designates interruption in falling-edge, 01 designates interruption in rising-edge)

b) Timer interrupts: 2 points including I605~I699 and I705~I799 (Timer resolution = 1ms)

c) High-speed counter interrupts: 8 points including I010, I020, I030, I040, I050, I060, I070, and I080. (used with API 53 DHSCS instruction to generate interrupt signals)

d) Communication interrupts: 3 points including I140, I150 and I160

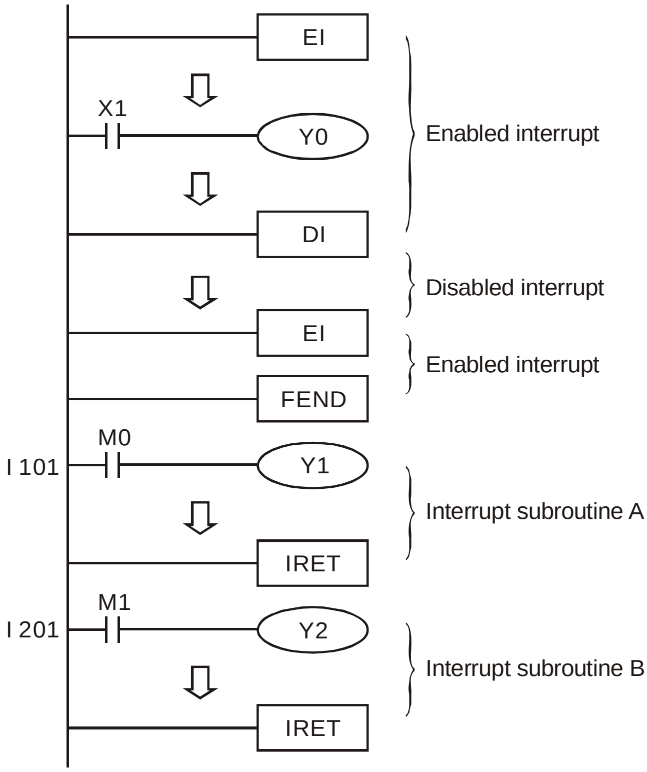
e) Associated flags:

Flag	Function
M1050	Disable external interruption I000 / I001
M1051	Disable external interruption I100 / I101
M1052	Disable external interruption I200 / I201
M1053	Disable external interruption I300 / I301
M1054	Disable external interruption I400 / I401
M1055	Disable external interruption I500 / I501, I600 / I601, I700 / I701
M1056	Disable timer interrupts I605~I699
M1057	Disable timer interrupts I705~I799
M1059	Disable high-speed counter interruptions I010~I080
M1280	I000/I001 Reverse interrupt trigger pulse direction (Rising/Falling)
M1284	I400/I401 Reverse interrupt trigger pulse direction (Rising/Falling)
M1286	I600/I601 Reverse interrupt trigger pulse direction (Rising/Falling)

Note: Default setting of I000(X0) is falling-edge triggered. When M1280=ON and EI is enabled, PLC will reverse X0 as rising-edge triggered. To reset X0 as falling-edge, reset M1280 first and execute DI instruction. After this, X0 will be reset as falling-edge when EI is executed again.

Program example:

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



API	Mnemonic	Function	Controllers			
06	FEND	The End of The Main Program (First End)	ES2/EX2	SS2	SA2	SX2

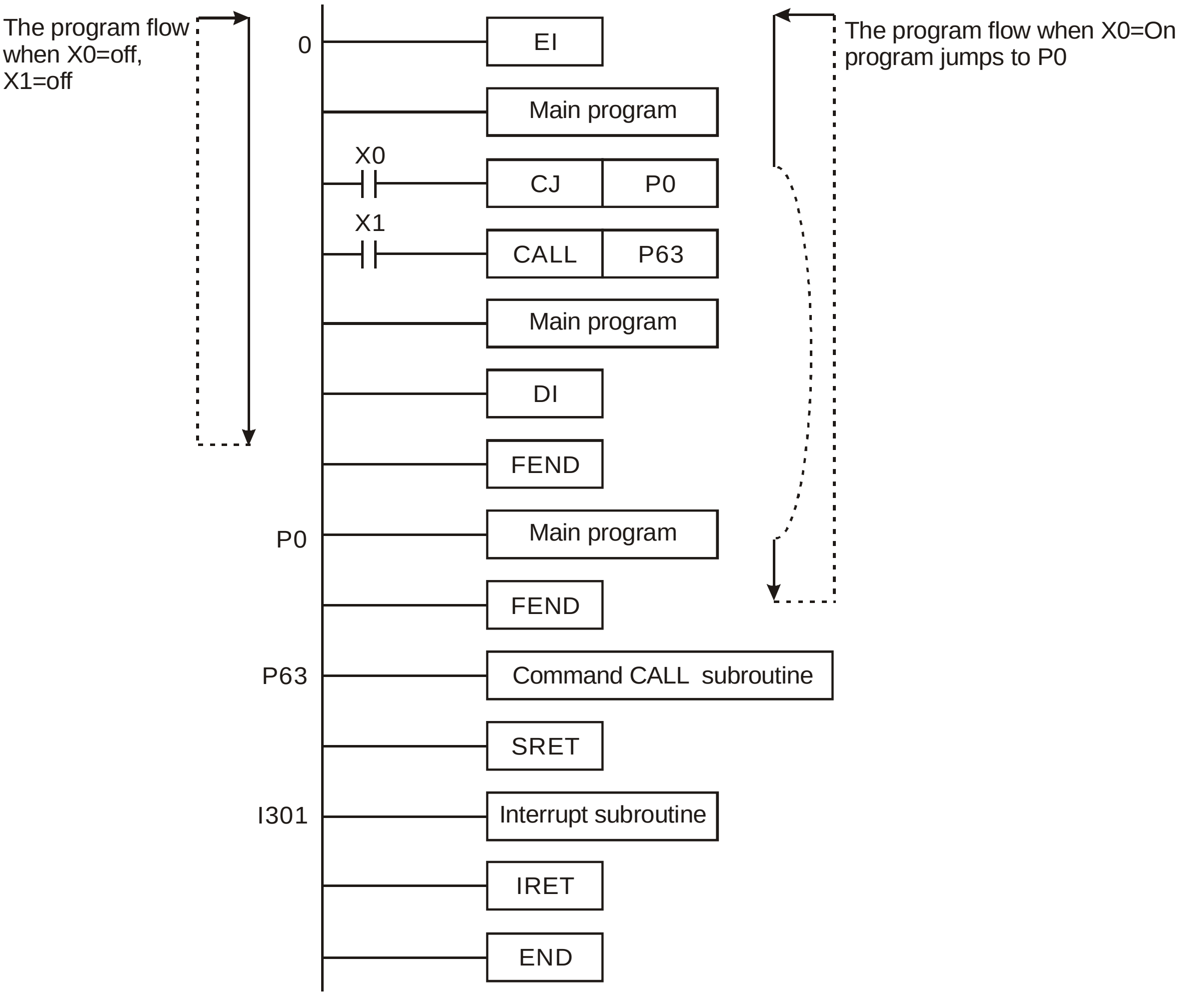
OP	Descriptions	Program Steps
N/A	No contact to drive the instruction is required.	FEND: 1 step

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

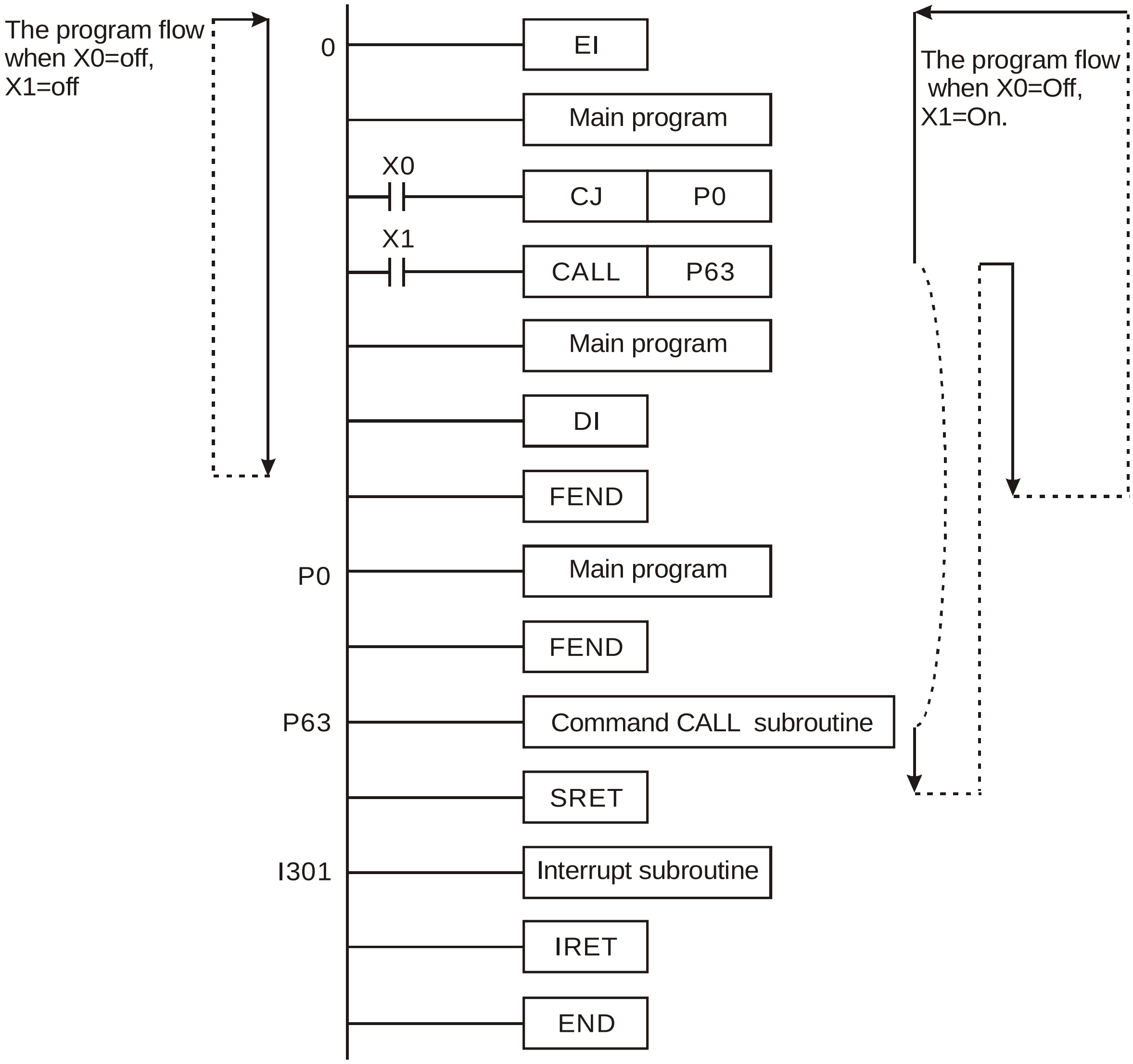
Explanations:

1. Use FEND instruction when the program uses either CALL instructions or interrupts. If no CALL instruction or interrupts are used, use END instruction to end the main program.
2. The instruction functions same as END instruction in PLC operation process.
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 several FEND instructions are in use, place the subroutine and interruption service programs between the final FEND and END instruction.
7. When CALL instruction is executed, executing FEND before SRET will result in errors.
8. When FOR instruction is executed, executing FEND before NEXT will result in errors.

CJ Instruction Program Flow



CALL Instruction Program Flow



API	Mnemonic			Function	Controllers
07		WDT	P	Watchdog Timer Refresh	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

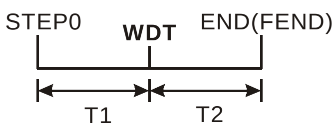
Explanations:

1. WDT instruction can be used to reset the Watch Dog Timer. If the PLC scan time (from address 0 to END or FEND instruction) is more than 200ms, the ERROR LED will flash. In this case, users have to turn the power OFF and then ON to clear the fault. PLC will determine the status of RUN/STOP according to RUN/STOP switch. If there is no RUN/STOP switch, PLC will return to STOP status automatically.
2. Time to use WDT:

a) When error occur in PLC system.

b) When the scan time of the program exceeds the WDT value in D1000. It can be modified by using the following two methods.

i. Use WDT instruction



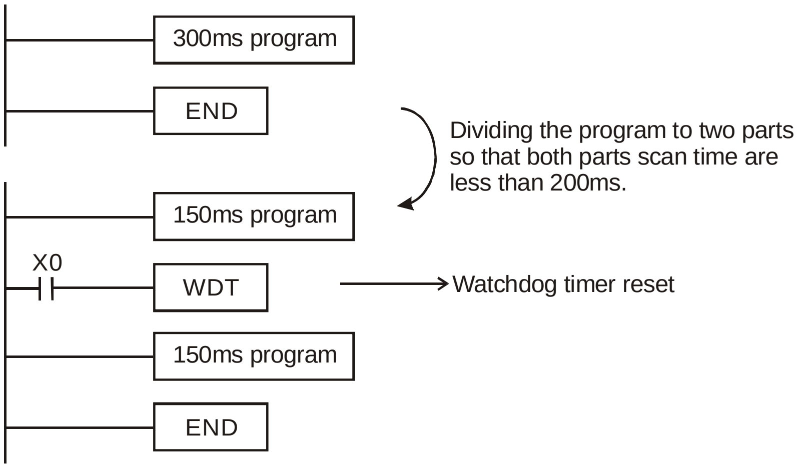
ii. Use the set value in D1000 (Default: 200ms) to change the time for watchdog.

Points to note:

1. When the WDT instruction is used it will operate on every program scan as long as its input condition has been made. To force the WDT instruction to operate for only ONE scan, users have to use the pulse (P) format of the WDT instruction, i.e. WDTP.
2. The watchdog timer has a default setting of 200ms. This time limit can be customized to users requirement by editing the content in D1000, the wathdog 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, making scan time of the first half and second half of the program being less than 200ms.



API	Mnemonic	Operands	Function	Controllers
08	FOR	S	Start of a FOR-NEXT Loop	ES2/EX2 SS2 SA2 SX2

Type / OP	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					*	*	*	*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: The number of times for the loop to be repeated.

API	Mnemonic	Function	Controllers
09	NEXT	End of a FOR-NEXT Loop	ES2/EX2 SS2 SA2 SX2

OP	Descriptions	Program Steps
N/A	No contact to drive the instruction is required.	NEXT: 1 step

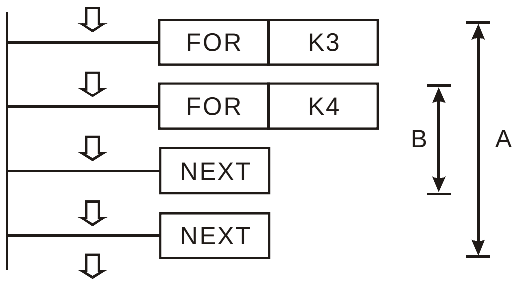
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Explanations:

1. FOR and NEXT instructions are used when loops are needed. No contact to drive the instruction is required.
2. "N" (number of times loop is repeated) may be within the range of K1 to K32767. If the range N≦K1, N is regarded as K1.
3. An error will occur in the following conditions:
   - NEXT instruction is before FOR instruction.
   - FOR instruction exists but NEXT instruction does not exist..
   - There is a NEXT instruction after the FEND or END instruction.
   - Number of FOR instructions differs from that of NEXT instructinos.
4. FOR~NEXT loops can be nested for maximum five levels. Be careful that if there are too many loops, the increased PLC scan time may cause timeout of watchdog timer and error. Users can use WDT instruction to modify this problem.

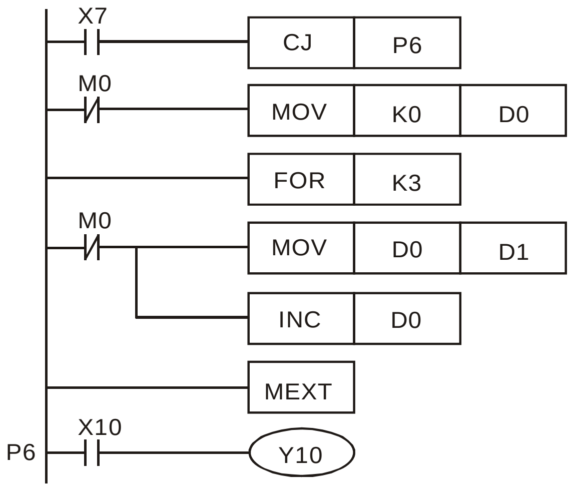
Program example 1:

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



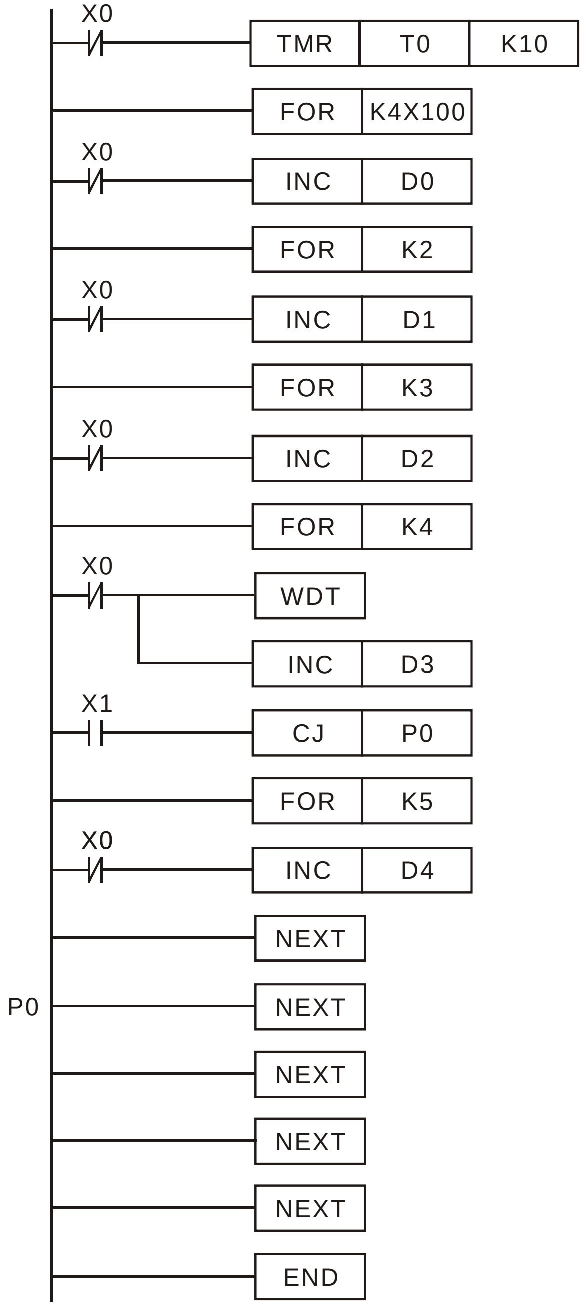
Program example 2:

When X7 = OFF, PLC will execute the program between FOR ~ NEXT. When X7 = ON, CJ instruction jumps to P6 and avoids executing the instructions between FOR ~ NEXT.



Program example 3:

Users can adopt CJ instruction to skip a specified FOR ~ NEXT loop. When X1 = ON, CJ instruction executes to skip the most inner FOR ~ NEXT loop.



API	Mnemonic			Operands	Function	Controllers			
10	D	CMP	P	S1 S2 D	Compare	ES2/EX2	SS2	SA2	SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	CMP, CMPP: 7 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DCMP, DCMPP: 13 steps
D		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

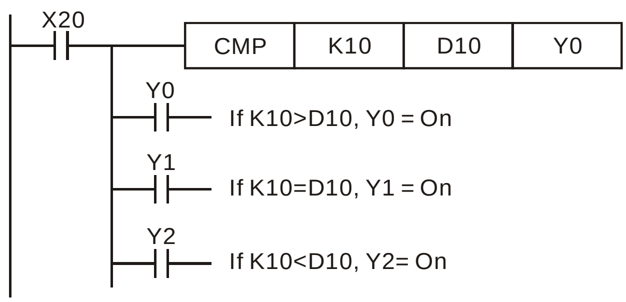
S_1: Comparison Value 1 **S_2**: Comparison Value 2 **D**: Comparison result

Explanations:

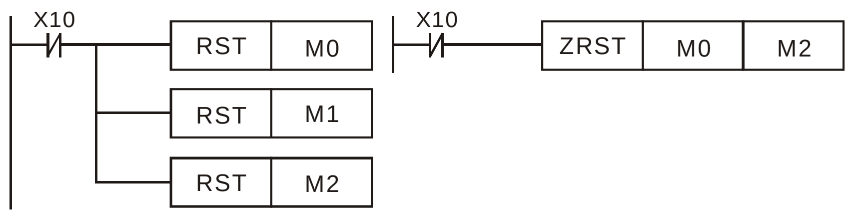
1. The contents of **S_1** and **S_2** are compared and **D** stores the comparison result.
2. The comparison values are signed binary values. If b15=1 in 16-bit instruction or b31=1 in 32-bit instruction, the comparison will regard the value as a negative binary value.
3. Operand **D** occupies 3 continuous devices. **D**, **D** +1, **D** +2 hold the comparison results, **D** = ON if **S_1** > **S_2**, **D** +1 = ON if **S_1** = **S_2**, **D** +2 = ON if **S_1** < **S_2**
4. If operand **S_1**, **S_2** use index register F, only 16-bit instruction is available.

Program example:

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



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



API	Mnemonic			Operands	Function	Controllers
11	D	ZCP	P	S1 S2 S D	Zone Compare	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	ZCP, ZCPP: 9 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DZCP, DZCPP: 17 steps
S					*	*	*	*	*	*	*	*	*	*	*	
D		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

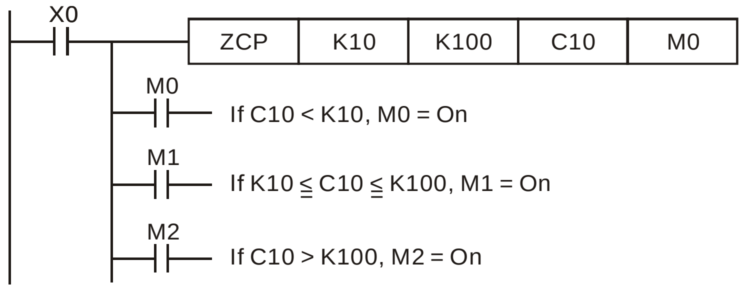
$\mathbf{S_1}$: Lower bound of zone comparison $\mathbf{S_2}$: Upper bound of zone comparison **S**: Comparison value **D**: Comparison result

Explanations:

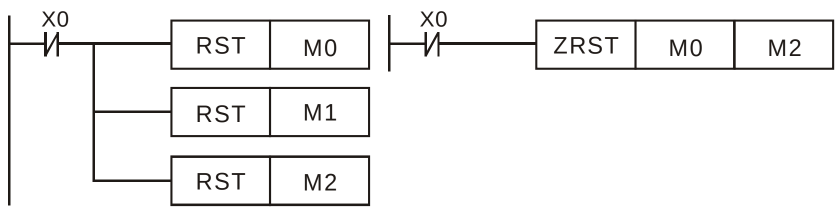
1. **S** is compared with its lower bound $\mathbf{S_1}$ and upper bound $\mathbf{S_2}$. **D** stores the comparison results.
2. The comparison values are signed binary values. If b15=1 in 16-bit instruction or b31=1 in 32-bit instruction, the comparison will regard the value as a negative binary value.
3. Operand $\mathbf{S_1}$ should be smaller than operand $\mathbf{S_2}$. When $\mathbf{S_1} > \mathbf{S_2}$, the instruction takes $\mathbf{S_1}$ as the 1st comparison value and performs normal comparison similar to CMP instruction.
4. If operand $\mathbf{S_1}$, $\mathbf{S_2}$, and **S** use index register F, only 16-bit instruction is available.
5. Operand **D** occupies 3 continuous devices. **D**, **D** +1, **D** +2 hold the comparison results, **D** = ON if $\mathbf{S_1} > \mathbf{S}$, **D** +1 = ON if $\mathbf{S_1} \leqq \mathbf{S} \leqq \mathbf{S_2}$, **D** +2 = ON if $\mathbf{S_2} < \mathbf{S}$

Program example:

1. If **D** is set as M0, then M0, M1, M2 will work as the program example below.
2. 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.



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



API	Mnemonic			Operands	Function	Controllers
						ES2/EX2 SS2 SA2 SX2
12	D	MOV	P	S D	Move	

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

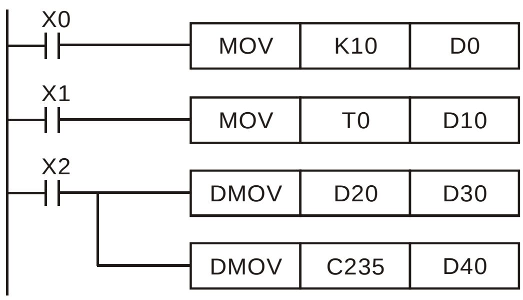
S: Source of data **D**: Destination of data

Explanations:

1. When this instruction is executed, the content of **S** will be moved directly to **D**. When this instruction is not executed, the content of **D** remains unchanged
2. If operand **S** and **D** use index register F, only 16-bit instruction is applicable

Program example:

1. MOV will move a 16-bit value from the source location to the destination.
   a) When X0 = OFF, the content of D0 remains unchanged. If X0 = ON, the data in K10 is moved to D0.
   b) When X1 = OFF, the content of D10 remains unchanged. If X1 = ON, the data of T0 is moved to D10 data register.
2. DMOV will move a 32-bit value from the source location to the destination.
   a) When X2 = OFF, the content of (D31, D30) and (D41, D40) remain unchanged.
   b) When 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	S m1 m2 D n	Shift Move	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
S							*	*	*	*	*	*	*	*	*	SMOV, SMOVP: 11 step
m_1					*	*										
m_2					*	*										
D								*	*	*	*	*	*	*	*	
n					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device **m_1**: Start digit to be moved from source device **m_2**: Number of digits to be moved **D**: Destination device **n**: Start digit of the destination device for the moved digits

Explanation:

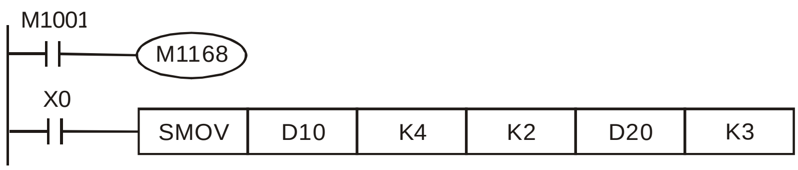
1. This instruction is able to re-allocate or combine data. When the instruction is executed, **m_2** digits of contents starting from digit **m_1** (from high digit to low digit) of **S** will be sent to **m_2** digits starting from digit **n** (from high digit to low digit) of **D**.
2. M1168 is used for designating SMOV working mode. When M1168 = ON, the instruction is in BIN mode. When M1168 = OFF, the instruction is in BCD mode.

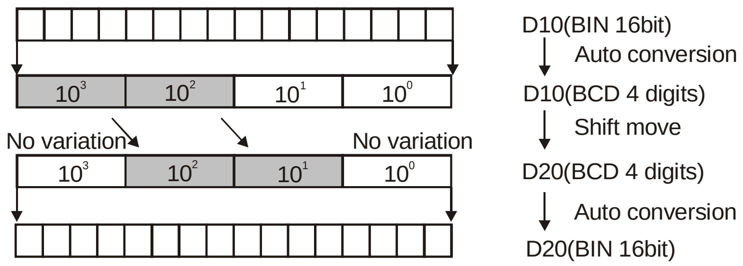
Points to note:

1. The range of **m_1**: 1 – 4
2. The range of **m_2**: 1 – **m_1**
3. The range of **n**: **m_2** – 4

Program example 1:

1. When M1168 = OFF (in BCD mode) and X0 = ON, the 4th (thousand) and 3rd (hundred) digit of the decimal value in D10 start to move to the 3rd (hundred) and 2nd (ten) digit of the decimal value in D20. 10^3 and 10^0 of D20 remain unchanged after this instruction is executed.
2. When the BCD value exceeds the range of 0 ~ 9,999, PLC detects an operation error and will not execute the instruction. M1067, M1068 = ON and D1067 stores the error code OE18 (hex).

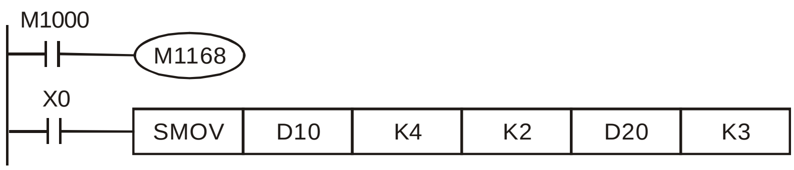


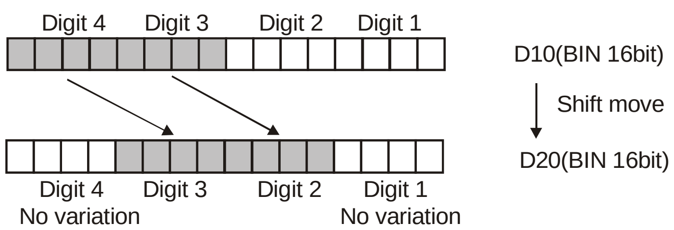


If D10 = K1234, D20 = K5678 before execution, D10 remains unchanged and D20 = K5128 after execution.

Program example 2:

When M1168 = ON (in BIN mode) and SMOV instruction is in use, D10 and D20 will not be converted in BCD format but be moved in BIN format (4 digits as a unit).



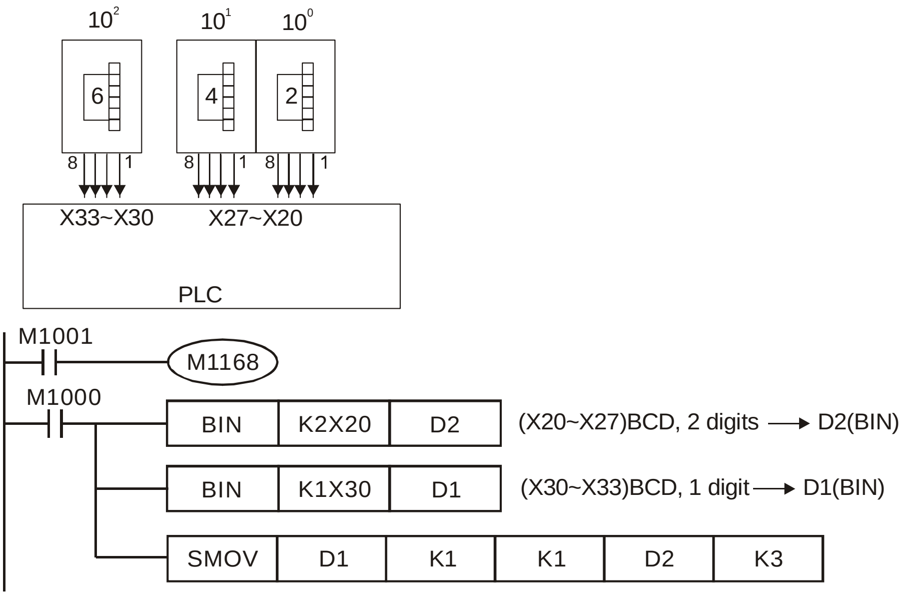


If D10 = H1234, D20 = H5678 before execution, D10 remains unchanged and D20 = H5128 after execution.

Program example 3:

1. This instruction can be used to combine the DIP switches connected to the input terminals without continuous numbers.
2. Move the 2 digits of the right DIP switch (X27~X20) to the 2 digits of D2, and the 1 digit of the DIP switch (X33~X30) to the 1st digit of D1.
3. Use SMOV instruction to move the 1st digit of D1 to the 3rd digit of D2 and combine the values from two DIP switches into one set of value.

.



API	Mnemonic			Operands	Function	Controllers
14	D	CML	P	S D	Compliment	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

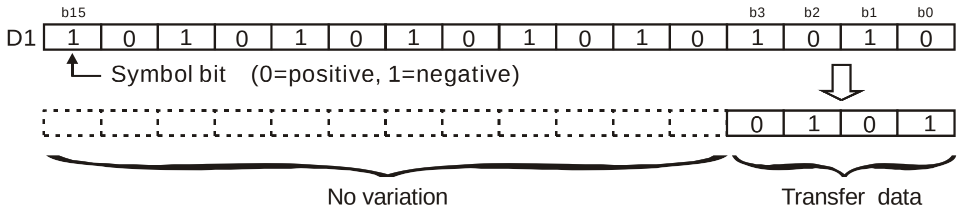
S: Source of data **D**: Destination device

Explanations:

1. The instruction reverses the bit pattern (0→1, 1→0) of all the contents in **S** and sends the contents to **D**.
2. If operand **S** and **D** use index register F, only 16-bit instruction is available

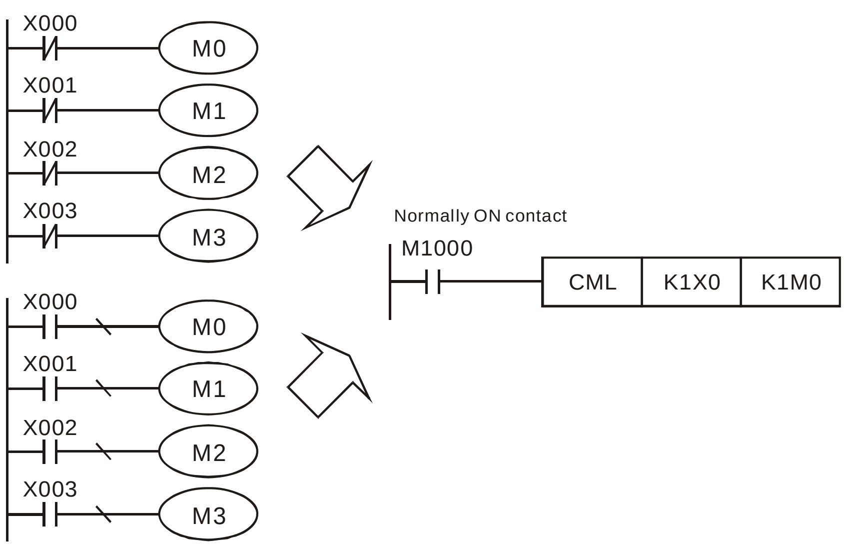
Program example 1:

When X10 = ON, b0 ~ b3 in D1 will be inverted and sent to Y0 ~ Y3



Program example 2:

The diagram below can be substituted by the instruction on the right.



API	Mnemonic		Operands	Function	Controllers
15	BMOV	P	S D n	Block Move	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S							*	*	*	*	*	*	*			BMOV, BMOVP: 7 steps
D								*	*	*	*	*	*			
n					*	*					*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

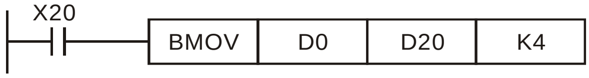
S: Start of source devices **D**: Start of destination devices **n**: Number of data to be moved

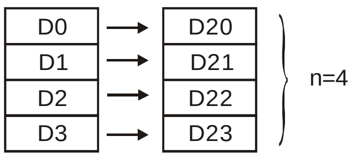
Explanations:

1. The program copies a specified block of devices to another destination. Contents in **n** registers starting from **S** will be moved to **n** registers starting from **D**. If **n** exceeds the actual number of available source devices, only the devices that fall within the valid range will be used
2. Range of **n**: 1 ~ 512.

Program example 1:

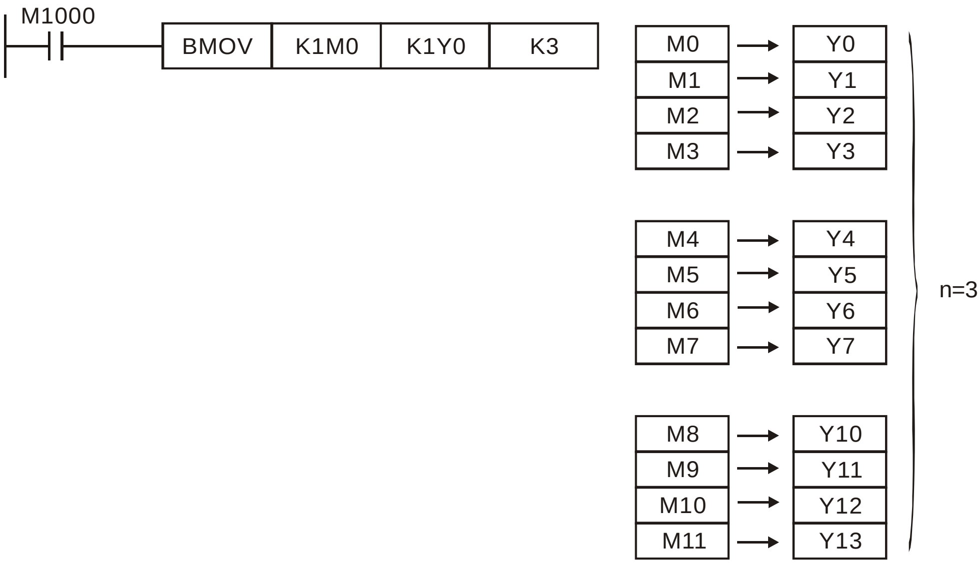
When X20 = ON, the contents in registers D0 ~ D3 will be moved to the 4 registers D20 ~ D23





Program example 2:

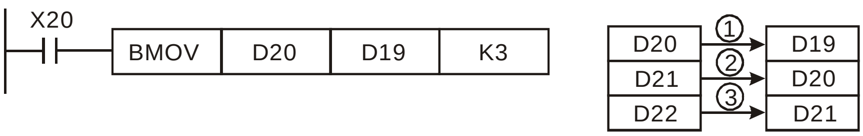
Assume the bit devices KnX, KnY, KnM and KnS are designated for moving, the number of digits of **S** and **D** has to be the same, i.e. their **n** has to be the same.



Program example 3:

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 Move	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*	*	*	*	*	*	*	*	*	*	FMOV, FMOVP: 7 steps
D								*	*	*	*	*	*			DFMOV, DFMOVP: 13
n					*	*										steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

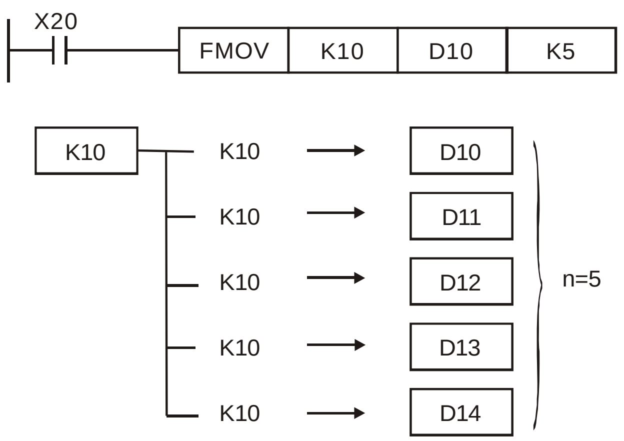
S: Source of data **D**: Destination of data **n**: Number of data to be moved

Explanations:

1. The contents in n registers starting from the device designated by **S** will be moved to n registers starting from the device designated by **D**. If n exceeds the actual number of available source devices, only the devices that fall within the valid range will be used
2. If operand **S** use index register F, only 16-bit instruction is available
3. The range of n: 1~ 512

Program example:

When X20 = ON, K10 will be moved to the 5 consecutive registers starting from D10



API	Mnemonic			Operands	Function	Controllers
17	D	XCH	P	D1 D2	Exchange	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
D_1								*	*	*	*	*	*	*	*	XCH, XCHP: 5 steps
D_2								*	*	*	*	*	*	*	*	DXCH, DXCHP: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

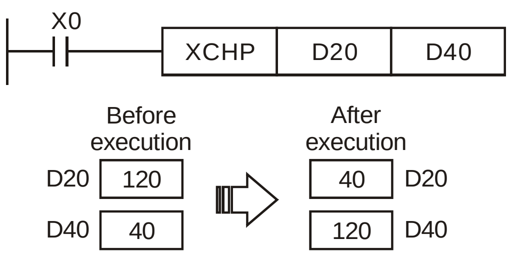
D_1: Device to be exchanged 1 **D_2**: Device to be exchanged 2

Explanations:

1. The contents in the devices designated by **D_1** and **D_2** will exchange
2. It is better to apply a pulse execution for this instruction (XCHP).
3. If operand **D1** and **D2** use index register F, only 16-bit instruction is available.

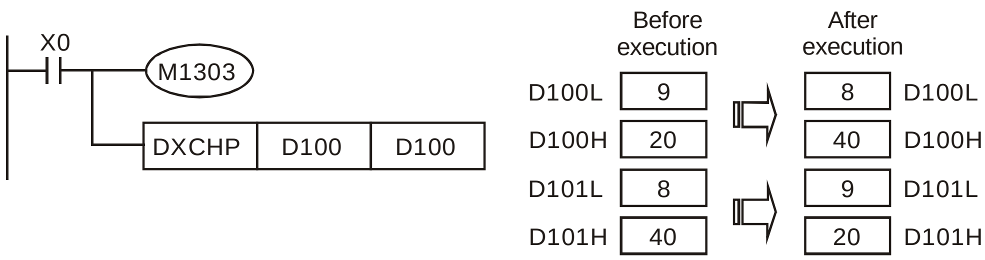
Program example:

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



Points to note:

1. As a 16-bit instruction, when the devices designated by **D_1** and **D_2** are the same and M1303 = ON, the upper and lower 8 bits of the designated devices exchange with each other.
2. As a 32-bit instruction, when the devices designated by **D_1** and **D_2** are the same and M1303 = ON, the upper and lower 16 bits in the designated device exchange with each other.
3. When X0 = ON and M1303 = ON, 16-bit contents in D100 and those in D101 will exchange with each other.



API	Mnemonic			Operands	Function	Controllers			
						ES2/EX2	SS2	SA2	SX2
18	D	BCD	P	S D	Convert BIN to BCD				

Type / OP	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							*	*	*	*	*	*	*	*	*	DBCD, DBCDP: 9 steps
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source of data **D**: Conversion result

Explanations:

1. The content in **S** (BIN value) is converted into BCD value and stored in **D**
2. As a 16-bit (32-bit) instruction, when the conversion result exceeds the range of 0 ~ 9,999 (0 ~ 99,999,999), and M1067, M1068 = ON, D1067 will record the error code 0E18 (hex)
3. If operand **S** and **D** use index register F, only 16-bit instruction is available.
4. Flags: M1067 (Program execution error), M1068 (Execution error locked), D1067 (error code)

Program example:

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

2. If 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	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

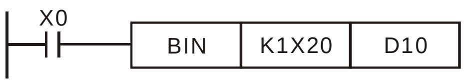
S: Source of data **D**: Conversion result

Explanations:

1. The content in **S** (BCD value) is converted into BIN value and stored in **D**.
2. The valid range of source **S**: BCD (0 to 9,999), DBCD (0 to 99,999,999)
3. 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.
4. If operand S and D use index register F, only 16-bit instruction is available.
5. Flags: M1067 (Program execution error), M1068 (Execution error locked), D1067 (error code)

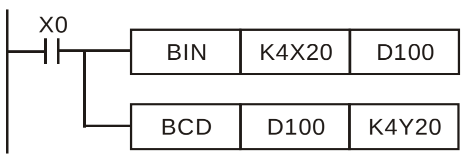
Program example:

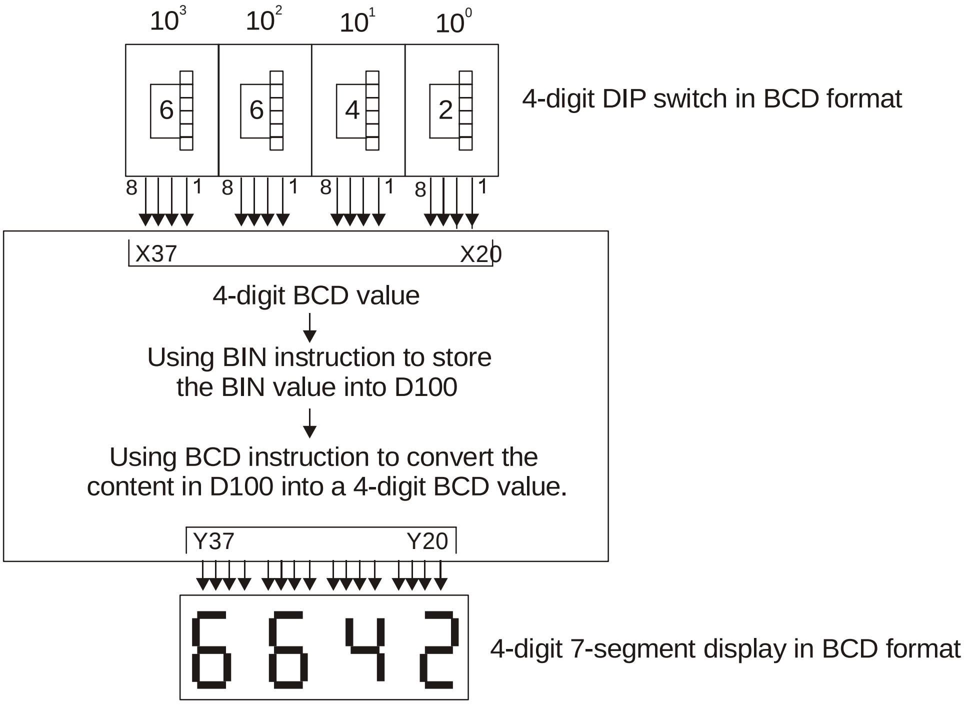
When X0 = ON, the BCD value of K1M0 will be converted to BIN value and stored in D10.



Points to note:

1. When PLC needs to read an external DIP switch in BCD format, BIN instruction has to be first adopted to convert the read data into BIN value and store the data in PLC.
2. On the contrary when PLC needs to display a value on a BCD format 7-segment displayer, BCD instruction is required to convert the internal data into BCD value then sent the value to the displayer.
3. When X0 = ON, the BCD value of K4X20 is converted into BIN value and sent to D100. The BIN value of D100 will then be converted into BCD value and sent to K4Y20.





10^3 10^2 10^1 10^0
6 6 4 2
4-digit DIP switch in BCD format
8 1 8 1 8 1 8 1
X37
X20
4-digit BCD value
Using BIN instruction to store the BIN value into D100
Using BCD instruction to convert the content in D100 into a 4-digit BCD value.
Y37
Y20
6642
4-digit 7-segment display in BCD format

API	Mnemonic			Operands	Function	Controllers
20	D	ADD	P	S1 S2 D	Addition	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	ADD, ADDP: 7 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DADD, DADDP: 13 steps
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Summand **S_2**: Addend **D**: Sum

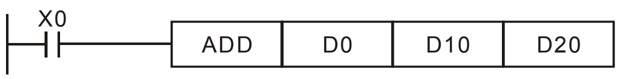
Explanations:

1. This instruction adds **S_1** and **S_2** in BIN format and store the result in **D**.
2. The most significant bit (MSB) is the sign bit of the data. 0 indicates positive and 1 indicates negative. All calculations is algebraically processed, e.g. 3 + (-9) = -6.
3. If **S_1**, **S_2** and **D** use device F, only 16-bit instruction is applicable.
4. Flags: M1020 (Zero flag), M1021 (Borrow flag), M1022 (Carry flag)

Program Example 1:

In 16-bit BIN addition:

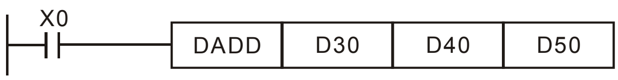
When X0 = ON, the content in D0 will plus the content in D10 and the sum will be stored in D20.



Program Example 2:

In 32-bit BIN addition:

When X0 = ON, the content in (D31, D30) will plus the content in (D41, D40) and the sum will be stored in (D51, D50). D30, D40 and D50 are low word; D31, D41 and D51 are high word



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

Operation of flags:

16-bit instruction:

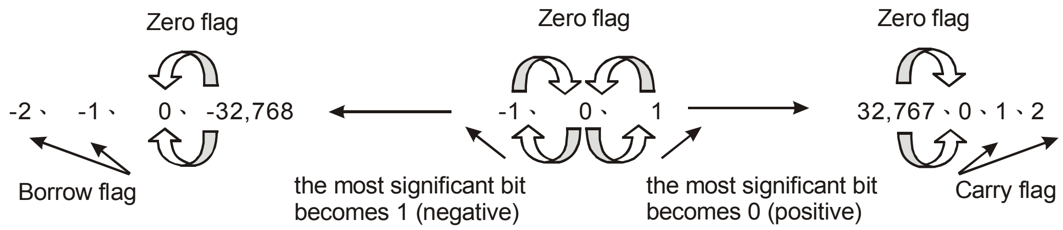
1. If the operation result is “0”, the zero flag M1020 will be ON.
2. If the operation result exceeds -32,768, the borrow flag M1021 will be ON.
3. If the operation result exceeds 32,767, the carry flag M1022 will be ON.

32-bit instruction:

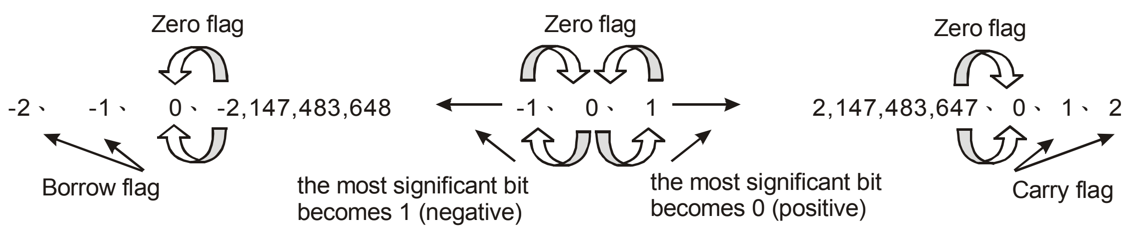
1. If the operation result is “0”, the zero flag, M1020 will be ON.

2. If the operation result exceeds -2,147,483,648, the borrow flag M1021 will be ON.
3. If the operation result exceeds 2,147,483,647, the carry flag M1022 will be ON

16-bit instruction:



32-bit instruction:



API	Mnemonic			Operands	Function	Controllers
21	D	SUB	P	(S1) (S2) (D)	Subtraction	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	SUB, SUBP: 7 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DSUB, DSUBP: 13 steps
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Minuend **S_2**: Subtrahend **D**: Remainder

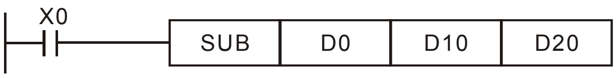
Explanations:

1. This instruction subtracts **S_1** and **S_2** in BIN format and stores the result in **D**
2. The MSB is the sign bit. 0 indicates positive and 1 indicates negative. All calculation is algebraically processed.
3. If **S_1**, **S_2** and **D** use device F, only 16-bit instruction is applicable.
4. Flags: M1020 (Zero flag), M1021 (Borrow flag), M1022 (Carry flag). The flag operations of ADD instruction can also be applied to the subtract instruction.

Program Example 1:

In 16-bit BIN subtraction:

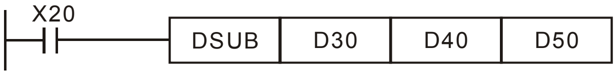
When X0 = ON, the content in D0 will minus the content in D10 and the results will be stored in D20



Program Example 2:

In 32-bit BIN subtraction:

When X10 = ON, the content in (D31, D30) will minus the content in (D41, D40) and the results will be stored in (D51, D50). D30, D40 and D50 are low word; D31, D41 and D51 are high word



(D31, D30) – (D41, D40) = (D51, D50)

API	Mnemonic			Operands	Function	Controllers
22	D	MUL	P	S1 S2 D	Multiplication	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*		MUL, DMULP: 7 steps
S_2					*	*	*	*	*	*	*	*	*	*		DMUL, DMULP: 13 steps
D								*	*	*	*	*	*	*		

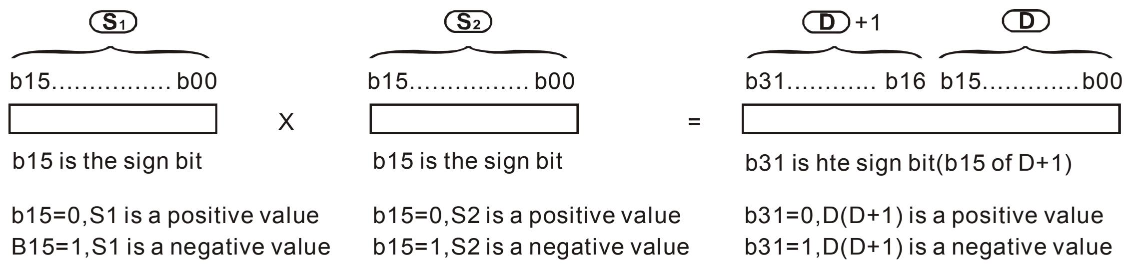
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Multiplicand **S_2**: Multiplicator **D**: Product

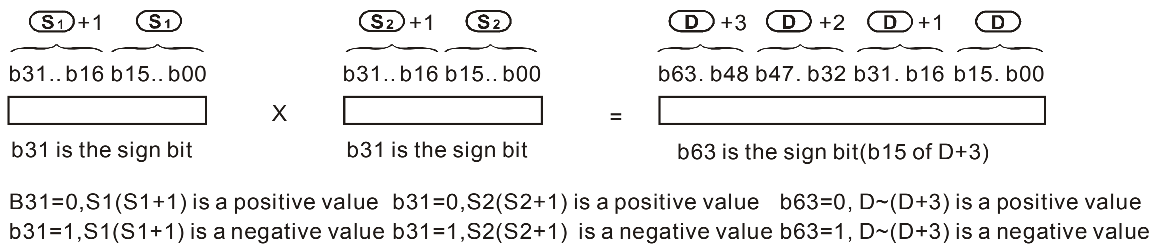
Explanations:

1. This instruction multiplies **S_1** by **S_2** in BIN format and stores the result in **D**. Care should be taken on positive/negative signs of **S_1**, **S_2** and **D** when doing 16-bit and 32-bit operations.
2. MSB = 0, positive; MSB = 1, negative.
3. If operands **S_1**, **S_2** use index F, then only 16-bit instruction is available.
4. If operand **D** use index E, then only 16-bit instruction is available.
5. 16-bit BIN multiplication



If **D** is specified with a bit device, it can designate K1 ~ K4 to store a 16-bit result. Users can use consecutive 2 16-bit registers to store 32-bit data.

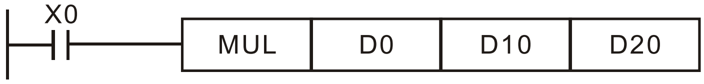
6. 32-bit BIN multiplication



If **D** is specified with a word device, it can specify K1~K8 to store a 32-bit result. Users can use 2 consecutive 32-bit registers to store 64-bit data.

Program Example:

The 16-bit D0 is multiplied by the 16-bit D10 and brings forth a 32-bit product. The higher 16 bits are stored in D21 and the lower 16-bit are stored in D20. ON/OFF of MSB indicates the positive/negative status of the operation result.



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

16-bit $\times$ 16-bit $=$ 32-bit

API	Mnemonic			Operands	Function	Controllers
23	D	DIV	P	S1 S2 D	Division	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*		DIV, DIVP: 7 steps
S_2					*	*	*	*	*	*	*	*	*	*		DDIV, DDIVP: 13 steps
D								*	*	*	*	*	*	*		

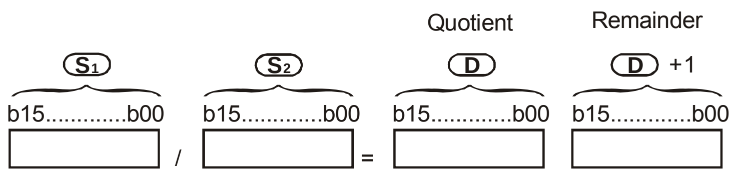
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Dividend **S_2**: Divisor **D**: Quotient and remainder

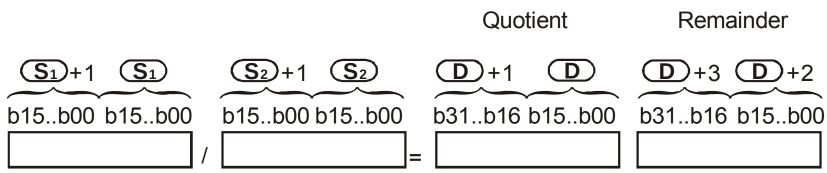
Explanation:

1. This instruction divides **S_1** and **S_2** in BIN format and stores the result in **D**. Care should be taken on positive/negative signs of **S_1**, **S_2** and **D** when doing 16-bit and 32-bit operations.
2. This instruction will not be executed when the divisor is 0. M1067 and M1068 will be ON and D1067 records the error code 0E19 (hex).
3. If operands **S_1**, **S_2** use index F, then only 16-bit instruction is available.
4. If operand **D** use index E, then only 16-bit instruction is available.
5. 16-bit BIN division:



If D is specified with a bit device, it can designate K1 ~ K4 to store a 16-bit result. Users can use consecutive 2 16-bit registers to store 32-bit data of the quotient and remainder.

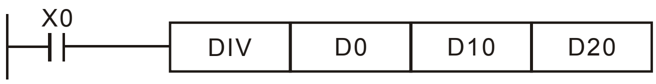
6. 32-bit BIN division:



If D is specified with a bit device, it can designate K1 ~ K8 to store a 32-bit result. Users can use consecutive 2 32-bit registers to store the quotient and remainder.

Program Example:

When X0 = ON, D0 will be divided by D10 and the quotient will be stored in D20 and remainder in D21. ON/OFF of the MSB indicates the positive/negative status of the result value..



API	Mnemonic			Operands	Function	Controllers
24	D	INC	P	D	Increment	ES2/EX2 SS2 SA2 SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

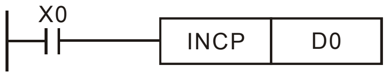
D: Destination device

Explanations:

1. If the instruction is not used in pulse execution mode, the content in the designated device **D** will plus "1" in every scan period
2. When INC is executed, the content in **D** will be incremented. However, in 16-bit instruction, if +32,767 is reached and "1" is added, it will write a value of –32,768 to the destination. In 32-bit instruction, if +2,147,483,647 is reached and "1" is added, it will write a value of -2,147,483,648 to the destination.
3. This instruction is generally used in pulse execution mode (INCP, DINCP).
4. If operand **D** uses index F, only a 16-bit instruction is applicable..
5. The operation results will not affect M1020 ~ M1022.

Program Example:

When X0 is triggered, the content of D0 will be incremented by 1.



API	Mnemonic			Operands	Function	Controllers
25	D	DEC	P	D	Decrement	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

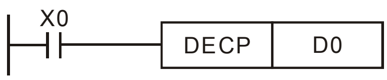
D: Destination device

Explanation:

1. If the instruction is not used in pulse execution mode, the content in the designated device D will minus "1" in every scan whenever the instruction is executed.
2. This instruction is generally used in pulse execution mode (DECP, DDECP).
3. In 16-bit instruction, if –32,768 is reached and "1" is minused, it will write a value of +32,767 to the destination. In 32-bit instruction, if -2,147,483,648 is reached and "1" is minused, it will write a value of +2,147,483,647 to the destination.
4. If operand **D** uses index F, only a 16-bit instruction is applicable.
5. The operation results will not affect M1020 ~ M1022

Program Example:

When X0 is triggered, the value in D0 will be decremented by 1.



API	Mnemonic		Operands	Function	Controllers
26	WAND	P	(S1) (S2) (D)	Logical Word AND	ES2/EX2 SS2 SA2 SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

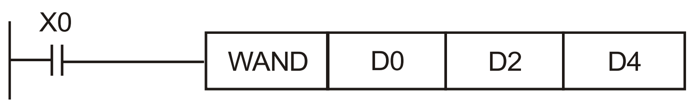
S_1: Source data device 1 **S_2**: Source data device 2 **D**: Operation result

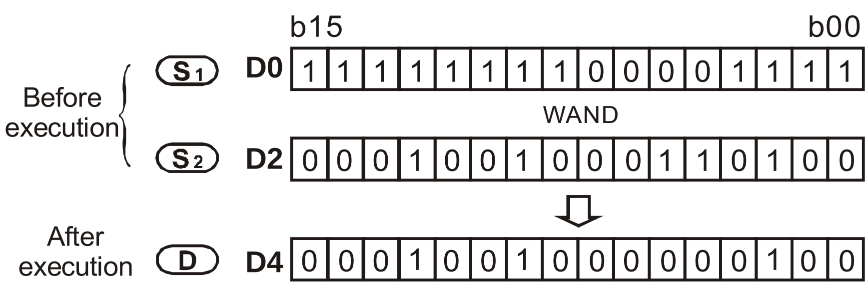
Explanations:

1. This instruction conducts logical AND operation of **S_1** and **S_2** in 16-bit mode and stores the result in **D**
2. For 32-bit operation please refer to DAND instruction..

Program Example:

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





API	Mnemonic		Operands	Function	Controllers
26	DAND	P	(S1) (S2) (D)	Logical DWord AND	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*		DAND, DANDP: 13 steps
S_2					*	*	*	*	*	*	*	*	*	*		
D								*	*	*	*	*	*	*		

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

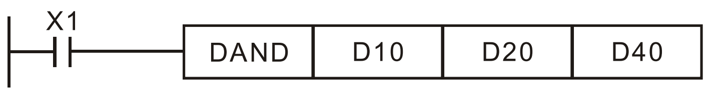
S_1: Source data device 1 **S_2**: Source data device 2 **D**: Operation result

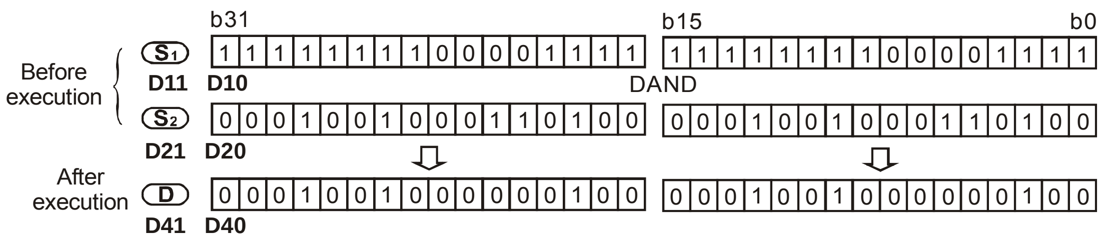
Explanations:

1. Logical double word (32-bit) AND operation.
2. This instruction conducts logical AND operation of **S_1** and **S_2** in 32-bit mode and stores the result in **D.**
3. If operands **S_1**, **S_2**, **D** use index F, only a 16-bit instruction is available.

Program Example:

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





API	Mnemonic			Operands	Function	Controllers
27		WOR	P	S1 S2 D	Logical Word OR	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	WOR, WORP: 7 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

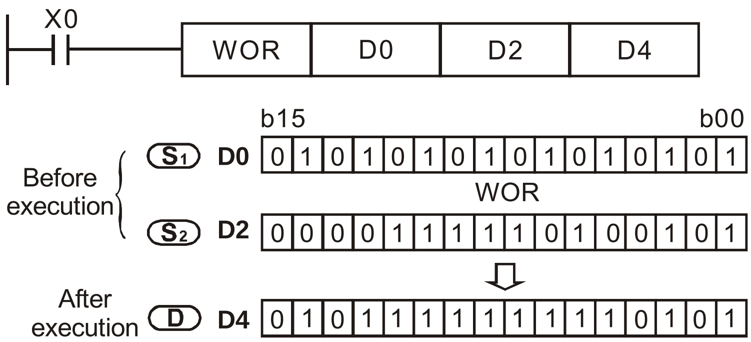
S_1: Source data device 1 **S_2**: Source data device 2 **D**: Operation result

Explanations:

1. This instruction conducts logical OR operation of **S_1** and **S_2** in 16-bit mode and stores the result in **D**.
2. For 32-bit operation please refer to DOR instruction.

Program Example:

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



API	Mnemonic			Operands	Function	Controllers
27	D	DOR	P	S1 S2 D	Logical DWord OR	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*		DOR, DORP: 13 steps
S_2					*	*	*	*	*	*	*	*	*	*		
D								*	*	*	*	*	*	*		

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

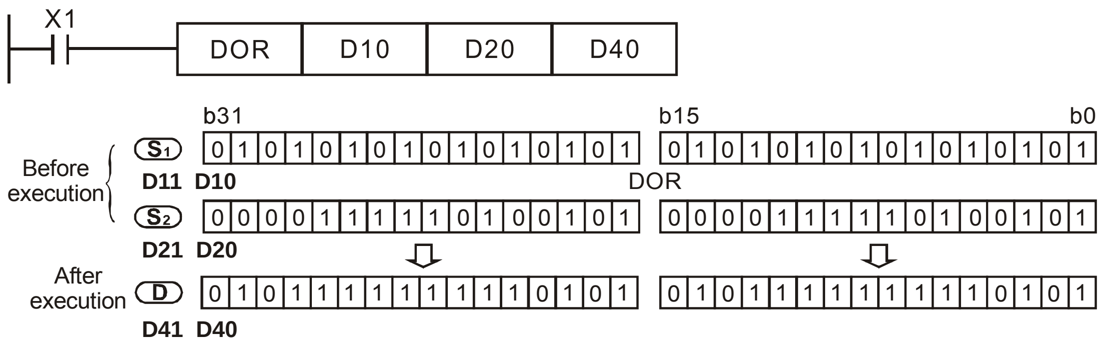
S_1: Source data device 1 **S_2**: Source data device 2 **D**: Operation result

Explanations:

1. Logical double word (32-bit) OR operation.
2. This instruction conducts logical OR operation of **S_1** and **S_2** in 32-bit mode and stores the result in **D**.
3. If operands **S_1**, **S_2**, **D** use index F, then only a 16-bit instruction is available.

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 OR is stored in (D41, D40).



API	Mnemonic			Operands	Function	Controllers
28		WXOR	P	S1 S2 D	Logical Word XOR	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices X	Y	M	S	Word devices K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	WXOR, WXORP: 7 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

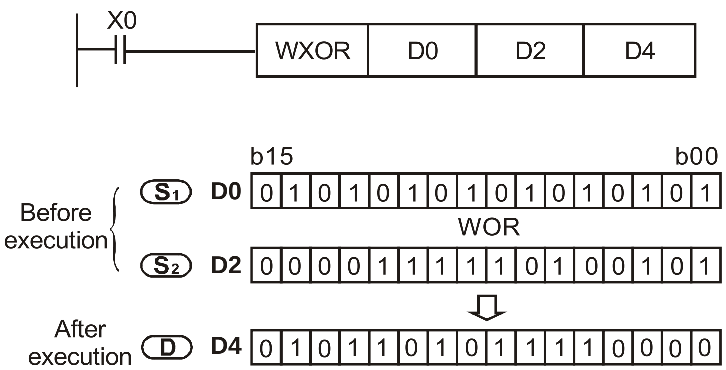
S_1: Source data device 1 **S_2**: Source data device 2 **D**: Operation result

Explanations:

1. This instruction conducts logical XOR operation of **S_1** and **S_2** in 16-bit mode and stores the result in **D**
2. For 32-bit operation please refer to DXOR instruction.

Program Example:

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



API	Mnemonic		Operands	Function	Controllers
28	DXOR	P	S1 S2 D	Logical DWord XOR	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											DXOR, DXORP: 13 steps
S_1					*	*	*	*	*	*	*	*	*	*		
S_2					*	*	*	*	*	*	*	*	*	*		
D								*	*	*	*	*	*	*		

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

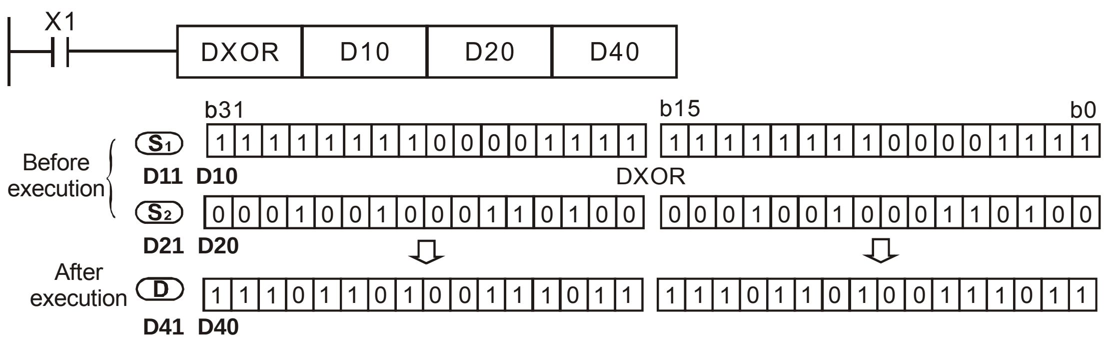
S_1: Source data device 1 **S_2**: Source data device 2 **D**: Operation result

Explanations:

1. Logical double word (32-bit) XOR operation.
2. This instruction conducts logical XOR operation of **S_1** and **S_2** in 32-bit mode and stores the result in **D**
3. If operands **S_1**, **S_2**, **D** use index F, only a 16-bit instruction is available.

Program Example:

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



API	Mnemonic			Operands	Function	Controllers
29	D	NEG	P	D	2's Complement (Negation)	ES2/EX2 SS2 SA2 SX2

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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

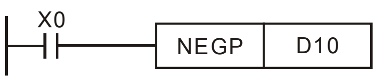
D: Device to store the operation result of 2's Compliment

Explanations:

1. This instruction conducts operation of 2's complement and can be used for converting a negative BIN value into an absolute value.
2. This instruction is generally used in pulse execution mode (NEGP, DNEGP).
3. If operand **D** uses index F, only a 16-bit instruction is available.

Program Example 1:

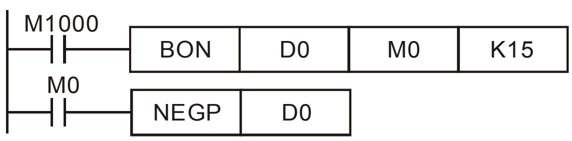
When X0 goes from OFF to ON, the phase of each bit in D10 will be reversed (0→1, 1→0) and then 1 will be added to the Least Significant Bit (LSB) of the register. Operation result will then be stored in D10.



Program Example 2:

To obtain the absolute value of a negative value:

1. When MSB (b15) of D0 is "1", M0 = ON. (D0 is a negative value).
2. When M0 = ON, the absolute value of D0 can be obtained by NEG instruction.



Program Example 3:

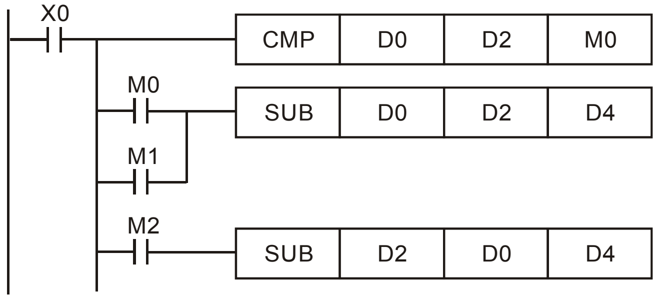
Obtain the absolute value of the remainder of the subtraction. When X0 = ON,

a) If D0 > D2, M0 = ON.

b) If D0 = D2, M1 = ON.

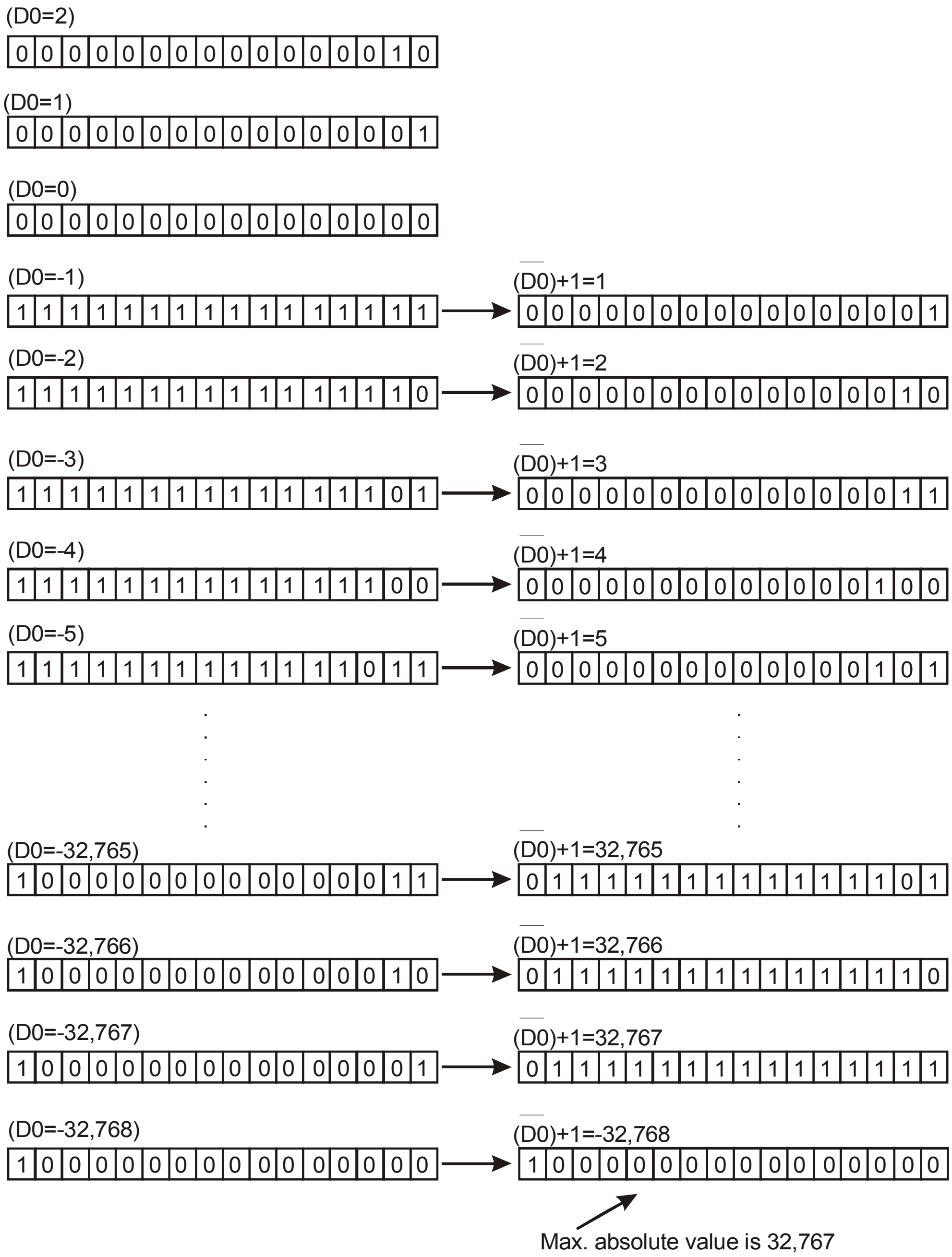
c) If D0 < D2, M2 = ON.

d) D4 is then able to remain positive.



Detailed explanations on negative value and its absolute value

1. MSB = 0 indicates the value is positive while MSB = 1 indicates the value is negative.
2. NEG instruction can be applied to convert a negative value into its absolute value.



API	Mnemonic			Operands	Function	Controllers ES2/EX2 SS2 SA2 SX2
30	D	ROR	P	D n	Rotation Right	

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
D								*	*	*	*	*	*	*	*	ROR, RORP: 5 steps
n					*	*										DROR, DRORP: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

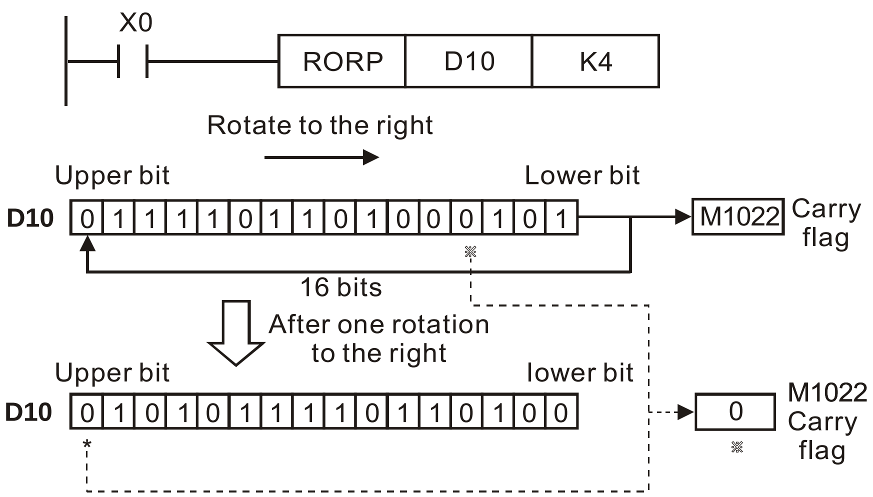
D: Device to be rotated **n**: Number of bits to be rotated in 1 rotation

Explanations:

1. This instruction rotates bit status of the device **D** to the right for **n** bits
2. The status of the last bit rotated (marked with ※) is copied to the carry flag M1022 (Carry flag)
3. This instruction is generally used in pulse execution mode (RORP, DRORP).
4. If operand **D** uses index F, only a 16-bit instruction is available.
5. If operand **D** is specified as KnY, KnM or KnS, only K4 (16-bit) or 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 to ON, the 16 bits (4 bits as a group) in D10 will rotate to the right, as shown in the figure below. The bit marked with ※ will be sent to carry flag M1022..



API	Mnemonic			Operands	Function	Controllers
31	D	ROL	P	(D) (n)	Rotate Left	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ROL, ROLP: 5 steps
D								*	*	*	*	*	*	*	*	DROL, DROLP: 9 steps
n					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

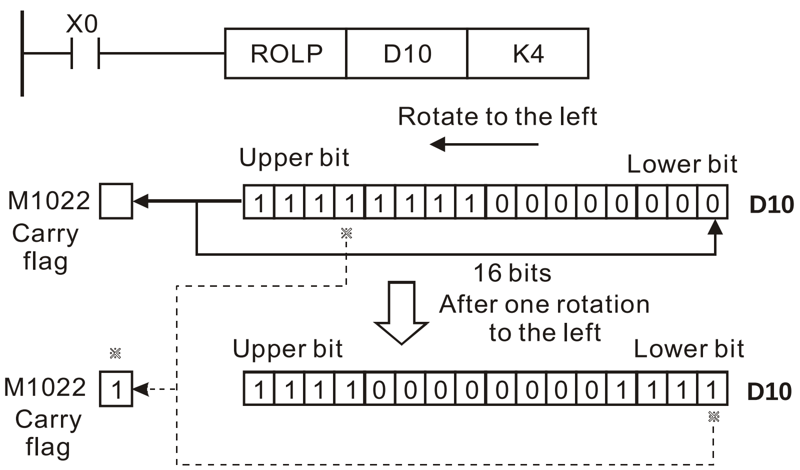
D: Device to be rotated　　**n**: Number of bits to be rotated in 1 rotation

Explanation:

1. This instruction rotates bit status of the device **D** to the left for **n** bits
2. The status of the last bit rotated (marked with ※) is copied to the carry flag M1022.
3. This instruction is generally used in pulse execution mode (ROLP, DROLP).
4. If operand **D** uses index F, only a 16-bit instruction is available.
5. If operand **D** is specified as KnY, KnM or KnS, only K4 (16-bit) or 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 to ON, all the 16 bits (4 bits as a group) in D10 will rotate to the left, as shown in the figure below. The bit marked with ※ will be sent to carry flag M1022.



API	Mnemonic			Operands	Function	Controllers
32	D	RCR	P	D n	Rotation Right with Carry	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	RCR, RCRP: 5 steps
D								*	*	*	*	*	*	*	*	DRCR, DRCRP: 9 steps
n					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

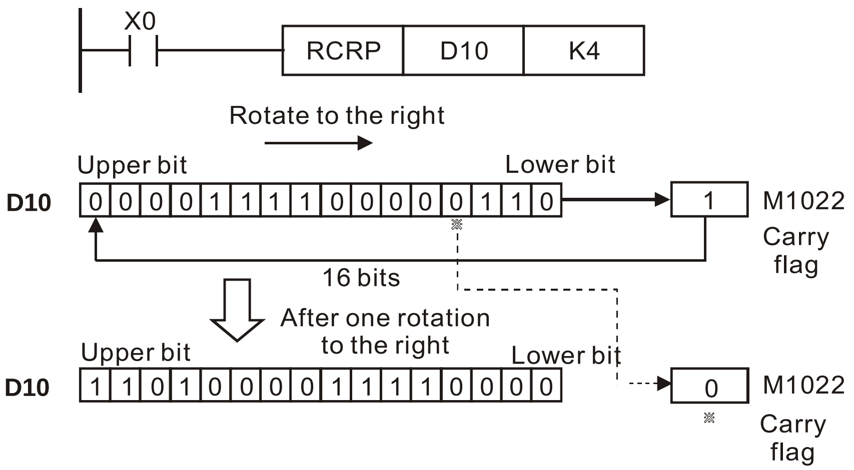
D: Device to be rotated **n**: Number of bits to be rotated in 1 rotation

Explanation:

1. This instruction rotates bit status of the device **D** together with M1022 to the right for **n** bits.
2. The status of the last bit rotated (marked with ※) is moved to the carry flag M1022.
3. This instruction is generally used in pulse execution mode (RCRP, DRCRP).
4. If operand **D** uses index F, only a 16-bit instruction is available.
5. If operand **D** is specified as KnY, KnM or KnS, only K4 (16-bit) or 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 to ON, the 16 bits (4 bits as a group) in D10 together with carry flag M1022 (total 17 bits) will rotate to the right, as shown in the figure below. The bit marked with ※ will be moved to carry flag M1022



API	Mnemonic			Operands	Function	Controllers
33	D	RCL	P	(D) (n)	Rotation Left with Carry	ES2/EX2 | SS2 | SA2 | SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

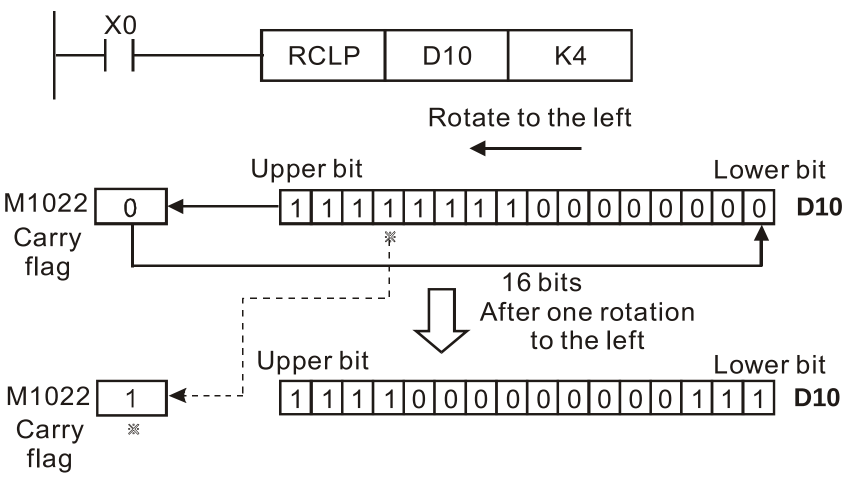
D: Device to be rotated　　**n**: Number of bits to be rotated in 1 rotation

Explanations:

1. This instruction rotates bit status of the device **D** together with M1022 to the left for **n** bits..
2. The status of the last bit rotated (marked with ※) is moved to the carry flag M1022.
3. This instruction is generally used in pulse execution mode (RCLP, DRCLP).
4. If operand **D** uses index F, only a 16-bit instruction is available.
5. If operand **D** is specified as KnY, KnM or KnS, only K4 (16-bit) or 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 to ON, the 16 bits (4 bits as a group) in D10 together with carry flag M1022 (total 17 bits) will rotate to the left, as shown in the figure below. The bit marked with ※ will be sent to carry flag M1022.



API	Mnemonic		Operands	Function	Controllers
34	SFTR	P	S D n1 n2	Bit Shift Right	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices X	Y	M	S	Word devices K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S	*	*	*	*												SFTR, SFTRP: 9 steps
D		*	*	*												
n_1					*	*										
n_2					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Start No. of source device **D**: Start No. of destination device $\mathbf{n_1}$: Length of data to be shifted $\mathbf{n_2}$: Number of bits to be shifted as a group

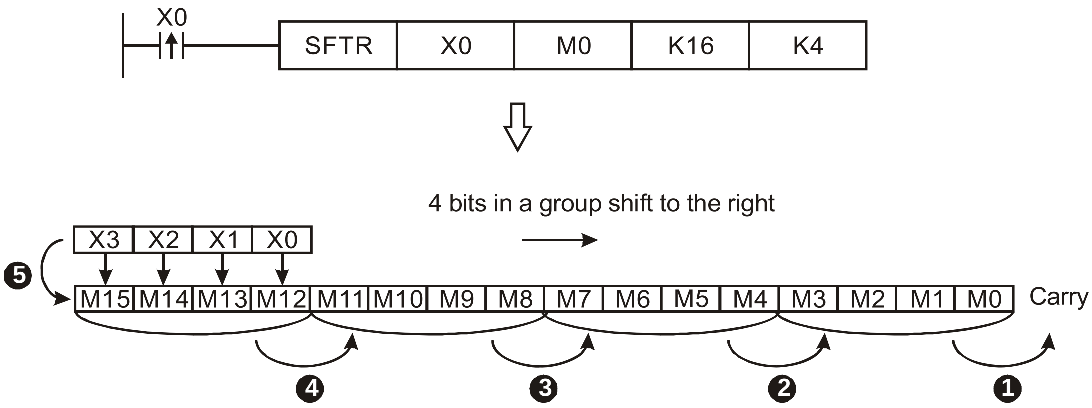
Explanation:

1. This instruction performs a right shift from source device of $\mathbf{n_2}$ bits starting from **S** to destination device of $\mathbf{n_1}$ bits starting from **D.**
2. This instruction is generally used in pulse execution mode (SFTRP).
3. Valid range of operand **n1**, **n2** : 1≤ **n2** ≤ **n1** ≤1024

Program Example:

1. When X0 is rising edge triggered, SFTR instruction shifts X0~X4 into 16 bit data M0~M15 and M0~M15 also shift to the right with a group of 4 bits.
2. The figure below illustrates the right shift of the bits in one scan.

   ❶ M3~M0 → Carry
   ❷ M7~M4 → M3~M0
   ❸ M11~M8 → M7~M4
   ❹ M15~M12 → M11~M8
   ❺ X3~X0 → M15~M12 completed



API		Mnemonic		Operands	Function	Controllers
35		SFTL	P	S D n1 n2	Bit Shift Left	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S	*	*	*	*												SFTL, SFTLP: 9 steps
D		*	*	*												
n_1					*	*										
n_2					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Start No. of source device **D**: Start No. of destination device $\mathbf{n_1}$: Length of data to be shifted $\mathbf{n_2}$: Number of bits to be shifted as a group

Explanations:

1. This instruction performs a left shift from source device of $\mathbf{n_2}$ bits starting from **S** to destination device of $\mathbf{n_1}$ bits starting from **D**
2. This instruction is generally used in pulse execution mode (SFTLP).
3. Valid range of operand **n1**, **n2** : 1≤ **n2** ≤ **n1** ≤1024

Program Example:

1. When X0 is rising edge triggered, SFTL instruction shifts X0~X4 into 16-bit data M0~M15 and M0~M15 also shift to the left with a group of 4 bits.
2. The figure below illustrates the left shift of the bits in one scan

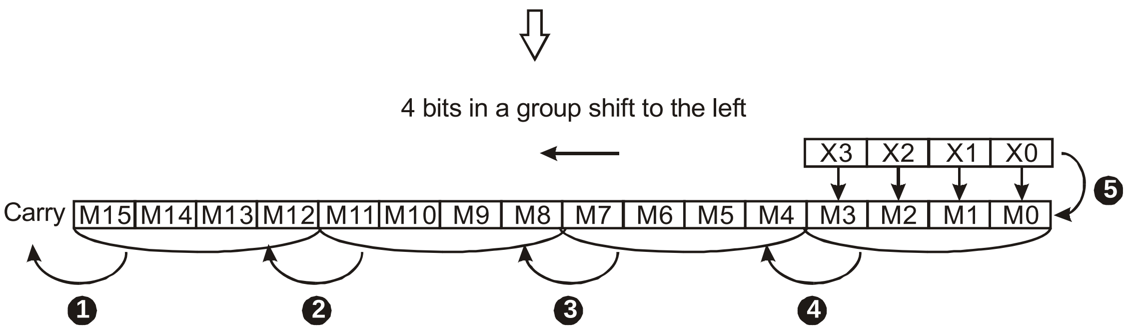
❶ M15~M12 → Carry

❷ M11~M8 → M15~M12

❸ M7~M4 → M11~M8

❹ M3~M0 → M7~M4

❺ X3~X0 → M3~M0 completed



API	Mnemonic		Operands	Function	Controllers
36	WSFR	P	(S) (D) (n1) (n2)	Word Shift Right	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
S							*	*	*	*	*	*	*			WSFR, WSFRP: 9 steps
D								*	*	*	*	*	*			
n_1					*	*										
n_2					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Start No. of source device **D**: Start No. of destination device **n_1**: Length of data to be shifted **n_2**: Number of devices to be shifted as a group

Explanations:

1. This instruction performs a right shift from source device of **n_2** registers starting from **S** to destination device of **n_1** registers starting from **D**.
2. This instruction is generally used in pulse execution mode (WSFRP).
3. The type of devices designated by **S** and **D** has to be the same, e.g. K_nX, K_nY, K_nM, and K_nS as a category and T, C, and D as another category
4. Provided the devices designated by **S** and **D** belong to K_n type, the number of digits of K_n in **S** and **D** has to be the same.
5. Valid range of operand **n1**, **n2** : 1≤ **n2** ≤ **n1** ≤512

Program Example 1:

1. When X0 is triggered, WSFRP instruction shifts D10~D13 into data stack D20~D35 and D20~D35 also shift to the right with a group of 4 registers.
2. The figure below illustrates the right shift of the registers in one scan.

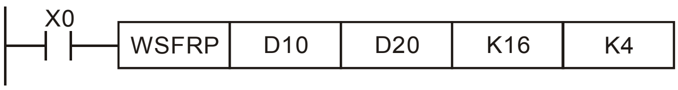
   ❶ D23~D20 → Carry

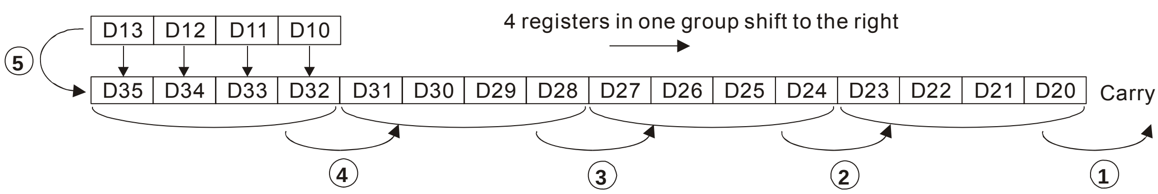
   ❷ D27~D24 → D23~D20

   ❸ D31~D28 → D27~D24

   ❹ D35~D32 → D31~D28

   ❺ D13 ~D10 → D35~D32 completed





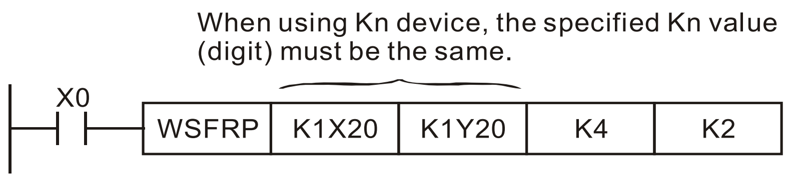
Program Example 2:

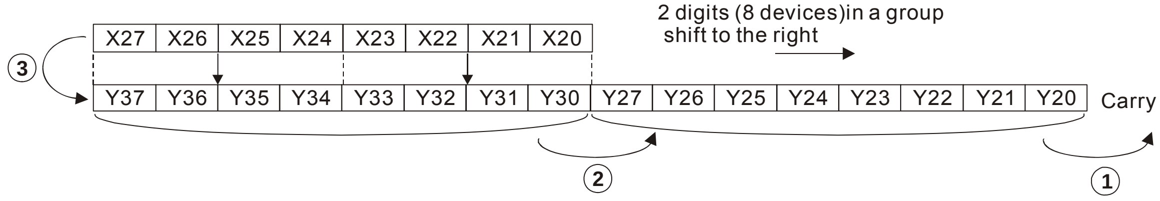
1. When X0 is triggered, WSFRP instruction shifts X20~X27 into data stack Y20~Y37 and Y20~Y37 also shift to the right with a group of 4 devices.
2. The figure below illustrates the right shift of the devices in one scan

   ❶ Y27~Y20 → carry

   ❷ Y37~Y30 → Y27~Y20

   ❸ X27~X20 → Y37~Y30 completed





API	Mnemonic		Operands	Function	Controllers
37	WSFL	P	(S) (D) (n1) (n2)	Word Shift Left	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S							*	*	*	*	*	*	*			WSFL, WSFLP: 9 steps
D								*	*	*	*	*	*			
n_1					*	*										
n_2					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Start No. of source device **D**: Start No. of destination device **n₁**: Length of data to be shifted **n_2**: Number of devices to be shifted as a group

Explanations:

1. This instruction performs a left shift from source device of **n_2** registers starting from **S** to destination device of **n_1** registers starting from **D**.
2. This instruction is generally used in pulse execution mode (WSFLP).
3. The type of devices designated by **S** and **D** has to be the same, e.g. K_nX, K_nY, K_nM, and K_nS as a category and T, C, and D as another category
4. Provided the devices designated by **S** and **D** belong to K_n type, the number of digits of K_n in **S** and **D** has to be the same.
5. Valid range of operand **n1**, **n2** : 1≤ **n2** ≤ **n1** ≤512

Program Example:

1. When X0 is triggered, WSFLP instruction shifts D10~D13 into data stack D20~D35 and D20~D35 also shift to the left with a group of 4 registers.
2. The figure below illustrates the left shift of the words in one scan

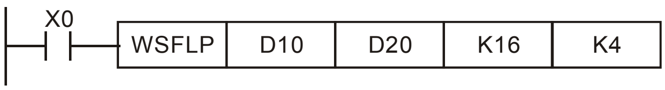
❶ D35~D32 → Carry

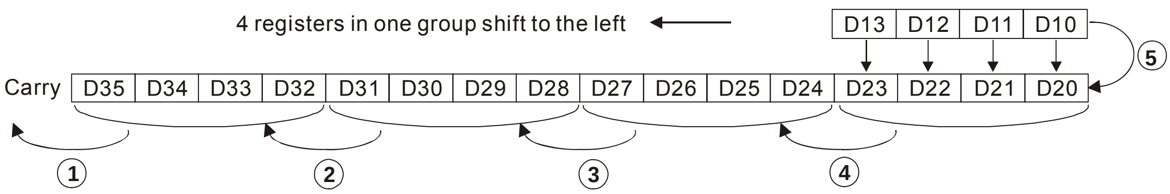
❷ D31~D28 → D35~D32

❸ D27~D24 → D31~D28

❹ D23~D20 → D27~D24

❺ D13~D10 → D23~D20 completed





API	Mnemonic		Operands	Function	Controllers
38	SFWR	P	(S) (D) (n)	Shift Register Write	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device **D**: Head address of data stack **n**: Length of data stack

Explanations:

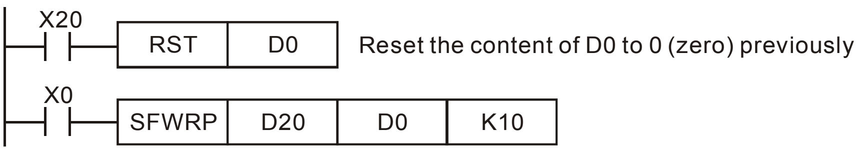
1. This instruction defines the data stack of **n** words starting from **D** as a "first-in, first out (FIFO)" data stack and specifies the first device as the pointer (**D**). When SFWRP is executed, content in pointer pluses 1, and the content in **S** will be written into the device designated by the pointer. When the content in pointer exceeds **n-1**, the instruction stops and carry flag M1022= ON.
2. This instruction is generally used in pulse execution mode (SFWRP).
3. Valid range of operand **n**: 2≤ **n** ≤512

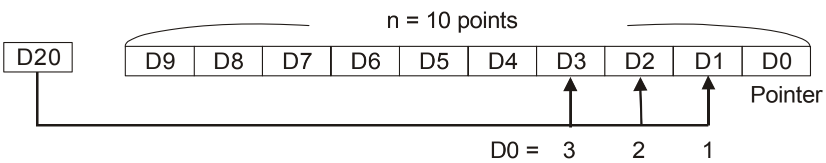
Program Example:

1. First, reset the content of D0. When X0 goes from OFF to ON, the content of D0 (pointer) becomes 1, and D20 is written into D1. If the content of D20 is changed and X0 is triggered again, pointer D0 becomes 2, and the content of D20 is then written into D2.
2. P The figure below illustrates the shift and writing process of the instruction.

   ❶ The content of D0 becomes 1.

   ❷. The content of D20 is written into D1.





Points to note:

This instruction can be used together with API 39 SFRD for the reading/writing of "first-in, first-out" stack data.

API	Mnemonic		Operands	Function	Controllers
39	SFRD	P	(S) (D) (n)	Shift Register Read	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S								*	*	*	*	*	*			SFRD, SFRDP: 7 steps
D								*	*	*	*	*	*	*	*	
n					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

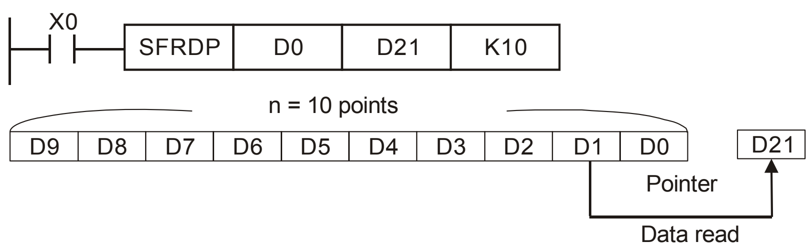
S: Head address of data stack **D**: Destination device **n**: Length of data stack

Explanation:

1. This instruction defines the data stack of **n** words starting from **S** as a FIFO data stack and specifies the first device as the pointer (**S**). The content of pointer indicates current length of the stack. When SFRDP is executed, first data (**S+1**) will be read out to **D**, all data in this stack moves up to fill the read device and content in pointer minuses 1. When the content in pointer = 0, the instruction stops and carry flag M1022= ON
2. This instruction is generally used in pulse execution mode (SFRDP).
3. Valid range of operand **n**: 2≤ **n** ≤512

Program Example:

1. When X0 goes from OFF to ON, D9~D2 are all shifted to the right and the pointer D0 is decremented by 1 when the content of D1 is read and moved to D21.
2. The figure below illustrates the shift and reading of the instruction.
   ❶ The content of D1 is read and moved to D21.
   ❷ D9~D2 are all shifted to the right.
   ❸ The content of D0 is decremented by 1.



API	Mnemonic		Operands	Function	Controllers
40	ZRST	P	D1 D2	Zone Reset	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
																ZRST, ZRSTP: 5 steps
D_1		*	*	*							*	*	*			
D_2		*	*	*							*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

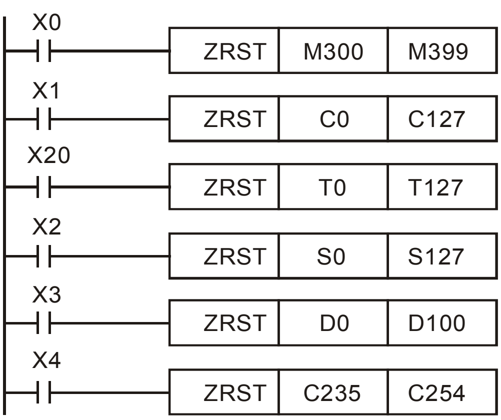
D_1: Starting device of the reset range **D_2**: End device of the reset range

Explanations:

1. When the instruction is executed, range **D_1** to **D_2** will be reset.
2. Operand **D_1** and **D_2** must be the same data type, Valid range: **$D_1 \leqq D_2$**
3. When **D_1** > **D_2**, only operand designated by **D_2** will be reset.
4. This instruction is generally used in pulse execution mode (ZRSTP).

Program Example:

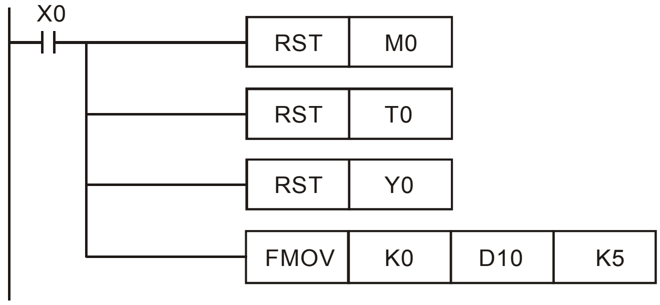
1. When X0 = ON, M300 to M399 will be reset.
2. When X1 = ON, C0 to C127 will all be reset, i.e. present value = 0 and associated contact/ output will be reset as well.
3. When X20 = ON, T0 to T127 will all be reset, i.e. present value = 0 and associated contact/ output will be reset as well.
4. When X2 = ON, the steps of S0 to S127 will be reset.
5. When X3 = ON, the data of D0 to D100 will be reset.
6. When X4 = ON, C235 to C254 will all be reset, i.e. present value = 0 and associated contact/ output will be reset as well.



Points to note:

1. Bit devices Y, M, S and word devices T, C, D can be individually reset by RST instruction.

2. For clearing multiple devices, API 16 FMOV instruction can be used to send K0 to word devices T, C, D or bit devices KnY, KnM, KnS.



API	Mnemonic		Operands	Function	Controllers
41	DECO	P	S D n	Decode	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S	*	*	*	*	*	*					*	*	*	*	*	DECO, DECOP: 7 steps
D		*	*	*							*	*	*	*	*	
n					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

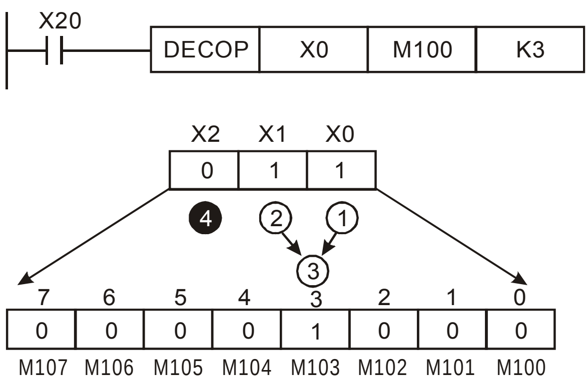
S: Source device to be decoded **D**: Device for storing the result **n**: Number of consecutive bits of **S**

Explanation:

1. The instruction decodes the lower "**n**" bits of **S** and stores the result of "2^n" bits in **D**.
2. This instruction is generally used in pulse execution mode (DECOP).
3. When operand **D** is a bit device, **n** = 1~8, when operand **D** is a word device, **n** = 1~4

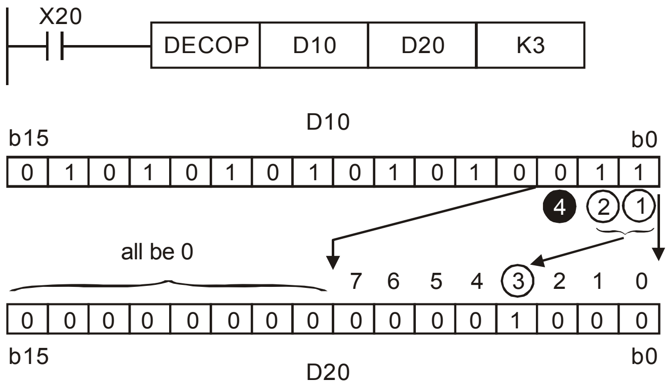
Program Example 1:

1. When **D** is used as a bit device, **n** = 1 ~ 8. Errors will occur if **n** = 0 or **n** > 8.
2. If **n** = 8, the decoded data is 2^8= 256 bits data.
3. When X20 goes from OFF to ON, the data of X0~X2 will be decoded to M100~M107.
4. If the source data is 3, M103 (third bit from M100) = ON.
5. After the execution is completed, X20 is turned OFF. The decoded results or outputs will retain their operation.



Program Example 2:

1. When **D** is used as a word device, n = 1 ~ 4. Errors will occur if n = 0 or n > 4.
2. When **n** = 4, the decoded data is 2^4 = 16 bits.
3. When X20 goes from OFF to ON, the data in D10 (b2 to b0) will be decoded and stored in D20 (b7 to b0). The unused bits in D20 (b15 to b8) will be set to 0.
4. The lower 3 bits of D10 are decoded and stored in the lower 8 bits of D20. The higher 8 bits of D20 are all 0.
5. After the execution is completed, X20 is turned OFF. The decoded results or outputs will retain their operation.



API	Mnemonic			Operands	Function	Controllers
42		ENCO	P	S D n	Encode	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
																DECO, DECOP: 7 steps
S	*	*	*	*							*	*	*	*	*	
D											*	*	*	*	*	
n					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

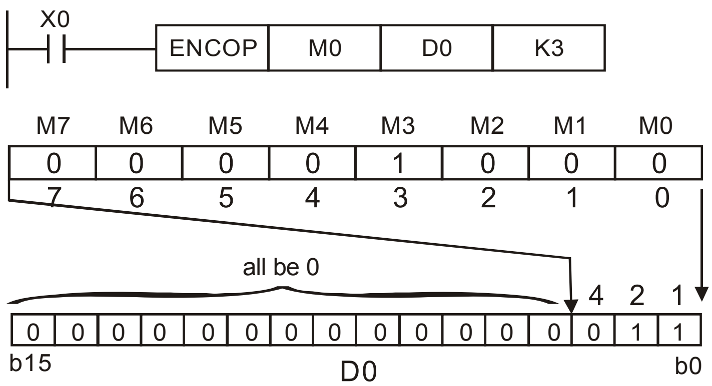
S: Source device to be encoded **D**: Device for storing the result **n**: Number of consecutive bits of **S**

Explanation:

1. The instruction encodes the lower "2^n" bits of source **S** and stores the result in **D**.
2. They highest active bit in **S** has the priority for encoding operation.
3. This instruction is generally used in pulse execution mode (ENCOP).
4. When operand **S** is a bit device, **n**=1~8, when operand **S** is a word device, **n**=1~4
5. If no bits in **S** is active (1), M1067, M1068 = ON and D1067 records the error code 0E1A (hex).

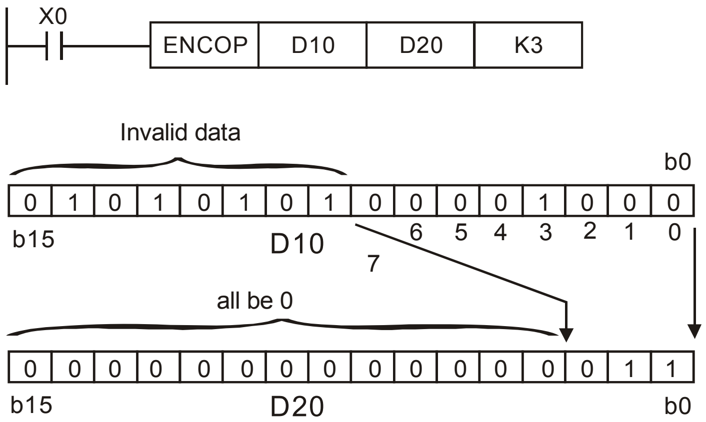
Program Example 1:

1. When **S** is used as a bit device, **n** = 1 ~ 8. Errors will occur if **n** = 0 or **n** > 8.
2. f **n** = 8, the decoded data is 2^8= 256 bits data.
3. When X0 goes from OFF to ON, the data in (M0 to M7) will be encoded and stored in lower 3 bits of D0 (b2 to b0). The unused bits in D0 (b15 to b3) will be set to 0.
4. After the execution is completed, X0 is turned OFF and the data in **D** remains unchanged.



Program Example 2:

1. When **S** is used as a word device, n = 1 ~ 4. Errors will occur if n = 0 or n > 4.
2. When **n** = 4, the decoded data is 2^4 = 16 bits data.
3. When X0 goes from OFF to ON, the 2^3 bits (b0 ~ b7) in D10 will be encoded and the result will be stored in the lower 3 bits of D20 (b2 to b0). The unused bits in D20 (b15 to b3) will be set to 0.
4. After the execution is completed, X0 is turned OFF and the data in **D** remains unchanged



API	Mnemonic			Operands	Function	Controllers
43	D	SUM	P	(S) (D)	Sum of Active bits	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

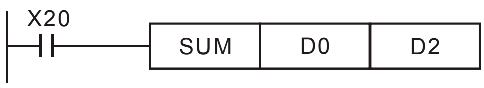
S: Source device **D**: Destination device for storing counted value

Explanation:

1. This instruction counts the total active bits in **S** and store the value in **D.**
2. **D** will occupy two registers when using in 32-bit instruction.
3. If operand **S**, **D** use index F, only a 16-bit instruction is available.
4. If there is no active bits, zero flag M1020 =ON.

Program Example:

When X20 = ON, all active bits in D0 will be counted and the result will be stored in D2.



0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0

D0

3

D2

API	Mnemonic				Operands	Function	Controllers
44	D	BON	P		S D n	Check specified bit status	ES2/EX2 SS2 SA2 SX2

Type OP	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
S					*	*	*	*	*	*	*	*	*	*	*	DBON, DBONP: 13 steps
D		*	*	*												
n					*	*					*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

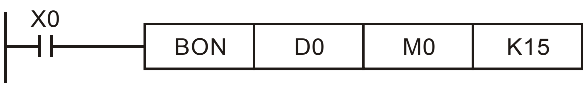
S: Source device **D**: Device for storing check result **n**: Bit number to be checked

Explanation:

1. The instruction checks the status of designated bit (specified by **n**) in **S** and stores the result in **D**
2. If operand **S** uses index F, only 16-bit instruction is available.
3. Valid range of operand **n** : **n** = 0~15 (16-bit), **n** = 0~31 (32-bit)

Program Example:

1. When X0 = ON, and bit15 of D0 = “1”, M0 will be ON. If the bit15 is “0”, M0 is OFF.
2. When X0 is OFF, M0 will retain its previous status.



b15															b0	
0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	M0=Off

D0

b15															b0	
1	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	M0=On

D0

API	Mnemonic			Operands	Function	Controllers
45	D	MEAN	P	S D n	Mean	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MEAN, MEANP: 7 steps
S							*	*	*	*	*	*	*			DMEAN, DMEANP: 13
D								*	*	*	*	*	*	*	*	steps
n					*	*	*	*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

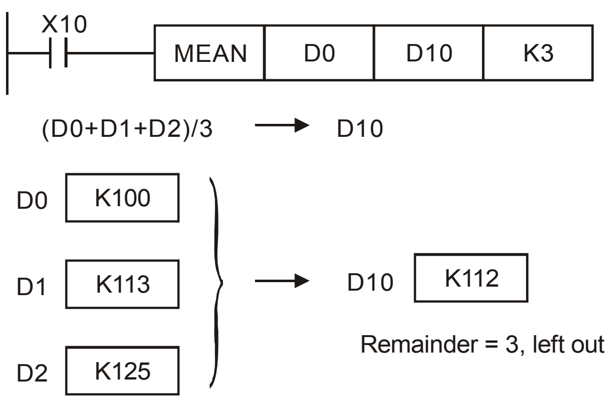
S: Source device **D**: Destination for storing result **n**: Number of consecutive device from **S**

Explanations:

1. The instruction obtains the mean value from **n** consecutive registers from **S** and stores the value in **D**.
2. Remainders in the operation will be ignored.
3. If **S** is not within the valid range, only those addresses within the valid range will be processed.
4. If **n** is out of the valid range (1~64), PLC will determine it as an "instruction operation error".
5. If operand **D** uses index F, only a 16-bit instruction is available.
6. Valid range of operand **n** : **n** = 1~64

Program Example:

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



API	Mnemonic	Operands	Function	Controllers
46	ANS	S m D	Timed Annunciator Set	ES2/EX2 SS2 SA2 SX2

Type / OP	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				*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Alarm timer **m**: Time setting **D**: Alarm

Explanations:

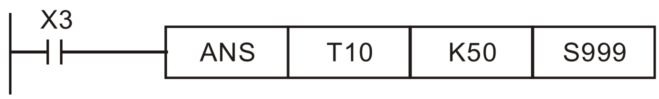
1. ANS instruction is used to drive the output alarm device in designated time.
2. Operand **S** valid range: T0~T183

   Operand **m** valid range: K1~K32,767 (unit: 100 ms)

   Operand **D** valid range: S912~S1023
3. Flag: M1048 (ON: Alarm is active), M1049 (ON: Alarm monitoring is enabled)
4. See ANR instruction for more information

Program Example:

If X3 = ON for more than 5 sec, alarm step relay S999 will be ON. S999 will remains ON after X3 is reset. (T10 will be reset, present value = 0)



API	Mnemonic			Function	Controllers			
47		ANR	P	Annunciator Reset	ES2/EX2	SS2	SA2	SX2

OP	Descriptions	Program Steps
N/A	Instruction driven by contact is necessary.	ANR, ANRP: 1 steps

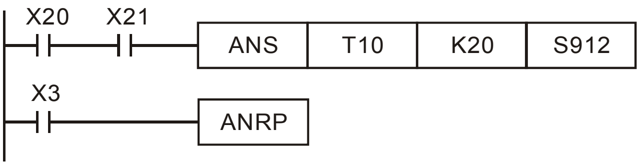
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Explanations:

1. ANR instruction is used to reset an alarm.
2. When several alarm devices are ON, the alarm with smaller number will be reset.
3. This instruction is generally used in pulse execution mode (ANRP).

Program Example:

1. If X20 and X21 are ON at the same time for more than 2 sec, the alarm S912 will be ON. If X20 or X21 is reset, alarm S912 will remain ON but T10 will be reset and present value is cleared.
2. If X20 and X21 are ON less than 2 sec, the present value of T10 will be cleared.
3. When X3 goes from OFF → ON, activated alarms S912 will be reset.
4. When X3 goes from OFF → ON again, the alarm device with second lower number will be reset.



Points to note:

Flags:

1. M1048 (indicating alarm status): When M1049 = ON, enabling any of the alarm S912~S1023 turns M1048 ON.
2. M1049 (Enabling alarm monitoring): When M1049 = ON, D1049 will automatically hold the lowest alarm number in active alarms.

Application example of alarm device (production line):

X0 = Forward switch　　X1 = Backward switch

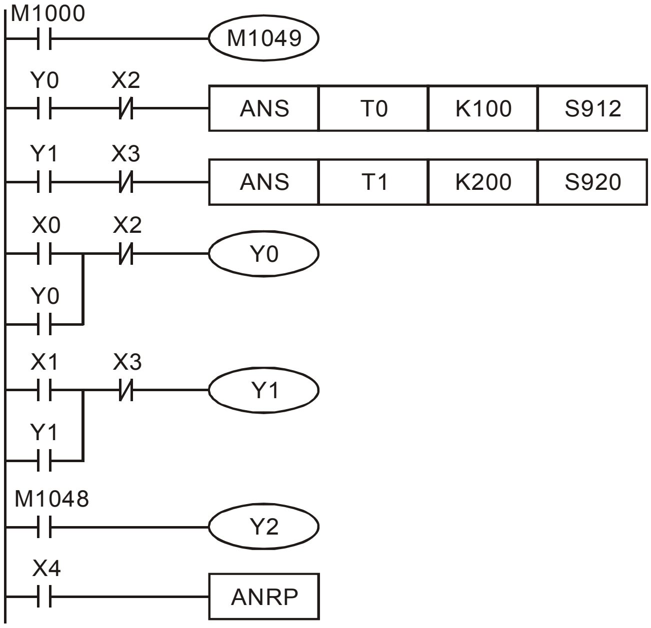
X2 = Front position switch　　X3 = Back position switch

X4 = Alarm reset button

Y0 = Forward　　Y1 = Backward

Y2 = Alarm indicator

S912 = Forward alarm　　S920 = Backward alarm



1. M1048 and D1049 are valid only when M1049 = ON.
2. When Y0 = ON for more than 10 sec and the product fails to reach the front position X2, S912 = ON
3. When Y1 = ON for more than 10 sec and the product fails to reach the back position X3, S920= ON.
4. When backward switch X1 = ON and backward device Y1 = ON, Y1 will go OFF only when the product reaches the back position switch X3.
5. Y2 is ON when any alarm is enabled.
6. Whenever X4 is ON, 1 active alarm will be reset. If there are several active alarms, the reset will start from the alarm with the lowest number and then the alarm with second lower number, etc.

API	Mnemonic			Operands	Function	Controllers
48	D	SQR	P	(S) (D)	Square Root	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SQR, SQRP: 5 steps
S					*	*							*			DSQR, DSQRP: 9 steps
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device **D**: Device for storing the result

Explanation:

1. This instruction performs a square root operation on **S** and stores 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. The operation result **D** should be integer only, and the decimal will be left out. When decimal is left out, borrow flag M1021 = ON.
4. When the operation result **D** = 0, zero flag M1020 = ON.

Program Example:

When X20 = ON, square root of D0 will be stored in D12.

$\sqrt{D0} \rightarrow D12$

API	Mnemonic			Operands	Function	Controllers
49	D	FLT	P	(S) (D)	Floating Point	ES2/EX2 SS2 SA2 SX2

Type / OP	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
S													*			DFLT, DFLTP: 9 steps
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

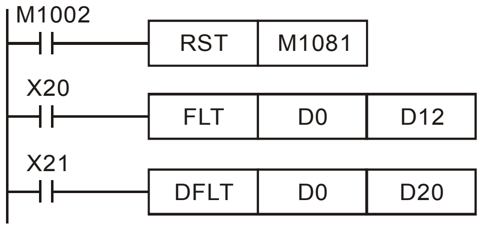
S: Source device **D**: Device for storing the conversion result

Explanations:

1. When M1081 = OFF, the source **S** is converted from BIN integer to binary floating point value. At this time, 16-bit instruction FLT occupies 1 register for **S** and 2 registers for **D**.
   a) If the absolute value of the conversion result ＞ max. floating value, carry flag M1022 = ON.
   b) If the absolute value of the conversion result ＜ min. floating value, carry flag M1021 = ON.
   c) If conversion result is 0, zero flag M1020 = ON.
2. When M1081 is ON, the source **S** is converted from binary floating point value to BIN integer. (Decimal ignored). At this time, 16-bit instruction FLT occupies 2 registers for **S** and 1 register for **D**. The operation is same as instruction INT.
   a) If the conversion result exceeds the available range of BIN integer in **D** (for 16-bit: -32,768 ~ 32,767; for 32-bit: -2,147,483,648 ~ 2,147,483,647), **D** will obtain the maximum or minimum value and carry flag M1022 = ON.
   b) If the decimal is ignored, borrow flag M1021=ON.
   c) If the conversion result = 0, zero flag M1020=ON.
   d) After the conversion, **D** stores the result in 16 bits.

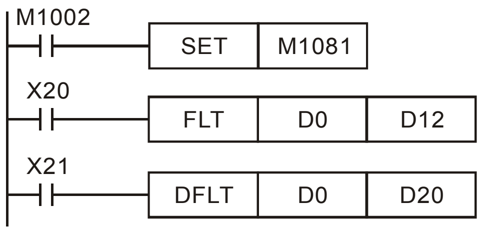
Program Example 1:

1. When M1081 = OFF, the BIN integer is converted into binary floating point value.
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. Assume D0 is K10. When X10 is ON, the converted 32-bit value will be H41200000 and stored in 32-bit register D12 (D13)
5. If 32-bit register D0 (D1)=K100,000, X21 = ON. 32-bit of floating point after conversion will be H47C35000 and it will be saved in 32-bit register D20 (D21)



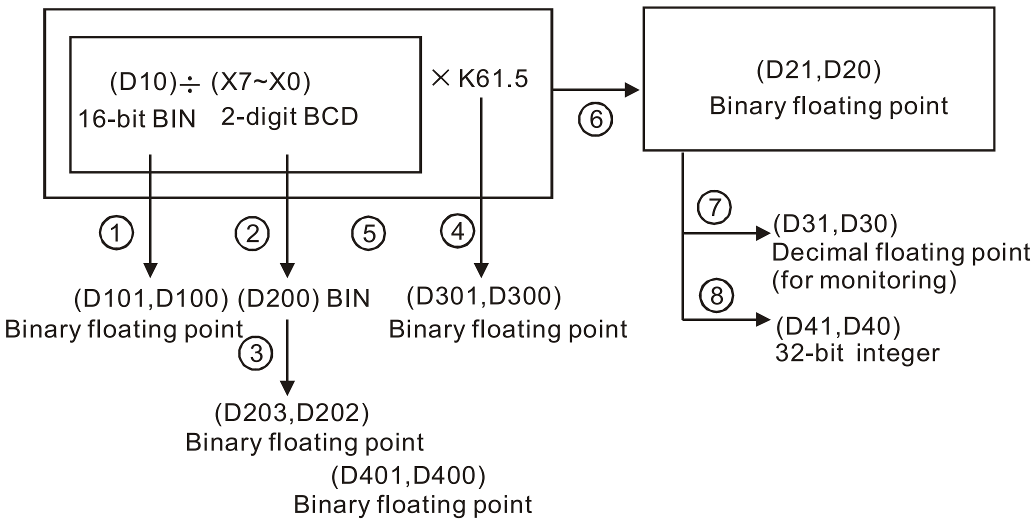
Program Example 2:

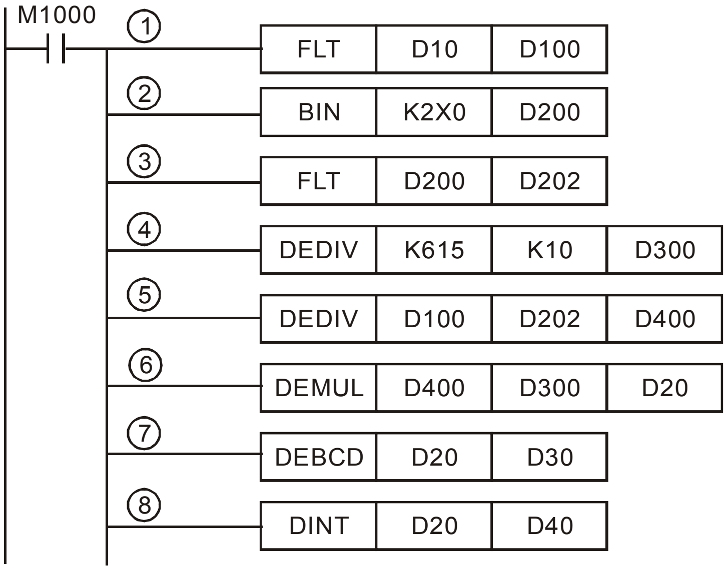
1. When M1081 = ON, the source data is converted from floating point value to BIN integer. (Decimal ignored)
2. When X20 = ON, D1 and D0 (floating point) are converted to D12 (BIN integer). If D0 (D1) = H47C35000, the result will be 100,000 which exceeds the available range of BIN integer in 16-bit register D12. In this case the result will be D12 = K32767, and M1022 = ON
3. When X21 = ON, D1 and D0 (floating point) are converted to D21, D20 (BIN integer). If D0 (D1) = H47C35000, the result is 100,000 and will be saved in 32-bit register D20 (D21).



Program Example 3:

Apply FLT instruction to complete the following operation





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 ÷ K10 to D301, D300 (floating point).
5. Divide the floating point:

   Save the result of (D101, D100) ÷ (D203, D202) to D401, D400 (floating point).
6. Multiply floating point:

   Save the result of (D401, D400) × (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	D n	Refresh	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

D: Start device for I/O refresh **n**: Number of devices for I/O refresh

Explanations:

1. PLC updates I/O status between END instruction and the start of next program scan. If an immediate I/O refresh is needed, REF can be applied for performing I/O refresh immediately.
2. **D** can only be a multiple of 10, i.e. X0 or Y0, and the instruction is NOT applicable for I/O points on DIO modules.
3. Only the I/O points on MPU can be specified for operand D for I/O refresh.
   - When **D** specifies X0 and **n** ≦ 8, only X0~X7 will be refreshed. If **n** > 8, all I/O points on MPU will be refreshed.
   - When **D** specifies Y0 and **n** = 8, only Y0~X7 will be refreshed. If **n** > 8, all I/O points on MPU will be refreshed.
   - When **D** specifies X10 or Y10, I/O points on MPU except for X0~X7 or Y0~Y3 will all be refreshed regardless of **n** value, i.e. only status of X0~X7 or Y0~Y3 remains.
4. For EX2/SX2 MPU only: If M1180 = ON and REF instruction executes, PLC will read the A/D value and update the read value to D1110~D1113. If M1181 = ON and REF instruction executes, PLC will output the D/A value in D1116 and D1117 immediately. When A/D or D/A values are refreshed, PLC will reset M1180 or M1181 automatically.
5. Range for **n (ES2/EX2):** 4 ~ total I/O points on MPU. **n** should always be a multiple of 4.
6. Range for **n (SS2/SA2/SX2):** 8 ~ total I/O points on MPU.

Program Example 1:

When X0 = ON, PLC will refresh the status of input points X0 ~ X7 immediately without delay.

Program Example 2:

When X0 = ON, the 4 output signals on Y0 ~ Y3 will be sent to output terminals immediately before the program proceeds to END instruction.

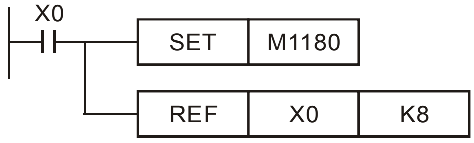
Program Example 3:

When X0 = ON, I/O points starting from X10 or Y4 will all be refreshed.

Or

Program Example 4:

For DVP-EX2/SX2 only: When X0 = ON and M1180 = ON, A/D signal in D1110~D1113 will be refreshed immediately regardless of the settings of operands **D** and **n**



API	Mnemonic		Operands	Function	Controllers
51	REFF	P	n	Refresh and Filter Adjust	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
n					*	*										REFF, REFFP: 3 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

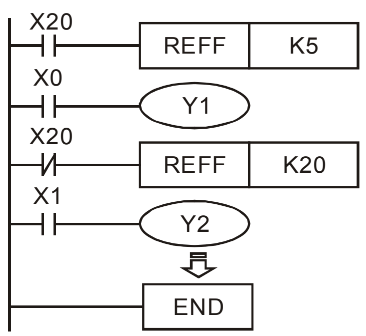
n: Response time (unit: ms)

Explanation:

1. PLC provides digital input filters to avoid interference. The response time (**n)** of X0 ~ X7 input filters can be adjusted by REFF instruction. The instruction sets the value specified in **n** to D1020 (X0 ~ X7 input filter time) directly.
2. When PLC turns from OFF to ON or the END instruction is reached, the response time is dictated by the value of D1020.
3. During program execution, the value in D1020 can be changed by using MOV instruction.
4. When using REFF instruction during program execution, the modified response time will be move to D1020 and refreshed until next program scan..
5. Range of **n**: = K2 ~ K20.

Program Example:

1. When the power of PLC turns from OFF to ON, the response time of X0~X7 inputs is specified by the value in D1020.
2. When X20 = ON, REFF K5 instruction is executed, response time changes to 5 ms and takes affect the next scan.
3. When X20 = OFF, the REFF instruction will not be executed, the response time changes to 20ms and takes affect the next scan.



Points to note:

Response time is ignored (no delay) when input points are occupied by external interrupts, high-speed counters or SPD instruction.

API	Mnemonic	Operands	Function	Controllers
52	MTR	S D1 D2 n	Input Matrix	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MTR: 9 steps
S	*															
D_1		*														
D_2		*	*	*												
n					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

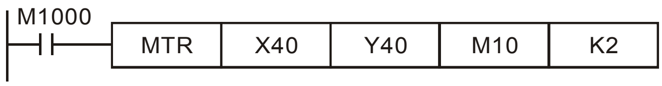
S: Head address of input device **D_1**: Head address of output device **D_2**: Head address of matrix scan **n**: Number of arrays in the matrix

Explanations:

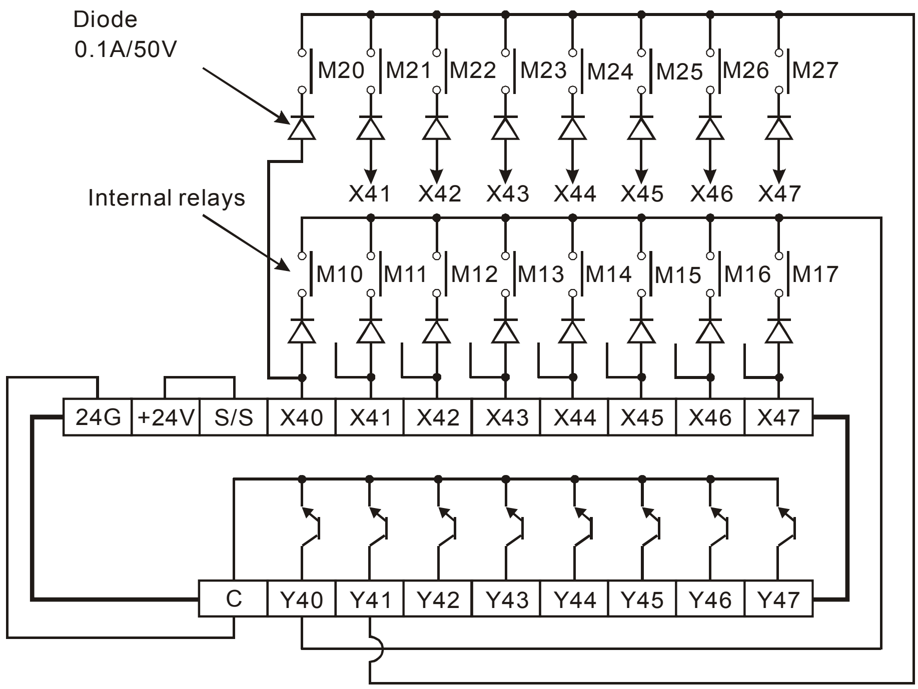
1. **S** is the source device of the matrix input and occupies 8 consecutive points.
   D_1 is the trigger device (transistor output Y) to read input signals and occupies **n** consecutive points
   D_2 is the head address of the matrix which stores the read status from inputs
2. This instruction allows 8 continuous input devices starting from **S** to be used **n** times, which means the operation result can be displayed with a matrix table starting from **D_2**. Each set of 8 input signals are grouped into an "array" and there are **n** number of arrays. Each array is selected to be read by triggering output devices starting from **D_1**. The result is stored in a matrix-table which starts at corresponding head address **D_2**.
3. Maximum 8 arrays can be specified (**n** = 8) to obtain 64 input points (8 × 8 = 64).
4. The processing time of each array is approximately 25ms, i.e. an 8 array matrix would cost 200ms to finish reading. In this case, input signals with ON/OFF speed faster than 200ms are not applicable in the matrix input.
5. It is recommended to use special auxiliary relay M1000 (normally open contact).
6. Whenever this instruction finishes a matrix scan, M1029 will be ON for one scan period..
7. There is no limitation on the number of times for using the instruction, but only one instruction can be executed in the same time.
8. Flag: M1029, execution completed flag.

Program Example:

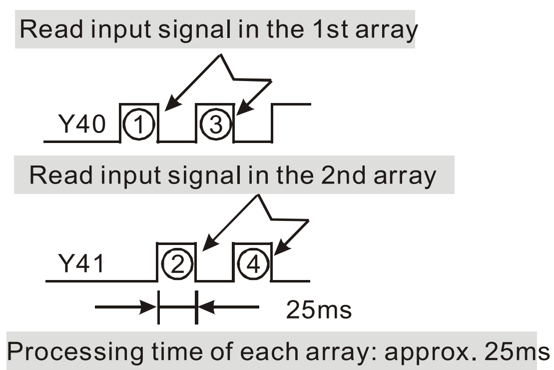
When PLC runs, MTR instruction executes. The status of input points X40~X47 is read 2 times in the driven order of output points Y40 and Y41, i.e. 16 signals will be generated and stored in internal relay M10~M17 and M20~M27.



The figure below illustrates the external wiring of the 2-array matrix input loop constructed by X40 ~ X47 and Y40 ~ Y41. The 16 switches correspond to the internal relays M10 ~ M17, M20 ~ M27. The wiring should be applied with MTR instruction.



When output Y40 is ON, only inputs in the first array are read. The results are stored in auxiliary relays M10~M17. After Y40 goes OFF, Y41 turns ON. This time only inputs in the second array are read. The results are stored in M20~M27.



Points to note:

1. Operand **S** must be a multiple of 10, e.g. 00, 10, 20, which means X0, X10… etc. and occupies 8 continuous devices.
2. Operand **D_1** should be a multiple of 10, i.e. 00, 10, 20, which means Y0, Y10… etc. and occupies **n** continuous devices
3. Operand **D_2** should be a multiple of 10, i.e. 00, 10, which means M0, M10, S0, S10… etc.
4. Valid range of **n** = 2~8

API	Mnemonic			Operands	Function	Controllers
53	D	HSCS		(S1) (S2) (D)	High Speed Counter Set	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
S_1					*	*	*	*	*	*	*	*	*	*		DHSCS: 13 steps
S_2												*				
D		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Comparative value **S_2**: No. of high-speed counter **D**: Compare result

Explanations:

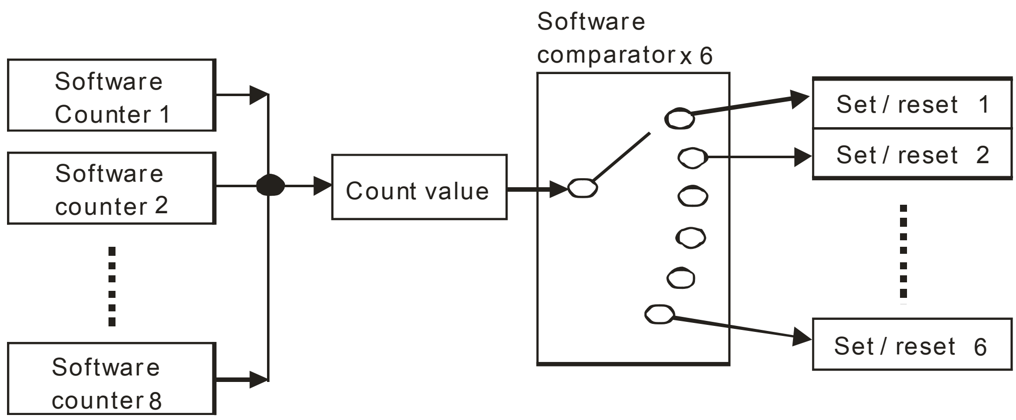
1. Functions related to high-speed counters adopt an interrupt process; therefore, devices specified in **D** which indicates comparison results are updated immediately. This instruction compares the present value of the designated high-speed counter **S_2** against a specified comparative value **S_1**. When the current value in counters equals **S_1**, device in **D** will be ON even when values in **S_1** and **S_2** are no longer equal.
2. If **D** is specified as Y0~Y3, when the instruction is executed and the count value equals to **S_1** , the compare result will immediately output to the external outputs Y0~Y3. However, other Y outputs will still be updated till the end of program. Also, M and S devices, not affected by the program scan time, will be immediate updated as the Y devices specified by this instruction.
3. Operand **D** can designate I0□0, □=1~8
4. High speed counters include software high speed counters and hardware high speed counters. In addtiion, there are also two types of comparators including software comparators and hardware comparators. For detailed explanations of high speed counters please refer to section 2.9 in this manual.
5. Explanations on software comparators for DHSCS/DHSCR instruction:
   - ➢ There are 6 software comparators available corresponding to associated high speed counter interrupts. Numbers of the applied interrupts should also be specified correctly in front of the associated interrupt subroutines in the program.
   - ➢ When programming DHSCS and DHSCR instructions, the total of Set/Reset comparisons for both instructions can not be more than 6, otherwise syntax check error will occur.

- Table of settings for software counters and software comparators:

Counter	C232	C233	C234	C235	C236	C237
DHSCS Hi-speed interrupt	I010	I050	I070	I010	I020	I030
Hi-speed compare Set / Reset	C232~C242 share 6 software comparators					

Counter	C238	C239	C240	C241	C242
DHSCS Hi-speed interrupt	I040	I050	I060	I070	I080
Hi-speed compare Set / Reset	C232~C242 share 6 software comparators				

- DVP-SS2 does not support the software high speed counter C232.
- Block diagram of software counters and comparators:

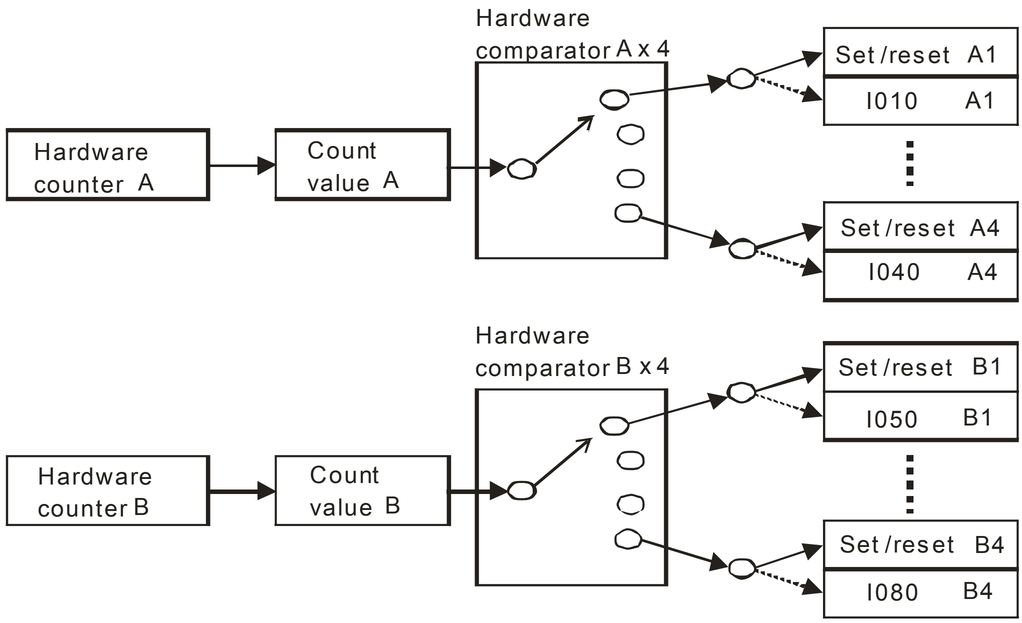


6. Explanations on hardware comparators DHSCS/DHSCR instruction:
   - There are 2 groups of hardware comparators provided respectively for 2 groups of hardware counters (A group and B group), and each group shares 4 comparators with individual Compare Set/Reset function.
   - When programming DHSCS and DHSCR instructions, the total of Set/Reset comparisons for both instructions can not be more than 4, otherwise syntax check error will occur.
   - Each high-speed counter interrupt occupies an associated hardware comparator, consequently the interrupt number can not be repeated. Also, I010~I040 can only be applied for group A comparators and I050~I080 for group B.
   - If DCNT instruction enables C243 as high speed counter (group A) and DHSC/DHSC instruction uses C245 as high speed counter (group A) at the same time, PLC takes C243 as the source counter automatically and no syntax check error will be detected.

➢ Table of settings for hardware counters and comparators:

Hardware counter	A group				B group			
	A1	A2	A3	A4	B1	B2	B3	B4
Counter No.	C243, C245~C248, C251,C252				C244, C249, C250, C253, C254			
High-speed counter interrupt	I010	I020	I030	I040	I050	I060	I070	I080
Hi-speed compare Set/Reset	Share 4 hardware comparators for group A				Share 4 hardware comparators for group B			

➢ Block diagram of hardware counters and comparators:



7. Difference between software and hardware comparators:

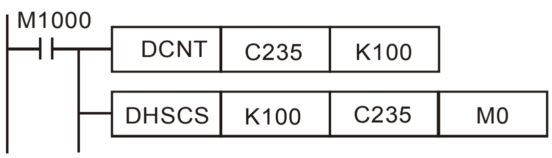
➢ 6 comparators are available for software counters while 8 comparators are available for 2 groups of hardware counters (4 comparators for each group)

➢ Output timing of software comparator → count value equals to comparative value in both counting up/down modes.

➢ Output timing of hardware comparator → count value equals to comparative value+1 in counting-up mode; count value equals to comparative value -1 in counting-down mode.

Program Example 1:

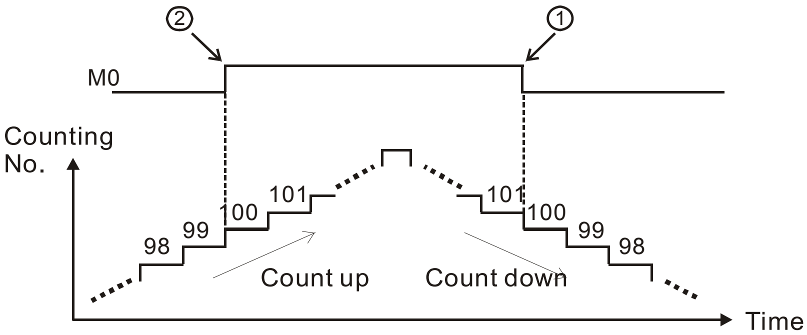
Set/reset M0 by applying software comparator



➢ When value in C235 varies from 99 to100, DHSCS instruction sets M0 ON. (M1235 =

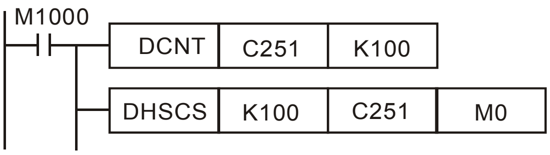
OFF, C235 counts up)

- When value in C235 varies from 101 to100, DHSCR instruction resets M0. (M1235 = ON, C235 counts down)
- Timing diagram for the comparison:

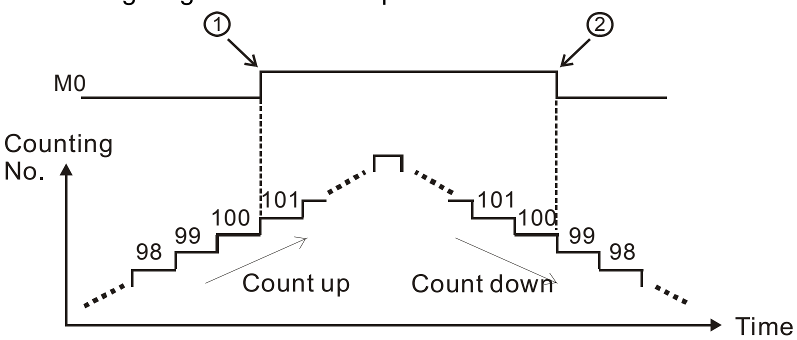


Program Example 2:

Set/reset M0 by applying hardware comparator

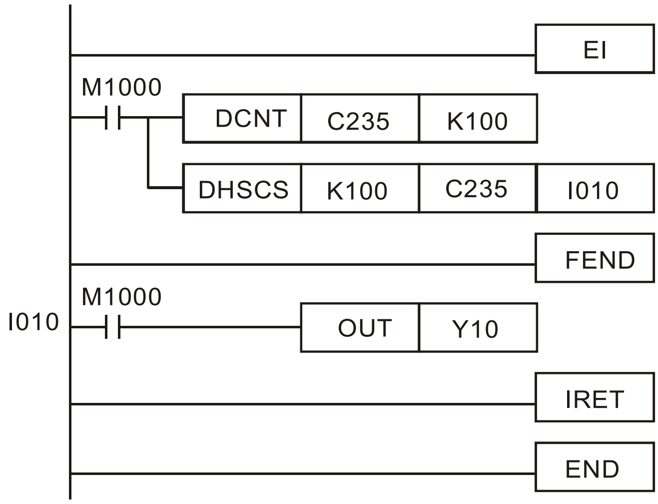


- When C251 counts up and the value in C251 varies from 100 to101, DHSCS instruction sets M0 ON.
- When C251 counts down and the value in C251 varies from 100 to 99, DHSCR instruction resets M0.
- Timing diagram for the comparison:



Program Example 3:

Executes interrupt subroutine by applying software comparator.



- ➢ When value in C235 varies from 99 to100, interrupt subroutine triggered by I010 executes immediately to set Y0 ON.

Points to note:

- ➢ If operand **D** is specified as S, M or Y0~Y3 for the above high speed comparison, the compare result will immediately output to the external points Y0~Y3 (Y0~Y5 for SS2/SX2). However, if D is specified as Y4~Y337, external outputs will be updated till the end of program (delay for one scan cycle).

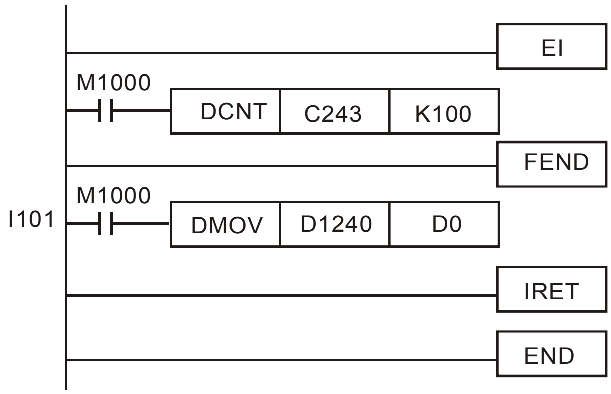
8. Count value storage function of high speed interrupt:
   - ➢ When X1, X3, X4 and X5 is applied for reset function and associated external interrupts are disabled, users can define the reset function as Rising/Falling-edge triggered by special M relays specified in the table: Applicable Software High Speed Counters. However, if external interrupts are applied, the interrupt instructions have the priority in using the input points. In addition, PLC will move the current data in the counters to the associated data registers below then reset the counters
   - ➢ When X0 (counter input) and X1 (external Interrupt I100/I101) work with C243, the count value will be moved to D1240 and D1241 when interrupt occurs and then the counter will be reset.
   - ➢ When X2 (counter input) and X3 (external Interrupt I300/I301) work with C244, the count value will be moved to D1242 and D1243 when interrupt occurs and then the counter will be reset.
   - ➢ When X0 (counter input) and X4 (external Interrupt I400/I401) work with C246, C248, C252, the count value will be moved to D1240 and D1241 when interrupt occurs and

then the counter will be reset.

- When X2 (counter input) and X5 (external Interrupt I500/I501) work with C244, C250, C254, the count value will be moved to D1242 and D1243 when interrupt occurs and then the counter will be reset.

Special D	D1241, D1240				D1243, D1242		
Counter	C243	C246	C248	C252	C244	C250	C254
Interrupt	X1(I100/I101)	X4(I400/I401)			X3(I300/I301)	X5(I500/I501)	

Program Example 4:



- If interrupt I101 is triggered from input point X1 while C243 is counting, I101 interrupt subroutine executes immediately and the count value in C243 will be moved to D0. After this, C243 is reset.

API	Mnemonic		Operands	Function	Controllers
54	D	HSCR	S₁ S₂ D	High Speed Counter Reset	ES2/EX2 SS2 SA2 SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Comparative value **S_2**: No. of high speed counter **D**: Comparison result

Explanations:

1. DHSCR compares the current value of the counter **S_2** against a compare value **S_1**. When the counters current value changes to a value equal to **S_1**, then device **D** is reset to OFF. Once reset, even if the compare result is no longer unequal, **D** will still be OFF.
2. If **D** is specified as Y0~Y3 in this instruction, the compare result will immediately output to the external outputs Y0~Y3 (reset the designated Y). However, other Y outputs will still be updated till the end of program (delay for one scan cycle). Also, M and S devices, not affected by the program scan time, will be immediately updated as well.
3. Operand **D** can be specified with high speed counters C232~C254 (SS2 does not support C232) the same as **S_2**. .
4. High speed counters include software high speed counters and hardware high speed counters. In addtiion, there are also two types of comparators including software comparators and hardware comparators. For detailed explanations of high speed counters please refer to section 2.9 in this manual.
5. For explanations on software counters and hardware counters, please refer to API53 DHSCS.
6. For program examples, please refer to Program Example1 and 2 in API53 DHSCS.

API	Mnemonic		Operands	Function	Controllers
55	D	HSZ	(S1) (S2) (S) (D)	High Speed Zone Compare	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
																DHSZ: 17 steps
S_1					*	*	*	*	*	*	*	*	*	*		
S_2					*	*	*	*	*	*	*	*	*	*		
S												*				
D		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

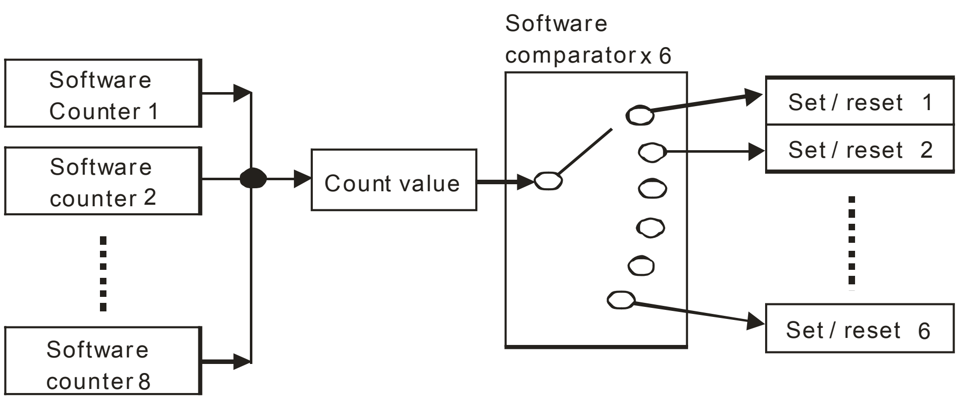
S_1: Lower bound of the comparison zone **S_2**: Upper bound of the comparison zone **S**: No. of high speed counter **D**: Comparison result (3 consecutive devices)

Explanations:

1. **S_1** should be equal to or smaller than **S_2** (**$S_1 \leqq S_2$**).
2. If **D** is specified as Y0~Y3 in this instruction, the compare result will immediately output to the external outputs Y0~Y3. However, other Y outputs will still be updated till the end of program. Also, M and S devices, not affected by the program scan cycle, will be immediately updated as well.
3. High speed counters include software high speed counters and hardware high speed counters. In addtiion, there are also two types of comparators including software comparators and hardware comparators. For detailed explanations of high speed counters please refer to section 2.9 in this manual.
4. Explanations on software comparators for DHSZ instruction
   - Corresponding table for software counters and comparators:

Counter	C232	C233	C234	C235	C236	C237	C238	C239	C240	C241	C242
Hi-speed compare Set/Reset	Share 6 software comparators										

   - Block diagram of software counters and comparators:

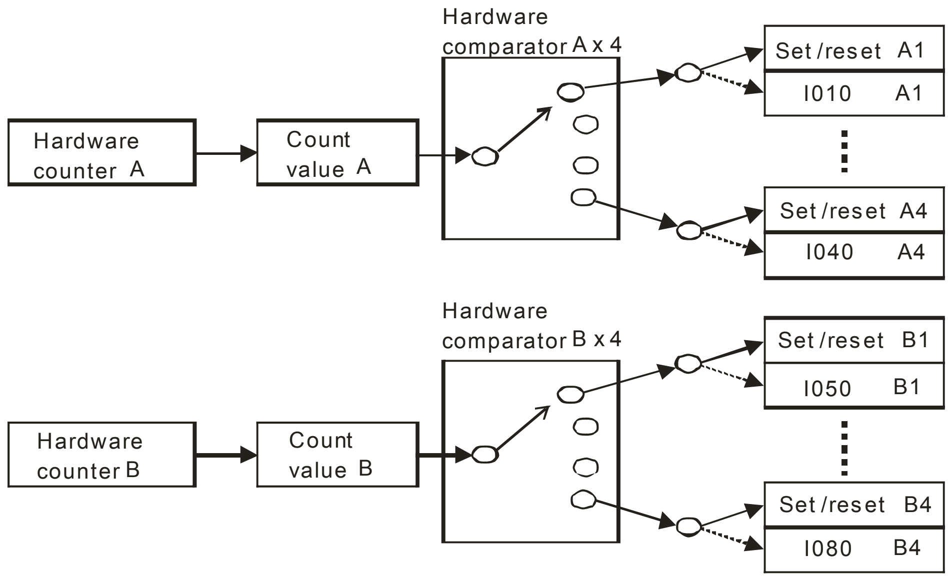


- There are 6 software zone comparators available exclusively for zone compare operation, hence the limit of 6 comparisons for zone compare does not include the comparisons of DHSCS and DHSCR.
- SS2 does not support software counter C232.

5. Explanations on hardware comparators for HSZ instruction:
   - Corresponding table for hardware counters and comparators

Hardware counter	A group				B group			
	A1	A2	A3	A4	B1	B2	B3	B4
Counter No.	C243, C245~C248, C251,C252				C244, C249, C250, C253, C254			
Hi-speed compare Set/Reset	Shares 4 hardware comparators for group A				Shares 4 hardware comparators for group B			

   - Block diagram of hardware counters and comparators:

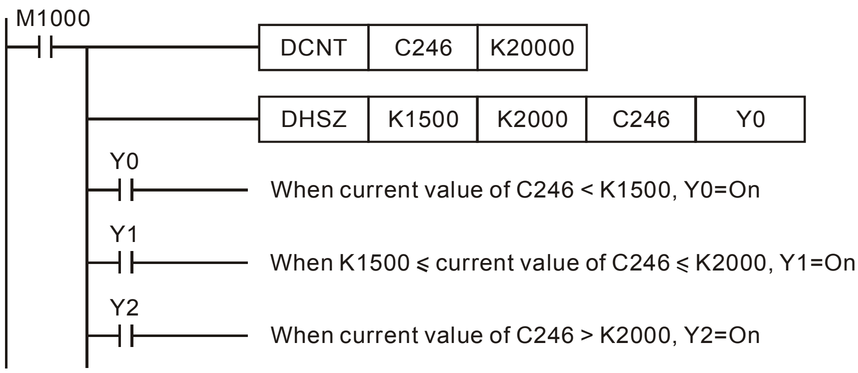


   - The two groups can only be used once for each group, occupying 2 comparators. For example, when DHSZ instruction uses A3 and A4 of group A comparators, only the other 2 comparators (A1, A2) are available for DHSCS and DHSCR instructions.
   - When DHSCS uses I030 or I040, comparators A3 and A4 are no longer available for DHSZ instruction. Also, when DHSCS uses I070 or I080, comparators B3 and B4 are no longer available for DHSZ instruction. If comparators are used repeatedly, the syntax error will be detected on the instruction behind.

Program Example 1: (Applying Hardware High Speed Counter)

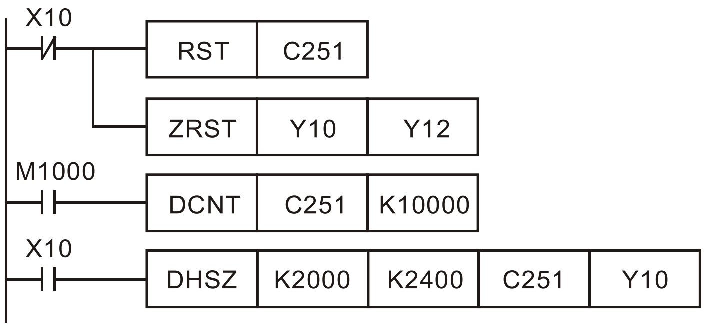
1. When **D** is specified as Y0, then Y0~Y2 will be occupied automatically.
2. When DHSZ is executed, the instruction compares the current value in C246 with the upper/lower bound (1500/2000) of the comparison zone, and Y0~Y2 will be ON according to

the comparison result.

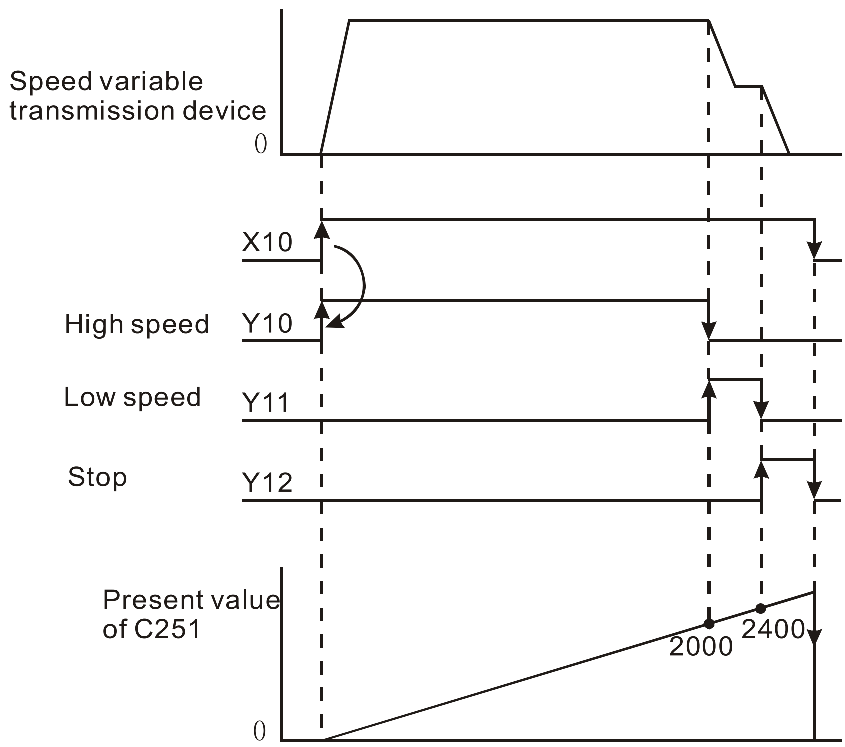


Program Example 2: (Applying DHSZ instruction for performing ramp down operation)

1. C251 is AB-phase high speed counter. When X10 = ON, DHSZ compare the present value with K2000. Present value≦K2000, Y10 = ON.
2. When X10 = OFF, Y10~Y12 are reset.



Timing diagram



API	Mnemonic	Operands	Function	Controllers
56	SPD	S1 S2 D	Speed Detection	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1	*															SPD: 7 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	
D											*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: External pulse input **S_2**: Pulse receiving time (ms) **D**: Detected result (5 consecutive devices)

Explanations:

1. The instruction counts the number of pulses received at input terminal **S_1** during the time **S_2** (ms) and stores the result in the register **D**.
2. ES2/EX2 before V0.92. External pulse input terminals designated in **S_1** :

Available input points	X0, X2	X1 (X0/X1)	X6, X7
Input mode	1-phase input (Supports single frequency)	AB-phase input (Supports quadruple frequency)	1-phase input (Supports single frequency)
Max frequency	100KHz	5KHz	10KHz

3. ES2/EX2 V1.00 or later. External pulse input terminals designated in **S_1** :

Available input points	X0, X2	X1 (X0/X1), X3 (X2/X3) X5 (X4/X5), X7 (X6/X7)	X4, X6
Input mode	1-phase input (Supports single frequency)	AB-phase input (Supports quadruple frequency)	1-phase input (Supports single frequency)
Max frequency	100KHz	5KHz	10KHz

4. SS2/SA2/SX2. External pulse input terminals designated in **S_1** :

Available input points	X0, X2	X1 (X0/X1), X3 (X2/X3) X5 (X4/X5), X7 (X6/X7)	X4, X6
Input mode	1-phase input (Supports single frequency)	AB-phase input (Supports quadruple frequency)	1-phase input (Supports single frequency)
Max frequency	SA2/SX2: 100kHz SS2: 20kHz	5KHz. X1(X0/X1) of SA2: 50kHz	10KHz

5. **D** occupies 5 consecutive registers, **D** + 1 and **D** store the results of previous pulse detection; **D** +3 and **D** + 2 store the current accumulated number of pulses; **D** + 4 store the current time remaining (max. 32,767ms).
6. If X0, X1, X2, X6 or X7 are used in a SPD instruction, their associated high-speed counters or external interrupts I000/I001, I100/I101, I200/I201, I600/I601 or I700/I701 can not be used.
7. For ES2/EX2 before V0.92: when X0, X2, X6 and X7 are used, they will be detected as 1-phase input. When X1 is used, X0(A) and X1(B) will be applied together as AB-phase input.
8. For SS2/SA2/SX2 and ES2/EX2 V1.00 or later: when X0, X2, X4 and X6 are used, they will be detected as 1-phase input. When X1, X3, x5, X7 are used, X0, X2, X4, X6 will be applied together as AB-phase input.
9. This instruction is mainly used to obtain the value of rotation speed and the results in **D** are in proportion to the rotation speed. Rotation speed **N** can be calculated by the following equation

$$N=\frac{60(D0)}{nt}\times 10^3(rpm)$$

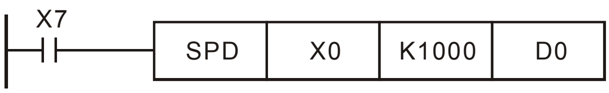
N: Rotation speed

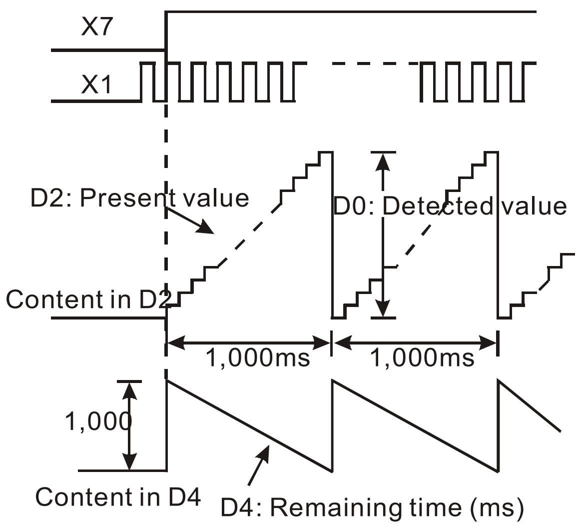
n: The number of pulses produced per rotation

t: Detecting time specified by $\mathbf{S_2}$ (ms)

Program Example:

1. When X7 = ON, D2 stores the high-speed pulses at X0 for 1,000ms and stops automatically. The results are stored in D0, D1.
2. When the 1000ms of counting is completed, D2 will be reset. When X7 turns ON again, D2 starts counting again.





API	Mnemonic		Operands	Function	Controllers
57	D	PLSY	S1 S2 D	Pulse Output	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	PLSY: 7 steps DPLSY: 13 steps
S_1					*	*	*	*	*	*	*	*	*	*	*	
S_2					*	*	*	*	*	*	*	*	*	*	*	
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Pulse output frequency **S_2**: Number of output pulses **D**: Pulse output device (Y0 ~ Y3 available)

Explanations:

1. When PLSY instruction has been executed, the specified quantity of pulses **S_2** will be output through the pulse output device **D** at the specified pulse output frequency **S_1**
2. **S_1** specifies the pulse output frequency

Output frequency range of MPU			
range	Output	Y0, Y2	Y1, Y3
	16-bit instruction	SS2: 0~10,000Hz ES2/EX2/SA2/SX2: 0~32,767 Hz	0~10,000Hz
	32-bit instruction	SS2: 0~10,000Hz ES2/EX2/SA2/SX2: 0~100,000 Hz	0~10,000Hz
If frequency equals or smaller than 0Hz is specified, pulse output will be disabled. If frequency bigger than max frequency is specified, PLC will output with max frequency.			

3. **S_2** specifies the number of output pulses.
   16-bit instruction: -32,768~32,767. 32-bit instruction: -2,147,483,648~2,147,483,647.
   When **S_2** is specified as K0, the pulse will be output continuously regardless of the limit of pulse number.
4. When D1220/D1221 = K1 or K2, the positive/negative sign of **S_2** denotes pulse output direction (Positive/negative).

5. Four pulse output modes:

Mode / Output	D1220						D1221					
	K0		K1	K2	K3		K0		K1	K2	K3#	
Y0	Pulse		Pulse	A	CW							
Y1		Pulse	Dir	B		Pulse						
Y2							Pulse		Pulse	A	CCW	
Y3								Pulse	Dir	B		Pulse

Pulse: Pulse A: A phase pulse CW: clockwise

Dir: Direction B: B phase pulse CCW: Counter-clockwise

Note # : When D1220 is specified as K3, D1221 is invalid.

6. Pulse output flags:

Output device	Y0	Y1	Y2	Y3
Completed Flag	M1029	M1030	M1102	M1103
Immediately pause	M1078	M1079	M1104	M1105
0.01~100Hz output	M1190	M1191	M1192	M1193

a) M1029 = ON after Y0/Y1 (D1220=K1, pulse/Dir) output is completed.
M1102 = ON after Y2/Y3 (D1221=K1, pulse/Dir) output is completed.
M1029 = ON after the Y0/Y2 (D1220 = K3, CW/CCW) output is completed.

b) The execution completed flag M1029, M1030, M1102, and M1103 should be manually reset by users after pulse output is completed.

c) When PLSY / DPLSY instruction is OFF, the pulse output completed flags will all be reset.

d) When M1190~M1192 = ON, the available output range for PLSY Y0~Y3 is 0.01~100Hz.

7. While the PLSY instruction is being executed, the output will not be affected if $\mathbf{S_2}$ is changed. To change the pulse output number, stop the PLSY instruction, then change the pulse number.

8. $\mathbf{S_1}$ can be changed during program execution and the change will take effects until the modified PLSY instruction is being executed.

9. The ratio of OFF time and ON time of the pulse output is 1:1.

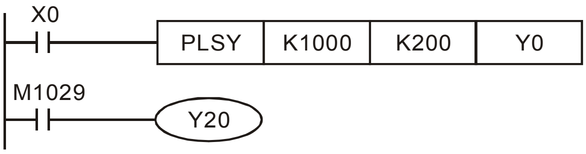
10. If operand $\mathbf{S_1}$, $\mathbf{S_2}$ use index F, only 16-bit instruction is available.

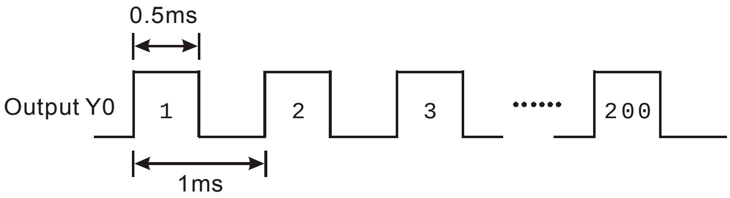
11. There is no limitation on the times of using this instruction, however the program allows only **4** instructions (PLSY, PWM, PLSR) to be executed at the same time. If Y1 is used for several high speed pulse output instructions, PLC will output according to the execution order of these instructions.

Program Example:

1. When X0 = ON, 200 pulses of 1kHz are generated from output Y0, after the pulse output has been completed, M1029 = ON to set Y20.

2. When X0 = OFF, pulse output Y0 will immediately stop. When X0 turns ON again, the pulse output will start from the first pulse.





Points to note:

1. Description of associated flags:

M1029: M1029 = ON when Y0 pulse output is completed.

M1030: M1030 = ON when Y1 pulse output is completed.

M1102: M1102 = ON when Y2 pulse output is completed.

M1103: M1103 = ON when Y3 pulse output is completed.

M1078: Y0 pulse output pause (immediately)

M1079: Y1 pulse output pause (immediately)

M1104: Y2 pulse output pause (immediately)

M1105: Y3 pulse output pause (immediately)

M1190 Se t Y0 high speed output as 0.01~100Hz

M1191 Se t Y1 high speed output as 0.01~100Hz

M1192 Se t Y2 high speed output as 0.01~100Hz

M1193 Se t Y3 high speed output as 0.01~100Hz

M1347: Auto reset Y0 when high speed pulse output completed

M1348: Auto reset Y1 when high speed pulse output completed

M1524: Auto reset Y2 when high speed pulse output completed

M1525: Auto reset Y3 when high speed pulse output completed

M1538: Indicating pause status of Y0

M1539: Indicating pause status of Y1

M1540: Indicating pause status of Y2

M1541: Indicating pause status of Y3

2. Description of associated special D registers:

D1030: Present number of Y0 output pulses (Low word).

D1031: Present number of Y0 output pulses (High word).

D1032: Present number of Y1 output pulses (Low word).

D1033: Present number of Y1 output pulses (High word).

D1336: Present number of Y2 output pulses (Low word).

D1337: Present number of Y2 output pulses (High word).

D1338: Present number of Y3 output pulses (Low word).

D1339: Present number of Y3 output pulses (High word).

D1220: Phase of the 1st group pulse output (Y0,Y1), please refer to explanations of the instruction.

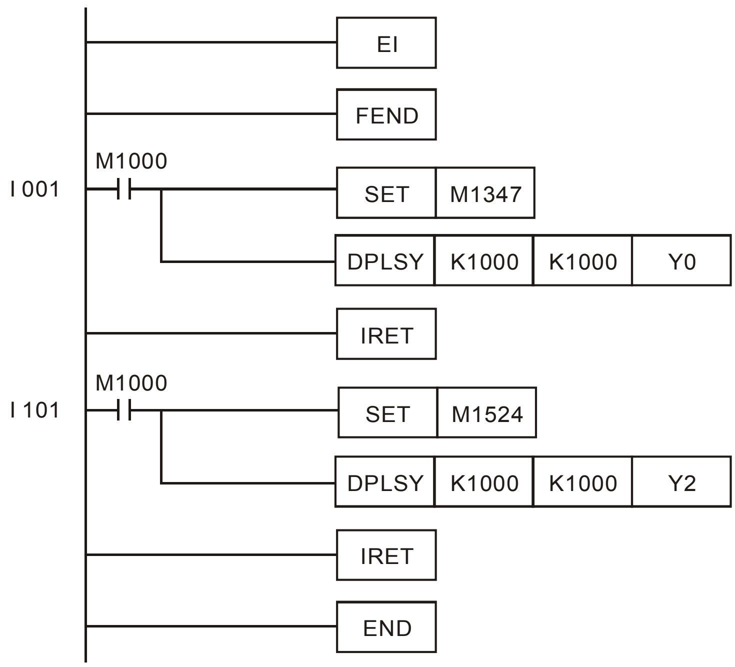
D1221: Phase of the 2nd group pulse output (Y2,Y3), please refer to explanations of the instruction.

3. More explanations for M1347,M1348, M1524, M1525:

Generally when pulse output is completed, PLSY instruction has to be reset so that the instruction can start pulse output one more time. When M1347, M1348, M1524 or M1525 is enabled, the associated output terminals (Y0~Y3) will be reset automatically when pulse output is completed, i.e. the PLSY instruction is reset. When PLC scans to PLSY instruction again, the pulse output starts automatically. In addition, PLC scans the 4 flags after END instruction, hence PLSY instruction in continuous pulse output mode requires a delay time of one scan cycle for next pulse output operation.

The function is mainly used in subroutines or interrupts which require high speed pulse output. Here are some examples:

Program Example 1:

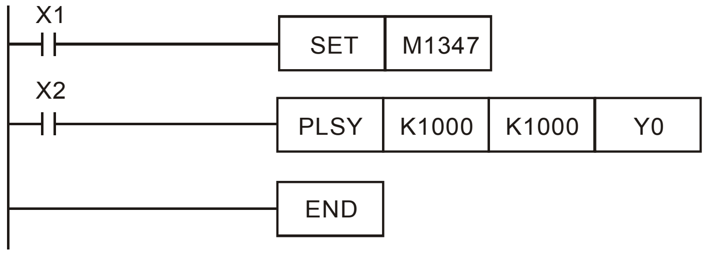


Explanations:

a) Whenever I001 is triggered, Y0 will output 1,000 pulses; whenever I101 is triggered, Y2 will output 1,000 pulses.

b) When pulse output is completed, there should be an interval of at least one scan cycle before next pulse output operation is triggered. .

Program Example 2:



Explanation:

When both X1 and X2 are ON, Y0 pulse output will operate continuously. However, there will be a delay of approx. 1 scan cycle every 1000 pulses.

API	Mnemonic	Operands	Function	Controllers
58	PWM	(S1) (S2) (D)	Pulse Width Modulation	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				**Word devices**											PWM: 7 steps
S_1					*	*	*	*	*	*	*	*	*	*	*	
S_2					*	*	*	*	*	*	*	*	*	*	*	
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Pulse output width (ms) **S_2**: Pulse output cycle (ms) **D**: Pulse output device (Y0, Y1, Y2,Y3)

Explanations:

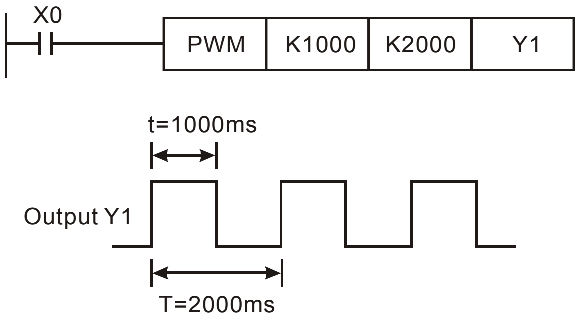
1. **S_1** is specified as pulse output width (t). **S_2** is specified as pulse output cycle (T).
   Rule: **S1** ≦ **S2**.

Reference Table for Output Cycle and Output Width					
Range of pulse output width / cycle	Output	Y0	Y2	Y1	Y3
	Pulse width	0~1000		0~32767	
	t / T	0~100.0ms, 0~10.00ms		0~32,767ms, 0~3,276.7ms	
Flag for switching unit		M1112	M1113	M1070	M1071

2. Pulse output devices for operand D: Y0, Y1, Y2, Y3,
3. When several pulse output instructions (PLSY, PWM, PLSR) use Y1 or Y3 as the output device in the same scan cycle, PLC will perform the instruction which is executed first.
4. When **S_1**≦0, **S_2**≦0 or **S_1**＞**S_2**, errors will occur (M1067 and M1068 will not be ON) and no output will be generated from pulse output devices. When **S_1** = **S_2**, the pulse output device will be ON continuously.
5. **S_1**, **S_2** can be changed when PWM instruction is being executed.
6. When M1112 = ON, the unit of Y0 output pulse is 10μs, when M1112 = OFF, the unit is 100μs.
7. When M1070 = ON, the unit of Y1 output pulse is 100μs, when M1070 = OFF, the unit is 1ms.
8. When M1113 = ON, the unit of Y2 output pulse is 10μs, when M1113 = OFF, the unit is 100μs.
9. When M1071 = ON, the unit of Y3 output pulse is 100μs, when M1071 = OFF, the unit is 1ms.
10. There is no limitation on the times of using this instruction in the program, but only 4 instructions can be executed at the same time.

Program Example:

When X0 = ON, Y1 output the pulse as shown opposite. When X0 = OFF, output Y1 turns OFF.



Note:

1. Flag description:

   M1070: Switching clock pulse of Y1 for PWM instruction (ON:100 us, OFF: 1ms)

   M1071: Switching clock pulse of Y3 for PWM instruction (ON:100 us, OFF: 1ms)

   M1112 Switching clock pulse of Y0 for PWM instruction (ON:10 us, OFF: 100 us)

   M1113 Switching clock pulse of Y2 for PWM instruction (ON:10 us, OFF: 100 us)

2. Special D registers description:

   D1030 PV of Y0 pulse output (Low word)

   D1031 PV of Y0 pulse output (High word)

   D1032: Low word of the present value of Y1 pulse output

   D1033 High word of the present value of Y1 pulse output

   D1336 PV of Y2 pulse output (Low word)

   D1337 PV of Y2 pulse output (High word)

   D1338: Low word of the present value of Y3 pulse output.

   D1339: High word of the present value of Y3 pulse output.

API	Mnemonic		Operands	Function	Controllers
59	D	PLSR	(S1) (S2) (S3) (D)	Pulse Ramp	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	PLSR: 9 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DPLSR: 17 steps
S_3					*	*	*	*	*	*	*	*	*	*	*	
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Maximum frequency (Hz) **S_2**: Number of pulses **S_3**: Ramp up/down time (ms)

D: Pulse output device (Y0, Y1, Y2 and Y3 are available)

Explanations:

1. PLSR instruction performs a frequency ramp up/down process when positioning. Speed ramp up process is activated between static status to the target speed. Pulse output persists in target speed before getting close to target position. When target position is near, speed ramp down process executes, and pulse output stops when target position is achieved.
2. Set range of **S_1** pulse output frequency:

Range of **S_1** pulse output frequency:			
Output frequency:	Output	Y0, Y2	Y1, Y3
	16-bit	SS2: 6~10,000Hz ES2/EX2/SA2/SX2: 6~32,767Hz	6~10,000Hz
	32-bit	SS2: 6~10,000Hz ES2/EX2/SA2/SX2: 0~100,000Hz	6~10,000Hz
If frequency smaller than 6Hz is specified, PLC will output 6Hz. If frequency bigger than max frequency is specified, PLC will output with max frequency.			

3. When output device is specified with Y0, Y2, the start/end frequency of Y0 is set by D1340 and start/end frequency of Y2 is set by D1352.
4. When output device is specified with Y1, Y3, the start/end frequency is 0Hz.
5. When D1220/D1221 = K1 or K2, positive/negative sign of **S_2** denotes pulse output direction.
6. PLSR instruction supports two modes of pulse output as below list.

Mode / Output	D1220			D1221		
	K0		K1	K0		K1
Y0	Pulse		Pulse			
Y1		Pulse	Dir			
Y2				Pulse		Pulse
Y3					Pulse	Dir

7. When assigning Y0 and Y2 output mode as Pulse, i.e. D1220 = K0, D1221 = K0, the available range for $\mathbf{S_2}$ is 1~32,767 (16-bit instruction) and 1~2,147,483,647 (32-bit instruction).
8. When assigning Y0 and Y2 output mode as Pulse/Dir, i.e. D1220 = K1, D1221 = K1, the available range for $\mathbf{S_2}$ is 1~32,767 or -1~-32,768 (16-bit instruction) and 1~2,147,483,647 or -1~-2,147,483,648 (32-bit instruction)
9. When assigning output device as Y1 and Y3, the available range for $\mathbf{S_2}$ is 1~32,767 (16-bit instruction) and 1~2,147,483,647 (32-bit instruction).
10. $\mathbf{S_3}$: Ramp up/down time (unit: ms, min. 20ms).
    When assigning output device as Y1 and Y3, the set value of ramp up and ramp down time should be the same.
    When assigning output device as Y0 and Y2, and if:
    - M1348 = OFF(Y0) and M1535 = OFF(Y2), the ramp up and ramp down time should be the same.
    - M1348 = ON and M1535 = ON, then $\mathbf{S_3}$ specifies ramp up time only. The ramp down time is specified by value set in D1348 (Y0) and D1349 (Y2).
11. Pulse output devices for operand D: Y0, Y1, Y2, Y3
12. When M1257 = OFF, ramp up/down curve of Y0 and Y2 is straight line. When M1257 = ON, ramp up/down curve will be S curve. The ramp up/down curve of Y1 and Y3 is fixed as straight line
13. The output will not be affected if $\mathbf{S_1}$, $\mathbf{S_2}$ or $\mathbf{S_3}$ are changed when PLSR instruction is being executed. PLSR instruction has to be stopped if changing values in $\mathbf{S_1}$, $\mathbf{S_2}$ or $\mathbf{S_3}$ is required.
14. Flags for indicating pulse output status:

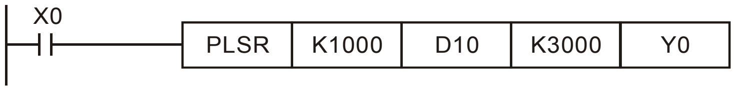
Output	Y0	Y1	Y2	Y3
Completion	M1029	M1030	M1102	M1103
Immediately Pause	M1078	M1079	M1104	M1105

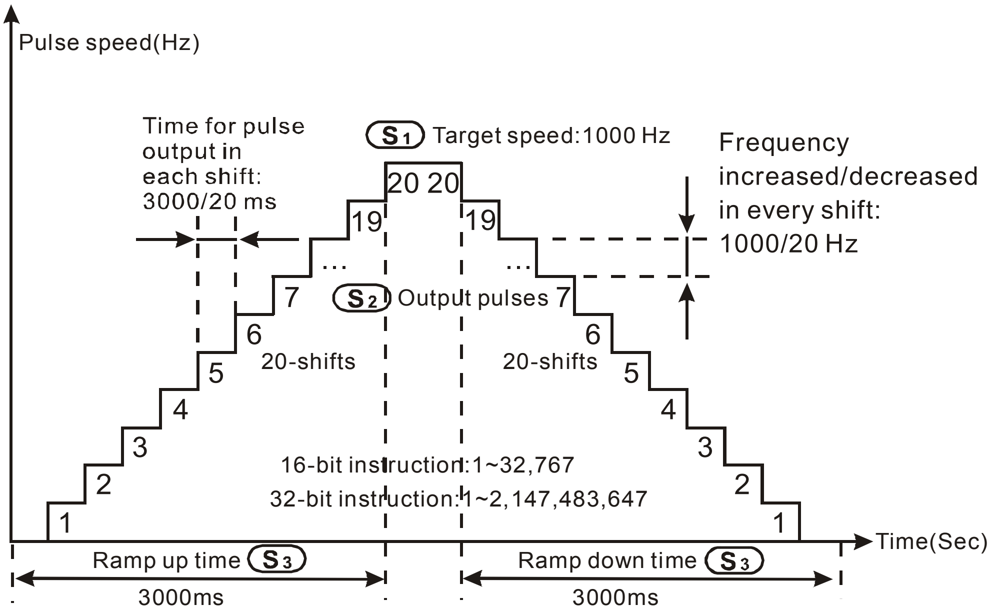
    a) When pulse output on Y0/Y1 specified as Pulse/Dir (D1220 = K1) is completed, completion flag M1029 = ON.
    b) When pulse output on Y2/Y3 specified as Pulse/Dir (D1221 = K1) is completed, completion flag M1102 = On。
    c) When PLSR/DPLSR instruction is activated again, the completion flags will automatically be reset.
15. During the ramp up process, the pulse numbers (frequency x time) of each speed shift may not all be integer values, but PLC will operate integer value only. In this case, the omitted decimals will result in errors between each speed shift, i.e. pulse number for each shift may differ due to this operation. For ensuring the required output pulse number, PLC will fill in pulses as need automatically in order to correct the deviation.

16. There is no limitation on the times of using this instruction in the program. However, only 4 instructions can be executed at the same scan time. When several pulse output instructions (PLSY, PWM, PLSR) use Y1 as the output device in the same scan cycle, PLC will execute pulse output according to the driven order of these instructions.
17. Set value falls out of the available range of operands will be automatically corrected with the min. or max available value.

Program Example:

1. When X0 = ON, PLSR performs pulse output on Y0 with a target speed of 1000Hz, output pulse number D10 and ramp up/down time of 3000ms. Ramp up process begins to increase 1000/20 Hz in every shift and every shift outputs D10/40 pulses for 3000/20 ms.
2. When X0 = OFF, the output stops immediately and starts from the count value in D1030, D1031 when PLSR is executed again.
3. Ramp up/down shifts for Y0, Y2: 20. Ramp up/down shifts for Y1, Y3: 10





Explanations on associated flags and registers:

1. Description on associated flags:

   For M1029, M1030, M1102, M1103, M1078, M1079, M1104, M1105, M1538, M1539, M1540, M1541, M1347, M1348, M1524, M1525, please refer to PLSY instruction.

   M1108: Y0 pulse output pause (ramp down). ON = pause, OFF = resume

   M1109: Y1 pulse output pause (ramp down). ON = pause, OFF = resume

   M1110: Y2 pulse output pause (ramp down). ON = pause, OFF = resume

   M1111: Y3 pulse output pause (ramp down). ON = pause, OFF = resume

M1156: Enabling the mask and alignment mark function on I400/I401(X4) corresponding to Y0.

M1257: Set the ramp up/down of Y0, Y2 to be “S curve.” ON = S curve.

M1158: Enabling the mask and alignment mark function on I600/I601(X6) corresponding to Y2.

M1534: Enable ramp-down time setting on Y0. Has to be used with D1348

M1535: Enable ramp-down time setting on Y2. Has to be used with D1349

2. Description on associated special registers:

For D1030~D1033, D1336~D1339, D1220, D1221, please refer to PLSY instruction

D1026: M1156 = ON, D1026 stores pulse number for masking Y0 (Low word).

D1027: M1156 = ON, D1026 stores pulse number for masking Y0 (High word).

D1135: M1158 = ON, D1135 stores pulse number for masking Y2 (Low word).

D1136: M1158 = ON, D1135 stores pulse number for masking Y2 (High word).

D1232: Output pulse number for ramp-down stop when Y0 mark sensor receives signals. (Low word).

D1233: Output pulse number for ramp-down stop when Y0 mark sensor receives signals. (High word).

D1234: Output pulse number for ramp-down stop when Y2 mark sensor receives signals (Low word).

D1235: Output pulse number for ramp-down stop when Y2 mark sensor receives signals (High word).

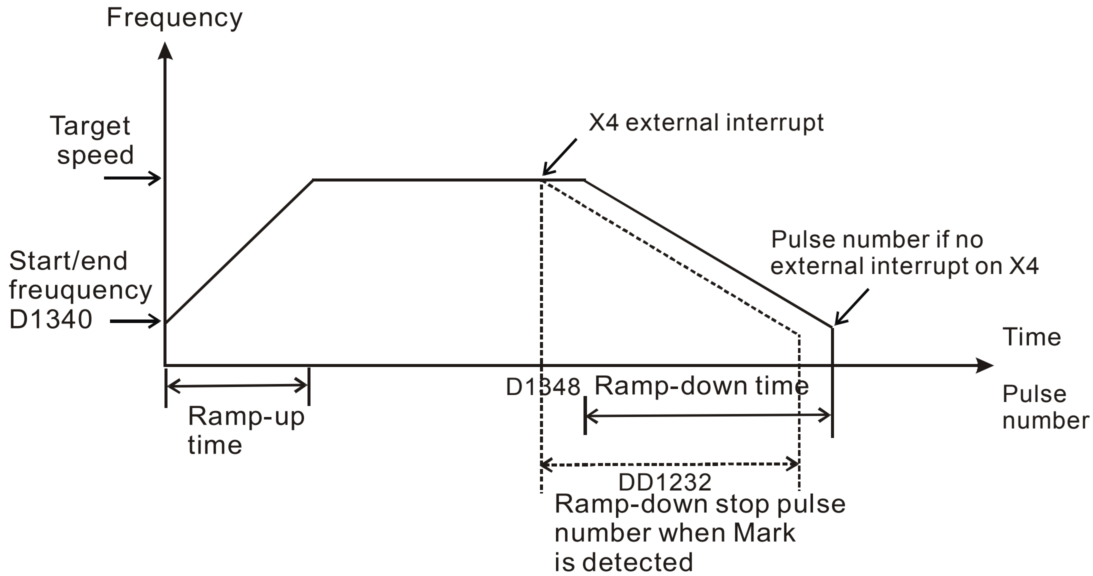
D1348: M1534 = ON, D1348 stores the ramp-down time of CH0(Y0, Y1) pulse output.

D1349: M1535 = ON, D1349 stores the ramp-down time of CH1(Y2, Y3) pulse output.

D1340 Start/end frequency of the pulse output CH0 (Y0, Y1)

D1352 Start/end frequency of the pulse output CH1 (Y2, Y3)

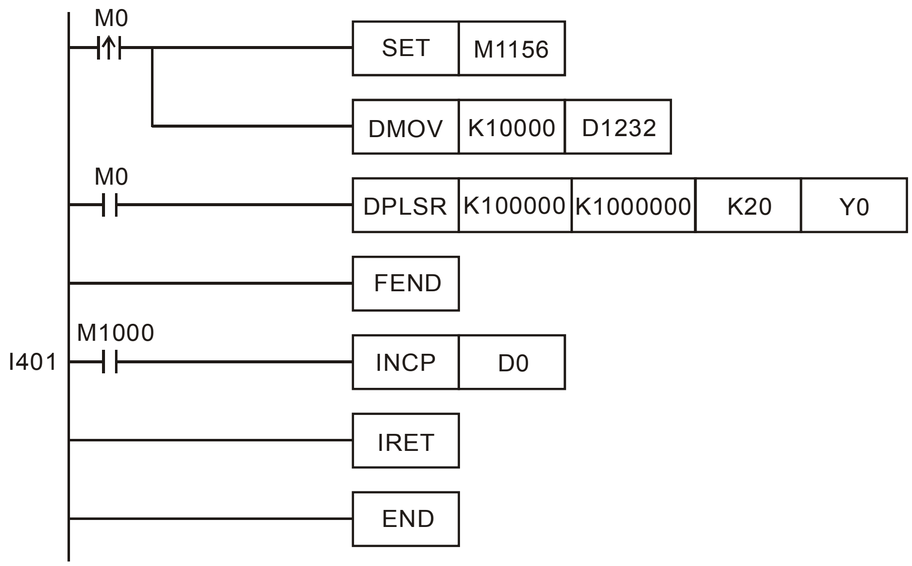
3. Operation of Mark function on Y0:



- When M1156/M1158 = ON, enable ramp-down pause (Mark function) on Y0/Y2 when X4/X6 receives interrupt signals.
- When Mark function is enabled, ramp down time is independent of the ramp up time. Users can set ramp up time in $\mathbf{S}_3$ and ramp down time in D1348/D1349. (Range: 20ms~32767ms)
- When Mark function is executed and the ramp-down stop pulses (DD1232/DD1234) are specified, PLC will execute ramp-down stop with specified pulses after Mark is detected. However, if DD1232/DD1234 are less than the specified ramp-down time (D1348 / D1349), PLC will fill DD1232/DD1234 with the value of ramp-down time. In addition, if DD1232/DD1234 is more than the half of total output pulses, PLC will modify DD1232/DD1234 to be less than half of the total output pulses.
- Ramp-down stop pulses (DD1232/DD1234) are 32-bit value. Set value K0 will disable the Mark function.
- Y0,Y2 relative parameters for Mask and Alignment Mark function:

Parameter / Output	Mark flag	Input points	Ramp down time	Pulse number for masking output	Pulse number for ramp-down of Mark function	Output pause (ramp down)	Pause status
Y0	M1156	X4	D1348	D1026, D1027	D1232, D1233	M1108	M1538
Y2	M1158	X6	D1349	D1135, D1136	D1234, D1235	M1110	M1540

Program example 1:

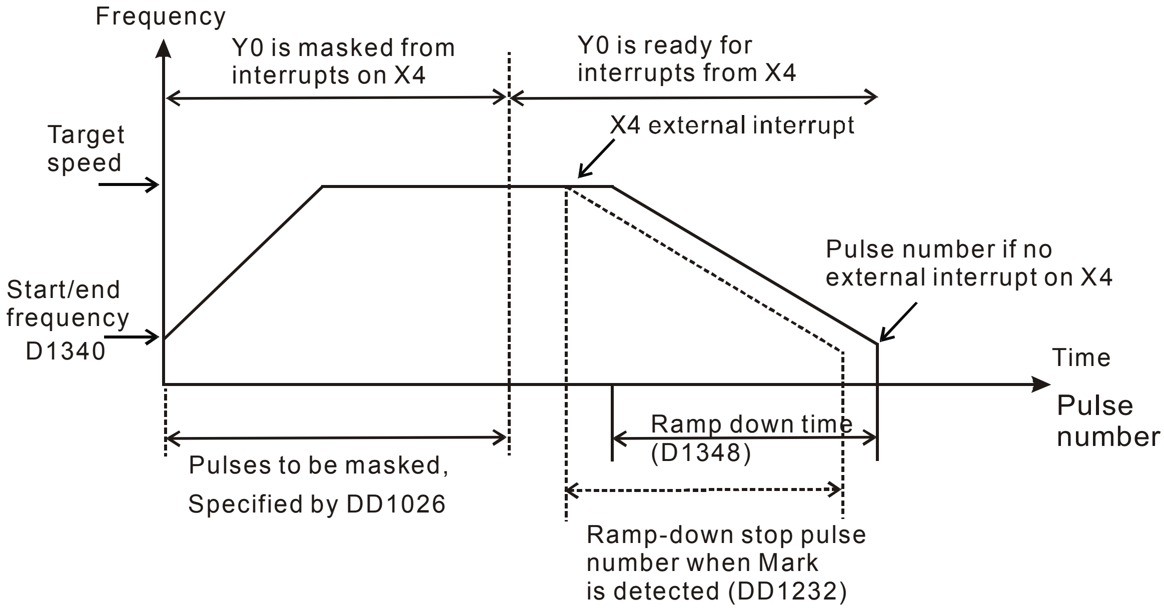


Explanations:

- When M0 is triggered, Y0 executes pulse output. If external interrupt is detected on X4, pulse output will perform ramp down process for 10,000 pulses and then stop. M1108 will be ON to indicate the pause status (ramp down). If no interrupt is detected, Y0 pulse output will stop after 1,000,000 pulses are completed.

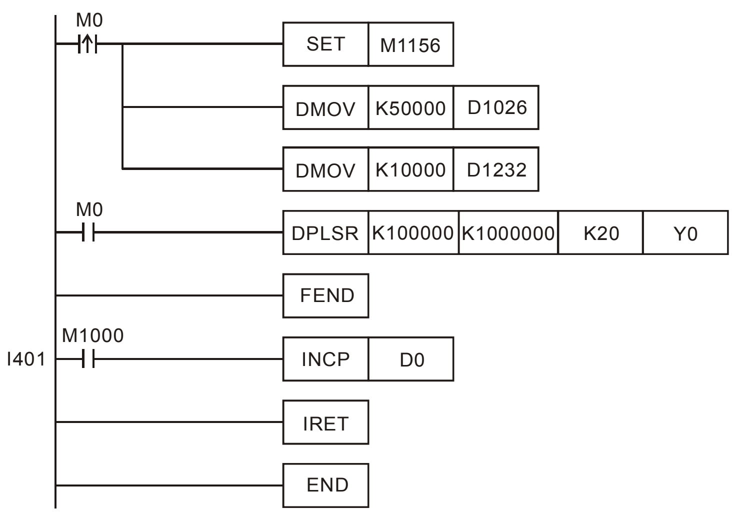
- When pulse output ramps down and stops after Mark is detected, M1538 will be ON to indicate the pause status. If users need to complete the remaining pulses, set OFF the flag M1108 and pulse output will resume.

4. Operation of Mask function on Y0:



- Mask function on Y0 will be enabled when D1026 and D1027 are specified with values other than 0. Mask function is disabled when D1026 and D1027 are specified with 0. If pulse output process can not reach the target speed, PLC will clear DD1026 to disable the Mask function. If the Mask range is set to be within the ramp-up section, PLC will automatically modify DD1026 to be longer than the ramp-up section. On the other hand, if DD1026 is set between ramp-down section, PLC will modify DD1026 to be the range before the beginning of ramp-down process. Mask function setting method on Y2 is the same as Y0.

Program example 2:



Explanations:

- When M0 is triggered, Y0 executes pulse output. When external interrupt is detected on X4 after 50,000 pulses, pulse output will perform ramp down process for 10,000 pulses and then stop. M1108 will be ON. If no interrupt is detected on X4, Y0 pulse output will stop after 1,000,000 pulses are completed.
- Interrupt triggered between 0 ~ 50,000 pulses will be invalid, i.e. no ramp-down process will be performed before 50,000 pulses are achieved.

Points to note:

1. When Mark function is executed with Mask function, PLC will check the validity of Mask range first, then ramp-down stop pulses of Mark function. If the above set values exceed the proper range, PLC will automatically modify the set values after the instruction is executed.
2. When PLSR or positioning instructions with ramp-up/down section are enabled, the user can check the pulses of ramp-up section in DD1127 and pulses of ramp-down section in DD1133.
3. Users can perform single speed positioning when ramp-up/down time setting is not specified.

API	Mnemonic	Operands	Function	Controllers
60	IST	S D1 D2	Initial State	ES2/EX2 SS2 SA2 SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

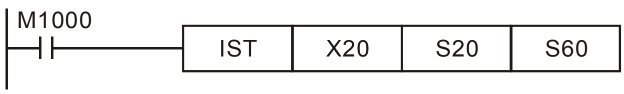
Operands:

S: Source device for assigning pre-defined operation modes (8 consecutive devices). **D_1** The smallest No. of step points in auto mode. **D_2**: The greatest No. of step points in auto mode.

Explanations:

1. The IST is a handy instruction specifically for the initial state of the step ladder operation modes.
2. The range of **D_1** and **D_2** : S20~S911, **D_1** < **D_2**.
3. IST instruction can only be used one time in a program.

Program Example 1:



S: X20: Individual operation (Manual operation)
X21: Zero return
X22: Step operation
X23: One cycle operation
X24: Continuous operation
X25: Zero return start switch
X26: Start switch
X27: Stop switch

1. When IST instruction is executed, the following special auxiliary relays will be assigned automatically.
   M1040: Movement inhibited
   M1041: Movement start
   M1042: Status pulse
   M1047: STL monitor enable
   S0: Manual operation/initial state step point
   S1: Zero point return/initial state step point
   S2: Auto operation/initial state step point

2. When IST instruction is used, S10~S19 are occupied for zero point return operation and cannot be used as a general step point. In addition, when S0~S9 are in use, S0 initiates "manual operation mode", S1 initiates "zero return mode" and S2 initiates "auto mode". Thus, the three step points of initial state have to be programmed in first priority.

3. When S1 (zero return mode) is initialized, i.e. selected, zero return will NOT be executed if any of the state S10~S19 is ON.
4. When S2 (auto mode) is initialized, i.e. selected, auto mode will NOT be executed if M1043 = ON or any of the state between $\mathbf{D_1}$ to $\mathbf{D_2}$ is ON.

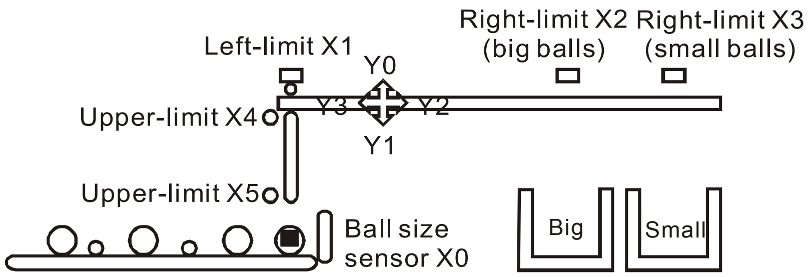
Program Example 2:

Robot arm control (by IST instruction):

1. Control purpose:

   Select the big balls and small balls and move them to corresponding boxes. Configure the control panel for each operation.
2. Motion of the Robot arm:

   lower robot arm, clip balls, raise robot arm, shift to right, lower robot arm, release balls, raise robot arm, shift to left to finish the operation cycle.
3. I/O Devices



4. Operation mode:

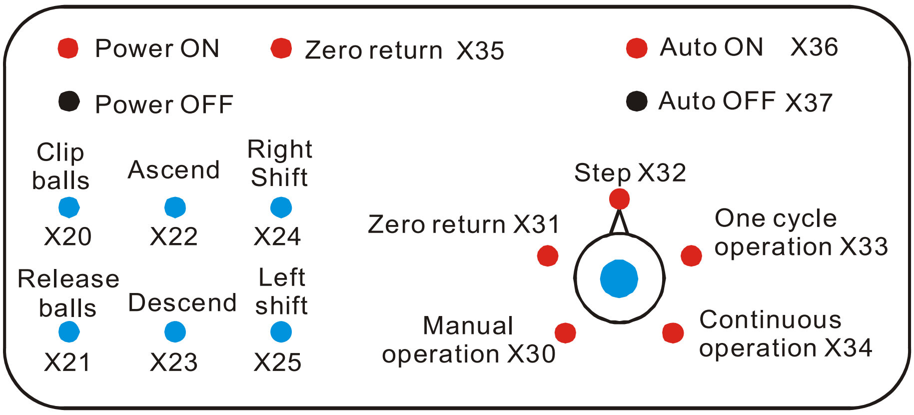
   Single step: Press single button for single step to control the ON/OFF of external load.

   Zero return: Press zero return button to perform homing on the machine.

   Auto (Single step / One cycle operation / Continuous operation):

   - Single step: the operation proceeds with one step every time when Auto ON is pressed.
   - One cycle operation: press Auto ON at zero position, the operation performs one full cycle operation and stops at zero point. If Auto OFF is pressed during the cycle, the operation will pause. If Auto ON is pressed again, the operation will resume the cycle and stop at zero point.
   - Continuous operation: press Auto ON at zero position, the operation will perform continuous operation cycles. If Auto OFF is pressed, the operation will stop at the end of the current cycle.

5. Control panel

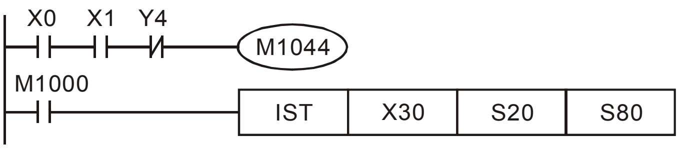


a) X0: ball size sensor.

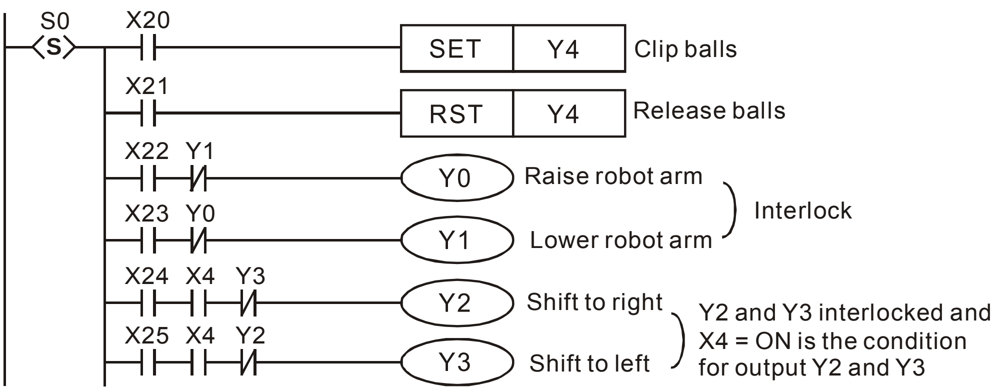
b) X1: left-limit of robot arm, X2: right-limit (big balls), X3: right-limit (small balls), X4: upper-limit of clamp, X5: lower-limit of clamp.

c) Y0: raise robot arm, Y1: lower robot arm, Y2: shift to right, Y3: shift to left, Y4: clip balls.

6. START circuit:

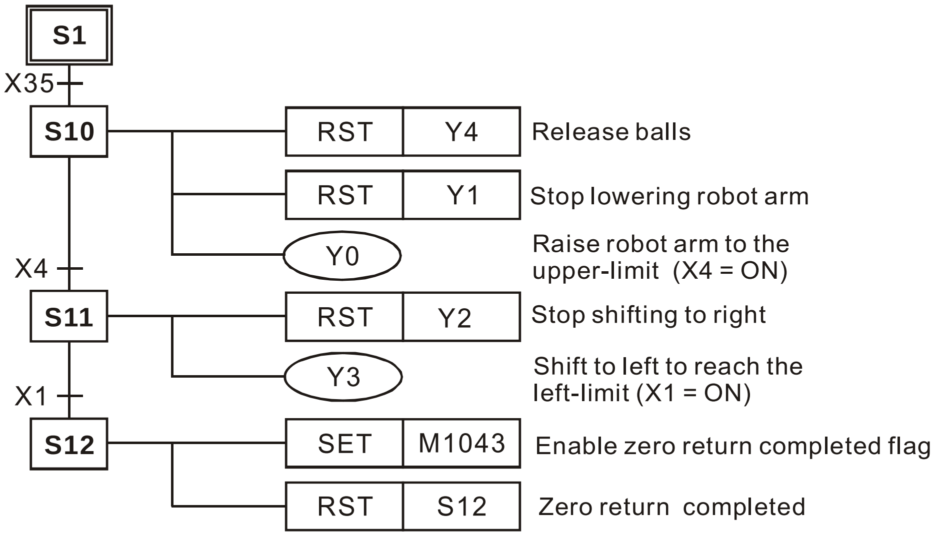


7. Manual mode:

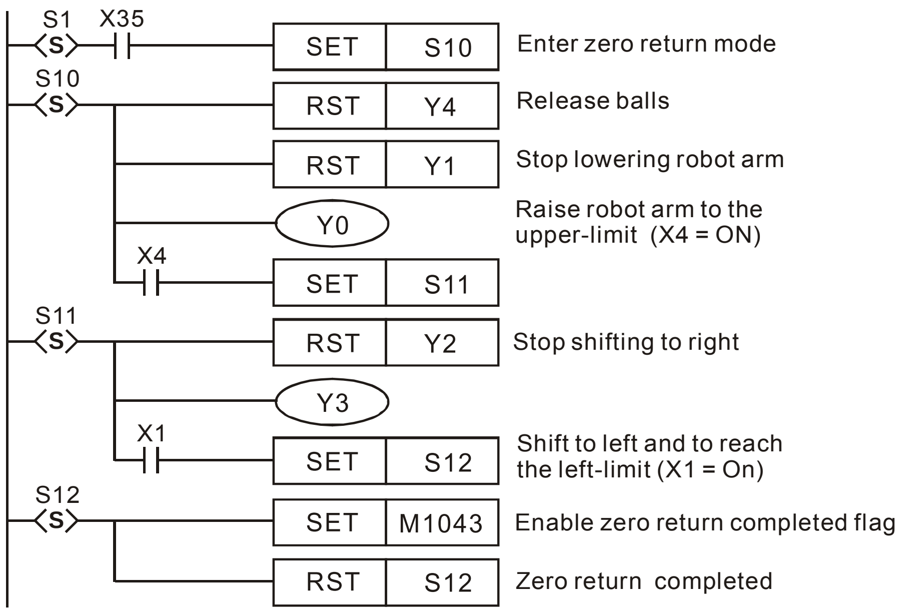


8. Zero return mode:

a) SFC:

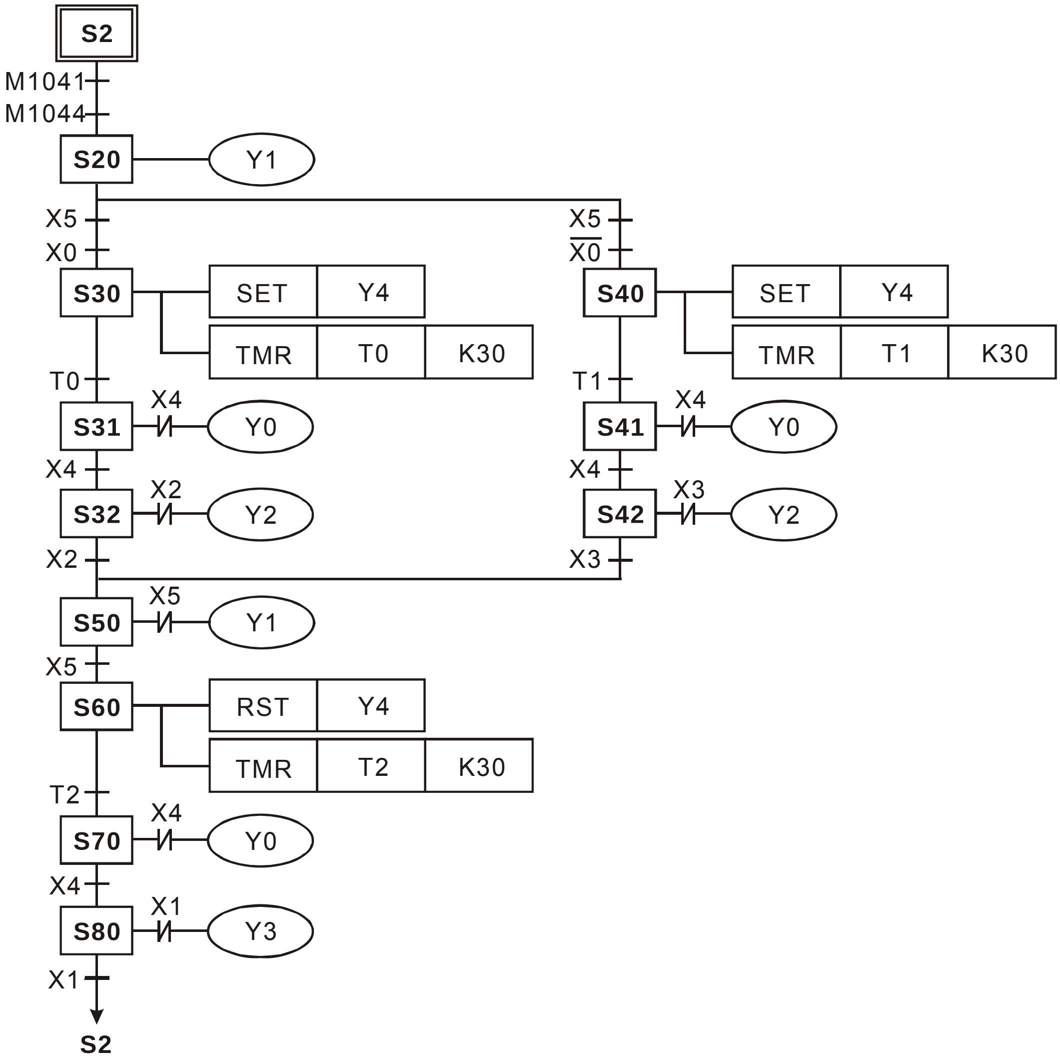


b) Ladder Diagram:

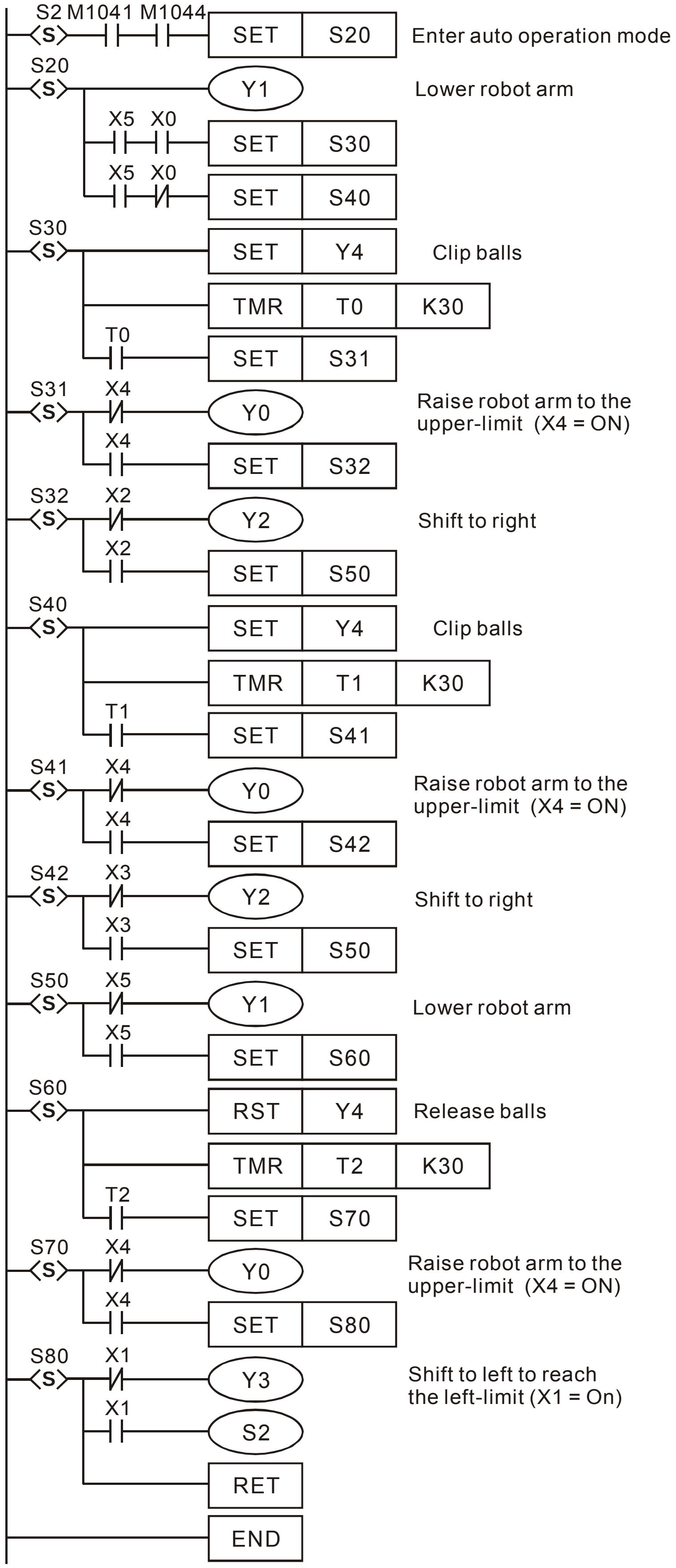


9. Auto operation (Single step / One-cycle operation / continuous operation):

a) SFC:



b) Ladder Diagram:



Flag explanation:

M1040:

Disable step transition. When M1040 = ON, all motion of step points are disabled.

1. **Manual operation mode:** M1040 remains ON in manual mode.
2. **Zero return mode/one cycle operation mode:** M1040 remains ON in the interval after Auto Stop and before Auto Start is pressed.
3. **Step operation mode:** M1040 remians ON until Auto Start is pressed.
4. **Continuous operation mode:** When PLC goes from STOP→RUN, M1040 = ON. When Auto Start is pressed, M1040 turns OFF.

M1041:

Step transition starts. This special M indicates the transition from step point S2 to the next step point.

1. **Manual operation mode/Zero return mode:** M1041 remians OFF.
2. **Step operation mode/One cycle operation mode:** M1041 = ON when Auto Start is pressed.
3. **Continuous operation mode:** M1041 stays ON when Auto Start is pressed and turns OFF when Auto Stop is pressed.

M1042:

Enable pulse operation: When Auto Start is pressed, PLC sents out pulse once for operation. .

M1043:

Zero return completed: M1043 = ON indicates that zero return is completed.

M1044:

Zero point condition: In continuous operation mode, M1044 has to be ON as a condition for enabling step transition from S2 to the next step point.

M1045:

Disable "all output reset" function.

- If the machine (not at the zero point) goes,
  - from manual (S0) to zero return (S1)
  - from auto (S2) to manual (S0)
  - from auto (S2) to zero return (S1)

And

M1045 = OFF, any of the S among $\mathbf{D_1}$ ~ $\mathbf{D_2}$ in action will be reset as well as the output Y.

M1045 = ON, output Y will be retained but the step in action will be reset.

- If the machine (at the zero point) goes from zero return (S1) to manual (S0), no matter M1045 is ON or OFF, Y output will be retained but the step in action will be reset.

M1046:

Indicates STL(Step Ladder) status. When STL operation is activate, M1046 = ON if any of the step point S is ON. If M1047 = ON, M1046 also activates to indicate ON status of step points. In addition, D1040 ~ D1047 records 8 step numbers from the current ON step to the previous 7 ON steps.

M1047:

Enable STL monitoring. When IST instruction executes, M1047 will be forced ON, i.e. M1047 remains ON in every scan cycle as long as IST instruction is executing. This flag is used to monitor all step points (S).

D1040~D1047:

Records 8 step numbers from the current ON step to the previous 7 ON steps.

API	Mnemonic			Operands	Function	Controllers
61	D	SER	P	S₁ S₂ D n	Search a Data Stack	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1							*	*	*	*	*	*	*			SER, SERP: 9 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DSER, DSERP: 17 steps
D								*	*	*	*	*	*			
N					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Start device of data stack **S_2**: Device to be searched **D**: Start device for storing search result (occupies 5 consecutive devices) **n**: Stack length

Explanations:

1. SER instruction searches for the value stored in **S_2** from the data stack starting with **S_1**, with a stack length **n**. The search results are stored in the 5 registers starting from **D**
2. **D** stores the total of the matched results; **D+1** stores the No. of device storing the first matched result; **D+2** stores the No. of device storing the last matched result; **D+3** stores the No. of device storing the smallest value; **D+4** stores the No. of device storing the biggest value..
3. If operand **S_2** uses index F, only 16-bit instruction is available
4. If the instruction applied 32-bit instruction, operands **S_1**, **S_2**, **D**, **n** will specify 32-bit registers.
5. The range of operand **n**: **n** = 1~256 (16-bit instruction), **n** = 1~128 (32-bit instruction)

Program Example:

1. When X0 = ON, the data stack D10~D19 are compared with D0 and the result is stored in D50~D54. If there is no matched result, the content of D50~D52 will all be 0.
2. D53 and D54 store the location of the smallest and biggest value. When there are more than one smallest and biggest values, the devices with bigger No. will be recorded.

```
 X0
─┤├──── SER  D10  D0  D50  K10
```

S_1	Content	Data to be compared	Data No.	Result
D10	88	**S_2**	0	
D11	100		1	Equal
D12	110		2	
D13	150		3	
D14	100	D0=K100	4	Equal
D15	300		5	
D16	100		6	Equal
D17	5		7	Smallest
D18	100		8	Equal
D19	500		9	Largest

n

D	Content	Explanation
D50	4	The total data numbers of equal value
D51	1	The number of the first equal value
D52	8	The number of the last equal value
D53	7	The number of the smallest value
D54	9	The number of the largest value

API	Mnemonic		Operands	Function	Controllers
62	D	ABSD	S1 S2 D n	Absolute Drum Sequencer	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1							*	*	*	*	*	*	*			ABSD: 9 steps
S_2											*	*	*			DABSD: 17 steps
D		*	*	*												
n					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

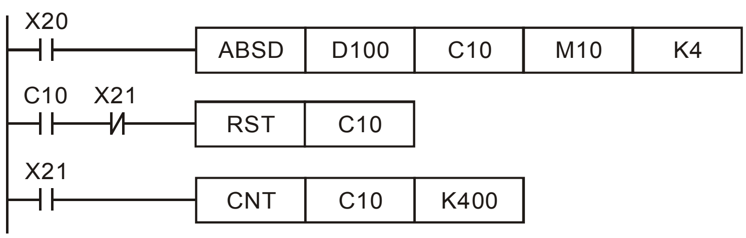
S_1: Start device of the data table **S_2**: No. of counter **D**: Start device for indicating comparison result **n**: Groups of data to be compared (**n**: 1~64)

Explanations:

1. ABSD instruction creates various output wave forms according to the current value of the counter designated by **S_2**. Usually, the instruction is applied for absolute cam control.
2. **S_2** of DABSD instruction can designate high speed counters. However, when the present value in the high speed counter is compared with the target value, the result cannot output immediately owing to the scan time. If an immediate output is required, please use DHSZ instruction that is exclusively for high speed counters.
3. When operand **S_1** uses KnX, KnY, KnM, KnS patterns, Kn should be K4 for 16-bit instruction and K8 for 32-bit instruction.

Program Example:

1. Before the execution of ABSD instruction, use MOV instruction to write all the set values into D100 ~ D107 in advance. The even-number D is for lower bound value and the odd-number D is for upper bound value.
2. When X10 = ON, the present value in counter C10 will be compared with the four groups of lower and upper bound values in D100 ~ D107. The comparison results will be stored in M10 ~ M13.
3. When X10 = OFF, the original ON/OFF status of M10 ~ M13 will be retained.

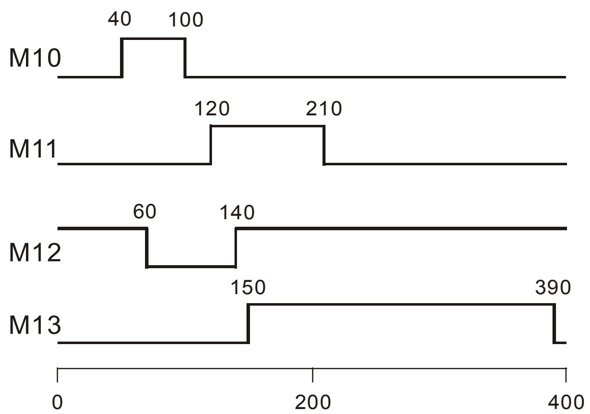


4. M10~ M13 = ON when the current value of C10 falls between lower and upper bounds.

Lower-bound value	Upper- bound value	Current value of C10	Output
D100= 40	D101 = 100	40≦C10≦100	M10 = ON
D102 = 120	D103 = 210	120≦C10≦210	M11 = ON
D104 = 140	D105 = 170	140≦C10≦170	M12 = ON
D106 = 150	D107 = 390	150≦C10≦390	M13 = ON

5. If the lower bound value is bigger than upper bound value, when C10<60 or C10 > 140, M12 = ON.

Lower- bound value	Upper- bound value	Current value of C10	Output
D100 = 40	D101 = 100	40≦C10≦100	M10 = ON
D102 = 120	D103 = 210	120≦C10≦210	M11 = ON
D104 = 140	D105 = 60	60≦C10≦140	M12 = OFF
D106 = 150	D107 = 390	150≦C10≦390	M13 = ON



API	Mnemonic	Operands	Function	Controllers			
63	INCD	S_1 S_2 D n	Incremental drum sequencer	ES2/EX2	SS2	SA2	SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Start device of the data table **S_2**: No. of counter **D**: Start device for indicating comparison result **n**: Number of data to be compared (**n**: 1~64)

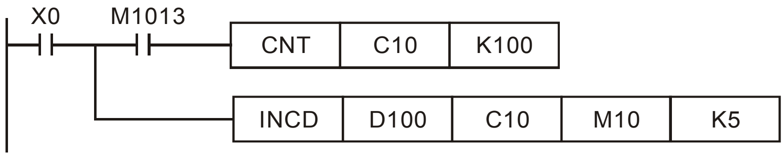
Explanations:

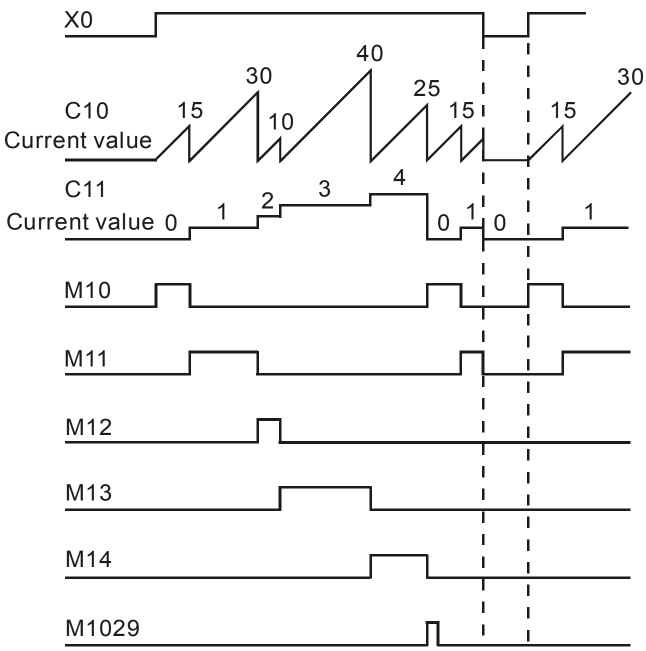
1. INCD instruction creates various output wave forms according to the current value of the counter designated by **S_2**. and **S_2+1**. Usually, the instruction is applied for relative cam control
2. The current value in **S_2** is compared with the set points specified by **S_1** (**n** consecutive devices) When value in **S_2** reaches the first set point, **S_2+1** counts once for indicating the number of present section, associated **D** turns ON, and **S_2** is reset then counts up from 0 again. When the drive contact of INCD instruction is OFF, the content in **S_2** and **S_2+1** will be cleared.
3. When operand **S_1** uses KnX, KnY, KnM, KnS patterns, Kn should be K4 for 16-bit instruction.
4. Operand **S_2** should be C0~C198 and occupies 2 consecutive counters.
5. When the comparison of **n** data has been completed, the execution completed flag M1029 = ON for one scan cycle.

Program Example:

1. Before the execution of INCD instruction, use MOV instruction to write all the set 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, C10 will be reset and count up from 0 again. Meanwhile C11 counts once for indicating the number of present section
3. When the content of C11 increase 1, M10~M14 will be ON sequentially. Please refer to the following timing diagram.
4. When the comparison of 5 data has been completed, the execution completed flag M1029 = ON for one scan cycle and C11 is reset for next comparison cycle.

5. 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 from the beginning.





API	Mnemonic	Operands	Function	Controllers			
64	TTMR	D n	Teaching Timer	ES2/EX2	SS2	SA2	SX2

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					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

D: Device No. for storing the ON time of the input **n**: setting of multiple (**n**: K0~K2)

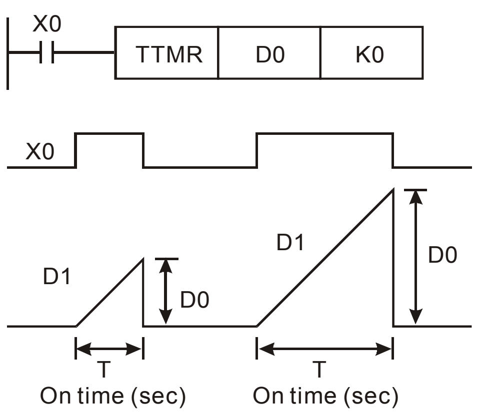
Explanations:

1. The ON time of the external button switch is measured and stored in **D + 1**(unit: 100ms). Value in **D + 1** is multiplied with a multiple specified by **n** and stored in **D** (unit: sec).
2. When **n** = K0, the value in **D + 1**(unit: 100ms) is multiplied with 1 and converted to **D** (unit: sec). When **n** = K1, the value in **D + 1**(unit: 100ms) is multiplied with 10 and converted to **D** (unit: sec). When **n** = K2, the value in **D + 1**(unit: 100ms) is multiplied with 100 and converted to **D** (unit: sec).
3. TTMR instruction can be used max 8 times in a program.

Program Example 1:

1. The duration that input X0 is pressed (ON duration of X0) will be stored in D1. The value in D1, multiplied by a multiple specified by **n**, is then moved to D0. In this case, the button switch can be used to adjust the set value of a timer.

2. When X0 = OFF, the content of D1 will be reset but the content of D0 remains.

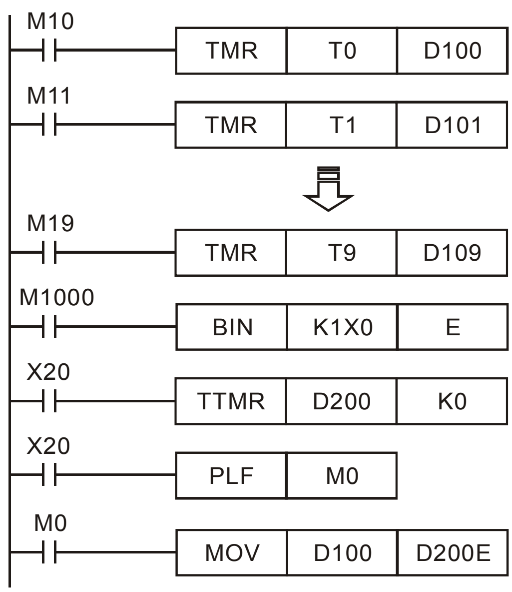


3. If ON duration of X0 is T sec, the relation between D0, D1 and **n** are shown as the table below.

n	D0 (unit: sec)	D1 (unit: 100 ms)
K0	T (sec) ×1	D1 = D0×10
K1	T (sec) ×10	D1 = D0
K2	T (sec) ×100	D1 = D0/10

Program Example 2:

1. Use TMR instruction to write in 10 groups of set time.
2. Write the set values into D100 ~ D109 in advance
3. The timer resolution is 0.1 sec for timers T0 ~ T9 and 1 sec for the teaching timer.
4. Connect the 1-bit DIP switch to X0 ~ X3 and use BIN instruction to convert the set value of the switch into a bin value and store it 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 teaching timer button X20 is released.
7. Use the set number of the DIP switch as the index pointer and send the content in D200 to D100E (D100 ~ D109).



Note:

The TTMR instruction can only be used 8 times in a program. If TTMR is used in a CALL subroutine or interrupt subroutine, it only can be use once.

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
65	STMR	S m D	Special Timer				

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		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: No. of timer (T0~T183) **m**: Set value in timer (**m** = 1~32,767, unit: 100ms)

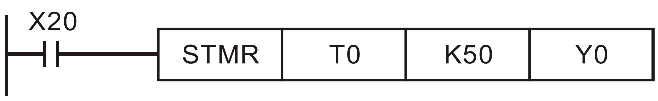
D: Start No. of output devices (occupies 4 consecutive devices)

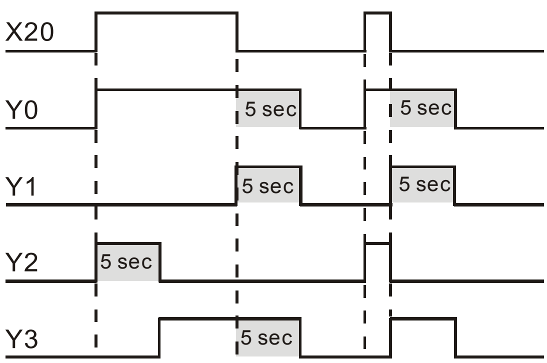
Explanations:

1. STMR instruction is specifically used for delay-OFF, ON/OFF triggered timer and flashing circuit.
2. The timer number (**S**) specified by STMR instruction can be used only once

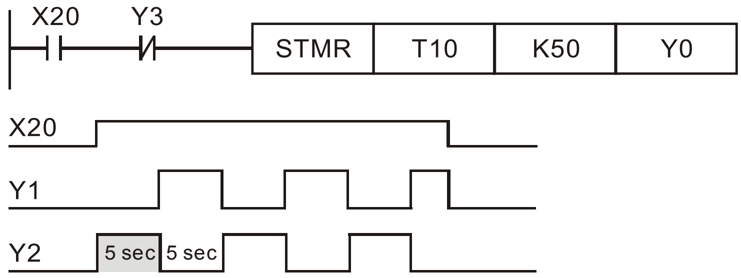
Program Example:

1. When X20 = ON, STMR sets T0 as the 5 sec special timer.
2. Y0 is the delay-OFF contact. When X20 is triggered, Y0 = ON; When X20 is OFF, Y0 = OFF after a 5 sec delay.
3. When X20 goes from ON to OFF, Y1 = ON for 5 seconds.
4. When X20 goes from OFF to ON, Y2 = ON for 5 seconds.
5. When X20 goes from OFF to ON, Y3 = ON after a 5 second delay. When X20 turns from ON to OFF, Y3 = OFF after a 5 second delay.





6. Apply a NC contact Y3 after the drive contact X20, and Y1, Y2 will form a flashing circuit output. When X20 turns OFF, Y0, Y1 and Y3 = OFF and the content of T10 will be reset.



API	Mnemonic			Operands	Function	Controllers
66		ALT	P	D	Alternate State	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
D		*	*	*												ALT, ALTP: 3 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

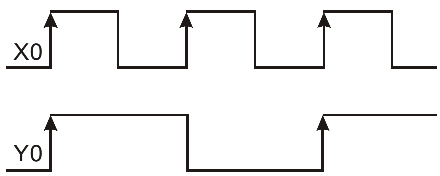
D: Destination device

Explanations:

1. The status of **D** is alternated every time when the ALT instruction is executed.
2. When ALT instruction is executed, ON/OFF state of **D** will be switched which is usually applied on switching two operation modes, e.g. Start/Stop
3. This instruction is generally used in pulse execution mode (ALTP).

Program Example 1:

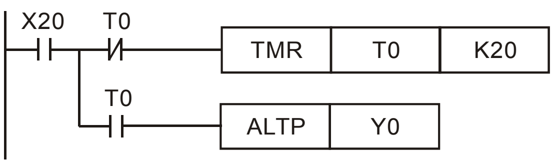
When X0 goes from OFF to ON, Y0 will be ON. When X0 goes from OFF to ON for the second time, Y0 will be OFF.



Program Example 2:

Creating a flashing circuit by applying ALTP with a timer

When X20 = ON, T0 will generate a pulse every two seconds and output Y0 will be switched between ON and OFF by the pulses from T0.



API	Mnemonic		Operands	Function	Controllers
67	D	RAMP	(S1) (S2) (D) (n)	Ramp variable Value	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1													*			RAMP: 9 steps
S_2													*			DRAMP: 17 steps
D													*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

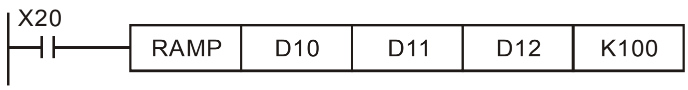
S_1: Start of ramp signal **S_2**: End of ramp signal **D**: Current value of ramp signal (occupies 2 consecutive devices) **n**: Times for scan (**n**: 1~32,767)

Explanations:

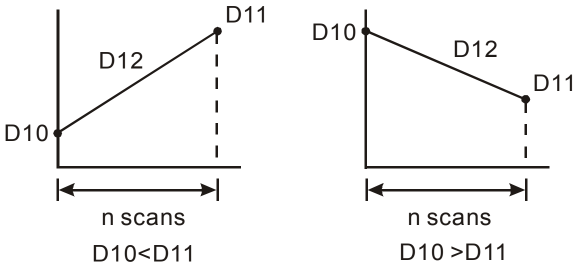
1. This instruction creates a ramp output. A ramp output linearity depends on a consistent scan time. Therefore, scan time has to be fixed before executing RAMP instruction.
2. When RAMP instruction is executed, the ramp signal will vary from **S_1** to **S_2**. Current value of ramp signal is stored in **D** and **D+1** stores the current number of accumulated scans. When ramp signal reaches **S_2**, or when the drive contact of RAMP instruction turns OFF, the content in **D** varies according to the setting of M1026 which is explained later in **Points to note**.
3. When **n** specifies a D register, the value in D cannot be modified during the execution of the instruction. Please modify the content of D when the instruction is stopped.
4. When this instruction is applied with analog output function, Ramp start and Ramp stop function can be achieved.

Program example:

1. Before executing the instruction, first drive M1039 = ON to fix the scan time. Use MOV instruction to write the fixed scan time to the special data register D1039. Assume the scan time is 30ms and take the below program for example, n = K100, the time for D10 to increase to D11 will be 3 seconds (30ms × 100).
2. When X20 goes OFF, the instruction will stop its execution. When X10 goes ON again, the content in D12 will be reset to 0 for recalculation
3. When M1026 = OFF, M1029 will be ON to indicate the completion of ramp process and the content in D12 will be reset to the set value in D10.
4. Set the Start and End of ramp signal in D10 and D11. When X20 = ON, D10 increases towards D11, the current value of the variation is stored in D12 and the number of current scans is stored in D13.



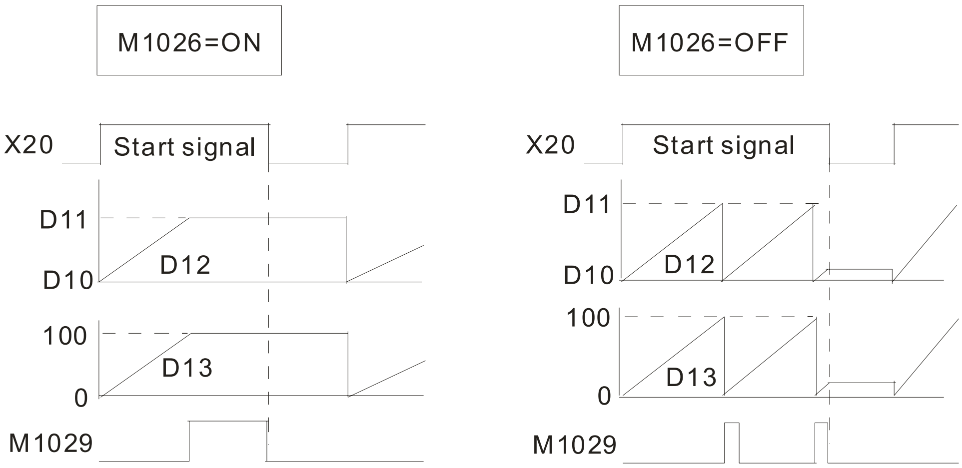
If X20 = ON,



The scan times is stored in D13

Points to note:

The variation of the content in D12 according to ON/OFF state of M1026 (Ramp mode selection):



API	Mnemonic		Operands	Function	Controllers
68	DTM	P	S_1 D m n	Data Transform and Move	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S													*			DTM: 9 steps
D													*			
m					*	*							*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Start device of the source data stack **D**: Start device of the destination data stack **m**: Transformation mode **n**: Length of source data stack

Explanations:

1. For parameter settings of operand **m**, please refer to the following description. K, H, D devices can be specified by operand **m.** If the set value is not in the available range, no transformation or move operation will be executed and no error will be detected.
2. K, H, D devices can be specified by operand **n,** which indicates the length of the source data stack. The available range for **n** is 1~256. If the set value falls out of available range, PLC will take the max value (256) or the min value (1) as the set value automatically.
3. Explanations on parameter settings of **m** operand:

   K0: With n = 4, transform 8-bit data into 16-bit data (Hi-byte, Lo-byte) in the following rule:

Hi-byte	Lo-byte
	①
	②
	③
	④

⇨

Hi-byte	Lo-byte
①	②
③	④

   K1: With n = 4, transform 8-bit data into 16-bit data (Lo-byte, Hi-byte) in the following rule:

Hi-byte	Lo-byte
	①
	②
	③
	④

⇨

Hi-byte	Lo-byte
②	①
④	③

K2: With n = 2, transform 16-bit data (Hi-byte, Lo-byte) into 8-bit data in the following rule:

Hi-byte	Lo-byte
①	②
③	④

Hi-byte	Lo-byte
	①
	②
	③
	④

K3: With n = 2, transform 16-bit data (Lo-byte, Hi-byte) into 8-bit data in the following rule:

Hi-byte	Lo-byte
①	②
③	④

Hi-byte	Lo-byte
	②
	①
	④
	③

K4: With n = 3, transform 8-bit HEX data into ASCII data (higher 4 bits, lower 4 bits) in the following rule:

Hi-byte	Lo-byte
	①
	②
	③

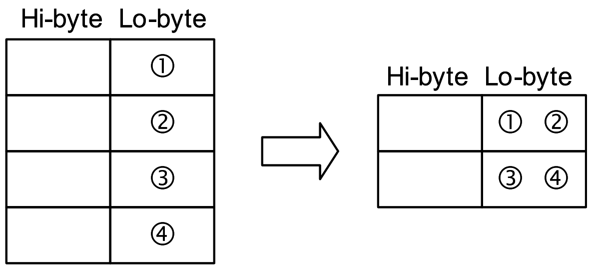
Hi-byte	Lo-byte
	①H
	①L
	②H
	②L
	③H
	③L

K5: With n = 3, transform 8-bit HEX data into ASCII data (lower 4 bits, higher 4 bits) in the following rule:

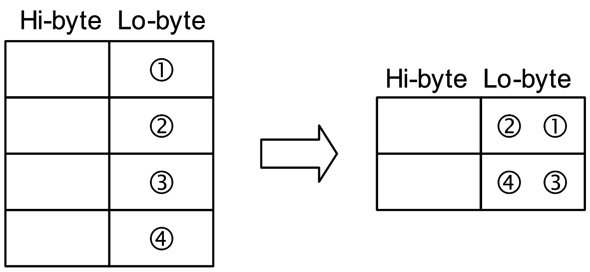
Hi-byte	Lo-byte
	①
	②
	③

Hi-byte	Lo-byte
	①L
	①H
	②L
	②H
	③L
	③H

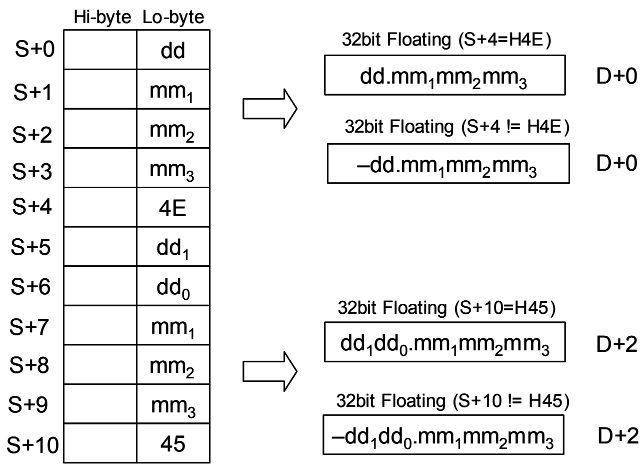
K6: When n = 4, transform 8-bit ASCII data (higher 4 bits, lower 4 bits) into HEX data in the following rule: (ASCII value to be transformed includes 0 ~ 9 (0x30~0x39), A ~ F (0x41~0x46), and a ~ f (0x61~0x66).)



K7: When n = 4, transform 8-bit ASCII data (lower 4 bits, higher 4 bits) into HEX data in the following rule:



K8: Transform 8-bit GPS data into 32-bit floating point data in the following rule:



K9: Calculate the optimal frequency for positioning instructions with ramp up/ down function. Users only need to set up the total number of pulses for positioning and the total time for positioning first, DTM instruction will automatically calculate the optimal max output frequency as well as the optimal start frequency for positioning instructions with ramp-up/down function such as PLSR, DDRVI and DCLLM.

Points to note:

1. When the calculation results exceed the max frequency of PLC, the output frequency will be set as 0.
2. When the total of ramp-up and ramp-down time exceeds the total time for operation, PLC will change the total time for operation (**S**+2) into "ramp-up time (**S**+3) + ramp-down time (**S**+4) + 1" automatically.

Explanation on operands:

S+0, **S**+1: Total number of pulses for operation (32-bit)

S+2: Total time for operation (unit: ms)

S+3: Ramp-up time (unit: ms)

S+4: Ramp-down time(unit: ms)

D+0, **D**+1: Optimal max output frequency (unit: Hz) (32-bit)

D+2: Optimal start frequency (Unit: Hz)

n: Reserved

3

K11: Conversion from Local Time to Local Sidereal Time

Unlike the common local time defined by time zones, local sidereal time is calculated based on actual longitude. The conversion helps the user obtain the more accurate time difference of each location within the same time zone.

Explanation on operands:

S+0, **S**+1: Longitude (32-bit floating point value; East: positive, West: negative)

S+2: Time zone (16-bit integer; unit: hour)

S+3~ **S**+8: Year, Month, Day, Hour, Minute, Second of local time (16-bit integer)

D+0~**D**+5: Year, Month, Day, Hour, Minute, Second of the converted local sidereal time (16-bit integer)

n: Reserved

Example:

Input: Longitude F121.55, Time zone: +8, Local time: AM 8:00:00, Jan/6/2011

Conversion results: AM 8:06:12, Jan/6/2011

K12: Proportional Value Calculation Function of Multi-point Areas (16-bit values)

Explanation on operands (16-bit values):

S: input value

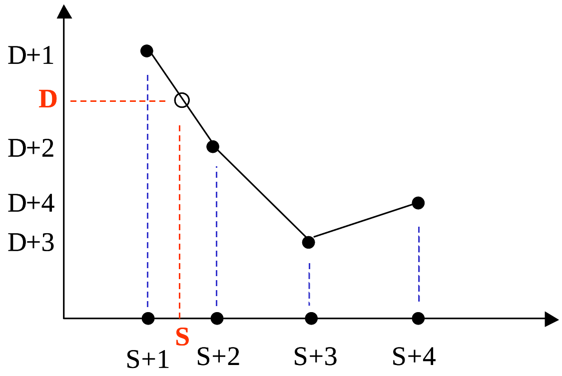
S+1, **S**+2….. **S**+n: set values of multi-point areas. **S**+1 must be the minimum value, **S**+2 must be larger than **S**+1 and so on. Therefore, **S**+n must be the maximum value.

D: output value gotten from the proportional value calculation

D+1, **D** +2 … **D**+n: the range of values gotten from the proportional value calculation

n: set values of multi-point areas. The range of set values is K2~K50. When the set value exceeds the range, it will not be executed.

The sample curve: (n is set to be K4)



The explanation of the sample:

1. When input value S is larger than S+1 (S_1 for short) and smaller than S+2 (S_2 for short), D+1 (D_1 for short) and D+2 (D_2 for short), D= ((S – S_1) x (D_2 – D_1) / (S_2 – S_1)) + D_1.
2. When input value S is smaller than S+1, D= D+1; when input value S is larger than S+n, D= D+n.
3. The operation of instructions uses floating-point values. After the decimal value of the output values is omitted, the value will be output in the 16-bit form.

K13: Proportional Value Calculation Function of Multi-point Areas (32-bit values)

The explanations of source and destination devices are illustrated as the explanation of K12, but devices S and D are indicated by 32-bit values.

K14: Proportional Value Calculation Function of Multi-point Areas (floating-point values)

The explanations of source and destination devices are illustrated as the explanation of K12, but devices S and D are indicated by 32-bit floating-point values.

K16: String combination

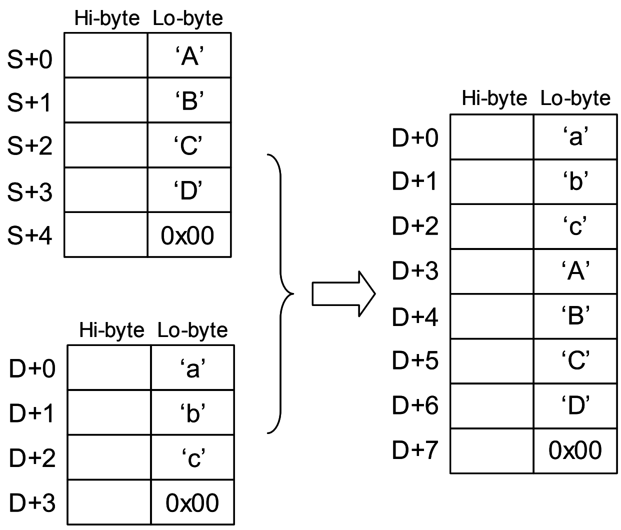
Explanation:

The system searches for the location of ETX (value 0x00) of the destination data string (lower 8 bits), then copies the data string starting of the source register (lower 8 bits) to the end of the destination data string. The source data string will be copied in byte order until the ETX (value 0x00) is reached.

Points to note:

The operand **n** sets the max data length after the string combination (max 256). If the ETX is not reached after the combination, the location indicated by **n** will be the ETX and filled with 0x00.

The combination will be performed in the following rule:

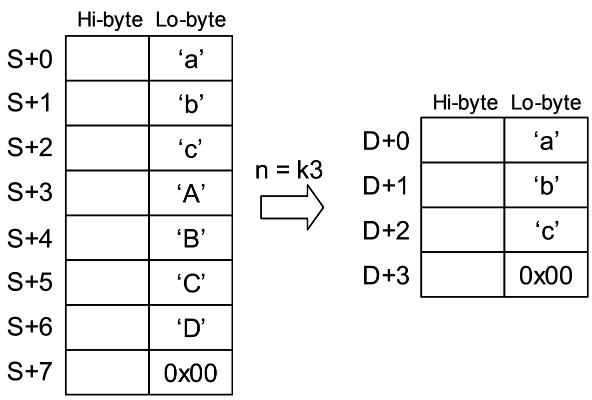


K17: String capture

Explanations:

The system copies the source data string (lower 8 bits) with the data length specified by operand n to the destination registers, where the n+1 register will be filled with 0x00. If value 0x00 is reached before the specified capture length n is completed, the capture will also be ended.

The capture will be performed in the following rule:



K18: Convert data string to floating point value

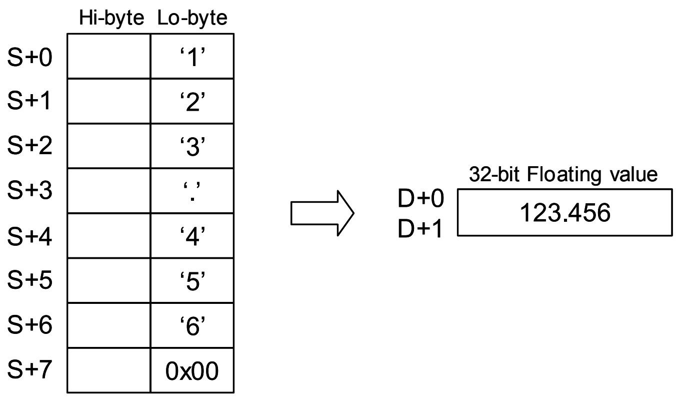
Explanations:

The system converts **n** words (lower 8 bits) of the source data string (decimal point is not included) to floating point value and stores the converted value in the destination device.

Points to note:

1. Operand **n** sets the number of total digits for the converted floating value. Max 8 digits are applicable and the value over **n** digit will be omitted. For example, **n** = K6, data string "123.45678" will be converted to "123.456".
2. When there are characters other than numbers 0~9 or the decimal point in the source data string, the character before the decimal point will be regarded as 0, and the value after the decimal point will be regarded as the ETX.
3. If the source data string contains no decimal point, the converted value will be displayed by a **n**-digit floating point value automatically.

The conversion will be performed in the following rule:



K19: Convert floating point value to data string

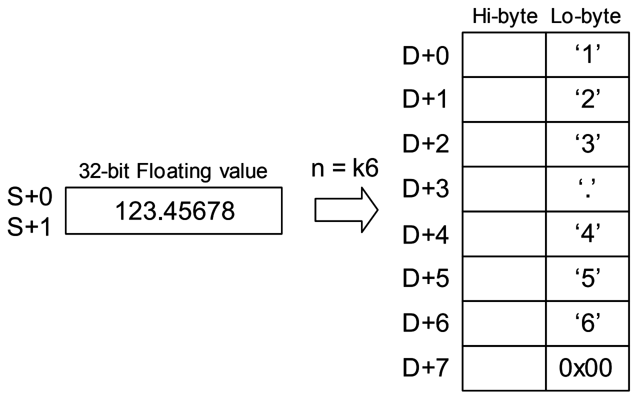
Explanations:

The system converts the floating point value in the source device S to data string with specified length **n** (decimal point is not included).

Points to note:

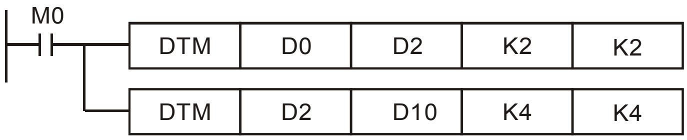
1. Operand **n** sets the number of total digits for the floating point value to be converted. Max 8 digits are applicable and the value over **n** digit will be omitted. For example, **n** = K6, floating value F123.45678 will be converted to data string "123.456".
2. When the digits of source value are more than the specified **n** digits, only the **n** digits from the left will be converted. For example, source value F123456.78 with **n**=K4 will be converted as data string "1234".
3. If the source value is a decimal value without integers, e.g. 0.1234, the converted data string will be ".1234" where the first digit is the decimal point.

The conversion will be performed in the following rule:



Program Example 1: K2, K4

1. When M0 = ON, transform 16-bit data in D0, D1 into ASCII data in the following order: H byte - L byte - H byte - Low byte, and store the results in D10 ~ D17.



2. Value of source devices D0, D1:

Register	D0	D1
Value	H1234	H5678

3. When the 1st DTM instruction executes (m=K2), ELC transforms the 16-bit data (Hi-byte, Lo-byte) into 8-bit data and move to registers D2~D5.

Register	D2	D3	D4	D5
Value	H12	H34	H56	H78

4. When the 2nd DTM instruction executes (m=K4), ELC transforms the 8-bit HEX data into ASCII data and move to registers D10~D17.

Register	D10	D11	D12	D13	D14	D15	D16	D17
Value	H0031	H0032	H0033	H0034	H0035	H0036	H0037	H0038

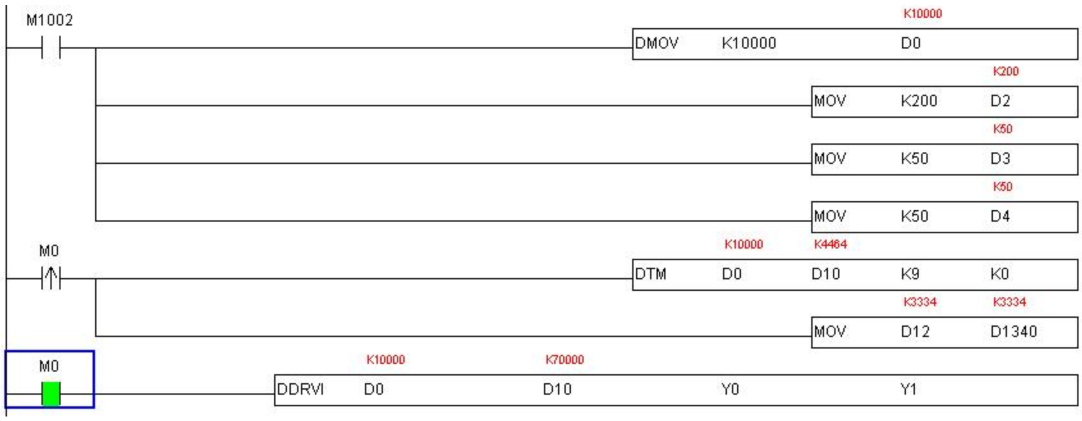
Program Example 2: K9

1. Set up total number of pulses, total time, ramp-up time and ramp-down time in source device starting with D0. Execute DTM instruction and the optimal max frequency as well as optimal start frequency can be obtained and executed by positioning instructions.
2. Assume the data of source device is set up as below:

Total Pulses	Total Time	Ramp-up Time	Ramp-down Time
D0, D1	D2	D3	D4
K10000	K200	K50	K50

3. The optimal positioning results can be obtained as below:

Optimal max frequency	Optimal start frequency
D10, D11	D12
K70000	K3334



API	Mnemonic		Operands	Function	Controllers
69	D	SORT	S m1 m2 D n	Data sort	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	SORT: 11 steps
S													*			DSORT: 21 steps
m_1					*	*										
m_2					*	*										
D													*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

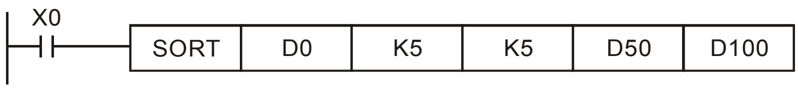
S: Start device for the source data **m_1**: Groups of data to be sorted (**m_1** =1~32) **m_2**: Number of columns in the table (**m_2** =1~6) **D**: Start device for the sorted data **n**: The No. of column to be sorted. (**n**=1~ **m2**)

Explanations:

1. The sorted data is stored in the **m_1** × **m_2** registers starting from the device designated in **D**. Therefore, if **S** and **D** designate the same register, the sorted results will be the same.
2. SORT instruction is completed after **m_1** times of scan. Once the SORT instruction is completed, the Flag M1029 (Execution completed flag) = ON.
3. There is no limitation on the times of using this instruction in the program. However, only one instruction can be executed at a time

Program Example:

When X0 = ON, the sorting process starts. When the sorting is completed, M1029 will be ON. DO NOT change the data to be sorted during the execution of the instruction. If the sorting needs to be executed again, turn X0 from OFF to ON again.



Example table of data sort

Columns of data: $\mathbf{m_2}$

Groups of data: $\mathbf{m_1}$

Column / Row	Data Column 1 Students No.	2 English	3 Math.	4 Physics	5 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

Sort data table when D100 = K3

Columns of data: $\mathbf{m_2}$

Groups of data: $\mathbf{m_1}$

Column / Row	Data Column 1 Students No.	2 English	3 Math.	4 Physics	5 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

Sort data table when D100 = K5

Columns of data: $\mathbf{m_2}$

Groups of data: $\mathbf{m_1}$

Column / Row	Data Column 1 Students No.	2 English	3 Math.	4 Physics	5 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

API	Mnemonic			Operands	Function	Controllers
70	D	TKY		(S) (D1) (D2)	Ten key input	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S	*	*	*	*												TKY: 7 steps
D_1								*	*	*	*	*	*	*	*	DTKY: 13 steps
D_2		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

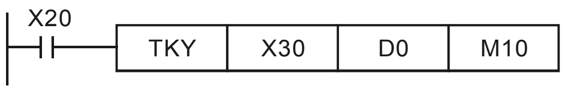
S: Start device for key input (occupies 10 consecutive devices) **D_1**: Device for storing keyed-in value **D_2**: Output signal (occupies 11 consecutive devices)

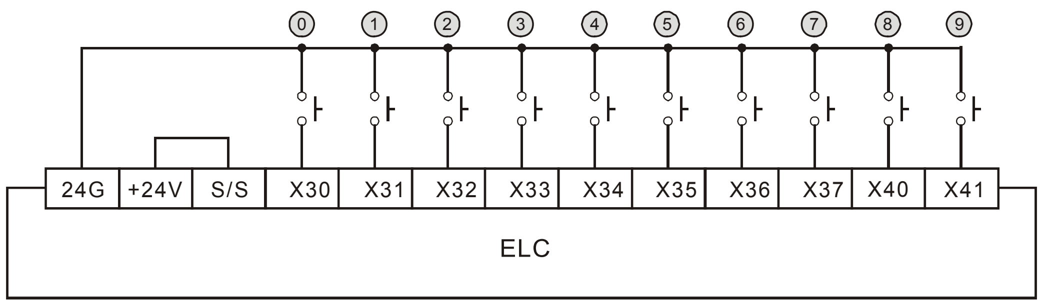
Explanations:

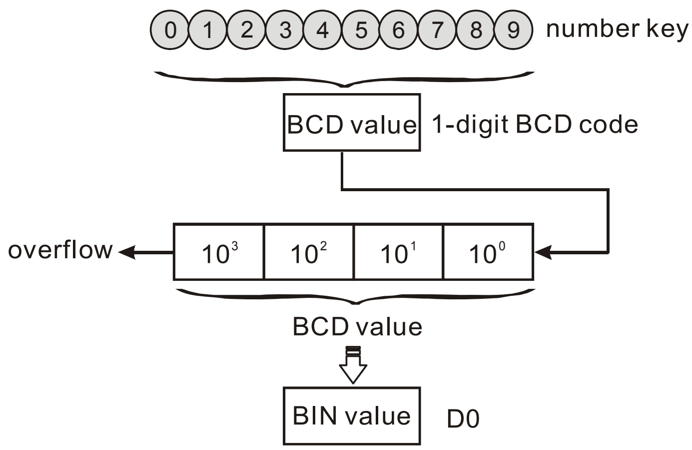
1. This instruction designates 10 external input points (corresponding to decimal numbers 0 ~ 9) starting from **S**, connecting to 10 keys respectively. Input point started from **S** triggers associated device in **D_2** and **D_2** maps to a decimal value, a 4-digit decimal value 0~9,999 (16-bit instruction) or an 8-digit value 0~99,999,999 (32-bit instruction). The decimal value is stored in **D_1**.
2. There is no limitation on the times of using this instruction in the program, however only one instruction is allowed to be executed at the same time.

Program Example:

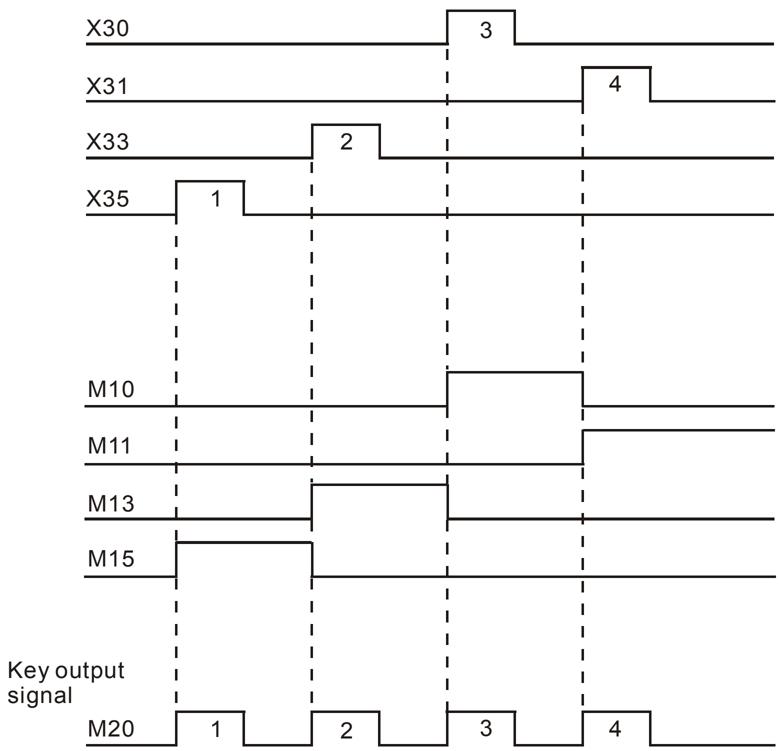
1. Connect the 10 input points starting from X30 to the 10 keys (0 ~ 9). When X20 = ON, the instruction will be executed and the key-in values will be stored in D0 in BIN form. The key status will be stored in M10 ~ M19.







2. As shown in the timing diagram below, four keys connected with X35, X33, X31 and X30 are pressed in order. Therefore, the number 5,301 is generated and stored in D0. 9,999 is the maximum value allowed for D0. If the entered number exceeds the available range, the highest digit performs overflow.
3. When X35 is pressed, M15 remains ON until another key is pressed and the rule applies to other inputs.
4. M20 = ON when any of the keys is pressed.
5. When X20 is OFF, the value in D0 remains unchanged but M10~M20 will be OFF.



API	Mnemonic		Operands	Function	Controllers
					ES2/EX2 SS2 SA2 SX2
71	D	HKY	S D1 D2 D3	Hexadecimal key input	

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	HKY: 9 steps
S	*															DHKY: 17 steps
D_1		*														
D_2											*	*	*	*	*	
D_3		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

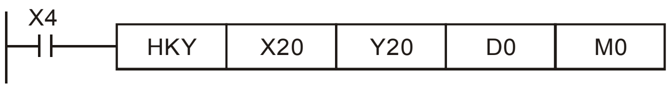
S: The start of input devices (occupies 4 consecutive devices) **D_1**: The start of output devices (occupies 4 consecutive devices) **D_2**: Device for storing key input value **D_3**: Key input status (occupies 8 consecutive devices)

Explanations:

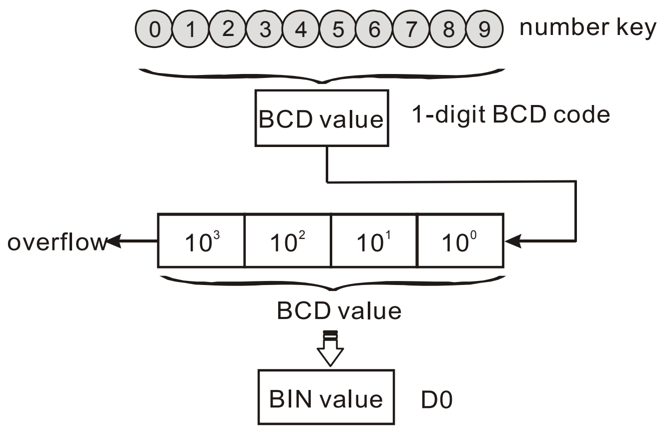
1. This instruction creates a 16-key keyboard by a multiplex of 4 consecutive external input devices from **S** and 4 consecutive external output devices from **D_1**. By matrix scan, the key input value will be stored in **D_2**. **D_3** stores the condition of keys A~F and indicates the key input status of both 0~9 and A~F..
2. M1029 = ON for a scan cycle every time when a key is pressed.
3. If several keys are pressed, only the first pressed key is valid.
4. **D_2** maps to a decimal value, a 4-digit decimal value 0~9,999 (16-bit instruction) or an 8-digit value 0~99,999,999 (32-bit instruction). If the entered number exceeds the available range, i.e. 4 digit in 16-bit and 8 digits in 32-bit instruction, the highest digit performs overflow
5. There is no limitation on the times of using this instruction in the program, but only one instruction is allowed to be executed in the same scan time.

Program Example:

1. Designate 4 input points X20 ~ X23 and the other 4 output points Y20 ~ Y23 to construct a 16-key keyboard. When X4 = ON, the instruction will be executed and the keyed-in value will be stored in D0 in BIN form. The key status will be stored in M10 ~ M19.



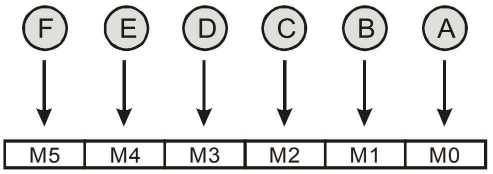
2. Input keys 0~9:



3. Input keys A~F:

a) When A is pressed, M0 will be ON and retained. When D is pressed next, M0 will be OFF, M3 will be ON and retained..

b) If two or more keys are pressed at the same time, only the key activated first is effective.



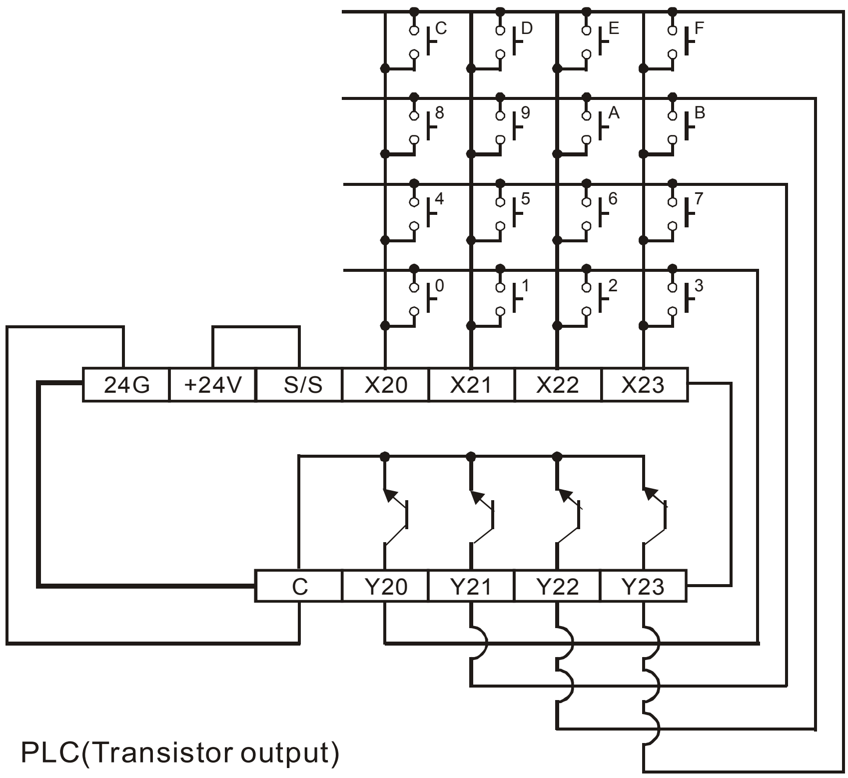
4. Key input status:

a) When any key of A ~ F is pressed, M6 = ON for one scan time.

b) When any key of 0 ~ 9 is pressed, M7 = ON for one scan time.

5. When the drive contact X4 = OFF, the value d in D0 remains unchanged but M0~M7 = OFF.

6. External wiring:



Points to note:

1. When HKY instruction is executed, 8 scan cycles (matrix scan) are required for reading the input value successfully. A scan cycle that is too long or too short may cause the input to be read incorrectly. In this case we suggest the following solutions:
   a) If the scan cycle is too short, I/O may not be able to respond in time, resulting in incorrect input values. To solve this problem please fix the scan time.
   b) If the scan period is too long, the key may respond slowly. In this case, write this instruction into the time-interrupt subroutine to fix the execution time for this instruction.
2. The function of flag M1167:
   a) When M1167 = ON, HKY instruction can input hexadecimal value consists of 0~F.
   b) When M1167 = OFF, A~F of HKY instruction are used as function keys.

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
72	DSW	S D₁ D₂ n	DIP Switch				

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S	*															DSW: 9 steps
D_1		*														
D_2											*	*	*			
n					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

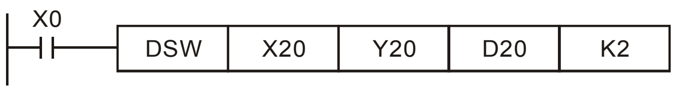
S: The Start of input devices **D_1**: The Start of output devices **D_2**: Device for storing switch input value **n**: Groups of switches (**n** = 1~2)

Explanations:

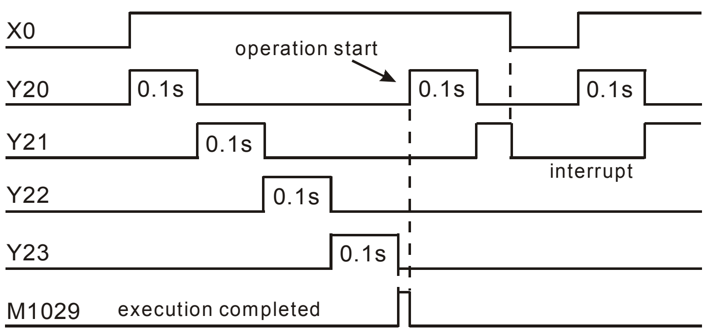
1. This instruction creates 1(2) group of 4-digit DIP switch by the combination of 4(8) consecutive input points starting from **S** and 4 consecutive output points starting from **D_1**. The set value will be read in **D_2** and the value in **n** specifies the number of groups (1~2) of the DIP switch.
2. **n** = K1, **D_2** occupies 1 register. **n** = K2, **D_2** occupies 2 consecutive registers..
3. There is no limitation on the times of using this instruction in the program, however only one instruction is allowed to be executed at the same scan time.

Program Example:

1. The first group of DIP switches consists of X20 ~ X23 and Y20 ~ Y23. The second group of switches consists of X24 ~ X27 and Y20 ~ Y23. When X10 = ON, the instruction will be executed and the set value of the first switch will be read and converted into BIN value then stored in D20. BIN value of 2nd switch will be stored in D21.

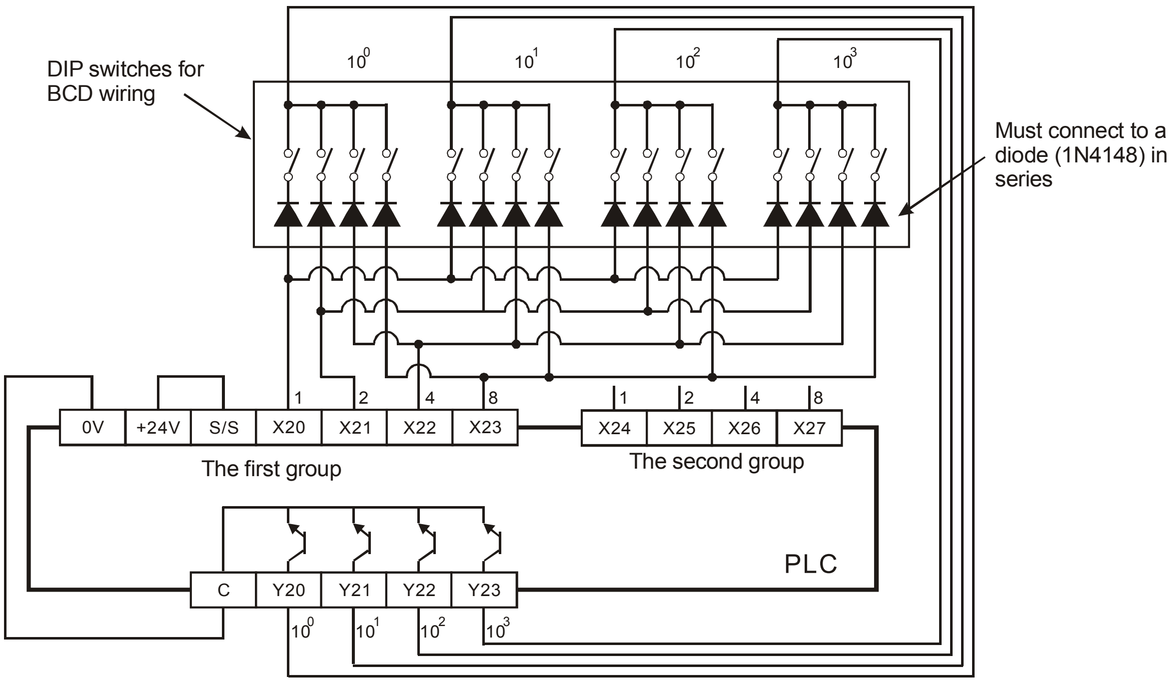


2. When X0 = ON, Y20~Y23 are scanned repeatedly. M1029 = ON for a scan time when a scan cycle from Y20 to Y23 is completed.



3. Please use transistor output for Y20 ~ Y23. Every pin 1, 2, 4, 8 shall be connected to a diode (0.1A/50V) in series before connecting to the input terminals on PLC.

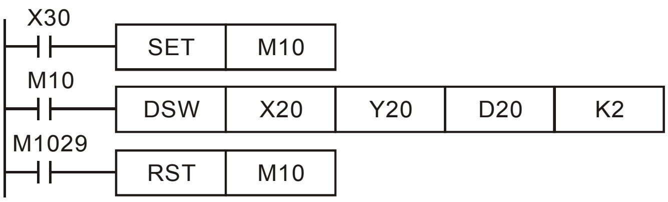
Wiring diagram of DIP switch:



Points to note:

When the terminals to be scanned are relay outputs, the following program methods can be applied:

1. When X30 = ON, DSW instruction will be executed. When X30 goes OFF, M10 remains ON until the current scan cycle of output terminals is completed..
2. If the drive contact X30 uses button switch, M10 turns off only when the current scan cycle on outputs is completed, so that a correct value from DIP switch can be read. In addition, the continuous scan cycle on outputs will be performed only when the drive contact is pressed and held. Applying this method can reduce the driving frequency of relay outputs so as to extend to life-span of relays.



API	Mnemonic	Operands	Function	Controllers
73	SEGD P	S D	7-segment decoder	ES2/EX2 SS2 SA2 SX2

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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device for decoding   **D**: Output device after decoding

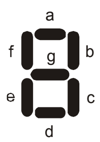
Explanations:

The instruction decodes the lower 4 bits (Hex data: 0 to 9, A to F) of source device **S** and stores the decoded data in lower 8 bits of **D** so as to form a 7-segment display.

Program Example:

When X20 = ON, the content of the lower 4 bits (b0~b3) of D10 will be decoded into the 7-segment display. . The decoded results will be stored in Y20~Y27. If the source data exceeds 4bits, still only lower 4 bits will be decoded.

Decoding table of the 7-segment display:

Hex	Bit combi-nation	Composition of the 7-segment display	B0(a)	B1(b)	B2(c)	B3(d)	B4(e)	B5(f)	B6(g)	Data displayed
0	0000		ON	ON	ON	ON	ON	ON	OFF	0
1	0001		OFF	ON	ON	OFF	OFF	OFF	OFF	1
2	0010		ON	ON	OFF	ON	ON	OFF	ON	2
3	0011		ON	ON	ON	ON	OFF	OFF	ON	3
4	0100		OFF	ON	ON	OFF	OFF	ON	ON	4
5	0101		ON	OFF	ON	ON	OFF	ON	ON	5
6	0110		ON	OFF	ON	ON	ON	ON	ON	6
7	0111		ON	ON	ON	OFF	OFF	ON	OFF	7
8	1000		ON	ON	ON	ON	ON	ON	ON	8
9	1001		ON	ON	ON	ON	OFF	ON	ON	9
A	1010		ON	ON	ON	OFF	ON	ON	ON	A
B	1011		OFF	OFF	ON	ON	ON	ON	ON	b
C	1100		ON	OFF	OFF	ON	ON	ON	OFF	C
D	1101		OFF	ON	ON	ON	ON	OFF	ON	d
E	1110		ON	OFF	OFF	ON	ON	ON	ON	E
F	1111		ON	OFF	OFF	OFF	ON	ON	ON	F

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
74	SEGL	S D n	7-segment with Latch				

Type / OP	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					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device storing the value to be displayed in 7-segment display **D**: Output device for 7-segment display

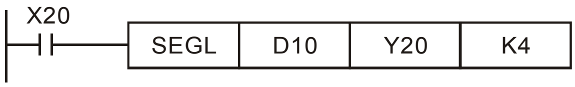
n: Configuration setting of output signal (**n** = 0~7)

Explanations:

1. This instruction occupies 8 or 12 consecutive external output points starting from **D** for displaying the data of 1 or 2 sets of 4-digit 7-segment display. Every digit of the 7-segment display carries a “Drive” which converts the BCD codes into 7-segment display signal. The drive also carries latch control signals to retain the display data of 7-segment display.
2. **n** specifies the number of sets of 7-segment display (1 set or 2 sets), and designates the positive / negative output of PLC and the 7-segment display.
3. When there is 1 set of 4-digit output, 8 output points will be occupied. When there are 2 sets of 4-digit output, 12 output points will be occupied
4. When the instruction is executed, the output terminals will be scanned circularly. When the drive contact goes from OFF to ON again during the execution of instruction, the scan will restart from the beginning of the output terminals.
5. Flag: When SEGL is completed, M1029 = ON for one scan cycle.
6. There is no limitation on the times of using this instruction in the program, however only one instruction is allowed to be executed at a time.

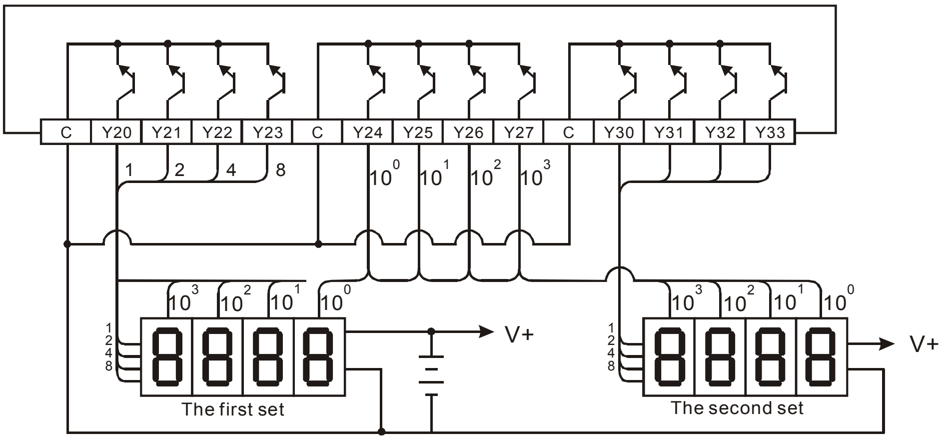
Program Example:

1. When X20 = ON, SEGL instruction executes and Y24~Y27 forms an output scan loop for 7-segment display. The value of D10 will be mapped to Y20~Y23, converted to BCD code and sent to the 1st set of 7-segment display. The value of D11 will be mapped to Y30~Y33, converted to BCD code and sent to the 2nd set of 7-segment display. If the values in D10 and D11 exceed 9,999, operational error will occur.



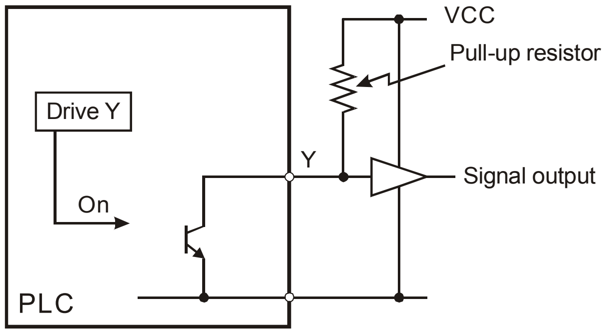
2. When X20 = ON, Y24~Y27 will be scanned in circles automatically. Each circle requires 12 scan cycles. M1029 = ON for a scan cycle whenever a circle is completed.
3. When there is 1 set of 4-digit 7-segment display, **n** = 0 ~ 3
   a) Connect the 7-segment display terminals 1, 2, 4, 8 in parallel then connect them to Y20 ~ Y23 on PLC. After this, connect the latch terminals of each digit to Y24 ~ Y27 on PLC.
   b) When X20 = ON, the content of D10 will be decoded through Y20 ~ Y23 and sent to 7-segment display in sequence by the circulation of Y24 ~ Y27
4. When there are 2 sets of 4-digit 7-segment display, **n** = 4 ~ 7
   a) Connect the 7-segment display terminals 1, 2, 4, 8 in parallel then connect them to Y30 ~ Y33 on PLC. After this, connect the latch terminals of each digit to Y24 ~ Y27 on PLC.
   b) The content in D10 is sent to the 1st set of 7-segment display. The content in D11 is sent to the 2nd set of 7-segment display. If D10 = K1234 and D11 = K4321, the 1st set will display 1 2 3 4, and the 2nd set will display 4 3 2 1.

Wiring of the 7-segment display scan output:



Points to note:

1. For executing this instruction, scan time must be longer than 10ms. If scan time is shorter than 10ms, please fix the scan time at 10ms.
2. If the output points of PLC is transistor output, please apply proper 7-segment display.
3. Operand **n** is used for setting up the polarity of the transistor output and the number of sets of the 4-digit 7-segment display.
4. The output point must be a transistor module of NPN output type with open collector outputs. The output has to connect to a pull-up resistor to VCC (less than 30VDC). When wiring, output should connect a pull-high resistor to VCC (less than 30VDC). Therefore, when output point Y is ON, the output signal will be LOW.



5. Positive logic (negative polarity) output of BCD code

BCD value				Y output (BCD code)				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

6. Negative logic (Positive polarity) output of BCD code

BCD value				Y output (BCD code)				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

7. Operation logic of output signal

Positive logic (negative polarity)		Negative logic (positive polarity)	
Drive signal (latch)	Data control signal	Drive signal (latch)	Data control signal
1	0	0	1

8. Parameter **n** settings:

Sets of 7-segment display	1 set				2 sets			
BCD code data control signal	+		−		+		−	
Drive (latch) signal	+	−	+	−	+	−	+	−
n	0	1	2	3	4	5	6	7

’＋’: Positive logic (Negative polarity) output

‘－’: Negative logic (Positive polarity) output

9. The polarity of PLC transistor output and the polarity of the 7-segment display input can be designated by the setting of **n**.

API	Mnemonic	Operands	Function	Controllers			
75	ARWS	(S) (D1) (D2) (n)	Arrow switch	ES2/EX2	SS2	SA2	SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S	*	*	*	*												ARWS: 9 steps
D_1											*	*	*	*	*	
D_2		*														
n					*	*										

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

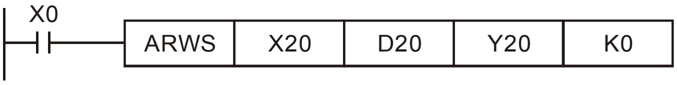
S: Start device for key input (occupies 4 consecutive devices) **D_1**: Device storing the value to be displayed in 7-segment display **D_2**: Output device for 7-segment display **n**: Configuration setting of output signal (**n** = 0~3). Please refer to explanations of SEGL instruction for the **n** usage.

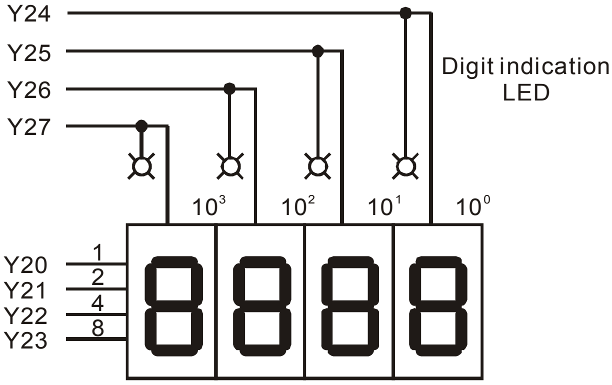
Explanations:

1. ARWS instruction displays the value set in device **D_1** on a set of 4-digit 7 segment display. PLC automatically converts the decimal value in **D_1** to BCD format for displaying on the 7 segment display. Each digit of the display can be modified by changing the value in **D_1** through the operation of the arrow switch.
2. Number of **D_2** only can be specified as a multiple of 10, e.g. Y0, Y10, Y20…etc.
3. Output points designated by this instruction should be transistor output.
4. When using this instruction, please fix the scan time, or place this instruction in the timer interruption subroutine (I610/I699, I710/I799).
5. There is no limitation on the times of using this instruction in the program, but only one instruction is allowed to be executed at a time.

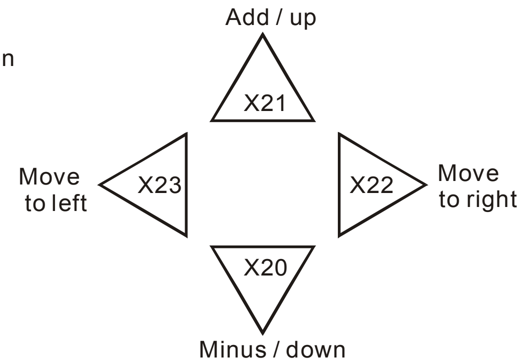
Program Example:

1. When the instruction is executed, X20 is defined as the Minus key, X21 is defined as the Add key, X22 is defined as the Right key and X23 is defined as the Left key. The keys are used to modify the set values (range: 0 ~ 9,999) stored in D20..
2. When X0 = ON, digit 10^3 will be the valid digit for setup. When Left key is pressed, the valid digit will shift as the following sequence: $10^3 \rightarrow 10^0 \rightarrow 10^1 \rightarrow 10^2 \rightarrow 10^3 \rightarrow 10^0$.
3. When Right key is pressed, the valid digit will shift as the following sequence: $10^3 \rightarrow 10^2 \rightarrow 10^1 \rightarrow 10^0 \rightarrow 10^3 \rightarrow 10^2$. Besides, the digit indicators (LED, Y24 to Y27) will be ON for indicating the position of the valid digit during shift operation.
4. When Add key is pressed, the content in the valid digit will change as 0 → 1 → 2 … → 8 → 9 → 0 →1. When Minus key is pressed, the content in the valid digit will change as 0 → 9 → 8 … → 1 → 0 → 9. The changed value will also be displayed in the 7-segment display.





7-segment display for the 4-digit set value



The 4 switches are used for moving the digits and modifying set values.

API	Mnemonic	Operands	Function	Controllers
76	ASC	S D	ASCII code conversion	ES2/EX2 SS2 SA2 SX2

Type / OP	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											*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

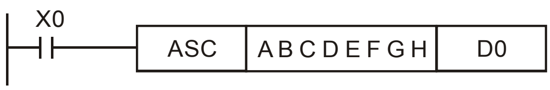
S: English letters to be converted into ASCII code **D**: Device for storing ASCII code

Explanation:

1. The ASC instruction converts 8 English letters stored in **S** and save the converted ASCII code in **D**. The value in **S** can be input by WPLSoft or ISPSoft.
2. If PLC is connected to a 7-segment display while executing ASC instruction, the error message can be displayed by English letters
3. Flag: M1161 (8/16 bit mode switch)

Program Example:

When X0 = ON, A~H is converted to ASCII code and stored in D0~D3.



	b15 High byte	Low byte b0
D0	42H (B)	41H (A)
D1	44H (D)	43H (C)
D2	46H (F)	45H (E)
D3	48H (H)	47H (G)

When M1161 = ON, every ASCII code converted from the letters will occupy the lower 8 bits (b7 ~ b0) of a register and the upper 8 bits are invalid (filled by 0), i.e. one register stores a letter

	b15 High byte	Low byte 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)

API	Mnemonic	Operands	Function	Controllers			
77	PR	(S) (D)	Print (ASCII Code Output)	ES2/EX2	SS2	SA2	SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S											*	*	*			PR: 5 steps
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

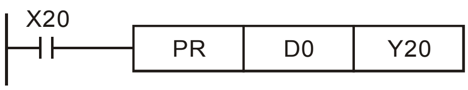
S: Device for storing ASCII code (occupies 4 consecutive devices) **D**: External ASCII code output points (occupies 10 consecutive devices)

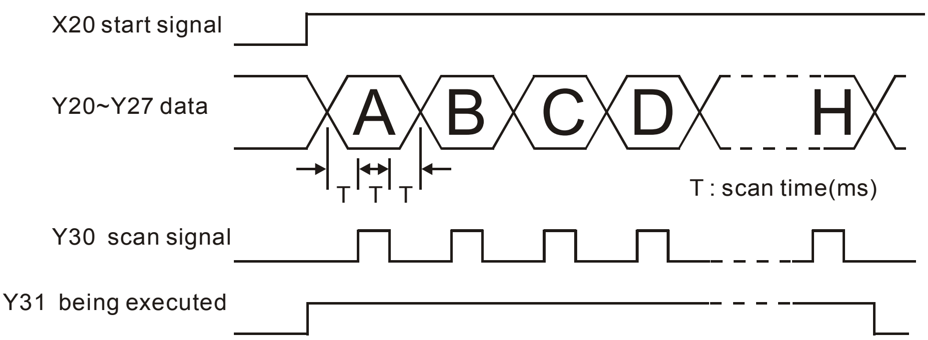
Explanations:

1. This instruction will output the ASCII codes in the 4 registers starting from **S** through output points started from **D.**
2. $\mathbf{D_0}$ ~ $\mathbf{D_7}$ map to source data (ASCII code) directly in order, $\mathbf{D_{10}}$ is the scan signal and $\mathbf{D_{11}}$ is the execution flag.
3. This instruction can only be used twice in the program.
4. Flags: M1029 (PR execution completed); M1027 (PR output mode selection).

Program Example 1:

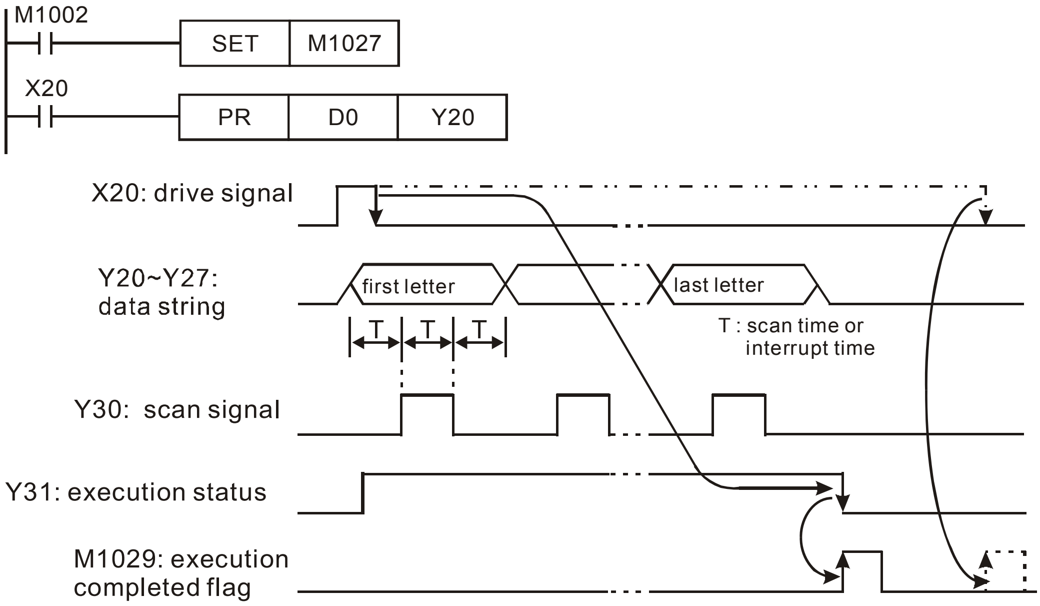
1. Use API 76 ASC to convert A ~ H into ASCII codes and store them in D0 ~ D3. After this, use this instruction to output the codes in sequence.
2. When M1027 = OFF and X20 = ON, the instruction will designate Y20 (lowest bit) ~ Y27 (highest bit) as the output points and Y30 as scan signals, Y31 as execution flag. In this mode, users can execute an output for 8 letters in sequence..
3. If X20 turns from ON → OFF during the execution of the instruction, the data output will be interrupted, and all the output points will be OFF. When X20 = ON again, the data output will start from the first letter again.





Program Example 2:

1. PR instruction supports ASCII data output of 8-bit data string when M1027 = OFF. When M1027 = ON, the PR instruction is able to execute an output of 1~16 bit data string.
2. When M1027 = ON and X20 = ON, this instruction will designate Y20 (lowest bit) ~ Y27 (highest bit) as the output points and Y30 as scan signals, Y31 as execution flag. In this mode, users can execute an output for 16 letters in sequence. In addition, if the drive contact X20 is OFF during execution, the data output will stop until a full data string is completed.
3. The data 00H (NULL) in a data string indicates the end of the string and the letters coming after will not be processed.
4. If the drive contact X20 is OFF during execution, the data output will stop until a full data string is completed. However, if X20 remains ON, execution completed flag M1029 will not be active as the timing diagram below.



Points to note:

1. Please use transistor output for the output points designated by this instruction.
2. When using this instruction, please fix the scan time or place this instruction in a timer interrupt subroutine.

API	Mnemonic			Operands	Function	Controllers
78	D	FROM	P	m1 m2 D n	Read CR data from Special Modules	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
m_1					*	*							*			FROM, FROMP: 9 steps
m_2					*	*							*			DFROM, DFROMP: 17 steps
D													*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

m_1: No. of special module **m_2**: CR# in special module to be read **D**: Device for storing read data **n**: Number of data to be read at a time

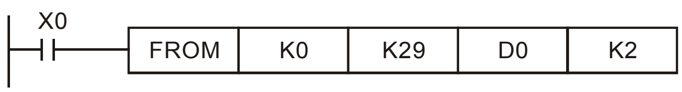
Explanations:

1. PLC uses this instruction to read CR (Control register) data from special modules.
2. Range of **m_1**: ES2/EX2/SS2: 0 ~ 7; SA2/SX2: 0~107.
3. Range of **m_2**: ES2/EX2: 0 ~ 255; SS2: 0~48; SA2/SX2: 0~499.
4. Range of **n**:.

Range of **n**	ES2/EX2	SS2	SA2/SX2
16-bit instruction	1~4	1~(49 - **m_2**)	1~(499 - **m_2**)
32-bit instruction	1~2	1~(49 - **m_2**)/2	1~(499 - **m_2**)/2

Program Example:

1. Read out the data in CR#29 of special module N0.0 to register D0 in PLC, and CR#30 of special module No.0 to register D1 in PLC. 2 consecutive 16-bit data are read at one time (**n** = 2).
2. When X0 = ON, the instruction executes; when X0 = OFF, the previous content in D0 and D1 won't be changed.



API	Mnemonic			Operands	Function	Controllers
79	D	TO	P	m1 m2 S n	Write CR data into Special Modules	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
m_1					*	*							*		
m_2					*	*							*		
S					*	*							*		
n					*	*							*		

Program Steps

TO, TOP: 9 steps

DTO, DTOP: 17 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

m_1: No. of special module **m_2**: CR# in special module to be written **S**: Data to be written in CR

n: Number of data to be written at a time

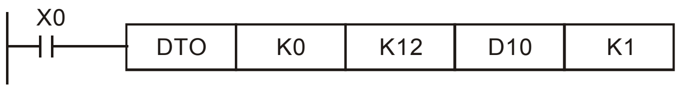
Explanations:

1. PLC uses this instruction to write data into CR (Control register) on special modules.
2. Setting range of **m_1**: ES2/EX2/SS2: 0 ~ 7; SA2/SX2: 0~107
3. Setting range of **m_2**: ES2/EX2: 0 ~ 255; SS2: 0~48; SA2/SX2: 0~499.
4. Setting range of **n**:.

Range of **n**	ES2/EX2	SS2	SA2/SX2
16-bit instruction	1~4	1~(49 - **m_2**)	1~(499 - **m_2**)
32-bit instruction	1~2	1~(49 - **m_2**)/2	1~(499 - **m_2**)/2

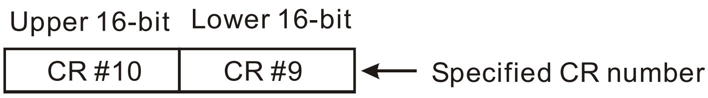
Program Example:

1. Use 32-bit instruction DTO to write the content in D11 and D10 into CR#13 and CR#12 of special module No.0. One 32-bit data is written at a time (**n** = 1)
2. When X0 = ON, the instruction executes; when X0 = OFF, the previous content in D10 and D11 won't be changed.

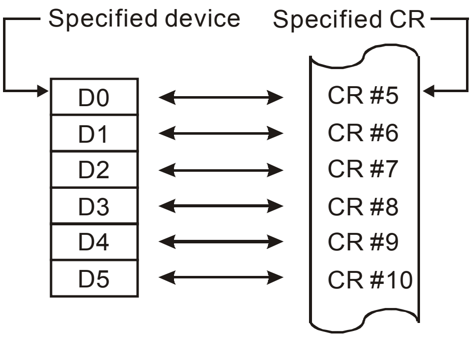


The rules for operand:

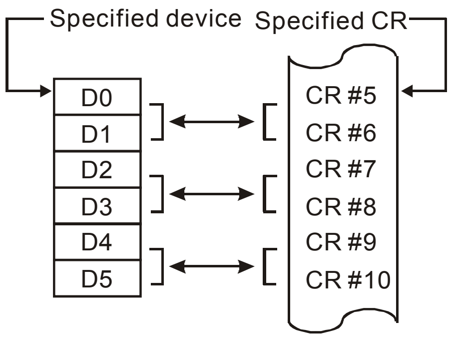
1. **m_1**: number of special module. The modules are numbered from 0 (closest to MPU) to 7 automatically by their distance from MPU. Maximum 8 modules are allowed to connect to MPU and will not occupy any digital I/O points
2. **m_2**: number of CR (Control Register). CR is the 16-bit memory built in the special module for control or monitor purpose, numbering in decimal. All operation status and settings of the special module are recorded in the CR.
3. FROM/TO instruction reads/writes 1 CR at a time. DFROM/DTO instruction reads/writes 2 CRs at a time.



4. **n**: Number of data to be written at a time. **n** = 2 in 16-bit instruction has the same operation results as **n** = 1 in 32-bit instruction.



16-bit instruction when n=6



32-bit instruction when n=3

API	Mnemonic	Operands	Function	Controllers
80	RS	S m D n	Serial Communication	ES2/EX2 SS2 SA2 SX2

Type / OP	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					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Start device for data to be sent **m**: Length of data to be sent (**m** = 0~256) **D**: Start device for data to be received **n**: Length of data to be received (**n** = 0~255)

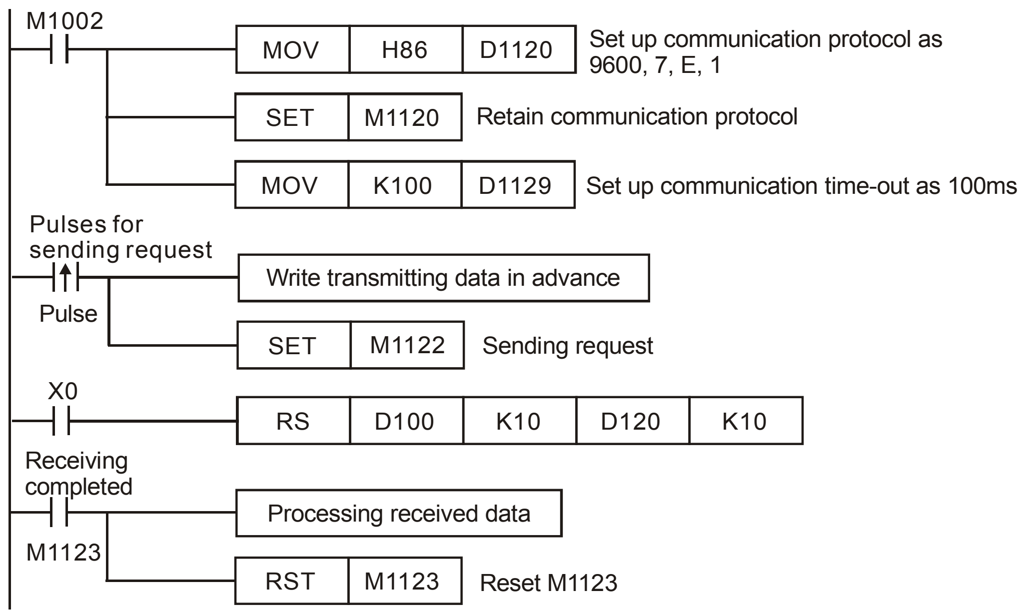
Explanations:

1. RS instruction is used for data transmitting and receiving between PLC and external/peripheral equipment (AC motor drive, etc.). Users have to pre-store word data in registers starting from **S**, set up data length **m**, specify the data receiving register **D** and the receiving data length **n**.
2. RS instruction supports communication on COM1 (RS-232), COM2 (RS-485) and COM3 (RS-485, ES2/EX2/SA2).
3. Designate **m** as K0 if data sending is not required. Designate **n** as K0 if data receiving is not required.
4. Modifying the communication data during the execution of RS instruction is invalid.
5. There is no limitation on times of using this instruction, however, only 1 instruction can be executed on one communication port at the same time..
6. If the communication format of the peripheral device is Modbus, DVP series PLC offers handy communication instructions MODRD, MODWR, and MODRW, to work with the device.
7. If the connected peripheral devices are Delta VFD series products, there are several communication instructions available including FWD, REV, STOP, RDST and RSTEF.

Program Example 1: COM2 RS-485

1. Write the data to be transmitted in advance into registers starting from D100 and set M1122 (Sending request) as ON.
2. When X10 = ON, RS instruction executes and PLC is ready for communication. D100 will then start to send out 10 data continuously. When data sending is over, M1122 will be automatically reset. (DO NOT apply RST M1122 in program). After approximate 1ms, PLC will start to receive 10 data and store the data in 10 consecutive registers starting from D120.
3. When data receiving is completed, M1123 will automatically be ON. When data processing on the received data is completed, M1123 has to be reset (OFF) and the PLC will be ready for communication again. However, DO NOT continuously execute RST M1123, i.e. it is

suggested to connect the RST M1123 instruction after the drive contact M1123.



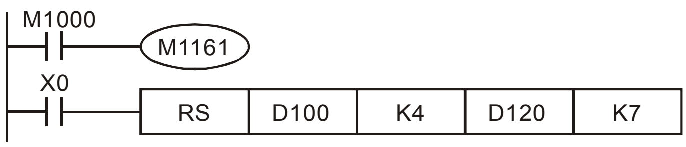
3

Program Example 2: COM2 RS-485

Switching between 8-bit mode (M1161 = ON) and 16-bit mode (M1161 = OFF)

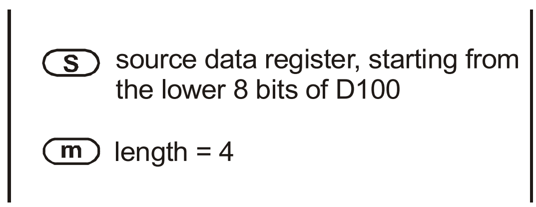
8-bit mode:

1. STX (Start of Text) and ETX (End of text) are set up by M1126 and M1130 together with D1124~D1126. When PLC executed RS instruction, STX and ETX will be sent out automatically.
2. When M1161 = ON, only the low byte (lower 8 bits) is valid for data communication, i.e. high byte will be ignored and low byte will be received and transmitted.

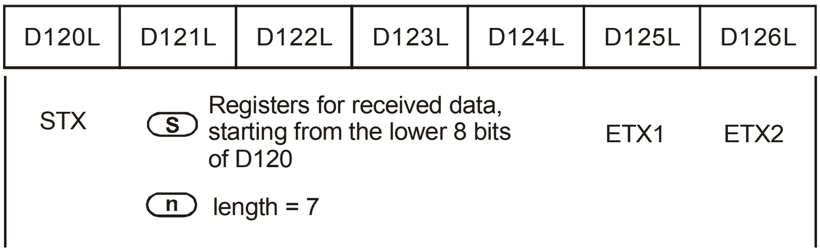


Sending data: (PLC -> external equipment)

STX	D100L	D101L	D102L	D103L	ETX1	ETX2



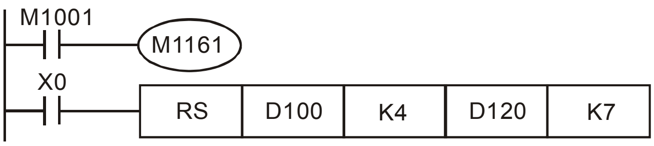
Receiving data: (External equipment -> PLC)



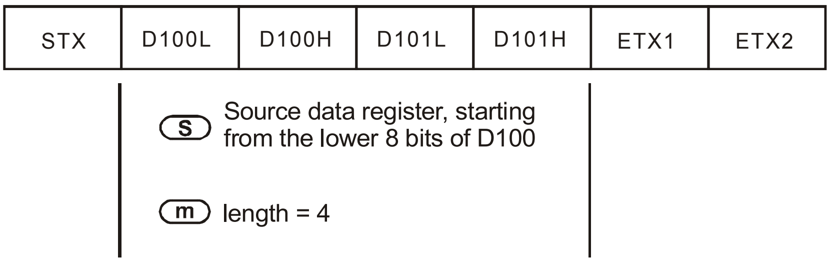
3. The STX and ETX of external equipments will be received by PLC in data receiving process, therefore, care should be taken on the setting of operand **n** (Length of data to be received).

16-bit mode:

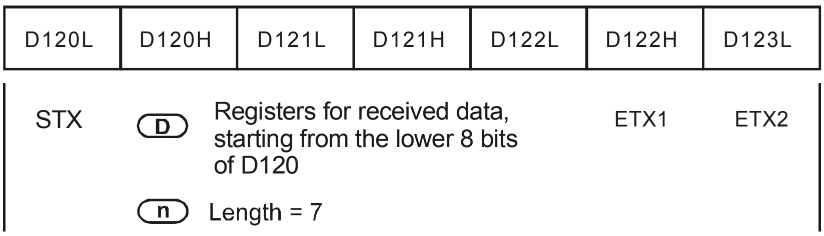
1. STX (Start of Text) and ETX (End of text) are set up by M1126 and M1130 together with D1124~D1126. When PLC executed RS instruction, STX and ETX will be sent out automatically.
2. When M1161 = OFF, the 16-bit mode is selected, i.e. both high byte and low byte of the 16-bit data will be received and transmitted.



Sending data: (PLC -> external equipment)



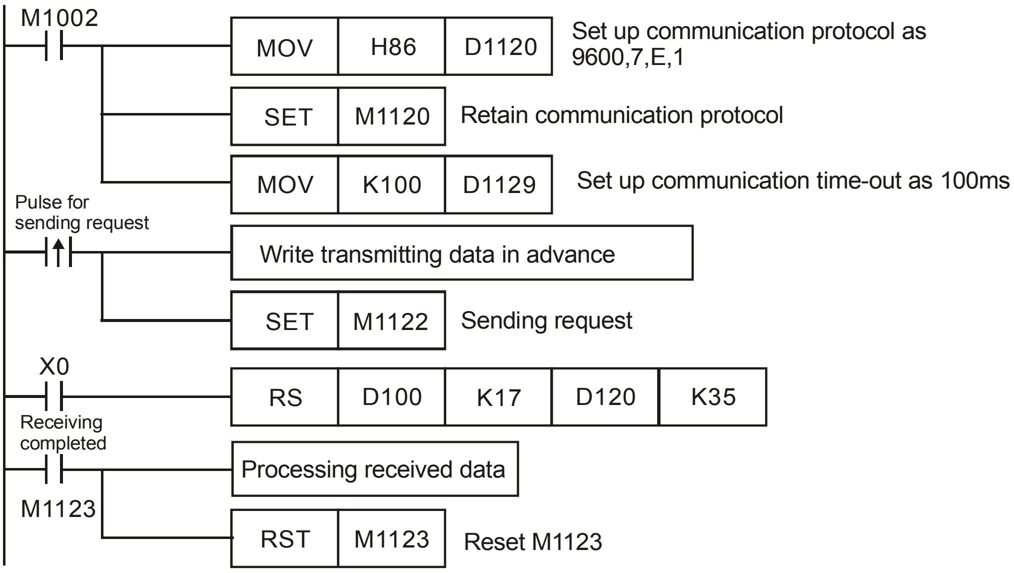
Receiving data: (External equipment -> PLC)



3. The STX and ETX of external equipments will be received by PLC in data receiving process, therefore, care should be taken on the setting of operand **n** (Length of data to be received)

Program Example 3: COM2 RS-485

1. Connect PLC to VFD-B series AC motor drives (AC motor drive in ASCII Mode; PLC in 16-bit mode and M1161 = OFF).
2. Write the data to be sent into registers starting from D100 in advance in order to read 6 data starting from address H2101 on VFD-B



PLC ⇨ VFD-B, PLC sends **“: 01 03 2101 0006 D4 CR LF “**

VFD-B ⇨ PLC, PLC receives **“: 01 03 0C 0100 1766 0000 0000 0136 0000 3B CR LF “**

Registers for sent data (PLC sends out messages)

Register	Data		Explanation	
D100 low	‘: ’	3A H	STX	
D100 high	‘0’	30 H	ADR 1	Address of AC motor drive: ADR (1,0)
D101 low	‘1’	31 H	ADR 0	
D101 high	‘0’	30 H	CMD 1	Instruction code: CMD (1,0)
D102 low	‘3’	33 H	CMD 0	
D102 high	‘2’	32 H	Start data address	
D103 low	‘1’	31 H		
D103 high	‘0’	30 H		
D104 low	‘1’	31 H		
D104 high	‘0’	30 H	Number of data (counted by words)	
D105 low	‘0’	30 H		
D105 high	‘0’	30 H		
D106 low	‘6’	36 H		
D106 high	‘D’	44 H	LRC CHK 1	Error checksum: LRC CHK (0,1)
D107 low	‘4’	34 H	LRC CHK 0	
D107 high	CR	D H	END	
D108 low	LF	A H		

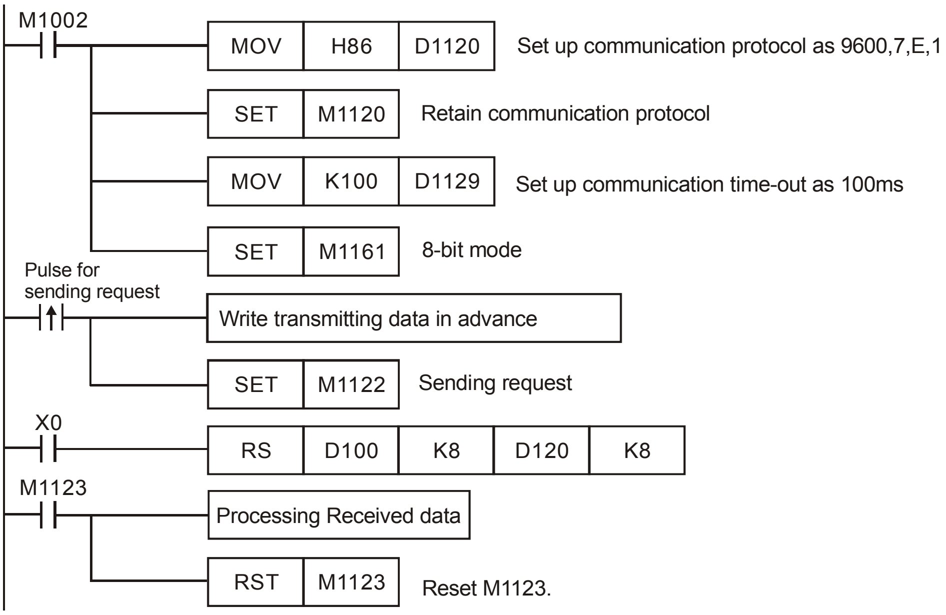
Registers for received data (VFD-B responds with messages)

Register	Data		Explanation
D120 low	‘: ’	3A H	STX
D120 high	‘0’	30 H	ADR 1
D121 low	‘1’	31 H	ADR 0
D121 high	‘0’	30 H	CMD 1
D122 low	‘3’	33 H	CMD 0
D122 high	‘0’	30 H	Number of data (counted by byte)
D123 low	‘C’	43 H	
D123 high	‘0’	30 H	Content of address 2101 H
D124 low	‘1’	31 H	
D124 high	‘0’	30 H	
D125 low	‘0’	30 H	
D125 high	‘1’	31 H	Content of address 2102 H
D126 low	‘7’	37 H	
D126 high	‘6’	36 H	
D127 low	‘6’	36 H	
D127 high	‘0’	30 H	Content of address 2103 H
D128 low	‘0’	30 H	
D128 high	‘0’	30 H	
D129 low	‘0’	30 H	
D129 high	‘0’	30 H	Content of address 2104 H
D130 low	‘0’	30 H	
D130 high	‘0’	30 H	
D131 low	‘0’	30 H	
D131 high	‘0’	30 H	Content of address 2105 H
D132 low	‘1’	31 H	
D132 high	‘3’	33 H	
D133 low	‘6’	36 H	
D133 high	‘0’	30 H	Content of address 2106 H
D134 low	‘0’	30 H	
D134 high	‘0’	30 H	
D135 low	‘0’	30 H	
D135 high	‘3’	33 H	LRC CHK 1
D136 low	‘B’	42 H	LRC CHK 0
D136 high	CR	D H	END
D137 low	LF	A H	

3. The status of Delta VFD series inverters can also be accessed by handy instruction API 105 RDST instruction through COM2/COM3 on PLC.

Program Example 4: COM2 RS-485

1. Connect PLC to VFD-B series AC motor drives (AC motor drive in RTU Mode; PLC in 16-bit mode and M1161 = ON).
2. Write the data to be sent into registers starting from D100 in advance. Write H12 (Forward running) into H2000 (VFD-B parameter address).



PLC ⇨ VFD-B, PLC sends: **01 06 2000 0012 02 07**

VFD-B ⇨ PLC, PLC receives: **01 06 2000 0012 02 07**

Registers for sent data (PLC sends out messages)

Register	Data	Explanation
D100 low	01 H	Address
D101 low	06 H	Function
D102 low	20 H	Data address
D103 low	00 H	
D104 low	00 H	Data content
D105 low	12 H	
D106 low	02 H	CRC CHK Low
D107 low	07 H	CRC CHK High

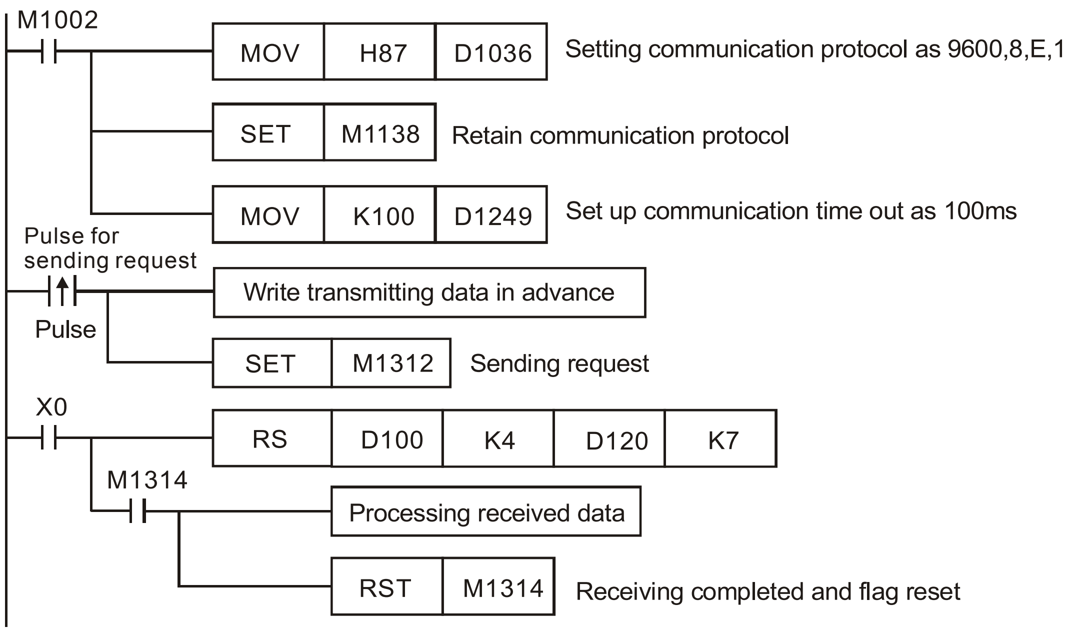
Registers for received data (VFD-B responds with messages)

Register	Data	Explanation
D120 low	01 H	Address
D121 low	06 H	Function
D122 low	20 H	Data address
D123 low	00 H	
D124 low	00 H	Data content
D125 low	12 H	
D126 low	02 H	CRC CHK Low
D127 low	07 H	CRC CHK High

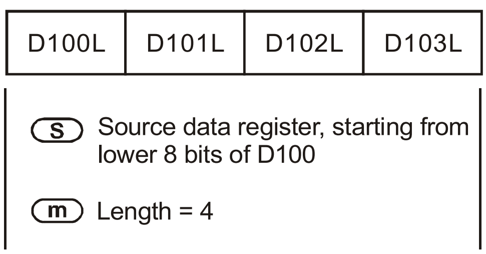
3. The forward running function of Delta's VFD series inverter can also be set by handy instruction API 102 FWD instruction through COM2/COM3 on PLC.

Program Example 5: COM1 RS-232

1. Only 8-bit mode is supported. Communication format and speed are specified by lower 8 bits of D1036.
2. STX/ETX setting function (M1126/M1130/D1124~D1126) is not supported.
3. High byte of 16-bit data is not available. Only low byte is valid for data communication.
4. Write the data to be transmitted in advance into registers starting from D100 and set M1312 (COM1 sending request) as ON
5. When X10 = ON, RS instruction executes and PLC is ready for communication. D0 will then start to send out 4 data continuously. When data sending is over, M1312 will be automatically reset. (DO NOT apply RST M1312 in program). After approximate 1ms, PLC will start to receive 7 data and store the data in 7 consecutive registers starting from D20.
6. When data receiving is completed, M1314 will automatically be ON. When data processing on the received data is completed, M1314 has to be reset (OFF) and the PLC will be ready for communication again. However, DO NOT continuously execute RST M1314, i.e. it is suggested to connect the RST M1314 instruction after the drive contact M1314



Sending data: (PLC→External equipment)



Receving data: (External equipment→PLC)

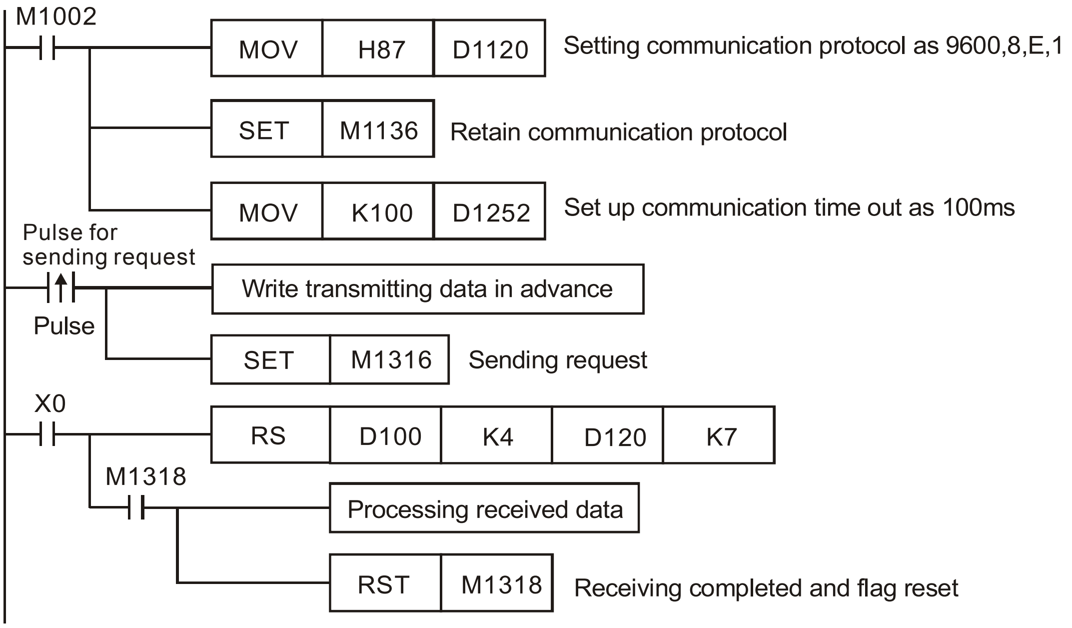
D120L	D121L	D122L	D123L	D124L	D125L	D126L

S Registers for received data, starting from lower 8 bits of D120

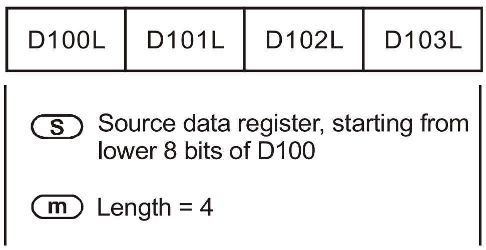
n Length = 7

Program Example 6: COM3 RS-485

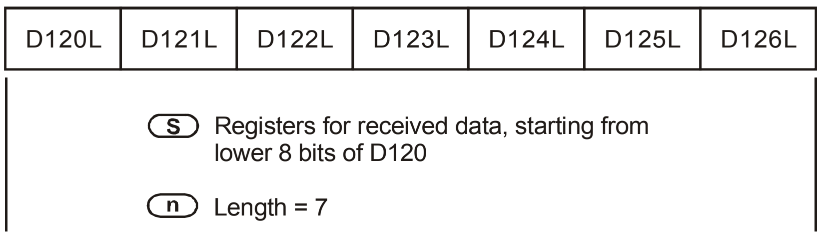
1. Only 8-bit mode is supported. Communication format and speed are specified by lower 8 bits of D1109.
2. STX/ETX setting function (M1126/M1130/D1124~D1126) is not supported.
3. High byte of 16-bit data is not available. Only low byte is valid for data communication.
4. Write the data to be transmitted in advance into registers starting from D100 and set M1316 (COM3 sending request) as ON
5. When X10 = ON, RS instruction executes and PLC is ready for communication. D0 will then start to send out 4 data continuously. When data sending is over, M1318 will be automatically reset. (DO NOT apply RST M1318 in program). After approximate 1ms, PLC will start to receive 7 data and store the data in 7 consecutive registers starting from D20.
6. When data receiving is completed, M1318 will automatically be ON. When data processing on the received data is completed, M1318 has to be reset (OFF) and the PLC will be ready for communication again. However, DO NOT continuously execute RST M1318, i.e. it is suggested to connect the RST M1318 instruction after the drive contact M1318.



Sending data: (PLC→External equipment)



Receving data: (External equipment→PLC)



Points to note:

1. **PLC COM1 RS-232:** Associated flags (Auxiliary relays) and special registers (Special D) for communication instructions RS / MODRD

Flag	Function	Action
M1138	COM1 retain communication settings. Communication settings will be reset (changed) according to the content in D1036 after every scan cycle. Users can set ON M1138 if the communication protocol requires to be retained. When M1138 = ON, communication settings will not be reset (changed) when communication instructions are being processed, even if the content in D1036 is changed. Supported communication instructions: RS / MODRW	User sets and resets
M1139	COM1 ASCII / RTU mode selection, ON: RTU mode, OFF: ASCII mode. Supported communication instructions: RS / MODRW	User sets and resets
M1312	COM1 sending request. Before executing communication instructions, users need to set M1312 to ON by trigger pulse, so that the data sending and receiving will be started. When the communication is completed, PLC will reset M1312 automatically. Supported communication instructions: RS / MODRW	User sets and system resets
M1313	COM1 data receiving ready. When M1313 is ON, PLC is ready for data receiving Supported communication instructions: RS / MODRW	System

Flag	Function	Action
M1314	COM1 Data receiving completed. When data receiving of communication instructions is completed, M1314 will be ON. Users can process the received data when M1314 is ON. When data processing is completed, M1314 has to be reset by users. Supported communication instructions: RS / MODRW	System sets and user resets
M1315	COM1 receiving error. M1315 will be set ON when errors occur and the error code will be stored in D1250. Supported communication instructions: RS / MODRW	System sets and user resets

Special register	Function
D1036	COM1 (RS-232) communication protocol. Refer to the following table in point 4 for protocol setting.
D1167	The specific end word to be detected for RS instruction to execute an interruption request (I140) on COM1 (RS-232). Supported communication instructions: RS
D1121	COM1 (RS-232) and COM2 (RS-485) communication address.
D1249	COM1 (RS-232) Communication time-out setting (unit: ms). If users set up time-out value in D1249 and the data receiving time exceeds the time-out value, M1315 will be set ON and the error code K1 will be stored in D1250. M1315 has to be reset manually when time-out status is cleared.
D1250	COM1 (RS-232) communication error code. Supported communication instructions: MODRW

2. **PLC COM2 RS-485:** Associated flags (Auxiliary relays) and special registers (Special D) for communication instructions RS / MODRD / MODWR / FWD / REV / STOP / RDST / RSTEF / MODRW.

Flag	Function	Action
M1120	Retain communication settings. Communication settings will be reset (changed) according to the content in D1120 after every scan cycle. Users can set ON M1120 if the communication protocol requires to be retained. When M1120 = ON, communication settings will not be reset (changed) when communication instructions are being processed, even if the content in D1120 is changed.	User sets/resets

Flag	Function	Action
M1121	Data transmission ready. M1121 = OFF indicates that RS-485 in COM2 is transmitting	System sets
M1122	Sending request. Before executing communication instructions, users need to set M1122 to ON by trigger pulse, so that the data sending and receiving will be started. When the communication is completed, PLC will reset M1122 automatically.	User sets, system resets
M1123	Data receiving completed. When data receiving of communication instructions is completed, M1123 will be ON. Users can process the received data when M1123 is ON. When data processing is completed, M1123 has to be reset by users. Supported communication instructions: RS	System sets ON and user resets
M1124	Data receiving ready. When M1124 is ON, PLC is ready for data receiving..	System sets
M1125	Communication ready status reset. When M1125 is set ON, PLC resets the communication (transmitting/receiving) ready status. M1125 has to be reset by users after resetting the communication ready status.	User sets/resets
M1126	Set STX/ETX as user-defined or system-defined in RS communication. For details please refer to the table in point 5. M1126 only supports RS instruction.	
M1130	Set STX/ETX as user-defined or system-defined in RS communication. For details please refer to the table in point 5. M1130 only supports RS instruction	
M1127	COM2 (RS-485) data sending/receiving/converting completed. RS instruction is NOT supported. Supported communication instructions: MODRD / MODWR / FWD / REV / STOP / RDST / RSTEF / MODRW	System sets and user resets
M1128	Transmitting/receiving status indication.	System sets
M1129	Receiving time out. If users set up time-out value in D1129 and the data receiving time exceeds the time-out value, M1129 will be set ON.	System sets and user resets

<table>
<tr><th>Flag</th><th>Function</th><th>Action</th></tr>
<tr><td>M1131</td><td>In ASCII mode, M1131 = ON only when MODRD/RDST/MODRW data is being converted to HEX.
Supported communication instructions:
MODRD / RDST / MODRW</td><td rowspan="4">System sets</td></tr>
<tr><td>M1140</td><td>MODRD/MODWR/MODRW data receiving error
Supported communication instructions:
MODRD / MODWR / MODRW</td></tr>
<tr><td>M1141</td><td>MODRD/MODWR/MODRW parameter error
Supported communication instructions:
MODRD / MODWR/ MODRW</td></tr>
<tr><td>M1142</td><td>Data receiving error of VFD-A handy instructions.
Supported communication instructions:
FWD / REV / STOP / RDST / RSTEF</td></tr>
<tr><td>M1143</td><td>ASCII / RTU mode selection. ON : RTU mode, OFF: ASCII mode.
Supported communication instructions:
RS / MODRD / MODWR / MODRW (When M1177 = ON, FWD / REV / STOP / RDST / RSTEF can also be applied.</td><td>User sets and resets</td></tr>
<tr><td>M1161</td><td>8/16-bit mode. ON: 8-bit mode. OFF: 16-bit mode
Supported communication instructions: RS</td><td rowspan="2">User sets</td></tr>
<tr><td>M1177</td><td>Enable the communication instruction for Delta VFD series inverter. ON: VFD-A (Default), OFF: other models of VFD
Supported communication instructions:
FWD / REV / STOP / RDST / RSTEF</td></tr>
</table>

Special register	Function
D1038	Delay time of data response when PLC is SLAVE in COM2, COM3 RS-485 communication, Range: 0~10,000. (Unit: 0.1ms). By using EASY PLC LINK in COM2, D1038 can be set to send next communication data with delay. (unit: one scan cycle)
D1050~D1055	Converted data for Modbus communication data processing. PLC automatically converts the ASCII data in D1070~D1085 into Hex data and stores the 16-bit Hex data into D1050~D1055 Supported communication instructions: MODRD / RDST

Special register	Function
D1070~D1085	Feedback data (ASCII) of Modbus communication. When PLC's RS-485 communication instruction receives feedback signals, the data will be saved in the registers D1070~D1085 and then converted into Hex in other registers. RS instruction is not supported.
D1089~D1099	Sent data of Modbus communication. When PLC's RS-485 communication instruction (MODRD) sends out data, the data will be stored in D1089~D1099. Users can check the sent data in these registers. RS instruction is not supported
D1120	COM2 (RS-485) communication protocol. Refer to the following table in point 4 for protocol setting.
D1121	COM1 (RS-232) and COM2 (RS-485) PLC communication address when PLC is slave.
D1122	COM2 (RS-485) Residual number of words of transmitting data.
D1123	COM2 (RS-485) Residual number of words of the receiving data.
D1124	COM2 (RS-485) Definition of start character (STX) Refer to the following table in point 3 for the setting. Supported communication instruction: RS
D1125	COM2 (RS-485) Definition of first ending character (ETX1) Refer to the following table in point 3 for the setting. Supported communication instruction: RS
D1126	COM2 (RS-485) Definition of second ending character (ETX2) Refer to the following table in point 3 for the setting. Supported communication instruction: RS
D1129	COM2 (RS-485) Communication time-out setting (unit: ms). If users set up time-out value in D1129 and the data receiving time exceeds the time-out value, M1129 will be set ON and the error code K1 will be stored in D1130. M1129 has to be reset manually when time-out status is cleared.
D1130	COM2 (RS-485) Error code returning from Modbus. RS instruction is not included. Supported communication instructions: MODRD / MODWR / FWD / REV / STOP / RDST / RSTEF / MODRW

Special register	Function
D1168	The specific end word to be detected for RS instruction to execute an interruption request (I150) on COM2 (RS-485). <u>Supported communication instruction:</u> RS
D1256~D1295	For COM2 RS-485 MODRW instruction. D1256~D1295 store the sent data of MODRW instruction. When MODRW instruction sends out data, the data will be stored in D1256~D1295. Users can check the sent data in these registers. <u>Supported communication instruction:</u> MODRW
D1296~D1311	For COM2 RS-485 MODRW instruction. D1296~D1311 store the converted hex data from D1070 ~ D1085 (ASCII). PLC automatically converts the received ASCII data in D1070 ~ D1085 into hex data. <u>Supported communication instruction:</u> MODRW

3. **PLC COM3 RS-485:** Associated flags (Auxiliary relays) and special registers (Special D) for communication instructions RS / MODRW and FWD / REV / STOP / RDST / RSTEF when M1177 = ON.

Flag	Function	Action
M1136	COM3 retain communication settings. Communication settings will be reset (changed) according to the content in D1109 after every scan cycle. Users can set ON M1136 if the communication protocol requires to be retained. When M1136 = ON, communication settings will not be reset (changed) when communication instructions are being processed, even if the content in D1109 is changed	User sets and resets
M1320	COM3 ASCII / RTU mode selection. ON : RTU mode, OFF: ASCII mode.	User sets and resets
M1316	COM3 sending request. Before executing communication instructions, users need to set M1316 to ON by trigger pulse, so that the data sending and receiving will be started. When the communication is completed, PLC will reset M1316 automatically.	User sets, system resets
M1317	Data receiving ready. When M1317 is ON, PLC is ready for data receiving.	System sets
M1318	COM3 data receiving completed.	System sets, user resets

Flag	Function	Action
M1319	COM3 data receiving error. M1319 will be set ON when errors occur and the error code will be stored in D1252	System sets, user resets

Special register	Function
D1038	Delay time of data response when PLC is SLAVE in COM2, COM3 RS-485 communication, Range: 0~10,000. (unit: 0.1ms). By using EASY PLC LINK in COM2, D1038 can be set to send next communication data with delay. (unit: one scan cycle)
D1109	COM3 (RS-485) communication protocol. Refer to the following table in point 4 for protocol setting.
D1169	The specific end word to be detected for RS instruction to execute an interruption request (I160) on COM3 (RS-485). Supported communication instructions: RS
D1252	COM3 (RS-485) Communication time-out setting (ms). If users set up time-out value in D1252 and the data receiving time exceeds the time-out value, M1319 will be set ON and the error code K1 will be stored in D1253. M1319 has to be reset manually when time-out status is cleared.
D1253	COM3 (RS-485) communication error code
D1255	COM3 (RS-485) PLC communication address when PLC is Slave.

4. Corresponding table between COM ports and communication settings/status.

	COM1	COM2	COM3	Function Description
Protocol setting	M1138	M1120	M1136	Retain communication setting
	M1139	M1143	M1320	ASCII/RTU mode selection
	D1036	D1120	D1109	Communication protocol
	D1121	D1121	D1255	PLC communication address
Sending request	-	M1161	-	8/16 bit mode selection
	-	M1121	-	Indicate transmission status
	M1312	M1122	M1316	Sending request
	-	M1126	-	Set STX/ETX as user/system defined. (RS)
	-	M1130	-	Set STX/ETX as user/system defined. (RS)
	-	D1124	-	Definition of STX (RS)

	COM1	COM2	COM3	Function Description
	-	D1125	-	Definition of ETX1 (RS)
	-	D1126	-	Definition of ETX2 (RS)
	D1249	D1129	D1252	Communication timeout setting (ms)
	-	D1122	-	Residual number of words of transmitting data
	-	D1256 ~ D1295	-	Store the sent data of MODRW instruction.
	-	D1089 ~ D1099	-	Store the sent data of MODRD / MODWR / FWD / REV / STOP / RDST / RSTEF instruction
Data receiving	M1313	M1124	M1317	Data receiving ready
	-	M1125	-	Communication ready status reset
	-	M1128	-	Transmitting/Receiving status Indication
	-	D1123	-	Residual number of words of the receiving data
	-	D1070 ~ D1085	-	Store the feedback data of Modbus communication. RS instruction is not supported.
	D1167	D1168	D1169	Store the specific end word to be detected for executing interrupts I140/I150/I160 (RS)
Receiving completed	M1314	M1123	M1318	Data receiving completed
	-	M1127	-	COM2 (RS-485) data sending / receiving / converting completed. (RS instruction is not supported)
	-	M1131	-	ON when MODRD/RDST/MODRW data is being converted from ASCII to Hex
	-	D1296 ~ D1311	-	Store the converted HEX data of MODRW instruction.
	-	D1050 ~ D1055	-	Store the converted HEX data of MODRD instruction
Errors	M1315	-	M1319	Data receiving error
	D1250	-	D1253	Communication error code
	-	M1129	-	COM2 (RS-485) receiving time out
	-	M1140	-	COM2 (RS-485) MODRD/MODWR/MODRW data receiving error

	COM1	COM2	COM3	Function Description
	-	M1141	-	MODRD/MODWR/MODRW parameter error (Exception Code exists in received data) Exception Code is stored in D1130
	-	M1142	-	Data receiving error of VFD-A handy instructions (FWD/REV/STOP/RDST/RSTEF)
	-	D1130	-	COM2 (RS-485) Error code returning from Modbus communication

5. Communication protocol settings: D1036(COM1 RS-232) / D1120(COM2 RS-485) / D1109(COM3 RS-485)

	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 bps 0010(H2): 150 bps 0011(H3): 300 bps 0100(H4): 600 bps 0101(H5): 1200 bps 0110(H6): 2400 bps 0111(H7): 4800 bps 1000(H8): 9600 bps 1001(H9): 19200 bps 1010(HA): 38400 bps 1011(HB): 57600 bps 1100(HC): 115200 bps 1101(HD): 500000 bps (COM2 / COM3) 1110 (HE): 31250 bps (COM2 / COM3) 1111 (HF): 921000 bps (COM2 / COM3)	
b8 (D1120)	STX	0: None	1: D1124
b9 (D1120)	ETX1	0: None	1: D1125
b10 (D1120)	ETX2	0: None	1: D1126
b11~b15	N/A		

6. When RS instruction is applied for communication between PLC and peripheral devices on COM2 RS-485, usually STX (Start of the text) and ETX (End of the text) have to be set into communication format. In this case, b8~10 of D1120 should be set to 1, so that users can set up STX/ETX as user-defined or system-defined by using M1126, M1130, and D1124~D1126. For settings of M1126 and M1130, please refer to the following table.

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="2">M1130</th></tr>
<tr><th>0</th><th>1</th></tr>
<tr><td rowspan="2">M1126</td><td>0</td><td>D1124: user defined
D1125: user defined
D1126: user defined</td><td>D1124: H 0002
D1125: H 0003
D1126: H 0000（no setting）</td></tr>
<tr><td>1</td><td>D1124: user defined
D1125: user defined
D1126: user defined</td><td>D1124: H 003A（’:’）
D1125: H 000D（CR）
D1126: H 000A（LF）</td></tr>
</table>

7. Example of setting communication format in D1120:

Communication format:

Baud rate: 9600, 7, N, 2

STX : “: “

ETX1 : “CR”

ETX2 : “LF”

Check to the table in point 4 and the set value H788 can be referenced corresponding to the baud rate. Set the value into D1120.

	b15															b0
D1120	0	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0
	0 N/A				7				8				8			

```
M1002
─┤ ├───────[ MOV | H788 | D1120 ]
```

When STX, ETX1 and ETX2 are applied, care should be taken on setting the ON/OFF status of M1126 and M1130.

8. D1250(COM1)、D1253(COM3) communication error code:

Value	Error Description
H0001	Communication time-out
H0002	Checksum error
H0003	Exception Code exists
H0004	Command code error / data error

Value	Error Description
H0005	Communication data length error

9. Corresponding table between D1167~D1169 and the associated interrupt pointers. (Only lower 8 bits are valid)

COM Port	I1□0 interrupt	Special D
COM1	I140	D1167
COM2	I150	D1168
COM3	I160	D1169

10. Take standard MODBUS format for example:

ASCII mode

Field Name	Descriptions
STX	Start word = ': ' (3AH)
Address Hi	Communication address: The 8-bit address consists of 2 ASCII codes
Address Lo	
Function Hi	Function code: The 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	

The communication protocol is in Modbus ASCII mode, i.e. every byte is composed of 2 ASCII characters. For example, 64Hex is '64' in ASCII, composed by '6' (36Hex) and '4' (34Hex). Every character '0'…'9', 'A'…'F' corresponds to an ASCII code.

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)

Address:

'0' '0': Broadcasting to all drives (Broadcast)

'0' '1': toward the drive at address 01

‘0’ ‘F’: toward the drive at address 15

‘1’ ‘0’: toward the drive at address 16

… and so on, max. address: 254 (‘FE’)

Function code:

‘0’ ‘3’: read contents from multiple registers

‘0’ ‘6’: write one word into a single register

‘1’ ‘0’: write contents to multiple registers

Data characters:

The data sent by the user

LRC checksum:

LCR checksum is 2’s complement of the value added from Address to Data Characters.

For example: 01H + 03H + 21H + 02H + 00H + 02H = 29H. 2’s complement of 29H = D7H.

End word (END):

Fix the END as END Hi = CR (0DH), END Lo = LF (0AH)

3

Example:

Read 2 continuous data stored in the registers of the drive at address 01H (see the table below). The start register is at address 2102H.

Inquiry message:

STX	‘: ’
Address	‘0’
	‘1’
Function code	‘0’
	‘3’
Start address	‘2’
	‘1’
	‘0’
	‘2’
Number of data (count by word)	‘0’
	‘0’
	‘0’
	‘2’
LRC Checksum	‘D’
	‘7’
END	CR
	LF

Response message:

STX	‘: ’
Address	‘0’
	‘1’
Function code	‘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 Checksum	‘7’
	‘1’
END	CR
	LF

RTU mode

Field Name	Descriptions
START	Refer to the following explanation

Field Name	Descriptions
Address	Communication address: n 8-bit binary
Function	Function code: n 8-bit binary
DATA (n-1)	Data: n × 8-bit data
…….	
DATA 0	
CRC CHK Low	CRC checksum: 16-bit CRC consists of 2 8-bit binary data
CRC CHK High	
END	Refer to the following explanation

START/END:

RTU Timeout Timer:

Baud rate(bps)	RTU timeout timer (ms)	Baud rate (bps)	RTU timeout timer (ms)
300	40	9,600	2
600	21	19,200	1
1,200	10	38,400	1
2,400	5	57,600	1
4,800	3	115,200	1

Address:

00 H: Broadcasting to all drives (Broadcast)

01 H: toward the drive at address 01

0F H: toward the drive at address 15

10 H: toward the drive at address 16

… and so on, max. address: 254 (‘FE’)

Function code:

03 H: read contents from multiple registers

06 H: write one word into single register

10 H: write contents to multiple registers

Data characters:

The data sent by the user

CRC checksum: Starting from Address and ending at Data Content. The calculation is as follows:

Step 1: Set the 16-bit register (CRC register) = FFFFH

Step 2: Operate XOR on the first 8-bit message (Address) and the lower 8 bits of CRC register. Store the result in the CRC register.

Step 3: Right shift CRC register for a bit and fill “0” into the highest bit.

Step 4: Check the lowest bit (bit 0) of the shifted value. If bit 0 is 0, fill in the new value obtained at step 3 to CRC register; if bit 0 is NOT 0, operate XOR on A001H and the shifted value and store the result in the CRC register.

Step 5: Repeat step 3 – 4 to finish all operation on all the 8 bits.

Step 6: Repeat step 2 – 5 until the operation of all the messages are completed. The final value obtained in the CRC register is the CRC checksum. Care should be taken when placing the LOW byte and HIGH byte of the obtained CRC checksum.

Example:

Read 2 continuous data stored in the registers of the drive at address 01H (see the table below). The start register is at address 2102H

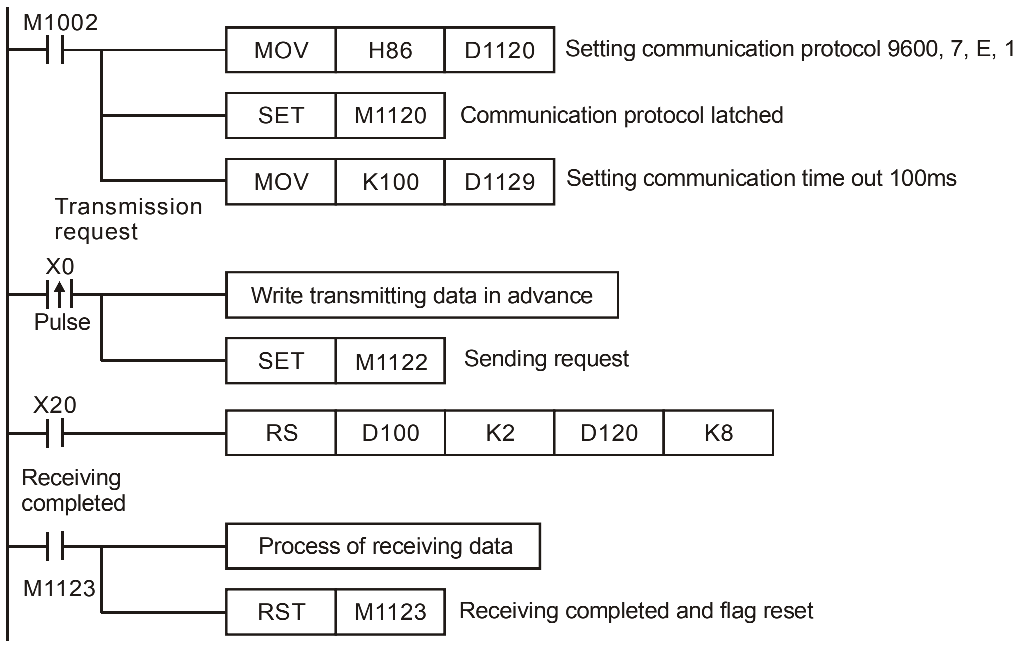
Inquiry message:

Field Name	Data (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

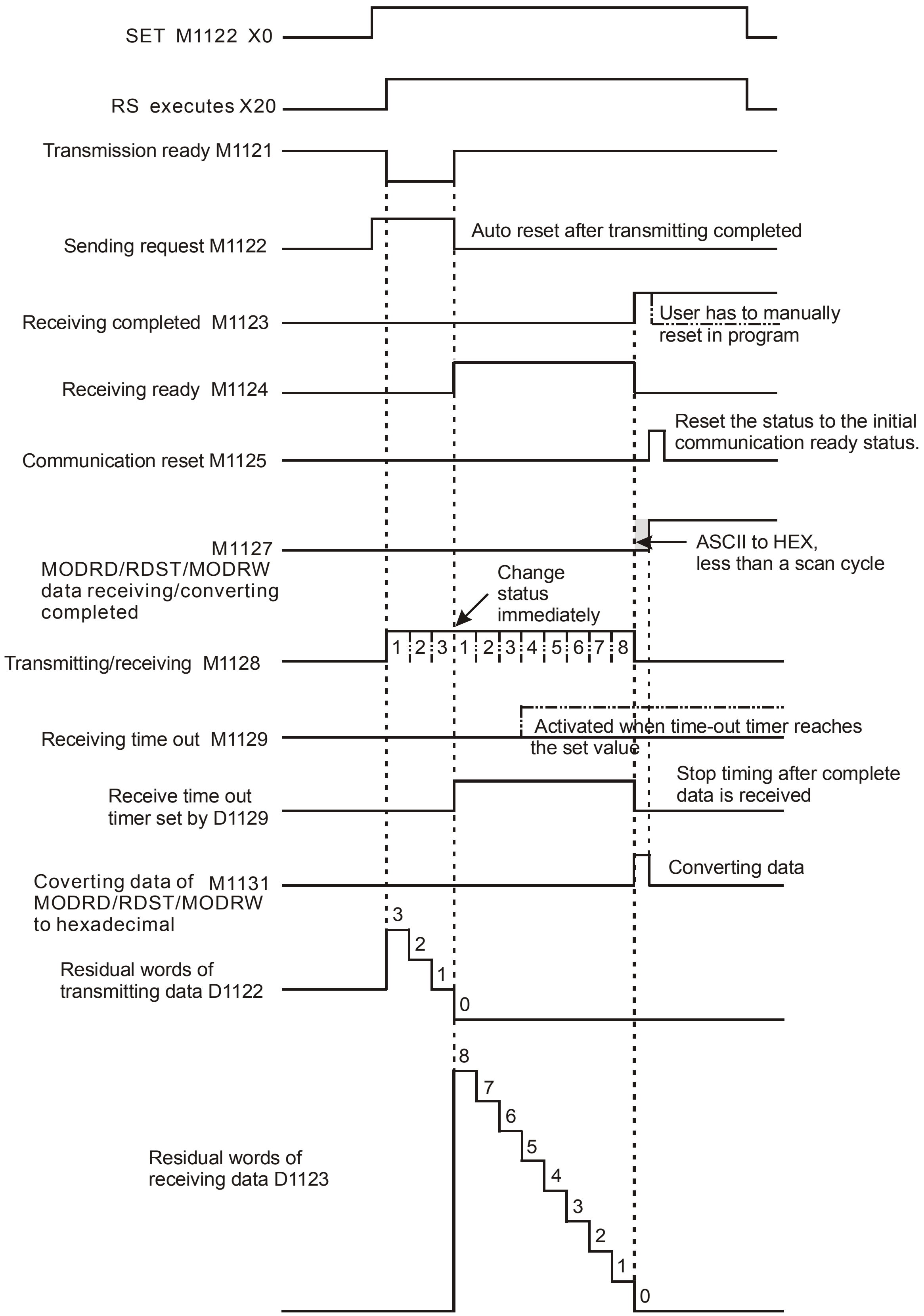
Response message:

Field Name	Data (Hex)
Address	01 H
Function	03 H
Number of data (count by byte)	04 H
Content of data address 2102H	17 H
	70 H
Content of data address 2103H	00 H
	00 H
CRC CHK Low	FE H
CRC CHK High	5C H

Example program of RS-485 communication:



Timing diagram:



API	Mnemonic			Operands	Function	Controllers
81	D	PRUN	P	S D	Parallel Run	ES2/EX2 SS2 SA2 SX2

Type / OP	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
S							*		*							DPRUN, DPRUNP: 9
D								*	*							steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

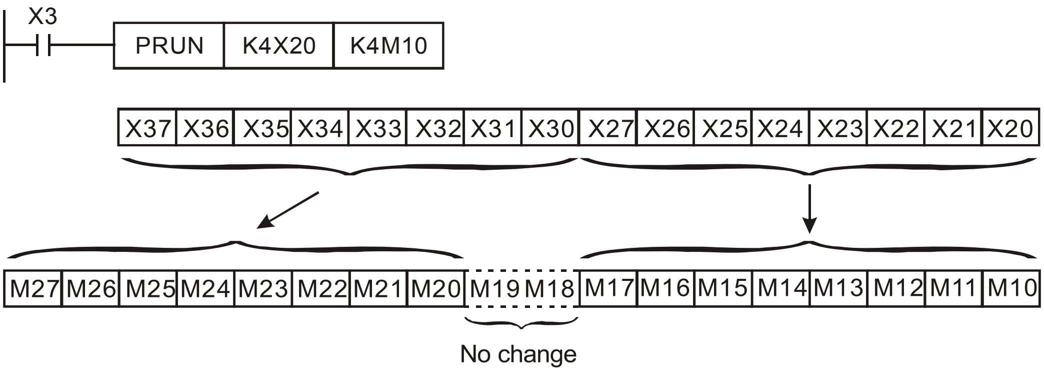
S: Source device **D**: Destination device

Explanations:

1. This instruction sends the content in **S** to **D** in the form of octal system
2. The start device of X, Y, M in KnX, KnY, KnM format 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.

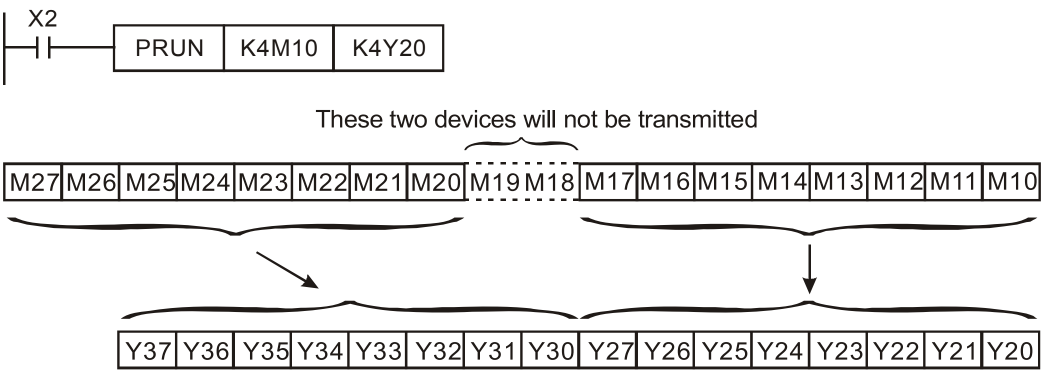
Program Example 1:

When X3 = ON, the content in K4X20 will be sent to K4M10 in octal form.



Program Example 2:

When X2 = ON, the content in K4M10 will be sent to K4Y20 in octal form.



API	Mnemonic		Operands	Function	Controllers
82	ASCI	P	S D n	Convert Hex to ASCII	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
																ASCI, ASCIP: 7 steps
S					*	*	*	*	*	*	*	*	*			
D								*	*	*	*	*	*			
n					*	*										

(Bit Devices: X, Y, M, S; Word devices: K, H, KnX, KnY, KnM, KnS, T, C, D, E, F)

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

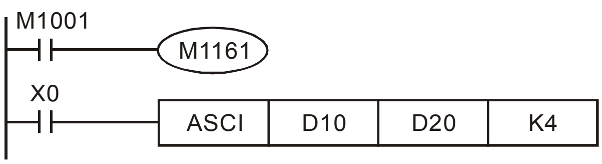
S: Source device **D**: Destination device **n**: Number of nibbles to be converted (**n** = 1~256)

Explanations:

1. 16-bit conversion mode: When M1161 = OFF, the instruction converts every nibble of the Hex data in **S** into ASCII codes and send them to the higher 8 bits and lower 8 bits of **D**. **n** = the converted number of nibbles.
2. 8-bit conversion mode: When M1161 = ON, the instruction converts every nibble of the Hex data in **S** into ASCII codes and send them to the lower 8 bits of **D**. **n** = the number of converted nibbles. (All higher 8 bits of **D** = 0).
3. Flag: M1161 (8/16 bit mode switch)
4. Available range for Hex data: 0~9, A~F

Program Example 1:

1. M1161 = OFF, 16-bit conversion.
2. When X0 = ON, convert the 4 hex values (nibbles) in D10 into ASCII codes and send the result to registers starting from D20.



3. Assume:

(D10) = 0123 H	'0' = 30H	'4' = 34H	'8' = 38H
(D11) = 4567 H	'1' = 31H	'5' = 35H	'9' = 39H
(D12) = 89AB H	'2' = 32H	'6' = 36H	'A' = 41H
(D13) = CDEF H	'3' = 33H	'7' = 37H	'B' = 42H

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

D10=0123 H

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

D20 high byte low byte

0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0
"1" → 31H								"0" → 30H							

D21 high byte low byte

0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0
"3" → 33H								"2" → 32H							

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

b15 D10 = H 0123 b0

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

b15 D11 = H 4567 b0

0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1
4				5				6				7			

Converted to

b15 D20 b0

0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	0
'7' → H 37								'6' → H 36							

b15 D21 b0

0	0	1	1	0	0	0	1	0	0	1	1	0	0	0	0
'1' → H 31								'0' → H 30							

b15 D22 b0

0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0
'3' → H 33								'2' → H 32							

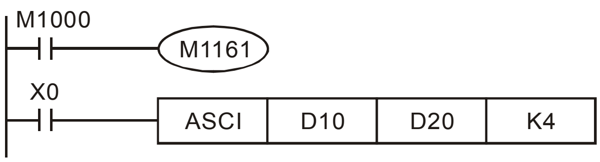
6. When **n** = 1 to 16:

n / D	K1	K2	K3	K4	K5	K6	K7	K8
D20 low byte	"3"	"2"	"1"	"0"	"7"	"6"	"5"	"4"
D20 high byte		"3"	"2"	"1"	"0"	"7"	"6"	"5"
D21 low byte			"3"	"2"	"1"	"0"	"7"	"6"
D21 high byte				"3"	"2"	"1"	"0"	"7"
D22 low byte					"3"	"2"	"1"	"0"
D22 high byte						"3"	"2"	"1"
D23 low byte							"3"	"2"
D23 high byte								"3"
D24 low byte								
D24 high byte				No change				
D25 low byte								
D25 high byte								
D26 low byte								
D26 high byte								
D27 low byte								
D27 high byte								

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

Program Example 2:

1. M1161 = ON, 8-bit conversion.
2. When X0 = ON, convert the 4 hex values (nibbles) in D10 into ASCII codes and send the result to registers starting from D20.



3. Assume:

(D10) = 0123 H	‘0’ = 30H	‘4’ = 34H	‘8’ = 38H
(D11) = 4567 H	‘1’ = 31H	‘5’ = 35H	‘9’ = 39H
(D12) = 89AB H	‘2’ = 32H	‘6’ = 36H	‘A’ = 41H
(D13) = CDEFH	‘3’ = 33H	‘7’ = 37H	‘B’ = 42H

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

D10=0123 H

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

ASCII code of "2" in D20 is 32H

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

ASCII code of "3" in D21 is 33H

0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
								3				3			

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

b15 D10 = H 0123 b0

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

Converted to

b15 D20 b0

0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
								'0' → H 30							

b15 D21 b0

0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
								'1' → H 31							

b15 D22 b0

0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
								'2' → H 32							

b15 D23 b0

0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
								'3' → H 33							

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

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

n / D	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			No change			“3”	“2”	“1”
D34							“3”	“2”
D35								“3”

API	Mnemonic			Operands	Function	Controllers
83		HEX	P	(S) (D) (n)	Convert ASCII to HEX	ES2/EX2 SS2 SA2 SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

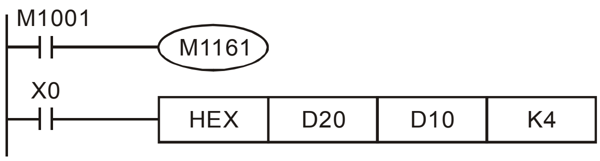
S: Source device **D**: Destination device **n**: number of bytes to be converted (**n** = 1~256)

Explanations:

1. 16-bit conversion mode: When M1161 = OFF, the instruction converts **n** bytes of ASCII codes starting from **S** into Hex data in byte mode and send them to high byte and low byte of **D**. **n** = the converted number of bytes.
2. 8-bit conversion mode: When M1161 = ON, the instruction converts **n** bytes (low bytes only) of ASCII codes starting from **S** into Hex data in byte mode and send them to the low byte of **D**. **n** = the converted number of bytes. (All higher 8 bits of **D** = 0)
3. Flag: M1161 (8/16 bit mode switch)
4. Available range for Hex data: 0~9, A~F

Program Example 1:

1. M1161 = OFF: 16-bit conversion.
2. When X0 = ON, convert 4 bytes of ASCII codes stored in registers D20~ D21 into Hex value and send the result in byte mode to register D10. **n** = 4



3. Assume:

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** = 4, the bit structure will be as:

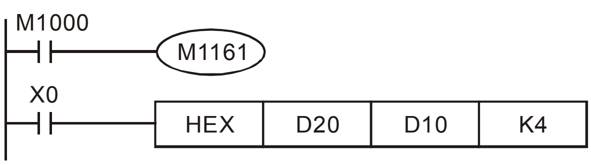
D20	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1
	44H → 'D'								43H → 'C'							
D21	0	1	0	0	0	1	1	0	0	1	0	0	0	1	0	1
	46H → 'F'								45H → 'E'							
D10	1	1	0	0	1	1	0	1	1	1	1	0	1	1	1	1
	C				D				E				F			

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

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

Program Example 2:

1. M1161 = ON: 8-bit conversion.



2. Assume:

S	ASCII code	HEX conversion	S	ASCII code	HEX conversion
D20	H 43	"C"	D25	H 39	"9"
D21	H 44	"D"	D26	H 41	"A"
D22	H 45	"E"	D27	H 42	"B"
D23	H 46	"F"	D28	H 34	"4"

S	ASCII code	HEX conversion	S	ASCII code	HEX conversion
D24	H 38	"8"	D29	H 35	"5"
D30	H 36	"6"	D33	H 31	"1"
D31	H 37	"7"	D34	H 32	"2"
D32	H 30	"0"	D35	H 33	"3"

3. When **n** is 2, the bit structure will be as

D20									0	1	0	0	0	0	1	1
									43H → 'C'							
D21									0	1	0	0	0	1	0	0
									44H → 'D'							
D10	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	1
									C				D			

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

<table>
<tr><th>D / n</th><th>D13</th><th>D12</th><th>D11</th><th>D10</th></tr>
<tr><td>1</td><td rowspan="12">The used registers which are not specified are all 0</td><td rowspan="8"></td><td rowspan="4"></td><td>***C H</td></tr>
<tr><td>2</td><td>**CD H</td></tr>
<tr><td>3</td><td>*CDE H</td></tr>
<tr><td>4</td><td>CDEF H</td></tr>
<tr><td>5</td><td>***C H</td><td>DEF8 H</td></tr>
<tr><td>6</td><td>**CD H</td><td>EF89 H</td></tr>
<tr><td>7</td><td>*CDE H</td><td>F89A H</td></tr>
<tr><td>8</td><td>CDEF H</td><td>89AB H</td></tr>
<tr><td>9</td><td>***C H</td><td>DEF8 H</td><td>9AB4 H</td></tr>
<tr><td>10</td><td>**CD H</td><td>EF89 H</td><td>AB45 H</td></tr>
<tr><td>11</td><td>*CDE H</td><td>F89A H</td><td>B456 H</td></tr>
<tr><td>12</td><td>CDEF H</td><td>89AB H</td><td>4567 H</td></tr>
<tr><td>13</td><td>***C H</td><td>DEF8 H</td><td>9AB4 H</td><td>5670 H</td></tr>
<tr><td>14</td><td>**CD H</td><td>EF89 H</td><td>AB45 H</td><td>6701 H</td></tr>
<tr><td>15</td><td>*CDE H</td><td>F89A H</td><td>B456 H</td><td>7012 H</td></tr>
<tr><td>16</td><td>CDEF H</td><td>89AB H</td><td>4567 H</td><td>0123 H</td></tr>
</table>

API	Mnemonic		Operands	Function	Controllers
84	CCD	P	S D n	Check Code	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S							*	*	*	*	*	*	*			CCD, CCDP: 7 steps
D									*	*	*	*	*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

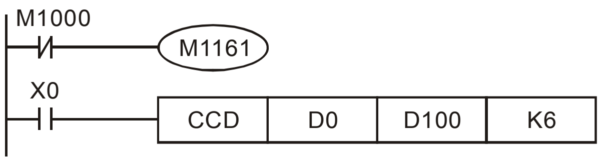
S: source data **D**: Destination device for storing check sum **n**: Number of byte (**n** = 1~256)

Explanations:

1. This instruction performs a sum check for ensuring the validity of the communication data.
2. 16-bit conversion: If M1161 = OFF, **n** bytes of data starting from low byte of **S** will be summed up, the checksum is stored in **D** and the parity bits are stored in **D+1**.
3. 8-bit conversion: If M1161 = ON, **n** bytes of data starting from low byte of **S** (only low byte is valid) will be summed up, the check sum is stored in **D** and the parity bits are stored in **D+1**.

Program Example 1:

1. M1161 = OFF, 16-bit conversion.
2. When X0 = ON, 6 bytes from low byte of D0 to high byte of D2 will be summed up, and the checksum is stored in D100 while the parity bits are stored in D101.



(S)	Content of data
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
D101	0 0 0 1 0 0 0 1

Total

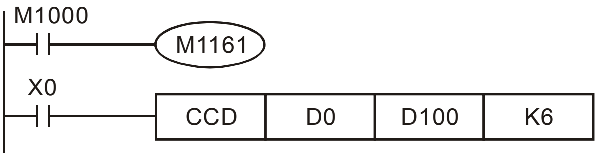
The parity is 1 when there is an odd number of 1.
The parity is 0 when there is an even number of 1.

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:

1. M1161 = ON, 8-bit conversion.
2. When X0 = ON, 6 bytes from low byte of D0 to low byte of D5 will be summed up, and the checksum is stored in D100 while the parity bits are stored in D101.



(S)	Content of data
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
D101	0 0 0 1 0 0 0 1

Total

The parity is 1 when there is a odd number of 1.
The parity is 0 when there is a even number of 1.

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			
					ES2/EX2	SS2	SA2	SX2
85	VRRD	P	(S) (D)	Volume Read				

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

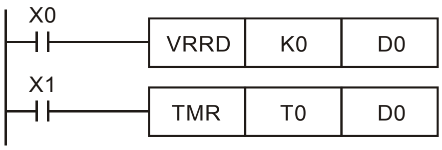
S: Variable resistor number (0~1) **D**: Destination device for storing read value

Explanations:

1. VRRD instruction is used to read the two variable resistors on PLC. The read value will be converted as 0 ~ 255 and stored in destination **D**.
2. If the VR volume is used as the set value of timer, the user only has to turn the VR knob and the set value of timer can be adjusted. When a value bigger than 255 is required, plus D with a certain constant.
3. Flags: M1178 and M1179. (See the Note)

Program Example:

1. When X0 = ON, the value of VR No.0 will be read out, converted into 8-bit BIN value (0~255), and stored in D0.
2. When X1 = ON, the timer which applies D0 as the set value will start timing.



Points to Note:

1. VR denotes Variable Resistor.
2. SX2 supports built-in 2 points of VR knobs which can be used with special D and M.

Device	Function
M1178	Enable knob VR0
M1179	Enable knob VR1
D1178	VR0 value
D1179	VR1 value

API	Mnemonic		Operands	Function	Controllers
86	VRSC	P	(S) (D)	Volume Scale Read	ES2/EX2 | SS2 | SA2 | SX2

Type OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*										VRSC, VRSCP: 5 steps
D								*	*	*	*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

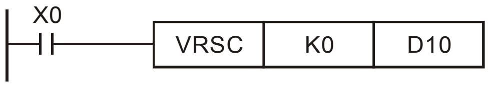
S: Variable resistor number (0~1) **D**: Destination device for storing scaled value

Explanations:

VRSC instruction reads the scaled value (0~10) of the 2 VRs on PLC and stores the read data in destination device **D** as an integer, i.e. if the value is between 2 graduations, the value will be rounded off.

Program Example 1:

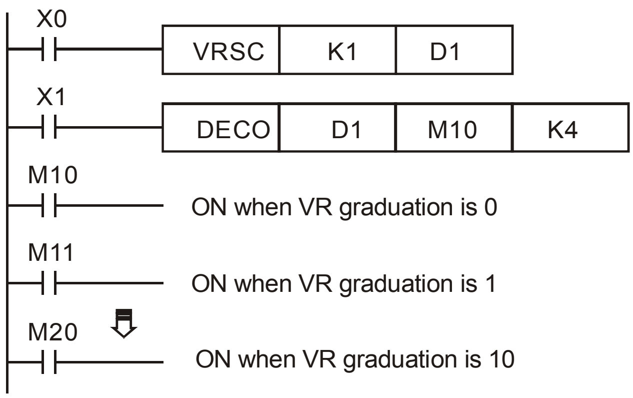
When X0 = ON, VRSC instruction reads the scaled value (0 to10) of VR No. 0 and stores the read value in device D10.



Program Example 2:

Apply the VR as digital switch: The graduations 0~10 of VR correspond to M10~M20, therefore only one of M10 ~M20 will be ON at a time. When M10~M20 is ON, use DECO instruction (API 41) to decode the scaled value into M10~M25.

1. When X0 = ON, the graduation (0~10) of VR No.1 will be read out and stored in D1.
2. When X1 = ON, DECO instruction will decode the graduation (0~10) into M10~M25.



API	Mnemonic			Operands	Function	Controllers
87	D	ABS	P	D	Absolute Value	ES2/EX2 SS2 SA2 SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

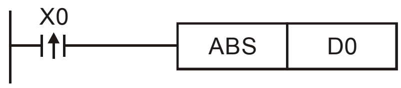
D: Device for absolute value operation

Explanation

1. The instruct ion conducts absolute value operation on **D**
2. This instruction is generally used in pulse execution mode (ABSP, DABSP).
3. If operand **D** uses index F, then only 16-bit instruction is available.

Program Example:

When X0 goes from OFF to ON, ABS instruction obtains the absolute value of the content in D0.



API	Mnemonic		Operands	Function	Controllers
88	D	PID	(S1) (S2) (S3) (D)	PID control	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1													*			PID : 9 steps
S_2													*			DPID: 17 steps
S_3													*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

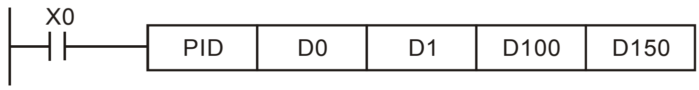
S_1: Set value (SV) **S_2**: Present value (PV) **S_3**: Parameter setting (for 16-bit instruction, uses 20 consecutive devices, for 32-bit instruction, uses 21 consecutive devices) **D**: Output value (MV)

Explanations:

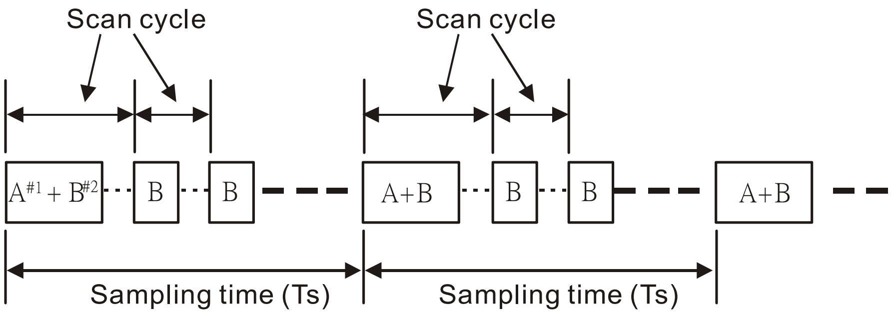
1. This instruction is specifically for PID control. PID operation will be executed only when the sampling time is reached. PID refers to “proportion, integration and derivative”. PID control is widely applied to many mechanical, pneumatic and electronic equipment.
2. After all the parameters are set up, PID instruction can be executed and the results will be stored in **D**. **D** has to be unlatched data register. (If users want to designate a latched data register area, please clear the latched registers to 0 in the beginning of user program.

Program Example:

1. Complete the parameter setting before executing PID instruction.
2. When X0 = ON, the instruction will be executed and the result will be stored in D150. When X0 = OFF, the instruction will not be executed and the previous data in D150 will stay intact.



3. Timing chart of the PID operation (max. operation time is approx. 80us)



Note: #1→ The time for equation calculation during PID operation (approx. 72us)
#2→ The PID operation time without equation calculation (approx. 8us)

Points to note:

1. There is no limitation on the times of using this instruction. However, the register No. designated in $\mathbf{S_3}$~ $\mathbf{S_3}$+19 cannot be repeated.
2. For 16-bit instruction, $\mathbf{S_3}$ occupies 20 registers. In the program example above, the area designated in $\mathbf{S_3}$ is D100 ~ D119.
3. Before the execution of PID instruction, users have to transmit the parameters to the designated register area by MOV instruction. If the designated registers are latched, use MOVP instruction to transmit all parameters only once
4. Settings of $\mathbf{S_3}$ in the 16-bit instruction:

Device No.	Function	Setup Range	Explanation
$\mathbf{S_3}$:	Sampling time (T_S)	1~2,000 (unit: 10ms)	Time interval between PID calculations and updates of MV. If T_S = 0, PID instruction will not be enabled. If T_S is less than 1 program scan time, PID instruction sets $\mathbf{S_3}$ as 1 program scan time, i.e. the minimum T_S has to be longer than the program scan time.
$\mathbf{S_3}$+1:	Propotional gain (K_P)	0~30,000(%)	The proportion for magnifying/minifying the error between SV and PV.
$\mathbf{S_3}$+2:	Integral gain (K_I)	0~30,000(%)	The proportion for magnifying/minifying the integral value (The accumulated error). For control mode K0~K5.
	Integral time constant (T_I)	0~30,000 (ms)	For control mode K10
$\mathbf{S_3}$+3:	Derivative gain (K_D)	-30,000~30,000 (%)	The proportion for magnifying/minifying the derivative value (The rate of change of the process error). For control mode K0~K5
	Derivative time constant (T_D)	-30,000~30,000 (ms)	For control mode K10
$\mathbf{S_3}$+4:	Control mode	0: Automatic control 1: Forward control (E = SV - PV).	

Device No.	Function	Setup Range	Explanation
		2: Reverse control (E = PV - SV). 3: Auto-tuning of parameter exclusively for the temperature control. The device will automatically become K4 when the auto-tuning is completed and K_P, K_I and K_D is set with appropriate value (not avaliable in the 32-bit instruction). 4: Exclusively for the adjusted temperature control (not avaliable in the 32-bit instruction). 5: Automatic mode with MV upper/lower bound control. When MV reaches upper/lower bound, the accumulation of integral value stops. 10: T_I / T_D mode with MV upper/lower bound control. When MV reaches upper/lower bound, the accumulation of integral value stops.	
S_3+5:	Tolerable range for error (E)	0~32,767	E = the error between SV and PV. If **S_3** +5 is set as 5, when E is between -5 and 5, E will be 0. When **S_3** +5 = K0, the function will not be enabled.
S_3+6:	Upper bound of output value (MV)	-32,768~32,767	Ex: if **S_3**+6 is set as 1,000, MV will be 1,000 when it exceeds 1,000. **S_3**+6 has to be bigger or equal to **S_3**+7, otherwise the upper bound and lower bound value will switch.
S_3+7:	Lower bound of output value (MV)	-32,768~32,767	Ex: if **S_3**+7 is set as -1,000, MV will be -1,000 when it is smaller than -1,000..
S_3+8:	Upper bound of integral value	-32,768~32,767	Ex: if **S_3**+8 is set as 1,000, the integral value will be 1,000 when it is bigger than 1,000 and the integration will stop. **S_3**+8 has to be bigger or equal **S_3** +9; otherwise the upper bound and lower bound value will switch
S_3+9:	Lower bound of integral value	-32,768~32,767	Ex: if **S_3**+9 is set as -1,000, the integral value will be -1,000 when it is smaller than -1,000 and the integration will stop.

Device No.	Function	Setup Range	Explanation
$\mathbf{S_3}$+10, 11:	Accumulated integral value	Available range of 32-bit floating point	The accumulated integral value is usually for reference. Users can clear or modify it (in 32-bit floating point) according to specific needs.
$\mathbf{S_3}$ +12:	The previous PV	-32,768~32,767	The previous PV is usually for reference. Users can clear or modify it according to specific needs.
$\mathbf{S_3}$+13 ~ $\mathbf{S_3}$+19	For system use only..		

5. For $\mathbf{S_3}$+1~3, when parameter setting exceeds its range, the upper / lower bound will be selected as the set value.
6. If the direction setting (Forward / Reverse) exceeds its range, it will be set to 0.
7. PID instruction can be used in interruption subroutines, step ladders and CJ instruction.
8. The maximum error of sampling time T_S = - (1 scan time + 1ms) ~ + (1 scan time). When the error affects the output, please fix the scan time or execute PID instruction in timer interrupt.
9. PV of PID instruction has to be stable before PID operation executes. If users need to take the value input from AIO modules for PID operation, care should be taken on the A/D conversion time of these modules
10. For 32-bit instruction, $\mathbf{S_3}$ occupies 21 registers. In the program example above, the area designated in $\mathbf{S_3}$ will be D100 ~ D120. Before the execution of PID instruction, users have to transmit the parameters to the designated register area by MOV instruction. If the designated registers are latched, use MOVP instruction to transmit all parameters only once.
11. Parameter table of 32-bit $\mathbf{S_3}$:

Device No.	Function	Set-point range	Explanation
$\mathbf{S_3}$:	Sampling time (T_S)	1~2,000 (unit: 10ms)	Time interval between PID calculations and updates of MV. If T_S= 0, PID instruction will not be enabled. If T_S is less than 1 program scan time, PID instruction sets $\mathbf{S_3}$ as 1 program scan time, i.e. the minimum T_S has to be longer than the program scan time.

Device No.	Function	Set-point range	Explanation
S_3+1:	Proportional gain (K_P)	0~30,000(%)	The proportion for magnifying/minifying the error between SV and PV.
S_3+2:	Integration gain (K_I)	0~30,000(%)	The proportion for magnifying/minifying the integral value (The accumulated error). For control mode K0~K2, K5.
	Integral time constant (T_I)	0~30,000 (ms)	For control mode K10
S_3+3:	Derivative gain (K_D)	-30,000~30,000 (%)	The proportion for magnifying/minifying the derivative value (The rate of change of the process error). For control mode K0~K2, K5.
	Derivative time constant (T_D)	-30,000~30,000 (ms)	For control mode K10
S_3+4:	Control mode	0: Automatic control 1: Forward control (E = SV - PV). 2: Reverse control (E = PV - SV). 5: Automatic mode with MV upper/lower bound control. When MV reaches upper/lower bound, the accumulation of integral value stops. 10: T_I / T_D mode with MV upper/lower bound control. When MV reaches upper/lower bound, the accumulation of integral value stops.	
S_3+5, 6:	Tolerable range for error (E), 32-bit	0~ 2,147,483,647	E = the error between SV and PV. If **S_3** +5 is set as 5, when E is between -5 and 5, E will be 0. When **S_3** +5 = K0, the function will not be enabled.
S_3+7, 8:	Upper bound of output value (MV) , 32-bit	-2,147,483,648 ~ 2,147,483,647	Ex: if **S_3**+6 is set as 1,000, MV will be 1,000 when it exceeds 1,000. **S_3**+6 has to be bigger or equal to **S_3**+7, otherwise the upper bound and lower bound value will switch

Device No.	Function	Set-point range	Explanation
S_3+9, 10:	Lower bound of output value (MV) , 32-bit	-2,147,483,648 ~ 2,147,483,647	Ex: if **S_3**+7 is set as -1,000, MV will be -1,000 when it is smaller than -1,000.
S_3+11, 12:	Upper bound of integral value, 32-bit	-2,147,483,648 ~ 2,147,483,647	Ex: if **S_3**+8 is set as 1,000, the integral value will be 1,000 when it is bigger than 1,000 and the integration will stop. **S_3**+8 has to be bigger or equal **S_3** +9; otherwise the upper bound and lower bound value will switch.
S_3+13, 14:	Lower bound of integral value, 32-bit	-2,147,483,648 ~ 2,147,483,647	Ex: if **S_3**+9 is set as -1,000, the integral value will be -1,000 when it is smaller than -1,000 and the integration will stop.
S_3+15, 16:	Accumulated integral value, 32-bit	Available range of 32-bit floating point	The accumulated integral value is usually for reference. Users can clear or modify it (in 32-bit floating point) according to specific needs.
S_3+17, 18:	The previous PV, 32-bit	-2,147,483,648 ~2,147,483,647	The previous PV is usually for reference. Users can clear or modify it according to specific needs.
S_3+19, 20	For system use only.		

12. The explanation of 32-bit **S_3** and 16-bit **S_3** are almost the same. The difference is the capacity of **S_3**+5 ~ **S_3**+20.

PID Equations:

1. When control mode (**S_3**+4) is selected as K0, K1, K2 and K5:
   - In this control mode, PID operation can be selected as Automatic, Forward, Reverse and Automatic with MV upper/lower bound control modes. Forward / Reverse direction is designated in **S_3**+4. Other relevant settings of PID operation are set by the registers designated in **S_3** ~ **S_3**+5.

- PID equation for control mode k0~k2:

$$MV = K_P * E(t) + K_I * E(t)\frac{1}{S} + K_D * PV(t)S$$

where

MV : Output value
K_P : Proprotional gain
$E(t)$: Error value
$PV\ (t)$: Present measured value
$SV\ (t)$: Target value
K_D : Derivative gain
$PV(t)S$: Derivative value of PV(t)
K_I : Integral gain
$E(t)\frac{1}{S}$: Integral value of E(t)

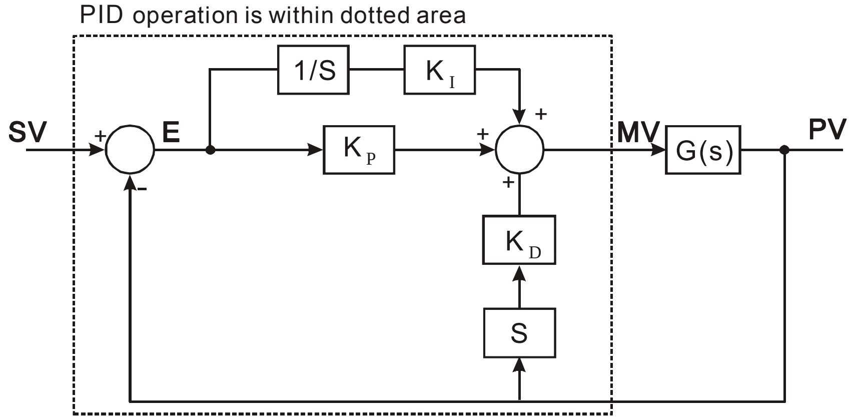
- When E(t) is smaller than 0 as the control mode is selected as forward or inverse, E(t) will be regarded as “0"

Control mode	PID equation
Forward, automatic	E(t) = SV – PV
Inverse	E(t) = PV – SV

- Control diagram:

In diagram below, S is derivative operation, referring to “(PV− previous PV) ÷ sampling time”. 1 / S is integral operation, referring to “previous integral value + (error value × sampling time)”. G(S) refers to the device being controlled.

PID operation is within dotted area



- The equation above illustrates that this operation is different from a general PID operation on the application of the derivative value. To avoid the fault that the transient derivative value could be too big when a general PID instruction is first executed, our PID instruction monitors the derivative value of the PV. When the variation of PV is excessive, the instruction will reduce the output of MV/.

2. When control mode ($\mathbf{S_3}$+4) is selected as K3 and K4:

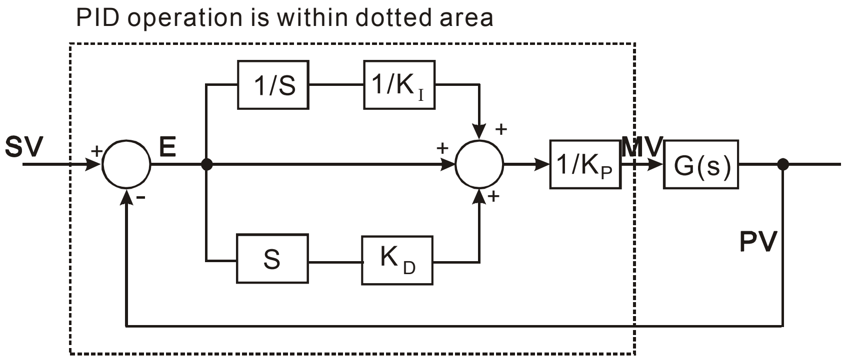
- The equation is exclusively for temperature control will be modified as:

$$MV = \frac{1}{K_P}\left[E(t) + \frac{1}{K_I}\left(E(t)\frac{1}{S}\right) + K_D * E(t)S\right],$$

where $\mathrm{E(t)} = \mathrm{SV(t)} - \mathrm{PV(t)}$

- Control diagram:

In diagram below, $1/K_I$ and $1/K_P$ refer to "divided by K_I" and "divided by K_P". Because this mode is exclusively for temperature control, users have to use PID instruction together with GPWM instruction. See **Application 3** for more details



- This equation is exclusively designed for temperature control. Therefore, when the sampling time (T_S) is set as 4 seconds (K400), the range of output value (MV) will be K0 ~ K4,000 and the cycle time of GPWM instruction used together has to be set as 4 seconds (K4000) as well.

- If users have no idea on parameter adjustment, select K3 (auto-tuning). After all the parameters are adjusted (the control direction will be automatically set as K4), users can modify the parameters to better ones according to the adjusted results.

3. When control mode ($\mathbf{S_3}$+4) is selected as K10:

- $\mathbf{S_3}$+2 (K_I) and $\mathbf{S_3}$+3 (K_D) in this mode will be switched to parameter settings of Integral time constant (T_I) and Derivative time constant (T_D).

- When output value (MV) reaches the upper bound, the accumulated integral value will not increase. Also, when MV reaches the lower bound, the accumulated integral value will not decrease.

- The equation for this mode will be modified as:

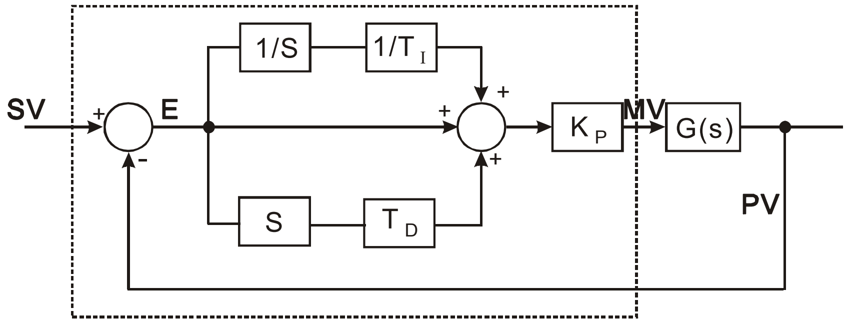
$$MV = K_P \times \left[E(t) + \frac{1}{T_I}\int E(t)dt + T_D\frac{d}{dt}E(t)\right]$$

Where

$$E(t) = SV(t) - PV(t)$$

- Control diagram:

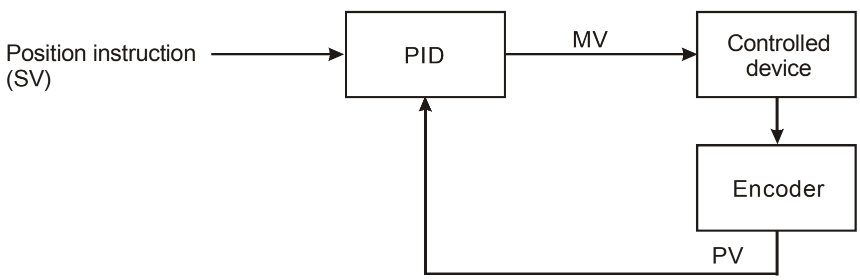
PID operation is within dotted area



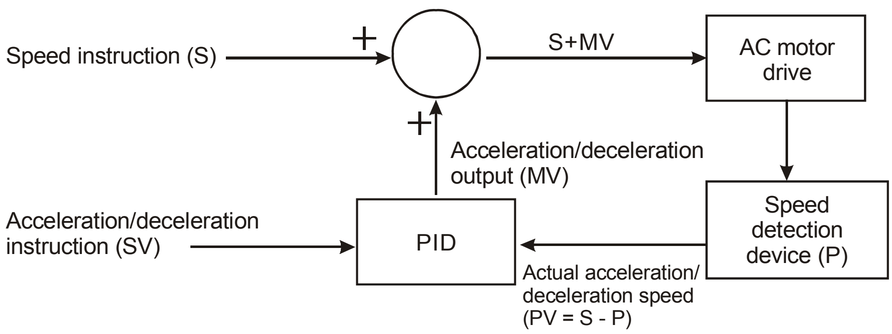
Notes and suggestion:

1. $\mathbf{S_3}$ + 3 can only be the value within 0 ~ 30,000.
2. There are a lot of circumstances where PID instruction can be applied; therefore, please choose the control functions appropriately. For example, when users select parameter auto-tuning for the temperature ($\mathbf{S_3}$ + 4 = K3), the instruction can not be used in a motor control environment otherwise improper control may occur.
3. When you adjust the three main parameters, K_P, K_I and K_D ($\mathbf{S_3}$ + 4 = K0 ~ K2), please adjust K_P first (according to your experiences) and set K_I and K_D as 0. When the output can roughly be controlled, proceed to increase K_I and K_D (see example 4 below for adjustment methods). K_P = 100 refers to 100%, i.e. the proportional gain to the error is 1. K_P < 100% will decrease the error and K_P > 100% will increase the error
4. When temperature auto-tuning function is selected($\mathbf{S_3}$ + 4 = K3, K4), it is suggested that store the parameters in D register in latched area in case the adjusted parameters will disappear after the power is cut off. There is no guarantee that the adjusted parameters are suitable for every control requirement. Therefore, users can modify the adjusted parameters according to specific needs, but it is suggested to modify only K_I or K_D.
5. PID instruction has to be controlled with many parameters; therefore care should be taken when setting each parameter in case the PID operation is out of control.

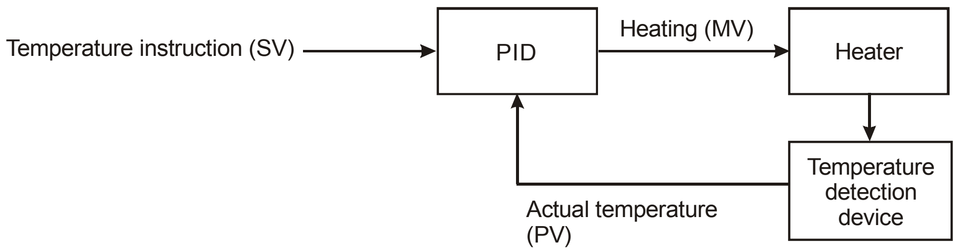
Example 1: Block diagram of application on positioning ($\mathbf{S_3}$+4 = 0)



Example 2: Block diagram of application on AC motor drive (**S_3**+4 = 0)



Example 3: Block diagram of application on temperature control (**S_3**+4 = 1)

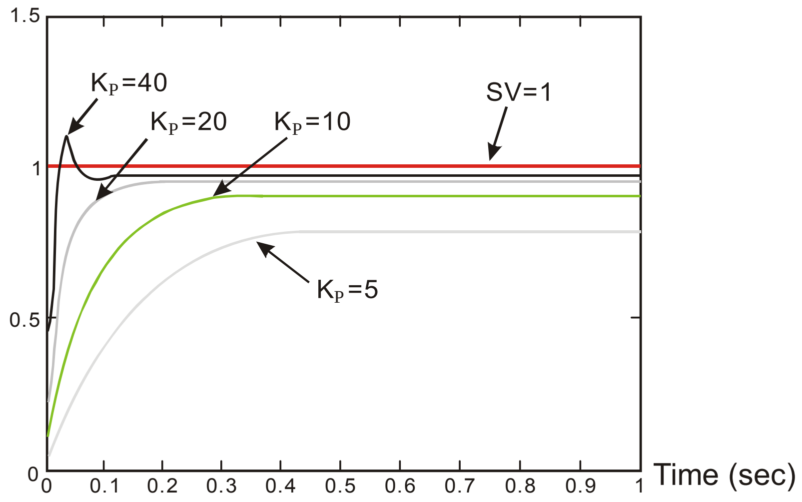


Example 4: PID parameters adjustment

Assume that the transfer function of the controlled device G(S) in a control system is a first-order function $G(s)=\frac{b}{s+a}$ (model of general motors), SV = 1, and sampling time (T_S) = 10ms. Suggested steps for adjusting the parameters are as follows:

Step1:

Set K_I and K_D as 0, and K_P as 5, 10, 20, 40. Record the SV and PV respectively and the results are as the figure below.



Step 2:

When K_P is 40, response overshoot occurs, so we will not select it.

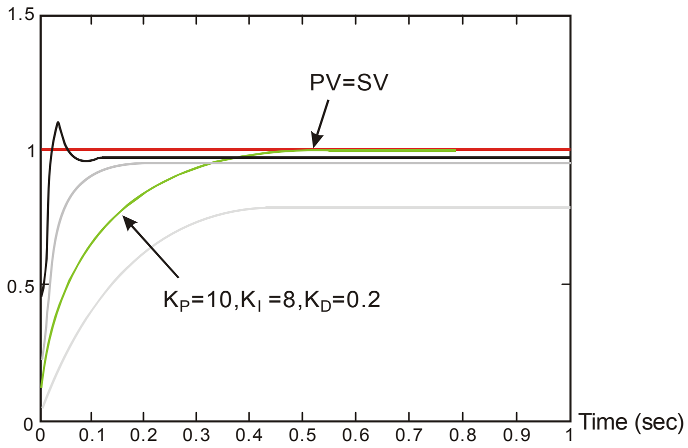
When K_P is 20, PV response is close to SV and won't overshoot, but transient MV will be to large due to a fast start-up. We can put it aside and observe if there are better curves.

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:

Select K_P = 10 and increase K_I gradually, e.g. 1, 2, 4, 8. K_I should not be bigger than K_P. Then, increase K_D as well, e.g. 0.01, 0.05, 0.1, 0.2. K_D should not exceed 10% of K_P. Finally we obtain the figure of PV and SV below.



Application 1:

PID instruction in pressure control system. (Use block diagram of example 1)

Control purpose:

Enabling the control system to reach the target pressure.

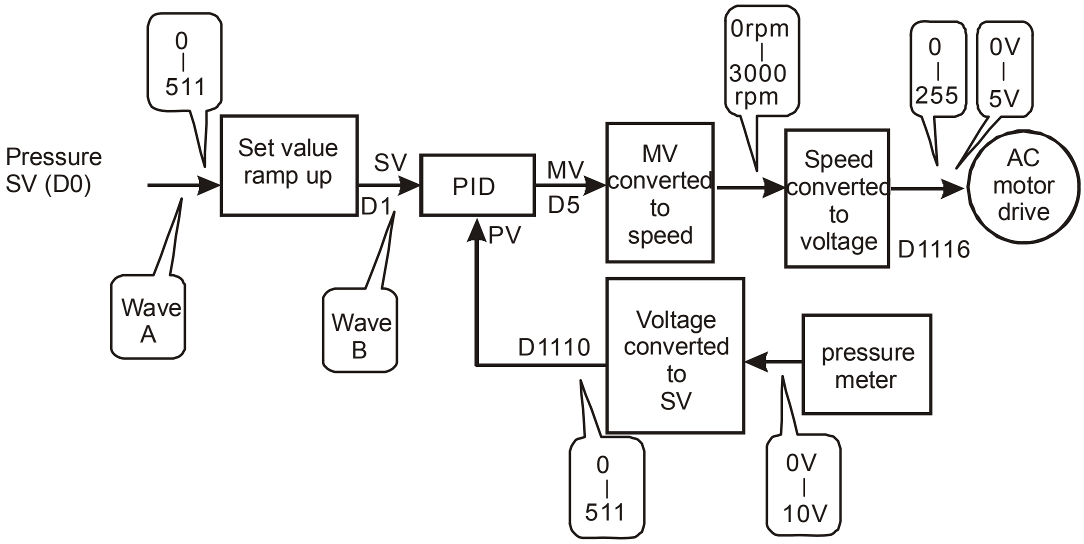
Control properties:

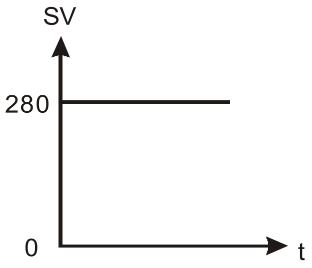
The system requires a gradual control. Therefore, the system will be overloaded or out of control if the process progresses too fast.

Suggested solution:

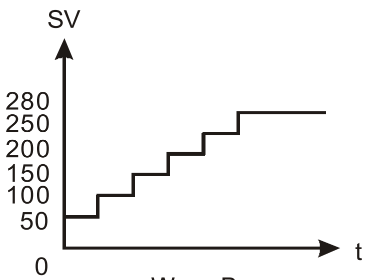
Solution 1: Longer sampling time

Solution 2: Using delay instruction. See the figure below





Wave A

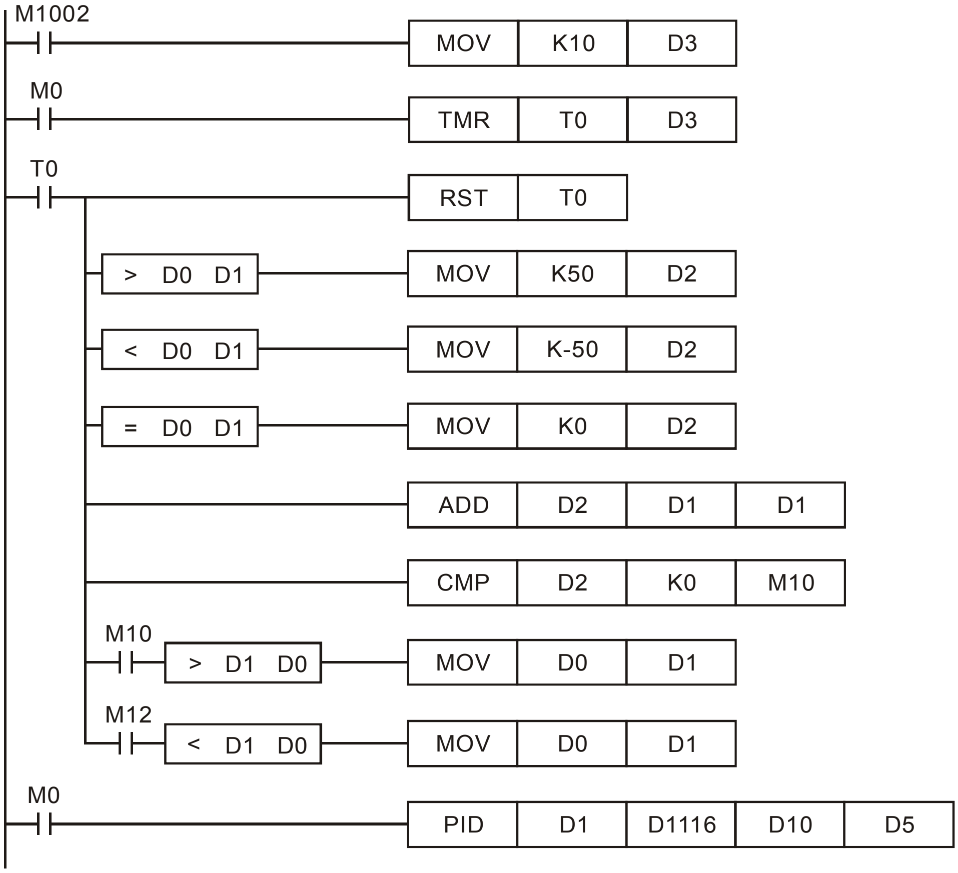


Wave B

D2 stores increased value of each shift
D3 stores the time interval of each shift

Values in can modify D2 and D3 according to actual requirement

Example program of SV ramp up function:



Application 2:

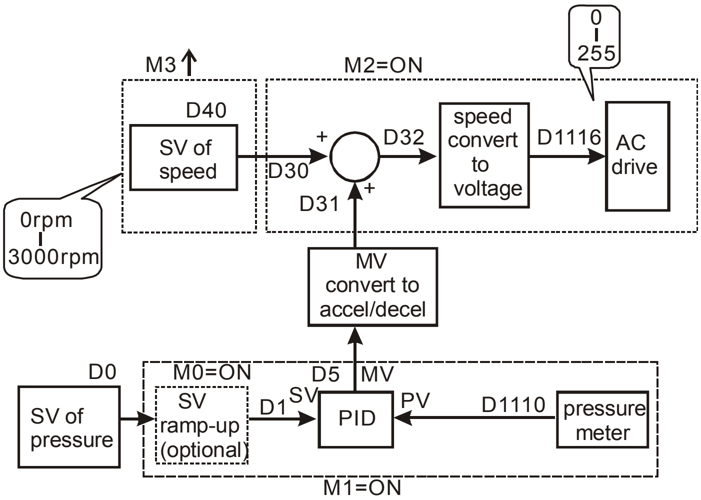
Speed control system and pressure control system work individually (use diagram of Example 2)

Control purpose:

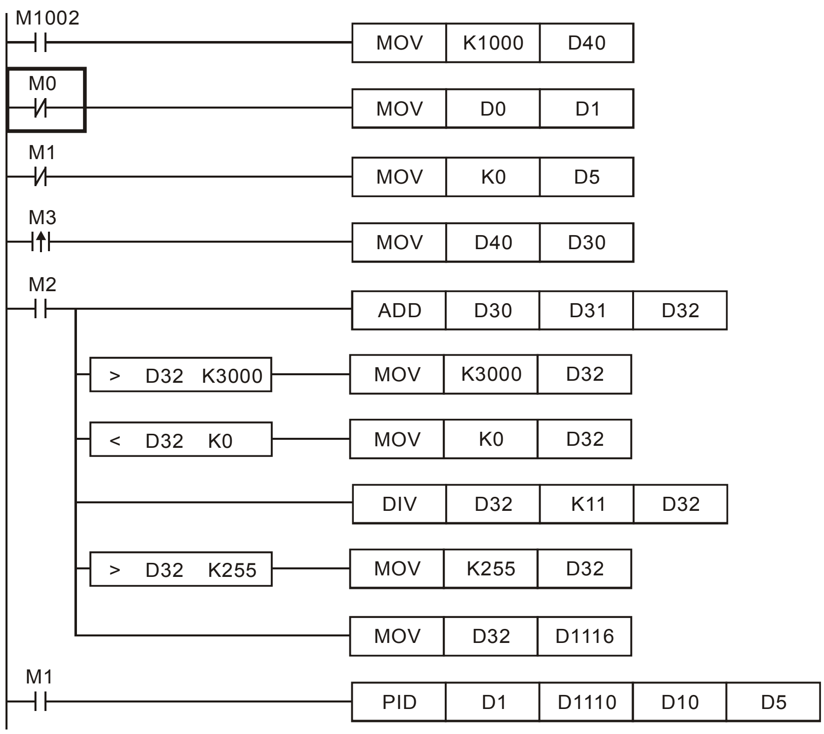
After the speed control operates in open loop for a period of time, adding pressure control system (PID instruction) to perform a close loop control.

Control properties:

Since the speed and pressure control systems are not interrelated, we have to structure an open loop for speed control first following by a close loop pressure control. If users afraid that the pressure control system changes excessively, consider adding the SC ramp-up function illustrated in **Application 1** into this control. See the control diagram below.



Part of the example program:



Application 3:

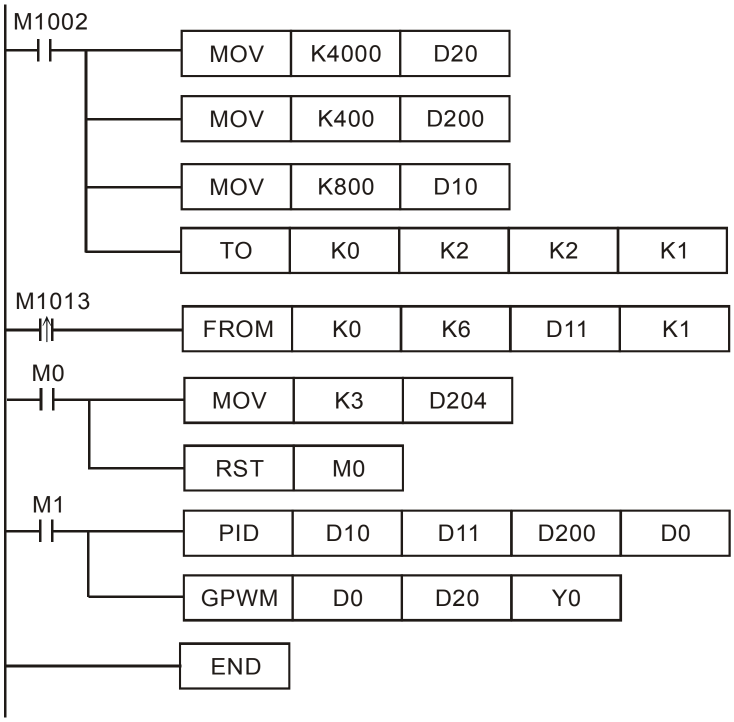
Using auto-tuning for temperature control

Control purpose:

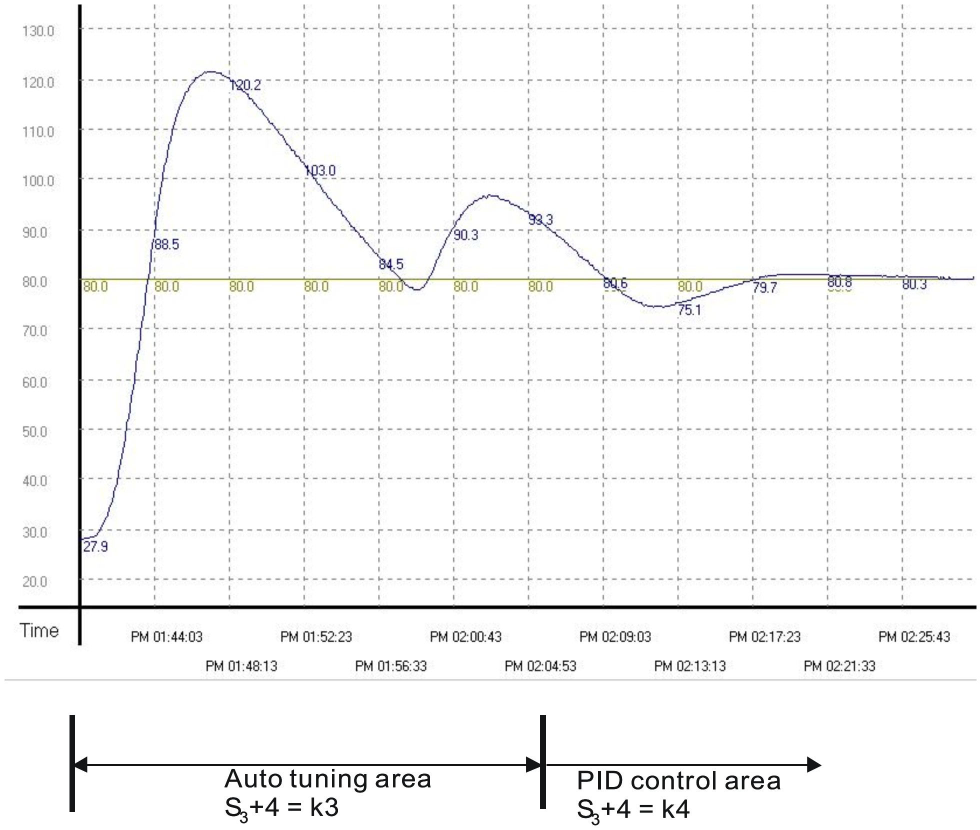
Calculating optimal parameter of PID instruction for temperature control

Control properties:

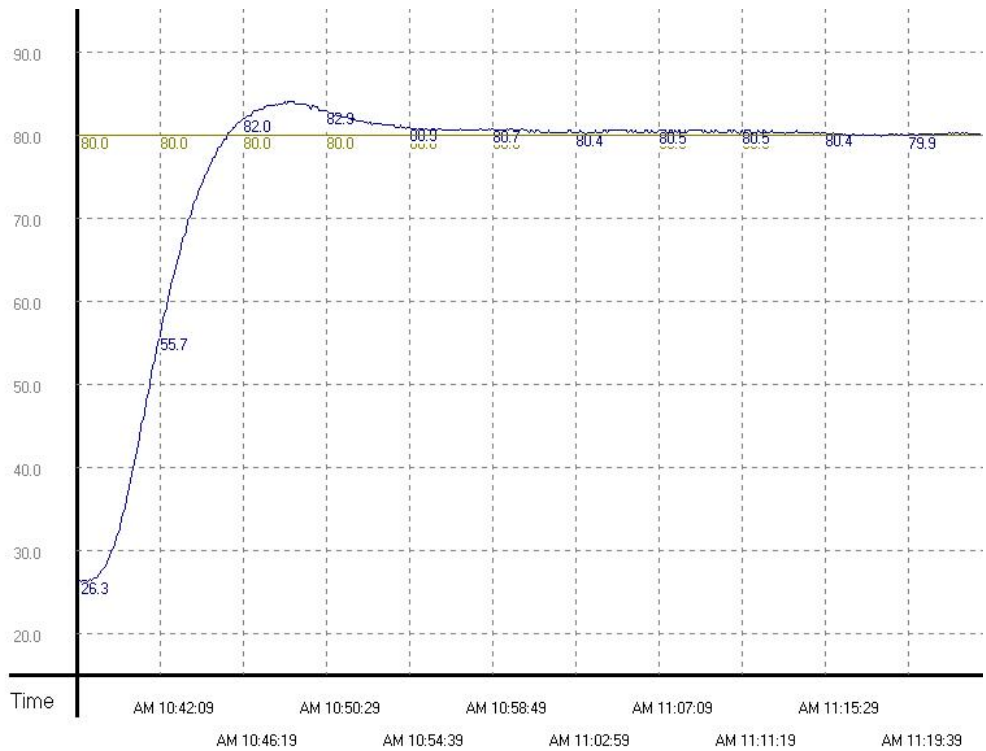
Users may not be familiar with a new temperature environment. In this case, selecting auto-tuning ($\mathbf{S_3}$+4 = K3) for an initial adjustment is suggested. After initial tuning is completed, the instruction will auto modify control mode to the mode exclusively for adjusted temperature ($\mathbf{S_3}$+4 = K4). In this example, the control environment is a heating oven. See the example program below.



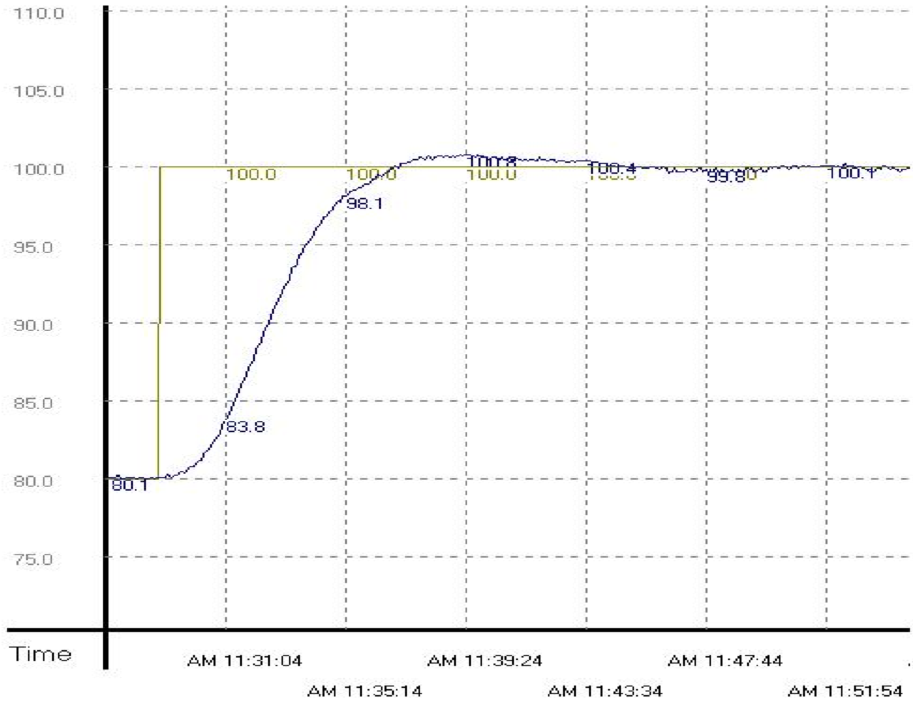
Results of initial auto-tuning



Results of using adjusted parameters generated by initial auto-tuning function.



From the figure above, we can see that the temperature control after auto-tuning is working fine and it spent only approximately 20 minutes for the control. Next, we modify the target temperature from 80°C to 100°C and obtain the result below.



From the result above, we can see that when the parameter is 100°C, temperature control works fine and costs only 20 minutes same as that in 80°C.

API	Mnemonic	Operands	Function	Controllers
89	PLS	(S)	Rising-edge output	ES2/EX2 SS2 SA2 SX2

Type / OP	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		*	*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

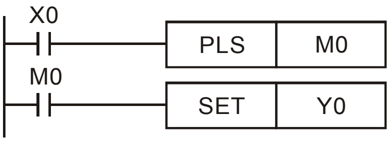
S: Rising pulse output device

Explanations:

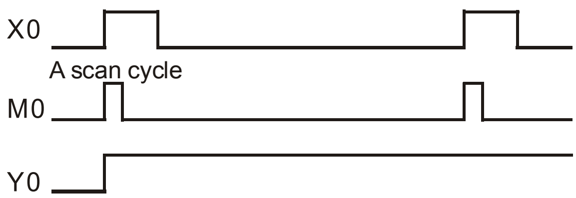
When X0 goes from OFF to ON (Rising-edge trigger), PLS instruction executes and **S** generates a cycle pulse for one operation cycle.

Program Example:

Ladder Diagram:



Timing Diagram:



Instruction Code:		Operation:
LD	X0	; Load NO contact of X0
PLS	**M0**	; M0 rising-edge output
LD	M0	; Load NO contact of M0
SET	Y0	; Y0 latched (ON)

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
90	LDP	S	Rising–edge detection operation				

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	*	*	*	*							*	*				

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: device to be rising-edge triggered

Explanations:

LDP should be connected to the left side bus line. When the associated device **S** is driven from OFF to ON, LDP will be ON for one scan cycle.

Program Example:

Ladder Diagram:

Instruction Code:		Operation:
LDP	**X0**	; Load rising-edge contact X0
AND	X1	; Connect NO contact X1 in series
OUT	Y1	; Drive Y1 coil

Points to Note:

1. If the associated rising-edge contact is ON before PLC is power on, the contact will be activated after PLC is power on.

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
91	LDF	S	Falling–edge detection operation				

Type / OP	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	*	*	*	*							*	*				

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: device to be falling pulse triggered

Explanations:

LDF should be connected to the left side bus line. When the associated device **S** is driven from ON to OFF, LDF will be ON for one scan cycle.

Program Example:

Ladder Diagram:

Instruction Code: Operation:

LDF	**X0**	; Load falling-edge contact X0
AND	X1	; Connect NO contact X1 in series.
OUT	Y1	; Drive Y1 coil

API	Mnemonic	Operands	Function	Controllers
92	ANDP	S	Rising-edge series connection	ES2/EX2 SS2 SA2 SX2

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	*	*	*	*							*	*				

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: rising-edge contact to be connected in series

Explanations:

ANDP instruction is used in the series connection of the rising-edge contact.

Program Example:

Ladder Diagram:

Instruction Code:		Operation:
LD	X0	; Load NO contact of X0
ANDP	**X1**	; X1 rising-edge contact in series connection
OUT	Y1	; Drive Y1 coil

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
93	ANDF	S	Falling-edge series connection				

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	ANDF: 3 steps
S	*	*	*	*							*	*				

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: falling edge contact to be connected in series

Explanations:

ANDF instruction is used in the series connection of the falling-edge contact.

Program Example:

Ladder Diagram:

Instruction Code:		Operation:
LD	X0	; Load NO contact of X0
ANDF	**X1**	; X1 falling-edge contact in series connection
OUT	Y1	; Drive Y1 coil

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
94	ORP	S	Rising-edge parallel connection				

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	*	*	*	*							*	*				

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

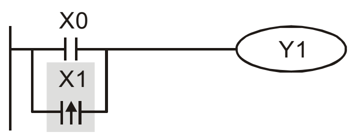
S: rising-edge contact to be connected in parallel

Explanations:

ORP instruction is used in the parallel connection of the rising-edge contact.

Program Example:

Ladder Diagram:



Instruction Code:

Instruction Code		Operation
LD	X0	; Load NO contact of X0
ORP	**X1**	; X1 rising-edge contact in parallel connection
OUT	Y1	; Drive Y1 coil

API	Mnemonic	Operands	Function	Controllers
95	ORF	S	Falling-edge parallel connection	ES2/EX2 SS2 SA2 SX2

Type / OP	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	*	*	*	*							*	*				

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

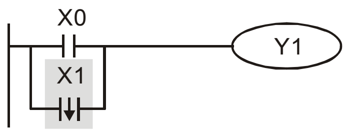
S: falling-edge contact to be connected in parallel

Explanations:

ORF instruction is used in the parallel connection of the falling-edge contact..

Program Example:

Ladder Diagram:



Instruction Code:	Operation:
LD X0	; Load NO contact of X0
ORF X1	; X1 falling-edge contact in parallel connection
OUT Y1	; Drive Y1 coil

API	Mnemonic	Operands	Function	Controllers
96	TMR	(S1) (S2)	Timer	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	TMR: 5 steps
S_1											*					
S_2					*								*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: No. of timer (T0~T255) **S_2**: Set value (K0~K32,767, D0~D9,999)

Explanations:

When TMR instruction is executed, the specific coil of timer is ON and the timer is enabled. When the set value of timer is achieved, the associated NO/NC contact will be driven.

Program example:

Ladder Diagram:

Instruction Code:			Operation:
LD	X0		; Load NO contact X0
TMR	**T5**	**K1000**	; T5 timer setting is K1000

API	Mnemonic	Operands	Function	Controllers
97	CNT	S1 S2	16-bit counter	ES2/EX2 SS2 SA2 SX2

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					*								*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: No. of 16-bit counter (C0~C199) **S_2**: Set value (K0~K32,767, D0~D9,999)

Explanations:

1. When the CNT instruction is executed, the specific coil of counter is driven from OFF to ON once, which means the count value of counter will be added by7 1. When the accumulated count value achieves the set value, the associated NO/NC contact will be driven.
2. When set value of counter is achieved and the counter is driven again, the count value and the status of the associated contact will remain intact. If users need to restart the counting or clear the count value, please use RST instruction.

Program example:

Ladder Diagram:

Instruction Code: Operation:

LD	X0	; Load NO contact X0
CNT	**C20 K100**	; C20 counter setting is K100

API	Mnemonic	Operands	Function	Controllers
97	DCNT	(S1) (S2)	32-bit counter	ES2/EX2 | SS2 | SA2 | SX2

Type / OP	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					*								*				

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: No. of 32-bit counter (C200~C254)

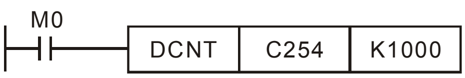
S_2: Set value (K-2,147,483,648~K2,147,483,647, D0~D9,999)

Explanations:

1. DCNT is the startup instruction for the 32-bit counters C200 to C254.
2. For general counting up/down counters C200~C231(SS2/SA2/SX2: C200~C232), the present value will plus 1 or minus 1 according to the counting mode set by flags M1200~M1231 when instruction DCNT is executed.
3. For high speed counters C232~C254(SS2/SA2/SX2: C233~C254), when the specified high speed counter input is triggered by pulse, the counters will start counting. For details about high-speed input terminals (X0~X7) and counting modes (count up/down), please refer to section **2.12 C (Counter).**
4. When DCNT instruction is OFF, the counter will stop counting, but the count value will not be cleared. Users can use RST instruction to remove the count value and reset the contact, or use DMOV instruction to move a specific value into the register. For high-speed counters C232~C254, use specified external input point to clear the count value and reset the contacts.

Program Example:

Ladder Diagram:



Instruction Code: Operation:

LD	M0	; Load NO contact M0
DCNT	**C254 K1000**	; C254 counter setting is K1000

API	Mnemonic	Operands	Function	Controllers
98	INV	-	Inverse operation	ES2/EX2 SS2 SA2 SX2

OP	Descriptions	Program Steps
N/A	Invert the current result of the internal PLC operations	INV: 1 step

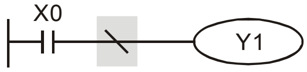
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Explanations:

INV instruction inverts the logical operation result.

Program Example:

Ladder Diagram:



Instruction Code: Operation:

Instruction Code:		Operation:
LD	X0	; Load NO contact X0
INV		; Invert the operation result
OUT	Y1	; Drive Y1 coil

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
99	PLF	S	Falling-edge output				

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		*	*													

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

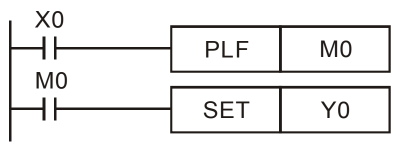
S: Falling pulse output device

Explanations:

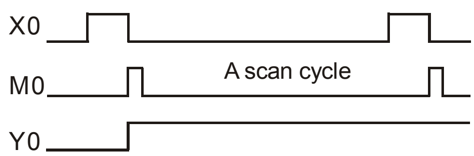
When X0 goes from ON to OFF (Falling-edge trigger), PLS instruction executes and **S** generates a cycle pulse for one operation cycle.

Program Example:

Ladder Diagram:



Timing Diagram:



Instruction Code:		Operation:
LD	X0	; Load NO contact X0
PLF	**M0**	; M0 falling-edge output
LD	M0	; Load NO contact M0
SET	Y0	; Y0 latched (ON)

API	Mnemonic	Operands	Function	Controllers
100	MODRD	S_1 S_2 n	Read Modbus Data	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
																MODRD: 7 steps
S_1					*	*							*			
S_2					*	*							*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

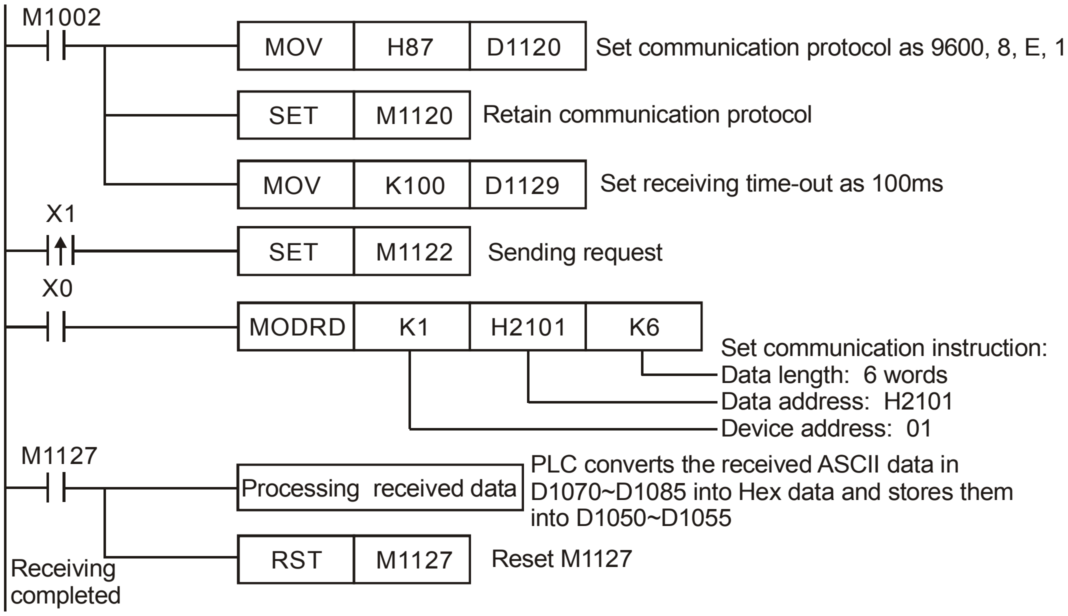
S_1: Device address (K0~K254)   **S_2**: Data address   **n**: Data length (K1 < **n** ≦ K6)

Explanations:

1. MODRD instruction supports COM2 (RS-485).
2. MODRD is an instruction exclusively for peripheral communication equipment in MODBUS ASCII/RTU mode. The built-in RS-485 communication ports in Delta VFD drives (except for VFD-A series) are all compatible with MODBUS communication format. MODRD can be used for communication (read data) of Delta drives.
3. If the address of **S_2** is illegal for the designated communication device, the device will respond with an error, PLC will record the error code in D1130 and M1141 will be ON.
4. The feedback (returned) data from the peripheral equipment will be stored in D1070 ~ D1085. After data receiving is completed, PLC will check the validity of the data automatically. If there is an error, M1140 will be ON.
5. The feedback data are all ASCII codes in ASCII mode, so PLC will convert the feedback data into hex data and store them in D1050 ~ D1055. D1050 ~ D1055 is invalid in RTU mode.
6. If peripheral device receives a correct record (data) from PLC after M1140/M1141 = ON, the peripheral device will send out feedback data and PLC will reset M1140/M1141 after the validity of data is confirmed.
7. There is no limitation on the times of using this instruction, but only one instruction can be executed at a time on the same COM port.
8. Rising-edge contact (LDP, ANDP, ORP) and falling-edge contact (LDF, ANDF, ORF) can not be used with MODRD instruction, otherwise the data stored in the receiving registers will be incorrect.
9. For associated flags and special registers, please refer to **Points to note** of API 80 RS instruction.

Program Example 1:

Communication between PLC and VFD-B series AC motor drives (ASCII Mode, M1143 = OFF)



PLC → VFD-B , PLC transmits: “01 03 2101 0006 D4”

VFD-B → PLC , PLC receives: “01 03 0C 0100 1766 0000 0000 0136 0000 3B”

Registers for data to be sent (sending messages)

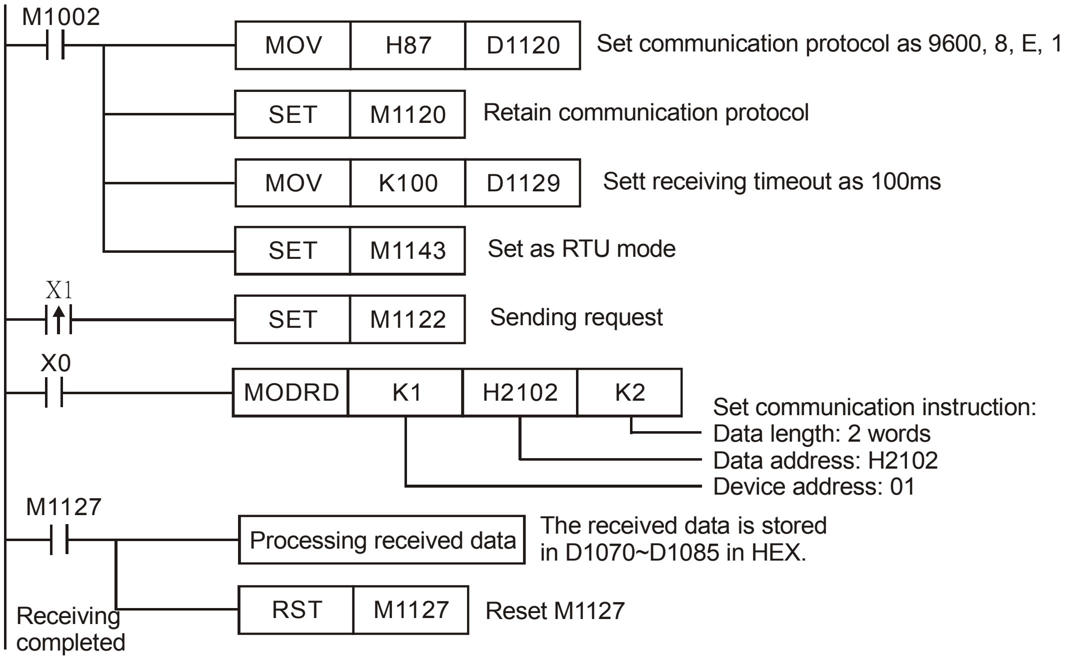
Register	Data		Descriptions	
D1089 low byte	‘0’	30 H	ADR 1	Address of AC motor drive: ADR (1,0)
D1089 high byte	‘1’	31 H	ADR 0	
D1090 low byte	‘0’	30 H	CMD 1	Command code: CMD (1,0)
D1090 high byte	‘3’	33 H	CMD 0	
D1091 low byte	2’	32 H	Starting data address	
D1091 high byte	‘1’	31 H		
D1092 low byte	‘0’	30 H		
D1092 high byte	‘1’	31 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	‘D’	44 H	LRC CHK 1	Checksum: LRC CHK (0,1)
D1095 high byte	‘4’	34 H	LRC CHK 0	

Registers for received data (responding messages)

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	‘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 2101 H	0100 H PLC automatically converts ASCII codes and store the converted value in D1050
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 2102 H	1766 H PLC automatically converts ASCII codes and store the converted value in D1051
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 2103 H	0000 H PLC automatically converts ASCII codes and store the converted value in D1052
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 2104 H	0000 H PLC automatically converts ASCII codes and store the converted value in D1053
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 2105 H	0136 H PLC automatically converts ASCII codes and store the converted value in D1054
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 2106 H	0000 H PLC automatically converts ASCII codes and store the converted value in D1055
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 PLC and VFD-B series AC motor drive (RTU Mode, M1143= ON)



PLC → VFD-B , PLC transmits: 01 03 2102 0002 6F F7

VFD-B → PLC, PLC receives: 01 03 04 1770 0000 FE 5C

Registers for data to be sent (sending messages)

Register	Data	Descriptions
D1089 low byte	01 H	Address of AC motor drive
D1090 low byte	03 H	Command code of AC motor drive
D1091 low byte	21 H	Starting data address
D1092 low byte	02 H	
D1093 low byte	00 H	Number of data (count by word)
D1094 low byte	02 H	
D1095 low byte	6F H	CRC CHK Low
D1096 low byte	F7 H	CRC CHK High

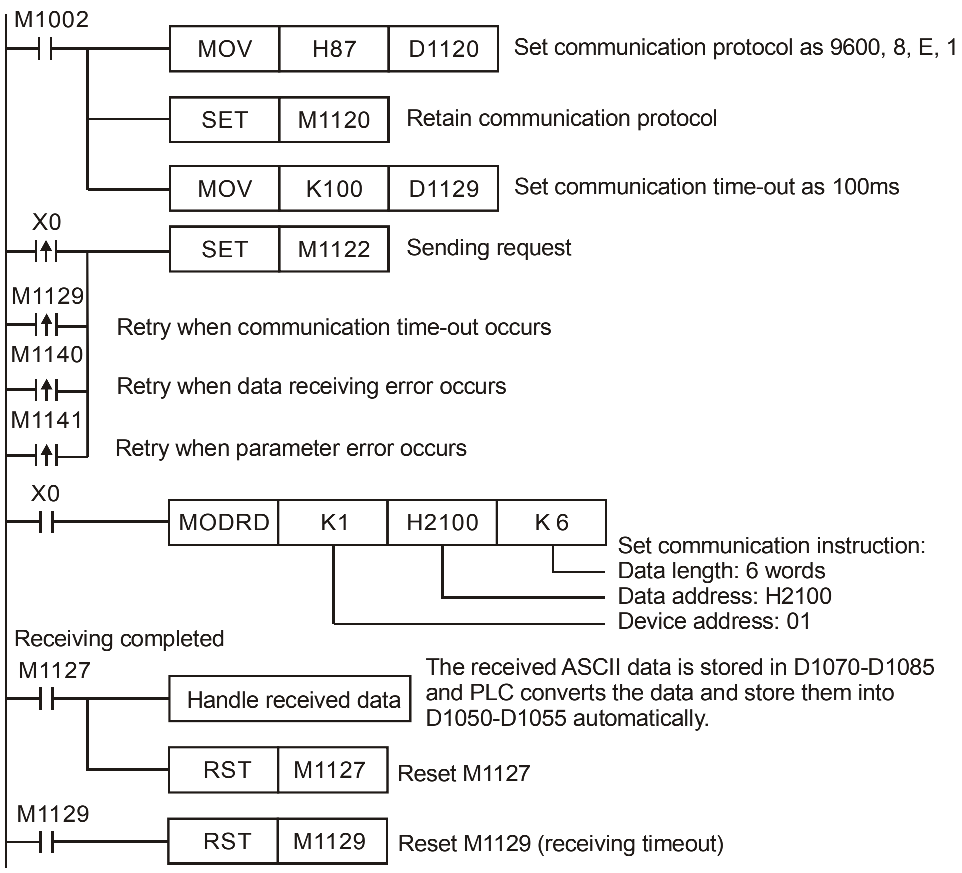
Registers for received data (responding messages)

Register	Data	Descriptions
D1070 low byte	01 H	Address of AC motor drive
D1071 low byte	03 H	Command code of AC motor drive
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	

Register	Data	Descriptions
D1077 low byte	FE H	CRC CHK Low
D1078 low byte	5C H	CRC CHK High

Program Example 3:

1. In the communication between PLC and VFD-B series AC motor drive (ASCII Mode, M1143 = OFF), executes Retry when communication time-out, data receiving error or parameter error occurs.
2. When X0 = ON, PLC will read the data of address H2100 in device 01(VFD-B) and stores the data in ASCII format in D1070 ~ D1085. PLC will automatically convert the data and store them in D1050 ~ D1055.
3. M1129 will be ON when communication time-out occurs. The program will trigger M1129 and send request for reading the data again.
4. M1140 will be ON when data receiving error occurs. The program will trigger M1140 and send request for reading the data again.
5. M1141 will be ON when parameter error occurs. The program will trigger M1141 and send request for reading the data again.



API	Mnemonic	Operands	Function	Controllers
101	MODWR	S1 S2 n	Write Modbus Data	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
																MODWR: 7 steps
S_1					*	*							*			
S_2					*	*							*			
n					*	*							*			

(Bit Devices: X, Y, M, S; Word devices: K, H, KnX, KnY, KnM, KnS, T, C, D, E, F)

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

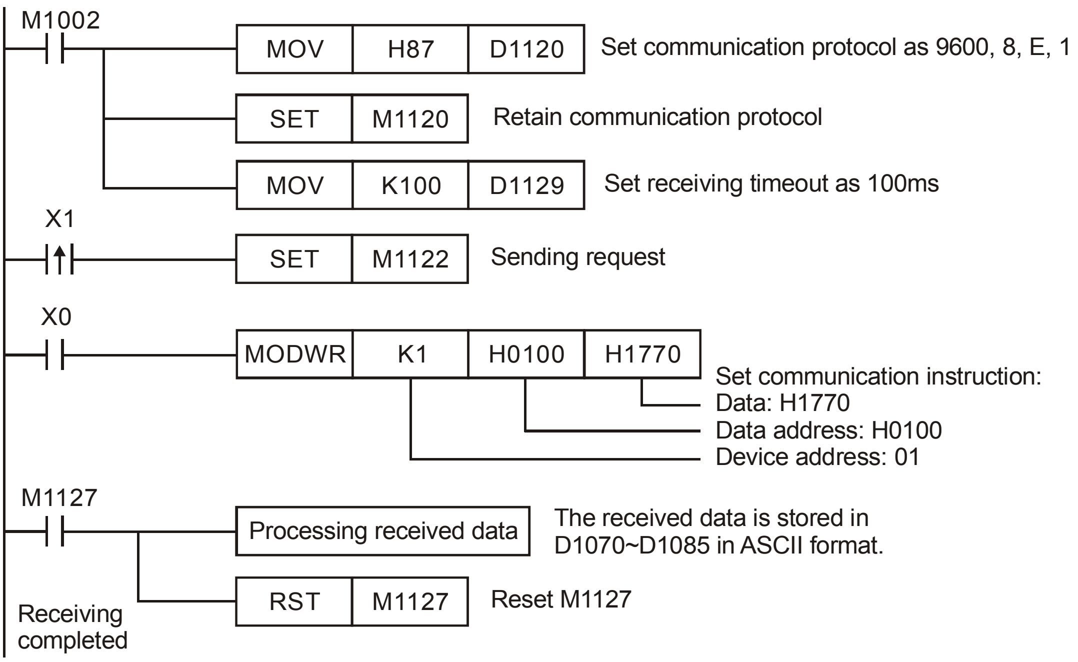
S_1: Device address (K0~K254) **S_2**: Data address **n**: Data to be written

Explanations:

1. MODWR instruction supports COM2 (RS-485).
2. MODWR is an instruction exclusively for peripheral communication equipment in MODBUS ASCII/RTU mode. The built-in RS-485 communication ports in Delta VFD drives (except for VFD-A series) are all compatible with MODBUS communication format. MODRD can be used for communication (write data) of Delta drives.
3. If the address of **S_2** is illegal for the designed communication device, the device will respond with an error, PLC will record the error code in D1130 and M1141 will be ON. For example, if 8000H is invalid to VFD-B, M1141 will be ON and D1130 = 2. For error code explanations, please see the user manual of VFD-B.
4. The feedback (returned) data from the peripheral equipment will be stored in D1070 ~ D1085. After data receiving is completed, PLC will check the validity of the data automatically. If there is an error, M1140 will be ON
5. If peripheral device receives a correct record (data) from PLC after M1140/M1141 = ON, the peripheral device will send out feedback data and PLC will reset M1140/M1141 after the validity of data is confirmed.
6. There is no limitation on the times of using this instruction, but only one instruction can be executed at a time on the same COM port.
7. If rising-edge contacts (LDP, ANDP, ORP) or falling-edge contacts (LDF, ANDF, ORF) is used before MODWR instruction, sending request flag M1122 has to be executed as a requirement.
8. For associated flags and special registers, please refer to **Points to note** of API 80 RS instruction

Program Example 1:

Communication between PLC and VFD-B series AC motor drives (ASCII Mode, M1143 = OFF)



PLC → VFD-B, PLC transmits: “01 06 0100 1770 71 ”

VFD-B → PLC, PLC receives: “01 06 0100 1770 71 ”

Registers for data to be sent (sending messages)

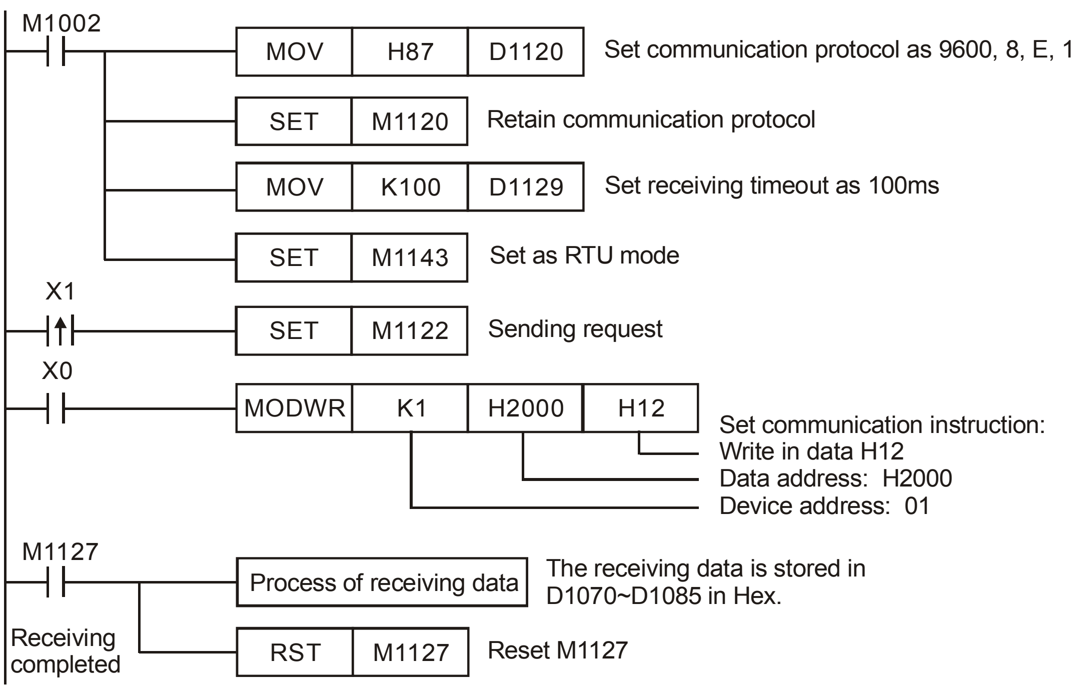
Register	Data		Descriptions	
D1089 low	‘0’	30 H	ADR 1	Address of AC motor drive: ADR (1,0)
D1089 high	‘1’	31 H	ADR 0	
D1090 low	‘0’	30 H	CMD 1	Command code of AC motor drive: CMD (1,0)
D1090 high	‘6’	36 H	CMD 0	
D1091 low	‘0’	30 H	Data address	
D1091 high	‘1’	31 H		
D1092 low	‘0’	30 H		
D1092 high	‘0’	30 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	‘7’	37 H	LRC CHK 1	Checksum: LRC CHK (0,1)
D1095 high	‘1’	31 H	LRC CHK 0	

Registers for received data (responding messages)

Register	Data		Descriptions
D1070 low	'0'	30 H	ADR 1
D1070 high	'1'	31 H	ADR 0
D1071 low	'0'	30 H	CMD 1
D1071 high	'6'	36 H	CMD 0
D1072 low	'0'	30 H	Data address
D1072 high	'1'	31 H	
D1073 low	'0'	30 H	
D1073 high	'0'	30 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	'7'	37 H	LRC CHK 1
D1076 high	'1'	31 H	LRC CHK 0

Program Example 2:

Communication between PLC and VFD-B series AC motor drives (RTU Mode, M1143 = ON)



PLC → VFD-B, PLC transmits: 01 06 2000 0012 02 07

VFD-B → PLC, PLC receives: 01 06 2000 0012 02 07

Registers for data to be sent (sending messages)

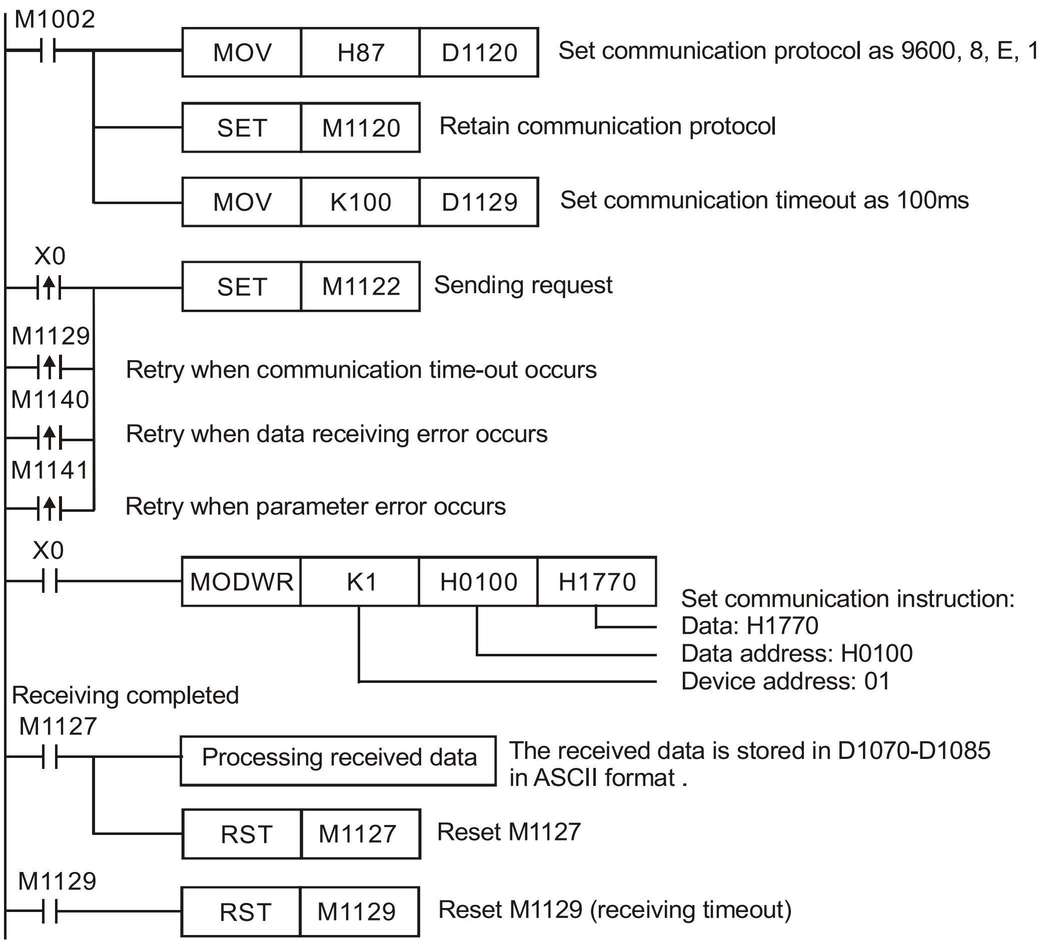
Register	Data	Descriptions
D1089 low	01 H	Address of AC motor drive
D1090 low	06 H	Command code of AC motor drive
D1091 low	20 H	Data address
D1092 low	00 H	
D1093 low	00 H	Data content
D1094 low	12 H	
D1095 low	02 H	CRC CHK Low
D1096 low	07 H	CRC CHK High

Registers for received data (responding messages)

Register	Data	Descriptions
D1070 low	01 H	Address of AC motor drive
D1071 low	06 H	Command code of AC motor drive
D1072 low	20 H	Data address
D1073 low	00 H	
D1074 low	00 H	Data content
D1075 low	12 H	
D1076 low	02 H	CRC CHK Low
D1077 low	07 H	CRC CHK High

Program Example 3:

1. In the communication between PLC and VFD-B series AC motor drive (ASCII Mode, M1143 = OFF), executes Retry when communication time-out, data receiving error or parameter error occurs
2. When X0 = ON, PLC will write data H1770 (K6000) into address H0100 in device 01 (VFD-B).
3. M1129 will be ON when communication time-out occurs. The program will trigger M1129 and send request for reading the data again.
4. M1140 will be ON when data receiving error occurs. The program will trigger M1140 and send request for reading the data again.
5. M1141 will be ON when parameter error occurs. The program will trigger M1141 and send request for reading the data again.



API	Mnemonic	Operands	Function	Controllers
102	FWD	(S1) (S2) (n)	Forward Operation of VFD	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	FWD: 7 steps
S_1					*	*							*			
S_2					*	*							*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

API	Mnemonic	Operands	Function	Controllers
103	REV	(S1) (S2) (n)	Reverse Operation of VFD	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	REV: 7 steps
S_1					*	*							*			
S_2					*	*							*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

API	Mnemonic	Operands	Function	Controllers
104	STOP	(S1) (S2) (n)	Stop VFD	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	STOP: 7 steps
S_1					*	*							*			
S_2					*	*							*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Device address **S_2**: Operation frequency of VFD **n**: Operation mode

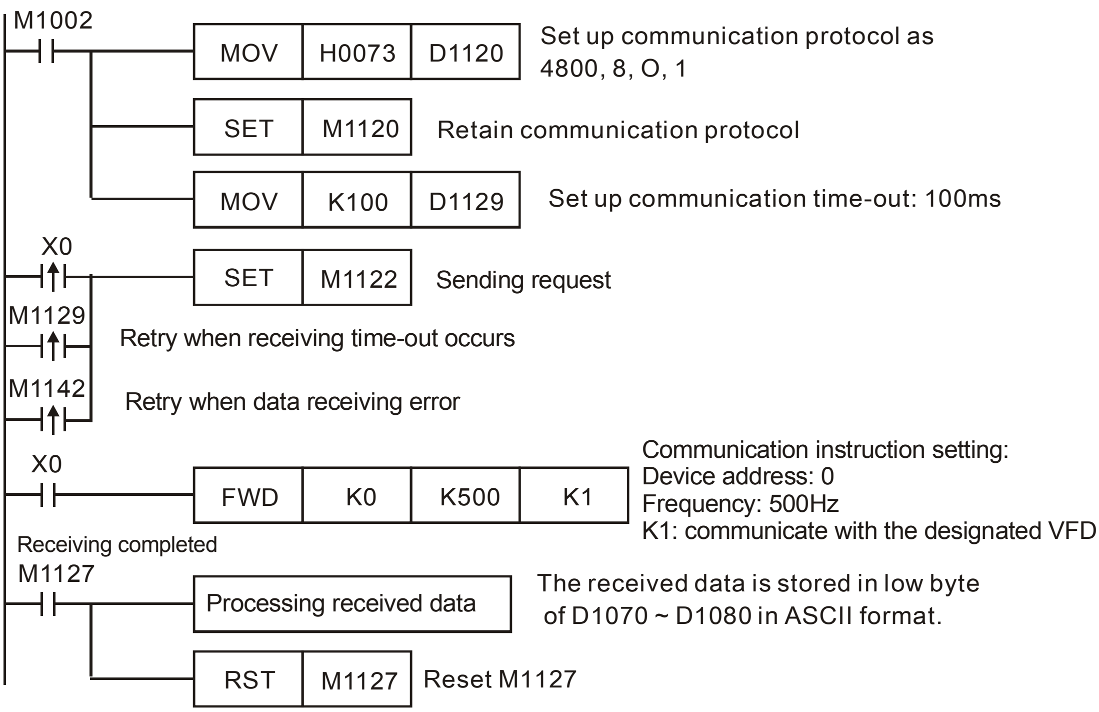
Explanations:

1. M1177 = OFF (Default), FWD, REV, STOP instructions support COM2(RS-485).
2. M1177= ON, FWD, REV, STOP instructions support COM2(RS-485), COM3(RS-485).
3. M1177 has to be set up in advance for selecting the target model of VFD. When M1177 = OFF (Default), FWD, REV, STOP instructions support Delta's VFD-A inverter. When M1177 = ON, these instructions support other models of VFD inverters, e.g. VFD-B, VFD.
4. There is no limitation on the times of using FWD, REV, STOP instruction, however only one instruction can be executed on single COM port at a time.

5. If rising-edge (LDP, ANDP, ORP) or falling-edge (LDF, ANDF, ORF) contacts are used before FWD, REV, STOP instructions, sending request flags M1122 (COM2) / M1316 (COM3) has to be enabled in advance for obtaining correct operation.
6. For detailed information of associated flags and special registers, please refer to RS instruction.
7. M1177 = OFF, only Delta VFD-A is supported and the definition of each operand is:
   a) $\mathbf{S_1}$ = Address of VFD-A. Range of $\mathbf{S_1}$: K0 ~ K31
   b) $\mathbf{S_2}$ = Operation frequency of VFD. Set value for VFD A-type inverter: K0 ~ K4,000 (0.0Hz ~ 400.0Hz).
   c) **n** = Communication mode. Range: K1 ~ K2. **n** = 1: communicate with VFD at designated address. **n** = 2: communicate with all connected VFDs. .
   d) The feedback data from the peripheral equipment will be stored in D1070 ~ D1080 After data receiving is completed, PLC will check if all data are correct automatically. If there is an error, M1142 will be ON. When **n** = 2, PLC will not receive any data.

Program Example: COM2 (RS-485)

1. Communication between PLC and VFD-A series inverter. Retry for communication time-out and data receiving error.



PLC ⇨ VFD-A, PLC sends: "**C ♥ ☺ 0001 0500** "

VFD-A ⇨ PLC, PLC receives: "**C ♥ ♠ 0001 0500** "

Registers for data to be sent (sending messages)

Register	Data		Descriptions
D1089 low	'C'	43 H	Header of control string
D1090 low	'♥'	03 H	Checksum
D1091 low	'☺'	01 H	Command acknowledgement (communication mode)
D1092 low	'0'	30 H	Communication address
D1093 low	'0'	30 H	
D1094 low	'0'	30 H	
D1095 low	'1'	31 H	
D1096 low	'0'	30 H	Operation command
D1097 low	'5'	35 H	
D1098 low	'0'	30 H	
D1099 low	'0'	30 H	

Registers for received data (responding messages)

Register	DATA		Explanation
D1070 low	'C'	43 H	Header of control string
D1071 low	'♥'	03 H	Checksum
D1072 low	'♠'	06 H	Acknowledge back. (Check feedback data) (correct: 06H, Error: 07 H)
D1073 low	'0'	30 H	Communication address
D1074 low	'0'	30 H	
D1075 low	'0'	30 H	
D1076 low	'1'	31 H	
D1077 low	'0'	30 H	Operation command
D1078 low	'5'	35 H	
D1079 low	'0'	30 H	
D1080 low	'0'	30 H	

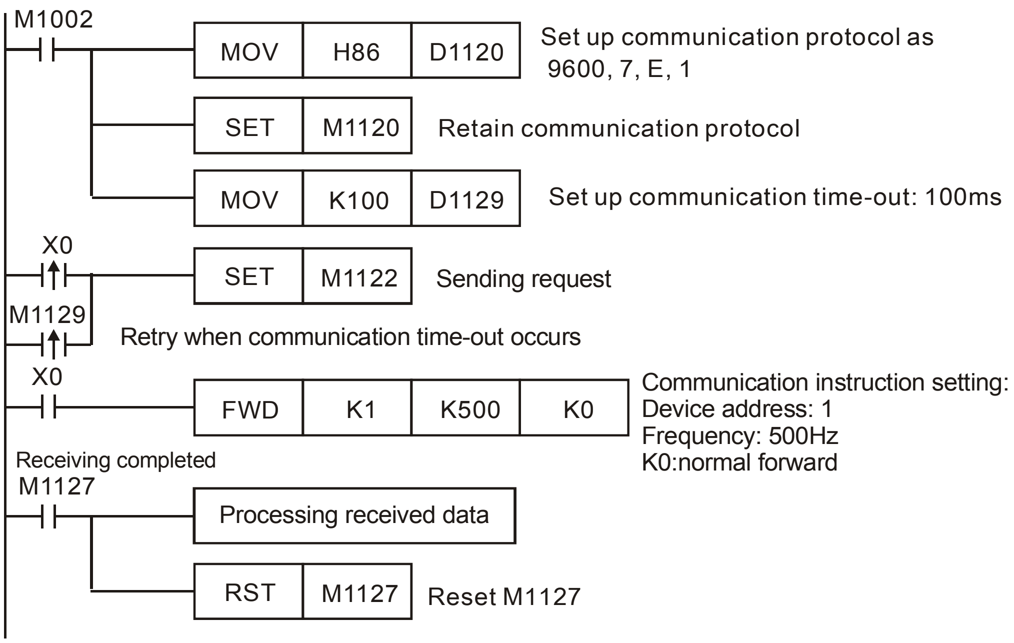
2. M1177 = ON, other Delta VFDs are supoported
   a) **S_1** = Address of VFD-A. Range of **S_1**: K0 ~ K255, when **S_1** is specified as K0, PLC will broadcast to all connected VFDs.
   b) **S_2** = Running frequency of VFD. Please refer to manuals of specific VFD. In STOP instruction, operand **S_2** is reserved.
   c) **n =** Operation mode.
      - In FWD instruction: **n** = 0 → Forward mode; **n** = 1 → Forward JOG. Other values will be regarded as normal forward mode.
      - In REV instruction: **n** = 0 → Reverse mode; **n** = 1 → Reverse JOG. Other values will be regarded as normal reverse mode

- In STOP instruction: operand **n** is reserved.

d) When Forward JOG is selected in FWR instruction, set value in **S_2** is invalid. If users need to modify the JOG frequency, please refer to manuals of specific VFDs.

Program Example: COM2 (RS-485)

Communication between PLC and VFD-B series inverter (ASCII Mode, M1143 = OFF), Retry when communication time-out occurs.



PLC ⇨ VFD, PLC sends: “:**01 10 2000 0002 04 0012 01F4 C2** ”

VFD ⇨ PLC, PLC sends: “:**01 10 2000 0002 CD** ”

Data to be sent (sending messages)

Data		Descriptions	
‘0’	30 H	ADR 1	Address of AC motor drive: ADR (1,0)
‘1’	31 H	ADR 0	
‘1’	31 H	CMD 1	Command code: CMD (1,0)
‘0’	30 H	CMD 0	
‘2’	32 H	Data Address	
‘0’	30 H		
‘0’	30 H		
‘0’	30 H		
‘0’	30 H	Data content	
‘0’	30 H		
‘0’	30 H		
‘2’	32 H		

'0'	30 H	Byte Count	
'4'	34 H		
'0'	30H	Data content 1	H1: forward operation
'0'	30 H		
'1'	31 H		
'2'	32 H		
'0'	30 H	Data content 2	Operation frequency = K500Hz H01F4
'1'	31 H		
'F'	46 H		
'4'	34 H		
'C'	43 H	LRC CHK 1	Error checksum: LRC CHK (0,1)
'2'	32 H	LRC CHK 0	

Received data (responding messages)

Data		Descriptions
'0'	30 H	ADR 1
'1'	31 H	ADR 0
'1'	31 H	CMD 1
'0'	30 H	CMD 0
'2'	32 H	Data Address
'0'	30 H	
'0'	30 H	
'0'	30 H	
'0'	30 H	Number of Register
'0'	30 H	
'0'	30 H	
'2'	32 H	
'C'	43 H	LRC CHK 1
'D'	44 H	LRC CHK 0

API	Mnemonic	Operands	Function	Controllers
105	RDST	(S) (n)	Read VFD Status	ES2/EX2 | SS2 | SA2 | SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
																RDST: 5 steps
S					*	*							*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Device address **n**: Status content to be retrieved

Explanations:

1. M1177 = OFF (Default), RDST instruction supports COM2(RS-485).
2. M1177= ON, RDST instruction supports COM2(RS-485), COM3(RS-485).
3. M1177 has to be set up in advance for selecting the target model of VFD. When M1177 = OFF (Default), RDST instruction supports Delta's VFD-A inverter. When M1177 = ON, the instruction supports other models of VFD inverters, e.g. VFD-B, VFD.
4. There is no limitation on the times of using RDST instruction, however only one instruction can be executed on single COM port at a time
5. Rising-edge contacts (LDP, ANDP, ORP) and falling-edge contacts (LDF, ANDF, ORF) can not be used with RDST instructions. Otherwise, the data in receiving registers will be incorrect.
6. For detailed information of associated flags and special registers, please refer to RS instruction.
7. M1177 = OFF, only VFD-A is supported
   a) Range of **S**: K0 ~ K31
   b) Range of **n**: K0 ~ K3
   c) **n**: Status content to be retrieved
      n=0, frequency
      n=1, output frequency
      n=2, output current
      n=3, Operation command
   d) The feedback data consists of 11 bytes (refer to VFD-A user manual), and will be stored in low bytes of D1070 ~ D1080.

"**Q, S, B, Uu, Nn, ABCD**"

Feedback	Explanation	Data storage
Q	Header of question string: 'Q' (51H).	D1070 low
S	Checksum: 03H.	D0171 low
B	Acknowledge back. Correct: 06H, Error: 07H.	D1072 low
U	Communication address (range: 00~31). Displayed in ASCII format.	D1073 low
U		D1074 low
N	Status content to be retrieved (00 ~ 03). Displayed in ASCII format.	D1075 low
N		D1076 low

Feedback	Explanation	Data storage
A	Retrieved status content. The content of "ABCD" differs according to value 00~03 set in NN. 00 ~ 03 indicates frequency, current and operation mode respectively. Please refer to the explanations below for details.	D1077 low
B		D1078 low
C		D1079 low
D		D1080 low
	Nn = "00" Frequency command = ABC.D (Hz) Nn = "01" Output frequency = ABC.D (Hz) Nn = "02" Output current = ABC.D (A) PLC will automatically convert the ASCII characters "ABCD" into D1050. For example, "ABCD" = "0600", PLC will convert ABCD into K0600 (0258 H) and store it in the special register D1050.	
	Nn = "03" Operation command	

'A' =

'0'	Stop,	'5'	JOG (forward)
'1'	Forward operation	'6'	JOG (reverse)
'2'	Stop,	'7'	JOG (reverse)
'3'	Reverse operation	'8'	Abnormal
'4'	JOG (forward),		

PLC will automatically convert the ASCII character in "A" into D1051. For example, "A" = "3", PLC will convert A into K3 and store it in the special register D1051.

'B' =

b7	b6	b5	b4	Frequency reference source
0	0	0	0	Digital keypad
0	0	0	1	1st Step Speed
0	0	1	0	2nd Step Speed
0	0	1	1	3rd Step Speed
0	1	0	0	4th Step Speed
0	1	0	1	5th Step Speed
0	1	1	0	6th Step Speed
0	1	1	1	7th Step Speed
1	0	0	0	JOG frequency
1	0	0	1	Analog input frequency command
1	0	1	0	RS-485 communication interface
1	0	1	1	Up/Down control

b3	=	0	Non-DC braking stop	1	DC braking stop
b2	=	0	Non-DC braking start	1	DC braking start
b1	=	0	Forward	1	Reverse
b0	=	0	Stop	1	Run

PLC will store bit status of "B" in special auxiliary relay M1168 (b0) ~ M1175 (b7).

"CD" =

"00"	No error	"10"	OcA
"01"	oc	"11"	Ocd
"02"	ov	"12"	Ocn
"03"	oH	"13"	GFF
"04"	oL	"14"	Lv
"05"	oL1	"15"	Lv1
"06"	EF	"16"	cF2
"07"	cF1	"17"	bb
"08"	cF3	"18"	oL2
"09"	HPF	"19"	

PLC will automatically convert the ASCII characters in "CD" into D1052. For example, "CD" = "16", PLC will convert CD into K16 and store it in the special register D10512

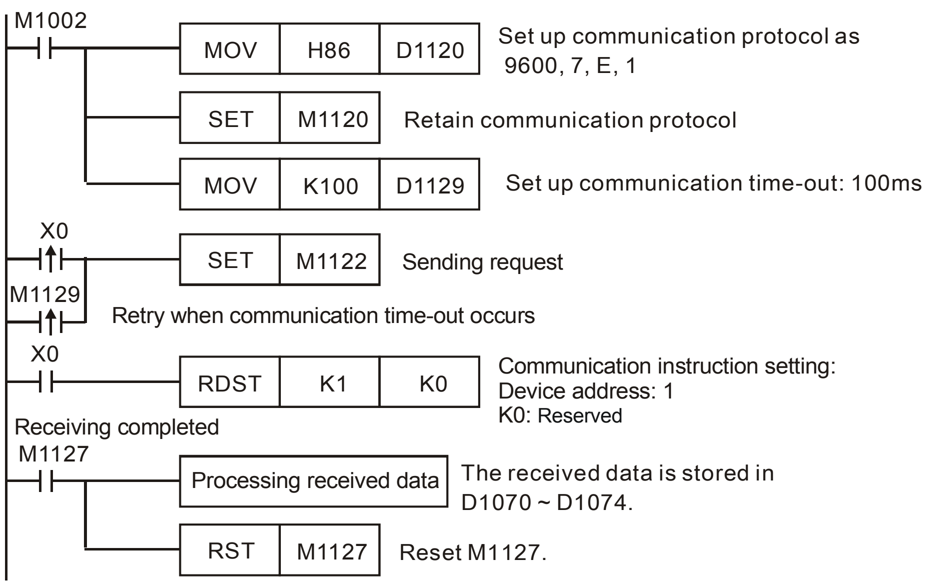
8. M1177 = ON, other Delta VFDs are supoported

a) Range of **S_1**: K1 ~ K255

b) The instruction will read VFD status at parameter address 2100H~2104H (Please refer to user manual of specific VFD for details.) and store the feedback data in D1070~D1074. However, the content in D1070~D1074 will not be updated when receiving error or timeout occurs. Therefore, please check the status of receiving completed flag before applying the received data

Program Example: COM2 (RS-485)

1. Communication between PLC and VFD-B series inverter (ASCII Mode, M1143 = OFF). Retry when communication time-out occurs.
2. Read VFD status at parameter address 2100H~2104H and store the received data in D1070 ~ D1074.



PLC ⇨ VFD-B, PLC sends: "**:01 03 2100 0005 D6** "

VFD-B ⇨ PLC, PLC receives: ":01 03 0A 00C8 7C08 3E00 93AB 0000 2A "

Data to be sent (sending messages)

Data		Descriptions	
'0'	30 H	ADR 1	AC drive address : ADR (1,0)
'1'	31 H	ADR 0	
'0'	30 H	CMD 1	Command code: CMD (1,0)
'3'	33 H	CMD 0	
2'	32 H	Starting data address	
'1'	31 H		
'0'	30 H		
'0'	30 H		

Data		Descriptions	
'0'	30 H	Number of data (count by word)	
'0'	30 H		
'0'	30 H		
'5'	35 H		
'D'	44 H	LRC CHK 1	Error checksum: LRC CHK (0,1)
'6'	36 H	LRC CHK 0	

Received data (responding messages)

Data		Descriptions	
'0'	30 H	ADR 1	
'1'	31 H	ADR 0	
'0'	30 H	CMD 1	
'3'	33 H	CMD 0	
'0'	30 H	Number of data (count by byte)	
'A'	41 H		
'0'	30 H	Content of address 2100 H	PLC automatically converts ASCII codes and store the converted value in D1070 = 00C8 H
'0'	30 H		
'C'	43 H		
'8'	38 H		
'7'	37 H	Content of address 2101 H	PLC automatically converts ASCII codes and store the converted value in D1071 = 7C08 H
'C'	43 H		
'0'	30 H		
'8'	38 H		
'3'	33 H	Content of address 2102 H	PLC automatically converts ASCII codes and store the converted value in D1072 = 3E00 H
'E'	45 H		
'0'	30 H		
'0'	30 H		
'9'	39 H	Content of address 2103H	PLC automatically converts ASCII codes and store the converted value in D1073 = 93AB H
'3'	33 H		
'A'	41 H		
'B'	42 H		
'0'	30 H	Content of address 2104 H	PLC automatically converts ASCII codes and store the converted value in D1074 = 0000 H
'0'	30 H		
'0'	30 H		
'0'	30 H		
'2'	32 H	LRC CHK 1	
'A'	41 H	LRC CHK 0	

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
106	RSTEF	S n	Reset Abnormal VFD				

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*							*			RSTEF: 5 steps
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

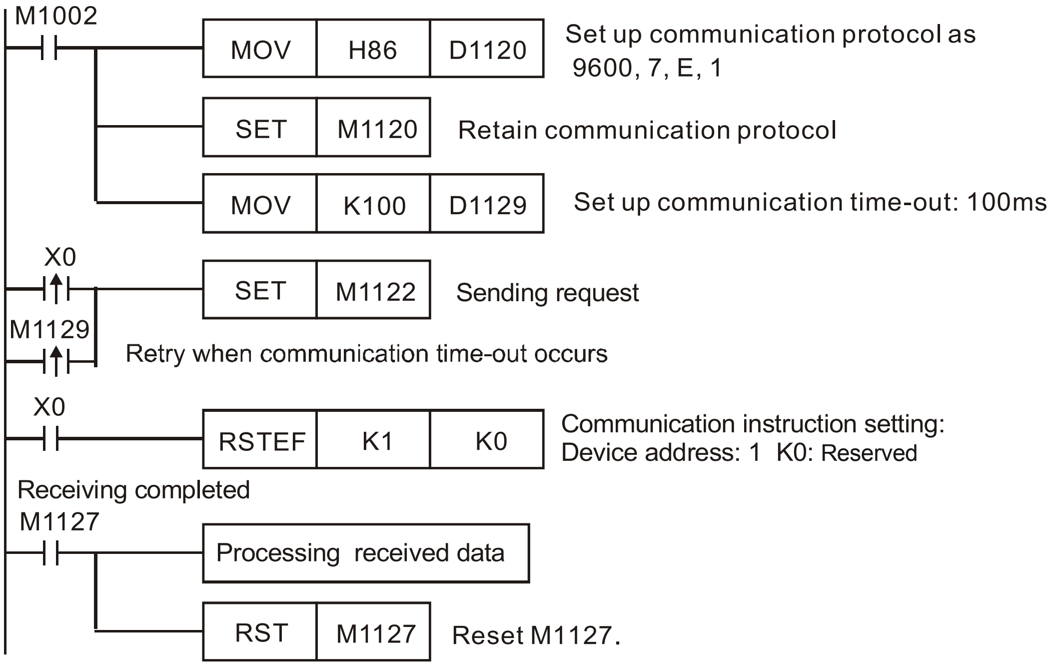
S: Address of communication device **n**: Operation mode

Explanations:

1. M1177 = OFF (Default), RSTEF instruction supports COM2(RS-485).
2. M1177= ON, RSTEF instruction supports COM2(RS-485), COM3(RS-485).
3. M1177 has to be set up in advance for selecting the target model of VFD. When M1177 = OFF (Default), RSTEF instruction supports Delta's VFD-A inverter. When M1177 = ON, these instructions support other models of VFD inverters, e.g. VFD-B, VFD.
4. There is no limitation on the times of using RSTEF instruction, however only one instruction can be executed on single COM port at a time.
5. If rising-edge (LDP, ANDP, ORP) or falling-edge (LDF, ANDF, ORF) contacts are used before RSTEF instruction, sending request flags M1122 (COM2) / M1316 (COM3) has to be enabled in advance for obtaining correct operation.
6. For detailed information of associated flags and special registers, please refer to RS instruction.
7. M1177 = OFF, only Delta VFD-A is supported and the definition of each operand is:
   a) $\mathbf{S_1}$ = Address of VFD-A. Range of $\mathbf{S_1}$: K0 ~ K31
   b) **n** = Communication mode. Range: K1 ~ K2. **n** = 1: communicate with VFD at designated address. **n** = 2: communicate with all connected VFDs. .
   c) RSTEF is a handy communication instruction used for reset when errors occur in AC motor drive operation.
   d) The feedback data from the peripheral equipment will be stored in D1070 ~ D1080. When **n** = 2, PLC will not receive any data.
8. M1177 = ON, other Delta VFDs are supoported
   - $\mathbf{S_1}$ = Address of VFD. Range of $\mathbf{S_1}$: K0 ~ K255, when $\mathbf{S_1}$ is specified as K0, PLC will broadcast to all connected VFDs

Program Example: COM2 (RS-485)

Communication between PLC and VFD-B series AC motor drives (ASCII Mode, M1143 = OFF). Retry when communication time-out occurs.



PLC ⇨ VFD, PLC sends: “:**01 06 2002 0002 D5** ”

VFD ⇨ PLC, PLC sends: “:**01 06 2002 0002 D5** ”

Data to be sent (sending messages):

Data		Descriptions	
‘0’	30 H	ADR 1	AC drive address : ADR (1,0)
‘1’	31 H	ADR 0	
‘0’	30 H	CMD 1	Command code: CMD (1,0)
‘6’	36 H	CMD 0	
‘2’	32 H	Data address	
‘0’	30 H		
‘0’	30 H		
‘2’	32 H		
‘0’	30 H	Data contents	
‘0’	30 H		
‘0’	30 H		
‘2’	32 H		
‘D’	44 H	LRC CHK 1	Error checksum: LRC CHK (0,1)
‘5’	35 H	LRC CHK 0	

Received data (responding messages)

Data		Descriptions
‘0’	30 H	ADR 1
‘1’	31 H	ADR 0
‘0’	30 H	CMD 1
‘6’	36 H	CMD 0
‘2’	32 H	Data address
‘0’	30 H	
‘0’	30 H	
‘2’	32 H	
‘0’	30 H	Data content
‘0’	30 H	
‘0’	30 H	
‘2’	32 H	
‘D’	44 H	LRC CHK 1
‘5’	35 H	LRC CHK 0

API	Mnemonic		Operands	Function	Controllers
107	LRC	P	S n D	LRC checksum	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Starting device for ASCII mode checksum **n**: Data length for LRC operation (**n** = K1~K256)

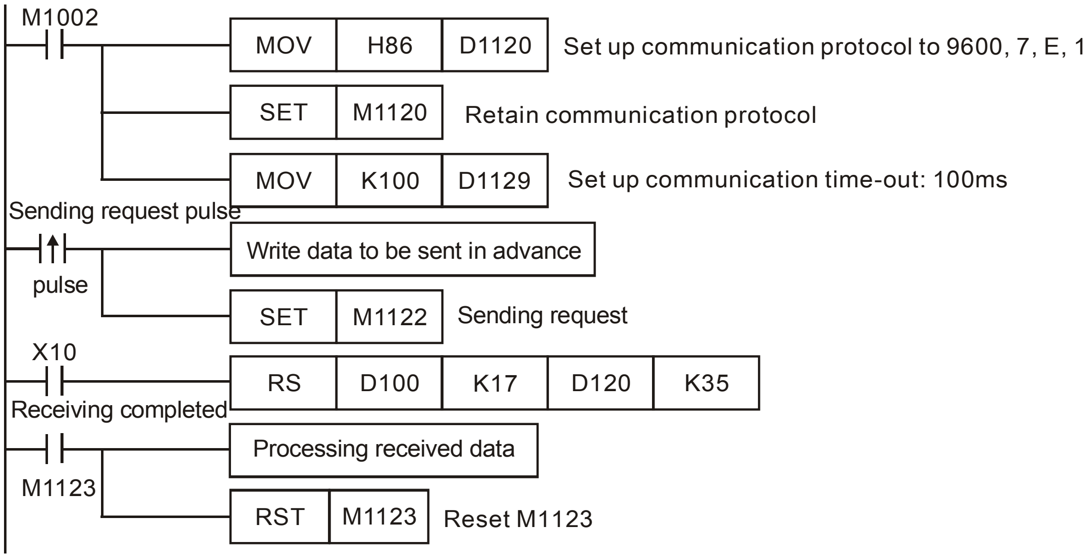
D: Starting device for storing the operation result

Explanations:

1. **n**: **n** must be an even number. If **n** is out of range, an error will occur and the instruction will not be executed. At this time, M1067 and M1068 = ON and error code H'0E1A will be recorded in D1067.
2. 16-bit mode: When LRC instruction operates with M1161 = OFF, hexadecimal data starting from **S** is divided into high byte and low byte and the checksum operation is operated on **n** number of bytes. After this, operation result will be stored in both hi-byte and low byte of **D**.
3. 8-bit mode: When LRC instruction operates with M1161 = ON, hexadecimal data starting from **S** is divided into high byte (invalid) and low byte and the checksum operation is operated on **n** number of low bytes. After this, operation result will be stored in low bytes of **D** (Consecutive 2 registers).
4. Flag: M1161 8/16-bit mode

Program Example:

Connect PLC to VFD series AC motor drive (ASCII mode, M1143 = OFF), (8-bit mode, M1161 = ON), Write the data to be sent into registers starting from D100 in advance for reading 6 data from address H0708 on VFD.

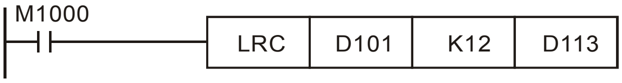


PLC ⇨ VFD, PLC sends: **": 01 03 07 08 0006 E7 CR LF "**

Registers for sent data (sending messages)

Register	Data		Explanation	
D100 low byte	': '	3A H	STX	
D101 low byte	'0'	30 H	ADR 1	Address of AC motor drive: ADR (1,0)
D102 low byte	'1'	31 H	ADR 0	
D103 low byte	'0'	30 H	CMD 1	Command code: CMD (1,0)
D104 low byte	'3'	33 H	CMD 0	
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 (words)	
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	Error checksum: LRC CHK (0,1)
D114 low byte	'7'	37 H	LRC CHK 1	
D115 low byte	CR	D H	END	
D116 low byte	LF	A H		

The error checksum LRC CHK (0, 1) can be calculated by LRC instruction (8-bit mode, M1161 = ON).



LRC checksum: 01 H + 03 H + 07 H + 08 H + 00 H + 06 H = 19 H. Operate 2's complement on 19H and the result is E7H. Store 'E'(45 H) in the low byte of D113 and '7' (37 H) in the low byte of D114.

Remarks:

ASCII mode communication data:

STX	':'	Start word = ':' (3AH)
Address Hi	'0'	Communication: 8-bit address consists of 2 ASCII codes
Address Lo	'1'	
Function Hi	'0'	Function code: 8-bit function consists of 2 ASCII codes
Function Lo	'3'	
DATA (n-1) ……. DATA 0	'2'	Data content: n × 8-bit data consists of 2n ASCII codes
	'1'	
	'0'	
	'2'	
	'0'	
	'0'	
	'0'	
	'2'	
LRC CHK Hi	'D'	LRC checksum: 8-bit checksum consists of 2 ASCII codes
LRC CHK Lo	'7'	
END Hi	CR	End word: END Hi = CR (0DH), END Lo = LF(0AH)
END Lo	LF	

LRC checksum: Operate 2's complement on the summed up value from communication address to the end of data, i.e. 01 H + 03 H + 21 H + 02 H + 00 H + 02 H = 29 H, the operation result of 29H is D7H.

API	Mnemonic		Operands	Function	Controllers
108	CRC	P	S n D	CRC checksum	ES2/EX2 SS2 SA2 SX2

OP \ 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
S													*			
n					*	*							*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

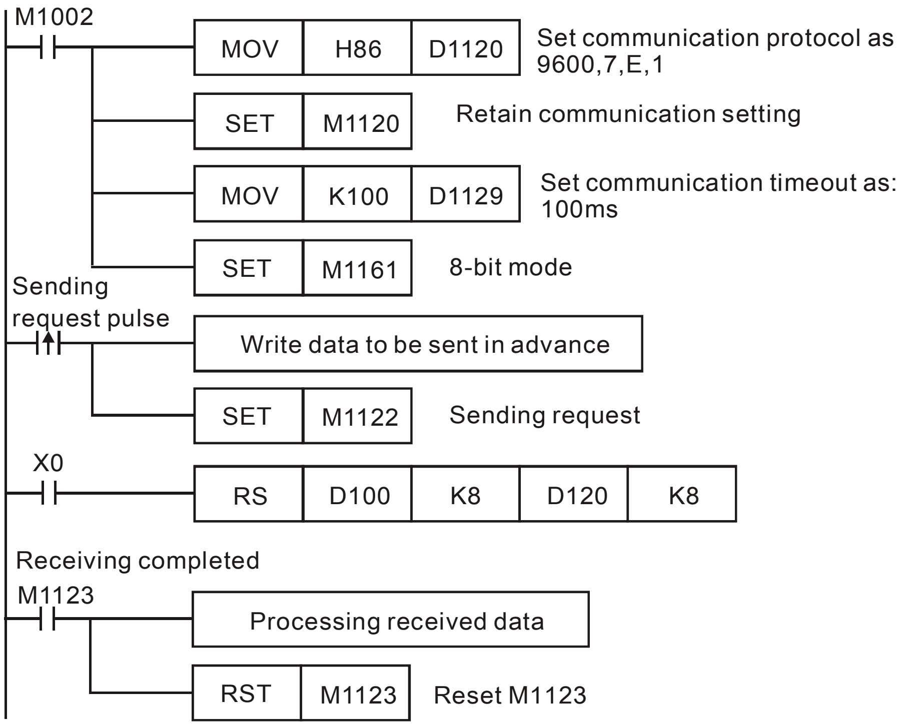
S: Starting device for RTU mode checksum **n**: Data length for CRC operation (**n** = K1~K256) **D**: Starting device for storing the operation result

Explanations:

1. **n**: **n** must be an even number. If **n** is out of range, an error will occur and the instruction will not be executed. At this time, M1067 and M1068 = ON and error code H'0E1A will be recorded in D1067.
2. 16-bit mode: When CRC instruction operates with M1161 = OFF, hexadecimal data starting from **S** is divided into high byte and low byte and the checksum operation is operated on **n** number of bytes. After this, operation result will be stored in both hi-byte and low byte of **D**.
3. 8-bit mode: When CRC instruction operates with M1161 = ON, hexadecimal data starting from **S** is divided into high byte (invalid) and low byte and the checksum operation is operated on **n** number of low bytes. After this, operation result will be stored in low bytes of **D** (Consecutive 2 registers).
4. Flag: M1161 8/16-bit mode

Program Example:

Connect PLC to VFD series AC motor drive (RTU mode, M1143 = ON), (8-bit mode, M1161 = ON), Write the data to be sent (H1770) into address H0706 on VFD.

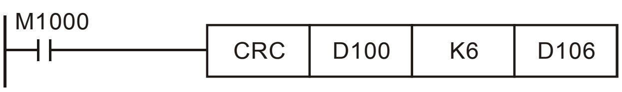


PLC ⇨ VFD, PLC sends: **01 06 0706 1770 66 AB**

Registers for sent data (sending messages)

Register	Data	Explanation
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 error checksum CRC CHK (0,1) can be calculated by CRC instruction (8-bit mode, M1161 = ON).



CRC checksum: 66 H is stored in low byte of D106 and AB H in low byte of of D107,

API	Mnemonic			Operands	Function	Controllers
110	D	ECMP	P	S1 S2 D	Floating point compare	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*							*			DECMP, DECMPP: 13 steps
S_2					*	*							*			
D		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

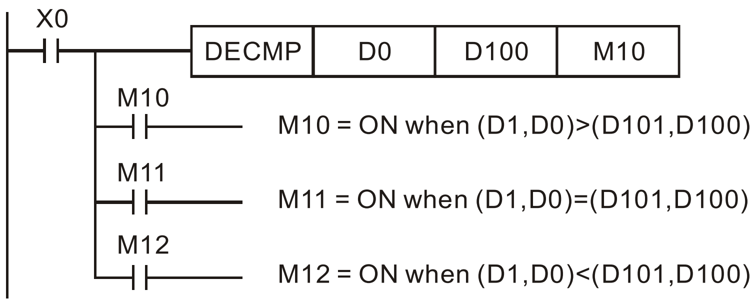
S_1: 1st comparison value **S_2**: 2nd comparison value **D**: Comparison result, 3 consecutive devices

Explanations:

1. The data of **S_1** is compared to the data of **S_2** and the result (>, =, <) is indicated by three bit devices in **D**.
2. If the source operand **S_1** or **S_2** is specified as constant K or H, the integer value will automatically be converted to binary floating point for comparison.

Program Example:

1. If the specified device is M10, M10~M12 will automatically be used.
2. When X0 = ON, one of M10~M12 will be 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 for achieving the results of ≧, ≦, ≠.
4. RST or ZRST instruction is required if users need to reset the comparison result.



API	Mnemonic			Operands	Function	Controllers
111	D	EZCP	P	S1 S2 S D	Floating point zone compare	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*							*			DEZCP, DEZCPP: 17 steps
S_2					*	*							*			
S					*	*							*			
D		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

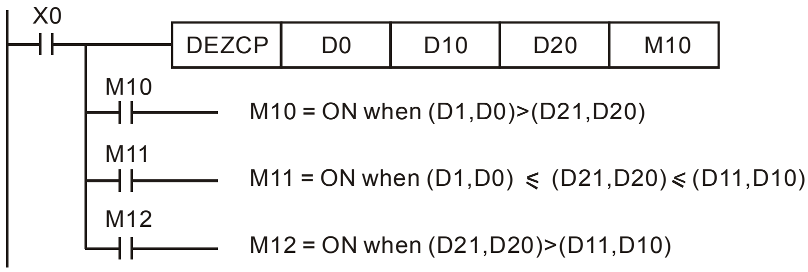
S_1: Lower bound of zone comparison **S_2**: Upper bound of zone comparison **S**: Comparison value **D**: Comparison result, 3 consecutive devices

Explanations:

1. The data of **S** is compared to the data range of **S_1** ~ **S_2** and the result (>, =, <) is indicated by three bit devices in **D**.
2. If the source operand **S_1** or **S_2** is specified as constant K or H, the integer value will automatically be converted to binary floating point for comparison.
3. Operand **S_1** should be smaller than operand **S_2**. When **S_1** > **S_2**, the instruction takes **S_1** as the 1st comparison value and performs normal comparison similar to ECMP instruction.

Program Example:

1. If the specified device is M10, M10~M12 will automatically be used.
2. When X0 = ON, one of M10~M12 will be ON. When X0 = OFF, DEZCP instruction is not executed, M10~M12 will retain their previous state before X0 = OFF.
3. RST or ZRST instruction is required if users need to reset the comparison result.



API	Mnemonic			Operands	Function	Controllers
112	D	MOVR	P	S D	Move floating point data	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device **D**: Destination device

Explanations:

1. Directly input floating point value in **S**.
2. When the instruction executed, content of **S** will be moved to **D**.

Program Example:

When X0 = OFF, D10 and D11 will not change. When X0 = ON, transmit F1.200E+0 (Input F1.2, and scientific notation F1.200E+0 will be displayed on ladder diagram. Users can set monitoring data format as float on the function View) to D10 and D11.

API	Mnemonic			Operands	Function	Controllers
116	D	RAD	P	(S) (D)	Degree → Radian	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*							*			DRAD, DRADP: 9 steps
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device (degree) **D**: Conversion result (radian)

Explanation:

1. Use the following formula to convert degree to radian:

   Radian = degree × (π/180)

2. Flags: M1020 Zero flag, M1021 Borrow flag, M1022 Carry flag

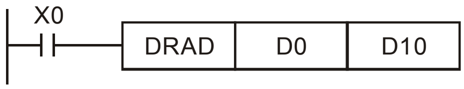
   If the absolute value of the result exceeds the max. floating point value, carry flag M1022 = ON.

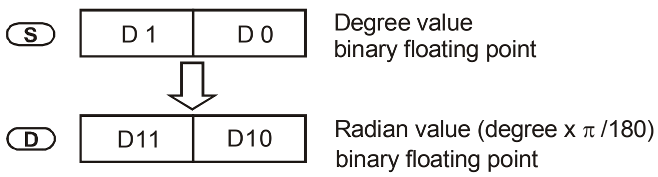
   If the absolute value of the result is less than min. floating point value, borrow flag M1021 = ON.

   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 the binary floating point result in (D11, D10).





API	Mnemonic			Operands	Function	Controllers			
						ES2/EX2	SS2	SA2	SX2
117	D	DEG	P	S D	Radian → Degree				

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*							*			DDEG, DDEGP: 9 steps
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device (radian)   **D**: Conversion result (degree)

Explanation

1. Use the following formula to convert radian to degree:

   Degree = Radian × (180/π)

Flags: M1020 Zero flag, M1021 Borrow flag and M1022 Carry flag.

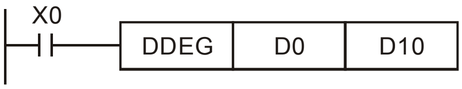
If the absolute value of the result exceeds the max. floating point value, carry flag M1022 = ON.

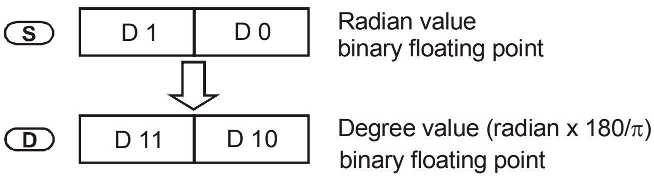
If the absolute value of the result is less than the min. floating point value, borrow flag M1021 = ON.

If the conversion result is 0, zero flag M1020 = ON.

Program Example:

When X0 = ON, convert the radian of the binary floating point in (D1, D0) to degree and save the binary floating point result in (D11, D10).





API	Mnemonic			Operands	Function	Controllers			
118	D	EBCD	P	(S) (D)	Float to scientific conversion	ES2/EX2	SS2	SA2	SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

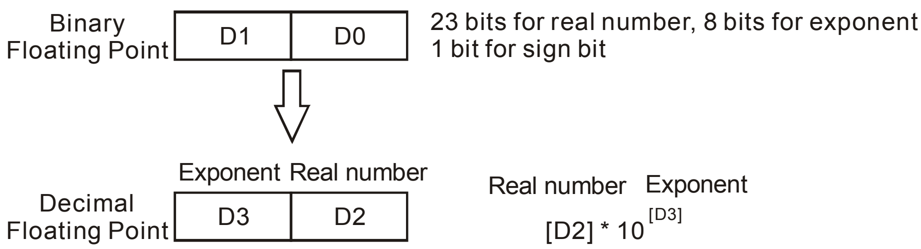
S: Source device **D**: Conversion result

Explanation

1. The instruction converts the binary floating point value in **S** to decimal floating point value and stores the results in the register specified by **D**.
2. PLC 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.
3. Flag: M1020 Zero flag, M1021 Borrow flag, M1022 Carry flag
   If absolute value of the result exceeds the max. floating point value, carry flag M1022 = ON.
   If absolute value of the result is less than the min. floating point value, 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 and the conversion result is stored in D3, D2.



API	Mnemonic			Operands	Function	Controllers
119	D	EBIN	P	(S) (D)	Scientific to float conversion	ES2/EX2 SS2 SA2 SX2

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
S													*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

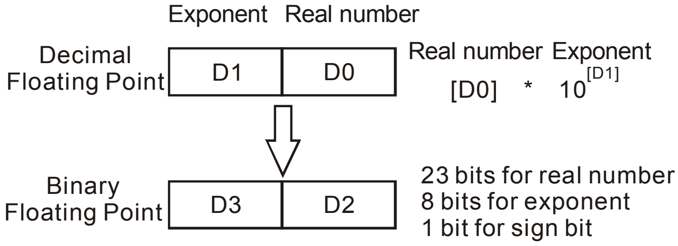
S: Source device **D**: Conversion result

Explanation:

1. The instruction converts the decimal floating point value in **S** to a binary floating point value and stores the results in the register specified by **D**.
2. For example, **S** = 1234, **S** +1 = 3. The decimal floating point value will be: 1.234 x 10^6
3. **D** must be binary floating point format. **S** and **S** +1 represent the real number and exponent of the floating point number.
4. EBIN instruction is the specific instruction used to convert decimal floating point value to binary floating point value
5. Range of real number: -9,999 ~ +9,999. Range of exponent: - 41 ~ +35. Range of PLC decimal floating point value. If the conversion result is 0, zero flag M1020 = ON.

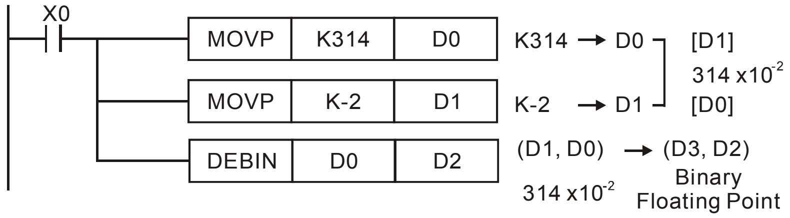
Program Example 1:

When X1 = ON, the decimal floating point value in (D1, D0) will be converted to binary floating point and the conversion result is stored in (D3, D2).



Program Example 2:

1. Use FLT instruction (API 49) to convert BIN integer into binary floating point value before performing floating point operation. The value to be converted must be BIN integer and use DEBIN instruction to convert the decimal floating point value into a binary one.
2. When X0 = ON, move K314 to D0 and K-2 to D1 to generate decimal floating point value (3.14 = 314 $\times$ 10^{-2}).


X0
MOVP
K314
D0
K314 → D0
[D1]
314 x10-2
[D0]
MOVP
K-2
D1
K-2 → D1
DEBIN
D0
D2
(D1, D0) → (D3, D2)
314 x10-2
Binary
Floating Point

API	Mnemonic			Operands	Function	Controllers			
120	D	EADD	P	S1 S2 D	Floating point addition	ES2/EX2	SS2	SA2	SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Augend **S_2**: Addend **D**: Addition result

Explanations:

1. **S_1** + **S_2** = **D**. The floating point value in **S_1** and **S_2** are added and the result is stored in **D**.
2. If the source operand **S_1** or **S_2** is specified as constant K or H, the constant will automatically be converted to binary floating point value for the addition operation.
3. **S_1** and **S_2** can designate the same register. In this case, if the instruction is specified as "continuous execution instruction" (generally DEADDP instruction) and the drive contact is ON, the register will be added once in every scan.
4. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag)

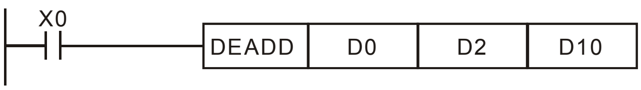
   If absolute value of the result exceeds max. floating point value, carry flag M1022 = ON.

   If absolute value of the result is less than min. floating point value, borrow flag M1021 = ON.

   If the conversion result is 0, zero flag M1020 = ON.

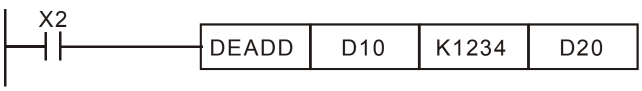
Program Example 1:

When X0 = ON, add the binary floating point value (D1, D0) with binary floating point value (D3, D2) and store the result in (D11, D10).



Program Example 2:

When X2 = ON, add the binary floating point value of (D11, D10) with K1234 (automatically converted to binary floating point value) and store the result in (D21, D20).



API	Mnemonic			Operands	Function	Controllers
121	D	ESUB	P	(S1) (S2) (D)	Floating point subtraction	ES2/EX2 SS2 SA2 SX2

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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

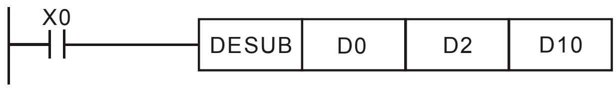
S_1: Minuend **S_2**: Subtrahend **D**: Subtraction result

Explanation:

1. **S_1 – S_2 = D**. The floating point value in **S_2** is subtracted from the floating point value in **S_1** and the result is stored in **D**. The subtraction is conducted in binary floating point format.
2. If **S_1** or **S_2** is designated as constant K or H, the instruction will convert the constant into a binary floating point value before the operation.
3. **S_1** and **S_2** can designate the same register. In this case, if the instruction is specified as "continuous execution instruction" (generally DESUBP instruction) and the drive contact is ON, the register will be subtracted once in every scan.
4. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag)
   If absolute value of the result exceeds max. floating point value, carry flag M1022 = ON.
   If absolute value of the result is less than min. floating point value, borrow flag M1021 = ON.
   If the conversion result is 0, zero flag M1020 = ON.

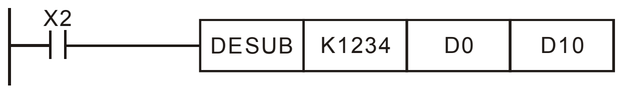
Program Example 1:

When X0 = ON, binary floating point value (D1, D0) minuses binary floating point value (D3, D2) and the result is stored in (D11, D10).



Program Example 2:

When X2 = ON, K1234 (automatically converted into binary floating point value) minuses binary floating point (D1, D0) and the result is stored in (D11, D10).



API	Mnemonic			Operands	Function	Controllers
122	D	EMUL	P	S1 S2 D	Floating point multiplication	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices												Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F		DEMUL, DEMULP: 13 steps
S_1					*	*							*				
S_2					*	*							*				
D													*				

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

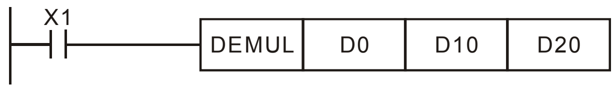
S_1: Multiplicand **S_2**: Multiplicator **D**: Multiplication result

Explanations:

1. **S_1** $\times$ **S_2** = **D**. The floating point value in **S_1** is multiplied with the floating point value in **S_2** and the result is **D**. The multiplication is conducted in binary floating point format
2. If **S_1** or **S_2** is designated as constant K or H, the instruction will convert the constant into a binary floating point value before the operation
3. **S_1** and **S_2** can designate the same register. In this case, if the instruction is specified as "continuous execution instruction" (generally DEMULP instruction) and the drive contact is ON, the register will be multiplied once in every scan.
4. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag)
   If absolute value of the result exceeds max. floating point value, carry flag M1022 = ON.
   If absolute value of the result is less than min. floating point value, borrow flag M1021 = ON.
   If the conversion result is 0, zero flag M1020 = ON.

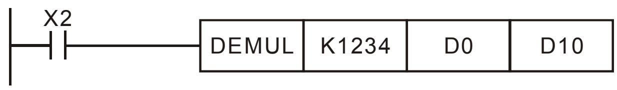
Program Example 1:

When X1 = ON, binary floating point (D1, D0) multiplies binary floating point (D11, D10) and the result is stored in (D21, D20).



Program Example 2:

When X2 = ON, K1234 (automatically converted into binary floating point value) multiplies binary floating point (D1, D0) and the result is stored in (D11, D10).



API	Mnemonic			Operands	Function	Controllers
123	D	EDIV	P	(S1) (S2) (D)	Floating point division	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
S_1					*	*							*			DEADD, DEADDP: 13 steps
S_2					*	*							*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Dividend **S_2**: Divisor **D**: Quotient and Remainder

Explanation:

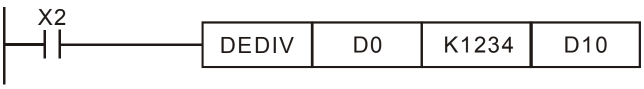
1. **S_1** ÷ **S_2** = **D**. The floating point value in **S_1** is divided by the floating point value in **S_2** and the result is stored in **D**. The division is conducted in binary floating point format.
2. If **S_1** or **S_2** is designated as constant K or H, the instruction will convert the constant into a binary floating point value before the operation.
3. If **S_2** = 0, operation error will occur, the instruction will not be executed
4. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag)
   If absolute value of the result exceeds max. floating point value, carry flag M1022 = ON.
   If absolute value of the result is less than min. floating point value, 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 value) and the result is stored in (D11, D10).



API	Mnemonic			Operands	Function	Controllers
124	D	EXP	P	(S) (D)	Float exponent operation	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

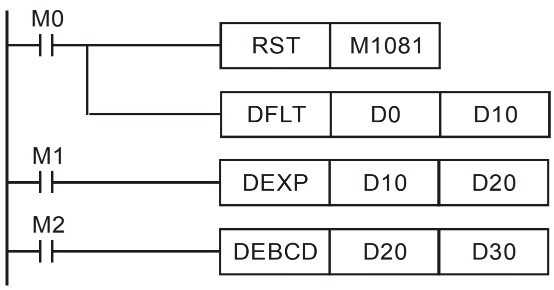
S: Exponent **D**: Operation result

Explanations:

1. The base is e = 2.71828 and exponent is **S**
2. EXP [**S +1, S**] = [**D** +1, **D**]
3. Both positive and negative values are valid for **S**. Register **D** has to be 32-bit format. Operation is conducted in floating point value, so the value in **S** needs to be converted into floating value before exponent operation.
4. The content in **D**: e^{S}, e =2.71828 and **S** is the specified exponent..
5. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag).
   If absolute value of the result is larger than max. floating value, carry flag M1022 = ON.
   If absolute value of the result is smaller than min. floating value, borrow flag M1021 = ON.
   If the conversion result is 0, zero flag M1020 = ON.

Program Example:

1. When M0 = ON, convert (D1, D0) to binary floating value and save the result in (D11, D10).
2. When M1= ON, perform exponent operation with (D11, D10) as the exponent. The value is saved in register (D21, D20) in binary floating format.
3. When M2 = ON, convert the value in (D21, D20) into decimal floating point value and save the result in (D31, D30). (At this time, D31 indicates powers of 10 for D30)



API	Mnemonic			Operands	Function	Controllers
125	D	LN	P	(S) (D)	Float natural logarithm operation	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*							*			DLN, DLNP: 9 steps
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device **D**: Operation result

Explanations:

1. Perform natural logarithm (LN) operation on operand **S**:

   LN[**S** +1, **S**]=[**D** +1, **D**]

2. Only a positive number is valid for **S**. Register **D** has to be 32-bit format. Operation is conducted in floating point value, so the value in **S** needs to be converted into floating value before natural logarithm operation.

3. e^D = **S**. The content of **D** = LN **S,** where the value in **S** is specified by users.

4. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag).

   If absolute value of the result is larger than max. floating value, carry flag M1022 = ON.

   If absolute value of the result is smaller than min. floating value, borrow flag M1021 = ON.

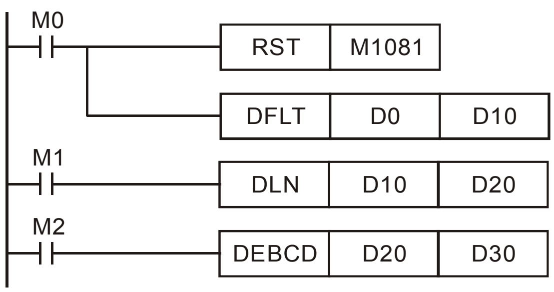
   If the conversion result is 0, zero flag M1020 = ON

Program Example:

1. When M0 = ON, convert (D1, D0) to binary floating value and save the result in (D11, D10).

2. When M1= ON, perform natural logarithm operation with (D11, D10) as the antilogarithm. The value is saved in register (D21, D20) in binary floating format.

3. When M2 = ON, convert the value in (D21, D20) into decimal floating point value and save the result in (D31, D30). (At this time, D31 indicates powers of 10 for D30)



API	Mnemonic			Operands	Function	Controllers
126	D	LOG	P	S1 S2 D	Float logarithm operation	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices X	Y	M	S	Word devices K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*							*			DLOG, DLOGP: 13 steps
S_2					*	*							*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Base **S_2**: Antilogarithm **D**: Operation result

Explanations:

1. Perform logarithm operation with **S_1** as the base and **S_2** as the antilogarithm and save the result in **D**.
2. Only a positive number is valid for **S**. Register **D** has to be 32-bit format. Operation is conducted in floating point value, so the value in **S** needs to be converted into floating value before logarithm operation.
3. Logarithm operation: $S_1^{D} = S_2$, $D = ?$ → $Log_{S_1}^{S_2} = D$

   Example: Assume $S_1 = 5$, $S_2 = 125$, $S_1^{D} = S_2$, $D = ?$ → $5^{D} = 125$ → $D = Log_{S_1}^{S_2} = log_5^{125} = 3$.
4. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag).

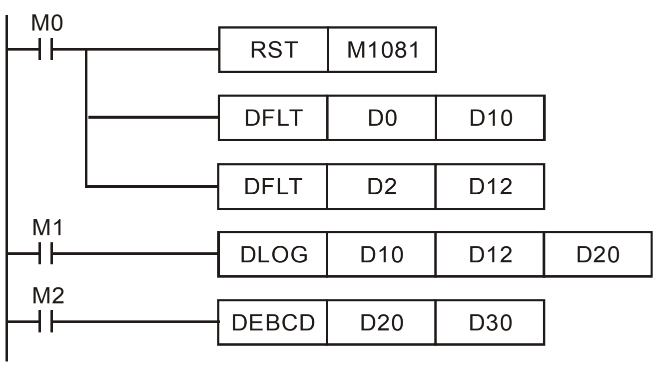
   If absolute value of the result is larger than max. floating value, carry flag M1022 = ON.

   If absolute value of the result is smaller than min. floating value, borrow flag M1021 = ON.

   If the conversion result is 0, zero flag M1020 = ON.

Program Example:

1. When M0 = ON, convert (D1, D0) and (D3, D2) to binary floating value and save the result in register (D11, D10) and (D13, D12) individually.
2. When M1= ON, perform logarithm operation with (D11, D10) as base and (D13, D12) as antilogarithm. The results are saved in register (D21, D20) in binary floating format.
3. When M2 = ON, convert the value in (D21, D20) into decimal floating point value and save the result in (D31, D30). (At this time, D31 indicates powers of 10 for D30)



API	Mnemonic			Operands	Function	Controllers
127	D	ESQR	P	(S) (D)	Floating point square root	ES2/EX2 SS2 SA2 SX2

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
S					*	*							*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

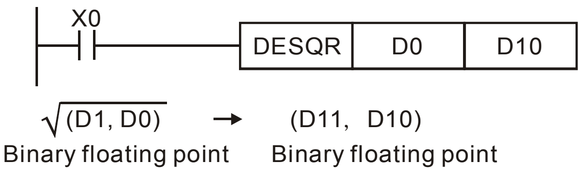
S: Source device **D**: Operation result

Explanations:

1. This instruction performs a square root operation on the floating point value in **S** and stores the result in **D**. All data will be operated in binary floating point format and the result will also be stored in floating point format.
2. If the source device **S** is specified as constant K or H, the integer value will automatically be converted to binary floating value.
3. If operation result of **D** is 0 (zero), 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 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), M1068 (Execution Error Locked)

Program Example 1:

When X0 = ON, the square root of binary floating point (D1, D0) is stored in (D11, D10) after the operation of square root.



Program Example 2:

When X2 = ON, the square root of K1234 (automatically converted to binary floating value) is stored in (D11, D10).

API	Mnemonic			Operands	Function	Controllers
128	D	POW	P	S1 S2 D	Floating point power operation	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DPOW, DPOWP: 13 steps
S_1					*	*							*			
S_2					*	*							*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

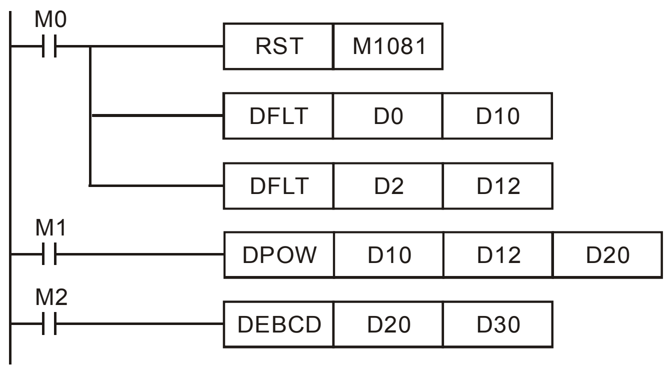
S_1: Base **S_2**: Exponent **D**: Operation result

Explanations:

1. Perform power operation on binary floating value **S_1** and **S_2** and save the result in **D**.
   POW [**S_1**+1, **S_1**]^[**S_2**+1, **S_2**] = **D**
2. Only a positive number is valid for **S**. Register **D** has to be 32-bit format. Operation is conducted in floating point value, so the value in **S_1** and **S_2** needs to be converted into floating value before power operation.
3. Example of power operation:
   When $\mathbf{S_1}^{S2}$ = D, D = ? Assume **S_1** = 5, **S_2** = 3, **D** = 5^3 =125
4. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag).
   If absolute value of the result is larger than max. floating value, carry flag M1022 = ON.
   If absolute value of the result is smaller than min. floating value, borrow flag M1021 = ON.
   If the conversion result is 0, zero flag M1020 = ON.

Program Example:

1. When M0 = ON, convert (D1, D0) and (D3, D2) to binary floating value and save the result in register (D11, D10) and (D13, D12) individually.
2. When M1 = ON, perform power operation with (D11, D10) as base and (D13, D12) as exponent. The value is saved in register (D21, D20) in binary floating format.
3. When M2 = ON, convert the value in (D21, D20) into decimal floating point value and save the result in (D31, D30). (At this time, D31 indicates powers of 10 for D30)



API	Mnemonic			Operands	Function	Controllers
129	D	INT	P	(S) (D)	Float to integer	ES2/EX2 SS2 SA2 SX2

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
S											*	*	*			DINT, DINTP: 9 steps
D											*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

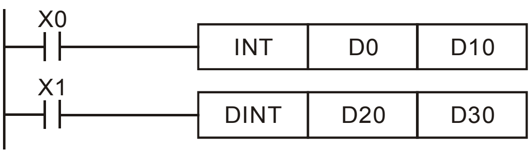
S: Source device **D**: Operation result

Explanations:

1. The binary floating point value in the register **S** is converted to BIN integer and stored in register **D**. The decimal of the operation result will be left out.
2. This instruction is the opposite of the API 49 (FLT) instruction.
3. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag).
   If the conversion result is 0, zero flag M1020 = ON.
   If there is any decimal left out, borrow flag M1021 = ON.
   If the conversion result is larger than the below range, carry flag M1022 = ON
   16-bit instruction: -32,768 ~ 32,767
   32-bit instruction: -2,147,483,648 ~ 2,147,483,647

Program Example:

1. 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 the result will be left out.
2. 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 the result will be left out.



API	Mnemonic			Operands	Function	Controllers
130	D	SIN	P	S D	Sine	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*							*			DSIN, DSINP: 9 steps
D													*			

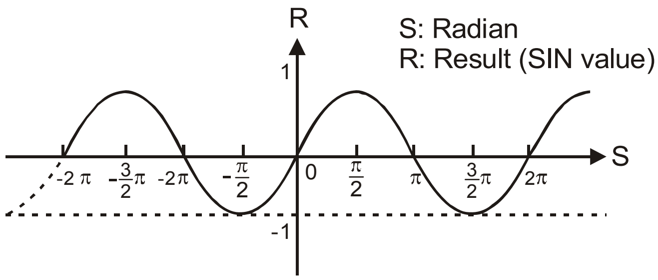
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device (0°≦**S**<360°) **D**: Operation result

Explanations:

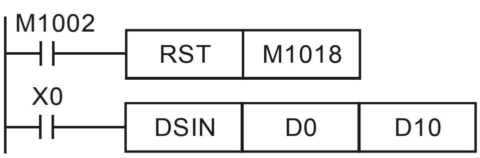
1. SIN instruction performs sine operation on **S** and stores the result in **D**.
2. The value in **S** can be set as radian or degree by flag M1018.
3. M1018 = OFF, radian mode. RAD = degree $\times\pi/180$.
4. M1018 = ON, degree mode. Degree range: 0°≦degree<360°.
5. Flag: M1018 (Flag for Radian/Degree)
6. See the figure below for the relation between the radian and the operation result:

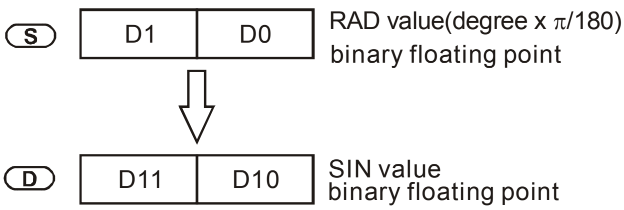


7. If operation result in **D** is 0, Zero flag M1020 = ON.

Program Example 1:

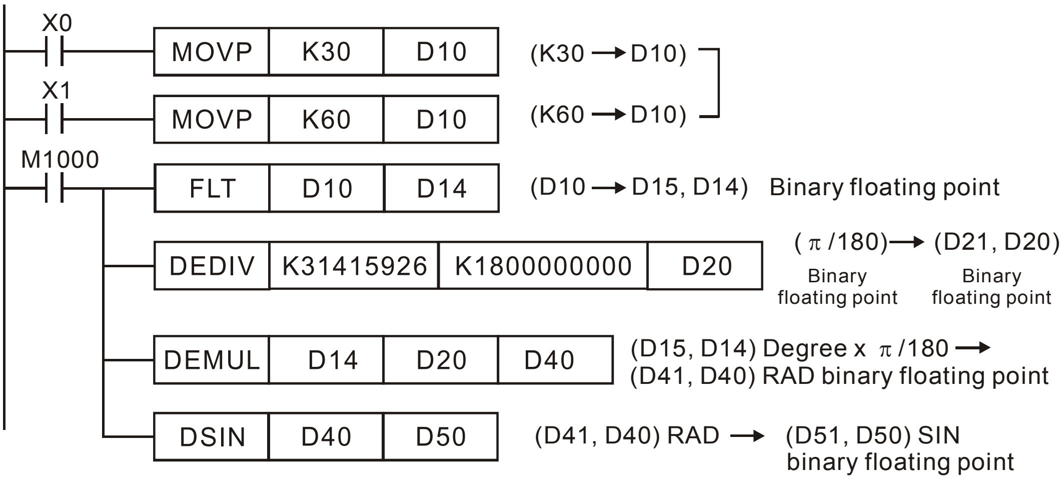
M1018 = OFF, radian mode. When X0 = ON, DSIN instruction conducts sine operation on binary floating value in (D1, D0) and stores the SIN value in (D11, D10) in binary floating format.





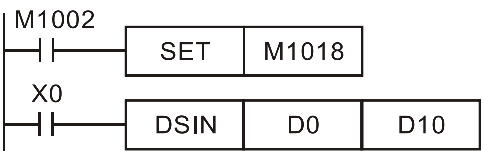
Program Example 2:

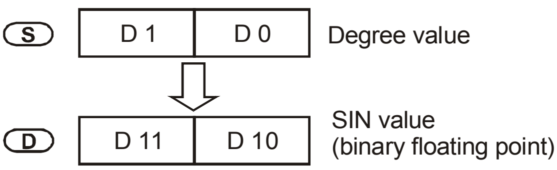
M1018 = OFF, radian mode. Select the degree value from inputs X0 and X1 and convert it to RAD value for further sine operation.



Program Example 3:

M1018 = ON, degree mode. When X0 = ON, DSIN instruction performs sine operation on the degree value (0°≦degree＜360°) in (D1, D0) and stores the SIN value in (D11, D10) in binary floating format.





API	Mnemonic			Operands	Function	Controllers			
131	D	COS	P	S D	Cosine	ES2/EX2	SS2	SA2	SX2

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													*			

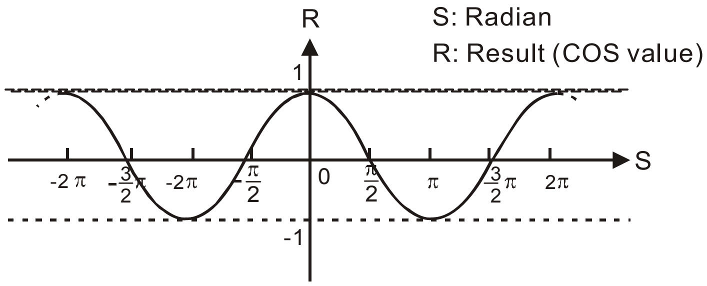
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device (0°≦**S**<360°) **D**: Operation result

Explanations:

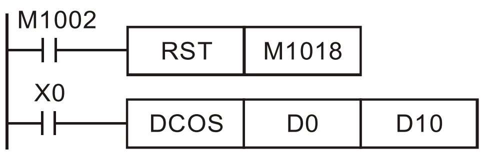
1. COS instruction performs cosine operation on **S** and stores the result in **D**.
2. The value in **S** can be set as radian or degree by flag M1018.
3. M1018 = OFF, radian mode. RAD = degree $\times \pi /180$.
4. M1018 = ON, degree mode. Degree range: 0°≦degree<360°.
5. Flag: M1018 (Flag for Radian/Degree)
6. See the figure below for the relation between the radian and the operation result:

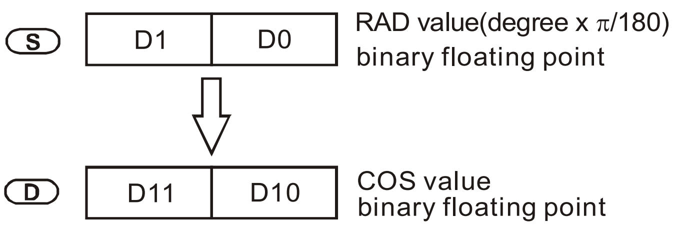


7. If operation result in **D** is 0, Zero flag M1020 = ON.

Program Example 1:

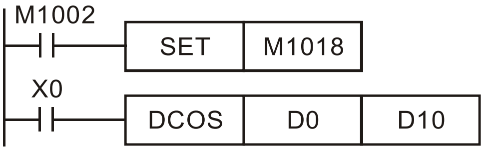
M1018 = OFF, radian mode. When X0 = ON, DCOS instruction conducts cosine operation on binary floating value in (D1, D0) and stores the COS value in (D11, D10) in binary floating format.

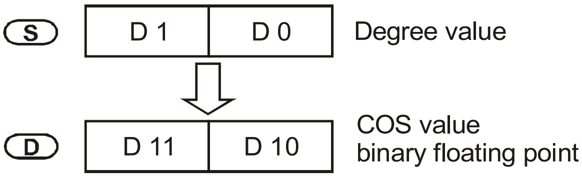




Program Example 2:

M1018 = ON, degree mode. When X0 = ON, DCOS instruction performs cosine operation on the degree value (0°≦degree＜360°) in (D1, D0) and stores the COS value in (D11, D10) in binary floating format..





API	Mnemonic			Operands	Function	Controllers
132	D	TAN	P	S D	Tangent	ES2/EX2 SS2 SA2 SX2

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

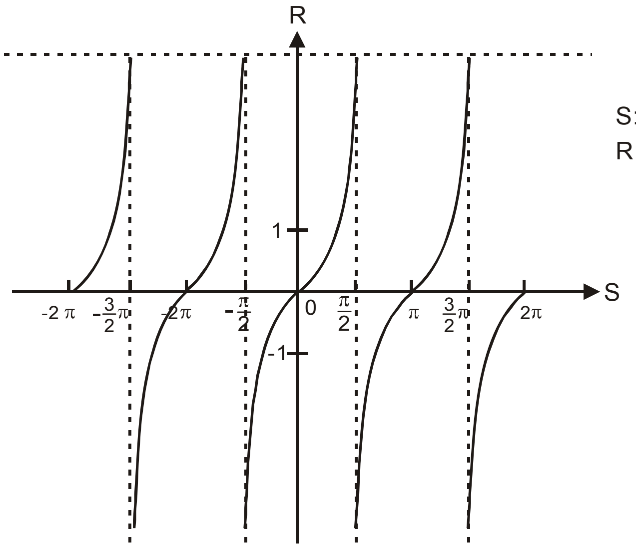
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device (0˚≦**S**＜360˚) **D**: Operation result

Explanations:

1. TAN instruction performs tangent operation on **S** and stores the result in **D**.
2. The value in **S** can be set as radian or degree by flag M1018.
3. M1018 = OFF, radian mode. RAD = degree $\times\pi$/180.
4. M1018 = ON, degree mode. Degree range: 0˚≦degree＜360˚.
5. Flag: M1018 (Flag for Radian/Degree)
6. See the figure below for the relation between the radian and the operation result

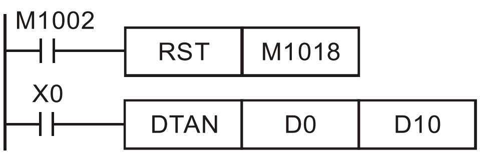


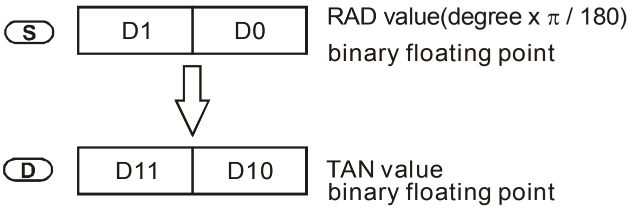
S: Radian
R: Result (TAN value)

7. If operation result in **D** is 0, Zero flag M1020 = ON.

Program Example 1:

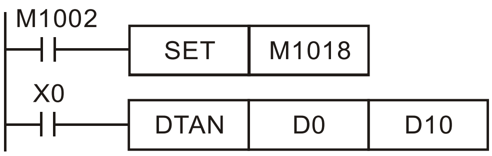
M1018 = OFF, radian mode. When X0 = ON, DTAN instruction performs tangent operation on the radian value in (D1, D0) and stores the TAN value in (D11, D10) in binary floating format.

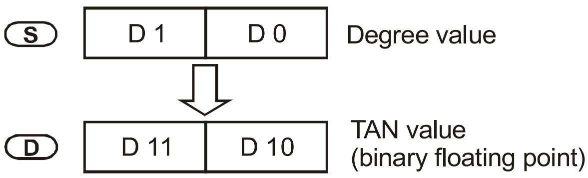




Program Example 2:

M1018 = ON, degree mode. When X0 = ON, DTAN instruction performs tangent operation on the degree value (0°≦degree＜360°) in (D1, D0) and stores the TAN value in (D11, D10) in binary floating format.





API	Mnemonic			Operands	Function	Controllers
133	D	ASIN	P	S D	Arc Sine	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*							*			DASIN, DASINP: 9 steps
D													*			

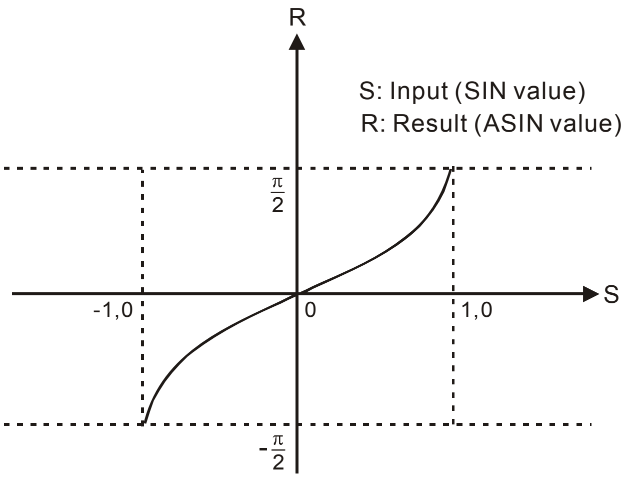
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device (binary floating value) **D**: Operation result

Explanations:

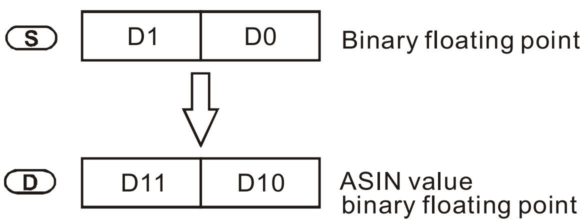
1. ASIN instruction performs arc sine operation on **S** and stores the result in **D**
2. ASIN value = SIN^{-1}
3. See the figure below for the relation between input **S** and the result:



4. If operation result in **D** is 0, Zero flag M1020 = ON.
5. The decimal value of the SIN value designated by **S** should be within -1.0 ~ +1.0. If the value exceeds the range, M1067 and M1068 will be ON and instruction will be disabled.

Program Example:

When X0 = ON, DASIN instruction performs arc sine operation on the binary floating value in (D1, D0) and stores the ASIN value in (D11, D10) in binary floating format..



API	Mnemonic			Operands	Function	Controllers
134	D	ACOS	P	S D	Arc Cosine	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*							*			DACOS, DACOSP: 9 steps
D													*			

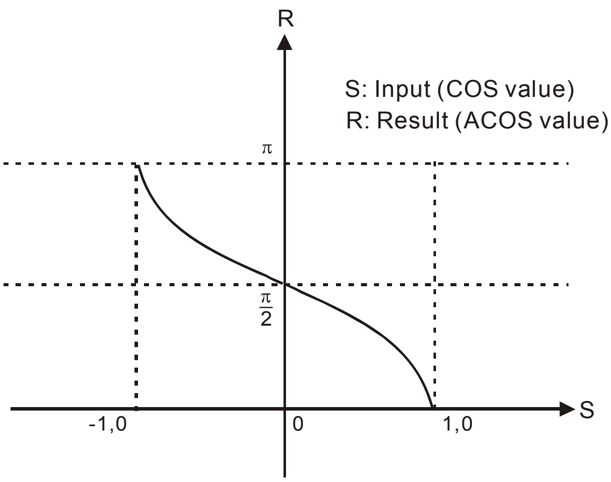
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device (binary floating value) **D**: Operation result

Explanations:

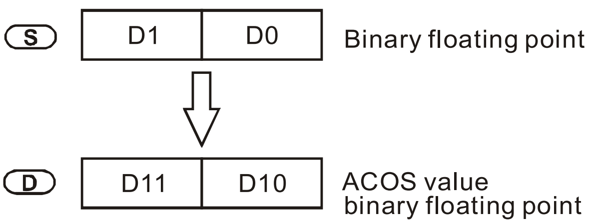
1. ACOS instruction performs arc cosine operation on **S** and stores the result in **D**
2. ACOS value = COS^{-1}
3. See the figure below for the relation between the input **S** and the result:



4. If operation result in **D** is 0, Zero flag M1020 = ON.
5. The decimal value of the COS value designated by **S** should be within -1.0 ~ +1.0. If the value exceeds the range, M1067 and M1068 will be ON and instruction will be disabled.

Program Example:

When X0 = ON, DACOS instruction performs arc cosine operation on the binary floating value in (D1, D0) and stores the ACOS value in (D11, D10) in binary floating format.



API	Mnemonic			Operands	Function	Controllers			
135	D	ATAN	P	S D	Arc Tangent	ES2/EX2	SS2	SA2	SX2

Type / OP	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													*			

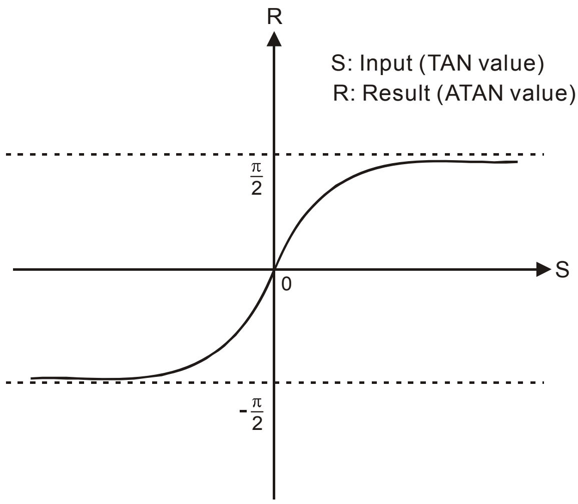
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device (binary floating value) **D**: Operation result

Explanations:

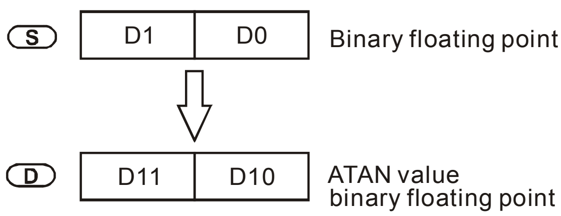
1. ATAN instruction performs arc tangent operation on **S** and stores the result in **D**
2. ATAN value=TAN^{-1}
3. See the figure below for the relation between the input and the result:



4. If operation result in **D** is 0, Zero flag M1020 = ON.

Program Example:

When X0 = ON, DATAN instruction performs arc tangent operation on the binary floating value in (D1, D0) and stores the ATAN value in (D11, D10) in binary floating format.



API	Mnemonic		Operands	Function	Controllers
143	DELAY	P	S	Delay	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*							*			DELAY, DELAYP: 3 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

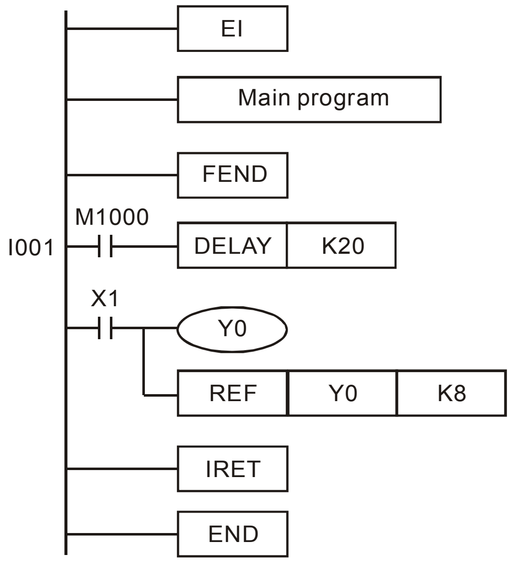
S: Delay time, unit: 0.1ms (K1~K1000)

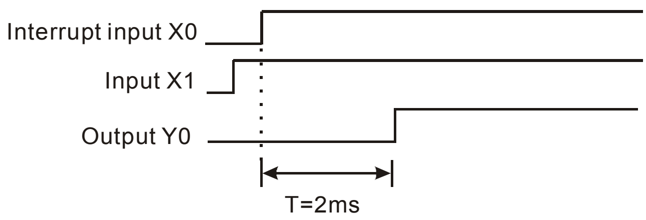
Explanations:

When DELAY instruction executes, in every scan cycle, the execution of the program after DELAY instruction will be delayed according to the delay time.

Program Example:

When interrupt input X0 is triggered from OFF to ON, interrupt subroutine executes DELAY instruction first, therefore the program after DELAY instruction (X1 = ON, Y0 = ON…) will be delayed for 2ms.





Points to note:

1. User can adjust the delay time according to the actual needs.
2. The delay time of DELAY instruction could be increased due to the execution of communication, high-speed counter and high-speed pulse output instructions.
3. The delay time of DELAY instruction could be increased due to the delay of transistor or relay when external output (transistor or relay) is specified.

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
144	GPWM	S1 S2 D	General PWM output				

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	GPWM: 7 steps
S_1													*			
S_2													*			
D		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

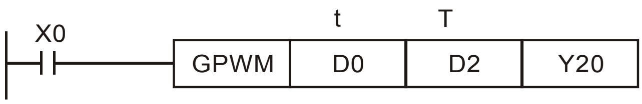
S_1: Width of output pulse **S_2**: Pulse output cycle (occupies 3 devices) **D**: Pulse output device

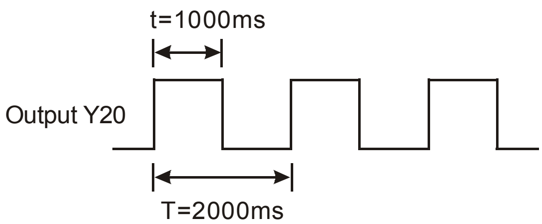
Explanations:

1. When GPWM instruction executes, pulse output will be executes on device specified by **D** according to pulse output width **S_1** and pulse output cycle **S_2**.
2. **S_1**: pulse output width. Range: t = 0~32,767ms.
3. **S_2**: pulse output cycle. Range: T = 1~32,767ms, **$S_1 \leqq S_2$**.
4. **S_2** +1 and **S_2** +2 are system-defined parameters, please don't use them.
5. **D:** pulse output device: Y, M and S.
6. When **$S_1 \leqq$** 0, no pulse output will be performed. When **$S_1 \geqq S_2$**, the pulse output device remains ON.
7. **S_1** and **S_2** can be modified when GPWM instruction is being executed

Program Example:

Assume D0 = K1000, D2 = K2000. When X0 = ON, Y20 will output pulses as the following diagram. When X0 = OFF, Y20 output will be OFF.





Points to note:

1. The instruction operates by the scan cycle; therefore the maximum error will be one PLC scan cycle. **S_1**, **S_2** and (**S_2** - **S_1**) should be bigger than PLC scan cycle, otherwise malfunction will occur during GPWM outputs.
2. Please note that placing this instruction in a subroutine will cause inaccurate GPWM outputs

API	Mnemonic			Operands	Function	Controllers
147	D	SWAP	P	S	Byte swap	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S								*	*	*	*	*	*	*	*	SWAP, SWAPP: 3 steps DSWAP, DSWAPP: 5 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

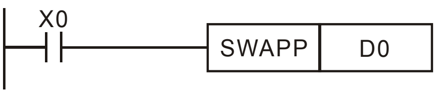
S: Device for byte swap.

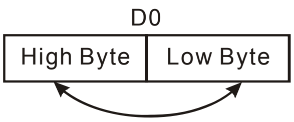
Explanations:

1. For 16-bit instruction, high byte and low byte of the register will be swapped.
2. For 32-bit instruction, byte swap is conducted on the 2 registers separately.
3. This instruction adopts pulse execution instructions (SWAPP, DSWAPP)
4. If operand **D** uses device F, only 16-bit instruction is available

Program Example 1:

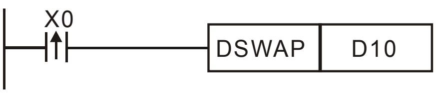
When X0 = ON, high byte and low byte of D0 will be swapped.

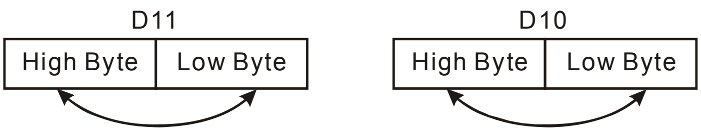




Program Example 2:

When X0 = ON, high byte and low byte of D11 will be swapped as well as the high byte and low byte of D10.





API	Mnemonic	Operands	Function	Controllers
150	MODRW	S1 S2 S3 S n	MODBUS Read/ Write	ES2/EX2 SS2 SA2 SX2

OP \ Type	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*							*			MODRW: 11 steps
S_2					*	*							*			
S_3					*	*							*			
S													*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Device address (K1~K254) **S_2**: Function code: K2(H2), K3(H3), K5(H5), K6(H6), K15(H0F) , K16(H10) **S_3**: Data address **S**: Data register **n**: Data length.

Explanations:

1. MODRW supports COM1 (RS-232), COM2 (RS-485), COM3 (RS-485).
2. **S_1**: Address of the device to be accessed. Range: K1~K254.
3. **S_2**: Function code. **H02**: read multiple bit devices of DVP-PLC; **H03**: read multiple word devices of AC motor drive or DVP-PLC; **H05**: force ON/OFF bit device; **H06**: write in single word device of AC motor drive or DVP-PLC; **H0F**: write in multiple bit devices of DVP-PLC; **H10**: write in multiple word devices of AC motor drive or DVP-PLC. Only these function codes are available currently; other function codes are not executable. Please refer to the program examples below for more information
4. **S_3**: Address of the data to be accessed. If the address is illegal for the designated communication device, the communication device will respond with an error message and DVP-PLC will store the error code and associated error flag will be ON.
   - Associated registers and flags indicating errors on PLC com ports: (For detailed information please refer to **Points to note** of API 80 RS instruction.)

PLC COM	COM1	COM2	COM3
Error flag	M1315	M1141	M1319
Error code	D1250	D1130	D1253

   - For example, if 8000H is illegal for DVP-PLC, the error will be in indicated by different set of flags and registers. For COM2, M1141 will be ON and D1130 = 2; for COM1, M1315 = ON and D1250 = 3, for COM3, M1319 = ON and D1253 = 3. Please check the user manual of DVP-PLC for error code explanations.
5. **S**: Registers for storing read/written data. Registers starting from **S** stores the data to be written into the communication device or the data read from the communication device. When COM2 sends the function code of reading(K2/K3), the registers from S directly receive the data string and stored the converted data in D1296~D1311. Please refer to program example

1 and 3 for further explanation. When COM1 or COM3 sends the function code of reading(K2/K3), the registers store the converted data directly. Refer to program example 2 and 4 for further explanations.

6. **n**: Data length for accessing.
   - When $\mathbf{S_2}$ (MODBUS function code) is specified as H05 which designates the PLC force ON/OFF status, **n** = 0 indicates ON and **n** = 1 indicates OFF.
   - When $\mathbf{S_2}$ is specified as H02, H03, H0F, H10 which designate the data length for accessing, the available set range will be **K1~Km**, where **m** value should be specified according to communication modes and COM ports as the table below. (H02/H0F, unit: Bit. H03/H10, unit: Word.)

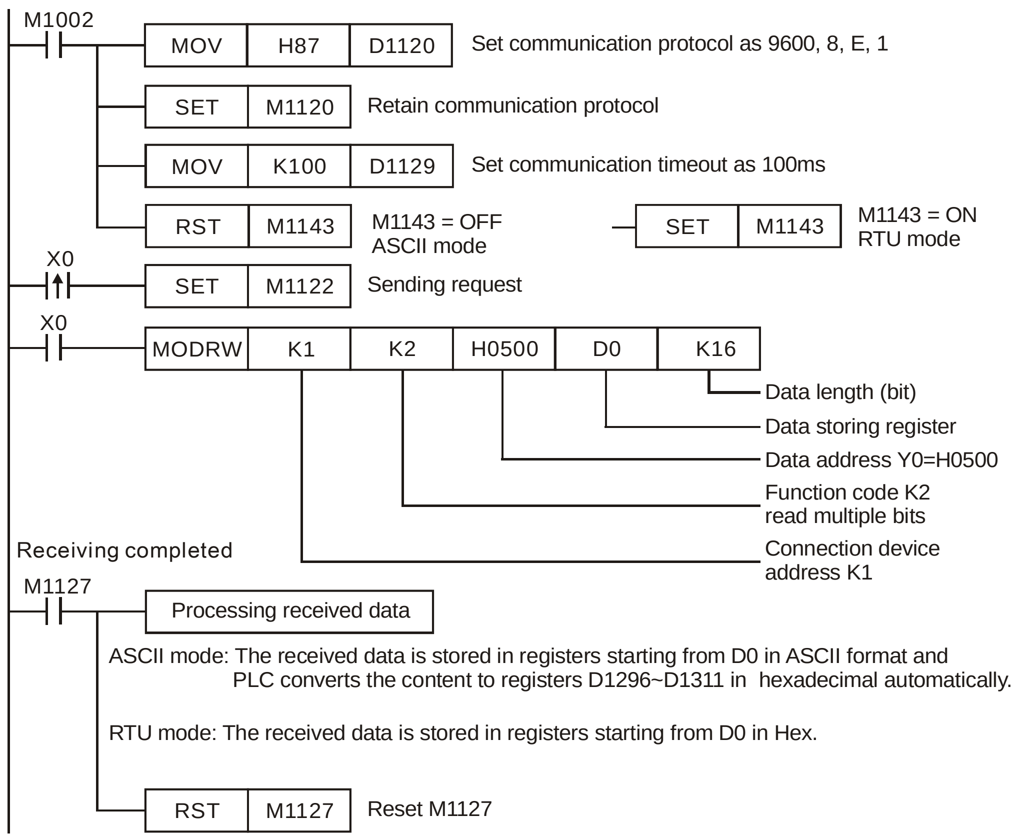
COM. mode	COM	H02	H03	H0F	H10
RTU	COM1	K 64	K 16	K 64	K 16
	COM2	K 64	K 16	K 64	K 16
	COM3	K 64	K 16	K 64	K 16
ASCII	COM1	K 64	K 16	K 64	K 16
	COM2	K 64	K 8	K 64	K 8
	COM3	K 64	K 16	K 64	K 16

7. There is no limitation on the times of using this instruction, however only one instruction can be executed on the same COM port at a time.
8. Rising-edge contact (LDP, ANDP, ORP) and falling-edge contact (LDF, ANDF, ORF) can not be used as drive contact of MODRW (Function code H02, H03) instruction, otherwise the data stored in the receiving registers will be incorrect.
9. If rising-edge contacts (LDP, ANDP, ORP) or falling-edge contacts (LDF, ANDF, ORF) is used before MODWR instruction, sending request flag M1122(COM2) / M1312(COM1) / M1316(COM3) has to be executed as a requirement.
10. MODRW instruction determines the COM port according to the communication request. The COM port determination is made following the order: COM1→COM3→COM2. Therefore, please insert every MODRW instruction right after the sending request instruction for avoiding errors on the target location for data access.
11. For detailed explanation of the associated flags and special registers, please refer to **Points to note** of API 80 RS instruction.

Program Example 1: COM2(RS-485), Function Code H02

1. Function code K2 (H02): read multiple bit devices, up to 64 bits can be read..
2. PLC1 connects to PLC2: (M1143 = OFF, ASCII mode), (M1143 = ON, RTU Mode)

3. In ASCII or RTU mode, when PLC's COM2 sends out data, the data will be stored in D1256~D1295. The feedback data will be stored in registers starting with **S** and converted into D1296~D1311 in Hex automatically.
4. Take the connection between PLC1 (PLC COM2) and PLC2(PLC COM1) for example, the tables below explains the status when PLC1 reads Y0~Y17 of PLC2.



ASCII Mode (M1143 = OFF):

When X0 = ON, MODRW instruction executes the function specified by Function Code 02.

PLC1⇨ PLC2，PLC1 sends: "**01 02 0500 0010 E8**"

PLC2 ⇨PLC1，PLC1 receives: **"01 02 02 3412 B5**"

Registers for data to be sent (sending messages)

Register	Data		Descriptions	
D1256 Low	'0'	30 H	ADR 1	Device address: ADR (1,0)
D1256 High	'1'	31 H	ADR 0	
D1257 Low	'0'	30 H	CMD 1	Control parameter: CMD (1,0)
D1257 High	'2'	32 H	CMD 0	
D1258 Low	'0'	30 H	Y0 = H0500 Starting Data Address	
D1258 High	'5'	35 H		
D1259 Low	'0'	30 H		
D1259 High	'0'	30 H		

Register	Data		Descriptions	
D1260 Low	'0'	30 H	Number of Data(count by bit)	
D1260 High	'0'	30 H		
D1261 Low	'1'	31 H		
D1261 High	'0'	30 H		
D1262 Low	'E'	45 H	LRC CHK 1	Checksum: LRC CHK (0,1)
D1262 High	'8'	38 H	LRC CHK 0	

Registers for received data (responding messages)

Register	Data		Descriptions	
D0 Low	'0'	30 H	ADR 1	
D0 High	'1'	31 H	ADR 0	
D1 Low	'0'	30 H	CMD 1	
D1 High	'2'	33 H	CMD 0	
D2 Low	'0'	30 H	Number of Data (count by Byte)	
D2 High	'2'	32 H		
D3 Low	'3'	33 H	Content of address 0500H~ 0515H	1234 H PLC automatically converts ASCII codes and store the converted value in D1296
D3 High	'4'	34 H		
D4 Low	'1'	31H		
D4 High	'2'	32H		
D5 Low	'B'	52H	LRC CHK 1	
D5 High	'5'	35 H	LRC CHK 0	

Analysis of the read status of PLC2 Y0~Y17: 1234H

Device	Status	Device	Status	Device	Status	Device	Status
Y0	OFF	Y1	OFF	Y2	ON	Y3	OFF
Y4	ON	Y5	ON	Y6	OFF	Y7	OFF
Y10	OFF	Y11	ON	Y12	OFF	Y13	OFF
Y14	ON	Y15	OFF	Y16	OFF	Y17	OFF

RTU Mode (M1143 = ON):

When X0 = ON, MODRW instruction executes the function specified by Function Code 02

PLC1⇨ PLC2，PLC1sends: "**01 02 0500 0010 79 0A**"

PLC2 ⇨ PLC1，PLC1receives: "**01 02 02 34 12 2F 75**"

Registers for data to be sent (sending messages)

Register	Data	Descriptions
D1256 Low	01 H	Address
D1257 Low	02 H	Function
D1258 Low	05 H	Y0 = H0500 Starting Data Address
D1259 Low	00 H	

Register	Data	Descriptions
D1260 Low	00 H	Number of Data (count by word)
D1261 Low	10 H	
D1262 Low	79 H	CRC CHK Low
D1263 Low	0A H	CRC CHK High

Registers for received data (responding messages)

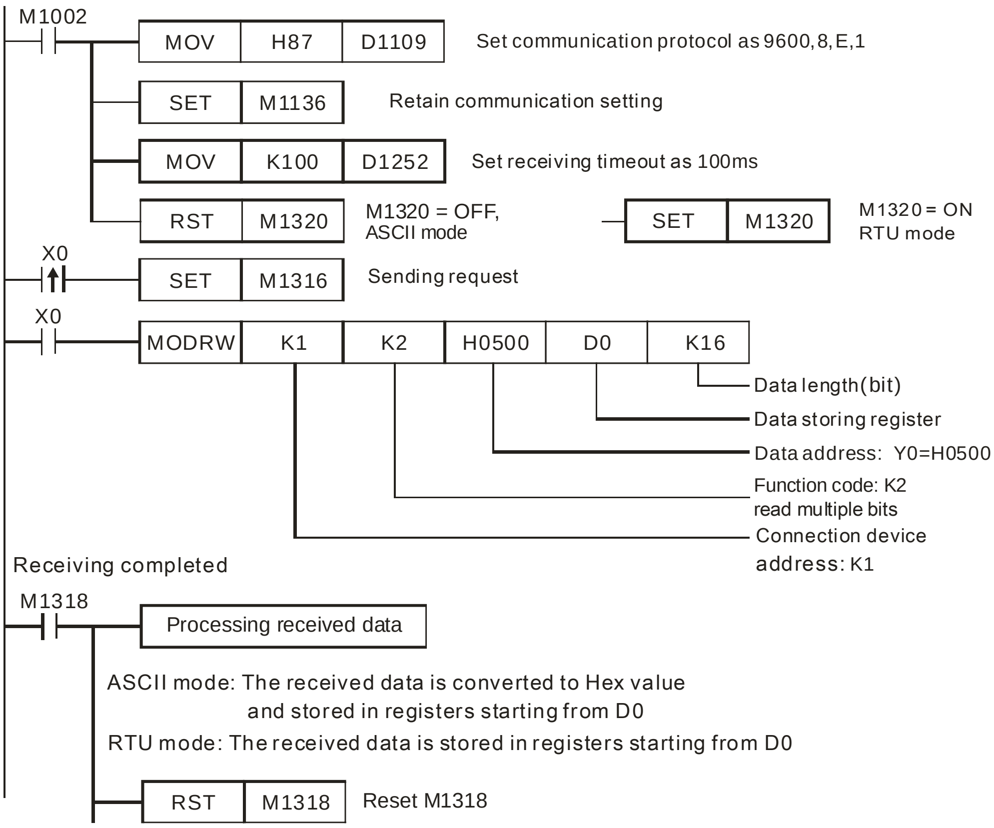
Register	Data	Descriptions
D0	1234 H	PLC stores the value 1234H into D1296
D1 Low	02 H	Function
D2 Low	02 H	Number of Data (Byte)
D3 Low	34 H	Content of address H0500~H0515
D4 Low	12 H	
D5 Low	2F H	CRC CHK Low
D6 Low	75 H	CRC CHK High

Analysis of the read status of PLC2 Y0~Y17: 1234H

Device	Status	Device	Status	Device	Status	Device	Status
Y0	OFF	Y1	OFF	Y2	ON	Y3	OFF
Y4	ON	Y5	ON	Y6	OFF	Y7	OFF
Y10	OFF	Y11	ON	Y12	OFF	Y13	OFF
Y14	ON	Y15	OFF	Y16	OFF	Y17	OFF

Program Example 2: COM1(RS-232) / COM3(RS-485), Function Code H02

1. Function code K2 (H02): read multiple bit devices. Up to 64 bits can be read.
2. PLC1 connects to PLC2: (M1320 = OFF, ASCII mode), (M1320 = ON, RTU mode)
3. For both ASCII and RTU modes, PLC COM1/COM3 only stores the received data in registers starting from **S**, and will not store the data to be sent. The stored data can be transformed and moved by using DTM instruction for applications of other purposes.
4. Take the connection between PLC1 (PLC COM3) and PLC2(PLC COM1) for example, the tables below explains the status when PLC1 reads Y0~Y17 of PLC2
   - If PLC1 applies COM1 for communication, the below program can be usable by changing:
     1. D1109→D1036: communication protocol
     2. M1136→M1138: retain communication setting
     3. D1252→D1249: Set value for data receiving timeout
     4. M1320→M1139: ASCII/RTU mode selection
     5. M1316→M1312: sending request
     6. M1318→M1314: receiving completed flag



- ASCII mode (COM3: M1320 = OFF, COM1: M1139 = OFF):

When X0 = ON, MODRW instruction executes the function specified by Function Code 02

PLC1⇨ PLC2, PLC1 sends: “**01 02 0500 0010 E8**”

PLC2 ⇨PLC1, PLC1 receives: “**01 02 02 3412 B5**”

PLC1 data receiving register D0

Register	Data	Descriptions
D0	1234H	PLC converts the ASCII data in address 0500H~0515H and stores the converted data automatically.

Analysis of the read status of PLC2 Y0~Y17: 1234H

Device	Status	Device	Status	Device	Status	Device	Status
Y0	OFF	Y1	OFF	Y2	ON	Y3	OFF
Y4	ON	Y5	ON	Y6	OFF	Y7	OFF
Y10	OFF	Y11	ON	Y12	OFF	Y13	OFF
Y14	ON	Y15	OFF	Y16	OFF	Y17	OFF

- RTU mode (COM3: M1320 = ON, COM1: M1139 = ON):

When X0 = ON, MODRW instruction executes the function specified by Function Code 02

PLC1 ⇨ PLC2, PLC1 sends: “**01 02 0500 0010 79 0A**”

PLC2 ⇨ PLC1, PLC1 receives: “**01 02 02 34 12 2F 75**”

PLC data receiving register:

Register	Data	Descriptions
D0	1234 H	PLC converts the data in address 0500H ~ 0515H and stores the converted data automatically.

Analysis of the read status of PLC2 Y0~Y17: 1234H

Device	Status	Device	Status	Device	Status	Device	Status
Y0	OFF	Y1	OFF	Y2	ON	Y3	OFF
Y4	ON	Y5	ON	Y6	OFF	Y7	OFF
Y10	OFF	Y11	On	Y12	OFF	Y13	OFF
Y14	ON	Y15	OFF	Y16	OFF	Y17	OFF

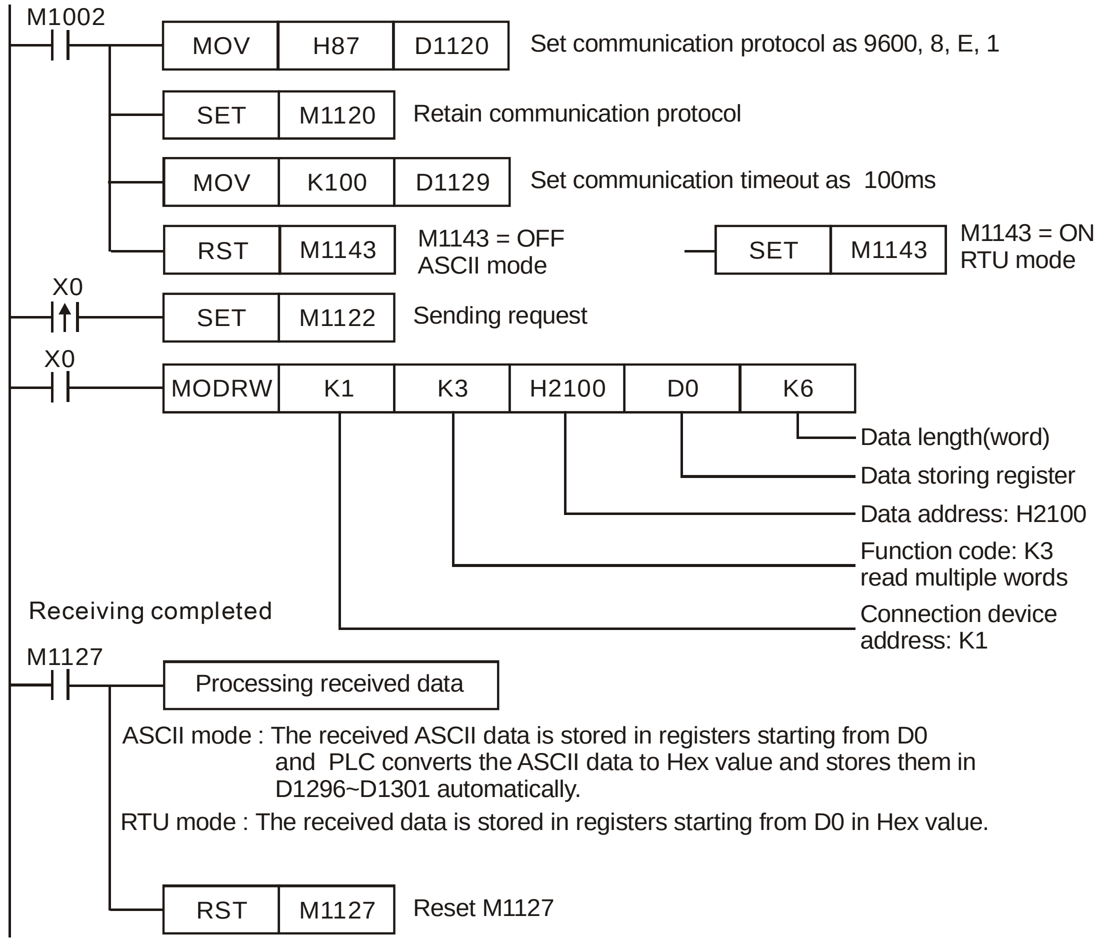
5. Relative flags and data registers when COM1 / COM2 / COM3 works as Master:

	COM2	COM1	COM3	Function
COM. setting	M1120	M1138	M1136	Retain communication setting
	M1143	M1139	M1320	ASCII/RTU mode selection
	D1120	D1036	D1109	Communication protocol
	D1121	D1121	D1255	PLC communication address
Sending request	M1122	M1312	M1316	Sending request
	D1129	D1249	D1252	Set value for data receiving timeout (ms)
Receiving completed	M1127	M1314	M1318	Data receiving completed
Errors	-	M1315	M1319	Data receiving error
	-	D1250	D1253	Communication error code
	M1129	-	-	Receiving timeout
	M1140	-	-	Data receiving error
	M1141	-	-	Parameter error. Exception Code is stored in D1130
	D1130	-	-	Error code (Exception code) returning from Modbus communication

Program Example 3: COM2 (RS-485), Function Code H03

1. Function code K3 (H03): read multiple Word devices. Up to 16 words can be read. For COM2 ASCII mode, only 8 words can be read.
2. For ASCII or RTU mode, PLC COM2 stores the data to be sent in D1256~D1295, converts the received data in registers starting from **S**, and stores the converted 16-bit data in D1296 ~ D1311.

3. Take the connection between PLC (PLC COM2) and VFD-B for example, the tables below explains the status when PLC reads status of VFD-B. (M1143 = OFF, ASCII Mode) (M1143 = ON, RTU Mode)



ASCII mode (M1143 = OFF):

When X0 = ON, MODRW instruction executes the function specified by Function Code 03

PLC ⇨ VFD-B, PLC sends: **"01 03 2100 0006 D5"**

VFD-B ⇨ PLC, PLC receives: **"01 03 0C 0100 1766 0000 0000 0136 0000 3B"**

Registers for data to be sent (sending messages)

Register	Data		Descriptions	
D1256 Low byte	'0'	30 H	ADR 1	Address of VFD-B: ADR (1,0)
D1256 High byte	'1'	31 H	ADR 0	
D1257 Low byte	'0'	30 H	CMD 1	Control parameter: CMD (1,0)
D1257 High byte	'3'	33 H	CMD 0	
D1258 Low byte	'2'	32 H	Data Address	
D1258 High byte	'1'	31 H		
D1259 Low byte	'0'	30 H		
D1259 High byte	'0'	30 H		

Register	Data		Descriptions	
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	‘D’	44 H	LRC CHK 1	Checksum: LRC CHK (0,1)
D1262 High byte	‘5’	35 H	LRC CHK 0	

Registers for received data (responding messages)

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 H2100	0100 H PLC COM2 automatically converts ASCII codes to Hex and stores the converted value in D1296
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 H2101	1766 H PLC COM2 automatically converts ASCII codes to Hex and stores the converted value in D1297
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 H2102	0000 H PLC COM2 automatically converts ASCII codes to hex and stores the converted value in D1298
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 H2103	0000 H PLC COM2 automatically converts ASCII codes to hex and stores the converted value in D1299
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	0136 H
D11 high byte	‘1’	31 H		

3

Register	Data		Descriptions	
D12 low byte	‘3’	33 H	address H2104	PLC COM2 automatically converts ASCII codes to hex and stores the converted value in D1300
D12 high byte	‘6’	36 H		
D13 low byte	‘0’	30 H	Content of address H2105	0000 H PLC COM2 automatically converts ASCII codes to hex and stores the converted value in D1301
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	

RTU mode (M1143 = ON):

When X0 = ON, MODRW instruction executes the function specified by Function Code 03

PLC ⇨ VFD-B, PLC sends: **” 01 03 2100 0006 CF F4”**

VFD-B ⇨ PLC, PLC receives: **“01 03 0C 0000 0503 0BB8 0BB8 0000 012D 8E C5”**

Registers for data to be sent (sending messages)

Register	Data	Descriptions
D1256 Low byte	01 H	Address
D1257 Low byte	03 H	Function
D1258 Low byte	21 H	Data Address
D1259 Low byte	00 H	
D1260 Low byte	00 H	Number of data (count by word)
D1261 Low byte	06 H	
D1262 Low byte	CF H	CRC CHK Low
D1263 Low byte	F4 H	CRC CHK High

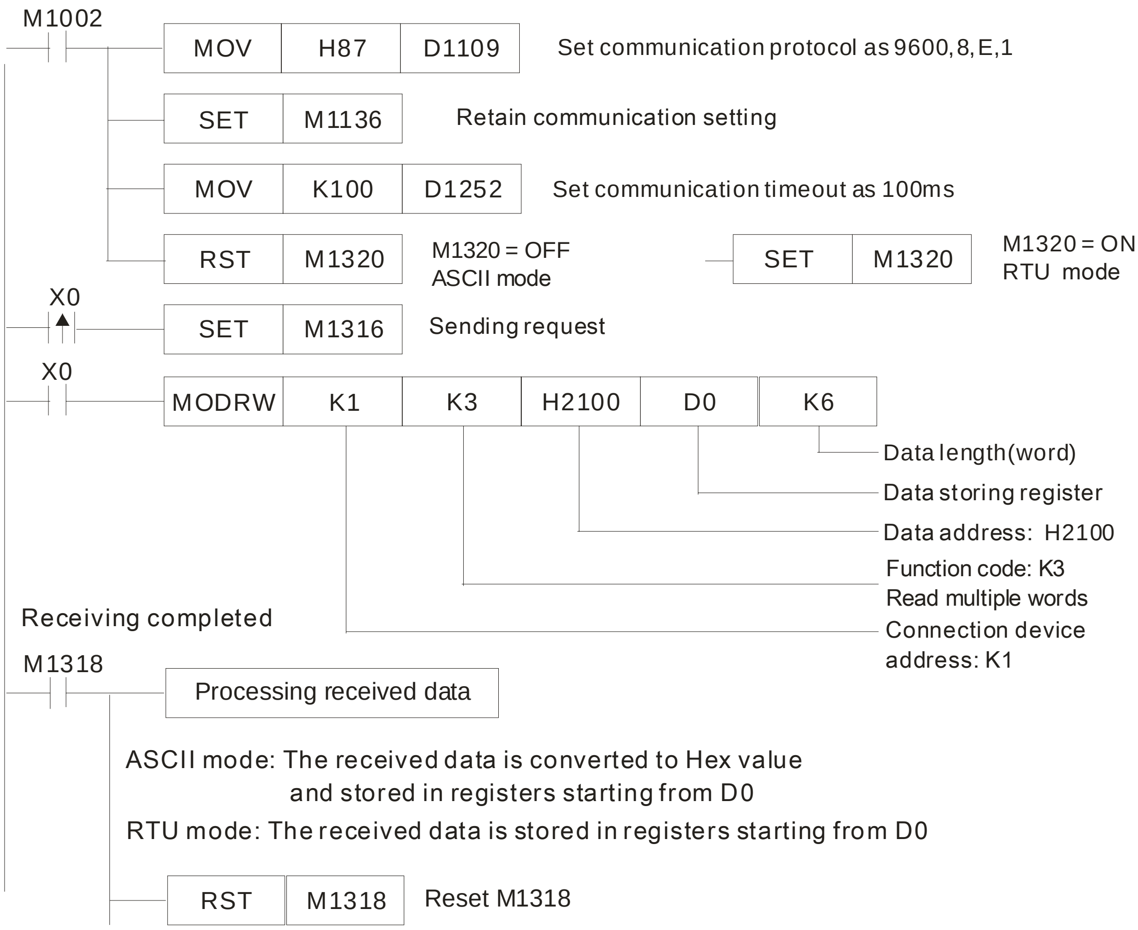
Registers for received data (responding messages)

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 H2100	0000 H PLC COM2 automatically stores the value in D1296
D4 low byte	00 H		
D5 low byte	05 H	Content of	0503 H

Register	Data	Descriptions	
D6 low byte	03 H	address H2101	PLC COM2 automatically store the value in D1297
D7 low byte	0B H	Content of address H2102	0BB8 H PLC COM2 automatically stores the value in D1298
D8 low byte	B8 H		
D9 low byte	0B H	Content of address H2103	0BB8 H PLC COM2 automatically store the value in D1299
D10 low byte	B8 H		
D11 low byte	00 H	Content of address H2104	0000 H PLC COM2 automatically store the value in D1300
D12 low byte	00 H		
D13 low byte	01 H	Content of address H2105	012D H PLC COM2 automatically store the value in D1301
D14 low byte	2D H		
D15 low byte	8E H	CRC CHK Low	
D16 low byte	C5 H	CRC CHK High	

Program example 4: COM1(RS-232) / COM3(RS-485), Function Code H03

1. Function code K3 (H03): read multiple Word devices, up to 16 words can be read. For COM2 ASCII mode, only 8 words can be read..
2. PLC COM1 / COM3 stores the received data in registers starting from **S**, and the stored data can be transformed and moved by using DTM instruction for applications of other purposes.
3. Take the connection between PLC and VFD-B for example, the tables below explains the status when PLC reads VFD-B status. (M1320 = OFF, ASCII Mode), (M1320 = ON, RTU Mode)
   - If PLC applies COM1 for communication, the below program can be usable by changing:
     1. D1109→D1036: communication protocol
     2. M1136→M1138: retain communication setting
     3. D1252→D1249: Set value for data receiving timeout
     4. M1320→M1139: ASCII/RTU mode selection
     5. M1316→M1312: sending request
     6. M1318→M1314: receiving completed flag



ASCII mode (COM3: M1320 = OFF, COM1: M1139 = OFF):

When X0 = ON, MODRW instruction executes the function specified by Function Code 03

PLC ⇨ VFD-B, PLC sends: "**01 03 2100 0006 D5**"

VFD-B ⇨ PLC, PLC receives: "**01 03 0C 0100 1766 0000 0000 0136 0000 3B**"

Registers for received data (responding messages)

Register	Data	Descriptions
D0	0100 H	PLC converts ASCII codes in 2100 H and stores the converted data automatically.
D1	1766 H	PLC converts ASCII codes in 2101 H and stores the converted data automatically.
D2	0000 H	PLC converts ASCII codes in 2102 H and stores the converted data automatically.
D3	0000 H	PLC converts ASCII codes in 2103 H and stores the converted data automatically.
D4	0136 H	PLC converts ASCII codes in 2104 H and stores the converted data automatically.

Register	Data	Descriptions
D5	0000 H	PLC converts ASCII codes in 2105 H and stores the converted data automatically.

RTU mode (COM3: M1320 = ON COM1: M1139 = ON):

When X0 = ON, MODRW instruction executes the function specified by Function Code 03

PLC ⇨ VFD-B, PLC sends: " **01 03 2100 0006 CF F4**"

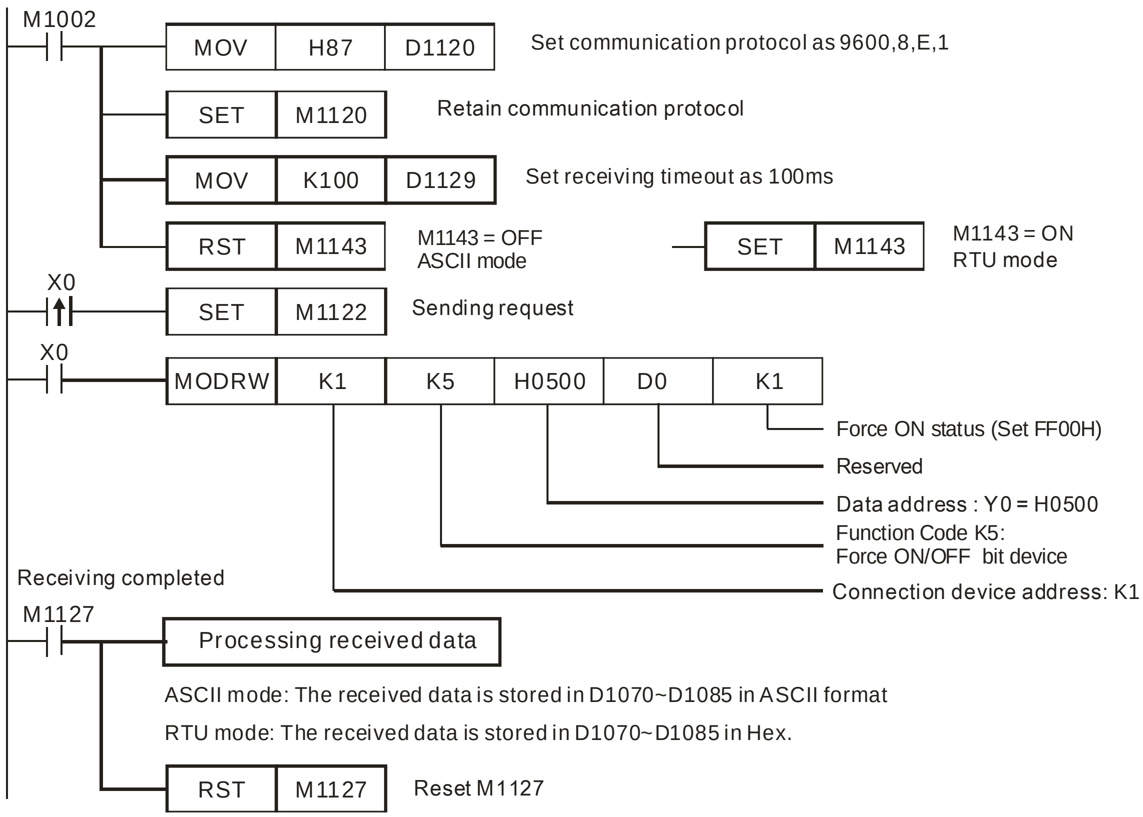
VFD-B ⇨ PLC, PLC receives: "**01 03 0C 0000 0503 0BB8 0BB8 0000 012D 8E C5**"

Registers for received data (responding messages)

Register	Data	Descriptions
D0	0000 H	PLC converts data in 2100 H and stores the converted data automatically.
D1	0503 H	PLC converts data in 2101 H and stores the converted data automatically.
D2	0BB8 H	PLC converts data in 2102 H and stores the converted data automatically.
D3	0BB8 H	PLC converts data in 2103 H and stores the converted data automatically.
D4	0136 H	PLC converts data in 2104 H and stores the converted data automatically.
D5	012D H	PLC converts data in 2105 H and stores the converted data automatically.

Program example 5: COM2(RS-485), Function Code H05

1. Function code K5(H05): Force ON/OFF bit device
2. PLC1 connects to PLC2: (M1143 = OFF, ASCII mode), (M1143 = ON, RTU Mode)
3. **n** = 1 indicates Force ON (set FF00H) and **n** = 0 indicates Force OFF (set 0000H)
4. For ASCII or RTU mode, PLC COM2 stores the data to be sent in D1256~D1295 and stores the received data in D1070~D1085
5. Take the connection between PLC1 (PLC COM2) and PLC2 (PLC COM1) for example, the tables below explain the status when PLC1 Force ON PLC2 Y0.



ASCII mode (M1143 = OFF):

When X0 = ON, MODRW instruction executes the function specified by Function Code 05

PLC1 ⇨ PLC2, PLC sends: "**01 05 0500 FF00 6F**"

PLC2 ⇨ PLC1, PLC receives: "**01 05 0500 FF00 6F**"

Registers for data to be sent (sending messages)

Register	Data		Descriptions	
D1256 low byte	'0'	30 H	ADR 1	Device address: ADR (1,0)
D1256 high byte	'1'	31 H	ADR 0	
D1257 low byte	'0'	30 H	CMD 1	CMD (1,0) Control parameter
D1257 high byte	'5'	35H	CMD 0	
D1258 low byte	'0'	30 H	Data Address	
D1258 high byte	'5'	35 H		
D1259 low byte	'0'	30 H		
D1259 high byte	'0'	30 H		
D1260 low byte	'F'	46 H	High byte to be force ON/OFF	
D1260 high byte	'F'	46 H		
D1261 low byte	'0'	30H	Low byte to be force ON/OFF	
D1261 high byte	'0'	30 H		
D1262 low byte	'6'	36 H	LRC CHK 1 LRC CHK 0	Checksum: LRC CHK (0,1)
D1262 high byte	'F'	46 H		

Registers for received data (responding messages)

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	'5'	35H	CMD 0
D1072 low byte	'0'	30 H	Data Address
D1072 high byte	'5'	35 H	
D1073 low byte	'0'	30 H	
D1073 high byte	'0'	30 H	
D1074 low byte	'F'	46 H	High byte to be force ON/OFF
D1074 high byte	'F'	46 H	
D1075 low byte	'0'	30H	Low byte to be force ON/OFF
D1075 high byte	'0'	30 H	
D1076 low byte	'6'	36 H	LRC CHK 1
D1076 high byte	'F'	46 H	LRC CHK 0

RTU mode (M1143 = ON)

When X0 = ON, MODRW instruction executes the function specified by Function Code 05

PLC1⇨ PLC2, PLC1 sends: "**01 05 0500 FF00 8C F6**"

PLC2 ⇨PLC1, PLC1 receives: "**01 05 0500 FF00 8C F6**"

Registers for data to be sent (sending messages)

Register	Data	Descriptions
D1256 Low byte	01 H	Address
D1257 Low byte	05 H	Function
D1258 Low byte	05 H	Data Address
D1259 Low byte	00 H	
D1260 Low byte	FF H	Data content (ON = FF00H)
D1261 Low byte	00 H	
D1262 Low byte	8C H	CRC CHK Low
D1263 Low byte	F6 H	CRC CHK High

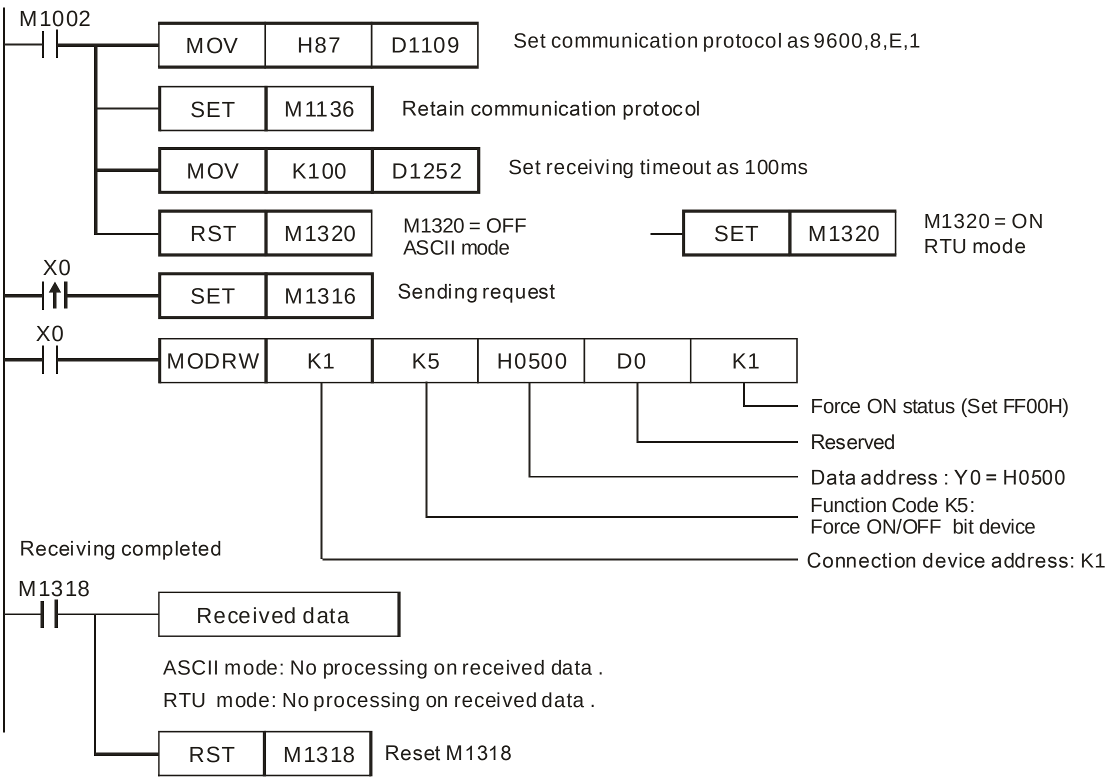
Registers for received data (responding messages)

Register	Data	Descriptions
D1070 Low byte	01 H	Address
D1071 Low byte	05 H	Function
D1072 Low byte	05 H	Data Address
D1073 Low byte	00 H	

Register	Data	Descriptions
D1074 Low byte	FF H	Data content (ON = FF00H)
D1075 Low byte	00 H	
D1076 Low byte	8C H	CRC CHK Low
D1077 Low byte	F6 H	CRC CHK High

Program example 6: COM1(RS-232) / COM3(RS-485), Function Code H05

1. Function Code K5 (H05): Force ON/OFF bit device.
2. PLC1 connects PLC2: (M1320 = OFF, ASCII Mode), (M1320 = ON, RTU Mode)
3. **n** = 1 indicates Force ON (set FF00H) and **n** = 0 indicates Force OFF (set 0000H)
4. PLC COM1/COM3 will not process the received data.
5. Take the connection between PLC1 (PLC COM3) and PLC2(PLC COM1) for example, the tables below explains the status when PLC1 reads Y0~Y17 of PLC2
   - If PLC1 applies COM1 for communication, the below program can be usable by changing:
     1. D1109→D1036: communication protocol
     2. M1136→M1138: retain communication setting
     3. D1252→D1249: Set value for data receiving timeout
     4. M1320→M1139: ASCII/RTU mode selection
     5. M1316→M1312: sending request
     6. M1318→M1314: receiving completed flag



ASCII mode (COM3: M1320 = OFF, COM1: M1139 = OFF):

When X0 = ON, MODRW instruction executes the function specified by Function Code 05

PLC1 ⇨ PLC2, PLC sends: "**01 05 0500 FF00 6F**"

PLC2 ⇨ PLC1, PLC receives: "**01 05 0500 FF00 6F**"

(No data processing on received data)

RTU mode (COM3: M1320 = ON, COM1: M1139 = ON):

When X0 = ON, MODRW instruction executes the function specified by Function Code 05

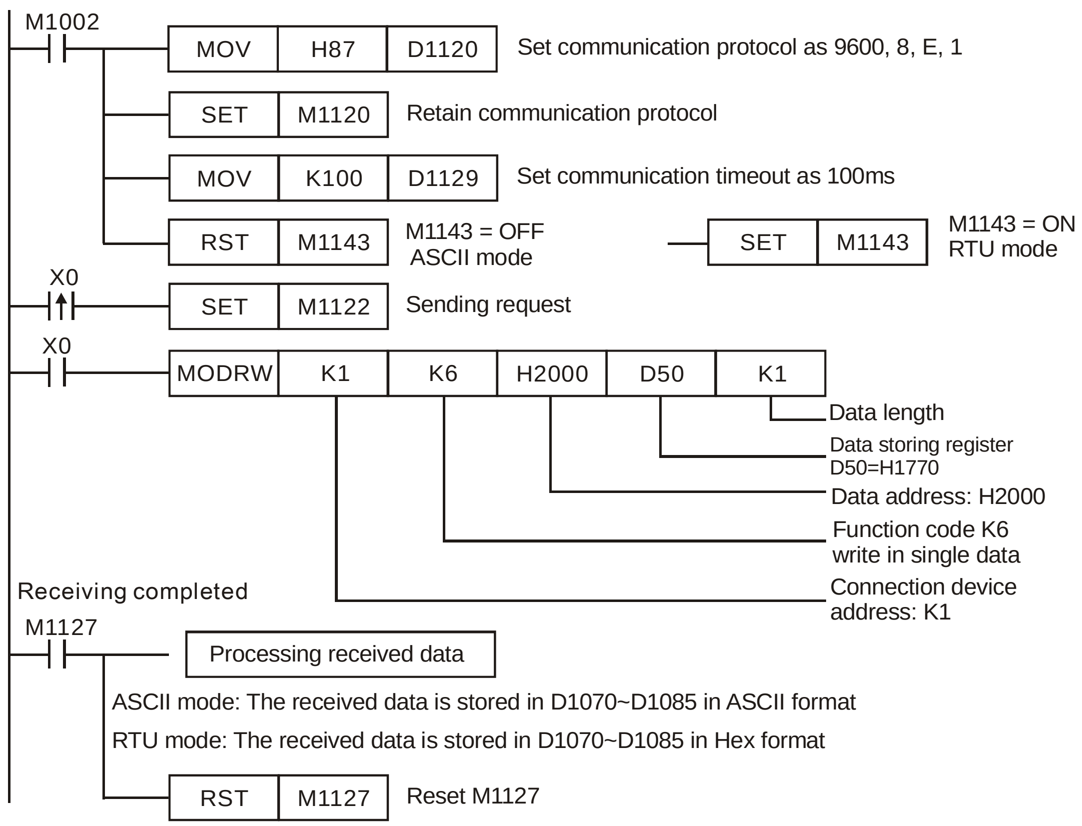
PLC1⇨ PLC2, PLC1 sends: "**01 05 0500 FF00 8C F6**"

PLC2 ⇨PLC1, PLC1 receives: "**01 05 0500 FF00 8C F6**"

(No data processing on received data)

Program Example 7: COM2(RS-485), Function Code H06

1. Function code K6 (H06): Write in single word device.
2. Set the value to be written into VFD-B in the register specified by operand **S**.
3. For ASCII or RTU mode, PLC COM2 stores the data to be sent in D1256~D1295, and received data in D1070~D1085.
4. Take the connection between PLC (PLC COM2) and VFD-B for example, the tables below explains the status when PLC reads status of VFD-B. (M1143 = OFF, ASCII Mode) (M1143 = ON, RTU Mode)



ASCII mode (M1143 = OFF)

When X0 = ON, MODRW instruction executes the function specified by Function Code 06

PLC ⇨ VFD-B, PLC sends: **“01 06 2000 1770 52”**

VFD-B ⇨ PLC, PLC receives: **“01 06 2000 1770 52”**

Registers for data to be sent (sending messages)

Register	Data		Descriptions	
D1256 Low byte	‘0’	30 H	ADR 1	Device address of VFD-B: ADR (1,0)
D1256 High byte	‘1’	31 H	ADR 0	
D1257 Low byte	‘0’	30 H	CMD 1	Control parameter: CMD (1,0)
D1257 High byte	‘6’	36 H	CMD 0	
D1258 Low byte	‘2’	32 H	Data Address	
D1258 High byte	‘0’	30 H		
D1259 Low byte	‘0’	30 H		
D1259 High byte	‘0’	30 H		
D1260 Low byte	‘1’	31 H	Data content	H1770 = K6000. The content of register D50
D1260 High byte	‘7’	37 H		
D1261 Low byte	‘7’	37 H		
D1261 High byte	‘0’	30 H		
D1262 Low byte	‘5’	35 H	LRC CHK 1	Checksum: LRC CHK (0,1)
D1262 High byte	‘2’	32 H	LRC CHK 0	

Registers for received data (responding messages)

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	‘2’	32 H	Data Address
D1072 High byte	‘0’	30 H	
D1073 Low byte	‘0’	30 H	
D1073 High byte	‘0’	30 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

RTU mode (M1143 = ON)

When X0 = ON, MODRW instruction executes the function specified by Function Code 06

PLC ⇨ VFD-B, PLC sends: **“01 06 2000 1770 8C 1E”**

VFD-B → PLC, PLC receives: “**01 06 2000 1770 8C 1E**”

Registers for data to be sent (sending messages)

<table>
<tr><th>Register</th><th>Data</th><th colspan="2">Descriptions</th></tr>
<tr><td>D1256 Low byte</td><td>01 H</td><td colspan="2">Address</td></tr>
<tr><td>D1257 Low byte</td><td>06 H</td><td colspan="2">Function</td></tr>
<tr><td>D1258 Low byte</td><td>20 H</td><td colspan="2" rowspan="2">Data Address</td></tr>
<tr><td>D1259 Low byte</td><td>00 H</td></tr>
<tr><td>D1260 Low byte</td><td>17 H</td><td rowspan="2">Data content</td><td rowspan="2">H1770 = K6000.
The content of register D50</td></tr>
<tr><td>D1261 Low byte</td><td>70 H</td></tr>
<tr><td>D1262 Low byte</td><td>8C H</td><td colspan="2">CRC CHK Low</td></tr>
<tr><td>D1263 Low byte</td><td>1E H</td><td colspan="2">CRC CHK High</td></tr>
</table>

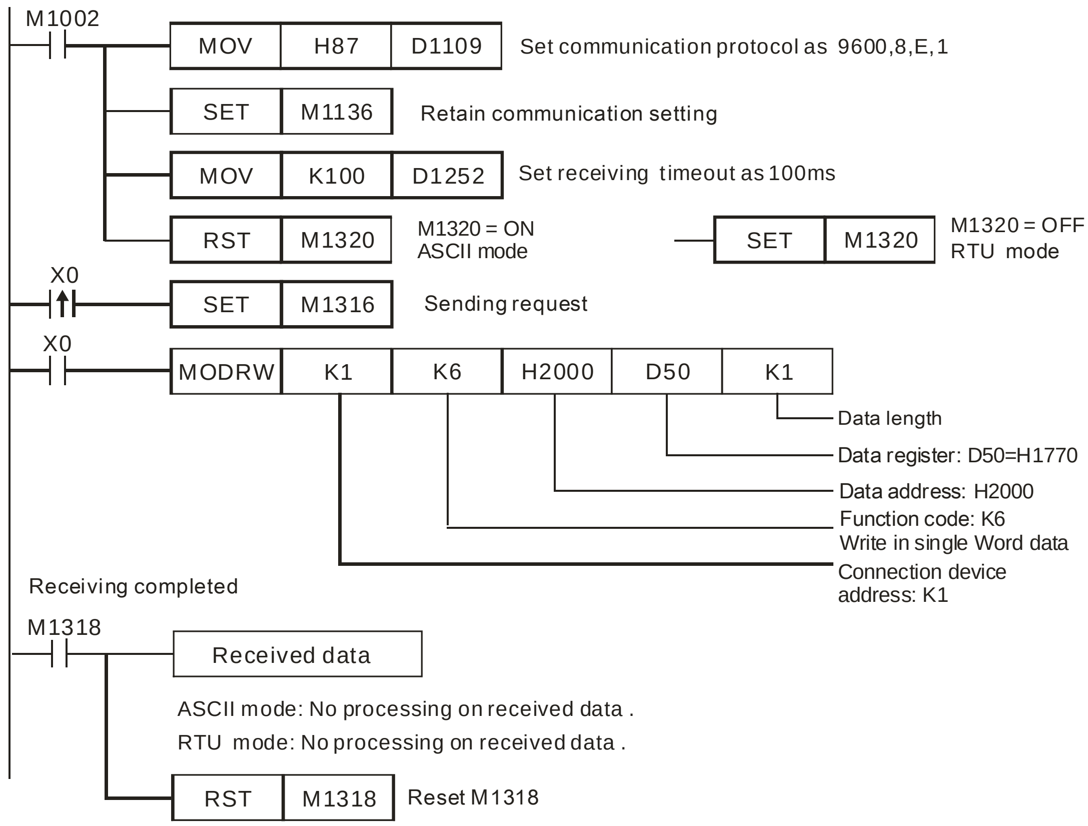
Registers for received data (responding messages)

<table>
<tr><th>Register</th><th>Data</th><th>Descriptions</th></tr>
<tr><td>D1070 Low byte</td><td>01 H</td><td>Address</td></tr>
<tr><td>D1071 Low byte</td><td>06 H</td><td>Function</td></tr>
<tr><td>D1072 Low byte</td><td>20 H</td><td rowspan="2">Data Address</td></tr>
<tr><td>D1073 Low byte</td><td>00 H</td></tr>
<tr><td>D1074 Low byte</td><td>17 H</td><td rowspan="2">Data content</td></tr>
<tr><td>D1075 Low byte</td><td>70 H</td></tr>
<tr><td>D1076 Low byte</td><td>8C H</td><td>CRC CHK Low</td></tr>
<tr><td>D1077 Low byte</td><td>1E H</td><td>CRC CHK High</td></tr>
</table>

Program example 8: COM1 (RS-232) / COM3 (RS-485), Function Code H06

1. Function code K6 (H06): Write in single Word device.
2. Set the value to be written into VFD-B in the register specified by operand **S**.
3. PLC COM1/COM3 will not process the received data.
4. Take the connection between PLC (PLC COM3) and VFD-B for example, the tables below explains the status when PLC COM3 writes in single Word device in VFD-B (M1320 = OFF, ASCII Mode), (M1320 = ON, RTU Mode)
   - If PLC applies COM1 for communication, the below program can be usable by changing:
     1. D1109→D1036: communication protocol
     2. M1136→M1138: retain communication setting
     3. D1252→D1249: Set value for data receiving timeout
     4. M1320→M1139: ASCII/RTU mode selection

5. M1316→M1312: sending request
6. M1318→M1314: receiving completed flag



ASCII mode (COM3: M1320 = OFF, COM1: M1139 = OFF):

When X0 = ON, MODRW instruction executes the function specified by Function Code 06

PLC ⇨ VFD-B, PLC sends: **"01 06 2000 1770 52"**

VFD-B ⇨ PLC, PLC receives: **"01 06 2000 1770 52"**

(No data processing on received data)

RTU mode (COM3: M1320 = ON, COM1: M1139 = ON)

When X0 = ON, MODRW instruction executes the function specified by Function Code 06

PLC ⇨ VFD-B, PLC sends: **"01 06 2000 1770 8C 1E"**

VFD-B → PLC, PLC receives: "**01 06 2000 1770 8C 1E**"

(No data processing on received data)

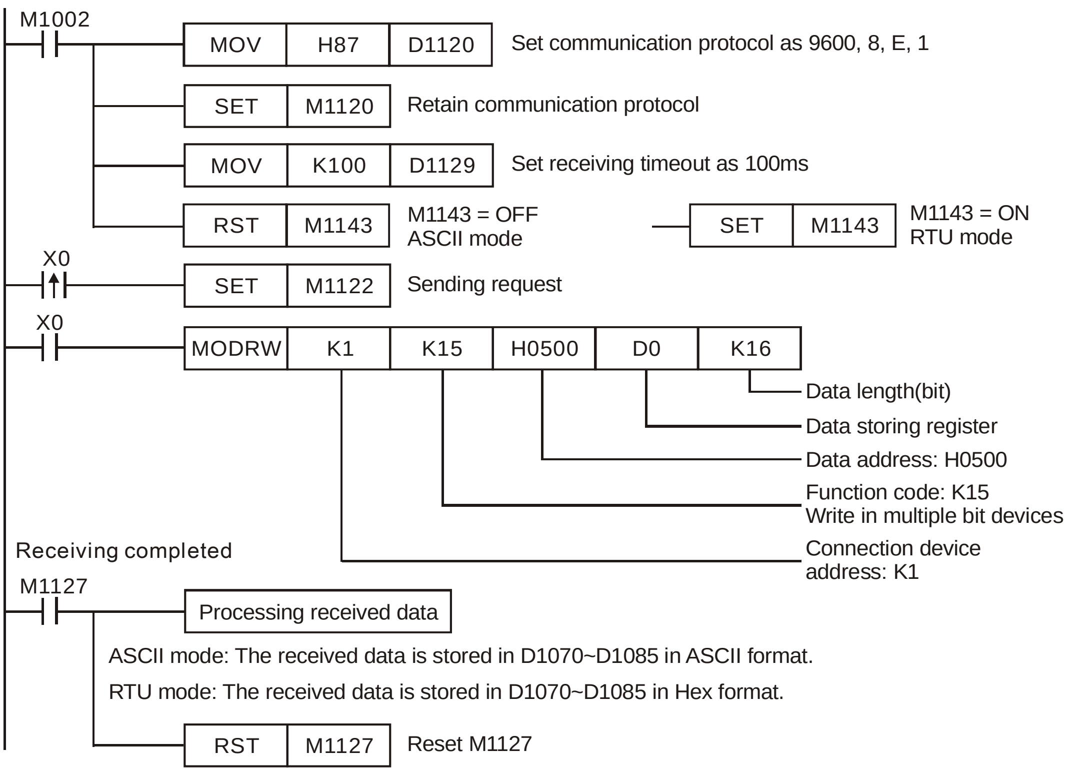
Program Example 9: COM2 (RS-485), Function Code H0F

1. Function code K15 (H0F): write in multiple bit devices. Up to 64bits can be written.
2. PLC1 connects to PLC2: (M1143 = OFF, ASCII Mode), (M1143 = ON, RTU Mode)
3. For ASCII or RTU mode, PLC COM2 stores the data to be sent in D1256~D1295 and the received data in D1070~D1085.

4. Take the connection between PLC1 (PLC COM2) and PLC2 (PLC COM1) for example, the tables below explain the status when PLC1 force ON/OFF Y0~Y17 of PLC2.

Set value: K4Y0=1234H

Device	Status	Device	Status	Device	Status	Device	Status
Y0	OFF	Y1	OFF	Y2	ON	Y3	OFF
Y4	ON	Y5	ON	Y6	OFF	Y7	OFF
Y10	OFF	Y11	ON	Y12	OFF	Y13	OFF
Y14	ON	Y15	OFF	Y16	OFF	Y17	OFF



ASCII mode (M1143 = OFF)

When X0 = ON, MODRW instruction executes the function specified by Function Code H0F.

PLC1 ⇨ PLC2, PLC sends: “ **01 0F 0500 0010 02 3412 93** ”

PLC2 ⇨ PLC1, PLC receives: “ **01 0F 0500 0010 DB** ”

Registers for data to be sent (sending messages)

Register	Data		Descriptions	
D1256 下	‘0’	30 H	ADR 1	Device address: ADR (1,0)
D1256 上	‘1’	31 H	ADR 0	
D1257 下	‘0’	30 H	CMD 1	Control parameter: CMD (1,0)
D1257 上	‘F’	46 H	CMD 0	
D1258 下	‘0’	30 H	Data Address	
D1258 上	‘5’	35 H		
D1259 下	‘0’	30 H		
D1259 上	‘0’	30 H		

Register	Data		Descriptions	
D1260 下	‘0’	30 H	Number of Data (count by bit)	
D1260 上	‘0’	30 H		
D1261 下	‘1’	31H		
D1261 上	‘0’	30 H		
D1262 下	‘0’	30 H	Byte Count	
D1262 上	‘2’	32 H		
D1263 下	‘3’	33 H	Data contents	1234H Content of register D0
D1263 上	‘4’	46 H		
D1264 下	‘1’	33 H		
D1264 上	‘2’	46 H		
D1265 下	‘9’	39 H	LRC CHK 1	Checksum: LRC CHK (0,1)
D1265 上	‘3’	33 H	LRC CHK 0	

Registers for received data (responding messages)

Register	Data		Descriptions
D1070 下	‘0’	30 H	ADR 1
D1070 上	‘1’	31 H	ADR 0
D1071 下	‘0’	31 H	CMD 1
D1071 上	‘F’	46 H	CMD 0
D1072 下	‘0’	30 H	Data Address
D1072 上	‘5’	35 H	
D1073 下	‘0’	30 H	
D1073 上	‘0’	30 H	
D1074 下	‘0’	30 H	Number of Data(count by bit)
D1074 上	‘0’	30 H	
D1075 下	‘1’	31 H	
D1075 上	‘0’	30 H	
D1076 下	‘D’	44 H	LRC CHK 1
D1076 上	‘B’	42 H	LRC CHK 0

RTU mode (M1143 = ON)

When X0 = ON, MODRW instruction executes the function specified by Function Code H0F

PLC1 ⇨ PLC2，PLC1 sends: “**01 0F 0500 0010 02 34 12 21 ED**”

PLC2 ⇨ PLC1，PLC1 receives: “**01 0F 0500 0010 54 CB**”

Registers for data to be sent (sending messages)

Register	Data	Descriptions	
D1256 下	01 H	Address	
D1257 下	0F H	Function	
D1258 下	05 H	Data Address	
D1259 下	00 H		
D1260 下	00 H	Number of Data(count by bit)	
D1261 下	10 H		
D1262 下	02 H	Byte Count	
D1263 下	34 H	Data content 1	Content of D0: H34
D1264 下	12 H	Data content 2	Content of D1: H12
D1265 下	21 H	CRC CHK Low	
D1266 下	ED H	CRC CHK High	

Registers for received data (responding messages)

Register	Data	Descriptions
D1070 下	01 H	Address
D1071 下	0F H	Function
D1072 下	05 H	Data Address
D1073 下	00 H	
D1074 下	00 H	Number of Data(count by bit)
D1075 下	10H	
D1076 下	54H	CRC CHK Low
D1077 下	CB H	CRC CHK High

Program example 10: COM1 (RS-232) / COM3 (RS-485), Function Code H0F

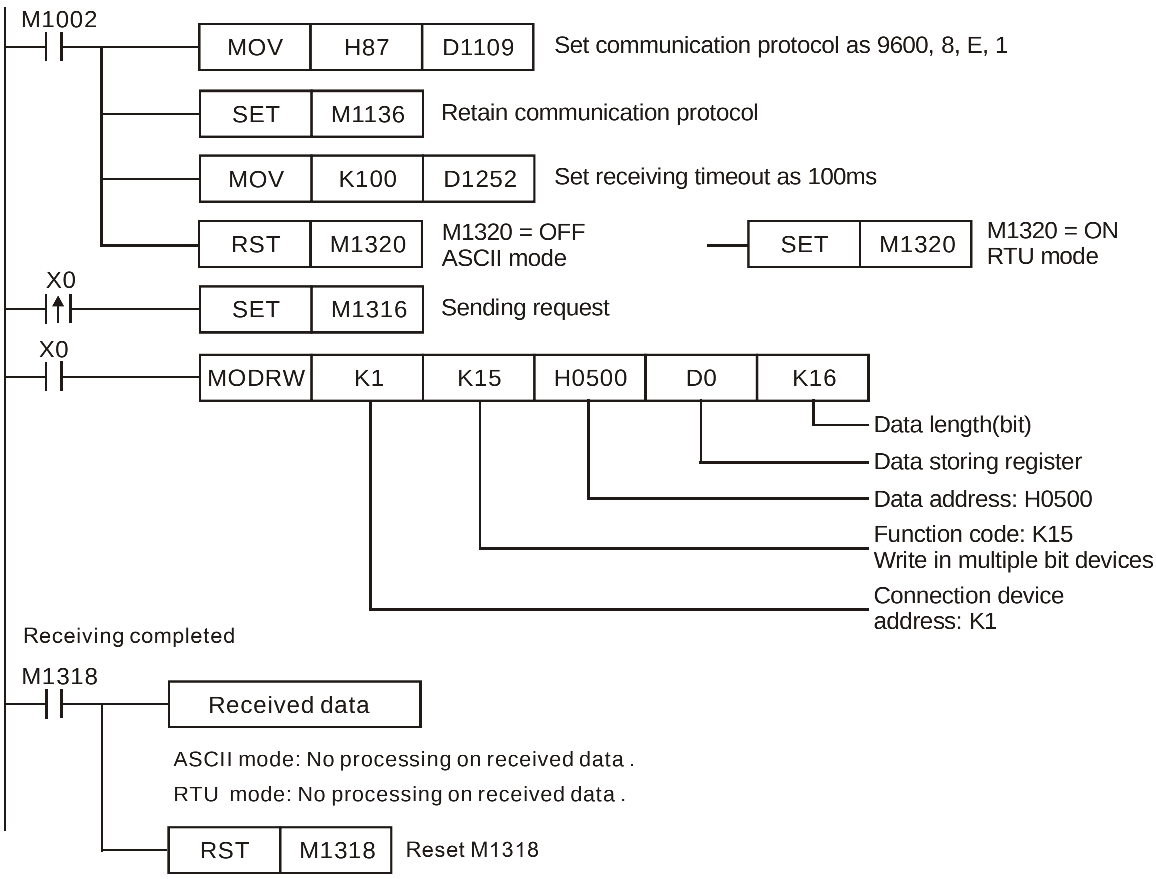
1. Function code K15 (H0F): write in multiple bit devices. Up to 64 bits can be written
2. PLC1 connects to PLC2: (M1143 = OFF, ASCII mode), (M1143 = ON, RTU mode)
3. PLC COM1/COM3 will not process the received data.
4. Take the connection between PLC1 (PLC COM3) and PLC2 (PLC COM1) for example, the tables below explain the status when PLC1 force ON/OFF Y0~Y17 of PLC2.

   Set value: K4Y0=1234H

Device	Status	Device	Status	Device	Status	Device	Status
Y0	OFF	Y1	OFF	Y2	ON	Y3	OFF
Y4	ON	Y5	ON	Y6	OFF	Y7	OFF
Y10	OFF	Y11	ON	Y12	OFF	Y13	OFF
Y14	ON	Y15	OFF	Y16	OFF	Y17	OFF

- If PLC applies COM1 for communication, the below program can be usable by changing:
  1. D1109→D1036: communication protocol
  2. M1136→M1138: retain communication setting

3. D1252→D1249: Set value for data receiving timeout
4. M1320→M1139: ASCII/RTU mode selection
5. M1316→M1312: sending request
6. M1318→M1314: receiving completed flag



ASCII mode (COM3: M1320 = OFF, COM1: M1139 = OFF):

When X0 = ON, MODRW executes the function specified by Function Code H0F

PLC1 ⇨ PLC2, PLC sends: " **01 0F 0500 0010 02 3412 93** "

PLC2 ⇨ PLC1, PLC receives: " **01 0F 0500 0010 DB** "

(No data processing on received data)

RTU mode (COM3: M1320 = ON, COM1: M1139 = ON):

When X0 = ON, MODRW executes the function specified by Function Code H0F

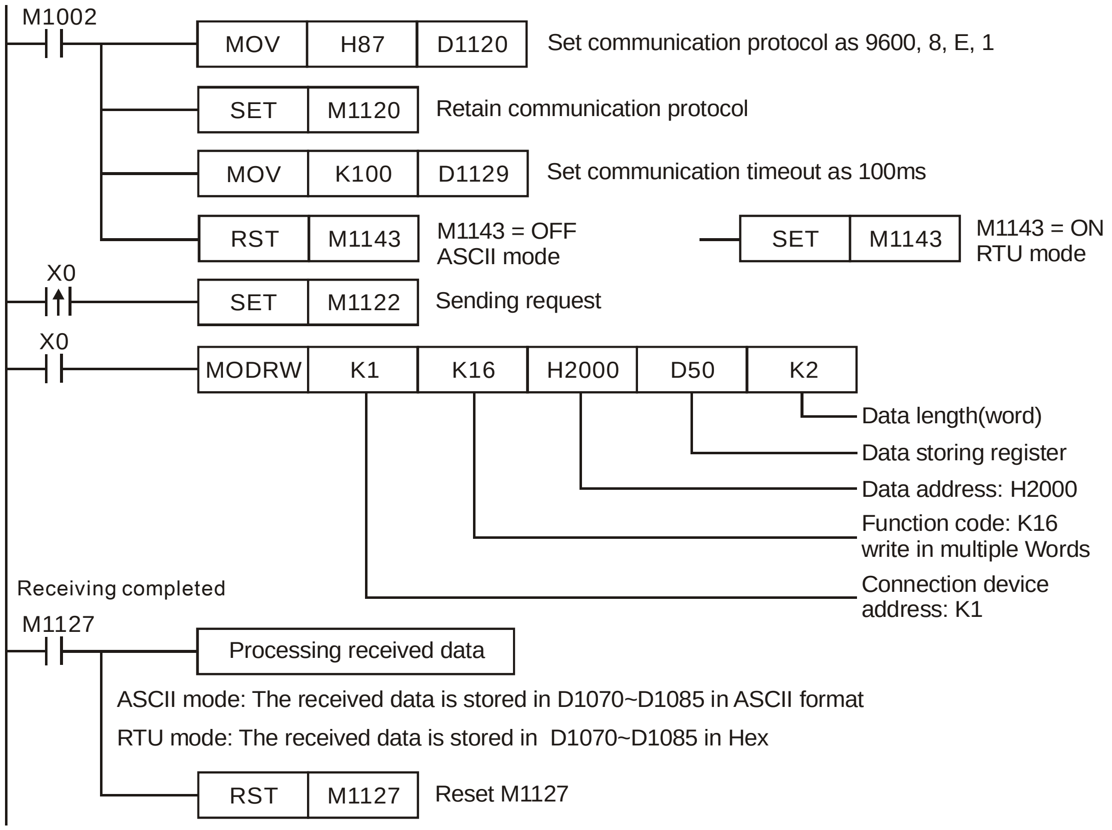
PLC1 ⇨ PLC2, PLC1 sends: "**01 0F 0500 0010 02 34 12 21 ED**"

PLC2 ⇨ PLC1, PLC1 receives: "**01 0F 0500 0010 54 CB**" ,

(No data processing on received data)

Program Example 11: COM2 (RS-485), Function Code H10

1. Function code K16 (H10): Write in multiple Word devices. Up to 16 Words can be written. For PLC COM2 ASCII mode, only 8 words can be written.
2. For ASCII or RTU mode, PLC COM2 stores the data to be sent in D1256~D1295, and the received data in D1070~D1085.
3. Take the connection between PLC COM2 and VFD-B AC motor drive for example, the tables below explain the status when PLC COM2 writes multiple word devices in VFD-B.



ASCII mode (M1143 = OFF)

When X0 = ON, MODRW instruction executes the function specified by Function Code H10

PLC ⇨VFD-B, PLC transmits: **"01 10 2000 0002 04 1770 0012 30"**

VFD⇨PLC, PLC receives: **"01 10 2000 0002 CD"**

Registers for data to be sent (sending messages)

Register	Data		Descriptions	
D1256 Low byte	'0'	30 H	ADR 1	Address of VFD: ADR (1,0)
D1256 High byte	'1'	31 H	ADR 0	
D1257 Low byte	'1'	31 H	CMD 1	Control parameter: CMD (1,0)
D1257 High byte	'0'	30 H	CMD 0	
D1258 Low byte	'2'	32 H	Data Address	

Register	Data		Descriptions	
D1258 High byte	‘0’	30 H		
D1259 Low byte	‘0’	30 H		
D1259 High byte	‘0’	30 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: H0012(K18)
D1265 High byte	‘0’	30 H		
D1266 Low byte	‘1’	31 H		
D1266 High byte	‘2’	32 H		
D1267 Low byte	‘3’	33 H	LRC CHK 1	LRC CHK (0,1) is error check
D1267 High byte	‘0’	30 H	LRC CHK 0	

Registers for received data (responding 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	‘2’	32 H	Data Address
D1072 High byte	‘0’	30 H	
D1073 Low byte	‘0’	30 H	
D1073 High byte	‘0’	30 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	‘C’	43 H	LRC CHK 1
D1076 High byte	‘D’	44 H	LRC CHK 0

RTU mode (M1143 = ON)

When X0 = ON, MODRW instruction executes the function specified by Function Code H10

PLC ⇨VFD-B,PLC transmits: **“01 10 2000 0002 04 1770 0012 EE 0C”**

VFD-B⇨PLC, PLC receives: **”01 10 2000 0002 4A08”**

Registers for data to be sent (sending messages)

Register	Data	Descriptions	
D1256 Low byte	01 H	Address	
D1257 Low byte	10 H	Function	
D1258 Low byte	20 H	Data Address	
D1259 Low byte	00 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 D50: H1770 (K6000)
D1264 Low byte	70 H		
D1265 Low byte	00 H	Data content 2	The content of D51: H0012 (K18)
D1266 Low byte	12 H		
D1262 Low byte	EE H	CRC CHK Low	
D1263 Low byte	0C H	CRC CHK High	

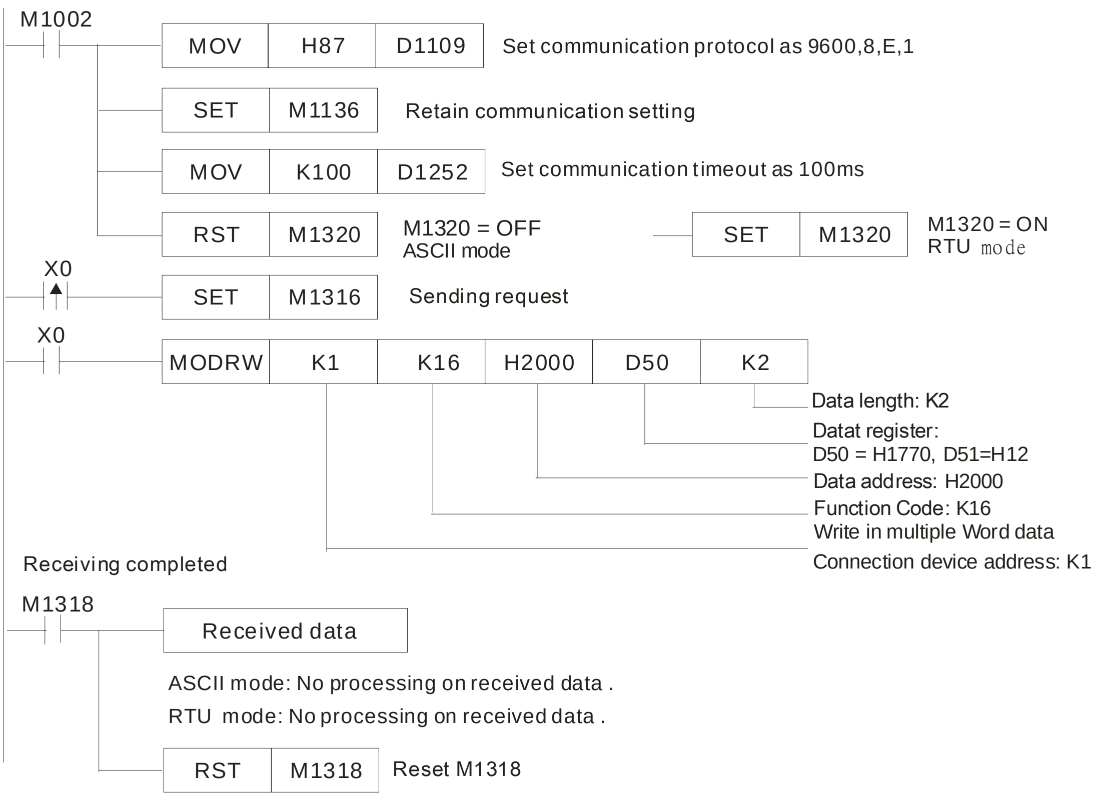
Registers for received data (responding messages)

Register	Data	Descriptions
D1070 Low byte	01 H	Address
D1071 Low byte	10 H	Function
D1072 Low byte	20 H	Data Address
D1073 Low byte	00 H	
D1074 Low byte	00 H	Number of Register
D1075 Low byte	02 H	
D1076 Low byte	4A H	CRC CHK Low
D1077 Low byte	08 H	CRC CHK High

Program example 12: COM1 (RS-232) / COM3 (RS-485), Function Code H10

1. Function code K16 (H10): Write in multiple Word devices. Up to 16 Words can be written. For PLC COM2 ASCII mode, only 8 words can be written.
2. PLC COM1/COM3 will not process the received data
3. Take the connection between PLC COM3 and VFD-B for example, the tables below explain the status when PLC COM3 writes multiple Words in VFD-B. (M1320 = OFF, ASCII mode) (M1320 = ON, RTU mode)
   - If PLC applies COM1 for communication, the below program can be usable by changing:
     1. D1109→D1036: communication protocol

2. M1136→M1138: retain communication setting
3. D1252→D1249: Set value for data receiving timeout
4. M1320→M1139: ASCII/RTU mode selection
5. M1316→M1312: sending request
6. M1318→M1314: receiving completed flag



- ASCII mode (COM3: M1320 = OFF, COM1: M1139 = OFF):

  When X0 = ON, MODRW executes the function specified by Function Code H10

  PLC ⇨VFD-B, PLC sends: **"01 10 2000 0002 04 1770 0012 30"**

  VFD⇨PLC, PLC receives: **"01 10 2000 0002 CD"**

  (No processing on received data)

- RTU Mode (COM3: M1320=On, COM1: M1139=On):

  When X0 = ON, MODRW executes the function specified by Function Code H10

  PLC ⇨VFD-B,PLC sends: "0**1 10 2000 0002 04 1770 0012 EE 0C"**

  VFD-B⇨PLC, PLC receives :" 01 10 2000 0002 4A08"

  (No processing on received data)

API	Mnemonic			Operands	Function	Controllers
154	D	RAND	P	S1 S2 D	Random number	ES2/EX2 SS2 SA2 SX2

Type / OP	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 DRAND, DRANDP: 13 steps
S_1					*	*	*	*	*	*	*	*	*	*	*	
S_2					*	*	*	*	*	*	*	*	*	*	*	
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

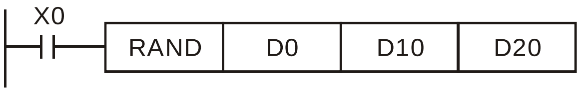
S_1: Lower bound of the random number **S_2**: Upper bound of the random number **D**: Operation result

Explanations:

1. The range of 16-bit operands **S_1**, **S_2**: K0≦**S_1** , **S_2**≦K32,767; the range of 32-bit operands **S_1**, **S_2**: K0≦**S_1** , **S_2**≦K2,147,483,647.
2. Entering **S_1** > **S_2** will result in operation error. The instruction will not be executed at this time, M1067, M1068 = ON and D1067 records the error code 0E1A (HEX)

Program Example:

When X10 = ON, RAND will produce the random number between the lower bound D0 and upper bound D10 and store the result in D20.



API	Mnemonic		Operands	Function	Controllers
155	D	ABSR	S D1 D2	Absolute position read	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DABSR: 13 steps
S	*	*	*	*												
D_1		*	*	*												
D_2								*	*	*	*	*	*	*		

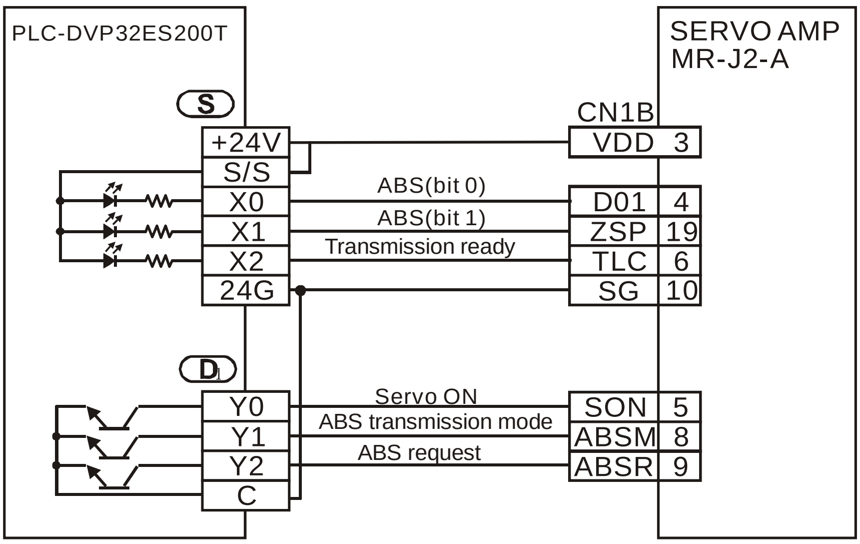
PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Input signal from servo (occupies 3 consecutive devices) **D_1**: Control signal for controlling servo (occupies 3 consecutive devices) **D_2**: Absolute position data (32-bit) read from servo

Explanations:

1. This instruction reads the absolute position (ABS) of servo drive with absolute position check function, e.g. MITSUBISHI MR-J2.
2. Only 32-bit instruction is applicable for ABSR instruction (DABSR) and it can only be used ONCE in the program.
3. **S**: input signal from servo. 3 consecutive devices **S**, **S** +1, **S** +2 are occupied. **S** and **S** +1 are connected to the ABS (bit0, bit1) of servo for data transmitting. **S** +2 is connected to servo for indicating transmission data being prepared.
4. **D_1**: control signal for controlling servo. 3 consecutive devices **D_1**, **D_1**+1, **D_1**+2 are occupied. **D_1** is connected to servo ON (SON) of servo, **D_1**+1 is connected to ABS transmission mode of servo and **D_1**+2 is connected to ABS request.

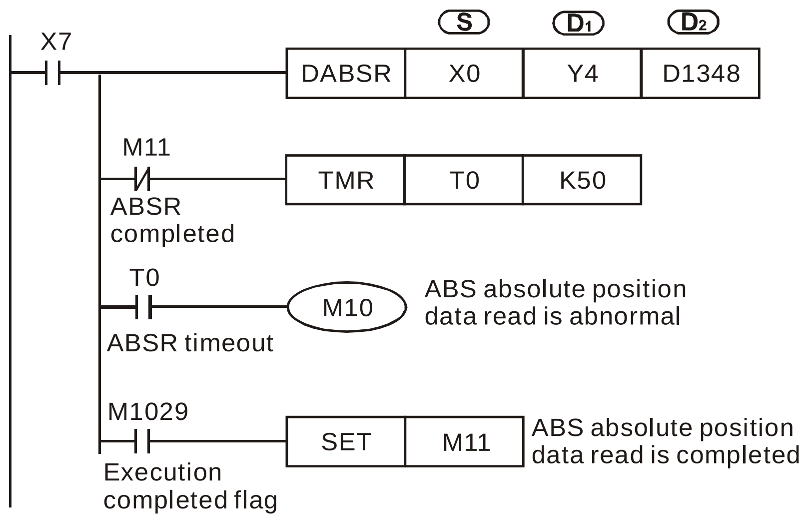


5. **D_2**: Absolute position data (32-bit) read from servo. 2 consecutive devices **D_2**, **D_2**+1 are occupied. **D_2** is low word and **D_2**+1 is high word. When DABSR instruction is completed, M1029 will be ON. M1029 has to be reset by users.

6. Please use NO contact as the drive contact of DABSR instruction. If the drive contact is OFF during the execution of DABSR, the instruction will be stopped and errors will occur on read data.
7. If the drive contact of DABSR instruction turns OFF after the instruction is completed, the servo ON (SON) signal connected to **D_1** will also turn OFF and the operation will be disabled.
8. Flags: For the descriptions of M1010, M1029, M1102, M1103, M1334, M1335, M1336, M1337, M1346, please refer to **Points to Note.**

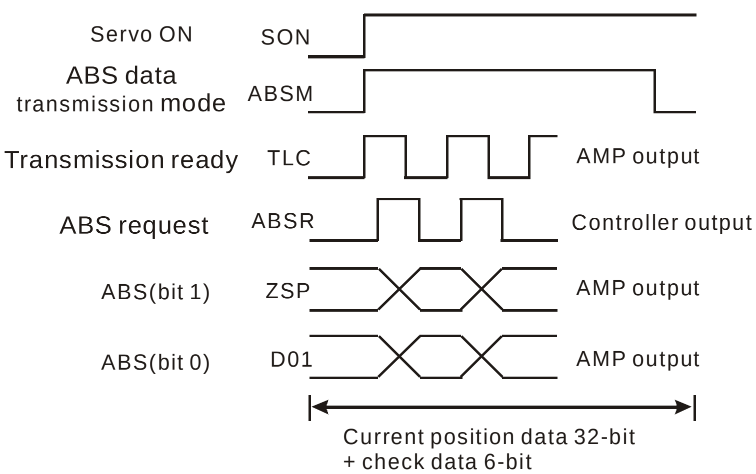
Program Example:

1. When X7 = ON, the 32-bit absolute position data read from servo will be stored in the registers storing present value of CH0 pulse output (D1348, D1349). At the same time, timer T10 is enabled and starts to count for 5 seconds. If the instruction is not completed within 5 seconds, M10 will be ON, indicating operation errors.
2. When enabling the connection to the system, please synchronize the power input of DVP-PLC and SERVO AMP or activate the power of SERVO AMP earlier than DVP-PLC.



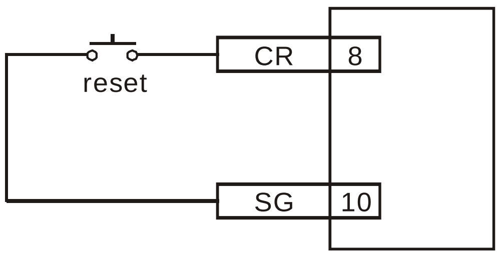
Points to note:

3. Timing diagram of the operation of DABSR instruction:



4. When DABSR instruciton executes, servo ON (SON) and ABS data transmission mode are driven for output.
5. By “transmission ready” and “ABS request” signals, users can confirm the transmitting and receiving status of both sides as well as processing the transmission of the 32-bit ABS position data and the 6-bit check data..
6. Data is transmitted by ABS (bit0, bit1).
7. This instruction is applicable for servo drive with absolute position check function, e.g. MITSUBISHI MR-J2-A.
8. Select one of the following methods for the initial ABSR instruction:
   - Execute API 156 ZRN instruction with reset function to complete zero return.
   - Apply JOG function or manual adjustment to complete zero return, then input the reset signal to the servo. Please refer to the diagram below for the wiring method of reset signal. For the detailed wiring between DVP-PLC and Mitsubishi MR-J2-A, please refer to API 159 DRVA instruction.

Ex: Mitsubishi MR-J2-A



API	Mnemonic			Operands	Function	Controllers
156	D	ZRN		S1 S2 S3 D	Zero return	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices X	Y	M	S	Word Devices K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*		DZRN: 17 steps
S_2					*	*	*	*	*	*	*	*	*	*		
S_3	*															
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Target frequency for zero return **S_2**: JOG frequency for DOG **S_3**: input device for DOG **D**: Pulse output device

Explanations:

1. **S_1** (zero return speed): max. 100kHz. **S_2** (JOG speed for DOG) has to be lower than **S_1**. JOG speed for DOG also refers to the start frequency.
2. **S_3** and **D** operands have to be used as an input/output set according to the table below, i.e. when **S_3** is specified as X4, **D** has to be specified as Y0; also when **S_3** is specified as X6, **D** has to be specified as Y2.
3. M1307 enables (ON) / disables (OFF) left limit switch of CH0 (Y0, Y1) and CH1 (Y2, Y3). M1307 has to be set up before the instruction executes. M1305 and M1306 can reverse the pulse output direction on Y1 and Y3 and have to be set up before instruction executes. Associated left limit switch for CH0 (Y0, Y1) is X5; associated left limit switch for CH1 (Y2, Y3) is X7. All functions, input points and output points are arranged as follows:

<table>
<tr><th>Channel / Input</th><th>CH0(Y0,Y1)</th><th>CH1(Y2,Y3)</th><th>Remark</th></tr>
<tr><td>DOG point</td><td>X4</td><td>X6</td><td></td></tr>
<tr><td>Left limit switch (M1307 = ON)</td><td>X5</td><td>X7</td><td></td></tr>
<tr><td>Reverse pulse output direction</td><td>M1305</td><td>M1306</td><td></td></tr>
<tr><td>Zero point selection</td><td>M1106</td><td>M1107</td><td>Please refer to point 7 for the explanation.</td></tr>
<tr><td>M1346=On
Start output clear signals</td><td>Y4</td><td>Y5</td><td>Please refer to point 8 for the explanation.</td></tr>
<tr><td rowspan="2">D1312 != 0</td><td colspan="2">M1308 = Off
(seeking Z-phase signal)</td><td rowspan="2">Please refer to point 9 for the explanation.</td></tr>
<tr><td>X2</td><td>X3</td></tr>
<tr><td>D1312 != 0</td><td colspan="2">M1308 = On
(outputting the designated number of pulses)</td><td>Please refer to point 10 for the explanation.</td></tr>
</table>

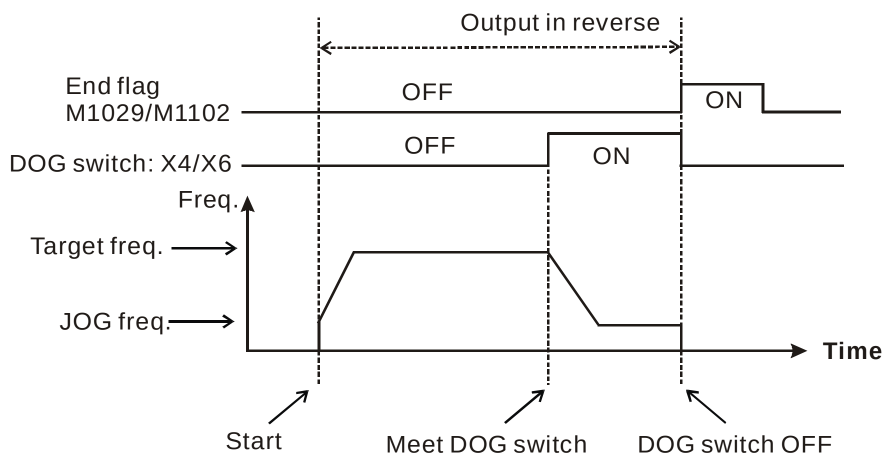
4. When **D** is specified as Y0, its direction signal output is Y1; when **D** is specified as Y2, its direction signal output is Y3.
5. When pulse output reaches zero point, pulse output execution completed flag M1029 (CH0), M1102 (CH1) is ON and the register indicating current position is reset to 0.
6. When DZRN instruction executes, external interrupt I400/I401(X4)) or I600/I601(X6) in program will be disabled until DZRN instruction is completed. Also. If left limit switch (X5 / X7) is enabled during instruction execution, external interrupt I500501(X5) or I700/I701(X7) will be disabled as well.
7. Zero point selection: the default position of zero point is on the left of DOG switch (the input point On→Off) (as mode 1 shows). If the user needs to change the zero point to the right of DOG switch, set ON M1106(CH0) or M1107(CH1) before DZRN instruction executes. (The function supports ES2/EX2 series, V1.20 or above.)
8. Start the pulse-clearing function of the output. When DOG leaves DOG switch and is going to stop, it will output another pulse (the width of On is about 20ms). When the pulse is On→Off, there will be a completed flag output. Please refer to state 4 for the timing diagram of this function. (The function supports ES2/EX2 series, V1.20 or above.)
9. When D1312 is not set to be 0, and M1308=Off, the function of seeking Z phase is started. When D1312 is a positive value (the maximum value is 10), it indicates that the search for Z-phase signal is toward the positive direction. When D1312 is a negative value (the minimum value is -10), it indicates that the search for Z-phase signal is toward the negative direction. For example, if D1312 is k-2, it means that DOG stops immediately after DOG leaves DOG switch and searches in the negative direction for second Z-phase signal (the fixed right-edge trigger) with JOG frequency.

Please refer to state 5 for the timing diagram of this function. (The function supports ES2/EX2 series of V1.20 or above, and SS2/SX2 series of V1.20 or above.)

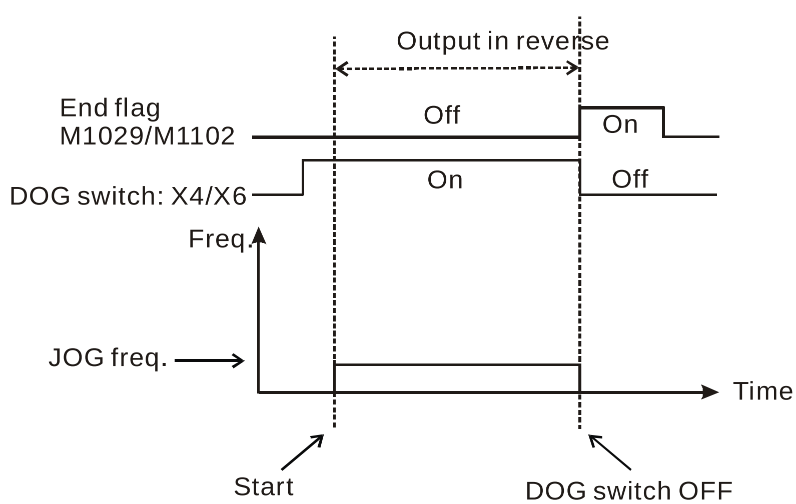
10. When D1312 is not set to be 0 and M1308=On, the function of outputting the designated number of pulses is started. When Dd1312 is a positive value (the maximum value is 30000), it indicates that the pulses are output in the positive direction. When D1312 is a negative value (the minimum value is -30000), it indicates that the pulses are output in the negative direction. For example, if D1312 is k-100, it means that DOG stops immediately after DOG leaves DOG switch and another 100 pulses will be output in the negative direction with JOG frequency. Please refer to state 6 for the timing diagram of this function. (The function supports ES2/EX2 series of V1.40 or above, and SS2/SX2 series of V1.20 or above.)

11. Timing Diagram:

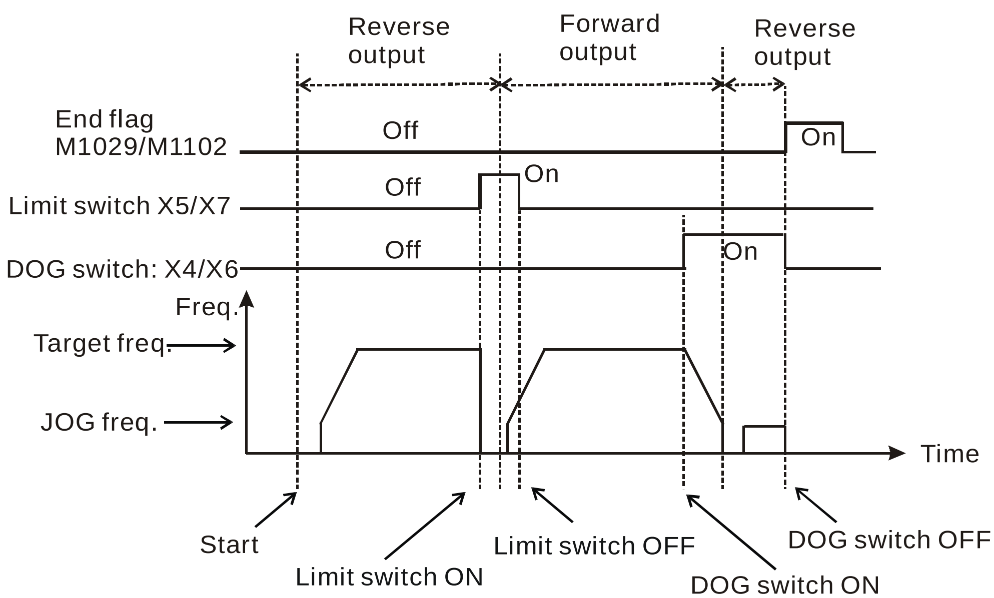
State 1: Current position at right side of DOG switch, pulse output in reverse, limit switch disabled.



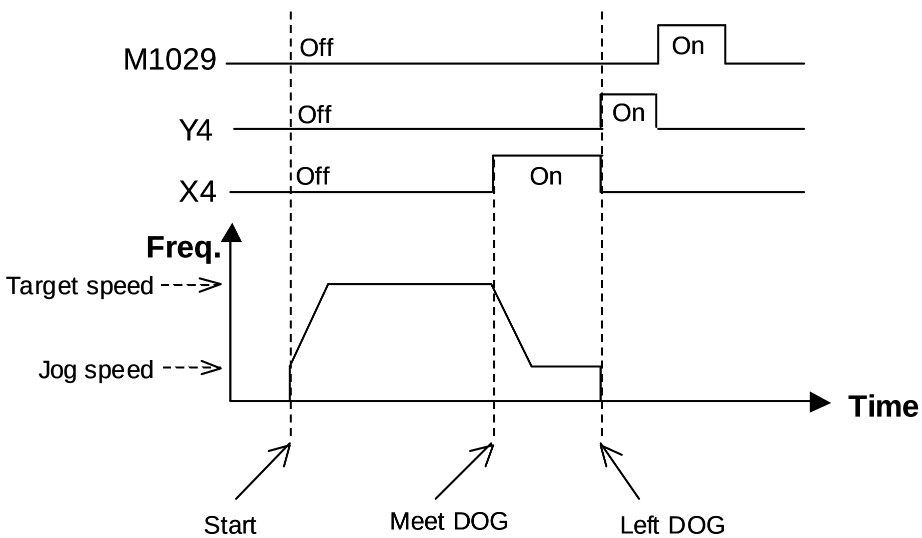
State 2: DOG switch is ON, pulse output in reverse, limit switch disabled.



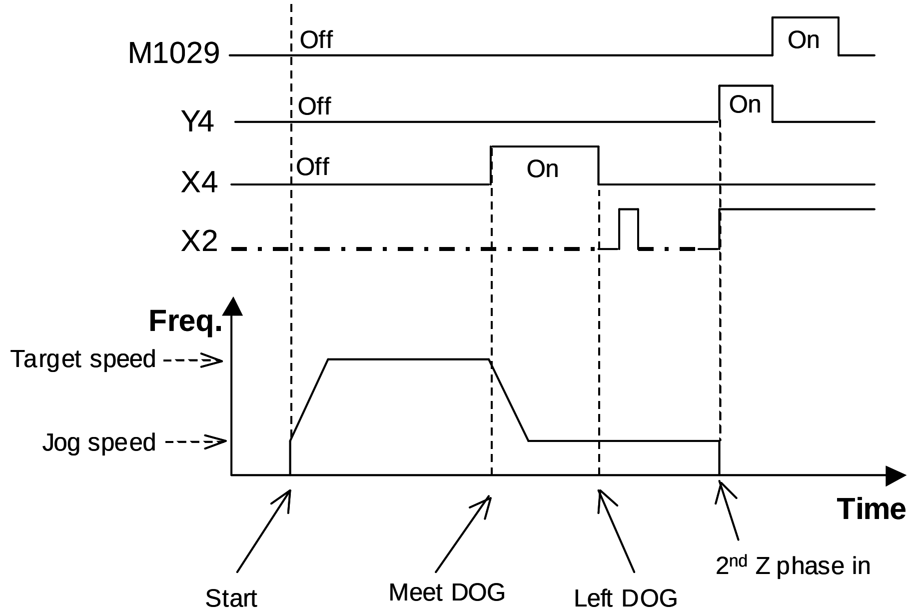
State 3: Current position at left side of zero point, pulse output in reverse, limit switch enabled.



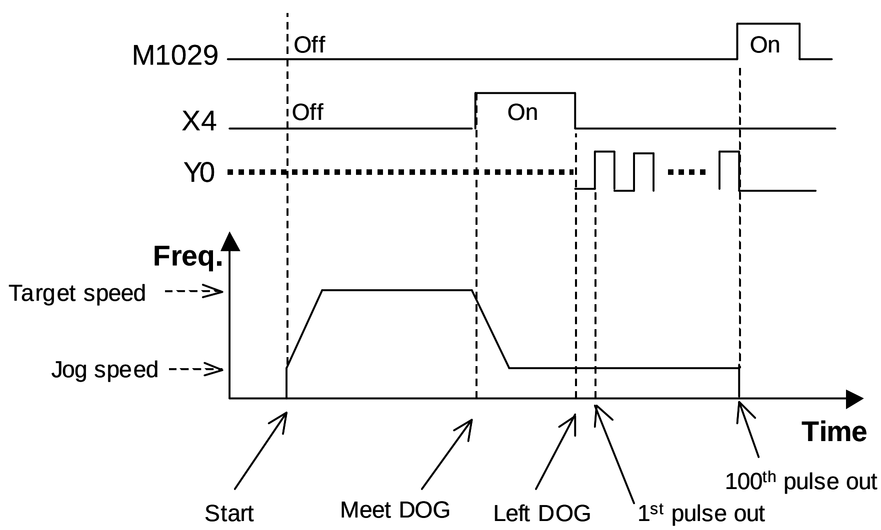
State 4: Current position at right side of zero point, M1346=On.



State 5: Current position at right side of zero point, D1312=-2, M1308=Off, M1346=On.

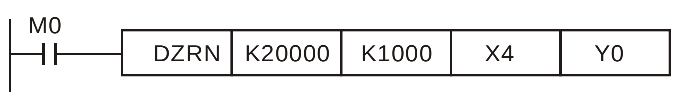


State 6: Current position at right side of zero point, D1312=-100, M1308=On.



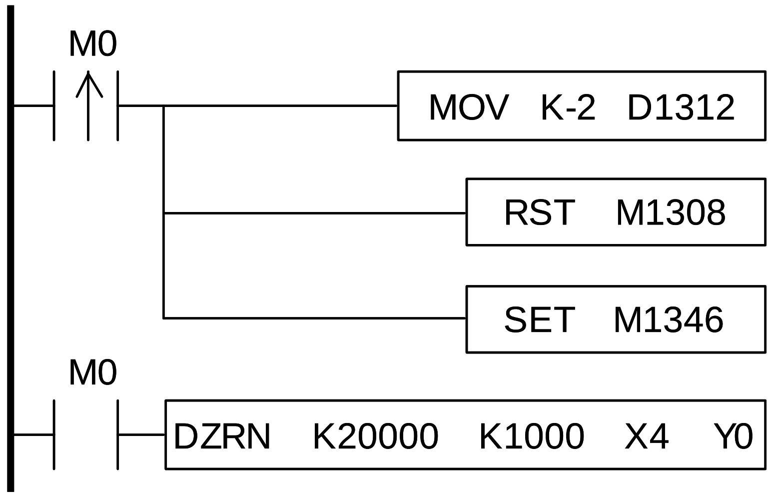
Program Example 1:

When M0 = ON, Y0 pulse output executes zero return with a frequency of 20kHz. When it reaches the DOG switch, X4 = ON and the frequency changes to JOG frequency of 1kHz. Y0 will then stop when X4 = OFF.



Program Example 2:

When M0 = ON, Y0 pulse output executes zero return with a frequency of 20kHz. When it reaches the DOG switch, X4 = ON and the frequency changes to JOG frequency of 1kHz. When X4 = OFF, it seeks the second X2(Z-phase) pulse input (right-edge trigger signal), and Y4 stops after a pulse (the width of On is 20ms) is output from it (M1029=On).



Points to note:

1. Associated Flags:

M1029	CH0 (Y0, Y1) pulse output execution completed
M1102	Y2/CH1 (Y2, Y3) pulse output execution completed
M1106:	Zero point selection. M1106=ON, change the zero point to the right of DOG switch for zero return on CH0
M1107:	Zero point selection. M1107=ON, change the zero point to the right of DOG switch for zero return on CH1
M1305:	Reverse Y1 pulse output direction in high speed pulse output instructions
M1306:	Reverse Y3 pulse output direction in high speed pulse output instructions
M1307:	For ZRN instruction, enable left limit switch
M1308:	Output specified pulses (D1312) or seek Z phase signal when zero point is achieved.
M1346:	Output clear signals when ZRN is completed

2. Special D registers:

D1312:	Specify the number of additional pulses for additional pulses output and Z-phase seeking function of ZRN instruction (Has to be used with M1308)

API	Mnemonic		Operands	Function	Controllers
157	D	PLSV	S D1 D2	Adjustable Speed Pulse Output	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word Devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	PLSV: 7 steps
S					*	*	*	*	*	*	*	*	*	*	*	DPLSV: 13 steps
D_1		*														
D_2		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

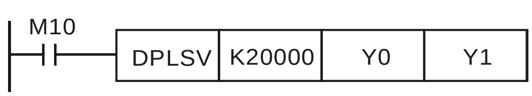
S: Pulse output frequency **D_1**: Pulse output device (Y0, Y2) **D_2**: Direction signal output

Explanations:

1. The instruction only supports the pulse output type: Pulse + Direction.
2. **S** is the designated pulse output frequency. Available range: -100,000Hz ~ +100,000 Hz. "+/-" signs indicate forward/reverse output direction. The frequency can be changed during pulse output. However, if the specified output direction is diferent from the current output direction, the instruction will stop for 1 scan cycle then restart with the changed frequency.
3. **D_1** is the pulse output device. It can designate CH0(Y0) and CH1(Y2).
4. **D_2** is the direction signal output device. It can designate CH0(Y1) and CH1(Y3).
5. The operation of **D_2** corresponds to the "+" or "-" of **S**. When **S** is "+", **D_2** will be OFF; when **S** is "-", **D_2** will be ON.
6. M1305 and M1306 can change the output direction of CH0/CH1 set in **D_2**. When **S** is "-", **D_2** will be ON, however, if M1305/M1306 is set ON before instruction executes, **D_2** will be OFF during execution of instruction.
7. PLSV instruction does not support settings for ramp up or ramp down. If ramp up/down process is required, please use API 67 RAMP instruction.
8. If the drive contact turns off during pulse output process, pulse output will stop immediately.

Program Example:

When M10 = ON, Y0 will output pulses at 20kHz. Y1 = OFF indicates forward direction.



API	Mnemonic		Operands	Function	Controllers			
					ES2/EX2	SS2	SA2	SX2
158	D	DRVI	S_1 S_2 D_1 D_2	Relative Position Control				

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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Number of pulses (relative positioning) **S_2**: Pulse output frequency **D_1**: Pulse output device **D_2**: Direction signal output

Explanations:

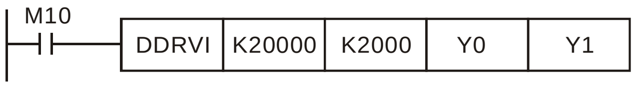
1. The instruction only supports the pulse output type: Pulse + Direction.
2. **S_1** is the number of pulses (relative positioning). Available range: -2,147,483,648 ~ +2,147,483,647. "+/-" signs indicate forward and reverse direction.
3. **S_2** is the pulse output frequency. Available range: 6 ~ 100,000Hz.
4. **D_1** is the pulse output device. It can designate CH0 (Y0) and CH1 (Y2).
5. **D_2** is the direction signal output device. It can designate CH0 (Y1) and CH1 (Y3).
6. The operation of **D_2** corresponds to the "+" or "-" of **S**. When **S** is "+", **D_2** will be OFF; when **S** is "-", **D_2** will be ON. **D_2** will not be OFF immediately after pulse output completion and will be OFF when the drive contact is OFF.
7. The set value in **S_1** is the relative position of
   - current position (32-bit data) of CH0 (Y0, Y1) which is stored in D1031(high), D1030 (low)
   - current position (32-bit data) of CH1 (Y2, Y3) which is stored in D1337(high), D1336 (low).

   In reverse direction pulse output, value in (D1031, D1330) and (D1336, D1337) decreases.
8. D1343 (D1353) is the ramp up/down time setting of CH0 (CH1). Available range: 20 ~ 32,767ms. Default: 100ms. PLC will take the upper/lower bound value as the set value when specified value exceeds the available range.
9. D1340 (D1352) is start/end frequency setting of CH0 (CH1). Available range: 6 to 100,000Hz. PLC will take the upper/lower bound value as the set value when specified value exceeds the available range.
10. M1305 and M1306 can change the output direction of CH0/CH1 set in **D_2**. When **S** is "-", **D_2** will be ON, however, if M1305/M1306 is set ON before instruction executes, **D_2** will be OFF during execution of instruction..
11. Ramp-down time of CH0 and CH1 can be particularly modified by using (M1534, D1348) and (M1535, D1349). When M1534 / M1535 = ON, CH0 / CH1 ramp-down time is specified by D1348 / D1349.

12. If M1078 / M1104 = ON during instruction execution, Y0 / Y2 will pause immediately and M1538 / M1540 = ON indicates the pause status. When M1078 / M1104 = OFF, M1538 / M1540 = OFF, Y0 / Y2 will proceed to finish the remaining pulses.
13. DRVI instruction supports Alignment Mark and Mask function. Please refer to the explanation in API 59 PLSR instruction.

Program Example:

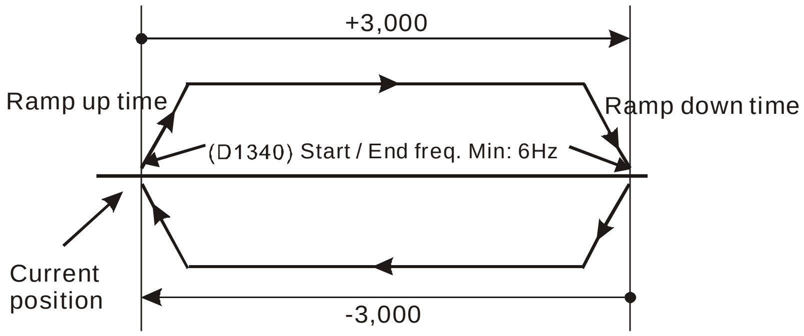
When M10= ON, 20,000 pulses (relative position) at 2kHz frequency will be generated from Y0. Y1= OFF indicates positive direction.



Points to note:

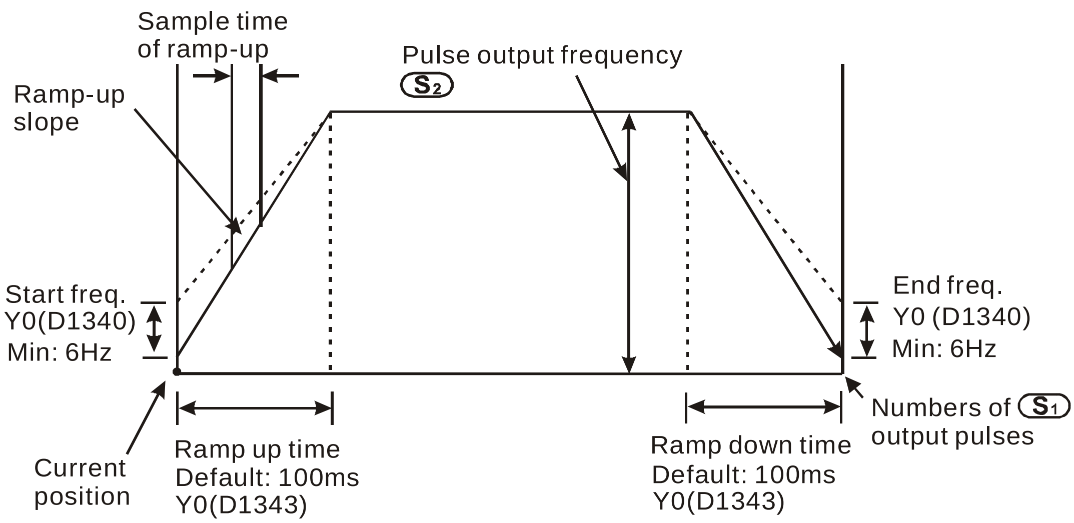
1. Operation of relative positioning:

   Pulse output executes according to the relative distance and direction from the current position



2. Registers for setting ramp up/down time and start/end frequency:

   - Output Y0:



- This instruction can be used many times in user program, but only one instruction will be activated at a time. For example, if Y0 is currently activated, other instructions use Y0 won't be executed. Therefore, instructions first activated will be first executed.
- After activating the instruction, all parameters cannot be modified unless instruction is OFF.

3. Associated Flags:

M1029 CH0 (Y0, Y1) pulse output execution completed.

M1102 CH1 (Y2, Y3) pulse output execution completed

M1078 CH0 (Y0, Y1) pulse output pause (immediate)

M1104 CH1 (Y2, Y3) pulse output pause (immediate)

M1108 CH0 (Y0, Y1) pulse output pause (ramp down).

M1110 CH1 (Y2, Y3) pulse output pause (ramp down)

M1156 Enabling the mask and alignment mark function on I400/I401(X4) corresponding to Y0.

M1158 Enabling the mask and alignment mark function on I600/I601(X6) corresponding to Y2.

M1305 Reverse Y1 pulse output direction in high speed pulse output instructions

M1306 Reverse Y3 pulse output direction in high speed pulse output instructions

M1347 Auto-reset Y0 when high speed pulse output completed

M1524 Auto-reset Y2 when high speed pulse output completed

M1534 Enable ramp-down time setting on Y0. Has to be used with D1348

M1535 Enable ramp-down time setting on Y2. Has to be used with D1349.

M1538 Indicating pause status of CH0 (Y0, Y1)

M1540 Indicating pause status of CH1 (Y2, Y3)

4. Special D registers:

D1030 Low word of the present value of Y0 pulse output

D1031 High word of the present value of Y0 pulse output

D1336 Low word of the present value of Y2 pulse output

D1337 High word of the present value of Y2 pulse output

D1340 Start/end frequency of the 1st group pulse output CH0 (Y0, Y1)

D1352 Start/end frequency of the 2nd group pulse output CH1 (Y2, Y3)

D1343 Ramp up/down time of the 1st group pulse output CH0 (Y0, Y1)

D1353 Ramp up/down time of the 2nd group pulse output CH1 (Y2, Y3)

D1348: CH0(Y0, Y1) pulse output. When M1534 = ON, D1348 stores the ramp-down time

D1349: CH1(Y2, Y3) pulse output. When M1535 = ON, D1349 stores the ramp-down time

D1232 Output pulse number for ramp-down stop when Y0 masking sensor receives signals. (LOW WORD)

D1233 Output pulse number for ramp-down stop when Y0 masking sensor receives signals. (HIGH WORD).

D1234 Output pulse number for ramp-down stop when Y2 masking sensor receives signals (LOW WORD).

D1235 Output pulse number for ramp-down stop when Y2 masking sensor receives signals (HIGH WORD).

D1026 Pulse number for masking Y0 when M1156 = ON (Low word)

D1027 Pulse number for masking Y0 when M1156 = ON (High word)

D1135 Pulse number for masking Y2 when M1158 = ON (Low word)

D1136 Pulse number for masking Y2 when M1158 = ON (High word)

API	Mnemonic		Operands	Function	Controllers
159	D	DRVA	(S₁) (S₂) (D₁) (D₂)	Absolute Position Control	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices X	Y	M	S	Word devices K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	DRVA: 9 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DDRVA: 17 steps
D_1		*														
D_2		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Numbers of pulses (Absolute positioning) **S_2**: Pulse output frequency **D_1**: Pulse output device **D_2**: Direction signal output

Explanations:

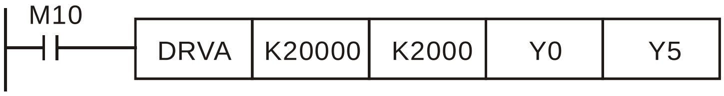
1. The instruction only supports the pulse output type: Pulse + Direction.
2. **S_1** is the number of pulses (Absolute positioning). Available range: -2,147,483,648 ~ +2,147,483,647. "+/-" signs indicate forward and reverse direction.
3. **S_2** is the pulse output frequency. Available range: 6 ~ 100,000Hz.
4. **D_1** is the pulse output device. It can designate CH0 (Y0) and CH1 (Y2).
5. **D_2** is the direction signal output device. If Y output is designated, only CH0 (Y1) and CH1 (Y3) are available.
6. **S_1** is the target position for absolute positioning. The actual number of output pulses (**S_1** – current position) will be calculated by PLC. When the result is positive, pulse output executes forward operation, i.e. **D_2** = OFF; when the results is negative, pulse output executes reverse operation, i.e. **D_2** = ON.
7. The set value in **S_1** is the absolute position from zero point. The calculated actual number of output pulses will be the relative position of
   - current position (32-bit data) of CH0 (Y0, Y1) which is stored in D1031(high), D1030 (low)
   - current position (32-bit data) of CH1 (Y2, Y3) which is stored in D1337(high), D1336 (low).

   In reverse direction pulse output, value in (D1031, D1330) and (D1336, D1337) decreases.
8. D1343 (D1353) is the ramp up/down time (between start frequency and pulse output frequency) setting of CH0 (CH1). Available range: 20 ~ 32,767ms. Default: 100ms. PLC will take 20ms as the set value when specified value is below 20ms or above 32,767ms.
9. D1340 (D1352) is start/end frequency setting of CH0 (CH1). Available range: 6 ~ 32,767Hz. PLC will take the start/end frequency as the pulse output frequency when pulse output frequency **S_2** is smaller or equals the start/end frequency.
10. M1305 and M1306 can change the output direction of CH0/CH1 set in **D_2**. When **S** is "-", **D_2** will be ON, however, if M1305/M1306 is set ON before instruction executes, **D_2** will be OFF during execution of instruction..

11. Ramp-down time of CH0 and CH1 can be particularly modified by using (M1534, D1348) and (M1535, D1349). When M1534 / M1535 = ON, CH0 / CH1 ramp-down time is specified by D1348 / D1349.
12. If M1078 / M1104 = ON during instruction execution, Y0 / Y2 will pause immediately and M1538 / M1540 = ON indicates the pause status. When M1078 / M1104 = OFF, M1538 / M1540 = OFF, Y0 / Y2 will proceed to finish the remaining pulses.
13. DRVA/DDRVA instructions do NOT support Alignment Mark and Mask function.

Program Example:

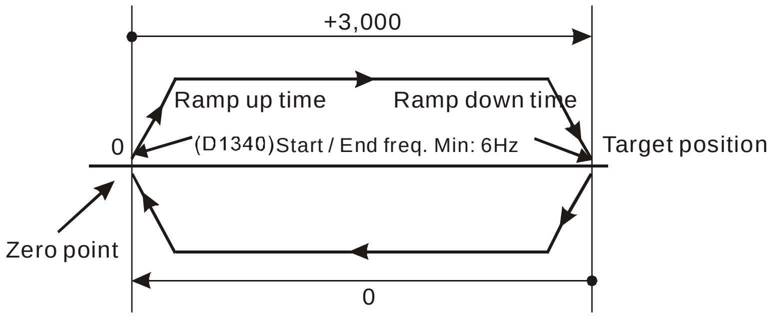
When M10 = ON, DRVA instruction executes absolute positioning on Y0 at target position 20000, target frequency 2kHz. Y5 = OFF indicates positive direction.



Points to note:

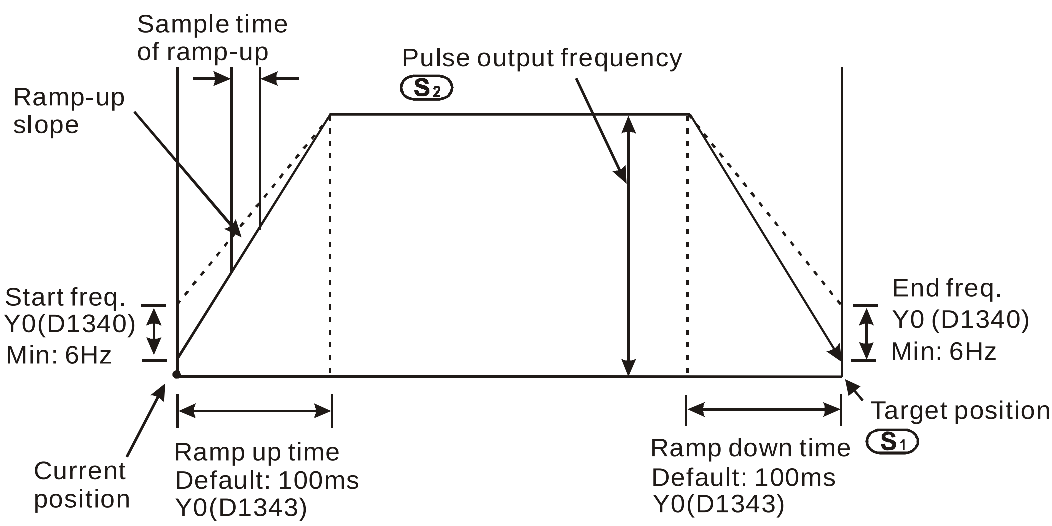
1. Operation of absolute positioning:

   Pulse output executes according to the specified absolute position from zero point



2. Registers for setting ramp up/down time and start/end frequency:

- Output Y0:



- This instruction can be used many times in user program, but only one instruction will be activated at a time. For example, if Y0 is currently activated, other instructions use Y0 won't be executed. Therefore, instructions first activated will be first executed.
- After activating the instruction, all parameters cannot be modified unless instruction is OFF.
- For associated special flags and special registers, please refer to **Points to note** of DDRVI instruction.

API	Mnemonic		Operands	Function	Controllers
160	TCMP	P	S1 S2 S3 S D	Time compare	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				**Word devices**											TCMP, TCMPP: 11 steps
S_1					*	*	*	*	*	*	*	*	*	*	*	
S_2					*	*	*	*	*	*	*	*	*	*	*	
S_3					*	*	*	*	*	*	*	*	*	*	*	
S											*	*	*			
D		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

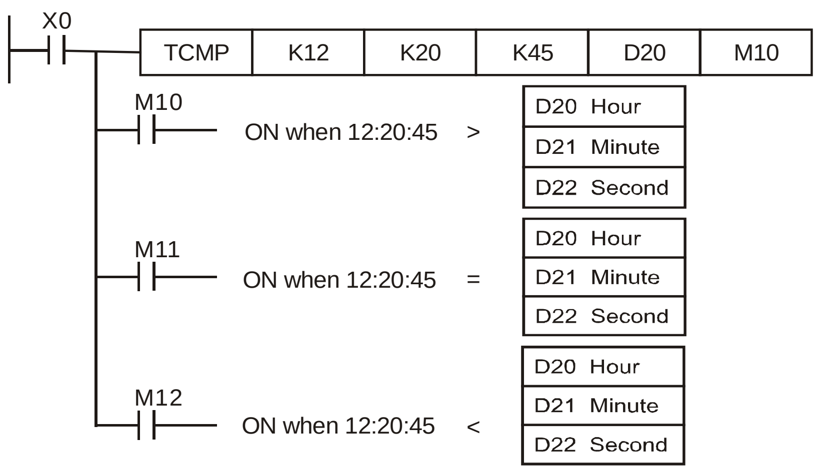
S_1: "Hour" for comparison (K0~K23) **S_2**: "Minute" for comparison (K0~K59) **S_3**: "Second" for comparison (K0~K59) **S**: Current time of RTC (occupies 3 consecutive devices) **D**: Comparison result (occupies 3 consecutive devices)

Explanations:

1. TCMP instruction compares the time set in **S_1**, **S_2**, **S_3** with RTC current value in **S** and stores the comparison result in **D**.
2. **S:** "Hour" of current time of RTC. Content: K0~K23. **S** +1: "Minute" of current time of RTC. Content: K0~K59. **S** +2**:** "Second" of current time of RTC. Content: K0~K59.
3. Usually the time of RTC in **S** is read by TRD instruction first then compared by TCMP instruction. If operand **S** exceeds the available range, operation error occurs and M1067 = ON, M1068 = ON. D1067 stores the error code 0E1A (HEX).

Program Example:

1. When X0 = ON, the instruction executes and the RTC current time in D20~D22 is compared with the set value 12:20:45. Comparison result is indicated by M10~M12. When X0 goes from ON→OFF, the instruction is disabled however the ON/OFF status of M10~M12 remains.
2. Connect M10 ~ M12 in series or in parallel to obtain the results of ≧, ≦, and ≠.



API	Mnemonic		Operands	Function	Controllers
161	TZCP	P	(S1) (S2) (S) (D)	Time zone compare	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1											*	*	*			TZCP, TZCPP: 9 steps
S_2											*	*	*			
S											*	*	*			
D		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

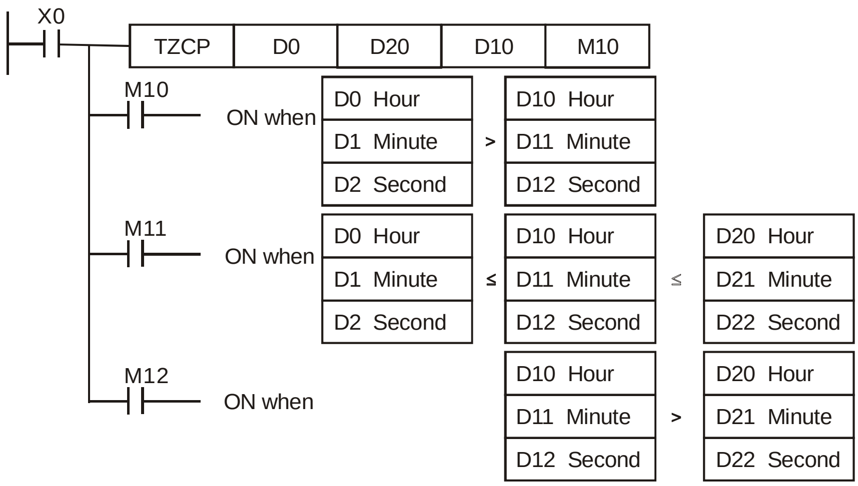
S_1: Lower bound of the time for comparison (occupies 3 consecutive devices) **S_2**: Upper bound of the time for comparison (occupies 3 consecutive devices) **S**: Current time of RTC (occupies 3 consecutive devices) **D**: Comparison result (occupies 3 consecutive devices)

Explanations:

1. TZCP instruction compares current RTC time in **S** with the range set in **S_1**~ **S_2** and the comparison result is stored in **D**.
2. **S_1**, **S_1** +1, **S_1** +2: The “hour”, “minute” and “second” of the lower bound value for comparison.
3. **S_2**, **S_2** +1, **S_2** +2: The “hour”, “minute” and “second” of the upper bound value for comparison.
4. **S**, **S** +1, **S** +2: The “hour”, “minute” and “second” of the current time of RTC.
5. Usually the time of RTC in **S** is read by TRD instruction first then compared by TZMP instruction. If operand **S**, **S_1**, **S_2** exceed the available range, operation error occurs and M1067 = ON, M1068 = ON. D1067 stores the error code 0E1A (HEX).
6. If **S** $<$ **S_1** and **S** $<$ **S_2**, **D** is ON. When **S** $>$ **S_1** and **S** $>$ **S_2**, **D**+2 is ON. For other conditions, **D** + 1 will be ON. (Lower bound **S_1** should be less than upper bound **S_2**.)

Program Example:

When X0 = ON, TZCP instruction executes and M10~M12 will be ON to indicate the comparison results. When X0 = OFF, the instruction is disabled but the ON/OFF status of M10~M12 remains.



API	Mnemonic		Operands	Function	Controllers
162	TADD	P	S1 S2 D	Time addition	ES2/EX2 SS2 SA2 SX2

Type / OP	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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

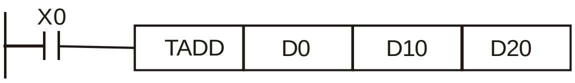
S_1: Time augend (occupies 3 consecutive devices) **S_2**: Time addend (occupies 3 consecutive devices) **D**: Addition result (occupies 3 consecutive devices)

Explanations:

1. TADD instruction adds the time value (Hour, Minute Second) **S_1** with the time value (Hour, Minute Second) **S_2** and stores the result in **D**.
2. If operand **S_1**, **S_2** exceed the available range, operation error occurs and M1067 = ON, M1068 = ON. D1067 stores the error code 0E1A (HEX).
3. If the addition result is larger than 24 hours, the carry flag M1022 will be ON and the value in **D** will be the result of “sum minuses 24 hours”.
4. If the sum equals 0 (00:00:00), Zero flag M1020 will be ON.

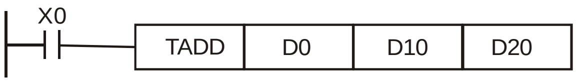
Program Example:

When X0 = ON, TADD instruction executes and the time value in D0~D2 is added with the time value in D10~D12. The addition result is stored in D20~D22.



D0 08(Hour)		D10 06(Hour)		D20 14(Hour)
D1 10(Min)	+	D11 40(Min)	→	D21 50(Min)
D2 20(Sec)		D12 06(Sec)		D22 26(Sec)
08:10:20		06:40:06		14:50:26

If the addition result is greater than 24 hours, the Carry flag M1022 = ON.



D0 18(Hour)		D10 11(Hour)		D20 06(Hour)
D1 40(Min)	+	D11 30(Min)	→	D21 10(Min)
D2 30(Sec)		D12 08(Sec)		D22 38(Sec)
18:40:30		11:30:08		06:10:38

API	Mnemonic		Operands	Function	Controllers
163	TSUB	P	S_1 S_2 D	Time subtraction	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices X	Y	M	S	Word devices K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1											*	*	*			TSUB, TSUBP: 7 steps
S_2											*	*	*			
D											*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

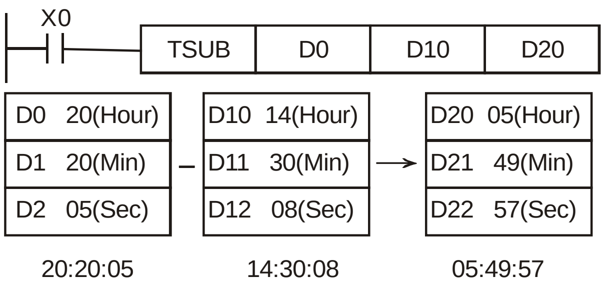
S_1: Time minuend (occupies 3 consecutive devices) **S_2**: Time subtrahend (occupies 3 consecutive devices) **D**: Subtraction result (occupies 3 consecutive devices)

Explanations:

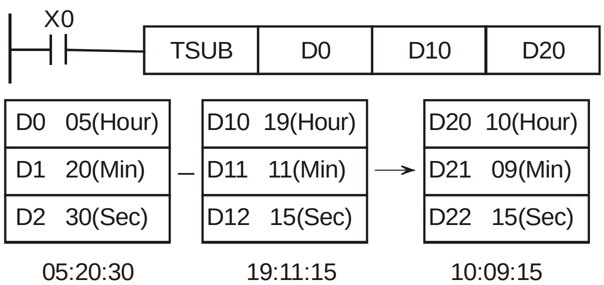
1. TSUB instruction subtracts the time value (Hour, Minute Second) **S_1** with the time value (Hour, Minute Second) **S_2** and stores the result in **D**.
2. If operand **S_1**, **S_2** exceed the available range, operation error occurs and M1067 = ON, M1068 = ON. D1067 stores the error code 0E1A (HEX).
3. If the subtraction result is a negative value (less than 0), Borrow flag M1020 = ON and the value in **D** will be the result of "the negative value pluses 24 hours".
4. If the subtraction result (remainder) equals 0 (00:00:00), Zero flag M1020 will be ON.
5. Besides using TRD instruction, MOV instruction can also be used to move the RTC value to D1315 (Hour), D1314 (Minute), D1313 (Second) for reading the current time of RTC..

Program Example:

When X0 = ON, TSUB instruction executes and the time value in D0~D2 is subtracted by the time value in D10~D12. The subtraction result is stored in D20~D22.



If the subtraction result is a negative value (less than 0), Borrow flag M1021 = ON.



API	Mnemonic		Operands	Function	Controllers			
166	TRD	P	D	Time read	ES2/EX2	SS2	SA2	SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	TRD, TRDP: 3 steps
D											*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operand:

D: Current time of RTC (occupies 7 consecutive devices)

Explanations:

1. TRD instruction reads the 7 real-time data of RTC (year (A.D.), day(Mon.Sun.), month, day, hour, minute, second from D1319~D1313 and stores the read data in registers specified by **D.**
2. Only when power is on can RTCs of SS2 series perform the fuction of timing. The RTC data registers D1319~D1313 are latched. When power is resumed, the RTC will resume the stored time value before power down. Therefore, we suggest users modify the RTC value every time when power is ON.
3. RTCs of SA2 V1.0 及 ES2/EX2/SX2 V2.0 series can still operate for one or two weeks after the power is off (they vary with the ambient temperature). Therefore, if the machine has not operated since one or two weeks ago, please reset RTC.
4. D1319 only stores the 2-digit year in A.D. If 4-digit year data is required, please refer to **Points to note** below.
5. For relative flags and registers please refer to **Points to note**.

Program Example:

When X0 = ON, TRD instruction reads the current time of RTC to the specified register D0~D6.

The content of D1318: 1 = Monday; 2 = Tuesday … 7 = Sunday.

Special D	Item	Content		Normal D	Item
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	Day	1~31	→	D3	Day
D1315	Hour	0~23	→	D4	Hour
D1314	Minute	0~59	→	D5	Minute
D1313	Second	0~59	→	D6	Second

Points to note:

1. There are two methods to correct built-in RTC:
   - Correcting by API167 TWR instruction
     Please refer to explanation of instruction TWR (API 167)
   - Setting by peripheral device
     Using WPLSoft / ISPSoft (Ladder editor)
2. Display 4-digit year data:
   - D1319 only stores the 2-digit year in A.D. If 4-digit year data is required, please insert the following instruction at the start of program.

   - The original 2-digit year will be switched to a 4-digit year, i.e. the 2-digit year will pluses 2,000. If users need to write in new time in 4-digit year display mode, only a 2-digit year data is applicable (0 ~ 99, indicating year 2000 ~ 2099). For example, 00 = year 2000, 50 = year 2050 and 99 = year 2099.
   - Flags and special registers for RTC

Device	Content	Function
M1016	Year display mode of RTC	OFF: D1319 stores 2-digit year data in A.D. ON: D1319 stores 2-digit year data in A.D + 2000
M1017	±30 seconds correction on RTC	Correction takes place when M1017 goes from OFF to ON (Second data in 0 ~ 29: reset to 0. Second data in 30 ~ 59: minute data pluses 1, second data resets)

Device	Content	Range
D1313	Second	0-59
D1314	Minute	0-59
D1315	Hour	0-23
D1316	Day	1-31
D1317	Month	1-12
D1318	Day (Mon. ~ Sun.)	1-7
D1319	Year	0-99 (two digit year data)

API	Mnemonic			Operands	Function	Controllers			
167		TWR	P	(S)	Time write	ES2/EX2	SS2	SA2	SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operand:

S: Set value for RTC (occupies 7 consecutive devices)

Explanations:

1. TWR instruction updates the RTC with the value set in **S**.
2. If the time data in **S** exceeds the valid calendar range, it will result in an "operation error". PLC will writes in the smallest valid value automatically, M1067 = ON, M1068 = ON, and error code 0E1A (HEX) is recorded in D1067
3. For explanations of associated flags and the characteristics of RTCS, please refer to **Points to note** of TRD instruction.

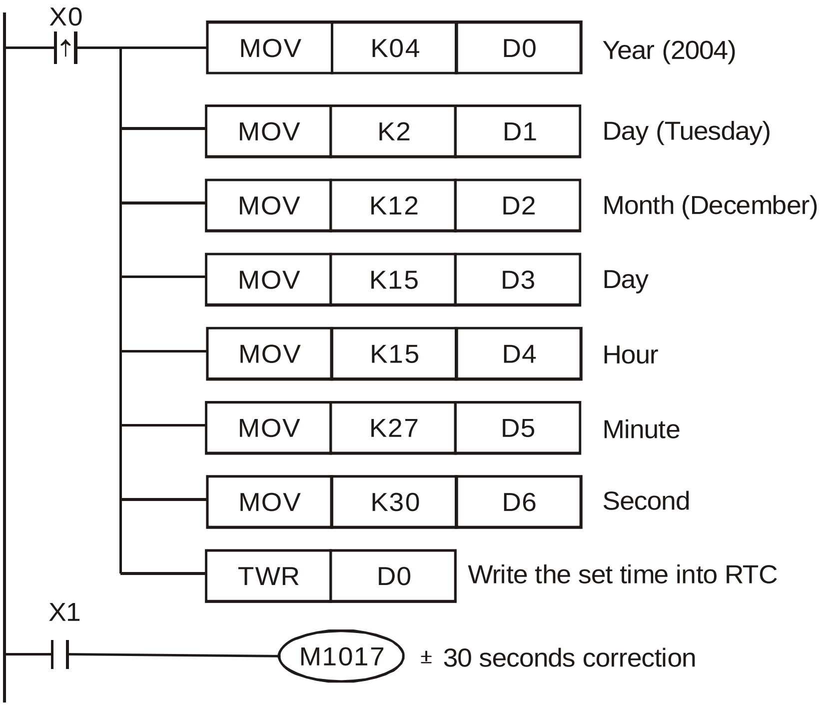
Program Example 1:

When X0 = ON, write the new time into RTC.

	Normal D	Item	Range		Special D	Item	
Set value	D20	Year (A.D.)	00~99	→	D1319	Year (A.D.)	RTC
	D21	Day (Mon.~Sun.)	1~7	→	D1318	Day (Mon.~Sun.)	
	D22	Month	1~12	→	D1317	Month	
	D23	Day	1~31	→	D1316	Day	
	D24	Hour	0~23	→	D1315	Hour	
	D25	Minute	0~59	→	D1314	Minute	
	D26	Second	0~59	→	D1313	Second	

Program Example 2:

1. Set the current time in RTC as 2004/12/15, Tuesday, 15:27:30.
2. The content of D0~D6 is the set value for adjusting RTC.
3. When X0 = ON, update the time of RTC with the set value.
4. When X1 = ON, perform ±30 seconds correction. Correction takes place when M1017 goes from OFF to ON (Second data in 0 ~ 29: reset to 0. Second data in 30 ~ 59: minute data pluses 1, second data resets).


X0
MOV K04 D0
Year (2004)
MOV K2 D1
Day (Tuesday)
MOV K12 D2
Month (December)
MOV K15 D3
Day
MOV K15 D4
Hour
MOV K27 D5
Minute
MOV K30 D6
Second
TWR D0
Write the set time into RTC
X1
M1017
± 30 seconds correction

API	Mnemonic			Operands	Function	Controllers
168	D	MVM	P	S1 S2 D	Transfer Designated Bits	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	MVM, MVMP: 7 steps
S_1							*	*	*	*	*	*	*	*	*	DMVM,DMVMP:
S_2					*	*	*	*	*	*	*	*	*	*	*	
D							*	*	*	*	*	*	*	*	*	13 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

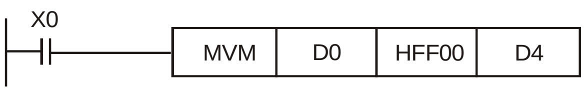
S_1: Source device 1 **S_2**: Bits to be masked (OFF) **D**: **D** =(**S_1** & **S_2**) | (**D** & ~ **S_2**)

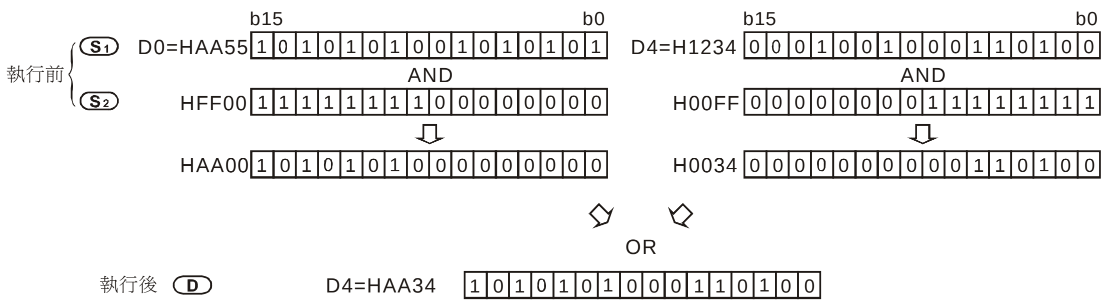
Explanations:

1. The instruction conducts logical AND operation between **S_1** and **S_2** first, logical AND operation between **D** and ~**S_2** secondly, and combines the 1st and 2nd results in **D** by logical OR operation.
2. Rule of Logical AND operation: 0 AND 1 = 0, 1 AND 0 = 0, 0 AND 0 = 0, 1 AND 1 = 1
3. Rule of Logical OR operation: 0 OR 1= 1, 1 OR 0 = 1, 0 OR 0 = 0, 1 OR 1 = 1.

Program Example 1 :

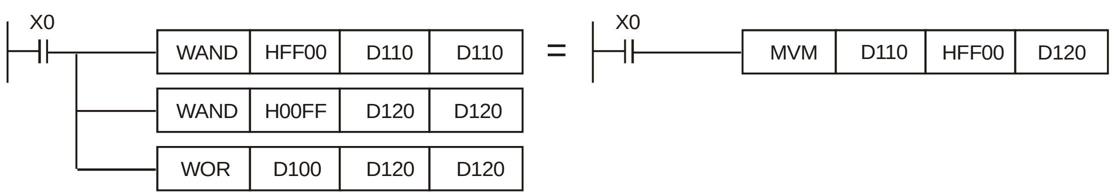
When X0 = ON, MVM instruction conducts logical AND operation between 16-bit register D0 and H'FF00 first, logical AND operation between D4 and H'00FF secondly, and combines the 1st and 2nd results in D4 by logical OR operation.





Program Example 2 :

Simplify instructions:



API	Mnemonic		Operands	Function	Controllers			
					ES2/EX2	SS2	SA2	SX2
169	D	HOUR	S D1 D2	Hour meter				

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*	*	*	*	*	*	*	*	*	*	HOUR: 7 steps
D_1													*			DHOUR: 13 steps
D_2		*	*	*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Set-point value for driving the output device (Unit: hour) **D_1**: Current time being measured

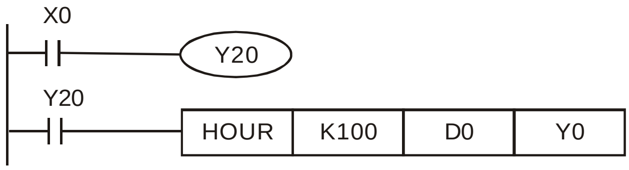
D_2: Output device

Explanations:

1. HOUR instruction drives the output device **D_2** when the measured current time **D_1** reaches the set-point value in **S**.
2. Range of **S**: K1~K32,767; unit: hour. Range of **D_1** in 16-bit instruction: K0~K32,767. Range of **D_1** +1 (current time less than an hour): K0 ~K3,599; unit: second.
3. When the ON-time of the drive contact reaches the set-point value, output device will be ON. The instruction can be applied for controlling the working hours of machine or conducting preventive maintenance.
4. After output device is ON, the current time will still be measured in **D_1**.
5. In 16-bit instruction, when the current time measured reaches the maximum 32,767 hours / 3,599 seconds, the timing will stop. To restart the timing, **D_1** and **D_1** + 1 have to be reset.
6. In 32-bit instruction, when the current time measured reaches the maximum 2,147,483,647 hours / 3,599 seconds, the timing will stop. To restart the timing, **D_1** ~ **D_1** + 2 have to be reset.
7. If operand **S** uses device F, only 16-bit instruction is available.
8. HOUR instruction can be used for four times in the program.

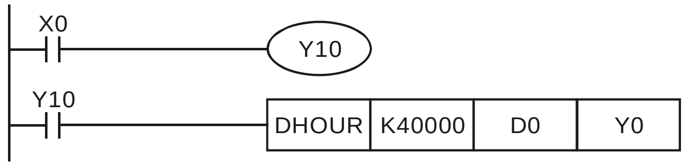
Program Example 1:

In 16-bit instruction, when X0 = ON, Y20 will be ON and the timing will start. When the timing reaches 100 hours, Y0 will be ON and D0 will record the current time measured (in hour). D1 will record the current time less than an hour (0 ~ 3,599; unit: second)..



Program Example 2:

In 32-bit instruction, when X0 = ON, Y10 will be ON and the timing will start. When the timing reaches 40,000 hours, Y0 will be ON. D1 and D0 will record the current time measured (in hour) and D2 will record the current time less than an hour (0 ~ 3,599; unit: second).



API	Mnemonic			Operands	Function	Controllers
170	D	GRY	P	(S) (D)	BIN → Gray Code	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*	*	*	*	*	*	*	*	*	*	GRY, GRYP: 5 steps
D								*	*	*	*	*	*	*	*	DGRY, DGRYP: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

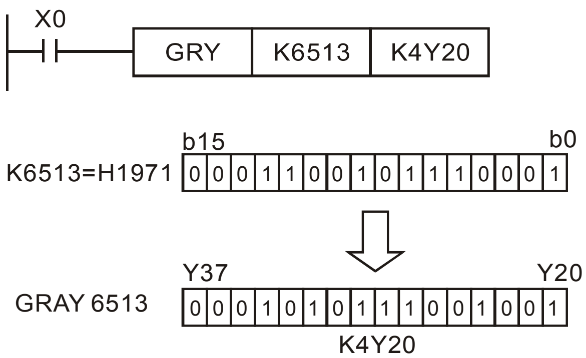
S: Source device **D**: Operation result (Gray code)

Explanations:

1. GRY instruction converts the BIN value in **S** to Gray Code and stores the converted result in specified register **D**.
2. Available range of **S**:
   16-bit instruction: 0~32,767
   32-bit instruction: 0~2,147,483,647
3. If operand **S** exceeds the available range, operation error occurs and M1067 = ON, M1068 = ON. D1067 stores the error code 0E1A (HEX)
4. If operands **S** and **D** use device F, only 16-bit instruction is applicable.

Program Example:

When X0 = ON, GRY instruction executes and converts K6513 to Gray Code. The operation result is stored in K4Y20, i.e. Y20 ~ Y37.



API	Mnemonic			Operands	Function	Controllers
171	D	GBIN	P	S D	Gray Code → BIN	ES2/EX2 SS2 SA2 SX2

OP \ Type	Bit Devices X	Y	M	S	Word devices K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S					*	*	*	*	*	*	*	*	*	*	*	GBIN, GBINP: 5 steps DGBIN, DGBINP: 9 steps
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

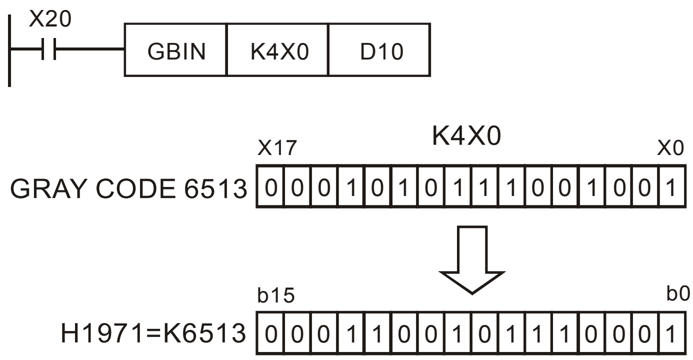
S: Source device **D**: Operation result (BIN value)

Explanations:

1. GBIN instruction converts the Gray Code in **S** to BIN value and stores the converted result in specified register **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 PLC inputs. The Gray code is converted to BIN value and stored in the specified register.
3. Available range of **S**:
   16-bit instruction ：0~32,767
   32-bit instruction ：0~2,147,483,647
4. If operand **S** exceeds the available range, operation error occurs and the instruction is disabled.
5. If operands **S** and **D** use device F, only 16-bit instruction is applicable.

Program Example:

When X20 = ON, the Gray Code value in the absolute position type encoder connected to X0~X17 inputs is converted to BIN value and stored in D10.



API	Mnemonic			Operands	Function	Controllers
172	D	ADDR	P	(S1) (S2) (D)	Floating point addition	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DADDR, DADDRP: 13 steps
S_1													*			
S_2													*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Floating point summand **S_2:** Floating point addend **D:** Sum

Explanations:

1. ADDR instruction adds the floating point summand **S_1** with floating point addend **S_2** and stores the operation result in **D**.
2. In ADDR instruction, floating point values can be directly entered into **S_1** and **S_2**.
3. In DADDR instruction, floating point values (e.g. F1.2) can be either entered directly into **S_1** and **S_2** or stored in data registers for operation.
4. When **S_1** and **S_2** is specified as data registers, the function of DADDR instruction is the same as API 120 EADD instruction.
5. **S_1** and **S_2** can designate the same register. In this case, if the instruction is specified as "continuous execution instruction" (generally DADDRP instruction) and the drive contact is ON, the register will be added once in every scan.
6. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag)
   If absolute value of the result exceeds max floating point value, carry flag M1022 = ON.
   If absolute value of the result is less than min. floating point value, borrow flag M1021 = ON.
   If the conversion result is 0, zero flag M1020 = ON

Program Example 1:

When X0 = ON, add floating point number F1.200E+0 (Input F1.2, and scientific notation F1.200E+0 will be displayed on ladder diagram. Users can set monitoring data format as float on the function View) with F2.200E+0 and store the obtained result F3.400E+0 in register D10 and D11.

X0	DADDR	F1.200E+0	F2.200E+0	D10

Program example 2:

When X0 = ON, add floating point value (D1, D0) with (D3, D2) and store the result in (D11, D10).

X0	DADDR	D0	D2	D10

API	Mnemonic			Operands	Function	Controllers
173	D	SUBR	P	S1 S2 D	Floating point subtraction	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	DSUBR: 13 steps
S_1													*			
S_2													*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

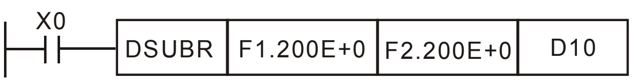
S_1: Floating point minuend **S_2:** Floating point subtrahend **D:** Remainder

Explanations:

1. SUBR instruction subtracts **S_1** with **S_2** and stores the operation result in **D**.
2. In SUBR instruction, floating point values can be directly entered into **S_1** and **S_2**..
3. In DSUBR instruction, floating point values (e.g. F1.2) can be either entered directly into **S_1** and **S_2** or stored in data registers for operation.
4. When **S_1** and **S_2** is specified as data registers, the function of DSUBR instruction is the same as API 121 ESUB instruction.
5. **S_1** and **S_2** can designate the same register. In this case, if the instruction is specified as "continuous execution instruction" (generally DSUBRP instruction) and the drive contact is ON, the register will be subtracted once in every scan.
6. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag)
   If absolute value of the result exceeds max floating point value, carry flag M1022 = ON.
   If absolute value of the result is less than min. floating point value, borrow flag M1021 = ON.
   If the conversion result is 0, zero flag M1020 = ON

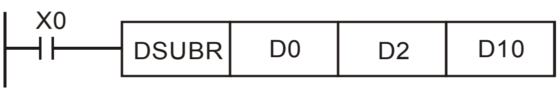
Program example 1:

When X0 = ON, subtract floating point number F1.200E+0 (Input F1.2, and scientific notation F1.200E+0 will be displayed on ladder diagram. Users can set monitoring data format as float on the function View) with F2.200E+0 and store the obtained result F-1.000E+0 in register D10 and D11.



Program example 2:

When X0 = ON, subtract the floating point value (D1, D0) with (D3, D2) and store the result in (D11, D10).



API	Mnemonic			Operands	Function	Controllers
174	D	MULR	P	S1 S2 D	Floating point multiplication	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
S_1													*			DMULR, DMULRP: 13 steps
S_2													*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

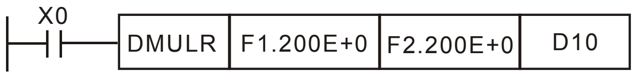
S_1: Floating point multiplicand **S_2:** Floating point multiplicator **D:** Product

Explanations:

1. MULR instruction multiplies **S_1** with **S_2** and stores the operation result in **D**.
2. In MULR instruction, floating point values can be directly entered into **S_1** and **S_2**.
3. In DMULR instruction, floating point values (e.g. F1.2) can be either entered directly into **S_1** and **S_2** or stored in data registers for operation.
4. When **S_1** and **S_2** is specified as data registers, the function of DMULR instruction is the same as API 122 EMUL instruction.
5. **S_1** and **S_2** can designate the same register. In this case, if the instruction is specified as "continuous execution instruction" (generally DMULRP instruction) and the drive contact is ON, the register will be multiplied once in every scan
6. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag)
   If absolute value of the result exceeds max floating point value, carry flag M1022 = ON.
   If absolute value of the result is less than min. floating point value, borrow flag M1021 = ON.
   If the conversion result is 0, zero flag M1020 = ON.

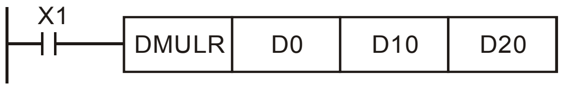
Program Example 1:

When X0= ON, multiply floating point number F1.200E+0 (Input F1.2, and scientific notation F1.200E+0 will be displayed on ladder diagram. Users can set monitoring data format as float on the function View) with F2.200E+0 and store the obtained result F2.640E+0 in register D10 and D11.



Program example 2:

When X1= ON, multiply the floating point value (D1, D0) with (D11, D10) and store the result in (D21, D20).



API	Mnemonic			Operands	Function	Controllers
175	D	DIVR	P	S_1 S_2 D	Floating point division	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1													*			DDIVR: 13 steps
S_2													*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

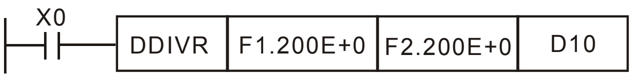
S_1: Floating point n dividend **S_2:** Floating point divisor **D:** Quotient

Explanations:

1. DIVR instruction divides **S_1** by **S_2** and stores the operation result in **D**
2. In DIVR instruction, floating point values can be directly entered into **S_1** and **S_2**.
3. In DDIVR instruction, floating point values (e.g. F1.2) can be either entered directly into **S_1** and **S_2** or stored in data registers for operation.
4. When **S_1** and **S_2** is specified as data registers, the function of DDIVR instruction is the same as API 123 EDIV instruction.
5. If **S_2** = 0, operation error occurs and M1067 = ON, M1068 = ON. D1067 stores the error code 0E19 (HEX).
6. Flags: M1020 (Zero flag), M1021 (Borrow flag) and M1022 (Carry flag)
   If absolute value of the result exceeds max floating point value, carry flag M1022 = ON.
   If absolute value of the result is less than min. floating point value, borrow flag M1021 = ON.
   If the conversion result is 0, zero flag M1020 = ON.

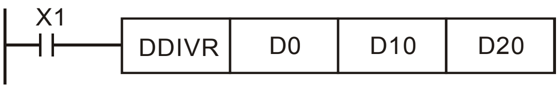
Program example 1:

When X0 = ON, divide floating point number F1.200E+0 (Input F1.2, and scientific notation F1.200E+0 will be displayed on ladder diagram. Users can set monitoring data format as float on the function View) with F2.200E+0 and store the obtained result F0.545E+0 in D10 and D11.



Program example 2:

When X1= ON, divide the floating point number value (D1, D0) by (D11, D10) and store the obtained quotient into registers (D21, D20).



API	Mnemonic			Operands	Function	Controllers			
176		MMOV	P	(S) (D)	16-bit→32-bit Conversion	ES2/EX2	SS2	SA2	SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

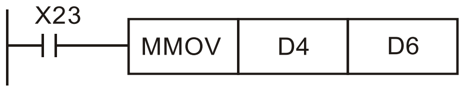
S: Source device (16-bit) **D:** Destination device (32-bit)

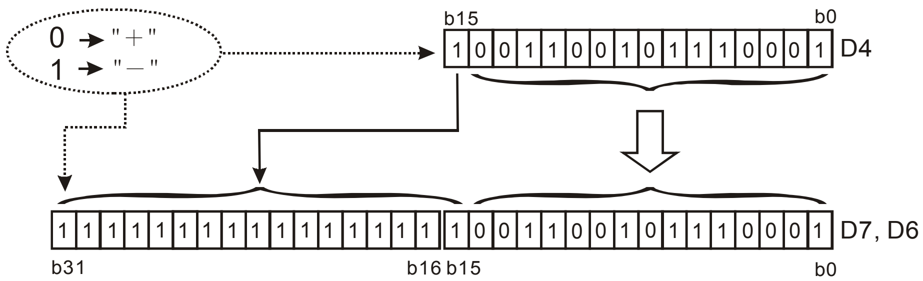
Explanations:

1. MMOV instruction sends the data in 16-bit device **S** to 32-bit device **D**. Sign bit (MSB) of source device will be copied to every bit in the high byte of **D**.

Program example:

When X23 = 0N, 16-bit data in D4 will be sent to D6 and D7.





In the example above, b15 in D4 will be sent to b15~b31 of D7/D6, therefore all bits in b15~b31 will be "negative."

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
177	GPS	S D	GPS data receiving				

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	GPS: 5 steps
S					*	*							*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Sentence identifier for GPS data receiving **D**: Destination device for feedback data

Explanations:

1. GPS data receiving instruction is only applicable on COM1 (RS-232), with communication format: 9600,8,N,1, protocol: NMEA-0183, and communication frequency: 1Hz.
2. Operand **S** is sentence identifier for GPS data receiving. K0: $GPGGA, K1: $GPRMC.
3. Operand **D** stores the received data. Up to 17 consecutive words will be occupied and can not be used repeatedly. Please refer to the table below for the explanations of each **D** device.
   - When **S** is set as K0, sentence identifier $GPGGA is specified. **D** devices refer to:

No.	Content	Range	Format	Note
D + 0	Hour	0 ~ 23	Word	
D + 1	Minute	0 ~ 59	Word	
D + 2	Second	0 ~ 59	Word	
D + 3~4	Latitude	0 ~ 90	Float	Unit: dd.mmmmmm
D + 5	North / South	0 or 1	Word	0(+)→North, 1(-)→South
D + 6~7	Longitude	0 ~ 180	Float	Unit: ddd.mmmmmm
D + 8	East / West	0 or 1	Word	0(+)→East, 1(-)→West
D + 9	GPS data valid / invalid	0, 1, 2	Word	0 = invalid
D + 10~11	Altitude	0 ~9999.9	Float	Unit: meter
D + 12~13	Latitude	-90 ~ 90	Float	Unit: ±dd.ddddd
D + 14~15	Longitude	-180 ~ 180	Float	Unit: ±ddd.ddddd

   - When **S** is set as K1, sentence identifier $GPRMC is specified. **D** devices refer to:

No.	Content	Range	Format	Note
D + 0	Hour	0 ~ 23	Word	
D + 1	Minute	0 ~ 59	Word	
D + 2	Second	0 ~ 59	Word	
D + 3~4	Latitude	0 ~ 90	Float	Unit: dd.mmmmmm

No.	Content	Range	Format	Note
D + 5	North / South	0 or 1	Word	0(+)→North, 1(-)→South
D + 6~7	Longitude	0 ~ 180	Float	Unit: ddd.mmmmmm
D + 8	East / West	0 or 1	Word	0(+)→East, 1(-)→West
D + 9	GPS data valid / invalid	0, 1, 2	Word	0 = invalid
D + 10	Day	1 ~ 31	Word	
D + 11	Month	1 ~ 12	Word	
D + 12	Year	2000 ~	Word	
D + 13~14	Latitude	-90 ~ 90	Float	Unit: ±dd.ddddd
D + 15~16	Longitude	-180 ~ 180	Float	Unit: ±ddd.ddddd

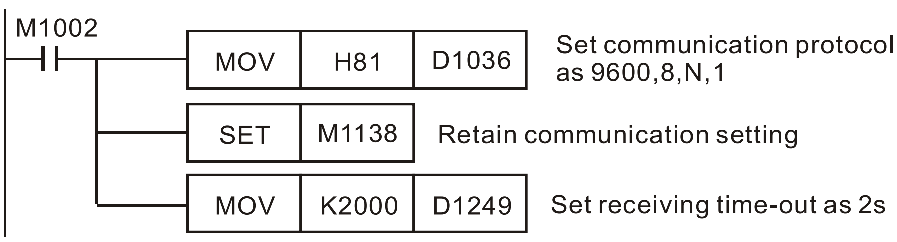
4. When applying GPS instruction, COM1 has to be applied in Master mode, i.e. M1312 has to be enabled to sending request. In addition, M1314 = ON indicates receiving completed. M1315 = ON indicates receiving error. (D1250 = K1, receiving time-out; D1250 = K2, checksum error)
5. Associated M flags and special D registers:

No.	Function
M1312	COM1 (RS-232) sending request
M1313	COM1 (RS-232) ready for data receiving
M1314	COM1 (RS-232) data receiving completed
M1315	COM1 (RS-232) data receiving error
M1138	Retaining communication setting of COM1
D1036	COM1 (RS-232) Communication protocol
D1249	COM1 (RS-232) data receiving time-out setting. (Suggested value: >1s)
D1250	COM1 (RS-232) communication error code

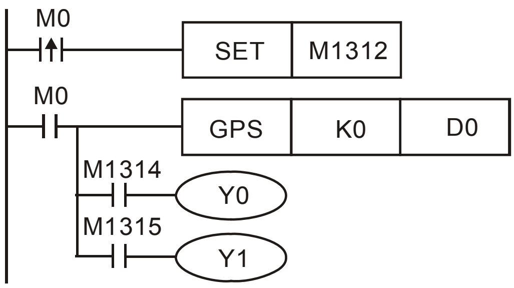
6. Before applying the received GPS data, please check the value in **D**+9. If **D**+9 = 0, the GPS data is invalid.
7. If data receiving error occurs, the previous data in **D** registers will not be cleared, i.e. the previous received data remains intact.

Program example: Sentence identifier: $GPGGA

1. Set COM1communication protocol first



2. Then enable M0 to execute GPS instruction with sentence identifier $GPGGA

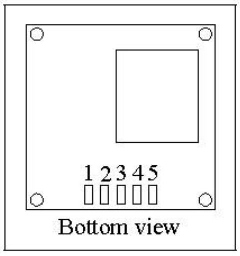


3. When receiving completed, M1314 = ON. When receiving failed, M1315 = ON. The received data will be stored in devices starting with D0.

No.	Content	No.	Content
D0	Hour	D8	East / West
D1	Minute	D9	GPS data valid / invalid
D2	Second	D10~D11	Altitude
D3~D4	Latitude	D12~D13	Latitude. Unit: ±dd.ddddd
D5	North / South	D14~D15	Longitude. Unit: ±ddd.ddddd
D6~D7	Longitude		

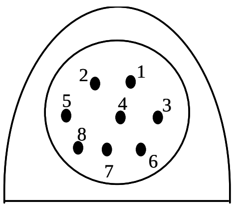
4. Pin number description on GPS module (LS20022)

Pin No. of GPS	1	2	3	4	5
Definition	VCC(+5V)	Rx	Tx	GND	GND



5. Pin number description on PLC COM1:

Pin No. of COM1	1	2	3	4	5	6	7	8
Definition	VCC(+5V)		--	Rx	Tx	--	--	GND



API	Mnemonic		Operands	Function	Controllers
178	D	SPA	(S) (D)	Solar Panel Positioning	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
S					*	*							*			DSPA: 9 steps
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Start device for input parameters **D**: Start device for output parameters

Explanations:

1. Operand **S** occupies 208 consecutive word registers. The function of each device is as below:

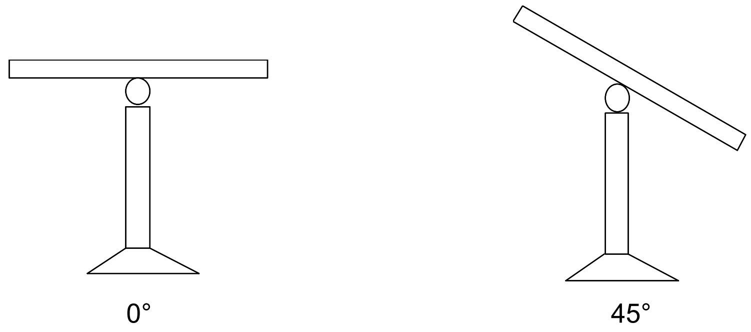
No.	Content	Range	Format	Note
S + 0	Year	2000 ~	Word	Please enter the correct time of the local longitude. Please refer to DTM (parameter 11) for the conversion formula. A simple illustration is as in point 6.
S + 1	Month	1 ~ 12	Word	
S + 2	Day	1 ~ 31	Word	
S + 3	Hour	0 ~ 23	Word	
S + 4	Minute	0 ~ 59	Word	
S + 5	Second	0 ~ 59	Word	
S + 6~7	Time difference (Δt) (sec)	± 8000	Float	
S + 8~9	Local time zone	± 12	Float	West: negative
S + 10~11	Longitude	± 180	Float	West: negative Unit: degree
S + 12~13	Latitude	± 90	Float	South: negative Unit: degree
S + 14~15	Elevation	0~ 6500000	Float	Unit: meter
S + 16~17	Pressure	0 ~ 5000	Float	Unit: millibar
S + 18~19	Mean annual temperature (MAT)	-273~6000	Float	Unit: °C
S + 20~21	Slope	± 360	Float	
S + 22~23	Azimuth	± 360	Float	
S + 24~25	Atmospheric refraction between sunrise and sunset	± 5	Float	
S +26~207	Reserved for system operation			

2. Operand **D** occupies 8 consecutive word registers. The function of each device is as below:

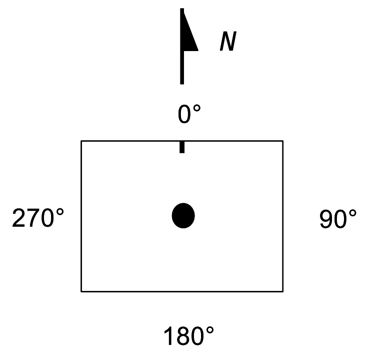
No.	Content	Range	Format	Note
D + 0~1	Zenith	0 ~ 90	Float	Horizontal=0
D + 2~3	Azimuth	0 ~ 360	Float	North point=0
D + 4~5	Incidence	0 ~ 90	Float	
D + 6	Converted DA value of Zenith	0 ~ 2000	Word	1LSB = 0.045 degree
D + 7	Converted DA value of Azimuth	0 ~ 2000	Word	1LSB = 0.18 degree

3. The execution time of SPA instruction costs up to 50ms, therefore we suggest users to execute this instruction with an interval not less than 1 sec, preventing the instruction from taking too much PLC operation time.

4. Definition of Zenith: 0° and 45°.



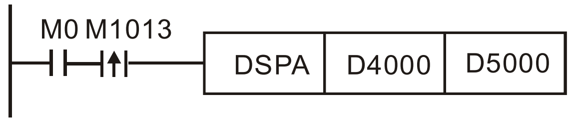
5. Definition of Azimuth:



6. The correct time of the local longitude: If we suppose that it is AM8:00:00 in Taipei, and the longitude is 121.55 degrees east, then the correct time of the local longitude in Taipei should be AM8:06:12. Please refer to API168 DTM instruction (parameter k11) for more explanation.

Program example:

1. Input parameters starting from D4000: 2009/3/23/(y/m/d),10:10:30, Δt = 0, Local time zone = +8, Longitude/Latitude = +119.192345 East, +24.593456 North, Elevation = 132.2M, Pressure = 820m, MAT = 15.0℃, Slope = 0 degree, Azimuth = -10 degree.



2. Output results: D5000: Zenith = F37.2394 degree; D5002: Azimuth = F124.7042 degree.

API	Mnemonic			Operands	Function	Controllers
179	D	WSUM	P	S D n	Sum of multiple devices	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	WSUM, WSUMP: 7 steps
S											*	*	*			DWSUM, DWSUMP: 13
n					*	*							*			steps
D											*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

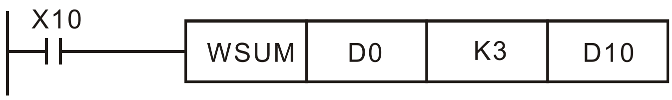
S: Source device **n**: Data length to be summed up **D**: Device for storing the result

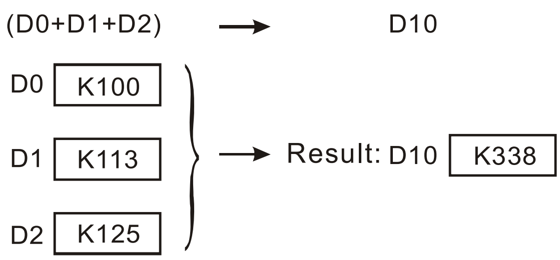
Explanations:

1. WSUM instruction sums up **n** devices starting from **S** and store the result in **D.**
2. If the specified source devices **S** are out of valid range, only the devices in valid range will be processed.
3. Valid range for **n**: 1~64. If the specified **n** value is out of the available range (1~64), PLC will take the upper (64) or lower (1) bound value as the set value.

Program example:

When X10 = ON, 3 consecutive devices (**n** = 3) from D0 will be summed up and the result will be stored in D10





API	Mnemonic		Operands	Function	Controllers
180	MAND	P	(S1) (S2) (D) (n)	Matrix AND	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F
S_1							*	*	*	*	*	*	*		
S_2							*	*	*	*	*	*	*		
D								*	*	*	*	*	*		
n					*	*							*		

Program Steps: MAND, MANDP: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Matrix source device 1 **S_2**: Matrix source device 2 **D**: Operation result

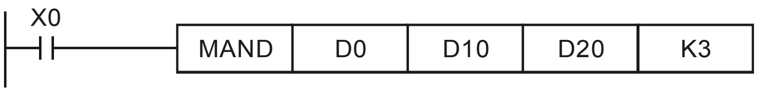
n: Matrix length (**n** = K1~K256)

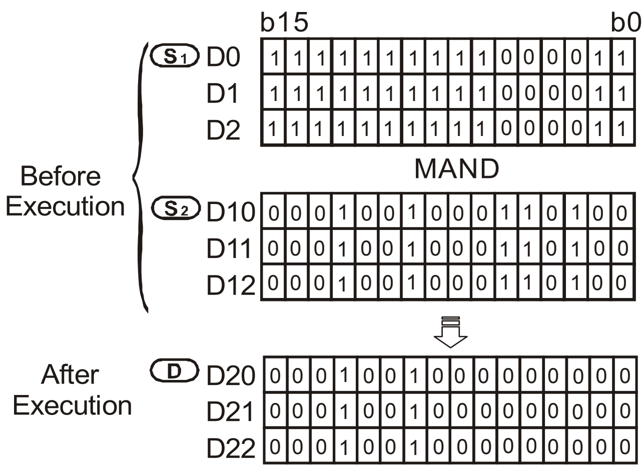
Explanations:

1. MAND instruction performs matrix AND operation between matrix source device 1 and 2 with matrix length **n** and stores the operation result in **D**.
2. Rule of AND operation: the result is 1 only when both two bits are 1; otherwise the result is 0.
3. If operands **S_1**, **S_2**, **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable.

Program Example:

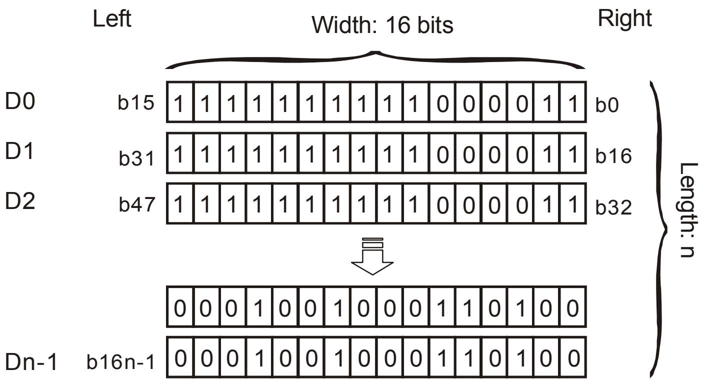
When X0 = ON, MAND performs matrix AND operation between 16-bit registers D0~D2 and 16-bit registers D10~D12. The operation result is then stored in 16-bit registers D20~D22.





Points to note:

1. A matrix consists of more than 1 consecutive 16-bit registers. The number of registers is indicated as the matrix length (n). A matrix contains 16 × n bits (points) and the matrix instructions conduct bit operation, i.e. operation is performed bit by bit.
2. Matrix instructions designate a single bit of the 16 × n bits (b_0 ~ b_{16n-1}) for operation. The bits in matrix are not operated as value operation.
3. The matrix instructions process the moving, copying, comparing and searching of one-to-many or many-to-many matrix operation, which are a very handy and important application instructions.
4. The matrix operation requires a 16-bit register for designating a bit among the 16n bits in the matrix. The register is the Pointer (Pr) of the matrix, designated by the user in the instruction. The valid range of Pr is 0 ~ 16n -1, corresponding to b0 ~ b16n-1 in the matrix.
5. The bit number decreases from left to right (see the figure below). With the bit number, matrix operation such as bit shift left, bit shift right, bit rotation can be performed and identified.



6. The matrix width (C) is fixed as 16 bits.
7. Pr: matrix pointer. E.g. if Pr is 15, the designated bit is b15.
8. Matrix length (R) is n: n = 1 ~ 256.

Example: This matrix is composed of D0, 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: This matrix is composed of K2X20, 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}

Fill “0” into the blank in R0(C_{15}-C_8), R1(C_{15}-C_8), and R2(C_{15}-C_8).

API	Mnemonic		Operands	Function	Controllers
181	MOR	P	S1 S2 D n	Matrix OR	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1							*	*	*	*	*	*	*			MOR, MORP: 9 steps
S_2							*	*	*	*	*	*	*			
D								*	*	*	*	*	*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Matrix source device 1 **S_2**: Matrix source device 2. **D**: Operation result

n: Matrix length (**n** = K1~K256)

Explanations:

1. MOR instruction performs matrix OR operation between matrix source device 1 and 2 with matrix length **n** and stores the operation result in **D**.
2. Rule of matrix OR operation: the result is 1 if either of the two bits is 1. The result is 0 only when both two bits are 0.
3. If operands **S_1**, **S_2**, **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable.

Program Example:

When X0 = ON, MOR performs matrix OR operation between 16-bit registers D0~D2 and 16-bit registers D10~D12. The operation result is then stored in 16-bit registers D20~D22.

X0	MOR	D0	D10	D20	K3

Before Execution

		b15															b0
S1	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

S2	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

After Execution

D	D20	0	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1
	D21	0	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1
	D22	0	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1

API	Mnemonic		Operands	Function	Controllers
182	MXOR	P	S1 S2 D n	Matrix XOR	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1							*	*	*	*	*	*	*			MXOR, MXORP: 9 steps
S_2							*	*	*	*	*	*	*			
D								*	*	*	*	*	*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Matrix source device 1   **S_2**: Matrix source device 2   **D**: Operation result

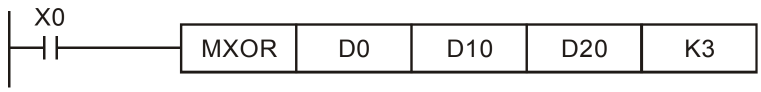
n: Matrix length (**n** = K1~K256)

Explanations:

1. MXOR instruction performs matrix XOR operation between matrix source device 1 and 2 with matrix length **n** and stores the operation result in **D**
2. Rule of matrix XOR operation: the result is 1 if the two bits are different. The result is 0 if the two bits are the same
3. If operands **S_1**, **S_2**, **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable..

Program Example:

When X0 = ON, MXOR performs matrix XOR operation between 16-bit registers D0~D2 and 16-bit registers D10~D12. The operation result is then stored in 16-bit registers D20~D22



Before Execution

		b15															b0
S1	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

MXOR

S2	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

After Execution

D	D20	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0
	D21	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0
	D22	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0

API	Mnemonic		Operands	Function	Controllers
183	MXNR	P	S1 S2 D n	Matrix XNR	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1							*	*	*	*	*	*	*			MXNR, MXNRP: 9 steps
S_2							*	*	*	*	*	*	*			
D								*	*	*	*	*	*			
n					*	*							*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Matrix source device 1 **S_2**: Matrix source device 2 **D**: Operation result

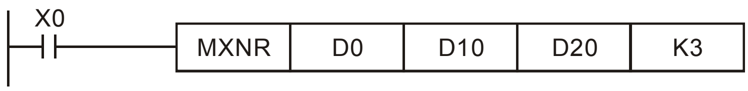
n: Matrix length (K1~K256)

Explanations:

1. MXNR instruction performs matrix XNR operation between matrix source device 1 and 2 with matrix length **n** and stores the operation result in **D**.
2. Rule of matrix XNR operation: The result is 1 if the two bits are the same. The result is 0 if the two bits are different.
3. If operands **S_1**, **S_2**, **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable.

Program Example:

When X0 = ON, MXNR performs matrix XNR operation between 16-bit registers D0~D2 and 16-bit registers D10~D12. The operation result is then stored in 16-bit registers D20~D22.



			b15															b0
Before Execution	S1	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
			MXNR															
	S2	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
After Execution	D	D20	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1
		D21	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1
		D22	1	0	1	0	0	1	0	1	0	0	0	0	1	1	1	1

API	Mnemonic		Operands	Function	Controllers
184	MINV	P	(S) (D) (n)	Matrix inverse	ES2/EX2 SS2 SA2 SX2

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			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

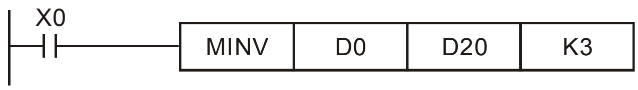
S: Matrix source device **D**: Operation result **n**: Matrix length (K1~K256)

Explanations:

1. MINV instruction performs inverse operation on matrix source device **S** with matrix length **n** and stores the result in **D**.
2. If operands **S**, **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable.

Program Example:

When X0 = ON, MINV performs inverse operation on 16-bit registers D0~D2. The operation result is then stored in 16-bit registers D20~D22



Before Execution (S)

	b15															b0
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

MINV

⇩

After Execution (D)

D20	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
D21	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
D22	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

API	Mnemonic		Operands	Function	Controllers
185	MCMP	P	(S1) (S2) (n) (D)	Matrix compare	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1							*	*	*	*	*	*	*			MCMP, MCMPP: 9 steps
S_2							*	*	*	*	*	*	*			
n					*	*							*			
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Matrix source device 1 **S_2**: Matrix source device 2 **n**: Matrix length (K1~K256)

D: Pointer Pr; comparison result (bit number)

Explanations:

1. MCMP instruction compares each bit between matrix **S_1** and matrix **S_2** and stores the bit number of the comparison result in **D**. The comparison starts from the next bit of the pointer.
2. The matrix comparison flag (M1088) decides to compare between equivalent values (M1088 = ON) or different values (M1088 = OFF). When the comparison is completed, it will stop immediately and M1091= ON to indicate that matched result is found. When the comparison progresses to the last bit, M1089 = ON to indicate that the comparison has come to the end of the matrix and the number of the last bit will be stored in **D**. In next scan cycle, comparison starts again from the first bit (bit 0), at the same time M1090 = ON to indicate the start of the comparison. When **D** (Pr) exceeds the valid range, M1092 = ON to indicate pointer error, and the instruction will be disabled.
3. The matrix operation requires a 16-bit register for designating a bit among the 16n bits in the matrix. The register is the Pointer (Pr) of the matrix, designated by the user in the instruction. The valid range of Pr is 0 ~ 16n -1, corresponding to b0 ~ b16n-1 in the matrix. The value of pointer should not be modified during the execution of matrix instructions so as to prevent execution errors.
4. When M1089 and M1091 take place at the same time, both flags will ON..
5. If operands **S_1**, **S_2**, or **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable.

Program Example:

When X0 goes from OFF to ON with M1090 = OFF (comparison starts from Pr), the search will start from the bit marked with "*" (current Pr value +1) for the bits with different status (M1088 = OFF).

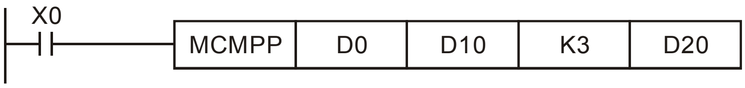
Assume pointer D20 = 2, the following four results (❶, ❷, ❸, ❹) can be obtained when X0 goes from OFF→ON for four times.

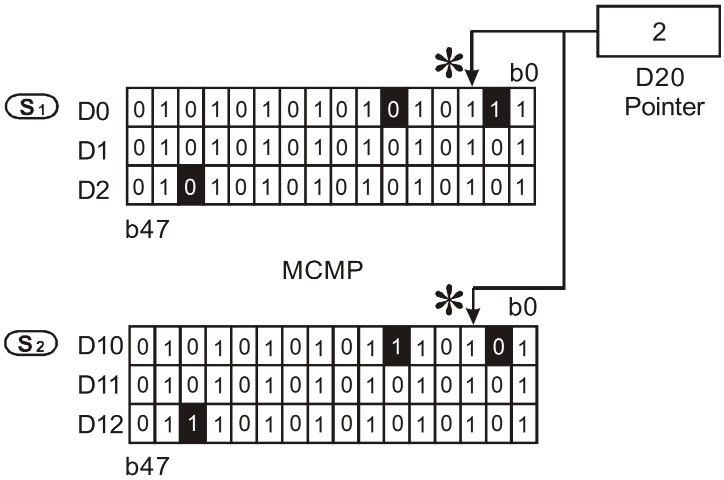
❶ D20 = 5, M1091 = ON (matched result found), M1089 = OFF

❷ D20 = 45, M1091 = ON, M1089 = OFF.

❸ D20 = 47, M1091 = OFF, M1089 = ON (comparison proceeds to he last bit)

❹ D20 = 1, M1091 = ON, = OFF.





Points to note:

Associated flags and registers:

M1088:	Matrix comparison. Comparing between equivalent values (M1088 = ON) or different values (M1088 = OFF)
D1089:	Indicating the end of Matrix. When the comparison reaches the last bit, M1089 = ON
D1090:	Indicating start of Matrix comparison. When the comparison starts from the first bit, M1090 = ON
D1091:	Indicating matrix searching results. When the comparison has matched results, comparison will stop immediately and M1091 = ON
D1092:	Indicating pointer error. When the pointer Pr exceeds the comparison range, M1092 = ON.

API	Mnemonic		Operands	Function	Controllers
186	MBRD	P	(S) (n) (D)	Matrix bit read	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
																MBRD, MBRDP: 7 steps
S							*	*	*	*	*	*	*			
n					*	*							*			
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Matrix source device **n**: Matrix length (K1~K256). **D**: Pointer Pr (bit number)

Explanations:

1. MBRD instruction reads the bit status of the matrix. When MBRD executes, the status of M1094 (Matrix pointer clear flag) will be checked first. If M1094 = ON, Pr value in **D** will be cleared and the instruction reads from the first bit. The bit status is read out and mapped to M1095 (Carry flag for matrix operation). After a bit is read, MBRD checks the status of M1093 (Matrix pointer increasing flag). If M1093 = ON, MBRD instruction will proceed to read the next bit, i.e. Pr value plus 1. When MBRD proceeds to the last bit, M1089 = ON, indicating the end of the Matrix, and **D** records the last bit number. After this, MBRD instruction stops.
2. The Pointer (Pr) of the matrix is designated by the user in the instruction. The valid range of Pr is 0 ~ 16n -1, corresponding to b0 ~ b16n-1 in the matrix. If the Pr value exceeds the valid range, M1092 = ON and the instruction will be disabled.
3. If operands **S** or **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable.

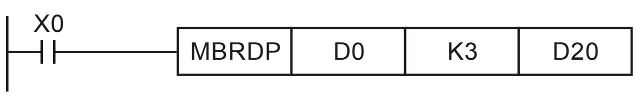
Program Example:

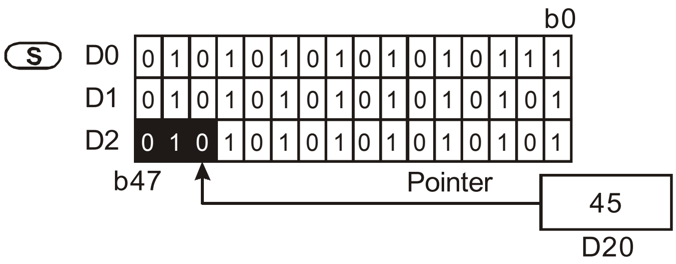
1. When X0 goes from OFF→ON with M1094 = ON (Clear Pr value) and M1093 = ON (Increase Pr value), the reading will start from the first bit and Pr value increases 1 after a bit is read.
2. Assume present value of pointer D20 = 45, the following 3 results (❶, ❷, ❸) can be obtained when X0 is executed from OFF→ON for 3 times.

   ❶ D20 = 45, M1095 = OFF, M1089 = OFF

   ❷ D20 = 46, M1095 = ON (bit status is ON), M1089 = OFF.

   ❸ D20 = 47, M1095 = OFF, M1089 = ON. (reading proceeds to the last bit)





Points to note:

Associated flags and registers:

M1089:	Indicating the end of Matrix. When the comparison reaches the last bit, M1089 = ON
M1092:	Indicating pointer error. When the pointer Pr exceeds the comparison range, M1092 = ON.
M1093	Matrix pointer increasing flag. Adding 1 to the current value of the Pr
M1094	Matrix pointer clear flag. Clear the current value of the Pr to 0
M1095	Carry flag for matrix rotation/shift/output

API	Mnemonic		Operands	Function	Controllers
187	MBWR	P	S n D	Matrix bit write	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S							*	*	*	*	*	*	*			MBWR, MBWRP: 7 steps
n					*	*							*			
D								*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

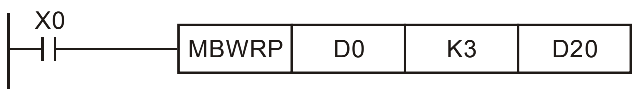
S: Matrix source device **n**: Matrix length (K1~K256) **D**: Pointer Pr (bit number).

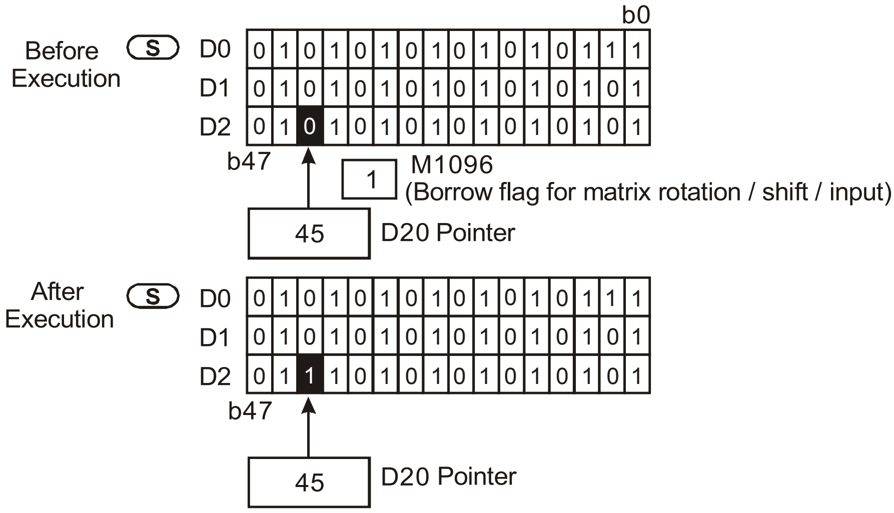
Explanations:

1. MBWR instruction writes the bit status of the matrix. When MBWR executes, the status of M1094 (Matrix pointer clear flag) will be checked first. If M1094 = ON, Pr value in **D** will be cleared and the instruction writes from the first bit. The bit status of M1096 (Borrow flag for matrix operation) is written into the first bit of the matrix. After a bit is written, MBWR checks the status of M1093 (Matrix pointer increasing flag). If M1093 = ON, MBWR instruction will proceed to write the next bit, i.e. Pr value plus 1. When MBWR proceeds to the last bit, M1089 = ON, indicating the end of the Matrix, and **D** records the last bit number. After this, MBWR instruction stops.
2. The Pointer (Pr) of the matrix is designated by the user in the instruction. The valid range of Pr is 0 ~ 16n -1, corresponding to b0 ~ b16n-1 in the matrix. If the Pr value exceeds the valid range, M1092 = ON and the instruction will be disabled.
3. If operands **S** or **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable.

Program Example:

1. When X0 goes from OFF→ON with M1094 = OFF (Starts from Pr value) and M1093 = ON (Increase Pr value), the writing will start from the bit number in Pr and Pr value increases 1 after a bit is written.
2. Assume present value of pointer D20 = 45 and M1096 = ON (1) , the following result can be obtained when X0 is executed once from OFF→ON.





Points to note:

Associated flags and registers:

M1089:	Indicating the end of Matrix. When the comparison reaches the last bit, M1089 = ON
M1092:	Indicating pointer error. When the pointer Pr exceeds the comparison range, M1092 = ON.
M1093	Matrix pointer increasing flag. Adding 1 to the current value of the Pr
M1094	Matrix pointer clear flag. Clear the current value of the Pr to 0
M1096	Borrow flag for matrix rotation/shift/input

API	Mnemonic	Operands	Function	Controllers
188	MBS P	S D n	Matrix bit shift	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

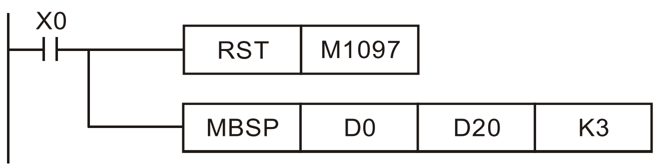
S: Matrix source device **D**: Operation result **n**: Matrix length (K1~K256)

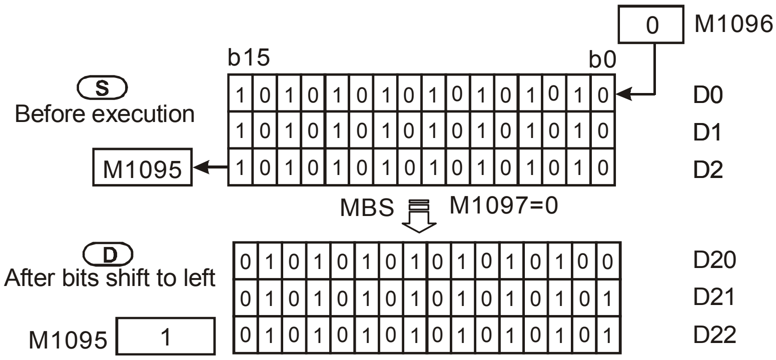
Explanations:

1. MBS instruction shifts the bits in the matrix to the left or the right. M1097 = OFF, bits shift to the left, M1097 = ON, bits shift to the right. The empty bit (left shift: b0; right shift: b16**n**-1) after every bit is shifted once will be filled with the value of M1096 (Borrow flag for matrix operation). The bit which is shifted out of the matrix (left shift: b16**n**-1; right shift: b0) will be sent to M1095 (Carry flag for matrix operation) and operation result is stored in **D**.
2. The pulse execution instruction (MBSP) is generally adopted.
3. If operands **S** or **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable
4. Associated flags:
   M1095: Carry flag for matrix rotation/shift/output
   M1096: Borrow flag for matrix rotation/shift/input
   M1097: Direction flag for matrix rotation/shift

Program Example 1:

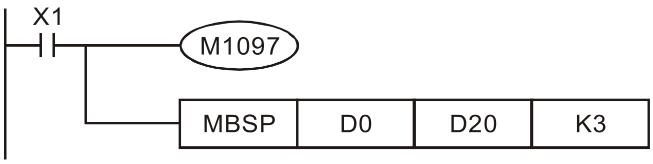
When X0 = ON, M1097 = OFF, indicating a left matrix shift is performed. Assume matrix borrow flag M1096 = OFF (0) and the 16-bit registers D0 ~ D2 will perform a left matrix shift and the result will be stored in the matrix of the 16-bit registers D20 ~ D22, meanwhile the matrix carry flag M1095 will be ON (1). .

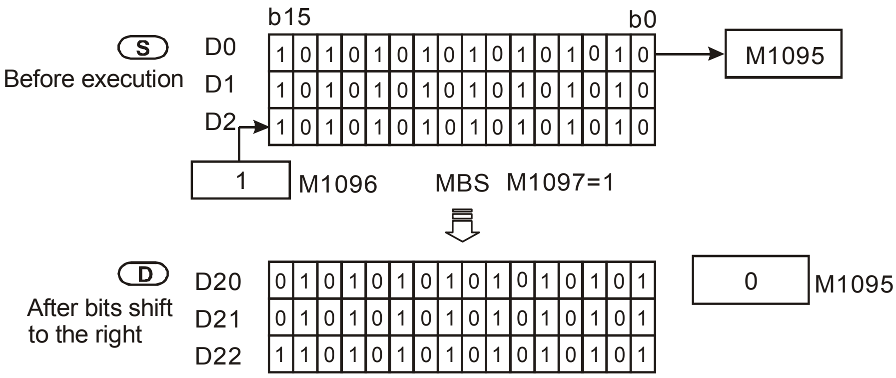




Program Example 2:

When X1 = ON, M1097 = ON, indicating a right matrix shift is performed. Assume matrix borrow flag M1096 = ON (1) and the 16-bit registers D0 ~ D2 will perform a right matrix shift and the result will be stored in the matrix of the 16-bit registers D20 ~ D22, meanwhile the matrix carry flag M1095 will be OFF (0).





API	Mnemonic			Operands	Function	Controllers
189		MBR	P	(S) (D) (n)	Matrix bit rotate	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

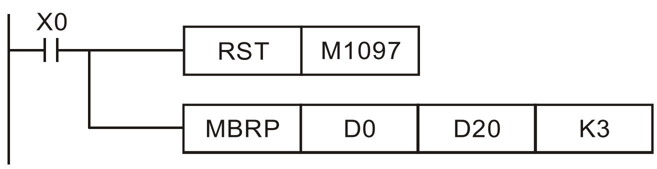
S: Matrix source device **D**: Operation result **n**: Matrix length (K1~K256)

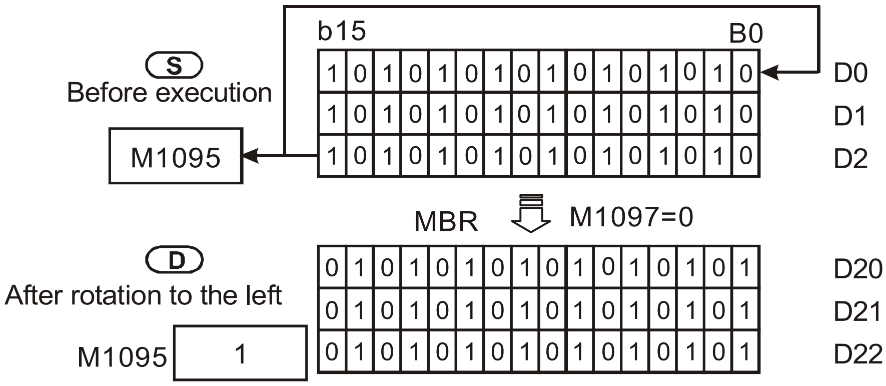
Explanations:

1. MBR instruction rotates the bits in the matrix to the left or the right. M1097 = OFF, bits rotate to the left, M1097 = ON, bits rotate to the right. The empty bit (left rotate: b0; right rotate: b16**n**-1) after rotation performed once will be filled with the bit which is rotated out of the matrix (left rotate: b16**n**-1; right rotate: b0) and the operation result is stored in **D**. In addition, the bit which is rotated out of the matrix will also be moved to M1095 (Carry flag for matrix operation).
2. The pulse execution instruction MBRP is generally adopted.
3. If operands **S** or **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable.
4. Associated flags:
   M1095: Carry flag for matrix rotation/shift/output.
   M1097: Direction flag for matrix rotation/shift

Program Example 1:

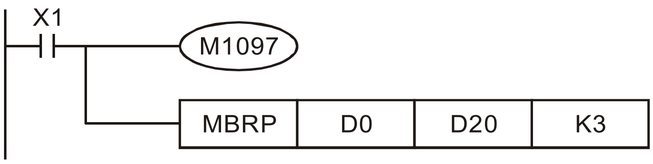
When X0 = ON, M1097 = OFF, indicating a left matrix rotation is performed. The 16-bit registers D0 ~ D2 will perform a left matrix rotation and the result will be stored in the matrix of the 16-bit registers D20 ~ D22. The matrix carry flag M1095 will be ON (1)

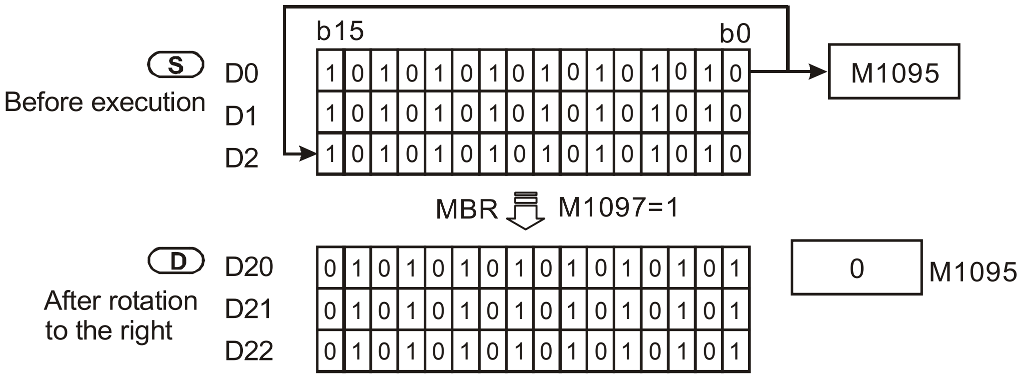




Program Example 2:

When X1 = ON, M1097 = ON, indicating a right matrix rotation is performed. The 16-bit registers D0 ~ D2 will perform a right matrix rotation and the result will be stored in the matrix of the 16-bit registers D20 ~ D22. The matrix carry flag M1095 will be OFF (0).





API	Mnemonic		Operands	Function	Controllers
190	MBC	P	S n D	Matrix bit status count	ES2/EX2 SS2 SA2 SX2

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

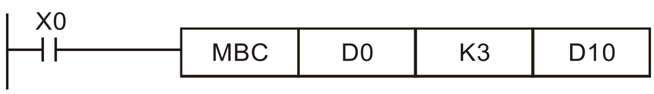
S: Matrix source device **n**: Matrix length (K1~K256) **D**: Operation result

Explanations:

1. MBC instruction counts the number of bit 1 or bit 0 in the matrix with matrix length **n** and stores the counted number in **D**.
2. If operands **S** or **D** use KnX, KnY, KnM, KnS format, only n = 4 is applicable.
3. When M1098 = ON, MBC instruction counts the number of bit 1. M1098 = OFF, MBC counts the number of bit 0. If bits counting result is 0, M1099 = ON
4. Associated flags:
   M1098: Counting the number of bits which are “1” or “0”
   M1099: ON when the bits counting result is “0”..

Program Example:

When X0 = ON with M1098 = ON, MBC instruction counts the number of bit 1 in D0~D2 and store the counted number in D10. When X0 = ON with M1098 = OFF, the instruction counts the number of bit 0 in D0~D2 and store the counted number in D10.



D0	1	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	1	0	0	0	0	1	1
D2	1	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
191	D	PPMR	S1 S2 S D	2-Axis Relative Point to Point Motion	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*							*			DPPMR: 17 steps
S_2					*	*							*			
S					*	*							*			
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Number of output pulses on X axis **S_2**: Number of output pulses on Y axis **S**: Max. point to point output frequency **D**: Pulse output device

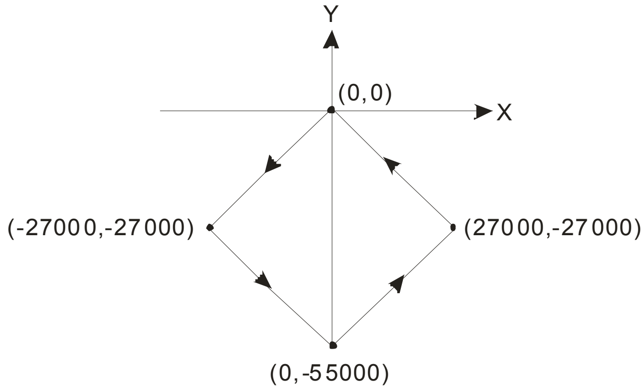
Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function.
2. The instruction only supports the pulse output type: Pulse / Direction.
3. **S_1** and **S_2** specify the number of output pulses (relative positioning) on X axis (Y0) and Y axis (Y2). Range: -2,147,483,648 ~ +2,147,483,647 (The "+/-" sign indicates the forward/backward direction). In forward direction, the present value of pulse output on CH0 (D1031 High, D1030 low), CH1 (D1337 high, D1336 low) increases. In reverse direction pulse output, value in (D1031, D1330) and (D1336, D1337) decreases.
4. **S**: If the max output frequency is smaller than 100Hz, the output will be operated at 100Hz. If the setting is bigger than 100kHz, the output will be operated at 100kHz
5. **D** can designate Y0 only.
   Y0 is the pulse output point of X axis;
   Y1 is the direction signal output of X axis.(OFF: positive; ON: negative)
   Y2 is the pulse output point of Y axis;
   Y3 is the direction signal output of Y axis (OFF: positive; ON: negative)
   When the pulse output is completed, the direction output signal will not be OFF unless the drive contact is OFF.
6. D1340 is start/end frequency setting of X/Y axis. When the set value is smaller than 6Hz, PLC will take 6 Hz as the set value. D1343 is the ramp up/down time setting of X/Y axis. If the ramp up/down time is shorter than 20ms, the frequency will be operated at 20ms. Default: 100ms.
7. When PPMR instruction is enabled, the start frequency and acceleration/deceleration time in Y axis will be the same as the settings in X axis. In addition, setting ramp-down time individually by D1348/D1349 is not recommended because it could lead to the inconsistency between X and Y axes. Also, the flags of "pulse output pause (immediate)" are not applicable. To stop the pulse output, simply turn off the drive contact of this instruction.

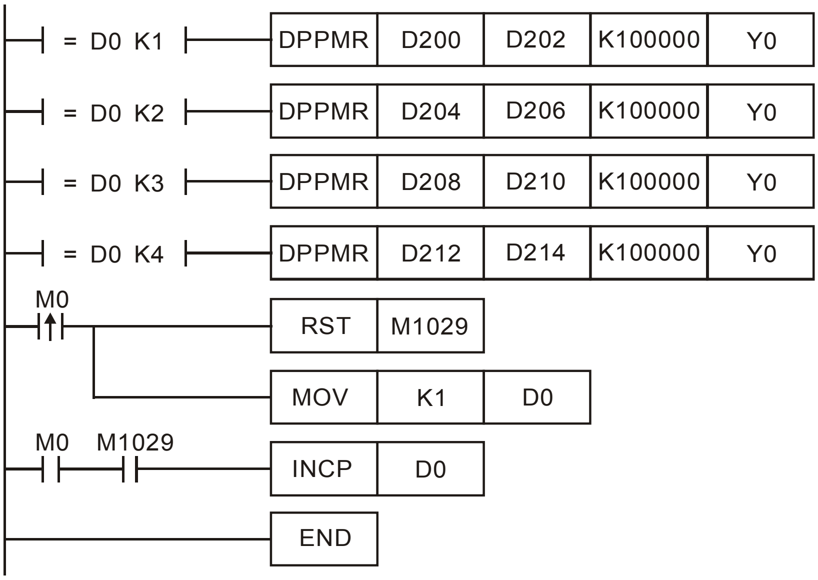
8. For pulse output with ramp-up/down section, if only 1 axis is specified with pulse output number, i.e. another axis is 0, the pulse output will only be performed on the axis with output pulse number. However, if the output pulse number is less than 20 in any of the 2 axes, the ramp-up/down section will be disabled and pulse output will be executed with the frequency not higher than 3kHz.
9. There is no limitation on the number of times for using the instruction. However, assume CH0 or CH1 pulse output is in use, the X/Y axis synchronized output will not be performed.
10. M1029 will be ON when 2-axis synchronized pulse output is completed.

Program Example:

1. Draw a rhombus as the figure below.



2. Steps:
   a) Set the four coordinates (0,0), (-27000, -27000), (0, -55000), (27000, -27000) (as the figure above). Calculate the relative coordinates of the four points and obtain (-27000, -27000), (27000, -28000), (27000, 27000), and (-27000, 27000). Place them in the 32-bit registers (D200, D202), (D204, D206), (D208, D210), (D212, D214).
   b) Design instructions as follows.
   c) RUN the PLC. Set ON M0 to start the 2-axis line drawing.



3. Operation:

   When PLC runs and M0 = ON, PLC will start the first point-to-point motion by 100KHz. D0 will plus 1 whenever a point-to-point motion is completed and the second point-to-point motion will start to execute automatically. The operation pattern repeats until the fourth point-to-point motion is completed.

Points to note:

Associated flags and registers:

M1029:	CH0 (Y0, Y1) pulse output execution completed
D1030:	Present number of Y0 output pulses (HIGH WORD).
D1031:	Present number of Y1 output pulses (LOW WORD).
D1336:	Present value of Y2 pulse output. D1336 (High word)
D1337:	Present value of Y2 pulse output. D1337(Low word)
D1340:	Start/end frequency of pulse output CH0 (Y0), CH1(Y2) for DPPMR/DPPMA instruction
D1343:	Ramp up/down time of pulse output CH0 (Y0), CH1(Y2) for DPPMR/DPPMA instruction

API	Mnemonic		Operands	Function	Controllers
192	D	PPMA	(S1) (S2) (S) (D)	2-Axis Absolute Point to Point Motion	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											DPPMA: 17 steps
S_1					*	*							*			
S_2					*	*							*			
S					*	*							*			
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

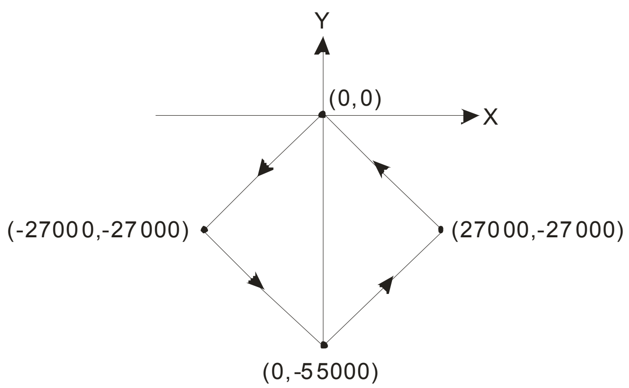
S_1: Number of output pulses on X axis **S_2**: Number of output pulses on Y axis **S**: Max. point to point output frequency **D**: Pulse output device

Explanations:

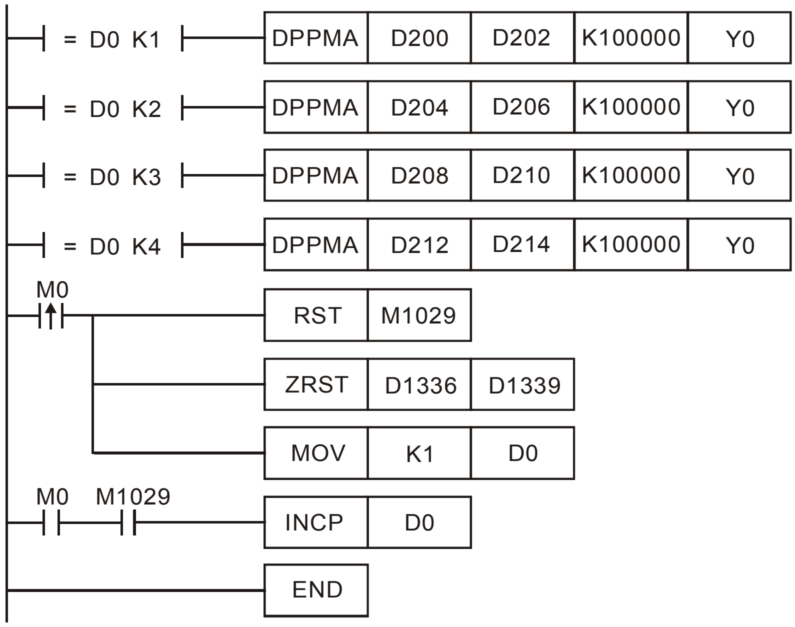
1. For ES2/EX2 models, only V1.20 or above supports the function.
2. The instruction only supports the pulse output type: Pulse / Direction.
3. **S_1** and **S_2** specify the number of output pulses (absolute positioning) on X axis (Y0) and Y axis (Y2). Range: -2,147,483,648 ~ +2,147,483,647 (The "+/-" sign indicates the forward/backward direction). In forward direction, the present value of pulse output on CH0 (D1031 High, D1030 low), CH1 (D1337 high, D1336 low) increases. In reverse direction pulse output, value in (D1031, D1330) and (D1336, D1337) decreases.
4. **D** can designate Y0 only.
   Y0 is the pulse output point of X axis;
   Y1 is the direction signal output of X axis.(OFF: positive; ON: negative)
   Y2 is the pulse output point of Y axis;
   Y3 is the direction signal output of Y axis (OFF: positive; ON: negative)
5. For the rest of the explanations on the instruction, special D and special M, please refer to API 191 DPPMR instruction.

Program Example:

1. Draw a rhombus as the figure below.



2. Steps:
   a) Set the four coordinates (-27000, -27000), (0, -55000), (27000, -27000) and (0,0) (as the figure above). Place them in the 32-bit registers (D200, D202), (D204, D206), (D208, D210), (D212, D214).
   b) Design instructions as follows.
   c) RUN the PLC. Set ON M0 to start the 2-axis line drawing.



3. Operation:

   When PLC runs and M0 = ON, PLC will start the first point-to-point motion by 100KHz. D0 will plus 1 whenever a point-to-point motion is completed and the second point-to-point motion will start to execute automatically. The operation pattern repeats until the fourth point-to-point motion is completed.

API	Mnemonic		Operands	Function	Controllers
193	D	CIMR	S1 S2 S D	2-Axis Relative Position Arc Interpolation	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*							*			DCIMR: 17 steps
S_2					*	*							*			
S													*			
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S1: Number of output pulses of X axis **S2:** Number of output pulses of Y axis **S:** Parameter setting **D:** Pulse output device

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function.
2. The instruction only supports the pulse output type: Pulse / Direction.
3. **S_1** and **S_2** specify the number of output pulses (relative positioning) on X axis (Y0) and Y axis (Y2). Range: -2,147,483,648 ~ +2,147,483,647 (The "+/-" sign indicates the forward/backward direction). In forward direction, the present value of pulse output on CH0 (D1031 High, D1030 low), CH1 (D1337 high, D1336 low) increases. In reverse direction pulse output, value in (D1031, D1330) and (D1336, D1337) decreases.
4. The low word of **S** (settings of direction and resolution): K0 refers to clockwise 20-segment output; K1 refers to counterclockwise 20-segment output; A 90° arc can be drawn (see figure 1 and 2).
5. The high word of **S** (settings of motion time, unit: 0.1sec): Setting range: K2 ~ K200 (0.2 sec. ~ 20 secs.) This instruction is restricted by the maximum pulse output frequency; therefore when the set time is faster than the actual output time, the set time will be automatically modified.

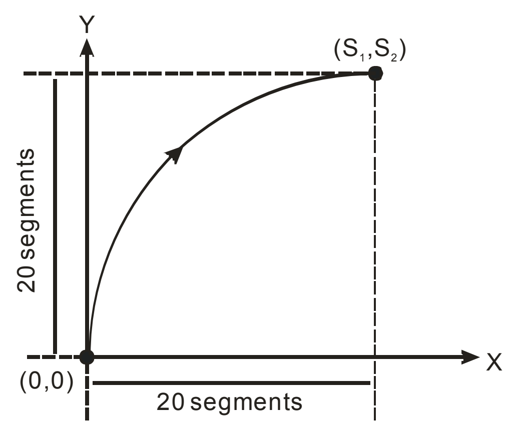


Figure 1

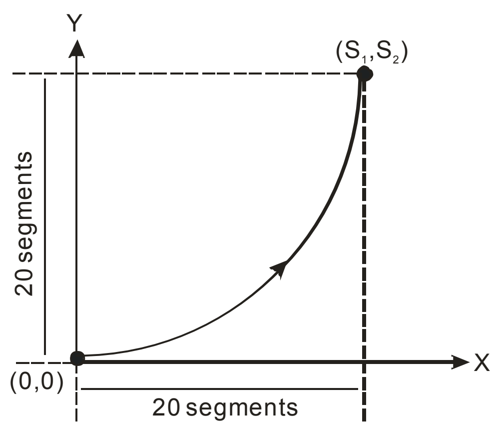


Figure 2

6. Draw four 90° arcs as the figure below.
   When the direction signal is ON, the direction is positive(QI, QIV). When the direction signal is OFF, the direction is negative(QII, QIII). When S is set as K0, the arcs will be clockwise (see figure 3). When S is set as K, the arcs will be counterclockwise (see figure 4).

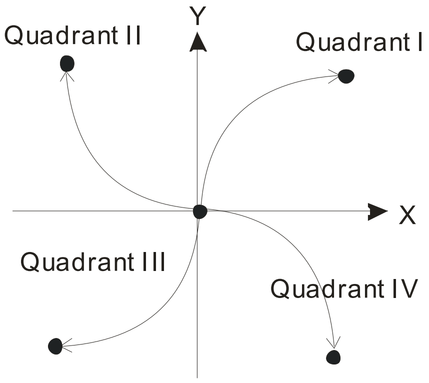


Figure 3

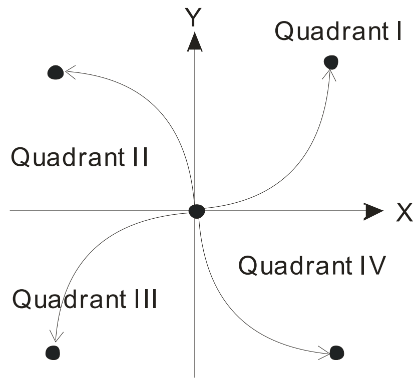


Figure 4

7. The settings of direction and resolution in the lower word of **S** can only be K0 ~ K1
8. The settings of motion time in the high word of **S** shall not be faster than the fastest suggested time. If the motion time is not specified, PLC will use the fastest suggested motion time as the setting. Refer to the table below.

Segments	Max. target position (pulse)	Fastest suggested set time (unit:100ms)
20-segments resolution	500 ~ 20,000	2
	20,000 ~ 29,999	3
	:	:
	Less than 10,000,000	Less than 200

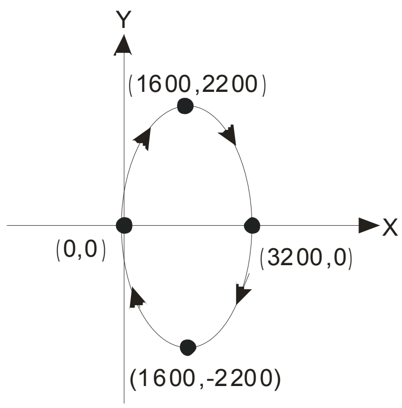
9. **D** can designate Y0 only.
   Y0 is the pulse output point of X axis;
   Y1 is the direction signal output of X axis.(OFF: positive; ON: negative)
   Y2 is the pulse output point of Y axis;
   Y3 is the direction signal output of Y axis (OFF: positive; ON: negative)
   When the pulse output is completed, the direction output signal will not be OFF unless the drive contact is OFF
10. When the 2-axis interpolation is being executed in 20 segments, it takes approximately 2ms for the initialization of this instruction. If only 1 axis is specified with pulse output number (with ramp-up/down section), i.e. another axis is 0, PLC will only execute single-axis positioning according to the specified motion time. If one of the two axes is specified with the pulse number less than 500, PLC will execute 2-axis linear interpolation automatically. However, when either axis is specified for pulse number over 10,000,000, the instruction will not work.
11. If the number of pulses which exceeds the above range is required, the user may adjust the gear ratio of the servo for obtaining the desired results.
12. Every time when the instruction is executed, only one 90° arc can be drawn. It is not necessary

that the arc has to be a 90° arc, i.e. the numbers of output pulses in X and Y axes can be different.

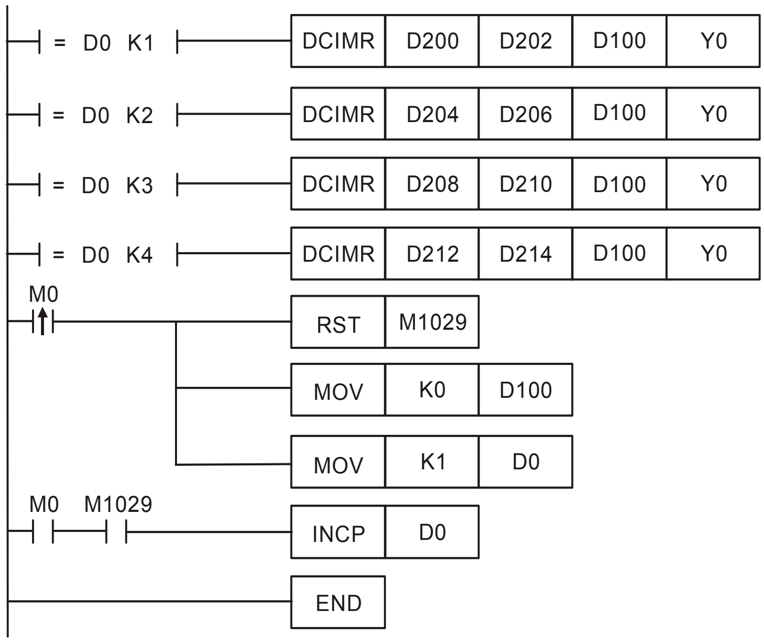
13. There are no settings of start frequency and ramp-up/down time.
14. There is no limitation on the number of times for using the instruction. However, assume CH0 or CH1 output is in use, the X/Y axis synchronized output will not be performed

Program Example 1:

1. Draw an ellipse as the figure below.



2. Steps:
   a) Set the four coordinates (0,0), (1600, 2200), (3200, 0), (1600, -2200) (as the figure above). Calculate the relative coordinates of the four points and obtain (1600, 2200), (1600, -2200), (-1600, -2200), and (-1600, 2200). Place them in the 32-bit registers (D200, D202), (D204, D206), (D208, D210), (D212, D214).
   b) Select “draw clockwise arc” and default “motion time” (**S** = D100 = K0).
   c) RUN the PLC. Set ON M0 to start the drawing of the ellipse.

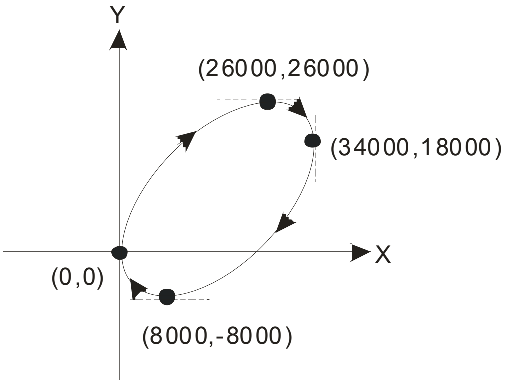


3. Operation:

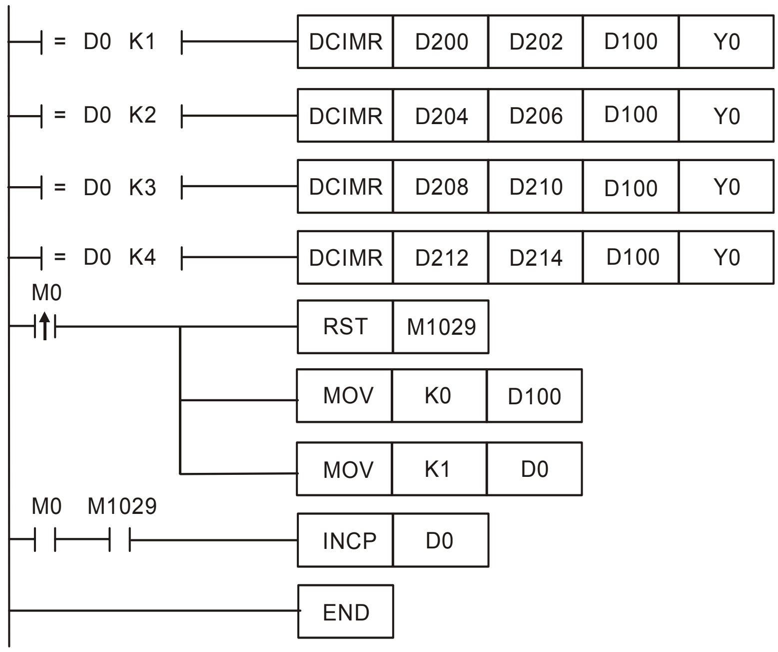
   When PLC runs and M0 = ON, PLC will start the drawing of the first segment of the arc. D0 will plus 1 whenever a segment of arc is completed and the second segment of the arc will start to execute automatically. The operation pattern repeats until the fourth segment of arc is completed.

Program Example 2:

1. Draw a tilted ellipse as the figure below.



2. Steps:
   a) Find the max. and min. coordinates on X and Y axes (0,0), (26000,26000), (34000,18000), (8000,-8000) (as the figure above). Calculate the relative coordinates of the four points and obtain (26000,26000), (8000,-8000), (-26000,-26000), (-8000,8000). Place them respectively in the 32-bit registers (D200,D202), (D204,D206), (D208,D210) and (D212,D214).
   b) Select "draw clockwise arc" and default "motion time" (**S** = D100 = K0).
   c) RUN the PLC. Set ON M0 to start the drawing of a tilted ellipse.



3. Operation:

   When PLC runs and M0 = ON, PLC will start the drawing of the first segment of the arc. D0 will plus 1 whenever a segment of arc is completed and the second segment of the arc will start to execute automatically. The operation pattern repeats until the fourth segment of arc is completed.

Points to note:

Description of associated flags and registers:

M1029: CH0 (Y0, Y1) pulse output execution completed

D1030: Present number of Y0 output pulses (HIGH WORD).

D1031: Present number of Y1 output pulses (LOW WORD).

D1336: Present value of Y2 pulse output. D1336 (High word)

D1337: Present value of Y2 pulse output. D1337(Low word)

API	Mnemonic		Operands	Function	Controllers
194	D	CIMA	S_1 S_2 S D	2-Axis Absolute Position Arc Interpolation	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices X	Y	M	S	Word devices K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*							*			DCIMA: 17 steps
S_2					*	*							*			
S													*			
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

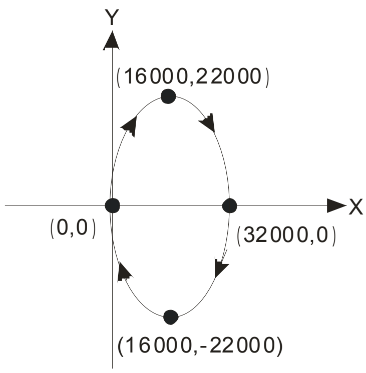
S1: Number of output pulses of X axis **S2:** Number of output pulses of Y axis **S:** Parameter setting **D:** Pulse output device

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function.
2. The instruction only supports the pulse output type: Pulse / Direction.
3. $\mathbf{S_1}$ and $\mathbf{S_2}$ specify the number of output pulses (absolute positioning) on X axis (Y0) and Y axis (Y2). Range: -2,147,483,648 ~ +2,147,483,647. When $\mathbf{S_1}$ and $\mathbf{S_2}$ are bigger than PV of pulse output in CH0 (D1031 High, D1030 low) / CH1 (D1337 high, D1336 low), pulse output will operate in positive direction and the direction signal output Y1, Y3 will be OFF. When $\mathbf{S_1}$ and $\mathbf{S_2}$ are smaller than PV of pulse output, pulse output will operate in negative direction and the direction signal output Y1, Y3 will be ON.
4. For the rest of the explanations on the instruction, special D and special M, please refer to API 193 DCIMR instruction.

Program Example 1:

1. Draw an ellipse as the figure below.

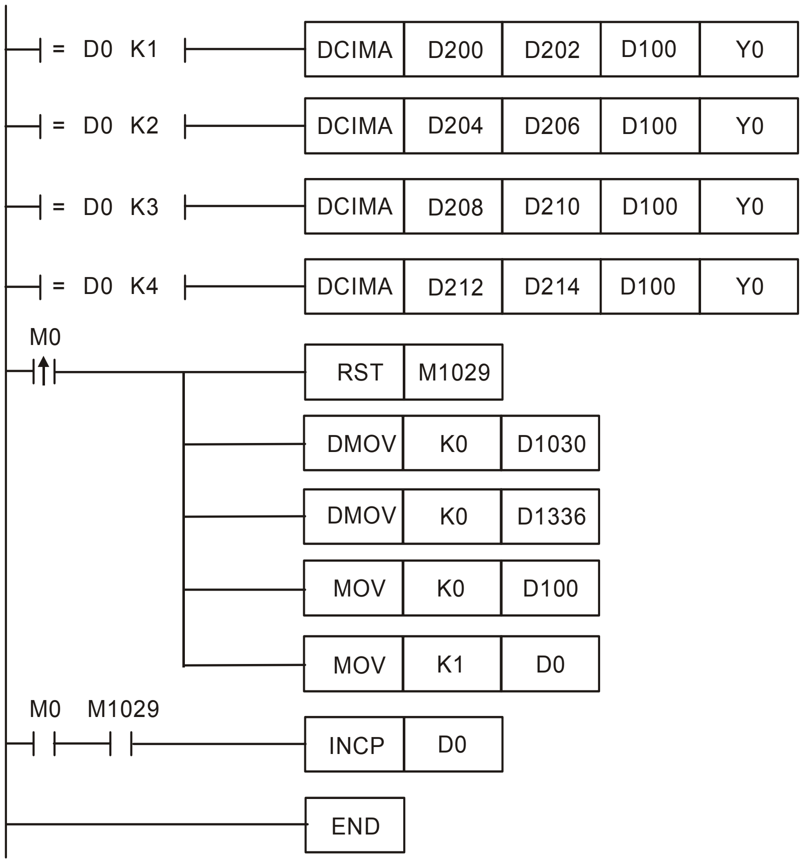


2. Steps:

   a) Set the four coordinates (0,0), (16000, 22000), (32000, 0), (16000, -22000) (as the figure above). Place them in the 32-bit registers (D200, D202), (D204, D206), (D208, D210), (D212, D214).

b) Select “draw clockwise arc” and default “motion time” (**S** = D100 = K0)

c) RUN the PLC. Set ON M0 to start the drawing of the ellipse.

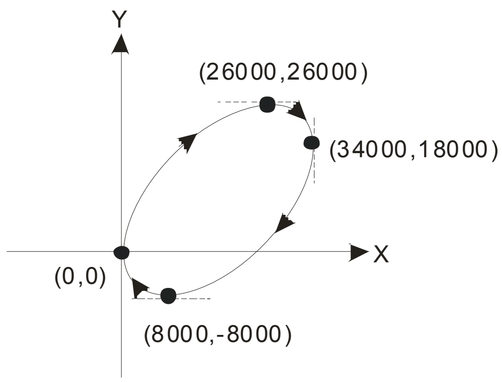


3. Operation:

   When PLC runs and M0 = ON, PLC will start the drawing of the first segment of the arc. D0 will plus 1 whenever a segment of arc is completed and the second segment of the arc will start to execute automatically. The operation pattern repeats until the fourth segment of arc is completed.

Program Example 2:

1. Draw a tilted ellipse as the figure below.

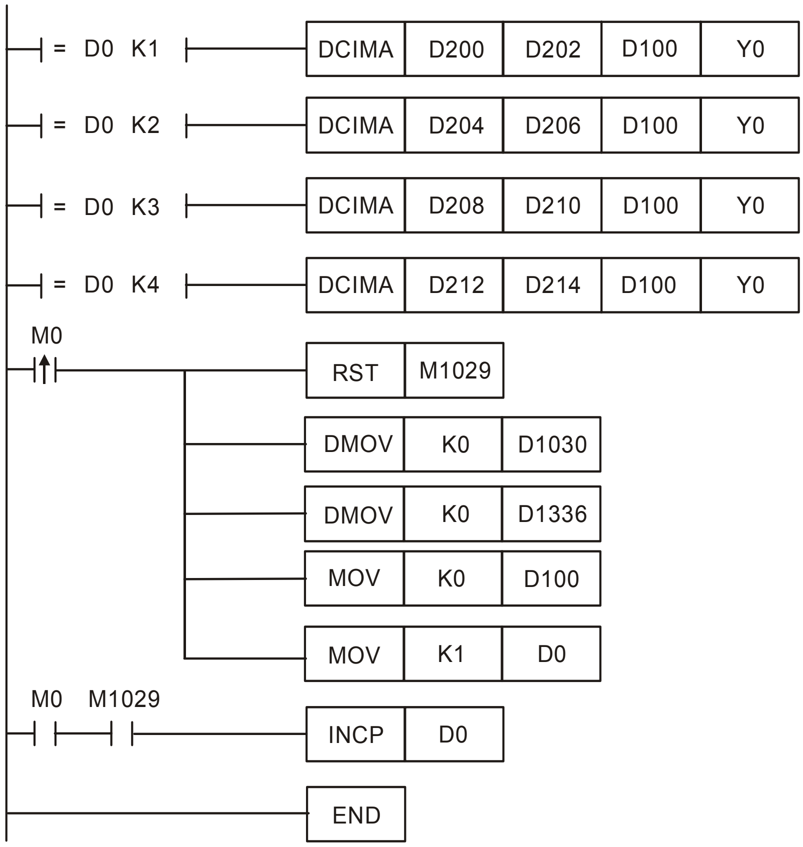


2. Steps:

   a) Find the max. and min. coordinates on X and Y axes (0,0), (26000,26000), (34000,18000), (8000,-8000) (as the figure above). Place them respectively in the 32-bit registers (D200,D202), (D204,D206), (D208,D210) and (D212,D214).

   b) Select “draw clockwise arc” and default “motion time” (**S** = D100 = K0).

   c) RUN the PLC. Set ON M0 to start the drawing of a tilted ellipse.



3. Operation:

   When PLC runs and M0 = ON, PLC will start the drawing of the first segment of the arc. D0 will plus 1 whenever a segment of arc is completed and the second segment of the arc will start to execute automatically. The operation pattern repeats until the fourth segment of arc is completed.

API	Mnemonic		Operands	Function	Controllers
195	D	PTPO	(S1) (S2) (D)	Single-axis pulse output by table	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1													*			DPTPO: 13 steps
S_2													*			
D		*														

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

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source start device **S_2**: Number of segments **D**: Pulse output device

Explanations:

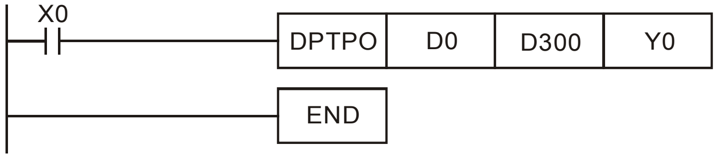
1. **S_1** specifies the output frequency and the number of pulses according to the number of segments set by **S_2**. Each segment occupies consecutive 4 registers in **S_1**. (**S_1**+0) and (**S_1**+1) stores the output frequency; (**S_1**+2) and (**S_1**+3) stores the number of output pulses.
2. Available output frequency for **S_1** : 6Hz~100,000Hz.
3. **S_2** + 0: total number of segments (range: 1 ~ 40). **S_2** + 1: The No. of current executing segment. The number in **S_2** + 1 will be updated when the PLC scan reaches this instruction.
4. **D** can only be designated with output devices Y0 and Y2, i.e. only pulse output is supported. Users need to apply other instructions if a control on direction signal output is required.
5. This instruction does not offer ramp up/down function. Therefore, when the instruction is disabled, the output pulses will stop immediately.
6. There is no limitation on the times of using this instruction, however during each scan cycle, output channel can be driven by one instruction at a time.
7. When the instruction is being executed, changes to the instruction parameter will be invalid.
8. Cyclic output can be performed on this instruction by driving ON M1262.

Program Example:

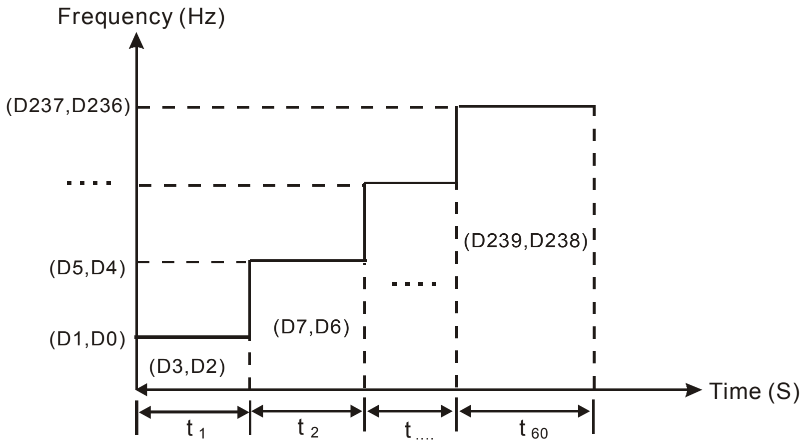
1. When M0 = ON, pulse output will be operated according to the set frequency and number of pulses in every segment.
2. Format of the table:

S_2 = D300, number of segments (D300 = K60)	**S_1** = D0, frequency (**S_1** + 0)	**S_1** = D0, number of output pulses (**S_1** + 2)
K1 (1st segment)	D1, D0	D3, D2
K2 (2nd segment)	D5, D4	D7, D6
:	:	:
K60 (60th segment)	D237, D236	D239, D238

3. Current executing segment can be monitored by D301.



4. Timing diagram:



Points to note:

1. Associated Flags:

M1029	CH0 (Y0) pulse output execution completed.
M1102	CH1 (Y2) pulse output execution completed
M1078	CH0 (Y0) pulse output pause (immediate)
M1104	CH1 (Y2) pulse output pause (immediate)
M1262	Enable cyclic output for table output function of DPTPO instruction. ON = enable.
M1538	Indicating pause status of Y0
M1540	Indicating pause status of Y2

2. Special registers:

D1030	Low word of the present value of Y0 pulse output
D1031	High word of the present value of Y0 pulse output
D1336	Low word of the present value of Y2 pulse output
D1337	High word of the present value of Y2 pulse output

API		Mnemonic	Operands	Function	Controllers
197	D	CLLM	(S1) (S2) (S3) (D)	Close loop position control	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1	*											*				DCLLM: 17 steps
S_2					*	*							*			
S_3					*	*							*			
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Feedback source device **S_2**: Target number of feedbacks **S_3**: Target frequency of output **D**: Pulse output device

Explanations:

1. The corresponding interrupt pointers of **S_1**:

Source device	X4	X6	C243 ~ C254	
Associted outout	Y0	Y2	Y0	Y2
No. of Interrupt pointer	I40□	I60□	I010	I030

□ = 1: rising-edige triggered; □ = 0: falling-edge triggered

a) When **S_1** designates input points X and the pulse output reaches the target number of feedbacks in **S_2**, the output will continue to operate by the frequency of the last shift (end frequency) until interrupts occur on input points X.

b) When **S_1** designates high speed counters and the pulse output reaches the target number of feedbacks in **S_2**, the output will continue to operate by the frequency of the last shift (end frequency) until the feedback pulses reaches the target number.

c) **S_1** can be a high speed counter C or an input point X with external interrupt. If **S_1** is C, DCNT instruction should be executed in advance to enable the high-speed counting function, and EI instruction with I0x0 should be enabled for external interrupts. If **S_1** is X, EI instruction with I0x0 should be enabled for external interrupts.

d) If **S_1** is specifed with counters, DHSCS instruction has to be programmed in user program. Please refer to **Program example 2** for details.

2. Range of **S_2**: -2,147,483,648 ~ +2,147,483,647 (+ / - indicates the positive / negative rotation direction). the present value of pulse output in CH0 (Y0, Y1) and CH1 (Y2, Y3) increases in positive direction and decreases in negative direction. Registers storing present value of pulse output: CH0(D1031 High, 1030 Low), CH1(D1337 High, D1336 Low)

3. If **S_3** is lower than 6Hz, the output will operate at 6Hz; if **S_3** is higher than 100kHz, the output will operate at 100kHz.

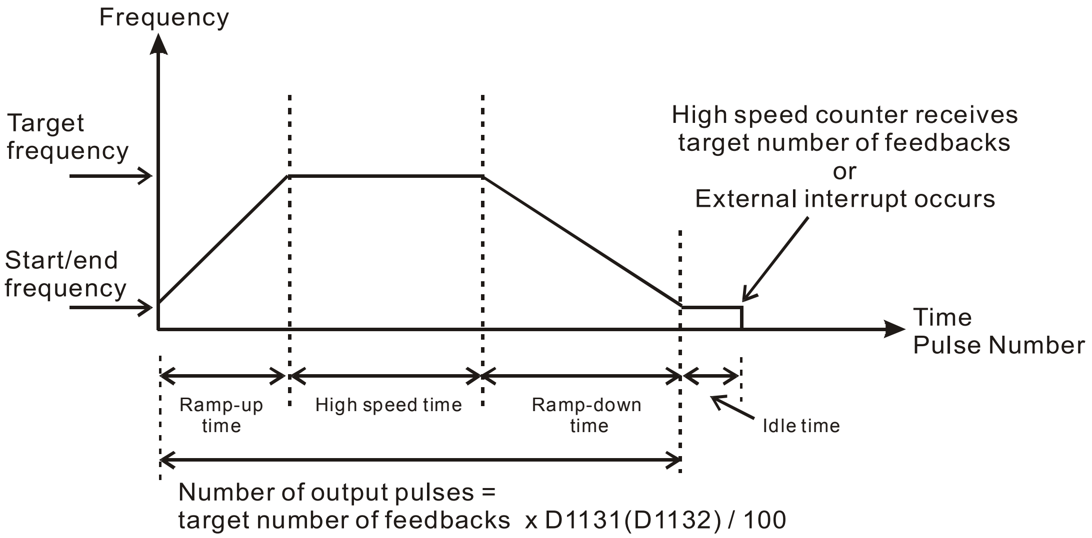
4. **D** can only designate Y0 (Direction signal output: Y1) or Y2 (Direction signal output: Y3). The

direction signal output will be OFF only when the drive contact of the instruction is OFF, i.e. completion of pulse output will not reset Y1 or Y3.

5. D1340 and D1352 stores the start/end frequencies of CH0 and CH1. Min. 6Hz, default: 100Hz.
6. D1343 and D1353 stores the ramp up/down time of CH0 and CH1. If the ramp up/down time is shorter than 20ms, PLC will operate in 20ms. Dafault: 100ms.
7. Ramp-down time of CH0 and CH1 can be particularlily specified by the setting of (M1534, D1348) and (M1535, D1349). When M1534 / M1535 is ON, ramp-down time of CH0 and CH1 is set by D1348 and D1349.
8. D1131 and D1132 are the output/input ratio(%) of the close loop control in CH0 and CH1. K1 refers to 1 output pulse out of 100 feedback pulses; K200 refers to 200 output pulses out of the 100 feedback pulses. In general percentage equation, the value set in D1131 and D1132 represents numerators (output pulses, available range: K1 ~ K10,000) and the denominator (the input feedbacks) is fixed as K100 (System defined).
9. M1305 and M1306 can reverse the direction of CH0, CH1 pulse output. For example, when direction signal output (Y1/Y3) is OFF, pulse output will operate in positive direction. If M1305/M1306 is set ON before the execution of this instruction, the pulse output will be reversed as negative output direction.
10. When **S_1** designates input points X with interrupt pointers, D1244 / D1255 can be applied for setting the idle time as limited pulse number, in case the interrupt is not properly triggered.
11. DCLLM instruction supports Alignment Mark and Mask function. Please refer to **PLSR** instruction for details.

Close Loop Explanations:

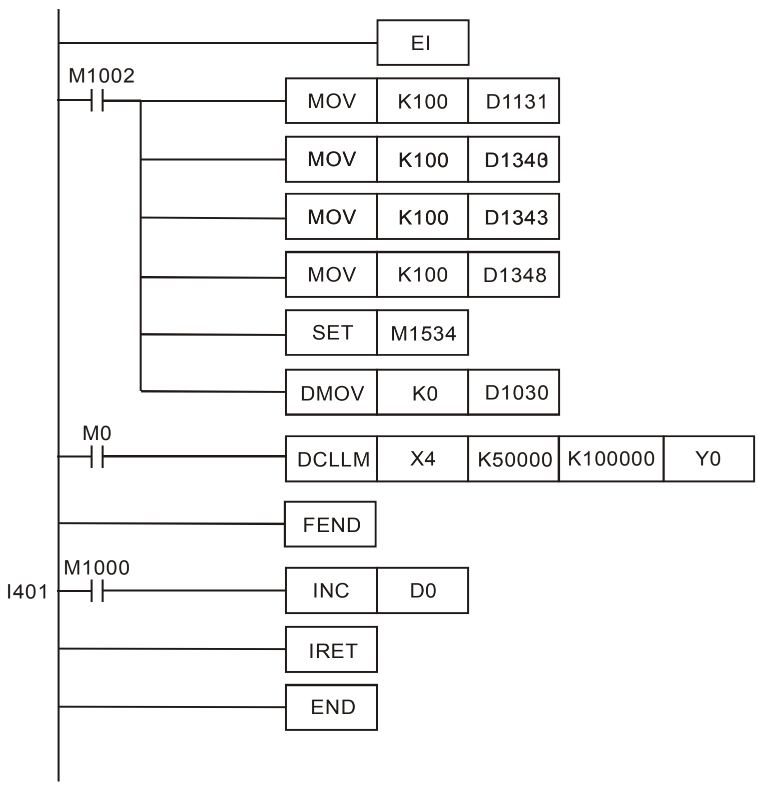
1. Function: Immediately stop the high-speed pulse output according to the number of feedback pulses or external interruption signals.
2. Timing diagram:



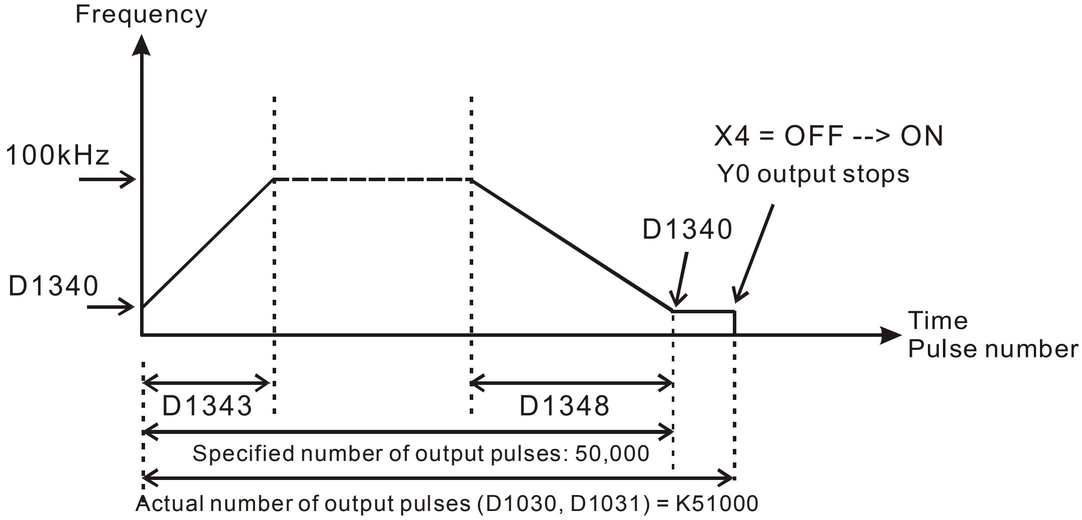
3. Principles for adjusting the completion time of positioning:
   a) The completion time of positioning refers to the total time of “ramp up + high speed + ramp down + idle” (see the figure above). When percentage value (D1131/D1132) is modified, the total number of output pulses will be increased or decreased as well as the completion time.
   b) When $\mathbf{S_1}$ designates input points X with interrupt pointers, D1244 / D1255 can be applied for setting the idle time as limited pulse number, in case the interrupt is not properly triggered.Users can determine if the execution result is good or bad by the length of the idling time. In theory, a bit of idling left is the best result for a positioning.
   c) Owing to the close loop operation, the length of idle time will not be the same in every execution. Therefore, when the content in the special D for displaying the actial number of output pulses is smaller or larger than the calculated number of output pulses (target number of feedbacks x percentage value / 100), users can improve the situation by adjusting the percentage value, ramp-up/ramp-down time or target frequency.

Program Example1: Immediate stop high-speed pulse output by external interrupt

1. Adopt X4 as the input for external interrupt and I401 (rising-edge trigger) as the interrupt pointer. Set target number of feedbacks = 50,000; target frequency = 100kHz; pulse output device: Y0, Y1 (CH0); start/end frequency (D1340) = 100Hz; ramp-up time (D1343) = 100ms; ramp-down time (D1348) = 100ms; percentage value (D1131) = 100; present value of output pulses (D1030, D1031) = 0.

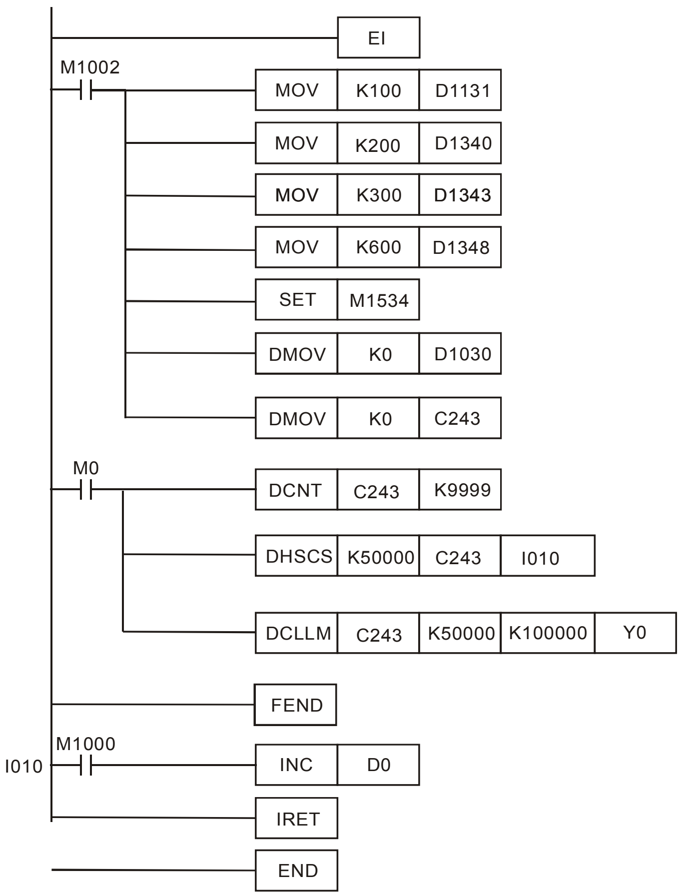


2. Execution results:

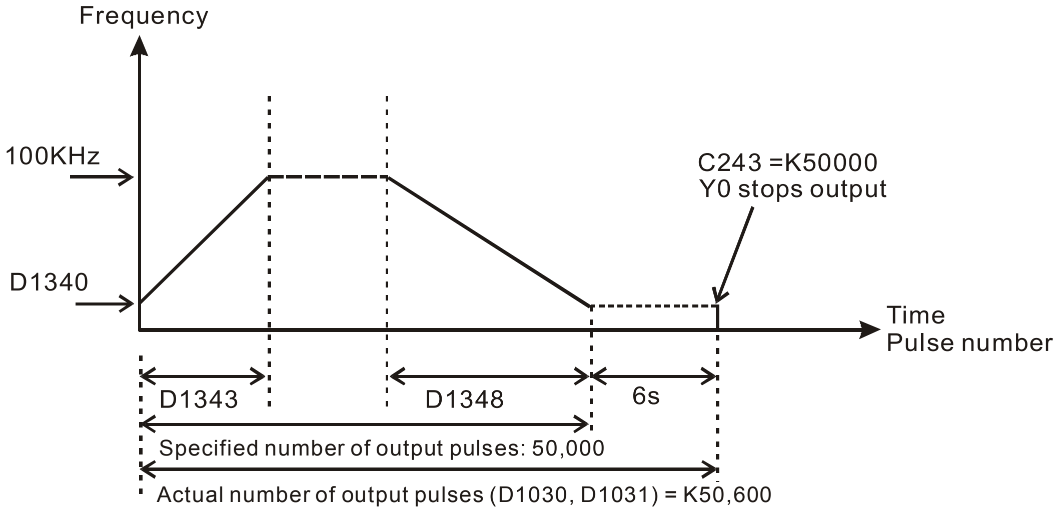


Program Example 2: Immediate stop high-speed pulse output by high speed counter

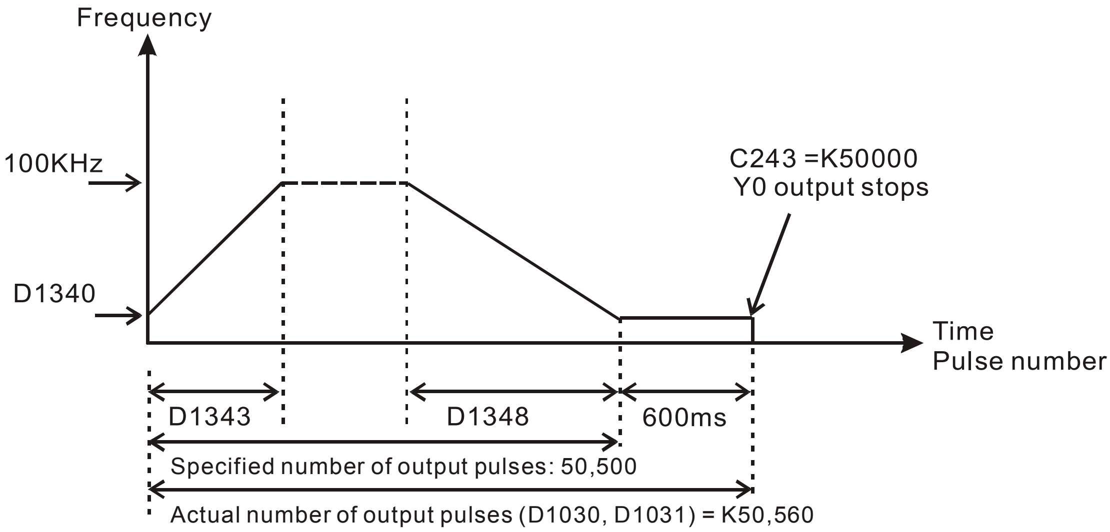
1. Adopt counter C243 (better to be reset before execution) with AB-phase input from the encoder. Set target number of feedbacks = 50,000; target frequency = 100kHz; pulse output device: Y0, Y1 (CH0); start/end frequency (D1340) = 100Hz; ramp-up time (D1343) = 100ms; ramp-down time (D1348) = 100ms; percentage value (D1131) = 100; present value of output pulses (D1030, D1031) = 0..



2. Assume the first execution results are as below:



3. Observe the results of the first execution:
   a) The actual output number 50,600 – specified output number 50,000 = 600
   b) 600 x (1/100Hz) = 6s (idle time)
   c) 3 seconds are too long. Therefore, increase the percentage value (D1131) to K101.
4. Obatin the results of the second execution:



5. Observe the results of the second execution:
   a) The actual output number 50,560 – specified output number 50,500 = 60
   b) 60 x (1/100Hz) = 600ms (idle time)
   c) 600ms is an appropriate value. Therefore, set the percentage value (D1131) as K101 to complete the design.

Points to note:

1. Associated flags:

M1029	CH0 (Y0, Y1) pulse output execution completed.
M1102	CH1 (Y2, Y3) pulse output execution completed.
M1078	M1078 = ON, CH0 (Y0, Y1) pulse output pause (immediate)
M1104	M1104 = ON CH1 (Y2, Y3) pulse output pause (immediate)
M1108	CH0 (Y0, Y1) pulse output pause (ramp down). M1108 = ON during ramp down.
M1110	CH1 (Y2, Y3) pulse output pause (ramp down). M1110 = ON during ramp down.
M1156	Enabling the mask and alignment mark function on I400/I401(X4) corresponding to Y0.
M1158	Enabling the mask and alignment mark function on I600/I601(X6) corresponding to Y2.
M1538	Indicating pause status of CH0 (Y0, Y1).M1538 = ON when output paused.
M1540	Indicating pause status of CH1 (Y2, Y3). M1540 = ON when output paused
M1305	Reverse CH0 (Y0, Y1) pulse output direction. M1305 = ON, pulse output direction is reversed.
M1306	Reverse CH1 (Y2, Y3) pulse output direction. M1306 = ON, pulse output direction is reversed
M1347	Auto-reset CH0 (Y0, Y1) when high speed pulse output completed. M1347 will be reset after CH0 (Y0, Y1) pulse output is completed.
M1524	Auto-reset CH1 (Y2, Y3) when high speed pulse output completed. M524 will be reset after CH1 (Y2, Y3) pulse output is completed.
M1534	Enable ramp-down time setting on Y0. Has to be used with D1348
M1535	Enable ramp-down time setting on Y2. Has to be used with D1349

2. Special registers:

D1026:	Pulse number for masking Y0 when M1156 = ON (Low word). The function is disabled when set value≦0. (Default = 0)
D1027:	Pulse number for masking Y0 when M1156 = ON (High word). The function is disabled when set value≦0. (Default = 0)
D1135:	Pulse number for masking Y2 when M1156 = ON (Low word). The function is disabled when set value≦0. (Default = 0)
D1136:	Pulse number for masking Y2 when M1156 = ON (High word). The function is disabled when set value≦0. (Default = 0)
D1030:	Low word of the present value of CH0 (Y0, Y1) pulse output
D1031:	High word of the present value of CH0 (Y0, Y1) pulse output
D1131:	Input/output percentage value of CH0 (Y0, Y1) close loop control. Default: K100

D1132: Input/output percentage value of CH1 (Y2, Y3) close loop control. Default: K100

D1244: Idle time (pulse number) setting of CH0 (Y0, Y1) The function is disabled if set value≦0.

D1245: Idle time (pulse number) setting of CH2 (Y2, Y3) The function is disabled if set value≦0.

D1336: Low word of the present value of CH1 (Y2, Y3) pulse output

D1337: High word of the present value of CH1 (Y2, Y3) pulse output

D1340: Start/end frequency of the 1st group pulse output CH0 (Y0, Y1). Default: K100

D1352: Start/end frequency of the 2st group pulse output CH1 (Y2, Y3). Default: K100

D1343: Ramp up/down time of the 1st group pulse output CH0 (Y0, Y1). Default: K100

D1353: Ramp up/down time of the 2nd group pulse output CH1 (Y2, Y3). Default: K100

D1348: CH0(Y0, Y1) pulse output. When M1534 = ON, D1348 stores the ramp-down time. Default: K100

D1349: CH1(Y2, Y3) pulse output. When M1535 = ON, D1349 stores the ramp-down time. Default: K100

API		Mnemonic		Operands	Function	Controllers
198	D	VSPO		S1 S2 S3 D	Variable speed pulse output	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1													*			DVSPO: 17 steps
S_2					*	*							*			
S_3					*	*							*			
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

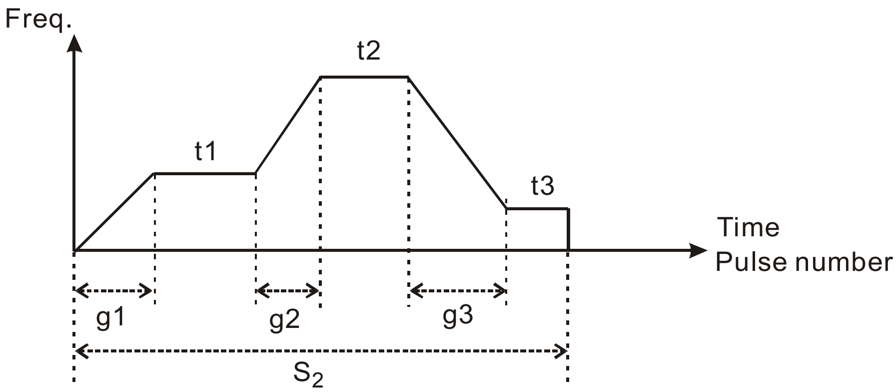
S_1: Target frequency of output **S_2**: Target number of pulses **S_3**: Gap time and gap frequency **D**: Pulse output device (Y0, Y2)

Explanations:

1. Max frequency for **S_1**: 100kHz. Target frequency can be modified during the execution of instruction. When **S_1** is modified, VSPO will ramp up/down to the target frequency according to the ramp-up gap time and gap frequency set in **S_3**.
2. **S_2** target number of pulses is valid only when the instruction is executed first time. **S_2** can NOT be modified during the execution of instruction. **S_2** can be a negative value, however, if the output direction is not specified in D1220/D1221, PLC will take this value as a positive value. When target number of pulses are specified with 0, PLC will perform continuous output.
3. **S_3** occupies 2 consecutive 16-bit devices. **S_3**+0 stores the gap frequency **S_3**+1 stores the gap time. Parameter setting can be modified during the execution of instruction. Set range for **S_3**+0: 6Hz ~ 32767Hz; set range for **S_3**+0: 1ms ~ 80ms. If set value exceeds the available range, PLC will take the upper or lower bound value.
4. **D** pulse output device supports only Y0 and Y2. If Y1 and Y3 is required for output direction control, D1220 or D1221 has tobe set as K1(Pulse/Dir).
5. Parameters set in **S_3** can only be modified while modifying the value in **S_1**. When target frequency is set as 0, PLC will ramp down to stop according to parameters set in **S_3**. When the output is stopped, PLC will enable the flags indicating pause status (Y0: M1538, Y2: M1540). If target frequency other than 0 is specified again, pulse output will ramp up to target frequency and operates untill target number of pulses are completed.

Function Explanations:

Pulse output diagram:



1. Definitions:

   t1 → target frequency of 1st shift

   t2 → target frequency of 2nd shift

   t3 → target frequency of 3rd shift

   g1 → ramp-up time of 1st shift

   g2 → ramp-up time of 2nd shift

   g3 → ramp-down time of 3rd shift

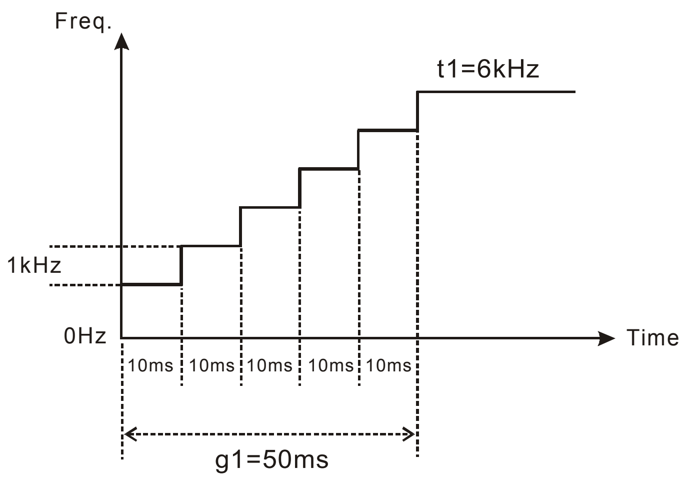
   S_2 → total output pulses

2. Explanations on each shift:

   ◆ 1st shift:

   Assume t1 = 6kHz, gap freqency = 1kHz, gap time = 10ms

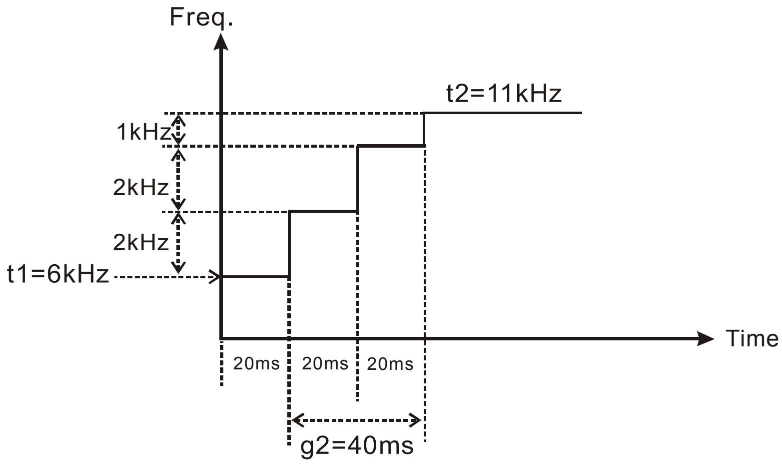
   Ramp-up steps of 1st shift:



- 2^{nd} shift:

  Assume t2 = 11kHz, internal frequency = 2kHz, gap time = 20ms

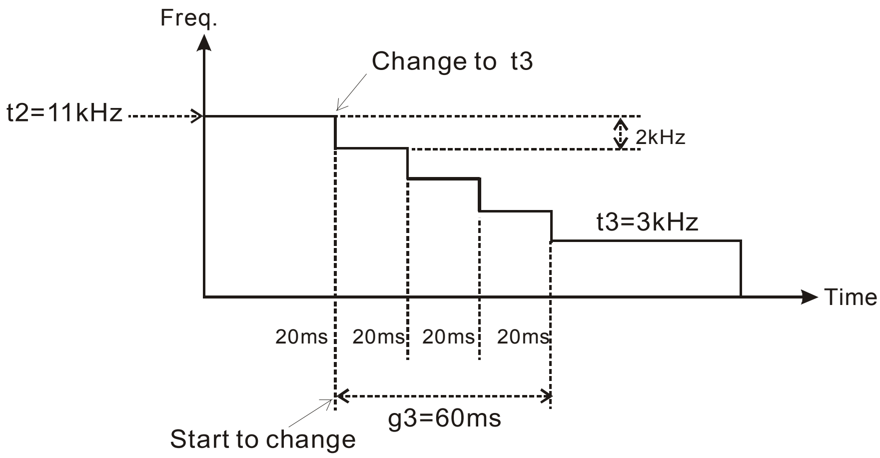
  Ramp-up steps of 2^{nd} shift:



- 3^{rd} shift:

  Assume t3 = 3kHz, gap frequency = 2kHz, gap time = 20ms

  Ramp-down steps of 3^{rd} shift:



- For program examples please refer to API 199

Points to note:

1. Associated flags:

M1029	CH0 (Y0, Y1) pulse output execution completed
M1102	CH1 (Y2, Y3) pulse output execution completed
M1078	Y0 pulse output pause (immediate)
M1104	Y2 pulse output pause (immediate)
M1305	Reverse Y1 pulse output direction in high speed pulse output instructions
M1306	Reverse Y3 pulse output direction in high speed pulse output instructions
M1538	Indicating pause status of Y0

M1540 Indicating pause status of Y2

2. Special register explanations:

D1030 Low word of the present value of Y0 pulse output

D1031 High word of the present value of Y0 pulse output

D1336 Low word of the present value of Y2 pulse output

D1337 High word of the present value of Y2 pulse output

D1220 Pulse output mode setting of CH0 (Y0, Y1). Please refer to PLSY instruction.

D1221 Pulse output mode setting of CH1 (Y2, Y3). Please refer to PLSY instruction

API	Mnemonic		Operands	Function	Controllers
199	D	ICF	S1 S2 D	Immediately change frequency	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1													*			DVSPO: 13 steps
S_2					*	*							*			
D		*														

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

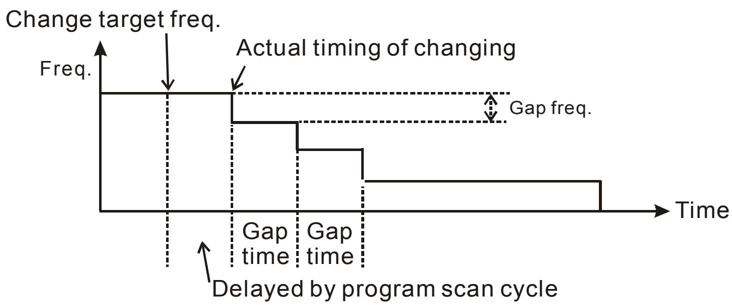
S_1: Target frequency to be changed **S_2**: Gap time and gap frequency **D**: Pulse output device (Y0, Y2)

Explanations:

1. Max frequency for **S_1**: 100kHz. When ICF instruction executes, frequecy changing will start immediately with ramp-up/down process.
2. ICF instruction has to be executed after the execution of DVSPO or DPLSY instructions. When the instruction is used together with DVSPO, operands **S_1**, **S_2**, **D** of DICF has to be assigned the same device with **S_1**, **S_3**, **D** of DVSPO. When the instruction is used with DPLSY, operands **S_1** and **D** has to be assigned the same device with **S_1** and **D** of DPLSY**.**
3. If ICF instruction is used with DPLSY instruction, operand **S_2** is invalid.
4. When ICF instruction is used with DVSPO instruction, parameter setting of **S_2** functions the same as **S_3** in DVSPO instruction, specifying the gap time and gap frequency of ramp-up/down process.
5. **D** pulse output device supports only Y0 and Y2.
6. The instruction is suggested to be applied in interrupt subroutines for obtaining the better response time and eexecution results
7. For associated flags and registers, please refer to **Points to note** of API 198 DVSPO instruction.

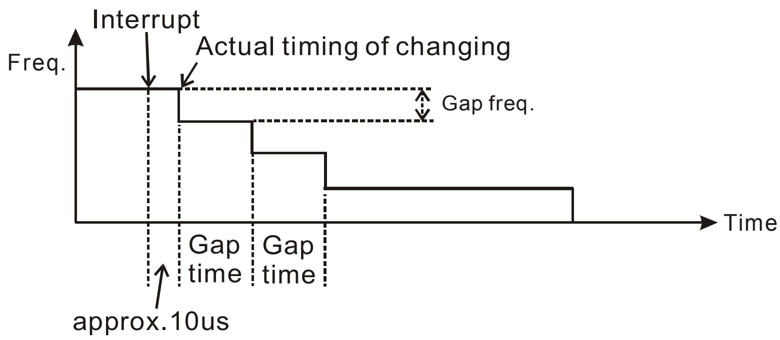
Function Explanations:

1. If users change the target frequency by using DVSPO instruction, the actual changing timing will be delayed due to the program scan time and the gap time as below.



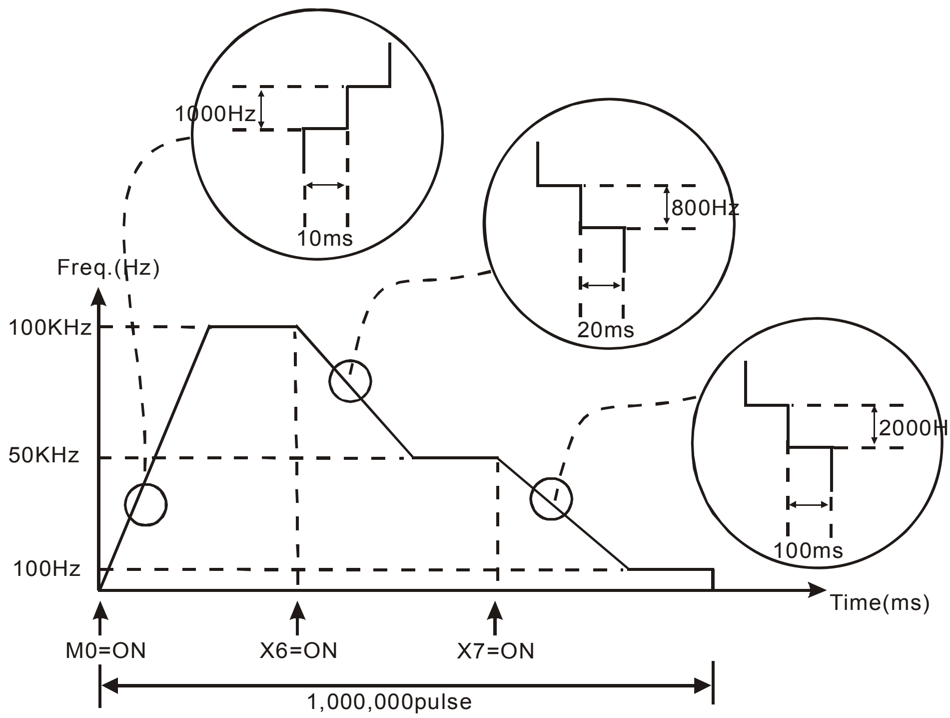
2. If users change the target frequency by applying DICF instruction in insterupt subroutines, the actual changing timing will be executed immediately with only an approx. 10us delay (execution time of DICF instruction).

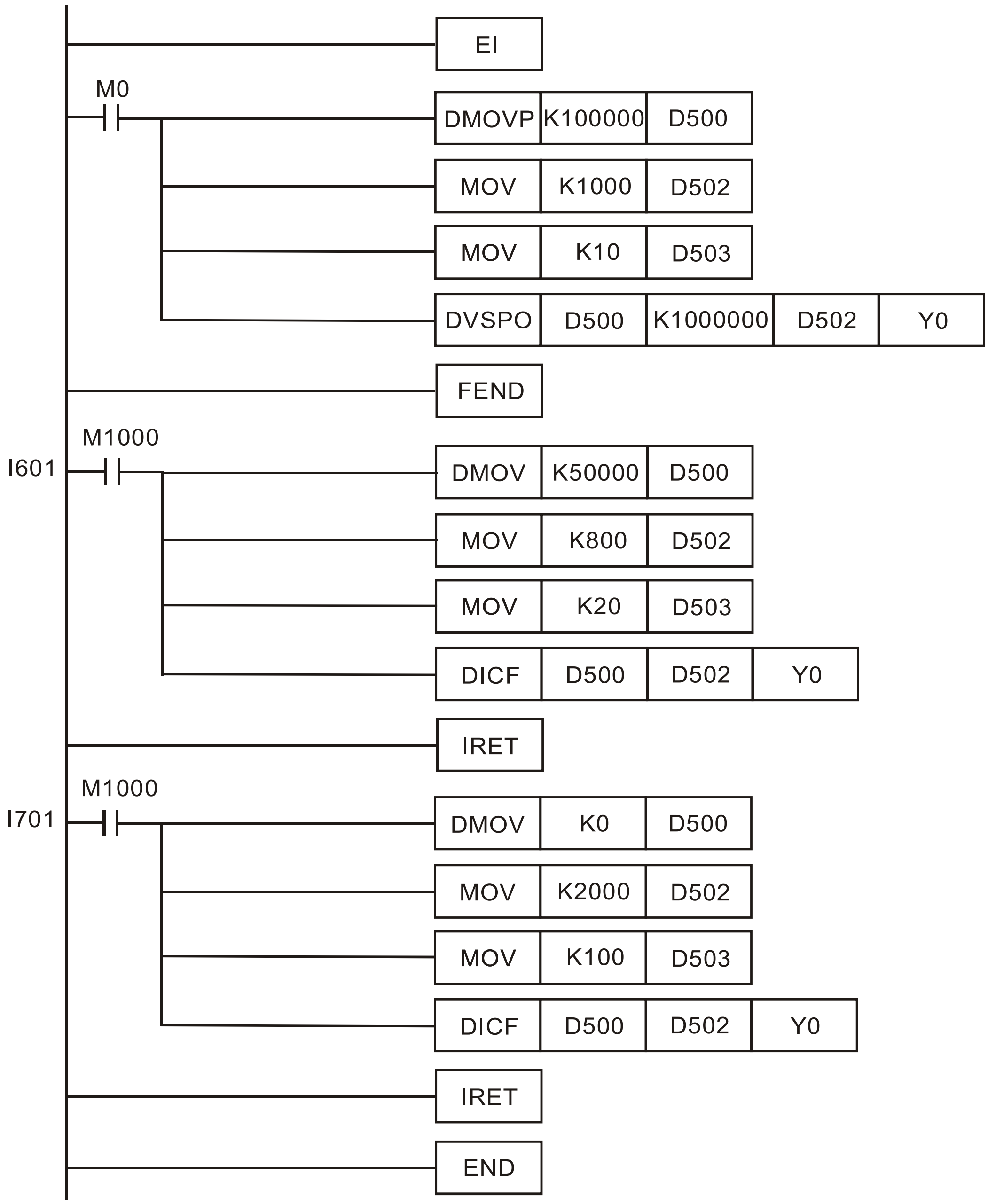
The timing diagram is as below:



Program Example:

1. When M0 = ON, pulse output ramps up to 100kHz. Total shifts: 100, Gap frequency: 1000Hz, Gap time: 10ms. Calculation of total shifts: (100,000 – 0) ÷ 1000 = 100.
2. When X6 external interrupt executes, target frequency is changed and ramp down to 50kHz immediately. Total shifts: 150, Gap frequency: 800Hz, Gap time: 20ms. Calculation of total shifts: (100,000 – 50,000) ÷ 800 = 125
3. When X7 external interrupt executes, target frequency is changed and ramp down to 100Hz immediately. Total shifts: 25, Gap frequency: 2000Hz, Gap time: 100ms. Calculation of total shifts: (50,000 – 100) ÷ 2000 = 25.
4. When pulse output reaches 100Hz, the frequency is kept constant and pulse output stops when 1,000,000 pulses is completed.




EI
M0
DMOVP K100000 D500
MOV K1000 D502
MOV K10 D503
DVSPO D500 K1000000 D502 Y0
FEND
M1000
I601
DMOV K50000 D500
MOV K800 D502
MOV K20 D503
DICF D500 D502 Y0
IRET
M1000
I701
DMOV K0 D500
MOV K2000 D502
MOV K100 D503
DICF D500 D502 Y0
IRET
END

API	Mnemonic		Operands	Function	Controllers
202	SCAL	P	S1 S2 S3 D	Proportional calculation	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*							*			SCAL,SCLAP: 9 steps
S_2					*	*							*			
S_3					*	*							*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source value **S_2**: Slope (unit: 0.001) **S_3**: Offset **D**: Operation result

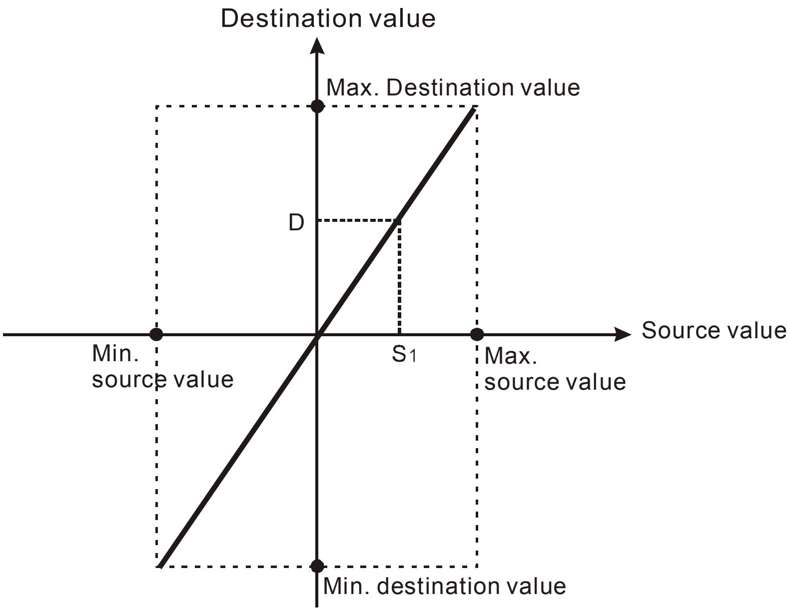
Range of operands **S_1, S_2, S_3**: -32768~32767.

Explanations:

1. SCAL instruction performs a proportional calculation according to the internal slope equation.
2. Operation equation in the instruction: **D** = (**S_1** × **S_2**) ÷ 1000 + **S_3**
3. Users have to obtain **S_2** and **S_3** (decimals are rounded up into 16-bit integers) by using the slope and offset equations below.

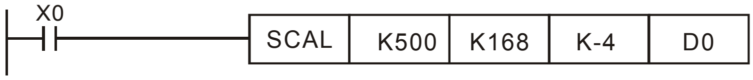
   Slope equation: **S_2** = [(max. destination value – min. destination value) ÷ (max. source value – min. source value)] × 1,000

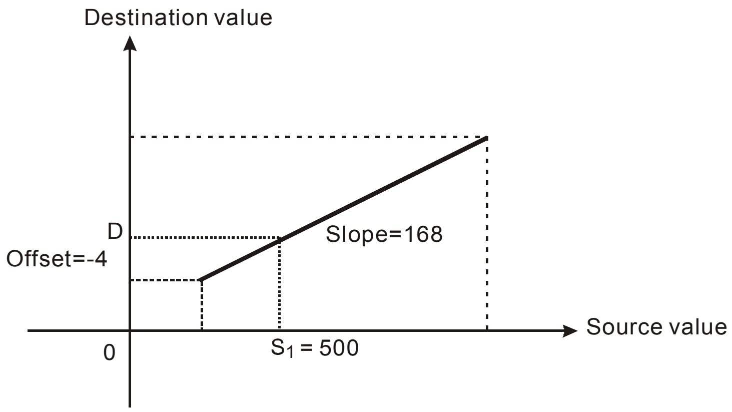
   Offset equation: **S_3** = min. destination value – min. source value × **S_2** ÷ 1,000
4. The output curve is shown as the figure:



Program Example 1:

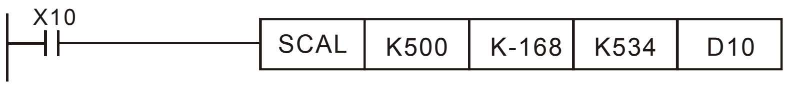
1. Assume $\mathbf{S_1}$ = 500, $\mathbf{S_2}$ = 168 and $\mathbf{S_3}$ = -4. When X0 = ON, SCAL instruction executes and the result of proportional calculation will be stored in D0.
2. Equation: D0 = (500 × 168) ÷ 1000 + (-4) = 80

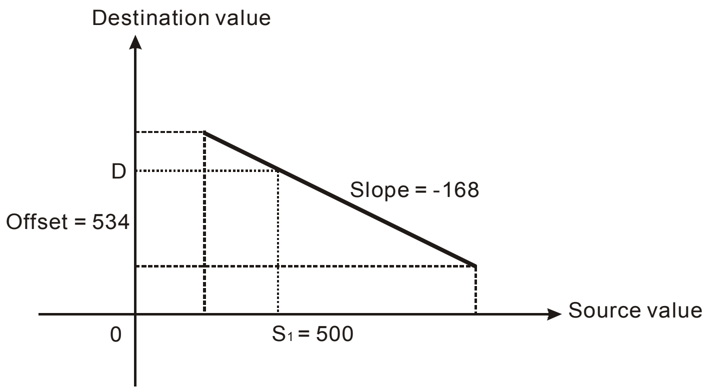




Program Example 2:

1. Assume $\mathbf{S_1}$ = 500, $\mathbf{S_2}$ = -168 and $\mathbf{S_3}$ = 534. When X0 = ON, SCAL instruction executes and the result of proportional calculation will be stored in D10..
2. Equation: D10 = (500 × -168) ÷ 1000+ 534 = 450





Points to note:

1. This instruction is applicable for known slope and offset. If slope and offset are unknown, please use SCLP instruction for the calculation.
2. $\mathbf{S_2}$ has to be within the range -32,768 ~ 32,767. If $\mathbf{S_2}$ exceeds the applicable range, use SCLP instruction instead.
3. When adopting the slope equation, the max source value must be larger than min source value, but the max destination value does not need to be larger than min destination value.
4. If **D** > 32,767, **D** will be set as 32,767. If **D** < -32,768, **D** will be set as -32,768.

API	Mnemonic			Operands	Function	Controllers
203	D	SCLP	P	S_1 S_2 D	Parameter proportional calculation	ES2/EX2 | SS2 | SA2 | SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
S_1					*	*							*			SCLP, SCLPP: 7 steps
S_2													*			DSCLP, DSCLPP: 13
D													*			steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source value **S_2**: Parameters **D**: Operation result

Explanations:

1. SCLP instruction performs a proportional calculation according to the internal slope equation as well as the parameters set in this instruction.
2. Settings of **S_2** for 16-bit instruction (occupies 4 consecutive devices):

Device No.	Parameter	Range
S_2	Max. source value	-32768~32767
S_2+1	Min. source value	-32768~32767
S_2+2	Max. destination value	-32768~32767
S_2+3	Min. destination value	-32768~32767

3. Settings of **S_2** for 32-bit instruction (occupies 8 consecutive devices).

Device No.	Parameter	Range	
		Integer	Floating point number
S_2、**S_2**+1	Max. source value	-2,147,483,648~2,147,483,647	Range of 32-bit floating point number
S_2+2、3	Min. source value		
S_2+4、5	Max. destination value		
S_2+6、7	Min. destination value		

4. Operation equation in the instruction: **D** = [(**S_1** – min. source value) × (max. destination value – min. destination value)] ÷ (max. source value – min. source value) + min. destination value
5. The equation to obtain the operation equation of the instruction:

 y = kx + b

 where

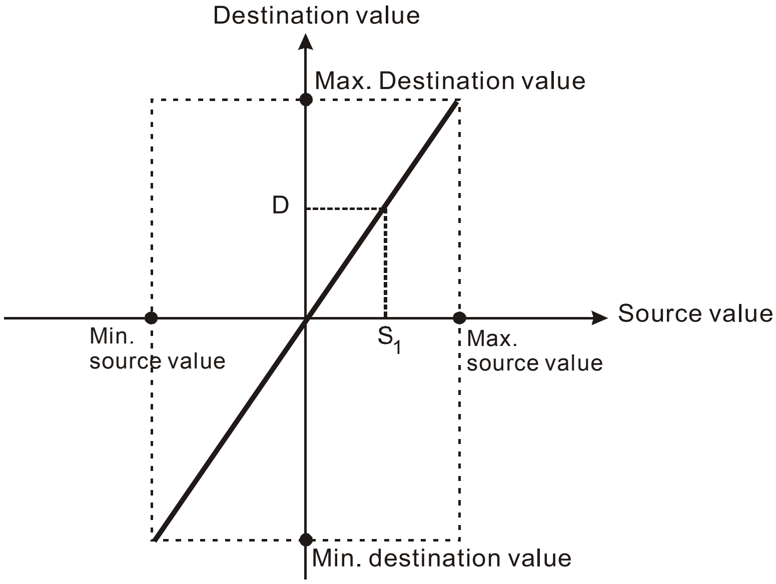
 y = Destination value (**D**)

 k = Slope = (max. destination value – min. destination value) ÷ (max. source value – min. source value)

 x = Source value (**S_1**)

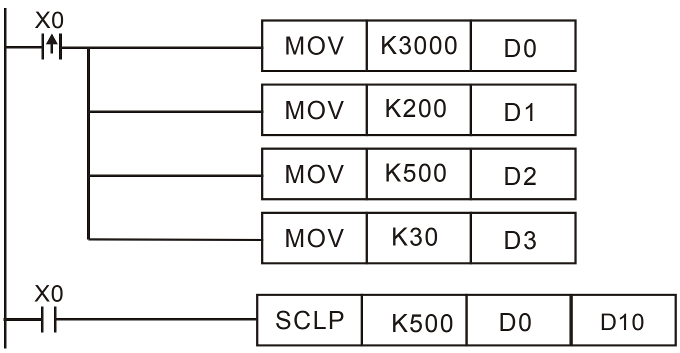
 b = Offset = Min. destination value – Min. source value × slope

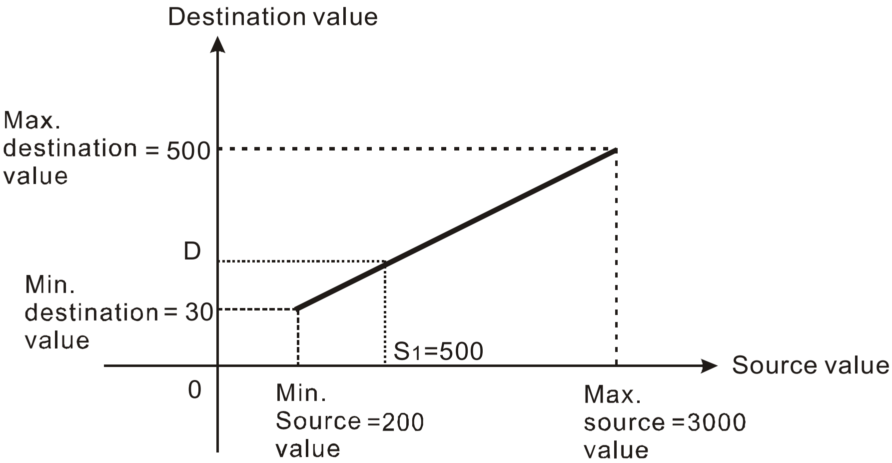
6. Substitute the above parameters into **y = kx + b** and the operation instruction can be obtained. y = kx + b = D = k $\mathbf{S_1}$ + b = slope × $\mathbf{S_1}$ + offset = slope × $\mathbf{S_1}$ + min. destination value – min. source value × slope = slope × ($\mathbf{S_1}$ – min. source value) + min. destination value = ($\mathbf{S_1}$ – min. source value) × (max. destination value – min. destination value) ÷ (max. source value – min. source value) + min. destination value
7. If $\mathbf{S_1}$ > max. source value, $\mathbf{S_1}$ will be set as max. source value. If $\mathbf{S_1}$ < min. source value, $\mathbf{S_1}$ will be set as min. source value. When the source value and parameters are set, the following output figure can be obtained:



Program Example 1:

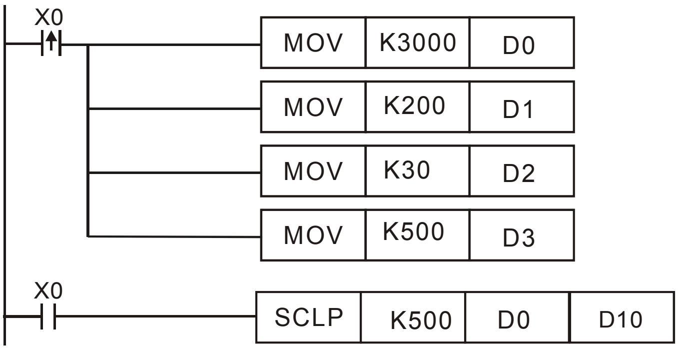
1. Assume source value $\mathbf{S_1}$ = 500, max. source value D0 = 3000, min. source value D1 = 200, max. destination value D2 = 500, and min. destination value D3 = 30. When X0 = ON, SCLP instruction executes and the result of proportional calculation will be stored in D10.
2. Equation: D10 = [(500 – 200) × (500 – 30)] ÷ (3000 – 200) + 30 = 80.35. Rounding off the result into an integer, D10 =80.

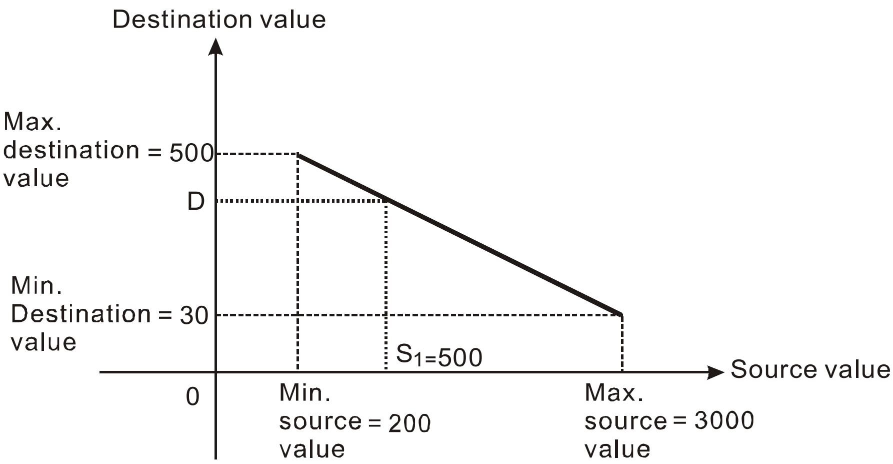




Program Example 2:

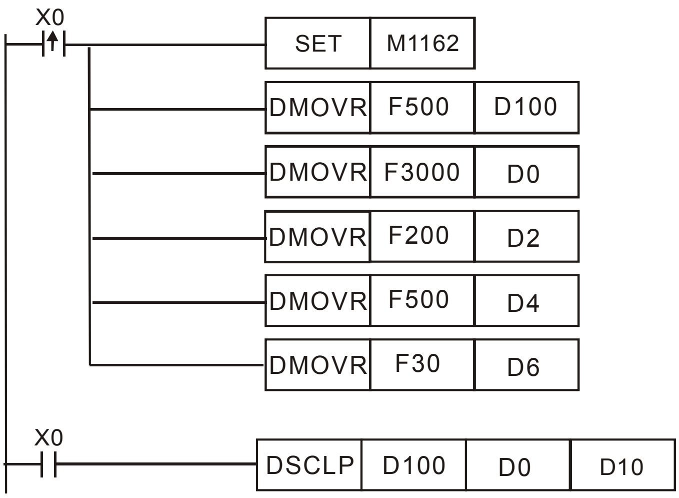
1. Assume source value S_1 = 500, max. source value D0 = 3000, min. source value D1 = 200, max. destination value D2 = 30, and min. destination value D3 = 500. When X0 = ON, SCLP instruction executes and the result of proportional calculation will be stored in D10.
2. Equation: D10 = [(500 – 200) × (30 – 500)] ÷(3000 – 200) + 500 = 449.64. Rounding off the result into an integer, D10 = 450.





Program Example 3:

1. Assume the source value $\mathbf{S_1}$, D100 = F500, max. source value D0 = F3000, min. source value D2 = F200, max. destination value D4 = F500, and min. destination value D6 = F30. When X0 = ON, M1162 is set up to adopt floating point operation. DSCLP instruction executes and the result of proportional calculation will be stored in D10.
2. Equation: D10 = [(F500 – F200) × (F500 – F30)] ÷ (F3000 – F200) + F30 = F80.35. Round off the result into an integer, D10 = F80.



Points to note:

1. Range of $\mathbf{S_1}$ for 16-bit instruction: max. source value ≥ $\mathbf{S_1}$ ≥ min. source value; -32,768 ~ 32,767. If the value exceeds the bounds, the bound value will be used for calculation.
2. Range of integer $\mathbf{S_1}$ for 32-bit instruction: max. source value ≥ $\mathbf{S_1}$ ≥ min. source value; -2,147,483,648 ~ 2,147,483,647. If the value exceeds the bounds, the bound value will be used for calculation.
3. Range of floating point $\mathbf{S_1}$ for 32-bit instruction: max. source value ≥ $\mathbf{S_1}$ ≥ min. source value; adopting the range of 32-bit floating point. If the value exceeds the bounds, the bound value will be used for calculation.
4. When adopting the slope equation, please note that the Max. source value must be larger than the min. source value. However the max. destination value does not need to be larger than the min. destination value.

API	Mnemonic		Operands	Function	Controllers
205	CMPT	P	(S1) (S2) (n) (D)	Compare table	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1											*	*	*			CMPT: 9 steps
S_2											*	*	*			CMPTP: 9 steps
n					*	*							*			
D								*	*	*	*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source device 1 **S_2**: Source device 2 **n**: Data length (**n** = 1~16) **D**: Destination device

Explanations:

1. **S_1** and **S_2** can be T/C/D devices, for C devices only 16-bit devices are applicable (C0~C199).
2. Range for operand **n**: 1~16. PLC will take the upper/lower bound value if set value exceeds the available range.
3. Data written in operand **D** will all be stored in 16-bit format. When data length is less than 16, the null bits are fixed as 0, e.g. if **n** = K8, bit 0~7 will be set according to compare results, and bit 8~15 will all be 0.

Program example:

When M0 = ON, compare the 16-bit value in D0~D7 with D20~D27 and store the results in D100.

M0					
─┤├─	CMPT	D0	D20	K8	D100

- Content in D0~D7:

No.	D0	D1	D2	D3	D4	D5	D6	D7
Value	K10	K20	K30	K40	K50	K60	K70	K80

- Content in D20~D27:

No.	D20	D21	D22	D23	D24	D25	D26	D27
Value	K12	K20	K33	K44	K50	K66	K70	K88

- After the comparison of CMPT instruction, the associated bit will be 1 if two devices have the same value, and other bits will all be 0. Therefore the results in D100 will be as below:

	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit8~15
D100	0	1	0	0	1	0	1	0	0…0
	H0052 (K82)								

API	Mnemonic	Operands	Function	Controllers
206	ASDRW	S_1 S_2 S	ASDA servo drive R/W	ES2/EX2 | SS2 | SA2 | SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											ASDRW: 7 steps
S_1					*	*							*			
S_2					*	*							*			
S													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Address of servo drive (K0~K254) **S_2**: Function code **S**: Register for read/written data

Explanations:

1. ASDRW communication instruction supports COM2 (RS-485) and COM3 (RS-485)
2. **S_1**: station number of servo drive. Range: K0~K254. K0 indicates broadcasting, i.e. PLC will not receive feedback data.
3. **S_2**: function code. Please refer to the table below.
4. **S**: Register for read/written data. Please refer to the table below for explanations.
5. Explanations of function code:

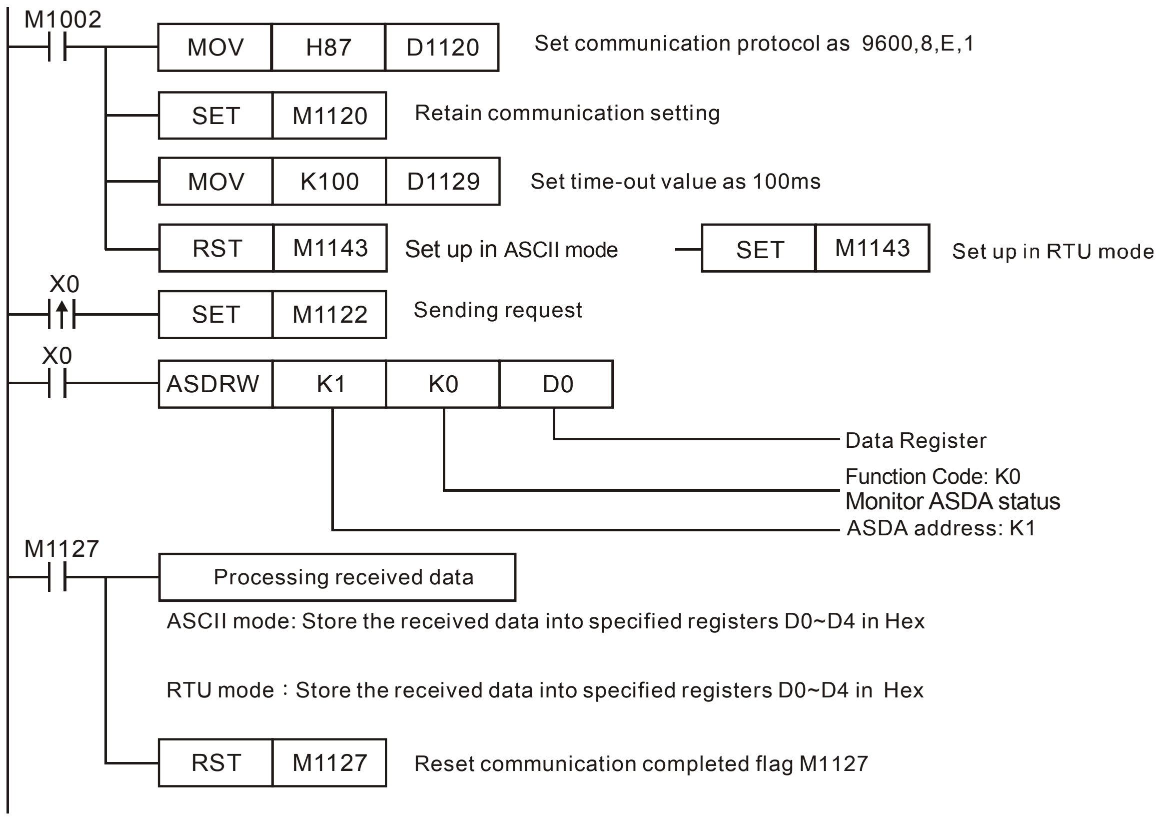
Exclusively for ASDA of A-type, AB type, A+ type, B type				
Code	Function	Parameter	Com. Addr.	Read/Write data (Settings)
K0(H0)	Status monitor	P0-04 ~ P0-08	0004H ~ 0008H	**S**+0 ~ **S**+4: Please refer to explanations in ASDA manuals.
K1(H1)	Block Data Read Register	P0-09 ~ P0-16	0009H ~ 0010H	**S**+0 ~ **S**+7: Please refer to explanations in ASDA manuals. B Type is not supported.
K2(H2)	Block Data Write Register	P0-09 ~ P0-16	0009H ~ 0010H	**S**+0 ~ **S**+7: Please refer to explanations in ASDA manuals. B Type is not supported.
K3(H3)	JOG Operation	P4-05	0405H	**S**: Range: 1~3000, 4999, 4998, 5000
K4(H4)	Servo ON/OFF	P2-30	021EH	**S**: K1 = ON, Others = OFF
K5(H5)	Speed Command (3 sets)	P1-09 ~ P1-11	0109H ~ 010BH	**S**+0 ~ **S**+2: Range: -5000~+5000
K6(H6)	Torque Command (3 sets)	P1-12 ~ P1-14	010CH ~ 010EH	**S**+0 ~ **S**+2: Range: -300~+300

For A2-type only				
Code	Function	Parameter	Com. Addr.	Read/Write data (Settings)
K16(H10)	Status monitor (Read)	P0-09 ~ P0-13	0012H ~ 001BH	**S**+0 ~ **S**+9: Please refer to explanations in ASDA-A2 manual.
K17(H11)	Status monitor selection (Write)	P0-17 ~ P0-21	0022H ~ 002BH	**S**+0 ~ **S**+9: Please refer to explanations in ASDA-A2 manual.
K18(H12)	Mapping parameter (Write)	P0-25 ~ P0-32	0032H ~ 0041H	**S**+0 ~ **S**+15: Please refer to explanations in ASDA-A2 manual.
K19(H13)	JOG Operation	P4-05	040AH	**S**: Range: 1~5000, 4999, 4998, 0
K20(H14)	Auxiliary Function (Servo ON/OFF)	P2-30	023CH	**S**: K1 = ON, Others = OFF
K21(H15)	Speed Command (3 sets)	P1-09 ~ P1-11	0112H ~ 0117H	**S**+0 ~ **S**+5: Range: -60000~+60000
K22(H16)	Torque Command (3 sets)	P1-12 ~ P1-14	0118H ~ 011DH	**S**+0 ~ **S**+5: Range: -300~+300
K23(H17)	Block Data Read / Write Register (for mapping parameter)	P0-35 ~ P0-42	0046H~ 0055H	**S**+0 ~ **S**+15: Please refer to explanations in ASDA-A2 manual.

6. For relative M flags and special D registers, please refer to explanations of API 80 RS instruction.

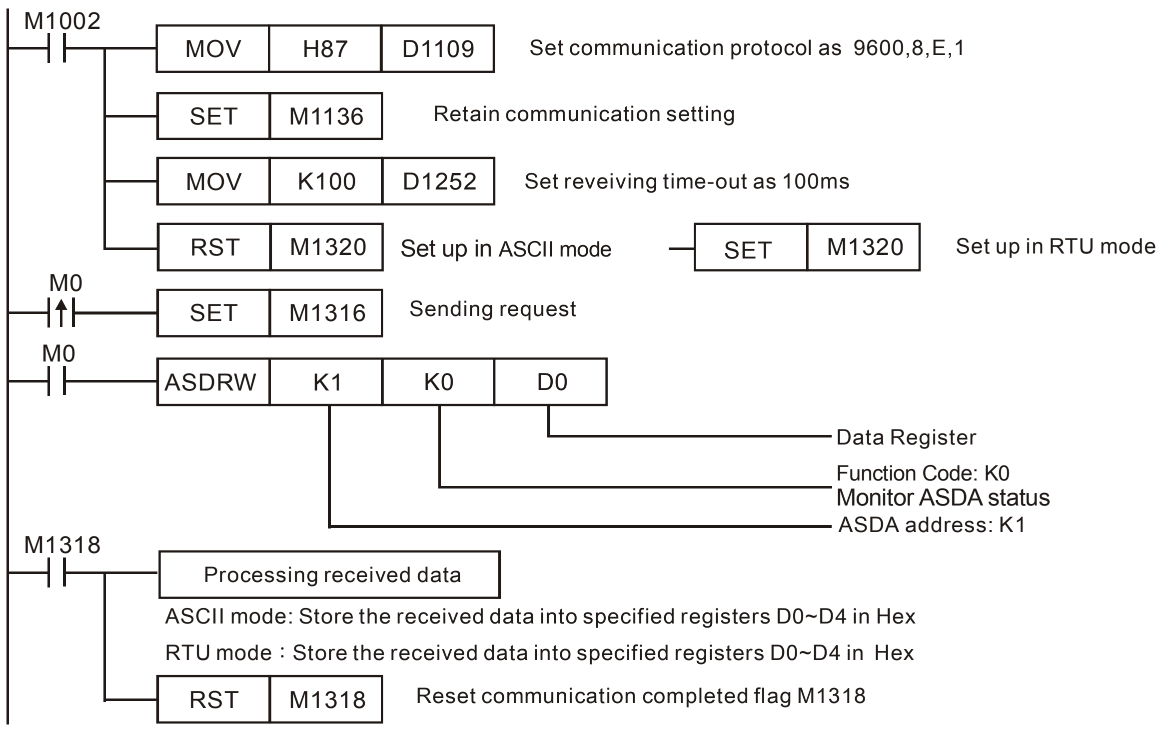
Program example 1: COM2 (RS-485)

1. When X0 = ON, PLC will send out communication commands by COM2 to read status of servo drive.
2. When PLC received the feedback data from ASDA, M1127 will be active and the read data will be stored in D0~D4.



Program example 2: COM3(RS-485)

1. When M0 = ON, PLC sends communication commands by COM3 to read servo drive status.
2. When PLC received the feedback data from ASDA, M1318 will be active and the read data will be stored in D0~D4.



Points to note:

Relative flags and special D registers of COM2/COM3 :

	COM2	COM3	Function Description
Protocol setting	M1120	M1136	Retain communication setting
	M1143	M1320	ASCII/RTU mode selection
	D1120	D1109	Communication protocol
	D1121	D1255	PLC communication address
Sending request	M1122	M1316	Sending request
	D1129	D1252	Communication timeout setting (ms)
Receiving completed	M1127	M1318	Data receiving completed
Errors	-	M1319	Data receiving error
	-	D1253	Communication error code
	M1129	-	Communication timeout setting (ms)
	M1140	-	COM2 (RS-485) MODRD/MODWR/MODRW data receiving error
	M1141	-	MODRD/MODWR/MODRW parameter error (Exception Code exists in received data) Exception Code is stored in D1130
	D1130	-	COM2 (RS-485) Error code (exception code) returning from Modbus communication

API	Mnemonic	Operands	Function	Controllers			
				ES2/EX2	SS2	SA2	SX2
207	CSFO	S S1 D	Catch speed and proportional output				

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	CSFO: 7 steps
S	*															
S_1													*			
D													*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S: Source device of signal input (Only X0~X3 are available) **S_1**: Sample time setting and the input speed information **D**: Output proportion setting and output speed information

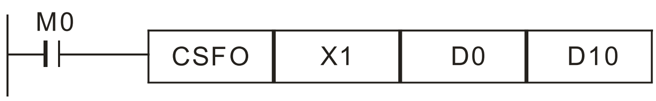
Explanations:

1. When **S** specifies X0, PLC only uses X0 input point and its associated high speed pulse output: Y0, in this case Y1 is normal output point. When **S** specifies X1, PLC uses X0 (A phase) and X1 (B phase) input points and their associated output: Y0 (Pulse) / Y1 (Dir). When **S** specifies X2, PLC only uses X2 input point and its associated high speed pulse output: Y2, in this case Y3 is normal output point. When **S** specifies X3, PLC uses X2 (A phase) and X3 (B phase) input points and their associated output: Y2 (Pulse) / Y3 (Dir).
2. The execution of CSFO requires hardware high speed counter function as well as the high speed output function. Therefore, when program scan proceeds to CSFO instruction with high speed counter input points (X0, X1) or (X2, X3) enabled by DCNT instruction, or high speed pulse outputs (Y0, Y1) or (Y2, Y3) enabled by other high speed output instructions, CSFO instruction will not be activated.
3. If **S** specifies X1 / X3 with 2-phase 2 inputs, the counting mode is fixed as quadruple frequency.
4. During pulse output process of Y0 or Y2, special registers (D1031, D1330 / D1337, D1336) storing the current number of output pulses will be updated when program scan proceeds to this instruction.
5. **S_1** occupies consecutive 4 16-bit registers. **S_1** +0 specifies the sampling times, i.e. when **S_1** +0 specifies K1, PLC catches the speed every time when 1 pulse is outputted. Valid range for **S_1** +0 in 1-phase 1-input mode: K1~K100, and 2-phase 2-input mode: K2~K100. If the specified value exceeds the valid range, PLC will take the lower/upper bound value as the set value. Sample time can be changed during PLC operation, however the modified value will take effect until program scan proceeds to this instruction. **S_1**+1 indicates the latest speed sampled by PLC (Read-only). Unit: 1Hz. Valid range: ±10kHz. **S_1**+2 and **S_1**+3 indicate the accumulated number of pulses in 32-bit data (Read-only).

6. S_1 +0 specifies the sampling times. The set value of sampling times is recommended to be bigger when the input speed increases, so as to achieve a higher accuracy for speed catching. For example, set S_1 +0 as K1 for the speed range 1Hz~1KHz, K10 for the speed range 10Hz~10KHz, K100 for the speed range 100Hz~10KHz. For single phase input, the max frequency is 10kHz; for 2-phase 2 inputs, the max frequency is 2kHz.
7. **D** occupies 3 consecutive 16-bit registers. **D** +0 specifies the output proportion value. Valid range: K1 (1%) ~ K10000 (10000%). If the specified value exceeds the valid range, PLC will take the lower/upper bound value as the set value. Output proportion can be changed during PLC operation, however the modified value will take effect until program scan proceeds to this instruction. **D**+2 and **D**+1 indicates the output speed in 32-bit data. Unit: 1Hz. Valid range: ±100kHz.
8. The speed sampled by PLC will be multiplied with the output proportion **D**+0, then PLC will generate the actual output speed. PLC will take the integer of the calculated value, i.e. if the calculated result is smaller than 1Hz, PLC will output with 0Hz. For example, input speed: 10Hz, output proportion: K5 (5%), then the calculation result will be 10 x 0.05 = 0.5Hz. Pulse output will be 0Hz; if output proportion is modified as K15 (15%), then the calculation result will be 10 x 0.15 = 1.5Hz. Pulse output will be 1Hz.

Program Example:

1. If D0 is set as K2, D10 is set as K100:
   When the sampled speed on (X0, X1) is +10Hz (D1 = K10), (Y0, Y1) will output pulses with +10Hz (D12, D11 = K10); When the sampled speed is -10Hz (D1 = K-10), (Y0, Y1) will output pulses with -10Hz (D12, D11 = K-10)
2. If D0 is set as K2, D10 is set as K1000:
   When the sampled speed on (X0, X1) is +10Hz (D1 = K10), (Y0, Y1) will output pulses with +100Hz (D12, D11 = K100); When the sampled speed is -100Hz (D1 = K-100), (Y0, Y1) will output pulses with -100Hz (D12, D11 = K-100)
3. If D0 is set as K10, D10 is set as K10:
   When the sampled speed on (X0, X1) is +10Hz (D1 = K10), (Y0, Y1) will output pulses with +1Hz (D12, D11 = K1); When the sampled speed is -10Hz (D1 = K-10), (Y0, Y1) will output pulses with -1Hz (D12, D11 = K-1)



API	Mnemonic		Operands	Function	Controllers
215~217	D	LD#	S_1 S_2	Contact Type Logic Operation	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	LD#: 5 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DLD#: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source device 1 S_2: Source device 2

Explanations:

1. This instruction conducts logic operation between the content in S_1 and S_2. If the result is not "0", the continuity of the instruction is enabled. If the result is "0", the continuity of the instruction is disabled.
2. LD# (#: &, |, ^) instruction is used for direct connection with Left bus bar.

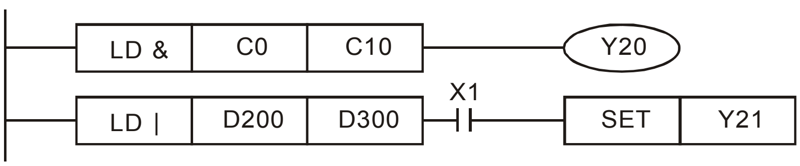
API No.	16 -bit instruction	32 -bit instruction	Continuity condition	Discontinuity condition
215	LD&	DLD&	S_1 & $S_2 \neq 0$	S_1 & $S_2 = 0$
216	LD|	DLD|	S_1 | $S_2 \neq 0$	S_1 | $S_2 = 0$
217	LD^	DLD^	S_1 ^ $S_2 \neq 0$	S_1 ^ $S_2 = 0$

3. Operation:

   & : Logic "AND" operation, | : Logic "OR" operation, ^ : Logic "XOR" operation
4. When 32-bit counters (C200 ~ C254) are used in this instruction, make sure to adopt 32-bit instruction (DLD#). If 16-bit instruction (LD#) is adopted, a "program error" will occur and the ERROR indicator on the MPU panel will flash.

Program Example:

1. When the result of logical AND operation between C0 and C10 ≠ 0, Y20 = ON.
2. When the result of logical OR operation between D200 and D300 ≠ 0 and X1 = ON, Y21 = ON and latched.



API		Mnemonic		Operands	Function	Controllers
218~ 220	D	AND#		(S1) (S2)	Serial Type Logic Operation	ES2/EX2 | SS2 | SA2 | SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	AND#: 5 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DAND#: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

$\mathbf{S_1}$: Source device 1 $\mathbf{S_2}$: Source device 2

Explanation:

1. This instruction conducts logic operation between the content in $\mathbf{S_1}$ and $\mathbf{S_2}$. If the result is not "0", the continuity of the instruction is enabled. If the result is "0", the continuity of the instruction is disabled.
2. AND# (#: &, |, ^) instruction is used for serial connection with contacts.

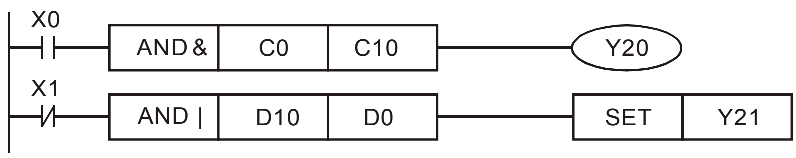
API No.	16 -bit instruction	32 -bit instruction	Continuity condition	Discontinuity condition
218	AND&	**D**AND&	$\mathbf{S_1}$ & $\mathbf{S_2} \neq 0$	$\mathbf{S_1}$ & $\mathbf{S_2} = 0$
219	AND|	**D**AND|	$\mathbf{S_1} \mid \mathbf{S_2} \neq 0$	$\mathbf{S_1} \mid \mathbf{S_2} = 0$
220	AND^	**D**AND^	$\mathbf{S_1}$ ^ $\mathbf{S_2} \neq 0$	$\mathbf{S_1}$ ^ $\mathbf{S_2} = 0$

3. Operation:

   & : Logic "AND" operation, | : Logic "OR" operation, ^ : Logic "XOR" operation
4. When 32-bit counters (C200 ~ C254) are used in this instruction, make sure to adopt 32-bit instruction (DAND#). If 16-bit instruction (AND#) is adopted, a "program error" will occur and the ERROR indicator on the MPU panel will flash

Program Example:

1. When X0 = ON, and the result of logical AND operation between C0 and C10 ≠ 0, Y20 = ON
2. When X1 = OFF, and the result of logical OR operation between D10 and D0 ≠ 0, Y21 = ON and latched



API		Mnemonic		Operands	Function	Controllers
221~223	D	OR#		S1 S2	Parallel Type Logic Operation	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
S_1					*	*	*	*	*	*	*	*	*	*	*	OR#: 5 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DOR#: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source device 1 **S_2**: Source device 2

Explanation:

1. This instruction conducts logic operation between the content in **S_1** and **S_2**. If the result is not "0", the continuity of the instruction is enabled. If the result is "0", the continuity of the instruction is disabled.
2. OR# (#**:** &, |, ^) instruction is used for parallel connection with contacts.

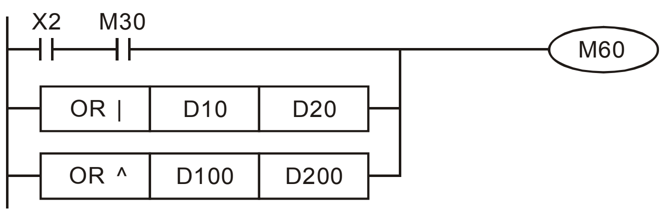
API No.	16 -bit instruction	32 -bit instruction	Continuity condition	Discontinuity condition
221	OR&	DOR&	S_1 & $S_2 \neq 0$	S_1 & $S_2 = 0$
222	OR|	DOR|	S_1 | $S_2 \neq 0$	S_1 | $S_2 = 0$
223	OR^	DOR^	S_1 ^ $S_2 \neq 0$	S_1 ^ $S_2 = 0$

3. Operation:

   & : Logic "AND" operation, | : Logic "OR" operation, ^ : Logic "XOR" operation
4. When 32-bit counters (C200 ~ C254) are used in this instruction, make sure to adopt 32-bit instruction (DOR#). If 16-bit instruction (OR#) is adopted, a "program error" will occur and the ERROR indicator on the MPU panel will flash

Program Example:

M60 will be ON either when both X2 and M30 are "ON", or 1: the result of logical OR operation between D10 and D20 ≠ 0, or 2: the result of logical XOR operation between CD100 and D200 ≠ 0.



API	Mnemonic		Operands	Function	Controllers
224~ 230	D	LD※	S_1 S_2	Contact Type Comparison	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	LD※: 5 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DLD※: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source device 1   **S_2**: Source device 2

Explanations:

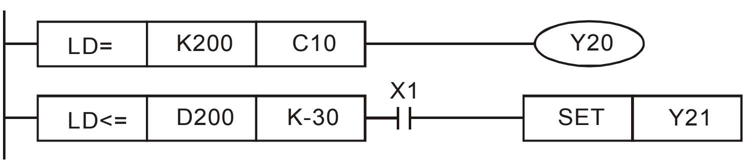
1. This instruction compares the content in **S_1** and **S_2**. Take API224 (LD=) for example, if the result is "=", the continuity of the instruction is enabled. If the result is "≠", the continuity of the instruction is disabled.
2. LD※ (※: =, >, <, <>, ≤, ≥) instruction is used for direct connection with left hand bus bar.

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**

3. When the MSB (16-bit instruction: b15, 32-bit instruction: b31) of **S_1** and **S_2** is 1, the comparison value will be viewed as a negative value in comparison.
4. When 32-bit counters (C200 ~ C254) are used in this instruction, make sure to adopt 32-bit instruction (DLD※). If 16-bit instruction (LD※) is adopted, a "program error" will occur and the ERROR indicator on the MPU panel will flash.

Program Example:

1. When the content in C10 = K200, Y20 = ON.
2. When the content in D200 > K-30 and X1 = ON, Y21 = ON and latched.



API	Mnemonic		Operands	Function	Controllers
232~238	D	AND※	S1 S2	Serial Type Comparison	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
	Bit Devices				Word devices											
S_1					*	*	*	*	*	*	*	*	*	*	*	AND※: 5 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DAND※: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source device 1 S_2: Source device 2

Explanations:

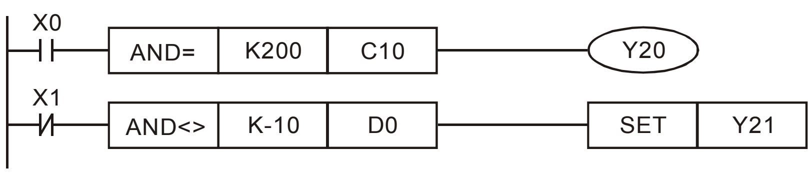
1. This instruction compares the content in S_1 and S_2. Take API232 (AND =) for example, if the result is "=", the continuity of the instruction is enabled. If the result is "≠", the continuity of the instruction is disabled.
2. AND※ (※: =, >, <, <>, ≤, ≥) instruction is used for serial connection with contacts.

API No.	16 -bit instruction	32 -bit instruction	Continuity condition	Discontinuity condition
232	AND＝	DAND＝	$S_1 = S_2$	$S_1 \neq S_2$
233	AND＞	DAND＞	$S_1 > S_2$	$S_1 \leqq S_2$
234	AND＜	DAND＜	$S_1 < S_2$	$S_1 \geqq S_2$
236	AND＜＞	DAND＜＞	$S_1 \neq S_2$	$S_1 = S_2$
237	AND＜＝	DAND＜＝	$S_1 \leqq S_2$	$S_1 > S_2$
238	AND＞＝	DAND＞＝	$S_1 \geqq S_2$	$S_1 < S_2$

3. When the MSB (16-bit instruction: b15, 32-bit instruction: b31) of S_1 and S_2 is 1, the comparison value will be viewed as a negative value in comparison.
4. When 32-bit counters (C200 ~ C254) are used in this instruction, make sure to adopt 32-bit instruction (DAND※). If 16-bit instruction (AND※) is adopted, a "program error" will occur and the ERROR indicator on the MPU panel will flash.

Program Example:

1. When X0 = ON, and the content in C10 = K200, Y20 = ON
2. When X1 = OFF and the content in D0 ≠ K-10, Y21= ON and latched.



API		Mnemonic		Operands	Function	Controllers
240~246	D	OR※		(S1) (S2)	Parallel Type Comparison	ES2/EX2 / SS2 / SA2 / SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1					*	*	*	*	*	*	*	*	*	*	*	OR※: 5 steps
S_2					*	*	*	*	*	*	*	*	*	*	*	DOR※: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source device 1 **S_2**: Source device 2

Explanations:

1. This instruction compares the content in **S_1** and **S_2**. Take API240 (OR =) for example, if the result is "=", the continuity of the instruction is enabled. If the result is "≠", the continuity of the instruction is disabled

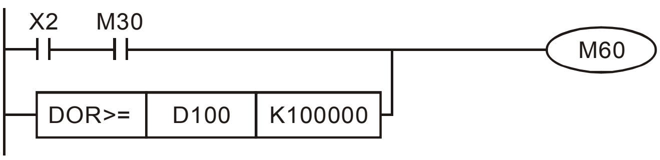
2. OR※ (※: =, >, <, <>, ≤, ≥) instruction is used for parallel connection with contacts.

API No.	16-bit instruction	32-bit instruction	Continuity condition	Discontinuity condition
240	OR＝	DOR＝	$S_1 = S_2$	$S_1 \neq S_2$
241	OR＞	DOR＞	$S_1 > S_2$	$S_1 \leqq S_2$
242	OR＜	DOR＜	$S_1 < S_2$	$S_1 \geqq S_2$
244	OR＜＞	DOR＜＞	$S_1 \neq S_2$	$S_1 = S_2$
245	OR＜＝	DOR＜＝	$S_1 \leqq S_2$	$S_1 > S_2$
246	OR＞＝	DOR＞＝	$S_1 \geqq S_2$	$S_1 < S_2$

3. When the MSB (16-bit instruction: b15, 32-bit instruction: b31) of **S_1** and **S_2** is 1, the comparison value will be viewed as a negative value in comparison..
4. When 32-bit counters (C200 ~ C254) are used in this instruction, make sure to adopt 32-bit instruction (DOR※). If 16-bit instruction (OR※) is adopted, a "program error" will occur and the ERROR indicator on the MPU panel will flash

Program Example:

M60 will be ON either when both X2 and M30 are "ON", or when the content in 32-bit register D100 (D101) ≥ K100,000.



API	Mnemonic		Operands	Function	Controllers
266	D	BOUT	(D) (n)	Output Specified Bit of a Word	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
D								*	*	*	*	*	*			BOUT: 5 steps
n					*	*	*	*	*	*	*	*	*	*	*	DBOUT: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

D: Destination output device   **n**: Device specifying the output bit

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function.
2. Available range for the value in operand **n**: K0~K15 for 16-bit instruction; K0~K31 for 32-bit instruction.
3. BOUT instruction performs bit output on the output device according to the value specified by operand **n.**

Status of Coils and Associated Contacts:

Evaluation result	BOUT instruction		
	Coil	Associated Contacts	
		NO contact（normally open）	NC contact（normally closed）
FALSE	OFF	Current blocked	Current flows
TRUE	ON	Current flows	Current blocked

Program Example:

Instruction:		Operation:
LDI	X0	Load NC contact X0
AND	X1	Connect NO contact X1 in series.
BOUT	**K4Y0 D0**	**When D0 = k1, executes output on Y1** **When D0 = k2, executes output on Y2**

API	Mnemonic		Operands	Function	Controllers
267	D	BSET	(D) (n)	Set ON Specified Bit of a Word	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	BSET: 5 steps
D								*	*	*	*	*	*			DBSET: 9 steps
n					*	*	*	*	*	*	*	*	*	*	*	

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

D: Destination device to be Set ON **n**: Device specifying the bit to be Set ON

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function.
2. Available range for the value in operand **n**: K0~K15 for 16-bit instruction; K0~K31 for 32-bit instruction.
3. When BSET instruction executes, the output device specified by operand **n** will be ON and latched. To reset the ON state of the device, BRST instruction is required.

Program Example:

Instruction:			Operation:
LDI	X0		Load NC contact X0
AND	X1		Connect NO contact X1 in series.
BSET	**K4Y0**	**D0**	**When D0 = k1, Y1 is ON and latched**
			When D0 = k2, Y2 = ON and latched

API	Mnemonic		Operands	Function	Controllers
268	D	BRST	D n	Reset Specified Bit of a Word	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
D								*	*	*	*	*	*			BRST: 5 steps
n					*	*	*	*	*	*	*	*	*	*	*	DBRST: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

D: Destination device to be reset **n:** Device specifying the bit to be reset

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function.
2. Available range for the value in operand **n**: K0~K15 for 16-bit instruction; K0~K31 for 32-bit instruction.
3. When BRST instruction executes, the output device specified by operand **n** will be reset (OFF).

Program Example:

Instruction:			Operation:
LD	X0		Load NO contact X0
BRST	**K4Y0**	**D0**	**When D0 = k1,**
			Y1 is OFF
			When D0 = k2,
			Y2 = OFF

API	Mnemonic		Operands	Function	Controllers
269	D	BLD	S n	Load NO Contact by Specified Bit	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S								*	*	*	*	*	*			BLD: 5 steps
n					*	*	*	*	*	*	*	*	*	*	*	DBLD: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

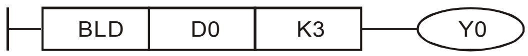
Operands:

S: Reference source device **n**: Reference bit

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function.
2. Available range for the value in operand **n**: K0~K15 for 16-bit instruction; K0~K31 for 32-bit instruction.
3. BLD instruction is used to load NO contact whose contact state is defined by the reference bit **n** in reference device **D**, i.e. if the bit specified by **n** is ON, the NO contact will be ON, and vice versa.

Program Example:



Instruction:			Operation:
BLD	**D0**	**K3**	**Load NO contact with bit status of bit3 in D0**
OUT	Y0		Drive coil Y0

API	Mnemonic		Operands	Function	Controllers			
					ES2/EX2	SS2	SA2	SX2
270	D	BLDI	(S) (n)	Load NC Contact by Specified Bit				

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S								*	*	*	*	*	*			BLDI: 5 steps
n					*	*	*	*	*	*	*	*	*	*	*	DBLDI: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

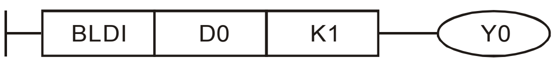
Operands:

S: Reference source device **n**: Reference bit

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function.
2. Available range for the value in operand **n**: K0~K15 for 16-bit instruction; K0~K31 for 32-bit instruction.
3. BLD instruction is used to load NC contact whose contact state is defined by the reference bit **n** in reference device **D**, i.e. if the bit specified by **n** is ON, the NC contact will be ON, and vice versa.

Program Example:



Instruction:			Operation:
BLDI	**D0**	**K1**	**Load NC contact with bit status of bit1 in D0**
OUT	Y0		Drive coil Y0

API	Mnemonic		Operands	Function	Controllers
271	D	BAND	(S) (n)	Connect NO Contact in Series by Specified Bit	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S								*	*	*	*	*	*			BAND: 5 steps
n					*	*	*	*	*	*	*	*	*	*	*	DBAND: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

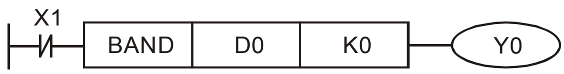
Operands:

S: Reference source device **n**: Reference bit

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function.
2. Available range for the value in operand **n**: K0~K15 for 16-bit instruction; K0~K31 for 32-bit instruction.
3. BAND instruction is used to connect NO contact in series, whose state is defined by the reference bit **n** in reference device **D**, i.e. if the bit specified by **n** is ON, the NO contact will be ON, and vice versa.

Program Example:



Instruction:			Operation:
LDI	X1		Load NC contact X1
BAND	**D0**	**K0**	**Connect NO contact in series , whose state is defined by bit0 of D0**
OUT	Y0		Drive coil Y0

API	Mnemonic		Operands	Function	Controllers			
					ES2/EX2	SS2	SA2	SX2
272	D	BANI	(S) (n)	Connect NC Contact in Series by Specified Bit				

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	
S								*	*	*	*	*	*			BANI: 5 steps
n					*	*	*	*	*	*	*	*	*	*	*	DBANI: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

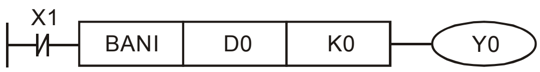
Operands:

S: Reference source device **n**: Reference bit

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function
2. Available range for the value in operand **n**: K0~K15 for 16-bit instruction; K0~K31 for 32-bit instruction.
3. BANI instruction is used to connect NC contact in series, whose state is defined by the reference bit **n** in reference device **D**, i.e. if the bit specified by **n** is ON, the NC contact will be ON, and vice versa.

Program Example:



Instruction:			Operation:
LDI	X1		Load NC contact X1
BANI	**D0**	**K0**	**Connect NC contact in series , whose state is defined by bit0 of D0**
OUT	Y0		Drive coil Y0

API	Mnemonic		Operands	Function	Controllers
273	D	BOR	(S) (n)	Connect NO Contact in Parallel by Specified Bit	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S								*	*	*	*	*	*			BOR: 5 steps
n					*	*	*	*	*	*	*	*	*	*	*	DBOR: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

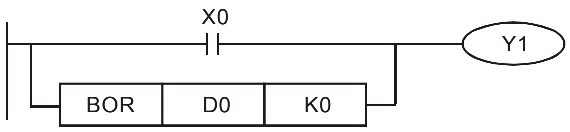
Operands:

S: Reference source device **n:** Reference bit

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function.
2. Available range for the value in operand **n**: K0~K15 for 16-bit instruction; K0~K31 for 32-bit instruction.
3. BOR instruction is used to connect NO contact in parallel, whose state is defined by the reference bit **n** in reference device **D**, i.e. if the bit specified by **n** is ON, the NO contact will be ON, and vice versa.

Program Example:



Instruction:			Operation:
LD	X0		Load NO contact X0
BOR	**D0**	**K0**	**Connect NO contact in parallel, whose state is defined by bit0 of D0**
OUT	Y1		Drive coil Y1

API	Mnemonic		Operands	Function	Controllers
274	D	BORI	(S) (n)	Connect NC Contact in Parallel by Specified Bit	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S								*	*	*	*	*	*			BORI: 5 steps
n					*	*	*	*	*	*	*	*	*	*	*	DBORI: 9 steps

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

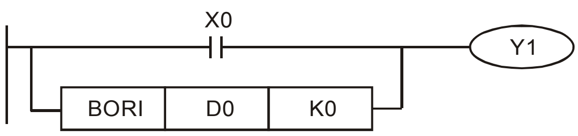
Operands:

S: Reference source device **n:** Reference bit

Explanations:

1. For ES2/EX2 models, only V1.20 or above supports the function
2. Available range for the value in operand **n**: K0~K15 for 16-bit instruction; K0~K31 for 32-bit instruction.
3. BORI instruction is used to connect NC contact in parallel, whose state is defined by the reference bit **n** in reference device **D**, i.e. if the bit specified by **n** is ON, the NC contact will be ON, and vice versa.

Program Example:



Instruction:			Operation:
LD	X0		Load NO contact X0
BORI	**D0**	**K0**	**Connect NC contact in parallel, whose state is defined by bit0 of D0**
OUT	Y1		Drive coil Y1

API	Mnemonic	Operands	Function	Controllers
275~ 280	FLD※	S1 S2	Floating Point Contact Type Comparison LD※	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1											*	*	*			FLD※: 9 steps
S_2											*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source device 1 **S_2**: Source device 2

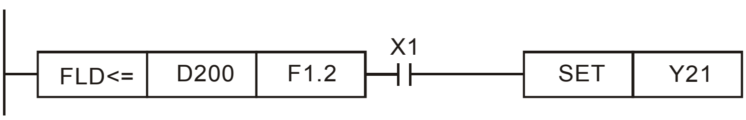
Explanations:

1. This instruction compares the content in **S_1** and **S_2**. Take "FLD=" for example, if the result is "=", the continuity of the instruction is enabled. If the result is "≠", the continuity of the instruction is disabled.
2. The user can specify the floating point value directly into operands **S_1** and **S_2** (e.g. F1.2) or store the floating point value in D registers for further operation.
3. FLD※ instruction is used for direct connection with left hand bus bar.

API No.	32 -bit instruction	Continuity condition	Discontinuity condition
275	FLD＝	$S_1 = S_2$	$S_1 \neq S_2$
276	FLD＞	$S_1 > S_2$	$S_1 \leqq S_2$
277	FLD＜	$S_1 < S_2$	$S_1 \geqq S_2$
278	FLD＜＞	$S_1 \neq S_2$	$S_1 = S_2$
279	FLD＜＝	$S_1 \leqq S_2$	$S_1 > S_2$
280	FLD＞＝	$S_1 \geqq S_2$	$S_1 < S_2$

Program Example:

When the content in D200(D201) ≤ F1.2 and X1 is ON, Y21 = ON and latched.



API	Mnemonic	Operands	Function	Controllers
281~ 286	FAND※	S_1 S_2	Floating Point Contact Type Comparison AND※	ES2/EX2 SS2 SA2 SX2

Type / OP	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S_1											*	*	*			FAND※: 9 steps
S_2											*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source device 1 **S_2**: Source device 2

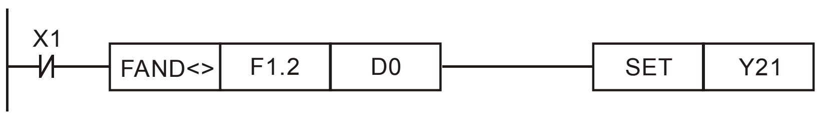
Explanations:

1. This instruction compares the content in **S_1** and **S_2**. Take "FAND =" for example, if the result is "=", the continuity of the instruction is enabled. If the result is "≠", the continuity of the instruction is disabled.
2. The user can specify the floating point value directly into operands **S_1** and **S_2** (e.g. F1.2) or store the floating point value in D registers for further operation.
3. FAND※ instruction is used for serial connection with contacts.

API No.	32-bit instruction	Continuity condition	Discontinuity condition
281	FAND＝	$S_1 = S_2$	$S_1 \neq S_2$
282	FAND＞	$S_1 > S_2$	$S_1 \leqq S_2$
283	FAND＜	$S_1 < S_2$	$S_1 \geqq S_2$
284	FAND＜＞	$S_1 \neq S_2$	$S_1 = S_2$
285	FAND＜＝	$S_1 \leqq S_2$	$S_1 > S_2$
286	FAND＞＝	$S_1 \geqq S_2$	$S_1 < S_2$

Program Example:

When X1 is OFF and the content in D100(D101) is not equal to F1.2, Y21 = ON and latched.



API	Mnemonic	Operands	Function	Controllers
287~292	FOR※	S1 S2	Floating Point Contact Type Comparison OR※	ES2/EX2 SS2 SA2 SX2

Type / OP	Bit Devices				Word devices											Program Steps
	X	Y	M	S	K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	FOR※: 9 steps
S_1											*	*	*			
S_2											*	*	*			

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

Operands:

S_1: Source device 1 **S_2**: Source device 2

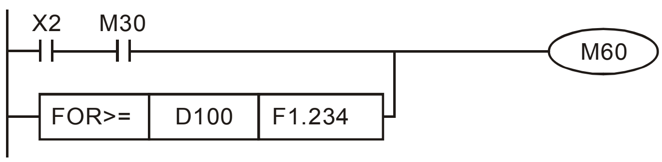
Explanations:

1. This instruction compares the content in **S_1** and **S_2**. Take "FOR =" for example, if the result is "=", the continuity of the instruction is enabled. If the result is "≠", the continuity of the instruction is disabled
2. The user can specify the floating point value directly into operands **S_1** and **S_2** (e.g. F1.2) or store the floating point value in D registers for further operation.
3. FOR※ instruction is used for parallel connection with contacts.

API No.	32-bit instruction	Continuity condition	Discontinuity condition
287	FOR＝	$S_1 = S_2$	$S_1 \neq S_2$
288	FOR＞	$S_1 > S_2$	$S_1 \leqq S_2$
289	FOR＜	$S_1 < S_2$	$S_1 \geqq S_2$
290	FOR＜＞	$S_1 \neq S_2$	$S_1 = S_2$
291	FOR＜＝	$S_1 \leqq S_2$	$S_1 > S_2$
292	FOR＞＝	$S_1 \geqq S_2$	$S_1 < S_2$

Program Example:

When both X2 and M30 are On and the content in D100(D101) ≥ F1.234, M60 = ON..



Communications

4

This chapter introduces information regarding the communications ports of the PLC. Through this chapter, the user can obtain a full understanding about PLC communication ports.

Chapter Contents

4.1 Communication Ports

DVP-ES2/EX2/SA2/SX2 offers 3 communication ports (COM1~COM3), and DVP-SS2 offers 2 COM ports (COM1~COM2). COM ports of the above models support DELTA Q-link communication format on HMI. Refresh rate of HMI can be increased by this function.

COM1: RS-232 communication port. COM1 can be used as master or slave and is the major COM port for PLC programming.
COM2 : RS-485 communication port. COM2 can be used as master or slave.
COM3 (ES2/EX2/SA2): RS-485 communication port. COM3 can be used as master or slave.
COM3 (SX2): USB communication port. COM3 can be used as slave only
Both 3 COM ports support Modbus ASCII or RTU communication format.

Communication Format:

<table>
<tr><th>COM port / Parameter</th><th>RS-232 (COM1)</th><th>RS-485 (COM2)</th><th>RS-485 (COM3)</th><th>RS-485 (COM3)</th></tr>
<tr><td>Baud rate</td><td>110~115200 bps</td><td colspan="2">110~921000 bps</td><td>110~115200 bps</td></tr>
<tr><td>Data length</td><td colspan="4">7~8bits</td></tr>
<tr><td>Parity</td><td colspan="4">Even / Odd / None parity check</td></tr>
<tr><td>Length of stop bit</td><td colspan="4">1~2 bits</td></tr>
<tr><td>Register for Setting</td><td>D1036</td><td>D1120</td><td colspan="2">D1109</td></tr>
<tr><td>Retain communication format</td><td>M1138</td><td>M1120</td><td colspan="2">M1136</td></tr>
<tr><td>ASCII mode</td><td colspan="3">Available for both Master/Slave</td><td>Available for Slave</td></tr>
<tr><td>RTU mode</td><td colspan="3">Available for both Master/Slave</td><td>Available for Slave</td></tr>
<tr><td>ASCII/RTU mode selection</td><td>M1139</td><td>M1143</td><td colspan="2">M1320</td></tr>
<tr><td>Communication address of Slave</td><td colspan="2">D1121</td><td colspan="2">D1255</td></tr>
<tr><td>Data length for access (ASCII)</td><td colspan="4">100 registers</td></tr>
<tr><td>Data length for access (RTU)</td><td colspan="4">100 registers</td></tr>
</table>

Default communication settings for all COM ports:

- Modbus ASCII
- 7 data bits
- 1 stop bit
- Even parity
- Baud rate: 9600

4.2 Communication Protocol ASCII mode

Communication Data Structure

9600 (Baud rate), 7 (data bits), Even (Parity), 1 (Start bit), 1 (Stop bit)

Field name	Content	Explanation
Start bit	STX	Start bit ‘:’ (3AH)
Communication address	ADR 1	Address consists of 2 ASCII codes
	ADR 0	
Command code	CMD 1	Command code consists of 2 ASCII codes
	CMD 0	
Data	DATA (0)	Data content consist of 2n ASCII codes, n≤205
	DATA (1)	
	……….	
	DATA (n-1)	
LRC checksum	LRC CHK 1	LRC checksum consists of 2 ASCII codes
	LRC CHK 0	
Stop bit	END1	Stop bit consists of 2 ASCII codes END1 = CR (0DH), END0 = LF (0AH)
	END0	

Corresponding table for Hexadecimal value and ASCII codes

ASCII	“0“	“1“	“2“	“3“	“4“	“5“	“6“	“7“
Hex	30H	31H	32H	33H	34H	35H	36H	37H
ASCII	“8“	“9“	“A“	“B“	“C“	“D“	“E“	“F“
Hex	38H	39H	41H	42H	43H	44H	45H	46H

4.2.1 ADR (Communication Address)

Valid communication addresses are in the range of 0~254. Communication address equals to 0 means broadcast to all PLCs. PLC will not respond to a broadcast message. PLC will reply a normal message to the master device when communication address is not 0.

Example, ASCII codes for communication address 16 in Decimal. (16 in Decimal = 10 in Hex)

(ADR 1, ADR 0)=’1’,’0’⇨’1’=31H, ‘0’ = 30H

4.2.2 CMD (Command code) and DATA

The content of access data depends on the command code.

Available setting for command code:

CMD(Hex)	Explanation	Device
01 (01 H)	Read status of contact	S, Y, M, T, C
02 (02 H)	Read status of contact	S, X, Y, M,T, C
03 (03 H)	Read content of register	T, C, D
05 (04 H)	Force ON/OFF single contact	S, Y, M, T, C
06 (06 H)	Set content of single register	T, C, D
15 (0F H)	Force ON/OFF multiple contacts	S, Y, M, T, C
16 (10 H)	Set content of multiple registers	T, C, D
17 (11 H)	Retrieve information of Slave	None
23 (17 H)	Simultaneous data read/write in a polling of EASY PLC LINK	None

Example: Read devices T20~T27 (address: H0614~H61B) from Slave ID#01(station number)

PC→PLC

": 01 03 06 14 00 08 DA CR LF"

Sent massage:

Field name	ASCII	Hex
STX	:	3A
Slave Address	01	30 31
Command code	03	30 33
Starting Address High	06	30 36
Starting Address Low	14	31 34
Number of Points High	00	30 30
Number of Points Low	08	30 38
LRC checksum	DA	44 41
END	CR LF	0D 0A

PLC→PC

": 01 03 10 00 01 00 02 00 03 00 04 00 05 00 06 00 07 00 08 C8 CR LF"

Responded massage:

Field name	ASCII	Hex
STX	:	3A
Slave Address	01	30 31
Command code	03	30 33

Field name	ASCII	Hex
Bytes Count	10	31 30
Data Hi (T20)	00	30 30
Data Lo (T20)	01	30 31
Data Hi (T21)	00	30 30
Data Lo (T21)	02	30 32
Data Hi (T22)	00	30 30
Data Lo (T22)	03	30 33
Data Hi (T23)	00	30 30
Data Lo (T23)	04	30 34
Data Hi (T24)	00	30 30
Data Lo (T24)	05	30 35
Data Hi (T25)	00	30 30
Data Lo (T25)	06	30 36
Data Hi (T26)	00	30 30
Data Lo (T26)	07	30 37
Data Hi (T27)	00	30 30
Data Lo (T27)	08	30 38
Check sum(LRC)	C8	43 38
END	CR LF	0D 0A

4.2.3 LRC CHK (checksum)

LRC (Longitudinal Redundancy Check) is calculated by summing up the Hex values from ADR1 to last data character then finding the 2's-complement negation of the sum.

Example: Read the content of register at address 0401H. 01H+03H+04H+01H+00+01H = 0AH.
The 2's-complement of 0AH: F6H

Field name	ASCII	Hex
STX	:	3A
Slave Address	**01**	30 31
Command code	**03**	30 33
Starting data address Hi	**04**	30 34
Starting data address Lo	**01**	30 31
Number of data Hi	**00**	30 30
Number of data Lo	**01**	30 31
LRC checksum	F6	46 36
END	CR LF	0D 0A

Exception response:

The PLC is expected to return a normal response after receiving command messages from the master device. The following table depicts the conditions that either a no response or an error response is replied to the master device.

1. The PLC did not receive a valid message due to a communication error; thus the PLC has no response. The master device will eventually process a timeout condition.

2. The PLC 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:
Sent message:

Field Name	ASCII	Hex
STX	:	3A
Slave Address	01	30 31
Command code	01	30 31
Starting Address Hi	04	30 34
Starting Address Lo	00	30 30
Number of Points Hi	00	30 30
Number of Points Lo	10	31 30
Error Check (LRC)	EA	45 41
END	CR LF	0D 0A

Feedback message:

Field Name	ASCII	Hex
STX	:	3A
Slave Address	01	30 31
Function	81	38 31
Exception Code	02	30 32
Error Check (LRC)	7C	37 43
END	CR LF	0D 0A

Exception code:	Explanation:
01	Illegal command code: The command code received in the command message is invalid for PLC.
02	Illegal device address: The device address received in the command message is invalid for PLC.
03	Illegal device content: The data received in the command message is invalid for PLC.
07	1. Checksum Error - Check if the checksum is correct 2. Illegal command messages - The command message is too short. - Length command message is out of range.

4.3 Communication Protocol RTU mode

Communication Data Structure

9600 (Baud rate), 8 (data bits), EVEN (Parity), 1 (Start bit), 1 (Stop bit)

START	No data input ⩾ 10 ms
Address	Communication Address: the 8-bit binary address
Command code	Command Code: the 8-bit binary address
DATA (n-1)	Data Contents: n × 8-bit BIN data, n≦202
…….	
DATA 0	
CRC CHK Low	CRC Checksum: The 16-bit CRC checksum is composed of 2 8-bit binary codes
CRC CHK High	
END	No data input ⩾ 10 ms

4.3.1 Address (Communication Address)

Valid communication addresses are in the range of 0~254. Communication address equals to 0 means broadcast to all PLCs. PLC will not respond to a broadcast message. PLC will reply a normal message to the master device when communication address is not 0.

Example, communication address should be set to 10 (Hex) when communicating with a PLC with address 16 (Dec) (16 in Decimal = 10 in Hex)

4.3.2 CMD (Command code) and DATA

The content of access data depends on the command code. For descriptions of available command codes, please refer to **4.2.2** in this chapter.
Example: read consecutive 8 words from address 0614H~H61B (T20~T27) of PLC Slave ID#1.

PC→PLC
" 01 03 06 14 00 08 04 80"

Sent message:

Field Name	Example (Hex)
START	No data input ≥ 10 ms
Slave Address	01
Command code	03
Starting Address	06
	14
Number of Points	00
	08
CRC CHK Low	04
CRC CHK High	80
END	No data input ≥ 10 ms

PLC→PC
" 01 03 10 00 01 00 02 00 03 00 04 00 05 00 06 00 07 00 08 72 98"
Feedback message:

Field Name	Example (Hex)
START	No data input ≥ 10 ms
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
CRC CHK Low	72
CRC CHK High	98
END	No data input ≥ 10 ms

4.3.3 CRC CHK (check sum)

The CRC Check starts from “Slave Address” and ends in “The last data content.” Calculation of CRC:

Step 1: Set the 16-bit register (CRC register) = FFFFH.

Step 2: Operate XOR on the first 8-bit message (Address) and the lower 8 bits of CRC register. Store the result in the CRC register

Step 3: Right shift CRC register for a bit and fill “0” into the highest bit.

Step 4: Check the lowest bit (bit 0) of the shifted value. If bit 0 is 0, fill in the new value obtained at step 3 to CRC register; if bit 0 is NOT 0, operate XOR on A001H and the shifted value and store the result in the CRC register.

Step 5: Repeat step 3 – 4 to finish all operation on all the 8 bits.

Step 6: Repeat step 2 – 5 until the operation of all the messages are completed. The final value obtained in the CRC register is the CRC checksum. Care should be taken when placing the LOW byte and HIGH byte of the obtained CRC checksum.

Calculation example of the CRC Check using the C language:

```
unsigned char* data      ← // index of the command message
unsigned char length    ← // length of the command message
unsigned int crc_chk(unsigned char* data, unsigned char length)
{
   int j;
   unsigned int reg_crc=0Xffff;
   while(length--)
```

```
  {
    reg_crc ^= *data++;
    for (j=0;j<8;j++)
    {
      If (reg_crc & 0x01) reg_crc=(reg_crc>>1) ^ 0Xa001; /* LSB(b0)=1 */
      else reg_crc=reg_crc >>1;
    }
  }
  return reg_crc;     // the value that sent back to the CRC register finally
}
```

Exception response:

The PLC is expected to return a normal response after receiving command messages from the master device. The following content depicts the conditions that either no response situation occurs or an error response is replied to the master device.

1. The PLC did not receive a valid message due to a communication error; thus the PLC has no response. In this case, condition of communication timeout has to be set up in the master device

2. The PLC receives a valid message without a communication error, but cannot accommodate it. In this case, 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:
Sent message:

Field Name	Example (Hex)
START	No data input ⩾ 10 ms
Slave Address	01
Command code	01
Starting Address	04
	00
Number of Points	00
	10
CRC CHK Low	3C
CRC CHK High	F6
END	No data input ⩾ 10 ms

Feedback message:

Field Name	Example (Hex)
START	No data input ≥ 10 ms
Slave Address	01
Function	81
Exception Code	02
CRC CHK Low	C1
CRC CHK High	91
END	No data input ≥ 10 ms

4.4 PLC Device Address

<table>
<tr><th rowspan="2">Device</th><th rowspan="2">Range</th><th colspan="3">Effective Range</th><th rowspan="2">MODBUS Address</th><th rowspan="2">Address</th></tr>
<tr><th>ES2/EX2</th><th>SS2</th><th>SA2/SX2</th></tr>
<tr><td>S</td><td>000~255</td><td rowspan="4">000~1023</td><td colspan="2" rowspan="4">000~1023</td><td>000001~000256</td><td>0000~00FF</td></tr>
<tr><td>S</td><td>256~511</td><td>000257~000512</td><td>0100~01FF</td></tr>
<tr><td>S</td><td>512~767</td><td>000513~000768</td><td>0200~02FF</td></tr>
<tr><td>S</td><td>768~1023</td><td>000769~001024</td><td>0300~03FF</td></tr>
<tr><td>X</td><td>000~377 (Octal)</td><td>000~377</td><td colspan="2">000~377</td><td>101025~101280</td><td>0400~04FF</td></tr>
<tr><td>Y</td><td>000~377 (Octal)</td><td>000~377</td><td colspan="2">000~377</td><td>001281~001536</td><td>0500~05FF</td></tr>
<tr><td rowspan="2">T</td><td>000~255 bit</td><td>000~255</td><td colspan="2">000~255</td><td>001537~001792</td><td>0600~06FF</td></tr>
<tr><td>000~255 word</td><td>000~255</td><td colspan="2">000~255</td><td>401537~401792</td><td>0600~06FF</td></tr>
<tr><td>M</td><td>000~255</td><td rowspan="16">0000
~
4095</td><td colspan="2" rowspan="16">0000~4095</td><td rowspan="6">002049~003584</td><td>0800~08FF</td></tr>
<tr><td>M</td><td>256~511</td><td>0900~09FF</td></tr>
<tr><td>M</td><td>512~767</td><td>0A00~0AFF</td></tr>
<tr><td>M</td><td>768~1023</td><td>0B00~0BFF</td></tr>
<tr><td>M</td><td>1024~1279</td><td>0C00~0CFF</td></tr>
<tr><td>M</td><td>1280~1535</td><td>0D00~0DFF</td></tr>
<tr><td>M</td><td>1536~1791</td><td rowspan="10">045057~047616</td><td>B000~B0FF</td></tr>
<tr><td>M</td><td>1792~2047</td><td>B100~B1FF</td></tr>
<tr><td>M</td><td>2048~2303</td><td>B200~B2FF</td></tr>
<tr><td>M</td><td>2304~2559</td><td>B300~B3FF</td></tr>
<tr><td>M</td><td>2560~2815</td><td>B400~B4FF</td></tr>
<tr><td>M</td><td>2816~3071</td><td>B500~B5FF</td></tr>
<tr><td>M</td><td>3072~3327</td><td>B600~B6FF</td></tr>
<tr><td>M</td><td>3328~3583</td><td>B700~B7FF</td></tr>
<tr><td>M</td><td>3584~3839</td><td>B800~B8FF</td></tr>
<tr><td>M</td><td>3840~4095</td><td>B900~B9FF</td></tr>
<tr><td rowspan="4">C</td><td rowspan="2">000~199 (16-bit)</td><td>000~199</td><td colspan="2">000~199</td><td>003585~003784</td><td>0E00~0EC7</td></tr>
<tr><td>000~199</td><td colspan="2">000~199</td><td>403585~403784</td><td>0E00~0EC7</td></tr>
<tr><td rowspan="2">200~255 (32-bit)</td><td>200~255</td><td colspan="2">200~255</td><td>003785~003840</td><td>0EC8~0EFF</td></tr>
<tr><td>200~255</td><td colspan="2">200~255</td><td>401793~401903</td><td>0700~076F</td></tr>
</table>

Device	Range	Effective Range			MODBUS Address	Address
		ES2/EX2	SS2	SA2/SX2		
					(Odd address valid)	
D	000~255	0000 ~ 9999	0000 ~ 4999	0000 ~ 9999	404097~405376	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				405377~408192	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	4096~4351				436865~440960	9000~90FF
D	4352~4999					9100~91FF
D	4608~4863					9200~92FF
D	4864~5119					9300~93FF
D	5120~5375		N/A			9400~94FF
D	5376~5631					9500~95FF
D	5632~5887					9600~96FF
D	5888~6143					9700~97FF
D	6144~6399					9800~98FF
D	6400~6655					9900~99FF
D	6656~6911					9A00~9AFF
D	6912~7167					9B00~9BFF
D	7168~7423					9C00~9CFF
D	7424~7679					9D00~9DFF
D	7680~7935					9E00~9EFF
D	7936~8191					9F00~9FFF
D	8192~8447				440961~442768	A000~A0FF
D	8448~8703					A100~A1FF
D	8704~8959					A200~A2FF
D	8960~9215					A300~A3FF
D	9216~9471					A400~A4FF
D	9472~9727					A500~A5FF
D	9728~9983					A600~A6FF
D	9984~9999					A700~A70F

4.5 Command Code

4.5.1 Command Code: 01, Read Status of Contact (Input point X is not included)

Number of Points (max) = 255 (Dec) = FF (Hex)
Example：Read contacts T20~T56 from Slave ID#1
PC→PLC “:01 01 06 14 00 25 BF CR LF”
Sent message:

Field Name	ASCII
STX	:
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
ETX 1	0D (Hex)
ETX 0	0A (Hex)

Assume Number of Points in sent message is **n** (Dec), quotient of **n**/8 is **M** and the remainder is **N.** When **N** = 0, Bytes Count in feedback message will be **M**; when N≠0, Bytes Count will be **M+1.**

PLC→PC “:01 01 05 CD 6B B2 0E 1B D6 CR LF”
Feedback message:

Field Name	ASCII
STX	:
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 (Hex)
END 0	0A (Hex)

4.5.2 Command Code: 02, Read Status of Contact (Input point X is included)

Example: Read status of contact Y024~Y070 from Slave ID#01

PC→PLC “: 01 02 05 14 00 25 BF CR LF”

Sent message:

Field Name	ASCII
STX	:
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 (Hex)
END 0	0A (Hex)

4

Assume Number of Points in sent message is **n** (Dec), quotient of **n**/8 is **M** and the remainder is **N.** When **N** = 0, Bytes Count in feedback message will be **M**; when N≠0, Bytes Count will be **M+1**.

PLC→PC “: 01 01 05 CD 6B B2 0E 1B E5 CR LF”

Feedback message:

Field Name	ASCII
STX	:
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 (Hex)
END 0	0A (Hex)

4.5.3 Command Code: 03, Read Content of Register (T, C, D)

Example: Read coils T20~T27 from Slave ID#01

PC→PLC “: 01 03 06 14 00 08 DA CR LF”

Sent message:

Field Name	ASCII
STX	:
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 (Hex)
END 0	0A (Hex)

PLC→PC

“:01 03 10 00 01 00 02 00 03 00 04 00 05 00 06 00 07 00 08 B8 CR LF”

Feedback message:

Field Name	ASCII
STX	:
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
Data Lo (T24)	05
Data Hi (T25)	00
Data Lo (T25)	06

Field Name	ASCII
Data Hi (T26)	00
Data Lo (T26)	07
Data Hi (T27)	00
Data Lo (T27)	08
Error Check (LRC)	C8
END 1	0D (Hex)
END 0	0A (Hex)

4.5.4 Command Code: 05, Force ON/OFF single contact

The Force data FF00 (Hex) indicates force ON the contact. The Force data 0000 (Hex) indicates force OFF the contact. Also, When MMNN = 0xFF00, the coil will be ON, when MMNN = 0x0000, the coil will be OFF. Other force data is invalid and will not take any effect.

Example: Force coil Y0 ON

PC→PLC “: 01 05 05 00 FF 00 F6 CR LF”

Sent message:

Field Name	ASCII
STX	:
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 (Hex)
END 0	0A (Hex)

PLC→PC “: 01 05 05 00 FF 00 F6 CR LF”

Feedback message:

Field Name	ASCII
STX	:
Slave Address	01
Command code	05
Coil Address Hi	05
Coil Address Lo	00
Force Data Hi	FF

Field Name	ASCII
Force Data Lo	00
Error Check (LRC)	F6
END 1	0D (Hex)
END 0	0A (Hex)

4.5.5 Command Code: 06, Set content of single register

Example: Set content of register T0: 12 34 (Hex)

PC→PLC "： 01 06 06 00 12 34 AD CR LF"

Sent message:

Field Name	ASCII
STX	:
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 (Hex)
END 0	0A (Hex)

PLC→PC "： 01 06 06 00 12 34 AD CR LF"

Feedback message:

Field Name	ASCII
STX	:
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 (Hex)
END 0	0A (Hex)

4.5.6 Command Code: 15, Force ON/OFF multiple contacts

Max contacts/coils available for Force ON/OFF: 255

Example: Set Coil Y007…Y000 = 1100 1101, Y011…Y010 = 01.

PC→PLC “: 01 0F 05 00 00 0A 02 CD 01 11 CR LF”

Sent message:

Field Name	ASCII
STX	:
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 (Hex)
END 0	0A (Hex)

PLC→PC “: 01 0F 05 00 00 0A E1 CR LF”

Feedback message:

Field Name	ASCII
STX	:
Slave Address	01
Command code	0F
Register T0 Address Hi	05
Register T0 Address Lo	00
Preset Data Hi	00
Preset Data Lo	0A
Error Check (LRC)	E1
END 1	0D (Hex)
END 0	0A (Hex)

4.5.7 Command Code: 16, Set content of multiple registers

Example: Set register T0 to 00 0A , T1 to 01 02 .

PC→PLC “: 01 10 06 00 00 02 04 00 0A 01 02 D6 CR LF”

Sent message:

Field Name	ASCII
STX	:
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)	D6
END 1	0D(Hex)
END 0	0A(Hex)

PLC→PC “: 01 10 06 00 00 02 E7 CR LF”

Feedback message:

Field Name	ASCII
STX	3A
Slave Address	01
Command code	10
Starting Address Hi	06
Starting Address Lo	00
Number of Registers Hi	00
Number of Registers Lo	02
Error Check (LRC)	E7
END 1	0D (Hex)
END 0	0A (Hex)

MEMO

Sequential Function Chart

5

This chapter provides information for programming in SFC mode.

Chapter Contents

5.1 Step Ladder Instruction [STL], [RET]

Mnemonic	Operands	Function	Program steps	Controllers
STL	S0~S1023	Starts STL program	1	ES2/EX2 SS2 SA2 SX2

Explanation:

STL Sn constructs a step point. When STL instruction appears in the program, the main program will enter a step ladder status controlled by steps. The initial STL program has to start from S0 ~ S9 as initial step points. The No. of Step points cannot be repeated.

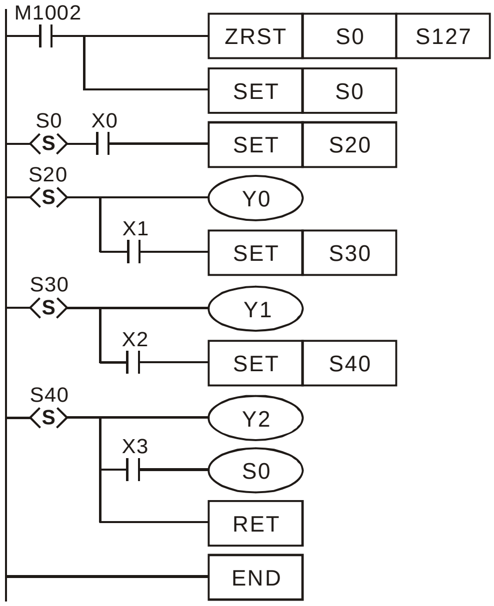
Mnemonic	Operands	Function	Program steps	Controllers
RET	None	Ends STL program	1	ES2/EX2 SS2 SA2 SX2

Explanation:

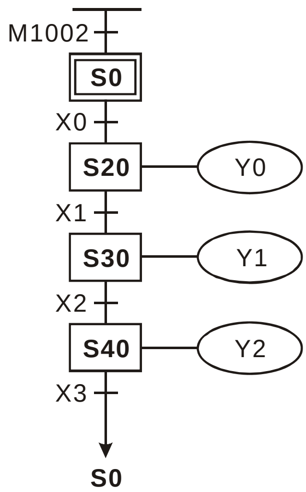
RET instruction indicates the end of a step ladder program starting from S0 ~ S9, i.e. the execution returns to main program after RET is executed. Maximum 10 initial steps (S0 ~ S9) can be applied and every initial step requires a RET instruction as an end of STL program. With the step ladder program composed of STL/RET instructions, SFC can perform a step by step control process.

Program Example:

Step ladder diagram:



SFC:



5.2 Sequential Function Chart (SFC)

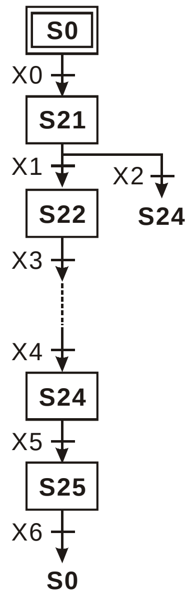
In the application of automation control, a seamless combination between electrical control and mechanical control is required for completing an automation process. The sequential control of automation process can be divided into several steps (states). Each step is designated with own action and the transition from one step to another generally requires some transition criteria (condition). The action of the previous step finishes as long as all criteria is true. When next step begins, the action of previous step will be cleared. The step-by-step transition process is the concept for designing sequential function chart (SFC).

Features:

<table>
<tr>
<td colspan="2">
1. Users do not have to consider the sequential relationship between outputs as general ladder logic because STL operation process can execute multiple outputs or interlocked outputs automatically. An easy sequential design between the steps is the only thing required to control the machines.

2. The actions in SFC are easy to understand. Also, it’s easy to do a trial operation, error detecting or period maintenance.

3. SFC functions as a flow chart. The STL operation works on the internal step relay S, which is also the step points representing each state in SFC. When current step is finished, the program proceeds to the next step according to the transition condition and the desired continuous control purpose can be achieved by this process.

4. Cycle process can be performed. Please refer to the SFC opposite. Initial step S0 transfers to general step S21 by transition condition X0. S21 transfers to S22 or jumps to S24 by the condition X1 and X2. The process finally proceeds to S25 then a single cycle process is completed when S25 returns to S0 with transition condition X6 fulfilled.
</td>
<td>
SFC:


</td>
</tr>
<tr>
<td colspan="3">Explanation on SFC Toolbar Icons in Ladder Editor (WPLSoft)</td>
</tr>
<tr>
<td>
</td>
<td colspan="2">Ladder diagram mode. The icon inserts general ladder diagram before the STL diagram, usually the instructions for initializing the STL program.</td>
</tr>
</table>

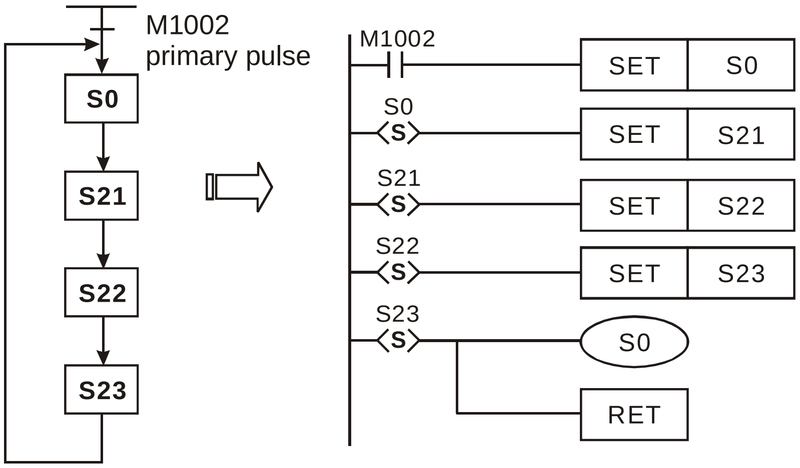
F2	Initial step in SFC. S0 ~ S9.are applicable
F3	General step. S10 ~ S1023 are applicable.
F4	Step jump. Used for a step to jump to another non-adjacent step. (Jumping up/down to non-adjacent steps in the same sequence, returning to initial step, or jumping among different sequences.)
F5	Transition condition. The transition condition to move between each step point.
F6	Alternative divergence. Alternative divergence is used for a step point to transfer to different corresponding step points by different transition conditions.
F7	Alternative convergence. Alternative convergence is used for two step points or more to transfer to the same step point according to transition condition.
F8	Simultaneous divergence. Simultaneous divergence is used for a step point to transfer to two step points or more by the same transition condition.
F9	Simultaneous convergence. Simultaneous convergence is used for two step points or more to transfer to the same step point with the same transition condition when multiple conditions are fulfilled at the same time.

5.3 The Operation of STL Program

Step ladder diagram (STL) is a programming method for users to write a program which functions similar to SFC. STL provides PLC program designers a more readable and clear programming method as drawing a flow chart. The sequences or steps in the below SFC is quite understandable and can be translated into the ladder diagram opposite.

STL program starts with STL instruction and ends with RET instruction. STL Sn constructs a step point. When STL instruction appears in the program, the main program will enter a step ladder status controlled by steps. RET instruction indicates the end of a step ladder program starting from initial steps S0 ~ S9 and every initial step requires a RET instruction as an end of STL program.

If there is no RET instruction at the end of a step sequence, errors will be detected by WPLSoft.

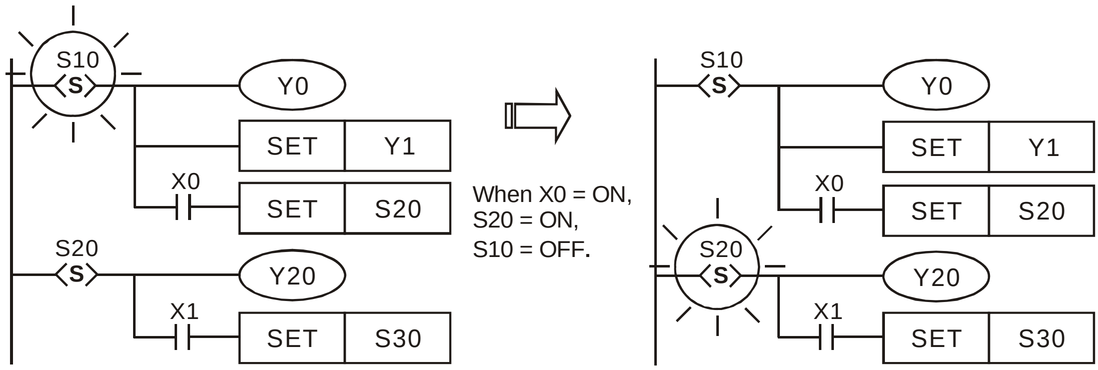


Actions of Step Points:

STL program is composed of many step points, and each step point represents a single task in the STL control process. To perform a sequential control result, every step point needs to do 3 actions.

1. Drive output coils
2. Designate the transition condition
3. Designate which step will take over the control from the current step

Example:



Explanation:

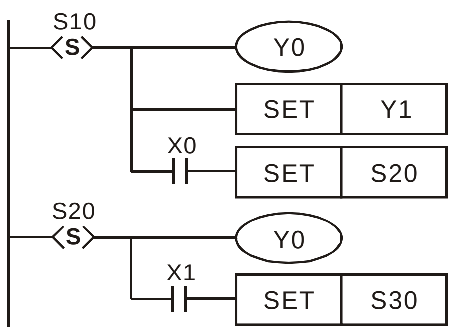
When S10 = ON, Y0 and Y1 will be ON. When X0 = ON, S20 will be ON and Y20 will be ON. When S10 = OFF, Y0 will be OFF but Y1 will still be ON (SET instruction is applied on Y1, so Y1 will be ON and latched.)

STL Transition:

When step point Sn is ON, its following output circuit will be activated. When Sn = OFF, its following output circuit will be OFF. The interval between the activation of the step point and its following output circuit is one scan cycle.

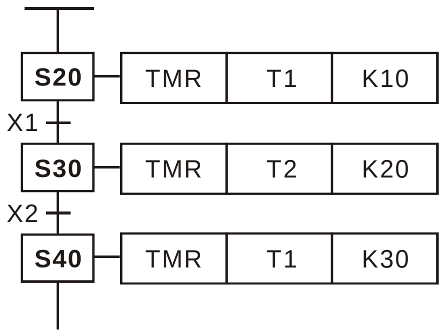
Repeated Usage of Output Coil:

4. Output coils of the same number could be used in different step points.
5. See the diagram opposite. There can be the same output device (Y0) among different steps (sequences). Y0 remains ON when S10 transfers to S20.
6. Y0 will be OFF due to the transition from S10 to S20. However when S20 is ON, Y0 will be ON again. Therefore in this case, Y0 remains ON when S10 transfers to S20.
7. For general ladder diagrams, repeated usages of output coils should be avoided. The No. of output coil used by a step should also avoid being used when the step ladder diagram returns to a general ladder diagram.



Repeated usage of timer:

See the opposite diagram. Timers can only be used repeatedly in non-adjacent steps.

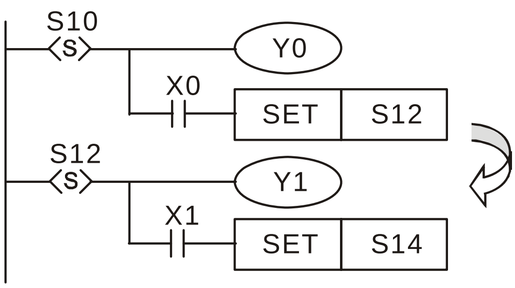


Transfer of Step Points:

SET Sn and OUT Sn instructions are used to enable (or transfer to) another step. Because there can be many step control sequences (i.e. the initial steps starting with S0 ~ S9) existing in the program. The transfer of a step can take place in the same step sequence, or be transferred to different step sequence. Usages of SET Sn and OUT Sn are different according to the transfer methods. Please see the explanations below

SET Sn

Used for driving the next step in the same sequence. After the transition, all output in the previous step will be OFF.



When SET S12 executes,
S10 transfers to S12 and
output Y10 in S10 will be OFF.

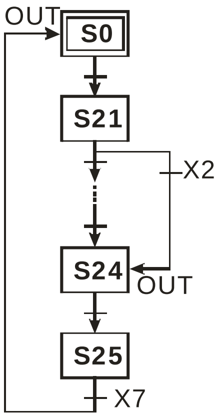
OUT Sn

Used for 1: returning to the initial step in the same step sequence, 2: jumping up/down to non-adjacent steps in the same sequence, or 3: driving steps in different sequences. After the transition, all outputs in the previous step will be cleared.

① Returning to the initial step in the same sequence.

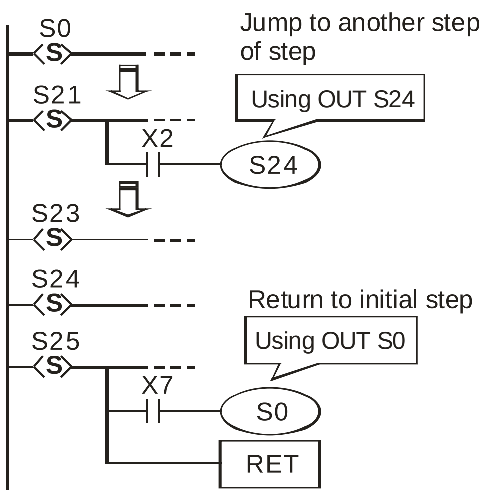
② Jumping up/down to non-adjacent steps in the same sequence.

SFC:



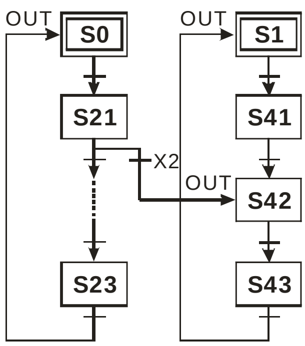
S25 returns to the initial step S0 by using OUT.

Ladder diagram:



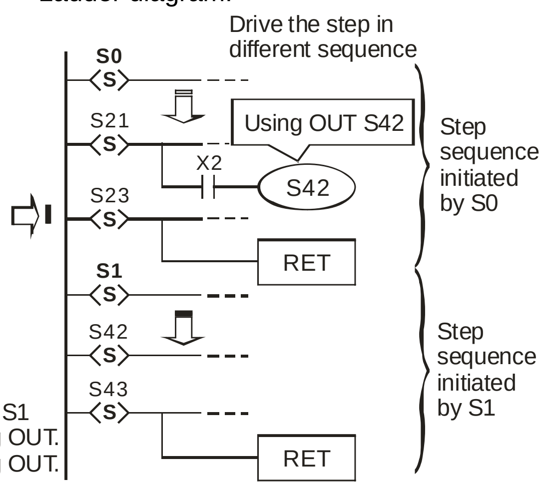
③ Driving steps in different sequences.

SFC:



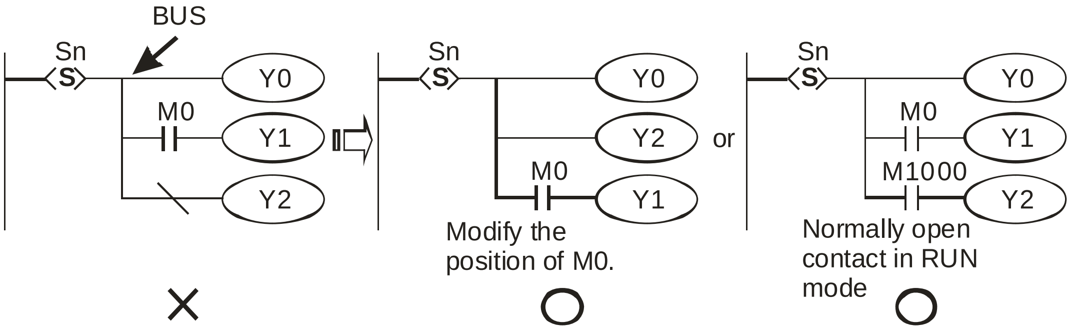
Two different step sequence: S0 and S1
S23 returns to initial step S0 by using OUT.
S43 returns to initial step S1 by using OUT.

Ladder diagram:



Cautions for Driving Output Point:

Once LD or LDI instructions are written into the second line after the step point, the bus will not be able to connect output coils directly otherwise errors will occur when compiling the ladder diagram. The following diagram explains the methods for correcting the ladder ion correct diagram.



Restrictions on Using Certain Instructions:

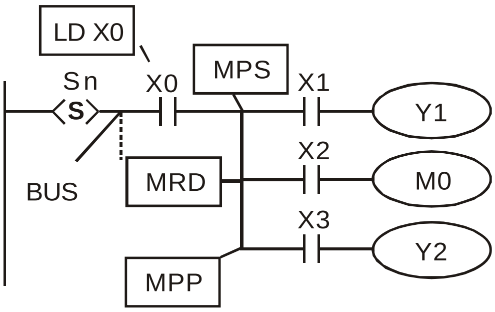
Serial/parallel circuits or instructions in general ladder diagram are also applicable in step points of STL diagram. However, there are restrictions on some of the instructions. Care should be taken when using the instructions listed in the table below.

Basic Instructions Applicable in a Step

Step point \ Basic instruction		LD/LDI/LDP/LDF AND/ANI/ANDP/ANDF OR/ORI/ORP/ORF INV/OUT/SET/RST	ANB/ORB MPS/MRD/MPP	MC/MCR
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. DO NOT use MC/MCR instruction in the step.
2. DO NOT use STL instruction in a general subroutine or interruption subroutine.
3. CJ instruction can be used in STL instruction, however this is not recommended because the actions will thus become more complicated.
4. Position of MPS/MRD/MPP instruction:

Ladder diagram:



Instruction code:

STL	Sn
LD	X0
MPS	
AND	X1
OUT	Y1
MRD	
AND	X2
OUT	M0
MPP	
AND	X3
OUT	Y2

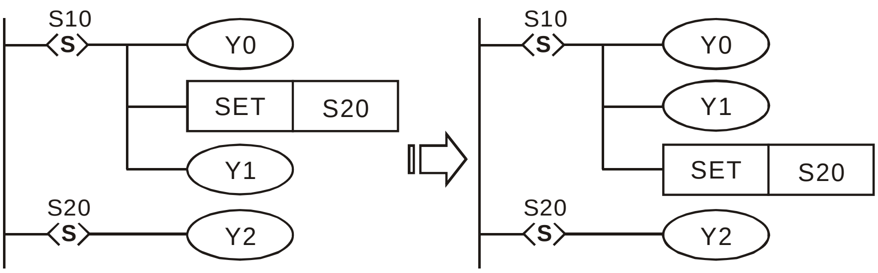
Explanation:

MPS/MRD/MPP instruction cannot be used directly on the new bus. You have to execute LD or LDI instruction first before applying MPS/MRD/MPP.

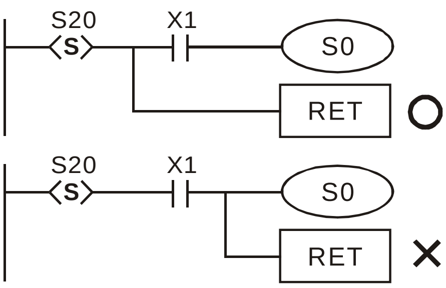
Other Points to Note:

1. The instruction used for transferring the step (SET S□ or OUT S□) are suggested to be executed after all the relevant outputs and actions in the current step are completed.

   The execution results by the PLC are the same. However, if there are many conditions or actions in S10, it is recommended to modify the diagram in the left into the diagram in the right, which executes SET S20 after all actions are completed. The sequence will be more understandable and clear with this modification.



2. As indicated in the below diagram, make sure to connect RET instruction directly after the step point rather than the NO or NC contact.



5.4 Points to Note for Designing a Step Ladder Program

1. The first step in the SFC is called the "initial step", S0 ~ S9. Use the initial step as the start of a sequence and ends with RET instruction.
2. If no STL instruction is in use, step point S can be used as a general-purpose auxiliary relay..
3. When STL instruction is in use, the No. of step S cannot be repeated.
4. Types of sequences:

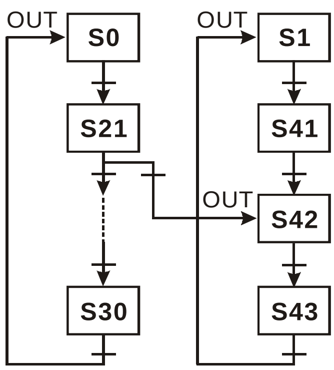
   Single sequence: Only one simple sequence without alternative divergence, alternative convergence, simultaneous divergence or simultaneous convergence in the program.

   Complicated single sequence: Only one sequence with alternative divergence, alternative convergence, simultaneous divergence and simultaneous convergence in the program.

   Multiple sequences: More than one sequence in a program, maximum 10 sequences, S0 ~ S9.

5. Sequence jump: Multiple sequences are allowed to be written into the step ladder diagram.

   - There are two sequences, S0 and S1. PLC writes in S0 ~ S30 first and S1 ~ S43 next..
   - Users can assign a step in the sequence to jump to any step in another sequence.
   - When the condition below S21 is fulfilled, the sequence will jump to step S42 in sequence S1, which is called "sequence jump."



6. Restrictions on diverging sequence: Please refer to section **5.5** for examples
   a) Max. 8 step points could be used for single divergence sequence.
   b) Max. 16 step points could be used for the convergence of multiple diverted sequences.
   c) Users can assign a step in the sequence to jump to any step in another sequence.

7. Reset step points and disable outputs
   a) Use the ZRST instruction to reset (turn off) a specified step sequence..
   b) Set ON the flag M1034 to disable Y outputs.

8. Latched step:
   The ON/OFF status of the latched step will be memorized when the power of the PLC is switched off. When the PLC is powered up again, PLC will resume the status before power-off and executes from the interrupted point. Please be aware of the area for the latched steps.

9. Special auxiliary relays and special registers: For more details please refer to **5.6 IST Instruction**.

Device	Description
M1040	Disabling step transition.
M1041	Step transition start. Flag for IST instruction.
M1042	Enabling pulse operation. Flag for IST instruction.
M1043	Zero return completed. Flag for IST instruction.
M1044	Zero point condition. Flag for IST instruction.
M1045	Disabling “all output reset” function. Flag for IST instruction.
M1046	Indicating STL status. M1046 = ON when any step is ON
M1047	Enabling STL monitoring
D1040	No. of the 1st step point which is ON.
D1041	No. of the 2nd step point which is ON
D1042	No. of the 3rd step point which is ON.
D1043	No. of the 4th step point which is ON
D1044	No. of the 5th step point which is ON.
D1045	No. of the 6th step point which is ON
D1046	No. of the 7th step point which is ON.
D1047	No. of the 8th step point which is ON

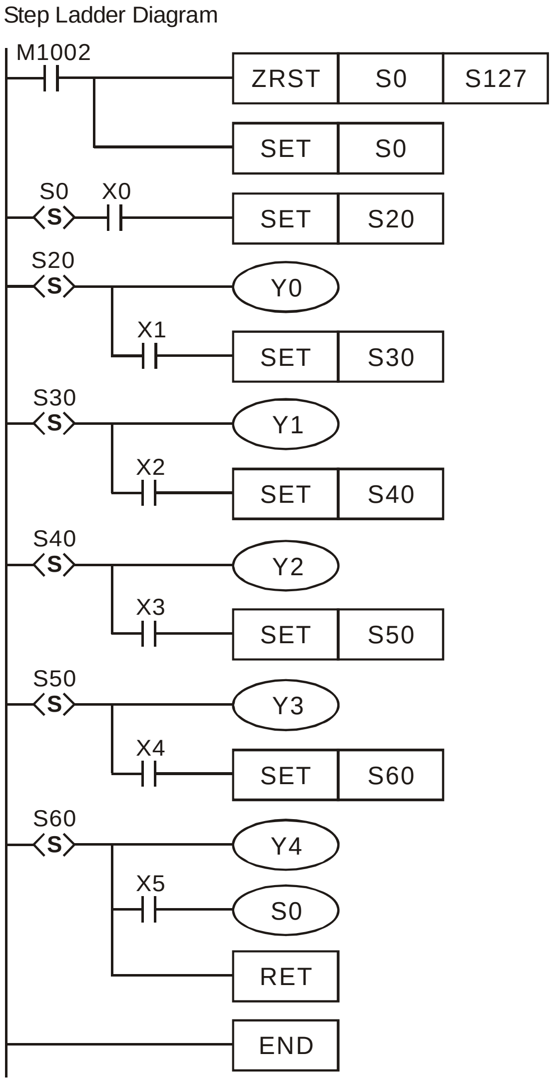
5.5 Types of Sequences

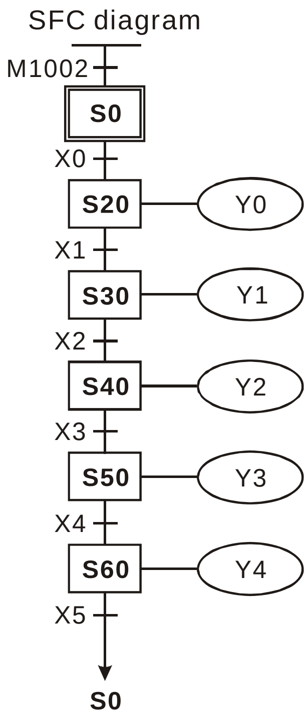
Single Sequence: The basic type of sequence

The first step in a step ladder diagram is called initial step, ranged as S0 ~ S9. The steps following the initial step are general steps numbered as S10 ~ S1023. When IST instruction is applied, S10 ~ S19 will become the steps for zero return operation.

1. Single Sequence without Divergence and Convergence

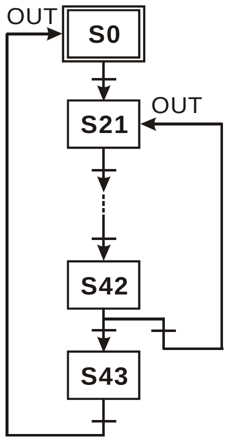
   After a sequence is completed, the control power on the steps will be transferred to the initial step.



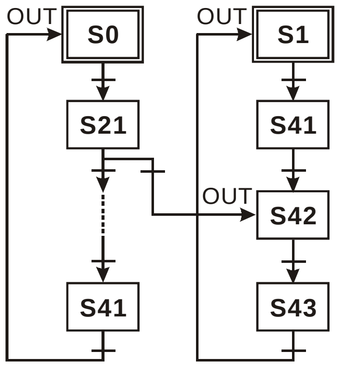


2. Step Jump

a) The control power over the step is transferred to a certain step on top.

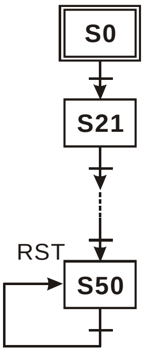


b) The control power over the step is transferred to the step in another sequence.



3. Reset Sequence

As the opposite diagram indicates, S50 will reset itself when the transition condition is fulfilled and the sequence ends here.

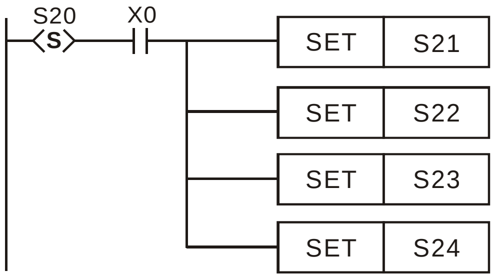


Complicated Single Sequence: Includes simultaneous divergence, alternative divergence, simultaneous convergence and alternative convergence

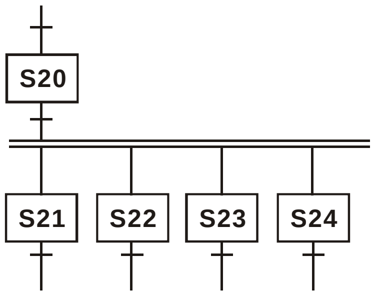
1. Structure of Simultaneous Divergence

When the condition at the current step is true, the step can be transferred to multiple steps. For example, when X0 = ON, S20 will be simultaneously transferred to S21, S22, S23 and S24.

Ladder diagram of simultaneous divergence:



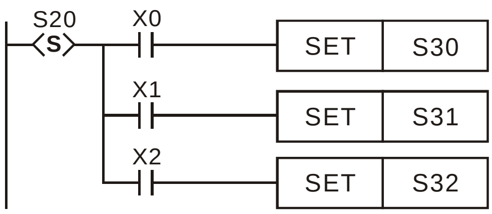
SFC diagram of simultaneous divergence:



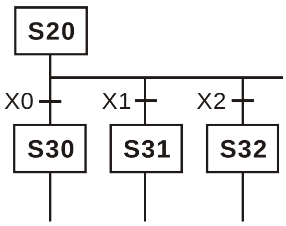
2. Structure of Alternative Divergence

When the individual condition at the current status is true, the step will be transferred to another individual step. For example, when X0 = ON, S20 will be transferred to S30; when X1 = ON, S20 will be transferred to S31; when X2 = ON, S20 will be transferred to S32.

Ladder diagram of alternative divergence:



SFC diagram of alternative divergence:

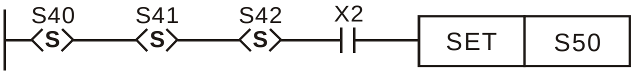


3. Structure of Simultaneous Convergence

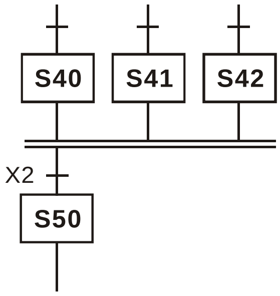
Consecutive STL instructions construct a simultaneous convergence structure. When the transition condition is true after continuous steps, the operation will be transferred to next step.

In simultaneous convergence, only when all sequences are completed will the transfer be allowed.

Ladder diagram of simultaneous convergence:



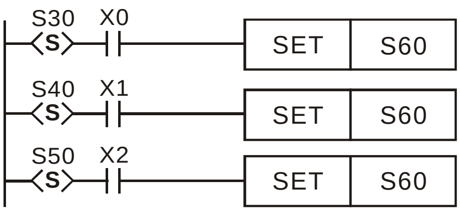
SFC diagram of simultaneous convergence:



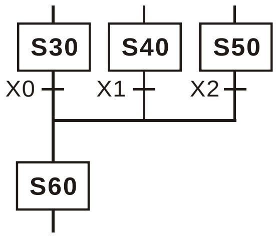
4. Structure of Alternative Convergence

The following ladder explains the structure of alternative convergence. Program operation will transfer to S60 as long as one of the transition conditions of S30, S40 or S50 is ON.

Ladder diagram of alternative convergence:

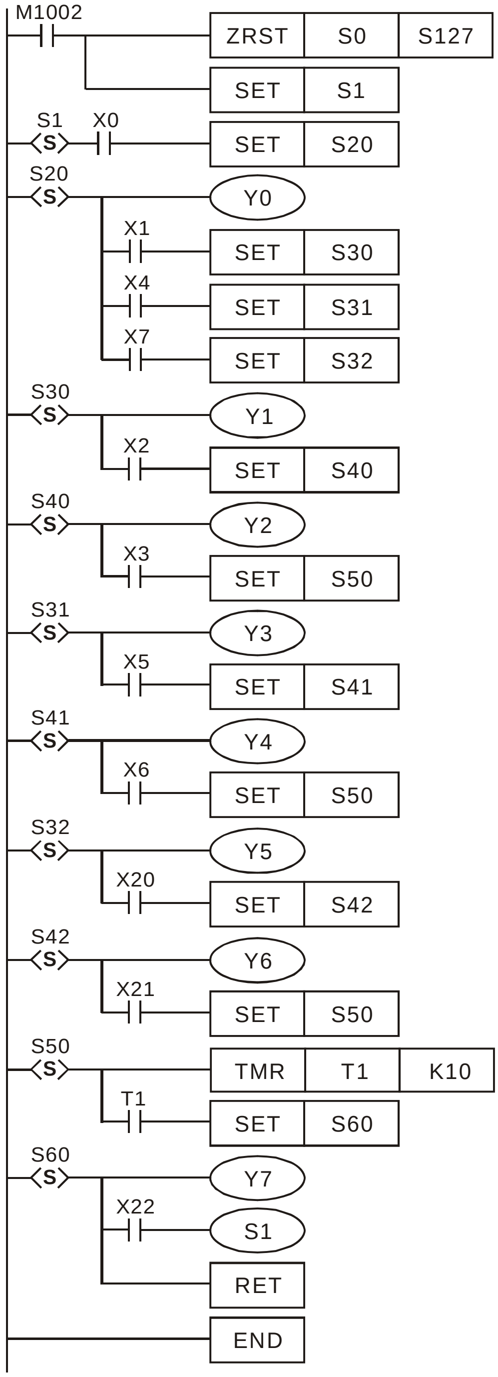


SFC diagram of alternative convergence:

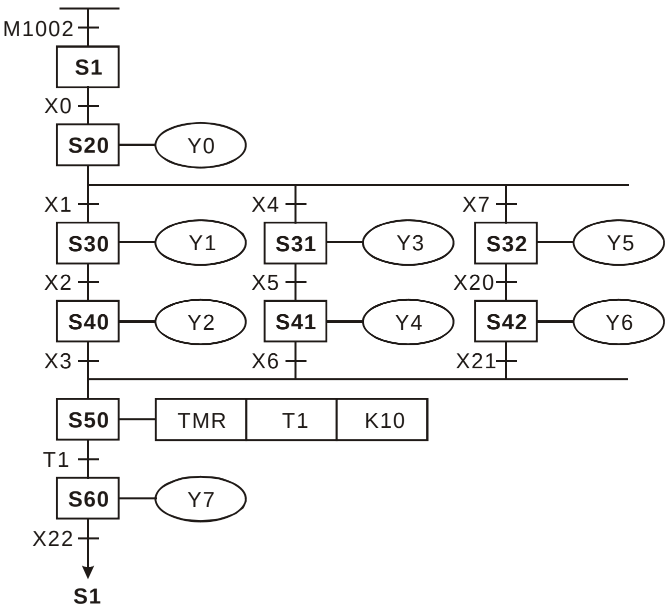


Example of alternative divergence & alternative convergence:

Step Ladder Diagram:

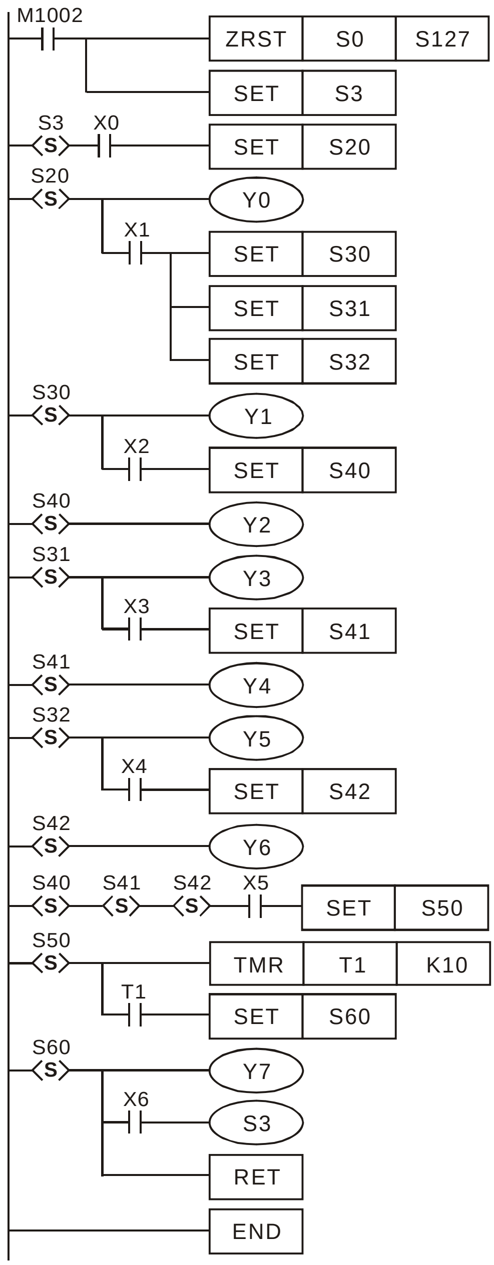


SFC Diagram:

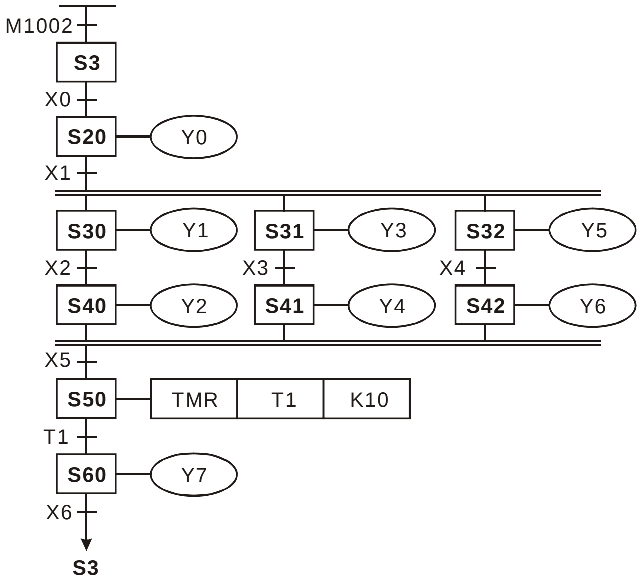


Example of simultaneous divergence & simultaneous convergence:

Step Ladder Diagram:

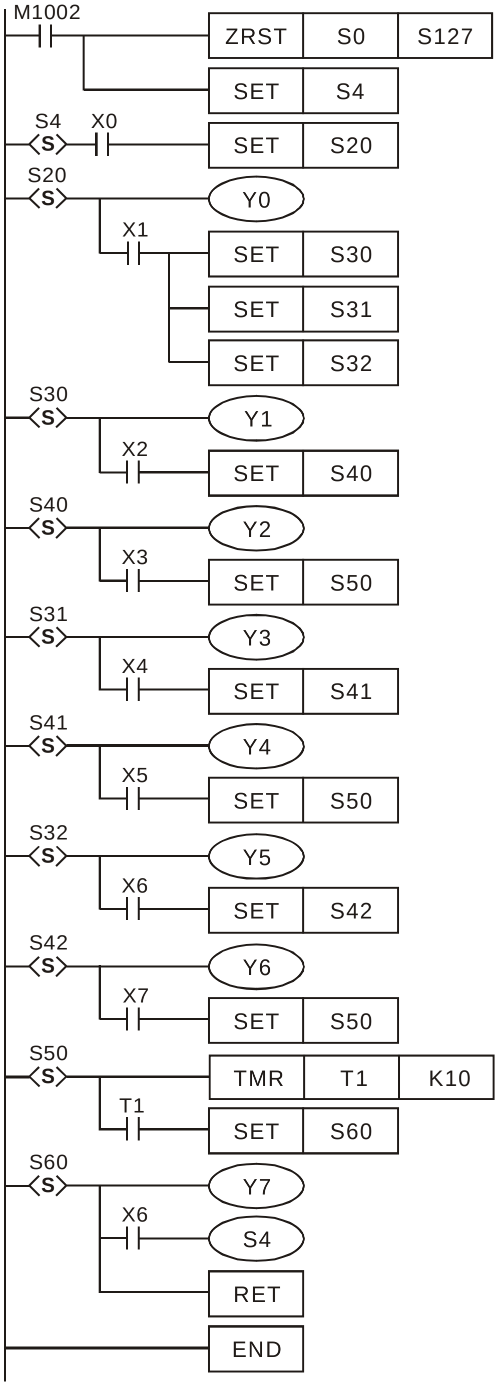


SFC Diagram:

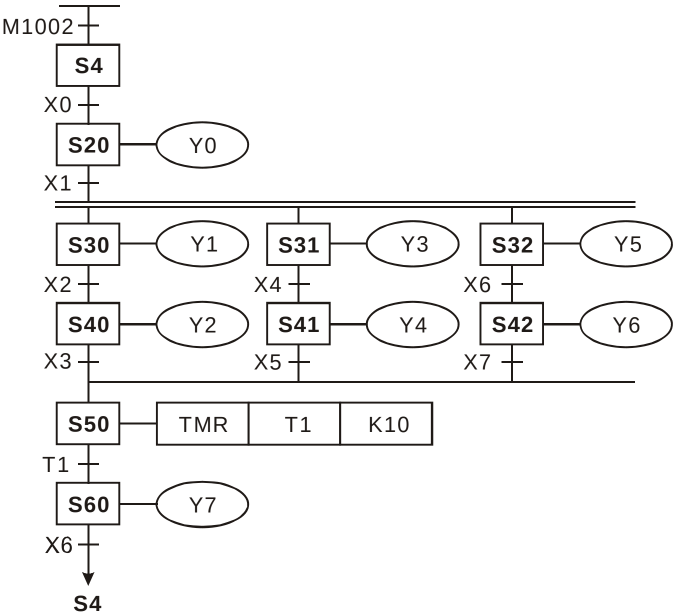


Example of the simultaneous divergence & alternative convergence:

Step Ladder Diagram:

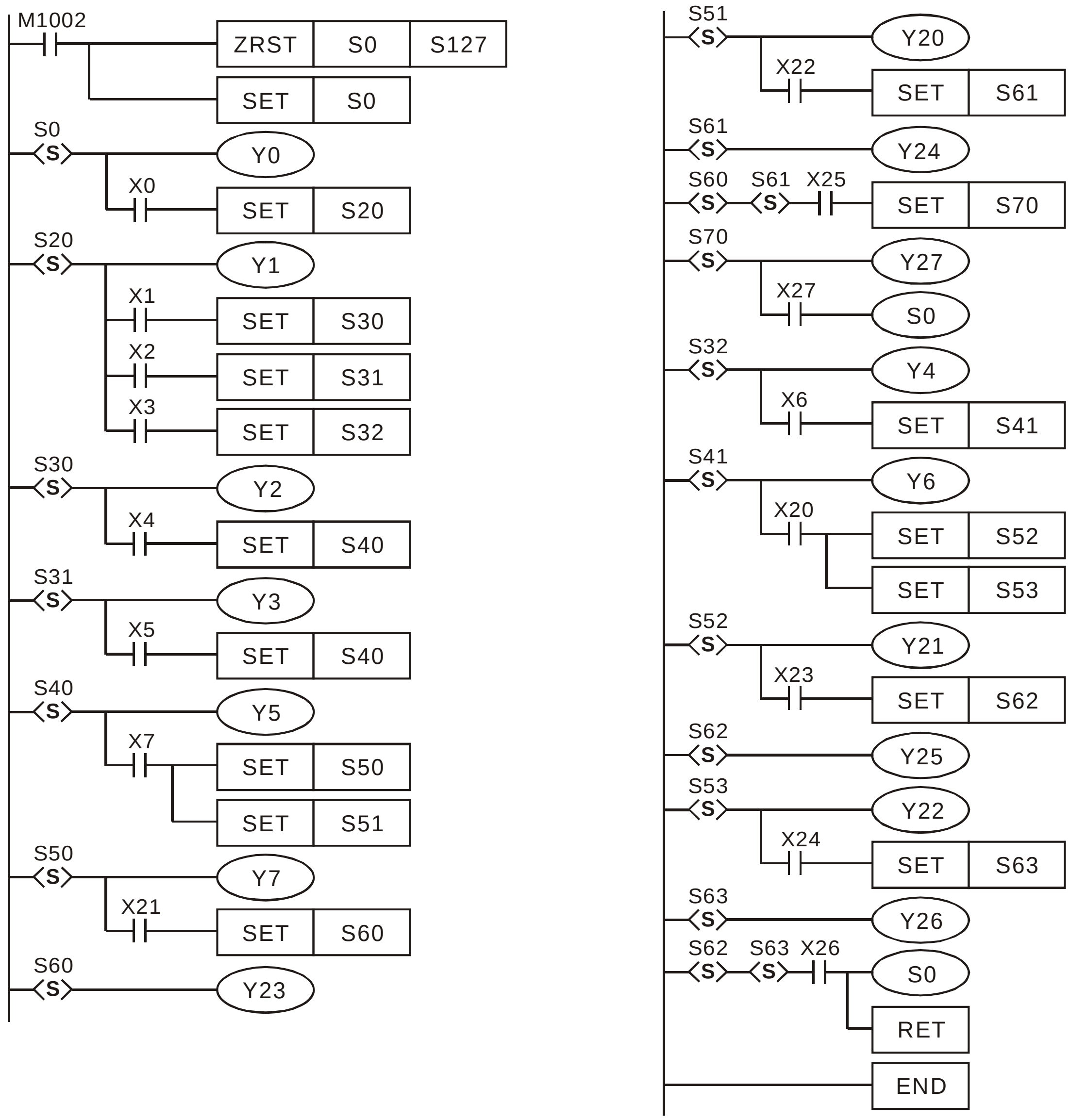


SFC Diagram:

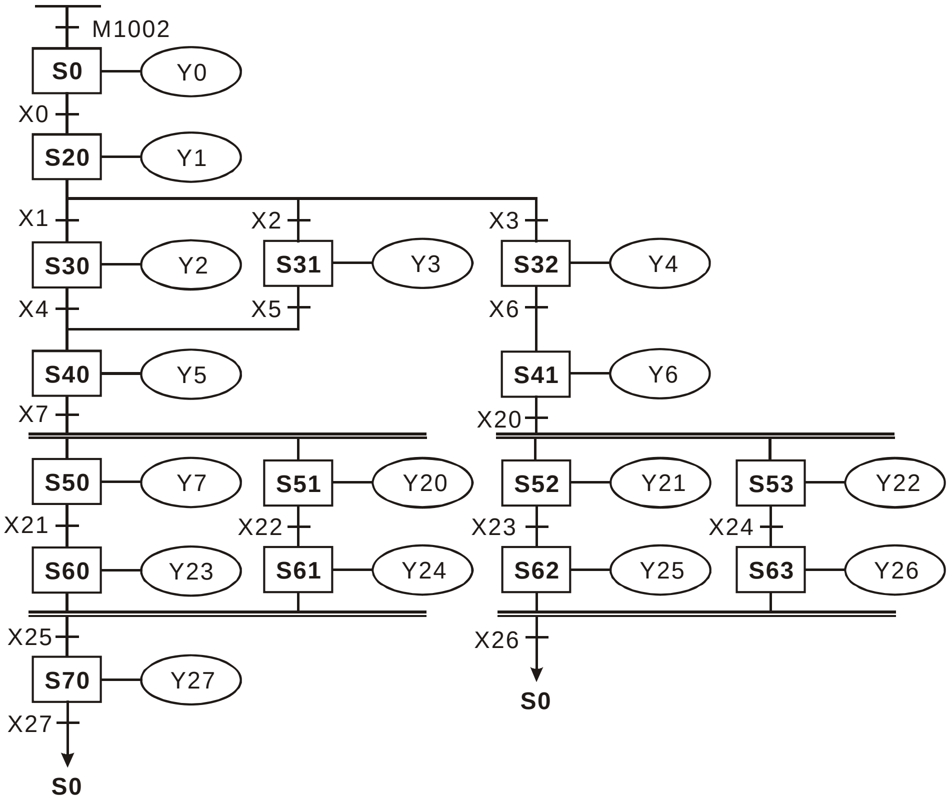


Combination example 1: (Includes alternative divergence/convergence and simultaneous divergence/convergence)

Step Ladder Diagram:

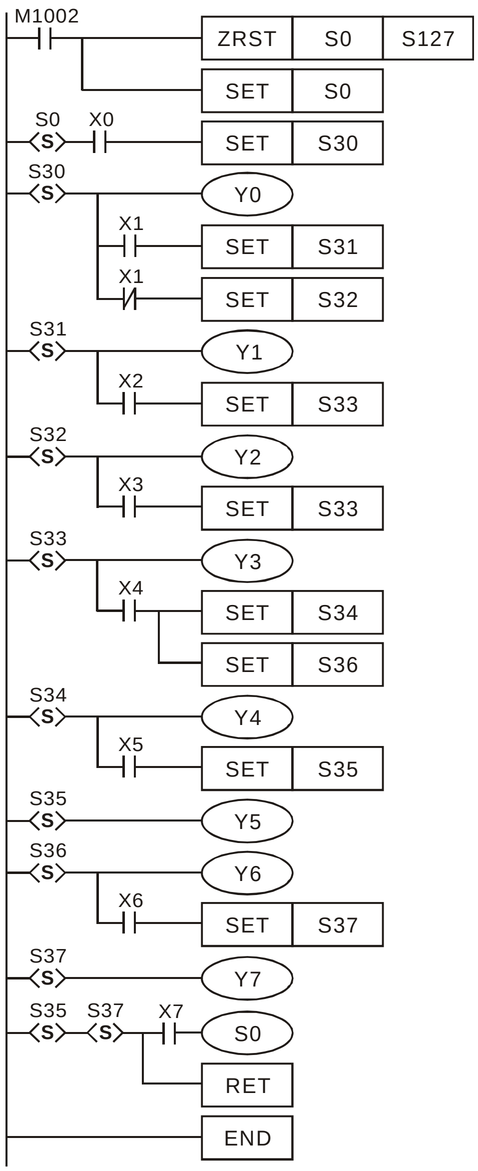


SFC Diagram:

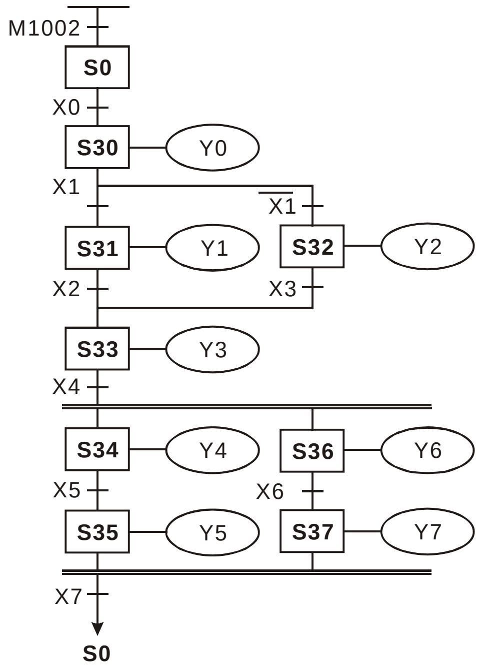

M1002
S0
Y0
X0
S20
Y1
X1
X2
X3
S30
Y2
S31
Y3
S32
Y4
X4
X5
X6
S40
Y5
S41
Y6
X7
X20
S50
Y7
S51
Y20
S52
Y21
S53
Y22
X21
X22
X23
X24
S60
Y23
S61
Y24
S62
Y25
S63
Y26
X25
X26
S70
Y27
S0
X27
S0

Combination example 2: (Includes alternative divergence/convergence and simultaneous divergence/convergence)

Step Ladder Diagram:



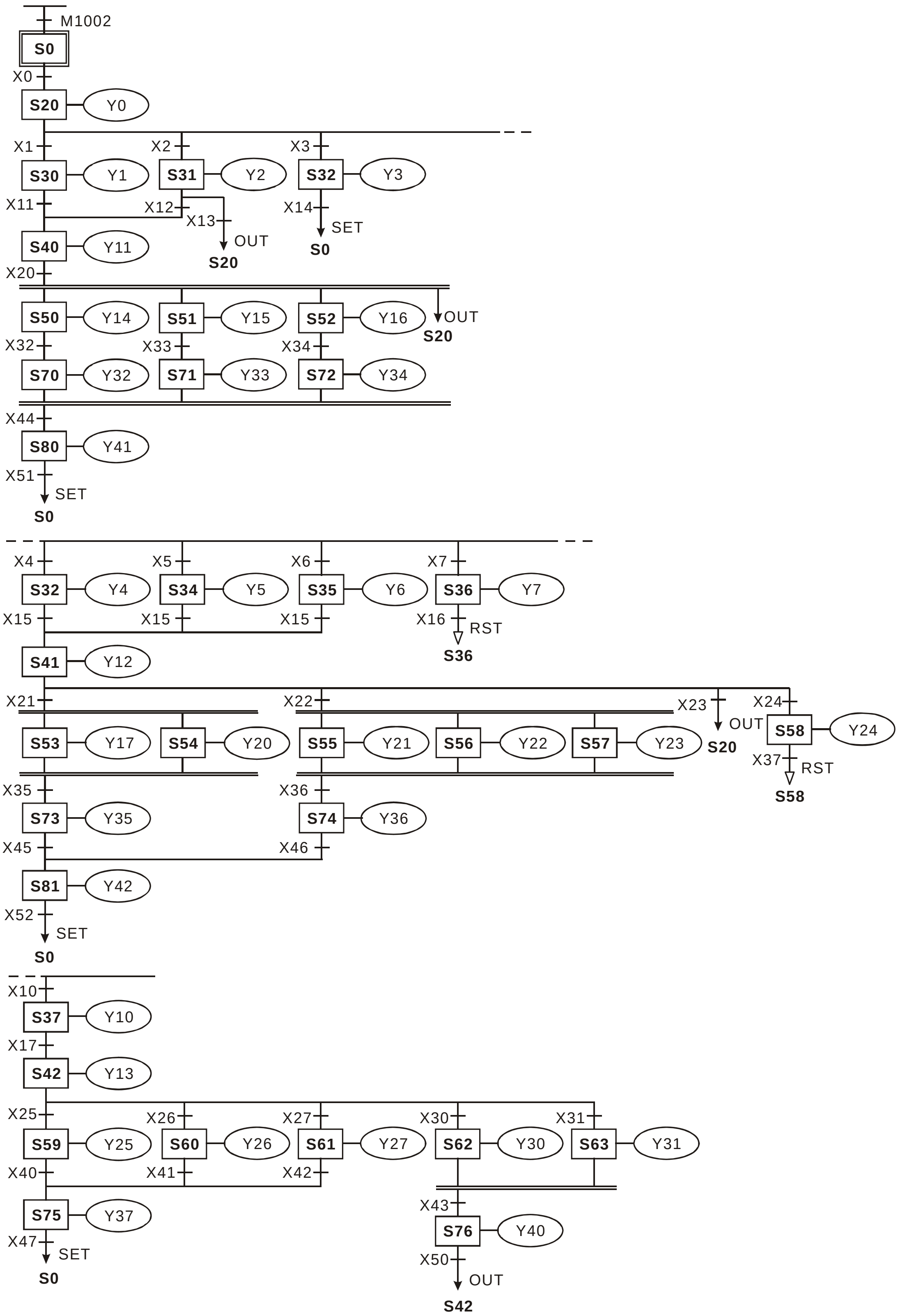
SFC Diagram:



Restrictions on Divergence Sequence:

1. Max. 8 step points could be used for single divergence sequence. As the diagram below, there are maximum 8 diverged steps S30 ~ S37 after step S20.
2. Max. 16 step points could be used for the convergence of multiple diverted sequences. As the diagram below, there are 4 steps diverged after S40, 7 steps diverged after S41, and 5 steps diverged after S42. There are maximum 16 loops in this sequence.
3. Users can assign a step in the sequence to jump to any step in another sequence.

SFC Diagram:


M1002
S0
X0
S20
Y0
X1
X2
X3
S30
Y1
S31
Y2
S32
Y3
X11
X12
X13
X14
SET
OUT
S40
Y11
S20
S0
X20
S50
Y14
S51
Y15
S52
Y16
OUT
S20
X32
X33
X34
S70
Y32
S71
Y33
S72
Y34
X44
S80
Y41
X51
SET
S0
X4
X5
X6
X7
S32
Y4
S34
Y5
S35
Y6
S36
Y7
X15
X15
X15
X16
RST
S41
Y12
S36
X21
X22
X23
X24
OUT
S53
Y17
S54
Y20
S55
Y21
S56
Y22
S57
Y23
S20
S58
Y24
X37
RST
S58
X35
X36
S73
Y35
S74
Y36
X45
X46
S81
Y42
X52
SET
S0
X10
S37
Y10
X17
S42
Y13
X25
X26
X27
X30
X31
S59
Y25
S60
Y26
S61
Y27
S62
Y30
S63
Y31
X40
X41
X42
S75
Y37
X43
S76
Y40
X47
SET
S0
X50
OUT
S42

5.6 IST Instruction

API	Mnemonic	Operands	Function	Controllers
60	IST	(S) (D1) (D2)	Initial State	ES2/EX2, SS2, SA2, SX2

Type / OP	Bit Devices X	Y	M	S	Word devices K	H	KnX	KnY	KnM	KnS	T	C	D	E	F	Program Steps
S	*	*	*													IST: 7 steps
D_1				*												
D_2				*												

PULSE				16-bit				32-bit			
ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2	ES2/EX2	SS2	SA2	SX2

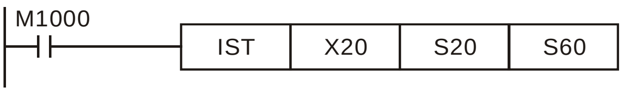
Operands:

S: Source device for assigning pre-defined operation modes (8 consecutive devices). **D_1** The smallest No. of step points in auto mode. **D_2**: The greatest No. of step points in auto mode.

Explanations:

1. The IST is a handy instruction specifically for the initial state of the step ladder operation modes.
2. The range of **D_1** and **D_2** : S20~S911, **D_1** < **D_2**.
3. IST instruction can only be used one time in a program.

Program Example 1:



S: X20: Individual operation (Manual operation)
X21: Zero return
X22: Step operation
X23: One cycle operation
X24: Continuous operation
X25: Zero return start switch
X26: Start switch
X27: Stop switch

1. When IST instruction is executed, the following special auxiliary relays will be assigned automatically.
   M1040: Movement inhibited
   M1041: Movement start
   M1042: Status pulse
   M1047: STL monitor enable
   S0: Manual operation/initial state step point
   S1: Zero point return/initial state step point
   S2: Auto operation/initial state step point

2. When IST instruction is used, S10~S19 are occupied for zero point return operation and cannot be used as a general step point. In addition, when S0~S9 are in use, S0 initiates

"manual operation mode", S1 initiates "zero return mode" and S2 initiates "auto mode". Thus, the three step points of initial state have to be programmed in first priority.

3. When S1 (zero return mode) is initialized, i.e. selected, zero return will NOT be executed if any of the state S10~S19 is ON.
4. When S2 (auto mode) is initialized, i.e. selected, auto mode will NOT be executed if M1043 = ON or any of the state between $\mathbf{D_1}$ to $\mathbf{D_{2l}}$ is ON.

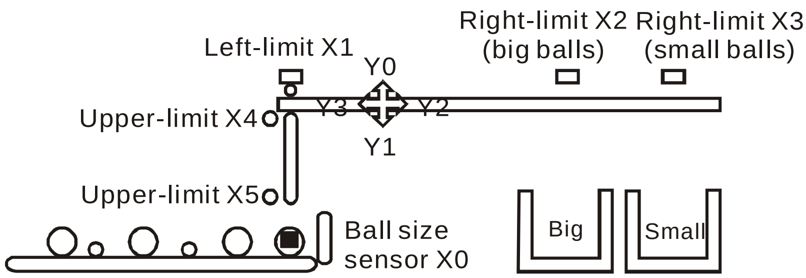
Program Example 2:

Robot arm control (by IST instruction):

1. Control purpose:

   Select the big balls and small balls and move them to corresponding boxes. Configure the control panel for each operation.
2. Motion of the Robot arm:

   lower robot arm, clip balls, raise robot arm, shift to right, lower robot arm, release balls, raise robot arm, shift to left to finish the operation cycle.
3. I/O Devices



4. Operation mode:

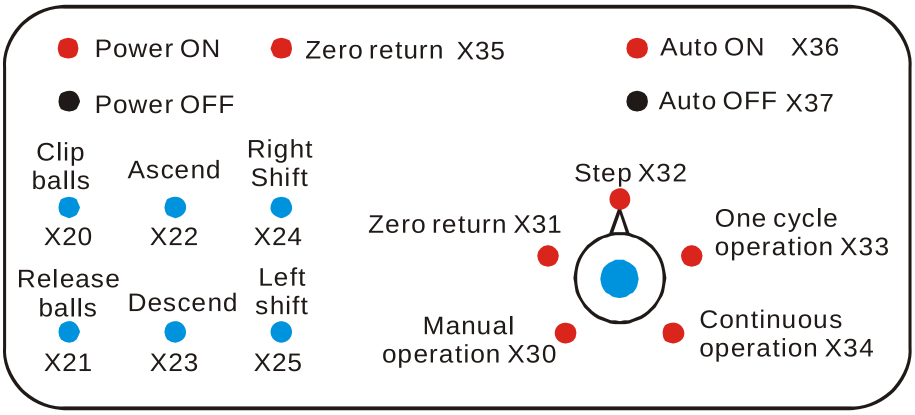
   Single step: Press single button for single step to control the ON/OFF of external load.

   Zero return: Press zero return button to perform homing on the machine.

   Auto (Single step / One cycle operation / Continuous operation):

   - Single step: the operation proceeds with one step every time when Auto ON is pressed.
   - One cycle operation: press Auto ON at zero position, the operation performs one full cycle operation and stops at zero point. If Auto OFF is pressed during the cycle, the operation will pause. If Auto ON is pressed again, the operation will resume the cycle and stop at zero point.
   - Continuous operation: press Auto ON at zero position, the operation will perform continuous operation cycles. If Auto OFF is pressed, the operation will stop at the end of the current cycle.

5. Control panel

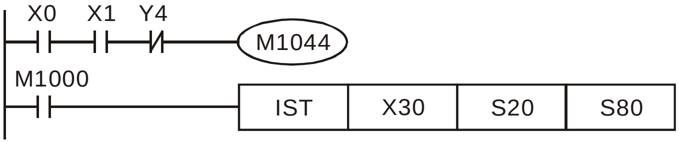


a) X0: ball size sensor.

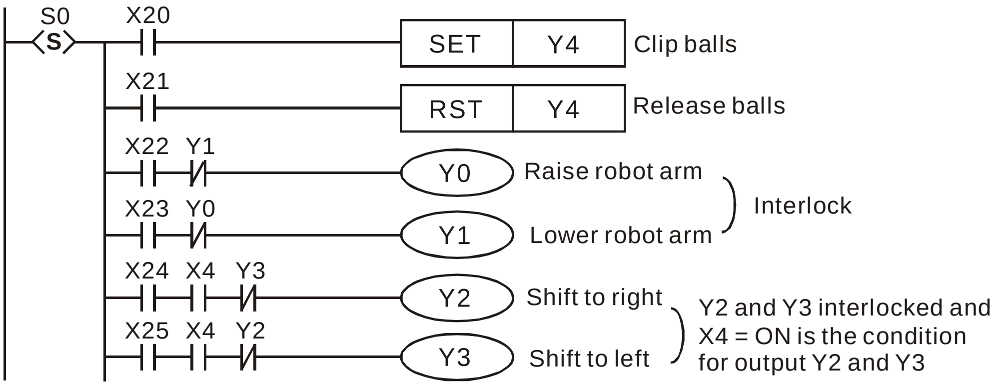
b) X1: left-limit of robot arm, X2: right-limit (big balls), X3: right-limit (small balls), X4: upper-limit of clamp, X5: lower-limit of clamp.

c) Y0: raise robot arm, Y1: lower robot arm, Y2: shift to right, Y3: shift to left, Y4: clip balls.

6. START circuit:

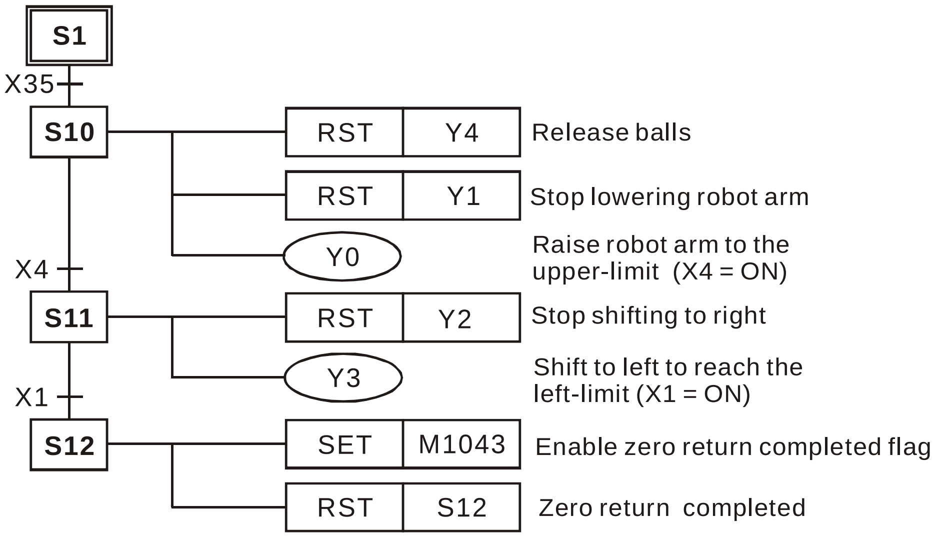


7. Manual mode:

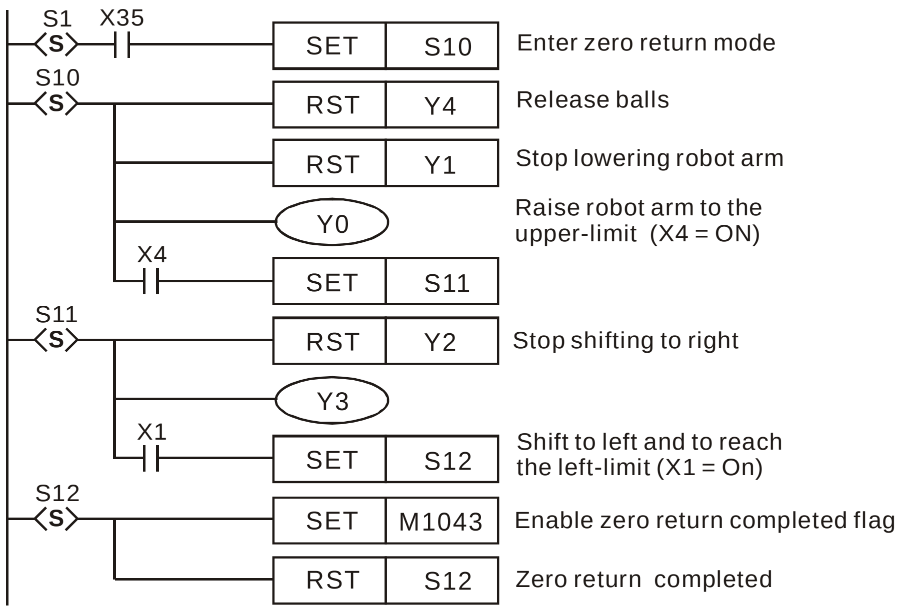


8. Zero return mode:

a) SFC:

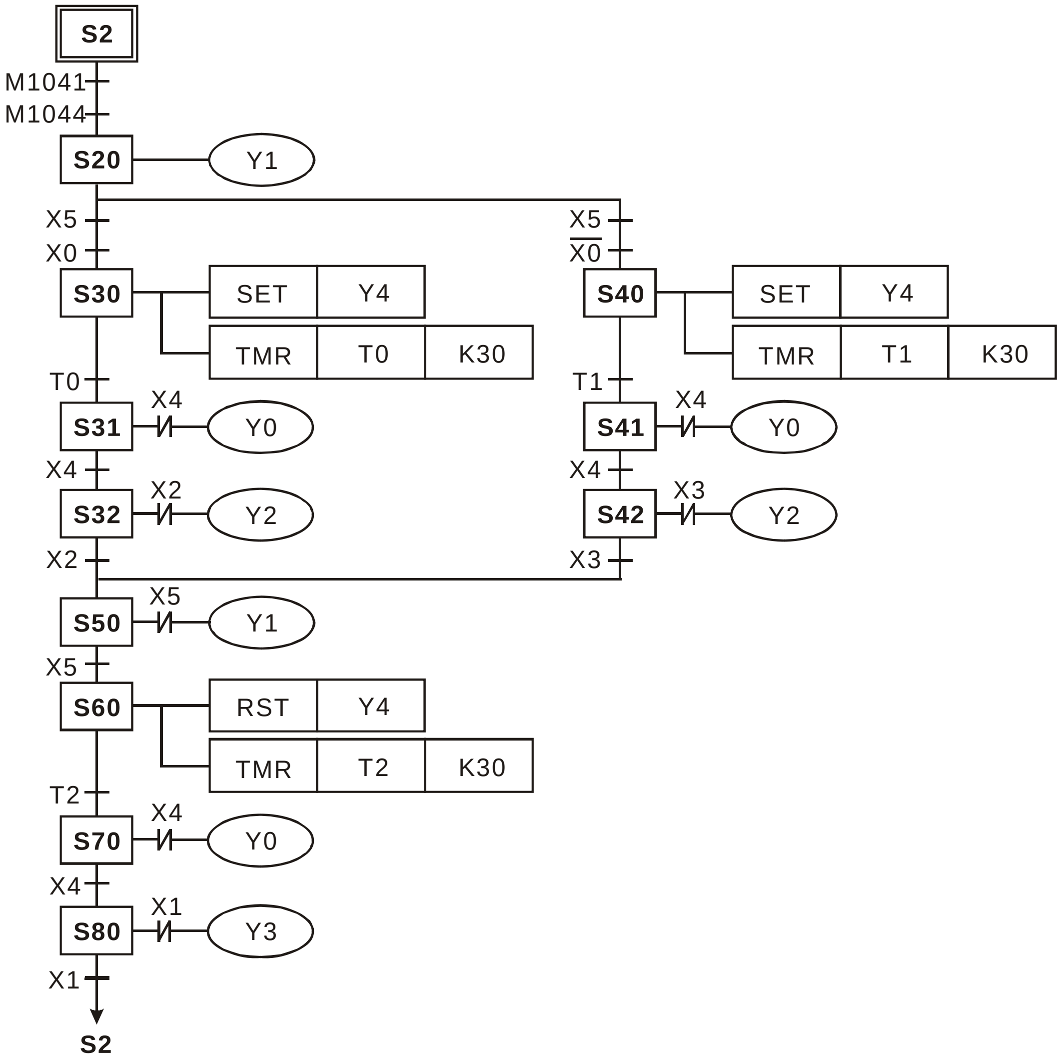


b) Ladder Diagram:

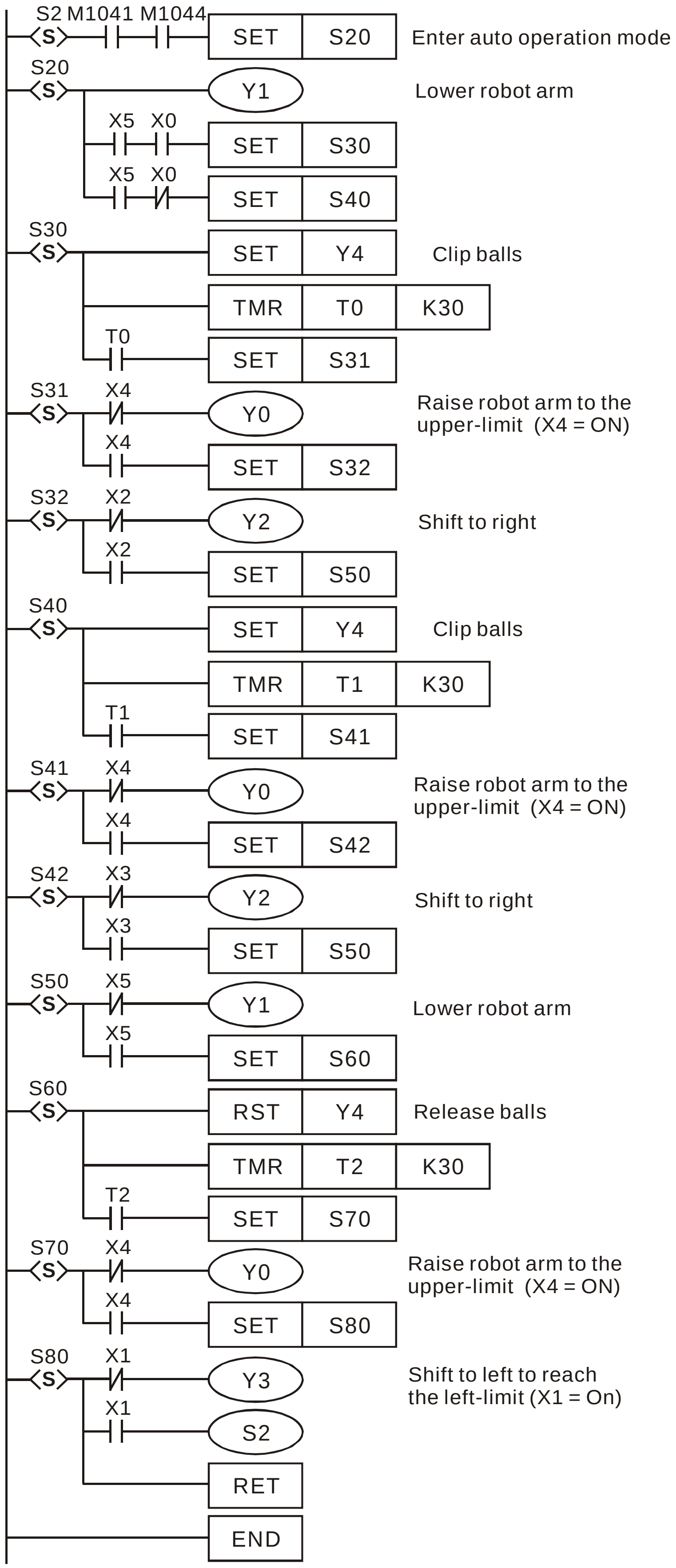


9. Auto operation (Single step / One-cycle operation / continuous operation):

a) SFC:



b) Ladder Diagram:



MEMO

Troubleshooting 6

This chapter offers error code table and information for troubleshooting during PLC operation.

Chapter Contents

6.1 Common Problems and Solutions

The following tables list some common problems and troubleshooting procedures for the PLC system in the event of faulty operation.

System Operation

Symptom	Troubleshooting and Corrective Actions
All LEDs are OFF	1. Check the power supply wiring. 2. Check If the power supplied to the PLC control units is in the range of the rating. 3. Be sure to check the fluctuation in the power supply. 4. Disconnect the power supply wiring to the other devices if the power supplied to the PLC control unit is shared with them. If the LEDs on the PLC 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. 5. If the POWER LED still does not light up when the power is on after the above corrective actions, the PLC should be sent back to the dealer or the distributor whom you purchased the product from.
ERROR LED is flashing	1. 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 WPLSoft or HPP. Error codes and error steps are stored in the following special registers. Error code: D1004 Error step: D1137 2. If the connections between the PLC are failed and the LED will flash rapidly, this indicates the DC24V power supply is down and please check for possible DC24V overload. 3. The LED will be steady if the program loop execution time is over the preset time (set in D1000), check the programs or the WDT (Watch Dog Timer). If the LED remains steady, download user program again and then power up to see if the LED will be OFF. If not, please check if there is any noise interference or any foreign object in the PLC.

Symptom	Troubleshooting and Corrective Actions
Diagnosing Input Malfunction	When input indicator LEDs are OFF, 1. Check the wiring of the input devices. 2. Check that the power is properly supplied to the input terminals. 3. If the power is properly supplied to the input terminal, there is probably an abnormality in the PLC’s input circuit. Please contact your dealer. 4. 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. When input indicator LEDs are ON, 1. Monitor the input condition using a programming tool. If the input monitored is OFF, there is probably an abnormality in the PLC’s input circuit. Please contact your dealer. 2. 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. 3. Check the settings of the I/O allocation.
Diagnosing Output Malfunction	When output indicator LEDs are ON, 1. Check the wiring of the loads. 2. Check if the power is properly supplied to the loads. 3. If the power is properly supplied to the load, there is probably an abnormality in the load. Check the load again. 4. If the power is not supplied to the load, there is probably an abnormality in the PLC’s output circuit. Pleas contact your dealer. When output indicator LEDs are OFF, 1. Monitor the output condition using a programming tool. If the output monitored is turned ON, there is probably a duplicated output error. 2. Forcing ON the output using a programming tool. If the output indicator LED is turned ON, go to input condition check. If the output LED remains OFF, there is probably an abnormality in the PLC’s output circuit. Please contact your dealer.

6.2 Error code Table (Hex)

After you write the program into the PLC, the illegal use of operands (devices) or incorrect syntax in the program will result in flashing of ERROR indicator and M1004 = ON. In this case, you can find out the cause of the error by checking the error code (hex) in special register D1004. The address where the error occurs is stored in the data register D1137. If the error is a general loop error, the address stored in D1137 will be invalid.

Error code	Description	Action
0001	Operand bit device S exceeds the valid range	Check 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	
0B01	Operand K, H available range error	
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 operand misuse	
0D0A	DHSZ operand misuse	
0D0B	IST operand misuse	
0E01	Operand bit device C exceeds the valid range	
0E04	Operand word device C register exceeds limit	

Error code	Description	Action
0E05	DCNT operand CXXX misuse	Check the D1137 (Error step number) Re-enter the instruction correctly
0E18	BCD conversion error	
0E19	Division error (divisor=0)	
0E1A	Device use is out of range (including index registers E, F)	
0E1B	Negative number after radical expression	
0E1C	FROM/TO communication error	
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	
0F09	Improper use of operands of WSFR, WSFL instructions	
0F0A	Times of using TTMR, STMR instruction exceed the range	
0F0B	Times of using SORT instruction exceed the range	
0F0C	Times of using TKY instruction exceed the range	
0F0D	Times of using HKY instruction exceed the range	
1000	ZRST operand misuse	
10EF	E and F misuse operand or exceed the usage range	
2000	Usage exceed limit (MTR, ARWS, TTMR, PR, HOUR)	

Error code	Description	Action
C400	An unrecognized instruction code is being used	A circuit error occurs if a combination of instructions is incorrectly 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 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 or interrupt program	
C40A	Use MC/MCR in subroutine Use MC/MCR in interrupt program	

Error code	Description	Action
C40B	MC / MCR does not begin from N0 or discontinuously	A circuit error occurs if a combination of instructions is incorrectly specified. Select programming mode and correct the identified error
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	
C40F	PLC program and data in parameters have not been initialized	
C41B	Invalid RUN/STOP instruction to extension module	
C41C	The number of input/output points of I/O extension unit is larger than the specified limit	
C41D	Number of extension modules exceeds the range	
C41F	Failing to write data into memory	
C430	Initializing parallel interface error	
C440	Hardware error in high-speed counter	
C441	Hardware error in high-speed comparator	
C442	Hardware error in MCU pulse output	
C443	No response from extension unit	
C4EE	No END command in the program	
C4FF	Invalid instruction (no such instruction existing)	

6.3 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 error (divisor=0)
0E1A	Floating point exceeds the usage range
0E1B	The value of square root is negative

MEMO