

# WECON

# Programming



**WECON Technology Co., Ltd.**

Website: <http://www.we-con.com.cn/en>

Technical Support: [chengxf@we-con.com.cn](mailto:chengxf@we-con.com.cn)

Skype: Jason.chen842

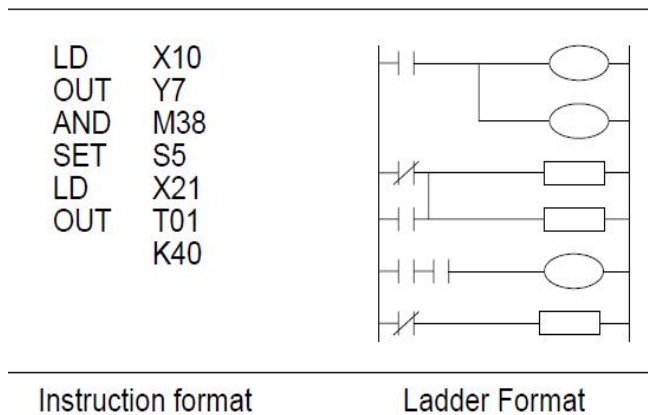
Phone: 86-591-87868869

# Basic Program Instructions

## 1. What is a Program?

A program is a connected series of instructions written in a language that the PLC can understand. There are two forms of program format; instruction, ladder.

panels only work with instruction format while most graphic programming tools will work with both instruction and ladder format.



## 2. Outline of Basic Devices Used in Programming

There are six basic programming devices. Each device has its own unique use. To enable quick and easy identification each device is assigned a single reference letter;

- X: This is used to identify all direct, physical inputs to the PLC.
- Y: This is used to identify all direct, physical outputs from the PLC.
- T: This is used to identify a timing device which is contained within the PLC.
- C: This is used to identify a counting device which is contained within the PLC.
- M and S: These are used as internal operation flags within the PLC.

All of the devices mentioned above are known as 'bit devices'. This is a descriptive title telling the user that these devices only have two states; ON or OFF, 1 or 0.

## 3. How to Read Ladder Logic

Ladder logic is very closely associated to basic relay logic. There are both contacts and coils that can be loaded and driven in different configurations. However, the basic principle remains the same.

A coil drives direct outputs of the PLC (ex. a Y device) or drives internal timers, counters or flags (ex. T, C, M and S devices). Each coil has associated contacts. These contacts are available in both "normally open" (NO) and "normally closed" (NC) configurations.

The term "normal(ly)" refers to the status of the contacts when the coil is not energized.

Using a relay analogy, when the coil is OFF, a NO contact would have no current flow, that is, a load being supplied through a NO contact would not operate. However, a NC contact would allow current to flow, hence the connected load would be active.

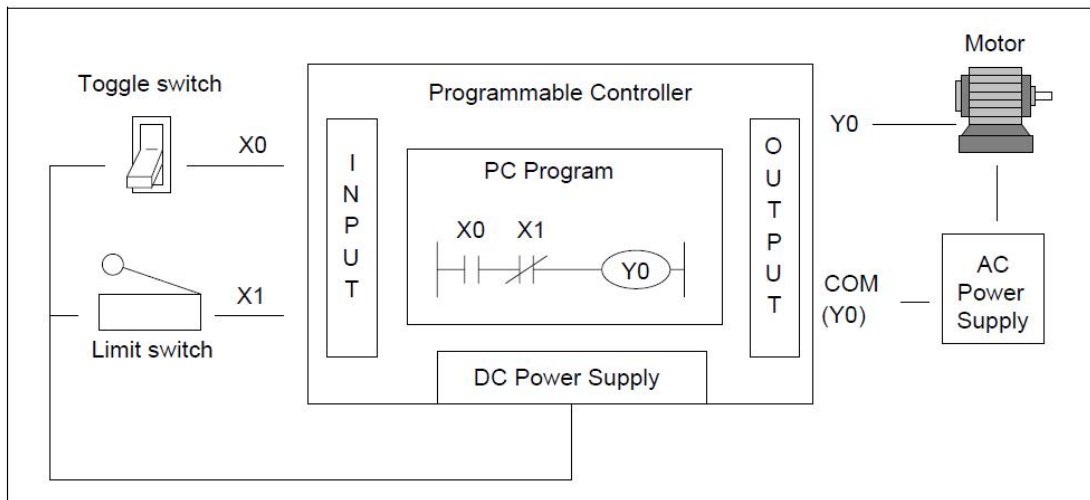
Activating the coil reverses the contact status, that is, the current would flow in a NO

contact and a NC contact would inhibit the flow.

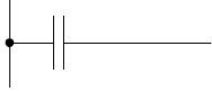
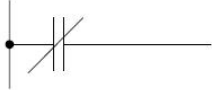
Physical inputs to the PLC (X devices) have no programmable coil. These devices may only be used in a contact format (NO and NC types are available).

#### Example:

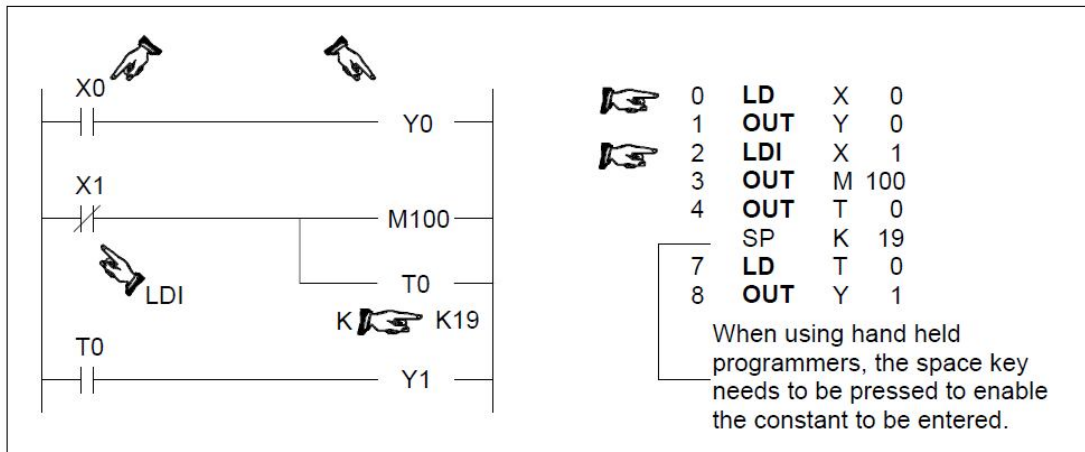
Because of the close relay association, ladder logic programs can be read as current flowing from the left vertical line to the right vertical line. This current must pass through a series of contact representations such as X0 and X1 in order to switch the output coil Y0 ON. Therefore, in the example shown, switching X0 ON causes the output Y0 to also switch ON. If however, the limit switch X1 is activated, the output Y0 turns OFF. This is because the connection between the left and the right vertical lines breaks so there is no current flow.



#### 4. Load, Load Inverse

Mnemonic	Function	Format	Devices	Program steps
LD (LoaD)	Initial logical operation contact type NO (normally open)		X, Y, M, S, T, C	1
LDI (LoaD Inverse)	Initial logical operation contact type NC (normally closed)		X, Y, M, S, T, C	1

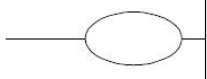
#### Program example:



Basic points to remember:

- Connect the LD and LDI instructions directly to the left hand bus bar.
- Or use LD and LDI instructions to define a new block of program when using the ORB

## 5.Out

Mnemonic	Function	Format	Devices	Program steps
OUT (OUT)	Final logical operation type coil drive		Y, M, S, T, C	Y, M:1 S, special M coils: 2 T:3 C (16 bit): 3 C (32 bit): 5

Basic points to remember:

- Connect the OUT instruction directly to the right hand bus bar.
- It is not possible to use the OUT instruction to drive 'X' type input devices.
- It is possible to connect multiple OUT instructions in parallel (for example see the previous page; M100/T0 configuration)

### 5.1 Timer and Counter Variations

When configuring the OUT instruction for use as either a timer (T) or counter (C) a constant must also be entered. The constant is identified by the letter "K" (for example see previous page; T0 K19).

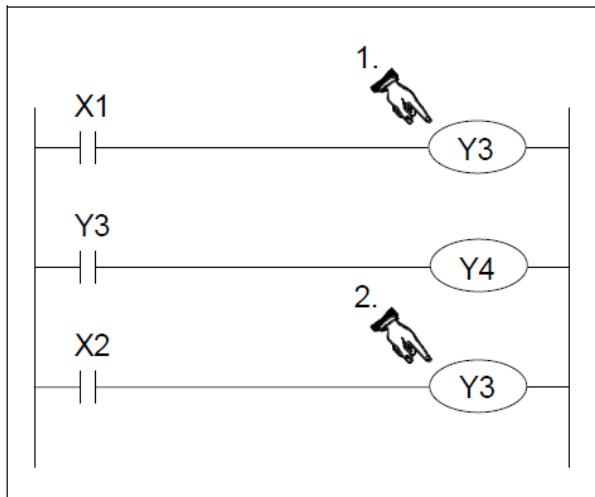
In the case of a timer, the constant "K" holds the duration data for the timer to operate, i.e. if a 100 msec timer has a constant of "K100" it will be (1005 100 msec) 10 seconds before the timer coil activates.

With counters, the constant identifies how many times the counter must be pulsed or triggered before the counter coil activates. For example, a counter with a constant of "8" must be triggered 8 times before the counter coil finally energizes.

The following table identifies some basic parameter data for various timers and counters;

Timer/Counter	Setting constant K	Actual setting	Program steps
1 msec Timer	1 to 32,767	0.001 to 32.767 sec	3
10 msec Timer		0.01 to 327.67 sec	
100 msec Timer		0.1 to 3276.7 sec	
16 bit Counter	1 to 32,767	1 to 32,767	5
32 bit Counter	-2,147,483,648 to 2,147,483,647	-2,147,483,648 to 2,147,483,647	

## 5.2 Double Coil Designation



Double or dual coiling is not a recommended practice. Using multiple output coils of the same device can cause the program operation to become unreliable. The example program shown opposite identifies a double coil situation; there are two Y3 outputs. The following sequence of events will occur when inputs X1 = ON and X2 = OFF;

1. The first Y3 turns ON because X1 is ON. The contacts associated with Y3 also energize when the coil of output Y3 energizes. Hence, output Y4 turns ON.
2. The last and most important line in this program looks at the status of input X2. If this is NOT ON then the second Y3 coil does NOT activate. Therefore the status of the Y3 coil updates to reflect this new situation, i.e. it turns OFF. The final outputs are then Y3 = OFF and Y4 = ON.

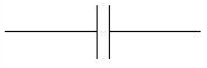

### Use of dual coils:

- Always check programs for incidents of dual coiling. If there are dual coils the program will not operate as expected - possibly resulting in unforeseen physical

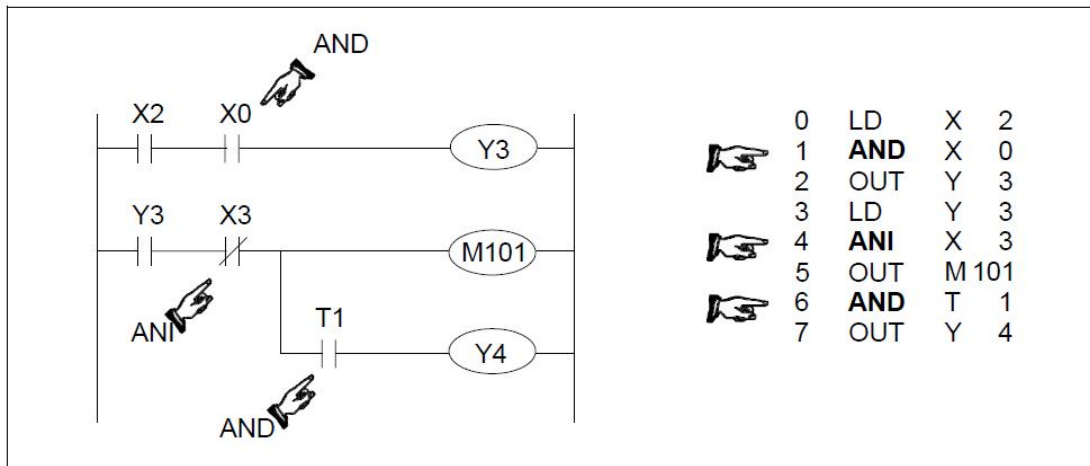
### The last coil effect:

- In a dual coil designation, the coil operation designated last is the effective coil. That is, it is the status of the previous coil that dictates the behavior at the current point in the program.

## 6. And, And Inverse

Mnemonic	Function	Format	Devices	Program steps
AND (AND)	Serial connection of NO (normally open) contacts		X, Y, M, S, T, C	1
ANI (AND Inverse)	Serial connection of NC (normally closed) contacts		X, Y, M, S, T, C	1

Program example:





Basic points to remember:

- Use the AND and ANI instructions for serial connection of contacts. As many contacts as required can be connected in series (see following point headed "Peripheral limitations").
- The output processing to a coil, through a contact, after writing the initial OUT instruction is called a "follow-on" output (for an example see the program above; OUT Y4). Followon outputs are permitted repeatedly as long as the output order is correct.

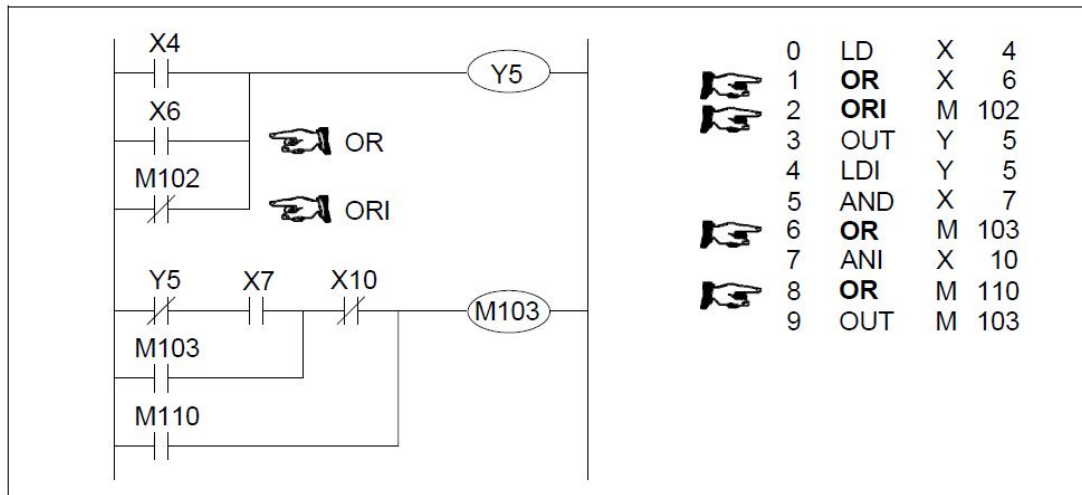
#### Peripheral limitations:

- The PLC has no limit to the number of contacts connected in series or in parallel. However, some programming panels, screens and printers will not be able to display or print the program if it exceeds the limit of the hardware. It is preferable for each line or rung of ladder program to contain up to a maximum of 10 contacts and 1 coil. Also, keep the number of follow-on outputs to a maximum of 24.

### 7. Or, Or Inverse

Mnemonic	Function	Format	Devices	Program steps
OR (OR)	Parallel connection of NO (normally open) contacts		X, Y, M, S, T, C	1
ORI (OR Inverse)	Parallel connection of NC (normally closed) contacts		X, Y, M, S, T, C	1

Program example:



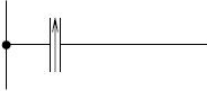

Basic points to remember:

- Use the OR and ORI instructions for parallel connection of contacts. To connect a block that contains more than one contact connected in series to another circuit block in parallel, use an ORB instruction.
- Connect one side of the OR/ORI instruction to the left hand bus bar.

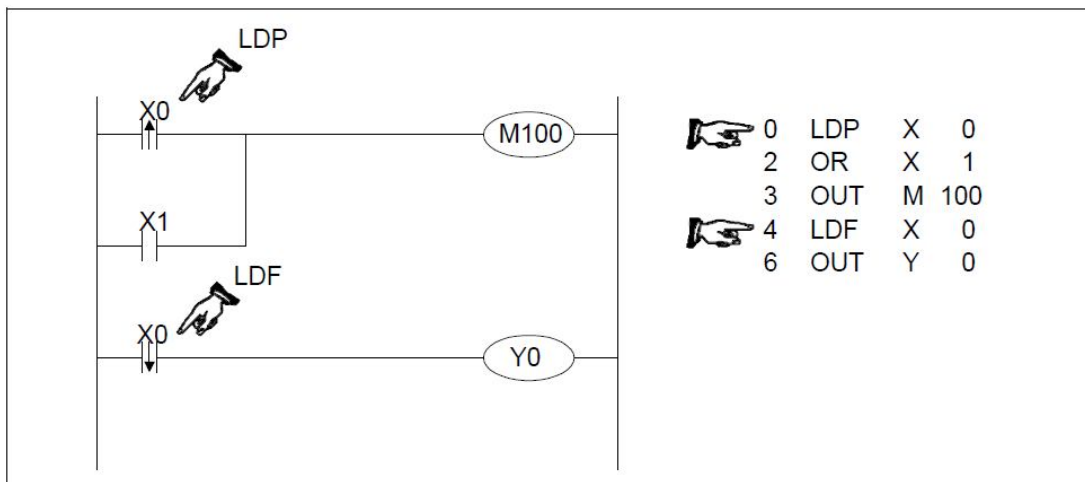
### Peripheral limitations:

- The PLC has no limit to the number of contacts connected in series or in parallel. However, some programming panels, screens and printers will not be able to display or print the program if it exceeds the limit of the hardware. It is preferable for each line or rung of ladder program to contain up to a maximum of 10 contacts and 1 coil. Also keep number of follow-on outputs to a maximum of 24.

## 8. Load Pulse, Load Trailing Pulse

Mnemonic	Function	Format	Devices	Program steps
LDP (Load Pulse)	Initial logical operation - Rising edge pulse		X, Y, M, S, T, C	2
LDF (Load Falling pulse)	Initial logical operation Falling / trailing edge pulse		X, Y, M, S, T, C	2



Program example:



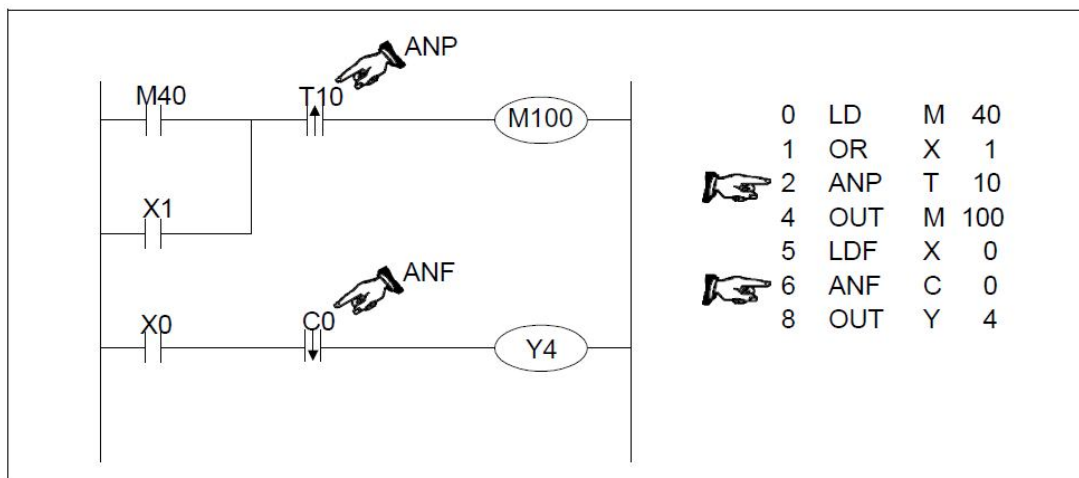
Basic points to remember:

- Connect the LDP and LDF instructions directly to the left hand bus bar.
- Or use LDP and LDF instructions to define a new block of program when using the ORB and ANB instructions (see later sections).
- LDP is active for one program scan after the associated device switches from OFF to ON.
- LDF is active for one program scan after the associated device switches from ON to OFF.

## 9. And Pulse, And Trailing Pulse

Mnemonic	Function	Format	Devices	Program steps
ANP (ANd Pulse)	Serial connection of Rising edge pulse		X, Y, M, S, T, C	2
ANF (ANd Falling pulse)	Serial connection of Falling / trailing edge pulse		X, Y, M, S, T, C	2



Program example:



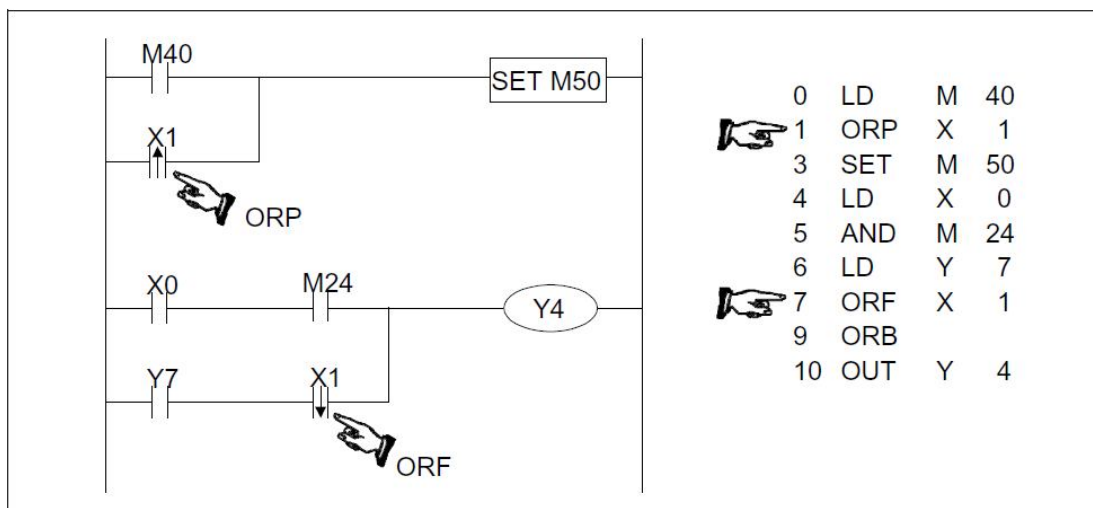
Basic points to remember:

- Use the ANDP and ANDF instructions for the serial connection of pulse contacts.
- Usage is the same as for AND and ANI; see earlier.
- ANP is active for one program scan after the associated device switches from OFF to ON.
- ANF is active for one program scan after the associated device switches from ON to OFF.

## 10. Or Pulse, Or Trailing Pulse

Mnemonic	Function	Format	Devices	Program steps
ORP (OR Pulse)	Parallel connection of Rising edge pulse		X, Y, M, S, T, C	2
ORF (OR Falling pulse)	Parallel connection of Falling / trailing edge pulse		X, Y, M, S, T, C	2

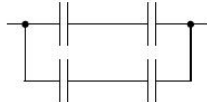
### Program example:



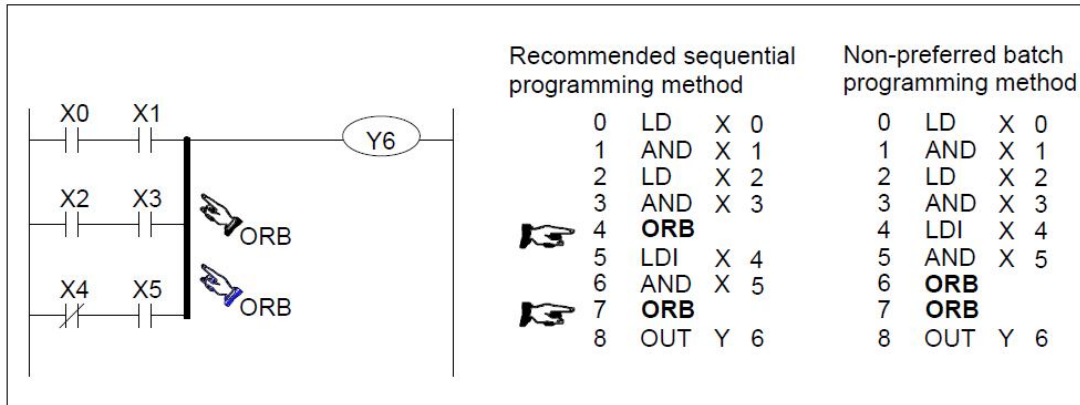
Basic points to remember:

- Use the ORP and ORF instructions for the parallel connection of pulse contacts.
- Usage is the same as for OR and ORI; see earlier.
- ORP is active for one program scan after the associated device switches from OFF to ON.
- ORF is active for one program scan after the associated device switches from ON to OFF.

## 2.11 Or Block

Mnemonic	Function	Format	Devices	Program steps
ORB (OR Block)	Parallel connection of multiple contact circuits		N/A	1

### Program example:



Basic points to remember:

- An ORB instruction is an independent instruction and is not associated with any device number.
- Use the ORB instruction to connect multi-contact circuits (usually serial circuit blocks) to the preceding circuit in parallel. Serial circuit blocks are those in which more than one contact connects in series or the ANB instruction is used.
- To declare the starting point of the circuit block use a LD or LDI instruction. After completing the serial circuit block, connect it to the preceding block in parallel using the ORB instruction.

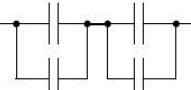
#### Batch processing limitations:

- When using ORB instructions in a batch, use no more than 8 LD and LDI instructions in the definition of the program blocks (to be connected in parallel). Ignoring this will result in a program error (see the right most program listing).

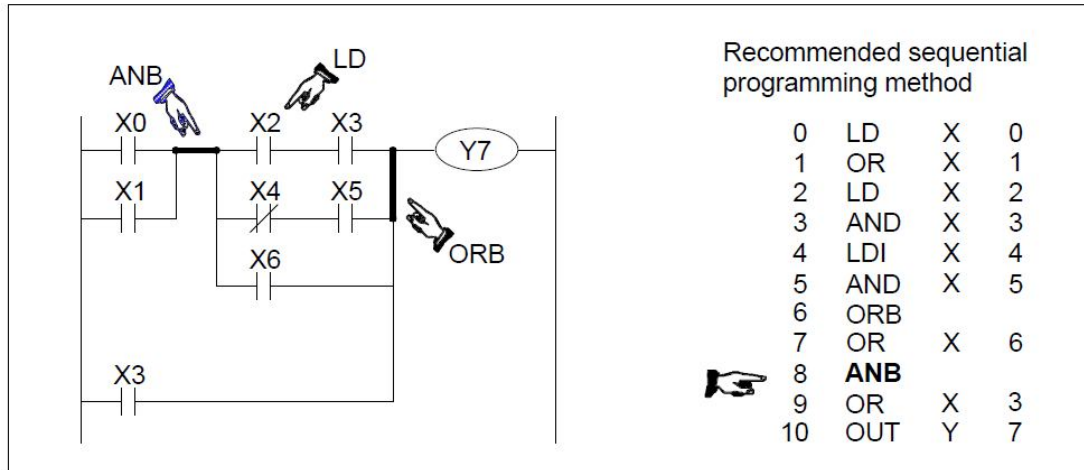
#### Sequential processing limitations:

- There are no limitations to the number of parallel circuits when using an ORB instruction in the sequential processing configuration (see the left most program listing).

## 12. And Block

Mnemonic	Function	Format	Devices	Program steps
ANB (ANd Block)	Serial connection of multiple parallel circuits		N/A	1

Program example:



Basic points to remember:

- An ANB instruction is an independent instruction and is not associated with any device number
- Use the ANB instruction to connect multi-contact circuits (usually parallel circuit blocks) to the preceding circuit in series. Parallel circuit blocks are those in which more than one contact connects in parallel or the ORB instruction is used.
- To declare the starting point of the circuit block, use a LD or LDI instruction. After completing the parallel circuit block, connect it to the preceding block in series using the ANB instruction.

#### Batch processing limitations:

- When using ANB instructions in a batch, use no more than 8 LD and LDI instructions in the definition of the program blocks (to be connected in parallel). Ignoring this will result in a program error (see ORB explanation for example).

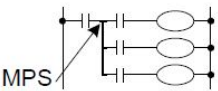
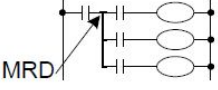
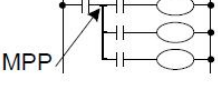
#### Sequential processing limitations:

- It is possible to use as many ANB instructions as necessary to connect a number of parallel circuit blocks to the preceding block in series (see the program listing).

FX Series Programmable Controllers **Basic Program Instructions 2**

2-13

### 13. MPS, MRD and MPP

Mnemonic	Function	Format	Devices	Program steps
MPS (Point Store)	Stores the current result of the internal PLC operations		N/A	1
MRD (Read)	Reads the current result of the internal PLC operations		N/A	1
MPP (PoP)	Pops (recalls and removes) the currently stored result		N/A	1

Basic points to remember:

- Use these instructions to connect output coils to the left hand side of a contact.

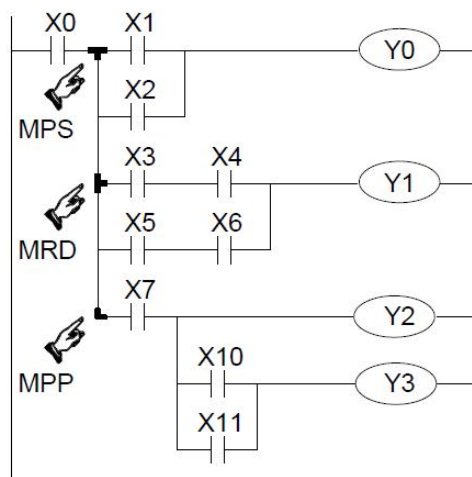
Without these instructions connections can only be made to the right hand side of the last contact.

- MPS stores the connection point of the ladder circuit so that further coil branches can recall the value later.
- MRD recalls or reads the previously stored connection point data and forces the next contact to connect to it.
- MPP pops (recalls and removes) the stored connection point. First, it connects the next contact, then it removes the point from the temporary storage area.
- For every MPS instruction there **MUST** be a corresponding MPP instruction.
- The last contact or coil circuit must connect to an MPP instruction.
- At any programming step, the number of active MPS-MPP pairs must be no greater than 11.

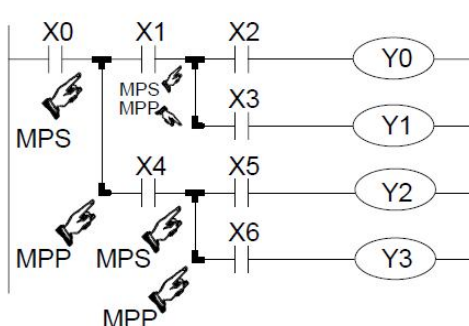
### MPS, MRD and MPP usage:

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

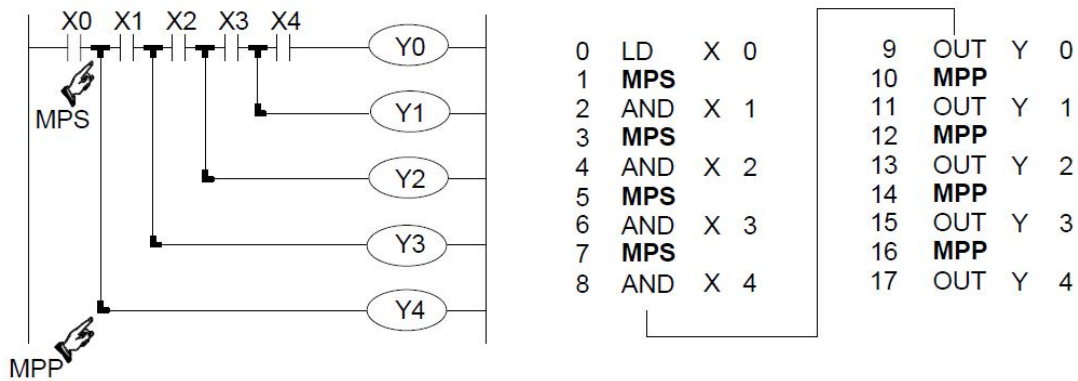
### Multiple program examples:



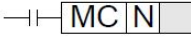
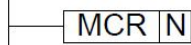
0	LD	X 0	
1	<b>MPS</b>		
2	LD	X 1	
3	OR	X 2	
4	ANB		
5	OUT	Y 0	
6	<b>MRD</b>		
7	LD	X 3	
8	AND	X 4	
9	LD	X 5	
10	AND	X 6	
11	ORB		
12	ANB		
13	OUT	Y 1	
14	<b>MPP</b>		
15	AND	X 7	
16	OUT	Y 2	
17	LD	X 10	
18	OR	X 11	
19	ANB		
20	OUT	Y 3	



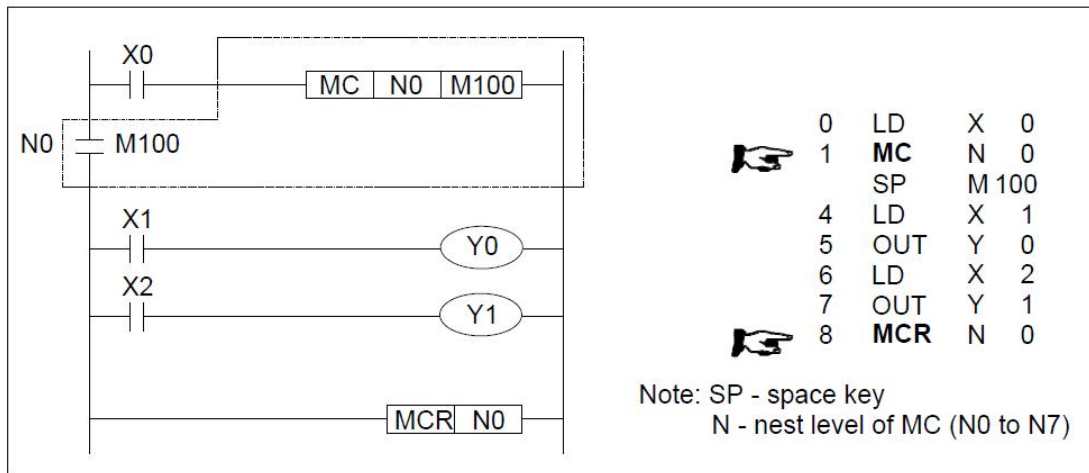
0	LD	X 0	
1	<b>MPS</b>		
2	AND	X 1	
3	<b>MPS</b>		
4	AND	X 2	
5	OUT	Y 0	
6	<b>MPP</b>		
7	AND	X 3	
8	OUT	Y 1	
9	<b>MPP</b>		
10	AND	X 4	
11	<b>MPS</b>		
12	AND	X 5	
13	OUT	Y 2	
14	<b>MPP</b>		
15	AND	X 6	
16	OUT	Y 3	



#### 14. Master Control and Reset

Mnemonic	Function	Format	Devices	Program steps
MC (Master Control)	Denotes the start of a master control block		Y, M (no special M coils allowed) N denotes the nest level (N0 to N7)	3
MCR (Master Control Reset)	Denotes the end of a master control block		N denotes the nest level (N0 to N7) to be reset.	2

#### Program example:



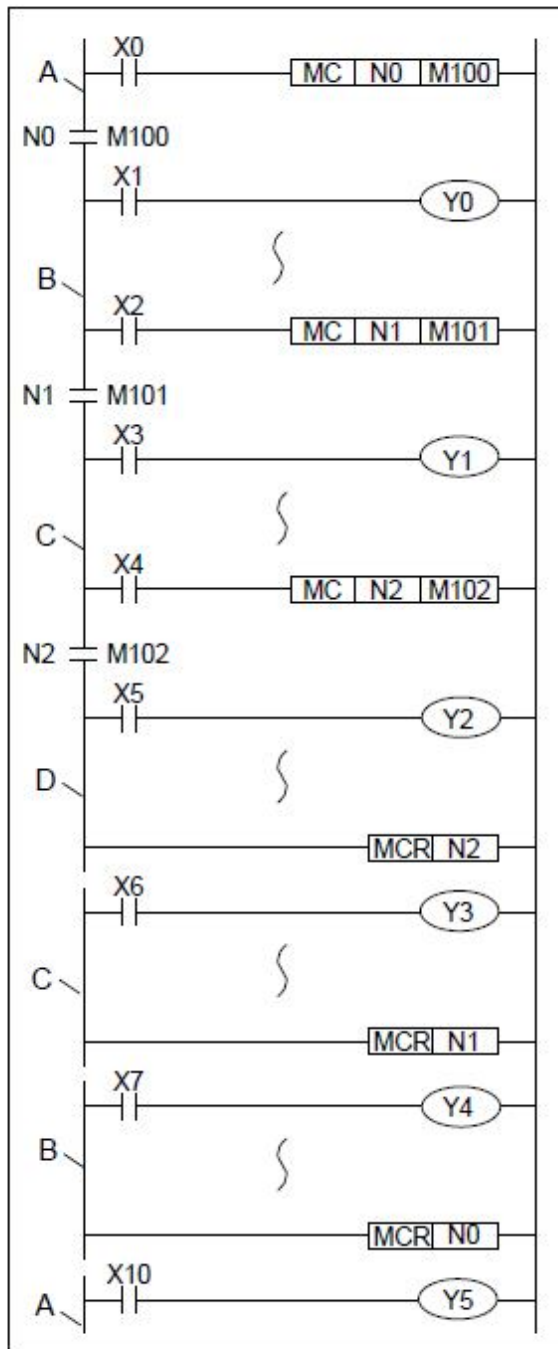
#### Basic points to remember:

- After the execution of an MC instruction, the bus line (LD, LDI point) shifts to a point after the MC instruction. An MCR instruction returns this to the original bus line.
- The MC instruction also includes a nest level pointer N. Nest levels are from the range N0 to N7 (8 points). The top nest level is '0' and the deepest is '7'.
- The MCR instruction resets each nest level. When a nest level is reset, it also resets ALL deeper nest levels. For example, MCR N5 resets nest levels 5 to 7.
- When input X0=ON, all instructions between the MC and the MCR instruction execute.
- When input X0=OFF, none of the instruction between the MC and MCR instruction

execute; this resets all devices except for retentive timers, counters and devices driven by SET/RST instructions.

- The MC instruction can be used as many times as necessary, by changing the device number Y and M. Using the same device number twice is processed as a double coil (see section 2.5.2). Nest levels can be duplicated but when the nest level resets, ALL occurrences of that level reset and not just the one specified in the local MC.

#### Nested MC program example:



#### Nested MC program example:

Level N0: Bus line (B) active when X0 is ON.

Level N1: Bus line (C) active when both X0 and X2 are ON.

Level N2: Bus line (D) active when X0, X2 and X4 are ON.

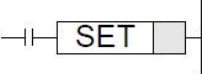

Level N1: MCRN2 executes and restores bus line (C). If the MCR had reset N0 then the original bus bar (A) would now be active as all master controls below nest level 0 would reset.

Level N0: MCRN1 executes and restores bus line (B).

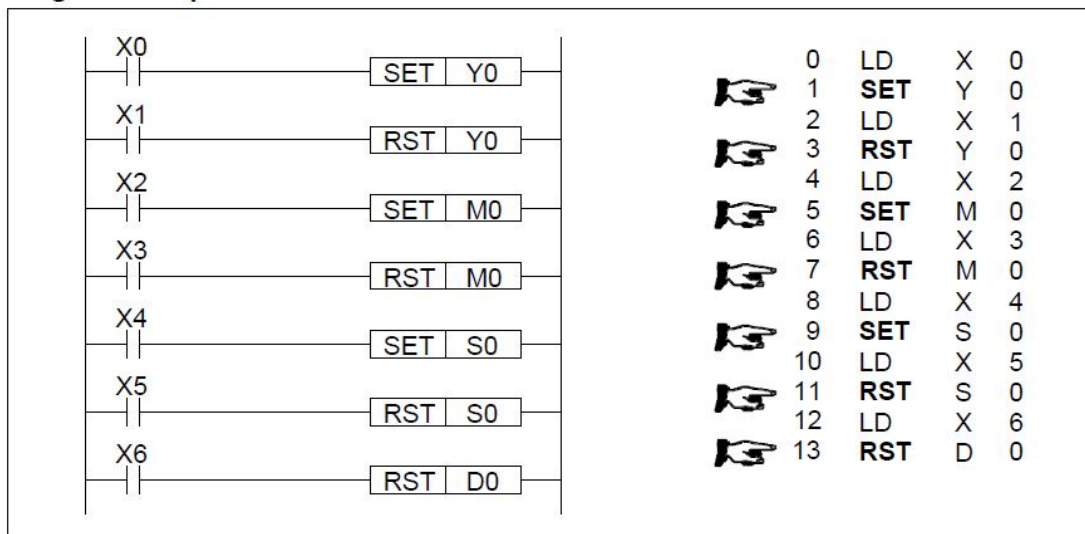
Initial state: MCR N0 executes and restores the initial bus line (A).

Output Y5 turns ON/OFF according to the ON/OFF state of X10, regardless of the ON/OFF status of inputs X0, X2 or X4.

## 15. Set and Reset

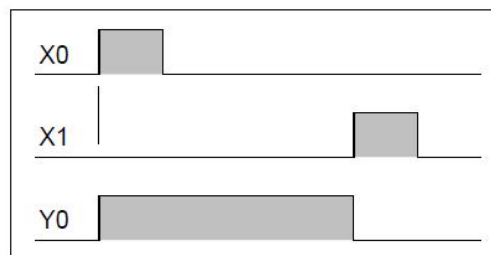
Mnemonic	Function	Format	Devices	Program steps
SET (SET)	Sets a bit device permanently ON		Y, M, S	Y,M:1 S, special M coils:2
RST (ReSeT)	Resets a bit device permanently OFF		Y, M, S, D, V, Z (see section 2.16 for timers and counters T,C)	D, special D registers, V and Z:3

### Program example:

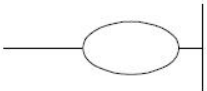



### Basic points to remember:

- Turning ON X0 causes Y0 to turn ON. Y0 remains ON even after X0 turns OFF.
- Turning ON X1 causes Y0 to turn OFF. Y0 remains OFF even after X1 turns OFF.
- SET and RST instructions can be used for the same device as many times as necessary. However, the last instruction activated determines the current status.
- It is also possible to use the RST instruction to reset the contents of data devices such as data registers, index registers etc. The effect is similar to moving 'K0' into the data device.

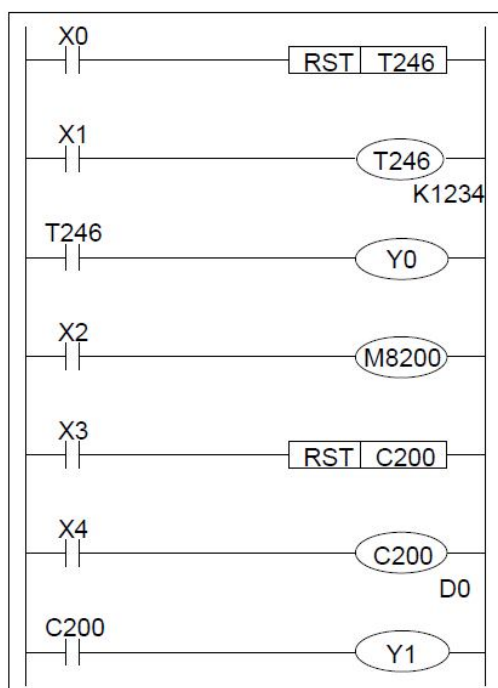


## 16. Timer, Counter (Out & Reset)

Mnemonic	Function	Format	Devices	Program steps
OUT (OUT)	Driving timer or counter coils		T, C	32 bit counters:5 Others: 3
RST (ReSeT)	Resets timer and counter, coils contacts and current values		T, C (see section 2.15 for other resettable devices)	

### Program example:

#### Program example:



#### 2.16.1 Basic Timers, Retentive Timers And Counters

These devices can all be reset at any time by driving the RST instruction (with the number of the device to be reset).

On resetting, all active contacts, coils and current value registers are reset for the selected device. In the example, T246, a 1msec retentive timer, is activate while X1 is ON. When the current value of T246 reaches the preset 'K' value, i.e. 1234, the timer coil for T246 will be activated. This drives the NO contact ON. Hence, Y0 is switched ON. Turning ON X0 will reset timer T246 in the manner described previously.

Because the T246 contacts are reset, the output Y0 will be turned OFF.

### 16.2 Normal 32 bit Counters

The 32 bit counter C200 counts (up-count, down-count) according to the ON/OFF state of M8200. In the example program shown on the previous page C200 is being used to count the number of OFF ~ ON cycles of input X4.

The output contact is set or reset depending on the direction of the count, upon reaching a value equal (in this example) to the contents of data registers D1,D0 (32 bit setting data is required for a 32 bit counter).

The output contact is reset and the current value of the counter is reset to '0' when input X3 is turned ON.

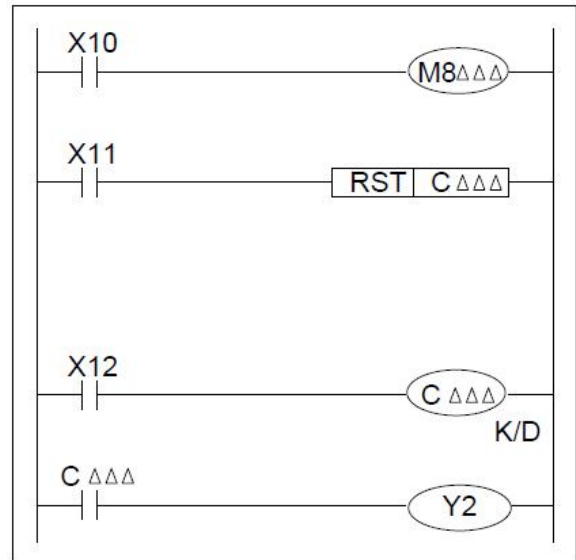
### 16.3 High Speed Counters

High speed counters have selectable count directions. The directions are selected by driving the appropriate special auxiliary M coil. The example shown to the right works in the following manner; when X10 is ON, counting down takes place. When X10 is OFF counting up takes place.

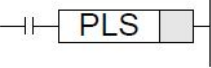
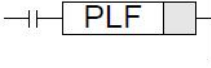
In the example the output contacts of counter C△△△ and its associated current count values are reset to “0” when X11 is turned ON. When X12 is turned ON the driven counter is enabled. This means it will be able to start counting its assigned input signal (this will not be X12 - high speed counters are assigned special input signals, please see page 4-22).

#### Availability of devices:

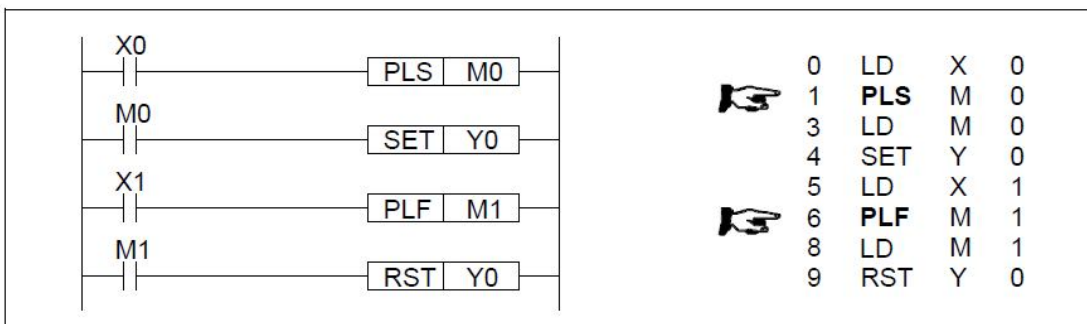
- Not all devices identified here are available on all programmable controllers. Ranges of active devices may vary from PLC to PLC. Please check the specific availability of these devices on the selected PLC before use. For more information on high speed counters please see page 4-22. For PLC device ranges please see chapter 8.



## 2.17 Leading and Trailing Pulse

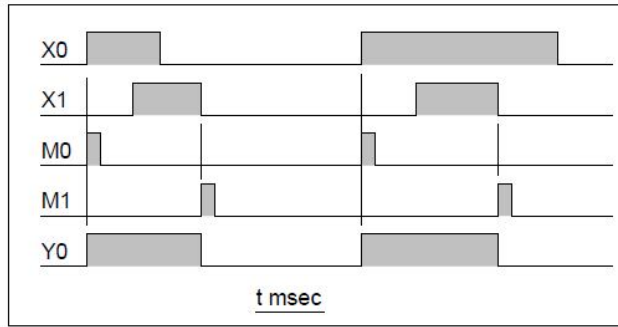
Mnemonic	Function	Format	Devices	Program steps
PLS (PuLSe)	Rising edge pulse		Y, M (no special M coils allowed)	2
PLF (PuLSe Falling)	Falling / trailing edge pulse		Y, M (no special M coils allowed)	2

#### Program example:




Basic points to remember:

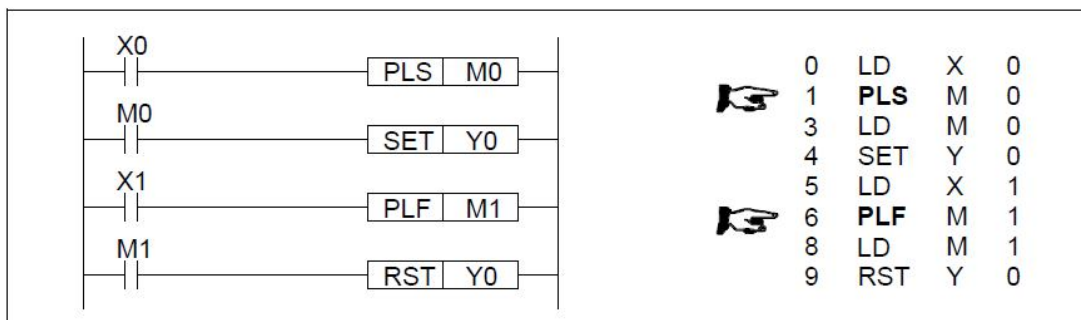
- When a PLS instruction is executed, object devices Y and M operate for one operation cycle after the drive input signal has turned ON.
- When a PLF instruction is executed, object devices Y and M operate for one operation cycle after the drive input signal has turned OFF.
- When the PLC status is changed from RUN to STOP and back to RUN with the input signals still ON, PLS M0 is operated again. However, if an M coil which is battery backed (latched) was used instead of M0 it would not re-activate. For the battery backed device to be re-pulsed, its driving input (ex. X0) must be switched OFF during the RUN/STOP/RUN sequence before it will be pulsed once more.



## 18. Inverse

Mnemonic	Function	Format	Devices	Program steps
INV (Inverse)	Invert the current result of the internal PLC operations		N/A	1

Program example:



Basic points to remember:

- The INV instruction is used to change (invert) the logical state of the current ladder network at the inserted position.
- Usage is the same as for AND and ANI; see earlier.

### Usages for INV

- Use the invert instruction to quickly change the logic of a complex circuit.
- It is also useful as an inverse operation for the pulse contact instructions LDP, LDF, ANP, etc.

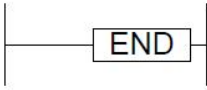
## 19. No Operation

Mnemonic	Function	Format	Devices	Program steps
NOP (No Operation)	No operation or null step	N/A	N/A	1

Basic points to remember:

- Writing NOP instructions in the middle of a program minimizes step number changes when changing or editing a program.
- It is possible to change the operation of a circuit by replacing programmed instructions with NOP instructions.
- Changing a LD, LDI, ANB or an ORB instruction with a NOP instruction will change the circuit considerably; quite possibly resulting in an error being generated.
- After the program 'all clear operation' is executed, all of the instructions currently in the program are over written with NOP's.

## 20. End

Mnemonic	Function	Format	Devices	Program steps
END (END)	Forces the current program scan to end		N/A	1

Basic points to remember:

- Placing an END instruction in a program forces that program to end the current scan and carry out the updating processes for both inputs and outputs.
- Inserting END instructions in the middle of the program helps program debugging as the section after the END instruction is disabled and isolated from the area that is being checked. Remember to delete the END instructions from the blocks which have already been checked.
- When the END instruction is processed the PCs watchdog timer is automatically refreshed.

### A program scan:

- A program scan is a single processing of the loaded program from start to finish, This includes updating all inputs, outputs and watchdog timers. The time period for one such process to occur is called the scan time. This will be dependent upon program length and complexity. Immediately the current scan is completed the next scan begins. The whole process is a continuous cycle. Updating of inputs takes place at the beginning of each scan while all outputs are updated at the end of the scan.