

**TP03 Serial Programmable Controller****Basic Program Instructions 1**

1	Basic Program Instructions .....	2
1.1	What is a Program? .....	2
1.2	Outline of Basic Devices Used in Programming.....	2
1.3	How to Read Ladder Logic .....	3
1.4	Load, Load Inverse.....	4
1.5	Out.....	5
1.5.1	Timer and Counter Variations.....	5
1.5.2	Double Coil Designation .....	5
1.6	And, And Inverse.....	7
1.7	OR, OR Inverse .....	8
1.8	Load Pulse, Load Trailing Pulse.....	9
1.8.1	Single Operation flags M2800 to M3071:.....	9
1.9	And Pulse, And Trailing Pulse .....	10
1.10	Or Pulse, Or Trailing Pulse.....	11
1.11	Or Block .....	12
1.12	And Block .....	13
1.13	MPS, MRD and MPP.....	14
1.14	Master Control and Reset.....	16
1.15	Set and Reset .....	18
1.16	Timer, Counter (Out & Reset).....	19
1.16.1	Basic Timers, Retentive Timers And Counters.....	19
1.16.2	Normal 32 bit Counters .....	19
1.16.3	High Speed Counters.....	20
1.17	Leading and Trailing Pulse.....	21
1.18	Inverse .....	22
1.19	No Operation .....	23
1.20	End .....	23

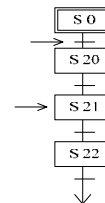
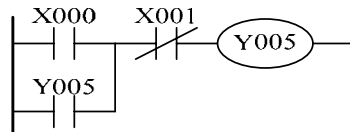
# 1 Basic Program Instructions

## 1.1 What is a Program?

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

```

0    LD    X000
1    OR    Y005
2    ANI   X002
3    OUT   Y005
  
```



Instruction format

Ladder format

SFC/STL format

## 1.2 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.

### Detailed device information:

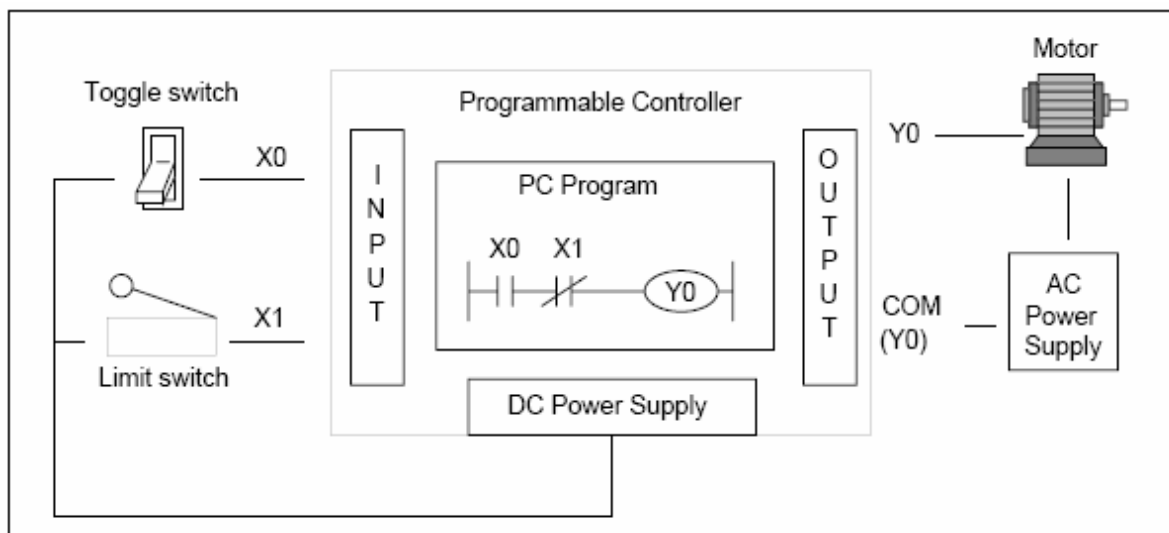
- Chapter 3 contains this information in detail. However, the above is all that is required for the rest of this chapter.

### 1.3 How to Read Ladder Logic

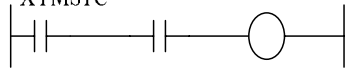
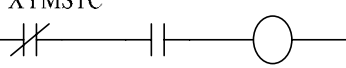
Ladder logic is very closely associated to basic relay logic. There are both contacts and coils that can be loaded and driven in different configurations. However, the basic principle remains the same. A coil drives direct outputs of the PLC (ex. a Y device) or drives internal timers, counters or flags (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:

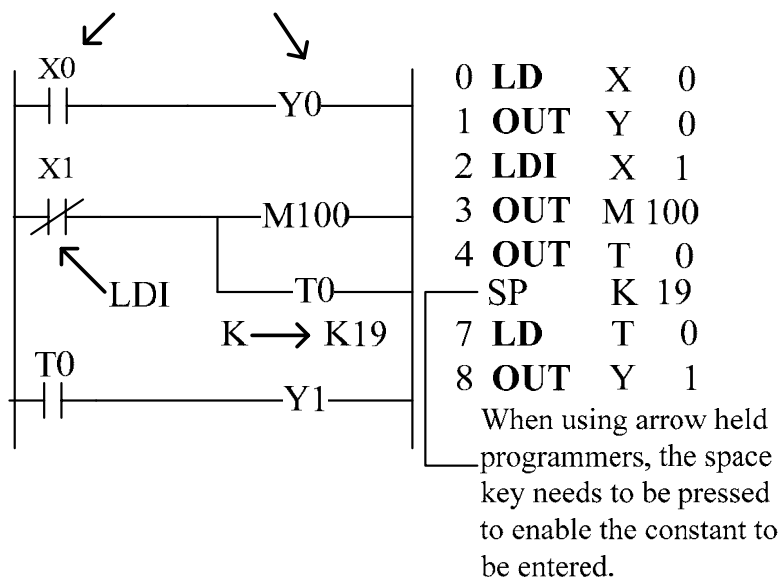
Because of the close relay association, ladder logic programs can be read as current flowing from the left vertical line to the right vertical line. This current must pass through a series of contact representations such as X0 and X1 in order to switch the output coil Y0 ON. Therefore, in the example 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.



## 1.4 Load, Load Inverse

Mnemonic	Function	Format	Devices	step
[LD]	Initial logical operation contact type NO(normally open)	XYMSTC 	X,Y,M,S,T,C	1
[LDI]	Initial logical operation contact type NC(normally closed)	XYMSTC 	X,Y,M,S,T,C	1

### Program example:



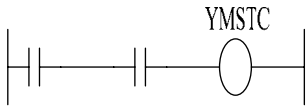
Basic points to remember:

- 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 and ANB instructions (see later sections).

### The OUT instruction:

- For details of the OUT instruction (including basic timer and counter variations) please see over the following page.

## 1.5 Out

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

Basic points to remember:

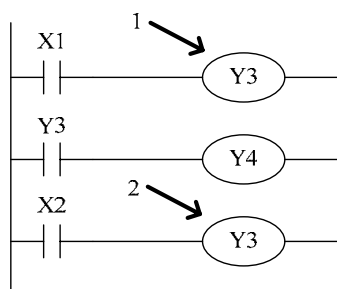
- Connect the OUT instruction directly to the right hand bus bar.
- It is not possible to use the OUT instruction to drive 'X' type input devices.
- It is possible to connect multiple OUT instructions in parallel.(for example see above:M100/T0 configuration)

### 1.5.1 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 above: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 (100x100 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
1ms Timer	1~32,767	0.001~32.676 sec	3
10ms Timer	1~32,767	0.01~327.67 sec	
100ms Timer		0.1~3,276.7 sec	
16 bit Counter	1~32,767	1~32,767	5
32 bit Counter	-2,147,483,648~ +2,147,483,647	-2,147,483,648~ +2,147,483,647	

### 1.5.2 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;

- 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.
- The last and most important line in this program looks at the status of input X2.If this

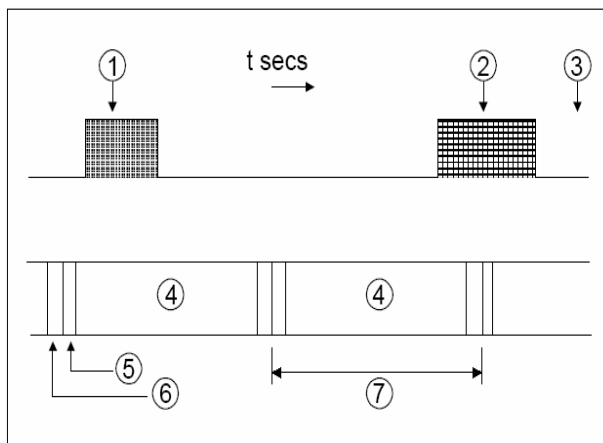
is NOT ON then the second Y3 coil does NOT activate. Therefore the status of the Y3 coil updates to reflect this new situation, i.e. it turns OFF. The final outputs are then Y3 = OFF and Y4 = ON.

#### 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.



#### Input durations:

The ON or OFF duration of the PLC inputs must be longer than the operation cycle time of the PLC.

Taking a 10 msec (standard input filter) response delay into account, the ON/OFF duration must be longer than 20 msec if the operation cycle (scan time) is 10 msec. Therefore, in this example, input pulses of more than



$$25\text{Hz} \left( \frac{1\text{sec}}{20\text{msec ON} + 20\text{msec OFF}} \right)$$

cannot be sensed.

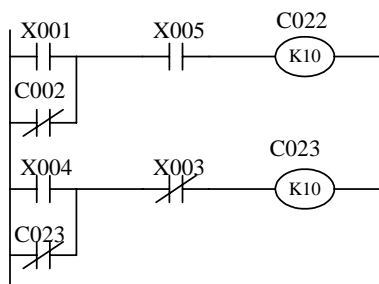
There are applied instructions provided to handle such high speed input requests.

- 1 : Input ON state NOT recognized
- 2 : Input ON state recognized
- 3 : Input OFF state NOT recognized
- 4 : 1 program processing
- 5 : Input processing
- 6 : Output processing
- 7 : A full program scan/operation cycle

## 1.6 And, And Inverse

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

Program example:



```
LD X002
ORI C022
AND X005
OUT C022 K10
```

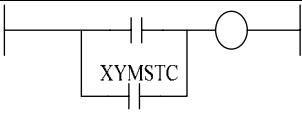
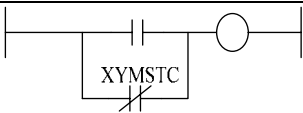
```
LD X004
ORI C023
ANI X003
OUT C023 K10
```

Basic points to remember:

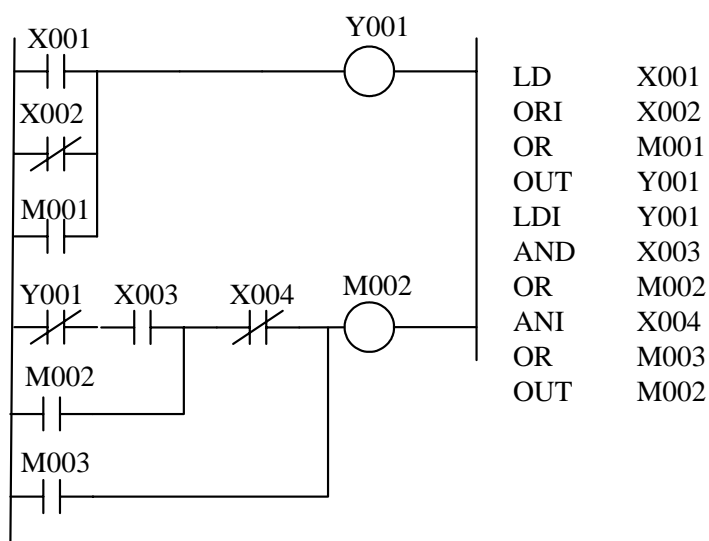
- Use the AND and ANI instructions for serial connection of contacts. As many contacts as required can be connected in series (the number of contacts in series is not limitation)
- The output processing to a coil, through a contact, after writing the

initial OUT instruction is called a “follow-on” output. Follow-on outputs are permitted repeatedly as long as the output order is correct.

## 1.7 OR, OR Inverse

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

Program example



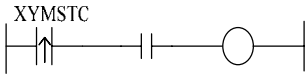
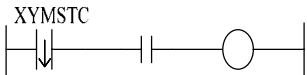
:

Basic points to remember:

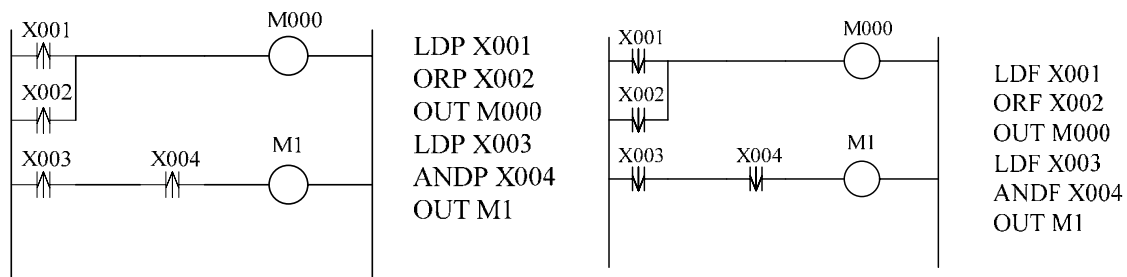
- Use the OR and ORI instructions for parallel connection of contacts. To connect a block that contains more than one contact connected in series to another circuit block in parallel, use an ORB instruction.
- Connect one side of the OR/ORI instruction to the left hand bus bar.



## 1.8 Load Pulse, Load Trailing Pulse

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

Program example:



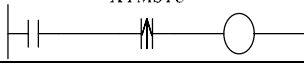
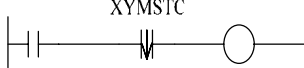
Basic points to remember:

- 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.

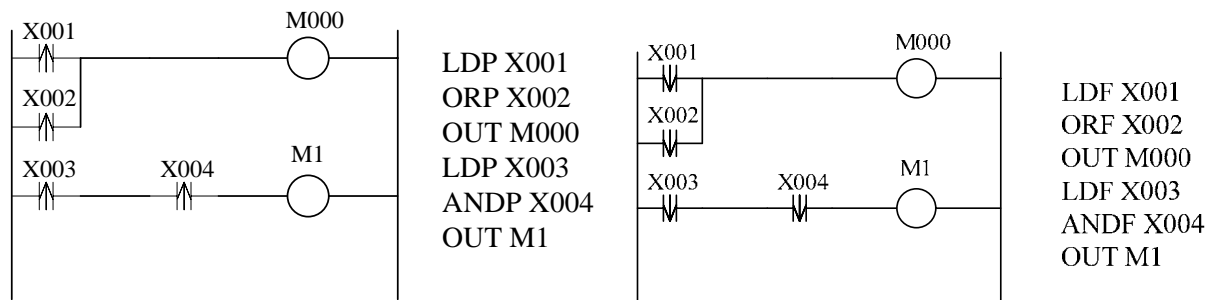
### 1.8.1 Single Operation flags M2800 to M3071:

- The pulse operation instructions, when used with auxiliary relays M2800 to M3071, only activate the first instruction encountered in the program scan, after the point in the program where the device changes. Any other pulse operation instructions will remain inactive.
- This is useful for use in STL programs (see chapter 3) to perform single step operation using a single device.
- Any other instructions (LD, AND, OR, etc.) will operate as expected.

## 1.9 And Pulse, And Trailing Pulse

Mnemonic	Function	Format	Devices	Program steps
[ANDP]	Serial connection of Rising edge pulse		X,Y,M,S ,T,C	2
[ANDF]	Serial connection of Falling/trailing edge pulse		X,Y,M,S ,T,C	2

### Program example:



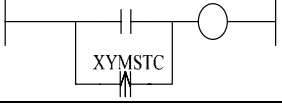
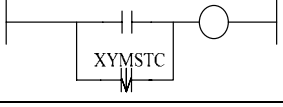
### Basic points to remember:

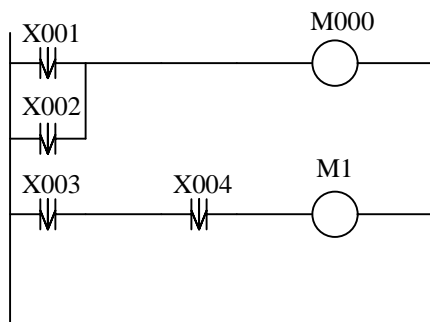
- Use the ANDP and ANDF instructions for the serial connection of pulse contacts.
- Usage is the same as for AND and ANI; see earlier.
- ANP is active for one program scan after the associated device switches from OFF to ON.
- ANF is active for one program scan after the associated device switches from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

### Single operation flags M2800 to M3071:

- When used with flags M2800 to M3071 only the first instruction will activate. For details see 1.8.1

### 1.10 Or Pulse, Or Trailing Pulse

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



```

LDF X001
ORF X002
OUT M000
LDF X003
ANDF X004
OUT M1

```

#### Program example:

Basic points to remember:

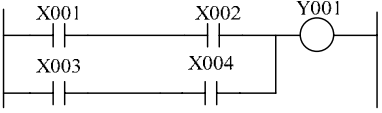
- Use the ORP and ORF instructions for the parallel connection of pulse contacts.
- Usage is the same as for OR and ORI; see earlier.
- ORP is active for one program scan after the associated device switches from OFF to ON.

- ORF is active for one program scan after the associated device switches from ON to OFF.

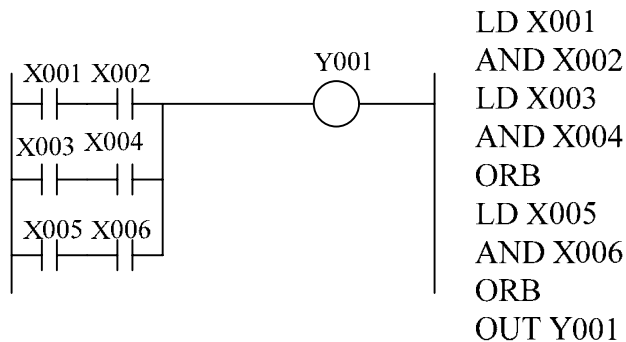
#### Single operation flags M2800 to M3071:

- When used with flags M2800 to M3071 only the first instruction will activate. For details see 1.8.1

### 1.11 Or Block

Mnemonic	Function	Format	Devices	Program steps
[ORB]	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.

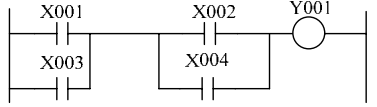
#### Batch processing limitations:

- 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).

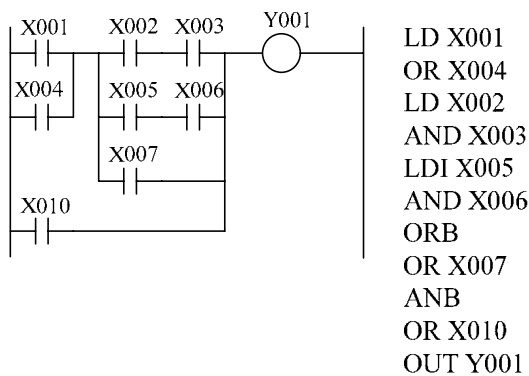
## 1.12 And Block

Mnemonic	Function	Format	Devices	Program steps
[ANB]	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 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.

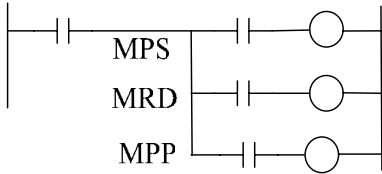
### Batch processing limitations:

- 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).

### 1.13 MPS, MRD and MPP

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

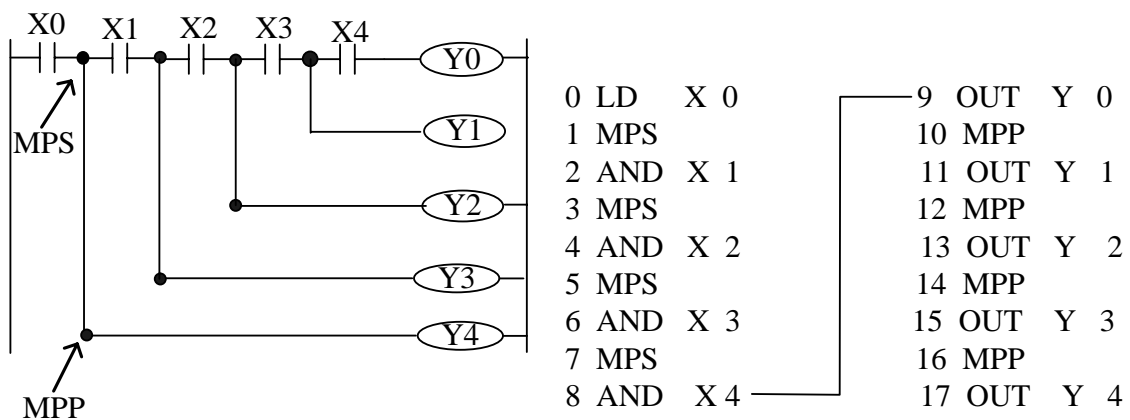
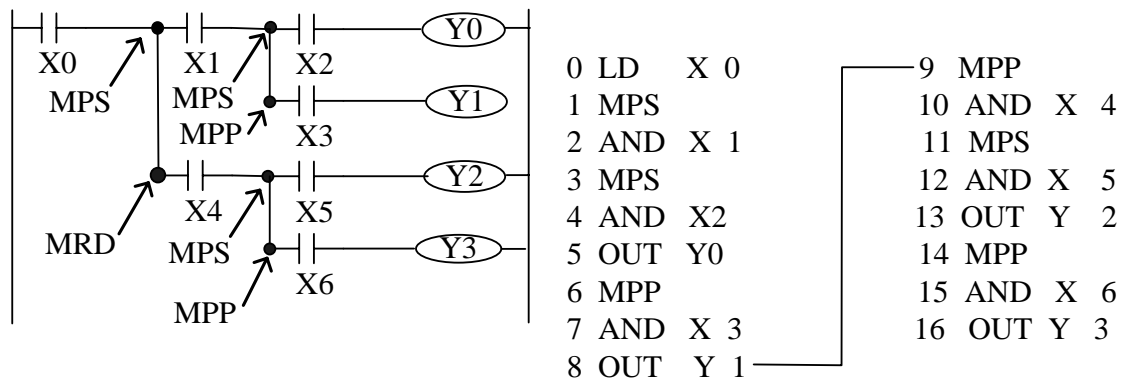
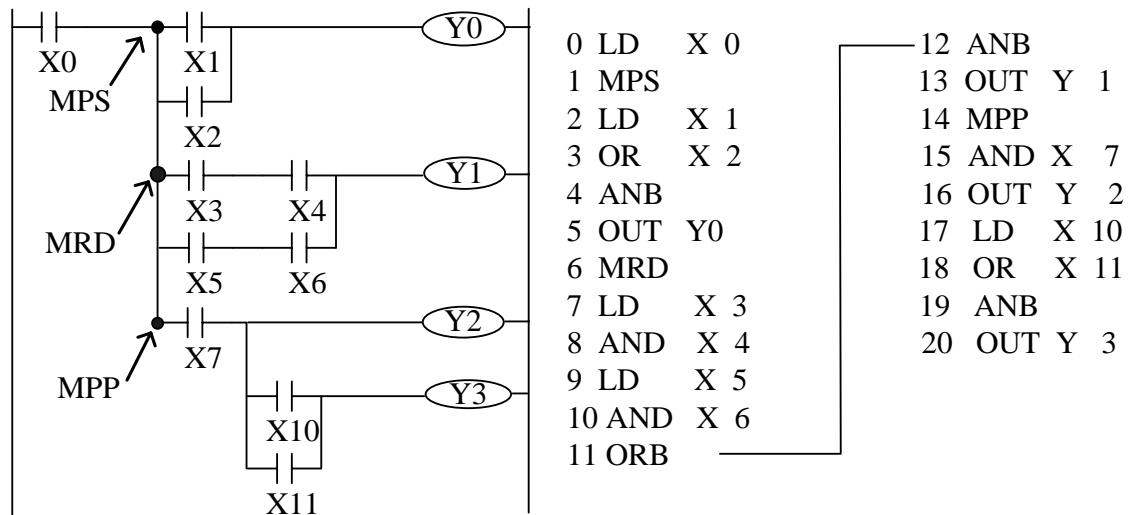
Basic points to remember:

- 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 8.

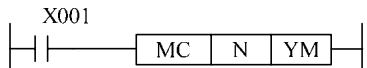
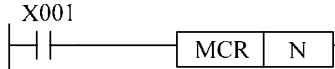
#### MPS, MRD and MPP usage:

- 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.

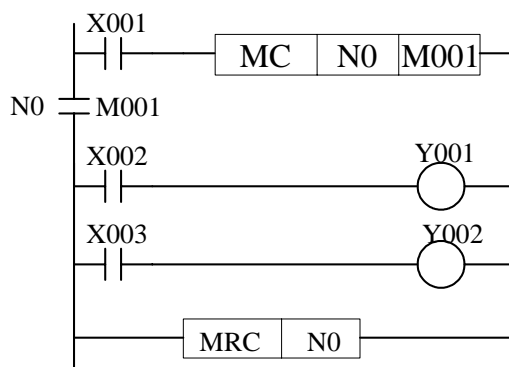
#### Program example



### 1.14 Master Control and Reset

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

#### Program example:



```
LD X001
MC N0
SP M001
LD X002
OUT Y001
LD X003
OUT Y002
MCR N0
```

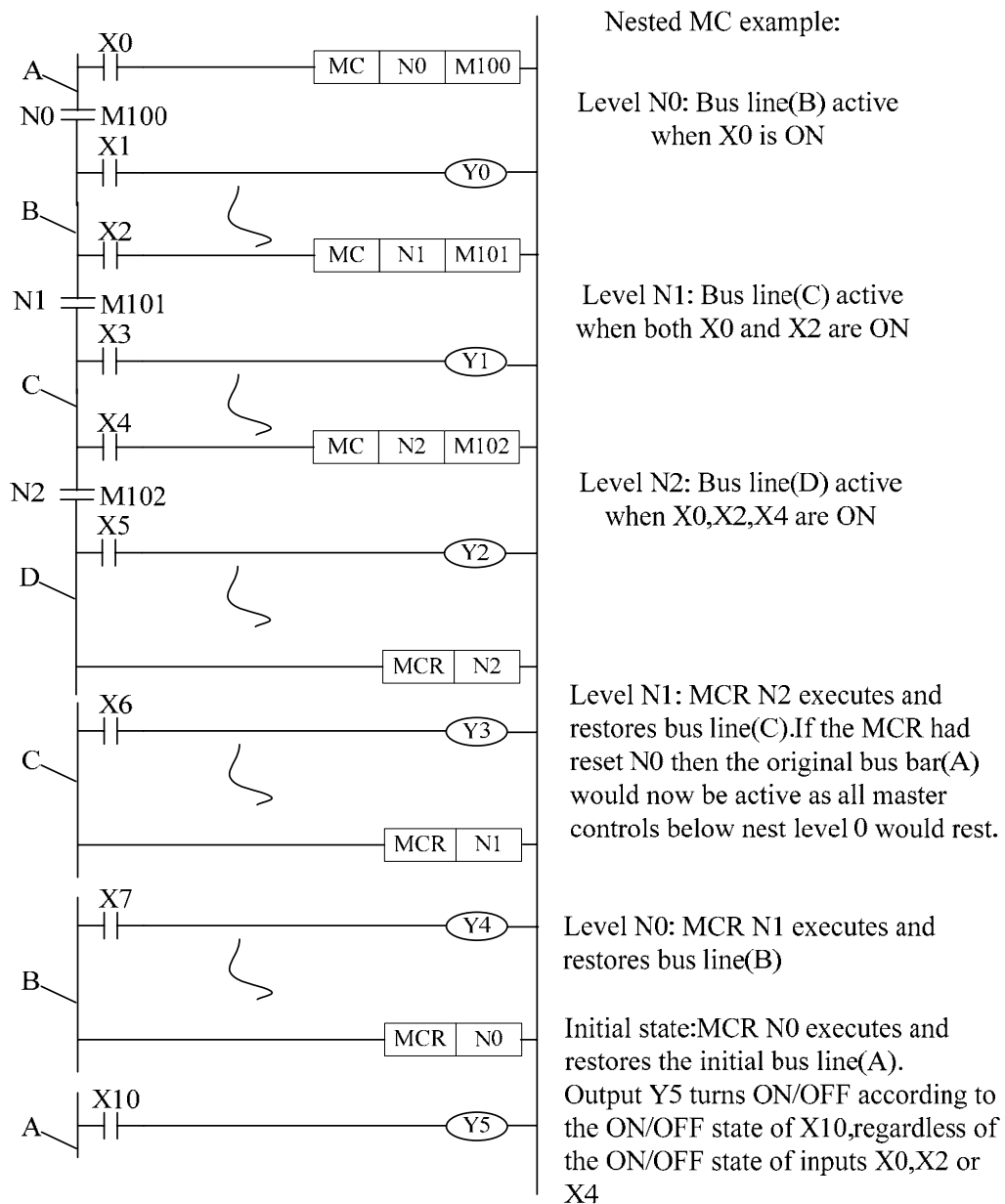
Basic points to remember:

- 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

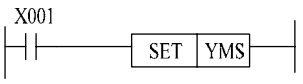
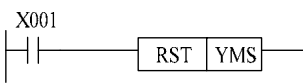
pointer N. Nest levels are from the range N0 to N7 (8 points). The top nest level is '0' and the deepest is '7'.

- 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 X1=ON, all instructions between the MC and the MCR instruction execute.
- When input X1=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 1.5.2). 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.

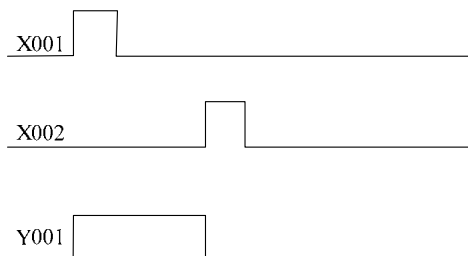
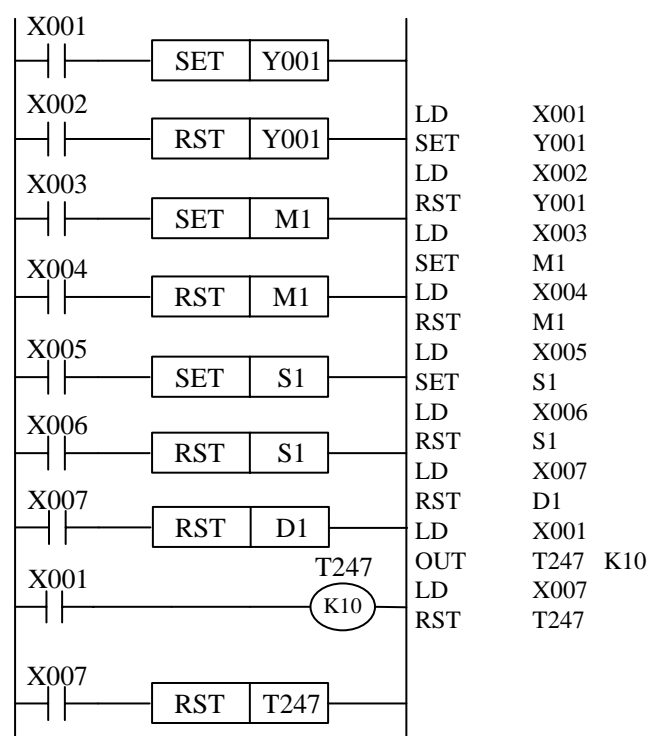




### 1.15 Set and Reset

Mnemonic	Function	Format	Devices	Program steps
[SET]	Sets a bit device permanently ON		Y,M,S	Y,M:1 S, special M : 2 D, V and Z:3
[RST]	Resets a bit device permanently OFF		Y,M,S,D,V,Z	

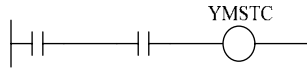
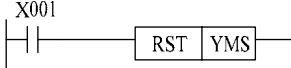
#### Program example:



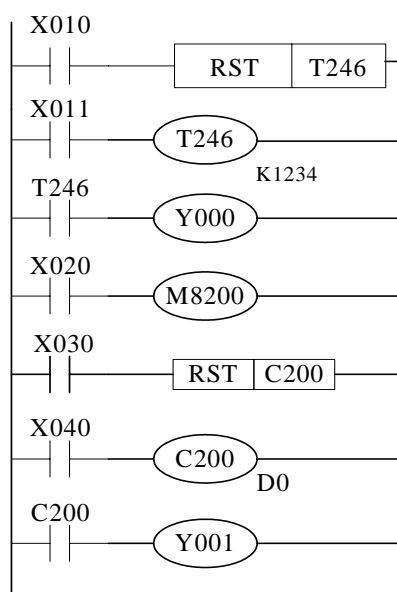
#### Basic points to remember:

- Turning ON X001 causes Y001 to turn ON. Y001 remains ON even after X001 turns OFF.
- Turning ON X002 causes Y001 to turn OFF. Y001 remains OFF even after X002 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.

## 1.16 Timer, Counter (Out & Reset)

Mnemonic	Function	Format	Devices	Program steps
[OUT]	Driving timer or counter coils		T,C	32 bit Counters:5 Others:3
[RST]	Reset timer and counter, coils contacts and current values		T,C	T,C:2

### Program example:



### 1.16.1 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 activated while X011 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 X010 will reset timer T246 in the manner described previously. Because the T246 contacts are reset, the output Y0 will be turned OFF.

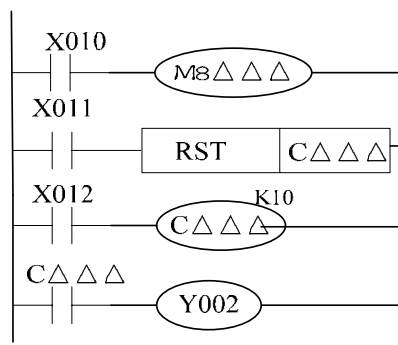
### Retentive timers:

- For more information on retentive timers please see **3.9.3**.

### 1.16.2 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 (see **1.16.1**) 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.

### 1.16.3 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 X010 is ON, counting down takes place.

When X010 is OFF, counting up takes place. In the example the output contacts of counter CΔΔΔ and its associated current count values are reset to "0".

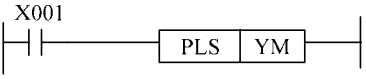
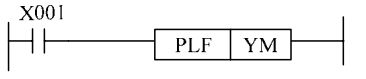
When X011 is turned ON. When X012 is turned ON the driven counter is enabled.

This means it will be able to start counting its assigned input signal (this will not be X012 - high speed counters are assigned special input signals, please see **3.11**).

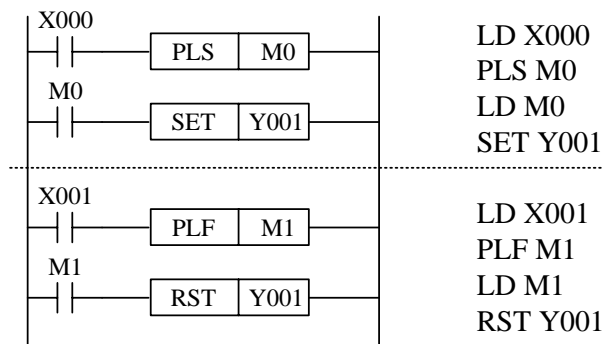
#### Availability of devices:

- 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 **3.11**. For PLC device ranges please see chapter 7.

### 1.17 Leading and Trailing Pulse

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

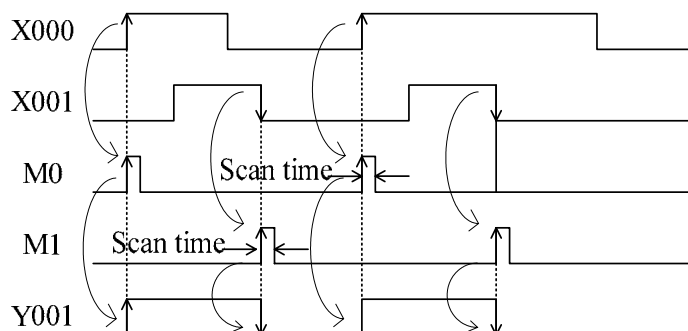
#### Program example:



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.

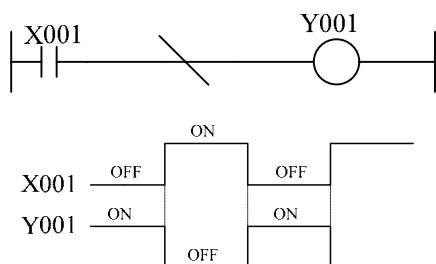


- When the PLC status is changed from RUN to STOP and back to RUN with the input signals still ON, PLS M0 is operated again. However, if an M coil which is battery backed (latched) was used instead of M0 it would not re-activate. For the battery backed device to be re-pulsed, its driving input (ex. X0) must be switched OFF during the RUN/STOP/RUN sequence before it will be pulsed once more.

### 1.18 Inverse

Mnemonic	Function	Format	Devices	Program steps
[INV]	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.

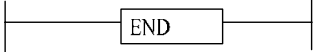
### 1.19 No Operation

Mnemonic	Function	Format	Devices	Program steps
[NOP]	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.

### 1.20 End

Mnemonic	Function	Format	Devices	Program steps
[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.

2	STL Programming.....	2
2.1	What is STL, SFC And IEC1131 Part 3? .....	2
2.2	How STL Operates.....	3
2.2.1	Each step is a program .....	3
2.3	How To Start And End An STL Program .....	4
2.3.1	Embedded STL programs.....	4
2.3.2	Activating new states .....	5
2.3.3	Terminating an STL Program.....	5
2.4	Moving Between STL Steps.....	6
2.4.1	Using SET to drive an STL coil .....	6
2.4.2	Using OUT to drive an STL coil .....	7
2.5	Rules and Techniques For STL programs .....	7
2.5.1	Basic Notes On The Behavior Of STL programs.....	7
2.5.2	Single Signal Step Control .....	9
2.6	Restrictions Of Some Instructions When Used With STL .....	9
2.7	Using STL To Select The Most Appropriate Program .....	10
2.8	Using STL To Activate Multiple Flows Simultaneously.....	11
2.9	General Rules For Successful STL Branching.....	12
2.10	Programming Examples .....	15
2.10.1	A Simple STL Flow.....	15
2.10.2	A Selective Branch/ First State Merge Example Program.....	16
2.11	Advanced STL Use .....	19



## 2 STL Programming

This chapter differs from the rest of the contents in this manual as it has been written with a training aspect in mind. STL/SFC programming, although having been available for many years, is still misunderstood and misrepresented. We at Mitsubishi would like to take this opportunity to try to correct this oversight as we see STL/SFC programming becoming as important as ladder style programming.

### 2.1 What is STL, SFC And IEC1131 Part 3?

The following explanation is very brief but is designed to quickly outline the differences and similarities between STL, SFC and IEC1131 part 3.

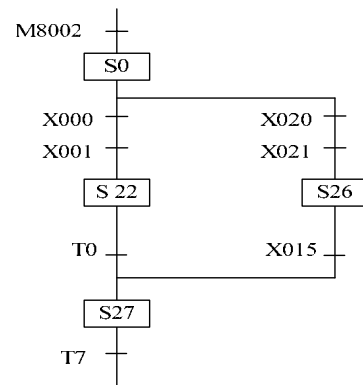
In recent years Sequential Function Chart (or SFC) style programming (including other similar styles such as Grafcet and Function plan) have become very popular through out Europe and have prompted the creation of IEC1131 part 3.

The IEC1131 SFC standard has been designed to become an interchangeable programming language. The idea being that a program written to IEC1131 SFC standards on one manufacturers PLC can be easily transferred (converted) for use on a second manufacturers PLC. STL programming is one of the basic programming instructions included in all FX PLC family members. The abbreviation STL actually means Step Ladder programming. STL programming is a very simple concept to understand yet can provide the user with one of the most powerful programming techniques possible. The key to STL lies in its ability to allow the programmer to create an operational program which 'flows' and works in almost exactly the same manner as SFC. This is not a coincidence as this programming technique has been developed deliberately to achieve an easy to program and monitor system. One of the key differences to Mitsubishi's STL programming system is that it can be entered into a PLC in 3 formats. These are:

- I) Instruction - a word/mnemonic entry system
- II) Ladder - a graphical program construction method using a relay logic symbols
- III) SFC - a flow chart style of STL program entry (similar to SFC)

#### General note:

- IEC1131-3: 03.1993 Programmable controllers; part 3: programming languages. The above standard is technically identical to the 'Euro-Norm' EN61131-3: 07.1993



## 2.2 How STL Operates

As previously mentioned, STL is a system which allows the user to write a program which functions in much the same way as a flow chart, this can be seen in the diagram opposite.

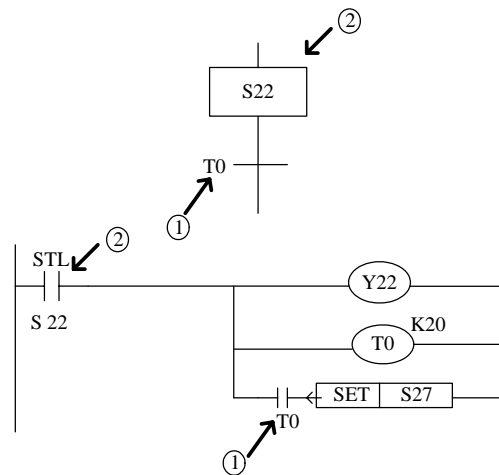
STL derives its strength by organizing a larger program into smaller more manageable parts. Each of these parts can be referred to as either a state or a step. To help identify the states, each is given a unique identification number. These numbers are taken from the state relay devices (see page 4-6 for more details).

### 2.2.1 Each step is a program

Each state is completely isolated from all other states within the whole program. A good way to envisage this, is that each state is a separate program and the user puts each of those programs together in the order that they require to perform their task. Immediately this means that states can be reused many times and in different orders. This saves on programming time AND cuts down on the number of programming errors encountered.

#### A Look inside an STL

On initial inspection the STL program looks as if it is a rather basic flow diagram. But to find out what is really happening the STL state needs to be put 'under a microscope' so to peak. When a single state is examined in more detail, the sub-program can be viewed. With the exception of the STL instruction, it will be immediately seen that the STL sub-program looks just like ordinary programming.



<sup>1</sup> The STL instruction is shown as a 'fat' normally open contact. All programming after an STL instruction is only active when the associated state coil is active.

<sup>2</sup> The transition condition is also written using standard programming. This idea re-enforces the concept that STL is really a method of sequencing a series of events or as mentioned earlier 'of joining lots of smaller programs together'.

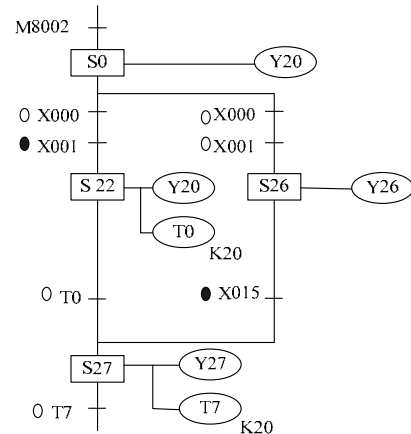
### Combined SFC Ladder representation

Sometimes STL programs will be written in hard copy as a combination of both flow diagram and internal sub-program. (example shown below).

Identification of contact states

- Please note the following convention is used:
  - Normally Open contact
  - Normally Closed contact

Common alternatives are 'a' and 'b' identifiers for normally Open, Normally Closed states or often a line drawn over the top of the Normally Closed contact name is used, e.g.X000.

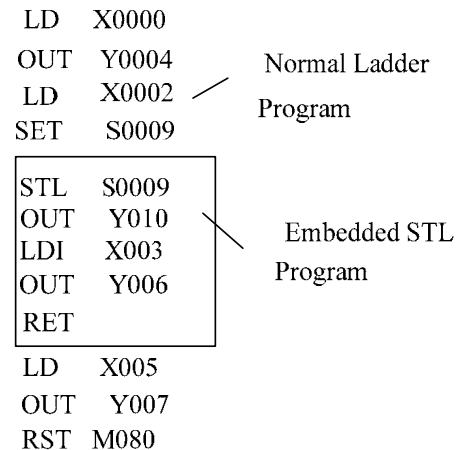


## 2.3 How To Start And End An STL Program

Before any complex programming can be undertaken the basics of how to start and more importantly how to finish an STL program need to be examined.

### 2.3.1 Embedded STL programs

An STL style program does not have to entirely replace a standard ladder logic program. In fact it might be very difficult to do so. Instead small or even large section of STL program can be entered at any point in a program. Once the STL task has been completed the program must go back to processing standard program instructions until the next STL program block. Therefore, identifying the start and end of an STL program is very important.



### 2.3.2 Activating new states

Once an STL step has been selected, how is it used and how is the program 'driven'? This is not so difficult; if it is considered that for an STL step to be active its associated state coil must be ON. Hence, to start an STL sequence all that has to be done is to drive the relevant state ON.

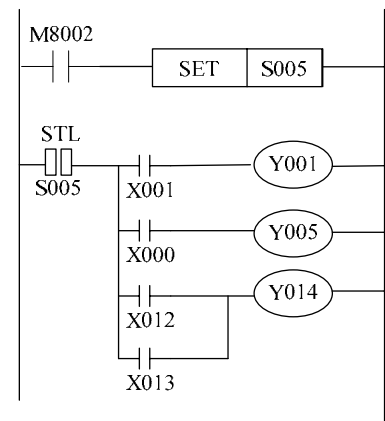
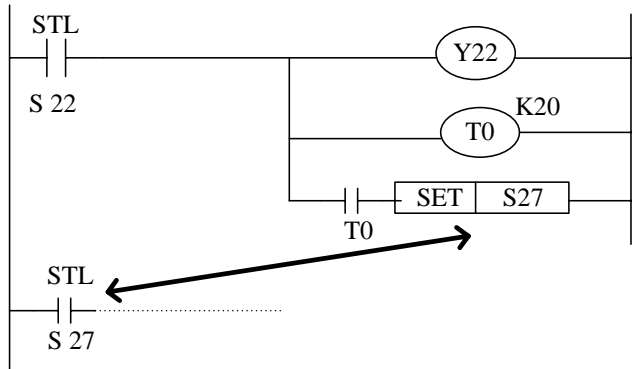
There are many different methods to drive a state, for example the initial state coils could be pulsed, SET or just included in an OUT instruction. However, within Mitsubishi's STL programming language an STL coil which is SET has a different meaning than one that is included in an OUT instruction.

**Note:** For normal STL operation it is recommended that the states are selected using the SET instruction. To activate an STL step its state coil is SET ON.

#### Initial Steps

For an STL program which is to be activated on the initial power up of the PLC, a trigger similar to that shown opposite could be used, i.e. using M8002 to drive the setting of the initial state.

The STL step started in this manner is often referred to as the initial step. Similarly, the step activated first for any STL sequence is also called the initial step.



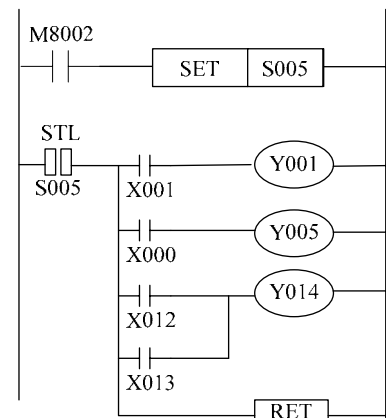
### 2.3.3 Terminating an STL Program

Once an STL program has been started the programmable controllers CPU will process all following instructions as being part of that STL program. This means that when a second program scan is started the normal instructions at the beginning of the program are considered to be within the STL program. This is obviously incorrect and the CPU will proceed to identify a programming error and disable the programmable controllers operation.

This scenario may seem a little strange but it does make sense when it is considered that the STL program must return control to the ladder program after STL operation is complete. This means the last step in an STL program needs to be identified in some way.

#### Returning to Standard Ladder

This is achieved by placing a RET or RETurn instruction as the last instruction in the last STL step of an STL program block. This instruction then returns



programming control to the ladder sequence.

**Note:** The RET instruction can be used to separate STL programs into sections, with standard ladder between each STL program. For display of STL in SFC style format the RET instruction is used to indicate the end of a complete STL program.

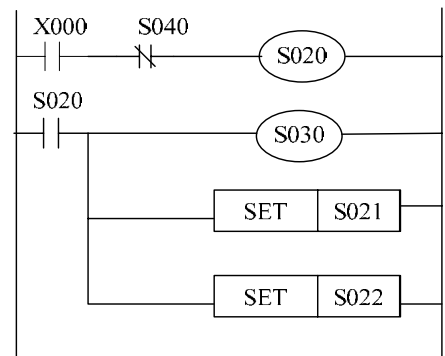
## 2.4 Moving Between STL Steps

To activate an STL step the user must first drive the state coil. Setting the coil has already been identified as a way to start an STL program, i.e. drive an initial state. It was also noted that using an OUT statement to driving a state coil has a different meaning to the SET instruction. These differences will now be explained:

### 2.4.1 Using SET to drive an STL coil

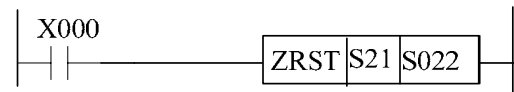
SET is used to drive an STL state coil to make the step active. Once the current STL step activates a second following step, the source STL coil is reset. Hence, although SET is used to activate a state the resetting is automatic.

However, if an STL state is driven by a series of standard ladder logic instructions, i.e. not a preceding STL state, then standard programming rules apply. In the example shown opposite S20 is not reset even after S30 or S21 have been driven. In addition, if S20 is turned OFF, S30 will also stop operating. This is because S20 has not been used as an STL state. The first instruction involving the status of S20 is a standard Load instruction and NOT an STL instruction.



#### **Note:**

If a user wishes to forcibly reset an STL step, using the RST or ZRST (FNC40) instructions would perform this task



- SET is used to drive an immediately following STL step which typically will have a larger STL state number than the current step.
- SET is used to drive STL states which occur within the enclosed STL program flow, i.e. SET is not used to activate a state which appears in an unconnected, second STL flow diagram.

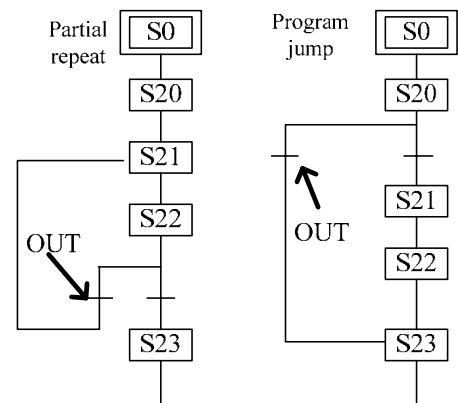
### 2.4.2 Using OUT to drive an STL coil

This has the same operational features as using SET. However, there is one major function which SET is not used. This is to make what is termed 'distant jumps'.

#### OUT is used for loops and jumps

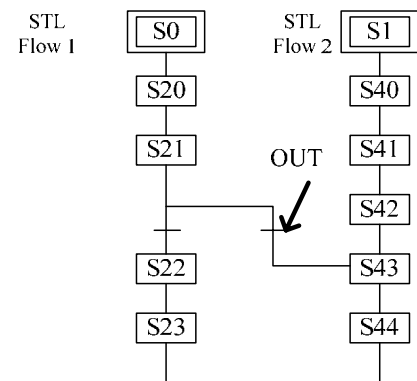
If a user wishes to 'jump' back up a program, i.e. go back to a state which has already been processed, the OUT instruction would be used with the appropriate STL state number.

Alternatively the user may wish to make a large 'jump' forwards skipping a whole section of STL programmed states.



#### Out is used for distant jumps

If a step in one STL program flow was required to trigger a step in a second, separate STL program flow the OUT instruction would be used.



**Note:** Although it is possible to use SET for jumps and loops use of OUT is needed for display of STL in SFC like structured format.

## 2.5 Rules and Techniques For STL programs

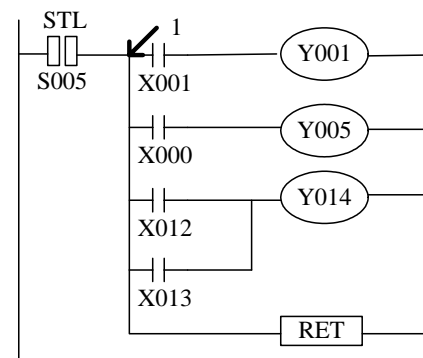
It can be seen that there are a lot of advantages to using STL style programming but there are a few points a user must be aware of when writing the STL sub-programs. These are highlighted in this section.

### 2.5.1 Basic Notes On The Behavior Of STL programs

- When an STL state becomes active its program is processed until the next step is triggered. The contents of the program can contain all of the programming items and features of a standard ladder program, i.e. Load, AND OR, OUT, ReSeT etc., as well as applied instructions.

- When writing the sub-program of an STL state, the first vertical 'bus bar' after the STL instruction can be considered in a similar manner as the left hand bus bar of a standard ladder program.

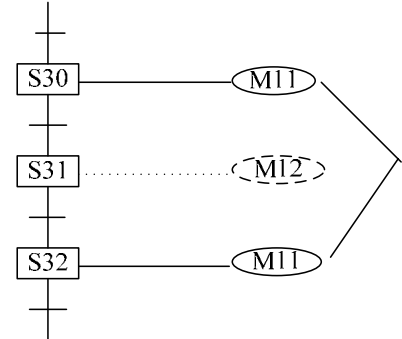
Each STL step makes its own bus bar. This means that a user, cannot use an MPS instruction directly after the STL instruction (see 1 ), i.e. There needs to be at least a



single contact before the MPS instruction.

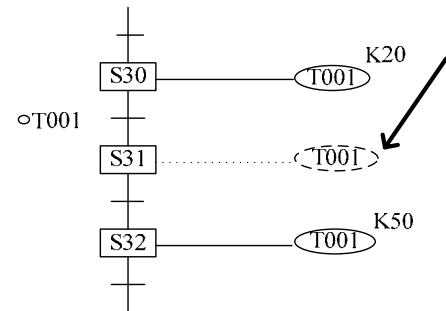
**Note:** Using out coils and even applied instructions immediately after an STL instruction is permitted.

- In normal programming using dual coils is not an acceptable technique. However repetition of a coil in separate STL program blocks is allowed. This is because the user can take advantage of the STL's unique feature of isolating all STL steps except the active STL steps. This means in practice that there will be no conflict between dual coils. The example opposite shows M11 used twice in a single STL flow.



**Caution:** The same coil should NOT be programmed in Steps that will be active at the same time as this will result in the same problem as other dual coils.

- When an STL step transfers control to the next STL step there is a period (one scan) while both steps are active. This can cause problems with dual coils; particularly timers. If timers are dual coiled care must be taken to ensure that the timer operation is completed during the active STL step.



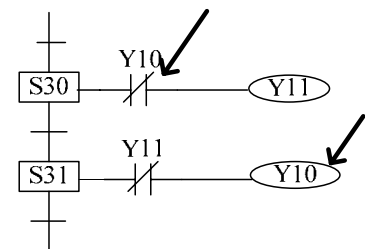
If the same timer is used in consecutive steps then it is possible that the timer coil is never deactivated and the contacts of the timer will not be reset leading to incorrect timer operation.

The example opposite identifies an unacceptable use of timer T001. When control passes from S30 to S31 T001 is not reset because its coil is still ON in the new step.

**Note:** As a step towards ensuring the correct operation of the dual timers they should not be used in consecutive STL steps.

Following this simple rule will ensure each timer will be reset correctly before its next operation.

- As already mentioned, during the transfer between steps, the current and the selected steps will be simultaneously active for one program scan. This could be thought of as a hand over or handshaking period. This means that if a user has two outputs contained in consecutive steps which must NOT be active



simultaneously they must be interlocked. A good example of this would be the drive signals to select a motors rotation direction. In the example Y11 and Y10 are shown interlocked with each other.

### 2.5.2 Single Signal Step Control

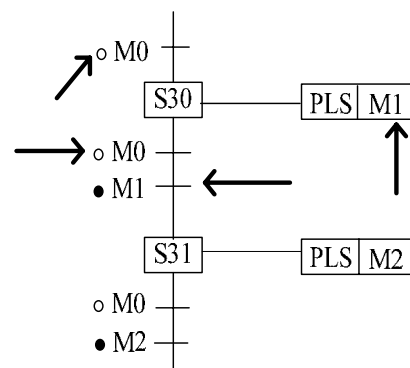
Transferring between active STL steps can be controlled by a single signal.

#### ● Using locking devices

In this example it is necessary to program separate locking devices, and the controlling signal must only pulse ON. This is to prevent the STL programs from running through.

The example shown below identifies the general program required for this method.

- S30 is activated when M0 is first pulsed ON.
- The operation of M1 prevents the sequence from continuing because although M0 is ON, the transfer requirements need M0 to be ON and M1 to be OFF.
- After one scan the pulsed M0 and the 'lock' device M1 are reset.
- On the next pulse of M0 the STL step will transfer program control from S31 to the next step in a similar manner. This time using M2 as the 'lock' device because dual coils in successive steps is not allowed.
- The reason for the use of the 'lock' devices M1 and M2 is because of the handshaking period when both states involved in the transfer of program control are ON for 1 program scan. Without the 'locks' it would be possible to immediately skip through all of the STL states in one go!

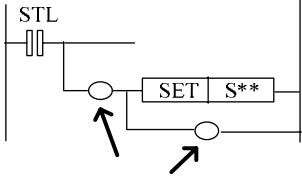
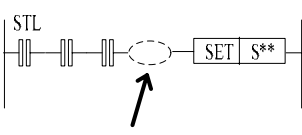


## 2.6 Restrictions Of Some Instructions When Used With STL

Although STL can operate with most basic and applied instructions there are a few exceptions. As a general rule STL and MC-MCR programming formats should not be combined. Other instruction restrictions are listed in the table below.

Operational State		Basic Instructions		
		LD/LDI/LDP/LDF, AND /ANI/ANDP/ANDF/OR/ORI/ORF, INV, OUT/SET/RST, PLS/PLF	ANB/ORB /MPS/MRD/MPP	MC/MCR
Initial and general states				X



Branching and merging states	Output processing				X
	Transfer processing			X	X

### Restrictions on using applied instructions

- Most applied instructions can be used within STL programs. Attention must be paid to the way STL isolates each non-active step. It is recommended that when applied instructions are used their operation is completed before the active STL step transfers to the next step.

Other restrictions are as follows:

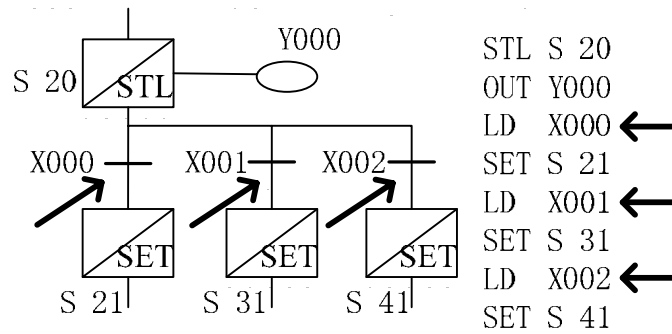
- FOR - NEXT structures can not contain STL program blocks.
- Subroutines and interrupts can not contain STL program blocks.
- STL program blocks can not be written after an FEND instruction.
- FOR - NEXT instructions are allowed within an STL program with a nesting of up to 4 levels.

### Using 'jump' operations with STL

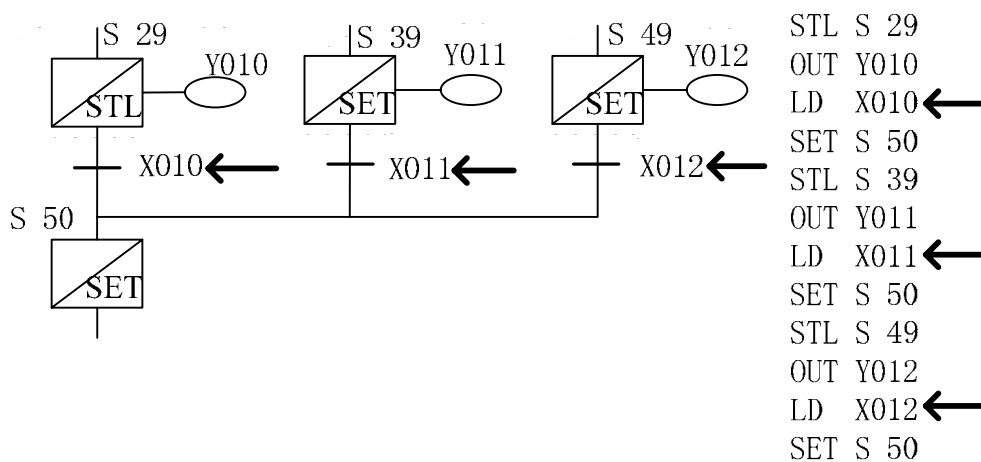
- Although it is possible to use the program jump operations (CJ instruction) within STL program flows, this causes additional and often unnecessary program flow complications. To ensure easy maintenance and quick error finding it is recommended that users do not write jump instructions into their STL programs.

## 2.7 Using STL To Select The Most Appropriate Program

So far STL has been considered as a simple flow charting programming language. One of STL's exceptional features is the ability to create programs which can have several operating modes. For example certain machines require a selection of 'manual' and 'automatic' modes, other machines may need the ability to select the operation or manufacturing processes required to produce products 'A', 'B', 'C', or 'D'. STL achieves this by allowing multiple program branches to originate from one STL state. Each branch is then programmed as an individual operating mode, and because each operating mode should act individually, i.e. there should be no other modes active; the selection of the program branch must be mutually exclusive. This type of program construction is called "Selective Branch Programming". An example instruction program can be seen below, (this is the sub-program for STL state S20 only) notice how each branch is SET by a different contact.

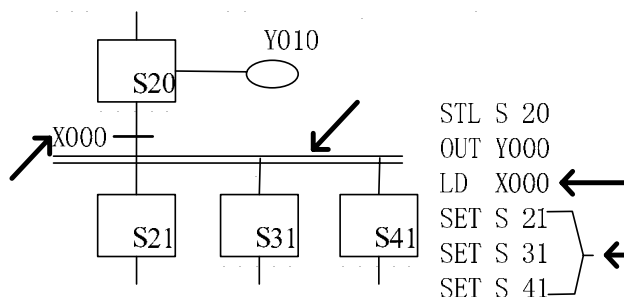


A programming construction to split the program flow between different branches is very useful but it would be more useful if it could be used with a method to rejoin a set of individual branches.



This type of STL program construction is called a “First State Merge” simply because the first state (in the example S29, S39 or S49) to complete its operation will cause the merging state (S50) to be activated. It should be noticed how each of the final STL states on the different program branches call the same “joining” STL state.

## 2.8 Using STL To Activate Multiple Flows Simultaneously



In the previous branching technique, it was seen how a single flow could be selected from a group. The following methods describe how a group of individual flows can be activated simultaneously. Applications could include vending machines which have to perform several tasks at once, e.g.

boiling water, adding different taste ingredients (coffee, tea, milk, sugar) etc. In the example below when state S20 is active and X0 is then switched ON, states S21, S31 and S41 are ALL SET ON as the next states. Hence, three separate, individual, branch flows

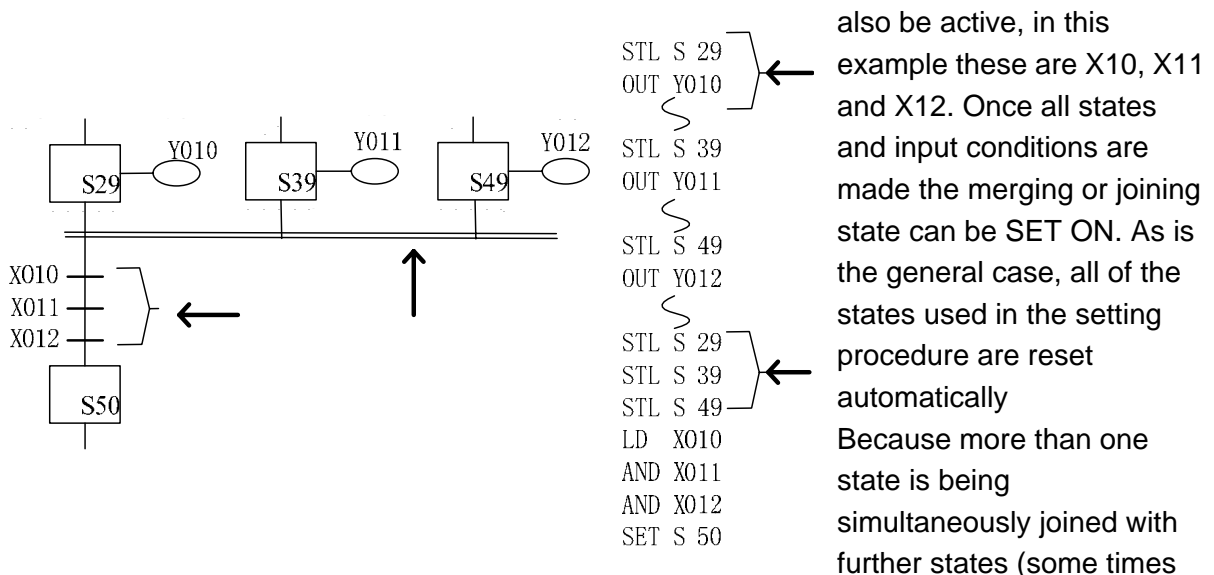
are 'set in motion' from a single branch point. This programming technique is often called a 'Parallel Branch'. To aid a quick visual distinction, parallel branches are marked with horizontal, parallel lines.

When a group of branch flows are activated, the user will often either;

- a) 'Race' each flow against its counter parts. The flow which completes fastest would then activate a joining function ("First State Merge" described in the previous section)  
OR
- b) The STL flow will not continue until ALL branch flows have completed there tasks.  
This is called a 'Multiple State Merge'.

An explanation of Multiple State Merge now follows below.

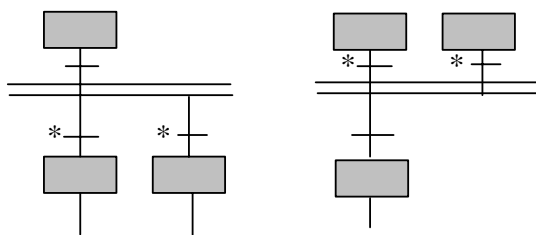
In the example below, states S29, S39 and S49 must all be active. If the instruction list is viewed it can be seen that each of the states has its own operating/processing instructions but that also additional STL instructions have been linked together (in a similar concept as the basic AND instruction). Before state S50 can be activated the trigger conditions must



described as a parallel merge), a set of horizontal parallel lines are used to aid a quick visual recognition.

## 2.9 General Rules For Successful STL Branching

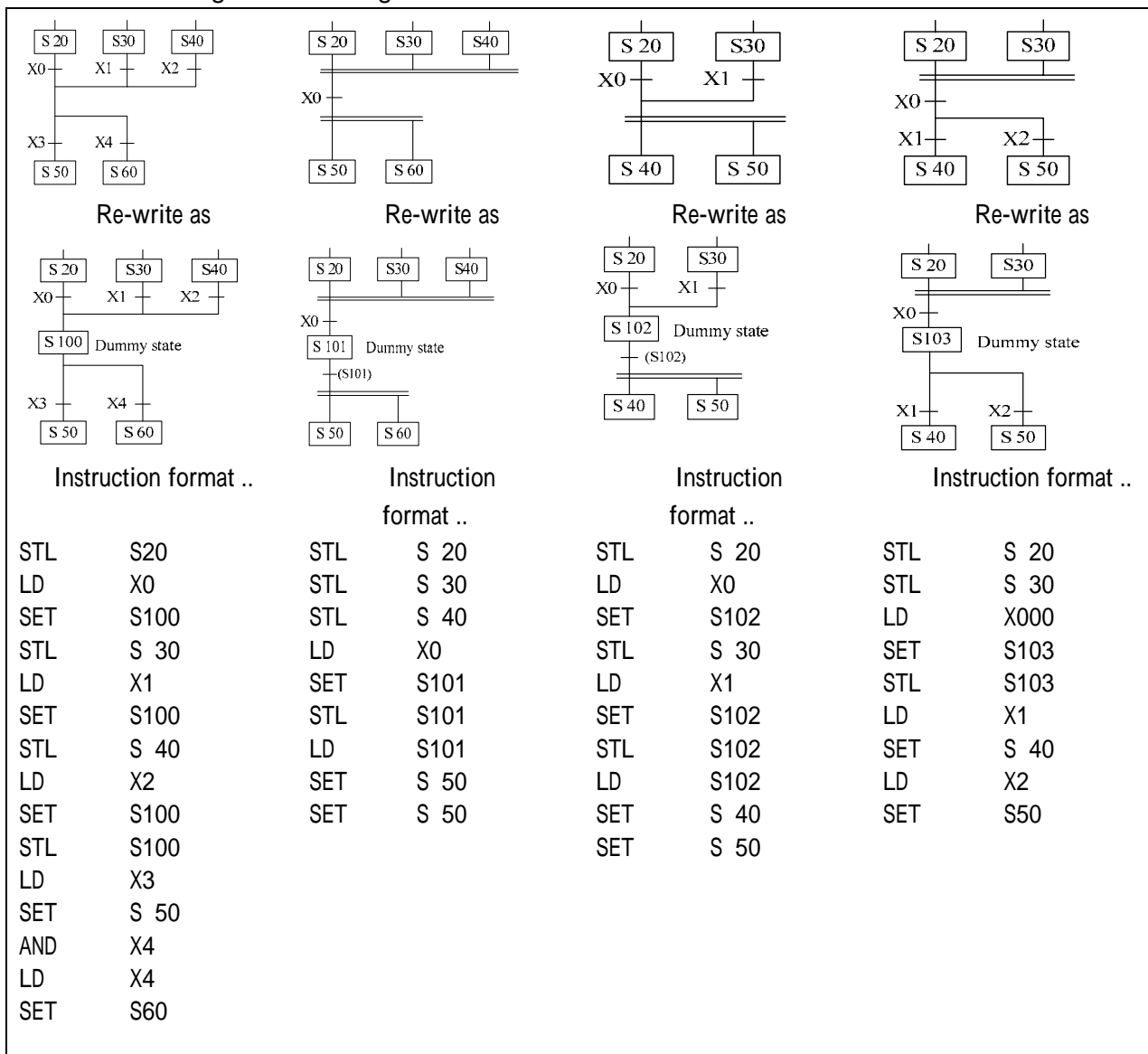
For each branch point 10 further branches may be programmed. There are no limits to the number of states contained in a single STL flow. Hence, the possibility exists for a single initial state to branch to 10 branch flows which in turn could each branch to a further 8 branch flows etc. If the programmable controllers program is read/written using instruction



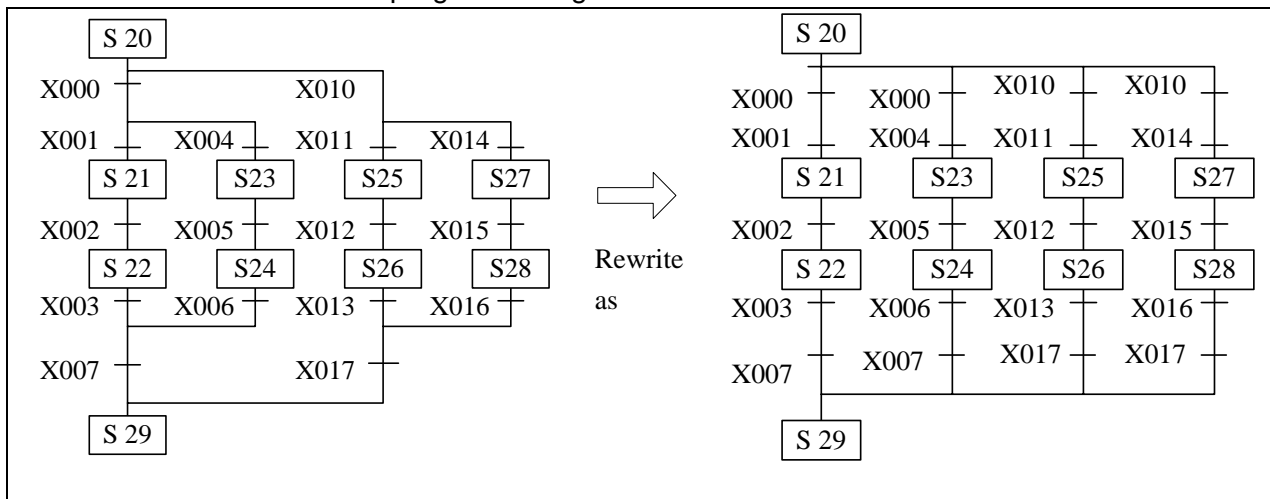
or ladder formats the above rules are acceptable. However, users of the programming package who are utilizing the STL programming feature are constrained by further restrictions to enable automatic STL program conversions (please see page 3-15 for more details).

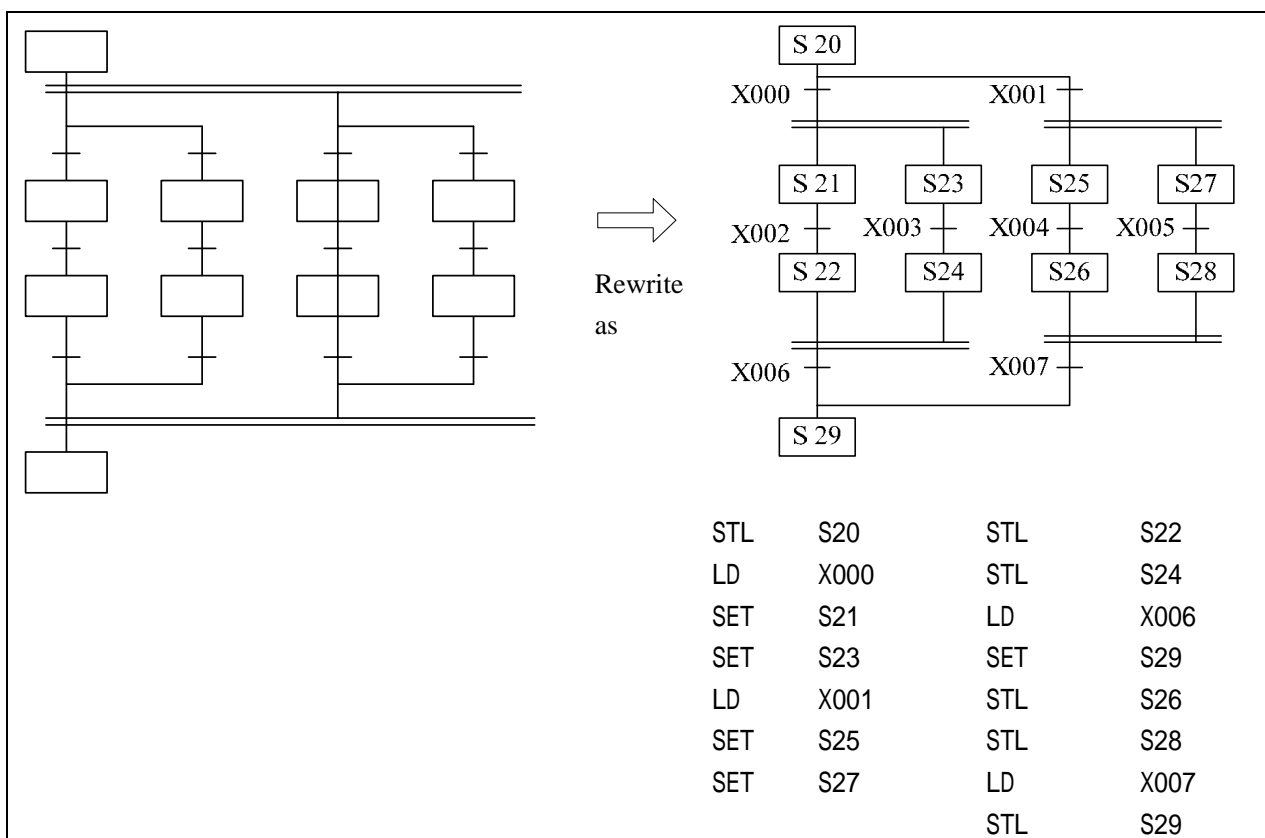
When using branches, different types of branching /merging cannot be mixed at the same branch point. The item marked with a 'S' are transfer condition which are not permitted.

The following branch configurations/modifications are recommended:



Further recommended program change:



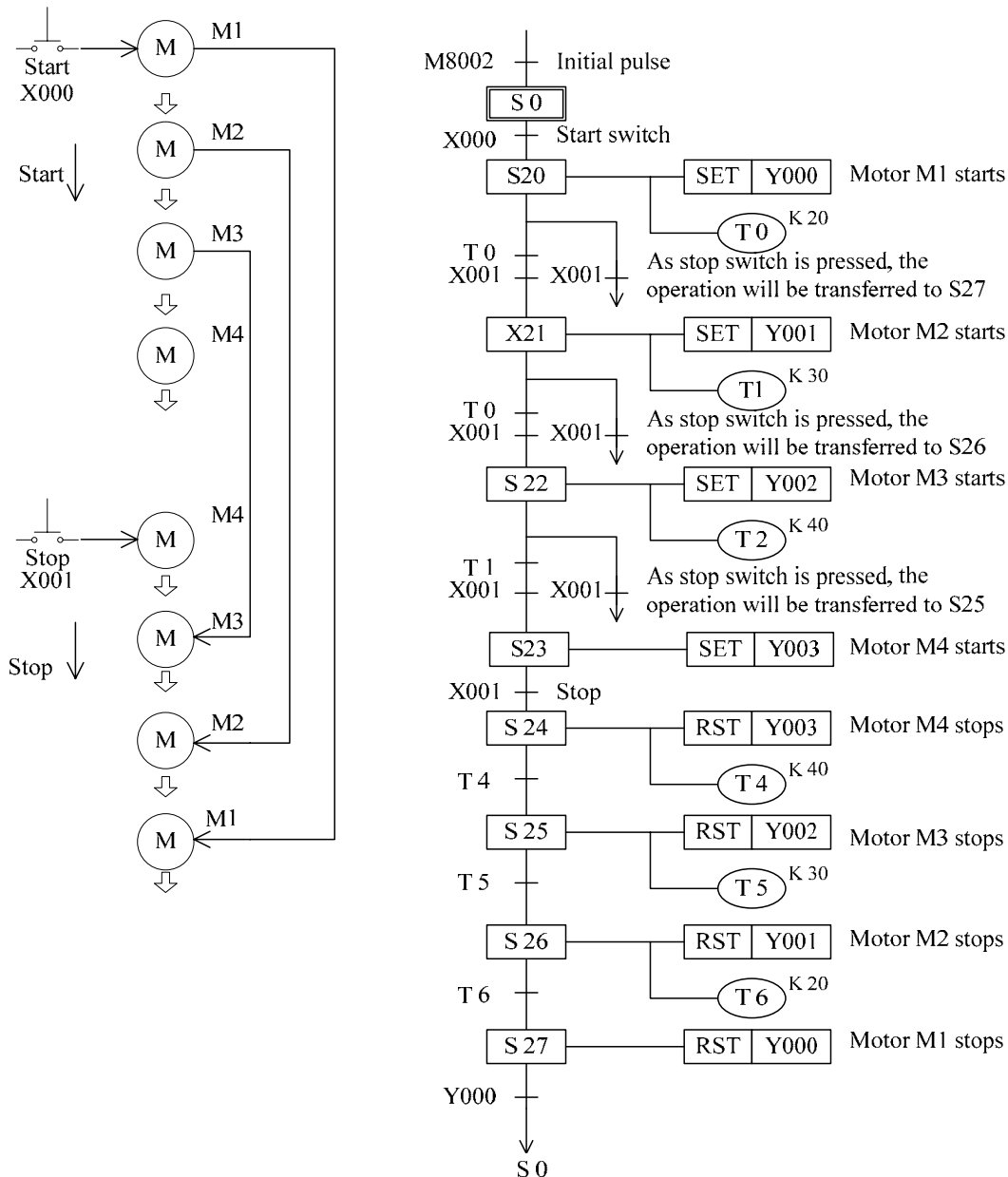


## 2.10 Programming Examples

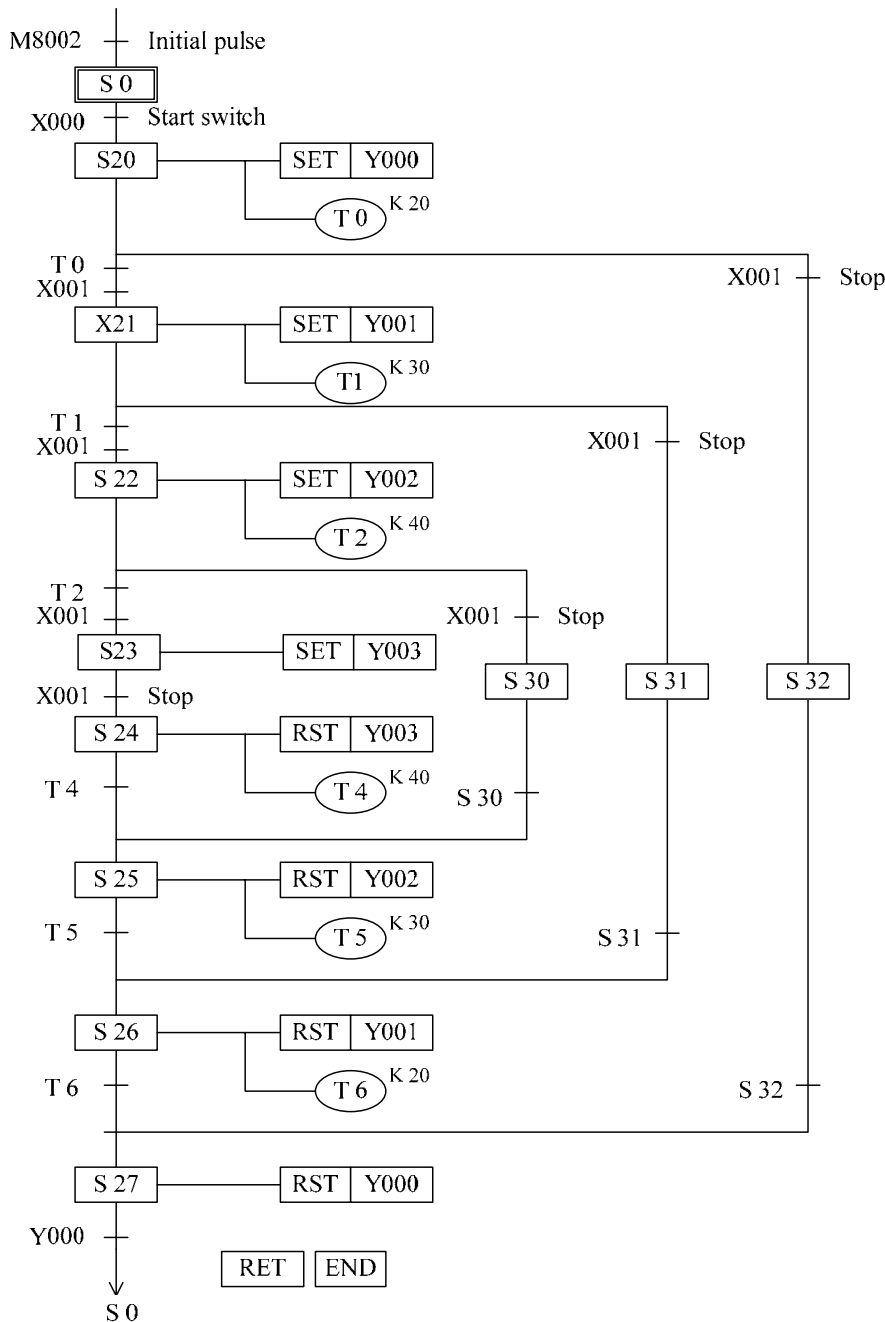
### 2.10.1 A Simple STL Flow

#### Example for sequential start and stop

The motors engage in operation from M1~M4, then stop adversely from M4~M1. SFC program transfer the states based on single flow.



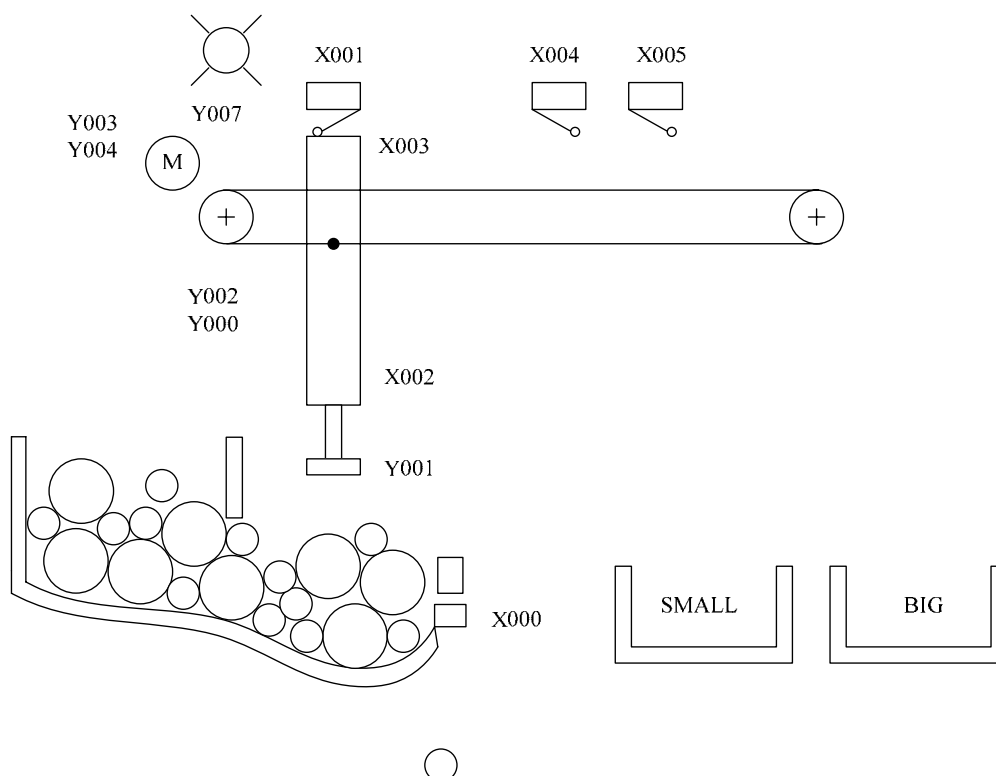
Such flow can be programmed by selectively activating multiple flows and inactivating them. The direction of the flow should be from up to bottom. All the flow can not be crossed except branching line or meeting line.



For instance, when X001 in S20 is ON and S32 is activated, the program will directly jump to S27. It is necessary to add a idle state for the branch should include one or more state.

### 2.10.2 A Selective Branch/ First State Merge Example Program

The following example depicts an automatic sorting robot. The robot sorts two sizes of ball bearings from a mixed 'source pool' into individual storage buckets containing only one type of ball bearing.



The sequence of physical events (from initial power On) are:

- 1) The pickup arm is moved to its zero-point when the start button (X12) is pressed. When the pickup arm reaches the zero-point the zero-point lamp (Y7) is lit.
- 2) The pickup arm is lowered (Y0) until a ball is collected (Y1). If the lower limit switch (X2) is made a small ball bearing has been collected; consequently no lower limit switch signal means a large ball bearing has been collected. Note, a proximity switch (X0) within the 'source pool' identifies the availability of ball bearings.
- 3) Depending on the collected ball, the pickup arm retracts (output Y2 is operated until X3 is received) and moves to the right (Y3) where it will stop at the limit switch (X4 or X5) indicating the container required for storage.
- 4) The program continues by lowering the pickup arm (Y0) until the lower limit switch (X2) is reached.
- 5) The collected ball being is released (Y1 is reset).
- 6) The pickup arm is retracted (Y2) once more.
- 7) The pickup arm is traversed back (Y4) to the zero-point (X1).

### Points to note

- The Selective Branch is used to choose the delivery program for either small ball bearings or large ball bearings. Once the destination has been reached (i.e. step S24 or S27 has been executed) the two independent program flows are rejoined at step S30.
- The example program shown works on a single cycle, i.e. every time a ball is to be retrieved the start button (X12) must be pressed to initiate the cycle. Full STL flow

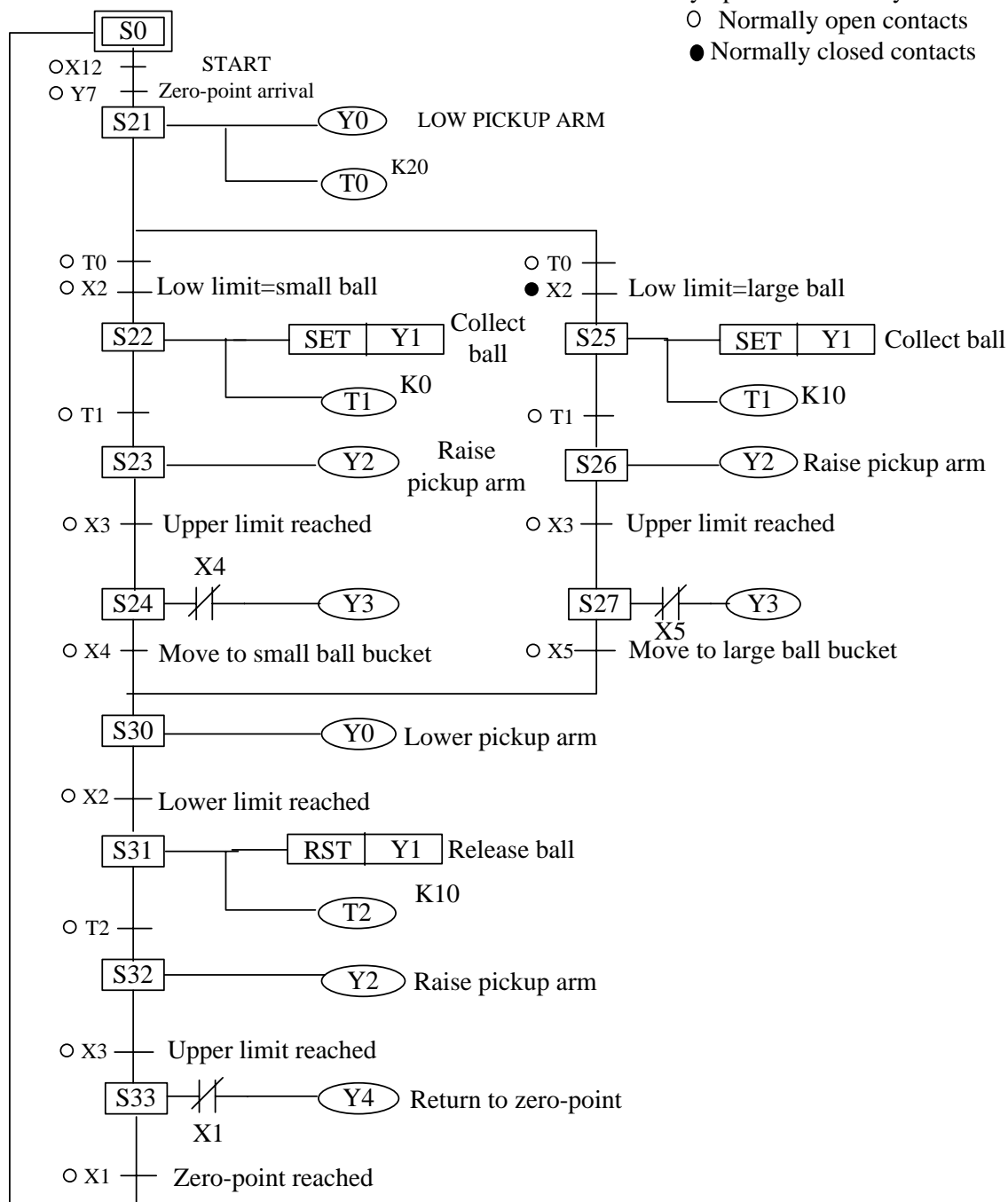


diagram/program.

This example uses the dot notation to identify normally open and normally closed contacts.

○ Normally open contacts

● Normally closed contacts



## 2.11 Advanced STL Use

STL programming can be enhanced by using the Initial State Applied Instruction. This instruction has a mnemonic abbreviation of IST and a special function number of 60. When the IST instruction is used an automatic assignment of state relays, special auxiliary relays (M coils) is made. The IST instruction provides the user with a pre-formatted way of creating a multi-mode program. The modes available are:

- a) Automatic:
  - Single step
  - Single cycle
  - Continuous
- b) Manual:
  - Operator controlled
  - Zero return

3	Devices in Detail .....	3
3.1	Inputs .....	3
3.2	Outputs.....	3
3.3	Auxiliary Relays .....	4
3.3.1	General Stable State Auxiliary Relays .....	4
3.3.2	Battery Backed/ Latched Auxiliary Relays.....	4
3.3.3	Special Diagnostic Auxiliary Relays .....	5
3.3.4	Special Single Operation Pulse Relays .....	5
3.4	State Relays .....	6
3.4.1	General Stable State - State Relays.....	6
3.4.2	Battery Backed/ Latched State Relays.....	6
3.4.3	STL Step Relays .....	7
3.4.4	Annunciator Flags.....	7
3.5	Pointers .....	9
3.6	Interrupt Pointers .....	10
3.6.1	Input Interrupts .....	10
3.6.2	Timer Interrupts .....	11
3.6.3	Disabling Individual Interrupts.....	11
3.6.4	Counter Interrupts .....	11
3.7	Constant K.....	12
3.8	Constant H.....	12
3.9	Timers .....	13
3.9.1	General timer operation .....	13
3.9.2	Selectable Timers.....	14
3.9.3	Retentive Timers.....	14
3.9.4	Timers Used in Interrupt and 'CALL' Subroutines.....	14
3.9.5	Timer Accuracy.....	14
3.10	Counters.....	16
3.10.1	General/ Latched 16bit UP Counters .....	16
3.10.2	General/ Latched 32bit Bi-directional Counters .....	17
3.11	High Speed Counters .....	18
3.11.1	Basic High Speed Counter Operation .....	18
3.12	Data Registers.....	19
3.12.1	General Use Registers.....	19
3.12.2	Special Diagnostic Registers.....	20
3.12.3	Externally Adjusted Registers.....	21
3.13	Index Registers .....	21
3.13.1	Modifying a Constant .....	22
3.13.2	Misuse of the Modifiers.....	23
3.13.3	Using Multiple Index Registers .....	23
3.14	Bits, Words, BCD and Hexadecimal.....	23
3.14.1	Bit Devices, Individual and Grouped.....	24
3.14.2	Word Devices.....	25
3.14.3	Interpreting Word Data .....	25

3.14.4 Two’s Compliment..... 27

3.15 Floating Point And Scientific Notation..... 28

3.15.1 Scientific Notation ..... 29

3.15.2 Floating Point Format ..... 30

### 3 Devices in Detail

#### 3.1 Inputs

**Device Mnemonic:** X

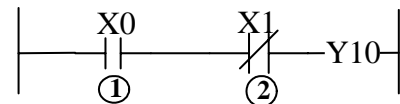
**Purpose:** Representation of physical inputs to the programmable controller (PLC)

**Alias:** I/P

Inp

(X) Input

Input contact



**Available forms:** NO ( 1 ) and NC ( 2 ) contacts only

**Devices numbered in:** Octal, i.e. X0 to X7, X10 to X17

**Further uses:** None

**Available devices:**

- Please see the information point on **3.2, Outputs**. Alternatively refer to the relevant tables for the selected PLC in chapter 7.

#### 3.2 Outputs

**Device Mnemonic:** Y

**Purpose:** Representation of physical outputs from the programmable controller

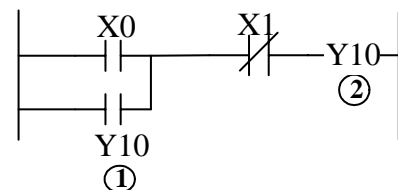
**Alias:** O/P

Otp

Out (Y)

Output (Y)

Output (coil/ relay/ contact)



**Available forms:** NO ( 1 ) and NC contacts and output coils ( 2 )

**Devices numbered in:** Octal, i.e. Y0 to Y7, Y10 to Y17

**Further uses:** None

**Available devices:**

PLC Inputs/outputs	20 points	30 points	40 points	60 points	Max
X (X000~X267 184 points)	X000~X013 12 points	X000~X017 16 points	X000~X027 24 points	X000~X043 36 points	X000~X177 128 points
Y(Y000~Y267 184 points)	Y000~Y007 8 points	Y000~Y005 14 points	Y000~Y017 16 points	Y000~Y027 24 points	Y000~Y177 128 points

- For more information about the device availability for individual PLC's, please see chapter 7.

### 3.3 Auxiliary Relays

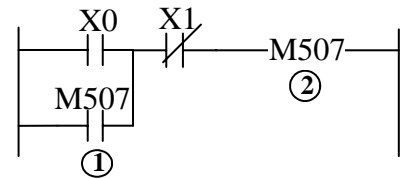
**Device Mnemonic:** M

**Purpose:** Internal programmable controller status flag

**Alias:** Auxiliary (coil/ relay/ contact/ flag)

M (coil/ relay/ contact /flag)

M (bit) device



**Available forms:** NO ( 1 ) and NC contacts and output coils ( 2 )

**Devices numbered in:** Decimal, i.e. M0 to M9, M10 to M19

**Further uses:** General stable state auxiliary relays - see 3.3.1

Battery backed/ latched auxiliary relays - see 3.3.2

Special diagnostic auxiliary relays - see 3.3.3

#### 3.3.1 General Stable State Auxiliary Relays

- A number of auxiliary relays are used in the PLC. The coils of these relays are driven by device contacts in the PLC in the same manner that the output relays are driven in the program.

All auxiliary relays have a number of electronic NO and NC contacts which can be used by the PLC as required. Note that these contacts cannot directly drive an external load.

Only output relays can be used to do this.

	General auxiliary relay 1	Battery backed /latched relay 2	Battery backed /latched relay 3	Special auxiliary relay
M	M0~M499 500 points	M500~M1023 524 points	M1024~M7679 6656 points	M8000~M8511 512 points

1: Non-retentive. However, the retentive-device range can be modified through PC-LINK parameter setting.

2: Retentive. However, the retentive-device range can be modified through PC-LINK parameter setting.

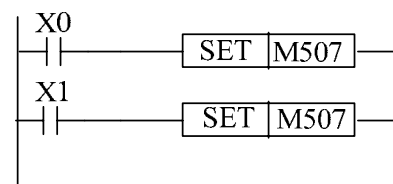
3: Retentive range is fixed, that is it can not be modified through PC-LINK.

For more information about device availability for individual PLC's, please see chapter 7.

#### 3.3.2 Battery Backed/ Latched Auxiliary Relays

There are a number of battery backed or latched relays whose status is retained in battery backed or EEPROM memory. If a power failure should occur all output and general purpose relays are switched off. When operation is resumed the previous status of these relays is restored. The example shows a self retaining circuit. Relay M507 is activated when X0 is turned ON. If X0 is turned OFF after the activation of M507, the ON status of M507 is self retained, i.e. the NO contact M507 drives the coil M507.

However, M507 is reset (turned OFF) when the input X1 is turned ON, i.e. the NC contact is broken.



A SET and RST (reset) instruction can be used to retain the status of a relay being activated momentarily.

#### External loads:

- Auxiliary relays are provided with countless number of NO contact points and NC contact points. These are freely available for use through out a PLC program. These contacts cannot be used to directly drive external loads. All external loads should be driven through the use of direct (Y) outputs.

### 3.3.3 Special Diagnostic Auxiliary Relays

A PLC has a number of special auxiliary relays. These relays all have specific functions and are classified into the following two types.

#### a) Using contacts of special auxiliary relays

- Coils are driven automatically by the PLC. Only the contacts of these coils may be used by a user defined program.

Examples: M8000: RUN monitor (ON during run)

M8002: Initial pulse (Turned ON momentarily when PLC starts)

M8012: 100 msec clock pulse

#### b) Driving coils of special auxiliary relays

- A PLC executes a predetermined specific operation when these coils are driven by the user.

Examples: M8033: All output statuses are retained when PLC operation is stopped

M8034: All outputs are disabled

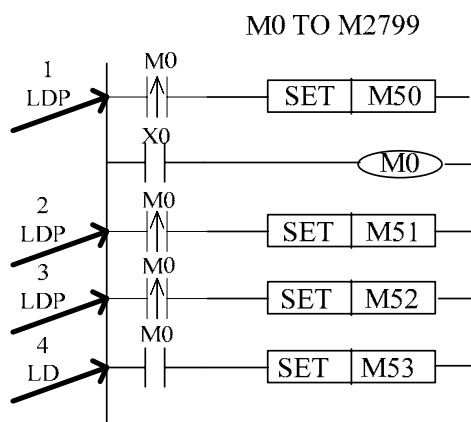
M8039: The PLC operates under constant scan mode

#### Available devices:

- Not all PLC's share the same range, quantity or operational meaning of diagnostic auxiliary relays. Please check the availability and function before using any device.

### 3.3.4 Special Single Operation Pulse Relays

When used with the pulse contacts LDP, LDF, etc., M devices in the range M2800 to M3072 have a special meaning. With these devices, only the next pulse contact instruction after the device coil is activated.



Turning ON X0 causes M0 to turn ON  
 Contacts ○1, ○2 and ○3 are pulse contacts and activate for 1 scan.  
 Contact ○4 is a normal LD contact and activates while M0 is ON

### 3.4 State Relays

**Device Mnemonic:** S

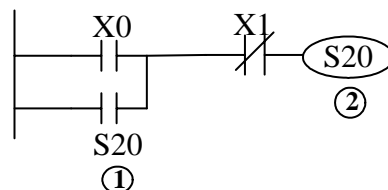
**Purpose:** Internal programmable controller status flag

**Alias:** State (coil/ relay/ contact/ flag)

S (coil/ relay/ contact /flag)

STL step (coil/ relay/ contact /flag)

Annunciator flag



**Available forms:** NO ( 1 ) and NC contacts and output coils ( 2 )

**Devices numbered in:** Decimal, i.e. S0 to S9, S10 to S19

**Further uses:** General stable state - state relays - see 3.4.1

Battery backed/ latched state relays - see 3.4.2

STL step relays - see 3.4.3

Annunciator flags - see 3.4.4

#### 3.4.1 General Stable State - State Relays

A number of state relays are used in the PLC. The coils of these relays are driven by device contacts in the PLC in the same manner that the output relays are driven in the program. All state relays have a number of electronic NO and NC contacts which can be used by the PLC as required. Note that these contacts cannot directly drive an external load. Only output relays can be used to do this.

**Available devices:**

- Please see the information point 3.4.2 'Battery backed/ latched state relays', or see the relevant tables for the selected PLC in chapter 7

#### 3.4.2 Battery Backed/ Latched State Relays

There are a number of battery backed or latched relays whose status is retained in battery backed or EEPROM memory. If a power failure should occur all output and general purpose relays are switched off. When operation is resumed the previous status of these relays is restored.

**Available devices:**

General state relay 1	Initial state relay	For zero-return of ITS	Battery backed/latched relay 2	Annunciator relay 2
S0~S499 500 points	S0~S9 10 points	S10~S19 10 points	S500 ~ S4095 3596 points	S900 ~ S999 100 points

1: Non-retentive. However, the retentive-device range can be modified through PC-LINK parameter setting.

2: Retentive. However, the retentive-device range can be modified through PC-LINK parameter setting.

- For more information about device availability for individual PLC's, see chapter 7.

**External loads:**

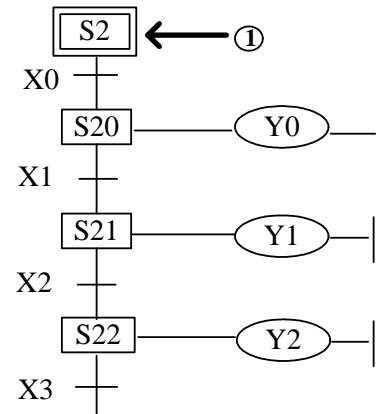
- State relays are provided with countless number of NO contact points and NC



contact points, and are freely available for use through out a PLC program. These contacts cannot be used to directly drive external loads. All external loads should be driven through the use of direct (ex. Y) outputs.

### 3.4.3 STL Step Relays

States (S) are very important devices when programming step by step process control. They are used in combination with the basic instruction STL. When all STL style programming is used certain states have a pre-defined operation. The step identified as 1 in the figure opposite is called an 'initial state'. All other state steps are then used to build up the full STL function plan. It should be remembered that even though remaining state steps are used in an STL format, they still retain their general or latched operation status. The range of available devices is as specified in the information point of the previous section.



#### Assigned states:

- When the applied instruction IST (Initial state function 60) is used, the following state devices are automatically assigned operations which cannot be changed directly by a users program:

S0 : Manual operation initial state

S1 : Zero return initial state

S2 : Automatic operation initial state

S10 to S19 : Allocated for the creation of the zero return program sequence

#### Monitoring STL programs:

- To monitor the dynamic-active states within an STL program, special auxiliary relay M8047 must be driven ON.

#### STL/SFC programming:

- For more information on STL/SFC style programming, please see chapter 2.

#### IST instruction:

- For more information on the IST instruction please see 4.7.1

### 3.4.4 Annunciator Flags

Some state flags can be used as outputs for external diagnosis (called annunciation) when certain applied instructions are used. These instructions are;

ANS function 46: Annunciator Set - see 4.5.7

ANR function 47: Annunciator Reset - see 4.5.8

When the annunciator function is used the controlled state flags are in the range S900 to S999 (100 points). By programming an external diagnosis circuit as shown below, and monitoring special data register D8049, the lowest activated state from the annunciator range will be displayed.

Each of the states can be assigned to signify an error or fault condition. As a fault occurs the associated state is driven ON. If more than one fault occurs simultaneously, the lowest fault number will be displayed. When the active fault is cleared the next lowest fault will then be processed.

This means that for a correctly prioritized diagnostic system the most dangerous or damaging faults should activate the lowest state flags, from the annunciator range. All state flags used for the annunciator function fall in the range of battery backed/ latched state registers.

Monitoring is enabled by driving special auxiliary relay M8049 ON.

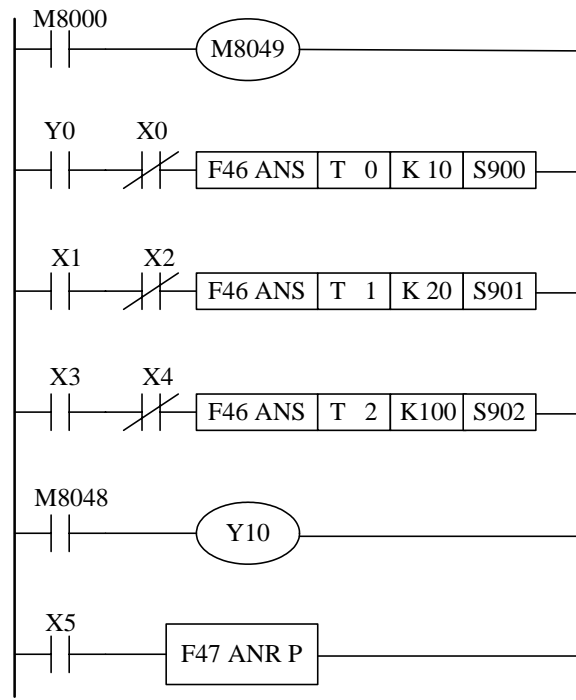
State S900 is activated if input X0 is not driven within one second after the output Y0 has been turned ON.

State S901 is activated when both inputs X1 and X2 are OFF for more than two seconds. If the cycle time of the controlled machine is less than ten seconds, and input X3 stays ON, state

S902 will be set ON if X4 is not activated within this machine cycle time.

If any state from S900 to S999 is activated, i.e. ON, special auxiliary relay M8048 is activated to turn on failure indicator output Y10.

The states activated by the users error / Failure diagnosis detection program, are turned OFF by activating input X5. Each time X5 is activated, the active annunciator states are reset in ascending order of state numbers



### 3.5 Pointers

**Device Mnemonic:** P

**Purpose:** Program flow control

Alias: Pointer

Program pointer : P

**Available forms:** Label: appears on the left of the left hand bus bar when the program is viewed in ladder mode.

**Devices numbered in:** Decimal, i.e. P0 to P9, P10 to P19

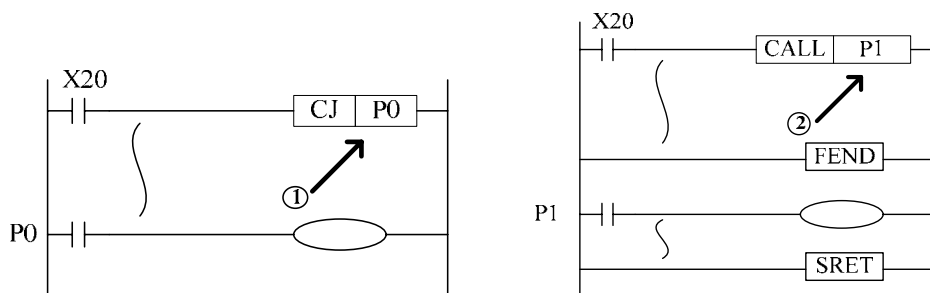
**Further uses:** Can be used with conditional jump statements (CJ function 00)

- see 4.1.1 and item 1 on the example device usage diagram.

Can be used with call statements

- see 4.1.2 and item 2 on the example device usage diagram

**Example device usage:**



**Available devices:**

- PLC has 256 pointers; available from the range of P0 to P255.

**Jumping to the end of the program:**

- When using conditional jump instructions (CJ, function 00) the program end can be jumped to automatically by using the pointer P63 within the CJ instruction. Labeling the END instruction with P63 is not required.

**Device availability:**

- For more information about device availability for individual PLC's, please see chapter 7.

### 3.6 Interrupt Pointers

**Device Mnemonic:** I

**Purpose:** Interrupt program marker

**Alias:** Interrupt

High speed interrupt: I

**Available forms:** Label: appears on the left of the left hand bus bar when the program is viewed in ladder mode

**Devices numbered in:** Special numbering system based on interrupt device used and input triggering method

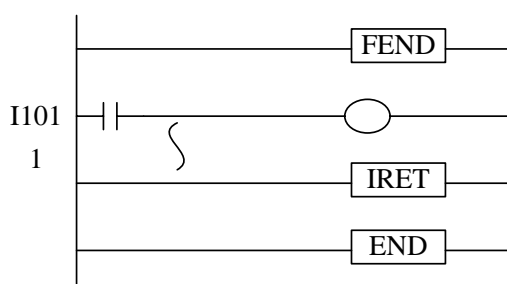
**Further uses:** Input interrupts - see 3.6.1

Timer interrupts - see 3.6.2

Disabling interrupts - see 3.6.3

Counter interrupts - see 3.6.4

**Example device usage:**



**Additional applied instructions:**

- Interrupts are made up of an interrupt device, an interrupt pointer and various usage of three, dedicated interrupt applied instructions;
  - IRET function 03: interrupt return - see 4.1.4
  - EI function 04: enable interrupt - see 4.1.4
  - DI function 05: disable interrupt - see 4.1.4

**Nested levels:**

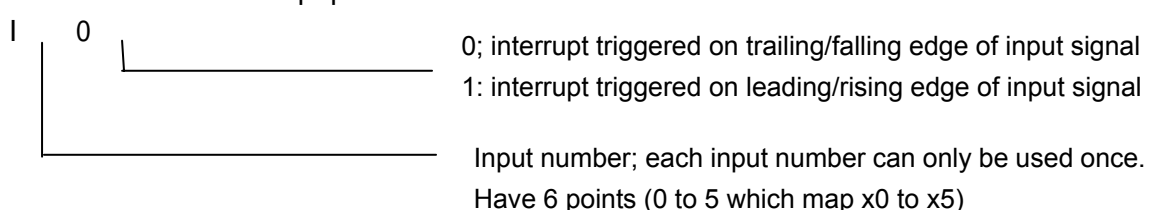
- While an interrupt is processing all other interrupts are disabled. To achieve nested interrupts the EI-DI instruction must be programmed within an interrupt routine. Interrupts can be nested for two levels.

**Pointer position:**

- Interrupt pointers may only be used after an FEND instruction (first end instruction, function 06).

#### 3.6.1 Input Interrupts

Identification of interrupt pointer number:



Example: I001

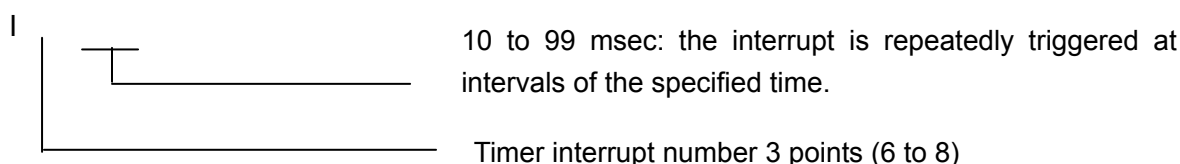
The sequence programmed after the label (indicated by the I001 pointer) is executed on the leading or rising edge of the input signal X0. The program sequence returns from the interruption program when an IRET instruction is encountered.

#### Rules of use:

- The following points must be followed for an interrupt to operate;
  - Interrupt pointers cannot have the same number in the '100's' position, i.e. I100 and I101 are not allowed.
  - The input used for the interrupt device must not coincide with inputs already allocated for use by other high speed instructions within the user program.

### 3.6.2 Timer Interrupts

Identification of interrupt pointer number:



Example: I610

The sequence programmed after the label (indicated by the I610 pointer) is executed at intervals of 10msec. The program sequence returns from the interruption program when an IRET instruction is encountered.

#### Rules of use:

- The following points must be followed for an interrupt to operate;
  - Interrupt pointers cannot have the same number in the '100's' position, i.e. I610 and I650 are not allowed.

### 3.6.3 Disabling Individual Interrupts

Individual interrupt devices can be temporarily or permanently disabled by driving an associated special auxiliary relay. The relevant coils are identified in the tables of devices in chapter 5. However for all PLC types the head address is M8050, this will disable interrupt I0

#### Driving special auxiliary relays:

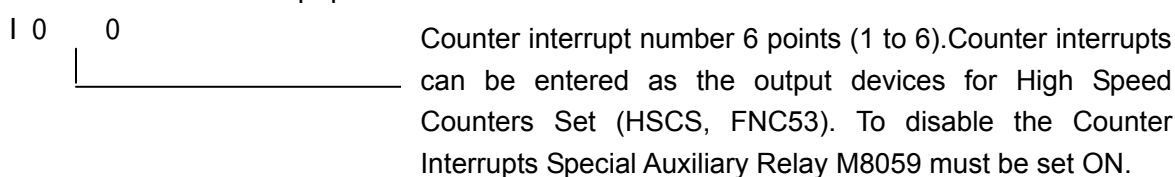
- Never drive a special auxiliary coil without first checking its use. Not all PLC's assign the same use to the same auxiliary coils.

#### Disabling high speed counter interrupts

- These interrupts can only be disabled as a single group by driving M8059 ON. Further details about counter interrupts can be found in the following section.

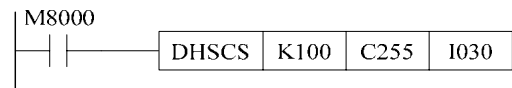
### 3.6.4 Counter Interrupts

Identification of interrupt pointer number:



Example:

The sequence programmed after the label (indicated by the I030 pointer) is executed once the value of High Speed Counter C255



reaches/equals the preset limit of K100 identified in the example HSCS.

**Additional notes:**

- Please see the following pages for more details on the HSSC applied instruction.
  - High Speed Counter Set, HSCS FNC 53 - see 4.6.4

### 3.7 Constant K

**Device Mnemonic:** K

**Purpose:** Identification of constant decimal values

**Alias:** Constant

K (value/ constant)

K

**Available forms:** Numeric data value, when used for 16bit data, values can be selected from the range -32,768 to +32,767. For 32bit data, values from the range -2,147,483,648 to + 2,147,483,647 can be used.

**Devices numbered in:** N/A. This device is a method of local instruction data entry. There is no limit to the number of times it can be used.

**Further uses:** K values can be used with timers, counters and applied instructions

**Example device usage:** N/A

### 3.8 Constant H

**Device Mnemonic:** H

**Purpose:** Identification of constant hexadecimal values

**Alias:** Constant

H (value/ constant)

Hex (value/ constant)

H

**Available forms:** Alpha-numeric data value, i.e. 0 to 9 and A to F (base 16). When used for 16bit data, values can be selected from the range 0 to FFFF. For 32bit data, values from the range 0 to FFFFFFFF can be used.

**Devices numbered in:** N/A. This device is a method of local instruction data entry. There is no limit to the number of times it can be used.

**Further uses:** Hex values can be used with applied instructions

**Example device usage:** N/A

### 3.9 Timers

**Device Mnemonic:** T

**Purpose:** Timed durations

**Alias:** Timer(s)

T

**Available forms:** A driven coil sets internal PLC contacts (NO and NC contacts available).

Various timer resolutions are possible, from 1 to 100 msec, but availability and quantity vary from PLC to PLC. The following variations are also available:

Selectable timer resolutions - see **3.9.2**

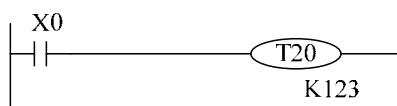
Retentive timers - see **3.9.3**

Timers used in interrupt and 'CALL' subroutines - see **3.9.4**

**Devices numbered in:** Decimal, i.e T0 to T9, T10 to T19.

**Further uses:** None

**Example device usage:**



**Available devices:**

function	100ms type 0.1~3276.7s	10ms type 0.01~327.67s	1ms accumulating type 0.001~32.767s	100ms accumulating type 0.1~3276.7s	1 ms type	Potentiometer 0~1024
general	T0~T199	T200~T245	T246~T249	T250~T255	T256~T511	2 point
For sub-routine	T192~T199					

**Timer accuracy:**

- See **3.9.5**

#### 3.9.1 General timer operation

Timers operate by counting clock pulses (1, 10 and 100 msec). The timer output contact is activated when the count data reaches the value set by the constant K. The overall duration or elapsed time, for a timers operation cycle, is calculated by multiplying the present value by the timer resolution, i.e.

A 10 msec timer with a present value of 567 has actually been operating for:

$$567 \times 10 \text{ msec}$$

$$567 \times 0.01 \text{ sec} = 5.67 \text{ seconds}$$

Timers can either be set directly by using the constant K to specify the maximum duration or indirectly by using the data stored in a data register (ex. D). For the indirect setting, data registers which are battery backed/ latched are usually used; this ensures no loss of data during power down situations. If however, the voltage of the battery used to perform the battery backed service, reduces excessively, timer malfunctions may occur.

### 3.9.2 Selectable Timers

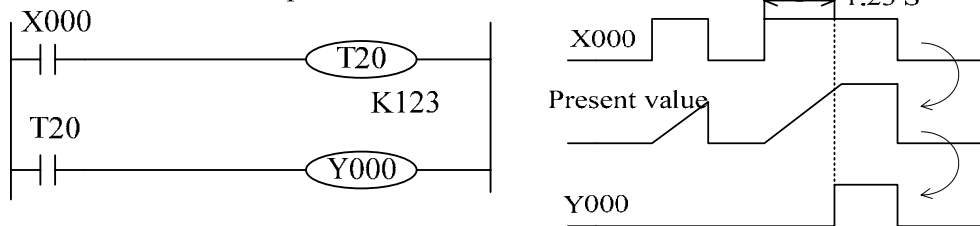
On certain programmable controllers, driving a special auxiliary coil redefines approximately half of the 100 msec timers as 10 msec resolution timers. The following PLC's and timers are subject to this type of selection.

### 3.9.3 Retentive Timers

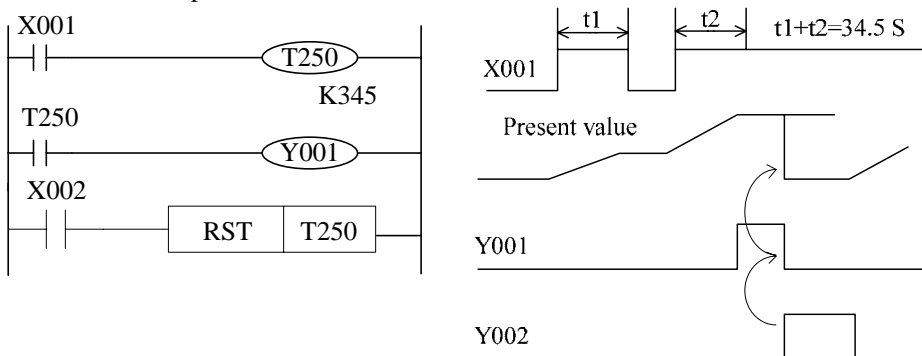
A retentive timer has the ability to retain the currently reached present value even after the drive contact has been removed. This means that when the drive contact is re-established a retentive timer will continue from where it last reached.

Because the retentive timer is not reset when the drive contact is removed, a forced reset must be used. The following diagram shows this in a graphical format.

Non-retentive timer operation



Retentive timer operation



### Using timers in interrupt or 'CALL' subroutines:

- Please see 3.9.4

### 3.9.4 Timers Used in Interrupt and 'CALL' Subroutines

If timers T192 to T199 and T246 to T249 are used in a CALL subroutine or an interruption routine, the timing action is updated at the point when an END instruction is executed. The output contact is activated when a coil instruction or an END instruction is processed once the timer's current value has reached the preset (maximum duration) value.

Timers other than those specified above cannot function correctly within the specified circumstances.

When an interrupt timer (1 msec resolution) is used in an interrupt routine or within a 'CALL' subroutine, the output contact is activated when the first coil instruction of that timer is executed after the timer has reached its preset (maximum duration) value.

### 3.9.5 Timer Accuracy



Timer accuracy can be affected by the program configuration. That is to say, if a timer contact is used before its associated coil, then the timer accuracy is reduced.

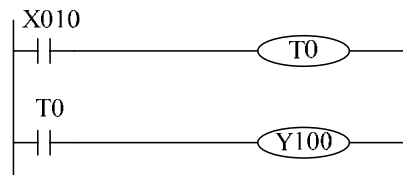
The following formulas give maximum and minimum errors for certain situations.

However, an average expected error would be approximately;

$$1.5 \times \text{The program scan time}$$

**Condition 1:**

The timer contact appears after the timer coil.



Maximum timing error:

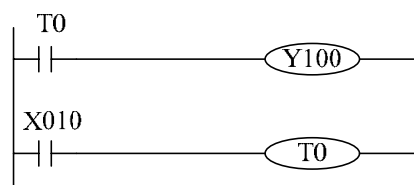
$$2 \times \text{Scan time} + \text{The input filter time}$$

Minimum timing error:

$$\text{Input filter time} - \text{The timer resolution}$$

**Condition 2:**

The timer contact appears before the timer coil.



Maximum timing error:

$$3 \times \text{Scan time} + \text{The input filter time}$$

Minimum timing error:

$$\text{Input filter time} - \text{The timer resolution}$$

**Internal timer accuracy:**

- The actual accuracy of the timing elements within the PLC hardware is;  $\pm 10$  pulses per million pulses. This means that if a 100 msec timer is used to time a single day, at the end of that day the timer will be within 0.8 seconds of the true 24 hours or 86,400 seconds. The timer would have processed approximately 864,000; 100 msec

### 3.10 Counters

**Device Mnemonic:** C

**Purpose:** Event driven delays

**Alias:** Counter(s)

C

**Available forms:** A driven coil sets internal PLC contacts (NO and NC contacts available).

Various counter resolutions are possible including;

General/latched 16bit up counters - see **3.10.1**

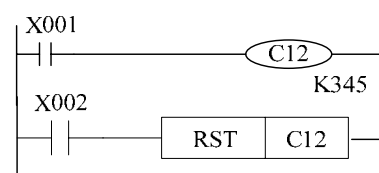
General/latched 32bit bi-directional counters – see **3.10.2**

(The availability and use of all these counters is PLC specific – please check availability before use)

**Devices numbered in:** Decimal, i.e C0 to C9, C10 to C19

**Further uses:** None

**Example device usage:**



**Available devices:**

General 16bit up counter 0~32,767	Latched 16bit up counter 0~32,767	Latched 32bit up counter -2,147,483,648~+2,147,483,647
C0~C099	C100~C199	C200~C255

**High speed counters:**

- For high speed counters please see **3.11**

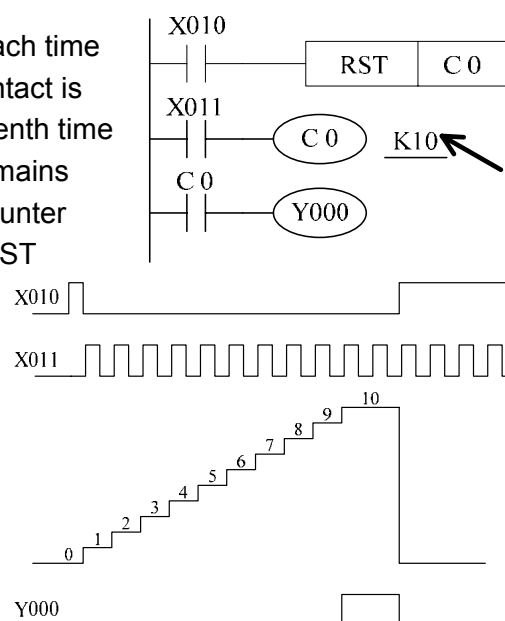
**Setting ranges for counters:**

- 16bit counters: -32,768 to +32,767
- 32bit counters: -2,147,483,648 to +2,147,483,647

#### 3.10.1 General/ Latched 16bit UP Counters

The current value of the counter increases each time coil C0 is turned ON by X011. The output contact is activated when the coil is turned ON for the tenth time (see diagram). After this, the counter data remains Unchanged when X011 is turned ON. The counter current value is reset to '0' (zero) when the RST

instruction is executed by turning ON X010 in the example. The output contact Y000 is also reset at the same time. Counters can be set directly using constant K or indirectly by using data stored in a data register (ex. D). In an indirect setting, the designation of D10 for example, which contains the value "123" has the same effect as a setting of "K123". If a value greater than the counter setting is written to a current value register, the counter counts up when the next input



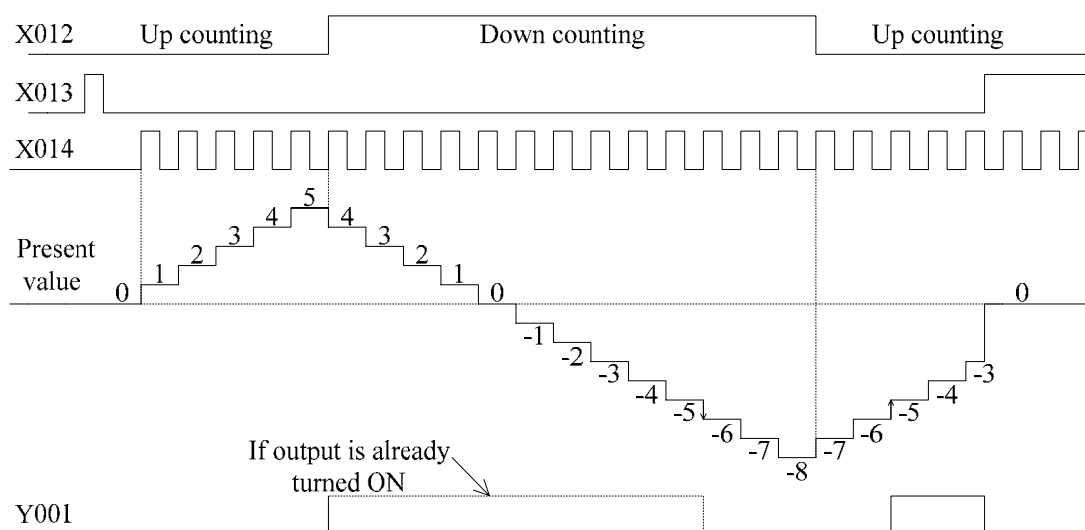
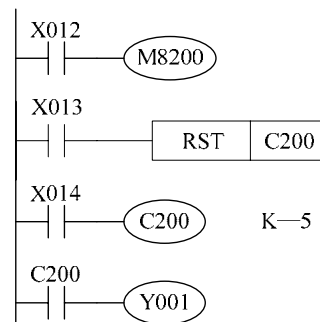
is turned ON. This is true for all types of counters. Generally, the count input frequency should be around several cycles per second.

### Battery backed/latched counters:

- Counters which are battery backed/ latched are able to retain their status information, even after the PLC has been powered down. This means on re-powering up, the latched counters can immediately resume from where they were at the time of the original PLC power down.

#### 3.10.2 General/ Latched 32bit Bi-directional Counters

The counter shown in the example below, activates when its coil is driven, i.e. the C200 coil is driven. On every occasion the input X014 is turned from OFF to ON the current value or current count of C200 is incremented.



The output coil of C200 is set ON when the current value increases from “-6” to “-5”. However, if the counters value decreases from “-5” to “-6” the counter coil will reset. The counters current value increases or decreases independently of the output contact state (ON/OFF). Yet, if a counter counts beyond +2,147,483,647 the current value will automatically change to -2,147,483,648. Similarly, counting below -2,147,483,648 will result in the current value changing to +2,147,483,647. This type of counting technique is typical for “ring counters”. The current value of the active counter can be reset to “0” (zero) by forcibly resetting the counter coil; in the example program by switching the input X013 ON which drives the RST instruction. The counting direction is designated with special auxiliary relays M8200 to M8234.

### Battery backed/ latched counters:

- Counters which are battery backed/ latched are able to retain their status

information, even after the PLC has been powered down. This means on re-powering up, the latched counters can immediately resume from where they were at the time of the original PLC power down.

#### Selecting the counting direction:

- If M8 for C is turned ON, the counter will be a down counter. Conversely, the counter is an up counter when M8 is OFF.

### 3.11 High Speed Counters

**Device Mnemonic:** C

**Purpose:** High speed event driven delays

**Alias:** Counter (s)

C

High speed counter (s)

Phase counters

**Available forms:** A driven coil sets internal PLC contacts (NO and NC contacts available).

There are various types of high speed counter available but the quantity and function vary from PLC to PLC.

**Devices numbered in:** Decimal, i.e C0 to C235 to C255

**Further uses:** None

**Example device usage:** For examples on each of the available forms please see the relevant sections.

#### 3.11.1 Basic High Speed Counter Operation

Although counters C235 to C255 (21 points) are all high speed counters, they share the same range of high speed inputs. Therefore, if an input is already being used by a high speed counter, it cannot be used for any other high speed counters or for any other purpose, i.e. as an interrupt input.

The selection of high speed counters are not free, they are directly dependent on the type of counter required and which inputs are available.

Available counter types;

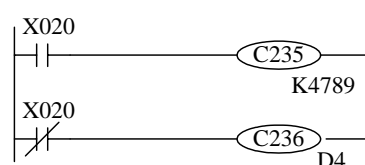
- 1 phase: C235 to C245
- 1 phase bi-directional: C246 to C249
- 2 phase bi-directional: C251 to C254

Please note ALL of these counters are 32bit devices.

High speed counters operate by the principle of interrupts. This means they are event triggered and independent of cycle time. The coil of the selected counter should be driven continuously to indicate that this counter and its associated inputs are reserved and that other high speed processes must not coincide with them.

#### Example:

When X020 is ON, high speed counter C235 is selected. The counter C235 corresponds to count input X000. X020 is NOT the counted signal. This is the continuous drive mentioned earlier. X000



does not have to be included in the program. The input assignment is hardware related and cannot be changed by the user.

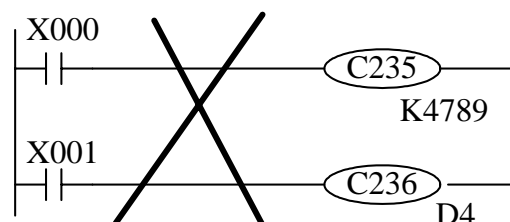
When X020 is OFF, coil C235 is turned OFF and coil C236 is turned ON. Counter C236 has an assigned input of X001; again the input X020 is NOT the counted input. The assignment of counters and input devices is dependent upon the PLC selected. This is explained in the relevant, later sections.

#### Driving high speed counter coils:

- The counted inputs are NOT used to drive the high speed counter coils.

This is because the counter coils need to be continuously driven ON to reserve the associated high speed inputs.

Therefore, a normal non-high speed drive contact should be used to drive the high speed counter coil. Ideally the special auxiliary contact M8000 should be used. However, this is not compulsory.



### 3.12 Data Registers

**Device Mnemonic:** D

**Purpose:** A storage device capable of storing numeric data or 16/32bit patterns

**Alias:** Data (register/ device/ word)

D (register)

D

Word

**Available forms:** General use registers

Battery backed/latched registers

Special diagnostic registers

File registers

**Devices numbered in:** Decimal, i.e. D0 to D9, D10 to D19

**Further uses:** Can be used in the indirect setting of counters and timers

**Example device usage:** None

**Available devices:**

General use registers	Latched registers	File registers R	Special diagnostic registers
D0~D199 200 points	D200~D511 312 points	D512 ~ D7999 7488 points	D8000~D8511 512 points

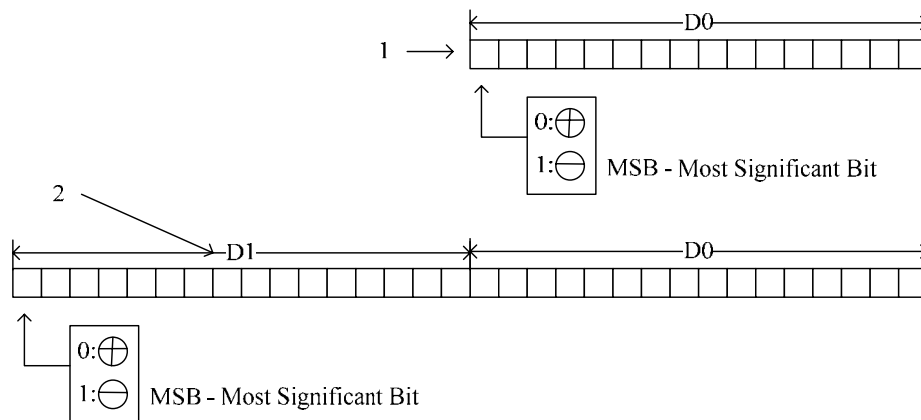
R - These devices are allocated by the user at the expense of available program steps.

#### 3.12.1 General Use Registers

Data registers, as the name suggests, store data. The stored data can be interpreted as a numerical value or as a series of bits, being either ON or OFF.

A single data register contains 16bits or one word. However, two consecutive data registers can be used to form a 32bit device more commonly known as a double word.

If the contents of the data register is being considered numerically then the Most Significant Bit (MSB) is used to indicate if the data has a positive or negative bias. As bit devices can only be ON or OFF, 1 or 0 the MSB convention used is, 0 is equal to a positive number and 1 is equal to a negative number.



The diagram above shows both single and double register configurations. In the diagram identified, it should be noted that the 'lower' register D0 no longer has a 'Most Significant Bit'. This is because it is now being considered as part of a 32bit-double word. The MSB will always be found in the higher 16 bits, i.e. in this case D1. When specifying a 32 bit data register within a program instruction, the lower device is always used e.g. if the above example was to be written as a 32bit instructional operand it would be identified as D0. The second register, D1, would automatically be associated.

Once the data is written to a general data register, it remains unchanged until it is overwritten. When the PLC is turned from RUN to STOP, all of the general data registers have their current contents overwritten with a 0 (zero).

#### Data retention:

- Data can be retained in the general use registers when the PLC is switched from RUN to STOP if special auxiliary relay M8033 is ON.

#### Data register updates:

- Writing a new data value to a data register will result in the data register being updated with the new data value at the end of the current program scan.

### 3.12.2 Special Diagnostic Registers

Special registers are used to control or monitor various modes or devices inside the PLC. Data written in these registers are set to the default values when the power supply to the PLC is turned ON.

- Note: When the power is turned ON, all registers are first cleared to 0 (zero) and then the default values are automatically written to the appropriate registers by the system software. For example, the watchdog timer data is written to D8000 by the system software. To change the setting, the user must write the required value over what is currently stored in D8000.

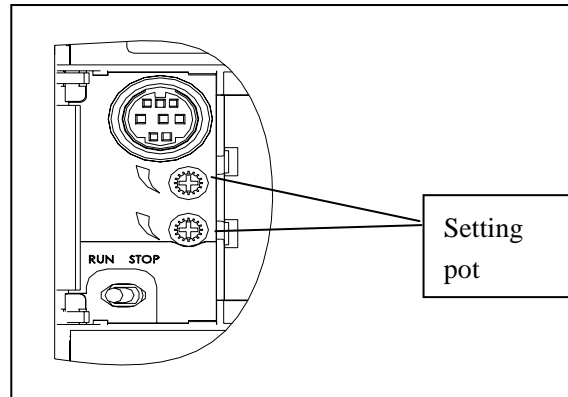
Data stored in the special diagnostic registers will remain unchanged when the PLC is switched from STOP mode into RUN.

**Use of diagnostic registers:**

- On no account should unidentified devices be used. If a device is used, it should only be for the purpose identified in this manual. Please see chapter 5 for tables containing data and descriptions of the available devices for each PLC.

**3.12.3 Externally Adjusted Registers**

The PLC has built in “setting pots” which are used to adjust the contents of certain dedicated data registers. The contents of these registers can range from 0 to 1023. This is a built in feature and requires no additional setup or programming. And an additional special function unit is also available which provides the same function. To use this unit requires the applied instructions VRRD function 85 (Volume Read) and VRSC function 86 (Volume Scale).



Number of setting pots	2 points: Supplied by using basic unit 6 points: Supplied by using the additional special function block.
Number of controlled data registers	Selected by the user when applied instructions VRRD and VRSC are used

**Uses:**

- This facility is often used to vary timer settings, but it can be used in any application where a data register is normally found, i.e. setting counters, supplying raw data, even selection operations could be carried out using this option.

**3.13 Index Registers**

**Device Mnemonic:** V,Z

**Purpose:** To modify a specified device by stating an offset.

**Alias:** (V/ Z) Register

Index (register/ addressing/ modifier)

Offset(s) (register/ addressing/ modifier)

Indices

Modifier

**Available forms:**

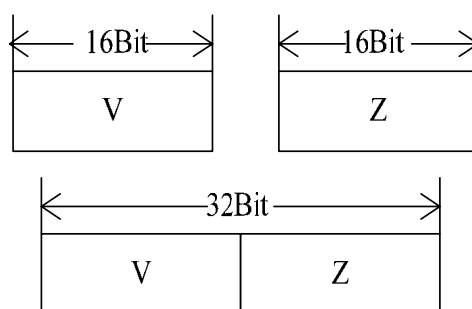
For 16bit data V or Z

(2 devices)

For 32bit data V and Z combined

(1 device - Z is specified)

Operation is similar to data registers.



**Devices numbered in:** N/A. there are 32 devices V0 – V15 and Z0 – Z15

**Further uses:** Can be used to modify the following devices under certain conditions;

X, Y, M, S, P, T, C, D, K, H, KnX, KnY, KnM, KnS

**Example device usage:**

The program shown right transfers data from D5V to D10Z.

If the data contained in register V is equal to 8 and the data in register Z is equal to 14, then:

V = 8

D5V

D5 + 8 = 13 → D13

Z = 14

D10Z

D10 + 14 = 24 → D24

Hence, the actual devices used after the modifiers V and Z have been taken into account are; D13 and D24 and not D5 and D10 respectively.

**Use of Modifiers with Applied Instruction Parameters:**

- All applied instruction parameters should be regarded as being able to use index registers to modify the operand except where stated otherwise.

### 3.13.1 Modifying a Constant

Constants can be modified just as easily as data registers or bit devices. If, for example, the constant K20 was actually written K20V the final result would equal:

K20 + the contents of V

Example:



$$\text{If } V=3276 \text{ then } K20V \rightarrow \frac{K \quad 20}{V \quad 3276} \frac{V \quad 3276}{3296}$$

### 3.13.2 Misuse of the Modifiers

Modifying Kn devices when Kn forms part of a device description such as KnY is not possible, i.e. while the following use of modifiers is permitted;

K3Z

K1M10V

Y20Z

Statements of the form:

K4ZY30

are not acceptable.

- Modifiers cannot be used for parameters entered into any of the 20 basic instructions, i.e. LD, AND, OR etc.

### 3.13.3 Using Multiple Index Registers

The use of multiple index registers is sometimes necessary in larger programs or programs which handle large quantities of data. There is no problem from the PLC's point of view in using both V and Z registers many times through out a program. The point to be aware of is that it is sometimes confusing for the user or a maintenance person reading such programs, as it is not always clear what the current value of V or Z is.

Example:

V = 10 (K10)

Z = 20 (K20)

D5V = D15 (D5 + V = D5 + 10 = D15)

D15Z = D35 (D15 + Z = D15 + 20 = D35)

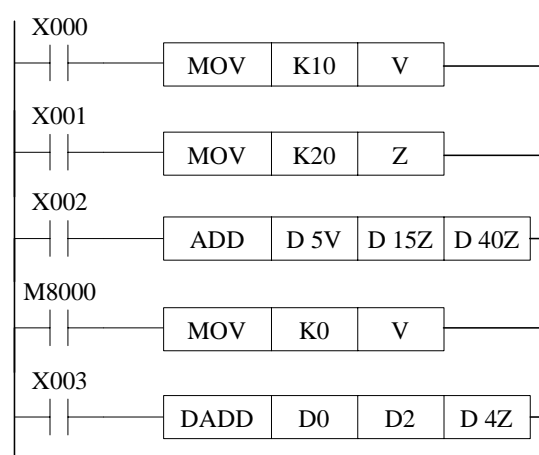
D40Z = D60 (D40 + Z = D40 + 20 = D60)

Both V and Z registers are initially set to K10 and K20 respectively.

The content of D15 is added to that of D35 and store in D60.

V is then reset to 0 (zero) and both V and Z are used in the double word addition (DADD).

The contents of D1, D0 are then added to D3, D2 and then finally stored in D25, D24.



## 3.14 Bits, Words, BCD and Hexadecimal

The following section details general topics relating to good device understanding. The section is split into several smaller parts with each covering one topic or small group of topics.

**Available devices:**

- For PLC specific available devices please see chapter 7.

### 3.14.1 Bit Devices, Individual and Grouped

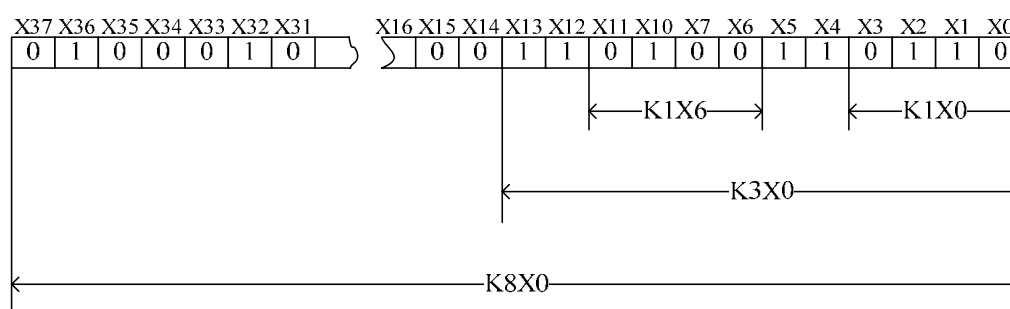
Devices such as X, Y, M and S are bit devices. Bit devices are bi-stable, this means there are only two states, ON and OFF or 1 and 0. Bit devices can be grouped together to form bigger representations of data, for example 8 consecutive bit devices are some-times referred to as a byte. Further more, 16 consecutive bit devices are referred to as a word and 32 consecutive bit devices are a double word.

The PLC identifies groups of bit devices which should be regarded as a single entity by looking for a range marker followed by a head address. This is of the form KnP where P represents the head address of the bit devices to be used. The Kn portion of the statement identifies the range of devices enclosed. “n” can be a number from the range 0 to 8. Each “n” digit actual represents 4 bit devices, i.e K1 = 4 bit devices and K8 = 32 bit devices. Hence all groups of bit devices are divisible by 4.

#### Assigning grouped bit devices:

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

K2M0, for example identifies 2 groups of 4 bits; M0 to M3 and M4 to M7, giving a total of 8 bit devices or 1 byte. The diagram below identifies more examples of Kn use.



- K1X0 : X0 to X3 4 bit devices with a head address of X0
- K1X6 : X6 to X11 4 bit devices with a head address of X6
- K3X0 : X0 to X13 12 bit devices with a head address of X0
- K8X0 : X0 to X37 32 bit devices with a head address of X0

#### Moving grouped bit devices:

- If a data move involves taking source data and moving it into a destination which is smaller than the original source, then the overflowing source data is ignored. For example; If K3M20 is moved to K1M0 then only M20 to M23 or K1M20 is actually moved. The remaining data K2M24 or M24 to M31 is ignored.

#### Assigning I/O:

- Any value taken from the available range of devices can be used for the head address 'marker' of a bit device group. However, it is recommended to use a 0 (zero) in the lowest digit place of X and Y devices (X0, X10, X20.....etc). For M and S devices, use of a multiple of “8” is the most device efficient. However, because the use of such numbers may lead to confusion in assigning device numbers, it recommended to use a

multiple of "10". This will allow good correlation to X and Y devices.

### 3.14.2 Word Devices

Word devices such as T, C, D, V and Z can store data about a particular event or action within the PLC. For the most part these devices are 16 bit registers. However, certain variations do have 32 bit capabilities, as can pairs of consecutive data registers or combined V and Z registers.

It may seem strange to quote the size of a word device in bits. This is not so strange when it is considered that the bit is the smallest unit of data within the PLC. So by identifying every thing in bit format a common denomination is being used, hence comparison etc is much easier.

Additional consequences of this bit interpretation is that the actual data can be interpreted differently. The physical pattern of the active bits may be the important feature or perhaps the numerical interpretation of the bit pattern may be the key to the program. It all comes down to how the information is read.

### 3.14.3 Interpreting Word Data

As word data can be read in many ways the significance of certain parts of the word data can change. PLC's can read the word data as:

- A pure bit pattern
- A decimal number
- A hexadecimal number
- Or as a BCD (Binary Coded Decimal) number

The following examples will show how the same piece of data can become many different things depending wholly on the way the information is read or interpreted.

#### a) Considering a bit pattern

The following bit pattern means nothing - it is simply 16 devices which have two states. Some of the devices are randomly set to one of the states. However, if the header notation (base 2) is added to the 16 bit data the sum, decimal, total of the active bits can be calculated, e.g.,

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



MSB	$2^{14}$	$2^{13}$	$2^{12}$	$2^{11}$	$2^{10}$	$2^9$	$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
1	0	0	1	1	1	1	0	0	1	1	1	0	1	0	1

$$\text{Decimal value} = (2^0 \times 1) + (2^2 \times 1) + (2^4 \times 1) + (2^5 \times 1) + (2^6 \times 1) + (2^9 \times 1) + (2^{10} \times 1) + (2^{11} \times 1) + (2^{12} \times 1)$$

$$\text{Decimal value} = 7797$$

This is in fact incorrect!

There is one bit device which has been shaded in. If its header notation is studied carefully it will be noted that it says MSB. This is the Most Significant Bit. This single bit device will determine if the data will be interpreted as a positive or negative number. In this example the MSB is equal to 1. This means the data is negative.

The answer however, is not -7797.

The reason this is not -7797 is because a negative value is calculated using two's compliment (described later) but can quickly be calculated in the following manner: Because this is a negative number, a base is set as -32768. This is the smallest number available with 16bit data. To this the positive sum of the active bits is added, i.e. -32768 +7797.

The correct answer is therefore -24971.

Remember this is now a decimal representation of the original 16 bit - bit pattern. If the original pattern was re-assessed as a hexadecimal number the answer would be different.

b) A hexadecimal view

Taking the same original bit pattern used in point a) and now adding a hexadecimal notation instead of the binary (base 2) notation the bit patterns new meaning becomes:

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



8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
1	0	0	1	1	1	1	0	0	1	1	1	0	1	0	1

Hexadecimal value =  $(1 \times 8) + (1 \times 1)$  ,  $((1 \times 8) + (1 \times 4) + (1 \times 2))$  ,  $(1 \times 4) + (1 \times 2) + (1 \times 1)$  ,  $(1 \times 4) + (1 \times 1)$

Hexadecimal value = 9E75

Two things become immediately obvious after a hexadecimal conversion. The first is that there is sign bit as hexadecimal numbers are always positive.

The second is there is an "E" appearing in the calculated data. This is actually acceptable as hexadecimal counts from 0 to 15. But as there are only ten digits (0 to 9), substitutes need to be found for the remaining base 16 numbers, i.e. 10, 11, 12, 13, 14 and 15. The first six characters from the alphabet are used as the replacement indices, e.g. A to F respectively.

As a result of base 16 counting, 4 binary bits are required to represent one base 16 or hexadecimal number. Hence, a 16 bit data word will have a 4 digit hexadecimal code. There is actually a forth interpretation for this bit sequence. This is a BCD or Binary Coded Decimal reading. The following section converts the original bit pattern into a BCD format.

c) A BCD conversion

Using the original bit pattern as a base but adding the following BCD headers allows the conversion of the binary data into a BCD format.

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



8	4	2	1	8	4	2	1	8	4	2	1	8	4	2	1
1	0	0	1	1	1	1	0	0	1	1	1	0	1	0	1

Binary Coded Decimal value= ERROR!!!!

It will be noticed that this will produce an ERROR. The conversion will not be correct. This is because BCD numbers can only have values from 0 to 9, but the second block

of 4 bit devices from the left would have a value of 14. Hence, the error.

The conversion process is very similar to that of hexadecimal except for the mentioned limit on values of 0 to 9. If the other blocks were converted just as an example the following values would be found;

$$\text{Extreme Left Hand Block} = ((1 \times 8) + (1 \times 1)) = 9$$

$$\text{Second Right Hand Block} = ((1 \times 4) + (1 \times 2) + (1 \times 1)) = 7$$

$$\text{Extreme Right Hand Block} = ((1 \times 4) + (1 \times 1)) = 5$$

BCD data is read from left to right as a normal number would be read. Therefore, in this example the “9” would actually represent “9000”. The second right hand block is actually “70” not “7”. The units are provided by the extreme right hand block, i.e. 5. The hundreds “100’s” would have been provided by the second left hand block (which is in error). It is also important to note that there is no sign with BCD converted data. The maximum number allowable for a single data word is “9999” and the minimum is “0000”.

### Word Data Summary

In each of the previous cases the original bit pattern had a further meaning. To recap the three new readings and the original bit pattern,

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

Decimal : -24971

Hexadecimal : 9E75

BCD : Error (9?75)

Each meaning is radically different from the next yet they are all different ways of describing the same thing. They are in fact all equal to each other!

### 3.14.4 Two's Complement

Programmable controllers, computers etc, use a format called 2's compliment. This is a mathematical procedure which is more suited to the micro processors operational hardware requirements. It is used to represent negative numbers and to perform subtraction operations.

The procedure is very simple, in the following example “15 - 7” is going to be solved:

Step1: Find the binary values (this example uses 8 bits)

$$15 = 00001111$$

$$7 = 00000111$$

Step2: Find the inversion of the value to be subtracted.

Procedure: invert all 1's to 0's and all 0's to 1's.

$$7 = 00000111$$

$$\text{Inverted } 7 = 11111000$$

Step3: Add 1 to the inverted number.

Procedure: add 1 to the right hand most bit. Remember this is binary addition hence, when a value of 2 is obtained 1 is moved in to the next left hand position and the remainder is set to 0(zero);

$$\begin{array}{r} \text{Inverted } 7 \\ 11111000 \end{array}$$

$$\begin{array}{r} \text{Additional } 1 \\ 00000001 \end{array}$$

Answer        11111001

This result is actually the same as the negative value for 7 i.e. -7.

Step4: Add the answer to the number the subtraction is being made from (i.e. 15).

Procedure: Remember  $1+1 = 0$  carry 1 in base 2 (binary).

Original value15        00001111

Answer found in step3    11111001

Solution                (1)00001000

The "(1)" is a carried "1" and is ignored as this example is only dealing with 8 bits.

Step 5: Convert the answer back.

00001000 = 8

The answer is positive because the MSB (the most left hand bit) is a 0 (zero). If a quick mental check is made of the problem it is indeed found that " $15-7 = 8$ ".

In fact no subtraction has taken place. Each of the steps has either converted some data or performed an addition. Yet the answer is correct  $15 - 7$  is 8. This example calculation was based on 8 bit numbers but it will work equally well on any other quantity of bits.

### 3.15 Floating Point And Scientific Notation

PLC's can use many different systems and methods to store data.

The most common have already been discussed in previous sections e.g. BCD, Binary, Decimal, Hex. These are what is known as 'integer' formats or 'whole number formats'.

As the titles suggest these formats use only whole numbers with no representation of fractional parts. However, there are two further formats which are becoming increasingly important and they are:

- a) Floating point and

## b) Scientific notation

Both of these formats are in fact closely related. They both lend themselves to creating very large or very small numbers which can describe both whole and fractional components.

**General note:**

- Sometimes the words 'Format', 'Mode' and 'Notation' are interchanged when descriptions of these numerical processes are made. However, all of these words are providing the same descriptive value and as such users should be aware of their existence.

**Some useful constants**

$\pi$	$3.141 \times 10^0$
$2\pi$	$6.283 \times 10^0$
$\pi/4$	$7.853 \times 10^{-1}$
$\pi^2$	$9.869 \times 10^0$
The speed of light	$2.997 \times 10^8$ m/s
Gravity, g	$9.807 \times 10^0$ m/s <sup>2</sup>
e	$2.718 \times 10^0$

Fixed points:

Boiling point of liquid oxygen	$-1.8297 \times 10^2$ °C
Melting point of ice	$0.00 \times 10^0$ °C
Triple point of water	$1.00 \times 10^{-2}$ °C
Boiling point of water	$1.00 \times 10^2$ °C

**3.15.1 Scientific Notation**

This format could be called the step between the 'integer' formats and the full floating point formats. In basic terms Scientific Notation use two devices to store information about a number or value. One device contains a data string of the actual characters in the number (called the mantissa), while the second device contains information about the number of decimal places used in the number (called the exponent). Hence, Scientific Notation can accommodate values greater/smaller than the normal 32 bit limits, i.e.

-2,147,483,648 to 2,147,483,647 where Scientific Notation limits are;

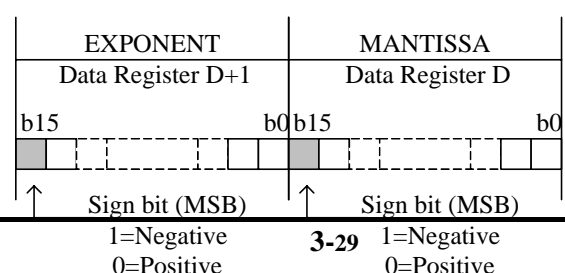
Maximums	Minimums
$9999 \times 10^{35}$	$9999 \times 10^{-41}$
$-9999 \times 10^{35}$	$-9999 \times 10$

Scientific Notation can be obtained by using the BCD, or EBCD, instruction (FNC 18 or FNC 118) with the float flag M8023 set ON. In this situation floating point format numbers are converted by the BCD instruction into Scientific Notation

Scientific Notation can be converted back to floating point format by using the BIN instruction (FNC 19) with the float flag M8023 set ON

The following points should be remembered about the use of Scientific Notation :

- The mantissa and exponent are stored in consecutive data registers.  
Each part is made up of 16 bits and can be assigned a positive or negative value



indicated by the value of the most significant bit (MSB, or bit 15 of the data register) for each number.

- The mantissa is stored as the first 4 significant figures without any rounding of the number, i.e. a floating point number of value  $2.34567 \times 10^3$  would be stored as a mantissa of 2345 at data register D and an exponent of 0 (zero) at data register D+1.
- The range of available mantissa values is 0, 1000 to 9999 and -1000 to -9999.
- The range of available exponent values is +35 through to -41.
- Scientific format cannot be used directly in calculations, but it does provide an ideal method of displaying the data on a monitoring interface.

### 3.15.2 Floating Point Format

Floating point format extends the abilities and ranges provided by Scientific Notation with the ability to represent fractional portions of whole numbers, for example; performing and displaying the calculation of 22 divided by 7 would yield the following results:

- Normal PLC operation using decimal (integers) numbers would equal 3 remainder 1
- In floating point it would equal 3.14285 (approximately)
- In Scientific format this calculation would be equal to  $3142 \times 10^{-3}$

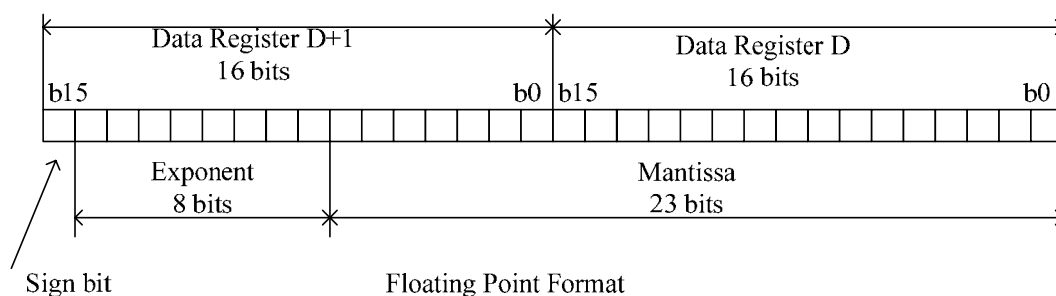
So it can be seen that a greater degree of accuracy is provided by floating point numbers, i.e. through the use of larger numerical ranges and the availability of more calculable digits. Hence, calculations using floating point data have some significant advantages.

Decimal data can be converted in to floating point by using the FLT, float instruction (FNC 49). When this same instruction is used with the float flag M8023 set ON, floating point numbers can be converted back to decimal.

The following points should be remembered about the use of Floating Point;

- Floating point numbers, no matter what numerical value, will always occupy two consecutive data registers (or 32 bits).
- Floating point values cannot be directly monitored, as they are stored in a special format recommended by the I.E.E.E (Institute of Electrical and Electronic Engineers) for personal and micro computer applications.
- Floating point numbers have both mantissa and exponents (see Scientific Notation for an explanation of these terms). In the case of floating point exponents, only 8 bits are used.

Additionally there is a single sign bit for the mantissa. The remaining bits of the 32 bit value, i.e. 23 bits, are used to 'describe' the mantissa value.



Valid ranges for floating point numbers as used :

Description	Sign	exponent	Mantissa	Remark
-------------	------	----------	----------	--------



Normal Float	0 or 1	11111110 00000001	11111111111111111111111111111111 11111111111111111111111111111110 00000000000000000000000000000001 00000000000000000000000000000000	Large number $\pm 3.403 \times 10^{38}$ Accuracy: 7 significant figures Smallest number $\pm 1.175 \times 10^{38}$
Zero	0 or 1	00000000	00000000000000000000000000000000	All digits are 0

## 4. Applied Instructions

Applied Instructions are the specialist instruction of PLC. They allow the user to perform complex data manipulations, mathematical operations while still being very easy to program and monitor. Each applied instruction has unique mnemonics and special function numbers. Each applied instruction will be expressed using a table similar to that shown below:

Mnemonic	Function	Operands	Program steps
		D	
CJ FNC 00(Conditional Jump)	A method of jumping to an identified pointer position	Valid pointers from the range 0 to 63	CJ,CJP:3 steps Jump pointer P : 1 step

The table will be found at the beginning of each new instruction description. The area identified as 'Operands' will list the various devices (operands) that can be used with the instruction. Various identification letters will be used to associate each operand with its function, i.e. destination, S- source, n, m- number of elements. Additional numeric suffixes will be attached if there are more than one operand with the same function.

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

The 'pulse' function allows the associated instruction to be activated on the rising edge of the control input. The instruction is driven ON for the duration of one program scan. Thereafter, while the control input remains ON, the associated instruction is not active. To re-execute the instruction the control input must be turned from OFF to ON again. The FLAGS section identifies any flags that are used by the instruction. Details about the function of the flag are explained in the instructions text.

- For instructions that operate continuously, i.e. on every scan of the program the instruction will operate and provide a new, different result, the following identification symbol will be used ' ' to represent a high speed changing state. Typical instructions covered by this situation have a strong incremental, index able element to their operation.
- In most cases the operands of applied instructions can be indexed by a users program. For those operands which cannot be indexed, the symbol ' ' has been used to signify an operand as being 'fixed' after it has been written.
- Certain instructions utilize additional data registers and/or status flags for example a math function such as ADD (FNC 20) can identify a zero result, borrow and carry conditions by using preset auxiliary relays, M8020 to M8021 respectively.

## 4.1 Program Flow-Functions00 to 09

### Contents:

CJ -	Conditional jump	FNC 00
CALL -	Call Subroutine	FNC 01
SRET -	Subroutine Return	FNC 02
IRET -	Interrupt Return	FNC 03
EI -	Enable Interrupt	FNC 04
DI -	Disable Interrupt	FNC 05
FEND -	First End	FNC 06
WDT -	Watchdog Timer	FNC 07
FOR -	Start of a For/Next Loop	FNC 08
NEXT -	End a For/Next Loop	FNC 09

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D1, S3 or for lists/abled devices D3+0, S+9 etc.

MSB -Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

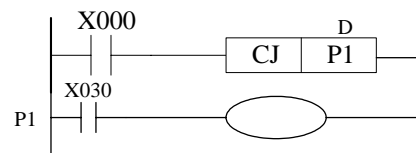
- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
- P - A 16 bit mode instruction modified to use pulse (single) operation.
- D - An instruction modified to operate in 32 bit operation.
- D P - A 32 bit mode instruction modified to use pulse (single) operation.
- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.
- An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

## 4.1.1 CJ (FNC 00)

Mnemonic	Function	Operands	Program steps
		D	
CJ FNC 00 (Conditional Jump)	jumps to an identified pointer position	Valid pointers from the range 0 to 62,64 to 255	CJ,CJP:3 steps Jump pointer P :1 step

**Operation:**

When the CJ instruction is active it forces the program to jump to an identified program marker. While the jump takes place the intervening pro-gram steps are skipped. This means they are not processed in any way. The resulting effect is to speed up the programs operational scan time.

**Points to note:**

a) Many CJ statements can reference a single pointer.

b) Each pointer must have a unique number. Using pointer P63 is equivalent to jumping to the END instruction.

c) Any program area which is skipped, will not update output statuses even if the input devices change.

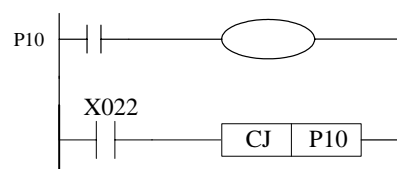
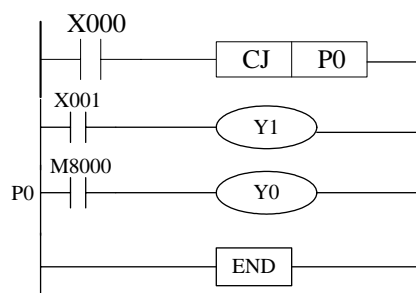
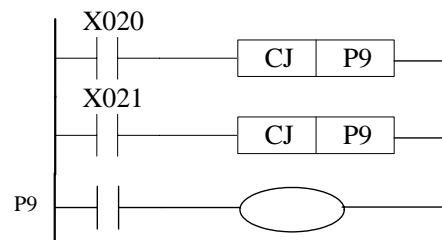
For example, the program opposite shows a situation which loads X001 to drive Y1. Assuming X001 is ON and the CJ instruction is activated the load X001, out Y1 is skipped. Now even if X001 is turned OFF Y1 will remain ON while the CJ instruction forces the program to skip to the pointer P0. The reverse situation will also apply, i.e. if X001 is OFF to begin with and the CJ instruction is driven, Y1 will not be turned ON if X001 is turned ON. Once the CJ instruction is deactivated X001 will drive Y1 in the normal manner.

This situation applies to all types of outputs, e.g. SET, RST, OUT, Y, M and S devices

d) The CJ instruction can jump to any point within the main program body or after an FEND instruction

e) A CJ instruction can be used to Jump forwards through a program, i.e. towards the END instruction OR it can jump backwards towards step 0. If a backwards jump is used care must be taken not to overrun the watchdog timer setting otherwise the PLC will enter an error situation.

f) Unconditional jumps can be entered by using special auxiliary coils such as M8000. In this situation while the PLC is in RUN the program will ALWAYS execute the CJ instruction in an unconditional manner.

**IMPORTANT:**

- Timers and counters will freeze their current values if they are skipped by a CJ instruction. For example if Y1 in the previous program (see point c) was replaced by T0 K100 and the

CJ instruction was driven, the contents of T0 would not change/increase until the CJ instruction is no longer driven, i.e. the current timer value would freeze.

High speed counters are the only exception to this situation as they are processed independently of the main program.

Using applied instructions:

- Applied instructions are also skipped if they are programmed between the CJ instruction and the destination pointer. However, The PLSY (FNC 57) and PWM (FNC 58) instructions will operate continuously if they were active before the CJ instruction was driven, otherwise they will be processed, i.e. skipped, as standard applied instructions.

Details of using CJ with other program flow instructions

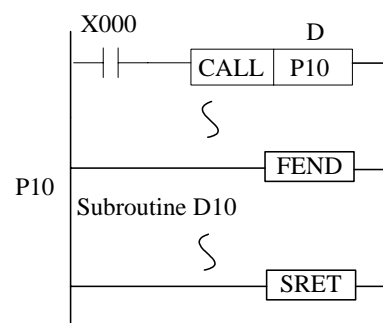
- • Further details can be found on pages 7-12 and 7-13 about the combined use of different program flow techniques (such as master control, MC etc).

## 4.1.2 CALL (FNC 01)

Mnemonic	Function	Operands	Program steps
		D	
CALL FNC 01 (Call Sub-routine)	Executes the subroutine program starting at the identified pointer position	Valid pointers from the range 0 to 62, 63 to 255 Nest levels: 5 including the initial CALL	CALL, CALLP: 3 steps Subroutine pointer PPP: 1 step

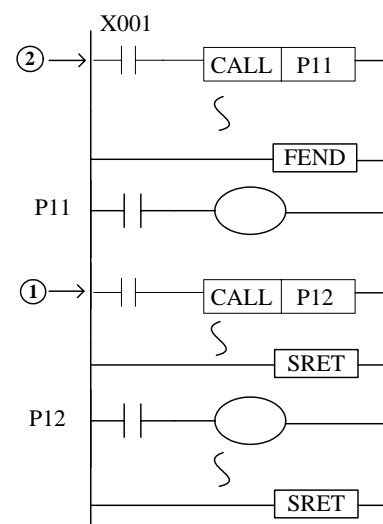
**Operation:**

When the CALL instruction is active it forces the program to run the subroutine associated with the called pointer (area identified as subroutine P10). A CALL instruction must be used in conjunction with FEND (FNC 06) and SRET (FNC 02) instructions. The program jumps to the subroutine pointer (located after an FEND instruction) and processes the contents until an SRET instruction is encountered. This forces the program flow back to the line of ladder logic immediately following the original CALL instruction

**Points to note:**

- Many CALL statements can reference a single subroutine.
- Each subroutine must have a unique pointer number. Subroutine pointers can be selected from the range P0 to P62. Subroutine pointers and the pointers used for CJ (FNC 00) instructions are NOT allowed to coincide.
- Subroutines are not normally processed as they occur after an FEND instruction. When they are called, care should be taken not to overrun the watchdog timer setting.
- Subroutines can be nested for 16 levels including the initial CALL instruction. As an example the program showed opposite shows a 2 level nest.

When X001 is activated the program calls subroutine P11. Within this subroutine is a CALL to a second subroutine P12. When both subroutines P11 and P12 are active simultaneously, they are said to be nested. Once subroutine P12 reaches its SRET instruction it returns the program control to the program step immediately following its original CALL (see 1). P11 then completes its operation, and once its SRET instruction is processed the program returns once again to the step following the CALL P11 statement (see 2).



**4.1.3 SRET (FNC 02)**

Mnemonic	Function	Operands	Program steps
		D	
SRET FNC 02 (Sub-routine return)	Returns operation from a subroutine program	N/A Automatically returns to the step immediately following the CALL instruction which activated the subroutine	SRET:1 step

**Operation:**

SRET signifies the end of the current subroutine and returns the program flow to the step immediately following the CALL instruction which activated the closing subroutine.

**Points to note:**

- SRET can only be used with the CALL instruction.
- SRET is always programmed after an FEND instruction - please see the CALL (FNC 01) instruction for more details.

**4.1.4 IRET, EI, DI (FNC 03, 04, 05)**

Mnemonic	Function	Operands	Program steps
		D	
IRET FNC 03 (Interrupt return)	Forces the program to return from the active interrupt routine	N/A Automatically returns to the main program step which was being processed at the time of the interrupt call.	IRET: 1 step
EI FNC 04 (Enable interrupts)	Enables interrupt inputs to be processed	N/A Any interrupt input being activated after an EI instruction and before FEND or DI instructions will be processed immediately unless it has been specifically disabled	EI: 1 step
DI FNC 05 (Disable interrupts)	Disables the processing of interrupt routines	N/A Any interrupt input being activated after a DI instruction and before an EI instruction will be stored until the next sequential EI instruction is processed.	DI: 1 step
I (Interrupt pointer)	Identifies the beginning of an interrupt routine	A 3 digit numeric code relating to the interrupt type and operation	I□□□: 1 step

**General description of an interrupt routine:**

An interrupt routine is a section of program which is, when triggered, operated immediately interrupting the main program flow. Once the interrupt has been processed the main program flow continues from where it was, just before the interrupt originally occurred.

**Operation:**

Interrupts are triggered by different input conditions, sometimes a direct input such as X0 is

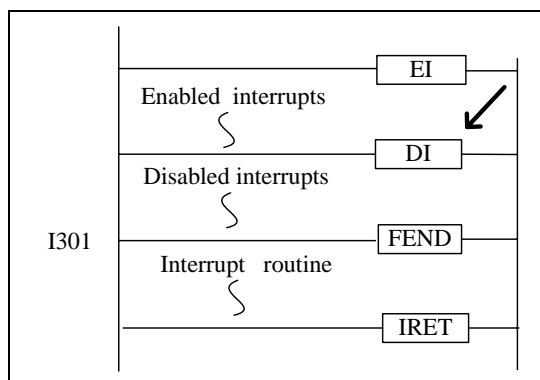
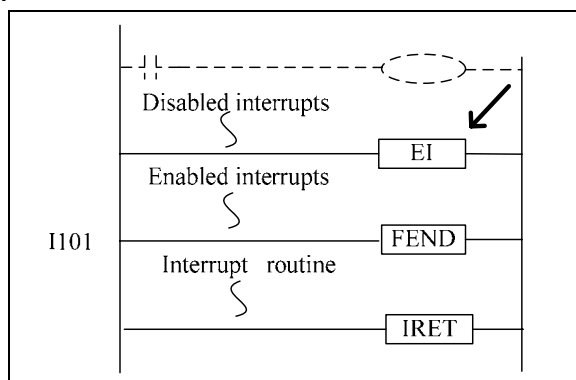
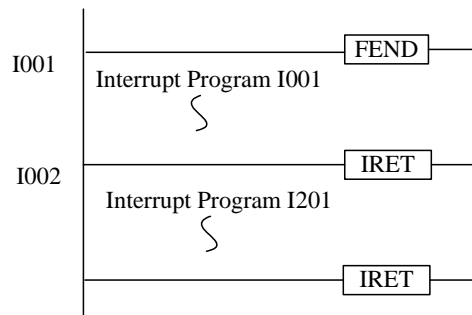
used other times a timed interval e.g. 30 msec can be used. To program and operate interrupt routines requires up to 3 dedicated instructions (those detailed in this section) and an interrupt pointer.

### Defining an interrupt routine:

An interrupt routine is specified between its own unique interrupt pointer and the first occurrence of an IRET instruction. Interrupt routines are ALWAYS programmed after an FEND instruction. The IRET instruction may only be used within interrupt routines.

### Controlling interrupt operations:

The PLC has a default status of disabling interrupt operation. The EI instruction must be used to activate the interrupt facilities. All interrupts which physically occur during the program scan period from the EI instruction until the FEND or DI instructions will have their associated interrupt routines run. If these interrupts are triggered outside of the enclosed range (EI-FEND or EI-DI, see diagram below) they will be stored until the EI instruction is processed on the following scan. At this point the interrupt routine will be run.



If an individual interrupt is to be disabled its associated special M coil must be driven ON. While this coil is ON the interrupt routine will not be activated. For details about the disabling M coils see the PLC device tables in chapter 7.

### Nesting interrupts:

Interrupts may be nested for two levels. This means that an interrupt may be interrupted during its operation. However, to achieve this, the interrupt routine which may be further interrupted must contain the EI and DI instructions; otherwise as under normal operation, when an interrupt routine is activated all other interrupts are disabled.

### Simultaneously occurring interrupts:

If more than one interrupt occurs sequentially, priority is given to the interrupt occurring first. If two or more interrupts occur simultaneously, the interrupt routine with the lower pointer number is given the higher priority.

### Using general timers within interrupt routines:

The PLC has a range of special timers which can be used within interrupt routines. Timers Used in Interrupt and 'CALL' Subroutines.

### Input trigger signals - pulse duration:

Interrupt routines which are triggered directly by interrupt inputs, such as X0 etc., require a signal duration of approximately 20µsec, i.e. the input pulse width is equal or greater than



200 $\mu$ sec. When this type of interrupt is selected, the hardware input filters are automatically reset to 50 $\mu$ sec. (under normal operating circumstances the input filters are set to 10msec.).

#### Pulse catch function:

Direct high speed inputs can be used to 'catch' short pulsed signals. When a pulse is received at an input a corresponding special M coil is set ON. This allows the 'captured' pulse to be used to trigger further actions, even if the original signal is now OFF. The PLC requires the EI instruction (FNC 04) to activate pulse catch for inputs X0 through X5, with M8170 to M8175 indicating the caught pulse. Note that, if an input device is being used for another high speed function, then the pulse catch for that device is disabled.

#### 4.1.5 FEND (FNC 06)

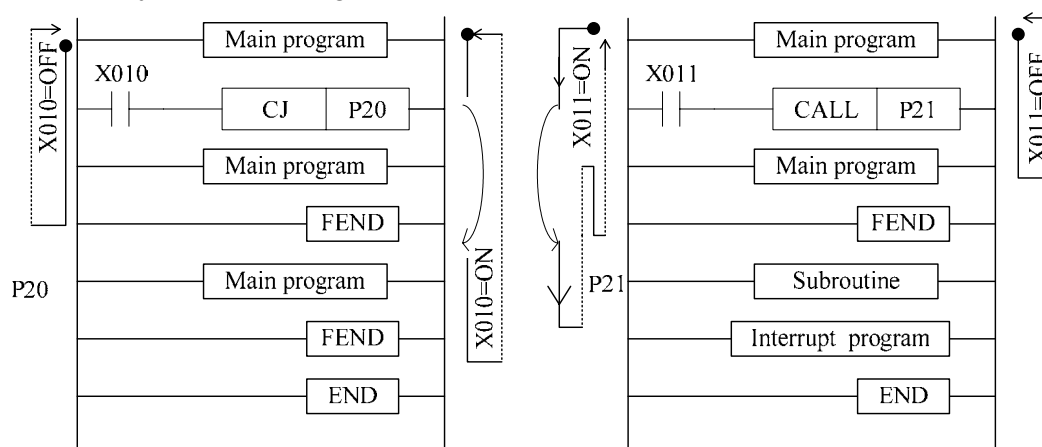
Mnemonic	Function	Operands	Program steps
		D	
FEND FNC 06 (First end)	Used to indicate the end of the main program block	N/A Note: Can be used with CJ (FNC 00), CALL (FNC 01) and interrupt routines	FEND: 1 step

#### Operation:

An FEND instruction indicates the first end of a main program and the start of the program area to be used for subroutines. Under normal operating circumstances the FEND instruction performs a similar action to the END instruction, i.e. output processing, input processing and watchdog timer refresh are all carried out on execution.

#### Points to note:

- The FEND instruction is commonly used with CJ-P-FEND, CALL-P-SRET and I-IRET program constructions (P refers to program pointer, I refers to interrupt pointer). Both CALL pointers/subroutines and interrupt pointers (I) subroutines are ALWAYS programmed after an FEND instruction, i.e. these program features NEVER appear in the body of a main program.



- Multiple occurrences of FEND instructions can be used to separate different subroutines (see diagram above).
- The program flow constructions are NOT allowed to be split by an FEND instruction.
- FEND can never be used after an END instruction.

## 4.1.6 WDT (FNC 07)

Mnemonic	Function	Operands	Program steps
		D	
WDT FNC 07 (Watch dog timer refresh)	Used to refresh the watch dog timer during a program scan	N/A Can be driven at any time within the main program body	WDT, WDTP: 1 step

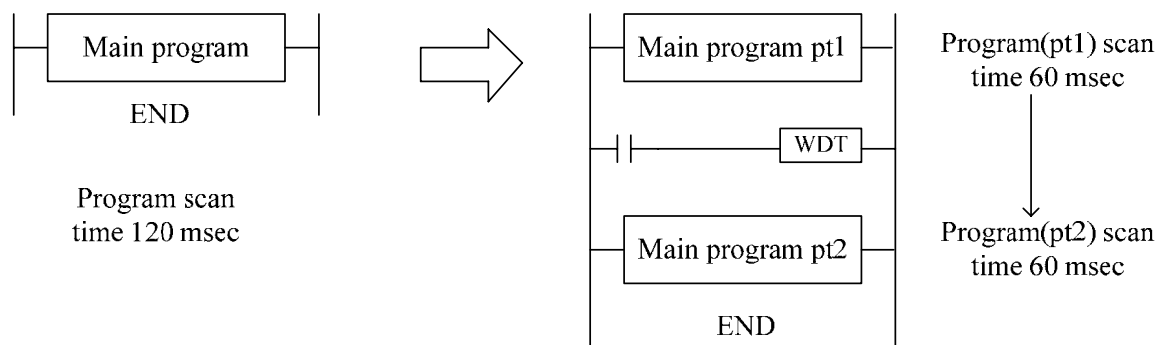
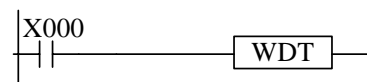
**Operation:**

The WDT instruction refreshes the PLC's watchdog timer.

The watchdog timer checks that the program scan

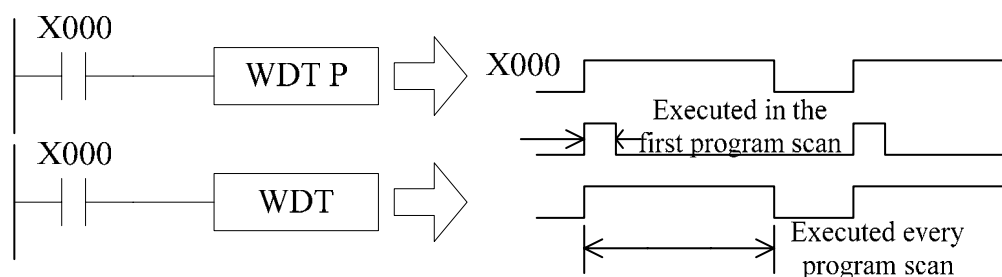
(operation) time does not exceed an arbitrary time limit. It is

assumed that if this time limit is exceeded there is an error at some point. The PLC will then cease operation to prevent any further errors from occurring. By causing the watchdog timer to refresh (driving the WDT instruction) the usable scan (program operation) time is effectively increased.

**Points to note:**

- a) When the WDT instruction is used it will operate on every program scan so long as its input condition has been made.

To force the WDT instruction to operate for only ONE scan requires the user to program some form of interlock. FX users have the additional option of using the pulse (P) format of the WDT instruction, i.e. WDTP.



- b) The watchdog timer has a default setting of 200 msec. This time limit may be customized to a users own requirement by editing the contents of data register D8000, the watchdog timer register.

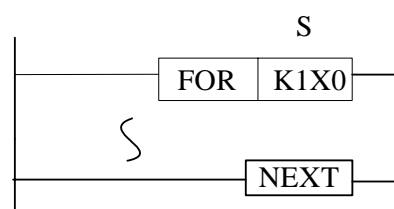


## 4.1.7 FOR, NEXT (FNC 08, 09)

Mnemonic	Function	Operands	Program steps
		D	
FOR FNC 08 (Start of a FOR-NEXT loop)	Identifies the start position and the number of repeats for the loop	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z	FOR: 3 step
NEXT FNC 09 (End of a FOR-NEXT loop)	Identifies the end position for the loop	N/A Note: The FOR-NEXT loop can be nested for 5 levels, i.e. 5 FOR-NEXT loops are programmed within each other	NEXT: 1 step

**Operation:**

The FOR and NEXT instructions allow the specification of an area of program, i.e. the program enclosed by the instructions, which is to be repeated S number of times.

**Points to note:**

a) The FOR instruction operates in a 16 bit mode hence, the value of the operand S may be within the range of 1 to 32,767. If a number between the range -32,768 and 0 (zero) is specified it is automatically replaced by the value 1, i.e. the FOR-NEXT loop would execute once.

b) The NEXT instruction has NO operand.

c) The FOR-NEXT instructions must be programmed as a pair e.g. for every FOR instruction there **MUST** be an associated NEXT instruction. The same applies to the NEXT instructions, there **MUST** be an associated FOR instruction. The FOR-NEXT instructions must also be programmed in the correct order. This means that programming a loop as a NEXT-FOR (the paired NEXT instruction proceeds the associated FOR instruction) is **NOT** allowed.

Inserting an FEND instruction between the FOR-NEXT instructions, i.e. FOR-FEND- NEXT, is NOT allowed. This would have the same effect as programming a FOR without a NEXT instruction, followed by the FEND instruction and a loop with a NEXT and no associated FOR instruction.

d) A FOR-NEXT loop operates for its set number of times **before** the main program is allowed to finish the current program scan.

e) When using FOR-NEXT loops care should be taken not to exceed the PLC's watchdog timer setting. The use of the WDT instruction and/or increasing the watchdog timer value is recommended.

**Nested FOR-NEXT loops:**

FOR-NEXT instructions can be nested for 16 levels. This means that 16 FOR-NEXT loops can be sequentially programmed within each other.

In the example a 3 level nest has been programmed. As each new FOR-NEXT nest level is

encountered the number of times that loop is repeated is increased by the multiplication of all of the surrounding/previous loops.

For example, loop C operates 4 times. But within this loop there is a nested loop, B. For every completed cycle of loop C, loop B will be completely executed, i.e. it will loop D0Z times. This again applies between loops B and A. The total number of times that loop A will operate for ONE scan of the program will equal;

- 1) The number of loop A operations multiplied by
- 2) The number of loop B operations multiplied by
- 3) The number of loop C operations

If values were associated to loops A, B and C, e.g. 7, 6 and 4 respectively, the following number of operations would take place in ONE program scan:

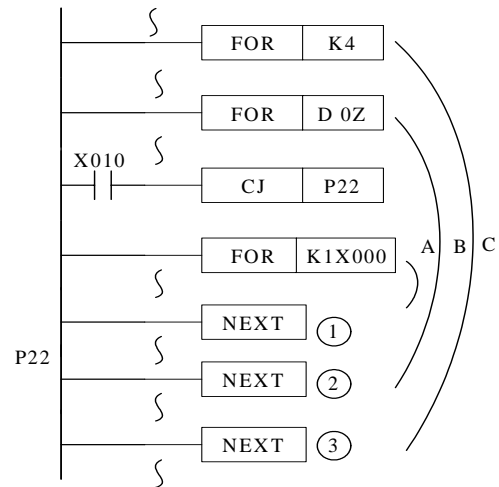
Number of loop C operations = 4 times

Number of loop B operations = 24 times ( $C \times B$ ,  $4 \times 6$ )

Number of loop A operations = 168 times ( $C \times B \times A$ ,  $4 \times 6 \times 7$ )

#### Note:

The use of the CJ programming feature, causing the jump to P22 allows the 'selection' of which loop will be processed and when, i.e. if X10 was switched ON, loop A would no longer operate.



## 4.2 Move And Compare - Functions 10 to 19

### Contents:

CMP -	Compare	FNC 10
ZCP -	Zone Compare	FNC 11
MOV -	Move	FNC 12
SMOV -	Shift Move	FNC 13
CML -	Compliment	FNC 14
BMOV -	Block Move	FNC 15
FMOV -	Fill Move	FNC 16
XCH -	Exchange	FNC 17
BCD -	Binary Coded Decimal	FNC 18
BIN -	Binary	FNC 19

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D1, S3 or for lists/abled devices D3+0, S+9 etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
- P - A 16 bit mode instruction modified to use pulse (single) operation.
- D - An instruction modified to operate in 32 bit operation.
- D P - A 32 bit mode instruction modified to use pulse (single) operation.

- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.

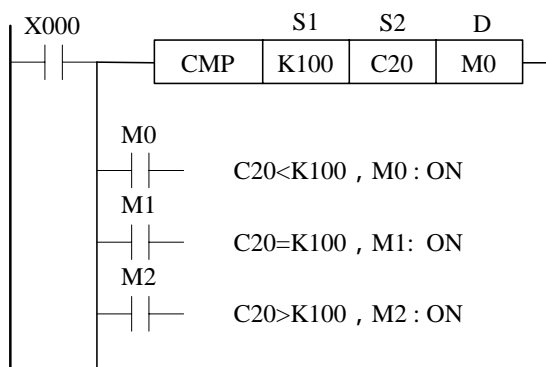
- An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

**4.2.1 CMP (FNC 10)**

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
CMP FNC 10 (Compare)	Compares two data values - results of <, = and > are given	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z		Y, M, S Note: 3 consecutive devices are used	CMP, CMPP: 7 steps DCMP, DCMPP: 13 steps

**Operation:**

The data of S1 is compared to the data of S2.  
The result is indicated by 3 bit devices specified from the head address entered as D.  
The bit devices indicate:  
S2 is less than S1 - bit device D is ON  
S2 is equal to S1 - bit device D+1 is ON  
S2 is greater than S1 - bit device D+2 is ON



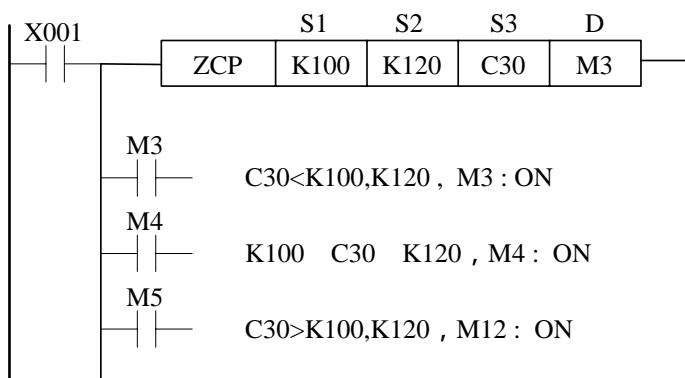
**Note:** The destination (D) device statuses will be kept even if the CMP instruction is deactivated. Full algebraic comparisons are used, i.e. -10 is smaller than +2 etc.

**4.2.2 ZCP (FNC 11)**

Mnemonic	Function	Operands				Program steps
		S1	S2	S3	D	
ZCP FNC 11 (Zone compare)	Compares a data value against a data range - results of <, = and > are given.	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z Note: S1 should be less than S2		Y, M, S Note: 3 consecutive devices are used.		ZCP, ZCPP: 9 steps DZCP, DZCPP: 17 steps

**Operation:**

The operation is the same as the CMP Instruction except a single data value (S3) is compared against a data range (S1-S2). S3 is less than S1 and S2 - bit device D is ON  
S3 is equal to or between S1 and S2 - bit device D+1 is ON  
S3 is greater than both S1 and S2 - bit device D+2 is ON

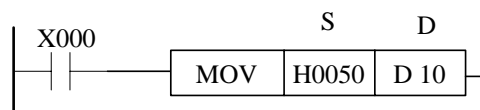


**4.2.3 MOV (FNC 12)**

Mnemonic	Function	Operands		Program steps
		S	D	
MOV FNC 12 (Move)	Moves data from one storage area to a new storage area	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D, V, Z	MOV, MOV P: 5 steps DMOV, DMOV P: 9 steps

**Operation:**

The contents of the source device (S) is copied to the destination (D) device when the control input is active. If the MOV instruction is not driven, no operation takes place.



**Note:** This instruction has a special programming technique which allows it to mimic the operation of newer applied instructions when used with older programming tools.

**4.2.4 SMOV (FNC 13)**

Mnemonic	Function	Operands					Program steps
		M1	M2	N	S	D	
SMOV FNC 13 (Shift move)	Takes elements of an existing 4 digit decimal number and inserts them into a new 4 digit number	K, H Note: available range 1 to 4			K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z Range 0 to 9,999 (decimal) or 0 to 9,999 (BCD) when M8168 is used	K, H, KnY, KnM, KnS, T, C, D, V, Z	SMOV, SMOV P: 11 steps

**Operation 1:**

This instruction copies a specified number of Digits from a 4 digit decimal source (S) and places them at a specified location within a destination (D)

number (also a 4 digit decimal). The existing data in the destination is overwritten.

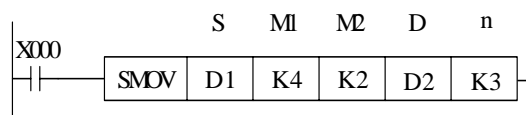
m1 - The source position of the 1st digit to be moved

m2 - The number of source digits to be moved

n- The destination position for the first digit

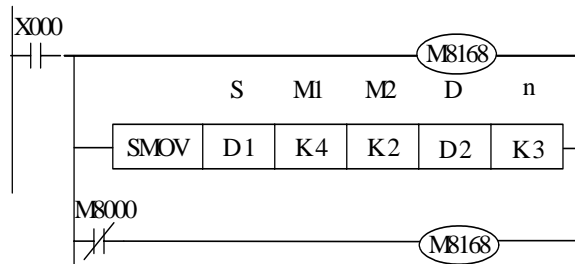
Note: The selected destination must NOT be smaller than the quantity of source data.

Digit positions are referenced by number: 1= units, 2= tens, 3= hundreds, 4=thousands.



**Operation 2:**

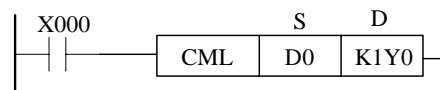
This modification of the SMOV operation allows BCD numbers to be manipulated in exactly the same way as the 'normal' SMOV manipulates decimal numbers, i.e. This instruction copies a specified number of digits from a 4 digit BCD source (S) and places them at a specified location within a destination (D) number (also a 4 digit BCD number). To select the BCD mode the SMOV instruction is coupled with special M coil M8168 which is driven ON. Please remember that this is a 'mode' setting operation and will be active, i.e. all SMOV instructions will operate in BCD format until the mode is reset, i.e. M8168 is forced OFF.

**4.2.5 CML (FNC 14)**

Mnemonic	Function	Operands		Program steps
		S	D	
CML FNC 14 (Compliment)	Copies and inverts the source bit pattern to a specified destination	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D, V, Z	CML, CMLP: 5 steps DCML, DCMLP: 9 steps

**Operation:**

A copy of each data bit within the source device (S) is inverted and then moved to a designated destination (D).



This means each occurrence of a '1' in the source data will become a '0' in the destination data while each source digit which is '0' will become a '1'. If the destination area is smaller than the source data then only the directly mapping bit devices will be processed.

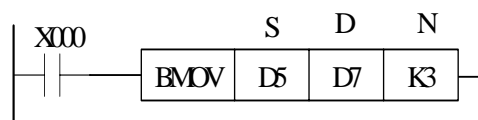


## 4.2.6 BMOV (FNC 15)

Mnemonic	Function	Operands			Program steps
		S	D	N	
BMOV FNC 15 (Block move)	Copies a specified block of multiple data elements to a new destination	KnX, KnY, KnM, KnS, T, C, D, V, Z File registers,	KnY, KnM, KnS, T, C, D, V, Z File registers	K, H, D  Note: N 512	BMOV, BMOVP: 7 steps

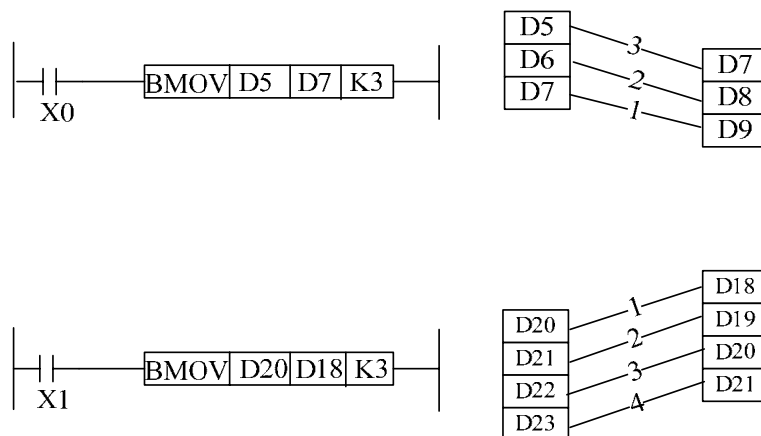
**Operation:**

A quantity of consecutively occurring data elements can be copied to a new destination. The source data is identified as a device head address (S) and a quantity of consecutive data elements (n). This is moved to the destination device (D) for the same number of elements (n).

**Points to note:**

- If the quantity of source devices (n) exceeds the actual number of available source devices, then only those devices which fall in the available range will be used.
- If the number of source devices exceeds the available space at the destination location, then only the available destination devices will be written to.
- The BMOV instruction has a built in automatic feature to prevent overwriting errors from occurring when the source (S - n) and destination (D -n) data ranges coincide. This is clearly identified in the following diagram:

(Note: The numbered arrows indicate the order in which the BMOV is processed)

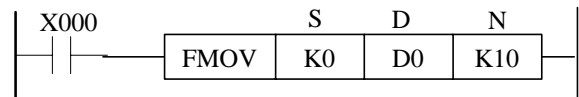


**4.2.7 FMOV (FNC 16)**

Mnemonic	Function	Operands			Program steps
		S	D	N	
FMOV FNC 16 (Fill move)	Copies a single data device to a range of destination devices	KnX, KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D, V, Z	K, H  Note: N≤512	FMOV, FMOV P: 7 steps DFMOV, DFMOV P: 13 steps

**Operation:**

The data stored in the source device (S) is copied to every device within the destination range. The range is specified by a device head address (D) and a quantity of consecutive



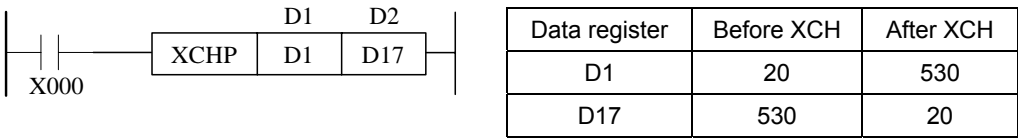
elements (n). If the specified number of destination devices (n) exceeds the available space at the destination location, then only the available destination devices will be written to.

Note: This instruction has a special programming technique which allows it to mimic the operation of newer applied instructions when used with older programming tools.

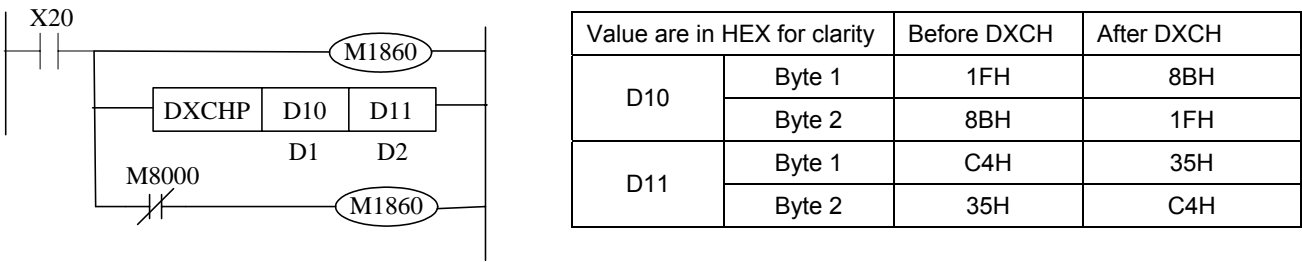
4.2.8 XCH (FNC 17)

Mnemonic	Function	Operands		Program steps
		D1	D2	
XCH FNC 17 (Exchange)	Data in the designated devices is exchanged	KnY, KnM, KnS, T, C, D, V, Z Note: when using the byte XCH (i.e.M8160 is ON) D1 and D2 must be the same device otherwise a program error will occur and M8067 will be turned ON.		XCH,XCHP: 5 steps DXCH,DXCHP: 9 steps

**Operation 1:** The contents of the two destination devices D1 and D2 are swapped, i.e. the complete word devices are exchanged. Ex.



**Operation 2:** This function is equivalent to FNC 147 SWAP. The bytes within each word of the designated devices D1 are exchanged when 'byte mode flag' M8160 is ON. Please note that the mode will remain active until it is reset, i.e. M8160 is forced OFF. Ex.



**4.2.9 BCD (FNC18)**

Mnemonic	Function	Operands		Program steps
		S	D	
BCD FNC 18 (Binary coded decimal)	Converts binary numbers to BCD equivalents / Converts floating point data to scientific format	KnX,KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D, V, Z	BCD, BCDP: 5 steps
		When using M8023 to convert data to scientific format, only double word (32 bit) data registers (D) may be used.		DBCD, DBCDP: 9 steps

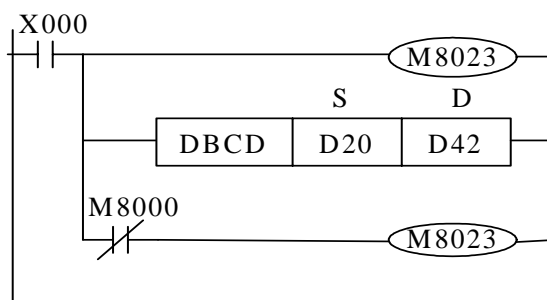
**Operation 1:**

The binary source data (S) is converted into an equivalent BCD number and stored at the destination device (D). If the converted

BCD number exceeds the operational ranges of 0 to 9,999 (16 bit operation) and 0 to 99,999,999 (32 bit operation) an error will occur. This instruction can be used to output data directly to a seven segment display.

**Operation 2:**

This function is equivalent to FNC 118 EBCD Data(S) is converted from 'floating point' format to 'scientific format' (D). This instruction requires double word (32 bit) operation and data registers as devices (S) and (D) to operate correctly.

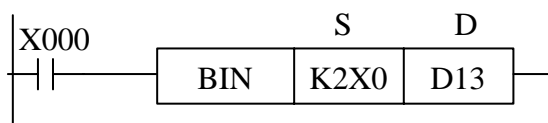


**4.2.10 BIN (FNC 19)**

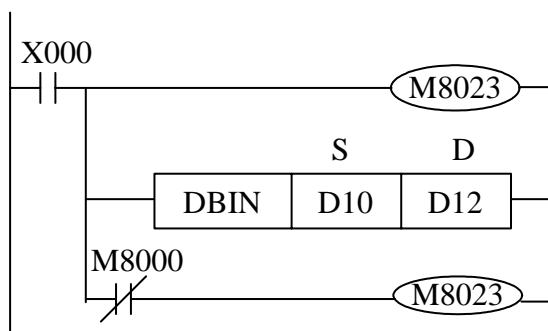
Mnemonic	Function	Operands		Program steps
		S	D	
BIN FNC 19 (Binary)	Converts BCD numbers to their binary equivalent /Converts scientific format data to floating point format	KnX,KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D, V, Z	BIN, BINP: 5 steps DBIN,DBINP: 9 steps
		When using M8023 to convert data to floating point format, only double word (32 bit) data registers (D) may be used.		

**Operation 1:**

The BCD source data (S) is converted into an equivalent binary number and stored at the destination device (D). If the source data is not provided in a BCD format an error will occur. This instruction can be used to read in data directly from thumbwheel switches.

**Operation 2:**

This function is equivalent to FNC 119 EBIN. Data (S) is converted from 'scientific format' to 'floating point' format (D). This instruction requires double word (32 bit) operation and data registers as devices (S) and (D) to operate correctly.



### 4.3 Arithmetic And Logical Operations - Functions 20 to 29

#### Contents:

ADD -	Addition	FNC 20
SUB -	Subtraction	FNC 21
MUL -	Multiplication	FNC 22
DIV -	Division	FNC 23
INC -	Increment	FNC 24
DEC -	Decrement	FNC 25
WAND -	Word AND	FNC 26
WOR -	Word OR	FNC 27
WXOR -	Word Exclusive OR	FNC 28
NEG -	Negation	FNC 29

#### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D1, S3 or for lists/abled devices D3+0, S+9 etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e.

positive = 0, and negative = 1.

LSB - Least Significant Bit.

#### Instruction modifications:

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
- P - A 16 bit mode instruction modified to use pulse (single) operation.
- D - An instruction modified to operate in 32 bit operation.
- D P- A 32 bit mode instruction modified to use pulse (single) operation.
- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.
- An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

**4.3.1 ADD (FNC 20)**

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
ADD FNC 20 (Addition)	The value of the two source devices is added and the result stored in the destination device	K, H, KnX, KnY, KnM, KnS,T, C, D, V, Z		KnY, KnM, KnS, T, C, D, V, Z	ADD, ADDP: 7 steps DADD,DADDP: 13 steps
		When using M8023 to add floating point data,only double word (32 bit) data registers (D) or constants (K/H) may be used.			

**Operation 1:**

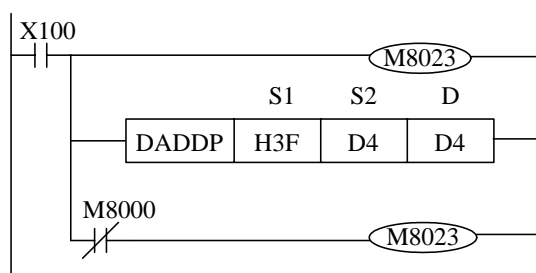
The data contained within the source devices (S1, S2) is combined and the total is stored at the specified destination device (D).

**Points to note:**

- All calculations are algebraically processed, i.e.  $5 + (-8) = -3$ .
- The same device may be used as a source (S1 or S2) and as the destination (D). If this is the case then the ADD instruction would actually operate continuously. This means on every scan the instruction would add the result of the last scan to the second source device. To prevent this from happening the pulse modifier should be used or an interlock should be programmed.
- If the result of a calculation is "0" then a special auxiliary flag, M8020 is set ON.
- If the result of an operation exceeds 32,767 (16 bit limit) or 2,147,483,647 (32 bit limit) the carry flag, M8022 is set ON. If the result of an operation exceeds -32,768 or -2,147,483,648 the borrow flag, M8021 is set ON. When a result exceeds either of the number limits, the appropriate flag is set ON (M8021 or M8022) and a portion of the carry/borrow is stored in the destination device. The mathematical sign of this stored data is reflective of the number limit which has been exceeded, i.e. when -32,768 is exceeded negative numbers are stored in the destination device but if 32,767 was exceeded positive numbers would be stored at D.
- If the destination location is smaller than the obtained result, then only the portion of the result which directly maps to the destination area will be written, i.e. if 25 (decimal) was the result, and it was to be stored at K1Y4 then only Y4 and Y7 would be active. In binary terms this is equivalent to a decimal value of 9 a long way short of the real result of 25!

**Operation 2:**

This function is equivalent to FNC 120 EADD. When 'floating point mode flag' M8023 is active, i.e. ON, DADD and DADDP instructions can be used to perform floating point additions.



When M8023 is reset, i.e. OFF floating point manipulation will not occur. Constants (K/H) and floating point numbers (stored in double data registers D) can be added in any configuration. The constants (K/H) will automatically be converted to the 'floating point format' for the addition operation. Answers for an operation can only be stored in double (32 bit) data registers. Items a) and b) above are also valid for this operating mode.

**Note:** The appropriate dedicated floating point instruction should be used instead E.g. Instead of DADD with M8023 ON, use FNC 120, DEADD.

#### 4.3.2 SUB (FNC 21)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
SUB FNC 21 (Subtract)	One source device is Subtracted from the other-the result is stored in the destination device	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D, V, Z		SUB, SUBP: 7 steps
				When using M8023 to Subtract floating point data, only double word (32 bit) data registers (D) or constants (K/H) may be used.	DSUB, SUBP: 13 steps

**Operation 1:** The data contained within the source device, S<sub>2</sub> is subtracted from the contents of source device S<sub>1</sub>. The result or remainder of this calculation is stored in the destination device D. Note: the 'Points to note', under the ADD instruction (previous page) can also be similarly applied to the subtract instruction.



**Operation 2:** This function is equivalent to FNC 121 ESUB. The information regarding 'Operation2:' of the ADD instruction apply similarly to this second operation of the SUB instruction (with the exception of a subtraction being performed instead of an addition). Again, only constants and double data words can be manipulated with only DSUB, DSUBP instruction formats being valid.

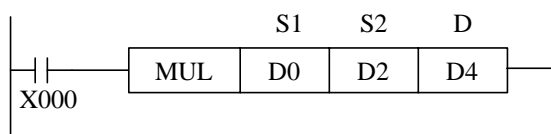


## 4.3.3 MUL (FNC 22)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
MUL FNC 22 (Multiplication)	Multiplies the two source devices together the result is stored in the destination device	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D, Z(V) Note: Z(V) may <b>NOT</b> be used for 32 bit operation	When using M8023 to subtract floating point data, only double word (32 bit) data registers (D) or constants (K/H) may be used.	MUL, MULP: 7 steps DMUL, DMULP: 13 steps

**Operation 1:**

The contents of the two source devices (S1, S2) are multiplied together and the result is stored at the destination device (D). Note the normal rules of algebra apply.

**Points to note:**

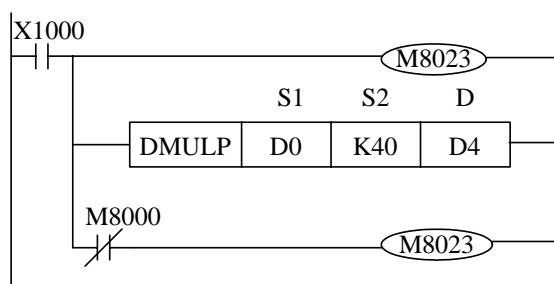
- When operating the MUL instruction in 16bit mode, two 16 bit data sources are multiplied together. They produce a 32 bit result. The device identified as the destination address is the lower of the two devices used to store the 32 bit result. Using the above example with some test data:  
 $5 (D0) \times 7 (D2) = 35$  - The value 35 is stored in (D4, D5) as a single 32 bit word.
- When operating the MUL instruction in 32 bit mode, two 32 bit data sources are multiplied together. They produce a 64 bit result. The device identified as the destination address is the lower of the four devices used to store the 64 bit result.
- If the location of the destination device is smaller than the obtained result, then only the portion of the result which directly maps to the destination area will be written, i.e if a result of 72 (decimal) is to be stored at K1Y4 then only Y7 would be active. In binary terms this is equivalent to a decimal value of 8, a long way short of the real result of 72!

**Operation 2:**

This function is equivalent to FNC 122 EMUL. When 'floating point mode flag' M8023 is active, i.e. ON, DMUL and DMULP instructions can be used to perform floating point multiplications.

When M8023 is reset, i.e. OFF floating point manipulation will not occur. Constants (K/H) and

floating point numbers (stored in double data registers D) can be used in any configuration. The constants (K/H) will automatically be converted to the 'floating point format' for the operation. Answers for an operation are stored (completely) in one pair of double (32 bits) data registers and not 2 pairs (64 bits) as used in 'Operation 1:'. The normal rules of algebra apply to floating point multiplication.



## 4.3.4 DIV (FNC 23)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
DIV FNC 23 (Division)	Divides one source value by another the result is stored in the destination device	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z		KnY,KnM,KnS, T, C, D, Z(V) Note: Z(V) may <b>NOT</b> be used for 32 bit operation	DIV,DIVP: 7steps
				When using M8023 to subtract floating point data, only double word (32 bit) data registers (D) or constants (K/H) may be used to perform	DDIV, DDIVP: 13 steps

**Operation 1:**

The primary source (S1) is divided by the secondary source (S2). The result is stored in the destination (D). Note the normal rules of algebra apply.

**Points to note:**

- When operating the DIV instruction in 16bit mode, two 16 bit data sources are divided into each other. They produce two 16 bit results. The device identified as the destination address is the lower of the two devices used to store the these results.  
This storage device will actually contain a record of the number of whole times S2 will divide into S1 (the quotient).  
The second, following destination register contains the remained left after the last whole division (the remainder). Using the previous example with some test data:  
 $51 (D0) \div 10 (D2) = 5(D4) 1(D5)$   
This result is interpreted as 5 whole divisions with 1 left over ( $5 \times 10 + 1 = 51$ ).
- When operating the DIV instruction in 32 bit mode, two 32 bit data sources are divided into each other. They produce two 32 bit results. The device identified as the destination address is the lower of the two devices used to store the quotient and the following two devices are used to store the remainder, i.e. if D30 was selected as the destination of 32 bit division operation then D30, D31 would store the quotient and D32, D33 would store the remainder. If the location of the destination device is smaller than the obtained result, then only the portion of the result which directly maps to the destination area will be written. If bit devices are used as the destination area, no remainder value is calculated.
- If the value of the source device S2 is 0 (zero) then an operation error is executed and the operation of the DIV instruction is cancelled.

**Operation 2:**

This function is equivalent to FNC 123 EDIV. The information regarding 'Operation2:' of the MUL instruction apply similarly to this second operation of the DIV instruction (with the exception of a division being performed instead of a multiplication). Again, only constants and double data words can be manipulated with only DDIV, DDIVP instruction formats being valid. Answers for an operation are stored (completely) in one pair of double (32 bits) data registers, i.e. answers are not split in to quotient and remainder as in 'Operation 1:'. The normal rules of algebra apply to floating point division.

**4.3.5 INC (FNC 24)**

Mnemonic	Function	Operands	Program steps
		D	
INC FNC 24 (Increment)	The designated device is incremented by 1 on every execution of the instruction	KnY, KnM, KnS, T, C, D, V, Z Standard V,Z rules apply for 32 bit operation	INC,INCP: 3 steps DINC, DINCP: 5 steps

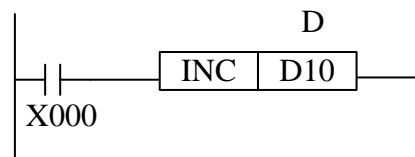
**Operation:**

On every execution of the instruction, the device specified as the destination D has its current value incremented (increased) by a value of 1.

In 16 bit operation, when +32,767 is reached, the next increment will write a value of -32,768 to the destination device.

In 32 bit operation, when +2,147,483,647 is reached the next increment will write a value of -2,147,483,648 to the destination device.

In both cases there is no additional flag to identify this change in the counted value.

**4.3.6 DEC (FNC 25)**

Mnemonic	Function	Operands	Program steps
		D	
DEC FNC 25 (Decrement)	The designated device is decremented by 1 on every execution of the instruction	KnY, KnM, KnS, T, C, D, V, Z Standard V,Z rules apply for 32 bit operation	DEC,DECP: 3 steps DDEC,DDECP: 5 steps

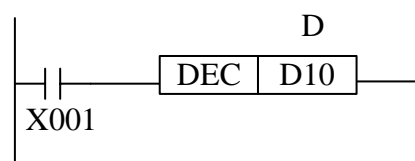
**Operation:**

On every execution of the instruction, the device specified as the destination D has its current value decremented (decreased) by a value of 1.

In 16 bit operation, when -32,768 is reached the next increment will write a value of +32,767 to the destination device.

In 32 bit operation, when -2,147,483,648 is reached the next increment will write a value of +2,147,483,647 to the destination device.

In both cases there is no additional flag to identify this change in the counted value.



**4.3.7 WAND (FNC 26)**

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
WAND FNC 26 (Logical word AND)	A logical AND is performed on the source devices result stored at destination	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z		KnY, KnM, KnS, T, C, D, V, Z	WAND, WANDP: 7 steps DAND, DANDP: 13 steps

**Operation:**

The bit patterns of the two source devices are analyzed (the contents of S2 is compared against the contents of S1). The

result of the logical AND analysis is stored in the destination device (D).

The following rules are used to determine the result of a logical AND operation. This takes place for every bit contained within the source devices:

General rule: (S1) Bit n WAND (S2) Bit n = (D) Bit n

1 WAND 1 = 1      0 WAND 1 = 0

1 WAND 0 = 0      0 WAND 0 = 0

**4.3.8 WOR (FNC 27)**

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
WOR FNC 27 (Logical word OR)	A logical OR is performed on the source devices result stored at destination	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z		KnY, KnM, KnS, T, C, D, V, Z	WOR, WOPR: 7 steps DOR, DORP: 13 steps

**Operation:**

The bit patterns of the two source devices are analyzed (the contents of S2 is compared against the contents of S1). The result of the

logical OR analysis is stored in the destination device (D).

The following rules are used to determine the result of a logical OR operation. This takes place for every bit contained within the source devices:

General rule: (S1) Bit n WOR (S2) Bit n = (D) Bit n

1 WOR 1 = 1      0 WOR 1 = 1

1 WOR 0 = 1      0 WOR 0 = 0



**4.3.9 WXOR (FNC 28)**

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
WXOR FNC 28 (Logical exclusive OR)	A logical XOR is performed on the source devices result stored at destination	K,H,KnX,KnY, KnM, KnS, T, C, D, V, Z		KnY, KnM, KnS, T, C, D, V, Z	WXOR, WXORP: 7 steps DXOR, DXORP: 13 steps

**Operation:**

The bit patterns of the two source devices are analyzed (the contents of S2 is compared against the contents of S1). The

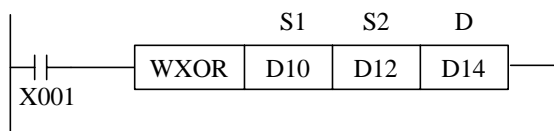
result of the logical XOR analysis is stored in the destination device (D).

The following rules are used to determine the result of a logical XOR operation. This takes place for every bit contained within the source devices:

General rule:  $(S1)Bit_n \text{ WXOR } (S2)Bit_n = (D)Bit_n$

1 WXOR 1 = 0      0 WXOR 1 = 1

1 WXOR 0 = 1      0 WXOR 0 = 0

**4.3.10 NEG (FNC 29)**

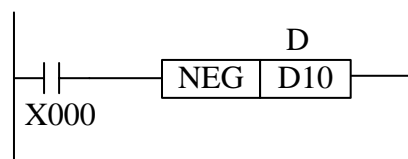
Mnemonic	Function	Operands			Program steps
		S1	S2	D	
NEG FNC 29 (Negation)	Logically inverts the contents of the designated device	K,H, KnX,KnY, KnM, KnS, T, C, D, V, Z		KnY, KnM, KnS, T, C, D, V, Z	NEG,NEGP: 3 steps DNEG, DNEGP: 5 steps

**Operation:**

The bit pattern of the selected device is inverted.

This means any occurrence of a '1' becomes a '0' and any occurrence of a '0' will be written as a '1'.

When this is complete, a further binary 1 is added to the bit pattern. The result is the total logical sign change of the selected devices contents, e.g. a positive number will become a negative number or a negative number will become a positive.



#### 4.4 Rotation And Shift - Functions 30 to 39

##### Contents:

ROR -	Rotation Right	FNC 30
ROL -	Rotation Left	FNC 31
RCR -	Rotation Right with Carry	FNC 32
RCL -	Rotation Left with Carry	FNC 33
SFTR -	(Bit) Shift Right	FNC 34
SFTL -	(Bit) Shift Left	FNC 35
WSFR -	Word Shift Right	FNC 36
WSFL -	Word Shift Left	FNC 37
SFWR -	Shift Register Write	FNC 38
SFRD -	Shift Register Read	FNC 39

##### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D<sub>1</sub>, S<sub>3</sub> or for lists/tables devices D<sub>3+0</sub>, S<sub>+9</sub> etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

##### Instruction modifications:

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
- P - A 16 bit mode instruction modified to use pulse (single) operation.
- D - An instruction modified to operate in 32 bit operation.
- D P - A 32 bit mode instruction modified to use pulse (single) operation.
- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.
- An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

4.4.1 ROR (FNC 30)

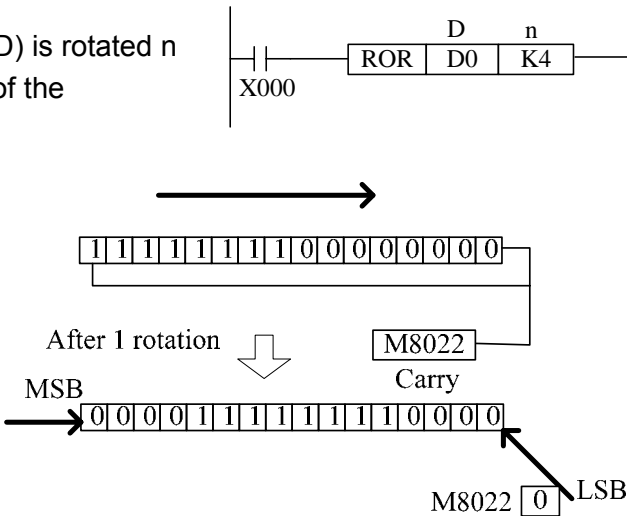
Mnemonic	Function	Operands		Program steps
		D	n	
ROR FNC 30 (Rotation right)	The bit pattern of the destination device is rotated 'n' places to the right on every execution	KnY, KnM, KnS, T, C, D, V, Z Note: 16 bit operation Kn=K4, 32 bit operation Kn=K8	K, H,  Note: 16 bit operation n 16 32 bit operation n 32	ROR, RORP: 5 steps  DROR, DRORP: 9 steps

Operation:

The bit pattern of the destination device (D) is rotated n bit places to the right on every operation of the instruction.

The status of the last bit rotated is copied to the carry flag M8022.

The example shown left is based on the instruction noted above it, where the bit pattern represents the contents of D0.



4.4.2 ROL (FNC 31)

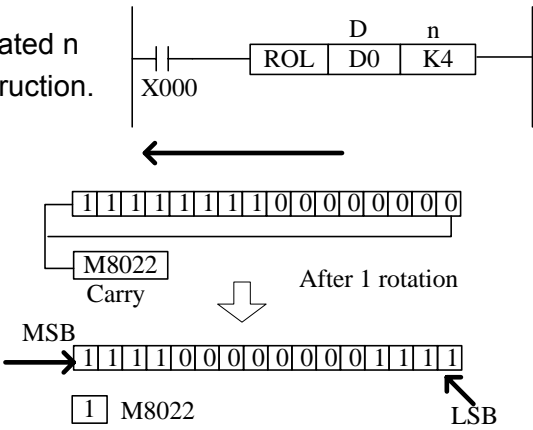
Mnemonic	Function	Operands		Program steps
		S	D	
ROL FNC 31 (Rotation left)	The bit pattern of the destination device is rotated 'n' places to the left on every execution	KnY, KnM, KnS, T, C, D, V, Z Note: 16 bit operation Kn= K4, 32 bit operation Kn= K8	K, H, Note: 16 bit operation n≤ 16 32 bit operation n≤ 32	ROL, ROLP: 5 steps  DROL, DROLP: 9 steps

Operation:

The bit pattern of the destination device (D) is rotated n bit places to the left on every operation of the instruction.

The status of the last bit rotated is copied to the carry flag M8022.

The example shown left is based on the instruction noted above it, where the bit pattern represents the contents of D0.





4.4.3 RCR (FNC 32)

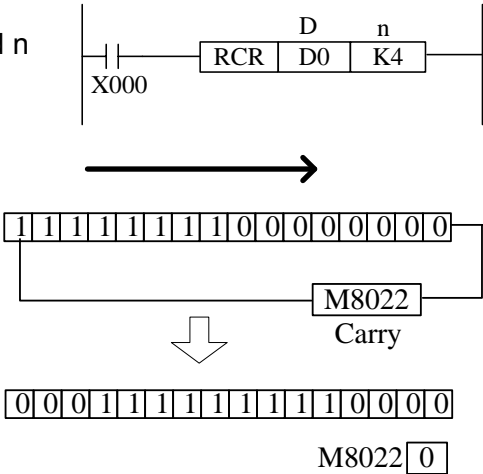
Mnemonic	Function	Operands		Program steps
		D	n	
RCR FNC 32 (Rotation right with carry)	The contents of the destination device are rotated right with 1 bit extracted to the carry flag	KnY, KnM, KnS, T, C, D, V, Z Note: 16 bit operation Kn= K4, 32 bit operation Kn= K8	K, H, Note: 16 bit operation n≤ 16 32 bit operation n≤ 32	RCR,RCRP: 5 steps DRCR, DRCRP: 9 steps

Operation:

The bit pattern of the destination device (D) is rotated n bit places to the right on every operation of the instruction.

The status of the last bit rotated is moved into the carry flag M8022. On the following operation of the instruction M8022 is the first bit to be moved back into the destination device.

The example shown left is based on the instruction noted above it, where the bit pattern represents the contents of D0.



4.4.4 RCL (FNC 33)

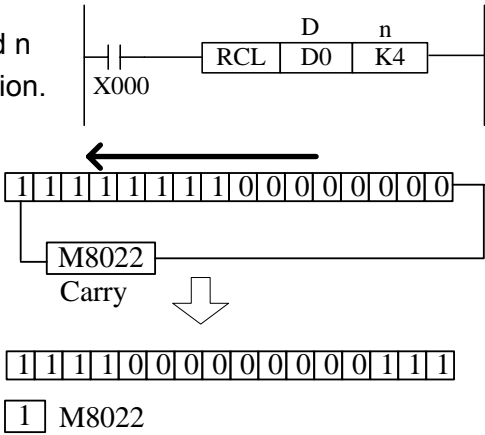
Mnemonic	Function	Operands		Program steps
		S	D	
RCL FNC 33 (Rotation left with carry)	The contents of the destination device are rotated left with 1 bit extracted to the carry flag	KnY, KnM, KnS, T, C, D, V, Z Note: 16 bit operation Kn= K4, 32 bit operation Kn= K8	K, H, Note: 16 bit operation n≤16 32 bit operation n≤32	RCL, RCLP: 5 steps DRCL, DRCLP: 9 steps

Operation:

The bit pattern of the destination device (D) is rotated n bit places to the left on every operation of the instruction.

The status of the last bit rotated is moved into the carry flag M8022. On the following operation of the instruction M8022 is the first bit to be moved back into the destination device.

The example shown left is based on the instruction noted above it, where the bit pattern represents the contents of D0.

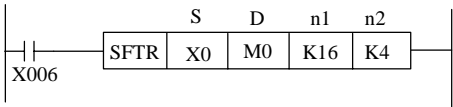


4.4.5 SFTR (FNC 34)

Mnemonic	Function	Operands				Program steps
		S	D	n1	n2	
SFTR FNC 34 (Bit shift right)	The status of the source devices are copied to a controlled bit stack moving the existing data to the right	X, Y, M, S	Y, M, S	K, H,  Note: n2 n1 . 1024		SFTR,SFTRP: 9 steps

Operation:

The instruction copies n2 source devices to a bit stack of length n1. For every new addition of n2 bits, the existing data within the bit stack is shifted n2 bits to the right. Any bit data moving to a position exceeding the n1 limit is diverted to an overflow area. The bit shifting operation will occur every time the instruction is processed unless it is modified with either the pulse suffix or a controlled interlock.



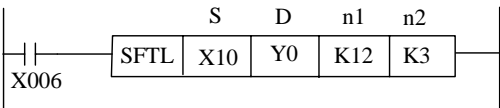
4.4.6 SFTL (FNC 35)

Mnemonic	Function	Operands				Program steps
		S	D	N1	N2	
SFTL FNC 35 (Bit shift left)	The status of the source devices are copied to a controlled bit stack moving the existing data to the left	X, Y, M, S	Y, M, S	K, H,  Note: n2 n1 . 1024		SFTL,SFTLP: 9 steps

Operation:

The instruction copies n2 source devices to a bit stack of length n1. For every new addition of n2 bits, the existing data within the bit stack is shifted n2 bits to the left. Any bit data moving to a position exceeding the n1 limit is diverted to an overflow area.

The bit shifting operation will occur every time the instruction is processed unless it is modified with either the pulse suffix or a controlled interlock.



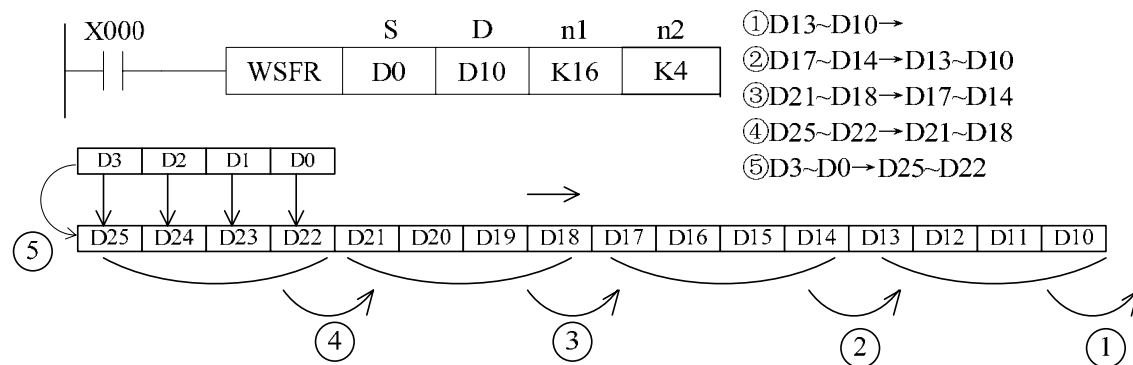
## 4.4.7 WSFR (FNC 36)

Mnemonic	Function	Operands				Program steps
		S	D	N1	N2	
WSFR FNC 36 (Word shift right)	The value of the source devices are copied to a controlled word stack moving the existing data to the right	KnX, KnY, KnM,KnS, T, C, D	KnY, KnM,KnS T, C, D	K, H,  Note: $n2 \leq n1 \leq 512$		SFTR,SFTRP: 9 steps

**Operation:**

The instruction copies  $n2$  source devices to a word stack of length  $n1$ . For each addition of  $n2$  words, the existing data within the word stack is shifted  $n2$  words to the right. Any word data moving to a position exceeding the  $n1$  limit is diverted to an overflow area.

The word shifting operation will occur every time the instruction is processed unless it is modified with either the pulse suffix or a controlled interlock.



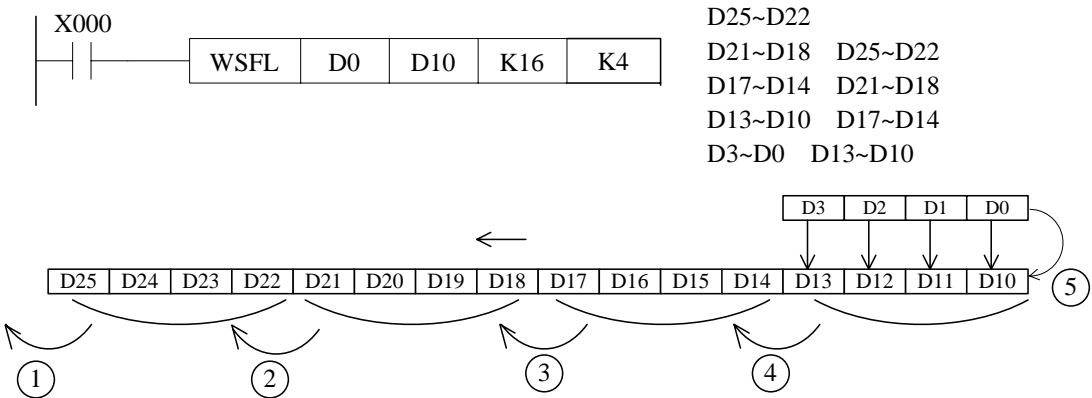
**Note:** when using bit devices as source (S) and destination (D) the Kn value must be equal.

4.4.8 WSFL (FNC 37)

Mnemonic	Function	Operands				Program steps
		S	D	N1	N2	
WSFL FNC 37 (Word shift left)	The value of the source devices are copied to a controlled word stack moving the existing data to the left	KnX, KnY, KnM,KnS, T, C, D	KnY,KnM, KnS, T, C, D	K, H,  Note: n2 n1 . 512		WSFL, WSFLP: 9 steps

Operation:

The instruction copies n2 source devices to a word stack of length n1. For each addition of n2 words, the existing data within the word stack is shifted n2 words to the left. Any word data moving to a position exceeding the n1 limit is diverted to an overflow area. The word shifting operation will occur every time the instruction is processed unless it is modified with either the pulse suffix or a controlled interlock.



**Note:** when using bit devices as source (S) and destination (D) the Kn value must be equal.

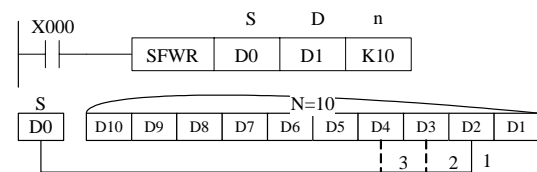
## 4.4.9 SFWR (FNC 38)

Mnemonic	Function	Operands			Program steps
		S	D	N	
SFWR FNC 38 (Shift register write)	This instruction creates and builds a FIFO stack n devices long –must be used with SFRD FNC 39	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D,	K, H,  Note: $2^{n-1} \leq 512$	SFWR, SFWRP: 7 steps

**Operation:**

The contents of the source device (S) are written to the FIFO stack. The position of insertion into the stack is automatically calculated by the PLC.

The destination device (D) is the head address of the FIFO stack. The contents of D identify where the next record will be stored (as an offset from D+1). If the contents of D exceed the value “n-1” (n is the length of the FIFO stack) then insertion into the FIFO stack is stopped. The carry flag M8022 is turned ON to identify this situation.

**Points to note:**

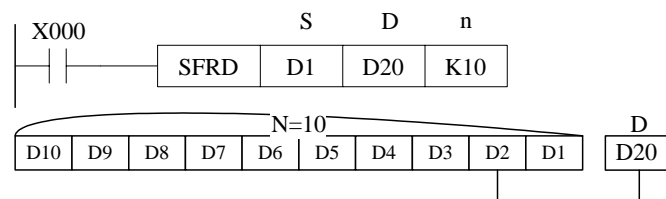
- FIFO is an abbreviation for 'First-In/ First-OUT'.
- Although n devices are assigned for the FIFO stack, only n-1 pieces of information may be written to that stack. This is because the head address device (D) takes the first available register to store the information regarding the next data insertion point into the FIFO stack.
- Before starting to use a FIFO stack ensure that the contents of the head address register (D) are equal to '0' (zero).
- This instruction should be used in conjunction with SFRD FNC 39. The n parameter in both instructions should be equal.

## 4.4.10 SFRD (FNC 39)

Mnemonic	Function	Operands			Program steps
		S	D	N	
SFRD FNC 39 (Shift register read)	This instruction reads and reduces FIFO stack- must be used with SFWR FNC 38	KnY, KnM, KnS, T, C, D,	KnY, KnM, KnS, T, C, D, V, Z	K, H,  Note: 2 n . 512	SFRD, SFRDP: 7 steps

**Operation:**

The source device (S) identifies the head address of the FIFO stack. Its contents reflect the last entry point of data on to the FIFO stack, i.e. where the end of the FIFO is (current position).



This instruction reads the first piece of data from the FIFO stack (register S+1), moves all of the data within the stack 'up' one position to fill the read area and decrements the contents of the FIFO head address (S) by 1. The read data is written to the destination device (D). When the contents of the source device (S) are equal to '0' (zero), i.e. the FIFO stack is empty, and the flag M8020 is turned ON.

**Points to note:**

- FIFO is an abbreviation for 'First-In/ First-OUT'.
- Only n-1 pieces of data may be read from a FIFO stack. This is because the stack requires that the first register, the head address (S) is used to contain information about the current length of the FIFO stack.
- This instruction will always read the source data from the register S+1.
- This instruction should be used in conjunction with SFWR FNC 38. The n parameter in both instructions should be equal.



## 4.5 Data Operation - Functions 40 to 49

### Contents:

ZRST -	Zone Reset	FNC 40
DECO -	Decode	FNC 41
ENCO -	Encode	FNC 42
SUM -	The Sum Of Active Bits	FNC 43
BON -	Check Specified Bit Status	FNC 44
MEAN -	Mean	FNC 45
ANS -	(Timed) Annunciator Set	FNC 46
ANR -	Annunciator Reset	FNC 47
SQR -	Square Root	FNC 48
FLT -	Float, (Floating Point)	FNC 49

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D<sub>1</sub>, S<sub>3</sub> or for lists/tables devices D<sub>3+0</sub>, S<sub>+9</sub> etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e.

positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
- P - A 16 bit mode instruction modified to use pulse (single) operation.
- D - An instruction modified to operate in 32 bit operation.
- D P - A 32 bit mode instruction modified to use pulse (single) operation.
- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.
- An operand which cannot be indexed, i.e. the addition of V or Z is either invalid or will have no effect to the value of the operand.

## 4.5.1 ZRST (FNC 40)

Mnemonic	Function	Operands		Program steps
		D1	D2	
ZRST FNC 40 (Zone Reset)	Used to reset a range of like devices in one operation	Y, M, S, T, C, D Note: D1 must be less than or equal ( $\leq$ ) to D2. Standard and High speed counters cannot be mixed		ZRST, ZRSTP: 5 steps

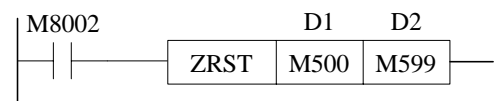
**Operation:**

The range of devices, inclusive of those specified as the two destinations are reset, i.e. for data devices the current value is set to 0 (zero) and for

bit elements, the devices are turned OFF, i.e. also set to 0 (zero).

The specified device range cannot contain mixed device types, i.e. C000 specified as the first destination device (D1) cannot be paired with T199 as the second destination device (D2). When resetting counters, standard and high speed counters cannot be reset as part of the same range.

If D1 is greater than ( $>$ ) D2 then only device D1 is reset.

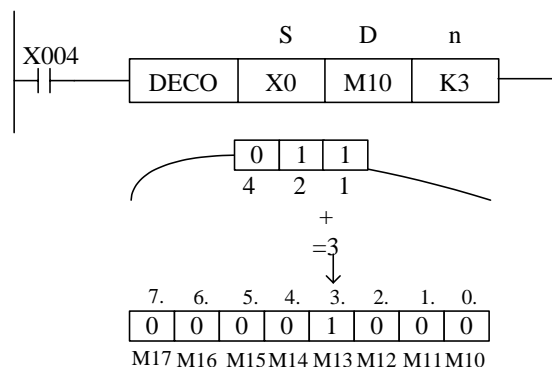


## 4.5.2 DECO (FNC 41)

Mnemonic	Function	Operands			Program steps
		S	D	N	
DECO FNC 41 (Decode)	Source data value Q identifies the Qth bit of the destination device which will be turned ON	K, H, X, Y, M, S, T, C, D, V, Z	Y, M, S, T, C, D	K, H, Note: D= Y,M,S then n range = 1-8 D= T,C,D then n range = 1-4 n= 0, then no processing	DECO, DECOP: 7 steps

**Operation:**

Source data is provided by a combination of operands S and n. Where S specifies the head address of the data and n, the number of consecutive bits. The source data is read as a single number (binary to decimal conversion) Q. The source number Q is the location of a bit within the destination device (D) which will be turned ON (see example opposite). When the destination device is a data device n must be within the range 1 to 4 as there are only 16 available destination bits in a single data word. All unused data bits within the word are set to 0.



## 4.5.3 ENCO (FNC 42)

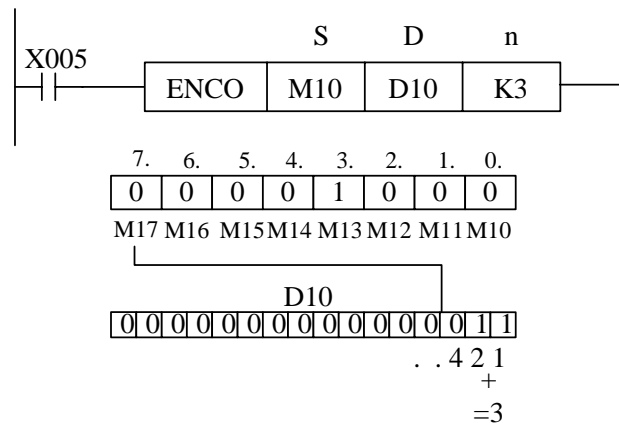
Mnemonic	Function	Operands			Program steps
		S	D	N	
ENCO FNC 42 (Encode)	Then location of the highest active bit is stored as a numerical position from the head address	X, Y, M, S, T, C, D, V, Z	T, C, D, V, Z	K, H, Note: S=X, Y, M, S then n range=1-8 S= T,C,D then n range = 1-4 n = 0, then no processing:	ENCO, ENCOP 7 steps

**Operation:**

The highest active bit within the readable range has its location noted as a numbered offset from the source head address (S). This is stored in the destination register (D).

**Points to note:**

- The readable range is defined by the largest number storable in a binary format within the number of destination storage bits specified by n, i.e. if n was equal to 4 bits a maximum number within the range 0 to 15 can be written to the destination device. Hence, if bit devices were being used as the source data, 16 bit devices would be used, i.e. the head bit device and 15 further, consecutive devices.
- If the stored destination number is 0 (zero) then the source head address bit is ON, i.e. the active bit has a 0 (zero) offset from the head address. However, if NO bits are ON within the source area, 0 (zero) is written to the destination device and an error is generated.
- When the source device is a data or word device n must be taken from the range 1 to 4 as there are only 16 source bits available within a single data word.



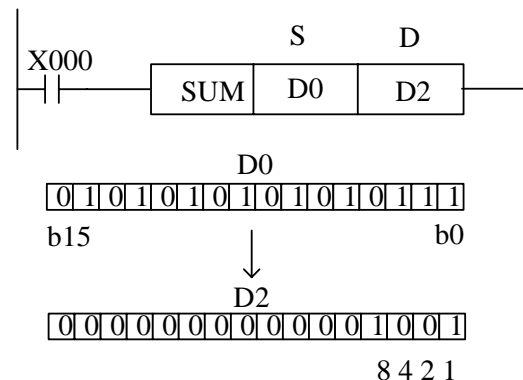
## 4.5.4 SUM (FNC 43)

Mnemonic	Function	Operands		Program steps
		S	D	
SUM FNC 43 (Sum of active bits)	The number (quantity) of active bits in the source data is stored in the destination device	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D, V, Z	SUM,SUMP: 7 steps DSUM,DSUMP: 9 steps

**Operation:**

The number of active (ON) bits within the source device (S), i.e. bits which have a value of "1" are counted. The count is stored in the destination register (D). If a double word format is used, both the source and destination devices use 32 bit, double registers. The destination device will always have its upper 16 bits set to 0 (zero) as the counted value can never be more than 32.

If no bits are ON then zero flag, M8020 is set.



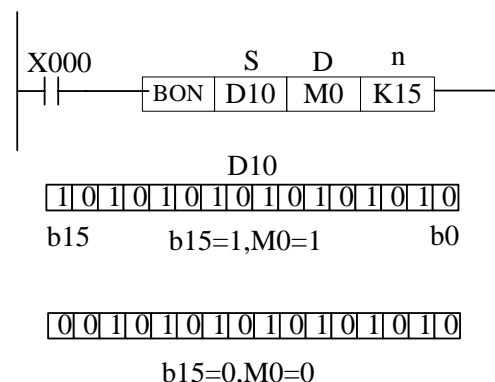
## 4.5.5 BON (FNC 44)

Mnemonic	Function	Operands			Program steps
		S	D	n	
BON FNC 44 (Check specified bit status)	The status of the specified bit in the source device is indicated at the destination	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z	Y, M, S	K,H, Note: 16 bit operation n=0 to 15 32 bit operation n=0 to 31	BON, BONP: 7steps DBONP,DBON: 13 steps

**Operation:**

A single bit position (n) is specified from within a source device/area (S). n could be regarded as a specified offset from the source head address (S), i.e. 0 (zero) being the first device (a 0 offset) where as an offset of 15 would actually be the 16th device. If the identified bit becomes active, i.e. ON, the destination device (D) is activated to "flag" the new status.

The destination device could be said to act as a mirror to the status of the selected bit source.



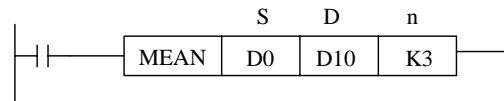
## 4.5.6 MEAN (FNC 45)

Mnemonic	Function	Operands			Program steps
		S	D	n	
MEAN FNC 45 (Mean)	Calculates the mean of a range of devices	KnX, KnY, KnM, KnS, T, C, D	KnY, KnM, KnS, T, C, D, V, Z	K,H, Note: n=1 to 64	MEAN,MEANP: 7 steps DMEAN,DMEANP: 13steps

**Operation:**

The range of source data is defined by operands Sand n. S is the head address of the source data and n specifies the number of consecutive source devices used.

The value of all the devices within the source range is summed and then divided by the number of devices summed, i.e. n. This generates an integer mean value which is stored in the destination device (D). The remainder of the calculated mean is ignored.

**General rule**

$$D = \frac{\sum_{S_0}^{S_n} S}{n} = \frac{(S_0 + S_1 + \dots + S_n)}{n}$$

**Example**

$$D10 = \frac{(D0 + D1 + D3)}{3}$$

**Points to note:**

If the source area specified is actually smaller than the physically available area, then only the available devices are used. The actual value of n used to calculate the mean will reflect the used, available devices. However, the value for n which was entered into the instruction will still be displayed. This can cause confusion as the mean value calculated manually using this original n value will be different from that which is displayed.

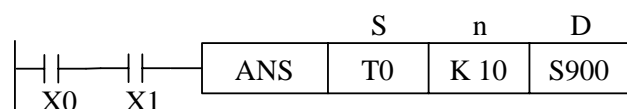
If the value of n is specified outside of the stated range (1 to 64) an error is generated.

## 4.5.7 ANS (FNC 46)

Mnemonic	Function	Operands			Program steps
		S	D	n	
ANS FNC 46 (Timed annunciator set)	This instruction starts a timer. Once timed out the selected annunciator flag is set ON	T Note: available range T0 to T199	S Note: annunciator range S900 to S999	K,H, Note : n range 1 to 32,767 – in units of 100msec	ANS: 7 steps

**Operation:**

This instruction, when energized, starts a timer (S) for n, 100 msec. When the timer completes its cycle the assigned annunciator (D) is set ON.



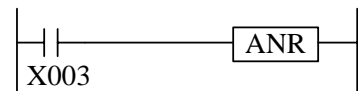
If the instruction is switched OFF during or after completion of the timing cycle the timer is automatically reset. However, the current status of the annunciator coil remains unchanged.

## 4.5.8 ANR (FNC 47)

Mnemonic	Function	Operands	Program steps
		D	
ANR FNC 47 (Annunciator reset)	The lowest active annunciator is reset on every operation of this instruction	N/A	ANR,ANRP: 1step

**Operation:**

Annunciators which have been activated are sequentially reset one-by-one, each time the ANR instruction is operated. If the ANR instruction is driven continuously it will carry out its resetting operation on every program scan unless it is modified by the pulse, P prefix or by a user defined program interlock.



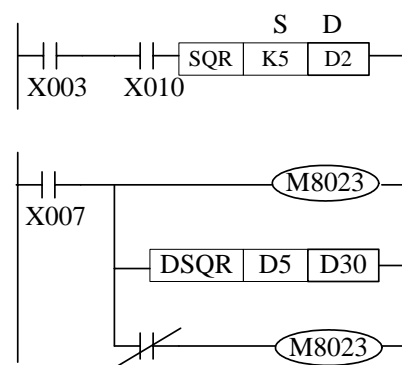
## 4.5.9 SQR (FC 48)

Mnemonic	Function	Operands		Program steps
		S	D	
SQR FNC 48 (Square root)	Performs a mathematical square root e.g $D = \sqrt{S}$	K,H,D	D	SQR, SQR P: 5 steps DSQR, DSQR P: 9 steps

**Operation1:**

This instruction performs a square root operation on source data (S) and stores the result at destination device (D). The operation is conducted entirely in whole integers rendering the square root answer rounded to the lowest whole number. For example, if (S) = 154, then (D) is calculated as being 12. M8020 is set ON when the square root operation result is equal to zero.

Answers with rounded values will activate M8021.

**General notes:**

Performing any square root operation (even on a calculator) on a negative number will result in an error. This will be identified by special M coil M8067 being activated:

$$\sqrt{-168} = \text{Error and M8067 will be set ON}$$

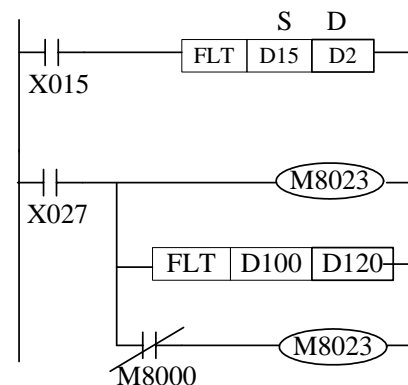
This is true for both operation modes.

## 4.5.10 FLT (FNC 49)

Mnemonic	Function	Operands		Program steps
		S	D	
FLT FNC 49 (Floating point)	Used to convert data to and from floating point format	D		FLT, FLTP: 5 steps DFLT,DFLTP: 9 steps

**Operation 1:**

When the float instruction is used without the float flag (M8023 = OFF) the source data (S) is converted in to an equivalent value stored in float format at the destination device (D). Please note that two consecutive devices (D and D+1) will be used to store the converted float number. This is true regardless of the size of the source data (S), i.e. whether (S) is a single device (16 bits) or a double device (32 bits) has no effect on the number of devices (D) used to store the floating point number. Examples:



Decimal source data (S)	Floating point destination value (D)
1	1
-26700	$-2.67 \times 10^4$
404	$4.04 \times 10^2$

**Operation 2:** This function is equivalent to FNC 129 INT.

When the float instruction is performed and the float flag M8023 is ON, the float operation will be conducted in reverse to Operation 1. Any floating point format number stored at source (S) will be converting to its decimal equivalent and stored at destination (D).

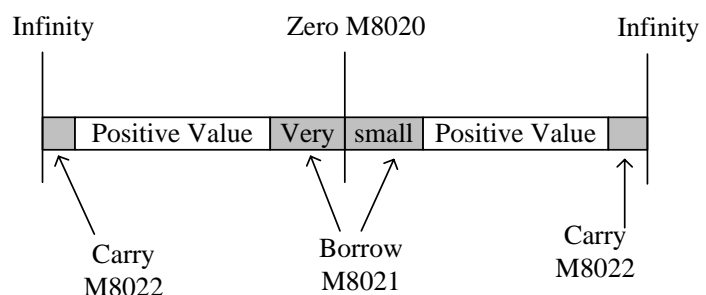
**Points to Note:**

- a) When floating point numbers are used the zero, borrow and carry flags (M8020, M8021 and M8022 respectively) operate at the following times; M8020, Zero: is activated when the result is Zero.

M8021, Borrow: is activated when the result is smaller than the smallest possible number.

The result is forced to equal the smallest number and the associated flag is set ON.

M8022, Carry: is activated when the result is larger than the largest possible number. The result is forced to equal the largest number and the associated flag is



set ON.

- b) Floating point numbers always occupy 32 consecutive bits, i.e. 2 consecutive data registers. When converting between float and decimal numbers please allow enough destination devices, i.e.

Instruction	Double word operation	Status of M8023	Number of source registers(S)	Number of destination registers(S)	Remark
FLT	NO	OFF	1(S)	2(D,D+1)	Convert to floating point
FLT(INT)		ON	2(S,S+1)	1(D)	Convert to decimal
DFLT	YES	OFF	2(S,S+1)	2(D,D+1)	Convert to floating point
DFLT(DINT)		ON	2(S,S+1)	2(D,D+1)	Convert to decimal



## 4.6 High Speed Processing - Functions 50 to 59

### Contents:

REF -	Refresh	FNC 50
MTR -	Input matrix	FNC 52
HSCS -	High speed counter set	FNC 53
HSCR -	High speed counter reset	FNC 54
HSZ -	High speed counter zone compare	FNC 55
SPD -	Speed detect	FNC 56
PLSY -	Pulse Y output	FNC 57
PWM -	Pulse width modulation	FNC 58
PLSR -	Ramp Pulse output	FNC 59

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D1, S3 or for lists/tables devices D3+0, S+9 etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
  - P - A 16 bit mode instruction modified to use pulse (single) operation.
  - D - An instruction modified to operate in 32 bit operation.
  - D P - A 32 bit mode instruction modified to use pulse (single) operation.
  - A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.
  - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.
-

## 4.6.1 REF (FNC 50)

Mnemonic	Function	Operands		Program steps
		D	n	
REF FNC 50 (Refresh)	Forces an immediate update of inputs or outputs as specified	X, Y , Note: D should always be a multiple of 10, i.e. 00, 10, 20, 30 etc.	K, H, Note: n should always be a multiple of 8, i.e. 8, 16, 24, 32 etc	REF, REFP: 5 steps

**Operation:**

Standard PLC operation processes output and input status between the END instruction of one program scan and step 0 of the following program scan. If an

immediate update of the I/O device status is required the REF instruction is used. The REF instruction can only be used to update or refresh blocks of 8 (n) consecutive devices. The head address of the refreshed devices should always have its last digit as a 0 (zero), i.e. in units of 10.

**Note:** A short delay will occur before the I/O device is physically updated, in the case of inputs a time equivalent to the filter setting, while outputs will delay for their set energized time

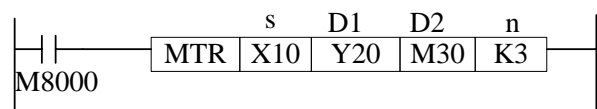


## 4.6.2 MTR (FNC 52)

Mnemonic	Function	Operands				Program steps
		S	D1	D2	n	
MTR FNC 52 (Input matrix)	Multiplexes a bank of inputs into a number of sets of devices. Can only be used ONCE	X	Y	Y,M,S	K,H	MTR: 9 steps
		Note: These operands should always be a multiple of 10, i.e. 00, 10, 20, 30 etc.			Note: n=2 to 8	

**Operation:**

This instruction allows a selection of 8 consecutive input devices (head address S) to be used multiple (n) times, i.e. each physical input has more than one, separate and quite different (D1) signal being processed. The result is stored in a matrix-table (head address D2).

**Points to note:**

- The MTR instruction involves high speed input/output switching. For this reason this instruction is only recommended for use with transistor output modules.
- For the MTR instruction to operate correctly, it must be driven continuously. It is recommended that special auxiliary relay M8000, the PLC RUN status flag, is used.

After the completion of the first full reading of the matrix, operation complete flag M8029 is turned ON. This flag is automatically reset when the MTR instruction is turned OFF.

- c) Each set of 8 input signals are grouped into a 'bank' (there are n number of banks).
- d) Each bank is triggered/selected by a dedicated output (head address D<sub>1</sub>). This means the quantity of outputs from D<sub>1</sub>, used to achieve the matrix are equal to the number of banks n.

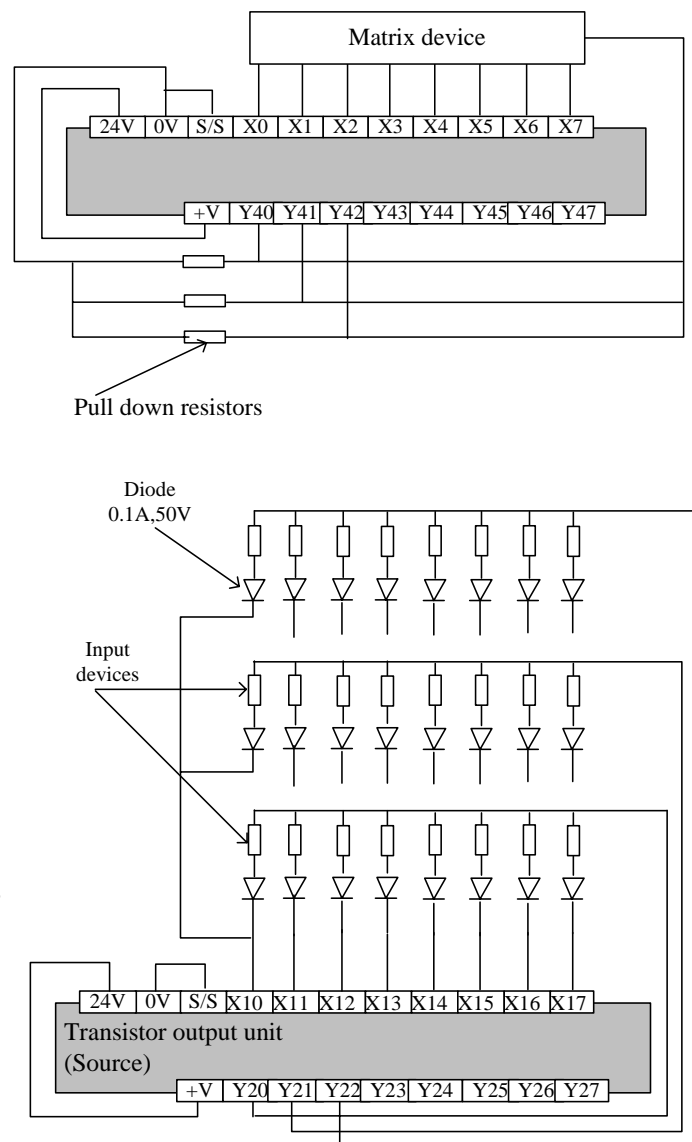
As there are now additional inputs entering the PLC these will each have a status which needs recording. This is stored in a matrix-table. The matrix-table starts at the head address D<sub>2</sub>. The matrix construction mimics the same 8 signal by n bank configuration. Hence, when a certain input in a selected bank is read, its status is stored in an equivalent position within the result matrix-table.

- e) The matrix instruction operates on an interrupt format, processing each bank of inputs every 20msec. This time is based on the selected input filters being set at 10msec. This would result in an 8 bank matrix, i.e. 64 inputs (8 inputs × 8 banks) being read in 160msec.

If high speed inputs (ex. X0) are specified for operand S, the reading time of each bank becomes only 10msec, i.e. a halving of the reading speed. However, additional pull down resistors are required on the drive outputs to ensure the high speed reading does not detect any residual currents from the last operation.

These should be placed in parallel to the input bank and should be of a value of approximately 3.3kΩ, 0.5W. For easier use, high speed inputs should not be specified at S.

- f) Because this instruction uses a series of multiplexed signals it requires a certain amount of 'hard wiring' to operate. The example wiring diagram to the right depicts the circuit used if the previous example instruction was programmed. As a general precaution to aid successful operation diodes should be placed after each input device



(see diagram opposite). These should have a rating of 0.1A, 50V.

g) Example Operation

When output Y20 is ON only those inputs in the first bank are read. These results are then stored; in this example, auxiliary coils M30 to M37. The second step involves Y20 going OFF and Y21 coming ON. This time only inputs in the second bank are read. These results are stored in devices M40 to M47. The last step of this example has Y21 going OFF and Y22 coming ON. This then allows all of the inputs in the third bank to be read and stored in devices M50 to M57. The processing of this instruction example would take  $20 \times 3 = 60\text{msec}$ .

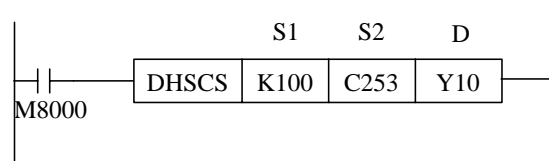
#### 4.6.3 HSCS (FNC 53)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
HSCS FNC 53 (High speed counter set)	Sets the selected output when the specified high speed counter value equals the test value	K, H, KnX, KnY, KnM, KnS, T, C, D, Z	C Note: C = 235 to 249, 251 to 254	Y, M, S Interrupt pointers I010 to I060 can be set	DHSCS: 13 steps

##### Operation:

The HSCS set, compares the current value of the selected high speed counter (S2) against a selected value (S1). When the counters current value changes to a value equal to S1 the device specified as the destination

(D) is set ON. The example above shows that Y10 would be set ON only when C253's value stepped from 99-100 OR 101-100. If the counters current value was forced to equal 100, output Y10 would **NOT** be set ON.



##### Points to note:

- It is recommended that the drive input used for the high speed counter functions; HSCS, HSCR, HSCZ is the special auxiliary RUN contact M8000.
- If more than one high speed counter function is used for a single counter the selected flag devices (D) should be kept within 1 group of 8 devices, i.e. Y0-7, M10-17.
- All high speed counter functions use an interrupt process, hence, all destination devices (D) are updated immediately.

##### Use of interrupt pointers

Can use interrupt pointers I010 through I060 (6 points) as destination devices (D). This enables interrupt routines to be triggered directly when the value of the specified high speed counter reaches the value in the HSCS instruction.

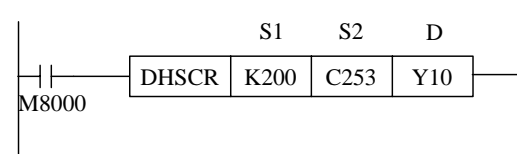
## 4.6.4 HSCR (FNC 54)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
HSCR FNC 54 (High speed counter reset)	Resets the selected output when the specified high speed counter equals the test value	K, H, KnX, KnY, KnM, KnS, T, C, D, Z	C Note: C = C235 to C249, C251 to C254	Y, M, S, C Note: If C, use same counter as S1	DHSCR: 13 steps

**Operation:**

The HSCR, compares the current value of the selected high speed counter (S2) against a selected value (S1). When the counters current value changes to a value equal to S1, the device specified as the destination (D) is reset. In the example above, Y10 would be reset only when C253's value stepped from 199 to 200 or from 201 to 200. If the current value of C253 was forced to equal 200 by test techniques, output Y10 would **NOT** reset.

For further, general points, about using high speed counter functions, please see the subsection 'Points to note' under the HSCS (FNC 53). Relevant points are; a, b, and c. Please also reference the note about the number of high speed instructions allowable.



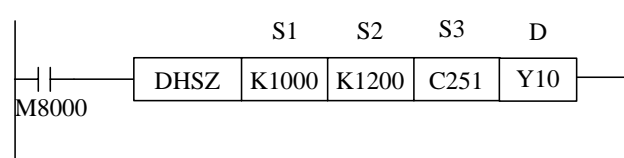
## 4.6.5 HSZ (FNC 55)

Mnemonic	Function	Operands				Program steps
		S1	S2	S3	n	
HSZ FNC 55 (High speed zone compare)	Operation 1: The current value of a high speed counter is checked against a specified range	K, H, KnX, KnY, KnM, KnS, T, C, D, Z  $S1 \leq S2$		C Note: C = 235 to 249, C251 to C253	Y, M, S Note: 3 consecutive devices are used	DHSZ: 17 steps

**Operation 1 - Standard:**

This instruction works in exactly the same way as the standard ZCP (FNC11). The only difference is that the device being compared is a high speed counter (specified as S3).

Also, all of the outputs (D) are updated immediately due to the interrupt operation of the DHSZ. It should be remembered that when a device is specified in operand D it is in fact a head address for 3 consecutive devices. Each one is used to represent the status of the current comparison, i.e. using the above example as a basis,



- Y10 (D) C251 is less than S<sub>1</sub>, K1000 (S<sub>3</sub> < S<sub>1</sub>)  
 Y11 (D+1) C251 is greater than or equal S<sub>1</sub>, K1000 but less than or equal S<sub>2</sub>, K1200  
 (S<sub>3</sub> S<sub>1</sub>, S<sub>3</sub> S<sub>2</sub>)  
 Y12 (D+2) C251 is greater than S<sub>2</sub>, K1200 (S<sub>3</sub> > S<sub>2</sub>)

For further, general points, about using high speed counter functions please see the subsection 'Points to note' under the HSCS (FNC 52). Relevant points are; a, b, and c. Please also reference the note about the number of high speed instructions allowable

#### 4.6.6 SPD (FNC 56)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
SPD FNC 56 (Speed detection)	Detects the number of 'encoder' pulses in a given time frame. Results can be used to calculate speed	X0 to X5	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z Unit is msec	T, C, D, Z (V) Note: 3 consecutive devices are used. In the case of D= Z monitor D8028, D8029 and D8030	SPD: 7 steps

##### Operation:

The number of pulses received at S<sub>1</sub> are counted and stored in D+1; this is the current count value.

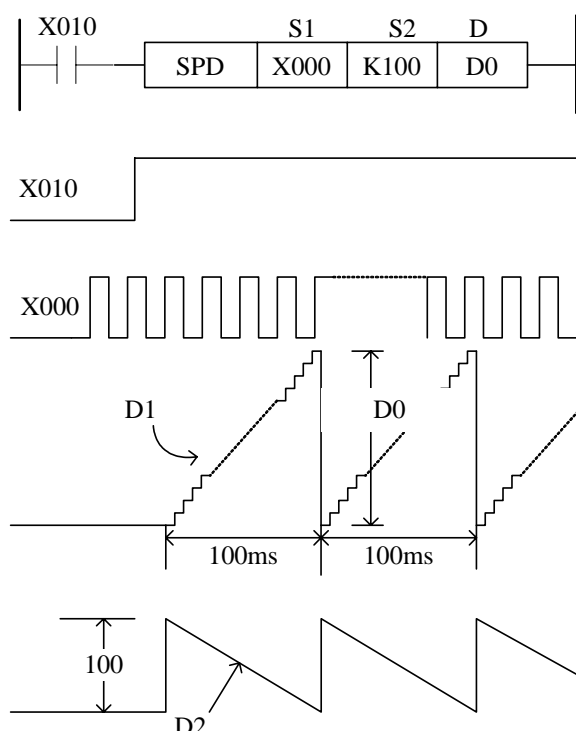
The counting takes place over a set time frame specified by S<sub>2</sub> in msec. The time remaining on the current 'timed count', is displayed in device D+2.

The numbers of counted pulses (of S<sub>1</sub>) from the last timed count are stored in D. The timing chart opposite shows the SPD operation in a graphical sense. Note:

Current count value, device D+1

Accumulated/ last count value, device D

Current time remaining in msec, device D+2



##### Points to note:

- When the timed count frame is completed the data stored in D+1 is immediately written to D. D+1 is then reset and a new time frame is started.
- Because this is both a high speed and an interrupt process only inputs X0 to X5 may be used as the source device S<sub>1</sub>. However, the specified device for S<sub>1</sub> must **NOT** coincide with any other high speed function which is operating, i.e. a high speed counter using the same input. The SPD instruction is considered to act as a single phase counter.

- c) Multiple SPD instructions may be used, but the identified source devices S1 restrict this to a maximum of 6 times.
- d) Once values for timed counts have been collected, appropriate speeds can be calculated using simple mathematics. These speeds could be radial speeds in rpm, linear speeds in M/min it is entirely down to the mathematical manipulation placed on the SPD results. The following interpretations could be used;

$$\text{Linear speed } N \text{ (km/h)} = \frac{3600 \times (D)}{n \times S2} \times 10^3$$

Where n = the number of linear encoder divisions per kilometer

$$\text{Radial speed } N \text{ (rpm)} = \frac{60 \times (D)}{n \times S2} \times 10^3$$

Where n = the number of encoder pulses per revolution of the encoder disk.

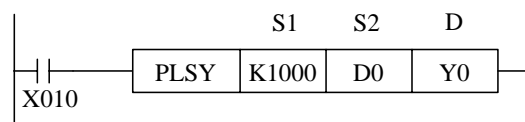
---

## 4.6.7 PLSY (FNC 57)

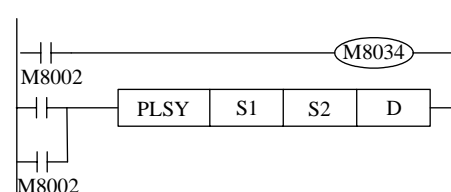
Mnemonic	Function	Operands			Program steps
		S1	S2	D	
PLSY FNC 57 (Pulse Y output)	Outputs a specified number of pulses at a set frequency	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z		Y Y000 , Y001 only	PLSY: 7 steps DPLSY: 13steps

**Operation:**

A specified quantity of pulses S2 is output through device D at a specified frequency S1. This instruction is used in situations where the quantity of outputs is of primary concern.

**Points to note:**

- The maximum frequency:  
16 bit: 1~321767 Hz  
32 bit: 1~100000 HZ
- The maximum number of pulses: 16 bit operation:  
1 to 32,767 pulses, 32 bit operation: 1 to 2,147,483,647 pulses.



Note: special auxiliary coil M8029 is turned ON when the specified number of pulses has been completed. The pulse count and completion flag (M8029) are reset when the PLSY instruction is de-energized. If "0" (zero) is specified the PLSY instruction will continue generating pulses for as long as the instruction is energized.

- A single pulse is described as having a 50% duty cycle. This means it is ON for 50% of the pulse and consequently OFF for the remaining 50% of the pulse. The actual output is controlled by interrupt handling, i.e. the output cycle is NOT affected by the scan time of the program.
- The data in operands S1 and S2 may be changed during execution. However, the new data in S2 will not become effective until the current operation has been completed, i.e. the instruction has been reset by removal of the drive contact.
- This instruction can only be used once within a program scan. Also, only one of either FNC57 PLSY or FNC 59 PLSR can be in the active program at once. It is possible to use subroutines or other such programming techniques to isolate different instances of these instructions. In this case, the current instruction must be deactivated before changing to the new instance
- Because of the nature of the high speed output, transistor output units should be used with this instruction. Relay outputs will suffer from a greatly reduced life and will cause false outputs to occur due to the mechanical 'bounce' of the contacts. To ensure a 'clean' output signal when using transistor units, the load current should be 200mA or higher. It may be found that 'pull up' resistors will be required.
- User can use the HSZ (FNC 55) instruction with the PLSY instruction when source device S1 is set to D8132. Please see page 5-59 for more details.



## 4.6.8 PWM (FNC 58)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
PWM FNC 58 (Pulse width modulation)	Generates a pulse train with defined pulse characteristics	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z Note: S1 S2		Y: Y000 , Y001 only	PWM: 7 steps

**Operation:**

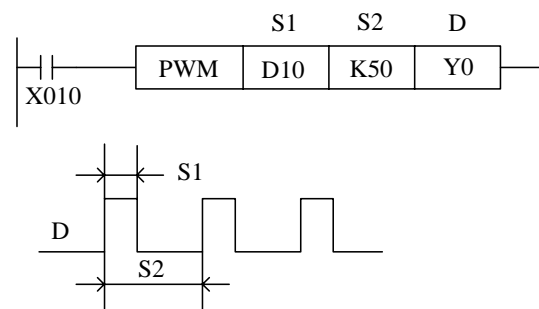
A continuous pulse train is output through device D when this instruction is driven. The characteristics of the pulse are defined as:

The distance, in time (msec), between two identical parts of consecutive pulses (S2).

And how long, also in time (msec), a single pulse will be active for (S1).

**Points to note:**

- Because this is a 16 bit instruction, the available time ranges for S1 and S2 are 1 to 3000.
- A calculation of the duty cycle is easily made by dividing S1 by S2. Hence S1 cannot have a value greater than S2 as this would mean the pulse is on for longer than the distance between two pulses, i.e. a second pulse would start before the first had finished. If this is programmed an error will occur.  
This instruction is used where the length of the pulse is the primary concern.
- The PWM instruction may only be used once in a users program.
- Because of the nature of the high speed output, transistor output units should be used with this instruction. Relay outputs will suffer from a greatly reduced life and will cause false outputs to occur due to the mechanical 'bounce' of the contacts. To ensure a 'clean' output signal when using transistor units, the load current should be 200mA or higher. It may be found that 'pull up' resistors will be required.



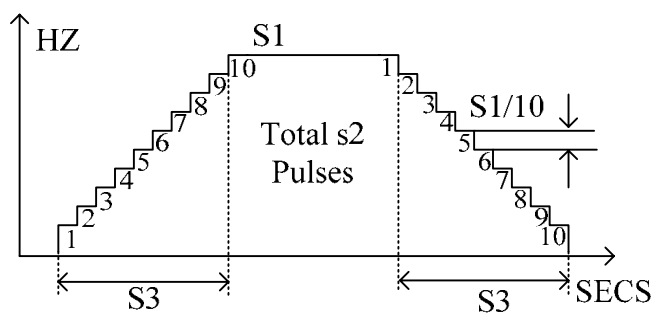
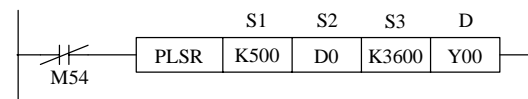
## 4.6.9 PLSR (FNC 59)

Mnemonic	Function	Operands				Program steps
		S1	S2	S3	D	
PLSR FNC 59 (Pulse ramp)	Outputs a specified number of pulses, ramping up to a set frequency and back down to stop	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z			Y: Y000 , Y001 only	PLSR: 9 steps DPLSR: 17 steps

**Operation:**

A specified quantity of pulses S2 is output through device D. The output frequency is first ramped up in 10 steps to the maximum frequency S1 in acceleration time S3 ms, then

ramped down to stop also in S3 ms. This instruction is used to generate simple acceleration/deceleration curves where the quantity of outputs is of primary concern.

**Points to Note:**

- Users may use frequencies of 10 to 100,000Hz. The frequency should be set to a multiple of 10. If not it will be rounded up to the next multiple of 10.  
The acceleration and deceleration steps are set to 1/10 of the maximum frequency. Take this in to consideration to prevent slipping, when using stepping motors.
- The maximum number of pulses: 16 bit operation: 110 to 32,767 pulses, 32 bit operation: 110 to 2,147,483,647 pulses.  
Correct pulse output can not be guaranteed for a setting of 110.
- The acceleration time must conform to the limitations described on the next page.
- The output device is limited to Y0 or Y1 only and should be transistor type.
- This instruction can only be used once within a program scan. Also, only one of either FNC 57 PLSY or FNC 59 PLSR can be in the active program at once.  
It is possible to use subroutines or other such programming techniques to isolate different instances of this instruction. In this case, the current instruction must be deactivated before changing to the new instance.
- If the number of pulses is not enough to reach the maximum frequency then the frequency is automatically cut
- Special auxiliary coil M8029 turns ON when the specified number of pulses has been completed. The pulse count and completion flag (M8029) are reset when the

PLSR instruction is de-energized.

**Acceleration time limitations**

The acceleration time S3 has a maximum limit of 5000 ms. However, the actual limits of S3 are determined by other parameters of the system according to the following 4 points.

- 1) Set S3 to be more than 10 times the maximum program scan time (D8012).

If set to less than this, then the timing of the acceleration steps becomes uneven.

$$S3 \geq \frac{9000}{S1} \times 5$$

- 2) The following formula gives the minimum value for S3.

- 3) The following formula gives the maximum value for S3.

$$S3 \leq \frac{S2}{S1} \times 818$$

- 4) The pulse output always increments in 10 step up to the maximum frequency as shown on the previous page.

If the parameters do not meet the above conditions, reduce the size of S1.

- Possible output frequency is limited to 10 to 100,000 Hz. If either the maximum frequency or the acceleration step size are outside this limit then they are automatically adjusted to bring the value back to the limit.
  - If the drive signal is switch off, all output stops. When driven ON again, the process starts from the beginning.
  - Even if the operands are changed during operation, the output profile does not change. The new values take effect from the next operation.
-

## 4.7 Handy Instructions - Functions 60 to 69

### Contents:

IST -	Initial State	FNC 60
SER -	Search	FNC 61
ABSD -	Absolute Drum	FNC 62
INCD -	Incremental Drum	FNC 63
TTMR -	Teaching Timer	FNC 64
STMR -	Special Timer - Definable	FNC 65
ALT -	Alternate State	FNC 66
RAMP -	Ramp - Variable Value	FNC 67
ROTC -	Rotary Table Control	FNC 68
SORT -	Sort Data	FNC 69

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D1, S3 or for lists/abled devices D3+0, S+9 etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.

P - A 16 bit mode instruction modified to use pulse (single) operation.

D - An instruction modified to operate in 32 bit operation.

D P - A 32 bit mode instruction modified to use pulse (single) operation.

- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.

- An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

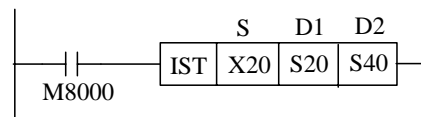
---

## 4.7.1 IST (FNC 60)

Mnemonic	Function	Operands			Program steps
		S	D1	D2	
IST FNC 60 (Initial state)	Automatically sets up a multi-mode STL operating system	X, Y, M, S, Note: uses 8 consecutive devices	S, Note: S20~S1023 ,D1 must be lower than D2		IST: 7 steps

**Operation:**

This instruction automatically sets up a multi-mode STL operating system. This consists of variations of 'manual' and 'automatic' operation modes.

**Points to note:**

- The IST instruction automatically assigns and uses many bit flags and word devices; these are listed in the boxed column on the right of this page.
- The IST instruction may only be used **ONCE**.  
It should be programmed close to the beginning of the program, before the controlled STL circuits.
- The required operation mode is selected by driving the devices associated with operands S+0 through to S+4(5 inputs). None of the devices within this range should be ON at the same time. It is recommended that these 'inputs' are selected through use of a rotary switch.  
If the currently selected operating mode is changed before the 'zero return complete' flag (M8043) is set, all outputs will be turned OFF.
- The 'zero position' is a term used to identify a datum position from where the controlled device, starts from and returns too after it has completed its task. Hence, the operating mode 'zero return', causes the controlled system to return to this datum.

**Assigned devices****Indirect user selected devices:**

- S+0 Manual operation
- S+1 Zero return
- S+2 Step operation
- S+3 One cycle operation
- S+4 Cyclic operation
- S+5 Zero return start
- S+6 Automatic operation start
- S+7 Stop

**Initial states:**

- S0 initiates 'manual' operation
- S1 initiates 'zero return' operation
- S2 initiates 'automatic' operation

**General states:**

- S10 to S19 'zero return' sequence
- D1 to D2 'automatic return' sequence

**Special bit flags:**

- M8040 = ON STL state transfer is inhibited
- M8041 = ON initial states are enabled
- M8042 = Start pulse given by start input
- M8043 = ON zero return completed
- M8044 = ON machine zero detected
- M8047 = ON STL monitor enabled

- The available operating modes are split into two main groups, manual and automatic. There are sub-modes to these groups. Their operation is defined as:

**Manual**

Manual (selected by device S+0)- Power supply to individual loads is turned ON and OFF

by using a separately provided means, often additional push buttons.

Zero Return (selected by device S+1) -Actuators are returned to their initial positions when the Zero input (S+5) is given.

#### **Automatic**

One Step (selected by device S+2)- The controlled sequence operates automatically but will only proceed to each new step when the start input (S+6) is given.

One Cycle (selected by device S+3) - The controlled actuators are operated for **one** operation cycle. After the cycle has been completed, the actuators stop at their 'zero' positions. The cycle is started after a 'start' input (S+6) has been given.

A cycle which is currently being processed can be stopped at any time by activating the 'stop' input (S+7). To restart the sequence from the currently 'paused' position the start input must be given once more.

Automatic (selected by device S+4)-Fully automatic operation is possible in this mode. The programmed cycle is executed repeatedly when the 'start' input (S+6) is given. The currently operating cycle will not stop immediately when the 'stop' input (S+7) is given. The current operation will proceed to then end of the current cycle and then stop its operation.

**Note:** Start, stop and zero inputs are often given by additional, manually operated push buttons.

Please note that the 'stop' input is only a program stop signal. It **cannot** be used as a replacement for an 'Emergency stop' push button. All safety, 'Emergency stop' devices should be hardwired systems which will effectively isolate the machine from operation and external power supplies. Please refer to local and national standards for applicable safety practices.

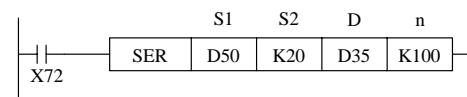
---

## 4.7.2 SER (FNC 61)

Mnemonic	Function	Operands				Program steps
		S1	S2	D	n	
SER FNC 61 (Search a Data Stack)	Generates a list of statistics about a single data value located/found in a data stack	KnX, KnY, KnM, KnS, T, C, D	KnX, KnY, KnM, KnS, T, C, D, V, Z K, H	KnY, KnM, KnS T, C, D Note: 5 consecutive devices are used	K, H, D  Note: n= 1~256 for 16 bit operation n= 1~128 for 32 bit operation	SER, SERP: 9 steps DSER, DSERP: 17 steps

**Operation:**

The SER instruction searches a defined data stack from head address S1, with a stack length n. The data searched for is specified in parameter S2 and the results of the search are stored at destination device D for 5 consecutive devices.



Destination device	Device description
D	Total number of occurrences of the searched value S2 (0 if no occurrences are found)
D+1	The position (within the searched data stack) of the first occurrence of the searched value S2
D+2	The position (within the searched data stack) of the last occurrence of the searched value S2
D+3	The position (within the searched data stack) of the smallest value found in the data stack (last occurrence is returned if there are multiple occurrences of the same value)
D+4	The position (within the searched data stack) of the largest value found in the data stack (last occurrence is returned if there are multiple occurrences of the same value)

**Points to note:**

- Normal rules of algebra are used to determine the largest and smallest values, i.e. -30 is smaller than 6 etc.
- If no occurrence of the searched data can be found then destination devices D, D+1 and D+2 will equal 0 (zero).
- When using data register s as the destination device D please remember that 16 bit operation will occupy 5 consecutive, data registers but 32 bit operation will occupy 10 data registers in pairs forming 5 double words.
- When multiple bit devices are used to store the result (regardless of 16 or 32 bit operation), only the specified size of group is written to for 5 consecutive occurrences, i.e. K1Y0 would occupy 20 bit devices from Y0 (K1 = 4 bit devices and there will be 5 groups for the 5 results). As the maximum data stack is 256 (0 to 255) entries long, the optimum group of bit devices required is K2, i.e. 8 bit devices.

## 4.7.3 ABSD (FNC 62)

Mnemonic	Function	Operands				Program steps
		S1	S2	D	n	
ABSD FNC 62 (Absolute drum sequencer)	Generates multiple output patterns in response to counter data	K X, K Y, K M, K S, (16 bit, =4;32 bit, =8), T, C, D	C 16 bit, C0~C199; 32 bit, C200~C255	Y, M, S	K, H  Note: N . 64	ABSD: 9 steps DABSD: 17 steps.

**Operation:**

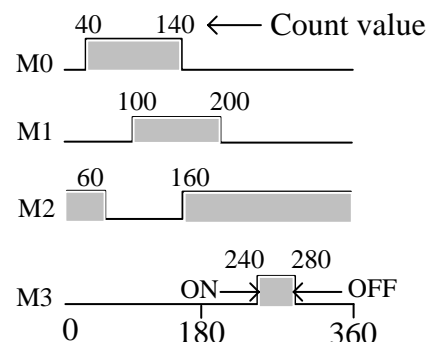
This instruction generates a variety of output patterns (there are n number of addressed outputs) in response to the current value of a selected counter, S2.

**Points to note:**

- The current value of the selected counter (S2) is compared against a user defined data table. This data table has a head address identified by operand S1. S1 should always have an even device number.
- For each destination bit (D) there are two consecutive values stored in the data table. The first allocated value represents the event number when the destination device (D) will be turned ON. The second identifies the reset event. The data table values are allocated as a consecutive pair for each sequential element between D and D+n.
- The data table has a length equal to  $2 \times n$  data entries. Depending on the format of the data table, a single entry can be one data word such as D300 or a group of 16 bit devices e.g. K4X000.
- Values from 0 to 32,767 may be used in the data table.
- The ABSD instruction may only be used **ONCE**.

From the example instruction and the data table below, the following timing diagram for elements M0 to M3 can be constructed.

When counter S2 equals the value below, the destination device D is		Assigned destination device D
turned ON	turned OFF	
D300-40	D301 - 140	M0
D302 - 100	D303 - 200	M1
D304 - 160	D305 - 60	M2
D306 - 240	D307 - 280	M3



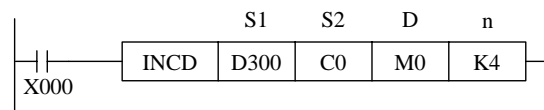


## 4.7.4 INCD (FNC 63)

Mnemonic	Function	Operands				Program steps
		S1	S2	D	n	
INCD FNC 63 (Incremental drum sequencer)	Generates a single output sequence in response to counter data	K X, K Y, K M, K S, (16 bit, =4).T, C, D	C Uses 2 consecutive Counters C0~C198	Y, M, S	K,H  Note: N . 64	INCD: 9 steps

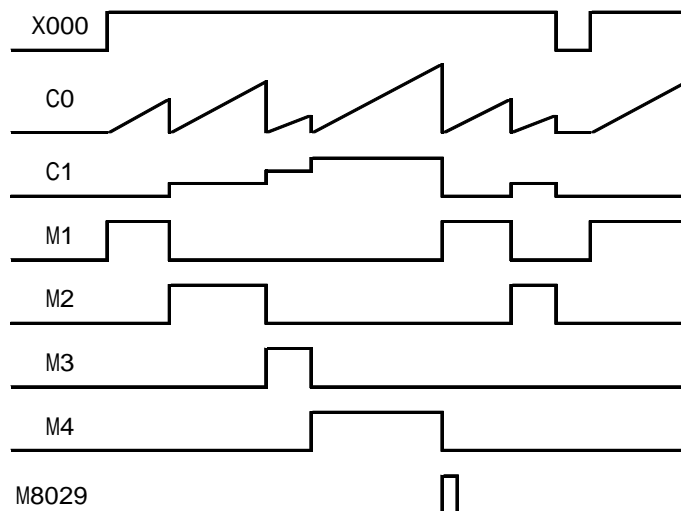
**Operation:**

This instruction generates a sequence of sequential output patterns (there are n number of addressed outputs) in response to the current value of a pair of selected counters (S2, S2+1).

**Points to note:**

- This instruction uses a 'data table' which contains a single list of values which are to be selected and compared by two consecutive counters (S2 and S2+1). The data table is identified as having a head address S1 and consists of n data elements.
- Counter S2 is programmed in a conventional way. The set value for counter S2 **MUST** be greater than any of the values entered into the data table. Counter S2 counts a user event and compares this to the value of the currently selected data element from the data table.  
When the counter and data value are equal, S2 increments the count of counter S2+1 and resets its own current value to '0' (zero). This new value of counter S2+1 selects the new data element from the data table and counter S2 now compares against the new data element's value.
- The counter S2+1 may have values from 0 to n. Once the nth data element has been processed, the operation complete flag M8029 is turned ON. This then automatically resets counter S2+1 hence, the cycle starts again with data element S1+0.
- Values from 0 to 32,767 may be used in the data table.
- The INCD instruction may only be used **ONCE**. From the example instruction and the data table identified left, the following timing diagram for elements M0 to M3 can be constructed.

Data table		Value of Counter S2+1
Data element	Data value / count value for counter S2	
D300	20	0
D301	30	1
D302	10	2
D303	40	3



#### 4.7.5 TTMR (FNC 64)

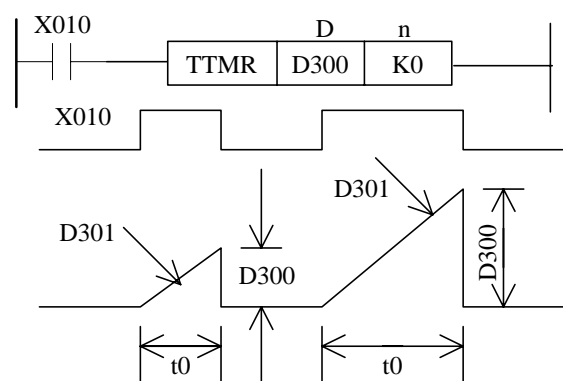
Mnemonic	Function	Operands		Program steps
		D	n	
TTMR FNC 64 (Teaching timer)	Monitors the duration of a signal and places the timed data into a data register	D Note: 2 devices 16 bit words are used D and D+1	K, H  Note: n= 0: (D) = (D+1) X1 n= 1: (D) = (D+1) X10 n= 2: (D) = (D+1) X100	TTMR: 5 steps

##### Operation:

The duration of time that the TTMR instruction is energized, is measured and stored in device D+1 (as a count of 100ms periods).

The data value of D+1 (in secs), multiplied by the factor selected by the operand n, is moved in to register D. The contents of D could be used as the source data for an indirect timer setting or even as raw data for manipulation.

When the TTMR instruction is de-energized D+1 is automatically reset (D is unchanged).



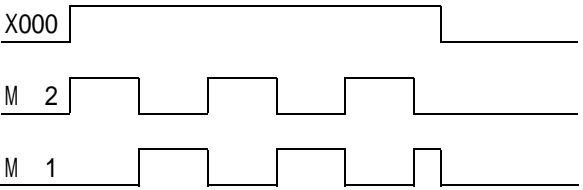
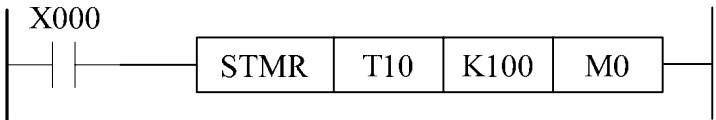
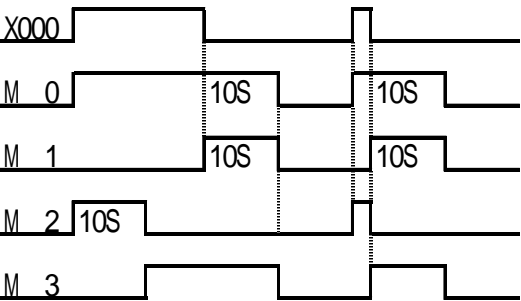
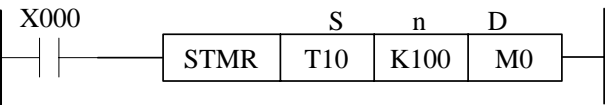
4.7.6 STMR (FNC 65)

Mnemonic	Function	Operands			Program steps
		S	n	D	
STMR FNC 65 (Special timer)	Provides dedicated off-delay, one shot and flash timers	T Note: Timers 0 to 199 (100msec devices)	K, H  Note: n= 1 to 32,767	Y, M, S Note: uses 4 consecutive devices D+0 to D+3	STMR: 7 steps

Operation:

The designated timers will operate for the duration n with the operational effect being

flagged by devices D+0to D+3.  
Device D+0 is an off-delay timer, D+1is a one shot timer. When D+3is used in the configuration below, D+1and D+2act in a alternate flashing sequence.



## 4.7.7 ALT (FNC 66)

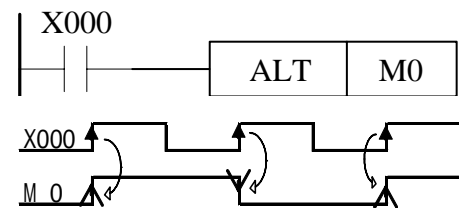
Mnemonic	Function	Operands	Program steps
		D	
ALT FNC 66 (Alternate state)	The status of the assigned device (D) is inverted on every operation of the instruction	Y, M, S	ALT, ALTP: 3 steps

**Operation:**

The status of the destination device (D) is alternated on every operation of the ALT instruction.

This means the status of each bit device will flip flop between ON and OFF. This will occur on every program scan unless a pulse modifier or a program interlock is used.

The ALT instruction is ideal for switching between two modes of operation e.g. start and stop, on and off etc.



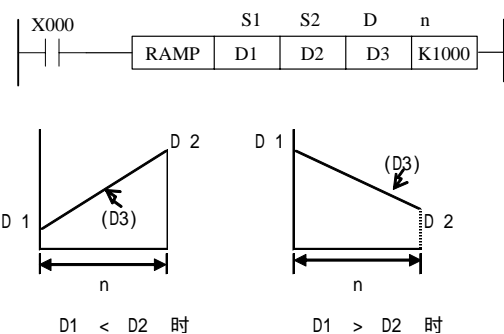
## 4.7.8 RAMP (FNC 67)

Mnemonic	Function	Operands				Program steps
		S1	S2	D	n	
RAMP FNC 67 (Ramp variable value)	Ramps a device from one value to another in the specified number of steps	D Note: Device D uses two consecutive registers identified as D and D+1 these are read only devices.			K, H  Note: n= 1 to 32,767	RAMP: 9 steps

**Operation:**

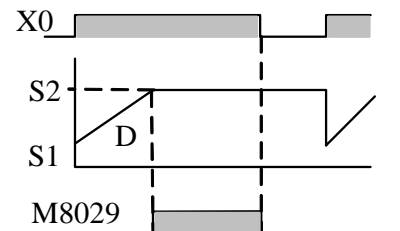
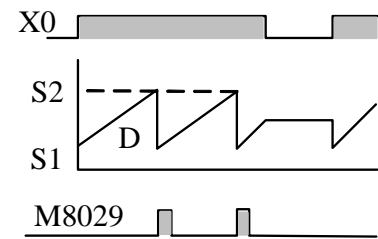
The RAMP instruction varies a current value (D) between the data limits set by the user (S1 and S2).

The 'journey' between these extreme limits takes n program scans. The current scan number is stored in device D+1. Once the current value of D equals the set value of S2 the execution complete flag M8029 is set ON. The RAMP instruction can vary both increasing and decreasing differences between S1 and S2.



**Points to note:**

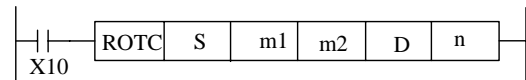
- a) Users may set the operation mode of the RAMP instruction by controlling the state of special auxiliary relay M8026. When M8026 is OFF, the RAMP instruction will be in repeat mode. This means when the current value of D equals S2 the RAMP instruction will automatically reset and start again, i.e. the contents of D will be reset to that of S1 and the device D+1 (the number of current scans) will reset to '0' (zero). This is shown in the diagram opposite. When M8026 is set ON, FX users will be operating the RAMP instruction in 'Hold mode'. This means once the current value of D equals that of S2, the RAMP instruction will 'freeze' in this state. This means the M8029 will be set ON for as long as the instruction remains energized and the value of D will not reset until the instruction is re-initialized, i.e. the RAMP instruction is turned from OFF to ON again.
- b) Users of FX0 and FX0N PLC's cannot change the operating mode of the RAMP instruction. For these PLC's the mode is fixed as in the same case as FX PLC's when M8026 has been set ON, i.e. HOLD mode.
- c) If the RAMP instruction is interrupted before completion, then the current position within the ramp is 'frozen' until the drive signal is re-established. Once the RAMP instruction is redriven registers D and D+1 reset and the cycle starts from its beginning again.
- d) If the RAMP instruction is operated with a constant scan mode, i.e. D8039 is written to with the desired scan time (slightly longer than the current scan time) and M8039 is set ON. This would then allow the number of scans n (used to create the ramp between S1 and S2) to be associated to a time. If 1 scan is equal to the contents of D8039 then the time to complete the ramp is equal to  $n \times D8039$ . The RAMP instruction may also be used with special M flags M8193 and M8194 to mimic the operation of the SER (FNC 61) and RS (FNC 80) respectively when being programmed on older versions of programming peripherals. See page 1-5 for more details.



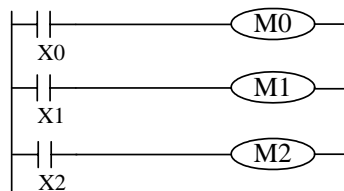
Mnemonic	Function	Operands					Program steps
		S1	M1	M2	D	n	
ROTC FNC 68 (Rotary table control)	Controls a rotary tables movement is response to a requested destination / position	D	K, H  M1 M2		D	K,H,D	ROTC: 9 steps

**Operation:**

The ROTC instruction is used to aid the tracking and positional movement of the rotary table as it moves to a specified destination.

**Points to note:**

- This instruction has many automatically defined devices. These are listed on the right of this page.
- The ROTC instruction may only be used **ONCE**.
- The ROTC instruction uses a built in 2-phase counter to detect both movement direction and distance travelled. Devices D+0 and D+1 are used to input the phase pulses, while device D+2 is used to input the 'zero position' on the rotary table. These devices should be programmed as shown in the example below (where the physical termination takes place at the associated X inputs).



The movement direction is found by checking the relationship of the two phases of the 2 phase counter, e.g

**Assigned devices****Indirect user selected devices:**

- D+0 A-phase counter signal - input
- D+1 B-phase counter signal - input
- D+2 Zero point detection - input
- D+3 High speed forward - output
- D+4 Low speed forward - output
- D+5 Stop - output
- D+6 Low speed reverse - output
- D+7 High speed reverse - output


**Rotary table constants:**


- m1 Number of encoder pulses per table revolution
- m2 Distance to be traveled at low speed (in encoder pulses)

**Operation variables:**


- S+0 Current position at the 'zero point'  
READ ONLY
- S+1 Destination position (selected station to be moved to) relative to the 'zero point' - User defined
- S+2 Start position (selected station to be moved) relative to the 'zero point' -User defined


A phase leads B phase

A-phase 

B-phase 

B phase leads A phase

A-phase 

B-phase 

- d) When the 'zero point' input (D+2) is received the contents of device S+0 is reset to '0' (zero). Before starting any new operation it is advisable to ensure the rotary table is initialized by moving the 'zero point' drive dog or marker around to the 'zero point' sensor. This could be considered as a calibration technique. The re-calibration of the rotary table should be carried out periodically to ensure a consistent/accurate operation.
- e) Devices D+3 to D+7 are automatically set by the ROTC instruction during its operation. These are used as flags to indicate the operation which should be carried out next.
- f) All positions are entered in the form of the required encoder pulses. This can be seen in the following example:

#### - Example:

A rotary table has an encoder which outputs 400 (m1) pulses per revolution. There are 8 stations (0 to 7) on the rotary table. This means that when the rotary table moves from one station to its immediately following station, 50 encoder pulses are counted. The 'zero position' is station '0' (zero). To move the item located at station 7 to station 3 the following values must be written to the ROTC instruction:

$S+1 = 3 \times 50 = 150$  (station 3's position in encoder pulses from the zero point)

$S+2 = 7 \times 50 = 350$  (station 7's position in encoder pulses from the zero point)

$m1 = 400$  (total number of encoder pulses per rev)

The rotary table is required approach the destination station at a slow speed starting from 1.5 stations before the destination. Therefore;

$m2 = 1.5 \times 50 = 75$  slow speed distance either side of the destination station (in encoder pulses)

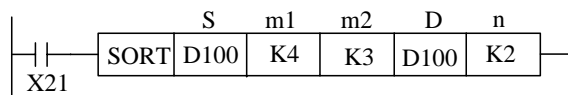
## 4.7.10 SORT (FNC 69)

Mnemonic	Function	Operands					Program steps
		S1	M1	M2	D	n	
SORT FNC 69 (SORT Tabulated Data)	Data in a defined table can be sorted on selected fields while retaining record integrity	D	K, H, D  Note: m1= 1 to 32 m2= 1 to 6		D	K, H D  Note: n = 1 to m2	SORT: 11 steps

**Operation:**

This instruction constructs a data table of m1 records with m2 fields having a start or head address of S. Then the data in field N is

sorted in to numerical order while retaining each individual records integrity. The resulting (new) data table is stored from destination device D.

**Points to note:**

- When a sort occurs each record is sorted in to ascending order based on the data in the selected sort field n.
- The source (S) and destination (D) areas can be the same BUT if the areas are chosen to be different, there should be no overlap between the areas occupied by the tables.
- Once the SORT operation has been completed the 'Operation Complete Flag' M8029 is turned ON. For the complete sort of a data table the SORT instruction will be processed m1 times.
- During a SORT operation, the data in the SORT table must not be changed. If the data is changed, this may result in an incorrectly sorted table.
- The SORT instruction may only be used **ONCE** in a program.

From the example instruction and the 'data table' below left, the following data manipulation will occur when nis set to the identified field

**Original**

		FIELD(m2)		
		1	2	3
R E C O R D  M1	1	D100 32	D104 162	D108 4
	2	D101 74	D105 6	D109 200
	3	D102 100	D106 80	D110 62
	4	D103 7	D107 34	D111 6



Table 1st table sort when n=2

		FIELD(m2)		
		1	2	3
R E C O R D  M1	1	D100 74	D104 6	D108 200
	2	D101 7	D105 34	D109 6
	3	D102 100	D106 80	D110 62
	4	D103 32	D107 162	D111 4

2 nd table sort when n=1

		FIELD(m2)		
		1	2	3
R E C O R D  M1	1	D100 7	D104 34	D108 6
	2	D101 32	D105 162	D109 4
	3	D102 74	D106 6	D110 200
	4	D103 100	D107 80	D111 62

## 4.8 External I/O Devices - Functions 70 to 79

### Contents:

TKY -	Ten Key Input	FNC 70
HKY -	Hexadecimal Input	FNC 71
DSW -	Digital Switch(Thumbwheel input)	FNC 72
SEGD -	Seven Segment Decoder	FNC 73
SEGL -	Seven Segment With Latch	FNC 74
ARWS -	Arrow Switch	FNC 75
ASC -	ASCII Code	FNC 76
PR-	'Print' To A Display	FNC 77

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D1, S3 or for lists/taled devices D3+0, S+9 etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.

P - A 16 bit mode instruction modified to use pulse (single) operation.

D - An instruction modified to operate in 32 bit operation.

D P - A 32 bit mode instruction modified to use pulse (single) operation.

- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.

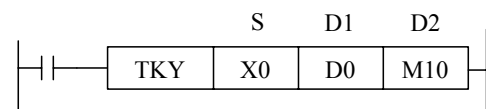
- An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

#### 4.8.1 TKY (FNC 70)

Mnemonic	Function	Operands			Program steps
		S	D1	D2	
TKY FNC 70 (Ten key input)	Reads 10 devices with associated decimal values into a single number	X, Y, M, S Note: uses 10 consecutive devices (identified as S+0 to S+9)	KnY, KnM, KnS, T, C, D, V, Z Note: uses 2 consecutive devices for 32 bit operation	Y, M, S Note: uses 11 consecutive devices (identified D2+0 to D2+10)	TKY: 7 steps DTKY: 13 steps

##### Operation:

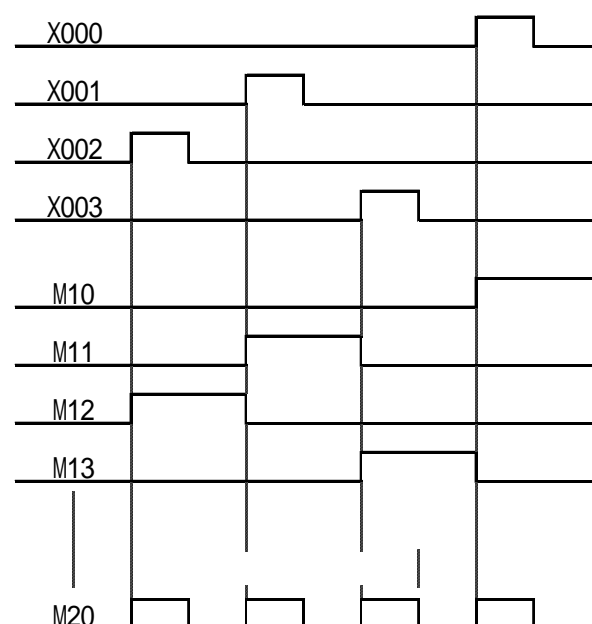
This instruction can read from 10 consecutive devices(S+0 to S+9) and will store an entered numeric string in device D1.



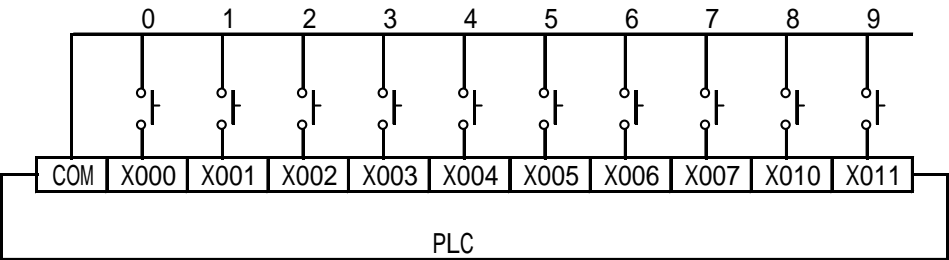
##### Points to note:

- When a source device becomes active its associated destination (bit) device D2 also becomes active. This destination device will remain active until another source device is operated. Each source device maps directly to its own D2 device, i.e. S+0 maps to D2+0, S+7 maps to D2+7 etc. These in turn, map directly to decimal values which are then stored in the destination data devices specified by D1.
- One source device may be active at any one time. The destination device D2+10 is used to signify that a key (one of the 10 source devices) has been pressed. D2+10 will remain active for as long as the key is held down. When the TKY instruction is active, every press of a key adds that digit to the stored number in D1. When the TKY is OFF, all of the D2 devices are reset, but the data value in D1 remains intact.
- When the TKY instruction is used with 16 bit operation, D1 can store numbers from 0 to 9,999 i.e. max. 4 digits. When the DTKY instruction is used (32 bit operation) values of 0 to 99,999,999 (max. 8 digits) can be accommodated in two consecutive devices D1 and D1+1.

In both cases if the number to be stored exceeds the allowable ranges, the highest digits will overflow until an allowable number is reached. The overflowed digits are lost and can no longer be accessed by the user. Leading



- zero's are not accommodated, i.e. 0127 will actually be stored as 127 only.
- d) The TKY instruction may only be used **ONCE**.
  - e) Using the above instruction as a brief example: If the 'keys' identified (a) to (d) are pressed in that order the number 2,130 will be entered into D1. If the key identified as (e) is then pressed the value in D1 will become 1,309. The initial '2' has been lost.

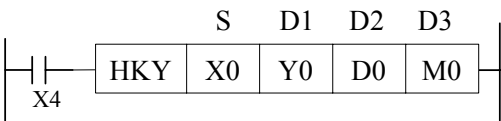


4.8.2 HKY (FNC 71)

Mnemonic	Function	Operands				Program steps
		S	D1	D2	D3	
HKY FNC 71 (Hexadecimal key input)	Multiplexes inputs and outputs to create a numeric keyboard with 6 function keys	X, Note: uses 4 consecutive devices	Y, Note: uses 4 consecutive devices	T, C, D, V, Z	Y, M, S	HKY: 9 steps DHKY: 17 steps

Operation 1 - Standard:

This instruction creates a multiplex of 4 outputs (D1) and 4 inputs (S) to read in 16 different devices. Decimal values of 0 to 9 can be stored while 6 further function flags may be set.



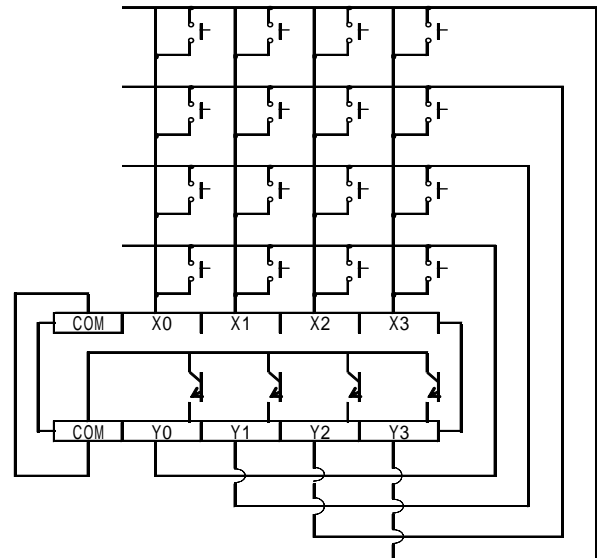
Points to note:

- a) Each of the first 10 multiplexed source devices (identified as 0 to 9) map directly to decimal values 0 to 9. When entered, i.e. a source device is activated, then its associated decimal value is added to the data string currently stored in D2. Activation of any of these keys causes bit device D3+7 to turn ON for the duration of that key press.

- b) The last 6 multiplexed source devices (identified as function keys A to F) are used to set bit devices D3+0 to D3+5 respectively. These bit flags, once set ON, remain ON until the next function key has been activated.

Activation of any of these keys causes bit device D3+6 to turn ON for the duration of that key press.

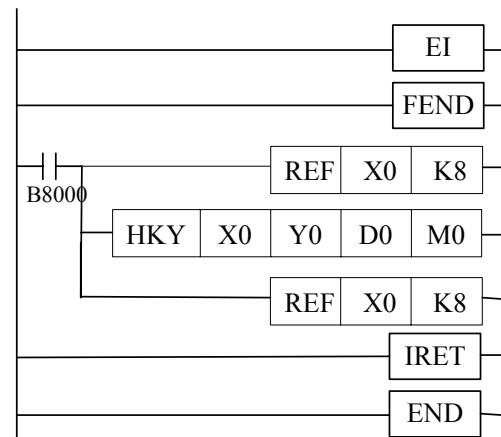
- c) In all key entry cases, when two or more keys are pressed, only the key activated first is effective. When the pressing of a key is sensed the M8029 (execution complete flag) is turned ON. When the HKY instruction is OFF, all D3 devices are reset but data value D2 remains intact.



- d) When the HKY instruction is used with 16 bit operation, D2 can store numbers from 0 to 9,999 i.e. max. 4 digits. When the DHKY instruction is used (32 bit operation) values of 0 to 99,999,999 (max. 8 digits) can be accommodated in two consecutive devices D2 and D2+1. In both cases if the number to be stored exceeds the allowable ranges, the highest digits will overflow until an allowable number is reached. The over-flowed digits are lost and can no longer be accessed by the user. Leading zero's are not accommodated, i.e. 0127 will actually be stored as 127 only. This operation is similar to that of the TKY instruction.

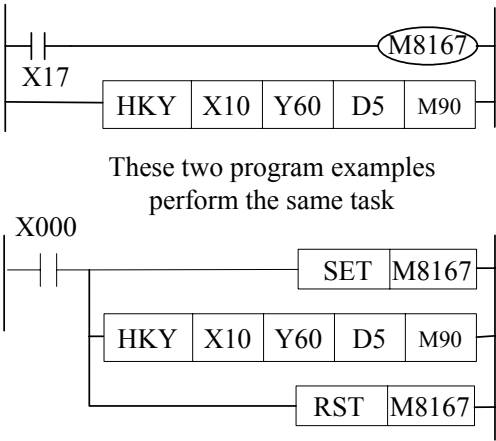
- e) The HKY instruction may only be used **ONCE**.

- f) Normal operation requires 8 scans to read the key inputs. To achieve a steady and repeatable performance, constant scan mode should be used, i.e. M8039 is set ON and a user defined scan time is written to register D8039. However, for a faster response the HKY instruction should be programmed in a timer interrupt routine as shown in the example opposite.



**Operation 2 - Using the HKY Instruction With M8167:**

When the HKY instruction is used with flag M8167 ON (as shown right), the operation of keys A through F allow actual entry of the Hexadecimal values of A through F respectively into the data device D2. This is in addition to the standard 0 through 9 keys. All other operation is as specified in 'Operation 1 - Standard'. Maximum storage values for this operation become FFFF in 16 bit mode and FFFFFFFF in 32 bit (double word) mode.

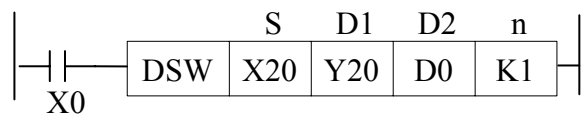


### 4.8.3 DSW (FNC 72)

Mnemonic	Function	Operands				Program steps
		S	D1	D2	n	
DSW FNC 72 (Digital switch)	Multiplexed reading of n sets of digital (BCD) thumbwheels	X Note: If n=2 then 8 devices else 4.	Y Note: uses 4 consecutive devices	T, C, D, V, Z Note: If n=2 then 2 devices else 1	K, H Note: n= 1 or 2	DSW: 9 steps

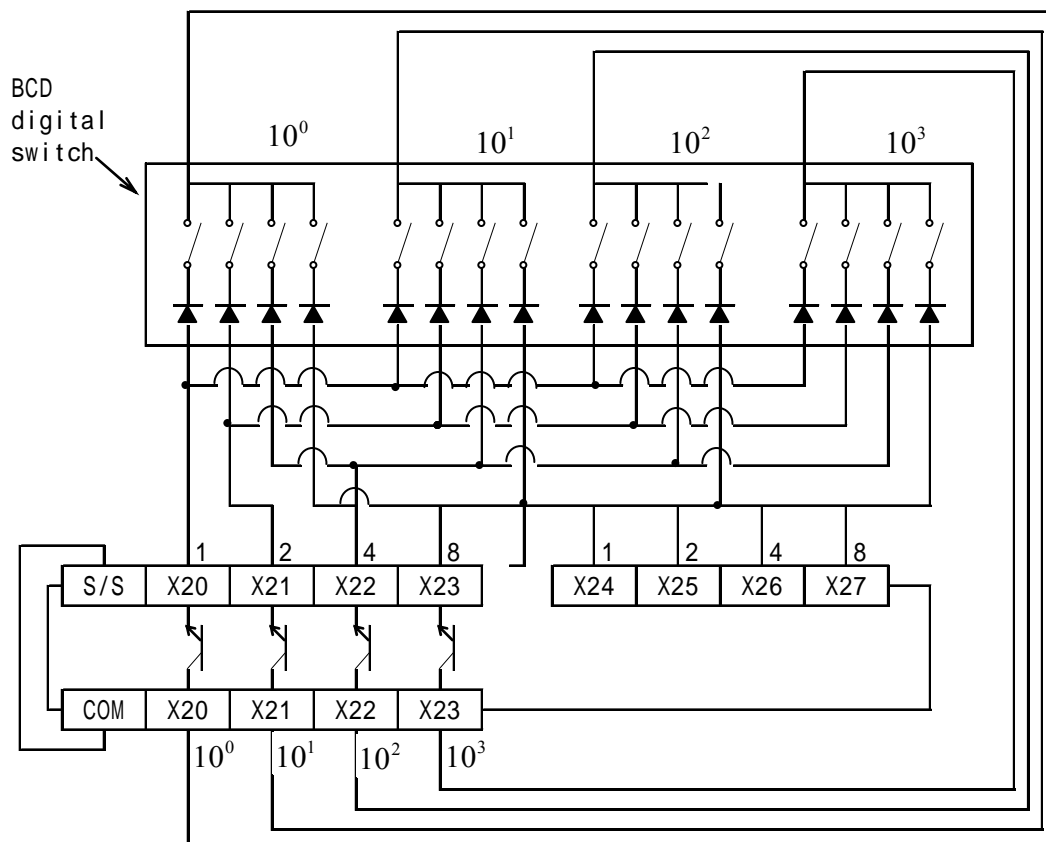
#### Operation:

This instruction multiplexes 4 outputs (D1) through 1 or 2(n) sets of switches. Each set of switches consists of 4 thumbwheels providing a single digit input.



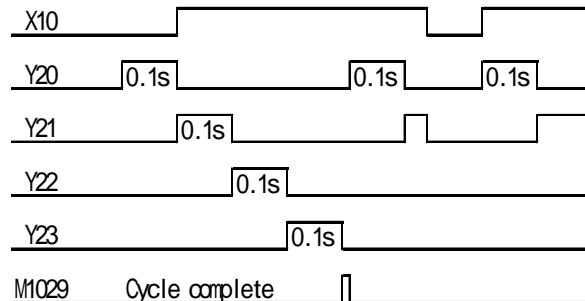
#### Points to note:

- When  $n = 1$  only one set of switches are read. The multiplex is completed by wiring the thumbwheels in parallel back to 4 consecutive inputs from the head address specified in operand S. The (4 digit) data read is stored in data device D2.



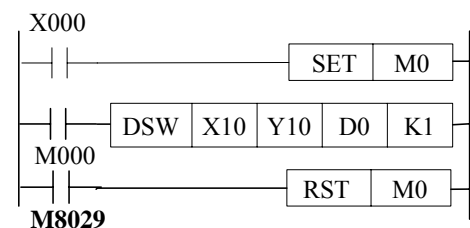
- b) When  $n = 2$ , two sets of switches are read. This configuration requires 8 consecutive inputs taken from the head address specified in operand S. The data from the first set of switches, i.e. those using the first 4 inputs, is read into data device D2. The data from the second set of switches (again 4 digits) is read into data device D2+1.

- c) The outputs used for multiplexing (D1) are cycled for as long as the DSW instruction is driven. After the completion of one reading, the execution complete flag M8029 is set. The number of outputs used does **not** depend on the number of switches  $n$ .



- d) If the DSW instruction is suspended during mid-operation, when it is restarted it will start from the beginning of its cycle and not from its last status achieved.

- e) It is recommended that transistor output units are used with this instruction. However, if the program technique at the right is used, relay output units can be successfully operated as the outputs will not be continually active.





#### 4.8.4 SEGD (FNC 73)

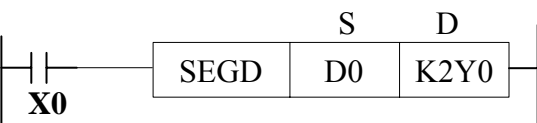
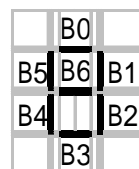
Mnemonic	Function	Operands		Program steps
		S	D	
SEGD FNC 73 (Seven segment decoder)	Hex data is decoded into a format used to drive seven segment displays	K, H KnX, KnY, KnM, KnS, T, C, D, V, Z Note: Uses only the lower 4 bits	KnY, KnM, KnS, T, C, D, V, Z Note: The upper 8 bits remain unchanged	SEGD, SEGDP: 5 steps

##### Operation:

A single hexadecimal digit (0 to 9, A to F) occupying the lower 4 bits of source device S is decoded into a data format used to

drive a seven segment display. A representation of the hex digit is then displayed. The decoded data is stored in the lower 8 bits of destination device D. The upper 8 bits of the same device are not

written to. The diagram opposite shows the bit control of the seven segment display. The active bits correspond to those set to 1 in the lower 8 bits of the destination device D.



It can be seen that B7 is NOT used. Hence B7 of the destination device D will always be OFF

#### 4.8.5 SEGL (FNC 74)

Mnemonic	Function	Operands			Program steps
		S	D	n	
SEGL FNC 74 (Seven segment with latch)	Writes data to multiplexed single digit displays – 4 digits per set, max. 2 sets	K, H KnX, KnY, KnM, KnS T, C, D, V, Z	Y Note: n = 0 to 3, 8 outputs are Used n = 4 to 7, 12 outputs are used	K, H, Note: n= 0 to 3, 1 set of 7 Seg active n= 4 to 7, 2 sets of 7 Seg active	SEGL: 7 steps

##### Operation:

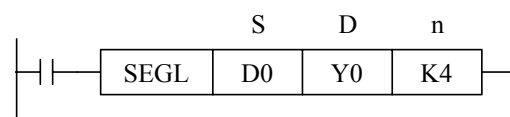
This instruction takes a source decimal value (S) and writes it to a set of 4 multiplexed, outputs (D).

Because the logic used with latched seven

segment displays varies between display manufactures, this instruction can be modified to suit most logic requirements. Configurations are selected depending on the value of n, see the following page.

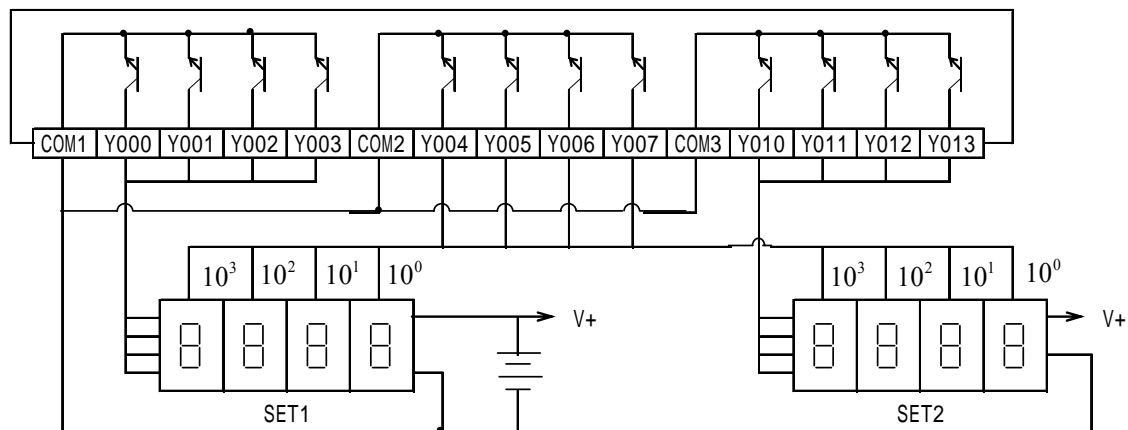
##### Points to note:

- Data is written to a set of multiplexed outputs (D+0 to D+7, 8 outputs) and hence seven



segment displays. A set of displays consists of 4 single digit seven segment units. A maximum of two sets of displays can be driven with this instruction. When two sets are used the displays share the same strobe outputs (D+4 to D+7 are the strobe outputs). An additional set of 4 output devices is required to supply the new data for the second set of displays (D+10 to D+13, this is an octal addition). The strobe outputs cause the written data to be latched at the seven segment display.

- b) Source data within the range of 0 to 9,999 (decimal) is written to the multiplexed outputs. When one set of displays are used this data is taken from the device specified as operand S. When two sets of displays are active the source device S+1 supplies the data for the second set of displays. This data must again be within the range 0 to 9,999. When using two sets of displays the data is treated as **two** separate numbers and is **not** combined to provide a single output of 0 to 99,999,999.
- c) The SEGL instruction takes 12 program scans to complete one output cycle regardless of the number of display sets used. On completion, the execution complete flag M8029 is set.

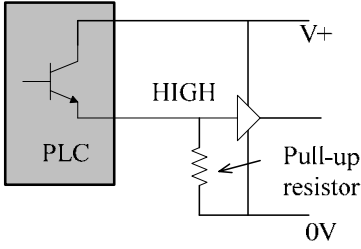
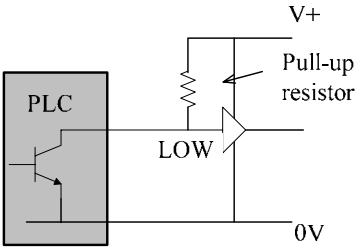


- d) If the SEGL instruction is suspended during mid-operation, when it is restarted it will start from the beginning of its cycle and not from its last status achieved.
- e) The PLC can operate a maximum of **TWO** SEGL instructions.

#### Selecting the correct value for operand n

The selection of parameter n depends on 4 factors;

- 1) The logic type used for the PLC output
- 2) The logic type used for the seven segment data lines
- 3) The logic type used for the seven segment strobe signal
- 4) How many sets of displays are to be used

Device considered		Positive logic	Negative logic
<b>PLC Logic</b>			
		With a source output, when the output is HIGH the internal logic is '1'	With a sink output, when the output is LOW the internal logic is '1'
<b>Seven segment Display logic</b>	Strobe signal logic	Data is latched and held when this signal is HIGH, i.e. its logic is '1'	Data is latched and held when this signal is LOW, i.e. its logic is '1'
	Data signal logic	Active data lines are held HIGH, i.e. they have a logic value of '1'	Active data lines are held LOW, i.e. they have a logic value of '1'

There are two types of logic system available, positive logic and negative logic. Depending on the type of system, i.e. which elements have positive or negative logic the value of n can be selected from the table below with the final reference to the number of sets of seven segment displays being used:

PLC Logic	Seven segment display logic		n	
	Data Logic	Strobe logic	1 display set	2 display sets
Positive (Source)	Positive (High)	Positive (High)	0	4
Negative (Sink)	Negative (Low)	Negative (Low)		
Positive (Source)	Positive (High)	Negative (Low)	1	5
Negative (Sink)	Negative (Low)	Positive (High)		
Positive (Source)	Positive (High)	Negative (Low)	2	6
Negative (Sink)	Negative (Low)	Positive (High)		
Positive (Source)	Positive (High)	Positive (High)	3	7
Negative (Sink)	Negative (Low)	Negative (Low)		

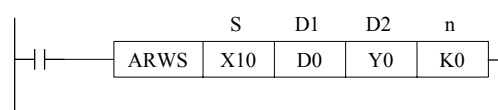
#### 4.8.6 ARWS (FNC 75)

Mnemonic	Function	Operands				Program steps
		S	D1	D2	n	
ARWS FNC 75 (Arrow switch)	Creates a user defined, (4 key) numeric data entry panel	X, Y, M, S Note: uses 4 consecutive devices	T, C, D, V, Z Note: data is stored in a decimal format	Y Note: uses 8 consecutive devices	K, H  Note: n= 0 to 3,	ARWS: 9 steps

##### Operation:

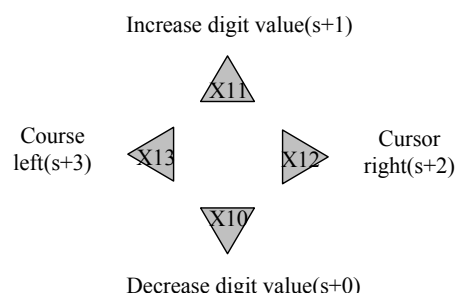
This instruction displays the contents of a single data device D1 on a set of 4 digits, seven segment displays. The data within D1 is actually in a standard decimal format but is automatically

converted to BCD for display on the seven segment units. Each digit of the displayed number can be selected and edited. The editing procedure directly changes the value of the device specified as D1.

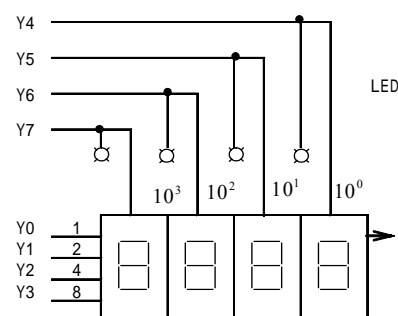


##### Points to note:

- The data stored in destination device D1 can have a value from the range 0 to 9,999 (decimal), i.e. 4 digit data. Each digit's data value, can be incremented (S+1) or decremented (S+0) by pressing the associated control keys. The edited numbers automatically 'wrap-around' from 9 - 0 - 1 and 1 - 0 - 9. The digit data is displayed by the lower 4 devices from D2, i.e. D2+0 to D2+3.



- On initial activation of the ARWS instruction, the digit in the numeric position 10 3 is currently selected. Each digit position can be sequentially 'cursored through' by moving to the left (S+2) or to the right (S+3). When the last digit is reached, the ARWS instruction automatically wraps the cursor position around, i.e. after position 10 3, position 10 0 is selected and vice-versa. Each digit is physically selected by a different 'strobe' output.



- To aid the user of an operation panel controlled with the ARWS instruction, additional lamps could be wired in parallel with the strobe outputs for each digit. This would indicate which digit was currently selected for editing.
- The parameter n has the same function as parameter n of the SEGL instruction –

please see page5-86, 'Selecting the correct value for operand n'. Note: as the ARWS instruction only controls one set of displays only values of 0 to 3 are valid for n.

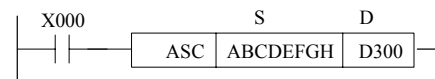
- e) The ARWS instruction can be used ONCE. This instruction should only be used on transistor output PLC's.

#### 4.8.7 ASC (FNC 76)

Mnemonic	Function	Operands		Program steps
		S	D	
ASC FNC 76 (ASCII code conversion)	An entered alphanumeric string can be converted to its ASCII codes	Alphanumeric data e.g.0-9, A - Z and a - z etc. Note: Only one, 8 character string may be entered at any one time.	T, C, D Note: uses 4 consecutive devices	ASC: 7 steps

##### Operation:

The source data string S consists of up to 8 characters taken from the printable ASCII character (Char) set. If less than 8 Char are used, the difference is made up with null Char (ASCII 00).



The source data is converted to its associated ASCII codes. The codes are then stored in the destination devices D, see example shown below.

D	Byte	
	High	Low
D300	42 (B)	41 (A)
D301	44 (D)	43 (C)
D302	46(F)	45 (E)
D303	48 (H)	47 (G)

##### Note:

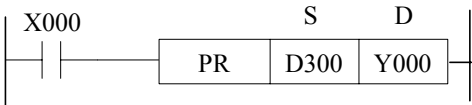
- ASCII Char **cannot** be entered from a hand held programmer.
- When=ON, only low 8 bytes of device D can be available for storing data and high 8 bytes will be written with 0.

4.8.8 PR (FNC 77)

Mnemonic	Function	Operands		Program steps
		S	D	
PR FNC 77 (Print)	Outputs ASCII data to items such as display units	T, C, D Note: 8 byte mode (M8027=OFF) uses 4 consecutive devices 16 byte mode (M8027= ON) uses 8 consecutive devices	Y Note: uses 10 consecutive devices.	PR: 5 steps

Operation:

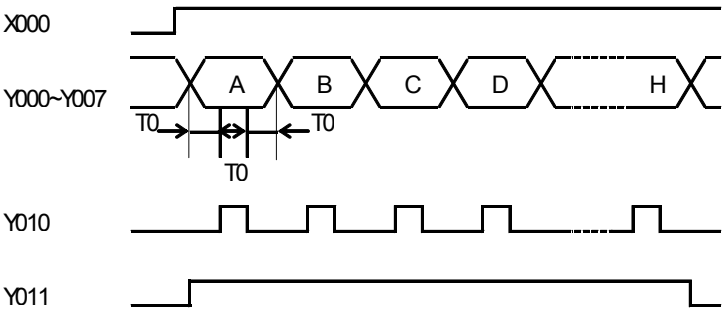
Source data (stored as ASCII values) is read byte by byte from the source data devices. Each byte is mapped directly to the first 8 consecutive destination devices D+0 to D+7). The final two destination bits provide a strobe signal (D+10, numbered in octal) and an execution/busy flag (D+11, in octal).



Points to note:

- a) The source byte-data maps the lowest bit to the first destination device D+0. Consequently the highest bit of the byte is sent to destination device D+7.
- b) The PR instruction may be used **ONCE** .
- c) The operation of the PR instruction is program scan dependent. Under standard circumstances it takes 3 program scans to send 1 byte. However, for a faster operation the PR instruction could be written into a timer interrupt routine similar to the one demonstrated for HKY on page 5-82.

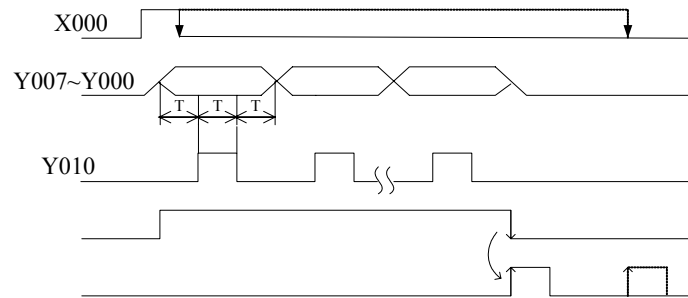
- d) 8 byte operation has the following timing diagram. It should be noted that when the drive input (in the example X0) is switched OFF the PR instruction will



cease operation. When it is restarted the PR instruction will start from the beginning of the message string. Once all 8 bytes have been sent the execution/busy flag is dropped and the PR instruction suspends operation.

- e) 16 byte operation requires the special auxiliary flag M8027 to be driven ON (it is recommended that M8000 is used as a drive input). In this operation mode the drive input (in the example X0) does not have to be active all of the time. Once the PR instruction is activated it will operate continuously until all 16 bytes of data have been

sent or the value 00H (null)  
has been sent. Once the  
operation is complete the  
execution/busy flag (D+11,  
octal) is turned OFF and  
M8029 the execution  
complete flag is set.



## 4.9 External Devices - Functions 80 to 88

### Contents:

RS -	RS Communications	FNC 80
PRUN -	Parallel Run	FNC 81
ASCI -	Hexadecimal to ASCII	FNC 82
HEX -	ASCII to Hexadecimal	FNC 83
CCD -	Check Code	FNC 84
VRRD	Volume Read	FNC 85
VRSC	Volume Scale	FNC 86
MBUS -	MBUS Serial Data Transmission	FNC 87
PID -	PID Control Loop	FNC 88

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D1, S3 or for lists/tables devices D3+0, S+9 etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
- P - A 16 bit mode instruction modified to use pulse (single) operation.
- D - An instruction modified to operate in 32 bit operation.
- D P - A 32 bit mode instruction modified to use pulse (single) operation.
- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.
- An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

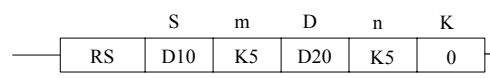


## 4.9.1 RS (FNC 80)

Mnemonic	Function	Operands					Program steps
		S	m	D	n	K	
RS FNC 80 (Serial Communications instruction)	Used to control serial communication s from/to the programmable controller	D (including file registers)	K, H, D  m = 0 to 255	D	K, H, D  m = 0 to 255	0,1	RS: 11 steps

**Operation:**

Such instruction is used to serially send or receive data without protocol, together with the optional RS-232, RS-485 expansion cards or built-in RS485 port.

**Points to note:**

- This instruction has many automatically defined devices. These are listed in the boxed column to the right of this page.
- The RS instruction has two parts, send (or transmission) and receive. The first elements of the RS instruction specify the transmission data buffer (S) as a head address, which contains m number of elements in a sequential stack.  
The specification of the receive data area is contained in the last two parameters of the RS instruction. The destination (D) for received messages has a buffer or stack length of n data elements. The size of the send and receive buffers dictates how large a single message can be. Buffer sizes may be updated at the following times:
  - Transmit buffer - before transmission occurs, i.e. before M8122 is set ON
  - Receive buffer - after a message has been received and before M8123 is reset.
- Data cannot be sent while a message is being received, the transmission will be delayed - see M8121.
- More than one RS instruction can be programmed but only one may be active at any one time.

**Data devices:**

- for RS485 port
  - Sending ready (M8121): the Relay will be set to 1 when it is sending-data request in receiving data. The relay will be automatically cleared to 0 in sending the data.
  - Sending request (M8122): When M8122 is set to 1 by the pulse for sending ready or for sending finish, the data string, which is from (S), whose length is m will be sent. M8122 will automatically reset to 0 which sending is finished.
  - Sending end (M8123): M8123 will be ON when sending is finished. Please reset M8123 only after the received data is saved in certain registers.
  - Overtime judging (M8129): Retry receiving will not begin within the specified times and Overtime flag will be ON. When send finished, M8123 will be reset to 0 and M8129 will be automatically reset.

- 5) Communication frame (D8120): refer to the said frame to MBUS instruction.
- 6) Remaining data number for sending data (D8122)
- 7) Number of received data (D8123)
- 8) Overtime watchdog time (D8129) : watchdog time for communication overtime (5~255):\*10ms.
- b) For expansion card RS485/ RS232
  - 1) Sending ready (M8321)
  - 2) Sending request (M8322)
  - 3) Sending end (M8323)
  - 4) Error flag (M8124)
  - 5) Overtime judging (M8329)
  - 6) Communication frame (D8320)
  - 7) Remaining data number for sending (D8322)

### 4.9.2 PRUN (FNC 81)

Mnemonic	Function	Operands		Program steps
		S	D	
PRUN FNC 81 (Parallel run)	Octal transmission	KnX, KnM	KnY, KnM	PRUN, PRUNP: 5 steps DPRUN, DPRUNP: 9 steps
		Note: n = 1 to 8 For ease and convenience, the head address Bit should be a multiple of '10', e.g. X10, M100, Y30 etc.		

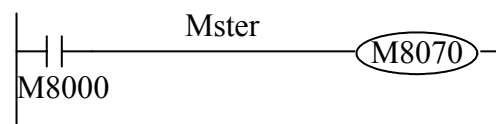
#### Operation:

This instruction allows source data to be moved into the bit transmission area. The actual control of the parallel link communication is by special M flags.

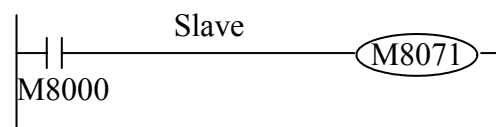


#### Points to note:

- a) Parallel link communications automatically take place when both systems are 'linked' and the Master station (M8070), Slave station flags (M8071) have been set ON (there is no need to have a PRUN instruction for communications).



There can only be one of each type of station as this system connects only two PLCs. The programs shown opposite



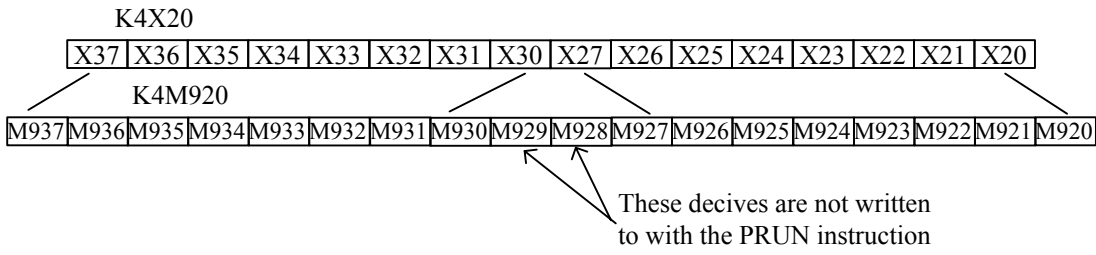
should be inserted into the appropriate PLC programs.

Once the station flags have been set, they can only be cleared by either forcibly resetting them when the PLC is in STOP mode or turning the power OFF and ON again.

- b) During automatic communications the following data is 'swapped' between the Master and Slave PLC's.

Master station		Communication direction	Slave Station	
M8070 = ON	Bit Data		Bit Data	M8071 = ON
	M800 to M899 (100 points)	→	M800 to M899 (100 points)	
	M900 to M999 (100 points)	←	M900 to M999 (100 points)	
	Data words		Data words	
	D490 to D499 (10 points)	→	D490 to D499 (10 points)	
	D500 to D509 (10 points)	←	D500 to D509 (10 points)	

- c) The PRUN instruction enables data to be moved into the bit transmission area or out of the (bit) data received area. The PRUN instruction differs from the move statement in that it operates in octal. This means if K4X20 was moved using the PRUN instruction to K4M920, data would not be written to M928 and M929 as these devices fall outside of the octal counting system. This can be seen in the diagram below.

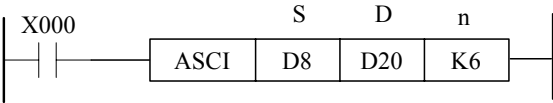


### 4.9.3 ASCII (FNC 82)

Mnemonic	Function	Operands			Program steps
		S	D	n	
ASCII FNC 82 (Converts HEX to ASCII)	Converts a data value from hexadecimal to ASCII	K, H, KnX, KnY, KnM, KnS T, C, D, V, Z	KnY, KnM, KnS, T, C, D	K, H Note: n = 1 to 256	ASCII, ASCIP: 7 steps

#### Operation:

This instruction reads n hexadecimal data characters from head source address (S) and converts them in to the equivalent ASCII code.



This is then stored at the destination (D) for n number of bytes.

#### Points to note:

Please note that data is converted 'as read', i.e. using the example above with the following data in (D9,D8) ABCD<sub>H</sub>, EF26<sub>H</sub>. Taking the first n hexadecimal characters (digits) from the right (in this case n= 6) and converting them to ASCII will store values in 6 consecutive bytes from D20, i.e. D20 = (67, 68), D21 = (69, 70) and D22 = (50, 54) respectively. In true characters symbols that would be read as **CDEF26**.

This can be shown graphically as in the table to the right. Please take special note that the source data (S) read from the most significant device to the least significant. While the destination data (D) is read in the opposite direction.

The ASCII instruction can be used with the M8161, 8 bit/16bit mode flag. The effect of this flag is exactly the same as. The example shows the effect when M8161 is OFF. If M8161 was set ON, then only the lower destination byte (b0-7) would be used to store data and hence 6 data registers would be required (D20 through D25).

Source (S)		Data						
D9	b12-15	A	→	Destination (D)		ASCII Code		Symbol
	b8-11	B				HEX	DEC	
	b4-7	C		D20	b8-15	43	67	'C'
	b0-3	D			b0-7	44	68	'D'
D8	b12-15	E	→	D21	b8-15	45	69	'E'
	b8-11	F	→		b0-7	46	70	'F'
	b4-7	2	→	D22	b8-15	32	50	'2'
	b0-3	6	→		b0-7	36	54	'6'

ASCII Character Codes

The table below identifies the usable hexadecimal digits and their associated ASCII codes

HEX Character		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII Code	HEX	30	31	32	33	34	35	36	37	38	39	41	42	43	44	45	46
	DEC	48	49	50	51	52	53	54	55	56	57	65	66	67	68	69	70
Character Symbol		'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'A'	'B'	'C'	'D'	'E'	'F'

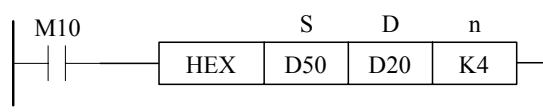
#### 4.9.4 HEX (FNC 83)

Mnemonic	Function	Operands			Program steps
		S	D	n	
HEX FNC 83 (Converts ASCII to HEX)	Converts a data value from ASCII in to a hexadecimal equivalent	K, H, KnX, KnY, KnM, KnS, T, C, D	KnY, KnM, KnS T, C, D, V, Z	K, H Note: n = 1 to 256	HEX, HEXP: 7 steps

##### Operation:

This instruction reads n ASCII data bytes from head source address (S) and converts them in to the equivalent Hexadecimal character.

This is then stored at the destination (D) for n number of bytes.



##### Points to note:

Please note that this instruction 'works in reverse' to the ASCI instruction, i.e. ASCII data stored in bytes is converted into associated hexadecimal characters. The HEX instruction can be used with the M8161 8bit/16bit flag. In this case the source data (S) is read from either the lower byte (8bits) when M8161 is ON, or the whole word when M8161 is OFF i.e. using the example above with the following data in devices D50 and D51 respectively (43H,41H) (42H,31H) and assuming M8161 is ON.

The ASCII data is converted to its hexadecimal equivalent and stored sequentially digit by digit from the destination head address.

If M8161 had been OFF, then the contents of D20 would read CAB1H.

Source (S)		ASCII Code		Symbol
		HEX	DEC	
D51	b8-15	43	67	'C'
	b0-7	41	65	'A'
D50	b8-15	42	66	'B'
	b0-7	31	49	'1'

Destination (D)		Data
D20	b12-15	-
	b8-11	-
	b4-7	A
	b0-3	1

For further details regarding the use of the HEX instruction and about the available ASCII data ranges, please see the following information point 'ASCII Character Codes' under the ASCI instruction on the previous page.

Important:

If an attempt is made to access an ASCII Code (HEX or Decimal) which falls outside of the ranges specified in the table on previous page, the instruction is not executed. Error 8067 is flagged in data register D8004 and error 6706 is identified in D8067. Care should be

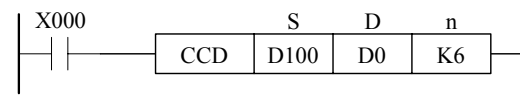
taken when using the M8161 flag, and additional in the specification of the number of element 'n' which are to be processed as these are the most likely places where this error will be caused.

#### 4.9.5 CCD (FNC 84)

Mnemonic	Function	Operands			Program steps
		S	D	n	
CCD FNC 84 (Check Code)	Checks the 'vertical' parity of a data stack	KnX, KnY, KnM, KnS T, C, D	KnY, KnM, KnS T, C, D	K, H D Note: n = 1 to 256	CCD, CCDP: 7 steps

##### Operation:

This instruction looks at a byte (8 bit) stack of data from head address (S) for n bytes and checks the vertical bit pattern for parity and sums the total data stack. These two pieces of data are then stored at the destination (D).



##### Points to note:

- The SUM of the data stack is stored at destination D while the Parity for the data stack is stored at D+1.
- During the Parity check an even result is indicated by the use of a 0 (zero) while an odd parity is indicated by a 1 (one).
- This instruction can be used with the 8 bit/ 16 bit mode flag M8161. The following results will occur under these circumstances.

M8161=OFF										
Source (S)			Bit pattern							
D100	H	FF	1	1	1	1	1	1	1	1
	L	FF	1	1	1	1	1	1	1	1
D101	H	FF	1	1	1	1	1	1	1	1
	L	00	0	0	0	0	0	0	0	0
D102	H	F0	1	1	1	1	0	0	0	0
	L	0F	0	0	0	0	1	1	1	1
Vertical Party D1			0	0	0	0	0	0	0	0
SUM D0			3FC							

M8161=ON										
Source (S)			Bit pattern							
D100	L	FF	1	1	1	1	1	1	1	1
D101	L	00	0	0	0	0	0	0	0	0
D102	L	0F	0	0	0	0	1	1	1	1
D103	L	F0	1	1	1	1	0	0	0	0
D104	L	F0	1	1	1	1	0	0	0	0
D105	L	0F	0	0	0	0	1	1	1	1
Vertical Party D1			1	1	1	1	1	1	1	1
SUM D0			2FD							

It should be noted that when M8161 is OFF 'n' represents the number of consecutive bytes checked by the CCD instruction. When M8161 is ON only the lower bytes of 'n' consecutive words are used.

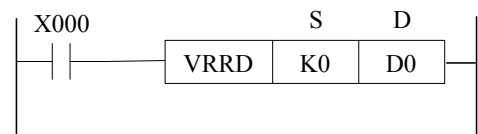
The 'SUM' is quite simply a summation of the total quantity of data in the data stack. The Parity is checked vertically through the data stack as shown by the shaded areas.

#### 4.9.6 VRRD (FNC 85)

Mnemonic	Function	Operands		Program steps
		S	D	
VRRD FNC 85 (Volume read)	Read volume from 2 VRs numbered No0,No1,and 6 VRs on expansion card , numbered No2~No7.	K, H Note: S= 0 to 7	KnY, KnM, KnS T, C, D, V, Z	VRRD, VRRDP: 5 steps

##### Operation:

The analog data is in 10 bit format, i.e. values from 0 to 255 are readable. The read data is stored at the destination device identified under operand D.

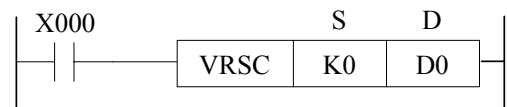


#### 4.9.7 VRSD (FNC 86)

Mnemonic	Function	Operands		Program steps
		S	D	
VRSC FNC 86 (Volume scale)	Read scale (0~10) from 2 VRs numbered No0,No1,and 6 VRs on expansion card , numbered No2~No7.	K, H Note: S= 0 to 7	KnY, KnM, KnS T, C, D, V, Z	VRSC, VRSCP: 5 steps

##### Operation:

The identified volume (S) on the FX-8AV is read as a rotary switch with 11 set positions (0 to 10). The position data is stored at device D as an integer from the range 0 to 10.





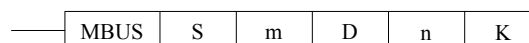
#### 4.9.8 MBUS (FNC 87)

Mnemonic	Function	Operands					Program steps
		S	m	D	n	K	
MBUS FNC 87	Enable Modbus function for optional communication card RS485 / RS232 (all types are available)	D	K,H,D m=0~255	D	K,H,D n=0~255	K,D 0,1	MBUS : 11 steps

##### Operation:

MBUS instruction can enable master communication.

The communication string sent is HEX codes, including command code, function code and communication data. The MBUS



instruction will send the command transferred into ASCII code to BUFF. The command is communication string consisting of certain mode such as RTU mode, together with CRC check code (2bytes) and end character (0DH+0AH).

Receiving string includes address, function code, communication data. The start character, end character and check code will not be saved.

- The communication frame of RS485 port can be set by special register D8120. The PLC will not accept the data modified in D8120 during MBUS operation.
- The communication frame of optional expansion card RS485/ RS232 can be set by special register D8320. PLC will not accept the data modified in D8320 during MBUS operation.
- The length of receiving data 'm' should be set to K0 as there is not data to be sent.
- The program can apply lots of instructions as RS, MBUS, DTLK and RMIO, however, it must be ensured that one communication port only can be driven by one instruction at one time. Off Time for Switching should not be less than a scan time.

Communication specification:

<communication format 「D8120」, 「D8320」 >

D8120, D8320 is mainly used with the instruction F87 (MBUS). Also, they can also be used as special register for other instructions.

However, when F87 (MBUS) is used in the program, the setting of D8120, D8320 relevant to other communication instruction or others will be disabled. Please set the D8120, D8320 according to following directions.

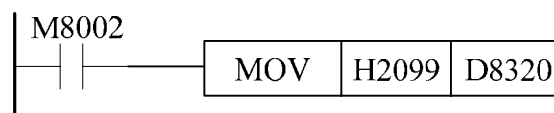
Bit	description	Content	
		0 (OFF)	1 (ON)
B0	Data length	7 bit	8 bit
B1 B2	Parity	B2,B1 (0,0): none (0,1): ODD (1,0): EVEN	
B3	Stop bit	1 bit	2 bit
B4 B5 B6 B7	Baud rate (bps)	B7,B6,B5,B4 (0,1,1,1):9,600 (1,0,0,0):19,200 (1,0,0,1):38,400 (1,0,1,0):57,600 (1,0,1,1):76,800	B7,B6,B5,B4 (1,1,0,0):128,000 (1,1,0,1):153,600 (1,1,1,0):307,200
B8 ~ B12 *1	Reserved		
B13	Modbus mode	(0) : RTU mode	(1) : ASCII mode
B14 ~ B15*1	Reserved		

\*1:B8 ~ B12 ,B14 ,B15 is particularly for other communication mode. When in F87 (MBUS) instruction, all these should be set to 0.

- Example for communication frame

Please set D8320 according to following steps or peripheral communication frame.

	b15			b12	b11	b8			b7	b4			b3	b0		
D8320	0	0	1	0	0	0	0	0	1	0	0	1	1	0	0	1
↓																
D8320	2099H															



The special relay and register related to the instruction:

c) for RS485 port

9) Sending ready (M8121): the Relay will be set to 1 when it is sending-data request in

receiving data. The relay will be automatically cleared to 0 in sending the data.

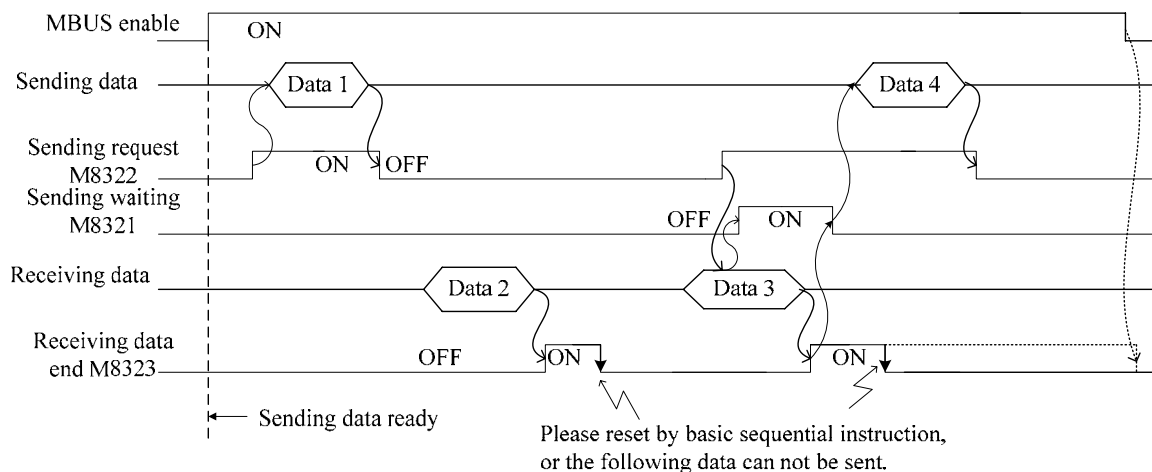
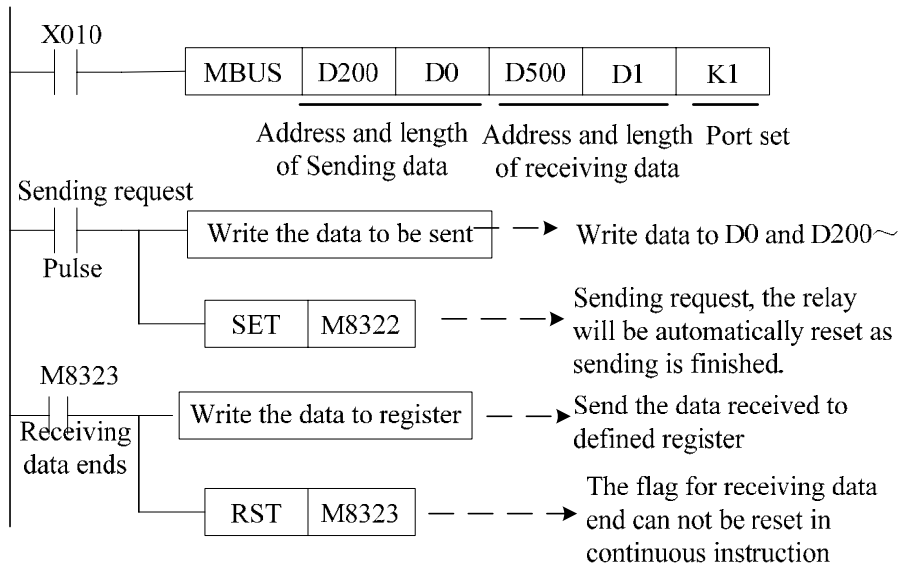
- 10) Sending request (M8122): When M8122 is set to 1 by the pulse for sending ready or for sending finish, the data string, which is from (S), whose length is m will be sent. M8122 will automatically reset to 0 which sending is finished.
- 11) Sending end (M8123): M8123 will be ON when sending is finished. Please reset M8123 only after the received data is saved in certain registers.
- 12) Error flag (M8124): receiving error (RTU mode: CRC error; ASCII mode: LRC or end character error).
- 13) Overtime judging (M8129): Retry receiving will not begin within the specified times and Overtime flag will be ON. When send finished, M8123 will be reset to 0 and M8129 will be automatically reset.
- 14) Communication frame (D8120): refer to the said frame to MBUS instruction.
- 15) Remaining data number for sending data (D8122)
- 16) Number of received data (D8123)
- 17) Overtime watchdog time (D8129) : watchdog time for communication overtime (5~255)\*10ms.

d) For expansion card RS485/ RS232

- 8) Sending ready (M8321)
- 9) Sending request (M8322)
- 10) Sending end (M8323)
- 11) Error flag (M8124)
- 12) Overtime judging (M8329)
- 13) Communication frame (D8320)
- 14) Remaining data number for sending (D8322)
- 15) Number of receiving data (D8323)
- 16) Overtime watchdog time (D8329)

### Sequence for sending and receiving

MBUS instruction specifies data start address and number of the data from PLC, together with first destination for the received data and max number for receivable data. Sequence for MBUS sending and receiving data is as following: (RS485 expansion card is applied.)



### Sending request M8322

- When X010 ON, MBUS instruction will be enabled, and The PLC will be ready to receive data.
- M8322 will be set ON by a pulse as in waiting for receiving data or in receiving data. PLC will send the data starting at D200 and data length D0 out. M8322 will be reset to 0 as sending ends.

### Receiving end M8323

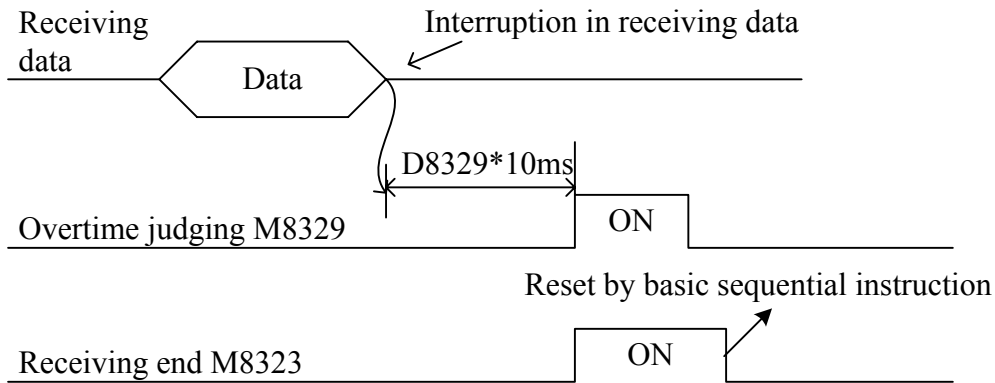
- When Receiving end flag M8323 is ON, The PLC will store all the received data to corresponding registers, then reset M8323 to OFF.
- As long as M8323 is reset to OFF, The PLC will be ready to receive data. If X010 is ON, MBUS instruction will be enabled. Such progress is operating repeatedly.
- When (D1) = 0, MBUS instruction is enabled, M8323 will not operate. Thusly, The PLC will not enter the next round of receiving data. If D1 = 1, the operation 'ON M8323, then OFF M8323' will enable The PLC the next round of receiving data.

### Overtime judging M8329

- If there is interruption in receiving data and the following time lasting as D8329 set, the overtime M8329 will be ON and receiving data will end.

M8329 will be automatically reset as M8323 reset the program.

Receiving the data (ASCII code) without end character also is available with this function.



### Overtime watchdog time

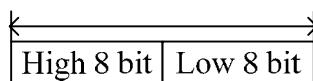
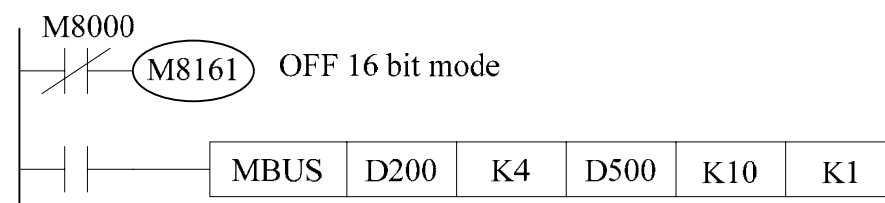
- Set the watchdog time for overtime.

The time = set value X10ms, efficient value for set is 5 ~ 255. D8329 will become 50ms as the set value exceeds the range.

For example: judging time for overtime is set to 50ms.



<16-bit data dealing> when M8161=OFF, (M8161 is special relay shared by RS, ASCI, HEX, CCD)



The 16-bit communication data will be divided to two 8-bit data. One is high 8 bits, the other is low 8 bits.

Sending data differs from ASCII mode and RTU mode

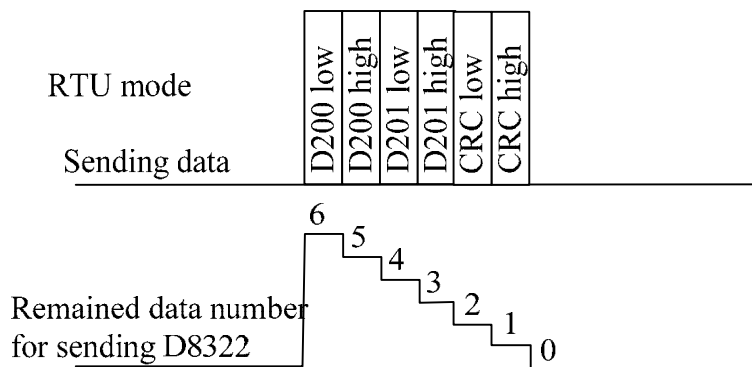
STX	D200 low	D200 high	D201low	D201high	Check code	ETX
Start character (3A)					LRC (ASCII)	End character (0D0A)
RTU mode (no)	↑  S . specifies start address M specifies bytes number to be sent				CRC (RTU)	RTU mode (no)

Receiving data

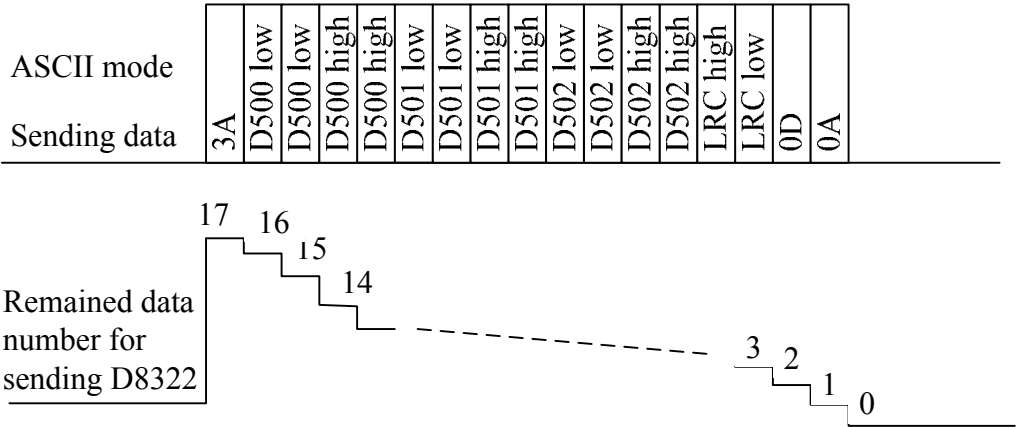
STX	D500 low	D500 high	D501 low	D501 high	D502 low	D502 high	Check code	ETX
Start character (3A)							LRC (ASCII)	End character (0D0A)
RTU (no)	↑ D . specifies start address Less than n, up limit point for the data to be received. End character EXT, or n will indicate receiving end.						CRC (RTU)	

(1) Sending data and remaining data to be send

**RTU mode**

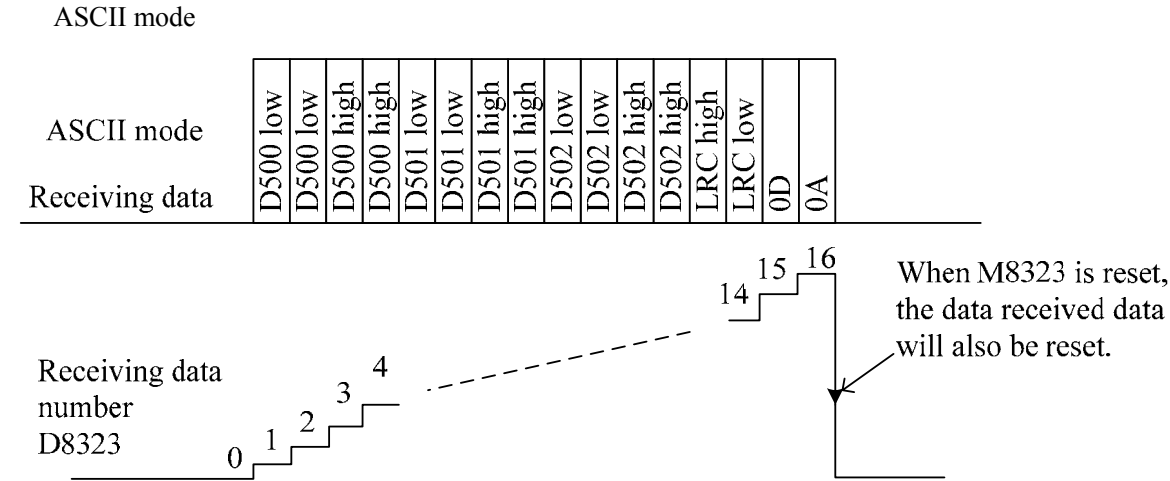
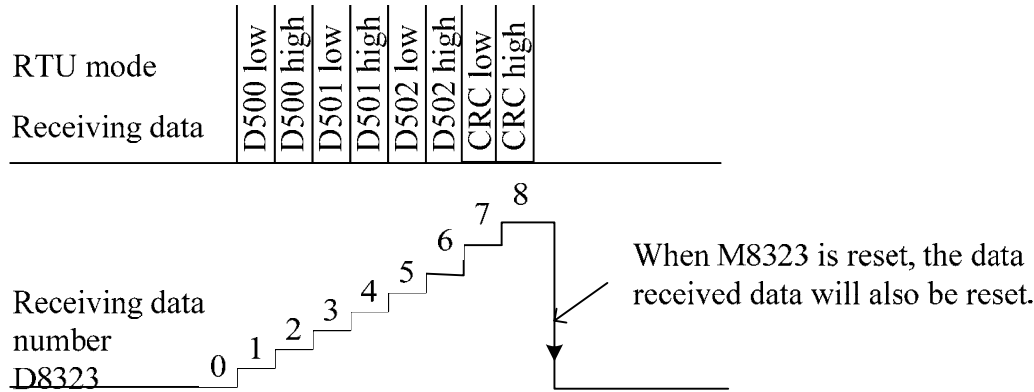


**ASCII mode**

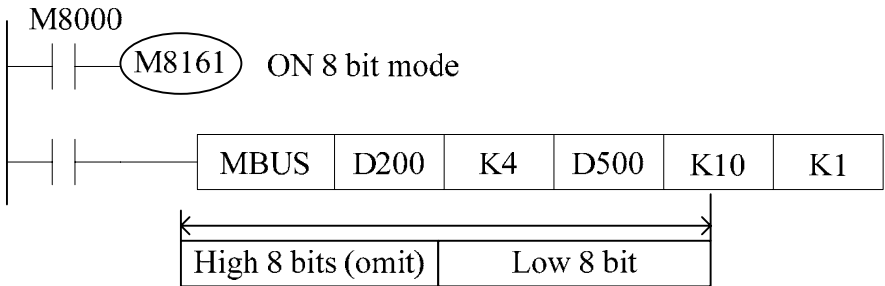


(2) Sending data and number of sending data

(3) RTU mode



<8 bit data dealing (expansion function)> M8161=ON (M8161 is special relay shared by RS,ASCII,HEX,CCD)



Only low 8 bits are significant

Sending data differ from ASCII mode and RTU mode

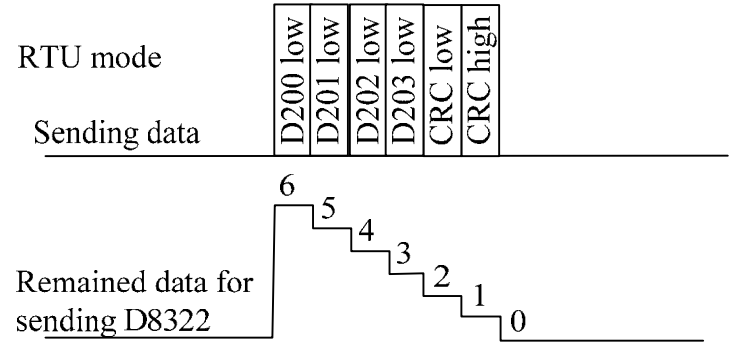
STX	D200 low	D201 low	D202 low	D203 low	Check code	ETX
Start character (3A)					LRC (ASCII)	End character (0D0A)
RTU mode (no)	S . specifies start address M specifies sending byte number				CRC (RTU)	RTU mode (no)

Receiving data

STX	D500 low	D501 low	D502 low	D503 low	D504 low	D505 low	Check code	ETX
Start character (3A)							LRC (ASCII)	End character (0D0A)
RTU (no)	D . specifies start address Less than n, up limit point for the data to be received. End character EXT, or n will indicate receiving end.						CRC (RTU)	

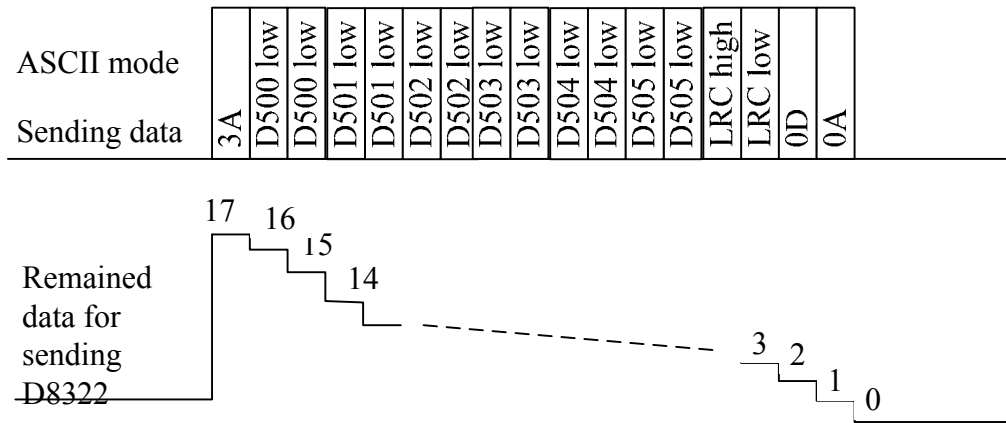
( 1 ) Receiving data and the number of the data remained

**RTU mode:**



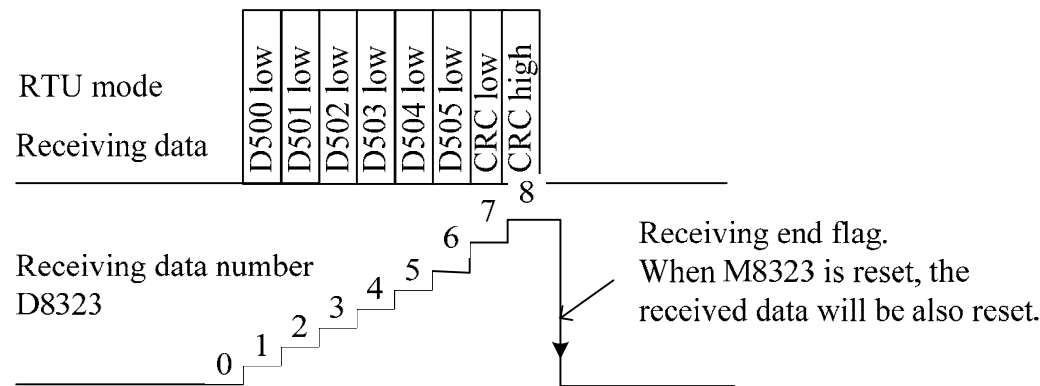
**ASCII mode:**



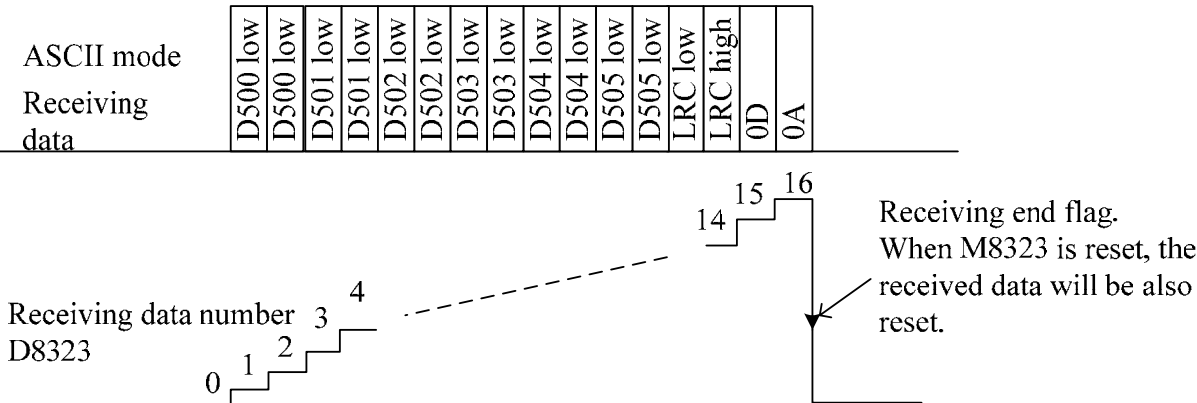


(2) Receiving data and the number of these data

**RTU mode:**



**ASCII mode:**



### 4.9.9 PID (FNC 88)

Mnemonic	Function	Operands				Program steps
		S1	S2	S3	D	
PID FNC 88 (PID control loop) register each	Receives a data input and calculates a corrective action to a specified level based on PID control	D		D S3:S3~S3+6	D	PID: 9 steps

#### Operation:

This instruction takes a current value (S2) and compares it to a predefined set value (S1).

The difference or error between the two

values is then processed through a PID loop to produce a correction factor which also takes into account previous iterations and trends of the calculated error. The PID process calculates a correction factor which is applied to the current output value and stored as a corrected output value in destination device (D). The setup parameters for the PID control loop are stored in 25 consecutive data registers S3+0 through S3+24.



#### Points to note:

- Every PID application is different. There will be a certain amount of “trial and error” necessary to set the variables at optimal levels.
- A Pre-tuning feature is available that can quickly provide initial values for the PID process.
- As 25 data register are required for the setup parameters for the PID loop, the head address of this data stack cannot be greater than D975. The contents of this data stack are explained later in this section. Multiple PID instructions can be programmed, however each PID loop must not have conflicting data registers.
- There are control limits in the PLC intended to help the PID controlled machines operate in a safe manner. If it becomes necessary to reset the Set Point Value (S1) during operation, it is recommended to turn the PID command Off and restore the command after entering the new Set Point Value. This will prevent the safety control limits from stopping the operation of the PID instruction prematurely.
- The PID instruction has a special set of error codes associated with it. Errors are identified in the normal manner. The error codes associated with the PID loop will be flagged by M8067 with the appropriate error code being stored in D8067. These error devices are not exclusive to the PID instruction so care should be taken to investigate errors properly. Please see chapter 6, ‘Diagnostic Devices’ for more information.
- A full PID iteration does not have to be performed. By manipulation of the setup parameters P (proportional), I (Integral) or D (derivative) loops may be accessed individually or in a user defined/selected group. This is detailed later in this section.

## PID Equations

Forward	$\Delta MV = Kp\{(EV_n - EV_{n-1}) + \frac{Ts}{Tl} EV_n + D_n\}$ $EV_n = PV_{nf} - SV$ $Dn = \frac{T_D}{Ts + \alpha_D T_D} (-2PV_{nf-1} + PV_{nf} + PV_{nf-2}) + \frac{\alpha_D T_D}{Ts + \alpha_D T_D} D_{n-1}$ $MV_n = \sum \Delta MV$
Reverse	$\Delta MV = Kp\{(EV_n - EV_{n-1}) + \frac{Ts}{Tl} EV_n + D_n\}$ $EV_n = SV - PV_{nf}$ $Dn = \frac{T_D}{Ts + \alpha_D T_D} (2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{\alpha_D T_D}{Ts + \alpha_D T_D} D_{n-1}$ $MV_n = \sum \Delta MV$

PVnf = PVn +  $\alpha$ (PVnf-1 - PVn)

EVn = the current Error Value

EVn-1 = the previous Error Value

SV = the Set Point Value (S1)

PVn = the current Process Value (S2)

PVnf = the calculated Process Value

PVnf-1 = the previous Process Value

PVnf-2 = the second previous Process Value

MV = the change in the Output

KD = the Derivative Filter Constant Manipulation Values

MVn = the current Output Manipulation Value (D)

Please see the Parameter setup section for a more detailed description of the variable parameters and in which memory register they must be set.

### Forward and Reverse operation (S3+1, b0)

The Forward operation is the condition where the Process Value, PVnf, is greater than the Set Point, SV. An example is a building that requires air conditioning. Without air conditioning, the temperature of the room will be higher than the Set Point so work is required to lower PVnf.

The Reverse operation is the condition where the Set Point is higher than the Process Value. An example of this is an oven. The temperature of the oven will be too low unless some work is done to raise it, i.e. - the heating element is turned on.

The assumption is made with PID control that some work will need to be performed to bring the system into balance. Therefore,  $\Delta MV$  will always have a value. Ideally, a system that is

Dn = the Derivative Value

Dn-1 = the previous Derivative Value

KP = the Proportion Constant

$\alpha$  = the Input Filter

TS = the Sampling Time

TI = the Integral Time Constant

TD = the Time Derivative Constant

stable will require a constant amount of work to keep the Set Point and Process Value equal.

### PID setup parameters; S3

The PID setup parameters are contained in a 25 register data stack. Some of these devices require data input from the user, some are reserved for the internal operation and some return output data from the PID operation.

**Parameters S3+0 through S3+6 must be set by the user.**

Parameter S3 + P	Parameter name/function	Description		Setting range
S3	Sampling time (Ts)	The time interval set between the reading the current Process Value of the system (PVnf)		1~32767[m s]
S3+1	Action - reaction direction and alarm control	BIT0	0:Forward operation 1:Reverse operation	Not applicable
		BIT1	Process Value(PVnf)alarm enable, OFF(0)/ON(1)	
		BIT2	Output Value (MV) alarm enable, OFF(0)/ON(1)	
		BIT3-15	Reserved	
S3+2	Input filter ( $\alpha$ )	Alters the effect of the input filter		0~99[%]
S3+3	Proportional gain (Kp)	This is a factor used to align the proportional output in a known magnitude to the change in the Process Value (PVnf). This is the <b>P</b> part of the PID loop.		1~32767[%]
S3+4	Integral time constant (TI)	This is the <b>I</b> part of the PID loop. This is the time taken for the corrective integral value to reach a magnitude equal to that applied by the proportional or <b>P</b> part of the loop. Selecting 0 (zero) for this parameter disables the <b>I</b> effect		1~32767[x1 00ms]
S3+5	Derivative gain (KD)	This is a factor used to align the derivative output in a known proportion to the change in the Process Value (PVnf)		0~100[%]
S3+6	Derivative time constant (TD)	This is the <b>D</b> part of the PID loop. This is the time taken for the corrective derivative value to reach a magnitude equal to that applied by the proportional or <b>P</b> part of the loop. Selecting 0 (zero) for this parameter disables the <b>D</b> effect.		1~32767[x1 00ms]
S3+7~S3+19	Reserved for use for the internal processing			
S3+20	Process Value, maximum positive change	Active when S3+1,b1 is set ON This is a user defined maximum limit for the Process Value (PVnf). If the Process Value (PVnf) exceeds the limit, S3+24, bit b0 is set On		0~32767
S3+21	Process Value, minimum value	Active when S3+1,b1 is set ON This is a user defined lower limit for the Process Value. If the Process Value (PVnf) falls below the limit,		

		S3+24, bit b1 is set On		
S3+22	Output Value, maximum positive change	Active when S3+1, b2 is set ON This is a user defined maximum limit for the quantity of positive change which can occur in one PID scan. If the Output Value (MV) exceeds this, S3+24, bit b2 is set On.		
S3+23	Output Value, Maximum negative change	Active when S3+1, b2 is set ON This is a user defined maximum limit for the quantity of negative change which can occur in one PID scan. If the Output Value (MV) falls below the lower limit, S3+24, bit b3 is set On.		
S3+24	Alarm flags <b>(Read Only)</b>	BIT0	High limit exceeded in Process Value (PVnf)	Not applicable
		BIT1	Below low limit for the Process Value (PVnf)	
		BIT2	Excessive positive change in Output Value (MV)	
		BIT3	Excessive negative change in Output Value (MV)	
		BIT4-15	Reserved	

### Configuring the PID loop

The PID loop can be configured to offer variations on PID control. These are as follows:

Control method	Selection via setup registers			Description
	S3 +3 (KP)	S3+ 4 (TI)	S3 + 6 (TD)	
P	User value	Set to 0 (zero)	Set to 0 (zero)	Proportional effect only
PI	User value	User value	Set to 0(zero)	Proportional and integral effect
PD	User value	Set to 0 (zero)	User value	Proportional and derivative effect
PID	User value	User value	User value	Full PID

It should be noted that in all situations there must be a proportional or 'P' element to the loop.

#### P - Proportional change

When a proportional factor is applied, it calculates the difference between the Current Error Value, EV<sub>n</sub>, and the Previous Error Value, EV<sub>n-1</sub>. The Proportional Change is based upon how fast the Process Value is moving closer to (or further away from) the Set Point Value NOT upon the actual difference between the PV<sub>nf</sub> and SV.

Note: Other PID systems might operate using an equation that calculates the Proportional change based upon the size of the Current Error Value only.

#### I - integral change

Once a proportional change has been applied to an error situation, 'fine tuning' the correction can be performed with the I or integral element. Initially only a small change is applied but as time increases and the error is not corrected the integral effect is increased. It is important to note how TI actually effects how fast the total integral correction is applied. The smaller TI is, the bigger effect the integral will have.

Note: The TI value is set in data register S3+4. Setting zero for this variable disables the

Integral effect.

### **The Derivative Change**

The derivative function supplements the effects caused by the proportional response. The derivative effect is the result of a calculation involving elements **T<sub>D</sub>**, **T<sub>S</sub>**, and the calculated error. This causes the derivative to initially output a large corrective action which dissipates rapidly over time. The speed of this dissipation can be controlled by the value **T<sub>D</sub>**: If the value of **T<sub>D</sub>** is small then the effect of applying derivative control is increased.

Because the initial effect of the derivative can be quite severe there is a 'softening' effect which can be applied through the use of **K<sub>D</sub>**, the derivative gain. The action of **K<sub>D</sub>** could be considered as a filter allowing the derivative response to be scaled between 0 and 100%. The phenomenon of chasing, or overcorrecting both too high and too low, is most often associated with the Derivative portion of the equation because of the large initial correction factor.

Note: The **T<sub>D</sub>** value is set in Data register S3+6. Setting zero for this variable disables the Derivative effect.

### **Effective use of the input filter $\alpha$ S3+2**

To prevent the PID instruction from reacting immediately and wildly to any errors on the Current Value, there is a filtering mechanism which allows the PID instruction to observe and account for any significant fluctuations over three samples.

The quantitative effect of the input filter is to calculate a filtered Input Value to the PID instruction taken from a defined percentage of the Current Value and the previous two filtered Input Values.

This type of filtering is often called first-order lag filter. It is particularly useful for removing the effects of high frequency noise which may appear on input signals received from sensors.

The greater the filter percentage is set the longer the lag time. When the input filter is set to zero, this effectively removes all filtering and allows the Current Value to be used directly as the Input Value.

### **Initial values for PID loops**

The PID instruction has many parameters which can be set and configured to the user's needs. The difficulty is to find a good point from which to start the fine tuning of the PID loop to the system requirements. The following suggestions will not be ideal for all situations and applications but will at least give users of the PID instruction a reasonable points from which to start.

A value should be given to all the variables listed below before turning the PID instruction ON. Values should be chosen so that the Output Manipulated Value does not exceed  $\pm 32767$ .

Recommended initial settings:

**T<sub>S</sub>** = Should be equal to the total program scan time or a multiple of that scan time, i.e. 2 times, 5 times, etc.

$\alpha$  = 50%

**K<sub>P</sub>** = This should be adjusted to a value dependent upon the maximum corrective action to

reach the set point - values should be experimented with from an arbitrary 75%

**T<sub>I</sub>** = This should ideally be 4 to 10 times greater than the **T<sub>D</sub>** time

**K<sub>D</sub>** = 50%

**T<sub>D</sub>** = This is set dependent upon the total system response, i.e. not only how fast the programmable controller reacts but also any valves, pumps or motors.

For a fast system reaction **T<sub>D</sub>** will be set to a quick or small time, this should however never be less than **T<sub>s</sub>**. A slower reacting system will require the **T<sub>D</sub>** duration to be longer. A beginning value can be **T<sub>D</sub>** twice the value of **T<sub>s</sub>**.

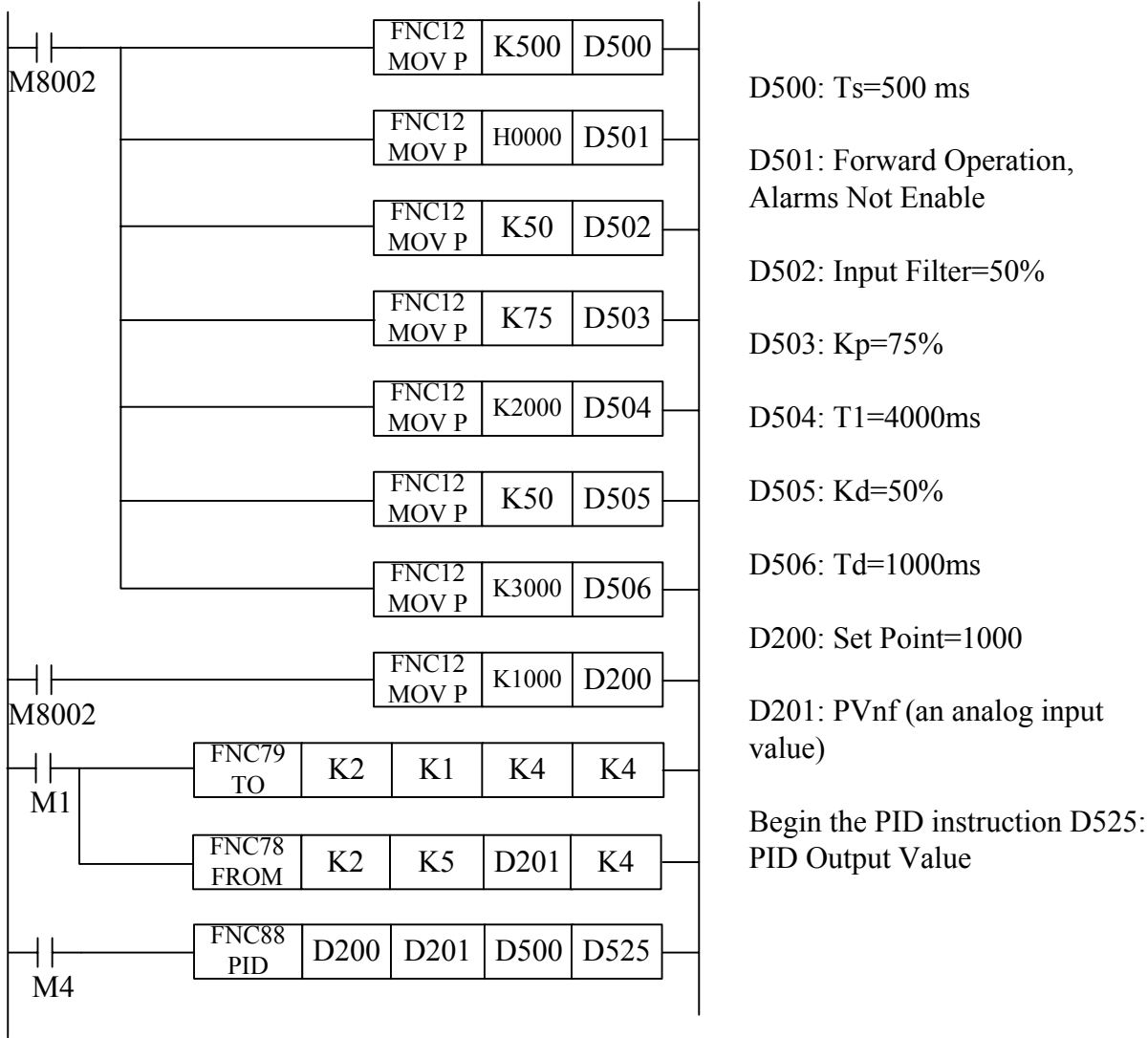
Care should be taken when adjusting PID variables to ensure the safety of the operator and avoid damage to the equipment.

**With ALL PID values there is a degree of experimentation required to tune the PID loop to the exact local conditions. A sensible approach to this is to adjust one parameter at a time by fixed percentages, i.e. say increasing (or decreasing) the K<sub>P</sub> value in steps of 10%. Selecting PID parameters without due consideration will result in a badly configured system which does not perform as required and will cause the user to become frustrated. Please remember the PID process is a purely mathematical calculation and as such has no regard for the 'quality' of the variable data supplied by the user/system - the PID will always process its PID mathematical function with the data available.**

### **Example PID Settings**

The partial program shown at below demonstrates which parameters must be set for the functioning of the FX2N. The first step sets the user values for S3+0 to S3+6. The PID instruction will be activated when M4 is On.

From the PID instruction at the bottom of the ladder, S1 = D200; S2 = D201; S3 = D500; and D or M=D525





## 4.10 Floating Point 1 & 2 - Functions 110 to 129

### Contents:

#### Floating Point 1

ECMP -	Float Compare	FNC 110
EZCP -	Float Zone Compare	FNC 111
-	Not Available	FNC 112 to 117
EBCD -	Float to Scientific	FNC 118
EBIN -	Scientific to Float	FNC 119

#### Floating Point 2

EADD -	Float Add	FNC 120
ESUB -	Float Subtract	FNC 121
EMUL -	Float Multiplication	FNC 122
EDIV -	Float Division	FNC 123
-	Not Available	FNC 124 to 126
ESQR -	Float Square Root	FNC 127
PPP -	Not Available	FNC 128
INT -	Float to Integer	FNC 129

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D1, S3 or for lists/tables devices D3+0, S+9 etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.

P - A 16 bit mode instruction modified to use pulse (single) operation.

D - An instruction modified to operate in 32 bit operation.

D P - A 32 bit mode instruction modified to use pulse (single) operation.

- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.

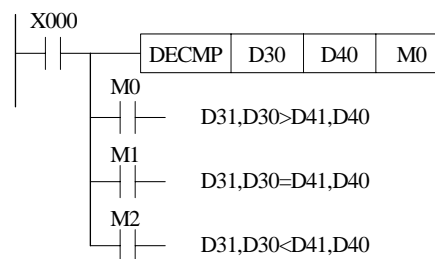
- An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

## 4.10.1 ECMP (FNC 110)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
ECMP FNC 110 (Floating Point Compare)	Compares two floating point values - results of <, = and > are given	K, H - integer value automatically converted to floating point D - must be in floating point format (32bits)		Y, M, S Note: 3 consecutive devices are used.	DECMP, DECMPP: 13 steps

**Operation:**

The data of S<sub>1</sub> is compared to the data of S<sub>2</sub>.  
 The result is indicated by 3 bit devices specified with the head address entered as D.  
 The bit devices indicate:  
 S<sub>2</sub> is less than < S<sub>1</sub> - bit device D is ON  
 S<sub>2</sub> is equal to = S<sub>1</sub> - bit device D+1 is ON  
 S<sub>2</sub> is greater than > S<sub>1</sub> - bit device D+2 is ON

**Points to note:**

The status of the destination devices will be kept even if the ECMP instruction is deactivated.

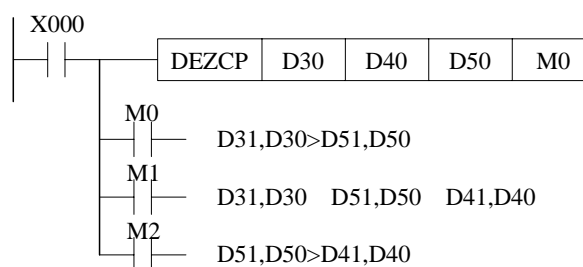
Full algebraic comparisons are used: i.e.  $-1.79 \times 10^{27}$  is smaller than  $9.43 \times 10^{-15}$

## 4.10.2 EZCP (FNC 111)

Mnemonic	Function	Operands				Program steps
		S1	S2	S3	D	
EZCP FNC 111 (Floating Point Zone Compare)	Compares a float range with a float value - results of <, = and > are given	K, H - integer value automatically converted to floating point D - must be in floating point format (32 bits). <b>Note:</b> S1 must be less than S2			Y, M, S Note: 3 consecutive devices are used	DEZCP, DEZCPP: 13 steps

**Operation:**

The operation is the same as the ECMP instruction except that a single data value (S<sub>3</sub>) is compared to a data range (S<sub>1</sub> - S<sub>2</sub>).  
 S<sub>3</sub> is less than S<sub>1</sub> and S<sub>2</sub> - bit device D is ON  
 S<sub>3</sub> is between S<sub>1</sub> and S<sub>2</sub> - bit device D+1 is ON  
 S<sub>3</sub> is greater than S<sub>2</sub> - bit device D+2 is ON



**4.10.3 EBCD (FNC 118)**

Mnemonic	Function	Operands		Program steps
		S	D	
EBCD FNC 118 (Float to Scientific conversion)	Converts floating point number format to scientific number format	D - must be in floating point format (32 bits).	D - 2 consecutive devices are used D - mantissa D+1 - exponent.	DEBCD, DEBCDP: 9 steps

**Operation:**

Converts a floating point value at S into separate mantissa and exponent parts at D and D+1 (scientific format).

**Points to note:**

- The instruction must be double word format. The destinations D and D+1 represent the mantissa and exponent of the floating point number respectively.
- To provide maximum accuracy in the conversion the mantissa D will be in the range 000 to 9999 (or 0) and the exponent D+1 corrected to an appropriate value.
- E.g.  $S = 3.4567 \times 10^{-5}$  will become  $D = 3456$ ,  $D+1 = -8$

**4.10.4 EBIN (FNC 119)**

Mnemonic	Function	Operands		Program steps
		S	D	
EBIN FNC 119 (Scientific to Float conversion)	Converts scientific number format to floating point number format	D - 2 consecutive devices are used S- mantissa S+1 - exponent	D - a floating point Value (32 bits).	DEBIN, DEBINP: 9 steps

**Operation:**

Generates a floating point number at D from scientific format data at source S.

**Points to note:**

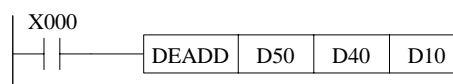
- The instruction must be double word format. The source data Sand S+1 represent the mantissa and exponent of the floating point number to be generated.
- To provide maximum accuracy in the conversion the mantissa S must be in the range 1000 to 9999 (or 0) and the exponent S+1 corrected to an appropriate value.
- E.g.  $S = 5432$ ,  $S+1 = 12$  will become  $D = 5.432 \times 10^9$

## 4.10.5 EADD (FNC 120)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
EADD FNC 120 (Floating Point Addition)	Adds two floating point numbers together	K, H - integer value automatically converted to floating point D - must be in floating point format (32 bits).		D - a floating point value (32 bits).	DEADD, DEADDP: 13 steps

**Operation:**

The floating point values stored in the source devices S1 and S2 are algebraically added and the result stored in the destination device D.

**Points to note:**

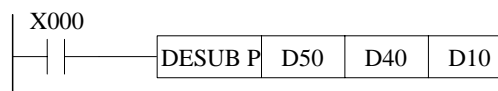
- The instruction must use the double word format; i.e., **DEADD** or **DEADDP**. All source data and destination data will be double word; i.e. uses two consecutive data registers to store the data (32 bits).  
Except for K or H, all source data will be regarded as being in floating point format and the result stored in the destination will also be in floating point format.
- If a constant K or H is used as source data, the value is converted to floating point before the addition operation.
- The addition is mathematically correct: i.e.,  $2.3456 \times 10^2 + (-5.6 \times 10^{-1}) = 2.34 \times 10^2$
- The same device may be used as a source and as the destination. If this is the case then, on continuous operation of the DEADD instruction, the result of the previous operation will be used as a new source value and a new result calculated.  
This will happen every program scan unless the pulse modifier or an interlock program is used.
- If the result of the calculation is zero "0" then the zero flag, M8020 is set ON.  
If the result of the calculation is larger than the largest floating point number then the carry flag, M8021 is set ON and the result is set to the largest value.  
If the result of the calculation is smaller than the smallest floating point number then the borrow flag, M8022 is set ON and the result is set to the smallest value.

**4.10.6 EAUB (FNC 121)**

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
ESUB FNC 121 (Floating Point)	Sub-traction) Subtracts one floating point number from another	K, H - integer value automatically converted to floating point D - must be in floating point number format (32 bits).		D - a floating point value(32 bits).	DESUB, DESUBP: 13 steps

**Operation:**

The floating point value of S2 is subtracted from the floating point value of S1 and the result stored in destination device D.

**Points to note:**

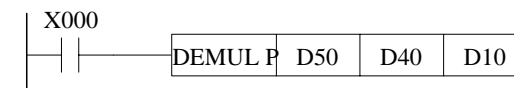
All points of the EADD instruction apply, except that a subtraction is performed.

**4.10.7 EMUL (FNC 122)**

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
EMUL FNC 122 (Floating Point Multiplication)	Multiplies two floating point numbers together	K, H - integer value automatically converted to floating point D - must be in floating point format (32 bits).		D - a floating point value (32 bits).	DEMUL, DEMULP: 13 steps

**Operation:**

The floating point value of S1 is multiplied with the floating point value of S2. The result of the multiplication is stored at D as a floating point value.

**Points to note:**

Point a, b, c and d of the EADD instruction apply, except that a multiplication is performed.

**4.10.8 EDIV (FNC 123)**

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
EDIV FNC 123 (Floating Point Division)	Divides one floating point number by another.	K, H - integer value automatically converted to floating point D - must be in floating point format (32 bits).		D - a floating point value (32 bits).	DEDIV, DEDIVP: 13 steps

**Operation:**

The floating point value of S1 is divided by the floating point value of S2. The result of the division is stored in D as a floating point value. No remainder is calculated.

**Points to note:**

Points a, b, c, d of the EADD instruction apply, except that a division is performed.

- If S2 is 0 (zero) then a divide by zero error occurs and the operation fails.

**4.10.9 ESQR (FNC 127)**

Mnemonic	Function	Operands		Program steps
		S	D	
ESQR FNC 127 (Floating Point Square Root)	Calculates the square root of a floating point value.	K, H - integer value automatically converted to floating point D - must be in floating point number format (32 bits).	D - a floating point value (32 bits).	DESQR, DESQRP: 9 steps

**Operation:**

A square root is performed on the floating point value of S and the result is stored in D.

**Points to note:**

Points a, b, c, d of the EADD instruction apply, except that a square root is performed.

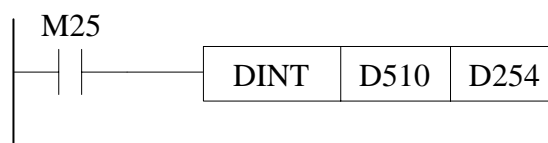
- If S is negative then an error occurs and error flag M8067 is set ON.

**4.10.10 INT (FNC 129)**

Mnemonic	Function	Operands		Program steps
		S	D	
INT FNC 129 (Float to Integer)	Converts a number from floating point format to decimal format	K, H - integer value automatically converted to floating point D - must be in floating point number format (32 bits).	D - decimal format for INT, INTP - 16 bits for DINT, DINTP - 32 bits	INT, INTP: 5 steps DINT, DINTP: 9 steps

**Operation:**

The floating point value of S is rounded down to the nearest integer value and stored in normal binary format in D.

**Points to note:**

- The source data is always a double (32 bit) word; a floating point value.  
For single word (16 bit) operation the destination is a 16 bit value.  
For double word (32 bit) operation the destination is a 32 bit value.
- This instruction is the inverse of the FLT instruction. (See page 5-49)
- If the result is 0 then the zero flag M8020 is set ON.  
If the source data is not a whole number it must be rounded down. In this case the borrow flag M8021 is set ON to indicate a rounded value.  
If the resulting integer value is outside the valid range for the destination device then an overflow occurs. In this case the carry flag M8022 is set on to indicate overflow.

**Note:** If overflow occurs, the value in the destination device will not be valid.

## 4.11 Trigonometry - FNC 130 to FNC 139

### Contents:

#### Floating point 3

SIN -	Sine	FNC 130
COS -	Cosine	FNC 131
TAN -	Tangent	FNC 132
ASIN -	ARC Sine	FNC 133
ACOS -	ARC Cosine	FNC 134
ATAN -	ARC Tangent	FNC 135
RAD -	Degree to Radian	FNC 136
DEG -	Radian to Degree	FNC 137
-	Not Available	FNC 138 to 139

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D1, S3 or for lists/tables devices D3+0, S+9 etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
- P - A 16 bit mode instruction modified to use pulse (single) operation.
- D - An instruction modified to operate in 32 bit operation.
- D P - A 32 bit mode instruction modified to use pulse (single) operation.
- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.
- An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.



## 4.11.1 SIN (FNC 130)

Mnemonic	Function	Operands		Program steps
		S	D	
SIN FNC 130 (Sine)	Calculates the sine of a floating point value	D - must be in floating point number format (32 bits).(radians)	D - a floating point value (32 bits).	DSIN, DSINP: 9 steps

**Contents:**

This instruction performs the mathematical SIN operation on the floating point value in S. The result is stored in D.

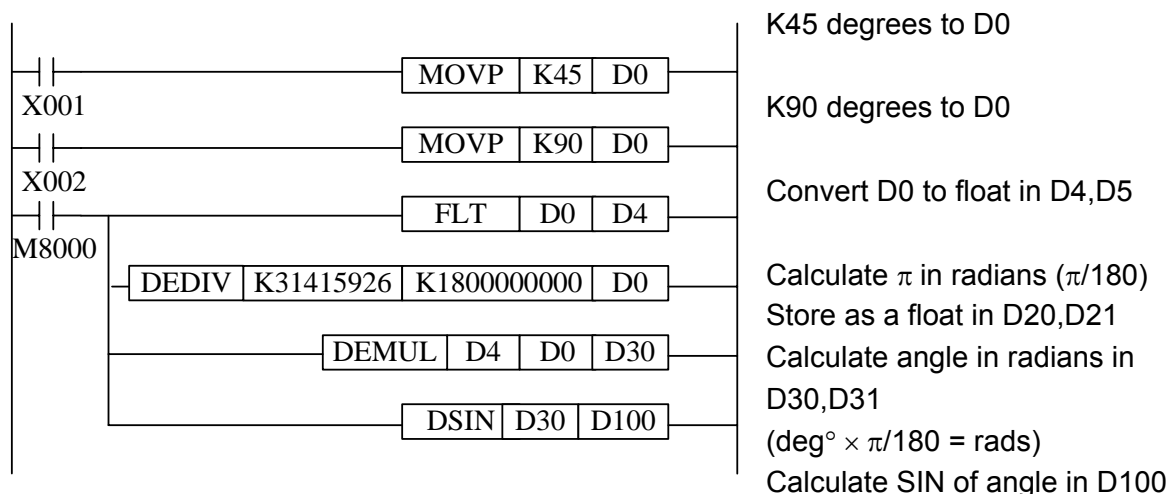
**Points to note:**

- a) The instruction must use the double word format: i.e., **DSIN** or **DSINP**. All source and destination data will be double word; i.e., uses two consecutive data registers to store the data (32 bits).

The source data is regarded as being in floating point format and the destination is also in floating point format.

**Radian Angles**

Below is an program example of how to calculate angles in radians using floating point.

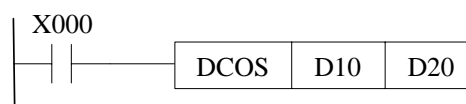


**4.11.2 COS (FNC 131)**

Mnemonic	Function	Operands		Program steps
		S	D	
COS FNC 131 (Cosine)	Calculates the cosine of a floating point value	D - must be in floating point number format (32 bits).	D - a floating point Value (32 bits).	DCOS, DCOSP: 9 steps

**Contents:**

This instruction performs the mathematical COS operation on the floating point value in S. The result is stored in D.

**Points to note:**

All the points for the SIN instruction apply, except that COS is calculated.

**4.11.3 TAN (FNC 132)**

Mnemonic	Function	Operands		Program steps
		S	D	
TAN FNC132 (Tangent)	Calculates the tangent of a floating point value	D - must be in floating point number format (32 bits).	D - a floating point Value (32 bits).	DTAN, DTANP: 9 steps

**Contents:**

This instruction performs the mathematical TAN operation on the floating point value in S. The result is stored in D.

**Points to note:**

All the points for the SIN instruction apply, except that COS is calculated.

## 4.11.4 ASIN (FNC 133)

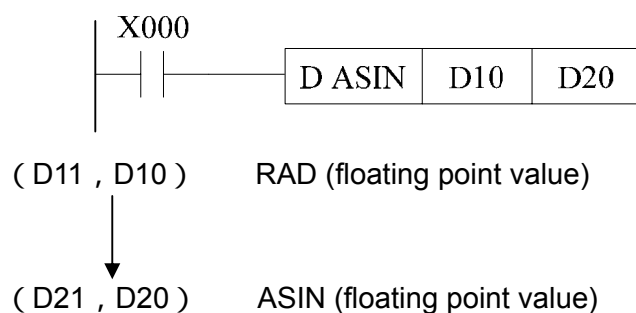
Mnemonic	Function	Operands		Program steps
		S	D	
ASIN FNC133	ARC SIN (floating point value)	D $-1 \leq S < 1$	D	DASIN, DASINP: 9 steps

**Contents:**

This instruction is ARC SIN ( inverse function of SIN ) the data in S, then send the result (floating point value) to D.



Example:



## 4.11.5 ACOS (FNC 134)

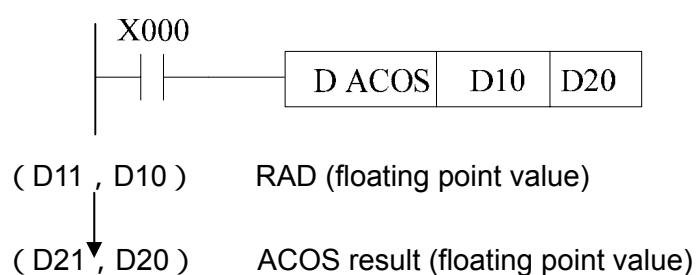
Mnemonic	Function	Operands		Program steps
		S	D	
ACOS FNC134	ARC COS (floating point value)	D $-1 \leq S < 1$	D	DACOS, DACOSP: 9 steps

**Contents:**

This instruction is ARC COS ( inverse function of COS ) the data in S, then send the result (floating point value) to D.



Example:



## 4.11.6 ATAN (FNC 135)

Mnemonic	Function	Operands		Program steps
		S	D	
ATAN FNC135	ARC TAN (floating point value)	D $-\pi/2 \sim \pi/2$	D	DATAN, DATANP: 9 steps

**Contents:**

This instruction is ARC TAN ( inverse function of TAN ) the data in S, then send the result (floating point value) to D.



Example:



( D11 , D10 )      RAD (floating point value)



( D21 , D20 )      ATAN result (floating point value)

## 4.11.7 RAD (FNC 136)

Mnemonic	Function	Operands		Program steps
		S	D	
RAD FNC136	Converter angle unit from DEG to RAD	D - a floating point Value	D - a floating point Value	DRAD DRADP: 9 steps

**Contents:**

This instruction converts angle unit from DEG to RAD



**4.11.8 DEG (FNC 137)**

Mnemonic	Function	Operands		Program steps
		S	D	
DEG FNC137	Converter angle unit from RAD to DEG	D - a floating point number format (32 bits).	D - a floating point number format (32 bits).	DDEG DDEGP: 9 steps

**Contents:**

This instruction converts angle unit from RAD to DEG



## 4.12 Data Operations 2 - FNC 140 to FNC 149

### Contents:

□□□ -	Not Available	FNC 140 to 146
SWAP -	Float to Scientific	FNC 147
□□□ -	Not Available	FNC 148 to 149

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant. Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D<sub>1</sub>, S<sub>3</sub> or for lists/tables devices D<sub>3+0</sub>, S<sub>+9</sub> etc.

MSB -Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

□□□ - An instruction operating in 16 bit mode, where □□□ identifies the instruction mnemonic.

□□□P - A 16 bit mode instruction modified to use pulse (single) operation.

D□□□ - An instruction modified to operate in 32 bit operation.

D□□□P - A 32 bit mode instruction modified to use pulse (single) operation.

★ - A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.

☆ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

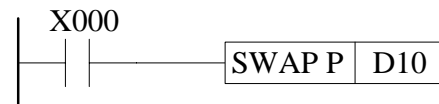
#### 4.12.1 SWAP (FNC 147)

Mnemonic	Function	Operands	Program steps
		S	
SWAP FNC 147 (Byte Swap) ★	The high and low byte of the Designated devices are exchanged	KnY, KnM, KnS, T, C, D, V, Z	SWAP,SWAPP: 5 steps DSWAP, DSWAPP: 9 steps

##### Contents:

The upper byte and the lower byte of the source device are swapped.

This instruction is equivalent to operation 2 of FNC 17 XCH



##### Points to note:

- In single word (16 bit) operation the upper and lower byte of the source device are exchanged.
- In double word (32 bit) operation the upper and lower byte of each or the two 16 bit devices are exchanged.

Result of DSWAP (P) D10:

Values are in Hex for clarity		Before DSWAP	After DSWAP
D10	Byte 1	1FH	8BH
	Byte 2	8BH	1FH
D11	Byte 1	C4H	35H
	Byte 2	35H	C4H

- If the operation of this instruction is allowed to execute each scan, then the value of the source device will swap back to its original value every other scan. The use of the pulse modifier or an interlock program is recommended.

#### 4.13 Position instruction – FNC 156 to FNC 159

##### Contents:

□□□ -	Not Available	FNC 150 to 155
ZRN -	reset to Zero point	FNC 156
PLSV -	variable speed pulse output	FNC 157
DRVI -	increment positioning	FNC 158
DRVA -	absolute positioning	FNC 159

##### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant. Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D<sub>1</sub>, S<sub>3</sub> or for lists/tables devices D<sub>3+0</sub>, S<sub>+9</sub> etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

##### Instruction modifications:

□□□ - An instruction operating in 16 bit mode, where □□□ identifies the instruction mnemonic.

□□□P - A 16 bit mode instruction modified to use pulse (single) operation.

D□□□ - An instruction modified to operate in 32 bit operation.

D□□□P - A 32 bit mode instruction modified to use pulse (single) operation.

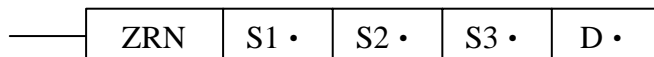
★ - A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.

☆ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.



#### 4.13.1 ZRN (FNC 156)

Mnemonic	Function	Operands				Program steps
		S1	S2	S3	D	
ZRN FNC 156	Return to zero-point after machine ON or initial setting.	K,H,KnY, KnM, KnS, T, C, D, V, Z		X,Y,M,S	Y	ZRN:9 steps DZRN:17 steps



- As for FNC158 (DRVI) and FNC159 (DRVA), PLC will control the present value to increase or decrease with the self-produced forward/reverse pulse and store the updated value to the register (Y000: [D8141, D8140], Y001: [D8143,D8142]). By these values, PLC will always know the machine position. However, when the power is turned off, the data will be lost. Consequently, in order to solve the problem, it is necessary to execute FNC156 (ZRN) as machine is ON or is initially set to store the zero point data to PLC.
  - Users may specify zero return speed [S1] as, 16-bit 10 to 32,767Hz or 32-bit 10 to 100kHz.
  - Users may specify the creep speed [S2] of 10 to 32,767Hz
  - If any device other than an input relay (X) is specified for the Near point signal [S3] it will be affected by the operation cycle of the PLC and the dispersion of the zero point may be large.
  - Only Y000 or Y001 can be used for the pulse output [D].

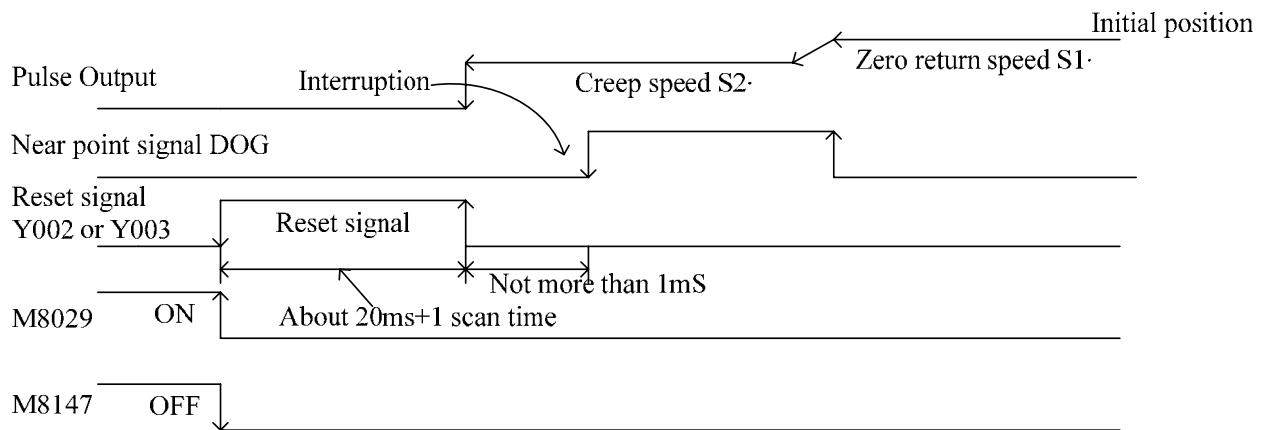
Output function of this instruction:

If M8140 is set to ON, the reset signal will be sent to the servo motor when the returning to zero point is completed.

The reset signal differs from various pulse output:

Pulse output [Y000]—> reset signal [Y002]

Pulse output [Y001]—> reset signal [Y003]



- The execution sequence for this instruction:
  - a) As the instruction is enabled, the machine will move at the speed S1 set.
    - In the zero point reset progress, the machine will stop as the enabling terminal is OFF.
    - If the enabling terminal is OFF and pulse output monitor device (Y000: M8147; Y001: M8148) is on, the machine will not accept such instruction.
  - b) When near signal (DOG) is ON from OFF, the machine will move at creeping speed S2.
  - c) AS near signal (DOG) is OFF from and pulse output stops, the data '0' will be written to present register (Y000: [D8141, D8140], Y001: [D8143, D8142]). When M8140 is ON, PLC will send out reset signal. After reset is finished, the M8029 will be ON, as well as pulse output monitor device (Y000: M8147; Y001: M8148) will be OFF.

Related device number:

D8141 (upper digit) & D8140 (lower digit): Current value register of Y000 (32-bit)

D8143 (upper digit) & D8142 (lower digit): Current value register of Y001 (32-bit)

M8145: Y000 pulse output stop (immediate)

M8146: Y001 pulse output stop (immediate)

M8147: Y000 pulse output monitor (BUS/READY)

M8148: Y001 pulse output monitor (BUS/READY)

Consideration:

Dog search function is not supported. Start zero return from the front side of the Near point signal.

In Zero point reset, the present value in register (Y000: [D8141, D8140], Y001: [D8143, D8142]) will decrease to 0.

Attention should be paid to the instruction drive timing.

#### 4.13.2 PLSV (FNC 157)

Mnemonic	Function	Operands			Program steps
		S	D1	D2	
PLSV FNC 157	Variable speed pulse output	K,H,KnY, KnM, KnS, T, C, D, V, Z	Y	X,Y,M	PLSY:7 steps DPLSY:13 steps

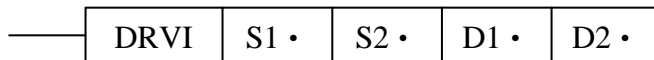
PLSV	S •	D1 •	D2 •
------	-----	------	------

- This is a variable speed output pulse instruction, with a rotation direction output.
  - a) Users may use output pulse frequencies [S1] of, 16-bit 1 to 32,767Hz/-1 to 32,767Hz or 32-bit 1 to 100kHz/-1 to 100kHz.
  - b) Only Y000 or Y001 can be used for the pulse output [D1].  
Because of the nature of the high speed output, transistor type output units should be used with this instruction. Relay type outputs will suffer a greatly reduced life, and will cause false outputs to occur.
  - c) Rotation direction signal output [D2] operated as follows: if [D2] = OFF, rotation = negative, if [D2] = ON, rotation = positive.
- The pulse frequency [S] can be changed even when pulses are being output.
- Acceleration/deceleration are not performed at start/stop. If cushion start/stop is required, increase or decrease the output pulse frequency [S] using the FNC67 RAMP instruction.
- If the instruction drive contact turns off while pulses are output, the machine will stop directly but not decelerate to 0.
- Related device numbers.
  - D8141 (upper digit) & D8140 (lower digit): Current value register of Y000 (32-bit)
  - D8143 (upper digit) & D8142 (lower digit): Current value register of Y001 (32-bit)
  - M8145 : Y000 pulse output stop (immediate)
  - M8146 : Y001 pulse output stop (immediate)
  - M8147 : Y000 pulse output monitor (BUS/READY)
  - M8148 : Y001 pulse output monitor (BUS/READY)

Attention should be paid to the instruction drive timing.

#### 4.13.3 DRVI (FNC 158)

Mnemonic	Function	Operands				Program steps
		S1	S2	D1	D2	
DRVI FNC 158	Increment positioning	K,H,KnY, KnM, KnS, T, C, D, V, Z		Y	Y,M, S	DRVI:9 steps DDRVI:17 steps



- This instruction is for single speed positioning in the form of incremental movements.
  - a) The maximum number of pulses [S1] available are: 16-bit -32,768 to 32,767 pulses or 32-bit. -2,147,483,648 to 2,147,483,648 pulses.
  - b) Users may use output pulse frequencies [S2], 16-bit 10 to 32,767Hz or 32-bit 10 to 100 kHz.
  - c) Only Y000 or Y001 can be used for the pulse output [D1].  
Because of the nature of the high speed output, transistor type output units should be used with this instruction. Relay type outputs will suffer a greatly reduced life, and will cause false outputs to occur.
  - d) Rotation direction signal output [D2] operated as follows: if [D2] = OFF, rotation = negative, if [D2] = ON, rotation = positive.
- Related Device:  
D8141 (upper digit) & D8140 (lower digit): Current value register of Y000 (32-bit)  
D8143 (upper digit) & D8142 (lower digit): Current value register of Y001 (32-bit)  
In reverse, the present value in register will decrease.
- If the contents of an operand are changed while the instruction is executed, it is not reflected on the operation. The new contents become effective when the instruction is next driven.
- If the instruction drive contact turns off while the instruction is being executed, the machine decelerates and stops. At this time the execution complete flag M8029 does not turn ON.
- Once the instruction drive contact is off, re-drive of the instruction is not possible while the pulse output flag (Y000 : [M8147] Y001 : [M8148]) is ON.
- For operation in the incremental drive method, the travel distance from the current position is specified with either a position or a negative symbol.
- The acceleration and deceleration time is set by D8148.

#### 4.13.4 DRVA (FNC 159)

Mnemonic	Function	Operands				Program steps
		S1	S2	D1	D2	
DRVA FNC 159	Absolute positioning	K,H,KnY, KnM, KnS, T, C, D, V, Z		Y	Y,M, S	DRVA:9 steps DDRVA:17 steps

—	DRVA	S1 •	S2 •	D1 •	D2 •
---	------	------	------	------	------

- This instruction is for single speed positioning using a zero home point and absolute measurements.
  - a) The target position for absolute positioning [S1] can be: 16-bit -32,768 to 32,767 pulses or 32-bit -2,147,483,648 to +2,147,483,647 pulses.
  - b) Users may use output pulse frequencies [S2], 16-bit 10 to 32,767Hz or 32-bit 10 to 100 kHz.
  - c) Only Y000 or Y001 can be used for the pulse output [D1].  
Because of the nature of the high speed output, transistor type output units should be used with this instruction. Relay type outputs will suffer a greatly reduced life, and will cause false outputs to occur.
  - d) Rotation direction signal output [D2] operated as follows: if [D2] = OFF, rotation = negative, if [D2] = ON, rotation = positive.
- Related Device:
  - D8141 (upper digit) & D8140 (lower digit): Current value register of Y000 (32-bit)
  - D8143 (upper digit) & D8142 (lower digit): Current value register of Y001 (32-bit)

In reverse, the present value in register will decrease.

- If the contents of an operand are changed while the instruction is executed, it is not reflected on the operation. The new contents become effective when the instruction is next driven.
- If the instruction drive contact turns off while the instruction is being executed, the machine decelerates and stops. At this time the execution complete flag M8029 does not turn ON.
- Once the instruction drive contact is off, re-drive of the instruction is not possible while the pulse output flag (Y000 : [M8147] Y001 : [M8148]) is ON.
- For operation in the incremental drive method, the travel distance from the current position is specified with either a position or a negative symbol.
- The acceleration and deceleration time is set by D8148.

## 4.14 Real Time Clock Control 160 to 169

### Contents:

TCMP -	Time Compare	FNC 160
TZCP -	Time Zone Compare	FNC 161
TADD -	Time Add	FNC 162
TSUB -	Time Subtract	FNC 163
□□□ -	Not Available	FNC 164 to 165
TRD -	Read RTC data	FNC 166
TWR -	Set RTC data	FNC 167
□□□ -	Not Available	FNC 168 to 169

### Symbols list:

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D<sub>1</sub>, S<sub>3</sub> or for lists/abled devices D<sub>3+0</sub>, S<sub>+9</sub> etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

### Instruction modifications:

□□□ - An instruction operating in 16 bit mode, where □□□ identifies the instruction mnemonic.

□□□P - A 16 bit mode instruction modified to use pulse (single) operation.

D□□□ - An instruction modified to operate in 32 bit operation.

D□□□P - A 32 bit mode instruction modified to use pulse (single) operation.

★ - A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.

☆ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

#### 4.14.1 TCMP (FNC 160)

Mnemonic	Function	Operands					Program steps
		S1	S2	S3	S	D	
TCMP FNC 160 (Time Compare)	Compares two times - results of <, = and > are given	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z			T, C, D	Y, M, S	TCMP, TCMP: 11 steps

##### Contents:

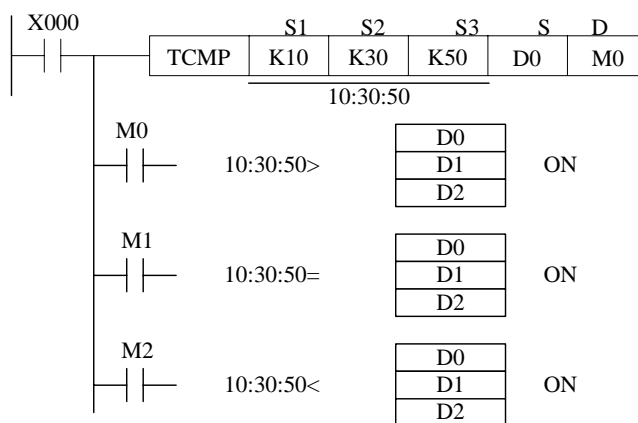
S1, S2 and S3 represent hours, minutes and seconds respectively. This time is compared to the time value in the 3 data devices specified by the head address S. The result is indicated in the 3 bit devices specified by the head address D.

The bit devices in D indicate the following:

D+0 is set ON, when the time in S is less than the time in S1, S2 and S3.

D+1 is set ON, when the time in S is equal to the time in S1, S2 and S3.

D+2 is set ON, when the time in S is greater than the time in S1, S2 and S3.



##### Points to note:

- The status of the destination devices is kept, even if the TCMP instruction is deactivated.
- The comparison is based on the time value specified in the source devices.
  - The valid range of values for S1 and S+0 is 0 to 23 (Hours).
  - The valid range of values for S2 and S+1 is 0 to 59 (Minutes).
  - The valid range of values for S3 and S+2 is 0 to 59 (Seconds).
- The current time of the real time clock can be compared by specifying D8015 (Hours), D8014 (Minutes) and D8013 (Seconds) as the devices for S1, S2 and S3 respectively.

#### 4.14.2 TZCP (FNC 161)

Mnemonic	Function	Operands				Program steps
		S1	S2	S	D	
TZCP FNC 161 (Time Zone Compare)	Compares a time to a specified time range - results of <, = and > are given	T, C, D S1 must be less than or equal to S2. Note: 3 consecutive devices are used for all			Y, M, S	TZCP, TZCPP: 9 steps

##### Contents:

S1, S2 and S represent time values. Each specifying the head address of 3 data devices.

S is compared to the time period defined by S1 and S2.

The result is indicated in the 3 bit devices specified by the head address D.

The bit devices in D indicate the following:

D+0 is set ON, when the time in

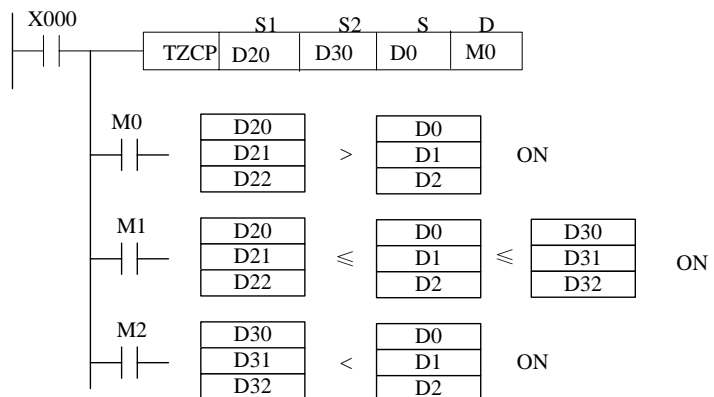
S is less than the times in S1 and S2.

D+1 is set ON, when the time in S is between the times in S1 and S2.

D+2 is set ON, when the time in S is greater than the times in S1 and S2.

##### Points to note:

- The status of the destination devices is kept, even if the TCMP instruction is deactivated.
- The comparison is based on the time value specified in the source devices.
  - The valid range of values for S1 and S+0 is 0 to 23 (Hours).
  - The valid range of values for S2 and S+1 is 0 to 59 (Minutes).
  - The valid range of values for S3 and S+2 is 0 to 59 (Seconds).





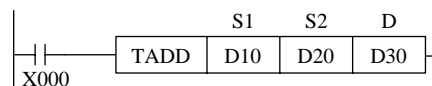
#### 4.14.3 TADD (FNC 162)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
TADD FNC 162 (Time Addition)	Adds two time values together to give a new time	T, C, D Note: 3 consecutive devices are used to represent hours, minutes and seconds respectively.			TADD, TADDP: 7 steps

##### Contents:

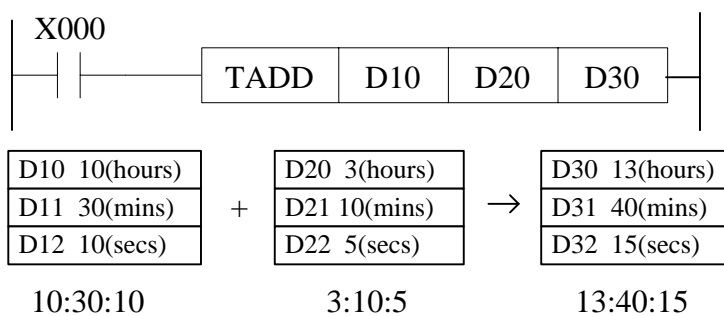
Each of S<sub>1</sub>, S<sub>2</sub> and D specify the head address of 3 data devices to be used a time value.

The time value in S<sub>1</sub> is added to the time value in S<sub>2</sub>, the result is stored to D as a new time value.

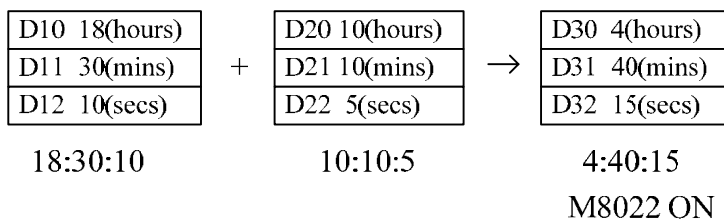


##### Points to note:

- a) The addition is performed according to standard time values. Hours, minutes and seconds are kept within correct limits. Any overflow is correctly processed.



- b) If the addition of the two times results in a value greater than 24 hours, the value of the result is the time remaining above 24 hours.



When this happens the carry flag M8022 is set ON.

- c) If the addition of the two times results in a value of zero (0:00:00: 0 hours, 0 minutes, 0 seconds) then the zero flag M8020 is set ON.
- d) The same device may be used as a source (S<sub>1</sub> or S<sub>2</sub>) and destination device. In this case the addition is continually executed; the destination value changing each program scan. To prevent this from happening, use the pulse modifier or an interlock program.

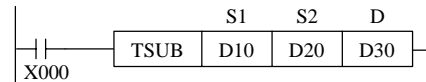
#### 4.14.4 TSUB (FNC 163)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
TSUB FNC 163 (Time Subtraction)	Subtracts onetime value from another to give a new time	T, C, D Note: 3 consecutive devices are used.			TSUB, TSUBP: 7 steps

##### Contents:

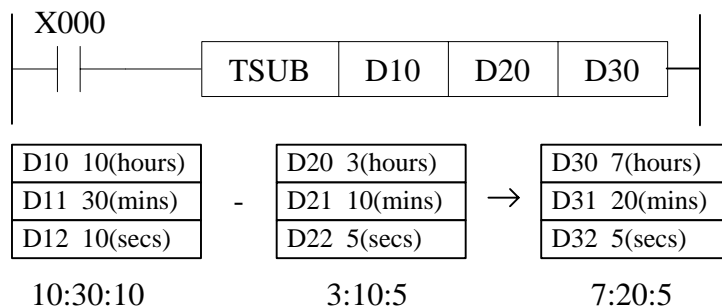
Each of S1, S2 and D specify the head address of 3 data devices to be used a time value.

The time value in S1 is subtracted from the time value in S2, the result is stored to D as a new time value.

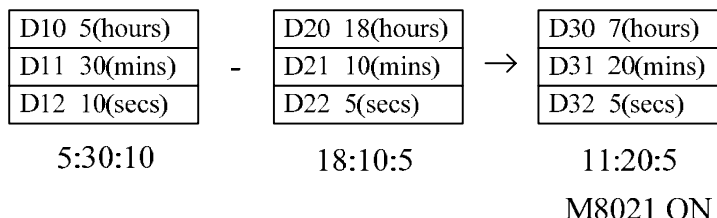


##### Points to note:

- a) The subtraction is performed according to standard time values. Hours, minutes and seconds are kept within correct limits. Any underflow is correctly processed.



- b) If the subtraction of the two times results in a value less than 00:00:00 hours, the value of the result is the time remaining below 00:00:00 hours.



When this happens the borrow flag M8021 is set ON.

- c) If the subtraction of the two times results in a value of zero (00:00:00 hours) then the zero flag M8020 is set ON.
- d) The same device may be used as a source (S1 or S2) and destination device. In this case the subtraction is continually executed; the destination value changing each program scan.

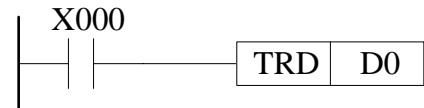
To prevent this from happening, use the pulse modifier or an interlock program.

#### 4.14.5 TRD (FNC 166)

Mnemonic	Function	Operands	Program steps
		D	
TRD FNC 166 (Time Read)	Reads the current value of the real time clock to a group of registers	T, C, D Note: 7 consecutive devices are used	TRD, TRDP: 5 steps

##### Contents:

The current time and date of the real time clock are read and stored in the 7 data devices specified by the head address D.



The 7 devices are set as follows:

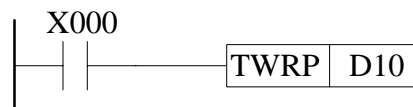
Device	Meaning	value		Device	Meaning
D8018	Year	2000~2099	→	D0	Year
D8017	Month	1~12	→	D1	Month
D8016	Date	1~31	→	D2	Date
D8015	Hours	0~23	→	D3	Hours
D8014	Minutes	0~59	→	D4	Minutes
D8013	Seconds	0~59	→	D5	Seconds
D8019	Day	0(Sun)~6(Sat)	→	D6	Day

#### 4.14.6 TWR (FNC 167)

Mnemonic	Function	Operands	Program steps
		S	
TWR FNC 167 (Time Write)	Sets the real time clock to the value stored in a group of registers	T, C, D Note: 7 consecutive devices are used.	TWR, TWRP: 5 steps

##### Contents:

The 7 data devices specified with the head address S are used to set a new current value of the real time clock.



##### The seven devices

Device	Meaning	Values		Device	Meaning
D10	Year	0~99	→	D8018	Year
D11	Month	1~12	→	D8017	Month
D12	Date	1~31	→	D8016	Date
D13	Hours	0~23	→	D8015	Hours
D14	Minutes	0~59	→	D8014	Minutes
D15	Seconds	0~59	→	D8013	Seconds
D16	Day	0(Sun)~6(Sat)	→	D8019	Day

##### Points to note:

This instruction removes the need to use M8015 during real time clock setting. When setting the time it is a good idea to set the source data to a time a number of minutes ahead and then drive the instruction when the real time reaches this value.

#### 4.15 Gray Codes - FNC 170 to FNC 179

**Contents:**

GRY -	Decimal to Gray Code	FNC 170
GBIN -	Gray Code to Decimal	FNC 171
-	Not Available	FNC 172 to 177

**Symbols list:**

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D<sub>1</sub>, S<sub>3</sub> or for lists/abled devices D<sub>3+0</sub>, S<sub>+9</sub> etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

**Instruction modifications:**

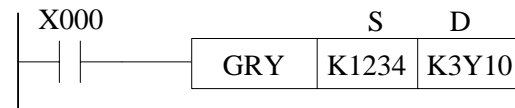
- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
  - P - A 16 bit mode instruction modified to use pulse (single) operation.
  - D - An instruction modified to operate in 32 bit operation.
  - D P - A 32 bit mode instruction modified to use pulse (single) operation.
  - A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.
  - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.
-

## 4.15.1 GRY (FNC 170)

Mnemonic	Function	Operands		Program steps
		S	D	
GRY FNC 170 (Gray Code)	Calculates the gray code value of an integer	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D, V, Z	GRY,GRYP: 5 steps DGRY,DGRYP: 9 steps

**Operation:**

The binary integer value in S is converted to the GRAY CODE equivalent and stored at D.

**Points to Note:**

The nature of gray code numbers allows numeric values to be quickly output without the need for a strobing signal. For example, if the source data is continually incremented, the new output data can be set each program scan.

## 4.15.2 GBIN (FNC 171)

Mnemonic	Function	Operands		Program steps
		S	D	
GBIN FNC 171 (Gray Code)	Calculates the integer value of a gray code	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z	KnY, KnM, KnS, T, C, D, V, Z	GBIN,GBINP: 5 steps DGBIN,DGBINP: 9 steps

**Operation:**

The GRAY CODE value in S is converted to the normal binary equivalent and stored at D.

**Points to Note:**

This instruction can be used to read the value from a gray code encoder.

If the source is set to inputs X0 to X17 it is possible to speed up the reading time by adjusting the refresh filter with FNC 51 REFF.

#### 4.16 Communication Codes - FNC 190 to FNC 199

**Contents:**

DTLK -	Data Link	FNC 190
RMIO -	Remote IO	FNC 191
TEXT	OP07/08 TEXT	FNC 192
-	Not Available	FNC 193 to 199

**Symbols list:**

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D1, S3 or for lists/tables devices D3+0, S+9 etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

**Instruction modifications:**

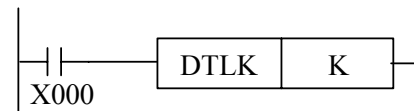
- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
  - P - A 16 bit mode instruction modified to use pulse (single) operation.
  - D - An instruction modified to operate in 32 bit operation.
  - D P - A 32 bit mode instruction modified to use pulse (single) operation.
  - A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.
  - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.
-

## 4.16.1 DTLK (FNC 190)

Mnemonic	Function	Operands	Program steps
		K	
DTLK FNC 190 (Data link)	Setup a small network which enables PLC controlling other 15 PLCs.	K,H:0,1 0: for built in RS485 port ; 1: for RS485 or RS232 expansion card	3 steps

**Operation:**

This instruction F190 DTLK used by PLC can setup a small network which enables PLC controlling other 15 PLC.

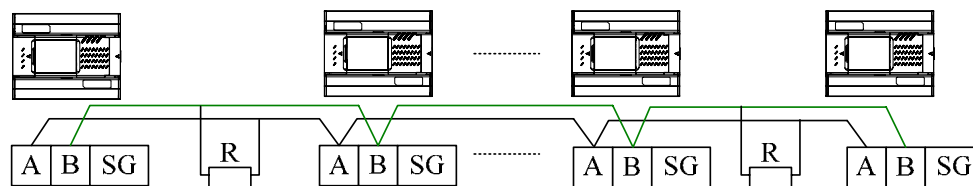


While two communication ports are ready for DTLK, only one firstly enabled is available. Communication frame and baud rate is set through D8120 or D8320, which is controlled by the different port.

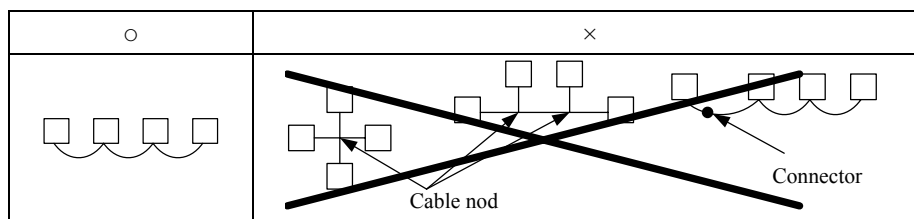
Both the port RS485/ RS232 expansion card (all type is available for expansion) , RS485 port (only built-in port in H type) are available for Data Link. However, both of them can not be enabled simultaneously.

Item	Specification
Communication standard	EIA RS-485
Baud rate	9600bps ~ 307200bps
Number of slaves	Max 15 slaves
Related devices	D0 ~ D157 , M2000 ~ M3023
Data length for each slave	Max 64 bits+8 word
Communication cable	Insulated twisted cable, 2 lines type, Total length: 500m (76800bit/s), 1km(38400bit/s)



**Wiring:**

- Note 1: SHL terminal should be 3 class ground or the production will be interrupted to error operation because of noise.
- Note 2: Branch of communication cable should not exceed 3.
- Note 3: R represents terminal resistor (120Ω, 1/4W).

**Related devices:**

## 1 ) Special relays

Special relays	Feature	Function	Description	Respond from
M8400	Read-only	Master error	The relay will be on as master is error.	L
M8401	Read-only	Slave 1 error	The relay will be on as slave 1 is error.	M/L
M8402	Read-only	Slave 2 error	The relay will be on as slave 2 is error.	M/L
...	...	...	...	..
M8414	Read-only	Slave 14 error	The relay will be on as slave 14 is error.	M/L
M8415	Read-only	Slave 15 error	The relay will be on as slave 15 is error.	M/L
M8416	Read-only	state	The relay will be on as DTLK is enabled.	M/L
M8417	Read-only	Data Link mode	The relay will be on as expansion card is in Data Link.	M/L
M8418	Read-only	Data Link mode	The relay will be on as RS485 port is in Data Link.	M/L

## 2 ) Data register

Special relays	Feature	Function	Description	Respond from
D8173	Read-only	Address number	Saving its own address number	M/L
D8174	Read-only	The number of slaves	Saving the number of slaves	M/L
D8175	Read-only	Refreshing range	Saving refreshing range (Data Link)	M/L
D8176	Write	Slave address setting	Setting its own address number	M/L
D8177	Write	Slavers number setting	Setting the number of slaves	M
D8178	Write	Data Link setting	Setting refreshing range (Data Link)	M
D8179	Read/write	Retry times	Setting retry times	M
D8180	Read/write	Time-out setting	Setting communication time-out ( Time-Out )	M
D8401	Read-only	Current communication scan time	Saving current communication scan time	M/L
D8402	Read-only	Max communication scan time	Saving Max communication scan time	M/L
D8403	Read-only	Error times for master	Error times for master	L
D8404	Read-only	Error times for slave 1	Error times for slave 1	M/L
D8405	Read-only	Error times for slave 2	Error times for slave 2	M/L
...	...	...	...	..
D8411	Read-only	Error times for slave 8	Error times for slave 8	M/L
...	...	...	...	..
D8417	Read-only	Error times for slave 14	Error times for slave 14	M/L
D8418	Read-only	Error times for slave 15	Error times for slave 15	M/L
D8419	Read-only	Error code for master	Error code for master	L
D8420	Read-only	Error code for slave 1	Error code for slave 1	M/L
D8421	Read-only	Error code for slave 2	Error code for slave 2	M/L
...	...	...	...	..
D8427	Read-only	Error code for slave 8	Error code for slave 8	M/L
...	...	...	...	..
D8433	Read-only	Error code for slave 14	Error code for slave 14	M/L
D8434	Read-only	Error code for slave 15	Error code for slave 15	M/L

**Setting:**

When the program is in operation, or TP 03 is power ON, all the setting for Data Link will take effect.

## 1 ) Setting the slaver address (D8176)

Set 0 ~ 15 to the special data register D8176, 0 is for master, and 1 ~ 15 is for slave.

## 2 ) Setting the slavers number (D8177)

Set 1 ~ 15 to the special data register D8177(default: 7). It is unnecessary for slavers

The slavers number should be set according to different condition in order to raise the refreshing speed.

## 3 ) Setting the refresh range (D8178)

Set 0 ~ 2 to special data register D8178 (default: 0). It is unnecessary for slaves.

D8178		0	1	2
Data Link mode		Mode 0	Mode 1	Mode 2
Refreshing range	Bit device (M)	0 point	32 point	64 point
	Word device (D)	4 point	4 point	8 point

The devices to be refreshed under different mode:

Address	Mode 0		Mode 1		Mode 2	
	(M)	( D )	( M )	( D )	( M )	( D )
No 0	-	D0~D3	M2000~M2031	D0~D3	M2000~M2063	D0~D7
No 1	-	D10~D13	M2064~M2095	D10~D13	M2064~M2127	D10~D17
No 2	-	D20~D23	M2128~M2159	D20~D23	M2128~M2191	D20~D27
No 3	-	D30~D33	M2192~M2223	D30~D33	M2192~M2255	D30~D37
No 4	-	D40~D43	M2256~M2287	D40~D43	M2256~M2319	D40~D47
No 5	-	D50~D53	M2320~M2351	D50~D53	M2320~M2383	D50~D57
No 6	-	D60~D63	M2384~M2415	D60~D63	M2384~M2447	D60~D67
No 7	-	D70~D73	M2448~M2479	D70~D73	M2448~M2511	D70~D77
No 8	-	D80~D83	M2512~M2543	D80~D83	M2512~M2575	D80~D87
No 9	-	D90~D93	M2576~M2607	D90~D93	M2576~M2639	D90~D97
No A	-	D100~D103	M2640~M2671	D100~D103	M2640~M2703	D100~D107
No B	-	D110~D113	M2704~M2735	D110~D113	M2704~M2767	D110~D117
No C	-	D120~D123	M2768~M2799	D120~D123	M2768~M2831	D120~D127
No D	-	D130~D133	M2832~M2863	D130~D133	M2832~M2895	D130~D137
No E	-	D140~D143	M2896~M2927	D140~D143	M2896~M2959	D140~D147
No F	-	D150~D153	M2960~M2991	D150~D153	M2960~M3023	D150~D157

## 4 ) setting retry times (D8179)

Set 0 ~ 10 to special data register D8179 (default: 3). It is unnecessary for slaves. If the master retry communication with the slave for more than the set times, the slave will be in communication error.

## 5 ) setting time out (D8180)

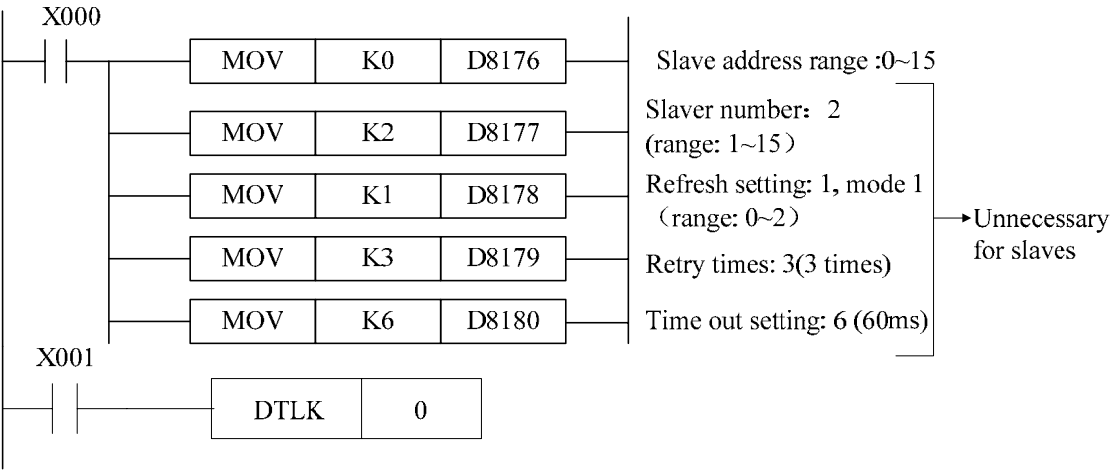
Set 5 ~ 255 to special data register D8180 (default: 5), the product of such value and 10 is the waiting time for communication time out (ms).

## 6 ) Current communication scan time (D8401)

The product of such value and 10 is the current communication scan time (ms).

## 7 ) Max communication scan time (D8402)

The example program for setting the said devices:



**Error code:**

When there is error, the special relays M8400 ~ M8415 will indicate the error condition and the error code will be stored in special data registers (D8419 ~ D8434).

Error code	Error	Error address	Check address	Description	Check point
01H	Communication time out error	L	M	There is no responding as the master sends the request to slave and time out.	Wiring, power supply and run/stop state
02H	Communication number error	L	M	Address is not set according to the certain relations between master and slave	Wiring
03H	Communication counting error	L	M	The data in communication counter does not conform to according to the certain relations between master and slave	Wiring
04H	Communication frame error	L	M, L	Communication frame of slave is error	Wiring and DTLK setting
11H	Communication over time error	M	L	After the slave responses to master, the master does not send another request to slavers.	Wiring, power supply and run/stop state
14H	Communication frame error	M	L	Communication frame of master is error	Wiring and DTLK setting
21H	Without slave	L	L *1	Address in the net is wrong	Address setting
22H	Address error	L	L *1	Slave address does not comply with the certain relations between master and slave	Wiring
23H	Communication counting error	L	L *1	The data in communication counter does not conform to according to the certain relations between master and slave	Wiring
31H	Receiving communication parameter error	L	L *2	Master send request before the slave accepts the set parameter.	Wiring, power supply and run/stop state
32H	Other error	L	L *1	Communication instruction error	Net setting

M: master

L: slave

\*1: another slave

\*2: Individual slave

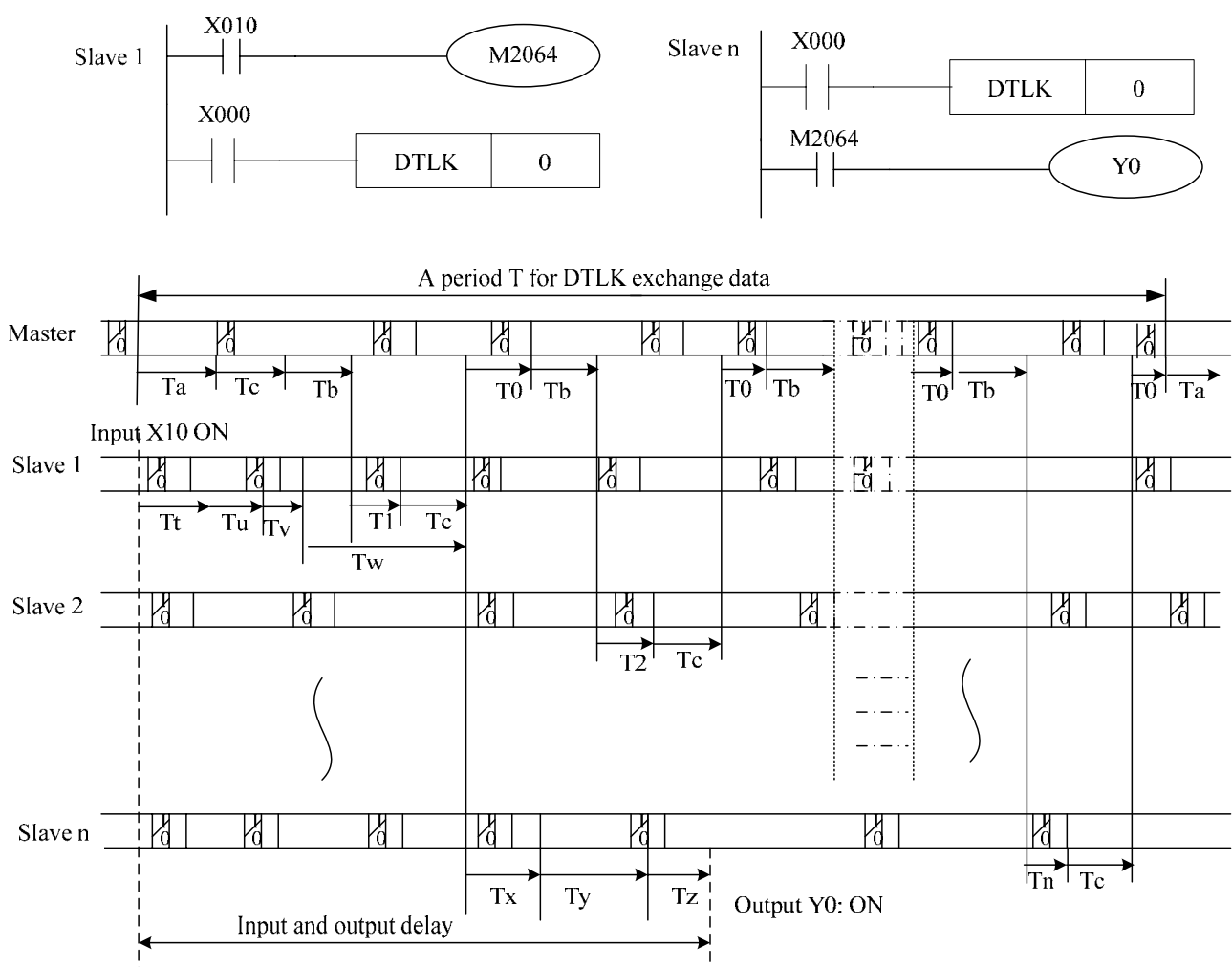
### Communication Timing Sequence and the Time Required for Transmission

- The communication for master-station and slave-stations is not synchronous with the scanning cycle of master-station.
- The master station will perform the linked data exchange and update the communication flag at the scan cycle after the communication completed.

Communication timing sequence diagram and communication delay diagram.

In Data Link net, there will be delay for receiving data. Please refer to following figure for communication timing sequence:

For example: the M2064 for slave 1 is controlled by X010. The state of M2064 will be sent to other node of the net as the instruction DTLK is enabled.



The time required to complete transmission

In data-link mode, the time T required for the master-station to complete communication with all slave-station can be devised as follows (not spend the SCAN TIME of master-station):

$$T = T_a + T_c + [T_b + T_n + T_c + T_0] \cdot n_1 \{ + [T_b + T_n + D8180 \cdot 10] \cdot n_2 \};$$

$T_a$ : the transmission time for master sending instruction for net configuration to

slave.

$T_b$ : the transmission time for master sending instruction for data-exchange to slave.

$T_c$ : the transmission time for the net exchanging data (differs from different DTLK mode).

$T_0$ : the time for master detecting communication states (0~1 SCAN TIME)

$T_n$ : the time for slave detecting communication states (0~1 SCAN TIME)

( $n1+n2$ ): the number of DTLK slave set in master (D8177=1~15),  $n1$ : actual slave number,  $n2$ : the number of the slave which is not recognized by master (0~15).

D8180 is timeout value.

Delay time:

$T_u$ : the time required for PLC to detect the input status (max. 1 SCAN TIME)

$T_v$ : the time between the PLC received input state and program started to be scanned.

$T_w$ : the time for operation result send out (max net scan time  $T$ );

$T_x$ : the time between the data received and data written to registers (max. 1 scan time);

$T_y$ : the time between program operated to output (1scan time);

$T_z$ : output port delay

The transmission time under different Baud rate:

Baud rate(bps)	$T_a$ ( ms )	$T_b$ ( ms )	$T_c$ ( ms )		
			DTLK mode 0	DTLK mode 1	DTLK mode 2
9600	21.8	12.6	31.0	40.1	67.6
19200	10.9	6.3	15.5	20.1	33.8
38400	5.5	3.2	7.8	10.0	16.9
57600	3.7	2.1	5.2	6.7	11.3
76800	2.8	1.6	3.9	5.0	8.5
128000	1.7	1.0	2.4	3.0	5.1
153600	1.4	0.8	2.0	2.5	4.3
307200	0.7	0.4	1.0	1.3	2.2

## 4.16.2 RMIO (FNC 191)

Mnemonic	Function	Operands	Program steps
		K	
RMIO FNC 191 (Remote IO)	setup a small network which enables PLC controlling other 4 PLCS.	K, H:0,1 0: for built in RS485 port ; 1: for RS485 or RS232 expansion card	3 steps

**Operation:**

This instruction F191 RMIO used by PLC can setup a small network which enables PLC controlling other 4 PLCs.

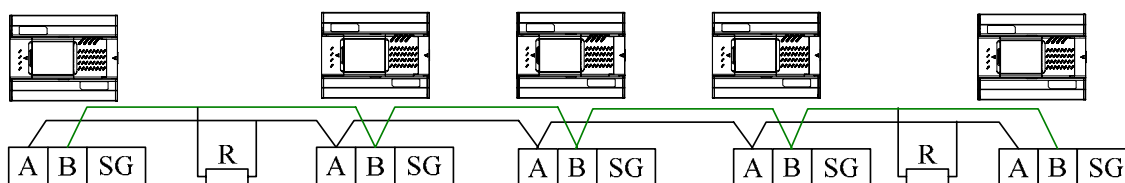
While two communication ports are ready for RMIO, only the one firstly enabled is available.

Communication frame and baud rate is set through D8120 or D8320, which is controlled by the different port.



- Note 1: When a PLC is set as a slave in RMIO mode, it is used as a expansion I/O for master and only RMIO instruction is available for operation.
- Note 2: As long as PLC as a slave in RMIO mode, only stop the operation of program can switch the RMIO to other mode.

In Remote I/O mode, the master PLC can control other 4 PLCs.



Item	Description	
standard	EIA RS485	
Baud rate	9600bps ~ 307200bps	
Number of slaves	Max 4 slave	
Related devices	Slave 1	Input: 36 points (M4200 ~ M4235) ; Output: 24point (M4600 ~ M4623)
	Slave 2	Input: 36 points (M4240 ~ M4275) ; Output: 24point (M4624 ~ M4647)
	Slave 3	Input: 36 points (M4280 ~ M4315) ; Output: 24point (M4648 ~ M4671)
	Slave 4	Input: 36 points (M4320 ~ M4355) ; Output: 24point (M4672 ~ M4695)
Cable	Insulated twisted cable, 2 lines type, Total length: 500m (76800bit/s), 1km(38400bit/s)	

Both the port RS485/ RS232 expansion card (all type is available for expansion) , RS485 port (only built-in port in H type) are available for Data Link. However, both of them can not be enabled simultaneously.

Note: Only basic unit can be set as a slave in RMIO mode.



**Related devices:**

## 1 ) Special relays

Special relays	Feature	Function	Description	Respond from
M8335	Read only	Communication state	ON as RMIO communication is enabled	M/L
M8336	Read only	Master error	ON as master error	L
M8337	Read only	Slave 1 error	On as slave 1 error	M/L
M8338	Read only	Slave 2 error	On as slave 2 error	M/L
M8339	Read only	Slave 3 error	On as slave 3 error	M/L
M8340	Read only	Slave 4 error	On as slave 4 error	M/L
M8341	Read only	RMIO mode	Expansion card is in RMIO mode	M/L
M8342	Read only	RMIO mode	RS485 port is in RMIO mode	M/L

## 2 ) Data register D

Special relays	Feature	Function	Description	Respond from
D8373	Read only	Address number	Saving its own address number	M/L
D8374	Read only	The number of slaves	Saving the number of slaves	M/L
D8376	Write	Address number setting	Setting its own address number	M/L
D8377	Write	Setting the number of slaves	setting the number of slaves	M
D8379	Read/write	Retry times	Setting retry times	M
D8380	Read/write	Time-out setting	Setting communication time-out ( Time-Out )	M/L
D8331	Read only	Current communication scan time	Saving current communication scan time	M
D8332	Read only	Max communication scan time	Saving Max communication scan time	M
D8333	Read only	Master error times	Master error times	L
D8334	Read only	Slave 1 error times	Slave 1 error times	M/L
D8335	Read only	Slave 2 error times	Slave 2 error times	M/L
D8336	Read only	Slave 3 error times	Slave 3 error times	M/L
D8337	Read only	Slave 4 error times	Slave 4 error times	M/L
D8338	Read only	Master error code	Master error code	L
D8339	Read only	Slave 1 error code	Slave 1 error code	M/L
D8340	Read only	Slave 2 error code	Slave 2 error code	M/L
D8341	Read only	Slave 3 error code	Slave 3 error code	M/L
D8342	Read only	Slave 4 error code	Slave 4 error code	M/L

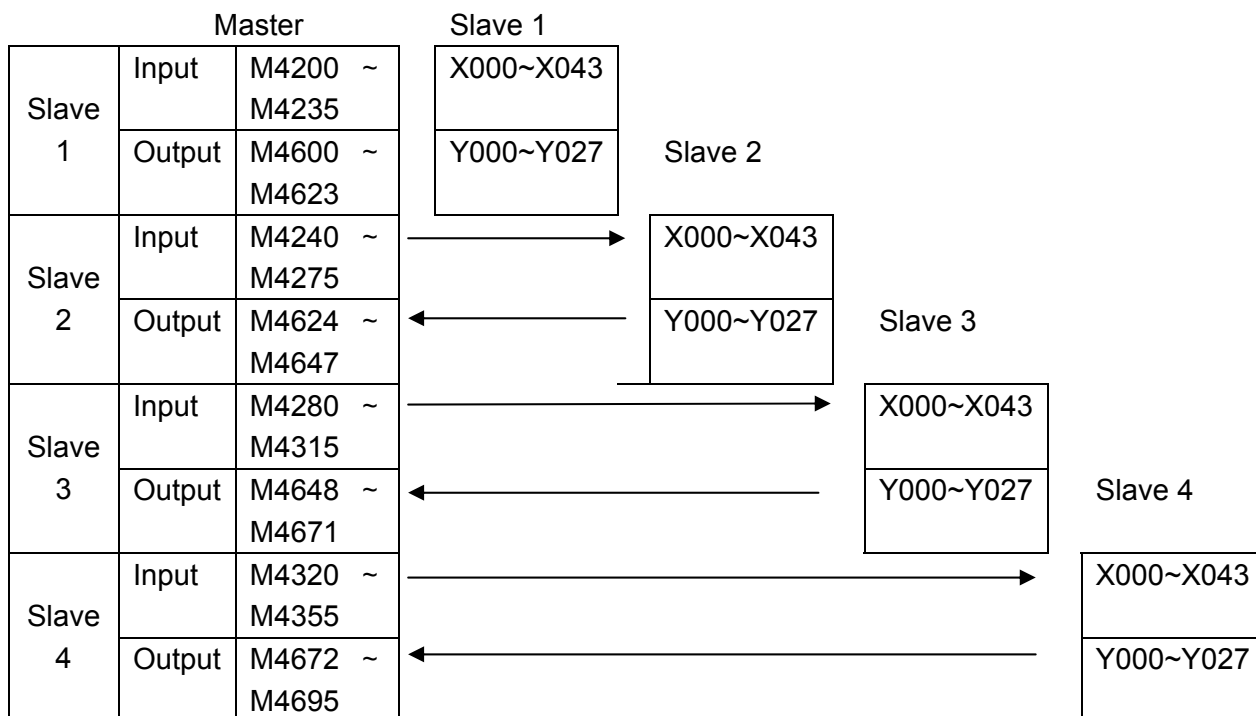
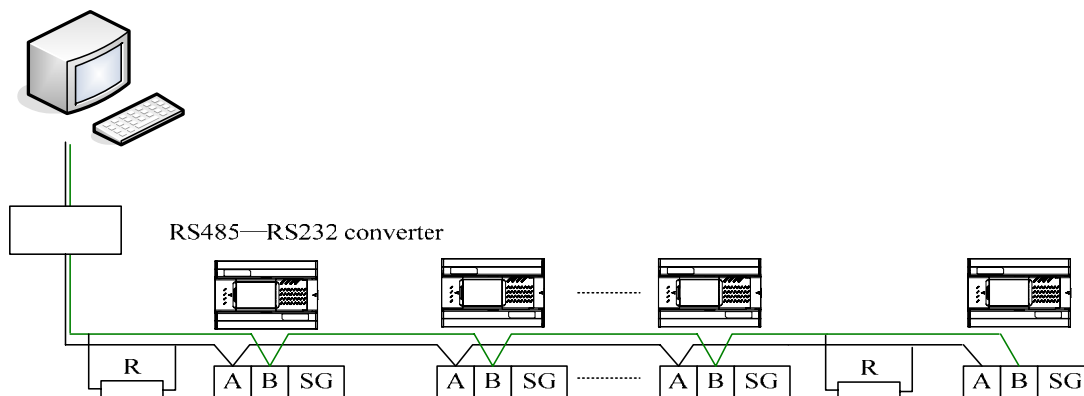
**Setting:**

When the program is in operation, or PLC is power ON, all the setting for Remote I/O will take effect.

- 1) Setting the slaver address (D8376)  
Set 0 ~ 4 to the special data register D8376, 0 is for master, and 1 ~ 4 is for slave.
- 2) Setting the slavers number (D8377)  
Set 1 ~ 4 to the special data register D8377(default: 4). It is unnecessary for slavers  
The slavers number should be set according to different condition in order to raise the refreshing speed.

**The related devices for Remote I/O:**

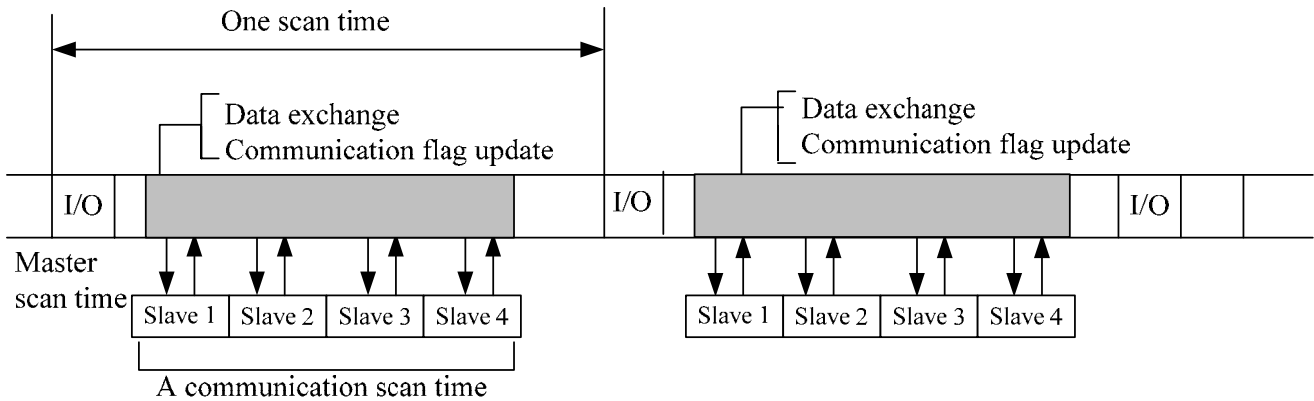
In Remote I/O mode, the related devices for master:

**Wiring:**

- Note 1: SHL terminal should be 3 class ground or the production will be interrupted to error operation because of noise.

- Note 2: Branch of communication cable should not exceed 3.
- Note 3: R represents terminal resistor (120Ω, 1/4W).

Communication sequence and the time required for transmission



### The Time Required for Transmission

The communication of master-station to slave-station, the data exchange of remote I/O and the update of communication flag are synchronous with the scan cycle of master station.

The process (1 communication period) will increase the SCAN TIME of master-station

When there is error in communication between master and slave, Remote I/O communication and PLC operation will stop and enter abnormal condition.

When an error occurs on the communication between the master station and slave-station, the remote I/O communication and PLC operation will be stopped and enter error mode.

Besides, all communication flag of master-station and slave-station are set to OFF.

Possible cause of error is as follows:

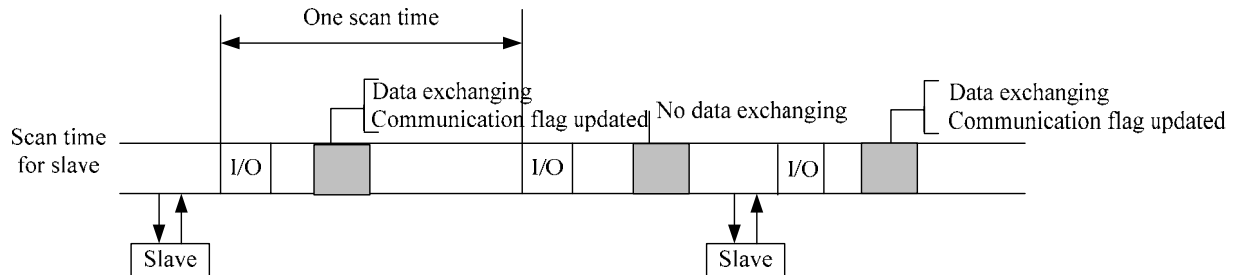
- ① CRC error
- ② Slave in STOP mode or ERROR mode
- ③ Slave not connected or connection wire broken

When the master-station is in STOP mode or ERROR mode, it will not communicate with any slave-station. The settings for communication format between master and slave are not same.

### Communication sequence for slave

The communication of slave to master is asynchronous with the scan time of slave.

After communication between master and slave is finished, the Remote I/O data and communication flag will be refreshed, which will last about 0.2ms.



### The time required for transmission

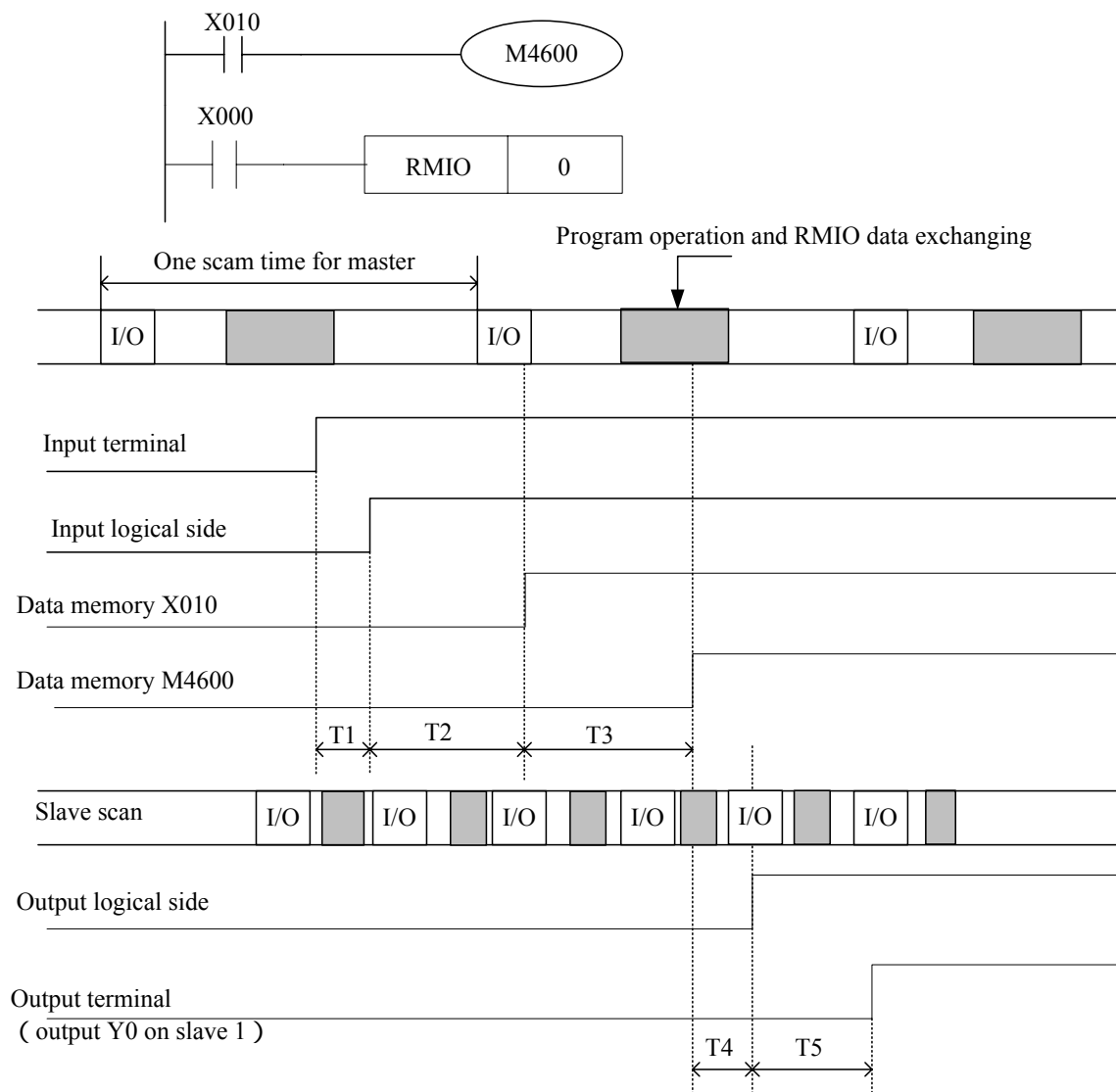
In remote I/O mode, the time T (the communication period, this period will be included in the master station SCAN TIME) required for master-station to complete the communication with all slave-stations is as follows :

Baud Rate (bps)	Communication time for each slave, T <sub>n</sub> (ms)	Time out, t (ms)	Communication time for master, T(ms)	Normal communication time for master and 4 slaves (ms)
9600	42	D8380*10	T <sub>n</sub> *n <sub>1</sub> + t*n <sub>2</sub> (n <sub>1</sub> : normal slave number; n <sub>2</sub> : slave number for time out)	168
19200	21			84
38400	11			44
57600	7			28
76800	6			24
128000	4			16
153600	3			12
307200	2			8

If there is communication error in slave, the communication time will be increased repeatedly (T<sub>n</sub> will be added to the time for each error)

Delay time:

When the remote I/O is receiving data, there will be some delay as in the following figure.



T1: delay for input (response time for OFF to ON)

T2: time for master writing data to coil register (max 1 scan time)

T3: program operation and output time

T4: time between the slave received data to output terminal

T5: delay for output (response time for ON to OFF)

**Error code:**

When there is error, the special relays M8400 ~ M8415 will indicates the error condition and the error code will be stored in special data registers (D8419 ~ D8434).

Error code	Error	Error address	Check address	Description	Check point
01H	Communication time out error	L	M	There is no responding as the master sends the request to slave and time out.	Wiring, power supply and run/ stop state
02H	Communication number error	L	M	Address is not set according to the certain relations between master and slave	Wiring
03H	Communication counting error	L	M	The data in communication counter does not conform to according to the certain relations between master and slave	Wiring
04H	Communication frame error	L	M, L	Communication frame of slave is error	Wiring and RMIO setting
11H	Communication over time error	M	L	After the slave responses to master, the master does not send another request to slavers.	Wiring, power supply and run/ stop state
14H	Communication frame error	M	L	Communication frame of master is error	Wiring and RMIO setting
21H	Without slave	L	L *1	Address in the net is wrong	Address setting
22H	Address error	L	L *1	Slave address does not comply with the certain relations between master and slave	Wiring
23H	Communication counting error	L	L *1	The data in communication counter does not conform to according to the certain relations between master and slave	Wiring
24H	Communication frame error	L	L *1	Communication frame of slave is error	Wiring and RMIO setting

M: master

L: slave

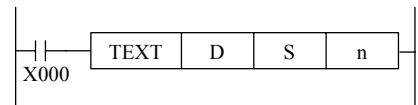
\*1: another slave

## 4.16.3 TEXT (FNC 192)

Mnemonic	Function	Operands			Program steps
		D	S	n	
TEXT FNC 192	display text (including register data) on the P07/08 LCD	D	D	K, H: 1,2	7 steps

**Operation:**

This instruction should be used with OP07/08. After F192 is enabled, the value 13 will be written to data register D8284, after OP07/08 saving the '13' in D8284, the value 13 also will be written to D8285 by OP07/08 itself.



As F192 is enabled, the certain text file will be saved to D8280 and D8281 (D8280 is for the file to be displayed in the first line of OP07/08, D8281 is for the second one) and the value to be displayed will be saved to D8295 and D8296.

The value in D8295 will be displayed in the '#' position of the first line, while the value in D8296 of the second line.

When there is '?' on LCD, you can input data, the input data for first line will be saved in the D register (Number = value in D8295 + 1). As for the second line, the input data in position '?' will be saved in D register (Number = value in D8296 + 1).

'#' and '?' can be placed anywhere in the text file. However, only the former 5 ones can be set as inputs or outputs.

**Example:**

LCD position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
--------------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----

**Text file 1 :**

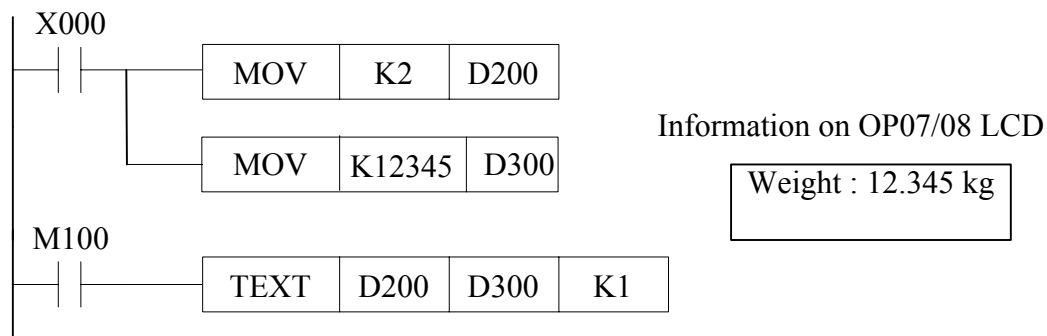
D register	2000		2001		2002		2003		2004		2005		2006		2007		2008		2009	
Content	L	e	n	g	t	h		:		#	#	.	#	#	#		c	m		

**Text file 2 :**

D register	2010		2011		2012		2013		2014		2015		2016		2017		2018		2019	
Content	W	e	i	g	t	h		:		#	#	.	#	#	#		k	g		

**Text file 3 :**

D register	2020		2021		2022		2023		2024		2025		2026		2027		2028		2029	
Content	U	n	i	t		p	r	i	c	e		:		\$	?	?	?	?	?	

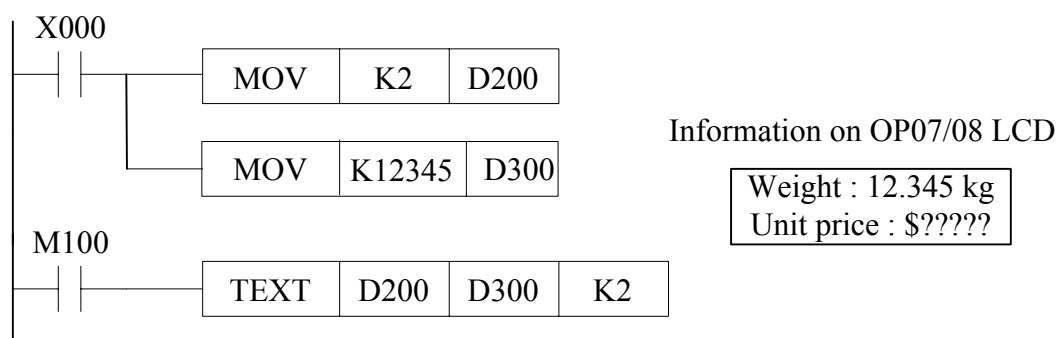


## Description:

1 , X000 ON, 2 will be moved to D200 while 12345 will be moved to D300 ;

2 , when M100 is ON, TEXT instruction is enabled. D8284 defaults 13 , D200 will be written to D8280; 300 to D8285. Then OP07/08 will enter F192 mode.

3 , F192 will operate for the first time. As D8280 = D200 = 2, OP07/08 will display the file 2 on the first line of LCD. Because there is a '#' in the file 2, 12345 in D300 will be displayed in the place of '#'. .



## Description:

1 , X000 ON, 2 will be moved to D200 while 12345 will be moved to D300 ;

2 , When M100 is ON, TEXT instruction is enabled. D8284 defaults 13 , D200 will be written to D8280; sum of data in D200 and 1 will be written to D8281, 300 to D8285, D8286. Then OP07/08 will enter F192 mode.

3 , F192 will operate for the first time. As D8280 = D200 = 2, D8281=3, OP07/08 will display the file 2 on the first line of LCD and file 3 on the second line. Moreover, 12345 in D300 will be displayed in the place of '#' and the input data by the keys will be stored in D301.



#### 4.17 Inline Comparisons - FNC 220 to FNC 249

**Contents:**

LD	-	LD compare	FNC 224 to 230
AND	-	AND compare	FNC 232 to 238
OR	-	OR compare	FNC 240 to 246

**Symbols list:**

D - Destination device.

S - Source device.

m, n- Number of active devices, bits or an operational constant.

Additional numeric suffixes will be attached if there are more than one operand with the same function e.g. D<sub>1</sub>, S<sub>3</sub> or for lists/tables devices D<sub>3+0</sub>, S<sub>+9</sub> etc.

MSB - Most Significant Bit, sometimes used to indicate the mathematical sign of a number, i.e. positive = 0, and negative = 1.

LSB - Least Significant Bit.

**Instruction modifications:**

- An instruction operating in 16 bit mode, where identifies the instruction mnemonic.
- P - A 16 bit mode instruction modified to use pulse (single) operation.
- D - An instruction modified to operate in 32 bit operation.
- D - A 32 bit mode instruction modified to use pulse (single) operation.
- A repetitive instruction which will change the destination value on every scan unless modified by the pulse function.
- An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

4.17.1 LD compare (FNC 224 to 230)

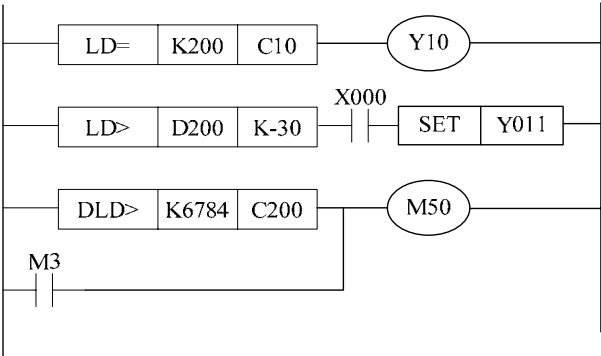
Mnemonic	Function	Operands		Program steps
		S	D	
LD (LoaD compare) where is =, >, <, <>, ,	Initial comparison contact. Active when the comparison S1 S2 is true	K,H, KnX, KnY, KnM, KnS, T, C, D, V, Z		LD : 5 steps DLD : 9 steps

**Operation:**

The value of S<sub>1</sub> and S<sub>2</sub> are tested according to the comparison of the instruction. If the comparison is true then the LD contact is active. If the comparison is false then the LD contact is not active.

**Points to note:**

The LD comparison functions can be placed anywhere in a program that a standard LD instruction can be placed. I.e., it always starts a new block.



F No	16 bit	32 bit	Active when	Inactive when
224	LD=	D LD=	S1=S2	S1 S2
225	LD>	D LD>	S1>S2	S1 S2
226	LD<	D LD<	S1<S2	S1 S2
228	LD<>	D LD<>	S1 S2	S1=S2
229	LD	D LD	S1 S2	S>1S2
230	LD	D LD	S1 S2	S1<S2

## 4.17.2 AND compare (FNC 232 to 238)

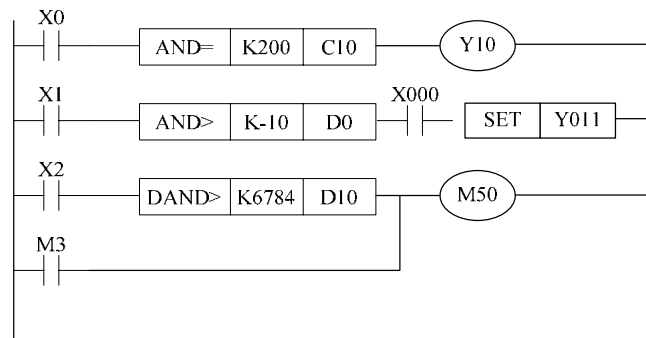
Mnemonic	Function	Operands		Program steps
		S	D	
AND (AND compare) where is =, >, <, <>, ≤, ≥	Serial comparison contact. Active when the comparison S1 S2 is true.	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z		AND : 5 steps DAND : 9 steps

**Operation:**

The value of S<sub>1</sub> and S<sub>2</sub> are tested according to the comparison of the instruction. If the comparison is true then the AND contact is active. If the comparison is false then the AND contact is not active.

**Points to note:**

The AND comparison functions can be placed anywhere in a program that a standard AND instruction can be placed. I.e., it is a serial connection contact.



F No	16 bit	32 bit	Active when	Inactive when
232	AND=	D AND =	S1=S2	S1 S2
233	AND >	D AND >	S1>S2	S1 S2
234	AND <	D AND <	S1<S2	S1 S2
236	AND<>	D AND<>	S1 S2	S1=S2
237	AND	D AND	S1 S2	S>1S2
238	AND	D AND	S1 S2	S1<S2

## 4.17.3 OR compare (FNC 240 to 246)

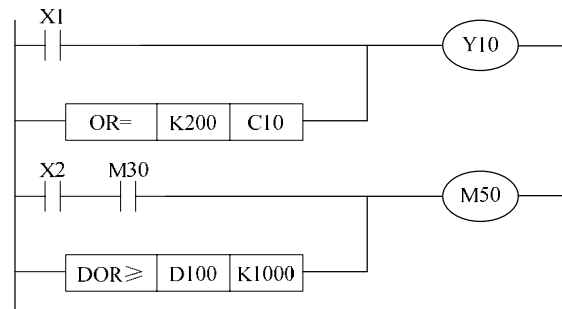
Mnemonic	Function	Operands		Program steps
		S	D	
OR (OR compare) where is =, >, <, <>, ≤, ≥	Parallel Comparison contact. Active when the comparison S1 S2 is true.	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z		OR : 5 steps DOR : 9 steps

**Operation:**

The value of S<sub>1</sub> and S<sub>2</sub> are tested according to the comparison of the instruction. If the comparison is true then the OR contact is active. If the comparison is false then the OR contact is not active.

**Points to note:**

The OR comparison functions can be placed anywhere in a program that a standard OR instruction can be placed. I.e., it is a parallel connection contact.



F No	16 bit	32 bit	Active when	Inactive when
240	OR=	D OR=	S1=S2	S1 S2
241	OR>	D OR>	S1>S2	S1 S2
242	OR<	D OR<	S1<S2	S1 S2
244	OR<>	D OR<>	S1 S2	S1=S2
245	OR	D OR	S1 S2	S>1S2
246	OR	D OR	S1 S2	S1<S2

5	Special relay .....	2
5.1	PC status (M) .....	2
5.2	Clock device ( M ) .....	2
5.3	Operation flags ( M ) .....	3
5.4	PC status ( D ) .....	3
5.5	RTC ( D ) .....	3
5.6	PC operation mode ( M ) .....	4
5.7	PC mode ( D ) .....	5
5.8	Step ladder flagss ( M ) .....	6
5.9	Step ladder flags ( D ) .....	6
5.10	Interruption disable ( M ) .....	7
5.11	UP/DOWN counting set device ( M ) .....	7
5.12	Register capacity ( D ) .....	8
5.13	Devices ( M ) .....	9
5.14	Error detection ( D ) .....	9
5.15	Communication and link ( M ) I .....	9
5.16	Communication and link ( D ) I .....	10

## 5 Special relay

### 5.1 PC status (M)

Device No.	Function	Operation
8000	RUN monitor (NO contact)	
8001	RUN monitor (NC contact)	
8002	Initial pulse (NO contact)	
8003	Initial pulse (NC contact)	
8004	Error occurrence	
8005	Battery voltage too low	ON when battery voltage is too low.
8006	Latch for batter low voltage	ON when battery voltage is too low. OFF when a new battery is installed.

### 5.2 Clock device ( M )

Device No.	Function	Operation
8010		
8011	10ms period oscillator	5ms ON/5ms OFF
8012	100ms period oscillator	50ms ON/50ms OFF
8013	1s period oscillator	0.5s ON/0.5s OFF
8014	1min period oscillator	30s ON/30s OFF
8015	Clock stop and set	Stop timing and reset the clock
8016	Stop displaying clock time	Stop displaying clock time
8017	+/-30s offset	+/-30 s offset for internal time
8018	RTC detection	Check whether RTC is enabled.
8019	RTC error	Clock is set out of the range.

### 5.3 Operation flags ( M )

Device No.	Function	Operation
8020	Zero	On when the result of add or subtract is 0
8021	Borrow	On when the result of subtract is smaller than the minimum negative number of the system
8022	Carry	ON when the result of add should be carry
8023		
8024	BMOV direction	(F15) 0: forward, 1: reverse
8026	RAMP mode	(F67) 0: reset, 1: keep
8027	PR mode	( F77 ) 0: 8bytes; 1: 16bytes
8029	Instruction execution ends	ON when the instruction as DSW(F72) is finished

### 5.4 PC status ( D )

Device No.	Function	Operation
8001	TP03 type	0x
8002	Version	0x100 represents 1.00 version
8003	ID number	Read only
8004	Error code	
8005	Warn code	
8006	Program capacity	

### 5.5 RTC ( D )

Device No.	Function	Operation
8010	Present scan time(0.1ms unit)	
8011	Min scan time	
8012	Max scan time	
8013	Second (0~59)	
8014	Minute (0~59)	
8015	Hour	
8016	Day	
8017	Month	
8018	Year (00~99)	
8019	Week	

## 5.6 PC operation mode ( M )

Device No.	Function	Operation
8031	Non-retentive register all clear(when executing END instruction)	When M8031 ON, the ON/ OFF status Y/M/S/T/C and present value of T/C/D are reset. However, special Data register will not be cleared.
8032	Retentive register all clear(when executing END instruction)	When M8032 ON, the retentive registers are reset.
8033	Register hold in stop mode <i>note:1</i>	Saving mode for Register 0: STOP→RUN, TP03 saves according to requirement 1: STOP→RUN,TP03 saves all data
8034	Output prohibit	1: output 0; 0: output Y
8035	Enforced operation mode	
8036	Enforced Run instruction	
8037	Enforced Stop instruction	
8039	Constant scan mode	1: ENABLE ; 0: DISABLE This register will not be initiated in Power ON.

Note 1:

In the following table, the column 'Latch from' and 'Latch end' can be modified inside the set range. 0: saving according to the requirement you set. 1: saving all the data regardless of the set requirements or set range.

Device	Mnemonic	Point	Start	End	Latch from	Latch end	Latch set range
Supplementary relay	M	3072	0	3071	500	1023	0-1023
State	S	1000	0	999	500	999	0-999
Timer	T	256	0	255			
Counter 16bit	C	500	0	199	100	199	0-199
Counter 32bit	C	56	200	255	200	255	200-255
Data register	D	8000	0	7999	200	511	0-511



**5.7 PC mode ( D )**

Device No.	Function	Operation
8039	Constant scan time	Default: 0, unit: ms

**5.8 Step ladder flagss ( M )**

Device No.	Function	Operation
8040	STL transfer disable	M8040 ON, STL transfer is disabled.
8041	STL transfer start	When M8041 ON, STL state transfer is enabled in automatic operation
8046	STL state ON	When M8047 is ON and any one of S0~S899 is on, M8064 will be ON.
8047	Enable STL monitor	As long as M8047 is ON, D8040~D8047 are enabled.
8048	Annunciator ON	IVM8049 ON, and any one of S900~S999 is on, M8048 will be ON
8049	Enable Annunciator	M8049 ON, D8049 is enabled.

**5.9 Step ladder flags ( D )**

Device No.	Function	Operation
8040	Address for ON State	
8041		
8042		
8043		
8044		
8045		
8046		
8047		
8048		
8049	The minimum address for ON State among (S900 ~ S999)	

### 5.10 Interruption disable ( M )

Device No.	Function	Operation
8050	Input interruption disable(I00x)	
8051	Input interruption disable(I10x)	
8052	Input interruption disable(I20x)	
8053	Input interruption disable(I30x)	
8054	Input interruption disable(I40x)	
8055	Input interruption disable(I50x)	
8056	Timing interruption disable(I6xx)	
8057	Timing interruption disable(I7xx)	
8058	Timing interruption disable(I8xx)	
8059	Counting interruption disable	I010~I060 interruption disable

### 5.11 UP/DOWN counting set device ( M )

Device No.	Function	Operation
8200	UP/DOWN counting set for C200	
8201	UP/DOWN counting set for C201	
8202	UP/DOWN counting set for C202	
8203	UP/DOWN counting set for C203	
8204	UP/DOWN counting set for C204	
8205	UP/DOWN counting set for C205	
8206	UP/DOWN counting set for C206	
8207	UP/DOWN counting set for C207	
8208	UP/DOWN counting set for C208	
8209	UP/DOWN counting set for C209	
8210	UP/DOWN counting set for C210	
8211	UP/DOWN counting set for C211	
8212	UP/DOWN counting set for C212	
8213	UP/DOWN counting set for C213	
8214	UP/DOWN counting set for C214	
8215	UP/DOWN counting set for C215	
8216	UP/DOWN counting set for C216	
8217	UP/DOWN counting set for C217	
8218	UP/DOWN counting set for C218	
8219	UP/DOWN counting set for C219	
8220	UP/DOWN counting set for C220	
8221	UP/DOWN counting set for C221	
8222	UP/DOWN counting set for C222	

8223	UP/DOWN counting set for C223	
8224	UP/DOWN counting set for C224	
8225	UP/DOWN counting set for C225	
8226	UP/DOWN counting set for C226	
8227	UP/DOWN counting set for C227	
8228	UP/DOWN counting set for C228	
8229	UP/DOWN counting set for C229	
8230	UP/DOWN counting set for C230	
8231	UP/DOWN counting set for C231	
8232	UP/DOWN counting set for C232	
8233	UP/DOWN counting set for C233	
8234	UP/DOWN counting set for C234	
8241	UP/DOWN counting set for C241	
8242	UP/DOWN counting set for C242	
8243	UP/DOWN counting set for C243	
8244	UP/DOWN counting set for C244	
8245	UP/DOWN counting set for C245	
8246	UP/DOWN counting set for C246	
8247	UP/DOWN counting set for C247	
8248	UP/DOWN counting set for C248	
8249	UP/DOWN counting set for C249	
8250	UP/DOWN counting set for C250	
8251	UP/DOWN counting monitor for C251	
8252	UP/DOWN counting set for C252	
8253	UP/DOWN counting monitor for C253	
8254	UP/DOWN counting set for C254	
8255	UP/DOWN counting set for C255	

## 5.12 Register capacity ( D )

Device No.	Function	Operation
8102	Data register content	

**5.13 Devices ( M )**

Device No.	Function	Operation
8061	PLC hardware check	PLC hardware error
8064	Parameter check	
8065	Syntax check	
8066	Program check	
8067	Operation check	
8068	Operation error latch	
8109	Output update check	
M8069	I/O bus check	

**5.14 Error detection ( D )**

Device No.	Function	Operation
8061	Error code	
8063	Error code	
8064	Error code	
8065	Error code	
8066	Error code	
8067	Error code	
8068	Error code	
8109	Address of Y in output update error	

**5.15 Communication and link ( M ) I**

For RS485 port

Device No.	Function	Operation
8121	RS485 communication port send data is ready	RS, MBUS
8122	RS485 communication port sending flag	RS, MBUS
8123	RS485 communication port receiving data end flag	RS, MBUS
8124	RS485 communication port MBUS	MBUS

	instruction error	
8129	RS485 communication port communication over time.	RS, MBUS

For expansion communication port

Device No.	Function	Operation
8321	Expansion communication port send data is ready	RS, MBUS
8322	Expansion communication port sending flag	RS, MBUS
8323	Expansion communication port receiving data end flag	RS, MBUS
8324	Expansion communication port MBUS instruction error	MBUS
8329	Expansion communication port communication over time.	RS, MBUS

For RMIO

Device No.	Function	Operation
8335	RMIO data in transmission	
8336	RMIO data transmission error ( master)	
8337	RMIO data transmission error (slave 1)	
8338	RMIO data transmission error (slave 2)	
8339	RMIO data transmission error (slave 3)	
8340	RMIO data transmission error (slave 4)	
8341	Expansion communication port is under RMIO	
8342	RS 485 communication port is under RMIO	

## 5.16 Communication and link ( D ) I

For RS485 port

Device No.	Function	Operation
8120	Communication format	RS485 communication port 89Hex
8121	Address	Read-only default: 01
8122	Remaining data number of RS485 sending data	
8123	Number of RS485 Data received	
8124	Start character	RS485 communication port, RS instruction 02Hex
8125	End character	RS485 communication port, RS instruction 03Hex
8129	Communication watchdog time	RS485 communication port, RS and MBUS instruction

For expansion communication port

Device No.	Function	Operation
8320	Communication format	Expansion communication port ( RS485/RS232 ) 89Hex
8321	Address	PC/PDA communication port 89Hex
8322	Remaining data number of sending data	Expansion communication port
8323	Number of RS485 Data received	Expansion communication port
8324	Start character	Expansion communication port, RS instruction 02Hex
8325	End character	Expansion communication port RS instruction 03Hex
8329	Communication watchdog time	Expansion communication port ( RS and MBUS )

For RMIO

Device No.	Function	Operation
8373	RMIO slave setting state	
8374	RMIO slave setting	
8376	RMIO slave	
8377	RMIO slave number setting	
8379	RMIO retry times	
8380	RMIO monitor time	
8331	Current scan time	
8332	Max scan time	
8333	Error counting number (master)	
8334	Error counting number (slave 1)	
8335	Error counting number (slave 2)	
8336	Error counting number (slave 3)	
8337	Error counting number (slave 4)	
8338	Error code (master )	
8339	Error code (slave 1)	
8340	Error code (slave 2)	
8341	Error code (slave 3)	
8342	Error code (slave 4)	

## 5.1 Communication and link ( M ) II

DTLK

Device No.	Function	Operation
8400	Data sending error (master)	
8401	Data sending error(slave 1)	
8402	Data sending error(slave 2)	
8403	Data sending error(slave 3)	
8404	Data sending error(slave 4)	
8405	Data sending error(slave 5)	
8406	Data sending error(slave 6)	
8407	Data sending error(slave 7)	
8408	Data sending error(slave 8)	
8409	Data sending error(slave 9)	
8410	Data sending error(slave 10)	
8411	Data sending error(slave 11)	
8412	Data sending error(slave 1 2)	
8413	Data sending error (slave 13)	
8414	Data sending error(slave 14)	
8415	Data sending error(slave 15)	
8416	Data sending	
8417	Expansion communication port is set as DTLK	
8418	RS485 port is set as DTLK	

## 5.2 Communication and link ( D ) II

DTLK

8173	Set state of master	Data Link
8174	Set state of slave	Data Link
8175	Set state of refresh range	Data Link
8176	set Master address	Data Link
8177	set Slaver address	Data Link
8178	set Refresh range	Data Link
8179	Retry times	Data Link
8180	Monitor time	Data Link
8401	Current scan time	
8402	Max scan time	
8403	Error counting number (master)	
8404	Error counting number (slave1)	
8405	Error counting number (slave2)	
8406	Error counting number (slave3)	
8407	Error counting number (slave4)	
8408	Error counting number (slave5)	
8409	Error counting number (slave6)	
8410	Error counting number (slave7)	



8411	Error counting number (slave8)	
8412	Error counting number (slave9)	
8413	Error counting number (slave10)	
8414	Error counting number (slave11)	
8415	Error counting number (slave12)	
8416	Error counting number (slave13)	
8417	Error counting number (slave14)	
8418	Error counting number (slave15)	
8419	Error code (master)	
8420	Error code (slave 1)	
8421	Error code (slave2)	
8422	Error code (slave3)	
8423	Error code (slave4)	
8424	Error code (slave5)	
8425	Error code (slave6)	
8426	Error code (slave7)	
8427	Error code (slave8)	
8428	Error code (slave9)	
8429	Error code (slave10)	
8430	Error code (slave11)	
8431	Error code (slave12)	
8432	Error code (slave13)	
8433	Error code (slave14)	
8434	Error code (slave15)	

### 5.3 High speed and position ( M )

8130	F55(HSZ) High speed counter zone compare mode	
8131	Finish flag for F55	
8132	F55(HSZ),F57(PLSY) speed mode	
8133	F55,F57 performing end flag	
8134	Reserved	
8135	Reserved	
8136	Reserved	
8137	Reserved	
8138	Reserved	
8139	Reserved	
8140	FNC156(ZRN)CLR signal output enable	
8141	Reserved	
8142	Reserved	
8143	Reserved	
8144	Reserved	
8145	Y000 pulse output stops	
8146	Y001 pulse output stops	
8147	Y000 pulse output monitoring (busy/read)	
8148	Y001 pulse output monitoring (busy/read)	
8149	Reserved	

### 5.4 Expansion ( M )

8158	Reserved	
8159	Reserved	
8160	F17(XCH) SWAP	
8161	8 octal processing mode (76,80,83,87,84)	
8162	High speed parallel link mode	
8163		
8164		
8165	Reserved	
8166	Reserved	
8167	F71(HKY)HEX data processing	
8168	F13(SMOV)DE HEX processing	
8169		

### 5.5 High speed and position ( D )

8130	High speed counter zone compare	
8131	Contains the number of the current record being processed in the HSZ comparison table when the PLSY operation has been enabled	
8132	Frequency (HSZ, PLSY)	
8133		
8134	Target pulse	
8135		
8136	Accumulated value for output pulse of Y000 and Y001	
8137		
8138		
8139		
8140	F57, 59(PLSR), Accumulated value for output pulse of Y000 or present value of position instruction.	
8141		
8142	F57, 59(PLSR), Accumulated value for output pulse of Y001 or present value of position instruction.	
8143		
8144		
8145	Offset speed for F156,F158,F159	
8146	Fastest speed	
8147		
8148	Initial value	
8149		

**5.6 OP07/08 ( M )**

8280	Key F1	
8281	Key F2	
8282	Key F3	
8283	Key F4	
8284	Key F5	
8285	Key F6	
8286	Key F7	
8287	Key F8	
8288	Key F9	
8289	Key F10	
8290	Key F11	
8291	Key F12	
8292	Up	
8293	Down	
8294	Left	
8295	Right	
8296	Key TMR	
8297	Key CNT	
8298	Key ENT	
8299	Key MOD1	
8300	Key MOD2	
8301	Key ESC	
8302	Reserved	
8303	Reserved	

**5.7 OP07/08 ( D )**

8280	First line content defaulted	
8281	Second line content defaulted	
8282	First line content user defined	
8283	First line content user defined	
8284	OP07/08 display mode setting	
8285	OP07/08 present display mode	
8286	OP07/08 display number range	
8287	Error code	
8288		
8289	Present number for timer mode	
8290	Present number for Counter mode	
8291	Present number for user mode1	
8292	Present number for user mode2	
8293	Present number for user mode3	
8294	Present number for user mode4	
8295	First line content for F192 mode	
8296	Second line content for F192 mode	
8297	Data format set 1	
8298	Data format set 2	
8299	Data format set 3	
8300	Data format set 4	

**5.8 AD/DA ( M )**

8257	Total quantity of AD modules is wrong	
8258	Total quantity of DA module channel is wrong	

**5.9 AD/DA ( D )**

8256	TP02-4AD number	
8257	TP03-AD number ( 0~7 )	
8258	TP02-2DA channels ( 0 , 2 )	
8259	TP03-DA channel ( 0~8 )	
8260	AD filter mode	
8261	AD1 ~ 4 channel mode set	
8262	AD5 ~ 8 channel mode set	
8263	AD9 ~ 12 channel mode set	
8264	AD13 ~ 16 channel mode set	
8265	AD17 ~ 20 channel mode set	
8266	AD21 ~ 24 channel mode set	
8267	AD25 ~ 28 channel mode set	
8268	AD29 ~ 32 channel mode set	
8269	AD33 ~ 36 channel mode set	
8270	AD37 ~ 40 channel mode set	
8271	AD41 ~ 44 channel mode set	
8272	AD45 ~ 48 channel mode set	
8273	AD49 ~ 52 channel mode set	
8274	AD53 ~ 56 channel mode set	
8275	AD57 ~ 60 channel mode set	
8276	Reserved	
8277	DA1 ~ 4 channel mode set	
8278	DA5 ~ 8 channel mode set	
8279	DA9 ~ 10 channel mode set	
8381	DA channel 1 data	
8382	DA channel 2 data	
8383	DA channel 3 data	
8384	DA channel 4 data	
8385	DA channel 5 data	
8386	DA channel 6 data	
8387	DA channel 7 data	
8388	DA channel 8 data	

8389	DA channel 9 data	
8390	DA channel 10 data	
8436	AD channel 1 data	
8437	AD channel 2 data	
8438	AD channel 3 data	
8439	AD channel 4 data	
8440	AD channel 5 data	
8441	AD channel 6 data	
8442	AD channel 7 data	
8443	AD channel 8 data	
8444	AD channel 9 data	
8445	AD channel 10 data	
8446	AD channel 11 data	
8447	AD channel 12 data	
8448	AD channel 13 data	
8449	AD channel 14 data	
8450	AD channel 15 data	
8451	AD channel 16 data	
8452	AD channel 17 data	
8453	AD channel 18 data	
8454	AD channel 19 data	
8455	AD channel 20 data	
8456	AD channel 21 data	
8457	AD channel 22 data	
8458	AD channel 23 data	
8459	AD channel 24 data	
8460	AD channel 25 data	
8461	AD channel 26 data	
8462	AD channel 27 data	
8463	AD channel 28 data	
8464	AD channel 29 data	
8465	AD channel 30 data	
8466	AD channel 31 data	
8467	AD channel 32 data	
8468	AD channel 33 data	
8469	AD channel 34 data	
8470	AD channel 35 data	
8471	AD channel 36 data	
8472	AD channel 37 data	
8473	AD channel 38 data	
8474	AD channel 39 data	
8475	AD channel 40 data	

8476	AD channel 41 data	
8477	AD channel 42 data	
8478	AD channel 43 data	
8479	AD channel 44 data	
8480	AD channel 45 data	
8481	AD channel 46 data	
8482	AD channel 47 data	
8483	AD channel 48 data	
8484	AD channel 49 data	
8485	AD channel 50 data	
8486	AD channel 51 data	
8487	AD channel 52 data	
8488	AD channel 53 data	
8489	AD channel 54 data	
8490	AD channel 55 data	
8491	AD channel 56 data	
8492	AD channel 57 data	
8493	AD channel 58 data	
8494	AD channel 59 data	
8495	AD channel 60 data	



## 6 Execution Times

### 6.1 Basic Instructions

Mnemonic	Object Devices	Steps	Execution Time in $\mu$ sec	
			ON	OFF
LD	X,Y,M,S,T,C Special M	1		
LDI				
AND				
ANI				
OR				
ORI				
LDP	X,Y,M,S,T,C	1		
LDF				
ANP				
ANF				
ORP				
ORF				
ANB	Not applicable	1		
ORB				
MPS				
MRD				
MPP				
INV				
MC	Nest level M,Y	3		
MCR	Nest level	2		
NOP	Not applicable	1		
END				
STL	S	1		
RET	Not applicable			
OUT	Y,M	1		
	S	2		
	Special M	2		
	T-K	3		
	T-D	3		
	C-K(16 bit)	3		
	C-D(16 bit)	3		
	C-K(32 bit)	5		
	C-D(32 bit)	5		
SET	Y,M	2		
	S			
	S when used in an STL step			
	Special M			

Mnemonic	Object Devices	Steps	Execution Time in $\mu$ sec	
			ON	OFF
RST	Y,M	1		
	S	2		
	Special M	2		
	T,C	2		
	D,V,Z, special D	3		
PLS		2		
PLF		2		
P	0~63	1		
I		1		

**Note 1:**

- “n” in the formulae to calculate the ON/OFF execution time, refers to the number of STL instructions at the current parallel/merge branch. Thus the value of “n” will fall in the range 1 to 8.

**6.2 Applied Instructions**

Instruction type	Application instruction			16/32 Bit	P	Processing time( $\mu$ s)	
	No.	Mnemonic	function			ON	OFF
Program flow	00	CJ	Conditional jump	16	√		
	01	CALL	Call subroutine	16	√		
	02	SRET	Subroutine return	16			
	03	IRET	Interrupt return	*1			
	04	EI	Enable interrupt	*1			
	05	DI	Disable interrupt	*1			
	06	FEND	First end	*1			
	07	WDT	Waterdog timer	16	√		
	08	FOR	Start of a for/next loop	*1			
	09	NEXT	End a for/next loop	*1			
Move and compare	10	CMP	Compare	16/ 32	√		
	11	ZCP	Zone compare	16/ 32	√		
	12	MOV	Move	16/ 32	√		
	13	SMOV	Shift move	16	√		
	14	CML	Compliment	16/ 32	√		
	15	BMOV	Block move	16	√		
	16	FMOV	Fill move	16/ 32	√		
	17	XCH	Exchange	16/ 32	√		
	18	BCD	BCD binary coded decimal	16/ 32	√		
	19	BIN	BIN binary	16/ 32	√		
Arithmetic and logic operations	20	ADD	Addition	16/ 32	√		
	21	SUB	Subtraction	16/ 32	√		
	22	MUL	Multiplication	16/ 32	√		

Instruction type	Application instruction			16/32 Bit	P	Processing time(μs)	
	No.	Mnemonic	function			ON	OFF
	23	DIV	Division	16/ 32	√		
	24	INC	Increment	16/ 32	√		
	25	DEC	Decrement	16/ 32	√		
	26	WAND	Word and	16/ 32	√		
	27	WOR	Word or	16/ 32	√		
	28	WXOR	Word exclusive or	16/ 32	√		
	29	NEG	Negation	16/ 32	√		

1. These instructions require NO preliminary contact devices such as LD, AND, OR etc.
2. Where "n" is referred to this identifies the quantity of registers to be manipulated. "n" can be equal or less than 512.
3. Where "n" is referred to this identifies the quantity of bit devices to be manipulated. "n" can be equal or less than selected operating mode, i.e. if 32 bit mode is selected then "n" can have a value equal or less than 32.
4. Where "n" is referred to this identifies the quantity of bit devices to be manipulated.
5. Where "n" is referred to this identifies the quantity devices to be manipulated. "n" can have any value taken from the range 2 through 512.
6. Where "n" is referred to this identifies the range of devices to be reset. The device type being reset is identified by the device letter in brackets in the '16/32 bit' column.
7. Where "n" is referred to this identifies the number of devices the mean is to be calculated from. The value of "n" can be taken from the range 1 through 64.
8. Where "n" is referred to this identifies the range of devices to be refreshed. The value of "n" is always specified in units of 8, i.e 8, 16, 24.....128. The maximum allowable range is dependent on the number of available inputs/outputs, i.e. FX0 is limited to 16 as a maximum batch that can be refreshed, where as FX can use 128.
9. Where "n" is referred to this identifies the time setting for the input filters operation. "n" can be selected from the range 0 through to 60 msec.
10. There are limits to the total combined use of these instructions.
11. Where "n" is referred to this identifies the number of output points. "n" may have a value equal or less than 64.
12. Where "n" is referred to this identifies the number of words read or written FROM/TO the special function blocks.
13. Where "n" is referred to this identifies the number of octal (8 bit) words read or written when two PLCs are involved in a parallel running function.
14. Where "n" is referred to this identifies the number of elements in a stack, for 16 bit operation n has a maximum of 256. However, for 32 bit operation n has a maximum of 128.

15. Where "m1" is referred to this identifies the number of elements in the data table. Values of m1 are taken from the range 1 to 32. For a the SORT instruction to completely process the data table the SORT instruction will be processed m1 times.

## 7. PLC Device Tables

Item		Specification	Remarks
Operation control method		Cyclic operation by stored program	
I/O control method		Batch processing method (when END instruction is executed)	I/O refresh instruction is available
Operation processing time		Basic instructions: 0.31 to 0.9 $\mu$ s Applied instructions: several $\mu$ s	
Programming language		Ladder/Boolean/SFC	Step ladder can be used to produce an SFC style program
Program capacity		8000 /16000 steps Provided by built in	
Number of instructions		Basic sequence instructions: 36 Step ladder instructions: 2 Applied instructions: 139	A Maximum 139 applied instructions are available
I/O configuration		Max hardware I/O configuration points 256, dependent on user selection (Max. software addressable Inputs 256, Outputs 256)	
Auxiliary relay (M coils)	General	7680 points	M0 to M7679
	Special	512 points	M8000 to M8511
State relays (S coils)	General	4096 points	S0 to S4095
	Latched	500 points (subset)	S500 to S999
	Initial	10 points(subset)	S0 to S9
	Annunciator	100 points	S900 to S999
Timers (T)	100 msec	Range: 0 to 3,276.7 sec ; 200 points	T0 to T199
	10 msec	Range: 0 to 327.67 sec ; 46 points	T200 to T245
	1 msec retentive	Range: 0 to 32.767 sec ; 4 points	T246 to T249
	100 msec retentive	Range: 0 to 3,276.7 sec ; 6 points	T250 to T255
Counters (C)	General 16 bit	Range: 1 to 32,767 counts 200 points	C0 to C199 Type: 16 bit up counter
	Latched 16 bit	100 points (subset)	C100 to C199 Type: 16 bit up counter
	General 32 bit	Range: -2,147,483,648 to 2,147,483,647 35 points	C200 to C234 Type: 32 bit up/down counter
	Latched 32 bit	15 points (subset)	C220 to C234 Type: 15 bit up/down counter
High speed counters (C)	1 phase	Range: -2,147,483,648 to +2,147,483,647 counts General rule: Select counter combinations with a combined counting frequency of 20kHz or less. Note all counters are latched	C235 to C240 6 points
	1 phase c/w start stop input		C241 to C245 5 points
	2 phase		C246 to C250 5 points
	A/B phase		C251 to C255 5 points

Item		Specification	Remarks
Data registers (D)	General	8000 points	D0 to D7999 Type: 16 bit data storage register pair for 32 bit device
	Special	512 points	From the range D8000 to D8511 Type: 16 bit data storage register
	Index	32 points	V0 to V15 and Z0 to Z15 Type: 16 bit data storage register
Pointers (P)	For use with CALL	256 points	P0 to P255
	For use With interrupts	6 input points, 3 timers, 6 counters	I00 to I50 I6 to I8 I010 to I060
Nest levels		8 points for use with MC and MCR	N0 to N7
Numbers	Decimal K	16 bit: -32,768 to +32,767 32 bit: -2,147,483,648 to +2,147,483,647	
	Hexadecimal H	16 bit: 0000 to FFFF 32 bit: 00000000 to FFFFFFFF	