

# Chapter 1 PLC Ladder Diagram and the Coding Rules of Mnemonic

In this chapter, we would like to introduce you the basic principles of ladder diagram, in addition, the coding rules of mnemonic will be introduced as well, it's essential for the user who use FP-07 as a programming tool. If you are familiar with PLC Ladder Diagram and mnemonic coding rules, you may skip this chapter.

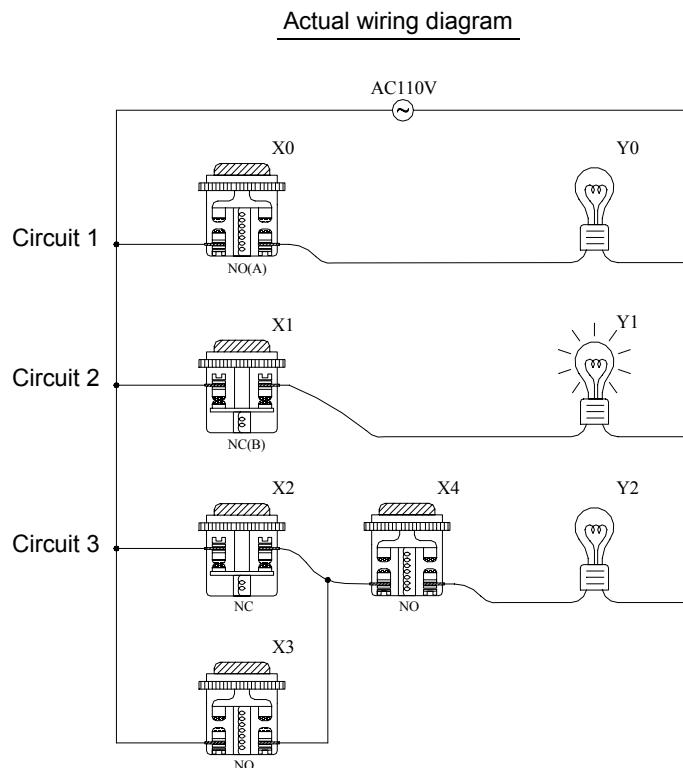
## 1.1 The Operation Principle of Ladder Diagram

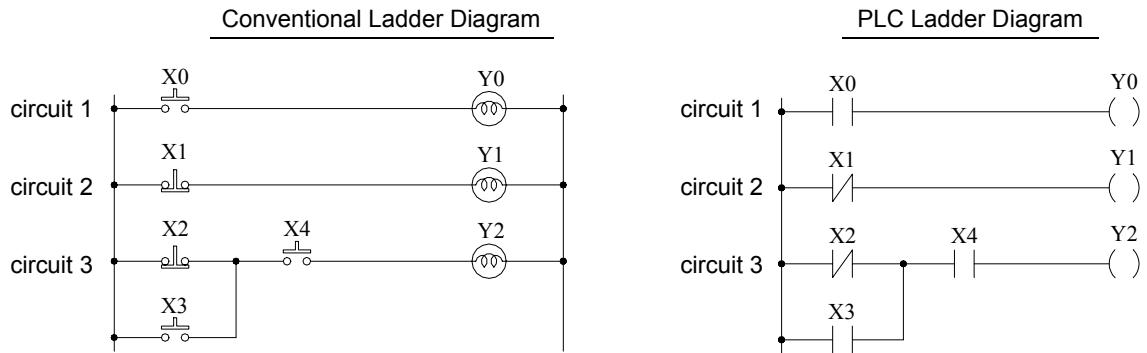
Ladder Diagram is a type of graphic language for automatic control systems it had been used for a long period since World War II. Until today, it is the oldest and most popular language for automatic control systems. Originally there are only few basic elements available such as A-contact (Normally ON), B contact (Normally OFF), output Coil, Timers and Counters. Not until the appearance of microprocessor based PLC, more elements for Ladder Diagram, such as differential contact, retentive coil (refer to page 1-6) and other instructions that a conventional system cannot provide, became available.

The basic operation principle for both conventional and PLC Ladder Diagram is the same. The main difference between the two systems is that the appearance of the symbols for conventional Ladder Diagram are more closer to the real devices, while for PLC system, symbols are simplified for computer display. There are two types of logic system available for Ladder Diagram logic, namely combination logic and sequential logic. Detailed explanations for these two logics are discussed below.

### 1.1.1 Combination Logic

Combination logic of the Ladder Diagram is a circuit that combines one or more input elements in series or parallel and then send the results to the output elements, such as Coils, Timers/Counters, and other application instructions.



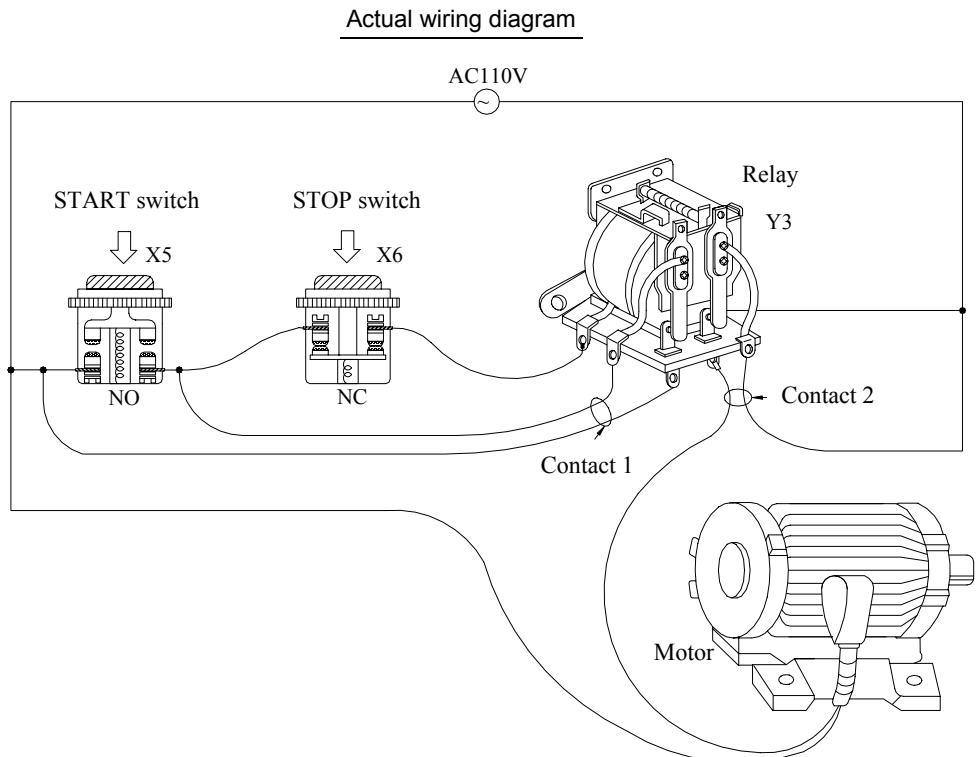


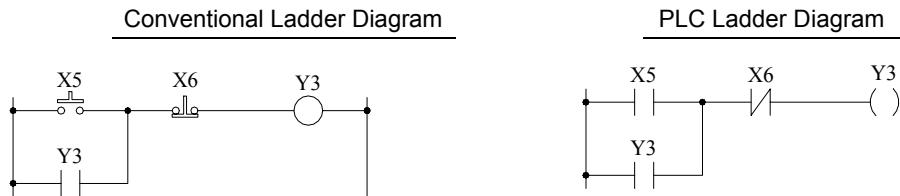
The above example illustrated the combination logic using the actual wiring diagram, conventional Ladder Diagram, and PLC Ladder Diagram. Circuit 1 uses a NO (Normally Open) switch that is also called "A" switch or contact. Under normal condition (switch is not pressed), the switch contact is at OFF state and the light is off. If the switch is pressed, the contact status turns ON and the light is on. In contrast, circuit 2 uses a NC (Normally Close) switch that is also called "B" switch or contact. Under normal condition, the switch contact is at ON state and the light is on. If the switch is pressed, the contact status turns OFF and the light also turns off.

Circuit 3 contains more than one input element. Output Y2 light will turn on under the condition when X2 is closed or X4 must switch ON too.

### 1.1.2 Sequential Logic

The sequential logic is a circuit with feedback control; that is, the output of the circuit will be feedback as an input to the same circuit. The output result remains in the same state even if the input condition changes to the original position. This process can be best explained by the ON/OFF circuit of a latched motor driver as shown in below.





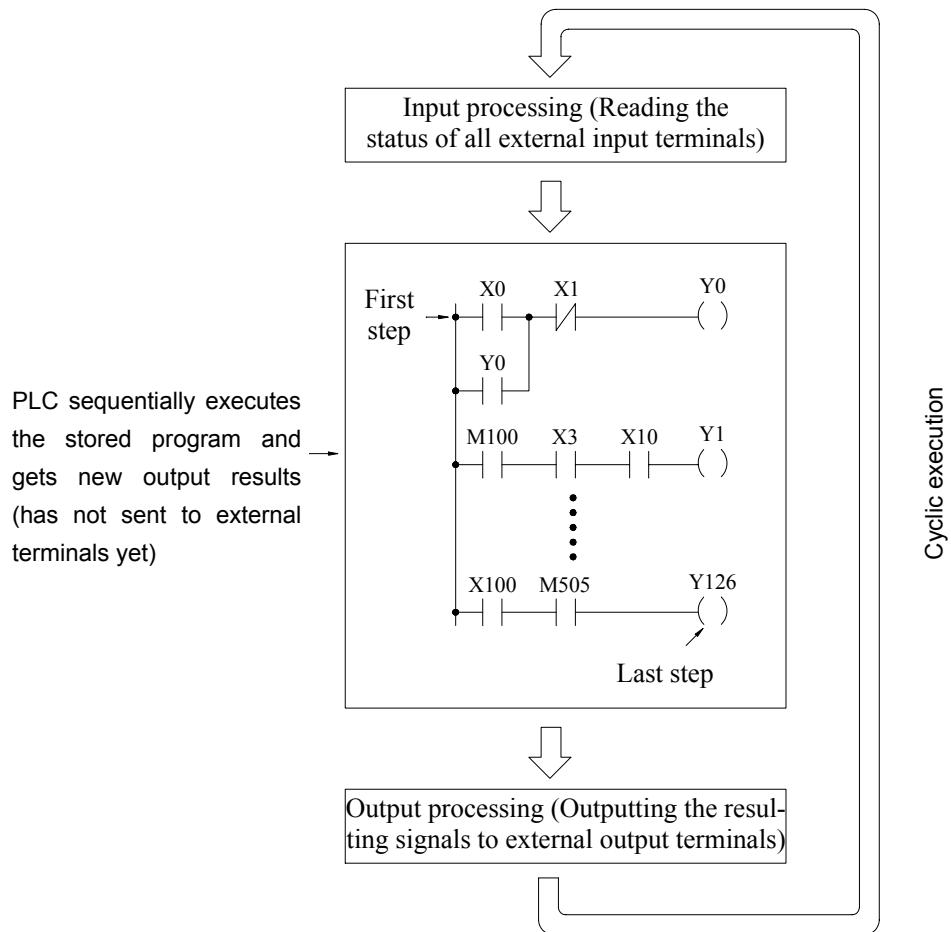
When we first connect this circuit to the power source, X6 switch is ON but X5 switch is OFF, therefore the relay Y3 is OFF. The relay output contacts 1and 2 are OFF because they belong to A contact (ON when relay is ON). Motor does not run. If we press down the switch X5, the relay turns ON as well as contacts 1and 2 are ON and the Motor starts. Once the relay turns ON, if we release the X5 switch (turns OFF), relay can retain its state with the feedback support from contact 1 and it is called Latch Circuit. The following table shows the switching process of the example we have discussed above.

	X5 switch (NO)	X6 switch (NC)	Motor (Relay) status
①	Released	Released	OFF
↓			
②	Pressed	Released	ON
↓			
③	Released	Released	ON
↓			
④	Released	Pressed	OFF
↓			
⑤	Released	Released	OFF

From the above table we can see that under different stages of sequence, the results can be different even the input statuses are the same. For example, let's take a look at stage ① and stage ③ , X5 and X6 switches are both released, but the Motor is ON (running) at stage ③ and is OFF (stopped) at stage ① . This sequential control with the feedback of the output to the input is a unique characteristic of Ladder Diagram circuit. Sometimes we call the Ladder Diagram a "Sequential Control Circuit" and the PLC a "Sequencer". In this section, we only use the A/B contacts and output coils as the example. For more details on sequential instructions please refer to chapter 5 - "Introduction to Sequential Instructions."

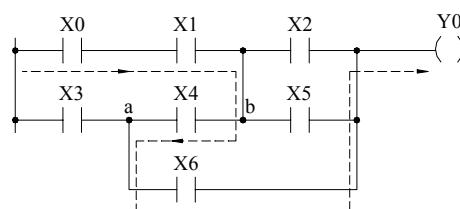
## 1.2 Differences Between Conventional and PLC Ladder Diagram

Although the basic operation principle for both conventional and PLC Ladder Diagram are the same, but in reality, PLC uses the CPU to emulate the conventional Ladder Diagram operations; that is, PLC uses scanning method to monitor the statuses of input elements and output coils, then uses the Ladder Diagram program to emulate the results which are the same as the results produced by the conventional Ladder Diagram logic operations. There is only one CPU, so the PLC has to sequentially examine and execute the program from its first step to the last step, then returns to the first step again and repeats the operation (cyclic execution). The duration of a single cycle of this operation is called the scan time. The scan time varies with the program size. If the scan time is too long, then input and output delay will occur. Longer delay time may cause big problems in controlling fast response systems. At this time, PLCs with short scan time are required. Therefore, scan time is an important specification for PLCs. Due to the advance in microcomputer and ASIC technologies nowadays the scan speed has been enhanced a great deal. A typical FB<sub>E</sub>-PLC takes approximately 0.33 ms for 1K steps of contact. The following diagram illustrates the scanning process of a PLC Ladder Diagram.

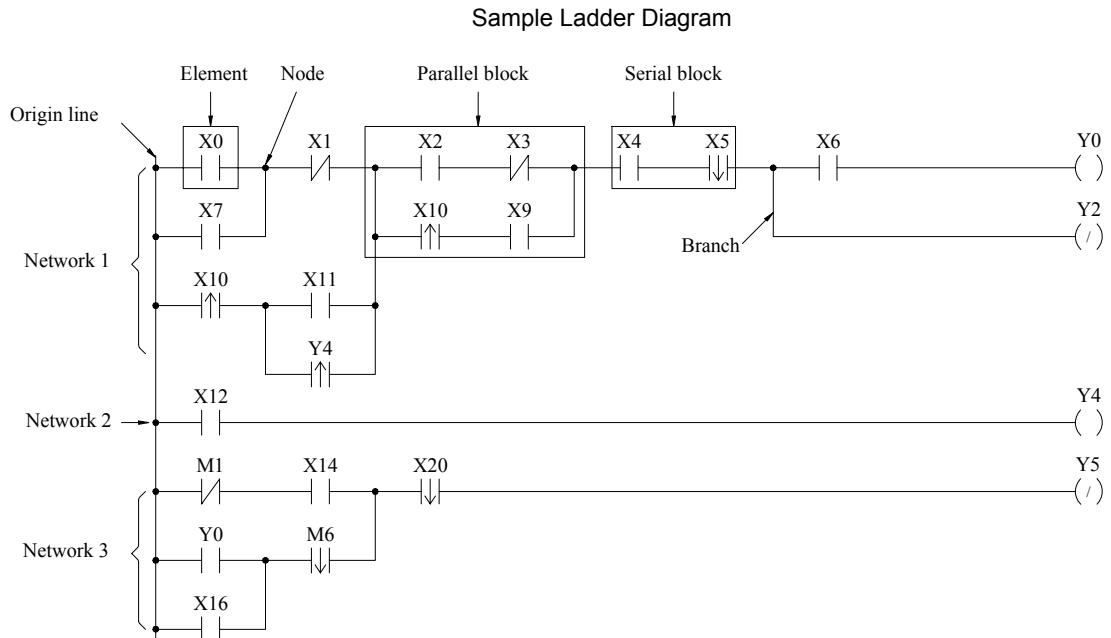


Besides the time scan difference mentioned above, the other difference between the conventional and PLC Ladder Diagram is “Reverse flow” characteristic. As shown in the diagram below, if X0, X1, X4 and X6 are ON, and the remaining elements are OFF. In a conventional Ladder Diagram circuit, a reverse flow route for output Y0 can be defined by the dashed line and Y0 will be ON. While for PLC, Y0 is OFF because the PLC Ladder Diagram scans from left to right, if X3 is off then CPU believes node “a” is OFF, although X4 and node “b” are all ON, since the PLC scan reaches X3 first. In other words, the PLC ladder can only allow left to right signal flow while conventional ladder can flow bi-directional.

#### Reverse flow of conventional Ladder diagram



### 1.3 Ladder Diagram Structure and Terminology



(Remark : The maximum size of FB-PLC network is 16 rows  $\times$  22 columns)

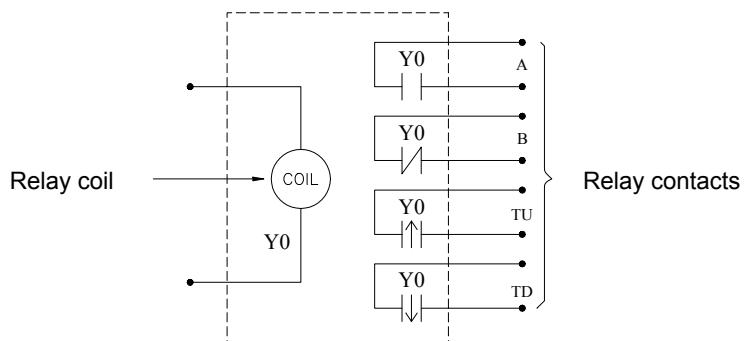
As shown above, the Ladder Diagram can be divided into many small cells. There are total 88 cells (8 rows  $\times$  11 columns) for this example Ladder Diagram. One cell can accommodate one element. A completed Ladder Diagram can be formed by connecting all the cells together according to the specific requirements. The terminologies related to Ladder Diagram are illustrated below.

#### ① Contact

Contact is an element with open or short status. One kind of contact is called "Input contact"(reference number prefix with X) and its status reference from the external signals (the input signal comes from the input terminal block). Another one is called "Relay contact" and its status reflects the status of relay coil (please refer to ②). The relation between the reference number and the contact status depends on the contact type. The contact elements provided by FB series PLC include: A contact, B contact, up/down differential (TU/TD) contacts and Open/Short contacts. Please refer to ④ for more details.

#### ② Relay

Same as the conventional relay, it consists of a Coil and a Contact as shown in the diagram below.



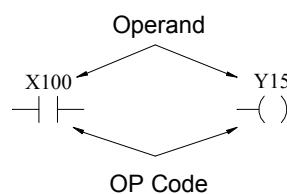
We must energize the coil of relay first (using OUT instruction) in order to turn on the relay. After the coil is energized, its contact status will be ON too. As shown in the example above, if Y0 turns ON, then the relay contact A is ON and contact B is OFF, TU contact only turns ON for one scan duration and TD contact is OFF. If Y0 turns OFF, then the relay contact A is ON and contact B is ON, TU contact is OFF and TD contact only turns ON for one scan duration (Please refer to chapter 5 "Introduction to Sequential Instructions" for operations of A,B,TU and TD contacts).

There are four types of FB-PLC relays, namely  $\text{Y} \triangle \triangle \triangle$  (output relay),  $\text{M} \triangle \triangle \triangle \triangle$  (internal relay),  $\text{S} \triangle \triangle \triangle$  (step relay) and  $\text{TR} \triangle \triangle$  (temporary relay). The statuses of output relays will be sent to the output terminal block.

③ Origin-line: The starting line at the left side of the Ladder Diagram.

④ Element: Element is the basic unit of a Ladder Diagram. An element consists of two parts as shown in the diagram below.

One is the element symbol which is called "OP Code" and another is the reference number part which is called "Operand".



Element type	Symbol	Mnemonic instructions	Remark
A Contact (Normally OPEN)	$\square \triangle \triangle \triangle \triangle$ $\text{---}   \text{---}$	(ORG、LD、AND、OR) $\square \triangle \triangle \triangle \triangle$	<input type="checkbox"/> can be X、Y、M、S、T、C (please refer to section 3.2)
B Contact (Normally CLOSE)	$\square \triangle \triangle \triangle \triangle$ $\text{---} / \text{---}$	(ORG、LD、AND、OR) NOT $\square \triangle \triangle \triangle \triangle$	
Up Differential Contact	$\square \triangle \triangle \triangle \triangle$ $\text{---} \uparrow \text{---}$	(ORG、LD、AND、OR) TU $\square \triangle \triangle \triangle \triangle$	
Down Differential Contact	$\square \triangle \triangle \triangle \triangle$ $\text{---} \downarrow \text{---}$	(ORG、LD、AND、OR) TD $\square \triangle \triangle \triangle \triangle$	<input type="checkbox"/> can be X、Y、M、S
Open Circuit Contact	$\text{---} \circ \text{---}$	(ORG、LD、AND、OR) OPEN	
Short Circuit Contact	$\bullet \text{---} \bullet$	(ORG、LD、AND、OR) SHORT	
Output Coil	$\square \triangle \triangle \triangle \triangle$ $\text{---} ( \text{---} )$	OUT $\square \triangle \triangle \triangle \triangle$	
Inverse Output Coil	$\square \triangle \triangle \triangle \triangle$ $\text{---} ( / \text{---} )$	OUT NOT $\square \triangle \triangle \triangle \triangle$	<input type="checkbox"/> can be Y、M、S
Latching Output Coil	$\text{Y} \triangle \triangle \triangle$ $\text{---} ( \text{---} \text{L} )$	OUT L $\text{Y} \triangle \triangle \triangle$	

Remark : please refer to section 3.2 for the ranges of X、Y、M、S、T and C contacts. Please refer to section 5.2 for the characteristics of X、Y、M、S、T and C contacts.

There are three special sequential instructions, namely OUT TRn, LD TRn and FOn, which were not displayed on the Ladder Diagram. Please refer to section 1.6 "Using the Temporary Relay" and section 6.1.4 "Function Output FO".

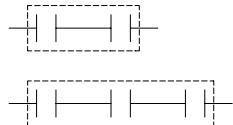
⑤ Node: The connection point between two or more elements (please refer to section 5.3)

⑥ Block: a circuit consists of two or more elements.

There are two basic types of blocks:

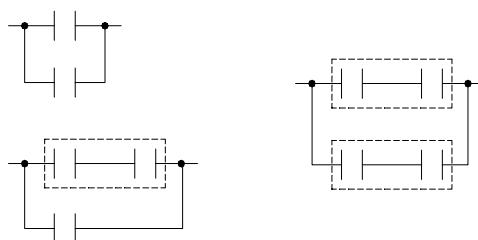
- Serial block : Two or more elements are connected in series to form a single row circuit.

Example:



- Parallel block: Parallel block is a type of a parallel closed circuit formed by connecting elements or serial blocks in parallel.

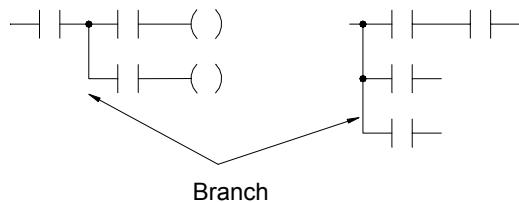
Example:



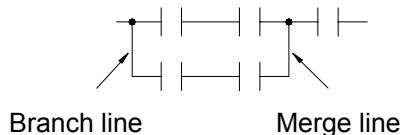
Remark: Complicated block can be formed by the combination of the single element, serial blocks and parallel blocks. When design a Ladder Diagram with mnemonic entry, it is necessary to break down the circuits into element, serial, and parallel blocks. Please refer to section 1.5.

⑦ Branch: In any network, branch is obtained if the right side of a vertical line is connected with two or more rows of circuits.

Example:

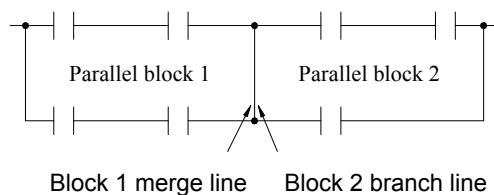


Merge line is defined as another vertical line at the right side of a branch line that merges the branch circuits into a closed circuit (forming a parallel block). This vertical line is called "Merge line".



If both the right and the left sides of the vertical line are connected with two or more rows of circuits, then it is both a branch line and a merge line as shown in the example below.

Example:



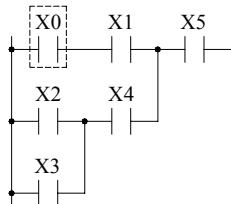
- ⑧ Network: Network is a circuit representing a specified function. It consists of the elements, branches, and blocks. Network is the basic unit in the Ladder Diagram which is capable of executing the completed functions, and the program of Ladder Diagram is formed by connecting networks together. The beginning of the network is the origin line. If two circuits are connected by a vertical line, then they belong to the same network. If there is no vertical line between the two circuits, then they belong to two different networks. Figure 1, shows three (1~3) networks.

#### 1.4 The Coding Rules of Mnemonic (Users of WinProladder can skip this section)

It's very easy to program FB-PLC with WinProladder software package, just key-in the ladder symbols as they appear on your CRT screen directly to form a ladder diagram program. But for the users who are using FP-07 to program FB-PLC they have to translate ladder diagram into mnemonic instructions by themselves. Since FP-07 only can input program with mnemonic instruction, this section till section 1.6 will furnish you with the coding rules to translate ladder diagrams into mnemonic instructions.

- The program editing directions are from left to right and from top to bottom. Therefore the beginning point of the network must be at the upper left corner of the network. Except the function instruction without the input control, the first instruction of a network must begin with the ORG and only one ORG instruction is permissible per network. Please refer to section 6.1.1 for further explanations.

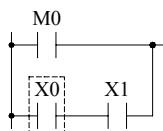
Example:



ORG	X	0
AND	X	1
LD	X	2
OR	X	3
AND	X	4
ORLD		
AND	X	5

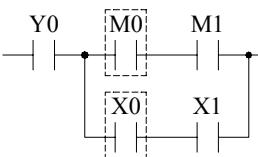
- Using LD instruction for connecting vertical lines (origin line or branch line) except at the beginning of the network.

Example 1:



ORG	M	0
LD	X	0
AND	X	1
ORLD		

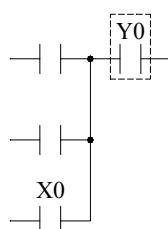
Example 2:



AND	Y	0
LD	M	0
AND	M	1
LD	X	0
AND	X	1
ORLD		

Remark 1: Using the AND instruction directly if only one row of elements is serially connected to the branch line.

Example:



AND	X	0
ORLD		
AND	Y	0

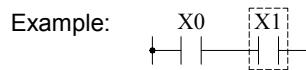
Remark 2: Also using the AND instruction directly if an OUT TR instruction has been used at a branch line to store the node statuses.

Example:

OUT TR0  
LD TR0

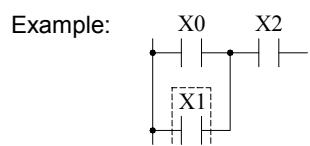
AND M 0  
OUT TR 0  
AND X 0  
OUT Y 1  
LD TR 0  
AND Y 0

- Using AND instruction for serial connection of a single element.

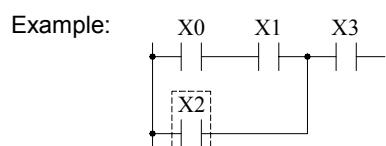
Example: 

⇒ ORG X 0  
AND X 1

- Using OR instruction for parallel connection of a single element.

Example: 

⇒ ORG X 0  
OR X 1  
AND X 2

Example: 

⇒ ORG X 0  
AND X 1  
OR X 2  
AND X 3

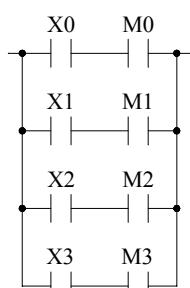
- If the parallel element is a serial block, ORLD instruction must be used.

Example:

⇒ ORG X 2  
LD X 0  
AND X 1  
ORLD  
AND X 3

Remark : If more than two blocks are to be connected in parallel, they should be connected in a top to bottom sequence. For example, block 1 and block 2 should be connected first, then connect block 3 to it and so on.

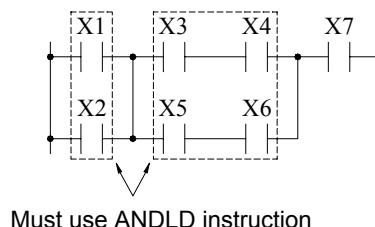
Example:



LD X 0  
AND M 0  
LD X 1  
AND M 1  
ORLD  
LD X 2  
AND M 2  
ORLD  
LD X 3  
AND M 3

- ANDLD instruction is used to connect parallel blocks in series.

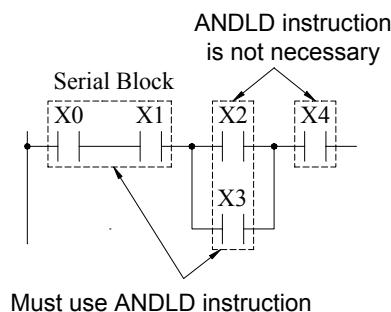
Example:



ORG	X	1
OR	X	2
LD	X	3
AND	X	4
LD	X	5
AND	X	6
ORLD		
ANDLD		
AND	X	7

- The ANDLD instruction must be used if the element or serial block is in front of the parallel block. If the parallel block is in front of the element or serial block, AND instruction can be used to connect all parts together.

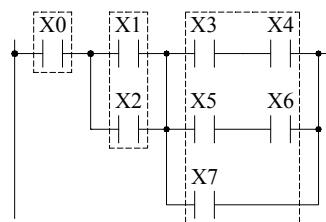
Example:



ORG	X	0
AND	X	1
LD	X	2
OR	X	3
ANDLD		
AND	X	4

Remark: If there are more than two blocks are to be connected serially, they should be connected in a top to bottom sequence. For example, block 1 and 2 should be connected first, then connect block 3 to it and so on.

Example:



ORG	X	0
LD	X	1
OR	X	2
ANDLD		
LD	X	3
AND	X	4
LD	X	5
AND	X	6
ORLD		
OR	X	7
ANDLD		

- The output coil instruction (OUT) can only be located at end of the network (the right end) and no other elements can be connected to it afterwards. The output coil can not connect to the origin line directly. If you want to connect the output coil to the origin line, connect it serially with a short circuit contact.



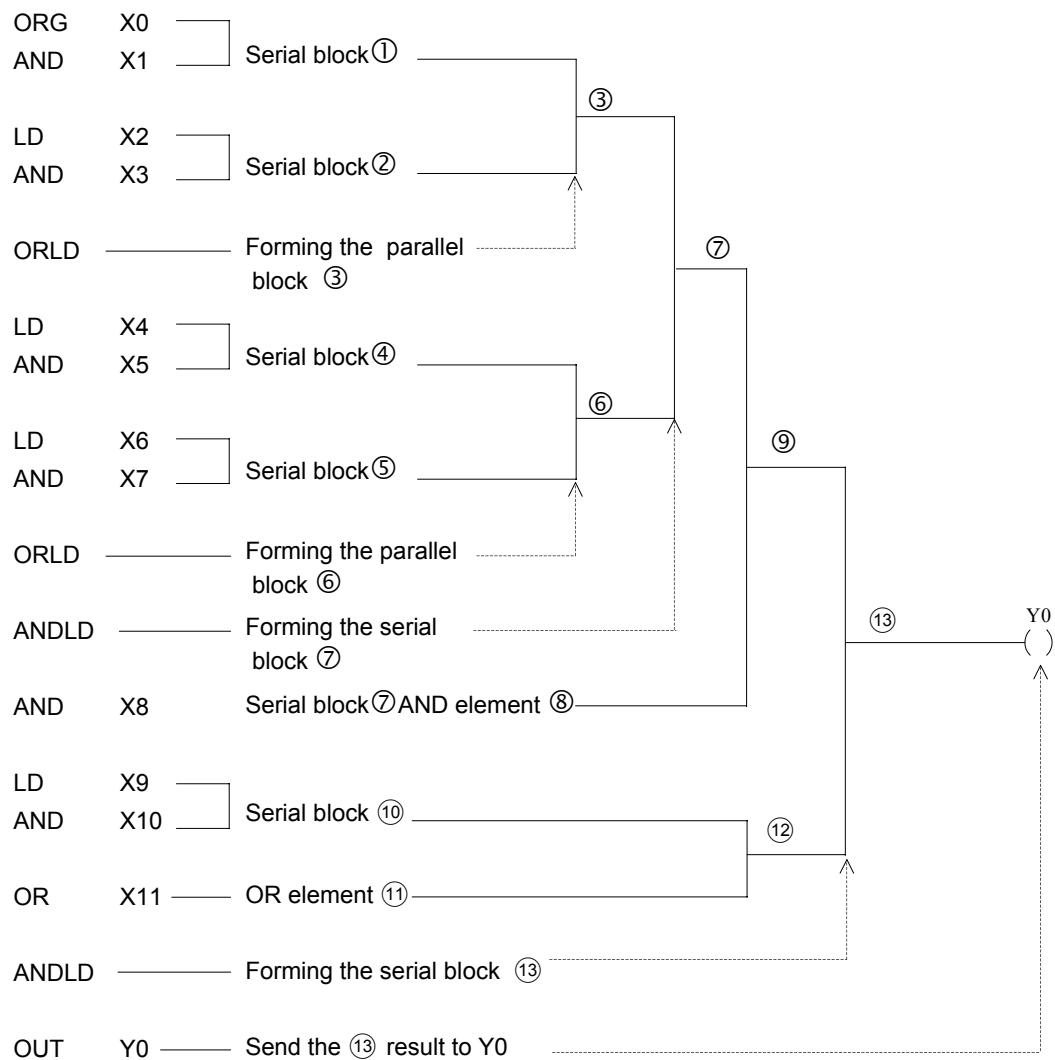
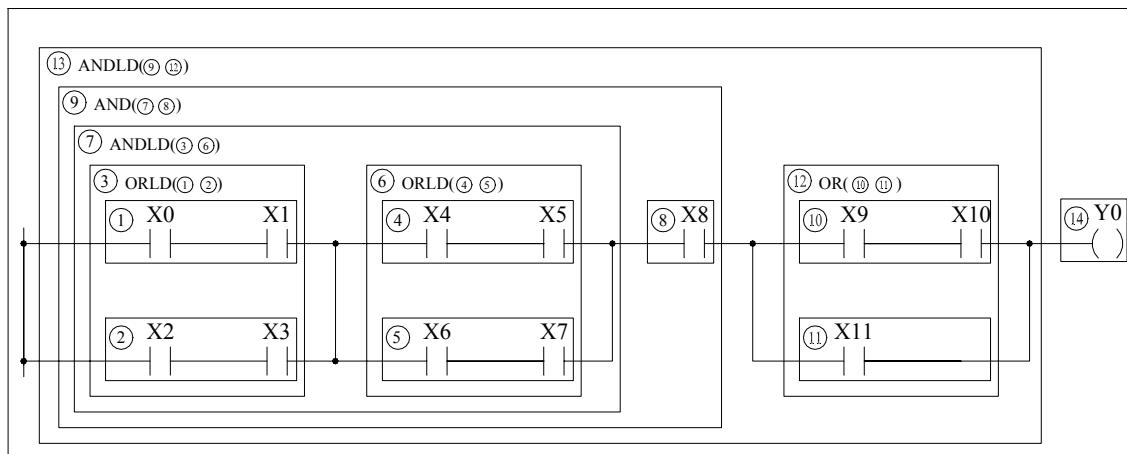
ORG SHORT  
OUT

Y 0

## 1.5 The De-Composition of a Network (Users of WinProladder can skip this section)

The key process of de-composition of a network is to separate the circuits that appear between two vertical lines into independent elements and serial blocks, then coding those elements and serial blocks according to the mnemonic coding rules and then connect them (with ANDLD or ORLD instruction) from left to right and top to bottom to form a parallel or a serial-parallel blocks, and finally to form a complete network.

Sample diagram:



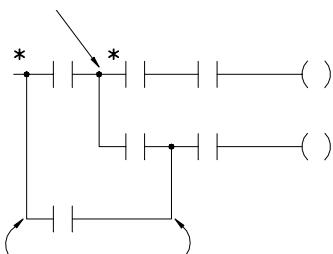
## 1.6 Using Temporary Relays (Users of WinProladder can skip this section)

The network de-composition method for mnemonic coding demonstrated in section 1.5 does not apply to the branched circuit or branched block. In order to input the program using the method shown in section 1.5, It must first to store the statuses of branched nodes in temporary relays. The program design should avoid having branched circuit or branched block as much as possible. Please refer the next section “Program Simplification Techniques”. Two situations that must use the TR are described at below.

- Branched circuit: Merge line does not exist at the right side of the branch line or there is a merge line at the right side of the branch line but they are not in the same row.

Example : \* indicates setting of TR relay

Without merge line

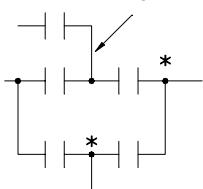


Although this branch has merge lines  
but they are not in the same row, so this  
is also a branched circuit

- Branched block : The horizontal parallel blocks with a branch in one of the blocks.

Example :

Merge line



Branch line

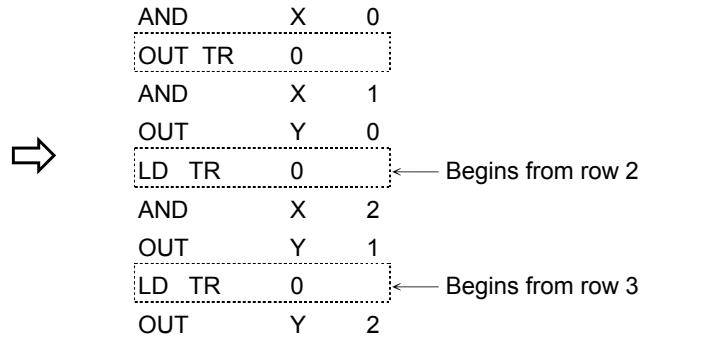
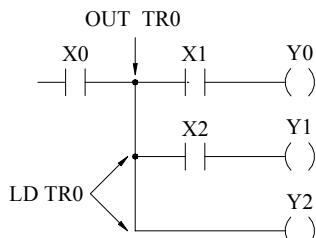
Remark 1: The OUT TR instruction must be programmed at the top of the branched point. LD TRn instruction is used at the starting point of the circuits after second rows of the branch line for regaining the branch line status before you can connect any element to the circuits. AND instruction must be used to connect the first element after OUT TRn or LD TRn instruction. LD instruction is not allowed in this case.

Remark 2: A network can have up to 40 TR points and the TR number can not be used repeatedly in the same network. It is recommended to use the numbers 1,2,3... with sequence. The TR number must be the same in the same branch line. For example, if a branch line uses OUT TR0, then starting from row 2, LD TR0 must be used for connection.

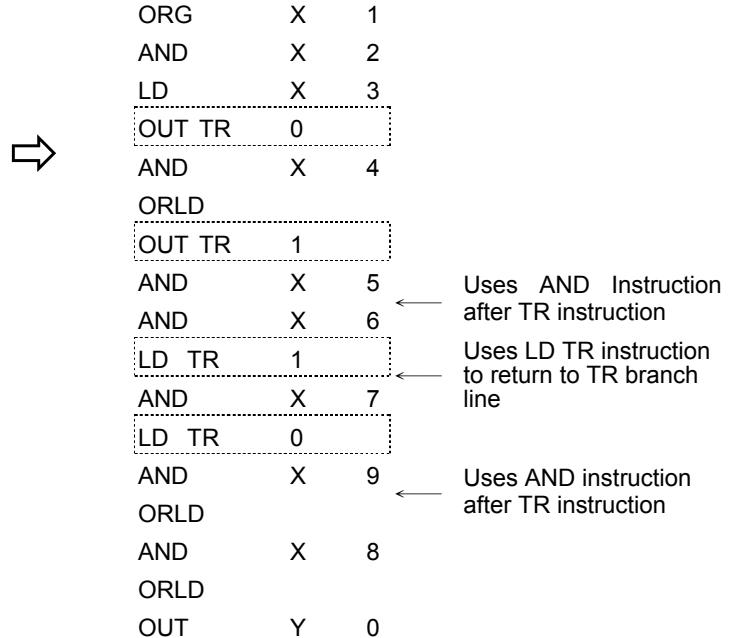
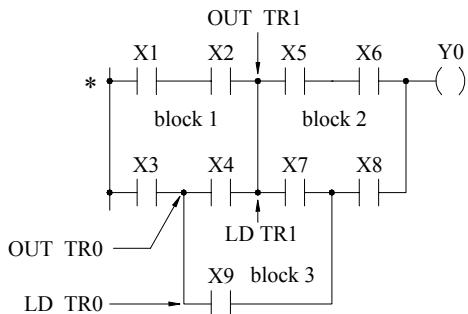
Remark 3: If the branch line of a branched circuit or a branched block is the origin line, then ORG or LD instructions can be used directly and TR contact is not necessary.

Remark 4: If any one of the branched circuit rows is not connected to the output coil (there are serially connected elements in between), and other circuits also exist after the second row, a TR instruction must be used at the branch points.

Example:



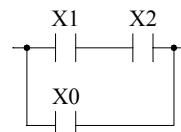
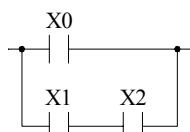
Example:



- The above sample diagram shows a typical example of connecting two parallel blocks in series. Block 3 is formed when the element X9 is introduced into the network and the two parallel blocks become the branched blocks.
- TR instruction is not necessary because the (\*) point is the origin line.
- If have already used TR relay to connect two blocks serially, then ANDLD instruction is not necessary.

## 1.7 Program Simplification Techniques

- If a single element is connected in parallel to a serial block, The ORLD instruction can be omitted if the serial block is connected on top of this single element.



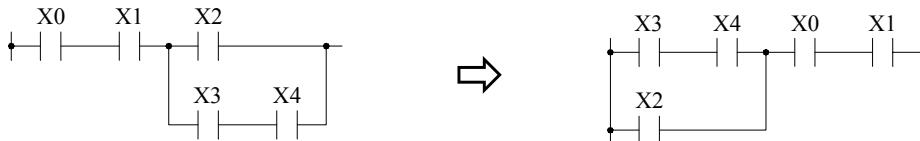
```

LD      X    0
LD      X    1
AND     X    2
ORLD
  
```

```

LD      X    1
AND     X    2
OR     X    0
  
```

- When a single element or a serial block is connected in parallel with a parallel block, ANDLD instruction can be omitted if put the parallel block in front.



```

ORG      X   0
AND      X   1
LD       X   2
LD       X   3
AND      X   4
ORLD
ANDLD
    
```

```

ORG      X   3
AND      X   4
OR       X   2
AND      X   0
AND      X   1
    
```

- If the branch node of a branch circuit is directly connected to the output coil, this coil could be located on top of the branch line (first row) to reduce the code.



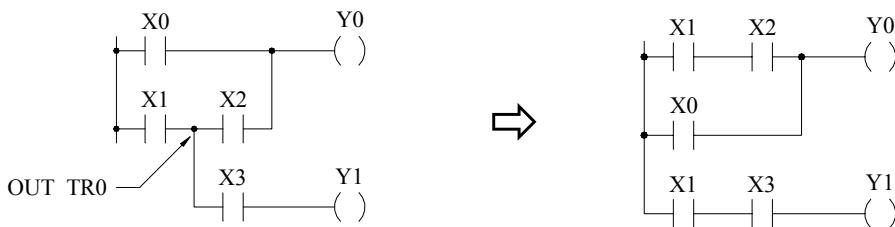
```

OUT TR  0
AND     X  0
OUT     Y  0
LD TR  0
OUT     Y  1
    
```

```

OUT     Y  1
AND     X  0
OUT     Y  0
    
```

- The diagram shown below indicates the TR relay and the ORLD instruction can be omitted.



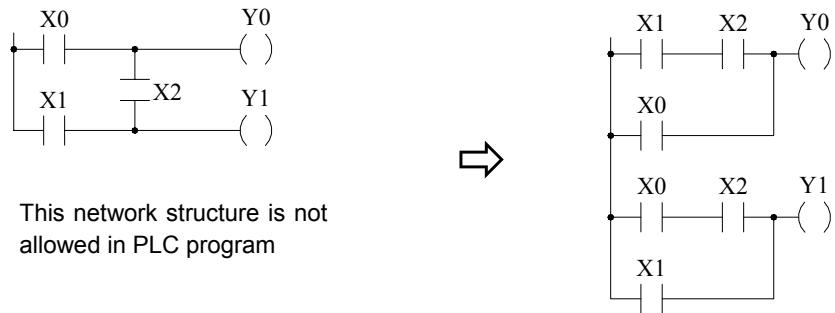
```

ORG      X   0
LD       X   1
OUT TR  0
AND     X  2
ORLD
OUT     Y  0
LD TR  0
AND     X  3
OUT     Y  1
    
```

```

ORG      X   1
AND     X  2
OR     X  0
OUT     Y  0
ORG     X  1
AND     X  3
OUT     Y  1
    
```

- Conversion of the bridge circuit



```

ORG      X      1
AND      X      2
OR       X      0
OUT      Y      0
ORG      X      0
AND      X      2
OR       X      1
OUT      Y      1

```

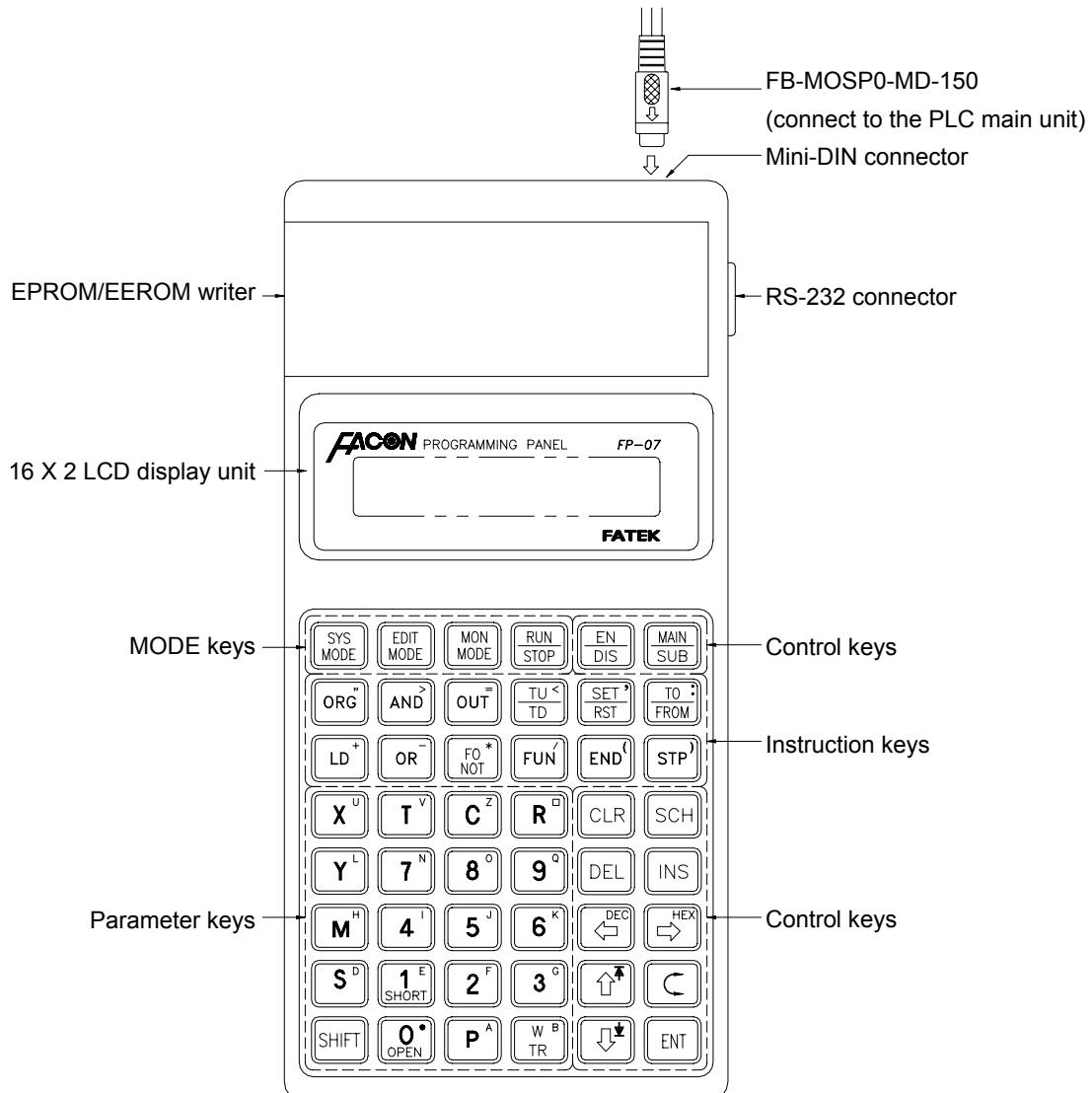
## Chapter 2 FP-07 Programming Panel

FP-07, a handheld programming device, can be used as “Programming Panel” (PP) and “Timer Counter Access Panel” (TCAP) for FB-PLC. It’s especially suited for timer, counter and register setting purposes. The differences in models and functions are listed below:

Model \ Function	Main Unit and Editing Capacity	Write/Copy Function	RS-232
FP-07A	FBE/FBN(13KW) FB (8KW)	N/A	No
FP-07B	FBE/FBN (13KW) FB (8KW)	27C256~27C010,28C256,28C010 29C010,W29EE011,W29C011A AT29C010, AT49F010	Yes

### 2.1 Introduction to FP-07

#### 2.1.1 Appearance



## 2.1.2 Keypads Arrangement

The keypads of FP-07 console is functionally divided into four groups:

- Mode keys: Four mode keys, , are used for selecting operation mode of FP-07.
- Control keys: Control keys are used for mode operations (all the blue keys on FP-07 except mode keys).
- Instruction keys: Instruction keys are used for entering FB-PLC instructions with parameters or data. All the black keys in the top two rows and the two keys, at the fourth row of FP-07 (refer to the description of special keys below) are the instruction keys.
- Parameter keys: Parameter keys are used for entering the operand's numbers or contents. All black keys, except instruction keys, are parameter keys.

In order to obtain optimum convenience and maximum input capability under a limited number of available keys, four groups of keys are designed as multi-purpose as described below:

- a. Alternation keys: Alternation keys are those with a horizontal line marked in the middle to separate two distinct functions (a total of six keys, ). By pressing the key ( as an example) for the first time, the function above the horizontal line (RUN) will be displayed on the LCD display unit. By pressing the key again, the function below the horizontal line (STOP) will be displayed. If the key is pressed for the third time, the LCD screen will display the function above the horizontal line (RUN) again. The process will repeat if the key is pressed repeatedly. By pressing (at the lower rightmost corner of the keypads), the function last shown on the LCD screen will take effect.
- b. Shift Key: After pressing this orange key (at the lower leftmost corner of the keypads), an S letter will first appear on the LCD display unit. If now any key at the upper rightmost corner with a small orange letter printed is pressed, the small orange letter (the "shift key letter") will be entered or the function described by the orange letter (such as or ) will be executed and the letter S on the LCD will disappear.
- c. Compound keys: There are two rows of white letters on each of the four keys, , those are neither shift keys nor alternation keys but keys that can perform the two functions represented by the two rows of white letters. Under special arrangement when one of these keys is pressed, FP-07 will carry out the function described by either the upper or the lower row in accordance with the current operation mode automatically without any further instruction given by the user.
- d. Double-definition keys: Two keys, , represent Timer and Counter, respectively and also the letters T and C. Similar to the compound keys, the two functions cannot be operated simultaneously. FP-07 will make necessary judgment itself automatically.

Remark 1: Pressing two or more keys at the same time is prohibited while operating the FP-07 programming panel. For example, after the key being pressed, it must wait until it is released before the next key can be pressed.

Remark 2 : keys are used for moving the cursor by one position to the direction of the arrow whenever one of these four keys is pressed. The cursor will move rapidly if one of these keys is pressed for more than 0.7 second without being released.

## 2.1.3 EPROM/EEPROM Writer (FP-07B Only)

EPROM (27C256, 27C512, 27C010, 27C1001) and EEPROM (28C256, 28C010, 29C010, W29EE011, W29C011A, AT29C010, AT49F010) are the two types of ROM units that can be used to store programs. In most cases, FP-07B can recognize the ROM type and carry out IC copying or reading automatically. For those few brands of IC that FP-07 cannot recognize, IC copying and reading may still work if the IC type is selected correctly.

Remark 1: EPROM 28C010, 29C010, AT29C010, AT49F010 and W29EE011 can be used in PLC by plugging them into the PLC ROM socket after having programs written/copied onto them. 28C256 can be used for program and/or data storage on FP-07B only.

Remark 2: 27C256 and 28C256 can only store programs and/or data up to 8KW (R0~R3839) while the other ICs can store programs and/or data up to 13KW (R0~R3839, D0~3071 and R5000~R8071).

## 2.1.4 RS-232 Communication Port

At present, this communication port can only be used for connecting the printer. Please refer to section 2.4.4 for the printing function.

## 2.1.5 Interface Connection between FP-07 and PLC

Since FP-07 does not have its own power supply, therefore all of its operations can only be carried out after the connection between FP-07 and the PLC main unit is completed by using an unique FP-07 communication cable (CPFB-150). If all operations are functioning normally and the PLC main unit selects FP-07 to be used as a programming panel (for FP-07 being used as a TCAP, please refer to page 2-40), FP-07 will display the PP initiating screen as shown below. It indicates that the connection has been established and is ready for operation.

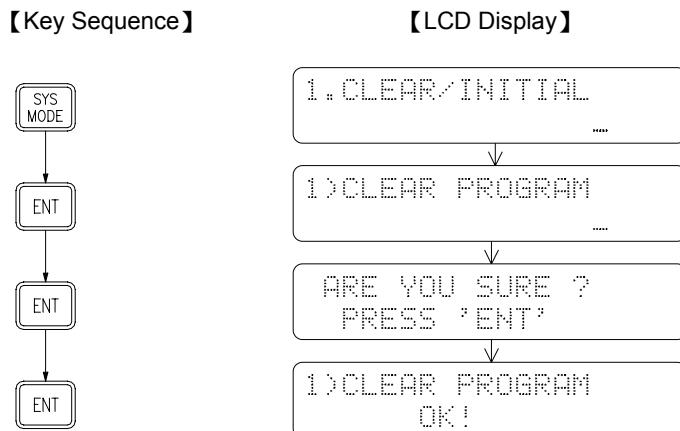
PP READY!

## 2.2 Program Edit, Run, Monitor, Forced Set/Reset and Enable/Disable

A simple example program is illustrated in this section to show how to edit (input) the control programs, to run or stop the PLC, to use the monitor mode to examine the program execution results while the PLC is running, to forced set/reset the status of digital point or set the value of register, to enable/disable the digital points by using the FP-07 with a fast and efficient way.

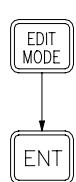
### 2.2.1 Program Edit

Please ensure that the program area in the PLC is empty (i.e. no program remained) before program editing for this example. The following keys can be used to clear the program area (This step can be omitted for a newly purchased PLC since the "CLEAR" operation has been performed before the shipment from the factory)

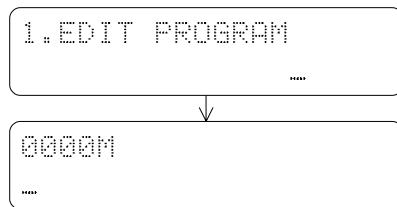


After verifying that the program area is empty, press the **EDIT MODE** key to enter the edit mode.

**【Key Sequence】**



**【LCD display】**



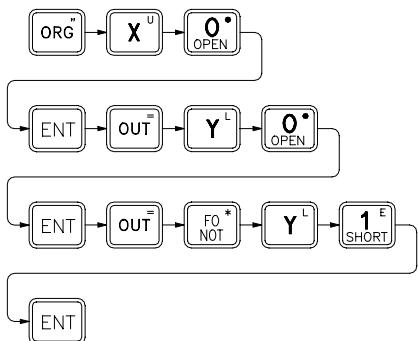
The above LCD display, "0000" indicates the current address in a program, and "M" indicates the main program area. 0000M means now we are at the beginning of the main program area. Following input instructions will occupy the areas 0001M, 0002M, 0003M and so on. When you enter the edit mode for the first time, FP-07 will enter the main program area automatically.

	<pre> ORG      X  0 OUT      Y  0 OUT NOT Y  1 ORG      M1922 OUT      Y  2 ORG      X  1 OR       Y  3 AND NOT X  2 OUT      Y  3 ORG      X  3 T200 [PV:] 10 ORG      T  200 OUT      Y  4 ORG      X  4 LD       X  5 C0 [PV:] 20 ORG      C  0 OUT      Y  5   </pre>
<ul style="list-style-type: none"> <li>• (1)~(7) indicate the starting points of the network.</li> <li>• X0~X5, Y0~Y5, M1922 etc. Please refer to Chapter 3.</li> <li>• Please refer to Chapter 5 through Chapter 8 for detail description of functionality of instruction used above.</li> </ul>	
<small>*the characters in <b>[ ]</b> are the directive string shown by FP-07 which are not entered by the user.</small>	

The following demonstration illustrates the programming procedures of the sample program shown above. The instructions are displayed on the LCD screen. If typing error occurs during the programming process before pressing the **ENT** key, simply press the **CLR** key to clear the incorrect instruction. If a typing error is detected after pressing the **ENT** key, you must find the incorrect instruction first then press the **DEL** key to delete the incorrect instruction or key in the correct instruction directly and press the **ENT** key to replace the incorrect one.

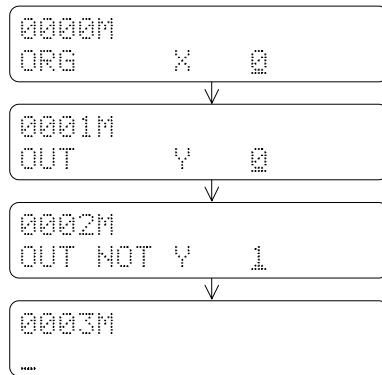
## 【Key Sequence】

(1)

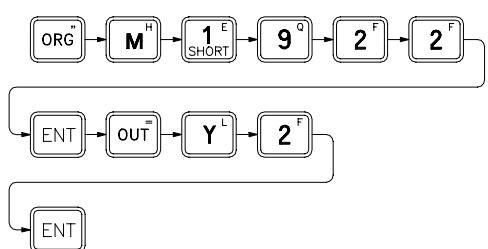


## 【LCD Display】

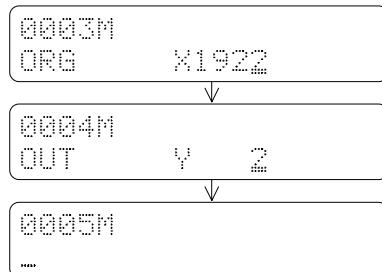
(1)



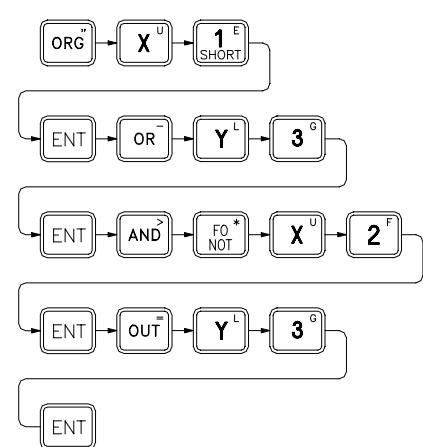
(2)



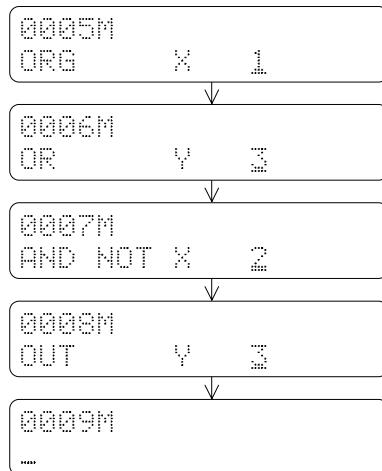
(2)



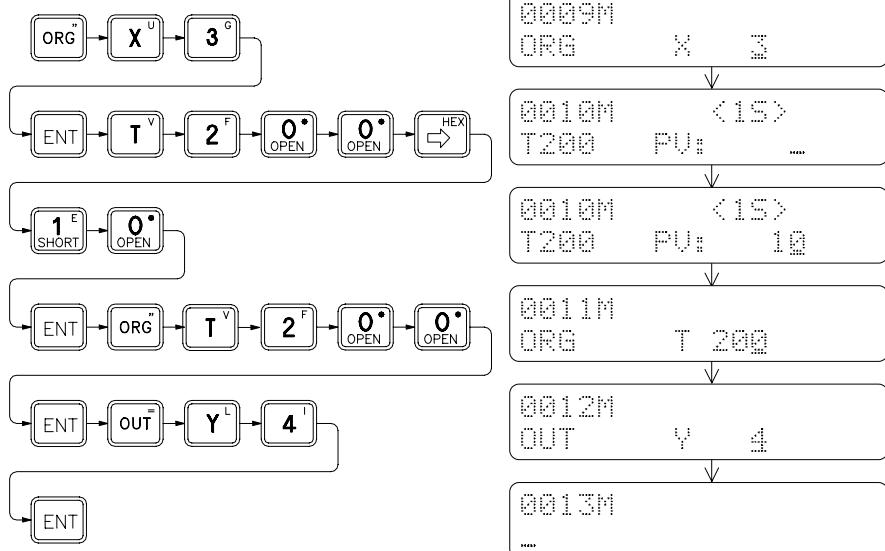
(3)



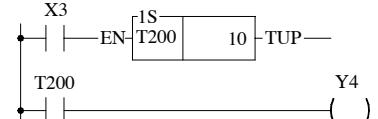
(3)



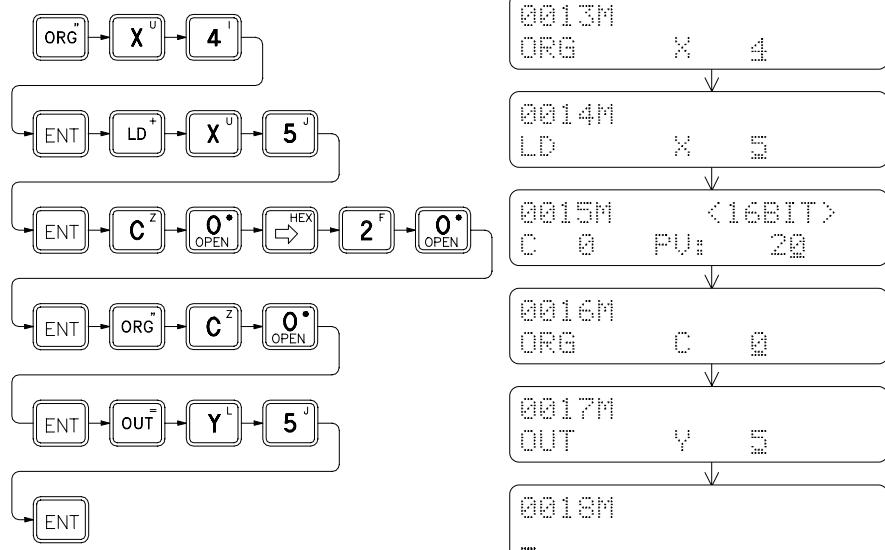
(4)(5)



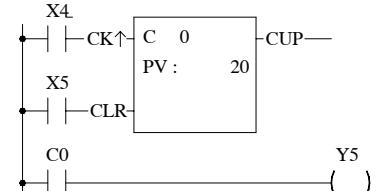
(4)(5)



(6)(7)



(6)(7)

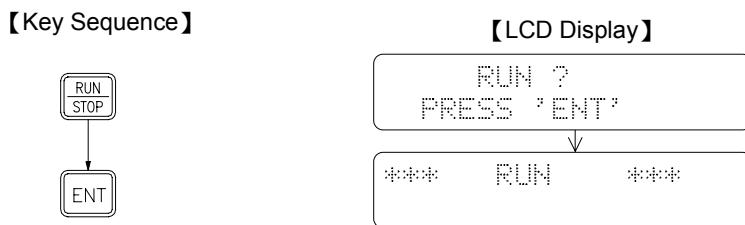


After entering all the instructions of a program, can continuously depress or press to let the edit point back to the start of ladder program (Similarly can either continuously press or press keys to get to the end point of the ladder program) then depress successively to check if the mnemonic codes are correct or not. If everything is correct, then it is the time to run the program.

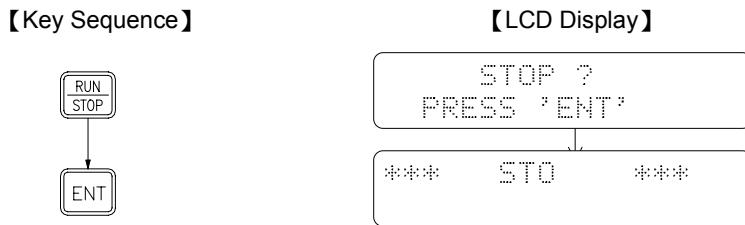
## 2.2.2 Program RUN

After pressing , there will be a message displayed on LCD asking you whether you want to change to RUN (if the PLC is currently at STOP state) or to STOP (if the PLC is at RUN state). Press to execute your choice as shown below:

( 1 ) Changing PLC from STOP to RUN:



( 2 ) Changing PLC from RUN to STOP:

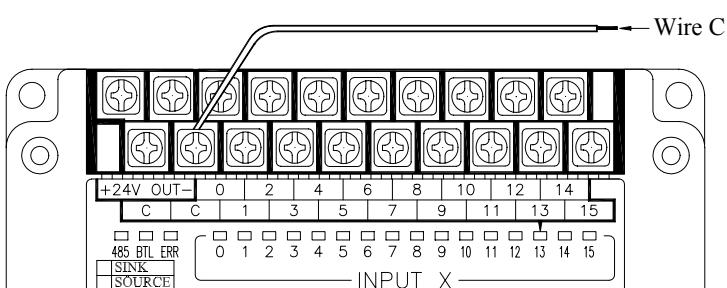


If you want to monitor the program execution of the example program, you must let the PLC in RUN state. So please first repeat the step(1) as shown above. After the PLC turns to RUN state, you can examine the program execution results by entering the monitor mode which will be described in the next section.

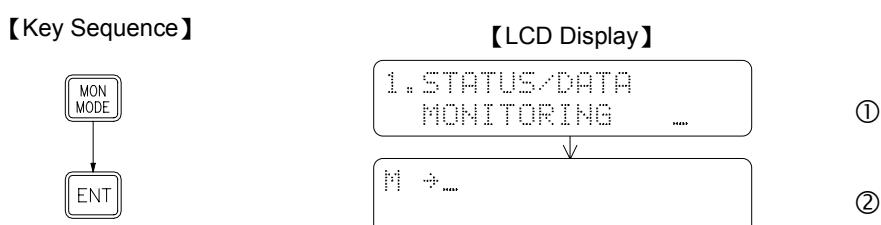
### 2.2.3 Program Execution Monitoring

In order to monitor the program execution results of sample example, first you have to connect a conductive wire (hereafter called wire C) to the input terminal C in order to activate X0~X5 input points as shown in Figure <1>. The purpose of doing so is to emulate the switch by using the wire to touch input. The other way to do this is to disable the X0~X5 input points first, then use  to set the X0~X5 status with a Forced SET/RST operation (for details please refer to 2.2.4).

Figure <1>



If you want to monitor the program execution results (the digital status or the data registers), you need to use function item 1 of the monitor mode which is called “STATUS/DATA MONITORING”. Following key operations shows a way to enter the “STATUS/DATA MONITORING” of the monitor mode.



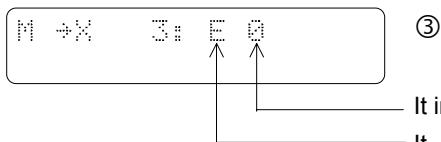
M indicates the Monitor mode status

Under the Status/Data monitoring mode, LCD screen can monitor two rows of data at the same time. But only one row of data that pointed by cursor can be entered at a time. Using the Row Change  can move the cursor between these two rows. In the following key operations, the first row shown in the LCD screen is for monitoring the digital status and the second row shown in the LCD screen is for monitoring the register data.

#### 【Key Sequence】



#### 【LCD Display】

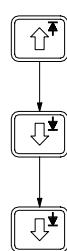


It indicates the X3 status is at "0"

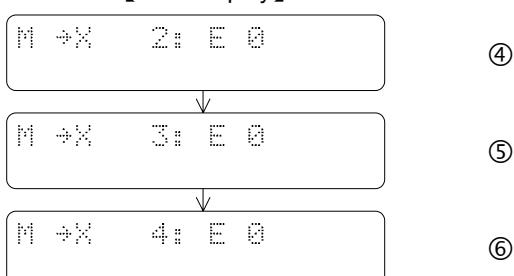
It indicates X3 is Enabled (refer to 2.2.4 for more details)

After X3 status appears on the LCD screen, can use   to monitor the preceding or the succeeding contact points. Starting with the message shown in LCD display ③, if you press  once, X2 status appears on the LCD screen. X1 status will display on the LCD screen if you press  one more time. If you want to monitor the status of X4, X5 and so on, press  to get this.

#### 【Key Sequence】

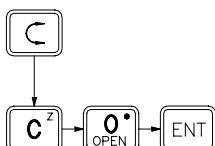


#### 【LCD Display】

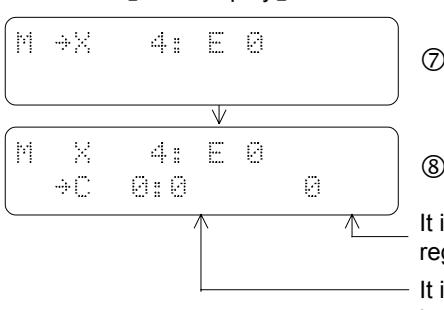


After pressing , the cursor now will move to the second row (the LCD display of first row remains unchanged). The rest of the input and operations will all be taken place at the second row.

#### 【Key Sequence】



#### 【LCD Display】



It indicates the current value of C0 register is at 0

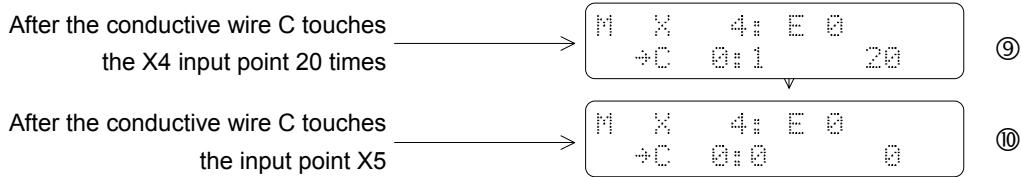
It indicates the contact point C0 status is at 0

In the above LCD display, the second row shows an example of monitoring register data. The message shown in this row indicates the C0 contact point status (the status is at "1" if the counter value is equal to the preset value), and the current value (counter value) of the C0 register.

When the LCD display shown above appears, can use the conducting wire C to touch the external input point X0~X5 to test this program. The operating results can be seen from PLC's output points (Y0~Y5). Furthermore, can use this monitoring display to examine the data that the output points (Led indicators) unable to show, such as the status of internal contact points, the current value and contents of T and C registers. The description of the functionality of the example program and the relationship of corresponding I/O points are listed in the table below. You can conduct your own experiments according to this table and observe the operating results.

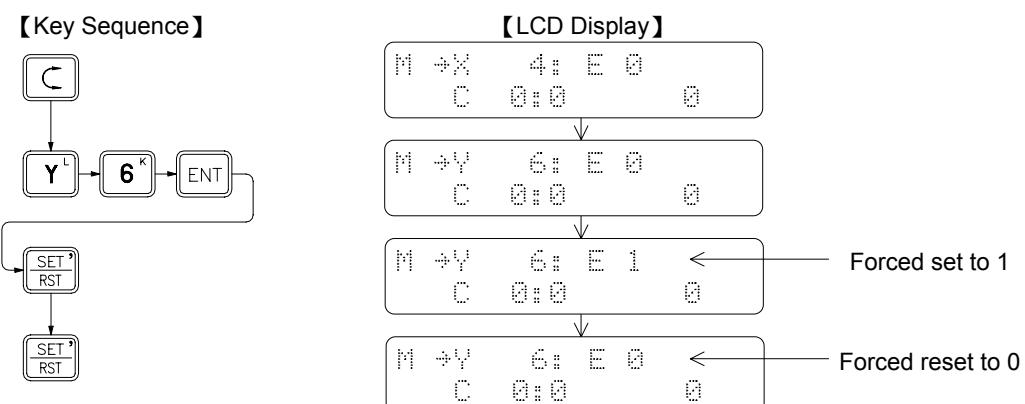
Network Number	Description of the circuit functions	Relationships between input (Xn) and Output (Yn)
(1)	This network sends the X0 status to Y0 and then inverts the X0 status and sends the results to Y1.	X0=1 then Y0=1, Y1=0 X0=0 then Y0=0, Y1=1
(2)	Sends the internal contact M1922 status (1 second) to Y2.	Y2 switches ON/OFF once every second (not input related)
(3)	Latch Circuit : X1 is the starting contact, as soon as X1 turns "ON", Y3 turns "ON" and retains the statuses. X2 is a reset contact. As soon as X2 turns "ON", Y3 turns "OFF" and retains the status.	If X1"ON" then Y3"ON" If X2"ON" then Y3"OFF"
(4) (5)	10 seconds Timer	If X3"ON", after 10 seconds, Y4"ON" If X3"OFF", Y4"OFF" immediately
(6) (7)	20 times Counter (counts 20 times)	Register C0 increments 1 for every X4 switching from OFF to ON until C0=20, Y5=1 If X5"ON", then register C0 clears to 0 and contact C0 is also at 0, therefore Y5=0

As shown in Display ⑧, every time the conducting wire C as shown in Figure <1> touches the input point X4, the current value of the register C0 will automatically increment by 1.(Remark: the current value of the register C0 may increment by more than one because several pulses may have been generated for each touch due to bouncing) The status of contact C0 switches to 1 when register C0 value reaches 20 as shown in Display ⑨. If the conducting wire C touches input point X5, then the status of contact returns C0 to 0 as shown in Display ⑩. Every time turn-on the PLC, the display format of current value of register C0 is in decimal number. If you want to display the value in hexadecimal number, press To return to the decimal number, press .



#### 2.2.4 Forced Set/Reset and Enable/Disable of Digital Status

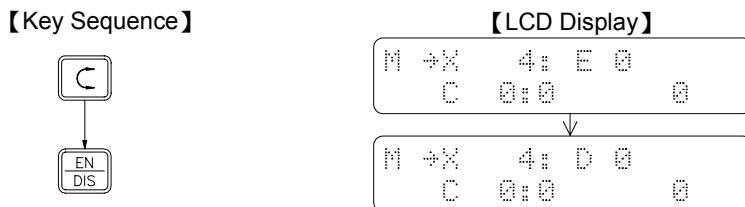
While in the monitor mode, for digital points, not only can monitoring its status, but also can force its status by using the keypad of FP-07. In general, forced set/reset is often used for the diagnosis and program testing purpose. The following key sequence continues the operation shown in Display ⑩. It demonstrates a key operation procedure of forced set/reset while monitoring the Y6 status. Y6 status is forced set to 1 first, and then is forced reset to 0.



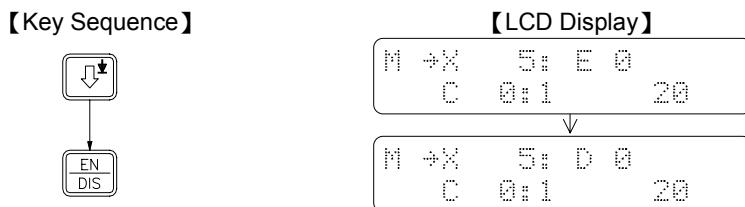
For input contacts with coils driven by an OUT instruction, the forced set/reset status can only be retained for a very short period of time (less than one scan time). Very soon the forced  status will be replaced by the new status of input or program output following an OUT instruction. PLC I/O status and OUT instructions are refreshed after each scan therefore the forced set/reset status can only be retained for a very short period which is the time between status forced and new replaced status taking place. The reason of the forced set status can retain in previous example is because Y6 is not driven by any ladder code in example program, that is, after the status is being written-in, there is no programmed operation to change the Y6 status again. But Y0~Y5 in the example program are controlled by the PLC program meaning any forced set/reset status will be overwritten by the new data generated from further program executions.

In order to forced set the statuses of input contacts (X0~X255) and coils of programs which are driven by OUT instructions, you must perform the “Disable” function first to temporarily allow the data out of the control of ladder diagram program and I/O refresh process. In this way, you will be able to retain the data while performing the data change. To return to the normal operation condition and put the data again under the control of the program, you must use the “Enable” function.

Using network (6) as an example, continuous from the display ⑩, first disable the X4 by using the “Disable” function and then using “Forced” function to control the ON/OFF state of X4 input contact instead of using the C wire. The key in sequence is shown below.



Following the above key sequence, as soon as press , C0 value changes to 1 instantly. C0 value will increment by 1 every time press  twice until this value reaches 20. When C0 reaches 20, “Count Up” is done and the contact status changes to 1 (same as the status shown in Display ⑨). Please follow the key sequence shown below to perform the “Clear” operation using the X5 input point.

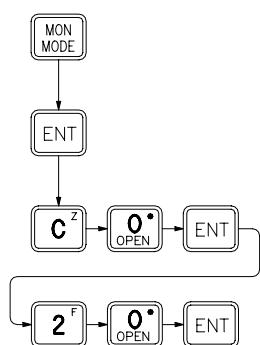


Following the above key sequence, X5 will switch to 1. The value and status of C0 will all clear to 0 if you press .

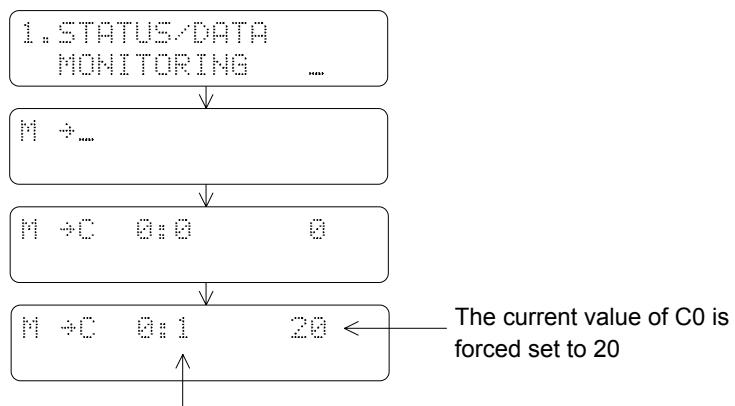
## 2.2.5 Set the Register Data

Similar to the digital status, the register data can also be monitored and changed (forced set) in the Monitor Mode. But the enable and disable operations cannot perform to register. The input registers (R3840~3903) or the registers which are written by the application instructions, can be set to certain value, but in a very short period of time those registers will be replaced by the new input register data or data generated from the operations of function instructions. The input registers data are refreshed each scan, while the data generated from the operations of function instructions changes only when function instruction is executed. The following key operations uses Network (6) as an example. First touch the input point X5 once with conductive wire C to clear register, then enter the monitor mode and set C0 value to 20 which cause C0 to “Count Up” and consequently change the C0 contact status to 1.

**【Key Sequence】**



**【LCD Display】**



The current value of C0 is forced set to 20. The contact status of contact C0 changes to 1 because the current value of C0 now is equal to the preset value (i.e. count-up) is done.

## 2.3 The Functions of FP-07

### 2.3.1 Function List

There are four operating modes for FP-07, which are System Mode, Edit Mode, Monitor Mode and RUN/STOP Mode. The function descriptions for each mode are listed at below.

#### ● System Mode

◎indicates the operable items when the password has not been closed.

Function	Descriptions
1. CLEAR/INITIAL	
1) CLEAR PROGRAM 2) CLEAR REGISTER 3) CLEAR COIL STATUS 4) ENABLE ALL DIGITAL (contact and coil) 5) SYSTEM INITIAL 6) DISABLE ALL DIGITAL (contact and coil)	<ul style="list-style-type: none"> <li>• Including Documents, Password, Program ID, Configuration, ROR (Read Only Register) data</li> <li>• Enables all contacts</li> <li>• Clears all data, returns PLC to its initial factory settings</li> </ul>
2. COPY/COMPARE (only for FP-07B)	
◎ 1) SAVE PROGRAM (PLC→PACK) 2) SAVE REGISTER (PLC→PACK) 3) LOAD PROGRAM (PACK→PLC) 4) LOAD REGISTER (PACK→PLC) 5) COPY (PACK→PACK) 6) COMPARE PROGRAM (PLC↔PACK) 7) COMPARE PROGRAM (PACK↔PACK) ◎ 8) WATCH PACK PROGRAM 9) ROM PACK BLANK CHECK A)SAVE PROGRAM (PLC→BUFFER)	<ul style="list-style-type: none"> <li>• Including Documents, Password, Program ID, Configuration, ROR data</li> <li>• HR、DR、ROR and SR data</li> <li>• Including Documents, Password, Program ID, Configuration, ROR data</li> <li>• HR、DR、ROR and SR data</li> <li>• Including Documents, Password, Program ID, Configuration, the data of HR, DR, ROR and SR</li> <li>• Only check the program part</li> <li>• Save The PLC program to retentive BUFFER located in FB-07</li> </ul>
3. PASSWORD/ID	
◎ 1) PASSWORD OPEN 2) PASSWORD CLOSE ◎ 3) PASSWORD SETTING 4) PROGRAM ID SETTING 5) PLC ID SETTING	<ul style="list-style-type: none"> <li>• These two functions are only applicable after the password has been set</li> <li>• Use the Program ID and PLC ID to prevent the Hard Copy of the program in ROM PACK</li> </ul>
4. PRINT OUT (only for FP-07B)	
◎ 1) PRINT MNEMONIC ◎ 2) PRINT MNEMONIC WITH DOCUMENT ◎ 3) PRINT PROGRAM ◎ 4) PRINT PROGRAM WITH DOCUMENT ◎ 5) PRINT CROSS-REFERENCE 6) PRINT REGISTER DATA	

Functions	Descriptions
5. CONFIGURATION  1) INTERNAL COIL PARTITION 2) STEP COIL PARTITION 3) 0.01S~1S TIMER PARTITION 4) 16-BIT COUNTER PARTITION 5) 32-BIT COUNTER PARTITION 6) DATA REGISTER PARTITION 7) READ-ONLY REGISTER PARTITION 8) HSC/HST/INT ASSIGNMENT 9) STATION NUMBER ASSIGNMENT A) MAX FREQ. OF X0~X15 B) DEFINE MD OF HPSO C) DEFINE NORMAL POLAR. OF HPSO	Please refer to "Default Configuration"  Only for FB <sub>E</sub> /FB <sub>N</sub>
6. SYSTEM MESSAGE	Once in this mode, will be able to observe PLC and PP versions, memory usage, password setting and system configurations and more by pressing  or 
7. AS A TC ACCESS PANEL	This function set the PP/TCAP flag of PLC to TCAP mode. Any FP-07 console connected to this PLC will be used as a TCAP (Timer Counter Access Panel)

● Edit Mode

Functions	Descriptions
◎ 1 EDIT PROGRAM 2 EDIT REGISTER DATA 3 SYNTAX CHECK 4 MOVE HR→ROR 5 CHECK DOUBLE COIL/T/C 6 EDIT HPSO INSTRUCTION 7 EDIT LINK INSTRUCTION 8 EDIT DOCUMENT	

● Monitor Mode

Functions	Descriptions
1 STATUS/DATA MONITORING	
◎ 2 PROGRAM MONITORING	Can monitor the program with the contact status display while PLC is in RUN state

● RUN/STOP Mode

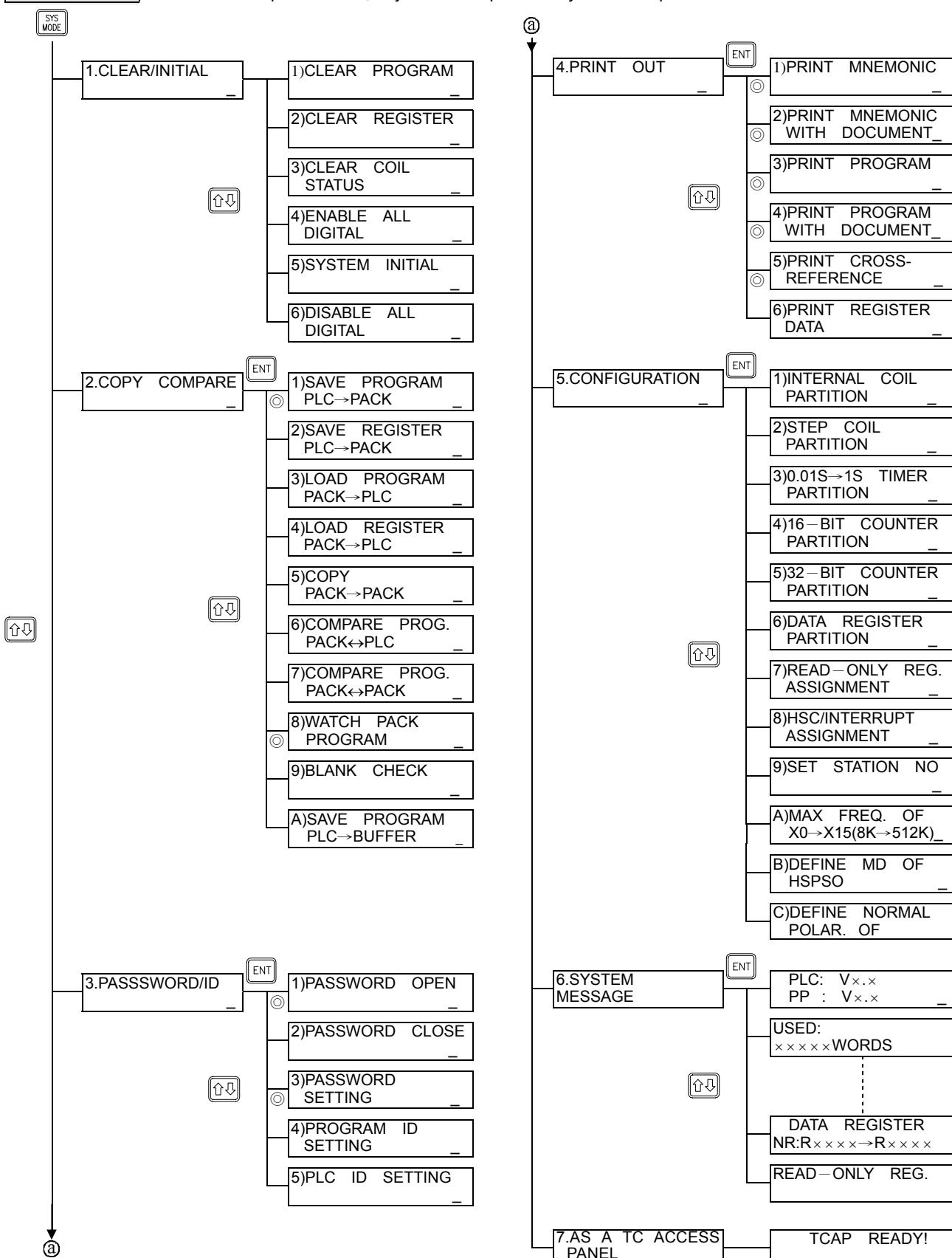
Functions	Descriptions
PLC RUN/STOP Control	

### 2.3.2 Operation Flowchart

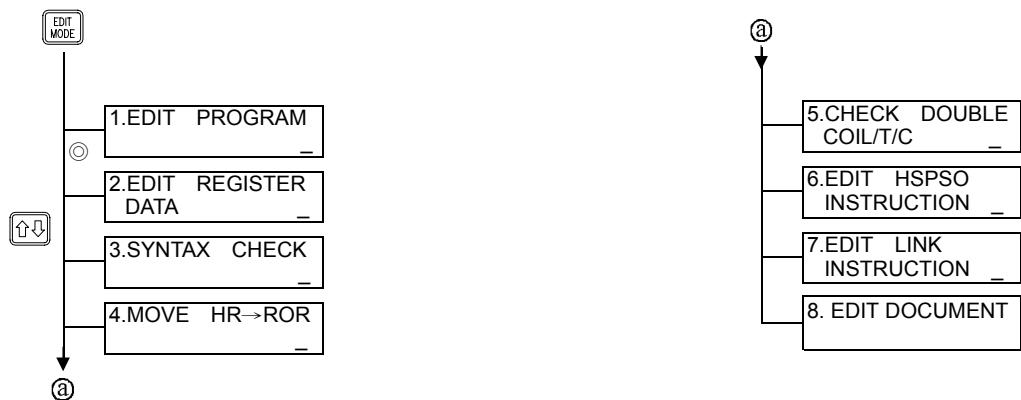
The diagram shown at below is the operation flowchart for System Mode, Edit Mode, Monitor Mode and RUN/STOP Mode.

#### A. SYS MODE

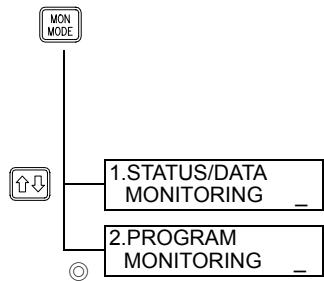
Note: Items prefix with symbol can operate only when the password has not been closed.



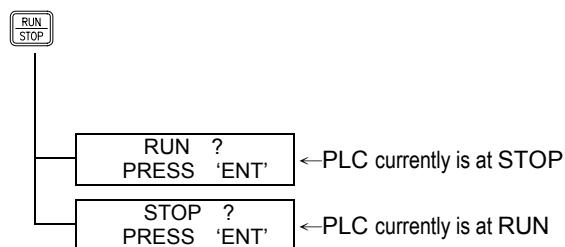
#### B. EDIT MODE



#### C. MONITOR MODE

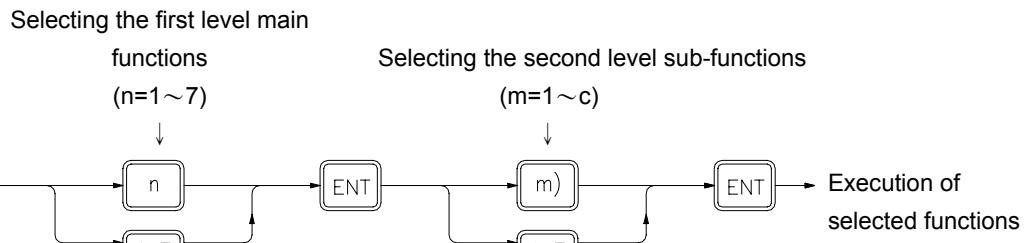


#### D. RUN/STOP MODE



## 2.4 Introduction to SYSTEM MODE Operation

Fundamental key operations of System Mode:

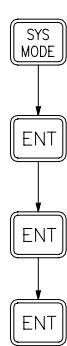


- System Mode includes eight main operation functions. N indicates the nth main function. Every main function also has n numbers of sub-functions. m indicates the mth sub-function within the nth main function.
- When you enter either the first or second level of function for the first time, you will be automatically prompted into main function 1 or sub-function 1 [n=1 or m=1]. If this is not the function you needed, you can either directly input the function number (n) or using **UP** or **DOWN** to search for the specific function you are looking for and then press **ENT** to execute the function which you have just selected.

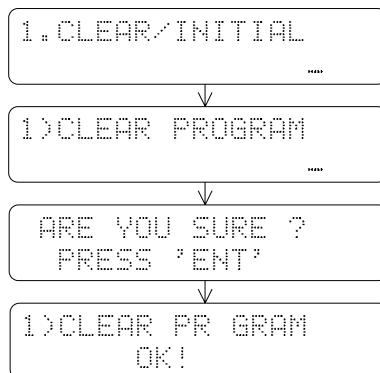
### 2.4.1 CLEAR/INITIAL

#### 2.4.1.1 Clear Program

【Key Sequence】

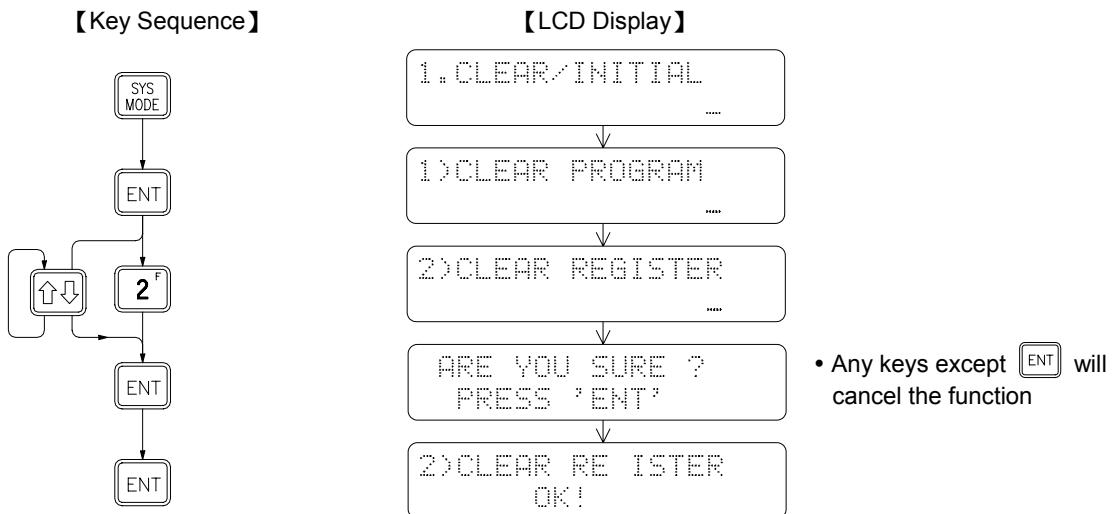


【LCD Display】

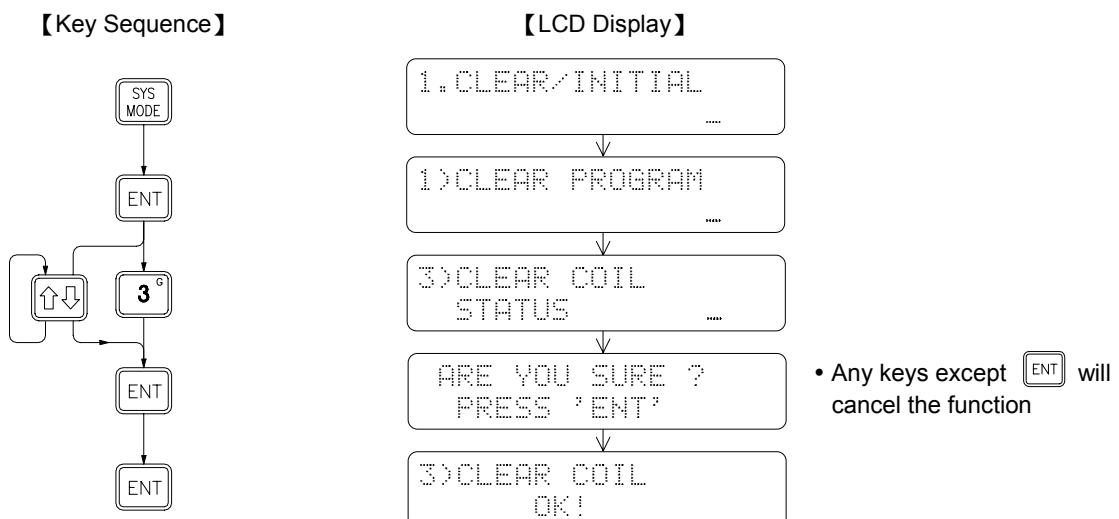


• Any keys except **ENT** will cancel the function

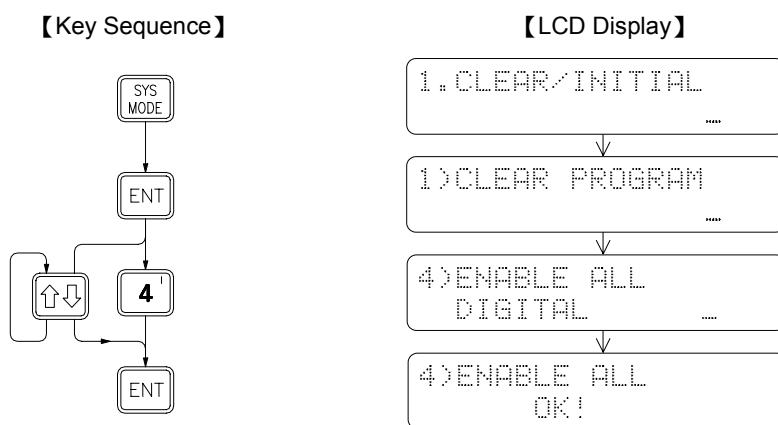
#### 2.4.1.2 Clear Register



#### 2.4.1.3 Clear Coil Status

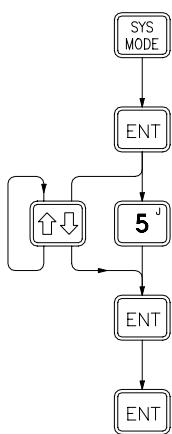


#### 2.4.1.4 Enable All Digital

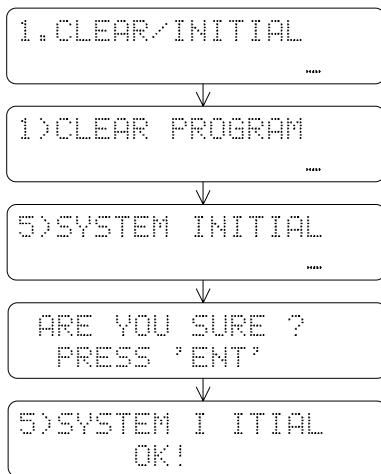


#### 2.4.1.5 System Initial

## 【Key Sequence】



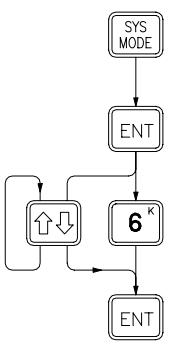
## 【LCD Display】



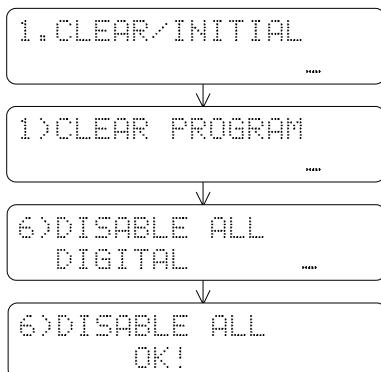
- Any keys except **ENT** will cancel the function

#### 2.4.1.6 Disable All Digital

## 【Key Sequence】



## 【LCD Display】



## 2.4.2 COPY/COMPARE

(Only for FP-07B)

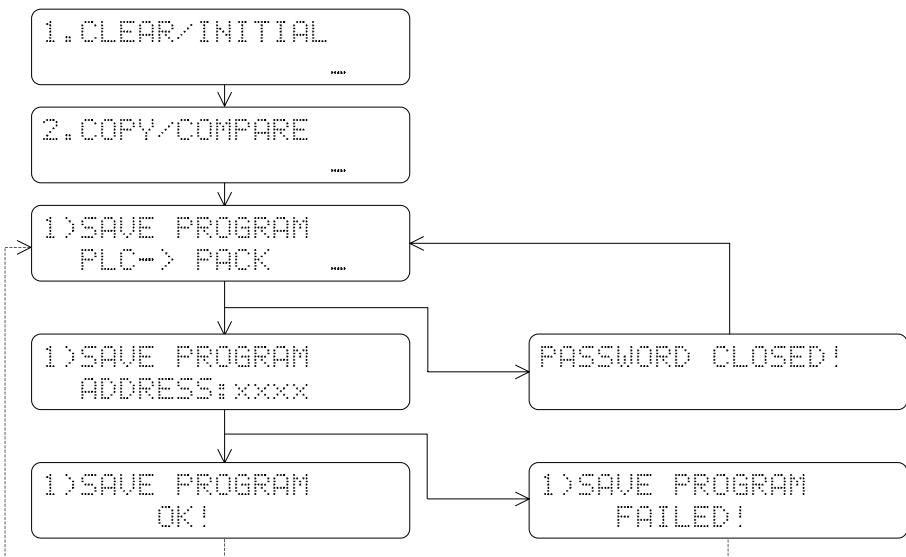
#### ⑩ 2.4.2.1 Save Program (PLC→PACK)

## 【Key Sequence】

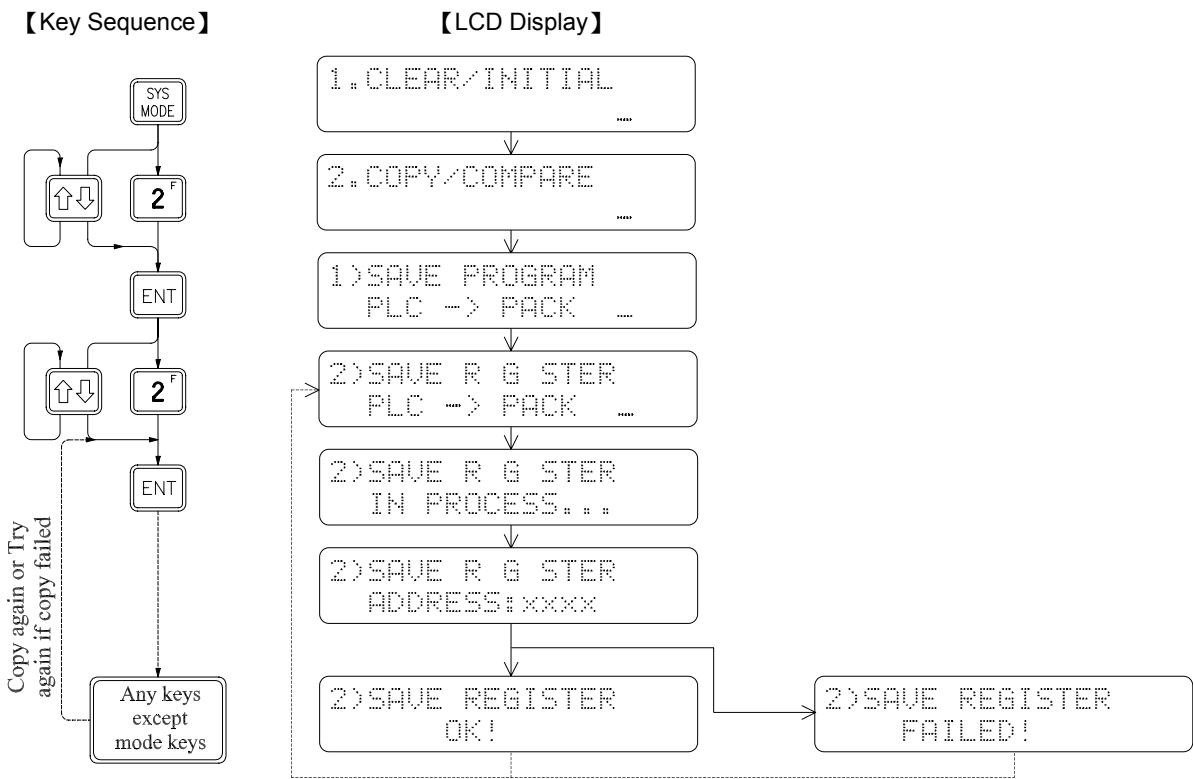
Copy again or Try again if copy failed

Any keys  
except  
mode keys

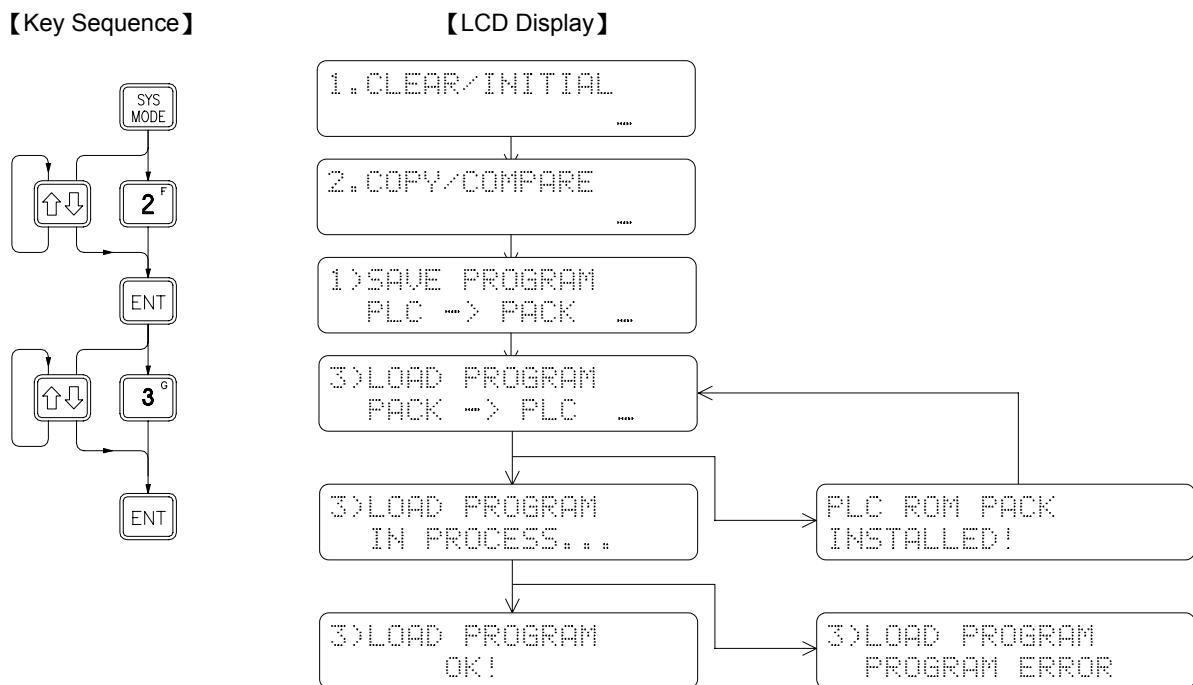
## 【LCD Display】



### 2.4.2.2 Save Register (PLC→PACK)

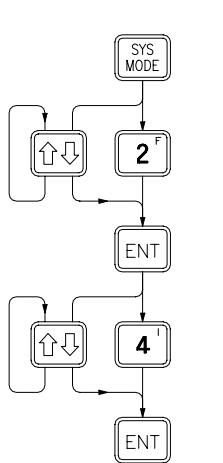


### 2.4.2.3 Load Program (PACK→PLC)

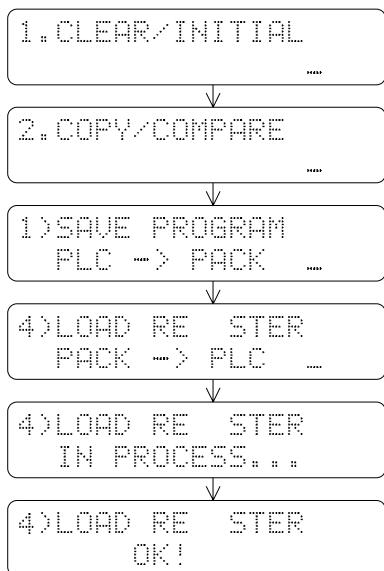


#### 2.4.2.4 Load Register (PACK→PLC)

## 【Key Sequence】

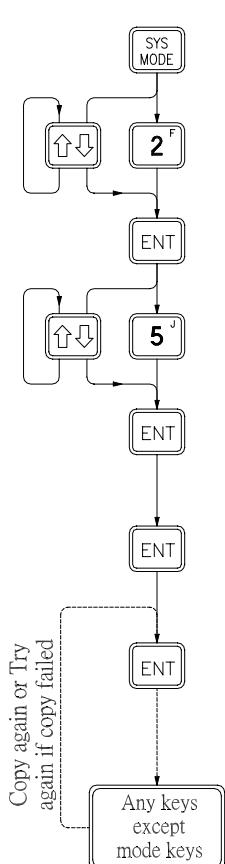


## 【LCD Display】

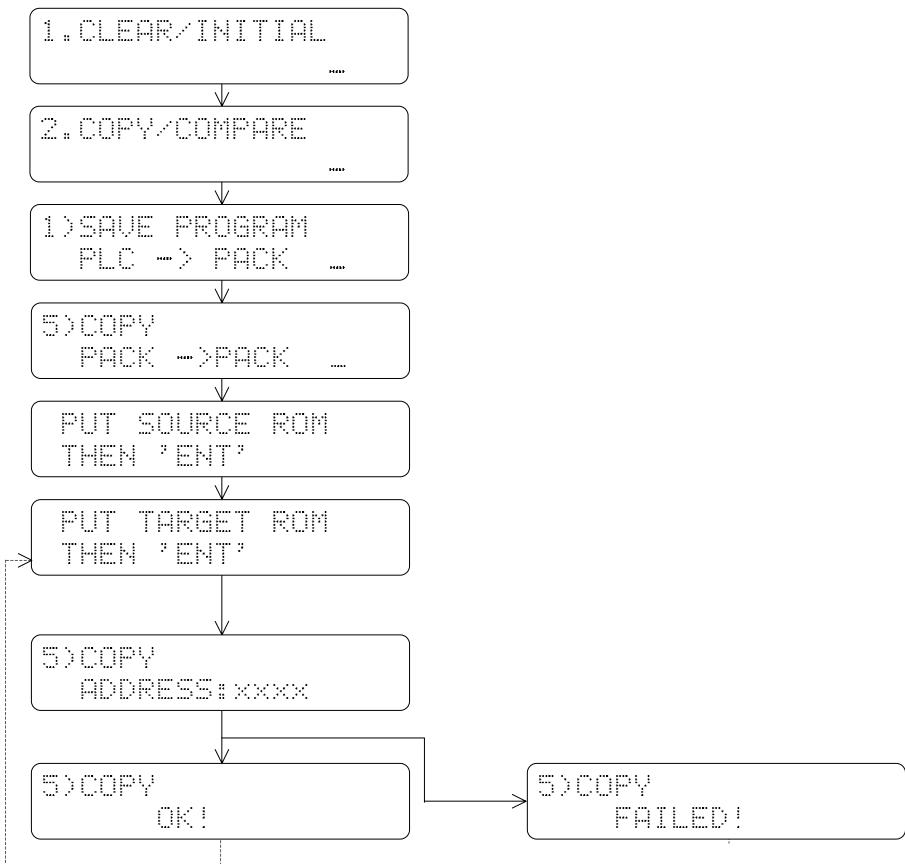


#### 2.4.2.5 Rom Pack Copy (PACK→PACK)

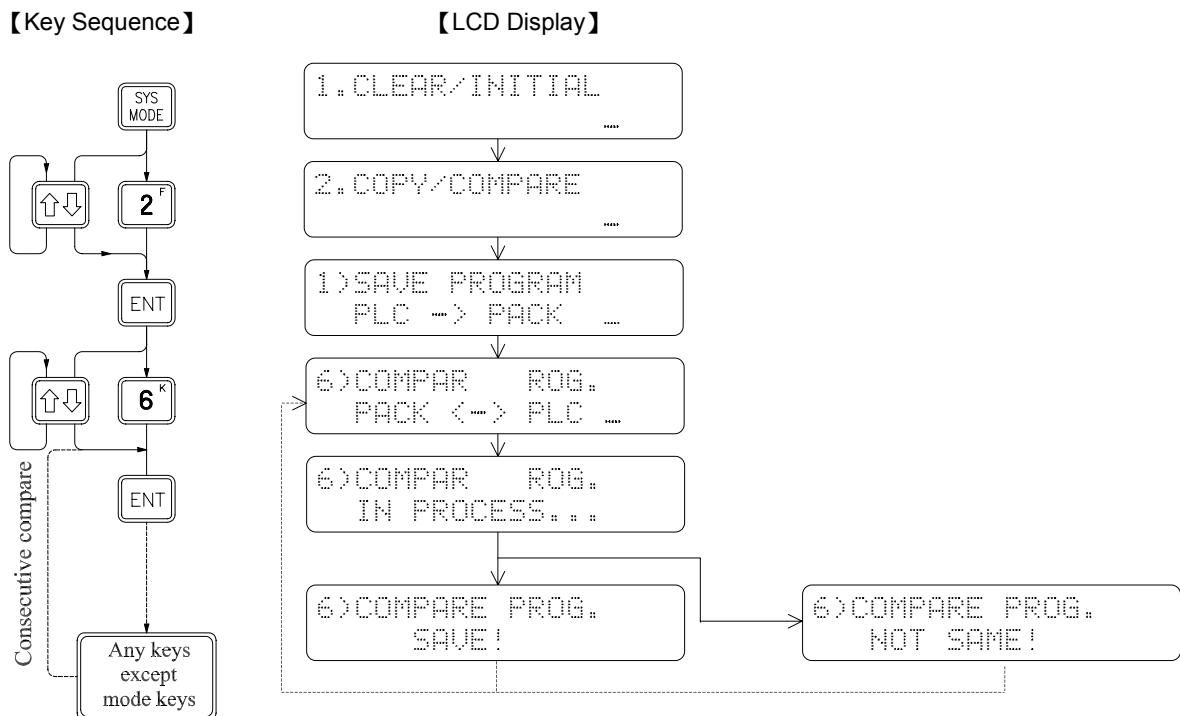
## 【Key Sequence】



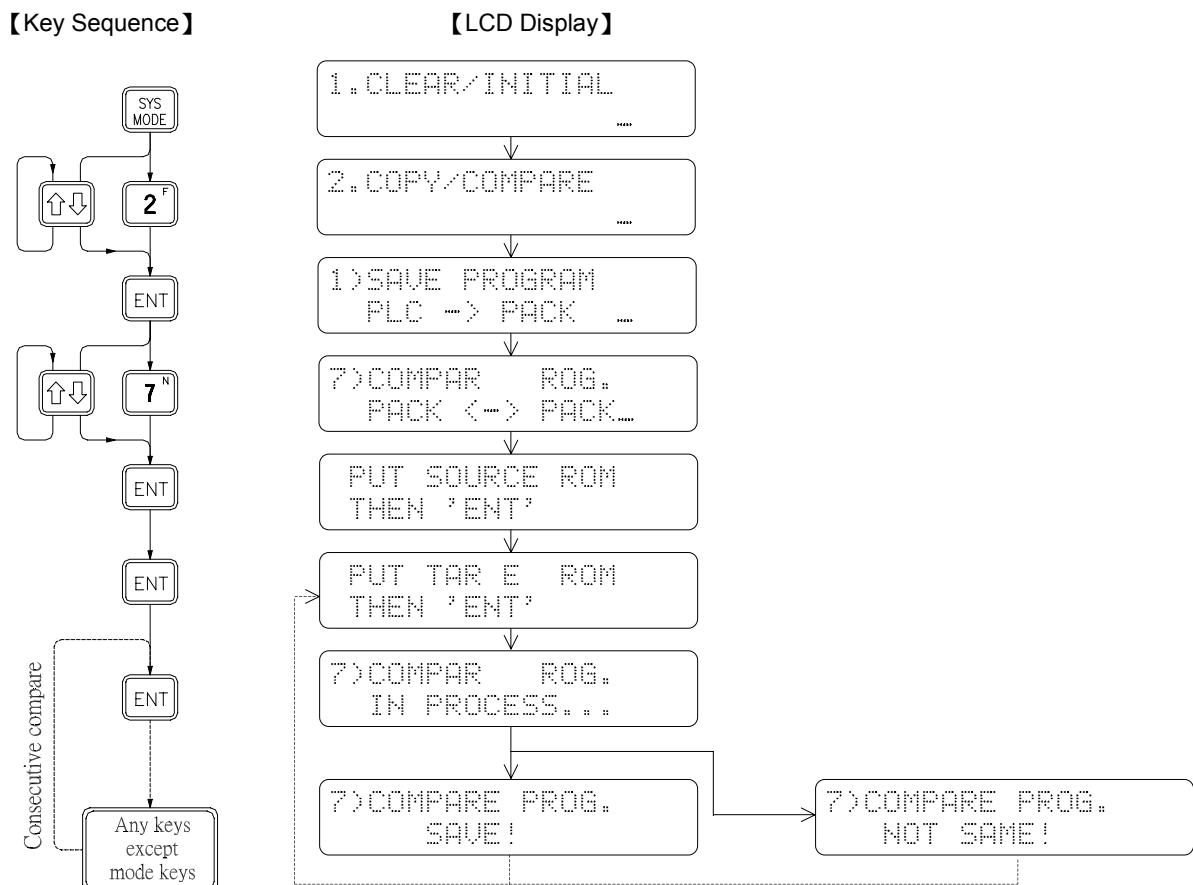
## 【LCD Display】



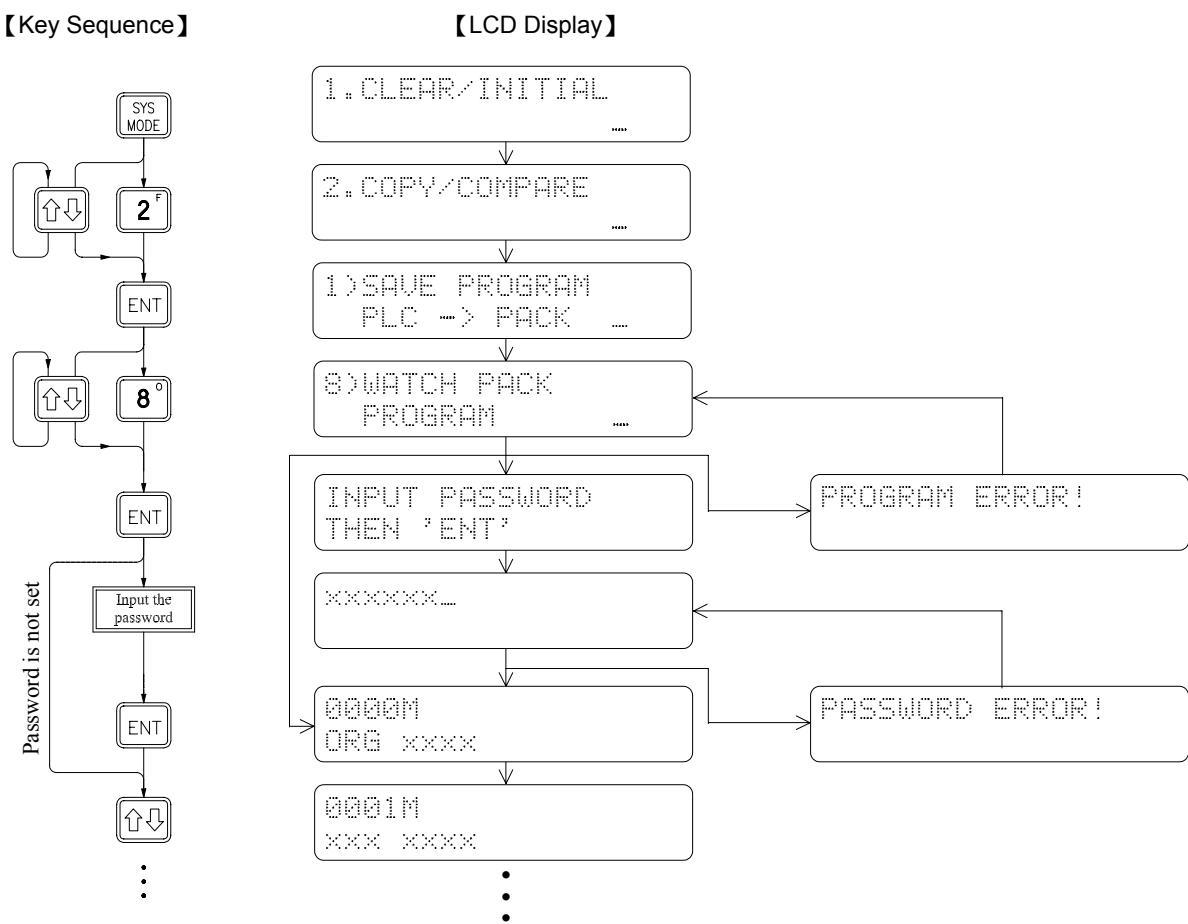
#### 2.4.2.6 Compare Program (PACK↔PLC)



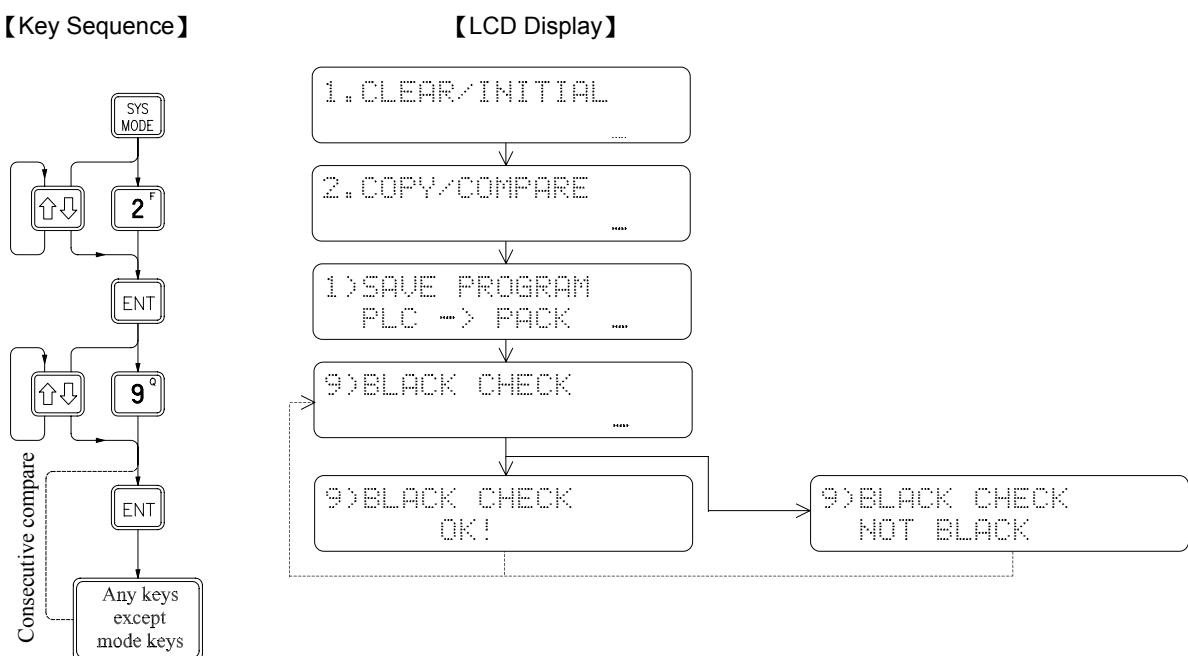
#### 2.4.2.7 Compare Program (PACK↔PACK)



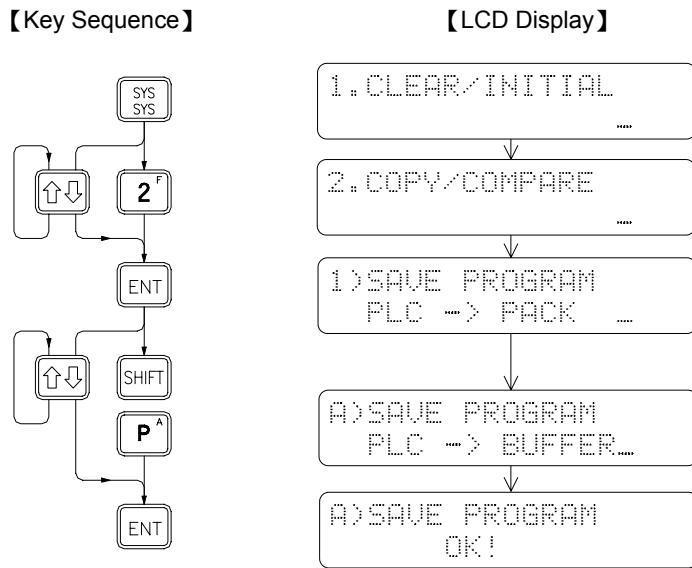
#### ⑧ 2.4.2.8 Watch Pack Program



#### 2.4.2.9 Rom Pack Blank Check



#### 2.4.2.10 Save Program (PLC→BUFFER)



This function uses the super capacitor in FP-07B to retain the ladder program while the FP-07B is power off. The program contents can be stored in the BUFFER area and last for 3-4 days if execute this function before turning the device off. If the FP-07B is connected to other PLC, the user can select to download the programs in the BUFFER area into the other PLC to achieve the similar effects of EEPROM program save/load. The store function in the BUFFER area will be cancelled if any other function in the system is executed.

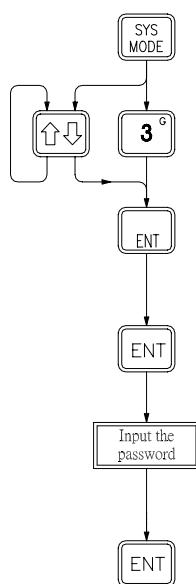
### 2.4.3 PASSWORD/ID Setting

- **PASSWORD:** The password prevents unauthorized access to the program. In order to edit, read and copy the program, you have to open the password first if it has been set. Otherwise the FP-07 will prohibit the user from executing such operations. Even you have opened the password, the PLC will automatically return to the password close mode if the power is turned off. Users can freely execute all FP-07 operations without any restrictions if the password has not been set.
- **ID:** The ID code is used to prevent the direct hard copy of the ROM PACK. Although the FP-07 can prevent unauthorized read or copy of its program if a password has been set, it can not prevent the direct hard copy of a ROM PACK using the EPROM writer sold in the market, because the EPROM does not have a read-protection design. You will not be able to read the ROM programs from this copied ROM pack if the password is closed, but it can perform the same operations as the original ROM upon inserting it into the PLC. This means a password is not enough to protect the PLC from unauthorized access of its program. Therefore we add an ID design for both ROM and ID in our FB-PLC. Before starting any PLC operations, the system will check the program ID in ROM first and then will compare it to the PLC ID. The PLC will not start its operation if the two ID codes do not match. Normally it is impossible to copy the PLC ID code which were stored in the PLC, therefore a hard copied ROM will not be able to operate on a PLC if the ROM ID does not match the PLC ID.

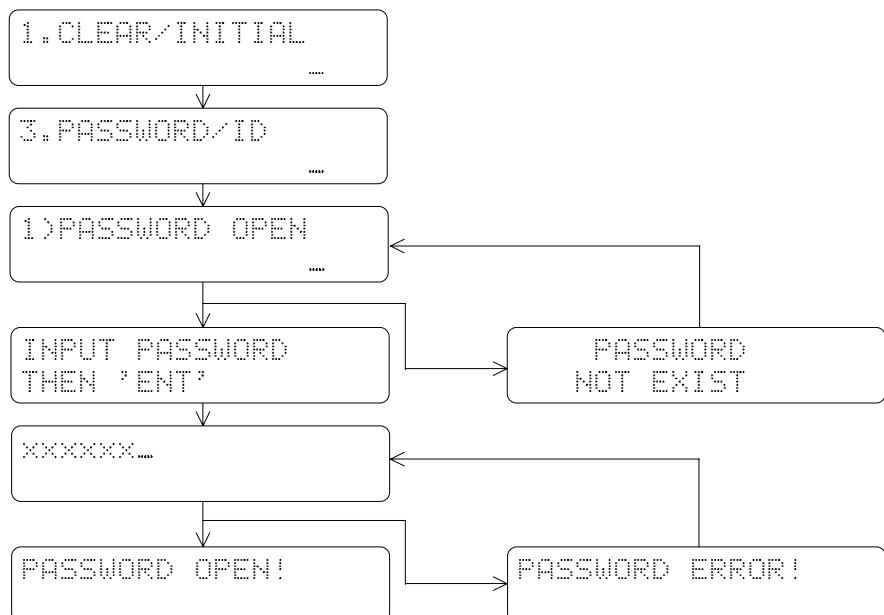
**Remarks:** If you set the ID without the password, it only prevents the Hard Copy of the ROM PACK, and the program can still be read out freely thus enabling an identical program without a specified ID code to be reproduced, and the reproduced program may operate normally upon downloading it into the PLC. Therefore, if you store the program using the RAM PACK, it is also necessary to set a password. If you store the program using the RAM inside the PLC instead of an external ROM, then the password protection is adequate enough.

#### ◎ 2.4.3.1 Password Open

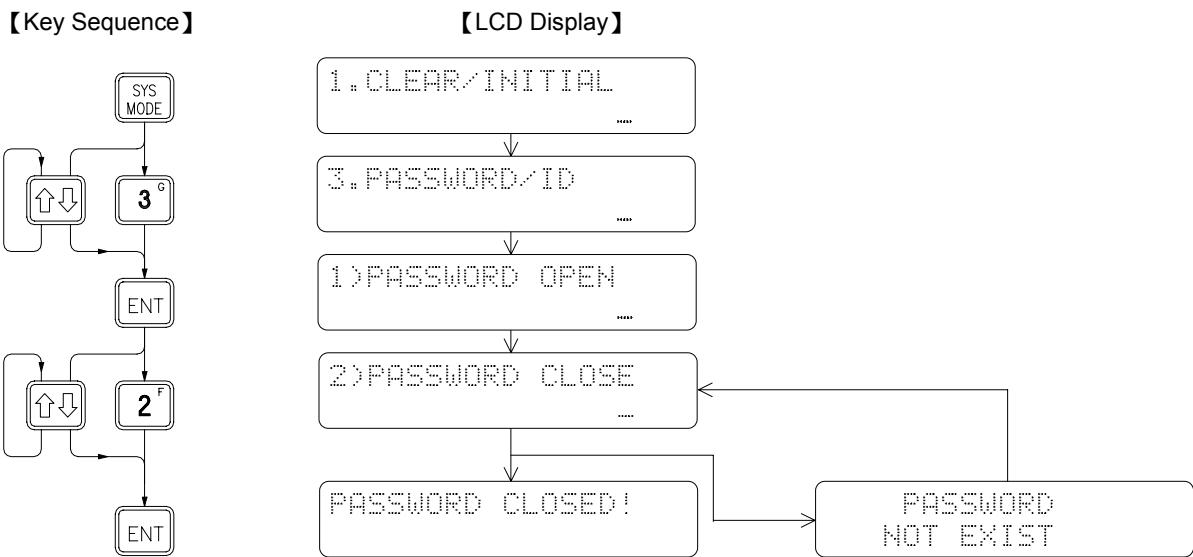
【Key Sequence】



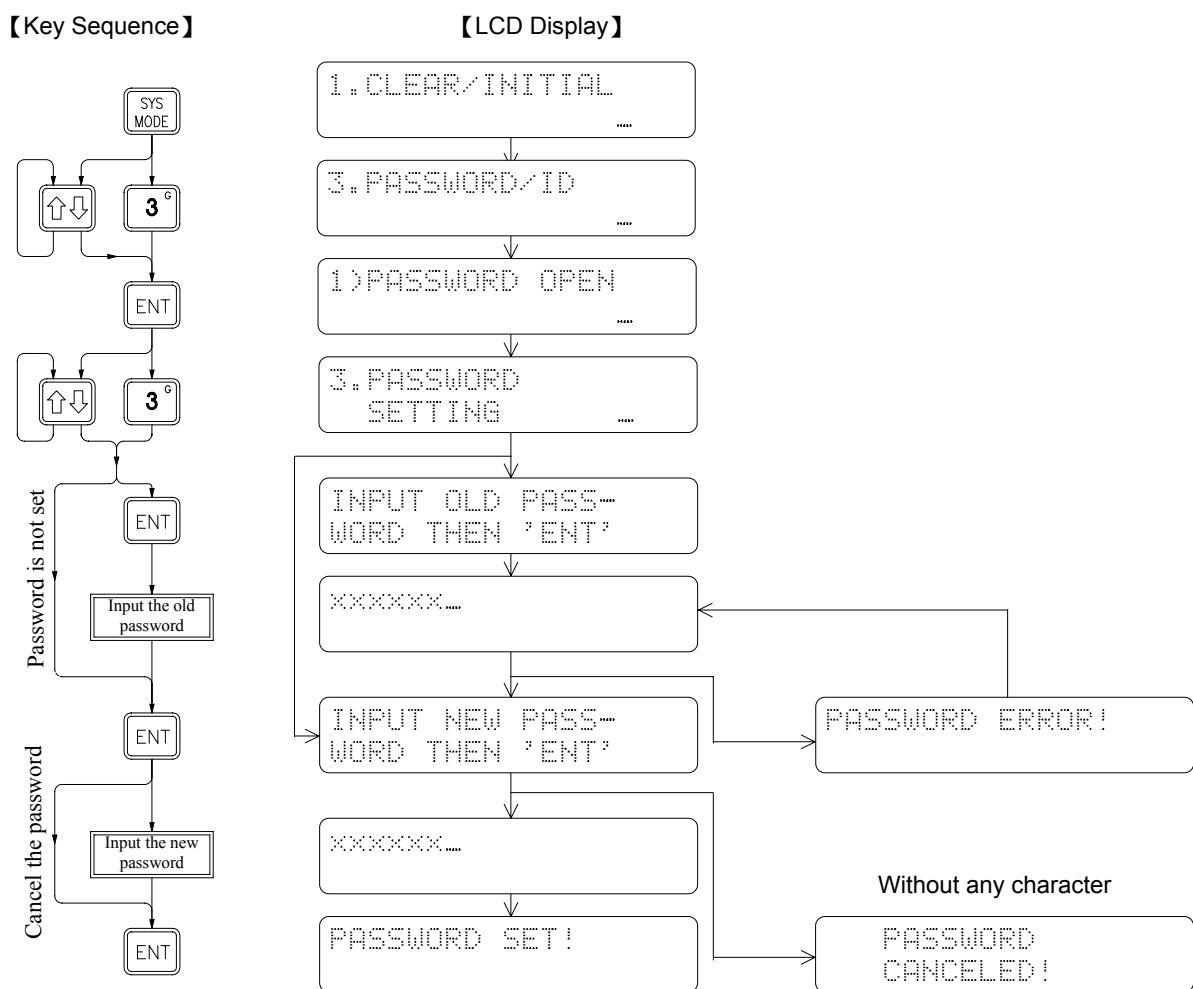
【LCD Display】



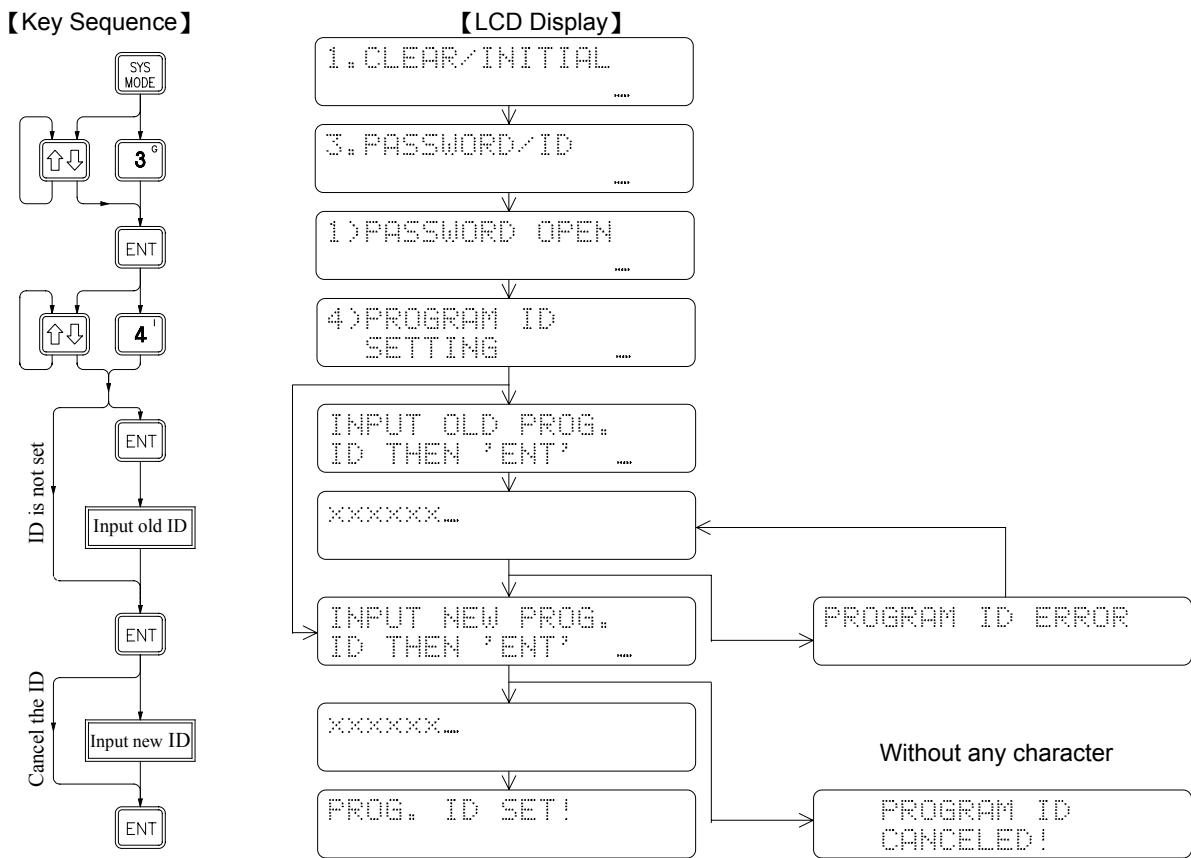
### 2.4.3.2 Password Close



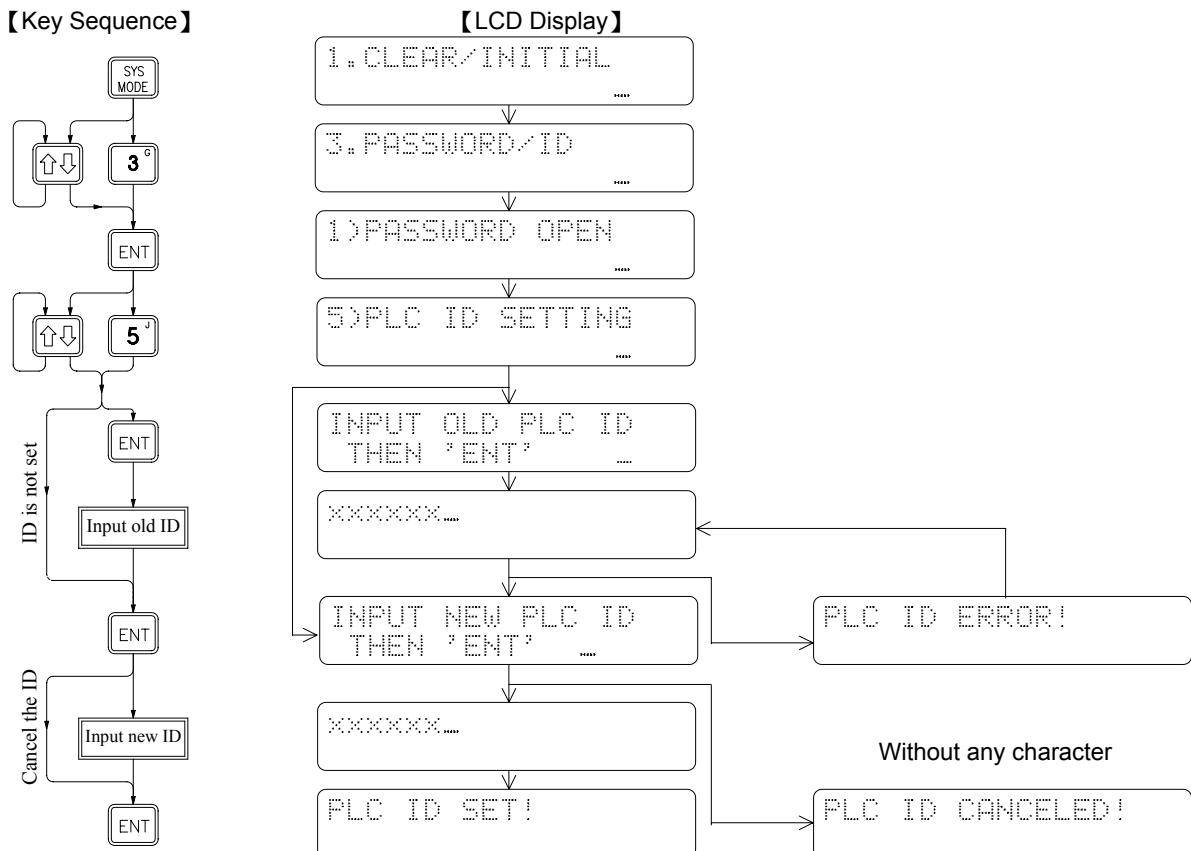
### ◎ 2.4.3.3 Password Setting



#### 2.4.3.4 Program ID Setting

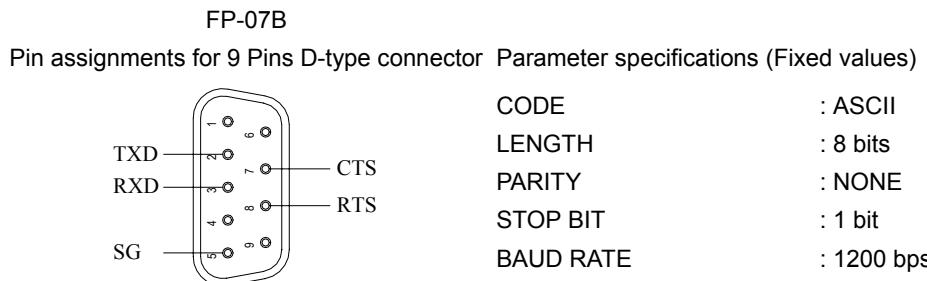


### 2.4.3.5 PLC ID Setting

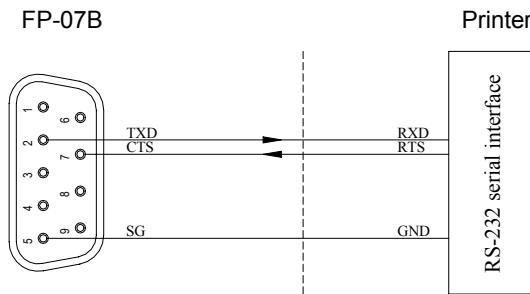


#### 2.4.4 Program and Register Data PRINT OUT (only FP-07B provides this function)

FP-07B is equipped with an RS-232 serial port. Program and register data printout is one of the three main functions available for this type of serial communication port. We set this type of serial communication port as a DTE (Data Terminal Equipment) interface. The wiring diagram for the RS-232 connector and the communication parameters setting for the printer interface are shown below.



- The printer interface of FP-07B uses TXD to transmit the program and data to the printer while CTS determines whether the printer is ready to receive the data sent by TXD. The printer uses RXD to receive the data sent from FP-07B via TXD. If the printer is ready to accept the data sent from FP-07B, RTS will send the READY signal to FP-07B via CTS. The schematic diagram shown below illustrates the wiring between the FP-07B and the printer via RS232 communication port.



- After finish wiring the signal lines shown in the above schematic diagram, it is also necessary to set the communication parameters in the printer to match the ones specified in FP-07B. Please keep in mind that you must adjust the communication parameters of the printer to match the ones in FP-07B because those parameters in FP-07B are fixed. The printer is ready to work after you set the printer parameters. The operations of the printer are demonstrated in the following section. If your printer does not have an RS-232 serial interface, consult the dealer to acquire a serial to parallel (RS-232 to Centronic) conversion adapter to connect your printer to FP-07B. Example of the signal lines wiring and the setting of parameters are shown below.

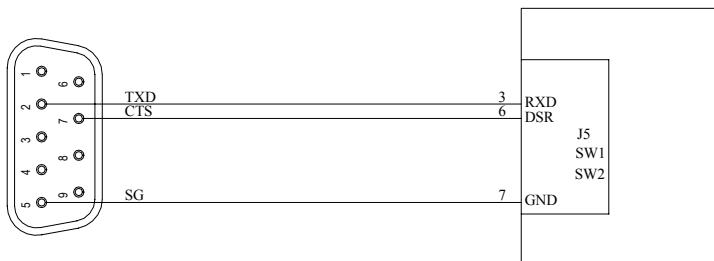
① Connection via RS-232 interface card (EPSON 8148)

**Wiring**

FP-07B

EPSON printer

Card 8148



SW1

**Setting**  
J5: ON  
JRS: ON  
J8: ON

1	2	3	4	5	6	7	8
O	O	O	O	O	O	O	O
F	F	F	F	N	F	F	F
F	F	F	F	F	F	F	F

SW2

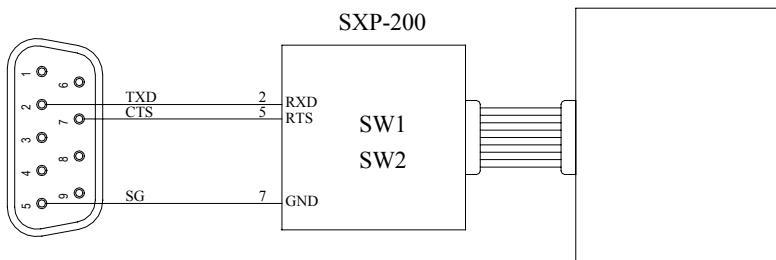
1	2	3	4	5	6
O	O	O	O	O	O
N	N	F	F	F	F

② Connection via serial to parallel conversion adapter (SXP-200)

**Wiring**

FP-07B

Parallel printer interface



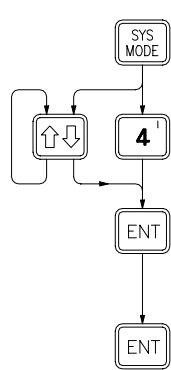
SW1

**Setting**

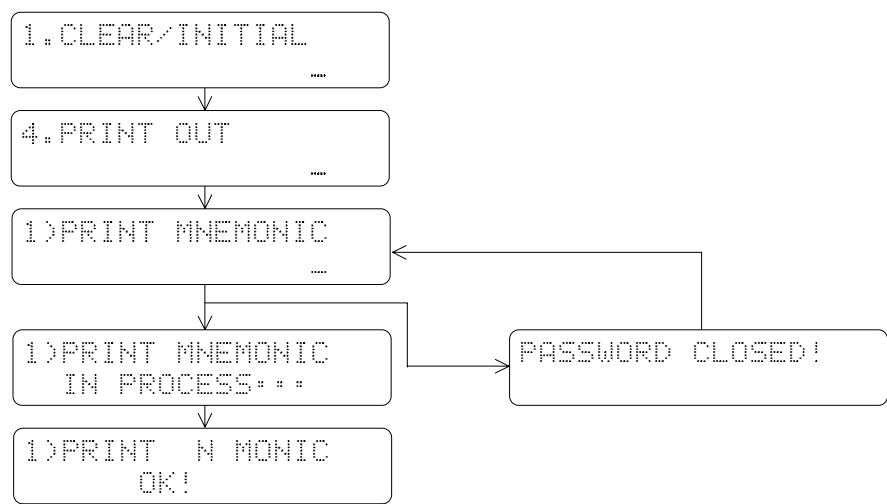
1	2	3	4	5	6	7	8
O	O	O	O	O	O	O	O
F	F	F	F	N	F	N	F
F	F	F	F	F	F	F	F

◎ 2.4.4.1 Print Mnemonic

【Key Sequence】

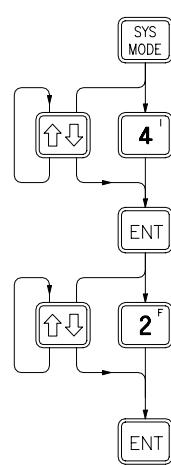


【LCD Display】

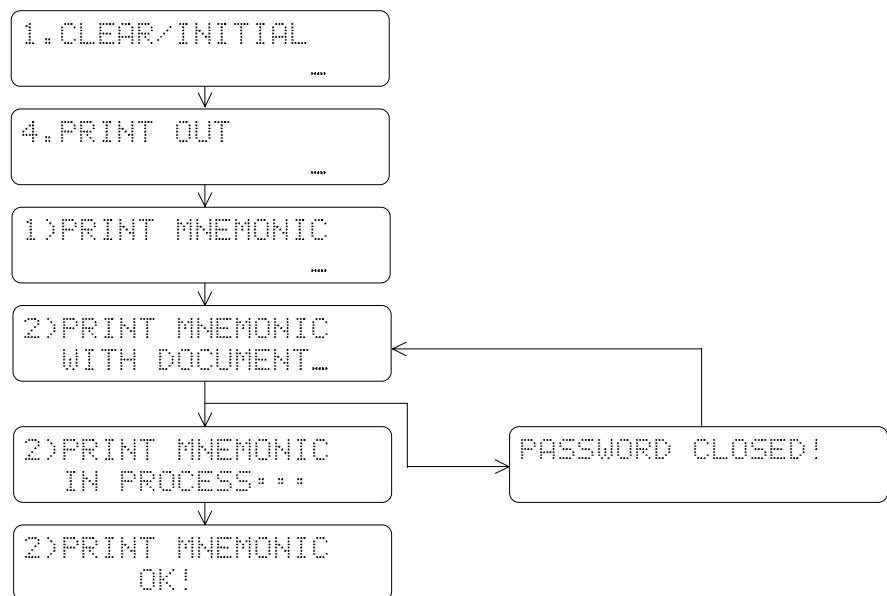


◎ 2.4.4.2 Print Mnemonic with Document

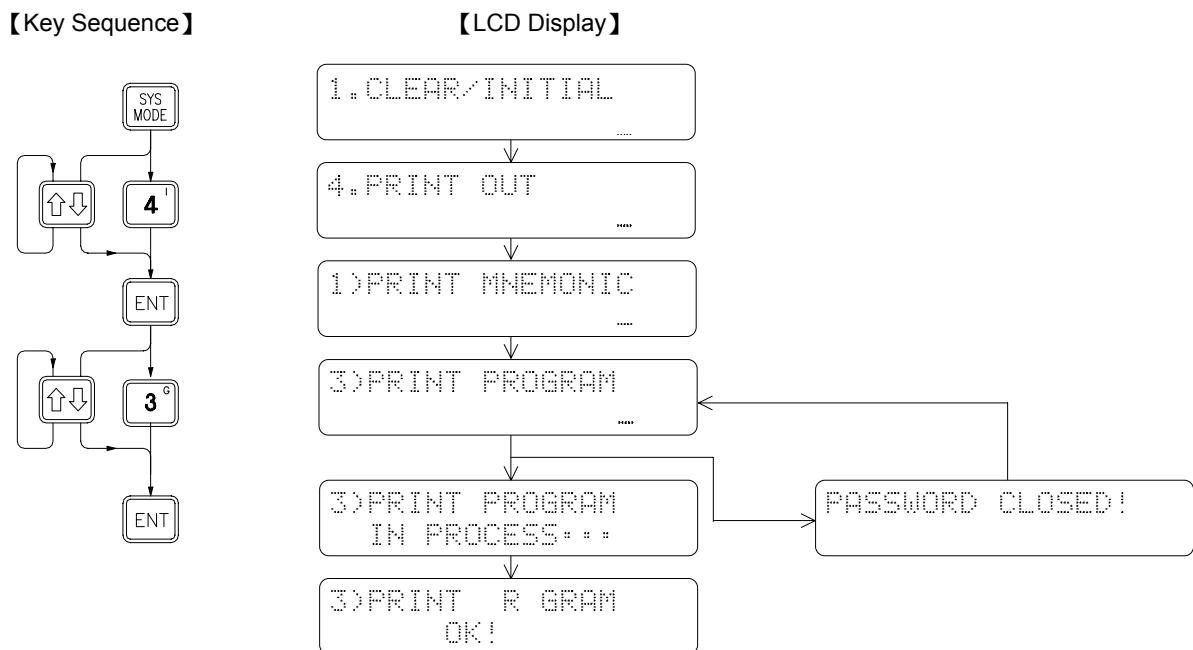
【Key Sequence】



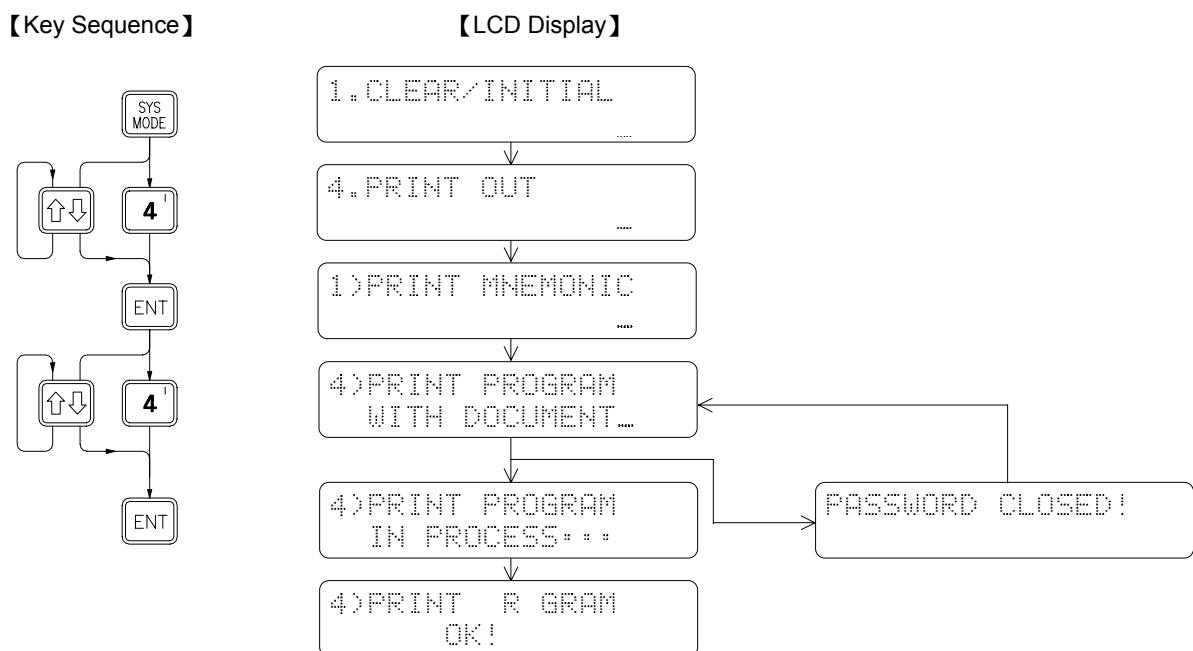
【LCD Display】



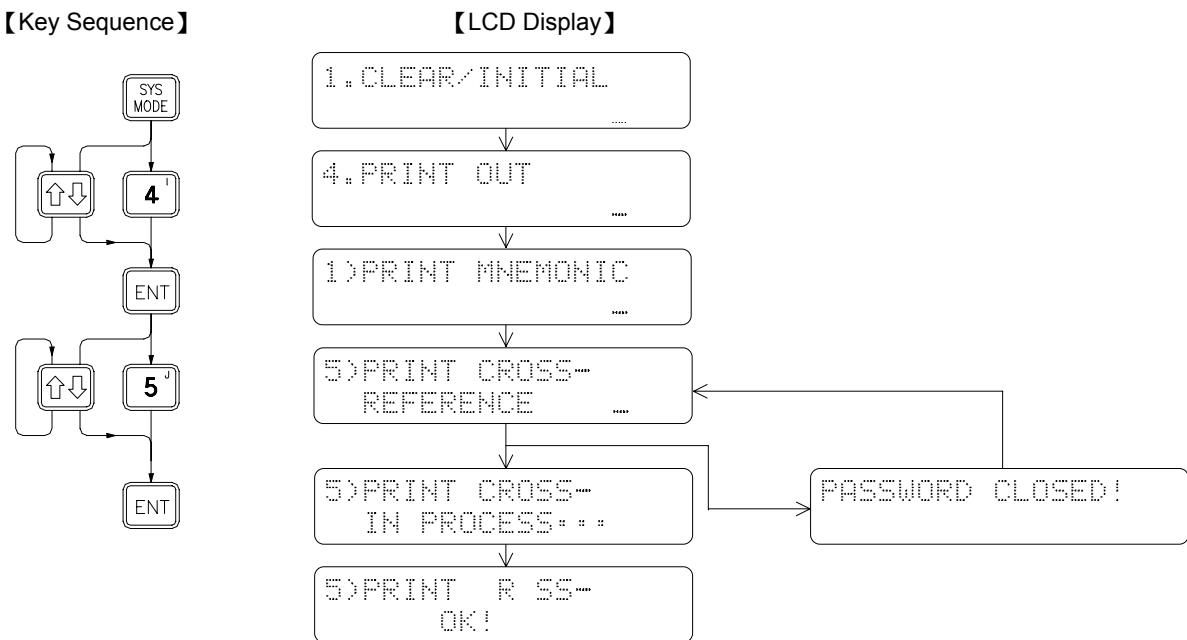
◎ **2.4.4.3 Print Program**



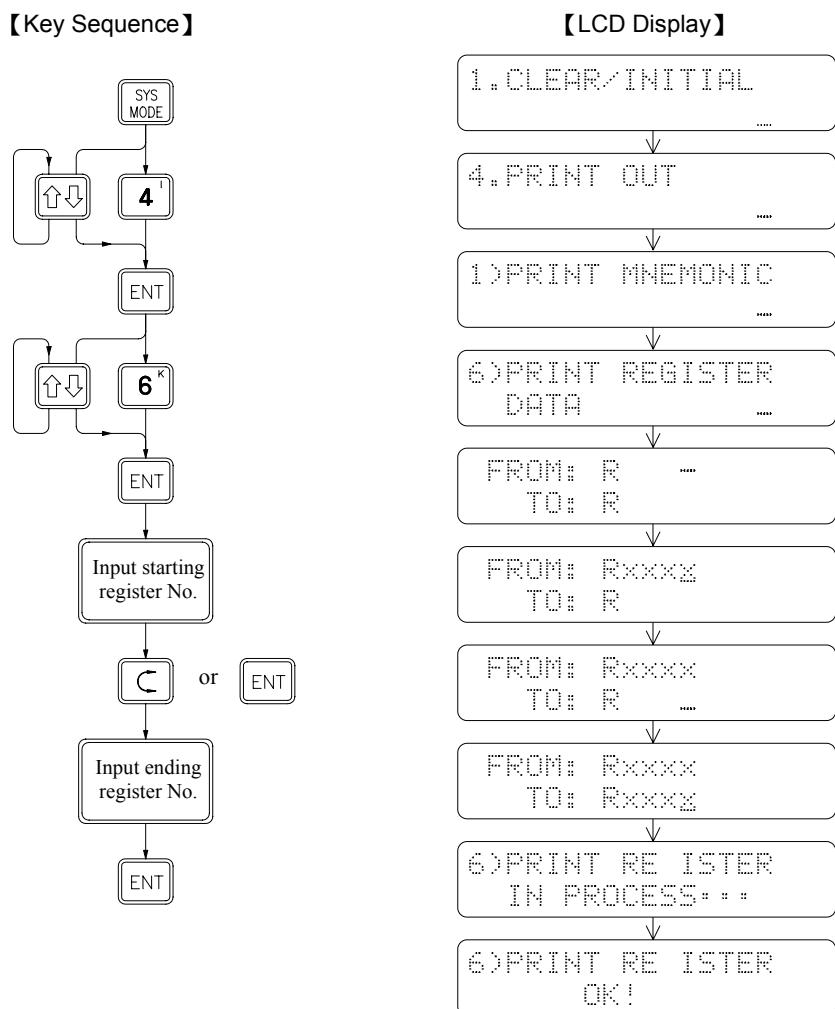
◎ **2.4.4.4 Print Program with Document**



#### ⑧ 2.4.4.5 Print Cross-Reference



#### 2.4.4.6 Print Register Data



## 2.4.5 CONFIGURATION

(For beginners, please skip this function)

The initial system configurations of FB-PLC, such as the Retentive/Non Retentive coils and registers partition and ROR assignment have already been set and adjusted for the best device performance. We call this initial setting as "Default Configuration". It is not necessary to reset or to readjust the default configurations for most applications. In order for the system to handle other special operations, a Configuration Setting function is provided for the users to readjust the configurations according to their needs.

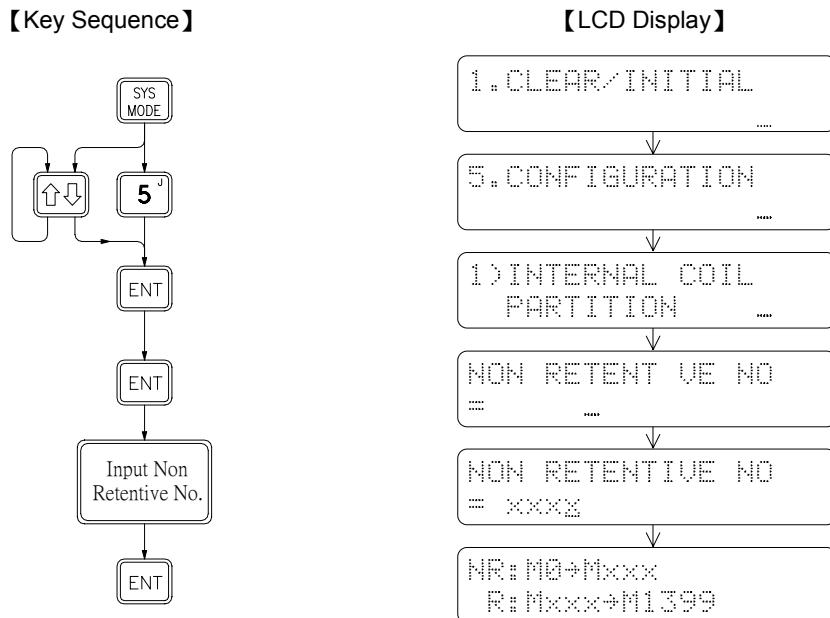
Listed below are the "Default Configurations" and adjustable ranges by the "configuration" function:

Configuration items		Default configuration	Valid range	Remarks
Internal Coil	Non Retentive	M0~M799	M0~M1399	M1400~M2001 are non retentive
	Retentive	M800~M1399	M0~M1399	
Step Coil	Non Retentive	S0~S499	S20~S999	Step points S0~S19 are fixed for non retentive
	Retentive	S500~S999	S20~S999	
Timer*	0.01S	T0~T49	T0~T255	
	0.1S	T50~T199	T0~T255	
	1S	T200~T255	T0~T255	
16-Bit Counter	Retentive	C0~C139	C0~C199	
	Non Retentive	C140~C199	C0~C199	
32-Bit Counter	Retentive	C200~C239	C200~C255	
	Non Retentive	C240~C255	C200~C255	
Data Register	Retentive	R0~R2999	R0~R3839	D0~D3171 are always retentive
	Non Retentive	R3000~R3839	R0~R3839	
Read-Only Register		0	R5000~R8071	
High Speed Timer (0.1ms)		R4152~R4154	Unchangeable	
High Speed Counter		0	HSC0~HSC7	
External Interrupt		0	INT0~INT15	
Station number		No.1	No.1~No.255	

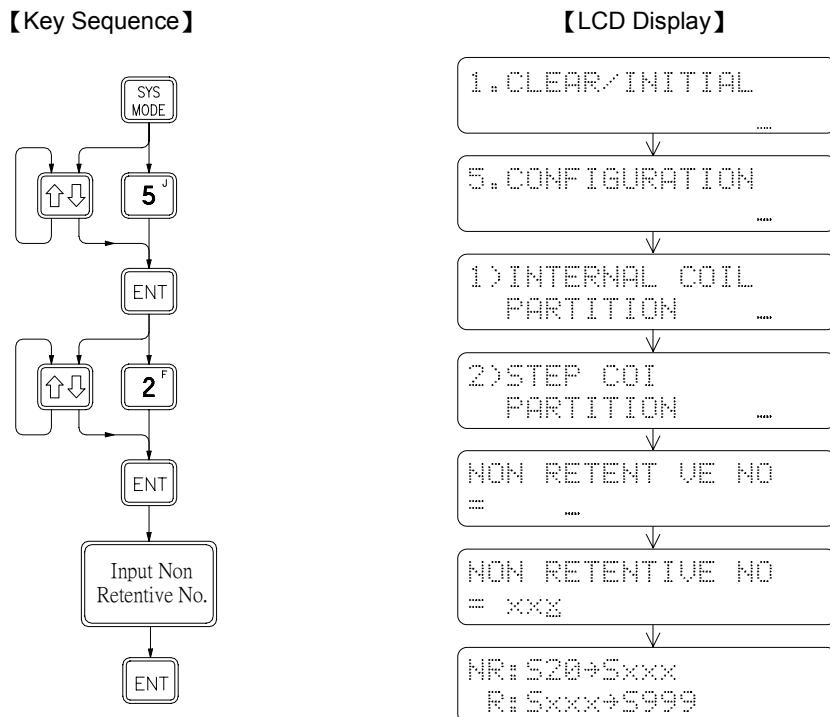
**Remark 1:** For the items marked with "\*", can only be modified while PLC is at initial state. After the program had written to the PLC, changing of these two items is prohibited. The only way to change configurations after had written a program into the PLC is to perform the system initial operation, which means you will lose all the programs and get a defaulted configuration again. Please pay more attention on this.

**Remark 2:** The registers in the range of R5000~R8071 if not used for Read-Only registers, could be used as normal read and write registers.

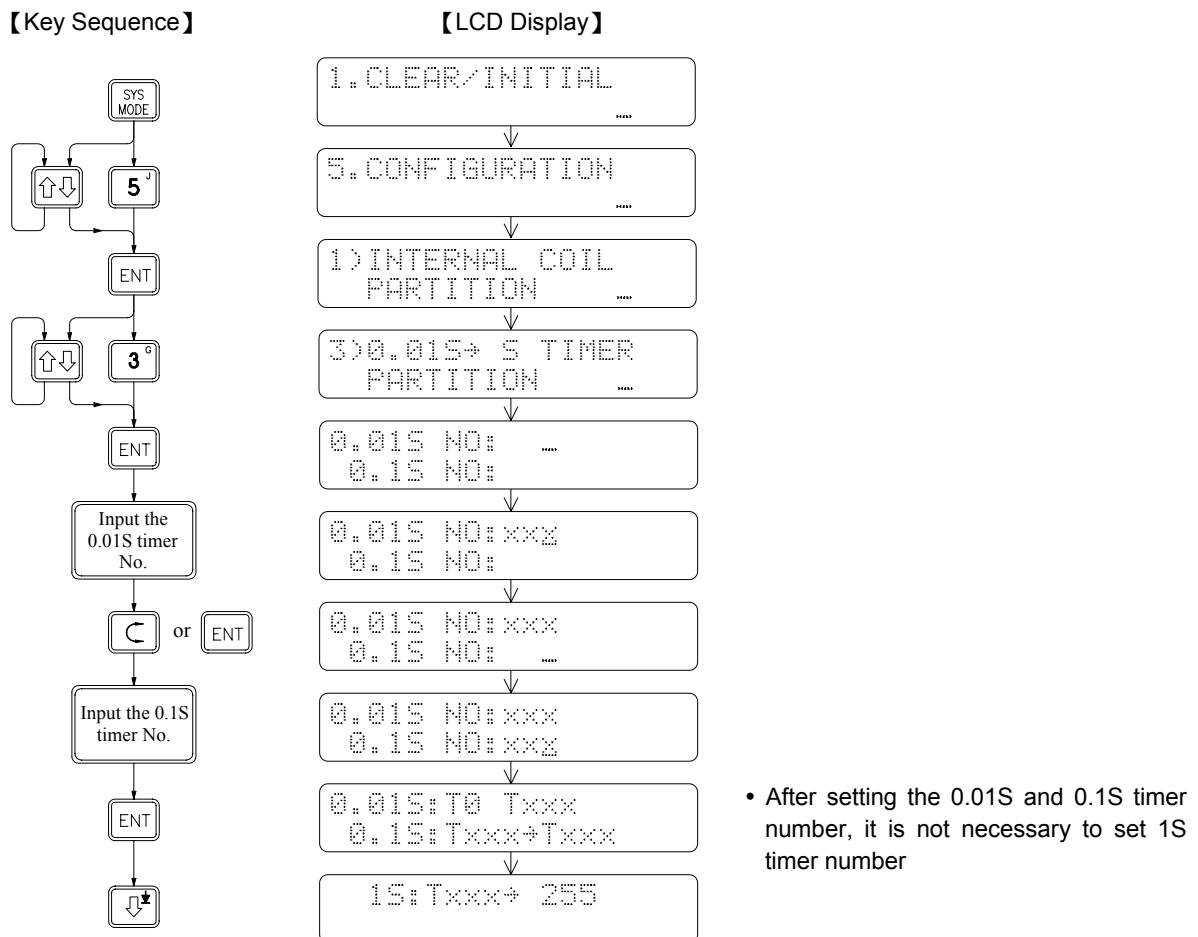
#### 2.4.5.1 Retentive/Non Retentive Internal Coil Partition



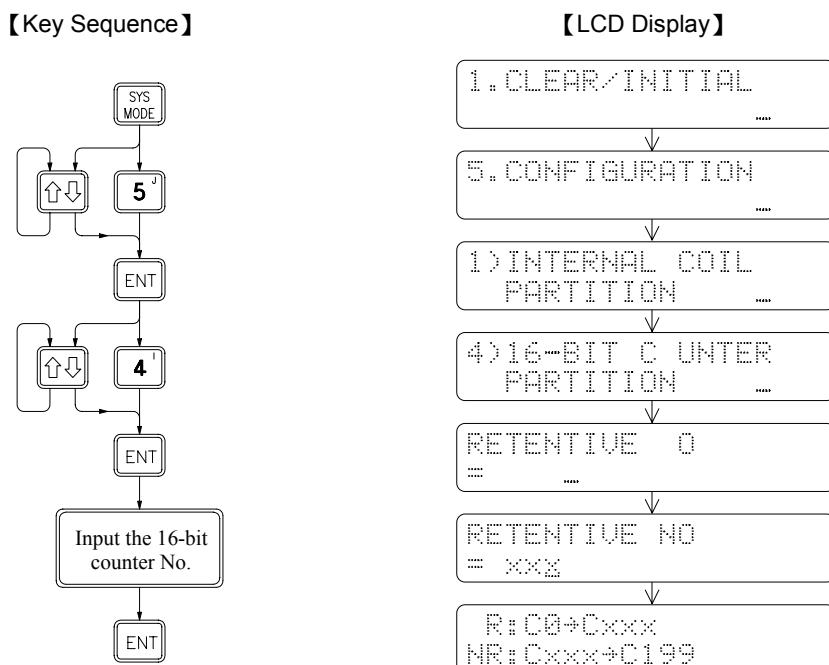
#### 2.4.5.2 Retentive/Non Retentive Step Coil Partition



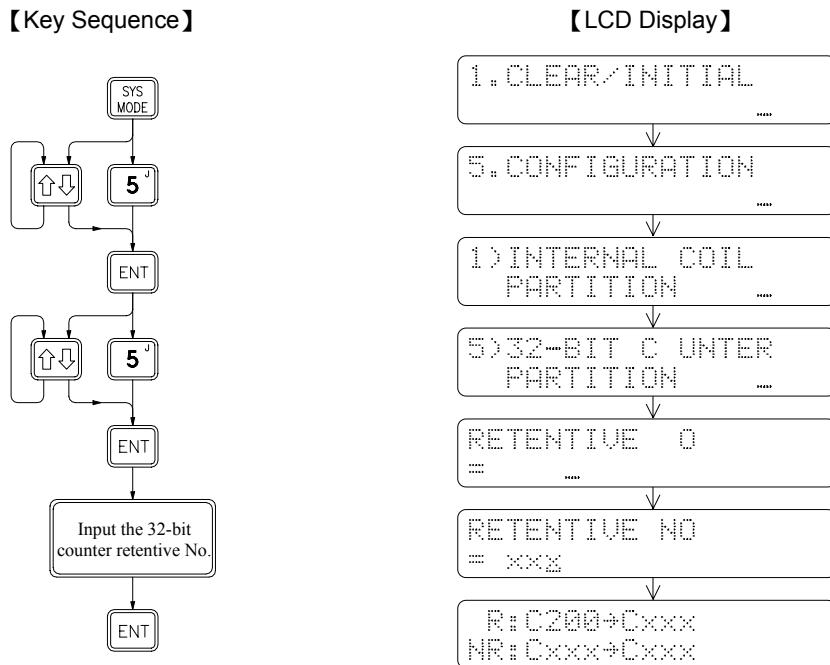
### 2.4.5.3 0.01S~1S Timer Partition



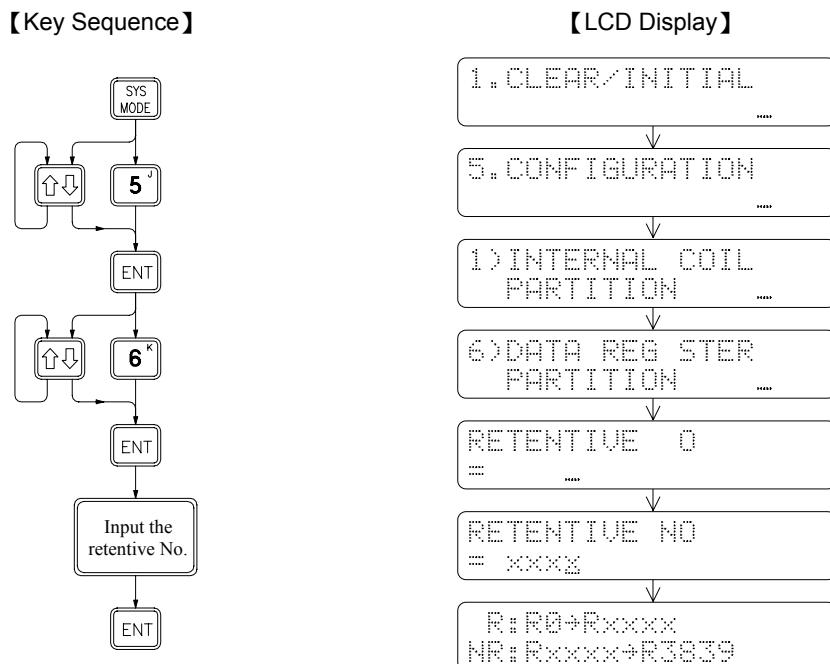
### 2.4.5.4 Retentive/Non Retentive 16-Bit Counter Partition



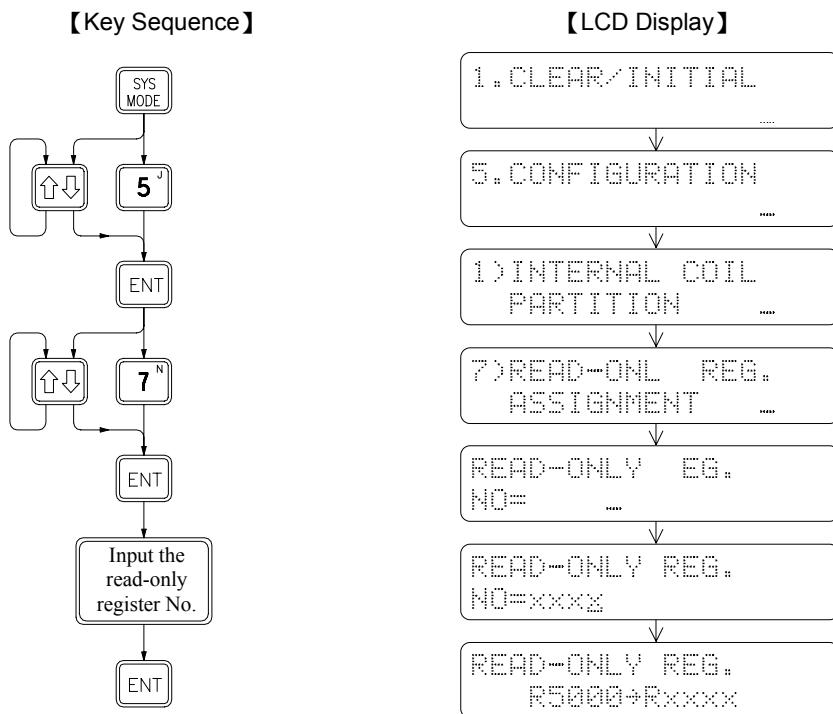
#### 2.4.5.5 Retentive/Non Retentive 32-Bit Counter Partition



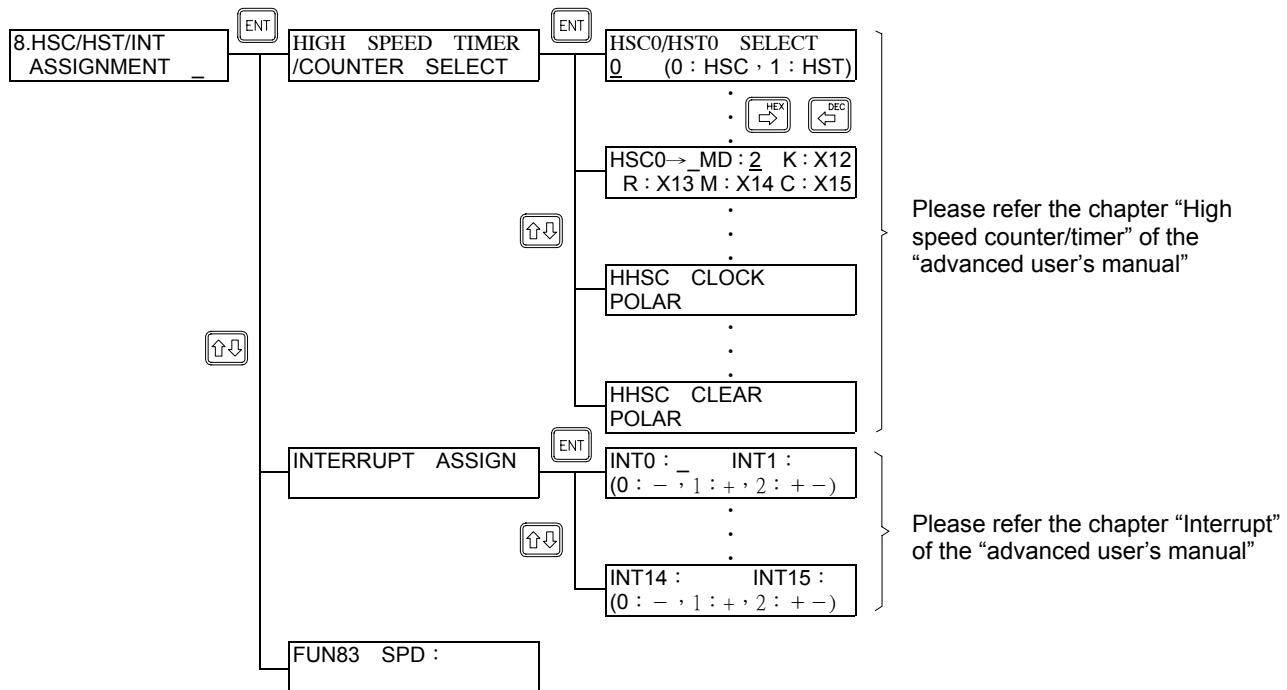
#### 2.4.5.6 Retentive/Non Retentive Data Register Partition



#### 2.4.5.7 Read-Only Register Assignment



#### 2.4.5.8 High-Speed Counter, High-Speed Timer (TB=0.1mS) and External Interrupts

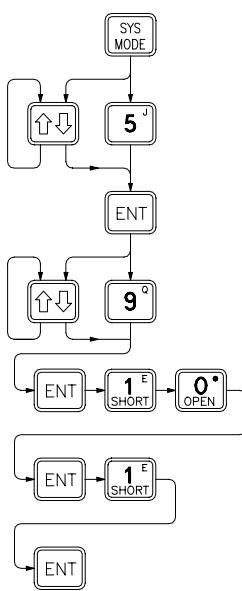


**Remarks 1:** There are 4 sets of hardware HSC0~3, and 4 sets of software HSC4~7 in FBE MC and FBN models, but only 2 sets of software HSC2 and HSC3 in MA model.

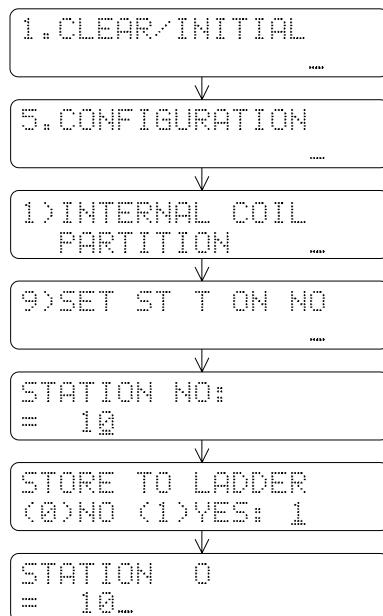
**2:** Counter MODE (MD) setting : 0 means U/D, 1 means U/D × 2 times precision, 2 means K/R, 3 means K/R × 2 times precision, 4 means A/B phase, 5 means A/B phase × 2 times precision, 6 means A/B phase × 3 times precision, 7 means A/B phase × 4 times precision.

#### 2.4.5.9 Station Number Setting

##### 【Key Sequence】



##### 【LCD Display】

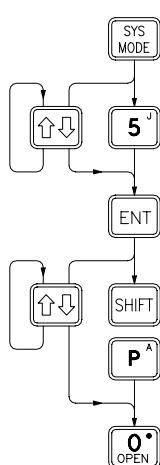


- Select whether to store the station number in the ladder area. If select, while Save the program to EPROM will also store the station number in it.

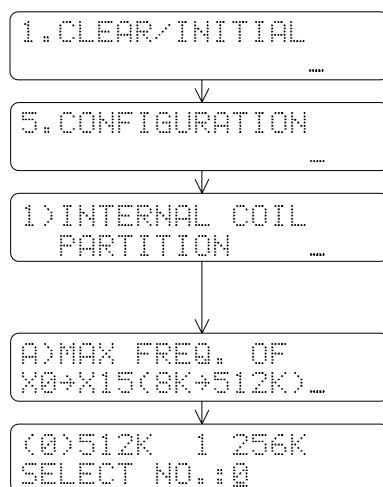
The station number of FBE MA main unit is always fixed as 1. The station numbers of MC and FBN main unit are also preset as 1 at factory. Any station number from 1 to 255 may be selected by using this function.

#### 2.4.5.10 Maximum Frequency Setting (X0~X15)

##### 【Key Sequence】

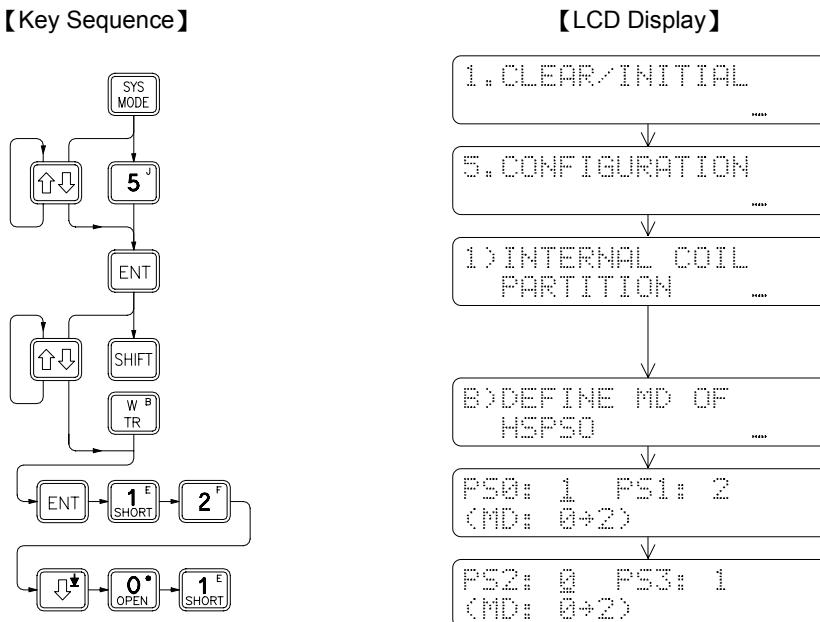


##### 【LCD Display】



- The first row will display the maximum input frequency(8K~512K)with rotation that allowable to set for X0~X15

#### 2.4.5.11 Define Mode of High-Speed Pulse Output (HSPSO)



There are four groups of HSPSO, PS0~PS3. Their corresponding output terminals are PS0(Y0 and Y1), PS1(Y2 and Y3), PS2(Y4 and Y5) and PS3(Y6 and Y7)

MD=0 (K/R) Mode : Y0 (Y2、Y4、Y6) pulse output

Y1 (Y3、Y5、Y7) direction output, where “ON” is the count-up and “OFF” is the count-down.

MD=1 (U/D) Mode : Y0 (Y2、Y4、Y6) up counting pulse output

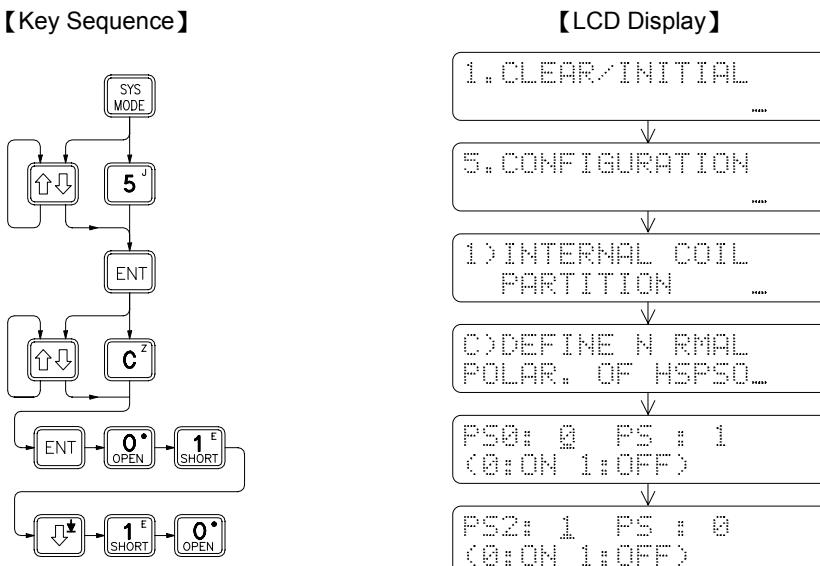
Y1 (Y3、Y5、Y7) down counting pulse output

MD=2 (A/B) Mode : Y0 (Y2、Y4、Y6) phase-A pulse output

Y1 (Y3、Y5、Y7) phase-B pulse output

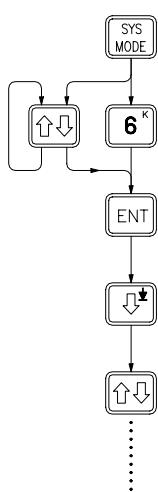
#### 2.4.5.12 Define Normal Polarity of High-Speed Pulse Output (HSPSO)

This function is to define the inactive state (On/Off) of output signal of HSPSO (PS0~PS3). The output signal of PS0, PS1, PS2 and PS3 are (Y0、Y1), (Y2、Y3), (Y4、Y5) and (Y6、Y7) respectively.

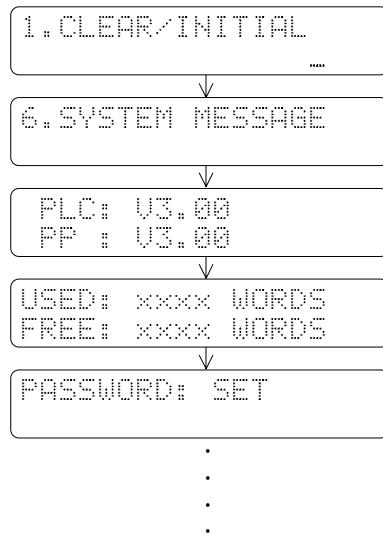


## 2.4.6 SYSTEM MESSAGE

【Key Sequence】



【LCD Display】

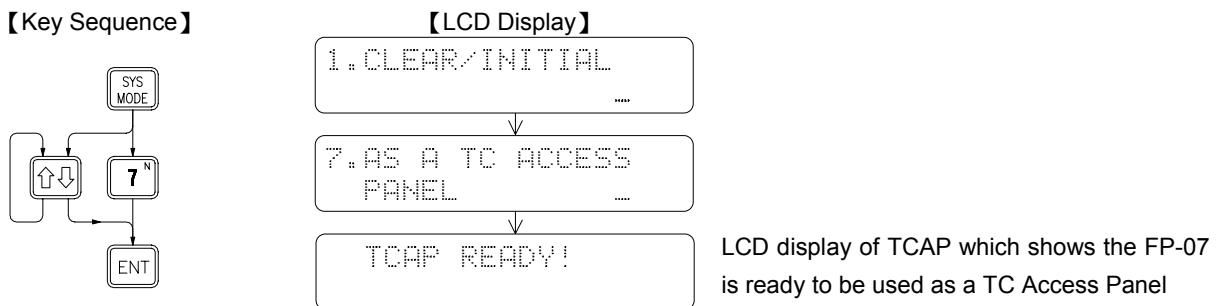


## 2.4.7 AS A TC ACCESS PANEL

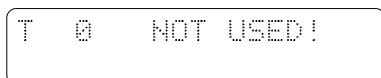
As we have mentioned before that FP-07 can be used either as PP or as TCAP. The PP/TCAP flag inside the PLC determinates the mode to be operated. The flag is set to PP before the shipment of FB-PLC or after the system initial operation were performed. After FP-07 is connected to PLC, FP-07 will check this PP/TCAP flag in PLC first. If the flag is set to PP, then FP-07 will operate as a Programming Panel. If the flag is set to TCAP, then the FP-07 will operate as a TC Access Panel. The status of PP/TCAP flag is retained unless you change it deliberately or reset the flag during the system initial operation. Following we would like to introduce the operations of FP-07 using as a TC Access Panel.

### 2.4.7.1 Change PP to TCAP Mode

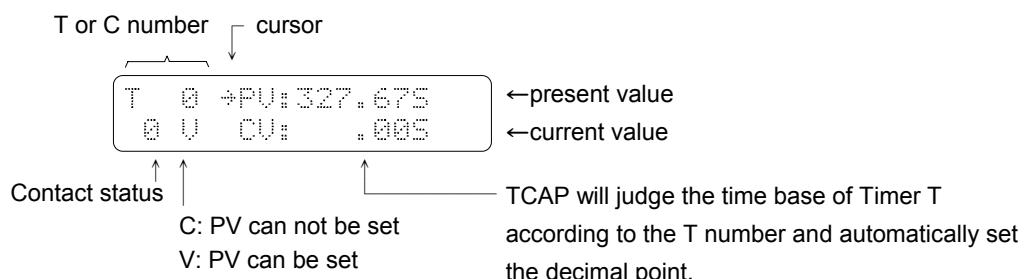
Once FP-07 connected to the PLC, if the PP/TCAP flag is set to PP, you can change the flag to TCAP by performing the following key operations.



After the LCD display shows the message "TCAP READY", you are now allowed to input the numbers of T or C. Once you confirmed the inputs, press **ENT**. If the ladder program does not contain the T/C numbers you have entered, then the TCAP will only display a brief message on the LCD screen to tell you the T or C are not used as shown in the display below.

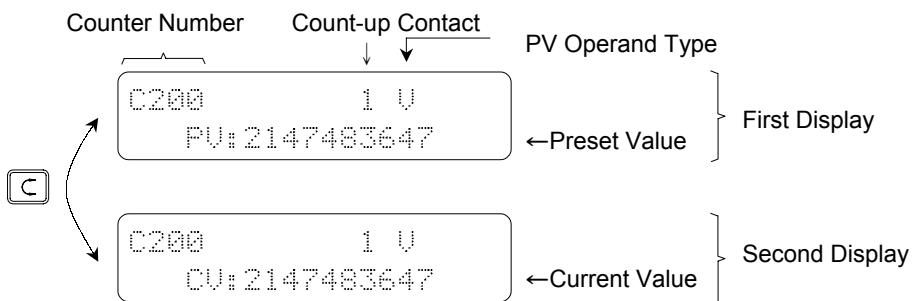


If the program contains the T/C number you have inputted, then TCAP will display the preset value PV, current value CV and T/C contact status on LCD screen. The data size of Timer and Counter (C0~C199) is 16-bit. All those messages are displayed on LCD screen as shown in below.



The contact status shown in the LCD display above is the time-up or count-up output contact of T (Timer) or C (Counter). The letter after the contact status display indicates the operand type of PV preset value. C indicates the PV value of T/C is a constant operand (number or WX) which can only be displayed but cannot be set. V indicates the PV value is a variable operand (R, WY, WM, WS and so on) which can be displayed as well as changed. If PV (must become a V operand first) or CV value is to be changed, the cursor must be moved to where PV or CV value is displayed. Cursor can be moved back and forth between PV/CV rows by pressing the **C** key.

Since a single LCD screen is not adequate enough to display the PV value, the CV value and the contact status of 32-bit C200-255 counters, therefore it is necessary to display these data in two separate screens. The first row contents of both screens are the same with the number C, the contact status and the PV operand type being displayed. The PV value and the CV value are displayed on the second row of screen 1 and screen 2 respectively. Two screens can be switched back and forth by using the **C** key.



As soon as FP-07 enters TCAP mode, the data displayed (including the TC number and the current cursor position) will be stored in a specific PLC memory buffer where the data can be retained after a power failure. Therefore, even when the PLC is re-started after a power off or FP-07 is connected again after being unplugged, FP-07 will enter the TCAP mode automatically and the LCD will display the same screen as previous.

#### 2.4.7.2 Operation of TCAP Mode

The effective keys when FP-07 is in TCAP mode are listed below:

**T** **C** **R** **SHIFT** **S**: These keys are used for monitoring T × × ×, C × × ×, R × × × × and D × × × ×.

**0** ~ **9**, **SHIFT** + (**P** ~ **Z**): These keys are used for entering the address (number) or value being monitored.

**↑**, **↓**: These two keys are used for decreasing (shifting up) or increasing (shifting down) the number (address).

**CLR**: This key is used for clear the items being monitored.

**ENT**: This key is used for confirming the input address (number) or data.

**C**: This key is used for moving the cursor back and forth between PV and CV when T or C is being monitored or between the monitored items in the two rows when R or D is being monitored.

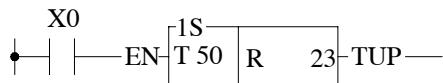
The table below shows the valid range of address and the value of TC and registers intended for display or setting when FP-07 is in the TCAP mode:

Items for display or setting			PV/CV Range	Remark
Timer T	0.01S Clock	T0~T49	0.00~327.67S	The items of time bases 0.01S~1S shown on the left are based on the "Default Configurations". If the configuration has been changed by CONFIGURATION function then the value in this table should be changed accordingly.
	0.1S Clock	T50~T199	0.0~3276.7S	
	1S Clock	T200~T255	0~32767S	
Counter C	16-Bit	C0~C199	0~32767	
	32-Bit	C200~C255	0~2147483647	
Register R	16-Bit	R0~R8071	-32768~32767	
	32-Bit	DR0~DR8070	-2147483648~2147483647	
Register D	16-Bit	D0~D3071	-327368~32767	The contents monitored can not be shown again after the power is turned off.
	32-Bit	DD0~DD3070	-2147483648~2147483647	

As described above, if PV wants to be set should be of V operand, or else it can only be displayed but cannot be changed. As long as T or C is in a non-reset status (i.e. the EN input of T is 1 or the CLR input of C is 0), the current CV value can be displayed and changed. As soon as a new CV value is entered and the **ENT** key is pressed, the current T/C value will start timed or counted from this new CV value. In case that the EN input of T is 0 or the CLR input of C is 1, the CV value of T or C will be reset to 0.

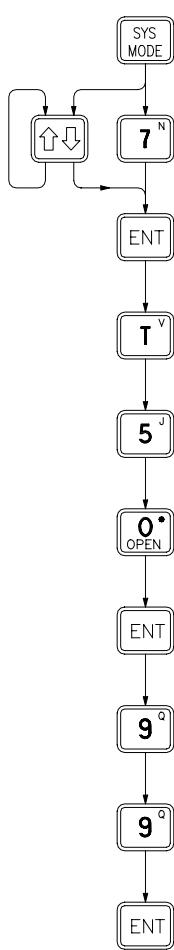
#### 〈Example 1〉 Timer Setting

Since TCAP will only display the T/C values used in the program, the prerequisite to use TCAP to set the value of a timer or a counter is that the T/C to be set must exist in the program. Assuming that in the program there is a T50 circuit, as shown in the diagram below, where the PV operand is R23 and the content of R23 is 0.

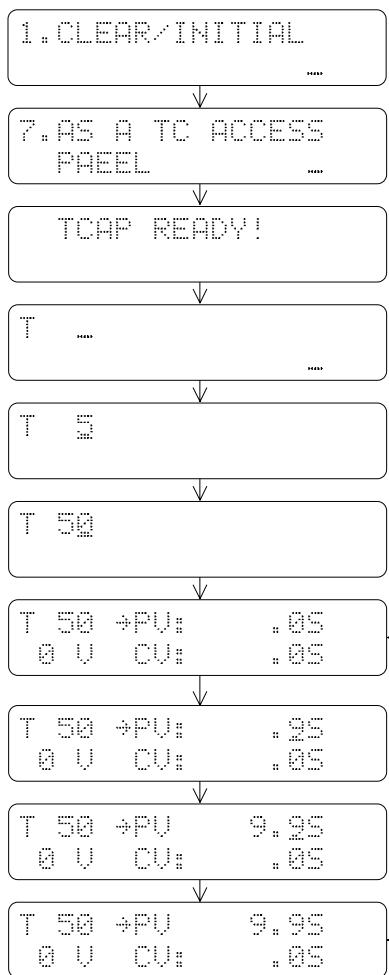


The following key operation uses the TCAP mode to change the timer of T50 from 0 second (as R23=0) to 9.9 seconds while X0 is OFF and the PLC is in RUN mode.

**【Key Sequence】**



**【LCD Display】**



This step can be skipped if the PLC is already set to TCAP mode.

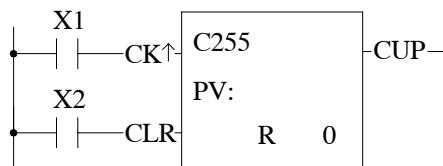
Although PV=CV, T50 contact is still equal to 0 because the input EN value is 0

PV value is underlined which indicates that the value is being edited and has not been entered in T50 yet.

The underline of PV value disappears which means that the value has been entered into T50.

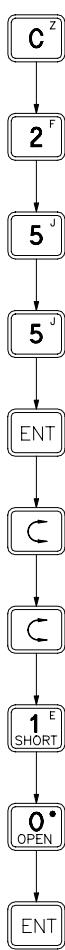
After finishing the above settings, now you can turn ON the switch X0 and monitor the contact T50 which will be turned ON 9.9 seconds later.

⟨ Example 2 ⟩ 32-Bit Counter settings

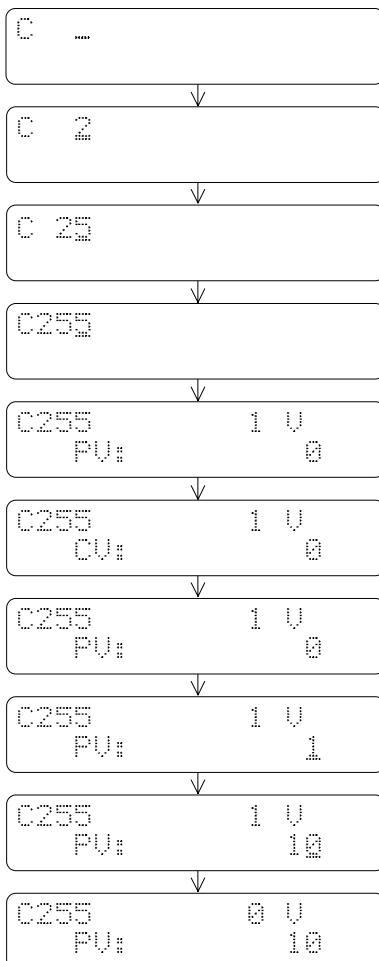


Continues from example 1, a procedure of setting a 32-bit counter is demonstrated. Before operation, assume CLR input of C255 is 0. The following key sequence will observe the PV and CV values of C255 then change the PV value to 10.

**【Key Sequence】**



**【LCD Display】**



When PLC enters RUN, contact C255 is ON because PV=CV=0

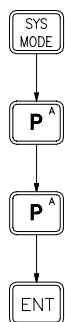
Contact C255 turns OFF because PV is set to 10 that resulted CV<PV

Please refer to section 2.6 for monitoring of R or D register

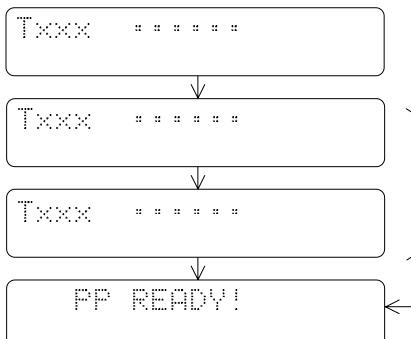
**2.4.7.3 Changes from TCAP Mode to PP Mode**

While working in the TCAP mode, you can change the PP/TCAP flag to PP at any time by following the key operations shown below.

**【Key Sequence】**



**【LCD Display】**

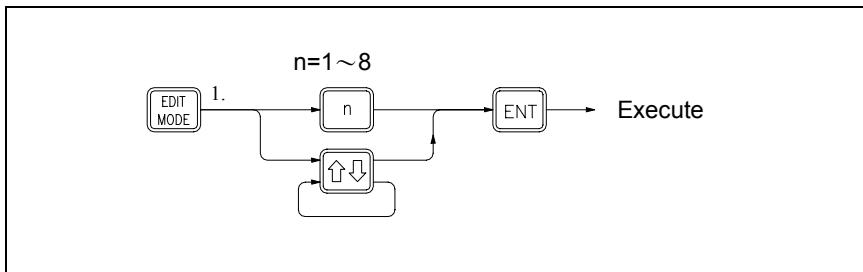


The LCD displays will not change.

Indicates the system is ready for PP operation.

## 2.5 Operation of EDIT MODE

Fundamental key operations of edit mode:

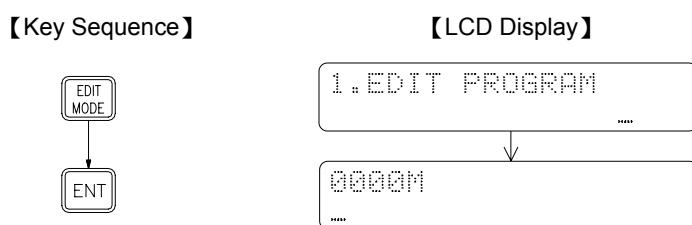


- System mode has a total 7 functions:
  1. EDIT PROGRAM
  2. EDIT REGISTER DATA
  3. SYNTAX CHECK
  4. MOVE HR→ROR
  5. CHECK DOUBLE COIL/T/C
  6. EDIT HPSO INSTRUCTION
  7. EDIT LINK INSTRUCTION
  8. EDIT DOCUMENT
- When the first time enter the Edit Mode, the LCD screen will be automatically prompted with main function 1 which is the “EDIT PROGRAM”. If it is not the desired function can either directly input the function number (n) or use to search for the desired function and then press to execute the function.

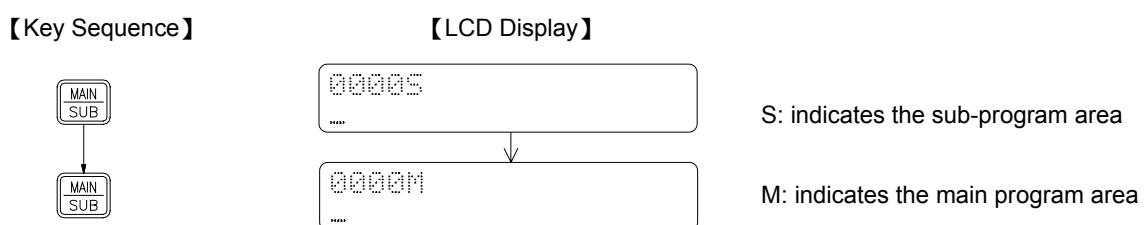
### 2.5.1 EDIT PROGRAM

Before start working on the following example, please perform the “CLEAR” operation shown in section 2.2.1 first.

- Enter the edit mode



(If want to enter the sub-program area, press . Press again will return to the main-program area.)



- Instruction keys, parameter keys and **[ENT]** key are the basic keys used in the program Edit Mode. Besides that other keys are also provided for searching, writing and correcting the programs.

**[MAIN SUB]**: This key is used to select the main-program or sub-program area because the main-program and sub-program are stored in different areas.

**[C]**: This key is used to select the first or second row displayed on LCD. Using this key can move the cursor to the row to be edited. In the program Edit Mode, this key is used to select the editing of the instructions (second row on display) or their documents (first row on display).

**[CLR]**: In the Edit Mode, edited information will not be saved in PLC until **[ENT]** is pressed. Before pressing **[ENT]**, data are stored in a temporary editing area for subsequent checking and correcting. **[CLR]** can be used to clear this temporary editing area. Once pressing this key, LCD screen will be cleared. With this key, the wrong data or instruction in edit process can be cleared if **[ENT]** has not been pressed.

**[DEL]**: After pressing **[ENT]**, instructions or data will be stored in the PLC program area. In this case, **[CLR]** can not be used to clear them. Instead **[DEL MODE]** is required to delete them from the program area of the PLC.

**[INS]**: This key is used to insert instructions in a program.

**[↑]** or **[↓]**: These keys are used for program address increment or decrement.

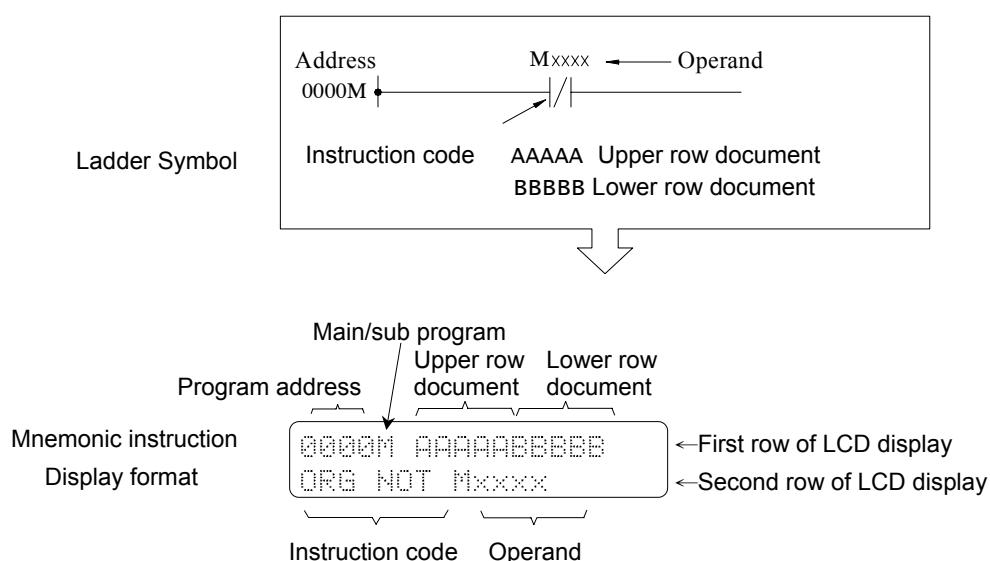
**[SHIFT] [↑]**: Move the cursor to the top ( address:0000M or 0000S ) of the main-program or sub-program.

**[SHIFT] [↓]**: Move the cursor to the bottom ("BOTTOM" will be displayed on LCD) of the main-program or sub-program.

**[SHIFT] [DEC]**: To choose the display or input in decimal format. (When enter EDIT mode for the first time, the display will be in decimal).

**[SHIFT] [HEX]**: To choose the display or input in hexadecimal format.

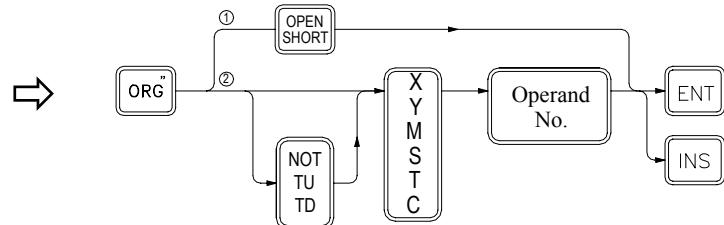
- The display format of mnemonic instruction



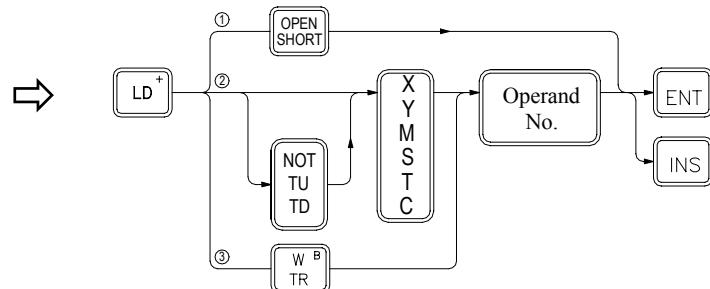
## 2.5.1.1 Sequential Instruction Editing

### 2.5.1.1.1 Fundamental Key Operations of Sequential Instruction

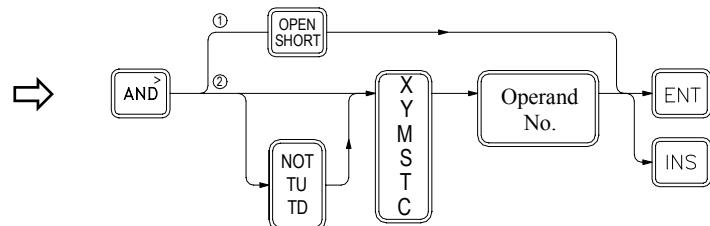
#### ● ORG instruction



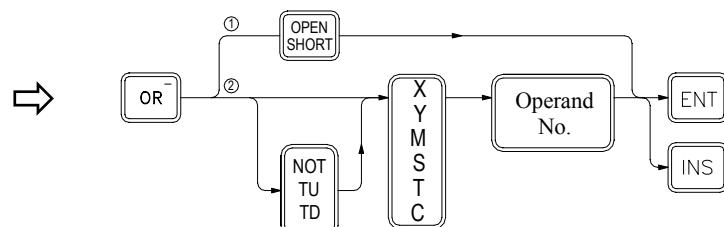
#### ● LD instruction



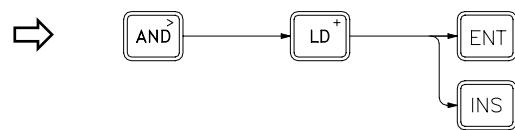
#### ● AND instruction



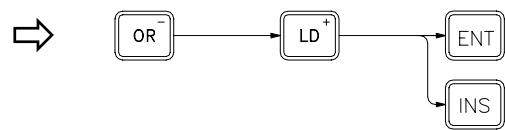
#### ● OR instruction



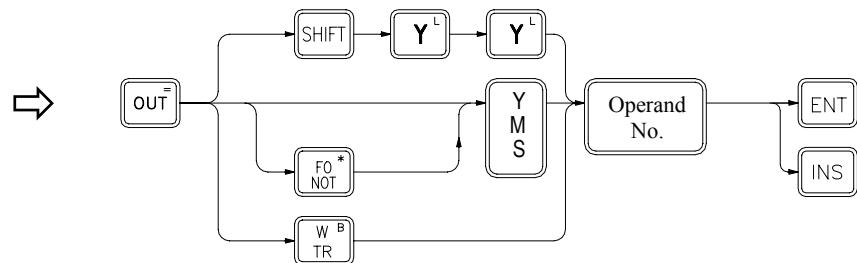
● ANDLD instruction



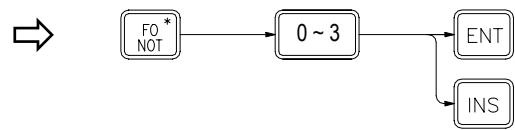
● ORLD instruction



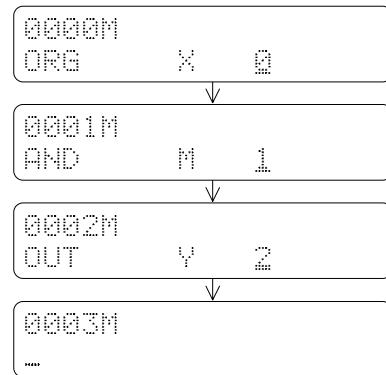
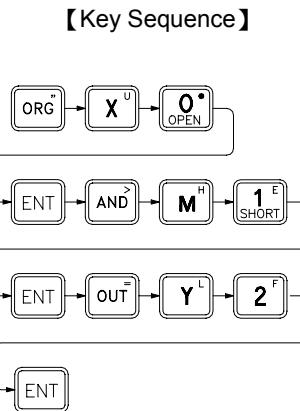
● OUT instruction



● FO instruction



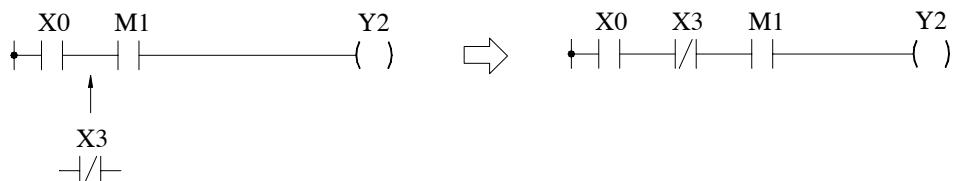
### 2.5.1.1.2 Input the Instruction



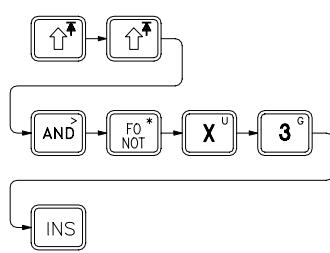
On pressing **ENT**, If the input is correct, the instruction will be written into the memory and the address shown on the LCD display will change to the next location.

### 2.5.1.1.3 Insert the Instruction

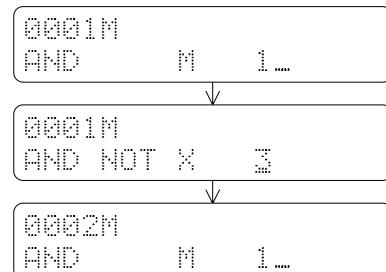
If want to insert a new instruction before an old instruction, first find the old instruction then type the new instruction at the address of the old instruction and press **INS**. The address of old instruction will move to the new location right after the new instruction which means all the step number of the old instructions after the one inserted will be increased by one. Continuing the example shown above, before inserting a B contact of X3 between the A contact X0 and A contact M1, use to find the step before which a new instruction is to be inserted (in this case AND M 1 at step 0001M) and key in the instruction to be inserted then press **INS** to complete the insert operation.



**【Key Sequence】**

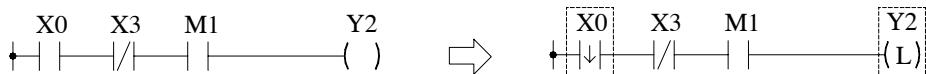


**【LCD Display】**

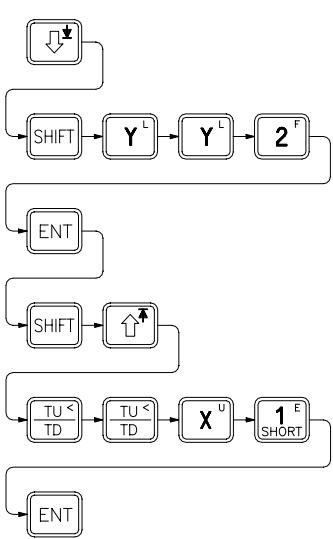


#### 2.5.1.1.4 Change Instruction

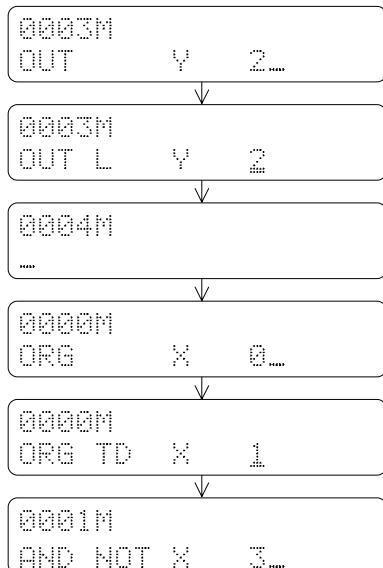
First find the old instruction to be changed (if it is a function instruction, should step to the address which show FUNXX of the instruction) then key in the new instruction and press ENT key to overwrite the old instruction. For example, if you want to change Y2 to a retentive output coil and A contact X0 to TD down differential contact X1, find the old instructions to be changed using , then key in the new instruction or modify the instruction and press to complete the change as shown below.



【Key Sequence】



【LCD Display】



- Find out the instruction to be changed (OUT Y2)

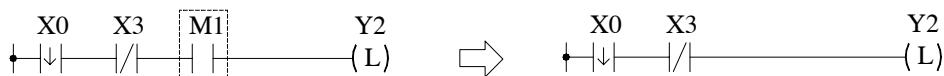
- Change OUT to OUT L

- Repeatedly pressing 4 times will do the same thing

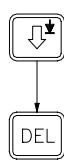
- Change A contact to TD contact and change X0 to X1

#### 2.5.1.1.5 Delete Instruction

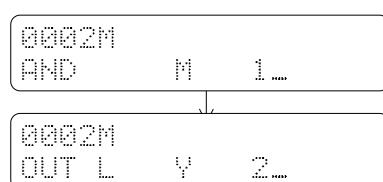
Find the instruction to be deleted. On pressing the key, the instruction under display is deleted. The example shown in below demonstrates how to delete the A contact M1.



【Key Sequence】

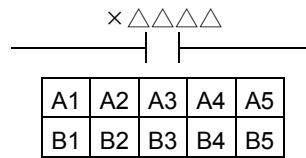


【LCD Display】

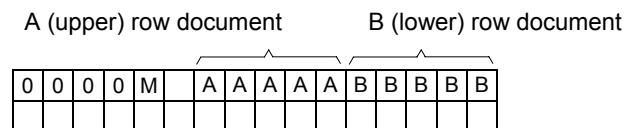


#### 2.5.1.1.6 Edit the Element Documents

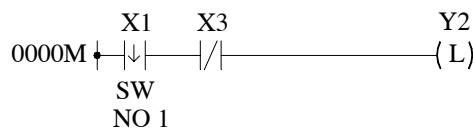
The element documents are the same for the instructions with identical operand number. For example, the element comments of AND X0 and OR NOT X0 are actually the same as shown in the diagram below. The documents consist of two rows of strings (5 characters for each row and a total of 10 characters) and place directly underneath the element.



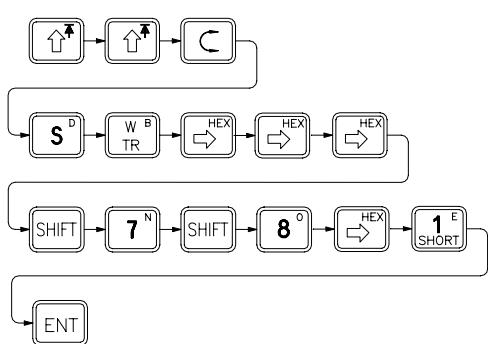
A and B rows shown in the diagram above occupy 10 characters space of the first row on the right side corner of FP-07 LCD display.



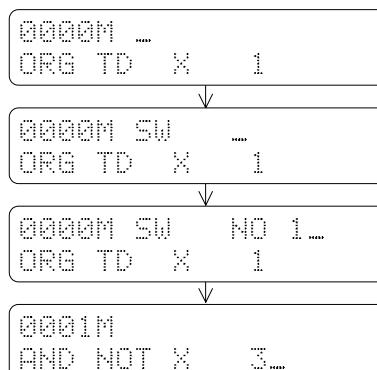
Example : Following the key operations shown in below to add the documents to TD contact X1.



【Key Sequence】



【LCD Display】



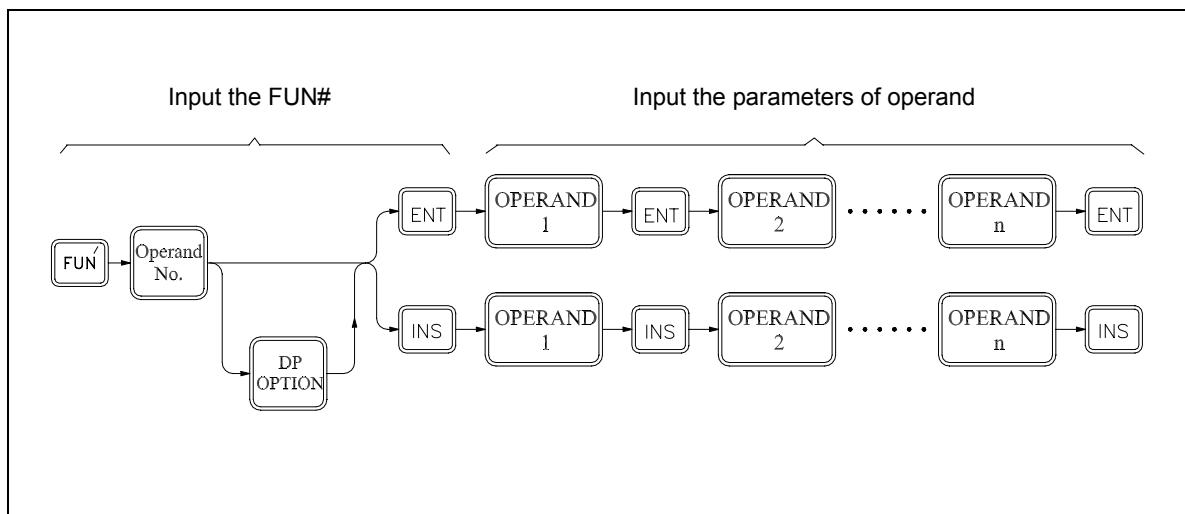
- key moves the cursor to the upper document area
- Cursor moves three characters horizontally on pressing the three times

### 2.5.1.2 Edit Function Instruction

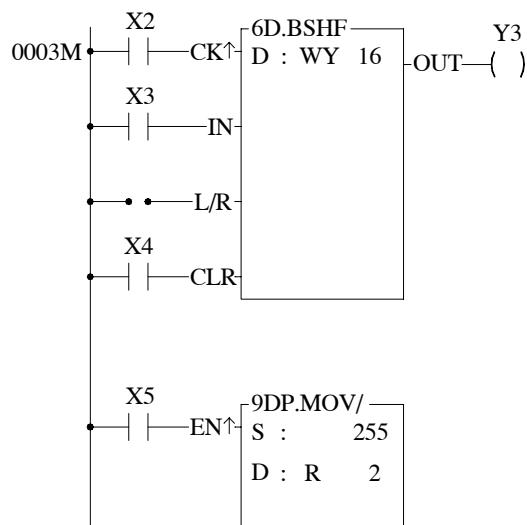
In this section, we only concentrate on the key operations of editing the function instructions. For explanations of function instructions, please refer to Chapter 6, "Introduction to function Instructions".

Each function instruction consists of an instruction name (Mnemonic) and a reference number except nine special instructions keys such as T, C, SET etc.. Besides those nine special instructions, other function instructions must be entered with their function number (FUNXXX). It is possible to add a postfix character D and P after the FUNXX on certain function instructions to produce sub-instructions. The key operations of function instructions are shown below.

Fundamental key operations of the function instructions

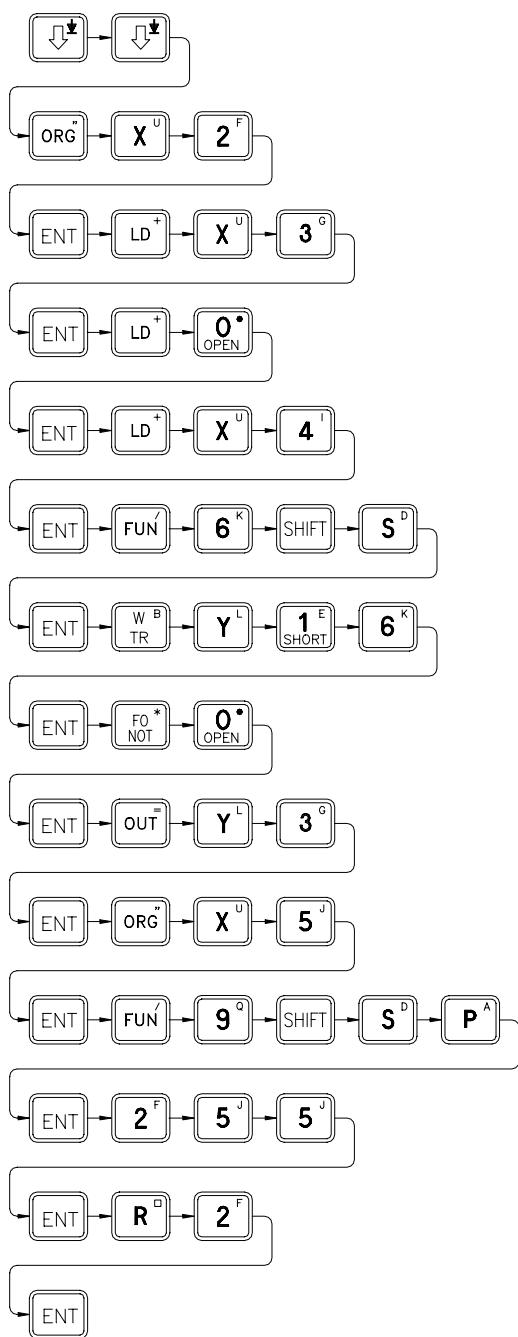


Continuing from the previous LCD display, input the function instructions listed below:

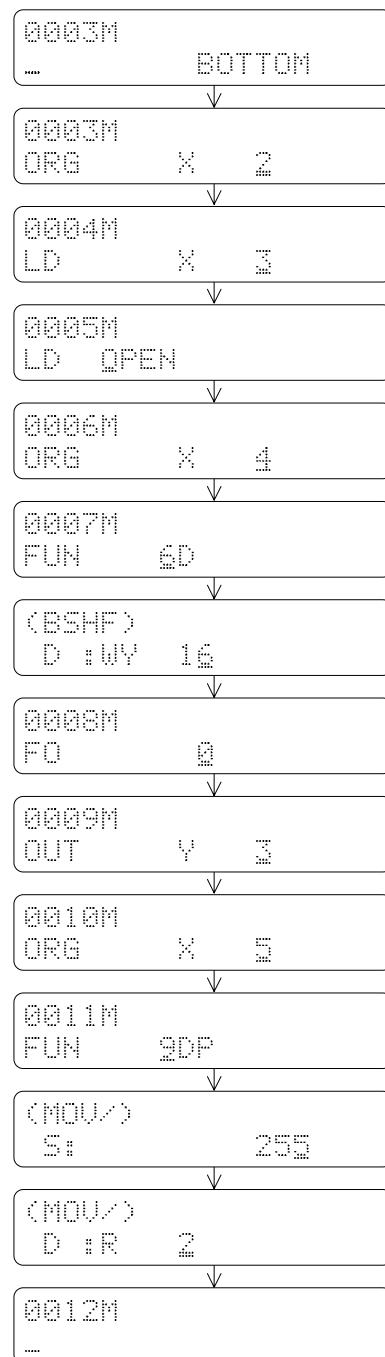


ORG	X	2
LD	X	3
LD OPEN		
LD	X	4
FUN	6D	
D : WY	16	
FO	0	
OUT	Y	3
ORG	X	5
FUN	9DP	
S :	255	
D : R	2	

### 【Key Sequence】



### 【LCD Display】



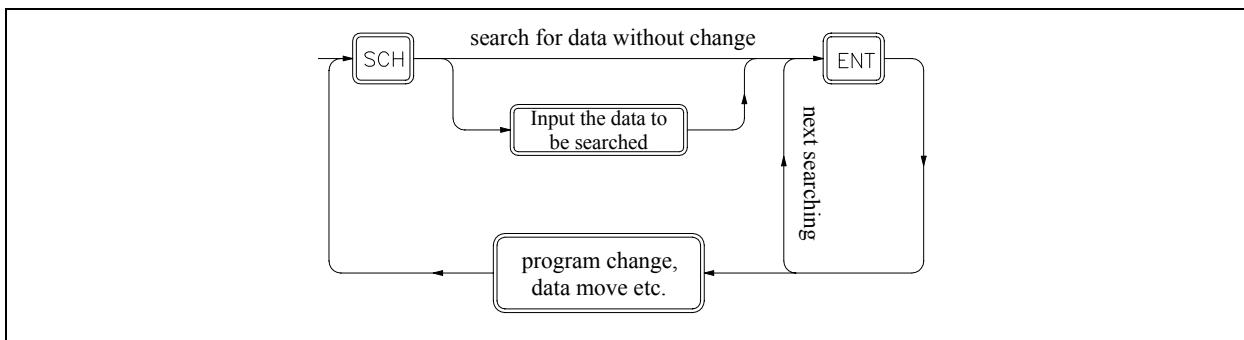
**Description:** In case an error occurs during editing, press **CLR** to clear the incorrect instruction or parameters, then key in the correct one and press **ENT**.

### 2.5.1.3 Search Program

In the process of editing, monitoring and searching the PLC program, it is very time consuming to search for the address of a specific instruction using if the size of the program is very large (ex. FB<sub>E</sub>-PLC has 13K steps). FB-PLC provides a program search operation using that gives you a convenient way to search through a long program for a specific instruction, address, operand, comment or parameter.

Type	Items can be searched	Examples
Address search	Main-program ( $\triangle\triangle\triangle\triangle M$ )	0001M, 0047M, .....
	Sub-program ( $\triangle\triangle\triangle\triangle S$ )	0007S, 1234S, .....
Instruction search	Instructions (either sequential instructions or application instructions)	ORG X0, OUT L Y2, FUN 20P, .....
	Parameters of function instructions	R100, WX0, T50, .....
	Element + Operand	TU X10, NOT M200, .....
	Operand	X0, M1000, .....
Document search	Instruction document	SW1, AUTO STOP, .....

Syntax chart of key operation of the program search



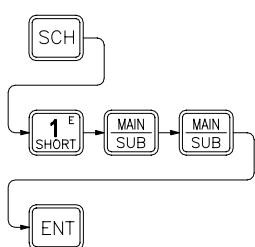
To search for a specific data in program memory, first require to input the data to be searched. The data have just entered are stored in a search buffer. After pressing , FP-07 will begin to search in the program memory of the PLC for the specific data stored in the search buffer. The data stored in search buffer are retained even after completion of the search operation. This means the consecutive data search and data change are possible. The data stored in search buffer will be cleared in case of power failure or mode (Edit, Monitor, System, RUN/STOP etc.) change.

#### 2.5.1.3.1 Search Address

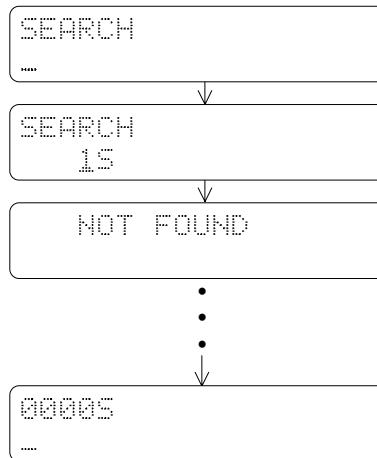
To search for a specific address in program memory, input the address you wish to search first. The system will search for that particular address in either main-program or sub-program area according to the input specification such as M (main-program area) or S (sub-program area). Which means you can search for a particular address in the sub-program area while working in the main-program area or vice versa. If found, the address will be displayed on LCD otherwise the search will stop at the last address of the program area and "NOT FOUND" will be displayed.

Continuing from the preceding LCD display, if want to search for a particular address (0001S) in the sub-program area while working in the main-program area (0011M), can perform the key operations shown below.

#### 【Key Sequence】



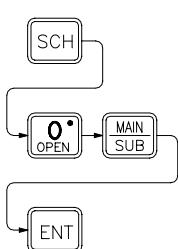
#### 【LCD Display】



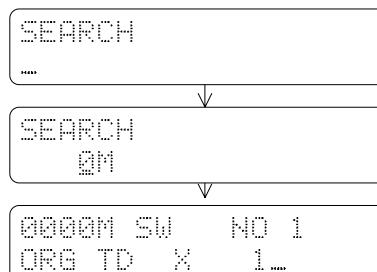
- Press **MAIN SUB** once, “M” appears than press this key again, “S” appears

The address is not found because the programs are not exist in the sub-program area. LCD display shows the search is stopped at the last address (000S) of the sub-program area. If want to return to the main-program area, perform the following key operations.

#### 【Key Sequence】



#### 【LCD Display】



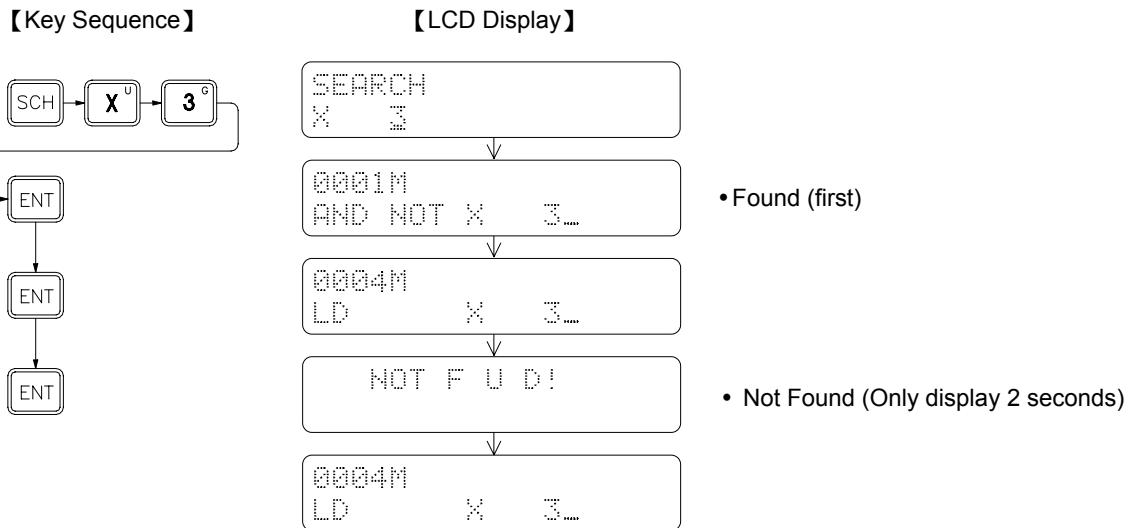
#### 2.5.1.3.2 Search Instruction

Instruction search is used to search for the specific instruction in the main/sub-program area. There are two ways can be used to do the instruction search.

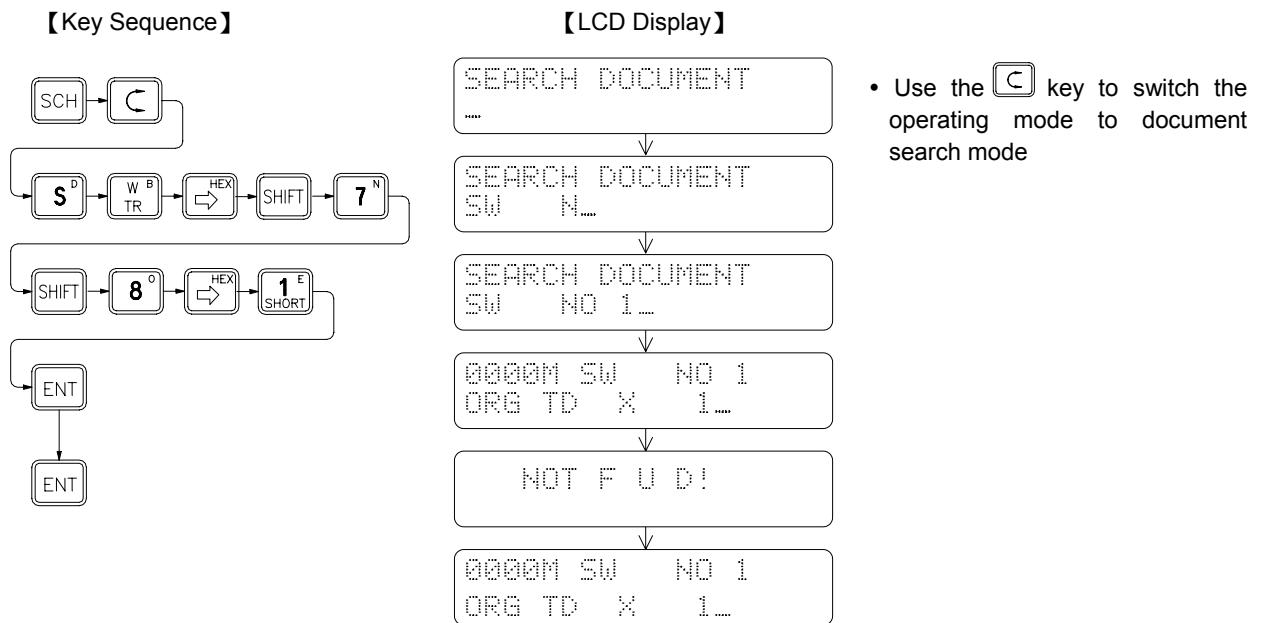
1. Input(or change)the instruction to be searched: FP-07 searches the data from the top of the program (0000M)to the last address of the program including the sub-program area.
2. Using the data retained in the Search Buffer to perform the search operation: FP-07 searches the data starting from the address next to the one displayed on LCD. Sub-program area is also been searched.

Input the instruction to be searched first, then press **ENT**. FP-07 will perform the search operation starting from the top of the program. If found, the instruction will be displayed on LCD. To continuously search for the same instruction, press **ENT** to resume the instruction search starting from the address next to the one of displayed. If **ENT** is pressed continuously, all the addresses having the specified instruction are successively displayed until the address with “NOT FOUND” is displayed. After the message “NOT FOUND” is displayed for 2 seconds, the address last found in the program is displayed. In addition, it is also possible to use the search instruction to search the instruction with partial specified (please refer to the table shown above to know the available specifies for instruction search).

● Using the operand to search for “X3”



● Using the document to search for “SW NO1”

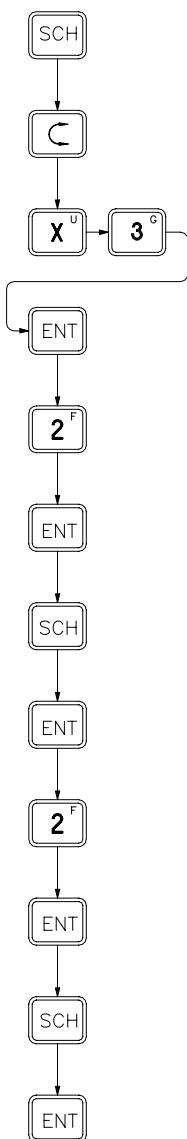


**Remark:** The difference of the display between instruction/address search and document search is that a message “DOCUMENT” is displayed on the right corner of the LCD screen when the document search is performed. **C** key can be used to select one of the two search methods.

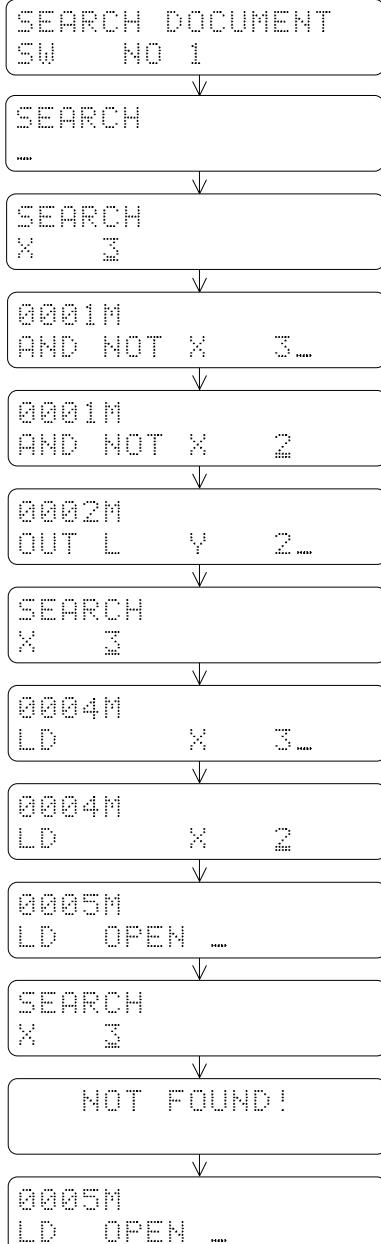
● Successive search and change

Following the preceding LCD display, search for all the instructions containing an operand X3 and then change the operand X3 to X2.

【Key Sequence】



【LCD Display】



- Search buffer contains the old data from last search operation (Document search)
- **C** key switches the mode from document search to address/instruction search
- Key in the new data to be search (Operand X3)
- Found the first instruction containing an operand X3
- Change operand X3 of the first found instruction to X2
- Found the second instruction containing an operand X3
- Change operand X3 of the found second instruction X2

**Remark 1:** In the process of successive search, if other keys other than the **ENT** key is pressed, such as pressing the parameter keys to change the operand data, you must press **SCH** key again to resume the search process.

**Remark 2:** As we have mentioned in above, a **SCH** key must be pressed to resume the search process after a data changing process. Users should keep in mind that the data stored in search buffer are retained during the whole process. The resumed search will start from the current address displayed on LCD. But if you input a new data to be searched at this time, search will automatically start from the top that means 0000M or 0000S.

## 2.5.2 EDIT REGISTER DATA

This function is provided mainly for editing (input) the data of the registers which number are consecutive. You must use this function to edit the ROR data. It is possible to use the register editing function of monitor mode to change the register's value, but you need to repeatedly press key for successive editing (please refer to the section 2.2.5). In comparison, the method we provided in this section is more convenient to do so.

- Key operations for entering the register data Edit Mode:



After entering the register data edit mode, you can directly select the number of the registers ( $R\triangle\triangle\triangle\triangle$ ,  $DR\triangle\triangle\triangle\triangle$  or the registers consist of 16 or 32 coils, such as  $WY\triangle\triangle\triangle$  and  $DWM\triangle\triangle\triangle\triangle$  etc.) which you wish to edit. The table shown in below listed the names of the registers which can be edited and the ranges of the corresponding register numbers.

Register type		Number range	Remark
16 bits	$R\triangle\triangle\triangle\triangle$	Data Registers	$R0 \sim R3839$
		Output Registers	$R3904 \sim R3967$
		HSC Registers	$R4096 \sim R4127$
		Calendar Registers	$R4128 \sim R4135$
		Special Registers*	$R4136 \sim R4167$ 及 $R3967 \sim R4095$
		Read-Only Registers	$R5000 \sim R8071$
32 bits	$D\triangle\triangle\triangle\triangle$	D Registers	$D0 \sim D3071$
	$WY\triangle\triangle\triangle$	Output Coils	$WY0, WY8, \dots, WY144$
		Internal Coils	$WM0, WM8, \dots, WM1384$
		Step Coils	$WS0, WS8, \dots, WS984$
	$DR\triangle\triangle\triangle\triangle$	Data Registers	$DR0 \sim DR3838$
		Output Registers	$DR3904 \sim DR3966$
		HSC Registers	$DR4096 \sim DR4126$
		Calendar Registers	$DR4128 \sim DR4134$
		Special Registers*	$DR4136 \sim DR4166$ and $DR3968 \sim DR4094$
		Read-Only Registers	$DR5000 \sim DR8070$
	$DD\triangle\triangle\triangle\triangle$	D Register	$DD0 \sim DD3070$
	$DWY\triangle\triangle\triangle$	Output Coils	$DWY0, DWY8, \dots, DWY128$
		Internal Coils	$DWM0, DWM8, \dots, DWM1368$
		Step Coils	$DWS0, DWS8, \dots, DWS968$

\*: except the special register marked " \* " (please refer to page 3-4)  
The rest of  $R5000 \sim R8071$  which are not configured as ROR could used as normal registers (R/W)

$\triangle\triangle\triangle\triangle$  or  $\triangle\triangle\triangle$  must be the multiples of 8

\*: except the special register marked "

$\triangle\triangle\triangle\triangle$  or  $\triangle\triangle\triangle$  must be the multiples of 8

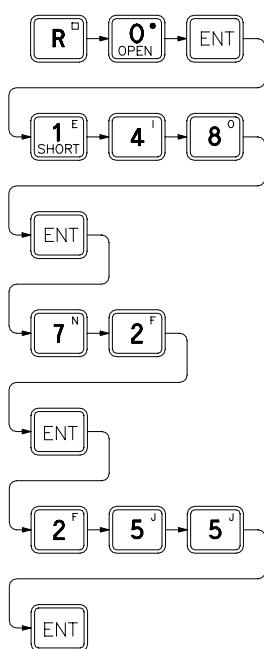
**Example 1** 16-Bit register editing (assuming already in the register data Edit Mode)

R0=148

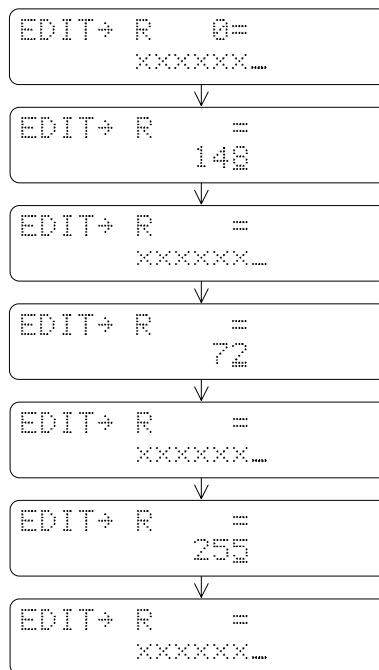
R1=72

R2=255 (or FFH)

【Key Sequence】



【LCD Display】



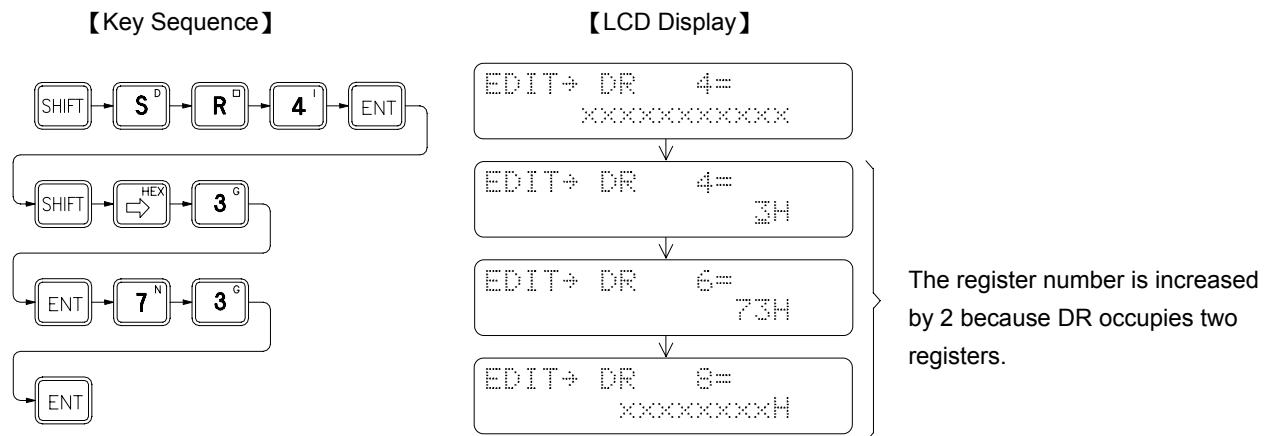
- $\times \times \times \times \times \times$  is the current value (before edit) of R0

- After input a new value, the displayed register number will be increased successively.

- After you entering the EDIT MODE of FP-07, the current value for registers starting with an R (R△△△△△ or DR△△△△△) are displayed in decimal format, while for the registers starting with an W (W□△△△△△ or DW□△△△△), are displayed in hexadecimal format. Pressing **SHIFT** **DEC** keys or **SHIFT** **HEX** keys can change the format as you desired.
- If you want to change the current value of a register, key in the new value directly and then press **ENT** key to complete the change. If not simply press **↓** key to display the next register.

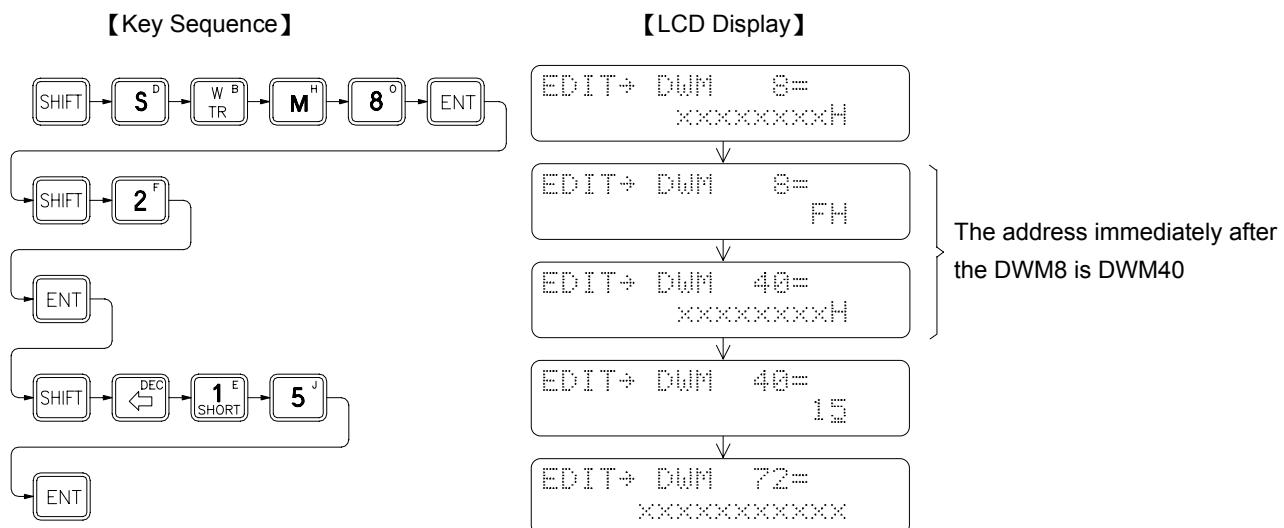
**Example 2** 32-Bit register edit

Continues from the Example 1, input 3H for registers R4~R5 and 73H for registers R6~R7.



**Example 3** The editing of 32-Bit register composed by coils

Continues from the Example 2, input FH for register DWM8 ( M8~M39 ) and 15 for register DWM40 ( The value of FH is equal to 15 while their input format are different ).

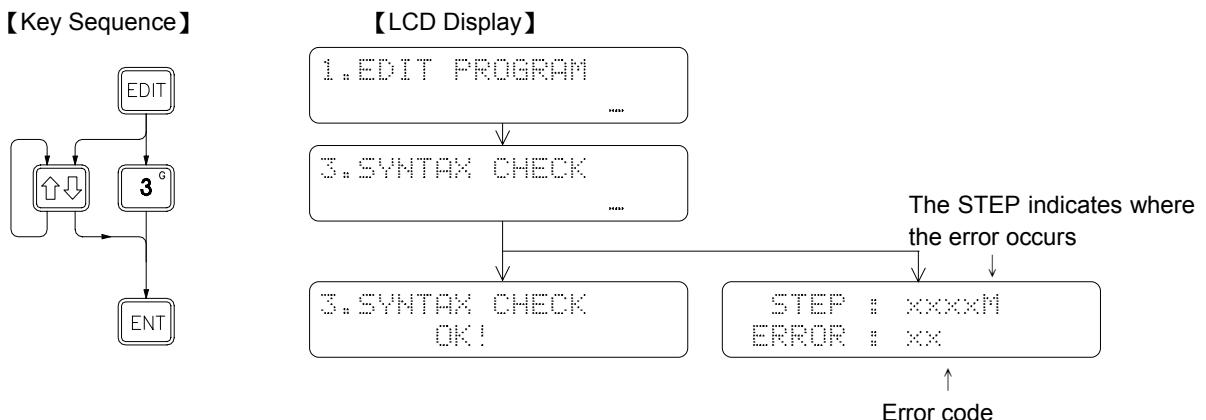


### **2.5.3 SYNTAX CHECK**

Every computer language has its own syntax rules to follow, otherwise the computer will not execute the instructions if there is a syntax error existed in the program. Ladder Diagram program used by the PLC has its syntax rules as well. Besides the syntax rules on designing of the Ladder Diagram program shown in Chapter 1, other syntax rules of FB-PLC are listed as follows.

1. The maximum permissible element size in a Ladder Diagram network is 16 rows × 11 columns. But this size can be expanded to 16 rows × 22 columns according to the specific needs (please refer to Figure 1 in page 1-5).
2. Contacts such as A, B, TU, TD, SHORT and OPEN can be located on any columns except the last column (column 11 or column 22).
3. Coils can only be located on the last column of the network (column 11 or column 22)
4. The width of all application instructions in the Ladder Diagram occupies 3 columns and the length is variable between 1~4 rows. Except the instructions listed in rule No.5 shown in below, the rest of the application instructions must be located on column 2, 3 and 4 counting from the end (column 8, 9, 10 or column 19, 20, 21)
5. Each of the six instructions, such as FUN 1(MCE), FUN 3(SKPE), FUN 65(LBL), FUN 68(rts), FUN 69(RTI), FUN 70(FOR), and FUN 71(NEXT), forms its own network. No other elements can be serially connected in front of these instructions; that is, these instructions are connected directly to the original line and occupy the first three columns (column 1, 2, 3). These instructions do not have the output function either and the instruction FUN68 and FUN69 can only be used in the sub-program area.
6. For each multiple-input instructions, every input point needs a serially connected element; that is, every FUN instructions with n inputs must have n numbers of network rows connecting to each input point.
7. Instructions FO# can only be used with the application instructions containing the function output (FO). Following every FO# instruction, there must be a corresponding OUT instruction.
8. Six instructions such as FUN 0(MC), FUN 2(SKP), FUN 66(JMP), FUN 67(CALL), FUN 70(FOR) and FUN 71(NEXT) which are used for controlling the program flow can only be located on the first column of the network and can not be connected to OUT or any other application instructions in parallel starting from row 2.
9. The ladder diagram is illegal if it has following condition.
  - (1) Cross a line
  - (2) FUNs with input overlap
10. The instructing combinations, which can not form the diagram, are not permissible.
  - (1) The contact element occupies the location of the coils and application instructions.
  - (2) If there are several OUT TR# instructions in a single network, when you want to get the TR statuses (LD TR#) from the memory, you should first get the TR# contact which stored at the last.
11. The name of LBL must be unique in the program.
12. The # of MC or SKP instruction can not be duplicated.
13. The # of TR can not be duplicated in the same network.
14. The number of the instructions in a network can not exceed 64 Words.
15. The number of instructions resulted from subtracting the total number of ORLD and ANDLD instructions from the total number of LD instructions can not exceed 8 for any combination of the instructions in a network. If the network including the OUT or application instructions, then the subtraction between number of the LD instructions and number of the ORLD+ANDLD instructions need to be recalculated.

### 2.5.3.1 Key Operations of Syntax Check



### 2.5.3.2 Syntax Error List

ERROR 1 : Instruction ORG is missing

ERROR 2 : AND, OR, LD, ORLD, ANDLD, OUT TR, FUN, C and T instructions can not be connected directly after the FUN and C instructions.

ERROR 3 : OR, ORLD, ANDLD and OUT TR instructions can not be connected directly after the OUT and T instruction.

ERROR 4 : OR, ORLD, ANDLD and OUT TR instructions can not be connected directly after the OUT TR instruction.

ERROR 5 : OR, ORLD, ANDLD and OUT TR instructions can not be connected directly after the LD TR instruction.

ERROR 6 : FUN instruction does not exist before the FO# instruction.

ERROR 7 : The # of FO# exceeds the limit.

ERROR 8 : The # of FO# can not be duplicated.

ERROR 9 : FO instruction can only connect one OUT instruction.

ERROR 10: FO instruction does not exist before the OUT instruction.

ERROR 11: In a network, OUT, FUN, T and C instructions can not be used after the MC, SKP, JMP, CALL, FOR and NEXT instructions.

ERROR 12: In a network, the # of OUT TR# has been repeatedly used.

ERROR 13: In a network, LD TR# is used without the OUT TR#.

ERROR 14: In a network, the number of the instructions resulted from LD+LD TR – ORLD – ANDLD is greater than 8 (If FUN or C instructions are appeared, the number need to be recalculated).

ERROR 15: After the LD instruction, its pairing instructions of ORLD and ANDLD can not be found.

ERROR 16: ANDLD instruction can not be used in pairs following a LD TR instruction.

ERROR 17: Before the ORLD instruction, its pairing instructions of LD and LD TR can not be found.

ERROR 18: Before the ANDLD instruction, its pairing instruction of LD can not be found.

ERROR 19: OUT, OUT TR and LD TR instructions are used prior the completion of the block editing.

ERROR 20: There are not enough LD or LD TR instructions available to match the input numbers of the FUN and C instructions.

ERROR 21: In forming the Ladder Diagram, the network exceeds 16 rows.

ERROR 22: In forming the Ladder Diagram, the contacts occupy the coil location.

ERROR 23: In forming the Ladder Diagram, the contacts occupy the location of application instruction.

ERROR 24: In forming the Ladder Diagram, MC, SKP, JMP, CALL, FOR and NEXT are not located on the first row.

ERROR 25: In forming the Ladder Diagram, the input contact location of FUN and C exceeds the limit of allowable input contact paths.

ERROR 26: In forming the Ladder Diagram, either two of the FUN and C input contact paths are touching or stacking to each other.

ERROR 27: In forming the Ladder Diagram, there is a vertical short circuit line contacting the edges of the application instruction.

ERROR 28: In forming the Ladder Diagram, you must use the ORLD and OR instructions to connect the contacts in the preset OUT TR# diagram.

ERROR 29: In forming the Ladder Diagram, ORLD instruction can not form a reasonable diagram.

ERROR 30: In forming the Ladder Diagram, LD TR instruction can only be used to form the diagram in forward direction.

ERROR 31: In forming the Ladder Diagram, LD TR instruction can cause the line-cross problem.

ERROR 32: In a program, the # of MC# and MCE# are duplicated.

ERROR 33: In a program, the # of SKP# and SKPE# are duplicated.

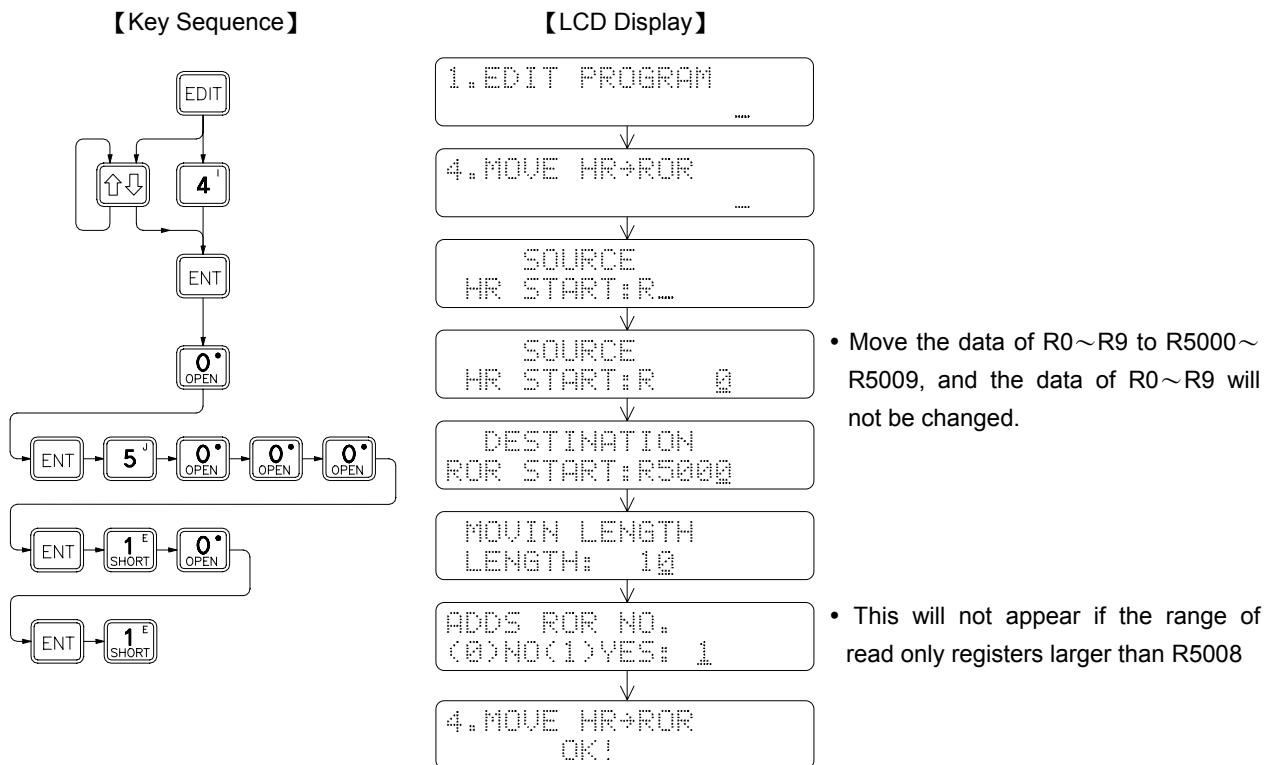
ERROR 34: In a program, the # of T# is duplicated.

ERROR 35: In a program, the # of C# is duplicated.

ERROR 36: The number of the instructions in a network exceeds 64 words.

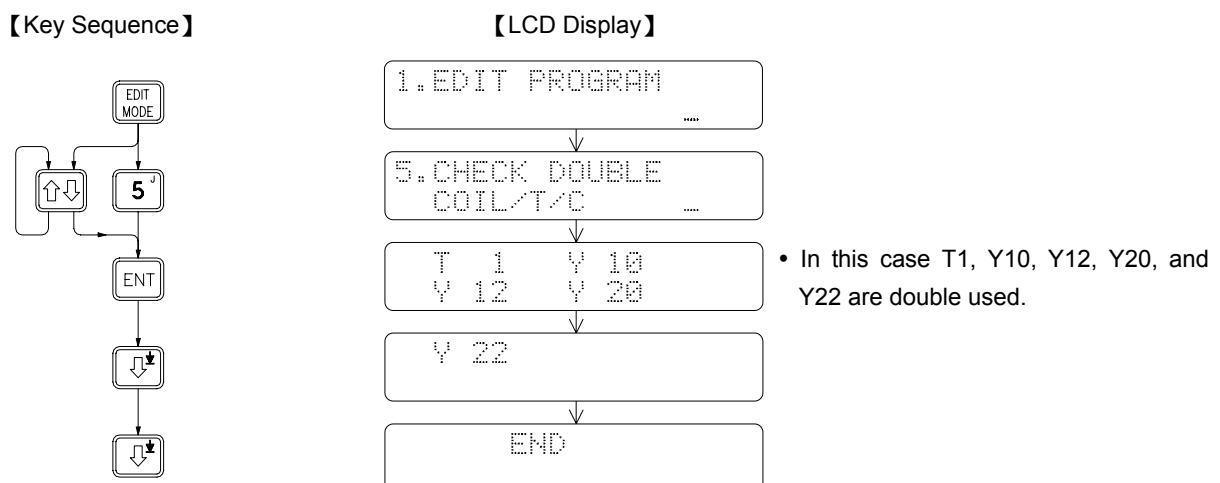
## 2.5.4 Move (HR→ROR)

This function is provided mainly for moving the contents of the data registers (HR) to read-only registers (ROR) area so that it can be burned into EPROM or EEPROM.



## 2.5.5 CHECK DOUBLE COIL/T/C

This function is provided mainly for checking Coil, Timer, Counter, if they are used more than once in the program. After executing, if it found, will display the double coil number.



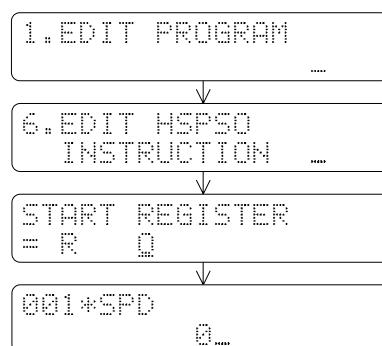
## 2.5.6 EDIT HPSO INSTRUCTION

There are some unique control instructions especially provided by FBe/FBN PLC for NC positioning. The users need only to choose the starting register (R0~R3828, R5000~R8060, D0~D3060) where these instructions are stored, then FP-07 will judge automatically on whether the number selected is a new or an old HPSO instruction area. The basic unit of HPSO instructions is a command. A complete command consists of 3-4 instructions. Press **[INS]** or **[DEL]** key to increase or decrease the number of command. Press one of **[SYS MODE]**, **[EDIT MODE]**, **[MON MODE]** or **[RUN STOP]** key to leave and save the editing, then follow the guide displayed on FP-07.

- Start HPSO instruction editing

【Key Sequence】

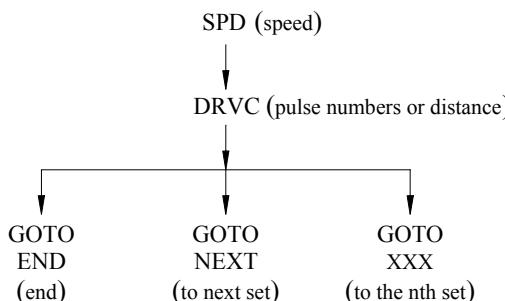
【LCD Display】



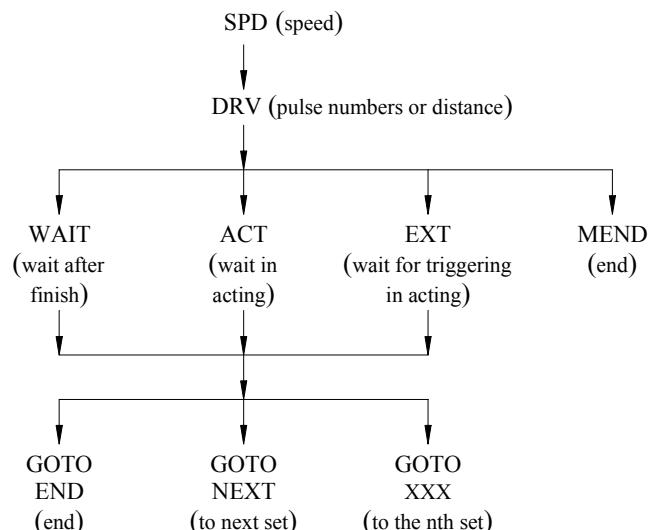
\* Press **[SHIFT]** and **[EDIT MODE]** keys may also start HPSO instruction editing when “FUN 140” is displayed in the editing ladder instruction area. Press **[SHIFT]** and **[EDIT MODE]** keys again to return to “FUN 140” for instruction editing.

- Formation of a NC command

1. Continuous multi-zone speed



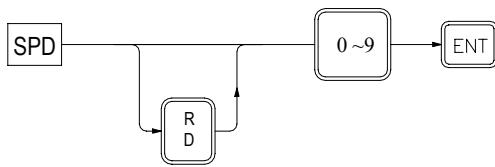
2. Last or single zone speed



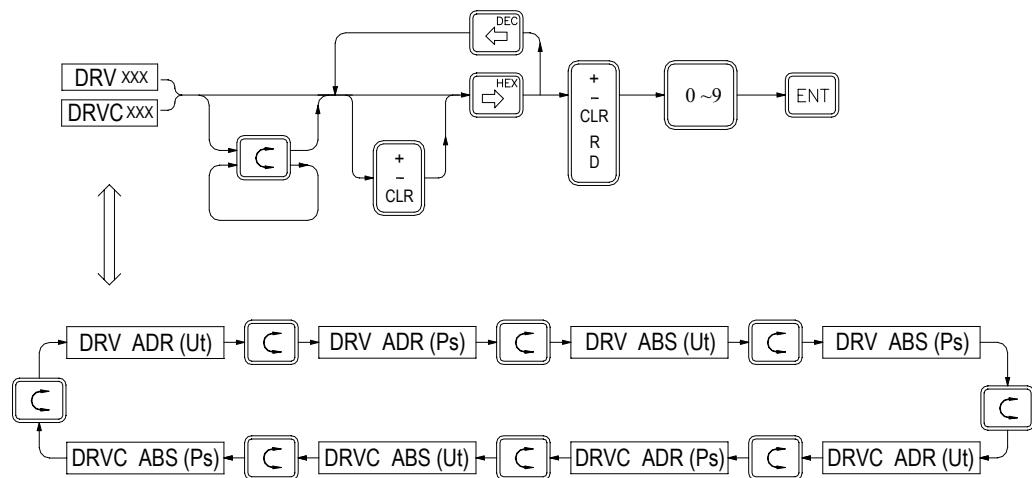
**Remark:** DRVC is used for performing continuous multi-zone speed changing control (up to 8 commands) and the last command must apply DRV instruction.

### 2.5.6.1 Fundamental Key Process of HPSO Instruction

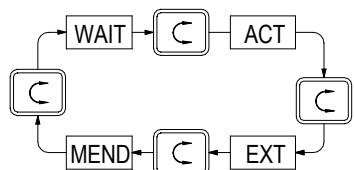
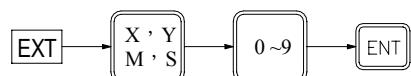
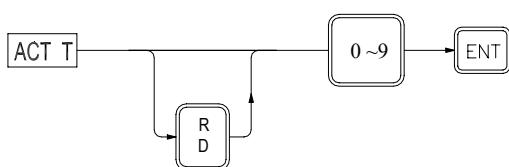
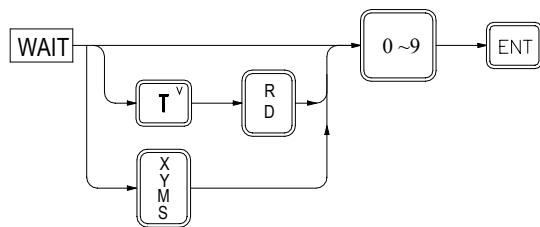
- SPD instruction



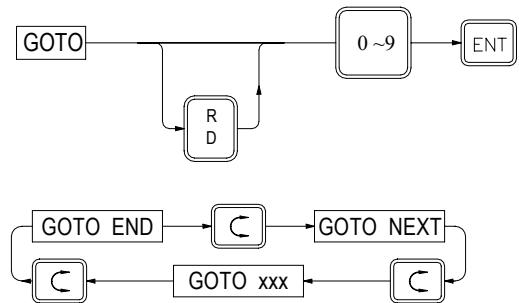
- DRVC、DRV instruction



- WAIT、ACT、EXT、MEND instruction



## ● GOTO instruction



### 2.5.6.2 Supplementary Editing Keys for NC Program Editing

: To switch different instructions on the same level.

: To insert an empty command in front of the current command.

: To delete the current command.

: To reset the parameters of current command displayed.

, : To move between DRV or DRVC instruction parameters.

or : To move up or down by one instruction.

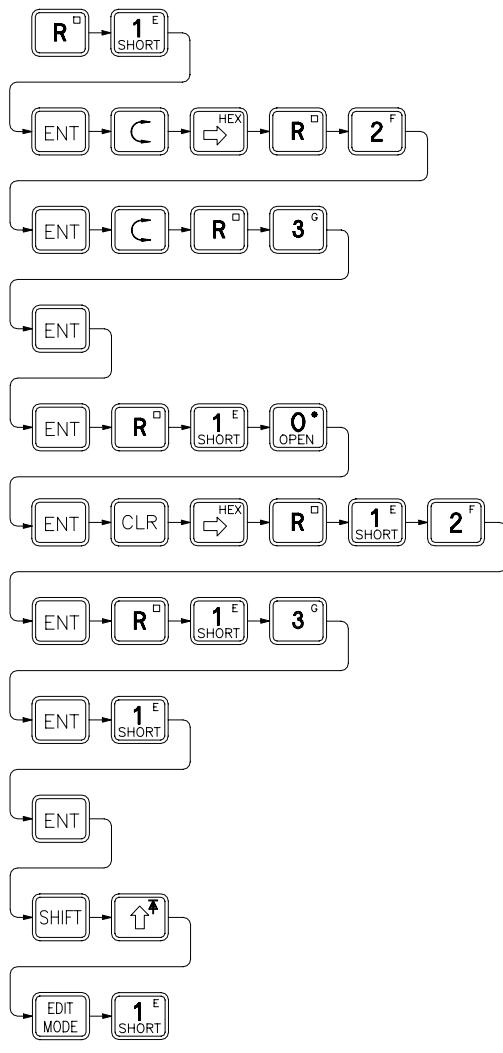
: To move up to the SPD of the first command.

: To move up to the SPD of the last command (a new empty command).

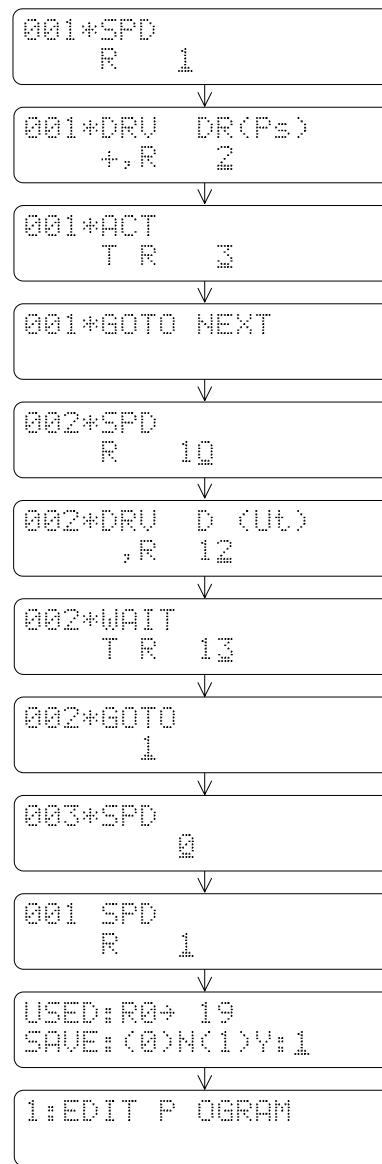
: To leave HPSO instruction editing. Then, FP-07 will display the range in the register being used by the whole commands and the user will be asked whether to save or not.

### 2.5.6.3 Editing Example

【Key Sequence】



【LCD Display】



- "001": Indicating the first set.

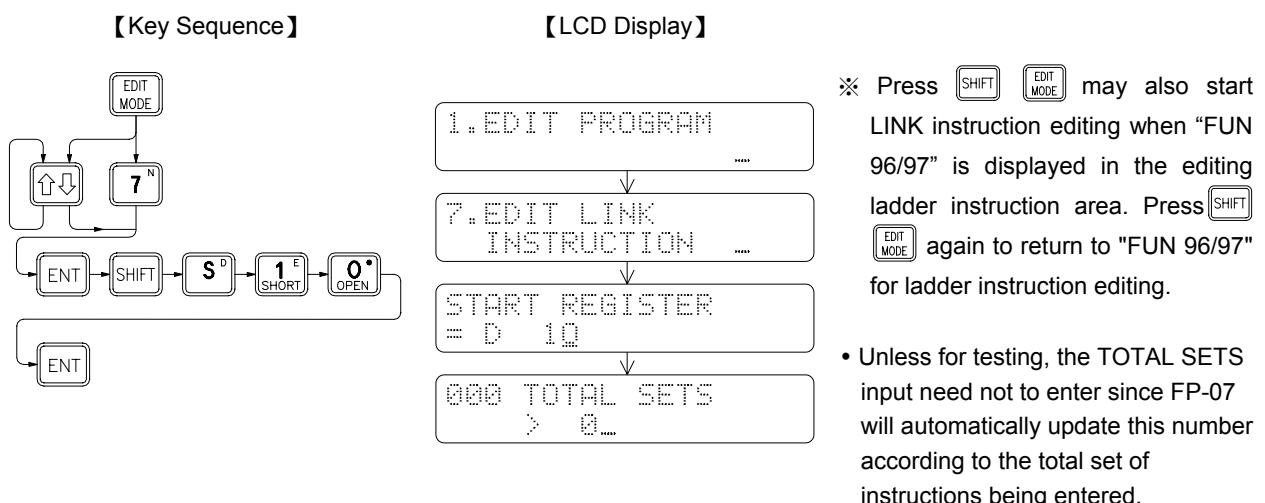
"\*" : Indicating the set has not been input completely.

- Save the edited command to (R0~R19)

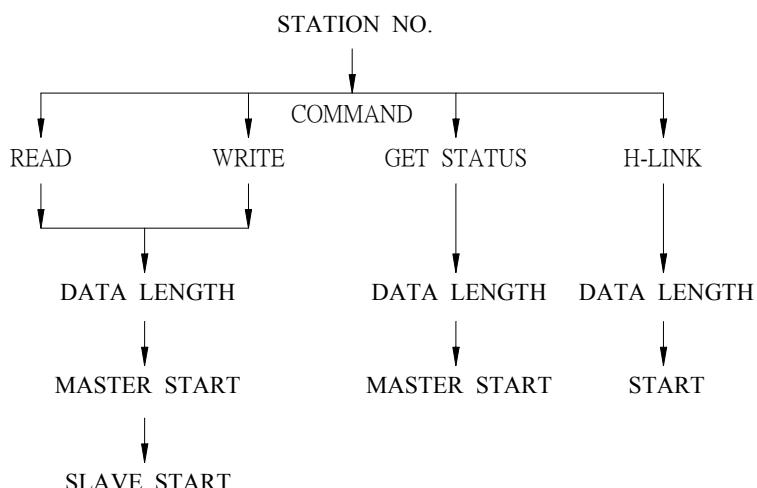
## 2.5.7 EDIT LINK INSTRUCTION

There are two unique LINK functions provided by FBe/FBN PLC for PLC networking. The LINK function is driven by the LINK instructions which are stored in the data registers. The users need only to input the starting register of the area which used to store the instructions for the link function, and then FP-07 can automatically distinguish if this area is for new entry or for editing. A LINK instruction consists of 4-5 fields of data. Once a new instruction is complete, FP-07 will store this instruction with 1 set of communication data into the data area. Press **[INS]** or **[DEL]** to insert or delete 1 set of LINK instruction. Press one of **SYS MODE**, **EDIT MODE**, **MON MODE** or **RUN STOP** to leave and save the editing, then follow the guide displayed on FP-07.

### ● LINK Instruction Editing



### ● The construction of the LINK instruction



**Remark 1:** Use **C** to select one of the four types link COMMAND.

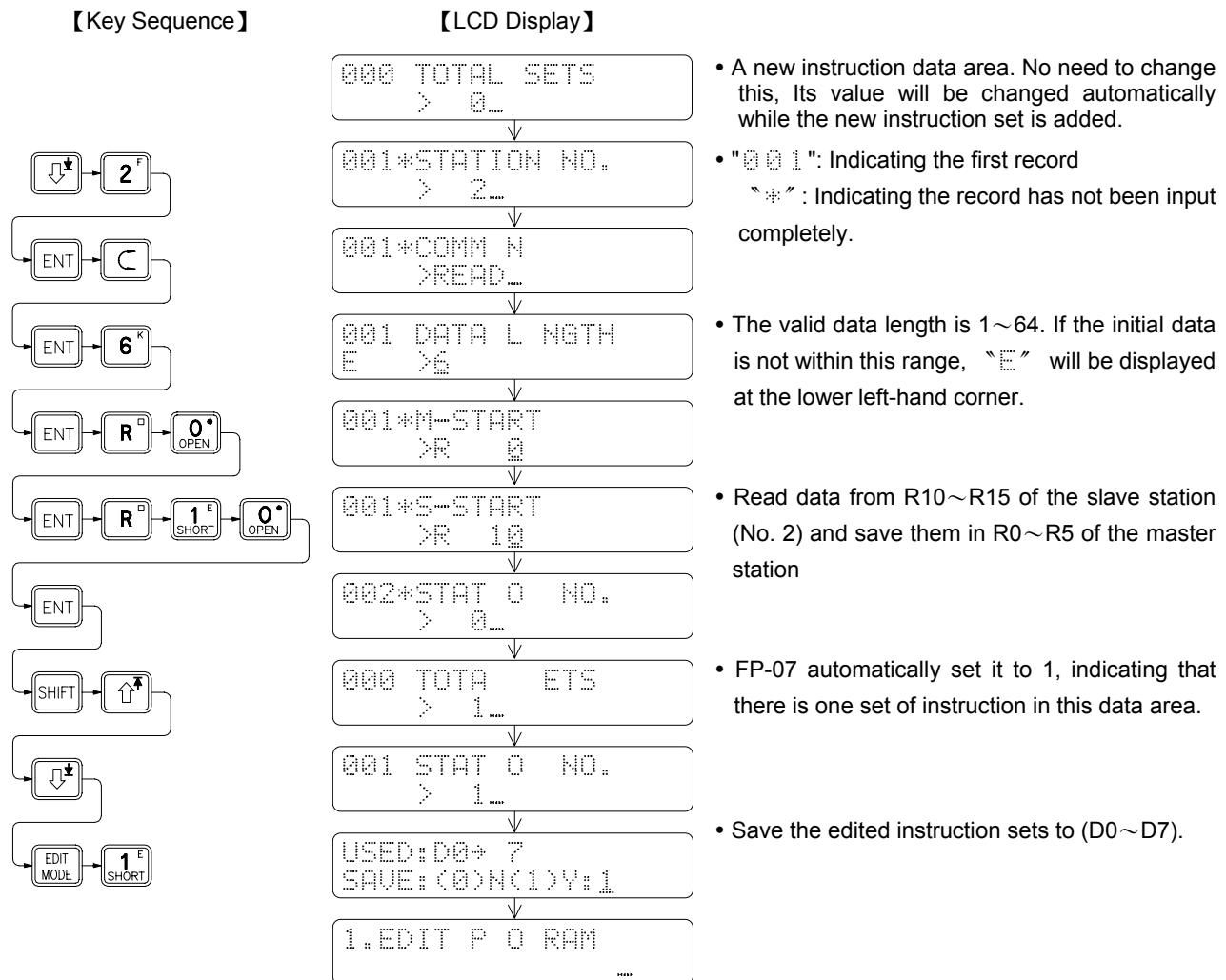
**Remark 2:** DATA LENGTH: 1~64

**Remark 3:** The available parameters for MASTER/SLAVE START include X, Y, M, S, T, C, MX, WY, WS, TR, CR, R and D.

### 2.5.7.1 Supplementary Editing Keys

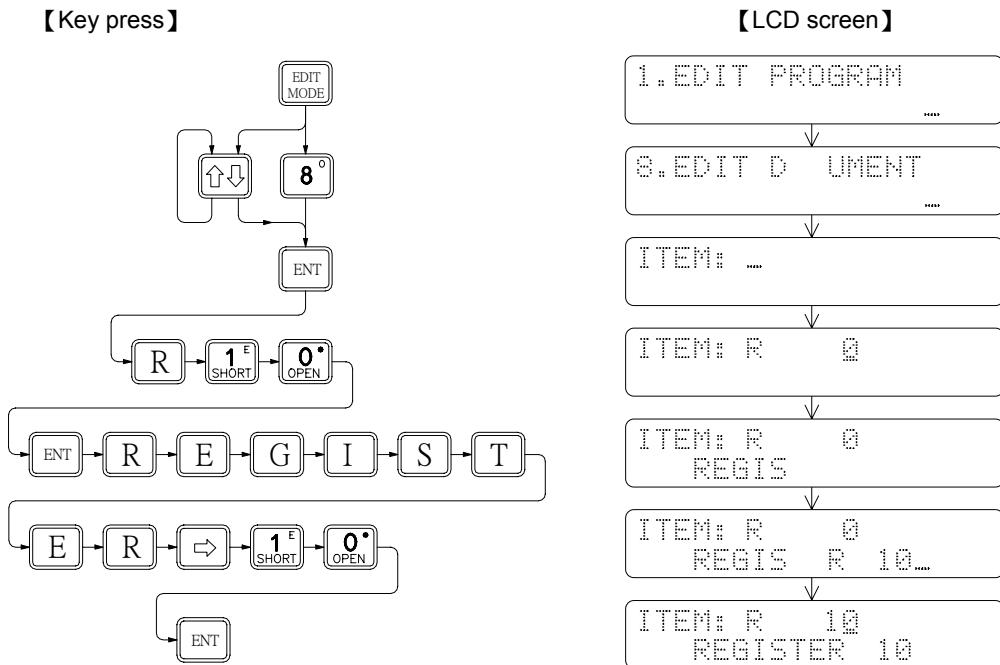
Please refer to the supplementary editing keys for HPSO in section 2.5.6.2.

### 2.5.7.2 Editing Example



## 2.5.8 EDIT DOCUMENT

This function provides the opportunity to add document for digital (X、Y、M、S、T、C) and register (R、D、WX、WY、WM、WS) with 16 characters ( Only support 10 characters while at instruction edit mode). Instead of showing reference number, all documented reference numbers will be shown by its document when displayed in FP-07 or WinProladder. If properly use this function, will increase the readability of your ladder program.



\* Key : Clear document while edit the document

Key : Move the cursor between first and second row

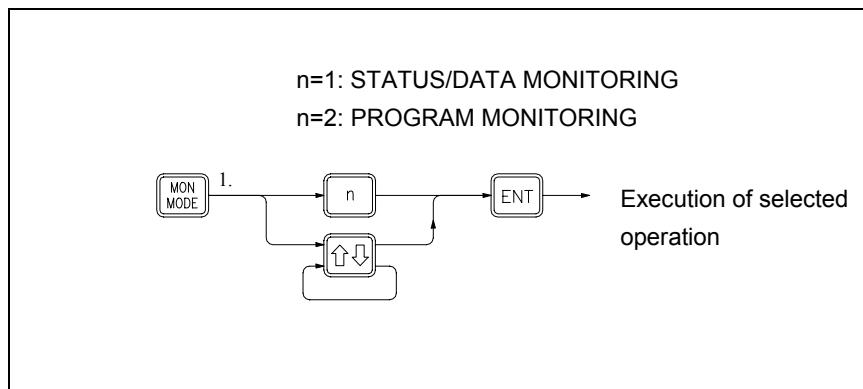
Key : Move the cursor right or left one character position while edit the document

## 2.6 The Operation of MONITOR MODE

There are two main operation functions available for MONITOR MODE.

1. STATUS/DATA MONITORING
2. PROGRAM MONITORING

Fundamental key operations of MONITOR MODE



### 2.6.1 STATUS/DATA MONITORING

This operation is used to monitor the digital status and register data. Forcing operation can be used in this mode to change the digital status and register data. In addition, digital element enable and disable control are also possible. The key operations for entering the "STATUS/DATA MONITORING" function and the corresponding LCD display is shown below.

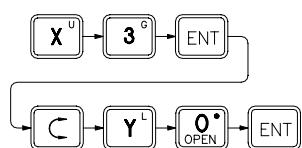


- "M" letter on the LCD screen indicates current operation mode is STATUS/DATA MONITORING mode.
- The arrow "↵" on the LCD screen prompts user to enter the digital or register reference number and then the status will be displayed on the row where the arrow is stay. Using the key to switch the arrow between these two rows.

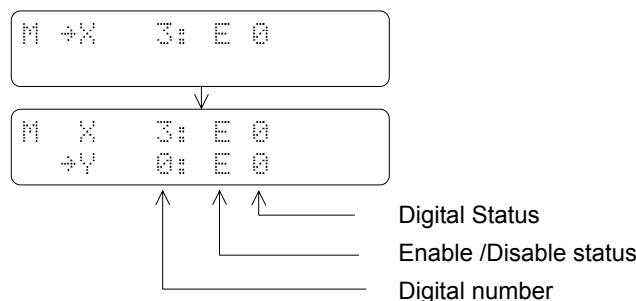
### 2.6.1.1 Digital Status Monitoring

This function allows you to monitor the statuses of all digital points, such as X△△△△, Y△△△△, M△△△△△ and S△△△△. If you want to monitor the status of a digital point after entering the STATUS/DATA MONITORING mode, follow the key operations shown below.

【Key Sequence】



【LCD Display】



The ability of forcing or enable/disable the digital status is the auxiliary function of the digital status monitor function. Therefore if you want to execute the force or enable/disable functions, it is required to monitor the digital status first. As shown in the above diagram, the enable/disable status is displayed as well while you are monitoring the digital status. At this time, you can enable or disable the digital by pressing the **[EN/DIS]** key. A specific function (Enable or Disable) can be selected by pressing the key alternately. The digital status can be forced to 1 or 0 by pressing the **[SET/RST]** key. Please refer to section 2.2.4 for detailed descriptions on force and enable/disable of the digital status.

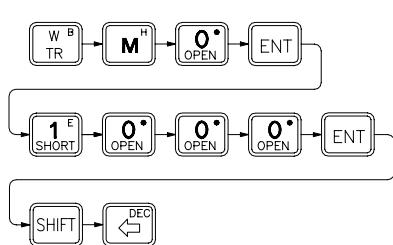
### 2.6.1.2 Register Data Monitoring

While monitoring the register data, the display data can be in decimal or hexadecimal format. FP-07 base on the register type will automatically arrange the display format. Registers consisting of 16 or 32 bits (W□△△△△△, DW□△△△△△) are displayed in hexadecimal format, with this format can easily to know the status of individual bits of register's data. Since those registers are bit oriented in application. The other register's data are displayed in decimal format. Also can use the **[SHIFT] [HEX]** or **[SHIFT] [DEC]** keys to change the display format as desired. After change the display format of certain type of register, all the register of that type will all display with the same format unless the monitoring register type is changed such as changing R△△△△ to W□△△△△△ or vice versa, or simply use the keys described above to change the display format.

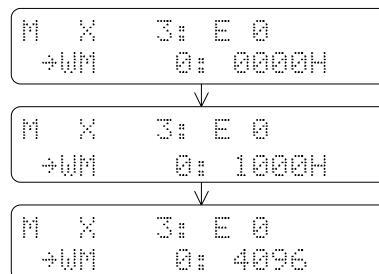
In the following example, we use the R registers and digital register to demonstrate the operation of register data monitor.

#### Example 1 16-Bit digital points register data monitoring

【Key Sequence】



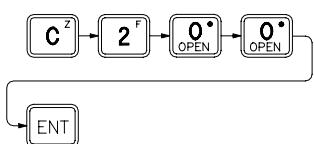
【LCD Display】



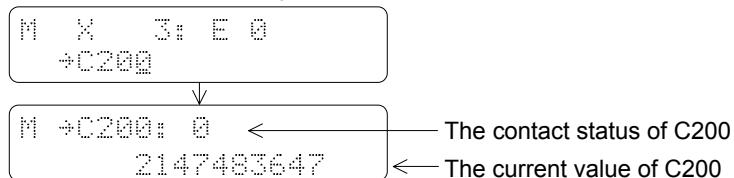
## Example 2 32-bit register status monitoring

The data of the 32-bit registers (DR△△△△, C200~C255 and DW□△△△△) may be up to 11 digits including the sign. Therefore a single LCD screen can only display one register data at a time. While displaying a 32-bit register, after pressing the **ENT** key, FP-07 will automatically display the register number and contact status (if it is a counter register) at first row, and the register data will be displayed in second row as shown in the below.

### 【Key Sequence】



### 【LCD Display】



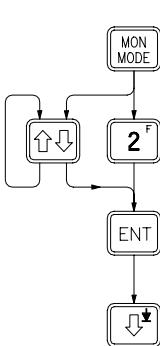
### 2.6.1.3 Change the Register Data

The register data displayed on the LCD screen can be changed as shown in example 1 where the data of WM0 is set to 1000H. However, the disable operation on register does not provide. Please refer to section 2.2.5 for more detailed explanations. To change the register data, besides by entering the register data monitor mode, using the register data edit function is a more convenient way. Please refer to section 2.5.2 for more explanations.

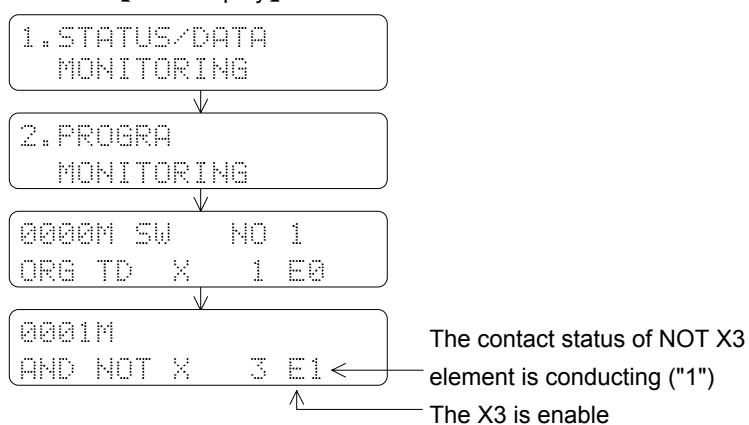
## 2.6.2 PROGRAM MONITORING

The PROGRAM MONITORING function can observe and check the program instructions either the PLC is at RUN or STOP state. If the displayed instruction is a contact element, then FP-07 will display the conducting status and enable/disable status of corresponding element. At the same time it can also perform the forcing or enable/disable operation on that digital element. The key operations are similar to the ones shown in the status monitor function. Under the PROGRAM MONITORING mode can also use the search function to find the desired instruction for monitoring. Key operations of entering the PROGRAM MONITORING mode and the LCD displays are shown below.

### 【Key Sequence】



### 【LCD Display】

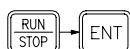


**Remark:** The digital status shown in the STATUS/DATA MONITORING function is the status of the digital numbers such as X0, Y0 etc, while the digital status shown in the PROGRAM MONITORING function is the conducting state of the digital elements such as NOT X0, TU Y0 etc. The status of X3 in the data monitor mode is 0 as shown in the example 1. But the status of the element NOT X3 in the program monitor mode is 1 as shown in the example above. This is due to the fact that X3 status is 0, but the conducting status of B contact of X3 is 1. The conducting status can only be retained for a single scan time for differential up and differential down contacts. For that contacts the conducting status may not be observed unless at the same moment the data is happen to read and displayed.

- After entering the PROGRAM MONITORING mode, can also perform the force and enable/disable operation directly on relay and contact instructions. Please refer to section 2.2.4 for key operations.
- The operation procedures for functions such as program browse and program search in the MONITOR MODE are similar to those operation procedures in the EDIT MODE. The only difference is that the data can only be observed in the MONITOR MODE while it can be edited in the EDIT MODE.

## 2.7 PLC Run/Stop Control

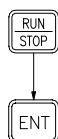
Fundamental key operations



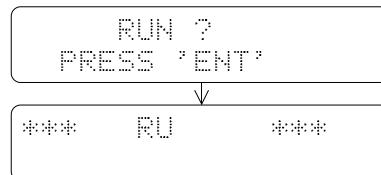
After pressing the keystrokes as shown at left, will toggle the executing state of the PLC. Which means if the PLC is at STOP state then it will change to RUN state, and vice versa.

- The key operations for the case of changing the PLC from STOP to RUN.

【Key Sequence】



【LCD Display】



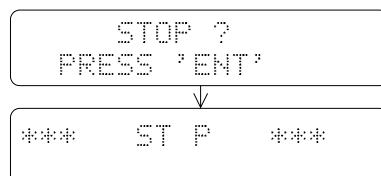
Message "RUN" blinks which indicates the PLC is running.

- The key operations for the case of changing the PLC from RUN to STOP.

【Key Sequence】

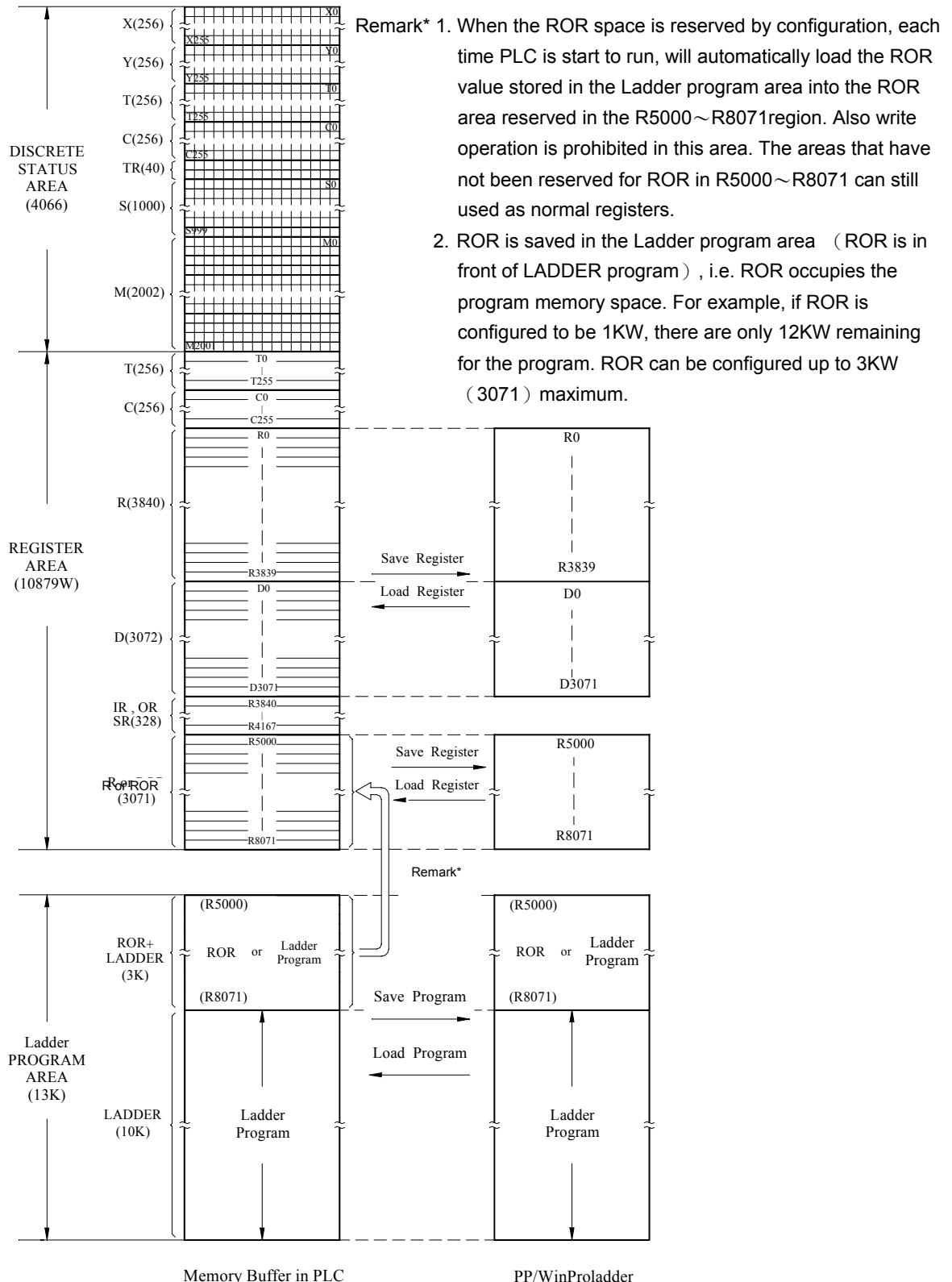


【LCD Display】



# Chapter 3 FB-PLC Memory Allocation

## 3.1 FB-PLC Memory Allocation

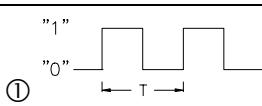
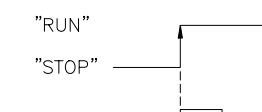
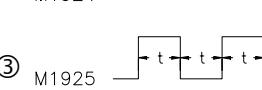


### 3.2 Digital and Register Allocations

Type	Symbol	Item		Range	Remarks	
Digital « Bit status »	X	Input contact		X0~X255 (256)	Map to external I/O terminals	
	Y	Output contact		Y0~Y255 (256)		
	TR	Temporary Relay		TR0~TR39 (40)	For branched points	
	M	Internal Relays	Non Retentive	M0~M799 (800) M1400~M1911 (512)	M0~M1399 can be configured as Non Retentive or Retentive, M1400~M1911 fixed to Non Retentive	
			Retentive	M800~M1399 (600)		
	Special Relay		M1912~M2001 (90)			
	S	Step Relays	Non Retentive	S0~S499 (500)		
			Retentive	S500~S999 (500)		
	T	Timer contact status		T0~T255 (256)		
	C	Counter contact status		C0~C255 (256)		
Register « Word Data »	TMR	Timer Registers (current value)		T0~T255 (256)		
	CTR	Current Register (CV)	16 BIT	Retentive	C0~C139 (140)	
			32 BIT	Non Retentive	C140~C199 (60)	
			32 BIT	Retentive	C200~C239 (40)	
			32 BIT	Non Retentive	C240~C255 (16)	
	DR or HR	Data Registers	Retentive		R0 ~ R3839 can configure as Retentive or Non Retentive, D0 ~D3071 fixed to Non Retentive	
			Non Retentive			
	IR	Input Registers		R3840~R3903 (64)	Map to external A/D input point	
	OR	Output Registers		R3904~R3967 (64)	Map to external D/A input point	
	SR (special register)	Special Registers		R3968~R4095 (128) R4136~R4167 (29)	Except R4152~R4154	
		HST Registers		R4152~R4154 (3)	Only provided by MC models	
		HSC Registers	Hardware (4sets)			
			Software(4sets)			
		Calendar Registers	Minute	R4129	R4128	
			Day	R4131	R4130	
			Year	R4133	R4132	
			Hrs+Min	R4135	R4134	
	ROR	ROR Registers		R5000~R8071(3072)	ROR + LADDER program=13KW	
				R5000~R8071 if not configure as ROR registers, can be used as normal registers (write, read)		
	XR	Index Registers		V、Z (2)	Please refer to section 6.1 and 6.2 for the indirect addressing	

Remark: All contents in non-retentive relays or registers will be clear to 0 when the PLC is cycled power OFF then ON or when the PLC changes from STOP→RUN. The retentive relays or registers will remain the same state before power off or STOP.

### 3.3 Special Relay Details

Relay No.	Functions	Description
1. Stop, Prohibited Control		
M1912	Emergency Stop control	<ul style="list-style-type: none"> <li>If ON PLC will be stopped (but not enter STOP mode) and all output OFF, this bit will be cleared automatically when the power is cycle off then on or the RUN command is executed</li> </ul>
M1913	External outputs disable control	<ul style="list-style-type: none"> <li>All outputs at terminal are turn off but the status of Y0~Y255 inside the PLC will not be affected</li> </ul>
M2001	Disable/enable status retentive control	<ul style="list-style-type: none"> <li>If M2001 is 0 or enabled, the disable/enable status of all contacts will be reset to enable each time the PLC is turned on or the PLC changes the state from STOP to RUN.</li> <li>If M2001 is disabled and force On, the Disable/Enable status of all contacts still can be kept no matter PLC is power off or changes the state from STOP to RUN. While testing, may disable and force On M2001 if it is necessary to disable contacts and keep disabled while STOP or power off. But don't forget to enable the M2001 after testing.</li> </ul>
2. CLEAR Control		
M1914	Non Retentive type Relays CLEAR	<ul style="list-style-type: none"> <li>Cleared When at 1</li> </ul>
M1915	Retentive type Relays CLEAR	<ul style="list-style-type: none"> <li>Cleared When at 1</li> </ul>
M1916	Non Retentive type registers CLEAR	<ul style="list-style-type: none"> <li>Cleared When at 1</li> </ul>
M1917	Retentive registers CLEAR	<ul style="list-style-type: none"> <li>Cleared When at 1</li> </ul>
M1918	Master Control (MC) selection	<ul style="list-style-type: none"> <li>If 0, the pulse activated functions within the master control loop will only execute once.</li> </ul> <p>For normal application please set this bit to 1 especially for the applications that have pulse activated functions within the master control loop</p>
M1919	Function output control	<ul style="list-style-type: none"> <li>If 0, when the function input EN is 0 (not execute) the status of the coils attached to the function outputs will not be overwritten.</li> <li>If 1, when the function input EN is 0 (not execute) the status of the coils attached to the function outputs will reflect the actual function output.</li> </ul>
※M1918/M1919 can be set to 0 or 1 at will around the whole program to meet the control requirements.		
3. Pulse Signals		
 M1920  M1921  M1922  M1923  M1924  M1925  M1926	0.01S Clock pulse 0.1S Clock pulse 1S Clock pulse 60S Clock pulse Initial (first scan) pulse ② Scan clock pulses ③ Unused	<div style="display: flex; align-items: center;"> <div style="margin-right: 20px;"> <span>①</span>              T is the pulse period         </div> <div style="margin-right: 20px;">           T(M1920)=0.01S            T(M1921)=0.1S            T(M1922)=1S            T(M1923)=60S         </div> </div> <div style="display: flex; align-items: center;"> <div style="margin-right: 20px;"> <span>②</span>              t is the scan time         </div> <div style="margin-right: 20px;">  </div> </div>

Relay No.	Functions	Description
■M1927	CTS input status of communication port1	<ul style="list-style-type: none"> <li>• 0 : CTS True (ON)</li> <li>• 1 : CTS False (OFF)</li> <li>• When communication port 1 is used for connection to a printer or a modem, can use this signal and a timer to detect whether the printer or the modem is ready (ON).</li> </ul>
<b>4.Error Messages</b>		
■M1928	Battery energy low	• 1: Indicating battery energy low
■M1929	Unused	
■M1930	No expansion unit or exceed the limit on number of I/O points	• 1: Indicating no expansion unit or exceed the limit on number of I/O points
■M1931	Immediate I/O not in the main unit range	• 1: Indicating that Immediate I/O not in the main unit range and the main unit cannot RUN
■M1932	Unused	
■M1933	Program control error	• 1: Indicating that Return instructions ( RTS or RTI ) of sub-program or interruption program have been skipped over and cannot be found (system STACK error)
■M1934   ■M1939	Unused	
<b>5.HSC0 · HSC1 Controls (for MC only)</b>		
M1940	HSC0 software MASK	• 1: MASKED
M1941	HSC0 software CLEAR	• 1: CLEARED
M1942	HSC0 software direction selection	• 0: Count-up, 1: Count-down
M1943	Unused	
M1944	Unused	
M1945	Unused	
M1946	HSC1 software MASK	• 1: MASKED
M1947	HSC1 software CLEAR	• 1: CLEARED
M1948	HSC1 software direction selection	• 0: Count-up, 1: Count-down
M1949	Unused	
M1950	Unused	
M1951	Unused	
<b>6.RTC control (for MA/MC only)</b>		
M1952	RTC setting	
M1953	± 30 second Adjustment	• Please refer to Chapter 13 of "Advanced Manual" for details
■M1954	RTC installation checking	
■M1955	Set value error	
<b>7.Communication PORT1~2 control</b>		
M1956	Reserved	
M1957	The CV value control after the timer "Time Up" for timer	<ul style="list-style-type: none"> <li>• 0: The CV value will continue timing until the upper limit is met after "Time Up".</li> <li>• 1: The CV value will stop at the PV value without further increase after "Time Up" (users may change the M1957 status within the program to control the individual timer ).</li> </ul>

Relay No.	Functions	Description
M1958	Communication port2 High Speed Network Selection	<ul style="list-style-type: none"> <li>• 0: Set Port 2 (RS-485) to Normal Speed LINK.</li> <li>• 1: Set Port 2 to High Speed CPU LINK.</li> </ul> <p>※M1958 is only effective at slave station</p>
M1959	Modem dialing mode selection	<ul style="list-style-type: none"> <li>• 0: Set dialing by TONE when Communication Port 1 is set to function with Modem.</li> <li>• 1: Set dialing by PULSE when Communication Port 1 is set to function with Modem.</li> </ul>
M1960	Communication port1 working status	<ul style="list-style-type: none"> <li>• 0: Communication Port 1 is transmitting.</li> <li>• 1: Communication Port 1 is Ready for next communication command.</li> </ul>
M1961	Communication port1 working status	<ul style="list-style-type: none"> <li>• 1: Indicating all transactions of FUN 97 (LINK Mode 0) have been completed and will stay ON for 1scan time.</li> </ul>
M1962	Communication port2 working status	<ul style="list-style-type: none"> <li>• 0: Communication Port 2 is transmitting.</li> <li>• 1: Communication Port 2 is Ready for next communication command.</li> </ul>
M1963	Communication port2 working status	<ul style="list-style-type: none"> <li>• 1: Indicating all transactions of FUN 96 (LINK Mode 0,3) have been completed and will stay ON for 1scan time.</li> </ul>
M1964	Modem dialing controls	<ul style="list-style-type: none"> <li>If Communication Port 1 is connected with Modem, when signal 0→1 will dial the phone number. When signal 1→0 will hang-up the phone.</li> </ul>
M1965	Dialing success flag	<ul style="list-style-type: none"> <li>• 1: Indicating that dialing is successful (when Communication Port 1 is connected with Modem).</li> </ul>
M1966	Dialing fail flag	<ul style="list-style-type: none"> <li>• 1: Indicating that dialing has failed (when Communication Port 1 is connected with Modem).</li> </ul>
M1967	Communication Port 2 High Speed Network Link working mode Selection	<ul style="list-style-type: none"> <li>• 0: Continuous cycle.</li> <li>• 1: One cycle only that stops when the last communication transaction is completed (only effective at the master station).</li> </ul>
M1968	Step program status	<ul style="list-style-type: none"> <li>• 1: Indicating that there are more than 16 active steps in the step program at the same time.</li> </ul>
M1969	Indirect addressing illegal write flag	<ul style="list-style-type: none"> <li>• 1: Indicating that a function with index addressing has been written to R3840~R4167 area (refer to Section 6.2 for details).</li> </ul>
M1970	Port0 status	<ul style="list-style-type: none"> <li>• 1: port0 has been received and transmitted a message.</li> </ul>
M1971	Port1 status	<ul style="list-style-type: none"> <li>• 1: port1 has been received and transmitted a message.</li> </ul>
M1972	Port2 status	<ul style="list-style-type: none"> <li>• 1: port2 has been received and transmitted a message.</li> </ul>
M1973	The CV value counting mode selection after the timer "Time-Up"	<ul style="list-style-type: none"> <li>• 0: Indicating that the CV value will continue counting up to the upper limit after "Time-Up".</li> <li>• 1: Indicating that the CV value will not increase further after "Time-Up" (the CV value stops at the PV value) (Users may set the M1973 status to have multiple or dynamic selection of counter timing mode) .</li> </ul>
M1974	Ramp function(FUN95) slope control	<ul style="list-style-type: none"> <li>• 0: Controlled by time.</li> <li>• 1: Constant slope</li> </ul>
M1975	Drum function (FUN112) limit restriction selection	<ul style="list-style-type: none"> <li>• 1: Set when for the circular applications where high limit &lt; low limit</li> </ul>

#### 8.HSC2~7 Controls ( For MC only if HSC2 & HSC3 are hardware high speed counters )

M1976	HSC2 software MASK	<ul style="list-style-type: none"> <li>• 1: MASKED</li> </ul>
M1977	HSC2 software CLEAR	<ul style="list-style-type: none"> <li>• 1: CLEARED</li> </ul>
M1978	HSC2 software Direction Selection	<ul style="list-style-type: none"> <li>• 0: Count-up, 1: Count-down</li> </ul>
M1979	HSC3 software MASK	<ul style="list-style-type: none"> <li>• 1: MASKED</li> </ul>
M1980	HSC3 software CLEAR	<ul style="list-style-type: none"> <li>• 1: CLEARED</li> </ul>

Relay No.	Functions	Description
M1981	HSC3 software Direction Selection	• 0: Count-up, 1: Count-down
M1982	HSC4 software MASK	• 1: MASKED
M1983	HSC4 software Direction Selection	• 0: Count-up, 1: Count-down
M1984	HSC5 software MASK	• 1: MASKED
M1985	HSC5 software Direction Selection	• 0: Count-up, 1: Count-down
M1986	HSC6 software MASK	• 1: MASKED
M1987	HSC6 software Direction Selection	• 0: Count-up, 1: Count-down
M1988	HSC7 software MASK	• 1: MASKED
M1989	HSC7 software Direction Selection	• 0: Count-up, 1: Count-down
M1990	Unused	
M1991		
9.PS0~3 Controls ( For MC only )		
M1992	HPSO PS0 working status	• 0: PS0 is sending the pulse • 1: PS0 is Ready for new command
M1993	HPSO PS1 working status	• 0: PS1 is sending the pulse • 1: PS1 is Ready for new command
M1994	HPSO PS2 working status	• 0: PS2 is sending the pulse • 1: PS2 is Ready for new command
M1995	HPSO PS3 working status	• 0: PS3 is sending the pulse • 1: PS3 is Ready for new command
M1996	HPSO PS0 working status	• 1: ON when the final step of FUN 140(HPSO) is completed
M1997	HPSO PS1 working status	• 1: ON when the final step of FUN 140(HPSO) is completed
M1998	HPSO PS2 working status	• 1: ON when the final step of FUN 140(HPSO) is completed
M1999	HPSO PS3 working status	• 1: ON when the final step of FUN 140(HPSO) is completed
M2000	HPSO Synchronized Multi-Axis Selection	• 1: Synchronized Multi-Axis

### 3.4 Special Registers Details

Register No.	Functions	Description
R3968   R3999	Raw temperature measurement value.	R3968 is for point 1, R3969 is for point 2, ..... and R3999 is for point 32.
R4000	High Byte: Temperature value average selection.  Low Byte: FUN 85(TPSNS) flag	High Byte: 0, no average on temperature 1, average by two readings 2, average by four readings 3, average by eight readings 4, average by sixteen readings
R4001	Scaling factor for positive temperature	Low Byte: Maintained by FUN85, do not modify it. Positive temperature value = raw value × R4001 ÷ 1024 (unipolar) raw value × 2 × R4001 ÷ 1024 (bipolar)

Register No.	Functions	Description
R4002	Scaling factor for negative temperature	Negative temperature value = raw value $\times$ R4002 $\div$ 1024 (-5V~5V) raw value $\times$ 2 $\times$ R4002 $\div$ 1024 (-10V~10V)
R4003	Thermocouple wire broken threshold value	Default value: 0~10, -10~10V = 901 0~5,0~10V = 451
R4004	Temperature scan time interval between channels (FUN85)	Unit in mS, default 330mS. That is, 24 points of temperature measurement will be completed in 2 seconds.
R4005	High Byte : Temperature control PWM period selection  Low Byte: Temperature control PID calculation cycle selection	High Byte : 0, period = 2 sec 1, period = 4 sec 2, period = 8 sec >2, period = 1 sec  Low Byte : 0, calculation cycle = 2 sec 1, calculation cycle = 4 sec 2, calculation cycle = 8 sec >2, calculation cycle = 1 sec
R4006	Threshold value for heating control output abnormal detecting	Used for FUN73 & FUN86
R4007	Threshold time for heating control output abnormal detecting	Used for FUN73 & FUN86
R4008	Threshold temperature for heating control output abnormal detecting	Used for FUN73 & FUN86
R4010   R4011	Installed temperature sensor flag	Each bit represent 1 sensor, if bit value = 1 means installed.
R4012   R4013	Temperature PID control flag	Each bit represent 1 temperature point, if bit value = 1 means enable control.
R4014	Temperature scan time interval between channels (FUN72 & FUN73)	Used for FUN72 & FUN73
R4015	Temperature value averaging selection.	=0, no average on temperature =1, average by two readings =2, average by four readings =3, average by eight readings =4, average by sixteen readings
R4016	Scaling factor of positive temperature for K type thermocouple  FUN72 & FUN73	Positive temperature value = raw value $\times$ R4016 $\div$ 1024 (unipolar) raw value $\times$ 2 $\times$ R4016 $\div$ 1024 (bipolar)
R4017	Scaling factor of negative temperature for K type thermocouple  FUN72 & FUN73	Negative temperature value = raw value $\times$ R4017 $\div$ 1024 (-5V~5V) raw value $\times$ 2 $\times$ R4017 $\div$ 1024 (-10V~10V)
R4018	Scaling factor of positive temperature for J type thermocouple  FUN72 & FUN73	Positive temperature value = raw value $\times$ R4018 $\div$ 1024 (unipolar) raw value $\times$ 2 $\times$ R4018 $\div$ 1024 (bipolar)
R4019	Scaling factor of negative temperature for J type thermocouple  FUN72 & FUN73	Negative temperature value = raw value $\times$ R4019 $\div$ 1024 (-5V~5V) raw value $\times$ 2 $\times$ R4019 $\div$ 1024 (-10V~10V)

Register No.	Functions	Description
R4020   R4023	Reserved	
R4024	Analog points detection control	<ul style="list-style-type: none"> <li>If R4024=55AAH, set AI = 8 points AO = 8 points</li> <li>If R4024=AA55H, analog points can be set manually by writing the R4025 &amp; R4026</li> <li>If R4024 is other than above two values then the analog points will detect by system. The scanning result will put in R4025 &amp; R4026</li> </ul>
R4025 R4026 R4027 R4028	Total AI points Total AO points Total DI points Total DO points	
R4029	Ladder checksum counter	
R4030   R4047	Reserved	
R4048	Output signal global off control	If R4048 = 6789H will turn off all output but the input still valid. This can be used for wiring checking.
R4049	CPU running status	= A55AH will force PLC to RUN = 0, Normal STOP = 1, Function in the program not support by this CPU = 2, PLC ID not matched with Program ID = 3, Program error = 4, Abnormal STOP = 5, Watch dog fail = 6, I/O point accessed by program is not available or improper usage of RTS, RTI instruction.
R4050   R4052	Reserved	
R4053	Port2 response delay time	Unit in mS, default 4 mS
R4054	Master station number of the high-speed network CPU link.	If the master station number is 1 can ignore this register. To set the master station number other than 1 should: Low byte : station number High byte: 55H
R4055	PLC station number	<ul style="list-style-type: none"> <li>If high byte is not equal 55H, R4055 will show the station number of this PLC</li> <li>If want to set PLC station number then R4055 should set to: Low byte : station number High byte: 55H</li> </ul>
R4056	High Byte: High speed pulse output frequency ACC./DEC. control  Low Byte: High speed pulse output frequency dynamic control	High Byte: =1, Auto ACC./DEC. while HPSO frequency is dynamically changed =0, No automatic ACC./DEC.  Low Byte: =5AH, can dynamically change HPSO frequency
R4057	Power off counter	

Register No.	Functions	Description
R4058 R4059	PLC station number with errors Port2 High Speed CPU LINK error code	(Port2 High Speed CPU LINK) (Port2 High Speed CPU LINK)  High Byte                          Low Byte R4059      error code      error count      H Error code: 0AH, Slave station no response 0BH, Data abnormal (CRC Error) 20H, Parity Error 40H, Framing Error 80H, Over-Run Error
R4060	HPSO error code	The error code are: 1: Parameter 0 error 2: Parameter 1 error 3: Parameter 2 error 4: Parameter 3 error 5: Parameter 4 error 7: Parameter 6 error 8: Parameter 7 error 9: Parameter 8 error 10: Parameter 9 error 30: Speed setting reference number error 31: Speed value error 32: Traveling distance reference number error 33: Traveling distance value error 34: Illegal positioning program 35: Step length error 36: Step number exceed 255 37: Highest frequency value error 38: Idle frequency value error 39: Movement compensation value too large 40: Movement value exceed range 41: DRVC instruction not allow ABS addressing
R4061 R4062 R4063	HPSO PS1 error code HPSO PS2 error code HPSO PS3 error code	Ditto Ditto Ditto
R4064 R4065	The latest step number of completed instruction of PS0 motion program The latest step number of completed instruction of PS1 motion program	
R4066 R4067 R4068   R4071	The latest step number of completed instruction of PS2 motion program The latest step number of completed instruction of PS3 motion program Used by the system	
R4072	Low Word of PS0 pulse count remaining for output	

Register No.	Functions	Description
R4073	High Word of PS0 pulse count remaining for output	
R4074	Low Word of PS1 pulse count remaining for output	
R4075	High Word of PS1 pulse count remaining for output	
R4076	Low Word of PS2 pulse count remaining for output	
R4077	High Word of PS2 pulse count remaining for output	
R4078	Low Word of PS3 pulse count remaining for output	
R4079	High Word of PS3 pulse count remaining for output	
R4080	Current PS0 output frequency Low Word	
R4081	Current PS0 output frequency High Word	
R4082	Current PS1 output frequency Low Word	
R4083	Current PS1 output frequency High Word	
R4084	Current PS2 output frequency Low Word	
R4085	Current PS2 output frequency High Word	
R4086	Current PS3 output frequency Low Word	
R4087	Current PS3 output frequency High Word	
R4088	Current Ps value Low Word in PS0	
R4089	Current Ps value High Word in PS0	
R4090	Current Ps value Low Word in PS1	
R4091	Current Ps value High Word in PS1	
R4092	Current Ps value Low Word in PS2	
R4093	Current Ps value High Word in PS2	
R4094	Current Ps value Low Word in PS3	
R4095	Current Ps value High Word in PS3	
█ R4136	Current scan time	• Error < ±1ms
█ R4137	Maximum scan time	• Re-calculate when PLC changes from STOP to RUN
█ R4138	Minimum scan time	
R4139	CPU Status	Bit0=0, PLC STOP =1, PLC RUN Bit1=1, Low battery Bit2=1, Ladder program checksum error Bit3=0, Ladder save in RAM =1, Ladder save in ROM-PACK Bit4=1, Watch-Dog error Bit5=1, MA type CPU Bit6=1, PLC ID Protection enable Bit7=1, Emergency STOP Bit8=0, V1.x version ASIC =1, V2.x version ASIC Bit9=1, V1.x version ladder program Bit10=1, ASIC malfunction Bit11=1, Ladder instruction CPU not support Bit12=0, V2.X version ASIC =1, V3.X version (Bit8=1) ASIC Bit13=1, MU type CPU Bit14=1, Calendar installed Bit15=1, MC type CPU

Register No.	Functions	Description																																										
R4140 R4141 R4142 R4143 R4144 R4145	Telephone Number	<ul style="list-style-type: none"> <li>• For MC only</li> </ul>																																										
R4146	Communication Port 1 communication parameter setting	<ul style="list-style-type: none"> <li>• If R4146 high byte value is not equal to 55H, the default parameters will be:           <table> <tr> <td>Baud Rate 9600</td> <td>Even Parity</td> <td>7 Data Bit</td> <td>1 Stop Bit</td> <td rowspan="2">Communication Port 1 is functioning normally or set to Modem or LINK</td> </tr> <tr> <td>Baud Rate 9600</td> <td>Even Parity</td> <td>8 Data Bit</td> <td>1 Stop Bit</td> </tr> </table> </li> <li>• If R4146 high byte value = 55H, the communication parameters are defined by low byte. Details are described as below:</li> </ul> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">High Byte</td> <td style="text-align: center;">Low Byte</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">R4146</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">55</td> <td style="border: 1px solid black; padding: 2px;">Communication Parameter</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">H</td> </tr> </table> <p>※ Each bit of R4146 low byte is defined as below:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Bit B2 B1 B0</th> <th>Value</th> <th>Baud Rate</th> </tr> </thead> <tbody> <tr> <td>0 0 0</td> <td>0</td> <td>19200</td> </tr> <tr> <td>0 0 1</td> <td>1</td> <td>9600</td> </tr> <tr> <td>0 1 0</td> <td>2</td> <td>4800</td> </tr> <tr> <td>0 1 1</td> <td>3</td> <td>2400</td> </tr> <tr> <td>1 0 0</td> <td>4</td> <td>1200</td> </tr> <tr> <td>1 0 1</td> <td>5</td> <td>600</td> </tr> <tr> <td>1 1 0</td> <td>6</td> <td>Cannot be used</td> </tr> <tr> <td>1 1 1</td> <td>7</td> <td>38400</td> </tr> </tbody> </table> <p>B3: must be0    B4: =0, 1 STOP Bit          =1, 2 STOP Bit    B5: =0, None Parity          =1, With Parity    B6: =0, 7 Data Bit          =1, 8 Data Bit    B7: =0, Even Parity          =1, Odd Parity    ( R4146 default value = 0)</p>	Baud Rate 9600	Even Parity	7 Data Bit	1 Stop Bit	Communication Port 1 is functioning normally or set to Modem or LINK	Baud Rate 9600	Even Parity	8 Data Bit	1 Stop Bit	High Byte	Low Byte	R4146	55	Communication Parameter	H	Bit B2 B1 B0	Value	Baud Rate	0 0 0	0	19200	0 0 1	1	9600	0 1 0	2	4800	0 1 1	3	2400	1 0 0	4	1200	1 0 1	5	600	1 1 0	6	Cannot be used	1 1 1	7	38400
Baud Rate 9600	Even Parity	7 Data Bit	1 Stop Bit	Communication Port 1 is functioning normally or set to Modem or LINK																																								
Baud Rate 9600	Even Parity	8 Data Bit	1 Stop Bit																																									
High Byte	Low Byte																																											
R4146	55	Communication Parameter	H																																									
Bit B2 B1 B0	Value	Baud Rate																																										
0 0 0	0	19200																																										
0 0 1	1	9600																																										
0 1 0	2	4800																																										
0 1 1	3	2400																																										
1 0 0	4	1200																																										
1 0 1	5	600																																										
1 1 0	6	Cannot be used																																										
1 1 1	7	38400																																										

Register No.	Functions	Description
R4147	Low byte: Port1 Time out value  High byte: Port1 inter message delay time	Low byte: Unit in 0.1S, default is 5. LINK instruction uses this time to determine if it still needs to wait the response message.  High byte: Transmission delay time between communication transactions of LINK instruction (FUN 97 Mode 0). Unit is 0.01S, default is 0
R4148	Low byte : Port1 fine time out value  High byte: End of Frame time value	Low byte: The unit of time value in this byte is 0.01S. If the resolution of time out time for the application must better than 0.1S then can use this byte to set the desired time out value (0~2.55S)  When use this byte to control the time out must set the low byte of R4147 to 0 otherwise it will not take effective. The default value of this byte is 0.  High byte: Use this byte to control the timing of end of frame detection for the operation of LINK1, LINK2 mode1 and mode2 which do not use END sync. character to determine the end of message. The unit is 0.001S. The default value of this byte is 12.
R4149	Low byte: Port0 wild receiving control  High byte: Port1 modem mode selection.	Low byte: =1, PLC will accept all 'application messages' coming from port0 even the target station number in the message is not matched with this PLC. ≠1, PLC only accepts the 'application messages' coming from port0 which target station number is matched with this PLC setting.  High byte: =0 , CPU LINK of Port1 is not connected through a Modem. =55H, CPU LINK of Port1 is connected through a Modem.
R4150	Power on I/O service delay time setting	• While power on when PLC is ready for I/O operation it will delay by this time before the actual I/O operation is started. The unit is 0.01S. The default value is 100.
R4151	1ms time base timer register	• The R4151 value will be increased by 1 every 1mS. It can be used for a more precise timing application.
R4152 R4153 R4154	CV register of HSTA high speed timing count ( 0.1ms )  High Word of HSTA CV register  PV register of HSTA	• When HSTA is act as 32-bit cyclic timer
R4155	Low byte: Port1 wild receiving control  High byte: Port2 wild receiving control	Low Byte: =1, PLC will accept all 'application messages' coming from port1 even the target station number in the message is not matched with this PLC setting.  High Byte: =1, PLC will accept all 'application messages' coming from port2 even the target station number in the message is not matched with this PLC setting.

Register No.	Functions	Description																																																																																																																				
R4156	Port0 wild programming message receiving control	<p>Low byte: =55H, PLC only accepts the ‘programming messages’ coming from Port0 which target station number is matched with this PLC.</p> <p>≠55, PLC will accept all ‘programming messages’ coming from Port0 even the target station number of the message is not matched with this PLC setting. (Default)</p>																																																																																																																				
R4157	Port2 time out value for non High Speed Link	This register is maintained by system. Do not modify the value of this register.																																																																																																																				
R4158	Port2 communication parameter for Non-High Speed CPU LINK	<ul style="list-style-type: none"> <li>• R4158 High Byte must be 55H, and then the communication parameter can be determined by the low Byte.</li> </ul> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="text-align: center;">High Byte</td> <td style="text-align: center;">Low Byte</td> </tr> <tr> <td style="border: 1px solid black; padding: 2px;">R4158</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">55H</td> <td style="border: 1px solid black; padding: 2px; text-align: center;">Communication Parameters</td> <td style="border: 1px solid black; padding: 2px; text-align: right;">H</td> </tr> </table> <ul style="list-style-type: none"> <li>• Each bit of 4158 Low Byte is defined as below:</li> </ul> <table style="margin-left: auto; margin-right: auto;"> <tr> <td>B7=0 , Even Parity =1 , Odd Parity</td> </tr> <tr> <td>B6=0 , 7 Data Bit =1 , 8 Data Bit</td> </tr> <tr> <td>B5=0 , None Parity =1 , With Parity</td> </tr> <tr> <td>B4=0 , 1 Stop Bit =1 , 2 Stop Bit</td> </tr> </table> <table border="1" style="margin-left: auto; margin-right: auto; width: fit-content;"> <thead> <tr> <th colspan="4">BIT</th> <th rowspan="2">Value</th> <th rowspan="2">Baud Rate</th> </tr> <tr> <th>B3</th> <th>B2</th> <th>B1</th> <th>B0</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>4800</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>9600</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td><td>2</td><td>19200</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>3</td><td>38400</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>0</td><td>4</td><td>76800</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>1</td><td>5</td><td>153600</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td><td>6</td><td>307200</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>7</td><td>614400</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td><td>8</td><td>9000</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>1</td><td>9</td><td>18000</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>A</td><td>36000</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>1</td><td>B</td><td>72000</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>0</td><td>C</td><td>144000</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td><td>D</td><td>288000</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>E</td><td>576000</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>F</td><td>1152000(Reserved)</td></tr> </tbody> </table> <p>The default value of R4158 is 5521H (9600, even parity, 7 data bit, 1 stop bit)</p>	High Byte	Low Byte	R4158	55H	Communication Parameters	H	B7=0 , Even Parity =1 , Odd Parity	B6=0 , 7 Data Bit =1 , 8 Data Bit	B5=0 , None Parity =1 , With Parity	B4=0 , 1 Stop Bit =1 , 2 Stop Bit	BIT				Value	Baud Rate	B3	B2	B1	B0	0	0	0	0	0	4800	0	0	0	1	1	9600	0	0	1	0	2	19200	0	0	1	1	3	38400	0	1	0	0	4	76800	0	1	0	1	5	153600	0	1	1	0	6	307200	0	1	1	1	7	614400	1	0	0	0	8	9000	1	0	0	1	9	18000	1	0	1	0	A	36000	1	0	1	1	B	72000	1	1	0	0	C	144000	1	1	0	1	D	288000	1	1	1	0	E	576000	1	1	1	1	F	1152000(Reserved)
High Byte	Low Byte																																																																																																																					
R4158	55H	Communication Parameters	H																																																																																																																			
B7=0 , Even Parity =1 , Odd Parity																																																																																																																						
B6=0 , 7 Data Bit =1 , 8 Data Bit																																																																																																																						
B5=0 , None Parity =1 , With Parity																																																																																																																						
B4=0 , 1 Stop Bit =1 , 2 Stop Bit																																																																																																																						
BIT				Value	Baud Rate																																																																																																																	
B3	B2	B1	B0																																																																																																																			
0	0	0	0	0	4800																																																																																																																	
0	0	0	1	1	9600																																																																																																																	
0	0	1	0	2	19200																																																																																																																	
0	0	1	1	3	38400																																																																																																																	
0	1	0	0	4	76800																																																																																																																	
0	1	0	1	5	153600																																																																																																																	
0	1	1	0	6	307200																																																																																																																	
0	1	1	1	7	614400																																																																																																																	
1	0	0	0	8	9000																																																																																																																	
1	0	0	1	9	18000																																																																																																																	
1	0	1	0	A	36000																																																																																																																	
1	0	1	1	B	72000																																																																																																																	
1	1	0	0	C	144000																																																																																																																	
1	1	0	1	D	288000																																																																																																																	
1	1	1	0	E	576000																																																																																																																	
1	1	1	1	F	1152000(Reserved)																																																																																																																	

Register No.	Functions	Description																
R4159	Low byte: Port2 time out value High byte: Transaction delay time for Port2 mode0 operation	Low byte : Unit in 0.01S, default is 50. LINK2 instruction uses this time to determine if it still needs to wait the response message. High byte: Transmission delay time between communication transactions of LINK2 instruction (FUN 96 Mode 0). Unit in 0.01S, default is 0.																
R4160	Port2 time out value for High Speed CPU LINK	This register is maintained by system. Do not modify the register value.																
R4161	Port2 communication parameter for High Speed CPU LINK	<ul style="list-style-type: none"> <li>The definition of communication parameter word is the same as that for R4158 but with the following exceptions</li> </ul> Data Bit is fixed as 8 Bit Baud Rate must $\geq$ 38400  Default value is 5565H (153600 bps, 8 data bit, even parity, 1 stop bit)																
R4162	Time base interrupt enable/disable control.	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>B7</td><td>B6</td><td>B5</td><td>B4</td><td>B3</td><td>B2</td><td>B1</td><td>B0</td></tr> <tr> <td>100mS</td><td>50mS</td><td>10mS</td><td>5mS</td><td>4mS</td><td>3mS</td><td>2mS</td><td>1mS</td></tr> </table> <p style="text-align: center;">Bit=0, Interruption enabled Bit=1, Interruption disabled</p>	B7	B6	B5	B4	B3	B2	B1	B0	100mS	50mS	10mS	5mS	4mS	3mS	2mS	1mS
B7	B6	B5	B4	B3	B2	B1	B0											
100mS	50mS	10mS	5mS	4mS	3mS	2mS	1mS											
R4163	Modem dialing control setting	Low byte: =1, Ignore the dialing tone and the busy tone when dialing. =2, Wait the dialing tone but ignore the busy tone when dialing. =3, Ignore the dialing tone but detect the busy tone when dialing. =4, Wait the dialing tone and detect the busy tone when dialing. =Any other value treated as value equal 4.																
R4164 R4165 R4166	V index register Z index register Reserved by the system																	
R4167	Total I/O points of main PLC unit	=1, 20-point main unit. =2, 28-point main unit. =3, 40-point main unit.																

Remark: All the special relays or registers attached with “▼” symbol shown in the above table are write prohibited.

For the special relays attached with “▼” symbol also has following characteristics

- . Forced and Enable/Disable operation is not allowed.
- . Can not referenced by TU/TD transitional contact (contact will always open)

## Chapter 4 FB-PLC Instruction Lists

### 4.1 Sequential Instructions

Instruction	Operand	Symbol	Function Descriptions	Execution Time	Instruction type	
ORG	X,Y,M, S,T,C		Starting a network with a normally open (A) contact	0.33uS	Network starting instructions	
ORG NOT			Starting a network with a normally closed (B) contact			
ORG TU			Starting a network with a differential up (TU) contact	0.54uS		
ORG TD			Starting a network with a differential down (TD) contact			
ORG OPEN			Starting a network with a open circuit contact	0.33uS		
ORG SHORT			Starting a network with a short circuit contact			
LD	X,Y,M, S,T,C		Starting a relay circuit from origin or branch line with a normally open contact	0.33uS	Origin or branch line starting instructions	
LD NOT			Starting a relay circuit from origin or branch line with a normally closed contact			
LD TU			Starting a relay circuit from origin or branch line with a differential up contact	0.54uS		
LD TD			Starting a relay circuit from origin or branch line with a differential down contact			
LD OPEN			Starting a relay circuit from origin or branch line with a open circuit contact	0.33uS		
LD SHORT			Starting a relay circuit from origin or branch line with a short circuit contact			
AND	X,Y,M, S,T,C		Serial connection of normally open contact	0.33uS	Serial connection instructions	
AND NOT			Serial connection of normally closed contact			
AND TU			Serial connection of differential up contact	0.54uS		
AND TD			Serial connection of differential down contact			
AND OPEN			Serial connection of open circuit contact	0.33uS		
AND SHORT			Serial connection of short circuit contact			
OR	X,Y,M, S,T,C		Parallel connection of normally open contact	0.33uS	Parallel connection instructions	
OR NOT			Parallel connection of normally closed contact			
OR TU			Parallel connection of differential up contact	0.54uS		
OR TD			Parallel connection of differential down contact			
OR OPEN			Parallel connection of open circuit contact	0.33uS		
OR SHORT			Parallel connection of short circuit contact			
ANDLD			Serial connection of two circuit blocks	0.33uS	Blocks merge instructions	
ORLD			Parallel connection of two circuit blocks			

Instruction	Operand	Symbol	Function Descriptions	Execution Time	Instruction type
OUT	Y,M,S	—( )	Send result to coil	0.33uS   1.09uS	Coil output instruction
OUT NOT		—( / )	Send inverted result to coil		
OUT L	Y	—( L )	Send result to an external output coil and appoint it as of retentive type		
OUT	TR		Save the node status to a temporary relay	0.33uS	Node operation instruction
LD			Load the temporary relay		
TU		—↑—	Take the transition up of the node status	0.33uS	Node operation instruction
TD		—↓—	Take the transition down of the node status	0.33uS	
NOT		—/—	Invert the node status	0.33uS	

- The 34 sequential instructions listed above are all applicable to every models of FB-PLC.

## 4.2 Function Instructions

There are more than 100 different FB-PLC function instructions. If put the “D” and “P” derivative instructions into account, the total number of instructions is over 300. On top of these, many function instructions have multiple input controls (up to 4 inputs) which can have up to 8 different types of operation mode combinations. Hence, the size of FB-PLC instruction sets is in fact not smaller than that of a large PLC. Having powerful instruction functions, though may help for establishing the complicated control applications, but also may impose a heavy burden on those users of small type PLC’s. For ease of use, FATEK PLC function instructions are divided into two groups, the Basic function group which includes 26 commonly used function instructions and 4 SFC instructions and the advanced function group which includes other more complicated function instructions, such as high-speed counters and interrupts. This will enable the beginners and the non-experienced users to get familiar with the basic function very quickly and to assist experienced users in finding what they need in the advanced set of function instructions.

The instructions attached with “★” symbol are basic functions which amounts to 26 function instructions and 4 SFC instructions. All the basic functions will be explained in next chapter. The details for the reset of functions please refer advanced manual.

### ■ General Timer/Counter Function Instructions

FUN No.	Name	Operand	Derivative Instruction	Function descriptions
★	T nnn	PV		General timer instructions (“nnn” range 0~255)
★	C nnn	PV		General counter instructions (“nnn” range 0~255)

### ■ Single Operand Function Instructions

★ 4	DIFU	D		To get the up differentiation of a D relay and store the result to D
★ 5	DIFD	D		To get the down differentiation of a D relay and store the result to D
★ 10	TOGG	D		Toggle the status of the D relay

■ SET/RESET Instructions

FUN No.	Name	Operand	Derivative instruction	Function descriptions
★	SET	D	DP	Set a discrete or all bits of a register to 1
★	RST	D	DP	Reset a discrete or all bits of a register to 0

■ SFC Instructions

★	STP	Snnn		STEP declaration
★	STPEND			End of the STEP program
★	TO	Snnn		STEP divergent instruction
★	FROM	Snnn		STEP convergent instruction

■ Arithmetical Operation Instructions

★ 11	(+)	Sa,Sb,D	DP	Perform addition of Sa and Sb and then store the result to D
★ 12	(-)	Sa,Sb,D	DP	Perform subtraction of Sa and Sb and then store the result to D
★ 13	(*)	Sa,Sb,D	DP	Perform multiplication of Sa and Sb and then store the result to D
★ 14	(/)	Sa,Sb,D	DP	Perform division of Sa and Sb and then store the result to D
15	(+1)	D	DP	Adds 1 to the D value
16	(-1)	D	DP	Subtracts 1 from the D value
23	DIV48	Sa,Sb,D	P	Perform 48 bits division of Sa and Sb and then store the result to D
24	SUM	S,N,D	DP	Take the sum of the successive N values beginning from S and store it in D
25	MEAN	S,N,D	DP	Take the mean average of the successive N values beginning from S and store it in D
26	SQRT	S,D	DP	Take the square root of the S value and store it in D
27	NEG	D	DP	Take the 2's complement (negative number) of the D value and store it back in D
28	ABS	D	DP	Take the absolute value of D and store it back in D
29	EXT	D	P	Take the 16 bit numerical value and extend it to 1 32 bit numerical value (value will not change)
30	PID	TS,SR,OR, PR,WR		PID operation

■ Logical Operation Instructions

★ 18	AND	Sa,Sb,D	DP	Perform logical AND for Sa and Sb and store the result to D
★ 19	OR	Sa,Sb,D	DP	Perform logical OR for Sa and Sb and store the result to D
35	XOR	Sa,Sb,D	DP	Take the result of the Exclusive OR logical operation made between Sa and Sb, and store it in D
36	XNR	Sa,Sb,D	DP	Take the result of the Exclusive OR logical operation made between Sa and Sb, and store it in D

■ Comparison Instructions

★ 17	CMP	Sa,Sb	DP	Compare the data at Sa and data at Sb and output the result to function outputs (FO)
37	ZNCMP	S,Su,SL	DP	Compare S with the zones formed by the upper limit Su and lower limit SL, and set the result to FO0~FO2

## ■ Data Movement Instructions

FUN No.	Name	Operand	Derivative instruction	Function descriptions
★ 8	MOV	S,D	DP	Transfer the W or DW data specified at S to D
★ 9	MOV/	S,D	DP	Invert the W or DW data specified at S, and then transfers the result to D
40	BITRD	S,N	DP	Read the status of the bits specified by N within S, and send it to F00
41	BITWR	D,N	DP	Write the INB input status into the bits specified by N within D
42	BITMV	S,Ns,D,Nd	DP	Write the status of bit specified by N within S into the bit specified by N within D
43	NBMV	S,Ns,D,Nd	DP	Write the Ns nibble within S to the Nd nibble within D
44	BYMV	S,Ns,D,Nd	DP	Write the byte specified by Ns within S to the byte specified by Nd within D
45	XCHG	Da,Db	DP	Exchange the values of Da and Db
46	SWAP	D	P	Swap the high-byte and low-byte of D
47	UNIT	S,N,D	P	Take the nibble 0 (NB0) of the successive N words starting from S and combine the nibbles sequentially then store in D
48	DIST	S,N,D	P	De-compose the word into successive N nibbles starting from nibble 0 of S, and store them in the NB0 of the successive N words starting from D

## ■ Shifting/Rotating Instructions

★ 6	BSHF	D	DP	Shift left or right 1 bit of D register
51	SHFL	D,N	DP	Shift left the D register N bits and move the last shifted out bits to OTB. The empty bits will be replaced by INB input bit
52	SHFR	D,N	DP	Shift right the D register N bits and move the last shifted out bits to OTB, The empty bits will be replaced by INB input bit
53	ROTL	D,N	DP	Rotate left the D operand N bits and move the last rotated out bits to OTB
54	ROTR	D,N	DP	Rotate right the D operand N bits and move the last rotated out bits to OTB

## ■ Code Conversion Instruction

★ 20	→BCD	S,D	DP	Convert binary data of S into BCD data and store the result to D
★ 21	→BIN	S,D	DP	Convert BCD data of S into binary data and store the result to D
57	DECOD	S,Ns,Nl,D	P	Decode the binary data formed by Nl bits starting from Ns bit within S, and store the result in the register starting from D
58	ENCOD	S,Ns,Nl,D	P	Encoding the Nl bits starting from the Ns bit within S, and store the result in D
59	→7SG	S,N,D	P	Convert the N+1 number of nibble data within S, into 7 segment code, then store in D

FUN No.	Name	Operand	Derivative instruction	Function descriptions
60	→ASC	S,D	P	Write the constant string S (max. 12 alpha-numeric or symbols) into the registers starting from D
61	→SEC	S,D	P	Convert the time data (hours, minutes, seconds) of the three successive registers starting from S into seconds data then store to D
62	→HMS	S,D	P	Convert the seconds data of S into time data (hours, minutes, seconds) and store the data in the three successive registers starting from D
63	→HEX	S,N,D	P	Convert the successive N ASCII data starting from S into hexadecimal data and store them to D
64	→ASC II	S,N,D	P	Convert the successive N hexadecimal data starting from S into ASCII codes and store them to D

#### ■ Flow Control Instructions

★ 0	MC	N		The start of master control loop
★ 1	MCE	N		The end of master control loop
★ 2	SKP	N		The start of skip loop
★ 3	SKPE	N		The end of skip loop
	END			End of Program
65	LBL	1~6 alphanumeric		Define the label with 1~6 alphanumeric string
66	JMP	LBL	P	Jump to LBL label and continues the program execution
67	CALL	LBL	P	Call the sub-program begin with LBL label
68	RTS			Return to the calling main program from sub-program
69	RTI			Return to interrupted main program from sub-program
70	FOR	N		Define the starting point of the FOR Loop and the loop count N
71	NEXT			Define the end of FOR loop

#### ■ Temperature Control Function Instructions

72	TP4	Tp,Pe,Sm,Ym, AR,TR,WR		Temperature module FB-2AJ(K,H,T)4 convenient instruction for measurement
73	TSTC	Tp,Pe,Sm,Ym, AR,TR,Yn,Sn, Zh,Sv,Os,PR, IR,DR,OR,WR		Temperature module FB-2AJ(K,H,T)4 convenient instruction for measurement and PID control
85	TPSNS	Tp,PI,Zn, Yn,SR,WR		Temperature module FB-4AJ(K)XX convenient instruction for measurement
86	TPCTL	Yn,Sn,Zn, Sv,Os,PR IR,DR,OR,WR		Temperature module FB-4AJ(K)XX convenient instruction for PID control

■ I/O Function Instructions

FUN No.	Name	Operand	Derivative instruction	Function descriptions
74	IMDIO	D,N	P	Update the I/O signal on the main unit immediately
75	FILT	N	P	Set the input filter time of high speed input X0~X15 to N mS
76	TKEY	IN,D,KL	D	Convenient instruction for 10 numeric keys input
77	HKEY	IN,OT,D,KL	D	Convenient instruction for 16 keys input
78	DSW	IN,OT,D	D	Convenient instruction for digital switch input
79	7SGDL	S,OT,N	D	Convenient instruction for multiplexing 7-segment display
80	MUXI	IN,OT,N,D		Convenient instruction for multiplexing input instruction
81	PLSO	MD, Fr, PC UY,DY,HO	D	Pulse output function (for bi-directional drive of step motor)
82	PWM	TO,TP,OT		Pulse width modulation output function
83	SPD	S,TI,D		Speed detection function
84	7SGMO	S,Yn,Dn, PT,IT,WS		Convenient instruction for 7-segment display module(FB-7SGXX)

■ Cumulative Timer Function Instructions

87	T.01S	CV,PV		Cumulative timer using 0.01S as the time base
88	T.1S	CV,PV		Cumulative timer using 0.1S as the time base
89	T1S	CV,PV		Cumulative timer using 1S as the time base

■ Watch Dog Timer Control Function Instructions

90	WDT	N	P	Set the WDT timer time out time to N mS
91	RSWDT		P	Reset the WDT timer to 0

■ High Speed Counter Control Function Instructions

92	HSCTR	CN	P	Read the current CV value of the hardware HSCs, HSC0~HSC3, or HST on ASIC to the corresponding CV register in the PLC respectively
93	HSCTW	CN,D	P	Write the CV or PV register of HSC0~HSC3 or HST in the PLC to CV or PV register of the hardware HSC or HST on ASIC respectively

■ Report Function Instructions

94	ASCWR	MD,S,Pt		Parse and generate the report message based on the ASCII formatted data starting from the address S. Then report message will send to port1
----	-------	---------	--	---

■ Ramp Function Instructions

95	RAMP	Tn,PV,SL, SU,D		Ascending/Descending convenient instruction
----	------	-------------------	--	---

■ Communication Function Instructions

96	LINK2	MD,S,Pt		Convenient instruction for port2 communication control
97	LINK1	MD,S,Pt		Convenient instruction for port1 communication control

■ Table Function Instructions

FUN No.	Name	Operand	Derivative instruction	Function descriptions
100	R→T	Rs,Td,L,Pr	DP	Store the Rs value into the location pointed by the Pr in Td
101	T→R	Ts,L,Pr,Rd	DP	Store the value at the location pointed by the Pr in Ts into Rd
102	T→T	Ts,Td,L,Pr	DP	Store the value at the location pointed by the Pr in Ts into the location pointed by the Pr in Td
103	BT_M	Ts,Td,L	DP	Copy the entire contents of Ts to Td
104	T_SWP	Ta,Tb,L	DP	Swap the entire contents of Ta and Tb
105	R-T_S	Rs,Ts,L,Pr	DP	Search the table Ts to find the location with data different or equal to the value of Rs. If found store the position value into the Pr
106	T-T_C	Ta,Tb,L,Pr	DP	Compare two tables Ta and Tb to search the entry with different or same value. If found store the position value into the Pr
107	T_FIL	Rs,Td,L	DP	Fill the table Td with Rs
108	T_SHF	IW,Ts,Td,L,OW	DP	Store the result into Td after shift left or right one entry of table Ts. The shift out data is send to OW and the shift in data is from IW
109	T_ROT	Ts,Td,L	DP	Store the result into Td after shift left or right one entry of table Ts.
110	QUEUE	IW,QU,L,Pr,OW	DP	Push IW into QUEUE or get the data from the QUEUE to OW (FIFO)
111	STACK	IW,ST,L,Pr,OW	DP	Push IW into STACK or get the data from the STACK to OW (LIFO)
112	BKCMP	Rs,Ts,L,D	DP	Compare the Rs value with the upper/lower limits of L, constructed by the table Ts, then store the comparison result of each pair into the relay designated by D (DRUM)
113	SORT	S,D,L	DP	Sorting the registers starting from S length L and store the sorted result to D

■ Matrix Instructions

120	MAND	Ma,Mb,Md,L	P	Store the results of logic AND operation of Ma and Mb into Md
121	MOR	Ma,Mb,Md,L	P	Store the results of logic OR operation of Ma and Mb into Md
122	MXOR	Ma,Mb,Md,L	P	Store the results of logic Exclusive OR operation of Ma and Mb into Md
123	MXNR	Ma,Mb,Md,L	P	Store the results of logic Exclusive OR operation of Ma and Mb into Md
124	MINV	Ms,Md,L	P	Store the results of inverse Ms into Md
125	MCMP	Ma,Mb,L Pr	P	Compare Ma and Mb to find the location with different value, then store the location into Pr
126	MBRD	Ms,L,Pr	P	Read the bit status pointed by the Pr in Ms to the OTB output
127	MBWR	Md,L,Pr	P	Write the INB input status to the bits pointed by the Pr in Ms
128	MBSHF	Ms,Md,L	P	Store the results to Md after shift one bit of the Ms. Shifted out bit will appear at OTB and the shift in bits comes from INB
129	MBROT	Ms,Md,L	P	Store the results to Md after rotate one bit of the Ms. Rotated out bit will appear at OTB.
130	MBCNT	Ms,L,D	P	Calculate the total number of bits that are 0 or 1 in Ms, then store the results into D

■ NC Positioning Instruction

FUN No.	Name	Operand	Derivative instruction	Function descriptions
140	HPSO	Ps,SR,WR		HPSO instruction of NC positioning control
141	MPARA	Ps,SR	P	Parameter setting instruction of NC positioning control
142	PSOFF	Ps	P	Stop the pulse output of NC positioning control
143	PSCNV	Ps,D	P	Convert the Ps positions of NC positioning to mm, Inch or Deg

■ Disable/Enable Control of Interrupt or Peripheral

145	EN	LBL	P	Enable HSC, HST, external INT or peripheral operation
146	DIS	LBL	P	Disable HSC, HST, external INT or peripheral operation

## Chapter 5 Sequential Instructions

The sequential instructions of FB-PLC shown in this chapter are also listed in section 4.1. Please refer to Chapter 1, "PLC Ladder diagram and the Coding rules of Mnemonic instruction", for the coding rules in applying those instructions. In this chapter, we only introduce the applicable operands, ranges and element characteristics, functionality.

### 5.1 Valid Range of the Operand of Sequential Instructions

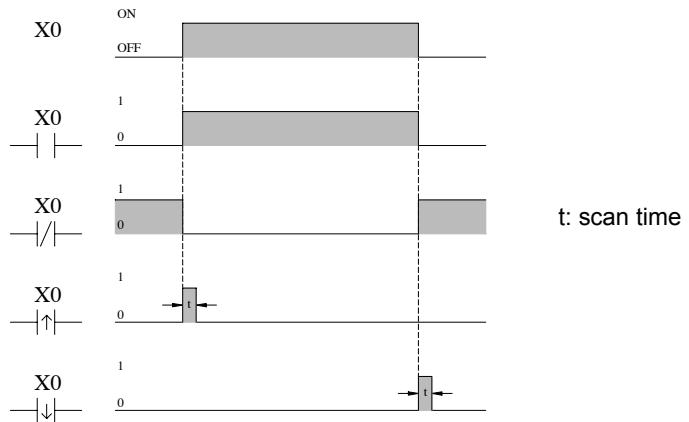
Operand Ranges Instruction	X X0   X255	Y Y0   Y255	M M0   M1911	SM M1912   M2001	S S0   S999	T T0   T255	C C0   C255	TR TR0   TR39	OPEN	SHORT
ORG	○	○	○	○	○	○	○		○	○
ORG NOT	○	○	○	○	○	○	○			
ORG TU	○	○	○	○*	○	○	○			
ORG TD	○	○	○	○*	○	○	○			
LD	○	○	○	○	○	○	○	○	○	○
LD NOT	○	○	○	○	○	○	○			
LD TU	○	○	○	○*	○	○	○			
LD TD	○	○	○	○*	○	○	○			
AND	○	○	○	○	○	○	○		○	○
AND NOT	○	○	○	○	○	○	○			
AND TU	○	○	○	○*	○	○	○			
AND TD	○	○	○	○*	○	○	○			
OR	○	○	○	○	○	○	○		○	○
OR NOT	○	○	○	○	○	○	○			
OR TU	○	○	○	○*	○	○	○			
OR TD	○	○	○	○*	○	○	○			
OUT		○	○	○*	○			○		
OUT NOT		○	○	○*	○					
OUT L		○								
ANDLD						—				
ORLD						—				
TU						—				
TD						—				
NOT						—				

※For the relays marked with a '■' symbol in the special relay table (please refer to section 3.3) is write prohibited. In addition, TU and TD contacts are not supported for those relays as well. The operands marked with a '\*' symbol in the table shown above should exclude those special relays.

## 5.2 Element Description

### 5.2.1 Characteristics of A,B,TU and TD Contacts

- Input X0 from the input terminal block

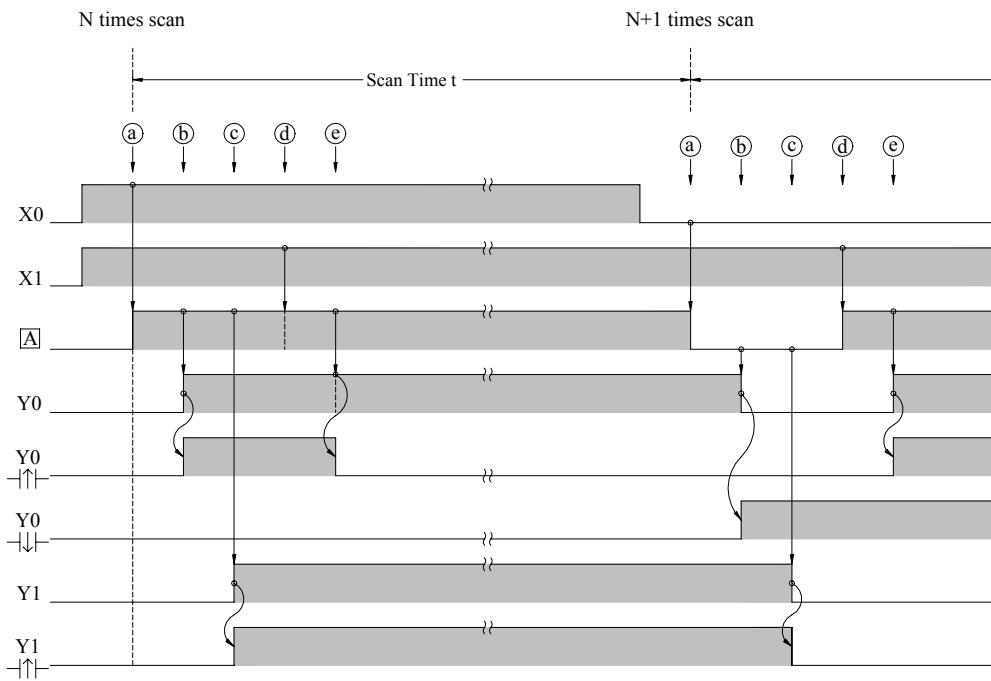


The waveform shown above reveals the function of A, B, TU and TD elements by exercising the external input X0 form OFF to ON then OFF.

- TU (Transition Up): This is the “Transition Up Contact”. Only a rising edge ( $0 \rightarrow 1$ ) of the referenced signal will turn on this element for one scan time.
- TD (Transition Down): This is the “Transition Down Contact”. Only a falling edge ( $1 \rightarrow 0$ ) of the referenced signal will turn on this element for one scan time.
- TU and TD contact will work normally as described above if the change of the status of the valid referenced operands listed in the “Valid Range of the Operand of Sequential instructions” table are not driven by the function instructions.

Remark: For TU(TD) elements which operand is of relay will turn on after the first time the corresponding relay get driven from 0 to 1(1 to 0). When the next time the corresponding relay get driven from 1 to 1(0 to 0) the TD(TU) element will turn OFF. Care should be taken while there is a multiple coil usage situation existed in the ladder program. This situation can be best illustrated at below. In the waveform we can see Y0 TU element only turn on between ④ and ⑤ time which only the Y0 TU elements existed between rung 1 and rung 2 can detect the Y0 rising edge, while other Y0 TU elements out side these two ladder rungs will never aware the occurrence of the rising edge. For the relays do not have the multiple coil usage in ladder program, The ON status of corresponding TU or TD element can be sustained for one scan time, but for relays which contrary to above, the turn on time will shorter than 1 scan time as illustrated at below.

WinProladder Diagram	Mnemonic code
	ORG X 0 ----- a OUT Y 0 ----- b OUT Y 1 ----- c ORG X 1 ----- d OUT Y 0 ----- e

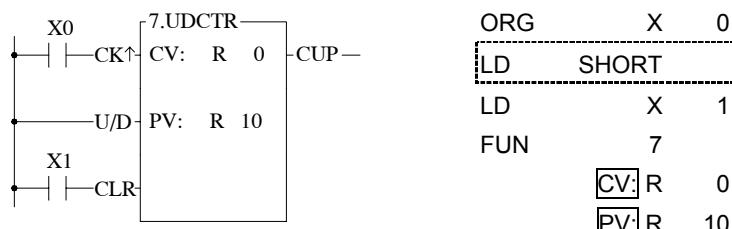


A : The internal accumulator of PLC

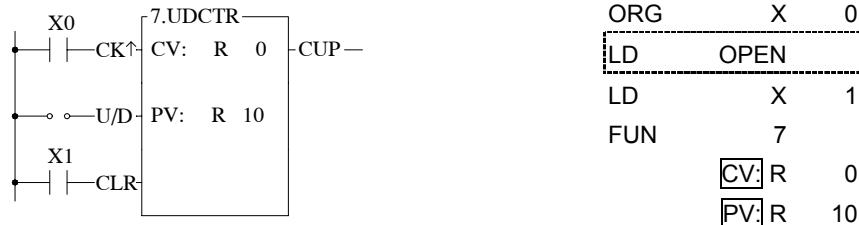
- Besides the TU/TD instructions which can detect the status change of reference operand, FB-PLC also provides the instructions to detect the change of node status (power flow). For details please refer the descriptions of FUN4 (DIFU) and FUN5 (DIFD) instructions at chapter 7.

### 5.2.2 OPEN and SHORT Contact

The status of OPEN and SHORT contact are fixed and can't be changed by any ladder instructions. Those two contacts are mainly used in the places of the Ladder Diagram where fixed contact statuses are required, such as the place where the input of an application instruction is used to select the mode. The sample program shown below gives an example of configuring an Up/Down counter (UDCTR) to an Up counter by using the SHORT contact.

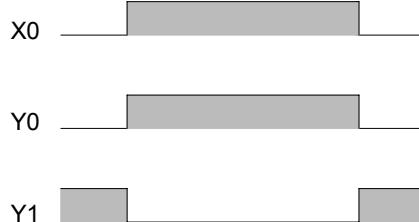


FUN7 is the UDCTR function. While rising edge of CK input occur, FUN7 will count up if the U/D status is 1 or count down if the U/D status is 0. The example shown above, U/D status is fixed at 1 since U/D is directly connected from the origin-line to a SHORT contact, therefore FUN7 becomes an Up counter. On the contrary, if the U/D input of FUN7 is connected with an OPEN contact from the origin-line, the FUN7 becomes a DOWN counter.



### 5.2.3 Output Coil and Inverse Output Coil

Output Coil writes the node status into an operand specified by the coil instruction. Invert Output Coil writes the complement status of node status into an operand specified by the coil instruction. The characteristics depicts at below.



### 5.2.4 Latching Output Coil

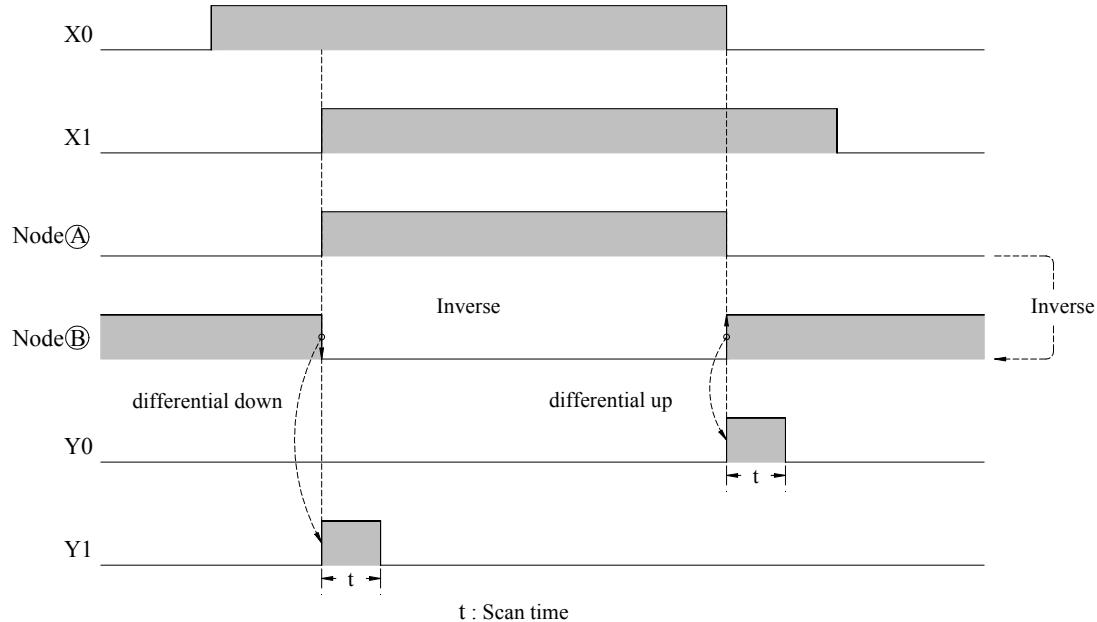
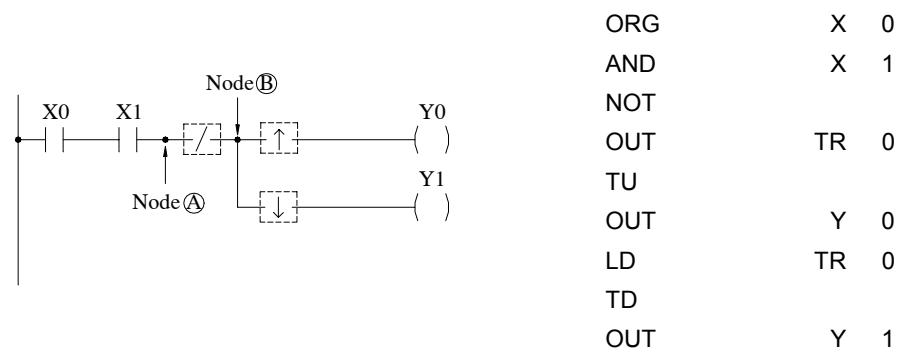
The coil element can be categorized into two types, namely Retentive and Non Retentive. For example, M0~M799 can be specified as the Retentive coils and M800~M1399 can be specified as the Non Retentive coils. One way to categorize the relay type is to divide the relays into groups. Though this method is simple but for the most applications the coils needed to be retentive may be in a random order. FB-PLC allows user to set the retentive status of coil individually. When input the program with mnemonics instructions, if put an "L" after the OUT instruction can declare this specific relay as retentive output. This can be shown in the diagram below.



From the above example, if turn the X0 "ON" then "OFF", Y0 will keep at "ON". When change the PLC state from RUN to STOP then RUN or turn the power off then on, the Y0 still keep at ON state. But if use the OUT Y0 instruction instead of the OUT L Y0 , Y0 status will be OFF.

### 5.3 Node Operation Instructions

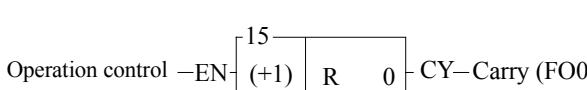
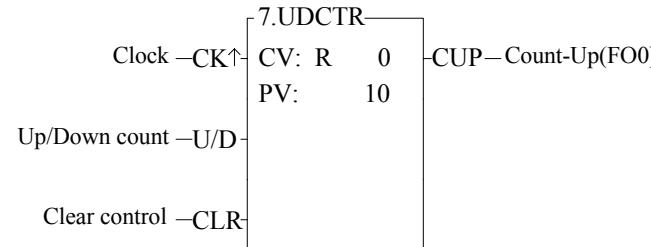
A node is the connection between elements in a ladder diagram consisting of sequential instruction elements (please refer to Section 1.2). There are four instructions dedicated for node status operation in FB-PLC. The two instructions, "OUT TR" and "LD TR", have been discussed in Section 1.6 of this manual. Using the diagram below, the three node operation instructions NOT, TU and TD, are illustrated.



# Chapter 6 Descriptions of Function Instructions

## 6.1 The Format of Function Instructions

In this chapter we will introduce the function instructions of FB-PLC in details. All the explanations for each function will be divided into four parts including input control, instruction number/name, operand and function output. If use the FP-07 to input the mnemonic instruction, except for the T, C, SET, RST and SFC instructions that can be entered directly by pressing a single key stroke on FP-07, other function instructions must be entered by key in the instruction number rather than the instruction name. An example is shown in below.

Ladder Diagram	FP-07 Mnemonic code
<p>Example 1: Single input instruction</p> 	<p>FUN 15 D: R 0</p>
<p>Example 2: Multiple input instruction</p> 	<p>FUN 7 CV: R 0 PV: 10</p>

Remark : The words inside the hollow box in mnemonic code field are the prompting message from FP-07 such as **D:**, **CV:**, and **PV:** and are not entered by the user.

### 6.1.1 Input Control

Except for the seven function instructions that do not have input control, the number of the input control of other FB-PLC function instructions can be ranged from one to four. Execution of the instructions and operations is dependent on the input control signal or the combinations of the several input control signals. The ladder programming software for FACON PLC - Winrollader can help user to complete the complex design and document works. In the ladder program window we can see all the function instructions were displayed by blocks surrounded with abbreviated words for ease of comprehension, include inputs, outs, function name, and parameter names. As shown in example 2 above, the first input mark "CK ↑" indicates when the "CK ↑" input changes from 0 to 1 (rising edge) the counter will be increased or decreased by 1 (depending on the "U/D" status). The second input mark "U/D" with a status of 1 represents the word above slash ("U") and the status 0 represents the word under slash ("D"), that is second input "U/D" states =1, the counter will be increased by 1 when "CK ↑" input from 0 to 1, and when "U/D"=0, the counter will be decreased by 1. The third input mark "CLR" indicates when this input is 1, the counter will be cleared to 0. Chapter 8~9 give the descriptions of input control of each function instruction.

Remark: There are total of seven instructions whose input control should be directly connected to the origin-line those are MCE, SKPE, LBL, RTS, RTI, FOR, and NEXT. Please refer to chapter 7 and 9 for more detailed explanations.

All input controls of the function instructions should be connected by the corresponding elements, otherwise a syntax error will occur. As shown in example 3 below, the function instruction FUN7 has three inputs and three elements before FUN7. ORG X0, LD X1 and LD X2 corresponds to the first input CK↑, second input U/D and third input CLR.

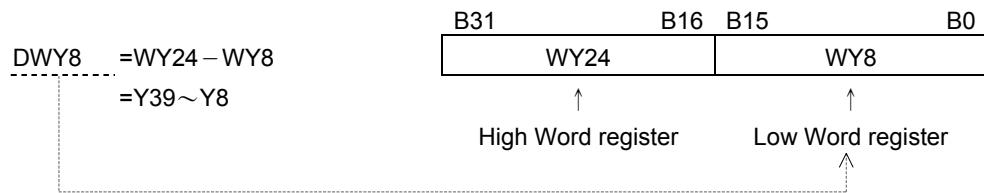
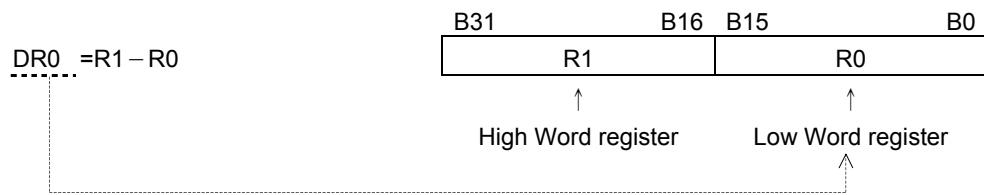
Example 3:

Ladder Diagram	FP-07 Mnemonic code
	<pre> ORG   X 0 LD    X 1 LD    X 2 FUN   7       [CV: R 0]       [PV: 10]   </pre> <p>FUN7 need three elements because it has three inputs</p>

### 6.1.2 Instruction Number and Derivative Instructions

As mentioned before, except for the nine instructions that can be entered using the dedicated keys on the keyboard, other function instructions must be entered using the "instruction number". Follow the instruction number there are postfixes D, P, DP can be added which can derive three additional function instructions.

D: Indicates a Double Word (32-bit). The 16-bit word is the basic unit of the registers in FB-PLC. The data length of R, T and C (except C200~C255) registers are 16-bit. If a register with 32-bit data length is required, then it is necessary to combine two consecutive 16-bit registers together such as R1-R0, R3-R2 etc. and those registers are represented by prefix a D letter before register name such as DR0 represents R1-R0 and DR2 represents R3-R2. If you enter DR0 or DWY8 in the monitor mode of FP-07, then a 32-bit long value (R1-R0 or WY24-WY8) will be displayed.

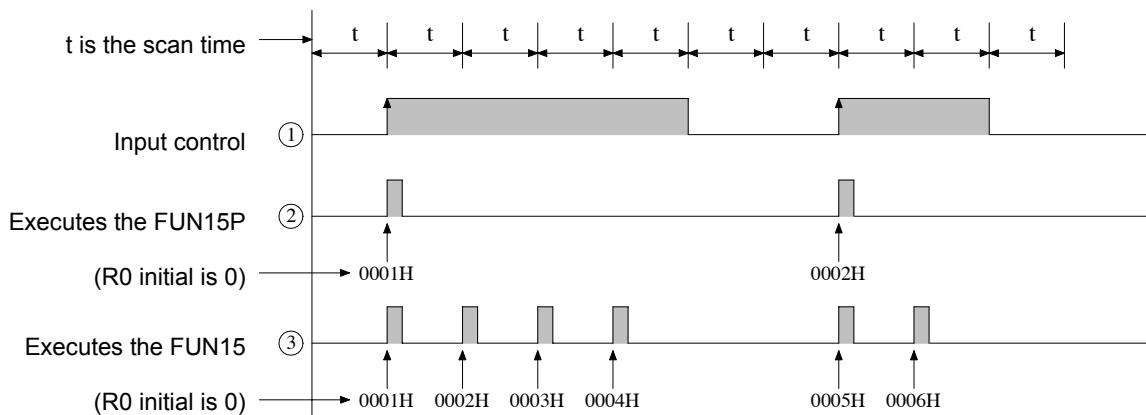


Remark: In order to differentiate between 16-bit and 32-bit instructions while using the ladder diagram and mnemonic code, we add the postfix letter D after the "Instruction number" to represent 32-bit instructions and the size of their operand are 32-bit as shown in example 4 on P.6-6. The instruction FUN 11D has a postfix letter D, therefore the source and destination operands need to prefix a letter D as well, such as the augend Sa : R0 is actually Sa=DR0=R1-R0 and Sb=DR2=R3-R2. Please also pay special attention to the length of the other operands except source and destination are only one word whether 16-bit or 32-bit instructions are used.

P: indicates the pulse mode instruction. The instruction will be executed when the status of input control changes from 0 to 1 (rising edge). As shown in example 1, if a postfix letter P is added to the instruction (FUN 15P), the instruction FUN 15P will only be executed when the status of input control signal changes from 0 to 1. The execution of the instruction is in level mode if it does not have a P postfix, this means the instruction will be executed for every scan until the status of input control changes from 1 to 0. The pulse input is indicated by a symbol " $\uparrow$ ", such as CK  $\uparrow$ , EN  $\uparrow$ , TG  $\uparrow$  etc.. In this operation manual, an example of the operation statement of a function instruction is shown below.

- When the operation control "EN" =1 or "EN  $\uparrow$ " (P instruction) from 0→1, .....

The first one indicates the execution requirement for non-P instruction (level mode) and the second one indicates the execution requirement for P instruction (pulse mode). The following waveform shows the result (R0) of FUN15 and FUN15P under the same input condition.



DP: Indicates the instruction is a 32-bit instruction operating with pulse mode.

Remark: P instruction is much more time saving than level instruction in program scanning, So user should use P instruction as much as possible.

### 6.1.3 Operand

The operand is used for data reference and storage. The data of source (S) operand are only for reference and will not be changed with the execution of the instruction. The destination (D) operand is used to store the result of operation and its data may be changed after the execution of the instruction. The following table illustrates the names and functions of FACON PLC function instruction's operands and types of contacts, coils, or registers that can be used as an operand.

- The names and functions of the major operands:

Abbreviation	Name	Descriptions
S	Source	The data of source (S) operand are only for reading and reference and will not be changed with the execution of the instruction. If there are more than one source operands, each operand will be identified by the footnote such as Sa and Sb.
D	Destination	The destination (D) operand is used to store the result of operation. The original data will be changed after operation. Only the coils and registers which are not write prohibited can be the destination operand.
L	Length	Indicates the data size or the length of the table, usually are constants.
N	Number	A constant most often used as numbers and times. If there are more than one constant, each constant will be identified by the footnotes such as Na, Nb, Ns etc..
Pr	Pointer	Used to point to a specific a block of data or a specific data or register in a table. Generally the Pr value can be varied, therefore cannot be constant or input register. (R3840~R3847)
CV	Current value	Used in T and C instruction to store the current value of T or C
PV	Set value	Used in T and C instructions for reference and comparison
T	Table	A combination of a set of consecutive registers forms a table. The basic operation units are word and double word. If there is more than one table, each table will be identified by footnotes such as Ta, Tb, Ts and Td etc..
M	Matrix	A combination of a set of consecutive registers forms a matrix. The basic operation unit is bit. If there is more than one matrix, each matrix will be identified by footnotes such as Ma, Mb, Ms and Md etc..

Besides the major operands mentioned above, there are other operands which are used for certain special purposes such as the operand Fr for frequency, ST for stack, QU for Queue etc.. Please refer to the instruction descriptions for more details.

- The types of the operand and their range: The types of operand for the function instructions are discrete, register and constant.

a) Discrete operand :

There are total five function instructions reference the discrete operand, namely SET, RST, DIFU, DIFD and TOGG. Those five instructions can only be used for operations of YΔΔΔ (external output), MΔΔΔΔ (internal and special) and SΔΔΔ (step) relays. The table shown below indicates the operands and ranges of the five function instructions.

Range	Y	M	SM	S
Ope- rand	Y0   Y255	M0   M1911	M1912   M2001	S0   S999
D	○	○	○*	○

Symbol "O" indicates the D (Destination operand) can use this type of coils as operands. The "\*" sign above the "O" shown in SM column indicates that should exclude the write prohibited relays as operands. Please refer to page 3-3 for introduction of the special relays.

b) Register operand :

The major operand for function instructions is register operand. There are two types of register operands: the native registers which already is of Words or Double Words data such as R, D, T, C. The other is derivative registers (WX, WY, WM, WS) which are formed by discrete bits. The types of registers that can be used as instruction operands and their ranges are all listed in the following table:

Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V Z
S	○	○	○	○	○	○	○	○	○	○*	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○
⋮														

The "○" symbol in the table indicates can apply this kind of data as operand. The "○\*" symbol indicates can apply this kind of data except the write prohibited registers as operand. To learn more about write prohibited registers please refer to page 3-6 for introduction of the special register.

When R5000~R8071 are not set to be read only registers, can used as normal registers (read, and write)

Remark 1: The registers with a prefix W, such as WX, WY, WM and WS are formed by 16 bits. For example, WX0 means the register is formed by X0(bit 0)~X15(bit 15). WY144 means the register is formed by Y144(bit 0)~Y159(bit 15). Please note that the discrete number must be the multiple of 8 such as 0, 8, 16, 24....

Remark 2: The last register (Word) in a table can not be represented as a 32-bit operand in the function because 2 Words are required for a 32-bit operand.

Remark 3: TMR ( T0~T255 ) and CTR ( C0~C255 ) are the registers of timers and counters respectively. Although they can be used as general registers, they also complicate the systems and make debugging more difficult. Therefore you should avoid writing anything into the TMR or CTR registers.

Remark 4: T0~T255 and C0~C199 are 16-bit register. But C200~C255 are 32-bit register, therefore can't be used as 16-bit operands.

Remark 5: Apart from being directly appointed by register's number (address) as the foregoing discussions, the register's operand in the range of R0~R8071 can be combined with pointer register V or Z to make indirect addressing. Please refer to the example in the next section (Section 6.2) for the description of using pointer register (XR) to make indirect addressing.

### c) Constant operands :

The range of 16-bit constant is between -32768~32767. The range of 32-bit constant is between -2147483648~2147483647. The constant for several function instructions can only be a positive constant. The range of 16-bit and 32-bit constants are listed in the table shown below.

Classification	Range
16-bit signed number	-32768~32767
16-bit un-signed number	0~32767
32-bit signed number	-2147483648~2147483647
32-bit un-signed number	0~2147483647
16/32-bit signed number	-32768~32767 or -2147483648~2147483647
16/32-bit un-signed number	0~32767 or 0~2147483647

It is possible that the length and size of a specific operand, such as L, bit size, N etc., are different, and the differences are all directly marked at the operand column. Please refer to the explanations of function instructions.

#### 6.1.4 Functions Output (FO)

The “Function Output” (FO) is used to indicate the operation result of the function instruction. Like control input, each function outputs shown in the screen of programming software are all attached with a word which comes from the abbreviation of the output functionality. Such as CY derived from CarrY. The maximum number of function outputs is 4 and those are denoted as FO0, FO1, FO2, FO3 respectively. The FO status must be taken out by FO instruction (there is a FO special key on FP-07 program writing device). The unused FO may be left without connecting to any elements, such as FO1 (CY) shown in Example 4 below.

Example 4 :

Ladder Diagram	Mnemonic Codes
	<pre> ORG      X  0 FUN      11D Sa:     R  0 Sb:     R  2 D :    R  4 FO       0 OUT     Y  0 FO       2 OUT     Y  1 </pre>

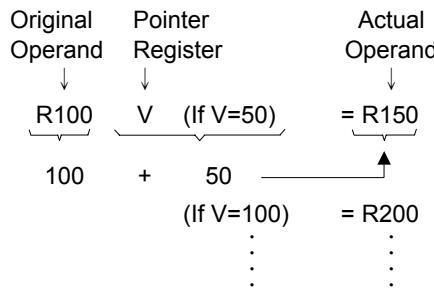
When M1919=0, the FO status will only be updated if the instruction is executed. It will keep the same status until a new FO status is generated after the instruction is executed again (memory keeping).

When M1919=1, the FO status will be reset to 0 (no memory keeping) if the instruction is not executed.

#### 6.2 Use Index Register(XR) for Indirect Addressing

In the FB-PLC function instructions, there are some operands that can be combined with pointer register (V or Z) to make indirect addressing (will be shown in the operand table if it applicable). However, only the registers in the range R0~R8071 can be combined with an pointer register to perform indirect addressing (other operands such as discrete, constant and D0~D3071 cannot be used for indirect addressing).

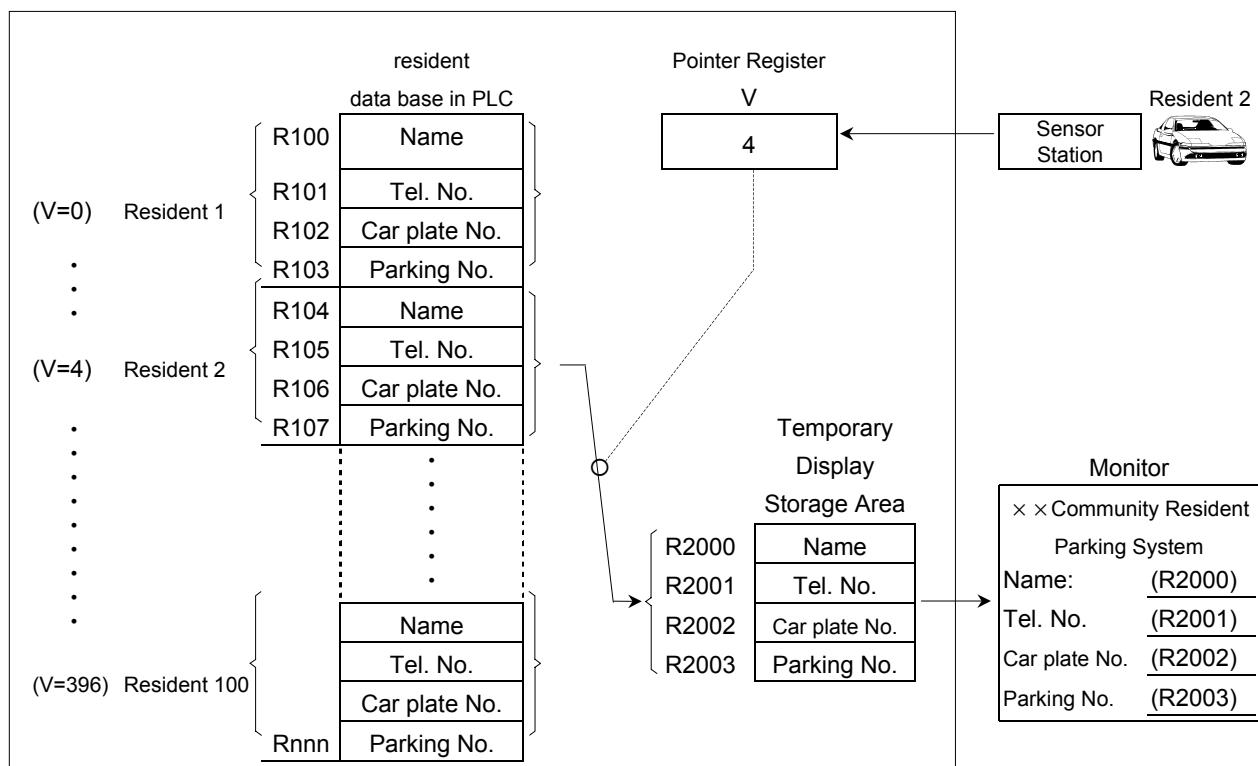
There are two pointer registers XR (V and Z). The V register in fact is the R4164 of special registers (R3840~R4167) and the Z register is the R4165. The actual addressed register by index addressing is just offset the original operand with the content of the index register.



As shown in the above diagram, you only need to change the V value to change the operand address. After combining the index addressing with the FB-PLC function instructions, a powerful and highly efficient control application can be achieved by using very simple instructions. Using the program shown in the diagram below as an example, you only need to use a block move instruction (BT\_M) to achieve a dynamic block data display, such as a parking management system.

**Indirect addressing program example**

Ladder Diagram	Mnemonic Codes
	ORG SHORT FUN 103 [Ts :] R100V [Td :] R2000 [L :] 4



**Description** Suppose that there are 100 resident parking spaces available in a parking management system for community residents. Each resident has a set of basic information including name, telephone number, number plate and parking number, that occupy four consecutive PLC registers as shown in the above diagram. A total of 400 registers (R100~R499) are occupied. Each resident is given a card with a unique card number (the number is 0 for resident 1, 4 for resident 2 etc.. ) for the sensing pass of the main entrance and parking lot. The card number will be sensed by the PLC and stored into the pointer register "V". The attendant's monitor (LCD or CRT) will only display the data grasped by R2001~R2003 in the PLC. For example, the card of residence 2 with the card number 4 is sensed, then the register V=4 and the PLC will immediately move the data in R104~R107 to the temporary display storage area (R2000~R2003). Hence, the attendant's monitor can display the data of residence 2 as soon as its card is sensed.

## Warning

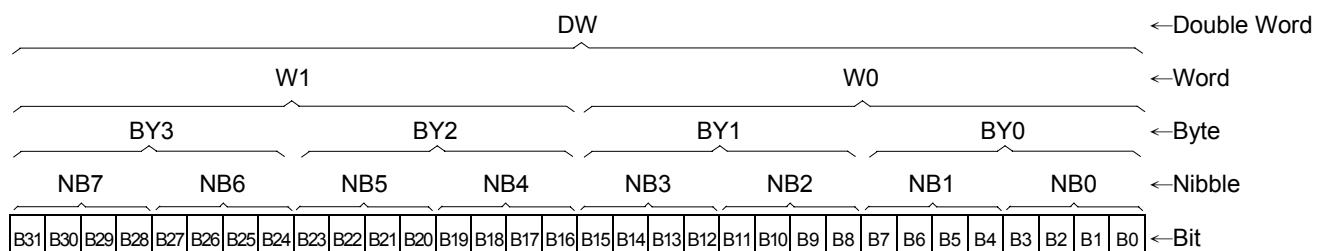
1. Although using pointer register for indirect addressing application is powerful and flexible, but changing the V and Z values freely and carelessly may cause great damages with erroneous writing to the normal data areas. The user should take special caution during operation.
2. In the data register range that can be used for indirect addressing application (R0~R8071), the 328 registers R3840~R4167 (i.e. IR, OR and SR) are important registers reserved for system or I/O usage. Writing at-will to these registers may cause system or I/O errors and may result in a major disaster. Due to the fact that users may not easily detect or control the flexible register address changes made by the V and Z values, FB-PLC will automatically check if the destination address is in the R3840~R4067 range. If it is, the write operation will not be executed and the M1969 flag “Illegal write of Indirect addressing” will be set as 1. In case it is necessary to write to the registers R3840~R4067, please use the direct addressing.

## 6.3 Numbering System

### 6.3.1 Binary Code and Relative Terminologies

Binary is the basic numbering system of digital computer. Since the PLC operates with discrete ON/OFF values, it is natural to use binary codes. The following terminologies should be fully understood before go to further topic of numbering system.

- Bit: (Abbreviated as B, such as B0, B1, and so on) It is the most basic unit of binary value. The status of bit is either “1” or “0”.
- Nibble: (Abbreviated as NB, such as NB0, NB1, and so on) It is formed by four consecutive bits (e.g. B3~B0) and can be used to represent a decimal number 0~9 or a hexadecimal number 0~F.
- Byte: (Abbreviated as BY, such as BY0, BY1, and so on) It is formed by two consecutive nibbles (or 8 bits, such as B7~B0) and can be used to represent a 2-digit hexadecimal number 00~FF.
- Word: (Abbreviated as W, such as W0, W1, and so on) It is formed by two consecutive bytes (or 16 bits, such as B15~B0) and can be used to represent a 4-digit hexadecimal number 0000~FFFF.
- Double Word: (Abbreviated as DW, such as DW0, DW1, and so on) It is formed by two consecutive words (or 32 bits, such as B31~B0) and can be used to represent an 8-digit hexadecimal number 00000000~FFFFFFF.



### 6.3.2 The Coding of Numeric Numbers for FB-PLC

FB-PLC use the binary numbering system for its internal operations that is the data of external BCD inputs must be converted to binary number before the PLC can process. As we know the binary code is very difficult to read and input to the PLC for human, therefore FP-07 and WinProladder use the decimal unit or hexadecimal unit to input or to display the data. But in reality, all the operations taking place in the PLC are performed with binary code.

Remark: If you input or display the data without going through the FP-07 or WinProladder (For instance, input data into or take out data from PLC through the I/O terminals using thumb wheel switch or seven segment display), then you have to use the Ladder program to perform the Decimal to Binary conversion. This enables you to input and display data without using the FP-07 and WinProladder. Please refer to FUN20(BIN→BCD) and FUN21(BCD→BIN).

### 6.3.3 Range of Numeric Value

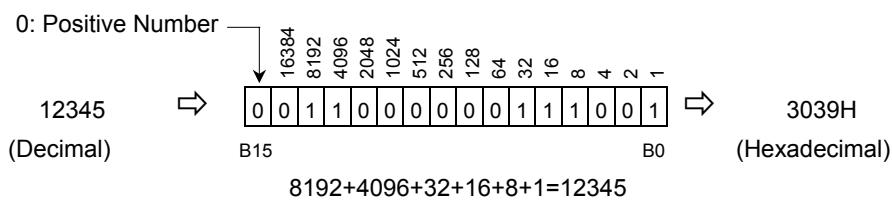
As we have mentioned before that FB-PLC uses binary numbers for its internal operations. 16-bit and 32-bit are two different numeric data of FB-PLC. The ranges of the two numeric values are shown below.

16-bit	-32768~32767
32-bit	-2147483648~2147483647

### 6.3.4 Representation of Numeric Value (Beginners can skip this section)

The representation and specification of 16-bit and 32-bit numeric values are provided below to enable the user to further understand the numeric value operation for more complicated applications.

The most significant bits MSB of 16-bits and 32-bits (B15 for 16-bit and B31 for 32-bit) are used to identify positive and negative numbers (0: positive and 1: negative). The remaining bits (B14~B0 or B30~B0) represent the magnitude of the number. The following example uses 16-bit for further explanations. Please note that everything also applies to 32-bit numbers and the only difference is the length.



In the above example, regardless of its size (16-bit or 32-bit), and starting with the least significant bit LSB (B0). B0 is 1, B1 is 2, B2 is 4, B3 is 8, and so on. The number represented by the neighboring left bit will double its value (1, 2, 4, 8, 16, and so on) and the value is the sum of the numbers represented by the bits that are equal to 1.

### 6.3.5 Representation of Negative Number

(Beginners should skip this section)

As prior discussion, when the MSB is 1, the number will be a negative number. The FB-PLC negative numbers are represented by 2'S Complement, i.e. to invert all the bits (B15~B0 or B31~B0) of its equivalent positive number (The so-called 1'S Complement is to change the bits equal 1 to 0 and the bits equal 0 to 1) then add 1. In the above example, the positive number is 12345. The calculation of its 2'S Complement (i.e. -12345) is described below:

12345	⇒	<table border="1"><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	⇒	3039H
0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1					
1'S Complement of 12345	⇒	<table border="1"><tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	0	⇒	CFC6H
1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	0					
2'S Complement of 12345 (-12345)	+	1																		

### 6.4 Overflow and Underflow of Increment (+1) or Decrement (-1)

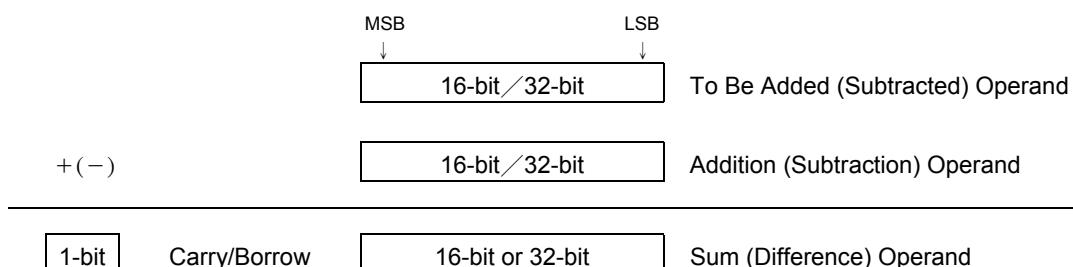
(Beginners should skip this section)

The maximum positive value that can be represented by 16-bit and 32-bit operands are 32767 and 2147483647, respectively. While the minimum negative values that can be represented by 16-bit and 32-bit operands are -32768 and -2147483648, respectively. When increase or decrease an operand (e.g. when Up/Down Count of a counter or the register value is +1 or -1), and the result exceeds the value of the positive limit of the operand, then "Overflow" (OVF) occurs. This will cause the value to cycle to its negative limit (e.g. add 1 to the 16-bit positive limit 32767 will change it to -32768). If the result is smaller than the negative limit of the operand, then "Underflow" (UDF) occurs. This will cause the value to cycle to its positive limit (e.g. deducting 1 from the negative limit -32768 will change it to 32767) as shown in the table below. The flag output of overflow or underflow exists in the FO of FB-PLC and can be used in cascaded instructions to obtain over 16-bit or 32-bit operation results.

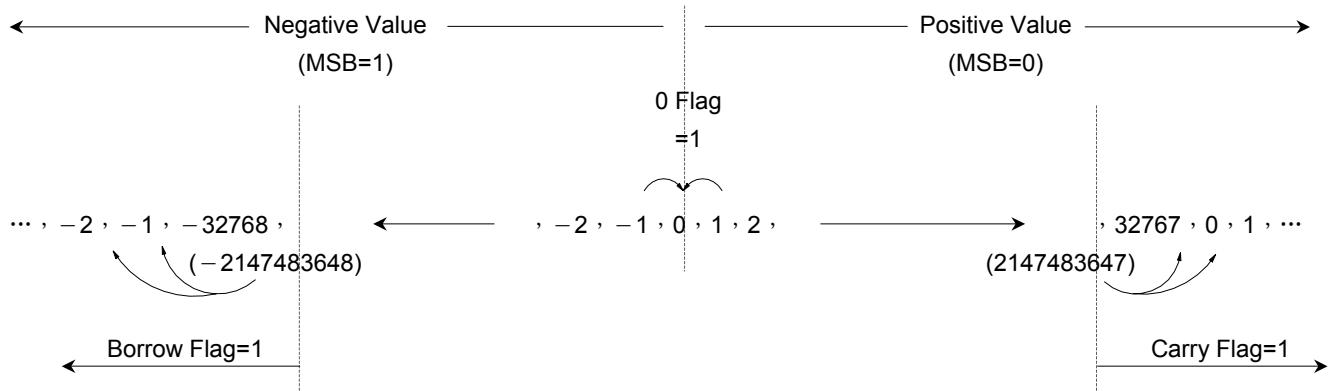
Overflow/ Underflow	Increase (Decrease) Result	16-bit Operand	32-bit Operand
Increase		OVF=1 	OVF=1 
Decrease		UDF=1 	UDF=1 

## 6.5 Carry and Borrow in Addition/Subtraction

Overflow/Underflow takes place when the operation of increment/decrement causes the value of the operand to exceed the positive/negative limit that can be represented in the PLC, consequently a flag of overflow/underflow is introduced. Carry/Borrow flag is different from overflow/underflow. At first, there must be two operands making addition (subtraction) where a sum (difference) and a flag of carry/borrow will be obtained. Since the number of bits of the numbers to be added (subtracted), to add (subtract) and of sum (difference) are the same (either 16-bit or 32-bit), the result of addition (subtraction) may cause the value of sum (difference) to exceed 16-bit or 32-bit. Therefore, it is necessary to use carry/borrow flag to be in coordination with the sum (difference) operand to represent the actual value. The carry flag is set when the addition (subtraction) result exceeds the positive limit (32767 or 2147483647) of the sum (difference) operand. The borrow flag is set when addition (subtraction) result exceeds the negative limit (-32768 or -2147483648) of the sum (difference) operand. Hence, the actual result after addition (subtraction) is equal to the carry/borrow plus the value of the sum (difference) operand. The FO of FB-PLC addition/subtraction instruction has both carry and borrow flag outputs for obtaining the actual result.



While all FB-PLC numerical operations use 2'S Complement, the representation of the negative value of the sum (difference) obtained from addition (subtraction) is different from the usual negative number representation. When the operation result is a negative value, 0 can never appear in the MSB of the sum (difference) operand. The carry flag represents the positive value 32768 (2147483648) and the borrow flag represents the negative value - 32768 (-2147483648).

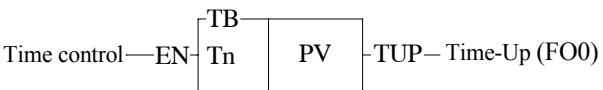


	MSB	LSB	Positive Value
C=1 B=0 Z=0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	32769
C=1 B=0 Z=0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	32768
C=0 B=0 Z=0	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	32767
C=0 B=0 Z=0	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	32766
C=0 B=0 Z=0	0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	32765
...	...	...	...
C=0 B=0 Z=0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	2
C=0 B=0 Z=0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	1
C=0 B=0 Z=1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
C=0 B=0 Z=0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	-1
C=0 B=0 Z=0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	-2
...	...	...	...
C=0 B=0 Z=0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	-32766
C=0 B=0 Z=0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	-32767
C=0 B=0 Z=0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	-32768
C=0 B=1 Z=0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	-32769
C=0 B=1 Z=0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	-32770
...	...	...	...
	↓	↓	↓
C = Carry	B = Borrow	Z = Zero	

## Chapter 7 Basic Function Instruction

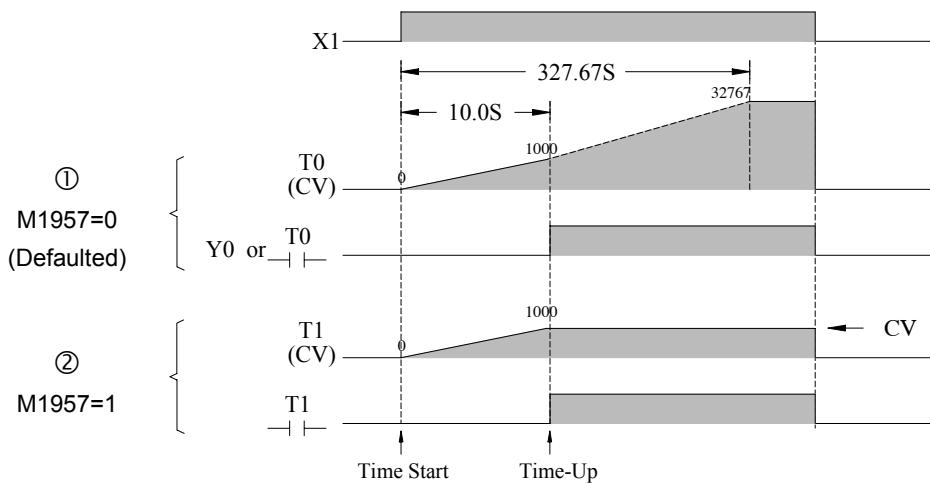
T .....	7- 2
C .....	7- 5
SET .....	7- 8
RST .....	7-10
0 : MC .....	7-12
1 : MCE .....	7-14
2 : SKP .....	7-15
3 : SKPE .....	7-17
4 : DIFU .....	7-18
5 : DIFD .....	7-19
6 : BSHF .....	7-20
7 : UDCTR .....	7-21
8 : MOV .....	7-23
9 : MOV／ .....	7-24
10 : TOGG .....	7-25
11 : (+) .....	7-26
12 : (−) .....	7-27
13 : (*) .....	7-28
14 : (／) .....	7-30
15 : (+1) .....	7-32
16 : (−1) .....	7-33
17 : CMP .....	7-34
18 : AND .....	7-35
19 : OR .....	7-36
20 : →BCD .....	7-37
21 : →BIN .....	7-38

## Basic Function Instruction

T	TIMER												T													
Symbol	<u>Ladder symbol</u>												<u>Operand</u>													
																										
<p>Tn: Timer Number. PV: Preset value of the timer.</p>																										
<p>TB: Time Base (0.01S, 0.1S, 1S)</p>																										
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K													
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	0   32767													
Tn					○																					
PV	○	○	○	○	○	○	○	○	○	○	○	○	○													
<ul style="list-style-type: none"> <li>The total number of timers is 256 (T0~T255) with three different time bases, 0.01S, 0.1S and 1S. The default number and allocation of timers is shown as below (Can be adjusted according to user's actual requirements by the "Configuration" function):           <ul style="list-style-type: none"> <li>T0~T49 : 0.01S timer (default as 0.00~327.67S) .</li> <li>T50~T199 : 0.1S timer (default as 0.0~3276.7S) .</li> <li>T200~T255 : 1S timer (default as 0~32767S) .</li> </ul> </li> <li>FB-PLC programming tool will lookup the timer's time base automatically according to the "Memory Configuration" after the timer number is keyed in. Timer's time = Time base x Preset value. In the example 1 below, the time base T0 = 0.01S and the PV value = 1000, therefore the T0 timer's time = 0.01S x 1000 = 10.00S.</li> <li>If PV is a register, then Timer's time = Time base x register content. Therefore, you only need to change the register content to change the timer's time. Please refer to Example 2.</li> <li>* The maximum error of a timer is a time base plus a scan time. In order to reduce the timing error in the application, please use the timer with a smaller time base.</li> </ul>																										
<table border="1"> <tr> <td>Description</td> <td colspan="13"></td></tr> </table> <ul style="list-style-type: none"> <li>When the time control "EN" is 1, the timer will start timing (the current value will accumulate from 0) until "Time Up" (i.e. CV<math>\geq</math>PV), then the Tn contact and TUP (FO0) will change to 1. As long as the timer control "EN" input is kept as 1, even the CV of Tn has reached or exceeded the PV, the CV of the timer will continue accumulating (with M1957 = 0) until it reaches the maximum limit (32767). The Tn contact status and flag will remain as 1 when CV<math>\geq</math>PV, unless the "EN" input is 0. When "EN" input is 0, the CV of Tn will be reset to 0 immediately and the Tn contact and "Time Up" flag TUP will also change to 0 (please refer to the diagram ① below).</li> <li>If the FB<sub>E</sub>/FB<sub>N</sub>-PLC OS version is higher than V3.0 (inclusive), the M1957 can be set to 1 so the CV will not accumulate further after "Time Up" and stops at the PV value. The default value of the M1957 is 0, therefore the status of M1957 can be set before executing any timer instruction in the program to individually set the timer CV to continue accumulating or stop at the PV after "Time Up" (please refer to the diagram ② below).</li> </ul>													Description													
Description																										

T	TIMER	T
Example 1	Constant preset value	

Ladder diagram	Key operations	Mnemonic code
		ORG X 1 T0 PV: 1000 F0 NOT OUT Y 0 ORG SHORT SET M 1957 ORG X 1 T1 PV: 1000



Example 2	Variable PV
<p>The preset value (PV) shown in example 1 is a constant which is equal to 1000. This value is fixed and can not be changed once programmed. In many circumstances, the preset time of the timers needs to be varied while PLC running. In order to change the preset time of a timer, can first use a register as the PV operand (R or WX, WY...) and then the preset time can be varied by changing the register content. As shown in this example, if set R0 to 100, then T becomes a 10S Timer, and hence if set R0 to 200, then T becomes a 20S Timer.</p>	

## Basic Function Instruction

T	TIMER	T
Ladder diagram	Key operations	Mnemonic code
<p>An example of applying the "time-up" status by using the T50 contact.</p>		ORG X 1 T 50 PV: R 0 ORG T 50 OUT Y 0

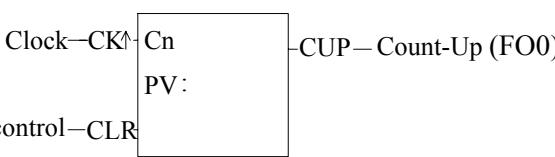
  

Timing diagram illustrating the timer's behavior:

- ① When  $R0 = 100 \Rightarrow Y0$  is high for 10.0S.
- ② When  $R0 = 200 \Rightarrow Y0$  is high for 20.0S.

**Remark:** If the preset value of the timer is equal to 0, then the timer's contact status and FO0 (TUP) become 1 ("EN" input must be at 1) immediately after the PLC finishes its first scan because "Time-Up" has occurred. (TUP) stays at 1 until "EN" input changes to 0.

C	COUNTER ( 16-Bit: C0~C199 · 32-Bit: C200~C255 )												C	
Symbol	<u>Ladder symbol</u>												<u>Operand</u>	
													Cn: The Counter number PV: Preset value	
	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
Oper- and		WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	0   2147483647
Cn							○							
PV		○	○	○	○	○	○	○	○	○	○	○	○	

- There are total 200 16-Bit counters (C0~C199). The range of preset value is between 0~32767. C0~C139 are Retentive Counters and the CV value will be retained when the PLC turns on or RUN again after a power failure or a PLC STOP. For Non Retentive Counters, if a power failure or PLC STOP occurs, the CV value will be reset to 0 when the PLC turns on or RUN again.
- There are total 56 32-Bit counters (C200~C255). The range of the preset value is between 0~2147483647. C200~C239 are Retentive Counters and C240~C255 are Non Retentive Counters.
- The default number and assignment of the counters are shown below, if necessary can use the "CONFIGURATION" function to change the settings.
- To insure the proper counting, the sustain time of input status of CLK should greater than 1 scan time.
- The max. counting frequency with this instruction can only up to 20Hz, for higher frequency please use the high-speed soft/hardware counter.

Description	
<ul style="list-style-type: none"> <li>When "CLR" is at 1, all of the contact Cn, FO0 (CUP), and CV value of the counter CV are cleared to 0 and the counter stops counting.</li> <li>When "CLR" is at 0, the counter is allowed to count up. The Counter counts up every time the clock "CK ↑" changes from 0 to 1 (adds 1 to the CV) until the cumulative current value is equal to or greater than the preset value (CV&gt;=PV), the counter "Count-Up" and the contact status of the counter Cn and FO0 (CUP) changes to 1. If the input status of clock continues to change, even the cumulative current value is equal and greater than the preset value, the CV value will still accumulate until it reaches the up limit at 32767 or 2147483647. The contact Cn and FO0 (CUP) stay at 1 as long as CV&gt;=PV unless the "CLR" input is set to 1. (please refer the diagram ① below) .</li> <li>If the FB<sub>E</sub>/FB<sub>N</sub>-PLC OS version is higher than V3.0 (inclusive), the M1973 can set to 1 so the CV will not accumulate further after "Count Up" and stops at the PV. M1973 default value is 0, therefore the status of M1973 can be set before executing any counter instruction in the program to individually set the counter CV to continue accumulating or stops at the PV after "Count Up" (please refer to the diagram ② below).</li> </ul>	

## Basic Function Instruction

C	COUNTER (16-Bit: C0~C199, 32-bit: C200~C255)	C	
Example 1	16-Bit Fixed Counter		
Ladder diagram		Key operations	Mnemonic code
<p>An example of applying the "Count-Up" status by using F00 directly.</p>			ORG SHORT RST M 1973 ORG X 0 LD X 1 C 1 [PV:] 5 FO 0 OUT Y 1 ORG SHORT SET M 1973 ORG X 0 LD X 1 C 2 [PV:] 5
			① M1973=0 (Defaulted) ② M1973=1
Example 2	32-Bit counter with variable preset value		
			<p>Like a timer, if the PV of a counter is changed to a register (such as R, D, and so on), the counter will use the register contents as the counting PV. Therefore, only need to change the register contents to change the PV of the counter while PLC is running. Below is an example of a 32-bit counter that uses the data register R0 as the PV (in fact it is the 32-bit PV formed by R1 and R0).</p>

C	COUNTER ( 16-Bit: C0~C199, 32-Bit: C200~C255 )	C
Ladder diagram	Key operations	Mnemonic code
<p>An example of applying the "time-up" status by using the C200 contact.</p>		ORG X 0 LD X 1 C200 [PV:] R 0 ORG C 200 OUT Y 1

① When R0=4  $\Rightarrow$  Y1      4 times

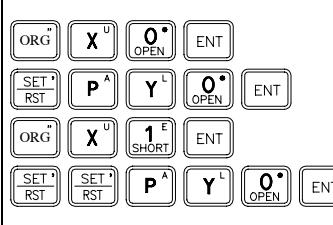
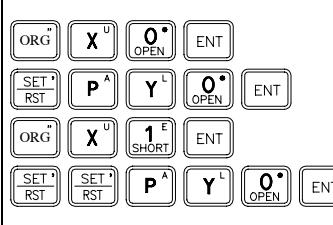
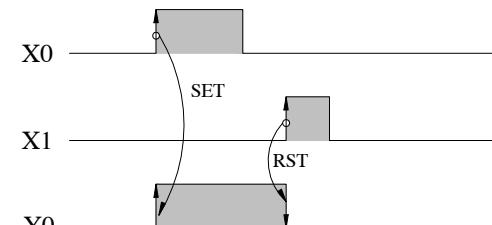
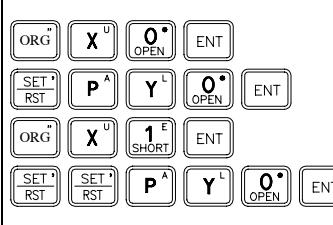
② When R0=9  $\Rightarrow$  Y1      9 times

Count Start      Count-Up      Count-Up

**Remark:** If the preset value of the counter is 0 and "CLR" input also at 0, then the Cn contact status and F00 (CUP) becomes 1 immediately after the PLC finishes its first scan because the "Count-Up" has occurred. It will stay at 1 regardless how the CV value varies until "CLR" input changes to 1.

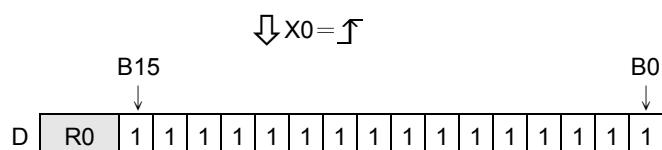
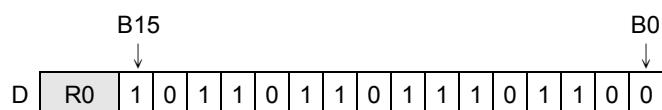
## Basic Function Instruction

SET <b>DP</b>	SET (Set coil or all the bits of register to 1)												SET <b>DP</b>																																													
Symbol																																																										
	<u>Ladder symbol</u>												<u>Operand</u>																																													
													D: destination to be set (the number of a coil or a register)																																													
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Range</th><th style="text-align: center;">Y</th><th style="text-align: center;">M</th><th style="text-align: center;">SM</th><th style="text-align: center;">S</th><th style="text-align: center;">WY</th><th style="text-align: center;">WM</th><th style="text-align: center;">WS</th><th style="text-align: center;">TMR</th><th style="text-align: center;">CTR</th><th style="text-align: center;">HR</th><th style="text-align: center;">OR</th><th style="text-align: center;">SR</th><th style="text-align: center;">ROR</th><th style="text-align: center;">DR</th></tr> </thead> <tbody> <tr> <td style="text-align: center;">Oper- and</td><td style="text-align: center;">Y0   Y255</td><td style="text-align: center;">M0   M1911</td><td style="text-align: center;">M1912   M2001</td><td style="text-align: center;">S0   S999</td><td style="text-align: center;">WY0   WY240</td><td style="text-align: center;">WM0   WM1896</td><td style="text-align: center;">WS0   WS984</td><td style="text-align: center;">T0   T255</td><td style="text-align: center;">C0   C255</td><td style="text-align: center;">R0   R3839</td><td style="text-align: center;">R3904   R3967</td><td style="text-align: center;">R3968   R4167</td><td style="text-align: center;">R5000   R8071</td><td style="text-align: center;">D0   D3071</td></tr> <tr> <td style="text-align: center;">D</td><td style="text-align: center;">○</td><td style="text-align: center;">○</td><td style="text-align: center;">○*</td><td style="text-align: center;">○</td><td style="text-align: center;">○*</td><td style="text-align: center;">○*</td><td style="text-align: center;">○</td></tr> </tbody> </table>													Range	Y	M	SM	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	Oper- and	Y0   Y255	M0   M1911	M1912   M2001	S0   S999	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	D	○	○	○*	○	○	○	○	○	○	○	○	○*	○*	○
Range	Y	M	SM	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR																																												
Oper- and	Y0   Y255	M0   M1911	M1912   M2001	S0   S999	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071																																												
D	○	○	○*	○	○	○	○	○	○	○	○	○*	○*	○																																												
Description	<ul style="list-style-type: none"> <li>When the set control "EN" =1 or "EN ↑" (P instruction) is from 0 to 1, sets the bit of a coil or all bits of a register to 1.</li> </ul>																																																									
Example 1	Single Coil Set																																																									
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Ladder Diagram</th><th style="text-align: center;">Key Operations</th><th style="text-align: center;">Mnemonic Codes</th></tr> </thead> <tbody> <tr> <td style="text-align: center; height: 150px;"></td><td style="text-align: center; height: 150px;">  </td><td style="text-align: center; height: 150px;"> ORG X 0  SET P Y 0  ORG X 1  RST P Y 0 </td></tr> </tbody> </table>				Ladder Diagram	Key Operations	Mnemonic Codes			ORG X 0 SET P Y 0 ORG X 1 RST P Y 0																																																	
Ladder Diagram	Key Operations	Mnemonic Codes																																																								
		ORG X 0 SET P Y 0 ORG X 1 RST P Y 0																																																								

## Basic Function Instruction

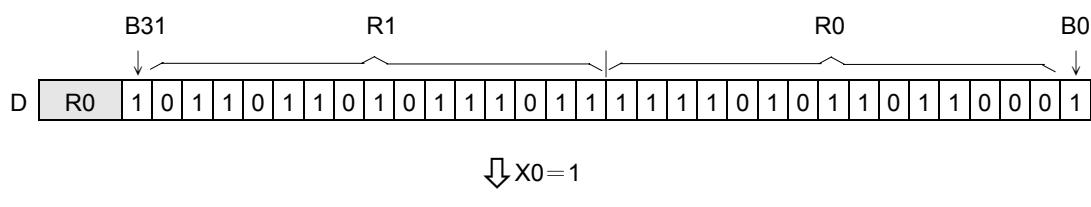
<b>SET DP</b>	SET (Set coil or all the bits of register to 1)	<b>SET DP</b>
Example 2	Set 16-Bit Register	

Ladder Diagram	Key Operations	Mnemonic Codes
		ORG            X            0 SET    P    R    0

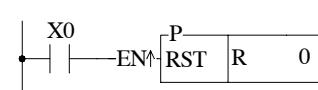
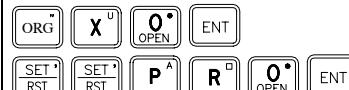
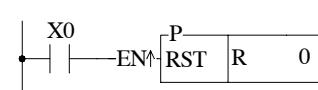
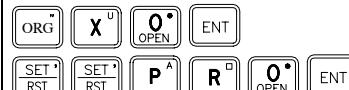
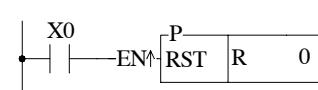
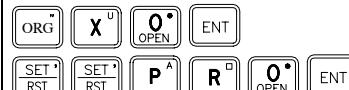


### Example 3 32-Bit Register Set

Ladder Diagram	Key Operations	Mnemonic Codes
 <pre>       X0               +-- EN              D              SET              R  0     </pre>	 <pre>       ORG   X^ OPEN ENT       SET  SHIFT S  R^ OPEN ENT     </pre>	ORG X 0 SET D R 0

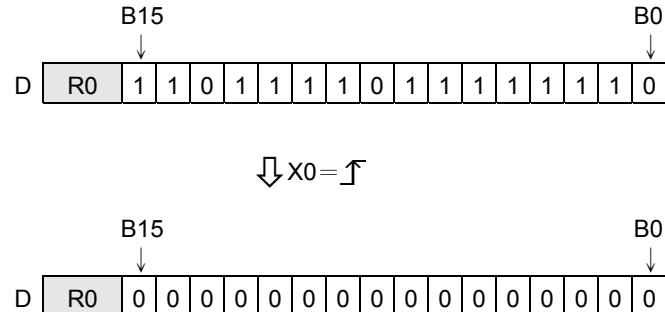


## Basic Function Instruction

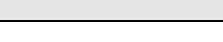
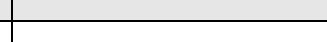
RST <b>DP</b>	RESET (Reset the coil or the register to 0)												RST <b>DP</b>														
Symbol																											
	<u>Ladder Symbol</u>												<u>Operand</u>														
													D: Destination to be reset (the number of a coil or a register)														
Description																											
● When the reset control "EN" =1 or "EN ↑" ( <b>P</b> instruction) from 0 to 1, resets the coil or register to 0.																											
Example 1	Single Coil Reset																										
Please refer to example 1 for the SET instruction shown in page 7-8.																											
Example 2	16-Bit Register Reset																										
<table border="1"> <thead> <tr> <th colspan="3">Ladder Diagram</th> <th colspan="3">Key Operations</th> <th colspan="3">Mnemonic Codes</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td>ORG      X      0</td> </tr> <tr> <td></td> <td></td> <td>RST    P    R    0</td> </tr> </tbody> </table>													Ladder Diagram			Key Operations			Mnemonic Codes					ORG      X      0			RST    P    R    0
Ladder Diagram			Key Operations			Mnemonic Codes																					
		ORG      X      0																									
		RST    P    R    0																									

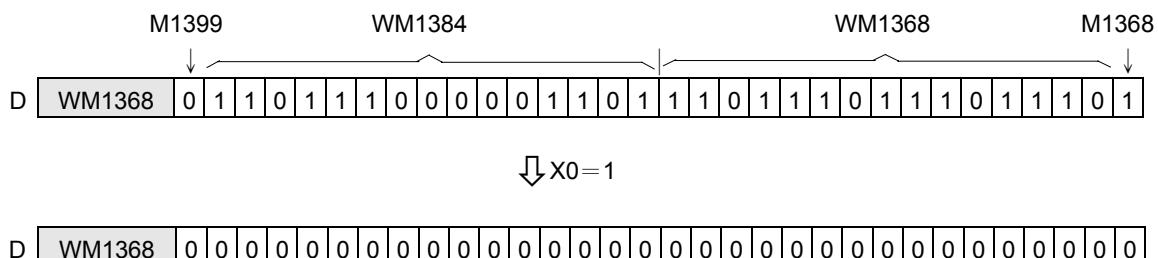
## Basic Function Instruction

RST <b>DP</b>	RESET (Reset the coil or register to 0)	RST <b>DP</b>
---------------	--	---------------



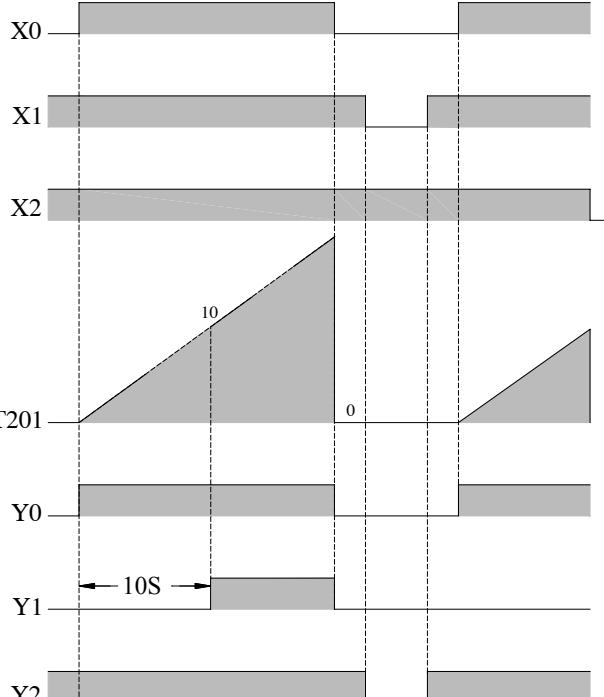
### Example 3 | 32-Bit Register Reset

Ladder Diagram	Key Operations	Mnemonic Codes
		ORG X 0 RST D WM1368



## Basic Function Instruction

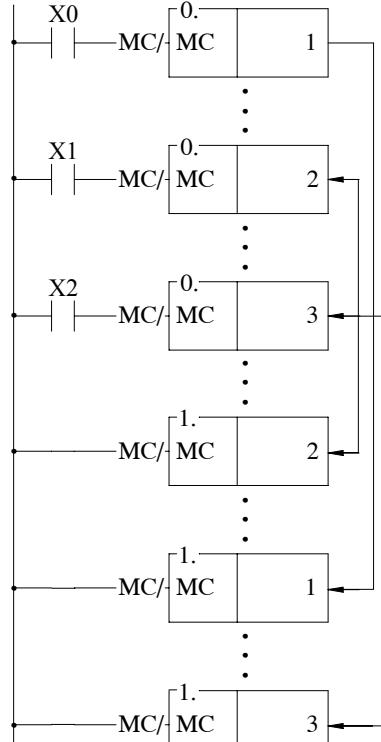
FUN 0 MC	MATER CONTROL LOOP START	FUN 0 MC																																						
Symbol	Ladder Symbol	Operand																																						
	<p>Master Control — EN/ [MC] N</p>	<p>N: Master Control Loop number (N=0~127) the number N cannot be used repeatedly.</p>																																						
Description	<ul style="list-style-type: none"> <li>There are a total of 128 MC loops (N=0~127). Every Master Control Start instruction, MC N, must correspond to a Master Control End instruction, MCE N, which has the same loop number as MC N. They must always be used in pairs and you should also make sure that the MCE N instruction is after the MC N instruction.</li> <li>When the Master Control input "EN/" is 1, then this MC N instruction will not be executed, as it does not exist.</li> <li>When the Master Control input "EN/" is 0, the master control loop is active, the area between the MC N and MCE N is called the Master Control active loop area. All the status of OUT coils or Timers within Master Control active loop area will be cleared to 0. Other instructions will not be executed.</li> </ul>																																							
Example	<p>Ladder Diagram</p> <p>Key Operations</p> <p>Mnemonic Codes</p> <table border="1"> <tbody> <tr> <td>ORG</td> <td>X</td> <td>0</td> </tr> <tr> <td>FUN</td> <td>0</td> <td></td> </tr> <tr> <td></td> <td>N :</td> <td>1</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>1</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>0</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>2</td> </tr> <tr> <td>T201</td> <td>PV :</td> <td>10</td> </tr> <tr> <td>ORG</td> <td>T</td> <td>201</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>1</td> </tr> <tr> <td>FUN</td> <td>1</td> <td></td> </tr> <tr> <td></td> <td>N :</td> <td>1</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>1</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>2</td> </tr> </tbody> </table>	ORG	X	0	FUN	0			N :	1	ORG	X	1	OUT	Y	0	ORG	X	2	T201	PV :	10	ORG	T	201	OUT	Y	1	FUN	1			N :	1	ORG	X	1	OUT	Y	2
ORG	X	0																																						
FUN	0																																							
	N :	1																																						
ORG	X	1																																						
OUT	Y	0																																						
ORG	X	2																																						
T201	PV :	10																																						
ORG	T	201																																						
OUT	Y	1																																						
FUN	1																																							
	N :	1																																						
ORG	X	1																																						
OUT	Y	2																																						

FUN 0 MC	MATER CONTROL LOOP START	FUN 0 MC
	 <p>The timing diagram illustrates the execution of a master control loop. The master control input (X0) is asserted at 10s. Subsequent inputs X1 and X2 are asserted at 10s. A timer T201 is triggered at 10s with a pulse width of 10s. The outputs Y0, Y1, and Y2 are asserted at 10s.</p>	

Remark1: MC/MCE instructions can be used in nesting or interleaving as shown to the right:

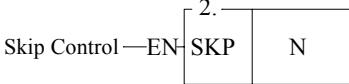
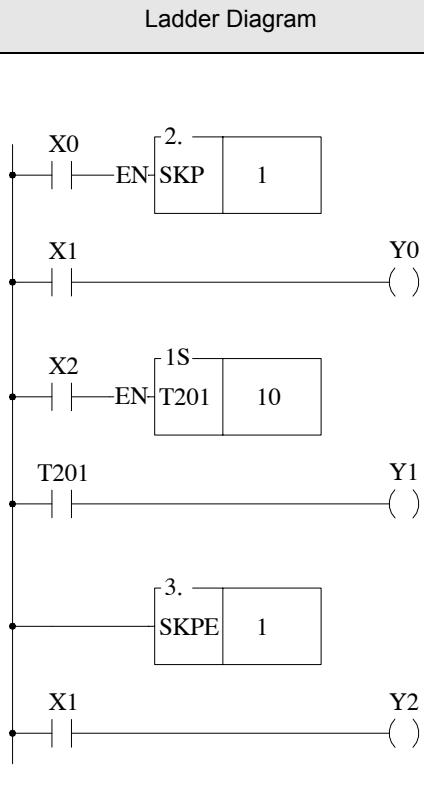
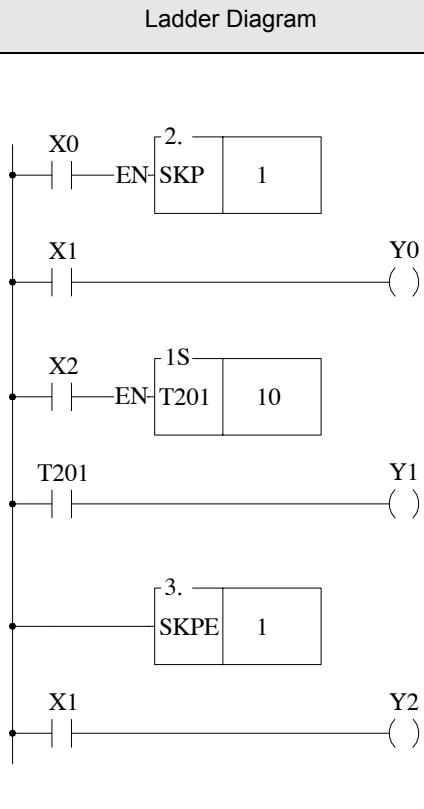
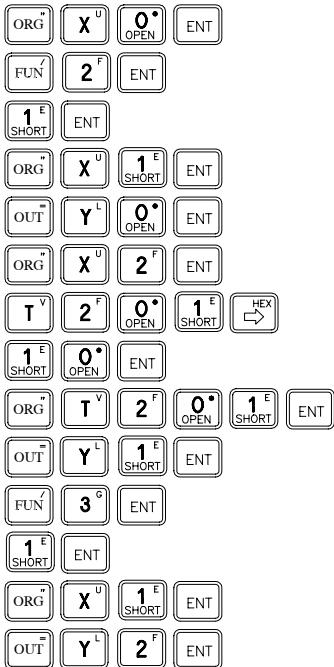
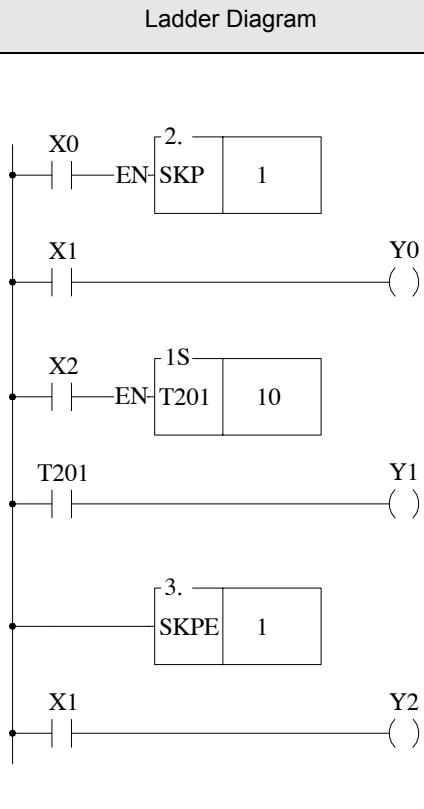
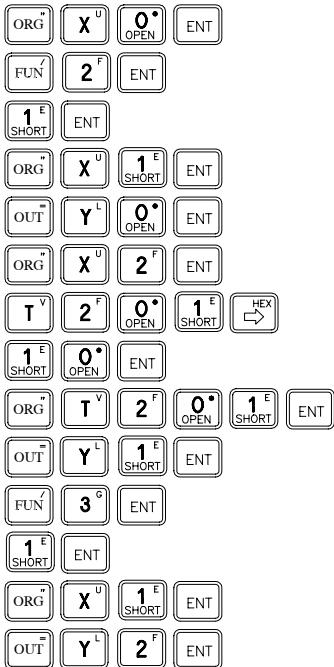
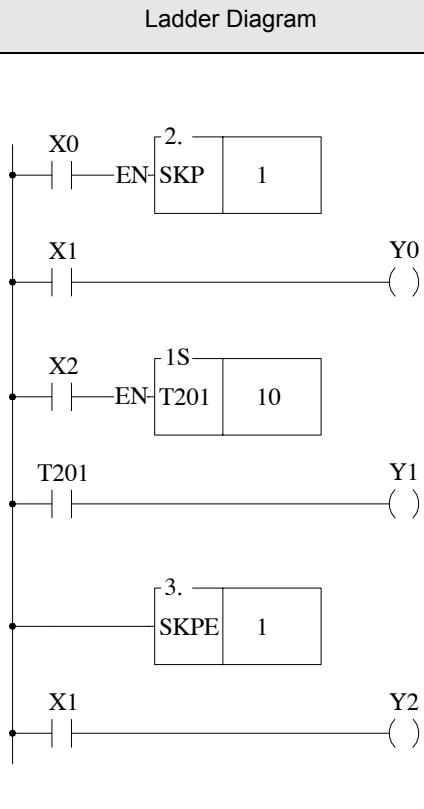
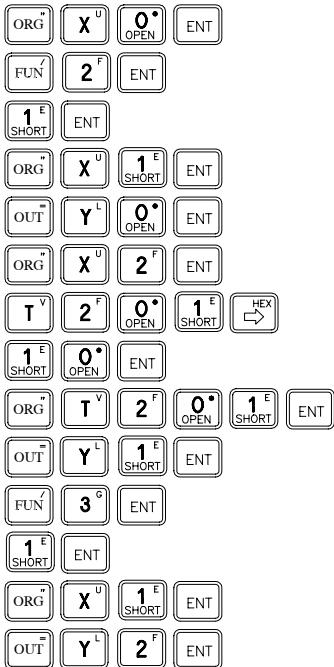
Remark2:

- When M1918=0 and the master input changes from 0→1, and if pulse type function instructions exist in the master control loop, then these instructions will have a chance to be executed only once (when the first time the master control input changes from 0→1). Afterwards, no matter how many times the master control input changes from 0→1, the pulse type function instructions will not be executed again.
- When M1918=1 and the master control input changes from 0→1, and if pulse type function instructions exist in the master control loop, then each time the master control input changes from 0→1 the pulse type function instructions in the master control loop will be executed as long as the action conditions are satisfied.
- When a counting instruction exists in the master control loop, set M1918 to 0 can avoid counting error.
- When the pulse type function instructions in the master control loop must act upon the 0→1 input change by the master control, the flag M1918 should be set to 1.

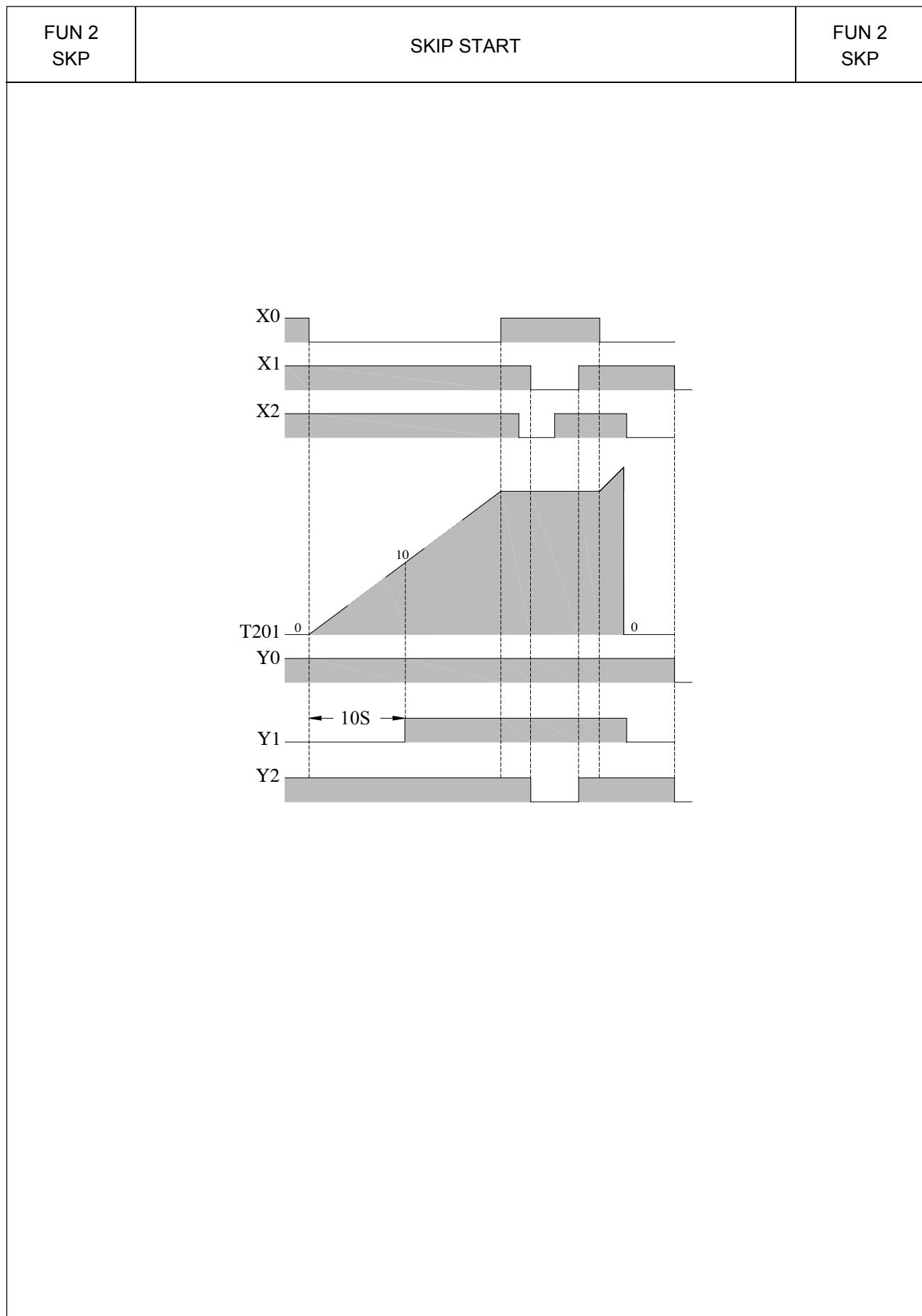


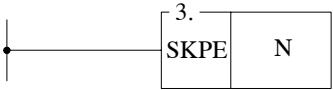
## Basic Function Instruction

FUN 1 MCE	MASTER CONTROL LOOP END	FUN 1 MCE
Symbol	<u>Ladder Symbol</u>	<u>Operand</u>
		N: Master Control End number (N=0~127) N can not be used repeatedly.
Description	<ul style="list-style-type: none"> <li>Every MCE N must correspond to a Master Control Start instruction. They must always be used as a pair and you should also make sure that the MCE N instruction is after the MC N instruction. After the MC N instruction has been executed, all output coil status and timers will be cleared to 0 and no other instructions will be executed. The program execution will resume until a MCE instruction which has the same N number as MC N instruction appears.</li> <li>MCE instruction does not require an input control because the instruction itself forms a network which other instructions can not connect to it. If the MC instruction has been executed then the master control operation will be completed when the execution of the program reaches the MCE instruction. If MC N instruction has never been executed then the MCE instruction will do nothing.</li> </ul>	
Description	<ul style="list-style-type: none"> <li>Please refer to the example and explanations for MC instruction.</li> </ul>	

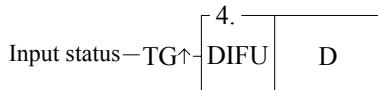
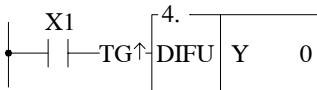
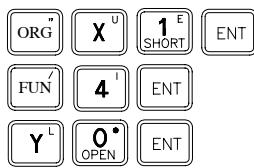
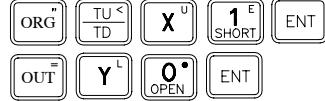
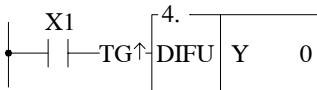
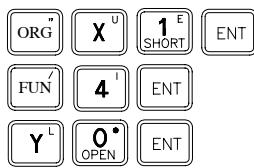
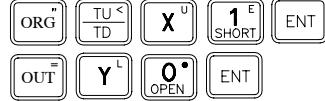
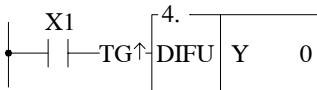
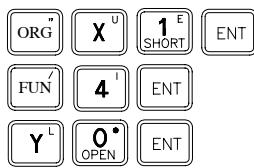
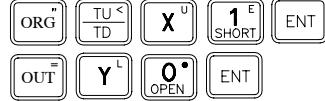
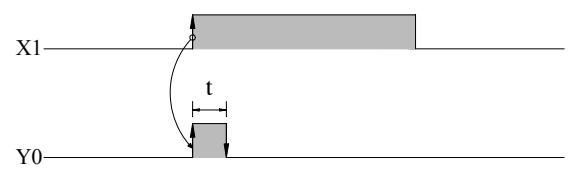
FUN 2 SKP	SKIP START	FUN 2 SKP																																													
Symbol	<u>Ladder Symbol</u> 	<u>Operand</u> N: Skip loop number (N=0~127), N can not be used repeatedly.																																													
Description																																															
	<ul style="list-style-type: none"> <li>● There are total 128 SKP loops (N=0~127). Every skip start instruction, SKP N, must correspond to a skip end instruction, SKPE N, which has the same loop number as SKP N. They must always be used as a pair and you should also make sure that the SKPE N instruction is after the SKP N instruction.</li> <li>● When the skip control "EN" is 0, then the Skip Start instruction will not be executed.</li> <li>● When the skip control "EN" is 1, the range between the SKP N and SKPE N which is so called the Skip active loop area will be skipped, that is all the instructions in this area will not be executed. Therefore the statuses of the discrete or registers in this Skip active loop area will be retained.</li> </ul>																																														
Example		<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Ladder Diagram</th> <th style="text-align: center;">Key Operations</th> <th style="text-align: center;">Mnemonic Codes</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">  </td> <td style="text-align: center;">  </td> <td style="text-align: center;"> <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">ORG</td> <td style="width: 33%;">X</td> <td style="width: 33%;">0</td> </tr> <tr> <td>FUN</td> <td>2</td> <td></td> </tr> <tr> <td></td> <td>N :</td> <td>1</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>1</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>0</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>2</td> </tr> <tr> <td>T201</td> <td>PV :</td> <td>10</td> </tr> <tr> <td>ORG</td> <td>T</td> <td>201</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>1</td> </tr> <tr> <td>FUN</td> <td>3</td> <td></td> </tr> <tr> <td></td> <td>N :</td> <td>1</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>1</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>2</td> </tr> </table> </td> </tr> </tbody> </table>	Ladder Diagram	Key Operations	Mnemonic Codes			<table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">ORG</td> <td style="width: 33%;">X</td> <td style="width: 33%;">0</td> </tr> <tr> <td>FUN</td> <td>2</td> <td></td> </tr> <tr> <td></td> <td>N :</td> <td>1</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>1</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>0</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>2</td> </tr> <tr> <td>T201</td> <td>PV :</td> <td>10</td> </tr> <tr> <td>ORG</td> <td>T</td> <td>201</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>1</td> </tr> <tr> <td>FUN</td> <td>3</td> <td></td> </tr> <tr> <td></td> <td>N :</td> <td>1</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>1</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>2</td> </tr> </table>	ORG	X	0	FUN	2			N :	1	ORG	X	1	OUT	Y	0	ORG	X	2	T201	PV :	10	ORG	T	201	OUT	Y	1	FUN	3			N :	1	ORG	X	1	OUT	Y	2
Ladder Diagram	Key Operations	Mnemonic Codes																																													
		<table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">ORG</td> <td style="width: 33%;">X</td> <td style="width: 33%;">0</td> </tr> <tr> <td>FUN</td> <td>2</td> <td></td> </tr> <tr> <td></td> <td>N :</td> <td>1</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>1</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>0</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>2</td> </tr> <tr> <td>T201</td> <td>PV :</td> <td>10</td> </tr> <tr> <td>ORG</td> <td>T</td> <td>201</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>1</td> </tr> <tr> <td>FUN</td> <td>3</td> <td></td> </tr> <tr> <td></td> <td>N :</td> <td>1</td> </tr> <tr> <td>ORG</td> <td>X</td> <td>1</td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>2</td> </tr> </table>	ORG	X	0	FUN	2			N :	1	ORG	X	1	OUT	Y	0	ORG	X	2	T201	PV :	10	ORG	T	201	OUT	Y	1	FUN	3			N :	1	ORG	X	1	OUT	Y	2						
ORG	X	0																																													
FUN	2																																														
	N :	1																																													
ORG	X	1																																													
OUT	Y	0																																													
ORG	X	2																																													
T201	PV :	10																																													
ORG	T	201																																													
OUT	Y	1																																													
FUN	3																																														
	N :	1																																													
ORG	X	1																																													
OUT	Y	2																																													

## Basic Function Instruction



FUN 3 SKPE	SKIP END	FUN 3 SKPE
Symbol	<p style="text-align: center;"><u>Ladder Symbol</u></p> 	<u>Operand</u>
		N : SKIP END Loop number (N=0~127) N can not be used repeatedly.
Description	<ul style="list-style-type: none"> <li>● Every SKPE N must correspond to a SKP N instruction. They must always be used as a pair and you should also make sure that the SKPE N instruction is behind the SKP N instruction.</li> <li>● SKPE instruction does not require an input control because the instruction itself forms a network which other instructions can not connect to it. If the SKP N instruction has been executed then the skip operation will be completed when the execution of the program reaches the SKPE N instruction. If SKP N instruction has never been executed then the SKPE instruction will do nothing.</li> </ul>	
Example	<ul style="list-style-type: none"> <li>● Please refer to the example and explanations for SKP N instruction.</li> </ul> <p><b>Remark :</b> SKP/SKPE instructions can be used by nesting or interleaving. The coding rules are the same as for the MC/MCE instructions. Please refer to the section of MC/MCE instructions.</p>	

## Basic Function Instruction

FUN 4 DIFU	DIFFERENTIAL UP	FUN 4 DIFU															
Symbol	<u>Ladder Symbol</u>	<u>Operand</u>															
		D: a specific coil number where the result of the Differential Up operation is stored.															
	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: center;">Range</th> <th style="text-align: center;">Y</th> <th style="text-align: center;">M</th> <th style="text-align: center;">SM</th> <th style="text-align: center;">S</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Oper- and</td> <td style="text-align: center;">Y0   Y255</td> <td style="text-align: center;">M0   M1911</td> <td style="text-align: center;">M1912   M2001</td> <td style="text-align: center;">S0   S999</td> </tr> <tr> <td style="text-align: center;">D</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td style="text-align: center;">○</td> </tr> </tbody> </table>	Range	Y	M	SM	S	Oper- and	Y0   Y255	M0   M1911	M1912   M2001	S0   S999	D	○	○	○*	○	
Range	Y	M	SM	S													
Oper- and	Y0   Y255	M0   M1911	M1912   M2001	S0   S999													
D	○	○	○*	○													
Description	<ul style="list-style-type: none"> <li>The DIFU instruction is used to output the up differentiation of a node status (status input to "TG ↑") and the pulse signal resulting from the status change at the rising edge of the "TG ↑" for one scan time is stored to a coil specified by D.</li> <li>The functionality of this instruction can also be achieved by using a TU contact.</li> </ul>																
Example	The results of the following two samples are exactly the same																
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">Ladder Diagram</th> <th style="text-align: center; padding: 5px;">Key Operations</th> <th style="text-align: center; padding: 5px;">Mnemonic Codes</th> </tr> </thead> <tbody> <tr> <td style="padding: 10px;"> <b>Example 1</b>   </td><td style="padding: 10px;">  </td><td style="padding: 10px;"> ORG      X      1  FUN      4  D :      Y      0 </td></tr> <tr> <td style="padding: 10px;"> <b>Example 2</b>   </td><td style="padding: 10px;">  </td><td style="padding: 10px;"> ORG    TU    X      1  OUT      Y      0 </td></tr> </tbody> </table>	Ladder Diagram	Key Operations	Mnemonic Codes	<b>Example 1</b> 		ORG      X      1 FUN      4 D :      Y      0	<b>Example 2</b> 		ORG    TU    X      1 OUT      Y      0							
Ladder Diagram	Key Operations	Mnemonic Codes															
<b>Example 1</b> 		ORG      X      1 FUN      4 D :      Y      0															
<b>Example 2</b> 		ORG    TU    X      1 OUT      Y      0															
		t: scan time															

FUN 5 DIFD	DIFFERENTIAL DOWN	FUN 5 DIFD
Symbol	Ladder Symbol	Operand
	<p>Input status — TG↓</p>	N: a specific coil number where the result of the Differential Down operation is stored.
Description	<ul style="list-style-type: none"> <li>The DIFD instruction is used to output the down differentiation of a node status (status input to "TG ↓") and the pulse signal resulting from the status change at the falling edge of the "TG ↓" for one scan time is stored to a coil specified by D.</li> <li>The functionality of this instruction can also be achieved by using a TD contact.</li> </ul>	
Example	<p>The results of the following two samples are exactly the same</p>	
Ladder Diagram		Key Operations
<b>Example 1</b> 		
<b>Example 2</b> 		

## Basic Function Instruction

FUN 6 <b>D P</b> BSHF	BIT SHIFT (Shifts the data of the 16-bit or 32-bit register to left or to right by one bit)	FUN 6 <b>D P</b> BSHF																																																		
Symbol																																																				
	<u>Ladder Symbol</u>	<u>Operand</u>																																																		
<p>Shift control—EN↑</p> <p>Fill-in bit—INB</p> <p>Shift direction—L/R</p> <p>Clear control—CLR</p>	<p>D : OTB— Shift-out bit (FO0)</p> <p>D: The register number for shifting</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th rowspan="2">Range</th> <th>WY</th> <th>WM</th> <th></th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> </tr> <tr> <th>WY0   WY240</th> <th>WM0   WM1896</th> <th>WS0   WS984</th> <th>T0   T255</th> <th>C0   C255</th> <th>R0   R3839</th> <th>R3904   R3967</th> <th>R3968   R4167</th> <th>R5000   R8071</th> <th>D0   D3071</th> </tr> </thead> <tbody> <tr> <td>D</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> </tr> </tbody></table>	Range	WY	WM		TMR	CTR	HR	OR	SR	ROR	DR	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	D	○	○	○	○	○	○	○	○*	○*	○																			
Range	WY		WM		TMR	CTR	HR	OR	SR	ROR	DR																																									
	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071																																										
D	○	○	○	○	○	○	○	○*	○*	○																																										
Description	<ul style="list-style-type: none"> <li>When the status of clear control "CLR" is at 1, then the data of register D and FO0 will all be cleared to 0. Other input signals are all in effect.</li> <li>When the status of clear control is "CLR" at 0, then the shift operation is permissible. When the shift control "EN" = 1 or "EN ↑" (P instruction) from 0 to 1, the data of the register will be shifted to right (L/R=0) or to left (L/R=1) by one bit. The shifted-out bit (MSB when shift to left and LSB when shift to right) for both cases will be sent to FO0. The vacated bit space (LSB when shift to left and MSB when shift to right) due to shift operation will be filled in by the input status of fill-in bit "INB".</li> </ul>																																																			
Example	Shifts the 16-bit register data																																																			
	<p>Ladder diagram</p>	<p>Key Operations</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>ORG</td> <td>X</td> <td>1</td> </tr> <tr> <td>LD<sup>+</sup></td> <td>X</td> <td>2</td> </tr> <tr> <td>LD<sup>+</sup></td> <td>X</td> <td>3</td> </tr> <tr> <td>LD<sup>+</sup></td> <td>X</td> <td>4</td> </tr> <tr> <td>FUN<sup>/</sup></td> <td>6<sup>K</sup></td> <td>P<sup>A</sup></td> <td>ENT</td> </tr> <tr> <td>R<sup>D</sup></td> <td>3<sup>G</sup></td> <td>ENT</td> </tr> <tr> <td>FO<sup>*</sup> NOT</td> <td>0<sup>E</sup> OPEN</td> <td>ENT</td> </tr> <tr> <td>OUT<sup>=</sup></td> <td>Y<sup>L</sup></td> <td>0<sup>E</sup> OPEN</td> <td>ENT</td> </tr> </table> <p>Mnemonic Codes</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>ORG</td> <td>X</td> <td>1</td> </tr> <tr> <td>LD</td> <td>X</td> <td>2</td> </tr> <tr> <td>LD</td> <td>X</td> <td>3</td> </tr> <tr> <td>LD</td> <td>X</td> <td>4</td> </tr> <tr> <td>FUN</td> <td>6P</td> <td></td> </tr> <tr> <td>D :</td> <td>R</td> <td>3</td> </tr> <tr> <td>FO</td> <td>0</td> <td></td> </tr> <tr> <td>OUT</td> <td>Y</td> <td>0</td> </tr> </table>	ORG	X	1	LD <sup>+</sup>	X	2	LD <sup>+</sup>	X	3	LD <sup>+</sup>	X	4	FUN <sup>/</sup>	6 <sup>K</sup>	P <sup>A</sup>	ENT	R <sup>D</sup>	3 <sup>G</sup>	ENT	FO <sup>*</sup> NOT	0 <sup>E</sup> OPEN	ENT	OUT <sup>=</sup>	Y <sup>L</sup>	0 <sup>E</sup> OPEN	ENT	ORG	X	1	LD	X	2	LD	X	3	LD	X	4	FUN	6P		D :	R	3	FO	0		OUT	Y	0
ORG	X	1																																																		
LD <sup>+</sup>	X	2																																																		
LD <sup>+</sup>	X	3																																																		
LD <sup>+</sup>	X	4																																																		
FUN <sup>/</sup>	6 <sup>K</sup>	P <sup>A</sup>	ENT																																																	
R <sup>D</sup>	3 <sup>G</sup>	ENT																																																		
FO <sup>*</sup> NOT	0 <sup>E</sup> OPEN	ENT																																																		
OUT <sup>=</sup>	Y <sup>L</sup>	0 <sup>E</sup> OPEN	ENT																																																	
ORG	X	1																																																		
LD	X	2																																																		
LD	X	3																																																		
LD	X	4																																																		
FUN	6P																																																			
D :	R	3																																																		
FO	0																																																			
OUT	Y	0																																																		
X3=1 (Left shift)	<p>B15      B0</p> <p>Y0      X2</p> <p>Shifts the 16-bit data to left by one bit</p>																																																			
X3=0 (Right shift)	<p>B15      B0</p> <p>X2      Y0</p> <p>Shifts the 16-bit data to right by one bit</p>																																																			

FUN 7 <b>D</b> UDCTR	UP/DOWN COUNTER (16-bit or 32-bit up and down 2-phase Counter)	FUN 7 <b>D</b> UDCTR																																																								
Symbol	<p><u>Ladder Symbol</u></p> <p><u>Operand</u></p> <p>CV: The number of the Up/Down Counter PV: Preset value of the counter or it's register number</p>																																																									
	<table border="1"> <thead> <tr> <th>Range</th><th>WX</th><th>WY</th><th>WM</th><th>WS</th><th>TMR</th><th>CTR</th><th>HR</th><th>IR</th><th>OR</th><th>SR</th><th>ROR</th><th>DR</th><th>K</th></tr> </thead> <tbody> <tr> <td>Oper- and</td><td>WX0   WX240</td><td>WY0   WY240</td><td>WM0   WM1896</td><td>WS0   WS984</td><td>T255   T255</td><td>C0   C255</td><td>R0   R3839</td><td>R3840   R3903</td><td>R3904   R3967</td><td>R3968   R4167</td><td>R5000   R8071</td><td>D0   D3071</td><td>16/32-bit +/- number</td></tr> <tr> <td>CV</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr> <td>PV</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○*</td><td>○*</td><td>○</td><td>○</td><td>○</td></tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	CV														PV	○	○	○	○	○	○	○	○	○*	○*	○	○	○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K																																													
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number																																													
CV																																																										
PV	○	○	○	○	○	○	○	○	○*	○*	○	○	○																																													
Description	<ul style="list-style-type: none"> <li>When the clear control "CLR" is 1, the counter's CV will be reset to 0 and the counter will not be able to count.</li> <li>When the clear control "CLR" is 0, counting will then be allowed. The nature of the instruction is a P instruction. Therefore, when the clock "CK ↑" is 0→1 (rising edge), the CV will increased by 1 (if U/D=1) or decreased by 1 (if U/D=0).</li> <li>When CV=PV, F00("Count-Up) will change to 1". If there are more clocks input, the counter will continue counting which cause CV≠PV. Then, F00 will immediately change to 0. This means the "Count-Up" signal will only be equal to 1 if CV=PV, or else it will be equal to 0 (Care should be taken to this difference from the "Count-Up" signal of the general counter).</li> <li>The upper limit of up count value is 32767 (16-bit) or 2147483647 (32-bit). After the upper limit is reached, if another up count clock is received, the counting value will become -32768 or -2147483648 (the lower limit of down count).</li> <li>The lower limit of down count value is -32767 (16-bit) or -2147483647 (32-bit). After the lower limit is reached, if another down count clock is received, the counting value will become 32768 or 2147483648 (the upper limit of up count).</li> <li>If U/D is fixed as 1, the instruction will become a single-phase up count counter. If U/D is fixed as 0, the instruction will become a single-phase down count counter.</li> </ul>																																																									
Example	The diagram below is an application example of UDCTR instruction being applied to an encoder.																																																									

## Basic Function Instruction

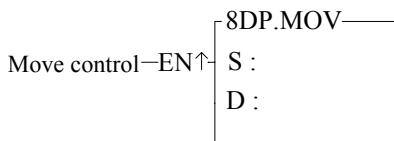
FUN 7 <b>D</b> UDCTR	UP/DOWN COUNTER (16-bit or 32-bit up/down 2-phase Counter)	FUN 7 <b>D</b> UDCTR																														
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #cccccc; padding: 5px;">Ladder Diagram</th><th style="background-color: #cccccc; padding: 5px;">Key Operations</th><th style="background-color: #cccccc; padding: 5px;">Mnemonic Codes</th></tr> </thead> <tbody> <tr> <td style="padding: 10px;"> </td><td style="padding: 10px; text-align: center;"> </td><td style="padding: 10px;"> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">ORG</td><td style="width: 33%;">X</td><td style="width: 33%;">18</td></tr> <tr> <td>LD</td><td>X</td><td>17</td></tr> <tr> <td>LD</td><td>X</td><td>16</td></tr> <tr> <td>FUN</td><td>7</td><td></td></tr> <tr> <td>R</td><td>0</td><td></td></tr> <tr> <td>SHIFT</td><td>OR</td><td>3</td></tr> <tr> <td>FO</td><td>NOT</td><td></td></tr> <tr> <td>OUT</td><td>Y</td><td>0</td></tr> </table> </td></tr> </tbody> </table>			Ladder Diagram	Key Operations	Mnemonic Codes			<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">ORG</td><td style="width: 33%;">X</td><td style="width: 33%;">18</td></tr> <tr> <td>LD</td><td>X</td><td>17</td></tr> <tr> <td>LD</td><td>X</td><td>16</td></tr> <tr> <td>FUN</td><td>7</td><td></td></tr> <tr> <td>R</td><td>0</td><td></td></tr> <tr> <td>SHIFT</td><td>OR</td><td>3</td></tr> <tr> <td>FO</td><td>NOT</td><td></td></tr> <tr> <td>OUT</td><td>Y</td><td>0</td></tr> </table>	ORG	X	18	LD	X	17	LD	X	16	FUN	7		R	0		SHIFT	OR	3	FO	NOT		OUT	Y	0
Ladder Diagram	Key Operations	Mnemonic Codes																														
		<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">ORG</td><td style="width: 33%;">X</td><td style="width: 33%;">18</td></tr> <tr> <td>LD</td><td>X</td><td>17</td></tr> <tr> <td>LD</td><td>X</td><td>16</td></tr> <tr> <td>FUN</td><td>7</td><td></td></tr> <tr> <td>R</td><td>0</td><td></td></tr> <tr> <td>SHIFT</td><td>OR</td><td>3</td></tr> <tr> <td>FO</td><td>NOT</td><td></td></tr> <tr> <td>OUT</td><td>Y</td><td>0</td></tr> </table>	ORG	X	18	LD	X	17	LD	X	16	FUN	7		R	0		SHIFT	OR	3	FO	NOT		OUT	Y	0						
ORG	X	18																														
LD	X	17																														
LD	X	16																														
FUN	7																															
R	0																															
SHIFT	OR	3																														
FO	NOT																															
OUT	Y	0																														

**Remark 1:** Since the counting operation of UDCTR is implemented by software scanning, therefore if the clock speed is faster than the scan speed, lose count may then happen (generally the clock should not exceed 20Hz depending on the size of the program). Please use the software or hardware high-speed counter in the PLC. Refer to the "High Speed Counter Application" in the Advanced Manual.

**Remark 2:** In order to ensure the proper counting, the sustain time of the status of clock input should greater than 1 scan time.

FUN 8 <b>D P</b> MOV	MOVE (Moves data from S to D)	FUN 8 <b>D P</b> MOV
-------------------------	----------------------------------	-------------------------

## Description

Ladder SymbolOperand

S: Source register number

D: Destination register number

The S, N, D may combine with V, Z to serve indirect addressing

Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V ` Z
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

## Description

- Moves (writes) the data of S to a specified register D when the move control input "EN" =1 or "EN ↑" (**P** instruction) from 0 to 1.

## Example

Writes a constant data into a 16-bit register.

Ladder Diagram	Key Operations	Mnemonic Codes
		ORG X 0 FUN 8P S : 10 D : R 0

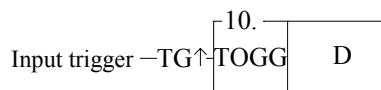
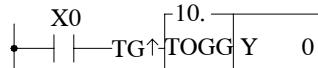
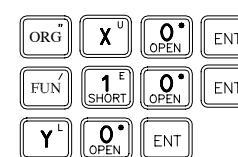
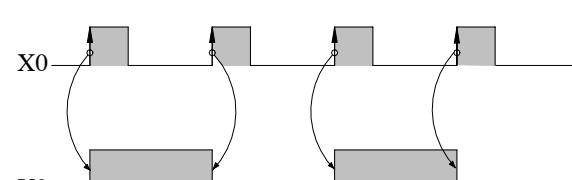
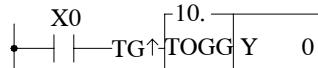
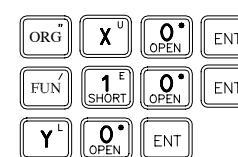
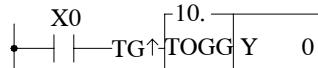
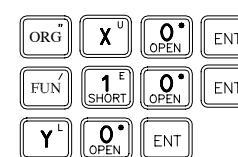
S K 10

↓ X0 = J

D R0 10

## Basic Function Instruction

FUN 9 <b>D P</b> MOV/	MOVE INVERSE (Inverts the data of S and moves the result to a specified device D)	FUN 9 <b>D P</b> MOV/																																																										
Symbol																																																												
	<u>Ladder Symbol</u>	<u>Operand</u>																																																										
		S: Source register number D: Destination register number S, N, D may combine with V, Z to serve indirect addressing																																																										
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th><th style="padding: 2px;">WX</th><th style="padding: 2px;">WY</th><th style="padding: 2px;">WM</th><th style="padding: 2px;">WS</th><th style="padding: 2px;">TMR</th><th style="padding: 2px;">CTR</th><th style="padding: 2px;">HR</th><th style="padding: 2px;">IR</th><th style="padding: 2px;">OR</th><th style="padding: 2px;">SR</th><th style="padding: 2px;">ROR</th><th style="padding: 2px;">DR</th><th style="padding: 2px;">K</th><th style="padding: 2px;">XR</th></tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;">Oper- and</td><td style="padding: 2px;">WX0   WX240</td><td style="padding: 2px;">WY0   WY240</td><td style="padding: 2px;">WM0   WM1896</td><td style="padding: 2px;">WS0   WS984</td><td style="padding: 2px;">T0   T255</td><td style="padding: 2px;">C0   C255</td><td style="padding: 2px;">R0   R3839</td><td style="padding: 2px;">R3840   R3847</td><td style="padding: 2px;">R3904   R3967</td><td style="padding: 2px;">R3968   R4167</td><td style="padding: 2px;">R5000   R8071</td><td style="padding: 2px;">D0   D3071</td><td style="padding: 2px;">16/32-bit +/- number</td><td style="padding: 2px;">V ` Z</td></tr> <tr> <td style="text-align: left; padding: 2px;">S</td><td style="padding: 2px;">○</td><td style="padding: 2px;">○</td></tr> <tr> <td style="text-align: left; padding: 2px;">D</td><td style="padding: 2px;"></td><td style="padding: 2px;">○</td><td style="padding: 2px;">○</td><td style="padding: 2px;">○</td><td style="padding: 2px;">○</td><td style="padding: 2px;">○</td><td style="padding: 2px;">○</td><td style="padding: 2px;"></td><td style="padding: 2px;">○</td><td style="padding: 2px;">○*</td><td style="padding: 2px;">○*</td><td style="padding: 2px;">○</td><td style="padding: 2px;">○</td></tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3847	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V ` Z	S	○	○	○	○	○	○	○	○	○	○	○	○	○	D		○	○	○	○	○	○		○	○*	○*	○	○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																														
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3847	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V ` Z																																														
S	○	○	○	○	○	○	○	○	○	○	○	○	○																																															
D		○	○	○	○	○	○		○	○*	○*	○	○																																															
Description	<ul style="list-style-type: none"> <li>● Inverts the data of S (changes the status from 0 to 1 and from 1 to 0) and moves the results to a specified register D when the move control input "EN" =1 or "EN ↑" (P instruction) from 0 to 1.</li> </ul>																																																											
Example	Moves the inverted data of a 16-bit register to another 16-bit register.																																																											
	<p style="text-align: center;"><u>Ladder Diagram</u></p>	<p style="text-align: center;"><u>Key Operations</u></p>	<p style="text-align: center;"><u>Mnemonic Codes</u></p> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">ORG</td> <td style="width: 33%;">X 0</td> <td style="width: 33%;">0</td> </tr> <tr> <td>FUN</td> <td>9</td> <td></td> </tr> <tr> <td>S :</td> <td>R 0</td> <td></td> </tr> <tr> <td>D :</td> <td>WY 8</td> <td></td> </tr> </table>	ORG	X 0	0	FUN	9		S :	R 0		D :	WY 8																																														
ORG	X 0	0																																																										
FUN	9																																																											
S :	R 0																																																											
D :	WY 8																																																											
	<p style="text-align: center;">B15</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="border: 1px solid black; padding: 2px;">S</td> <td style="border: 1px solid black; padding: 2px;">R0</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">5555H</td> </tr> </table>	S	R0	0	1	0	1	0	1	0	1	0	1	0	1	5555H	<p style="text-align: center;">B0</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="border: 1px solid black; padding: 2px;">D</td> <td style="border: 1px solid black; padding: 2px;">WY8</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">1</td> <td style="border: 1px solid black; padding: 2px;">0</td> <td style="border: 1px solid black; padding: 2px;">AAAAAH</td> </tr> </table>	D	WY8	1	0	1	0	1	0	1	0	1	0	1	0	AAAAAH																												
S	R0	0	1	0	1	0	1	0	1	0	1	0	1	5555H																																														
D	WY8	1	0	1	0	1	0	1	0	1	0	1	0	AAAAAH																																														

FUN 10 TOGG	<b>TOGGLE SWITCH</b> (Changes the output status when the rising edge of control input occur)	FUN 10 TOGG															
Symbol	<u>Ladder Symbol</u>	<u>Operand</u>															
		D: the coil number of the toggle switch															
	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="background-color: #cccccc;">Range</th> <th style="background-color: #cccccc;">Y</th> <th style="background-color: #cccccc;">M</th> <th style="background-color: #cccccc;">SM</th> <th style="background-color: #cccccc;">S</th> </tr> </thead> <tbody> <tr> <td style="background-color: #cccccc;">Oper- and</td> <td>Y0   Y255</td> <td>M0   M1911</td> <td>M1912   M2001</td> <td>S0   S999</td> </tr> <tr> <td style="background-color: #cccccc;">D</td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td><input checked="" type="radio"/>*</td> <td><input type="radio"/></td> </tr> </tbody> </table>	Range	Y	M	SM	S	Oper- and	Y0   Y255	M0   M1911	M1912   M2001	S0   S999	D	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>	
Range	Y	M	SM	S													
Oper- and	Y0   Y255	M0   M1911	M1912   M2001	S0   S999													
D	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>													
Description	<ul style="list-style-type: none"> <li>The coil D changes its status (from 1 to 0 and from 0 to 1) each time the input "TG ↑" is triggered from 0 to 1 (rising edge).</li> </ul>																
Example	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">Ladder Diagram</th> <th style="text-align: center; padding: 5px;">Key Operations</th> <th style="text-align: center; padding: 5px;">Mnemonic Codes</th> </tr> </thead> <tbody> <tr> <td style="padding: 10px;">  </td> <td style="padding: 10px;">  </td> <td style="padding: 10px;"> ORG      X    0  FUN     10  [D : ]    Y    0 </td> </tr> </tbody> </table> 		Ladder Diagram	Key Operations	Mnemonic Codes			ORG      X    0 FUN     10 [D : ]    Y    0									
Ladder Diagram	Key Operations	Mnemonic Codes															
		ORG      X    0 FUN     10 [D : ]    Y    0															

## Basic Function Instruction

FUN 11 <b>D P</b> (+)	<b>ADDITION</b> (Performs addition of the data specified at Sa and Sb and stores the result in D)	FUN 11 <b>D P</b> (+)																																																																								
<b>Symbol</b>																																																																										
	<u>Ladder Symbol</u>	<u>Operand</u>																																																																								
		<p>Sa: Augend Sb: Addend D : Destination register to store the results of the addition S, N, D may combine with V, Z to serve indirect addressing</p>																																																																								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th> <th style="padding: 2px;">WX</th> <th style="padding: 2px;">WY</th> <th style="padding: 2px;">WM</th> <th style="padding: 2px;">WS</th> <th style="padding: 2px;">TMR</th> <th style="padding: 2px;">CTR</th> <th style="padding: 2px;">HR</th> <th style="padding: 2px;">IR</th> <th style="padding: 2px;">OR</th> <th style="padding: 2px;">SR</th> <th style="padding: 2px;">ROR</th> <th style="padding: 2px;">DR</th> <th style="padding: 2px;">K</th> <th style="padding: 2px;">XR</th> </tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;">Oper- and</td> <td style="padding: 2px;">WX0   WX240</td> <td style="padding: 2px;">WY0   WY240</td> <td style="padding: 2px;">WM0   WM1896</td> <td style="padding: 2px;">WS0   WS984</td> <td style="padding: 2px;">T255</td> <td style="padding: 2px;">C255</td> <td style="padding: 2px;">R0   R3839</td> <td style="padding: 2px;">R3840   R3903</td> <td style="padding: 2px;">R3904   R3967</td> <td style="padding: 2px;">R3968   R4167</td> <td style="padding: 2px;">R5000   R8071</td> <td style="padding: 2px;">D0   D3071</td> <td style="padding: 2px;">16/32-bit +/- number</td> <td style="padding: 2px;">V ` Z</td> </tr> <tr> <td style="text-align: left; padding: 2px;">Sa</td> <td style="padding: 2px;">○</td> </tr> <tr> <td style="text-align: left; padding: 2px;">Sb</td> <td style="padding: 2px;">○</td> </tr> <tr> <td style="text-align: left; padding: 2px;">D</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○*</td> <td style="padding: 2px;">○*</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○</td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255	C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V ` Z	Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	D	○	○	○	○	○	○	○	○	○*	○*	○	○	○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																												
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255	C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V ` Z																																																												
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○																																																													
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○																																																													
D	○	○	○	○	○	○	○	○	○*	○*	○	○	○																																																													
<b>Description</b>	<ul style="list-style-type: none"> <li>● Performs the addition of the data specified at Sa and Sb and writes the results to a specified register D when the add control input "EN" =1 or "EN ↑" (<b>P</b> instruction) from 0 to 1. If the result of addition is equal to 0 then set FO0 to 1. If carry occurs (the result exceeds 32767 or 2147483647) then set FO1 to 1. If borrow occurs (adding negative numbers resulting in a sum less than -32768 or -2147483648), then set the FO2 to 1. All the FO statuses are retained until this instruction is executed again and overwritten by a new result.</li> </ul>																																																																									
<b>Example</b>	16-bit addition																																																																									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 2px;">Ladder Diagram</th> <th style="text-align: center; padding: 2px;">Key Operations</th> <th style="text-align: center; padding: 2px;">Mnemonic Codes</th> </tr> </thead> <tbody> <tr> <td style="text-align: center; padding: 10px;"> </td><td style="text-align: center; padding: 10px;"> </td><td style="text-align: center; padding: 10px;"> <p>ORG      X      0 FUN      11P Sa :      R      0 Sb :      R      1 D :      R      2 FO      1 OUT      Y      0</p> </td></tr> <tr> <td style="text-align: center; padding: 10px;"> <table border="1" style="margin-bottom: 5px;"> <tr> <td style="padding: 2px;">Sa</td> <td style="padding: 2px;">R0</td> <td style="padding: 2px;">12345</td> </tr> <tr> <td style="padding: 2px;">Sb</td> <td style="padding: 2px;">R1</td> <td style="padding: 2px;">20425</td> </tr> </table> <p style="text-align: right; margin-right: 10px;">R0 + R1 = 32770</p> <p style="text-align: center; margin-top: 10px;"><math>\downarrow X0 = \text{True}</math></p> <table border="1" style="margin-top: 10px;"> <tr> <td style="padding: 2px;">D</td> <td style="padding: 2px;">R2</td> <td style="padding: 2px;">2</td> </tr> </table> <p style="text-align: right; margin-right: 10px;">32768 + 2 = 32770</p> <p style="text-align: center; margin-top: 10px;">Y0 = 1 (carry 1 represents +32768)</p> </td><td></td></tr> </tbody> </table>	Ladder Diagram	Key Operations	Mnemonic Codes			<p>ORG      X      0 FUN      11P Sa :      R      0 Sb :      R      1 D :      R      2 FO      1 OUT      Y      0</p>	<table border="1" style="margin-bottom: 5px;"> <tr> <td style="padding: 2px;">Sa</td> <td style="padding: 2px;">R0</td> <td style="padding: 2px;">12345</td> </tr> <tr> <td style="padding: 2px;">Sb</td> <td style="padding: 2px;">R1</td> <td style="padding: 2px;">20425</td> </tr> </table> <p style="text-align: right; margin-right: 10px;">R0 + R1 = 32770</p> <p style="text-align: center; margin-top: 10px;"><math>\downarrow X0 = \text{True}</math></p> <table border="1" style="margin-top: 10px;"> <tr> <td style="padding: 2px;">D</td> <td style="padding: 2px;">R2</td> <td style="padding: 2px;">2</td> </tr> </table> <p style="text-align: right; margin-right: 10px;">32768 + 2 = 32770</p> <p style="text-align: center; margin-top: 10px;">Y0 = 1 (carry 1 represents +32768)</p>	Sa	R0	12345	Sb	R1	20425	D	R2	2																																																									
Ladder Diagram	Key Operations	Mnemonic Codes																																																																								
		<p>ORG      X      0 FUN      11P Sa :      R      0 Sb :      R      1 D :      R      2 FO      1 OUT      Y      0</p>																																																																								
<table border="1" style="margin-bottom: 5px;"> <tr> <td style="padding: 2px;">Sa</td> <td style="padding: 2px;">R0</td> <td style="padding: 2px;">12345</td> </tr> <tr> <td style="padding: 2px;">Sb</td> <td style="padding: 2px;">R1</td> <td style="padding: 2px;">20425</td> </tr> </table> <p style="text-align: right; margin-right: 10px;">R0 + R1 = 32770</p> <p style="text-align: center; margin-top: 10px;"><math>\downarrow X0 = \text{True}</math></p> <table border="1" style="margin-top: 10px;"> <tr> <td style="padding: 2px;">D</td> <td style="padding: 2px;">R2</td> <td style="padding: 2px;">2</td> </tr> </table> <p style="text-align: right; margin-right: 10px;">32768 + 2 = 32770</p> <p style="text-align: center; margin-top: 10px;">Y0 = 1 (carry 1 represents +32768)</p>	Sa	R0	12345	Sb	R1	20425	D	R2	2																																																																	
Sa	R0	12345																																																																								
Sb	R1	20425																																																																								
D	R2	2																																																																								

FUN 12 <b>D P</b> ( - )	SUBTRACTION (Performs subtraction of the data specified at Sa and Sb and stores the result in D)	FUN 12 <b>D P</b> ( - )																																																																								
Symbol																																																																										
	<u>Ladder Symbol</u>	<u>Operand</u>																																																																								
Subtraction control-EN↑	<p>12DP.(-)</p> <p>Sa: D=0 — Difference=0 (FO0)  Sb: CY — Carry (FO1)  D : BR — Borrow (FO2)</p>	<p>Sa: Minuend  Sb: Subtrahend  D : Destination register to store the results of the subtraction  Sa, Sb, D may combine with V, Z to serve indirect addressing</p>																																																																								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th><th style="padding: 2px;">WX</th><th style="padding: 2px;">WY</th><th style="padding: 2px;">WM</th><th style="padding: 2px;">WS</th><th style="padding: 2px;">TMR</th><th style="padding: 2px;">CTR</th><th style="padding: 2px;">HR</th><th style="padding: 2px;">IR</th><th style="padding: 2px;">OR</th><th style="padding: 2px;">SR</th><th style="padding: 2px;">ROR</th><th style="padding: 2px;">DR</th><th style="padding: 2px;">K</th><th style="padding: 2px;">XR</th></tr> <tr> <th style="text-align: left; padding: 2px;">Oper- and</th><th style="padding: 2px;">WX0   WX240</th><th style="padding: 2px;">WY0   WY240</th><th style="padding: 2px;">WM0   WM1896</th><th style="padding: 2px;">WS0   WS984</th><th style="padding: 2px;">T255</th><th style="padding: 2px;">C0   C255</th><th style="padding: 2px;">R0   R3839</th><th style="padding: 2px;">R3840   R3903</th><th style="padding: 2px;">R3904   R3967</th><th style="padding: 2px;">R3968   R4167</th><th style="padding: 2px;">R5000   R8071</th><th style="padding: 2px;">D0   D3071</th><th style="padding: 2px;">16/32-bit +/- number</th><th style="padding: 2px;">V · Z</th></tr> </thead> <tbody> <tr> <td style="padding: 2px;">Sa</td><td style="padding: 2px;">○</td><td style="padding: 2px;">○</td></tr> <tr> <td style="padding: 2px;">Sb</td><td style="padding: 2px;">○</td><td style="padding: 2px;">○</td></tr> <tr> <td style="padding: 2px;">D</td><td style="padding: 2px;"></td><td style="padding: 2px;">○</td><td style="padding: 2px;">○*</td><td style="padding: 2px;">○*</td><td style="padding: 2px;">○</td><td style="padding: 2px;"></td><td style="padding: 2px;">○</td></tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V · Z	Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	D		○	○	○	○	○	○	○	○*	○*	○		○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																												
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V · Z																																																												
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○																																																													
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○																																																													
D		○	○	○	○	○	○	○	○*	○*	○		○																																																													
Description	<ul style="list-style-type: none"> <li>● Performs the subtraction of the data specified at Sa and Sb and writes the results to a specified register D when the subtract control input "EN" =1 or "EN ↑" (<b>P</b> instruction) from 0 to 1. If the result of subtraction is equal to 0 then set FO0 to 1. If carry occurs (subtracting a negative number from a positive number and the result exceeds 32767 or 2147483647), then set FO1 to 1. If borrow occurs (subtracting a positive number from a negative number and the resulted difference is less than -32768 or -2147483648), then set FO2 to 1. All the FO statuses are retained until this instruction is executed again and overwritten by a new result.</li> </ul>																																																																									
Example	16-bit subtraction																																																																									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">Ladder Diagram</th><th style="text-align: center; padding: 5px;">Key Operations</th><th style="text-align: center; padding: 5px;">Mnemonic Codes</th></tr> </thead> <tbody> <tr> <td style="text-align: center; padding: 10px;"> <p>X0 — EN —&gt; 12.(-)</p> <p>Sa: R 0      D=0-</p> <p>Sb: R 1      CY —</p> <p>D : R 2      BR — ( )</p> <p>Y2</p> </td><td style="text-align: center; padding: 10px;"> </td><td style="text-align: center; padding: 10px;"> <p>ORG      X      0</p> <p>FUN      12</p> <p><b>Sa :</b>      R      0</p> <p><b>Sb :</b>      R      1</p> <p><b>D :</b>      R      2</p> <p>FO      2</p> <p>OUT      Y      2</p> </td></tr> </tbody> </table>	Ladder Diagram	Key Operations	Mnemonic Codes	<p>X0 — EN —&gt; 12.(-)</p> <p>Sa: R 0      D=0-</p> <p>Sb: R 1      CY —</p> <p>D : R 2      BR — ( )</p> <p>Y2</p>		<p>ORG      X      0</p> <p>FUN      12</p> <p><b>Sa :</b>      R      0</p> <p><b>Sb :</b>      R      1</p> <p><b>D :</b>      R      2</p> <p>FO      2</p> <p>OUT      Y      2</p>																																																																			
Ladder Diagram	Key Operations	Mnemonic Codes																																																																								
<p>X0 — EN —&gt; 12.(-)</p> <p>Sa: R 0      D=0-</p> <p>Sb: R 1      CY —</p> <p>D : R 2      BR — ( )</p> <p>Y2</p>		<p>ORG      X      0</p> <p>FUN      12</p> <p><b>Sa :</b>      R      0</p> <p><b>Sb :</b>      R      1</p> <p><b>D :</b>      R      2</p> <p>FO      2</p> <p>OUT      Y      2</p>																																																																								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">Sa</td><td style="width: 33%;">R0</td><td style="width: 33%;">—5</td></tr> <tr> <td>Sb</td><td>R1</td><td>32767</td></tr> </table> <p style="margin-left: 150px;">R0 — R1 = — 32772</p> <p style="text-align: center; margin-top: 10px;">↓ X0 = 1</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">D</td><td style="width: 33%;">R2</td><td style="width: 33%;">—4</td></tr> </table> <p style="margin-left: 150px;">—32768 — 4 = — 32772</p> <p style="text-align: center; margin-top: 10px;">Y2 = 1 (borrow 1 represents —32768) Please refer to section 6.5</p>	Sa	R0	—5	Sb	R1	32767	D	R2	—4																																																																
Sa	R0	—5																																																																								
Sb	R1	32767																																																																								
D	R2	—4																																																																								

## Basic Function Instruction

FUN 13 <b>D P</b> ( *)	<b>MULTIPLICATION</b> (Performs multiplication of the data specified at Sa and Sb and stores the result in D)	FUN 13 <b>D P</b> ( *)																																																																											
Symbol																																																																													
	<u>Ladder Symbol</u>	<u>Operand</u>																																																																											
	<p>Multiplication -EN↑ control</p> <p>13DP.(*)</p> <p>Sa: D=0—Product=0 (FO0) Sb: D&lt;0—Product is negative (FO1)</p>	<p>Sa: Multiplicand Sb: Multiplier D : Destination register to store the results of the multiplication. Sa, Sb, D may combine with V, Z to serve indirect addressing</p>																																																																											
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Range</th><th>WX</th><th>WY</th><th>WM</th><th>WS</th><th>TMR</th><th>CTR</th><th>HR</th><th>IR</th><th>OR</th><th>SR</th><th>ROR</th><th>DR</th><th>K</th><th>XR</th></tr> <tr> <th style="text-align: left;">Oper- and</th><th>WX0   WX240</th><th>WY0   WY240</th><th>WM0   WM1896</th><th>WS0   WS984</th><th>T0   T255</th><th>C0   C255</th><th>R0   R3839</th><th>R3840   R3903</th><th>R3904   R3967</th><th>R3968   R4167</th><th>R5000   R8071</th><th>D0   D3071</th><th>16/32-bit +/- number</th><th>V · Z</th></tr> </thead> <tbody> <tr> <td>Sa</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr> <tr> <td>Sb</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr> <tr> <td>D</td><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td><td>○*</td><td>○*</td><td>○</td><td>○</td><td></td><td>○</td></tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V · Z	Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○	D		○	○	○	○	○	○		○*	○*	○	○		○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																															
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V · Z																																																															
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																															
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																															
D		○	○	○	○	○	○		○*	○*	○	○		○																																																															
Description	<ul style="list-style-type: none"> <li>● Performs the multiplication of the data specified at Sa and Sb and writes the results to a specified register D when the multiplication control input "EN" =1 or "EN ↑ " (<b>P</b> instruction) from 0 to 1. If the product of multiplication is equal to 0 then set FO0 to 1. If the product is a negative number, then set FO1 to 1.</li> </ul>																																																																												
Example 1	16-bit multiplication																																																																												
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Ladder Diagram</th><th style="text-align: center;">Key Operations</th><th style="text-align: center;">Mnemonic Codes</th></tr> </thead> <tbody> <tr> <td> <p>X0 — EN↑</p> <p>13P.(*)</p> <p>Sa: R 0 — D=0— Sb: R 1 — D&lt;0— D : R 2</p> </td><td> <p>ORG X 0 FUN 13P Sa : R 0 Sb : R 1 D : R 2</p> </td><td> <p>ORG X 0 FUN 13P Sa : R 0 Sb : R 1 D : R 2</p> </td></tr> <tr> <td></td><td> <p style="text-align: center;">Sa      R0           12345           Multiplicand</p> <p style="text-align: center;">Sb      R1           4567           Multiplier</p> <hr/> <p style="text-align: center;">D      R3      R2           56379615           Product</p> </td><td></td></tr> </tbody> </table>	Ladder Diagram	Key Operations	Mnemonic Codes	<p>X0 — EN↑</p> <p>13P.(*)</p> <p>Sa: R 0 — D=0— Sb: R 1 — D&lt;0— D : R 2</p>	<p>ORG X 0 FUN 13P Sa : R 0 Sb : R 1 D : R 2</p>	<p>ORG X 0 FUN 13P Sa : R 0 Sb : R 1 D : R 2</p>		<p style="text-align: center;">Sa      R0           12345           Multiplicand</p> <p style="text-align: center;">Sb      R1           4567           Multiplier</p> <hr/> <p style="text-align: center;">D      R3      R2           56379615           Product</p>																																																																				
Ladder Diagram	Key Operations	Mnemonic Codes																																																																											
<p>X0 — EN↑</p> <p>13P.(*)</p> <p>Sa: R 0 — D=0— Sb: R 1 — D&lt;0— D : R 2</p>	<p>ORG X 0 FUN 13P Sa : R 0 Sb : R 1 D : R 2</p>	<p>ORG X 0 FUN 13P Sa : R 0 Sb : R 1 D : R 2</p>																																																																											
	<p style="text-align: center;">Sa      R0           12345           Multiplicand</p> <p style="text-align: center;">Sb      R1           4567           Multiplier</p> <hr/> <p style="text-align: center;">D      R3      R2           56379615           Product</p>																																																																												

## Basic Function Instruction

FUN 13 <b>D P</b> ( *)	<b>MULTIPLICATION</b> (Performs multiplication of the data specified at Sa and Sb and stores the result in D)	FUN 13 <b>D P</b> ( *)
Example 2	32-bit multiplication	

Ladder Diagram	Key Operations	Mnemonic Codes
 X0 — EN —> 13D.(*) Sa: R 0 —> D=0— Sb: R 2 —> D<0— D : R 4		ORG      X      0 FUN      13D Sa :      R      0 Sb :      R      2 D :      R      4

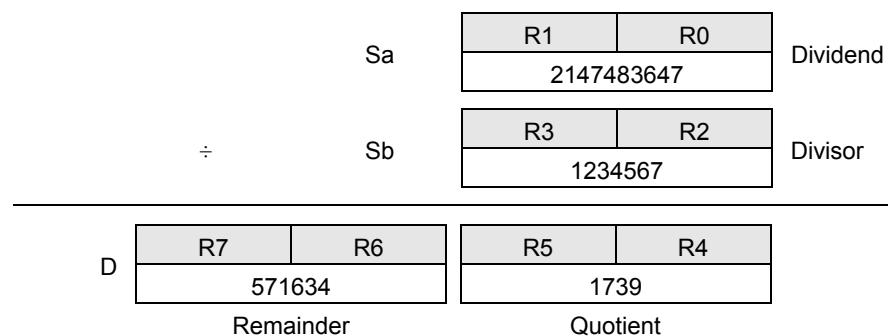


## Basic Function Instruction

FUN 14 <b>D P</b> ( $\diagup$ )	DIVISION (Performs division of the data specified at Sa and Sb and stores the result in D)	FUN 14 <b>D P</b> ( $\diagdown$ )																		
Symbol																				
	<u>Ladder Symbol</u>	<u>Operand</u>																		
Division control -EN↑	<p>14DP.( / )</p> <p>Division control -EN↑</p> <p>Sa: Sb: D :</p> <p>D=0 – Quotient=0 (FO0) ERR – Divisor is 0 (FO1)</p>	<p>Sa: Dividend Sb: Divisor D : Destination register to store the results of the division. Sa, Sb, D may combine with V, Z to serve indirect addressing</p>																		
Description	<ul style="list-style-type: none"> <li>● Performs the division of the data specified at Sa and Sb and writes the quotient and remainder to registers specified by register D when the division control input "EN" =1 or "EN ↑" (<b>P</b> instruction) from 0 to 1. If the quotient of division is equal to 0 then set FO0 to 1. If the divisor Sb=0 then set the error flag FO1 to 1 without executing the instruction.</li> </ul>																			
Example 1	16-bit division																			
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="3" style="text-align: center;">Ladder Diagram</th> <th colspan="3" style="text-align: center;">Key Operations</th> <th colspan="3" style="text-align: center;">Mnemonic Codes</th> </tr> </thead> <tbody> <tr> <td colspan="3"> <p>14.( / )</p> <p>X0</p> <p>EN</p> <p>Sa: R 0</p> <p>Sb: R 1</p> <p>D: R 2</p> <p>D=0 –</p> <p>ERR –</p> </td> <td colspan="3"> <p>ORG      X<sup>U</sup>      0<sup>OPEN</sup>      ENT</p> <p>FUN      1<sup>E</sup> SHORT      4<sup>I</sup>      ENT</p> <p>R<sup>D</sup>      0<sup>OPEN</sup>      ENT</p> <p>R<sup>D</sup>      1<sup>E</sup> SHORT      ENT</p> <p>R<sup>D</sup>      2<sup>F</sup>      ENT</p> </td> <td colspan="3"> <p>ORG      X      0</p> <p>FUN      14</p> <p>Sa :      R      0</p> <p>Sb :      R      1</p> <p>D :      R      2</p> </td> </tr> </tbody> </table>	Ladder Diagram			Key Operations			Mnemonic Codes			<p>14.( / )</p> <p>X0</p> <p>EN</p> <p>Sa: R 0</p> <p>Sb: R 1</p> <p>D: R 2</p> <p>D=0 –</p> <p>ERR –</p>			<p>ORG      X<sup>U</sup>      0<sup>OPEN</sup>      ENT</p> <p>FUN      1<sup>E</sup> SHORT      4<sup>I</sup>      ENT</p> <p>R<sup>D</sup>      0<sup>OPEN</sup>      ENT</p> <p>R<sup>D</sup>      1<sup>E</sup> SHORT      ENT</p> <p>R<sup>D</sup>      2<sup>F</sup>      ENT</p>			<p>ORG      X      0</p> <p>FUN      14</p> <p>Sa :      R      0</p> <p>Sb :      R      1</p> <p>D :      R      2</p>			
Ladder Diagram			Key Operations			Mnemonic Codes														
<p>14.( / )</p> <p>X0</p> <p>EN</p> <p>Sa: R 0</p> <p>Sb: R 1</p> <p>D: R 2</p> <p>D=0 –</p> <p>ERR –</p>			<p>ORG      X<sup>U</sup>      0<sup>OPEN</sup>      ENT</p> <p>FUN      1<sup>E</sup> SHORT      4<sup>I</sup>      ENT</p> <p>R<sup>D</sup>      0<sup>OPEN</sup>      ENT</p> <p>R<sup>D</sup>      1<sup>E</sup> SHORT      ENT</p> <p>R<sup>D</sup>      2<sup>F</sup>      ENT</p>			<p>ORG      X      0</p> <p>FUN      14</p> <p>Sa :      R      0</p> <p>Sb :      R      1</p> <p>D :      R      2</p>														
	<p style="margin-bottom: 10px;">÷</p> <p style="text-align: center;">Sa      R0 256      Dividend</p> <p style="text-align: center;">Sb      R1 12      Divisor</p> <hr style="width: 100%; border: 0; border-top: 1px solid black; margin: 10px 0;"/> <p style="text-align: center;">D      R3 4      Remainder      R2 21      Quotient</p>																			

FUN 14 DP (/)	DIVISION (Performs division of the data specified at Sa and Sb and stores the result in D)	FUN 14 DP (/)
Example 2	32-bit division	

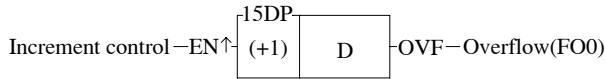
Ladder Diagram	Key Operations	Mnemonic Codes
		ORG X 0 FUN 14D Sa : R 0 Sb : R 2 D : R 4



## Basic Function Instruction

FUN 15 <b>D P</b> (+1)	INCREMENT (Adds 1 to the D value)	FUN 15 <b>D P</b> (+1)
---------------------------	--------------------------------------	---------------------------

Ladder symbol



Operand

D : The register to be increased  
D may combine with V, Z to serve indirect addressing

Range	WY	WM	WS	TMR	CTR	HR	OR	HR	HSCR	RTCR	SR	ROR	DR
Operand	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3919	R3920   R4047	R4096   R4127	R4128   R4135	R4136   R4167	R5000   R8071	D0   D3071
D	○	○	○	○	○	○	○	○	○	○*	○*	○	

- Adds 1 to the register D when the increment control input "EN" =1 or "EN ↑" (P instruction) from 0 to 1. If the value of D is already at the upper limit of positive number 32767 or 2147483647, adding one to this value will change it to the lower limit of negative number -32768 or -2147483648. At the same time, the overflow flag F00 (OVF) is set to 1.

### Example

16-bit increment register

Ladder diagram	Key operations	Mnemonic code
<p>X0 — EN ↑ (+1) R 0V OVF-</p>		<p>ORG TU X 0 FUN 15 D : R 0V</p>

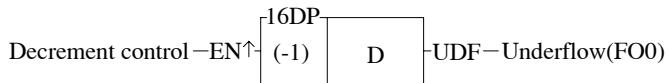
When V=100, 0 + 100 = 100

D R100 1

↓ X0 = ↗

D R100 2

FUN 16 <b>D P</b> (-1)	DECREMENT (Subtracts 1 from the D value)	FUN 16 <b>D P</b> (-1)
---------------------------	---	---------------------------

Ladder symbolOperand

D : The register to be decreased  
D may combine with V, Z to serve indirect addressing

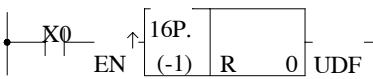
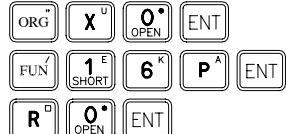
Range	WY	WM	WS	TMR	CTR	HR	OR	HR	HSCR	RTCR	SR	ROR	DR
Operand	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3919	R3920   R4047	R4096   R4127	R4128   R4135	R4136   R4167	R5000   R8071	D0   D3071
D	○	○	○	○	○	○	○	○	○	○	○*	○*	○

## Description

- Subtracts 1 from the register D when the decrement control input "EN" =1 or "EN ↑" (**P** instruction) from 0 to 1. If the value of D is already at the lower limit of negative number -32768 or -2147483648, subtracting one from this value will change it to the upper limit of positive number 32767 or 2147483647. At the same time, the underflow flag FO0 (UDF) is set to 1.

## Example

16-bit decrement register

Ladder diagram	Key operations	Mnemonic code
		ORG X 0 FUN 16P D : R 0

D [R0 | 0]

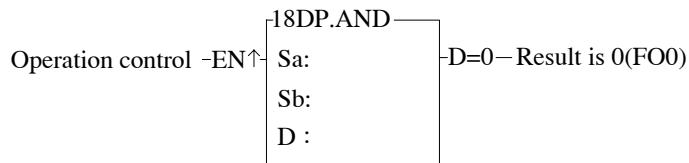
↓ X0 = ↑

D [R0 | -1]

## Basic Function Instruction

FUN 17 <b>D P</b> CMP	COMPARE (Compares the data of Sa and Sb and outputs the results to function Outputs)	FUN 17 <b>D P</b> CMP																																																																	
<u>Ladder symbol</u>		<u>Operand</u>																																																																	
		<p>Sa: The register to be compared</p> <p>Sb: The register to be compared</p> <p>Sa, Sb may combine with V, Z to serve indirect addressing</p>																																																																	
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Range</th><th>WX</th><th>WY</th><th>WM</th><th>WS</th><th>TMR</th><th>CTR</th><th>HR</th><th>IR</th><th>OR</th><th>HR</th><th>HSCR</th><th>RTCR</th><th>SR</th><th>ROR</th><th>DR</th><th>K</th></tr> </thead> <tbody> <tr> <td>Operand</td><td>WX0   WX240</td><td>WY0   WY240</td><td>WM0   WM1896</td><td>WS0   WS984</td><td>T0   T255</td><td>C0   C255</td><td>R0   R3839</td><td>R3804   R3903</td><td>R3904   R3919</td><td>R3920   R4047</td><td>R4096   R4127</td><td>R4128   R4135</td><td>R4136   R4167</td><td>R5000   R8071</td><td>D0   D3071</td><td>16/32 bit +/-number</td></tr> <tr> <td>Sa</td><td><input type="radio"/></td><td><input type="radio"/></td></tr> <tr> <td>Sb</td><td><input type="radio"/></td><td><input type="radio"/></td></tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	HR	HSCR	RTCR	SR	ROR	DR	K	Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3804   R3903	R3904   R3919	R3920   R4047	R4096   R4127	R4128   R4135	R4136   R4167	R5000   R8071	D0   D3071	16/32 bit +/-number	Sa	<input type="radio"/>	Sb	<input type="radio"/>																													
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	HR	HSCR	RTCR	SR	ROR	DR	K																																																			
Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3804   R3903	R3904   R3919	R3920   R4047	R4096   R4127	R4128   R4135	R4136   R4167	R5000   R8071	D0   D3071	16/32 bit +/-number																																																			
Sa	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																																																				
Sb	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																																																				
<ul style="list-style-type: none"> <li>Compares the data of Sa and Sb when the compare control input "EN" =1 or "EN ↑" (P instruction) from 0 to 1. If the data of Sa is equal to Sb, then set FO0 to 1. If the data of Sa&gt;Sb, then set FO1 to 1. If the data of Sa&lt;Sb, then set FO2 to 1. If the data of Sa &lt; Sb, then set the FO2 to 1.</li> </ul>																																																																			
Example	Compares the data of 16-bit register																																																																		
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Ladder diagram</th><th>Key operations</th><th>Mnemonic code</th></tr> </thead> <tbody> <tr> <td> </td><td> </td><td> <p>ORG      X    0</p> <p>FUN      17</p> <p>Sa : R 0</p> <p>Sb : R 1</p> <p>FO      2</p> <p>OUT     Y 0</p> </td></tr> </tbody> </table>		Ladder diagram	Key operations	Mnemonic code			<p>ORG      X    0</p> <p>FUN      17</p> <p>Sa : R 0</p> <p>Sb : R 1</p> <p>FO      2</p> <p>OUT     Y 0</p>																																																												
Ladder diagram	Key operations	Mnemonic code																																																																	
		<p>ORG      X    0</p> <p>FUN      17</p> <p>Sa : R 0</p> <p>Sb : R 1</p> <p>FO      2</p> <p>OUT     Y 0</p>																																																																	
<ul style="list-style-type: none"> <li>From the above example, we first assume the data of R0 is 1 and R1 is 2, and then compare the data by executing the CMP instruction. The FO0 and FO1 are set to 0 and FO2 (a&lt;b) is set to 1 since a&lt;b.</li> <li>If you want to have the compound results, such as <math>\geq</math> , <math>\leq</math> , <math>&lt;</math> , <math>&gt;</math> etc., please send = , <math>&lt;</math> and <math>&gt;</math> results to relay first and then combine the result from the relays.</li> <li>M1919=0, when this command is not executed, FO0, FO1, FO2 will remain in the status at last execution.</li> <li>M1919=1, when this command is not executed, FO0, FO1, FO2 are all cleared to 0.</li> <li>Control M1919 properly to obtain memory-holding function for functional command output.</li> </ul>																																																																			

FUN 18 <b>D P</b> AND	LOGICAL AND	FUN 18 <b>D P</b> AND
--------------------------	-------------	--------------------------

Ladder symbol

Sa: The register to be ANDed

Sb: The register to be ANDed

D : The register to store the result of AND

The Sa, Sb, D may combine with V, Z to serve indirect addressing application

Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	HR	HSCR	RTCR	SR	ROR	DR	K
Ope- rand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3804   R3903	R3904   R3919	R3920   R4047	R4096   R4127	R4128   R4135	R4136   R4167	R5000   R8071	D0   D3071	16/32 bit +/-number
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○	○	○	○*	○*	○	

- Performs logical AND operation for the data of Sa and Sb when the operation control input "EN" =1 or "EN ↑" ( **P** instruction) from 0 to 1. This operation compares the corresponding bits of Sa and Sb (B0~B15 or B0~B31). The bit in the D is set to 1 if both of the corresponding bits data of Sa and Sb is 1. The bit in the D is set to 0 if one of the corresponding bits is 0.

## Example      Operation of 16-bit logical AND

Ladder diagram	Key operations	Mnemonic code
<p>18P.AND Sa: R 0 Sb: R 1 D: R 2</p>		ORG X 0 FUN 18P Sa: R 0 Sb: R 1 D: R 2

B15	↓	Sa	R0   1   0   1   1   1   0   1   1   0   1   1   0   1   1   0   1	B0	↓	Sb	R1   1   1   1   0   1   1   1   0   1   0   1   0   0   1   1   0

↓ X0 =

B15	↓	D	R2   1   0   1   0   1   0   0   0   1   0   0   1   0   0   0   0	B0	↓		

## Basic Function Instruction

FUN 19 <b>D P</b> OR	LOGICAL OR	FUN 19 <b>D P</b> OR
-------------------------	------------	-------------------------

<u>Ladder symbol</u>		<u>Operand</u>
<p>Operation control -EN↑</p> <p>19DP.OR</p> <p>Sa: Sb: D :</p>		<p>Sa: The register to be ORed Sb: The register to be ORed D : The register to store the result of OR The Sa, Sb, D may combine with V, Z to serve indirect addressing</p>

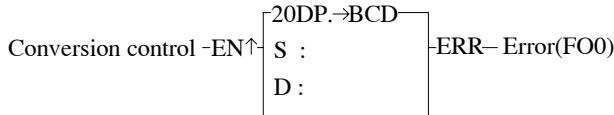
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	HR	HSCR	RTCR	SR	ROR	DR	K
Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255	C255	R0	R3804	R3904	R3920	R4096	R4128	R4136	R5000	D0   D3071	16/32 bit +/-number
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○		○	○	○*	○*	○	

- Performs logical OR operation for the data of Sa and Sb when the operation control input "EN" =1 or "EN ↑" (**P** instruction) from 0 to 1. This operation compares the corresponding bits of Sa and Sb (B0~B15 or B0~B31). The bit in the D is set to 1 if one of the corresponding of Sa or Sb is 1. The bit in the D is set to 0 if both of the corresponding bits of Sa and Sb is 0.

Example	Operation of 16-bit logical OR																																																
Ladder diagram		Key operations	Mnemonic code																																														
<p>X0</p> <p>EN</p> <p>19.OR</p> <p>Sa: R 0</p> <p>Sb: R 1</p> <p>D: R 2</p>			ORG X 0 FUN 19 Sa: R 0 Sb: R 1 D: R 2																																														
<p>B15</p> <p>B0</p> <p>↓</p> <p>↓</p> <p>Sa</p> <table border="1"> <tr><td>R0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>R1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> </table> <p>↓ X0=1</p> <p>B15</p> <p>B0</p> <p>↓</p> <p>↓</p> <p>D</p> <table border="1"> <tr><td>R2</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> </table>	R0	1	0	1	1	1	1	0	1	1	0	1	1	0	1	1	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	R2	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	
R0	1	0	1	1	1	1	0	1	1	0	1	1	0	1	1																																		
R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1																																		
R2	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1																																		

FUN 20 <b>D P</b> →BCD	BIN TO BCD CONVERSION (Converts BIN data of the device specified at S into BCD and stores the result in D)	FUN 20 <b>D P</b> →BCD
---------------------------	---	---------------------------

## Ladder symbol



## Operand

S : The register to be converted

D : The register to store the converted data  
(BCD code)

The S, D may combine with V, Z to serve indirect addressing

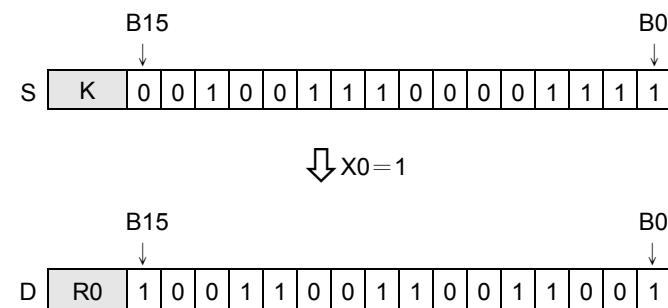
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	HR	HSCR	RTCR	SR	ROR	DR	K
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3804   R3903	R3940   R3919	R3920   R4047	R4096   R4127	R4128   R4135	R4136   R4167	R5000   R8071	D0   D3071	16/32 bit +/- number
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
D		○	○	○	○	○	○		○	○	○	○	○*	○*	○	

- FB-PLC uses binary code to store and to execute calculations. If want to send the internal PLC data to the external displays such as seven-segment displays, it is more convenient for us to read the result on screen by converting the BIN data to BCD data. For example, it is more clear for us to read the reading "12" instead of the binary code "1100."
  - Converts BIN data of the device specified at S into BCD and writes the result in D when the operation control input "EN" =1 or "EN ↑" ( instruction) from 0 to 1. If the data in S is not a BCD value (0~9999 or 0~99999999), then the error flag F00 is set to 1 and the old data of D are retained.

## Example

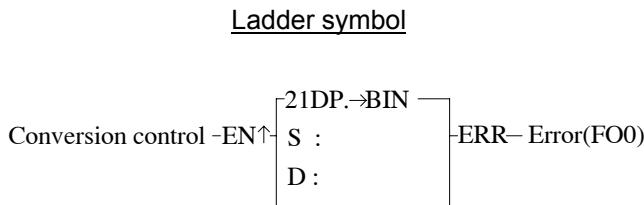
## 16-bit BIN to BCD conversion

Ladder diagram	Key operations	Mnemonic code																
<pre>       graph LR         X0((X0)) --- EN[EN]         EN --- 20.→BCD          20BCD[S : 9999]         20BCD --- D[R 0]         20BCD --- ERR      </pre>	<table border="1"> <tbody> <tr> <td>ORG</td> <td>X<sup>U</sup></td> <td>O<sup>•</sup> OPEN</td> <td>ENT</td> </tr> <tr> <td>FUN</td> <td>2<sup>F</sup></td> <td>O<sup>•</sup> OPEN</td> <td>ENT</td> </tr> <tr> <td>9<sup>0</sup></td> <td>9<sup>0</sup></td> <td>9<sup>0</sup></td> <td>9<sup>0</sup> ENT</td> </tr> <tr> <td>R<sup>o</sup></td> <td>O<sup>•</sup> OPEN</td> <td>ENT</td> <td></td> </tr> </tbody> </table>	ORG	X <sup>U</sup>	O <sup>•</sup> OPEN	ENT	FUN	2 <sup>F</sup>	O <sup>•</sup> OPEN	ENT	9 <sup>0</sup>	9 <sup>0</sup>	9 <sup>0</sup>	9 <sup>0</sup> ENT	R <sup>o</sup>	O <sup>•</sup> OPEN	ENT		<p>ORG      X      0      FUN      20      [S]      9999      [D]      R      0</p>
ORG	X <sup>U</sup>	O <sup>•</sup> OPEN	ENT															
FUN	2 <sup>F</sup>	O <sup>•</sup> OPEN	ENT															
9 <sup>0</sup>	9 <sup>0</sup>	9 <sup>0</sup>	9 <sup>0</sup> ENT															
R <sup>o</sup>	O <sup>•</sup> OPEN	ENT																



## Basic Function Instruction

FUN 21 <b>D P</b> →BIN	<b>BCD TO BIN CONVERSION</b> (Converts BCD data of the device specified at S into BIN and stores the result in D)	FUN 21 <b>D P</b> →BIN
---------------------------	--	---------------------------



S : The register to be converted  
D : The register to store the converted data  
(BIN code)  
The S, D may combine with V, Z to serve

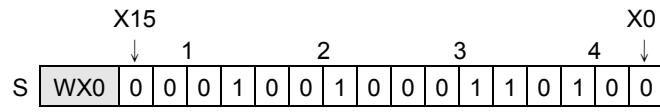
## indirect addressing

Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	HR	HSCR	RTCR	SR	ROR	DR
Oper- and	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3920	R4096	R4128	R4136	R5000	D0
	WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3919	R4047	R4127	R4135	R4167	R8071	D3071
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○	○	○	○*	○*	○

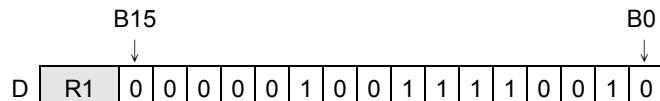
- The decimal (BCD) data must be converted to binary (BIN) data first in order for PLC to accept the data which is originally in decimal unit (BCD code) inputted from external device such as digital switch because the BCD data can not be accepted by PLC for its operations.
  - Converts BCD data of the device specified at S into BIN and writes the result in D when the operation control input "EN" =1 or "EN ↑" (**P** instruction) from 0 to 1. If the data in S is not in BCD, then the error flag F00 is set to 1 and the old data of D are retained.
  - Constant is converted to BIN automatically when store in program and can not be used as a source operand of this function.

#### Example 16-bit BCD to BIN conversion

Ladder diagram	Key operations	Mnemonic code
 <b>21P.→BIN</b> S : WX 0 D : R 1		ORG X 0 FUN 21P S : WX 0 D : R 1



$\downarrow x_0 = \top$



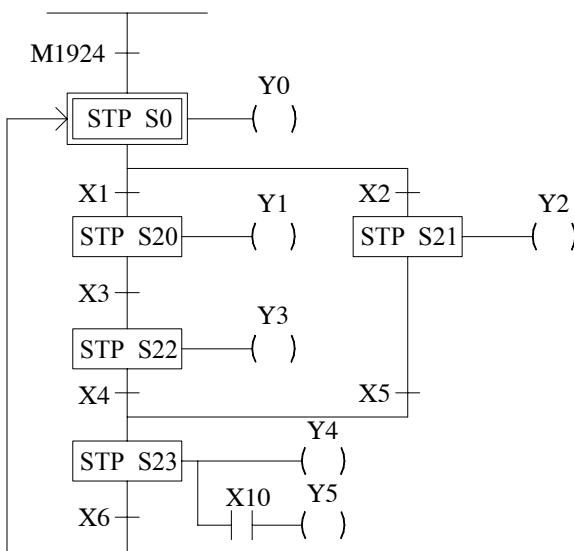
# Chapter 8 Step Instruction Description

Structured programming design is a major trend in software design. The benefits are high readability, easy maintenance, convenient updating and high quality and reliability. For the control applications, consisted of many sequential tasks, designed by conventional ladder program design methodology usually makes others hard to maintain. Therefore, it is necessary to combine the current widely used ladder diagrams with the sequential controls made especially for machine working flow. With help from step instructions, the design work will become more efficient, time saving and controlled. This kind of design method that combines process control and ladder diagram together is called the step ladder language.

The basic unit of step ladder diagram is a step. A step is equivalent to a movement (stop) in the machine operation where each movement has an output. The complete machine or the overall sequential control process is the combination of steps in serial or parallel. Its step-by-step sequential execution procedure allows others to be able to understand the machine operations thoroughly, so that design, operation, and maintenance will become more effective and simpler.

## 8.1 The Operation Principle of Step Ladder Diagram

### 【Example】



### 【Description】

1. **STP Sxxx** is the symbol representing a step Sxxx that can be one of S0 ~ S999. When executing the step (status ON), the ladder diagram on the right will be executed and the previous step and output will become OFF.
2. M1924 is on for a scan time after program start. Hence, as soon as ON, the stop of the initial step S0 is entered (S0 ON) while the other steps are kept inactive, i.e. Y1~Y5 are all OFF. This means M1924 ON → S0 ON → Y0 ON and Y0 will remain ON until one of the contacts X1 or X2 is ON.
3. Assume that X2 is ON first; the path to S21 will then be executed.  
$$X2 \text{ ON} \Rightarrow \begin{cases} S21 \text{ ON} \\ S0 \text{ OFF} \end{cases} \Rightarrow \begin{cases} Y2 \text{ ON} \\ Y0 \text{ OFF} \end{cases}$$

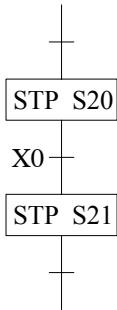
Y2 will remain ON until X5 is ON.
4. Assume that X5 is ON, the process will move forward to step S23.  
i.e.  $X5 \text{ ON} \Rightarrow \begin{cases} S23 \text{ ON} \\ S21 \text{ OFF} \end{cases} \Rightarrow \begin{cases} Y4 \text{ ON} \\ Y2 \text{ OFF} \end{cases}$ 

Y4 and Y5 will remain ON until X6 is ON.  
※ If X10 is ON, then Y5 will be ON.
5. Assume that X6 is ON, the process will move forward to S0.  
i.e.  $X6 \text{ ON} \Rightarrow \begin{cases} S0 \text{ ON} \\ S23 \text{ OFF} \end{cases} \Rightarrow \begin{cases} Y0 \text{ ON} \\ Y4, Y5 \text{ OFF} \end{cases}$ 

Then, a control process cycle is completed and the next control process cycle is entered.

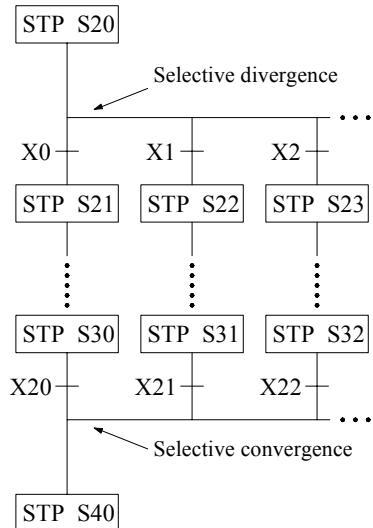
## 8.2 Basic Formation of Step Ladder Diagram

### ① Single path



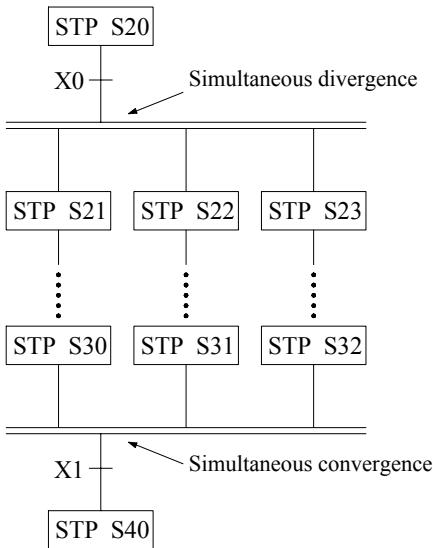
- Step S20 alone moves to step S21 through X0.
- X0 can be changed to other serial or parallel combination of contacts.

### ② Selective divergence/convergence



- Step S20 selects an only one path which divergent condition first met. E.g. X2 is ON first, then only the path of step S23 will be executed.
- A divergence may have up to 8 paths maximum.
- X1, X2, ...., X22 can all be replaced by the serial or parallel combination of other contacts.

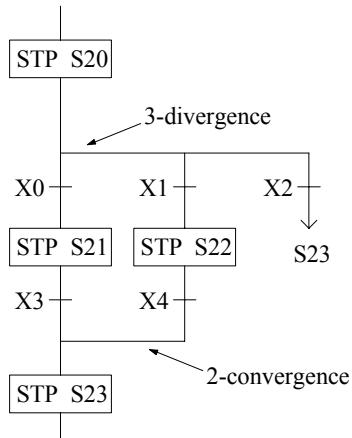
### ③ Simultaneous divergence/convergence



- After X0 is ON, step S20 will simultaneously execute all paths below it, i.e. all S21, S22, S23, and so on, are in action.
- All divergent paths at a convergent point will be executed to the last step (e.g. S30, S31 and S32). When X1 is ON, they can then transfer to S40 for execution.
- The number of divergent paths must be the same as the number of convergent paths. The maximum number of divergence/convergence path is 8.

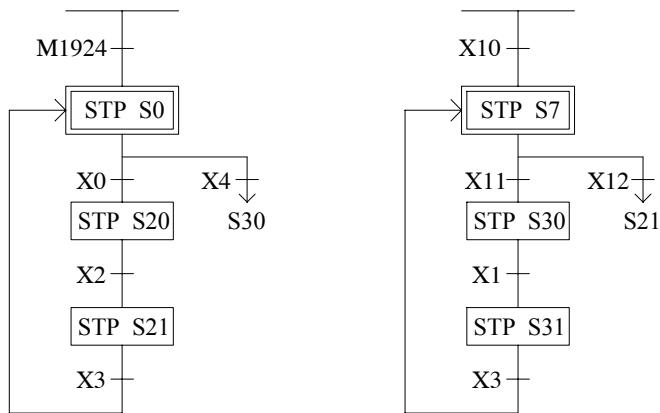
#### ④ Jump

##### a. The same step loop



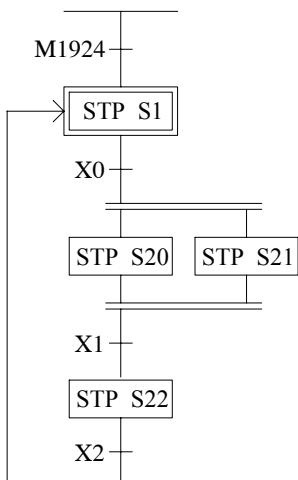
- There are 3 paths below step S20 as shown on the left. Assume that X2 is ON, then the process can jump directly to step S23 to execute without going through the process of selective convergence.
- The execution of simultaneous divergent paths can not be skipped.

##### b. Different step loop

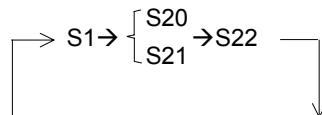


#### ⑤ Closed Loop and Single Cycle

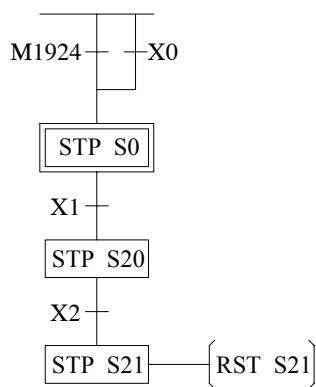
##### a. Closed Loop



- The initial step S1 is ON, endless cycle will be continued afterwards.

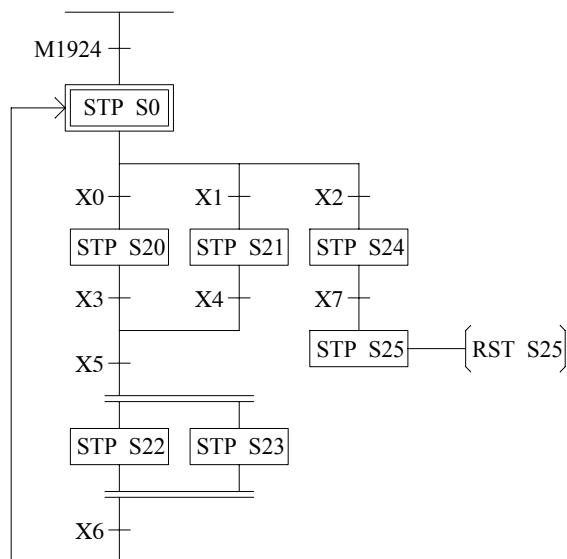


### b. Single Cycle

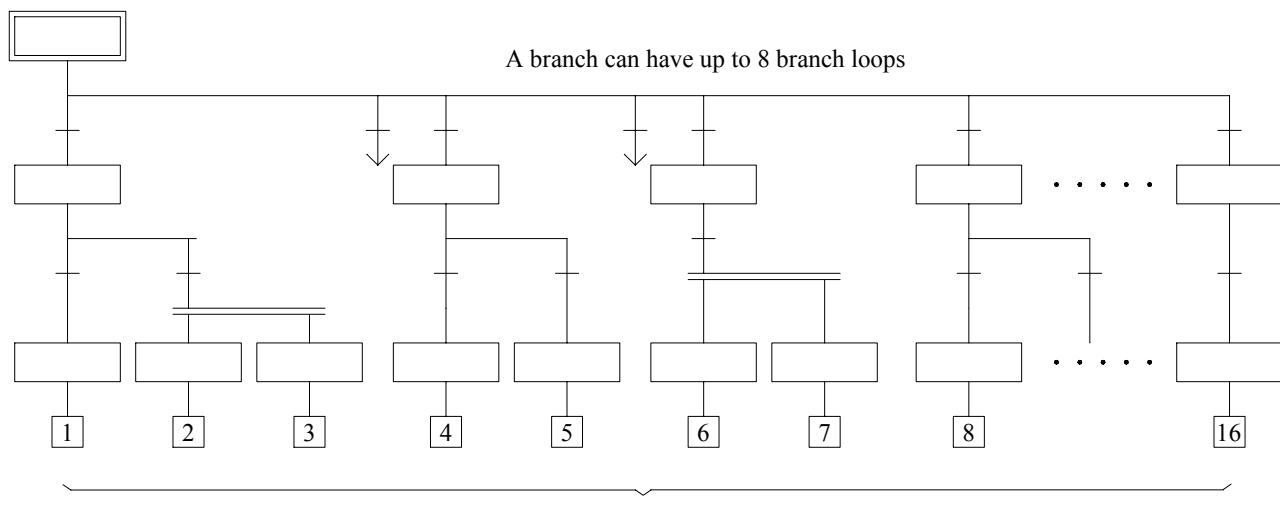


- When step S20 is ON, if X2 is also ON, then “RST S21” instruction will let S21 OFF which will stop the whole step process.

### c. Mixed Process



### ⑥ Combined Application



### 8.3 Introduction of Step Instructions: STP, FROM, TO and STPEND

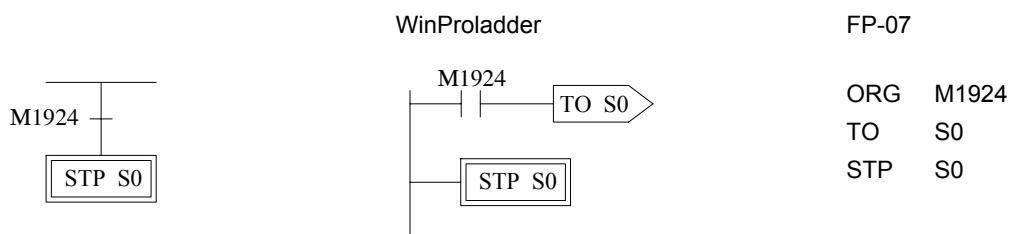
- **STP Sx** :  $S0 \leq Sx \leq S7$  (Displayed in WinProladder)

or

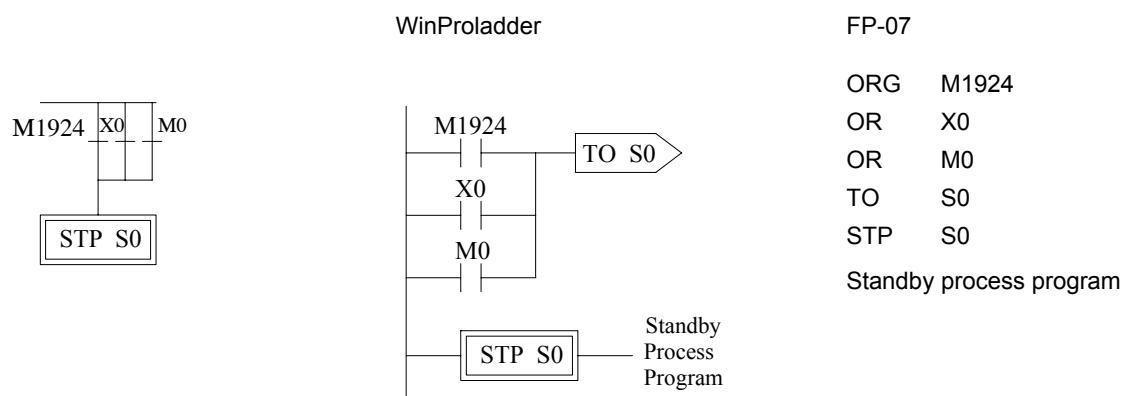
- STP Sx :  $S0 \leq Sx \leq S7$  (Displayed in FP-07)

This instruction is the initial step instruction from where the step control of each machine process can be derived. Up to 8 initial steps can be used in the FB series, i.e. a PLC can make up to 8 process controls simultaneously. Each step process can operate independently or generate results for the reference of other processes.

【Example 1】 Go to the initial step S0 after each start (ON)



【Example 2】 Each time the device starts to run or the manual button is pressed or the device malfunctions, then the device automatically enters the initial step S0 to standby.



【Description】 X0: Manual Button, M0: Abnormal Contact.

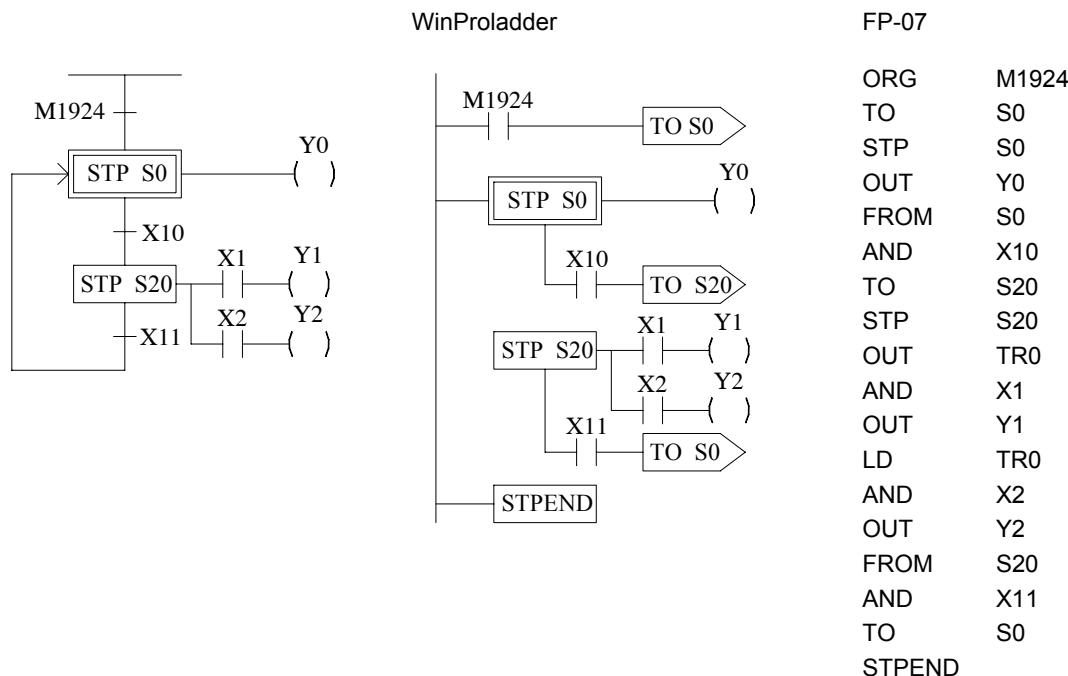
● **STP Sxxx** :  $S20 \leq Sxxx \leq S999$  (Displayed in WinProladder)

or

**STP Sxxx** :  $S20 \leq Sxxx \leq S999$  (Displayed in FP-07)

This instruction is a step instruction, each step in a process represents a step of sequence. If the status of step is ON then the step is active and will execute the ladder program associate to the step.

【Example】



【Description】 1. When ON, the initial step S0 is ON and Y0 is ON.

2. When transfer condition X10 is ON (in actual application, the transferring condition may be formed by the serial or parallel combination of the contacts X, Y, M, T and C), the step S20 is activated. The system will automatically turn S0 OFF in the current scan cycle and Y0 will be reset automatically to OFF.

$$\text{i.e. } X10 \text{ ON} \Rightarrow \begin{cases} S20 \text{ ON} \\ S0 \text{ OFF} \end{cases} \Rightarrow \begin{cases} X1 \text{ ON} & \rightarrow Y1 \text{ ON} \\ X2 \text{ ON} & \rightarrow Y2 \text{ ON} \\ Y0 \text{ OFF} & \end{cases}$$

3. When the transfer condition X11 is ON, the step S0 is ON, Y0 is ON and S20, Y1 and Y2 will turn OFF at the same time.

$$\text{i.e. } X11 \text{ ON} \Rightarrow \begin{cases} S0 \text{ ON} \\ S20 \text{ OFF} \end{cases} \Rightarrow \begin{cases} Y0 \text{ ON} \\ Y1 \text{ OFF} \\ Y2 \text{ OFF} \end{cases}$$

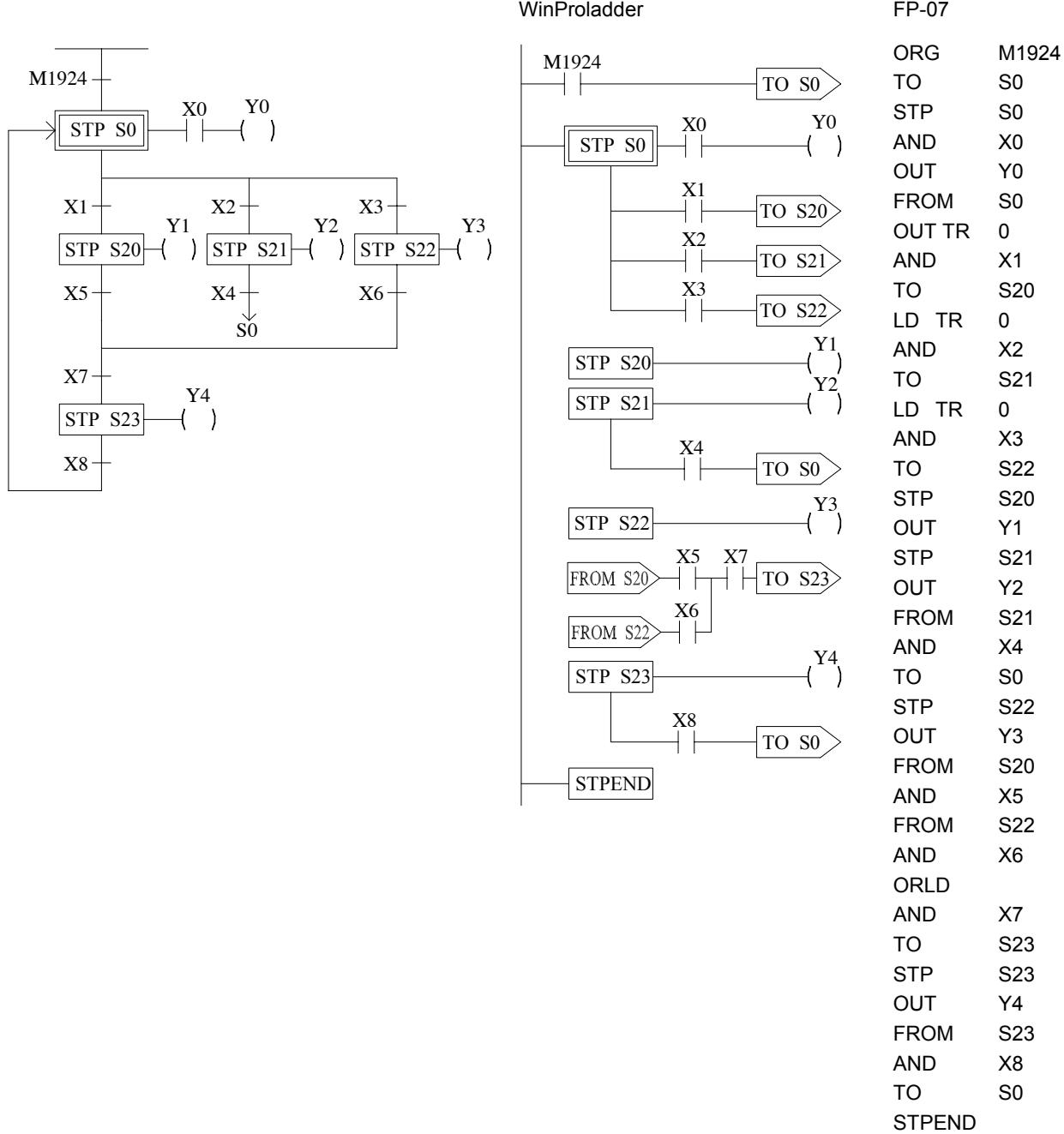
● **FROM Sxxx** :  $S0 \leq Sxxx \leq S999$  (Displayed in WinProladder)

or

**FROM Sxxx** :  $S0 \leq Sxxx \leq S999$  (Displayed in FP-07)

The instruction describes the source step of the transfer, i.e. moving from step Sxxx to the next step in coordination with transfer condition.

**【Example】**



- 【Description】** : 1. When ON, the initial step S0 is ON. If X0 is ON, then Y0 will be ON.
2. When S0 is ON: a. if X1 is ON, then step S20 will be ON and Y1 will be ON.  
b. if X2 is ON, then step S21 will be ON and Y2 will be ON.  
c. if X3 is ON, then step S22 will be ON and Y3 will be ON.  
d. if X1, X2 and X3 are all ON simultaneous, then step S20 will have the priority to be ON first and either S21 or S22 will not be ON.  
e. if X2 and X3 are ON at the same time, then step S21 will have the priority to be ON first and S22 will not be ON.
3. When S20 is ON, if X5 and X7 are ON at the same time, then step S23 will be ON, Y4 will be ON and S20 and Y1 will be OFF.
4. When S21 is ON, if X4 is ON, then step S0 will be ON and S21 and Y2 will be OFF.
5. When S22 is ON, if X6 and X7 are ON at the same time, then step S23 will be ON, Y4 will be ON and S22 and Y3 will be OFF.
6. When S23 is ON, if X8 is ON, then step S0 will be ON and S23 and Y4 will be OFF.

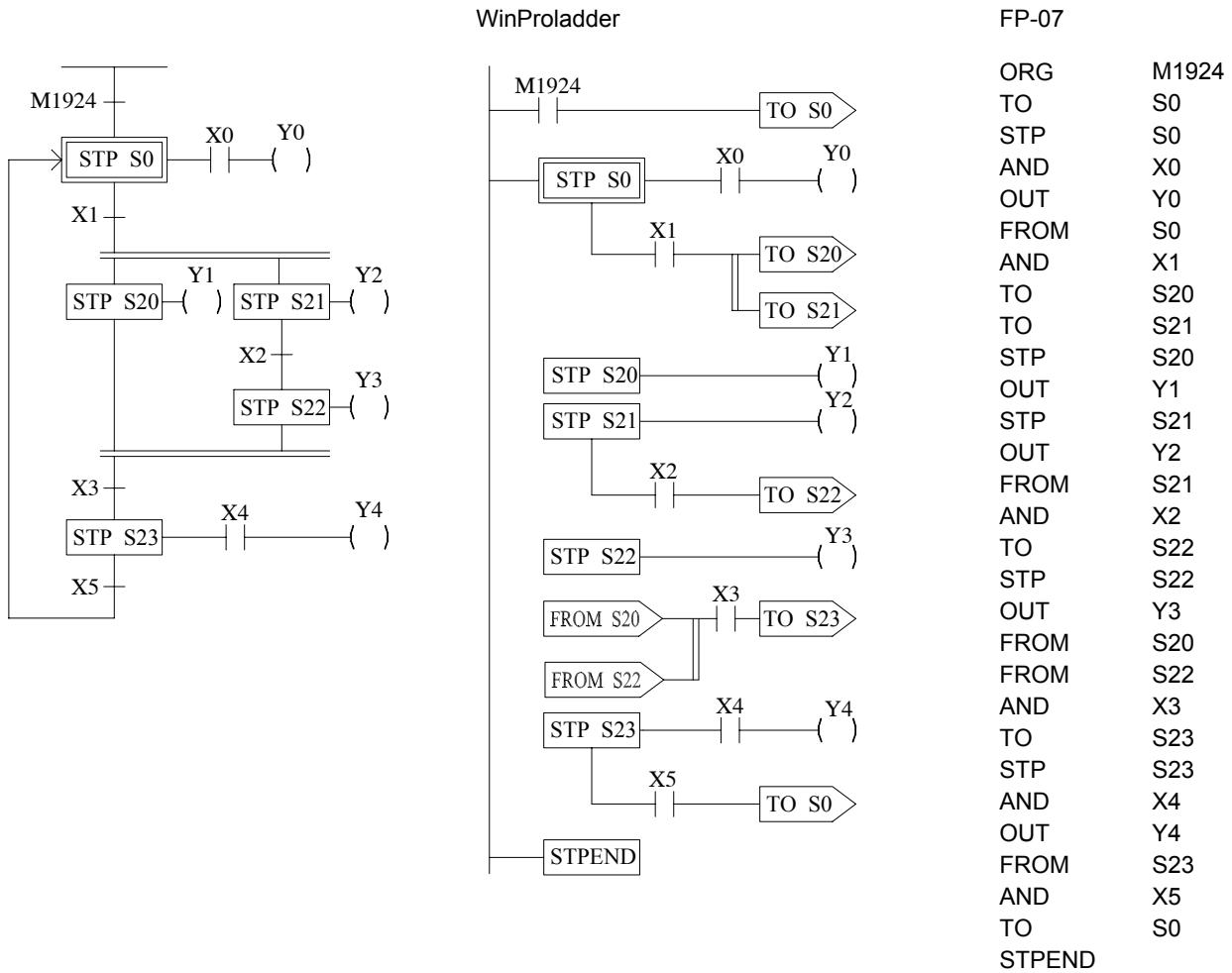
● TO Sxxx :  $S0 \leq Sxxx \leq S999$  (Displayed in WinProladder)

or

TO Sxxx :  $S0 \leq Sxxx \leq S999$  (Displayed in FP-07)

This instruction describes the step to be transferred to.

**【Example】**



**【Description】** : 1. When ON, the initial step S0 is ON. If X0 is ON, then Y0 will be ON.

2. When S0 is ON: if X1 is ON, then steps S20 and S21 will be ON simultaneously and Y1 and Y2 will also be ON.
3. When S21 is ON: if X2 is ON, then step S22 will be ON, Y3 will be ON and S21 and Y2 will be OFF.
4. When S20 and S22 are ON at the same time and the transferring condition X3 is ON, then step S23 will be ON (if X4 is ON, then Y4 will be ON) and S20 and S22 will automatically turn OFF and Y1 and Y3 will also turn OFF.
5. When S23 is ON: if X5 is ON, then the process will transfer back to the initial step, i.e. S0 will be ON and S23 and Y4 will be OFF.

● **STPEND** : ( Displayed in WinProladder )

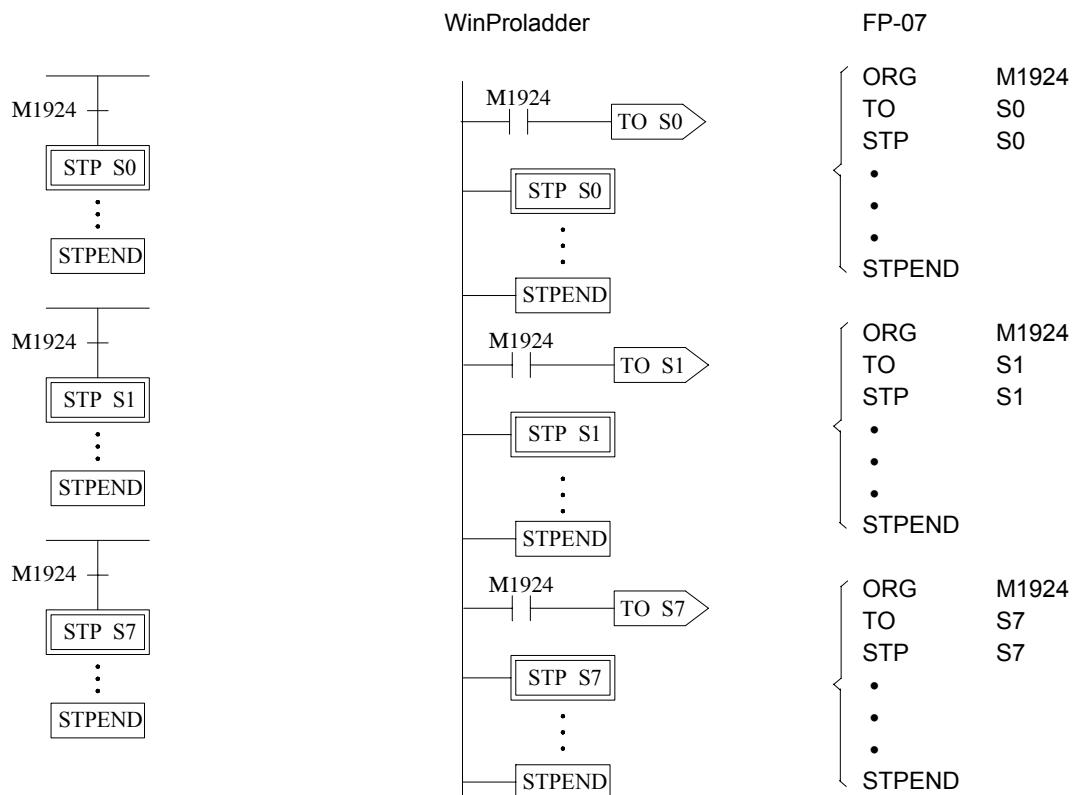
or

**STPEND** : ( Displayed in FP-07 )

This instruction represents the end of a process. It is necessary to include this instruction so all processes can be operated correctly.

A PLC can have up to 8 step processes (S0~S7) and is able to control them simultaneously. Therefore, up to 8 STPEND instructions can be obtained.

#### 【Example】

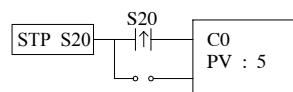


【Description】 When ON, the 8 step processes will be active simultaneously.

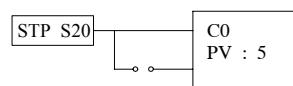
## 8.4 Notes for Writing a Step Ladder Diagram

### 【Notes】

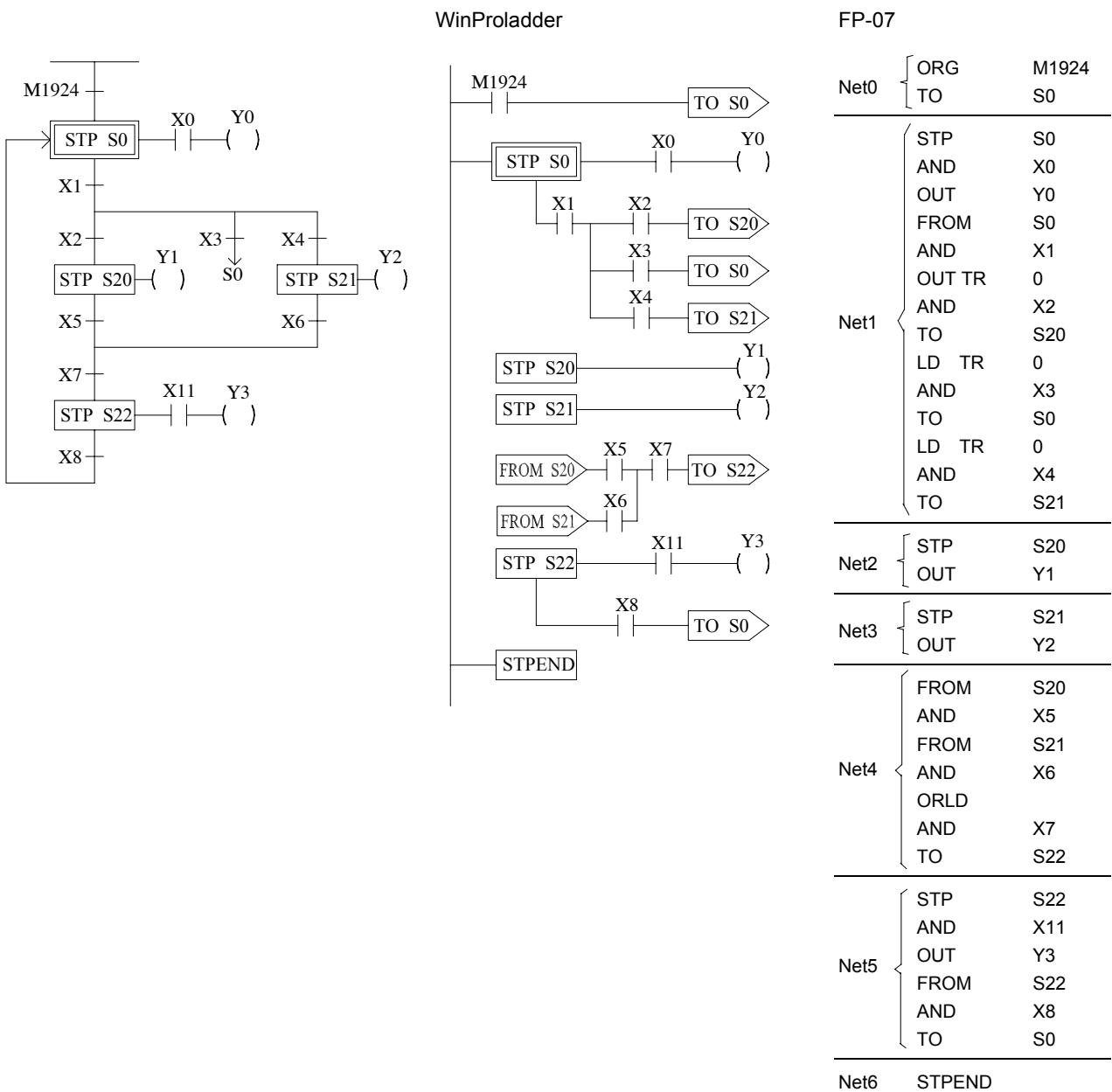
- In actual applications, the ladder diagram can be used together with the step ladder.
- There are 8 steps, S0~S7, that can be used as the starting point and are called the “initial steps”.
- When PLC starts operating, it is necessary to activate the initial step. The M1924 (the first scan ON signal) provided by the system may be used to activate the initial step.
- Except the initial step, the start of any other steps must be driven by other step.
- It is necessary to have an initial step and the final STPEND instruction in a step ladder diagram to complete a step process program.
- There are 980 steps, S20~S999, available that can be used freely. However, used numbers cannot be repeated. S500~S999 are retentive(The range can be modified by users), can be used if it is required to continue the machine process after power is off.
- Basically a step must consists of three parts which are control output, transition conditions and transition targets.
- MC and SKP instructions cannot be used in a step program and the sub-programs. It's recommended that JMP instruction should be avoided as much as possible.
- If the output point is required to stay ON after the step is divergent to other step, it is necessary to use the SET instruction to control the output point and use RST instruction to clear the output point to OFF.
- Looking down from an initial step, the maximum number of horizontal paths is 16. However, a step is only allowed to have up to 8 branch paths.
- When M1918=0 (default) , if a PULSE type function instruction is used in master control loop (FUN 0) or a step program, it is necessary to connect a TU instruction before the function instruction. For example,



When M1918=1, the TU instruction is not required, e.g.:



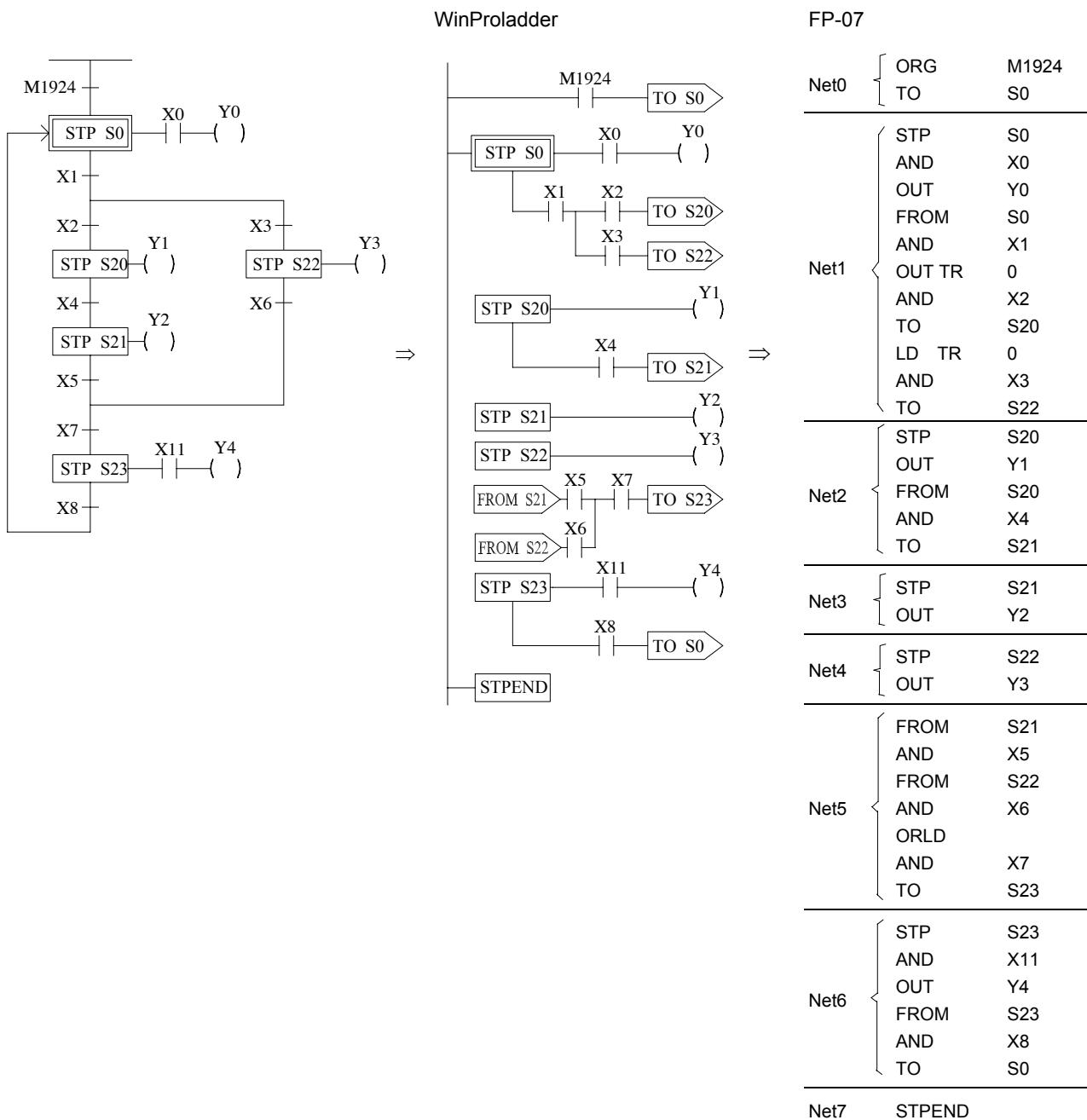
**Example 1**



## Description

1. Input the condition to initial step S0
  2. Input the S0 and the divergent conditions of S20, S0 and S21
  3. Input the S20
  4. Input the S21
  5. Input the convergence of S20 and S21
  6. Input the S22

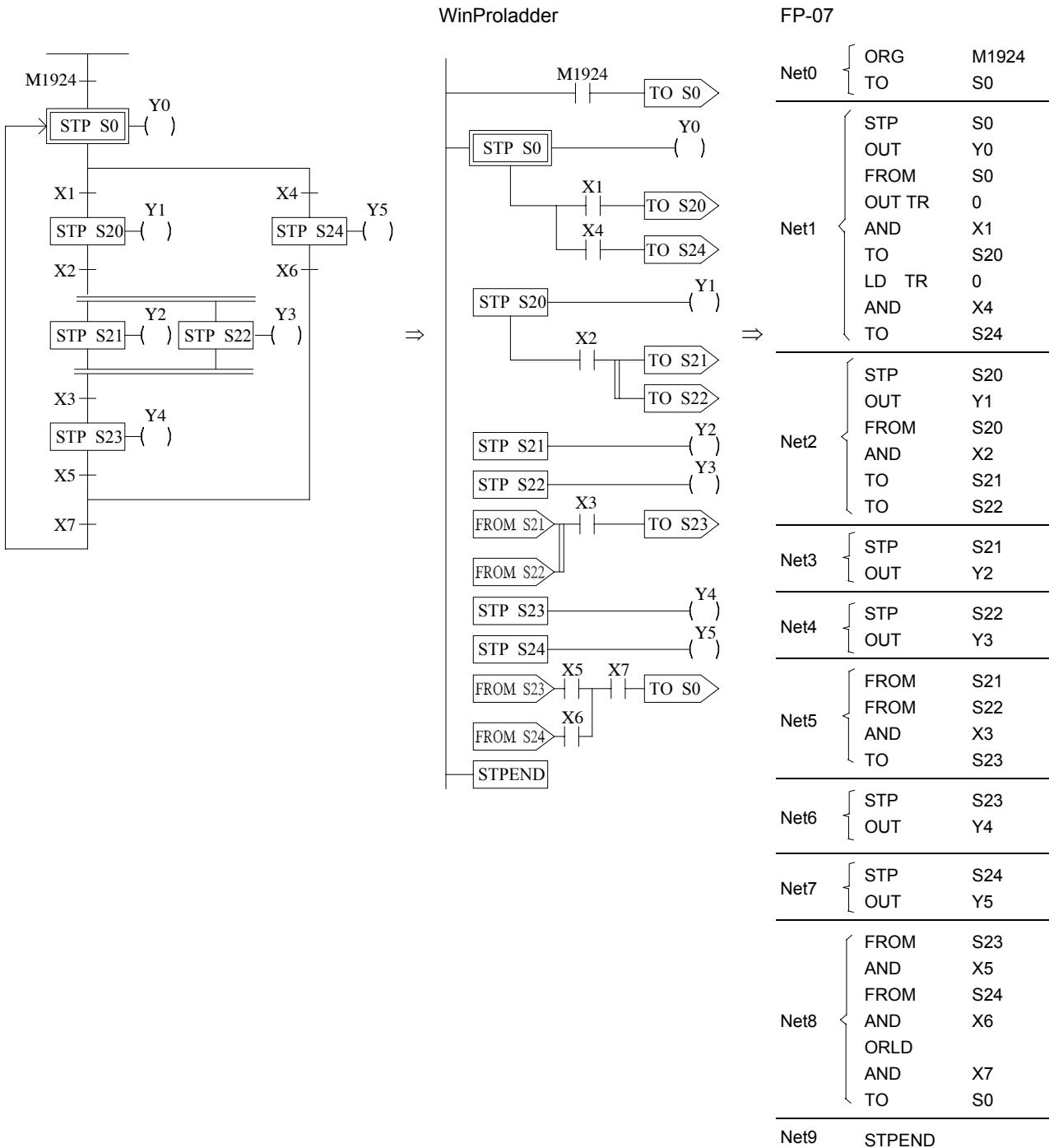
## Example 2



### Description

1. Input the condition to initial step S0
2. Input the S0 and the divergent condition of S20 and S22
3. Input the S20
4. Input the S21
5. Input the S22
6. Input the convergence of S21 and S22
7. Input the S23

### Example 3

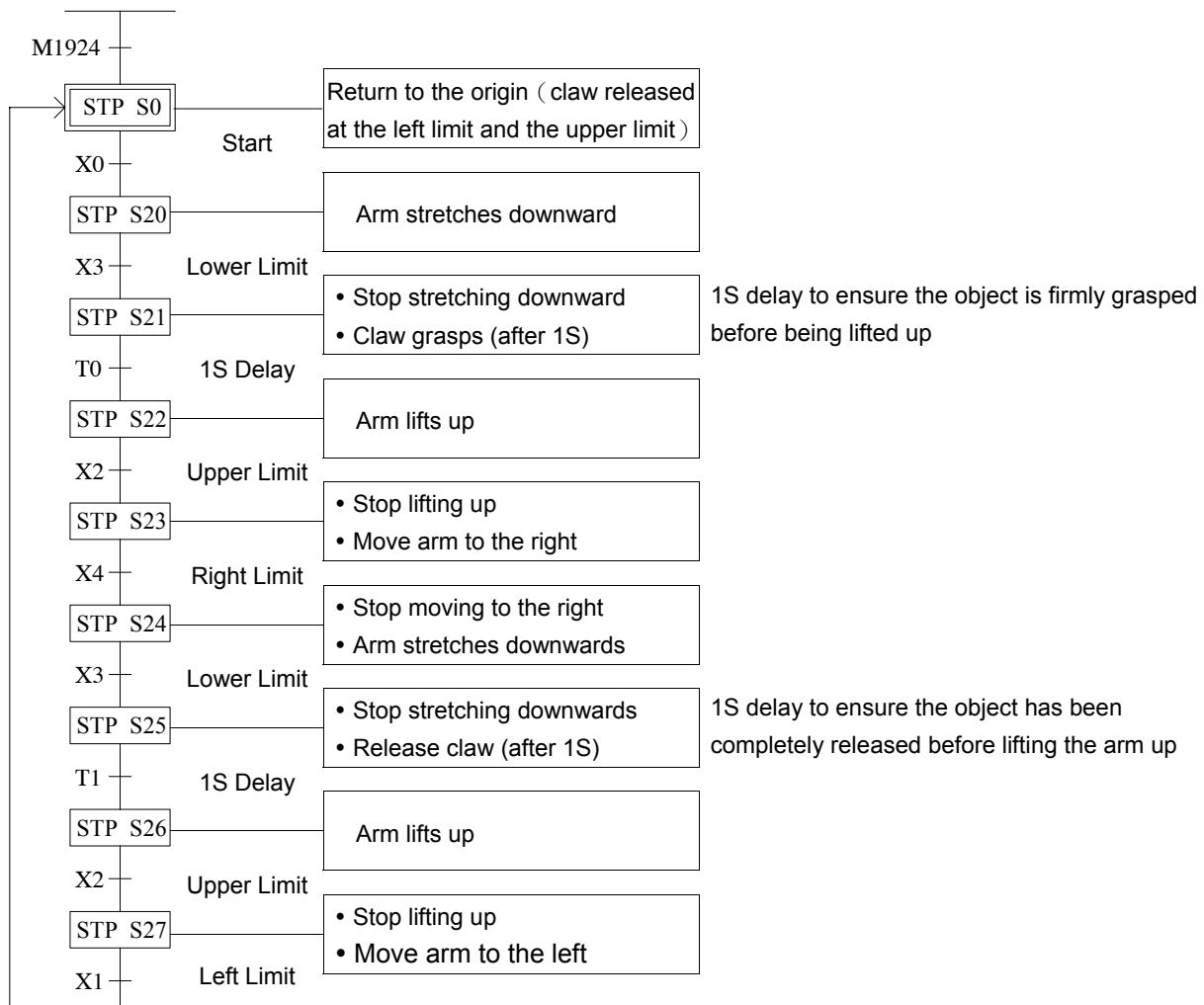
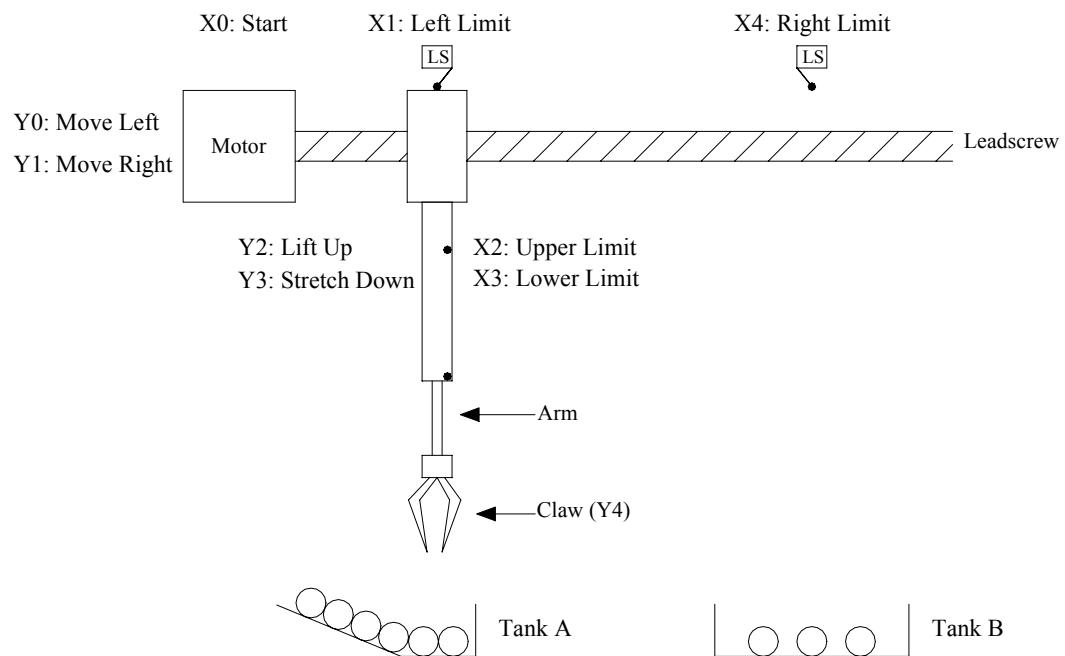


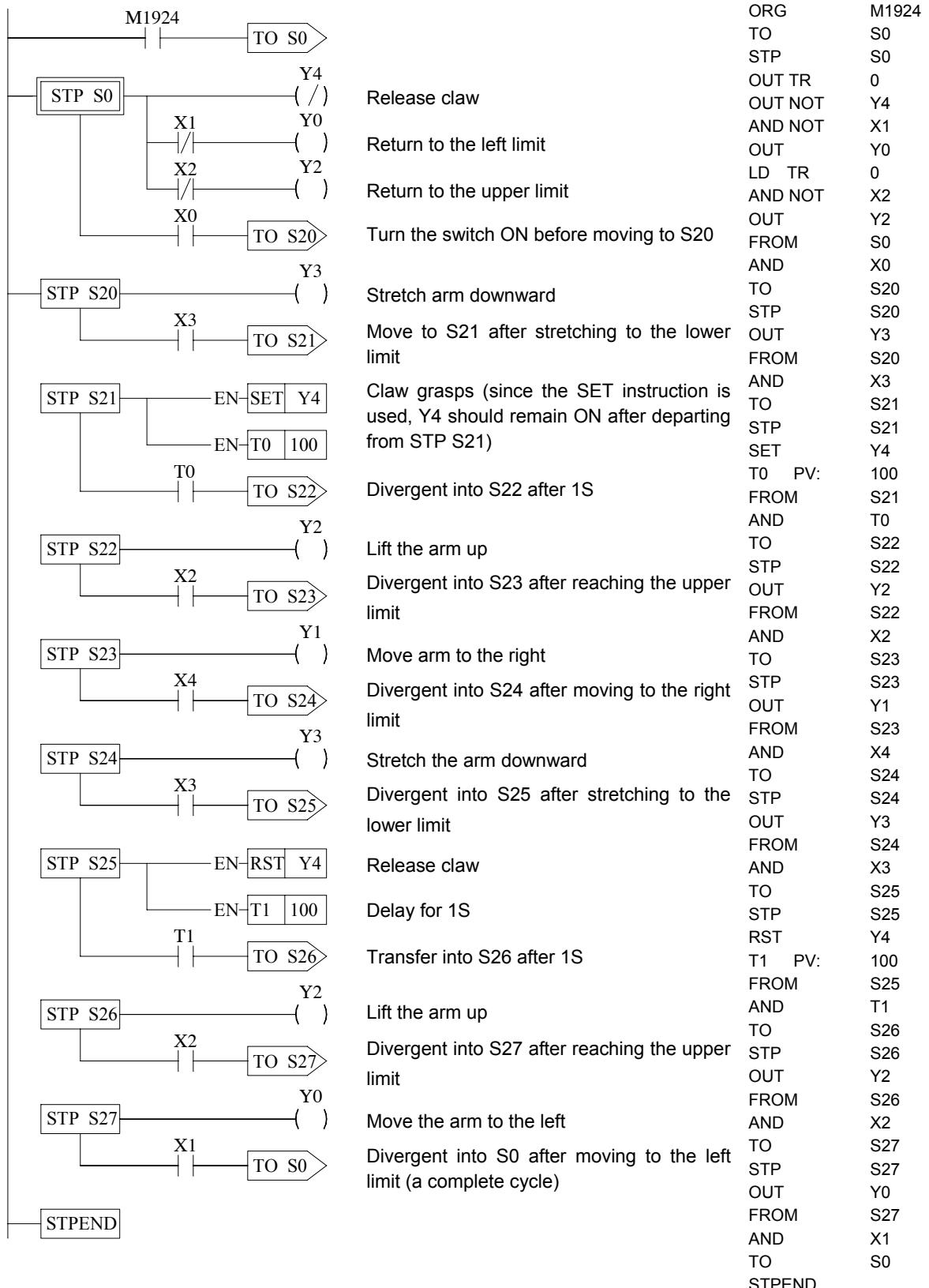
#### Description

1. Input the condition to initial step S0
2. Input the S0 and the divergences of S20 and S24
3. Input the S20
4. Input the S20 and the divergences of S21 and S22
5. Input the S21
6. Input the S22
7. Input the convergences of S21 and S22
8. Input the S23
9. Input the S24
10. Input the convergences of S23 and S24

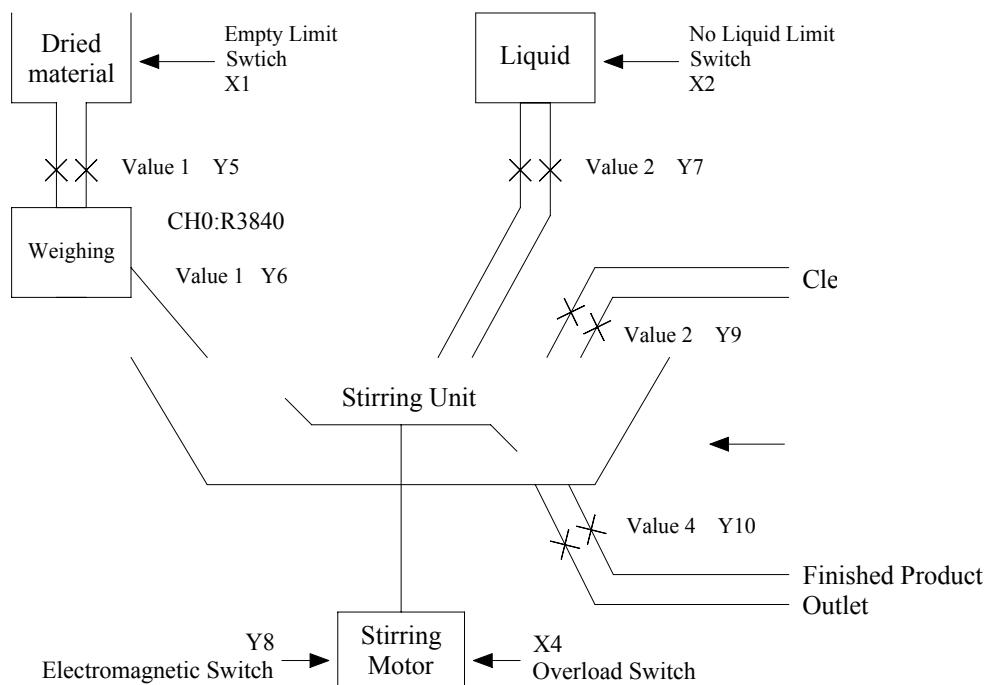
## 8.5 Application Examples

**Example 1** Grasp an object from tank A and put it in Tank B





**Example 2** Liquid Stirring Process



- ♦ Input Points: Empty limit switch X1  
No liquid limit switch X2  
Empty limit switch X3  
Over-load switch X4  
Warning clear button X5  
Start button X6  
Water washing button X7

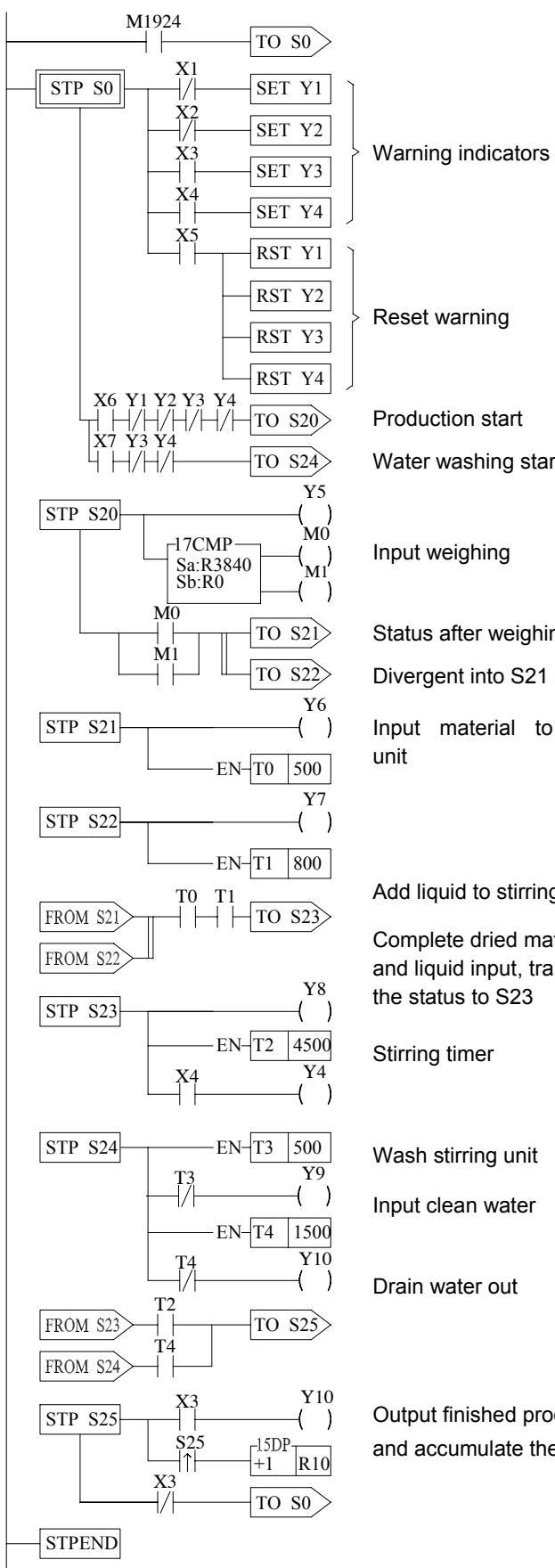
- ♦ Warning Indicators: Empty dried material Y1  
Insufficient liquid Y2  
Empty stirring unit Y3  
Motor over-load Y4

- ♦ Output Points: Dried material inlet valve Y5  
Dried material inlet valve Y6  
Liquid inlet valve Y7  
Motor start electromagnetic valve Y8  
Clean water inlet valve Y9  
Finished product outlet valve Y10

- ♦ Weighing Output: CH0 ( R3840 )

- ♦ M1918=0

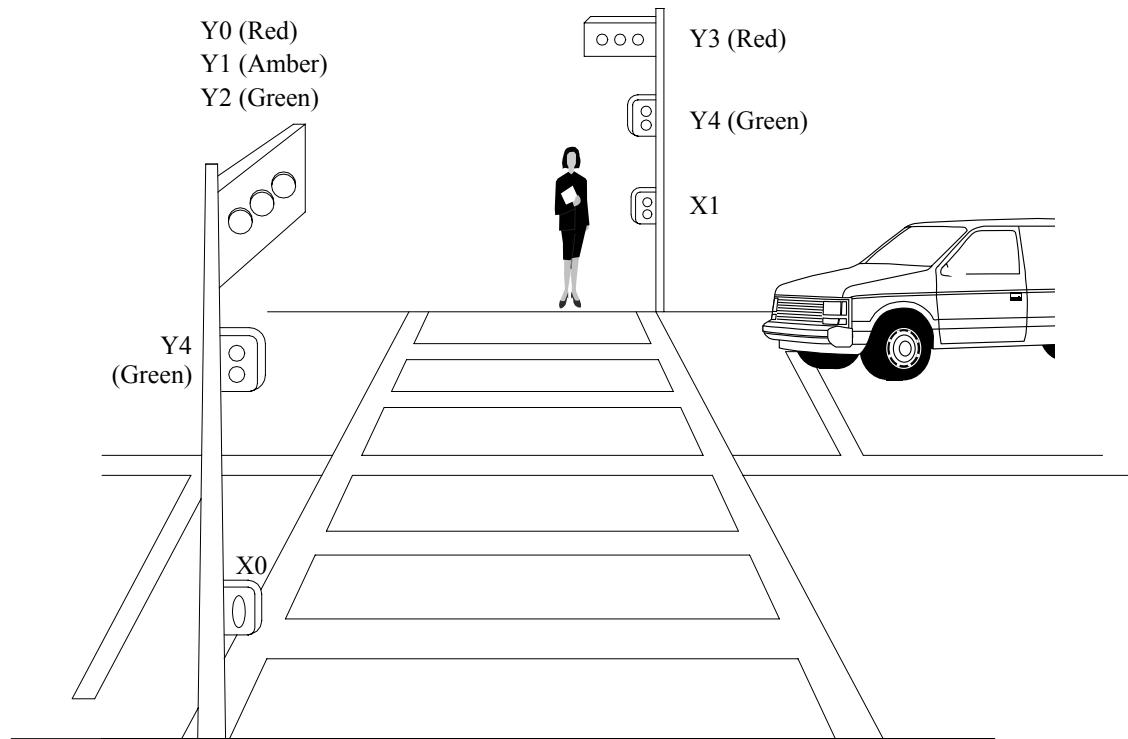
WinProladder



FP-07

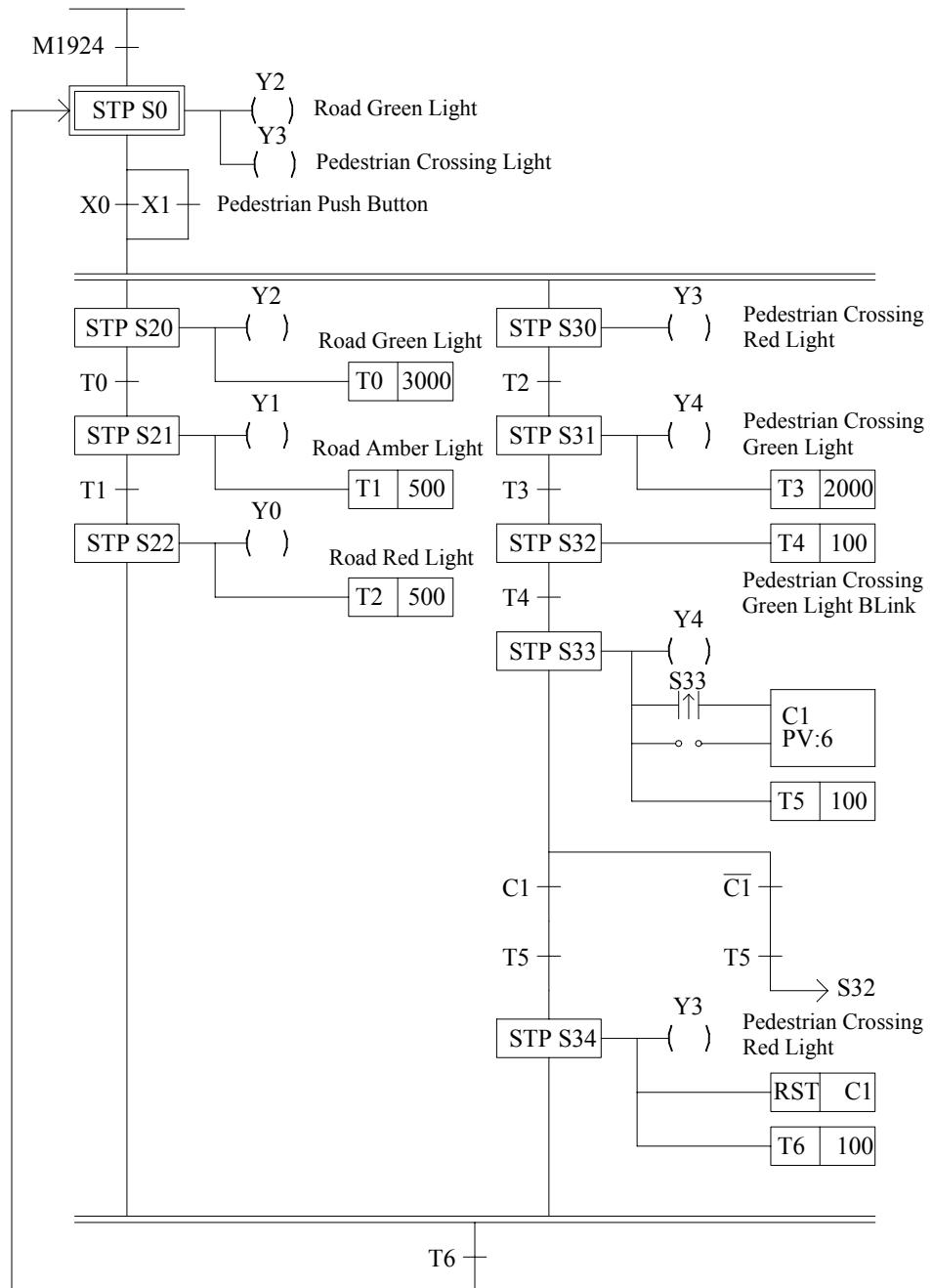
ORG	M1924	STP	S22
TO	S0	OUT	Y7
-----		T1	PV: 800
STP	S0	FROM	S21
OUT TR	0	AND NOT	X1
AND NOT	X1	FROM	S22
SET	Y1	AND	T0
LD TR	0	AND	T1
AND NOT	X2	TO	S23
SET	Y2	-----	
LD TR	0	STP	S23
AND	X3	OUT TR	0
SET	Y3	OUT	Y8
LD TR	0	LD TR	0
AND	X4	T2	PV: 4500
SET	Y4	AND	X4
LD TR	0	OUT	Y4
AND	X5	-----	
RST	Y1	STP	S24
RST	Y2	OUT TR	0
RST	Y3	T3	PV: 500
RST	Y4	LD TR	0
AND NOT	T3	-----	
FROM	S0	OUT	Y9
OUT TR	1	LD TR	0
AND	X6	T4	PV: 1500
AND NOT	Y1	LD TR	0
AND NOT	Y2	AND NOT	T4
AND NOT	Y3	OUT	Y10
AND NOT	Y4	FROM	S23
TO	S20	AND	T2
LD TR	1	FROM	S24
AND	X7	AND	T4
AND NOT	Y3	ORLD	
AND NOT	Y4	TO	S25
TO	S24	-----	
STP	S20	STP	S25
OUT TR	0	OUT TR	0
OUT	Y5	AND	X3
FUN	17	OUT	Y10
Sa:R3840		LD TR	0
Sb:R0		AND TU	S25
FO	0	FUN	15DP
OUT	M0	D:R10	
FO	1	FROM	S25
OUT	M1	AND NOT	X3
FROM	S20	TO	S0
LD	M0	-----	
OR	M1	STPEND	
ANDLD			
TO	S21		
TO	S22		
-----			
STP	S21		
OUT	Y6		
T0	PV: 500		

**Example 3** Pedestrian Crossing Lights



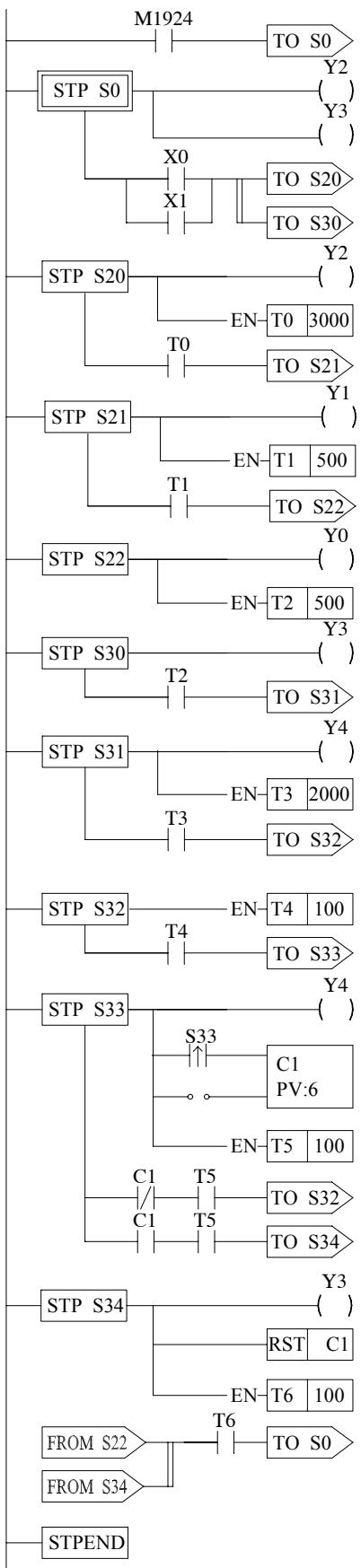
- ♦ Input Points: Pedestrian Push Button X0  
Pedestrian Push Button X1
- ♦ Output Points: Road Red Light Y0  
Road Amber light Y1  
Road Green Light Y2  
Pedestrian Crossing Red Light Y3  
Pedestrian Crossing Green Light Y4
- ♦ M1918=0

● Pedestrian Crossing Lights Control Process Diagram



● Pedestrian Crossing Lights Control Program

WinProladder



FP-07

```

    ORG M1924
    TO S0
    -----
    STP S0
    OUT Y2
    OUT Y3
    FROM S0
    LD X0
    OR X1
    ANDLD
    TO S20
    TO S30
    -----
    STP S20
    OUT Y2
    T0 PV: 3000
    FROM S20
    AND T0
    TO S21
    -----
    STP S21
    OUT Y1
    T1 PV: 500
    FROM S21
    AND T1
    TO S22
    -----
    STP S22
    OUT Y0
    T2 PV: 500
    FROM S22
    AND T2
    TO S31
    -----
    STP S30
    OUT Y3
    FROM S30
    AND T2
    TO S31
    -----
    STP S31
    OUT Y4
    T3 PV: 2000
    FROM S31
    AND T3
    TO S32
    -----
    STPEND
  
```

## 8.6 Syntax Check Error Codes for Step Instruction

The error codes for the usage of step instruction are as follows:

- E51 : TO(S0-S7) must begin with ORG instruction.
- E52 : TO(S20-S999) can't begin with ORG instruction.
- E53 : TO instruction without matched FROM instruction.
- E54 : To instruction must comes after TO, AND, OR, ANDLD or ORLD instruction.
- E56 : The instructions before FROM must be AND, OR, ANDLD or ORLD
- E57 : The instruction after FROM can't be a coil or a function
- E58 : Coil or function must before FROM while in STEP network.
- E59 : More than 8 TO# at same network.
- E60 : More than 8 FROM# at same network.
- E61 : TO(S0-S7) must locate at first row of the network.
- E62 : A contact occupies the location for TO instruction.
- E72 : Duplicated TO Sxx instruction.
- E73 : Duplicated STP sxx instruction.
- E74 : Duplicated FROM sxx instruction.
- E76 : STP(S0~S7) without a matched STPEND or STPEND without a matched STP(S0~S7).
- E78 : TO(S20~S999), STP (S20~S999) or FROM instructions comes before or without STP(S0~S19).
- E79 : STP Sxx or FROM Sxx instructions comes before or without TO Sxx.
- E80 : FROM Sxx instruction comes before or without STP Sxx.
- E81 : The max. level of branches must <=16.
- E82 : The max. no. of branches with same level must <=16.
- E83 : Not place the step instruction with TO->STP->FROM sequence.
- E84 : The definition of STP# sequence not follow the TO# sequence.
- E85 : Convergence do not match the corresponding divergence.
- E86 : Illegal usage of STP or FROM before convergent with TO instruction.
- E87 : STP# or FROM# comes before corresponding TO#.
- E88 : During this branch, STP# or FROM# comes before the corresponding TO#.
- E89 : FROM# comes before corresponding TO# or STP#.
- E90 : Invalid To# usage in the simultaneous branch.
- E91 : Flow control function can not be used in the step ladder region.

## Chapter 9 Advanced Application Instructions

● Arithmetical operation instructions	(FUN23~30) .....	9-2	~ 9-9
● Logical operation instructions	(FUN35~36) .....	9-10	~ 9-11
● Comparison instructions	(FUN37) .....	9-12	
● Data movement instructions	(FUN40~48) .....	9-13	~ 9-21
● Shifting/Rotating instructions	(FUN51~54) .....	9-22	~ 9-25
● Code conversion instructions	(FUN57~64) .....	9-26	~ 9-38
● Flow control instructions	(FUN65~71) .....	9-39	~ 9-46
● Temperature control instructions 1	(FUN72~73) .....	9-47	~ 9-48
● I/O instructions	(FUN74~84) .....	9-49	~ 9-62
● Temperature control instructions 2	(FUN85~86) .....	9-63	~ 9-64
● Cumulative timer instructions	(FUN87~89) .....	9-65	~ 9-66
● Watchdog timer instructions	(FUN90~91) .....	9-67	~ 9-68
● High speed counting/timing instructions	(FUN92~93) .....	9-69	~ 9-70
● Report printing instructions	(FUN94) .....	9-71	~ 9-72
● Slow up/Slow down instructions	(FUN95) .....	9-73	~ 9-74
● Communication instructions	(FUN96~97) .....	9-75	~ 9-76
● Table instructions	(FUN100~113) .....	9-77	~ 9-94
● Matrix instructions	(FUN120~130) .....	9-95	~ 9-106
● NC position instructions	(FUN140~143) .....	9-107	~ 9-110
● Interrupt control instructions	(FUN145~146) .....	9-111	~ 9-112

## Arithmetical operation instructions

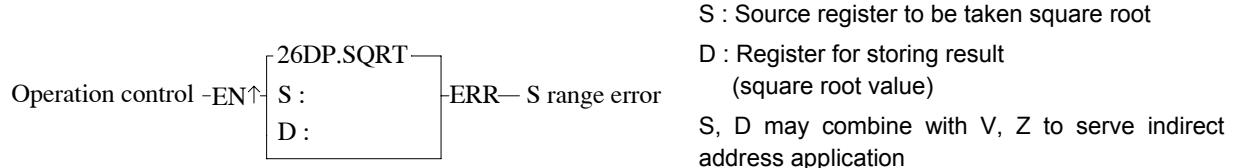
FUN 23 <b>P</b> DIV48	48-BIT DIVISION	FUN 23 <b>P</b> DIV48																																									
Operation control-EN↑	<p>23P.DIV48</p> <p>Sa : Starting register of dividend Sb : Starting register of divisor D : Starting register for storing the division result (quotient) Sa , Sb , can combine V,Z for index addressing.</p>																																										
	<table border="1"> <thead> <tr> <th>Range</th> <th>HR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td rowspan="2">Oper- and</td> <td>R0   R3839</td> <td>R3904   R3967</td> <td>R3968   R4167</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td>V   Z</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Sa</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> </tr> <tr> <td>Sb</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> </tr> <tr> <td>D</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td>○</td> </tr> </tbody> </table>	Range	HR	OR	SR	ROR	DR	XR	Oper- and	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	V   Z							Sa	○	○	○	○	○	○	Sb	○	○	○	○	○	○	D	○	○	○*	○*	○	○	
Range	HR	OR	SR	ROR	DR	XR																																					
Oper- and	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	V   Z																																					
Sa	○	○	○	○	○	○																																					
Sb	○	○	○	○	○	○																																					
D	○	○	○*	○*	○	○																																					
	<ul style="list-style-type: none"> <li>When operation control “EN”=1 or “EN ↑ ” (<b>P</b> instruction) changes from 0→1, will perform the 42 bits division operation. Dividend and divisor are each formed by three consecutive registers starting by Sa and Sb respectively. If the result is zero, ‘D=0’ output will be set to 1. If divisor is zero then the ‘ERR’ will be set to 1 and the resultant register will keep unchanged.</li> <li>All operands involved in this function are all 42 bits, so Sa, Sb and D are all comprised by 3 consecutive registers.</li> </ul>																																										
Example: 48-bit division	In this example dividend formed by register R2, R1, R0 will be divided by divisor formed by register R5, R4, R3. The quotient will store in R8, R7, and R6.																																										
	<p>23P.DIV48</p> <p>X0 — EN</p> <p>Sa: R 0      D=0— Sb: R 3      ERR— D : R 6</p>																																										
	<p style="text-align: center;">÷</p> <table> <tr> <td style="vertical-align: top; padding-right: 20px;">Sa</td> <td style="border: 1px solid black; padding: 5px; text-align: center;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 33.33%;">R2</td><td style="width: 33.33%;">R1</td><td style="width: 33.33%;">R0</td></tr> <tr><td colspan="3" style="text-align: center;">2147483647</td></tr> </table> </td> </tr> <tr> <td style="vertical-align: top; padding-right: 20px;">Sb</td> <td style="border: 1px solid black; padding: 5px; text-align: center;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 33.33%;">R5</td><td style="width: 33.33%;">R4</td><td style="width: 33.33%;">R3</td></tr> <tr><td colspan="3" style="text-align: center;">1234567</td></tr> </table> </td> </tr> <tr> <td colspan="2" style="text-align: right; padding-top: 20px;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 33.33%;">R8</td><td style="width: 33.33%;">R7</td><td style="width: 33.33%;">R6</td></tr> <tr><td colspan="3" style="text-align: center;">1739</td></tr> </table> <p style="text-align: center;">Quotient</p> </td> </tr> </table>	Sa	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 33.33%;">R2</td><td style="width: 33.33%;">R1</td><td style="width: 33.33%;">R0</td></tr> <tr><td colspan="3" style="text-align: center;">2147483647</td></tr> </table>	R2	R1	R0	2147483647			Sb	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 33.33%;">R5</td><td style="width: 33.33%;">R4</td><td style="width: 33.33%;">R3</td></tr> <tr><td colspan="3" style="text-align: center;">1234567</td></tr> </table>	R5	R4	R3	1234567			<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 33.33%;">R8</td><td style="width: 33.33%;">R7</td><td style="width: 33.33%;">R6</td></tr> <tr><td colspan="3" style="text-align: center;">1739</td></tr> </table> <p style="text-align: center;">Quotient</p>		R8	R7	R6	1739																				
Sa	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 33.33%;">R2</td><td style="width: 33.33%;">R1</td><td style="width: 33.33%;">R0</td></tr> <tr><td colspan="3" style="text-align: center;">2147483647</td></tr> </table>	R2	R1	R0	2147483647																																						
R2	R1	R0																																									
2147483647																																											
Sb	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 33.33%;">R5</td><td style="width: 33.33%;">R4</td><td style="width: 33.33%;">R3</td></tr> <tr><td colspan="3" style="text-align: center;">1234567</td></tr> </table>	R5	R4	R3	1234567																																						
R5	R4	R3																																									
1234567																																											
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 33.33%;">R8</td><td style="width: 33.33%;">R7</td><td style="width: 33.33%;">R6</td></tr> <tr><td colspan="3" style="text-align: center;">1739</td></tr> </table> <p style="text-align: center;">Quotient</p>		R8	R7	R6	1739																																						
R8	R7	R6																																									
1739																																											

FUN 24 <b>D P</b> SUM	SUM (Summation of block data)	FUN 24 <b>D P</b> SUM																																																																																																																								
<p>Operation control-EN↑</p> <p>24DP.SUM</p> <p>S : R0</p> <p>N : 6</p> <p>D : R100</p>	<p>S : Starting number of source register</p> <p>N : Number of registers to be summed (successive N data units starting from S)</p> <p>D : The register which stored the result (summation)</p> <p>S, N, D, can associate with V, Z index register to serve the indirect addressing application.</p>	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Range \ Operand</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>WX0</td> <td>WX0</td> <td>WY0</td> <td>WM0</td> <td>WS0</td> <td>T0</td> <td>C0</td> <td>R0</td> <td>R3840</td> <td>R3904</td> <td>R3968</td> <td>R5000</td> <td>D0</td> <td>1</td> <td>V</td> </tr> <tr> <td>WX240</td> <td> </td> <td> </td> <td> </td> <td>WS984</td> <td>T255</td> <td>C255</td> <td>R3839</td> <td>R3903</td> <td>R3967</td> <td>R4167</td> <td>R8071</td> <td> </td> <td> </td> <td>Z</td> </tr> <tr> <td>WM1896</td> <td>WX240</td> <td>WY240</td> <td>WM1896</td> <td> </td> <td> </td> <td> </td> <td>R3839</td> <td>R3903</td> <td> </td> <td> </td> <td> </td> <td>D3071</td> <td>511</td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td>WS984</td> <td>T255</td> <td>C255</td> <td> </td> <td> </td> <td> </td> <td> </td> <td> </td> <td></td> <td></td> <td></td> </tr> <tr> <td>S</td> <td>○</td> </tr> <tr> <td>N</td> <td>○</td> </tr> <tr> <td>D</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> <td>○</td> </tr> </tbody> </table>	Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	WX0	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	1	V	WX240				WS984	T255	C255	R3839	R3903	R3967	R4167	R8071			Z	WM1896	WX240	WY240	WM1896				R3839	R3903				D3071	511						WS984	T255	C255									S	○	○	○	○	○	○	○	○	○	○	○	○	○	○	N	○	○	○	○	○	○	○	○	○	○	○	○	○	○	D		○	○	○	○	○	○		○	○*	○*	○		○
Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																																																												
WX0	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	1	V																																																																																																												
WX240				WS984	T255	C255	R3839	R3903	R3967	R4167	R8071			Z																																																																																																												
WM1896	WX240	WY240	WM1896				R3839	R3903				D3071	511																																																																																																													
				WS984	T255	C255																																																																																																																				
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																												
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																												
D		○	○	○	○	○	○		○	○*	○*	○		○																																																																																																												
<ul style="list-style-type: none"> <li>● When operation control “EN”=1 or “EN ↑ ” (<b>P</b> instruction) changes from 0→1, it puts the successive N units of 16bit or 32 bit (<b>D</b> instruction) registers for addition calculation to get the summation, and stores the result into the register which is designated by D.</li> <li>● When the value of N is 0 or greater than 511, the operation will not be performed.</li> <li>● Communication port1 or port2 can be used to serve as a general purpose ASCII communication interface. If the data error detecting method is Check-Sum, this instruction can be used to generate the sum value for sending data or to use this instruction to check if the received data is error or not.</li> </ul> <p>⟨ Example 1 ⟩ When M1 changes from OFF→ON, following instruction will calculates the summation for 16-bit data.</p> <p>M1 → EN↑</p> <p>24P.SUM</p> <p>S : R0</p> <p>N : 6</p> <p>D : R100</p> <p>R0=0030H R1=0031H R2=0032H R3=0033H R4=0034H R5=0035H</p> <p>→ R100=012FH</p> <p>● The left illustrates that 6 16-bit registers starting from R0 is calculated for summation, and the result is stored into the R100 register.</p>	<p>M1 → EN</p> <p>24D.SUM</p> <p>S : R0</p> <p>N : 3</p> <p>D : R100</p> <p>R1 , R0=00310030H R3 , R2=00330032H R5 , R4=00410039H</p> <p>→ R101 , R100=00A5009BH</p> <p>● The left illustrates that three 32-bit registers starting from DR0, is calculated for their summation, and the result is stored into the DR100 register.</p>																																																																																																																									

## Arithmetical operation instructions

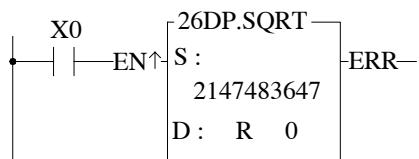
FUN 25 <b>D P</b> MEAN	MEAN (Average of the block data)	FUN 25 <b>D P</b> MEAN																																																																										
<p>Operation control-EN↑</p>	<p>S : Source register number N : Number of registers to be averaged (N units of successive registers starting from S) D : Register number for storing result (mean value) The S, N, D may combine with V, Z to serve indirect address application</p>																																																																											
<table border="1"> <thead> <tr> <th>Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>Oper- and</td> <td>WX0   WX240</td> <td>WY0   WY240</td> <td>WM0   WM1896</td> <td>WS0   WS984</td> <td>T0   T255</td> <td>C0   C255</td> <td>R0   R3839</td> <td>R3840   R3903</td> <td>R3904   R3967</td> <td>R3968   R4167</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td>2   256</td> <td>V   Z</td> </tr> <tr> <td>S</td> <td>○</td> <td></td> <td>○</td> </tr> <tr> <td>N</td> <td>○</td> </tr> <tr> <td>D</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> <td></td> <td>○</td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z	S	○	○	○	○	○	○	○	○	○	○	○	○		○	N	○	○	○	○	○	○	○	○	○	○	○	○	○	○	D		○	○	○	○	○	○	○	○*	○*	○			○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																														
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z																																																														
S	○	○	○	○	○	○	○	○	○	○	○	○		○																																																														
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																														
D		○	○	○	○	○	○	○	○*	○*	○			○																																																														
<ul style="list-style-type: none"> <li>When operation control "EN" = 1 or "EN ↑" (P instruction) from 0 to 1, add the N successive 16-bit or 32-bit (D instruction) numerical values starting from S, and then divided by N. Store this mean value (rounding off numbers after the decimal point) in the register specified by D.</li> <li>While the N value is derived from the content of the register, if the N value is not between 2 and 256, then the N range error "ERR" will be set to 1, and do not execute the operation.</li> </ul>																																																																												
	<ul style="list-style-type: none"> <li>At left, the example program gets the mean value of the 3 successive 16-bit registers starting from R0, and stores the results into the 16-bit register R10</li> </ul>																																																																											
<p>S { (N=3)  </p> <table border="1"> <tr><td>R0</td><td>123</td></tr> <tr><td>R1</td><td>9</td></tr> <tr><td>R2</td><td>788</td></tr> </table> <p style="text-align: right;"><math>\frac{123 + 9 + 788}{3}</math> =306 (Rounding off the remainder)</p> <p>D   R10   306</p>	R0	123	R1	9	R2	788																																																																						
R0	123																																																																											
R1	9																																																																											
R2	788																																																																											

FUN 26 <b>D P</b> SQRT	SQUARE ROOT	FUN 26 <b>D P</b> SQRT
---------------------------	-------------	---------------------------

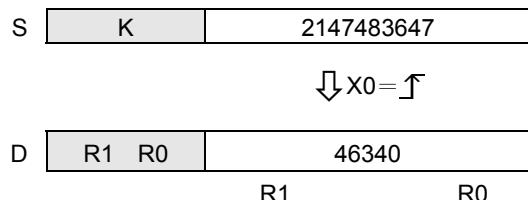


Range \ Oper- and	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit	V Z
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When operation control "EN" = 1 or "EN ↑" (**P** instruction) from 0 to 1, take the square root (rounding off numbers after the decimal point) of the data specified by the S field, and store the result into the register specified by D.
- While the S value is derived from the content of the register, if the value is negative, then the S value error flag "ERR" will be set to 1, and do not execute the operation.



- The instruction at left calculates the square root of the constant 2147483647, and stores the result in R0.



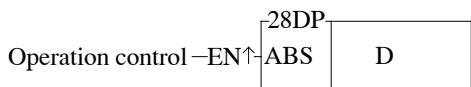
$$\sqrt{2147483647} = 46340.95$$

↑  
rounding off

## Arithmetical operation instructions

FUN 27 <b>D P</b> NEG	NEGATION (Take the negative value)	FUN 27 <b>D P</b> NEG																																																																
	<p>Operation control —EN↑</p>	<p>D : Register to be negated</p> <p>D may combine with V, Z to serve indirect address application</p>																																																																
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding-right: 10px;">Oper- and</th> <th style="text-align: center; padding-right: 10px;">Range</th> <th style="text-align: center; padding-right: 10px;">WY</th> <th style="text-align: center; padding-right: 10px;">WM</th> <th style="text-align: center; padding-right: 10px;">WS</th> <th style="text-align: center; padding-right: 10px;">TMR</th> <th style="text-align: center; padding-right: 10px;">CTR</th> <th style="text-align: center; padding-right: 10px;">HR</th> <th style="text-align: center; padding-right: 10px;">OR</th> <th style="text-align: center; padding-right: 10px;">SR</th> <th style="text-align: center; padding-right: 10px;">ROR</th> <th style="text-align: center; padding-right: 10px;">DR</th> <th style="text-align: center; padding-right: 10px;">XR</th> </tr> </thead> <tbody> <tr> <td></td> <td style="text-align: center; padding-right: 10px;">WY0</td> <td style="text-align: center; padding-right: 10px;">WM0</td> <td style="text-align: center; padding-right: 10px;">WS0</td> <td style="text-align: center; padding-right: 10px;">T0</td> <td style="text-align: center; padding-right: 10px;">C0</td> <td style="text-align: center; padding-right: 10px;">R0</td> <td style="text-align: center; padding-right: 10px;">R3904</td> <td style="text-align: center; padding-right: 10px;">R3968</td> <td style="text-align: center; padding-right: 10px;">R5000</td> <td style="text-align: center; padding-right: 10px;">D0</td> <td style="text-align: center; padding-right: 10px;">V</td> <td></td> </tr> <tr> <td></td> <td style="text-align: center; padding-right: 10px;"> </td> <td style="text-align: center; padding-right: 10px;">`</td> </tr> <tr> <td></td> <td style="text-align: center; padding-right: 10px;">WY240</td> <td style="text-align: center; padding-right: 10px;">WM1896</td> <td style="text-align: center; padding-right: 10px;">WS984</td> <td style="text-align: center; padding-right: 10px;">T255</td> <td style="text-align: center; padding-right: 10px;">C255</td> <td style="text-align: center; padding-right: 10px;">R3839</td> <td style="text-align: center; padding-right: 10px;">R3967</td> <td style="text-align: center; padding-right: 10px;">R4167</td> <td style="text-align: center; padding-right: 10px;">R8071</td> <td style="text-align: center; padding-right: 10px;">D3071</td> <td style="text-align: center; padding-right: 10px;">Z</td> <td></td> </tr> <tr> <td style="text-align: center;">D</td> <td style="text-align: center; padding-right: 10px;"><input type="radio"/></td> <td style="text-align: center; padding-right: 10px;"><input type="radio"/>*</td> <td style="text-align: center; padding-right: 10px;"><input type="radio"/>*</td> <td style="text-align: center; padding-right: 10px;"><input type="radio"/></td> <td style="text-align: center; padding-right: 10px;"><input type="radio"/></td> <td style="text-align: center; padding-right: 10px;"><input type="radio"/></td> </tr> </tbody> </table>	Oper- and	Range	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR		WY0	WM0	WS0	T0	C0	R0	R3904	R3968	R5000	D0	V													`		WY240	WM1896	WS984	T255	C255	R3839	R3967	R4167	R8071	D3071	Z		D	<input type="radio"/> *	<input type="radio"/> *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>								
Oper- and	Range	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR																																																						
	WY0	WM0	WS0	T0	C0	R0	R3904	R3968	R5000	D0	V																																																							
											`																																																							
	WY240	WM1896	WS984	T255	C255	R3839	R3967	R4167	R8071	D3071	Z																																																							
D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> *	<input type="radio"/> *	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																																																						
		<ul style="list-style-type: none"> <li>When operation control "EN" = 1 or "EN ↑" (<b>P</b> instruction) from 0 to 1, negate (ie. calculate 2's complement) the value of the content of the register specified by D, and store it back in the original D register.</li> <li>If the value of the content of D is negative, then the negation operation will make it positive.</li> </ul>																																																																
		<ul style="list-style-type: none"> <li>The instruction at left negates the value of the R0 register, and stores it back to R0.</li> </ul>																																																																
	<p>D  12345</p> <p style="text-align: center;">↓ X0 = ↴</p> <p>D  -12345</p>	<p>3039H</p> <p>CFC7H</p>																																																																

FUN 28 <b>D P</b> ABS	ABSOLUTE (Take the absolute value)	FUN 28 <b>D P</b> ABS
--------------------------	---------------------------------------	--------------------------



D : Register to be taken absolute value

D may combine with V, Z to serve indirect address application

Oper- and	Range	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR	
		WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	V   D3071	Z
		D	○	○	○	○	○	○	○	○*	○*	○	○

- When operation control "EN" = 1 or "EN ↑" (**P** instruction) from 0 to 1, calculate the absolute value of the content of the register specified by D, and write it back into the original D register.



- The instruction at left calculates the absolute value of the R0 register, and stores it back in R0.

D [R1 R0] –12345  CFC7H

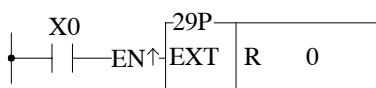
↓ X0 = ↑

D [R1 R0] 12345  3039H

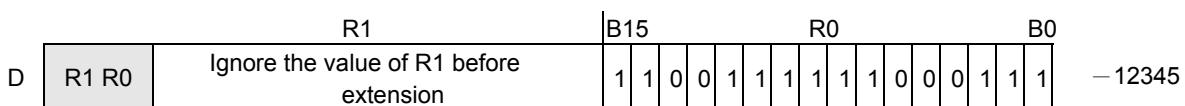
## Arithmetical operation instructions

FUN 29 P EXT	SIGN EXTENSION	FUN 29 P EXT
Operation control – EN↑ 	D : Register to be taken sign extension  D may combine with V, Z to serve indirect address application	

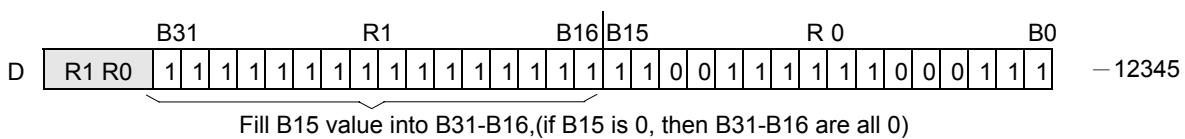
- When operation control "EN" = 1 or "EN ↑" (**P** instruction) from 0 to 1, this instruction will sign extent the 16 bit numerical value specified by D to 32-bit value and store it into the 32-bit register comprised by the two successive words, D + 1 and D. (Both values are the same, only it was originally formated as a 16 bit numerical value, and was then extended to be formated as a 32 bit numerical value.)
  - This instruction extent the numerical value of a 16-bit register into an equivalent numerical value in a 32-bit register (for example 33FFH converts to 000033FFH), Its main function is for numerical operations (+,-,\* ,/,CMP.....) which can take the 16 bit or 32 bit numerical values as operand. Before operation all the operand should be adjusted to the same length for proper operation.



- The instruction at left takes a 16 bit numerical value R0, and extends it to an equivalent value in 32 bits, then stores it into a 32 bit register ( $DR0=R1R0$ ) comprised R0 and R1



$\downarrow x_0 = \uparrow$



Before extension (16 bits) R0= CFC7H= -12345  
 After extension (32 bits) R1R0=FFFFCFC7H= -12345 } The two numerical values are actually the same

FUN 30 PID	General purpose PID operation (Brief description)	FUN 30 PID																																			
<p>Mode — A/M</p> <p>Bumpless — BUM</p> <p>Direction — D/R</p>	<p>30.PID</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>Ts :</td> <td>ERR - Setting error</td> </tr> <tr> <td>SR :</td> <td>HA - High alarm</td> </tr> <tr> <td>OR :</td> <td>PR :</td> </tr> <tr> <td>WR :</td> <td>LA - Low alarm</td> </tr> </table>	Ts :	ERR - Setting error	SR :	HA - High alarm	OR :	PR :	WR :	LA - Low alarm	<p>Ts : PID Operation time interval</p> <p>SR : Starting register of process control parameter table comprised by 8 consecutive registers.</p> <p>OR : PID output register</p> <p>PR : Starting register of the process parameter table comprised by 7 consecutive registers.</p> <p>WR : Starting register of working variable for PID internal operation. It requires 7 registers and can't be re-used in other part of the ladder program.</p>																											
Ts :	ERR - Setting error																																				
SR :	HA - High alarm																																				
OR :	PR :																																				
WR :	LA - Low alarm																																				
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">Oper- and</th> <th rowspan="2">Range</th> <th>HR</th> <th>ROR</th> <th>DR</th> <th>K</th> </tr> <tr> <th>R0 R3839</th> <th>R5000 R8071</th> <th>D0 D3071</th> <th></th> </tr> </thead> <tbody> <tr> <td>Ts</td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td>1~3000</td> </tr> <tr> <td>SR</td> <td><input type="radio"/></td> <td><input checked="" type="radio"/>*</td> <td><input type="radio"/></td> <td></td> </tr> <tr> <td>OR</td> <td><input type="radio"/></td> <td><input checked="" type="radio"/>*</td> <td><input type="radio"/></td> <td></td> </tr> <tr> <td>PR</td> <td><input type="radio"/></td> <td><input checked="" type="radio"/>*</td> <td><input type="radio"/></td> <td></td> </tr> <tr> <td>WR</td> <td><input type="radio"/></td> <td><input checked="" type="radio"/>*</td> <td><input type="radio"/></td> <td></td> </tr> </tbody> </table>	Oper- and	Range	HR	ROR	DR	K	R0 R3839	R5000 R8071	D0 D3071		Ts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1~3000	SR	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>		OR	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>		PR	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>		WR	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>		
Oper- and	Range			HR	ROR	DR	K																														
		R0 R3839	R5000 R8071	D0 D3071																																	
Ts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1~3000																																	
SR	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>																																		
OR	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>																																		
PR	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>																																		
WR	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>																																		

- PID function according to the current value of process variable (PV) derived from the external analog signal and the setting value (SP) of process performs the calculation, which base on the PID formula. The result of calculation is the control output for the controlled process, which can feed directly to the AO module or other output interface or leaved for further process. The usage of PID control for process if properly can achieve a fast and smooth result of PV tracking toward SP change or immune to the disturbance of process.

- The PID formula in digital form:

$$Mn = [(1000/Pb) \times En] + \sum_{0}^{n} [(1000/Pb) \times Ti \times Ts \times En] - [(1000/Pb) \times Td \times (PVn - PVn-1)/Ts] + Bias$$

Mn : Control output at time "n"

Pb : Proportional band ( range : 2~5000, unit 0.1%. Kc (gain) =1000/ Pb )

Ti : Intergal time constant ( range : 0~9999 corresponds to 0.00~99.99 Repeats/Minute )

Td : Differential time constant ( range : 0~9999 corresponds to 0.00~99.99 Minutes )

PVn : Process value at time "n"

PV n-1 : Process value at time "n"

En : Error at time "n" =set value ( SP ) – process value at time "n" ( PVn )

Ts : Interval time of PID calculation ( range: 1~3000, unit : 0.01 S )

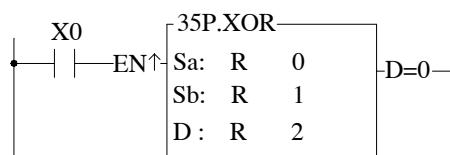
Bias : Control output offset ( range: 0~4095 )

- For detail description of this function, please refer chapter 21.

## Logical operation instruction

FUN 35 <b>D P</b> XOR	EXCLUSIVE OR	FUN 35 <b>D P</b> XOR
<p>Operation control -EN↑</p> <pre> graph LR     EN[Operation control -EN↑] --&gt; 35DP["35DP.XOR"]     Sa[ ] --- 35DP     Sb[ ] --- 35DP     35DP -- "D=0—Result as 0" --&gt; D[ ]     </pre>	<p>Sa : Source data a for exclusive or operation  Sb : Source data b for exclusive or operation  D : Register storing XOR results  Sa, Sb, D may combine with V, Z to serve indirect address application</p>	

- When operation control "EN" = 1 or "EN ↑" (**P** instruction) changes from 0 to 1, will perform the logical XOR (exclusive or) operation of data Sa and Sb. The operation of this function is to compare the corresponding bits of Sa and Sb (B0~B15 or B0~B31), and if bits at the same position have different status, then set the corresponding bit within D as 1, otherwise as 0.
  - After the operation, if all the bits in D are all 0, then set the 0 flag "D = 0" to 1.



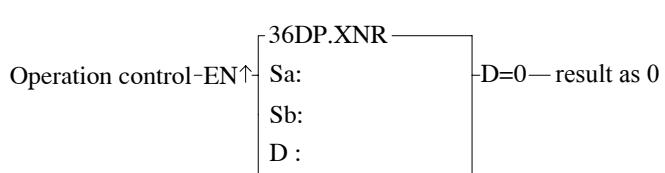
- The instruction at left makes a logical XOR operation using the R0 and R1 registers, and stores the result in R2.

Sa	R0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0

$$\downarrow x_0 = \underline{\uparrow}$$

D R2 0 1 0 1 0 1 0 1 1 1 1 0 0 1 0 1 1

FUN 36 <b>D P</b> XNR	ENCLUSIVE OR	FUN 36 <b>D P</b> XNR
--------------------------	--------------	--------------------------



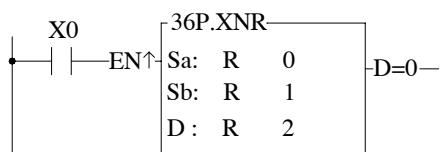
Sa : Data a for XNR operation

Sb : Data b for XNR operation

D : Register storing XNR results

Sa, Sb, D may combine with V, Z to serve indirect address application

- When operation control "EN" = 1 or "EN ↑" (**P** instruction) changes from 0 to 1, will perform the logical XNR (inclusive or) operation of data Sa and Sb. The operation of this function is to compare the corresponding bits of Sa and Sb (B0~B15 or B1~B31), and if the bit has the same value, then set the corresponding bit within D as 1. If not then set it to 0.
  - After the operation, if the bits in D are all 0, then set the 0 flag "D=0" to 1.



- The instruction at left makes a logical XNR operation of the R0 and R1 registers, and the results are stored in the R2 register.

Sa	R0	1	0	1	1	1	0	1	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1

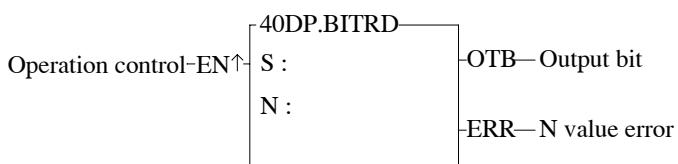
$$\downarrow x_0 = \underline{\uparrow}$$

D	R2	1	0	1	0	1	0	1	0	0	0	0	1	1	0	1	0	0
---	----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

## Comparison instructions

FUN 37 <b>D P</b> ZNCMP	ZONE COMPARE	FUN 37 <b>D P</b> ZNCMP																																																																										
<p>Operation control-EN↑</p> <p>37DP.ZNCMP</p> <p>S :      INZ – Inside zone           S&gt;U – Higher than upper limit           S&lt;L – Lower than lower limit           ERR – Limit value error</p>	<p>S : Register for zone comparison           Su: The upper limit value           Sl : The lower limit value           S, Su, Sl may combine with V, Z to serve indirect address application</p>																																																																											
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th> <th style="text-align: center; padding: 2px;">WX</th> <th style="text-align: center; padding: 2px;">WY</th> <th style="text-align: center; padding: 2px;">WM</th> <th style="text-align: center; padding: 2px;">WS</th> <th style="text-align: center; padding: 2px;">TMR</th> <th style="text-align: center; padding: 2px;">CTR</th> <th style="text-align: center; padding: 2px;">HR</th> <th style="text-align: center; padding: 2px;">IR</th> <th style="text-align: center; padding: 2px;">OR</th> <th style="text-align: center; padding: 2px;">SR</th> <th style="text-align: center; padding: 2px;">ROR</th> <th style="text-align: center; padding: 2px;">DR</th> <th style="text-align: center; padding: 2px;">K</th> <th style="text-align: center; padding: 2px;">XR</th> </tr> <tr> <th style="text-align: left; padding: 2px;">Oper- and</th> <th style="text-align: center; padding: 2px;">WX0   WX240</th> <th style="text-align: center; padding: 2px;">WY0   WY240</th> <th style="text-align: center; padding: 2px;">WM0   WM1896</th> <th style="text-align: center; padding: 2px;">WS0   WS984</th> <th style="text-align: center; padding: 2px;">T0   T255</th> <th style="text-align: center; padding: 2px;">C0   C255</th> <th style="text-align: center; padding: 2px;">R0   R3839</th> <th style="text-align: center; padding: 2px;">R3840   R3903</th> <th style="text-align: center; padding: 2px;">R3904   R3967</th> <th style="text-align: center; padding: 2px;">R3968   R4167</th> <th style="text-align: center; padding: 2px;">R5000   R8071</th> <th style="text-align: center; padding: 2px;">D0   D3071</th> <th style="text-align: center; padding: 2px;">16/32-bit +/- number</th> <th style="text-align: center; padding: 2px;">V . Z</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">S</td> <td style="text-align: center; padding: 2px;">○</td> </tr> <tr> <td style="padding: 2px;">SU</td> <td style="text-align: center; padding: 2px;">○</td> </tr> <tr> <td style="padding: 2px;">SL</td> <td style="text-align: center; padding: 2px;">○</td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V . Z	S	○	○	○	○	○	○	○	○	○	○	○	○	○	○	SU	○	○	○	○	○	○	○	○	○	○	○	○	○	○	SL	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																														
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V . Z																																																														
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																														
SU	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																														
SL	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																														
<ul style="list-style-type: none"> <li>● When operation control "EN" = 1 or "EN ↑" (<b>P</b> instruction) changes from 0 to 1, compares S with upper limit Su and lower limit Sl. If S is between the upper limit and the lower limit (<math>S_L \leq S \leq S_u</math>), then set the inside zone flag "INZ" to 1. If the value of S is greater than the upper limit Su, then set the higher than upper limit flag "S&gt;U" to 1. If the value of S is smaller then the lower limit Sl, then set the lower than lower limit flag "S&lt;L" as 1.</li> <li>● The upper limit Su should be greater than the lower limit Sl. If <math>S_u &lt; S_l</math>, then the limit value error flag "ERR" will set to 1, and this instruction will not carry out.</li> </ul>	<p>X0      EN↑</p> <p>37P.ZNCMP</p> <p>S : R 0      INZ ( )           SU : R 1      S&gt;U –           SL : R 2      S&lt;L –           ERR –</p>	<ul style="list-style-type: none"> <li>● The instruction at left compares the value of R0 with the upper and lower limit zones formed by R1 and R2. If the values of R0~R2 are as shown in the diagram at bottom left, then the result can then be obtained as at the right of this diagram.</li> <li>● If want to get the status of out side the zone, then OUT NOT Y0 may be used, or an OR operation between the two outputs S&gt;U and S&lt;L may be carried out, and move the result to Y0.</li> </ul>																																																																										
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">S</td> <td style="width: 10%;">R0</td> <td style="width: 10%;">200</td> <td rowspan="3" style="width: 10%; vertical-align: middle; text-align: center;">           ( Upper limit value )            X0 =  </td> <td rowspan="3" style="width: 10%; vertical-align: middle; text-align: center;">           Y0  </td> </tr> <tr> <td>S<sub>U</sub></td> <td>R1</td> <td>300</td> </tr> <tr> <td>S<sub>L</sub></td> <td>R2</td> <td>100</td> </tr> </table> <p style="margin-top: 10px;">Before-execution</p>	S	R0	200	( Upper limit value ) X0 =	Y0 	S <sub>U</sub>	R1	300	S <sub>L</sub>	R2	100	<p>( Upper limit value )      X0 = </p> <p>( Lower limit value )      </p>	<p>Results of execution</p>																																																															
S	R0	200	( Upper limit value ) X0 =			Y0 																																																																						
S <sub>U</sub>	R1	300																																																																										
S <sub>L</sub>	R2	100																																																																										

FUN 40 <b>D P</b> BITRD	BIT READ	FUN 40 <b>D P</b> BITRD
----------------------------	----------	----------------------------



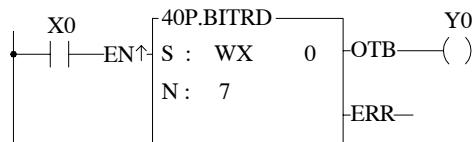
S : Source data to be read

N : The bit number of the S data to be read out.

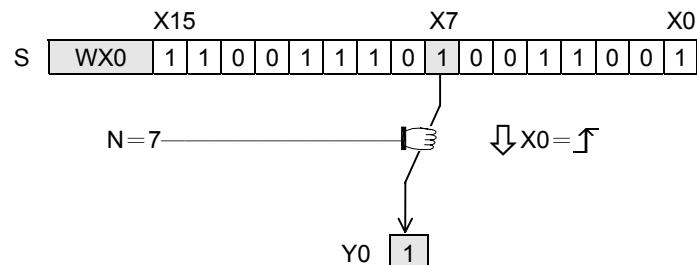
S, N may combine with V, Z to serve indirect address application

Range Oper- and	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V · Z
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	0~31

- When read control "EN" = 1 or "EN ↑" (**P** instruction) changes from 0 to 1, take the Nth bit of the S data out , and put it to the output bit "OTB".
- When read control "EN" = 0 or "EN ↑" (**P** instruction) is not change from 0 to 1, The output "OTB" can be selected to keep at the last state( if M1919=0 ) or set to zero ( if M1919=1 ).
- When the operand is 16 bit, the effective range for N is 0~15. For 32 bit operand (**D** instruction) it is 0~31. N beyond this range will set the N value error flag "ERR" to 1, and do not carry out this instruction.



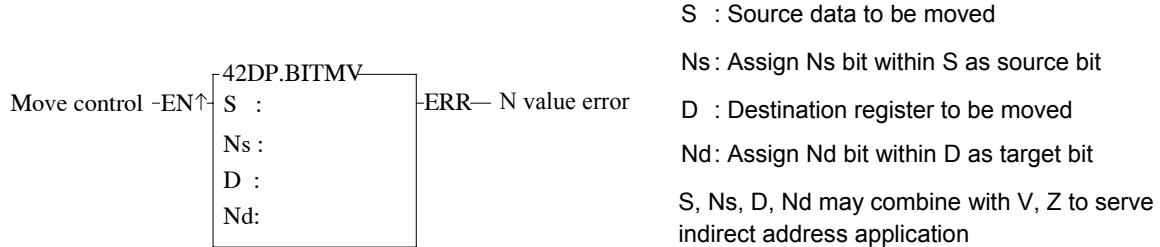
- The instruction at left reads the 7th bit (X7) status from WX0 (X0~X15) and output to Y0. The results are as follows:



## Data movement instructions

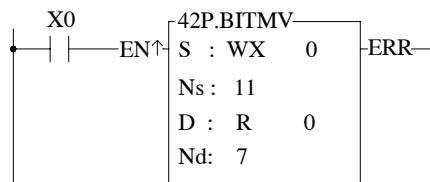
FUN 41 <b>D P</b> BITWR	BIT WRITE	FUN 41 <b>D P</b> BITWR																																																									
	<p>41DP.BITWR</p> <p>D : Register for bit write N : The bit number of the D register to be written. D, N may combine with V, Z to serve indirect address application</p>																																																										
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="3">Range Oper- and</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> <tr> <th>WX0   WX240</th> <th>WY0   WY240</th> <th>WM0   WM1896</th> <th>WS0   WS984</th> <th>T0   T255</th> <th>C0   C255</th> <th>R0   R3839</th> <th>R3840   R3903</th> <th>R3904   R3967</th> <th>R3968   R4167</th> <th>R5000   R8071</th> <th>D0   D3071</th> <th>0   15 or 31</th> <th>V . Z</th> </tr> <tr> <th>D</th> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td style="text-align: center;">○*</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○</td> </tr> </thead> <tbody> <tr> <th>N</th> <td style="text-align: center;">○</td> </tr> </tbody> </table>	Range Oper- and	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	0   15 or 31	V . Z	D	○	○	○	○	○	○	○	○	○*	○*	○	○	○	N	○	○	○	○	○	○	○	○	○	○	○	○	○	
Range Oper- and	WX		WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																												
	WX0   WX240		WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	0   15 or 31	V . Z																																												
	D	○	○	○	○	○	○	○	○	○*	○*	○	○	○																																													
N	○	○	○	○	○	○	○	○	○	○	○	○	○																																														
	<ul style="list-style-type: none"> <li>When write control "EN" = 1 or "EN ↑" (<b>P</b> instruction) changes from 0 to 1, will write the write bit (INB) into the Nth bit of register D.</li> <li>When the operand is 16 bit, the effective range of N is 0~15. For 32 bit (<b>D</b> instruction) operand it is 0~31. N beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.</li> </ul>																																																										
	<p>● The instruction at left writes the status of the write bit INB into B3 of R0. Assuming X1 = 1, the result will be as follows:</p>																																																										
	<p>X1 = 1</p> <p>N=3</p> <p>D [ R0   B15   B14   B13   B12   B11   B10   B9   B8   B7   B6   B5   B4   B3   1   B1   B0 ]</p> <p>Bits other than B3 remain unchanged</p>																																																										

FUN 42 <b>D P</b> BITMV	BIT MOVE	FUN 42 <b>D P</b> BITMV
----------------------------	----------	----------------------------

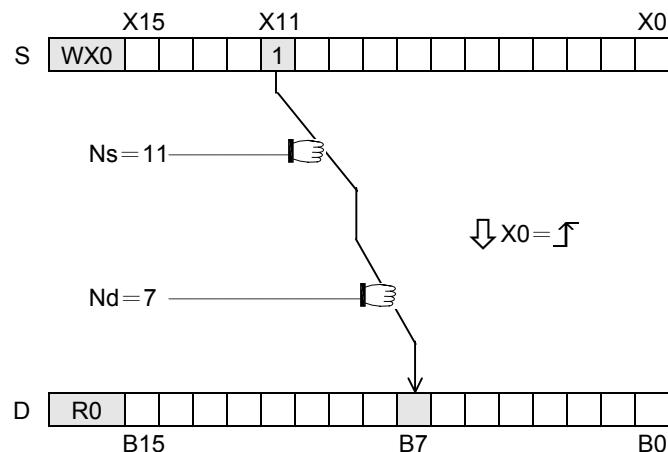


Range \ Oper- and	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V 、 Z
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0~31	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0~31	○

- When move control "EN" = 1 or "EN ↑" (**P** instruction) changes from 0 to 1, will move the bit status specified by Ns within S into the bit specified by Nd within D.
- When the operand is 16 bit, the effective range of N is 0~15. For 32 bit (**D** instruction) operand the effective range is 0~31. N beyond this range will set the N value error flag "ERR" to 1, and do not carry out this instruction.



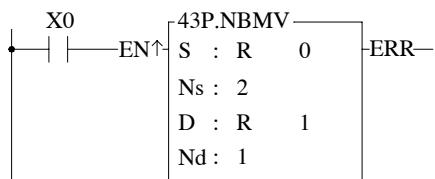
- The instruction at left moves the status of B11 (X11) within S into the B7 position within D. Except bit B7, other bits within D does not change.



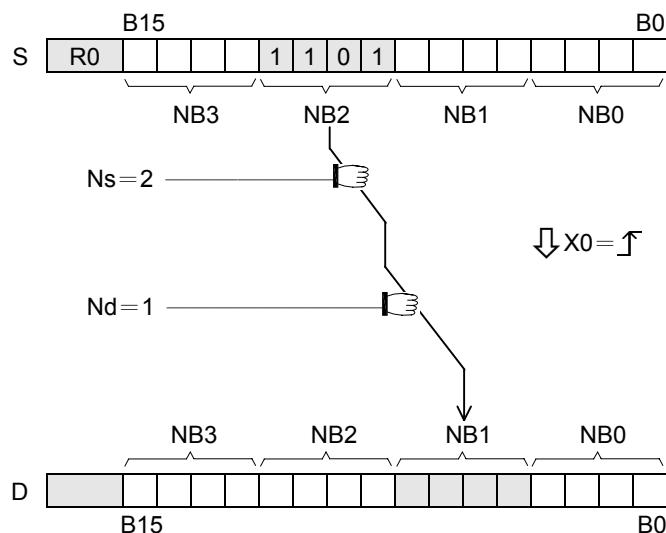
## Data movement instructions

FUN 43 <b>D P</b> NBMV	NIBBLE MOVE	FUN 43 <b>D P</b> NBMV																																																																																							
Move control -EN↑ S : Ns : D : Nd:	43DP.NBMV -ERR— N value error	S : Source data to be moved Ns: Assign Ns nibble within S as source nibble D : Destination register to be moved Nd: Assign Nd nibble within D as target nibble S, Ns, D, Nd may combine with V, Z to serve indirect address application																																																																																							
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th rowspan="2">Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> <tr> <th>WX0   WX240</th> <th>WY0   WY240</th> <th>WM0   WM1896</th> <th>WS0   WS984</th> <th>T255   C255</th> <th>C0   R3839</th> <th>R0   R3903</th> <th>R3840   R3967</th> <th>R3904   R4167</th> <th>R3968   R4167</th> <th>R5000   R8071</th> <th>D0   D3071</th> <th>16/32-bit +/- number</th> <th>V · Z</th> </tr> </thead> <tbody> <tr> <td>S</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr> <tr> <td>Ns</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>0~7</td><td>○</td></tr> <tr> <td>D</td><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td><td>○</td><td>○*</td><td>○*</td><td>○</td><td></td><td></td><td>○</td></tr> <tr> <td>Nd</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>0~7</td><td>○</td></tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255   C255	C0   R3839	R0   R3903	R3840   R3967	R3904   R4167	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V · Z	S	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Ns	○	○	○	○	○	○	○	○	○	○	○	○	0~7	○	D		○	○	○	○	○		○	○*	○*	○			○	Nd	○	○	○	○	○	○	○	○	○	○	○	○	0~7	○
Range		WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																										
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255   C255	C0   R3839	R0   R3903	R3840   R3967	R3904   R4167	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V · Z																																																																											
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																											
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0~7	○																																																																											
D		○	○	○	○	○		○	○*	○*	○			○																																																																											
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0~7	○																																																																											

- When move control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, will move the Ns'th nibble from within S to the nibble specified by Nd within D. (A nibble is comprised by 4 bits. Starting from the lowest bit of the register, B0, each successive 4 bits form a nibble, so B0~B3 form nibble 0, B4~B7 form nibble 1, etc...)
- When the operand is 16 bit, the effective range of Ns or Nd is 0~3. For 32 bit (**D** instruction) operand the range is 0~7. Beyond this range, will set the N value error flag "ERR" to 1 , and do not carry out this instruction.



The instruction at left moves the third nibble NB2 (B8~B11) within S to the first nibble NB1 (B4~B7) within D. Other nibbles within D remain unchanged.



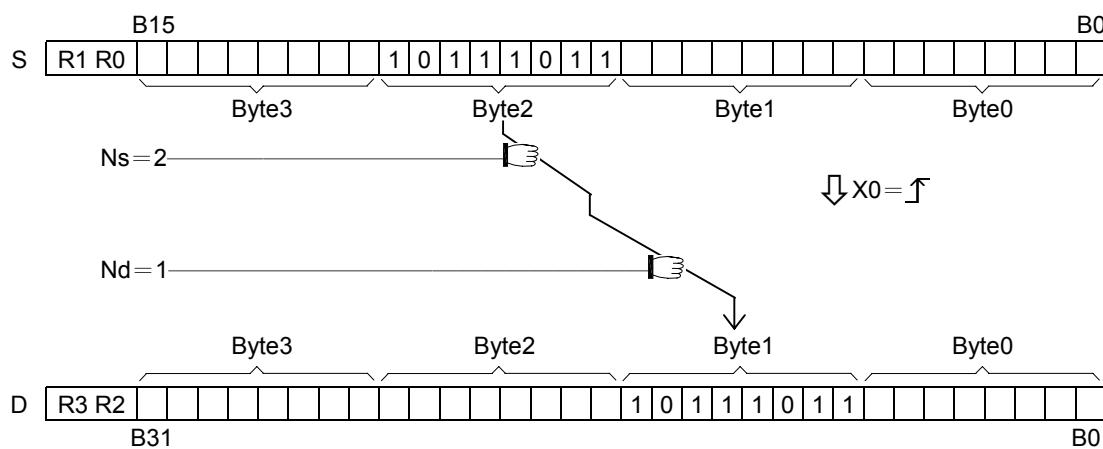
FUN 44 <b>D P</b> BYMV	BYTE MOVE	FUN 44 <b>D P</b> BYMV
---------------------------	-----------	---------------------------

Move control -EN↑ [44DP.BYMV] S : ERR—N value error  
Ns :  
D :  
Nd:

- S : Source data to be moved
- Ns : Assign Ns byte within S as source byte
- D : Destination register to be moved
- Nd : Assign Nd byte within D as target byte
- S, Ns, D, Nd may combine with V, Z to serve indirect address application

- When move control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, move Nsth byte within S to Ndth byte position within D. (A byte is comprised of 8 bits. Starting from the lowest bit of the register, B0, each successive eight bits form a byte, so B0~B7 form byte 0, B8~B15 form byte 1, etc...)
  - When the operand is 16 bit, the effective range of Ns or Nd is 0~1. For 32 bit (**D** instruction) operand, the range is 0~3. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

- The instruction at left moves the third byte (B16~B23) within S (32 bit register composed of R1R0), to the first byte within D (32 bit register composed of R3R2). Other bytes within D remain unchanged.



## Data movement instructions

FUN 46 P SWAP	BYTE SWAP	FUN 46 P SWAP
------------------	-----------	------------------

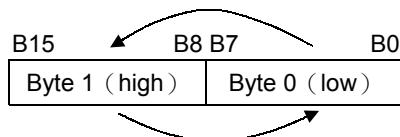
Swap control-EN↑ 46P SWAP D

D : Register for byte data swap

D may combine with V, Z to serve indirect address application

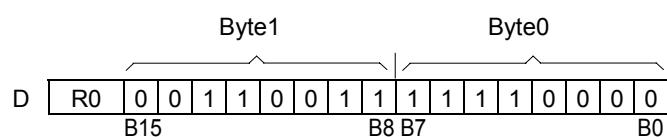
Range	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
Oper- and	WY0	WM0	WS0	T0	C0	R0	R3904	R3968	R5000	D0	V
											‘
	WY240	WM1896	WS984	T255	C255	R3839	R3967	R4167	R8071	D3071	Z
D	○	○	○	○	○	○	○	○*	○*	○	○

- When swap control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, swap the data of the low byte, Byte 0 (B0~B7), and the high byte, Byte 1 (B8~B15), in the 16 bit register specified by D.

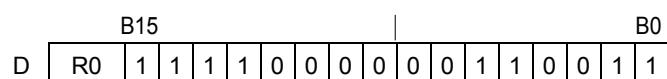


The timing diagram illustrates the sequence of events for the **X0**, **EN**, **SWAP**, **R**, and **0** signals. The **X0** signal is high during the first half of the period. The **EN** signal is high during the second half of the period. The **SWAP** signal is asserted at the start of the second half. The **R** signal is asserted at the end of the second half. The **0** signal is asserted during the second half.

- The instruction at left swaps the data of the low byte (B0~B7) and the high byte (B8~B15) in R0. The results are as follows:

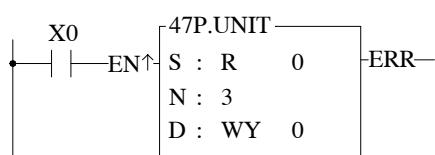


$\downarrow x_0 = \uparrow$

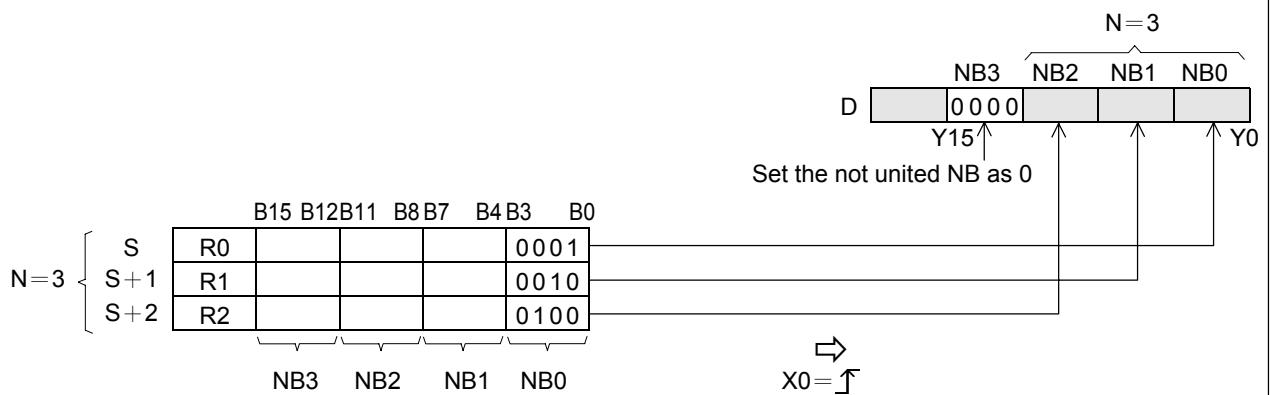


## Data movement instructions

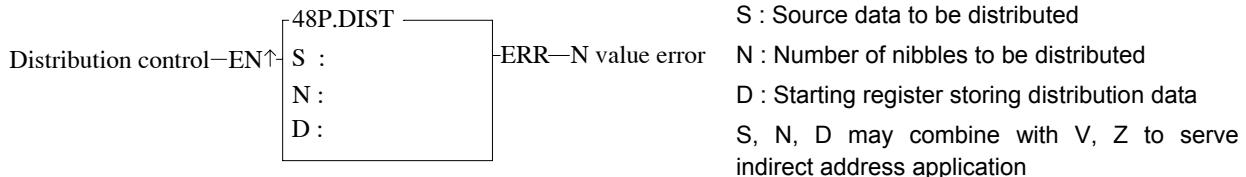
FUN 47 P UNIT	NIBBLE UNITE	FUN 47 P UNIT																																																																																									
	<p>Unite control —EN↑</p> <p>S : Starting source register to be united N : Number of nibbles to be united D : Registers storing united data S, N, D may combine with V, Z to serve indirect address application</p>																																																																																										
	<table border="1"> <thead> <tr> <th>Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td rowspan="2">Oper- and</td> <td>WX0</td> <td>WY0</td> <td>WM0</td> <td>WS0</td> <td>T0</td> <td>C0</td> <td>R0</td> <td>R3840</td> <td>R3904</td> <td>R3968</td> <td>R5000</td> <td>D0</td> <td>1</td> <td>V</td> </tr> <tr> <td>WX240</td> <td>WY240</td> <td>WM1896</td> <td>WS984</td> <td>T255</td> <td>C255</td> <td>R3839</td> <td>R3903</td> <td>R3967</td> <td>R4167</td> <td>R8071</td> <td>D3071</td> <td>4</td> <td>Z</td> </tr> <tr> <td>S</td> <td>○</td> </tr> <tr> <td>N</td> <td>○</td> </tr> <tr> <td>D</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> <td>○</td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	1	V	WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D3071	4	Z	S	○	○	○	○	○	○	○	○	○	○	○	○	○	○	N	○	○	○	○	○	○	○	○	○	○	○	○	○	○	D		○	○	○	○	○	○		○	○*	○*	○		○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																													
Oper- and	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	1	V																																																																													
	WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D3071	4	Z																																																																													
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																													
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																													
D		○	○	○	○	○	○		○	○*	○*	○		○																																																																													
	<ul style="list-style-type: none"> <li>When unite control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, take out the lowest nibbles NB0, of N successive registers starting from S, and fill them into NB0, NB1, ....NBn-1 of D in ascending order. Nibbles not yet filled in D (when N is odd) are filled with 0. (A nibble is comprised by 4 bits. Starting from the lowest bit in the register, B0, each successive four bits form a nibble, so B0~B3 form nibble 0, B4~B7 form nibble 1, etc...).</li> <li>This instruction only provides WORD (16 bit) operand. Because of this, there are usually only 4 nibbles can be involved. Therefore the effective range of N is 1~4. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.</li> </ul>																																																																																										



- The instruction at left takes out NB0 from 3 registers, R0, R1 and R2, and fills them into NB0~NB2 within WY0 register.

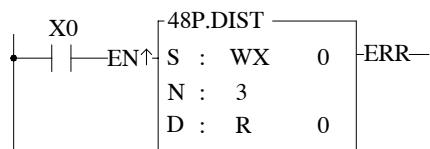


FUN 48 <b>P</b> DIST	NIBBLE DISTRIBUTE	FUN 48 <b>P</b> DIST
-------------------------	-------------------	-------------------------

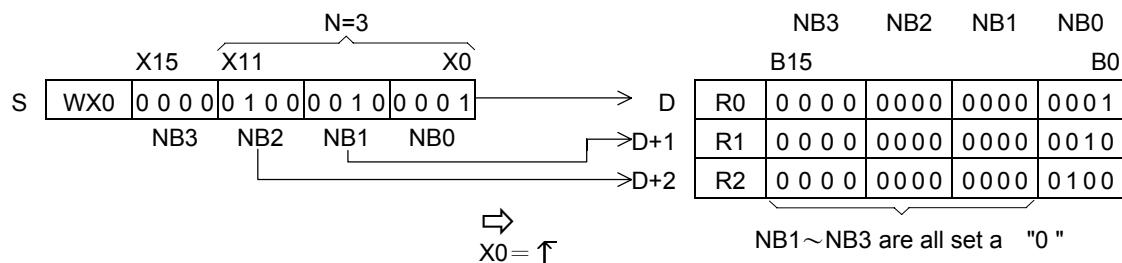


Range Oper- and	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16-bit +/- number	V 、 Z
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	1~4	○
D		○	○	○	○	○	○		○*	○*	○			○

- When distribution control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, will take N successive nibbles starting from the lowest nibble NB0 within S, and distribute them in ascending order into the N nibbles of N registers starting from D. The nibbles other than NB0 in each of the registers within D are all set to zero. (A nibble is comprised by 4 bits. Starting from the lowest bit in a register, B0, each successive 4 bits form a nibble, so B0~B3 form nibble 0, B4~B7 form nibble 1, etc...)
- This instruction only provides WORD (16 bit) operand. Therefore there are usually only 4 nibbles can be involved, so the effective value of N is 1~4. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.



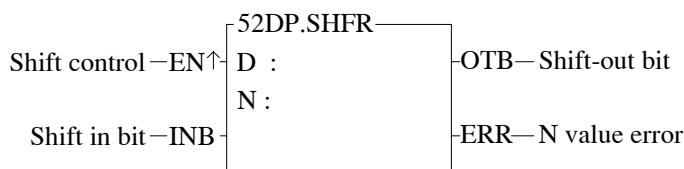
- The instruction at left writes NB0~NB2 from the WX0 register into the NB0 of the 3 consecutive registers R0~R2.



## Shifting/Rotating instructions

FUN 51 <b>D P</b> SHFL	SHIFT LEFT	FUN 51 <b>D P</b> SHFL																																																																											
<p>Shift control—EN↑</p> <p>Shift in bit—INB</p>	<p>D : Register to be shifted N : Number of bits to be shifted N, D may combine with V, Z to serve indirect address application</p>																																																																												
<table border="1"> <thead> <tr> <th>Range \ Operand</th><th>WX</th><th>WY</th><th>WM</th><th>WS</th><th>TMR</th><th>CTR</th><th>HR</th><th>IR</th><th>OR</th><th>SR</th><th>ROR</th><th>DR</th><th>K</th><th>XR</th></tr> </thead> <tbody> <tr> <td>WX0</td><td>WY0</td><td>WM0</td><td>WS0</td><td>T0</td><td>C0</td><td>R0</td><td>R3840</td><td>R3904</td><td>R3968</td><td>R5000</td><td>D0</td><td>1</td><td>1</td><td>V</td></tr> <tr> <td>WX240</td><td>WY240</td><td>WM1896</td><td>WS984</td><td>T255</td><td>C255</td><td>R3839</td><td>R3903</td><td>R3967</td><td>R4167</td><td>R8071</td><td>D3071</td><td>16</td><td>32</td><td>Z</td></tr> <tr> <td>D</td><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○*</td><td>○*</td><td>○</td><td></td><td></td><td>○</td></tr> <tr> <td>N</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr> </tbody> </table>	Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	1	1	V	WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D3071	16	32	Z	D		○	○	○	○	○	○	○	○*	○*	○			○	N	○	○	○	○	○	○	○	○	○	○	○	○	○	○		
Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																															
WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	1	1	V																																																															
WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D3071	16	32	Z																																																															
D		○	○	○	○	○	○	○	○*	○*	○			○																																																															
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																															
<ul style="list-style-type: none"> <li>When shift control "EN" = 1 or "EN ↑" (<b>P</b> instruction) has a transition from 0 to 1, will shift the data of the D register towards the left by N successive bits (in ascending order). After the lowest bit B0 has been shifted left, its position will be replaced by shift-in bit INB, while the status of shift-out bits B15 or B31 (<b>D</b> instruction) will appear at shift-out bit "OTB".</li> <li>If the operand is 16 bit, the effective range of N is 1~16. For 32 bits (<b>D</b> instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.</li> </ul>		<ul style="list-style-type: none"> <li>The instruction at left shifts the data in register R0 towards the left by 4 successive bits. The results are shown below.</li> </ul>																																																																											

FUN 52 <b>D P</b> SHFR	SHIFT RIGHT	FUN 52 <b>D P</b> SHFR
---------------------------	-------------	---------------------------



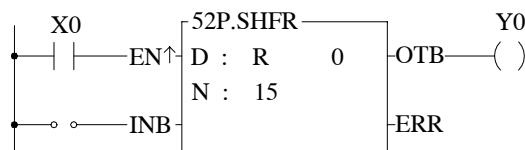
D : Register to be shifted

N : Number of bits to be shifted

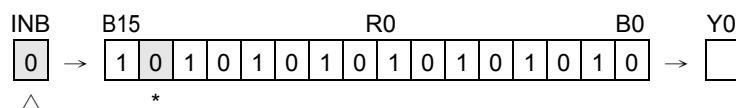
D, N may combine with V, Z to serve indirect address application

Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	1   or 16 32	V ` Z
D	○	○	○	○	○	○	○	○	○	○*	○*	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

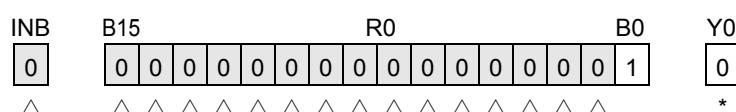
- When shift control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, will shift the data of D register towards the right by N successive bits (in descending order). After the highest bits, B15 or B31 (**D** instruction) have been shifted right, their positions will be replaced by the shift-in bit INB, while shift-out bit B0 will appear at shift-out bit "OTB".
- If the operand is 16 bit, the effective range of N is 1~16. For 32 bits (**D** instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.



- The instruction at left shifts the data in R0 register towards the right by 15 successive bits. The results are shown below.



$$\Downarrow X0 = \boxed{1}$$

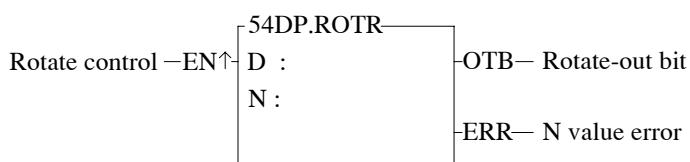


## Shifting/Rotating instructions

FUN 53 <b>D P</b> ROTL	ROTATE LEFT	FUN 53 <b>D P</b> ROTL																																																												
<p>Rotate control—EN↑</p> <p>D : [ ]</p> <p>N : [ ]</p> <p>OTB—Rotate-out bit</p> <p>ERR—N value error</p>	<p>53DP.ROTL</p> <p>D : Register to be rotated</p> <p>N : Number of bits to be rotated</p> <p>D, N may combine with V, Z to serve indirect address application</p>																																																													
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th> <th style="padding: 2px;">WX</th> <th style="padding: 2px;">WY</th> <th style="padding: 2px;">WM</th> <th style="padding: 2px;">WS</th> <th style="padding: 2px;">TMR</th> <th style="padding: 2px;">CTR</th> <th style="padding: 2px;">HR</th> <th style="padding: 2px;">IR</th> <th style="padding: 2px;">OR</th> <th style="padding: 2px;">SR</th> <th style="padding: 2px;">ROR</th> <th style="padding: 2px;">DR</th> <th style="padding: 2px;">K</th> <th style="padding: 2px;">XR</th> </tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;">Oper- and</td> <td style="padding: 2px;">WX0   WX240</td> <td style="padding: 2px;">WY0   WY240</td> <td style="padding: 2px;">WM0   WM1896</td> <td style="padding: 2px;">WS0   WS984</td> <td style="padding: 2px;">T0   T255</td> <td style="padding: 2px;">C0   C255</td> <td style="padding: 2px;">R0   R3839</td> <td style="padding: 2px;">R3840   R3903</td> <td style="padding: 2px;">R3904   R3967</td> <td style="padding: 2px;">R3968   R4167</td> <td style="padding: 2px;">R5000   R8071</td> <td style="padding: 2px;">D0   D3071</td> <td style="padding: 2px;">1   or 16</td> <td style="padding: 2px;">1   32</td> <td style="padding: 2px;">V ` Z</td> </tr> <tr> <td style="text-align: left; padding: 2px;">D</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○*</td> <td style="padding: 2px;">○*</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○</td> </tr> <tr> <td style="text-align: left; padding: 2px;">N</td> <td style="padding: 2px;">○</td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	1   or 16	1   32	V ` Z	D	○	○	○	○	○	○	○	○	○*	○*	○	○	○	○	N	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	1   or 16	1   32	V ` Z																																															
D	○	○	○	○	○	○	○	○	○*	○*	○	○	○	○																																																
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																
<ul style="list-style-type: none"> <li>When rotate control "EN" = 1 or "EN ↑" (<b>P</b> instruction) has a transition from 0 to 1, will rotate the data of D register towards the left by N successive bits (in ascending order, ie. in a 16-bit instruction, B0→B1, B1→B2, ..., B14→B15, B15→B0. In a 32-bit instruction, B0→B1, B1→B2, ..., B30→B31, B31→B0). At the same time, the status of the rotated out bits B15 or B31 (<b>D</b> instruction) will appear at rotate-out bit "OTB".</li> <li>If the operand is 16 bit, the effective range of N is 1~16. For 32 bits (<b>D</b> instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.</li> </ul>	<p>X0</p> <p>EN↑</p> <p>D : R 0</p> <p>N : 9</p> <p>OTB—( )</p> <p>Y0</p> <p>ERR</p>	<ul style="list-style-type: none"> <li>The instruction at left rotates data from the R0 register towards the left 9 successive bits. The results are shown below.</li> </ul>																																																												
<p>R0</p> <p>B0</p> <p>Y0</p> <p>X0 = 1</p>	<p>B15 R0 B0</p> <p>Y0</p>																																																													

FUN 54 **D P**  
ROTR

## ROTATE RIGHT

FUN 54 **D P**  
ROTR

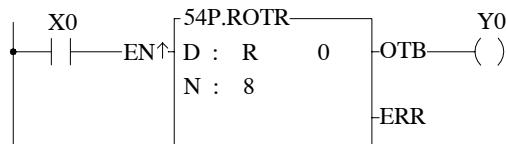
D : Register to be rotated

N : Number of bits to be rotated

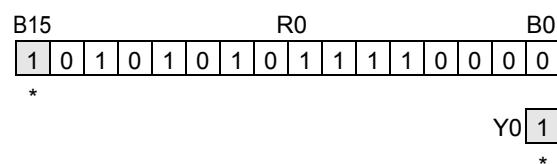
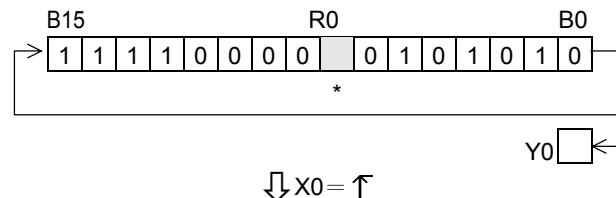
D, N may combine with V, Z to serve indirect address application

Oper- and	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	
		WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	1   16	1   32	V · Z
		D		○	○	○	○	○	○	○	○*	○*	○	○	○	
N		○	○	○	○	○	○	○	○	○	○	○	○	○	○	

- When rotate control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, will rotate the bit data of D register towards the right by N successive bits (in descending order, ie. in a 16-bit instruction, B15→B14, B14→B13, ..., B1→B0, B0→B15. In a 32-bit instruction, B31→B30, B30→B29, ..., B1→B0, B0→B31). At the same time, the status of the rotated out B0 bits will appear at the rotate-out bit "OTB".
- If the operand is 16 bit, the effective range of N is 1~16. For 32 bits (**D** instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.



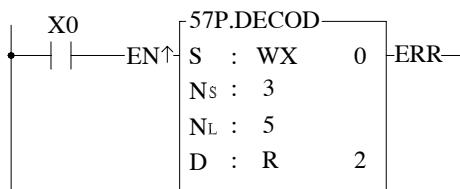
- The instruction rotates data from R0 register towards the right 8 successive bits. The results are shown below.



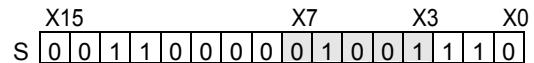
## Code conversion instructions

FUN 57 P DECOD	DECODE	FUN 57 P DECOD																																																																																								
<p>57P.DECOD</p> <p>Decode control - EN↑</p> <p>S :                   ERR - Range error</p> <p>NS :                  </p> <p>NL :                  </p> <p>D :                  </p>	<p>S : Source data register to be decoded (16 bits)</p> <p>NS : Starting bits to be decoded within S</p> <p>NL : Length of decoded value (1~8 bits)</p> <p>D : Starting register storing decoded results (2~256 points = 1~16 words)</p> <p>S, NS, NL, D may combine with V, Z to serve indirect address application</p>																																																																																									
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th rowspan="2">Range \ Operand</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> <tr> <th>WX0   WX240</th> <th>WY0   WY240</th> <th>WM0   WM1896</th> <th>WS0   WS984</th> <th>T0   T255</th> <th>C0   C255</th> <th>R0   R3839</th> <th>R3840   R3903</th> <th>R3904   R3967</th> <th>R3968   R4167</th> <th>R5000   R8071</th> <th>D0   D3071</th> <th>16-bit +/- number</th> <th>V Z</th> </tr> </thead> <tbody> <tr> <td>S</td> <td>○</td> </tr> <tr> <td>NS</td> <td>○</td> <td>0~15</td> <td>○</td> </tr> <tr> <td>NL</td> <td>○</td> <td>1~8</td> <td>○</td> </tr> <tr> <td>D</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td>○</td> <td></td> <td>○</td> </tr> </tbody> </table>	Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16-bit +/- number	V Z	S	○	○	○	○	○	○	○	○	○	○	○	○	○	○	NS	○	○	○	○	○	○	○	○	○	○	○	○	0~15	○	NL	○	○	○	○	○	○	○	○	○	○	○	○	1~8	○	D	○	○	○	○	○	○	○	○	○*	○*	○	○		○	
Range \ Operand		WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																											
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16-bit +/- number	V Z																																																																												
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																												
NS	○	○	○	○	○	○	○	○	○	○	○	○	0~15	○																																																																												
NL	○	○	○	○	○	○	○	○	○	○	○	○	1~8	○																																																																												
D	○	○	○	○	○	○	○	○	○*	○*	○	○		○																																																																												

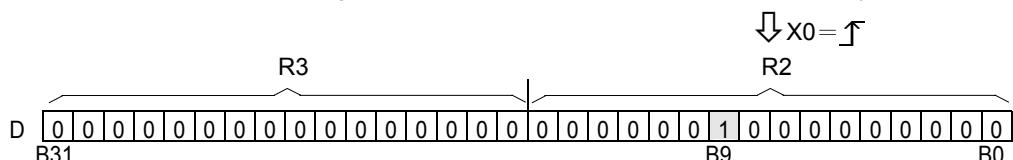
- This instruction, will set a single bit among the total of  $2^{NL}$  discrete points (D) to 1 and the others bit are set to 0. The bit number to be set to 1 is specified by the value comprised by  $BN_S \sim BN_S + NL - 1$  of S (which is called the decode value,  $BN_S$  is the starting bit of the decode value, and  $BN_S + NL - 1$  is the end value),.
- When decode control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, will take out the value  $BN_S \sim BN_S + NL - 1$  from S. And with this value to locate the bit position and set D accordingly, and set all the other bit to zero
- This instruction only provides 16 bit operand, which means S only has B0~B15. Therefore the effective range of NS is 0~15, and the NL length of the decode value is limited to 1~8 bits. Therefore the width of the decoded result D is  $2^{1-8}$  points = 2~256 points = 1~16 words (if 16 points are not sufficient, 1 word is still occupied). If the value of NS or NL is beyond the above range, will set the range-error flag "ERR" to 1, and do not carry out this instruction.
- If the end bit value exceeds the B15 of S, then will extend toward B0 of S + 1. However if this occurs then S+1 can't exceed the range of specific type of operand (ie. If S is of D type register then S+1 can't be D3072). If violate this, then this instruction only takes out the bits from starting bit BNs to its highest limit as the decode value.



- The instruction at left takes out the data of five successive bits from X3 to X7 within the WX0 register and decodes it. The results are then stored in the 32-bit register starting at R2.



Length of decode value NL=5, so bit value is formed by X7~X3 (equal 9)



Because NL=5, the width of D is  $2^5 = 32$  point = 2 word. That is, D is formed by R3R2, and the decoded value is 01001=9, therefore B9 (the 10th point) within D is set to 1, and all other points are 0.

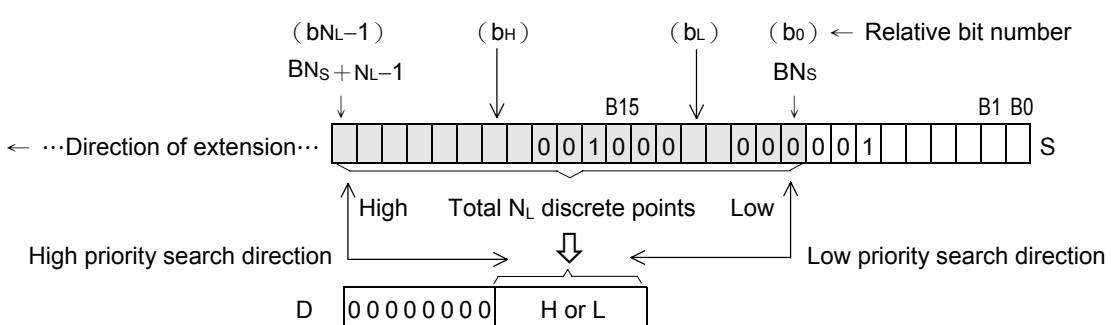
FUN 58 <b>P</b> ENCOD	ENCODE	FUN 58 <b>P</b> ENCOD
<p>Encode control—EN↑</p> <p>High/Low priority—H/L</p>	<p>58P.ENCOD—</p> <p>S : Starting register to be encoded</p> <p>N<sub>S</sub> : Bit position within S as the encoding start point</p> <p>N<sub>L</sub> : Number of encoding discrete points (2~256)</p> <p>D : Number of register storing encoding results (1 word)</p> <p>ERR—Range error</p>	<p>S : Starting register to be encoded</p> <p>N<sub>S</sub> : Bit position within S as the encoding start point</p> <p>N<sub>L</sub> : Number of encoding discrete points (2~256)</p> <p>D : Number of register storing encoding results (1 word)</p> <p>S, N<sub>S</sub>, N<sub>L</sub>, D may combine with V, Z to serve indirect address application</p>

Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16-bit +/- number	V ` Z
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N <sub>S</sub>	○	○	○	○	○	○	○	○	○	○	○	○	○	0~15
N <sub>L</sub>	○	○	○	○	○	○	○	○	○	○	○	○	○	2~256
D		○	○	○	○	○	○		○	○*	○*	○		○

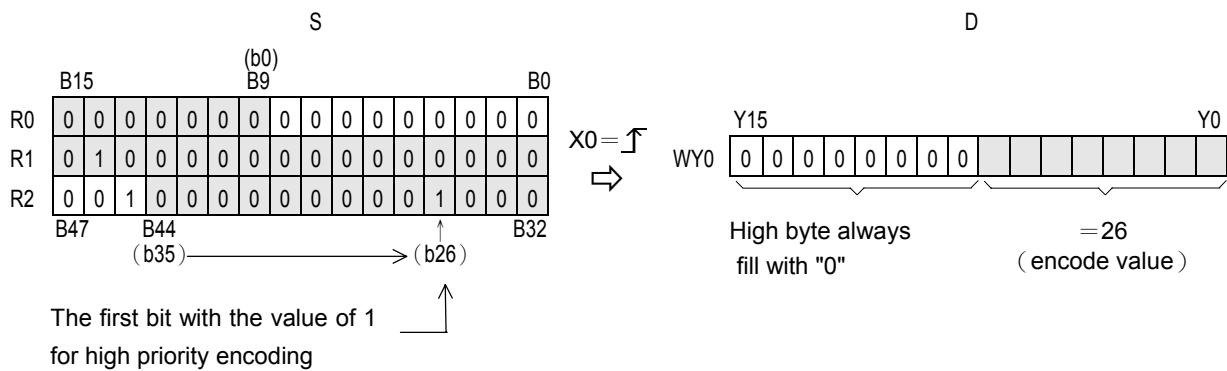
- When encode control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, will starting from the points specified by N<sub>S</sub> within S, take out towards the left (high position direction) N<sub>L</sub> number of successive bits BN<sub>S</sub>~BN<sub>S</sub>+N<sub>L</sub>-1 (BN<sub>S</sub> is called the encoding start point, and its relative bit number is b<sub>0</sub>; BN<sub>S</sub>+N<sub>L</sub>-1 is called the encoding end point, and its relative bit number is BN<sub>L</sub>-1). From left to right do higher priority (when H/L=1) encoding or from right to left do lower priority (when H/L=0) encoding (i.e. seek the first bit with the value of 1, and the relative bit number of this point will be stored into the low byte (B0~B7) of encoded resultant register D, and the high byte of D will be filled with 0.



- As shown in the diagram above, for high priority encoding, the bit first to find is b<sub>H</sub> (with a value of 12), and for low priority encoding, the bit first to find b<sub>L</sub> (with a value of 4). Among the N<sub>L</sub> discrete points there must be at least one bit with value of 1. If all bits are 0, will not to carry out this instruction, and the all zero flag "D=0" will set to 1.
- Because S is a 16-bit register, N<sub>S</sub> can be 0~15, and is used to assign a point of B0~B15 within S as the encoding start point (b<sub>0</sub>). The value of N<sub>L</sub> can be 2~256, and it is used to identify the encoding end point, i.e. it assigns N<sub>L</sub> successive single points starting from the start point (b<sub>0</sub>) towards the left (high position direction) as the encoding zone (i.e. b<sub>0</sub>~bNL-1). If the value of N<sub>S</sub> or N<sub>L</sub> exceeds the above value, then do not carry out this instruction, and set the range-error flag "ERR" as 1.

## Code conversion instructions

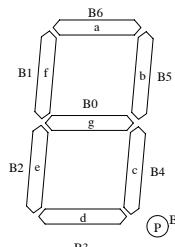
FUN 58 P ENCOD	ENCODE	FUN 58 P ENCOD
	<ul style="list-style-type: none"> <li>If the encoding end point (<math>bNL-1</math>) beyond the B15 of S, then continue extending towards S+1, S+2, but it must not exceed the range of specific type of operand. If it goes beyond this, then this instruction can only take the discrete points between b0 and the highest limit into account for encoding.</li> </ul> <p><b>58P.ENCOD</b></p> <pre>     graph LR         X0 --&gt; EN[EN]         HLL[H/L] --&gt; EN         EN --&gt; 58P["58P.ENCOD"]         58P -- D0 --&gt; D0[D=0]         58P -- ERR --&gt; ERR[ERR]     </pre>	<ul style="list-style-type: none"> <li>The instruction at left is a high priority encode example. When X0 goes from 0 to 1, will take out toward left 36 successive bits starting from B9 (b0) specified by Ns within S, and perform high priority encoding (because H/L = 1). That is, starting from b35 (encoding end point), move right to find the first bit with the value of 1. The resultant value of this example is b26, so the value of D is 001AH=26, as shown in the diagram below.</li> </ul>



FUN 59 <b>P</b> →7SG	7-SEGMENT CONVERSION												FUN 59 <b>P</b> →7SG																																																																											
S : Source data to be converted																																																																																								
N : The nibble number within S for conversion																																																																																								
Conversion control—EN↑ 59P.→7SG																																																																																								
S :                   ERR—N value error      D : Register storing 7-segment result																																																																																								
N :                   S, N, D may combine with V, Z to serve indirect address application																																																																																								
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding-right: 10px;">Range</th> <th>WX</th><th>WY</th><th>WM</th><th>WS</th><th>TMR</th><th>CTR</th><th>HR</th><th>IR</th><th>OR</th><th>SR</th><th>ROR</th><th>DR</th><th>K</th><th>XR</th> </tr> <tr> <th style="text-align: left; padding-right: 10px;">Oper- rand</th> <td>WX0   WX240</td><td>WY0   WY240</td><td>WM0   WM1896</td><td>WS0   WS984</td><td>T0   T255</td><td>C0   C255</td><td>R0   R3839</td><td>R3840   R3903</td><td>R3904   R3967</td><td>R3968   R4167</td><td>R5000   R8071</td><td>D0   D3071</td><td>16-bit +/- number</td><td>V ` Z</td> </tr> </thead> <tbody> <tr> <td style="padding-right: 10px;">S</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr> <tr> <td style="padding-right: 10px;">N</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>0~3</td><td>○</td></tr> <tr> <td style="padding-right: 10px;">D</td><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td><td>○</td><td>○*</td><td>○*</td><td>○</td><td></td><td>○</td></tr> </tbody> </table>														Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- rand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16-bit +/- number	V ` Z	S	○	○	○	○	○	○	○	○	○	○	○	○	○	○	N	○	○	○	○	○	○	○	○	○	○	○	○	0~3	○	D		○	○	○	○	○	○		○	○*	○*	○		○
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																										
Oper- rand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16-bit +/- number	V ` Z																																																																										
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																										
N	○	○	○	○	○	○	○	○	○	○	○	○	0~3	○																																																																										
D		○	○	○	○	○	○		○	○*	○*	○		○																																																																										
<ul style="list-style-type: none"> <li>● When conversion control "EN" = 1 or "EN ↑" (<b>P</b> instruction) has a transition from 0 to 1, will convert N+1 number of nibbles (A nibble is comprised by 4 successive bits, so B0~B3 of S form nibble 0, B4~B7 form nibble 1, etc...)within S to 7-segment code, and store the code into a low byte of D (High bytes does not change). The 7 segment within D are put in sequence, with "a" segment placed at B6, "b" segment at B5, .... , "g" segment at B0. B7 is not used and is fixed as 0. For details please refer the "7-segment code and display pattern table" shown in page 9-31.</li> <li>● Because this instruction is limited to 16 bits, and S only has 4 nibbles (NB0~NB3), the effective range of N is 0~3. Beyond this range, will set the N value flag error "ERR" to 1, and does not carry out this instruction.</li> <li>● Care should be taken on total nibbles to be converted is N+1. N=0 means one digit to convert, N=1 means two digits to convert etc...</li> <li>● When using the FATEK 7-segment expansion module(FB-7SG) and the FUN84 (7SEG0) handy instruction for mixing decoding and non-decoding application, FUN59 and FUN84 can be combined to simplify the program design.(Please refer the example in chapter 17)</li> </ul>																																																																																								

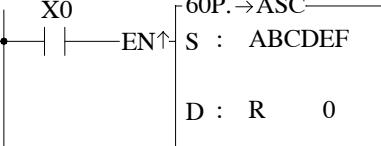
## Code conversion instructions

FUN 59 P →7SG	7-SEGMENT CONVERSION	FUN 59 P →7SG
<b>Example 1</b> When M1 OFF→ON, convert hexadecimal to 7-Segment		
	<b>59P.→7SG</b> S : R0 N : 0 D : R100	<ul style="list-style-type: none"> <li>Figure left shown the conversion of first digit(nibble) of R0 to 7-segment and store in low byte of R100, the high byte of R100 remain unchanged.</li> </ul>
R0=0001H	Original R100=0000H → R100=0030H (1)	
<b>Example 2</b> When M1 ON, convert the hexadecimal to 7-Segment		
	<b>59.→7SG</b> S : R0 N : 1 D : R100	<ul style="list-style-type: none"> <li>Instruction at left will convert the first and the second digit of R0 to 7-segment and store in R100.</li> <li>The low byte of R100 stores first digit.</li> <li>The high byte of R100 stores second digit.</li> </ul>
R0=0056H	→ R100=5B5FH (56)	
<b>Example 3</b> When M1 ON, converting hexadecimal to 7-Segment		
	<b>59.→7SG</b> S : R0 N : 2 D : R100	<ul style="list-style-type: none"> <li>Instruction at left will convert the first, second and third digit of R0 to 7-segment and store in R100 and R101.</li> <li>The low byte of R100 stores first digit.</li> <li>The high byte of R100 stores second digit.</li> <li>The low byte of R101 stores third digit.</li> <li>The high byte of R101 remain unchanged.</li> </ul>
R0=0A48H	Original R101=0000H → R100=337FH (48) R101=0077H (A)	
<b>Example 4</b> When M1 ON, convert hexadecimal to 7-Segment		
	<b>59.→7SG</b> S : R0 N : 3 D : R100	<ul style="list-style-type: none"> <li>Instruction at left will convert 1~4 digit of R0 to 7-segment and store in R100 and R101.</li> <li>The low byte of R100 stores first digit.</li> <li>The high byte of R100 stores second digit.</li> <li>The low byte of R101 stores third digit.</li> <li>The high byte of R101 stores 4<sup>th</sup> digit.</li> </ul>
R0=2790H	→ R100=7B7EH (90) R101=6D72H (27)	

FUN 59 P →7SG		7-SEGMENT CONVERSION								FUN 59 P →7SG	
Nibble data of S		7-segment display format 	Low byte of D								Display pattern
Hexadecimal number	Binary number		B7 ●	B6 a	B5 b	B4 c	B3 d	B2 e	B1 f	B0 g	
0	0000		0	1	1	1	1	1	1	0	0
1	0001		0	0	1	1	0	0	0	0	1
2	0010		0	1	1	0	1	1	0	1	2
3	0011		0	1	1	1	1	0	0	1	3
4	0100		0	0	1	1	0	0	1	1	4
5	0101		0	1	0	1	1	0	1	1	5
6	0110		0	1	0	1	1	1	1	1	6
7	0111		0	1	1	1	0	0	1	0	7
8	1000		0	1	1	1	1	1	1	1	8
9	1001		0	1	1	1	1	0	1	1	9
A	1010		0	1	1	1	0	1	1	1	A
B	1011		0	0	0	1	1	1	1	1	b
C	1100		0	1	0	0	1	1	1	0	C
D	1101		0	0	1	1	1	1	0	1	d
E	1110		0	1	0	0	1	1	1	1	E
F	1111		0	1	0	0	0	1	1	1	F

7-segment display pattern table

## Code conversion instructions

FUN 60 P →ASC	ASCII CONVERSION	FUN 60 P →ASC																																																								
	<p style="text-align: center;">60P.→ASC</p> <p>Conversion control—EN↑</p> <div style="border: 1px solid black; padding: 5px; display: inline-block;">           S : ΔΔΔΔΔΔΔΔΔ            ΔΔΔΔ            D :         </div>	<p>S : Alphanumerics to be converted into ASCII code</p> <p>D : Starting register storing ASCII results</p>																																																								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #cccccc;">Range</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>Alphanumeric</th> </tr> </thead> <tbody> <tr> <td rowspan="2" style="background-color: #cccccc;">Oper- and</td> <td>WY0   WY240</td> <td>WM0   WM1896</td> <td>WS0   WS984</td> <td>T0   T255</td> <td>C0   C255</td> <td>R0   R3839</td> <td>R3904   R3967</td> <td>R3968   R4167</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td>1~12 alphanumeric</td> </tr> <tr> <td></td> </tr> <tr> <td style="background-color: #cccccc;">S</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○</td> </tr> <tr> <td style="background-color: #cccccc;">D</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> </tr> </tbody> </table>	Range	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	Alphanumeric	Oper- and	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	1~12 alphanumeric											S										○	D	○	○	○	○	○	○	○	○*	○*	○	
Range	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	Alphanumeric																																															
Oper- and	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	1~12 alphanumeric																																															
S										○																																																
D	○	○	○	○	○	○	○	○*	○*	○																																																
		<ul style="list-style-type: none"> <li>● When conversion control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, will convert alphabets and numbers stored in S (S has a maximum of 12 alphanumeric character) into ASCII and store it into registers starting from D. Each 2 alphanumeric characters occupy one 16-bit register.</li> <li>● The application of this instruction, most often, stores alphanumeric information within a program, and waits until certain conditions occur, then converts this alphanumeric information into ASCII and conveys it to external display devices which can accept ASCII code.</li> </ul>																																																								
	 <div style="border: 1px solid black; padding: 5px; display: inline-block;">           60P.→ASC            S : ABCDEF            D : R 0         </div>	<ul style="list-style-type: none"> <li>● The instruction at left converts the 6 alphabets -ABCDEF into ASCII then stores it into 3 successive registers starting from R0.</li> </ul>																																																								
	<p style="text-align: center;">S</p> <p>Alphabet ABCDEF</p> <p>X0 →</p>	<p style="text-align: center;">D</p> <table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <thead> <tr> <th></th> <th style="text-align: center;">High Byte</th> <th style="text-align: center;">Low Byte</th> </tr> </thead> <tbody> <tr> <td>R0</td> <td style="text-align: center;">42 (B)</td> <td style="text-align: center;">41 (A)</td> </tr> <tr> <td>R1</td> <td style="text-align: center;">44 (D)</td> <td style="text-align: center;">43 (C)</td> </tr> <tr> <td>R2</td> <td style="text-align: center;">46 (F)</td> <td style="text-align: center;">45 (E)</td> </tr> </tbody> </table>		High Byte	Low Byte	R0	42 (B)	41 (A)	R1	44 (D)	43 (C)	R2	46 (F)	45 (E)																																												
	High Byte	Low Byte																																																								
R0	42 (B)	41 (A)																																																								
R1	44 (D)	43 (C)																																																								
R2	46 (F)	45 (E)																																																								

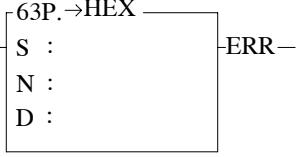
FUN 61 P →SEC	Hour:Minute:Second to Seconds Conversion	FUN 61 P →SEC																																																						
	<p>Conversion control—EN↑</p> <p>S : Starting calendar data register to be converted D : Starting register storing results</p>																																																							
	<table border="1"> <thead> <tr> <th>Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> </tr> </thead> <tbody> <tr> <td>Operand</td> <td>WX0   WX240</td> <td>WY0   WY240</td> <td>WM0   WM1896</td> <td>WS0   WS984</td> <td>T0   T255</td> <td>C0   C255</td> <td>R0   R3839</td> <td>R3840   R3903</td> <td>R3904   R3967</td> <td>R3968   R4167</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td>-117968399   117964799</td> </tr> <tr> <td>S</td> <td>○</td> </tr> <tr> <td>D</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	-117968399   117964799	S	○	○	○	○	○	○	○	○	○	○	○	○	D		○	○	○	○	○		○	○*	○*	○		
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K																																											
Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	-117968399   117964799																																											
S	○	○	○	○	○	○	○	○	○	○	○	○																																												
D		○	○	○	○	○		○	○*	○*	○																																													
	<ul style="list-style-type: none"> <li>When conversion control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, will convert the hour: minute: second data of S~S+2 into an equivalent value in seconds and store it into the 32-bit register formed by combining D and D+1. If the result = 0, then set the "D = 0" flag as 1.</li> <li>Among the FB-PLC instructions, the hour: minute: second time related instructions (FUN61 and 62) use 3 words of register to store the time data, as shown in the diagram below. The first word is the second register, the second word is the minute register, and finally the third word is the hour register, and in the 16 bits of each register, only B14~B0 are used to represent the time value. While bit B15 is used to express whether the time values are positive or negative. When B15 is 0, it represents a positive time value, and when B15 is 1 it represents a negative time value. The B14~B0 time value is represented in binary, and when the time value is negative, B14~B0 is represented with the 2's complement. The number of seconds that results from this operation is the result of summation of seconds from the three registers representing hours: minutes: seconds.</li> </ul>																																																							
	<p>The B15 of each registers is used to represent the sign of each time value</p>	<p>B31 is used to represent the positive or negative nature of the sec. value</p>																																																						
	<ul style="list-style-type: none"> <li>Besides FUN61 or 62 instruction which treat hour: minute: second registers as an integral data, other instructions treat it as individual registers.</li> <li>The example program at below converts the hour: minute: second data formed by R20~R22 into their equivalent value in seconds then stored in the 32-bit register formed by R50~R51. The results are shown below.</li> </ul>	<p>S { R20 : 0E11H = 3601 sec R21 : FD2FH = -721 min R22 : 03F3H = 1011 hr</p> <p>↓ X0 = 1</p> <p>D { R50 : EE45H R51 : 0036H } = 3599941 sec</p>																																																						

Code conversion instructions

FUN 62 P →HMS	SECOND→HOUR : MINUTE : SECOND	FUN 62 P →HMS																																																																																																																																																																																																																																																																																																																																	
Conversion control →EN↑	<p>62P.→HMS</p> <p>S :                      D=0 — Result as 0  D :                      OVR— Over range</p>	<p>S : Starting register of second to be converted</p> <p>D : Starting register storing result of conversion (hour : minute : second)</p>																																																																																																																																																																																																																																																																																																																																	
	<table border="1"> <thead> <tr> <th>Range \ Oper- and</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> </tr> </thead> <tbody> <tr> <td>WX0</td> <td>○</td> <td>-117968399</td> </tr> <tr> <td>WX240</td> <td>○</td> <td>117964799</td> </tr> <tr> <td>WY0</td> <td>○</td> <td></td> </tr> <tr> <td>WY240</td> <td>○</td> <td></td> </tr> <tr> <td>WM0</td> <td>○</td> <td></td> </tr> <tr> <td>WM1896</td> <td>○</td> <td></td> </tr> <tr> <td>WS0</td> <td>○</td> <td></td> </tr> <tr> <td>WS984</td> <td>○</td> <td></td> </tr> <tr> <td>T0</td> <td>○</td> <td></td> </tr> <tr> <td>T255</td> <td>○</td> <td></td> </tr> <tr> <td>C0</td> <td>○</td> <td></td> </tr> <tr> <td>C255</td> <td>○</td> <td></td> </tr> <tr> <td>R0</td> <td>○</td> <td></td> </tr> <tr> <td>R3840</td> <td>○</td> <td></td> </tr> <tr> <td>R3904</td> <td>○</td> <td></td> </tr> <tr> <td>R3968</td> <td>○</td> <td></td> </tr> <tr> <td>R4167</td> <td>○</td> <td></td> </tr> <tr> <td>R5000</td> <td>○</td> <td></td> </tr> <tr> <td>R8071</td> <td>○</td> <td></td> </tr> <tr> <td>D0</td> <td>○</td> <td></td> </tr> <tr> <td>D3071</td> <td>○</td> <td></td> </tr> <tr> <td>K</td> <td>○</td> <td></td> </tr> </tbody> </table>	Range \ Oper- and	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	WX0	○	○	○	○	○	○	○	○	○	○	○	○	-117968399	WX240	○	○	○	○	○	○	○	○	○	○	○	○	117964799	WY0	○	○	○	○	○	○	○	○	○	○	○	○		WY240	○	○	○	○	○	○	○	○	○	○	○	○		WM0	○	○	○	○	○	○	○	○	○	○	○	○		WM1896	○	○	○	○	○	○	○	○	○	○	○	○		WS0	○	○	○	○	○	○	○	○	○	○	○	○		WS984	○	○	○	○	○	○	○	○	○	○	○	○		T0	○	○	○	○	○	○	○	○	○	○	○	○		T255	○	○	○	○	○	○	○	○	○	○	○	○		C0	○	○	○	○	○	○	○	○	○	○	○	○		C255	○	○	○	○	○	○	○	○	○	○	○	○		R0	○	○	○	○	○	○	○	○	○	○	○	○		R3840	○	○	○	○	○	○	○	○	○	○	○	○		R3904	○	○	○	○	○	○	○	○	○	○	○	○		R3968	○	○	○	○	○	○	○	○	○	○	○	○		R4167	○	○	○	○	○	○	○	○	○	○	○	○		R5000	○	○	○	○	○	○	○	○	○	○	○	○		R8071	○	○	○	○	○	○	○	○	○	○	○	○		D0	○	○	○	○	○	○	○	○	○	○	○	○		D3071	○	○	○	○	○	○	○	○	○	○	○	○		K	○	○	○	○	○	○	○	○	○	○	○	○	
Range \ Oper- and	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K																																																																																																																																																																																																																																																																																																																						
WX0	○	○	○	○	○	○	○	○	○	○	○	○	-117968399																																																																																																																																																																																																																																																																																																																						
WX240	○	○	○	○	○	○	○	○	○	○	○	○	117964799																																																																																																																																																																																																																																																																																																																						
WY0	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
WY240	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
WM0	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
WM1896	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
WS0	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
WS984	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
T0	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
T255	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
C0	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
C255	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
R0	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
R3840	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
R3904	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
R3968	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
R4167	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
R5000	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
R8071	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
D0	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
D3071	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
K	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																																							
	<p>● When conversion control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, will convert the second data from the S~S+1 32-bit register into the equivalent hour : minute : second time value and store it in the three successive registers D~D+2. All the data in this instruction is represented in binary (if there is a negative value it is represented using the 2's complement.)</p> <p>The bit B31 of the second register is used as the sign bit of the second value.</p>	<p>The bits B15 of each register are used as the sign bit of the hour : minute : second value.</p> <p>As shown in the diagram above, after convert to hour : minute : second value, the minute : second value can only be in the range of -59 to 59, and the hour number can be in the range of -32768 to 32767 hours. Because of this, the maximum limit of D is -32768 hours, -59 minutes, -59 seconds to 32767 hours, 59 minutes, 59 seconds, the corresponding second value of S which is in the range of -117968399 to 117964799 seconds. If the S value exceeds this range, this instruction cannot be carried out, and will set the over range flag "OVR" to 1. If S = 0 then result is 0 flag "D = 0" will be set to 1.</p> <p>The program in the diagram below is an example of this instruction. Please note that the content of the registers are denoted by hexadecimal, and on the right is its equivalent value in decimal notation.</p>																																																																																																																																																																																																																																																																																																																																	
		<p>R0 5D17H } 6315287 sec  R1 0060H }</p> <p>↓ X0 = 1</p> <p>R10 002FH } 47 sec  R11 000EH } 14 min  R12 06DAH } 1754 hr</p>																																																																																																																																																																																																																																																																																																																																	

FUN 63 <b>P</b> →HEX	Conversion of ASCII code to hexadecimal value	FUN 63 <b>P</b> →HEX
-------------------------	---	-------------------------

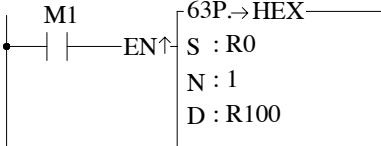
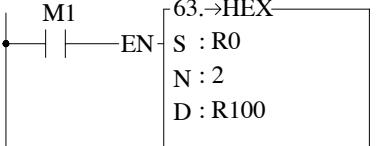
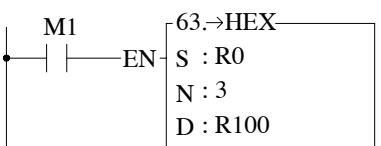
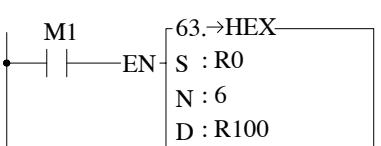
S : Starting source register.  
 N : Number of ASCII codes to be converted to hexadecimal values.  
 D : The starting register that stores the result (hexadecimal value).  
 S, N, D, can associate with V, Z to do the indirect addressing application.



Range \ Oper- and	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16-bit +number	V ` Z
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	1~511	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When conversion control “EN” =1 or “EN ↑ ” (**P** instruction) changes from 0→1, it will convert the N successive hexadecimal ASCII character('0'~'9','A'~'F') convey by 16 bit registers (Low Byte is effective) into hexadecimal value, and store the result into the register starting with D. Every 4 ASCII code is stored in one register. The nibbles of register, which does not involve in the conversion of ASCII code will remain unchanged.
- The conversion will not be performed when N is 0 or greater than 511.
- When there is ASCII error (neither 30H~39H nor 41H~46H), the output “ERR” is ON.
- The main purpose of this instruction is to convert the hexadecimal ASCII character ('0'~'9','A'~'F'), which is received by communication port1 or communication port2 from the external ASCII peripherals, to the hexadecimal values that the CPU can process directly.

## Code conversion instructions

<b>FUN 63 P</b> $\rightarrow$ HEX	Conversion of ASCII code to hexadecimal value	<b>FUN 63 P</b> $\rightarrow$ HEX
<p>⟨ Example 1 ⟩ When M1 from OFF→ON, ASCII code converted to hexadecimal value.</p>		
	<ul style="list-style-type: none"> <li>Converts the ASCII code of R0 into hexadecimal value and store to nibble0 (nibble1~nibble3 remain unchanged) of R100</li> </ul>	
Originally R100=0000H R0=0039H (9) → R100=0009H		
<p>⟨ Example 2 ⟩ When M1 is ON, ASCII code converted to hexadecimal value.</p>		
	<ul style="list-style-type: none"> <li>Converts the ASCII code of R0 and R1 into hexadecimal value and store to low byte (high byte remain unchanged) of R100</li> </ul>	
Originally R100=0000H R0=0039H (9)      R1=0041H (A) → R100=009AH		
<p>⟨ Example 3 ⟩ When M1 is ON, ASCII code converted to hexadecimal value.</p>		
	<ul style="list-style-type: none"> <li>Converts the ASCII code of R0 and R1 into hexadecimal value and store result into R100 (nibble 3 remain unchanged)</li> </ul>	
Originally R100=0000H R0=0039H (9) R1=0041H (A) R2=0045H (E) → R100=09AEH		
<p>⟨ Example 4 ⟩ When M1 is ON, ASCII code converted to hexadecimal value.</p>		
	<ul style="list-style-type: none"> <li>Converts the ASCII code of R0~R5 into hexadecimal value and store it to R100~R101</li> </ul>	
Originally R100=0000H R0=0031H (1) R1=0032H (2) R2=0033H (3) R3=0034H (4) R4=0035H (5) → R100=3456H R5=0036H (6)      R101=0012H		

FUN 64 <b>P</b> →ASCII	Conversion of hexadecimal value to ASCII code	FUN 64 <b>P</b> →ASCII
---------------------------	---	---------------------------

64P.→ASCII

Conversion control—EN↑

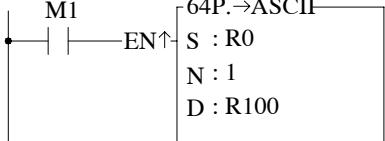
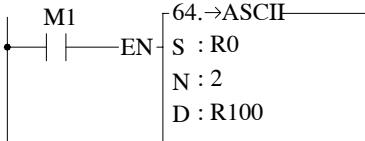
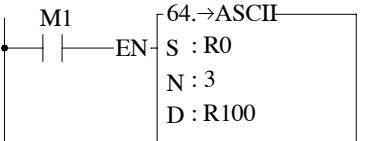
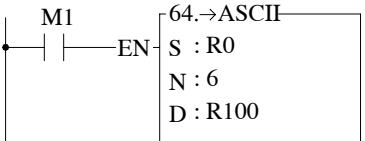
S :	S : Starting source register
N :	N : Number of hexadecimal digit to be converted to ASCII code.
D :	D : The starting register storing result.

S, N, D, can associate with V, Z to do the indirect addressing application.

Oper- and	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
		WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16-bit + number	V 、 Z
S		○	○	○	○	○	○	○	○	○	○	○	○		○
N		○	○	○	○	○	○	○	○	○	○	○	○	1~511	○
D			○	○	○	○	○	○		○*	○*	○			○

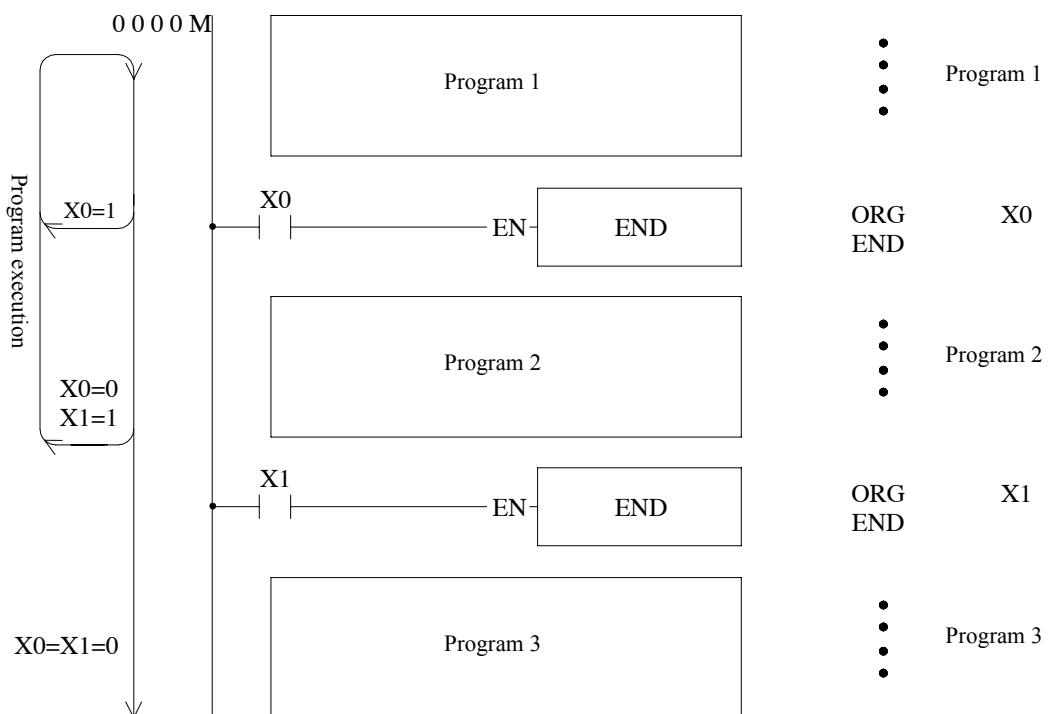
- When conversion control “EN” =1 or “EN ↑” (**P** instruction) changes from 0→1, will convert the N successive nibbles of hexadecimal value in registers start from S into ASCII code, and store the result to low byte (high byte remain unchanged) of the registers which start from D.
- The conversion will not be performed when the value of N is 0 or greater than 511.
- The main purpose of this instruction is to convert the numerical value data, which PLC has processed, to ASCII code and transmit to ASCII peripherals by communication port1 or communication port 2.

## Code conversion instructions

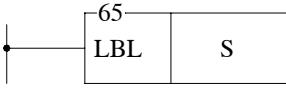
<b>FUN 64 P</b> $\rightarrow$ ASCII	Conversion of hexadecimal value to ASCII code	<b>FUN 64 P</b> $\rightarrow$ ASCII
		⟨ Example 1 ⟩ When M1 changes from OFF→ON, it converts hexadecimal value to ASCII code.
 <ul style="list-style-type: none"> <li>• Converts the Nibble 0 of R0 to ASCII code and stores it into R100 (High byte does not change).</li> </ul>		
R0=0009H	$\Rightarrow$	R100=0039H (9)
		⟨ Example 2 ⟩ When M1 is ON, it converts hexadecimal value to ASCII code.
 <ul style="list-style-type: none"> <li>• Converts the NB0~NB1 of R0 to ASCII code and stores it into R100 ~ R101 (high bytes remain unchanged).</li> </ul>		
R0=009AH	$\Rightarrow$	R100=0039H (9) R101=0041H (A)
		⟨ Example 3 ⟩ When M1 is ON, it converts hexadecimal value to ASCII code.
 <ul style="list-style-type: none"> <li>• Converts the NB0~NB2 of R0 to ASCII code and stores it into R100~R102</li> </ul>		
R0=0123H	$\Rightarrow$	R100=0031H (1) R101=0032H (2) R102=0033H (3)
		⟨ Example 4 ⟩ When M1 is ON, it converts hexadecimal value to ASCII code.
 <ul style="list-style-type: none"> <li>• Converts the NB0~NB5 of R0~R1 to ASCII code and stores it into R100~R105</li> </ul>		
R0=3456H R1=0012H	$\Rightarrow$	R100=0031H (1) R101=0032H (2) R102=0033H (3) R103=0034H (4) R104=0035H (5) R105=0036H (6)

END	PROGRAM END	END
End control -EN	END	No operand

- When end control "EN" = 1, this instruction is activated. Upon executing the END instruction and "EN" = 1, the program flow will immediately returns to the starting point (0000M) to restart the next scan – i.e. all the programs after the END instruction will not be executed. When "EN" = 0, this instruction is ignored, and programs after the END instruction will continue to be executed as the END instruction is not exist.
- This instruction may be placed more than one point within a program, and its input (end control "EN") controls the end point of program execution. It is especially useful for debugging and for testing.
- It's not necessary to put any END instructions in the main program, CPU will automatic restart to start point when reach the end of main program.



## Flow control instructions

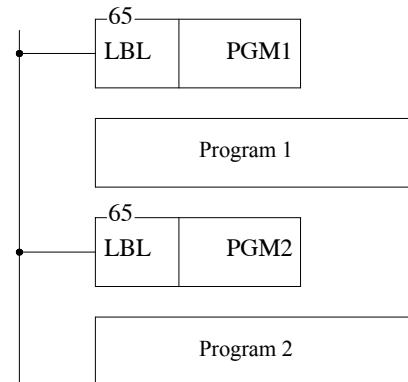
FUN 65 LBL	LABEL	FUN 65 LBL
	S : Alphanumeric, 1~6 characters	

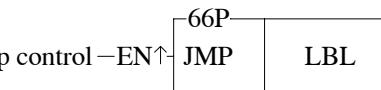
- This instruction is used to make a tag on certain address within a program, to provide a target address for execution of JUMP, CALL instruction and interrupt service. It also can be used for document purpose to improve the readability and interpretability of the program.
- This instruction serves only as the program address marking to provide the control of procedure flow or for remark. The instruction itself will not perform any actions; whether the program contains this instruction or not, the result of program execution will not be influenced by this instruction.
- The label name can be formed by any 1~6 alphanumeric characters and can't be duplicate in the same program. The following label names are reserved for interrupt function usage. These "reserved words", can't be used for normal program labels.

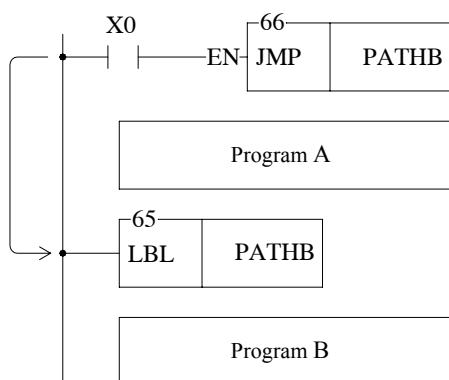
Reserved words	Description
X0+I~X15+I (INT0~INT15)	labels for external input (X0~X15) interrupt
X0-I~X15-I (INT0~INT15-)	service routine.
HSC0I~HSC7I	labels for high speed counter HSC0~HSC7 interrupt service routine.
1MSI (1MS) , 2MSI (2MS) , 3MSI (3MS) , 4MSI (4MS) , 5MSI (5MS) , 10MSI (10MS) , 50MSI (50MS) , 100MSI (100MS)	Labels for 8 kinds of internal timer interrupt service routine.
HSTAI (ATMRI)	Label for High speed fixed timer interrupt service routine.
PSO0I~PSO3I	Labels for the pulse output command finished interrupt service routine.

Only the interrupt service routine can use the label names listed on above table, if mistaken on using the reserved label on the normal subroutine can cause the CPU fail or unpredictable operation.

The label of following diagram illustration served only as program remarks (it is not treated as a label for call or jump target). For the application of labeling in jump control, please refer to JMP instruction for explanation. As to the labeling serves as subroutine names, please refer to CALL instruction for details.



FUN 66 <b>P</b> JMP	JUMP	FUN 66 <b>P</b> JMP
	 LBL : The program label to be jumped	
	<ul style="list-style-type: none"> <li>When jump control "EN"=1 or "EN ↑" (<b>P</b> instruction) changes from 0→1, PLC will jump to the location behind the marked label and continuous to execute the program.</li> <li>This instruction is especially suit for the applications where some part of the program will be executed only under certain condition. This can shorter the scan time while not executes the whole program.</li> <li>This instruction allows jump backward (i.e. the address of LBL is comes before the address of JMP instruction). However, care should be taken if the jump action cause the scan time exceed the limit set by the watchdog timer, the WDT interrupt will be occurred and stop executing.</li> <li>The jump instruction allows only for jumping among main program or jumping among subroutine area, it can't jump across main/subroutine area.</li> </ul>	



- In the left diagram, when X0=1, the program will jump directly to the LBL position named PATHB and continuing to execute program B. Therefore it will skip the program A and none of the instructions of program A will be executed. The status of registers and the coils associated with program A will keep unchanged (as if there is no program section A).

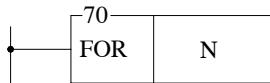
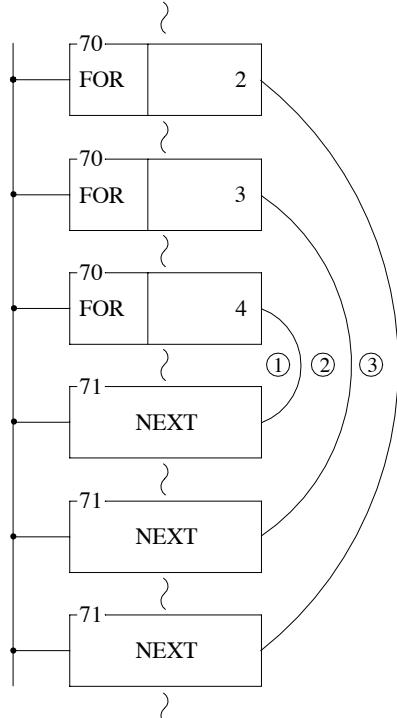
## Flow control instructions

FUN 67 <b>P</b> CALL	CALL	FUN 67 <b>P</b> CALL
LBL : The subroutine label name to be called.		
<ul style="list-style-type: none"> <li>When call control “EN”=1 or “EN ↑” (<b>P</b> instruction) changes from 0→1, PLC will call (perform) the subroutine bear the same label name as the one being called. When execute the subroutine, the program will execute continuous as normal program does but when the program encounter the RTS instruction then the flow of the program will return back to the address immediately after the CALL instruction.</li> </ul>		
<ul style="list-style-type: none"> <li>All the subroutines must end with one “return from subroutine instruction RTS” instruction; otherwise it will cause executing error or CPU shut down. Nevertheless, an RTS instruction can be shared by subroutines (so called as multiple entering subroutines; even though the entry points are different, they have a same returning path) as illustrated in the right diagram subroutine SUB1~3.</li> </ul>		
<ul style="list-style-type: none"> <li>When main program called a subroutine, the subroutine also can call the other subroutines (so called the nested subroutines) for up to 5 levels at the most (include the interrupt routine).</li> </ul>		
<ul style="list-style-type: none"> <li>Interrupt service programs (HSC0I~HSC7I, PSO0I~PSO3I, X0+I~X15+I, INT0~INT15, X0-I~X15-I, INT0~INT15-, HSTAI/ATMRI, 1MSI/1MS, 2MSI/2MS, 3MSI/3MS, 4MSI/4MS, 5MSI/5MS, 10MSI/10MS, 50MSI/50MS, 100MSI/100MS) are also a kind of subroutine. It is also placed in sub program area. However, the calling of interrupt service program is triggered off by the signaling of hardware to make the CPU perform the corresponding interrupt service program (which we called as the calling of the interrupt service program). The interrupt service program can also call subroutine or interrupted by other interrupts with higher priority. Since it is also a subroutine (which occupied one level), it can only call or interrupted by 4 levels of subroutine or interrupt service program. Please refer to RTI instruction for explanation.</li> </ul>		

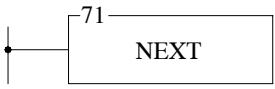
FUN 68 RTS	RETURN FROM SUBROUTINE	FUN 68 RTS
		
<ul style="list-style-type: none"> <li>● This instruction is used to represent the end of a subroutine. Therefore it can only appear within the subroutine area. Its input side has no control signal, so there is no way to serially connect any contacts. This instruction is self sustain, and is directly connected to the power line.</li> <li>● When PLC encounter this instruction, it means that the execution of a subroutine is finished. Therefore it will return to the address immediately after the CALL instruction, which were previously executed and will continue to execute the program.</li> <li>● If this instruction encounters any of the three flow control instructions MC, SKP, or JMP, then this instruction may not be executed (it will be regarded as not exist). If the above instructions are used in the subroutine and causing the subroutine not to execute the RTS instruction, then PLC will halt the operation and set the M1933( flow error flag) to 1. Therefore, no matter what the flow is going, it must always ensure that any subroutine must be able to execute a matched RTS instruction.</li> <li>● For the usage of the RTS instruction please refer to instructions for the CALL instruction.</li> </ul>		

## Flow control instructions

FUN 69 RTI	RETURN FROM INTERRUPT	FUN 69 RTI
		
<ul style="list-style-type: none"> <li>● The function of this instruction is similar to RTS. Nevertheless, RTS is used to end the execution of sub program, and RTI is used to end the execution of interrupt service program. Please refer to the explanation of RTS instruction.</li> <li>● A RTI instruction can be shared by more than one interrupt service program. The usage is the same as the sharing of an RTS by many subroutines. Please refer to the explanation of CALL instruction.</li> <li>● The difference between interrupts and call is that the sub program name (LBL) of a call is defined by user, and the label name and its call instruction are included in the main program or other sub program. Therefore, when PLC performs the CALL instruction and the input “EN”=1 or “EN ↑” (<b>P</b> instruction) changes from 0→1, the PLC will call (execute) this sub program. For the execution of interrupt service program, it is directly used with hardware signals to interrupt CPU to pause the other less important works, and then to perform the interrupt service program corresponding to the hardware signal (we call it the calling of interrupt service program). In comparing to the call instruction that need to be scanned to execute, the interrupt is a more real time in response to the event of the outside world. In addition, the interrupt service program cannot be called by label name; therefore we preserve the special “reserved words” label name to correspond to the various interrupts offered by PLC (check FUN65 explanation for details). For example, the reserved word X0+I is assigned to the interrupt occurred at input point X0; as long as the sub program contains the label of X0+I, when input point X0 interrupt is occurred (X0: <b>J</b>), the PLC will pause the other lower priority program and jump to the subroutine address which labeled as X0+I to execute the program immediately.</li> <li>● If there is a interrupt occurred while CPU is handling the higher priority (such as hardware high speed counter interrupt) or same priority interrupt program (please refer to Chapter 10 for priority levels), the PLC will not execute the interrupt program for this interrupt until all the higher priority programs were finished.</li> <li>● If the RTI instruction cannot be reached and performed in the interrupt service routine, may cause a serious CPU shut down. Consequently, no matter how you control the flow of program, it must be assured that the RTI instruction will be executed in any interrupt service program.</li> <li>● For the detailed explanation and example for the usage of interrupts, please refer to Chapter 10 for explanation.</li> </ul>		

FUN 70 FOR	FOR	FUN 70 FOR																																									
		N : Number of times of loop execution																																									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th><th style="padding: 2px;">WX</th><th style="padding: 2px;">WY</th><th style="padding: 2px;">WM</th><th style="padding: 2px;">WS</th><th style="padding: 2px;">TMR</th><th style="padding: 2px;">CTR</th><th style="padding: 2px;">HR</th><th style="padding: 2px;">IR</th><th style="padding: 2px;">OR</th><th style="padding: 2px;">SR</th><th style="padding: 2px;">ROR</th><th style="padding: 2px;">DR</th><th style="padding: 2px;">K</th></tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;">Oper- and</td><td style="padding: 2px;">WX0   WX240</td><td style="padding: 2px;">WY0   WY240</td><td style="padding: 2px;">WM0   WM1896</td><td style="padding: 2px;">WS0   WS984</td><td style="padding: 2px;">T0   T255</td><td style="padding: 2px;">C0   C255</td><td style="padding: 2px;">R0   R3839</td><td style="padding: 2px;">R3840   R3903</td><td style="padding: 2px;">R3904   R3967</td><td style="padding: 2px;">R3968   R4167</td><td style="padding: 2px;">R5000   R8071</td><td style="padding: 2px;">D0   D3071</td><td style="padding: 2px;">1   16383</td></tr> <tr> <td style="text-align: left; padding: 2px;">N</td><td style="padding: 2px;">○</td><td style="padding: 2px;">○</td></tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	1   16383	N	○	○	○	○	○	○	○	○	○	○	○	○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K																														
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	1   16383																														
N	○	○	○	○	○	○	○	○	○	○	○	○																															
<ul style="list-style-type: none"> <li>● This instruction has no input control, is connected directly to the power line, and cannot be in series with any conditions.</li> <li>● The programs within the FOR and NEXT instructions form a program loop (the start of the loop program is the next instruction after FOR, and the last is the instruction before NEXT). When PLC executes the FOR instruction, it first records the N value after that instruction (loop execution number), then for N times successively execution from start to last of the programs in the loop. Then it jumps out of the loop, and continues executes the instruction immediately after the NEXT instruction.</li> <li>● The loop can have a nested structure, i.e. the loop includes other loops, like an onion. 1 loop is called a level, and there can be a maximum of 5 levels. The FOR and NEXT instructions must be used in pairs. The first FOR instruction and the last NEXT instruction are the outermost (first) level of a nested loop. The second FOR instruction and the second last NEXT instruction are the second level, the last FOR instruction and the first NEXT instruction form the loop's innermost level.</li> </ul>																																											
		<ul style="list-style-type: none"> <li>● In the example in the diagram at left, loop ① will be executed <math>4 \times 3 \times 2 = 24</math> times, loop ② will be executed <math>3 \times 2 = 6</math> times, and loop ③ will be executed 2 times.</li> <li>● If there is a FOR instruction and no corresponding NEXT instruction, or the FOR and NEXT instructions in the nested loop have not been used in pairs, or the sequence of FOR and NEXT has been misplaced, then a syntax error will be generated and this program may not be executed.</li> <li>● In the loop, the JMP instruction may be used to jump out of the loop. However, care must be taken that once the loop has been entered (and executed to the FOR instruction), no matter how the program flow jumps, it must be able to reach the NEXT instruction before reaching the END instruction or the bottom of the program. Otherwise FB-PLC will halt the operation and show an error message.</li> <li>● The effective range of N is 1~16383 times. Beyond this range FB-PLC will treat it as 1. Care should be taken, if the amount of N is too large and the loop program is too big, a WDT may occur.</li> </ul>																																									

## Flow control instructions

FUN 71 NEXT	LOOP END	FUN 71 NEXT
		
<ul style="list-style-type: none"><li>• This instruction and the FOR instruction together form a program loop. The instruction itself has no input control, is connected directly to the power line, and cannot be in series with any conditions.</li><li>• When PLC has not yet entered the loop (has not yet executed to the FOR instruction, or has executed but then jumped out), but the NEXT instruction is reached, then PLC will not take any action, just as if this instruction did not exist.</li><li>• For the usage of this instruction please refer to the explanations for the FOR instruction on the preceding page.</li></ul>		

FUN 72 TP4	The Convenient instruction for temperature measuring module (Brief description of function)	FUN 72 TP4																	
Execution control —EN	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>72.TP4</td> <td>Tp : PI : Sm : Ym : AR : TR : WR :</td> <td>ERR – Parameter error -ALM – Sensor line breaking</td> </tr> <tr> <td>Y0</td> <td>Y255</td> <td>R0</td> <td>R3839</td> <td>R3840</td> <td>R5000</td> <td>D0</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td>R3903</td> <td>R8071</td> <td>D3071</td> </tr> </table>	72.TP4	Tp : PI : Sm : Ym : AR : TR : WR :	ERR – Parameter error -ALM – Sensor line breaking	Y0	Y255	R0	R3839	R3840	R5000	D0					R3903	R8071	D3071	<p>Tp : Type of Temperature sensor, it can be J or K Type thermo-coupler or PT-100 RTD.</p> <p>PI : Polarity and the voltage range setting for temperature module.</p> <p>Sm : Starting temperature point measured by the temperature module.</p> <p>Ym : Starting output for preserved for controlled temperature measurement module.</p> <p>AR : Analogue input register preserved for controlled temperature module.</p> <p>TR : Starting register for temperature readings storing.</p> <p>WR: Starting working register for this instruction instance.</p>
72.TP4	Tp : PI : Sm : Ym : AR : TR : WR :	ERR – Parameter error -ALM – Sensor line breaking																	
Y0	Y255	R0	R3839	R3840	R5000	D0													
				R3903	R8071	D3071													
Brief description of instruction function																			
<ul style="list-style-type: none"> <li>● This is a dedicate instruction for the FB-J(K)4 or FB-RTD4 multiplexing temperature measurement module. With this instruction, the user can acquire temperature readings by simply fill a table formed by registers. Each instance of instruction can handle one FB-J(K)4 or FB-RTD4 module.</li> <li>● This instruction must incorporate with FB-J(K)4 or FB-RTD4 multiplexing temperature measurement module in its usage. Hereby it introduced briefly about the function of this instruction only. For details of the function, explanation, usages and examples, please refer to Chapter 20 “Temperature measurement of FB-PLC and PID Control”.</li> </ul>																			

## Temperature control instructions 1

FUN 73 TSTC	Convenient instruction for temperature measuring of temperature module + PID temperature control	FUN 73 TSTC																																																																																																																														
<p>Execution control- EN</p> <p>Heating/Cooling-H/C</p>	<p>73.TSTC</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>Tp :</td><td>ERR—Parameter error</td></tr> <tr><td>Pl :</td><td>Temperature sensor line breaking</td></tr> <tr><td>Sm :</td><td>Temperature control warning</td></tr> <tr><td>Ym :</td><td></td></tr> <tr><td>AR :</td><td></td></tr> <tr><td>TR :</td><td></td></tr> <tr><td>Yh :</td><td></td></tr> <tr><td>Sh :</td><td></td></tr> <tr><td>Zh :</td><td></td></tr> <tr><td>Sv :</td><td></td></tr> <tr><td>Os :</td><td></td></tr> <tr><td>PR :</td><td></td></tr> <tr><td>IR :</td><td></td></tr> <tr><td>DR :</td><td></td></tr> <tr><td>OR :</td><td></td></tr> <tr><td>WR :</td><td></td></tr> </table>	Tp :	ERR—Parameter error	Pl :	Temperature sensor line breaking	Sm :	Temperature control warning	Ym :		AR :		TR :		Yh :		Sh :		Zh :		Sv :		Os :		PR :		IR :		DR :		OR :		WR :		<p>Tp : Type of temperature sensor, it can be J or K Type thermo-coupler or PT-100 RTD.</p> <p>PI : Polarity and the voltage range setting for temperature module.</p> <p>Sm: Starting temperature point measured by controlled temperature module.</p> <p>Ym: Starting output point preserved for controlled temperature module.</p> <p>AR: Analogue input register preserve for controlled temperature module.</p> <p>TR: Starting register for temperature readings storing.</p> <p>Yh : Starting point of PWM temperature control output point.</p> <p>Sh : starting temperature point for processing by this instruction instance.</p> <p>Zh : Number of temperature points processed by this instruction instance.</p> <p>Sv : Starting register for temperature setting value storing.</p> <p>Os : Starting register for temperature deviation value storing.</p> <p>PR: Starting register for gain setting value storing.</p> <p>IR : Starting register for integral time constant setting value storing.</p> <p>DR: Starting register for differential time constant setting value storing.</p> <p>OR: Starting register for temperature control value output storing.</p> <p>WR: Starting working register for this instruction instance.</p>																																																																																														
Tp :	ERR—Parameter error																																																																																																																															
Pl :	Temperature sensor line breaking																																																																																																																															
Sm :	Temperature control warning																																																																																																																															
Ym :																																																																																																																																
AR :																																																																																																																																
TR :																																																																																																																																
Yh :																																																																																																																																
Sh :																																																																																																																																
Zh :																																																																																																																																
Sv :																																																																																																																																
Os :																																																																																																																																
PR :																																																																																																																																
IR :																																																																																																																																
DR :																																																																																																																																
OR :																																																																																																																																
WR :																																																																																																																																
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding-bottom: 2px;">Range</th> <th style="text-align: center; padding-bottom: 2px;">Y</th> <th style="text-align: center; padding-bottom: 2px;">HR</th> <th style="text-align: center; padding-bottom: 2px;">IR</th> <th style="text-align: center; padding-bottom: 2px;">DR</th> <th style="text-align: center; padding-bottom: 2px;">ROR</th> <th style="text-align: center; padding-bottom: 2px;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: left; padding-top: 2px;">Oper- and</td> <td style="text-align: center; padding-top: 2px;">Y0   Y255</td> <td style="text-align: center; padding-top: 2px;">R0   R3839</td> <td style="text-align: center; padding-top: 2px;">R3840   R3903</td> <td style="text-align: center; padding-top: 2px;">D0   D3071</td> <td style="text-align: center; padding-top: 2px;">R5000   R8071</td> <td></td> </tr> <tr> <td style="text-align: left;">Tp</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: center;">0~2</td> </tr> <tr> <td style="text-align: left;">Pl</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: center;">0~3</td> </tr> <tr> <td style="text-align: left;">Sm</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: center;"><math>n \times 4</math> n = 0~7</td> </tr> <tr> <td style="text-align: left;">Ym</td> <td style="text-align: center;">○</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: left;">AR</td> <td></td> <td style="text-align: center;">○</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: left;">TR</td> <td style="text-align: center;">○</td> <td></td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td></td> <td></td> </tr> <tr> <td style="text-align: left;">Yh</td> <td style="text-align: center;">○</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: left;">Sh</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: center;">0~23</td> </tr> <tr> <td style="text-align: left;">Zh</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td style="text-align: center;">1~24</td> </tr> <tr> <td style="text-align: left;">Sv</td> <td style="text-align: center;">○</td> <td></td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td></td> <td></td> </tr> <tr> <td style="text-align: left;">Os</td> <td style="text-align: center;">○</td> <td></td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td></td> <td></td> </tr> <tr> <td style="text-align: left;">PR</td> <td style="text-align: center;">○</td> <td></td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td></td> <td></td> </tr> <tr> <td style="text-align: left;">IR</td> <td style="text-align: center;">○</td> <td></td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td></td> <td></td> </tr> <tr> <td style="text-align: left;">DR</td> <td style="text-align: center;">○</td> <td></td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td></td> <td></td> </tr> <tr> <td style="text-align: left;">OR</td> <td style="text-align: center;">○</td> <td></td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td></td> <td></td> </tr> <tr> <td style="text-align: left;">WR</td> <td style="text-align: center;">○</td> <td></td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td></td> <td></td> </tr> </tbody> </table>	Range	Y	HR	IR	DR	ROR	K	Oper- and	Y0   Y255	R0   R3839	R3840   R3903	D0   D3071	R5000   R8071		Tp						0~2	Pl						0~3	Sm						$n \times 4$ n = 0~7	Ym	○						AR		○					TR	○		○	○*			Yh	○						Sh						0~23	Zh						1~24	Sv	○		○	○*			Os	○		○	○*			PR	○		○	○*			IR	○		○	○*			DR	○		○	○*			OR	○		○	○*			WR	○		○	○*				
Range	Y	HR	IR	DR	ROR	K																																																																																																																										
Oper- and	Y0   Y255	R0   R3839	R3840   R3903	D0   D3071	R5000   R8071																																																																																																																											
Tp						0~2																																																																																																																										
Pl						0~3																																																																																																																										
Sm						$n \times 4$ n = 0~7																																																																																																																										
Ym	○																																																																																																																															
AR		○																																																																																																																														
TR	○		○	○*																																																																																																																												
Yh	○																																																																																																																															
Sh						0~23																																																																																																																										
Zh						1~24																																																																																																																										
Sv	○		○	○*																																																																																																																												
Os	○		○	○*																																																																																																																												
PR	○		○	○*																																																																																																																												
IR	○		○	○*																																																																																																																												
DR	○		○	○*																																																																																																																												
OR	○		○	○*																																																																																																																												
WR	○		○	○*																																																																																																																												
<p>Description</p> <ul style="list-style-type: none"> <li>● This instruction is used for the measuring for FB-J(K)4 or FB-RTD4 temperature measuring module and PID temperature control. With this instruction, the user may easily reach multi-points PID loop temperature control by table filling method.</li> <li>● This instruction must incorporate with FB-J(K)4 or FB-RTD4 multiplexing temperature measuring module in its usage. Hereby it introduced briefly about the function of this instruction only. For details of the function, explanation, usages, and examples, please refer to Chapter 20 “Temperature measuring of FB-PLC and PID Control”.</li> </ul>																																																																																																																																

FUN 74 P IMDIO	IMMIDIATE I/O	FUN 74 P IMDIO																
	<p style="text-align: center;">74P.IMDIO</p> <p>Refresh control —EN↑</p> <div style="border: 1px solid black; padding: 5px; display: inline-block;">         D : <input type="text"/>          N : <input type="text"/> </div> <p style="margin-left: 200px;">D : Starting number of I/O points to be refreshed N : Number of I/O points to be refreshed</p>																	
	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th rowspan="2">Oper- and</th> <th>Range</th> <th>X</th> <th>Y</th> <th>K</th> </tr> <tr> <th>Xn of Main Unit.</th> <th>Yn of Main Unit.</th> <th>1   24</th> </tr> </thead> <tbody> <tr> <td>D</td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td><input type="radio"/></td> </tr> <tr> <td>N</td> <td></td> <td></td> <td><input type="radio"/></td> </tr> </tbody> </table>	Oper- and	Range	X	Y	K	Xn of Main Unit.	Yn of Main Unit.	1   24	D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	N			<input type="radio"/>	
Oper- and	Range		X	Y	K													
	Xn of Main Unit.	Yn of Main Unit.	1   24															
D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>															
N			<input type="radio"/>															

- For normal PLC scan cycle, the CPU gets the entire input signals before the program is executed, and then perform the executing of program based on the fresh input signals. After finished the program execution the CPU will update all the output signals according to the result of program execution. Only after the complete scan has been finished will all the output results be transferred all at once to the output. Thus for the input event to output responses, there will be a delay of at least 1 scan time (maximum of 2 scan time). With this instruction, the input signals or output signals specified by this instruction can be immediately refresh to get the faster input to output response without the limitation imposed by the scan method.
- When refresh control "EN" = 1 or "EN ↑" (P instruction) has a transition from 1 to 0, then the status of N input points or output points (D~D+N-1) will be refreshed.
- The I/O points for FB-PLC's immediate I/O are only limited to I/O points on the main unit. The table below shows permissible I/O numbers for 20, 28, and 40 point main units:

Main-unit type Permissible numbers	20 points	28 points	40 points
Input signals	X0~X11	X0~X15	X0~X23
Output signals	Y0~Y7	Y0~Y11	Y0~Y15

- If the intended refresh I/O signals of this instruction is beyond the range of I/O points specified on above table then PLC will be unable to operate and the M1931 error flag will be set to 1. ( for example, if in a program, D=X7, N=10, which means X7 to X16 are to be immediately retrieved. Supposing the main unit is FB-28MB, then its biggest input point is X15, and clearly X16 has already exceeded the main unit's input point number so under such case M1931 error flag will be set to 1).
- With this instruction, PLC can immediately refresh input/output signals. However, the delay of the hardware or the software filter impose on the I/O signals still exist. Please pay attention on this.

## I/O instructions

FUN 75 <b>P</b> FILT	FILTER ADJUST	FUN 75 <b>P</b> FILT
Input control –EN↑		N : Filter time 0~30 (mS)

- This instruction is especially use for the 16 input points X0~X15 of main unit for the software integral (filter) time adjustment. When input control “EN”=1 or “EN ↑ ” (**P** instruction) changes from 0→1, will sets the input filter time to be NmS for the 16 points from X0~X15.
- As a matter of fact, the 16 input points of X0~X15 have all been pre-processed by Dardware Digital Filter to enhance the noise immunity capability. The highest input frequency of X0~X15 that the hardware digital filter can be configured is range from 4KHZ~512KHZ. The best setting can be achieved dynamically according to the field condition.
- Except the 16 input point of X0~X15, all the other input points had been added with roughly 4ms of RC filter circuit to enhance the noise immunity, thus these signal are not suitable for high speed operation. If use this function to set the filter time to zero (default 4ms) and configure the Hardware Digital Filter to Max. frequency (default) then X0~X15 inputs can be used for high speed application.

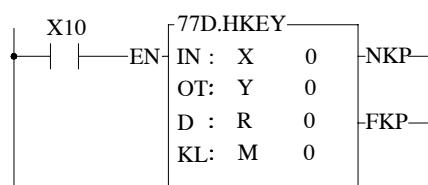
FUN 76 <b>D</b> TKEY	DECIMAL- KEY INPUT	FUN 76 <b>D</b> TKEY																																																																
<p>Input control—EN</p> <p>76D.TKEY</p> <p>IN : X0 Y0 M0 S0 WY0 WM0 WS0 T0 C0 R0 R3904 R3968 R5000 D0 V D : X240 Y240 M1896 S984 WY240 WM1896 WS984 T255 C255 R3839 R3967 R4167 R8071 D3071 Z KL:</p>	<p>KPR—Key-in action</p> <p>IN : Key input point D : register storing key-in numerals KL: starting coil to reflect the input status D may combine with V, Z to serve indirect address application</p>																																																																	
<table border="1"> <thead> <tr> <th>Range</th> <th>X</th> <th>Y</th> <th>M</th> <th>S</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>Operand</td> <td>X0</td> <td>Y0</td> <td>M0</td> <td>S0</td> <td>WY0</td> <td>WM0</td> <td>WS0</td> <td>T0</td> <td>C0</td> <td>R0</td> <td>R3904</td> <td>R3968</td> <td>R5000</td> <td>D0</td> <td>V</td> </tr> <tr> <td></td> <td> </td> <td>Z</td> </tr> <tr> <td>X240</td> <td>Y240</td> <td>M1896</td> <td>S984</td> <td>WY240</td> <td>WM1896</td> <td>WS984</td> <td>T255</td> <td>C255</td> <td>R3839</td> <td>R3967</td> <td>R4167</td> <td>R8071</td> <td>D3071</td> <td></td> <td></td> </tr> </tbody> </table>	Range	X	Y	M	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR	Operand	X0	Y0	M0	S0	WY0	WM0	WS0	T0	C0	R0	R3904	R3968	R5000	D0	V																Z	X240	Y240	M1896	S984	WY240	WM1896	WS984	T255	C255	R3839	R3967	R4167	R8071	D3071				
Range	X	Y	M	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR																																																			
Operand	X0	Y0	M0	S0	WY0	WM0	WS0	T0	C0	R0	R3904	R3968	R5000	D0	V																																																			
															Z																																																			
X240	Y240	M1896	S984	WY240	WM1896	WS984	T255	C255	R3839	R3967	R4167	R8071	D3071																																																					
<ul style="list-style-type: none"> <li>This instruction has designated 10 input points IN~IN+9 (IN0~IN9) to one decimal number entry (IN-&gt;0, IN+1-&gt;1...). According to the key-in sequence (ON) of these input points, it is possible to enter 4 or 8 decimal numbers into the registers specified by D.</li> <li>When input control "EN" = 1, this instruction will monitor the 10 input points starting from IN and put the corresponding number into D register while the key were depressed. It will wait until the input point has released, then monitor the next "ON" input point, and shift in the new number into D register (high digit is older than low digit). For the 16-bit operand, D register can store up to 4 digits, and for the 32-bit operand 8 digits may be stored. When the key numbers full fill the D register, new key-in number will kick out the oldest key number of the D register. The key-in status of the 10 input points starting from IN will be recorded on the 10 corresponding coil starting from KL. These coils will set to 1 while the corresponding key is depressed and remain unchanged even if the corresponding key is released. Until other key is depressed then it will return to zero. As long as any input point is depressed (ON), then the key-in flag KPR will set to 1. Only one of IN0~IN9 key can be depressed at the same time. If more than one is pressed, then the first one is the only one taken. Below is a schematic diagram of the function with 16-bit operand.</li> <li>When input control "EN" = 0, this instruction will not be executed. KPR output and KL coil status will be 0. However, the numerical values of D register will remain unchanged.</li> </ul>		<p>• The instruction at left represents the input point X0 with the number "0", X1 is represented by 1, ... , M0 records the action of X0, M1 records the action of X1 ... , and the input numerical values are stored in the R0 register.</p>																																																																

## I/O instructions

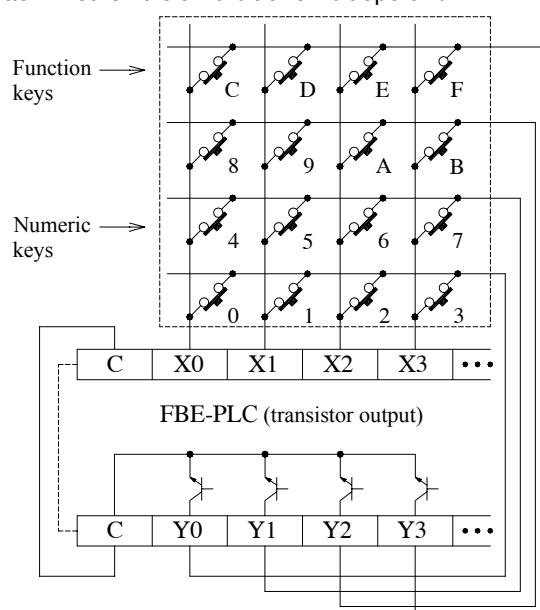
FUN 76 <b>D</b> TKEY	DECIMAL- KEY INPUT	FUN 76 <b>D</b> TKEY																																																																																																																																																																																																																													
The following diagram is the input wiring schematic for this example:																																																																																																																																																																																																																															
<p style="text-align: center;">FBe-PLC input side</p>																																																																																																																																																																																																																															
<ul style="list-style-type: none"> <li>If the X0~X3 key-in sequence follow the ① ② ③ ④ ⑤ ⑥ ⑦ sequence in the following diagram. At step ① and ⑦ the X20 is 0, so there was no key generated, only steps ② ③ ④ ⑤ ⑥ are effective. Because the register can only hold 4 key numbers, Of these 5 steps the first key was kick out. The key strokes 3302 of the steps ③ ④ ⑤ ⑥ are entered in the R0 register.</li> </ul> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>X20</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>X0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>X1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>X2</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>X3</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>M0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>M1</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>M2</td> <td>0</td> <td>1</td> </tr> <tr> <td>M3</td> <td>0</td> </tr> <tr> <td>Y0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>R0</td> <td>0000</td> <td>0001</td> <td>0013</td> <td>0133</td> <td>1330</td> <td>3302</td> <td></td> </tr> </table>			X20	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	X0	0	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	X1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	X2	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	X3	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	M0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	M2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	M3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Y0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	R0	0000	0001	0013	0133	1330	3302													
X20	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																											
X0	0	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0																																																																																																																																																																																																												
X1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0																																																																																																																																																																																																												
X2	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1																																																																																																																																																																																																												
X3	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1																																																																																																																																																																																																												
M0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0																																																																																																																																																																																																												
M1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1																																																																																																																																																																																																												
M2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1																																																																																																																																																																																																												
M3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																												
Y0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0																																																																																																																																																																																																												
R0	0000	0001	0013	0133	1330	3302																																																																																																																																																																																																																									

FUN 77 <b>D</b> HKEY	HEX-KEY INPUT	FUN 77 <b>D</b> HKEY																																																																																																
Execution control—EN	<p>77D.HKEY</p> <table border="1" style="margin-left: 10px;"> <tr> <td>IN :</td> <td>NKP—Number key press</td> </tr> <tr> <td>OT:</td> <td></td> </tr> <tr> <td>D :</td> <td>FKP—Function key press</td> </tr> <tr> <td>KL:</td> <td></td> </tr> </table>	IN :	NKP—Number key press	OT:		D :	FKP—Function key press	KL:		<p>IN : Key scan input point number            OT: Starting Multiplex scan output point (4 points)            D : Register storing "key-in numbers"            KL: Starting relay for key status            D may combine with V, Z to serve indirect address application</p>																																																																																								
IN :	NKP—Number key press																																																																																																	
OT:																																																																																																		
D :	FKP—Function key press																																																																																																	
KL:																																																																																																		
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding-right: 10px;">Range</th> <th>X</th><th>Y</th><th>M</th><th>S</th><th>WY</th><th>WM</th><th>WS</th><th>TMR</th><th>CTR</th><th>HR</th><th>OR</th><th>SR</th><th>ROR</th><th>DR</th><th>XR</th> </tr> </thead> <tbody> <tr> <td style="text-align: left; padding-right: 10px;">Oper- and</td><td>X0   X240</td><td>Y0   Y240</td><td>M0   M1896</td><td>S0   S984</td><td>WY0   WY240</td><td>WM0   WM1896</td><td>WS0   WS984</td><td>T0   T255</td><td>C0   C255</td><td>R0   R3839</td><td>R3904   R3967</td><td>R3968   R4167</td><td>R5000   R8071</td><td>D0   D3071</td><td>V   Z</td> </tr> <tr> <td style="text-align: left; padding-right: 10px;">IN</td><td>○</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td style="text-align: left; padding-right: 10px;">OT</td><td></td><td>○</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td style="text-align: left; padding-right: 10px;">D</td><td></td><td></td><td></td><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○*</td><td>○*</td><td>○</td><td>○</td> </tr> <tr> <td style="text-align: left; padding-right: 10px;">KL</td><td></td><td>○</td><td>○</td><td>○</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> </tbody> </table>	Range	X	Y	M	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR	Oper- and	X0   X240	Y0   Y240	M0   M1896	S0   S984	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	V   Z	IN	○															OT		○														D					○	○	○	○	○	○	○	○*	○*	○	○	KL		○	○	○												
Range	X	Y	M	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR																																																																																			
Oper- and	X0   X240	Y0   Y240	M0   M1896	S0   S984	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	V   Z																																																																																			
IN	○																																																																																																	
OT		○																																																																																																
D					○	○	○	○	○	○	○	○*	○*	○	○																																																																																			
KL		○	○	○																																																																																														

- The numeric (0~9) key function of this instruction is similar as for the TKEY instruction. The hardware connection for TKEY and HKEY is different. For TKEY instruction each key have one input point to connect, while HKEY use 4 input points and 4 output points to form a 4x4 multiplex 16 key input. 4x4 means that there can be 16 input keys, so in addition to the 10 numeric keys, the other 6 keys can be used as function keys (just like the usual discrete input). The actions of the numeric keys and the function keys are independent and have no effect on each other.
- When execution control "EN" = 1, this instruction will scan the numeric keys and function keys in the matrix formed by the 4 input points starting from IN and the 4 output points starting from OT. For the function of the numeric keys and "NKP" output please refer to the TKEY instruction. The function keys maintain the key-in status of the A~F keys in the last 6 relays specified by KL (the first 10 store the key-in status of the numeric keys). If any one of the A~F keys is depressed, FKP (FO1) will set to 1. The OT output points for this instruction must be transistor outputs.
- The biggest number for a 16-bit operand is 4 digits (9999), and for 32-bit operand is 8 digits (99999999). However, there are only 6 function keys (A~F), no matter whether it is a 16-bit or 32-bit operand.



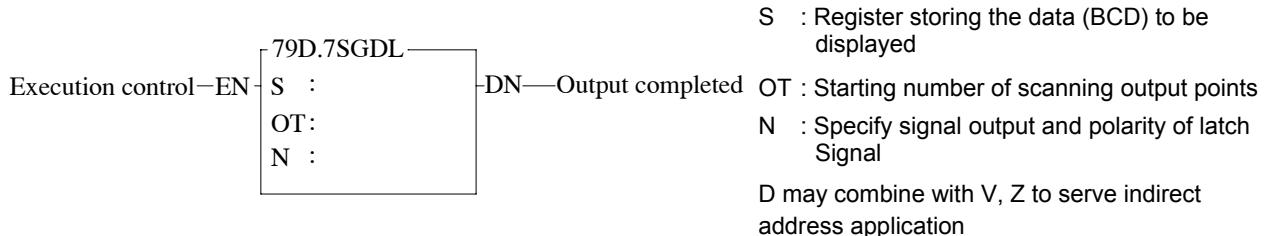
- The instruction in the diagram above uses X0~X3 and Y0~Y3 to form a multiplex key input. It can input numeric values of 8 digits and stores the results in R1R0. The input status of the function keys is stored in M10(A)~M15(F).



## I/O instructions

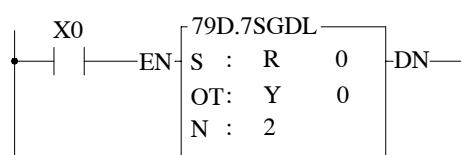
FUN 78 <b>D</b> DSW	DIGITAL SWITCH INPUT	FUN 78 <b>D</b> DSW																																																																																		
<p>Input control—EN—</p> <p>IN : Switch input points OT: Multiplex scan output points (4 points)</p>	<p>DN — Readout completed ERR — Reading error</p> <p>D : register storing readout value D may combine with V, Z to serve indirect address application D</p>	<p>IN : Switch input points OT: Multiplex scan output points (4 points) D : register storing readout value D may combine with V, Z to serve indirect address application D</p>																																																																																		
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; background-color: #cccccc;">Range</th> <th style="text-align: center;">X</th> <th style="text-align: center;">Y</th> <th style="text-align: center;">WY</th> <th style="text-align: center;">WM</th> <th style="text-align: center;">WS</th> <th style="text-align: center;">TMR</th> <th style="text-align: center;">CTR</th> <th style="text-align: center;">HR</th> <th style="text-align: center;">OR</th> <th style="text-align: center;">SR</th> <th style="text-align: center;">ROR</th> <th style="text-align: center;">DR</th> <th style="text-align: center;">XR</th> </tr> <tr> <th style="text-align: center; background-color: #cccccc;">Oper- and</th> <td style="text-align: center;">X0</td> <td style="text-align: center;">Y0</td> <td style="text-align: center;">WY0</td> <td style="text-align: center;">WM0</td> <td style="text-align: center;">WS0</td> <td style="text-align: center;">T0</td> <td style="text-align: center;">C0</td> <td style="text-align: center;">R0</td> <td style="text-align: center;">R3904</td> <td style="text-align: center;">R3968</td> <td style="text-align: center;">R5000</td> <td style="text-align: center;">D0</td> <td style="text-align: center;">V</td> </tr> <tr> <td style="text-align: center;">X240</td> <td style="text-align: center;">Y240</td> <td style="text-align: center;">WY240</td> <td style="text-align: center;">WM1896</td> <td style="text-align: center;">WS984</td> <td style="text-align: center;">T255</td> <td style="text-align: center;">C255</td> <td style="text-align: center;">R3839</td> <td style="text-align: center;">R3967</td> <td style="text-align: center;">R4167</td> <td style="text-align: center;">R8071</td> <td style="text-align: center;">D3071</td> <td style="text-align: center;">Z</td> </tr> </thead> <tbody> <tr> <td style="text-align: center;">IN</td> <td style="text-align: center;">○</td> <td></td> </tr> <tr> <td style="text-align: center;">OT</td> <td></td> <td style="text-align: center;">○</td> <td></td> </tr> <tr> <td style="text-align: center;">D</td> <td></td> <td></td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td style="text-align: center;">○*</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○</td> </tr> </tbody> </table>	Range	X	Y	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR	Oper- and	X0	Y0	WY0	WM0	WS0	T0	C0	R0	R3904	R3968	R5000	D0	V	X240	Y240	WY240	WM1896	WS984	T255	C255	R3839	R3967	R4167	R8071	D3071	Z	IN	○													OT		○												D			○	○	○	○	○	○	○	○*	○*	○	○	
Range	X	Y	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR																																																																							
Oper- and	X0	Y0	WY0	WM0	WS0	T0	C0	R0	R3904	R3968	R5000	D0	V																																																																							
X240	Y240	WY240	WM1896	WS984	T255	C255	R3839	R3967	R4167	R8071	D3071	Z																																																																								
IN	○																																																																																			
OT		○																																																																																		
D			○	○	○	○	○	○	○	○*	○*	○	○																																																																							
<p>● When input control "EN" = 1, this instruction will readout one digit data from the 4 input points starting from IN (IN0~IN3). It takes 4 scans to read out a group of 4-digit BCD values (0000~9999) and store them into D register. With a 32-bit operand, each scan can get 2 digits of data by reading the additional digit from IN4~IN7 and store it in the D+1 register. Each bit of OT0~OT3 will sequentially set to 1 and get the digit data respectively into <math>10^0</math>(ones), <math>10^1</math>(tens), <math>10^2</math>(hundreds), and <math>10^3</math>(thousands). As long as EN is 1, PLC will scan and read out in continuous cycles. When each complete cycle is finished (i.e. the 4 digit readout of <math>10^0</math>~<math>10^3</math> is completed), the readout completed flag "DN" is set to 1. However, it is only kept for one scan. If any digital readout value is not within the range of 0~9 (BCD), then reading error "ERR" will be set to 1 and the value of that group of digits will be set to 0000.</p> <p>● This instruction can only be used once in a program and its output points must be transistor outputs.</p>	<p>X10 — EN—</p> <p>78.DSW</p> <p>IN : X 0 — DN —</p> <p>OT: Y 0 —</p> <p>D : R 0 — ERR —</p>	<ul style="list-style-type: none"> <li>In this example, when X10 is 1, then the numeric value of the thumb wheel switch (5678 in this example) will be read out and stored into the R0 register.</li> <li>The bits (8,4,2,1) with same digit should be connect together and series with a diode (as shown in diagram below).</li> <li>With 32-bit operand a set of similar thumb wheel switch may be added to X4~X7 (Y0~Y3 are shared with another group).</li> </ul>																																																																																		
<p>10<sup>3</sup> (5)      10<sup>2</sup> (6)      10<sup>1</sup> (7)      10<sup>0</sup> (8)</p> <p>BCD thumb wheel switch</p> <p>C X0 X1 X2 X3 X4 X5 X6 X7</p> <p>first group input      second group input (only effective in 32-bit operand)</p> <p>FBE-PLC</p>																																																																																				

FUN 79 <b>D</b> 7SGDL	7-SEGMENT OUTPUT WITH LATCH	FUN 79 <b>D</b> 7SGDL
--------------------------	-----------------------------	--------------------------

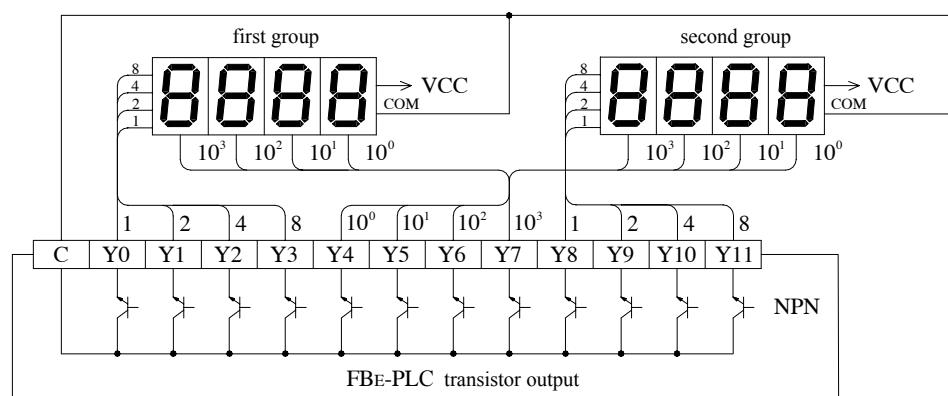


Range \ Operand	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
Y0	Y240	WX240	WY240	WM1896	WS984	T255	C0	R0	R3840	R3904	R3968	R5000	D0	16-bit number	V Z
OT	○								R3903	R3967	R4167	R8071	D3071		
N															0~3

- When input control "EN" = 1, the 4 nibbles of the S register, from digit 0 to digit 3, are sequentially sent out to the 4 output points, OT0~OT3. While output the digit data, the latch signal of that digit (OT4 corresponds to digit 0, OT5 corresponds to digit 1, etc...) at the same time is also sent out so that the digital value will be loaded and latched into the 7-segment display respectively.
- When in D (32-bit) instruction, nibbles 0~3 from the S register, and nibbles 0~3 from the S+1 register are transferred separately to OT0~OT3 and OT8~OT11. Because they are transferred at the same time, they can use the same latch signal. 16-bit instructions do not use OT8~OT11.
- As long as "EN" remains 1, PLC will execute the transfer cyclically. After each transfer of a complete group of numerical values (nibbles 0~3 or 0~7), the output completed flag "DN" will set to 1. However, it will only be kept for 1 scan.



- In this example, when X0=1, the 4 nibbles of R0 will be transferred to the first group 7-segment display in the diagram below. The 4 nibbles of R1 will be transferred to the second group 7-segment display.



## I/O instructions

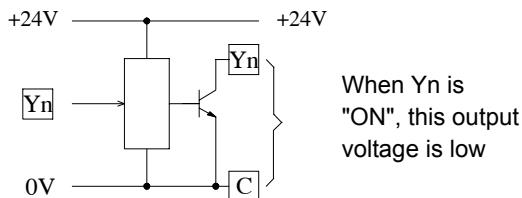
FUN 79 **D**  
7SGDL

### 7-SEGMENT OUTPUT WITH LATCH

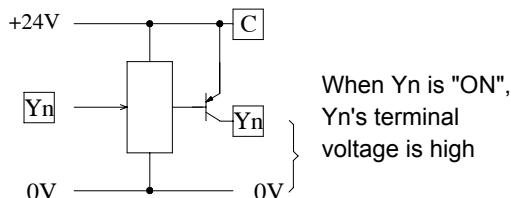
FUN 79 **D**  
7SGDL

- FACON PLC's transistor output has both a negative logic transistor output (NPN transistor - when the output status is ON, the terminal voltage of the transistor output is low), and a positive logic transistor output (PNP - when the output status is ON, the terminal voltage of the transistor output is high). Their structure is as follows:

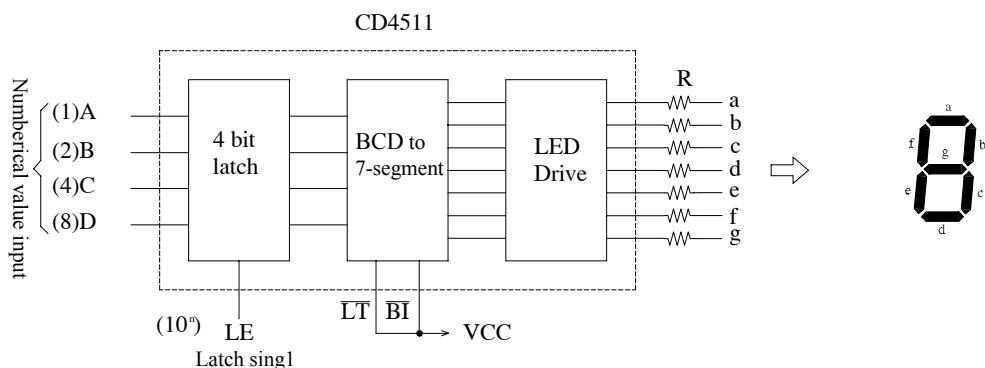
FBE-PLC negative logic output (NPN transistor)



FBE-PLC positive logic output (PNP transistor)



- The data inputs (8,4,2,1) and latch signals of the 7-segment displays on the shelf for positive and negative logic are all available. For example, for numerical value "8", the positive logic input should be 1000, and the negative logic input 0111. Similarly, when the latch signal is 0, the positive logic latch permits the display numerical values to enter through the latch (i.e. be loaded). When the latch signal is 1, the numerical values in the latch are latched (maintained), and with negative logic they are not. The following diagram of a CD-4511 7-segment display IC is an example of a positive logic numerical value input with latch.



- Because the PLC output and the 7-segment display input polarity can be positive and negative logic. Therefore, the polarities between output and input must be coordinated to get the correct result. This instruction uses N to specify the polarity relation between the PLC transistor output, and the 7-segment display. The table below shows all the possibility.

Numerical value input (8~1)	Latch signal ( $10^0$ - $10^3$ )	Value of N
Same	Same	0
	Different	1
Different	Same	2
	Different	3

- In the diagram above, CD4511 is used as an example. If use NPN output, the data input polarity is different to PLC, and its latch input polarity is the same as PLC, so N value should chosen as 2.

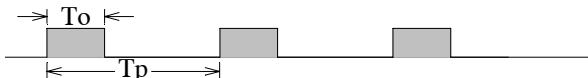
FUN 80 MUXI	MULTIPLEX INPUT	FUN 80 MUXI																																																																																										
	<p>Execution control—EN</p>	<p>IN : Multiplex input point number OT: Multiplex output point number (must be transistor output point) N : Multiplex input lines (2~8) D : Register for storing results D may combine with V, Z to serve indirect address application</p>																																																																																										
	<table border="1"> <thead> <tr> <th>Range</th> <th>X</th> <th>Y</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>Operand</td> <td>X0   X240</td> <td>Y0   Y240</td> <td>WY0   WY240</td> <td>WM0   WM1896</td> <td>WS0   WS984</td> <td>T0   T255</td> <td>C0   C255</td> <td>R0   R3839</td> <td>R3904   R3967</td> <td>R3968   R4167</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td>2   8</td> <td>V   Z</td> </tr> <tr> <td>IN</td> <td>○</td> <td></td> </tr> <tr> <td>OT</td> <td></td> <td>○</td> <td></td> </tr> <tr> <td>N</td> <td></td> <td>○</td> <td></td> </tr> <tr> <td>D</td> <td></td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> <td>○</td> <td></td> </tr> </tbody> </table>	Range	X	Y	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	XR	Operand	X0   X240	Y0   Y240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   8	V   Z	IN	○														OT		○													N													○		D			○	○	○	○	○	○	○*	○*	○		○		
Range	X	Y	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	XR																																																																														
Operand	X0   X240	Y0   Y240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   8	V   Z																																																																														
IN	○																																																																																											
OT		○																																																																																										
N													○																																																																															
D			○	○	○	○	○	○	○*	○*	○		○																																																																															
	<ul style="list-style-type: none"> <li>This instruction uses the multiplex method to read out N lines of input status from 8 consecutive input points (IN0~IN7) starting from the input point specified by IN. With this method we can obtain 8×N input status, but only need to use 8 input points and N output points.</li> <li>The multiplex scanning method goes through N output points starting from the OT output point. Each scan one of the N bits will set to 1 and the corresponding line will be selected. OT0 responsible for first line, while OT1 responsible for second line, etc. Until it read all the N lines the 8×N status that has been read out is then stored into the register starting at D, and the execution completed flag "DN" is set as 1 (but is only kept for one scanning period).</li> <li>With every scan, this instruction retrieves a line for 8 input status, so N lines require N scan cycles before they can be completed.</li> </ul>																																																																																											

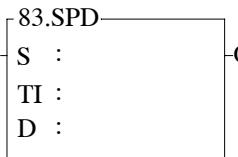
## I/O instructions

FUN 81 <b>D</b> PLSO	PULSE OUTPUT	FUN 81 <b>D</b> PLSO																																																																																																														
<pre> graph LR     EN[Output control—EN] --&gt; MD[MD]     MD --&gt; OUT[OUT—Output go]     Fr[Fr] --&gt; OUT     PC[PC] --&gt; OUT     PAU[Pause control—PAU] --&gt; PC     PC --&gt; DN[DN—Output completed]     UY[UY: or CK] --&gt; DN     UD[Up/Down direction—U/D or DIR] --&gt; DR[DR: or DR]     DR --&gt; HO[HO]     HO --&gt; OUT     HO --&gt; DN     HO --&gt; ERR[ERR—Error]   </pre>	<p style="margin-left: 100px;">81D.PLSO</p> <p>MD : Output mode selection    Fr : Pulse frequency    PC : Output pulse count    UY : Up pulse output point (MD=0).    DY : Down pulse output point (MD=0).    HO : Cumulative output pulse register.    (Can be not assigned).    CK : Pulse output point (MD=1).    DR : Up/Down output point (MD=1).    DIR: 1- up; 0- down.</p>																																																																																																															
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th> <th style="text-align: center; padding: 2px;">Y</th> <th style="text-align: center; padding: 2px;">WX</th> <th style="text-align: center; padding: 2px;">WY</th> <th style="text-align: center; padding: 2px;">WM</th> <th style="text-align: center; padding: 2px;">WS</th> <th style="text-align: center; padding: 2px;">TMR</th> <th style="text-align: center; padding: 2px;">CTR</th> <th style="text-align: center; padding: 2px;">HR</th> <th style="text-align: center; padding: 2px;">OR</th> <th style="text-align: center; padding: 2px;">SR</th> <th style="text-align: center; padding: 2px;">ROR</th> <th style="text-align: center; padding: 2px;">DR</th> <th style="text-align: center; padding: 2px;">K</th> </tr> <tr> <th style="text-align: left; padding: 2px;">Oper- and</th> <th style="text-align: center; padding: 2px;">Yn of Main Unit</th> <th style="text-align: center; padding: 2px;">WX0   WX240</th> <th style="text-align: center; padding: 2px;">WY0   WY240</th> <th style="text-align: center; padding: 2px;">WM0   WM1896</th> <th style="text-align: center; padding: 2px;">WS0   WS984</th> <th style="text-align: center; padding: 2px;">T0   T255</th> <th style="text-align: center; padding: 2px;">C0   C255</th> <th style="text-align: center; padding: 2px;">R0   R3839</th> <th style="text-align: center; padding: 2px;">R3904   R3967</th> <th style="text-align: center; padding: 2px;">R3968   R4167</th> <th style="text-align: center; padding: 2px;">R5000   R8071</th> <th style="text-align: center; padding: 2px;">D0   D3071</th> <th style="text-align: center; padding: 2px;">16/32-bit +/- number</th> </tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;">MD</td> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;">0~1</td> </tr> <tr> <td style="text-align: left; padding: 2px;">Fr</td> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;">8~2000</td> </tr> <tr> <td style="text-align: left; padding: 2px;">PC</td> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;">○</td> </tr> <tr> <td style="text-align: left; padding: 2px;">UY , CK</td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;"></td> </tr> <tr> <td style="text-align: left; padding: 2px;">DY , DR</td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;"></td> </tr> <tr> <td style="text-align: left; padding: 2px;">HO</td> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;"></td> </tr> </tbody> </table>	Range	Y	WX	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	Oper- and	Yn of Main Unit	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	MD													0~1	Fr		○	○	○	○	○	○	○	○	○	○	○	8~2000	PC		○	○	○	○	○	○	○	○	○	○	○	○	UY , CK	○													DY , DR	○													HO			○	○	○	○	○	○	○	○*	○*	○	
Range	Y	WX	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K																																																																																																			
Oper- and	Yn of Main Unit	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number																																																																																																			
MD													0~1																																																																																																			
Fr		○	○	○	○	○	○	○	○	○	○	○	8~2000																																																																																																			
PC		○	○	○	○	○	○	○	○	○	○	○	○																																																																																																			
UY , CK	○																																																																																																															
DY , DR	○																																																																																																															
HO			○	○	○	○	○	○	○	○*	○*	○																																																																																																				
<ul style="list-style-type: none"> <li>● When MD=0, this instruction performs the pulse output control as following:</li> <li>● Whenever the output control “EN” changes from 0→1, it first performs the reset action, which is to clear the output flag “OUT” and “DN” as well as the pulse out register HO to be 0. It gets the pulse frequency and output pulse count values, and reads status of up and down direction “U/D”, so as to determine the direction to be upward or downward. As the reset finished, this instruction will check the input status of pause output “PAU”. No action will be taken if the pause output is 1 (output pause). If the PAU is 0, it will start to output the ON/OFF pulse with 50% duty at the frequency Fr to the UY(U/D=1) or DY(U/D=0) point. It will increment the value of HO register each time when a pulse is output, and will stop the output when HO register’s pulse count is equal to or greater than the cumulative pulse count of PC register and set the output complete flag “DN” to 1. During the time when output pulse is transmitting the output transmitting flag “OUT” will be set to 1, otherwise it will be 0.</li> <li>● Once it starts to transmit pulse, the output control “EN” should kept to 1. If it is changed to 0, it will stop the pulse sending (output point become OFF) and the flag “OUT” changes back to 0, but the other status or data will keep unchanged. However, when its “EN” changes again from 0 to 1, it will lead to a reset action and treat as a new start; the entire procedure will be restarted again.</li> <li>● If you want to pause the pulse output and not to restart the entire procedure, the ‘pause output’ “PAU” input can be used to pause it. When “PAU” =1, this instruction will pause the pulse transmitting (output point is OFF, flag “OUT” change back to 0 and the other status or data keeps unchanged). As it waits until the “PAU” changes back from 1 to 0, this instruction will return to the status before it is paused and continues the pulse transmitting output.</li> <li>● During the pulse transmission, this instruction will keep monitoring the value of pulse frequency Fr and output pulse count PC. Therefore, as long as the pulse output is not finished, it may allow the changing of the pulse frequency and pulse count. However, the up/down direction “U/D” status will be got only once when it takes the reset action (“EN” changes from 0→1), and will keep the status until the pulse output completed or another reset occur. That is to say, except that at the very moment of reset, the change of “U/D” does not influence the operation of this instruction.</li> <li>● The main purpose of this instruction is to drive the stepping motor with the UY (upward) and DY (downward) two directional pulses control, so as to help you control the forward or reverse rotating of stepping motor. Nevertheless, if you need only single direction revolving, you can assign just one of the UY or DY (which will save one output point), and leaving the other output blank. In such case, the instruction will ignore the up/down input status of “U/D”, and the output pulse will send to the output point you assigned.</li> </ul>																																																																																																																

FUN 81 <b>D</b> PLSO	PULSE OUTPUT	FUN 81 <b>D</b> PLSO
<ul style="list-style-type: none"> <li>When MD=1, the pulse output will reflect on the control output DIR (pulse direction. DIR=1, up; DIR=0, down) and CK (pulse output).</li> <li>This instruction can only be used once, and UY (CK) and DY (DR) must be transistor output point on the PLC main unit.</li> <li>The effective range of output pulse count PC for 16 bit operand is 0~32767. For the 32 bit operand(<b>D</b> instruction), it is 0~2147483647. If the PC value = 0, it is treated as infinite pulse count, and this instruction will transmit pulses without end with HO value and “DN” flag set at 0 all the time. The effective range of pulse frequency (Fr) is 8~2000. If the value PC or Fr exceeds the range, this instruction will not be carried out and the error flag “ERR” will set to 1.</li> </ul>		
	<ul style="list-style-type: none"> <li>In this example, the program controls the stepping motor to drive forward for 80 pulses (steps) at the speed of 100Hz first, and then makes it turn reverse for 40 pulses the speed of 50Hz. Make sure that the up/down direction, frequency Fr and the pulse count PC must be set before the reset take action (“EN” changes from 0→1).</li> </ul>	

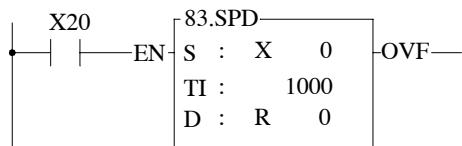
## I/O instructions

FUN 82 PWM	PULSE WIDTH MODULATION	FUN 82 PWM																																																																							
Execution control — EN <div style="border: 1px solid black; padding: 5px; display: inline-block;"> <b>82.PWM</b>            To : <input type="text"/>            Tp : <input type="text"/>            OT: <input type="text"/> </div>	To : Pulse ON width (0~32767mS) Tp : Pulse period (1~32676mS) OT: Pulse output point																																																																								
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th><th style="padding: 2px;">Y</th><th style="padding: 2px;">WX</th><th style="padding: 2px;">WY</th><th style="padding: 2px;">WM</th><th style="padding: 2px;">WS</th><th style="padding: 2px;">TMR</th><th style="padding: 2px;">CTR</th><th style="padding: 2px;">HR</th><th style="padding: 2px;">IR</th><th style="padding: 2px;">OR</th><th style="padding: 2px;">SR</th><th style="padding: 2px;">ROR</th><th style="padding: 2px;">DR</th><th style="padding: 2px;">K</th></tr> <tr> <th style="text-align: left; padding: 2px;">Oper- rand</th><th style="padding: 2px;">Yn of main unit</th><th style="padding: 2px;">WX0   WX240</th><th style="padding: 2px;">WY0   WY240</th><th style="padding: 2px;">WM0   WM1896</th><th style="padding: 2px;">WS0   WS984</th><th style="padding: 2px;">T0   T255</th><th style="padding: 2px;">C0   C255</th><th style="padding: 2px;">R0   R3839</th><th style="padding: 2px;">R3840   R3903</th><th style="padding: 2px;">R3904   R3967</th><th style="padding: 2px;">R3968   R4167</th><th style="padding: 2px;">R5000   R8071</th><th style="padding: 2px;">D0   D3071</th><th style="padding: 2px;">0   32767</th></tr> </thead> <tbody> <tr> <td style="padding: 2px;">To</td><td style="padding: 2px;"><input type="radio"/></td><td style="padding: 2px;"><input type="radio"/></td></tr> <tr> <td style="padding: 2px;">Tp</td><td style="padding: 2px;"><input type="radio"/></td><td style="padding: 2px;"><input type="radio"/></td></tr> <tr> <td style="padding: 2px;">OT</td><td style="padding: 2px;"><input checked="" type="radio"/></td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> </tbody> </table>	Range	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	Oper- rand	Yn of main unit	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	0   32767	To	<input type="radio"/>	Tp	<input type="radio"/>	OT	<input checked="" type="radio"/>																																					
Range	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K																																																											
Oper- rand	Yn of main unit	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	0   32767																																																											
To	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																																																												
Tp	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																																																												
OT	<input checked="" type="radio"/>																																																																								
<ul style="list-style-type: none"> <li>● When execution control "EN" = 1, will send the pulse to output point OT with the "ON" state for To ms and period as Tp. OT must be a transistor output point on the main unit. When "EN" is 0, the output point will be OFF.</li> </ul>  <ul style="list-style-type: none"> <li>● The units for Tp and To are mS, resolution is 1 mS. The minimum value for To is 0 (under such case the output point OT will always be OFF), and its maximum value is the same as Tp (under such case the output point OT will always be on). If To &gt; Tp there will be an error, this instruction will not be carried out, and the error flag "ERR" will set to 1.</li> <li>● This instruction can only be used once.</li> </ul>																																																																									

FUN 83 SPD	SPEED DETECTION	FUN 83 SPD																																																																																								
Detection control—EN 	S : Pulse input point for speed detection TI: Sampling duration (units in mS) D : Register storing results																																																																																									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th> <th style="text-align: center; padding: 2px;">X</th> <th style="text-align: center; padding: 2px;">WX</th> <th style="text-align: center; padding: 2px;">WY</th> <th style="text-align: center; padding: 2px;">WM</th> <th style="text-align: center; padding: 2px;">WS</th> <th style="text-align: center; padding: 2px;">TMR</th> <th style="text-align: center; padding: 2px;">CTR</th> <th style="text-align: center; padding: 2px;">HR</th> <th style="text-align: center; padding: 2px;">IR</th> <th style="text-align: center; padding: 2px;">OR</th> <th style="text-align: center; padding: 2px;">SR</th> <th style="text-align: center; padding: 2px;">ROR</th> <th style="text-align: center; padding: 2px;">DR</th> <th style="text-align: center; padding: 2px;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;">Operand</td> <td style="text-align: center; padding: 2px;">X0</td> <td style="text-align: center; padding: 2px;">WX0</td> <td style="text-align: center; padding: 2px;">WY0</td> <td style="text-align: center; padding: 2px;">WM0</td> <td style="text-align: center; padding: 2px;">WS0</td> <td style="text-align: center; padding: 2px;">T0</td> <td style="text-align: center; padding: 2px;">C0</td> <td style="text-align: center; padding: 2px;">R0</td> <td style="text-align: center; padding: 2px;">R3840</td> <td style="text-align: center; padding: 2px;">R3904</td> <td style="text-align: center; padding: 2px;">R3968</td> <td style="text-align: center; padding: 2px;">R5000</td> <td style="text-align: center; padding: 2px;">D0</td> <td style="text-align: center; padding: 2px;">1</td> </tr> <tr> <td style="text-align: left; padding: 2px;"></td> <td style="text-align: center; padding: 2px;">X7</td> <td style="text-align: center; padding: 2px;">WX240</td> <td style="text-align: center; padding: 2px;">WY240</td> <td style="text-align: center; padding: 2px;">WM1896</td> <td style="text-align: center; padding: 2px;">WS984</td> <td style="text-align: center; padding: 2px;">T255</td> <td style="text-align: center; padding: 2px;">C255</td> <td style="text-align: center; padding: 2px;">R3839</td> <td style="text-align: center; padding: 2px;">R3903</td> <td style="text-align: center; padding: 2px;">R3967</td> <td style="text-align: center; padding: 2px;">R4167</td> <td style="text-align: center; padding: 2px;">R8071</td> <td style="text-align: center; padding: 2px;">D3071</td> <td style="text-align: center; padding: 2px;">32767</td> </tr> <tr> <td style="text-align: left; padding: 2px;">S</td> <td style="text-align: center; padding: 2px;"><input checked="" type="radio"/></td> <td></td> </tr> <tr> <td style="text-align: left; padding: 2px;">TI</td> <td></td> <td style="text-align: center; padding: 2px;"><input checked="" type="radio"/></td> </tr> <tr> <td style="text-align: left; padding: 2px;">D</td> <td></td> <td></td> <td style="text-align: center; padding: 2px;"><input checked="" type="radio"/></td> </tr> </tbody> </table>	Range	X	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	Operand	X0	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	1		X7	WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D3071	32767	S	<input checked="" type="radio"/>														TI		<input checked="" type="radio"/>	D			<input checked="" type="radio"/>																						
Range	X	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K																																																																												
Operand	X0	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	1																																																																												
	X7	WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D3071	32767																																																																												
S	<input checked="" type="radio"/>																																																																																									
TI		<input checked="" type="radio"/>																																																																																								
D			<input checked="" type="radio"/>																																																																																							

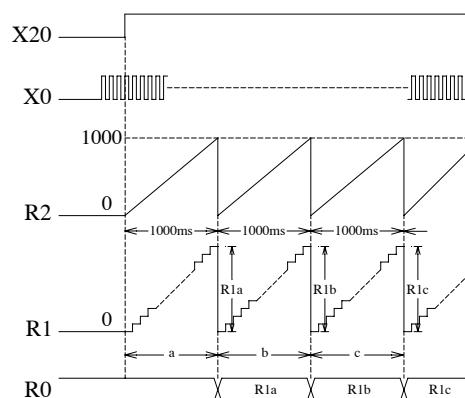
- This instruction uses the interrupt feature of the 8 high speed input points (X0~X7) on the PLC main unit to detect the frequency of the input signal. Within a specific sampling time (TI), it will calculate the input pulse count for S input point, and indirectly find the revolution speed of rotating devices (such as motors).
- While use this instruction to detect the rotating speed of devices, The application should design to generate more pulse per revolution in order to get better result, but the sum of input frequency of all detected signals should under 5KHz, otherwise the WDT may occur.
- The D register for storing results uses 3 successive 16-bit registers starting from D (D0~D2). Besides D0 which is used to store counting results, D1 and D2 are used to store current counting values and sampling duration.
- When detection control "EN" = 1, it starts to calculate the pulse count for the S input point, which can be shown in D1 register. Meanwhile the sampling timer (D2) is switched on and keeps counting until the value of D2 is reach to the sampling period (TI). The final counted value is stored into the D0 register, and then a new counting cycle is started again. The sampling counting will go on repeating until "EN" = 0.
- Because D0 only has 16 bits, so the maximum count is 32767. If the sampling period is too long or the input pulse is too fast then the counted value may exceed 32767, under that case the overflow flag will set to 1, and the counting action will stop.
- Because the sampling period TI is already known and if every revolution of attached rotating device produces "n" pulses, then the following equation can be used to get the revolution

$$\text{speed : } N = \frac{(D0) \times 60}{n \times TI} \times 10^3 \quad (\text{rpm})$$



- In the above example, if every revolution of the rotating device produces 20 pulses (n = 20), and the R0 value is 200, then the revolution per minute speed "N" is as

$$\text{follows: } N = \frac{(200) \times 60}{60 \times 1000} \times 10^3 = 200 \quad \text{rpm}$$



## I/O instructions

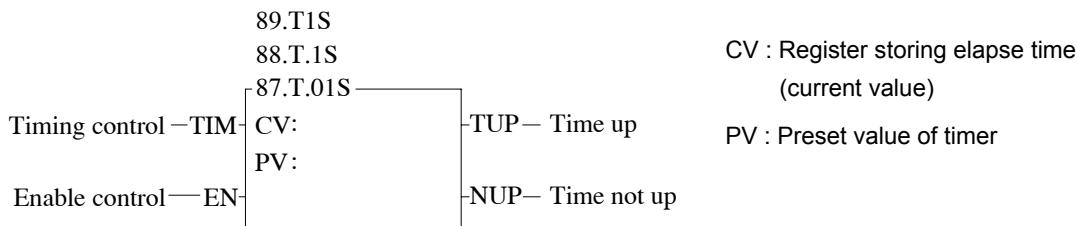
FUN 84 7SGMO	The handy instruction of FB-7SG module														FUN 84 7SGMO																																																																																																																																															
84.7SGMO																																																																																																																																																														
Execution control—EN															S : Data register to be displayed.																																																																																																																																															
Decode selection—N/D															Yn : Output point preserved for controlled display module.																																																																																																																																															
Preceded zero selection—L/N															Dn : Total digit (characters) to be displayed.																																																																																																																																															
															PT : Decimal point flashing designation (Invalid for non-decoding display).																																																																																																																																															
															IT : Brightness.																																																																																																																																															
															WS: Working register for this instruction instance.																																																																																																																																															
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Range</th> <th>Y</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th></th> </tr> </thead> <tbody> <tr> <td rowspan="2">Oper- and</td> <td>Y0</td> <td>WX0</td> <td>WY0</td> <td>WM0</td> <td>WS0</td> <td>T0</td> <td>C0</td> <td>R0</td> <td>R3840</td> <td>R3904</td> <td>R3968</td> <td>R5000</td> <td>D0</td> <td></td> <td>16-bit + number</td> </tr> <tr> <td>Y240</td> <td>WX240</td> <td>WY240</td> <td>WM1896</td> <td>WS984</td> <td>T255</td> <td>C255</td> <td>R3839</td> <td>R3903</td> <td>R3967</td> <td>R4167</td> <td>R8071</td> <td>D3071</td> <td></td> <td></td> </tr> <tr> <td>S</td> <td>○</td> <td></td> </tr> <tr> <td>Yn</td> <td>○</td> <td></td> </tr> <tr> <td>Dn</td> <td>○</td> <td>1-8</td> </tr> <tr> <td>PT</td> <td>○</td> <td>0-FFH</td> </tr> <tr> <td>IT</td> <td>○</td> <td>1-16</td> </tr> <tr> <td>WS</td> <td></td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> </tbody> </table>	Range	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K		Oper- and	Y0	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0		16-bit + number	Y240	WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D3071			S	○	○	○	○	○	○	○	○	○	○	○	○	○	○		Yn	○															Dn	○	○	○	○	○	○	○	○	○	○	○	○	○	○	1-8	PT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	0-FFH	IT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	1-16	WS			○	○	○	○	○	○	○	○	○*	○*	○																	
Range	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K																																																																																																																																																
Oper- and	Y0	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0		16-bit + number																																																																																																																																															
	Y240	WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D3071																																																																																																																																																	
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																
Yn	○																																																																																																																																																													
Dn	○	○	○	○	○	○	○	○	○	○	○	○	○	○	1-8																																																																																																																																															
PT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	0-FFH																																																																																																																																															
IT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	1-16																																																																																																																																															
WS			○	○	○	○	○	○	○	○	○*	○*	○																																																																																																																																																	
<ul style="list-style-type: none"> <li>This instruction is dedicated for 7-segment display module (FB-7SG). It use the table driven method to assign the display data address, number of displaying characters, brightness, position of decimal point, decode or non decode display, as well as if the leading zero displayed or not. It can greatly reduce the programming time and make the program simplified.</li> <li>For the detailed explanation and example, please refer to chapter 17 “FB-7SG 7 segment LED display module”.</li> </ul>																																																																																																																																																														

FUN 85 TPSNS	The convenient instruction for temperature measurement module (Brief function description)	FUN 85 TPSNS																																																						
<b>Execution control</b> EN <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> 85.TPSNS  Tp : -ERR—Parameter error  PI :  Zn : -ALM—Temprature sensor  Yn : line broking  SR :  WR : </div>	Tp : Type of temperature sensor, it can be J or K Type thermocouple. PI : Polarity and the voltage range setting for temperature module. Zn : Total temperature points selection. Yn : Starting output point preserve for controlled temperature module. SR : Starting register for temperature measuring value storing. WR: Starting working register for the instance of this instruction.																																																							
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Oper- and</th> <th style="text-align: left; padding: 2px;">Range</th> <th style="text-align: left; padding: 2px;">Y</th> <th style="text-align: left; padding: 2px;">HR</th> <th style="text-align: left; padding: 2px;">ROR</th> <th style="text-align: left; padding: 2px;">DR</th> <th style="text-align: left; padding: 2px;">K</th> </tr> </thead> <tbody> <tr> <td></td><td style="padding: 2px;">Y0   Y255</td><td style="padding: 2px;">Y0   R3839</td><td style="padding: 2px;">R0   R8071</td><td style="padding: 2px;">R5000   D3071</td><td style="padding: 2px;">D0   D3071</td><td style="padding: 2px;"></td></tr> <tr> <td style="text-align: center;">Tp</td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;">0~1</td><td style="text-align: center;"></td></tr> <tr> <td style="text-align: center;">PI</td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;">0~3</td><td style="text-align: center;"></td></tr> <tr> <td style="text-align: center;">Zn</td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;">6 , 12 , 18 , 24</td><td style="text-align: center;"></td></tr> <tr> <td style="text-align: center;">Yn</td><td style="text-align: center;"><input type="radio"/></td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;"></td><td style="text-align: center;"></td></tr> <tr> <td style="text-align: center;">SR</td><td style="text-align: center;"></td><td style="text-align: center;"><input type="radio"/></td><td style="text-align: center;"><input type="radio"/>*</td><td style="text-align: center;"><input type="radio"/></td><td style="text-align: center;"></td><td style="text-align: center;"></td></tr> <tr> <td style="text-align: center;">WR</td><td style="text-align: center;"></td><td style="text-align: center;"><input type="radio"/></td><td style="text-align: center;"><input type="radio"/>*</td><td style="text-align: center;"><input type="radio"/></td><td style="text-align: center;"></td><td style="text-align: center;"></td></tr> </tbody> </table>	Oper- and	Range	Y	HR	ROR	DR	K		Y0   Y255	Y0   R3839	R0   R8071	R5000   D3071	D0   D3071		Tp					0~1		PI					0~3		Zn					6 , 12 , 18 , 24		Yn	<input type="radio"/>						SR		<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>			WR		<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>		
Oper- and	Range	Y	HR	ROR	DR	K																																																		
	Y0   Y255	Y0   R3839	R0   R8071	R5000   D3071	D0   D3071																																																			
Tp					0~1																																																			
PI					0~3																																																			
Zn					6 , 12 , 18 , 24																																																			
Yn	<input type="radio"/>																																																							
SR		<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>																																																				
WR		<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>																																																				
<p><b>Brief function description</b></p> <ul style="list-style-type: none"> <li>● This instruction is dedicated for FB-4AJ(K)xx multiplexing temperature measuring module. With this instruction, the user can easily acquire multi points of temperature measuring values to provide monitoring or serve as the Process Variable (PV) for PID temperature control.</li> <li>● This instruction must incorporate with FB-4AJ(K)xx multiplexing temperature measuring module in its usage. Hereby it introduced briefly about the function of this instruction only. For details of the function, explanation, usages and examples, please refer to Chapter 20 “Temperature measuring of FB-PLC and PID Control”.</li> </ul>																																																								

## Temperature control instructions 2

FUN 86 TPCTL	Temperature measurement and control of temperature module (Brief description of its functions)	FUN 86 TPCTL																				
<p>Execution control—EN</p> <p>Heating/Cooling—H/C</p>	<p>86.TPCTL</p> <table border="1" style="margin-left: 20px;"> <tr> <td>Yn :</td> <td>ERR—Parameter error</td> </tr> <tr> <td>Sn :</td> <td>ALM—Temperature control warning</td> </tr> <tr> <td>Zn :</td> <td></td> </tr> <tr> <td>Sv :</td> <td></td> </tr> <tr> <td>Os :</td> <td></td> </tr> <tr> <td>PR :</td> <td></td> </tr> <tr> <td>IR :</td> <td></td> </tr> <tr> <td>DR :</td> <td></td> </tr> <tr> <td>OR :</td> <td></td> </tr> <tr> <td>WR :</td> <td></td> </tr> </table>	Yn :	ERR—Parameter error	Sn :	ALM—Temperature control warning	Zn :		Sv :		Os :		PR :		IR :		DR :		OR :		WR :		<p>Yn : Starting number for PWM temperature control output.</p> <p>Sn : The assigning of PID temperature control to be performed starting from which point.</p> <p>Zn : The number of PID temperature control points controlled by this instruction.</p> <p>Sv : Starting register number for temperature setting value storing.</p> <p>Os : Starting register number for temperature deviation value storing.</p> <p>PR : Starting register number for gain setting value storing.</p> <p>IR : Starting register number for integral time constant setting value storing.</p> <p>DR : Starting register number for differential time constant setting value storing.</p> <p>OR : Starting register number for temperature control value output storing.</p> <p>WR : Starting working register number for this instruction.</p>
Yn :	ERR—Parameter error																					
Sn :	ALM—Temperature control warning																					
Zn :																						
Sv :																						
Os :																						
PR :																						
IR :																						
DR :																						
OR :																						
WR :																						
<p><b>Brief function description</b></p> <ul style="list-style-type: none"> <li>This instruction treats the temperature value which measured by FB-4AJ(K)xx multiplexing temperature measuring module under the FUN85 (TPSNS) instruction as Process Variable (PV), and gets the user defined temperature Set Point (SP) together to be processed by software PID arithmetic operation to reach a proper output control value, so as to control the temperature to fall within the range which user expected.</li> <li>This instruction must incorporate with FB-4AJ(K)xx multiplexing temperature measuring module and convenient instruction of FUN85 for its usage. Hereby it introduced briefly about the function of this instruction only. For details of the instruction function, explanation, usages and examples, please refer to descriptions of Chapter 20 “Temperature measuring of FB-PLC and PID Control”.</li> </ul>																						

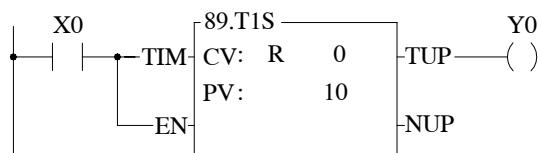
FUN87 T.01S FUN88 T.1S FUN89 T1S	CUMULATIVE TIMER	FUN87 T.01S FUN88 T.1S FUN89 T1S
--	------------------	--



Oper- and	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
		WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C199	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	0   32767
		CV												
PV		○	○	○	○	○	○	○	○	○*	○*	○	○	○

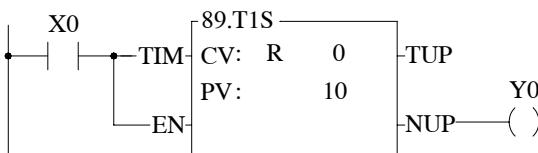
- The operation for this instruction is the same as that for the basic timer (T0~T255), except that the basic timer only has a "timing control" input - when its input is 1 it starts timing, and when input is 0 it get clear. Every time the input changes, it starts timing again and is unable to accumulate. Timing with this instruction is only permissible when enable control "EN" = 1. With this instruction, when timing control "TIM" is 1, it is the same as a basic timer, but when "TIM" is 0, it does not clear, but keeps the current value. If the timer need to clear, then change enable control "EN" to 0. When timing control "TIM" is once again to be 1, it will continue to accumulate from the previous value when the timer last paused. In addition, this instruction also has two outputs, "Time up TUP" (when time up it is 1, usually it is 0) and "Time not up" (usually it is 1, when time is up it is 0). Users can utilize input and output combinations to produce timers with various different functions. For example:

- On delay energizing timer:



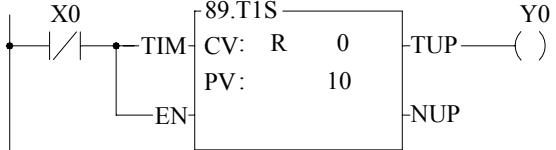
- This timer's output (Y0 in this example) is normally not energized. When this timer's input control (X0 in this example) is activated (ON), only after delay by 10 sec will output Y0 become energized (ON).

- On delay de-energizing timer:



- The output Y0 of this timer is usually energized. When this timer's input control X0 is on, only after delay by 10 sec will the output become de-energized (OFF).

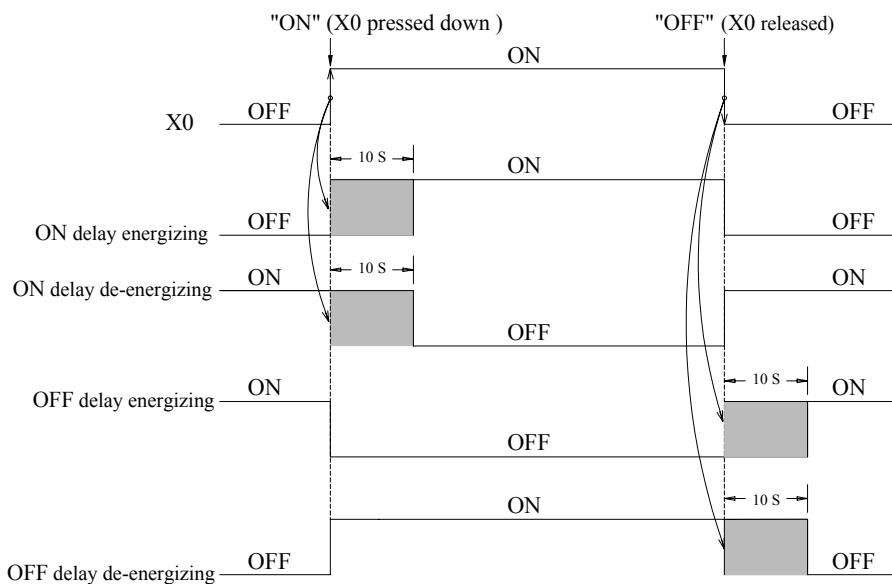
## Cumulative timer instructions

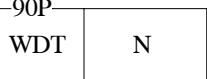
FUN87 T.01S FUN88 T.1S FUN89 T1S	<b>CUMULATIVE TIMER</b>	FUN87 T.01S FUN88 T.1S FUN89 T1S
	<ul style="list-style-type: none"> <li>● Off delay energizing timer:</li> </ul>  <p>The ladder logic diagram shows a timer block (89.T1S) with an enable input (EN) and a normally open output (TUP). The coil of the timer is controlled by input X0. The output Y0 is connected to the normally open contact of the timer block.</p> <ul style="list-style-type: none"> <li>● This timer's output Y0 is usually de-energized. When this timer's input control X0 is off, only after delay by 10 sec will output Y0 become energized (ON).</li> </ul>	<ul style="list-style-type: none"> <li>● This timer's output Y0 is usually de-energized. When this timer's input control X0 is off, only after delay by 10 sec will output Y0 become energized (ON).</li> </ul>

- Off delay de-energizing timer:

- This timer's output Y0 is usually energized. When this timer's timing control X0 is off, only after delay by 10 sec will output Y0 become de-energized (OFF).

- The diagram below shows the relation on input and output for the above 4 kinds of timers.



FUN 90 <b>P</b> WDT	WATCHDOG TIMER	FUN 90 <b>P</b> WDT
Execution control—EN↑		N : The watchdog time. The range of N is 5~120, unit in 10mS (i.e. 50ms~1.2 sec)

- When execution control "EN" = 1 or "EN ↑ " (**P** instruction) transition from 0 to 1, will set the watchdog time to Nx10ms. If the scan time exceeds this preset time, PLC will shut down and not execute the application program.
- The WDT feature is designed mainly as a safety consideration from the system view for the application. For example, if the CPU of PLC is suddenly damaged, and there is no way to execute the program or refresh I/O, then after the WDT time expired, the WDT will automatically switch off all the I/Os, so as to ensure safety. In certain applications, if the scan time is too long, it may cause safety problems or problems of non-conformance with control requirements. This instruction can be used to establish the limitation of the scan time that you require.
- Once the WDT time has been set it will always be kept, and there is no need to set it again on each scan. Therefore, in practice this instruction should use the **P** instruction.
- Default WDT time is 0.25 sec.
- For the operation principles of WDT please refer to the RSWDT(FUN 91) instruction.

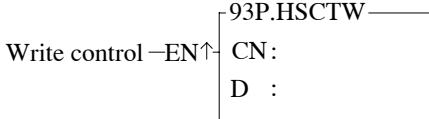
## Watchdog timer instructions

FUN 91 <b>P</b> RSWDT	RESET WATCHDOG TIMER	FUN 91 <b>P</b> RSWDT
Execution control –EN↑		This instruction has no operand.
<ul style="list-style-type: none"><li>When execution control "EN" = 1 or "EN ↑" (<b>P</b> instruction), the WDT timer will be reset (i.e. WDT will start timing again from 0).</li><li>The functions of WDT have already been described in FUN90 (WDT instruction). The operation principles of watch dog timer are as follows:  The watchdog timer is normally implemented by a hardware one-shot timer (it can not be software, otherwise if CPU fail, the timer becomes ineffective, and safeguards are quite impossible). "One-shot" means that after triggered the timer once, the timing value will immediately be reset to 0 and timing will restart. If WDT has begun timing, and never triggered it again, then the WDT timing value will continue accumulating until it reach the preset value of N, at that time WDT will be activated, and PLC will be shut down. If trigger the WDT once every time before the WDT time N has been reached, then WDT will never be activated. PLC can use this feature to ensure the safety of the system. Each time when PLC enters into system housekeeping after finished the program scanning and I/O refresh, it will usually trigger WDT once, so if the system functions normally and scan time does not exceed WDT time then WDT is never activated. However, if CPU is damaged and unable to trigger WDT, or the scan time is too long, then there will not be enough time to trigger WDT within the period N, WDT will be activated and will shut off PLC.</li><li>In some applications, when you set the WDT time (FUN90) to desire, the scan time of your program in certain situations may temporarily exceed the preset time of WDT. This situation can be anticipated and allowed for, and you naturally do not wish PLC to shut down for this reason. You can use this instruction to trigger WDT once and avoid the activation of WDT. This is the main purpose of this instruction.</li></ul>		

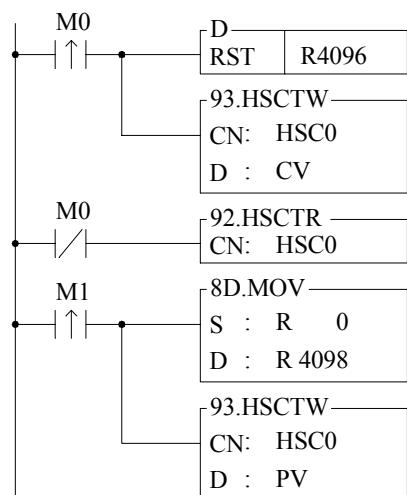
FUN 92 P HSCTR	Hardware High Speed Counter Current Value (CV) Access	FUN 92 P HSCTR																		
CN : Hardware high speed counter number																				
Reatout control—EN↑	 CN:	0: SC0 or HST0 1: SC1 or HST1 2: SC2 or HST2 3: SC3 or HST3 4: STA																		
<ul style="list-style-type: none"> <li>The HSC0~HSC3 counters of FB-PLC are 4 sets of 32bit high speed counter with the variety counting modes such as up/down pulse, pulse-direction, AB-phase. All the 4 high speed counters are built in the ASIC hardware and could perform count, compare, and send interrupt independently without the intervention of the CPU. In contrast to the software high speed counters HSC4~HSC7, which employ interrupt method to request for CPU processing, hence if there are many counting signals or the counting frequency is high, the PLC performance (scanning speed) will be degraded dramatically. Since the current values CV of HSC0~HSC3 are built in the internal hardware circuits of ASIC, the user control program (ladder diagram) cannot retrieve them directly from ASIC. Therefore, it must employ this instruction to get the CV value from hardware HSC and put it into the register which control program can access. The following is the arrangement of CV, PV in ASIC and their corresponding CV, PV registers of PLC for HSC0~HSC3.</li> </ul>																				
<table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; width: 30%;">PLC register</th> <th style="text-align: center; width: 40%;"></th> <th style="text-align: center; width: 30%;">ASIC</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">HSC0</td> <td style="text-align: center;">           CV register  → DR4096            PV register  → DR4098         </td> <td style="text-align: center;">           CV →             PV →  </td> </tr> <tr> <td style="text-align: center;">HSC1</td> <td style="text-align: center;">           CV register  → DR4100            PV register  → DR4102         </td> <td style="text-align: center;">           CV →             PV →  </td> </tr> <tr> <td style="text-align: center;">HSC2</td> <td style="text-align: center;">           CV register  → DR4104            PV register  → DR4106         </td> <td style="text-align: center;">           CV →             PV →  </td> </tr> <tr> <td style="text-align: center;">HSC3</td> <td style="text-align: center;">           CV register  → DR4108            PV register  → DR4110         </td> <td style="text-align: center;">           CV →             PV →  </td> </tr> <tr> <td style="text-align: center;">HSTA</td> <td style="text-align: center;">           CV register  → DR4112            PV register  → R4154         </td> <td style="text-align: center;">           CV →             PV →  </td> </tr> </tbody> </table>			PLC register		ASIC	HSC0	CV register  → DR4096 PV register  → DR4098	CV → PV →	HSC1	CV register  → DR4100 PV register  → DR4102	CV → PV →	HSC2	CV register  → DR4104 PV register  → DR4106	CV → PV →	HSC3	CV register  → DR4108 PV register  → DR4110	CV → PV →	HSTA	CV register  → DR4112 PV register  → R4154	CV → PV →
PLC register		ASIC																		
HSC0	CV register  → DR4096 PV register  → DR4098	CV → PV →																		
HSC1	CV register  → DR4100 PV register  → DR4102	CV → PV →																		
HSC2	CV register  → DR4104 PV register  → DR4106	CV → PV →																		
HSC3	CV register  → DR4108 PV register  → DR4110	CV → PV →																		
HSTA	CV register  → DR4112 PV register  → R4154	CV → PV →																		

- When access control “EN” =1 or “EN ↑ ” (P instruction) changes from 0→1, will gets the CV value of HSC designated by CN from ASIC and puts into the HSC corresponding CV register (i.e. the CV of HSC0 will be read and put into DR4096 or the CV of HSC1 will be read and put into DR4100).
- Although the PV within ASIC has a corresponding PV register in CPU, but it is not necessary to access it (actually it can't be) for that the PV value within ASIC comes from the PV register in CPU.
- HSTA is a timer, which use 0.1ms as its time base. The content of CV represents elapse time counting at 0.1mS tick.
- For detailed applications, please refer to Chapter 11 “The high speed counter and high speed timer of FB-PLC”.

## High speed counting/timing instructions

FUN 93 <b>P</b> HSCTW	Hardware High Speed Counter Current Value and Preset Value(CV) Writing	FUN 93 <b>P</b> HSCTW
	<p style="text-align: right;">CN : Hardware high speed counter to be written            0: HSC0 or HST1            1: HSC1 or HST2            2: HSC2 or HST3            3: HSC3 or HST4            4: HSTA</p> <p style="text-align: right;">D : Write target (0 represents CV, 1 represents PV)</p> 	

- Please refer first to FUN92 for the relation between the CV or PV value of HSC0~HSC3 and HSTA within ASIC and their corresponding CV and PV registers in CPU.
- When write control “EN”=1 or “EN ↑” (**P** instruction) changes from 0→1, it writes the content of CV or PV register of high speed counter designed by CN of CPU, to the corresponding CV or PV of HSC within ASIC.
- It is quite often to set the PV value for most application program, When the count value reaches the preset value, the counter will send out interrupt signal immediately. By way of the interrupt service program, you can implement different kinds of precision counting or positioning control.
- When there is an interrupt of power supply for FB-PLC, the values of current value registers CV of HSC0~HSC3 within ASIC will be read out and wrote into the HSC0~HSC3 CV registers (with power retentive function) of CPU automatically. When power comes up, these CV values will be restored to ASIC. However, if your application demands that when power is on, the values should be cleared to 0 or begin counting from a certain value, then you have to use this instruction to write in the CV value for HSC in ASIC.
- When write a non-zero value into the PV register of HSTA will cause the HSTA1 interrupt subroutine to be executed for every PV×0.1ms.
- For detailed applications, please refer Chapter 11 “The high speed counter and high speed timer of FB-PLC”.



- As the program in the left diagram, when M0 changes from 0 →1, it clears the current value of HSC0 to 0, and writes into ASIC hardware through FUN93.
- When M0 is 0, it reads out the current counting value.
- When M1 changes from 0→1, it moves DR0 to DR4098, and writes into ASIC hardware through FUN93.
- Whenever the current value equals to the DR0, The HSC0I interrupt sub program will be executed.

FUN 94 ASCWR	ASCII WRITE	FUN 94 ASCWR
<p>Output control —EN↑—</p> <p>Pause output —PAU—</p> <p>Abort output —ABT—</p> <pre> graph LR     EN[EN↑] --&gt; MD[MD]     PAU[PAU] --&gt; Pt[Pt]     ABT[ABT] --&gt; DN[DN]     MD --- Pt     Pt --- PAU     Pt --- ABT     ABT --- DN   </pre> <p>MD : Output mode =0, output to communication port1. others, reserved for future usage.</p> <p>S : Starting register of file data.</p> <p>Pt : Starting working register for this instruction instance. It taken up 8 registers and can't be reused in other part of program.</p>		<p>MD: Output mode =0, output to communication port1. others, reserved for future usage.</p> <p>S : Starting register of file data.</p> <p>Pt : Starting working register for this instruction instance. It taken up 8 registers and can't be reused in other part of program.</p>

Range Oper- and	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3967   R4167	R5000   R8071	D0   D3071	0   1
MD													○
S	○	○	○	○	○	○	○	○	○	○	○	○	
Pt		○	○	○	○	○	○		○	○*	○*	○	

- When MD=0 and output control “EN ↑” changes from 0→1, it transmits the ASCII data which starting from S to the communication port 1 (Port1), until reach end of file.
- S file data can be edited with the programming software PROLADDER or WinProladder (please refer to the explanation of chapter 15 “ASCII function application”). If necessary the user can also edit the ASCII file directly by change the value of data registers. However, the edited data must be follow the ASCII file format (the details described in chapter 15), otherwise, this instruction will halt the transmission and set the error flag “ERR” to 1. If the entire file is correctly and successfully transmitted, then the output is completed and “DN” is set to 1.
- The control input of this instruction is of positive edge triggered. Once “EN ↑” changes from 0→1 then this instruction starts the execution, until finished the transmission of the entire file then the execution is completed. During the transmission, the action flag “ACT” will be kept at 1 all the time. Only when output pause, error, or abort occurs, will it change back to 0.
- This instruction can be repeatedly used, but only one will be executed (transmit data) at any certain time. It is the obligation of user to make sure the right execution sequence.
- While this instruction is in execution, if the pause “PAU” is 1, this instruction will pause the transmission of file data. It will resume transmission when the pause “PAU” backs to 0.
- While this instruction is in execution, if the abort “ABT” is 1, this instruction will abandon the transmission of file data, and then it is able to take next instruction for execution.
- Whenever using the FUN94 (ASCWR) instruction, it must first set the DIP switches on the CPU main unit to SW-1 OFF & SW-2 ON position.
- For detail applications, please refer to chapter 15 “The Application of ASCII function”.

## Report printing instructions

FUN 94 ASCWR	ASCII WRITE	FUN 94 ASCWR
<ul style="list-style-type: none"><li>● Interface signals: M1927: This signal is control by CPU, it is applied in ASCWR MD:0<ul style="list-style-type: none"><li>: ON, it represents that the RTS (connect to the CTS of PLC) of the printer is “False”. I.e. the printer is not ready or abnormal.</li><li>: OFF, it represents that the RTS of the Printer is “True”; Printer is Ready.</li></ul></li></ul> <p>Note: Using the M1927 associates with timer can detect if the printer is abnormal or not.</p> <p>R4158: The setting of communication parameters (refer to section 12.6.2)</p>		

FUN 95 RAMP	Ramp Function for D/A Output	FUN 95 RAMP																																																																																													
<pre> graph LR     EN[EN↑] --&gt; 95RAMP[95.RAMP]     PAU[PAU] --&gt; 95RAMP     UDU[D] --&gt; 95RAMP     95RAMP --&gt; ERR[ERR]     95RAMP --&gt; ASL[ASL]     95RAMP --&gt; ASU[ASU]     95RAMP --&gt; Tn[Tn]     95RAMP --&gt; PV[PV]     95RAMP --&gt; SL[SL]     95RAMP --&gt; SU[SU]   </pre>	<p style="text-align: center;"><b>Ramp Function for D/A Output</b></p> <p>Tn : Timer for ramp function  PV : Preset value of ramp timer (the unit is 0.01 second)  or the increment value of every 0.01 second  SL : Lower limit value  (ramp floor value).  SU : Upper limit value  (ramp ceiling value).  D : Register storing current ramping value.  D+1 : Working register  SU, SL could be positive or negative value when incorporate with AO module application.</p>																																																																																														
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th> <th style="text-align: center; padding: 2px;">WX</th> <th style="text-align: center; padding: 2px;">WY</th> <th style="text-align: center; padding: 2px;">WM</th> <th style="text-align: center; padding: 2px;">WS</th> <th style="text-align: center; padding: 2px;">TMR</th> <th style="text-align: center; padding: 2px;">CTR</th> <th style="text-align: center; padding: 2px;">HR</th> <th style="text-align: center; padding: 2px;">IR</th> <th style="text-align: center; padding: 2px;">OR</th> <th style="text-align: center; padding: 2px;">SR</th> <th style="text-align: center; padding: 2px;">ROR</th> <th style="text-align: center; padding: 2px;">DR</th> <th style="text-align: center; padding: 2px;">K</th> </tr> <tr> <th style="text-align: left; padding: 2px;">Oper- rand</th> <th style="text-align: center; padding: 2px;">WX0 WX240</th> <th style="text-align: center; padding: 2px;">WY0 WY240</th> <th style="text-align: center; padding: 2px;">WM0 WM1896</th> <th style="text-align: center; padding: 2px;">WS0 WS984</th> <th style="text-align: center; padding: 2px;">T0 T255</th> <th style="text-align: center; padding: 2px;">C0 C255</th> <th style="text-align: center; padding: 2px;">R0 R3839</th> <th style="text-align: center; padding: 2px;">R3840 R3903</th> <th style="text-align: center; padding: 2px;">R3904 R3967</th> <th style="text-align: center; padding: 2px;">R3968 R4167</th> <th style="text-align: center; padding: 2px;">R5000 R8071</th> <th style="text-align: center; padding: 2px;">D0 D3071</th> <th style="text-align: center; padding: 2px;">16-bit +/- number</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">Tn</td> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;"></td> </tr> <tr> <td style="padding: 2px;">PV</td> <td style="text-align: center; padding: 2px;">○</td> </tr> <tr> <td style="padding: 2px;">SL</td> <td style="text-align: center; padding: 2px;">○</td> </tr> <tr> <td style="padding: 2px;">SU</td> <td style="text-align: center; padding: 2px;">○</td> </tr> <tr> <td style="padding: 2px;">D</td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○</td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	Oper- rand	WX0 WX240	WY0 WY240	WM0 WM1896	WS0 WS984	T0 T255	C0 C255	R0 R3839	R3840 R3903	R3904 R3967	R3968 R4167	R5000 R8071	D0 D3071	16-bit +/- number	Tn					○								PV	○	○	○	○	○	○	○	○	○	○	○	○	SL	○	○	○	○	○	○	○	○	○	○	○	○	SU	○	○	○	○	○	○	○	○	○	○	○	○	D	○	○	○	○	○	○	○	○	○	○	○*	○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K																																																																																		
Oper- rand	WX0 WX240	WY0 WY240	WM0 WM1896	WS0 WS984	T0 T255	C0 C255	R0 R3839	R3840 R3903	R3904 R3967	R3968 R4167	R5000 R8071	D0 D3071	16-bit +/- number																																																																																		
Tn					○																																																																																										
PV	○	○	○	○	○	○	○	○	○	○	○	○																																																																																			
SL	○	○	○	○	○	○	○	○	○	○	○	○																																																																																			
SU	○	○	○	○	○	○	○	○	○	○	○	○																																																																																			
D	○	○	○	○	○	○	○	○	○	○	○*	○																																																																																			
<b>Description</b>	<ul style="list-style-type: none"> <li>● Tn must be a 0.01 sec time base timer and never used in other part of program.</li> <li>● PV is the preset value of ramp timer. Its unit is 10ms (0.01 second).</li> <li>● When input control “EN ↑ ” changes from 0→1, it first reset the timer Tn to 0. When “U/D”=1 it will load the value of SL to register D. And when M1974 = 0 it will be increased by SU-SL / PV every 0.01 sec or when M1974 = 1 it will increase by PV every 0.01 sec. When the D value reaches the SU value the output “ASU” =1. When “U/D”=0 it will load the value of SU to register D. When M1974 = 0 it will be decreased by SU-SL / PV every 0.01 sec or when M1974 = 1 it will be decreased by PV every 0.01 sec. When the D value reaches the SL value the output “ASL” =1.</li> <li>● The ramping direction(U/D) is determined at the time when input control “EN ↑ ” changes from 0→1. After the output D start to ramp, the change of U/D is no effect.</li> <li>● If it is required to pause the ramping action, it must let the input control “PAU” = 1; when “PAU”=0, and the ramping action is not completed, it will continue to complete the ramping action.</li> <li>● The value of SU must be larger than SL, otherwise the ramp function will not be performed, and the output “ERR” will set to 1.</li> <li>● This instruction use the register D to store the output ramping value; if the application use the D/A module to send the speed command, then speed command can be derived from the RAMP function to get a more smooth movement.</li> <li>● In addition to use register D to store the ramping value, this instruction also used the register D+1 to act as internal working register; therefore the other part of program can not use the register D+1.</li> </ul>																																																																																														

## Slow up/Slow down instructions

FUN 95 RAMP	Ramp Function for D/A Output	FUN 95 RAMP
<b>Program example</b>		
<pre> M0 --- EN↑ M1 --- PAU M2 --- U/D               +--- 95.RAMP           Tn :T20           PV :R100           S_L :R101           S_U :R102           D :R103                       +--- ERR ---( )           +--- M101           +--- ASL ---( )           +--- M102           +--- ASU ---( )               +--- M0 --- EN                       +--- 8.MOV               S : R103               D : R3904   </pre>	<b>Ramp Function for D/A Output</b>	

Move the ramping value to AO output register R3904

T20: Ramp timer (timer with 0.01 second time base)

R100: preset value of ramp timer (the unit is 0.01 second, 100 for a second).

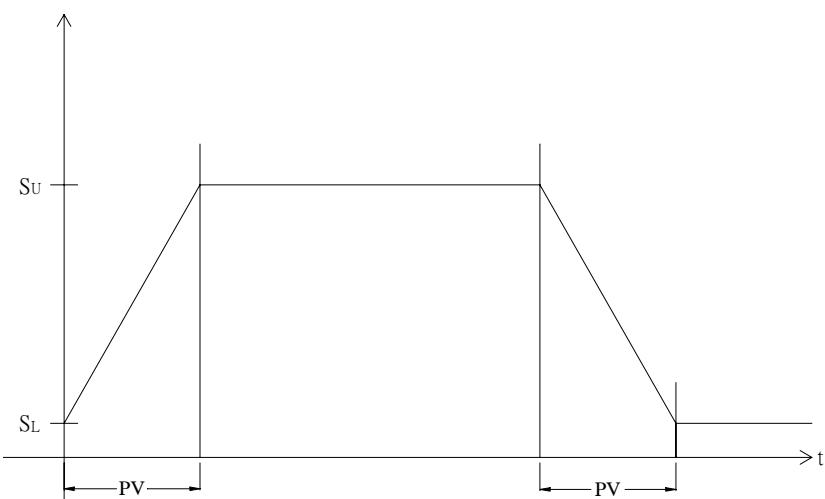
R101: Lower limit value.

R102: Upper limit value.

R103: Register storing current ramp value.

R104: Working register

- If M1974=0, When input control M0 changes from 0→1, it first reset the timer T20 to 0. If M2=1, it will load the R101 (lower limit) value into the R103, and it will increase the output with fixed value (R102-R101 / R100) for every 0.01 second and stores it to register R103. When the T2 timer going up to the preset value R100, the output value equals to R102, and the output M102 will set to 1. If M2=0, will load the R102 (upper limit) value into the R103, and it will decrease the output amount with fixed ratio (R102-R101 / R100) for every 0.01 second and store it to register R103. The T2 timer going up to the preset value R100, the output value equals to R102, and the output M101 will set to 1.
- M1=1, pause the ramping action.
- The value of R102 must be greater than R101, otherwise the ramp action will not be performed, and the output M100 will set to 1.



FUN 96 LINK2	Convenient Instruction for Communication Port2 (RS-485) (Brief description of function)	FUN 96 LINK2																													
	<p>MD : Communication mode, MD0~MD3. S : Starting register of communication program. Pt : Starting working register for instruction operation.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Range</th> <th>HR</th> <th>ROR</th> <th>DR</th> <th>K</th> </tr> </thead> <tbody> <tr> <td rowspan="2">Oper- and</td> <td>R0</td> <td>R5000</td> <td>D0</td> <td></td> </tr> <tr> <td>  R3839</td> <td>  R8071</td> <td>  D3071</td> <td></td> </tr> <tr> <td>MD</td> <td></td> <td></td> <td></td> <td>0~3</td> </tr> <tr> <td>S</td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td></td> </tr> <tr> <td>Pt</td> <td><input type="radio"/></td> <td><input type="radio"/>*</td> <td><input type="radio"/></td> <td></td> </tr> </tbody> </table>	Range	HR	ROR	DR	K	Oper- and	R0	R5000	D0		 R3839	 R8071	 D3071		MD				0~3	S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		Pt	<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>		
Range	HR	ROR	DR	K																											
Oper- and	R0	R5000	D0																												
	 R3839	 R8071	 D3071																												
MD				0~3																											
S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																												
Pt	<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>																												

- When use this instruction, the PLC will automatically set operation mode of port 2 to be "ladder instruction control interface" when the PLC is at RUN mode, and give the control right of port2 to ladder instruction. When PLC stops, the Port2 will return back to "standard interface" and not to be controlled by this instruction. This instruction provides 4 instruction modes MD0~MD3. Of which, three instruction modes MD0~MD2, are "regular link network", and the MD3 is the "high speed link network". The following are the function description of respective modes. For the details, please refer to section 13.1.2 for explanation.

- MD0 : Master station mode for FACON CPU LINK.

For any PLC, whose ladder program contains the FUN96:MD0 instruction, will become master station of FACON CPU LINK network. The master station PLC will base on the communication program stored in data registers in which the target station, data type, data length, etc, were specified to read or write slave station via "FACON FB-PLC Communication Protocol" command. With this approach up to 254 PLC stations can share the data each other

- MD1 : Active ASCII data transmission mode.

With this mode, the FUN96 instruction will parse the communication program stored in data registers and base on the parsing result send the data from port2 to ASCII peripherals (such as computer, other brand PLC, inverter, moving sign, etc, this kind of device can command by ASCII message). The operation can set to be (1) transmit only, which ignores the response from peripherals, (2) transmit and then to receive the response from peripherals. When operate with mode (2) then the user must base on the communication protocol of peripheral to parsing and prepare the response message by writing the ladder instructions.

- MD2 : Passive ASCII data receiving mode.

With this mode, the FUN96 will first wait to receive ASCII messages sent by external ASCII peripherals (such as computer, other brand PLC, card reader, bar code reader, electronic weight, etc. this kind of device can send ASCII message). Upon receiving the message, the user can base on the communication protocol of peripheral to parsing and react accordingly. The operation can set to (1) receive only without responding, or (2) receive then responding. For operation mode (2) the user can use the table driver method to write a communication program and after received a message this instruction can base on this communication program automatically reply the message to peripheral.

- MD3 : Master station mode of FACON high speed CPU LINK.

The most distinguished difference between this mode and MD0 is that the communication response of MD3 is much faster than MD0. With The introduction of MD3 mode CPU LINK, The FACON PLC can easily to implement the application of distributed control and real time data monitoring.

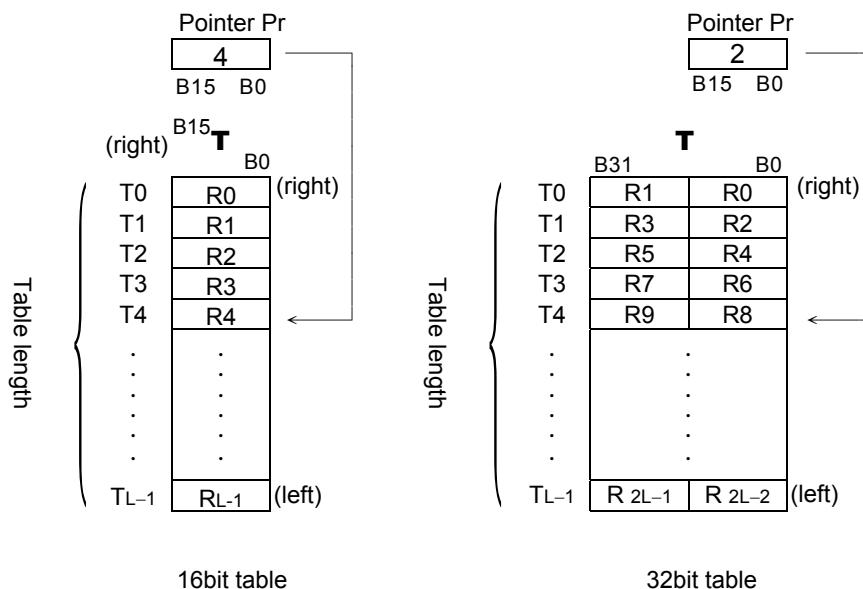
## Communication instructions

FUN 97 LINK1	Convenient Instruction for Communication Port2 (RS-485) (Brief description of function)	FUN 97 LINK1																									
	<p>Execution control—EN ↑</p> <p>Pause —PAU</p> <p>Abort—ABT</p> <pre> graph LR     EN[Execution control—EN ↑] --&gt; Box[97.LINK2]     PAU[Pause —PAU] --&gt; Box     ABT[Abort—ABT] --&gt; Box     MD[MD : ] --- ACT[ACT—]     S[S : ] --- ERR[ERR—]     Pt[Pt : ] --- DN[DN—]   </pre>	<p>MD : Communication mode, MD0~MD2.</p> <p>S : Starting register for communication program.</p> <p>Pt : Starting working register for instruction operation.</p>																									
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: center;">Range</th> <th style="text-align: center;">HR</th> <th style="text-align: center;">ROR</th> <th style="text-align: center;">DR</th> <th style="text-align: center;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Ope- rand</td> <td style="text-align: center;">R0   R3839</td> <td style="text-align: center;">R5000   R8071</td> <td style="text-align: center;">D0   D3071</td> <td></td> </tr> <tr> <td style="text-align: center;">MD</td> <td></td> <td></td> <td></td> <td style="text-align: center;">0~2</td> </tr> <tr> <td style="text-align: center;">S</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> <tr> <td style="text-align: center;">Pt</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/>*</td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> </tbody> </table> <ul style="list-style-type: none"> <li>● The operation and usage of the three instruction modes of MD0~MD2 is identical to that of MD0~MD2 for FUN96, please refer to FUN96(LINK2) and chapter 13 for explanation.</li> </ul>			Range	HR	ROR	DR	K	Ope- rand	R0   R3839	R5000   R8071	D0   D3071		MD				0~2	S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		Pt	<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>	
Range	HR	ROR	DR	K																							
Ope- rand	R0   R3839	R5000   R8071	D0   D3071																								
MD				0~2																							
S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																								
Pt	<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>																								

## Table Instructions

Fun No.	Mnemonic	Functionality	Fun No.	Mnemonic	Functionality
100	R→T	Register to table data move	107	T_FIL	Table fill
101	T→R	Table to register data move	108	T_SHF	Table shift
102	T→T	Table to table data move	109	T_ROT	Table rotate
103	BT_M	Block table move	110	QUEUE	Queue
104	T_SWP	Block table swap	111	STACK	Stack
105	R-T_S	Register to table search	112	BKCMP	Block compare
106	T-T_C	Table to table compare	113	SORT	Data Sort

- A table consists of 2 or more consecutive registers (16 or 32 bits). The number of registers that comprise the table is called the table length (L). The operation object of the table instructions always takes the register as unit (i.e. 16 or 32 bit data).
- The operation of table instructions are used mostly for data processing such as move, copy, compare, search etc, between tables and registers, or between tables. These instructions are convenient for application.
- Among the table instructions, most instructions use a pointer to specify which register within a table will be the target of operation. The pointer for both 16 and 32-bit table instructions will always be a 16-bit register. The effective range of the pointer is 0 to L-1, which corresponds to registers T<sub>0</sub> to T<sub>L-1</sub> (a total of L registers). The table shown below is a schematic diagram for 16-bit and 32-bit tables.
- Among the table operations, shift left/right, rotate left/right operations include a movement direction. The direction toward the higher register is called left, while the direction toward the lower register is called right, as shown in the diagram below.



## Table instructions

FUN100 <b>D P</b> R→T	REGISTER TO TABLE MOVE	FUN100 <b>D P</b> R→T																																																																																									
<p>Move control -EN↑</p> <p>Pointer increment -INC</p> <p>Pointer clear -CLR</p>	<p>100DP.R→T</p> <p>Rs : END— Move to end</p> <p>Td : ERR— Pointer error</p> <p>L : </p> <p>Pr : </p>	<p>Rs : Source data , can be constant or register</p> <p>Td : Source register for destination table</p> <p>L : Length of destination table</p> <p>Pr : Pointer register</p> <p>Rs, Td can associate with V,Z index register as indirect addressing</p>																																																																																									
<table border="1"> <thead> <tr> <th>Range \ Operand</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> <tr> <th>WX</th> <td>WX0   WX240</td> <td>WY0   WY240</td> <td>WM0   WM1896</td> <td>WS0   WS984</td> <td>T0   T255</td> <td>C0   C255</td> <td>R0   R3839</td> <td>R3840   R3903</td> <td>R3904   R3967</td> <td>R3968   R4167</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td>16/32bit +/- number</td> <td>V Z</td> </tr> </thead> <tbody> <tr> <td>Rs</td> <td>○</td> </tr> <tr> <td>Td</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td>○</td> <td></td> <td>○</td> </tr> <tr> <td>L</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○</td> <td></td> <td></td> <td></td> <td>○*</td> <td>○</td> <td>2~2048</td> <td></td> </tr> <tr> <td>Pr</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td>○</td> <td></td> <td></td> </tr> </tbody> </table>	Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	WX	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32bit +/- number	V Z	Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Td		○	○	○	○	○		○	○*	○*	○	○		○	L							○				○*	○	2~2048		Pr		○	○	○	○	○		○	○*	○*	○	○			
Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																													
WX	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32bit +/- number	V Z																																																																													
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																													
Td		○	○	○	○	○		○	○*	○*	○	○		○																																																																													
L							○				○*	○	2~2048																																																																														
Pr		○	○	○	○	○		○	○*	○*	○	○																																																																															
<ul style="list-style-type: none"> <li>When move control "EN" = 1 or "EN ↑" (<b>P</b> instruction) transition from 0 to 1, the contents of the source register Rs will be written onto the register Tdpr indicated by the pointer Pr within the destination table Td (length is L). Before executing, this instruction will first check the pointer clear "CLR" input signal. If "CLR" is 1, it will first clear the pointer Pr, and then carry out the move operation. After the move has been completed, it will then check the Pr value. If the Pr value has already reached L-1 (point to the last register in the table) then it will only set the move-to-end flag "END" to 1, and finish execution of this instruction. If the Pr value is less than L-1, then it must again check the pointer increment "INC" input signal. If "INC" is 1, then Pr value will be also increased. Besides, pointer clear "CLR" is able to operate independently, without being influenced by other input.</li> <li>The effective range of the pointer is 0 to L-1. Beyond this range, the pointer error "ERR" will be set to 1, and this instruction will not be performed.</li> </ul>	<p>● The example at left at the very beginning pointer Pr = 4, the entire content of table Td is 0, and the Rs value is 8888. The diagram below shows the operation results when X1 have the transition of 0→1 twice.</p> <p>● Because INC is 1, Pr will increase by 1 each time the instruction is executed.</p>																																																																																										

## Table instructions

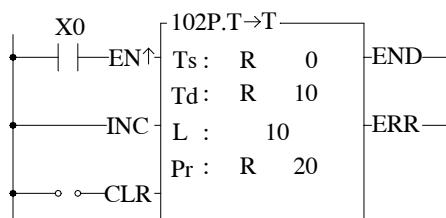
FUN101 <b>D P</b> T→R	TABLE TO REGISTER MOVE	FUN101 <b>D P</b> T→R																																																																																								
<p>Move control -EN↑</p> <p>Pointer increment -INC</p> <p>Pointer clear -CLR</p>	<pre>101DP.T→R Ts : R0 L : 9 Pr : R19 Rd: R20</pre> <p>-END— Move to end -ERR— Pointer error</p>	<p>Ts : Source table starting register L : Length of source table Pr : Pointer register Rd : Destination register Ts, Rd may combine with V, Z to serve indirect address application</p>																																																																																								
<table border="1" style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th> <th style="text-align: left; padding: 2px;">WX</th> <th style="text-align: left; padding: 2px;">WY</th> <th style="text-align: left; padding: 2px;">WM</th> <th style="text-align: left; padding: 2px;">WS</th> <th style="text-align: left; padding: 2px;">TMR</th> <th style="text-align: left; padding: 2px;">CTR</th> <th style="text-align: left; padding: 2px;">HR</th> <th style="text-align: left; padding: 2px;">IR</th> <th style="text-align: left; padding: 2px;">OR</th> <th style="text-align: left; padding: 2px;">SR</th> <th style="text-align: left; padding: 2px;">ROR</th> <th style="text-align: left; padding: 2px;">DR</th> <th style="text-align: left; padding: 2px;">K</th> <th style="text-align: left; padding: 2px;">XR</th> </tr> <tr> <th style="text-align: left; padding: 2px;">Oper- and</th> <th style="text-align: left; padding: 2px;">WX0</th> <th style="text-align: left; padding: 2px;">WY0</th> <th style="text-align: left; padding: 2px;">WM0</th> <th style="text-align: left; padding: 2px;">WS0</th> <th style="text-align: left; padding: 2px;">T0</th> <th style="text-align: left; padding: 2px;">C0</th> <th style="text-align: left; padding: 2px;">R0</th> <th style="text-align: left; padding: 2px;">R3840</th> <th style="text-align: left; padding: 2px;">R3904</th> <th style="text-align: left; padding: 2px;">R3968</th> <th style="text-align: left; padding: 2px;">R5000</th> <th style="text-align: left; padding: 2px;">D0</th> <th style="text-align: left; padding: 2px;">16/32bit +/- number</th> <th style="text-align: left; padding: 2px;">V ' Z</th> </tr> <tr> <th style="text-align: left; padding: 2px;">Ts</th> <td style="text-align: center; padding: 2px;">○</td> </tr> <tr> <th style="text-align: left; padding: 2px;">L</th> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;"></td> </tr> <tr> <th style="text-align: left; padding: 2px;">Pr</th> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;">2~2048</td> <td style="text-align: center; padding: 2px;"></td> </tr> <tr> <th style="text-align: left; padding: 2px;">Rd</th> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;"></td> <td style="text-align: center; padding: 2px;">○</td> </tr> </thead></table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	16/32bit +/- number	V ' Z	Ts	○	○	○	○	○	○	○	○	○	○	○	○	○	○	L							○	○	○	○	○*	○			Pr		○	○	○	○	○	○	○	○	○*	○*	○	2~2048		Rd		○	○	○	○	○	○	○	○	○*	○*	○		○
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																												
Oper- and	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	16/32bit +/- number	V ' Z																																																																												
Ts	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																												
L							○	○	○	○	○*	○																																																																														
Pr		○	○	○	○	○	○	○	○	○*	○*	○	2~2048																																																																													
Rd		○	○	○	○	○	○	○	○	○*	○*	○		○																																																																												

|  | ``` 101P.T→R Ts : R0 L : 9 Pr : R19 Rd: R20 ```   -END— Move to end -ERR— Pointer error | - When move control "EN" = 1 or "EN ↑" (**P** instruction) transition from 0 to 1, the value of the register Tspr specified by pointer Pr within source table Ts (length is L) will be written into the destination register Rd. Before executing, this instruction will first check the input signal of pointer clear "CLR". If "CLR" is 1, it will first clear Pr and then carry out the move operation. After completing the move operation, it will then check the value of Pr. If the Pr value has already reached L-1 (point to the last register in the table), then it sets the move-to-end flag to 1, and finishes executing of this instruction. If Pr is less than L-1, it check the status of "INC". If "INC" is 1, then it will increase Pr and finish the execution of this instruction. Besides, pointer clear "CLR" can execute independently and is not influenced by other inputs. - The effective range of the pointer is 0 to L-1. Beyond this range the pointer error "ERR" will be set to 1 and this instruction will not be carried out. |
|  |  |  |

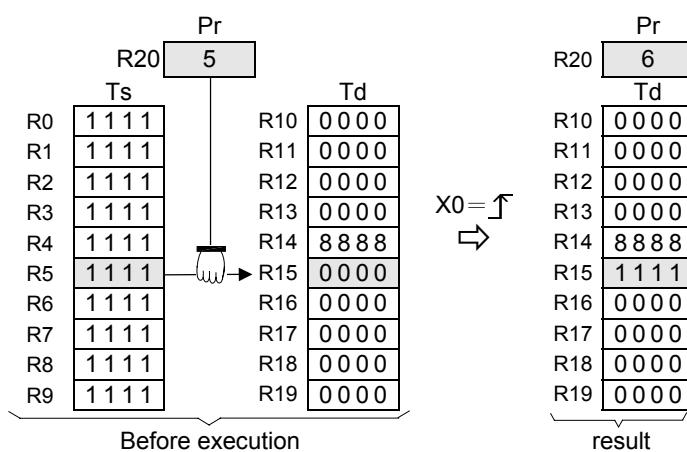
## Table instructions

FUN102 <b>D P</b> T→T	TABLE TO TABLE MOVE	FUN102 <b>D P</b> T→T																																																																																										
<p>Move control -EN↑</p> <p>Pointer increment -INC</p> <p>Pointer clear -CLR</p>	<p>102DP.T→T</p> <p>Ts :      END — Move to end</p> <p>Td :      ERR — Pointer error</p> <p>L :      Pr :</p>	<p>Ts : Starting number of source table register Td : Starting number of destination table register L : Table (Ts and Td) length Pr : Pointer register Ts, Rd may combine with V, Z to serve indirect address application</p>																																																																																										
	<table border="1"> <thead> <tr> <th>Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>Operand</td> <td>WX0   WX240</td> <td>WY0   WY240</td> <td>WM0   WM1896</td> <td>WS0   WS984</td> <td>T0   T255</td> <td>C0   C255</td> <td>R0   R3839</td> <td>R3840   R3903</td> <td>R3904   R3967</td> <td>R3968   R4167</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td>2   2048</td> <td>V   Z</td> </tr> <tr> <td>Ts</td> <td>○</td> </tr> <tr> <td>Td</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> <td>○</td> </tr> <tr> <td>L</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○</td> <td></td> <td></td> <td>○*</td> <td>○</td> <td>○</td> <td></td> <td></td> </tr> <tr> <td>Pr</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   2048	V   Z	Ts	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Td		○	○	○	○	○	○		○	○*	○*	○		○	L							○			○*	○	○			Pr		○	○	○	○	○	○		○	○*	○*	○			
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																														
Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   2048	V   Z																																																																														
Ts	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																														
Td		○	○	○	○	○	○		○	○*	○*	○		○																																																																														
L							○			○*	○	○																																																																																
Pr		○	○	○	○	○	○		○	○*	○*	○																																																																																

- When move control "EN" = 1 or "EN ↑" (**P** instruction) have a transition from 0 to 1, the register Tspr pointed by pointer Pr within the source table will be moved to a register Tdpr, which also pointed by the pointer Pr in the destination table. Before execution, it will first check the input signal of pointer clear "CLR". If "CLR" is 1, it will first clear Pr to 0 and then do the move (in this case Ts0→Td0). After the move action has been completed it will then check the value of pointer Pr. If the Pr value has already reached L-1 (point to the last register on the table), then it will set the move-to-end flag "END" to 1 and finish executing of this instruction. If the Pr value is less than L-1, it will check the status of "INC". If "INC" is 1, then the Pr value will be increased by 1 before execution. Besides, pointer clear "CLR" can execute independently, and will not be influenced by other input.
- The effective range of the pointer is 0 to L-1. Beyond this range, the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.



● The diagram at left below is the status before execution. When X0 from 0→1, the content of R5 in Ts table will copy to R15 and pointer R20 will be increased by 1.



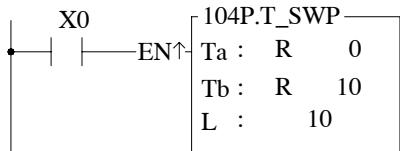
## Table instructions

FUN103 <b>D P</b> BT_M	BLOCK TABLE MOVE	FUN103 <b>D P</b> BT_M																																																																										
<p>Move control—EN↑</p> <p>103DP.BT_M</p> <p>Ts : R 0 Td : R 10 L : 10</p>	<p>Ts : Starting register for source table Td : Starting register for destination table L: Lengths of source and destination tables Ts, Rd may combine with V, Z to serve indire</p>																																																																											
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: left;">Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td style="background-color: #cccccc;">Oper- and</td> <td>WX0   WX240</td> <td>WY0   WY240</td> <td>WM0   WM1896</td> <td>WS0   WS984</td> <td>T0   T255</td> <td>C0   C255</td> <td>R0   R3839</td> <td>R3840   R3903</td> <td>R3904   R3967</td> <td>R3968   R4167</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td>2   256</td> <td>V   Z</td> </tr> <tr> <td>Ts</td> <td>○</td> </tr> <tr> <td>Td</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td>○</td> <td></td> <td>○</td> </tr> <tr> <td>L</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○</td> <td></td> <td></td> <td></td> <td>○*</td> <td>○</td> <td>○</td> <td></td> <td>○</td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z	Ts	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Td		○	○	○	○	○	○	○	○*	○*	○	○		○	L						○				○*	○	○		○	<ul style="list-style-type: none"> <li>● In this instruction the source table and destination table are the same length. When this instruction was executed all the data in the Ts table is completely copied to Td. No pointer is involved in this instruction.</li> <li>● When move control "EN" = 1 or "EN ↑" (<b>P</b> instruction) have a transition from 0 to 1, all the data from source table Ts (length L) is copied to the destination table Td, which is the same length.</li> <li>● One table is completely copied every time this instruction is executed, so if the table length is long, it will be very time consuming. In practice, P modifier should be used to avoid time waste caused by each scan repeating the same movement action.</li> </ul> <p>● The diagram at left below is the status before execution. When X0 from 0→1, the content of R0~R9 in Ts table will copy to R10~R19.</p> <p>X0</p> <p>EN↑</p> <p>103P.BT_M</p> <p>Ts : R 0 Td : R 10 L : 10</p> <p>X0=↑</p> <p>Before executed</p> <p>Execute result</p>
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																														
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z																																																														
Ts	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																														
Td		○	○	○	○	○	○	○	○*	○*	○	○		○																																																														
L						○				○*	○	○		○																																																														

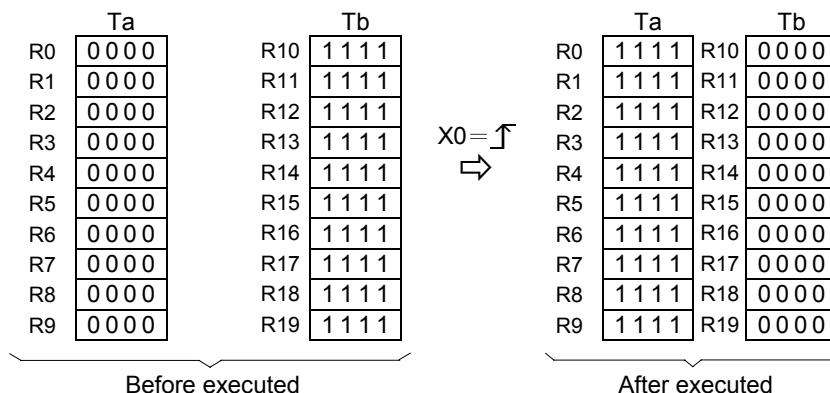
## Table instructions

FUN104 <b>D P</b> T_SWP	BLOCK TABLE SWAP	FUN104 <b>D P</b> T_SWP																																																													
<p>Move control—EN↑</p> <p>104DP.T_SWP</p> <p>Ta : _____</p> <p>Tb : _____</p> <p>L : _____</p>	<p>Ta : Starting register of Table a Tb : Starting register of Table b L : Lengths of Table a and b Ts, Rd may combine with V, Z to serve indirect address application</p>																																																														
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: center;">Range</th> <th style="text-align: center;">WY</th> <th style="text-align: center;">WM</th> <th style="text-align: center;">WS</th> <th style="text-align: center;">TMR</th> <th style="text-align: center;">CTR</th> <th style="text-align: center;">HR</th> <th style="text-align: center;">OR</th> <th style="text-align: center;">SR</th> <th style="text-align: center;">ROR</th> <th style="text-align: center;">DR</th> <th style="text-align: center;">K</th> <th style="text-align: center;">XR</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Oper- and</td> <td style="text-align: center;">WY0   WY240</td> <td style="text-align: center;">WM0   WM1896</td> <td style="text-align: center;">WS0   WS984</td> <td style="text-align: center;">T0   T255</td> <td style="text-align: center;">C0   C255</td> <td style="text-align: center;">R0   R3839</td> <td style="text-align: center;">R3904   R3967</td> <td style="text-align: center;">R3968   R4167</td> <td style="text-align: center;">R5000   R8071</td> <td style="text-align: center;">D0   D3071</td> <td style="text-align: center;">2   256</td> <td style="text-align: center;">V   Z</td> </tr> <tr> <td style="text-align: center;">Ta</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td style="text-align: center;">○*</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○</td> </tr> <tr> <td style="text-align: center;">Tb</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○*</td> <td style="text-align: center;">○*</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○</td> </tr> <tr> <td style="text-align: center;">L</td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;">○</td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;">○*</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○</td> <td style="text-align: center;">○</td> </tr> </tbody> </table>	Range	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	XR	Oper- and	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z	Ta	○	○	○	○	○	○	○	○*	○*	○	○	Tb	○	○	○	○	○	○	○	○*	○*	○	○	L					○			○*	○	○	○	
Range	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	XR																																																			
Oper- and	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z																																																			
Ta	○	○	○	○	○	○	○	○*	○*	○	○																																																				
Tb	○	○	○	○	○	○	○	○*	○*	○	○																																																				
L					○			○*	○	○	○																																																				

- This instruction swaps the contents of Tables a and b, so the table must be the same length, and the registers in the table must be writeable. Since a complete swap is done with each time the instruction is executed, no pointer is needed.
- When move control "EN" = 1 or "EN ↑" (**P** instruction) have a transition from 0 to 1, the contents of Table a and Table b will be completely swapped.
- This instruction will swap all the registers specified in L each time the instruction is executed, so if the table length is big, it will be very time consuming, therefore P instruction should be used.

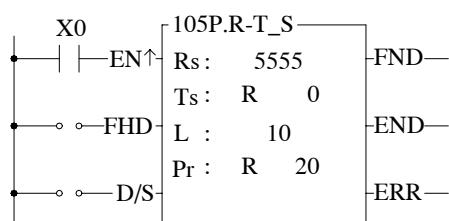


- The diagram at left below is the status before execution. When X0 from 0→1, the contents of R0~R9 in Ts table will swap with R10~R19.

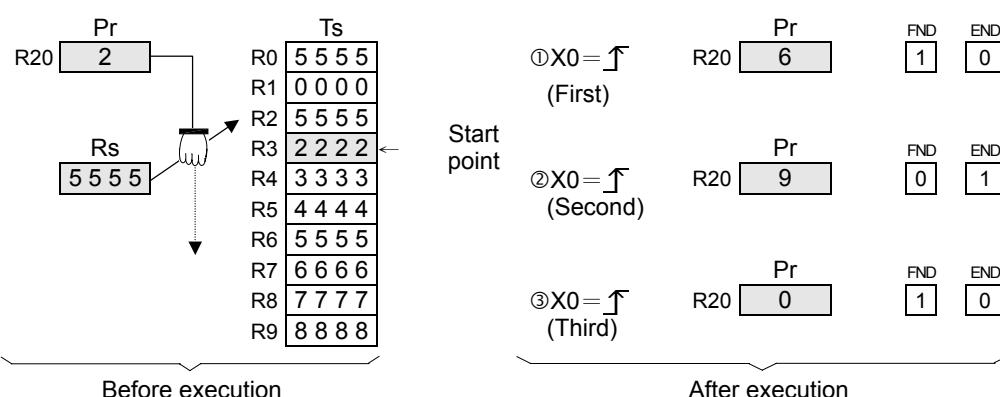


FUN105 DP R-T_S	REGISTER TO TABLE SEARCH	FUN105 DP R-T_S																																																																										
<p>Search control -EN↑ Search from head -FHD Different/same option -D/S</p> <table border="1"> <thead> <tr> <th>Range \ Operand</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>Rs</td> <td>○</td> <td>V Z</td> </tr> <tr> <td>Ts</td> <td>○</td> <td></td> </tr> <tr> <td>L</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○</td> <td></td> <td></td> <td></td> <td>○*</td> <td>○</td> <td>2~256</td> <td></td> </tr> <tr> <td>Pr</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> </tbody> </table>	Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	V Z	Ts	○	○	○	○	○	○	○	○	○	○	○	○	○		L							○				○*	○	2~256		Pr		○	○	○	○	○	○		○	○*	○*	○			<p>Rs : Data to search, It can be a constant or a register Ts : Starting register of table being searched L : Label length Pr : Pointer of table Rs, Ts may combine with V, Z to serve indirect address application</p>
Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																														
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	V Z																																																														
Ts	○	○	○	○	○	○	○	○	○	○	○	○	○																																																															
L							○				○*	○	2~256																																																															
Pr		○	○	○	○	○	○		○	○*	○*	○																																																																

- When search control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, will search from the first register of Table Ts (when "FHD" = 1 or Pr value has reached L-1), or from the next register (Tspr + 1) pointed by the pointer within the table ("FHD" = 0, while Pr value is less than L-1) to find the first data different with Rs(when D/S = 1) or find the first data the same with Rs (when D/S = 0). If it find a data match the condition it will immediately stop the search action, and the pointer Pr will point to that data and found objective flag "FND" will set to 1. When the searching has searched to the last register of the table, the execution of the instruction will stop, whether it was found or not. In that case the search-to-end flag "END" will be set to 1 and the Pr value will stop at L-1. When this instruction next time is executed, Pr will automatically return to the head of the table (Pr = 0) before the search begin.
- The effective range of Pr is 0 to L-1. If the value exceeds this range then the pointer error flag "ERR" will change to 1, and this instruction will not be carried out.



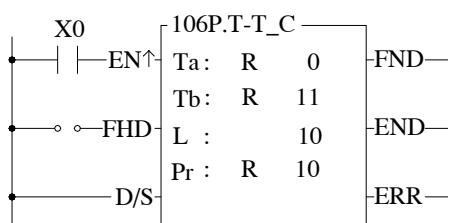
- The instruction at left is searching the table for a register with the value 5555 (because D/S = 0, it is searching for same value). Before execution, the pointer point to R2, but the starting point of the search is Pr + 1 (i.e. it starts from R3). After X0 has transition from 0→1 3 times, the results of each search may be obtained as shown in the diagram below.



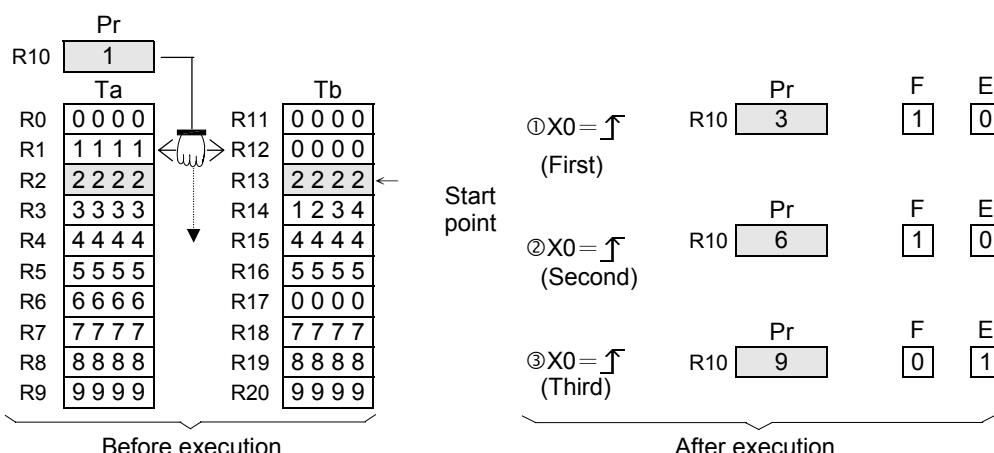
## Table instructions

FUN106 <b>D P</b> T-T_C	TABLE TO TABLE COMPARE	FUN106 <b>D P</b> T-T_C																																																																																										
<p>Comparison control -EN↑</p> <p>Compare from head -FHD</p> <p>Different/same option -D/S</p>	<p>106DP.T-T_C</p> <p>Ta : FND— Found objective Tb : END— Compare to end L : Pr : ERR— Pointer error</p>	<p>Ta : Starting register of Table a Tb : Starting register of Table b L : Lengths of Table Pr : Pointer Ta, Tb may combine with V, Z to serve indirect address application</p>																																																																																										
	<table border="1"> <thead> <tr> <th>Range</th><th>WX</th><th>WY</th><th>WM</th><th>WS</th><th>TMR</th><th>CTR</th><th>HR</th><th>IR</th><th>OR</th><th>SR</th><th>ROR</th><th>DR</th><th>K</th><th>XR</th></tr> </thead> <tbody> <tr> <td>Operand</td><td>WX0   WX240</td><td>WY0   WY240</td><td>WM0   WM1896</td><td>WS0   WS984</td><td>T0   T255</td><td>C0   C255</td><td>R0   R3840</td><td>R3904   R3903</td><td>R3967   R3968</td><td>R5000   R4167</td><td>D0   R8071</td><td>  D3071</td><td>2   256</td><td>V   Z</td></tr> <tr> <td>Ta</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr> <tr> <td>Tb</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr> <tr> <td>L</td><td></td><td></td><td></td><td></td><td></td><td></td><td>○</td><td></td><td></td><td>○*</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>Pr</td><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td><td>○</td><td>○*</td><td>○</td><td>○</td><td></td><td></td></tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3840	R3904   R3903	R3967   R3968	R5000   R4167	D0   R8071	 D3071	2   256	V   Z	Ta	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Tb	○	○	○	○	○	○	○	○	○	○	○	○	○	○	L							○			○*	○	○	○		Pr		○	○	○	○	○	○		○	○*	○	○			
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																														
Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3840	R3904   R3903	R3967   R3968	R5000   R4167	D0   R8071	 D3071	2   256	V   Z																																																																														
Ta	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																														
Tb	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																														
L							○			○*	○	○	○																																																																															
Pr		○	○	○	○	○	○		○	○*	○	○																																																																																

- When comparison control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, then starting from the first register in the tables Ta and Tb (when "FHD" = 1 or Pr value has reached L-1) or starting from the next pair of registers (Ta<sub>pr+1</sub> and Tb<sub>pr+1</sub>) pointed by Pr ("FHD" = 0, while Pr is less than L-1), this instruction will search for pairs of registers with different values (when "D/S" = 1) or the same value (when "D/S" = 0). When search found (either different or the same), it will immediately stop the search and the pointer Pr will point to the register pairs met the search criteria. The found flag "FND" will be set to 1. When it has searched to the last register of the table, the instruction will stop executing. whether it found or not. The compare-to-end flag "END" will be set to 1, and the pointer value will stop at L-1. When this instruction is executed next time, Pr will automatically return to the head of the table to begin the search.
- The effective range of Pr is 0 to L-1. The Pr value should not changed by other programs during the operation. As this will affect the result of the search. If the Pr value not in the effective range, the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.

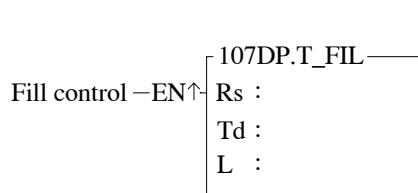


The instruction at left starts from the register next to the register pointed by the pointer (because "FHD" is 0) to search for register pairs with different data (because "D/S" is 1) within the 2 tables. At the very beginning, Pr points to Ta1 and Tb1. There are 3 different pairs of data at the position 1,3,6 of the table. However, it does not compare from the beginning, and this instruction will start searching from position 3 downwards. After X0 has changed 3 times from 0 to 1, the results are shown in the diagram below.



## Table instructions

FUN107 <b>D P</b>	TABLE FILL	FUN107 <b>D P</b>
T_FIL		T_FIL



Rs : Source data to fill, can be a constant or a register

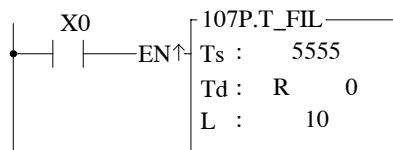
Td : Starting register of destination table

L : Table length

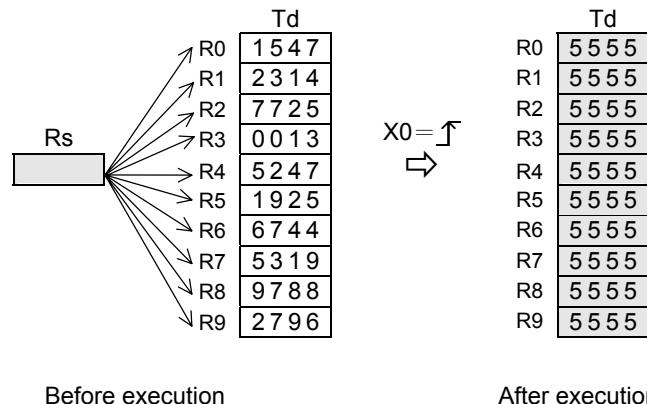
Rs, Td may combine with V, Z to serve indirect address application

Oper- and	Range		WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V ` Z		
Ts	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
Td		○	○	○	○	○	○		○	○*	○*	○	○		○	
L						○				○*	○	○	2~256			

- When fill control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, the Rs data will be filled into all the registers of the table Td.
- This instruction is mainly used for clearing the table (fill 0) or unifying the table (filling in the same values). It should be used with the P instruction.



- The instruction at left will fill 5555 into the whole table Td. The results are as shown in the diagram below.



Before execution

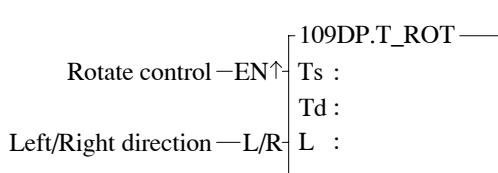
After execution

## Table instructions

FUN108 <b>D P</b> T_SHF	TABLE SHIFT	FUN108 <b>D P</b> T_SHF																																																																																																																																																																																																																																																																																																											
<p>Shift control—EN↑</p> <p>Left/Right direction—L/R</p> <p>IW : 108DP.T_SHF</p> <p>Ts :</p> <p>Td :</p> <p>L :</p> <p>OW:</p>	<p>IW : Data to fill the room after shift operation, can be a constant or a register</p> <p>Ts : Source table</p> <p>Td : Destination table storing shift results</p> <p>L : Lengths of tables Ts and Td</p> <p>OW: Register to accept the shifted out data</p> <p>Ts, Td may combine with V, Z to serve indirect address application</p>																																																																																																																																																																																																																																																																																																												
<table border="1"> <thead> <tr> <th>Range \ Operand</th><th>WX</th><th>WY</th><th>WM</th><th>WS</th><th>TMR</th><th>CTR</th><th>HR</th><th>IR</th><th>OR</th><th>SR</th><th>ROR</th><th>DR</th><th>K</th><th>XR</th></tr> </thead> <tbody> <tr> <td>WX0</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>V</td></tr> <tr> <td>WX240</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>Z</td></tr> <tr> <td>WM</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>WS</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>T255</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>C0</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>C255</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>R0</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>R3840</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>R3903</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>R3967</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>R4167</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>R5000</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>R8071</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>D0</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>D3071</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>16/32-bit +/- number</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>V</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> <tr> <td>Z</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td></tr> </tbody> </table>	Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	WX0	○	○	○	○	○	○	○	○	○	○	○	○	○	V	WX240	○	○	○	○	○	○	○	○	○	○	○	○	○	Z	WM	○	○	○	○	○	○	○	○	○	○	○	○	○		WS	○	○	○	○	○	○	○	○	○	○	○	○	○		T255	○	○	○	○	○	○	○	○	○	○	○	○	○		C0	○	○	○	○	○	○	○	○	○	○	○	○	○		C255	○	○	○	○	○	○	○	○	○	○	○	○	○		R0	○	○	○	○	○	○	○	○	○	○	○	○	○		R3840	○	○	○	○	○	○	○	○	○	○	○	○	○		R3903	○	○	○	○	○	○	○	○	○	○	○	○	○		R3967	○	○	○	○	○	○	○	○	○	○	○	○	○		R4167	○	○	○	○	○	○	○	○	○	○	○	○	○		R5000	○	○	○	○	○	○	○	○	○	○	○	○	○		R8071	○	○	○	○	○	○	○	○	○	○	○	○	○		D0	○	○	○	○	○	○	○	○	○	○	○	○	○		D3071	○	○	○	○	○	○	○	○	○	○	○	○	○		16/32-bit +/- number	○	○	○	○	○	○	○	○	○	○	○	○	○		V	○	○	○	○	○	○	○	○	○	○	○	○	○		Z	○	○	○	○	○	○	○	○	○	○	○	○	○		
Range \ Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																																																																																																																																																																																																																																															
WX0	○	○	○	○	○	○	○	○	○	○	○	○	○	V																																																																																																																																																																																																																																																																																															
WX240	○	○	○	○	○	○	○	○	○	○	○	○	○	Z																																																																																																																																																																																																																																																																																															
WM	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
WS	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
T255	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
C0	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
C255	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
R0	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
R3840	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
R3903	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
R3967	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
R4167	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
R5000	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
R8071	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
D0	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
D3071	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
16/32-bit +/- number	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
V	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
Z	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																																																																																																																																																																																																
<p>● When shift control "EN" = 1 or "EN ↑ " (<b>P</b> instruction) has a transition from 0 to 1, all the data from table Ts will be taken out and shifted one position to the left (when "L/R" = 1) or to the right (when "L/R" = 0). The room created by the shift operation will be filled by IW and the results will be written into table Td. The data shifted out will be written into OW.</p> <p>X0 — EN↑</p> <p>X1 — L/R</p> <p>IW : 108P.T_SHF</p> <p>Ts : R 10</p> <p>Td : R 0</p> <p>L : 10</p> <p>OW: R 11</p>	<p>● In the program at left, Ts and Td is the same table. Therefore, the table shifts itself and then writes back to itself (the table must be writable). It first perform a shift left operation (let X1 = 1, and X0 go from 0→1) then perform a shift to right operation (let X1 = 0, and makes X0 go from 0→1). The result are shown at right in the diagram below.</p>																																																																																																																																																																																																																																																																																																												
<p>(Shift left)</p> <p>(Shift right)</p> <p>Ts(Td)</p> <p>(Shift left)</p> <p>(Shift right)</p> <p>OW</p> <p>Before execution</p> <p>①First time</p> <p>②Second time</p>																																																																																																																																																																																																																																																																																																													

## Table instructions

FUN109 <b>DP</b> T_ROT	<b>TABLE ROTATE</b>	FUN109 <b>DP</b> T_ROT
---------------------------	---------------------	---------------------------



Ts : Source table for rotate

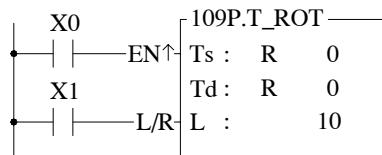
Td : Destination table storing results of rotation

L : Lengths of table

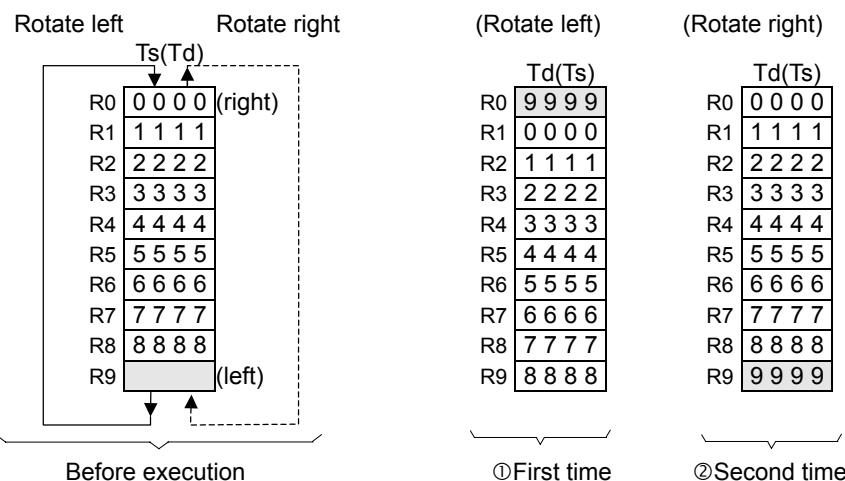
Ts, Td may combine with V, Z to serve indirect address application

Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z
Ts	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Td		○	○	○	○	○	○	○	○	○*	○*	○	○	○
L							○			○*	○	○	○	

- When rotation control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, the data from the table of Ts will be rotated 1 position to the left (when "L/R" = 1) or 1 position to the right (when "L/R" = 0). The results of the rotation will then be written onto table Td.



- In the program at left, Ts and Td is the same table. The table after rotation will write back to itself. It first perform one left rotation (let X1 = 1, and X0 go from 0→1), and then performs one right rotation (let X1 = 0, and X0 go from 0→1). The results are shown at right in the diagram below.



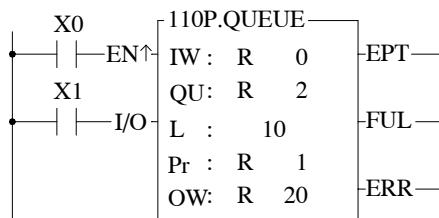
## Table instructions

FUN110 <b>D P</b> QUEUE		QUEUE													FUN110 <b>D P</b> QUEUE																																																																																																																						
IW : Data pushed into queue, can be a constant or a register																																																																																																																																					
QU : Starting register of queue																																																																																																																																					
L : Size of queue																																																																																																																																					
Pr : Pointer register																																																																																																																																					
OW : Register accepting data popped out from queue																																																																																																																																					
QU may combine with V, Z to serve indirect address application																																																																																																																																					
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th> <th style="padding: 2px;">WX</th> <th style="padding: 2px;">WY</th> <th style="padding: 2px;">WM</th> <th style="padding: 2px;">WS</th> <th style="padding: 2px;">TMR</th> <th style="padding: 2px;">CTR</th> <th style="padding: 2px;">HR</th> <th style="padding: 2px;">IR</th> <th style="padding: 2px;">OR</th> <th style="padding: 2px;">SR</th> <th style="padding: 2px;">ROR</th> <th style="padding: 2px;">DR</th> <th style="padding: 2px;">K</th> <th style="padding: 2px;">XR</th> <th colspan="2" style="text-align: right; padding: 2px;">16/32-bit +/- number</th> </tr> <tr> <th style="text-align: left; padding: 2px;">Oper- and</th> <td style="padding: 2px;">WX0   WX240</td> <td style="padding: 2px;">WY0   WY240</td> <td style="padding: 2px;">WM0   WM1896</td> <td style="padding: 2px;">WS0   WS984</td> <td style="padding: 2px;">T255</td> <td style="padding: 2px;">C0   C255</td> <td style="padding: 2px;">R0   R3839</td> <td style="padding: 2px;">R3840   R3903</td> <td style="padding: 2px;">R3904   R3967</td> <td style="padding: 2px;">R3968   R4167</td> <td style="padding: 2px;">R5000   R8071</td> <td style="padding: 2px;">D0   D3071</td> <td colspan="2" style="text-align: right; padding: 2px;">V . Z</td> </tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;">IW</td> <td style="padding: 2px;">○</td> <td colspan="2" style="text-align: right; padding: 2px;"></td> </tr> <tr> <td style="text-align: left; padding: 2px;">QU</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○*</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○</td> <td colspan="2" style="text-align: right; padding: 2px;"></td> </tr> <tr> <td style="text-align: left; padding: 2px;">L</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○*</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○</td> <td colspan="2" style="text-align: right; padding: 2px;">2~256</td> </tr> <tr> <td style="text-align: left; padding: 2px;">Pr</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○*</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○</td> <td colspan="2" style="text-align: right; padding: 2px;"></td> </tr> <tr> <td style="text-align: left; padding: 2px;">OW</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○*</td> <td style="padding: 2px;">○</td> <td style="padding: 2px;">○</td> <td colspan="2" rowspan="3" style="text-align: right; padding: 2px;"></td> </tr> </tbody> </table>																	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	16/32-bit +/- number		Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	V . Z		IW	○	○	○	○	○	○	○	○	○	○	○	○	○	○			QU	○	○	○	○	○	○	○	○	○	○	○	○*	○	○			L	○	○	○	○	○	○	○	○	○	○	○	○*	○	○	2~256		Pr	○	○	○	○	○	○	○	○	○	○	○	○*	○	○			OW	○	○	○	○	○	○	○	○	○	○	○	○*	○	○		
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	16/32-bit +/- number																																																																																																																						
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	V . Z																																																																																																																								
IW	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																																																							
QU	○	○	○	○	○	○	○	○	○	○	○	○*	○	○																																																																																																																							
L	○	○	○	○	○	○	○	○	○	○	○	○*	○	○	2~256																																																																																																																						
Pr	○	○	○	○	○	○	○	○	○	○	○	○*	○	○																																																																																																																							
OW	○	○	○	○	○	○	○	○	○	○	○	○*	○	○																																																																																																																							
<ul style="list-style-type: none"> <li>Queue is also a kind of table. It is different from ordinary table in that its queue register numbers go from 1 to L and not from 0 to L-1. In other words QU<sub>1</sub>~QU<sub>L</sub> respectively correspond to pointers Pr = 1 to L, and Pr = 0 is used to show that the queue is empty.</li> <li>Queue is a first in first out (FIFO) device, i.e. - the data that first pushed into the queue will be the first to pop out from the queue. A queue is comprised of L consecutive 16 or 32 bit registers (<b>D</b> instruction) starting from the QU register, as in the diagram below:</li> </ul>																																																																																																																																					
<p>① ~ ⑤ is the sequence number of operation</p>																																																																																																																																					

- When execution control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, the status of in/out control "I/O" determines whether the IW data will be pushed into the queue (when "I/O" = 1) or be popped out and transferred to OW (when "I/O" = 0). As shown in the diagram above, the IW data will always be pushed into the first (QU1) register of the queue. After it has been pushed in, Pr will immediately be increased by 1, so that the pointer can always point to the first data that was pushed into the queue. When it is popped out, the data pointed by Pr will be transferred directly to OW. Pr will be reduced by 1, so that it still point to the first data remained in the queue.

FUN110 <b>D P</b> QUEUE	QUEUE	FUN110 <b>D P</b> QUEUE
----------------------------	-------	----------------------------

- If no data has yet been pushed into the queue or the pushed in data has already been popped out ( $Pr = 0$ ), then the queue empty flag will be set to 1. In this case, even if there is further popping out action, this instruction will not be executed. If data is only pushed in and not popped out, or pushed in is more than that popped out, then the queue finally becomes full (pointer  $Pr$  indicates the  $QU_L$  position), and the queue full flag is changed to 1. In this case, if there is more pushing in action, this instruction will not execute. The pointer for this instruction is used during access of the queue, to indicate the data that was pushed in the earliest. Other programs should not be allowed to change it, or else an operation error will be created. If there is a specific application, which requires the setting of a  $Pr$  value, then its permissible range is 0 to L (0 means empty, and 1 to L respectively correspond to  $QU_1$  to  $QUL$ ). Beyond this range, the pointer error flag "ERR" will be set as 1, and this instruction will not be carried out.



- The program at left assumes the queue content is the same with the queue at preceding page. It will first perform queue push operation , and then perform pop out action. The results are shown below. Under any circumstance,  $Pr$  always point to the first (oldest) data that was remained in queue.

Pr	5
QU	
QU1	5 5 5 5 R2
QU2	4 4 4 4 R3
QU3	3 3 3 3 R4
QU4	2 2 2 2 R5
QU5	1 1 1 1 R6
QU6	
QU7	
QU8	
QU9	
QU10	

OW  
x x x R20  
↑  
OW unchanged

After push in (X1=1 , X0 from 0→1)

Pr	4
QU	
QU1	5 5 5 5 R2
QU2	4 4 4 4 R3
QU3	3 3 3 3 R4
QU4	2 2 2 2 R5
QU5	
QU6	
QU7	
QU8	
QU9	
QU10	

OW  
1 1 1 1 R20  
↑  
OW unchanged

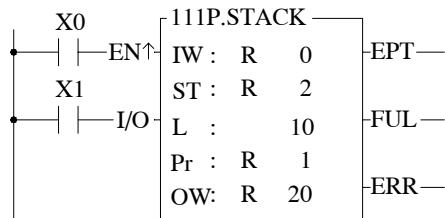
After pop off (X1=0 , X0 from 0→1)

## Table instructions

FUN111 <b>D P</b> STACK	STACK	FUN111 <b>D P</b> STACK																																																																																																					
Execution control — EN↑	IW : ST : L : Pr : OW:	EPT — Stack empty FUL — Stack full ERR — Pointer error																																																																																																					
In/Out control — I/O		IW : Data pushed into stack, can be a constant or a register ST : Starting register of stack L : Size of stack Pr : Pointer register OW : Register accepting data popped out from stack ST may combine with V, Z to serve indirect address application																																																																																																					
	<table border="1"> <thead> <tr> <th>Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> <tr> <th>Oper- and</th> <th>WX0   WX240</th> <th>WY0   WY240</th> <th>WM0   WM1896</th> <th>WS0   WS984</th> <th>T0   T255</th> <th>C0   C255</th> <th>R0   R3839</th> <th>R3840   R3903</th> <th>R3904   R3967</th> <th>R3968   R4167</th> <th>R5000   R8071</th> <th>D0   D3071</th> <th>16/32-bit +/- number</th> <th>V . Z</th> </tr> </thead> <tbody> <tr> <td>IW</td> <td>○</td> <td></td> </tr> <tr> <td>ST</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> </tr> <tr> <td>L</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○</td> <td></td> <td></td> <td>○*</td> <td>○</td> <td>○</td> <td>2~256</td> </tr> <tr> <td>Pr</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> </tr> <tr> <td>OW</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V . Z	IW	○	○	○	○	○	○	○	○	○	○	○	○	○		ST		○	○	○	○	○	○		○	○*	○*	○		L							○			○*	○	○	2~256	Pr		○	○	○	○	○	○		○	○*	○*	○		OW	○	○	○	○	○	○	○	○	○	○*	○*	○		
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																																									
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number	V . Z																																																																																									
IW	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																										
ST		○	○	○	○	○	○		○	○*	○*	○																																																																																											
L							○			○*	○	○	2~256																																																																																										
Pr		○	○	○	○	○	○		○	○*	○*	○																																																																																											
OW	○	○	○	○	○	○	○	○	○	○*	○*	○																																																																																											
		<ul style="list-style-type: none"> <li>Like queue, stack is also a kind of table. The nature of its pointer is exactly the same as with queue, i.e. Pr = 1 to L, which corresponds to ST<sub>1</sub> to ST<sub>L</sub>, and when Pr = 0 the stack is empty.</li> <li>Stack is the opposite of queue, being a last in first out (LIFO) device. This means that the data that was most recently pushed into the stack will be the first to be popped out of the stack. The stack is comprised of L consecutive 16 or 32-bit (<b>D</b> instruction) registers starting from ST, as shown in the following diagram:</li> </ul> <pre>     graph TD         Pr[Pr 4] --&gt; ST[ST]         ST -- "Bottom of stack" --&gt; ST5[ST5 ⑤555]         ST5 -- push --&gt; ST4[ST4 ④4444]         ST4 -- push --&gt; ST3[ST3 ③3333]         ST3 -- push --&gt; ST2[ST2 ②2222]         ST2 -- push --&gt; ST1[ST1 ①1111]         ST1 -- push --&gt; ST5         ST5 -- "I/O=1" --&gt; IW_IW[IW ⑤555]         IW_IW --&gt; ST5         ST5 -- "I/O=0" --&gt; OW_xxxx[OW xxxx]         OW_xxxx --&gt; ST5         ST5 -- "I/O=0" --&gt; Pop[pop(I/O=0)]         Pop --&gt; ST5         ST5 -- "I/O=0" --&gt; Pr_Dec[1.STpr→OW 2.Pr-1→Pr]         Pr_Dec --&gt; ST5     </pre>																																																																																																					
		<ul style="list-style-type: none"> <li>When execution control "EN" = 1 or "EN ↑" (<b>P</b> instruction) has a transition from 0 to 1, the status of in/out control "I/O" determines whether the IW data will be pushed into the stack (when "I/O" = 1), or the data pointed by Pr within the stack (the data most recently pushed into the stack) will be moved out and transferred to OW (when "I/O" = 0). Note that the data pushed in is stacking, so before pushed in, Pr will increase by 1 to point to the top of the stack then the data will be pushed in. When it is popped out, the data pointed by pointer Pr (the most recently pushed in data) will be transferred to OW. After then Pr will decrease by 1. Under any circumstances, the pointer Pr will always point to the data that was pushed into the stack most recently.</li> </ul>																																																																																																					

FUN111 <b>D P</b> STACK	STACK	FUN111 <b>D P</b> STACK
----------------------------	-------	----------------------------

- When no data has yet been pushed into the stack or the pushed in data has already been popped out ( $Pr = 0$ ), the stack empty flag "EPT" will set to 1. In this case any further pop up actions, will be ignored. If more data is pushed than popped out, sooner or latter the stack will be full (pointer  $Pr$  points to  $ST_L$  position), and the stack full flag "FUL" will set to 1. In this case any further push actions, will be ignored. As with queue, the stack pointer in normal case should not be changed by other instructions. If there is a special application which requires to set the  $Pr$  value, then its effective range is 0 to  $L$  (0 means empty, 1 to  $L$  respectively correspond to  $ST_1$  to  $ST_L$ ). Beyond this range, the pointer error flag "ERR" will set to 1, and the instruction will not be carried out.



- The program at left assumes that the initial content of the stack is just as in the diagram of a stack on the preceding page. The operation illustrated in this example is to push a data and than pop it from stack. The results are shown below. Under any circumstances,  $Pr$  always point to the data that was most recently pushed into the stack.

Pr	R1
5	
ST	
ST1	1 1 1 1 R2
ST2	2 2 2 2 R3
ST3	3 3 3 3 R4
ST4	4 4 4 4 R5
ST5	5 5 5 5 R6
ST6	
ST7	
ST8	
ST9	
ST10	

OW  
x x x R20  
↑  
OW unchanged

After push(X1=1, X0 from 0→1)

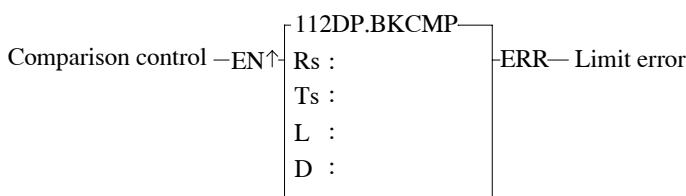
Pr	
4	
QU	
ST1	1 1 1 1 R2
ST2	2 2 2 2 R3
ST3	3 3 3 3 R4
ST4	4 4 4 4 R5
ST5	
ST6	
ST7	
ST8	
ST9	
ST10	

OW  
5 5 5 5 R20

After pop up(X1=0, X0 from 0→1)

Table instructions

FUN112 <b>D P</b> BKCMP	BLOCK COMPARE ( DRUM )	FUN112 <b>D P</b> BKCMP
----------------------------	------------------------	----------------------------



Rs : Data for compare, can be a constant or a register

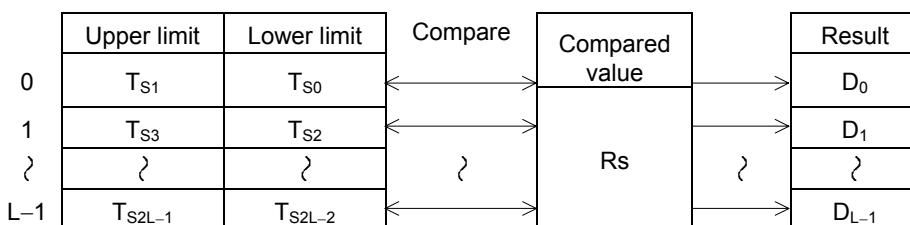
Ts : Starting register block storing upper and lower limit

L : Number of pairs of upper and lower limits

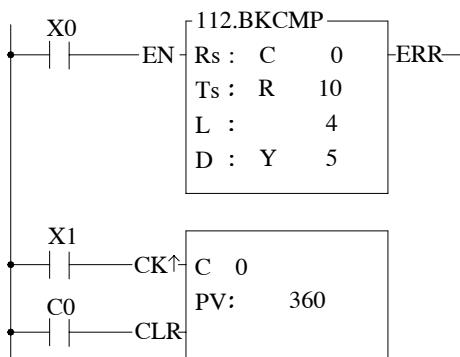
D : Starting relay storing results of comparison

Range Oper- and	Y	M	S	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	Y0   Y255	M0   M999	S0   S999	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	16/32-bit +/- number
Rs				○	○	○	○	○	○	○	○	○	○	○	○	○
Ts				○	○	○	○	○	○	○	○	○	○	○	○	○
L										○				○*	○	1~256
D	○	○	○													

- When comparison control "EN" = 1 or "EN ↑" (**P** instruction) has a transition from 0 to 1, comparisons will be performed one by one between the contents of Rs and the upper and lower limits form by L pairs of 16 or 32-bit (**D** modifier) registers starting from the Ts register (starting from T0 each adjoining 2 register units form a pair of upper and lower limits). If the value of Rs falls within the range of the pair, then the bit within the comparison results relay D which corresponds to that pair will be set to 1. Otherwise it will be set as 0 until comparison of all the L pairs of upper and lower limits is completed.
- When M1975=0, if there is any pair where the upper limit value is less than the lower limit value, then the limit error flag "ERR" will be set to 1, and the comparison output for that pair will be 0.
- When M1975=1, there is no restriction on the relation of upper limit and lower limit, this can apply for 360 ° rotary electronic drum switch application.



- Actually this instruction is a drum switch, which can be used in interrupt program and when incorporated with immediate I/O instruction (IMDIO) can achieve an accurate electronic drum.



- In this program, C0 represents the rotation angle (Rs) of a drum shaft. The block compare instruction performs a comparison between Rs and the 4 pairs (L = 4) of upper and lower limits, R10,R11, R12,R13, R14,R15 and R16,R17. The comparison results can be obtained from the four drum output points Y5 to Y8.
- The input point X1 is a rotation angle detector mounted on the drum shaft. With each one degree rotation of the drum shaft angle, X1 produces a pulse. When the drum shaft rotates a full cycle, X1 produces 360 pulses.

FUN112 <b>D P</b> BKCMP	BLOCK COMPARE ( DRUM )	FUN112 <b>D P</b> BKCMP
----------------------------	------------------------	----------------------------

- The program in the diagram above coordinates a rotary encoder or other rotating angle detection device (directly connect to a rotating mechanism), which can form a mechanical device equivalent to the mechanical structure of an actual drum (see mechanism shown within dotted line in diagram below). While the upper and lower limits are being adjusted, you can change at will the range of the activated angle of the drum. This cannot be done with the traditional drum mechanism.

Equivalent mechanical drum emulated by above program

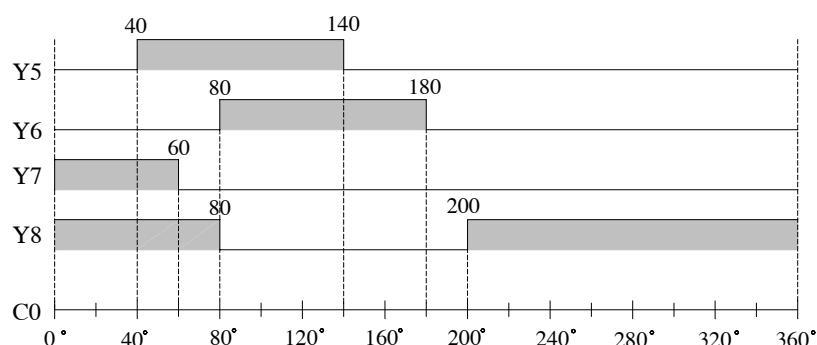
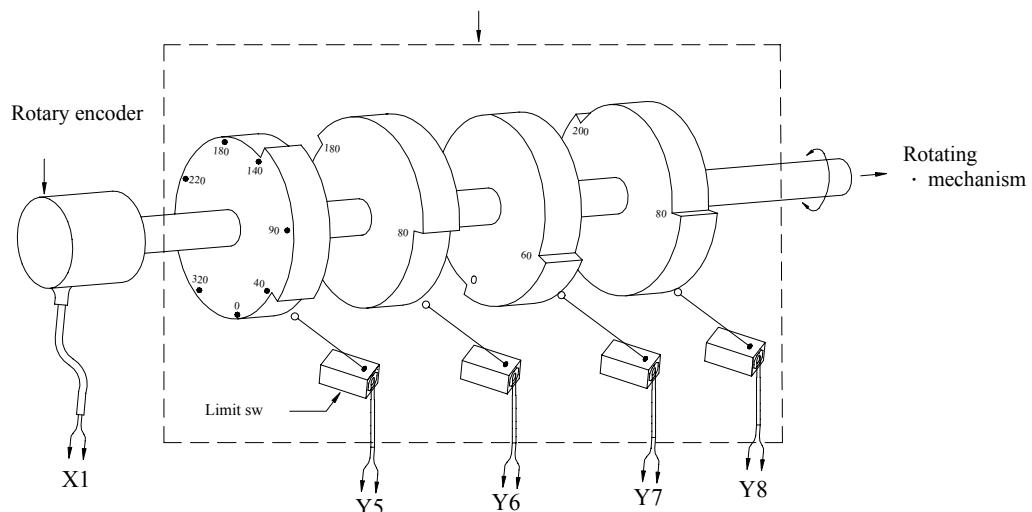


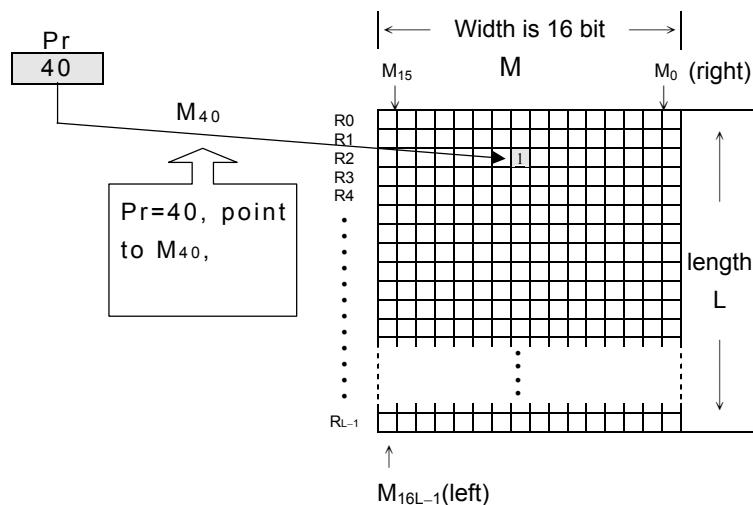
Table instructions

FUN113 DP SORT	DATA SORTING	FUN113 DP SORT																																														
	<p>-113DP.SORT-</p> <p>Sort control -EN↑ S :      ERR -</p> <p>D :     </p> <p>L :</p>	<p>S : Starting register of source registers to sort</p> <p>D : Starting register of destination registers to store the data after sorted</p> <p>L : Total register for sorting</p>																																														
	<table border="1"> <thead> <tr> <th rowspan="2">Range Oper- and</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> </tr> <tr> <th>T0 T255</th> <th>C0 C255</th> <th>R0 R3839</th> <th>R3840 R3903</th> <th>R3904 R3967</th> <th>R3968 R4167</th> <th>R5000 R8071</th> <th>D0 D3071</th> <th>2 127</th> </tr> </thead> <tbody> <tr> <td>S</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr> <tr> <td>D</td><td></td><td></td><td></td><td>○</td><td></td><td></td><td>○*</td><td>○</td></tr> <tr> <td>L</td><td></td><td></td><td>○</td><td></td><td></td><td></td><td>○</td><td>○</td></tr> </tbody> </table>	Range Oper- and	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	T0 T255	C0 C255	R0 R3839	R3840 R3903	R3904 R3967	R3968 R4167	R5000 R8071	D0 D3071	2 127	S	○	○	○	○	○	○	○	○	D				○			○*	○	L			○				○	○	
Range Oper- and	TMR		CTR	HR	IR	OR	SR	ROR	DR	K																																						
	T0 T255	C0 C255	R0 R3839	R3840 R3903	R3904 R3967	R3968 R4167	R5000 R8071	D0 D3071	2 127																																							
S	○	○	○	○	○	○	○	○																																								
D				○			○*	○																																								
L			○				○	○																																								
	<ul style="list-style-type: none"> <li>When sort control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, will sort the registers with ascending order (if A/D = 1) or descending order (if A/D = 0) and put the sorted result to the registers starting by D register.</li> <li>The valid data length of sort operation is between 2 and 127, other length will set the "ERR" to 1 and the sort operation will not perform.</li> </ul>																																															
	<p>X0      EN↑      113DP.SORT</p> <p>A/D      S : R 0 D : R 10 L : 10</p>	<ul style="list-style-type: none"> <li>The example at left sorts the table comprised of R0~R9 and stores the sorted data to the table locate at R10~R19.</li> </ul>																																														
	<table border="1"> <thead> <tr> <th></th> <th>S</th> <th>D</th> </tr> </thead> <tbody> <tr> <td>R0</td> <td>1 5 4 7</td> <td>R10 0 0 1 3</td> </tr> <tr> <td>R1</td> <td>2 3 1 4</td> <td>R11 1 5 4 7</td> </tr> <tr> <td>R2</td> <td>7 7 2 5</td> <td>R12 1 9 2 5</td> </tr> <tr> <td>R3</td> <td>0 0 1 3</td> <td>R13 2 3 1 4</td> </tr> <tr> <td>R4</td> <td>5 2 4 7</td> <td>R14 2 7 9 6</td> </tr> <tr> <td>R5</td> <td>1 9 2 5</td> <td>R15 5 2 4 7</td> </tr> <tr> <td>R6</td> <td>6 7 4 4</td> <td>R16 5 3 1 9</td> </tr> <tr> <td>R7</td> <td>5 3 1 9</td> <td>R17 6 7 4 4</td> </tr> <tr> <td>R8</td> <td>9 7 8 8</td> <td>R18 7 7 2 5</td> </tr> <tr> <td>R9</td> <td>2 7 9 6</td> <td>R19 9 7 8 8</td> </tr> </tbody> </table> <p>X0 = ↗ ⇒</p> <p>Before      After</p>		S	D	R0	1 5 4 7	R10 0 0 1 3	R1	2 3 1 4	R11 1 5 4 7	R2	7 7 2 5	R12 1 9 2 5	R3	0 0 1 3	R13 2 3 1 4	R4	5 2 4 7	R14 2 7 9 6	R5	1 9 2 5	R15 5 2 4 7	R6	6 7 4 4	R16 5 3 1 9	R7	5 3 1 9	R17 6 7 4 4	R8	9 7 8 8	R18 7 7 2 5	R9	2 7 9 6	R19 9 7 8 8														
	S	D																																														
R0	1 5 4 7	R10 0 0 1 3																																														
R1	2 3 1 4	R11 1 5 4 7																																														
R2	7 7 2 5	R12 1 9 2 5																																														
R3	0 0 1 3	R13 2 3 1 4																																														
R4	5 2 4 7	R14 2 7 9 6																																														
R5	1 9 2 5	R15 5 2 4 7																																														
R6	6 7 4 4	R16 5 3 1 9																																														
R7	5 3 1 9	R17 6 7 4 4																																														
R8	9 7 8 8	R18 7 7 2 5																																														
R9	2 7 9 6	R19 9 7 8 8																																														

## Matrix Instructions

<b>Fun No.</b>	<b>Mnemonic</b>	<b>Functionality</b>	<b>Fun No.</b>	<b>Mnemonic</b>	<b>Functionality</b>
120	MAND	Matrix AND	126	MBRD	Matrix Bit Read
121	MOR	Matrix OR	127	MBWR	Matrix Bit Write
122	MXOR	Matrix XOR	128	MBSHF	Matrix Bit Shift
123	MXNR	Matrix XNOR	129	MBROT	Matrix Bit Rotate
124	MINV	Matrix Inverse	130	MBCNT	Matrix Bit Count
125	MCMP	Matrix Compare			

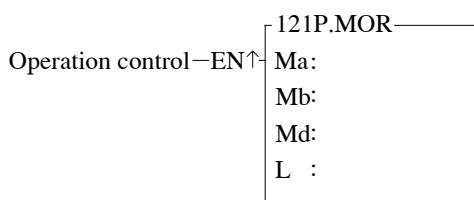
- A matrix is comprised of 2 or more consecutive 16-bit registers. The number of registers comprising the matrix is called the matrix length (L). One matrix altogether has  $L \times 16$  bits (points), and the basic unit of the object for each operation is bit.
- The matrix instructions treats the  $16 \times L$  matrix bits as a set of series points( denoted by  $M_0$  to  $M_{16L-1}$ ). Whether the matrix is formed by register or not, the operation object is the bit not numerical value.
- Matrix instructions are used mostly for discrete status processing such as moving, copying, comparing, searching, etc, of single point to multipoint (matrix), or multipoint-to-multipoint. These instructions are convenient, important for application.
- Among the matrix instructions, most instruction need to use a 16-bit register as a pointer to points a specific point within the matrix. This register is known as the matrix pointer (Pr). Its effective range is 0 to  $16L-1$ , which corresponds respectively to the bits  $M_0$  to  $M_{16L-1}$  within the matrix.
- Among the matrix operations, there are shift left/right, rotate left/right operations. We define the movement toward higher bit is left direction, while the movement toward lower bit is right direction, as shown in the diagram below.



## Matrix instructions

FUN120 P MAND	MATRIX AND	FUN120 P MAND																																																																																																											
Operation control—EN↑ <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <b>120P.MAND</b>            Ma:            Mb:            Md:            L :         </div>	Ma: Starting register of source matrix a Mb: Starting register of source matrix b Md : Starting register of destination matrix L : Length of matrix (Ma, Mb and Md) Ma, Mb, Md may combine with V, Z to serve indirect address application																																																																																																												
<p>The table shows the operational range for various parameters:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>Oper- and</td> <td>WX0 ↓ WX240</td> <td>WY0 ↓ WY240</td> <td>WM0 ↓ WM1896</td> <td>WS0 ↓ WS984</td> <td>T0 ↓ T255</td> <td>C0 ↓ C255</td> <td>R0 ↓ R3839</td> <td>R3840 ↓ R3903</td> <td>R3904 ↓ R3967</td> <td>R3968 ↓ R4167</td> <td>R5000 ↓ R8071</td> <td>D0 ↓ D3071</td> <td>2 ↓ 256</td> <td>V ↓ Z</td> </tr> <tr> <td>Ma</td> <td>○</td> </tr> <tr> <td>Mb</td> <td>○</td> </tr> <tr> <td>Md</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> </tr> <tr> <td>L</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○</td> <td></td> <td></td> <td></td> <td>○*</td> <td>○</td> <td>○</td> <td>○</td> <td></td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0 ↓ WX240	WY0 ↓ WY240	WM0 ↓ WM1896	WS0 ↓ WS984	T0 ↓ T255	C0 ↓ C255	R0 ↓ R3839	R3840 ↓ R3903	R3904 ↓ R3967	R3968 ↓ R4167	R5000 ↓ R8071	D0 ↓ D3071	2 ↓ 256	V ↓ Z	Ma	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Mb	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Md		○	○	○	○	○	○	○	○*	○*	○	○	○	○	L						○				○*	○	○	○																				
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																																															
Oper- and	WX0 ↓ WX240	WY0 ↓ WY240	WM0 ↓ WM1896	WS0 ↓ WS984	T0 ↓ T255	C0 ↓ C255	R0 ↓ R3839	R3840 ↓ R3903	R3904 ↓ R3967	R3968 ↓ R4167	R5000 ↓ R8071	D0 ↓ D3071	2 ↓ 256	V ↓ Z																																																																																															
Ma	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																															
Mb	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																																															
Md		○	○	○	○	○	○	○	○*	○*	○	○	○	○																																																																																															
L						○				○*	○	○	○																																																																																																
<ul style="list-style-type: none"> <li>When operation control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, this instruction will perform a logic AND (only if 2 bits are 1 will the result be 1, otherwise it will be 0) operation between two source matrixes with a length of L, Ma and Mb. The result will then be stored in the destination matrix Md, which is also the same length (the AND operation is done by bits with the same bit numbers). For example, if Ma<sub>0</sub> = 0, Mb<sub>0</sub> = 1, then Md<sub>0</sub> = 0; if Ma<sub>1</sub> = 1, Mb<sub>1</sub> = 1, then Md<sub>1</sub> = 1; etc, right up until AND reaches Ma<sub>16L-1</sub> and Mb<sub>16L-1</sub>.</li> </ul>	<p>The diagram shows three vertical columns of square grids representing matrices. The left column is labeled 'Ma', the middle 'Mb', and the right 'Md'. Each grid has a height labeled 'L' and a width of 16. Below the grids, an arrow labeled 'AND' points downwards, indicating the bit-wise AND operation being performed on corresponding bits of Ma and Mb to produce the result in Md.</p>																																																																																																												
<p>The logic diagram shows a contact X0 connected to the EN↑ input of the 120P.MAND block. The block also receives parameters Ma: R 0, Mb: R 10, Md: R 20, and L : 5.</p>	<ul style="list-style-type: none"> <li>In the program at left, when X0 goes from 0→1, then matrix Ma, comprised by R0 to R4, and matrix Mb, comprised by R10 to R14, will do an AND operation. The results will be stored back in matrix Md, comprised by R20 to R24. The result is shown at right in the diagram below.</li> </ul>																																																																																																												
<p>Before execution:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Ma<sub>15</sub></th> <th>Ma</th> <th>Ma<sub>0</sub></th> <th>Mb<sub>15</sub></th> <th>Mb</th> <th>Mb<sub>0</sub></th> <th>Md<sub>15</sub></th> <th>Md</th> <th>Md<sub>0</sub></th> </tr> </thead> <tbody> <tr> <td>R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> </tr> <tr> <td>R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]</td> <td>[ ]</td> <td>R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]</td> <td>R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> </tr> <tr> <td>R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> <td>R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> <td>R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> </tr> <tr> <td>R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> </tr> <tr> <td>R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> </tr> </tbody> </table> <p>Below the tables are ranges: Ma<sub>79</sub> to Ma<sub>64</sub>, Mb<sub>79</sub> to Mb<sub>64</sub>, and Md<sub>79</sub> to Md<sub>64</sub>.</p>	Ma <sub>15</sub>	Ma	Ma <sub>0</sub>	Mb <sub>15</sub>	Mb	Mb <sub>0</sub>	Md <sub>15</sub>	Md	Md <sub>0</sub>	R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]	[ ]	R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]	R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	<p>After execution:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Ma<sub>15</sub></th> <th>Ma</th> <th>Ma<sub>0</sub></th> <th>Mb<sub>15</sub></th> <th>Mb</th> <th>Mb<sub>0</sub></th> <th>Md<sub>15</sub></th> <th>Md</th> <th>Md<sub>0</sub></th> </tr> </thead> <tbody> <tr> <td>R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> </tr> <tr> <td>R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]</td> <td>[ ]</td> <td>R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]</td> <td>R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> </tr> <tr> <td>R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> <td>R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> <td>R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]</td> </tr> <tr> <td>R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td>[ ]</td> <td>R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> </tr> <tr> <td>R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td>[ ]</td> <td>R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> </tr> </tbody> </table> <p>Below the tables are ranges: Ma<sub>79</sub> to Ma<sub>64</sub>, Mb<sub>79</sub> to Mb<sub>64</sub>, and Md<sub>79</sub> to Md<sub>64</sub>.</p>	Ma <sub>15</sub>	Ma	Ma <sub>0</sub>	Mb <sub>15</sub>	Mb	Mb <sub>0</sub>	Md <sub>15</sub>	Md	Md <sub>0</sub>	R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]	[ ]	R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]	R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
Ma <sub>15</sub>	Ma	Ma <sub>0</sub>	Mb <sub>15</sub>	Mb	Mb <sub>0</sub>	Md <sub>15</sub>	Md	Md <sub>0</sub>																																																																																																					
R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]																																																																																																					
R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]	[ ]	R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]	R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]																																																																																																					
R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]																																																																																																					
R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]																																																																																																					
R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]																																																																																																					
Ma <sub>15</sub>	Ma	Ma <sub>0</sub>	Mb <sub>15</sub>	Mb	Mb <sub>0</sub>	Md <sub>15</sub>	Md	Md <sub>0</sub>																																																																																																					
R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R20 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]																																																																																																					
R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]	[ ]	R1 [1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0]	R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R11 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R21 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]																																																																																																					
R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R2 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R12 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]	[ ]	R22 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]																																																																																																					
R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]	[ ]	R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]																																																																																																					
R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]	[ ]	R24 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]																																																																																																					

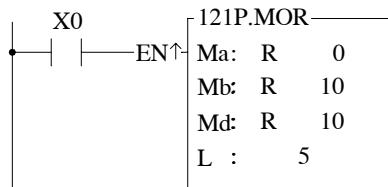
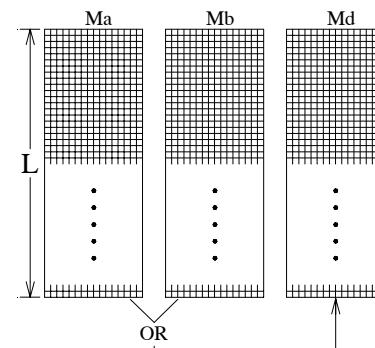
FUN121 P MOR	MATRIX OR	FUN121 P MOR
-----------------	-----------	-----------------



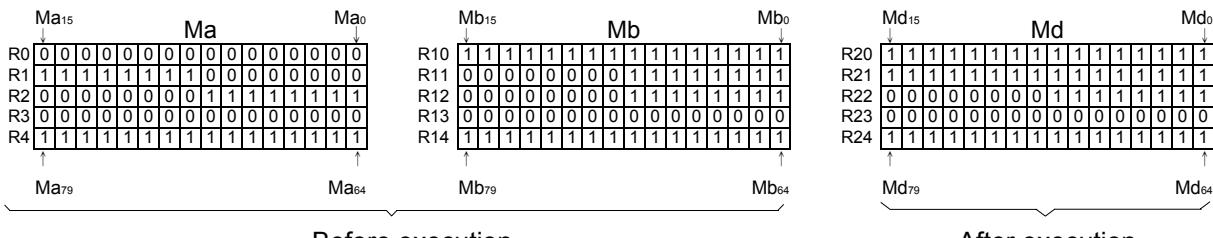
Ma : Starting register of source matrix a  
 Mb : Starting register of source matrix b  
 Md : Starting register of destination matrix  
 L : Length of matrix (Ma, Mb and Md)  
 Ma, Mb, Md may combine with V, Z to serve indirect address application

Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
Oper- and	WX0 ↓ WX240	WY0 ↓ WY240	WM0 ↓ WM1896	WS0 ↓ WS984	T0 ↓ T255	C0 ↓ C255	R0 ↓ R3839	R3840 ↓ R3903	R3904 ↓ R3967	R3968 ↓ R4167	R5000 ↓ R8071	D0 ↓ D3071	2 ↓ 256	V ↓ Z
Ma	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Mb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Md		○	○	○	○	○	○		○*	○*	○	○	○	○
L							○			○*	○	○	○	

- When operation control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, this instruction will perform a logic OR (If any 2 of the bits are 1, then the result will be 1, and only if both are 0 will the result be 0) operation between 2 source matrixes with a length of L, Ma and Mb. The result will then be stored in the destination matrix Md, which is also the same length (the OR operation is done by bits with the same bit numbers). For example, if Ma<sub>0</sub> = 0, Mb<sub>0</sub> = 1, then Md<sub>0</sub> = 1; if Ma<sub>1</sub> = 0, Mb<sub>1</sub> = 0, then Md<sub>1</sub> = 0; etc, right up until OR reaches Ma<sub>16L-1</sub> and Mb<sub>16L-1</sub>.



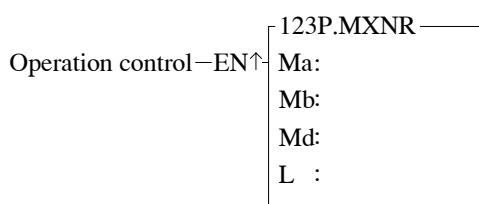
- In the program at left, when X0 goes from 0→1, then matrix Ma, comprised by R0 to R4, and matrix Mb, comprised by R10 to R14, will do an OR operation. The results will then be stored into the destination matrix Md, comprised by R10 to R14. In this example, Mb and Md is the same matrix, so after operation the source matrix Mb will replaced by the new value. The result is shown at right in the diagram below.



## Matrix instructions

FUN122 P MXOR	MATRIX EXCLUSIVE OR (XOR)	FUN122 P MXOR																																																																																									
<p>Operation control – EN↑</p> <p>122P.MXOR</p> <p>Ma: Mb: Md: L :</p>	<p>Ma: Starting register of source matrix a Mb: Starting register of source matrix b Md: Starting register of destination matrix L : Length of matrix (Ma, Mb and Md) Ma, Mb, Md may combine with V, Z to serve indirect address application</p>																																																																																										
<table border="1"> <thead> <tr> <th>Range</th><th>WX</th><th>WY</th><th>WM</th><th>WS</th><th>TMR</th><th>CTR</th><th>HR</th><th>IR</th><th>OR</th><th>SR</th><th>ROR</th><th>DR</th><th>K</th><th>XR</th></tr> <tr> <th>Oper- and</th><td>WX0 ↓ WX240</td><td>WY0 ↓ WY240</td><td>WM0 ↓ WM1896</td><td>WS0 ↓ WS984</td><td>T0 ↓ T255</td><td>C0 ↓ C255</td><td>R0 ↓ R3839</td><td>R3840 ↓ R3903</td><td>R3904 ↓ R3967</td><td>R3968 ↓ R4167</td><td>R5000 ↓ R8071</td><td>D0 ↓ D3071</td><td>2 ↓ 256</td><td>V · Z</td></tr> </thead> <tbody> <tr> <td>Ma</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr> <tr> <td>Mb</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td></tr> <tr> <td>Md</td><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td><td>○</td><td>○*</td><td>○*</td><td>○</td><td>○</td><td></td><td>○</td></tr> <tr> <td>L</td><td></td><td></td><td></td><td></td><td></td><td>○</td><td></td><td></td><td></td><td>○*</td><td>○</td><td>○</td><td></td><td>○</td></tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0 ↓ WX240	WY0 ↓ WY240	WM0 ↓ WM1896	WS0 ↓ WS984	T0 ↓ T255	C0 ↓ C255	R0 ↓ R3839	R3840 ↓ R3903	R3904 ↓ R3967	R3968 ↓ R4167	R5000 ↓ R8071	D0 ↓ D3071	2 ↓ 256	V · Z	Ma	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Mb	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Md		○	○	○	○	○		○	○*	○*	○	○		○	L						○				○*	○	○		○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																													
Oper- and	WX0 ↓ WX240	WY0 ↓ WY240	WM0 ↓ WM1896	WS0 ↓ WS984	T0 ↓ T255	C0 ↓ C255	R0 ↓ R3839	R3840 ↓ R3903	R3904 ↓ R3967	R3968 ↓ R4167	R5000 ↓ R8071	D0 ↓ D3071	2 ↓ 256	V · Z																																																																													
Ma	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																													
Mb	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																													
Md		○	○	○	○	○		○	○*	○*	○	○		○																																																																													
L						○				○*	○	○		○																																																																													
<ul style="list-style-type: none"> <li>When operation control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, this instruction will performs a logic XOR (if the 2 bits are different, then the result will be 1, otherwise it will be 0)between 2 source matrixes with a length of L, Ma and Mb. The result will then be stored back into the destination matrix Md, which also has a length of L. For example the XOR operation is done by bits with the same bit numbers - for example, if Ma<sub>0</sub> = 0, Mb<sub>0</sub> = 1, then Md<sub>0</sub> = 1; if Ma<sub>1</sub> = 1, Mb<sub>1</sub> = 1, then Md<sub>1</sub> = 0; etc, right up until XOR reaches Ma<sub>16L-1</sub> and Mb<sub>16L-1</sub>.</li> </ul>																																																																																											
<p>X0</p> <p>EN↑</p> <p>122P.MXOR</p> <p>Ma: R 0 Mb: R 10 Md: R 20 L : 5</p>	<ul style="list-style-type: none"> <li>In the program at left, when X0 goes from 0→1, will perform a XOR operation between matrix Ma, comprised by R0 to R4, and matrix Mb, comprised by R10 to R14. The results will then be stored in destination matrix Md, comprised by R20 to R24. The results are shown at right in the diagram below.</li> </ul>																																																																																										
<p>Ma<sub>15</sub>      Ma      Ma<sub>0</sub></p> <p>R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] R1 [1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0] R2 [0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1] R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</p> <p>Ma<sub>79</sub>      Ma<sub>64</sub></p> <p>Before execution</p>	<p>Mb<sub>15</sub>      Mb      Mb<sub>0</sub></p> <p>R10 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1] R11 [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1] R12 [0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1] R13 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] R14 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</p> <p>Mb<sub>79</sub>      Mb<sub>64</sub></p> <p>After execution</p>	<p>Md<sub>15</sub>      Md      Md<sub>0</sub></p> <p>R20 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1] R21 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1] R22 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] R23 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] R24 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</p> <p>Md<sub>79</sub>      Md<sub>64</sub></p>																																																																																									

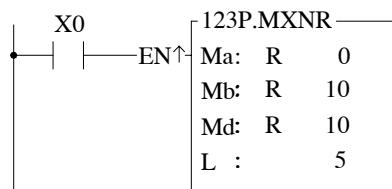
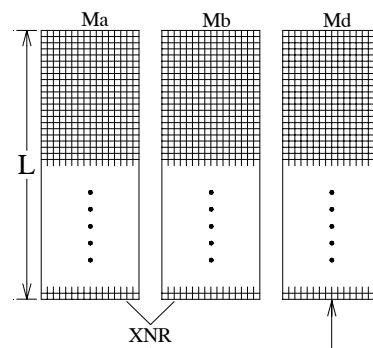
FUN123 P MXNR	MATRIX ENCLUSIVE OR ( XNR )	FUN123 P MXNR
------------------	-----------------------------	------------------



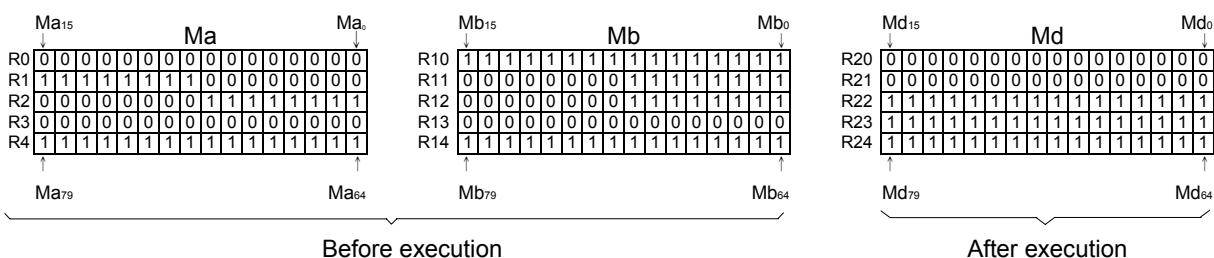
Ma : Starting register of source matrix a  
 Mb : Starting register of source matrix b  
 Md : Starting register of destination matrix  
 L : Length of matrix (Ma, Mb and Md)  
 Ma, Mb, Md may combine with V, Z to serve indirect address application

Oper- and	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
		WX0 ↓ WX240	WY0 ↓ WY240	WM0 ↓ WM1896	WS0 ↓ WS984	T0 ↓ T255	C0 ↓ C255	R0 ↓ R3839	R3840 ↓ R3903	R3904 ↓ R3967	R3968 ↓ R4167	R5000 ↓ R8071	D0 ↓ D3071	2 ↓ 256	V ↓ Z
		Ma	○	○	○	○	○	○	○	○	○	○	○	○	○
Ma		○	○	○	○	○	○	○	○	○	○	○	○	○	○
Mb		○	○	○	○	○	○	○	○	○	○	○	○	○	○
Md			○	○	○	○	○	○	○	○*	○*	○	○	○	○
L							○				○*	○	○	○	

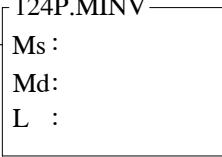
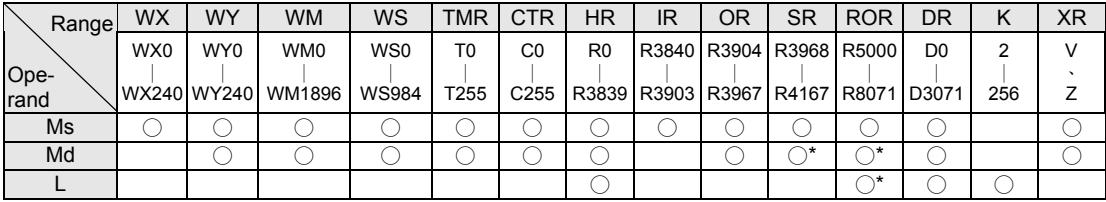
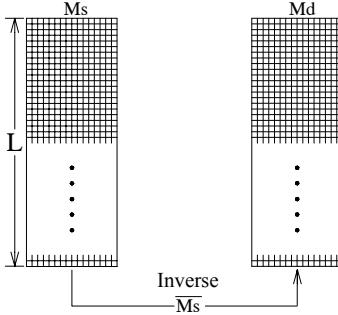
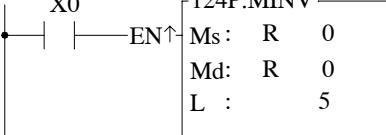
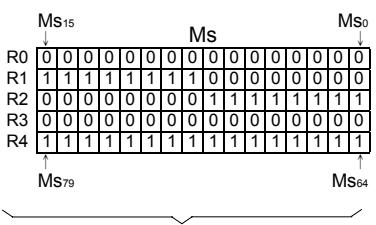
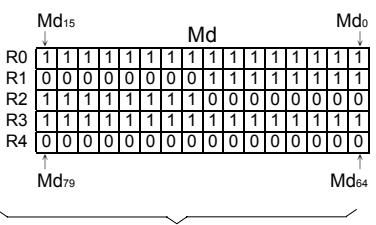
- When operation control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, will perform a logic XNR operation (if the 2 bits are the same, then the result will be 1, otherwise it will be 0) between 2 source matrixes with a length of L, Ma and Mb. The results will then be stored into the destination matrix Md, which also has the same length (the XNR operation is done by bits with the same bit numbers). For example, if Ma<sub>0</sub> = 0, Mb<sub>0</sub> = 1, then Md<sub>0</sub> = 0; Ma<sub>1</sub> = 0, Mb<sub>1</sub> = 0, then Md<sub>1</sub> = 1; etc, right up until XNR reaches Ma<sub>16L-1</sub> and Mb<sub>16L-1</sub>.



- When operation control "EN" = 1 or "EN ↑" (P instruction) goes from 0 to 1, will perform a XNR operation between Ma matrix comprised by R0~R9 and Mb matrix comprised by R10~R19. The results will then be stored into the destination matrix Md comprised by R10~R19. The results are shown at right in the diagram below.

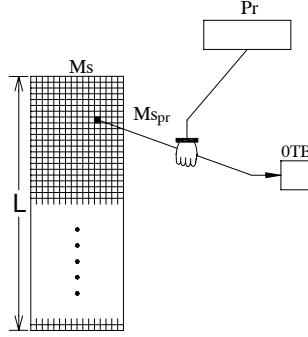
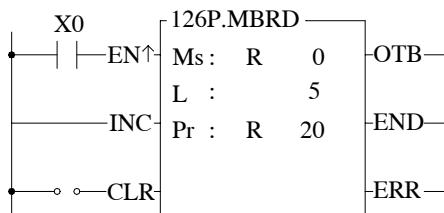
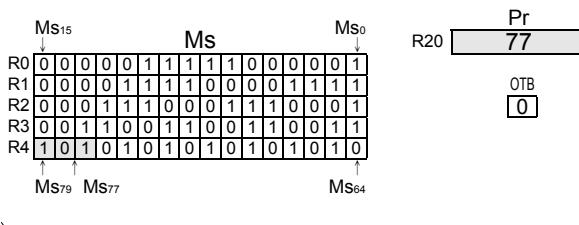
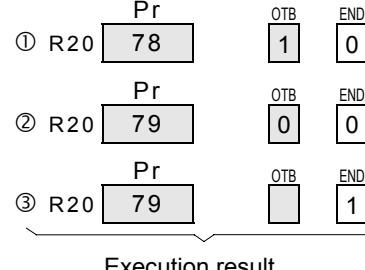


## Matrix instructions

FUN124 P MINV	MATRIX INVERSE	FUN124 P MINV
Operation control—EN↑ 	<b>124P.MINV</b> Ms : _____ Md : _____ L : _____	Ms : Starting register of source matrix Md : Starting register of destination L : Length of matrix (Ms and Md) Ma, Md may combine with V, Z to serve indirect address application
		
<ul style="list-style-type: none"> <li>When operation control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, source register Ms, which has a length of L, will be completely inverted (all the bits with a value of 1 will change to 0, and all those with a value of 0 will change to 1). The results will then be stored into destination matrix Md.</li> </ul>		
	<b>124P.MINV</b> Ms : R 0 Md : R 0 L : 5	<ul style="list-style-type: none"> <li>In the program at left, when X0 goes from 0→1, the matrix comprised by R0 to R4 will be inverted, and then store back into itself (because in this example Ms and Md are the same matrix). The results obtained are shown at right in the diagram below.</li> </ul>
		Before execution                      After execution

FUN125 P MCMP	MATRIX COMPARE	FUN125 P MCMP																																																																																										
Comparison control -EN↑	125P.MCNP																																																																																											
Compare from head -FHD	Ma: Mb: L : Pr :	-FND—Found objective -END—Compare to end -ERR—Pointer error																																																																																										
Different/Same option -D/S		Md: Starting register of matrix a Mb: Starting register of matrix b L : Length of matrix (Ma, Mb) Pr : Pointer register <i>Ma, Mb may combine with V, Z to serve indirect address application</i>																																																																																										
	<table border="1"> <thead> <tr> <th>Range</th><th>WX</th><th>WY</th><th>WM</th><th>WS</th><th>TMR</th><th>CTR</th><th>HR</th><th>IR</th><th>OR</th><th>SR</th><th>ROR</th><th>DR</th><th>K</th><th>XR</th></tr> </thead> <tbody> <tr> <td>Oper- and</td><td>WX0 WX240</td><td>WY0 WY240</td><td>WM0 WM1896</td><td>WS0 WS984</td><td>T0 T255</td><td>C0 C255</td><td>R0 R3839</td><td>R3840 R3903</td><td>R3904 R3967</td><td>R3968 R4167</td><td>R5000 R8071</td><td>D0 D3071</td><td>2 256</td><td>V Z</td></tr> <tr> <td>Ma</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td><td>○</td></tr> <tr> <td>Mb</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td></td><td>○</td></tr> <tr> <td>L</td><td></td><td></td><td></td><td></td><td></td><td></td><td>○</td><td></td><td></td><td></td><td>○*</td><td>○</td><td>○</td><td></td></tr> <tr> <td>Pr</td><td></td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○</td><td>○*</td><td>○*</td><td>○</td><td>○</td><td></td><td></td></tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0 WX240	WY0 WY240	WM0 WM1896	WS0 WS984	T0 T255	C0 C255	R0 R3839	R3840 R3903	R3904 R3967	R3968 R4167	R5000 R8071	D0 D3071	2 256	V Z	Ma	○	○	○	○	○	○	○	○	○	○	○	○		○	Mb	○	○	○	○	○	○	○	○	○	○	○	○		○	L							○				○*	○	○		Pr		○	○	○	○	○	○	○	○*	○*	○	○			
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																														
Oper- and	WX0 WX240	WY0 WY240	WM0 WM1896	WS0 WS984	T0 T255	C0 C255	R0 R3839	R3840 R3903	R3904 R3967	R3968 R4167	R5000 R8071	D0 D3071	2 256	V Z																																																																														
Ma	○	○	○	○	○	○	○	○	○	○	○	○		○																																																																														
Mb	○	○	○	○	○	○	○	○	○	○	○	○		○																																																																														
L							○				○*	○	○																																																																															
Pr		○	○	○	○	○	○	○	○*	○*	○	○																																																																																
● When comparison control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, then beginning from the top pair of bits (Ma <sub>0</sub> and Mb <sub>0</sub> ) within the 2 matrixes Ma and Mb (when "FHD" = 1 or Pr value is equal to 16L-1), or beginning from the next pair of bits (Ma <sub>pr + 1</sub> and Mb <sub>pr + 1</sub> ) pointed by pointer Pr (when "FHD" = 0 and Pr value is less than L-1), this instruction will compare and search for pairs of bits with different value (when D/S = 1) or the same value (when D/S = 0). Once match found, pointer Pr will point to the bit number in the matrix met the search condition. The found objective flag "FND" will be set to 1. When it has searched to the final pair of bits in the matrix (Ma <sub>16L-1</sub> , Mb <sub>16L-1</sub> ), this execution of the instruction will finish, no matter it has found or not. If this happen then The compare-to-end flag "END" will be set as 1, and the Pr value will set to 16L-1 and the next time that this instruction is executed, Pr will automatically return to the starting point of the matrix (Pr = 0) to begin the comparison search.																																																																																												
● The range for the pointer value is 0 to 16L-1. The Pr value should not be changed by other instructions, as this will affect the result of search. If the Pr value exceeds its range, then the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.																																																																																												
	125P.MCNP Ma: R 0 -FND— Mb: R 10 -END— L : 5 Pr : R 20 -ERR—	● In the program at left, the "FHD" input is 0, so starting from a position 1 greater than the pointer value at that time (marked by *), the instruction will do a search for bits with different status (because D/S = 1). When X0 has a transition from 0→1 three times, the results are shown at right in the diagram below.																																																																																										
	Before execution	Execution result																																																																																										

## Matrix instructions

FUN126 P MBRD	MATRIX BIT READ	FUN126 P MBRD																																																																											
	<p>126P.MBRD</p> <p>Readout control—EN↑ : Ms : OTB—Output bit      Pointer increment—INC : L : END—Read to end      Pointer clear—CLR : Pr : ERR—Pointer error</p>	<p>Ms : Starting register of matrix      L : Matrix length      Pr : Pointer register      Ms may combine with V, Z to serve indirect address application</p>																																																																											
	<table border="1"> <thead> <tr> <th>Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>Operand</td> <td>WX0 ↓ WX240</td> <td>WY0 ↓ WY240</td> <td>WM0 ↓ WM1896</td> <td>WS0 ↓ WS984</td> <td>T255</td> <td>C0 ↓ C199</td> <td>R0 ↓ R3839</td> <td>R3840 ↓ R3903</td> <td>R3904 ↓ R3967</td> <td>R3968 ↓ R4167</td> <td>R5000 ↓ R8071</td> <td>D0 ↓ D3071</td> <td>2 ↓ 256</td> <td>V ↓ Z</td> </tr> <tr> <td>Ms</td> <td>○</td> </tr> <tr> <td>L</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○</td> <td></td> <td></td> <td>○*</td> <td>○</td> <td>○</td> <td></td> <td>○</td> </tr> <tr> <td>Pr</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○</td> <td>○*</td> <td>○</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Operand	WX0 ↓ WX240	WY0 ↓ WY240	WM0 ↓ WM1896	WS0 ↓ WS984	T255	C0 ↓ C199	R0 ↓ R3839	R3840 ↓ R3903	R3904 ↓ R3967	R3968 ↓ R4167	R5000 ↓ R8071	D0 ↓ D3071	2 ↓ 256	V ↓ Z	Ms	○	○	○	○	○	○	○	○	○	○	○	○	○	○	L							○			○*	○	○		○	Pr		○	○	○	○	○	○		○	○*	○				
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																															
Operand	WX0 ↓ WX240	WY0 ↓ WY240	WM0 ↓ WM1896	WS0 ↓ WS984	T255	C0 ↓ C199	R0 ↓ R3839	R3840 ↓ R3903	R3904 ↓ R3967	R3968 ↓ R4167	R5000 ↓ R8071	D0 ↓ D3071	2 ↓ 256	V ↓ Z																																																															
Ms	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																															
L							○			○*	○	○		○																																																															
Pr		○	○	○	○	○	○		○	○*	○																																																																		
	<ul style="list-style-type: none"> <li>When readout control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, the status of the bit Mspr pointed by pointer Pr within matrix Ms will be read out and appear at the output bit "OTB". Before the readout, this instruction will first check the input -pointer clear "CLR". If "CLR" is 1, then the Pr value will be cleared to 0 first before the readout action is carried out. After the readout is completed, If the Pr value has already reached 16L-1 (the final bit), then the read-to-end flag "END" will be set to 1. If Pr is less than 16L-1, then the status of pointer increment "INC" will be checked. If "INC" is 1, then Pr will be increased by 1. Besides this, pointer clear "CLR" can execute independently, and is not affected by other input.</li> </ul>																																																																												
	<ul style="list-style-type: none"> <li>The effective range of the pointer is 0 to 16L-1. Beyond this range the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.</li> </ul>																																																																												
	<p>● In the program at left, INC = 1, so every time there is one readout the pointer will be increased by 1. With this way each bit in Ms may be read out successively, as shown at left in the diagram below. When X0 goes 3 times from 0→1, the results are shown at right in the diagram below .</p>																																																																												
																																																																													

FUN127 P MBWR	MATRIX BIT WRITE	FUN127 P MBWR																																																																	
<p>127P.MBWR</p> <p>Write control -EN↑</p> <p>Write-in bit -INB</p> <p>pointer increment -INC</p> <p>Pointer clear -CLR-</p>	<p>END— Write to end</p> <p>ERR— Pointer error</p> <p>Md : Starting register of matrix</p> <p>L : Matrix length</p> <p>Pr : Pointer register</p> <p>Md may combine with V, Z to serve indirect address application</p>																																																																		
	<table border="1"> <thead> <tr> <th>Range</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>Operand</td> <td>WY0 WY240</td> <td>WM0 WM1896</td> <td>WS0 WS984</td> <td>T0 T255</td> <td>C0 C255</td> <td>R0 R3839</td> <td>R3904 R3967</td> <td>R3968 R4167</td> <td>R5000 R8071</td> <td>D0 D3071</td> <td>2 256</td> <td>V Z</td> </tr> <tr> <td>Md</td> <td>○</td> <td></td> <td>○</td> </tr> <tr> <td>L</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○</td> <td></td> <td></td> <td>○*</td> <td>○</td> <td>○</td> <td></td> </tr> <tr> <td>Pr</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> </tbody> </table>	Range	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	XR	Operand	WY0 WY240	WM0 WM1896	WS0 WS984	T0 T255	C0 C255	R0 R3839	R3904 R3967	R3968 R4167	R5000 R8071	D0 D3071	2 256	V Z	Md	○	○	○	○	○	○	○	○	○	○		○	L						○			○*	○	○		Pr	○	○	○	○	○	○	○	○	○*	○			
Range	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	XR																																																							
Operand	WY0 WY240	WM0 WM1896	WS0 WS984	T0 T255	C0 C255	R0 R3839	R3904 R3967	R3968 R4167	R5000 R8071	D0 D3071	2 256	V Z																																																							
Md	○	○	○	○	○	○	○	○	○	○		○																																																							
L						○			○*	○	○																																																								
Pr	○	○	○	○	○	○	○	○	○*	○																																																									
<ul style="list-style-type: none"> <li>When write control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, the status of the write-in bit "INB" will be written into the bit Mdpr pointed by pointer Pr within matrix Md. Before the write-in takes place, the status of pointer clear "CLR" will be checked. If "CLR" is 1, then Pr will be cleared to 0 before the write-in action. After the write-in action has been completed, the Pr value will be checked again. If the Pr value has already reached 16L-1 (last bit), then the write-to-end flag will be set to 1. If the Pr value is less than 16L-1 and "INC" is 1, then the pointer will increase by 1. Besides this, pointer clear "CLR" can execute independently, and is not affected by other input.</li> </ul>																																																																			
<p>127P.MBWR</p> <p>X0 — EN↑</p> <p>X1 — INB</p> <p>INC</p> <p>CLR</p>	<ul style="list-style-type: none"> <li>The effective range of Pr is 0 to 16L-1. Beyond this range, the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.</li> </ul>	<p>In the program at left, pointer will be increased each time execution (because "INC" is 1). As shown in the diagram below, when X0 has a transition from 0→1, the status of INB (X1) will be written into the Mdpr (Md78) position, and pointer Pr will increase by 1 (changing to 79). In this case, although Pr is pointing to the end, it has not yet been written into Md79, so "END" flag is still 0. Only the next attempt to write to Md79 will set "END" to 1.</p>																																																																	

## Matrix instructions

FUN128 P MBSHF	MATRIX BIT SHIFT	FUN128 P MBSHF																																																																										
<p>Shift control -EN↑</p> <p>Ms :</p> <p>Md:</p> <p>Fill-in bit -INB</p> <p>Left/Right direction -L/R</p>	<p>128P.MBSHF</p> <p>OTB — Shift out bit</p>	<p>Ms : Starting register of source matrix</p> <p>Md: Starting register of destination matrix</p> <p>L : Length of matrix (Ms and Md)</p> <p>Ms, Md may combine with V, Z to serve indirect address application</p>																																																																										
<table border="1"> <thead> <tr> <th>Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>Operand</td> <td>WX0   WX240</td> <td>WY0   WY240</td> <td>WM0   WM1896</td> <td>WS0   WS984</td> <td>T0   T255</td> <td>C0   C255</td> <td>R0   R3839</td> <td>R3840   R3903</td> <td>R3904   R3967</td> <td>R3968   R4167</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td>2   256</td> <td>V   Z</td> </tr> <tr> <td>Ms</td> <td>○</td> </tr> <tr> <td>Md</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○*</td> <td>○*</td> <td>○</td> <td></td> <td>○</td> </tr> <tr> <td>L</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○*</td> <td>○</td> <td>○</td> <td></td> <td>○</td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z	Ms	○	○	○	○	○	○	○	○	○	○	○	○	○	○	Md		○	○	○	○	○	○	○	○	○*	○*	○		○	L										○*	○	○		○	
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																														
Operand	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z																																																														
Ms	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																														
Md		○	○	○	○	○	○	○	○	○*	○*	○		○																																																														
L										○*	○	○		○																																																														
<ul style="list-style-type: none"> <li>When shift control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, source matrix Ms will be retrieved and completely shifted one position to the left (when L/R = 1) or one position to the right (when L/R = 0). The space caused by the shift (with a left shift it will be M<sub>0</sub>, and with a right shift it will be M<sub>16L-1</sub>), is replaced by the status of fill-in bit "INB". The status of the bits popped out (with a left shift it will be M<sub>16L-1</sub>, and with a right shift it will be M<sub>0</sub>) will appear at the output bit "OTB". Then the results of this shifted matrix will be filled into the destination matrix Md.</li> </ul>	<p>L/R=1</p> <p>Ms</p> <p>INB</p> <p>OTB</p> <p>Md</p> <p>L</p> <p>Shift left 1 bit</p>	<p>L/R=0</p> <p>Ms</p> <p>OTB</p> <p>INB</p> <p>Md</p> <p>L</p> <p>Shift right 1 bit</p>																																																																										
<p>X0</p> <p>X0</p> <p>X0</p> <p>EN↑</p> <p>INB</p> <p>L/R</p>	<p>128P.MBSHF</p> <p>Ms : R 0</p> <p>Md: R 0</p> <p>L : 5</p>	<ul style="list-style-type: none"> <li>The program at left is an example where Ms and Md are the same matrix. When X0 goes from 0→1, Ms will be completely retrieved and moved to the left (because L/R = 1) by 1 bit. It will then be stored back to Md, and the results are shown at right in the diagram below.</li> </ul>																																																																										
<p>X1</p> <p>[1]</p>																																																																												
<p>Ms<sub>15</sub></p> <p>Ms</p> <p>Ms<sub>0</sub></p> <p>R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</p> <p>R1 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</p> <p>R2 [1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0]</p> <p>R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</p> <p>R4 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</p> <p>Ms<sub>79</sub></p> <p>Ms<sub>64</sub></p> <p>Before execution</p>	<p>OTB</p> <p>[0]</p> <p>Md<sub>15</sub></p> <p>Md</p> <p>Md<sub>0</sub></p> <p>R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]</p> <p>R1 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0]</p> <p>R2 [1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1]</p> <p>R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]</p> <p>R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0]</p> <p>Md<sub>79</sub></p> <p>Md<sub>64</sub></p> <p>After execution</p>																																																																											

FUN129 P MBROT	MATRIX BIT ROTATE	FUN129 P MBROT																																																																								
<p>Rotate control -EN↑</p> <p>Left/Right direction -L/R</p>	<p>129P.MBROT</p> <p>Ms :                   OTB— Rotated-out bit</p> <p>Md :                   </p> <p>L :                   </p>	<p>Ms : Starting register of source matrix</p> <p>Md : Starting register of destination matrix</p> <p>L : Length of matrix (Ms and Md)</p> <p>Ms, Md may combine with V, Z to serve indirect address application</p>																																																																								
	<table border="1"> <thead> <tr> <th>Range</th> <th>WX</th> <th>WY</th> <th>WM</th> <th>WS</th> <th>TMR</th> <th>CTR</th> <th>HR</th> <th>IR</th> <th>OR</th> <th>SR</th> <th>ROR</th> <th>DR</th> <th>K</th> <th>XR</th> </tr> </thead> <tbody> <tr> <td>Oper- and</td> <td>WX0   WX240</td> <td>WY0   WY240</td> <td>WM0   WM1896</td> <td>WS0   WS984</td> <td>T0   T255</td> <td>C0   C255</td> <td>R0   R3839</td> <td>R3840   R3903</td> <td>R3904   R3967</td> <td>R3968   R4167</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td>2   256</td> <td>V   Z</td> </tr> <tr> <td>Ms</td> <td>○</td> </tr> <tr> <td>Md</td> <td></td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td>○</td> <td></td> <td>○*</td> <td>○*</td> <td>○</td> <td>○</td> <td>○</td> </tr> <tr> <td>L</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>○*</td> <td>○</td> <td>○</td> <td>○</td> <td></td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z	Ms	○	○	○	○	○	○	○	○	○	○	○	○	○	Md		○	○	○	○	○	○		○*	○*	○	○	○	L									○*	○	○	○		
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																												
Oper- and	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3839	R3840   R3903	R3904   R3967	R3968   R4167	R5000   R8071	D0   D3071	2   256	V   Z																																																												
Ms	○	○	○	○	○	○	○	○	○	○	○	○	○																																																													
Md		○	○	○	○	○	○		○*	○*	○	○	○																																																													
L									○*	○	○	○																																																														
<ul style="list-style-type: none"> <li>When rotate control "EN" = 1 or "EN ↑" (P instruction) has a transition from 0 to 1, matrix Ms will be completely retrieved and rotated by one bit towards the left (when L/R = 1) or to the right (when L/R = 0). The space created by the rotation (with a left rotation it will be M0, and with a right rotation it will be M<sub>16L-1</sub>) will be replaced by the status of the rotated-out bit (with a left rotation it will be M<sub>16L-1</sub>, and with a right rotation it will be M0). The rotated-out bit will not only be used to fill the above-mentioned space, it will also be transferred to rotated-out bit "OTB".</li> </ul>																																																																										
<p>X0</p> <p>EN↑</p> <p>L/R</p>	<p>129P.MBROT</p> <p>Ms : R 0                   OTB—</p> <p>Md : R 0                   </p> <p>L : 5                   </p>	<ul style="list-style-type: none"> <li>In the program at left, Ms and Md are the same matrix. When X0 goes from 0→1, then the whole of Ms is retrieved and rotated right (because L/R = 0) by 1 bit. It is then stored back into Ms itself (because in this example Ms and Md are the same matrix). The results are shown at right in the diagram below.</li> </ul>																																																																								
	<p>Before execution</p> <table border="1"> <tr> <td>Ms<sub>15</sub></td> <td>Ms</td> <td>Ms<sub>0</sub></td> </tr> <tr> <td>R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td></td> <td></td> </tr> <tr> <td>R1 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td></td> <td></td> </tr> <tr> <td>R2 [1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0]</td> <td></td> <td></td> </tr> <tr> <td>R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td></td> <td></td> </tr> <tr> <td>R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td></td> <td></td> </tr> <tr> <td>↑ Ms<sub>79</sub></td> <td></td> <td>↑ Ms<sub>64</sub></td> </tr> </table>	Ms <sub>15</sub>	Ms	Ms <sub>0</sub>	R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]			R1 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]			R2 [1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0]			R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]			R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]			↑ Ms <sub>79</sub>		↑ Ms <sub>64</sub>	<p>After execution</p> <table border="1"> <tr> <td>Md<sub>15</sub></td> <td>Md</td> <td>Md<sub>0</sub></td> </tr> <tr> <td>R0 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td></td> <td></td> </tr> <tr> <td>R1 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td></td> <td></td> </tr> <tr> <td>R2 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td></td> <td></td> </tr> <tr> <td>R3 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]</td> <td></td> <td></td> </tr> <tr> <td>R4 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]</td> <td></td> <td></td> </tr> <tr> <td>↑ Md<sub>79</sub></td> <td></td> <td>↑ Md<sub>64</sub></td> </tr> </table> <p>OTB [0]</p>	Md <sub>15</sub>	Md	Md <sub>0</sub>	R0 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]			R1 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]			R2 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]			R3 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]			R4 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]			↑ Md <sub>79</sub>		↑ Md <sub>64</sub>																														
Ms <sub>15</sub>	Ms	Ms <sub>0</sub>																																																																								
R0 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]																																																																										
R1 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]																																																																										
R2 [1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0]																																																																										
R3 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]																																																																										
R4 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]																																																																										
↑ Ms <sub>79</sub>		↑ Ms <sub>64</sub>																																																																								
Md <sub>15</sub>	Md	Md <sub>0</sub>																																																																								
R0 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]																																																																										
R1 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]																																																																										
R2 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]																																																																										
R3 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]																																																																										
R4 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]																																																																										
↑ Md <sub>79</sub>		↑ Md <sub>64</sub>																																																																								

## Matrix instructions

FUN130 <b>P</b> MBCNT	MATRIX BIT STATUS COUNT	FUN130 <b>P</b> MBCNT																																																																																								
<p>Count control—EN↑ 1 or 0 option—1/0</p>	<p>130P.MBCNT</p> <p>Ms : Starting register of matrix L : Matrix length D : Register storing count results Ms may combine with V, Z to serve indirect address application</p>	<p>Ms : Starting register of matrix L : Matrix length D : Register storing count results Ms may combine with V, Z to serve indirect address application</p>																																																																																								
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Range</th> <th style="text-align: center; padding: 2px;">WX</th> <th style="text-align: center; padding: 2px;">WY</th> <th style="text-align: center; padding: 2px;">WM</th> <th style="text-align: center; padding: 2px;">WS</th> <th style="text-align: center; padding: 2px;">TMR</th> <th style="text-align: center; padding: 2px;">CTR</th> <th style="text-align: center; padding: 2px;">HR</th> <th style="text-align: center; padding: 2px;">IR</th> <th style="text-align: center; padding: 2px;">OR</th> <th style="text-align: center; padding: 2px;">SR</th> <th style="text-align: center; padding: 2px;">ROR</th> <th style="text-align: center; padding: 2px;">DR</th> <th style="text-align: center; padding: 2px;">K</th> <th style="text-align: center; padding: 2px;">XR</th> </tr> </thead> <tbody> <tr> <td style="text-align: left; padding: 2px;">Operand</td> <td style="text-align: center; padding: 2px;">WX0</td> <td style="text-align: center; padding: 2px;">WY0</td> <td style="text-align: center; padding: 2px;">WM0</td> <td style="text-align: center; padding: 2px;">WS0</td> <td style="text-align: center; padding: 2px;">T0</td> <td style="text-align: center; padding: 2px;">C0</td> <td style="text-align: center; padding: 2px;">R0</td> <td style="text-align: center; padding: 2px;">R3840</td> <td style="text-align: center; padding: 2px;">R3904</td> <td style="text-align: center; padding: 2px;">R3968</td> <td style="text-align: center; padding: 2px;">R5000</td> <td style="text-align: center; padding: 2px;">D0</td> <td style="text-align: center; padding: 2px;">2</td> <td style="text-align: center; padding: 2px;">V</td> </tr> <tr> <td style="text-align: left; padding: 2px;"> </td> <td style="text-align: center; padding: 2px;">WX240</td> <td style="text-align: center; padding: 2px;">WY240</td> <td style="text-align: center; padding: 2px;">WM1896</td> <td style="text-align: center; padding: 2px;">WS984</td> <td style="text-align: center; padding: 2px;">T255</td> <td style="text-align: center; padding: 2px;">C255</td> <td style="text-align: center; padding: 2px;">R3839</td> <td style="text-align: center; padding: 2px;">R3903</td> <td style="text-align: center; padding: 2px;">R3967</td> <td style="text-align: center; padding: 2px;">R4167</td> <td style="text-align: center; padding: 2px;">R8071</td> <td style="text-align: center; padding: 2px;">D3071</td> <td style="text-align: center; padding: 2px;">256</td> <td style="text-align: center; padding: 2px;">Z</td> </tr> <tr> <td style="text-align: left; padding: 2px;">Ms</td> <td style="text-align: center; padding: 2px;">○</td> </tr> <tr> <td style="text-align: left; padding: 2px;">L</td> <td style="text-align: center; padding: 2px;"> </td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;"> </td> <td style="text-align: center; padding: 2px;"> </td> <td style="text-align: center; padding: 2px;"> </td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;"> </td> </tr> <tr> <td style="text-align: left; padding: 2px;">D</td> <td style="text-align: center; padding: 2px;"> </td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;"> </td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○*</td> <td style="text-align: center; padding: 2px;">○</td> <td style="text-align: center; padding: 2px;"> </td> <td style="text-align: center; padding: 2px;"> </td> </tr> </tbody> </table>	Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	Operand	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	2	V		WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D3071	256	Z	Ms	○	○	○	○	○	○	○	○	○	○	○	○	○	○	L							○				○*	○	○		D		○	○	○	○	○	○		○	○*	○*	○		
Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR																																																																												
Operand	WX0	WY0	WM0	WS0	T0	C0	R0	R3840	R3904	R3968	R5000	D0	2	V																																																																												
	WX240	WY240	WM1896	WS984	T255	C255	R3839	R3903	R3967	R4167	R8071	D3071	256	Z																																																																												
Ms	○	○	○	○	○	○	○	○	○	○	○	○	○	○																																																																												
L							○				○*	○	○																																																																													
D		○	○	○	○	○	○		○	○*	○*	○																																																																														
<ul style="list-style-type: none"> <li>When count control "EN" = 1 or "EN ↑" (<b>P</b> instruction) has a transition from 0 to 1, then among the 16L bits of the Ms matrix, this instruction will count the total amount of bits with a status of 1 (when input "1/0" = 1) or the total amount of bits with a status of 0 (when input "1/0" = 0). The results of the counting will be stored into the register specified by D. If the value of these amounts is 0, then the Result-is-0 flag "D = 0" will be set to 1.</li> </ul> <p></p> <ul style="list-style-type: none"> <li>The program at left sets X1 first as 0 (to count bits with status of 0) and then as 1 (to count bits with status of 1) and let the signal X0 has a transition from 0→1 for both case, the execution results are shown at right in the diagram below .</li> </ul>		<p>Count of '0' bit</p> <p>Count of '1' bit</p>																																																																																								

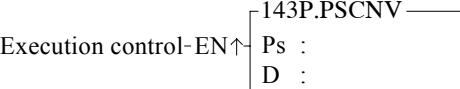
FUN140 HPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION (Brief description on function)	FUN140 HPSO																									
Execution control- EN Pause -PAU Abort -ABT	<p>140.HPSO</p> <p>Ps : The Pulse Output (0~3) selection 0:Y0 &amp; Y1 1:Y2 &amp; Y3 2:Y4 &amp; Y5 3:Y6 &amp; Y7 SR : Positioning program starting register. WR : Starting working register of instruction operation, total 7 registers, can not be used in any other part of program.</p> <table border="1"> <thead> <tr> <th>Range</th> <th>HR</th> <th>DR</th> <th>ROR</th> <th>K</th> </tr> </thead> <tbody> <tr> <td>Operand</td> <td>R0 R3839</td> <td>D0 D3071</td> <td>R5000 R8071</td> <td>2 256</td> </tr> <tr> <td>Ps</td> <td></td> <td></td> <td></td> <td>0~3</td> </tr> <tr> <td>SR</td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td></td> </tr> <tr> <td>WR</td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td><input checked="" type="radio"/></td> <td>*</td> </tr> </tbody> </table>	Range	HR	DR	ROR	K	Operand	R0 R3839	D0 D3071	R5000 R8071	2 256	Ps				0~3	SR	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		WR	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	*	
Range	HR	DR	ROR	K																							
Operand	R0 R3839	D0 D3071	R5000 R8071	2 256																							
Ps				0~3																							
SR	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																								
WR	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	*																							
<p><b>Command descriptions</b></p> <ul style="list-style-type: none"> <li>The NC positioning program of HPSO (FUN140) instruction is a program written and edited with text. The executing unit of program is divided by step (which includes output frequency, traveling distance, and transferring conditions). For one FUN140 instruction, can program 250 steps of positioning points at the most. Each step of positioning program requires 9 registers. For detailed application, please refer to chapter 14 "the NC positioning control of FB-PLC".</li> <li>The benefits of storing the positioning program in the register is that, while in application which use the MMI (man machine interface) as the operation console can save the positioning programs to MMI. Whenever the change of the positioning programs is requested, the download of positioning program can be simply done by a series of write register commands.</li> <li>The NC positioning of this instruction doesn't provide the linear interpolation function.</li> <li>When execution control "EN"=1, if Ps0~3 is not controlled by other FUN140 instruction (the status of Ps0=M1992, Ps1=M1993, Ps2=M1994, and Ps3=M1995 is ON respectively), it will start to execute from the next step of positioning point (when goes to the last step, it will be restarted from the first step); if Ps0~3 is controlled by other FUN140 instruction (the status of Ps0=M1992, Ps1=M1993, Ps2=M1994, and Ps3=M1995 are OFF), this instruction will wait and acquires the control right of output point immediately right after other FUN140 release the output.</li> <li>When execution control input "EN" =0, it stops the pulse output immediately.</li> <li>When output pause "PAU" =1 and execution control was 1, it will pause the pulse output. When output pause "PAU" =0 and execution control is still 1, it will continue the unfinished pulse output.</li> <li>When output abort "ABT"=1, it will halt and stop pulse output immediately. (When the execution control input "EN" becomes 1 next time, it will restart from the first step of positioning point to execute.)</li> <li>While send the output pulse, the output indication "ACT" is ON.</li> <li>When there is an execution error, the output indication "ERR" will be ON. (The error code is stored in the error code register.)</li> <li>When the execution of each step of positioning program is completed, the output indication "DN" will be ON.</li> </ul> <p>*** The working mode of Pulse Output must be configured (without setting, Y0~Y7 will be treated as normal output) to any one of following modes, before the HPSO instruction can be worked.</p> <p>U/D Mode: Y0 (Y2, Y4, Y6), as up pulse. Y1 (Y3, Y5, Y7), as down pulse.</p> <p>K/R Mode: Y0 (Y2, Y4, Y6), as the pulse out. Y1 (Y3, Y5, Y7), as the direction.</p> <p>A/B Mode: Y0 (Y2, Y4, Y6), as A phase pulse. Y1 (Y3, Y5, Y7), as B phase pulse.</p> <ul style="list-style-type: none"> <li>The output polarity for Pulse Output can select to be Normally ON or Normally OFF.</li> <li>The working mode of Pulse Output can be configured by PROLADDER in "HSC" setting page.</li> </ul>																											

## NC position instructions

FUN141 MPARA	NC POSITIONING PARAMETER VALUE SETTING (Brief description on function)	FUN141 MPARA																				
<p>EN</p> <p>141.MPARA</p> <p>Ps :                   ERR</p> <p>SR :</p>	<p>Ps : The pulse output (0~3) selection</p> <p>SR : Starting register for parameter table; it has 18 parameters totally, and occupy 24 registers.</p>																					
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: center;">Range</th> <th style="text-align: center;">HR</th> <th style="text-align: center;">DR</th> <th style="text-align: center;">ROR</th> <th style="text-align: center;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Oper- and</td> <td style="text-align: center;">R0   R3839</td> <td style="text-align: center;">D0   D3071</td> <td style="text-align: center;">R5000   R8071</td> <td style="text-align: center;">2   256</td> </tr> <tr> <td style="text-align: center;">Ps</td> <td></td> <td></td> <td></td> <td style="text-align: center;">0~3</td> </tr> <tr> <td style="text-align: center;">SR</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> </tbody> </table>			Range	HR	DR	ROR	K	Oper- and	R0   R3839	D0   D3071	R5000   R8071	2   256	Ps				0~3	SR	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
Range	HR	DR	ROR	K																		
Oper- and	R0   R3839	D0   D3071	R5000   R8071	2   256																		
Ps				0~3																		
SR	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																			
<p>Operation descriptions</p> <ul style="list-style-type: none"> <li>• It is not necessary to use this instruction. if the system default for parameter values is matching what user demanded, then this instruction is not needed. However, if it needs to change the parameter value dynamically, this instruction is required.</li> <li>• This instruction incorporates with FUN140 for positioning control purpose.</li> <li>• Whether the execution control input “EN” = 0 or 1, this instruction will be performed.</li> <li>• When there are any errors in parameter value, the output indication “ERR” will be ON. (The error code is stored in the error code register.)</li> <li>• For detailed functional description and usage, please refer to chapter 14 “The NC positioning control of FB-PLC” for explanation.</li> </ul>																						

FUN142 <b>P</b> PSOFF	STOP THE HPSO PULSE OUTPUT (Brief description on function)	FUN142 <b>P</b> PSOFF
	<p>Execution control-EN↑ </p> <p>Ps : 0~3 Enforce the Pulse Output PSOn (n= Ps) to stop.</p>	
<u>Command descriptions</u>		
<ul style="list-style-type: none"> <li>When execution control “EN” =1 or “EN ↑ ” (<b>P</b> instruction) changes from 0→1, this instruction will enforce the assigned number set of HPSO (High Speed Pulse Output) to stop pulse output.</li> <li>While in the application for mechanical original point reset, as soon as reach the original point can use this instruction to stop the pulse output immediately, so as to make the original point stop at the same position every time when performing mechanical original point resetting.</li> <li>For detailed functional description and usage, please refer to chapter 14 “The NC positioning control of FB-PLC” for explanation.</li> </ul>		

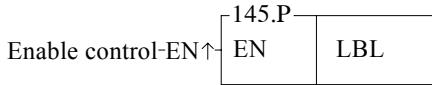
## NC position instructions

FUN143 <b>P</b> PSCNV	CONVERT THE CURRENT PULSE VALUE TO DISPLAY VALUE (mm, Deg, Inch, PS)      (Brief description on function)	FUN143 <b>P</b> PSCNV																				
	<p>Ps : 0~3; it converts the number of the pulse position to be the mm (Deg, Inch, PS) that has same unit as the set value, so as to make current position displayed.</p> <p>D : Register that stores the current position after conversion. It uses 2 registers, e.g. if D = D10, which means D10 is Low Word and D11 is High Word.</p>																					
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: center;">Range</th> <th style="text-align: center;">HR</th> <th style="text-align: center;">DR</th> <th style="text-align: center;">ROR</th> <th style="text-align: center;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Oper- rand</td> <td style="text-align: center;">R0   R3839</td> <td style="text-align: center;">D0   D3071</td> <td style="text-align: center;">R5000   R8071</td> <td style="text-align: center;">2   256</td> </tr> <tr> <td style="text-align: center;">Ps</td> <td></td> <td></td> <td></td> <td style="text-align: center;">0 ~3</td> </tr> <tr> <td style="text-align: center;">D</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> </tbody> </table>			Range	HR	DR	ROR	K	Oper- rand	R0   R3839	D0   D3071	R5000   R8071	2   256	Ps				0 ~3	D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	
Range	HR	DR	ROR	K																		
Oper- rand	R0   R3839	D0   D3071	R5000   R8071	2   256																		
Ps				0 ~3																		
D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																			

### Command descriptions

- When execution control “En” =1 or “EN ↑”(**P** instruction) changes from 0→1, this instruction will convert the assigned current pulse position (PS) to be the mm (or Deg, Inch, or PS) that has same unit as the set value, so as to make current position displaying.
- Only when the FUN140 instruction is executed, then it can get the correct conversion value by executing this instruction.
- For detailed functional description and usage, please refer to chapter 14 “The NC positioning control of FB-PLC” for explanation.

FUN145 <b>P</b> EN	ENABLE CONTROL OF THE INTERRUPT AND PERIPHERAL	FUN145 <b>P</b> EN
-----------------------	--	-----------------------



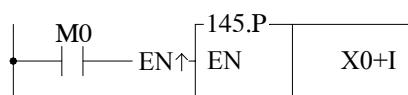
LBL : External input or peripheral label name that to be enabled.

- When enable control “EN” =1 or “EN ↑” (**P** instruction) changes from 0→1, it allows the external input or peripheral interrupt action which is assigned by LBL.
- The enabled interrupt label name is as follows:(Please refer the section 10.3 for details)

LBL name	Description	LBL name	Description	LBL name	Description
HSTA1	HSTA High speed counter interrupt	X4+I	X4 positive edge interrupt	X10+I	X10 positive edge interrupt
HSC0I	HSC0 High speed counter interrupt	X4-I	X5 negative edge interrupt	X10-I	X10 negative edge interrupt
HSC1I	HSC1 High speed counter interrupt	X5+I	X5 positive edge interrupt	X11+I	X11 positive edge interrupt
HSC2I	HSC2 High speed counter interrupt	X5-I	X5 negative edge interrupt	X11-I	X11 negative edge interrupt
HSC3I	HSC3 High speed counter interrupt	X6+I	X6 positive edge interrupt	X12+I	X12 positive edge interrupt
X0+I	X0 positive edge interrupt	X6-I	X6 negative edge interrupt	X12-I	X12 negative edge interrupt
X0-I	X0 negative edge interrupt	X7+I	X7 positive edge interrupt	X13+I	X13 positive edge interrupt
X1+I	X1 positive edge interrupt	X7-I	X7 negative edge interrupt	X13-I	X13 negative edge interrupt
X1-I	X1 negative edge interrupt	X8+I	X8 positive edge interrupt	X14+I	X14 positive edge interrupt
X2+I	X2 positive edge interrupt	X8-I	X8 negative edge interrupt	X14-I	X14 negative edge interrupt
X2-I	X2 negative edge interrupt	X9+I	X9 positive edge interrupt	X15+I	X15 positive edge interrupt
X3+I	X3 positive edge interrupt	X9-I	X9 negative edge interrupt	X15-I	X15 negative edge interrupt
X3-I	X3 negative edge interrupt				

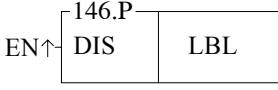
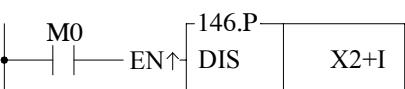
- In practical application, some interrupt signals should not be allowed to work at sometimes, however, it should be allowed to work at some other times. Employing FUN146 (DIS) and FUN145 (EN) instructions could attain the above mentioned demand.

#### Program example



- When M0 changes from 0→1, it allows X0 to send interrupt when X0 changes from 0→1. CPU can rapidly process the interrupt service program of X0+I.

Interrupt control instructions

FUN146 P DIS	DISABLE CONTROL OF THE INTERRUPT AND PERIPHERAL		FUN146 P DIS																																																																																				
																																																																																							
	<p>LBL : Interrupt label intended to disable or peripheral name to be disabled.</p>																																																																																						
	<ul style="list-style-type: none"> <li>When prohibit control “EN” =1 or “EN ↑” (P instruction) changes from 0→1, it disable the interrupt or peripheral operation designated by LBL.</li> <li>The interrupt label name is as follows:</li> </ul>																																																																																						
	<table border="1"> <thead> <tr> <th>LBL name</th> <th>Description</th> <th>LBL name</th> <th>Description</th> <th>LBL name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>HSTAI</td> <td>HSTA High speed counter interrupt</td> <td>X4+I</td> <td>X4 positive edge interrupt</td> <td>X10+I</td> <td>X10 positive edge interrupt</td> </tr> <tr> <td>HSC0I</td> <td>HSC0 High speed counter interrupt</td> <td>X4-I</td> <td>X5 negative edge interrupt</td> <td>X10-I</td> <td>X10 negative edge interrupt</td> </tr> <tr> <td>HSC1I</td> <td>HSC1 High speed counter interrupt</td> <td>X5+I</td> <td>X5 positive edge interrupt</td> <td>X11+I</td> <td>X11 positive edge interrupt</td> </tr> <tr> <td>HSC2I</td> <td>HSC2 High speed counter interrupt</td> <td>X5-I</td> <td>X5 negative edge interrupt</td> <td>X11-I</td> <td>X11 negative edge interrupt</td> </tr> <tr> <td>HSC3I</td> <td>HSC3 High speed counter interrupt</td> <td>X6+I</td> <td>X6 positive edge interrupt</td> <td>X12+I</td> <td>X12 positive edge interrupt</td> </tr> <tr> <td>X0+I</td> <td>X0 positive edge interrupt</td> <td>X6-I</td> <td>X6 negative edge interrupt</td> <td>X12-I</td> <td>X12 negative edge interrupt</td> </tr> <tr> <td>X0-I</td> <td>X0 negative edge interrupt</td> <td>X7+I</td> <td>X7 positive edge interrupt</td> <td>X13+I</td> <td>X13 positive edge interrupt</td> </tr> <tr> <td>X1+I</td> <td>X1 positive edge interrupt</td> <td>X7-I</td> <td>X7 negative edge interrupt</td> <td>X13-I</td> <td>X13 negative edge interrupt</td> </tr> <tr> <td>X1-I</td> <td>X1 negative edge interrupt</td> <td>X8+I</td> <td>X8 positive edge interrupt</td> <td>X14+I</td> <td>X14 positive edge interrupt</td> </tr> <tr> <td>X2+I</td> <td>X2 positive edge interrupt</td> <td>X8-I</td> <td>X8 negative edge interrupt</td> <td>X14-I</td> <td>X14 negative edge interrupt</td> </tr> <tr> <td>X2-I</td> <td>X2 negative edge interrupt</td> <td>X9+I</td> <td>X9 positive edge interrupt</td> <td>X15+I</td> <td>X15 positive edge interrupt</td> </tr> <tr> <td>X3+I</td> <td>X3 positive edge interrupt</td> <td>X9-I</td> <td>X9 negative edge interrupt</td> <td>X15-I</td> <td>X15 negative edge interrupt</td> </tr> <tr> <td>X3-I</td> <td>X3 negative edge interrupt</td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>			LBL name	Description	LBL name	Description	LBL name	Description	HSTAI	HSTA High speed counter interrupt	X4+I	X4 positive edge interrupt	X10+I	X10 positive edge interrupt	HSC0I	HSC0 High speed counter interrupt	X4-I	X5 negative edge interrupt	X10-I	X10 negative edge interrupt	HSC1I	HSC1 High speed counter interrupt	X5+I	X5 positive edge interrupt	X11+I	X11 positive edge interrupt	HSC2I	HSC2 High speed counter interrupt	X5-I	X5 negative edge interrupt	X11-I	X11 negative edge interrupt	HSC3I	HSC3 High speed counter interrupt	X6+I	X6 positive edge interrupt	X12+I	X12 positive edge interrupt	X0+I	X0 positive edge interrupt	X6-I	X6 negative edge interrupt	X12-I	X12 negative edge interrupt	X0-I	X0 negative edge interrupt	X7+I	X7 positive edge interrupt	X13+I	X13 positive edge interrupt	X1+I	X1 positive edge interrupt	X7-I	X7 negative edge interrupt	X13-I	X13 negative edge interrupt	X1-I	X1 negative edge interrupt	X8+I	X8 positive edge interrupt	X14+I	X14 positive edge interrupt	X2+I	X2 positive edge interrupt	X8-I	X8 negative edge interrupt	X14-I	X14 negative edge interrupt	X2-I	X2 negative edge interrupt	X9+I	X9 positive edge interrupt	X15+I	X15 positive edge interrupt	X3+I	X3 positive edge interrupt	X9-I	X9 negative edge interrupt	X15-I	X15 negative edge interrupt	X3-I	X3 negative edge interrupt				
LBL name	Description	LBL name	Description	LBL name	Description																																																																																		
HSTAI	HSTA High speed counter interrupt	X4+I	X4 positive edge interrupt	X10+I	X10 positive edge interrupt																																																																																		
HSC0I	HSC0 High speed counter interrupt	X4-I	X5 negative edge interrupt	X10-I	X10 negative edge interrupt																																																																																		
HSC1I	HSC1 High speed counter interrupt	X5+I	X5 positive edge interrupt	X11+I	X11 positive edge interrupt																																																																																		
HSC2I	HSC2 High speed counter interrupt	X5-I	X5 negative edge interrupt	X11-I	X11 negative edge interrupt																																																																																		
HSC3I	HSC3 High speed counter interrupt	X6+I	X6 positive edge interrupt	X12+I	X12 positive edge interrupt																																																																																		
X0+I	X0 positive edge interrupt	X6-I	X6 negative edge interrupt	X12-I	X12 negative edge interrupt																																																																																		
X0-I	X0 negative edge interrupt	X7+I	X7 positive edge interrupt	X13+I	X13 positive edge interrupt																																																																																		
X1+I	X1 positive edge interrupt	X7-I	X7 negative edge interrupt	X13-I	X13 negative edge interrupt																																																																																		
X1-I	X1 negative edge interrupt	X8+I	X8 positive edge interrupt	X14+I	X14 positive edge interrupt																																																																																		
X2+I	X2 positive edge interrupt	X8-I	X8 negative edge interrupt	X14-I	X14 negative edge interrupt																																																																																		
X2-I	X2 negative edge interrupt	X9+I	X9 positive edge interrupt	X15+I	X15 positive edge interrupt																																																																																		
X3+I	X3 positive edge interrupt	X9-I	X9 negative edge interrupt	X15-I	X15 negative edge interrupt																																																																																		
X3-I	X3 negative edge interrupt																																																																																						
	<ul style="list-style-type: none"> <li>In practical application, some interrupt signals should not be allowed to work at certain situation. To achieve this, this instruction may be used to disable the interrupt signal.</li> </ul>																																																																																						
	<p><b>Program example</b></p>																																																																																						
																																																																																							
	<ul style="list-style-type: none"> <li>When M0 changes from 0→1, it prohibits X2 from sending interrupt when X2 changes from 0→1.</li> </ul>																																																																																						

## Chapter 10 FB-PLC Interrupt Function

### 10.1 The Principle and the Structure of Interrupt Function

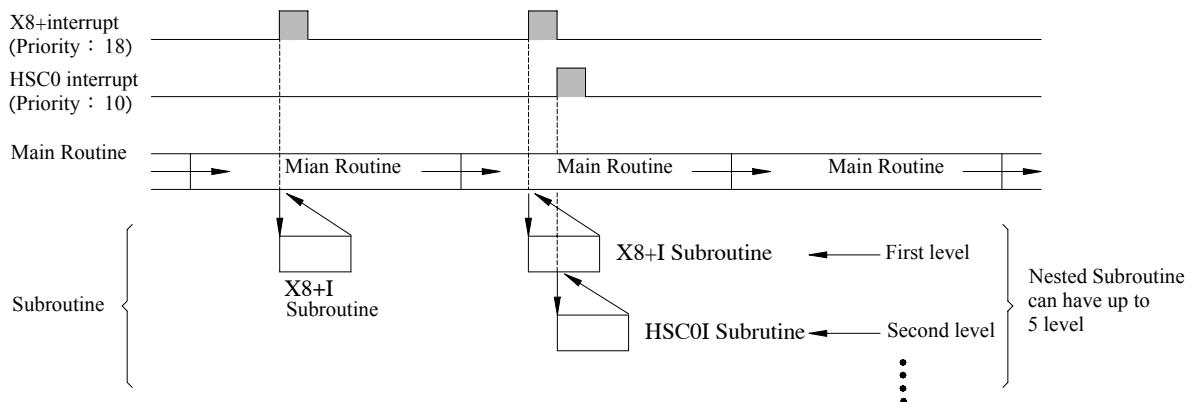
There are many jobs that FB-PLC needs to carry out. For example, there are 13K words user's program need to be solved, 512 points of I/O status need to be captured or updated, 3 communication ports need to be serviced, and etc. However, jobs can only be executed one at a time as there is merely one CPU available. Therefore, PLC service one job after another in sequence until all the jobs are executed once. Then, it will return to the first job to repeat the same cycle. The time interval of each execution is called the "scan time" of PLC. The CPU execution speed is extremely fast in comparison with human response. As far as human feeling is concerned, PLC almost completes all jobs at the same time when PLC can normally complete the foregoing huge workload within tens of milliseconds (ms). Hence, can meet the requirements of the most practical control cases.

In most application cases, the control method described above is very much sufficient. But for some applications that require a high-speed response (such as positioning control) when the required response speed is down to few micro-seconds ( $\mu$ s), a delay in scan time will certainly mean an increase in error. Under the circumstances, only applying the "Interrupt" function can achieve the precision requirement.

The so-called "Interrupt" means the interrupt request to the CPU during normal scan cycle when an immediate response is required. After receiving such request, the CPU will promptly stop all scanning work to prioritize to perform and complete the corresponding service work before return (the so-called "Return from Interrupt" or RTI) to where interrupt occurred and resume the interrupted scanning work.

The service work needed to carry out while interrupt occurred is called Interrupt Service Routine, which is a subroutine consisted by a series of ladder codes. It is placed in the subroutine area and begin with the LBL instruction with reserved label name (please refer to Section 10.3). Since it is placed in the subroutine area, it will not be executed in a normal PLC scanning cycle (PLC only constantly scans the main program area but not the subroutine area).

In normal case, the CPU can promptly execute the corresponding interrupt routine within tens of micro-seconds when an interrupt occurred. When there are more than one interrupt occurred at the same time (e.g. FB-PLC has 42 interrupts source), only the interrupt with highest priority can be executed. All the other interrupt routines need to wait until it became the highest priority among the pending interrupts. Consequently, a response delay of hundreds of microseconds, or even few milliseconds, may be caused. Hence, in a multiple interrupt inputs structure, an interrupt priority is given to each interrupt in accordance with its importance. In case another interrupt request is made when the PLC is carrying out the interrupt service routine for an interrupt request that has a higher priority than the new interrupt request, the CPU will wait until the execution of the subroutine is completed before accepting the new interrupt request. However, if the priority of the new interrupt request is higher than the one being executed, the CPU will stop the running of the current interrupt service routine immediately to execute the interrupt service routine with a higher priority. After completing the execution, the CPU will return to the previously interrupted service routine with a lower priority to continue the incomplete work. This kind of interrupt in an interrupt execution is called the "Nested Interrupt". FB-PLC can have up to 5 levels of nested interrupts. The diagram below shows the examples of single interrupts and nested interrupt:

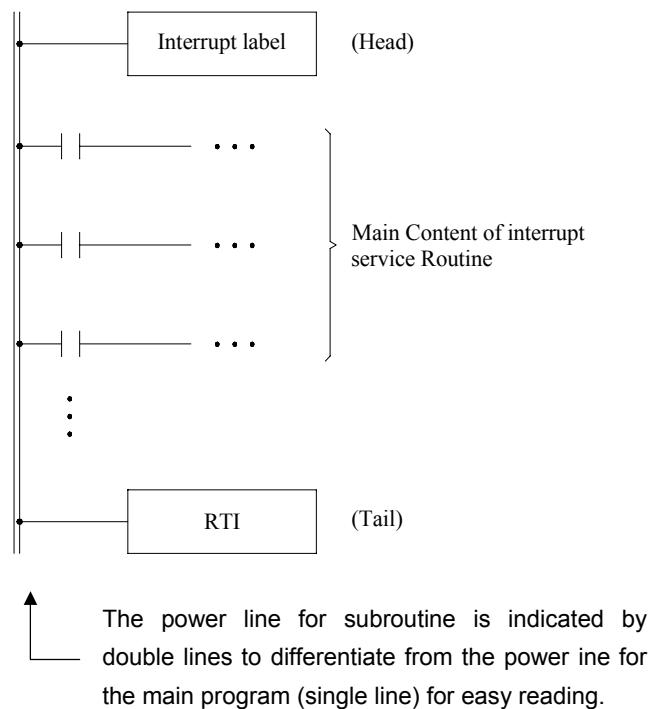


## 10.2 Structure and Application of Interrupt Service Routine

Although both “Interrupt” and “Call” are having subroutines, but the calling methods (to jump to subroutine for execution) are different. When the CALL command [FUN67] is executed by “Call” in the main program, the CPU will execute the subroutine with the label name designated by the CALL command. The CPU will return to the main program after the RTS (Return from Subroutine) command is executed.

The calling of “Interrupt” is triggered by, instead of using software commands, the hardware interrupt signal to the CPU. The CPU will identify the source of the interrupt and jump automatically to the “Interrupt Service Routine” with the label name of the interrupt in the subroutine for execution. It will return to the main program after the RTI (Return from Interrupt) command is executed. Therefore, there is no ladder code relevant to interrupt in the main program area.

As mentioned before, interrupt service routine must be placed in the sub program area. The structure is shown as the diagram on the right where a “head”, a “tail” and the main body of the service routine are included. The “head” is the “interrupt label name” of the interrupt (to be discussed in the next section). The “tail” is the RTI command [FUN69], to tell the CPU that the interrupt subroutine is ended and it should jump to the place where it was interrupted, please refer to FUN69 (RTI) instruction. In between the “head” and the “tail” is the main body of the interrupt service routine used to tell the CPU what control actions should be executed when interrupt occurs.



### 10.3 Interrupt Source, Label and Priority for FB-PLC

As described in the last section, every “Interrupt Service Routine” should have a unique “Interrupt Label”. There are 49 corresponding “Interrupt Labels” for interrupts, namely “Interrupt Reserve Words”, can be used in the sub program area of FB-PLC. These labels are dedicated to the interrupt routines hence cannot use for normal subroutine or jump target.

The “Interrupt Label” (Interrupt Reserve Word) are all suffix with an “I” letter. For examples, the interrupt label for high-speed counter HSC0 should be “HSC0I” and the interrupt label for X0+ should be “X0+I”. The “Interrupt Labels” and their priorities for the 49 FB-PLC interrupt sources of FB-PLC are shown as below.

The following table is the interrupt sources and their label names. To compatible with previous versions of programming tool, besides HSC/HST, the label names in old versions are also enlisted (label name with parenthesis). The new label names are prefer than old while in usage (HSTA1, 1MSI~100MSI, X0+I~X15-I are prior in using). When there is an interrupt with a label naming by new convention cannot work, it may be the problem of version incompatible. If this is the case please alter the interrupt label name to the labels of old versions, such as ATMRI, 1MS~100MS, INT0~INT15-. If possible, it is recommended to update the programming tool (PROLADDER or FP-07).

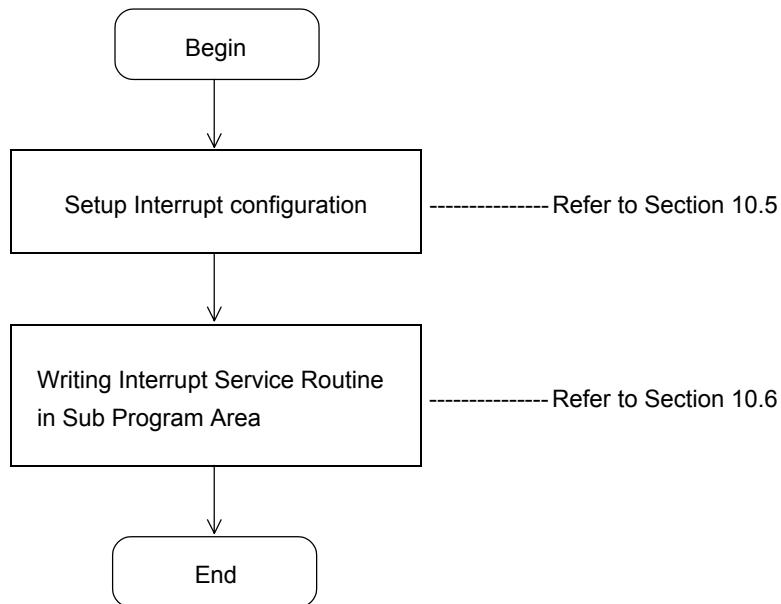
(The priority of interrupt is inversely proportional to the value of priority)

Interrupt Source	Priority	Interrupt Label	Condition for Interrupt	Note
High Speed Timer	1	HSTA1 (ATMRI)	Timing from HSTA to ( CV=PV )	No interrupt when act as a cyclic timer
Internal Time Base	2	1MSI (1MS)	One interrupt every 1mS	One kind of time base interrupt is allowed at a time (please refer to Section 10.5.2). Therefore, the actual number of interrupts is 42.
	3	2MSI (2MS)	One interrupt every 2mS	
	4	3MSI (3MS)	One interrupt every 3mS	
	5	4MSI (4MS)	One interrupt every 4mS	
	6	5MSI (5MS)	One interrupt every 5mS	
	7	10MSI (10MS)	One interrupt every 10mS	
	8	50MSI (50MS)	One interrupt every 50mS	
	9	100MSI (100MS)	One interrupt every 100mS	
HSC / HST	10	HSC0I/HST0I	Counting/Timing from HSC0/HST0 to (CV=PV)	HSC0~HSC3 are labeled as HSC0I~HSC3I when configured as high speed counter; and are labeled as HST0I~HST3I for high speed timer.
	11	HSC1I/HST1I	Counting/Timing from HSC1/HST1 to (CV=PV)	
	12	HSC2I/HST2I	Counting/Timing from HSC2/HST2 to (CV=PV)	
	13	HSC3I/HST3I	Counting/Timing from HSC3/HST3 to (CV=PV)	
PSO	14	PSO0I	Pulse output of PSO0 completed	
	15	PSO1I	Pulse output of PSO1 completed	
	16	PSO2I	Pulse output of PSO2 completed	
	17	PSO3I	Pulse output of PSO3 completed	

Interrupt Source	Priority	Interrupt Label	Condition for Interrupt	Note
Interrupt from External Hardware Input or Software High-Speed Timer	18	X0+I (INT0)	Interrupt when 0→1 (↑) of X0	The counter input and control input of the software high speed counter HSC4 ~ HSC7 which were implemented by the interrupt function can be designated as any one input of X0~X15. Therefore, the interrupt priority of the software high speed counter depends on the input it utilized.
	19	X0-I (INT0-)	Interrupt when 1→0 (↓) of X0	
	20	X1+I (INT1)	Interrupt when 0→1 (↑) of X1	
	21	X1-I (INT1-)	Interrupt when 1→0 (↓) of X1	
	22	X2+I (INT2)	Interrupt when 0→1 (↑) of X2	
	23	X2-I (INT2-)	Interrupt when 1→0 (↓) of X2	
	24	X3+I (INT3)	Interrupt when 0→1 (↑) of X3	
	25	X3-I (INT3-)	Interrupt when 1→0 (↓) of X3	
	26	X4+I (INT4)	Interrupt when 0→1 (↑) of X4	
	27	X4-I (INT4-)	Interrupt when 1→0 (↓) of X4	
	28	X5+I (INT5)	Interrupt when 0→1 (↑) of X5	
	29	X5-I (INT5-)	Interrupt when 1→0 (↓) of X5	
	30	X6+I (INT6)	Interrupt when 0→1 (↑) of X6	
	31	X6-I (INT6-)	Interrupt when 1→0 (↓) of X6	
	32	X7+I (INT7)	Interrupt when 0→1 (↑) of X7	
	33	X7-I (INT7-)	Interrupt when 1→0 (↓) of X7	
	34	X8+I (INT8)	Interrupt when 0→1 (↑) of X8	
	35	X8-I (INT8-)	Interrupt when 1→0 (↓) of X8	
	36	X9+I (INT9)	Interrupt when 0→1 (↑) of X9	
	37	X9-I (INT9-)	Interrupt when 1→0 (↓) of X9	
	38	X10+I (INT10)	Interrupt when 0→1 (↑) of X10	
	39	X10-I (INT10-)	Interrupt when 1→0 (↓) of X10	
	40	X11+I (INT11)	Interrupt when 0→1 (↑) of X11	
	41	X11-I (INT11-)	Interrupt when 1→0 (↓) of X11	
	42	X12+I (INT12)	Interrupt when 0→1 (↑) of X12	
	43	X12-I (INT12-)	Interrupt when 1→0 (↓) of X12	
	44	X13+I (INT13)	Interrupt when 0→1 (↑) of X13	
	45	X13-I (INT13-)	Interrupt when 1→0 (↓) of X13	
	46	X14+I (INT14)	Interrupt when 0→1 (↑) of X14	
	47	X14-I (INT14-)	Interrupt when 1→0 (↓) of X14	
	48	X15+I (INT15)	Interrupt when 0→1 (↑) of X15	
	49	X15-I (INT15-)	Interrupt when 1→0 (↓) of X15	

## 10.4 How to Use Interrupt of FB-PLC

The applications of interrupt in internal timing, external input, HSC/HST or PSO are similar. Since the applications of HSC/HST and PSO have been described in other chapters/sections, only examples of internal timing and external input will be described in this section.



## 10.5 Interrupt Configuration

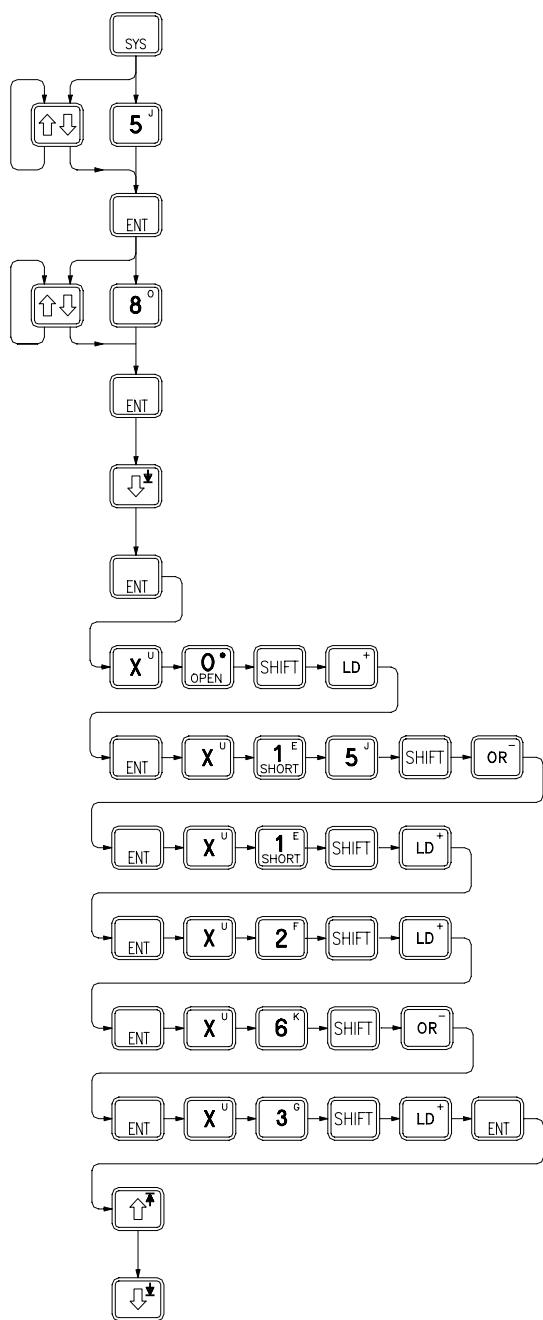
In fact, interrupt configuration is simply to determine whether the application of a certain interrupt is to be used or not.

Interrupt configuration can be divided into configuration relevant to I/O or irrelevant to I/O two categories. HSTA, HSC/HST, PSO and external interrupt are all relevant to I/O and should be performed by the configuration function of programming tool. The programming tool will automatically enable the interrupt of the device once it is configured.

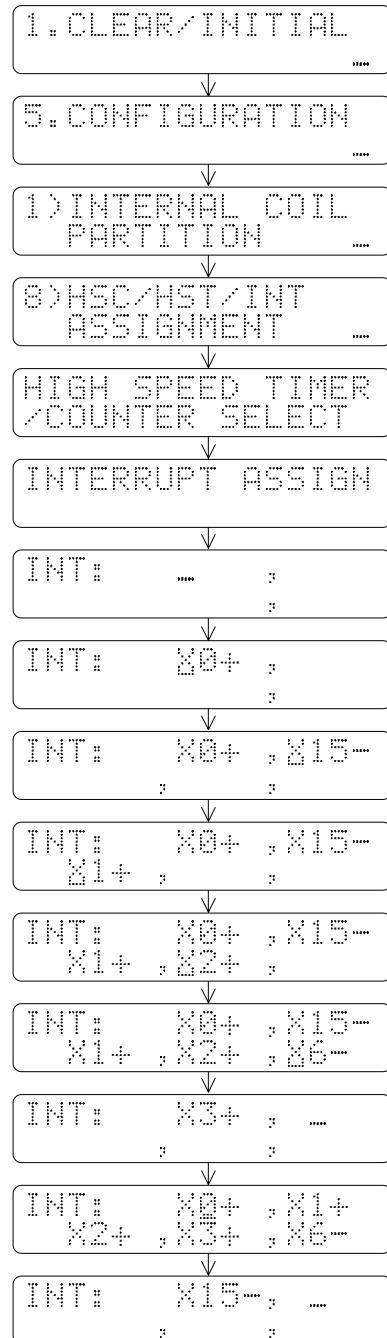
The configuration of internal time base interrupt (1MSI~100MSI), which is irrelevant to I/O, need not to be configured. As long as the time base interrupt reserved words, which is placed in front of the interrupt service subroutine, appears in the sub program area, it imply the interrupt has been planned. If more than one such interrupts appear, can use low byte, B0~B7, of the special register R4162 to control the interrupt of 1MSI~100MSI to be executed or not.

### 10.5.1 Configuration Example of Using FP-07 as an “External Interrupt”

【Keypad Operation】



【LCD Display】



- External interrupt shares the 16 high-speed input points, X0~X15, with HSC and SPD instructions. Therefore, the number of the input points used by HSC or SPD cannot configure for external interrupt.
- Note: SPD instruction can only uses X0~X7 8 input points for average speed detection.
- Once the interrupt configuration is determined, it cannot be changed in PLC RUN. But the EN command [FUN145] and DIS command [FUN146] provided by FB-PLC can dynamically enable/disable the operation of interrupt of external, HSC and HSTA in PLC RUN. Please refer to the description of the two instructions.

### 10.5.2 Internal Time Base Interrupt Configuration by R4162

When the internal time base interrupt reserved words (8 kinds, 1MSI~100MSI) appears in the sub program area, it imply that the designated interrupt has been planned and can be masked by using the 8 bits of the low byte in the register R4162 as shown in below:

R4162:	B7	B6	B5	B4	B3	B2	B1	B0
	100MS	50MS	10MS	5MS	4MS	3MS	2MS	1MS

- When bit status =0: Enable the time base interrupt (not masked)
- When bit status =1: Disable the time base interrupt (masked)

- Among B0~B7, if more than one of the bits is 0, FB-PLC will enable the one with the smallest time base and disable the others. If the content of R4162 is 00H, then all time base interrupts will not be masked. However, if 1 MS and 2MS~100MS time base interrupt subroutine are all appeared in subprogram area, only the 1MS timing interrupt will be executed, and the others will not be executed.
- It is with great flexibility since the user can dynamically change the time base or pause or enable the interrupt by using the ladder program to change the value of R4162 at any time in PLC RUN.
- The default of R4162 is 0; it represents that 1MS~100MS time base interrupt are not been masked. As long as any one of time base interrupt processing subroutine exists in the sub program area, it will be executed periodically.
- Since a considerable CPU time is required for execution of every interrupt, the smaller the interrupt time base, the more interrupts required and the longer CPU time occupied. Therefore, application should be made only when necessary to avoid degradation of CPU performance.

## 10.6 Examples of Interrupt Routine

### Example 1 Precision position control by positioning switch .

X0 : Position Sensor

X1 : Emergency Stop

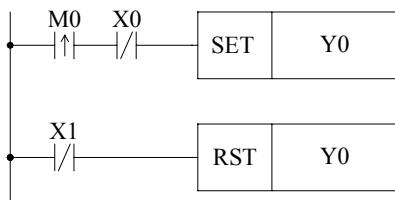
Y1 : Power motor

### 【External Interrupt Configuration】

INT# X0# ;  
; ;

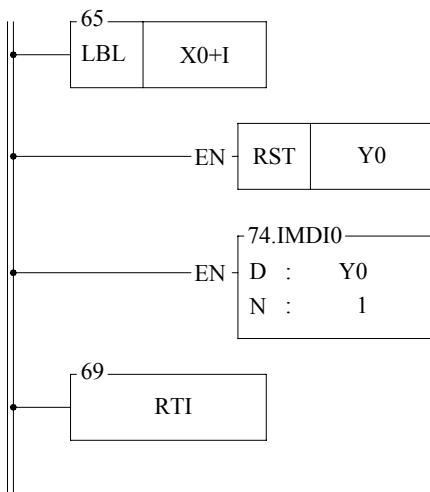
- Configure Input Interrupt on when 0→1 of X0

### 【Main program】



- M0 (start) changes from 0→1, the motor is ON.

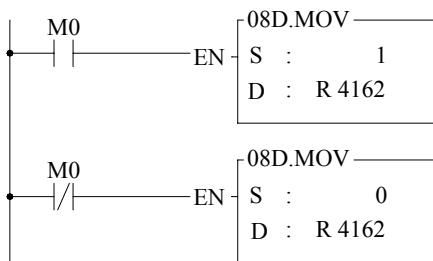
### 【Subroutine】



- When the sensor, X0, detects the arriving of positioning location, i.e. X0 change from 0→1, the hardware will automatically execute the interrupt subroutine
- As motor Y0 changes to 0, it stops the motor immediately.
- Output Y0 immediately to reduce delay caused by scan time
- It must employ immediate input/output instruction in the interrupt subroutine to meet the real time high speed precision control requirement.

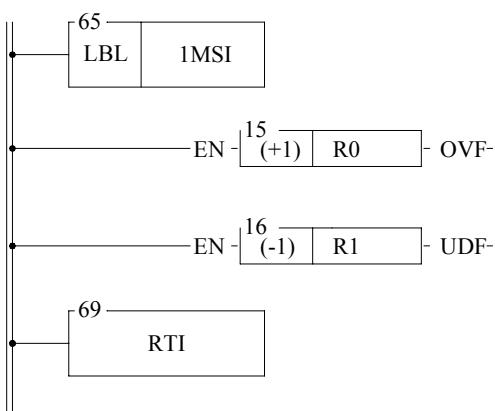
### 【Example 2】 1MS Internal Time base Interrupt

#### 【Main program】



- When M0=1, 1MS timing interrupt is disabled  
(1MS timing interrupt being masked)
- When M0=0, 1MS timing interrupt is enabled

### 【Subroutine】



- After 1MS time base interrupt is started, the system will automatically execute the interrupt subroutine every 1MS
- R0 is used as the up counting cyclic timer for every 1MS time base
- R1 is used as the down counting cyclic timer for every 1MS time base

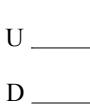
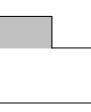
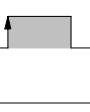
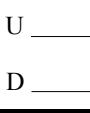
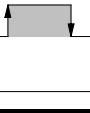
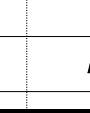
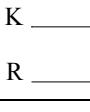
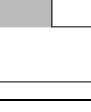
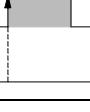
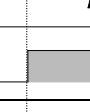
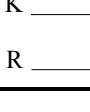
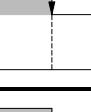
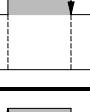
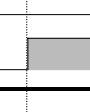
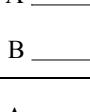
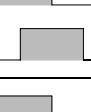
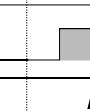
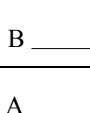
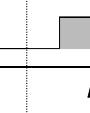
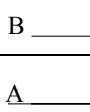
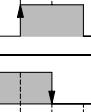
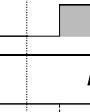
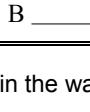
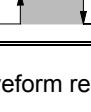
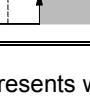
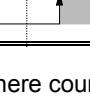
# Chapter 11 FB-PLC High-Speed Counter and Timer

## 11.1 FB-PLC High-Speed Counter

The counting frequency of an ordinary PLC's software counter can only reach tens of Hz (depending on the scan time). If the frequency of input signal is higher than that, it is necessary to utilize high-speed counter (HSC), otherwise loss count or even out of counting may occur. There are usually two types of HSC implemented for PLC. The hardware high-speed counter (HHSC) employed special hardware circuit and the software high-speed counter (SHSC) which when counting signal changes state will interrupt CPU to perform the increment/decrement counting operation. FB-PLC provides up to 4 HHSCs (in ASIC chips) and 4 SHSCs. All of them are all 32 bit high speed counter.

### 11.1.1 Counting Modes of FB-PLC High-Speed Counter

As shown in the table below, each of the four FB-PLC HHSCs and SHSCs provides 8 and 3, respectively, kind of counting modes to choose from:

Counting Mode			HHSC (HSC0~HSC3)	SHSC (HSC4~HSC7)	Counting Waveform	
Up-down pulse	MD 0	U/D	○	○	Up Counting (+1)	Down Counter (-1)
					 	 
Pulse-direction	MD 1	U/D×2	○		 	 
					 	 
AB phase	MD 3	K/R×2	○		 	 
					 	 
	MD 5	A/B×2	○		 	 
					 	 
	MD 7	A/B×4	○		 	 

- The up/down arrow (↑, ↓) on the positive/negative edge in the waveform represents where counting (+1 or -1) occurs.

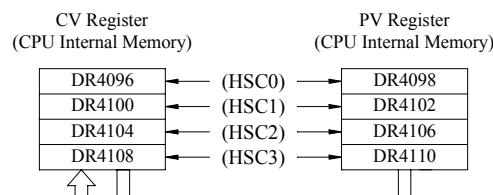
## 11.2 System Architecture of FB-PLC High-Speed Counter

The diagrams below are the system architecture for FB-PLC HHSC and SHSC where each one of them has multi-purpose input and counting functions. Some of the functions are built-in (such as CV register number, PV register number, interrupt label and relay number for software MASK, CLEAR and direction selection) that user need not to assign for configuration. However, some functions, with a "\*" marked in the diagrams below, must use the programming tool to configure the HSC (such as HSC application selection, counting mode, application of each function input, inverse polarity and appointment of corresponding input point number Xn) etc. For detailed structure and operation of the 8 kind of counting modes that assigned in configuration, please refer to section 11.2.1~11.2.3 for explanation.

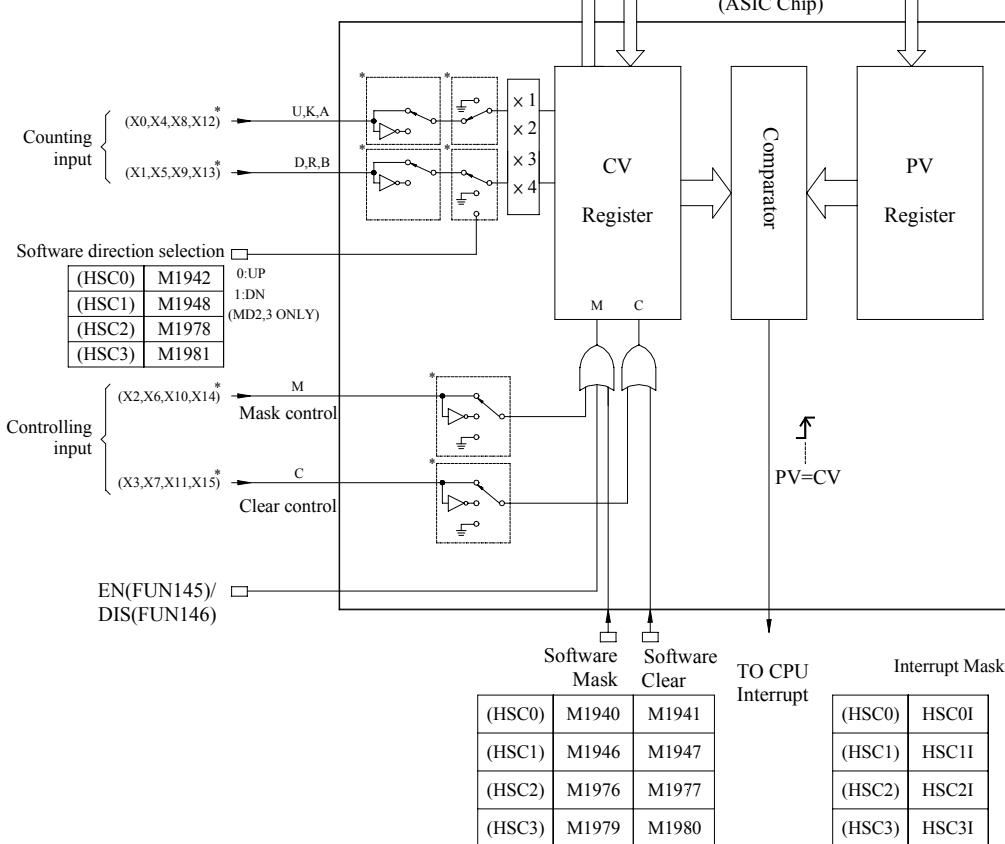
Note: CV (Current Value); PV (Preset Value).

- Use FUN92 to read out current counting value from ASIC chip hardware counter to put it into CPU internal CV register.

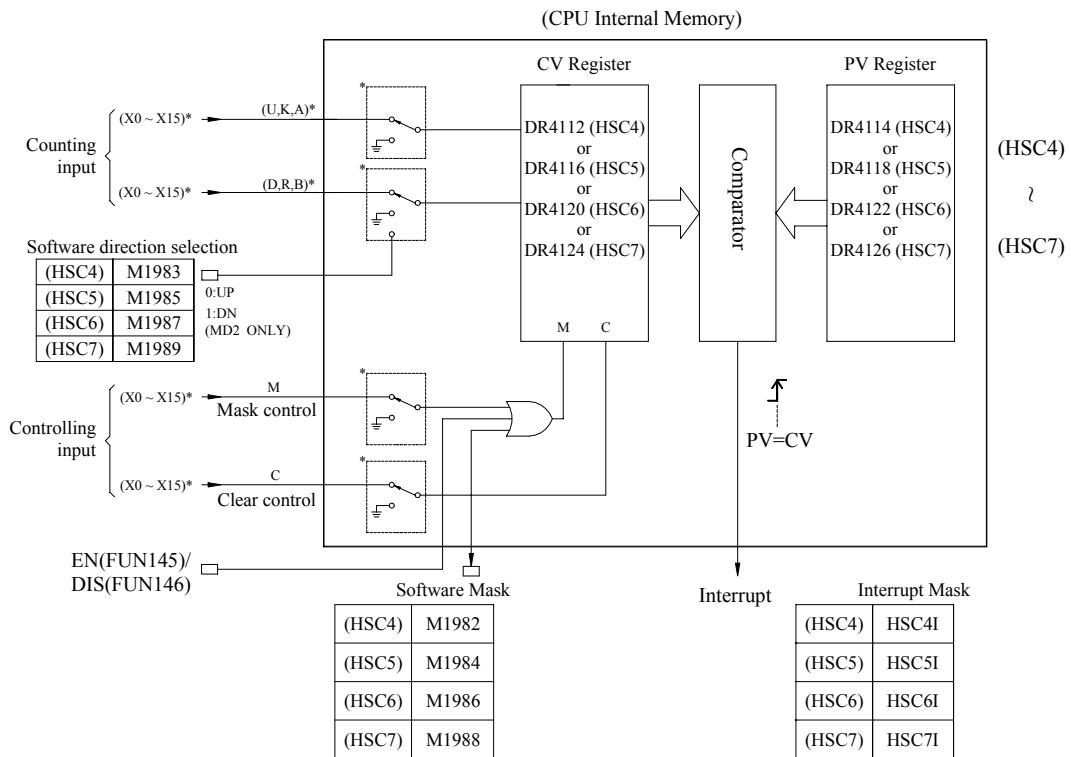
- Use FUN93 to write CV register content to ASIC chip. Resets and updates the CV of hardware counter in ASIC chip with CPU CV register.



- Use FUN93 to write CPU's internal PV register value to PV register of hardware counter of ASIC chip.



System Architecture of HHSC ( HSC0~HSC3 )



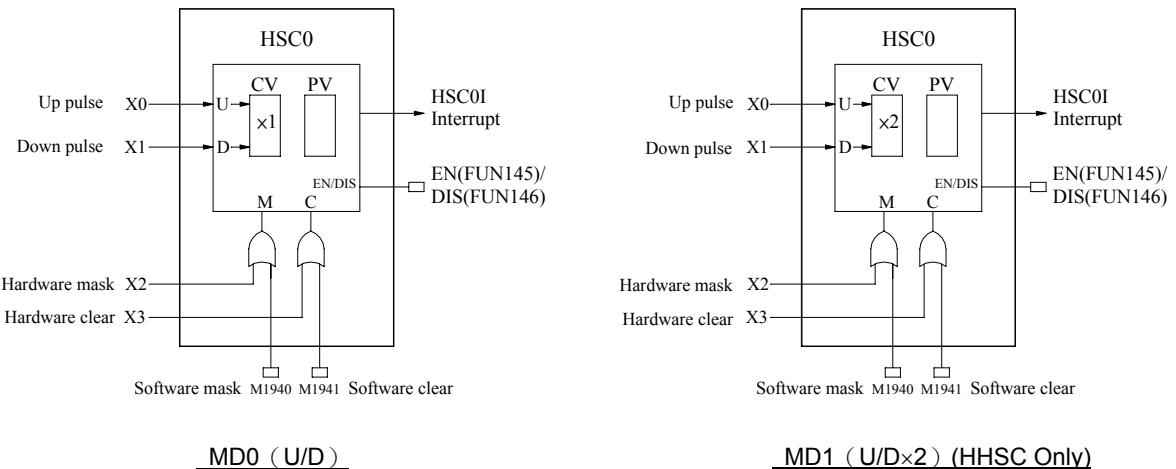
### System Architecture of SHSC ( HSC4~HSC7 )

- All control signals of HHSC and SHSC are default as Active High (i.e. Status =1 for active and 0 for non-active). In order to cooperate with the sensor's polarity, the HHSC counting inputs (U, D, K, R, A and B) and control inputs (M and C) can be selected for polarity inverse.
- By default when the MASK control signal, M, is 1, the HSC counting pulse will be masked without any counting being performed and all HSC internal status (such as CV and PV) will remain unchanged. The HSC will function normally only when M returns to "0". Some sensors have Enable outputs which function is on the contrary to MASK. Counters will not count when Enable = 0 and can only start functioning when Enable = 1. Then, function of inverse polarity input of MASK can be selected to cooperate with the sensors having Enable output.
- When the CLEAR control signal, C, is 1, the HSC internal CV register will be cleared to 0 and no counting will be performed. The HSC will start counting from 0 when C returns to 0. Ladder program can also directly clear the CV register (DR4112, DR4116, DR4120, and DR4124), so as to clear the current counting value to 0.
- The four sets of FB-PLC HHSC are located in the ASIC chips where the CV or the PV registers the user can't access directly. What the user can access are the CV registers (DR4096~DR4110) located in the CPU internal memory. Ideally, the contents of CV and PV registers in the chips should be updated simultaneously with the CV and PV registers in the CPU internal memory. However, to keep the correspondence between the two must be loaded or read by the CPU when they, in fact, belong to two different hardware circuits. It is necessary to use FUN93 to load the CV and the PV registers inside the CPU to the respective CV and PV registers (to allow HHSC to start counting from this initial value. Then, FUN92 can be used to read back the counting value of the HHSC CV register in the chips to the CV register in the CPU (i.e. the CV register in the CPU has the bi-direction function). Since read can only be carried out when FUN92 is executed (so-called "sampling" reading), it might result in difference between the HHSC CV value in the chips and the CV value in the CPU, the deviation will get greater especially when the counting frequency is high.

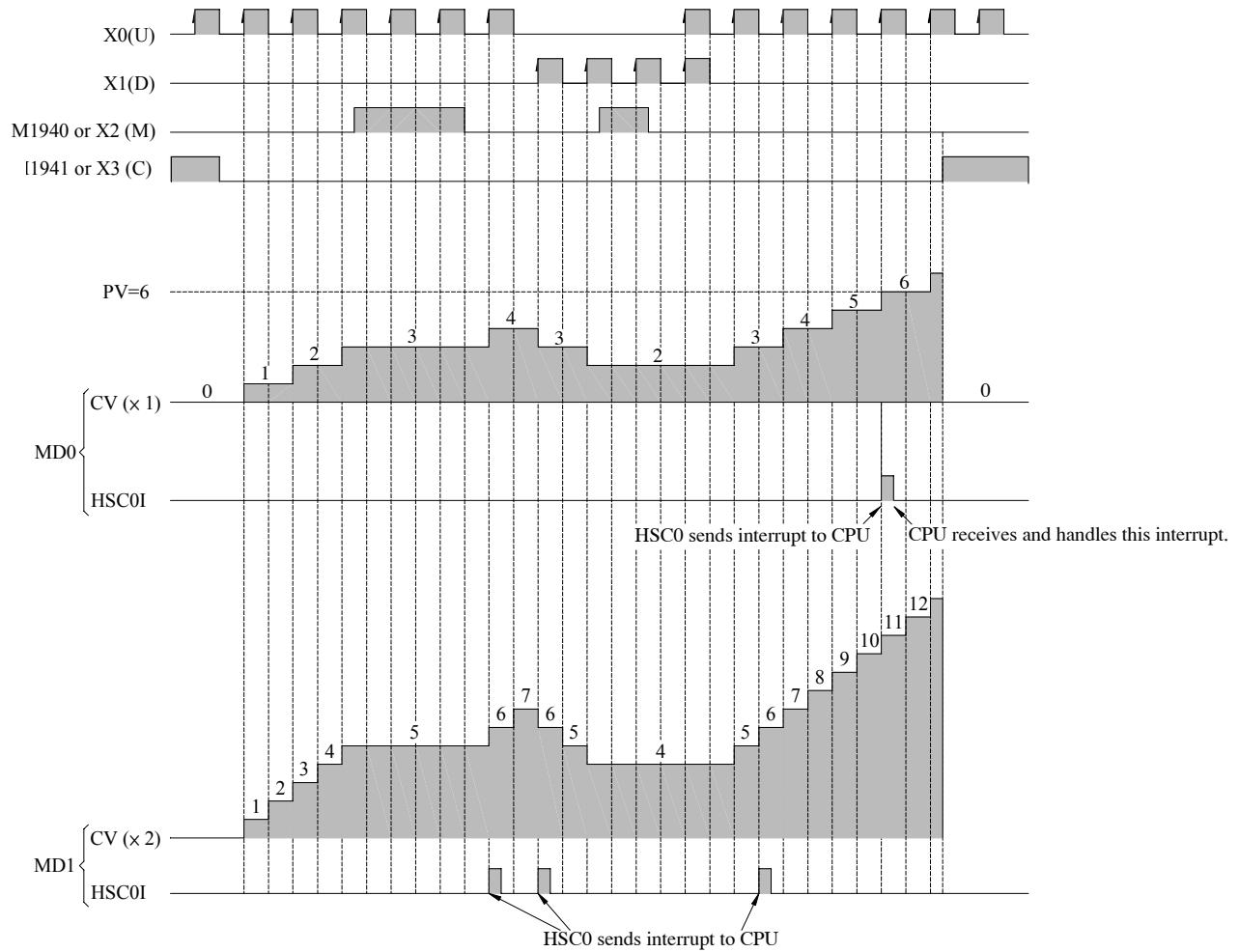
- When the counting frequency is not high or the demand for positioning precision is not so much, using FUN92 in the main program to read the current counting value and then incorporate comparator instruction is adequate for a simple counting positioning control.
- When the demand for positioning precision is higher, or in the multi-zone count setting control, it may use the FUN92 to read the current counting value while in the time base interrupt routine and incorporate compare instruction to perform more precise counting positioning control (for this application mode, the counter is confined to 16 bit application).
- As the demand for positioning precision is extremely high, it must use the preset interrupt function of hardware counter. The preset value can load by FUN93 into the PV register of HHSC in the chipset. When CV value of HHSC reaches this preset value, the hardware comparator in the HHSC will send interrupt to CPU at the very moment CV=PV, and jump to interrupt subroutine to do real time control or procession.
- SHSC, on the other hand, uses the interrupt method to request an interrupt signal to the CPU when the counting input is on the rising edge. Then, the CPU will determine whether it should decrease or increase the internal CV register (since the CV register itself in the CPU is a SHSC CV register, no FUN92 or FUN93 is required). Each time when CV is updated, if the CPU find that it is equal to the PV register value, the CPU will jump immediately to the corresponding SHSC interrupt service routine for processing. Whenever there has a change in SHSC counting or control input can cause the CPU to be interrupted. The higher the counting frequency, the more of CPU time will be occupied. The CPU responding time will be considerably increased or even Watchdog time-out will be caused to force the PLC to stop operating. Therefore, it is preferred to use HHSC first; if it needs to use SHSC, the sum of all FB-PLC SHSC input frequencies should not exceed 8KHz.
- None of the special relay controls, such as software MASK, CLEAR and direction control, is real time. This means that although MASK, CLEAR or direction change has been set during routine scanning, the signal will only be transmitted to HSC when I/O updating is under way after the completion of routine scanning. Hence, it is not suitable for the real time control in HSC operation (which should be mainly used for initial setting before HSC operation). Should real time control be required, please use hardware to control input or apply the FUN145(EN), FUN146(DIS), FUN92(HSCTR), and FUN93(HSCTW) etc. instructions for control.
- Every HSC is equipped with the functions, ENable(FUN145) and DISable(FUN146), and can only be operated when HSC is enabled. When HSC is disabled, the whole HSC, no matter it is HHSC or SHSC, will be in a complete stagnant status (CV and PV values will remain the same without interrupt as if no HSC exists) until the Enable command is executed. When Enable is executed, HSC will return to the status and the function before Disable. HSC is default as enable when HSC configuration is made. Control of enable and disable can be made at any time in the routine according to actual requirement.
- In the interrupt service subroutines of HSTAI, 1MS~100MS, if it is used with the FUN92 to read the current counting value of hardware high speed counter, only the 16 bit of Low Word (R4096, R4100, R4104, R4108) is effective.

### 11.2.1 The Up/Down Pulse Input Mode High-Speed Counter (MD0, MD1)

The up/down pulse input of high-speed counter has up counting pulse input (U) and down counting pulse input (D) that are independent to each other without any phase relationship. Each of them will +1 (U) or -1 (D) on the CV value when the rising edge of the pulse input occurs (both positive and negative edge for MD1). This also applies when the rising (or falling) edge of the U and D pulse occur simultaneously (it will offset with each other). Both of the two modes have the built-in software MASK and CLEAR (CLEAR is not available for SHSC) control functions, when the control function are not in use should keep the status (such as M1940 and M1941) as "0". Apart from the built-in software MASK and CLEAR, the controls of hardware MASK and CLEAR can also be configured. The MASK control is first performed by the OR operation of the hardware and software control, then the result is send to the HSC MASK control M, and so does CLEAR. Taking HSC0 as an example, the function schematic diagrams for MD0 and MD1 configured separately are shown as below.

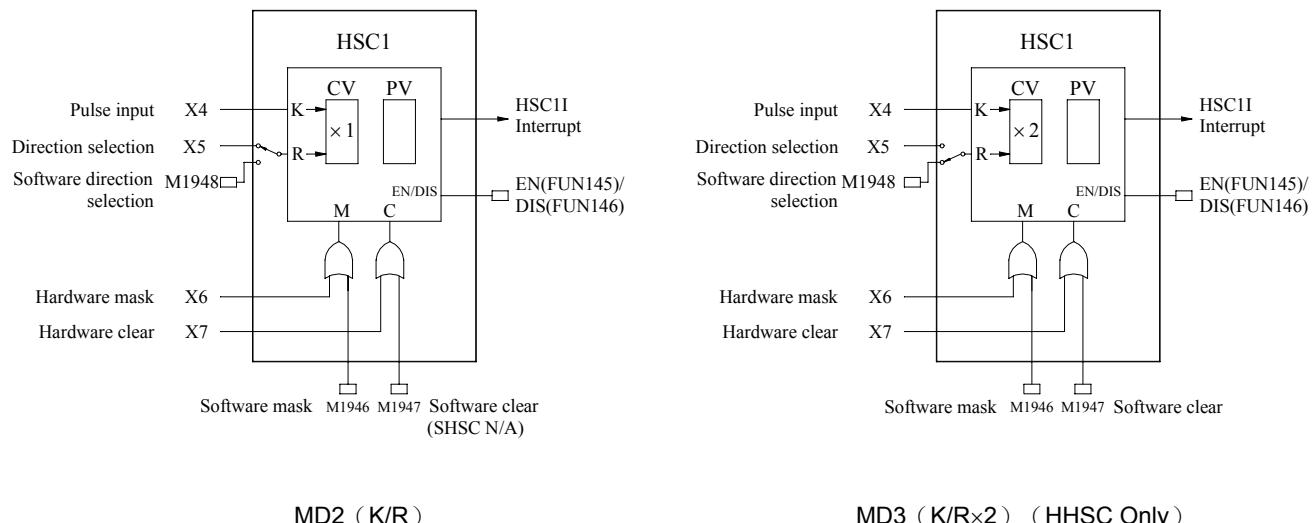


The Waveforms of the HSC, which is configured as up/down pulse input mode, and PV value is preset to 6:



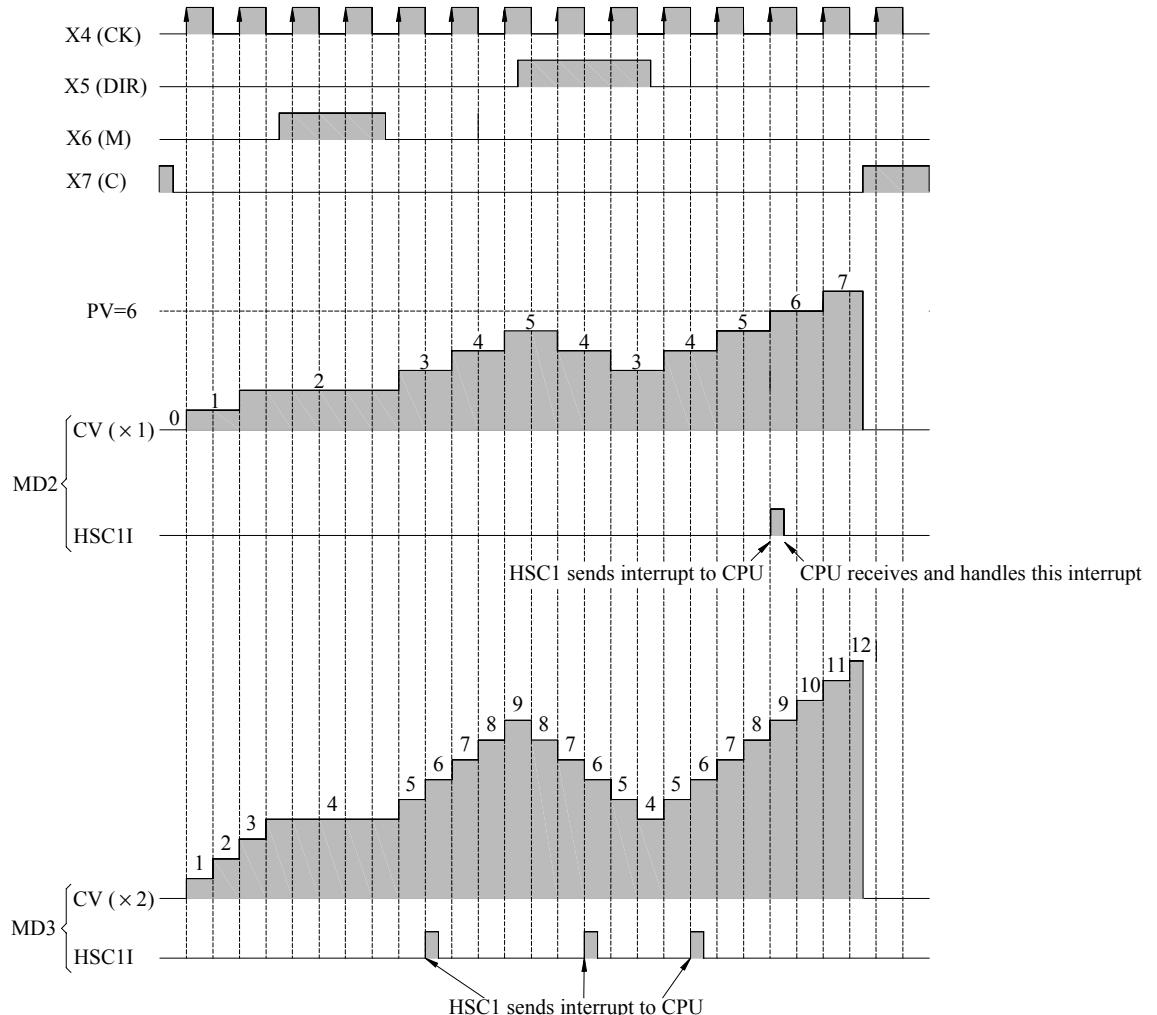
## 11.2.2 Pulse-direction Input Mode High-Speed Counter (MD2, MD3)

The pulse-direction input mode high-speed counter only has one counting pulse input K (Clock). It requires another direction input R (Direction) to decide whether the CV value should +1 (R=0) or -1 (R=1) when the rising edge (both rising and falling edges for MD3) of counting pulse arrives. The same applies to counting of MD2 and MD3 except that MD2 only counts on the rising edge (+1 or -1) and MD3 counts on both rising and falling edges of CK pulse (twice the counts of MD2). These two modes have built-in software MASK, software CLEAR (SHSC does not have clear). When control function is not in use, it must keep the status (such as M1946 and M1947 in this example) to be 0. Apart from the built-in software MASK and CLEAR, the controls of hardware MASK and CLEAR can also be configured. The MASK control is first performed by the OR operation of the hardware and software control, then the result is send to the HSC MASK control M, and so does CLEAR. The function schematic diagrams of HSC1 configured individually for MD2 and MD3 are shown as below.



Direction selection of MD2 and MD3 HSC, for HSC or SHSC, can be come from the external inputs (such as X5 in this example) or the special relay in CPU (such as M1948 in this example) to reduce the usage of external input points.

The diagram below is the waveform diagram for the relationship between counting and control of the two HSC. In this example the PV value is to 6.

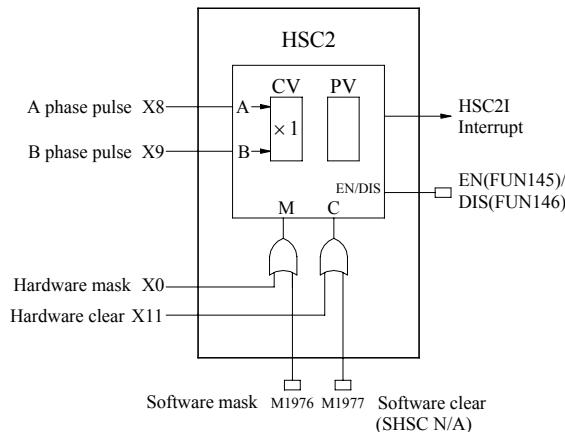


### 11.2.3 AB Phase Input Mode High-Speed Counter (MD4,MD5,MD6,MD7)

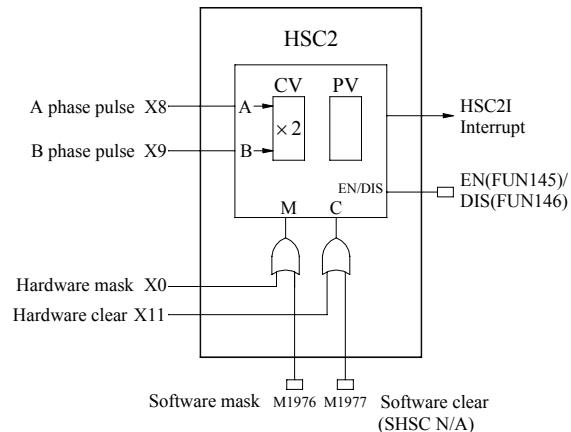
The AB phase high-speed counter is equipped with phase A and phase B pulse input with counting value +1 or -1, depending on the phase relationship between the two, i.e. the related counting of the two phases. If phase A is ahead of phase B, the CV value should be +1, else, the CV value should be -1. The counting of the four modes, MD4 (A/B), MD5 (A/B $\times$ 2), MD6 (A/B $\times$ 3) and MD7 (A/B $\times$ 4), of AB phase HSC are similar. Their differences are:

- ① MD4 (A/B) : The rising edge of A is +1 when A is ahead of B and the falling edge of A is -1 when A is behind B.
- ② MD5 (A/B $\times$ 2): The rising and falling edges of A are +1 when A is ahead of B, and -1 when A is behind B (twice the counts of MD4).
- ③ MD6 (A/B $\times$ 3): The rising and falling edges of A and rising edge of B are +1 when A is ahead of B. The rising and falling edges of A and the falling edge of B are -1 when A is behind B (three times the counts of MD4).
- ④ MD7 (A/B $\times$ 4): The rising and falling edges of A and B are +1 when A is ahead of B and the rising and falling edges of A and B are -1 when A is behind B (four times the counts of MD4).

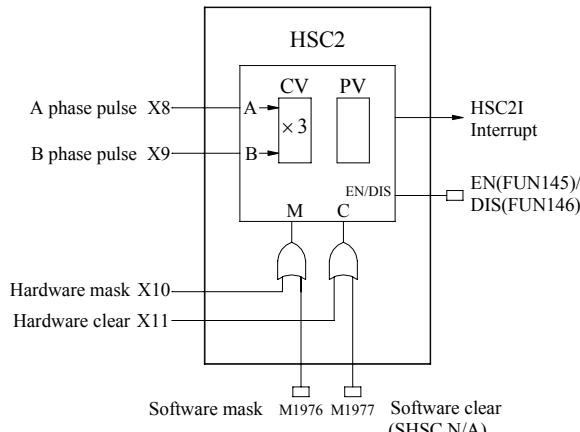
Other MD4~MD7 HSC modes also have built-in software MASK, software CLEAR (SHSC does not have clear). When control function is not in use, it must keep the status (such as M1946 and M1947 in this example) to be 0. Apart from the built-in software MASK and CLEAR, the controls of hardware MASK and CLEAR can also be configured. The MASK control is first performed by the OR operation of the hardware and software control, then the result is send to the HSC MASK control M, and so does CLEAR. The function schematic diagrams of HSC2 for the four MD4~MD7 HSC modes are shown as below.



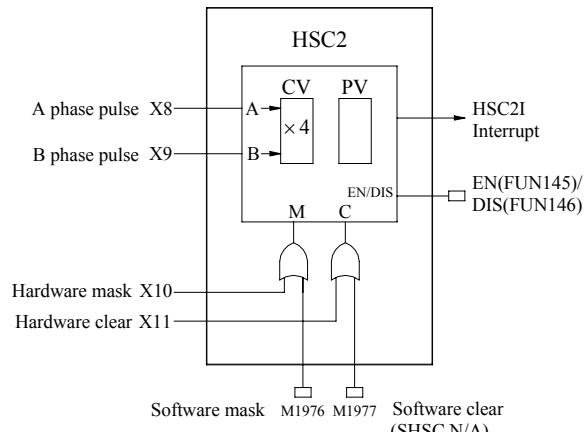
MD4 (A/B)



MD5 (A/B×2) (HHSC Only)

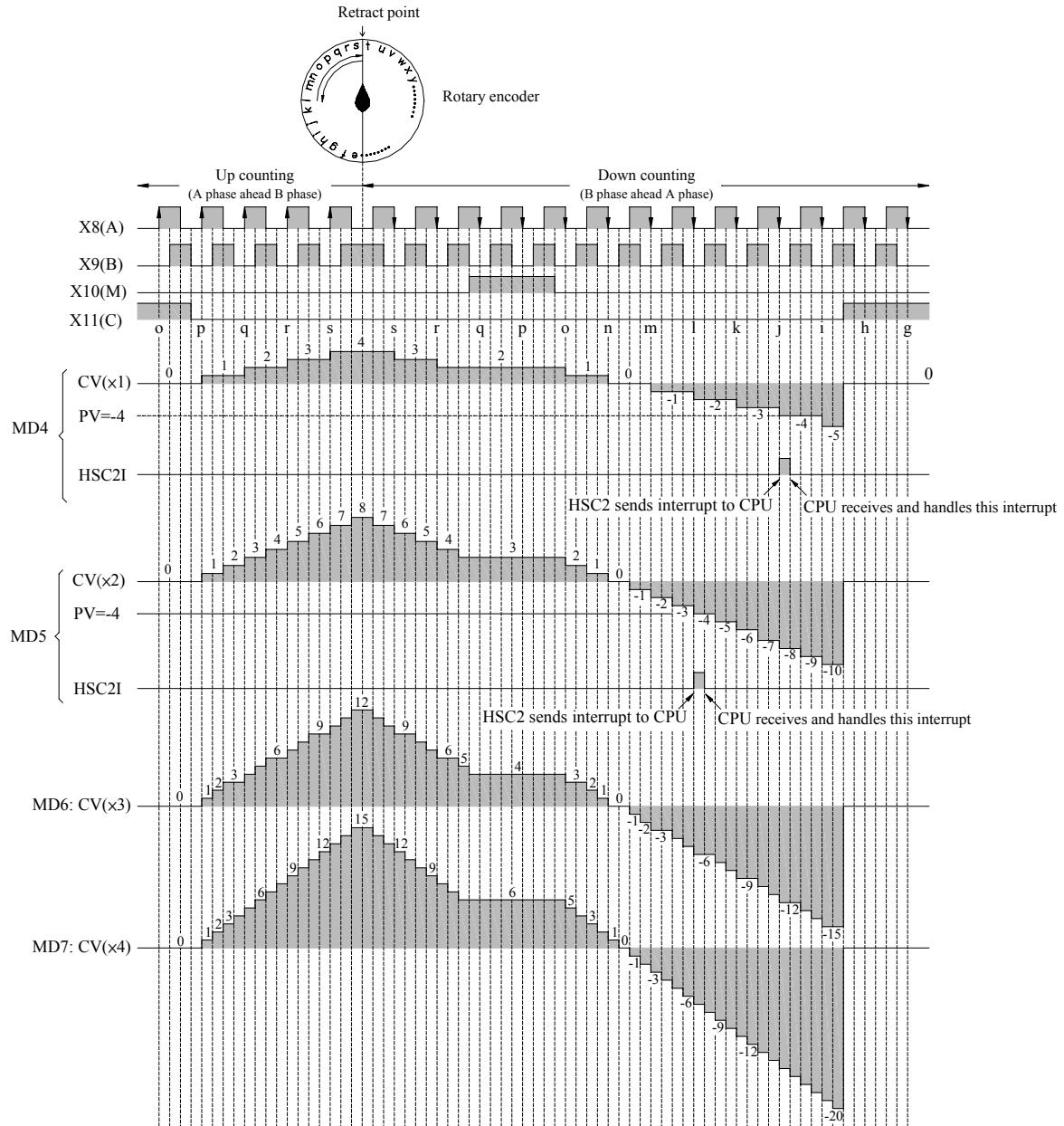


MD6 (A/B×3) (HHSC Only)

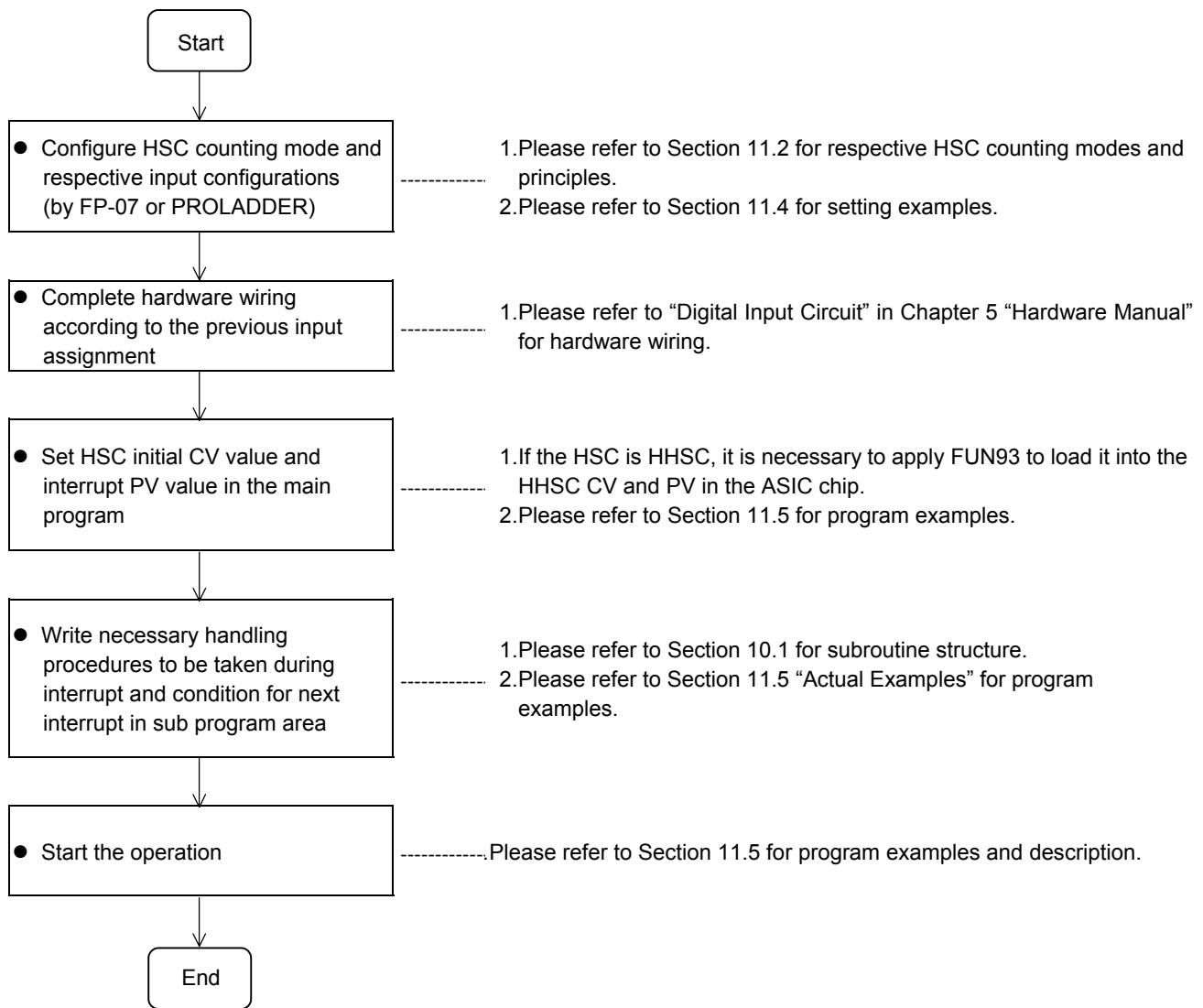


MD7 (A/B×4) (HHSC Only)

The diagram below is the waveform diagram for the relationship between counting and control of the four HSC modes in this example when the PV value is set as at 6.



### 11.3 Procedure for FB-PLC High-Speed Counter Application



### 11.4 HSC/HST Configuration

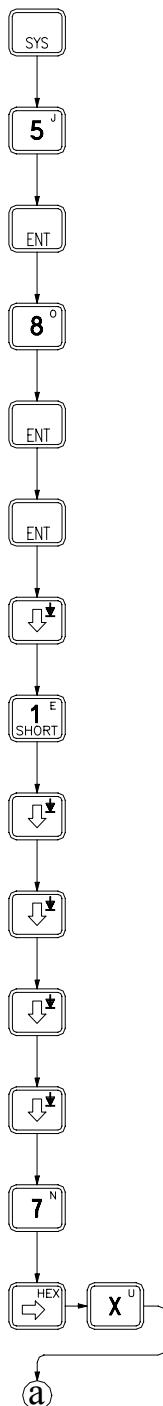
The screen of FP-07 will be taken as an example to describe HSC Configuration in this section (as to PROLADDER, please refer to the description for PROLADDER). The HSC Configuration, in sequence, includes the following 5 items:

- ① Select assignment for HSC/HST (only HHSC provides this item selection function). Proceed to next item if selection is HSC. No other items are required if configured as HST.
- ② Assign respective HSC counting modes (MD0~MD7). After keying in the mode number, FP-07 will automatically display the HSC counting and control input names of the mode and reserve space for users to key in the external input point number Xn. The blank mode field indicates the HSC is not in use.
- ③ Determine whether the respective counting inputs (U, D, K, R, A and B) and control inputs (M and C) are to be applied or not (reserve the space if not in use and fill in the Xn value if it is to be applied. As respective Xn input values of HHSC are fixed, it requires only to key in alphabet "X" and FP-07 will automatically make up the preset number n).

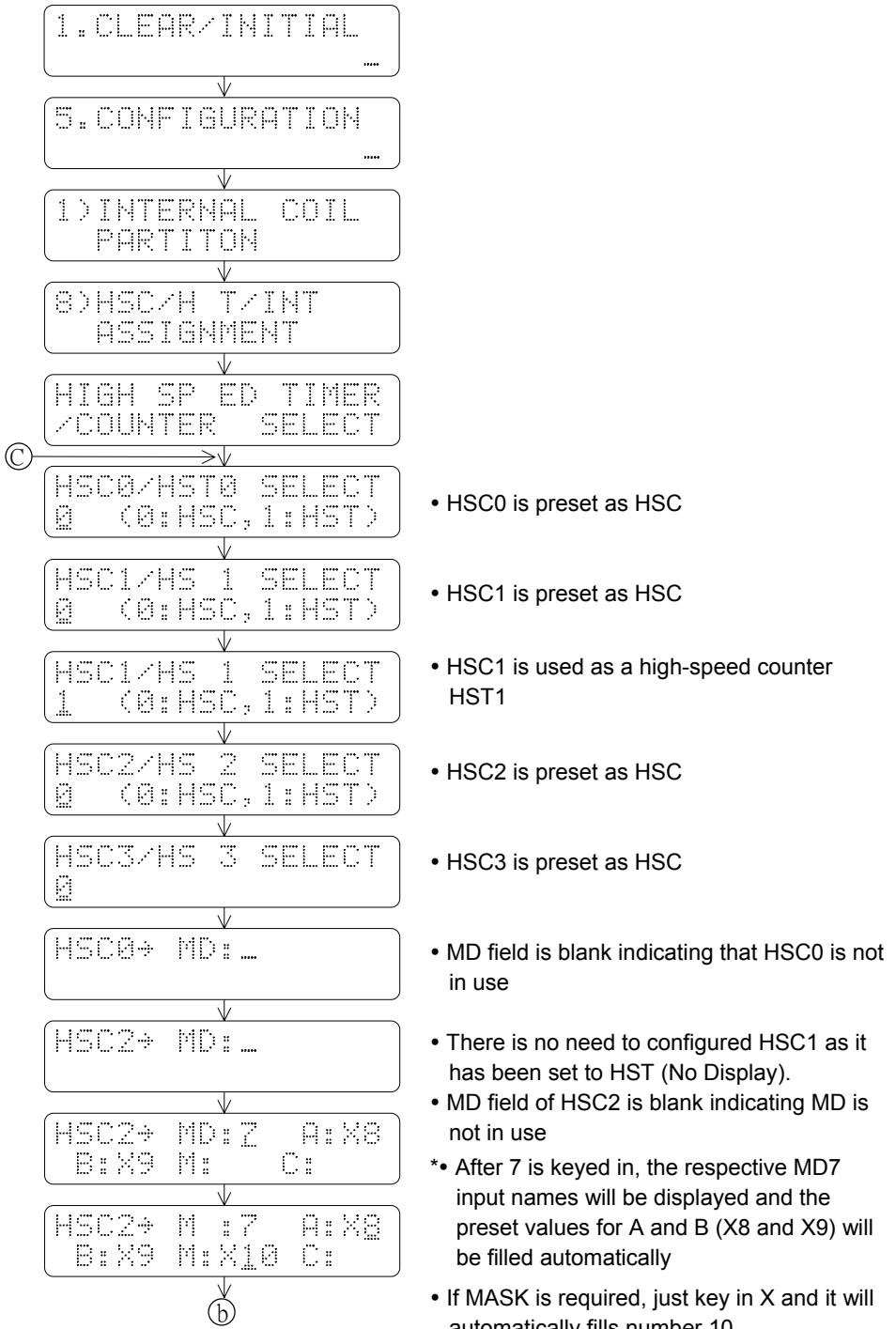
- ④ Select whether the polarity of each HHSC counting input (U, D, K, R, A and B) is inverse or not, so as to match the polarity of the encoder (0: Not inverse, 1: Inverse. Preset as 0).
- ⑤ Select whether the polarity of each HHSC control input (M and C) is inverse or not, so as to match the polarity of the encoder (0: Not inverse, 1: Inverse. Preset as 0).

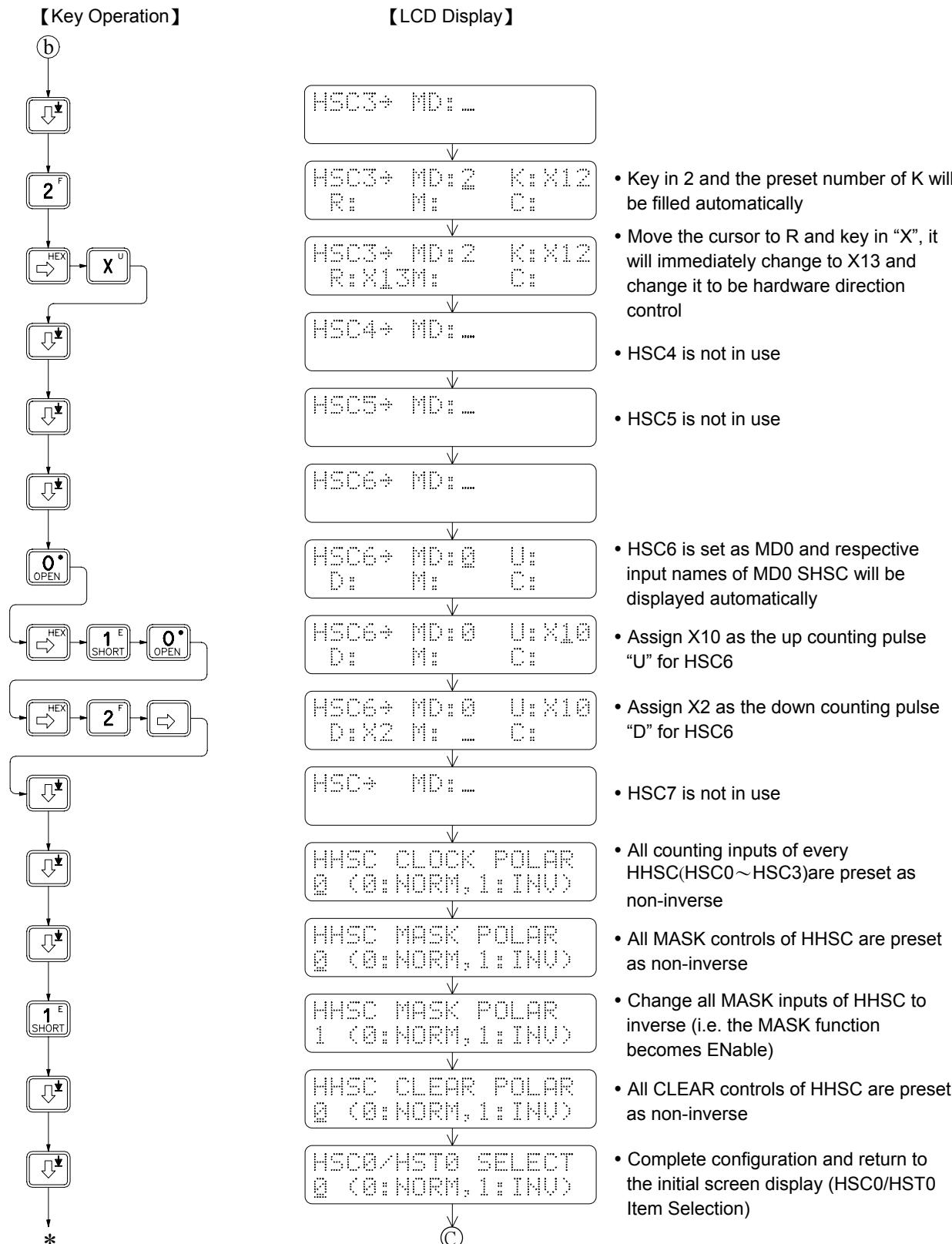
Examples applying FP-07 to perform the above mentioned ①~⑤ configurations

【Key Operation】



【LCD Display】





- Input value modification can be made by directly key in the new value to overwrite. Use **CLR** key to delete any input value, if required.
- A blank field (without any value input) indicates the application of the HSC or the input is not required.
- "CLOCK" in the previous example represents the "Counting Input", i.e. U and D, K and R or A and B, of HHSC.
- "POLAR" represents "POLARITY", i.e. selection of inverse or non-inverse.

- The input point for respective HHSC counting and control inputs are fixed. Therefore, in the “Configuration Examples” of the previous example, it needs only to key in “X” for each HHSC input to indicate that the input is to be applied and FP-07 or PROLADDER will automatically make up the preset number for X, to which no change will be allowed. The user may assign respective SHSC counting or control inputs between X0~X15 freely. Hence, it is necessary to key in both the “X” and the number n for SHSC input point number to make it complete. All preset or selectable input point numbers, software MASK, software CLEAR, direction selection and other related numbers of HHSC and SHSC are summarized in the table below:

Type		HHSC				SHSC			
		HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
CV Register Number		DR4096	DR4100	DR4104	DR4108	DR4112	DR4116	DR4120	DR4124
PV Register Number		DR4098	DR4102	DR4106	DR4110	DR4114	DR4118	DR4122	DR4126
Counting Input	U, K or A	X0	X4	X8	X12	X0~X15	X0~X15	X0~X15	X0~X15
	D, R or B	X1	X5	X9	X13	X0~X15	X0~X15	X0~X15	X0~X15
Control Input	M	X2	X6	X10	X14	X0~X15	X0~X15	X0~X15	X0~X15
	C	X3	X7	X11	X15	X0~X15	X0~X15	X0~X15	X0~X15
Software MASK Relay		M1940	M1946	M1976	M1979	M1982	M1984	M1986	M1988
Software CLEAR Relay		M1941	M1947	M1977	M1980	Not available			
Software Direction Selection (MD2,3 Only)		M1942	M1948	M1978	M1981	M1983	M1985	M1987	M1989
Interrupt Subroutine Label		HSC0I	HSC1I	HSC2I	HSC3I	HSC4I	HSC5I	HSC6I	HSC7I

\* MA model CPU provides two SHSCs - HSC2 and HSC3

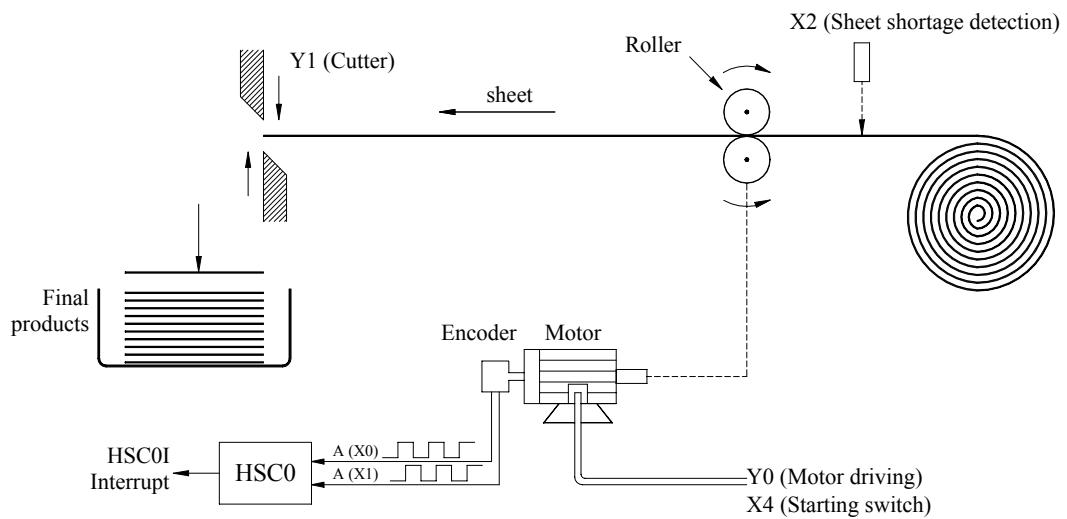
As far as HHSC is concerned, since all input points are fixed, some input points, if required, will need to exist but cannot be selected by the user (e.g. HSC2 is MD7; in this example, whose A/B input must be used in pair, as X8 and X9. For HSC3 is MD2; K must be X12). FP-07 or PROLADDER will automatically fill these input points, and they should not be changed. For those input points that do not need to be applied in pair or the user can select their existence (such as M and C of HHSC, U and D of MD0 or all SHSC inputs), FP-07 or PROLADDER will leave a space for selection input.

- The input point of X0~X15 in the table above can only be assigned once (i.e. used as one function), which can't repeat to be used.
- FBN-20MC contains a set of high speed differential input interface (X0~X3); the frequency can reach up to 512 KHz.
- FBN-28MC contains 2 sets of high speed differential input interface (X0~X7); the frequency can reach up to 512 KHz.
- FBN-20MC contains 4 sets of high speed differential input interface (X0~X15); the frequency can reach up to 512 KHz.
- The max. input frequency of hardware HSC of FBE or FBN (exclude high speed differential input) MC models, can reach 20KHz for single phase, and AB phase input frequency can reach up to 10KHz.
- The total input frequencies of SHSC can't be exceed 8 KHz; the higher the frequency, the more it occupy the system (CPU) time, and the scanning duration will be extended abruptly.

## 11.5 Examples for Application of High-Speed Counter

**Example 1** This example uses high-speed counter for equal-length cutting control.

### Mechanism



**HSC Configuration** (Just set HSC0 to MD7 and complete the configuration)

HSC0/HST0 SELECT  
0 <0:HSC, 1:HST>

- No change is required on preset values

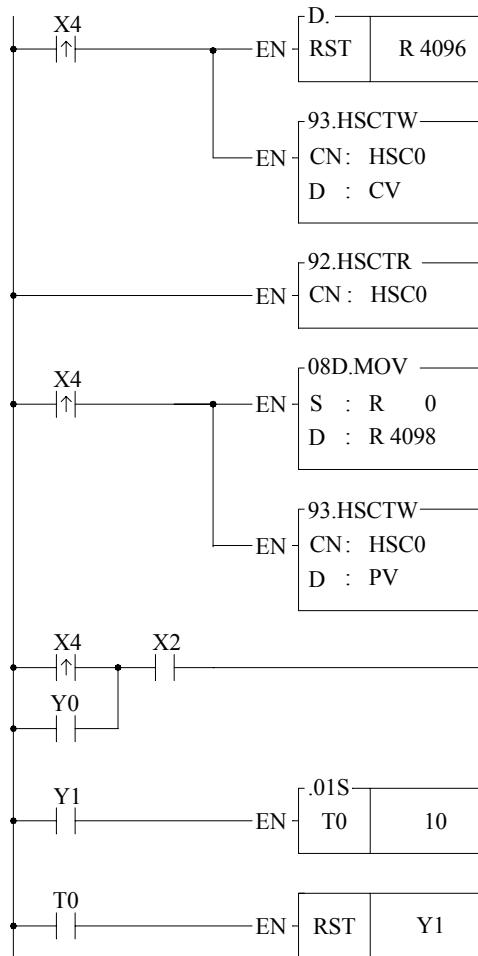
HSC0# MD:#7 A:#0  
B:#1 M:# C:#

- Setting HSC0 to be MD7 (A/B×4), and resolution of cutting will be increased by 4 times.
- M and C are not required for application

- All other settings (polarity of counting and control inputs) are default (non-inverse) and need not to be changed.

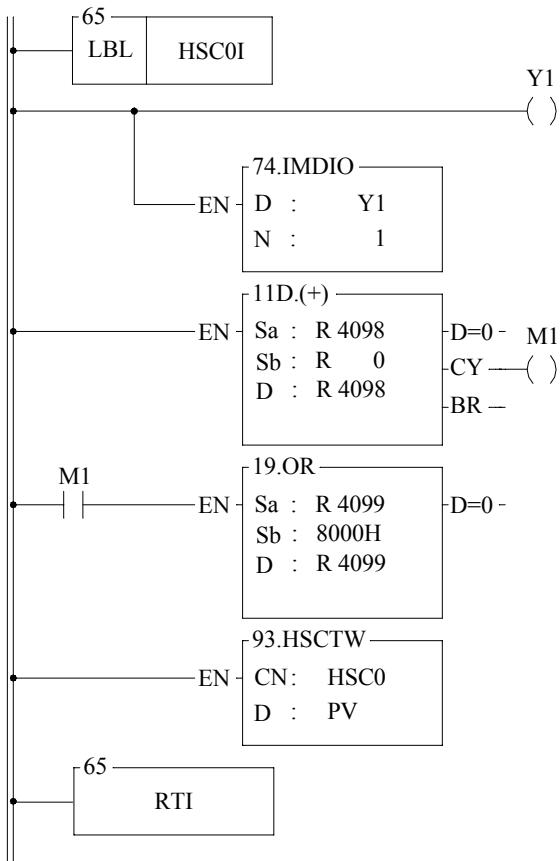
## Control Program

### 【Main Program】



- CLEAR the current value register DR4096 of HSC0
- Use FUN 93 to write the contents of the current value register into the CV register of HSC0 in the ASIC chip  
CN =0 indicates HSC0  
D =0 indicates CV
- Use FUN 92 to read the counting value of the HSC0 CV register in the ASIC chip (store into DR4096)
- Store the counting of cutting length DR0 into DR4098 and use FUN93 to store the value into the PV register of HSC0 in the ASIC chip  
CN =0 indicates HSC0  
D =1 indicates PV
- Start the motor
- Turn the cutter Y1 ON for 0.1 second

### 【Subroutine】

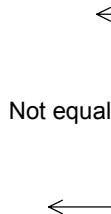


- When HSC0 CV=PV in the ASIC chip, the hardware will automatically execute the interrupt subroutine labeled HSC0I
- When counting is up, turn Y1 ON (to cut materials)
- Output Y1 immediately to reduce the error caused by scan time

- Calculate new cutting position and load HSC0 PV

HSC0 PV  $\begin{cases} 2147483647(7FFFFFFFH) \\ \downarrow +1 \\ 2147483647(80000000H) \end{cases}$

FUN11D  $\begin{cases} 2147483647(7FFFFFFFH) \\ \downarrow +1 \\ 0 (\text{and CY}=1) \end{cases}$



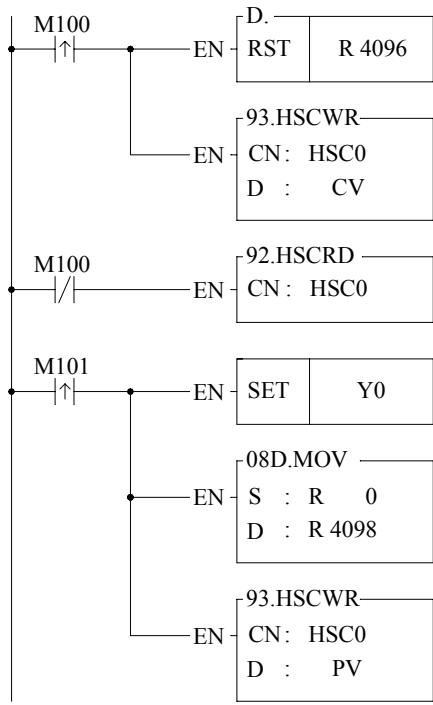
Therefore, it is necessary to use the CY output of FUN11D to rectify DR4098 (1 $\rightarrow$ b31)

### 【Description】

1. The main program will initialize the HSC0 CV (CV=0) in advance and move the cropping length (DR0) to the HSC0 PV before starts Y0 to turn on the motor for material conveying.
2. When CV reaches PV, the length of R0 is added to the PV before being reloaded into HSC0 PV.
3. When all materials are rolled out, the material shortage detector X2 will be ON and stop the motor.

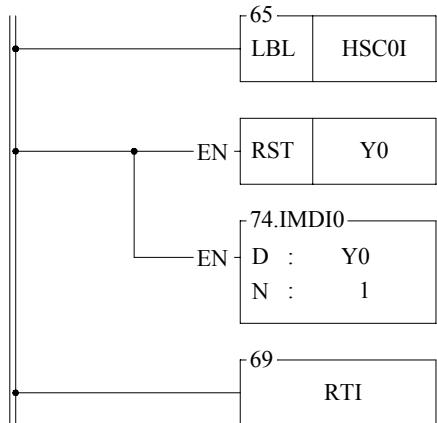
**Example 2** Example of high speed counting up action processed by Interrupt

**【Main Program】**



- As M100 change from 0→1, it clears the current value register to 0
- Employ FUN93 to write the content of current value register into the CV of HSC0 in ASIC chip (reset)  
CN =0, represents HSC0  
D =0, represents CV
- Employ FUN92 to read out the current counting value of HSC0 in ASIC chip, and store it into the CV register (DR4096)  
CN=0, represents HSC0
- As M101 change from 0→1, start Y0 ON (begin to operate)
- Move target point DR0 to preset register DR4098
- Employ FUN93 to write the content of preset register into HSC0 PV in ASIC chip, which serves as setting value of counting up interrupt  
CN=0, represents HSC0  
D =1, represents PV

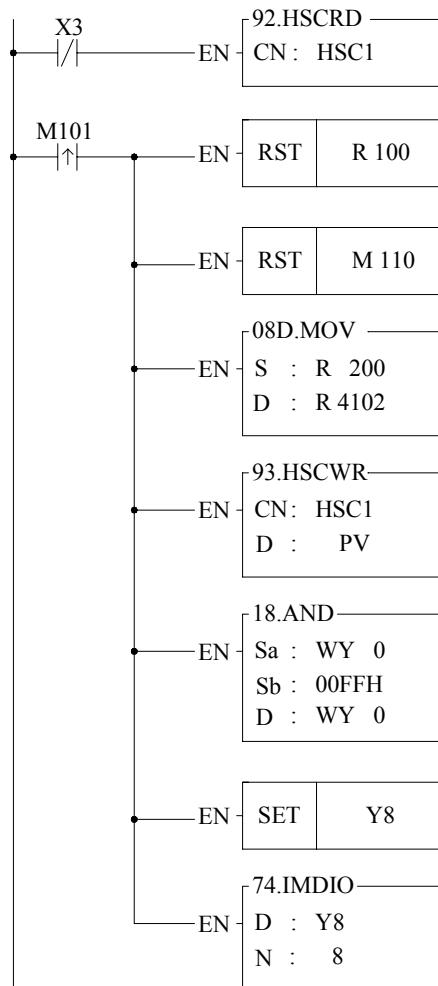
**【Subroutine】**



- Hardware high speed counter #0 interrupt label
- When time up, it sets Y0 OFF (stop)
- Let Y0 out immediately, so as to stop promptly (otherwise Y0 will have a scan time output delay)

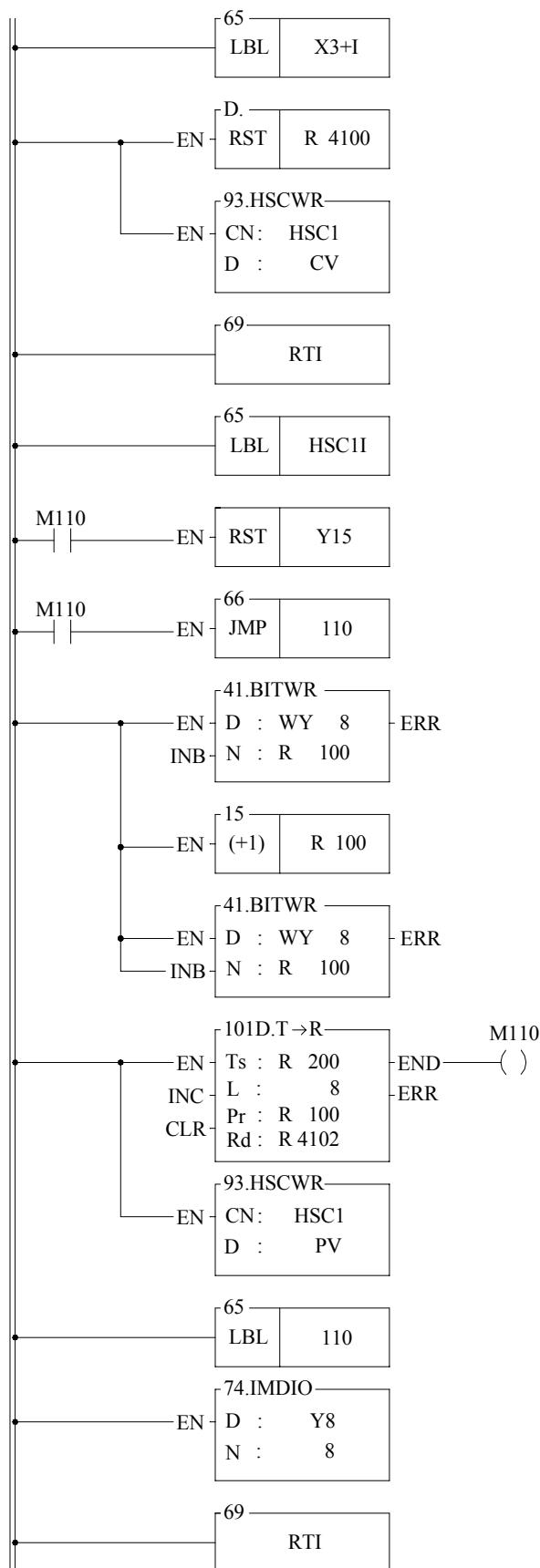
**Example 3** Example of Immediate response of multi-zone high speed counting up by Interrupt Processing

**【Main program】**



- Employ FUN92 to read out the current value of HSC1 in ASIC chip, and store it into current value register DR4100  
CN =1, represents HSC1
- As M101 change from 0→1, clears the pointer register to 0
- Clears the flag of the last zone to be OFF
- Move the counting of travel length DR200 (section 0) to preset register DR4102
- Employ FUN93 to write preset register content into HSC1 PV in ASIC chip, which serve as counting up setting value.  
CN =1, represents HSC1  
D =1, represents PV
- Clear Y8~Y15 to be OFF
- Set Y8 ON, it represents that it is at the zone 0 currently
- Set Y8~Y15 output t immediately

【Subroutine】



- Label name for the X3 rising edge interrupt service subroutine of X3+1  
(it must assign X3 to be the rising edge interrupt input)
- When X3 changes from 0→1, it clears the current register to be 0
- Employ FUN93 to write the current register content to the HSC1 CV in ASIC chip (reset).  
CN =1, represents HSC1  
D =1, represents CV
- Labeled as HSC1I hardware high speed counter interrupt service subroutine.
- Turn Y15 OFF when the last zone finished.
- Make the previous zone output OFF
- Set the pointer point to the next zone
- Set the output of next zone to be ON
- Move the counting value of next zone (beginning from DR200 pointer pointed register) to the preset register DR4102
- When it's the last zone, the M110 is ON
- Employ the FUN93 to write the preset value into the HSC1 PV in the ASIC chip, which serves as counting up interrupt setting point.  
CN =1, represents HSC1  
D =1, represents PV
- Y8~Y15 output transmitting immediately

## 11.6 FB-PLC High-Speed Timer

The minimum timing unit (time base) of an ordinary PLC can only reach 1ms, on which the deviation in scan time should also be added. Therefore, it is necessary to apply high-speed timer (HST) if a more precise timing (e.g. using timer to cooperate with HSC for frequency measurement) is required.

FB-PLC is built in a high-speed timer (HSTA) with a time base of 16 bits/0.1ms and, as described previously, four 32-bit high-speed counters (HSC0~HSC3) of HHSC that can be converted to high speed timer (HST0~HST3) with a time base of 32 bits/0.1ms for using. Thus, FB-PLC can have up to five high-speed timers. As HSC and INT, all HST can be enabled or disabled (default as enable) by the instructions EN (FUN145) and DIS (FUN146). HSTA and HST0~HST3 are respectively described as below.

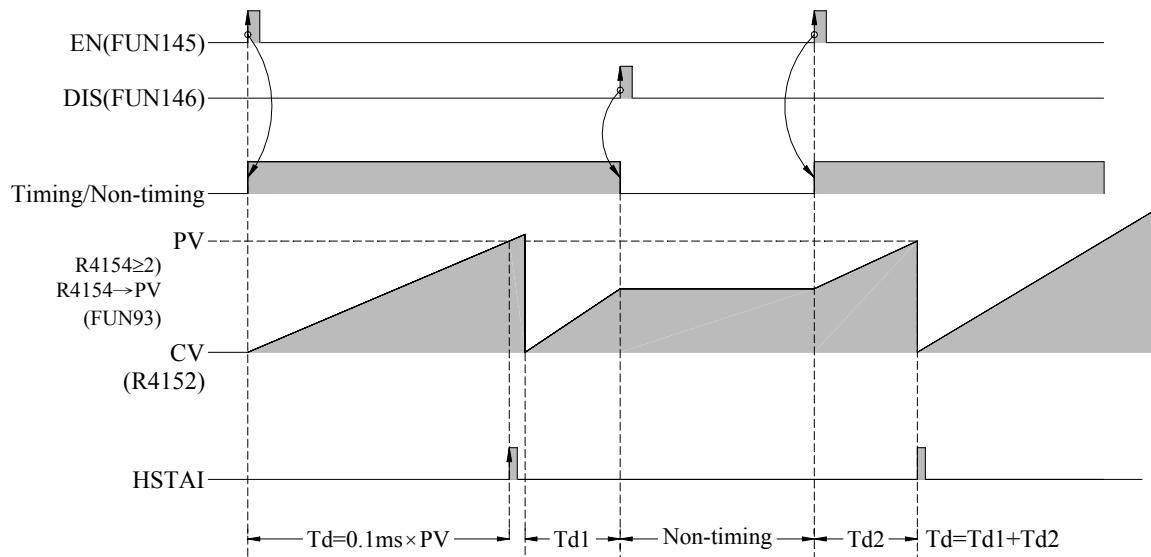
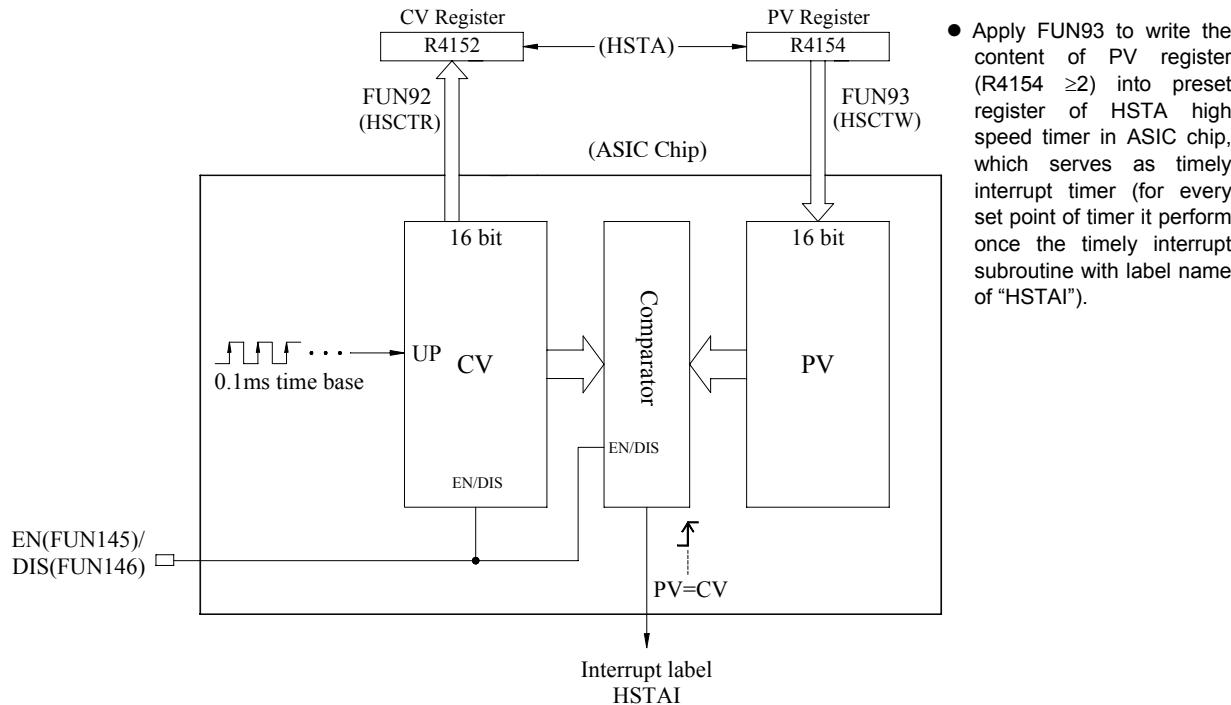
The finest time base for most of the ordinary PLC is 10ms. Though some PLC may have HST with a time base of 1ms. When deviations in the PLC scan time is taken into consideration (e.g. if the scan time is 10ms when the time base is 1ms, the total deviation still exceeds 10ms), the figure of 1ms becomes meaningless. Therefore, these PLCs can't be applied in high precision timing. FB-PLC, having a time base of 0.1ms, has no deviation in scan time for its time up is sent out by interrupt to provide a precision 100 times better than ordinary PLCs' timer application and can be used for many applications demanding precision timing.

### 11.6.1 HSTA High-Speed Timer

HSTA is a 16-bit hardware timer built in the ASIC chip. As HHSC, it must use the instruction FUN93 (HSCTW) to load the HSTA and PV register (R4154) to the HSTA PV in the chip, and with the instruction FUN92 (HSCTR) to read for CV. HSTA can be used as a timer having two different functions. FB-PLC will use it as a general 16-bit delay timer when PV  $\geq 2$  and as a 32-bit cyclic timer when PV=0.

## A. HSTA 16-Bit High-Speed Delay Timer (Timely Interrupt Timer)

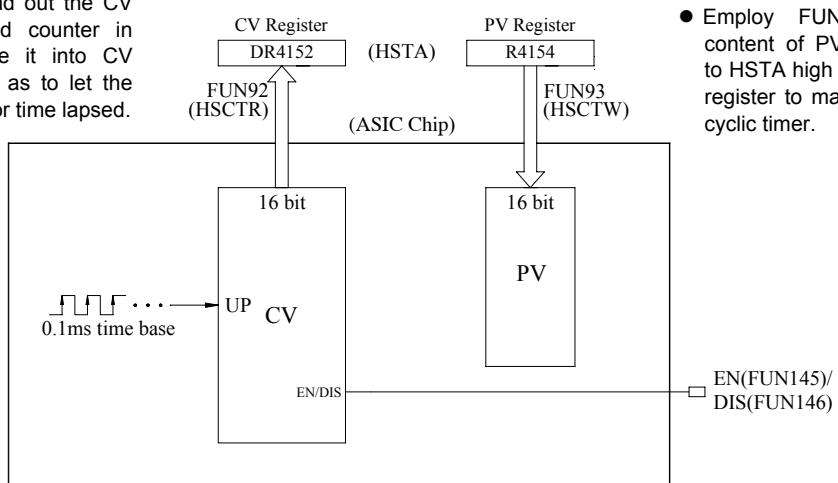
After HSTA starts timing, the delay timer will delay for a time of  $PV \times 0.1\text{ms}$  before sending an interrupt out. When  $PV > 0$ , HSTA served as a delay timer which is 16-bit and its PV value can be set as 0002H~FFFFH (65535 having no positive or negative sign), i.e. the delay time can be set as 0.2ms~6.5535 seconds. Except that having a more precise time base and being able to send an interrupt out immediately at time-up to provide a much higher timing precision, the applications of HSTA are the same as an ordinary delay timer. The diagram below is the structure diagram for HSTA being used as a delay timer. Please refer to Section 11.6.3 “Program Examples” for detailed function and application.



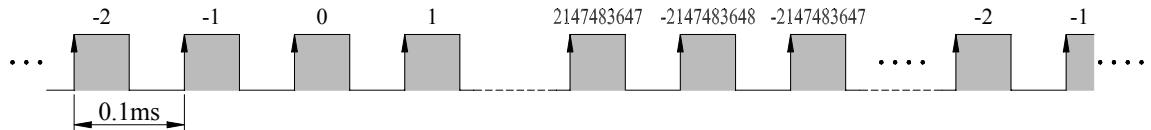
## B. HSTA 32-Bit High-Speed Cyclic Timer

The so-called “Cyclic Timer” is a timer that adds 1 to its current value for every fixed interval and will persistently carry out up counting cyclic timing. Its CV value will cycle around as 0, 1, 2, ... 2147483647, -2147483648, -2147483647, ... -2, -1, 0, 1, 2, ... (as the time base is 0.1ms, CV value x 0.1ms will be its accumulative time). In fact, the cyclic timer is an up counting cyclic timing clock having a time base of 0.1ms that can operate endlessly and be used to read any two events at the time when they occurred and to calculate the time interval between the occurrence of the said two events. The Diagram B as shown below is the structure diagram for HSTA being used as a 32-bit cyclic timer. As shown in diagram, when cyclic timer PV=0, it will not send out the interrupt. To obtain the timing value, it is necessary to use FUN92 to access the CV value from the ASIC chip and save it to the 32-bit CV register (DR4152) in the PLC. The typical application of the cyclic timer is for more precision of turning speed (RPM) detection under the circumstances when the change in turning speed (RPM) is huge or when it is extremely low. Please refer to Example of Section 11.6.3 for description.

- Employ FUN92 to read out the CV of HSTA high speed counter in ASIC chip and store it into CV register (DR4152) so as to let the user know the value for time lapsed.



- Employ FUN93 to write the content of PV register (R4154=0) to HSTA high speed timer set point register to make it serve as 32 bit cyclic timer.

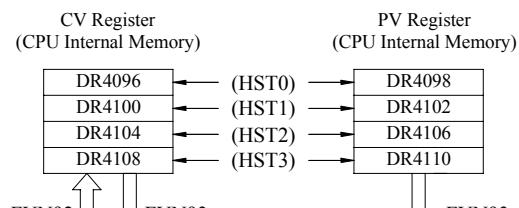


## 11.6.2 HST0~HST3 High-Speed delay timers

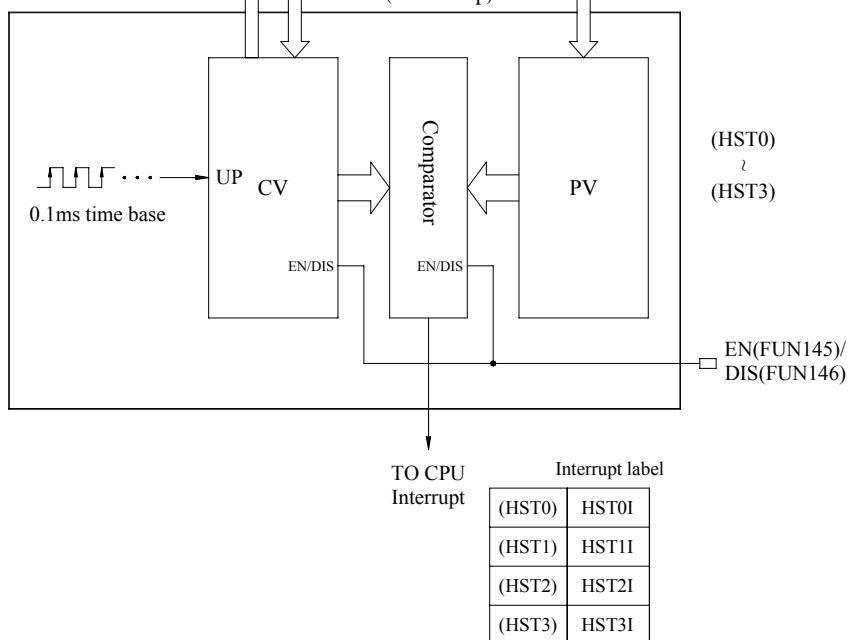
### A. HST0~HST3 High-Speed Delay Timer (Timely Interrupt Timer)

HHSC (HSC0~HSC3) can be planned as four 32-bit high-speed delay timers, HST0~HST3. They have the same functions and time base as a 16-bit HSTA delay timer except that HST0~HST3 are 32 bit to plan HHSC as HST only needs to select “1” in the HSC/HST Item Selection under Item 8 “HSC/HST/INT” of FP-07 or PROLADDER “Configuration”. Please refer to the example (to configure HSC1 as HST1) in Section 11.4 “HSC/HST Configuration”. The diagram below is the function structure diagram for HHSC being planned as a HST. Its applications are the same as that of a 16-bit HSTA. Please refer to Section 11.6.4 “Program Examples”.

- Employ FUN92 to read out the current timing value in ASIC chip and store it into the CV register of CPU. So as to let user know the current timing value.
- It may also employ FUN93 to write the content of CV register from CPU into the CV in the ASIC chip so as to reset the timing value to be the CV register value of CPU.



- Apply FUN93 to write the content of PV register into the set point value register in ASIC chip which is served as set point for timing up interrupt.

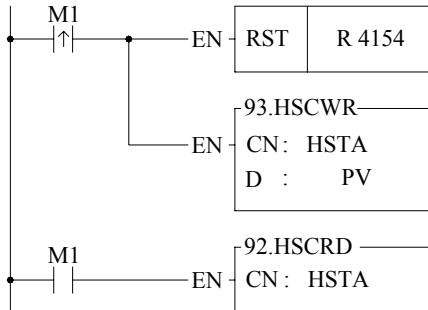


### B. HST0~HST3 32bit Cyclic Timer

According to demand, configured the HHSC(HSC0~HSC3) to be the 32-bit timers of HST0~HST3. For interval of every 0.1ms, the current timing value register in ASIC chip will be increased by 1. User may use FUN92 instruction to read out the current timing value and store it into the CV registers (DR4096, DR4100, DR4104, and DR4108) of CPU. Therefore the content of CV register of CPU become 0, 1, 2, ..... , 7FFFFFFFH, 80000000H, ..... , FFFFFFFFH, 0, 1, ..... etc. variation of values for 32-bit. With the timing calculation technique to count the interval between two events, it can obtain infinite number of 0.1ms 32-bit timers.

### 11.6.3 Examples for Application of High-Speed Timer HSTA

#### Example 1 HSTA serve as 32-bit cyclic timer

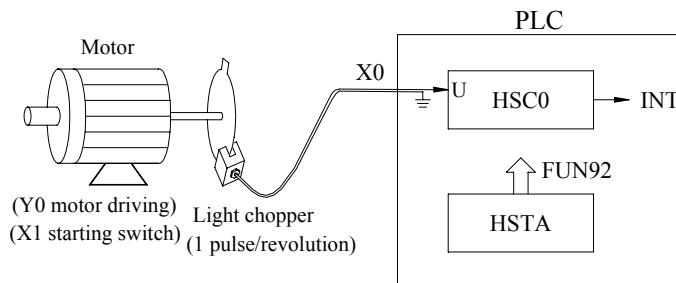


- R4154=0, treat HSTA as 32 bit cyclic timer
- Employ FUN93 to write the preset value into the HSTA PV in the ASIC chip  
CN =4, represents HSTA  
D =1, represents PV
- Employ FUN 92 to read out the current timing value of HSTA in ASIC chip and store it to R4152  
(DR4152 value change from 0,1,2, ……,FFFFFFFFFF,0,1,2,……cyclic variation, the unit is 0.1mS)
- CN =4, represents HSTA

#### Example 2 Application example for cyclic timer

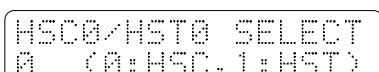
This example uses HSTA as a cyclic timer, cooperating with HSC0, to read the time interval for accumulation of 10 pulses and sending an interrupt out each time as 10 pulses are accumulated and, reciprocally, find out the required RPM (the number of pulses is fixed when the time varies).

#### Mechanism

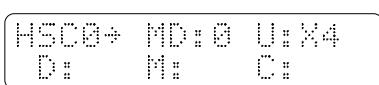


#### HSC and HST Configuration

- ① As HSTA is built in, no configuration is required. Simply make PV (R4154)=0 to make it as a 32-bit cyclic timer.
- ② To cooperate with the photo interrupter, set the HSC0 as an up counting counter having single input (MD0, but use only U input).



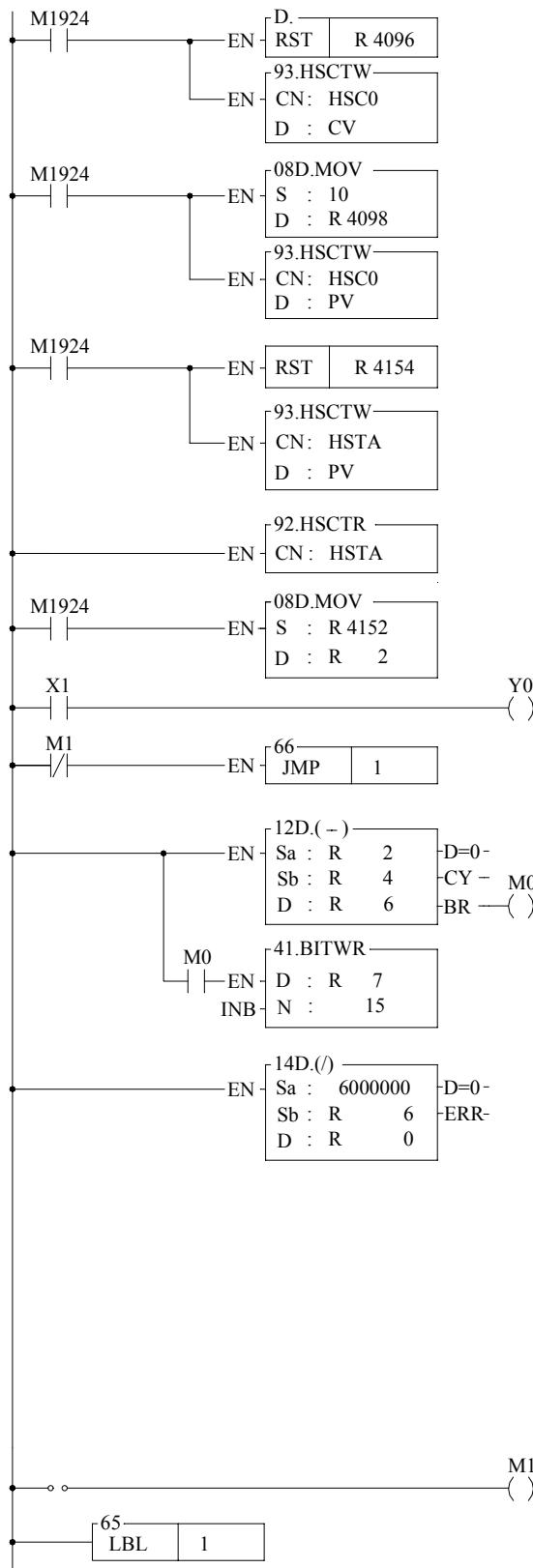
- No change of the preset value is required



- Configure HSC0 as an up counting counter with single input only
- No application of other inputs is required

All other settings (polarity of counting and control inputs) are preset (non-inverse) and should not be changed.

## 【Main Program】



- The initial value of HSC0 CV register is 0
- Write 0(DR4096) into CV register (reset the counts) in ASIC chip; FUN93 CN=0 indicates HSC0 and D=0 indicates CV
- The initial value of HSC0 PV register (DR4098) is 10
- Write 10(DR4098) into the preset register in ASIC chip, which acts as interrupt value for counting up; FUN93 CN=0 indicates HSC0 and D=1 indicates PV
- Make R4154=0
- Write 0(R4154) into the preset register, and HSTA is configured as a 32-bit high-speed cyclic timer; FUN93 CN=4 indicates HSTA and D=1 indicates PV
- Read the current timing value
- The initial value of HSTA CV register is stored to DR2
- Find interval for each HSC0 interrupt

$$\Delta T(\text{DR6} \times 0.1 \text{ out the ims})$$

CV value of HST (DR4152)  $\left\{ \begin{array}{l} 2147483647(7FFFFFFF) \\ -2147483648(80000000) \end{array} \right. \begin{array}{l} \text{-----A} \\ \downarrow +1 \\ \text{-----B} \end{array}$   
 FUN12D  $\left\{ \begin{array}{l} B-A=2147483647(80000001) \\ \text{And BR}=1 \end{array} \right.$

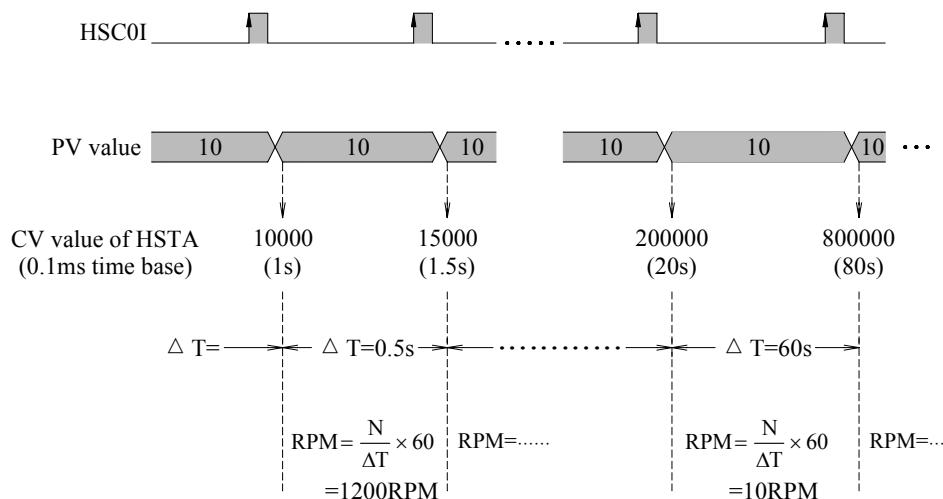
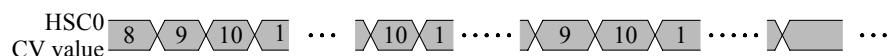
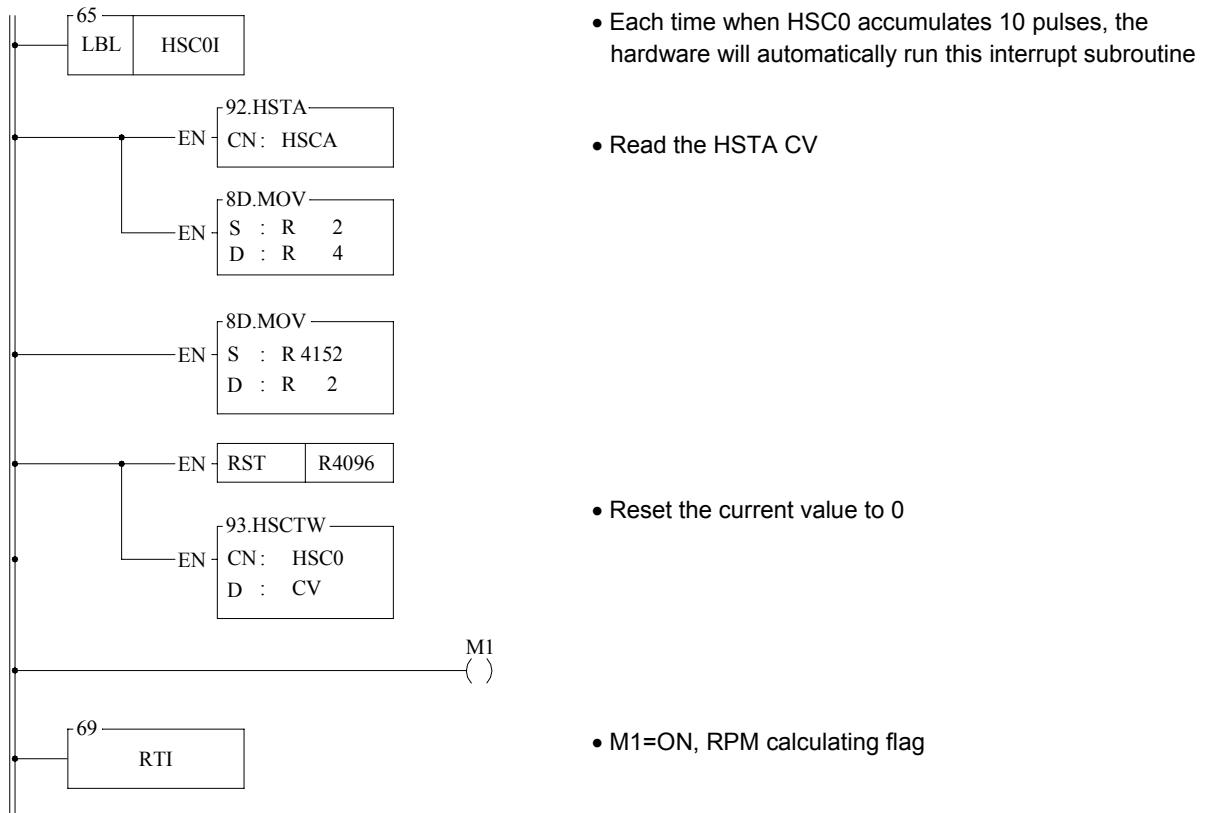
Consequently it need to use FUN12D BR output for DR6 rectification ( $1 \rightarrow b31$ )

- Rotating speed =  $\frac{N}{\Delta T} \times 60 \text{ RPM}$
- $N=10, \Delta T = \Delta CV \times 0.1\text{ms} = \frac{(\text{currentCV}-\text{previousCV})}{10000\text{S}}$
- Therefore rotating speed =  $\frac{6000000}{\Delta CV} \text{ RPM}$

- R0=RPM

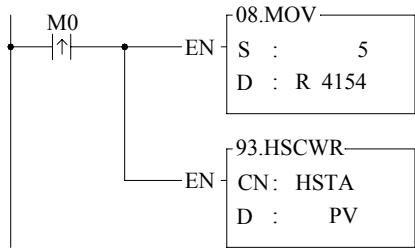
- Clear the calculation flag of RPM

### 【Subroutine】



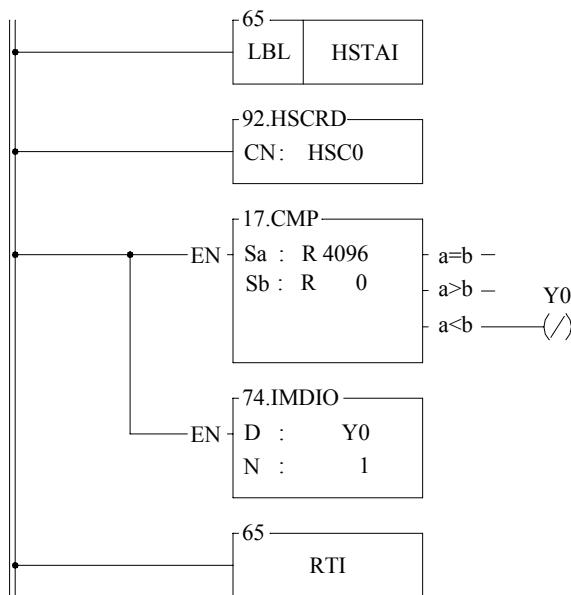
### Example 3 HSTA Serve as Timely Interrupt Timer Program

#### 【Main Program】



- Set up the period of timely interrupt time. R4154=5 represents that it performs the interrupt service subroutine with the label name of HSTA every 0.5mS.
- Employ FUN93 to write the preset value into HSTA PV in ASIC chip, which serve as time up for interrupt preset value. CN =4, represents HSTA  
D =1, represents PV

#### 【Subroutine】



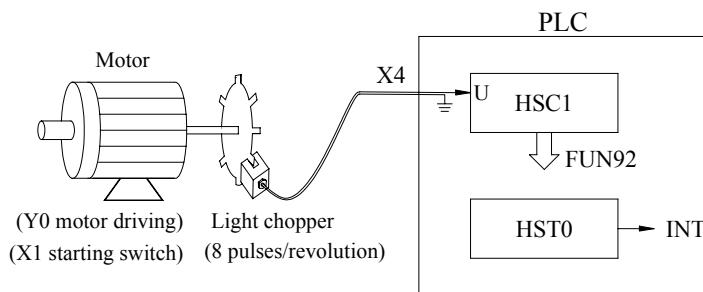
- Interrupt service subroutine with the label name of HSTA.
- Read the current value of hardware high speed counter HSC0 once every 0.5mS.
- To tell whether the current counting value is greater than R0, if yes, then Y0 will be ON.  
※ While reading the current counting value from hardware high speed counter (HSC0~HSC3) in HSTA timely interrupt program, only the 16-bit of Low Word is effective.
- Update output Y0 immediately, so as to reach the high speed output reaction  
(otherwise there will be introduced a delay in scan time)

#### 11.6.4 Examples for Application of High-Speed Timer HST0~HST3

##### Example 1 Application example for delay timer

This example configures HSC0 HST as a HST0 delay timer. At the same time, by connecting the high-speed counter HSC1 with a rotary motor of an automatic wood drilling machine and sending out an interrupt at a fixed period. Each time interrupt occur will read the counting value of the counter. Then, by comparing the change in speed between the number of the motor's rotation when no loading is applied (operating without drilling) and that when the drill head is pressing down (drilling), the change of the motor's RPM can be calculated. It is understood that resistance will be less and motor's RPM will be faster when the drill head is normal (sharp) than when the drill head is blunt. When the drill head is broken, it works like operating without drilling that no resistance exists and RPM is the fastest. Usually the difference in rotating speed among the three conditions is not significant and which cannot be sampled and detected by an ordinary timer having a more than tens of ms of deviation. However, applied with an HST having a time base of 0.1ms that incorporating interrupt, the drill head's status (normal, blunt or broken) can be detected and, thus, warning can be given or operation can be stopped in due time for drill head replacement. 【The time is fixed and the number of pulses varies】

##### Mechanism



##### HSC and HST Configuration

HSC0/HST0 SELECT  
1 <0:HSC, 1:HST>

- HSC0 is set as HST0

HSC1/HST SELECT  
0 <0:HSC, 1:HST>

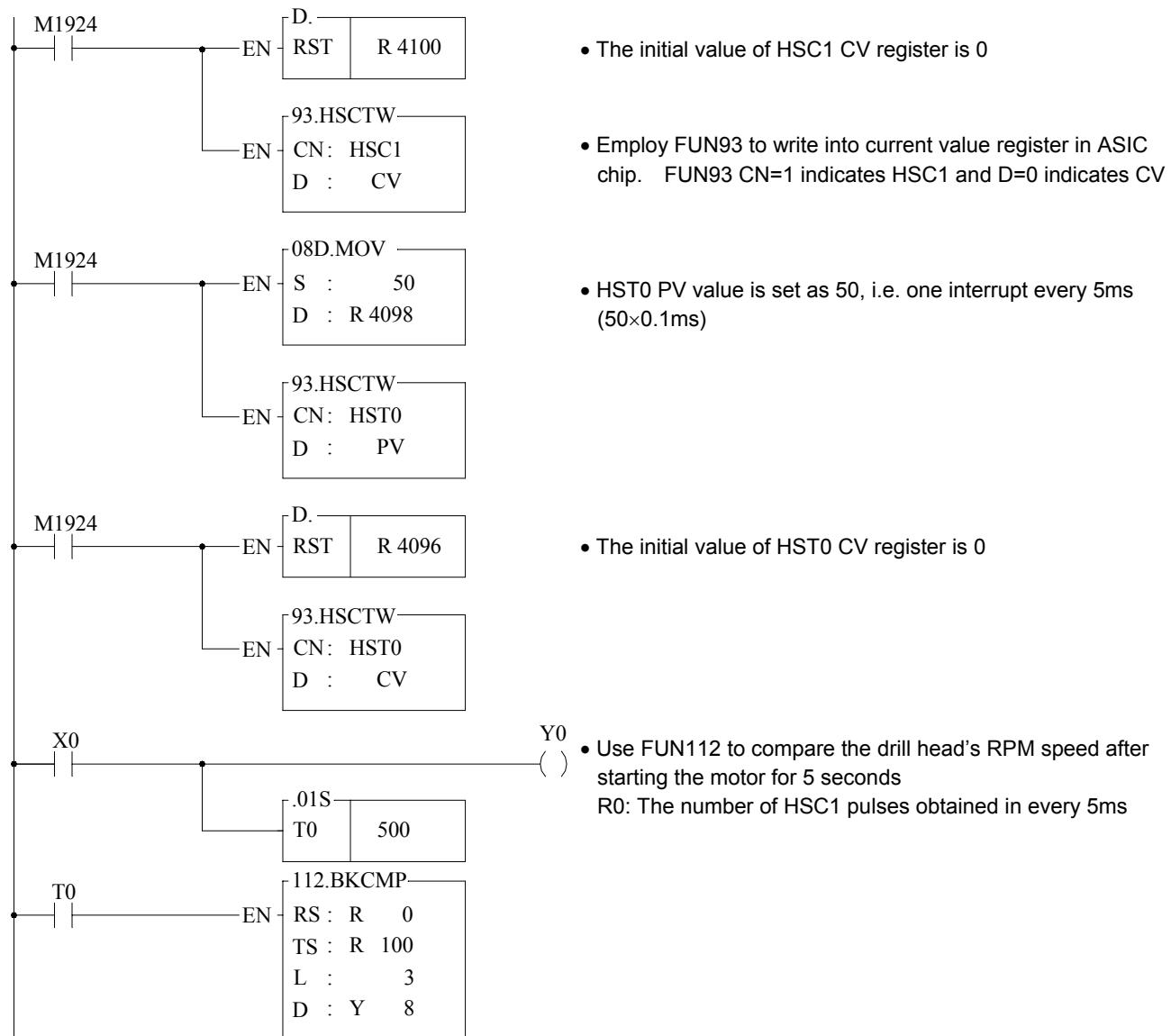
- HSC1 is preset as HSC

HSC1+ MD:0 U:X4  
D: M: C:

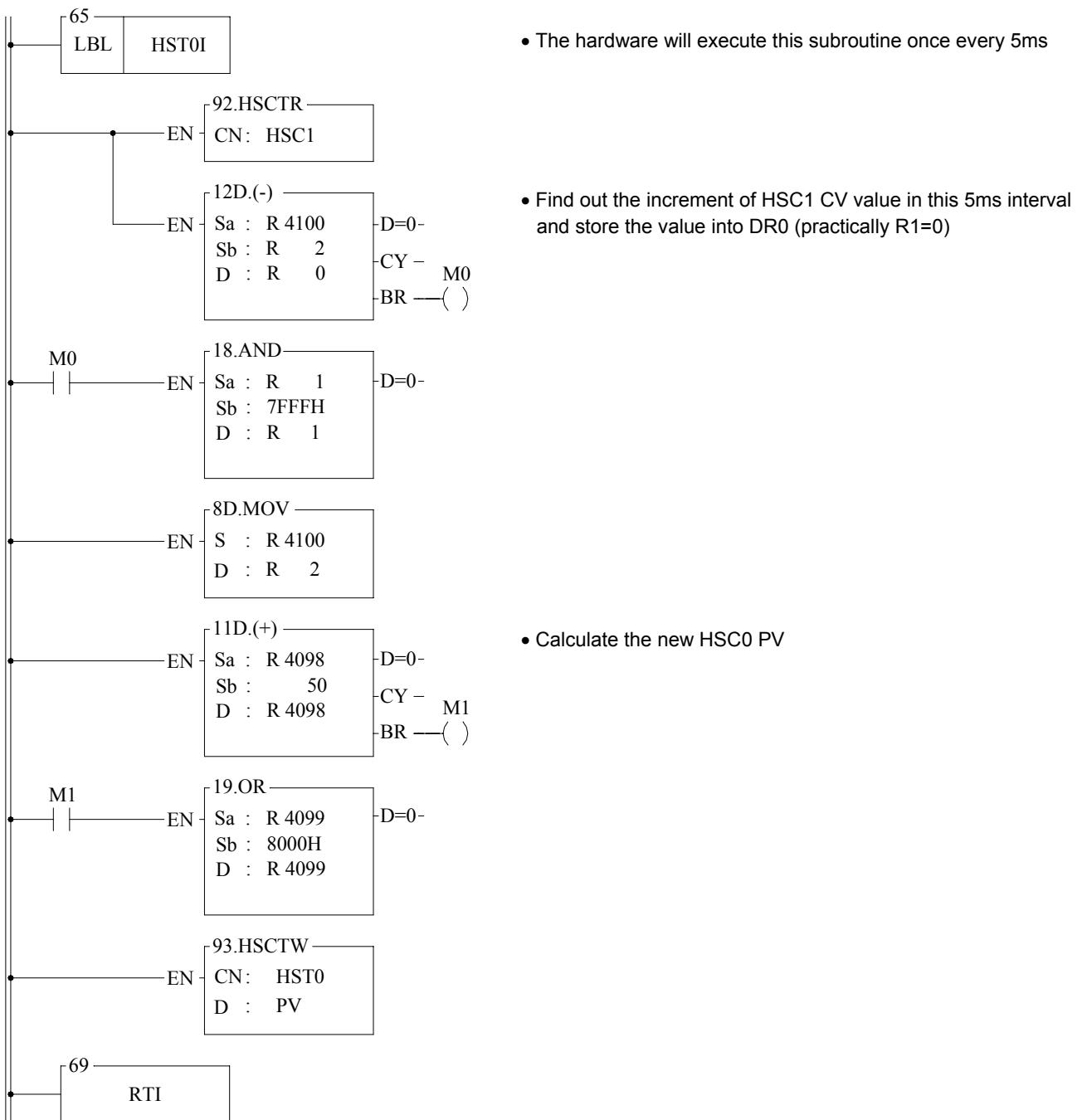
- HSC1 is set as MD0, an up counting counter with single input. Other inputs will not be used.

- All other settings (polarity of counting and control inputs) are default (Non-inverse) and should not be changed.

## 【Main Program】

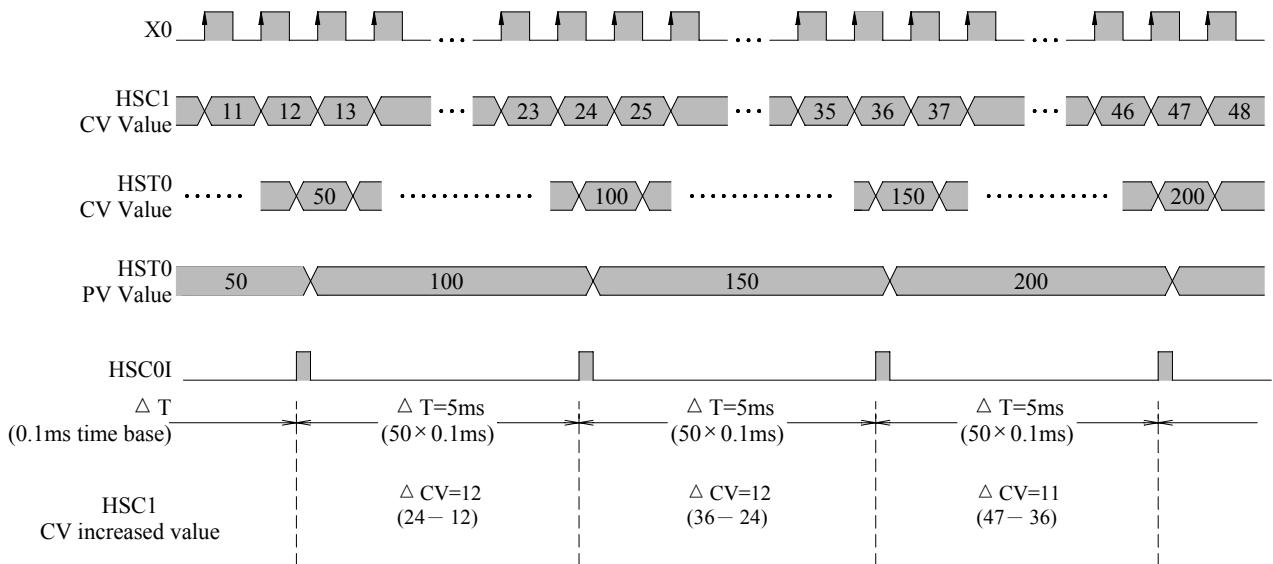


### 【Subroutine】

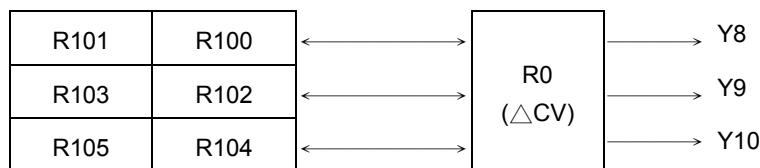


### 【Description】

Supposed that the drill head's normal RPM is 18000rpm and the photo interrupter will generate 8 pulses in one revolution, then the frequency of the pin U of HSC1 is  $18000/60 \times 8 = 2400\text{Hz}$ , i.e. 120 pulses will be generated for every 5ms. Therefore, HST0 can be used to send an interrupt and read the HSC1 CV value every 5ms to get the RPM value.

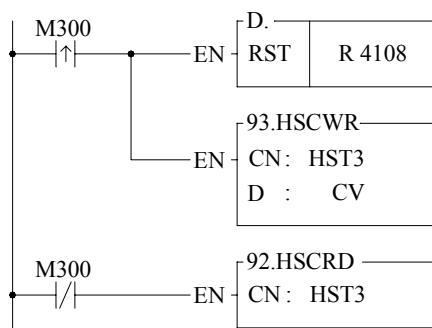


Upper Limit Lower Limit



\* Setting different upper and lower limits to category the RPM condition

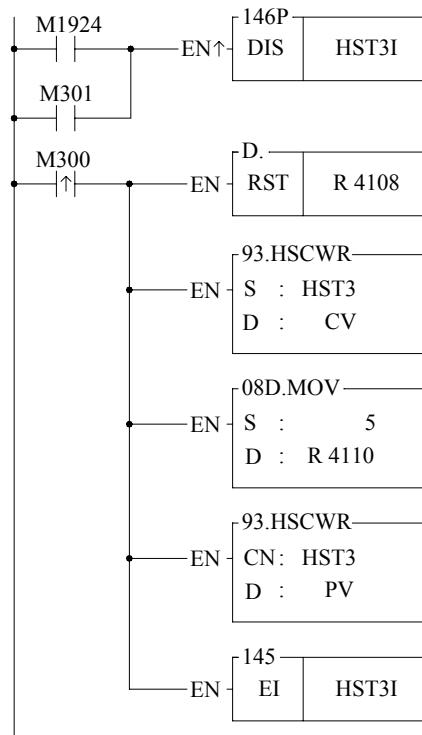
#### Example 2 Hardware high speed timer HST3 serve as 32-bit cyclic timer



- As M300 change from 0→1, clear the current value register to 0
- Employ FUN 93 to write the content of current value register into the HST3 CV (reset) in ASIC chip
  - CN =3, represents HST3
  - D =0, represents CV
- Employ FUN92 to read out the current timing value of HST3 in ASIC chip and store it into the current value register DR4108
  - (DR4108 value cyclically changes from 0, 1, 2, ..... FFFFFFFF, 0, 1, 2, ..... the unit is 0.1mS)
  - CN =3, represents HST3

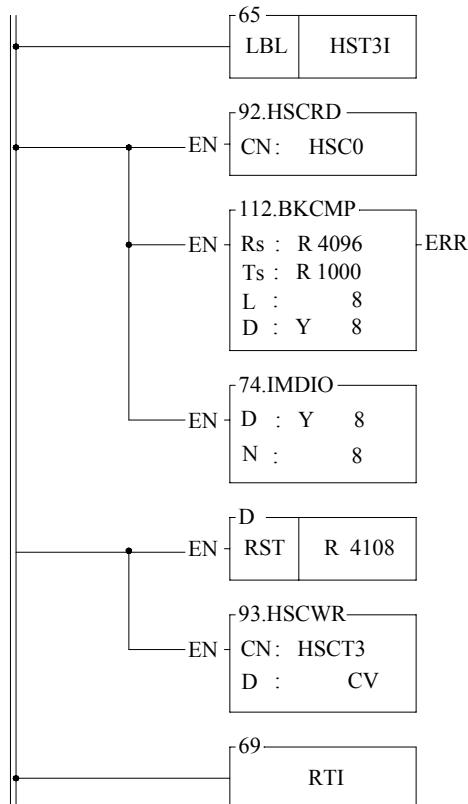
### Example 3 Hardware High Speed Timer HST3 Serve as Periodic Interrupt Timer

#### 【Main Program】



- Turn on or M301 is ON, it prohibits the HST3 from sending periodic interrupt
- As M300 change from 0→1, clear the current register to 0
- Employ FUN93 to write the content of current value register into the HST3 CV (reset) in ASIC chip.  
CN =3, represents HST3; D=0, represents CV
- Set up periodic interrupt interval; DR4110=5 represent every 0.5mS perform once the interrupt service subroutine with label name of HST3I.
- Employ FUN93 to write the content of preset register into the HST3 PV in ASIC chip, which serve as time up interrupt preset value.  
CN=3 represents HST3; D=1 represents PV
- Enable the HST3 interrupt

#### 【Subroutine】



- Hardware high speed Interrupt service subroutine with the label name of HST3I.
- Read the current value of hardware high speed counter HSC0 once every 0.5mS.
- To tell which zone of the electronic drum does the current counting value fall, and set the corresponding output point to be ON.
- Update output Y8~Y15 immediately
- Clear the current value register to be 0
- Employ FUN93 to write the content of current value register into the HST3 CV in ASIC chip (reset).  
CN=3 represents HST3; D=0, represents CV

## Chapter 12 FB-PLC Communication

The main unit of MC model of FB-PLC has three built-in communication ports with the interface of HCMOS, RS-232 and RS-485; that share the 15-pin D-sub female connector located on the left side of the main unit. Please refer to chapter 1 of "Hardware Manual" for the signal distribution details. FB-PLC also provides many communication converters (cables) for the conversion between the interface of RS-232 and RS-485, or between the interface of HCMOS and RS-232 or RS-485. By matching with suitable communication converters, the three built-in FB-PLC communication ports can be converted into one of the three additional configurations, i.e. two RS-232 plus one RS-485, one RS-232 plus two RS-485 or three RS-485.

Note : The main unit of MA model only has one communication port with HCMOS interface.

### 12.1 Functions and Applications of FB-PLC Communication Ports

FB-PLC not only has three different hardware interfaces but also has three types of software interfaces. The table below shows the types of software interfaces that the three FB-PLC communication ports can be defined:

Software Interface	Type Available			Note
	port0	port1	port2	
Standard Interface	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Controlled by the CPU via the "Standard Communication Driver of FATEK"
Modem interface		<input type="radio"/>		Controlled by the CPU via the "MODEM Interface Driver" and "Standard Communication Driver of FATEK"
Ladder Program Control Interface		<input type="radio"/>	<input type="radio"/>	The user controls these ports via ladder diagram program
Setting Method of Interface	—	By SW1	Auto	Please refer to Section 12.6.3

- Standard Interface: This type of interface can be set in port0~port2 (only this type of interface is allowed in port0) and is called "Standard Interface" due to the fact that the port is controlled by the FB-PLC Standard Communication Driver (applying "FB-PLC Communication Protocol"). It is necessary to conform to "FB-PLC Communication Protocol" before connection with "Standard Interface" can be made for communication. This is the system default.
- Modem interface: This type of interface can only be selected in port1 that is controlled by the FB-PLC built-in "MODEM Interface Driver" for receiving phone calls or dialing. After connecting, control will be passed to "Standard Communication Driver"; all operations will be the same as for "Standard Interface" afterwards.
- Ladder Program Control Interface: This type of interface can be selected in port1 or port2. The port under this type of interface is controlled by the handy instructions of ladder program (such as FUN94, FUN 96 and FUN97). Thus, the user may control these communication ports for various communication applications via the ladder program.

The following sections will describe the functions and applications of the three FB-PLC communication ports in each of the three different types of software interface with or without communication converters (cables) being added (called the "Basic Application" or the "Derived Application", respectively).

### 12.1.1 Communication Port 0 (port0): HCMOS (5V) Serial Interface

#### Function Specification

- Communication parameters:
  - Baud Rate : 9600 bps (Default; 19200,38400 also allowed)
  - Data Length : 7 Bits (Fixed)
  - Parity : Even (Fixed)
  - Stop Bit : 1 Bit (Fixed)
- Communication distance ≤ 2 meters

#### Basic Application

The port0 has the hardware interface of HCMOS and the software interface of “Standard Interface” (i.e. applying “FB-PLC Communication Protocol”) and which is mainly used to connect the FP-07 handheld programmer.

#### Derived Application

- ① By adding FB-232P0-xxx-xx cable with converter to convert the hardware interface of port0 into RS-232C interface:  
For connection with peripherals having RS-232C interface, such as PC, MMI, SCADA,....
- ② By adding FB-485P0 or FB-232P0-xxx-xx and FB-485 to convert port0 into RS-485 interface:  
For connection with peripherals having RS-485 interface, such as PC, MMI, SCADA,....  
By this way, it may be connected to the FB-PLC CPU Link Network.

### 12.1.2 Communication Port 1 (port1): RS-232C Serial Interface

#### Function Specification

- The signal specification meets the EIA RS-232C standard and the communication parameters are adjustable. The maximum communication speed can reach 38.4Kbps. The default communication parameters are as below:  
Baud Rate : 9600 bps ; Data Length : 7 Bits ; Parity : Even ; Stop Bit : 1 Bit

#### Basic Application

The following three types of software interfaces can be selected through SW1 (2-pin DIP switch) on the main unit:  
(Please refer to Section 12.6.3 for setting methods)

① RS-232 Standard Interface :

For connection with peripherals having RS-232 interface, such as PC, MMI, SCADA, ....

② RS-232 Modem Interface :

It will provide the remote diagnostics and trouble debugging or remote data collection or alarm and fault report via the Modem interface.

③ RS-232 Ladder Program Control Interface :

The user can control port1 through ladder diagram program, e.g. FUN94 (ASCWR) instruction is to take over the control of port1 and can be connected with the printer having RS-232 hardware interface for report printing; FUN97 (LINK1) instruction is to take over the control of port1 to make point to point connection between FB-PLC or intelligent ASCII peripherals.

### Derived Application

After adding a RS-232↔RS-485 converter (FB-485) to port1, port1 can be converted from RS-232 interface to RS-485 interface and can make multidrop connections. The following two applications are derived:

- ① The FB-485 converter is added under the "Standard Interface" to convert port1 to "RS-485 Standard Interface" :  
For connection with peripherals having RS-485 interface, such as PC, MMI, SCADA,...; or it may be connected to the FB-PLC CPU Link network.
- ② By adding the FB-485 converter under the "Ladder Diagram Control Interface", port1 can be converted into "RS-485 Ladder Program Control Interface" :  
The applications described as bellow:
  - By FUN97 : MD0 instruction, it can be as the master of FB-PLC CPU Link Network.
  - By FUN97 : MD1 instruction, it can be as the ASCII sender and connect with the intelligent peripherals, such as other PLC, servo driver, temperature controller, inverter....
  - By FUN97 : MD2 instruction, it can be as the ASCII receiver and connect with intelligent peripherals, such as magnetic card reader, barcode reader and electronic weighing scale....

### 12.1.3 Communication Port 2 (port2): RS-485 Serial Interface

#### Function Specification

- The signal specification meets the EIA RS-485 standard and the communication parameters are adjustable. The maximum communication speed can reach 614.4Kbps. This port with the ability for real-time distributed control (by FUN96 : MD3, high speed CPU link ). The default communication parameters are as bellow:  
Baud Rate : 9600 bps ; Data Length : 7 Bits ; Parity : Even ; Stop Bit : 1 Bit

#### Basic Application

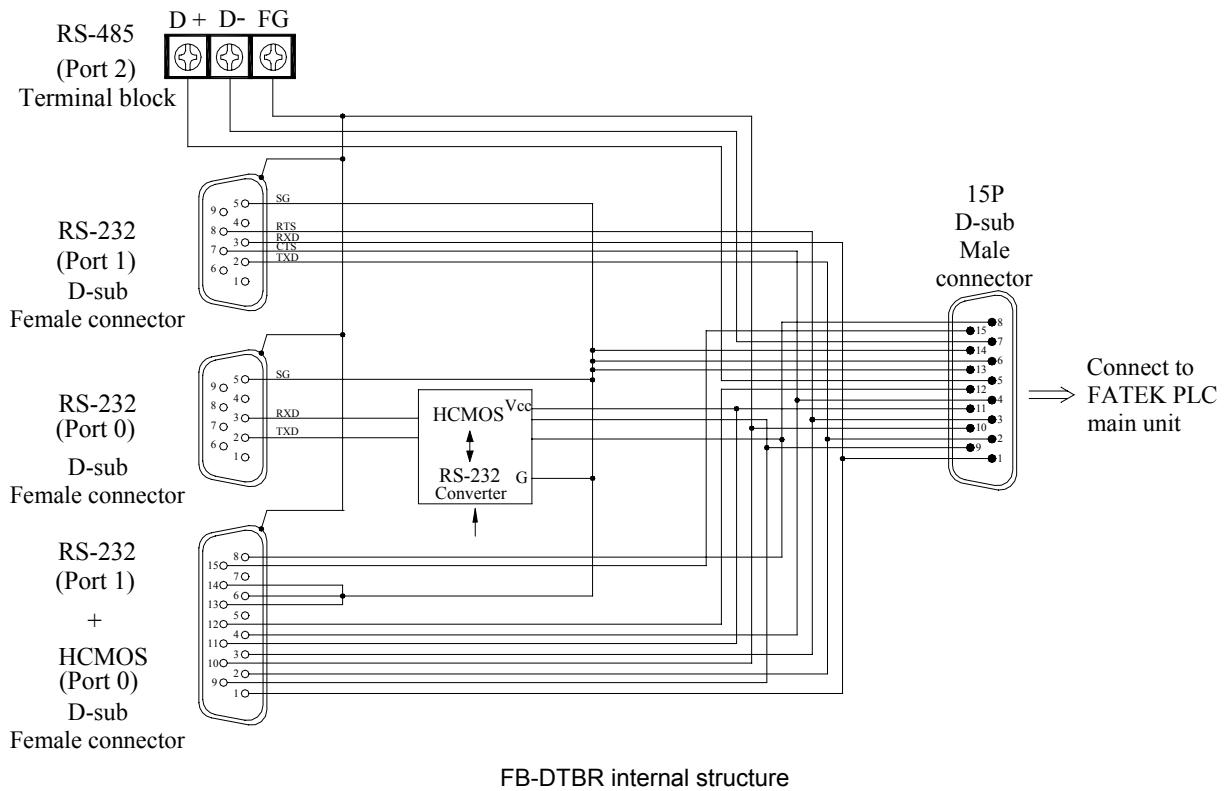
The port2 software interface includes "RS-485 Standard Interface" and "RS-485 Ladder Diagram Control Interface". The type of the software interface is determined by the user's ladder program. This means that when the instruction FUN96 (LINK2) appears in the ladder program and being executed, the CPU will automatically set the interface type of port2 as "RS-485 Ladder Program Control Interface" or, else, "RS-485 Standard Interface".

Details are described as below:

- ① The "RS-485 Standard Interface":  
For connection with peripherals having RS-485 interface, such as PC, MMI, SCADA,...; or it may be connected to the FB-PLC CPU Link network.
- ② The "RS-485 Ladder Program Control Interface":  
The applications described as bellow:
  - By FUN96 : MD0 instruction, it can be as the master of FB-PLC CPU Link Network.
  - By FUN96 : MD3 instruction, it can be as the master of FB-PLC High Speed CPU Link Network.
  - By FUN96 : MD1 instruction, it can be as the ASCII sender and connect with the intelligent peripherals, such as other PLC, servo driver, temperature controller, inverter....
  - By FUN96 : MD2 instruction, it can be as the ASCII receiver and connect with intelligent peripherals, such as magnetic card reader, barcode reader and electronic weighing scale....

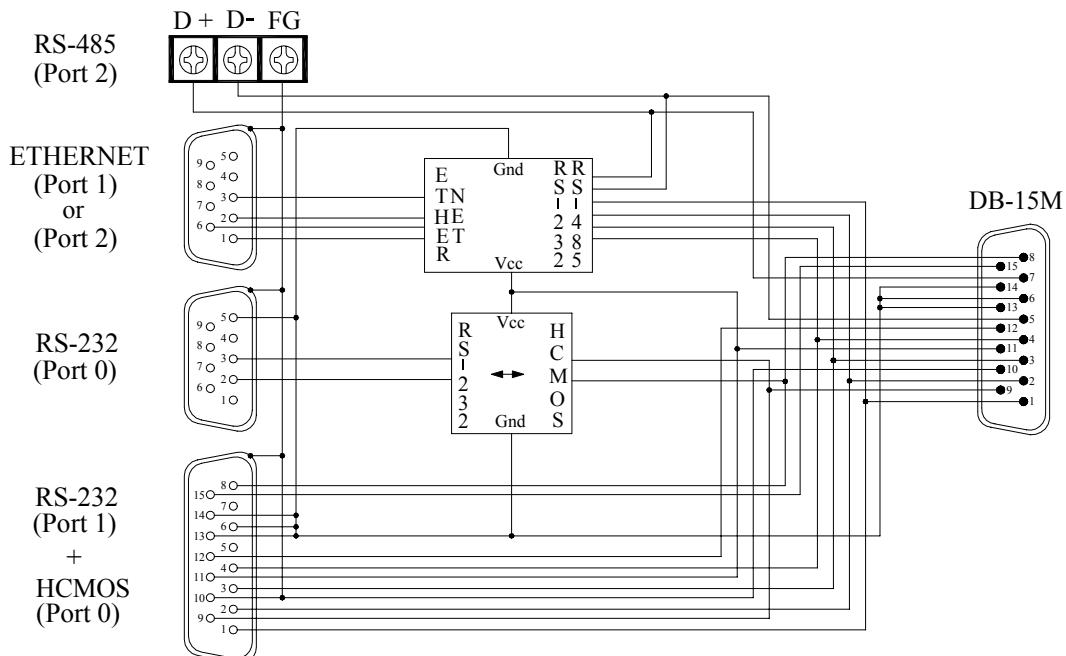
## 12.2 FB-DTBR Communication Distributor

As described previously, the three built-in communication ports of the MC main unit share the 15-pin D-sub female connector located on the left side of the main unit. FB-DTBR is the signals distributor, which is connected to the 15-pin D-sub and distributes the interface signals to the three independent communication connector for easy connection. Please refer to chapter 1 of "Hardware Manual" for the connection between FB-DTBR and the PLC main unit. The diagram below shows the specification and internal circuit of FB-DTBR.



FB-DTBR internal structure

## 12.3 FB-DTBR-E Communication Distributor With Ethernet interface



FB-DTBR-E internal structure

## 12.4 Communication Connectors and Communication Converters of FB-PLC

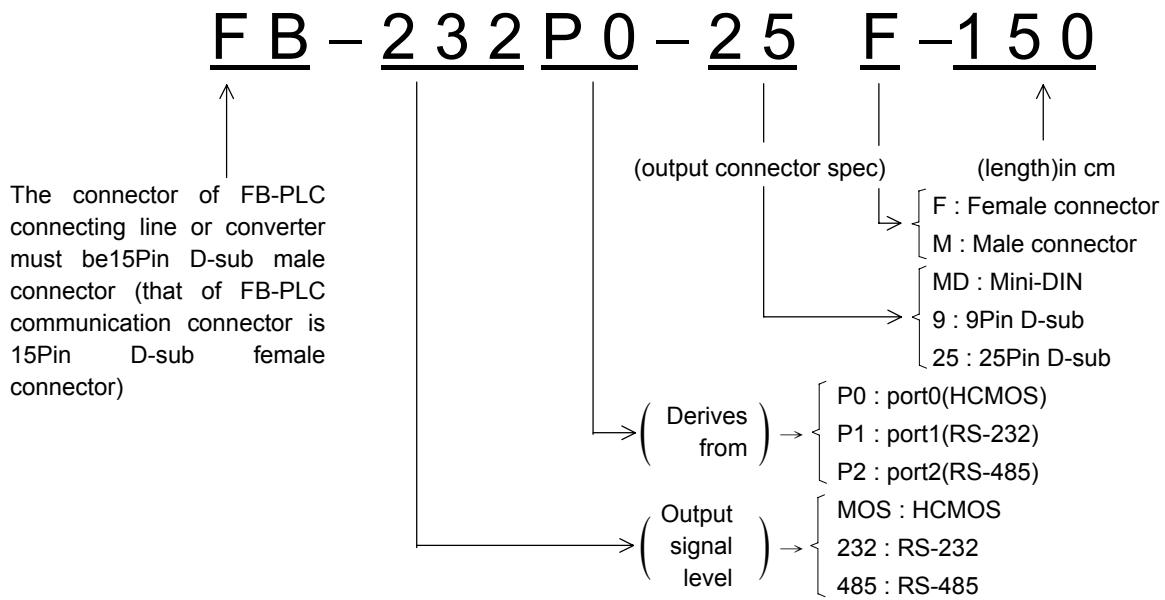
The FB-DTBR has already distributed 3 communication ports to a terminal base and 2 standard 9Pin D-Sub connectors. User may use standard connecting cable available on the market for connection. Given that the pin layout of 15Pin D-sub of FB-PLC is not standard and no commercial connecting cables are available, users who do not apply FB-DTBR may use FB-PLC communication connectors or cables of different specs provided by FATEK. To meet various interface requirements, there are also communication connector or cable with signal level conversion, which we call communication converter or communication cable with converter. The communication connector (or cable) and communication converter (or cable with converter) are identical in appearance. But to make the distinction between them easier, all PVC-coated communication connectors (or cables) are cream color (the same color as the shell of main unit), while converters (or cables with converters) are in dark gray color.

### 12.4.1 Communication Connectors/Cables/Converters/Cables-with-converters

Type	Product No.	Spec Description	Color
Connector or Cable	FB-MOSP0-MD-150	Connecting cable that connects Main Unit and FP-07	Cream (same as that of main unit)
	FB-232P1-9M-150	Communication cable that derives from port1 and connects to 9Pin D-sub male; cable length is 150 cm.	
	FB-232P1-9F-150	Communication cable that derives from port1 and connects to 9Pin D-sub female; cable length is 150 cm.	
	FB-232P1-25M-150	Communication cable that derives from port1 and connects to 25Pin D-sub male; cable length is 150 cm.	
	FB-232P1-25F-150	Communication cable that derives from port1 and connects to 25Pin D-sub female; cable length is 150 cm.	
	FB-485P2	Connector that derives from port2 to 3Pin terminal	
	FB-3EXT-15	Flat cable with 3 extended 15Pin D-sub that transparently derives from the main unit 15Pin D-sub; cable length is 15 cm.	
Converter or Cable with converter	FB-232P0-9M-150	Communication cable that derives from port0 and connects to 9Pin D-sub male (with the signal conversion of HCMOS↔RS-232)	Dark Gray
	FB-232P0-9F-150	Communication cable that derives from port0 and connects to 9Pin D-sub female (with the signal conversion of HCMOS↔RS-232)	
	FB-232P0-25M-150	Communication cable that derives from port0 and connects to 25Pin D-sub male (with the signal conversion of HCMOS↔RS-232)	
	FB-232P0-25F-150	Communication cable that derives from port0 and connects to 25Pin D-sub female (with the signal conversion of HCMOS↔RS-232)	
	FB-485P0	Converter that derives from port0 and connects to 3Pin terminal (with the signal conversion of HCMOS↔RS-485)	
	FB-485	General purpose converter that converts RS-232↔RS422/RS485	

When selecting from communication connector/cable/converter/cable-with-converter listed above, beware of the port (port0, 1 or 2), hardware interface spec (HCMOS, RS-232, RS-485), mechanical specs of connector (model, pin number, male, female) required. The "Numbering rules of communication connector/cable/converter/cable-with-converter" provided in the next section will help you make the accurate selection.

#### 12.4.2 Numbering Rules of Communication Connector/Cable/Converter/Cable-with-converter

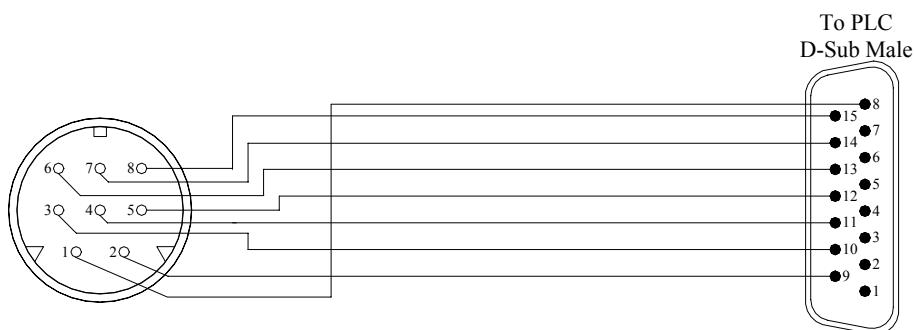


- For product No. that doesn't indicate length, it means there is no cable in its outer appearance. It is called a connector or converter. For example, FB-485P0 is a connector type converter that converts 15Pin D-sub to 3Pin terminal.
- FB-485 is the stand alone general purpose converter for the conversion of RS-232↔RS-422/RS-485, which is not numbered according to the numbering rules presented in this section.

#### 12.4.3 Internal connection of Communication Connector/Cable/Converter/Cable-with-converter

##### A: Communication Connector / Cable

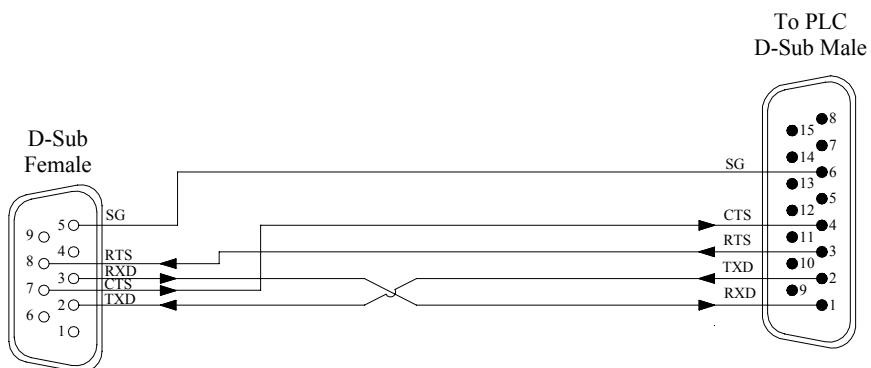
- ① FB-MOSP0-MD-150: The communication cable for connecting main unit and FP-07x ; the cable with cream color and length is 150 cm.



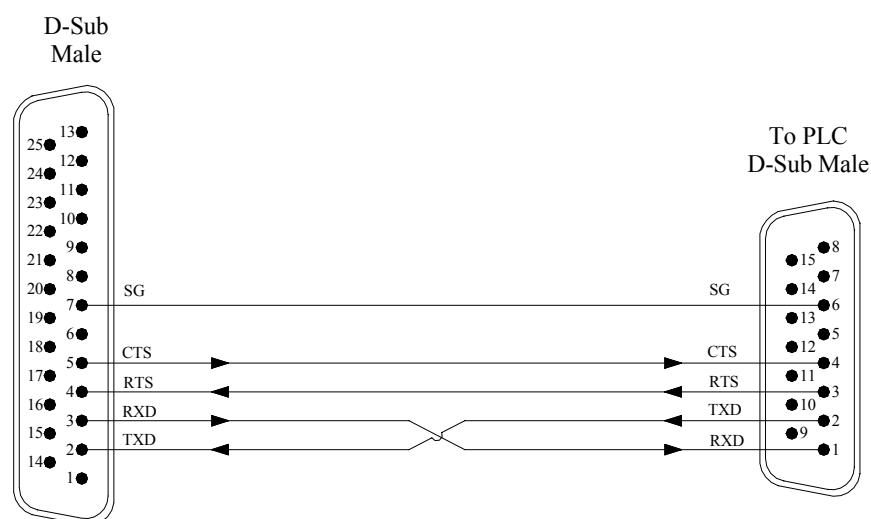
② FB-232P1-9M-150/FB-232P1-9M-30: The communication cable with cream color and length is 150 / 30 cm.



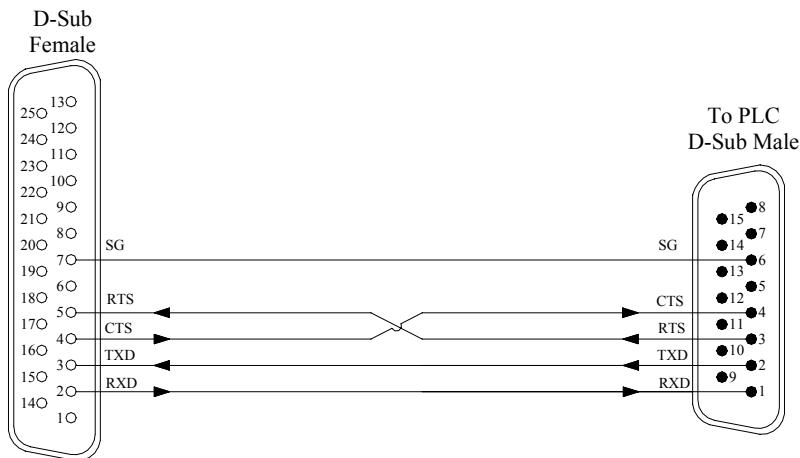
③ FB-232P1-9F-150: The communication cable with cream color and length is 150 cm.



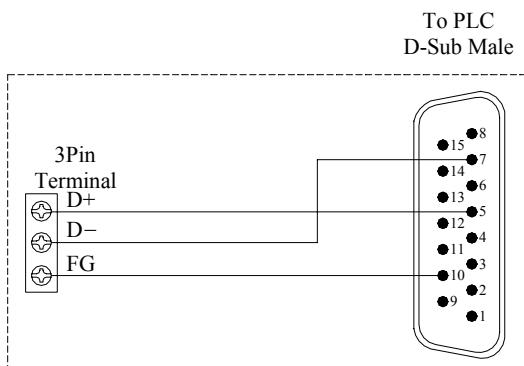
④ FB-232P1-25M-150: The communication cable with cream color and length is 150 cm.



⑤ FB-232P1-25F-150: The communication cable with cream color and length is 150 cm.

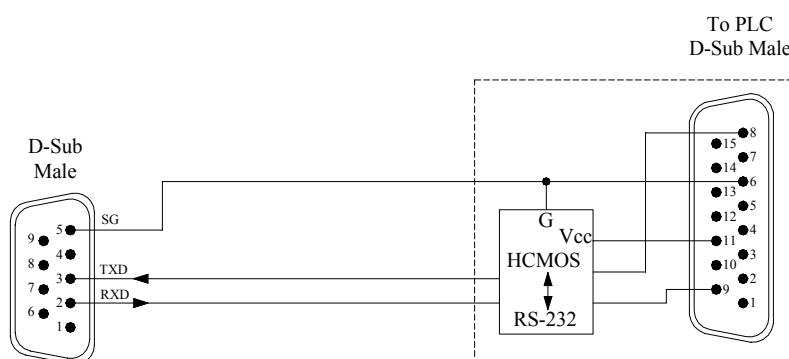


⑥ FB-485P2: The communication connector with cream color.

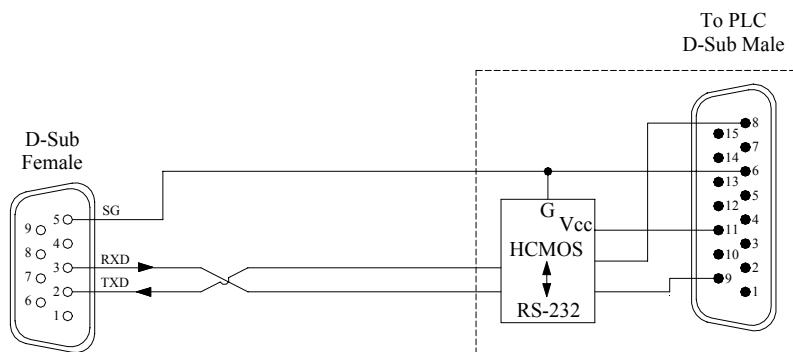


#### B: Communication Converter / Cable-with-converter

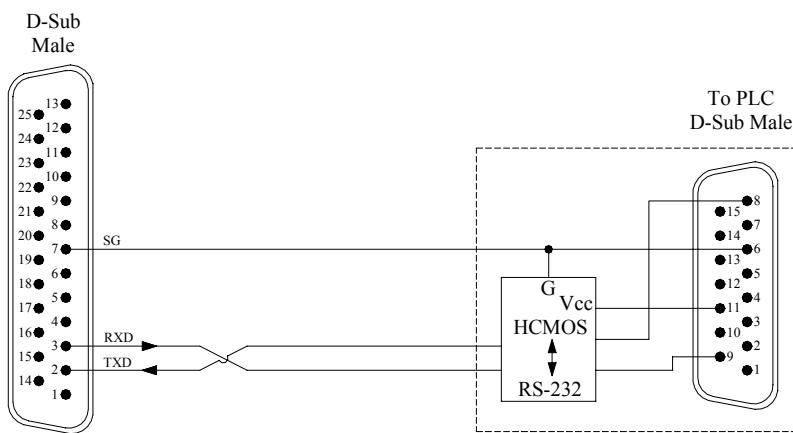
⑦ FB-232P0-9M-150: The communication cable (built in signal conversion) with dark gray color and length is 150 cm.



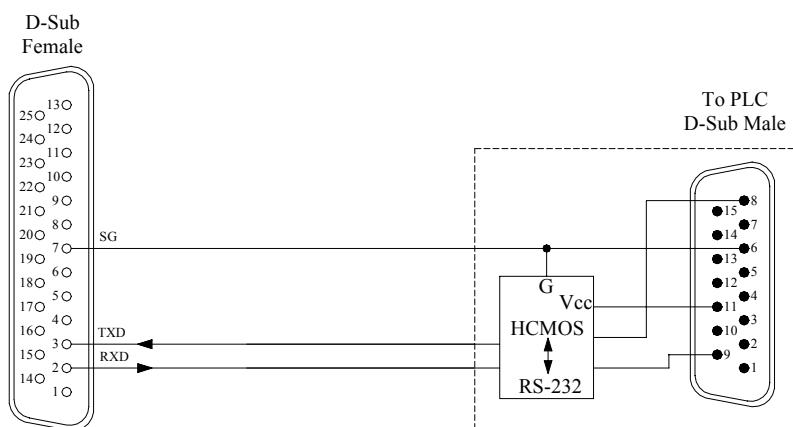
- ⑧ FB-232P0-9F-150: The communication cable (built in signal conversion) with dark gray color and length is 150 cm.



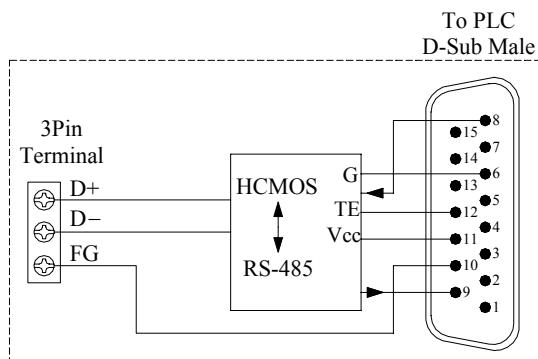
- ⑨ FB-232P0-25M-150: The communication cable (built in signal conversion) with dark gray color and length is 150 cm.



- ⑩ FB-232P0-25F-150: The communication cable (built in signal conversion) with dark gray color and length is 150 cm.



- ⑪ FB-485P0: The communication converter (built in signal conversion) with dark gray color.



### C: User Guide for Self-made Communication Cable

The table below provides an overview of prevailing standard 9Pin and 25Pin D-sub connector and their signal/pin assignment. Users may refer to chapters 1 of "Hardware Manual" on " Pin assignment of 15Pin-D-sub connector of main unit " to make their own communication cable.

Connector type \ Pin		Signal				
		TXD		RTS	CTS	SG
9Pin D-sub	MALE	3	2	7	8	5
	FEMALE	2	3	8	7	5
25Pin D-sub	MALE	2	3	4	5	7
	FEMALE	3	2	5	4	7

- When the definition of pin signal is uncertain, one may apply the following simple method to find the correct pin definition. For self-made RS-232C connection cable, if the definition of pin signal is uncertain, with a DC volt-meter (DVM) to measure the voltage of transmission pin (TXD) and receiving pin (RXD), so as to make connection cable without difficulty.

9Pin connector: The 5<sup>th</sup> pin is for signal ground (SG);

Use the DVM to measure the voltage of pin 2 (red probe) and pin 5 (black probe),  
 if it is around -9V, it indicates that the pin 2 is the transmission pin (TXD);  
 if it is around 0V, it indicates that the pin 2 is the receiving pin (RXD).

Use the DVM to measure the voltage of pin 3 (red probe) and pin 5 (black probe),  
 if it is around -9V, it indicates that the pin 3 is the transmission pin (TXD);  
 if it is around 0V, it indicates that the pin 3 is the receiving pin (RXD).

25Pin connector: The 7<sup>th</sup> pin is for signal ground (SG);

Use the DVM to measure the voltage of pin 2 (red probe) and pin 7 (black probe),  
 if it is around -9V, it indicates that the pin 2 is the transmission pin (TXD);  
 if it is around 0V, it indicates that the pin 2 is the receiving pin (RXD).

Use the DVM to measure the voltage of pin 3 (red probe) and pin 7 (black probe),  
 if it is around -9V, it indicates that the pin 3 is the transmission pin (TXD);  
 if it is around 0V, it indicates that the pin 3 is the receiving pin (RXD).

## 12.5 How to Make Benefit via Communication Functions of FB-PLC

For the connection between FP-PLC and computers, intelligent peripherals and other PLCs, please refer to Section 2-2 of "Hardware Manual" on the diagram of "PLC and Peripheral Systems."

Although all 3 communication ports of FB-PLC can be converted to RS-232C or RS-485 interface, it's advised that "Basic Applications" described in Section 12.1 be followed in their actual application (i.e. without the addition of communication converter or cable-with-converter). This is the most economical way. The installation of communication converter or cable-with-converter may be considered when basic applications do not meet the needs.

Of the three communication ports, only port2 with the ability to make very fast real-time response (i.e. communication data are processed promptly after receiving/sending without being delayed by scan time). Its speed can reach up to 614.4Kbps and it adopts RTU code which is 100% faster than ASCII codes. Port0 and Port 1 use ASCII codes for communication with maximum speed at 38.4bps. Their received/sent communication data will not be processed until housekeeping following ladder program scanning. Thus data transmission is delayed by the duration of scan time. Thus in application, port2 may be reserved for "High-speed CPU Link Network" (via FUN96:MD3 instruction) to meet the requirement of distributed real-time control, while port0 and port1 are applied of data collection and monitoring in connection with intelligent peripherals, man-machine interface and graphic supervising.

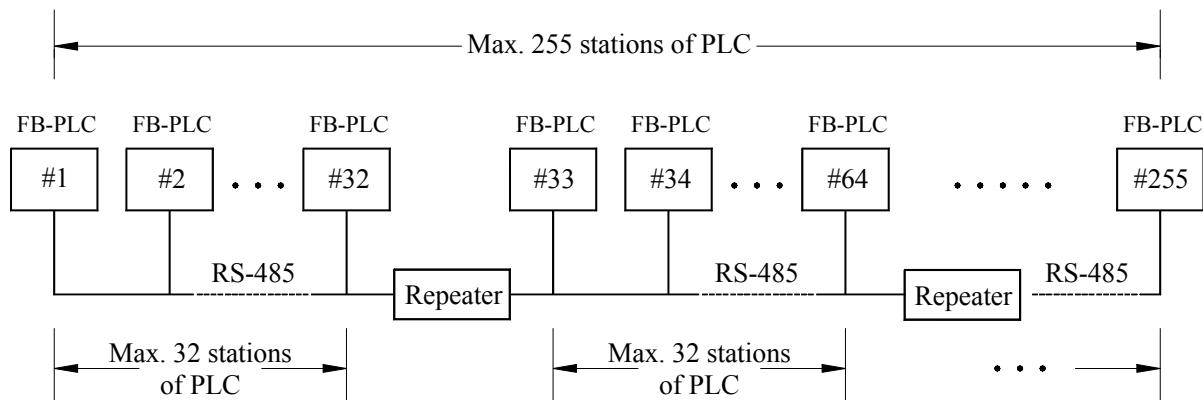
## 12.6 Notifications in arrangement of RS-485 Communication

Of the three communication interfaces of FB-PLC, HCMOS and RS-232C interfaces can only make point to point connection, while RS-485 can have multdrop connections. For connection distance, RS-232 and RS-485 shall observe EIA standard, while HCMOS is limited to 2 meters.

The three communication interfaces should observe the basic principle of short connection and distant from high-noise source, because port0 and port1 are point to point connection with short connection length. The commercially available standard communication cable or that provided by FATEK will serve the purpose. But for high-speed RS-485 network which has a variety of issues to be dealt with, including high speed, long distance, great signal attenuation, multiple stations, plus poor grounding, noise interference, terminal impedance mismatching, and incorrect network topology, it is prone to have communication quality problem if the arrangement of connection is not carefully handled. Thus points to note in the hardware arrangement of RS-485 network are itemized in this section.

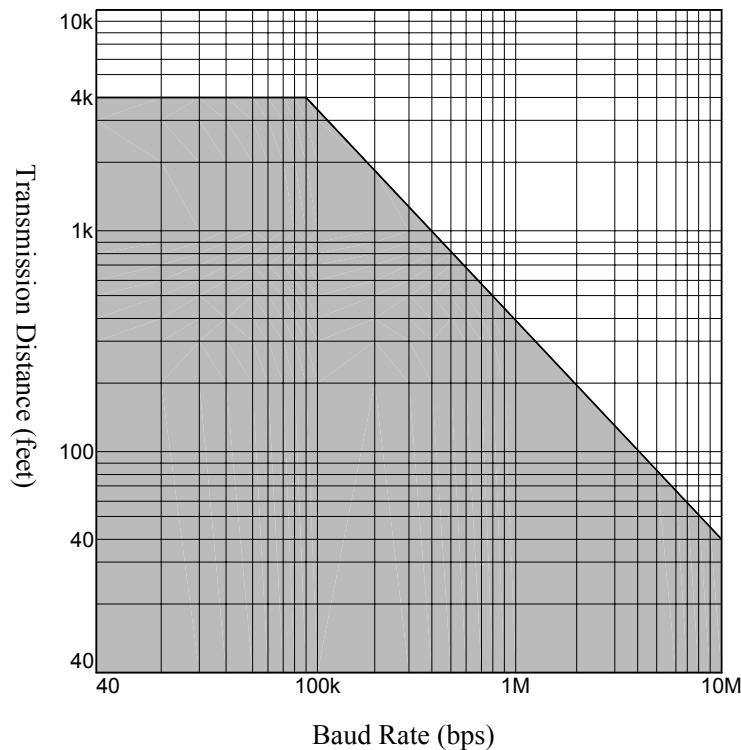
### Station Number Limitation

It is allowed to have maximum 255 stations of PLC connecting in the CPU link network via the RS-485 interface, but the hardware driving capacity of the RS-485 transceiver using in FB series products is limited not to exceed 32 stations. So the repeater is necessary between the group which connecting less than 32 stations of PLC. The diagram below shows the limitation and connection:



## Distance limitation

Below is a graph of relation between baud rate and transmission distance of RS-485 standard interface.

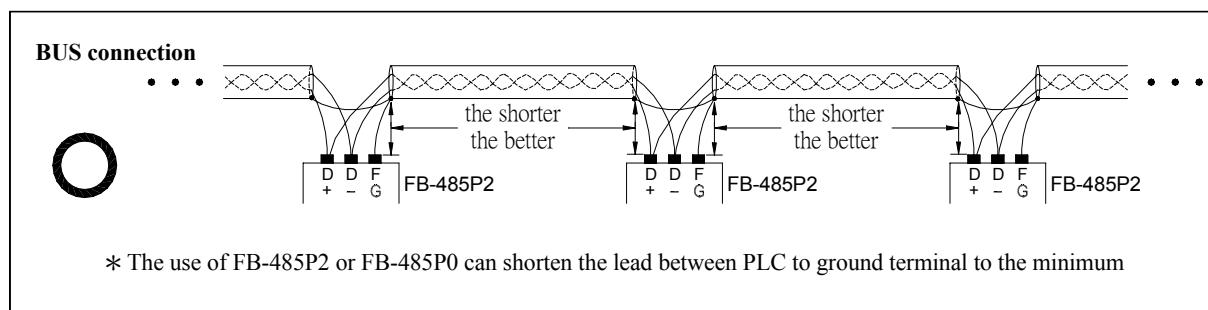


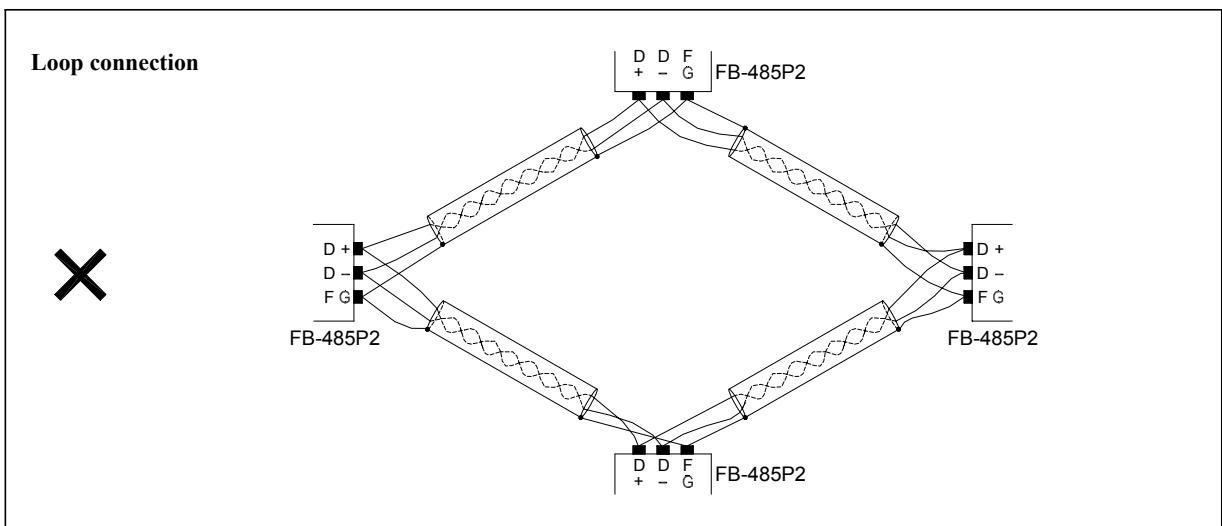
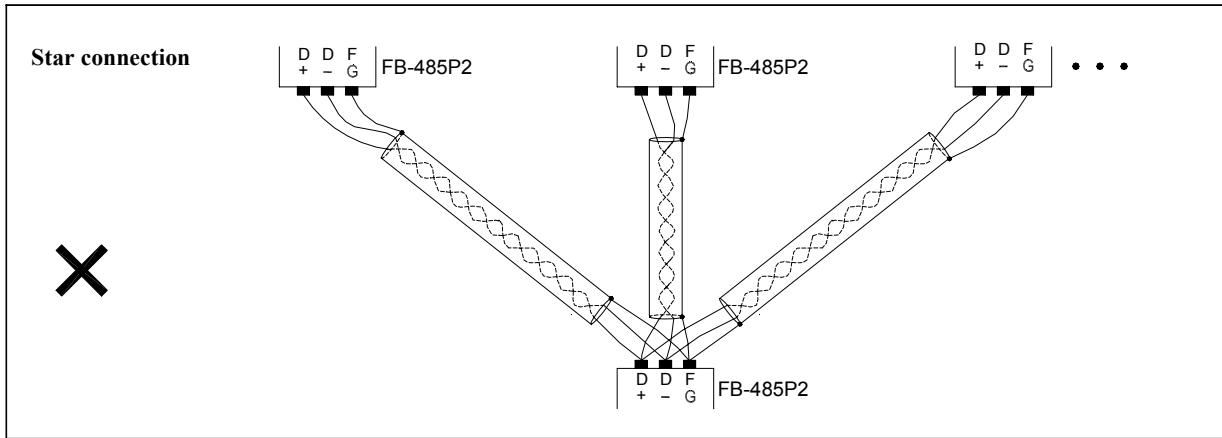
## Transmission cable

The twisted pair cable with shielding is strongly recommended. The quality of transmission cable has tremendous impact on transmission signal. Poor-quality twisted pair (such as PVC-medium) has great signal attenuation that tends to shorten the transmission distance and is susceptible to noise interference. In the occasions of high baud rate, long distance or great noise, high-quality twisted pair is recommended (polyethylene-medium, such as Belden 9841) because the medium loss of the PVC twisted pair will be as much as 1000 times. But in low baud rate occasions, PVC twisted pair is an acceptable and economic alternative.

## Connection topology

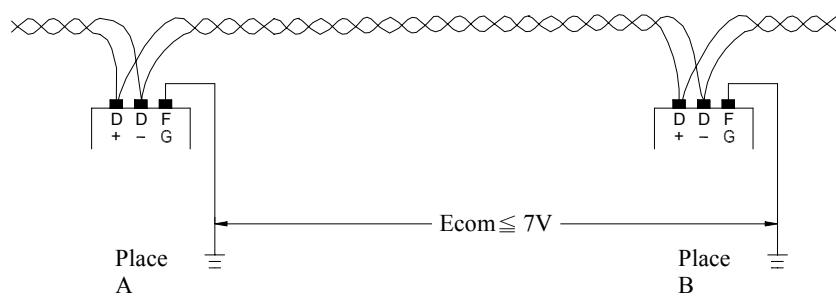
Topology refers to the graphic structure of transmission connection. The connection of RS-485 network must be in bus topology. That is all transmission cables are connected from first station to second station, then from second station to third station, and so on. As shown below, neither star connection nor loop connection is allowed.





#### FG grounding

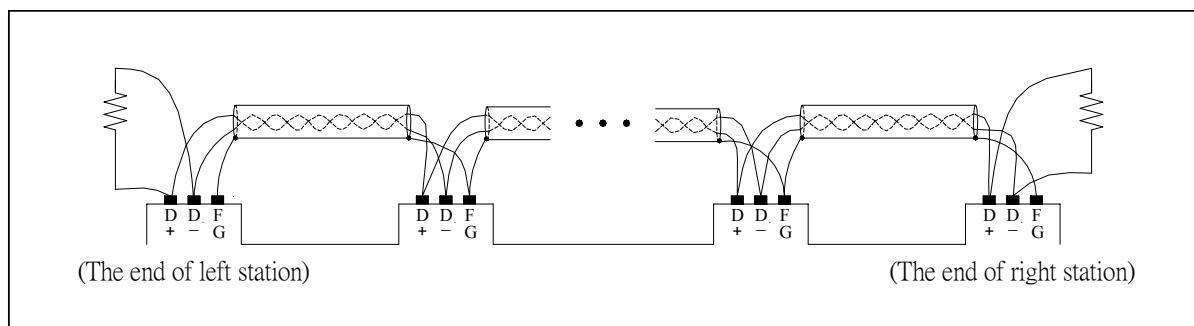
Althoug 2-wires twisted pair cable is enough for the connection of RS-485 network, but it is susceptible to noise interference and it must have the prerequisite that the ground potential difference (common-mode potential) between any two stations may not exceed the maximum allowable common-mode potential of transceiver IC of RS-485 interface. The allowable common-mode potential of FB-PLC should not exceed 7V. Otherwise, the RS-485 network won't be able to work normally.



The twisted pair with shielding is the best suggestion for the wiring of RS-485 network, regardless of the ground potential, that connects the FG of each station (as shown in "Connection Topology" above) via the shielding to eliminate common-mode potential and provide the shortest loop for transmission signals to effectively enhance noise immunity.

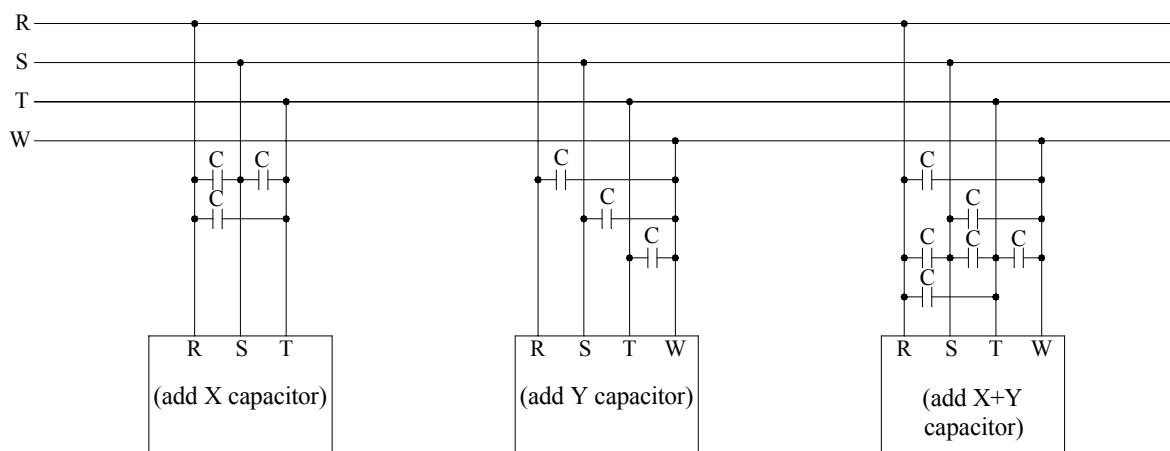
## Terminator

All transmission circuits have their characteristic impedance due to the presence of various transmission cables ( $120\Omega$  in the case of twisted pair). When signals are transmitted toward the end, reflection occurs when terminal impedance differs from the characteristic impedance that leads to distortion of waveform (depressed or raised). Such distortion is not obvious in short transmission. But as transmission cable increases in length, the distortion gets worse to the point that the transmission becomes inaccurate. In such event, terminators should be installed at left and right terminals of bus. Most of applications, the terminators are not necessary for FB-PLC, but when the communication is not good enough because of long distance and wiring of poor twisted pair, it may try to install the terminators at left and right terminals of bus (at D+ and D-) with resistor  $120\Omega(1/4w)$  as shown below. It is the last solution for FB-PLC to improve the communication quality that lets the jumper (JP2) be open of the main unit except the most left and right one's if the terminators are installed but still poor communication.



## Actions against noise interference

If interference persists after RS-485 connections are arranged by the materials and rules recommended earlier, it's an indication that there are many noise sources in the vicinity of RS-485. Aside from keeping transmission cables as far from noise source as possible (such as electromagnetic valve, inverter, servo driver or other power devices), adding noise compressor to noise source is the most effective means of interference control. Please refer to Section 6.3.6 of "Hardware Manual" for related descriptions. Below is a diagram illustrating the methods of noise control over inverter, servo driver or other high noise-power equipment (the addition of X capacitor, Y capacitor or X+Y capacitor).



$$C = 0.22\mu\text{f} \sim 0.47\mu\text{f} / \text{AC630V}$$

## ⚠ Attention

- The wiring of communication network, the addition or removal of stations must be carried out under PLC power-off condition. Avoid power on operation, particularly when PLC is running. Otherwise, inaccurate output of PLC may result.

## 12.7 How to Use FB-PLC Communication Port

The basic elements for communication equipments are that the ① hardware interface and mechanism ② communication parameters ③ software interface (communication protocol) of the sending/receiving equipment must be consistent. The same goes for PLC. PLC can communicate with PLC or other peripherals after the aforesaid three basic elements are satisfied.

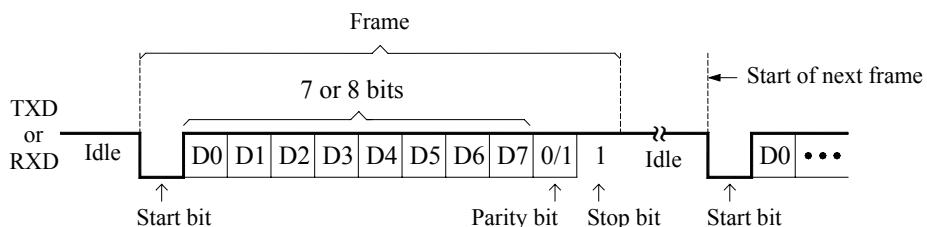
### 12.7.1 Matching with Hardware Interface and Mechanism

The peripheral interface which FB-PLC can communicate with include FP-07 specific HCMOS interface and other two EIA standard interfaces (RS-232C and S-485). As described in Sections 12.2 and 12.3, you can select FB-DTBR communication distributor or communication connector/cable/converter/cable-with-converter to match up with any peripheral hardware interface and mechanism desired. Or it's not enough; you can make your own communication cables according to the instructions provided in Section 12.2.3.

### 12.7.2 Setting of Communication Parameters

Of the three (port0,1and 2) FB-PLC communication ports, the user may set the communication parameters. The default settings of these three ports are identical as below:

Baud Rate	9600 bps
Data Length	7 Bits
Parity Check	Even
Stop Bits	1 Bit



● Data format of asynchronous serial interface

#### A: Setting of port0 communication parameters

Port 0 communication parameters are set by the content of R4050 special register. R4050 is divided into high byte (B15~B8) and low byte (B7 ~ B0). If the content of high byte is not equal to 55H, the communication parameter is the default, regardless of the content of low byte (The default of R4050 is 0000H). Only when the content of high byte of R4050 is equal to 55H will PLC set port0 communication parameters according to the definition of low byte; and only the baud rate of communication parameters may be changed as below:

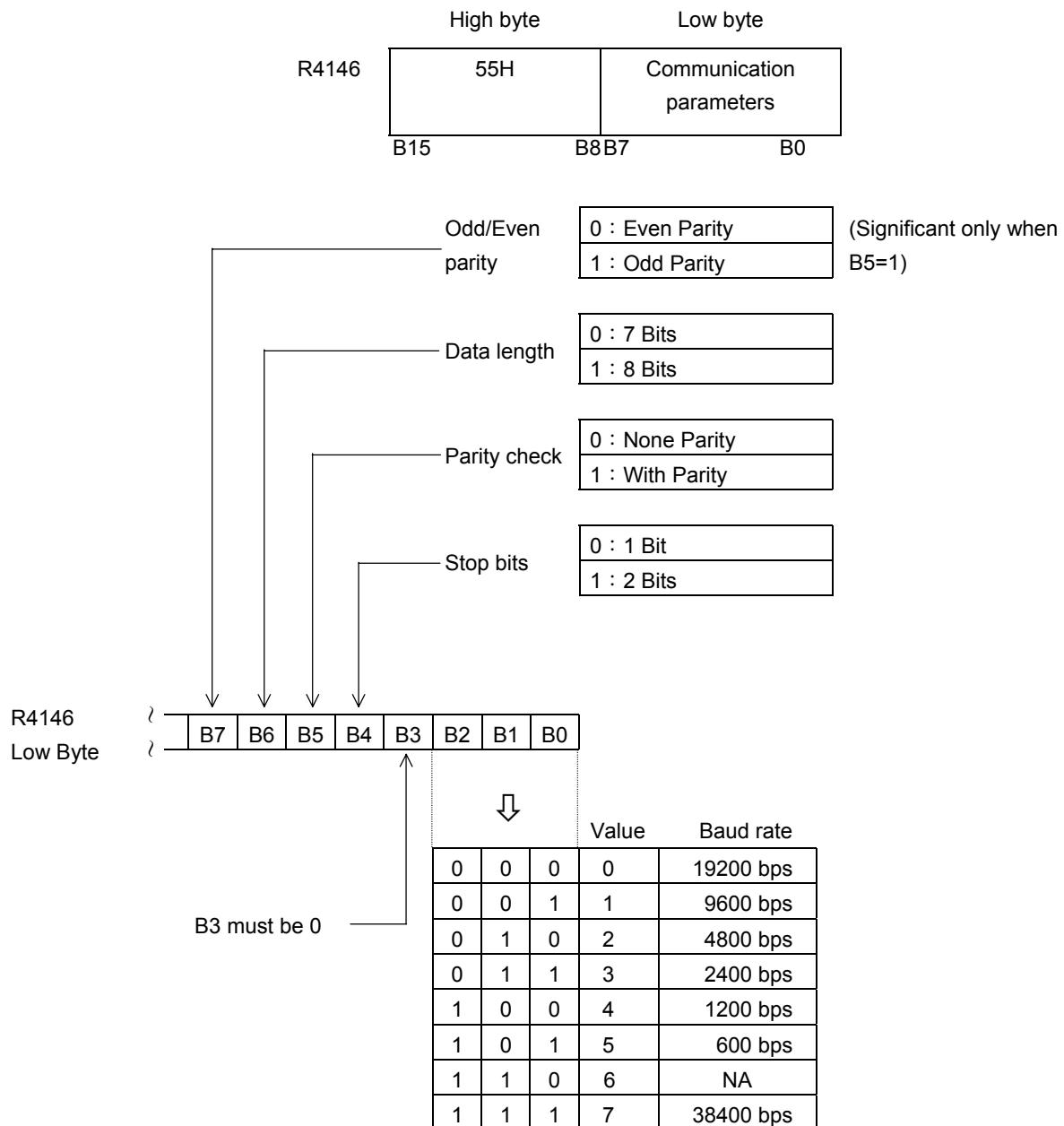
While R4050 = 5500H, Baud Rate=19200 bps  
= 5501H, Baud Rate=9600 bps  
= 5502H, Baud Rate=38400 bps

The other parameters are fixed as follows:

Data Length : 7 Bits ; Parity Check : Even ; Stop Bit : 1 Bit

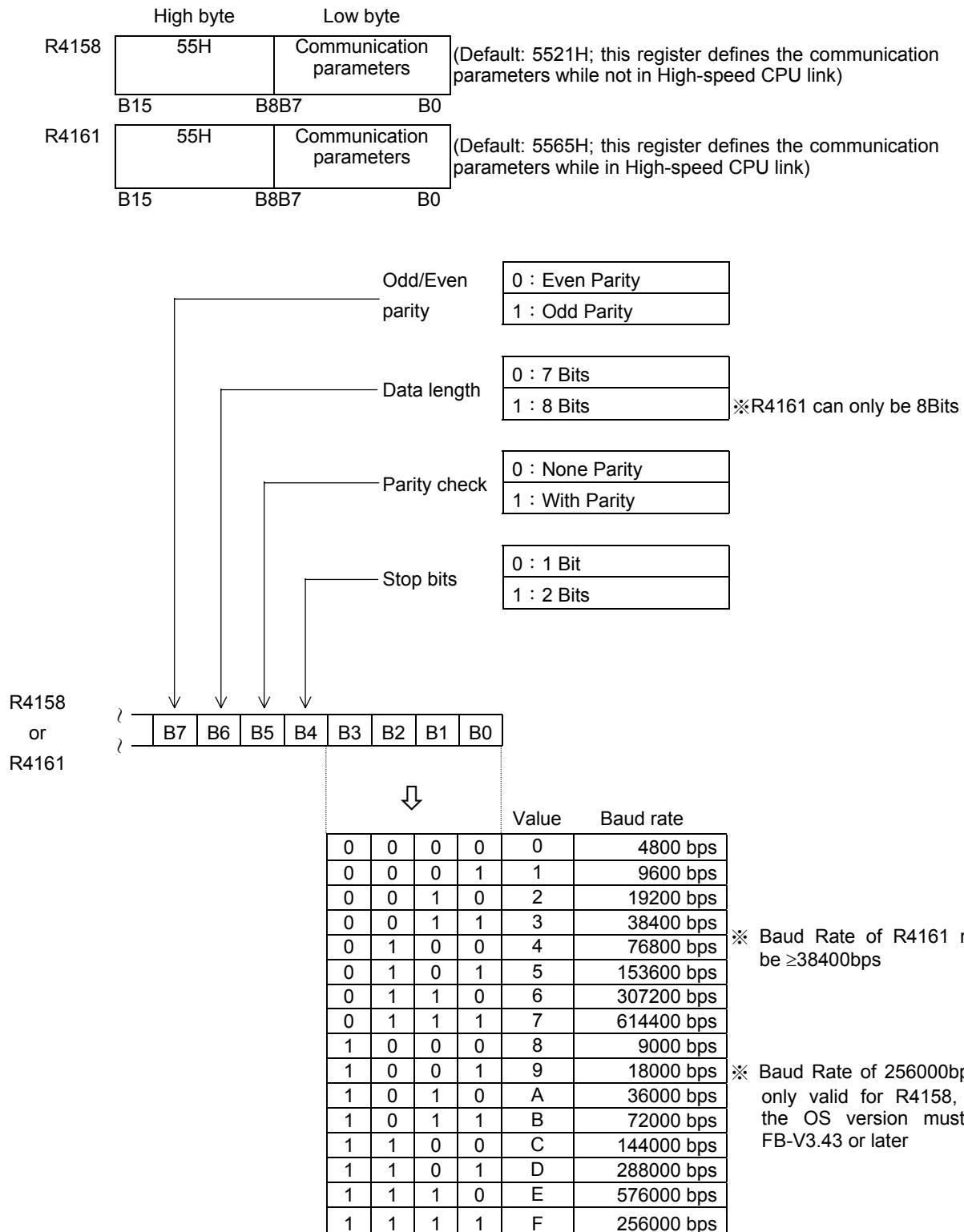
## B: Setting of port1 communication parameters

Port 1 communication parameters are set by the content of R4146 special register. R4146 is divided into high byte (B15~B8) and low byte (B7 ~ B0). If the content of high byte is not equal to 55H, the communication parameters of port1 are automatically set at the default, regardless of the content of low byte (The default of R4146 is 0000H). Only when the content of high byte of R4146 is equal to 55H will PLC set port1 communication parameters according to the definition of low byte. The figure below shows the definition of low byte.



### C: Setting of port2 communication parameters

Port2 communication parameters can be adjusted according to the content of special register R4158 (not in High-speed CPU link) and R4161 (in High-speed CPU link) respectively. When high byte (B15 ~ B8) of R4158 or R4161 is 55H, the content of low byte (B7 ~ B0) determines the communication parameters. If the content of R4158 high byte is not 55H, the communication parameters are the default as mentioned before. If the content of R4161 high byte is not 55H, or data length is not 8 bits or baud rate is less than 38400bps, R4161 is automatically adjusted to default value 5565H (it means Baud rate:153600bps ; Data length: 8 Bits ; Parity check: Even ; Stop bit:1 Bit).



### 12.7.3 Setting of Software Interface

As described in Section 12.1, the communication ports of FB-PLC have three types of software interface. That of port0 is "Standard Interface" only; port2 has "Standard Interface" and "Ladder Diagram Control Interface", the type of interface is determined by the user's ladder program, it means that when the instruction FUN96 (LINK2) appears in the ladder program and being executed, the CPU will automatically set the interface type of port2 as "RS-485 Ladder Diagram Control Interface" or, else, "RS-485 Standard Interface". Thus of the three communication ports, only port1 requires software interface setting which is achieved by the SW1 DIP switch on MC main unit.



Software interface	Switch position		Description	Remark
	Bit1	Bit2		
0	OFF	OFF	Designate port1 as " <b>Standard Interface</b> "	Default setting
1	ON	OFF	Designate port1 as " <b>Modem Interface</b> "	
2	OFF	ON	Port1 as " <b>Ladder Program Control Interface</b> "	
3	ON	ON	Initialize the communication parameters of port0 by default	

\*The setting of this switch is valid only while power off

## 12.8 Description and Application of Software Interface

### 12.8.1 Standard Interface

Standard interface of communication port is controlled by the PLC system. The communication transactions are managed by the "Standard Communication Driver" (i.e. FB-PLC Communication Protocol). Any access to the port must comply with the format of said protocol before PLC responds, including start of text, station number, command code, text body, error-check code, end of text. For details, please refer to "FB-PLC Communication Protocol". FP-07, Proladder and many MMI and graphic supervising softwares have the communication drivers that comply with this protocol. When hardware interface and communication parameters are conforming, PLC may communicate with these peripherals mentioned above via the "Standard Interface" communication port.

### 12.8.2 Modem Interface

This type of interface can only be selected in port1. Although CPU still manages the communication transactions of port1 by "Standard Communication Driver", it has to go through modem, for dialing or receiving. Prior to the initiation of communication, port control is managed by "Modem Interface Driver" and PLC cannot be accessed. Once the connection between the modems are successful, port control is turned over to "Standard Communication Driver" and port1 enters into "Standard Interface". This section will describe the operation of modem connection for active dialing and passive receiving.

Under modem interface, MC main unit can, according to the setting of phone number registers (R4140 ~ R4145), select to dial to a distant modem or receive a call from a distant modem through RS-232C interface of port1. Once the connection becomes successful, data can be sent or received through phone line. Below is a description of the two modes.

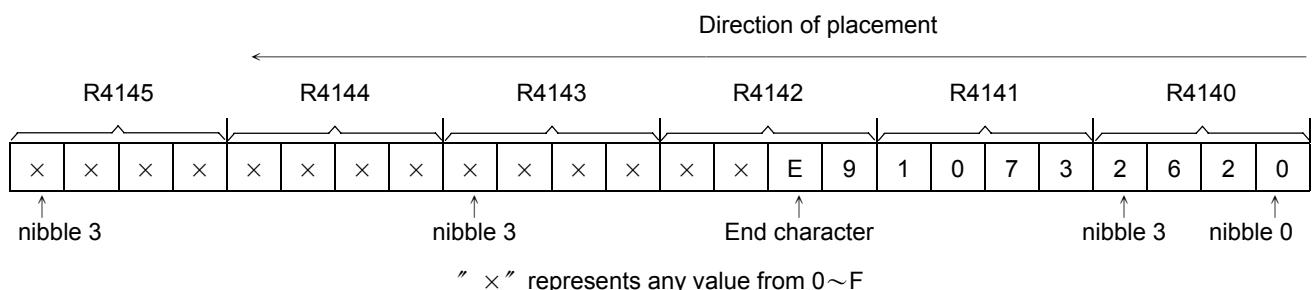
### A. Passive receiving mode

When there are no "valid phone number" stored in the phone number registers of MC main unit (see below), the main unit enters passive receiving mode automatically, where modem is set in receiving mode and waits for the calling of a distant modem. When the connection between the modems are successful, PLC exits the receiving mode and enters the connection mode. At this time, the distant modem can access or control the connected main unit. Beware that the main unit will check the content of phone number registers the instant its power is on or modem is on (OFF→ON). Thus any change to R4140~R4145 (such as addition or deletion of phone numbers) will be activated only after the main unit or modem is turned off and then turned on again.

### B. Active dialing mode

When there is "valid phone number" stored in the phone number registers of MC main unit, the main unit will automatically enter the active dialing mode the instant both the main unit and modem are turned on (OFF→ON), and the phone number storing in R4140 ~ R4145 is dialed from port1 through modem to attempt connection with the distant modem. Once the connection is successful, it enters the connection mode. At this time, distant modem can access or control the connected main unit. If the connection fails, the main unit will undergo dialing the second time, and the third time if the dialing fails again. If all three attempts fail (takes about 4 minutes), the main unit will exit dialing mode automatically and enter passive receiving mode, waiting for the call of distant modem.

The phone number storing in phone number registers will be considered as valid only when the number is stored according to the format described below. First the phone number must be expressed by hexadecimal digit where only 0 ~ 9 and "E" are significant. "A" represents dialing delay which applies to dial extension or international call (one "A" represents 2-second delay). "B" represents "#" character (for calling pager), and "C" represents "\*" character. Valid digits 0~9 represent phone number and "E" represents the end of said phone number. Each register has 4 hexadecimal digits, thus R4140 ~ R4145 have in total 24 hexadecimal digits. After deducting the end character "E", R4140 ~ R4145 can store a phone number with 23 digits at maximum. Phone number is placed in the sequence from nibble 0 of R4140 to nibble 3 of R4145. For instance, the valid storage of a phone number 02-6237019 goes as follows:



As shown above, R4140 stores 2620H, R4141 stores 1073H, R4142 stores XXE9H, and R4143~R4145 can have any value. Please note that the last digit of the phone number must be immediately followed by end character "E." Any figure after "E" which can be 0 ~ F will be ignored. There can only be 0 ~ C before "E" and the presence of any other figure will render the phone number invalid.

If the phone number with extension e.g. 02-28082192 ext 100, as described above, R4140 stores 2A20H, R4141 stores 2808H, R4142 stores A291H, R4143 stores AAAAH, R4144 stores 001AH, R4145 stores 000EH; where the characters "A" are the delay for dialing.

In practical application, the phone call in relation to the provision of technical service will be paid by the service provider. If that's the case, there cannot be any valid phone number stored in the phone number registers of customer's PLC main unit. It means that the PLC of the customer's will go into receiving mode once it's turned on and wait for the call from service provider. If the phone call is to be paid by the customer, the phone number of technical service provider will first be stored in customer's PLC main unit. When the customer turns on modem and PLC, the PLC main unit will dial to the service provider automatically. In light that the phone number of service provider might change, we provide a phone number write-in and dial-back function in Proladder package software. When the service provider changes phone number, the PLC of customer's still stores the old phone number and can't make connection to the modem of service provider. At this time, PLC will dial and redial three times and switches into receiving mode in 4 minutes after the redials fail. At this time, service provider will dial to the customer, download its new phone number in customer's PLC main unit phone number registers and give the dial-back command. Upon receiving the dial-back command, the PLC of customer's will enter dialing mode and call service provider at the new number just downloaded. Although such process requires service provider to call customer first, the cost is minimal since the time of write-in and dial-back of the new number takes little time.

When Proladder executes the command of "phone number write-in and dial-back", it will retrieve the old number in customer's PLC main unit for your reference (just in case you need the write back the old number) before executing the command. It will disconnect the call after the command is completed.

### 12.8.3 Ladder Program Control Interface

This type of interface can be selected in port1 and port2. There are 3 handy instructions to control communication ports, of which, FUN94 (ASCWR) and FUN97 (LINK1) can control port1, while FUN96 (LINK2) can control port2.

FUN94 takes port1 as the output interface of ASCII file (output only), to transmit to receiving equipment that communicate by ASCII code, such as printer, terminal. The most typical application is for printout of production information. In the Proladder software package, there provides an "ASCII file editor" for the user to edit the format of ASCII file and store it into PLC for printing, such as production report or material request table,.... For details of the application, please refer to Chapter 14 on the "Application of ASCII File Output Function."

FUN97 and FUN96 control port1 and port2 respectively as resource sharing between PLC and PLC or connection with other intelligent peripherals. FUN97 has three instruction modes, while FUN96 has four's. For application, please refer to Chapter 13 on the "Application of FB-PLC Link Functions."

## Chapter 13 The Applications for FB-PLC Link Function

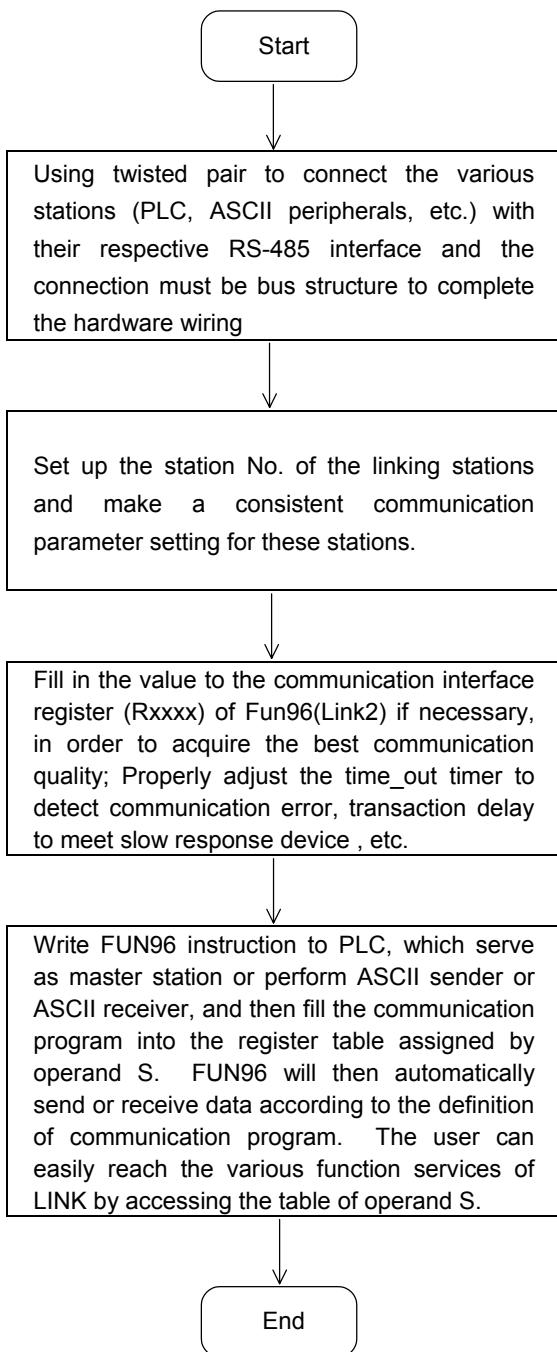
As previously revealed in Chapter 12 that the FB-PLCs connect through the two communication ports - port1 or port2 - to make multi\_drop link operation (both of the ports must be applied in the “ladder diagram control interface”). Of which, Port 2 is controlled by FUN96 “LINK2” instruction while Port 1 by FUN97 (LINK1). For the application of connecting multiple stations, Port2 has a built-in RS-485 interface for multiple stations linkage that can directly link to other PLCs or peripherals with identical RS-485 interface. However, since Port1 has a built-in RS-232 interface that allows one on one links only, it must employ the FB-485 communication adaptor to convert the RS-232 interface into a multiple linking RS-485 interface before it can pave multiple linkages to other RS-485 equipment.

The FUN96 (LINK2) instruction provides MD0 to MD3 four kinds of instruction mode, and the FUN97 (LINK1) instruction provides MD0 to MD2 three kinds of instruction mode. Except that the MD3 mode of FUN96 is a “High Speed Link Network” mode, the others are for “Ordinary Network Link” mode. Except that the setting of maximum speed transferring rate could be different, the other parameters, operations, and usages for “Ordinary Network Link” of FUN96 & FUN 97 are similar. The following list enlisted the description for the difference on various instruction modes for the two LINK instructions of FUN96 and FUN97.

Category		Item	Baud Rate	Data Length	Transmitting code	Error detection	Command processing speed
FUN96 (LINK2)	High Speed LINK (MD3)	38.4Kbps   614.4Kbps	8bits	Binary code	CRC-16	Immediately	
	Ordinary LINK (MD0~MD2)	4.8Kbps   614.4Kbps	7 or 8bits Adjustable	ASCII code	Checksum	Processing during Housekeeping	
FUN97 (LINK1)	Ordinary LINK (MD0~MD2)	600bps   38.4Kbps					

## 13.1 Application for FUN96 (port2) instruction

### 13.1.1 Procedure for FUN96 (LINK2) usage



- Please refer to section 12.5 for the explanation on hardware wiring layout for communication port.
- Station number can be set to any one between 1 to 254 without replication. The setting of station No. can be performed under PROLADDER or No. 5 of the function item of Configuration in FP-07.
- For communication parameter, please refer to section 12.6.2 for description of communication parameter setting.
- Please refer to program example in section 13.1.2 for description and definition of interface processing signals.
- Please refer to program example in section 13.1.2 for description of definition and usage of the operand S.

### 13.1.2 Explanation of respective modes and application program example for FUN 96 (LINK2)

This section will base on the four instruction modes (MD0 to MD3) of FUN96 (LINK2) instruction to explain their usages, with respective practical application program examples.

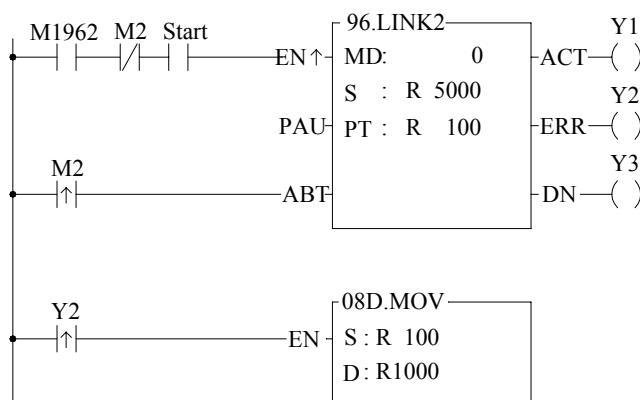
FUN 96 LINK2	Convenient instruction for FUN96(LINK2): MD0 communication network (which makes PLC as the master station in CPU LINK network through Port2)	FUN 96 LINK2																									
	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th rowspan="2">Oper- and</th> <th rowspan="2">Range</th> <th>HR</th> <th>ROR</th> <th>DR</th> <th>K</th> </tr> <tr> <th>R0   R3839</th> <th>R5000   R8071</th> <th>D0   D3071</th> <th></th> </tr> </thead> <tbody> <tr> <td>MD</td> <td></td> <td></td> <td></td> <td>0~3</td> </tr> <tr> <td>S</td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td></td> </tr> <tr> <td>Pt</td> <td><input type="radio"/></td> <td><input type="radio"/>*</td> <td><input type="radio"/></td> <td></td> </tr> </tbody> </table>	Oper- and	Range	HR	ROR	DR	K	R0   R3839	R5000   R8071	D0   D3071		MD				0~3	S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		Pt	<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>		<p>MD : 0, serves as master station for Fatek CPU LINK (employs Fatek communication protocol)</p> <p>S : Starting register of communication program (see example for its explanation)</p> <p>Pt : Starting register for instruction operation (see example for its explanation). It controls 8 registers, the other programs can not repeat in using.</p>
Oper- and	Range			HR	ROR	DR	K																				
		R0   R3839	R5000   R8071	D0   D3071																							
MD				0~3																							
S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																								
Pt	<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>																								
<b>Descriptions</b>																											
<ol style="list-style-type: none"> <li>1. FUN96 (LINK2): MD0 instruction provides data sharing among the Fatek PLCs.</li> <li>2. The master PLC may through its built-in RS-485 interface connects with 254 slave PLCs and share data with each other.</li> <li>3. Only the master PLC needs to use LINK2 instruction (thus, port2 defined as "ladder diagram control interface") while the other slave PLCs need not to use the instruction (thus, defined as "standard interface").</li> <li>4. It employs the program coding method or table filling method to plan for the data flow controls; i.e. from which one of the slave PLC to get which type of data and save them to the master PLC, or from the master PLC to write which type of data to the assigned slave PLC. It needs only seven registries to make definition; every seven registers define one packet of data transaction.</li> <li>5. When execution control "EN ↑" changes from 0→1 and both inputs Pause "PAU" and Abort "ABT" are 0, and if Port2 hasn't been controlled by other FUN96 instructions (i.e. M1962 = 1), this instruction will control the Port2 immediately and set the M1962 to be "0" (which means it is being occupied), then going on a packet of data transaction immediately. If Port2 has been controlled (M1962 = 0), then this instruction will enter into the standby status until the controlling FUN96 instruction complete its transaction or pause/abort its operation to release the control right (M1962=1), and then this instruction will become enactive, set M1962 to be 0, and going on the data transaction immediately.</li> <li>6. While in transaction processing, if operation control "PAU" becomes 1, this instruction will pause and release the control right (M1962 set to be 1) after it finishes the on going transaction. Next time, when this instruction takes over the transmission right again , it will keep going on the next packet of data transaction (this means that the pause operation is based on a packet of data transaction).</li> <li>7. While in transaction processing, if operation control "ABT" becomes 1, this instruction will halt immediately and release the control right (M1962 set to be 1). Next time, when this instruction takes over the transmission right again, it will restart from the first packet of data transaction.</li> <li>8. While it is in the data transaction, the output indication "ACT" will be ON.</li> <li>9. If there is error occurred when it finishes a packet of data transaction, the output indication "ERR" will be ON.</li> <li>10. If there is no error occurred when it finishes a packet of data transaction, the output indication "DN" will be ON.</li> </ol>																											

## FUN96:MD0 Instruction guide

FUN 96 LINK2	Convenient instruction for FUN96(LINK2): MD0 communication network (which makes PLC as the master station in CPU LINK network through Port2)	FUN 96 LINK2
【Interface signals】		
M1962: This signal is generated from CPU. ON, it represents that Port2 is free and ready. OFF, it represents that Port2 is occupied, data transaction is going.		
M1963: This signal is generated from CPU. When the communication program completed the last packet of data transaction, M1963 will be ON for a scan time (for successive data transaction). When the communication program completed the last packet of data transaction, M1963 will be ON (for single packet of data transmission).		
R4053: Response delay time setting (the unit is in mS and the default is 4; it means 4mS delay); When slave PLC or PLC linking through Port2 with computer, man machine interface (MMI), or graphic supervisor, you may set the response delay time. Since Port2 is designed as high speed RS-485, it replies at immense speed with nearly no bit of time wasted. Therefore, it must have response delay so that computer, MMI, or graphic supervisor could be in time to receive data replied by PLC without loss.		
R4157: The Port2 Rx/Tx Time-out setting. The system will produce pertaining setting value according to R4158 communication parameter setting; the user needs not to set it.		
R4158: The register for communication parameter setting of Port2. (please refer to section 12.6.2 for Port2 communication parameter setting descriptions)		
R4159: The content of Low Byte defines the Time-out span of LINK2 instruction; its unit is 0.01 second (the default is 50, which means 0.5 second) The LINK2 instruction (only master PLC needs) employs Time-out span to judge whether the slave PLC on line or not. When the master PLC sent out the read/write command to the slave PLC, the slave PLC didn't reply within this period means that there is abnormal event in communication called Time-out. When there are multi-PLCs linking, properly adjust this value (greater than 1 scan time of the slave PLC with the longest scann time) to shorten the communication response time among the active linking PLCs if there are many slave PLCs power off (The time-out cases will happen).  : The content of High Byte defines the transmission delay time between two packets of data transaction for LINK2 instruction; its unit is 0.01 second (the default is 0). For point to point link, this value can be set as 0 to shorten the communication transaction time and promote the communication efficiency. In the case of linking multi-PLCs and if the scan time of master PLC is far longer than any slave PLC, this value can be set to 0 to shorten the communication transaction time and promote the communication efficiency. When there are multi-PLCs to link in parallel by using master/slave method and the scan time of master PLC is close to that of slave PLCs, it must properly adjust this value (greater than 1 scan time of the slave PLC with the longest scan time) to reach the best, error-free communication quality.		

## FB-PLC acts as the master of Fatek CPU LINK network through Port 2

## Program example Automatic cycling transmission



- Configure R5000~R5199 as the read only register (ROR) before programming, after then, when storing program, the ladder program will automatically contains the communication program.
- When ABT is not controlled, it is not necessary to input the M2 contact instruction.
- When there is communication error, gets and stores the error message to R1000 & R1001 would be helpful for error analysis or logging.

## Description

- Explanation for operand S of FUN96: MD0 (R5000 just only for example, other registers can be used also)

R5000: Starting register of communication program (data transaction table) by filling table method (Not easy)

R5000	Total transactions	• Low Byte is valid; one transaction takes 7 registers to describe, which means 7 registers define a packet of data transaction.
R5001	Slave station No. which is about to transact with	• Low Byte is valid, 0~254 (0 means that master PLC broadcasts the data to all slave PLC, the slave PLC does not reply).
R5002	Command code	• Low Byte is valid; =1, means reading data from slave PLC; =2, means writing data to slave PLC.
R5003	Data length of this transaction	• Low Byte is valid; the range is 1~64. It defines the data length of this transaction.
R5004	Data type of Master PLC	• Low Byte is valid, and its range is 0 to 13; it defines the data type of master PLC (see next page).
R5005	Starting reference of Master PLC	• Word is valid; it defines the starting address of data (master).
R5006	Data type of slave PLC	• Low Byte is valid, and its range is 0 to 13; it defines the data type of slave PLC (see next page).
R5007	Starting reference of Slave PLC	• Word is valid; it defines the starting address of data (slave).
R5008	Slave station No. which is about to transact with	Description of the 2_nd packet of transaction
R5009	Command Code	
R5010	Data length of this transaction	
R5011	Data type of Master PLC	
R5012	Starting reference of Master PLC	
R5013	Data type of slave PLC	
R5014	Starting reference of Slave PLC	

## FUN96:MD0 program example

FB-PLC acts as the master of Fatek CPU LINK network through Port 2

- Master/Slave data type, code and reference number

Data code	Data type	Reference number
0	X (discrete input)	0~255
1	Y (discrete output)	0~255
2	M (internal relay M)	0~1911
3	S (step relay S)	0~999
4	T (timer contact)	0~255
5	C (counter contact)	0~255
6	WX (word of discrete input ,16 bits)	0~240, it must be the multiple of 8.
7	WY (word of discrete output ,16 bits)	0~240, it must be the multiple of 8.
8	WM (word of internal relay,16 bits)	0~1896, it must be the multiple of 8.
9	WS (word of step relay,16 bits)	0~984, it must be the multiple of 8.
10	TR (timer register)	0~255
11	CR (counter register)	0~199
12	R (data register Rxxxx)	0~3839
13	D (data register Dxxxx)	0~3071

Note: The data type for master and slave must be consistent. i.e. if the master station is any value between 0 to 5, the slave station must also be any value between 0 to 5; if the master station is any value between 6 to 13, the slave station must also be any value between 6 to 13.

- Explanation for operand Pt of FUN96:MD0 (R100 just only for example,other registers can be used also)

High Byte		Low Byte	
R100	Result code	Transaction No.	
R101	Station number	Command code	
R102	For internal operation		
R103	For internal operation		
R104	For internal operation		
R105	For internal operation		
R106	For internal operation		
R107	For internal operation		

- Result code indicates the transaction result; 0= normal, other value =abnormal.
- Transaction No. indicates which one is in processing (beginning from 0).
- Station number: the slave station No. which is in transaction.  
Command code  
=44H, reading successive discrete status from slave PLC.  
=45H, writing successive discrete status to slave PLC.  
=46H, reading successive registers from slave PLC.  
=47H, writing successive registers to slave PLC.
- R104's B0=1, Port2 has been occupied and this instruction is waiting to acquire the transmission right for data transaction.  
B4=1, this instruction is not first time performing.  
B12, output indication for "ACT"  
B13, output indication for "ERR".  
B14, output indication for "DN".

Result code: 0, this transaction is successful.

2, data length error (data length is 0 or greater than 64 in one transaction).

3, command code error (command code is greater than 2).

4, data type error (data type is greater than 13, please refer to data type code).

5, reference number error (please refer to reference number).

6, inconsistency in data type (e.g. master station is 0~5 while slave is 6~13).

A, communicating, but no response from slave station (Time-out error).

B, communication error (received error data).

## FB-PLC acts as the master of Fatek CPU LINK network through Port 2

- To make it easy to edit, read, and maintain the communication program, we have extended following related instructions under FUN96:MD0, 3 and FUN97:MD0 instructions. The user may edit, and modify the communication program directly in PROLADDER (if you are intending to edit the communication program with the PROLADDER in DOS version, key in the complete FUN96 or FUN97 instruction and then move cursor to position of FUN96 or FUN97 instruction and press "ALT" "Z" at the same time and it will display and allow to edit the communication program. While editing the communication program, simultaneously pressed "Shift" "INS" means to insert a frame of data transaction at the cursor position; simultaneously pressed "Shift" "DEL" means to delete the cursor position indicated frame of transaction; simultaneously pressed "ALT" "INS" or "Shift" "+" means to append a frame of data transaction to the bottom).

## Extension instructions for communication

Frame No.	Instruction	Operand	Explanation
nnn	Station#	Station number (xxx)	<p>Describing the station number of slave PLC which is about to transact with.</p> <p>Station number=0, The master PLC broadcasts the data to all slave PLCs and slave PLCs will not reply while in FUN96:MD 0 or FUN97:MD 0. (Station No. can't be 0 for FUN96:MD3)</p> <p>Station number=1~254 For FUN96:MD0 or FUN97:MD0, it means the station number of the slave PLC which is about to transact with the master PLC; For FUN96:MD3, it means station number of the PLC that is about to broadcast in high speed CPU link.</p>
	Command	Read Write  H_Link	<p>Master PLC read data from the slave PLC.</p> <p>Master PLC write data to the slave PLC.</p> <p>(Read,Write can only be used for FUN96:MD0 or FUN97:MD0)</p> <p>High speed CPU link (only for FUN96:MD3, and it must employ H_link for all transactions; can not mix with Read, and Write when using)</p>
	Length	1~64 or 1~32	<p>Data length of this transaction.</p> <p>For FUN96:MD0 or FUN97:MD0, the length is 1~64.</p> <p>For FUN96:MD3, the length is 1~32.</p>

FUN96:MD0 program example

FB-PLC acts as the master of Fatek CPU LINK network through Port 2

Frame No.	Instruction	Operand	Explanation
nnn	M_Start	X0 ~ X255 Y0 ~ Y255 M0 ~ Y1911 S0 ~ S999 T0 ~ T255 C0 ~ C255 WX0 ~ WX240 WY0 ~ WY240 WM0 ~ WM1896 WS0 ~ WS984 TR0 ~ TR255 CR0 ~ CR199 R0 ~ R3839 D0 ~ D3071	Describing the data type & reference number of this packet of transaction for the master PLC.  (for FUN96:MD0 or FUN97:MD0)  The number for WX, WY, WM, and WS must be the multiple of 8.
	S_Start	X0 ~ X255 Y0 ~ Y255 M0 ~ Y1911 S0 ~ S999 T0 ~ T255 C0 ~ C255 WX0 ~ WX240 WY0 ~ WY240 WM0 ~ WM1896 WS0 ~ WS984 TR0 ~ TR255 CR0 ~ CR199 R0 ~ R3839 D0 ~ D3071	Describing the data type & reference number of this frame of transaction for the slave PLC.  (for FUN96:MD0 or FUN97:MD0)  The number for WX, WY, WM, and WS must be the multiple of 8.
	Start	R0~R3839 D0~D3071	Data type & reference number for high speed CPU link transaction  (for FUN96:MD3)

FB-PLC acts as the master of Fatek CPU LINK network through Port 2

**Example: Programming for data transaction with instruction method**

R5000: Starting register of communication program (It's very easy to plan the data flow by this method)

Content of registers	Description	Planning the transaction with extended instructions			
R5000:5	5 packet of transactions in total.	Total transactions:5			
R5001:0 R5002:2 R5003:16 R5004:12 R5005:500 R5006:13 R5007:0	Broadcasting from master PLC Write data to all slave PLCs Length of data is 16 Data type of master PLC is R Reference number of master PLC is 500, i.e. R500 Data type of slave PLC is D Reference number of slave PLC is 0, i.e. D0	000	Station#	0	
			Command	Write	
			Length	16	
			M_start	R500	
			S_start	D0	
<ul style="list-style-type: none"> <li>• Master PLC broadcasts the R500~R515 to all slave PLCs' D0~D15</li> </ul>					
R5008:2 R5009:1 R5010:10 R5011:12 R5012:20 R5013:12 R5014:200	The slave PLC in transaction is the station No.2 Read data from slave PLC Data length is 10 Data type of master PLC is R. Reference number of master PLC is 20, i.e. R20 Data type of slave PLC is R Reference number of slave PLC is 200, i.e. R200	001	Station#	2	
			Command	Read	
			Length	10	
			M_start	R20	
			S_start	R200	
<ul style="list-style-type: none"> <li>• Read R200~R209 from slave PLC No.2 to R20~R29 of master PLC</li> </ul>					
R5015:3 R5016:1 R5017:20 R5018:2 R5019:1000 R5020:2 R5021:100	The slave PLC in transaction is the station No.3 Read data from slave PLC Data length is 20 Data type of master PLC is M. Reference number of master PLC is 1000, i.e. M1000 Data type of slave PLC is M Reference number of slave PLC is 100, i.e. M100	002	Station#	3	
			Command	Read	
			Length	20	
			M_start	M1000	
			S_start	M100	
<ul style="list-style-type: none"> <li>• Read M100~M119 from slave PLC No.3 to M1000~M1019 of master PLC</li> </ul>					
R5022:4 R5023:2 R5024:20 R5025:2 R5026:1000 R5027:3 R5028:100	The slave PLC in transaction is the station No.4 Write data to slave PLC Data length is 20 Data type of master PLC is M. Reference number of master PLC is 1000, i.e. M1000 Data type of slave PLC is S Reference number of slave PLC is 100, i.e. S100	003	Station#	4	
			Command	Write	
			Length	20	
			M_start	M1000	
			S_start	S100	
<ul style="list-style-type: none"> <li>• Master PLC writes M1000~M1019 to S100~S119 of slave PLC No.4, i.e.</li> </ul>					
to write from M100~M119 of slave PLC No. 3 to S100~S119 of slave PLC No.4					
R5029:4 R5030:1 R5031:4 R5032:9 R5033:0 R5034:6 R5035:0	The slave PLC in transaction is the station No.4 Read data from slave PLC Data length is 4 (4 words this situation) Data type of master PLC is WS. Reference number of master PLC is 0, i.e. WS0 Data type of slave PLC is WX Reference number of slave PLC is 0, i.e. WX0	004	Station#	4	
			Command	Read	
			Length	4	
			M_start	WS0	
			S_start	WX0	
<ul style="list-style-type: none"> <li>• Read X0~X63 of slave PLC No.4 to S0~S63 of master PLC</li> </ul>					

## FUN96:MD0 program example

FB-PLC acts as the master of Fatek CPU LINK network through Port 2

### Explanation on program example

- When execution control "EN ↑" changes from 0→1, and Port2 is not occupied by other FUN96 (M1962 ON) and M2=OFF, LINK2 instruction will start the data transaction. The M1962 is OFF during data transaction, and when the transaction is finished, the M1962 becomes ON. Employ the OFF↔ON change of M1962 (FUN96 execution control "EN ↑"=0→1 means starting) may automatically starts for every frame of data transaction successively (when the last packet of transaction is completed, it will automatically return to the first packet of transaction to obtain the automatic cycling transmission).
- When abort control M2 changes from 0→1, it aborts transmission immediately (if the data is in transmitting, it will stop transmitting immediately). Next time when starts the transaction, it will begin from the first packet of transactions.

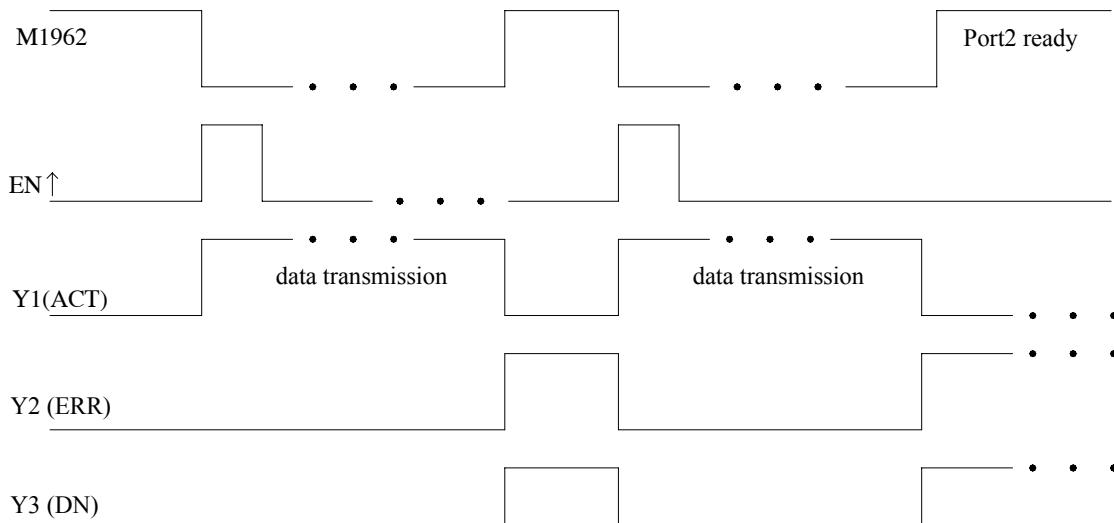
#### ● Output indication

"ACT" ON: The Y1 ON, transaction is going

"ERR" ON: The Y2 ON, error occurred in previous packet of transaction (refer to result code).

"DN" ON: The Y3 ON, previous packet of transaction is completed and is error free.

#### ● Waveform of input control and output indication



Note 1: Of Y2 and Y3, only one of them will be in ON status and not both to be ON at the same time.

2: After the last frame of transaction completed, the M1963 will be ON for one scan time.

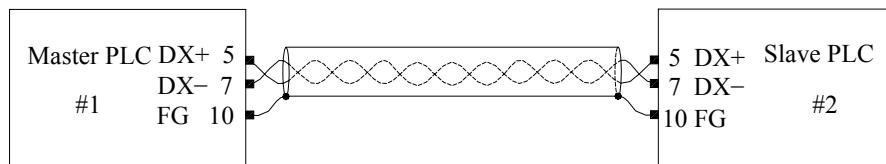
**FB-PLC acts as the master of Fatek CPU LINK network through Port 2**

⟨ Point to point wiring ⟩ Point to point link of master PLC and slave PLC through RS-485.

The communication port of PLC is a 15 Pin D-Sub female connector, therefore a 15 Pin D-Sub cable with both ends to be male connector is needed to link the PLCs.

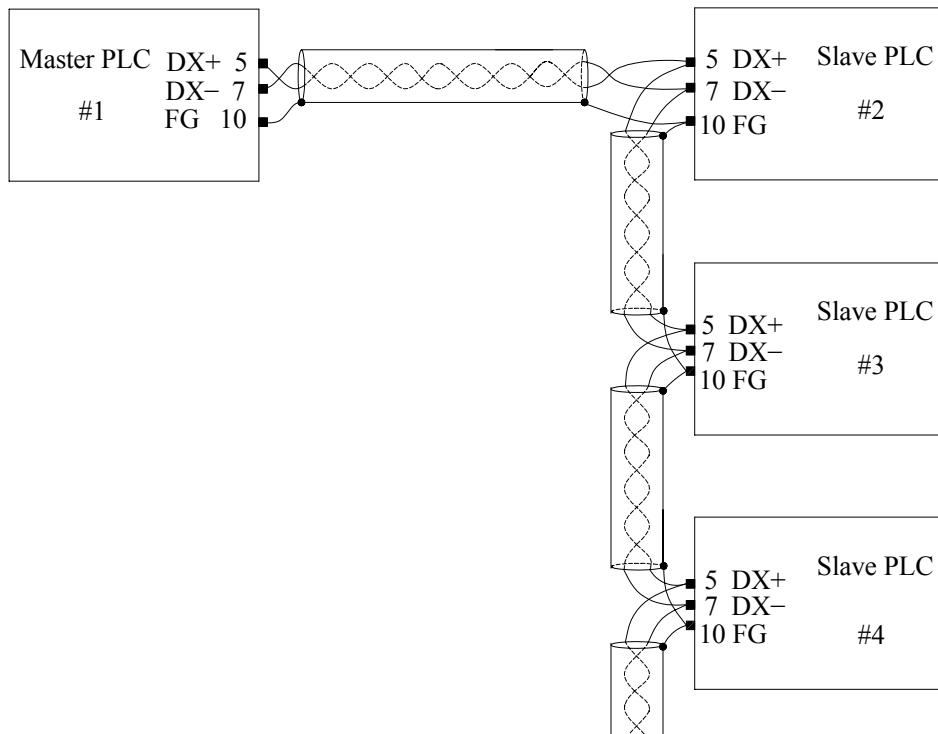
Connecting as follows:

PIN_5 (DX+)	<----->	PIN_5 (DX+)
PIN_7 (DX-)	<----->	PIN_7 (DX-)
PIN_10 (FG)	<----->	PIN_10 (FG)



Note: It's more easy to use FB-485P2 adapter (with terminal block) to connect multi-stations in parallel.

⟨ Multi\_drop wiring ⟩ Master PLC links with multi slave PLCs through built-in RS-485.

**【Cautions】**

1. The RS-485 wiring must employ twisted pair as the transmission cable.
2. Star topology of the wiring must be avoided ; it must be cascaded with stations one after one.
3. The outer layer of weaved net for twisted pair must connect to the FG (to prevent from interference and decrease the common mode interference).
4. Avoid the wiring operation when the PLC is in "RUN" mode, the interference from human body may cause the PLC into "STOP" mode that need to be shut down and restart again.

## FUN96:MD1 instruction guide

FUN 96 LINK2	Convenient instruction for FUN96(LINK2): MD1 (Which makes PLC serve as "ASCII sender" through Port2)	FUN 96 LINK2																									
	<p>MD : 1, link with intelligent peripherals that equipped with ASCII interface.</p> <p>S : Starting register for data transmission table (see example for explanation)</p> <p>Pt : Starting register for instruction operation (see example for explanation). It controls 8 registers at least, the other programs cannot repeat in use.</p>																										
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: center;">Range</th> <th style="text-align: center;">HR</th> <th style="text-align: center;">ROR</th> <th style="text-align: center;">DR</th> <th style="text-align: center;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Oper- and</td> <td style="text-align: center;">R0   R3839</td> <td style="text-align: center;">R5000   R8071</td> <td style="text-align: center;">D0   D3071</td> <td></td> </tr> <tr> <td style="text-align: center;">MD</td> <td></td> <td></td> <td></td> <td style="text-align: center;">0~3</td> </tr> <tr> <td style="text-align: center;">S</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> <tr> <td style="text-align: center;">Pt</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input checked="" type="radio"/>*</td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> </tbody> </table>			Range	HR	ROR	DR	K	Oper- and	R0   R3839	R5000   R8071	D0   D3071		MD				0~3	S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		Pt	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>	
Range	HR	ROR	DR	K																							
Oper- and	R0   R3839	R5000   R8071	D0   D3071																								
MD				0~3																							
S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																								
Pt	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>																								
<p><b>Descriptions</b></p> <ol style="list-style-type: none"> <li>1. FUN96(LINK2):MD1 instruction provides the Fatek PLC to act as the ASCII sender to link with the intelligent peripherals that equipped with ASCII interface.</li> <li>2. A master PLC may connect to multi sets of peripherals that have identical communication protocol through built-in RS-485 interface.</li> <li>3. The communication protocol/format is written with LADDER program, which must be consistent with the linked ASCII peripherals.</li> <li>4. When execution control “EN ↑” turns from 0→1 and both pause “PAU” and abort “ABT” are 0, and if Port2 is not controlled by other FUN96 instruction (which means M1962=1), this instruction will control Port2 immediately and set M1962 to be “0” (being controlled) to proceed data transaction. If Port2 is being controlled (M1962=0), this instruction will enter into the wait state until the other controlling FUN96 instruction complete or pause/abort its operation and released the control right (M1962=1), and this instruction will enact again out of wait state to set the M1962 to be “0” and proceed the transmission transaction.</li> <li>5. During transaction, if the pause “PAU” becomes 1, this instruction will pause and release the control right (set M1962 to be 1) after it completed the transmitting of the on-going data transmission.</li> <li>6. During transaction, if the abort “ABT” becomes 1, this instruction will halt the transmission and release the control right immediately (set M1962 to be 1).</li> <li>7. While transaction is going, the output indication “ACT” will be ON.</li> <li>8. When a packet of data transaction is finished (transmission finished or “transmit then receive” completed), if there is error occurred, the output indication “ERR” will be ON.</li> <li>9. When a packet of data transaction is finished (transmission finished or “transmit then receive” completed), if there is no error occurred, the output indication “DN” will be ON.</li> </ol>																											

Convenient instruction for FUN96 (LINK2): MD1  
(Which makes PLC serve as "ASCII sender" through Port2)

【Interface signals】

M1962: This signal is generated from CPU

ON means Port2 is ready.

OFF means Port2 is busy.

M1963: This signal is generated from CPU; the same as M1962.

ON, it means data transaction has been completed.

R4148: High byte of R4148, Time-out setting for receiving, which is used to determine whether a packet of data has been received completely. The unit is 0.001 second and default is 0CH(12mS).  
Detailed description will be followed.

R4157: Port2 Rx/Tx time-out setting; the system will base on R4158 communication parameter to acquire the pertaining set point, hence the user need not to set.

R4158: The register for communication parameter setting of port 2.

(refer to section 12.6.2 for explanation on Port2 communication parameter setting)

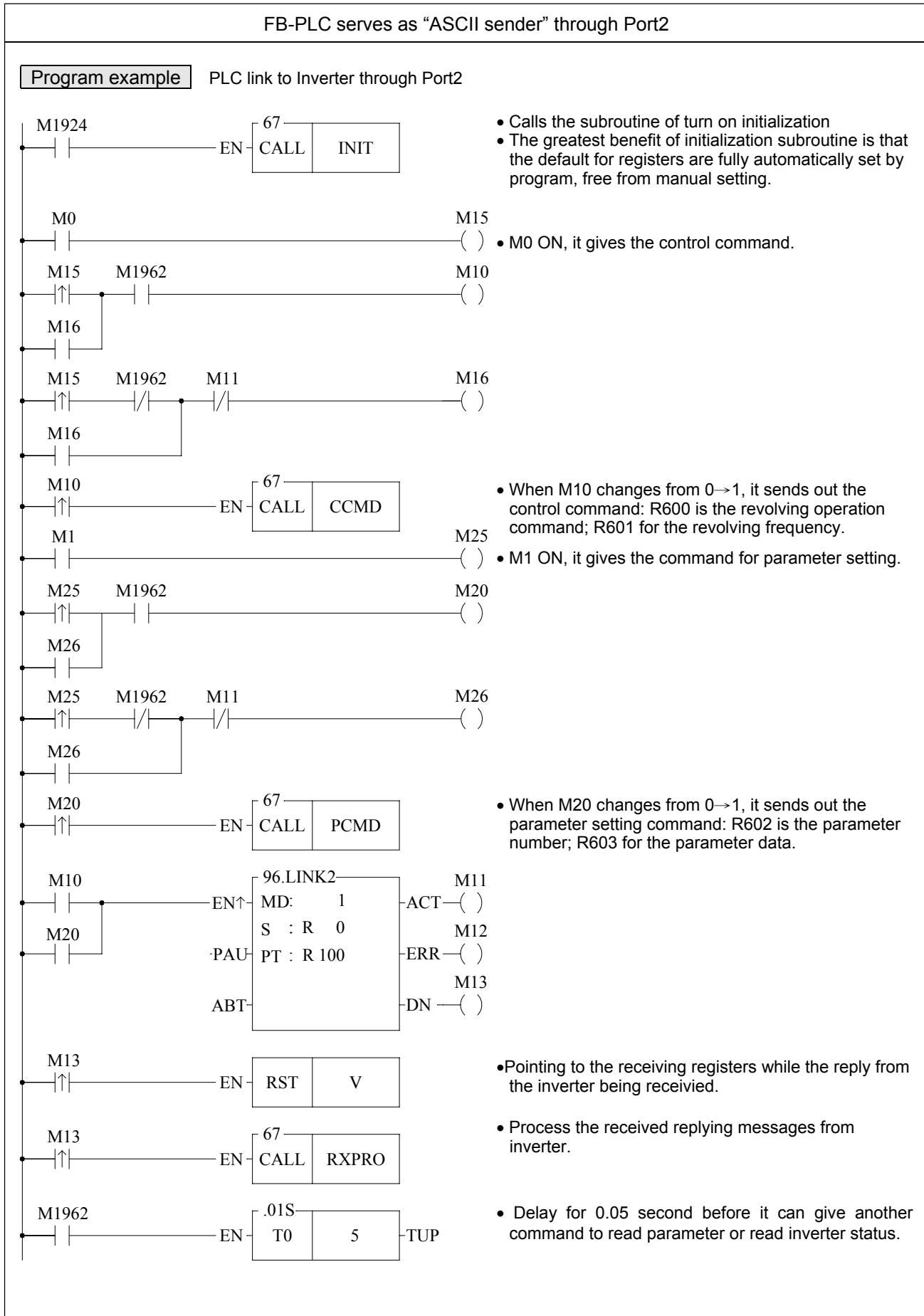
R4159: Low byte of R4159, it defines the Time-out span of link2 instruction; the unit is 0.01 second (the default is 32H, i.e. 0.5 second)

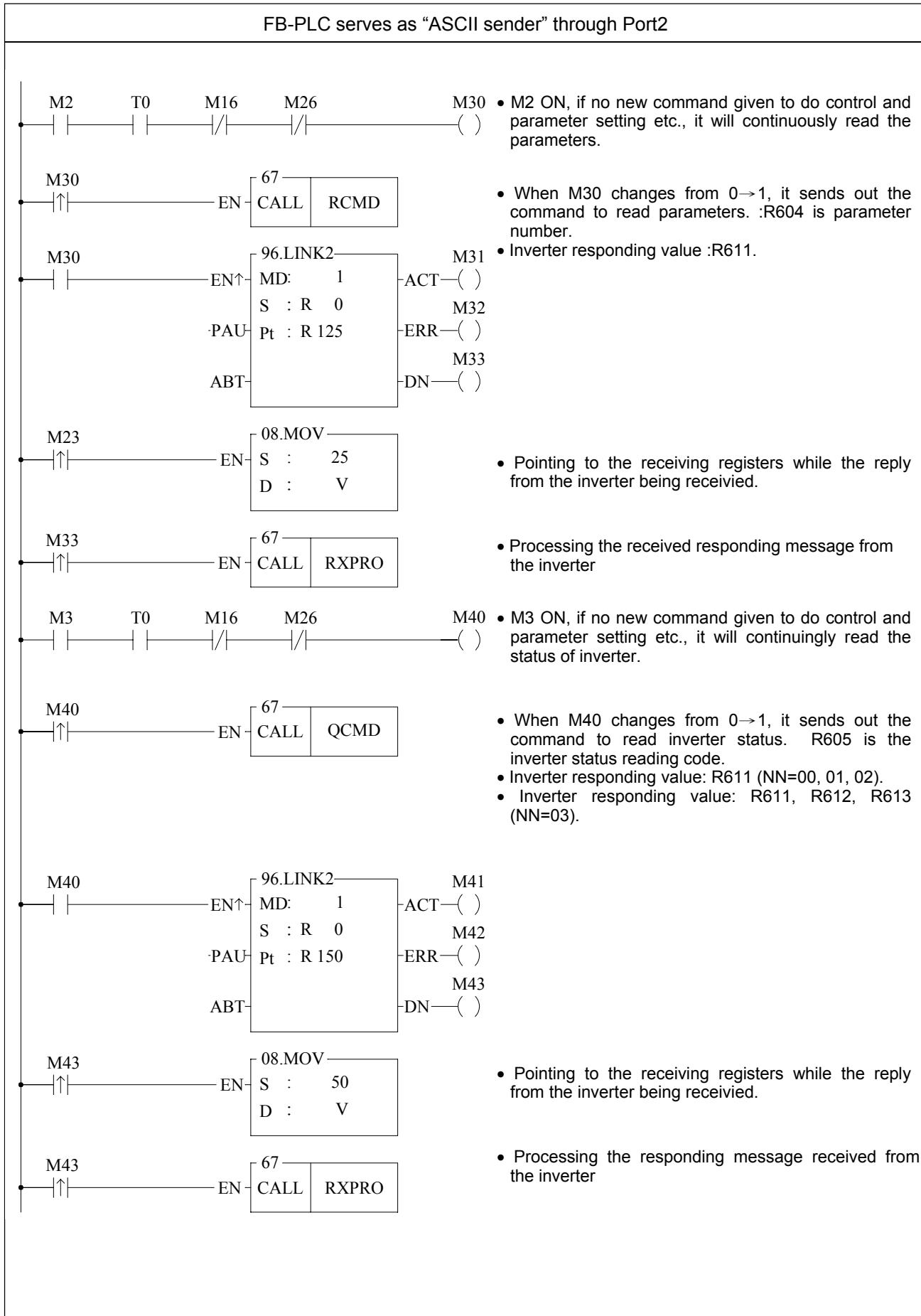
LINK2 instruction depends on Time-out span to detect whether the communication partner is free from error on line; when the LINK2 MD1 setting is in "transmit then receive" mode (example will be followed), the Time-out error will occur if PLC send a packet of data to the peripheral but it didn't reply within this duration.

When LINK2 MD1 setting is "transmit" only (example will be followed), low byte of R4159 is meaningless.

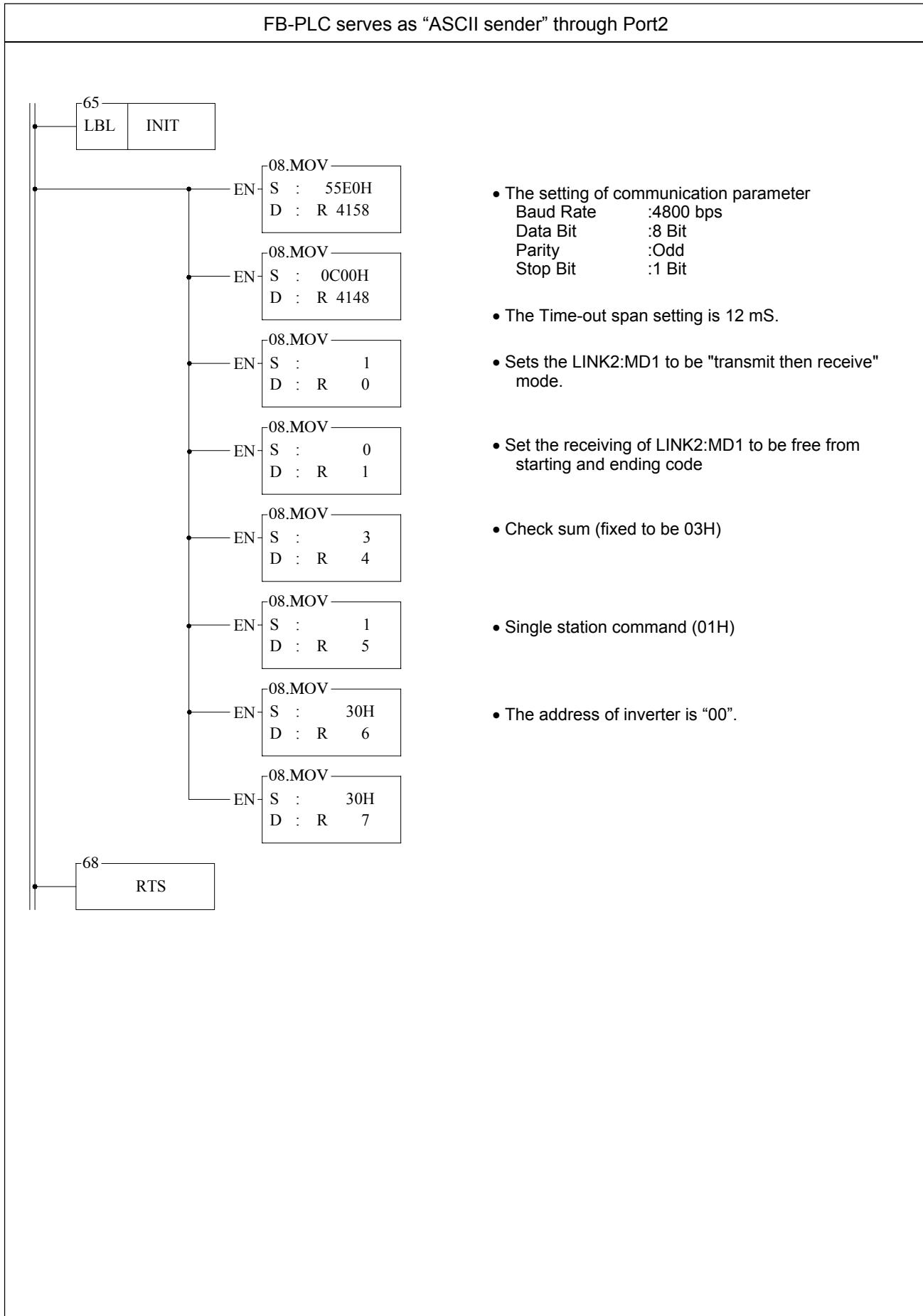
: High byte of R4159, for FUN96:MD1, the recommended setting is 0.

## FUN96:MD1 Program example

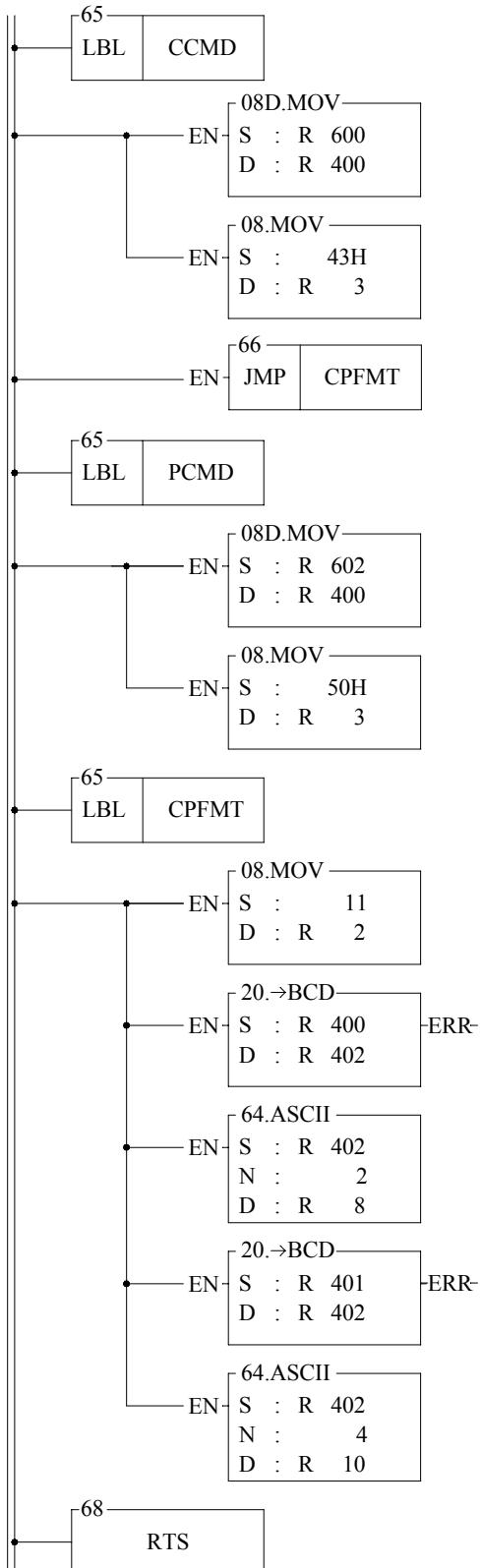




## FUN96:MD1 Program example



## FB-PLC serves as "ASCII sender" through Port2



- Control command; command format: "C S A UU MM FFFF"

- R6000 is the revolving operation command.
- R601 is the frequency of revolving.

- The ASCII code of "C"

- Parameter setting command; command format: "P S A UU NN DDDD".

- R602 is the parameter number.
- R603 is the parameter data.

- The ASCII code of "P"

- The length of data for transmission is 11.

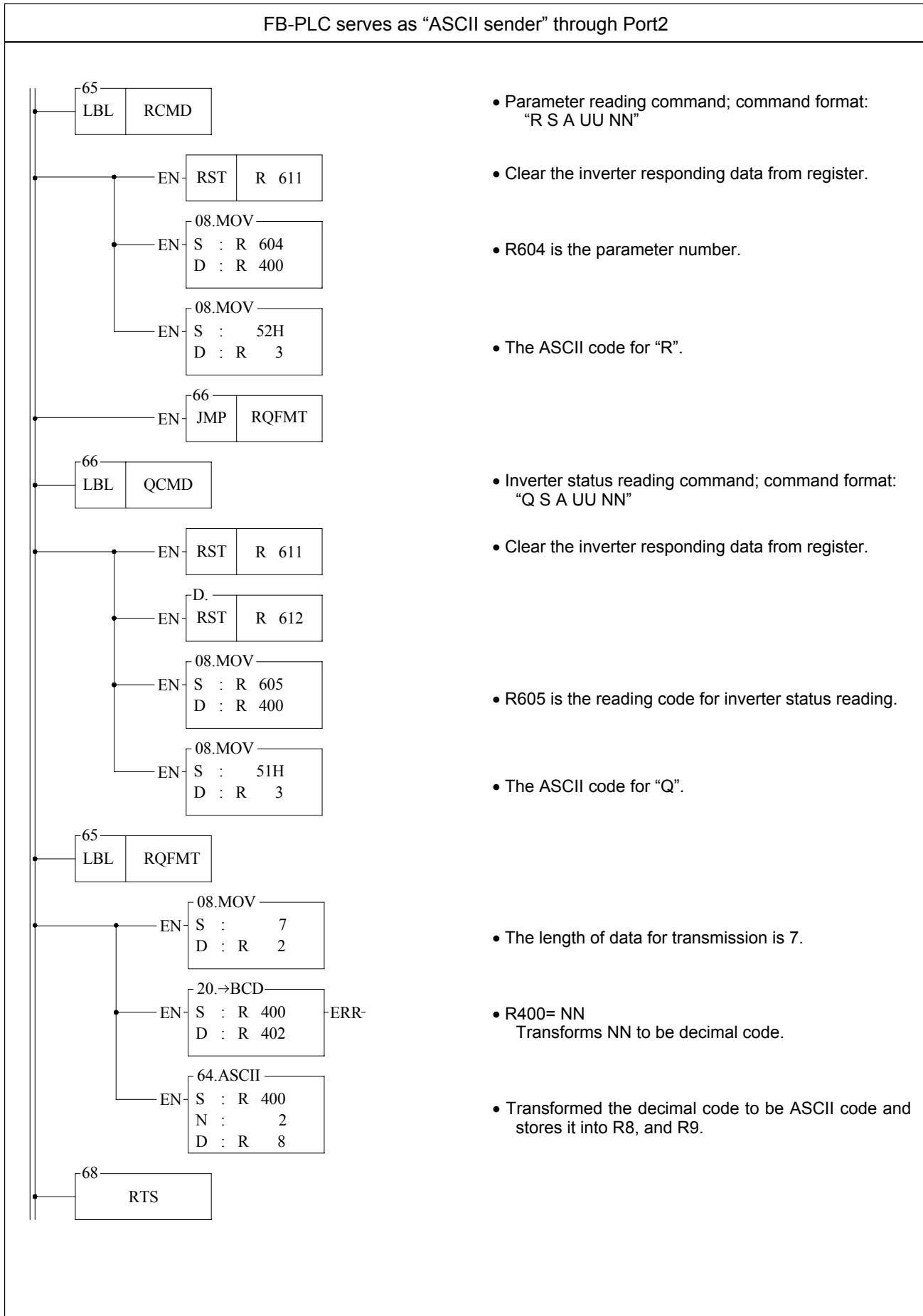
- R400=MM or NN  
Converts MM or NN to be decimal code.

- Converts the decimal code to be ASCII code and store it into R8, and R9.

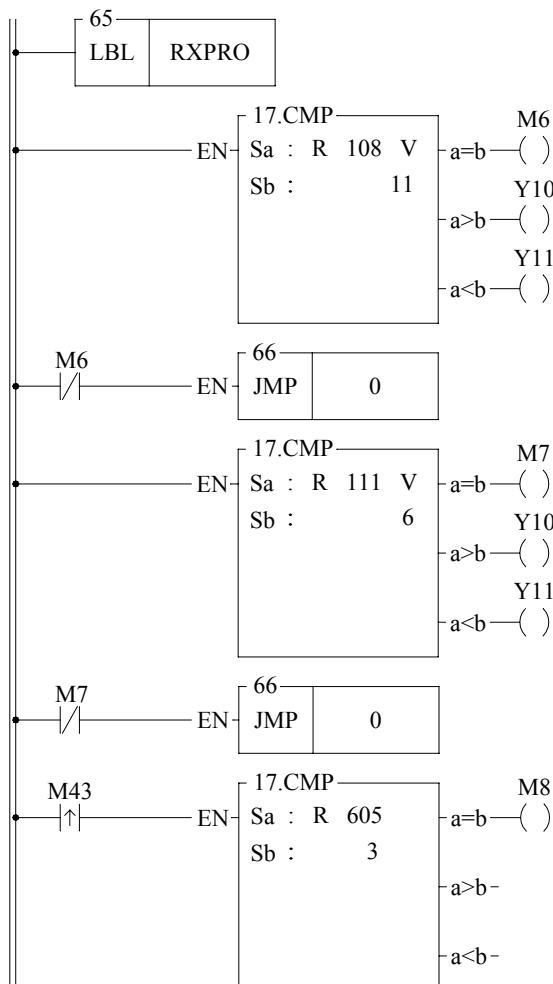
- R401=FFFF or DDDD; transforms FFFF or DDDD to be decimal code.

- Transforms the decimal code to be ASCII code and stores it into R10, R11, R12, and R13.

## FUN96:MD1 Program example



## FB-PLC serves as "ASCII sender" through Port2

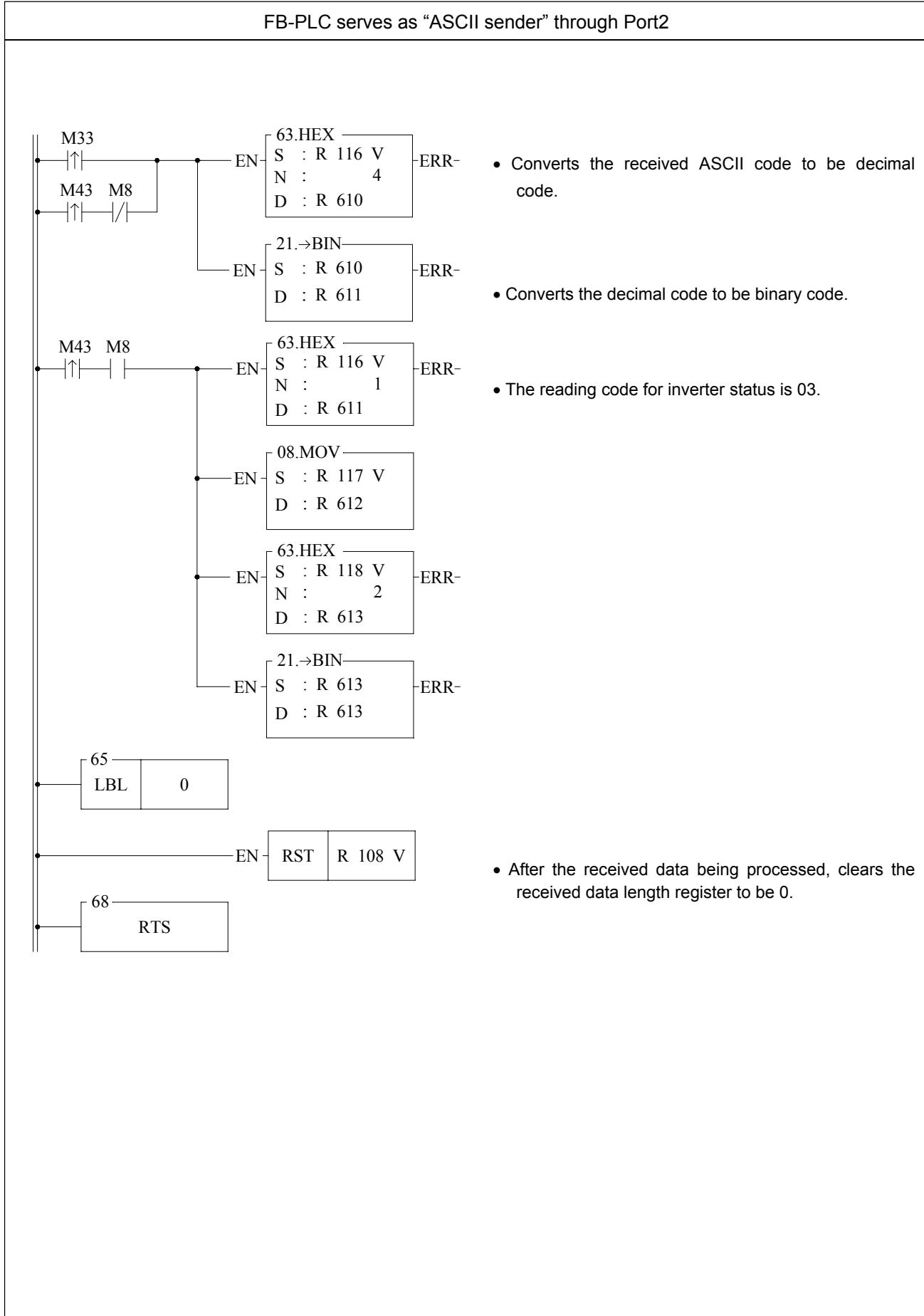


- To judge whether the responding data length is correct or not, if it is abnormal, then Y10 or Y11 is ON.

- To judge whether the responding command is correct or not, if it is abnormal, then Y10 or Y11 is ON.

- To judge whether the inverter status code is 03 or not, if yes, then M8 is ON.

## FUN96:MD1 Program example



## FB-PLC serves as "ASCII sender" through Port2

● Explanation of FUN96: MD1 parameter S.

R0: Starting register of data transmission table (R0 just only for example)

R0	Transmit only/transmit then receive	<ul style="list-style-type: none"> <li>• Low byte is valid, 0: transmit only, no response from the counter partner. 1: transmit then receive the responding message.</li> </ul>
R1	Starting /Ending code of receiving	<ul style="list-style-type: none"> <li>• High byte: Describing the starting code of responding message while receiving Low byte: Describing the ending code of responding message while receiving.</li> </ul>
R2	Length of transmission	<ul style="list-style-type: none"> <li>• The maximum length of data to be transmitted is 511.</li> </ul>
R3	Data 1	<ul style="list-style-type: none"> <li>• Low byte is valid</li> </ul>
R4	Data 2	<ul style="list-style-type: none"> <li>• Low byte is valid</li> </ul>
R5	Data 3	<ul style="list-style-type: none"> <li>• Low byte is valid</li> </ul>
R7	Data 4	<ul style="list-style-type: none"> <li>• Low byte is valid</li> </ul>
•		
•		
•		
	Data N	<ul style="list-style-type: none"> <li>• Low byte is valid</li> </ul>

Note 1: When selecting the transmit-only mode, the Starting /Ending code of receiving is meaningless.

- 2: When it is in the "transmit then receive" mode, before the starting of transmission, it must first to estimate the starting and ending code of responding message from communication partner and write them into the receiving starting/ending code register (e.g. R1=0203H, 02H stands for starting code and 03H for ending code), so as to ensure the receiving to be free from error. The communication protocol with starting/ending code makes the identifying of every packet of messages easy, and the communication program is simple and efficient.
- 3: When it is in the "transmit then receive" mode, fills the high byte of starting/ending code register with 0 if no starting code in responding message; if no ending code in responding message, fills 0 to the low byte of starting/ending code register. Adjusts the high byte of R4148 (Time-out span) to judge whether a packet of data has been received completely; the unit is 0.001 second (the default is 0CH, 12mS). The communication protocol without ending code depends on Time-out span to tell whether it has received completely a packet of data (the setting of Time-out must be greater than the maximum response delay time between data bytes when communication partner is replying), thus it may ensure the receiving of the whole packet to be complete. Generally speaking, the data in transmitting is transmitted one byte after another continuously; therefore, if there is pause (greater than Time-out duration), it means the packet of message is transmitted completely.

## FUN96:MD1 Program example

FB-PLC serves as “ASCII sender” through Port2

- Explanation of FUN96:MD1 parameter Pt.

	High Byte	Low Byte	
R100	Result code	0	<ul style="list-style-type: none"> <li>The result code stores the operation result, 0=Normal; the other values, Abnormal.</li> </ul>
R101	For internal operation use		<ul style="list-style-type: none"> <li>For internal operation use: it is the registers require to be used by CPU when performing LINK2 instruction.</li> </ul>
R102	For internal operation use		
R103	For internal operation use		
R104	For internal operation use		<ul style="list-style-type: none"> <li>The B0 of R104 is 1 means that Port2 is busy; this instruction is waiting to take the transaction right</li> </ul>
R105	For internal operation use		<ul style="list-style-type: none"> <li>B12= “ACT” output indication</li> <li>B13= “ERR” output indication</li> <li>B14= “DN” output indication.</li> </ul>
R106	For internal operation use		
R107	For internal operation use		
R108	Total amount of data received		<ul style="list-style-type: none"> <li>The total amount of data byte that is received (the register for received data length; it includes the starting and ending code that is received).</li> </ul>
R109	1		<ul style="list-style-type: none"> <li>The first byte of data received (if there is the starting code, it is the starting code); High byte =0.</li> </ul>
R110	2		<ul style="list-style-type: none"> <li>The second byte of data received; High byte =0.</li> </ul>
•	3		<ul style="list-style-type: none"> <li>The third byte of data received; High byte =0.</li> </ul>
•			
	N		<ul style="list-style-type: none"> <li>The N_th byte of data received (if there is the ending code, it is the ending code); High byte =0.</li> </ul>

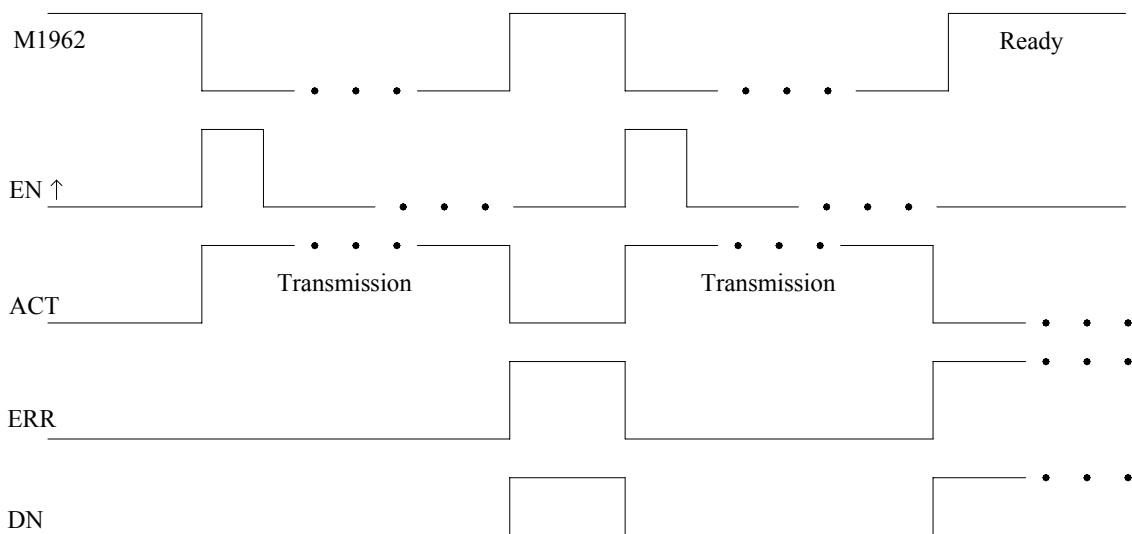
Result code: 0, transaction is successful.

2, data length error (the value is 0, or the packet of transaction is greater than 511)

A, no response from the counter partner.

B, communication abnormal (received error data)

- The waveform for input control and output indication



Note: Of “ERR” and “DN”, only one of them will be in ON status and not both to be ON at the same time.

FUN 96 LINK2	Convenient instruction for FUN96(LINK2): MD2 (Which makes PLC serve as “ASCII receiver” through Port2)	FUN 96 LINK2																									
	<p>Execution control - EN ↑</p> <p>Pause - PAU</p> <p>Abort - ABT</p> <p>96.LINK2</p> <p>MD : 2</p> <p>S :</p> <p>Pt :</p> <p>ACT</p> <p>ERR</p> <p>DN</p>	<p>MD : 2, PLC waiting to receive the message sent by intelligent peripherals</p> <p>S : Starting register of data transmission table (see example for explanation)</p> <p>Pt : Starting register for instruction operation (see example for explanation). It controls 8 registers at least, the other programs cannot repeat in using.</p>																									
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: center;">Range</th> <th style="text-align: center;">HR</th> <th style="text-align: center;">ROR</th> <th style="text-align: center;">DR</th> <th style="text-align: center;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Oper- rand</td> <td style="text-align: center;">R0   R3839</td> <td style="text-align: center;">R5000   R8071</td> <td style="text-align: center;">D0   D3071</td> <td></td> </tr> <tr> <td style="text-align: center;">MD</td> <td></td> <td></td> <td></td> <td style="text-align: center;">0~3</td> </tr> <tr> <td style="text-align: center;">S</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> <tr> <td style="text-align: center;">Pt</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/>*</td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> </tbody> </table>			Range	HR	ROR	DR	K	Oper- rand	R0   R3839	R5000   R8071	D0   D3071		MD				0~3	S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		Pt	<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>	
Range	HR	ROR	DR	K																							
Oper- rand	R0   R3839	R5000   R8071	D0   D3071																								
MD				0~3																							
S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																								
Pt	<input type="radio"/>	<input type="radio"/> *	<input type="radio"/>																								

### Descriptions

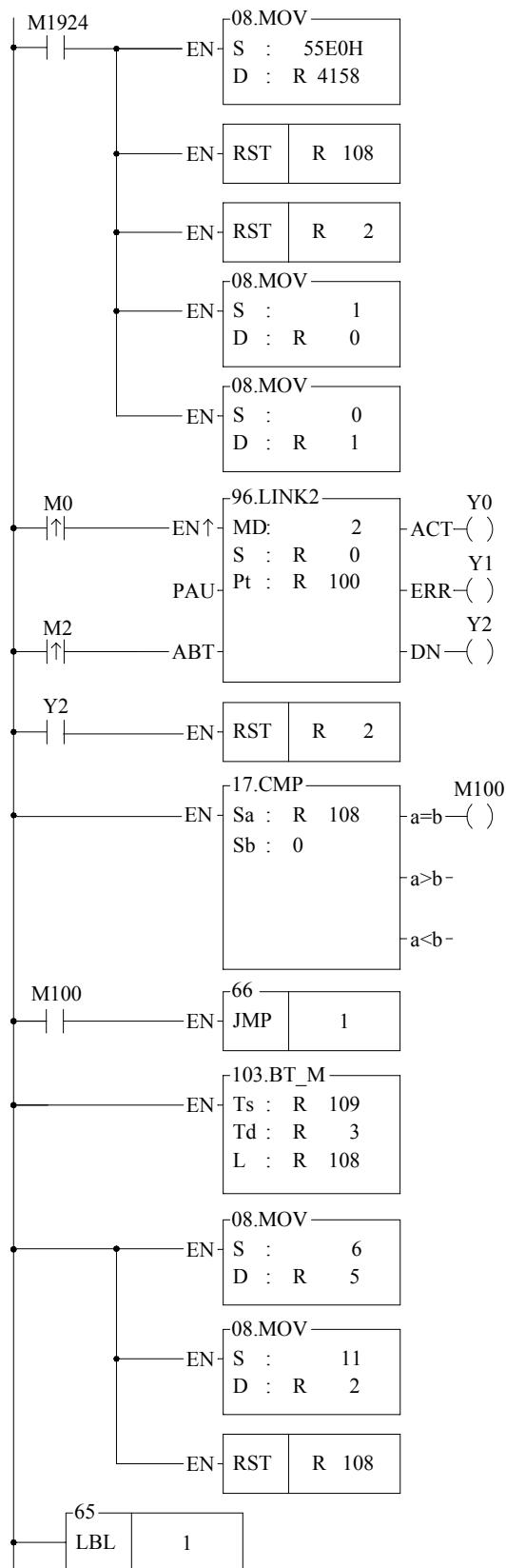
1. FUN96: MD2 instruction provides Fatek PLC with ability to receive message sent by peripherals with ASCII interface at any time.
2. The communication protocol is written with LADDER program, which must be consistent to the ASCII peripherals.
3. When execution control “EN ↑” turns from 0→1 and both pause “PAU” and abort “ABT” are 0, and if Port2 is not controlled by other FUN96 instruction (which means M1962=1), this instruction will control Port2 immediately and set M1962 to be “0” (being controlled). If Port2 is being controlled (M1962=0), this instruction will enter into the wait state until the other controlling FUN96 instruction complete or pause/abort its operation and released the control right (M1962=1), and this instruction will enact again out of wait state to enter into the receiving state and set the M1962 to be “0”.
4. When the operation pause “PAU” or abort “ABT” becomes 1, it gives up the receiving immediately (M1962 ON).
5. While it is in the receiving state, the output indication “ACT” is ON.
6. When a packet of data transaction finished (receive finished or receive then transmit completed), if there is error occurred, the output indication “ERR” will be ON for one scan time.
7. When a packet of data transaction finished (receive finished or receive then transmit completed), if there is no error occurred, the output indication “DN” will be ON for one scan time.

## FUN96:MD2 instruction guide

FUN 96 LINK2	Convenient instruction for FUN96(LINK2): MD2 (Which makes PLC serve as "ASCII receiver" through Port2)	FUN 96 LINK2
<p><b>【Interface signals】</b></p> <p>M1962 : This signal is generated from CPU. ON means Port2 is ready. OFF means Port2 is busy.</p> <p>R4148 : High Byte, the setting point for Time-out span on receiving; it is used to judge whether a packet of data has been received completely. Its unit is 0.001 second (the default is 0CH, 12 mS) (detailed explanation will be followed).</p> <p>R4157 : The Port2 Rx/Tx Time-out setting. The system will produce pertaining setting value according to R4158 communication parameter; the user needs not to set it</p> <p>R4158 : The register for communication parameter setting of Port2. (please refer to section 12.6.2 for communication parameter setting descriptions)</p> <p>R4159 : The Low Byte defines the Time-out span of FUN96:MD2 instruction; its unit is 0.01 second (the default is 32H, which means 0.5 second). When the PLC received the message and must respond to it (receive then transmit mode), but the LADDER program is unable to process and send out the responding message during this period of time, the CPU will give up response this time and automatically restore back to receiving state. When FUN96:MD2 is set to be "receive only" mode (example to be followed), this value is meaningless.</p> <p>: High Byte, it is meaningless while FUN96:MD2</p>		
<p>Note 1: Once FUN96:MD2 activated, it will stay in receiving state all the time; unless the input signal of PAU" or "ABT" becomes ON, then it will jump out of receiving state and stop receiving and waiting for next time it will be activated again.</p> <p>2: When there is change on Starting/Ending code for receiving, it must make the input signal of PAU" or "ABT" becomes ON once, and re-activate the receive control "EN ↑ " from 0→1 to start message receiving</p>		

## FB-PLC serves as "ASCII receiver" through Port2

**Program example** The PLC simulates the Inverter to reply the received data to the Master PLC which sent out the data.



- Communication parameter setting:  
Baud Rate:4800 · Data Bit:8  
Parity: Odd · Stop Bit:1
- Clear the length of receiving data to be 0.
- Clear the length of reply data to be 0
- Set the receiving mode to be "receive then transmit"
- Set the receiving message to be free from starting code and ending code.
- When transmitting complete, clear the length of transmitting (for receive only mode, this instruction is not needed)
- Determines whether a new packet of message is received; if yes, M100=OFF and process the received data.
- Copy all of the received data to reply registers.
- R108 is the length of received data.
- The responding command is 6.
- Fill in the length of reply data which is equal to 11, to start the reply transmission.
- Clear the length of received data (ready to receive new data).

## FUN96:MD2 program example

FB-PLC serves as "ASCII receiver" through Port2

- FUN96: MD2 explanation of parameter S.

R0: Starting register for data receiving table (R0 just only for example)

R0	Receive only/Receive then transmit	<ul style="list-style-type: none"> <li>• Low Byte is valid,</li> </ul> <p>0: "receive only" mode. 1: "receive then transmit" mode.</p>
R1	Starting/Ending code of receiving	<ul style="list-style-type: none"> <li>• High Byte : Describing the starting code of receiving Low Byte : Describing the ending code of receiving.</li> </ul>
R2	Length of reply data	<ul style="list-style-type: none"> <li>• Maximum of length is 511. It will start to transmit the reply data as long as the length is not 0.</li> </ul>
R3	Reply data 1	<ul style="list-style-type: none"> <li>• Low Byte is valid</li> </ul>
R4	Reply data 2	<ul style="list-style-type: none"> <li>• Low Byte is valid</li> </ul>
•		
•		
	Reply data N	<ul style="list-style-type: none"> <li>• Low Byte is valid</li> </ul>

Note 1: When selecting the "receive only" mode, CPU fills the received data into the receiving registers and set the length after it has received a packet of message, and starts to receive the next packet of message immediately.

2: When selecting the "receive then transmit" mode, CPU fills the received data into the receiving registers and set the length after it has received a packet of message; then it starts to wait for the reply data length which is not zero to start transmitting reply data (therefore when select this mode, it must control the reply data length to be zero before the reply data completely filled into the reply registers; when the reply data fills into the reply registers finished, it may then set the length of reply data).

3: It must fill the starting code and ending code into the starting/ending code register before the starting of receiving (e.g. R1=0A0DH, 0AH stands for starting code and 0DH for ending code), so as to ensure it to be free from receiving error.

The communication protocol with starting/ending code makes the identifying of every packet of messages easy, and the communication program is simple and efficient.

4: If the receiving message without starting code, fills the high byte of starting/ending code with 0; if the receiving message without ending code, fills the low byte of starting/ending code with 0. Adjusting High Byte of R4148 (Time-out span) to detect whether a packet of message has been received completely, the unit is 0.001 second (default is 0CH, 12 mS). The communication protocol without ending code depends on Time-out span to tell whether it has received completely for a packet of data (the setting point of Time-out must be greater than the maximum delay time between data bytes to be received), thus it may ensure the receiving of the whole packet to be completed. Generally speaking, the data in transmitting is transmitted one byte after another continuously; therefore, if there is pause (greater than Time-out duration), it means that the packet of message is transmitted completely.

5: When selecting "receive only" mode, if the message received has no ending code, the interval between every packet of data sent by the sending party must be greater than the receiver's receiving Time-out span, otherwise the receiving party won't be able to distinguish between each packet of data correctly.

## FB-PLC serves as "ASCII receiver" through Port2

- FUN96:MD2 explanation of parameter Pt.

	High Byte	Low Byte	
R100	Result code	0	<ul style="list-style-type: none"> <li>The result code stores the operation result, 0=Normal; the other values, abnormal</li> </ul>
R101	For internal operation use		<ul style="list-style-type: none"> <li>For internal operation use: it is the registers required to be used when performing LINK2 instruction.</li> </ul>
R102	For internal operation use		
R103	For internal operation use		
R104	For internal operation use		<ul style="list-style-type: none"> <li>The B0 of R104 is 1 means that Port2 is being occupied, this instruction is waiting to get the control right of Port2.</li> </ul>
R105	For internal operation use		<p>B12= "ACT" indication            B13= "ERR" indication            B14= "DN" indication</p>
R106	For internal operation use		
R107	For internal operation use		
R108	Length of received data		<ul style="list-style-type: none"> <li>The total amount of data byte that has received (the register for received data length; it includes the starting and ending code that has received).</li> </ul>
R109	1		<ul style="list-style-type: none"> <li>The first Byte of received data (if there is the starting code, it is the starting code);</li> <li>High Byte = 0</li> </ul>
R110	2		<ul style="list-style-type: none"> <li>The second Byte of received data; High Byte = 0.</li> </ul>
•			
•			
•			
	N		<ul style="list-style-type: none"> <li>The N_th Byte of received data (if there is ending code, it is the ending code);</li> <li>High Byte = 0</li> </ul>

Note: When CPU received a packet of message, it filled the data to receiving registers and set up the received data length. Before the LADDER program starts to receive, you may clear the register of received data length to be 0; it means the receiving of a new packet of message when compared and found that the received data length is not zero. After the LADDER program gets the received data, it clears the received data length register to be 0. Just compare to see the received data length register is not zero means the receiving of a packet of new message, and so it may easily to process the receiving action.

Result code: 0, data transaction is successful.

2, the data length is error (the value is 0, or the transaction is greater than 511)

A, unable to reply message within Time-out span ("receive then transmit" mode).

B, communication abnormal (received error data)

## FUN96:MD2 program example

FB-PLC serves as “ASCII receiver” through Port2

- Explanation of input control

1. When the execution control input M0 change from 0→1, if Port2 is not controlled by other FUN96 (M1962 ON) and it enters into the receiving state immediately (M1962 keeping OFF all the time)
2. When "ABT" input M2 changes from 0→1, it jumps out of receiving state (M1962 ON)

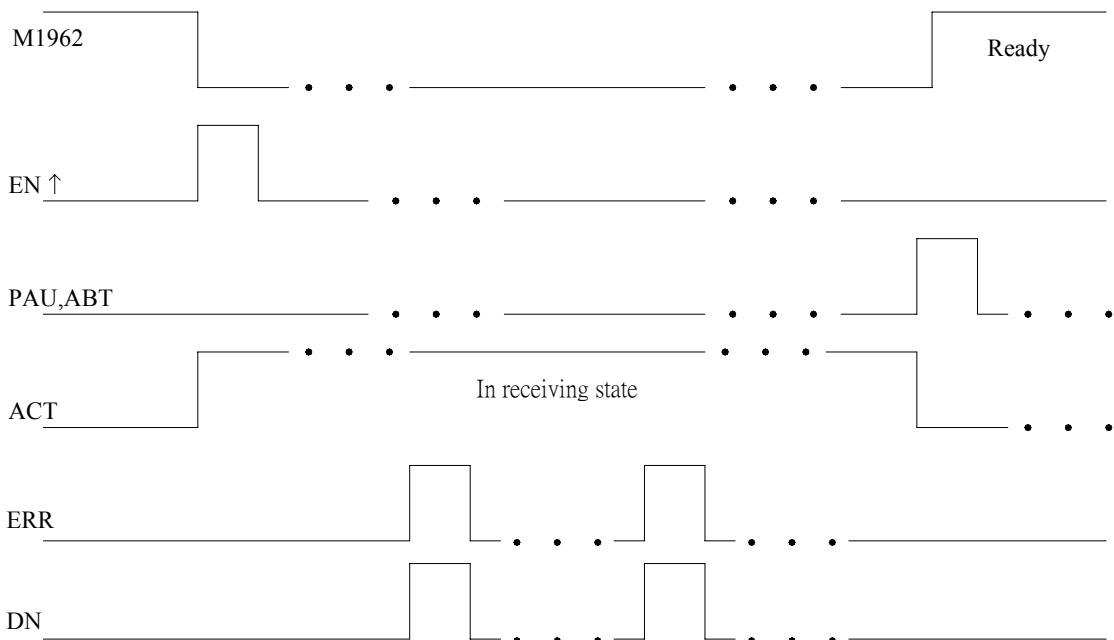
- Output indication

“ACT” ON : In receiving state

“ERR” ON : Error occurred in previous packet of transaction, it will be ON for a scan time

“DN” ON : The previous packet of transaction completed without error, ON for a scan time.

- Waveform of input control and output indication



Note: Of “ERR” and “DN”, there is only one will be ON; not both to be ON.

FUN 96 LINK2	Convenient instruction for FUN96(LINK2): MD3 communication link (it makes the PLC serve as the master of "Fatek high speed CPU LINK network" through Port2)	FUN 96 LINK2																									
	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: center;">Range</th> <th style="text-align: center;">HR</th> <th style="text-align: center;">ROR</th> <th style="text-align: center;">DR</th> <th style="text-align: center;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Ope- rand</td> <td style="text-align: center;">R0 R3839</td> <td style="text-align: center;">R5000 R8071</td> <td style="text-align: center;">D0 D3071</td> <td></td> </tr> <tr> <td style="text-align: center;">MD</td> <td></td> <td></td> <td></td> <td style="text-align: center;">0~3</td> </tr> <tr> <td style="text-align: center;">S</td> <td style="text-align: center;"><input type="circle"/></td> <td style="text-align: center;"><input type="circle"/></td> <td style="text-align: center;"><input type="circle"/></td> <td></td> </tr> <tr> <td style="text-align: center;">Pt</td> <td style="text-align: center;"><input type="circle"/></td> <td style="text-align: center;"><input type="circle"/>*</td> <td style="text-align: center;"><input type="circle"/></td> <td></td> </tr> </tbody> </table>	Range	HR	ROR	DR	K	Ope- rand	R0 R3839	R5000 R8071	D0 D3071		MD				0~3	S	<input type="circle"/>	<input type="circle"/>	<input type="circle"/>		Pt	<input type="circle"/>	<input type="circle"/> *	<input type="circle"/>		MD : 3, high speed linking between Fatek PLC and PLC S : Starting register for communication program (example illustrated). Pt : Starting register for instruction operation (example illustrated), it controls 8 registers and the other program can not repeat in use.
Range	HR	ROR	DR	K																							
Ope- rand	R0 R3839	R5000 R8071	D0 D3071																								
MD				0~3																							
S	<input type="circle"/>	<input type="circle"/>	<input type="circle"/>																								
Pt	<input type="circle"/>	<input type="circle"/> *	<input type="circle"/>																								
<b>Descriptions</b>																											
<ol style="list-style-type: none"> <li>1. FUN96(LINK2): MD3 instruction provides high speed data sharing between Fatek PLC and other PLC (data response time will not be influenced by the scan time of PLC).</li> <li>2. A master PLC can link with 254 slave PLCs at the most to share data through its built-in RS-485 interface.</li> <li>3. LINK2 instruction is required only by master PLC, not by the slave PLC.</li> <li>4. The station number of master PLC must be No.1, or it should be assigned by R4054 register if which is not No.1 but need to be as the master.</li> <li>5. The setting of M1958 for slave PLC must be ON (M1958 OFF is for non-high speed link), but it's not necessary for master PLC.</li> <li>6. In high speed linking, the maximum Baud Rate is 614.4K bps and minimum is 38.4K bps (adjustable); the data length is fixed at 8 Bits. Data is transmitted with binary code (which is twice time as fast as ASCII Code), and the error checking is adopting CRC-16, which is more reliable than Checksum.</li> <li>7. The principle of high speed linking data transmission is based upon the COMMON DATA MEMORY concept to design; e.g. as the master PLC sent out the content of R0 to R31, .the contents of R0~R31 for all the slave PLCs will be the same as the master's; when slave PLC no.2 sent out the contents of R32~R47, the R32~R47 contents of master PLC and other slave PLCs will be the same as PLC station no.2's, etc.</li> <li>8. When PLC is in STOP mode, the Port2 enters into the standard interface mode that it can connect to PROLADDER, MMI, or graphic supervisor (the communication parameter is set by R4158).</li> <li>9. It employs the program coding or table filling method to plan for data flow control; i.e. for what kind of data being sent from which PLC station to all the PLC on line, it takes only 7 registers (5 of which is being physically used, and 2 reserved) to define; every 7 registers define once communication transaction.</li> <li>10. When execution control "EN ↑" changes from 0→1 and both pause "PAU" and abort "ABT" are 0, this instruction will control Port2 and set M1962 to be "0" (being controlled) and processing the data transaction immediately, suppose the Port2 is not controlled by other FUN96 instruction (M1962=1). If Port2 is being controlled (M1962=0), this instruction will enter into wait state until the controlling FUN96 instruction complete the transmission or pause/abort the operation to release the controlling right (M1962=1); then it enacts from wait state, engages in the transmitting transaction and sets M1962 to be "0".</li> <li>11. When pause "PAU" or abort "ABT" of input is 1, it jumps out of high speed data link immediately (M1962 ON).</li> <li>12. Within the high speed linking, the output indication "ACT" is ON; Port2 is unable to accept any other FUN96 instruction.</li> <li>13. When there is error occurred while it is starting the high speed linking, the output indication "ERR" will be ON, and the high speed linking will not be performed.</li> </ol>																											

## FUN96:MD3 instruction guide

FUN 96 LINK2	Convenient instruction for FUN96(LINK2): MD3 communication link (it makes the PLC serve as the master of "Fatek high speed CPU LINK network" through Port2)	FUN 96 LINK2
-----------------	--	-----------------

### 【Interface signals】

M1958 : While in the PLC high speed data linking, slave PLC must set M1958 ON (not necessary for master PLC)  
For non high speed data linking of PLC, the slave PLC must set M1958 OFF.

M1962 : The signal is generated from CPU.

ON represents the Port2 is available for FUN96 command.

OFF represents the Port2 is engaged in high speed linking; it can't take any other FUN96 instruction.

M1963 : The signal is generated from CPU.

When M1967 is ON (this signal is controlled by the user program) and after the last packet of communication transaction is completed, the CPU sets M1962 and M1963 ON, and the high speed data transmission will be stopped; it must control "ABT" (transmission abort) to be ON, and then restart execution control "EN ↑" to change from 0→1 before the high speed linking can restart.

When M1967 is OFF (this signal is controlled by the user program), the high speed data transmission will automatically restart a new transmission from the first packet of communication transaction (M1962 and M1963 is keeping OFF state) after the last packet of communication transaction is completed.

M1967 : One-time or cycling control (controlled by the user program)

ON, one cycle, it will stop after the last packet of data transaction is performed completely.

OFF, successive cycles, it will restart from first packet of transaction when it has finished the last packet of transaction.

R4054 : It assigns the PLC station which is not no.1 to act as the master of high speed linking.

High byte	Low byte		
R4054	55	Station number.	H

When the station number of the PLC is not number 1, fills its station number (low byte of R4055 stores the station number) into the low byte of R4054 and writes to high byte of R4054 with 55H, and then controls the execution control input "EN ↑" from 0→1; even though the PLC station which is not no.1, it can still be the master station for high speed linking.

R4055 : When high byte of R4055 is not 55H, Low byte of R4055 shows the station number of PLC.

When high byte of R4055 is 55H, Low byte of R4055 defines the station number of PLC.

R4058 : Showing the station number of slave PLC which is abnormal while high speed linking (0: Represents normal; if many slave PLC were abnormal in the mean time, it is possible to see only one number; after the debugging of abnormal and clear R4058 to be 0 until the value of R4058 keeping to be 0, it will then network works normal). In communication transaction program or table, it must exist the case for slave station to send data to other stations then can the master PLC detect whether the slave station is online without error; if in the communication transaction program or table, there is only the master station sending data to slave stations, the master PLC can't detect whether slave PLC is on line without error. The user must employ programming skill to add abnormal detecting program to the master PLC and slave PLC to do the error checking (as a matter of fact, the program is very simple; just makes the PLC, which is sending data, to create an ON↔OFF variation signal. Once the receiving PLC does not detect the ON↔OFF variation signal in a period of time, it means that there is communication error).

**FB-PLC acts as the master of "Fatek high speed CPU LINK network" through Port2**

R4059 : Error logging of abnormal slave PLC while high speed linking.

High byte	Low byte
R4059	Abnormal code      Abnormal count

Low byte: Abnormal count summation

High byte: Abnormal code

  OAH, No response from slave station

  OBH, Error data (CRC Error)

  20H, rarity Error

  40H, Framing Error

  80H, Over\_Run Error

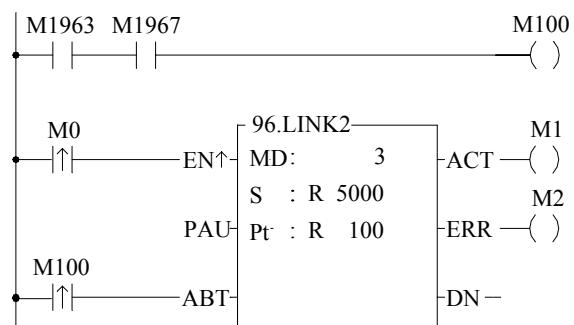
Explanation for the checking method for abnormal communication is the same as that for R4058.

R4160 : Port2 Rx/Tx Time-out setting (in high speed linking). The system will base on the setting of R4161 communication parameter to produce pertaining set point; the user need not to set it.

R4161 : communication parameter setting register for LINK2 high speed linking.

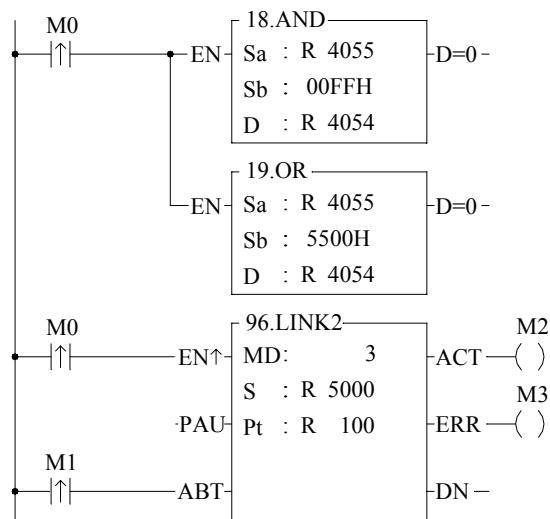
(please refer to explanation for Port2 communication parameter setting in section 12.6.2)

**Program example 1** PLC no. 1 serves as the master of high speed data linking



- Planning R5000 ~ R5199 to be ROR, the communication program will be stored together with LADDER program.
- When M1967 is ON, performs one cycle transmission. It must start the abortion, then restart M0 before it can perform high speed data link again.

**Program example 2** PLC which station number is not no.1 serves as the master of high speed data linking.



- Get PLC station number and write it into R4054
- Set the high byte for R4054 to be 55H
- Planning R5000~R5199 to be ROR, the communication program will be stored together with LADDER program.
- When ABT is not controlled, M1 instruction needs not to input.

## FUN96:MD3 program example

FB-PLC acts as the master of "Fatek high speed CPU LINK network" through Port2

**Program example 3** The same machine sets or equipments (with same LADDER program) perform multi-station data collection or distributed control through RS-485 high speed linking.

The principle for high speed data linking is based on COMMON DATA MEMORY concept to design; while designing, it must devise a successive data block and evenly distributed to respective PLCs to do data exchange among PLCs. e.g.:

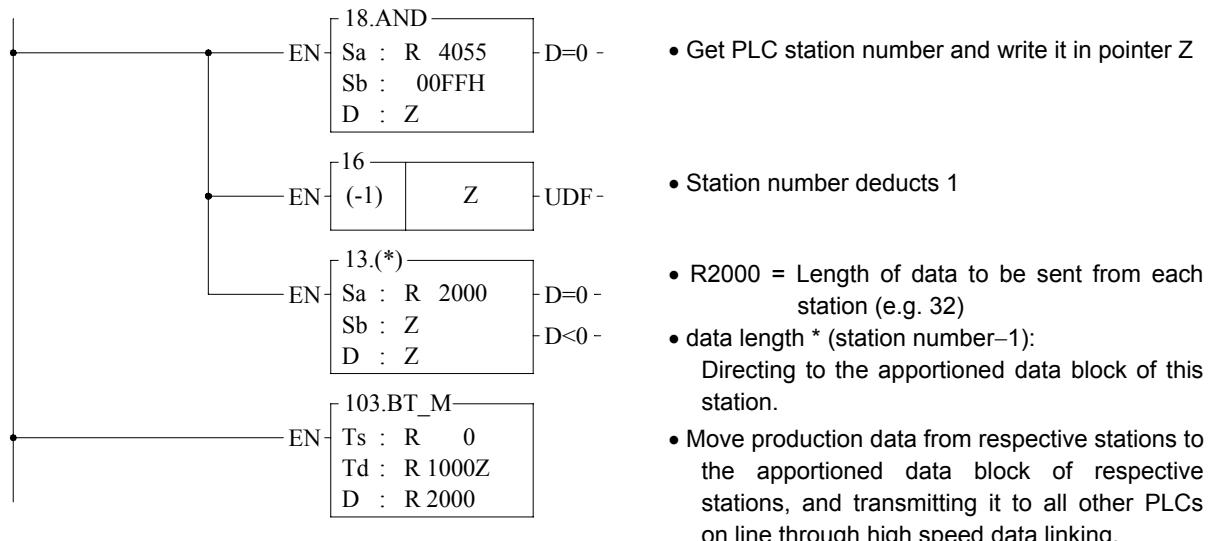
R1000～R1031: The data block of PLC no. 1 (through high speed linking, the other PLCs' content of R1000～R1031 become the same as that of PLC no.1).

R1032～R1063: The data block of PLC no. 2 (through high speed linking, the other PLCs' content of R1032～R1063 become the same as that of PLC no.2).

- 
- 
- 
- 

For example, get the production data (stored at R0～R31) from each machine set, and collectively gathering R1000～R1639 (suppose there are 20 sets linking) stored in master PLC through RS-485 high speed data linking; it needs merely the master PLC of high speed linking to connect to MMI or graphic supervisor, then it can monitor and store, for follow up processing, the production data of respective machine sets with real time effect.

Note: If it is simply for data collection and monitoring and no need to do real time control, employs the FUN96: MD0 can easily and concisely accomplish the assignment; when requiring real time control or supervising , it must employ FUN96: MD3 to accomplish a speedy, precisely controlling demand.



FB-PLC acts as the master of "Fatek high speed CPU LINK network" through Port2

● Explanation for parameter S of FUN96: MD3

R5000: Starting register for communication program (data transmission table)

R5000	Packets of data transaction	<ul style="list-style-type: none"> <li>Low Byte is valid. A packet of transmission demands 7 registers to describe; i.e. 7 registers define a packet of data.</li> </ul>	
R5001	Station number to be transmitted	<ul style="list-style-type: none"> <li>Low Byte is valid. 1~255.</li> </ul>	
R5002	Command code	<ul style="list-style-type: none"> <li>Low Byte is valid. It can only be 4 (high speed linking command).</li> </ul>	
R5003	Length of this packet of data	<ul style="list-style-type: none"> <li>Low Byte is valid. 1~32, defines the data length of one transaction.</li> </ul>	
R5004	Data type	<ul style="list-style-type: none"> <li>Low Byte is valid. 12=R; 13=D.</li> </ul>	
R5005	Data starting reference	<ul style="list-style-type: none"> <li>Word is valid. Defines starting number of working data.</li> </ul>	
R5006	Reserved	<ul style="list-style-type: none"> <li>Code for data type</li> </ul>	
R5007	Reserved	<ul style="list-style-type: none"> <li>12: R data register 13: D data register</li> </ul>	
R5008	Number of station to be transmitted	<ul style="list-style-type: none"> <li>Data starting reference 0~3839</li> </ul>	
R5009	04	<ul style="list-style-type: none"> <li>0~3071</li> </ul>	
R5010	Length of this packet of data	Description for the second packet of transmission (transaction)	
R5011	Data type		
R5012	Data starting reference		
R5013	Reserved		
R5014	Reserved		

● Explanation for parameter Pt of FUN96: MD3

	High Byte	Low Byte
R100	Result code	
R101	For internal working use	
R107	For internal working use	

Result code: 0: Correct format

- 2: Data length error (Length is 0 or greater than 32)
- 3: Command code error (Command is not equal to 4)
- 4: Data type error (Data type is not 12 nor 13)
- 5: Data reference error

## FUN96:MD3 program example

FB-PLC acts as the master of "Fatek high speed CPU LINK network" through Port2

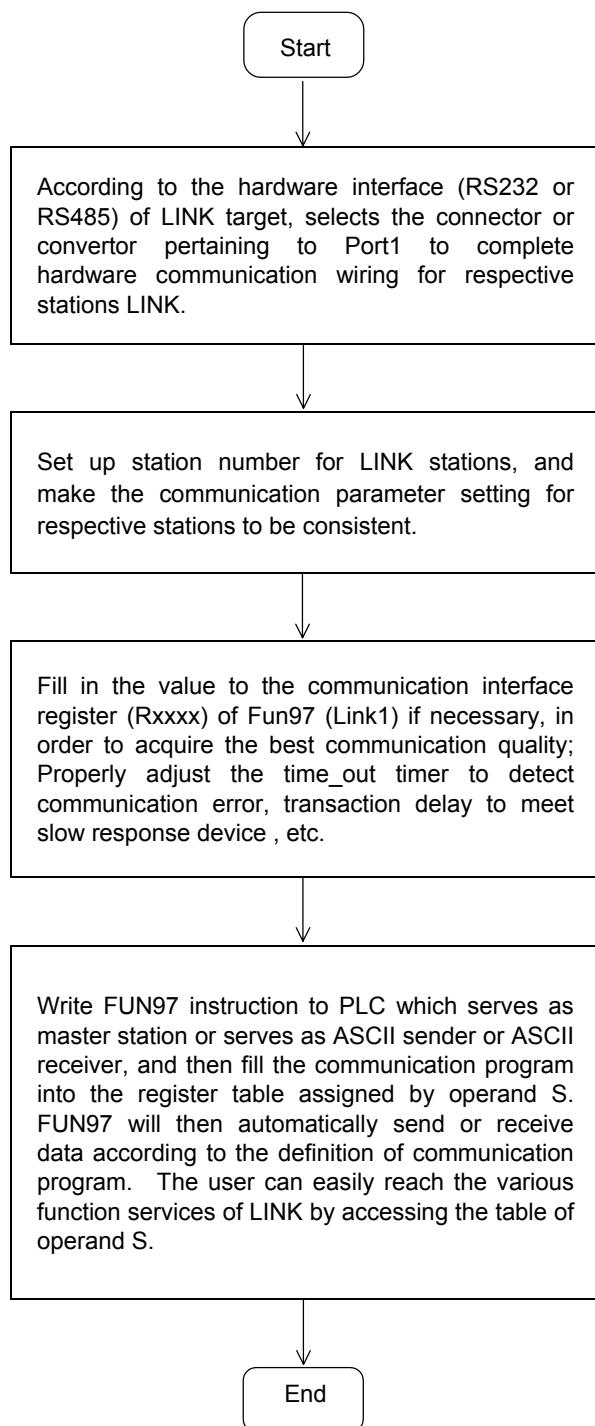
### Example for communication transaction planning

Content of registers	Description	Planning the communication transaction with extended communication instructions
R5000:4	4 packets of transaction in total	Total Sets:4 (4 packets of transaction in total)
R5001:1 R5002:4 R5003:32 R5004:12 R5005:1000 R5006: R5007:	PLC No. 1 (master PLC) High speed linking command Data length is 32 Data type is R Data number is 1000, i.e. R1000 Reserved Reserved	000 Station# 1 Command HS_Link Length 32 Start R1000
<ul style="list-style-type: none"> <li>The master PLC broadcasts R1000~R1031 to the R1000~R1031 of all stations</li> </ul>		
R5008:2 R5009:4 R5010:32 R5011:12 R5012:1032 R5013: R5014:	PLC No. 2 (slave PLC) High speed linking command Data length is 32 Data type is R Data number is 1032, i.e. R1032 Reserved Reserved	001 Station# 2 Command HS_Link Length 32 Start R1032
<ul style="list-style-type: none"> <li>PLC of station no.2 broadcasts R1032~R1063 to the R1032~R1063 of all stations</li> </ul>		
R5015:3 R5016:4 R5017:32 R5018:12 R5019:1064 R5020: R5021:	PLC No. 3 (slave PLC) High speed linking command Data length is 32 Data type is R Data number is 1064, i.e. R1064 Reserved Reserved	002 Station# 3 Command HS_Link Length 32 Start R1064
<ul style="list-style-type: none"> <li>PLC of station no.3 broadcasts R1064~R1095 to the R1064~R1095 of all stations</li> </ul>		
R5022:21 R5023:4 R5024:6 R5025:13 R5026:500 R5027: R5028:	PLC No. 21 (slave PLC) High speed linking command Data length is 6 Data type is D Data number is 500, i.e. D500 Reserved Reserved	003 Station# 21 Command HS_Link Length 6 Start D500
<ul style="list-style-type: none"> <li>PLC of station no.21 broadcasts D500~D505 to the D500~D505 of all stations</li> </ul>		

Note: For the explanation of extension instructions for communication, please refer to page13-7.

## 13.2 Application of FUN97 (port1) instruction

### 13.2.1 Procedure for FUN97 (LINK1) usage



- Please refer to communication cable (connector) of section 12.1
- Station number can be any one between 1 ~ 254 without repetition. The setting of station number may be performed under PROLADDER or function item 5 (configuration) of FP-07's system function.
- The communication parameter of Port1 is set by the value of R4146 register. Please refer to Port1 communication parameter of section 12.6.2 for details.
- Please refer to example in section 13.2.2 for definition and explanation of interface processing signal.
- Please refer to example in section 13.2.2 for definition and usage of parameter S.

### 13.2.2 Explanation of respective modes and application program example for FUN97 (LINK1)

This section illustrates with practical application program to show usages for 3 instruction modes (MD0~MD2) of FUN97(LINK1) instruction.

## FUN97:MD0 instruction guide

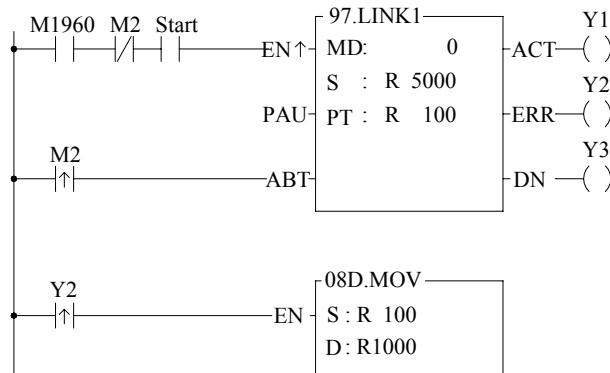
FUN 97 LINK1	Convenient instruction for FUN97(LINK1): MD0 communication network (which makes PLC as the master station in CPU LINK network through Port1)	FUN 97 LINK1																								
	<p>MD: 0, acts as master station of Fatek CPU LINK (employs Fatek communication protocol) S : Starting register for communication program (example illustrated). Pt : Starting register for instruction operation (example illustrated); it controls 8 registers, and the other programs can not repeat in using.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Range</th> <th>HR</th> <th>ROR</th> <th>DR</th> <th>K</th> </tr> <tr> <td rowspan="2">Operand</td> <td>R0   R3839</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td></td> </tr> <tr> <td>MD</td> <td></td> <td></td> <td>0~2</td> </tr> <tr> <td>S</td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td><input type="radio"/></td> <td></td> </tr> <tr> <td>Pt</td> <td><input type="radio"/></td> <td><input checked="" type="radio"/></td> <td><input type="radio"/></td> <td></td> </tr> </table>	Range	HR	ROR	DR	K	Operand	R0   R3839	R5000   R8071	D0   D3071		MD			0~2	S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		Pt	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>		
Range	HR	ROR	DR	K																						
Operand	R0   R3839	R5000   R8071	D0   D3071																							
	MD			0~2																						
S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																							
Pt	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>																							
<b>Descriptions</b>																										
<ol style="list-style-type: none"> <li>1. FUN97(LINK1):MD0 instruction provides data sharing among the Fatek PLCs.</li> <li>2. A master PLC can pass through RS-485 (FB-485) interface to connect with 254 slave PLCs and share data with each other.</li> <li>3. Only the SW1 of master PLC CPU board has to be set as 1=OFF, 2=ON (turn off setting and restart).</li> <li>4. Only the master PLC needs to employ LINK1 instruction, the slave PLCs need not to.</li> <li>5. It employs the program coding method or table filling method to plan for the data flow controls; i.e. from which one of the slave PLC to get which type of data and save them to the master PLC, or from the master PLC to write which type of data to the assigned slave PLC. It needs only seven registries to make definition; every seven registers define one packet of data transaction.</li> <li>6. When execution control “EN ↑” changes from 0→1 and both inputs Pause “PAU” and Abort “ABT” are 0, and if Port1 hasn’t been controlled by other FUN97 instructions (i.e. M1960 = 1), this instruction will control the Port1 immediately and set the M1960 to be “0” (which means it is being occupied), then going on a packet of data transaction immediately. If Port1 has been controlled (M1960 = 0), then this instruction will enter into the standby status until the controlling FUN97 instruction complete its transaction or pause/abort its operation to release the control right (M1960=1), and then this instruction will become enactive, set M1960 to be 0, and going on the data transaction immediately.</li> <li>7. While in transaction processing, if operation control “PAU” becomes 1, this instruction will pause and release the control right (M1960 set to be 1) after it finishes the on going transaction. Next time, when this instruction takes over the transmission right again , it will keep going on the next packet of data transaction (this means that the pause operation is based on a packet of data transaction).</li> <li>8. While in transaction processing, if operation control “ABT” becomes 1, this instruction will halt immediately and release the control right (M1960 set to be 1). Next time, when this instruction takes over the transmission right again, it will restart from the first packet of data transaction.</li> <li>9. While it is in the data transaction, the output indication “ACT” will be ON.</li> <li>10. If there is error occurred when it finishes a packet of data transaction, the output indication “ERR” will be ON.</li> <li>11. If there is no error occurred when it finishes a packet of data transaction, the output indication “DN” will be ON.</li> <li>12. The connecting pin No.3(RTS) of Port1 must be short to connect to pin No.4(CTS).</li> </ol>																										

FUN97 LINK1	Convenient instruction for FUN97(LINK1): MD0 communication network (which makes PLC as the master station in CPU LINK network through Port1)	FUN97 LINK1
<b>【Interface processing signal】</b>		
M1960 : This signal is generated from CPU. ON, it represents that Port1 is free and ready. OFF, it represents that Port1 is occupied, data transaction is going.		
M1961 : This signal is generated from CPU. When the communication program completed the last packet of data transaction, M1961 will be ON for a scan time (for successive data transaction). When the communication program completed the last packet of data transaction, M1961 will be ON (for single packet of data transmission).		
R4146 : The register for communication parameter setting of Port1. (please refer to explanation for Port1 communication parameter setting of section 12.6.2)		
R4147 : The content of Low Byte defines the Time-out span of LINK1 instruction; its unit is 0.1 second. (the default is 5, which means 0.5 second) The LINK1 instruction (only master PLC needs) employs Time-out span to judge whether the slave PLC on line or not. When the master PLC sent out the read/write command to the slave PLC, the slave PLC didn't reply within this period means that there is abnormal event in communication called Time-out. When there are multi-PLCs linking, properly adjust this value (greater than 1 scan time of the slave PLC with the longest scan time) to shorten the communication response time among the active linking PLCs if there are many slave PLCs power off (The time-out cases will happen).  : The content of High Byte defines the transmission delay time between two packets of data transaction for LINK2 instruction; its unit is 0.01 second (the default is 0). For point-to-point link, this value can be set as 0 to shorten the communication transaction time and promote the communication efficiency. In the case of linking multi-PLCs and if the scan time of master PLC is far longer than any slave PLC, this value can be set to 0 to shorten the communication transaction time and promote the communication efficiency. When there are multi-PLCs to link in parallel by using master/slave method and the scan time of master PLC is close to that of slave PLCs, it must properly adjust this value (greater than 1 scan time of the slave PLC with the longest scan time) to reach the best, error-free communication quality.		
R4148 : When Low Byte of R4147 is not 0, Low Byte of R4148 makes no effect. When Low Byte of R4147 is 0, Low Byte of R4148 defines the Time-out span of LINK1 instruction, the unit is 0.01 second (for fine tuning ;the default is 0). The function is identical to explanation for R4147 low byte.		

## FUN97:MD0 Program Example

FB-PLC acts as the master of “Fatek CPU LINK network” through Port1 and FB-485

### Program example Automatic cycling transmission



- Configure R5000~R5199 as the read only register (ROR) before programming, after then, when storing program, the ladder program will automatically contains the communication program .
- When ABT is not controlled, it is not necessary to input the M2 contact instruction.
- When there is communication error, gets and stores the error message to R1000 & R1001 would be helpful for error analysis or logging.

### Explanation

●Explanation of parameter S for FUN97: MD0 (R5000 just only for example, other registers can be used also).

R5000: Starting register of communication program (data transaction table) by filling table method (Not easy)	
R5000	Total transactions
R5001	Slave station No. which is about to transact with
R5002	Command code
R5003	Data length of this transaction
R5004	Data type of Master PLC
R5005	Starting reference of Master PLC
R5006	Data type of slave PLC
R5007	Starting reference of Slave PLC
R5008	Slave station No. which is about to transact with
R5009	Command code
R5010	Data length of this transaction
R5011	Data type of Master PLC
R5012	Starting reference of Master PLC
R5013	Data type of slave PLC
R5014	Starting reference of Slave PLC

} Description of the 2\_nd packet of transaction

FB-PLC acts as the master of “Fatek CPU LINK network” through Port1 and FB-485

●Master/Slave data type, code and reference number

Data code	Data type	Starting code
0	X (discrete input)	0~255
1	Y (discrete output)	0~255
2	M (internal relay M)	0~1911
3	S (step relay S)	0~999
4	T (timer contact)	0~255
5	C (counter contact)	0~255
6	WX (word of discrete input, 16 bits)	0~240, it must be the multiple of 8.
7	WY (word of discrete output, 16 bits)	0~240, it must be the multiple of 8.
8	WM (word of internal relay, 16 bits)	0~1896, it must be the multiple of 8.
9	WS (word of step relay, 16 bits)	0~984, it must be the multiple of 8.
10	TR (timer register)	0~255
11	CR (counter register)	0~199
12	R (data register Rxxxx)	0~3839
13	D (data register Dxxxx)	0~3071

Note: The data type for master and slave must be consistent. i.e. if the master station is any value between 0 to 5, the slave station must also be any value between 0 to 5; if the master station is any value between 6 to 13, the slave station must also be any value between 6 to 13.

●Explanation for operand Pt of FUN97:MD0 (R100 just only for example, other registers can be used also)

	High Byte	Low Byte
R100	Result code	Transaction No.
R101	Station number	Command code
R102	For internal working use	
R103	For internal working use	
R104	For internal working use	
R105	For internal working use	
R106	For internal working use	
R107	Internal working usage	

- Result code indicates the transaction result; 0= normal, other value =abnormal.
- Transaction No. indicates which one is in processing (beginning from 0).
- Station number: the slave station No. which is in transaction.
- Command code  
 =44H, reading successive discrete status from slave PLC.  
 =45H, writing successive discrete status to slave PLC.  
 =46H, reading successive registers from slave PLC.  
 =47H, writing successive registers to slave PLC.
- R104's B0=1, Port1 has been occupied and this instruction is waiting to acquire the transmission right for data transaction.
- B4=1, t this instruction is not first time performing.
- B12, output indication for “ACT”
- B13, output indication for “ERR”.
- B14, output indication for “DN”.

Result code: 0, transaction is successful.

- 1, the setting of CPU DIP switch (SW1) is error (it must be 1=OFF, 2=ON), turn off and set as describing.
- 2, data length error (data length is 0 or greater than 64 in one transaction).
- 3, command code error (command code is greater than 2).
- 4, data type error (data type is greater than 13, please refer to data type code).
- 5, reference number error (please refer to reference number).
- 6, inconsistency in data type (e.g. master station is 0~5 while slave is 6~13).
- A, communicating, but no response from slave station (Time-out error).
- B, communication error (received error data).

## FUN97:MD0 Program Example

FB-PLC acts as the master of “Fatek CPU LINK network” through Port1 and FB-485

### Programming for data transaction with instruction method

(please refer to extension instructions for communication)

R5000: Starting register of communication program (It's very easy to plan the data flow by this method)

Content of registers	Description	Planning the transaction with extended instructions
R5000:5	5 packets of transactions in total.	Total transactions:5
R5001:0 R5002:2 R5003:16 R5004:12 R5005:500 R5006:13 R5007:0	Broadcasting from master PLC Write data to all slave PLCs Length of data is 16 Data type of master PLC is R Reference number of master PLC is 500, i.e. R500 Data type of slave PLC is D Reference number of slave PLC is 0, i.e. D0	000 Station# 0 Command Write Length 16 M_start R500 S_start D0
<ul style="list-style-type: none"> <li>Master PLC broadcasts the R500~R515 to all slave PLCs' D0~D15</li> </ul>		
R5008:2 R5009:1 R5010:10 R5011:12 R5012:20 R5013:12 R5014:200	The slave PLC in transaction is the station No.2 Read data from slave PLC Data length is 10 Data type of master PLC is R. Reference number of master PLC is 20, i.e. R20 Data type of slave PLC is R Reference number of slave PLC is 200, i.e. R200	001 Station# 2 Command Read Length 10 M_start R20 S_start R200
<ul style="list-style-type: none"> <li>Read R200~R209 from slave PLC No.2 to R20~R29 of master PLC</li> </ul>		
R5015:3 R5016:1 R5017:20 R5018:2 R5019:1000 R5020:2 R5021:100	The slave PLC in transaction is the station No.3 Read data from slave PLC Data length is 20 Data type of master PLC is M. Reference number of master PLC is 1000, i.e. M1000 Data type of slave PLC is M Reference number of slave PLC is 100, i.e. M100	002 Station# 3 Command Read Length 20 M_start M1000 S_start M100
<ul style="list-style-type: none"> <li>Read M100~M119 from slave PLC No.3 to M1000~M1019 of master PLC</li> </ul>		
R5022:4 R5023:2 R5024:20 R5025:2 R5026:1000 R5027:3 R5028:100	The slave PLC in transaction is the station No.4 Write data to slave PLC Data length is 20 Data type of master PLC is M. Reference number of master PLC is 1000, i.e. M1000 Data type of slave PLC is S Reference number of slave PLC is 100, i.e. S100	003 Station# 4 Command Write Length 20 M_start M1000 S_start S100
<ul style="list-style-type: none"> <li>Master PLC writes M1000~M1019 to S100~S119 of slave PLC No.4, i.e. to write from M100~M119 of slave PLC No. 3 to S100~S119 of slave PLC No.4</li> </ul>		
R5029:4 R5030:1 R5031:4 R5032:9 R5033:0 R5034:6 R5035:0	The slave PLC in transaction is the station No.4 Read data from slave PLC Data length is 4 (4 words this situation) Data type of master PLC is WS. Reference number of master PLC is 0, i.e. WS0 Data type of slave PLC is WX Reference number of slave PLC is 0, i.e. WX0	004 Station# 4 Command Read Length 4 M_start WS0 S_start WX0
<ul style="list-style-type: none"> <li>Read X0~X63 of slave PLC No.4 to S0~S63 of master PLC</li> </ul>		

Note: For explanation of extended instruction for communication format, please refer to page 13-7.

FB-PLC acts as the master of “Fatek CPU LINK network” through Port1 and FB-485

**Explanation of program example**

1. When execution control “EN ↑” changes from 0→1, and Port1 is not occupied by other FUN97 (M1960 ON) and M2=OFF, LINK1 instruction will start the data transaction. The M1960 is OFF during data transaction, and when the transaction is finished, the M1960 becomes ON. Employ the OFF↔ON change of M1960 (FUN97 execution control “EN ↑”=0→1 means starting) may automatically starts for every frame of data transaction successively (when the last packet of transaction is completed, it will automatically return to the first packet of transaction to obtain the automatic cycling transmission).
2. When abort control M2 changes from 0→1, it aborts transmission immediately (if the data is in transmitting, it will stop transmitting immediately). Next time when starts the transaction; it will begin from the first packet of transactions.

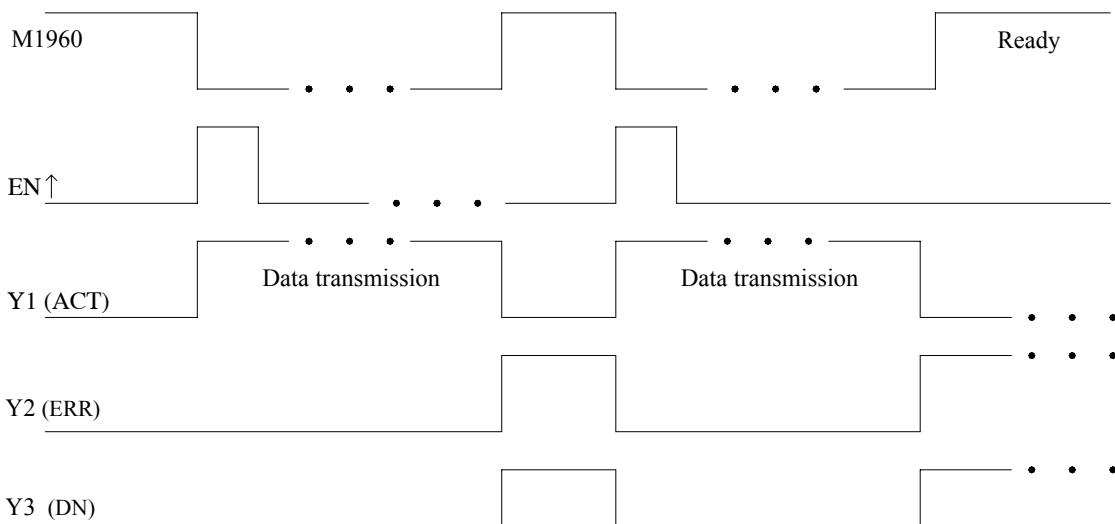
● Output indication

“ACT” ON : The Y1 ON, transaction is going

“ERR” ON: The Y2 ON, error occurred in previous packet of transaction (refer to result code).

“DN” ON: The Y3 ON, previous packet of transaction is completed and is error free.

● Waveform of input control and output indication



Note 1: Of Y2 and Y3, only one of them will be in ON status and not both to be ON at the same time.

2: After the last packet of transaction completed, the M1961 will be ON for one scan time..

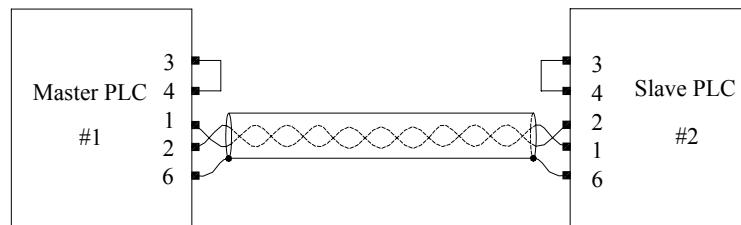
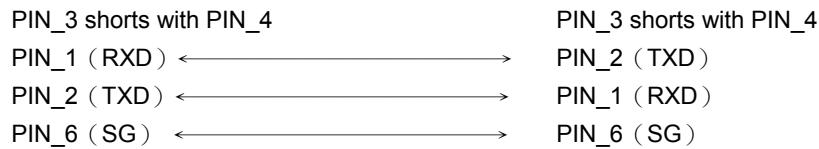
## FUN97:MD0 Program Example

FB-PLC acts as the master of “Fatek CPU LINK network” through Port1 and FB-485

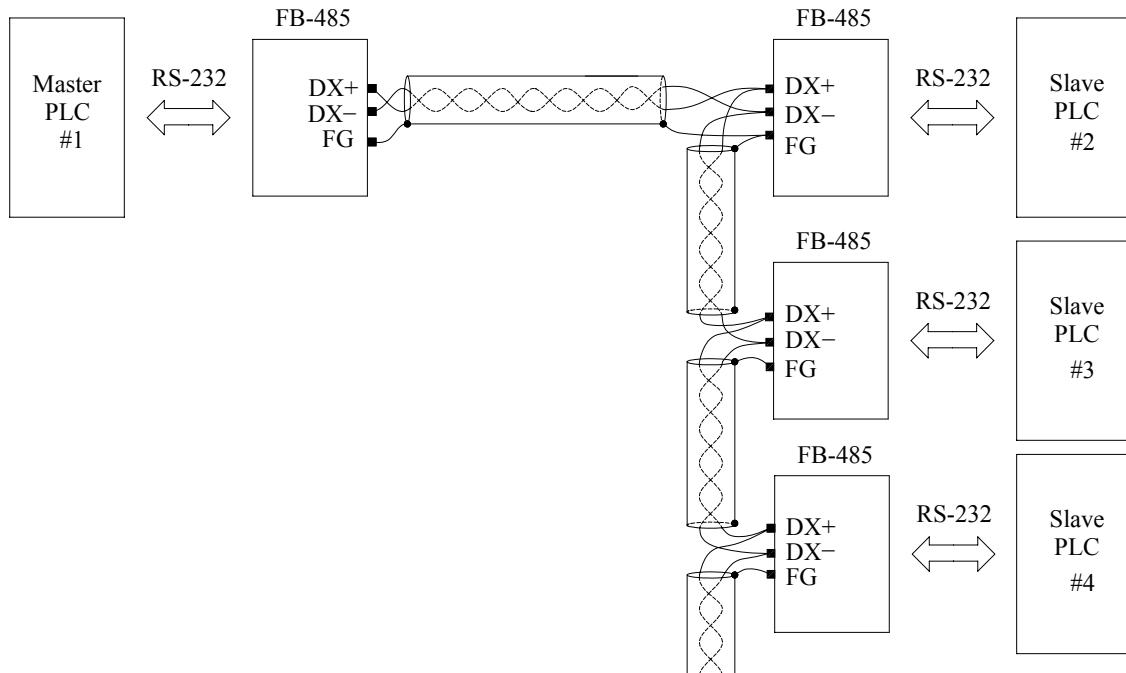
**Point to point wiring** Point to point link of master PLC and slave PLC through RS-232.

The communication port of PLC is a 15 Pin D-Sub female connector, therefore a 15 Pin D-Sub cable with both ends to be male connector is needed to link the PLCs.

The connecting is as follows:



**Multi\_drop wiring** Pass through FB-485 (RS-232↔RS-485) converter, the master PLC paves the data link with multi slave PLCs by way of RS-485 network.

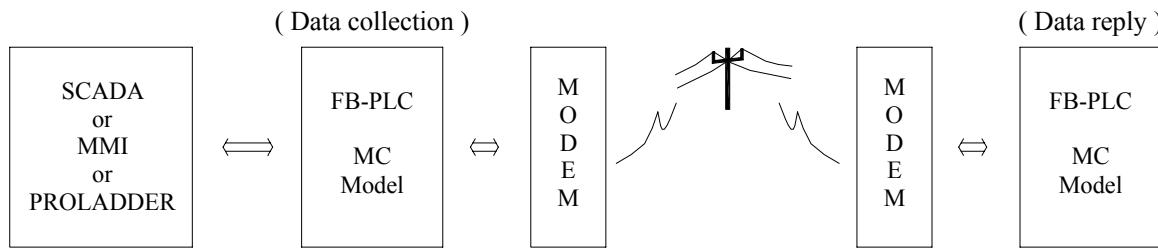


### 【Cautions】

- 1.The RS-485 wiring must employ twisted pair as the transmission cable.
2. Star topology of the wiring must be avoided; it must be cascaded with stations one after one.
3. The outer layer of weaved net for twisted pair must connect to the FG (to prevent from interference and decrease the common mode interference).

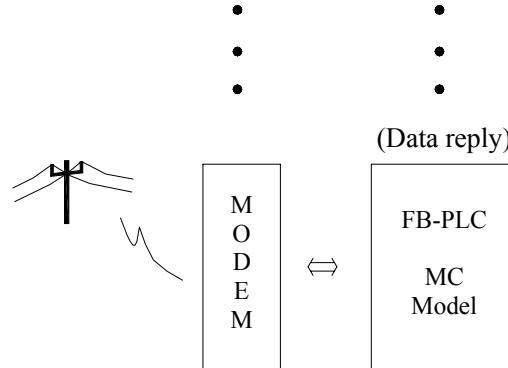
## CPU Link by way of Port1 to connect to Modem.

- PLC can connect to MODEM through communication port1, and by way of telecommunication network to link and share data with remote PLC. Its application is as follows:
  - Perform automatic data collection from the remote end.
  - Automatically report for alarm and abnormal conditions
  - Associate with current available graphic supervising software or MMI etc. standard products to constitute a wide area network automatic monitoring system. It doesn't need to develop specific designing, so as to reduce the development risk and time limit.
- Hardware configuration, and setting:



## Data collecting PLC:

- DIP switch (SW1) setting for CPU
  - 1:OFF
  - 2:ON (LINK function)
- Don't need to store phone number within the CPU
- R4149 High Byte set to be 55H (MODEM function)



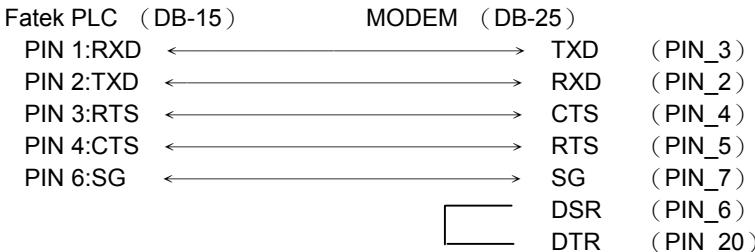
## Data reply PLC:

- DIP switch (SW1) setting for CPU
  - 1:OFF
  - 2:ON (LINK function)
- R4149 High Byte sets to be 55H (MODEM function)
- R4140~R4145 sets the phone number for general data collecting PLC end (extension phone function allowed). e.g. Phone number is 02-28082192, then R4140=8220H, R4141=1280H, and R4142=0E29H.  
If phone number is: 02-28082192 ext 100, then R4140=2A20H, R4141=2808H, R4142=A291H, R4143=AAAAH, R4144=001AH, R4145=000EH.
- Explanation: R4140~R4145 is telephone number register for dialing;  
“E” is the ending character of phone number; “A” is the dial delaying character (usually the dialing of extension number or international long distance call can be reached by making use of dial delaying, the delayed time for a delaying character is based on MODEM setting, which is about 2 second). “B” stands for “#” character (can dial B. B. CALL), and “C” stands for “\*” character.
- It employs LINK1 (FUN97:MD0) instruction to write data to the general data collecting PLC or to read data from general data collecting PLC (refer to LINK1 Instruction user guide).
- The maximum communication Baud Rate can reach 38400 bps  
(both of the communication ends must be consistent in setting)
- This configuration does not offer calling back function.

## FUN97:MD0 Program Example

### CPU Link by way of Port1 to connect to Modem.

- The wiring of PLC communication port 1 and MODEM:



#### 【MODEM dialing interface signal】

M1959: OFF, dialing by "Tone"      ON, dialing by "Pulse"

M1964: OFF→ON, dial up      ON→OFF, hang up

R4163: The Low Byte of R4163 is used to control the application of X instruction while MODEM dialing.

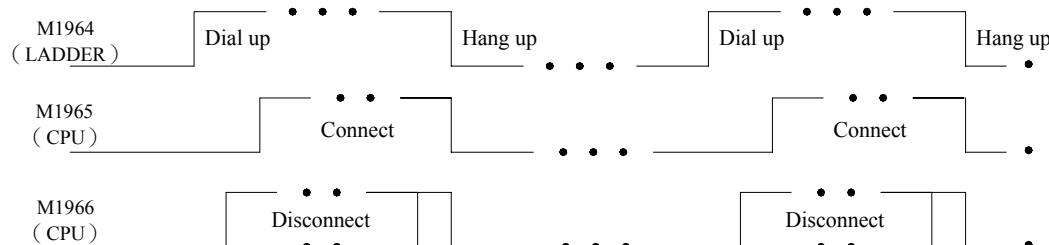
=1, it does not detect dial tone nor busy tone while MODEM dialing.

=2, it detects only dial tone but does not detect busy tone while MODEM dialing.

=3, it dials directly without detecting dial tone, but will detect busy tone after MODEM dialing.

=4, it detects both dial tone and busy tone for MODEM dialing.

For the other values, it works as 4; different country system needs to adjust the setting pertaining to the country.



Note 1: Of M1965 and M1966, there will be only one ON, not both to be ON at the same time.

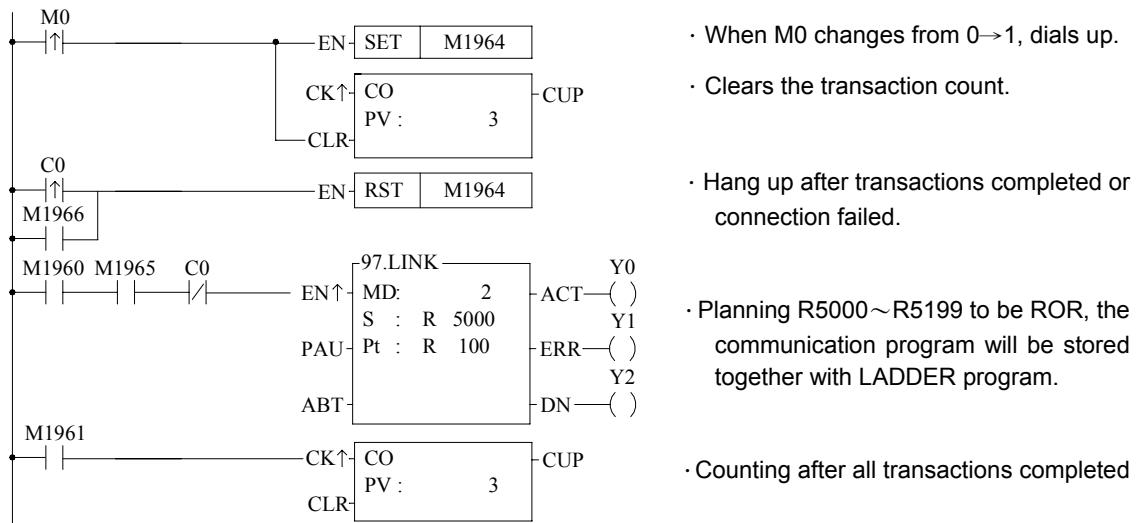
2: The waiting time for dial connection is 1 minute; if unable to connect, it will redial twice (totally 3 times). If all of the dial connection tries failed, CPU will set M1966 to be ON (connection failed).

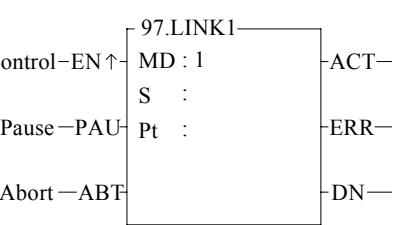
3: When the quality of communication is not stable and easy to disconnect, you may employ the abnormal detecting function of LINK1 instruction to control M1964 redials for connection (delay time of redial must be more than 10 seconds).

4: When PLC change from RUN to STOP, the CPU will automatically change MODEM to be receiving state, which could accept the remote side dial connection.

5: When PLC is not in dialing or MODEM connection states, CPU will automatically change MODEM to be receiving state, which could accept the remote side dial connection.

#### ⟨Program example⟩



FUN 97 LINK1	Convenient instruction for FUN97(LINK1): MD1 (Which makes PLC serve as "ASCII sender" through Port1)	FUN 97 LINK1
		<p>MD : 1, link with intelligent peripherals that equipped with ASCII interface.</p> <p>S : Starting register of data transmission table (see example for explanation)</p> <p>Pt : Starting register for instruction operation (see example for explanation). It controls 8 registers at least, the other programs cannot repeat in using.</p>
<p><b>Descriptions</b></p> <ol style="list-style-type: none"> <li>1. FUN97 (LINK1): MD1 instruction provides the Fatek PLC to act as the ASCII sender to link with the intelligent peripherals that equipped with ASCII interface.</li> <li>2. The SW1 of CPU board must be set to 1=OFF, 2=ON (shut down setting to restart).</li> <li>3. Port1 is RS-232 interface, if it is going to link to multi stations through RS-485 interface, just append an FB-485 converter (transform RS-232 to RS-485) and it will work.</li> <li>4. The communication protocol/format is written with LADDER program, which must be consistent with the linked ASCII peripherals.</li> <li>5. When execution control “EN ↑” turns from 0→1 and both pause “PAU” and abort “ABT” are 0, and if Port1 is not controlled by other FUN97 instruction (which means M1960=1), this instruction will control Port1 immediately and set M1960 to be “0” (being controlled) to proceed data transaction. If Port1 is being controlled (M1960=0), this instruction will enter into the wait state until the other controlling FUN97 instruction complete or pause/abort its operation and released the control right (M1960=1), and this instruction will enact again out of wait state to set the M1960 to be “0” and proceed the transmission transaction.</li> <li>6. During transaction, if the pause “PAU” becomes 1, this instruction will pause and release the control right (set M1960 to be 1) after it completed the transmitting of the on-going data transmission.</li> <li>7. During transaction, if the abort “ABT” becomes 1, this instruction will halt the transmission and release the control right immediately (set M1960 to be 1).</li> <li>8. While transaction is going, the output indication “ACT” will be ON.</li> <li>9. When a packet of data transaction is finished (transmission finished or "transmit then receive" completed), if there is error occurred, the output indication “ERR” will be ON.</li> <li>10. When a packet of data transaction is finished (transmission finished or "transmit then receive" completed), if there is no error occurred, the output indication “DN” will be ON.</li> <li>11. The connecting pin No.3 (RTS) of Port1 must be short to connect to pin No.4 (CTS).</li> </ol>		

Convenient instruction for FUN97 (LINK1): MD1  
(Which makes PLC serve as "ASCII sender" through Port1)

【Interface signal】

M1960 : This signal is generated from CPU

ON means Port1 is ready.

OFF means Port1 is busy.

M1961 : This signal is generated from CPU; the same as M1960.

ON, it means data transaction has been completed.

R4146 : The register for communication parameter setting of port 1. (please refer to section 12.6.2 for communication parameter setting)

R4147 : Low byte of R4147, it defines the Time-out span of link1 instruction; the unit is 0.1 second (the default is 05H, i.e. 0.5 second)

LINK1 instruction depends on Time-out span to detect whether the communication partner is free from error on line; when the LINK1 MD1 setting is in "transmit then receive" mode (example will be followed), the Time-out error will occur if PLC sent a packet of data to the peripheral but it didn't reply within this duration.

When LINK1 MD1 setting is "transmit" only (example will be followed), low byte of R4147 is meaningless.

: High byte of R4147, for FUN97:MD1, the recommended setting is 0.

R4148 : When the low byte of R4147 is not 0, the low byte of R4148 is meaningless.

When R4147 low byte is 0, the low byte of R4148 defines the Time-out span of LINK1 instruction, the unit is 0.01 second (for fine tuning; the default is 0). Its function is the same as that described for R4147 low byte.

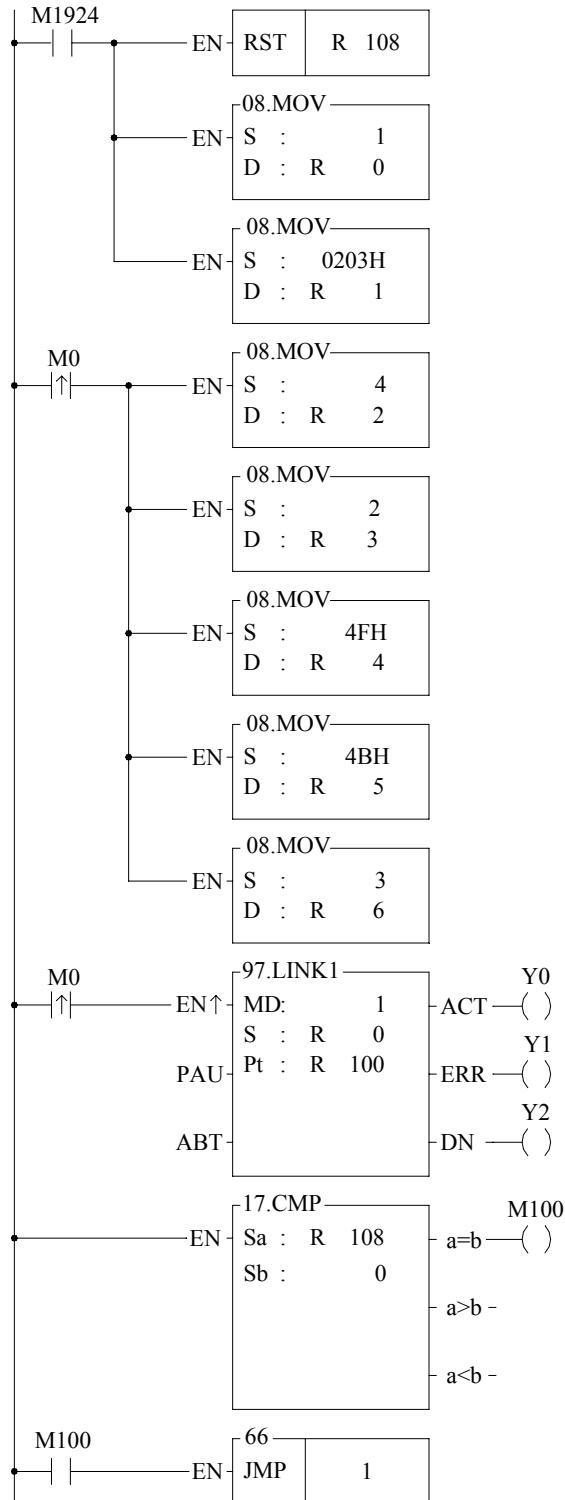
R4148 : High byte of R4148, Time-out setting for receiving, which is used to determine whether a packet of data has been received completely. The unit is 0.001 second and default is 0CH(12mS).

Detailed description will be followed.

## FB-PLC acts as an "ASCII sender" through Port1

## Program example for loop back test

PLC station A sends data to PLC station B (PLC station B sends the received original data back to the PLC station A, loopback test), and checks whether the responding message of PLC station B is the same as its original data that had sent out; therefore, it can do simple test on software and hardware of PLC Port1 whether it is normal and error free.

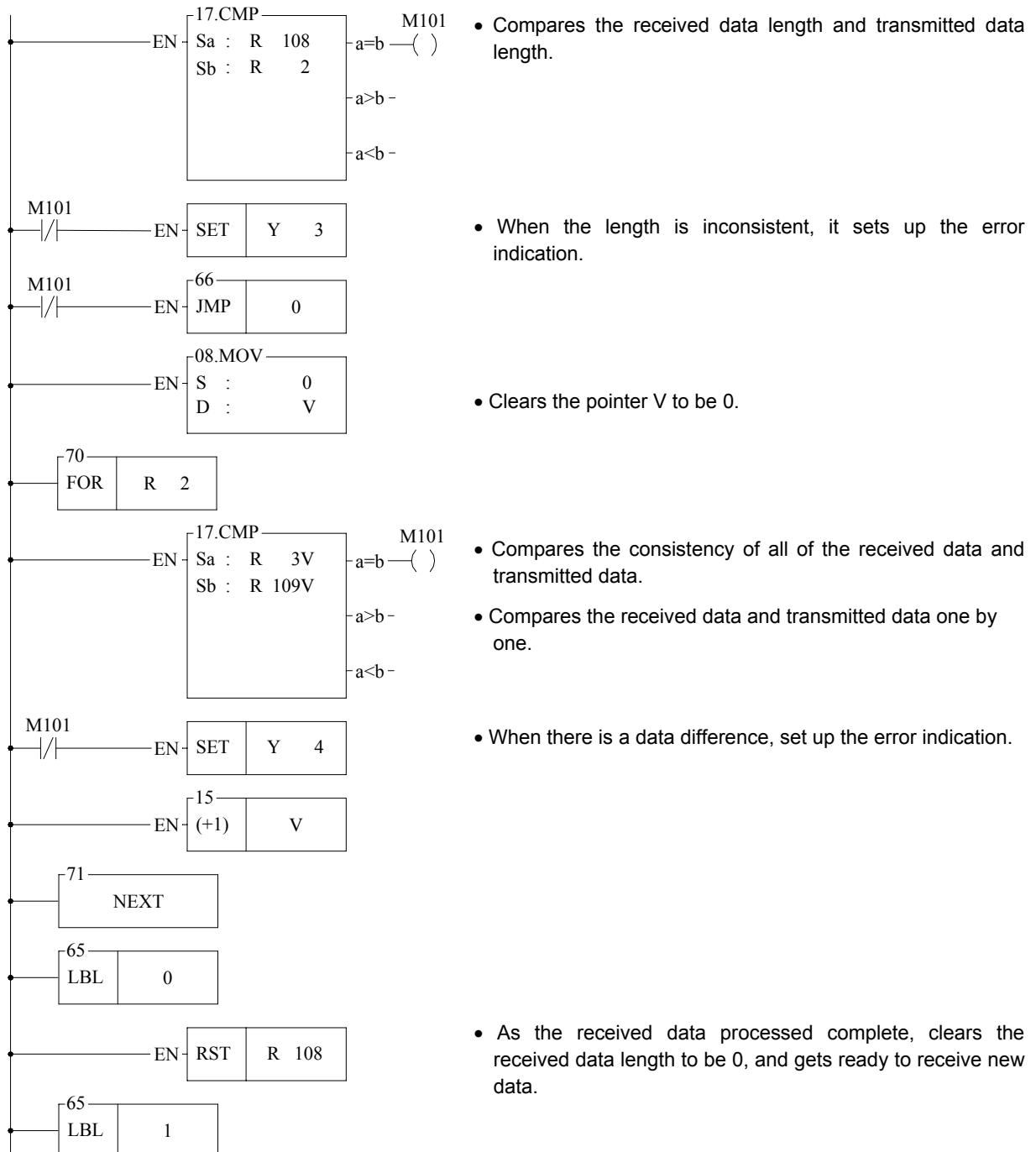


- Clears the received data length to be 0 (for "transmit" only, this instruction is not required).
- Setting of the operation mode:
  - Set to be "transmit then receive" mode (R0=1)
- Set the starting code (02H) and ending code (03H) for responding message in receiving. (without starting and ending codes, R1=0 can also receive regularly)
- Packing data to be transmitted:
  - Set the transmitting data length (R2=N).
- Fills in the data that is to be transmitted:
  - Fill in data 1 (R3=' STX' )
  - Fill in data 2 (R4=' O' )
  - Fill in data 3 (R5=' K' )
  - Fill in data 4 (R6=' ETX' )
- When selecting "transmit then receive" mode, it employs the comparing instruction to judge whether the responding message from the counter partner is received; if it is received, then M100=OFF, and it will process the received data.  
(For "transmit" mode, this program is not required)

## FUN97:MD1 program example

FB-PLC acts as an “ASCII sender” through Port1

- The processing program for data received.
- For details of the data received, please refer to the explanation of following page.



**FB-PLC acts as an “ASCII sender” through Port1**

- Explanation of parameter S for FUN97: MD1

Starting register of data transmission table (R0 just only for example)

R0	Transmit only/transmit then receive	<ul style="list-style-type: none"> <li>Low byte is valid, 0: transmit only, no response from the counter partner. 1: transmit then receive the responding message.</li> </ul>
R1	Starting /Ending code of receiving	<ul style="list-style-type: none"> <li>High Byte : Describing the starting code of responding message while receiving Low Byte : Describing the ending code of responding message while receiving.</li> </ul>
R2	Length of transmission	<ul style="list-style-type: none"> <li>The maximum length of data to be transmitted is 511.</li> </ul>
R3	Data 1	<ul style="list-style-type: none"> <li>Low Byte is valid</li> </ul>
R4	Data 2	<ul style="list-style-type: none"> <li>Low Byte is valid</li> </ul>
R5	Data 3	<ul style="list-style-type: none"> <li>Low Byte is valid</li> </ul>
R7	Data 4	<ul style="list-style-type: none"> <li>Low Byte is valid</li> </ul>
•	•	<ul style="list-style-type: none"> <li>Low Byte is valid</li> </ul>
•	•	<ul style="list-style-type: none"> <li>Low Byte is valid</li> </ul>
		<ul style="list-style-type: none"> <li>Low Byte is valid</li> </ul>
	Data N	<ul style="list-style-type: none"> <li>Low Byte is valid</li> </ul>

Note 1: When selecting the transmit-only mode, the Starting /Ending code of receiving is meaningless.

- 2: When it is in the "transmit then receive" mode, before the starting of transmission, it must first to estimate the starting and ending code of responding message from communication partner and write them into the receiving starting/ending code register (e.g. R1=0203H, 02H stands for starting code and 03H for ending code), so as to ensure the receiving to be free from error. The communication protocol with starting/ending code makes the identifying of every packet of messages easy, and the communication program is simple and efficient.
- 3: When it is in the "transmit then receive" mode, fills the high byte of starting/ending code register with 0 if no starting code in responding message; if no ending code in responding message, fills 0 to the low byte of starting/ending code register. Adjusts the high byte of R4148 (Time-out span) to judge whether a packet of data has been received completely; the unit is 0.001 second (the default is 0CH, 12mS). The communication protocol without ending code depends on Time-out span to tell whether it has received completely a packet of data (the setting of Time-out must be greater than the maximum response delay time between data bytes when communication partner is replying), thus it may ensure the receiving of the whole packet to be complete. Generally speaking, the data in transmitting is transmitted one byte after another continuously; therefore, if there is pause (greater than Time-out duration), it means the packet of message is transmitted completely.

## FUN97:MD1 program example

FB-PLC acts as an “ASCII sender” through Port1

- Explanation of FUN97: MD1 parameter Pt.

	High Byte	Low Byte
R100	Result code	0
R101	For internal operation use	
R102	For internal operation use	
R103	For internal operation use	
R104	For internal operation use	
R105	For internal operation use	
R106	For internal operation use	
R107	For internal operation use	
R108	Total amount of data received	
R109	1	
R110	2	
•	3	
•		
•		
	N	

- The result code stores the operation result, 0=Normal; the other values, Abnormal.
- For internal operation use: it is the registers required by CPU when performing LINK1 instruction.
- The B0 of R104 is 1 means that Port1 is busy; this instruction is waiting to take the transaction right
  - B12= “ACT” output indication
  - B13= “ERR” output indication
  - B14= “DN” output indication.
- The total amount of data byte that is received (the register for received data length; it includes the starting and ending code that is received).
- The first byte of data received (if there is the starting code, it is the starting code); High Byte =0.
- The second byte of data received; High Byte =0.
- The third byte of data received; High Byte =0.
- The N\_th byte of data received (if there is ending code, it is the ending code); High Byte =0.

Result code: 0, transaction is successful.

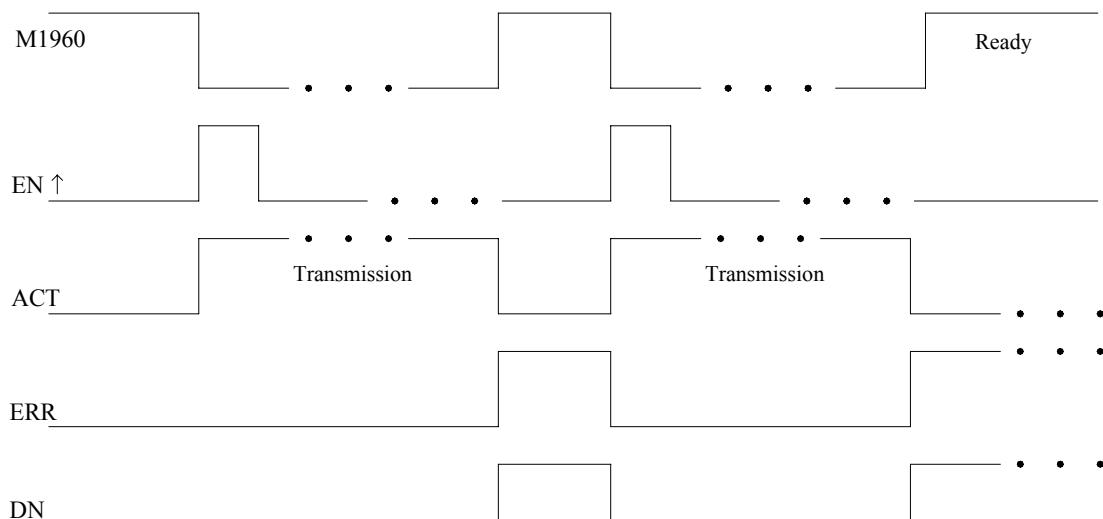
1, the setting of the DIP switch (SW1) of CPU board is error (it must be 1=OFF, 2=ON), shut down setting and restart.

2, data length error (the value is 0, or the packet of transaction is greater than 511)

A, no response from the counter partner.

B, communication abnormal (received error data)

- The waveform for input control and output indication



Note: Of “ERR” and “DN”, only one of them will be in ON status and not both to be ON at the same time.

FUN 97 LINK1	Convenient instruction for FUN97(LINK1): MD2 (Which makes PLC serve as “ASCII receiver” through Port1)	FUN 97 LINK1																									
	<p>MD : 2, PLC waiting to receive the message sent by intelligent peripherals      S : Starting register of data transmission table (see example for explanation)      Pt : Starting register for instruction operation (see example for explanation). It controls 8 registers at least, the other programs cannot repeat in using.</p>																										
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: center;">Range</th> <th style="text-align: center;">HR</th> <th style="text-align: center;">ROR</th> <th style="text-align: center;">DR</th> <th style="text-align: center;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Oper- and</td> <td style="text-align: center;">R0   R3839</td> <td style="text-align: center;">R5000   R8071</td> <td style="text-align: center;">D0   D3071</td> <td></td> </tr> <tr> <td style="text-align: center;">MD</td> <td></td> <td></td> <td></td> <td style="text-align: center;">0~2</td> </tr> <tr> <td style="text-align: center;">S</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> <tr> <td style="text-align: center;">Pt</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input checked="" type="radio"/>*</td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> </tbody> </table>			Range	HR	ROR	DR	K	Oper- and	R0   R3839	R5000   R8071	D0   D3071		MD				0~2	S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		Pt	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>	
Range	HR	ROR	DR	K																							
Oper- and	R0   R3839	R5000   R8071	D0   D3071																								
MD				0~2																							
S	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																								
Pt	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>																								

### Descriptions

1. FUN97 (LINK1): MD2 instructions provides Fatek PLC with ability to receive message sent by peripherals with ASCII interface at any time.
2. The SW1 of CPU board must be set to 1=OFF, 2=ON (shut down setting then restart).
3. The communication protocol is written with LADDER program, which must be consistent to the ASCII peripherals.
4. When execution control “EN ↑” turns from 0→1 and both pause “PAU” and abort “ABT” are 0, and if Port1 is not controlled by other FUN97 instruction (which means M1960=1), this instruction will control Port1 immediately and set M1960 to be “0” (being controlled). If Port1 is being controlled (M1960=0), this instruction will enter into the wait state until the other controlling FUN97 instruction complete or pause/abort its operation and released the control right (M1960=1), and this instruction will enact again out of wait state to enter into the receiving state and set the M1960 to be “0”.
5. When the operation pause “PAU” or abort “ABT” becomes 1, it gives up the receiving immediately (M1960 ON).
6. While it is in the receiving state, the output indication “ACT” is ON.
7. When a packet of data transaction finished (receive finished or receive then transmit completed), if there is error occurred, the output indication “ERR” will be ON for one scan time.
8. When a packet of data transaction finished (receive finished or receive then transmit completed), if there is no error occurred, the output indication “DN” will be ON for one scan time.
9. The connecting pin No.3 (RTS) of Port1 must be short in connection with pin No.4 (CTS).

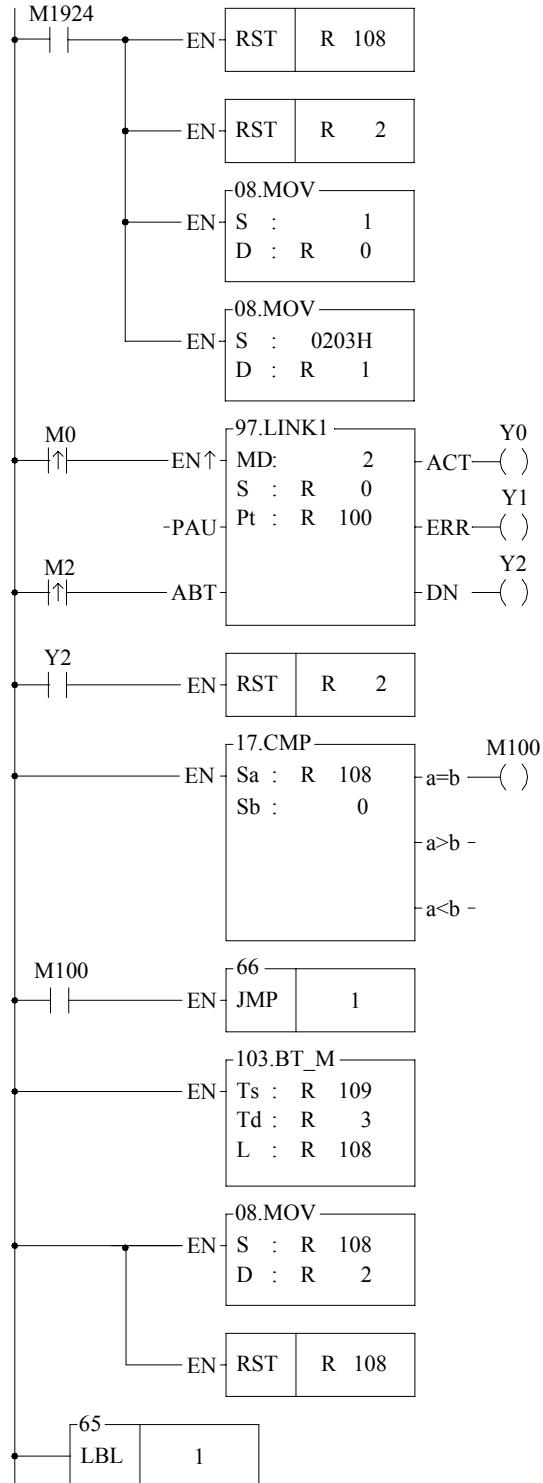
## FUN97: MD2 instruction guide

FUN 97 LINK1	Convenient instruction for FUN97 (LINK1): MD2 (Which makes PLC serve as "ASCII receiver" through Port1)	FUN 97 LINK1
<b>【Interface processing signal】</b>		
M1960 : This signal is generated from CPU		
ON means Port1 is ready.		
OFF means Port1 is busy.		
R4146 : The register for communication parameter setting of Port1		
(Please refer to Port1 communication parameter setting for explanation).		
R4147 : The Low Byte defines the Time-out span of FUN97:MD2 instruction; its unit is 0.1 second (the default		
is 5, which means 0.5 second). When the PLC received the message and must respond to it		
("receive then transmit" mode), but the LADDER program is unable to process and send out the		
responding message during this period of time, the CPU will give up response this time and		
automatically restore back to receiving state. When FUN97: MD2 is set to be "receive only" mode		
(example to be followed), this value is meaningless.		
: High Byte, it is meaningless while FUN97: MD2		
R4148 : When the low byte of R4147 is not 0, the low byte of R4148 is meaningless.		
When R4147 low byte is 0, the low byte of R4148 defines the Time-out span of LINK1 instruction; the		
unit is 0.01 second (for fine tuning; the default is 0). Its function is the same as that described for		
R4147 low byte.		
: High Byte, the setting point for Time-out span on receiving; it is used to judge whether a packet of		
data has been received completely. Its unit is 0.001 second (the default is 0CH, 12mS) (detailed		
explanation will be followed).		
Note 1: Once FUN97: MD2 activated, it will stay in receiving state all the time; unless the input signal of PAU" or		
"ABT" becomes ON, then it will jump out of receiving state and stop receiving and waiting for next time it		
will be activated again.		
2: When there is change on Starting/Ending code for receiving, it must make the input signal of PAU" or		
"ABT" becomes ON once, and re-activate the receive control "EN ↑ " from 0→1 to start message receiving		

## FB-PLC acts as "ASCII receiver" through Port1

## Program example for loop back reply

This PLC station sends back the received data to the master, which had sent out the data.



- Clears the received data length to be 0.
- Clears the transmitted data length to be 0. (for "receive" only, this program is not required).
- Sets up the operation mode:
  - Sets "receive then transmit" mode.
- Sets up the starting code (02H) and ending code (03H) (R1=0, it will receive regularly even without the starting and ending code)
- When transmission complete, clears the transmitted data length to be 0 (for "receive" only mode, this instruction is not needed)
- While selecting "receive then transmit" mode, it employs the comparing instruction to tell whether a new packet of message is received; if it is, the M100=OFF and it will process the received data.
- Copy all of the received data to responding registers.
- R108 is the length of received data.
- After the received data processed, fills the received data length to be the sending back data length to start the responding transmission.
- Clears the received data length to be 0 (ready to receive new data).

## FUN97:MD2 program example

FB-PLC acts as "ASCII receiver" through Port1

- Explanation for FUN97: MD2 parameter S

R0: Starting register for data receiving table (R0 just only for example)

R0	Receive only/Receive then transmit	<ul style="list-style-type: none"> <li>• Low Byte is valid,</li> </ul> <p>0: "receive only" mode.</p> <p>1: "receive then transmit" mode.</p> <ul style="list-style-type: none"> <li>• High Byte : Describing the starting code for receiving</li> <li>• Low Byte : Describing the ending code for receiving.</li> </ul>
R1	Starting/Ending code of receiving	<ul style="list-style-type: none"> <li>• Maximum of length is 511.</li> <li>It will start to transmitter the reply data as long as the length is not 0.</li> </ul>
R2	Length of reply data	
R3	Reply data 1	<ul style="list-style-type: none"> <li>• Low Byte is valid</li> </ul>
R4	Reply data 2	<ul style="list-style-type: none"> <li>• Low Byte is valid</li> </ul>
•		
•		
•		
	Reply data N	<ul style="list-style-type: none"> <li>• Low Byte is valid</li> </ul>

Note 1: When selecting the "receive only" mode, CPU fills the received data into the receiving registers and set the length after it has received a packet of message, and starts to receive the next packet of message immediately.

2: When selecting the "receive then transmit" mode, CPU fills the received data into the receiving registers and set the length after it has received a packet of message; then it starts to wait for the reply data length which is not zero to start transmitting reply data (therefore when select this mode, it must control the reply data length to be zero before the reply data completely filled into the reply registers; when the reply data fills into the reply registers finished, it may then set the length of reply data).

3: It must fills the starting code and ending code into the starting/ending code register before the starting of receiving (e.g. R1=0A0DH, 0AH stands for starting code and 0DH for ending code), so as to ensure it to be free from receiving error.

The communication protocol with starting/ending code makes the identifying of every packet of messages easy, and the communication program is simple and efficient.

4: If the receiving message without starting code, fills the high byte of starting/ending code with 0; if the receiving message without ending code, fills the low byte of starting/ending code with 0. Adjusting High Byte of R4148 (Time-out span) to detect whether a packet of message has been received completely, the unit is 0.001 second (default is 0CH, 12mS). The communication protocol without ending code depends on Time-out span to tell whether it has received completely for a packet of data (the setting point of Time-out must be greater than the maximum delay time between data bytes to be received), thus it may ensure the receiving of the whole packet to be completed. Generally speaking, the data in transmitting is transmitted one byte after another continuously; therefore, if there is pause (greater than Time-out duration), it means that the packet of message is transmitted completely.

5 : When selecting "receive" only mode, if the message received has no ending code, the interval between every packet of data sent by the sending party must be greater than the receiver's receiving Time-out span, otherwise the receiving party won't be able to distinguish between each packet of data correctly.

## FB-PLC acts as "ASCII receiver" through Port1

- Explanation for FUN97: MD2 parameter Pt

	High Byte	Low Byte	
R100	Result code	0	<ul style="list-style-type: none"> <li>The result code stores the operation result, 0=normal; the other values, abnormal</li> </ul>
R101	For internal operation use		<ul style="list-style-type: none"> <li>For internal operation use: it is the registers required by CPU when performing LINK1 instruction.</li> </ul>
R102	For internal operation use		
R103	For internal operation use		
R104	For internal operation use		<ul style="list-style-type: none"> <li>The B0 of R104 is 1 means that Port1 is being occupied, this instruction is waiting to get the control right of Port1.</li> </ul>
R105	For internal operation use		<ul style="list-style-type: none"> <li>B12= "ACT" indication</li> </ul>
R106	For internal operation use		<ul style="list-style-type: none"> <li>B13= "ERR" indication</li> </ul>
R107	For internal operation use		<ul style="list-style-type: none"> <li>B14= "DN" indication</li> </ul>
R108	Length of received data		<ul style="list-style-type: none"> <li>The total amount of data byte that has received (the register for received data length; it includes the starting and ending code that has received).</li> </ul>
R109		1	<ul style="list-style-type: none"> <li>The first byte of data received (if there is the starting code, it is the starting code); High Byte=0</li> </ul>
R110		2	<ul style="list-style-type: none"> <li>The second byte of data received; High Byte =0.</li> </ul>
•			
•			
•			
		N	<ul style="list-style-type: none"> <li>The N_th byte of data received (if there is the ending code, it is the ending code); High Byte=0</li> </ul>

Note : When CPU received a packet of message, it filled the data to receiving registers and set up the received data length. Before the LADDER program starts to receive, you may clear the register of received data length to be 0; it means the receiving of a new packet of message when compared and found that the received data length is not zero. After the LADDER program gets the received data, it clears the received data length register to be 0. Just compare to see the received data length register is not zero means the receiving of a packet of new message, and so it may easily to process the receiving action.

Result code: 0, transaction is successful.

- the setting for DIP switch (SW1) of CPU board is incorrect (must be 1=OFF, 2=ON), shut down setting then restart.
- data length is error (the value is 0, or the transaction is greater than 511)
  - unable to reply message within Time-out span ("receive then transmit" mode).
  - communication abnormal (received error data)

## FUN97:MD2 program example

FB-PLC acts as “ASCII receiver” through Port1

- Explanation of input control

1. When the execution control input M0 change from 0→1, if Port1 is not controlled by other FUN97 (M1960 ON) and it enters into the receiving state immediately (M1960 keeping OFF all the time)
2. When "ABT" input M2 changes from 0→1, it jumps out of receiving state (M1960 ON)

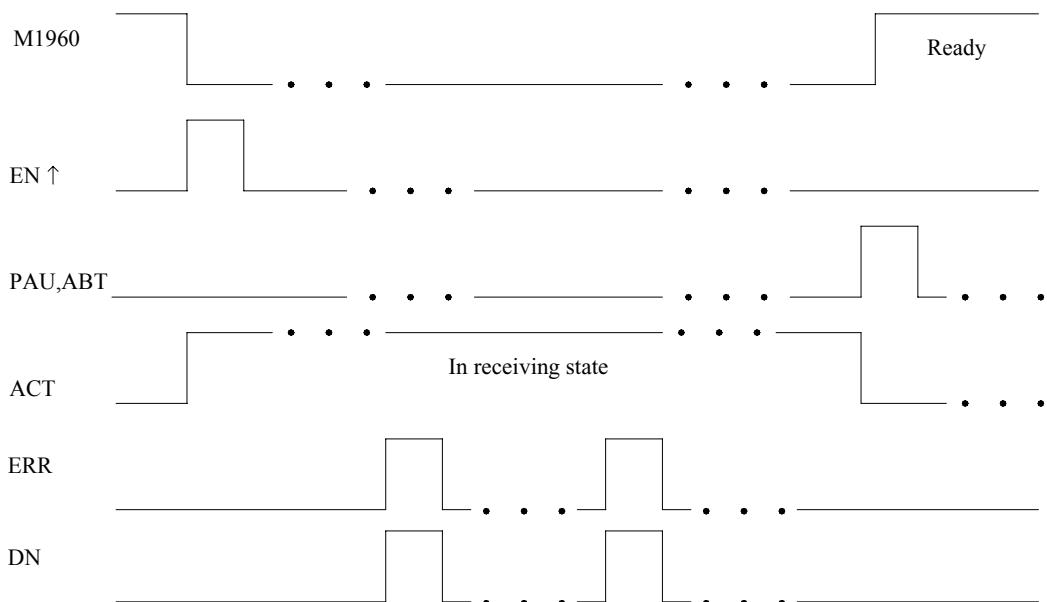
- Output indication

“ACT” ON : In receiving state

“ERR” ON : Error occurred in previous packet of transaction, it will be ON for a scan time

“DN” ON : The previous packet of transaction completed without error, ON for a scan time.

- Waveform of input control and output indication



Note: Of “ERR” and “DN”, there is only one will be ON; not both to be ON at the same time.

## Chapter 14 The NC positioning control of FB-PLC

People use ordinary motor to exercise positioning control in early stage; since the speed and precision demand was not so high then, it was enough to fulfill the demand. As the increasing of mechanical operation speed for the efficiency purpose, finished product quality standard, and precision demands are getting higher, the stopping position control of motor is no more what the ordinary motor is capable to do. The best solution for this problem is to adopt NC positioning controller which incorporate with stepping or servo motor to do the position control. In the past, the extremely high cost limited the prevailing of its usage; however, the technology advance and cost decreasing, which made the pricing affordable, had helped to increase the prevailing of usage gradually. To cope with this trend, the FB-PLC integrated into its internal ASIC chip the special NC positioning controller that is available on the market, therefore makes it free from the bothersome data transaction and linking procedure between PLC and special NC positioning controller. Furthermore, it greatly lowered the entire gadget cost hence provides the user the solution for a good bargain, high quality, simple, and convenient integrated NC positioning control with PLC.

### 14.1 The methods of NC positioning

The methods for controlling interface of PLC and stepping or servo driver are as follows:

- Giving command by way of digital I/O: Easy to use but less dexterity in application.
- Giving command by way of analogue output: Better dexterity in controlling reaction but it is with a higher cost and easy to be interfered by noise.
- Giving command by way of communication: There is no standard for communication protocol and it is confined in communication reaction thus constitutes a bottleneck for application.
- Giving command by way of high speed pulse: The cost is low and is easy to precisely controlled.

Of these methods, controlling stepping or servo driver with high speed pulse is more frequently used method. The main unit of PLC contains multi-axis high speed pulse output and hardware high speed counter, and it can provide easy using, designing for positioning program editing. Therefore it makes the related application even more convenient and comfortable.

Following two kinds are frequently used NC server system that constituted by PLC associates with server drivers:

- **Semi closed loop control**

The PLC is responsible for sending high speed pulse command to server driver. The shift detection signal installed on servo motor will forward directly to server driver, closed loop reaches only to server driver and servo motor. The superior point is that the control is simple and the precision is satisfactory (which is suitable for most of the applications). The defect is that it can't fully reflect the actual shift amount after the transmission element; furthermore, the element being consumed, become aging, or has defect will not be able to be compensated nor checked to verify.

- **Closed loop control**

The PLC is responsible for sending high speed pulse command to server driver. In addition to that the shift detection signal installed on servo motor which will be forwarded directly to server driver, the attached shifting detector installed after the transmission element can fully reflect the actual shift amount and forward it to the high speed counter that PLC contains. So as to make the control becomes more delicate, and help to avoid the defect of above mentioned semi close loop.

### 14.2 Absolute coordinate and relative coordinate

The designation of moving distance can be assigned by absolute location (absolute coordinate positioning), or assigned by relative distance (relative coordinate positioning). And the DRV instruction is used to drive motor.

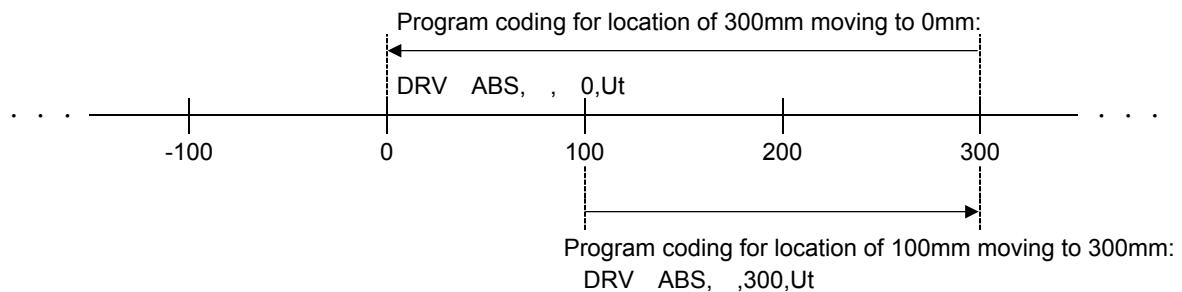
While marking the moving distance with absolute coordinate,

if it is located at 100mm at the present, for moving to 300 mm, the positioning instruction is : DRV ABS, ,300, Ut  
if it is located at 300mm at the present, for moving to 0mm, the positioning instruction is : DRV ABS, , 0, Ut.

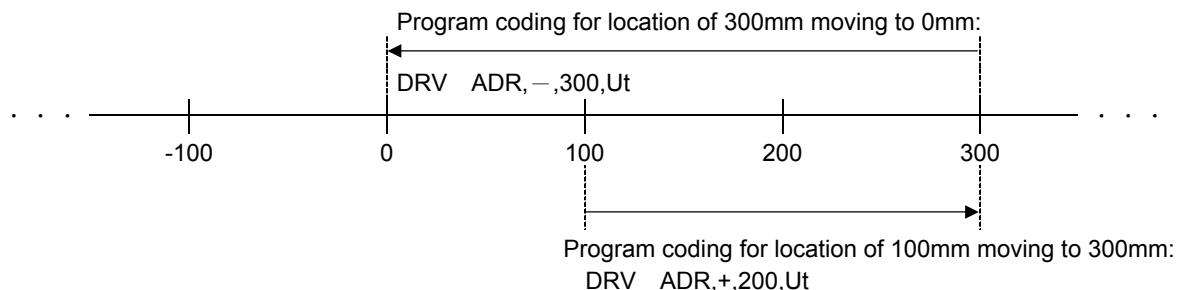
While marking the moving distance with relative coordinate,

if it is located at 100mm at the present, for moving to 300 mm, the positioning instruction is : DRV ADR, +, 200, Ut.  
if it is located at 300mm at the present, for moving to 0mm, the positioning instruction is : DRV ADR, -, 300, Ut.

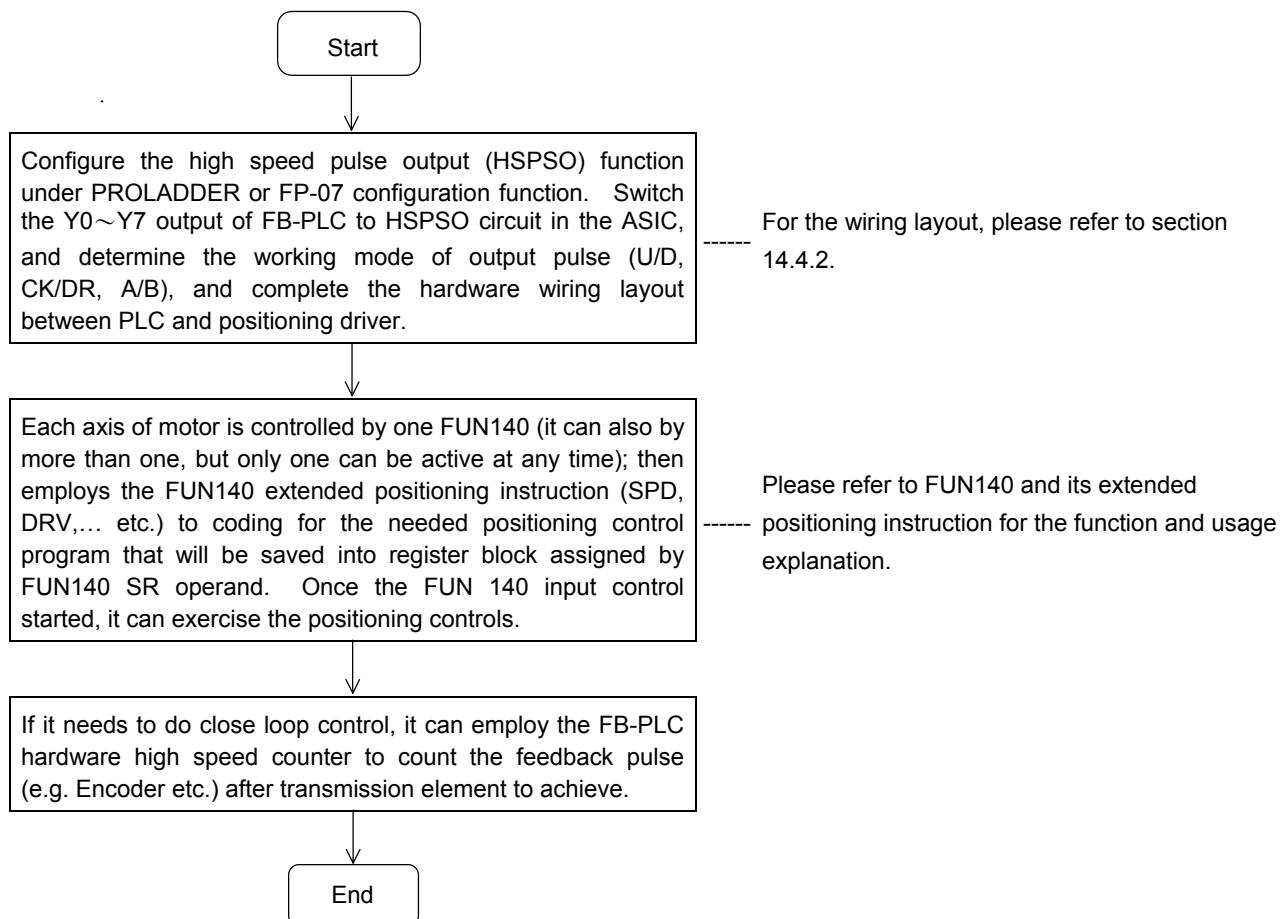
- Absolute coordinate labeling



- Relative coordinate labeling



### 14.3 Procedures of using FB-PLC positioning control



## 14.4 Explanation for the positioning control hardware of FB-PLC

### 14.4.1 Structure of output circuit of HPSO

According to different main unit, it provides 1 axis ( $FB_E\text{-}20MCT/FB_N\text{-}19MCT$ ), 2 axis ( $FB_E\text{-}28MCT/FB_N\text{-}26MCT$ ), and 4 axis ( $FB_E\text{-}40MCT/FB_N\text{-}36MCT$ ) of NC position control respectively. For the frequency of output pulse, it includes 20KHz (single phase) /10KHz (double phase) of single ended transistor output model ( $FB_E\text{-}xxMCT$ ), and high speed differential output model ( $FB_N\text{-}xxMCT$ ) which can reach 512KHz (for both single/double phase), two series of models.

High speed pulse output circuit share to use the  $Y_0 \sim Y_7$  exterior output of FB-PLC. While it is not yet using the HPSO function (haven't configured the PSO function under configuration function), the  $Y_0 \sim Y_7$  exterior output of FB-PLC is corresponding to the  $Y_0 \sim Y_7$  status of internal output relay. When the HPSO has been configured, the  $Y_0 \sim Y_7$  exterior output will switch directly to HPSO output circuit within ASIC, which has no relation with  $Y_0 \sim Y_7$  relay inside PLC.

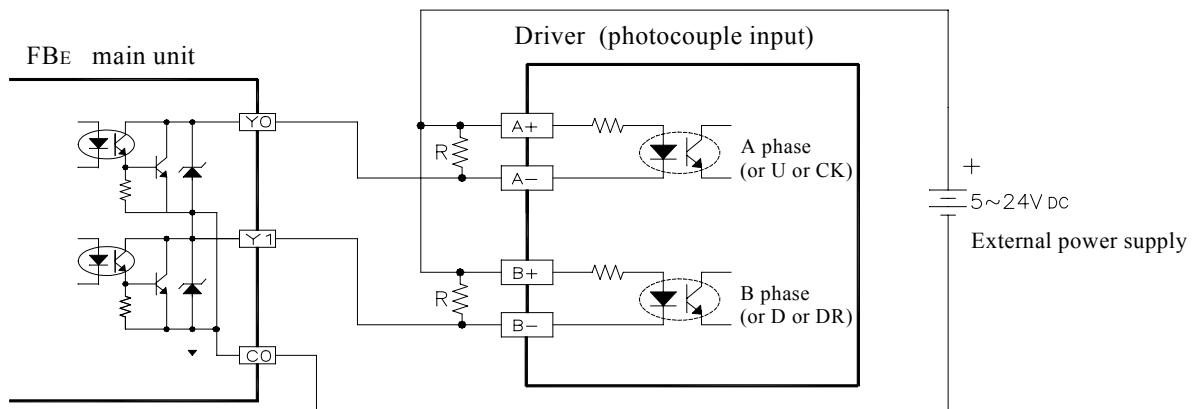
The following is the detailed signals list for respective axis output of main unit and the selectable output modes:

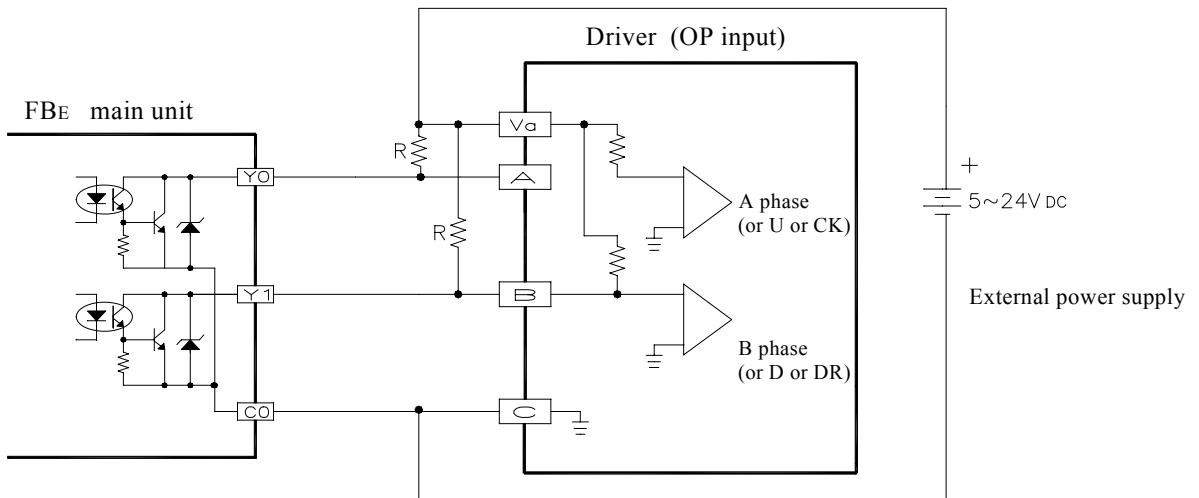
Axis No.	Exterior output	Output modes			Remark
		U/D output	K/R output	A/B output	
PSO0	$Y_0, Y_1$	$Y_0=U, Y_1=D$	$Y_0=K, Y_1=R$	$Y_0=A, Y_1=B$	Valid for all $FB_x\text{-}xxMCT$ main unit
PSO1	$Y_2, Y_3$	$Y_2=U, Y_3=D$	$Y_2=K, Y_3=R$	$Y_2=A, Y_3=B$	Not for $FB_E\text{-}20MCT & FB_N\text{-}19MCT$ .
PSO2	$Y_4, Y_5$	$Y_4=U, Y_5=D$	$Y_4=K, Y_5=R$	$Y_4=A, Y_5=B$	Only for $FB_E\text{-}40MCT & FB_N\text{-}36MCT$ .
PSO3	$Y_6, Y_7$	$Y_6=U, Y_7=D$	$Y_6=K, Y_7=R$	$Y_6=A, Y_7=B$	

### 14.4.2 Hardware wiring layout for FB-PLC positioning control

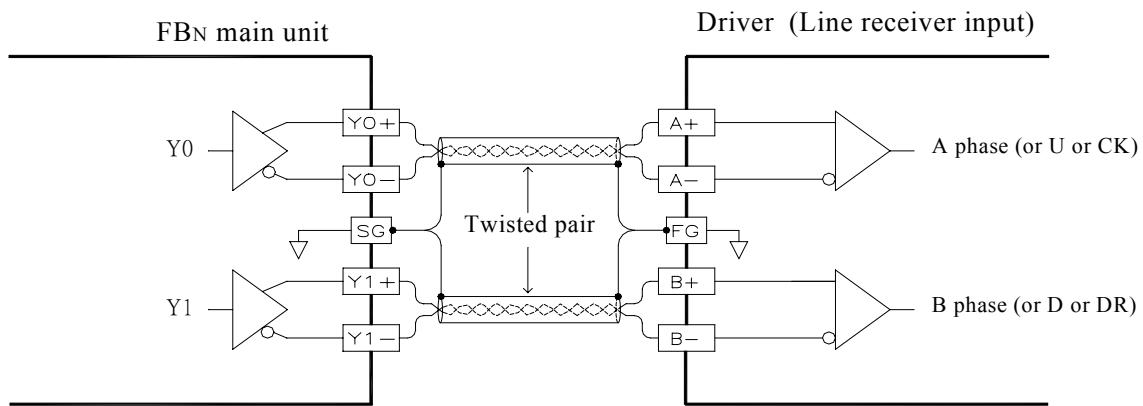
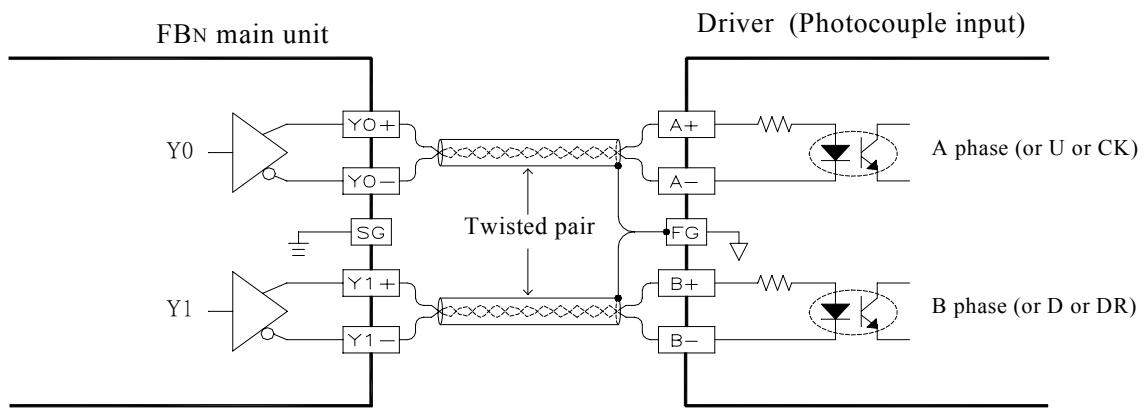
Take the 0\_th axis (PSO0) of  $FB_E$  and  $FB_N$  main unit for example, it is illustrated with diagrams as follows; the others are the same.

#### A, $FB_E$ single ended output wiring layout.





B、FBN differential output wiring layout



( For line receiver input, it must make PLC connect to FG of driver to eliminate common mode voltage )

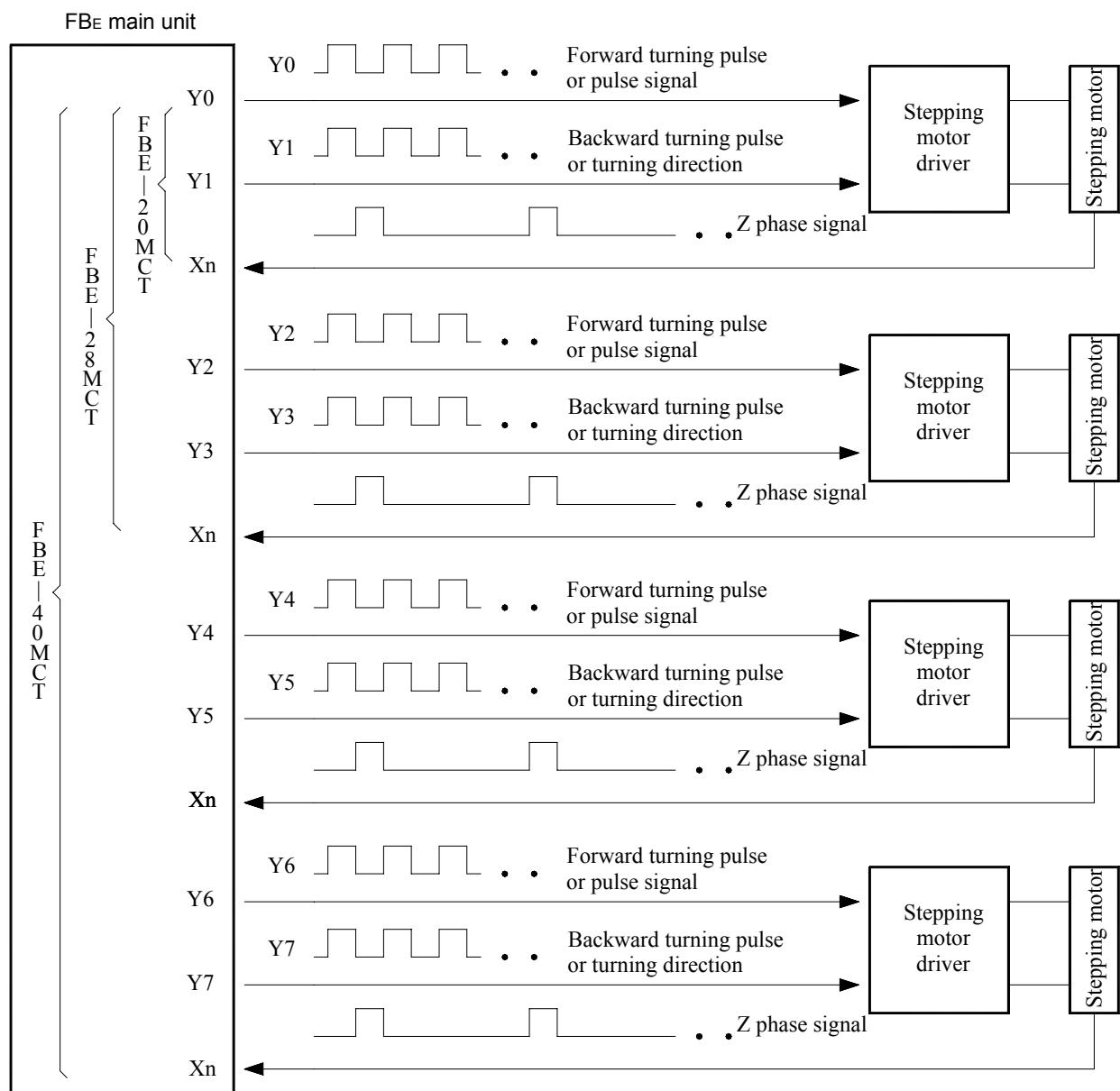
## 14.5 The Explanation for the Position Control Function of FB-PLC

The position control function of FB-PLC incorporates the dedicated NC position controller, which is available in the market, into the PLC. This makes the PLC and NC controller be able to share the same data block without the demand of complicated works like data exchange and synchronized controlling between these two systems. And it can still use the usual NC positioning control instruction (e.g. SPD , DRV,... etc.).

One main unit can control up to 4 axis of their position control, and can drive multi axis simultaneously. However, it provides only point to point positioning and speed control, but it does not provide linear or circular interpolation function. When the system is applying for more than 4 axis, it can also employ CPU LINK function of FB-PLC to attain control over more positioning actions.

The NC position control instruction for  $FB_E$ 、 $FB_N$  main units are identical to each other. The difference is only on the different circuit output, as previously revealed. Hereby we assume that  $FB_E$  main unit is used in the control of stepping motor with lower speed, and  $FB_N$  main unit is used in high speed servo motor control. Consequently, we illustrate only with the connecting diagram of  $FB_E$  main unit that driving stepping motor and the diagram of  $FB_N$  main unit that driving servo motor. Of course we can also use  $FB_E$  main unit to drive servo motor or use  $FB_N$  mainunit to drive stepping motor instead, they can still work perfectly, as long as its circuit structure (single ended or differential) and frequency can match.

#### 14.5.1 Interface of Stepping Motor



- Stepping motor is designed to receive input pulse to attain to the control of desired angle or distance, therefore the turning angle and the input pulse count has a positive correlation ship, and the turning speed also depends on the input pulse frequency.

$$N (\text{RPM}) = 60 \times f / n$$

N : Revolving speed of motor (RPM)

f : Pulse frequency (Ps/Sec)

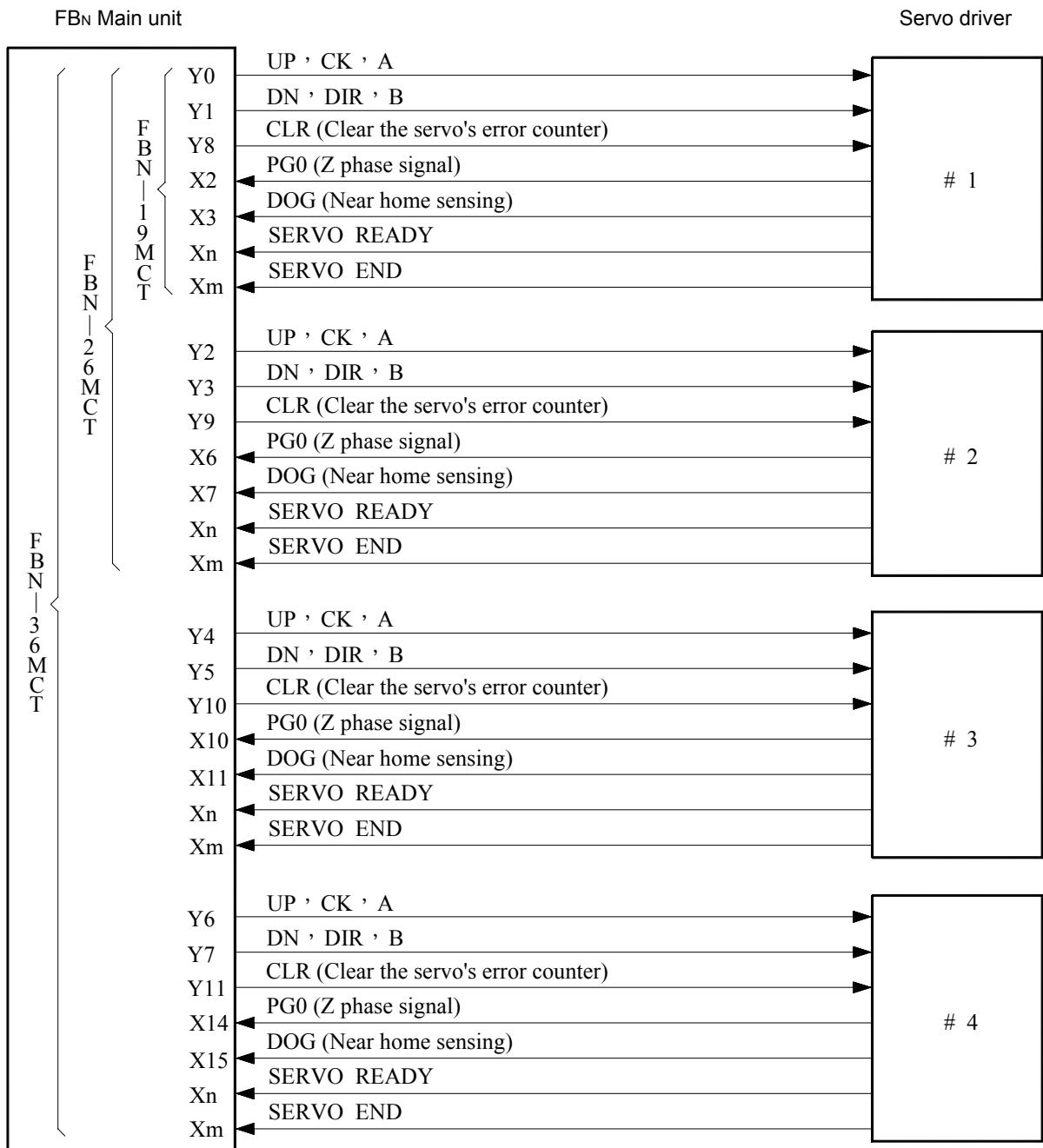
n : Pulse counts for motor to turn for a revolution (Ps/ Rev).

$$n = 360 / \theta_s$$

$\theta_s$  : Angle (Deg)

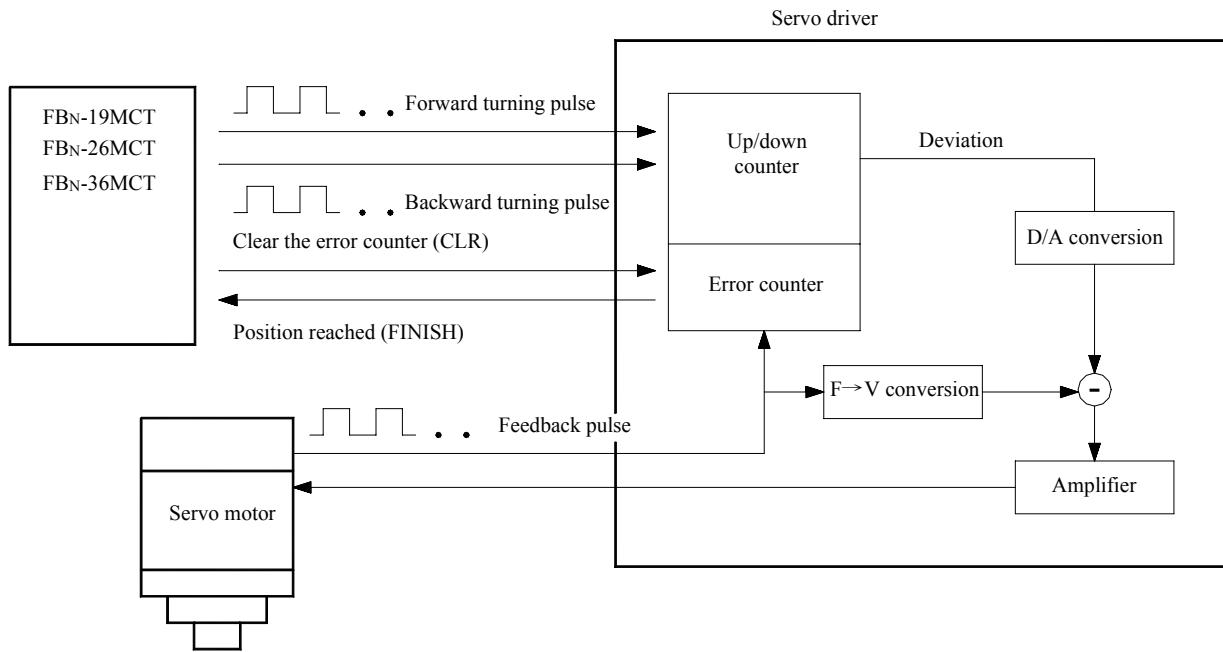
Phase	Basic pulse angle	FULL		HALF	
		Pulse angle	Pulse counts for turning one revolution	Pulse angle	Pulse counts for turning one revolution
5 phase	0.36 °	0.36 °	1000	0.18 °	2000
	0.72 °	0.72 °	500	0.36 °	1000
4 phase	0.90 °	0.90 °	400	0.45 °	800
2 phase	1.80 °	1.80 °	200	0.90 °	400

### 14.5.2 Interface of Servo Motor



- ※ Except that the Y0~Y7 of above diagram are for dedicated purpose, Y8~Y11 and respective inputs can be adjusted for using according to demand.
- ※ The left over travel, right over travel limit switchs for safety detection also need to be connected to PLC to assure proper operation.

### 14.5.3 Working Diagram Illustration for Servo Motor



- The Encoder of servo motor feedback the shifting detection signal to servo driver. The driver gets the pulse frequency, and pulse count of input signal (pulse command), as well as the frequency and pulse count of feedback signal processed with internal error counter and frequency to voltage conversion circuit, and acquired the pulse and turning speed deviations. Using these operations to control the servo motor, so as to obtain a high speed, precise speed and positional closed-loop processing system.
- The revolving speed of servo motor depends on the pulse frequency of input signal; the turning stroke of motor is determined by pulse count.
- Generally speaking, the final control error deviation of servo motor is  $\pm 1$  pulse.

### 14.6 Explanation of Function for NC Position Control Instruction

The NC position control of FB-PLC has following four related instructions:

- FUN140 (HPSO) high speed pulse output instruction, which includes following 8 extension positioning instructions:

1. SPD	5. ACT	Used for positioning program coding and stored to SR operand area of FUN140
2. DRV	6. EXT	
3. DRVC	7. GOTO	
4. WAIT	8. MEND	

- FUN141 (MPARA) positioning parameter setting instruction
- FUN142 (PSOFF) enforcing pulse output stop instruction.
- FUN143 (PSCNV) converting the current pulse value to displaying value instruction.

The following function explanations are for the above mentioned 4 instructions:

FUN 140 HPSO	High Speed Pulse Output (Including the Extended Positioning Instruction)	FUN 140 HPSO																									
Execution control- EN Pause- PAU Abort- ABT	<p>Ps : The set number of Pulse Output (0~3)            0:Y0 &amp; Y1            1:Y2 &amp; Y3            2:Y4 &amp; Y5            3:Y6 &amp; Y7</p> <p>SR: Starting register for positioning program            (example explanation)</p> <p>WR: Starting register for instruction operation (example explanation). It controls 7 registers, which the other program cannot repeat in using.</p>	Ps : The set number of Pulse Output (0~3) 0:Y0 & Y1 1:Y2 & Y3 2:Y4 & Y5 3:Y6 & Y7 SR: Starting register for positioning program (example explanation) WR: Starting register for instruction operation (example explanation). It controls 7 registers, which the other program cannot repeat in using.																									
<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="background-color: #cccccc;">Range</th> <th style="background-color: #cccccc;">HR</th> <th style="background-color: #cccccc;">DR</th> <th style="background-color: #cccccc;">ROR</th> <th style="background-color: #cccccc;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Oper- rand</td> <td style="text-align: center;">R0   R3839</td> <td style="text-align: center;">D0   D3071</td> <td style="text-align: center;">R5000   R8071</td> <td style="text-align: center;"></td> </tr> <tr> <td style="text-align: center;">Ps</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;">0~3</td> </tr> <tr> <td style="text-align: center;">SR</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"></td> </tr> <tr> <td style="text-align: center;">WR</td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input checked="" type="radio"/>*</td> <td style="text-align: center;"></td> </tr> </tbody> </table>			Range	HR	DR	ROR	K	Oper- rand	R0   R3839	D0   D3071	R5000   R8071		Ps	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	0~3	SR	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		WR	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/> *	
Range	HR	DR	ROR	K																							
Oper- rand	R0   R3839	D0   D3071	R5000   R8071																								
Ps	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	0~3																							
SR	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>																								
WR	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/> *																								

#### Instruction Explanation

1. The NC positioning program of FUN140 (HPSO) instruction is a program written and edited with text programming. We named every position point as a step (which includes output frequency, traveling distance, and transfer conditions). For one FUN140, it can be arranged with 250 steps of positioning points at the most, with every step of positioning point controlled by 9 registers.
2. The best benefit to store the positioning program into the registers is that in the case of association with MMI (Man Machine Interface) to operate settings, it may save and reload the positioning program via MMI when replacing the molds.
3. The NC positioning of this instruction without linear interpolation function.
4. When execution control “EN”=1, if the other FUN140 instructions to control Ps0~3 are not active (corresponding status of Ps0=M1992, Ps1=M1993, Ps2=M1994, and Ps3=M1995 will be ON), it will start to execute from the next step of positioning point (when goes to the last step, it will be restarted from the first step to perform); if Ps0~3 is controlled by other FUN140 instruction (corresponding status of Ps0=M1992, Ps1=M1993, Ps2=M1994, and Ps3=M1995 would be OFF), this instruction will acquire the pulse output right of positioning control once the controlling FUN140 has released the control right.
5. When execution control input “EN” =0, it stops the pulse output immediately.
6. When output pause “PAU” =1 and execution control was 1 beforehand, it will pause the pulse output. When output pause “PAU” =0 and execution control is still 1, it will continue the unfinished pulse output.
7. When output abort “ABT”=1, it stops pulse output immediately. (When the execution control input “EN” becomes 1 next time, it will restart from the first step of positioning point to execute.)
8. While the pulse is in output transmitting, the output indication “ACT” is ON.
9. When there is execution error, the output indication “ERR” will be ON.  
 (The error code is stored in the error code register.)
10. When each step of positioning point is complete, the output indication “DN” will be ON.

## NC Positioning Control Instruction

FUN 140 HPSO	High Speed Pulse Output (Including the Extended Positioning Instruction)	FUN 140 HPSO																									
<p>*** The working mode of Pulse Output must be set (without setting, Y0~Y7 will be treated as general output) to be one of U/D, K/R, or A/B mode, thus the Pulse Output may have a regular output.</p> <p>U/D Mode: Y0 (Y2, Y4, Y6), it sends out upward counting pulse. Y1 (Y3, Y5, Y7), it sends out downward counting pulse.</p> <p>K/R Mode: Y0 (Y2, Y4, Y6), it sends the pulse out. Y1 (Y3, Y5, Y7), it sends out the directional signal; ON=upward counting, OFF= downward counting.</p> <p>A/B Mode: Y0 (Y2, Y4, Y6), it sends out the phase A pulse. Y1 (Y3, Y5, Y7), it sends out the phase B pulse.</p> <ul style="list-style-type: none"> <li>• The output polarity for Pulse Output can select to be Normal ON or Normal OFF.</li> <li>• The working mode of Pulse Output can be set in PROLADDER “HSC” (DOS version) setting page.</li> </ul>																											
<p><b>【Interface processing signals】</b></p> <table> <tbody> <tr> <td>M1992: ON, Ps0 Ready OFF, Ps0 is in action</td> <td>M1996: ON, Ps0 has finished the last step.</td> </tr> <tr> <td>M1993: ON, Ps1 Ready OFF, Ps1 is in action</td> <td>M1997: ON, Ps1 has finished the last step.</td> </tr> <tr> <td>M1994: ON, Ps2 Ready OFF, Ps2 is in action</td> <td>M1998: ON, Ps2 has finished the last step.</td> </tr> <tr> <td>M1995: ON, Ps3 Ready OFF, Ps3 is in action</td> <td>M1999: ON, Ps3 has finished the last step.</td> </tr> <tr> <td colspan="2">M2000: ON, multi axes acting simultaneously (At the same scan, when execution control “EN”= 1 of FUN140 instructions which control Ps0~3, their pulses output will be sent at the same time without any time lag). : OFF, as the FUN140 for Ps0~3 starts, corresponding axis pulse output will be sent immediately; since the ladder program is executed in sequence, therefore even the FUN140 for Ps0~3 started at the same scan, there must be some time lag between them.</td></tr> </tbody> </table>			M1992: ON, Ps0 Ready OFF, Ps0 is in action	M1996: ON, Ps0 has finished the last step.	M1993: ON, Ps1 Ready OFF, Ps1 is in action	M1997: ON, Ps1 has finished the last step.	M1994: ON, Ps2 Ready OFF, Ps2 is in action	M1998: ON, Ps2 has finished the last step.	M1995: ON, Ps3 Ready OFF, Ps3 is in action	M1999: ON, Ps3 has finished the last step.	M2000: ON, multi axes acting simultaneously (At the same scan, when execution control “EN”= 1 of FUN140 instructions which control Ps0~3, their pulses output will be sent at the same time without any time lag). : OFF, as the FUN140 for Ps0~3 starts, corresponding axis pulse output will be sent immediately; since the ladder program is executed in sequence, therefore even the FUN140 for Ps0~3 started at the same scan, there must be some time lag between them.																
M1992: ON, Ps0 Ready OFF, Ps0 is in action	M1996: ON, Ps0 has finished the last step.																										
M1993: ON, Ps1 Ready OFF, Ps1 is in action	M1997: ON, Ps1 has finished the last step.																										
M1994: ON, Ps2 Ready OFF, Ps2 is in action	M1998: ON, Ps2 has finished the last step.																										
M1995: ON, Ps3 Ready OFF, Ps3 is in action	M1999: ON, Ps3 has finished the last step.																										
M2000: ON, multi axes acting simultaneously (At the same scan, when execution control “EN”= 1 of FUN140 instructions which control Ps0~3, their pulses output will be sent at the same time without any time lag). : OFF, as the FUN140 for Ps0~3 starts, corresponding axis pulse output will be sent immediately; since the ladder program is executed in sequence, therefore even the FUN140 for Ps0~3 started at the same scan, there must be some time lag between them.																											
<table border="1"> <thead> <tr> <th style="text-align: center;">Ps No.</th> <th style="text-align: center;">Current output frequency</th> <th style="text-align: center;">Current PS position</th> <th style="text-align: center;">The remaining PS counts to be transmitted</th> <th style="text-align: center;">Error code</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Ps0</td> <td style="text-align: center;">DR4080</td> <td style="text-align: center;">DR4088</td> <td style="text-align: center;">DR4072</td> <td style="text-align: center;">R4060</td> </tr> <tr> <td style="text-align: center;">Ps1</td> <td style="text-align: center;">DR4082</td> <td style="text-align: center;">DR4090</td> <td style="text-align: center;">DR4074</td> <td style="text-align: center;">R4061</td> </tr> <tr> <td style="text-align: center;">Ps2</td> <td style="text-align: center;">DR4084</td> <td style="text-align: center;">DR4092</td> <td style="text-align: center;">DR4076</td> <td style="text-align: center;">R4062</td> </tr> <tr> <td style="text-align: center;">Ps3</td> <td style="text-align: center;">DR4086</td> <td style="text-align: center;">DR4094</td> <td style="text-align: center;">DR4078</td> <td style="text-align: center;">R4063</td> </tr> </tbody> </table> <p>※ R4056: When the value of low byte=5AH, it can be dynamically changed for its output frequency during the high speed pulse output transmitting at any time. When the value of low byte is not 5AH, it can not be dynamically changed for its output frequency during the high speed pulse output transmitting. When the value of high byte is 1, auto slow down will be initiated when dynamic changing frequency. When the value of high byte is not 1, auto slow down will be negated when dynamic changing frequency. The default value of R4056 is 0</p> <p>R4064: The step number (positioning point) which has been completed of Ps0. R4065: The step number (positioning point) which has been completed of Ps1. R4066: The step number (positioning point) which has been completed of Ps2. R4067: The step number (positioning point) which has been completed of Ps3.</p>			Ps No.	Current output frequency	Current PS position	The remaining PS counts to be transmitted	Error code	Ps0	DR4080	DR4088	DR4072	R4060	Ps1	DR4082	DR4090	DR4074	R4061	Ps2	DR4084	DR4092	DR4076	R4062	Ps3	DR4086	DR4094	DR4078	R4063
Ps No.	Current output frequency	Current PS position	The remaining PS counts to be transmitted	Error code																							
Ps0	DR4080	DR4088	DR4072	R4060																							
Ps1	DR4082	DR4090	DR4074	R4061																							
Ps2	DR4084	DR4092	DR4076	R4062																							
Ps3	DR4086	DR4094	DR4078	R4063																							

FUN 140 HPSO	High Speed Pulse Output (Including the Extended Positioning Instruction)	FUN 140 HPSO
<ul style="list-style-type: none"> <li>Format of positioning program:</li> </ul> <p>SR: Starting register of registers block which reserved to store positioning program, explained as follows:</p> <pre> graph TD     SR[A55AH] --- SR_S[SR]     SR_S --- TotalSteps[Total steps: 1~250]     SR_S --- Step1[Step 1: SR+2 to SR+10]     SR_S --- StepN[Step N: SR+Nx9+2]     </pre> <p>The effective positioning program; its starting register must be A55AH</p> <p>1~250</p> <p>The first positioning point (step) of positioning program (every step controlled by 9 registers).</p> <p>The <math>N_{th}</math> step of positioning program.</p>		

## NC Positioning Control Instruction

FUN 140 HPSO	High Speed Pulse Output (Including the Extended Positioning Instruction)	FUN 140 HPSO														
● Explanation for working register of instruction operation:																
WR is the starting register.																
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="padding: 2px;">WR+0</td><td style="padding: 2px;">Being executed or stopped step</td></tr> <tr><td style="padding: 2px;">WR+1</td><td style="padding: 2px;">Working flag</td></tr> <tr><td style="padding: 2px;">WR+2</td><td style="padding: 2px;">Controlled by system</td></tr> <tr><td style="padding: 2px;">WR+3</td><td style="padding: 2px;">Controlled by system</td></tr> <tr><td style="padding: 2px;">WR+4</td><td style="padding: 2px;">Controlled by system</td></tr> <tr><td style="padding: 2px;">WR+5</td><td style="padding: 2px;">Controlled by system</td></tr> <tr><td style="padding: 2px;">WR+6</td><td style="padding: 2px;">Controlled by system</td></tr> </table>			WR+0	Being executed or stopped step	WR+1	Working flag	WR+2	Controlled by system	WR+3	Controlled by system	WR+4	Controlled by system	WR+5	Controlled by system	WR+6	Controlled by system
WR+0	Being executed or stopped step															
WR+1	Working flag															
WR+2	Controlled by system															
WR+3	Controlled by system															
WR+4	Controlled by system															
WR+5	Controlled by system															
WR+6	Controlled by system															
<p>WR+0: If this instruction is in execution, the content of this register represents the step (1~N) being performed.      if this instruction is not in execution, the content of this register represents the step where it stopped at present      When execution control “EN” =1, it will perform the next step, i.e. the current step plus 1 (if the current step is at the last step, it will restart to perform from the first step).      Before starting the execution control “EN” =1, the user can renew the content of WR+0 to determine starting from which step to perform (when the content of WR+0 =0, and execution control “EN” =1, it represents that the execution starts from the first step).</p>																
<p>WR+1:B0~B7, total steps</p> <ul style="list-style-type: none"> <li>B8 =ON, output paused</li> <li>B9 =ON, waiting for transfer condition</li> <li>B10=ON, endless output (the stroke operand of DRV command is set to be 0 Ut)</li> <li>B12=ON, pulse output transmitting (the status of output indicator “ACT”)</li> <li>B13=ON, instruction execution error (the status of output indicator “ERR”)</li> <li>B14=ON, finished being executed step (the status of output indicator “DN”)</li> </ul> <p>*** Once the FUN140 instruction has been started (the B12 of WR+1=ON) and if suspended by a shut down for emergency or switchover from auto to manual mode while the pulse output has not yet completed, this instruction will be negated at next execution. It must clear the WR+1 register to be 0 before restarts the instruction next time, so as to make the instruction be initiated again; otherwise, the pulse output will not appear!</p> <p>*** Whether execution control “EN” =0 or 1, executing the FUN140 instruction every scan , there won’t have the situation mentioned above.</p> <p>*** When step which has been completed, the output indication “DN” will turn ON and keep such status if suspending ; the user may turn OFF the status of “DN” by using the rising edge of output coil controlled by “DN” to clear the content of WR+1 register to be 0, and it can be attained.</p>																

FUN 140 HPSO	High Speed Pulse Output (Including the Extended Positioning Instruction)	FUN 140 HPSO
<p>Error indication                      Error code</p> <p>R4060 (Ps0)    0 : Error free</p> <p>R4061 (Ps1)    1 : Parameter 0 error</p> <p>R4062 (Ps2)    2 : Parameter 1 error</p> <p>R4063 (Ps3)    3 : Parameter 2 error</p> <p>                        4 : Parameter 3 error</p> <p>                        5 : Parameter 4 error</p> <p>                        7 : Parameter 6 error</p> <p>                        8 : Parameter 7 error</p> <p>                        9 : Parameter 8 error</p> <p>                        10 : Parameter 9 error</p> <p>                        30 : Error of variable address for speed setting</p> <p>                        31 : Error of setting value for speed setting</p> <p>                        32 : Error of variable address for stroke setting</p> <p>                        33 : Error of setting value for stroke setting</p> <p>                        34 : Illegal positioning program</p> <p>                        35 : Length error of total step</p> <p>                        36 : Over the maximum step</p> <p>                        37 : Limited frequency error</p> <p>                        38 : Initiate/stop frequency error</p> <p>                        39 : Over range of compensation value for movement</p> <p>                        40 : Over range of moving stroke</p> <p>                        41 : ABS positioning is not allowed within DRVC commands</p>	 <p>The possible error codes for FUN141 execution</p> <p>The possible error codes for FUN140 execution</p>	

Note: The content of error indication register will keep the latest error code. Making sure that no more error to happen, you can clear the content of error indication register to be 0; as long as the content maintains at 0, it represents that there's no error happened.

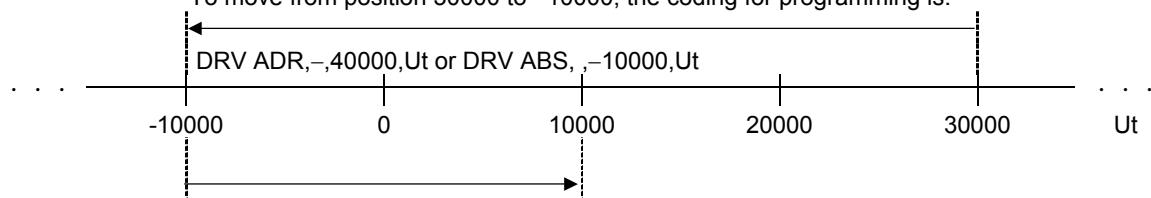
## NC Positioning Control Instruction

FUN 140 HPSO	High Speed Pulse Output (Including the Extended Positioning Instruction)	FUN 140 HPSO									
<ul style="list-style-type: none"> <li>To make it easy to edit, read, and maintain the positioning program, we have extended following related instructions under FUN140 instruction. The user may edit, and modify the positioning program directly in PROLADDER (if you are editing the program with PROLADDER in DOS version, key in the complete FUN140 instruction and then move cursor to location of FUN140 instruction and press “ALT” “Z” at the same time and it will display and allow to edit the positioning program. While editing the positioning program, simultaneously pressed “Shift” “INS” means to insert a positioning point at the cursor location; simultaneously pressed “Shift” “DEL” means to delete the positioning point at the cursor location; simultaneously pressed “ALT” “INS” or “Shift” “+” means to add a positioning point to the bottom).</li> <li>Extended positioning instructions are listed as follows:</li> </ul>											
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #cccccc;"> <th style="text-align: left; padding: 2px;">Instruction</th> <th style="text-align: left; padding: 2px;">Operand</th> <th style="text-align: left; padding: 2px;">Explanation</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">SPD</td><td style="padding: 2px;">XXXXXX or Rxxxx or Dxxxx</td><td style="padding: 2px;"> <ul style="list-style-type: none"> <li>Moving speed in frequency or velocity (FUN141 Parameter_0=0 represents velocity; Parameter_0=1 or 2 for frequency; the system default is frequency). The operand can be input directly with constant or variable (Rxxxx, Dxxxx); when the operand is variable, it needs 2 registers, e.g. D10 represents D10 (Low Word) and D11 (High Word), which is the setting of frequency or velocity.</li> <li>When selecting to use the velocity setting, the system will automatically convert the velocity setting to corresponding output frequency.</li> <li>Output frequency range: <math>10 \leq \text{output frequency} \leq 512000 \text{ Hz}</math>.</li> </ul> <p>*** When the output frequency is 0, this instruction will wait until the setting value isn't 0 to execute the positioning pulse output.</p> <p>*** When the setting value of frequency is smaller than 10, pulse output goes with 10Hz.</p> </td></tr> <tr> <td style="padding: 2px;">DRV</td><td style="padding: 2px;"> ADR , + , XXXXXXXX , Ut  ADR , + , XXXXXXXX , Ps  ADR , - , XXXXXXXX , Ut  ADR , - , XXXXXXXX , Ps  ADR , , XXXXXXXX , Ut  ADR , , -XXXXXXX , Ut  ADR , , XXXXXXXX , Ps  ADR , , -XXXXXXX , Ps  ADR , + , Rxxxx , Ut  ADR , + , Rxxxx , Ps  ADR , - , Rxxxx , Ut  ADR , - , Rxxxx , Ps  ADR , , Rxxxx , Ut  ADR , , Rxxxx , Ps  ADR , + , Dxxxx , Ut  ADR , + , Dxxxx , Ps  ADR , - , Dxxxx , Ut  ADR , - , Dxxxx , Ps  ADR , , Dxxxx , Ut  ADR , , Dxxxx , Ps  ABS , , XXXXXXXX , Ut  ABS , , -XXXXXXX , Ut  ABS , , XXXXXXXX , Ps  ABS , , -XXXXXXX , Ps  ABS , , Rxxxx , Ut  ABS , , Rxxxx , Ps  ABS , , Dxxxx , Ut  ABS , , Dxxxx , Ps </td><td style="padding: 2px;"> <ul style="list-style-type: none"> <li>Moving stroke setting in Ps or mm,Deg,Inch (When FUN141 Parameter_0=1, the setting stroke in Ut is Ps; Parameter_0=0 or 2, the setting stroke in Ut is mm, Deg, Inch; the system default for Ut is Ps).</li> <li>When 4_th operand of DRV is Ut (not Ps) , according to parameter setting of 1, 2, 3 of FUN141, the system will convert the corresponding pulse count to output.</li> <li>There are 4 operands to construct DRV instruction as follows:</li> </ul> <p>1_st operand: coordinate selection.          ADR or ABS: ADR, relative distance movement          ABS, absolute position movement</p> <p>2_nd operand:          revolving direction selection (Valid for ADR only).          '+' , forward or clockwise          '-' , backward or counterclockwise          ' ' , direction is determined by the setting value          (positive value: forward; negative value: backward)</p> <p>3_rd operand: moving stroke setting          XXXXXXXX: It can directly input with constant or variable (Rxxxx, Dxxxx); it needs 2 registers when adopting the variable, e.g. R0 represents R0 (Low Word) and R1 (High Word) as the setting of moving stroke.</p> <p>*** When the setting of moving stroke is 0 and 4_th operand is Ut, it represents to revolve endless, and current PS position will not be updated.</p> <p>Stroke setting range: <math>-99999999 \leq \text{stroke setting} \leq 99999999</math></p> <p>4_th operand: resolution of stroke setting          Ut or Ps:for Ut, the resolution is one unit;          (it is determined by parameter 0, 3 of FUN141); for Ps, the enforced resolution is one Ps.</p> </td></tr> </tbody> </table>			Instruction	Operand	Explanation	SPD	XXXXXX or Rxxxx or Dxxxx	<ul style="list-style-type: none"> <li>Moving speed in frequency or velocity (FUN141 Parameter_0=0 represents velocity; Parameter_0=1 or 2 for frequency; the system default is frequency). The operand can be input directly with constant or variable (Rxxxx, Dxxxx); when the operand is variable, it needs 2 registers, e.g. D10 represents D10 (Low Word) and D11 (High Word), which is the setting of frequency or velocity.</li> <li>When selecting to use the velocity setting, the system will automatically convert the velocity setting to corresponding output frequency.</li> <li>Output frequency range: <math>10 \leq \text{output frequency} \leq 512000 \text{ Hz}</math>.</li> </ul> <p>*** When the output frequency is 0, this instruction will wait until the setting value isn't 0 to execute the positioning pulse output.</p> <p>*** When the setting value of frequency is smaller than 10, pulse output goes with 10Hz.</p>	DRV	ADR , + , XXXXXXXX , Ut ADR , + , XXXXXXXX , Ps ADR , - , XXXXXXXX , Ut ADR , - , XXXXXXXX , Ps ADR , , XXXXXXXX , Ut ADR , , -XXXXXXX , Ut ADR , , XXXXXXXX , Ps ADR , , -XXXXXXX , Ps ADR , + , Rxxxx , Ut ADR , + , Rxxxx , Ps ADR , - , Rxxxx , Ut ADR , - , Rxxxx , Ps ADR , , Rxxxx , Ut ADR , , Rxxxx , Ps ADR , + , Dxxxx , Ut ADR , + , Dxxxx , Ps ADR , - , Dxxxx , Ut ADR , - , Dxxxx , Ps ADR , , Dxxxx , Ut ADR , , Dxxxx , Ps ABS , , XXXXXXXX , Ut ABS , , -XXXXXXX , Ut ABS , , XXXXXXXX , Ps ABS , , -XXXXXXX , Ps ABS , , Rxxxx , Ut ABS , , Rxxxx , Ps ABS , , Dxxxx , Ut ABS , , Dxxxx , Ps	<ul style="list-style-type: none"> <li>Moving stroke setting in Ps or mm,Deg,Inch (When FUN141 Parameter_0=1, the setting stroke in Ut is Ps; Parameter_0=0 or 2, the setting stroke in Ut is mm, Deg, Inch; the system default for Ut is Ps).</li> <li>When 4_th operand of DRV is Ut (not Ps) , according to parameter setting of 1, 2, 3 of FUN141, the system will convert the corresponding pulse count to output.</li> <li>There are 4 operands to construct DRV instruction as follows:</li> </ul> <p>1_st operand: coordinate selection.          ADR or ABS: ADR, relative distance movement          ABS, absolute position movement</p> <p>2_nd operand:          revolving direction selection (Valid for ADR only).          '+' , forward or clockwise          '-' , backward or counterclockwise          ' ' , direction is determined by the setting value          (positive value: forward; negative value: backward)</p> <p>3_rd operand: moving stroke setting          XXXXXXXX: It can directly input with constant or variable (Rxxxx, Dxxxx); it needs 2 registers when adopting the variable, e.g. R0 represents R0 (Low Word) and R1 (High Word) as the setting of moving stroke.</p> <p>*** When the setting of moving stroke is 0 and 4_th operand is Ut, it represents to revolve endless, and current PS position will not be updated.</p> <p>Stroke setting range: <math>-99999999 \leq \text{stroke setting} \leq 99999999</math></p> <p>4_th operand: resolution of stroke setting          Ut or Ps:for Ut, the resolution is one unit;          (it is determined by parameter 0, 3 of FUN141); for Ps, the enforced resolution is one Ps.</p>
Instruction	Operand	Explanation									
SPD	XXXXXX or Rxxxx or Dxxxx	<ul style="list-style-type: none"> <li>Moving speed in frequency or velocity (FUN141 Parameter_0=0 represents velocity; Parameter_0=1 or 2 for frequency; the system default is frequency). The operand can be input directly with constant or variable (Rxxxx, Dxxxx); when the operand is variable, it needs 2 registers, e.g. D10 represents D10 (Low Word) and D11 (High Word), which is the setting of frequency or velocity.</li> <li>When selecting to use the velocity setting, the system will automatically convert the velocity setting to corresponding output frequency.</li> <li>Output frequency range: <math>10 \leq \text{output frequency} \leq 512000 \text{ Hz}</math>.</li> </ul> <p>*** When the output frequency is 0, this instruction will wait until the setting value isn't 0 to execute the positioning pulse output.</p> <p>*** When the setting value of frequency is smaller than 10, pulse output goes with 10Hz.</p>									
DRV	ADR , + , XXXXXXXX , Ut ADR , + , XXXXXXXX , Ps ADR , - , XXXXXXXX , Ut ADR , - , XXXXXXXX , Ps ADR , , XXXXXXXX , Ut ADR , , -XXXXXXX , Ut ADR , , XXXXXXXX , Ps ADR , , -XXXXXXX , Ps ADR , + , Rxxxx , Ut ADR , + , Rxxxx , Ps ADR , - , Rxxxx , Ut ADR , - , Rxxxx , Ps ADR , , Rxxxx , Ut ADR , , Rxxxx , Ps ADR , + , Dxxxx , Ut ADR , + , Dxxxx , Ps ADR , - , Dxxxx , Ut ADR , - , Dxxxx , Ps ADR , , Dxxxx , Ut ADR , , Dxxxx , Ps ABS , , XXXXXXXX , Ut ABS , , -XXXXXXX , Ut ABS , , XXXXXXXX , Ps ABS , , -XXXXXXX , Ps ABS , , Rxxxx , Ut ABS , , Rxxxx , Ps ABS , , Dxxxx , Ut ABS , , Dxxxx , Ps	<ul style="list-style-type: none"> <li>Moving stroke setting in Ps or mm,Deg,Inch (When FUN141 Parameter_0=1, the setting stroke in Ut is Ps; Parameter_0=0 or 2, the setting stroke in Ut is mm, Deg, Inch; the system default for Ut is Ps).</li> <li>When 4_th operand of DRV is Ut (not Ps) , according to parameter setting of 1, 2, 3 of FUN141, the system will convert the corresponding pulse count to output.</li> <li>There are 4 operands to construct DRV instruction as follows:</li> </ul> <p>1_st operand: coordinate selection.          ADR or ABS: ADR, relative distance movement          ABS, absolute position movement</p> <p>2_nd operand:          revolving direction selection (Valid for ADR only).          '+' , forward or clockwise          '-' , backward or counterclockwise          ' ' , direction is determined by the setting value          (positive value: forward; negative value: backward)</p> <p>3_rd operand: moving stroke setting          XXXXXXXX: It can directly input with constant or variable (Rxxxx, Dxxxx); it needs 2 registers when adopting the variable, e.g. R0 represents R0 (Low Word) and R1 (High Word) as the setting of moving stroke.</p> <p>*** When the setting of moving stroke is 0 and 4_th operand is Ut, it represents to revolve endless, and current PS position will not be updated.</p> <p>Stroke setting range: <math>-99999999 \leq \text{stroke setting} \leq 99999999</math></p> <p>4_th operand: resolution of stroke setting          Ut or Ps:for Ut, the resolution is one unit;          (it is determined by parameter 0, 3 of FUN141); for Ps, the enforced resolution is one Ps.</p>									

FUN 140 HPSO	High Speed Pulse Output (Including the Extended Positioning Instruction)	FUN 140 HPSO
Instruction	Operand	Explanation
DRVC	ADR , + , XXXXXXXX , Ut or or or or ABS , - , Rxxxxx , Ps or Dxxxxx	<p>The usage of DRVC and the operand explanation is the same as DRV's instruction.</p> <p>*** DRVC is used to do successive speed changing control (8 speeds at the most).</p> <p>*** Of the successive speed changing control, only the first DRVC instruction can use the absolute value coordinate for positioning.</p> <p>*** The revolution direction of DRVC can only be decided by '+' or '-'.</p> <p>*** The revolution direction only determined by the first DRVC of successive DRVC instructions; i.e. the successive speed changing control can only be the same direction.</p> <p>*** The output frequency of DRVC must be <math>\geq 141\text{Hz}</math>.</p> <p>For example: successive 3 speed changing control</p> <pre> 001 SPD 10000          * Pulse frequency = 10KHz.       DRVC ADR , + , 20000 , Ut * Forward 20000 units.       GOTO NEXT 002 SPD 50000          * Pulse frequency = 50 KHz       DRVC ADR , + , 60000 , Ut * Forward 60000 units.       GOTO NEXT 003 SPD 3000           * Pulse frequency = 3KHz.       DRV  ADR , + , 5000 , Ut  * Forward 5000 units.       WAIT X0                 * Wait until X0 ON to restart from       GOTO 1                  the first step to execute.     </pre> <p>Note: The number of DRVC instructions must be the number of successive speeds deducted by 1, i.e. the successive speed changing control must be ended with the DRV instruction.</p> <ul style="list-style-type: none"> <li>The above mentioned example is for successive 3 speeds changing control, which used 2 DRVC instructions and the third must use DRV instruction.</li> <li>Diagram illustration for the above mentioned example:</li> </ul>

Note: Comparison explanation between the relative coordinate positioning (ADR) and the absolute coordinate positioning (ABS)

To move from position 30000 to -10000, the coding for programming is:



To move from position -10000 to 10000, the coding for programming is:  
DRV ADR,+20000, Ut or DRV ABS, ,10000, Ut

## NC Positioning Control Instruction

FUN 140 HPSO	High Speed Pulse Output (Including the Extended Positioning Instruction)	FUN 140 HPSO
-----------------	---	-----------------

Instruction	Operand	Explanation
WAIT	Time, XXXXX or Rxxxx or Dxxxx  or X0~X255 or Y0~Y255 or M0~M1911 or S0~S999	<ul style="list-style-type: none"> <li>When pulse output is complete, performing the wait instruction for going to the next step. There are 5 kind of operands that explained as follows:</li> </ul> <p>Time: The waiting time (the unit is 0.01 second), it can be directly input with constant or variable (Rxxxx or Dxxxx); when it is time up, performs the step that assigned by GOTO.</p> <p>X0~X255: Waiting until the input status is ON, it performs the step that assigned by GOTO.</p> <p>Y0~Y255: Waiting until the output status is ON, it performs the step that assigned by GOTO.</p> <p>M0~M1911: Waiting until the internal relay is ON, it performs the step that assigned by GOTO.</p> <p>S0~S999: Waiting until the step relay is ON, it performs the step that assigned by GOTO.</p>
ACT	Time · XXXXX or Rxxxx or Dxxxx	<ul style="list-style-type: none"> <li>After the time to output pulses described by operand of ACT, it performs immediately the step that assigned by GOTO, i.e. after the pulse output for a certain time, it performs the next step immediately. The action time (the unit is 0.01 second) can be directly input with constant or variable (Rxxxx or Dxxxx); when the action time is up, it performs the step assigned by GOTO.</li> </ul>
EXT	X0~X255 or Y0~Y255 or M0~M1911 or S0~S999	<ul style="list-style-type: none"> <li>External trigger instruction; when it is in pulse output (the number of pulses sending is not complete yet), if the status of external trigger is ON, it will perform the step assigned by GOTO immediately. If the status of external trigger is still OFF when the pulse output has been complete, it is the same as WAIT instruction; waiting the trigger signal ON, then perform the step assigned by GOTO.</li> </ul>
GOTO	NEXT or 1~N or Rxxxx or Dxxxx	<ul style="list-style-type: none"> <li>When matching the transfer condition of WAIT, ACT, EXT instruction, by using GOTO instruction to describe the step to be executed.</li> </ul> <p>NEXT: It represents to perform the next step.</p> <p>1~N: To perform the described number of step.</p> <p>Rxxxx: The step to be performed is stored in register Rxxxx.</p> <p>Dxxxx: The step to be performed is stored in register Dxxxx.</p>
MEND		The end of the positioning program.

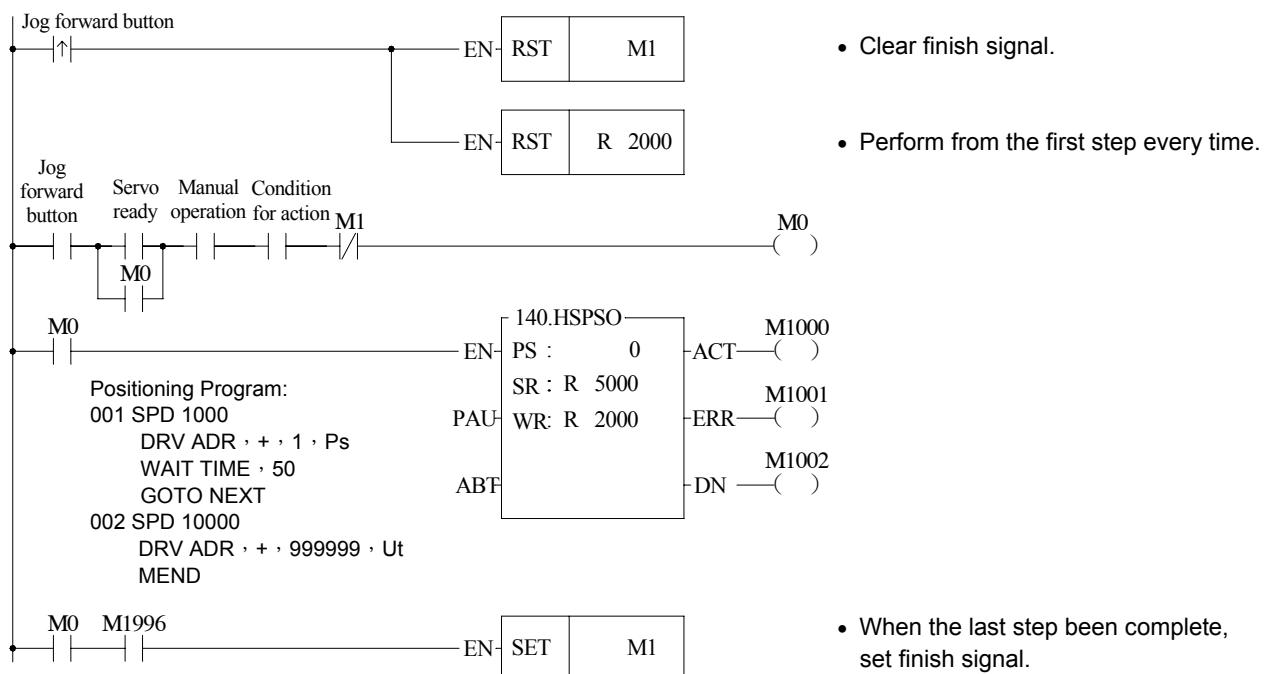
FUN 140 HPSO	High Speed Pulse Output (Including the Extended Positioning Instruction)	FUN 140 HPSO
<ul style="list-style-type: none"> <li>The coding for positioning programming:</li> </ul> <p>First, it must complete the FUN140 instruction before the editing of positioning program, and assigned in FUN140 instruction the starting register of registers block to store positioning program. While editing the positioning program, it will store the newly edited positioning program to the assigned registers block; for every one positioning point (called as one step) edited, it is controlled by 9 registers. If there are N positioning points, it will be controlled by <math>N \times 9 + 2</math> registers in total.</p> <p>Note: The registers storing the positioning program can not be repeated in using!</p> <ul style="list-style-type: none"> <li>Format and example for the positioning program 1:</li> </ul> <pre> 001 SPD 5000      ; Pulse frequency = 5KHz.       DRV ADR,+,10000,Ut ; Moving forward 10000 units.       WAIT Time,100      ; Wait for 1 second.       GOTO NEXT         ; Perform the next step.  002 SPD R1000      ; Pulse frequency is stored in DR1000 (R1001 and R1000).       DRV ADR,+,D100,Ut ; Moving forward, the stroke is stored in DD100 (D101 and D100).       WAIT Time,R500     ; The waiting time is stored in R500.       GOTO NEXT         ; To perform the next step.  003 SPD R1002      ; Pulse frequency is stored in DR1002 (R1003 and R1002).       DRV ADR,+,D102,Ut ; Moving backward, the stroke is stored in DD102 (D103 and D102).       EXT X0            ; When external trigger X0 (slow down point) ON, it performs the next       GOTO NEXT         ; step immediately.  004 SPD 2000      ; Pulse frequency = 2KHz.       DRV ADR,+,R4072,Ps ; Keep outputting the remain (stored in DR4072).       WAIT X1            ; Wait until X1 ON,       GOTO 1             ; Perform the first step.     </pre> <ul style="list-style-type: none"> <li>Format and example for the positioning program 2:</li> </ul> <pre> 001 SPD R0          ; Pulse frequency is stored in DR0 (R1 &amp; R0).       DRV ABS,,D0,Ut   ; Move to the position stored in DD0 (D1 &amp; D0).       WAIT M0          ; Wait until M0 ON,       GOTO NEXT        ; Perform the next step.  002 SPD R2          ; Pulse frequency is stored in DR2 (R3 &amp; R2).       DRV ADR,,D2,Ut   ; Moving stroke is stored in DD2 (D3 &amp; D2); working direction determined                         ; by the sign of setting value       MEND             ; End of positioning program     </pre>		

## Example for FUN140 Program Application

### Program example: Jog forward

As the jog forward button has been pressed for less than 0.5 second (changeable), it sends out only one (changeable) pulse;

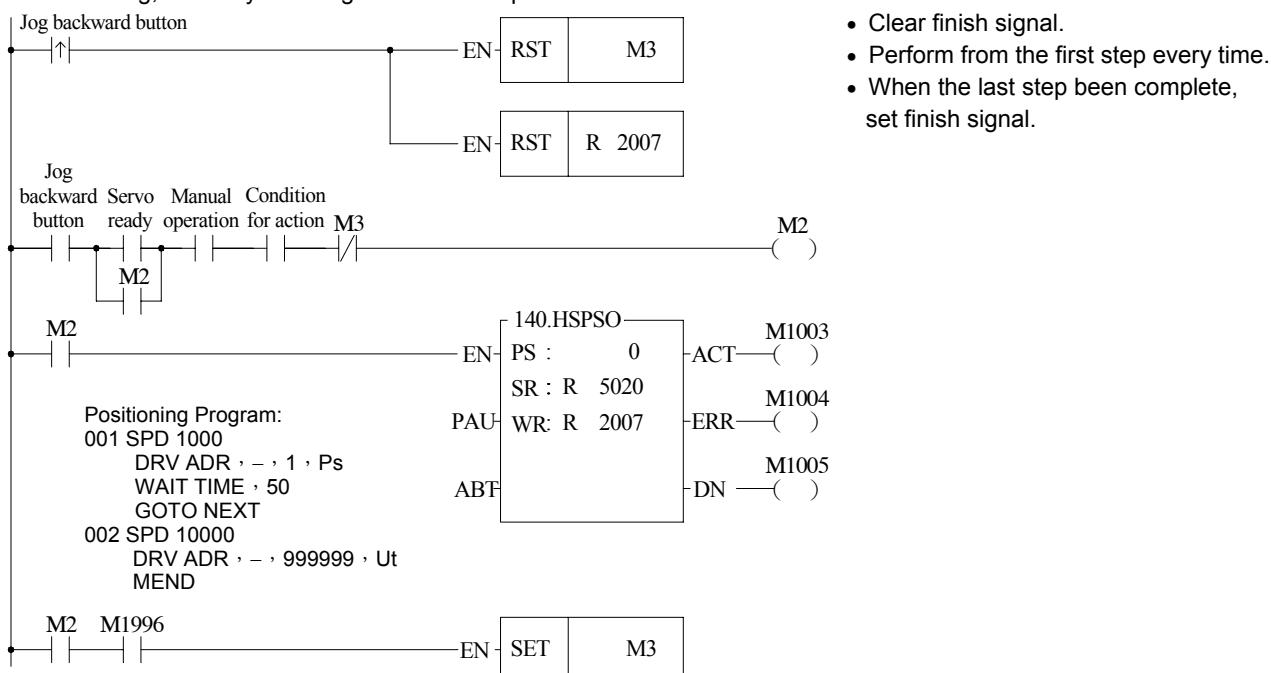
As the jog forward button has been pressed for more than 0.5 second (changeable), it continuously sends pulses out (the frequency is 10KHz, changeable), until the release of the jog forward button to stop the pulse transmitting; or it may be designed to send N pulses out at the most.

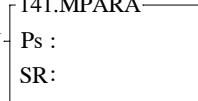


### Program example: Jog Backward

As the jog backward button has been pressed for less than 0.5 second (changeable) it sends out only one (changeable) pulse;

As the jog backward button has been pressed for more than 0.5 second (changeable), it continuously sends pulses out (the frequency is 10KHz, changeable), until the release of the jog backward button to stop the pulse transmitting; or it may be designed to send N pulses out at the most.



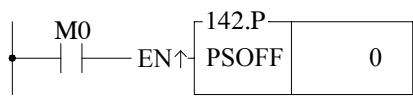
FUN 141 MPARA	Instruction of Parameter Setting for Positioning Program	FUN 141 MPARA																																																																						
Execution control - EN	<p style="text-align: center;">141.MPARA</p>  <p>Ps: The set number of Pulse Output (0~3). SR: Starting register for parameter table, it has totally 18 parameters which controlled by 24 registers.</p> <table border="1" data-bbox="595 494 976 651"> <thead> <tr> <th>Range</th> <th>HR</th> <th>DR</th> <th>ROR</th> <th>K</th> </tr> </thead> <tbody> <tr> <td>Ope- rand</td> <td>R0   R3839</td> <td>D0   D3071</td> <td>R5000   R8071</td> <td></td> </tr> <tr> <td>Ps</td> <td></td> <td></td> <td></td> <td>0~3</td> </tr> <tr> <td>SR</td> <td>○</td> <td>○</td> <td>○</td> <td></td> </tr> </tbody> </table>	Range	HR	DR	ROR	K	Ope- rand	R0   R3839	D0   D3071	R5000   R8071		Ps				0~3	SR	○	○	○																																																				
Range	HR	DR	ROR	K																																																																				
Ope- rand	R0   R3839	D0   D3071	R5000   R8071																																																																					
Ps				0~3																																																																				
SR	○	○	○																																																																					
<p><b>Instruction explanation</b></p> <ol style="list-style-type: none"> <li>This instruction is not necessary if the system default for parameter value is matching what users need. However, if it needs to open the parameter value to do dynamic modification, this instruction is required.</li> <li>This instruction incorporates with FUN140 for positioning control purpose.</li> <li>Whether the execution control input "EN" = 0 or 1, anyway, this instruction will be performed.</li> <li>When there is error in parameter value, the output indication "ERR" will be ON, and the error code is appeared in the error code register.</li> </ol> <p>Explanation for the parameter table:</p> <p>SR =Starting register of parameter table, suppose it is R2000.</p> <table border="1" data-bbox="245 1066 1277 2021"> <tbody> <tr> <td>R2000 (SR+0)</td> <td>0~2</td> <td>Parameter 0</td> <td>System default =1</td> </tr> <tr> <td>R2001 (SR+1)</td> <td>1~65535 Ps/Rev</td> <td>Parameter 1</td> <td>System default =2000</td> </tr> <tr> <td>DR2002 (SR+2)</td> <td>1~999999 <math>\mu</math>M/Rev 1~999999 mDeg/Rev 1~999999 <math>\times</math> 0.1 mInch/Rev</td> <td>Parameter 2</td> <td>System default =2000</td> </tr> <tr> <td>R2004 (SR+4)</td> <td>0~3</td> <td>Parameter 3</td> <td>System default =2</td> </tr> <tr> <td>DR2005 (SR+5)</td> <td>10~512000 Ps/Sec 1~153000</td> <td>Parameter 4</td> <td>System default =512000</td> </tr> <tr> <td>DR2007 (SR+7)</td> <td>10~512000 Ps/Sec 1~153000</td> <td>Parameter 5</td> <td>System default =10000</td> </tr> <tr> <td>R2009 (SR+9)</td> <td>0~10000 Ps/Sec 0~153000</td> <td>Parameter 6</td> <td>System default =0</td> </tr> <tr> <td>R2010 (SR+10)</td> <td>0~32767</td> <td>Parameter 7</td> <td>System default =0</td> </tr> <tr> <td>R2011 (SR+11)</td> <td>0~30000</td> <td>Parameter 8</td> <td>System default =5000</td> </tr> <tr> <td>R2012 (SR+12)</td> <td>0~1</td> <td>Parameter 9</td> <td>System default =0</td> </tr> <tr> <td>R2013 (SR+13)</td> <td>-32768~32767</td> <td>Parameter 10</td> <td>System default =0</td> </tr> <tr> <td>R2014 (SR+14)</td> <td>-32768~32767</td> <td>Parameter 11</td> <td>System default =0</td> </tr> <tr> <td>R2015 (SR+15)</td> <td>Reserved</td> <td>Parameter 12</td> <td>System default =0</td> </tr> <tr> <td>R2016 (SR+16)</td> <td>Reserved</td> <td>Parameter 13</td> <td>System default =1</td> </tr> <tr> <td>DR2017 (SR+17)</td> <td>-999999~999999</td> <td>Parameter 14</td> <td>System default =0</td> </tr> <tr> <td>DR2019 (SR+19)</td> <td>10~512000 Ps/Sec 1~153000</td> <td>Parameter 15</td> <td>System default =20000</td> </tr> <tr> <td>DR2021 (SR+21)</td> <td>10~512000 Ps/Sec 1~153000</td> <td>Parameter 16</td> <td>System default =1000</td> </tr> <tr> <td>R2023 (SR+23)</td> <td>0~255</td> <td>Parameter 17</td> <td>System default =10</td> </tr> </tbody> </table>	R2000 (SR+0)	0~2	Parameter 0	System default =1	R2001 (SR+1)	1~65535 Ps/Rev	Parameter 1	System default =2000	DR2002 (SR+2)	1~999999 $\mu$ M/Rev 1~999999 mDeg/Rev 1~999999 $\times$ 0.1 mInch/Rev	Parameter 2	System default =2000	R2004 (SR+4)	0~3	Parameter 3	System default =2	DR2005 (SR+5)	10~512000 Ps/Sec 1~153000	Parameter 4	System default =512000	DR2007 (SR+7)	10~512000 Ps/Sec 1~153000	Parameter 5	System default =10000	R2009 (SR+9)	0~10000 Ps/Sec 0~153000	Parameter 6	System default =0	R2010 (SR+10)	0~32767	Parameter 7	System default =0	R2011 (SR+11)	0~30000	Parameter 8	System default =5000	R2012 (SR+12)	0~1	Parameter 9	System default =0	R2013 (SR+13)	-32768~32767	Parameter 10	System default =0	R2014 (SR+14)	-32768~32767	Parameter 11	System default =0	R2015 (SR+15)	Reserved	Parameter 12	System default =0	R2016 (SR+16)	Reserved	Parameter 13	System default =1	DR2017 (SR+17)	-999999~999999	Parameter 14	System default =0	DR2019 (SR+19)	10~512000 Ps/Sec 1~153000	Parameter 15	System default =20000	DR2021 (SR+21)	10~512000 Ps/Sec 1~153000	Parameter 16	System default =1000	R2023 (SR+23)	0~255	Parameter 17	System default =10
R2000 (SR+0)	0~2	Parameter 0	System default =1																																																																					
R2001 (SR+1)	1~65535 Ps/Rev	Parameter 1	System default =2000																																																																					
DR2002 (SR+2)	1~999999 $\mu$ M/Rev 1~999999 mDeg/Rev 1~999999 $\times$ 0.1 mInch/Rev	Parameter 2	System default =2000																																																																					
R2004 (SR+4)	0~3	Parameter 3	System default =2																																																																					
DR2005 (SR+5)	10~512000 Ps/Sec 1~153000	Parameter 4	System default =512000																																																																					
DR2007 (SR+7)	10~512000 Ps/Sec 1~153000	Parameter 5	System default =10000																																																																					
R2009 (SR+9)	0~10000 Ps/Sec 0~153000	Parameter 6	System default =0																																																																					
R2010 (SR+10)	0~32767	Parameter 7	System default =0																																																																					
R2011 (SR+11)	0~30000	Parameter 8	System default =5000																																																																					
R2012 (SR+12)	0~1	Parameter 9	System default =0																																																																					
R2013 (SR+13)	-32768~32767	Parameter 10	System default =0																																																																					
R2014 (SR+14)	-32768~32767	Parameter 11	System default =0																																																																					
R2015 (SR+15)	Reserved	Parameter 12	System default =0																																																																					
R2016 (SR+16)	Reserved	Parameter 13	System default =1																																																																					
DR2017 (SR+17)	-999999~999999	Parameter 14	System default =0																																																																					
DR2019 (SR+19)	10~512000 Ps/Sec 1~153000	Parameter 15	System default =20000																																																																					
DR2021 (SR+21)	10~512000 Ps/Sec 1~153000	Parameter 16	System default =1000																																																																					
R2023 (SR+23)	0~255	Parameter 17	System default =10																																																																					

## NC Positioning Instruction

FUN 141 MPARA	Instruction of Parameter Setting for Positioning Program			FUN 141 MPARA			
Explanation for the parameter:							
<ul style="list-style-type: none"> <li>Parameter 0: The setting of unit, its default is 1.           <ul style="list-style-type: none"> <li>When the setting value is 0, the moving stroke and speed setting in the positioning program will all be assigned with the unit of mm, Deg, Inch, so called machine unit.</li> <li>When the setting value is 1, the moving stroke and speed setting in the positioning program will all be assigned with the unit of Pulse, so called motor unit.</li> <li>When the setting value is 2, the moving stroke setting in the positioning program will all be assigned with the unit of mm, Deg, Inch, and the speed setting will all be assigned with the unit of Pulse/Sec, which is called as compound unit.</li> </ul> </li> </ul>							
Parameter 0, unit setting	“0” machine unit	“1” motor unit	“2” compound unit				
Parameter 1, 2	Must be set	No need to set	Must be set				
Parameter 3, 7, 10, 11	mm , Deg , Inch	Ps	mm , Deg , Inch				
Parameter 4,5,6,15,16	Cm/Min , Deg/Min , Inch/Min	Ps/Sec	Ps/Sec				
<ul style="list-style-type: none"> <li>Parameter 1: Pulse count/1 revolution, its default is 2000, i.e. 2000 Ps/Rev.           <ul style="list-style-type: none"> <li>The pulse counts needed to turn the motor for one revolution  <math>A = 1 \sim 65535</math> (for value greater than 32767, it is set with hexadecimal) Ps/Rev             </li> </ul> </li> </ul>							
<ul style="list-style-type: none"> <li>Parameter 2: Movement/1 revolution, its default is 2000, i.e. 2000 Ps/Rev.           <ul style="list-style-type: none"> <li>The movement while motor turning for one revolution.  <math>B = 1 \sim 999999 \mu\text{M}/\text{Rev}</math>  <math>1 \sim 999999 \text{ mDeg}/\text{Rev}</math>  <math>1 \sim 999999 \times 0.1 \text{ mlInch}/\text{Rev}</math> </li> </ul> </li> </ul>							
<ul style="list-style-type: none"> <li>Parameter 3: The resolution of moving stroke setting, its default is 2.</li> </ul>							
<span style="border-bottom: 1px solid black; padding-bottom: 2px;">Parameter 0</span> <span style="border-bottom: 1px solid black; padding-bottom: 2px;">Parameter 3</span>	Set value=0, machine unit; Set value=2, compound unit;	Set value=1, motor unit					
	mm	Deg	Inch	Ps			
Set value =0	$\times 1$	$\times 1$	$\times 0.1$	$\times 1000$			
Set value =1	$\times 0.1$	$\times 0.1$	$\times 0.01$	$\times 100$			
Set value =2	$\times 0.01$	$\times 0.01$	$\times 0.001$	$\times 10$			
Set value =3	$\times 0.001$	$\times 0.001$	$\times 0.0001$	$\times 1$			
<ul style="list-style-type: none"> <li>Parameter 4: The limited speed setting, its default is 512000, i.e. 512000 Ps/Sec.           <ul style="list-style-type: none"> <li>Motor and compound unit: <math>10 \sim 512000 \text{ Ps/Sec}</math>.</li> <li>Machine unit: <math>1 \sim 153000</math> (cm/Min, <math>\times 10</math> Deg/Min, Inch/Min).</li> </ul> <p style="text-align: center;">However, the limited frequency can't be greater than 512000 Ps/Sec.</p> <math display="block">f_{\max} = (V_{\max} \times 1000 \times A) / (6 \times B) \leq 512000 \text{ Ps/Sec}</math> <math display="block">f_{\min} \geq 10 \text{ Ps/Sec}</math> </li> </ul>							
<p>Note: A = Parameter 1, B =Parameter 2.</p>							

FUN 141 MPARA	Instruction of Parameter Setting for Positioning Program	FUN 141 MPARA
<ul style="list-style-type: none"> <li>● Parameter 5: Reserved, it is recommended for jog speed, the default = 10000 Ps/Sec.</li> <li>● Parameter 6: Initiate/Stop speed, the default = 0. <ul style="list-style-type: none"> <li>• Motor and compound unit: 0~10000 Ps/Sec.</li> <li>• Machine unit: 0~15300 (cm/Min, ×10 Deg/Min, Inch/Min).</li> </ul> <p style="margin-left: 20px;">However, the limited frequency can't be greater than 512000 Ps/Sec.</p> </li> <li>● Parameter 7: Backlash compensation, the default =0. <ul style="list-style-type: none"> <li>• Setting range: 0~32767 Ps.</li> <li>• While backward traveling, the traveling distance will be added with this value automatically.</li> </ul> </li> <li>● Parameter 8: Acceleration/Deceleration time setting, the default = 5000, and the unit is mS. <ul style="list-style-type: none"> <li>• Setting range: 0~30000 mS.</li> <li>• The setting value represents the time required to accelerate from idle state upto limited speed state or decelerate from the limited speed state down to the idle state.</li> </ul> </li> <li>● Parameter 9: Coordinate direction setting, the default =0. <ul style="list-style-type: none"> <li>• Setting value =0, while in forward pulse output, the current Ps value is adding up. While in backward pulse output, the current Ps value is deducting down.</li> <li>• Setting value =1, while in forward pulse output, the current Ps value is deducting down. While in backward pulse output, the current Ps value is adding up.</li> </ul> </li> <li>● Parameter 10: Forward movement compensation, the default = 0. <ul style="list-style-type: none"> <li>• Setting range: -32768~32767 Ps.</li> <li>• When it is in forward pulse output, it will automatically add with this value as the moving distance.</li> </ul> </li> <li>● Parameter 11: Backward movement compensation, the default =0. <ul style="list-style-type: none"> <li>• Setting range: -32768~32767 Ps.</li> <li>• When it is in backward pulse output, it will automatically add with this value as the moving distance.</li> </ul> </li> <li>● Parameter 12: Reserved.</li> <li>● Parameter 13: Reserved.</li> <li>● Parameter 14: Reserved, it is recommended to be used as machine's home position, the default = 0.</li> <li>● Parameter 15: Reserved, it is recommended to be used as return home speed, the default = 20000 Ps/Sec.</li> <li>● Parameter 16: Reserved, it is recommended to be used as slow down speed while returning home , the default = 1000 Ps/Sec.</li> <li>● Parameter 17: Reserved, it is recommended to be used as the setting value of Z phase count, the default = 10.</li> </ul>		

## NC Positioning Instruction

FUN 142 PSOFF	Enforcing to Stop Pulse Output	FUN 142 PSOFF
Stop control EN↑		N: 0~3, enforces the assigned set number of Pulse Output to stop its output.
<b>Instruction Explanation</b>		<ol style="list-style-type: none"><li>1. When stop control “EN” =1, or “EN ↑” (P instruction) changes from 0→1, this instruction will enforce the assigned set number of Pulse Output to stop its output.</li><li>2. When applying in the process of return home , as the home has returned, it can immediately stop the pulse output by using this instruction, so as to make it stop at the same position every time when performing machine homing.</li></ol>
<b>Program example</b>		 <p>; When M0 changes from 0→1, it enforces the Ps0 to stop the pulse output.</p>

FUN 143 PSCNV	Converting the Current Pulse Value to the Displaying Value (mm, Deg, Inch, PS)	FUN 143 PSCNV
------------------	---	------------------

Execution control-EN↑ [ 143.PSCNV ]  
 Ps : \_\_\_\_\_  
 D : \_\_\_\_\_

Ps: 0~3; converting the assigned pulse position to mm (Deg, Inch, PS) which has the same unit as the set point, so as to make the current position displayed.

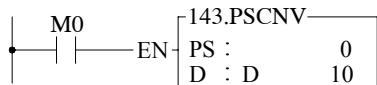
D: Registers that store the current position after conversion.  
 It uses 2 registers, e.g. D10 represents D10 (Low Word) and D11 (High Word) two registers.

Range	HR	DR	ROR	K
	R0	D0	R5000	
Ps				0~3
D	○	○	○*	

#### Instruction Explanation

- When execution control “EN” =1 or “EN ↑ ” (P instruction) changes from 0→1, this instruction will convert the assigned current pulse position (PS) to be the mm (or Deg, Inch, or PS) that has the same unit as the set value, so as to make current position displaying.
- After the FUN140 instruction has been performed, it will then be able to get the correct conversion value by executing this instruction.

#### Program Example



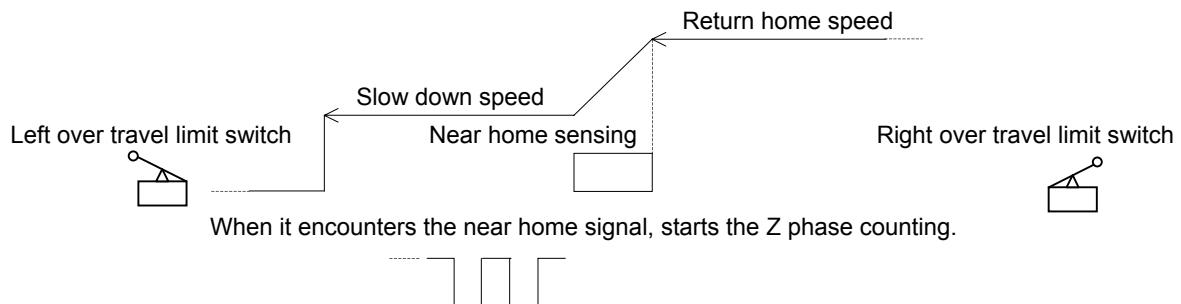
; When M0=1, it converts the current pulse position of Ps0 (DR4088) to the mm (or Deg or Inch or PS) that has the same unit as the set value, and store it into the DD10 to make the current position displaying.

## 14.7 Machine Homing

The machine set which undertakes relative model Encoder as shifting detector usually need the reset action for the reference of positioning coordinate; we called this action as machine homing (seeking for zero reference).

The machine homing diagram for NC servo unit is as follows:

Method 1:



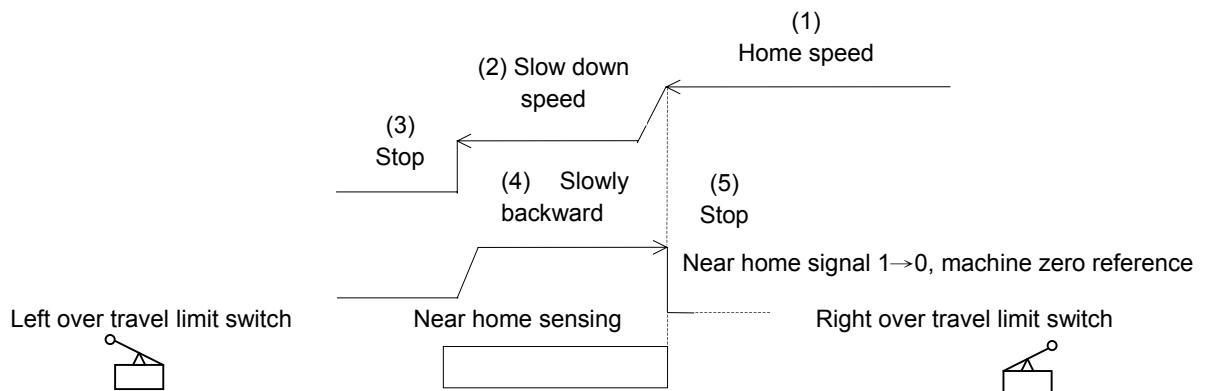
Z phase counting is up, the pulse output stops, then send out the CLR signal to clear the error counter of servo driver.

e.g.:

X3: Near home sensing input is configured as interrupt input; in the case of machine homing, it starts HSC4 to begin counting in INT3 interrupt service subroutine.

X2: Z phase counting input, it is configured as UP input of HSC4; the X2+ is prohibited to interrupt in regular time, when executing machine homing and X3 near home interrupt occurred, it starts HSC4 to begin Z phase counting. When HSC4 counting is up, it stops the pulse output, prohibit the X2+ interrupt, set home position to signal, and sends out the CLR signal to clear the error counter of servo driver. Please consult program example.

Method 2: According to application demand, it may slow down when encountering the near home sensor, while over the sensor a little far away, stop the pulse output, and then traveling slowly with backward direction; the very moment when it get out of near home sensor (the sensing signal changes from 1→0), it is treated as machine home. This program is simpler!



X3: Near home sensing input; it is configured as falling edge interrupt input.

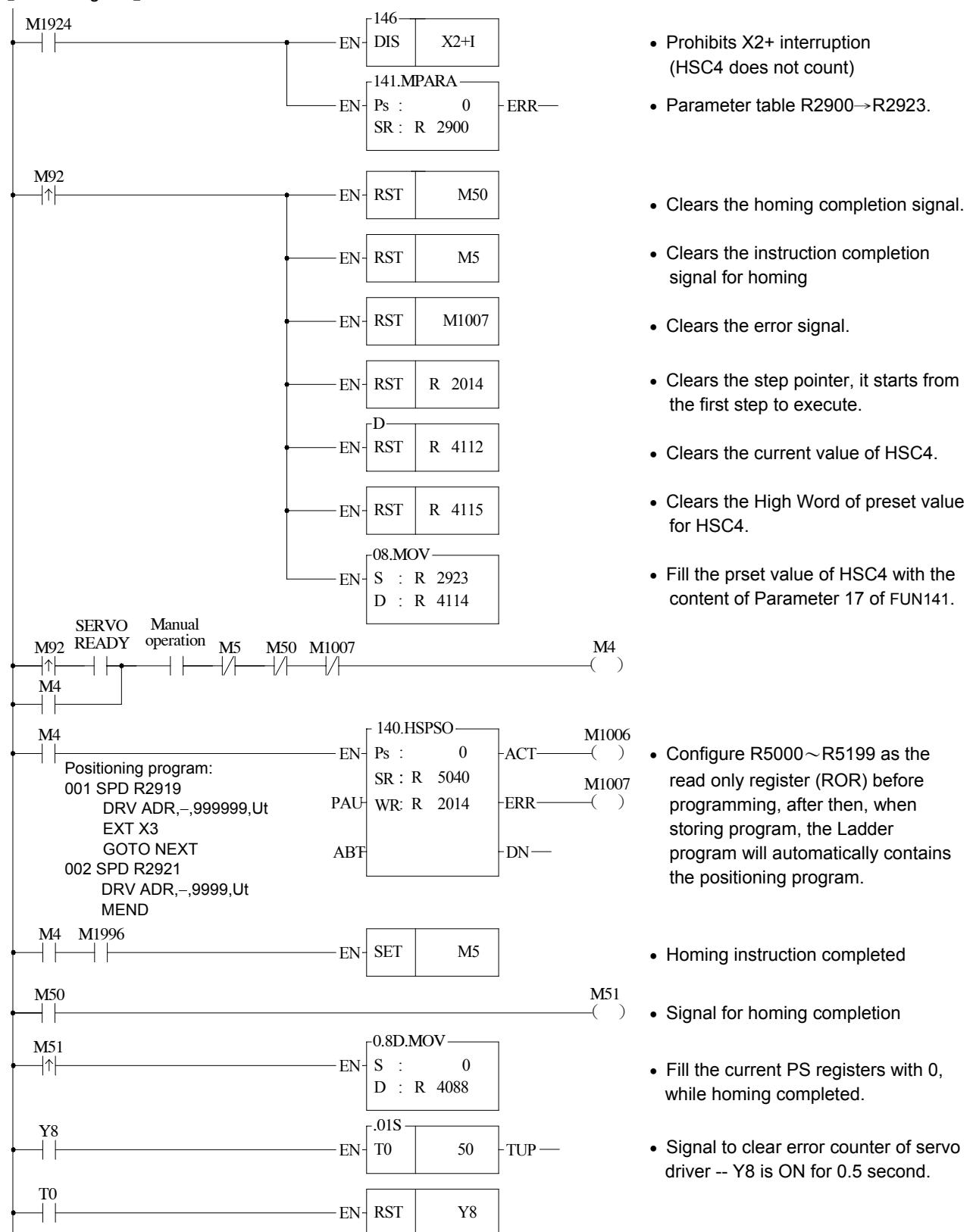
- Once encountering the near home sensor, it will enable X3 falling edge interrupt, and slow down to stop within the near home sensing range.
- Slowly backward traveling until the near home sensing signal changes from 1→0.
- When the near home sensing signal changes from 1→0, it performs the INT3- interrupt service subroutine immediately.
- The INT3- interrupt service subroutine: Stops the pulse output immediately, prohibits the X3- interrupt, sets home position to signal, and sends out CLR signal to clear the error counter of servo driver. (Please consult the example program.)

### Program Example 1: Machine homing (method 1)

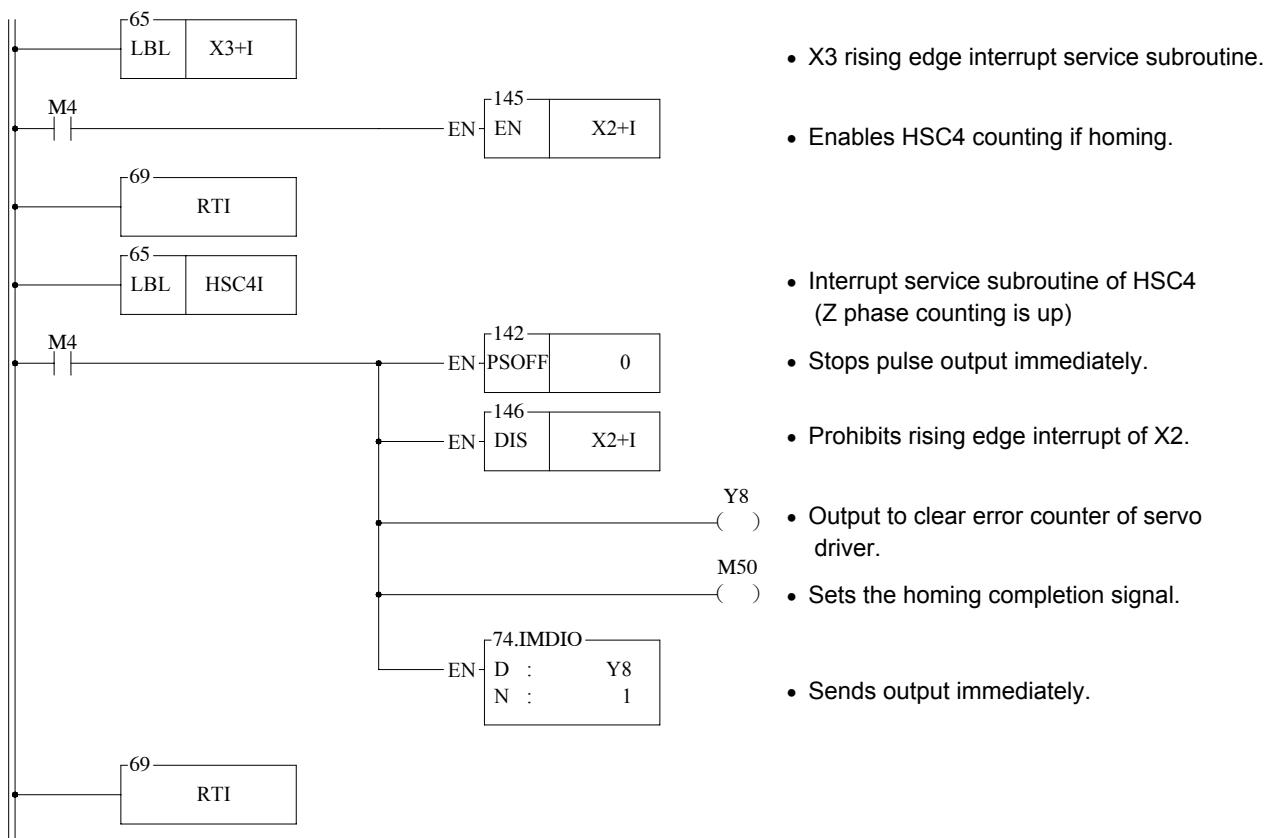
X2: Configured as the UP input of HSC4, and connected to Z phase input.

X3: Configured as the rising edge interrupt input, and connected to near home sensing input.

#### Main Program



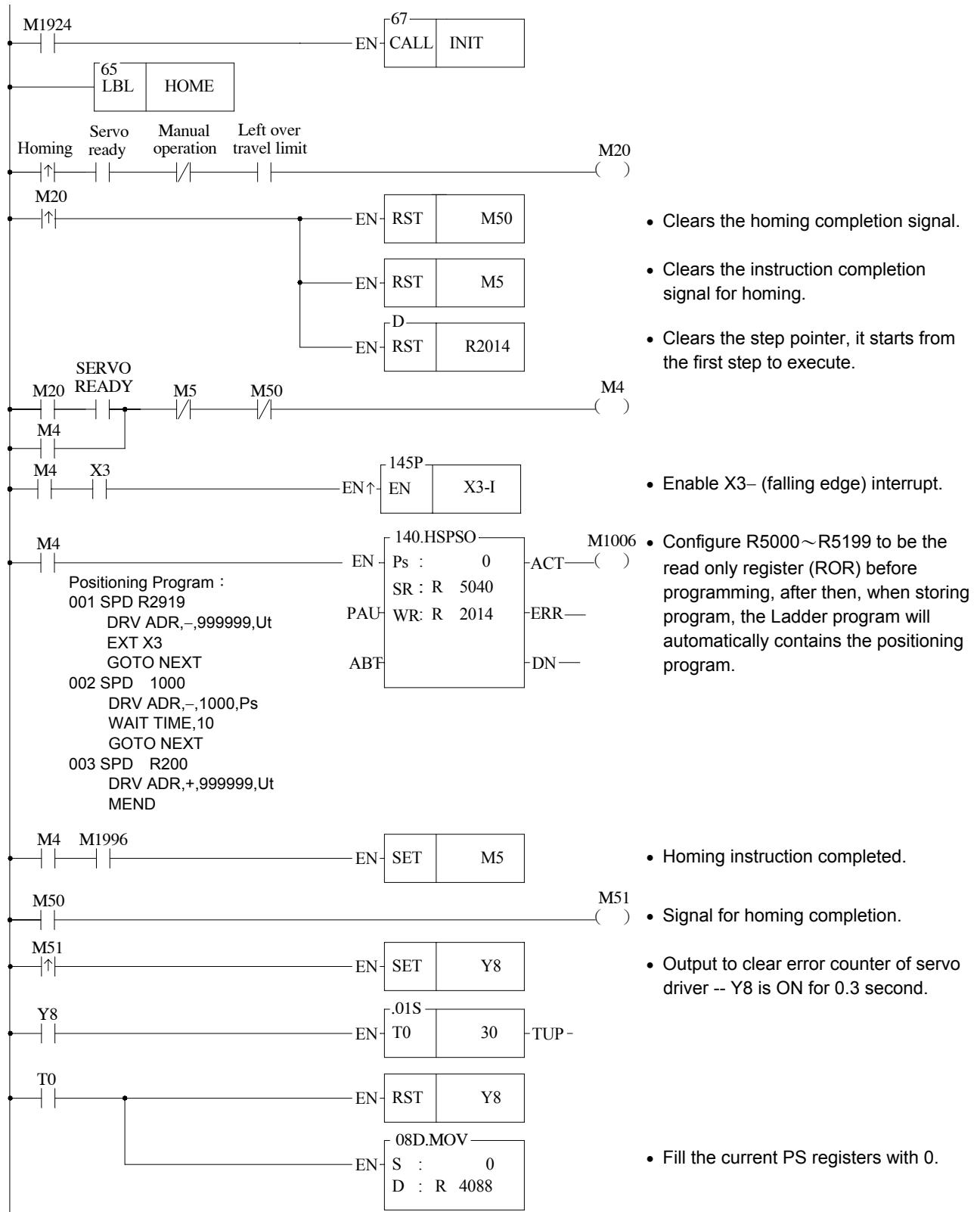
【Sub Program】



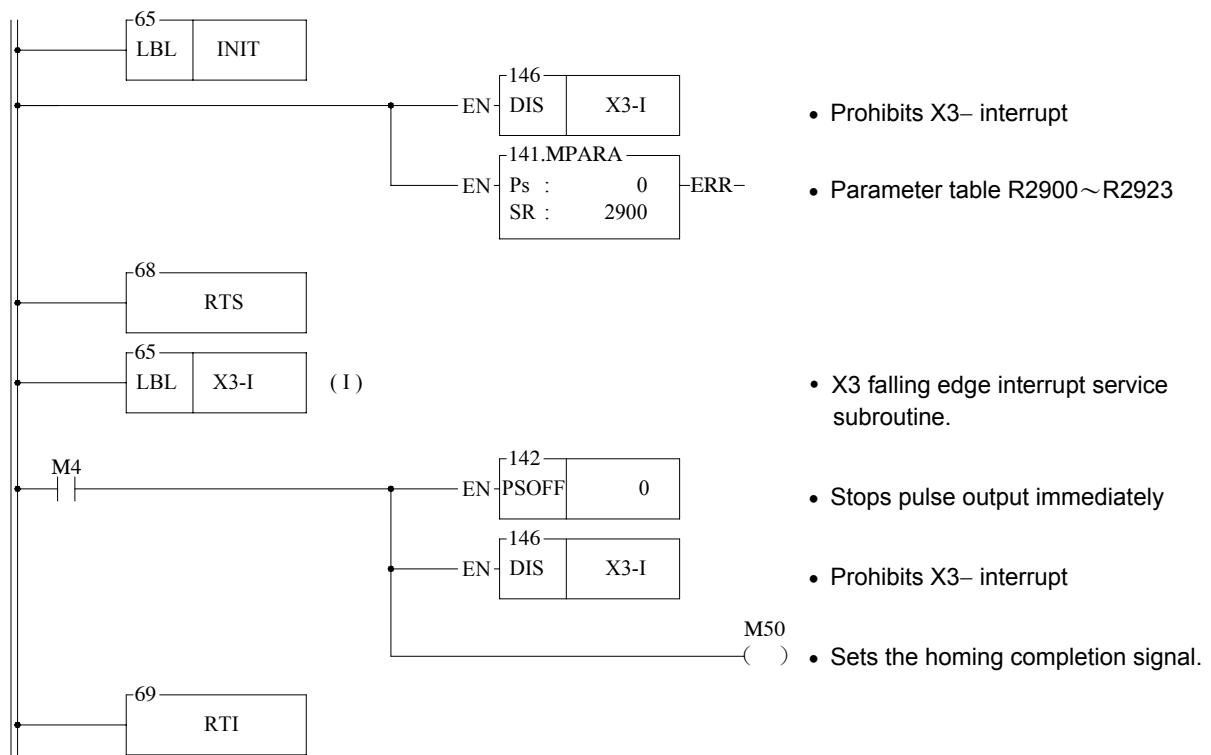
## Program Example 2: Machine homing (method 2)

X3: Connected to near home sensing input, and configured as falling edge interrupt input.

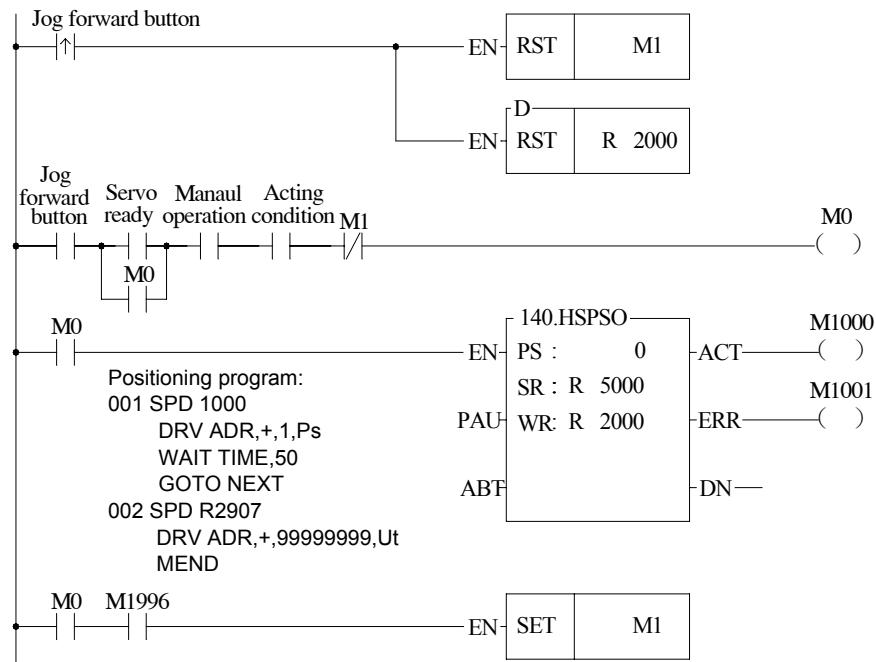
### 【Main Program】



【Sub Program】



### Program Example 3: JOG Forward

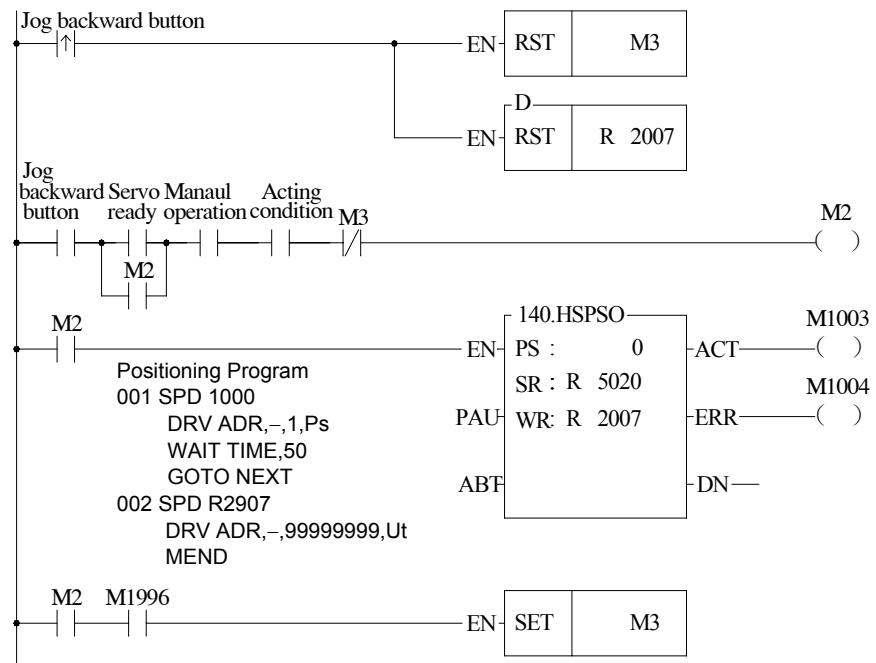


- Clears the completion signal

- Starts from the first step every jog execution.

- As the execution of last step completed, it sets up the completion signal.

### Program Example 4: JOG Backward



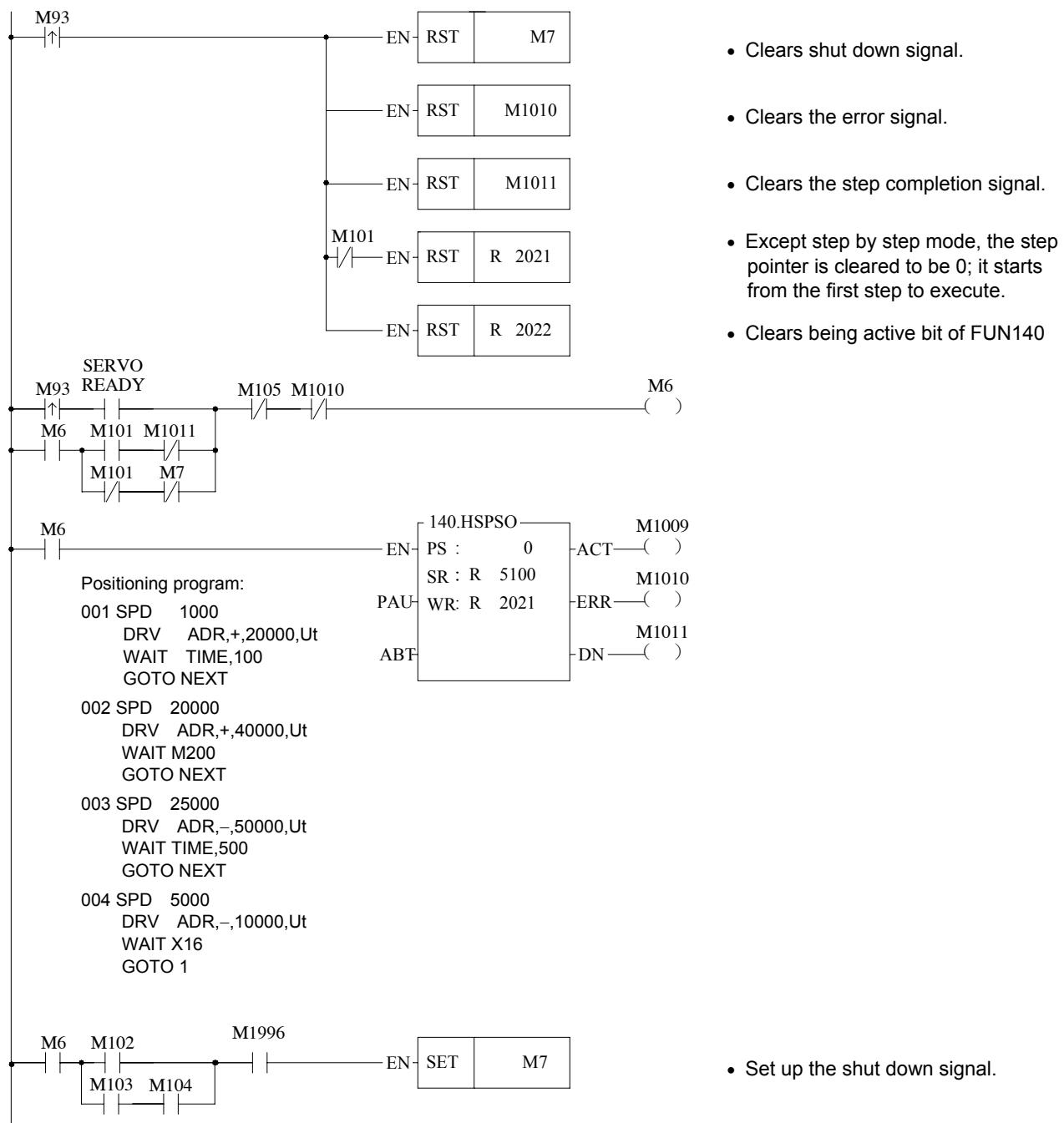
- Clears the completion signal.

- Starts from the first step every jog execution.

- As the execution of the last step completed, it sets up the completion signal.

**Program Example 5: Step by step, One cycle, Continuous positioning control.**

M93 : Start  
 M101 : Step by step operation mode  
 M102 : One cycle operation mode  
 M103 : Continuous operation mode  
 M104 : Regular shut down.  
 M105 : Emergency stop.



# Chapter 15 Application of ASCII Output Function

The FB-PLC's ASCII file output function allows the PLC to directly drive ASCII output devices such as printers and terminals, and let them print or display English document data or display screens such as production reports, materials details and warning messages. For application of the ASCII file output function, it is first necessary to set the port 1 communication port on the hardware to ASCII output interface (mode 2) as described above. Besides this, the ASCII file data to be output must be edited to fit the required format of the FB-PLC FUN 94 (ASCWR) instruction. Then using this instruction, it will be sent out via port 1 to the ASCII output device connected with port 1.

## 15.1 Format of ASCII File Data

ASCII file data may be divided into fixed, unchanging background file data and dynamically changing variable data. The background file data may be in English characters, numerals, symbols, graphs, etc, and the variable data can only be printed out as binary, decimal, or hexadecimal numeric value data.

ASCII code is a byte length code, which has a total of 256 combinations. Of these, the first 128 (0-127) are fairly clearly defined and are used by most of the ASCII peripherals. For codes greater than 128 each manufacturer has different definitions and graphics and there are no uniform specifications. FB-PLC designed the FUN 94 (ASCWR) instruction to be solely responsible for transmission, and not for editing. This work is done by the ASCII editor of the PROLADDER software package. Below is the editing command format adopted by the PROLADDER software package editor.

### 1. Basic command Symbols

-  Linefeed

A line slanting down from right to left means that no matter where the printing is up to, if this symbol is encountered, then the printing head or the terminal display will move to the beginning (the very left) of the next line and go on printing or displaying from that point. A series of "/" will create a succession of linefeeds (one "/" will cause one linefeed).

-  Pagefeed

A line slanting down from left to right means that when this symbol is encountered the printing head or the terminal display will move to the beginning (top left hand corner) of the next page, and continue printing or displaying from that point. A series of "\" will create a succession of pagefeeds. (One "\"" will cause one pagefeed).

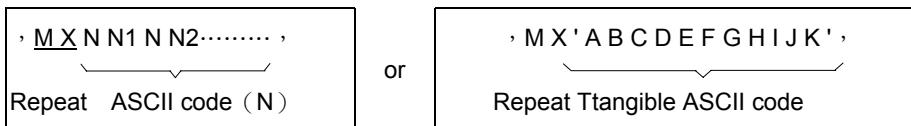
-  Comma

Used to separate statements in the file data. All the data included between two commas is a complete and executable statement (must not be used for beginning and end of file). Note that although the shape of a comma is the same as the shape of a single quotation mark, their positions are different (the comma is in a position near the center of the letter, while the single quotation mark is near the top right corner). The function meaning that they represent is completely different. Please refer to Item 2, background data format - statements.

-  File end

At the end of the ASCII file END is added to show that the ASCII file is finished.

## 2. Background Data Format



- MX:

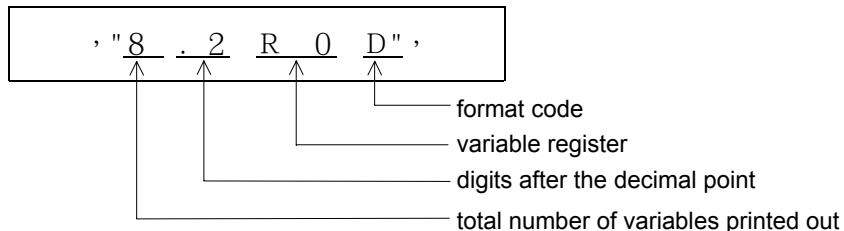
Represents the number of repeats. M can be 1 to 999. The ASCWR instruction can send out M times successively all the hexadecimal ASCII code or tangible ASCII code data contained between X and the first comma ( , ). If there is no data after X (ie, the comma comes directly after X), then the ASCWR instruction will send out M successive space codes. If you only have to send out the ASCII code or the tangible ASCII code once, then MX can be discarded.

- ASCII code data format: This data format has an N two-digit hexadecimal value. Every two adjoining hexadecimal numerals starting from the righthand side of X will be regarded as an ASCII code. NN can be any ASCII code, including tangible or intangible ASCII code such as English characters, numeric symbols or control codes. However, its main use is as a special tangible code for control codes which cannot be represented by tangible character fonts or cannot find a font or symbol on the PROLADDER ASCII editor. For tangible characters or symbols that can be directly represented on the ASCII editor by tangible keys, it should be more convenient to use the original printing out format. For example, if you want to print out the character "A", with the original printing method you can type A via the keyboard. But if you want to use ASCII code, you must check the table on which "A" is represented by 41 H, and then enter 41. It is obviously a lot less convenient.
- Original printing out tangible ASCII code data format: What is enclosed within two single quotation marks ' ', can only be tangible ASCII code such as English characters, numerals, symbols, and graphics (characters that can be found on or input via the ASCII editor keyboard). The ASCWR instruction will faithfully print out all the characters that are contained in ' ', so if you need to print out a single quotation mark itself, you must have two successive single quotation marks. For example:

**'I'M A BOY' will be printed out as I' M A BOY**

If the graphics or symbols of the ASCII output device cannot be found on the ASCII editor keyboard, then you naturally are unable to do input using this format. In such a case you can check the ASCII code for that symbol or graphic, and use ASCII code to input and print out.

## 3. Variable Data Format



A data statement within two double quotation marks " ", is used to specify the register address of variable data, and what format or format code it will be printed out .

- Total number of variables printed out: In this example, "8" are used to print out the reserved 8 digit columns of the variable (R0) numeric value (including negative signs). If the variable value is larger than the total number of printed out digits then the high digit will be cut out. If the number of digits is insufficient, the remaining positions will be occupied by spaces.
- Digits after the decimal point: The number of digits after the decimal point within the total number of digits of the variable. In this example, in a total number of 8 digits, there are 2 places after the decimal point. The decimal point symbol "." itself occupies one position so the integer will remain 5 digits.

- Variable register: Can be R, D,WX, WY,etc, of a 16-bit register, or DR,DD,DWX, DWY, etc, of a 32-bit register. The contents of these registers can be retrieved and printed out using the format and format code specified by the contents of " ".
- Format code: Can use hexadecimal H, decimal D or binary B format for printing out (when format code is not specified, it will be decimal - therefore D can be omitted).

This example assumes that the content value of R0 is -32768. In the 8.2 format, the print out result is

	-	3	2	7	.	6	8
--	---	---	---	---	---	---	---

If the format changes from 8.2 to 5.1 then the print out result changes to

2	7	6	.	8
---	---	---	---	---

## 15.2 Application Examples of ASCII File Output

The file data print out will start from the top left hand corner of each page. It will print from left to right with lines going from top to bottom (please refer to the format in the diagram below). When the final character in a line is reached (this varies according to the output device - a printer can have 80 characters or 132 characters), the printer will automatically jump to the start (left-hand side) of the next line. If it has not yet printed to the final character, but encounters the linefeed command (/) or the page feed command (\), then it will jump to the start of the next line or the next page, and start printing from that point.

Suppose that the production statistics table for the manufacturing division of a certain company has the following format. This can be used as an example to explain the editing and printing out of its ASCII file data. (For the hardware connections and mode settings please refer to Chapter 11).

The diagram illustrates the structure of an ASCII file output. At the top, there is a header section consisting of two lines of text: "PRODUCTION REPORT" and "DATE: 1/20/99". Below this is a data section containing seven entries, each with a label and a value. The data entries are:

16 spaces	TOTAL NUMBER	(A) :	1000 PCS
NUMBER OF YIELD	(B) :	983 PCS	
NUMBER OF REPAIR	(C) :	17 PCS	
STANDARD TIME	(D) :	8.5 MIN/PCS	
TOTAL WORKING TIME	(E) :	8500 MIN	
ACTUAL WORKING TIME	(F) :	9190 MIN	
EFFICIENCY	(G) :	92.49 %	

At the bottom, there is a REMARK line with the formula: A × D = E, E / F = G.

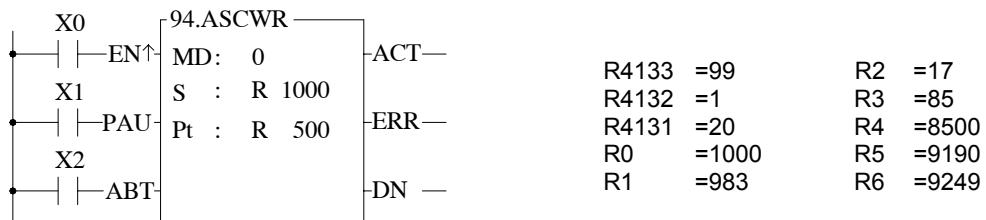
Before editing this file, you must first tell the file editor starting from which register within PLC the file to be edited shall be stored. When editing the file data, you must differentiate whether the file data to be edited (printed out) is fixed background data or variable data. The background data can be input using ASCII characters or symbol graphic of the original print out format (using what is contained inside ' '), or it can directly use the ASCII code of its character or symbol graphics. As for the variable data section, because it is stored in registers (so as long as the variable value changes, the print out numerical value will change with it), the print out message must contain the register number and print out format, such as number of characters, digits after the decimal point, etc, as well as the format code that is used for the print out (contained inside " "). In the example in the table above, the year, month, day data and the total number (A) to efficiency (G) figures are all variable data. It assumes that the year, month, day data accesses the year, month, day registers (R4133 to R4131) within the real time clock register RTCR. R0 stores the total number (A), R1 stores the number of yield (B), etc, and R6 stores the efficiency (G) value. Below is the ASCII file data for this statistical table example:

```
///,28X,'PRODUCTION REPORT',//,28X,'=====',
52X,'Date:','2R4132',//,"2R4131",'//,16X,'TOTAL NUMBER
(A) :,"10R0",' PCS',//,16X,'NUMBER OF YIELD (B) :,"10R1",
PCS',//,16X,'NUMBER TO REPAIR (C) :,"10R2",' PCS',//,16X,'STANDARD TIME
(D) :,"10.1R3",' MIN/PCS',//,16X,'TOTAL WORKING TIME (E) :,"10R4",
MIN',//,16X,'ACTUAL WORKING TIME(F) :,"10R5",' MIN',//,16X,'EFFICIENCY
(G) :," 10.2R6",' %',//,22X,'REMARK: AXD=E, E/F=G',END
```

\* : In the above example ' =====' can be replaced by 18X'=' or 18X3D.

During the process of file output, when the output reaches variable data, the CPU will retrieve and do output with the numerical values at that time of the register whose address are contained within the " ". Therefore, if a variable is printed out both at the beginning and end of a file, a different numerical value may be obtained (when it has printed to halfway the register value changes).

After the file editing has been completed, the FUN94 instruction can be used to print out its background and dynamic data. If this file is edited (stored) starting from R1000, then when it is outputting, S must be specified as R1000 before there can be an accurate output, as seen in the program example in the diagram below left. Supposing that the numerical value of the variable register is as shown in the diagram below right, then when X1 and X2 are 0, and X0 goes from 0 to 1, this instruction will print out the statistical table from the previous page, from Port 1 of PLC.

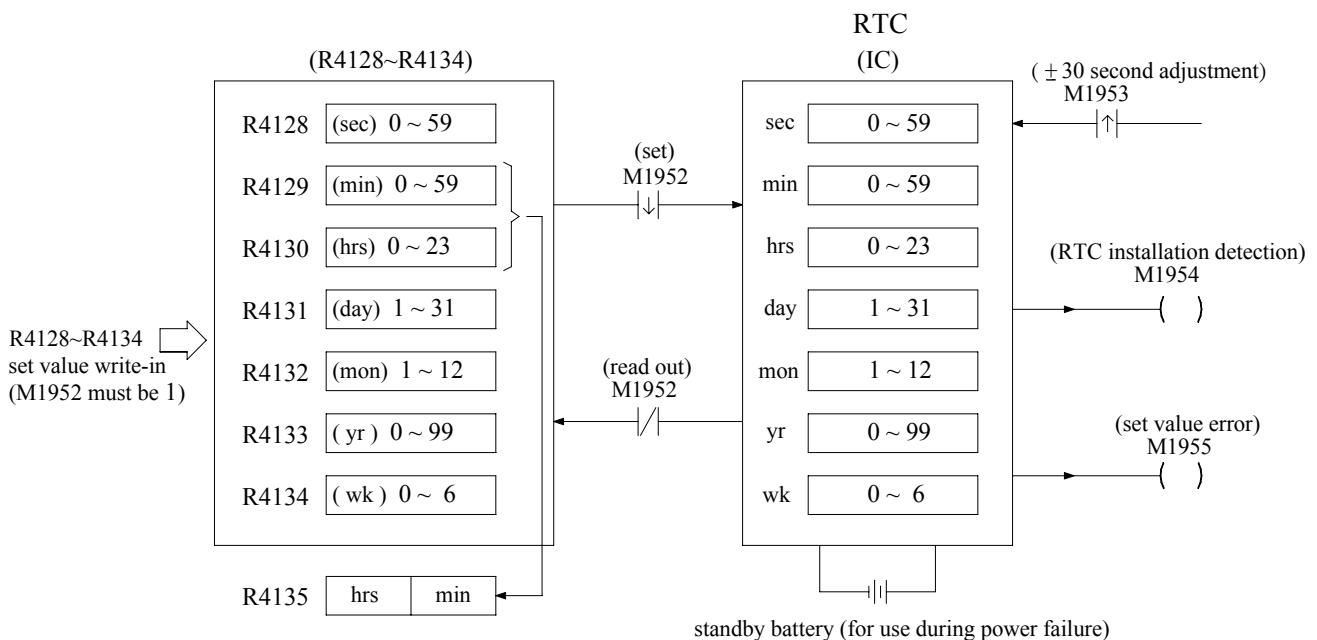


## Chapter 16 Real Time Clock (RTC)

A real time clock (RTC) is optional to be attached to the FB-PLC's CPU unit. No matter whether the PLC is switched on or off, the RTC will always keep accurate time. It provides 7 kinds of time value data-week, year, month, day, hour, minute and second. Users can take advantage of the real time clock to do 24 hour controls throughout the year (for example, businesses or factories can switch lights on and off at set times each day, control gate access, and do pre-cooling and pre-heating before business or operations begin). It can enable your control system to automatically coordinate with people's living schedules, and not only will it raise the level of automatic control, it will improve efficiency.

### 16.1 Correspondence between RTC and the RTCR within PLC

Within PLC, there are special purpose registers (RTCR) for storing the time values of the RTC. There are 8 RTCR registers in all, going from R4128 to R4135. R4128 to R4134 are used to store the 7 kinds of time values mentioned above, from weeks to seconds. Because in practical daily application, certain hour and minute time data is often used, we have specially merged the time values of the hour register (R4130) and minute register (R4129) within RTCR, and put them in R4135 high byte and low byte, so they can be accessed by the user. The diagram below shows the correspondence between RTC and the RTCR within PLC, as well as the control switch and status flag (M1952-M1955) related to RTC accessing.

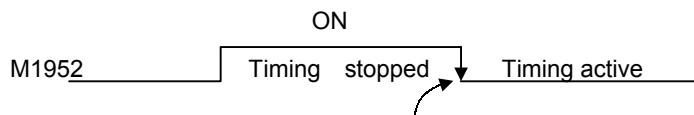


## 16.2 RTC Access Control and Setting

Within PLC, the RTCR registers have been allocated to store the time values of RTC, and this is of great convenience to the user. However, if you want to load the set values of RTCR into RTC or read out what is in RTC onto RTCR, and tune the time value etc, then the setting must be done using the special relays (M1952 and M1953) for RTC access. Below is an explanation of the access and adjustment procedures, and the status flag relays.

### 1. RTC setting:

The (RTCR→RTC) setting action is only executed once at the moment that relay M1952 goes from 1→0 (falling edge).



At the moment when M1952 goes from 1 to 0, the set values of R4128 to R4134 within RTCR will be written into the corresponding hardware registers within RTC. After M1952 has returned to 0 the timing action will start. Also, with each scan, CPU will retrieve time values from RTC in the opposite direction and write them onto RTCR.

**Note:** If you want to load the set values into RTC, you must first make M1952 as 1 and then load the set values into RTCR. The loading of the set values into RTCR can be done via MOVE instruction. However, you must first halt the RTC read out (make M1952 as 1), otherwise the data that you just wrote into RTCR will immediately be overridden by the time data being read back from RTC in the opposite direction.

### 2. RTC read out (RTC→RTCR):

whenever the M1952 relay is 0 (RTC timing active). With every scan, CPU will take the time value data within RTC and move it to RTCR. When it is 1, it will not read out. In this case RTCR can load in the set values and they won't be overridden.

### 3. ± 30 second adjustment:

At the moment that the status of relay M1953 goes from 0→1, CPU will check the value of the second register (R4128) within RTC. If its value is between 0 and 29 seconds then it will be cleared to 0. If its value is between 30 and 59 seconds then besides being cleared to 0, the minute register (R4129) will be increased by 1 (ie, one minute will be added). This can be used to adjust your RTC time value.

### 4. M1954 RTC installation detecting flag:

When RTC is fitted to the PLC, relay M1954 will be set as 1; otherwise it will be 0.

### 5. M1955 set value error flag:

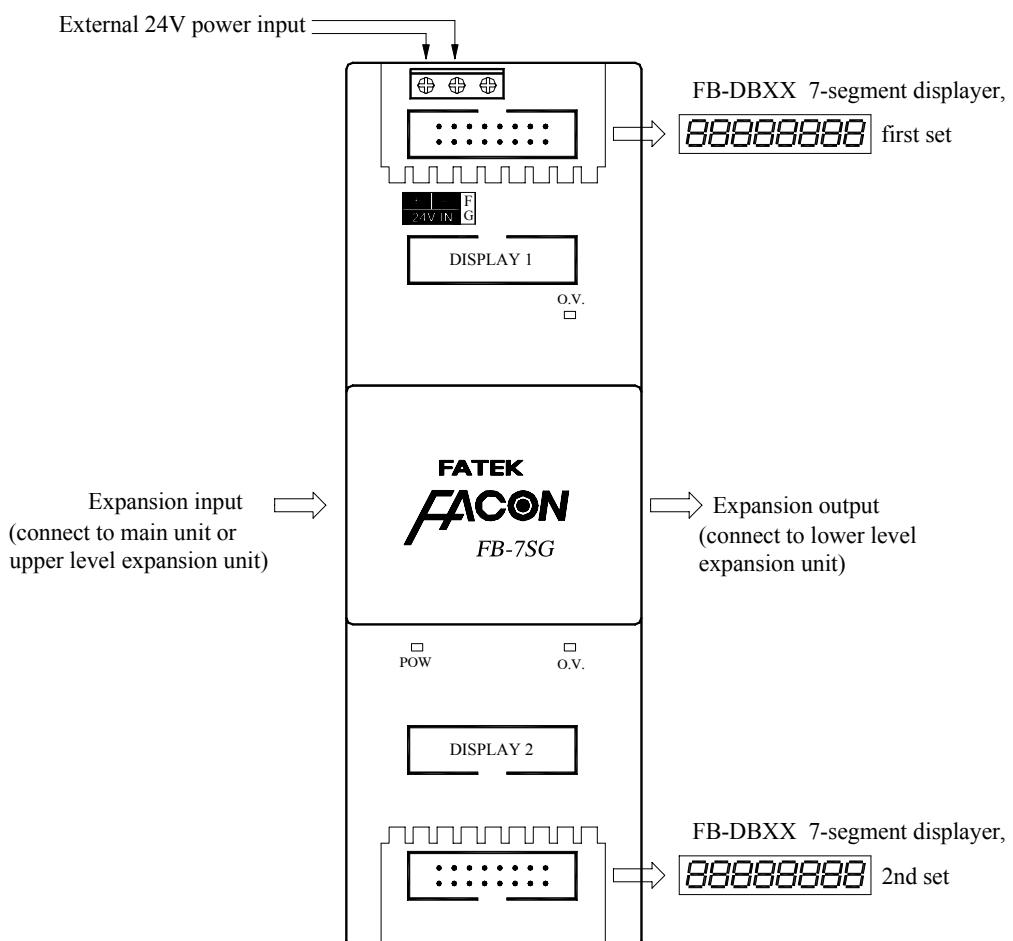
When the time value which is set to RTC's IC is illegal, then the error flag relay M1955 will be set as 1, and the setting action will not be executed.

# Chapter 17 FB-7SG 7-Segment LED Display Module

## 17.1 FB-7SG Brief Introduction

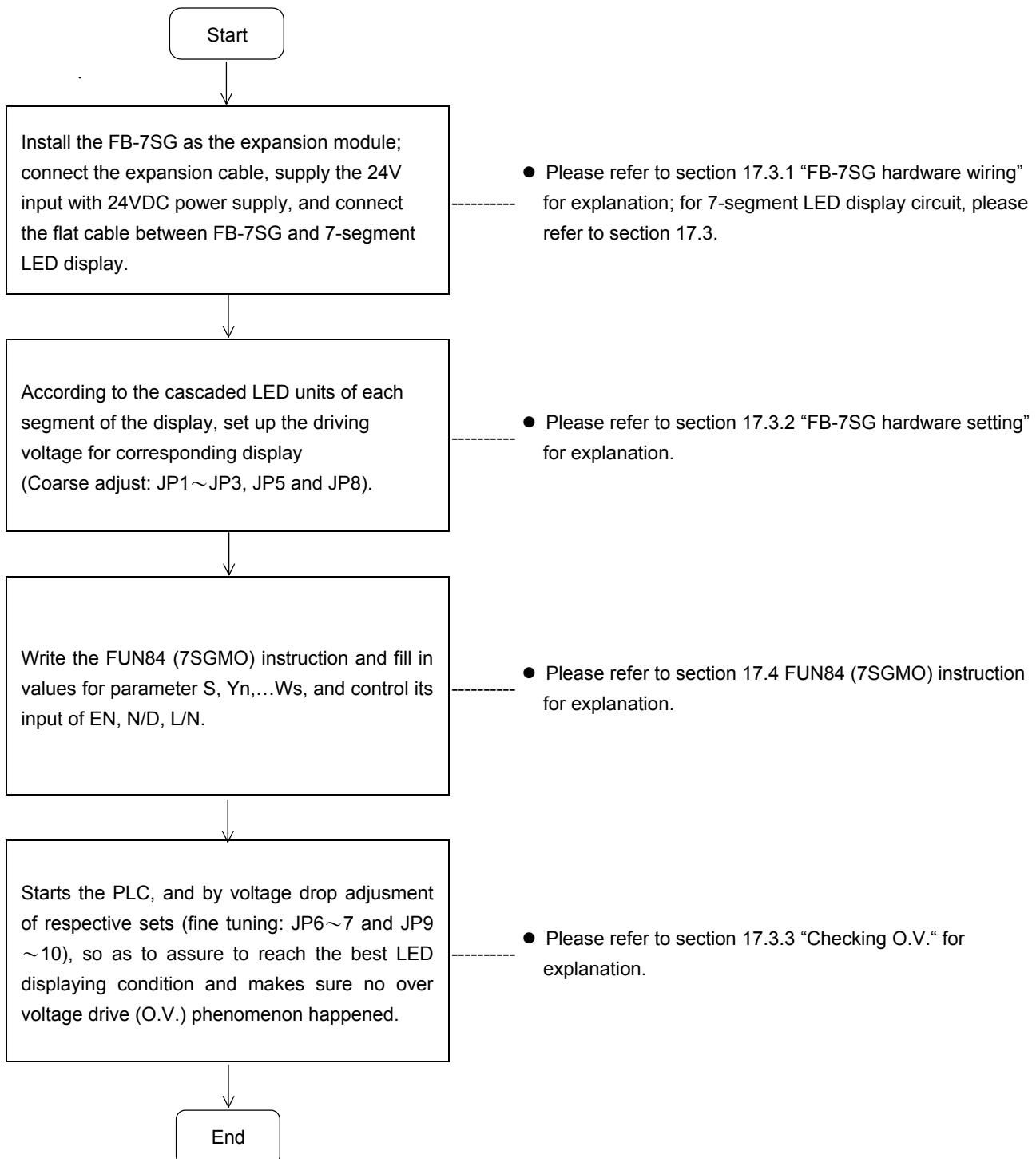
FB-7SG includes 7SG1 and 7SG2 two type of modules, they are equipped with 1 or 2 display IC(s) which can display 8 numerical characters (digits) each. And they can drive 8 or 16 common cathode 7-segment LED display. The following diagram illustrated the example of FB-7SG2.

### Outlook View



The FB-7SG is equipped with dedicated 7-segment LED display IC, which performs multiplexing scan for 1~8 digits of 7-segment LED display. The user merely needs to connect with a 16 pins flat cable, and can attain to 8 digits of numerical characters displaying or 64 points of independent LED lamp displaying (one digit of displaying can form to 8 independent display, or it can select the mixture of digits and independent points of displaying). Every set of display occupies 16 point of expansion output points. Since the maximum expansion output are 248 points (while connecting to CPU unit with 20 I/O points), it can control up to 15 sets of display, i.e. 120 digits or 960 independent points displaying.

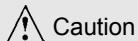
## 17.2 Procedure for FB-7SG 7-segment LED Display Module Usage



## 17.3 Hardware Connection and Configuration for FB-7SG

### 17.3.1 FB-7SG Hardware Wiring Layout

The hardware connection of FB-7SG is illustrated as the diagram above. In addition to the basic wiring of 24VDC input supplied by external power supply, expansion input and the expansion output, the output of FB-7SG needs merely 16-pin flat cable with IDC connector to connect to 7-segment LED display board and it will work.

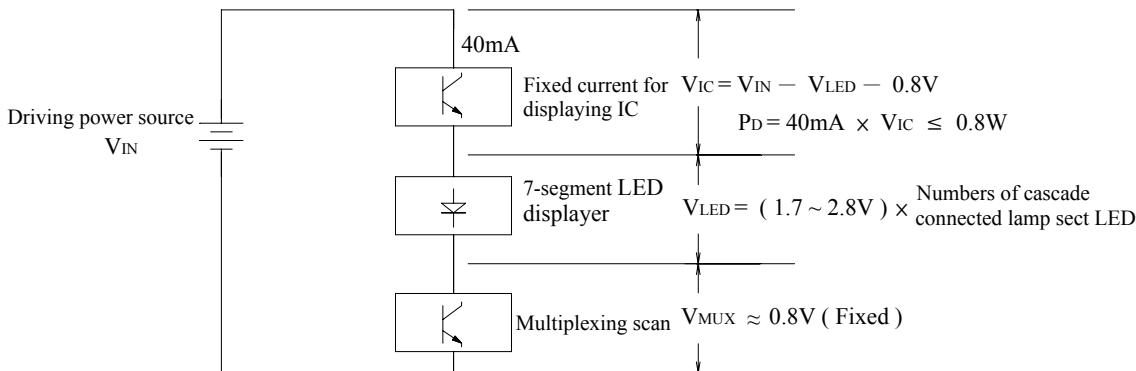


Caution

Because the connectors of expansion input and expansion output as well as respective sets of LED display output's are all adopting the same kind of 16-pin IDC connectors, they must be correctly inserted as the diagram illustrated without confusion, so as to make the FB-7SG normally work; otherwise, it will disable the system working or even burn out the FB-7SG module or 7-segment LED display.

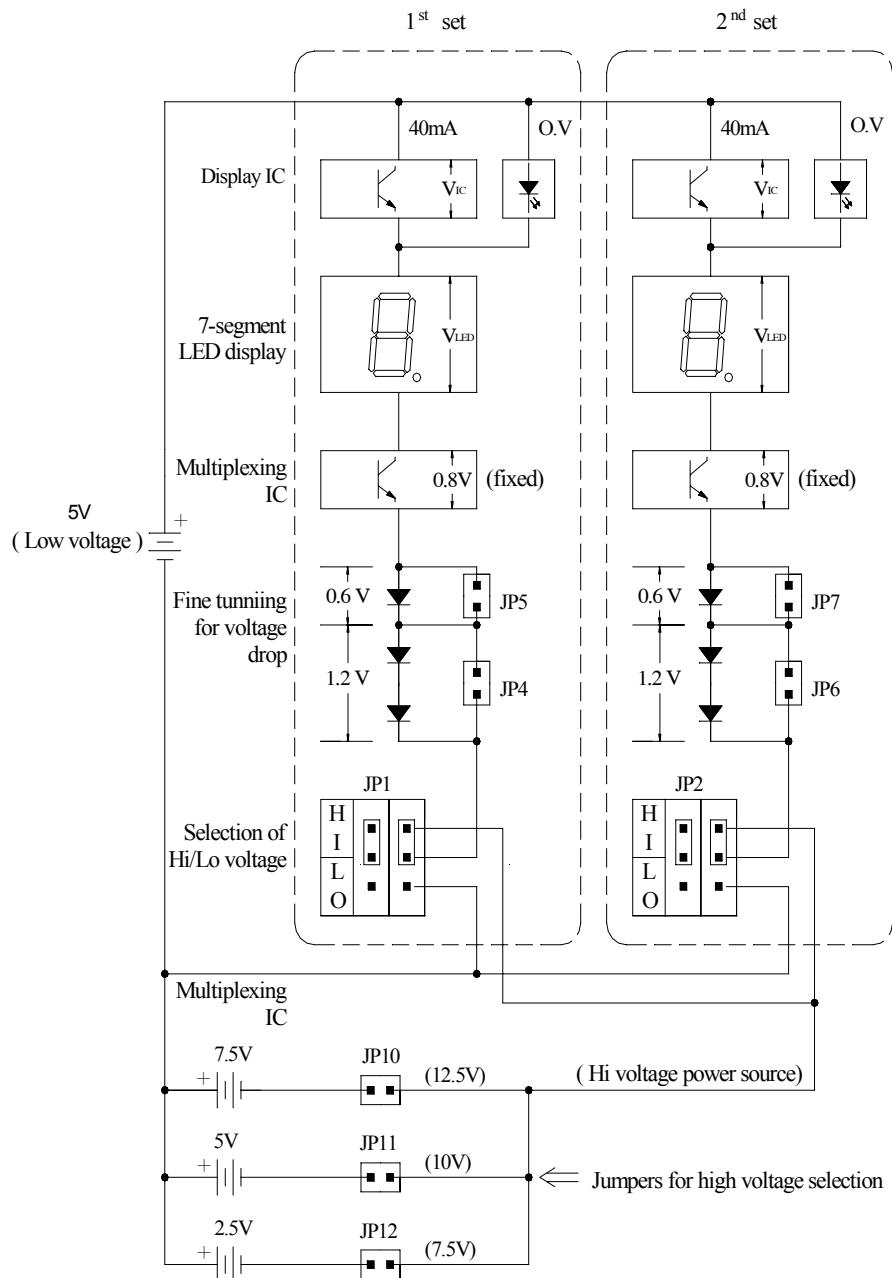
### 17.3.2 The Hardware Setting of FB-7SG

The following diagram shows the output driving circuit of FB-7SG internal display IC:



The display IC consumes 40mA constant current, therefore its power dissipation depends utterly on the voltage drop of  $V_{IC}$  ( $P_D = 40mA \times V_{IC}$ ). As shown in the above mentioned  $V_{IC} = V_{IN} - V_{LED} - 0.8V$ , the  $V_{IC}$  is influenced by driving power source voltage  $V_{IN}$  and 7-segment segment voltage drop  $V_{LED}$  because the safety power dissipation must be confined under 0.8W in the inferior temperature circumstance condition, i.e. the  $V_{IC}$  must be smaller than 2V. If the  $V_{IC}$  is too low, it will cause the display insufficient in brightness or can't even been displayed; if the  $V_{IC}$  is too high, it will cause the display incorrect (those which shouldn't light up are also lighting), or even damaged the display IC.

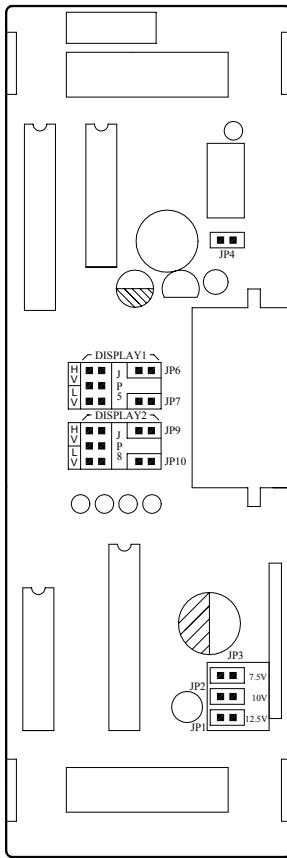
The voltage drop of LED usually falls between 1.7V~2.8V; nevertheless, the 7\_segment LED display's respective section (or called as respective segment, such as a~g) are usually formed with 1~5 piece of cascaded LED. Therefore, their voltage drop of respective segment could have a large difference ranging from 1.7V~14V, and to drive with unitary voltage for various LED display seems to be impossible. For the convenience of driving most of 7-segment LED displays, the FB-7SG provides 5V (low voltage), 7.5V, 10V, and 12.5V (these are categorized as high voltage) four driving voltages, and cascaded with diode and Jumper to make 0.6~1.8V ( $V_{IC}$ ) range of voltage fine tuning. It not only could drive various kinds of voltage drop LEDs, but also to assure the  $V_{IC}$  not to burn out the display IC for exceeding 2V. The following diagram shows the output circuit for LED display of FB-7SG, high/low voltage setting (shared in using), and choice of high/low voltage driving for respective set of display and jumpers for voltage drop fine tuning, as well as its diagram for actual location setting layout. The hardware settings mentioned here are base on the pertaining settings of driving voltage  $V_{IN}$ , choice of high/low voltage driving, and fine tuning of voltage drop to make the 7-segment LED display to reach the best illumination displaying, without burn out the display IC nor shorten their usage durations.



LED driving circuit of FB-7SG

Caution

1. The jumpers (JP5 and JP8) for choice of High/low voltage driving must be both vertically inserted to the HI side or both to LO side; if one to be HI and another LO, or horizontally placed, it will cause the abnormal working or damaging the display IC.
2. The jumpers (JP1~JP3) of high voltage setting can only set to be one jumper short (ON) at any time; otherwise, it will cause the power circuit short that disables its working or burns out the circuit.



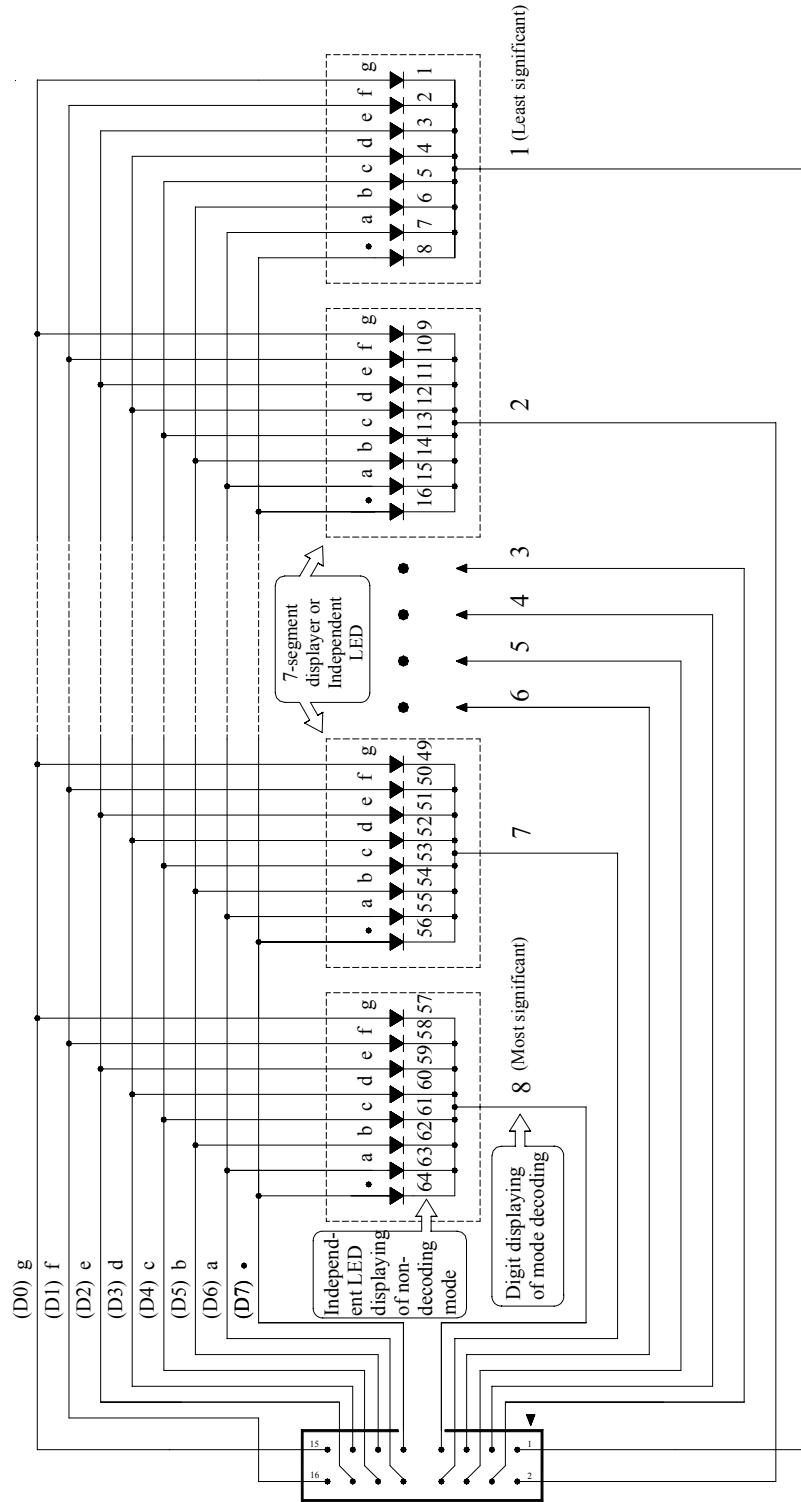
Layout of jumpers location

### 17.3.3 Checking the Over Voltage (O.V.) of FB-7SG

As above mentioned, the FB-7SG must be based on the above mentioned jumpers setting to place Vic below 2V, after choosing the 7-segment LED display. Nonetheless, it's difficult for user to measure the Vic under multiplexing scan. Therefore, the FB-7SG has been designed to have the over voltage (O.V.) indicator for user to check whether there is the over voltage driving happening. The O.V. indicator is easy to find on the front pannel of FB-7SG module.

To reach the outcome of O.V. displaying, it must first turn on all of the LED segments (Including fraction points, Totally 64 points) of the tested set of 7-segment LED display to make sense. In this case, if the O.V. indicator distinguished, it means that there doesn't exist the O.V. condition; if the O.V. indicator turns on, it means the O.V. happened (if not all the tested LED segments been turned on, the O.V. indicator may become flashing or lighted all the time, then the O.V. indicator does not make any sense). The easiest way to turn on all LED segments for O.V. test is that by forced both "N/D" and "L/N" inputs of FUN84(7SGMO) to "1" (please refer to P17-10), in addition to that the user can write the ladder program to turn on all LED segments then do O.V. check and adjustment. On the other hand, while using the FB-7SG module, before the FUN84 (7SGMO) is written into the ladder program for display control, start (let PLC be "RUN") the PLC first, all of the LED segments and friction points of the 7-segment LED display, which connected to FB-7SG module, will be all turned on. With this characteristic, the user can also view that whether the respective LED segments of display are working normally and to do O. V. check and adjustment as well.

#### 17.4 The Detailed Layout for 7-segment LED Display and Independent LED Display Circuit



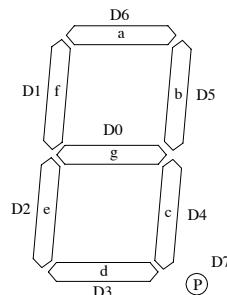
The circuit above is the layout of wire connection (common cathode) for FB-7SG module and 7-segment LED displays or independent LED displays; user may make the display by himself according to this wiring layout, and connect to any display output of FB-7SG with 16 pin flat cable. For convenient in using, the FATEK provides 4 kinds of size of 7-segment display board or final product which the user may select to use. The following are the specifications:

Model number	Size and characters	The dividable characters of display board
DB.56 (DB.56LED)	0.56 inch ×8	8 characters ×1 or 4 characters ×2
DB.8 (DB.8LED)	0.8 inch ×8	8 characters ×1 or 4 characters ×2
DB2.3 (DB2.3LED)	2.3 inch ×8	It may base on a character for division, and to assemble from 1~8 characters.
DB4.0 (DB4.0LED)	4.0 inch ×4	

※ The model number in the parenthesis represents the final product of 7-segment display.

## 17.5 Decode and Non-decode Display

The 7-segment display has 7 LED segments (a~g) and another fraction point for displaying as following illustrated figure. Its main purpose is for the display of number 0~9, there are two ways of displaying method for the control of respective LED segments.



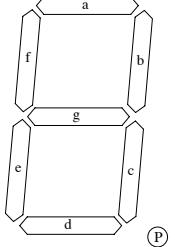
① **Non-decode display:** (To turn on or turn off respective LED segments are independently controlled by the user's application program).

This means that the displaying method of independent control over LED segments must be controlled by the user's application program according to the number to be displayed, e.g. "3", and make the corresponding segments light up (i.e. D0, D3, D4, D5, D6 turns on, and D1, D2, D7 turns off). Every segment must have one corresponding bit to control. For each numeric character display, the user must use 8 bits (D0~D7) controlling the light up or distinguish of 7 segments include fraction point. In application, the user must first configure the corresponding segments to number of 0~9 for their luminiferous on/off table (i.e. the "encoding" of number to be displayed), and then based on number to be displayed to transmit the segments controlling signal to 7-segment display for displaying. Since it controls 8 bits indirectly, usually we don't recommend to use it for normal numeric character displaying; it's mainly used for independent indication displaying. The respective segments of 7-segment LED and their corresponding control bits D0~D7 are illustrated as the figure above. If it needs to display numeric character with non-decoding, it may employ FUN59 ( $\rightarrow$ 7SG) instruction to decode and convert the non displaying nibble value, thus it may save the complicated encoding works done by user.

Note: When a set of display needs to partially display with numeric character, and partially with independent LED displaying, it must employ the non-decoding displaying method. The independent LED can be correspondingly controlled one after another, and numeric character displaying adopting FUN59 ( $\rightarrow$ 7SG) instruction to help to convert the numeric character to displaying character form and it can be easily attained. Please refer to example 3 of section 17.8.

- ② **Decode display:** Directly express the nibble of BCD value and by the following list of default encoding form to display the corresponding numeric digits.

Because the numeric character 0~9 could be expressed with 4 bits (called as Nibble) of BCD value, the so called "decode display" is using the hardware circuit to perform the conversion of the above mentioned numerical character 0~9 BCD code to a~g segment signals, and transmit to the 7-segment display for displaying. Since the decoding job is completed by the circuit, the user needs merely to pass the nibble of BCD code to the display module, and the 7-segment display will be able to correctly display the numeric form of 0~9. The display model is simple and convenient, but it has only 16 kind of characters for choice (as the following list of numeric forms for display), and it can't independently control the respective segments as freely as the non-decode ones. Because the nibble can express 16 kind of messages, in addition to the BCD code of 0~9, it can still display another 6 text forms. The following list is the decoding of nibble value to be displayed in numeric form for FB-7SG module.

Nibble value		Structure of 7-segment display	Segment off (0) on (1)							Displaying of numeric form
Hex decimal	Binary		a	b	c	d	e	f	g	
0	0000	 The diagram shows a 7-segment display with segments labeled a through g. Segment 'a' is the top horizontal bar. Segments 'b' and 'f' are the vertical bars on the left and right respectively. Segment 'c' is the bottom horizontal bar. Segments 'd' and 'e' are the two vertical bars in the center. Segment 'g' is the middle horizontal bar.	1	1	1	1	1	1	0	0
1	0001		0	1	1	0	0	0	0	1
2	0010		1	1	0	1	1	0	1	2
3	0011		1	1	1	1	0	0	1	3
4	0100		0	1	1	0	0	1	1	4
5	0101		1	0	1	1	0	1	1	5
6	0110		1	0	1	1	1	1	1	6
7	0111		1	1	1	0	0	1	0	7
8	1000		1	1	1	1	1	1	1	8
9	1001		1	1	1	1	0	1	1	9
A	1010		0	0	0	0	0	0	1	-
B	1011		1	0	0	1	1	1	1	E
C	1100		0	1	1	0	1	1	1	H
D	1101		0	0	0	1	1	1	0	L
E	1110		1	0	0	1	1	1	1	P
F	1111		0	0	0	0	0	0	0	

## 17.6 The Power Input Specifications and Power Dissipation of FB-7SG

The FB-7SG built in an isolated power supply driven by external 24VDC input, which supplies the power for FB-7SG internal circuit and 7-segment LED display. The accommodated input of voltage range is DC 24V±20%.

The static power dissipation of FB-7SG module itself is 2Wmax; the dynamic power dissipation is getting greater along with the increasing of the driving of 7-segment display. Since the driving current of respective segment for any set of FB-7SG displaying IC is 40mA, one numeric digit consists of 8 segments is fixed at 320mA, and maximum power dissipation of any set can be calculated by following equation:

$$P_d = 320\text{mA} \times V_{IN} \text{ (LED driving voltage)} \div 0.8 \text{ (power efficiency) W}$$

$$\text{Total dissipation} = 2 + P_d \times n \text{ (W)}$$

For example, the FB-7SG2 (2 sets of output) in the maximum power dissipation ( $V_{IN} = 12.5V$ , 8 digits of segments are all at the brightest condition), the total power dissipation will be

$$2W + (320\text{mA} \times 12.5V \div 8) = 7W$$

## 17.7 Explanation of FUN84: 7SGMO , which is the Convenient Instruction for FB-7SG

The following pages are the instruction explanation of FUN84 (7SGMO).

## 7SGMO Instruction Explanation

FUN 84 7SGMO	Convenient instruction proper to FB-7SG Module	FUN 84 7SGMO
-----------------	--	-----------------

84.7SGMO

Execution control—EN	S : <input type="text"/>	S : Starting address of registers to be displayed.
Non-decode / Decode—N/D	Yn : <input type="text"/>	Yn: Starting address of displaying output of this module.
Leading zero / Nonleading zero—L/N	Dn : <input type="text"/>	Dn: Number of characters (digits) to be displayed.
	Pt : <input type="text"/>	Pt : Designation of fraction point.
	IT : <input type="text"/>	IT : Designation of brightness level.
	WS : <input type="text"/>	Ws: Working register for the operation of this instruction.

Range Oper- and	Y Y240	WX WX240	WY WY240	WM WM1896	WS WS984	TMR T255	CTR C255	HR R3839	IR R3903	OR R3968	SR R4167	ROR R8071	DR D3071	K
	Y0   Y240	WX0   WX240	WY0   WY240	WM0   WM1896	WS0   WS984	T0   T255	C0   C255	R0   R3840	R3840   R3903	R3904   R3968	R3968   R4167	R5000   R8071	D0   D3071	16/32 bits positive number
S		○	○	○	○	○	○	○	○	○	○	○	○	○
Yn	○													
Dn		○	○	○	○	○	○	○	○	○	○	○	○	1-8
Pt		○	○	○	○	○	○	○	○	○	○	○	○	0-FFH
It		○	○	○	○	○	○	○	○	○	○	○	○	1-16
Ws			○	○	○	○	○	○		○*	○*	○*	○	

### Input Control

EN: Execution control; =1, update the displaying; =0, not to update the displaying.

N/D: Decode/Non\_decode selection

L/N: Leading zero/Non-leading zero selection

These two selections have the compound function as follows

N/D	L/N	Displaying mode	※Refer to section 17.5 for explanation of decode/non-decode displaying	
0	0	Decode, Non-leading zero	※Suppose S is 8-digit of numeric characters, its value is 123.	
0	1	Decode, Leading zero.	Non-leading zero displaying	: 123
1	0	Non-decode	Leading zero displaying	: 00000123
1	1	Test mode	(Leading zero has meaning only when in decode displaying)	

### Explanation of Operand

S: Starting address of the registers whose contents will be displayed; it's the least significant digit for display. Each nibble will be displayed at one digit while in decoding mode, it needs 2 registers S and S+1 to display a set of 8 digits. In non-decoding mode, it needs S~S+3 totally 4 registers to display a set of 8-digit. In decoding displaying, fills the BCD value (2 words, 8×4bits) to be displayed into S~S+1, and this instruction will display the 8-digit numeric number. While in non-decoding displaying, the user must fill the 8-digit with 64 bits of on/off signals into S~S+3 (totally 4 registers), then, it is able to display the value of that 8-digit.

FUN 84 7SGMO	Convenient instruction proper to FB-7SG Module	FUN 84 7SGMO
-----------------	--	-----------------

Yn: FB-7SG is the expansion module, the physical hardware output address of it is the summation of the outputs of CPU board and the outputs of expansion modules before it. For example, there are FBx-28MX CPU board (with 12 outputs) and FB-40EA expansion module (with 16 outputs) before it, consequently the starting output address of the FB-7SG module is  $Y_{(12+16)} = Y_{28}$ . The value of Yn must be consistent to the physical output address of FB-7SG to work correctly, and each set (8-digit) of display needs successive 16 outputs beginning from Yn to control. Therefore the Yn of the second set of 7-segment display will be the Yn of first set of display plus 16.

Each set (8-digit) of 7-segment display needs independent FUN84 instruction to control, it means FB-7SG2 needs 2 FUN84 instructions to work.

Dn: Valid digits to show for a set of 7-segment display (8-digit). While it didn't need to display all of the digits, it may use the displaying digits assigned by Dn. Dn can be set from 1~8. When Dn is set to be smaller, the scanning of displaying will be increased hence the brightness level will be getting higher.

Pt: Specify the bit of a word data to designate the position of fraction point; only low byte (B7~B0) is used to assign the fraction point of that digit to be lighten up. e.g. Pt = 0001H, it means to light up the fraction point of the least significant digit (1st digit), and 0081H means to light up the fraction points of the most significant digit (8th digit) and the first digit.

IT: Level of the brightness; it could be 1~16, the brightness is proportional to the setting value.

Ws: The working register for operation of this instruction, which controls one word. It can't repeat in using by other program.

#### Instruction Explanation

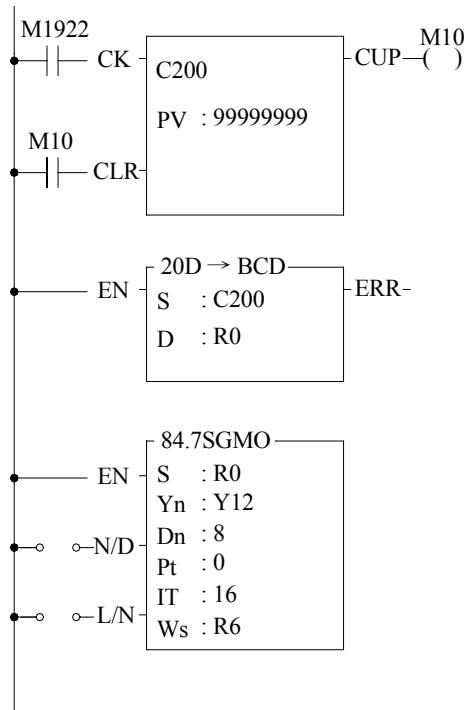
- When execution control "EN" =1, this instruction will update the displaying output with the contents of the specified registers.  
When "EN" =0, it will not update the displaying, and the display will stay as before.
- Each FUN84 instruction can display up to 8 digits at the most; for digit exceeding 8 digits,it must employ another FUN84 instruction, it means FB-7SG2 needs 2nd FUN84 instruction to work.
- For the selection of Non-decode/Decode ("N/D") and the selection of Leading zero/Non\_leading zero ("L/N"), please refer to previous page of displaying mode list for explanation.

## 17.8 Program Example

The following example1~example3 are using the same hardware (FBE-28MC+FB-7SG1), and it performs the decode (numeric character display), non-decode (independent LED display), and the mixture of mentioned above to achieve the 7-segment or independent LED displaying.

### Example 1: Numeric character display by "decode" mode

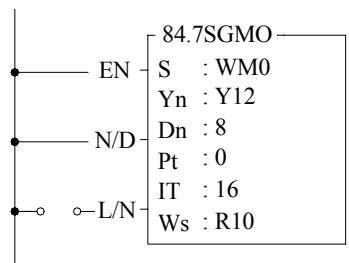
Using counter C200 to count the internal 1 second clock of PLC , and converts the current value of C200 to be BCD code for FB-7SG1 to show.



- Increasing C200 per second up to value of 99999999 and then return to 0→1→2.... cycling counting.
- Converts the current value of C200 (binary) to be decimal format (BCD) and stores it into DR0 for displaying.
- Select "decode" mode to show the 8 digits BCD code of DR0 on 7-segment display, and not to light up the fraction point; brightness level is the most.

### Example 2: Independent LED display by "Non\_decode" mode

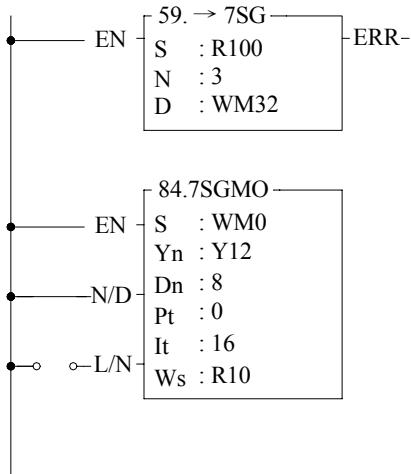
To turn on or turn off the 64 independent LEDs (number 1~64) showing in the circuit on section 17.4 through FB-7SG, we have to select the "Non-decode" mode to work, and with internal relay M0~M63 for corresponding mapping.



- LED1~LED64 will indicate the on/off status of M0~M63.  
LED1 shows the status of M0  
...  
LED64 shows the status of M63

### Example 3: Mixture display of numeric character and independent LEDs by "Non\_decode" mode

Replacing the LED33~LED64 of example 2 by 7-segment LED display to show the content of DR100; and LED1~LED32 remain to indicate the status of internal relay M0~M31.



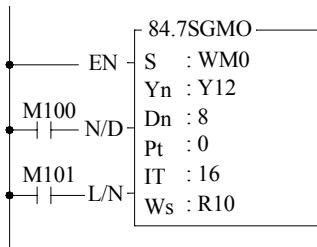
- Converts the current BCD value of R100 (N=3,it means 4 digits that Nibble0~Nibble3 ) to be 7-segment display patterns and stores them into DWM32 (M32~M63) for 4-digit of 7-segment display to show.
  - Select "Non-decode" mode to display
  - LED1~LED64 will indicate the on/off status of M0~M63,where LED1 shows the status of M0  
...  
LED32 shows the status of M31
- LED33~LED64 were replaced by 4-digit of 7-segment display and showing the content of DR100.

- ※ When there are independent LEDs to display, it must employ "Non-decode" mode. While in "Non-decode" displaying, it can employ FUN59 converting BCD code to 7-segment displaying pattern, so as to easily display the numeric character.

### Example 4: Over Voltage Checking and Lamp Test

The handy instruction FUN84 supports the "Lamp Test" function while both control inputs "N/D" and "L/N" of this instruction are all ON, there will turn on all segments of the 7-segment display include fraction points if this instruction being executed. At the same time, the user may check the O.V. indicator for over voltage adjustment. If the O.V. indicator distinguished, it means that there doesn't exist the O.V. condition, it is required; if the O.V. indicator turns on, it means the O.V. happened, it is necessary to set the jumpers mentioned in 17.3.2 for normal operation.

\* The O.V. indicator may become flashing or lighted all the time if not all the LED segments been turned on, in this case the O.V. indicator does not make any sense.



- Disable and force on M100 and M101 to perform the lamp test and over voltage adjustment.

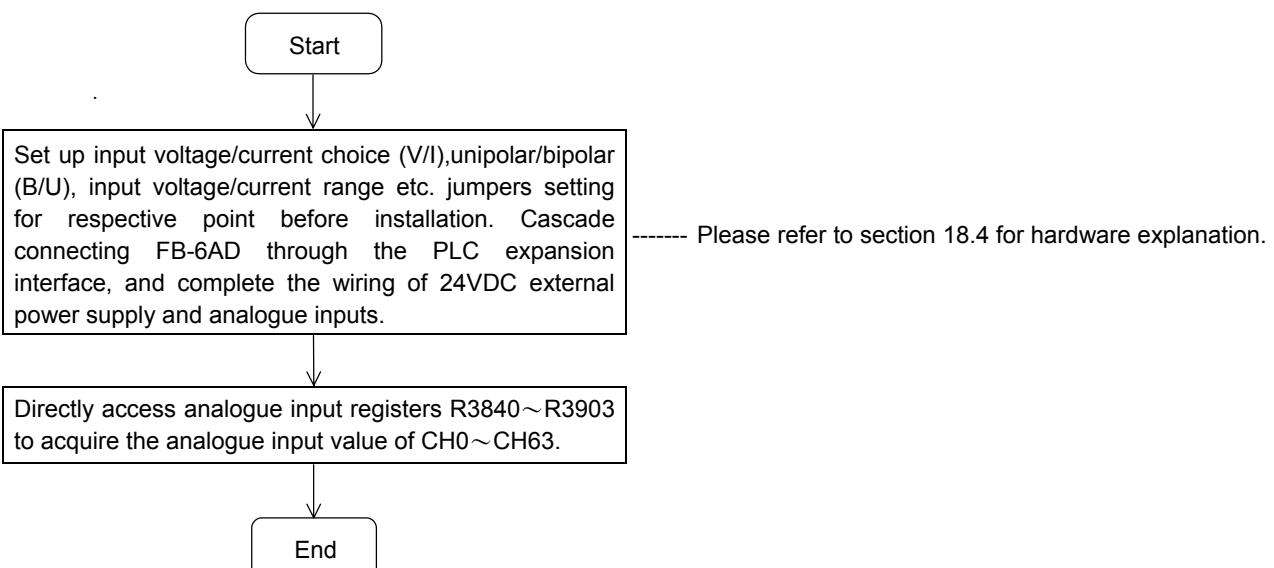
## Chapter 18 FB-6AD Analog Input Module

The resolution of FB-PLC analogue input (or called as A/D input) is 12 bits. The OS version of main unit before V3.2x has only 8 points of analogue input for FB-PLC (which goes together with old A/D module of FB-8AD). Starting from OS version V3.30, the analogue input can reach as many as 64 points, and its module changes to FB-6AD with new model of slim shape. Each FB-6AD has 6 points of input; therefore, it can expand upto 11 FB-6AD input modules with 64 points of analogue input in total (the last two points of the 11th module are invalid).

### 18.1 Specifications of FB-6AD Functions

Item	Specifications				Remark
Input points	6 points (Channels)				
Digital input value	-2048 ~ +2047				
Span of analog input	Bipolar*	10V*	1*.Voltage: -10 ~ 10V	5. Current: -20 ~ 20mA	<ul style="list-style-type: none"><li>• There are 8 kinds of input in total, user may set by himself.</li><li>* : It means the default setting.</li></ul>
		5V	2.Voltage: -5 ~ 5V	6. Current: -10 ~ 10mA	
	Unipolar	10V	3.Voltage: 0 ~ 10V	7. Current: 0 ~ 20mA	
		5V	4.Voltage: 0 ~ 5V	8. Current: 0 ~ 10mA	
Finest resolution	Voltage: 1.22mV (when input set to 0 ~ 5V) Current: 2.44µA (when input set to 0 ~ 10mA)				=Analogue input signal/4096
Accuracy	Within ±1% of full scale				
Conversion rate	Update the A/D readings every scan				
Maximum absolute input signal	Voltage: ±15V (max) Current: ±30mA (max)				It may cause the destruction to hardware if exceeds this value.
Input resistance	40KΩ (voltage input), 250Ω (current input)				
Insulation	Photocouple isolation				No isolation between channels
External power supply	24VDC±20%, Current < 200mA/@24VDC				

### 18.2 The Procedure of Using FB-6AD Analogue Input Module



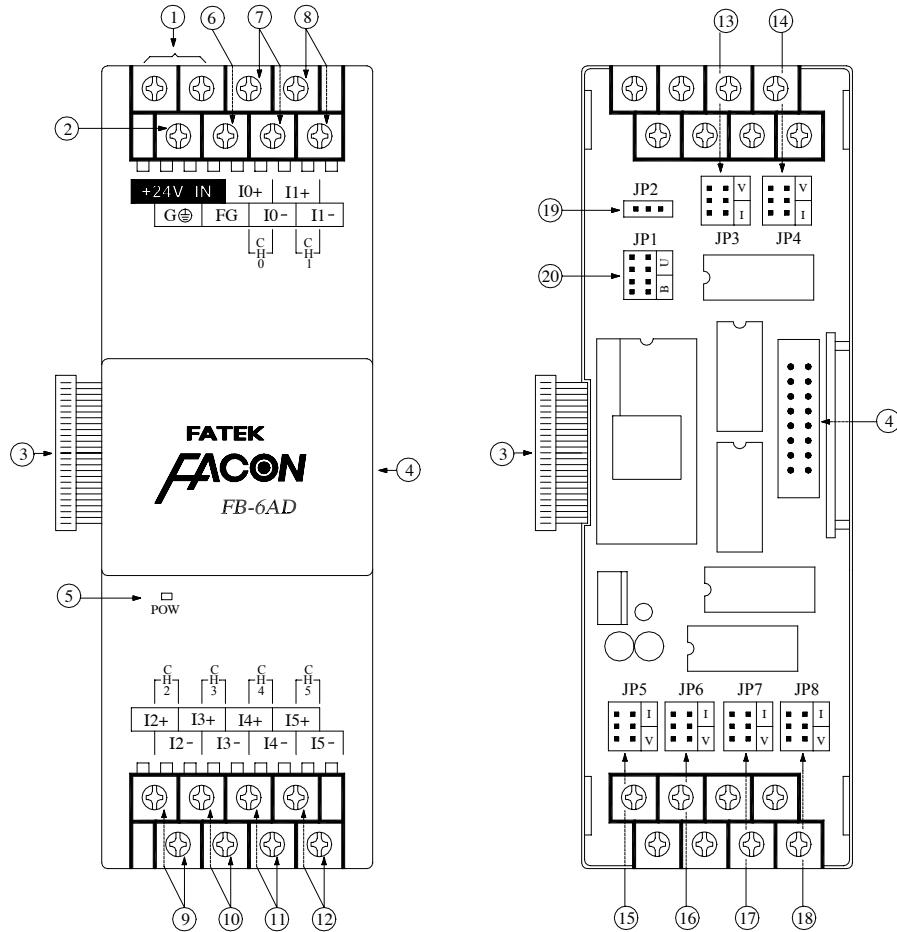
### 18.3 Address Allocation of FB-PLC Analogue Inputs

The memory mapping of FB-6AD inputs is beginning from the module closest to main unit, it is orderly numbered as CH0~CH5 (1st module), CH6~CH11 (2nd module), CH12~CH17 (3rd module)..... and increased with occurring order number, i.e. for each module, it adds with 6 and is totally 64 inputs from CH0~CH63, and they are corresponding to the respective internal analogue input register of PLC (so called as IR register) R3840~R3903 as listed in following table. As long as there is expanded FB-6AD module connection, the PLC main unit will automatically check to verify the quantity of FB-6AD connected, and store the respective A/D value beginning from CH0 orderly into the IR register R3840~R3903; user just access from R3840~R3903 and can acquire the corresponding input span. For the relationship between accessed value and input signal, please refer to section 18.6.

Analogue input register (IR)	Content of IR								Input label of FB-6AD								
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	
R3840	B11	B11	B11	B11	B11											CH0	
R3841	"															CH1	
R3842	"															CH2	
R3843	"															CH3	
R3844	"															CH4	
R3845	"															CH5	
R3846	"															CH0	
R3847	"															CH1	
R3848	"															CH2	
R3849	"															CH3	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
R3896	"															CH2	
R3897	"															CH3	
R3898	"															CH4	
R3899	"															CH5	
R3900	"															CH0	
R3901	"															CH1	
R3902	"															CH2	
R3903	"															CH3	

(Sign extended of B11)

## 18.4 Explanation of FB-6AD Hardware



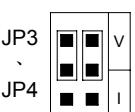
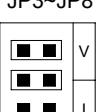
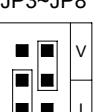
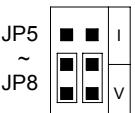
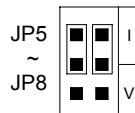
A: Outlook of top view

B: PCB top view (uncovered)

- ① External power input terminal: Power supply of analogue circuit for FB-6AD, the voltage can be  $24VDC \pm 20\%$  and should be supplied with 4W of power at least.
- ② Protecting ground terminal: To connect to the safety Earth Ground of the power system.
- ③ Expansion input cable: It should be connected to the front expansion unit, or the expansion output of main unit.
- ④ Expansion output connector: Provides the connection for next expansion unit.
- ⑤ Power indicator: It indicates whether the power supply at analogue circuit and external input power source are normal.
- ⑥ Framing ground: To connect to the shielding of analogue input, please refer to the wiring connection diagram of next page.
- ⑦ ~ ⑫: Input terminal of CH0 ~ CH5.

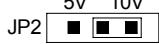
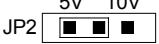
⑯~⑰: Selective jumpers of voltage(V)/current(I) for CH0~CH5.

All of the 6 analogue inputs of FB-6AD can either be voltage input or current input. The voltage or current input is sharing to use the same pair of input terminal ( $I_{in+}$  and  $I_{in-}$ ), and voltage or current is depending on the voltage(V)/current(I) jumpers pair to define (the voltage V is close to terminal side, otherwise is the current I, as shown in the JP3~JP8 of diagram B above). The V/I selective jumpers must be placed according to the text label direction (V, I are both vertically placed) to keep vertical as following diagram illustration; horizontally placed will result in error.

		X
Voltage input (V)	Current input (I)	
JP3 JP4 (CH0~CH1) 	JP3 JP4 (CH0~CH1) 	JP3~JP8  or 
JP5 ~ JP8 (CH2~CH5) 	JP5 ~ JP8 (CH2~CH5) 	Jumper horizontally placed or not placed in pair are both incorrect.

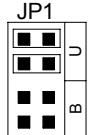
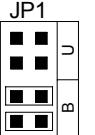
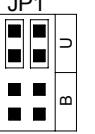
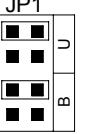
⑯ 5V/10V or 10mA/20mA selection: Maximum input span selection

All Channels must be collectively selected and can't be independently chosen.

Jumper setting		10V/20mA span	5V/10mA span
		JP2 5V 10V 	JP2 5V 10V 
Analogue input	Unipolar (U)	0V~10V 0mA~20mA	0V~5V 0mA~10mA
	Bipolar (B)	-10V~10V -20mA~20mA	-5V~5V -10mA~10mA

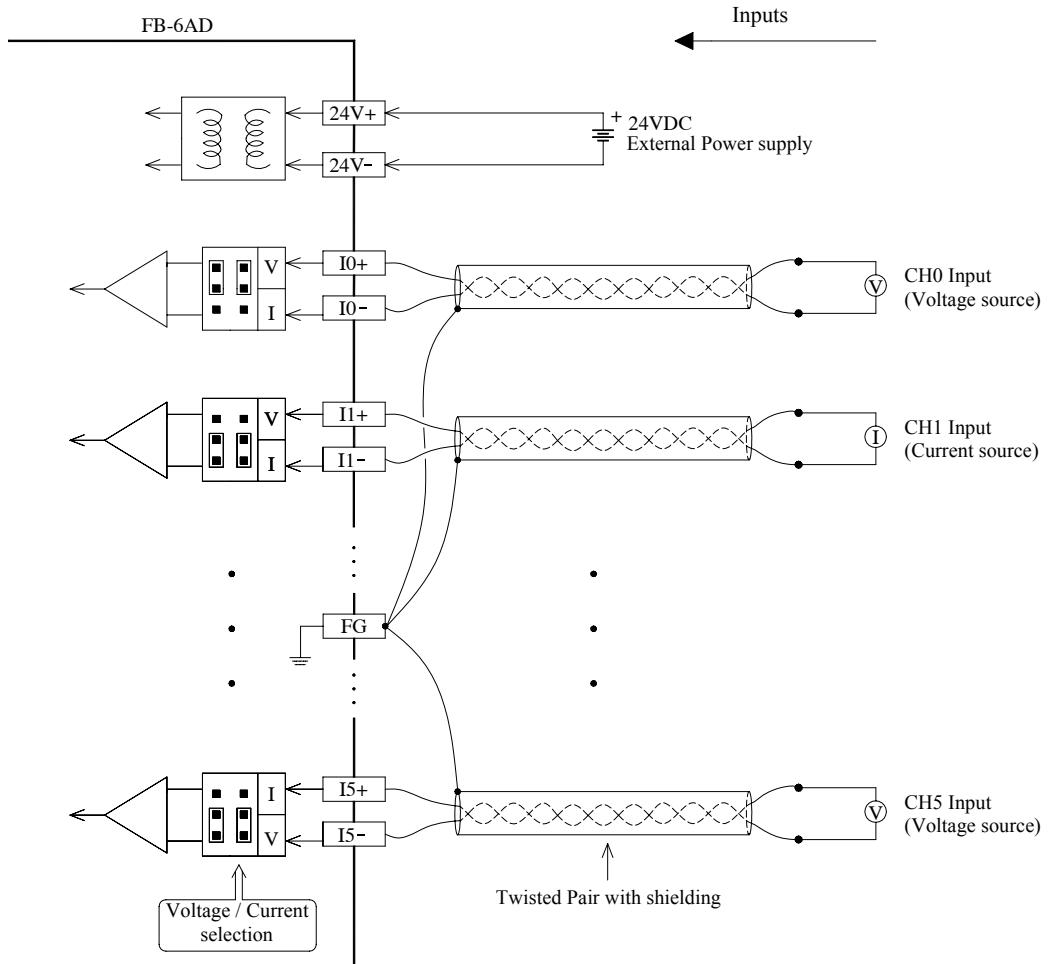
⑯ U/B selection: Unipolar (U) or Bipolar (B) selection

The jumper must according to the U/B text label direction (both B, U are horizontal) to be horizontally placed; it mustn't be vertically placed.

		X
Unipolar (U)	Bipolar (B)	
JP1 	JP1 	JP1  or 

Jumper vertically placed or not inserted in pair are both incorrect

## 18.5 The Input Circuit of FB-6AD



## 18.6 The Input Characteristic and Jumper Setting of FB-6AD

The 8 kind of input range selections of FB-6AD must be based on the settings of V/I, U/B, 5V/10V jumpers to define, that described in previous section. Hereby it will be illustrated with diagram to explain the input conversion characteristics of B/U, 5V/10V jumpers setting (4 kind of selections). These four conversion curves incorporating V/I (voltage/ current) input setting can yield the above mentioned 8 kind of inputs. Please refer to the diagram illustration in section 18.4 for the explanation of V/I selection.

Diagram 1: Bipolar 10V (20mA) Span

Input range	Voltage	-10V~10V
	Current	-20mA~20mA

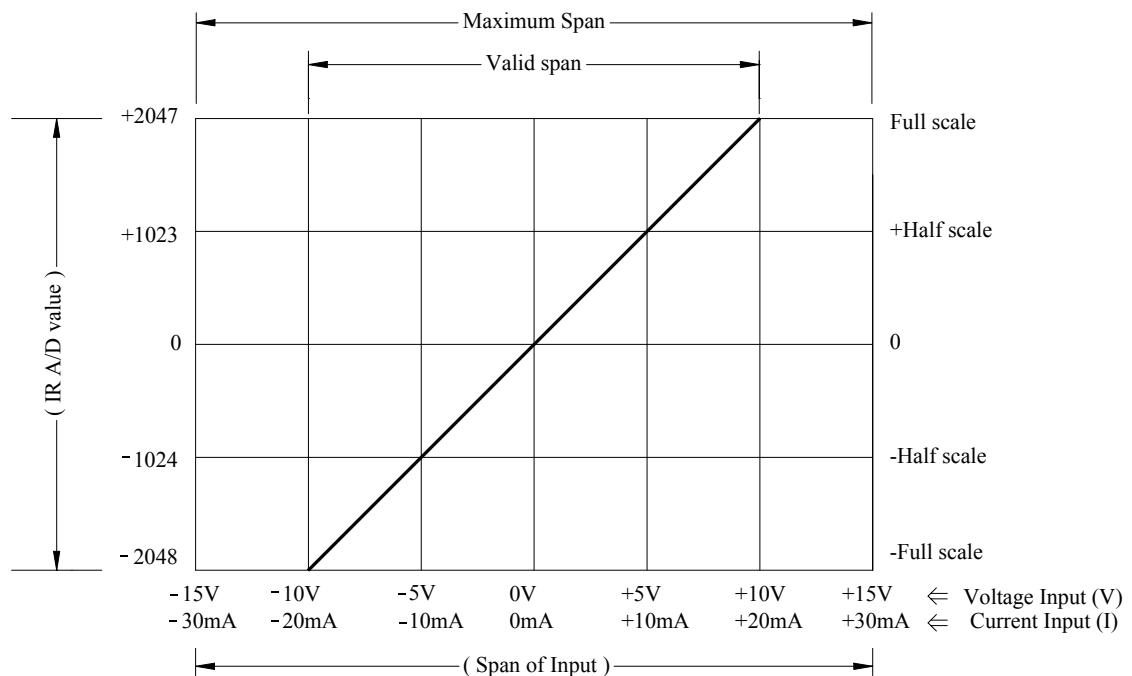
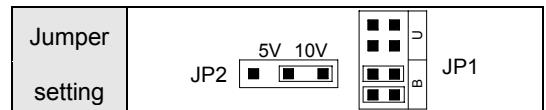


Diagram 2: Bipolar 5V (10mA) Span

Input range	Voltage	-5V~5V
	Current	-10mA~10mA

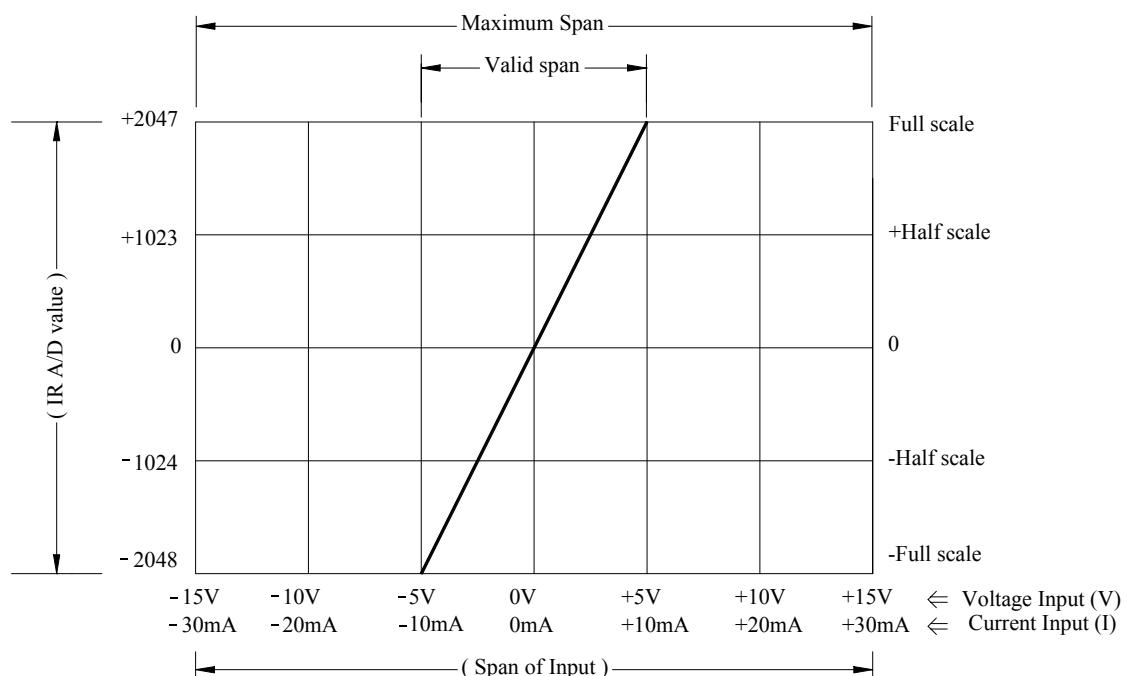
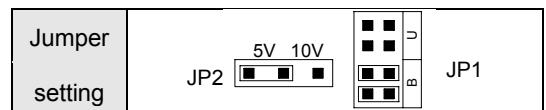


Diagram 3: Unipolar 10V (20mA) Span

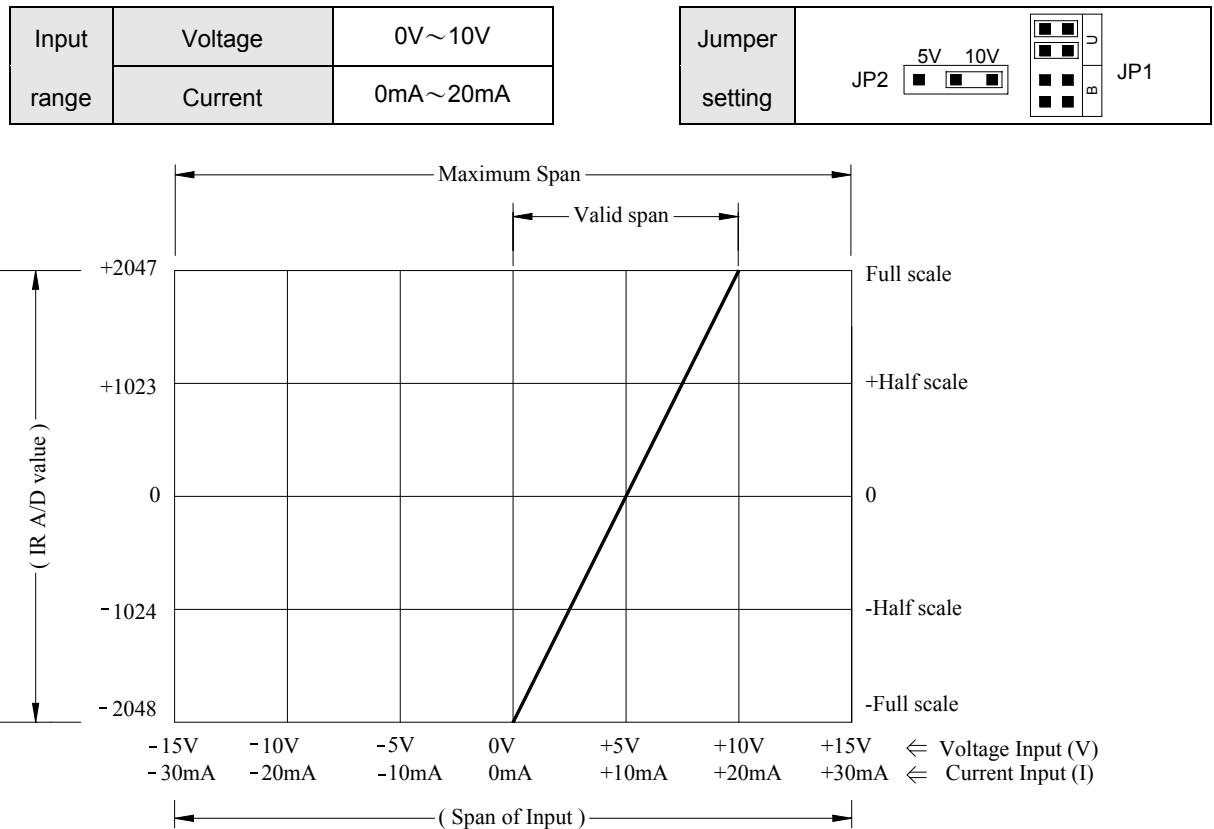
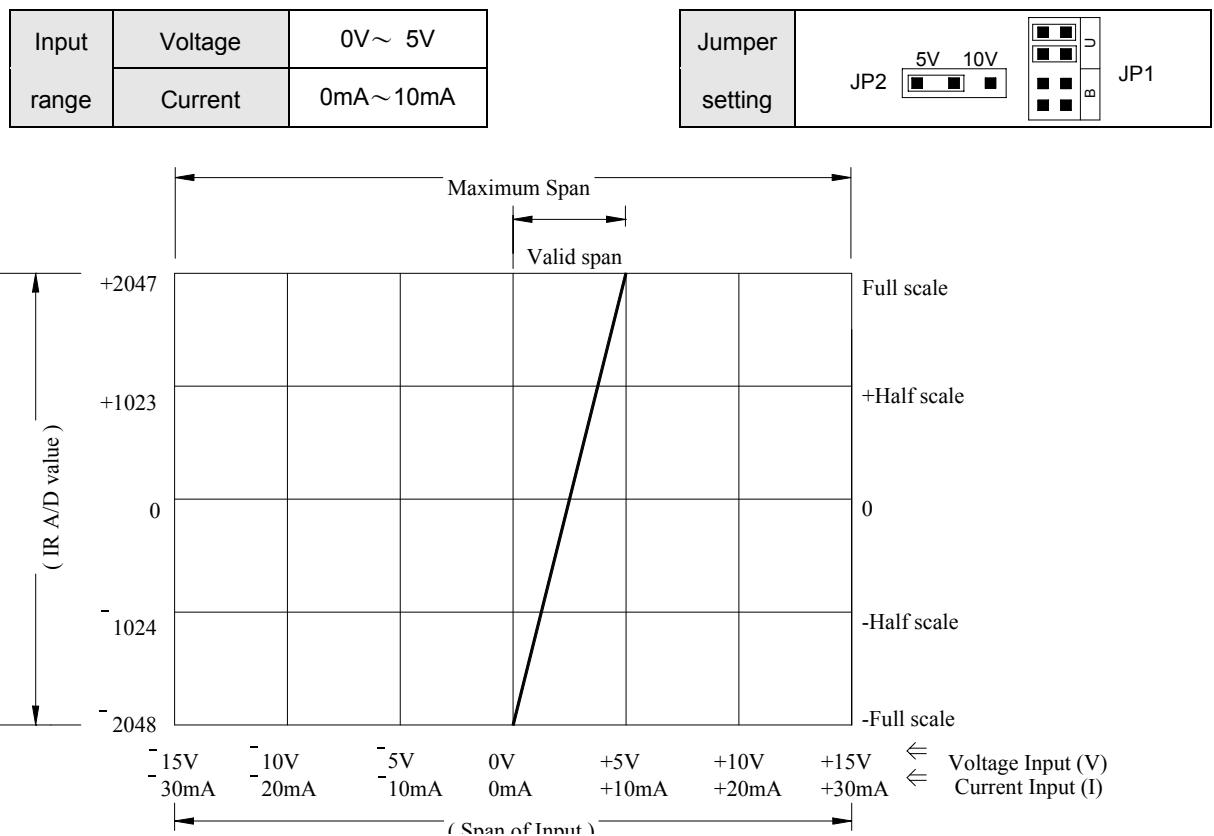


Diagram 4: Unipolar 5V (10mA) Span



## 18.7 Notifications for the operation of FB-6AD

### A Matching with the OS version of Main Unit and FB-6AD

FB-6AD must run on the main unit with OS version later than (include) V3.30 to work normally. If installing FB-6AD to any main unit with version before V3.30, then only the first analog input (CH0) can work normally, all other inputs will not be able to work correctly. Consequently, for main unit with version before V3.30, please use FB-8AD analogue module and can only install with one module with 8 points of analogue input totally.

Note: To tell the version of the main unit, you can just open up the cover at the center of the CPU module and see sticker with

FB-MAC  
V3.\*\*

or  
FB-MU  
V3.\*\*

The “3.xx” is the version of main unit.

### B FB-6AD can not install together with FB-4AJ(K)xx temperature module or FB-8AD analogue input module!

### C The processing for Unipolar Inputs

The minimum value (0V or 0mA) should be 0 for the analogue input of unipolar, and should be 4095 for its maximum input. Nevertheless, the full resolution of 4096 of FB-6AD is expressed with -2048 (minimum) ~ 2047 (maximum), if the user intends to make it become 0~4095, it must be added with a deviation value of 2048 to IR (R3840~R3903) to acquire.

### D Tackling on the OFFSET Mode Input

Confined in the limitation of space, the FB-6AD provides only normal mode for analog inputs. For the process of input for signal source of offset mode (take 4~20mA input for example), the user can set A/D input range to be 0~20mA, convert the IR value to unipolar (0~4095), lessen the offset (4mA) value (4095x4/20=819), then times the maximum input amount (20mA), and divide by the maximum span (4mA~20mA); and it can acquire the offset input conversion from 4mA~20mA reflect to 0~4095, the procedure is as follows:

- a. Set the A/D input range of analogue input module to be 0~20mA.
- b. Add the IR (R3840~R3903) value with 2048 and then store it into register Rn (the value of Rn is 0~4095).
- c. Deduct 819 ( $4095 \times \frac{4}{20}$ ) from value of register Rn, and store the calculated value back to register Rn; if the value is negative, clear the content of register Rn to 0 (the value of Rn is 0~3276).
- d. The value of register Rn times 20 and then divide by 16 ( $Rn \times \frac{20}{16}$ ), and it will convert the 4mA~20mA input to range of 0~4095.
- e. To sum up the items from a~d, the mathematical equation is as follows:  
Offset mode conversion value =  $\left[ IR + 2048 - (4095 \times \frac{4}{20}) \right] \times \frac{20}{16}$ ; the value is 0~4095.

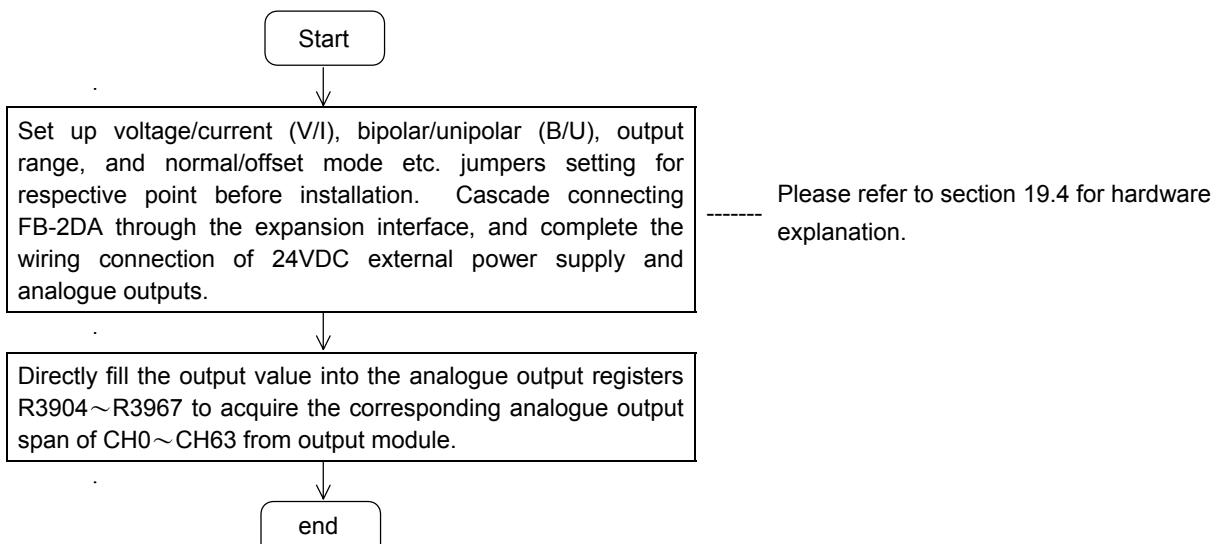
## Chapter 19 FB-2DA Analog Output Module

The resolution of FB-PLC analogue output (or called as D/A output) is 12 bits. The OS version of main unit before V3.2x has only 8 points of total analogue output. Starting from OS version V3.30, the analogue output can reach as many as 64 points, and the output module changes to FB-2DA new style model with slim shape. Each module has two outputs and can expand to connect as many as 32 modules.

### 19.1 Specifications of FB-2DA Functions

Item		Specifications				Remark	
Output point		2 points (channels)					
Digital output value		-2048~+2047					
Kind of analog output signal	Normal Mode	Bipolar*	1*.Voltage: -10~10V 2.Voltage: -5~5V	3.Current: -20~20mA 4.Current: -10~10mA	<ul style="list-style-type: none"> <li>There are 16 kinds of output signal in total, user may set by himself.</li> </ul> <p>* : It means the default setting.</p>		
		Uni-polar	5.Voltage: 0~10V 6.Voltage: 0~5V	7.Current: 0~20mA 8.Current: 0~10mA			
	Offset mode	Bipolar	9.Voltage: -6~10V 10.Voltage: -3~5V	11.Current: -12~20mA 12.Current: -6~10mA			
		Uni-polar	13.Voltage: 2~10V 14.Voltage: 1~5V	15.Current: 4~20mA 16.Current: 2~10mA			
Finest resolution I	Normal mode		Voltage: 1.22mV (while 0~5V output) Current: 2.44μA (while 0~10mA output)				
	Offset mode		Voltage: 0.98mV (while 1~5V output) Current: 1.95μA (while 2~10mA output)				
Accuracy		Within ±1% of full scale					
Conversion rate		Update all outputs every scan					
Maximum accommodation for resistance loading		Voltage: 500Ω ~ 1MΩ Current: 0Ω ~ 500Ω				The deviation will be enlarged if exceeding this range	
Insulation		Photocouple isolation				No isolation between channels.	
External power supply		24VDC±20%, Current < 200mA/@24VDC					

### 19.2 The Procedure of Using FB-2DA Analogue Output Module



### 19.3 Address Allocation of FB-PLC Analogue Outputs

Each FB-2DA module provides 2 point of outputs. The memory mapping of outputs is beginning from the module closest to main unit; it is orderly numbered as CH0~CH1 (1st module), CH2~CH3 (2nd module), CH4~CH5 (3rd module)..... and increased with occurring order number, which reaches 64 points in total (32 modules), and they are corresponding to the respective internal analogue output registers (so called OR register) R3904~R3967. User needs only to expand connecting FB-2DA through expansion interface, and main unit will automatically detect the quantity of the outputs and send out the register value to corresponding output of each FB-2DA. The following table is detailed OR registers (R3904~R3967) corresponding to the expansion analogue outputs (CH0~CH63). The relationship between the register value and output span, please refer to the explanation of section 19.6.

Analogue output register (OR)	Content of OR								Output label of FB-2DA								
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0	
R3904	-----				B11	Output value of CH0				B0						CH0	
R3905	-----					Output value of CH1										CH1	
R3906	-----					Output value of CH2										CH0	
R3907	-----					Output value of CH3										CH1	
R3906	-----					Output value of CH4										CH0	
R3907	-----					Output value of CH5										CH1	
.																.	
.																.	
.																.	
.																.	
R3966	-----					Output value of CH62										CH0	
R3967	-----				B11	Output value of CH63				B0						CH1	

(Sign extended of B11)

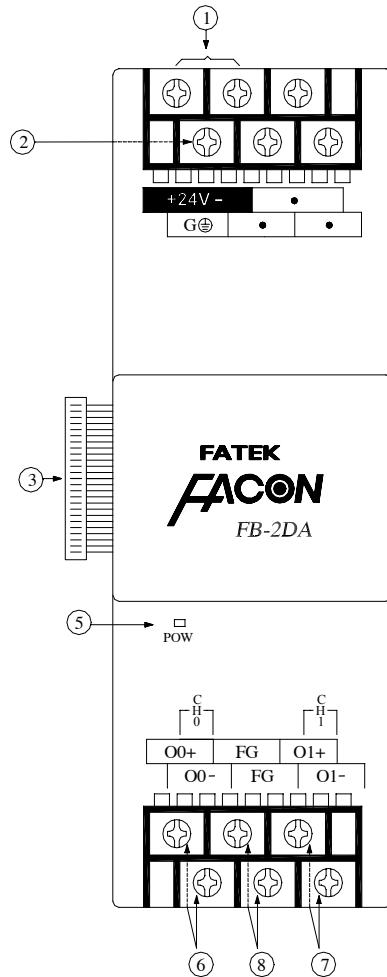
1st module

2nd module

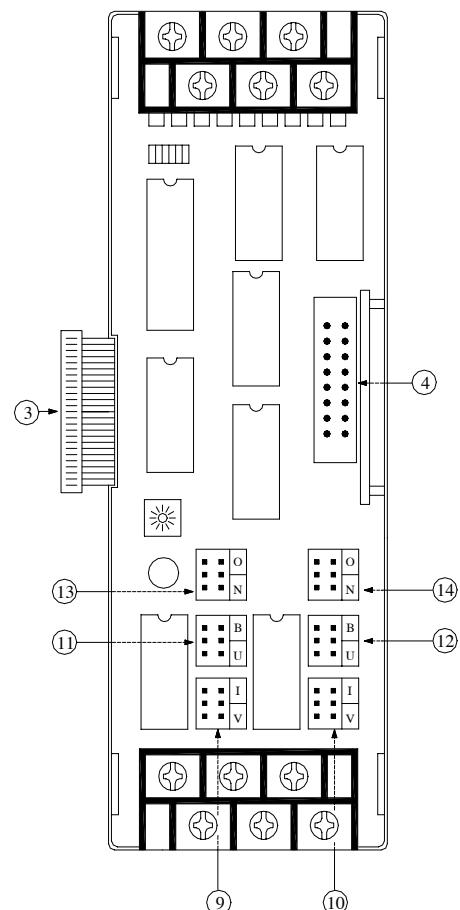
3rd module

32th module

## 19.4 Explanation of FB-2DA Hardware



A: Outlook of top view



B: PCB top view (uncovered)

- ① External power input terminal: Power supply for analogue circuit of FB-2DA module, the voltage can be  $24VDC \pm 20\%$  and should be supplied with 4W of power at least.
- ② Protecting ground terminal: To connect to the safety Earth Ground of power system.
- ③ Expansion input cable: It should be connected to the front expansion unit, or the expansion output of main unit.
- ④ Expansion output connector : Provides the connection for next expansion unit.
- ⑤ Power indicator: It indicates whether the power supply of analogue circuit and external input power source are normal.
- ⑥ 、 ⑦: CH0~CH1 output terminal
- ⑧ Framing Ground.

⑨、⑩: The voltage(V) / current (I) output selection of CH0~CH1

Since the voltage output and current output are sharing to use a pair of terminal, it must depend on the jumper to select voltage output or current output. Both of the jumpers must be placed according to the text label direction (vertically) and in pair to be placed on V or I position, as illustrated in following diagram.

Voltage output (V)	Current output (I)	
JP5、JP6 	JP5、JP6 	JP5、JP6 or JP5、JP6 

Jumper horizontally placed or separated to be up/down placed are all incorrect.

⑪、⑫: The selection of unipolar(U) / bipolar(B) of CH0~CH1.

The two jumpers must be placed as the text label direction (vertical) to insert to B or U position in pair.

Unipolar output (U)	Bipolar output (B)	
JP3、JP4 	JP3、JP4 	JP3、JP4 or JP3、JP4 

Jumper horizontally placed or separated to be up/down placed are all incorrect.

⑬、⑭: The Offset(O) / Normal(N) mode selection of CH0~CH1; the two jumpers must be placed as text label direction (vertical) to insert to O or N position in pair.

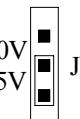
Normal mode (N)	Offset (O)	
JP1、JP2 	JP1、JP2 	JP1、JP2 or JP1、JP2 

Jumper horizontally placed or separated to be up/down placed are all incorrect.

- ⑯ selection of the maximum output span: 5V/10mA or 10V/20mA.

This selection defines the output span both of CH0 and CH1, and labeled with only 5V/10V character.

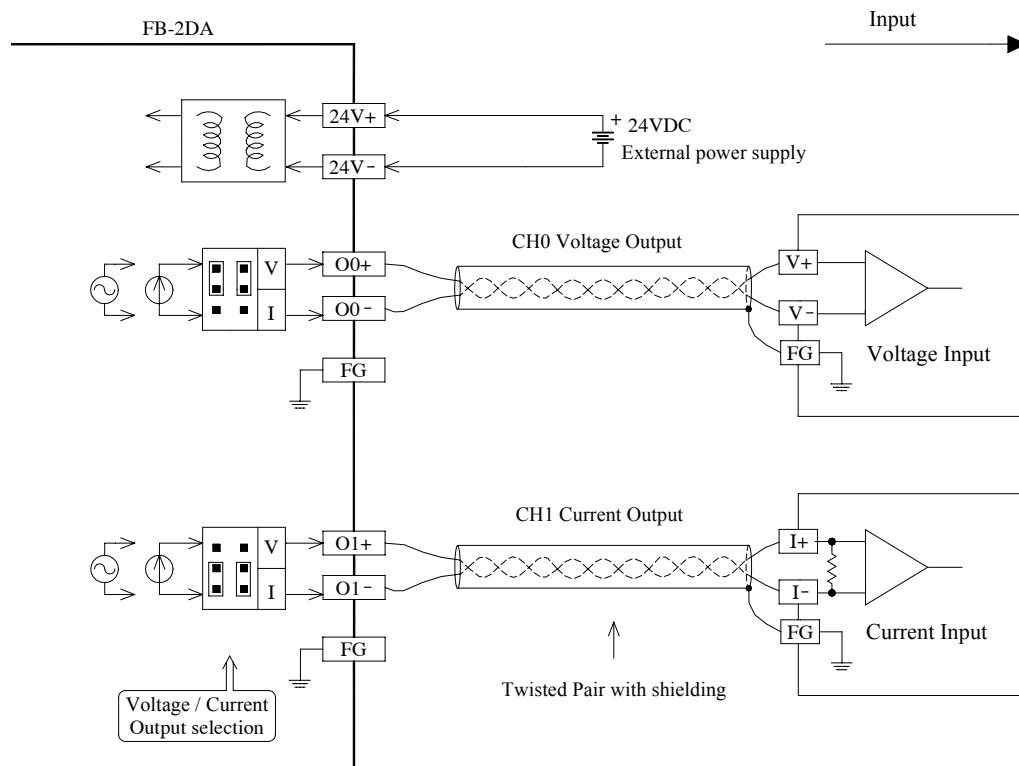
The jumper is located at the vertical board under PCB top view B, which is labeled as JP8 jumper. It has to take off the shell of FB-2DA to see it. The labeling and setting method are illustrated as following diagram:

	
Span of 10V or 20mA	Span of 5V or 10mA

● -10~10V	● -20~20mA	● -5~5V	● -10~10mA
● 0~10V	● 0~20mA	● 0~5V	● 0~10mA
● -6~10V	● -12~20mA	● -3~5V	● -6~10mA
● 2~10V	● 4~20mA	● 1~5V	● 2~10mA

## 19.5 The Output Circuit of FB-2DA



## 19.6 The Output Characteristic and Jumper Setting of D/A

As previously mentioned, the FB-2DA can yield 16 kind of outputs by the jumpers setting of V/I, B/U, O/N and 10V/5V. Hereby it tackles on the bipolar/unipolar (B/U) and span (10V/5V) two kinds of jumper setting to make 4 output conversion curves and are illustrated as following diagrams. These 4 conversion curves incorporating V/I and O/N setting can make the above mention 16 kind of outputs. For the selection of V/I and O/N, please refer to section 19.4.

Diagram 1: Bipolar 10V (20mA) Output Span

V/I type \ O/N mode	Normal mode (N)	Offset mode (O)	Jumper setting	JP8	JP3 · JP4
V/I type	(N)	(O)			
Voltage output (V)	-10~10V	-6~10V		10V	10V 5V
Current output (I)	-20~20mA	-12~20mA		5V	B U

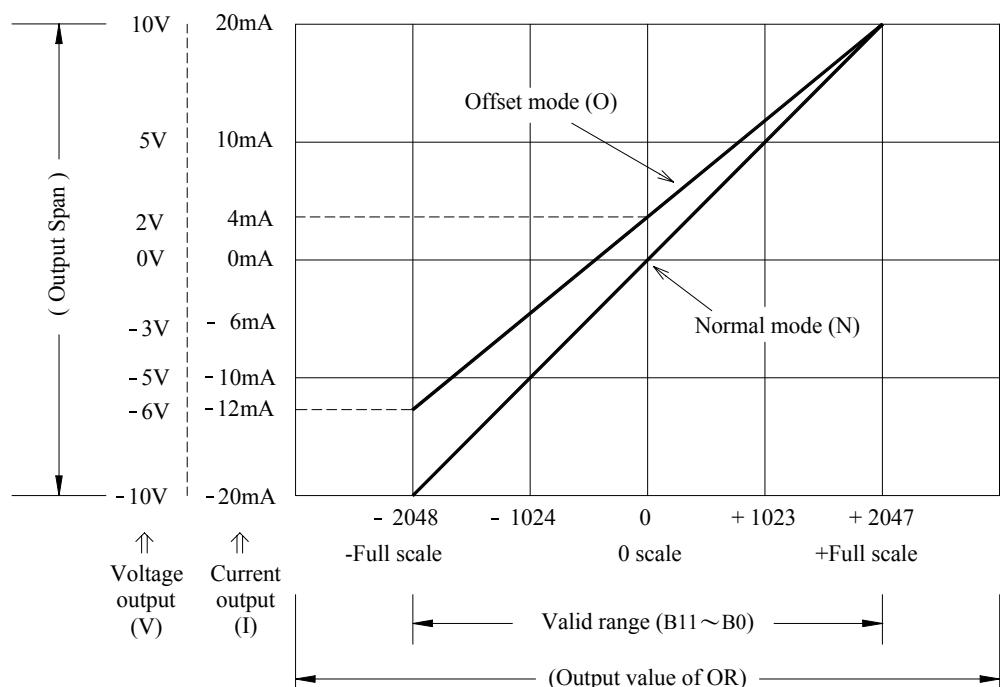


Diagram 2: Bipolar 5V (10mA) Output Span

V/I type \ O/N mode	Normal mode (N)	Offset mode (O)	Jumper setting	JP8	JP3 · JP4
V/I type	(N)	(O)			
Voltage output (V)	-5~5V	-3~5V		10V	10V 5V
Current output (I)	-10~10mA	-6~10mA		5V	B U

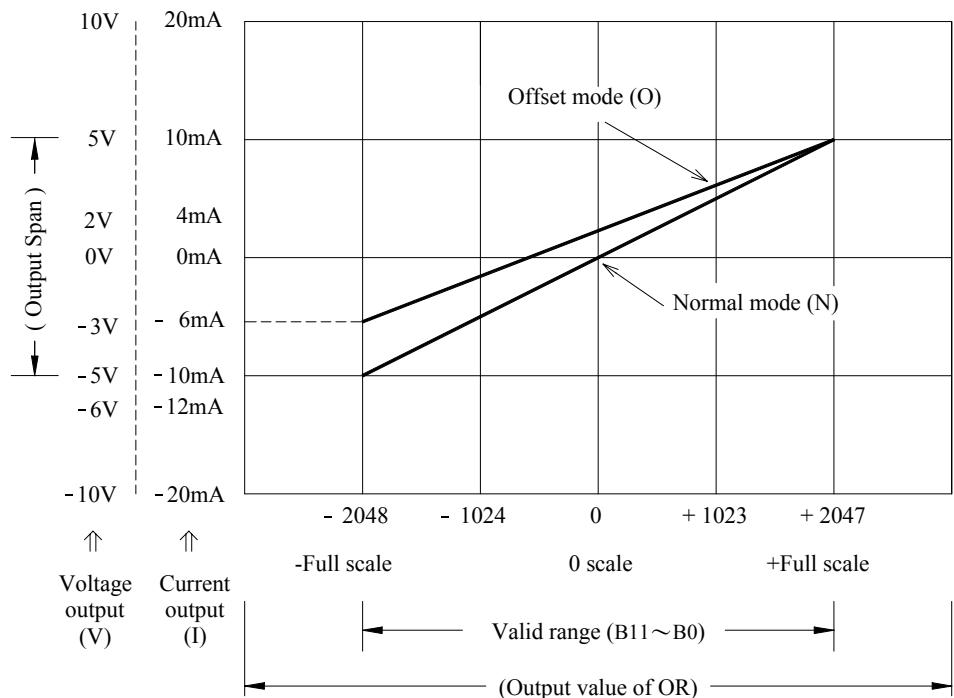


Diagram 3: Unipolar 10V (20mA) Output Span

V/I type	O/N mode	Normal mode (N)	Offset mode (O)	Jumper setting	JP8	JP3 · JP4
Voltage output (V)		0~10V	2~10V			
Current output (I)		0~20mA	4~20mA		10V 5V	B U

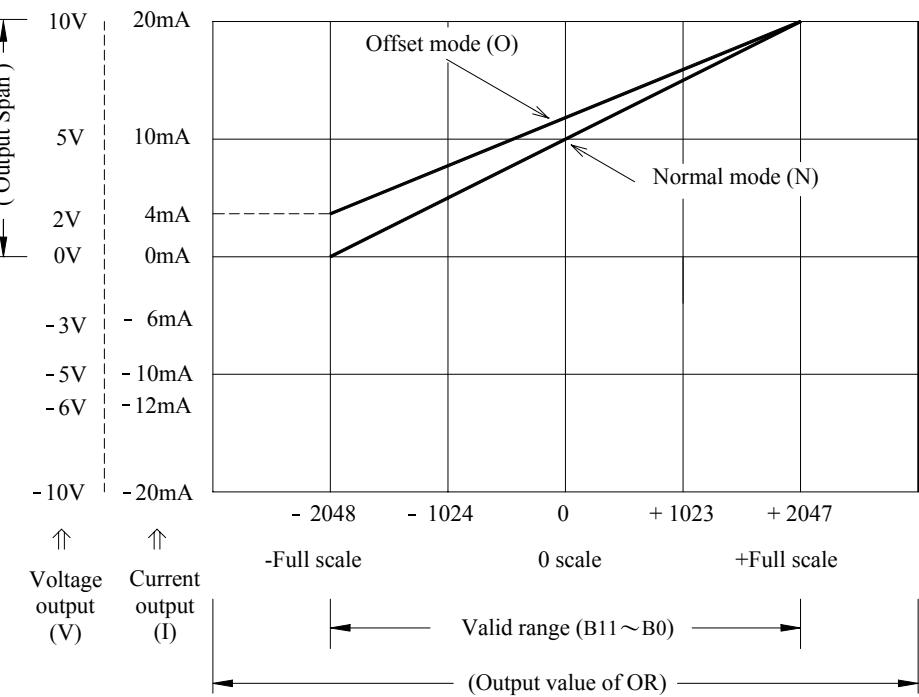
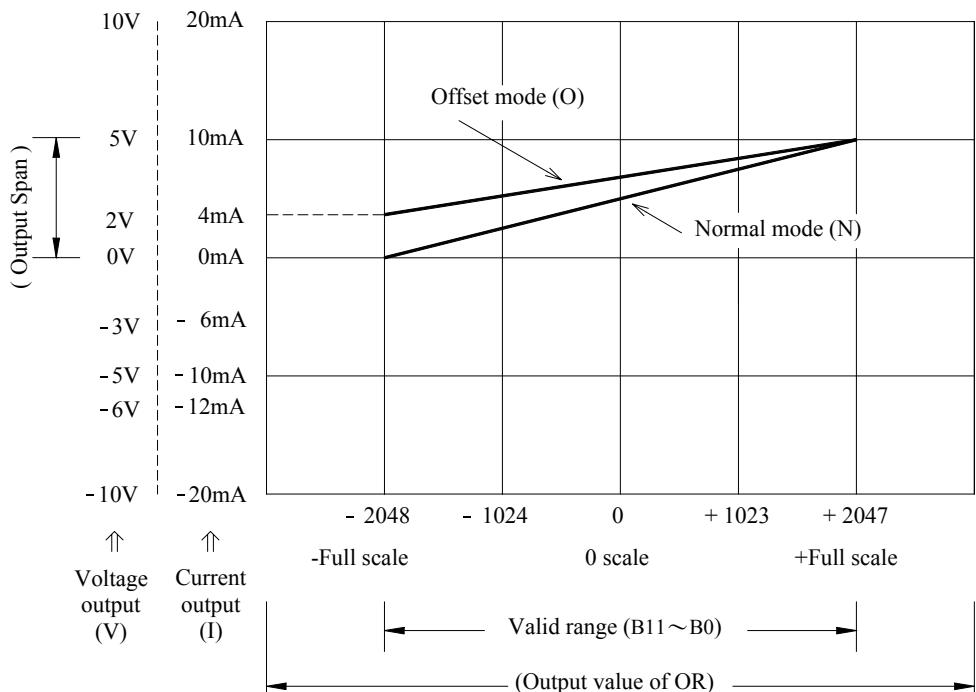


Diagram 4: Unipolar 5V (10mA) Output Span

V/I type	O/N mode (N)	Normal mode (O)
Voltage output (V)	0~5V	1~5V
Current (I)	0~10mA	2~10mA

Jumper setting	JP8	JP3 · JP4
	10V	B
	5V	U



## 19.7 Notifications for the operation of FB-2DA

### A FB-2DA Matching with the OS Version of Main Unit

FB-2DA can be installed to reach 64 points of D/A outputs (it must be the main unit with V3.30 and later version). If the OS version of main unit before version V3.30, it can cascade to connect 4 modules at the most, hence get a total analogue output of 8 points only.

### B The Processing of Unipolar output

FB-2DA expresses with -2048 as the minimum output value for 0V or 0mA and 2047 as the maximum output value for selected maximum output span while in unipolar output; the ladder program operation is used 0~4095 as the minimum and maximum output value for calculation; therefore, before filling the calculated output value to analogue output register OR (R3904 ~ R3967), it must first convert the 0~4095 value to corresponding value of -2048~2047 to get a correct output span. The treatment is quite simple; just make the calculated output value (0~4095) deducts 2048 and stores it to analogue output register OR (R3904~R3967) and that's done.

## Chapter 20 Temperature Measurement of FB-PLC and PID Control

FB-PLC provides two kinds of modules with different outlook and size for temperature measurement. One kind of the modules is called slim module with 2 points of general purpose analog input & 4 points of temperature input, named by FB-2AJ4 (for J-type thermocouple),FB-2AK4 (for K-type thermocouple),FB-2AH4 (for PT-100) and FB-2AT4 (for PT-1000). They can be expanded up to 8 modules, with 32 temperature inputs in total at the most. The other kind has a built-in 4 points analog input module with large number of temperature measuring points, named by FB-4AJxx or FB4Akxx (the xx could provide 12, 18, and 24 totally 3 kinds of measuring points). This module with large number of points can only be used alone and can not be installed together with other temperature measuring module or with other analog input modules.

Both of the above-mentioned temperatures measuring modules have their properly convenient instructions that are used for multiplexing temperature measurement. FB-2AJ(K)4/FB-2AH(T)4 employs FUN72(TP4) while FB-4AJ(K)xx employs FUN85(TPSNS) to get the engineering value of temperature measurement . As to the temperature control, it also has its properly convenient PID instructions. FB-2AJ(K)4/FB-2AH(T)4 employs FUN73(TSTC) while FB-4AJ(K)xx employs FUN86(TPCTL) to perform the PID operation to control the heating or cooling of the temperature process.

### 20.1 Specifications of temperature measuring modules of FB-PLC

#### 20.1.1 FB-2AJ(K)4 : with 4 points of J(K) thermocouple input and 2 points of analog input

Specifications			Module									
Items			FB-2AJ4		FB-2AK4							
Analog Input	Input points		2 (1 <sup>st</sup> & 2 <sup>nd</sup> analog inputs as the general purpose input) + 1 (3 <sup>rd</sup> analog input for temperature measurement)									
	Resolution		12 bits									
	Span	*Bipolar	*10V	1.Voltage	-10V ~ 10V	5.Current	-20mA~20mA					
			5V	2.Voltage	-5V ~ 5V	6.Current	-10mA~10mA					
		Unipolar	10V	3.Voltage	0V ~ 10V	7.Current	0mA~20mA					
			5V	4.Voltage	0V ~ 5V	8.Current	0mA~10mA					
Temperature Input	Input points		4 ( Multiplexing via 3 <sup>rd</sup> analog input )									
	Expansion allowed		32 points ( 8 modules )									
	Sensor		J-type thermocouple		K-type thermocouple							
	Valid range	*Bipolar	*10V	-200°C~750°C		-200°C~900°C						
			5V	-200°C~420°C		-200°C~450°C						
		Unipolar	10V	0°C~750°C		0°C~900°C						
			5V	0°C~420°C		0°C~450°C						
	Resolution		1°C									
	Compensation		Built-in cold junction compensation									
	Update rate		2 Sec. (Adjustable)									
Accuracy		Within ±1% of full scale										
Insulation		Photocouple isolation										
Power supply		24VDC±10%、5VA										

\* : It means default setting.

**20.1.2 FB-4AJ(K)12/18/24: with 12/18/24 points of J(K) thermocouple input and 4 points of analog input**

Specifications			Module								
Items			FB-4AJ(K)12	FB-4AJ(K)18	FB-4AJ(K)24						
Analog Input	Input points			4 (1 <sup>st</sup> ~4 <sup>th</sup> analog inputs as the general purpose input) + 4 (5 <sup>th</sup> ~8 <sup>th</sup> analog inputs for temperature measurement)							
	Span	*Bipolar	*10V	1.Voltage	-10V ~ 10V	5.Current	-20mA~20mA				
			5V	2.Voltage	-5V ~ 5V	6.Current	-10mA~10mA				
		Unipolar	10V	3.Voltage	0V ~ 10V	7.Current	0mA~20mA				
			5V	4.Voltage	0V ~ 5V	8.Current	0mA~10mA				
	Input points (Fixed)			12	18	24					
	Sensor			J-type thermocouple (K-type thermocouple)							
	Valid range	*Bipolar	*10V	-200°C~750°C		(-200°C~900°C)					
			5V	-200°C~420°C		(-200°C~450°C)					
		Unipolar	10V	0°C~750°C		(0°C~900°C)					
			5V	0°C~420°C		(0°C~450°C)					
Resolution			1°C								
Compensation			Built-in cold junction compensation								
Update rate			2 Sec. (Adjustable)								
Accuracy			Within ±1% of full scale								
Insulation			Photocouple isolation								
Power supply			24VDC±10%、5VA								

\* : It means default setting.

**20.1.3 FB-2AH(T)4: with 4 points of 3-wires PT-100 (PT-1000) RTD input and 2 points of analog input**

Specifications			Module								
Items			FB-2AH4 (PT-100)	FB-2AT4 (PT-1000)							
Analog Input	Input points			2 (1 <sup>st</sup> & 2 <sup>nd</sup> analog inputs as the general purpose input) + 1 (3 <sup>rd</sup> analog input for temperature measurement)							
	Resolution			12 bits							
	Span (Bipolar)	*10V	1.Voltage	-10V ~ 10V	3.Current	-20mA~20mA					
		5V	2.Voltage	-5V ~ 5V	4.Current	-10mA~10mA					
	Input points			4 (Multiplexing via 3 <sup>rd</sup> analog input)							
	Expansion allowed			32 points (8 modules)							
	Valid range	DIN	*10V	-49.8°C ~ 146.6°C							
			5V	-12.3°C ~ 83.6°C							
		JIS	*10V	-50.7°C ~ 149.2°C							
			5V	-12.5°C ~ 85.1°C							
Resolution			0.1°C								
Update rate			2 Sec. (Adjustable)								
Accuracy			Within ±1% of full scale								
Insulation			Photocouple isolation								
Power supply			24VDC±10%、5VA								

\* : It means default setting.

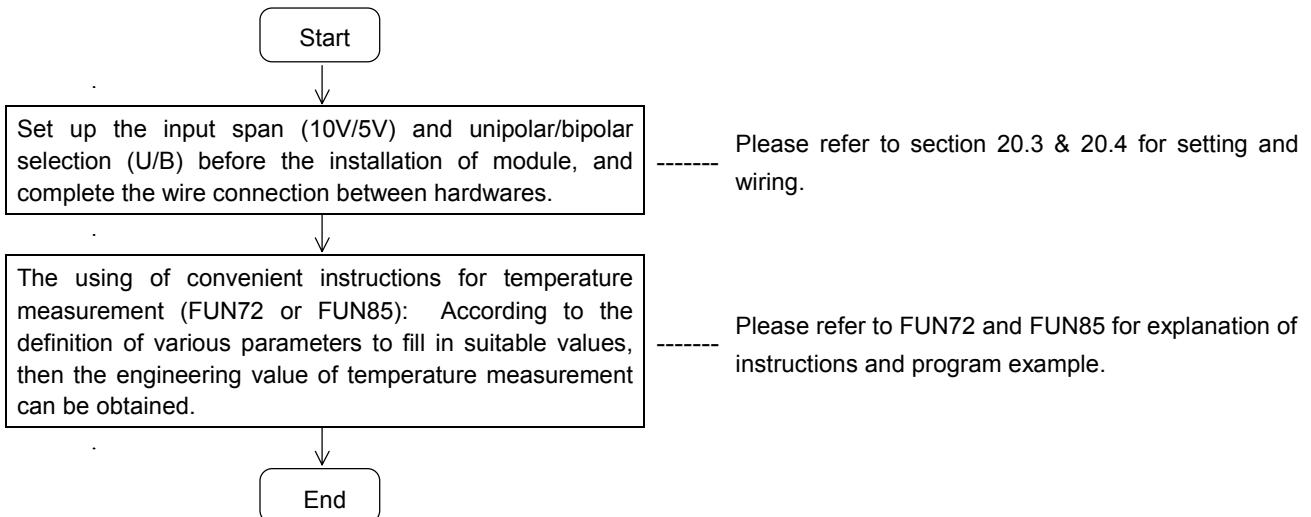
Note: The temperature modules mentioned above all with built-in general purpose analog inputs and dedicated analog input for temperature measurement, the memory mapping of these modules as followings:

Addressing of FB-2AJ(K)4 and FB-2AH(T)4 : The 1<sup>st</sup> and 2<sup>nd</sup> analog inputs are the general purpose input by accessing R3840 and R3841; and the 3<sup>rd</sup> analog input is dedicated for 4 points of temperature measurement (by multiplexing method) by accessing R3842 (if this module is the 1<sup>st</sup> analog input expansion module).

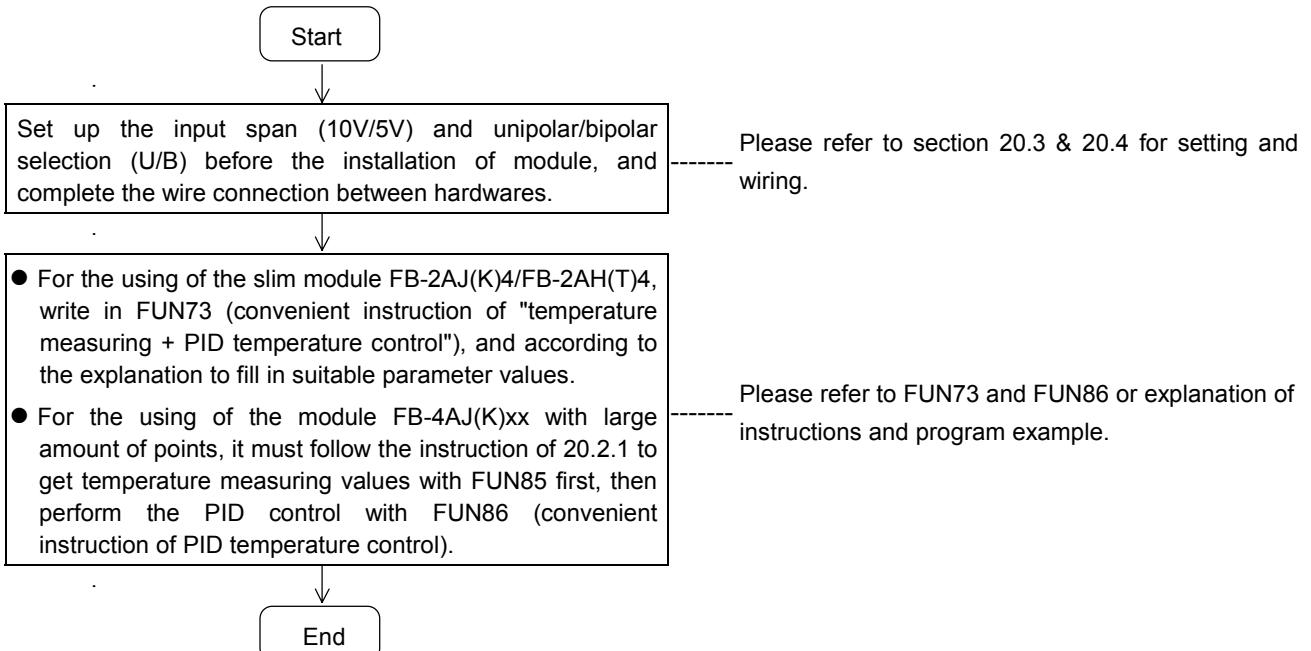
Addressing of FB-4AJ(K)xx : This kind of modules could only be installed alone, therefore, the 1<sup>st</sup> ~4<sup>th</sup> analog inputs are the general purpose input by accessing R3840~R3843, and 5<sup>th</sup>~8<sup>th</sup> analog inputs are dedicated for upto 24 points of temperature measurement (by multiplexing method, one analog input for 6 points of temperature measurement).

## 20.2 The procedure of using temperature measuring module of FB-PLC Explanation

### 20.2.1 Temperature measurement only

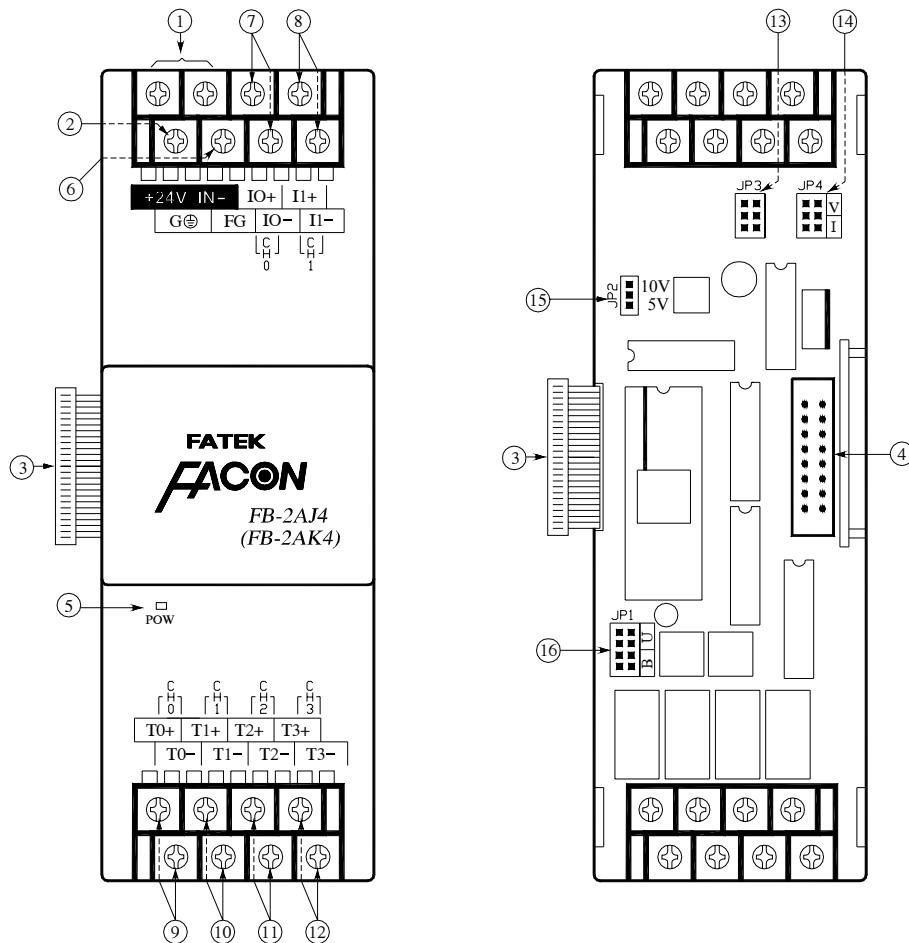


### 20.2.2 Closed loop PID temperature control



## 20.3 Explanation on the hardware of temperature measuring module

### 20.3.1 The outlook of FB-2AJ(K)4 and top view of PC-board



A : Outlook of top view

B : PC-board top view (uncovered)

① : External power input terminal –

Power supply for analogue circuit of FB-2AJ(K)4 module, supply voltage is 24VDC±20% .

② : Protection ground terminal –

To connect to the safety earth ground of the power system.

③ : Expansion input cable –

It must be connected to the front of expansion unit or main unit.

④ : Expansion output connector –

Provide the connection for next expansion unit.

⑤ : Power indicator –

Indicating the status of external power input and power supply of FB-2AJ(K)4 analogue circuit.

⑥ : Framing ground terminal –

To connect to the shielding of the analog input wiring.

⑦ : Analog input terminal for AI0 –

To connect to the 1<sup>st</sup> general purpose analog input.

⑧ : Analog input terminal for AI1 –

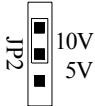
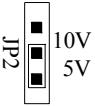
To connect to the 2<sup>nd</sup> general purpose analog input.

⑨～⑫ : Temperature input terminals for CH0～CH3 – To connect to the corresponding thermocouple.

⑬ · ⑭ Selection jumpers of voltage(V)/current(I) input for AI0 (JP3) and AI1 (JP4)

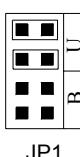
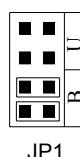
⑯ : Selection jumper of input span 10V/5V

The selection to define the input span of all analog inputs of this module. If setting the jumper at 10V position, it represents the measurement range of 10V/20mA/1000°C; if setting the jumper at 5V position, it represents the measurement range of 5V/10mA/500°C.

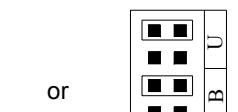
Jumper Setting			 10V 5V	 10V 5V
Span of analog Input	Unipolar (U)	Voltage (V)	0V~10V	0V~5V
		Current (I)	0mA~20mA	0mA~10mA
	Bipolar (B)	Voltage (V)	-10V~10V	-5V~5V
		Current (I)	-20mA~20mA	-10mA~10mA
Span of Temperature input	Unipolar (U)		0°C~1000°C	0°C~500°C
	Bipolar (B)		-1000°C~1000°C	-500°C~500°C

⑰ : Unipolar (U) / Bipolar (B) selection

The selection to define the input polarity of all analog inputs of this module. These two jumpers must be inserted horizontally in pairs according to the U, B text direction (which is horizontally printed in its direction) to position B or U as following illustration.

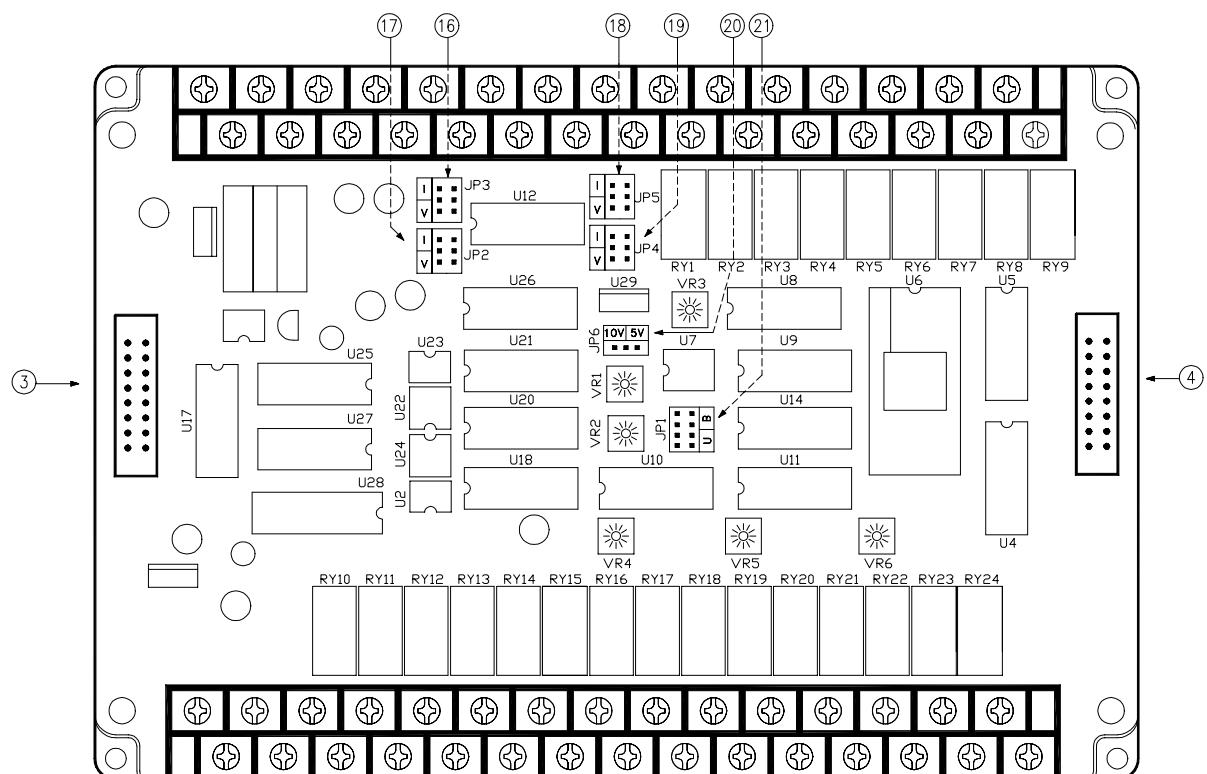
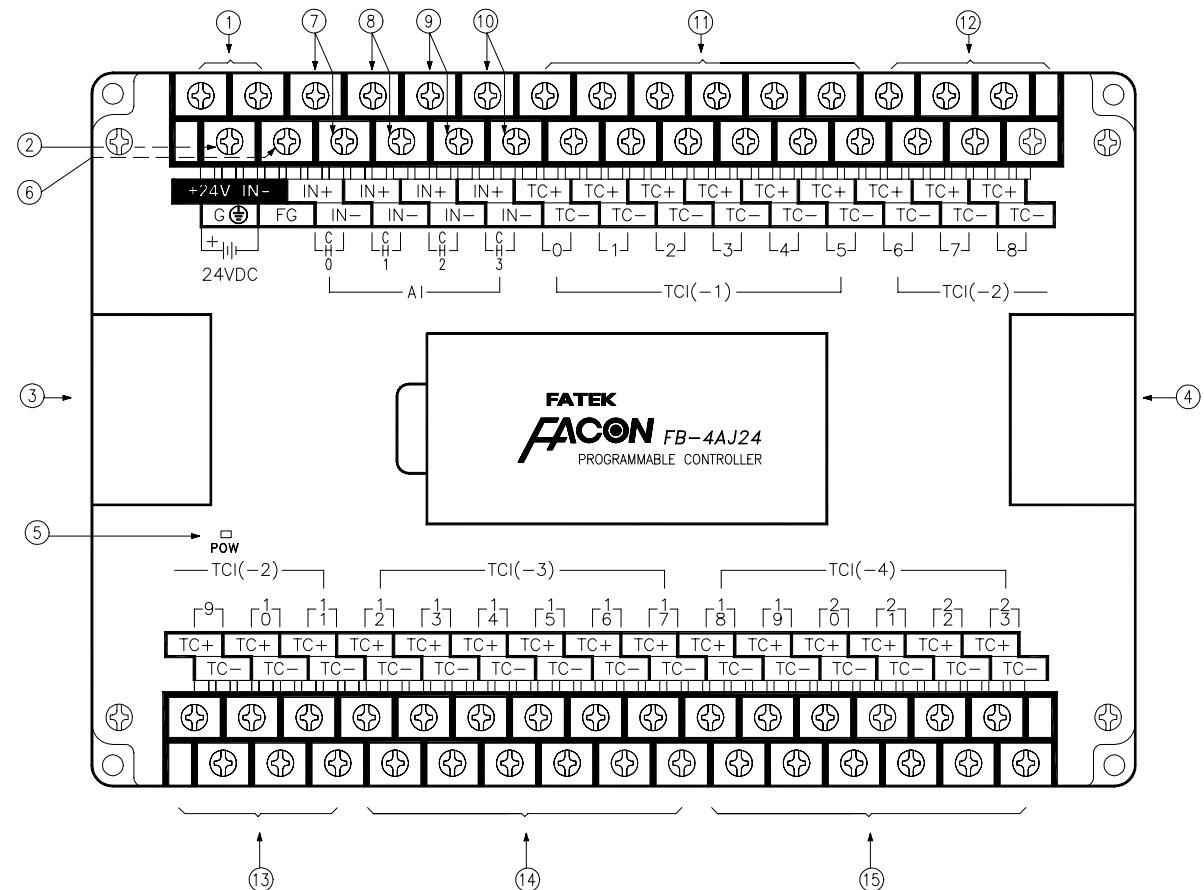
	
Unipolar (U)	Bipolar (B)
	

JP1                          or                          JP1



Jumper vertically inserted or not inserted in pairs are all incorrect

### 20.3.2 The outlook of FB-4AJ(K)xx and top view of PC Board



- ① : External power input terminal –  
Power supply for analogue circuit of FB-4AJ(K)xx module, supply voltage is 24VDC±20% .
- ② : Protection ground terminal –  
To connect to the safety earth ground of the power system.
- ③ : Expansion input cable –  
It must be connected to the front of the expansion unit or main unit.
- ④ : Expansion output connector –  
Provide the connection of next expansion unit.
- ⑤ : Power indicator –  
Indicating the status of the external power input and the power supply for analogue circuit of this module.
- ⑥ : Framing ground terminal –  
To connect to the shielding of analog input wiring.

⑦ : 1<sup>st</sup> analog input terminal (CH0) –

⑧ : 2<sup>nd</sup> analog input terminal (CH1) –

⑨ : 3<sup>rd</sup> analog input terminal (CH2) –

⑩ : 4<sup>th</sup> analog input terminal (CH3) –

This kind of module can be used alone and can't be installed together with other analog input module; therefore it has these 4 points of analog input at the most.

⑪ : 1<sup>st</sup> group of thermocouple input terminals (TC0~TC5) – To connect to corresponding thermocouple

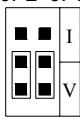
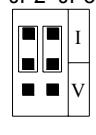
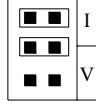
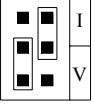
⑫~⑬ : 2<sup>nd</sup> group of thermocouple input terminals (TC6~TC11) – To connect to corresponding thermocouple

⑭ : 3<sup>rd</sup> group of thermocouple input terminals (TC12~TC17) – To connect to corresponding thermocouple  
(only for FB-4AJ(K)18 or FB-4AJ(K)24)

⑮ : 4<sup>th</sup> group of thermocouple input terminals (TC18~TC23) – To connect to corresponding thermocouple  
(only for FB-4AJ(K)24)

⑯~⑲ : Selection jumpers for voltage(V)/current (I) input of analog inputs CH0~CH3

The four analog inputs of FB-4AJ(K)xx can be selected individually to be voltage or current input (JP2 for CH0·JP3 for CH1 · JP4 for CH2 · JP5 for CH3). The selection and insertion of jumper must be identical in direction to the printed text V and I next to it and should be vertically inserted into the position of V or I in pairs.

○	×
Voltage (V)	Current (I)
JP2~JP5 	JP2~JP5 
 or 	
Either jumpers vertically placed or not inserted in pairs are both incorrect.	

⑯ : Selection jumper of input span 10V/5V

The selection to define the input span of all analog inputs of this module. If setting the jumper at 10V position, it represents the measurement range of 10V/20mA/1000°C; if setting the jumper at 5V position, it represents the measurement range of 5V/10mA/500°C.

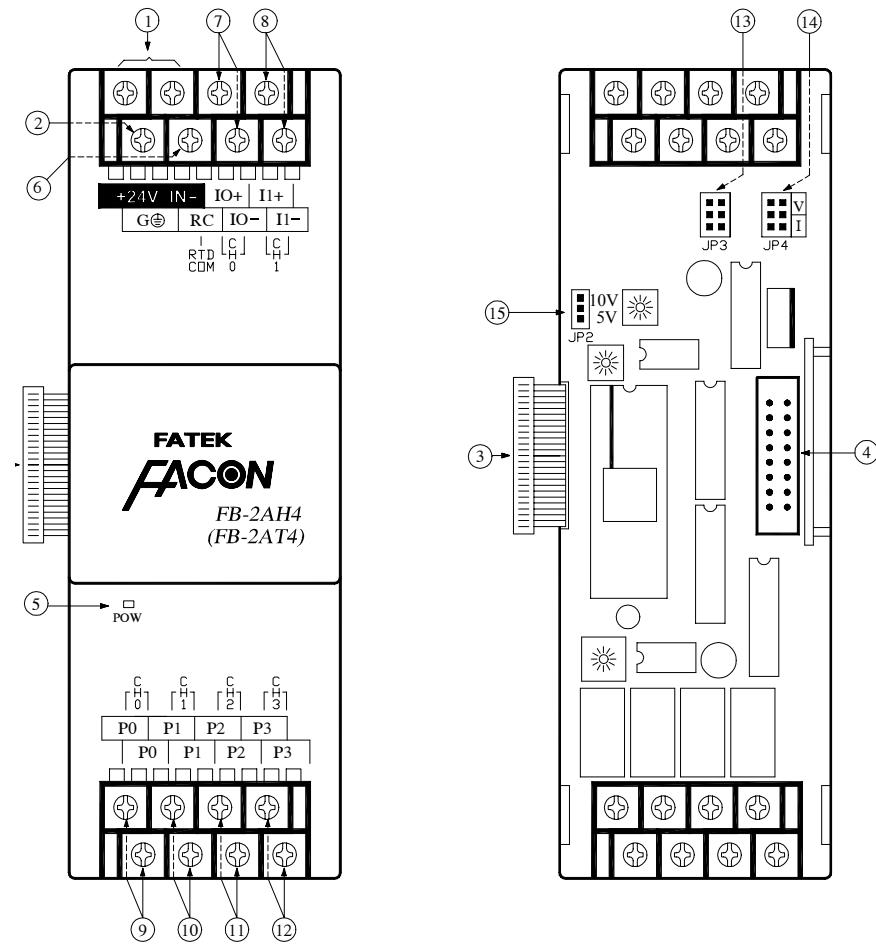
Jumper setting			JP6 10V 5V	JP6 10V 5V
Analog input	Unipolar (U)	Voltage (V)	0V~10V	0V~5V
		Current (I)	0mA~20mA	0mA~10mA
	Bipolar (B)	Voltage (V)	-10V~10V	-5V~5V
		Current (I)	-20mA~20mA	-10mA~10mA
Temperature span	Unipolar (U)		0°C~500°C	
	Bipolar (B)		-500°C~500°C	

⑰ : Unipolar (U) / Bipolar (B) selection

The selection to define the input polarity of all analog inputs of this module. These two jumpers must be horizontally placed in pairs to position B or U, according to the marked text direction of B and U beside the jumper.

Unipolar (U)	Bipolar (B)	
		 or <p>Either jumpers vertically placed or not inserted in pairs are both incorrect.</p>

### 20.3.3 The outlook of FB-2AH(T)4 and top view of PC-board



A : Outlook of top view

B : PC-board top view (uncovered)

- ① : External power input terminal –  
Power supply for analogue circuit of FB-2AH(T)4 module, supply voltage is 24VDC±20% .
- ② : Protection ground terminal –  
To connect to the safety earth ground of the power system.
- ③ : Expansion input cable –  
It must be connected to the front of expansion unit or main unit.
- ④ : Expansion output connector –  
Provide the connection for next expansion unit.
- ⑤ : Power indicator –  
Indicating the status of external power input and power supply of FB-2AH(T)4 analogue circuit.
- ⑥ : Common terminal for 3-wires RTD input –  
To connect to the common wire (in general ,the color is red) of each 3-wires RTD input.
- ⑦ : Analog input terminal for AI0 –  
To connect to the 1<sup>st</sup> general purpose analog input.
- ⑧ : Analog input terminal for AI1 –  
To connect to the 2<sup>nd</sup> general purpose analog input.
- ⑨ : Input terminal for 1<sup>st</sup> RTD input –  
To connect to the signal wires (in general, the color is white) of 1<sup>st</sup> 3-wires RTD input.
- ⑩ : Input terminal for 2<sup>nd</sup> RTD input –  
To connect to the signal wires (in general, the color is white) of 2<sup>nd</sup> 3-wires RTD input

⑪ : Input terminal for 3<sup>rd</sup> RTD input –

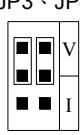
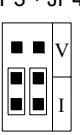
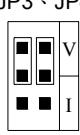
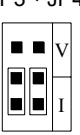
To connect to the signal wires (in general, the color is white) of 3<sup>rd</sup> 3-wires RTD input.

⑫ : Input terminal for 4<sup>th</sup> RTD input –

To connect to the signal wires (in general, the color is white) of 4<sup>th</sup> 3-wires RTD input.

⑬ · ⑭ Selection jumpers of voltage(V)/current(I) input for AI0 (JP3) and AI1 (JP4)

The jumper must be inserted vertically in pairs according to the V, I text direction (which is vertically printed in its direction) to position V or I as following illustration.

○	✗
<b>Voltage (V)</b> 	<b>Current (I)</b> 
<b>JP3 · JP4</b> 	<b>JP3 · JP4</b> 

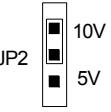
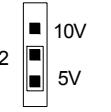
JP3 · JP4      JP3 · JP4  
or

Either jumpers horizontally placed or not inserted in pairs are both incorrect.

⑮ : Selection jumper of input span 10V/5V

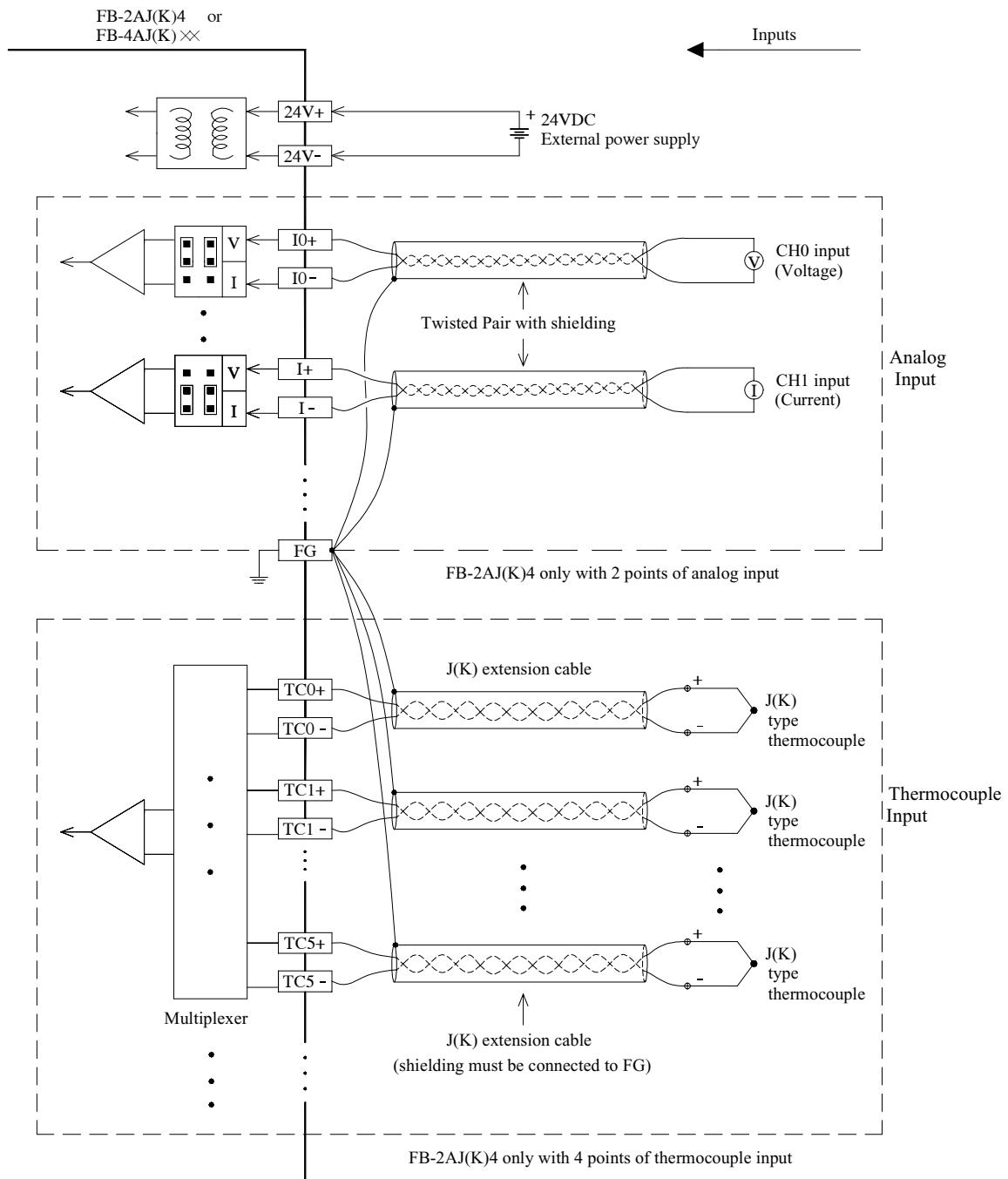
The selection to define the input span of all analog inputs of this module. The polarity of this module supports bipolar only. If setting the jumper at 10V position, it means the range of measurement is  $\pm 10V/\pm 20mA$  and the temperature range is  $-49.8^{\circ}C \sim 146.6^{\circ}C$  (DIN) or  $-50.7^{\circ}C \sim 149.2^{\circ}C$  (JIS); if setting the jumper at 5V position, it means the range of measurement is  $\pm 5V/\pm 10mA$  and the temperature range is  $-12.3^{\circ}C \sim 83.6^{\circ}C$  (DIN) or  $-12.5^{\circ}C \sim 85.1^{\circ}C$  (JIS)

Note : FB-2AH(T)4 is fixed at bipolar, without unipolar/bipolar setting!

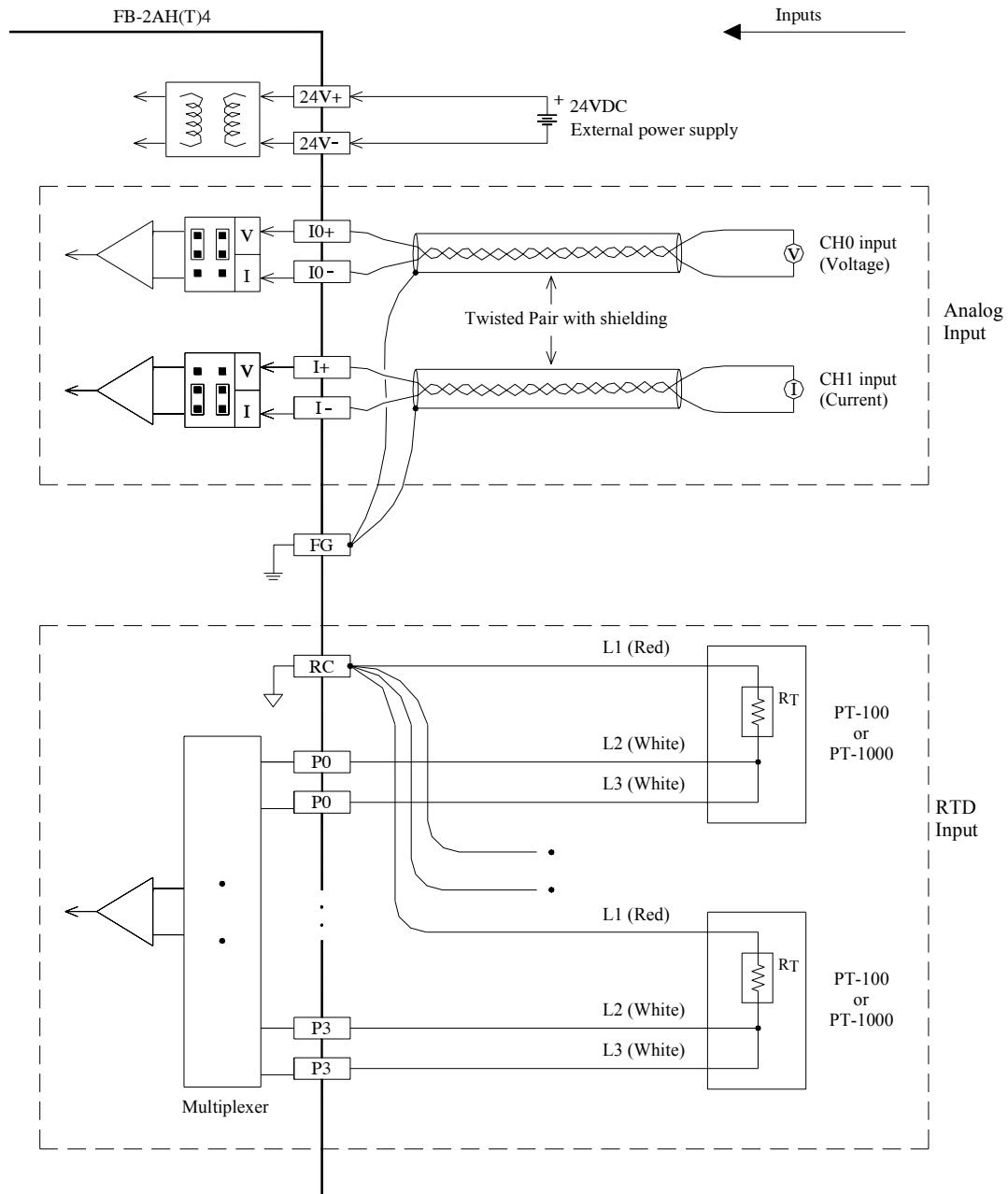
Range		Jumper setting	
Span of Analog Input	Voltage (V)		
	Current (I)	$-10V \sim 10V$	$-5V \sim 5V$
Span of RTD Input	DIN	$-49.8^{\circ}C \sim 146.6^{\circ}C$	$-12.3^{\circ}C \sim 83.6^{\circ}C$
	JIS	$-50.7^{\circ}C \sim 149.2^{\circ}C$	$-12.5^{\circ}C \sim 85.1^{\circ}C$

## 20.4 Wiring of the temperature modules

### 20.4.1 Wiring of the J/K thermocouple input module



## 20.4.2 Wiring of 3-wires PT-100/PT-1000 RTD input module



## 20.5 The input characteristic and jumper setting of temperature module

### 20.5.1 Temperature module of thermocouple inputs

The characteristics of general purpose analog inputs of FB-2AJ(K)4/FB-4AJ(K)xx are identical to the FB-6AD's. Therefore it will not be explained here, please refer to Chapter 18 for details. This section will only tackle on the subject of temperature measuring. The functions and characters of temperature measurement circuit of FB-4AJ(K)xx are all the same as FB-2AJ(K)4's. The conversion character of which is graphically illustrated as follows. Please note that effective measuring range of J-type thermocouple falls in  $-200^{\circ}\text{C} \sim 750^{\circ}\text{C}$ , but is  $-200^{\circ}\text{C} \sim 900^{\circ}\text{C}$  for K-type. Therefore the marking scale below  $200^{\circ}\text{C}$  and over  $750^{\circ}\text{C}$  or  $900^{\circ}\text{C}$  on the conversion curve does not make sense. Also, it is not possible for temperature to fall below the absolute zero. No matter it's for unipolar/bipolar or  $1000^{\circ}\text{C}/500^{\circ}\text{C}$  span, the content of IR (R3840~R3903) at most it can reach 1842. For the scale exceeding 1842, it is treated for line broken check only.

Figure 1: Bipolar  $1000^{\circ}\text{C}$  input span

Jumper setting	10V	B
----------------	-----	---

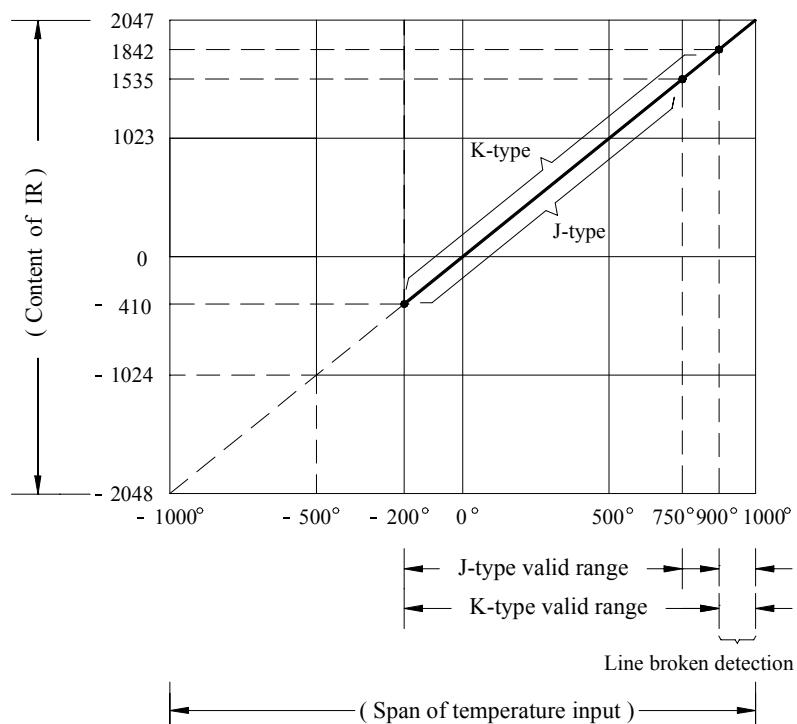


Figure 2: Bipolar 500°C input span

Jumper setting	5V	B
----------------	----	---

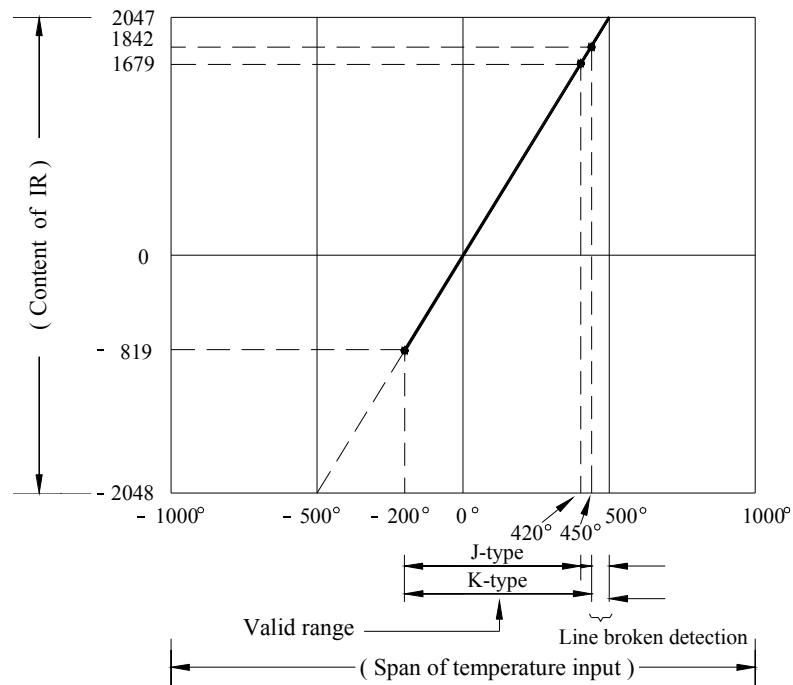


Figure 3: Unipolar 1000°C input span

Jumper Setting	10V	U
----------------	-----	---

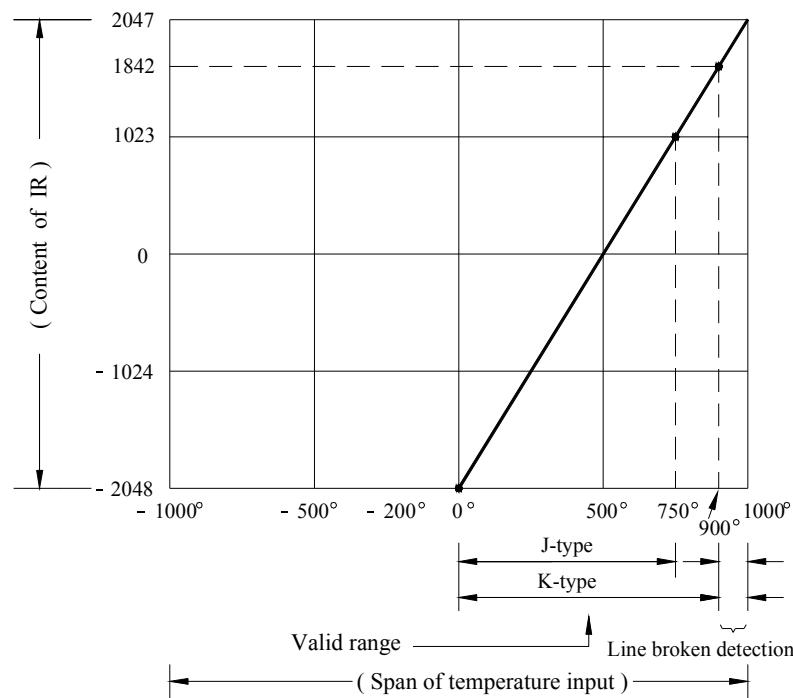
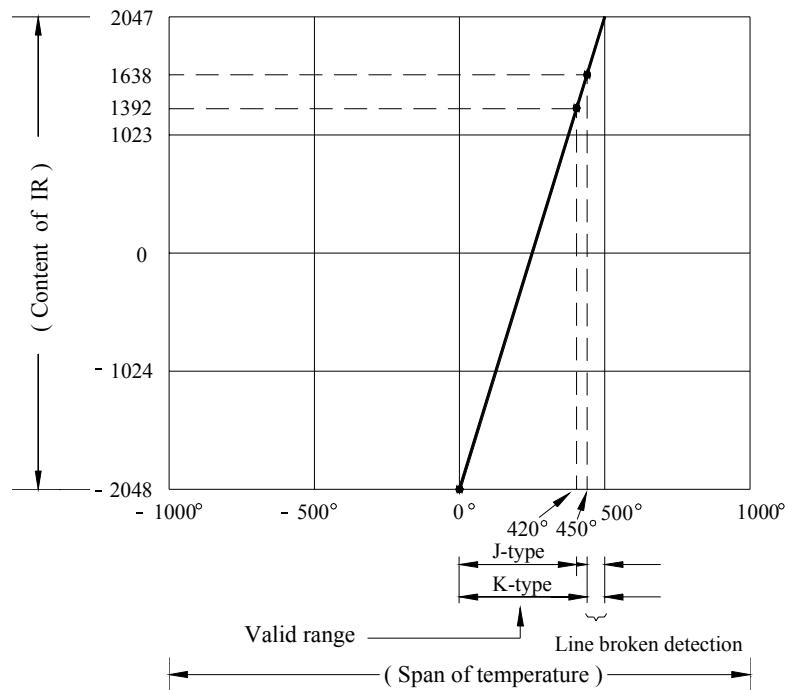


Figure 4: Unipolar 500°C input span

Jumper setting	5V	U
----------------	----	---



#### 20.5.2 Temperature module of 3-wires RTD inputs

The characteristics of general purpose analog inputs of FB-2AH(T)4 are identical to the FB-6AD's. Therefore it will not be explained here, please refer to Chapter 18 for details. This section will only tackle on the subject of temperature measuring. By setting the jumper (JP2) to select the measurement range; if setting the jumper at 10V position, the temperature range is  $-49.8^{\circ}\text{C} \sim 146.6^{\circ}\text{C}$  (DIN) or  $-50.7^{\circ}\text{C} \sim 149.2^{\circ}\text{C}$  (JIS) and if the jumper at 5V position, the range is  $-12.3^{\circ}\text{C} \sim 83.6^{\circ}\text{C}$  (DIN) or  $-12.5^{\circ}\text{C} \sim 85.1^{\circ}\text{C}$  (JIS). Please refer to section 20.3.3 for details. The conversion character of this module is illustrated as follows.

Figure 5: Bipolar  $-49.8^{\circ}\text{C} \sim 146.6^{\circ}\text{C}$  (DIN) ,  $-50.7^{\circ}\text{C} \sim 149.2^{\circ}\text{C}$  (JIS)

Jumper setting | 10V

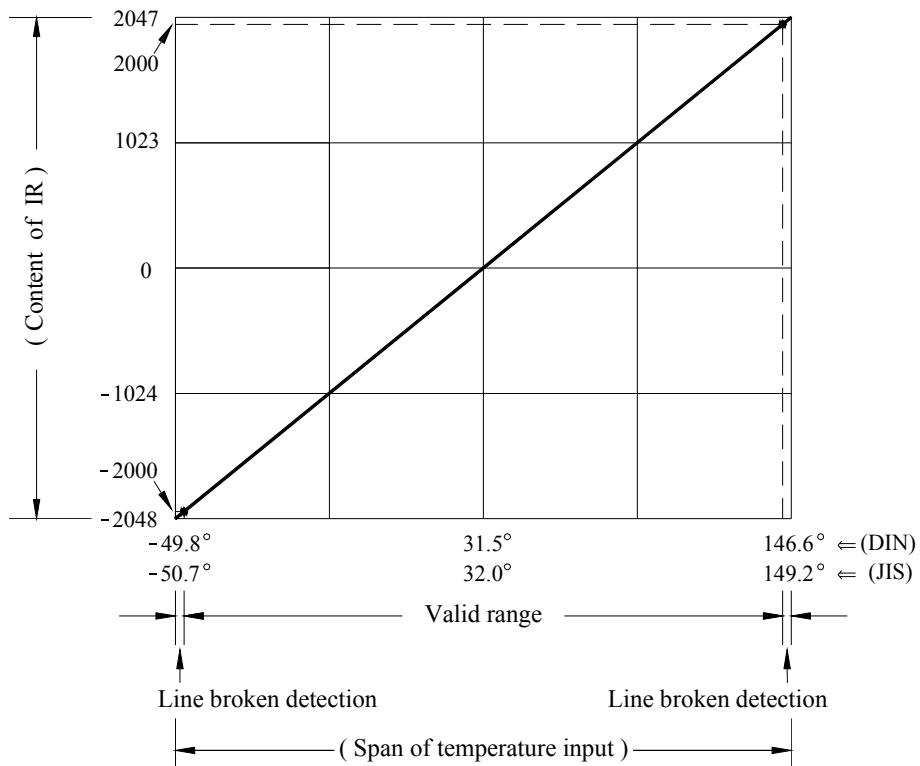
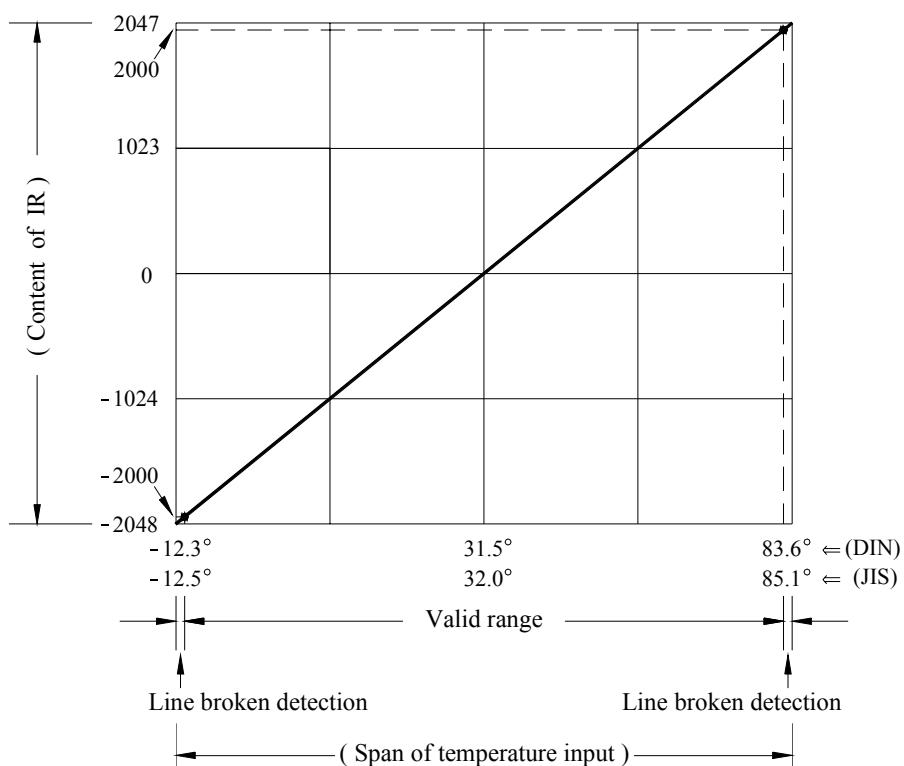


Figure 6: Bipolar  $-12.3^{\circ}\text{C} \sim 83.6^{\circ}\text{C}$  (DIN) ,  $-12.5^{\circ}\text{C} \sim 85.1^{\circ}\text{C}$  (JIS)

Jumper setting | 5V



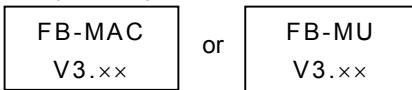
## 20.6 Notifications for the operation of temperature modules

- A、 Matching with the version of main unit

The temperature measuring module FB-4AJ(K)xx must run under the main unit with OS version V.3.30 or later that it can work correctly.

The temperature measuring modules FB-2AJ(K)4 and FB-2AH(T)4 must run under the main unit with OS version V.3.43 or later that it can work correctly.

Note: To tell the version of the main unit, you can just open up the cover on center of the module and check the sticker bearing print out like



The "V3.xx" indicates the OS version of the main unit.

- B、 FB-2AJ(K)4/FB-2AH(T)4 can not be used together with FB-4AJ(K)xx module or FB-8AD analog input module.

- C、 FB-4AJ(K)xx can be installed alone only; it can not exist together with other analog input module or temperature measuring module.

- D、 The unipolar processing of FB-2AJ(K)4 and FB-4AJ(K)xx

The minimum value (0V or 0mA) for unipolar analog input is expressed as -2048 and maximum value is 2047. For easier processing of the calculation, it is necessary to add up the content of IR (R3840~R3903) with a deviation value of 2048, hence to adjust the unipolar analog input value to be 0~4095.

- E、 FB-2AH(T)4 only supports bipolar analog input; it means the resolution will be half if the input is unipolar signal

## 20.7 Instructions explanation and program example for temperature measurement and PID temperature control of FB-PLC

The followings are the instructions explanation and program example for temperature measurement and PID temperature control of FB-PLC.

## Measuring instruction proper to FB-2AJ(K)4/FB-2AH(T)4

FUN 72 TP4	Convenient instruction proper to FB-2AJ(K)4/FB-2AH(T)4 temperature module	FUN 72 TP4																																																																							
	<p>Execution control—EN</p> <table border="1" style="margin-left: 20px; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">72.TP4</td> </tr> <tr> <td>Tp :</td> <td>ERR—parameter error</td> </tr> <tr> <td>Pl :</td> <td>ALM—sensor line broken</td> </tr> <tr> <td>Sm :</td> <td></td> </tr> <tr> <td>Ym :</td> <td></td> </tr> <tr> <td>AR :</td> <td></td> </tr> <tr> <td>TR :</td> <td></td> </tr> <tr> <td>WR:</td> <td></td> </tr> </table> <table border="1" style="margin-left: 20px; border-collapse: collapse; width: 150px;"> <tr> <th rowspan="2">Oper- and</th> <th>Y</th> <th>HR</th> <th>IR</th> <th>ROR</th> <th>DR</th> <th>K</th> </tr> <tr> <th>Y0   Y255</th> <th>R0   R3839</th> <th>R3840   R3903</th> <th>R5000   R8071</th> <th>D0   D3071</th> <th></th> </tr> </table> <table border="1" style="margin-left: 20px; border-collapse: collapse; width: 150px;"> <tr> <td>Tp</td> <td></td> <td></td> <td></td> <td></td> <td>0~3</td> </tr> <tr> <td>Pl</td> <td></td> <td></td> <td></td> <td></td> <td>0~3</td> </tr> <tr> <td>Sm</td> <td></td> <td></td> <td></td> <td></td> <td>n×4 · n=0~7</td> </tr> <tr> <td>Ym</td> <td>○</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>AR</td> <td></td> <td>○</td> <td></td> <td></td> <td></td> </tr> <tr> <td>TR</td> <td>○</td> <td></td> <td>○*</td> <td>○</td> <td></td> </tr> <tr> <td>WR</td> <td>○</td> <td></td> <td>○*</td> <td>○</td> <td></td> </tr> </table>	72.TP4		Tp :	ERR—parameter error	Pl :	ALM—sensor line broken	Sm :		Ym :		AR :		TR :		WR:		Oper- and	Y	HR	IR	ROR	DR	K	Y0   Y255	R0   R3839	R3840   R3903	R5000   R8071	D0   D3071		Tp					0~3	Pl					0~3	Sm					n×4 · n=0~7	Ym	○					AR		○				TR	○		○*	○		WR	○		○*	○		<p>Tp : Type of sensor  =0, K-type thermocouple  =1, J-type thermocouple  =2, PT-100 RTD  =3, PT-1000 RTD</p> <p>Pl : Setting of polarity and span  =0 , 0~10V (Unipolar)  =1 , 0~5V (Unipolar)  =2 , -10~10V (Bipolar)  =3 , -5~5V (Bipolar)</p> <p>Unipolar: U/B jumper set at U  Bipolar: U/B jumper set at B  Span : 5V/10V jumper setting</p> <p>Sm : Starting point of temperature measurement of this module.  Sm=0 , 4 , 8…… , 28</p> <p>Ym : Starting address of discrete output of this module for multiplexing temperature input; it takes 8 points. When expansion module with discrete output will be installed after the temperature module, the discrete output address of which must be added 8.</p> <p>AR : Address of analog input for temperature measurement of this module; which is the 3<sup>rd</sup> analog input. When expansion module with analog input will be installed after the temperature module, the analog address of which must be added 3.</p> <p>TR : Starting register of the engineering value of temperature measurement, 4 registers in total.</p> <p>WR : Starting of working register for this instruction. It takes 8 registers and can't be repeated in using.</p>
72.TP4																																																																									
Tp :	ERR—parameter error																																																																								
Pl :	ALM—sensor line broken																																																																								
Sm :																																																																									
Ym :																																																																									
AR :																																																																									
TR :																																																																									
WR:																																																																									
Oper- and	Y	HR	IR	ROR	DR	K																																																																			
	Y0   Y255	R0   R3839	R3840   R3903	R5000   R8071	D0   D3071																																																																				
Tp					0~3																																																																				
Pl					0~3																																																																				
Sm					n×4 · n=0~7																																																																				
Ym	○																																																																								
AR		○																																																																							
TR	○		○*	○																																																																					
WR	○		○*	○																																																																					

### Function guide and notifications

FB-2AJ(K)4/FB-2AH(T)4 multiplexing temperature module occupies 3 points of analog input address and 8 points of discrete output address in physical, more detail as followings:

- FB-2AJ4 provides 2 points of general purpose analog inputs (1<sup>st</sup> and 2<sup>nd</sup> analog input) and 4 points of J-type thermocouple inputs for temperature measurement (With the combination of 3<sup>rd</sup> analog input and 8 points of discrete output making 4 points of temperature measurement).
- FB-2AK4 provides 2 points of general purpose analog inputs (1<sup>st</sup> and 2<sup>nd</sup> analog input) and 4 points of K-type thermocouple inputs for temperature measurement (With the combination of 3<sup>rd</sup> analog input and 8 points of discrete output making 4 points of temperature measurement).
- FB-2AH4 provides 2 points of general purpose analog inputs (1<sup>st</sup> and 2<sup>nd</sup> analog input) and 4 points of 3-lines PT-100 RTD inputs for temperature measurement (With the combination of 3<sup>rd</sup> analog input and 8 points of discrete output making 4 points of temperature measurement).
- FB-2AT4 provides 2 points of general purpose analog inputs (1<sup>st</sup> and 2<sup>nd</sup> analog input) and 4 points of 3-lines PT-1000 RTD inputs for temperature measurement (With the combination of 3<sup>rd</sup> analog input and 8 points of discrete output making 4 points of temperature measurement).

FUN 72 TP4	Convenient instruction proper to FB-2AJ(K)4/FB-2AH(T)4 temperature module	FUN 72 TP4
<ul style="list-style-type: none"> <li>● The selection of input span of FB-2AJ(K)4 temperature module can be 5V (500°C) (when jumper setting at the position of 5V) or 10V (1000°C)(when jumper setting at the position of 10V); the input polarity can be set as unipolar (U/B jumper setting at U) or bipolar (U/B jumper setting at B):           <p>When setting at 10V(1000°C) and unipolar, the range of measurement is 0°C~750°C (J-type) or 0°C~900°C (K-type)</p> <p>When setting at 5V(500°C) and unipolar, the range of measurement is 0°C~420°C (J-type) or 0°C~450°C (K-type)</p> <p>When setting at 10V(1000°C) and bipolar, the range of measurement is -200°C~750°C (J-type) or -200°C~900°C(K-type)</p> <p>When setting at 5V(500°C) and bipolar, the range of measurement is -200°C~420°C (J-type) or -200°C~450°C(K-type)</p> </li> <li>● The selection of input span of FB-2AH(T)4 temperature module can be 5V (when jumper setting at the position of 5V) or 10V (when jumper setting at the position of 10V); the input polarity is fixed for bipolar :           <p>When setting at 10V, the range of measurement is -49.8°C~146.6°C (DIN) or -50.7°C~149.2°C (JIS)</p> <p>When setting at 5V, the range of measurement is -12.3°C~83.6°C (DIN) or -12.5°C~85.1°C (JIS)</p> </li> <li>● FB-2AJ(K)4/FB-2AH(T)4 multiplexing temperature module occupies 3 points of analog input address and 8 points of discrete output address in physical;           <ul style="list-style-type: none"> <li>• when expension module with analog input will be installed after this kind of module, the analog address of which must be added 3;</li> <li>• when expension module with discrete output will be installed after this kind of module, the discrete output address of which must be added 8.</li> </ul> </li> <li>● Modules FB-2AJ(K)4/FB-2AH(T)4 can't be used together with module FB-8AD or FB-4AJ(K)xx.</li> <li>● For the selection of thermocouple, K-type thermocouple is recommended.</li> <li>● It is recommended to select 0~5V for the span and polarity of input if it meets the requirement.</li> <li>● Connect the "FG" terminal with the shielding of thermocouple if it is with for better measurement.</li> <li>● The "G" terminal must be connected to the safty earth ground of the power system.</li> </ul>		

## Measuring instruction proper to FB-2AJ(K)4/FB-2AH(T)4

FUN 72 TP4	Convenient instruction proper to FB-2AJ(K)4/FB-2AH(T)4 temperature module	FUN 72 TP4
<b>User guide for convenient instruction FUN72</b>		
FB-2AJ(K)4 temperature module:		
<ul style="list-style-type: none"> <li>● When execution control “EN”=1, this instruction will perform multiplexing temperature measurement and store the primitive value into R3968(TP0)~R3971(TP3) or R3972(TP4)~3975(TP7),···or R3996(TP28)~R3999(TP31); the value falls in 0~4095(unipolar) or -2048~2047 (bipolar). And then base on the setting of temperature sensor (Tp), input span and polarity (PI) of the temperature module to scale the primitive values to engineering values and store them to temperature measurement registers (TR+0 as the 1<sup>st</sup> point, ..., TR+3 as the 4<sup>th</sup> point).</li> </ul>		
FB-2AH(T)4 temperature module:		
<ul style="list-style-type: none"> <li>● When execution control “EN”=1, this instruction will perform multiplexing temperature measurement and base on the setting of temperature sensor (Tp), input span and polarity (PI) of the temperature module to scale the primitive values to engineering values and store them to temperature measurement registers (TR+0 as the 1<sup>st</sup> point, ..., TR+3 as the 4<sup>th</sup> point). Then scale the engineering values by the range of 0~4095 and store them into R3968(TP0)~R3971(TP3) or R3972(TP4)~3975(TP7),···or R3996(TP28)~R3999(TP31); the value falls in 0~4095.</li> <li>● When the setting of Tp, PI, Sm comes error, this instruction will not be performed and the output indication “ERR” will be ON.</li> <li>● When the sensor is K-type thermocouple (it needs FB-2AK4 module): <ul style="list-style-type: none"> <li>1.As the setting of input span and polarity is 0~10V, the range of measurement will be 0~900°C. When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>2.As the setting of input span and polarity is 0~5V, the range of measurement will be 0~450°C. When the display value is greater than 450°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>3.As the setting of input span and polarity is -10~10V, the range of measurement will be -200~900°C. When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>4.As the setting of input span and polarity is -5~5V, the range of measurement will be -200~450°C. When the display value is greater than 450°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> </ul> </li> <li>● When the sensor is J-type thermocouple (it needs FB-2AJ4 module): <ul style="list-style-type: none"> <li>1.As the setting of input span and polarity is 0~10V, the range of measurement will be 0~750°C. When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>2.As the setting of input span and polarity is 0~5V, the range of measurement will be 0~420°C. When the display value is greater than 450°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>3.As the setting of input span and polarity is -10~10V, the range of measurement will be -200~750°C. When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>4.As the setting of input span and polarity is -5~5V, the range of measurement will be -200~420°C. When the displayed temperature value is greater than 450°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> </ul> </li> <li>● When the sensor is RTD type of PT-100 (it needs FB-2AH4) or PT-1000 (it needs FB-2AT4): <ul style="list-style-type: none"> <li>1. As the setting of input span is -10~10V, the range of measurement will be -49.8°C~146.6°C (DIN) or -50.7°C~149.2°C (JIS)</li> <li>2. As the setting of input span is -10~10V, the range of measurement will be -12.3°C~83.6°C (DIN) or -12.5°C~85.1°C (JIS)</li> <li>3. When the display value is greater than 900.0°C, it means the line broken of the sensor and the output indication “ALM” will be ON.</li> </ul> </li> </ul>		
<p>Note: When there exists the line broken of the sensor, it can be told from the content of WR+0 working register, which tells the input point(s) of line broken.</p>		

FUN 72 TP4	Convenient instruction proper to FB-2AJ(K)4/FB-2AH(T)4 temperature module	FUN 72 TP4
<ul style="list-style-type: none"> <li>● Sm : Starting point of temperature measurement of this module. It must be the multiple of 4 , <math>0 \leq Sm \leq 28</math>.</li> <li>● Ym : Starting address of discrete output of this module for multiplexing temperature input; it takes 8 points of discrete output.</li> <li>● AR : Address of analog input (<math>3^{rd}</math>) for temperature measurement of this module.</li> <li>● TR : Starting register of the engineering value of temperature measurement, 4 registers in total. TR+0 stores the 1<sup>st</sup> temperature,..., TR+3 stores the 4<sup>th</sup> temperature.</li> <li>● WR : Starting of working register for this instruction. It takes 8 registers and can't be repeated in using. The content of WR+0 register indicates the status of the sensor which is line broken or not. Bit definition of WR+0 explained as follows: Bit0 =1 indicating that the 1<sup>st</sup> point of sensor is line broken; ...; Bit3=1 indicating that the 4<sup>th</sup> point of sensor is line broken. Registers WR+2~WR+7 are used by this instruction.</li> <li>● If it only needs to measure temperature, there should be a corresponding FUN72 instruction each for every temperature module to perform the measurement.</li> <li>● No matter the FUN72 is placed in main program or in sub-program, and whether the execution control "EN"=0 or 1, this instruction must be executed every scan.</li> </ul> <p><b>Explanation of specific registers for FUN72</b></p> <ul style="list-style-type: none"> <li>● R3968~R3999 : Registers storing the primitive temperature value. R3968 storing the 1<sup>st</sup> point, R3969 storing the 2<sup>nd</sup> point, etc. and R3999 storing the 32<sup>th</sup> point. The value is from 0~4095 (unipolar) or -2048~2047 (bipolar).</li> <li>● R4014 : Time interval between the measurement points while multiplexing. Which the user can set up. The unit is in mS and the default value is 500; it means it needs 500 mS to measure one point of temperature. This means the update rate of the temperature is 2 seconds (<math>500\text{mS} \times 4 = 2000\text{mS}</math>) When the value of R4014 is 250, it means it needs 250mS to measure one point of temperature. The update rate of the temperature is 1 second (<math>250\text{mS} \times 4 = 1000\text{mS}</math>) When the value of R4014 is 1000, it means it needs 1000mS to measure one point of temperature. The update rate of the temperature is 4 seconds (<math>1000\text{mS} \times 4 = 4000\text{mS}</math>) When the value of R4014 is 2000, it means it needs 2000mS to measure one point of temperature. The update rate of the temperature is 8 seconds (<math>2000\text{mS} \times 4 = 8000\text{mS}</math>)</li> <li>● R4015 : Times for the average of measurement, which can be set by the user. =0, no average; every acquired value is the measured value (default) =1, average of 2 times; the average on the acquired 2 times of values is the measured value. =2, average of 4 times; the average on the acquired 4 times of values is the measured value. =3, average of 8 times; the average on the acquired 8 times of values is the measured value. =4, average of 16 times; the average on the acquired 16 times of values is the measured value.</li> <li>● R4016 : The factor for linear scaling to calculate the engineering value of K-type thermocouple while in positive temperature; the default value is 248. The expression for engineering value is as follows: <math display="block">\text{Engineering value} = (\text{Primitive temperature value} \times R4016) / 1024 \text{ (Unipolar)}</math><math display="block">\text{Engineering value} = (\text{Primitive temperature value} \times 2 \times R4016) / 1024 \text{ (Bipolar)}</math> When there is a slight difference in measurement result between the standard meter and the FB-PLC's temperature module, if the user would like to use the value acquired by standard meter for correction, the user can tune the value of R4016 to get a better result in temperature measurement. This register provides fine tuning for positive temperature.</li> <li>● R4017 : The factor for linear scaling to calculate the engineering value of K-type thermocouple while in negative temperature; the default value is 286. The expression for engineering value is as follows: <math display="block">\text{Engineering value} = (\text{Primitive temperature value} \times R4017) / 1024 \text{ (-5~5V)}</math><math display="block">\text{Engineering value} = (\text{Primitive temperature value} \times 2 \times R4017) / 1024 \text{ (-10~10V)}</math> When there is a slight difference in measurement result between the standard meter and the FB-PLC's temperature module, if the user would like to use the value acquired by standard meter for correction, the user can tune the value of R4017 to get a better result in temperature measurement. This register provides fine tuning for negative temperature.</li> </ul>		

## Measuring instruction proper to FB-2AJ(K)4/FB-2AH(T)4

FUN 72 TP4	Convenient instruction proper to FB-2AJ(K)4/FB-2AH(T)4 temperature module	FUN 72 TP4
	<ul style="list-style-type: none"> <li>● R4018 : The factor for linear scaling to calculate the engineering value of J-type thermocouple while in positive temperature; the default value is 240. The expression for engineering value is as follows:  <math display="block">\text{Engineering value} = (\text{Primitive temperature value} \times R4018) / 1024 \text{ (Unipolar).}</math> <math display="block">\text{Engineering value} = (\text{Primitive temperature value} \times 2 \times R4018) / 1024 \text{ (Bipolar).}</math> When there is a slight difference in measurement result between the standard meter and the FB-PLC's temperature module, if the user would like to use the value acquired by standard meter for correction, the user can tune the value of R4018 to get a better result in temperature measurement. This register provides fine tuning for positive temperature.</li> <li>● R4019 : The factor for linear scaling to calculate the engineering value of J-type thermocouple while in negative temperature; the default value is 280. The expression for engineering value is as follows:  <math display="block">\text{Engineering value} = (\text{Primitive temperature value} \times R4019) / 1024 (-5 \sim 5V).</math> <math display="block">\text{Engineering value} = (\text{Primitive temperature value} \times 2 \times R4019) / 1024 (-10 \sim 10V).</math> When there is a slight difference in measurement result between the standard meter and the FB-PLC's temperature module, if the user would like to use the value acquired by standard meter for correction, the user can tune the value of R4019 to get a better result in temperature measurement. This register provides fine tuning for negative temperature.</li> <li>● R4020 : High Byte of R4020 to tell the alpha value of RTD, =0, <math>\alpha=0.00385</math> (DIN) =1, <math>\alpha=0.00392</math> (JIS) : Low Byte of R4020 to tell where the registers storing the wire resistance for compensation,  =1, the wire resistance for compensation for 3-wires RTD input storing in registers Rxxxx  =2, the wire resistance for compensation for 3-wires RTD input storing in registers Dxxxx The starting address of above mentioned registers is storing in R4021. The default of R4020 is 0001H.</li> <li>● R4021 : Storing the starting address of the registers to store the wire resistance for compensation for 3-wires RTD input; the default of R4021 is 8000, it means the starting register to store the wire resistance for compensation is R8000 by default. The unit of the resistance is <math>0.1\Omega</math>. While in long distance measurement and the accuracy will be affected by the wire resistance of the connection between the RTD sensor and temperature module, under such situation, the user has to measure the wire resistance of each loop and input them to the registers mentioned above; otherwise, forget these.</li> <li>● R4022 : The factor for linear scaling to calculate the engineering value of PT-100 ; the default value is 1024 The expression for engineering value is as follows:  <math display="block">\text{Engineering value} = (\text{Primitive temperature value} \times R4022) / 1024</math></li> <li>● R4023 : The factor for linear scaling to calculate the engineering value of PT-1000 ; the default is 1024 The expression for engineering value is as follows:  <math display="block">\text{Engineering value} = (\text{Primitive temperature value} \times R4023) / 1024</math> When it needs to do the calibration between the standard meter and the FB-PLC's temperature module, the user can tune the value of R4022 or R4023 to get a better result of measurement.</li> <li>● R4010 : Each bit of R4010 to tell the status of the sensor's installation. Bit0=1 means that 1<sup>st</sup> point of temperature sensor is installed. Bit1=1 means that 2<sup>nd</sup> point of temperature sensor is installed.  •  •  Bit15=1 means that 16<sup>th</sup> point of temperature sensor is installed.(The default of R4010 is FFFFH)</li> <li>● R4011 : Each bit of R4011 to tell the status of the sensor's installation. Bit0=1 means that 17<sup>th</sup> point of temperature sensor is installed. Bit1=1 means that 18<sup>th</sup> point of temperature sensor is installed.  •  •  Bit15=1 means that 32<sup>th</sup> point of temperature sensor is installed. (The default of R4011 is FFFFH)</li> <li>● When the temperature sensor is installed (the corresponding bit of R4010 or R4011 must be 1), the system will perform the line broken detection to the sensor. If there is line broken happened to the sensor, there will have the warning and the line broken value will be displayed.</li> <li>● When the temperature sensor is not installed (the corresponding bit of R4010 or R4011 must be 0), the system won't perform the line broken detection to the sensor and there will not have the warning; the temperature value will be displayed as 0.</li> <li>● Depends on the sensor's installation, the ladder program may control the corresponding bit of R4010 and R4011 to tell FUN72 to perform or not to perform the line broken detection.</li> </ul>	

FUN 72 TP4	Convenient instruction proper to FB-2AJ(K)4/FB-2AH(T)4 temperature module	FUN 72 TP4
<b>Program example 1</b> The main unit is FBx-28MC(A), the FB-2AK4 temperature module is attached to the main unit, and the FB-2AJ4 temperature module is attached to FB-2AK4 module. The setting of polarity and span are 0~10V for both of temperature modules.		
	<p>※ The analog input address for temperature measurement of FB-2AK4 is R3842.</p> <p>※ The analog input address for temperature measurement of FB-2AJ4 is R3845.</p>	

## Measuring instruction proper to FB-2AJ(K)4/FB-2AH(T)4

FUN 72 TP4	Convenient instruction proper to FB-2AJ(K)4/FB-2AH(T)4 temperature module	FUN 72 TP4
<b>Program example 2</b>		
	The main unit is FBx-28MC(A), the FB-2AH4 temperature module is attached to the main unit, and the FB-2AT4 temperature module is attached to FB-2AH4 module. The spans are setting at 10V for both of temperature modules. (The polarity is fixed at bipolar).	
	<p>※ The analog input address for temperature measurement of FB-2AH4 is R3842.</p> <p>※ The analog input address for temperature measurement of FB-2AT4 is R3845.</p>	
	<ul style="list-style-type: none"> <li>When M0=1, to measure the temperature of 1<sup>st</sup> (Sm=0) ~ 4<sup>th</sup> point of PT-100 RTD inputs and store the engineering values of measurement into R0 ~ R3; also, store the primitive values into R3968~R3971.</li> <li>When there is line broken in PT-100 RTD, M1 will be ON and the line broken value of this point will be displayed.</li> </ul>	<b>72.TP4</b> Tp : 2 Pl : 2 Sm : 0 Ym: Y 12 AR : R 3842 TR : R 0 WR: R 100  ERR— ALM——( ) M1
	<ul style="list-style-type: none"> <li>When M0=1, to measure the temperature of 5<sup>th</sup> (Sm=4) ~ 8<sup>th</sup> point of PT-1000 RTD inputs and store the engineering values of measurement into R4~R7; also, store the primitive values into R3972~R3975</li> <li>When there is line broken in PT-1000 RTD, M2 will be ON and the line broken value of this point will be displayed.</li> </ul>	<b>72.TP4</b> Tp : 3 Pl : 2 Sm : 4 Ym: Y 20 AR : R 3845 TR : R 4 WR: R 108  ERR— ALM——( ) M2
	<ul style="list-style-type: none"> <li>When M0=1, M1000~M1007 tells the status of line broken of corresponding sensor.</li> </ul>	<b>43.NBMV</b> S : R 100 Ns : 0 D : WM1000 Nd : 0  EN ——————  <b>43.NBMV</b> S : R 108 Ns : 0 D : WM1000 Nd : 1

FUN 72 TP4	Convenient instruction proper to FB-2AJ(K)4/FB-2AH(T)4 temperature module	FUN 72 TP4
<b>Program example 3</b> The main unit is FBx-40MC(A), and 4 modules of FB-2AK4 are attached. The setting of span and polarity are all at 0~5V.		
<ul style="list-style-type: none"> <li>The status of M800~M831 are controlled by the MMI or external inputs to tell the status of sensor's installation; if it has the sensor, perform line broken detection, and not to perform the check if it hasn't. (It needs the retentive function, so M800~M1399 are the better choice).</li> <li>When M0=1, to measure the temperature of 1<sup>st</sup> (Sm=0) ~ 4<sup>th</sup> point of K-type thermocouple inputs and store the engineering values of measurement into R0~R3; also, store the primitive values into R3968~R3971.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	<b>EN</b> <b>08D.MOV</b> S : WM 800 D : R 4010	<b>EN</b> <b>72.TP4</b> Tp : 0 Pl : 1 Sm : 0 Ym : Y 16 AR : R 3842 TR : R 0 WR : R 100
<ul style="list-style-type: none"> <li>When M0=1, to measure the temperature of 5<sup>th</sup> (Sm=4) ~ 8<sup>th</sup> point of K-type thermocouple inputs and store the engineering values of measurement into R4~R7; also, store the primitive values into R3972~R3975.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	<b>EN</b> <b>72.TP4</b> Tp : 0 Pl : 1 Sm : 4 Ym : Y 24 AR : R 3845 TR : R 4 WR : R 108	<b>EN</b> <b>72.TP4</b> Tp : 0 Pl : 1 Sm : 4 Ym : Y 24 AR : R 3845 TR : R 4 WR : R 108
<ul style="list-style-type: none"> <li>When M0=1, to measure the temperature of 9<sup>th</sup> (Sm=8) ~ 12<sup>th</sup> point of K-type thermocouple inputs and store the engineering values of measurement into R8~R11; also, store the primitive values into R3976~R3979.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	<b>EN</b> <b>72.TP4</b> Tp : 0 Pl : 1 Sm : 8 Ym : Y 32 AR : R 3848 TR : R 8 WR : R 116	<b>EN</b> <b>72.TP4</b> Tp : 0 Pl : 1 Sm : 8 Ym : Y 32 AR : R 3848 TR : R 8 WR : R 116
<ul style="list-style-type: none"> <li>When M0=1, to measure the temperature of 13<sup>th</sup> (Sm=12) ~ 16<sup>th</sup> point of K-type thermocouple inputs and store the engineering values of measurement into R12~R15; also, store the primitive values into R3980~R3983.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	<b>EN</b> <b>72.TP4</b> Tp : 0 Pl : 1 Sm : 12 Ym : Y 40 AR : R 3851 TR : R 12 WR : R 124	<b>EN</b> <b>72.TP4</b> Tp : 0 Pl : 1 Sm : 12 Ym : Y 40 AR : R 3851 TR : R 12 WR : R 124
<ul style="list-style-type: none"> <li>When M0=1, M1000~M1015 tells the line broken status of corresponding sensor.</li> </ul>	<b>EN</b> <b>43.NBMV</b> S : R 100 Ns : 0 D : WM 1000 Nd : 0	<b>EN</b> <b>43.NBMV</b> S : R 108 Ns : 0 D : WM 1000 Nd : 1
	<b>EN</b> <b>43.NBMV</b> S : R 116 Ns : 0 D : WM 1000 Nd : 2	<b>EN</b> <b>43.NBMV</b> S : R 124 Ns : 0 D : WM 1000 Nd : 3

## Measuring instruction proper to FB-2AJ(K)4/FB-2AH(T)4

FUN 72 TP4	Convenient instruction proper to FB-2AJ(K)4/FB-2AH(T)4 temperature module	FUN 72 TP4
<b>Program example 4</b> The main unit is FBx-40MC(A), and 4 modules of FB-2AH4 are attached. The spans are all setting at 5V (FB-2AH4 supports bipolar only).		
<ul style="list-style-type: none"> <li>The status of M800~M831 are controlled by the MMI or external inputs to tell the status of sensor's installation; if it has the sensor, perform line broken detection, and not to perform the check if it hasn't. (It needs the retentive function, so M800~M1399 are the better choice).</li> <li>When M0=1, to measure the temperature of 1<sup>st</sup> (Sm=0) ~4<sup>th</sup> point of PT-100 RTD inputs and store the engineering values of measurement into R0~R3; also, store the primitive values into R3968~R3971.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	08D.MOV EN S : WM 800 D : R 4010	72.TP4 EN Tp : 2 Pl : 3 Sm : 0 Ym: Y 16 AR : R 3842 TR : R 0 WR: R 100
<ul style="list-style-type: none"> <li>When M0=1, to measure the temperature of 5<sup>th</sup> (Sm=4) ~8<sup>th</sup> point of PT-100 RTD inputs and store the engineering values of measurement into R4~R7; also, store the primitive values into R3972~R3975.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	72.TP4 EN Tp : 2 Pl : 3 Sm : 4 Ym: Y 24 AR : R 3845 TR : R 4 WR: R 108	
<ul style="list-style-type: none"> <li>When M0=1, to measure the temperature of 9<sup>th</sup> (Sm=8) ~12<sup>th</sup> point of PT-100 RTD inputs and store the engineering values of measurement into R8~R11; also, store the primitive values into R3976~R3979.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	72.TP4 EN Tp : 2 Pl : 3 Sm : 8 Ym: Y 32 AR : R 3848 TR : R 8 WR: R 116	
<ul style="list-style-type: none"> <li>When M0=1, to measure the temperature of 13<sup>th</sup> (sm=12) ~16<sup>th</sup> point of PT-100 RTD inputs and store the engineering values of measurement into R12~R15; also, store the primitive values into R3980~R3983.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	72.TP4 EN Tp : 2 Pl : 3 Sm : 12 Ym: Y 40 AR : R 3851 TR : R 12 WR: R 124	
<ul style="list-style-type: none"> <li>When M0=1, M1000~M1015 tells the line broken status of corresponding sensor.</li> </ul>	43.NBMV EN S : R 100 Ns : 0 D : WM 1000 Nd : 0	43.NBMV EN S : R 108 Ns : 0 D : WM 1000 Nd : 1
	43.NBMV EN S : R 116 Ns : 0 D : WM 1000 Nd : 2	43.NBMV EN S : R 124 Ns : 0 D : WM 1000 Nd : 3

Temperature instruction proper to FB-2AJ(K)4/FB2AH(T)4

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB2AH(T)4	FUN 73 TSTC																																		
<p>Execution control—EN</p> <p>Heating/Cooling—H/C</p> <table border="1" style="margin-left: 10px; margin-top: 10px;"> <tr><td colspan="2">73.TSTC</td></tr> <tr><td>Tp :</td><td>-ERR—Parameter error</td></tr> <tr><td>Pl :</td><td>-AO0—Sensor line broken</td></tr> <tr><td>Sm :</td><td>-AO1—Warning indication</td></tr> <tr><td>Ym :</td><td></td></tr> <tr><td>AR :</td><td></td></tr> <tr><td>TR :</td><td></td></tr> <tr><td>Yh :</td><td></td></tr> <tr><td>Sh :</td><td></td></tr> <tr><td>Zh :</td><td></td></tr> <tr><td>Sv :</td><td></td></tr> <tr><td>Os :</td><td></td></tr> <tr><td>PR :</td><td></td></tr> <tr><td>IR :</td><td></td></tr> <tr><td>DR :</td><td></td></tr> <tr><td>OR :</td><td></td></tr> <tr><td>WR:</td><td></td></tr> </table>	73.TSTC		Tp :	-ERR—Parameter error	Pl :	-AO0—Sensor line broken	Sm :	-AO1—Warning indication	Ym :		AR :		TR :		Yh :		Sh :		Zh :		Sv :		Os :		PR :		IR :		DR :		OR :		WR:		<p>Tp : Type of sensor =0, K-type thermocouple =1, J-type thermocouple =2, PT-100 RTD =3, PT-1000 RTD</p> <p>PI : Setting of polarity and span =0, 0 ~10V (Unipolar) =1, 0 ~5V (Unipolar) =2, -10 ~10V (Bipolar) =3, -5 ~5V (Bipolar)</p> <p>Unipolar: U/B jumper set at U Bipolar: U/B jumper set at B Span : 5V/10V jumper setting</p> <p>Sm : Starting point of temperature measurement of this module. Sm=0, 4, 8....., 28</p> <p>Ym : Starting address of discrete output of this module for multiplexing temperature input; it takes 8 points. When expansion module with discrete output will be installed after the temperature module, the discrete output address of which must be added 8.</p> <p>AR : Address of analog input for temperature measurement of this module; which is the 3<sup>rd</sup> analog input. When expansion module with analog input will be installed after the temperature module, the analog address of which must be added 3.</p> <p>TR : Starting register of the engineering value of temperature measurement, 4 registers in total.</p> <p>Yh : Starting address of PID ON/OFF output; it takes Zh points.</p> <p>Sh : Starting point of PID control of this instruction; Sh = 0~31.</p> <p>Zh : Number of the PID control of this instruction; 1≤Zh≤32 and 1≤Sh+Zh≤32</p> <p>Sv : Starting register of the setpoint; it takes Zh registers.</p> <p>Os : Starting register of the in-zone offset; it takes Zh registers.</p> <p>PR : Starting register of the gain (Kc); it takes Zh registers.</p> <p>IR : Starting register of integral tuning constant (Ti); it takes Zh registers.</p> <p>DR : Starting register of derivative tuning constant (Td); it takes Zh registers.</p> <p>OR : Starting register of the PID analog output; it takes Zh registers.</p> <p>WR : Starting of working register for this instruction. It takes 17 registers and can't be repeated in using.</p>	<p>FUN 73 TSTC</p>
73.TSTC																																				
Tp :	-ERR—Parameter error																																			
Pl :	-AO0—Sensor line broken																																			
Sm :	-AO1—Warning indication																																			
Ym :																																				
AR :																																				
TR :																																				
Yh :																																				
Sh :																																				
Zh :																																				
Sv :																																				
Os :																																				
PR :																																				
IR :																																				
DR :																																				
OR :																																				
WR:																																				

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB-2AH(T)4	FUN 73 TSTC	
<b>Function guide and notifications</b>			
<ul style="list-style-type: none"> <li>● FUN73 Convenient instruction combines the temperature measurement with PID control and it is dedicated for the modules of FB-2AJ(K)4 and FB-2AH(T)4.</li> <li>● FB-2AJ(K)4/FB-2AH(T)4 multiplexing temperature module occupies 3 points of analog input address and 8 points of discrete output address in physical, more detail as followings: <ul style="list-style-type: none"> <li>● FB-2AJ4 provides 2 points of general purpose analog inputs (1<sup>st</sup> and 2<sup>nd</sup> analog input) and 4 points of J-type thermocouple inputs for temperature measurement (With the combination of 3<sup>rd</sup> analog input and 8 points of discrete output making 4 points of temperature measurement).</li> <li>● FB-2AK4 provides 2 points of general purpose analog inputs (1<sup>st</sup> and 2<sup>nd</sup> analog input) and 4 points of K-type thermocouple inputs for temperature measurement (With the combination of 3<sup>rd</sup> analog input and 8 points of discrete output making 4 points of temperature measurement).</li> <li>● FB-2AH4 provides 2 points of general purpose analog inputs (1<sup>st</sup> and 2<sup>nd</sup> analog input) and 4 points of 3-lines PT-100 RTD inputs for temperature measurement (With the combination of 3<sup>rd</sup> analog input and 8 points of discrete output making 4 points of temperature measurement).</li> <li>● FB-2AT4 provides 2 points of general purpose analog inputs (1<sup>st</sup> and 2<sup>nd</sup> analog input) and 4 points of 3-lines PT-1000 RTD inputs for temperature measurement (With the combination of 3<sup>rd</sup> analog input and 8 points of discrete output making 4 points of temperature measurement).</li> <li>● The selection of input span of FB-2AJ(K)4 temperature module can be 5V (500°C) (when jumper setting at the position of 5V) or 10V (1000°C)(when jumper setting at the position of 10V); the input polarity can be set as unipolar (U/B jumper setting at U) or bipolar (U/B jumper setting at B): <ul style="list-style-type: none"> <li>When setting at 10V(1000°C) and unipolar, the range of measurement is 0°C~750°C (J-type) or 0°C~900°C (K-type)</li> <li>When setting at 5V(500°C) and unipolar, the range of measurement is 0°C~420°C (J-type) or 0°C~450°C (K-type)</li> <li>When setting at 10V(1000°C) and bipolar, the range of measurement is -200°C~750°C (J-type) or -200°C~900°C(K-type)</li> <li>When setting at 5V(500°C) and bipolar, the range of measurement is -200°C~420°C (J-type) or -200°C~450°C(K-type)</li> </ul> </li> </ul> </li> <li>● The selection of input span of FB-2AH(T)4 temperature module can be 5V (when jumper setting at the position of 5V) or 10V (when jumper setting at the position of 10V); the input polarity is fixed for bipolar : <ul style="list-style-type: none"> <li>When setting at 10V, the range of measurement is -49.8°C~146.6°C (DIN) or -50.7°C~149.2°C (JIS)</li> <li>When setting at 5V, the range of measurement is -12.3°C~83.6°C (DIN) or -12.5°C~85.1°C (JIS)</li> </ul> </li> <li>● FB-2AJ(K)4/FB-2AH(T)4 multiplexing temperature module occupies 3 points of analog input address and 8 points of discrete output address in physical; <ul style="list-style-type: none"> <li>• when expansion module with analog input will be installed after this kind of module, the analog address of which must be added 3;</li> <li>• when expansion module with discrete output will be installed after this kind of module, the discrete output address of which must be added 8.</li> </ul> </li> <li>● Modules FB-2AJ(K)4/FB-2AH(T)4 can't be used together with module FB-8AD or FB-4AJ(K)xx.</li> <li>● For the selection of thermocouple, K-type thermocouple is recommended.</li> <li>● It is recommended to select 0~5V for the span and polarity of input if it meets the requirement.</li> <li>● Connect the "FG" terminal with the shielding of thermocouple if it is with for better measurement.</li> <li>● The "G<sup>⊕</sup>" terminal must be connected to the safty earth ground of the power system.</li> </ul>			

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB-2AH(T)4	FUN 73 TSTC
<ul style="list-style-type: none"> <li>● Fun73 instruction employs the multiplexing temperature module [ FB-2AJ(K)4/FB-2AH(T)4 ] to get the current value of temperature and let it be as so called Process Variable (PV); after the calculation of software PID expression, it will respond the error with an output signal according to the setting of Set Point (SP),the error's integral and the rate of change of the process variable. Through the closed loop operation, the steady state of the process may be expected.</li> <li>● Convert the output of PID calculation to be the time proportional on/off (PWM) output, and via transistor output to control the SSR for heating or cooling process; this is a good performance and very low cost solution.</li> <li>● Through the analog output module (D/A module), the output of PID calculation may control the SCR or proportional valve to get more precise process control.</li> <li>● Digitized PID expression is as follows:</li> </ul> $M_n = [K_c \times E_n] + \sum_{0}^{n} [K_c \times T_i \times T_s \times E_n] - [K_c \times T_d \times (P V_{n-1} - P V_n) / T_s]$ <p>Where,</p> <p>M<sub>n</sub> : Output at time "n".</p> <p>K<sub>c</sub> : Gain (Range: 1~999 ; Pb=100(%) / K<sub>c</sub>)</p> <p>T<sub>i</sub> : Integral tuning constant (Range:0~999, equivalent to 0.00~9.99 Repeat/Minute)</p> <p>T<sub>d</sub> : Derivative tuning constant (Range:0~999, equivalent to 0.00~9.99 Minute)</p> <p>PV<sub>n</sub>: Process variable at time "n"</p> <p>PV<sub>n-1</sub>: Process variable when loop was last solved</p> <p>E<sub>n</sub> : Error at time "n" ; E= SP – PV<sub>n</sub></p> <p>T<sub>s</sub> : Solution interval for PID calculation (Valid value are 10, 20, 40, 80 ;the unit is in 0.1Sec)</p>		

## Temperature instruction proper to FB-2AJ(K)4/ FB-2AH(T)4

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB-2AH(T)4	FUN 73 TSTC
<b>Principle of PID parameter adjustment</b>		
<ul style="list-style-type: none"> <li>As the gain (Kc) adjustment getting larger, the larger the proportional contribution to the output. This can obtain a sensitive and rapid control reaction. However, when the gain is too large, it may cause oscillation. Do the best to adjust "Kc" larger (but not to the extent of making oscillation), which could increase the process reaction and reduce the steady state error.</li> <li>Integral item may be used to eliminate the steady state error. The larger the number (Ti, integral tuning constant), the larger the integral contribution to the output. When there is steady state error, adjust the "Ti" larger to decrease the error. When the "Ti" = 0, the integral item makes no contribution to the output. For exa, if the reset time is 6 minutes, Ti=100/6=17 ; if the integral time is 5 minutes, Ti=100/5=20.</li> <li>Derivative item may be used to make the process smoother and not too over shoot. The larger the number (Td, derivative tuning constant), the larger the derivative contribution to the output. When there is too over shoot, adjust the "Td" larger to decrease the amount of over shoot. When the "Td" = 0, the derivative item makes no contribution to the output. For exa, if the rate time is 1 minute, then the Td = 100; if the differential time is 2 minute, then the Td = 200.</li> <li>Properly adjust the PID parameters can obtain an excellent result for temperature control.</li> <li>The default of gain value (Kc) is as follows: <ul style="list-style-type: none"> <li>When the setting of span and polarity of the module is 0~10V, the default of gain (Kc) is 60.</li> <li>When the setting of span and polarity of the module is 0~5V, the default of gain (Kc) is 30.</li> <li>When the setting of span and polarity of the module is -10~10V, the default of gain (Kc) is 120.</li> <li>When the setting of span and polarity of the module is -5~5V, the default of gain (Kc) is 60.</li> </ul> </li> <li>The default of integral tuning constant is 17, it mens the reset time is 6 minutes (Ti=100/6=17).</li> <li>The default of derivative tuning constant is 100, it means the rate time is 1 minutes (Td=100).</li> </ul>		

### User guide to Convenient instruction FUN73

#### FB-2AJ(K)4 temperature module:

- When execution control "EN"=1, this instruction will perform multiplexing temperature measurement and store the primitive value into R3968(TP0)~R3971(TP3) or R3972(TP4)~3975(TP7),…or R3996(TP28)~R3999(TP31); the value falls in 0~4095(unipolar) or -2048~2047 (bipolar). And then base on the setting of temperature sensor (Tp), input span and polarity (PI) of the temperature module to scale the primitive values to engineering values and store them to temperature measurement registers (TR+0 as the 1<sup>st</sup> point, ..., TR+3 as the 4<sup>th</sup> point).

#### FB-2AH(T)4 temperature module:

- When execution control "EN"=1, this instruction will perform multiplexing temperature measurement and base on the setting of temperature sensor (Tp), input span and polarity (PI) of the temperature module to scale the primitive values to engineering values and store them to temperature measurement registers (TR+0 as the 1<sup>st</sup> point, ..., TR+3 as the 4<sup>th</sup> point). Then scale the engineering values by the range of 0~4095 and store them into R3968(TP0)~R3971(TP3) or R3972(TP4)~3975(TP7),…or R3996(TP28)~R3999(TP31); the value falls in 0~4095.
- When the setting of Tp, PI, Sm comes error, this instruction will not be performed and the output indication "ERR" will be ON.
- When the sensor is K-type thermocouple (it needs FB-2AK4 module):
  - As the setting of input span and polarity is 0~10V, the range of measurement will be 0~900°C.  
When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication "AO0" will be ON.
  - As the setting of input span and polarity is 0~5V, the range of measurement will be 0~450°C.  
When the display value is greater than 450°C, it means the line broken of the thermocouple and the output indication "AO0" will be ON.
  - As the setting of input span and polarity is -10~10V, the range of measurement will be -200~900°C.  
When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication "AO0" will be ON.
  - As the setting of input span and polarity is -5~5V, the range of measurement will be -200~450°C.  
When the display value is greater than 450°C, it means the line broken of the thermocouple and the output indication "AO0" will be ON.

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB-2AH(T)4	FUN 73 TSTC
<ul style="list-style-type: none"> <li>● When the sensor is J-type thermocouple (it needs FB-2AJ4 module):           <ol style="list-style-type: none"> <li>1. As the setting of input span and polarity is 0~10V, the range of measurement will be 0~750°C. When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication "AO0" will be ON.</li> <li>2. As the setting of input span and polarity is 0~5V, the range of measurement will be 0~420°C. When the display value is greater than 450°C, it means the line broken of the thermocouple and the output indication "AO0" will be ON.</li> <li>3. As the setting of input span and polarity is -10~10V, the range of measurement will be -200~750°C. When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication "AO0" will be ON.</li> <li>4. As the setting of input span and polarity is -5~5V, the range of measurement will be -200~420°C. When the displayed temperature value is greater than 450°C, it means the line broken of the thermocouple and the output indication "AO0" will be ON.</li> </ol> </li> <li>● When the sensor is RTD type of PT-100 (it needs FB-2AH4) or PT-1000 (it needs FB-2AT4):           <ol style="list-style-type: none"> <li>1. As the setting of input span is -10~10V, the range of measurement will be -49.8°C~146.6°C (DIN) or -50.7°C~149.2°C (JIS)</li> <li>2. As the setting of input span is -10~10V, the range of measurement will be -12.3°C~83.6°C (DIN) or -12.5°C~85.1°C (JIS)</li> <li>3. When the display value is greater than 900.0°C, it means the line broken of the sensor and the output indication "AO0" will be ON.</li> </ol> </li> </ul> <p>Note: When there exists the line broken of the sensor, it can be told from the content of WR+0 working register which tells the input point(s) of line broken.</p> <ul style="list-style-type: none"> <li>● Sm: Starting point of temperature measurement of this module. It must be the multiple of 4 , <math>0 \leq Sm \leq 28</math>.</li> <li>● Ym: Starting address of discrete output of this module for multiplexing temperature input; it takes 8 points of discrete output.</li> <li>● AR : Address of analog input (3<sup>rd</sup>) for temperature measurement of this module.</li> <li>● TR : Starting register of the engineering value of temperature measurement, 4 registers in total. TR+0 stores the 1<sup>st</sup> temperature,..., TR+3 stores the 4<sup>th</sup> temperature.</li> <li>● PID operation will begin after FUN73 has measured the temperature of every point.</li> <li>● When execution control "EN" = 1, it depends on the input status of H/C for PID operation to make heating (H/C=1) or cooling (H/C=0) control. The current values of measured temperature are through the multiplexing temperature module [ FB-2AJ(K)4/FB-2AH(T)4 ] to get ; the set points of desired temperature are stored in the registers starting from Sv. With the calculation of software PID expression, it will respond the error with an output signal according to the setting of setpoint, the error's integral and the rate of change of the process variable. Convert the output of PID calculation to be the time proportional on/off (PWM) output, and via transistor output to control the SSR for heating or cooling process; where there is a good performance and very low cost solution. It may also apply the output of PID calculation (stored in registers starting from OR), by way of D/A analog output module, to control SCR or proportional valve, so as to get more precise process control.</li> <li>● When Sh, Zh setting error, this instruction will not be executed and the instruction output "ERR" will be ON.</li> <li>● This instruction compares the current value with the set point to check whether the current temperature falls within deviation range (stored in register starting from Os). If it falls in the deviation range, it will set the in-zone bit of that point to be ON; if not, clear the in-zone bit of that point to be OFF, and make instruction output "AO1" ON.</li> <li>● In the mean time, this instruction will also check whether highest temperature warning (the register for the set point of highest temperature warning is R4008). When successively scanning for ten times the current values of measured temperature are all higher than or equal to the highest warning set point, the warning bit will set to be ON and instruction output "AO1" will be on. This can avoid the safety problem aroused from temperature out of control, in case the SSR or heating circuit becomes short.</li> <li>● This instruction can also detect the unable to heat problem resulting from the SSR or heating circuit runs open, or the obsolete heating band. When output of temperature control turns to be large power (set in R4006 register) successively in a certain time (set in R4007 register), and can not make current temperature fall in desired range, the warning bit will set to be ON and instruction output "AO1" will be ON.</li> </ul>		

## Temperature instruction proper to FB-2AJ(K)4/ FB-2AH(T)4

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB-2AH(T)4	FUN 73 TSTC
<ul style="list-style-type: none"> <li>● Yh : Starting address of PID ON/OFF output; it takes Zh points</li> <li>● Sh : Starting point of PID control of this instruction; where <math>0 \leq Sh \leq 31</math>.</li> <li>● Zh : Number of the PID control of this instruction; where <math>1 \leq Zh \leq 32</math> and <math>1 \leq Sh+Zh \leq 32</math></li> <li>● Sv : Starting register of the setpoint; it takes Zh registers.</li> <li>● Os : Starting register of the in-zone offset; it takes Zh registers.</li> <li>● PR : Starting register of the gain (Kc); it takes Zh registers.</li> <li>● IR : Starting register of integral tuning constant (Ti); it takes Zh registers</li> <li>● DR : Starting register of derivative tuning constant (Td); it takes Zh registers.</li> <li>● OR : Starting register of the PID analog output; it takes Zh registers.</li> <li>● WR : Starting of working register for this instruction. It takes 17 registers and can't be repeated in using.            The content of WR+0 register indicates the status of the sensor which is line broken or not.            Bit definition of WR+0 explained as follows:            Bit0=1 indicating that the Sm+0 point of sensor is line broken...            Bit3=1 indicating that the Sm+3 point of sensor is line broken.            The content of the two registers WR+8 and WR+9 indicating that whether the current temperature falls within the deviation range (stored in registers starting from Os). If it falls in the deviation range, the in-zone bit of that point will be set ON; if not, the in-zone bit of that point will be cleared OFF.            Bit definition of WR+8 explained as follows:            Bit0=1, it represents that the temperature of the Sh+0 point is in-zone...            Bit15=1, it represents that the temperature of the Sh+15 point is in-zone.            Bit definition of WR+9 explained as follows:            Bit0=1, it represents that the temperature of the Sh+16 point is in-zone...            Bit15=1, it represents that the temperature of 32th point is in-zone.            The content of the two registers WR+10 and WR+11 are the warning bit registers, they indicate that whether there exists the highest temperature warning or heating circuit opened.            Bit definition of WR+10 explained as follows:            Bit0=1, it means that there exists the highest warning or heating circuit opened at the Sh+0 point...            Bit15=1, it means that there exists the highest warning or heating circuit opened at the Sh+15 point.            Bit definition of WR+11 explained as follows:            Bit0=1, it means that there exists the highest warning or heating circuit opened at the Sh+16 point...            Bit15=1, it means that there exists the highest warning or heating circuit opened at the 32th point.            Registers of WR+2~WR+7 and WR+12~WR+16 are used by this instruction.         </li> <li>● This instruction can only be used to perform heating or cooling control of positive temperature while the sensor is the thermocouple.</li> <li>● Whether the FUN73 is placed in main or sub program and no matter the execution control "EN"=0 or 1, this instruction must be executed at every scan.</li> </ul>		

### Specific registers related to instruction of FUN73

- R4014 : Time interval between the measurement points while multiplexing. Which the user can set up. The unit is in mS and the default value is 500; it means it needs 500mS to measure one point of temperature. This means the update rate of the temperature is 2 seconds ( $500\text{mS} \times 4 = 2000\text{mS}$ ). When the value of R4014 is 250, it means it needs 250mS to measure one point of temperature; the update rate of the temperature is 1 second ( $250\text{mS} \times 4 = 1000\text{mS}$ ). When the value of R4014 is 1000, it means it needs 1000mS to measure one point of temperature; the update rate of the temperature is 4 second ( $1000\text{mS} \times 4 = 4000\text{mS}$ ). When the value of R4014 is 2000, it means it needs 2000mS to measure one point of temperature; the update rate of the temperature is 4 second ( $2000\text{mS} \times 4 = 8000\text{mS}$ ).

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB-2AH(T)4	FUN 73 TSTC
<ul style="list-style-type: none"> <li>● R4015 : Times for the average of measurement, which can be set by the user.           <ul style="list-style-type: none"> <li>=0, no average ; every acquired value is the measured value (default)</li> <li>=1, average of 2 times; the average on the acquired 2 times of values is the measured value.</li> <li>=2, average of 4 times; the average on the acquired 4 times of values is the measured value.</li> <li>=3, average of 8 times; the average on the acquired 8 times of values is the measured value.</li> <li>=4, average of 16 times; the average on the acquired 16 times of values is the measured value.</li> </ul> </li> <li>● R4016 : The factor for linear scaling to calculate the engineering value of K-type thermocouple while in positive temperature; the default value is 248. The expression for engineering value is as follows:            Engineering unit temperature value = (Original temperature value ×R4016) /1024 (Unipolar).            Engineering unit temperature value = (Original temperature value ×2×R4016) /1024 (Bipolar).         </li> <p>When there is a slight difference in measurement result between the standard meter and the FB-PLC's temperature module, if the user would like to use the value acquired by standard meter for correction, the user can tune the value of R4016 to get a better result in temperature measurement. This register provides fine tuning for positive temperature.</p> <li>● R4017 : The factor for linear scaling to calculate the engineering value of K-type thermocouple while in negative temperature; the default value is 286. The expression for engineering value is as follows:            Engineering value = (Primitive temperature value ×R4017) /1024 (-5~5V).            Engineering value = (Primitive temperature value ×2×R4017) /1024 (-10~10V).         </li> <p>When there is a slight difference in measurement result between the standard meter and the FB-PLC's temperature module, if the user would like to use the value acquired by standard meter for correction, the user can tune the value of R4017 to get a better result in temperature measurement. This register provides fine tuning for negative temperature.</p> <li>● R4018 : The factor for linear scaling to calculate the engineering value of J-type thermocouple while in positive temperature; the default value is 240.            The expression for engineering value is as follows:            Engineering value = (Primitive temperature value ×R4018) /1024 (Unipolar).            Engineering value = (Primitive temperature value ×2×R4018) /1024 (Bipolar).         </li> <p>When there is a slight difference in measurement result between the standard meter and the FB-PLC's temperature module, if the user would like to use the value acquired by standard meter for correction, the user can tune the value of R4018 to get a better result in temperature measurement. This register provides fine tuning for positive temperature.</p> <li>● R4019 : The factor for linear scaling to calculate the engineering value of J-type thermocouple while in negative temperature; the default value is 280.            The expression for engineering value is as follows:            Engineering value = (Primitive temperature value ×R4019) /1024 (-5~5V).            Engineering value = (Primitive temperature value ×2×R4019) /1024 (-10~10V).         </li> <p>When there is a slight difference in measurement result between the standard meter and the FB-PLC's temperature module, if the user would like to use the value acquired by standard meter for correction, the user can tune the value of R4019 to get a better result in temperature measurement. This register provides fine tuning for negative temperature.</p> <li>● R4020 : High Byte of R4020 to tell the alpha value of RTD, =0, <math>\alpha=0.00385</math> (DIN) =1, <math>\alpha=0.00392</math> (JIS)       <ul style="list-style-type: none"> <li>: Low Byte of R4020 to tell where the registers storing the wire resistance for compensation,</li> <li>=1, the wire resistance for compensation for 3-wires RTD input storing in registers Rxxxx</li> <li>=2, the wire resistance for compensation for 3-wires RTD input storing in registers Dxxxx</li> </ul> <p>The starting address of above mentioned registers is storing in R4021 .        The default of R4020 is 0001H.</p> </li> <li>● R4021: Storing the starting address of the registers to store the wire resistance for compensation for 3-wires RTD input; the default of R4021 is 8000, it means the starting register to store the wire resistance for compensation is R8000 by default. The unit of the resistance is <math>0.1\Omega</math>. While in long distance measurement and the accuracy will be affected by the wire resistance of the connection between the RTD sensor and temperature module, under such situation, the user has to measure the wire resistance of each loop and input them to the registers mentioned above; otherwise, forget these.</li> <li>● R4022 : The factor for linear scaling to calculate the engineering value of PT-100 ; the default value is 1024            The expression for engineering value is as follows:            Engineering value = (Primitive temperature value ×R4022) /1024</li> <li>● R4023 : The factor for linear scaling to calculate the engineering value of PT-1000 ; the default is 1024            The expression for engineering value is as follows:            Engineering value = (Primitive temperature value ×R4023) /1024  <p>When it needs to do the calibration between the standard meter and the FB-PLC's temperature module, the user can tune the value of R4022 or R4023 to get a better result of measurement.</p> </li> </ul>		

## Temperature instruction proper to FB-2AJ(K)4/ FB-2AH(T)4

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB-2AH(T)4	FUN 73 TSTC
<ul style="list-style-type: none"> <li>● R4010 : Each bit of R4010Bit15=1 means that to tell the status of the sensor's installation.            Bit0=1 means that 1<sup>st</sup> point of temperature sensor is installed.            Bit1=1 means that 2<sup>nd</sup> point of temperature sensor is installed.            . . .            Bit15=1 means that 16<sup>th</sup> point of temperature sensor is installed.            (The default of R4010 is FFFFH)</li> <li>● R4011: R4011 : Each bit of R4011 to tell the status of the sensor's installation.            Bit0=1 means that 17<sup>th</sup> point of temperature sensor is installed.            Bit1=1 means that 18<sup>th</sup> point of temperature sensor is installed.            . . .            Bit15=1 means that 32<sup>nd</sup> point of temperature sensor is installed.            (The default of R4011 is FFFFH)</li> <li>● When the temperature sensor is installed (the corresponding bit of R4010 or R4011 must be 1), the system will perform the line broken detection to the sensor. If there is line broken happened to the sensor, there will have the warning and the line broken value will be displayed.</li> <li>● When the temperature sensor is not installed (the corresponding bit of R4010 or R4011 must be 0), the system won't perform the line broken detection to the sensor and there will not have the warning; the temperature value will be displayed as 0.</li> <li>● Depends on the sensor's installation , the ladder program may control the corresponding bit of R4010 and R4011 to perform or not to perform the line broken detection.</li> <li>● R4005 : The content of Low Byte to define the solution interval between PID calculation            =0, perform the PID calculation every 2 seconds (System default).            =1, perform the PID calculation every 4 seconds.            =2, perform the PID calculation every 8 seconds.            ≥3, perform the PID calculation every 1 second. (R4014 must be 250 to make sense )            : The content of High Byte to define the cycle time of PID ON/OFF (PWM) output.            =0 , PWM cycle time is 2 seconds (system default)            =1 , PWM cycle time is 4 seconds.            =2 , PWM cycle time is 8 seconds.            ≥3 , PWM cycle time is 1 second.</li> </ul> <p>Note 1 : When changing the value of R4005, the execution control "EN" of FUN73 must be set at 0. The next time when execution control "EN" =1, it will base on the latest set point to perform the PID calculation.</p> <p>Note 2 : The smaller the cycle time of PWM, the more even can it perform the heating. However, the error caused by the PLC scan time will also become greater. For the best control, it can base on the scan time of PLC to adjust the solution interval of PID calculation and the PWM cycle time.</p> <ul style="list-style-type: none"> <li>● R4006 : The setting point of large power output detection for SSR or heating circuit opened, or heating band obsolete. The unit is in % and the setting range falls in 80~100(%); system default is 90(%).</li> <li>● R4007 : The setting time to detect the continuing duration of large power output while SSR or heating circuit opened, or heating band obsolete. The unit is in second and the setting range falls in 300~65535 (seconds); system default is 600 (seconds).</li> <li>● R4008 : The setting point of highest temperature warning for SSR, or heating circuit short detection. The unit is in degree and the setting range falls in 50~65535; system default is 350 (degrees).</li> <li>● R4012 : Each bit of R4012 to tell the need of PID temperature control.            Bit0=1 means that 1<sup>st</sup> point needs PID temperature control.            Bit1=1 means that 2<sup>nd</sup> point needs PID temperature control.            . . .            16<sup>th</sup> point needs PID temperature control.            (The default of R4012 is FFFFH)</li> <li>● R4013 : Each bit of R4013 to tell the need of PID temperature control.            Bit0=1 means that 17<sup>th</sup> point needs PID temperature control.            Bit1=1 means that 18<sup>th</sup> point needs PID temperature control.            . . .            Bit15=1 means that 32<sup>nd</sup> point needs PID temperature control.            (The default of R4013 is FFFFH)</li> <li>● While execution control "EN"=1 and the corresponding bit of PID control of that point is ON (corresponding bit of R4012 or R4013 must be 1), the FUN73 instruction will perform the PID operation and respond to the calculation with the output signal.</li> <li>● While execution control "EN"=1 and the corresponding bit of PID control of that point is OFF (corresponding bit of R4012 or R4013 must be 0), the FUN73 will not perform the PID operation and the output of that point will be OFF.</li> <li>● The ladder program may control the corresponding bit of R4012 and R4013 to tell the FUN73 to perform or not to perform the PID control, and it needs only one FUN73 instruction. (The temperature module must be identical in sensor type and the setting of input span and polarity must be the same.)</li> </ul>		

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB-2AH(T)4	FUN 73 TSTC
<b>Program example 1</b> The main unit is FBx-40MC(A), and 4 temperature modules of FB-2AK4 are attached. The settings of input span and polarity are all at 0~10V.		
*** It takes only one FUN73 instruction to perform 16 points of PID temperature control when the temperature modules are identical in sensor type and the settings of input span and polarity are the same.		
*** When performing the FUN73 instruction of the first time, the system will automatically assign to each point its system default of gain (Kc), integral tuning constant (Ti), and derivative tuning constant (Td), etc. The user may change the settings if necessary.		
	<ul style="list-style-type: none"> <li>The status of M800~M831 are controlled by the MMI or external inputs to tell the status of sensor's installation; if it has the sensor, perform line broken detection, and not to perform the check if it hasn't. (It needs the retentive function, so M800~M1399 are the better choice).</li> <li>When temperature sensor installed (the corresponding bit of R4010 or R4011 is 1) and there is line broken of the sensor, the line broken value of that point will be displayed.</li> <li>When temperature sensor is not installed (the corresponding bit of R4010 or R4011 is 0), there will not perform the line broken detection; the temperature of that point is displayed 0.</li> </ul>	
	<ul style="list-style-type: none"> <li>The status of M832~M863 are controlled by the MMI or external inputs to tell whether it needs the PID control of the corresponding point; perform the PID operation when the bit is ON, and not to perform if it is OFF (It needs the retentive function, so M800~M1399 are the better choice).</li> <li>When temperature control bit is ON (the corresponding bit of R4012 or R4013 is 1), FUN73 performs the PID operation of that point to obtain a suitable output signal.</li> <li>When temperature control bit is OFF (the corresponding bit of R4012 or R4013 is 0), FUN73 will not perform the PID operation of that point and output will be OFF.</li> </ul>	
M3	<ul style="list-style-type: none"> <li>When M3=ON, to measure the temperature of 13<sup>th</sup> (Sm=12) ~16<sup>th</sup> point of K-type thermocouple inputs and store the engineering values of measurement into R12~R15; also, store the primitive values into R3980~R3983.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	72.TP4 Tp : 0 Pl : 0 Sm : 12 Ym: Y 40 AR: R 3851 TR : R 12 WR: R 240 ERR— ALM—
M3	<ul style="list-style-type: none"> <li>When M3=ON, to measure the temperature of 9<sup>th</sup> (Sm=8) ~12<sup>th</sup> point of K-type thermocouple inputs and store the engineering values of measurement into R8~R11; also, store the primitive values into R3976~R3979.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	72.TP4 Tp : 0 Pl : 0 Sm : 8 Ym: Y 32 AR: R 3848 TR : R 8 WR: R 230 ERR— ALM—
M3	<ul style="list-style-type: none"> <li>When M3=ON, to measure the temperature of 5<sup>th</sup> (Sm=4) ~8<sup>th</sup> point of K-type thermocouple inputs and store the engineering values of measurement into R4~R7; also, store the primitive values into R3972~R3975.</li> <li>When there is line broken of the sensor, the value of line broken will be displayed.</li> </ul>	72.TP4 Tp : 0 Pl : 0 Sm : 4 Ym: Y 24 AR: R 3845 TR : R 4 WR: R 220 ERR— ALM—

## Temperature instruction proper to FB-2AJ(K)4/ FB-2AH(T)4

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB-2AH(T)4	FUN 73 TSTC
*When total control points(Zh) of the FUN73 are greater than 4, it must be Sh≥Sm and the registers storing the current values (starting from TR) must be continuous.		
<ul style="list-style-type: none"> <li>When M3=ON, to measure the temperature of 1<sup>st</sup> (Sm=0) ~ 4<sup>th</sup> point of K-type thermocouple inputs and store the engineering values of measurement into R0~R3 and store the original values into R3968~R3971.</li> <li>When completing the measurement of 1<sup>st</sup> ~ 16<sup>th</sup> point, it will perform the PID heating control of 16 (Zh) points from 1<sup>st</sup> (Sh=0) point to the 16<sup>th</sup> point.</li> <li>R0~R15 : Registers of current engineering value .</li> <li>Y48~Y63 : PID ON/OFF ( PWM ) output; it must be the transistor output.</li> <li>R100~R115 : Registers of set point.</li> <li>R120~R135 : Registers of deviation zone, it determines whether temperature falls in setting range. E.g. Set point is 200 and deviation zone is 5, then 195≤ Current value ≤ 205 means the temperature is in zone.</li> <li>R140~R155 : Setting point of gain (Kc).</li> <li>R160~R175 : Setting point of integral tuning constant (Ti)</li> <li>R180~R195 : Setting point of derivative tuning constant (Td).</li> <li>R300~R315 : Output of PID calculation (value from 0~4095).</li> <li>R200~R216 : Working registers.</li> </ul>	<b>73.TSTC</b> Tp : 0 Pl : 0 Sm : 0 Ym: Y16 AR: R3842 TR: R0 Yh : Y48 Sh : 0 Zh : 16 Sv : R 100 Os : R 120 PR : R 140 IR : R 160 DR : R 180 OR : R 300 WR: R 200	ERR— AO0— AO1—
<ul style="list-style-type: none"> <li>When M3=ON, M1000 ~ M1015 tells the line broken status of corresponding sensor.</li> </ul>	<b>43.NBMV</b> S : R 200 Ns : 0 D : WM1000 Nd : 0	
<ul style="list-style-type: none"> <li>When M3=ON, M1016 ~ M1031 tells the warning status of highest temperature warning or the heating circuit opened.</li> <li>M1032 ~ M1047 tells the status of temperature in zone.</li> </ul>	<b>43.NBMV</b> S : R 220 Ns : 0 D : WM1000 Nd : 1	
<ul style="list-style-type: none"> <li>When M3=ON, M1016 ~ M1031 tells the warning status of highest temperature warning or the heating circuit opened.</li> <li>M1032 ~ M1047 tells the status of temperature in zone.</li> </ul>	<b>43.NBMV</b> S : R 230 Ns : 0 D : WM1000 Nd : 2	
<ul style="list-style-type: none"> <li>When M3=ON, M1016 ~ M1031 tells the warning status of highest temperature warning or the heating circuit opened.</li> <li>M1032 ~ M1047 tells the status of temperature in zone.</li> </ul>	<b>43.NBMV</b> S : R 240 Ns : 0 D : WM1000 Nd : 3	
<ul style="list-style-type: none"> <li>When M3=ON, M1016 ~ M1031 tells the warning status of highest temperature warning or the heating circuit opened.</li> <li>M1032 ~ M1047 tells the status of temperature in zone.</li> </ul>	<b>08.MOV</b> S : R 210 D : WM1016	
<ul style="list-style-type: none"> <li>When M3=ON, M1016 ~ M1031 tells the warning status of highest temperature warning or the heating circuit opened.</li> <li>M1032 ~ M1047 tells the status of temperature in zone.</li> </ul>	<b>08.MOV</b> S : R 208 D : WM1032	

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB2AH(T)4	FUN 73 TSTC
<b>Program example 2</b> The main unit is FBx-40MC(A), and 4 temperature modules of FB-2AH4 are attached. The settings of input span are all at 5V (the polarity are fixed at bipolar).		
*** It takes only one FUN73 instruction to perform 16 points of PID temperature control when the temperature modules are identical in sensor type and the settings of input span and polarity are the same.		
*** When performing the FUN73 instruction of the first time, the system will automatically assign to each point its system default of gain (Kc), integral tuning constant (Ti), and derivative tuning constant (Td), etc. The user may change the settings if necessary.		
<ul style="list-style-type: none"> <li>The status of M800~M831 are controlled by the MMI or external inputs to tell the status of sensor's installation; if it has the sensor, perform line broken detection, and not to perform the check if it hasn't. (It needs the retentive function, so M800~M1399 are the better choice).</li> <li>When temperature sensor installed (the corresponding bit of R4010 or R4011 is 1) and there is line broken of the sensor, the line broken value of that point will be displayed.</li> <li>When temperature sensor is not installed (the corresponding bit of R4010 or R4011 is 0), there will not perform the line broken detection; the temperature of that point is displayed 0.</li> </ul>		

Temperature instruction proper to FB-2AJ(K)4/ FB-2AH(T)4

FUN 73 TSTC	Convenient instruction of PID temperature control for FB-2AJ(K)4/FB-2AH(T)4	FUN 73 TSTC
----------------	---	----------------

\*When total control points(Zh) of the FUN73 are greater than 4, it must be Sh $\geq$ Sm and the registers storing the current values (starting from TR) must be continuous.

73.TSTC  
 Tp : 2  
 Pl : 3  
 Sm : 0  
 Ym: Y16  
 AR: R3842  
 TR: R0  
 Yh: Y48  
 Sh : 0  
 Zh : 16  
 Sv : R 100  
 Os : R 120  
 PR : R 140  
 IR : R 160  
 DR : R 180  
 OR : R 300  
 WR: R 200

- When M3=ON, to measure the temperature of 1<sup>st</sup> (Sm=0) ~4<sup>th</sup> point of PT-100 RTD inputs and store the engineering values of measurement into R0~R3 and store the original values into R3968~R3971.
- When completing the measurement of 1<sup>st</sup> ~16<sup>th</sup> point, it will perform the PID heating control of 16 (Zh) points from 1<sup>st</sup> (Sh=0) point to the 16<sup>th</sup> point.
- R0~R15 : Registers of current engineering value .
- Y48~Y63 : PID ON/OFF (PWM) output; it must be the transistor output.
- R100~R115 : Registers of set point.
- R120~R135 : Registers of deviation zone, it determines whether temperature falls in setting range.  
E.g. Set point is 200 and deviation zone is 5,  
then 195 $\leq$  Current value  $\leq$  205 means the temperature is in zone.
- R140~R155 : Setting point of gain (Kc).
- R160~R175 : Setting point of integral tuning constant (Ti)
- R180~R195 : Setting point of derivative tuning constant (Td).
- R300~R315 : Output of PID calculation (value from 0~4095).
- R200~R216 : Working registers.

43.NBMV  
 S : R 200  
 Ns : 0  
 D : WM1000  
 Nd : 0

43.NBMV  
 S : R 220  
 Ns : 0  
 D : WM1000  
 Nd : 1

43.NBMV  
 S : R 230  
 Ns : 0  
 D : WM1000  
 Nd : 2

43.NBMV  
 S : R 240  
 Ns : 0  
 D : WM1000  
 Nd : 3

08.MOV  
 S : R 210  
 D : WM1016

08.MOV  
 S : R 208  
 D : WM1032

- When M3=ON, M1016 ~ M1031 tells the warning status of highest temperature warning or the heating circuit opened.
- M1032~M1047 tells the status of temperature in zone.

FUN 85 TPSNS	Convenient instruction proper to FB-4AJ(K)xx temperature module	FUN 85 TPSNS																																																												
Execution control—EN	<p>85.TPSNS</p> <table border="1"> <tr> <td>Tp :</td> <td>ERR—Parameter error</td> </tr> <tr> <td>Pl :</td> <td>ALM—Sensor line broken</td> </tr> <tr> <td>Zn :</td> <td></td> </tr> <tr> <td>Yn :</td> <td></td> </tr> <tr> <td>SR :</td> <td></td> </tr> <tr> <td>WR :</td> <td></td> </tr> </table> <table border="1"> <thead> <tr> <th>Range</th> <th>Y</th> <th>HR</th> <th>ROR</th> <th>DR</th> <th>K</th> </tr> </thead> <tbody> <tr> <td>Operand</td> <td>Y0   Y255</td> <td>R0   R3839</td> <td>R5000   R8071</td> <td>D0   D3071</td> <td></td> </tr> <tr> <td>Tp</td> <td></td> <td></td> <td></td> <td></td> <td>0~1</td> </tr> <tr> <td>Pl</td> <td></td> <td></td> <td></td> <td></td> <td>0~3</td> </tr> <tr> <td>Zn</td> <td></td> <td></td> <td></td> <td></td> <td>12, 18, 24</td> </tr> <tr> <td>Yn</td> <td>○</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>SR</td> <td></td> <td>○</td> <td>○*</td> <td>○</td> <td></td> </tr> <tr> <td>WR</td> <td></td> <td>○</td> <td>○*</td> <td>○</td> <td></td> </tr> </tbody> </table>	Tp :	ERR—Parameter error	Pl :	ALM—Sensor line broken	Zn :		Yn :		SR :		WR :		Range	Y	HR	ROR	DR	K	Operand	Y0   Y255	R0   R3839	R5000   R8071	D0   D3071		Tp					0~1	Pl					0~3	Zn					12, 18, 24	Yn	○					SR		○	○*	○		WR		○	○*	○		<p>Tp: Type of sensor;  =0, K-type thermocouple  =1, J-type thermocouple</p> <p>P1 : Setting of polarity and span;  =0 , 0~10V (Unipolar)  =1 , 0~ 5V (Unipolar)  =2 , -10~10V (Bipolar)  =3 , - 5~ 5V (Bipolar)</p> <p>Unipolar: U/B jumper set at U  Bipolar : U/B jumper set at B  Span : 5V/10V jumper setting</p> <p>Zn : Setting of input points for temperature;  = 12, 18, 24</p> <p>Yn : Starting address of discrete output of this module for multiplexing temperature input; it takes 8 points. When expansion module with discrete output will be installed after the temperature module, the discrete output address of which must be added 8.</p> <p>SR : Starting register of the engineering value of temperature measurement; it takes Zn registers.</p> <p>WR : Starting of working register for this instruction. It takes 5 registers and can't be repeated in using.</p>
Tp :	ERR—Parameter error																																																													
Pl :	ALM—Sensor line broken																																																													
Zn :																																																														
Yn :																																																														
SR :																																																														
WR :																																																														
Range	Y	HR	ROR	DR	K																																																									
Operand	Y0   Y255	R0   R3839	R5000   R8071	D0   D3071																																																										
Tp					0~1																																																									
Pl					0~3																																																									
Zn					12, 18, 24																																																									
Yn	○																																																													
SR		○	○*	○																																																										
WR		○	○*	○																																																										

Note 1: FUN85 is the Convenient instruction dedicated for the multiplexing temperature modules: FB-4AJ(K)xx; where xx may be 12,18,24 ; it means the temperature inputs.

Note 2: The FB-4AJ(K)xx temperature module can only be installed alone, it can't work together with FB-8AD, FB-2AJ(K)4,FB-2AH(T)4 or FB-6AD modules.

#### Function guide and notifications

FB-4AJ(K)xx multiplexing temperature module occupies 8 points of analog input address and 8 points of discrete output address in physical, more detail as followings:

- FB-4AJxx (where, xx may be 12,18,24) provides 4 points of general purpose analog inputs (1<sup>st</sup> ~ 4<sup>th</sup> analog input) and xx points of J-type thermocouple inputs for temperature measurement (With the combination of 5<sup>th</sup> ~8<sup>th</sup> analog inputs and 8 points of discrete output making upto 24 points of temperature measurement).
- FB-4AKxx (where, xx may be 12,18,24) provides 4 points of general purpose analog inputs (1<sup>st</sup> ~ 4<sup>th</sup> analog input) and xx points of K-type thermocouple inputs for temperature measurement (With the combination of 5<sup>th</sup> ~8<sup>th</sup> analog inputs and 8 points of discrete output making upto 24 points of temperature measurement)
- The selection of input span of FB-4AJ(K)xx temperature module can be 5V (500°C) or 10V (1000°C); the input polarity can be set as unipolar (U/B jumper setting at U) or bipolar (U/B jumper setting at B):
  - When setting at10V(1000°C) and unipolar,  
the range of measurement is 0°C~750°C (J-type) or 0°C~900°C (K-type)
  - When setting at 5V(500°C) and unipolar,  
the range of measurement is 0°C~420°C (J-type) or 0°C~450°C (K-type)
  - When setting at 10V(1000°C) and bipolar,  
the range of measurement is -200°C~750°C (J-type) or -200°C~900°C(K-type)
  - When setting at 5V(500°C) and bipolar,  
the range of measurement is -200°C~420°C (J-type) or -200°C~450°C(K-type)
- FB-4AJ(K)xx multiplexing temperature module occupies 8 points of analog input address and 8 points of discrete output address in physical;
  - This kind of temperature module can only be installed alone, it can't work together with FB-8AD, FB-6AD, FB-2AJ(K)4,FB-2AH(T)4 or FB-4A(JK)xx modules.
  - when expansion module with discrete output will be installed after this kind of module, the discrete output address of which must be added 8.
- The memory mapping of general purpose analog inputs as follows:  
Address of 1<sup>st</sup> analog input is R3840; Address of 2<sup>nd</sup> analog input is R3841  
Address of 3<sup>rd</sup> analog input is R3842; Address of 4<sup>th</sup> analog input is R3843
- If the setting of input polarity is unipolar, the primitive value of general purpose analog input is -2048~2047, so, it may be added with the offset 2048 to convert it to be the range of 0~4095 for later operation.
- For the selection of thermocouple, K-type thermocouple is recommended.
- It is recommended to select 0~5V for the span and polarity of input if it meets the requirement.
- Connect the “FG” terminal with the shielding of thermocouple if it is with for better measurement.
- The “G” terminal must be connected to the safty earth ground of the power system.

## Measuring instruction proper to FB-4AJ(K)xx temperature module

FUN 85 TPSNS	Convenient instruction proper to FB-4AJ(K)xx temperature module	FUN 85 TPSNS
<b>User guide for convenient instruction FUN85</b>		
<ul style="list-style-type: none"> <li>● When execution control “EN”=1, this instruction will perform multiplexing temperature measurement and store the primitive value into R3968 (TP0) …R3991 (TP23) ; the value falls in 0~4095 (unipolar) or -2048 ~2047 (bipolar). And then base on the setting of temperature sensor (Tp), input span and polarity (Pl) of the temperature module to scale the primitive values to engineering values and store them into temperature measurement registers (SR+0 as the 1<sup>st</sup> point, ..., SR+23 as the 24<sup>th</sup> point)</li> <li>● When the setting of Tp , Pl , Zn comes error, this instruction will not be performed and the output indication “ERR” will be ON.</li> <li>● When the sensor is K-type thermocouple (it needs FB-4AKxx module): <ul style="list-style-type: none"> <li>1. As the setting of input span and polarity is 0~10V, the range of measurement will be 0~900°C. When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>2. As the setting of input span and polarity is 0~5V, the range of measurement will be 0~450°C. When the display value is greater than 450°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>3. As the setting of input span and polarity is -10~10V, the range of measurement will be -200~900°C. When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>4. As the setting of input span and polarity is -5~5V, the range of measurement will be -200~450°C. When the display value is greater than 450°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON..</li> </ul> </li> <li>● When the sensor is J-type thermocouple (it needs FB-4AJxx module): <ul style="list-style-type: none"> <li>1. As the setting of input span and polarity is 0~10V, the range of measurement will be 0~750°C. When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>2. As the setting of input span and polarity is 0~5V, the range of measurement will be 0~420°C. When the display value is greater than 450°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>3. As the setting of input span and polarity is -10~10V, the range of measurement will be -200~750°C. When the display value is greater than 900°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> <li>4. As the setting of input span and polarity is -5~5V, the range of measurement will be -200~420°C. When the displayed temperature value is greater than 450°C, it means the line broken of the thermocouple and the output indication “ALM” will be ON.</li> </ul> </li> <li>● SR : Starting register of the engineering value of temperature measurement; it needs Zn registers in total. SR+0 stores the 1<sup>st</sup> point of temperature value, SR+1 stores the 2<sup>nd</sup> point of temperature value....</li> <li>● WR : Starting of working register for this instruction. It takes 5 registers and can't be repeated in using. The content of register WR+0 and WR+1 indicates the status of the sensor which is line broken or not. Bit definition of WR+0 explained as follows: Bit0 =1 indicating that the 1<sup>st</sup> point of sensor is line broken; ... B15=1 indicating that the 16<sup>th</sup> point of sensor is line broken. Bit definition of WR+1 explained as follows: Bit0 =1 indicating that the 17<sup>th</sup> point of sensor is line broken; ... B7=1 indicating that the 24<sup>th</sup> point of sensor is line broken. Registers WR+2~WR+7 are used by this instruction.</li> <li>● FUN85 can only be used once.</li> <li>● No matter the FUN85 is placed in main program or in sub-program, and whether the execution control “EN”=0 or 1, this instruction must be executed every scan.</li> </ul>		

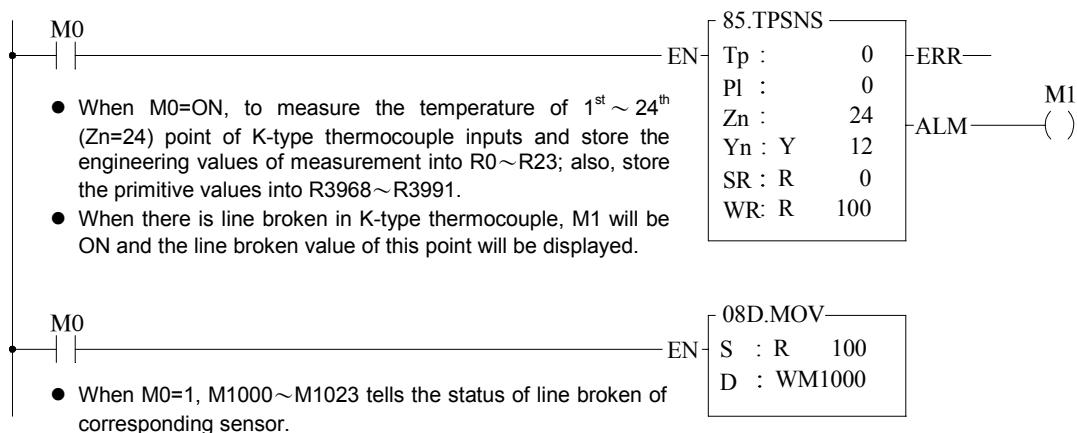
FUN 85 TPSNS	Convenient instruction proper to FB-4AJ(K)xx temperature module	FUN 85 TPSNS
<b>Specific registers for FUN85</b>		
<ul style="list-style-type: none"> <li>● R3968～R3991: Registers storing the primitive temperature value. R3968 storing the 1<sup>st</sup> point, R3969 storing the 2<sup>nd</sup> point, etc. and R3991 storing the 24<sup>th</sup> point. The value is from 0～4095 (unipolar) or -2048～2047 (bipolar).</li> </ul>		
<ul style="list-style-type: none"> <li>● R4000 : Low Byte of R4000 is generated from the system; FUN85 instruction will base on the setting of "temperature sensor (TP)" and "input span and polarity (PI)" to create the default and write it into the low byte of R4000. It is used to determine whether R4000～R4004 needs to be initialized; It is not allowed to change the low byte of R4000 by the user.</li> </ul>		
<ul style="list-style-type: none"> <li>  : High Byte of R4000 to tell the times for the average of measurement, which can be set by the user.</li> </ul>		
<ul style="list-style-type: none"> <li>  =0, no average; every acquired value is the measured value (default)</li> </ul>		
<ul style="list-style-type: none"> <li>  =1, average of 2 times; the average on the acquired 2 times of values is the measured value.</li> </ul>		
<ul style="list-style-type: none"> <li>  =2, average of 4 times; the average on the acquired 4 times of values is the measured value.</li> </ul>		
<ul style="list-style-type: none"> <li>  =3, average of 8 times; the average on the acquired 8 times of values is the measured value.</li> </ul>		
<ul style="list-style-type: none"> <li>  =4, average of 16 times; the average on the acquired 16 times of values is the measured value.</li> </ul>		
<ul style="list-style-type: none"> <li>● R4001 : The factor for linear scaling to calculate the engineering value of K-type thermocouple while in positive temperature;</li> </ul>		
<ul style="list-style-type: none"> <li>  when the setting of input span and polarity for K-type thermocouple is 0～10V or -10～10V,     the default value is 248.</li> </ul>		
<ul style="list-style-type: none"> <li>  when the setting of input span and polarity for K-type thermocouple is 0～5V or -5～5V,     the default value is 124.</li> </ul>		
<ul style="list-style-type: none"> <li>  when the setting of input span and polarity for J-type thermocouple is 0～10V or -10～10V,     the default value is 240.</li> </ul>		
<ul style="list-style-type: none"> <li>  when the setting of input span and polarity for J-type thermocouple is 0～5V or -5～5V,     the default value is 120.</li> </ul>		
<p>The expression for engineering value is as follows:</p>		
<p style="padding-left: 2em;">Engineering value = (Primitive temperature value ×R4001) /1024 (Unipolar).</p>		
<p style="padding-left: 2em;">Engineering value = (Primitive temperature value ×2×R4001) /1024 (Bipolar).</p>		
<p>When there is a slight difference in measurement result between the standard meter and the FB-PLC's temperature module, if the user would like to use the value acquired by standard meter for correction, the user can tune the value of R4001 to get a better result in temperature measurement. This register provides fine tuning for positive temperature.</p>		
<ul style="list-style-type: none"> <li>● R4002 : The factor for linear scaling to calculate the engineering value of K-type thermocouple while in negative temperature;</li> </ul>		
<ul style="list-style-type: none"> <li>  when the setting of input span and polarity for K-type thermocouple is -10～10V or -5～5V,     the default value is 286.</li> </ul>		
<ul style="list-style-type: none"> <li>  when the setting of input span and polarity for J-type thermocouple is -10～10V or -5～5V,     the default value is 280.</li> </ul>		
<p>The expression for engineering value is as follows:</p>		
<p style="padding-left: 2em;">Engineering value = (Primitive temperature value ×R4002) /1024 (-5～5V).</p>		
<p style="padding-left: 2em;">Engineering value = (Primitive temperature value ×2×R4002) /1024 (-10～10V).</p>		
<p>When there is a slight difference in measurement result between the standard meter and the FB-PLC's temperature module, if the user would like to use the value acquired by standard meter for correction, the user can tune the value of R4002 to get a better result in temperature measurement. This register provides fine tuning for negative temperature.</p>		
<ul style="list-style-type: none"> <li>● R4003 : The setting value for line broken detection of thermocouple;</li> </ul>		
<ul style="list-style-type: none"> <li>  when the setting of input span and polarity is 0～10V or -10～10V,     the default value is 901.</li> </ul>		
<ul style="list-style-type: none"> <li>  when the setting of input span and polarity is 0～5V or -5～5V,     the default value is 451.</li> </ul>		

## Measuring instruction proper to FB-4AJ(K)xx temperature module

FUN 85 TPSNS	Convenient instruction proper to FB-4AJ(K)xx temperature module	FUN 85 TPSNS
<ul style="list-style-type: none"> <li>● R4004 : Time interval between the measurement points while multiplexing. Which the user can set up. The unit is in mS and the default value is 333, it means it needs 333mS to measure one point of temperature. This means the update rate of the temperature is 2 seconds (<math>333\text{mS} \times 6 = 1998\text{mS}</math>)           <p>When the value of R4004 is 166, it means it needs 166mS to measure one point of temperature. The update rate of the temperature is 1 second (<math>166\text{mS} \times 6 = 996\text{mS}</math>)</p> <p>When the value of R4014 is 666, it means it needs 666mS to measure one point of temperature. The update rate of the temperature is 4 seconds (<math>666\text{mS} \times 6 = 3996\text{mS}</math>)</p> <p>When the value of R4014 is 1333, it means it needs 1333mS to measure one point of temperature. The update rate of the temperature is 8 seconds (<math>1333\text{mS} \times 4 = 7998\text{mS}</math>)</p> </li> <li>● R4010 : Each bit of R4010 to tell the status of the sensor's installation. Bit0=1 means that 1<sup>st</sup> point of temperature sensor is installed. • • Bit15=1 means that 16<sup>th</sup> point of temperature sensor is installed. (The default of R4010 is FFFFH)</li> <li>● R4011 : Each bit of R4011 to tell the status of the sensor's installation. Bit0=1 means that 17<sup>th</sup> point of temperature sensor is installed. • • Bit7=1 means that 24<sup>th</sup> point of temperature sensor is installed. (The default of R4011 is FFFFH)</li> <li>● When the temperature sensor is installed (the corresponding bit of R4010 or R4011 must be 1), the system will perform the line broken detection to the sensor. If there is line broken happened to the sensor, there will have the warning and the line broken value will be displayed.</li> <li>● When the temperature sensor is not installed (the corresponding bit of R4010 or R4011 must be 0), the system won't perform the line broken detection to the sensor and there will not have the warning; the temperature value will be displayed as 0.</li> <li>● Depends on the sensor's installation, the ladder program may control the corresponding bit of R4010 and R4011 to tell FUN85 to perform or not to perform the line broken detection.</li> </ul>		

**Program example** In following examples, the main unit is FBx-28MC(A), and the FB-4AK24 temperature module is attached ; the setting of the input span and polarity is 0~10V.

### Program example 1



Measuring instruction proper to FB-4AJ(K)xx temperature module

FUN 85 TPSNS	Convenient instruction proper to FB-4AJ(K)xx temperature module	FUN 85 TPSNS
<b>Program example 2</b>		
<ul style="list-style-type: none"> <li>The status of M800 ~ M823 are controlled by the MMI or external inputs to tell the status of sensor's installation; if it has the sensor, perform line broken detection, and not to perform the check if it hasn't. (It needs the retentive function, so M800~M1399 are the better choice).</li> </ul>	<b>08D.MOV</b> S : WM 800 D : R 4010	<ul style="list-style-type: none"> <li>When M0= ON, to measure the temperature of 1<sup>st</sup> ~ 24<sup>th</sup> (Zn=24) point of K-type thermocouple inputs and store the engineering values of measurement into R0~R23; also, store the primitive values into R3968~R3991.</li> <li>When the sensor is installed (the corresponding bit of R4010 or R4011 is 1) and if there is line broken of the sensor, M1 will be ON and the line broken value will be displayed.</li> <li>When the sensor is not installed (the corresponding bit of R4010 or R4011 is 0), there is no line broken detection for that point and the temperature value will be displayed by 0.</li> </ul>

## Temperature instruction proper to FB-4AJ(K)xx module

FUN 86 TPCTL	Convenient instruction of PID temperature control proper to FB-4AJ(K)xx module	FUN 86 TPCTL																																																																																							
<p>Execution control—EN</p> <p>Heating/Cooling—H/C</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td colspan="2" style="text-align: center;">86.TPCTL</td> </tr> <tr> <td>Yn :</td> <td>-ERR—Parameter error</td> </tr> <tr> <td>Sn :</td> <td>-ALM—Warning indication</td> </tr> <tr> <td>Zn :</td> <td></td> </tr> <tr> <td>Sv :</td> <td></td> </tr> <tr> <td>Os :</td> <td></td> </tr> <tr> <td>PR :</td> <td></td> </tr> <tr> <td>IR :</td> <td></td> </tr> <tr> <td>DR :</td> <td></td> </tr> <tr> <td>OR :</td> <td></td> </tr> <tr> <td>WR :</td> <td></td> </tr> </table> <p>Range Operand</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Range</th> <th>Y</th> <th>HR</th> <th>ROR</th> <th>DR</th> <th>K</th> </tr> </thead> <tbody> <tr> <td>Yn</td> <td>○</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Sn</td> <td></td> <td></td> <td></td> <td></td> <td>0~23</td> </tr> <tr> <td>Zn</td> <td></td> <td></td> <td></td> <td></td> <td>1~24</td> </tr> <tr> <td>Sv</td> <td>○</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> <tr> <td>Os</td> <td>○</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> <tr> <td>PR</td> <td>○</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> <tr> <td>IR</td> <td>○</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> <tr> <td>DR</td> <td>○</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> <tr> <td>OR</td> <td>○</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> <tr> <td>WR</td> <td>○</td> <td>○*</td> <td>○</td> <td></td> <td></td> </tr> </tbody> </table>	86.TPCTL		Yn :	-ERR—Parameter error	Sn :	-ALM—Warning indication	Zn :		Sv :		Os :		PR :		IR :		DR :		OR :		WR :		Range	Y	HR	ROR	DR	K	Yn	○					Sn					0~23	Zn					1~24	Sv	○	○*	○			Os	○	○*	○			PR	○	○*	○			IR	○	○*	○			DR	○	○*	○			OR	○	○*	○			WR	○	○*	○			<p>Yn: Starting address of PID ON/OFF output; it takes Zn points.</p> <p>Sn: Starting point of PID control of this instruction; Sn = 0~23.</p> <p>Zn: Number of the PID control of this instruction; 1≤Zn≤24 and 1≤Sn+Zn≤24</p> <p>Sv: Starting register of the setpoint; it takes Zn registers.</p> <p>Os: Starting register of the in-zone offset; it takes Zn registers.</p> <p>PR: Starting register of the gain (Kc); it takes Zn registers.</p> <p>IR: Starting register of integral tuning constant (Ti); it takes Zn registers..</p> <p>DR: Starting register of derivative tuning constant (Td); it takes Zn registers.</p> <p>OR: Starting register of the PID analog output; it takes Zn registers.</p> <p>WR: Starting of working register for this instruction. It takes 9 registers and can't be repeated in using.</p>
86.TPCTL																																																																																									
Yn :	-ERR—Parameter error																																																																																								
Sn :	-ALM—Warning indication																																																																																								
Zn :																																																																																									
Sv :																																																																																									
Os :																																																																																									
PR :																																																																																									
IR :																																																																																									
DR :																																																																																									
OR :																																																																																									
WR :																																																																																									
Range	Y	HR	ROR	DR	K																																																																																				
Yn	○																																																																																								
Sn					0~23																																																																																				
Zn					1~24																																																																																				
Sv	○	○*	○																																																																																						
Os	○	○*	○																																																																																						
PR	○	○*	○																																																																																						
IR	○	○*	○																																																																																						
DR	○	○*	○																																																																																						
OR	○	○*	○																																																																																						
WR	○	○*	○																																																																																						

Note: FUN86 must incorporate with FUN85 when using.

### Function guide and notifications

- Fun85 instruction employs the multiplexing temperature module FB-4AJ(K)xx (where, xx may be 12,16,24) to get the current value of temperature and let it be as so called Process Variable (PV); after the calculation of software PID expression, it will respond the error with an output signal according to the setting of Set Point (SP),the error's integral and the rate of change of the process variable. Through the closed loop operation, the steady state of the process may be expected.
- Convert the output of PID calculation to be the time proportional on/off (PWM) output, and via transistor output to control the SSR for heating or cooling process; this is a good performance and very low cost solution.
- Through the analog output module (D/A module), the output of PID calculation may control the SCR or proportional valve to get more precise process control.
- Digitized PID expression is as follows:

$$M_n = [K_c \times E_n] + \sum_{0}^n [K_c \times T_i \times T_s \times E_n] - [K_c \times T_d \times (P_{Vn} - P_{Vn-1}) / T_s]$$

Where,

Mn: Output at time "n".

Kc: Gain (Range: 1~999 ; Pb=100(%) / Kc)

Ti: Integral tuning constant (Range:0~999, equivalent to 0.00~9.99 Repeat/Minute)

Td: Derivative tuning constant (Range:0~999, equivalent to 0.00~9.99 Minute)

PVn: Process variable at time "n"

PV n-1: Process variable when loop was last solved

En: Error at time "n" ; E= SP – PVn

Ts: Solution interval for PID calculation (Valid value are 10, 20, 40, 80 ;the unit is in 0.1Sec)

FUN 86 TPCTL	Convenient instruction of PID temperature control proper to FB-4AJ(K)xx module	FUN 86 TPCTL
-----------------	---	-----------------

**Principle of PID parameter adjustment**

- As the gain (Kc) adjustment getting larger, the larger the proportional contribution to the output. This can obtain a sensitive and rapid control reaction. However, when the gain is too large, it may cause oscillation. Do the best to adjust "Kc" larger (but not to the extent of making oscillation), which could increase the process reaction and reduce the steady state error.
- Integral item may be used to eliminate the steady state error. The larger the number (Ti, integral tuning constant), the larger the integral contribution to the output. When there is steady state error, adjust the "Ti" larger to decrease the error.  
When the "Ti" = 0, the integral item makes no contribution to the output.  
For exa. , if the reset time is 6 minutes, Ti=100/6=17 ; if the integral time is 5 minutes, Ti=100/5=20.
- Derivative item may be used to make the process smoother and not too over shoot. The larger the number (Td, derivative tuning constant), the larger the derivative contribution to the output. When there is too over shoot, adjust the "Td" larger to decrease the amount of over shoot.  
When the "Td" = 0, the derivative item makes no contribution to the output.  
For exa, if the rate time is 1 minute, then the Td = 100; if the differential time is 2 minute, then the Td = 200.
- Properly adjust the PID parameters can obtain an excellent result for temperature control.
- The default of gain value (Kc) is as follows:  
When the setting of span and polarity of the module is 0~10V, the default of gain (Kc) is 60.  
When the setting of span and polarity of the module is 0~5V, the default of gain (Kc) is 30.  
When the setting of span and polarity of the module is -10~10V, the default of gain (Kc) is 120.  
When the setting of span and polarity of the module is -5~5V, the default of gain (Kc) is 60.
- The default of integral tuning constant is 17, it means the reset time is 6 minutes (Ti=100/6=17).
- The default of derivative tuning constant is 100, it means the rate time is 1 minutes (Td=100).

**Instruction guide**

- FUN86 instruction must be incorporated with FUN85 ; the FUN85 instruction is for temperature measurement and it must be enabled, then, can the FUN86 start working.
- When execution control "EN" = 1, it depends on the input status of H/C for PID operation to make heating (H/C=1) or cooling (H/C=0) control. The current values of measured temperature are through the multiplexing temperature module FB-4AJ(K)xx to get ; the set points of desired temperature are stored in the registers starting from Sv. With the calculation of software PID expression, it will respond the error with an output signal according to the setting of set point, the error's integral and the rate of change of the process variable. Convert the output of PID calculation to be the time proportional on/off (PWM) output, and via transistor output to control the SSR for heating or cooling process; where there is a good performance and very low cost solution. It may also apply the output of PID calculation (stored in registers starting from OR), by way of D/A analog output module, to control SCR or proportional valve, so as to get more precise process control.
- When the setting of Sn, Zn ( $0 \leq Sn \leq 23$  and  $1 \leq Zn \leq 24$ , as well as  $1 \leq Sn + Zn \leq 24$ ) comes error, this instruction will not be executed and the instruction output "ERR" will be ON.
- This instruction compares the current value with the set point to check whether the current temperature falls within deviation range (stored in register starting from Os). If it falls in the deviation range, it will set the in-zone bit of that point to be ON; if not, clear the in-zone bit of that point to be OFF, and make instruction output "ALM" to be ON.

## Temperature instruction proper to FB-4AJ(K)xx module

FUN 86 TPCTL	Convenient instruction of PID temperature control proper to FB-4AJ(K)xx module	FUN 86 TPCTL
<ul style="list-style-type: none"> <li>● In the mean time, this instruction will also check whether highest temperature warning (the register for the set point of highest temperature warning is R4008). When successively scanning for ten times the current values of measured temperature are all higher than or equal to the highest warning set point, the warning bit will set to be ON and instruction output "ALM" will be on. This can avoid the safety problem aroused from temperature out of control, in case the SSR or heating circuit becomes short.</li> <li>● This instruction can also detect the unable to heat problem resulting from the SSR or heating circuit runs open, or the obsolete heating band. When output of temperature control turns to be large power (set in R4006 register) successively in a certain time (set in R4007 register), and can not make current temperature fall in desired range, the warning bit will set to be ON and instruction output "ALM" will be ON.</li> <li>● WR: Starting of working register for this instruction. It takes 9 registers and can't be repeated in using.            The content of the two registers WR+0 and WR+1 indicating that whether the current temperature falls within the deviation range (stored in registers starting from Os). If it falls in the deviation range, the in-zone bit of that point will be set ON; if not, the in-zone bit of that point will be cleared OFF.            Bit definition of WR+0 explained as follows:            Bit0=1, it represents that the temperature of the Sn+0 point is in-zone...            Bit15=1, it represents that the temperature of the Sn+15 point is in-zone.            Bit definition of WR+1 explained as follows:            Bit0=1, it represents that the temperature of the Sn+16 point is in-zone...            Bit7=1, it represents that the temperature of 24th point is in-zone.            The content of the two registers WR+2 and WR+3 are the warning bit registers, they indicate that whether there exists the highest temperature warning or heating circuit opened.            Bit definition of WR+2 explained as follows:            Bit0=1, it means that there exists the highest warning or heating circuit opened at the Sn+0 point...            Bit15=1, it means that there exists the highest warning or heating circuit opened at the Sn+15 point.            Bit definition of WR+11 explained as follows:            Bit0=1, it means that there exists the highest warning or heating circuit opened at the Sn+16 point...            Bit7=1, it means that there exists the highest warning or heating circuit opened at the 24th point.            Registers of WR+4 ~ WR+8 are used by this instruction.</li> </ul>		
<ul style="list-style-type: none"> <li>● This instruction can only be used to perform heating or cooling control of positive temperature.</li> <li>● Whether the FUN86 is placed in main or sub program and no matter the execution control "EN"=0 or 1, this instruction must be executed every scan.</li> </ul>		
<b>Specific registers related to FUN86</b>		
<ul style="list-style-type: none"> <li>● R4005 : The content of Low Byte to define the solution interval between PID calculation            =0, perform the PID calculation every 2 seconds (System default).            =1, perform the PID calculation every 4 seconds.            =2, perform the PID calculation every 8 seconds.            ≥3, perform the PID calculation every 1 second. (R4004 must be 166 to make sense)            : The content of High Byte to define the cycle time of PID ON/OFF (PWM) output.            =0, PWM cycle time is 2 seconds (system default)            =1, PWM cycle time is 4 seconds.            =2, PWM cycle time is 8 seconds.            ≥3, PWM cycle time is 1 second.</li> </ul>		
<p>Note 1: When changing the value of R4005, the execution control "EN" of FUN86 must be set at 0. The next time when execution control "EN" =1, it will base on the latest set point to perform the PID calculation.</p> <p>Note 2: The smaller the cycle time of PWM, the more even can it perform the heating. However, the error caused by the PLC scan time will also become greater. For the best control, it can base on the scan time of PLC to adjust the solution interval of PID calculation and the PWM cycle time.</p>		

FUN 86 TPCTL	Convenient instruction of PID temperature control proper to FB-4AJ(K)xx module	FUN 86 TPCTL
<ul style="list-style-type: none"> <li>● R4006: The setting point of large power output detection for SSR or heating circuit opened, or heating band obsolete. The unit is in % and the setting range falls in 80~100(%); system default is 90(%).</li> <li>● R4007: The setting time to detect the continuing duration of large power output while SSR or heating circuit opened, or heating band obsolete. The unit is in second and the setting range falls in 300~65535 (seconds); system default is 600 (seconds).</li> <li>● R4008: The setting point of highest temperature warning for SSR, or heating circuit short detection. The unit is in degree and the setting range falls in 50~65535; system default is 350 (degrees).</li> <li>● R4012: Each bit of R4012 to tell the need of PID temperature control.           <ul style="list-style-type: none"> <li>Bit0=1 means that 1<sup>st</sup> point needs PID temperature control.</li> <li>Bit1=1 means that 2<sup>nd</sup> point needs PID temperature control.</li> <li>.</li> <li>.</li> <li>Bit15=1 means that 16<sup>th</sup> point needs PID temperature control. (The default of R4012 is FFFFH)</li> </ul> </li> <li>● R4013: Each bit of R4013 to tell the need of PID temperature control.           <ul style="list-style-type: none"> <li>Bit0=1 means that 17<sup>th</sup> point needs PID temperature control.</li> <li>Bit1=1 means that 18<sup>th</sup> point needs PID temperature control.</li> <li>.</li> <li>.</li> <li>Bit7=1 means that 24<sup>th</sup> point needs PID temperature control. (The default of R4013 is FFFFH)</li> </ul> </li> <li>● While execution control “EN”=1 and the corresponding bit of PID control of that point is ON (corresponding bit of R4012 or R4013 must be 1), the FUN73 instruction will perform the PID operation and respond to the calculation with the output signal.</li> <li>● While execution control “EN”=1 and the corresponding bit of PID control of that point is OFF (corresponding bit of R4012 or R4013 must be 0), the FUN73 will not perform the PID operation and the output of that point will be OFF.</li> <li>● The ladder program may control the corresponding bit of R4012 and R4013 to tell the FUN73 to perform or not to perform the PID control, and it needs only one FUN86 instruction.</li> </ul>		

Temperature instruction proper to FB-4AJ(K)xx module

FUN 86 TPCTL	Convenient instruction of PID temperature control proper to FB-4AJ(K)xx module	FUN 86 TPCTL
<b>Program example</b> In following examples, the main unit is FBx-28MC(A), and the FB-4AK24 temperature module is attached ; the setting of the input span and polarity is 0~10V.		
<b>Program example 1</b>		

The ladder logic diagram illustrates two parallel logic paths. Both paths begin with a normally closed contact labeled M0 in series with the enable signal (EN). The top path leads to a function block 85.TPSNS with the following settings:

- Tp : 0
- Pl : 0
- Zn : 24
- Yn : Y12
- SR : R0
- WR: R50

The output of this block connects to both the error indicator (ERR) and the status output M1. The bottom path leads to a function block 08D.MOV with the following settings:

- S : R50
- D : WM 1000

This block outputs to the data register D.

Temperature instruction proper to FB-4AJ(K)xx module

FUN 86 TPCTL	Convenient instruction of PID temperature control proper to FB-4AJ(K)xx module	FUN 86 TPCTL
<ul style="list-style-type: none"> <li>When M2=ON, it will perform the PID heating control of 20 (Zn=20) points from the 1<sup>st</sup> (Sn=0) point to the 20<sup>th</sup> point.</li> <li>Y40~Y49: PID ON/OFF (PWM) output; it must be the transistor output.</li> <li>R100~R119: Registers of set point.</li> <li>R120~R139: Registers of deviation zone, it determines whether the temperature falls in setting range. E.g. Set point is 200 and deviation zone is 5, then <math>195 \leq \text{Current value} \leq 205</math> means the temperature is in zone.</li> <li>R140~R159: Setting point of gain (Kc).</li> <li>R160~R179: Setting point of integral tuning constant (Ti).</li> <li>R180~R199: Setting point of derivative tuning constant (Td).</li> <li>R200~R219: Output of PID calculation (value from 0~4095).</li> <li>R220~R228: Working registers</li> <li>When one of the temperatures is not in zone, or there exists highest temperature warning or heating circuit opened, the status of M3 will be ON.</li> </ul>	<ul style="list-style-type: none"> <li>When M2=ON, the M1024~M1043 tells the point which temperature is in zone, and M1048~M1067 tells the point which has highest temperature warning or heating circuit opened.</li> <li>When M4=ON, it will perform the PID cooling control of 3 (Zn=3) points from the 21<sup>st</sup> (Sn=20) point to the 23<sup>rd</sup> point.</li> <li>Y58~Y60: PID ON/OFF (PWM) output; it must be the transistor output.</li> <li>R300~R302: Registers of set point.</li> <li>R305~R307: Registers of deviation zone, it determines whether the temperature falls in setting range. E.g. Set point is 200 and deviation zone is 5, then <math>195 \leq \text{Current value} \leq 205</math> means the temperature is in zone.</li> <li>R310~R312: Setting point of gain (Kc).</li> <li>R315~R317: Setting point of integral tuning constant (Ti)</li> <li>R320~R322: Setting point of derivative tuning constant (Td).</li> <li>R325~R327: Output of PID calculation (value from 0~4095).</li> <li>R330~R338: Working registers.</li> </ul>	

Note: When performing the instruction of the first time, the FUN86 will automatically assign to each point its system default for gain (Kc), integral tuning constant (Ti), and derivative tuning constant (Td), etc. They may be changed if necessary.

Temperature instruction proper to FB-4AJ(K)xx module

FUN 86 TPCTL	Convenient instruction of PID temperature control proper to FB-4AJ(K)xx module	FUN 86 TPCTL
<b>Program example 2</b>		
<ul style="list-style-type: none"> <li>When M0=ON, to measure the temperature of 1<sup>st</sup> ~ 24<sup>th</sup> (Zn=24) point of K-type thermocouple inputs and store the engineering values of measurement into R0~R23; also, store the primitive values into R3968~R3991.</li> <li>When there is line broken in K-type thermocouple, M1 will be ON and the line broken value of this point will be displayed.</li> </ul>		

Temperature instruction proper to FB-4AJ(K)xx module

FUN 86 TPCTL	Convenient instruction of PID temperature control proper to FB-4AJ(K)xx module	FUN 86 TPCTL
	<p>● The status of M832~M851 are controlled by the MMI or external inputs to tell whether it needs the PID control of the corresponding point; perform the PID operation when the bit is ON, and not to perform if it is OFF (It needs the retentive function, so M800~M1399 are the better choice).</p> <pre>     M2 --- EN[86.TPCTL]                Yn : Y 30                Sn : 0                Zn : 20                Sv : R 100                Os : R 120                PR : R 140                IR : R 160                DR : R 180                OR : R 200                WR : R 220                ERR --- M3                ALM --- ( )   </pre> <p>● When M2=ON, it will perform the PID heating control of 20 (<math>Z_n=20</math>) points from the 1<sup>st</sup> (<math>S_n=0</math>) point to the 20<sup>th</sup> point.</p> <ul style="list-style-type: none"> <li>● Y30~Y49: PID ON/OFF (PWM) output; it must be the transistor output.</li> <li>● R100~R119: Registers of set point.</li> <li>● R120~R139: Registers of deviation zone, it determines whether the temperature falls in setting range.</li> </ul> <p>E.g. Set point is 200 and deviation zone is 5, then <math>195 \leq \text{Current value} \leq 205</math> means the temperature is in zone.</p> <ul style="list-style-type: none"> <li>● R140~R159: Setting point of gain (<math>K_c</math>).</li> <li>● R160~R179: Setting point of integral tuning constant (<math>T_i</math>).</li> <li>● R180~R199: Setting point of derivative tuning constant (<math>T_d</math>).</li> <li>● R200~R219: Output of PID calculation (value from 0~4095).</li> <li>● R220~R228: Working registers</li> <li>● When one of the temperatures is not in zone, or there exists highest temperature warning or heating circuit opened, the status of M3 will be ON.</li> <li>● When temperature control bit is ON (the corresponding bit of R4012 or R4013 is 1), FUN86 performs the PID operation of that point to obtain a suitable output signal.</li> <li>● When temperature control bit is OFF (the corresponding bit of R4012 or R4013 is 0), FUN86 will not perform the PID operation of that point and output will be OFF.</li> </ul> <p>Note: When performing the instruction of the first time, the FUN86 will automatically assign to each point its system default for gain (<math>K_c</math>), integral tuning constant (<math>T_i</math>), and derivative tuning constant (<math>T_d</math>), etc. They may be changed if necessary.</p> <p>● When M2=ON, the M1024~M1043 tells the point which temperature is in zone, and M1048 ~ M1067 tells the point which has highest temperature warning or heating circuit opened.</p> <pre>     M2 --- EN[08D.MOV]                S : R 220                D : WM1024                EN[08D.MOV]                S : R 222                D : WM1048   </pre>	

# Chapter 21 General purpose PID control

## 21.1 Introduction of PID control

As the general application of process control, the open loop methodology may be good enough for most situations, because the key control elements or components are more sophisticated, and the performances of which are getting better, there is no doubt, the stability and reliability may meet the desired requirement. It is the way to get not bad C/P value with great economic consideration. But the characteristics of the elements or components may change following the time eclipse and the controlling process may be affected by the change of loading or external disturbances, the performance of open loop becomes looser; it is the weakness of such solution. Thus, closed loop (with the sensors to feedback the real conditions of controlling process for loop calculation) PID control is one of the best choices for manufacturing process to make perfect quantity and best products.

FB-PLC provides digitized PID mathematical algorithm for general purpose application, it is enough for most of applications, but the response time of loop calculation will have the limitation by the scan time of PLC, thus it must be taken into consideration while in very fast closed loop control.

For an introduction to key parts of a control loop, refer to the block diagram shown below. The closed path around the diagram is the "loop" referred to in "closed loop control".

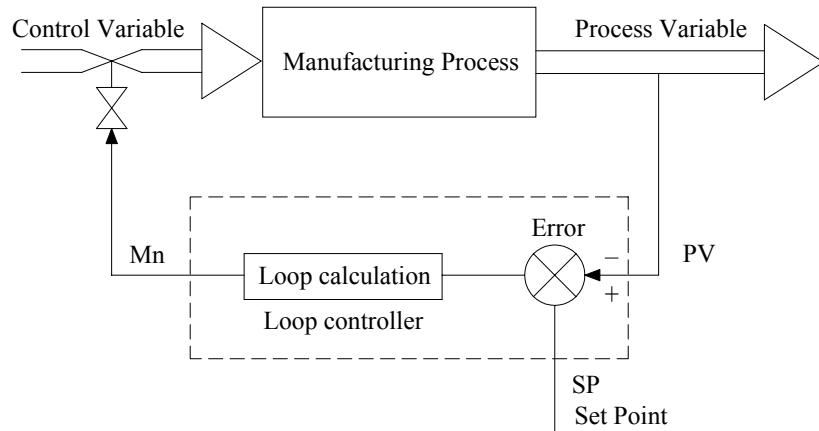


Figure21-1. Typical Analog Loop Control System

## 21.2 How to select the controller

Depends on the requirement, the users may apply the suitable controller for different applications; it is much better of the thinking that the control algorithm is so simple and easy to operate and the final result will be good enough, that's all. Therefore comes the answers, there are three types of controller could be activated from the PID mathematical expression, these are so called "Proportional Controller", "Proportional + Integral Controller" and "Proportional + Integral + Derivative Controller". The digitized mathematical expression of each controller shown bellows.

### 21.2.1 Proportional Controller

The digitized mathematical expression as follows:

$$M_n = (1000/P_b) \times (E_n) + \text{Bias}$$

Where,

$M_n$  : Output at time "n".

$P_b$  : Proportional band

- the expression stating the percent change in error required to change the output full scale.  
[ Range : 2~5000 , unit in 0.1% ;  $K_c(\text{gain})=1000/P_b$  ]

$E_n$  : The difference between the set point (SP) and the process variable (PV) at time "n";

$$E_n = SP - PV_n$$

$T_s$  : Solution interval between calculations ( Range : 1~3000, unit in 0.01S )

Bias : Offset to the output ( Range : 0~4095 )

The algorithm of "Proportional Controller" is very simple and easy to implement, and it takes less time for loop calculation. Most of the general applications, this kind of controller is good enough, but it needs to adjust the offset ( Bias ) to the output to eliminate the steady state error due to the change of set point.

### 21.2.2 Proportional + Integral Controller

The digitized mathematical expression as follows:

$$M_n = (1000/P_b) \times (E_n) + \sum_0^n [(1000/P_b) \times T_i \times T_s \times E_n] + \text{Bias}$$

Where,

$M_n$  : Output at time "n".

$P_b$  : Proportional band [ Range : 2~5000 , unit in 0.1% ;  $K_c(\text{gain})=1000/P_b$  ]

$E_n$  : The difference between the set point (SP) and the process variable (PV) at time "n";  
 $E_n = SP - PV_n$

$T_i$  : Integral tuning constant ( Range : 0~9999 , it means 0.00~99.99 Repeats/Minute )

$T_s$  : Solution interval between calculations ( Range : 1~3000, unit in 0.01S )

Bias : Offset to the output ( Range : 0~4095 )

The most benefit of the controller with integral item is to overcome the shortage of the "Proportional Controller" mentioned above; via the integral contribution, the steady state error may disappear, thus it is not necessary to adjust the offset manually while changing the set point. Almost, the offset ( Bias ) to the output will be 0.

### 21.2.3 Proportional + Integral + Derivative Controller

The digitized mathematical expression as follows:

$$M_n = (1000/P_b) \times (E_n) + \sum_0^n [(1000/P_b) \times T_i \times T_s \times E_n] - [(1000/P_b) \times T_d \times (P_{Vn} - P_{Vn-1})/T_s] + \text{Bias}$$

Where,

- M<sub>n</sub> : Output at time "n".
- P<sub>b</sub> : Proportional band [ Range : 2~5000 , unit in 0.1% ; K<sub>c</sub>(gain)=1000/P<sub>b</sub> ]
- E<sub>n</sub> : The difference between the set point (SP) and the process variable (PV) at time "n";  
$$E_n = SP - P_{Vn}$$
- T<sub>i</sub> : Integral tuning constant ( Range : 0~9999 , it means 0.00~99.99 Repeats/Minute )
- T<sub>d</sub> : Derivative tuning constant ( Range : 0~9999 , it means 0.00~99.99 Minute )
- P<sub>Vn</sub> : Process variable at time "n"
- P<sub>Vn-1</sub> : Process variable when loop was last solved
- T<sub>s</sub> : Solution interval between calculations ( Range : 1~3000, unit in 0.01S )
- Bias : Offset to the output ( Range : 0~4095 )

Derivative item of the controller may have the contribution to make the response of controlling process smoother and not too over shoot. But because it is very sensitive of the derivative contribution to the process reaction, most of applications, it is not necessary of this item and let the tuning constant (T<sub>d</sub>) be equal to 0.

## 21.3 Explanation of the PID instruction and example program follows

The followings are the instruction explanation and program example for PID (FUN30) loop control of FB-PLC.

## Mathematics instructions

FUN 30 PID	Convenient instruction of PID loop operation	FUN 30 PID																																			
	<p style="text-align: center;">30.PID</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 30%;">Auto/Manual—A/M</td> <td style="width: 10%;"><b>Ts :</b></td> <td style="width: 10%;"><b>ERR</b>—Invalid setting</td> <td style="width: 10%;"><b>SR :</b></td> <td style="width: 10%;"><b>HA</b>—High Alarm</td> <td style="width: 10%;"><b>OR :</b></td> <td style="width: 10%;"><b>PR :</b></td> <td style="width: 10%;"><b>WR :</b></td> <td style="width: 10%;"><b>LA</b>—Low Alarm</td> </tr> <tr> <td>Bumpless Transfer—BUM</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>Direct/Reverse—D/R</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>	Auto/Manual—A/M	<b>Ts :</b>	<b>ERR</b> —Invalid setting	<b>SR :</b>	<b>HA</b> —High Alarm	<b>OR :</b>	<b>PR :</b>	<b>WR :</b>	<b>LA</b> —Low Alarm	Bumpless Transfer—BUM									Direct/Reverse—D/R									<p style="text-align: center;"><b>Ts</b> : Solution interval between calculations (1~3000 ; unit in 0.01S)</p> <p style="text-align: center;"><b>SR</b> : Starting register of loop settings ; it takes 8 registers in total.</p> <p style="text-align: center;"><b>OR</b> : Output register of PID loop operation.</p> <p style="text-align: center;"><b>PR</b> : Starting register of loop parameters; it takes 7 registers.</p> <p style="text-align: center;"><b>WR</b> : Starting register of working registers for this instruction ; it takes 5 registers and can't be repeated in using.</p>								
Auto/Manual—A/M	<b>Ts :</b>	<b>ERR</b> —Invalid setting	<b>SR :</b>	<b>HA</b> —High Alarm	<b>OR :</b>	<b>PR :</b>	<b>WR :</b>	<b>LA</b> —Low Alarm																													
Bumpless Transfer—BUM																																					
Direct/Reverse—D/R																																					
	<table border="1" style="margin-left: auto; margin-right: auto; width: fit-content;"> <thead> <tr> <th style="text-align: center;">Range</th> <th style="text-align: center;">HR</th> <th style="text-align: center;">ROR</th> <th style="text-align: center;">DR</th> <th style="text-align: center;">K</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Oper- and</td> <td style="text-align: center;">R0   R3839</td> <td style="text-align: center;">R5000   R8071</td> <td style="text-align: center;">D0   D3071</td> <td></td> </tr> <tr> <td style="text-align: center;"><b>Ts</b></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;">1~3000</td> </tr> <tr> <td style="text-align: center;"><b>SR</b></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input checked="" type="radio"/>*</td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> <tr> <td style="text-align: center;"><b>OR</b></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input checked="" type="radio"/>*</td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> <tr> <td style="text-align: center;"><b>PR</b></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input checked="" type="radio"/>*</td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> <tr> <td style="text-align: center;"><b>WR</b></td> <td style="text-align: center;"><input type="radio"/></td> <td style="text-align: center;"><input checked="" type="radio"/>*</td> <td style="text-align: center;"><input type="radio"/></td> <td></td> </tr> </tbody> </table>	Range	HR	ROR	DR	K	Oper- and	R0   R3839	R5000   R8071	D0   D3071		<b>Ts</b>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1~3000	<b>SR</b>	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>		<b>OR</b>	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>		<b>PR</b>	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>		<b>WR</b>	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>		
Range	HR	ROR	DR	K																																	
Oper- and	R0   R3839	R5000   R8071	D0   D3071																																		
<b>Ts</b>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	1~3000																																	
<b>SR</b>	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>																																		
<b>OR</b>	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>																																		
<b>PR</b>	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>																																		
<b>WR</b>	<input type="radio"/>	<input checked="" type="radio"/> *	<input type="radio"/>																																		

- The FB-PLC software algorithm uses mathematical functions to simulate a three-mode (PID) analog controlling technique to provide direct digital control. The control technique responds to an error with an output signal. The output is proportional to the error, the error's integral and the rate of change of the process variable. Control algorithms include, P, PI, PD and PID which all include the features of auto/manual operation, bumpless/balanceless transfers, reset wind-up protection, and adaptive tuning of gain, integral, and derivative terms.

- The digitized mathematical expression of FB-PLC PID instruction as bellow:

$$Mn = (1000/Pb) \times (En) + \sum_0^n [(1000/Pb) \times Ti \times Ts \times En] - [(1000/Pb) \times Td \times (PVn - PVn-1)/Ts] + Bias$$

Where,

- Mn : Output at time "n"
- Pb : Proportional band  
- the expression stating the percent change in error required to change the output full scale.  
〔 Range : 2~5000 , unit in 0.1% ; Kc(gain)=1000/Pb 〕
- Ti : Integral tuning constant (Range : 0~9999 , it means 0.00~99.99 Repeats/Minute)
- Td : Derivative tuning constant (Range : 0~9999 , it means 0.00~99.99 Minute )
- PVn : Process variable at time "n"
- PVn-1 : Process variable when loop was last solved
- En : The difference between the set point (SP) and the process variable (PV) at time "n";  
En = SP - PVn
- Ts : Solution interval between calculations (Range : 1~3000, unit in 0.01S)
- Bias : Offset to the output (Range : 0~4095)

FUN30 PID	Convenient instruction of PID loop operation	FUN30 PID
<b>Principle of PID parameter adjustment</b>		
<ul style="list-style-type: none"> <li>● As the proportional band (Pb) adjustment getting smaller, the larger the proportional contribution to the output. This can obtain a sensitive and rapid control reaction. However, when the proportional band is too small, it may cause oscillation. Do the best to adjust "Pb" smaller (but not to the extent of making oscillation), which could increase the process reaction and reduce the steady state error.</li> <li>● Integral item may be used to eliminate the steady state error. The larger the number (Ti, integral tuning constant), the larger the integral contribution to the output. When there is steady state error, adjust the "Ti" larger to decrease the error. When the "Ti" = 0, the integral item makes no contribution to the output. For ex, if the reset time is 6 minutes, <math>Ti=100/6=17</math> ; if the integral time is 5 minutes, <math>Ti=100/5=20</math>.</li> <li>● Derivative item may be used to make the process smoother and not too over shoot. The larger the number (Td, derivative tuning constant), the larger the derivative contribution to the output. When there is too over shoot, adjust the "Td" larger to decrease the amount of over shoot. When the "Td" = 0, the derivative item makes no contribution to the output. For ex, if the rate time is 1 minute, then the <math>Td = 100</math>; if the rate time is 2 minute, then the <math>Td = 200</math>.</li> <li>● Properly adjust the PID parameters can obtain an excellent result for loop control.</li> </ul>		

**Instruction description**

- When control input "A/M"=0, it performs manual control and will not execute the PID calculation. Directly fill the output value into the output register (OR) to control the loop operation.
- When control input "A/M"=1, it defines the auto mode of loop control; the output of the loop operation is loaded by the PID instruction every time it is solved. It is equal to Mn (control loop output) in the digital approximation equation.
- When control input "BUM"=1, it defines bumpless transfer while the loop operation changing from manual into auto mode.
- When control input "A/M"=1, and direction input "D/R"=1, it defines the direct control for loop operation; it means the output increases as error increases
- When control input "A/M"=1, and direction input "D/R"=0, it defines the reverse control for loop operation; it means the output decreases as error increases
- When comes the error setting of loop setting points or loop parameters, the PID operation will not be performed and the output indication "ERR" will be ON
- While the engineering value of the controlling process is greater than or equal to the user set High Limit, the output indication "HA" will be ON regardless of "A/M" state.
- While the engineering value of the controlling process is less than or equal to the user set Low Limit, the output indication "LA" will be ON regardless of "A/M" state.

## Mathematics instructions

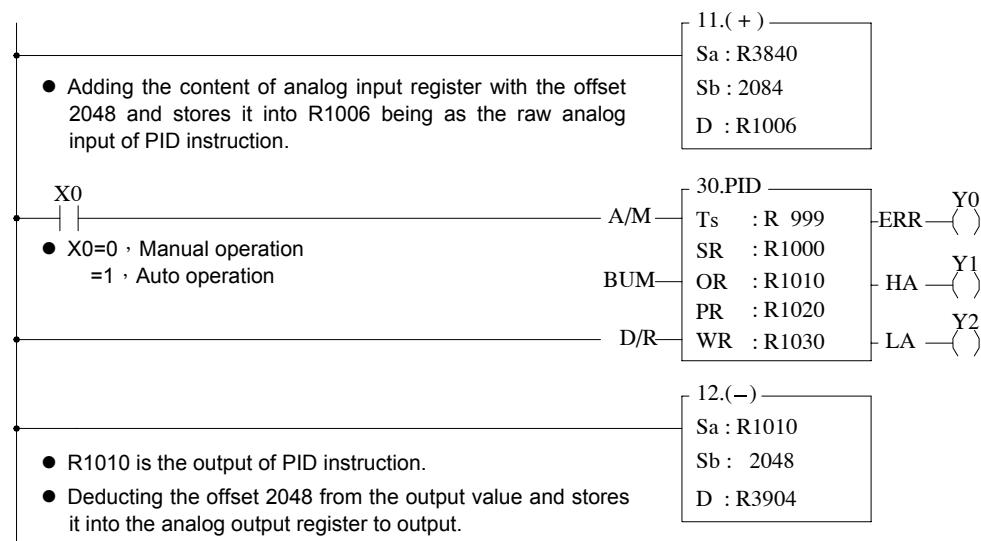
FUN30 PID	Convenient instruction of PID loop operation	FUN30 PID
<p>● <b>Description of operand Ts :</b></p> <ul style="list-style-type: none"> <li>● Ts : It defines the solution interval between PID calculations, the unit is in 0.01 sec; this term may be constant or variable data.</li> </ul> <p>● <b>Description of operand SR (Loop setting registers) :</b></p> <ul style="list-style-type: none"> <li>● SR+0 = Scaled Process Variable : This register is loaded by the PID instruction every time it gets solved. A linear scaling is done on SR+6 using the high and low engineering range found in SR+4 and SR+5.</li> <li>● SR+1 = Setpoint (SP) : The user must load this register with the desired setpoint the loop should control at. The setpoint is entered in engineering units, it must be the range : LER <math>\leq</math> SP <math>\leq</math> HER</li> <li>● SR+2 = High Alarm Limit (HAL) : The user must load this register with the value at which the process variable should be alarmed as a high alarm (above the setpoint). This value is entered as the actual alarm point in engineering units and it must be the range : LER <math>\leq</math> LAL &lt; HAL <math>\leq</math> HER</li> <li>● SR+3 = Low Alarm Limit (LAL) : The user must load this register with the value at which the process variable should be alarmed as a low alarm (below the setpoint). This value is entered as the actual alarm point in engineering units and it must be the range : LER <math>\leq</math> LAL &lt; HAL <math>\leq</math> HER</li> <li>● SR+4 = High Engineering Range (HER) : The user must load this register with the highest value for which the measurement device is spanned. (For example a thermocouple might be spanned for 0 to 500 degrees centigrade, resulting in a 0 to 10V analog input to the FB-PLC (0V=0°C, 10V=500 °C); the high engineering range is 500, this is the value entered into SR+4.) The high engineering range must be : -9999 &lt; HER <math>\leq</math> 9999</li> <li>● SR+5 = Low Engineering Range (LER) : The user must load this register with the lowest value for which the measurement device is spanned. The low engineering range must be : -9999 <math>\leq</math> LER <math>\leq</math> LAL &lt; HAL <math>\leq</math> HER</li> <li>● SR+6 = Raw Analog Measurement (RAM) : The USER'S PROGRAM must load this register with the process variable (measurement). It is the value that the content of analog input register (R3840 ~R3903) is added by the offset of 2048. It must be the range : 0 <math>\leq</math> RAM <math>\leq</math> 4095</li> <li>● SR+7 = Offset of Process Variable (OPV) : The user must load this register with the value as described follows: OPV must be 0 if the raw analog signal and the measurement span of the analog input module are all 0~20mA, there is no loss of the measurement resolution; OPV must be 819 if the raw analog signal is 4~20mA but the measurement span of the analog input module is 0 ~ 20mA, there will have few loss of the measurement resolution (4095 x 4 / 20 = 819) . It must be the range : 0 <math>\leq</math> OPV &lt; 4095</li> <li>● When the setting mentioned above comes error, it will not perform PID operation and the output indication "ERR" will be ON.</li> </ul> <p>● <b>Description of operand OR :</b></p> <ul style="list-style-type: none"> <li>● OR : Output register, this register is loaded directly by the user while the loop in manual operation mode. While the loop in auto operation mode, this register is loaded by the PID instruction every time it is solved. It is equal to Mn (control loop output) in the digital approximation equation. It must be the range : 0 <math>\leq</math> OR <math>\leq</math> 4095</li> </ul>		

FUN 30 PID	Convenient instruction of PID loop operation	FUN 30 PID
<p>● <b>Description of operand PR (Loop parameters) :</b></p> <ul style="list-style-type: none"> <li>● PR+0 = Proportional Band (Pb) : The user must load this register with the desired proportional constant. The proportion constant is entered as a value between 0002 and 5000 where the smaller the number, the larger the proportional contribution. (This is because the equation uses 1000 divided by Pb.) It must be the range : <math>2 \leq Pb \leq 5000</math>, unit is in 0.1% <math>K_c(\text{gain})=1000/Pb</math></li> <li>● PR+1 = Reset Time Constant (Ti) : The user may load this register to add INTEGRAL action to the calculation. The value entered is "Repeats/Minute" and is entered as a number between 0000 and 9999. (The actual range is 00.00 to 99.99 Repeats/Minute.) The larger the number, the larger the integral contribution to the output. It must be the range : <math>0 \leq Ti \leq 9999</math> (0.00~99.99 Repeats/Minute)</li> <li>● PR+2 = Rate Time Constant (Td) : The user may load this register to add DERIVATIVE action to the calculation. The value is entered as minutes and is entered as a number between 0000 and 9999. (The actual range is 00.00 to 99.99 minutes.) The larger the number, the larger the derivative contribution to the output. It must be the range : <math>0 \leq Td \leq 9999</math> (0.00~99.99 Minutes)</li> <li>● PR+3 = Bias : The user may load this register if a bias is desired to be added to the output when using PI or PID control. A bias must be used when running PROPORTIONAL only control. The bias is entered as a value between 0 and 4095 and is added directly to the calculated output. Bias is not required for most applications and may be left at 0. It must be the range : <math>0 \leq \text{Bias} \leq 4095</math></li> <li>● PR+4 = High Integral Wind_up Limit (HIWL) : The user must load this register with the output value, (0 to 4095), at which the loop should go into "anti-reset wind-up" mode. Anti-reset wind-up consists of solving the digital approximation for the integral value. For most applications this should be set to 4095. It must be the range : <math>0 \leq HIWL \leq 4095</math></li> <li>● PR+5 = Low Integral Wind_up Limit (LIWL) : The user must load this register with the output value, (0 to 4095), at which the loop should go into "anti-reset wind-up" mode. It functions in the same manner as PR+4. For most applications this should be set to 0. It must be the range : <math>0 \leq LIWL \leq 4095</math></li> <li>● PR+6 = PID Method : <ul style="list-style-type: none"> <li>=0 , Standard PID method;</li> <li>=1 , Minimum Overshoot Method;</li> </ul> Method 0 is preferred because most applications use PI control (<math>Td=0</math>). The user may try method 1 when using PID control and the result is not stable.</li> <li>● When the setting mentioned above comes error, it will not perform PID operation and the output indication "ERR" will be ON.</li> </ul>		

## Mathematics instructions

FUN 30 PID	Convenient instruction of PID loop operation	FUN 30 PID
<p>● <b>Description of operand WR (Working registers) :</b></p> <ul style="list-style-type: none"> <li>● WR+0 = Loop status register :           <ul style="list-style-type: none"> <li>Bit0 =0 , Manual operation mode</li> <li>=1 , Auto mode</li> <li>Bit1 : This bit will be a 1 during the scan the solution is being solved, and it is ON for a scan time.</li> <li>Bit2=1 , Bumpless transfer</li> <li>Bit4 : The status of "ERR" indication</li> <li>Bit5 : The status of "HA" indication</li> <li>Bit6 : The status of "LA" indication</li> </ul> </li> <li>● WR+1 = Loop timer register : This register stores the cyclic timer reading from the system's 1ms cyclic timer each time the loop is solved. The elapsed time is calculated by calculating the difference between the current reading of the system's 1ms cyclic timer and the value stored in this register. This difference is compared to <math>10 \times</math> the solution interval. If the difference is greater than or equal to the solution interval, the loop should be solved this scan.</li> <li>● WR+2 = Low order integral summation : This register stores the low order 16 bits of the 32 bit sum created by the integral term.</li> <li>● WR+3 = High order integral summation : This register stores the high order 16 bits of the 32 bit sum created by the integral term.</li> <li>● WR+4 = Process variable - previous solution : The raw analog input (Register SR+6) at the time the loop was last solved. This is used for the derivative control mode.</li> </ul>		

### Program example



FUN 30 PID	Convenient instruction of PID loop operation	FUN 30 PID
R999 : The setting of solution interval between calculations; for example the content of R999 is 200, it means it will perform this PID operation every 2 seconds.		R1020 : The setting of proportional band; for example the content of R1020 is 20, it means the proportional band is 2.0% and the gain is 50.
R1000 : Scaled process variable, which is the engineering unit loaded by the PID instruction every time it gets solved. A linear scaling is done on R1006 using the high and low engineering range found in R1004 and R1005.		R1021 : The setting of integral tuning constant; for example the content of R1021 is 17, it means the reset time is 6 minutes ( $100/6 \approx 17$ ).
R1001 : Setpoint, it is the desired value the loop should control at; which is entered in engineering unit. For example the span of controlling process is $0^{\circ}\text{C} \sim 500^{\circ}\text{C}$ , the setting of R1001 is equal to 100, it means the desired result is at $100^{\circ}\text{C}$ .		R1022 : The setting of derivative tuning constant; for example the content of R1022 is 0, it means PI control.
R1002 : The setting of high alarm limit; which is entered in engineering unit. The example mentioned above, if the setting of R1002 is equal to 105, it means there will have the high alarm while the loop is greater than or equal to $105^{\circ}\text{C}$ .		R1023 : The setting of the bias to the output; most applications let it be 0.
R1003 : The setting of low alarm limit; which is entered in engineering unit. The example mentioned, if the setting of R1003 is equal to 95, it means there will have the low alarm while the loop is less than or equal to $95^{\circ}\text{C}$ .		R1024 : The setting of high integral wind-up; most applications let it be 4095.
R1004 : The setting of high engineering range. The example mentioned, if the setting of R1004 is equal to 500, it means the highest value of this loop is $500^{\circ}\text{C}$ .		R1025 : The setting of low integral wind-up; most applications let it be 0.
R1005 : The setting of low engineering range. The example mentioned, if the setting of R1005 is equal to 0, it means the lowest value of this loop is $0^{\circ}\text{C}$ .		R1026 : The setting of PID method; most applications let it be 0.
R1006 : Raw analog measurement; it is the value that the content of analog input register (R3840~R3903) is added by the offset of 2048.		R1030 = Loop status register Bit0 =0, Manual operation mode =1, Auto operation mode
R1007 : Offset of process variable; let it be 0 if the raw analog signal and the span of the analog input module are all $0 \sim 10\text{V}$ .		Bit1 : This bit will be a 1 during the scan the solution is being solved, and it is ON for a scan time. Bit2=1 , Bumpless transfer Bit4 : The status of "ERR" indication Bit5 : The status of "HA" indication Bit6 : The status of "LA" indication
		R1031~R1034: They are the working registers, please refer to the description of operand WR.