ADVANCED AND EVER ADVANCING MITSUBISHI ELECTRIC

MITSUBISHI

PROGRAMMABLE CONTROLLERS

MELSEC-F

PROGRAMMING MANUAL

THE FX SERIES OF PROGRAMMABLE CONTROLLER
(FX_0, FX_{0S}, FX_{0N}, FX, FX_{2C}, FX_{2N}, FX_{2NC})

FX

FX Series Programmable Controllers

Programming Manual

Manual number : JY992D48301
Manual revision : J
Date : November 1999

Foreword

- This manual contains text, diagrams and explanations which will guide the reader in the correct programming and operation of the PLC.
- Before attempting to install or use the PLC this manual should be read and understood.
- If in doubt at any stage of the installation of the PLC always consult a professional electrical engineer who is qualified and trained to the local and national standards which apply to the installation site.
- If in doubt about the operation or use of the PLC please consult the nearest Mitsubishi Electric distributor.
- This manual is subject to change without notice.

FAX BACK - Combined Programming Manual (J)

Mitsubishi has a world wide reputation for its efforts in continually developing and pushing back the frontiers of industrial automation. What is sometimes overlooked by the user is the care and attention to detail that is taken with the documentation. However,to continue this process of improvement, the comments of the Mitsubishi users are always welcomed. This page has been designed for you,the reader,to fill in your comments and fax them back to us. We look forward to hearing from you.

Please tick the box of your choice;

Fax numbers:		Your name.....
Mitsubishi Electric....		.....
America	(01) 847-478-2253	Your company .....
Australia	(02) 638-7072	.....
Germany	(0 21 02) 4 86-1 12	Your location: .....
South Africa	(0111) 444-8304	.....
United Kingdom	(01707) 278-695	

What condition did the manual arrive in?	☐Good	☐Minor damage	☐Unusable
Will you be using a folder to store the manual?		☐Yes	☐No
What do you think to the manual presentation?		☐Tidy	☐Un-friendly
Are the explanations understandable?	☐Yes	☐Not too bad	☐Unusable

Which explanation was most difficult to understand: .....
.....

Are there any diagrams which are not clear? ☐Yes ☐No

If so,which:.....

What do you think to the manual layout? ☐Good ☐Not too bad ☐Un-helpful

If there one thing you would like to see improved,what is it?.....
.....
.....

Could you find the information you required easily using the index and/or the contents,if possible please identify your experience: .....
.....
.....
.....
.....

Do you have any comments in general about the Mitsubishi manuals?.....
.....
.....
.....
.....

Thank you for taking the time to fill out this questionnaire. We hope you found both the product and this manual easy to use.

Guidelines for the Safety of the User and Protection of the Programmable Controller (PLC)

This manual provides information for the use of the FX family of PLC's. The manual has been written to be used by trained and competent personnel. The definition of such a person or persons is as follows;

a) Any engineer who is responsible for the planning, design and construction of automatic equipment using the product associated with this manual should be of a competent nature, trained and qualified to the local and national standards required to fulfill that role. These engineers should be fully aware of all aspects of safety with regards to automated equipment.

b) Any commissioning or service engineer must be of a competent nature, trained and qualified to the local and national standards required to fulfill that job. These engineers should also be trained in the use and maintenance of the completed product. This includes being completely familiar with all associated documentation for the said product. All maintenance should be carried out in accordance with established safety practices.

c) All operators of the completed equipment should be trained to use that product in a safe and coordinated manner in compliance to established safety practices. The operators should also be familiar with documentation which is connected with the actual operation of the completed equipment.

Note : the term 'completed equipment' refers to a third party constructed device which contains or uses the product associated with this manual.

Note's on the Symbols used in this Manual

At various times through out this manual certain symbols will be used to highlight points of information which are intended to ensure the users personal safety and protect the integrity of equipment. Whenever any of the following symbols are encountered its associated note must be read and understood. Each of the symbols used will now be listed with a brief description of its meaning.

Hardware Warnings

1) Indicates that the identified danger **WILL** cause physical and property damage.

2) Indicates that the identified danger could **POSSIBLY** cause physical and property damage.

3) Indicates a point of further interest or further explanation.

Software Warnings

4) Indicates special care must be taken when using this element of software.

5) Indicates a special point which the user of the associate software element should be aware of.

6) Indicates a point of interest or further explanation.

Contents

Chapter Contents

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

1. Introduction

1.1 Overview

1) Scope of this manual
This manual gives details on all aspects of operation and programming for FX, FX2C, FX0N, FX0S, FX0, FX2N and FX2NC programmable controllers (PLCs). For all information relating to the PLC hardware and installation, refer to the appropriate manual supplied with the unit.

2) How to use this manual
This manual covers all the functions of the highest specification Programmable (Logic) Controller (PLC). For this reason, the following indicator is included in relevant section titles to show which PLCs that section applies to;

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Shaded boxes indicate the applicable PLC type

- "FX0(S)" - All FX0 and FX0S PLCs
- "FX0N" - All FX0N PLCs
- "FX" - All FX and FX2 PLCs (CPU ver 2.30 or earlier)
- "FX(2C)" - All FX and FX2 PLCs (CPU versions 3.07 or later)
- - All FX2C PLCs (see page 1-4)
- "FX2N(C)" - All FX2N and FX2NC PLCs

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

If an indicator box is half shaded, as shown to the left, this means that not all the functions described in the current section apply to that PLC. The text explains in further detail or makes an independent reference.

If there are no indicator boxes then assume the section applies to all PLC types unless otherwise stated.

3) FX family
This is a generic term which is often used to describe all Programmable Controllers without identifying individual types or model names.

4) CPU version numbers and programming support
As Mitsubishi upgrades each model different versions have different capabilities.
   - Please refer to section 1.4 for details about version numbers and capabilities.
   - Please refer to section 1.5 for details about peripheral support for each model.

1.2 What is a Programmable Controller?

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

A Programmable Logic Controller (PLC or programmable controller) is a device that a user can program to perform a series or sequence of events. These events are triggered by stimuli (usually called inputs) received at the PLC or through delayed actions such as time delays or counted occur-rences. Once an event triggers, it actuates in the outside world by switching ON or OFF electronic control gear or the physical actuation of devices. A programmable controller will continually 'loop' through its internal 'user defined' program waiting for inputs and giving outputs at the programmed specific times.

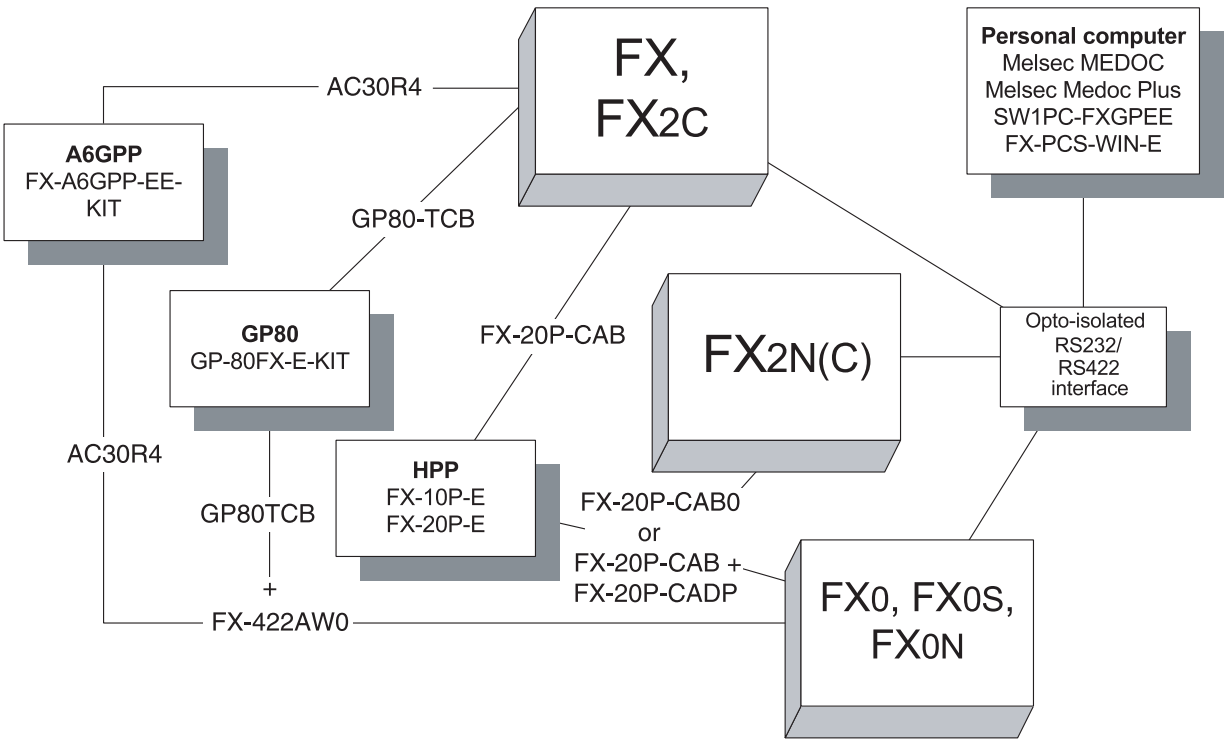
Note on terminology:

The term programmable controller is a generic word used to bring all the elements making the control system under one descriptive name. Sometimes engineers use the term 'Programmable Logic Controller', 'PLC' or 'programmable controller' to describe the same control system.
The construction of a programmable controller can be broken down into component parts. The element where the program is loaded, stored and processed is often known as the Main Processing Unit or MPU. Other terms commonly heard to describe this device are 'base unit', 'controller' and 'CPU'. The term CPU is a little misleading as todays more advanced products may contain local CPU devices. A Main CPU (or more correctly a Main Processing Unit) controls these local CPUs through a communication network or bus.

1.3 What do You Need to Program a PLC?

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

A variety of tools are available to program the Mitsubishi FX family of PLCs. Each of these tools can use and access the instructions and devices listed in this manual for the identified PLC.



1.4 CPU version numbers

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Over time Mitsubishi adds newer and better features to develop and enhance the products. Because of the nature of PLCs, that can be likened to 'industrial computers', changes sometimes occur within the units main CPU (Central Processing Unit). These changes are similar to those experienced by office and home computer users, that is, going to a version up processor. The following lists identify the CPU versions that had significant upgrades or new functions and features added.

1.4.1 FX0N CPU versions

CPU Ver 1.20 The following features were added:
Software control for protocol 1 and 4 communications with the FX0N-485ADP, 1:N network.

CPU Ver 1.40 The following features were added:
Software control for communications using the FX0N-485ADP, peer to peer (N:N) network.

1.4.2 FX and FX2C CPU versions

CPU Ver 3.07 *The following instructions were added:*
ASCI (FNC82), CCD (FNC84), FLT (FNC49), HEX (FNC83), RS (FNC80), SER (FNC61), SORT (FNC69), SQR (FNC48)

The following instructions were upgraded:
EI (FNC04), BMOV (FNC15), HSCS (FNC53), PLSY (FNC57), FMOV (FNC16), MEAN (FNC45), ABSD (FNC62), DSW (FNC72),SEGL (74), PR (FNC 77)

The following device ranges were added:
Input and output devices are independently addressable upto 256 points in software. Total combined input and output points (hardware or software) is 256.

Auxiliary relays increased to 1536 points (M0-M1535)
Data registers increased to 1000 points (D0-D999)
Optional RAM File Registers added, 2000 points (D6000 -D7999)
Pointers increased to 128 points (P0 - P127)

CPU Ver 3.11 *The following instructions were added:*
PID (FNC88)

CPU Ver 3.2 *The following features were added:*
Software control for protocol 4 communications with the FX-485ADP, 1:N network.

CPU Ver 3.30 *The following features were added:*
Software control for protocol 1 communications with the FX-485ADP, 1:N network.

The following instructions were phased out (removed):
ANRD (FNC91), ANWR (FNC92), BLK (FNC97), MCDE (FNC98), MNET (FNC90)

1.5 Special considerations for programming equipment

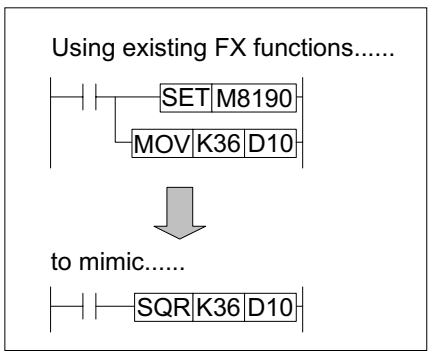
1.5.1 FX CPU version 3.07 or later and FX2C

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Programming tools operating old system software can not access the new features added to the FX CPU from version 3.07 (and available on all FX2C units). However, programming certain 'standard' applied instructions in conjunction with special auxiliary coils (M coils) can achieve the same 'effective instruction' as the new instructions. The following tables identify which version of peripheral software will work directly with all of the 'new' features and which peripheral software versions require use of modified instructions.

Peripherals Table			
Description	**Model Number**	**System software version which will.....**	
		....require the use of auxiliary M coils	**....program all instructions directly**
Hand held programmer (HHP)	FX-10P-E	V 1.10	from V 2.00
HHP cassette	FX-20P-MFXA-E	V 1.20	from V 2.00
Programming software	FX-PCS/AT-E-KIT	V 1.01	from V 2.00
	FX-A6GPP-E-KIT	V 1.00	from V 2.00
Data access units	FX-10DU-E	V 1.10	from V 2.00
	FX-20DU-E	V 1.10	from V 2.00
	Other DU units		from V 1.00

Existing Instruction And Special M Coil Combination To Mimic The Operation Of The Identified Instruction					
Existing FX instruction			**used to mimic the operation of.....**		
Mnemonic	**FNC number**	**Modifying M coil**	**Mimicked instruction**	**Mnemonic**	**FNC Number**
MOV	12	M8190	Square root	SQR	48
MOV	12	M8191	Float	FLT	49
RAMP	67	M8193	Data search	SER	61
RAMP	67	M8194	RS232 instruction	RS	80
FMOV	16	M8196	Hex to ASCII conversion	ASCI	82
FMOV	16	M8197	ASCII to Hex conversion	HEX	83
FMOV	16	M8195	Sum check	CCD	84



Example usage

This format is very important for the instruction to operate correctly.The user must program the 'mimic' instruction with the modifying M coil in a SET instruction immediately before the instruction to be modified.

Using the new Interrupt Pointers:

To program new Interrupt Pointers I010 through I060 in to the HSCS (FNC 53) instruction with older programming equipment, substitute the following special M codes for the appropriate Interrupt Pointer; see the table right.

Existing Instruction And Special M Coil Combination To Mimic The Operation Of The Identified Interrupt pointer	
Existing Auxiliary Coil used to replace the identified Interrupt Pointer	Interrupt Pointer
M8181	I010
M8182	I020
M8183	I030
M8184	I040
M8185	I050
M8186	I060

Using M8198 with the BMOV instruction:

With old software and peripherals, file registers can not be used as a destination device in the BMOV (FNC 15) instruction. To BMOV data into file registers with old equipment set special M coil M8198 on. This switches the source and destination parameters; i.e., the source is then treated as the destination and the destination becomes the source.

General note:

Ignore the special programming techniques identified in this section if using updated programming software or peripherals; then normal operation, as identifiedin the following sections, will apply.

1.5.2 FX2N(C) CPU all versions

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The introduction of this CPU provides the FX user with many new devices and instructions. To use the full features of the FX2N(C) units the user must upgrade older software and hardware programming tools.
However, because of the downward compatibility of the FX2N(C), it is not necessary to upgrade existing programming tools for use with FX2N(C) units up to the equivalent functionality of FX CPU ver 3.30 units.

Peripherals Table		
Description	**Model Number**	**System software version with full support for FX2N(c)**
Hand held programmer (HHP)	FX-10P-E	from V 3.00
HHP cassette	FX-20P-MFXA-E	from V 3.00
Data access units	FX-10DU-E	from V 4.00
	FX-20DU-E	Supports up to FX devices only
	FX-25DU-E	from V 2.00
	FX-30DU-E	from V 3.00
	FX-40DU-E(S)	Supports up to FX devices only
	FX-40DU-TK-ES	from V 3.00
	FX-50DU-TK(S)-E	from V 2.10
	F940GOT-SWD(LWD)-E	All versions

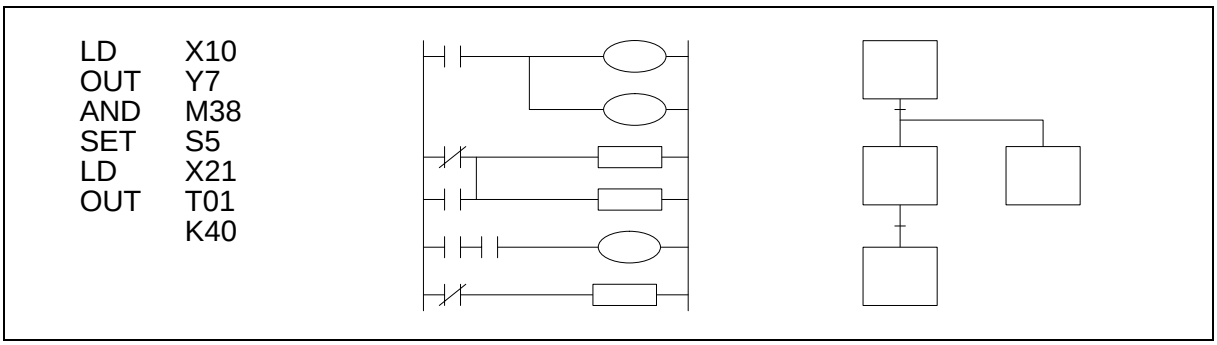
MEMO

Chapter Contents

2. Basic Program Instructions

2.1 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. Not all programming tools can work in all programming forms. Generally hand held programming panels only work with instruction format while most graphic programming tools will work with both instruction and ladder format. Specialist programming software will also allow SFC style programming.



Instruction format Ladder Format SFC Format

2.2 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 4 contains this information in detail. However, the above is all that is required for the rest of this chapter.

2.3 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.

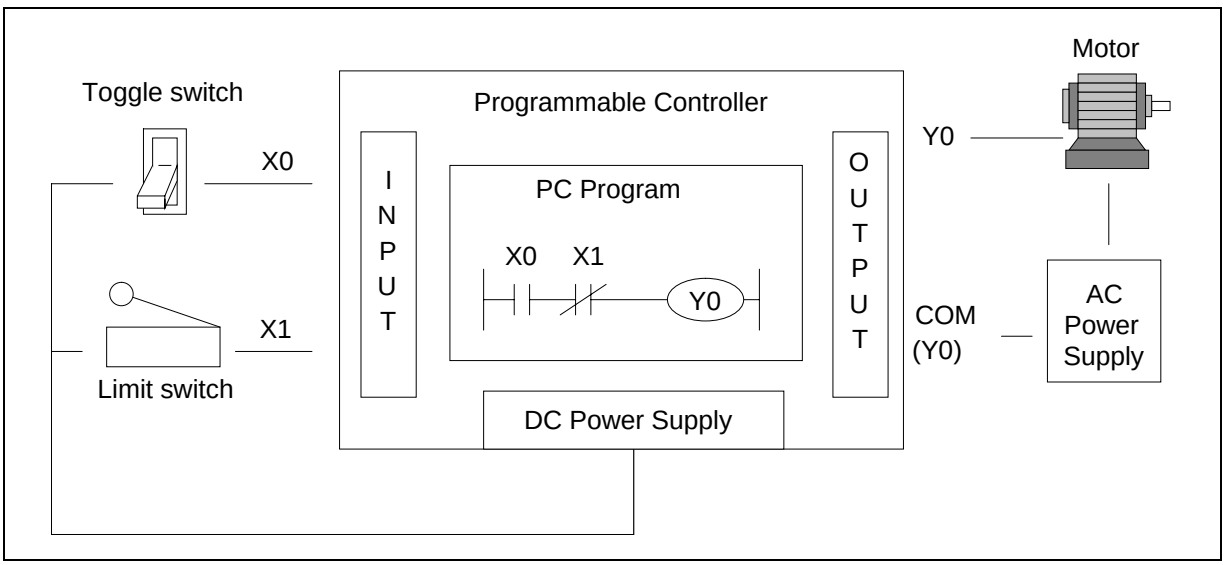
The term “normal(ly)” refers to the status of the contacts when the coil is not energized. Using a relay analogy, when the coil is OFF, a NO contact would have no current flow, that is, a load being supplied through a NO contact would not operate. However, a NC contact would allow current to flow, hence 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).

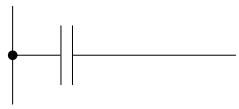
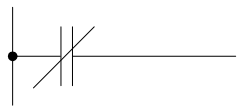
Example:

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 activates, 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.

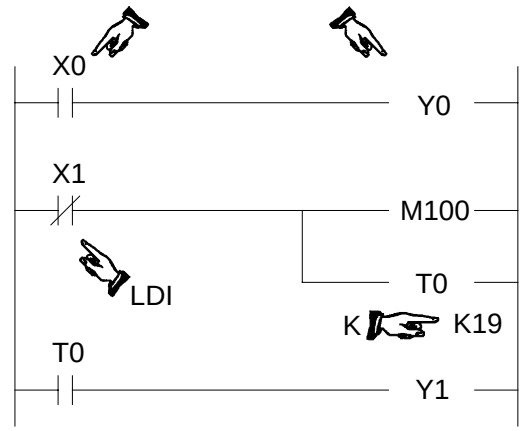


2.4 Load, Load Inverse

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

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

Program example:




```
0   LD    X   0
1   OUT   Y   0
2   LDI   X   1
3   OUT   M   100
4   OUT   T   0
    SP    K   19
7   LD    T   0
8   OUT   Y   1
```

When using hand held programmers, the space key needs to be pressed to enable the constant to be entered.

Basic points to remember:

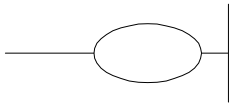
- Connect the LD and LDI instructions directly to the left hand bus bar.
- Or use LD and LDI instructions to define a new block of program when using the ORB and ANB instructions (see later sections).

The OUT instruction:

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

2.5 Out

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

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

Basic points to remember:

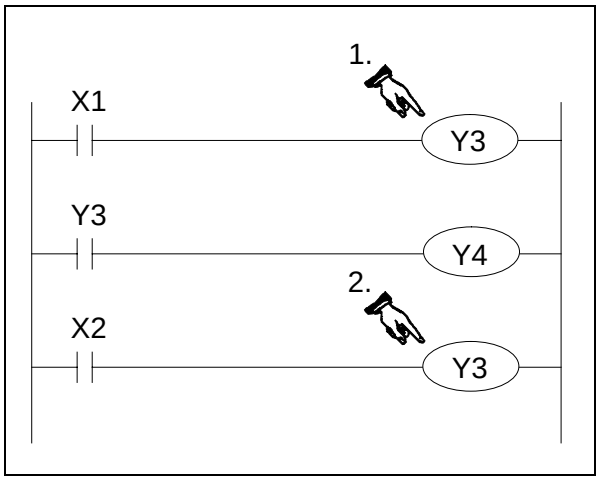
- Connect the OUT instruction directly to the right hand bus bar.
- It is not possible to use the OUT instruction to drive 'X' type input devices.
- It is possible to connect multiple OUT instructions in parallel (for example see the previous page; M100/T0 configuration)

2.5.1 Timer and Counter Variations

When configuring the OUT instruction for use as either a timer (T) or counter (C) a constant must also be entered. The constant is identified by the letter "K" (for example see previous page; T0 K19).
In the case of a timer, the constant "K" holds the duration data for the timer to operate, i.e. if a 100 msec timer has a constant of "K100" it will be (1005 100 msec) 10 seconds before the timer coil activates.
With counters, the constant identifies how many times the counter must be pulsed or triggered before the counter coil activates. For example, a counter with a constant of "8" must be triggered 8 times before the counter coil finally energizes.
The following table identifies some basic parameter data for various timers and counters;

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

2.5.2 Double Coil Designation



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

1.The first Y3 tuns ON because X1 is ON. The contacts associated with Y3 also energize when the coil of output Y3 energizes. Hence, output Y4 turns ON.

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

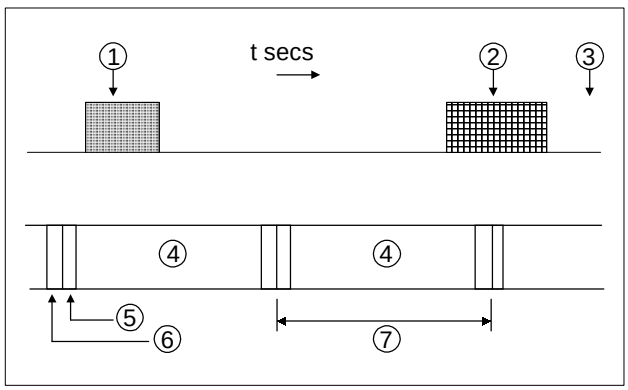
If this is NOT ON then the second Y3 coil does NOT activate. Therefore the status of the Y3 coil updates to reflect this new situation, i.e. it turns OFF. The final outputs are then Y3 = OFF and Y4 = ON.

Use of dual coils:

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

The last coil effect:

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



Input durations:

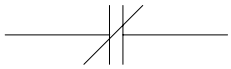
The ON or OFF duration of the PLC inputs must be longer than the operation cycle time of the PLC.
Taking a 10 msec (standard input filter) response delay into account, the ON/OFF duration must be longer than 20 msec if the operation cycle (scan time) is 10 msec.

Therefore, in this example, input pulses of more than 25Hz (1sec/(20msec ON + 20msec OFF)) cannot be sensed.
There are applied instructions provided to handle such high speed input requests.

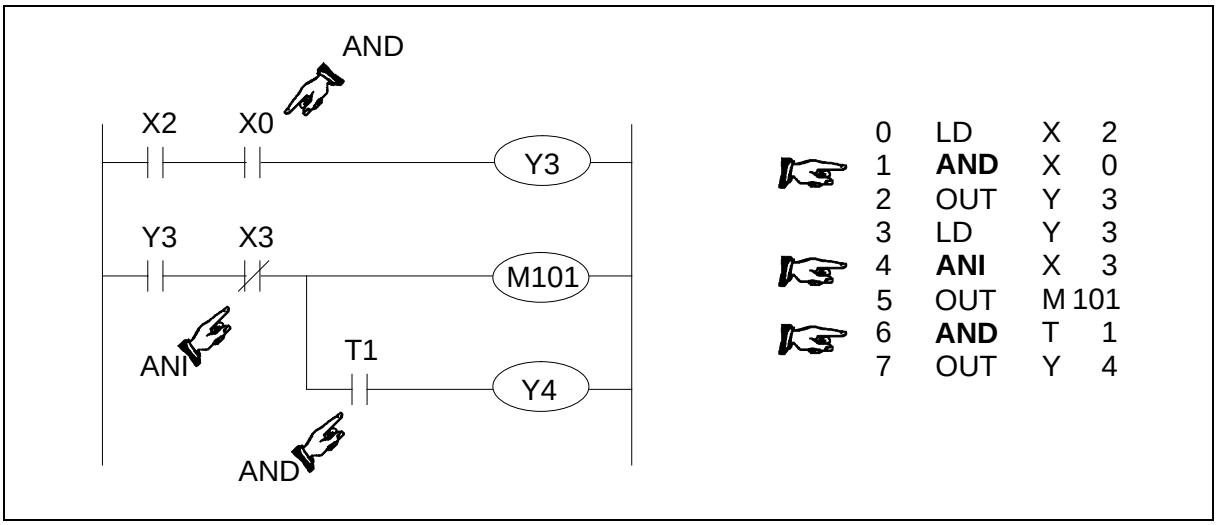
①: Input ON state NOT recognized
②: Input ON state recognized
③: Input OFF state NOT recognized
④: 1 program processing
⑤: Input processing
⑥: Output processing
⑦: A full program scan/operation cycle

2.6 And, And Inverse

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

Program example:



Basic points to remember:

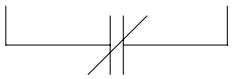
- Use the AND and ANI instructions for serial connection of contacts. As many contacts as required can be connected in series (see following point headed “Peripheral limitations”).
- The output processing to a coil, through a contact, after writing the initial OUT instruction is called a “follow-on”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

Peripheral limitations:

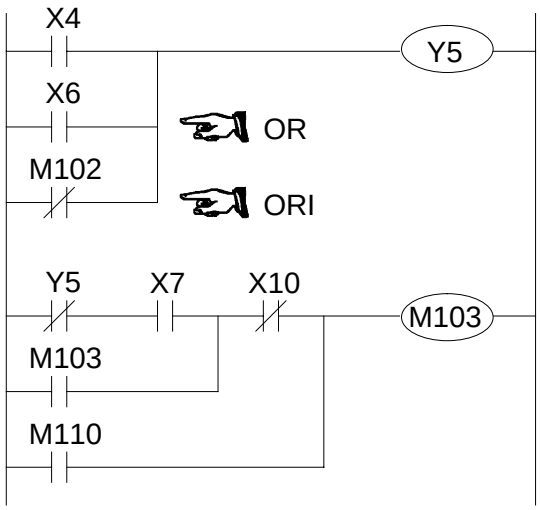
- The PLC has no limit to the number of contacts connected in series or in parallel. However, some programming panels, screens and printers will not be able to display or print the program if it exceeds the limit of the hardware. It is preferable for each line or rung of ladder program to contain up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

2.7 Or, Or Inverse

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

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

Program example:




```
0   LD    X   4
1   OR    X   6
2   ORI   M   102
3   OUT   Y   5
4   LDI   Y   5
5   AND   X   7
6   OR    M   103
7   ANI   X   10
8   OR    M   110
9   OUT   M   103
```

Basic points to remember:

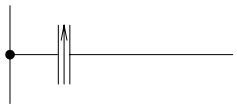
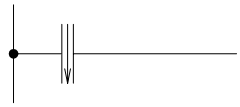
- Use the OR and ORI instructions for parallel connection of contacts. To connect a block that contains more than one contact connected in series to another circuit block in parallel, use an ORB instruction.
- Connect one side of the OR/ORI instruction to the left hand bus bar.

Peripheral limitations:

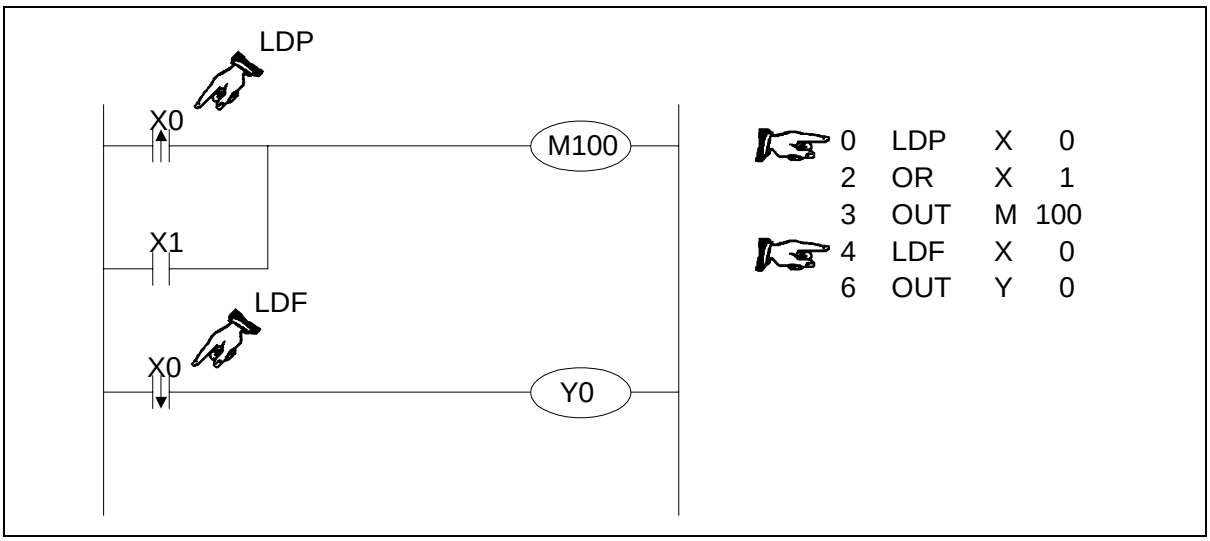
- The PLC has no limit to the number of contacts connected in series or in parallel. However, some programming panels, screens and printers will not be able to display or print the program if it exceeds the limit of the hardware. It is preferable for each line or rung of ladder program to contain up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

2.8 Load Pulse, Load Trailing Pulse

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

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

Program example:



Basic points to remember:

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

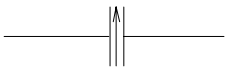
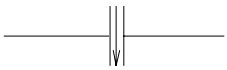
Single Operation flags M2800 to M3071:

- 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.

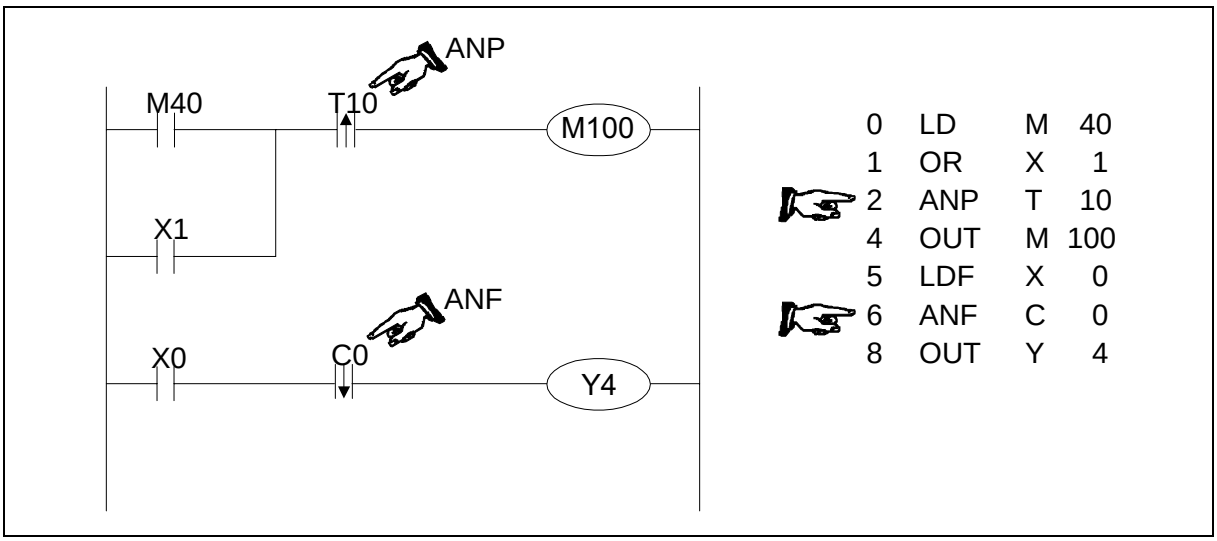
For more details please see page 4-5.

2.9 And Pulse, And Trailing Pulse

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

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

Program example:



Basic points to remember:

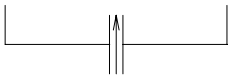
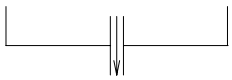
- Use the ANDP and ANDF instructions for the serial connection of pulse contacts.
- Usage is the same as for AND and ANI; see earlier.
- ANP is active for one program scan after the associated device switches from OFF to ON.
- ANF is active for one program scan after the associated device switches from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

Single operation flags M2800 to M3071:

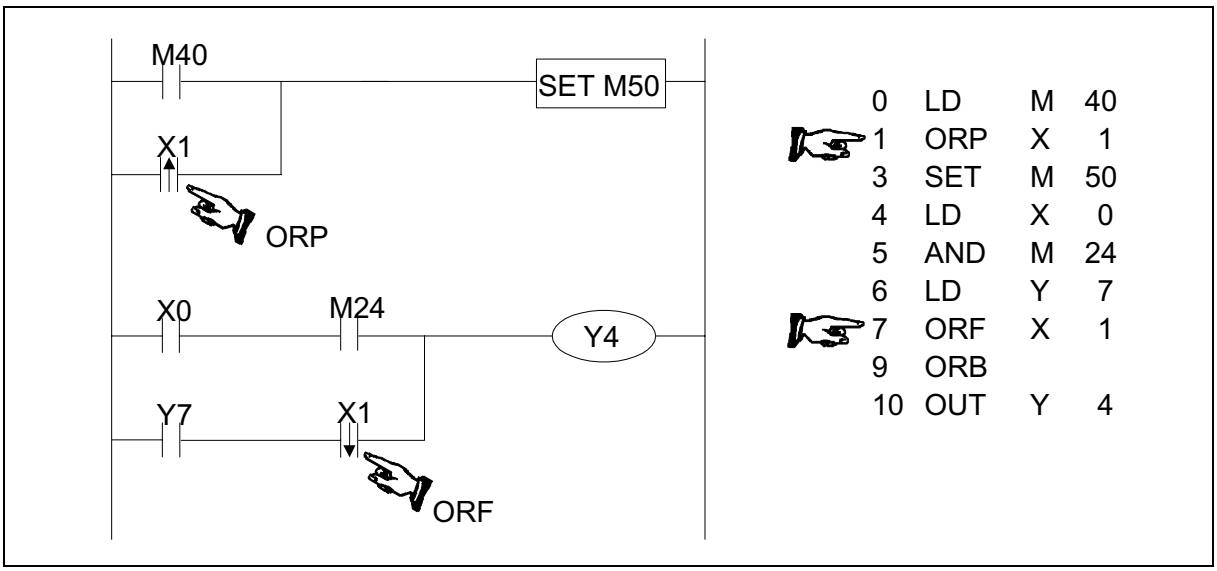
- When used with flags M2800 to M3071 only the first instruction will activate. For details see page 2-8

2.10 Or Pulse, Or Trailing Pulse

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

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

Program example:



Basic points to remember:

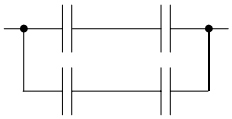
- Use the ORP and ORF instructions for the parallel connection of pulse contacts.
- Usage is the same as for OR and ORI; see earlier.
- ORP is active for one program scan after the associated device switches from OFF to ON.
- ORF is active for one program scan after the associated device switches from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

Single operation flags M2800 to M3071:

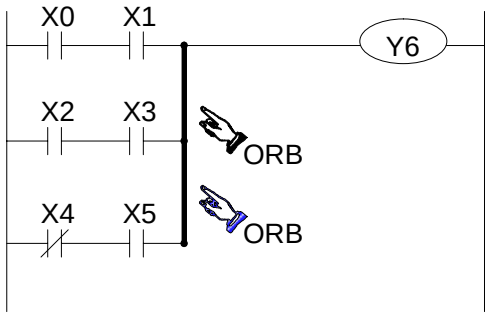
- When used with flags M2800 to M3071 only the first instruction will activate. For details see page 2-8

2.11 Or Block

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

Program example:



Recommended sequential programming method

```
0  LD   X 0
1  AND  X 1
2  LD   X 2
3  AND  X 3
4  ORB
5  LDI  X 4
6  AND  X 5
7  ORB
8  OUT  Y 6
```

Non-preferred batch programming method

```
0  LD   X 0
1  AND  X 1
2  LD   X 2
3  AND  X 3
4  LDI  X 4
5  AND  X 5
6  ORB
7  ORB
8  OUT  Y 6
```

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:

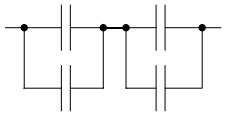
- When using ORB instructions in a batch, use no more than 8 LD and LDI instructions in the definition of the program blocks (to be connected in parallel). Ignoring this will result in a program error (see the right most program listing).

Sequential processing limitations:

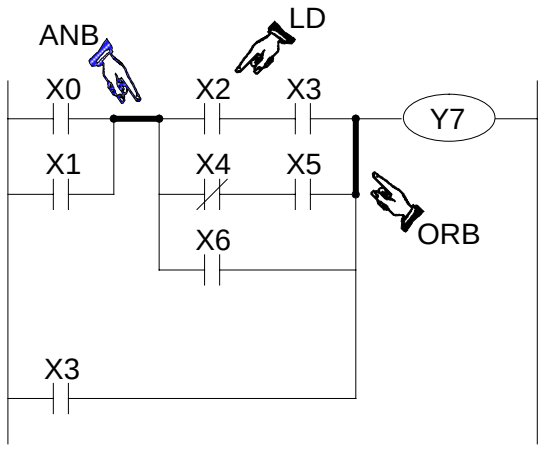
- There are no limitations to the number of parallel circuits when using an ORB instruction in the sequential processing configuration (see the left most program listing).

2.12 And Block

FX0(S) FX0N FX FX(2C) FX2N(C)

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

Program example:



Recommended sequential programming method

	0	LD	X	0
	1	OR	X	1
	2	LD	X	2
	3	AND	X	3
	4	LDI	X	4
	5	AND	X	5
	6	ORB		
	7	OR	X	6
☞	8	**ANB**		
	9	OR	X	3
	10	OUT	Y	7

Basic points to remember:

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

Batch processing limitations:

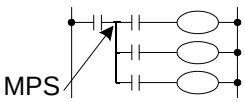
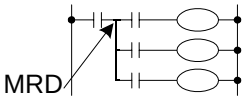
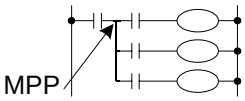
- When using ANB instructions in a batch, use no more than 8 LD and LDI instructions in the definition of the program blocks (to be connected in parallel). Ignoring this will result in a program error (see ORB explanation for example).

Sequential processing limitations:

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

2.13 MPS, MRD and MPP

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

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

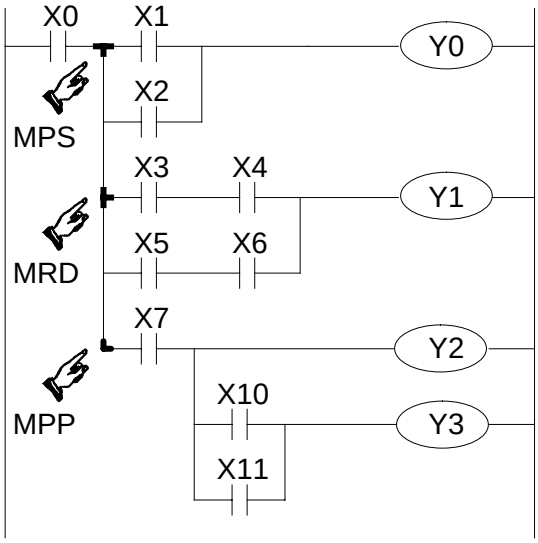
Basic points to remember:

- Use these instructions to connect output coils to the left hand side of a contact. Without these instructions connections can only be made to the right hand side of the last contact.
- MPS stores the connection point of the ladder circuit so that further coil branches can recall the value later.
- MRD recalls or reads the previously stored connection point data and forces the next contact to connect to it.
- MPP pops (recalls and removes) the stored connection point. First, it connects the next contact, then it removes the point from the temporary storage area.
- For every MPS instruction there MUST be a corresponding MPP instruction.
- The last contact or coil circuit must connect to an MPP instruction.
- At any programming step, the number of active MPS-MPP pairs must be no greater than 11.

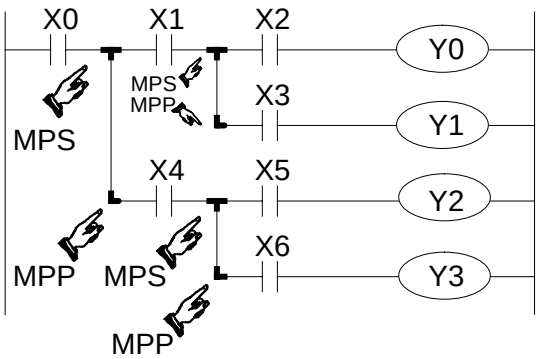
MPS, MRD and MPP usage:

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

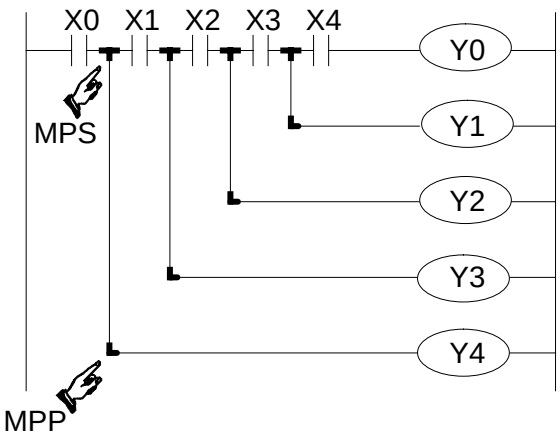
Multiple program examples:




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




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



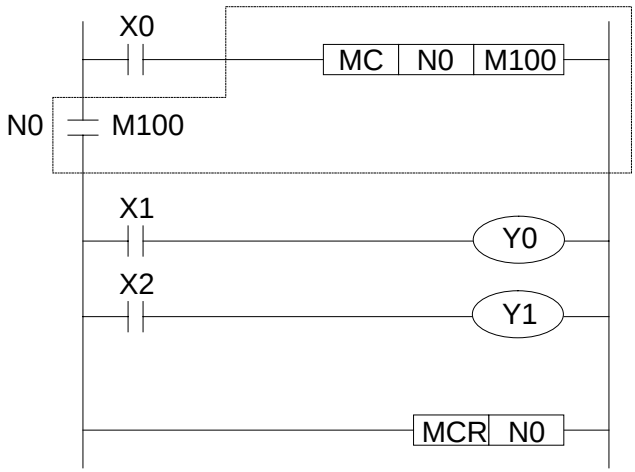

```
 0  LD    X  0
 1  MPS
 2  AND   X  1
 3  MPS
 4  AND   X  2
 5  MPS
 6  AND   X  3
 7  MPS
 8  AND   X  4
 9  OUT   Y  0
10  MPP
11  OUT   Y  1
12  MPP
13  OUT   Y  2
14  MPP
15  OUT   Y  3
16  MPP
17  OUT   Y  4
```

2.14 Master Control and Reset

FX0(S) FX0N FX FX(2C) FX2N(C)

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

Program example:

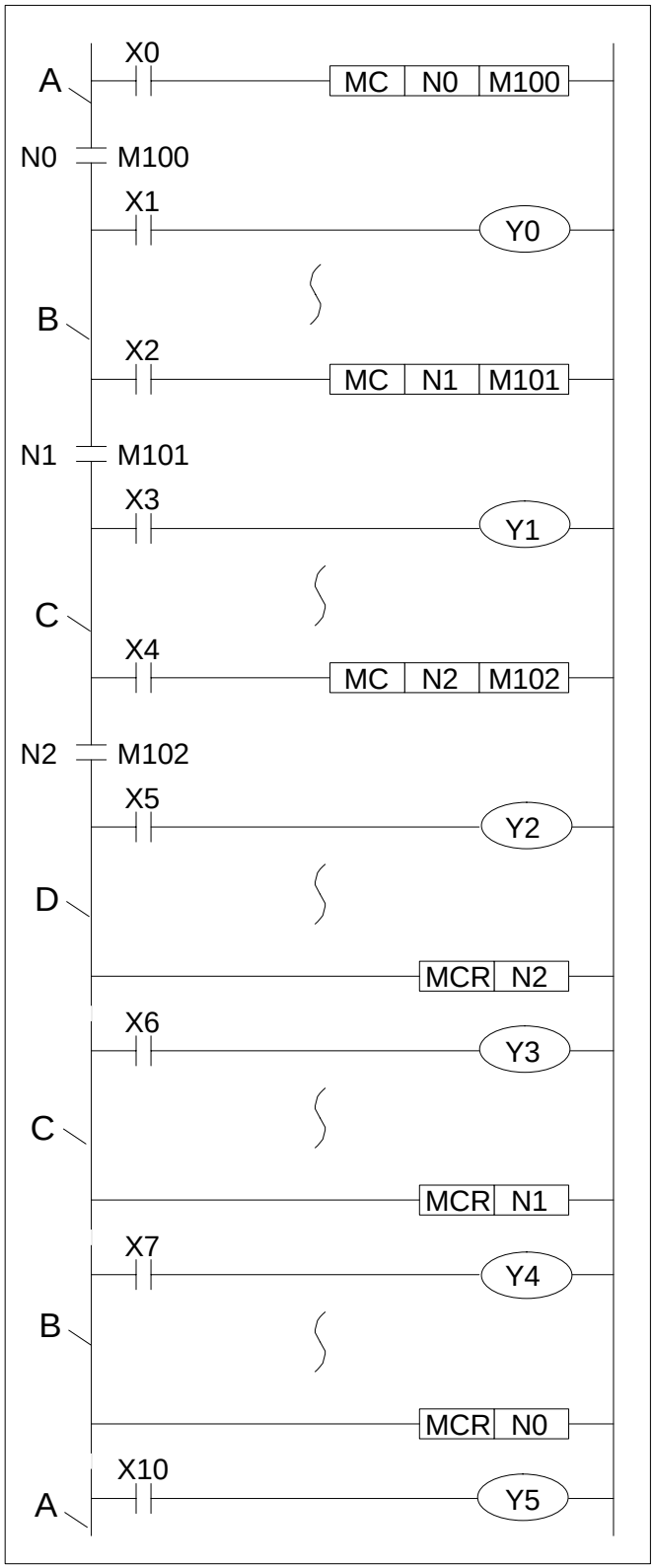



```
0   LD    X   0
1   MC    N   0
      SP      M 100
4   LD    X   1
5   OUT   Y   0
6   LD    X   2
7   OUT   Y   1
8   MCR   N   0
```

Note: SP - space key
N - nest level of MC (N0 to N7)

Basic points to remember:

- After the execution of an MC instruction, the bus line (LD, LDI point) shifts to a point after the MC instruction. An MCR instruction returns this to the original bus line.
- The MC instruction also includes a nest level pointer N. Nest levels are from the range N0 to N7 (8 points). The top nest level is ‘0’ and the deepest is ‘7’.
- The MCR instruction resets each nest level. When a nest level is reset, it also resets ALL deeper nest levels. For example, MCR N5 resets nest levels 5 to 7.
- When input X0=ON, all instructions between the MC and the MCR instruction execute.
- When input X0=OFF, none of the instruction between the MC and MCR instruction execute; this resets all devices except for retentive timers, counters and devices driven by SET/RST instructions.
- The MC instruction can be used as many times as necessary, by changing the device number Y and M. Using the same device number twice is processed as a double coil (see section 2.5.2). Nest levels can be duplicated but when the nest level resets, ALL occurrences of that level reset and not just the one specified in the local MC.



Nested MC program example:

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

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

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

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

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

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

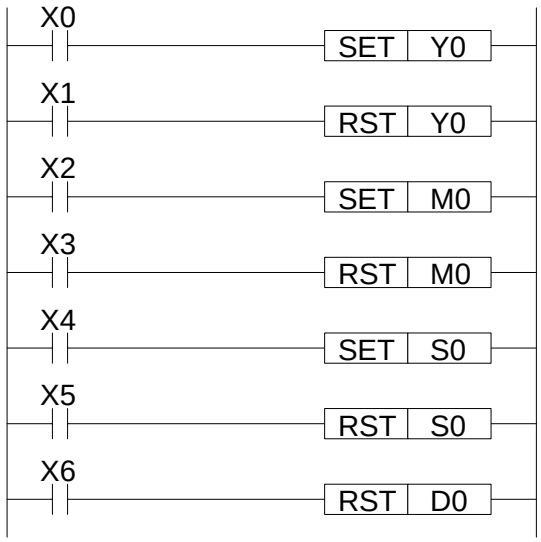
Output Y5 turns ON/OFF according to the ON/OFF state of X10, regardless of the ON/OFF status of inputs X0, X2 or X4.

2.15 Set and Reset

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

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

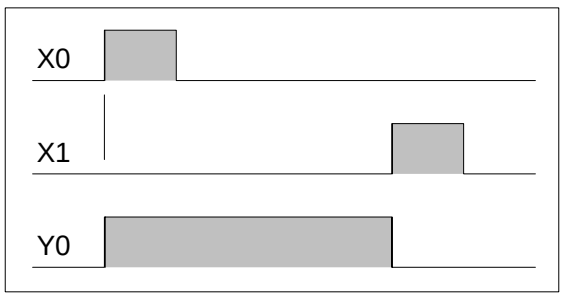
Program example:




```
0  LD   X 0
1  SET  Y 0
2  LD   X 1
3  RST  Y 0
4  LD   X 2
5  SET  M 0
6  LD   X 3
7  RST  M 0
8  LD   X 4
9  SET  S 0
10 LD   X 5
11 RST  S 0
12 LD   X 6
13 RST  D 0
```

Basic points to remember:

- Turning ON X0 causes Y0 to turn ON. Y0 remains ON even after X0 turns OFF.
- Turning ON X1 causes Y0 to turn OFF. Y0 remains OFF even after X1 turns OFF.
- SET and RST instructions can be used for the same device as many times as necessary. However, the last instruction activated determines the current status.



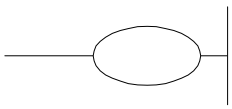
- It is also possible to use the RST instruction to reset the contents of data devices such as data registers, index registers etc. The effect is similar to moving ‘K0’ into the data device.

Resetting timers and counters:

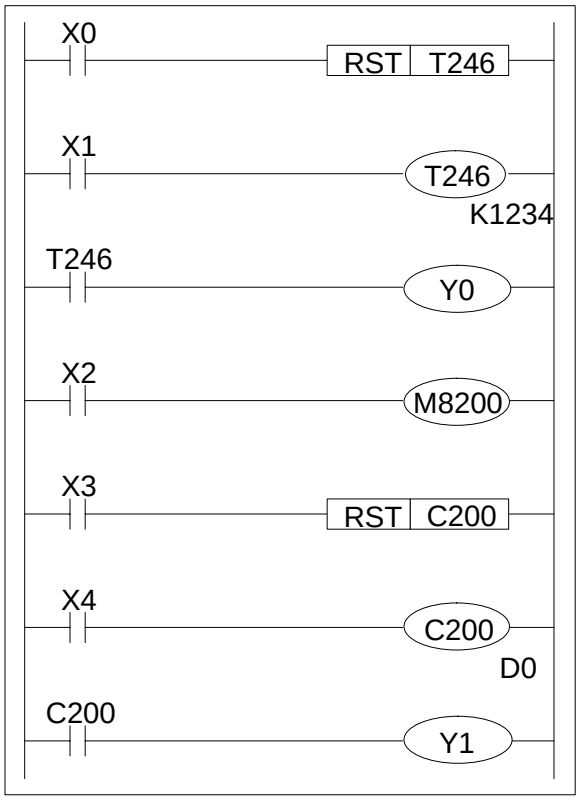
- Please see next page.

2.16 Timer, Counter (Out & Reset)

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Mnemonic	Function	Format	Devices	Program steps
OUT *(OUT)*	Driving timer or counter coils		T, C	*32 bit counters:5 Others: 3*
RST *(ReSeT)*	Resets timer and counter, coils contacts and current values		T, C (see section 2.15 for other resetable devices)	*T, C:2*

Program example:



2.16.1Basic Timers, Retentive Timers And Counters

These devices can all be reset at any time by driving the RST instruction (with the number of the device to be reset).
On resetting, all active contacts, coils and current value registers are reset for the selected device. In the example, T246, a 1msec retentive timer, is activate while X1 is ON. When the current value of T246 reaches the preset 'K' value, i.e. 1234, the timer coil for T246 will be activated. This drives the NO contact ON. Hence, Y0 is switched ON. Turning ON X0 will reset timer T246 in the manner described previously.
Because the T246 contacts are reset, the output Y0 will be turned OFF.

Retentive timers:

- For more information on retentive timers please see page 4-17.

2.16.2 Normal 32 bit Counters

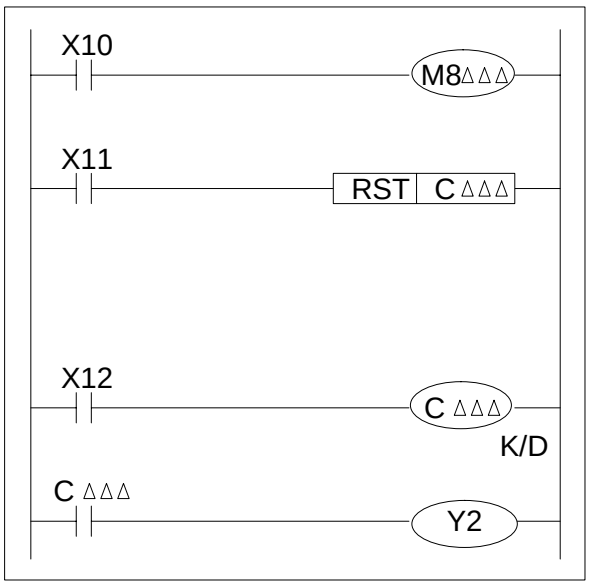
The 32 bit counter C200 counts (up-count, down-count) according to the ON/OFF state of M8200. In the example program shown on the previous page C200 is being used to count the number of OFF ~ ON cycles of input X4.
The output contact is set or reset depending on the direction of the count, upon reaching a value equal (in this example) to the contents of data registers D1,D0 (32 bit setting data is required for a 32 bit counter).
The output contact is reset and the current value of the counter is reset to '0' when input X3 is turned ON.

32 bit counters:

- For more information on 32 bit counters please see page 4-21.

2.16.3 High Speed Counters

High speed counters have selectable count directions. The directions are selected by driving the appropriate special auxiliary M coil. The example shown to the right works in the following manner; when X10 is ON, counting down takes place. When X10 is OFF counting up takes place.
In the example the output contacts of counter CΔΔΔ and its associated current count values are reset to "0" when X11 is turned ON. When X12 is turned ON the driven counter is enabled. This means it will be able to start counting its assigned input signal (this will not be X12 - high speed counters are assigned special input signals, please see page 4-22).



Availability of devices:

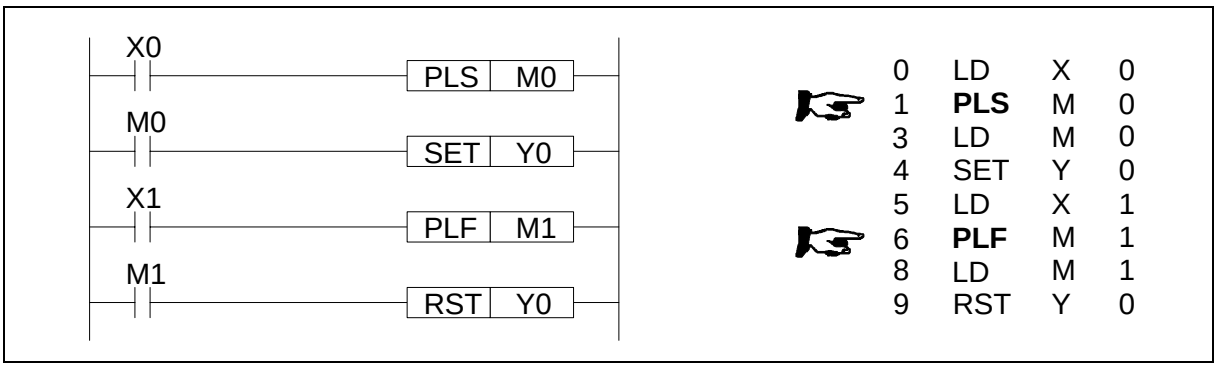
- Not all devices identified here are available on all programmable controllers. Ranges of active devices may vary from PLC to PLC. Please check the specific availability of these devices on the selected PLC before use. For more information on high speed counters please see page 4-22. For PLC device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

2.17 Leading and Trailing Pulse

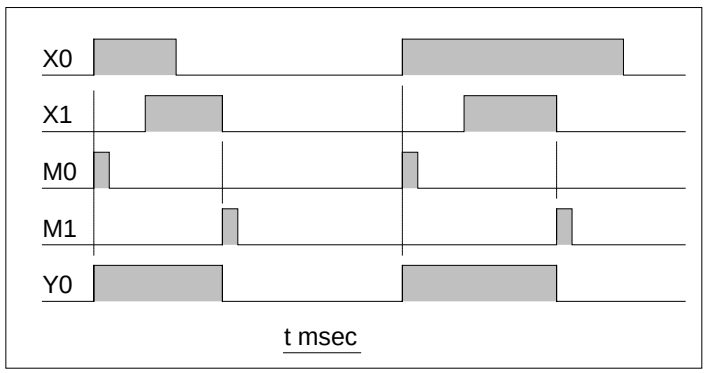
FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

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

Program example:



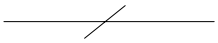
Basic points to remember:



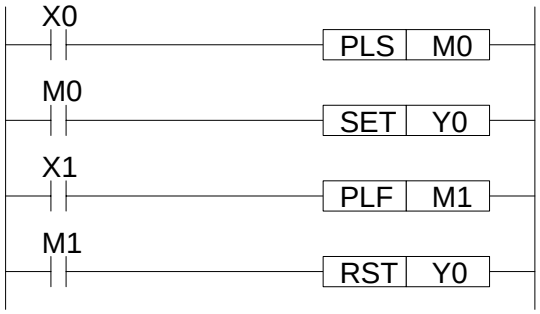
- When a PLS instruction is executed, object devices Y and M operate for one operation cycle after the drive input signal has turned ON.
- When a PLF instruction is executed, object devices Y and M operate for one operation cycle after the drive input signal has turned OFF.
- When the PLC status is changed 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.

2.18 Inverse

FX0(S) FX0N FX FX(2C) FX2N(C)

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

Program example:




```
0  LD   X  0
1  PLS  M  0
3  LD   M  0
4  SET  Y  0
5  LD   X  1
6  PLF  M  1
8  LD   M  1
9  RST  Y  0
```

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.

2.19 No Operation

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

Basic points to remember:

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

2.20 End

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Format	Devices	Program steps
END *(END)*	Forces the current program scan to end	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.

MEMO

Chapter Contents

3. STL Programming

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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.

3.1 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 Funktionplan) 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)

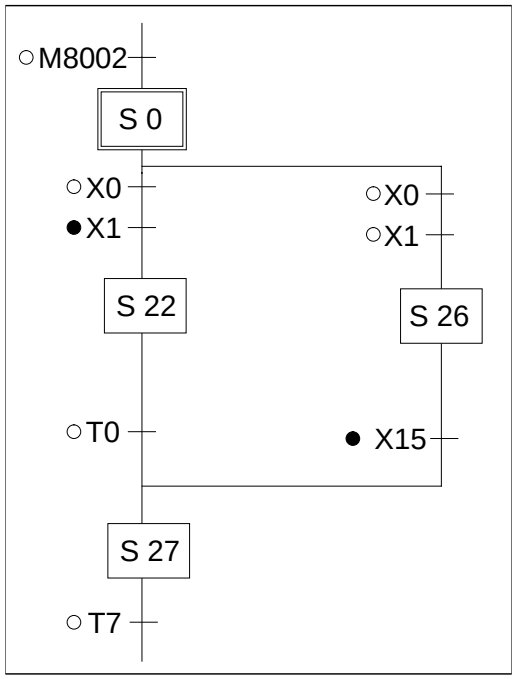
Examples of these programming methods can be seen on page 2-1.

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

3.2 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).



3.2.1 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

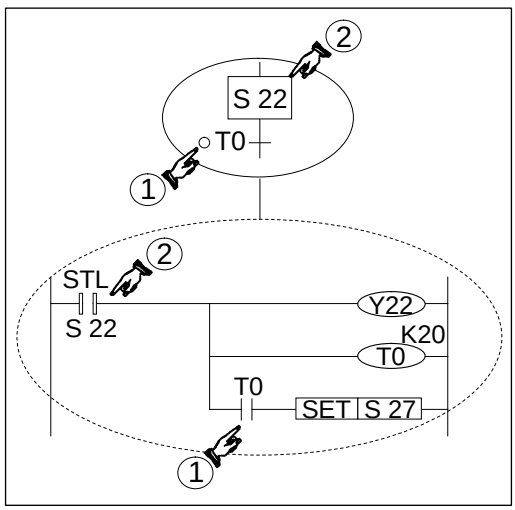
On initial inspection the STL program looks as if it is a rather basic flow diagram. But to find out what is really happening the STL state needs to be put 'under a microscope' so to speak. When a single state is examined in more detail, the sub-program can be viewed.

With the exception of the STL instruction, it will be immediately seen that the STL sub-program looks just like ordinary programming.

① 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.

② 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'.

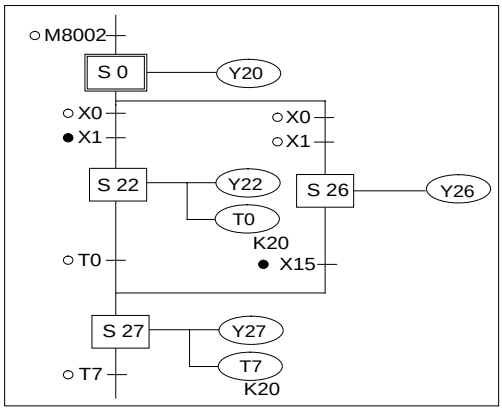


Combined SFC Ladder representation

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.

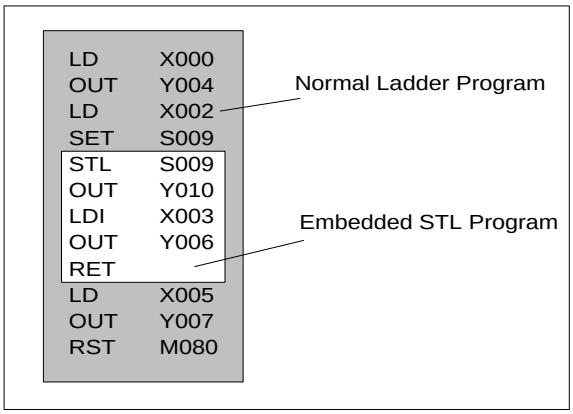


3.3 How To Start And End An STL Program

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.

3.3.1 Embedded STL programs

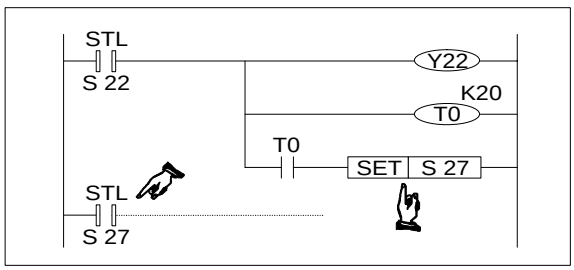
An STL style program does not have to entirely replace a standard ladder logic program. In fact it might be very difficult to do so. Instead small or even large section of STL program can be entered at any point in a program. Once the STL task has been completed the program must go back to processing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



3.3.2 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.

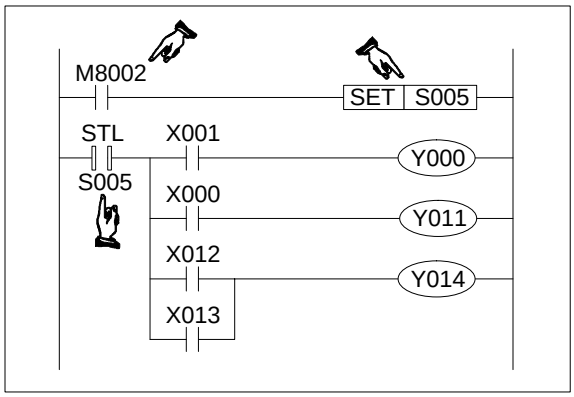
There are many different methods to drive a state, for example the initial state coils could be pulsed, SET or just included in an OUT instruction. However, within Mitsubishi’s STL programming language an STL coil which is SET has a different meaning than one that is included in an OUT instruction.



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.

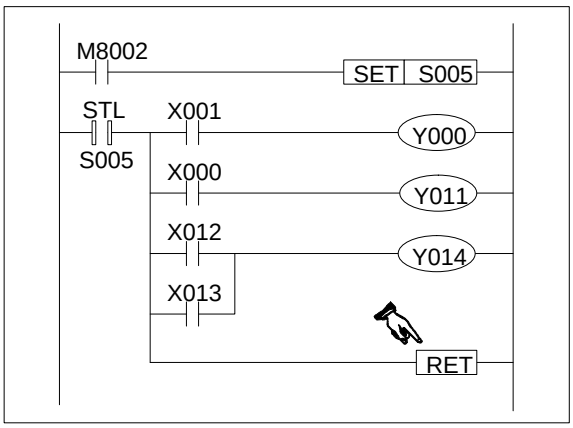


3.3.3 Terminating an STL Program

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.

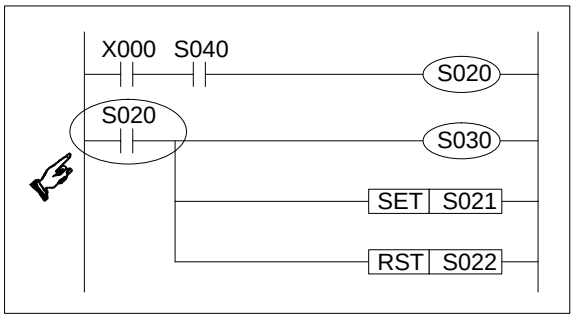
3.4 Moving Between STL Steps

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 difference will now be explained:

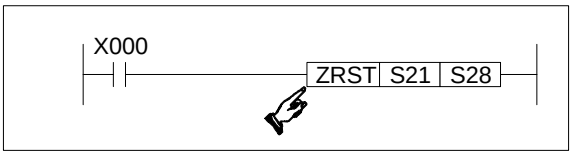
3.4.1 Using SET to drive an STL coil

- 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 (FNC 40) instructions would perform this task.



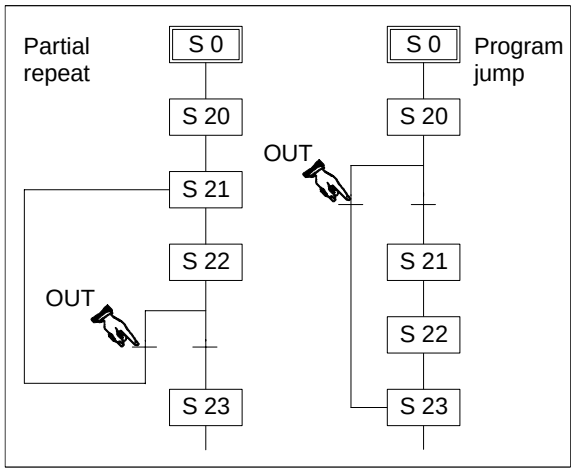
- SET is used to drive an immediately following STL step which typically will have a larger STL state number than the current step.
- SET is used to drive STL states which occur within the enclosed STL program flow, i.e. SET is not used to activate a state which appears in an unconnected, second STL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

3.4.2 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’.

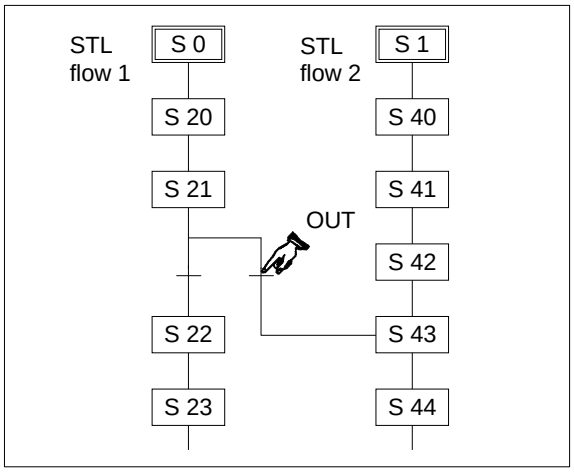
OUT is used for loops and jumps

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.

3.5 Rules and Techniques For STL programs

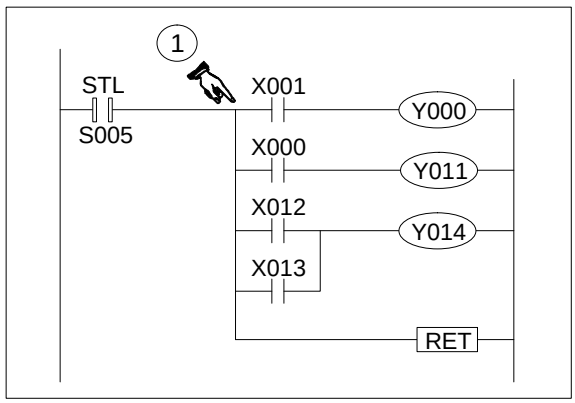
It can be seen that there are a lot of advantages to using STL style programming but there are a few points a user must be aware of when writing the STL sub-programs.
These are highlighted in this section.

3.5.1 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.

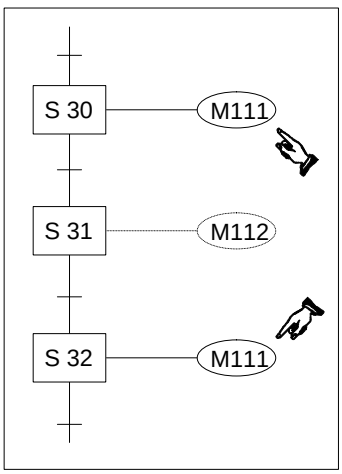
  Each STL step makes its own bus bar. This means that a user, cannot use an MPS instruction directly after the STL instruction (see ①), i.e. There needs to be at least a single contact before the MPS instruction.



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

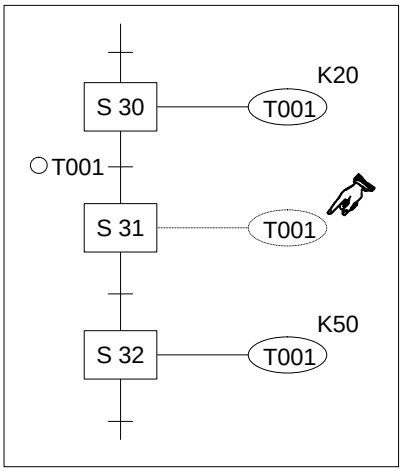
- In normal programming using dual coils is not an acceptable technique. However repetition of a coil in separate STL program blocks is allowed.

  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 M111 used twice in a single STL flow.



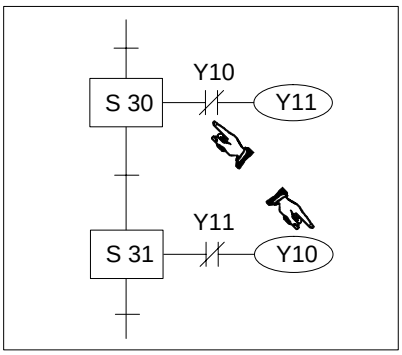
Caution: The same coil should NOT be programmed in steps that will be active at the same time as this will result in the same problem as other dual coils.

- 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.



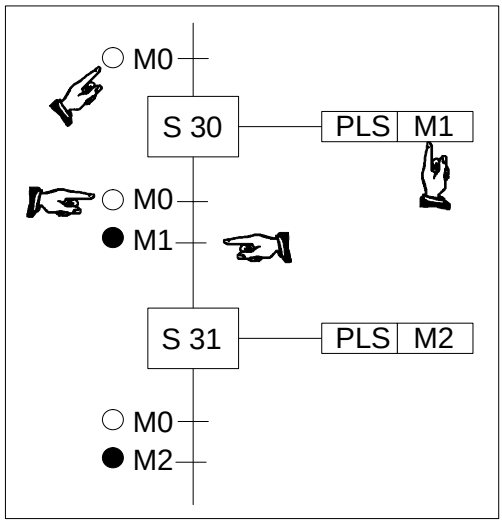
3.5.2 Single Signal Step Control

Transferring between active STL steps can be controlled by a single signal. There are two methods the user can program to achieve this result.

Method 1 - Using locking devices

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

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.



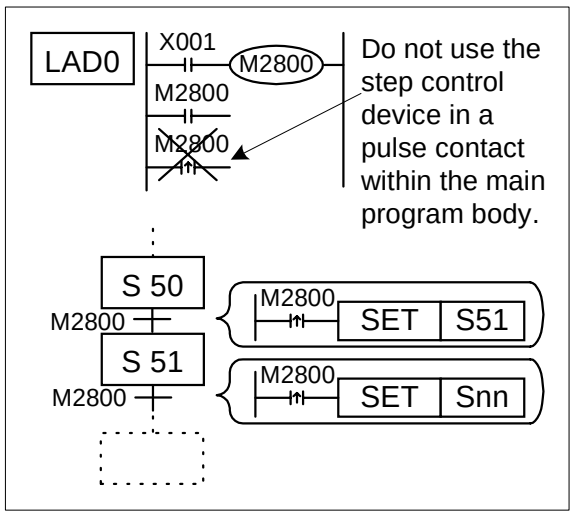
- S30 is activated when M0 is first pulsed ON.
- The operation of M1 prevents the sequence from continuing because although M0 is ON, the transfer requirements, need M0 to be ON and M1 to be OFF.
- After one scan the pulsed M0 and the 'lock' device M1 are reset.
- On the next pulse of M0 the STL step 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!

Method 2 - Special Single Pulse Flags

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Using the pulse contacts (LDP, LDF, ANP, etc.) and a special range of M devices (M2800 to M3071) the FX2N(C) PLC's achieves the same result as method 1. The special feature of these devices prevents run through of the states, as only the first occurrence of the LDP instruction will activate.

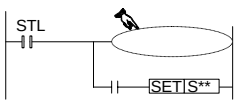
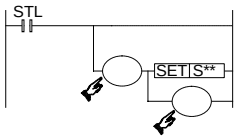
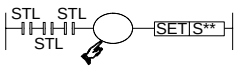
The example program below shows the necessary instructions.



- Assume S50 is already active.
- When X01 activates M2800, this in turn activates the LDP M2800 instruction in S50 and the flow moves on to step S51.
- The LDP M2800 instruction in the transition part of S51 does not execute because this is the second occurrence of M2800 in a pulse contact.
- When X01 next activates M2800, the LDP instruction in S51 is the first occurrence because S50 is now inactive. Thus, control passes to the next step in the same manner.

3.6 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, AND, ANI, OR,ORI, NOP, OUT, SET, RST, PLS,PLF	ANB, ORB, MPS,MRD, MPP	MC, MCR
Initial and general states			✓	✓	✗
Branch-ing and merging states	Output processing		✓	✓	✗
	Transfer processing		✓	✗	✗

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.

For more details please see the operational compatibility listed in the two tables on pages 7-12,7-13.

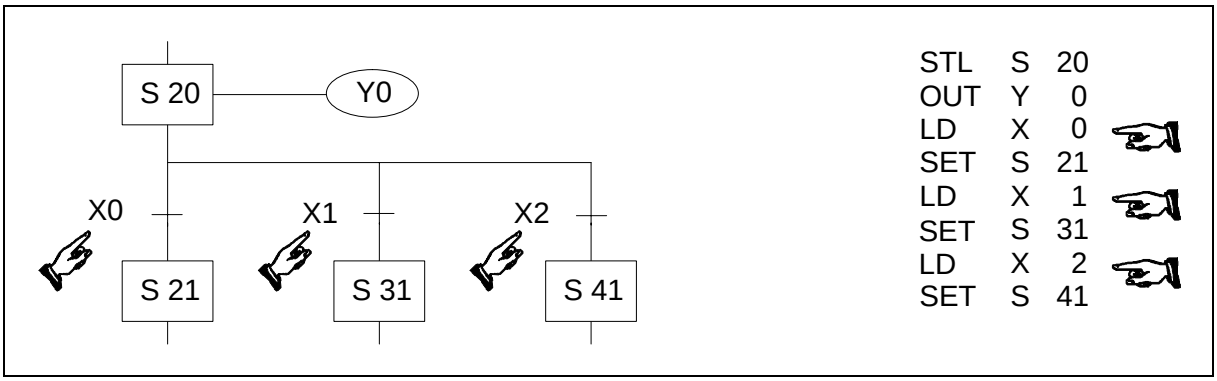
Using ‘jump’ operations with STL

- 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.

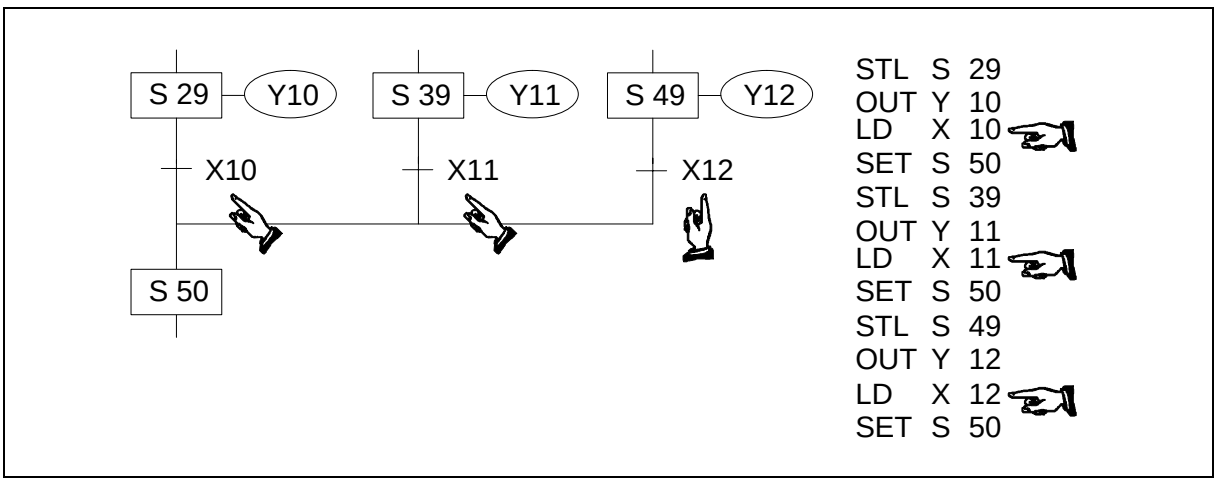
3.7 Using STL To Select The Most Appropriate Program

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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.

Limits on the number of branches

- Please see page 3-14 for general notes on programming STL branches.

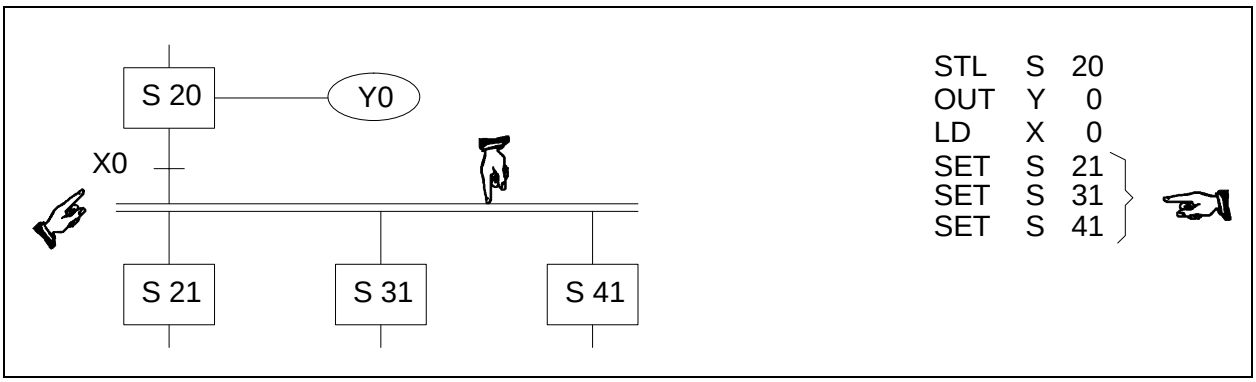
Notes on using the FX-PCS/AT-EE software

- Please see page 3-15 for precautions when using the FX-PCS-AT/EE software.

3.8 Using STL To Activate Multiple Flows Simultaneously

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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.

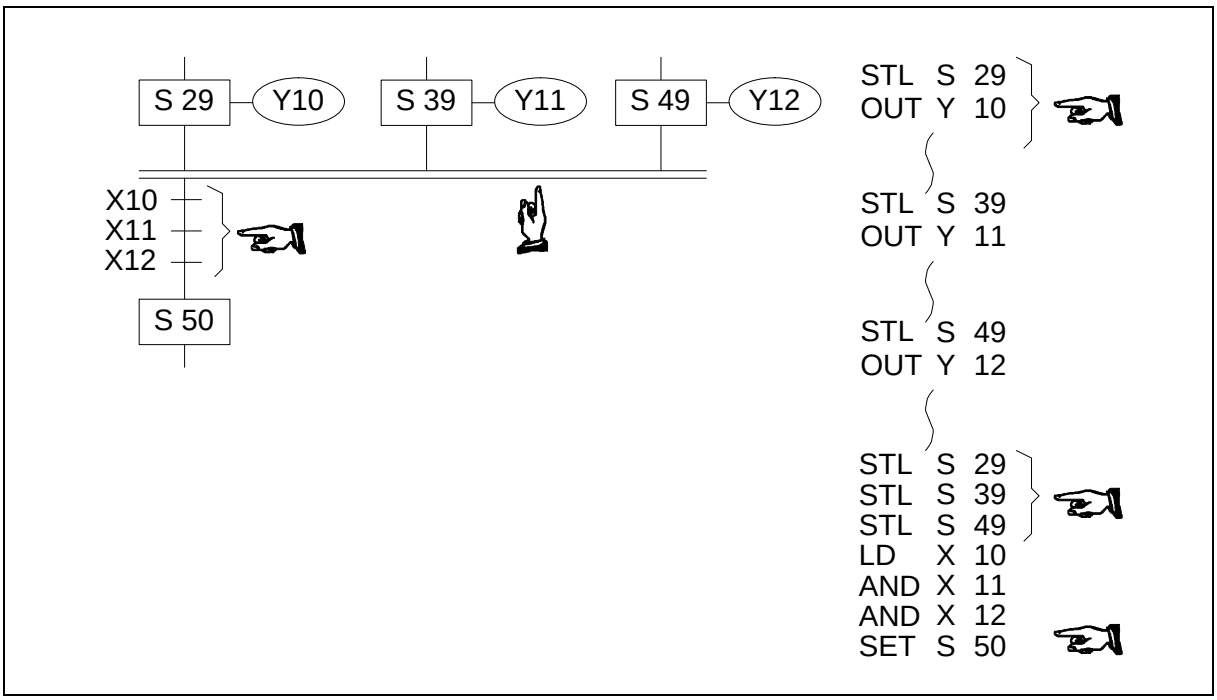


When a group of branch flows are activated, the user will often either;

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 also be active, in this example these are X10, X11 and X12. Once all states and input conditions are made the merging or joining state can be SET ON. As is the general case, all of the states used in the setting procedure are reset automatically.



Because more than one state is being simultaneously joined with further states (some times described as a parallel merge), a set of horizontal parallel lines are used to aid a quick visual recognition.

Limits on the number of branches

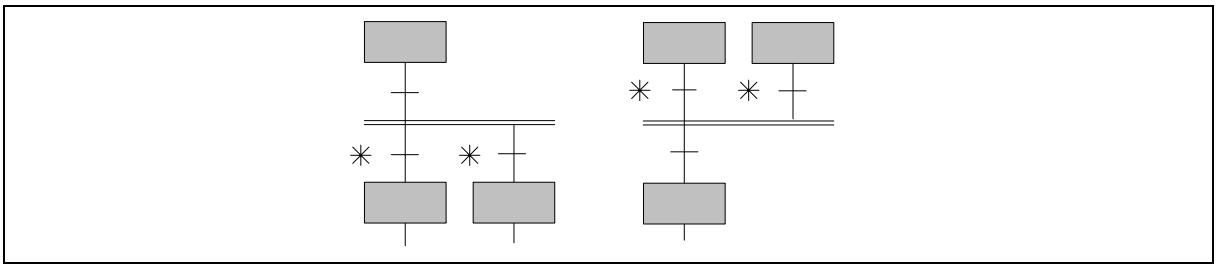
- Please see page 3-14 for general notes on programming STL branches.

Notes on using the FX-PCS/AT-EE software

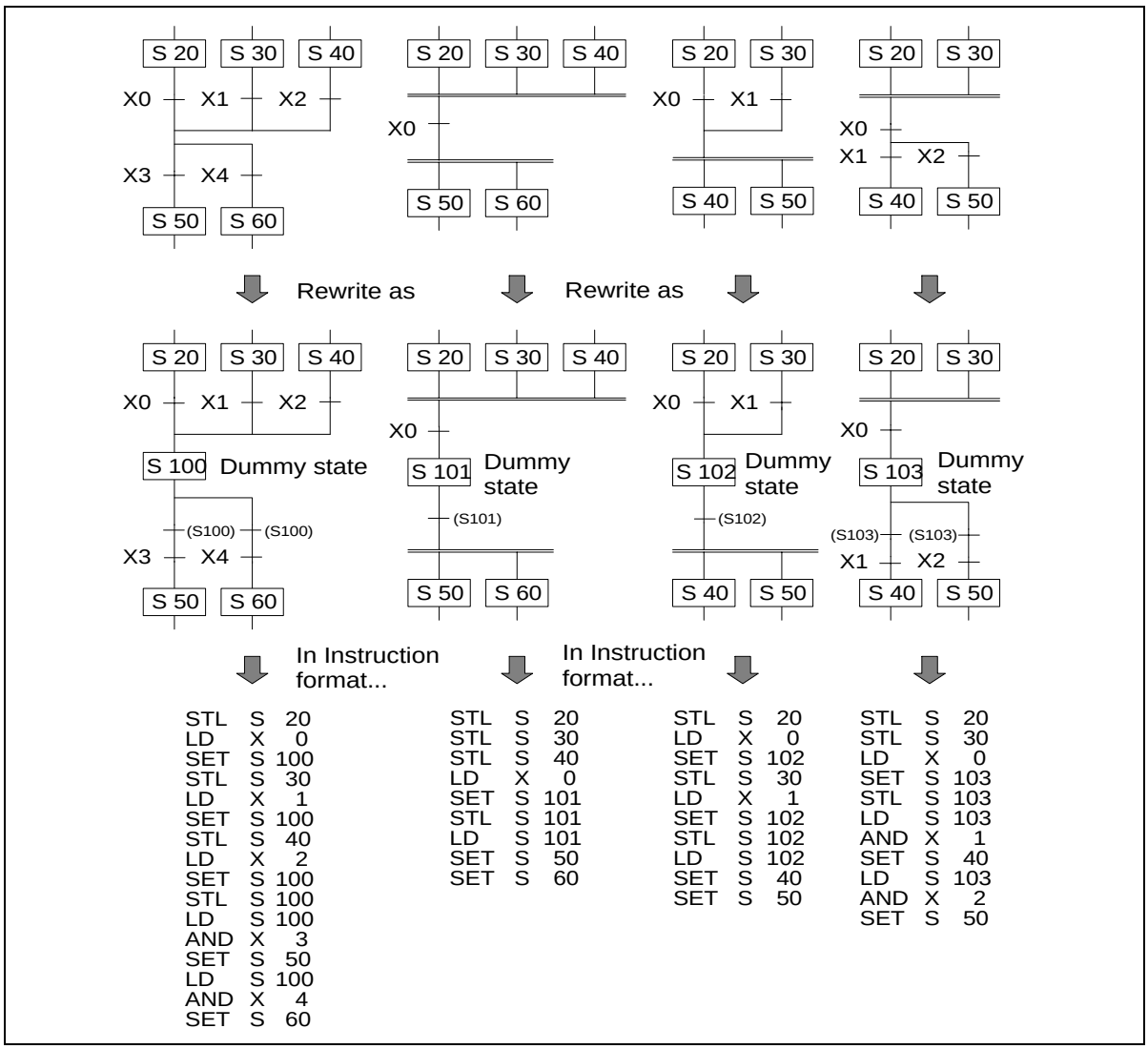
- Please see page 3-15 for precautions when using the FX-PCS-AT/EE software.

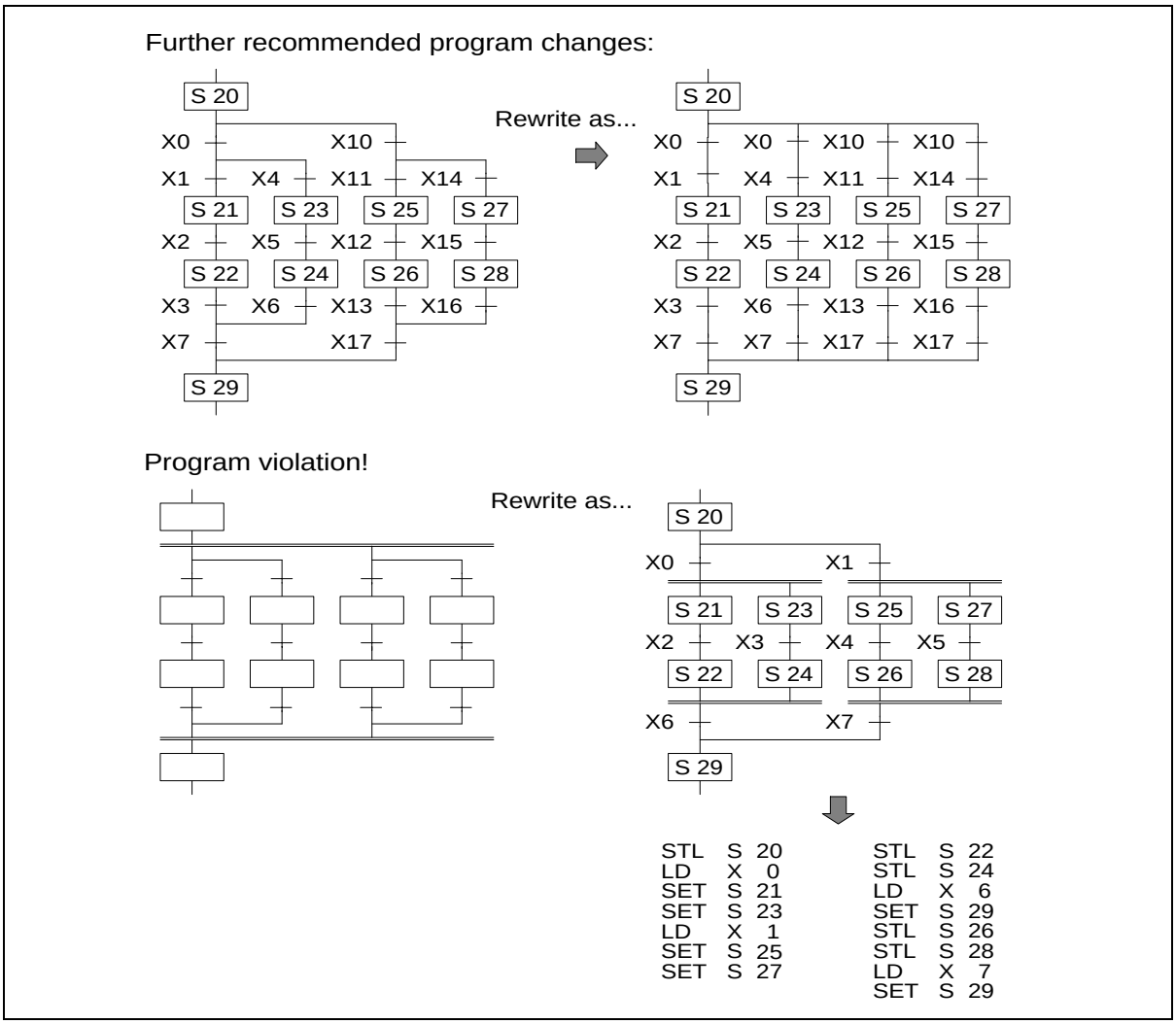
3.9 General Rules For Successful STL Branching

For each branch point 8 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 8 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 FX-PCS/AT-EE 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:





3.10 General Precautions When Using The FX-PCS/AT-EE Software

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

This software has the ability to program in SFC flow diagrams. As part of this ability it can read and convert existing STL programs back into SFC flows even if they were never originally programmed using the FX-PCS/AT-EE software. As an aid to allowing this automatic SFC flow generation the following rules and points should be noted:

1) When an STL flow is started it should be initialized with one of the state devices from the range S0 to S9.

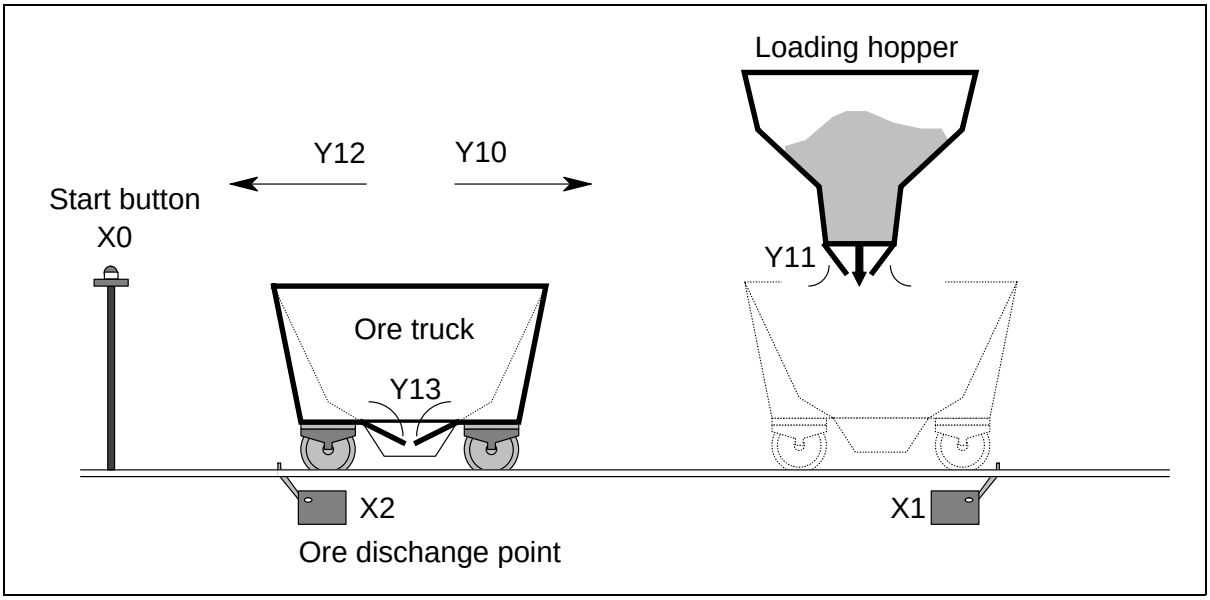
2) Branch selection or merging should always be written sequentially moving from left to right. This was demonstrated on page 3-11, i.e. on the selective branch S21 was specified before S31 which was specified before S41. The merge states were programmed in a similar manner, S29 proceeded S39 which proceeded S49.

3) The total number of branches which can be programmed with the STL programming mode are limited to a maximum of 16 circuits for an STL flow. Each branch point is limited to a maximum of 8 branching flows. This means two branch points both of 8 branch flows would equal the restriction. These restrictions are to ensure that the user can always view the STL flow diagram on the computer running the FX-PCS-AT/ EE software and that when it is needed, the STL program flow can be printed out clearly.

3.11 Programming Examples

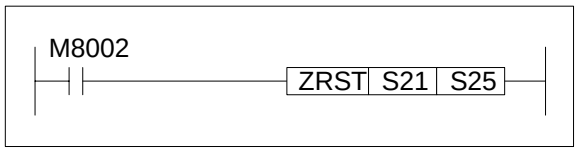
FX0(S) FX0N FX FX(2C) FX2N(C)

3.11.1 A Simple STL Flow



This simple example is an excerpt from a semi-automatic loading-unloading ore truck program. This example program has a built in, initialization routine which occurs only when the PLC is powered from OFF to ON. This is achieved by using the special auxiliary relay M8002.

This activates a Zone ReSeT (ZRST is applied instruction 40) instruction which ensures all of the operational STL states within the program are reset. The program example opposite shows an M8002/ZRST example.

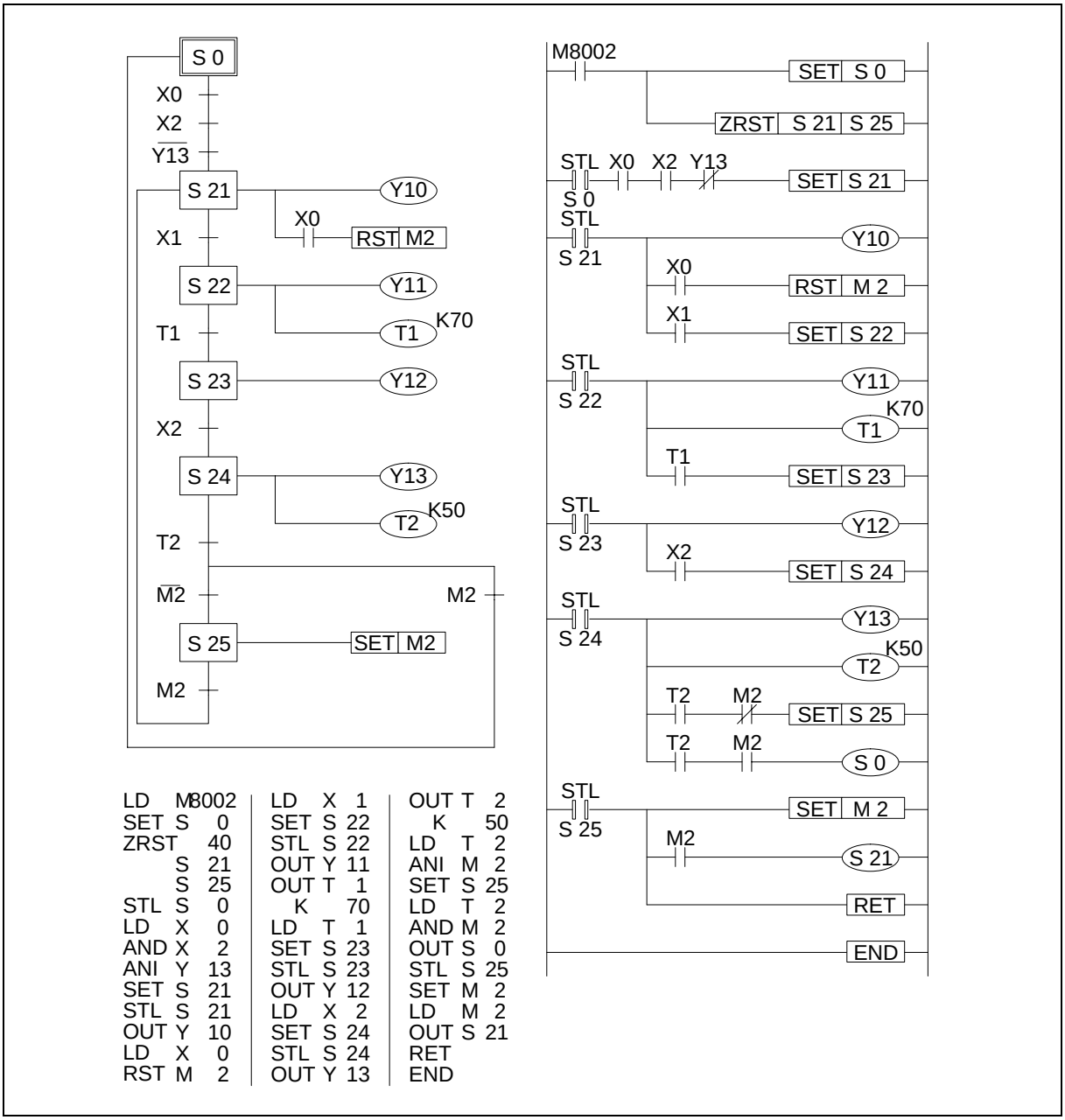


The push button X0 acts as a start button and a mode selection button. The STL state S0 is initialized with the ZRST instruction. The system waits until inputs X0 and X2 are given and Y 13 is not active. In the scenario this means the ore truck is positioned at the ore discharge point, i.e. above the position sensor X2. The ore truck is not currently discharging its load, i.e. the signal to open the trucks unloading doors (Y13) is not active and the start button (X0) has been given. Once all of the points have been met the program steps on to state S21.

On this state the ore cart is moved (Y10) and positioned (X1) at the loading hopper. If the start button (X0) is pressed during this stage the ore cart will be set into a repeat mode (M2 is reset) where the ore truck is immediately returned to the loading hopper after discharging its current load. This repeat mode must be selected on every return to the loading station.

Once at the loading point the program steps onto state S22. This state opens the hoppers doors (Y11) and fills the truck with ore. After a timed duration, state S23 is activated and the truck returns (Y12) to the discharge point (X2).

Once at the discharge point the truck opens its bottom doors (Y13). After a timed duration in which the truck empties its contents, the program checks to see if the repeat mode was selected on the last cycle, i.e. M2 is reset. If M2 was reset (in state S21) the program 'jumps' to step S21 and the ore truck is returned for immediate refilling. If M2 is not reset, i.e. it is active, the program cycles back to STL state S0 where the ore truck will wait until the start push button is given.
This is a simple program and is by no means complete but it identifies the way a series of tasks have been mapped to an STL flow.



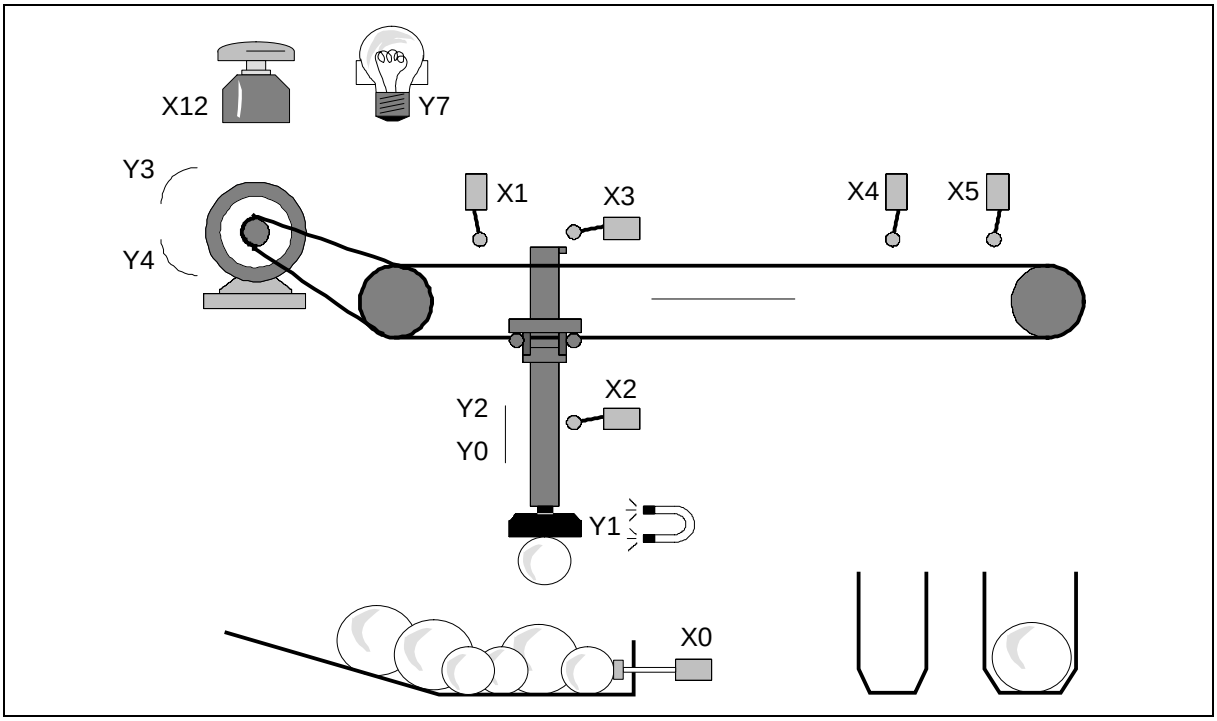

```
LD   M8002    LD   X   1    OUT T   2
SET  S   0    SET  S  22      K    50
ZRST     40   STL  S  22    LD   T   2
         21   OUT  Y  11    ANI  M   2
         25   OUT  T   1    SET  S  25
STL  S   0      K    70    LD   T   2
LD   X   0    LD   T   1    AND  M   2
AND  X   2    SET  S  23    OUT  S   0
ANI  Y  13    STL  S  23    STL  S  25
SET  S  21    OUT  Y  12    SET  M   2
STL  S  21    LD   X   2    LD   M   2
OUT  Y  10    SET  S  24    OUT  S  21
LD   X   0    STL  S  24    RET
RST  M   2    OUT  Y  13    END
```

Identification of normally closed contacts

This example has used the line convention to identify normally closed contacts, for further variations and different methods used to perform this task please see the information note page 3-3.

3.11.2 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.



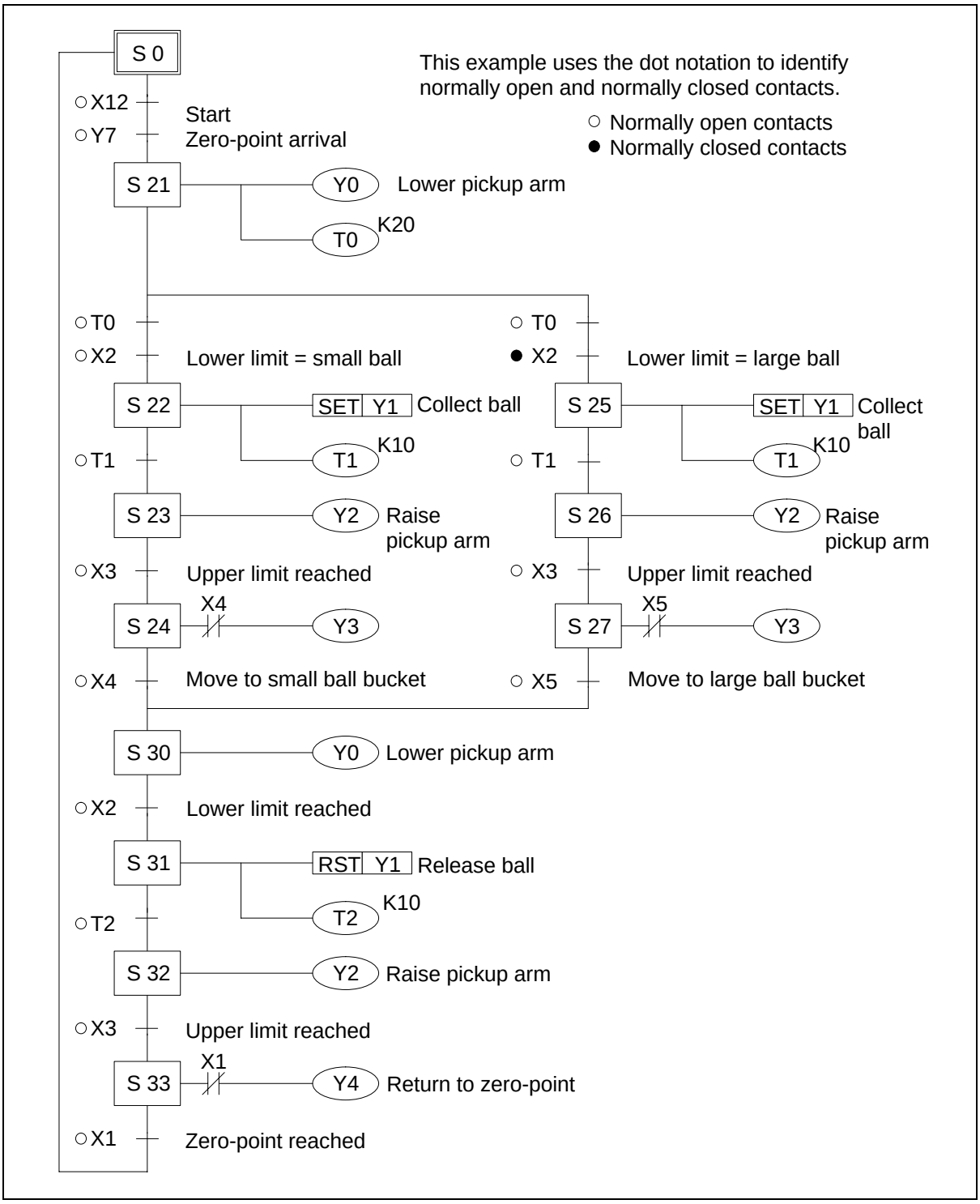
The sequence of physical events (from initial power On) are:

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.



3.12 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

More details on this instruction can be found on page 5-67.

Chapter Contents

4. Devices in Detail

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

4.1 Inputs

Device Mnemonic: X

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

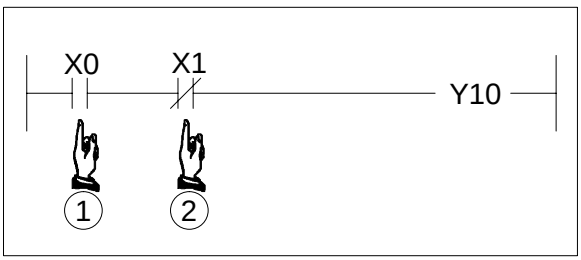
Alias: I/P
Inp
(X) Input
Input contact

Available forms: NO (①) and NC (②) contacts only
(see example device usage for references)

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

Further uses: None

Example device usage:



Available devices:

- Please see the information point on page 4-2, Outputs. Alternatively refer to the relevant tables for the selected PLC in chapter 8.

Configuration details:

- Please see chapter 9

4.2 Outputs

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Device Mnemonic: Y

Purpose: Representation of physical outputs from the programmable controller

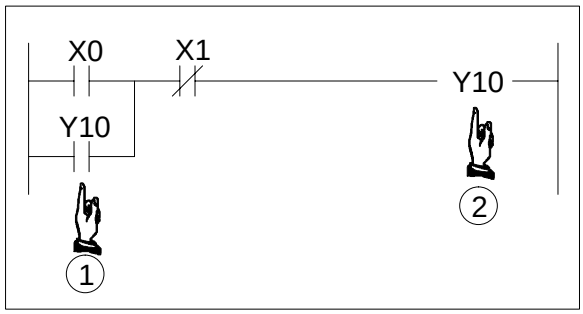
Alias: O/P
Otp
Out (Y)
Output (Y)
Output (coil/ relay/ contact)

Available forms: NO (①) and NC contacts and output coils (②)
(see example device usage for references)

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

Further uses: None

Example device usage:



Available devices:

PLC	Maximum number of inputs	Maximum number of outputs	Absolute total available I/O
FX0(S)	Set by selected base unit		30
FX0N	84 Max. input config'	(40)	128
	(60)	64 Max. output config'	
FX	128	128	256
FX(2C)	256 (addressable in software)	256 (addressable in software)	256 (Total addressed in software/hardware)
FX2N(C)			

- Please note, these are all the absolute maximums which are available. The values are subject to variations caused by unit selection. For configuration details please see chapter 9.
- For more information about the device availability for individual PLC's, please see chapter 8.

4.3 Auxiliary Relays

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Device Mnemonic: M

Purpose: Internal programmable controller status flag

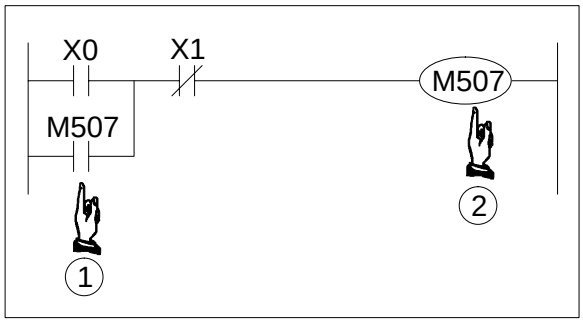
Alias: Auxiliary (coil/ relay/ contact/ flag)
M (coil/ relay/ contact /flag)
M (bit) device

Available forms: NO (①) and NC contacts and output coils (②)
(see example device usage for references)

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

Further uses: General stable state auxiliary relays - see page 4-3
Battery backed/ latched auxiliary relays - see page 4-4
Special diagnostic auxiliary relays - see page 4-5

Example device usage:



4.3.1 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.

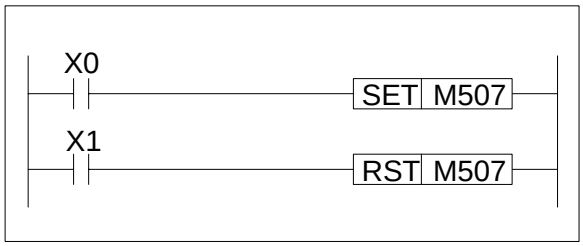
Available devices:

PLC	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)
General auxiliary relays	496 (M0 - 495)	384 (M0 - 383)	500 (M0 - 499)	500 (M0 - 499)	500 (M0 - 499)
Battery backed/ latched relays	16 (M496 - 511)	128 (M384 - 511)	524 (M500 - 1023)	1036 (M500 - 1535)	2572 (M500 - 3071)
Total available	512	512	1024	1536	3072

- For more information about device availability for individual PLC's, please see chapter 8. For device availability when using an FX fitted with an FX2-40AW/AP please see page 9-6.

4.3.2 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 circuit shown on page 4-3 is an example of 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.

4.3.3 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

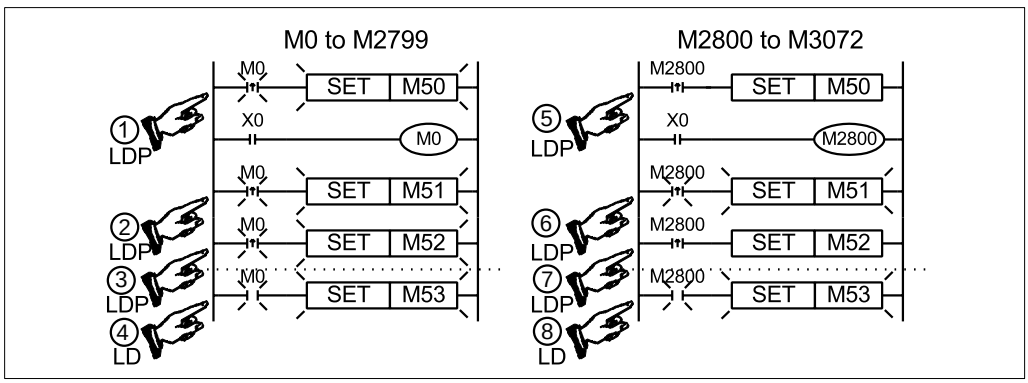
Available devices:

- 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. PLC specific diagnostic ranges and meanings are available in chapter 6.

4.3.4 Special Single Operation Pulse Relays

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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.

Turning ON X0 causes M2800 to turn ON.

- Contact ⑥ is a pulse contact and activates for 1 scan.
- Contacts ⑤ and ⑦ are pulse contacts of the same M device as contact ⑥. Contact ⑥ has already operated, so contact ⑤ and ⑦ do not operate.
- Contact ⑧ is a normal LD contact and activates while M2800 is ON.

4.4 State Relays

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Device Mnemonic: S

Purpose: Internal programmable controller status flag

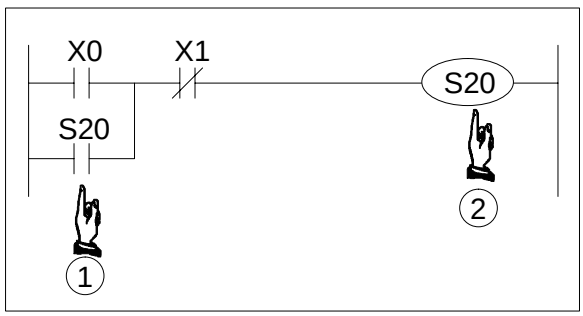
Alias: State (coil/ relay/ contact/ flag)
S (coil/ relay/ contact /flag)
STL step (coil/ relay/ contact /flag)
Annunciator flag

Available forms: NO (①) and NC contacts and output coils (②)
(see example device usage for references)

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

Further uses: General stable state - state relays - see page 4-6
Battery backed/ latched state relays - see page 4-7
STL step relays - see page 4-8
Annunciator flags - see page 4-9

Example device usage:



4.4.1 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 on page 4-7 ‘Battery backed/ latched state relays’, or see the relevant tables for the selected PLC in chapter 8.

4.4.2 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:

PLC	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)
General state relays	64 (S0 - 63)	N/A	500 (S0 - 499)		
Battery backed/ latched relays	N/A	128 (S0 - 127)	500 (S500 - 999)		
Total available	64	128	1000		

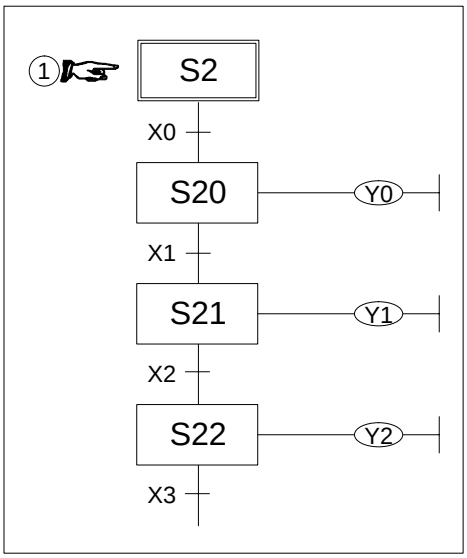
- For more information about device availability for individual PLC's, see chapter 8.

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.

4.4.3 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 3.

IST instruction:

- For more information on the IST instruction please see page 5-67.

4.4.4 Annunciator Flags

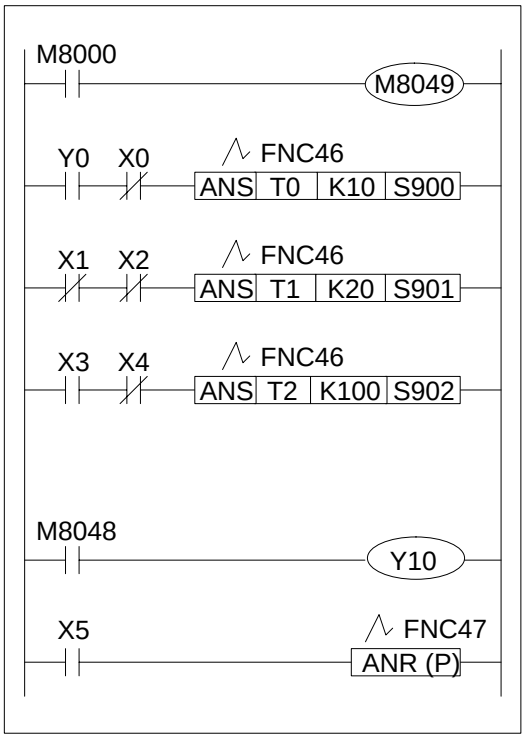
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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 page 5-47
ANR function 47: ANnunciator Reset - see page 5-47

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.



4.5 Pointers

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Device Mnemonic: P

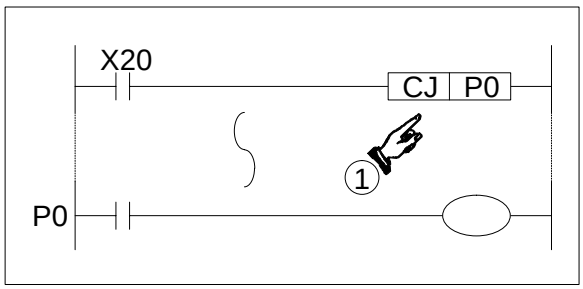
Purpose: Program flow control
Alias: Pointer
Program pointer
P

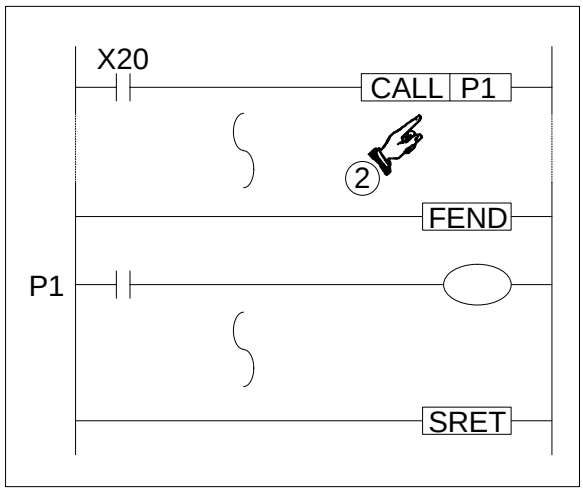
Available forms: Label: appears on the left of the left hand bus bar when the program is viewed in ladder mode.

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 page 5-5 and item ① on the example device usage diagram.
Can be used with call statements (CALL function 01 -not available on FX0 and FX0N PLC's)
- see page 5-7 and item ② on the example device usage diagram

Example device usage:





Available devices:

- FX0(S), FX0N and FX PLC's have 64 pointers; available from the range of P0 to P63.
- FX(2C) and FX2N(C) PLC's have 128 pointers; available from the range of P0 to P127.

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. Labelling the END instruction with P63 is not required.

Device availability:

- For more information about device availability for individual PLC's, please see chapter 8.

4.6 Interrupt Pointers

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Device Mnemonic: I

Purpose: Interrupt program marker

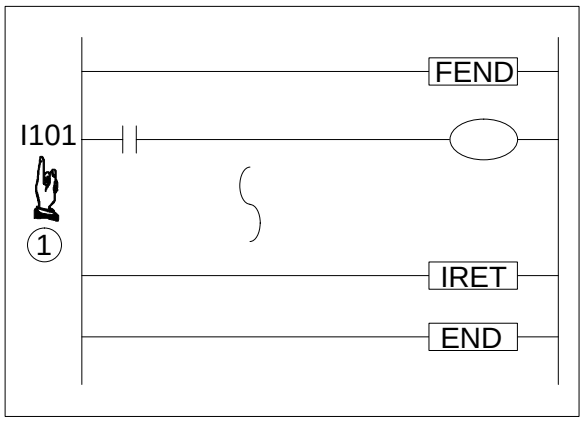
Alias: Interrupt
High speed interrupt
I

Available forms: Label: appears on the left of the left hand bus bar when the program is viewed in ladder mode
(see ¿ in the example device usage diagram).

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

Further uses: Input interrupts - see page 4-12
Timer interrupts - see page 4-12
Disabling interrupts - see page 4-13
Counter interrupts - see page 4-13

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 page 5-9
  - EI function 04: enable interrupt - see page 5-9
  - DI function 05: disable interrupt - see page 5-9

Nested levels:

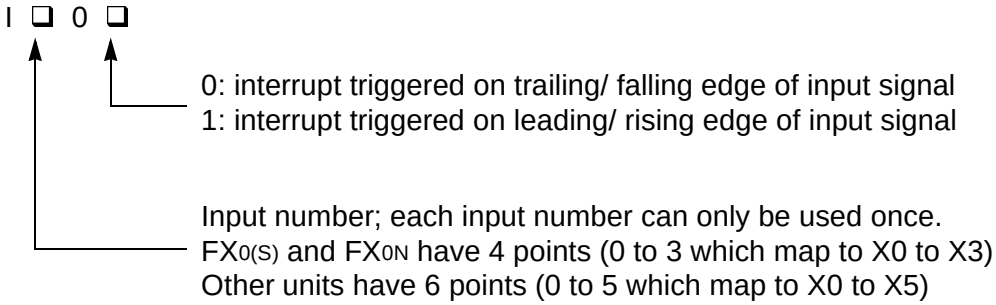
- While an interrupt is processing all other interrupts are disabled. To achieve nested interrupts the EI-DI instruction must be programmed within an interrupt routine.
  Interrupts can be nested for two levels.

Pointer position:

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

4.6.1 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.

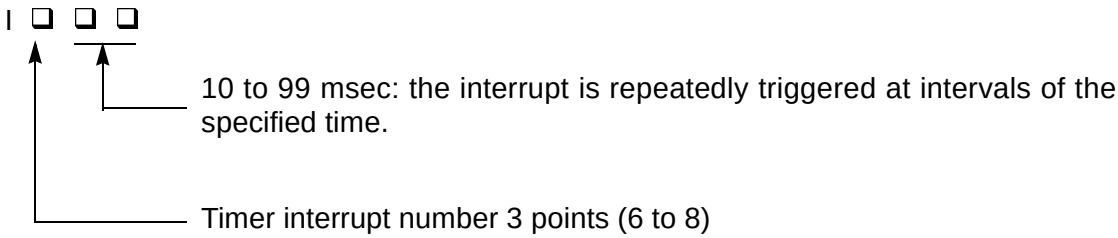
Rules of use:

- 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.

4.6.2 Timer Interrupts

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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.

4.6.3 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 6. 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.

4.6.4 Counter Interrupts

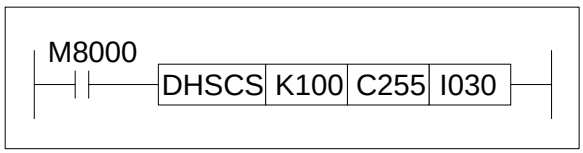
$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$

Identification of interrupt pointer number:

I 0 ❑ 0

Counter interrupt number 6 points (1 to 6). Counter interrupts can be entered as the output devices for High Speed Counter Set (HSCS, FNC 53). To disable the Counter Interrupts Special Auxiliary Relay M8059 must be set ON.

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 page 5-55

4.7 Constant K

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

4.8 Constant H

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

4.9 Timers

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Device Mnemonic: T

Purpose: Timed durations

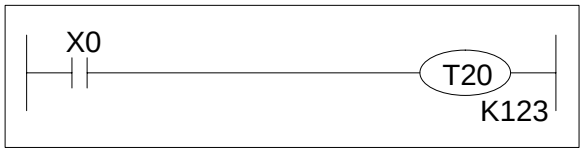
Alias: Timer(s)
T

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 page 4-16
Retentive timers - see page 4-17
Timers used in interrupt and 'CALL' subroutines - see page 4-18

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

Further uses: None

Example device usage:



Available devices:

Timer Resolution	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)
100 msec	56 (T0 - 55)	63 (T0 - 62)	200 (T0 - 199)		
10 msec	\ 24 (T32 - 55)	\ 31 (T32 - 62)	46 (T200 - 245)		
1 msec	N/A	1 (T63)	N/A		
Retentive 1 msec	N/A	N/A	4 (T246 - 249)		
Retentive 100 msec	N/A	N/A	6 (T250 - 255)		

\ Selectable timers taken from the main range of 100 msec timers, see page 4-16.

Timer accuracy:

- See page 4-18.

4.9.1 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× 10 msec
567× 0.01 sec = 5.67 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.

4.9.2 Selectable Timers

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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.

- FX0, FX0S driving M8028 ON, timers T32 to 55 (24 points) are changed to 10 msec resolution.
- FX0N driving M8028 ON, timers T32 to 62 (31 points) are changed to 10 msec resolution.

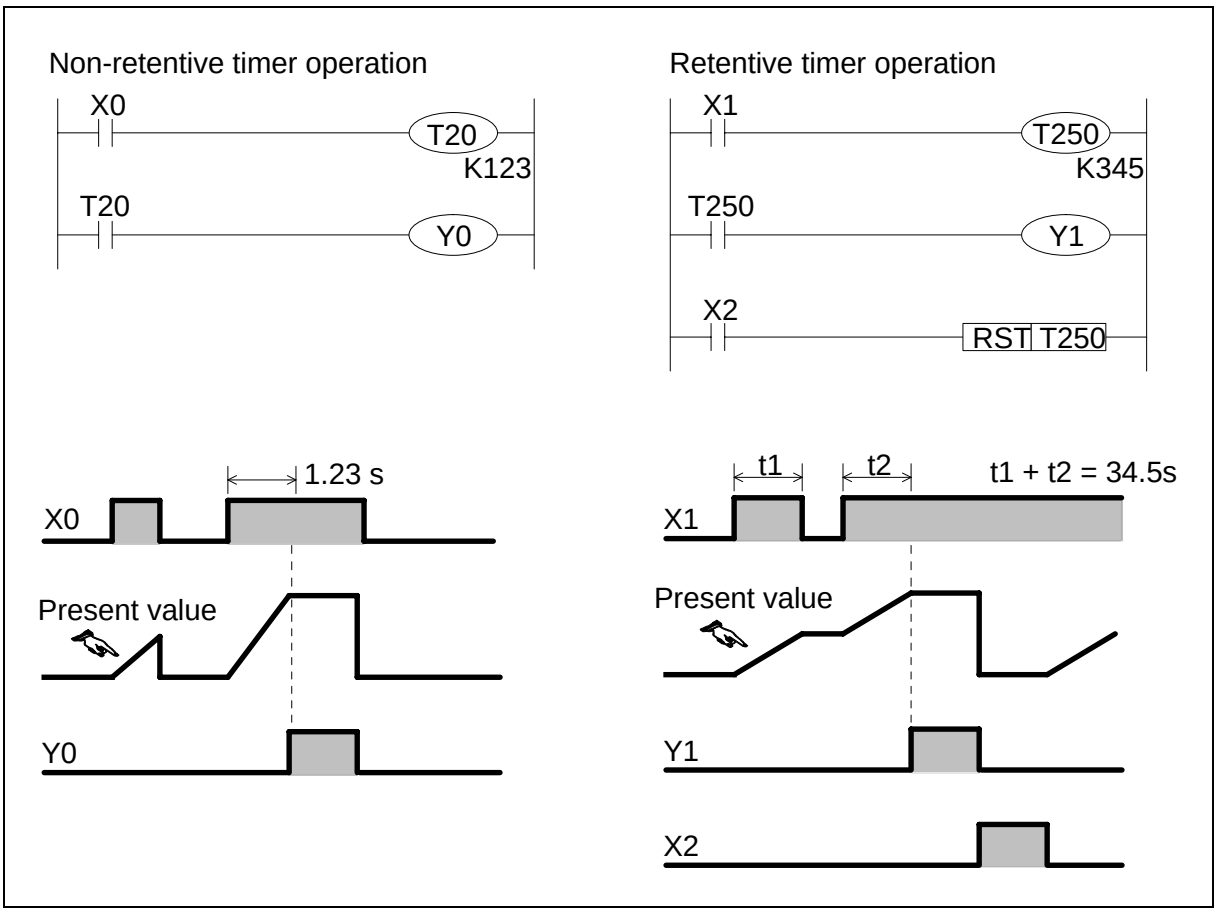
Driving special auxiliary coils:

- Please check the definition of special auxiliary coils before using them.
  Not all PLC's associate the same action to the same device.

4.9.3 Retentive Timers

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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.



Using timers in interrupt or 'CALL' subroutines:

- Please see page 4-18.

Available devices:

- Please see the information table on page 4-15.

4.9.4 Timers Used in Interrupt and 'CALL' Subroutines

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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 timers 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.

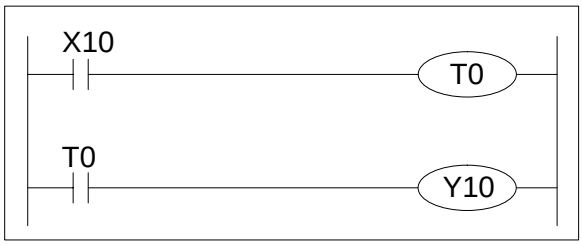
4.9.5 Timer Accuracy

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 × The program scan time

Condition 1:

The timer contact appears after the timer coil.



Maximum timing error:

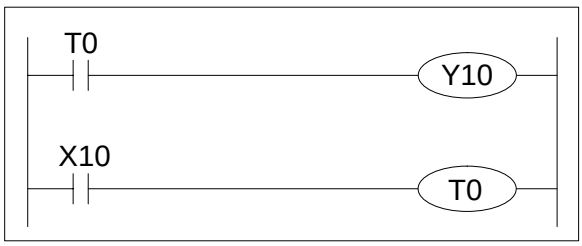
2 × Scan time + The input filter time

Minimum timing error:

Input filter time - The timer resolution

Condition 2:

The timer contact appears before the timer coil.



Maximum timing error:

3 × Scan time + The input filter time

Minimum timing error:

Input filter time- The timer resolution

Internal timer accuracy:

- The actual accuracy of the timing elements within the PLC hardware is; ± 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 pulses.

4.10 Counters

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Device Mnemonic: C

Purpose: Event driven delays

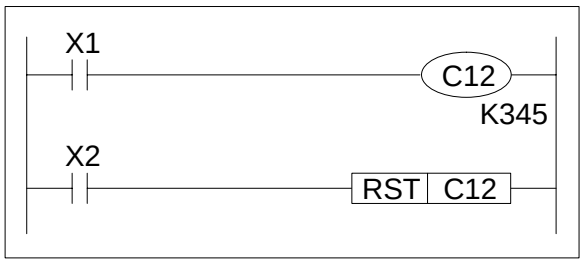
Alias: Counter(s)
C

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 page 4-20
General/latched 32bit bi-directional counters - see page 4-21
(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:

Counter Resolution	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)
General 16bit up counter	14 (C0 - 13)	16 (C0 - 15)	100 (C0 - 99)		
Latched 16bit up counter	2 (C14 - 15)	16 (C16 - 31)	100 (C100 - 199)		
General 32bit bi-directional counter	N/A	N/A	20 (C200 - 219)		
Latched 32bit bi-directional counter	N/A	N/A	15 (C220 - 234)		

High speed counters:

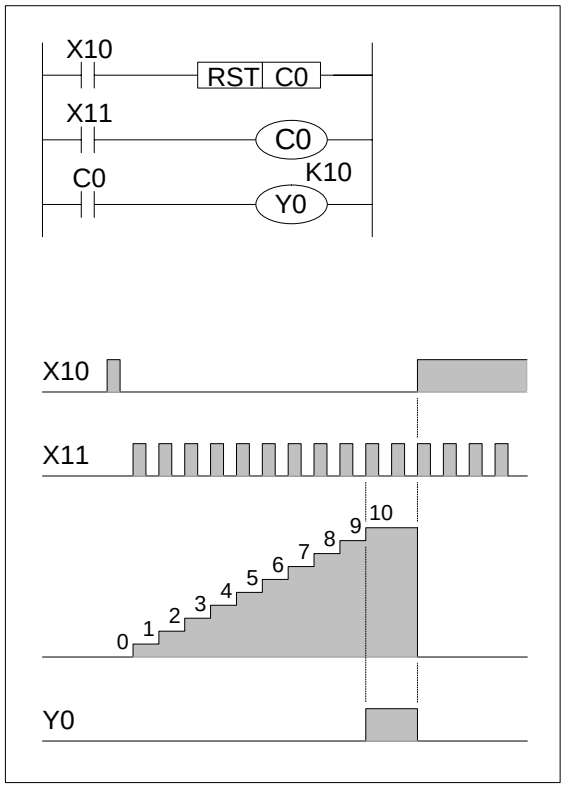
- For high speed counters please see page 4-22.

Setting ranges for counters:

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

4.10.1 General/ Latched 16bit UP Counters

The current value of the counter increases each time coil C0 is turned ON by X11. 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 X11 is turned ON. The counter current value is reset to '0' (zero) when the RST instruction is executed by turning ON X10 in the example. The output contact Y0 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.

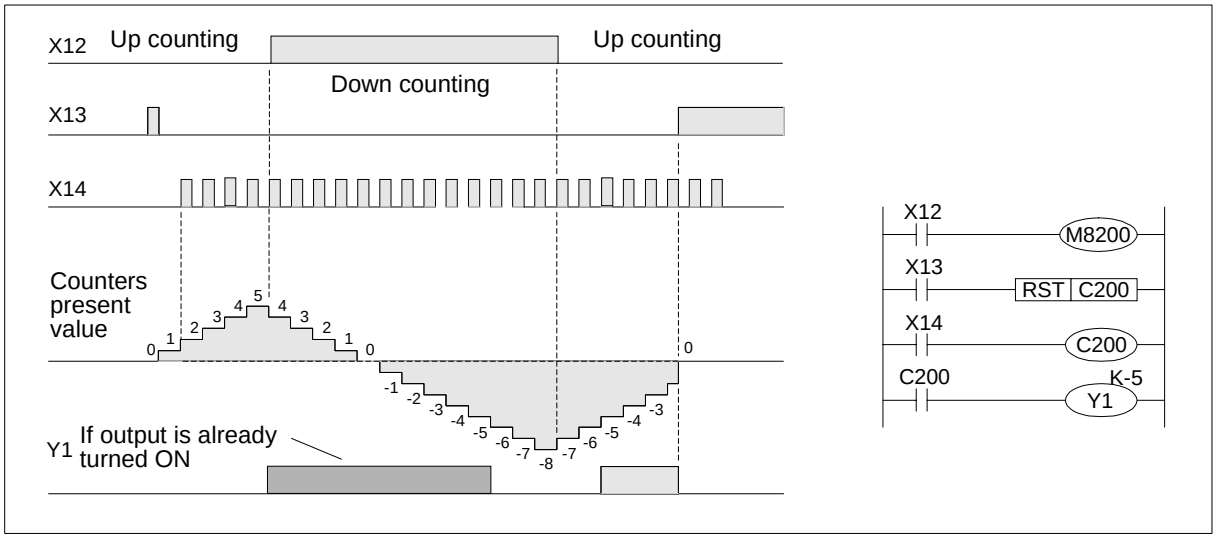
Available devices:

- Please see the information table on page 4-19.

4.10.2 General/ Latched 32bit Bi-directional Counters

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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 X14 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 rest to "0" (zero) by forcibly resetting the counter coil; in the example program by switching the input X13 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.

Available devices:

- Please see the information table on page 4-19.

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.

4.11 High Speed Counters

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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. Please check the following sections for device availability;

FX0,FX0S and FX0N - see page 4-24
FX, FX2C, FX2N(C) - see page 4-25

The following sections refer to counter types;
1 phase counters (user start and reset) - see page 4-29
1 phase counters (assigned start and reset) - see page 4-30
2 phase bi-directional counters - see page 4-31
A/B phase counters - see page 4-32

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

Further uses: None

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

Basic high speed counter operation:

- For information on basic high speed counters please see page 4-23.

4.11.1 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;

a) 1 phase with user start/reset: C235 to C240

b) 1 phase with assigned start/reset: C241 to C245

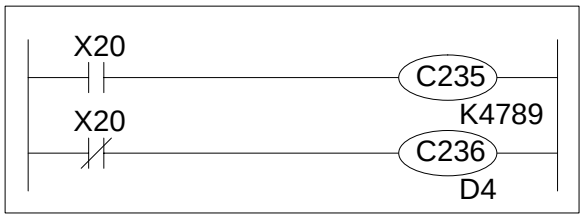
c) 2 phase bi-directional: C246 to C250

d) A/B phase type: C251 to C255

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:

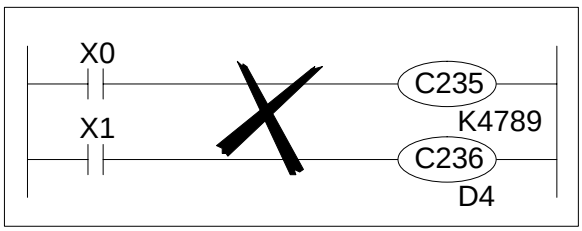
When X20 is ON, high speed counter C235 is selected. The counter C235 corresponds to count input X0. X20 is NOT the counted signal. This is the continuous drive mentioned earlier. X0 does not have to be included in the program. The input assignment is hardware related and cannot be changed by the user.



When X20 is OFF, coil C235 is turned OFF and coil C236 is turned ON. Counter C236 has an assigned input of X1, again the input X20 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.



4.11.2 Availability of High Speed Counters on FX0, FX0S and FX0N PLC's

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The following device table outlines the range of available high speed counters on both the FX0, FX0S and the FX0N;

INPUT	1 Phase counter user start/reset				1 Phase counter assigned start/reset			2 Phase counter bi-directional			A/B Phase counter		
	C235	C236	C237	C238	C241	C242	C244	C246	C247	C249	C251	C252	C254
X0	U/D				U/D		U/D	U	U	U	A	A	A
X1		U/D			R		R	D	D	D	B	B	B
X2			U/D			U/D			R	R		R	R
X3				U/D		R	S			S			S

Key:
U - up counter input **D** - down counter input
R - reset counter (input) **S** - start counter (input)
A - A phase counter input **B** - B phase counter input

C235 - Counter is backed up /latched on both FX0, FX0S and FX0N

C236 - Counter is backed up /latched on FX0N only (FX0, FX0S has no backup/latch on this device)

Input assignment:

- Different types of counters can be used at the same time but their inputs must not coincide. Inputs X0 to X3 cannot be used for more than one counter. For example, if C251 is used the following counters and instructions cannot be used; C235, C236, C241, C244, C247, C249, C252, C254, I0❑❑, I1❑❑.

Counter speeds and operational rules:

Unit type	Max. 1 phase counting speed	Sum of the speeds of the active 1 phase counters	Max. 2 phase counting speed	Max. number of 2 phase counters	Max. combined sum of 1 and 2 phase counting speeds
FX0, FX0N	5kHz	≤ 5kHz	2 kHz	1	1 phase and 2 phase counters cannot be mixed
FX0S	7kHz	≤ 14kHz	2 kHz	1	≤ 14kHz see note below

- All inputs identified are 5 kHz inputs.
- Only one 2 phase or A/B phase counter should be operated at any one time.
- A high speed counter specified in an applied instruction may not be modified by V or Z indexes.

Calculating the maximum combined counting speed on FX0S:

This is calculated as follows:*(2 phase counter speed x number of counted edges)* + *(the sum of the speeds of the active 1 phase counters).*

4.11.3 Availability of High Speed Counters on FX, FX2C PLC's

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The following device table outlines the range of available high speed counters on both the FX, FX2C.

INPUT	1 Phase counter user start/reset						1 Phase counter assigned start/reset					2 Phase counter bi-directional					A/B Phase counter				
	C235	C236	C237	C238	C239	C240	C241	C242	C243	C244	C245	C246	C247	C248	C249	C250	C251	C252	C253	C254	C255
X0 ◆	U/D						U/D			U/D		U	U		U		A	A		A	
X1 p		U/D					R			R		D	D		D		B	B		B	
X2 ◆			U/D					U/D			U/D		R		R			R		R	
X3 ◆				U/D				R		S	R			U	S	U			A		A
X4 p					U/D				U/D					D		D			B		B
X5 p						U/D			R					R		R			R		R
X6										S					S					S	
X7											S					S					S

Key:
U - up counter input **D** - down counter input
R - reset counter (input) **S** - start counter (input)
A - A phase counter input **B** - B phase counter input
C235 - Counter is backed up / latched

Input assignment:

- X6 and X7 are also high speed inputs, but function only as start signals. They cannot be used as the counted inputs for high speed counters.
- Different types of counters can be used at the same time but their inputs must not coincide. For example, if counter C247 is used, then the following counters and instructions cannot be used;
  C235, C236, C237, C241, C242, C244, C245, C246, C249, C251, C252, C254, I0❑❑, I1❑❑, I2❑❑.
- The inputs marked ❑ are 7 kHz inputs, while those marked ◆ are 10 kHz inputs.

Counter speeds:

- The maximum counting speed is dependent on the type, quantity of counters and on how many high speed counter instructions are being used. The following tables give the approximate maximum counting speed for each identified case.
- Please take care when using the speed instruction (SPD, FNC 56). This instruction is treated as if it was a single phase counter. This must be accounted for when the sum counting speeds are calculated.

1 Phase Counters

Counter input	Number of counters	Frequency in kHz		
		No execution of high speed instructions	Execution of (D)HSCS/R (1 to 6 instructions)	Execution of (D)HSZ (1 to 2 instructions)
X0, X2, X3 (10 kHz inputs)	1	10	7	5
	2	10	4	2.5
	3	6.6	2.5	
X0 to X5 (When X0, X2 and X3 are not used exclu-sively)	1	7	5	4
	2	3.5	2.5	1.5
	3	2.5	2	
	4		1.5	
	5			
	6			

A/B Phase Counters

Counter input	Number of counters	Frequency in kHz		
		No execution of high speed instructions	Execution of (D)HSCS/R (1 to 6 instructions)	Execution of (D)HSZ (1 to 2 instructions)
C251 - C255	1	2	2	2
	2		1.5	1.5

A/B Phase Counters Used with Either a 1 or 2 Phase Counter

The frequency of the A/B phase counter must be kept below 1 kHz. The maximum frequency of the 1, 2 phase counter is listed in the following table:

Counter input	Number of counters	Frequency in kHz		
		No execution of high speed instructions	Execution of (D)HSCS/R (1 to 6 instructions)	Execution of (D)HSZ (1 to 2 instructions)
With 1 A/B phase counter at 1 kHz	1	5	4	3
	2	4	2	1
	3	3		
	4	2	1	

Note: Bi-directional counters are designed such that the up count signal and the down count signal never operate at the same time. Therefore it is really using only one phase at one time. Thus, they can be treated in the same way as the 1 phase counters when calculating the combined frequency.

Combined frequencies:

- The combined frequency is the sum of the maximum frequencies of all the signals simultaneously appearing at the inputs of the PLC. The criteria is that in order for the high speed counters to count correctly they must have a combined frequency of less than 20 kHz.

Example:

1 Phase counters	Corresponding input	Maximum signal frequency
C235	X0	4.2 kHz
C237	X2	4 kHz
C240	X3	6 kHz
Combined frequency		14.2kHz

The combined frequency of 14.2 kHz is lower than the maximum of 20 kHz, so this example is valid.

A/B Phase counters:

When calculating a combined frequency which includes an A/B phase counter, the maximum counting frequency should be multiplied by a factor of 4 before adding the maximum frequencies of the combining counters.

Example:

1 Phase counters	Corresponding input	Maximum signal frequency
1- Phase C237	X2	3 kHz
Bi-directional C246	X0, X1	4 kHz
A/B Phase C255	X3, X4	1 kHz × 4
Combined frequency		3 + 4 + (1 × 4) = 11kHz

The combined frequency of 11 kHz is lower than the maximum of 20 kHz, so this example is valid.

2 Phase counters:

- When pulses arrive at the up and down count inputs at the same time, treat this as 2 single phase counters when calculating the combined frequency.

ClockWise - Counter-ClockWise format encoders:

- When encoders with CW and CCW format inputs are used, the bi-directional counters can count at a much higher frequency than the A/B phase counters, there is also no loss in resolution.

4.11.4 Availability of High Speed Counters on FX2N(C) PLC's

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

The following device table outlines the range of available high speed counters on FX2N(C).

INPUT	1 Phase counter user start/reset						1 Phase counter assigned start/reset					2 Phase counter bi-directional					A/B Phase counter				
	C235	C236	C237	C238	C239	C240	C241	C242	C243	C244	C245	C246	C247	C248	C249	C250	C251	C252	C253	C254	C255
X0	U/D						U/D			U/D		U	U		U		A	A		A	
X1		U/D					R			R		D	D		D		B	B		B	
X2			U/D					U/D			U/D		R		R			R		R	
X3				U/D				R		S	R			U		U			A		A
X4					U/D				U/D					D		D			B		B
X5						U/D			R					R		R			R		R
X6										S					S					S	
X7											S					S					S

Key:
U - up counter input **D** - down counter input
R - reset counter (input) **S** - start counter (input)
A - A phase counter input **B** - B phase counter input
C235 - Counter is backed up/latched

Input assignment:

- X6 and X7 are also high speed inputs, but function only as start signals. They cannot be used as the counted inputs for high speed counters.
- Different types of counters can be used at the same time but their inputs must not coincide. For example, if counter C247 is used, then the following counters and instructions cannot be used;
  C235, C236, C237, C241, C242, C244, C245, C246, C249, C251, C252, C254, I0❑❑, I1❑❑, I2❑❑.

Counter Speeds:

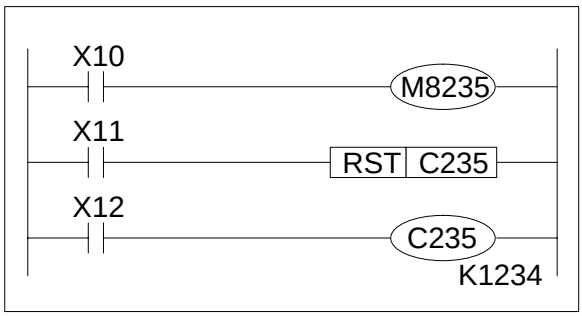
- General counting frequencies:
  - Single phase and bi-directional counters; up to 10 kHz.
  - A/B phase counters; up to 5 kHz.
  - Maximum total counting frequency; 20 kHz (A/B phase counter count twice).
- Inputs X0 and X1 are equipped with special hardware that allows very high speed counting as follows:
  - Single phase or bi-directional counting with C235, C236 or C246; up to 60 kHz.
  - Two phase counting with C251; up to 30 kHz.

If any high speed comparison instructions (FNC's 53, 54, 55) are used, X0 and X1 must resort to software counting. In this case, please see the table below:

Function Number	Max. Combined Signal Frequency
53 or 54	11 kHz
55	5.5 kHz

4.11.5 1 Phase Counters - User Start and Reset (C235 - C240)

These counters only use one input each. When direction flag M8235 is ON, counter C235 counts down. When it is OFF, C235 counts up.
When X11 is ON, C235 resets to 0 (zero). All contacts of the counter C235 are also reset.
When X12 is ON, C235 is selected. From the previous counter tables, the corresponding counted input for C235 is X0. C235 therefore counts the number of times X0 switches from OFF to ON.



Device specification:

- All of these counters are 32bit up/down ring counters. Their counting and contact operations are the same as normal 32bit up/down counters described on page 4-21. When the counters current value reaches its maximum or setting value, the counters associated contacts are set and held when the counter is counting upwards. However, when the counter is counting downwards the contacts are reset.

Setting range:

- -2,147,483,648 to +2,147,483,647

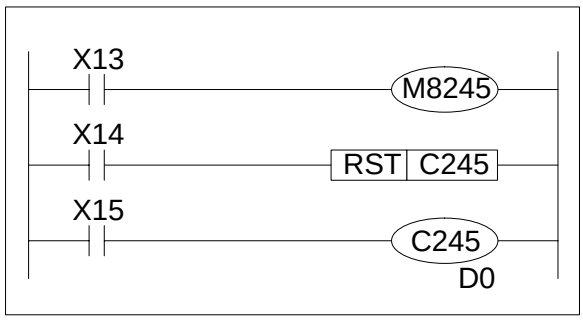
Direction setting:

- The counting direction for 1 phase counters is dependent on their corresponding flag M8☆☆☆; where ☆☆☆ is the number of the corresponding counter, (C235 to C240).
  When M8☆☆☆ is ON the counter counts down,
  When M8☆☆☆ is OFF the counter counts up.

Using the SPD instruction:

- Care should be taken when using the SPD applied instruction (FNC 56). This instruction has both high speed counter and interrupt characteristics, therefore input devices X0 through X5 may be used as the source device for the SPD instruction. In common with all high speed processes the selected source device of the SPD instruction must not coincide with any other high speed function which is operating, i.e. high speed counters or interrupts using the same input.
  When the SPD instruction is used it is considered by the system to be a 1 phase high speed counter. This should be taken into account when summing the maximum combined input signal frequencies - see the previous section.

4.11.6 1 Phase Counters - Assigned Start and Reset (C241 to C245)



These counters have one countable input and 1 reset input each. Counters C244 and C245 also have a start input.
When the direction flag M8245 is ON, C245 counts down. When it is OFF C245 will count up.
When X14 is ON, C245 resets in the same manner as normal internal 32bit counters, but C245 can also be reset by input X3. This is assigned automatically when counter C245 is used (see previous counter tables).
Counter C245 also has an external start contact, again automatically assigned. This is actually input X7. Once again this data can be found on the previous counter tables.
When X7 is ON, C245 starts counting, conversely when X7 is OFF C245 stops counting. The input X15 selects and reserves the assigned inputs for the selected counter, i.e. in this case C245.
The reason why these counters use assigned start (X7) and reset (X3) inputs is because they are not affected by the cycle (scan) time of the program. This means their operation is immediate and direct.
In this example C245 actual counts the number of OFF to ON events of input X2.
Note: Because C245 is a 32bit counter, its setting data, specified here by a data register also has to be of a 32bit format. This means that data registers D1 and D0 are used as a pair to provide the 32bit data format required.

Device specification:

- All of these counters are 32bit up/down ring counters. Their counting and contact operations are the same as normal 32bit up/down counters described on page 4-21. When the counters current value reaches its maximum or setting value, the counters associated contacts are set and held when the counter is counting upwards. However, when the counter is counting downwards the contacts are reset.

Setting range:

- -2,147,483,648 to +2,147,483,647

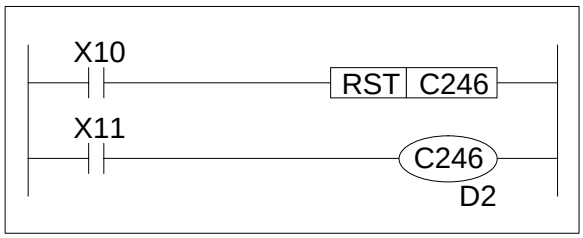
Direction setting:

- The counting direction for 1 phase counters is dependent on their corresponding flag M8☆☆☆; where ☆☆☆ is the number of the corresponding counter, (C241 to C245).
  - When M8☆☆☆ is ON the counter counts down.
  - When M8☆☆☆ is OFF the counter counts up.

4.11.7 2 Phase Bi-directional Counters (C246 to C250)

These counters have one input for counting up and one input for counting down. Certain counters also have reset and start inputs as well.

When X10 is ON, C246 resets in the same way as standard 32bit counters.



Counter C246 uses inputs;
X0 to count up and
X1 to count down

For any counting to take place the drive input X11 must be ON to set and reserve the assigned inputs for the attached counter, i.e. C246.

Note:

X0 moving from OFF to ON will increment C246 by one
X1 moving from ON to OFF will decrement C246 by one

Bi-directional counter C250 can be seen to have X5 as its reset input and X7 as its start input. Therefore, a reset operation can be made externally without the need for the RST C250 instruction.

X13 must be ON to select C250. But start input X7 must be ON to allow C250 to actually count. If X7 goes OFF counting ceases.
Counter C250 uses input X3 to count up and input X4 to count down.

Device size:

- All of these counters have 32bit operation.

Setting range:

- -2,147,483,648 to +2,147,483,647

Direction setting:

- The counting direction for 1 phase counters is dependent on their corresponding flag M8☆☆☆; where ☆☆☆ is the number of the corresponding counter, (C241 to C245).
  - When M8☆☆☆ is ON the counter counts down,
  - When M8☆☆☆ is OFF the counter counts up.

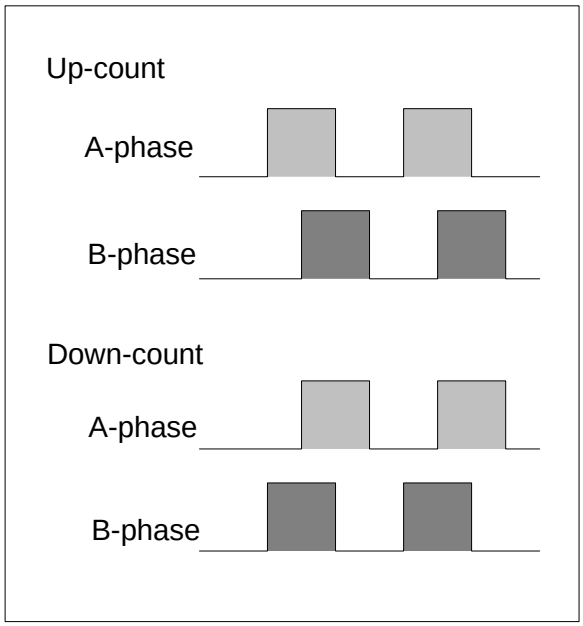
4.11.8 A/B Phase Counters (C252 to C255)

With these counters only the input identified in the previous high speed counter tables can be used for counting. The counting performed by these devices is independent of the program cycle (scan) time. Depending on the counter used, start, reset and other associated inputs are automatically allocated.

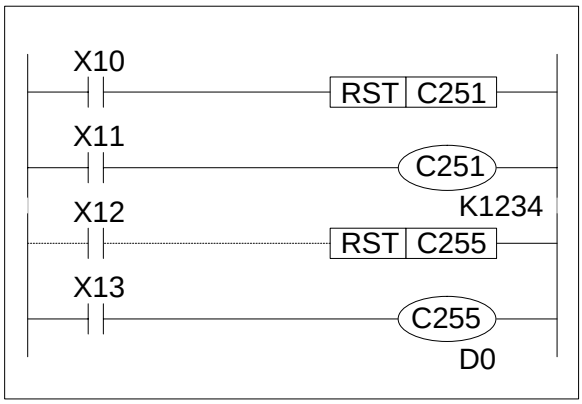
The A phase, B phase input signal not only provide the counted signals but their relationship to each other will also dictate the counted direction.

While the wave form of the A phase is in the ON state and...
the B phase moves from OFF to ON the counter will be counting up.

However, if...
the B phase moves from ON to OFF the counter will be in a down configuration.
One count is registered after both A and B phase inputs have been given and released in the correct order.



C251 counts the ON/OFF events of input X0 (the A phase input) and input X1 (the B phase input) while X11 is ON.
C255 starts counting immediately when X7 is turned ON while X13 is ON. The counting inputs are X3 (A phase) and X4 (B phase).
C255 is reset when X5 is turned ON. It can also be reset with X12 in the sequence.



Device specification:

- A maximum of 2 points - 2 phase, 32bit, up/down counters can be used. The operation of the output contact in relation to the counted data is the same as standard 32bit counters described in section 4.11.

Setting range:

- -2,147,483,648 to +2,147,483,647

Direction setting:

- Check the corresponding special relay M8☆☆☆ to determine if the counter is counting up or down.

4.12 Data Registers

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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 - see page 4-34
Battery backed/latched registers - see page 4-35
Special diagnostic registers - see page 4-35
File registers - see page 4-36
RAM file registers - see page 4-36
Externally adjusted registers - see page 4-37

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:

	FX0	FX0N	FX (CPU Ver. 2.3)	FX(2C)	FX2N(C)
General use registers	30 (D0 - 29)	128 (D0 - 127)	200 (D0 - 199)		
Latched registers	2 (D30 - 31)	128 (D128 - 256)	312 (D200 - 512)	800 (D200 - 999)	7800 (D200 - 7999)
Diagnostic registers	27 (D8000 - 8069)	39 (D8000 - 8255)	256 (D8000 - 8255)		
File registers R	N/A	1500 (D1000 - 2499)	2000 (D1000 - 2999)		7000 (D1000 - 7999)
RAM file registers M	N/A			2000 (D6000 - 7999)	N/A
Adjustable registers F	1 (D8013)	2 (D8030 - 8031)	N/A		

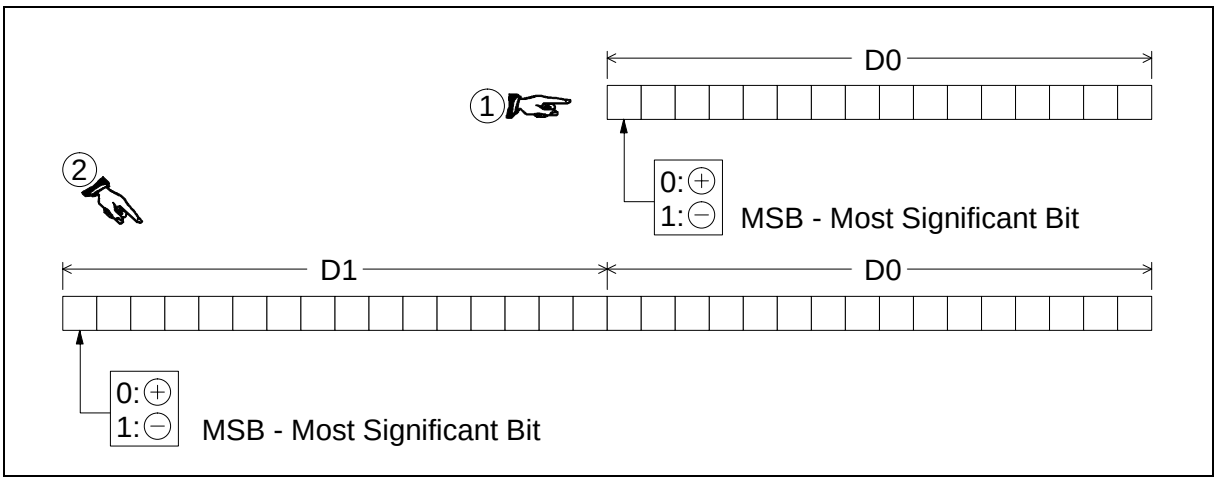
R - These devices are allocated by the user at the expense of available program steps. On FX2N(C) these devices are a subset of the latched registers.

F - These devices are also included under the count for diagnostic registers.

M - These devices are activated when special auxiliary relay M8074 is turned ON. No program steps are occupied by RAM file registers.

4.12.1 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 as ¡, 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.

4.12.2 Battery Backed/ Latched Registers

Once data is written to a battery backed register, it remains unchanged until it is overwritten. When the PLC's status is changed from RUN to STOP, the data in these registers is retained. The range of devices that is battery backed can be changed by adjusting the parameters of the PLC. For details of how to do this please refer to the appropriate programming tools manual.

Using the FX2-40AW/AP:

- When using an FX with either the FX2-40AWor the FX2-40AP a proportion of the latched data registers are automatically assigned for communications use by the FX2-40AW/AP module.

  Communication between Master and Slave 100 points M800 to M899
  10 points D490 to D499

  Communication between Slave and Master 100 points M900 to M999
  10 points D500 to D509

4.12.3 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 6 for tables containing data and descriptions of the available devices for each PLC.

4.12.4 File Registers

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

File registers are available in two forms:

- Program memory registers - these occupy program steps in blocks of 500 and are available on FX0N, FX, FX2C and FX2N(C) products
- RAM registers - these occupy a special data area and are available on all FX2C units and FX units with CPU version 3.07 or greater.

Program memory registers

File registers can be secured in the program memory (RAM, EEPROM or EPROM) in units of 500 points. These registers can be accessed with a peripheral device. While the PLC is operating, data in the file registers can be read to the general-use/ battery backed/ latched registers by using the BMOV instruction.
File registers are actually setup in the parameter area of the PLC. For every block of 500 file registers allocated and equivalent block of 500 program steps are lost.

Note: The device range for file registers in the FX2N(C) overlaps with the latched data registers. The allocation of these devices as file registers ensures that the data is kept with the program.

Writing to file registers:

- FX0N and FX file register data can only be changed by a peripheral device such as a hand held programmer or a personal computer running the appropriate software. For details of how to carry out the changes please reference the relevant operation manual for guidance.
- FX(2C) and FX2N(C) file register data can also be changed by the program using the BMOV instruction.
- Only file registers in RAM or internal memory can be changed during RUN, but both RAM, internal and EEPROM memory cassette memories can be changed when the PLC is in STOP mode.

Special caution when using FX0N:

- No file registers can be modified during RUN.

Special caution when using FX:

- While the FX PLC is in RUN mode, alteration of the file registers D1000 to D1119 (120 points) is not allowed. Attempts to change these devices during RUN may result in a program error when the PLC is next switched into RUN.

RAM registers

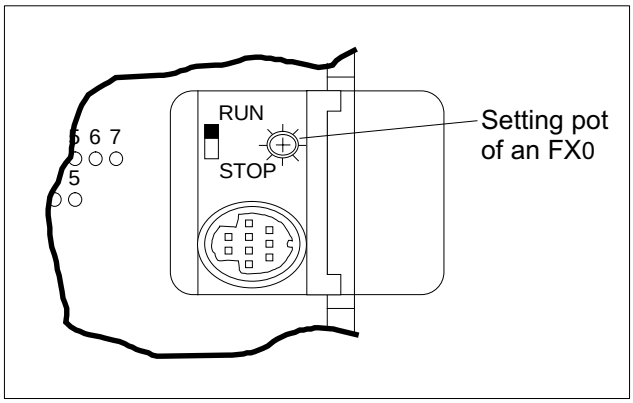
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

RAM file registers are 2000 points which can be activated by driving a special auxiliary relay M8074 ON. These registers can then be accessed just like normal program file registers. RAM registers occupy no program steps but do occupy the sampling trace data register area when M8074 is active.

Available devices:

- Please refer to the table on page 4-33 or chapters 6 and 8, where further details of the availability of devices can be found.

4.12.5 Externally Adjusted Registers



The FX0 and FX0N have 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 255. This is a built in feature and requires no additional setup or programming.
The FX, FX2C and FX2N(C) do not have this feature, however, an additional special function unit is available which provides the same function. For the FX and FX2C the unit is the FX-8AV. For FX2N(C) the unit is the FX2N-8AV-BD. To use this unit requires the applied instructions VRRD function 85 (Volume Read) and VRSC function 86 (Volume Scale).

	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)
Number of setting pots	1 point	2 points	8 points: Supplied by using the additional special function block FX-8AV or FX2N-8AV-BD		
Number of controlled data registers	1: D8013	1: D8013 or D8030, 2: D8031	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.

4.13 Index Registers

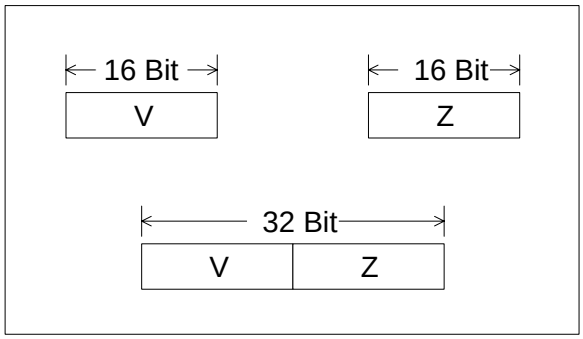
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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. FX0(S),FX0N, FX, FX2C there are two devices V and/ or Z.
For FX2N(C) there are 16 devices V0 - V7 and Z0 - Z7

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 Ì D13
Z = 14
D10Z
D10 + 14 = 24 Ì 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.

Special note for FX0 and FX0N users:

- Users of FX0 and FX0N PLC's should note that when high speed counters (C235 to C255) are used as operands in applied instructions, they may not be modified with V or Z index registers.

4.13.1 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:

If V = 3276 then K20V ⇨

K	20
V	(3276)
	3296

4.13.2 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:

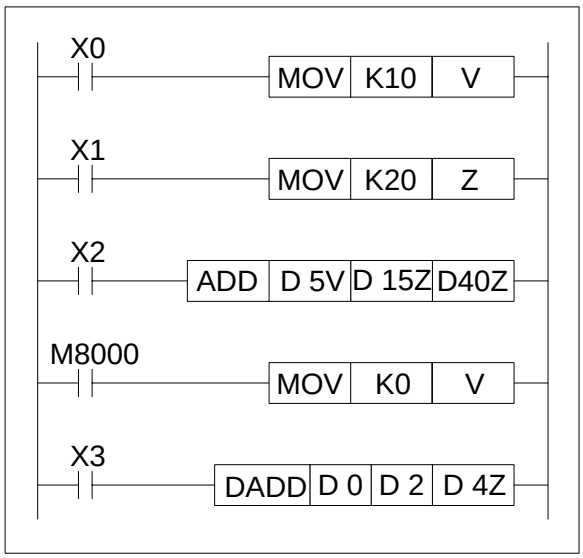
K4ZY30

are not acceptable.

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

4.13.3 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 contents 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.

4.14 Bits, Words, BCD and Hexadecimal

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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. Some of the covered topics are;

Available devices:

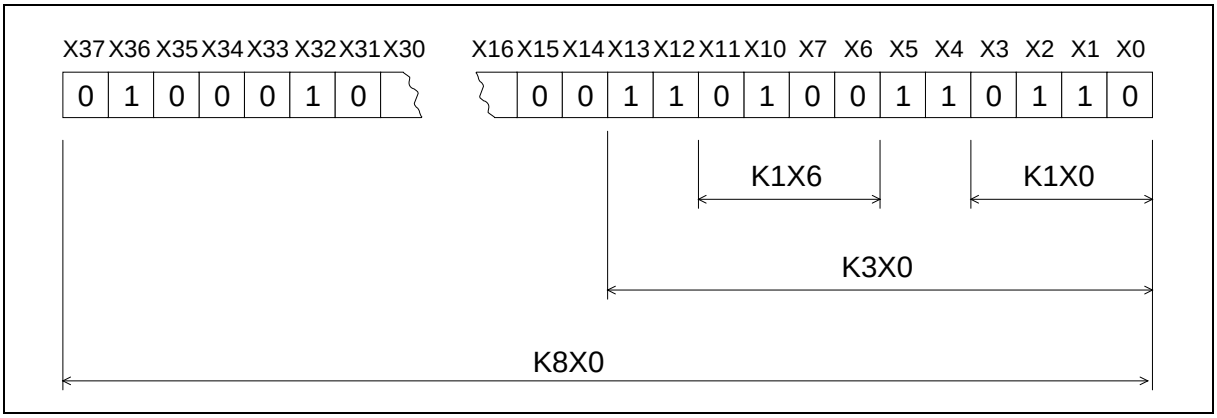
- For PLC specific available devices please see chapter 8.

4.14.1 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.
The diagram and example on the following page explain this idea further.....

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.

4.14.2 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.

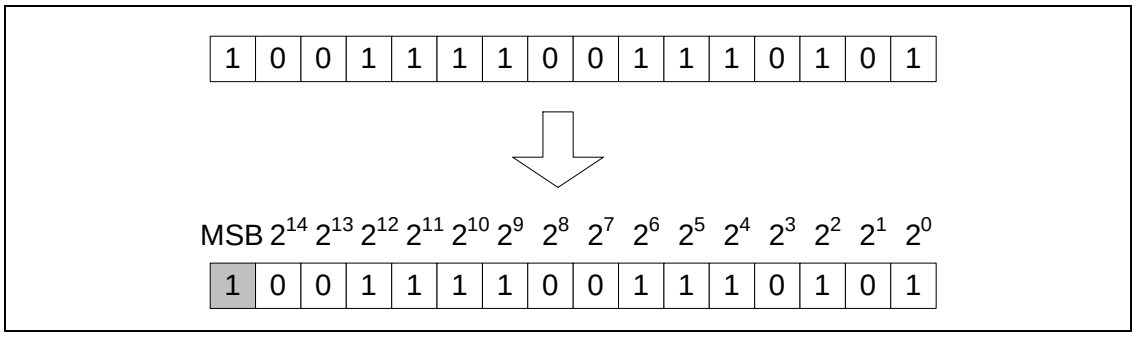
4.14.3 Interpreting Word Data

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.,



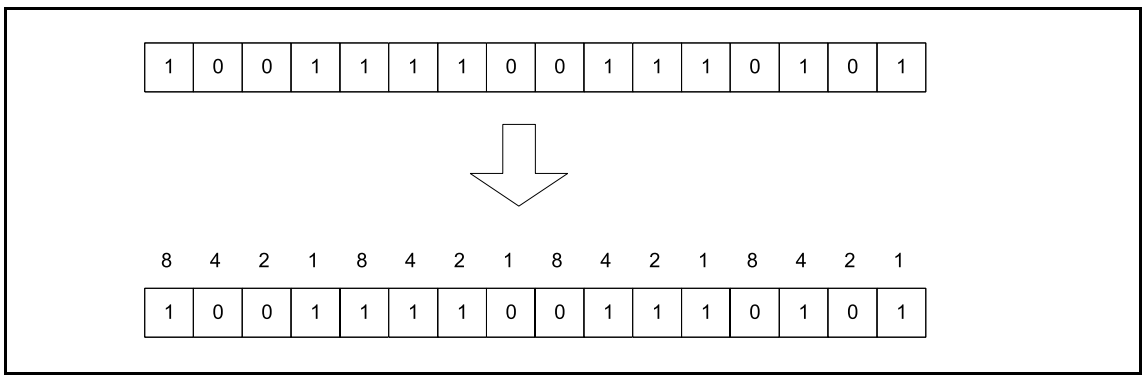
Decimal value = $(2^0 \times 1) + (2^2 \times 1) + (2^4 \times 1) + (2^5 \times 1)$
$+(2^5 \times 1) + (2^9 \times 1) + (2^{10} \times 1) + (211 \times 1) + (2^{12} \times 1)$

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:



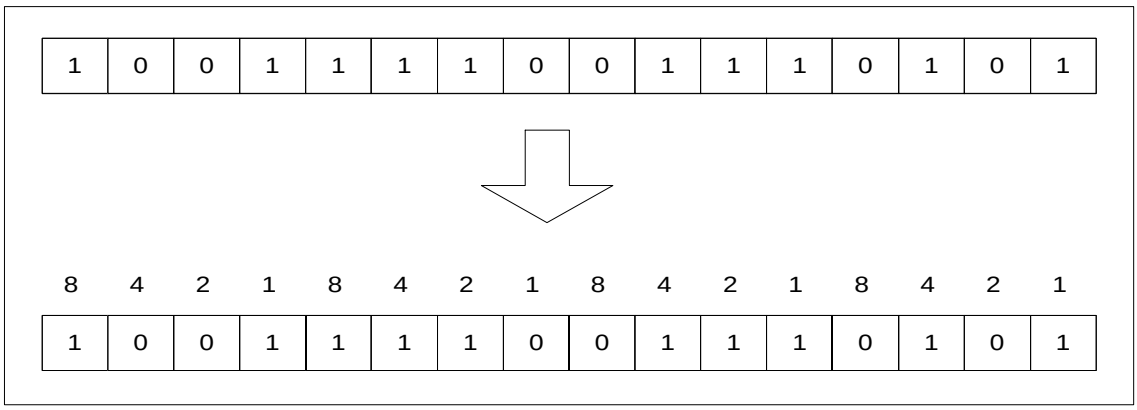
Hexadecimal value = ((1 x 8) + (1 x 1)), ((1 x 8) + (1 x 4) + (1 x 2)),
((1 x 4) + (1 x 2) + (1 x 1)), ((1 x 4) + (1 x 1))

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) ABCD 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.



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;

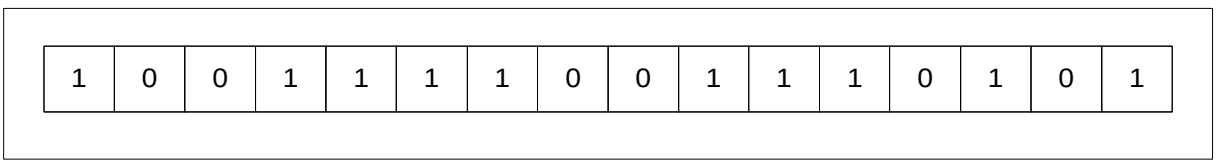
Extreme Left Hand Block= ((1 × 8) + (1 × 1)) = 9
Second Right Hand Block= ((1 × 4) + (1 × 2) + (1 × 1)) = 7
Extreme Right Hand Block= ((1 × 4) + (1 × 1)) = 5

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,



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!

4.14.4 Two's Compliment

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 1ís to 0ís and all 0ís to 1's.

7	=	00000111
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);

Inverted7	11111000
Additional1	00000001
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.

4.15 Floating Point And Scientific Notation

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

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

Fixed points:

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

4.15.1 Scientific Notation

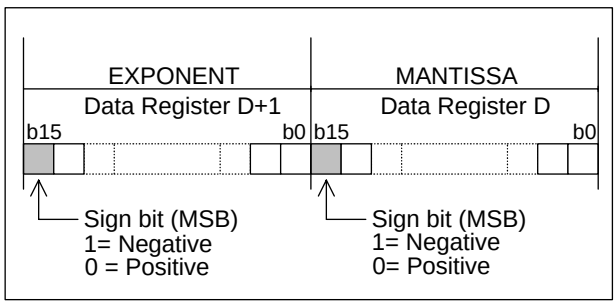
This format could be called the step between the 'integer' formats and the full floating point formats. In basic terms Scientific Notation use two devices to store information about a number or value. One device contains a data string of the actual characters in the number (called the mantissa), while 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
9999×10^{35}	9999×10^{-41}
-9999×10^{35}	-9999×10^{-41}

Scientific Notation can be obtained by using the BCD, or EBCD in FX2N, 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 - see page 5-22 for details. When using the FX2N the INT instruction (FNC 129) can be used.
Scientific Notation can be converted back to floating point format by using the BIN instruction (FNC 19) with the float flag M8023 set ON - see page 5-22 for details.

The following points should be remembered about the use of Scientific Notation within appropriate FX units;

- 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 2.34567×10^3 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.

4.15.2 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:

a) Normal FX operation using decimal (integers) numbers would equal 3 remainder 1

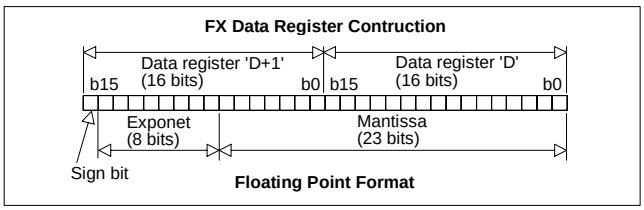
b) In floating point it would equal 3.14285 (approximately)

c) In Scientific format this calculation would be equal to 3142×10^{-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 fag M8023 set ON, floating point numbers can be converted back to decimal. see page 5-49 for more details.
The following points should be remembered about the use of Floating Point within appropriate FX units;

- 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.

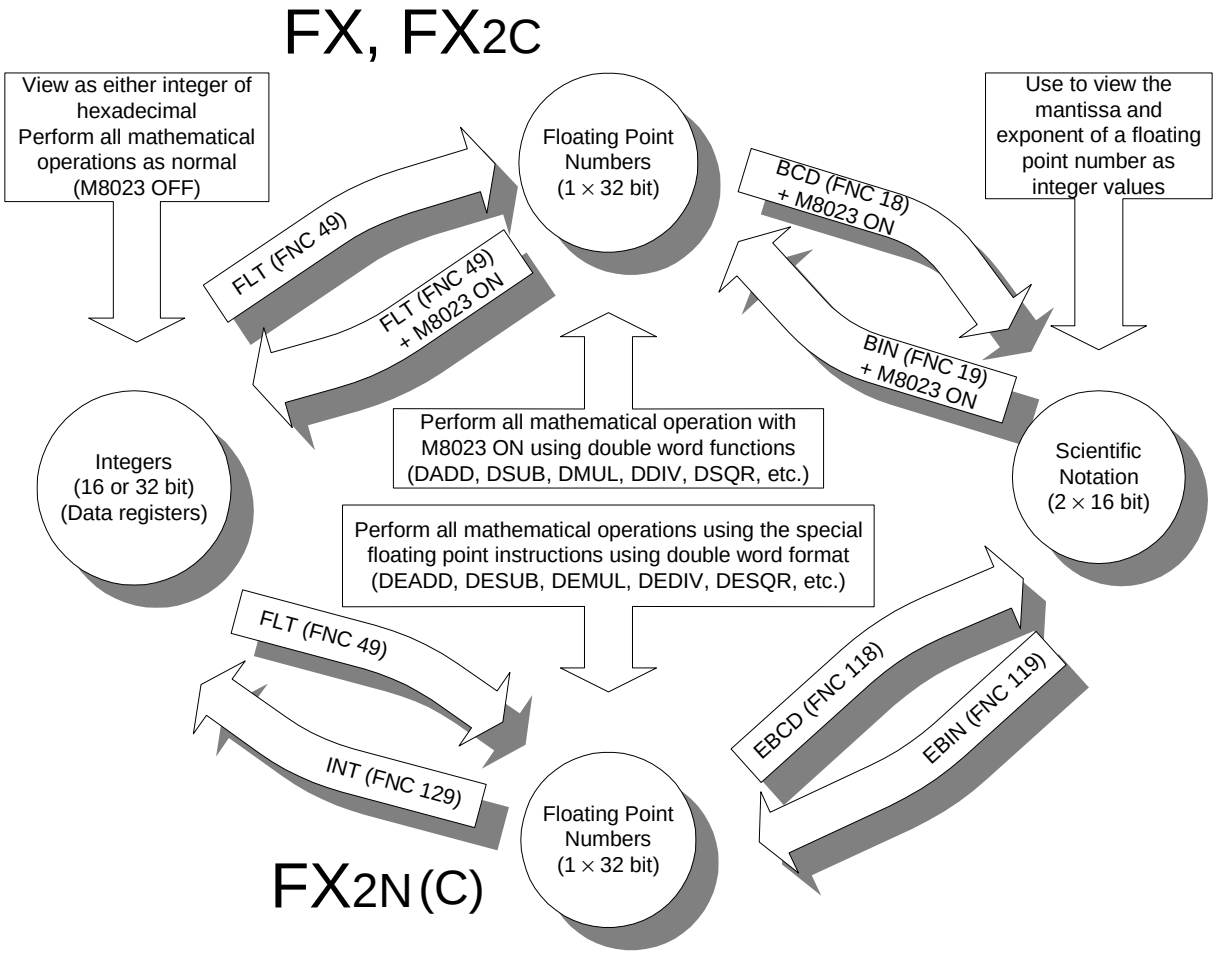


Valid ranges for floating point numbers as used in FX Main Processing Units:

Description	Sign	Exponent (bit pattern)	Mantissa (bit pattern)	Remark
Normal Float	0 or 1	11111110 00000001	11111111111111111111111 11111111111111111111110 00000000000000000000001 00000000000000000000000	Largest number +/- 3.403×10^{38} Accuracy: 7 significant figures Smallest number +/- 1.175×10^{-38}
Zero	0 or 1	00000000	00000000000000000000000	All digits are 0 (zero)

4.15.3 Summary Of The Scientific Notation and Floating Point Numbers

The instruction needed to convert between each number format are shown below in a diagrammatically format for quick and easy reference.



MEMO

Chapter Contents

5. Applied Instructions

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Applied Instructions are the 'specialist' instructions of the FX family of PLC's. 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:3steps 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. D- destination, S- source, n, m- number of elements. Additional numeric suffixes will be attached if there are more than one operand with the same function.

Not all instructions and conditions apply to all PLC's. Applicable CPU's are identified by the boxes in the top right hand corner of the page. For more detailed instruction variations a second indicator box is used to identify the availability of pulse, single (16 bit) word and double (32 bit) word format and to show any flags that are set by the instruction.

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Carry M8022
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		

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, indexable 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.

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.1 Program Flow-Functions00 to 09

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

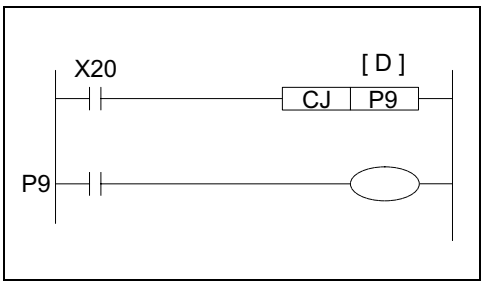
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.1.1 CJ (FNC 00)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands	Program steps
		D	
CJ FNC 00 (Conditional Jump)	Jumps to the identified pointer position	Valid pointers from the range 0 to 63	CJ, CJP:3steps Jump pointer PPP: 1 step

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

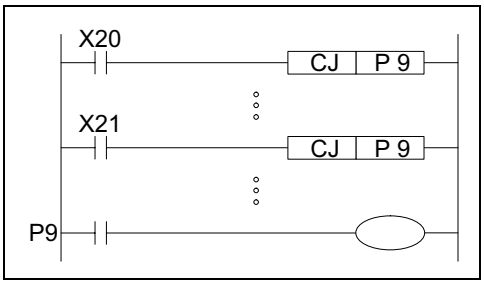
When the CJ instruction is active it forces the program to jump to an identified program marker. While the jump takes place the intervening pro-gram steps are skipped. This means they are not processed in any way. The resulting effect is to speed up the programs operational scan time.

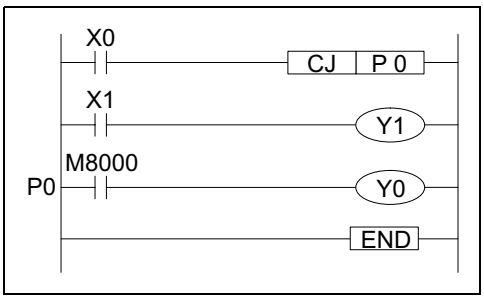
Points to note:

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.

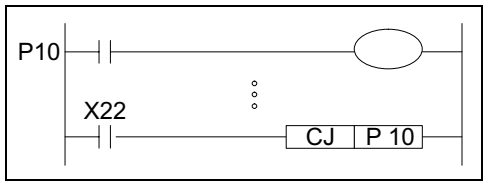
c) Any program area which is skipped, will not update output statuses even if the input devices change. For example, the program opposite shows a situation which loads X1 to drive Y1. Assuming X1 is ON and the CJ instruction is activated the load X1, out Y1 is skipped. Now even if X1 is turned OFF Y1 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 X1 is OFF to begin with and the CJ instruction is driven, Y1 will not be turned ON if X1 is turned ON. Once the CJ instruction is deactivated X1 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. to-wards 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. For more information on the watchdog timer please see page 5-12.



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

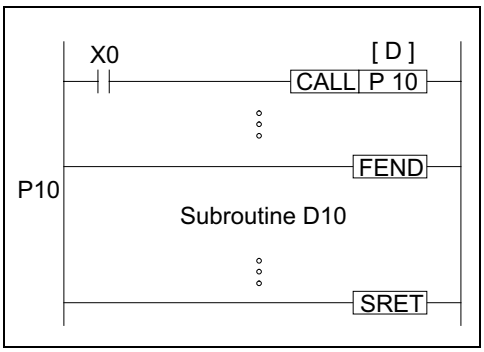
- Further details can be found on pages 7-12 and 7-13 about the combined use of different program flow techniques (such as master control, MC etc).

5.1.2 CALL (FNC 01)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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 Nest levels: 5 including the initial CALL	CALL, CALLP: 3 step Subroutine pointer PPP: 1 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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.

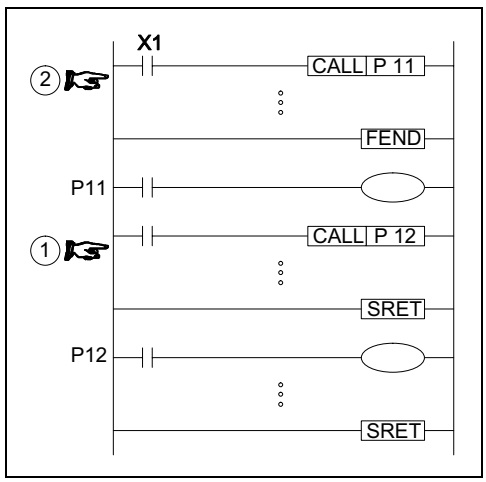
Points to note:

a) Many CALL statements can reference a single subroutine.

b) 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.

c) 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. For more information on watchdog timers please see page 5-12.

d) Subroutines can be nested for 5 levels including the initial CALL instruction. As an example the program shown opposite shows a 2 level nest. When X1 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).



Special subroutine timers:

- Because of the chance of intermittent use of the subroutines, if timed functions are required the timers used must be selected from the range T192 to T199 and T246 to T249.

Details of using CALL 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).

5.1.3 SRET (FNC 02)

$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$

Mnemonic	Function	Operands	Program steps
		D	
SRET FNC 02 (Subroutine 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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$	$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$	$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$

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:

a) SRET can only be used with the CALL instruction.

b) SRET is always programmed after an FEND instruction - please see the CALL (FNC 01) instruction for more details.

5.1.4 IRET, EI, DI (FNC 03, 04, 05)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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:

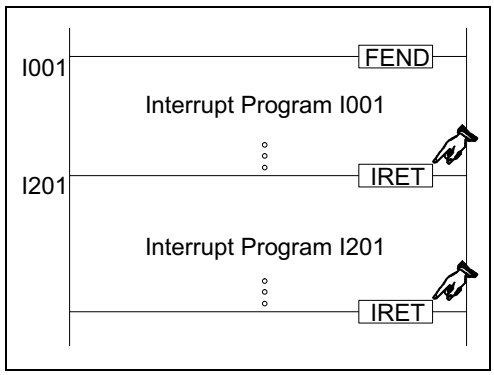
An interrupt routine is a section of program which is, when triggered, operated immediately interrupting the main program flow. Once the interrupt has been processed the main program flow continues from where it was, just before the interrupt originally occurred.

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. The availability of different interrupt types and the number operational points for each PLC type are detailed on page 4-12, Interrupt Pointers. To program and operate interrupt routines requires up to 3 dedicated instructions (those detailed in this section) and an interrupt pointer.

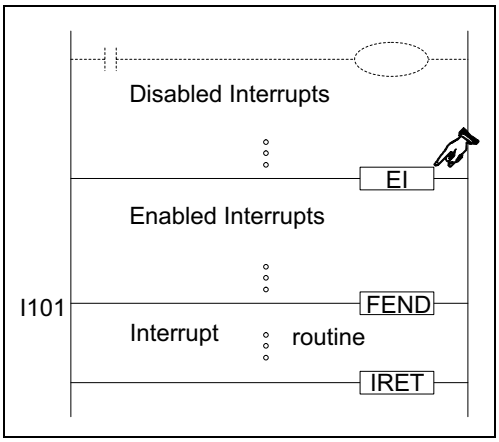
Defining an interrupt routine:

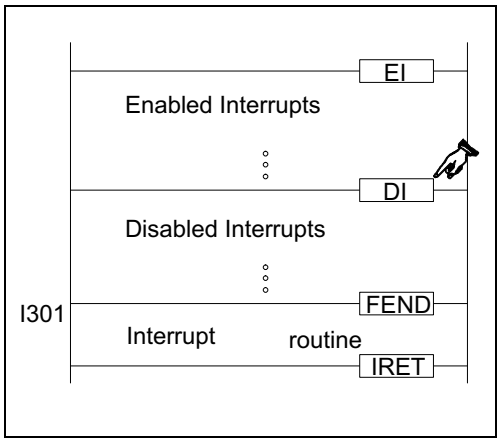
An interrupt routine is specified between its own unique interrupt pointer and the first occurrence of an IRET instruction.
Interrupt routines are ALWAYS programmed after an FEND instruction.
The IRET instruction may only be used within interrupt routines.



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 8.

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:

FX PLC's have a range of special timers which can be used within interrupt routines. For more information please see page 4-18, 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 200µsec, i.e. the input pulse width is equal or greater than 200µsec. When this type of interrupt is selected, the hardware input filters are automatically reset to 50µsec. (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. FX0, FX0S and FX0N units have this function permanently active for inputs X0 through X3 with special M coils storing the pulse data at M8056 to M8059. FX(2C) and FX2N units require 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.

5.1.5 FEND (FNC 06)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

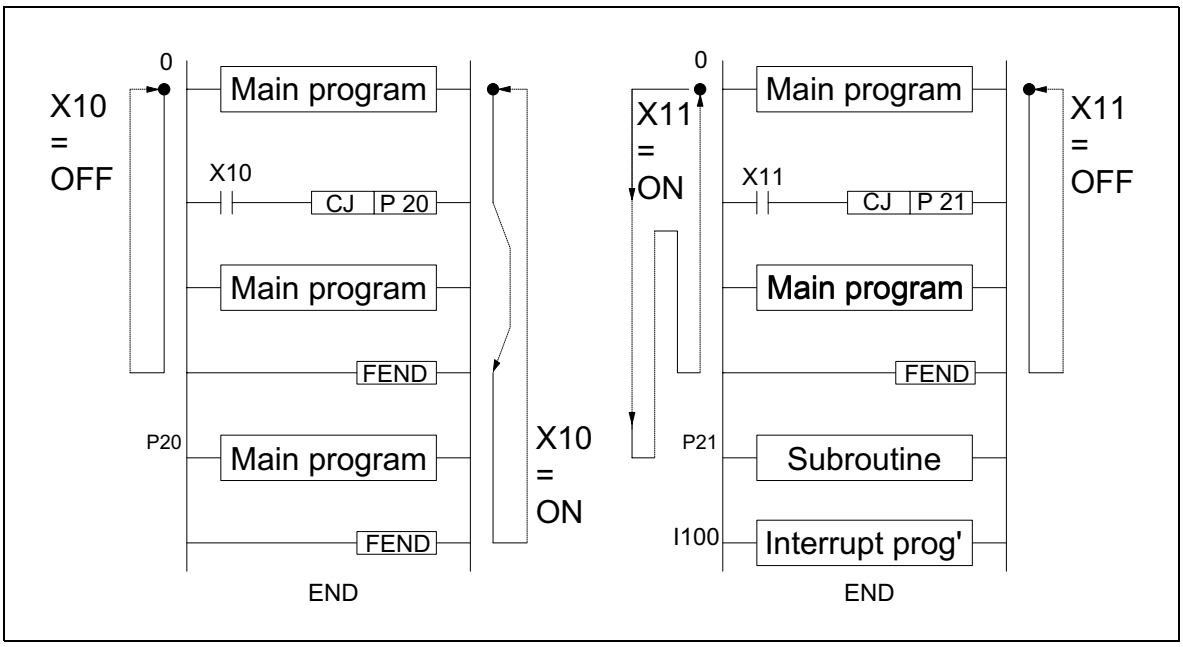
Mnemonic	Function	Operands	Program steps
		D	
FEND FNC 06 (First end)	Used to indicate the end of the main program block	N/A Note: Can be used with CJ (FNC 00), CALL (FNC 01) and interrupt routines	FEND: 1 step

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:

a) 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.



b) Multiple occurrences of FEND instructions can be used to separate different subroutines (see diagram above).

c) The program flow constructions are NOT allowed to be split by an FEND instruction.

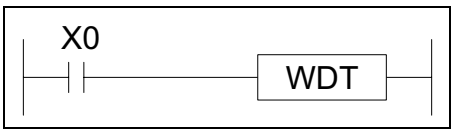
d) FEND can never be used after an END instruction.

5.1.6 WDT (FNC 07)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

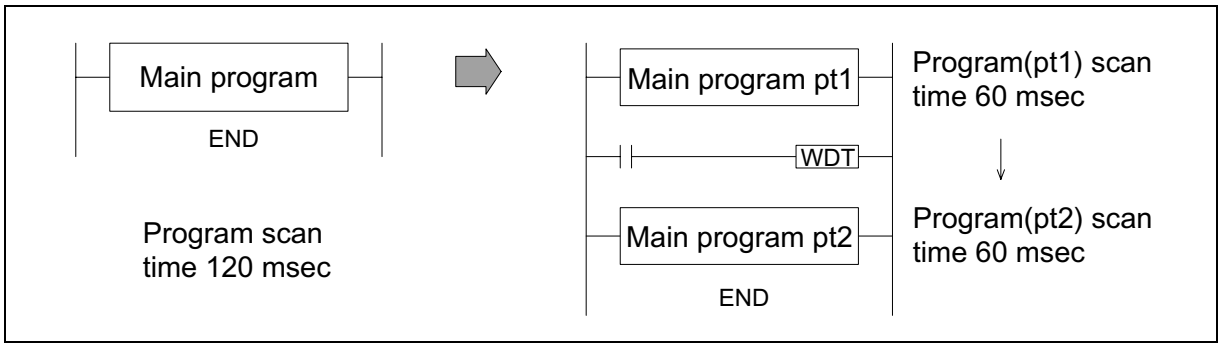
Mnemonic	Function	Operands	Program steps
		D	
WDT FNC 07 (Watch dog timer refresh)	Used to refresh the watch dog timer during a program scan	N/A Can be driven at any time within the main program body	WDT, WDTP: 1 step

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



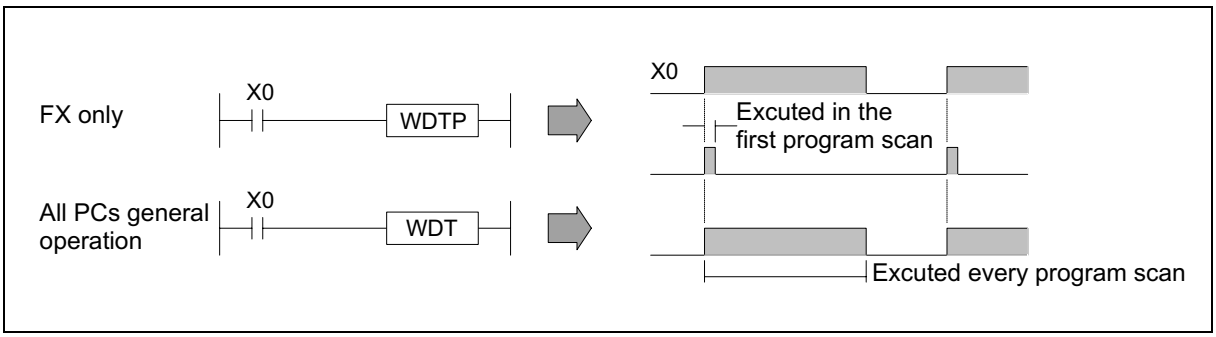
Operation:

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.

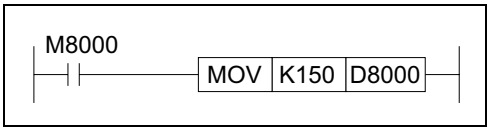


Points to note:

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 100 msec for FX PLC's and 200 msec for FX0/FX0N/ FX2N PLC's. This time limit may be customized to a users own requirement by editing the contents of data register D8000, the watchdog timer register.

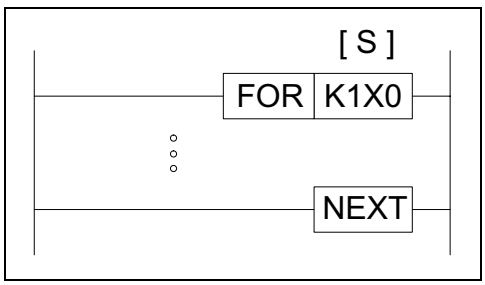


5.1.7 FOR, NEXT (FNC 08, 09)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands	Program steps
		S	
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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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.

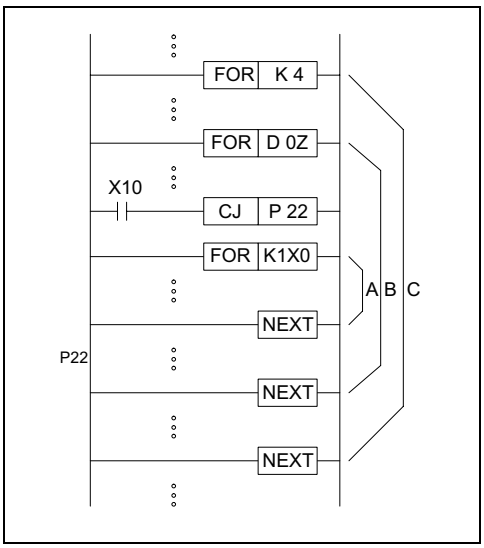
d) A FOR-NEXT loop operates for its set number of times **before** the main program is allowed to finish the current program scan.

e) When using FOR-NEXT loops care should be taken not the 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 5 levels. This means that 5 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.

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.2 Move And Compare - Functions 10 to 19

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

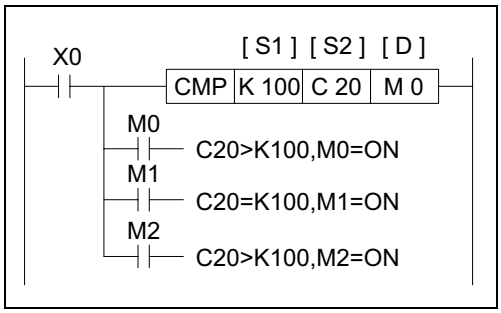
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.2.1 CMP (FNC 10)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The data of S_1 is compared to the data of S_2. The result is indicated by 3 bit devices specified from the head address entered as D. The bit devices indicate:
S_2 is less than S_1 - bit device D is ON
S_2 is equal to S_1 - bit device D_{+1} is ON
S_2 is greater than S_1 - bit device D_{+2} is ON

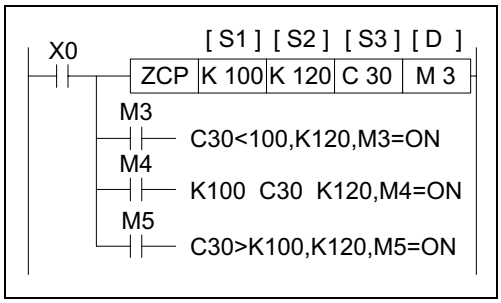
Note: The destination (D) device statuses will be kept even if the CMP instruction is deactivated. Full algebraic comparisons are used, i.e. -10 is smaller than +2 etc.

5.2.2 ZCP (FNC 11)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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: S_1 should be less than S_2			Y, M, S Note: 3 consecutive devices are used.	ZCP,Z CPP: 9 steps DZCP, DZCPP: 17 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

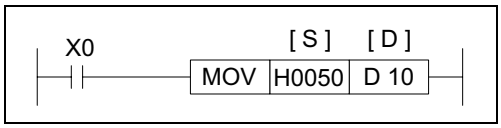
The operation is the same as the CMP instruction except a single data value (S_3) is compared against a data range (S_1-S_2).
S_3 is less than $S_1$and S_2- bit device D is ON
S_3 is equal to or between S_1 and S_2 - bit device D_{+1} is ON
S_3 is greater than both S_1 and S_2 - bit device D_{+2} is ON

5.2.3 MOV (FNC 12)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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, MOVP: 5 steps DMOV, DMOVP: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



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.

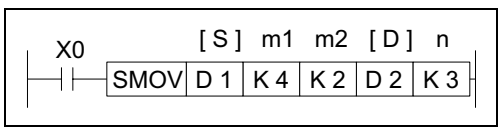
Note: This instruction has a special programming technique which allows it to mimic the operation of newer applied instructions when used with older programming tools. See page 1-5 for more details.

5.2.4 SMOV (FNC 13)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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	K, H, KnY, KnM, KnS, T,C,D,V,Z	SMOV, SMOVP: 11 steps
					Range 0 to 9,999 (decimal) or 0 to 9,999 (BCD) when M8168 is used - see note opposite		

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)

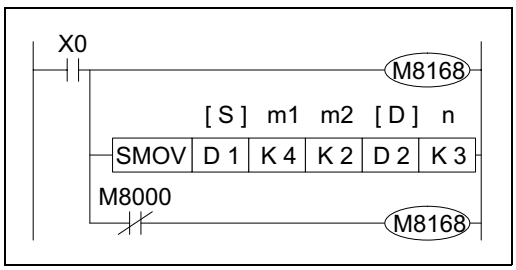


Operation 1:

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. The decimal manipulation mode is available to all FX and FX2C units.
Key:
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.

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Operation 2: (Applicable units are FX units with CPU's ver 3.07 or greater and FX2C's). 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.

General note:

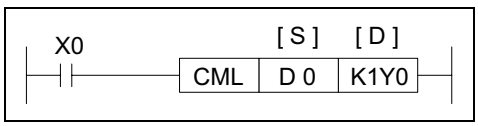
For more information about 'decimal' and 'Binary Coded Decimal' (BCD) numbers please see the section titled 'Interpreting Word Data' on page 4-42 for more details.

5.2.5 CML (FNC 14)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		S	D	
CML FNC 14 (Compli- ment)	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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

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

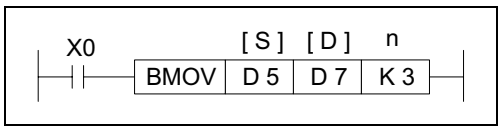
This means each occurrence of a '1' in the source data will become a '0' in the destination data while each source digit which is '0' will become a '1'. If the destination area is smaller than the source data then only the directly mapping bit devices will be processed.

5.2.6 BMOV (FNC 15)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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 (RAM) File registers,	KnY, KnM, KnS, T, C, D, V, Z (RAM) File registers, see note d)	K, H D (FX2C, FX2N only) ⊠ Note: n≤ 512	BMOV, BMOVP: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

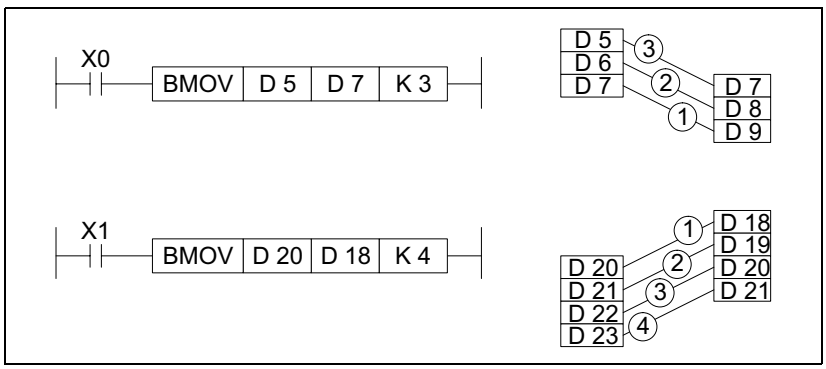
A quantity of consecutively occurring data elements can be copied to a new destination. The source data is identified as a device head address (S) and a quantity of consecutive data elements (n). This is moved to the destination device (D) for the same number of elements (n).

Points to note:

a) 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.

b) If the number of source devices exceeds the available space at the destination location, then only the available destination devices will be written to.

c) 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)



FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

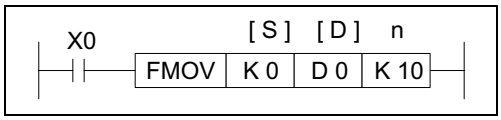
d) Using file registers as the destination devices [D]may only be performed on FX Main Processing Units (MPUs) with a CPU version 3.07 or greater or on any FX2C or FX2N(C) MPU.

5.2.7 FMOV (FNC 16)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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,FMOVP:7 steps DFMOV,DFMOVP: 13 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



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. Please note that double word (32 bit) operation can only be performed by FX units with ver 3.07 CPU's or greater and FX2C units.

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. See page 1-5 for more details.

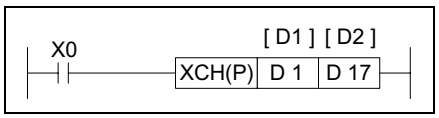
5.2.8 XCH (FNC 17)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

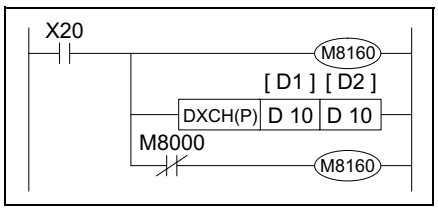
PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)

Operation 1: (Applicable units: FX and FX2C).The contents of the two destination devices D1 and D2are swapped, i.e. the complete word devices are exchanged. Ex.



Data register	Before XCH	After XCH
D1	20	530
D17	530	20

Operation 2: (Applicable units: FX(2C)) 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.



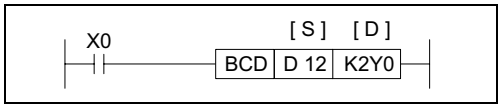
Values are in Hex for clarity		Before DXCH	After DXCH
D10	Byte 1	1FH	8BH
	Byte 1	8BH	1FH
D11	Byte 1	C4H	35H
	Byte 1	35H	C4H

5.2.9 BCD (FNC18)

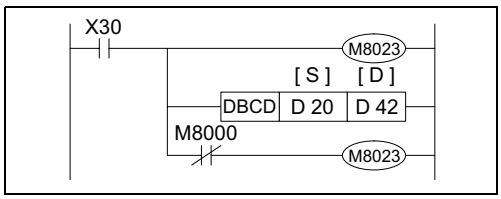
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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 DBCD, DBCDP: 9 steps
		When using M8023 to convert data to scientific format, only double word (32 bit) data registers (D) may be used. See page 4-46 for more details regarding floating point format.		

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation 1: (Applicable to all units)
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.



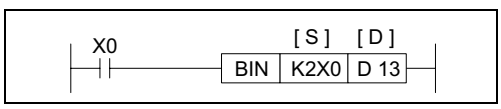
Operation 2: (Applicable units: FX(2C))
This function is equivalent to FNC 118 EBCD Data (S)is converted from 'floating point' format to 'scientific format' (D). This instruction requires double word (32 bit) operation and data registers as devices (S)and (D)to operate correctly.

5.2.10 BIN (FNC 19)

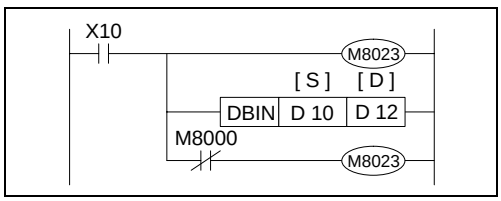
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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. See page 4-46 for more details regarding floating point format.		

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation 1: (Applicable to all units)
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: (Applicable units: FX(2C))
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.

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.3 Arithmetic And Logical Operations - Functions 20 to 29

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

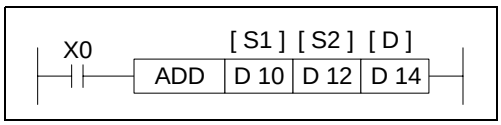
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.3.1 ADD (FNC 20)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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. See page 4-46 for more details regarding floating point format.			

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		Zero M8020 Borrow M8021 Carry M8022



Operation 1: (Applicable to all units)

The data contained within the source devices (S_1,S_2) is combined and the total is stored at the specified destination device (D).

Points to note:

a) All calculations are algebraically processed, i.e. 5 + (-8)= -3.

b) The same device may be used as a source (S_1 or S_2) 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.

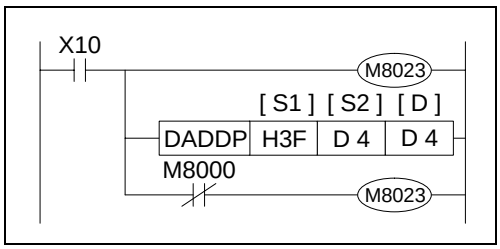
c) If the result of a calculation is “0" then a special auxiliary flag, M8020 is set ON.

d) 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.

e) 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!

Continued over the page....

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation 2: (Applicable units: FX(2C))
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.

FX2N Support for floating point operations

Note: The use of M8023 is not supported in FX2N units.
The appropriate dedicated floating point instruction should be used instead E.g. Instead of DADD with M8023 ON, use FNC 120, DEADD.

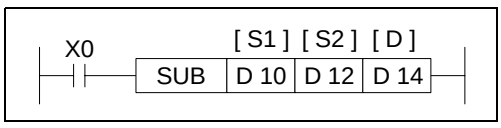
- See section 5.11

5.3.2 SUB (FNC 21)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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: 7steps DSUB, DSUBP: 13 steps
		When using M8023 to subtract floating point data, only double word (32 bit) data registers (D) or constants (K/H) may be used. See page 4-46 for more details regarding floating point format.			

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Zero M8020 Borrow M8021 Carry M8022
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



Operation 1: (Applicable to all units)

The data contained within the source device, S2 is subtracted from the contents of source device S1. 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.

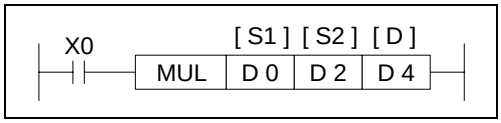
Operation 2: (Applicable units: FX(2C)) This function is equivalent to FNC 121 ESUB. The information regarding 'Operation2:' of the ADD instruction apply similarly to this second operation of the SUB instruction (with the exception of a subtraction being performed instead of an addition). Again, only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

5.3.3 MUL (FNC 22)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
MUL FNC 22 (Multiplica -tion)	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 See page 4-46 for more details regarding floating point format. When using M8023 to subtract floating point data, only double word (32 bit) data registers (D) or constants (K/H) may be used.		KnY,KnM,KnS, T, C, D, Z(V) Note: Z(V) may **NOT** be used for 32 bit operation	MUL, MULP: 7steps DMUL, DMULP: 13 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation 1: (Applicable to all units)
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:

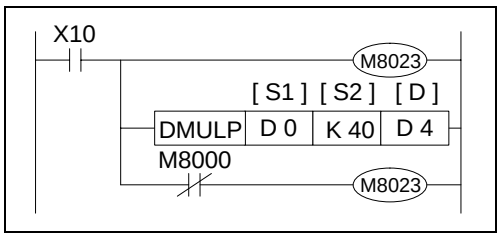
a) When operating the MUL instruction in 16bit mode, two 16 bit data sources are multiplied together. They produce a 32 bit result. The device identified as the destination address is the lower of the two devices used to store the 32 bit result. Using the above example with some test data:
5 (D0) × 7 (D2) = 35 - The value 35 is stored in (D4, D5) as a single 32 bit word.

b) 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.

c) 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!

Viewing 64 bit numbers

- It is currently impossible to monitor the contents of a 64 bit result. However, the result can be monitored in two smaller,32 bit, blocks, i.e. a 64 bit result is made up of the following parts: (upper 32 bits) × 2^{32} + (lower 32 bits).



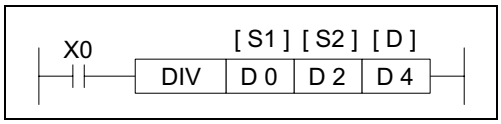
Operation 2: (Applicable units: FX(2C))
This function is equivalent to FNC 122 EMUL. When 'floating point mode flag' M8023 is active, i.e. ON, DMUL and DMULP instructions can be used to perform floating point multiplications.
When M8023 is reset, i.e. OFF floating point manipulation will not occur. Constants (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.

5.3.4 DIV (FNC 23)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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)	DIV,DIVP: 7steps DDIV, DDIVP: 13 steps
		See page 4-46 for more details regarding floating point format.		Note: Z(V) may **NOT** 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.used to perform			

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation 1: (Applicable to all units)

The primary source (S_1) is divided by the secondary source (S_2). The result is stored in the destination (D). Note the normal rules of algebra apply.

Points to note:

a) 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 S_2 will divide into S_1 (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) ÷ 10 (D2) = 5(D4) 1(D5)
This result is interpreted as 5 whole divisions with 1 left over ($5 \times 10 + 1 = 51$).

b) 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.

c) If the value of the source device S_2 is 0 (zero) then an operation error is executed and the operation of the DIV instruction is cancelled.

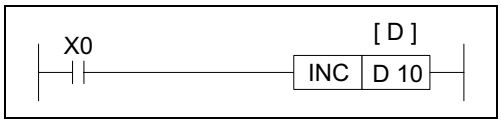
Operation 2: (Applicable units FX(2C)) This function is equivalent to FNC 123 EDIV. The information regarding 'Operation2:' of the MUL instruction apply similarly to this second operation of the DIV instruction (with the exception of a division being performed instead of a multiplication). Again, only 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.

5.3.5 INC (FNC 24)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

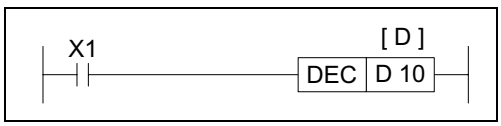
On every execution of the instruction the device specified as the destination D, has its current value incremented (increased) by a value of 1.
In 16 bit operation, when +32,767 is reached, the next increment will write a value of -32,768 to the destination device.
In 32 bit operation, when +2,147,483,647 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.

5.3.6 DEC (FNC 24)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

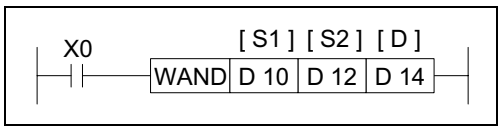
On every execution of the instruction the device specified as the destination D, has its current value decremented (decreased) by a value of 1.
In 16 bit operation, when -32,768 is reached the next increment will write a value of +32,767 to the destination device.
In 32 bit operation, when -2,147,483,648 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.

5.3.7 WAND (FNC 26)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The bit patterns of the two source devices are analyzed (the contents of $S_2$is compared against the contents of S_1). 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: (S_1) Bit n WAND (S_2) Bit n = (D) Bit n

1 WAND 1 = 1 0 WAND 1 = 0

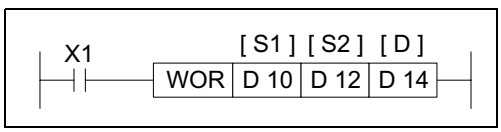
1 WAND 0 = 0 0 WAND 0 = 0

5.3.8 WOR (FNC 27)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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,WORP: 7 steps DOR, DORP: 13 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The bit patterns of the two source devices are analyzed (the contents of $S_2$is compared against the contents of S_1). 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: (S_1) Bit n WOR (S_2) Bit n = (D) Bit n

1 WOR 1 = 1 0 WOR 1 = 1

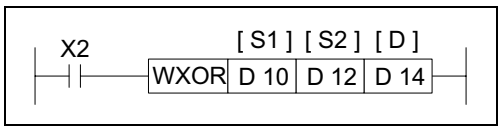
1 WOR 0 = 1 0 WOR 0 = 0

5.3.9 WXOR (FNC 28)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The bit patterns of the two source devices are analyzed (the contents of S_2 is compared against the contents of S_1). 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: (S_1)Bit $_n$ WXOR (S_2)Bit $_n$ = (D)Bit $_n$

1 WXOR 1 = 0 0 WXOR 1 = 1

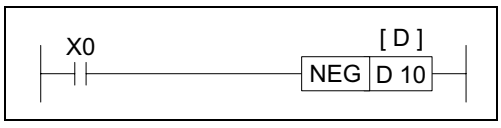
1 WXOR 0 = 1 0 WXOR 0 = 0

5.3.10 NEG (FNC 29)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



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.

MEMO

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.4 Rotation And Shift - Functions 30 to 39

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

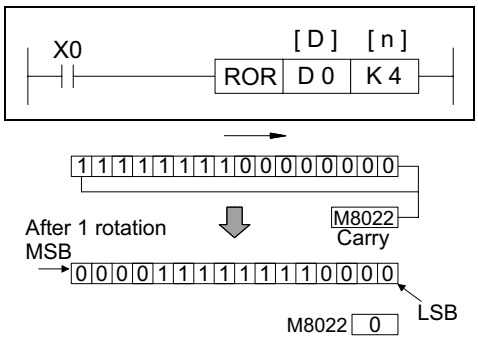
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.4.1 ROR (FNC 30)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Carry M8022
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



Operation:

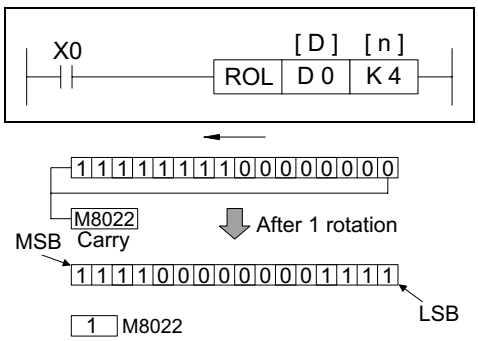
The bit pattern of the destination device (D) is rotated n bit places to the right on every operation of the instruction.
The status of the last bit rotated is copied to the carry flag M8022.
The example shown left is based on the instruction noted above it, where the bit pattern represents the contents of D0.

5.4.2 ROL (FNC 31)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Carry M8022
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



Operation:

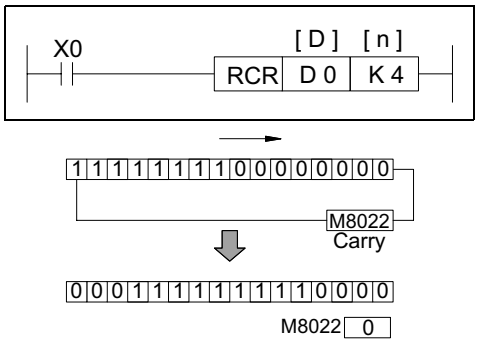
The bit pattern of the destination device (D) is rotated n bit places to the left on every operation of the instruction.
The status of the last bit rotated is copied to the carry flag M8022.
The example shown left is based on the instruction noted above it, where the bit pattern represents the contents of D0.

5.4.3 RCR (FNC 32)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Carry M8022
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



Operation:

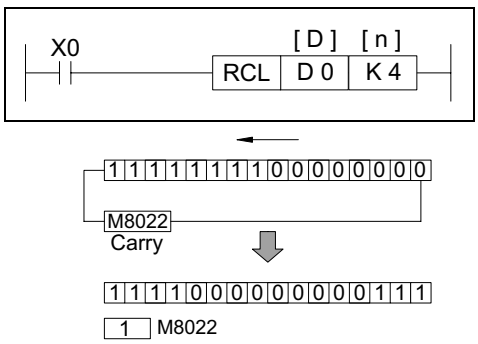
The bit pattern of the destination device (D)is rotated n bit places to the right on every operation of the instruction.
The status of the last bit rotated is moved into the carry flag M8022. On the following operation of the instruction M8022 is the first bit to be moved back into the destination device.
The example shown left is based on the instruction noted above it, where the bit pattern represents the contents of D0.

5.4.4 RCL (FNC 33)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Carry M8022
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



Operation:

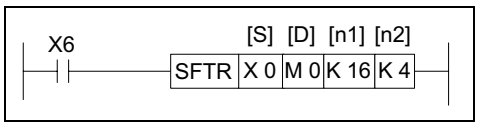
The bit pattern of the destination device (D)is rotated n bit places to the left on every operation of the instruction.
The status of the last bit rotated is moved into the carry flag M8022. On the following operation of the instruction M8022 is the first bit to be moved back into the destination device.
The example shown left is based on the instruction noted above it, where the bit pattern represents the contents of D0.

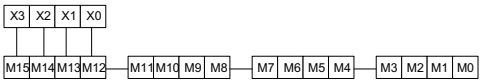
5.4.5 SFTR (FNC 34)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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: FX users: $n_2 \le n_1 \le 1024$ FX0,FX0N users: $n_2 \le n_1 \le 512$		SFTR,SFTRP: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)





Operation:

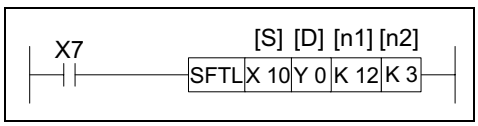
The instruction copies n_2 source devices to a bit stack of length n_1. For every new addition of n_2 bits, the existing data within the bit stack is shifted n_2 bits to the right. Any bit data moving to a position exceeding the n_1 limit is diverted to an overflow area. The bit shifting operation will occur every time the instruction is processed unless it is modified with either the pulse suffix or a controlled interlock.

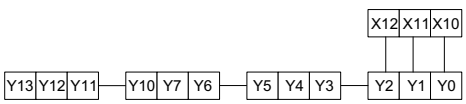
5.4.6 SFTL (FNC 35)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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: FX users: $n_2 \le n_1 \le 1024$ FX0,FX0N users: $n_2 \le n_1 \le 512$		SFTL,SFTLP: 9steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)





Operation:

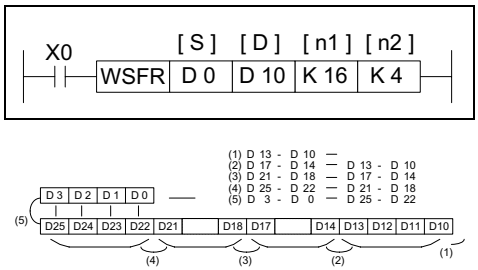
The instruction copies n_2 source devices to a bit stack of length n_1. For every new addition of n_2 bits, the existing data within the bit stack is shifted $n_2$bits to the left. Any bit data moving to a position exceeding the n_1 limit is diverted to an overflow area. The bit shifting operation will occur every time the instruction is processed unless it is modified with either the pulse suffix or a controlled interlock.

5.4.7 WSFR (FNC 36)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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: FX users: n2 ≤ n1 ≤ 512		WSFR, WSFRP: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The instruction copies n2 source devices to a word stack of length n1. For each addition of n2words, the existing data within the word stack is shifted n2words to the right. Any word data moving to a position exceeding the n1limit 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.

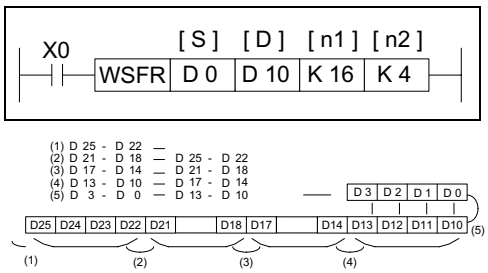
Note: when using bit devices as source (S) and destination (D) the Kn value must be equal.

5.4.8 WSFL (FNC 37)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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: FX users: n2 ≤ n1 ≤ 512		WSFL, WSFLP: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The instruction copies n2 source devices to a word stack of length n1. For each addition of n2words, the existing data within the word stack is shifted n2words 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.

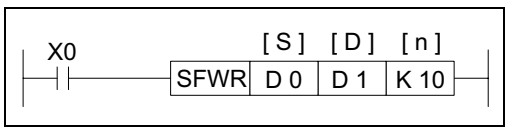
Note: when using bit devices as source (S) and destination (D) the Kn value must be equal.

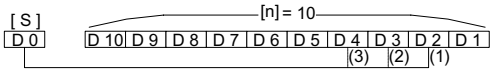
5.4.9 SFWR (FNC 38)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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≤ 512	SFWR, SFWRP: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Carry M8022
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		





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:

a) FIFO is an abbreviation for ‘First-In/ First-OUT’.

b) 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.

c) Before starting to use a FIFO stack ensure that the contents of the head address register (D) are equal to ‘0’ (zero).

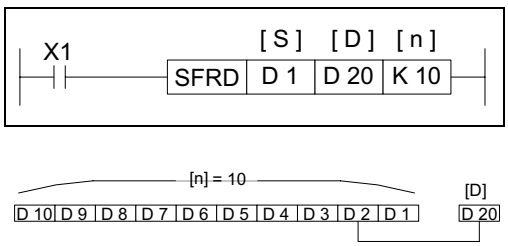
d) This instruction should be used in conjunction with SFRD FNC 39. The n parameter in both instructions should be equal.

5.4.10 SFRD (FNC 39)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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	KnY, KnM, KnS, T, C, D, KnY, KnM,KnS, T, C, D, V, Z	K,H, ⊠ Note: 2≤ n≤ 512	SFRD, SFRDP: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Zero M8020
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



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, the flag M8020 is turned ON.

Points to note:

a) FIFO is an abbreviation for 'First-In/ First-OUT'.

b) 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.

c) This instruction will always read the source data from the register S_{+1}.

d) This instruction should be used in conjunction with SFWR FNC 38. The n parameter in both instructions should be equal.

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.5 Data Operation - Functions 40 to 49

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

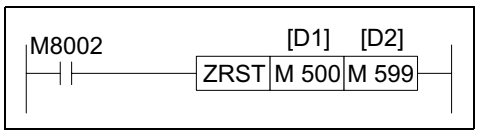
⊠ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.5.1 ZRST (FNC 40)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		S	D	
ZRST FNC 40 (Zone Reset)	Used to reset a range of like devices in one operation	Y, M,S, T, C, D Note: D_1must be less than or equal (≤) to D_2. Standard and High speed counters cannot be mixed.		ZRST, ZRSTP: 5 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



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).

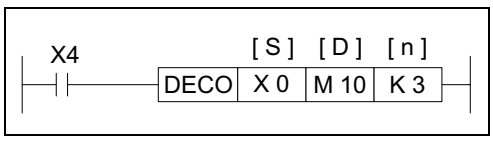
The specified device range cannot contain mixed device types, i.e. C000 specified as the first destination device (D_1) cannot be paired with T199 as the second destination device (D_2). When resetting counters, standard and high speed counters cannot be reset as part of the same range.
If $D_1$is greater than (>) D_2 then only device D_1 is reset.

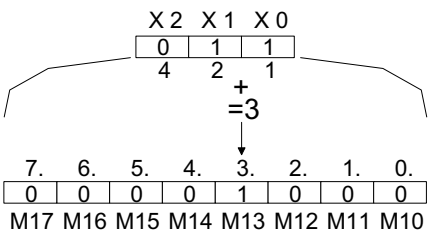
5.5.2 DECO (FNC 41)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)





Operation:

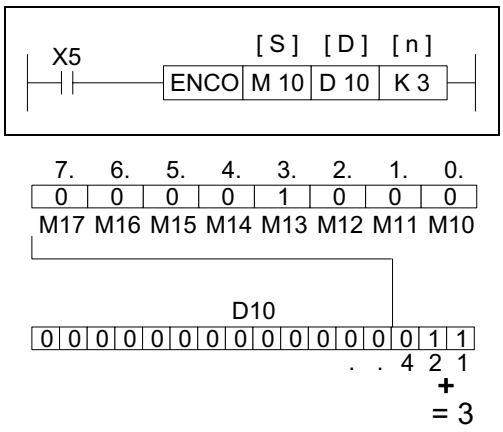
Source data is provided by a combination of operands S and n. Where S specifies the head address of the data and n, the number of consecutive bits. The source data is read as a single number (binary to decimal conversion) Q. The source number Q is the location of a bit within the destination device (D) 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.

5.5.3 ENCO (FNC 42)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



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:

a) 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.

b) 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.

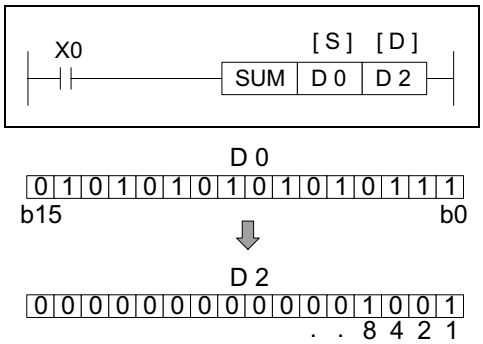
c) When the source device is a data or word device n must be taken from the range 1 to 4 as there are only 16 source bits available within a single data word.

5.5.4 SUM (FNC 43)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Zero M8020
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



Operation:

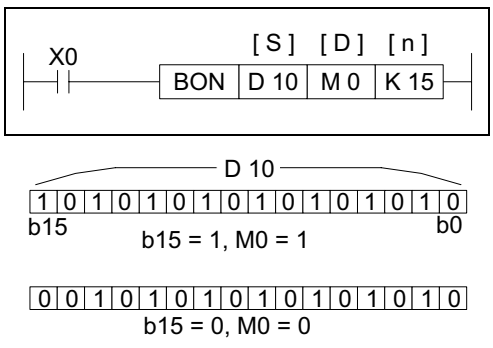
The number of active (ON) bits within the source device (S), i.e. bits which have a value of “1" are counted. The count is stored in the destination register (D). If a double word format is used, both the source and destination devices use 32 bit, double registers. The destination device will always 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.

5.5.5 BON (FNC 44)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

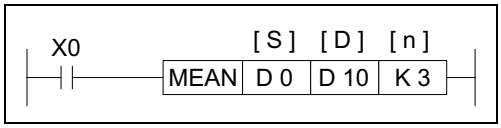
A single bit position (n) is specified from within a source device/area (S). n could be regarded as a specified offset from the source head address (S), i.e. 0 (zero) being the first device (a 0 offset) where as an offset of 15 would actually be the 16th device.
If the identified bit becomes active,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

5.5.6 MEAN (FNC 45)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



General rule

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

Example

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

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.

Points to note:

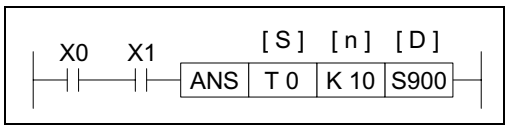
If the source area specified is actually smaller than the physically available area, then only the available devices are used. The actual value of n used to calculate the mean will reflect the used, available devices. However, the value for n which was entered into the instruction will still be displayed. 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 nis specified outside of the stated range (1 to 64) an error is generated.

5.5.7 ANS (FNC 46)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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 ⊠ Note: n range 1 to 32,767 - in units of 100msec	ANS: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



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.

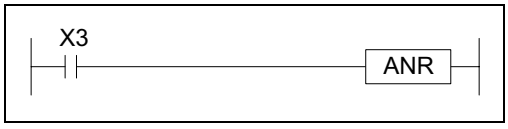
Note: This is only one method of driving annunciator coils, others such as direct setting can also be used.

5.5.8 ANR (FNC 47)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

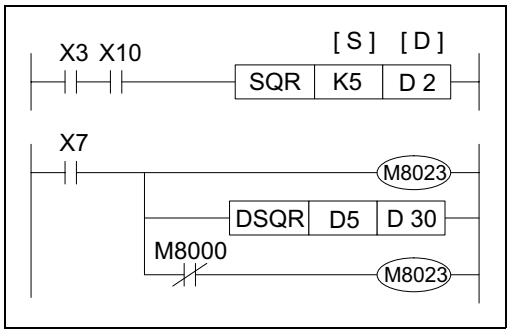
Annunciators which have been activated are sequentially reset one-by-one, each time the ANR instruction is operated.
If the ANR instruction is driven continuously it will carry out its resetting operation on every program scan unless it is modified by the pulse, P prefix or by a user defined program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

5.5.9 SQR (FNC 48)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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, SQRP: 5 steps DSQR, DSQRP: 9 steps
		When using M8023 in float mode, only double word (32bit) data can be processed. See page 4- 46 for more details regarding floating point.		

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Zero M8020 Borrow M8021
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



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.

Operation 2: This function is equivalent to FNC 127 ESQR This operation is similar to Operation 1. However, it is only activated when the mode setting float flag, M8023 is used. This then allows the SQR instruction to process answers in floating point format. The source data (S) must either be supplied in floating point format for data register use, or it can be supplied as a constant (K,H). When constants are used as a source, they are automatically converted to floating point format. Operation 2 is only valid for double word (32 bit) operation, hence both (S) and (D) will be 32 bit values and the SQR instruction will be entered as DSQR or DSQRP.

General note:

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:

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

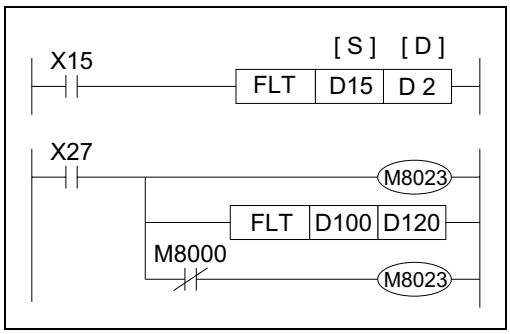
This is true for both operating modes.

5.5.10 FLT (FNC 49)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		S	D	
FLT FNC 49 (Floating point)	Used to convert data to and from floating point format	D	D	FLT, FLTP: 5 steps DFLT, DFLTP: 9 steps
		M8023 = OFF data is converted from decimal to floating point format M8023 = ON data is converted from floating point format to decimal format		

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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 destination devices (D) used to store the floating point number. Examples:

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

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

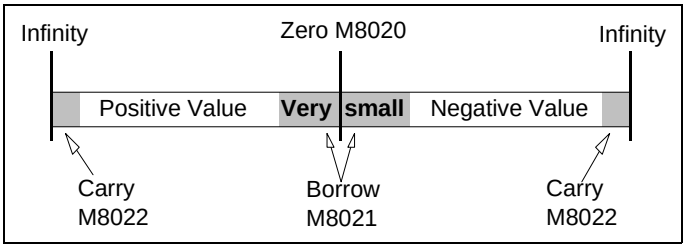
Operation 2: (Applicable units: FX(2C)) This function is equivalent to FNC 129 INT.
When the float instruction is performed and the float flag M8023 is ON, the float operation will be conducted in reverse to Operation 1. Any floating point format number stored at source (S) will be converting to its decimal equivalent and stored at destination (D).

Continued over the page...

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 al-low enough destination devices, i.e.



Instruction	Double word operation	Status of M8023	Number of source registers (S)	Number of destination registers (D)	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

General note:

For more information about float and scientific notations please see Chapter 4, Advanced Devices, page 4-46

Applied Instructions:

FX0(S) FX0N FX FX(2C) FX2N(C)

5.6 High Speed Processing - Functions 50 to 59

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

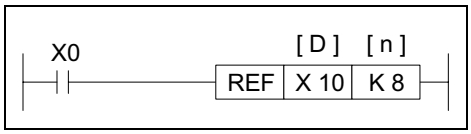
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.6.1 REF (FNC 50)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		D	n	
REF FNC 50 (Refresh) ✈å	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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



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.

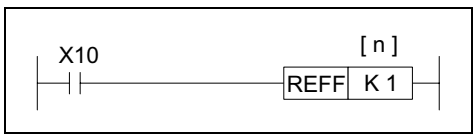
Note: A short delay will occur before the I/O device is physically updated, in the case of inputs a time equivalent to the filter setting, while outputs will delay for their set energized time.

5.6.2 REFF (FNC 51)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands	Program steps
		n	
REFF FNC 51 (Refresh and filter adjust)	Inputs are refreshed, and their input filters are reset to the newly designated value	K, H, ⊠ Note: n= 0 to 60 msec (0 = 50µs) X000 to X007 (X000 to X017 for FX2N) are automatically designated when using this instruction	REFF, REFFP: 3 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

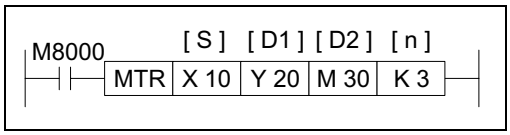
PLC's are provided with input filters to overcome problems generated by mechanical switch gear. However, as this involves ensuring a steady input signal is received for a fixed time duration, the use of input filters slows down the PLC response times. For high speed applications, especially where solid state switching provides the input signal, input filter times may be reduced. The default setting for the input filters is approximately 10 msec. Using this instruction input filter times of 0 to 60 msec may be selected. The setting '0' (zero) is actually 50 µsec. This is the minimum available setting. It is automatically selected when direct input, interrupts or high speed counting functions are used. The REFF instruction needs to be driven for each program scan if it is to be effective, otherwise, the standard 10 msec filter time is used.

5.6.3 MTR (FNC 52)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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, ⊠ Note: n=2 to 8	MTR: 9 steps
		Note: These operands should always be a multiple of 10, i.e. 00, 10, 20, 30 etc.				

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Operation Complete M8029
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



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 (D_1) signal being processed. The result is stored in a matrix-table (head address D_2).

Points to note:

a) The MTR instruction involves high speed input/output switching. For this reason this instruction is only recommended for use with transistor output modules.

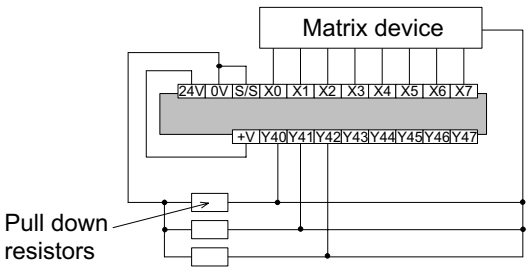
b) For the MTR instruction to operate correctly, it must be driven continuously. It is recommended that special auxiliary relay M8000, the PLC RUN status flag, is used. After the completion of the first full reading of the matrix, operation complete flag M8029 is turned ON. This flag is automatically 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).

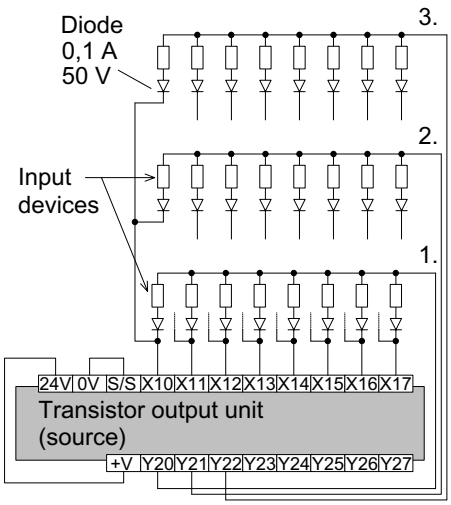
d) Each bank is triggered/selected by a dedicated output (head address D_1). This means the quantity of outputs from D_1, used to achieve the matrix are equal to the number of banks n. As there are now additional inputs entering the PLC these will each have a status which needs recording. This is stored in a matrix-table. The matrix-table starts at the head address D_2. 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 wouldresultinan8bankmatrix, i.e. 64inputs(8inputs´8banks) beingreadin160msec.

If high speed inputs (ex. X0) is 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 places 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 × 3 = 60msec.

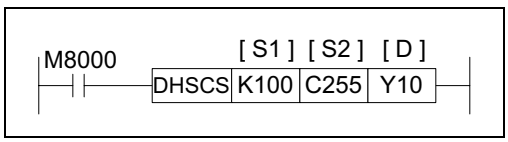
Notice how the resulting matrix-table does not use any of the P8 and P9 bit devices when state S or auxiliary M relays are used as the storage medium.

5.6.4 HSCS (FNC 53)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands			Program steps
		S1	S2	n	
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, V, Z	C Note: C = 235 to 254, or available high speed counters	Y, M, S Interrupt pointers I010 to I060 can be set on FX units from CPU ver 3.07 and FX2C units	DHSCS: 13 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



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 S1the device specified as the destination (D)is set ON.The example above shows that Y10 would be set ON only when C255'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:

a) It is recommended that the drive input used for the high speed counter functions; HSCS, HSCR, HSCZ is the special auxiliary RUN contact M8000.

b) 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.

c) All high speed counter functions use an interrupt process, hence, all destination devices (D) are updated immediately.

Note:

FX0,FX0N users - Max. 4 simultaneously active HSCS/R instructions. FX users Max. 6 simultaneously active HSCS/R and HSZ instructions. Please remember that the use of high speed counter functions has a direct impact on the maximum allowable counting speed! See page 4-22 for further details.

Use of interrupt pointers

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

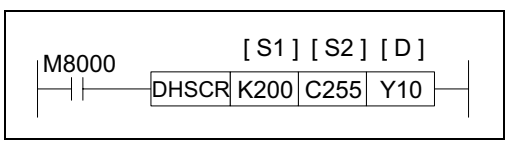
FX(2C) and FX2N MPUs can use interrupt pointers I010 through I060 (6 points) as destination devices (D). This enables interrupt routines to be triggered directly when the value of the specified high speed counter reaches the value in the HSCS instruction.

5.6.5 HSCR (FNC 54)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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, V, Z	C Note: C = C235 to C255, or available high speed counters	Y, M, S C Note: If C, use same counter as S2	DHSCR: 13 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

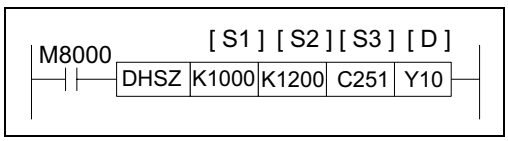
The HSCR, compares the current value of the selected high speed counter (S2) against a selected value (S1). When the counters current value changes to a value equal to S1, the device specified as the destination (D) is reset. In the example above, Y10 would be reset only when C255's value stepped from 199 to 200 or from 201 to 200. If the current value of C255 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.

5.6.6 HSZ (FNC 55)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands				Program steps
		S1	S2	S3	D	
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, V, Z		C Note: C = 235 to 255,	Y, M, S Note: 3 consecutive devices are used	DHSZ: 17 steps
	Operation 2: The designated range is held in a data table driving 'Y' outputs directly	D	K,H Using values from 1 to 128 (deci-mal)		M8130 (only) This flag can only be used with one DHSZ instr' at a time	
	Operation 3: The designated range is held in a data table driving PLSY frequencies directly using D8132				M8132 (only) This flag can only be used with one	

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation 1 - Standard: (Applicable to all units)
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 S1, K1000 (S3< S1)
Y11 (D+1) C251 is greater than S1, K1000 but less than S2, K1200 (S3> S1, S3< S2)
Y12 (D+2) C251 is greater than S2, K1200 (S3> S2)

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.

Operation 2 - Using HSZ With A Data Table: (Applicable units: FX(2C) and FX2N)
Operation 2 is selected when the destination device (D) is assigned special M coil M8130. This then allows devices (S1, S2) to be used to define a data table using (S1) as the head address and (S2) as the number of records in the table - maximum number of records is 128. Each record occupies 4 consecutive data registers proportioned in the following manner (for a single record of data registers D through D+3):

	Single Record	
Data registers	D, D+1	Used as a double (32 bit) data register to contain the comparison data
	D+2	Stores the I/O device number, in HEX, of the 'Y' Output device to be controlled, i.e. H10=Y10. Note: Hex digits A through F are not used.
	D+3	Stores the action (SET/RESET) to be performed on the Output device D+2. Note: For a SET (ON) operation D+3 must equal 1, for a RESET (OFF) D+3 must equal 0.

The following points should be read while studying the example on the right of the page. Please note, all normal rules associated with high speed counters still apply.

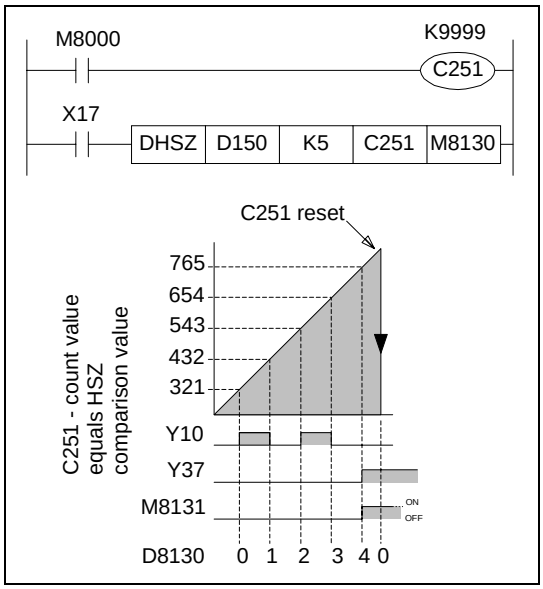
The data table is processed one 'record number' at a time, i.e only 1 record is ever active as the comparison data. The currently active record number is stored in data register D8130. As the comparison value for the active record is 'reached', the assigned 'Y' device is SET or RESET and the active 'Record number' is incremented by 1. Once all records in a data table have been processed, the current record pointer (D8130) is reset to 0 (the table is then ready to process again) and the operation complete flag M8131 is set ON.

Record number [D8130]	**Comparison value (lower/upper register)** [D, D+1]	**Selected 'Y' Output Device** [D+2]	**SET/RESET 'Y'Device** (1=SET, 0=RESET) [D+3]
0	[D150, D151] K321	[D152] H10 (Y10)	[D153] K1
1	[D154, D155] K432	[D156] H10 (Y10)	[D157] K0
2	[D158, D159] K543	[D160] H10 (Y10)	[D161] K1
3	[D162, D163] K765	[D164] H10 (Y10)	[D165] K0
4	[D166, D167] K765	[D168] H37 (Y37)	[D169] K1

If the high speed counter is reset (by program or hardware input), when it resumes counting and reaches the first record's comparison value, the M8131 flag will be reset. Both the status of M8131 and contents of D8130 are not editable by the user. If the DHSZ instruction is turned OFF then all associated flags are reset.

Care should be exercised when resetting the high speed counter or turning OFF the DHSZ instruct as all associated 'Y' output devices will remain in their last state, i.e. if an output was ON it will remain ON until independently reset by the user.

The data within inactive records can be changed during operation allowing data tables to be updated. Any change made is processed at the end of the current program scan. The HSZ instruction will continue to process only the active data record, i.e. it will not reset due to the updating of an inactive data record.



When the DHSZ instruction is initially activated it will not process a comparison until the following program scan as the CPU requires a slight time delay to initialize the comparison table.

Operation 3 - Combined HSZ and PLSY Operation: (Applicable units: $FX_{(2C)}$ and $FX_{2N(C)}$)
Operation 3 allows the HSZ and PLSY instructions to be used together as a control loop. This operation is selected when the destination device (D) is assigned special M coil M8132. This then allows devices (S_1, S_2) to be used to define a data table using (S_1) as the head address and (S_2) as the number of records in the table - maximum number of records is 128. Each record occupies 4 consecutive data registers (D through D_{+3}) proportioned in to two 32 bit data areas.

The first pair of data registers (D,D_{+1}) contain the comparison value for use with the high speed counter. The second pair of data registers (D_{+2},D_{+3}) contain a value (from 0 to 1000) which represents an output frequency in Hz. This value is loaded in to special data register D8132 when the comparison made by the DHSZ instruction gives a 'TRUE' output.

Record number [D8131]	Comparison value (lower/upper register) [D, D_{+1}]	Output Frequency For PLSY Instruction [D_{+2}, D_{+3}]
0	[D180, D181] K40	[D182, D183] K100
1	[D184, D185] K100	[D186, D187] K600
2	[D188, D189] K400	[D190, D191] K550
3	[D192, D193] K800	[D194, D195] K40
4	[D196, D197] K0	[D198, D199] K0

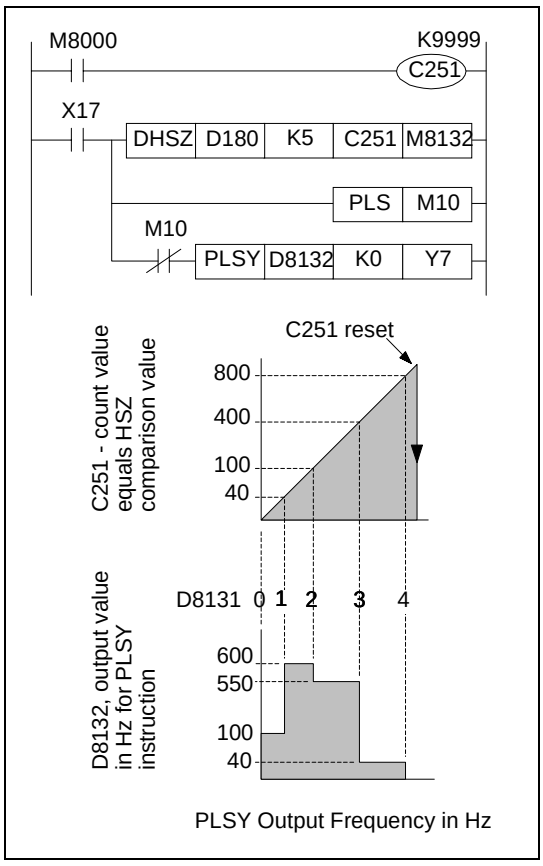
Special data register D8132 can be used as the source data for a PLSY (FNC57) output enabling the output to be varied with relative count data.

As with Operation 2 only one record in the data table is active at any one time. The current 'Record number' being processed is stored in data register D8131. To observe the current comparative value, data registers D8134 and D8135 should be monitored as a double word (32 bit) device.

Once the final entry in the data table has been processed, the operation complete flag M8133 is set ON and the record counter (D8131) cycles back to the first record.

It is recommended that if the high speed counter and PLSY operations form a closed loop that the last record entry in the data table is set to K0 for the comparison value and K0 for the PLSY output frequency. This will bring the controlled system to a stop and the 'Record number' counter will not be able to cycle back to the start of the data table until the associated high speed counter is reset by either pro-gram or hardware methods. This situation can be easily monitored by checking the paired data registers D8134 and D8135 for the '0' value.

It is recommended that the operation of the PLSY instruction is delayed for 1 scan to allow the DHSZ data table to be constructed on initial operation. A suggested program using a pulsed flag is shown in the example on this page.

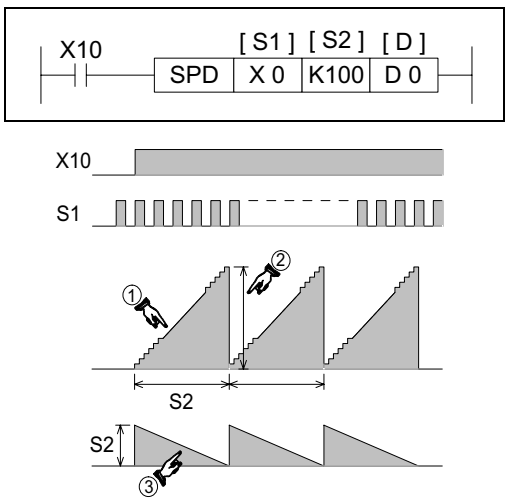


5.6.7 SPD (FNC 56)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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	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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The number of pulses received at S_1 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_2 in msec. The time remaining on the current 'timed count', is displayed in device D_{+2}. The number of counted pulses (of S_1) 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:

a) 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.

b) Because this is both a high speed and an interrupt process only inputs X0 to X5 may be used as the source device S_1. However, the specified device for S_1 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 S_1 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;

$$\text{Linear speed N (km/h)} = \frac{3600 \times (D)}{n \times S_2} \times 10^3$$

where n = the number of linear encoder divisions per kilometer.

$$\text{Radial speed N (rpm)} = \frac{60 \times (D)}{n \times S_2} \times 10^3$$

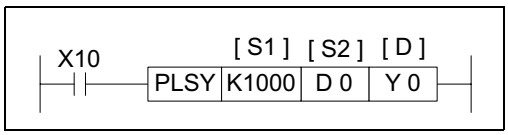
where n = the number of encoder pulses per revolution of the encoder disk.

5.6.8 PLSY (FNC 57)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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 Note: FX0(S)/FX0N users: Y000 only ☒. FX users: any YPPP. FX2N(C) users: Y000 or Y001 only ☒.	PLSY: 7 steps DPLSY: 13steps

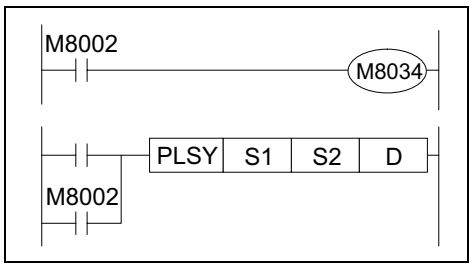
PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Operation Complete M8029
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



Operation:

A specified quantity of pulses S2is 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:



a) Users of the FX PLC can specify output frequencies (S1) of 1 to 1000Hz. Users of FX Version 2.2 or earlier will need to initialize the PLSY instruction. The program to the right can be used to achieve this. FX0/FX0N users may use frequencies of 10to 2000Hz. FX2N users may use frequencies of 2 to 20000Hz.

b) 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.

c) 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.

d) 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.

e) 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 instructions. In this case, the current instruction must be deactivated before changing to the new instance.

f) 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.

g) $FX_{(2C)}$ and $FX_{2N(C)}$ units can use the HSZ (FNC 55) instruction with the PLSY instruction when source device S_1 is set to D8132. Please see page 5-59 for more details.

h) FX units with CPU version 3.07 or greater and FX_{2C} units can monitor the number of pulses which have been output as a double word using devices D8136 and D8137.

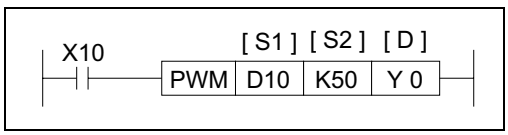
$FX_{2N(C)}$ units can also monitor the number of pulses output to Y0 using devices D8140 and D8141 and the number of output pulses output to Y1 using devices D8142 and D8143. The total number of pulses output can be monitored using D8136 and D8137.

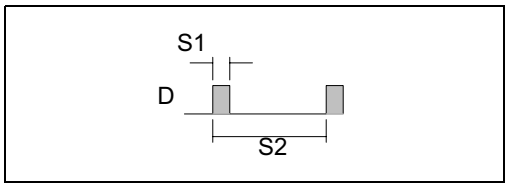
5.6.9 PWM (FNC 58)

$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$

Mnemonic	Function	Operands			Program steps
		S_1	S_2	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 Note: $FX_{0(S)}$/FX_{0N} users: Y001 only ⊠. FX users: any YPPP. $FX_{2N(C)}$ users: Y000 or Y001 only ⊠	PWM: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
$FX_{0(s)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$	$FX_{0(s)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$	$FX_{0(s)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$





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 (S_2).
And how long, also in time (msec), a single pulse will be active for (S_1).

Points to note:

a) Because this is a 16 bit instruction, the available time ranges for S_1 and S_2 are 1 to 32,767.

b) A calculation of the duty cycle is easily made by dividing S_1 by S_2. Hence S_1 cannot have a value greater than S_2 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.

c) The PWM instruction may only be used once in a users program.

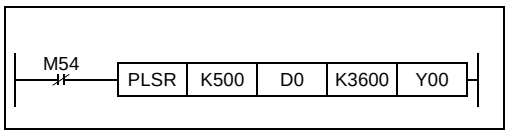
d) Because of the nature of the high speed output, transistor output units should be used with this instruction. Relay outputs will suffer from a greatly reduced life and will cause false outputs to occur due to the mechanical 'bounce' of the contacts. To ensure a 'clean' output signal when using transistor units, the load current should be 200mA or higher. It may be found that 'pull up' resistors will be required.

5.6.10 PLSR (FNC 59)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

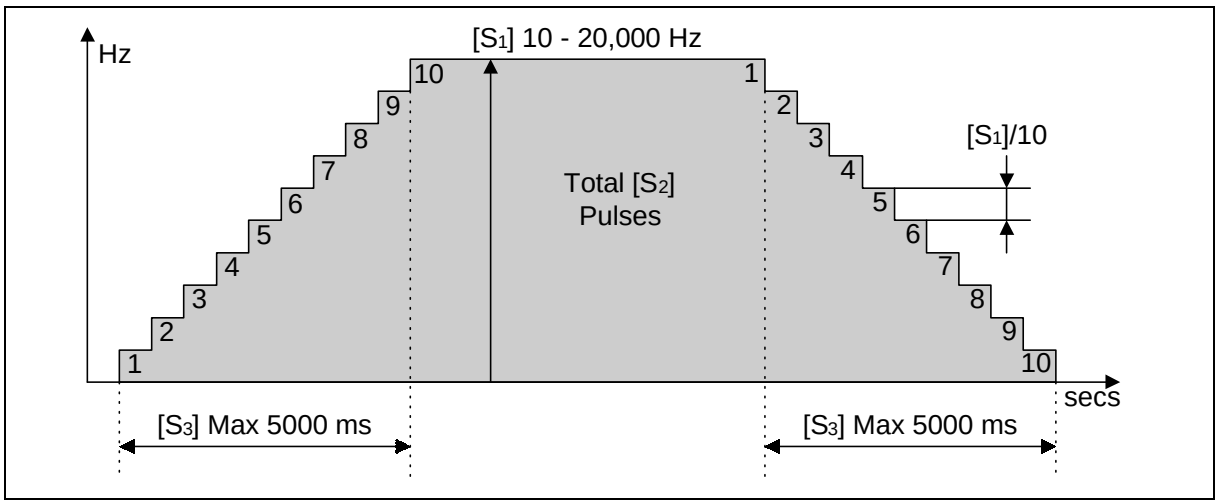
Mnemonic	Function	Operands				Program steps
		S1	S2	S3	D	
PLSR FNC 59 (Pulse ramp)	Outputs a specified number of pulses, ramping up to a set frequency and back down to stop	K, H, KnX, KnY, KnM, KnS, T, C, D, V, Z Note: S1 S2			Y FX2N users: Y000 or Y001 only.	PLSR: 9 steps DPLSR: 17 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Operation Complete M8029
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)		



Operation:

A specified quantity of pulses S2is 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:

a) FX2N users may use frequencies of 10 to 20,000Hz. 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.

b) 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.

c) The acceleration time must conform to the limitations described on the next page.

d) The output device is limited to Y0 or Y1 only and should be transistor type.

e) 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 differen instances of this instructions. In this case, the current instruction must be deactivated before changing to the new instance.

f) If the number of pulses is not enough to reach the maximum frequency then the frequency is automatically cut

g) 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.

h)

Acceleration time limitations

The acceleration time S_3 has a maximum limit of 5000 ms. However, the actual limits of $S_3$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.

$$S_3 \geq \frac{90000}{S_1} \times 5$$

2) The following formula gives the minimum value for S_3.

3) The following formula gives the maximum value for S_3.

$$S_3 \leq \frac{S_2}{S_1} \times 818$$

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 S_1.

- Possible output frequency is limited to 2 to 20,000 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.

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.7 Handy Instructions - Functions 60 to 69

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

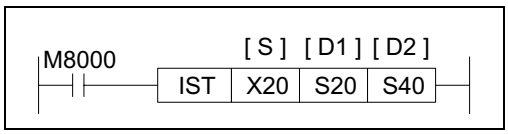
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.7.1 IST (FNC 60)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands			Program steps
		S1	S2	S3	
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: FX0 users S20 to S63 FX0N users S20 to S127 FX users S20 to S899 D1must be lower than D2		IST: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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:

a) 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.

b) The IST instruction may only be used **ONCE**.
It should be programmed close to the beginning of the program, before the controlled STL circuits.

c) The required operation mode is selected by driving the devices associated with operands
S+0through 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.

d) 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 'zero' position is sometimes also referred to as a home position, safe position, neutral position or a datum position.

e) 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.

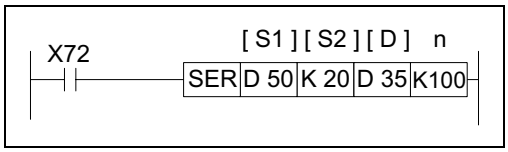
Note: Start, stop and zero inputs are often given by additional, manually operated push buttons.
Please note that the 'stop' input is only a program stop signal. It **cannot** be used as a replacement for an 'Emergency stop' push button. All safety, 'Emergency stop' devices should be hardwired systems which will effectively isolate the machine from operation and external power supplies. Please refer to local and national standards for applicable safety practices.

5.7.2 SER (FNC 61)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



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:

a) Normal rules of algebra are used to determine the largest and smallest values, i.e. -30 is smaller than 6 etc.

b) If no occurrence of the searched data can be found then destination devices D, D+1 and D+2will equal 0 (zero).

c) 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.

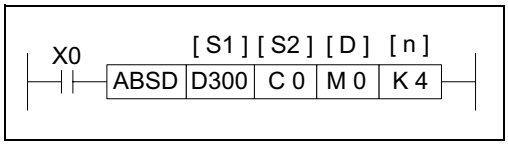
d) When multiple bit devices are used to store the result (regardless of 16 or 32 bit operation), only the specified size of group is written to for 5 consecutive occurrences, i.e. K1Y0 would occupy 20 bit devices from Y0 (K1 = 4 bit devices and there will be 5 groups for the 5 results). As the maximum data stack is 256 (0 to 255) entries long, the optimum group of bit devices required is K2, i.e. 8 bit devices.

5.7.3 ABSD (FNC 62)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands				Program steps
		S1	S2	D	n	
ABSD FNC 62 (Absolute drum sequencer)	Generates multiple output patterns in response to counter data	KnX, KnY, KnM, KnS, (in groups of 8) T, C, D	C	Y,M,S Note: n consec-utive devices are used	K,H ⊠ Note: n≤ 64	ABSD: 9 steps DABSD: 17 steps. see f).
		Note: High speed counters are not allowed				

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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:

a) 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. S1should always have an even device number.

b) 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.

c) The data table has a length equal to 2 × 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.

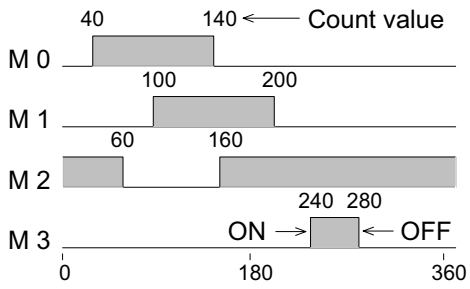
d) Values from 0 to 32,767 may be used in the data table.

e) The ABSD instruction may only be used **ONCE**.

f) FX CPU's ver 3.07 or greater and FX2C units have double word option on this instruction.

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

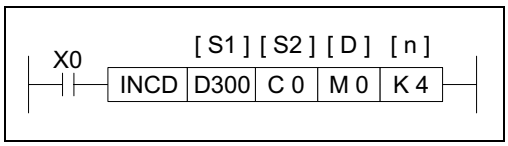


5.7.4 INCD (FNC 63)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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	KnX, KnY, KnM, KnS, (in groups of 8) T, C, D	C Uses 2 consecutive counters	Y, M, S Note: n consecutive devices are used	K,H ☒ Note: n≤ 64	INCD: 9 steps
		Note: High speed counters are not allowed				

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Operation Complete M8029
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



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:

a) This instruction uses a 'data table' which contains a single list of values which are to be selected and compared by two consecutive counters (S2and S2+1). The data table is identified as having a head address S1and consists of n data elements.

b) 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+1and resets its own current value to '0' (zero).
This new value of counter S2+1selects the new data element from the data table and counter S2now compares against the new data elements value.

c) 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+1hence, the cycle starts again with data element S1+0.

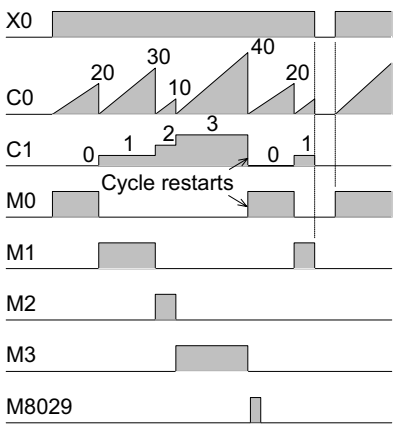
d) Values from 0 to 32,767 may be used in the data table.

e) 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

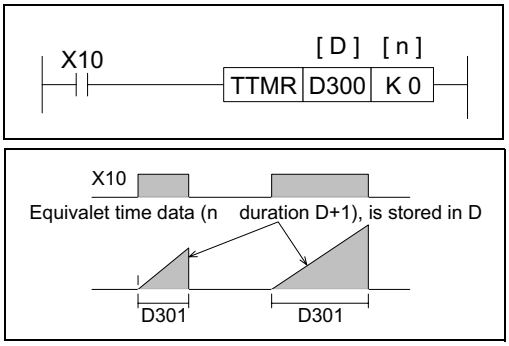


5.7.5 TTMR (FNC 64)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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) × 1 n= 1: (D) = (D+1) × 10 n= 2: (D) = (D+1) × 100	TTMR: 5 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

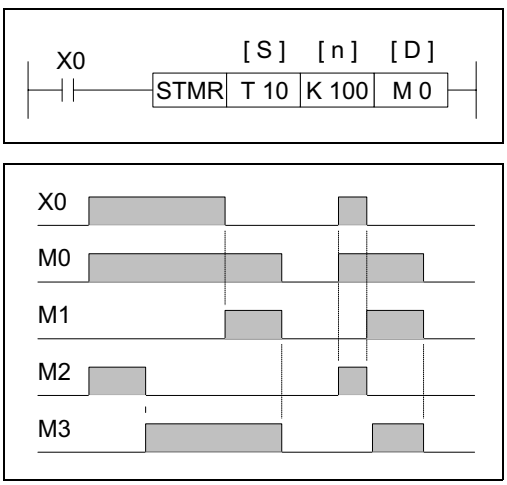
The duration of time that the TTMR instruction is energized, is measured and stored in device D+1 (as a count of 100ms periods).
The data value of D+1 (in secs), multiplied by the factor selected by the operand n, is moved in to register D. The contents of D could be used as the source data for an indirect timer setting or even as raw data for manipulation.
When the TTMR instruction is de-energized D+1 is automatically reset (D is unchanged).

5.7.6 STMR (FNC 65)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

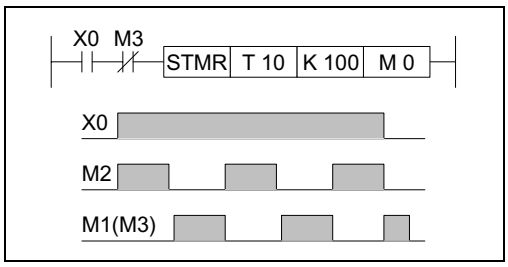
Mnemonic	Function	Operands			Program steps
		S	n	D	
STMR FNC 65 (Special timer)	Provides dedicated off-delay, one shot and flash timers	T Note: Timers 0 to 199 (100msec devices)	K, H ✉ Note: n= 1 to 32,767	Y, M, S Note:uses 4 consecutive devices D+0to D+3	STMR: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The designated timer Swill operate for the duration n with the operational effect being flagged by devices D+0to D+3. Device D+0is 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.

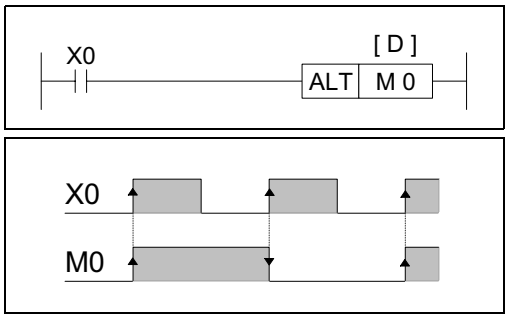


5.7.7 ALT (FNC 66)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

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

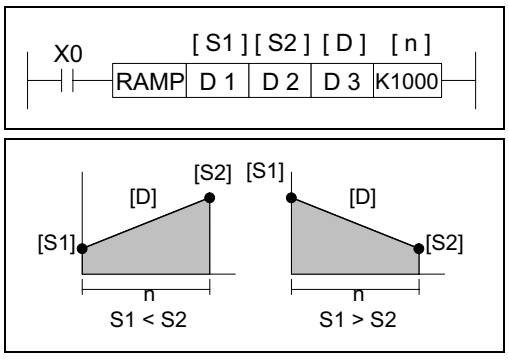
This means the status of each bit device will flip-flop between ON and OFF. This will occur on every program scan unless a pulse modifier or a program interlock is used.
The ALT instruction is ideal for switching between two modes of operation e.g. start and stop, on and off etc.

5.7.8 RAMP (FNC 67)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Operation Complete M8029
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)		

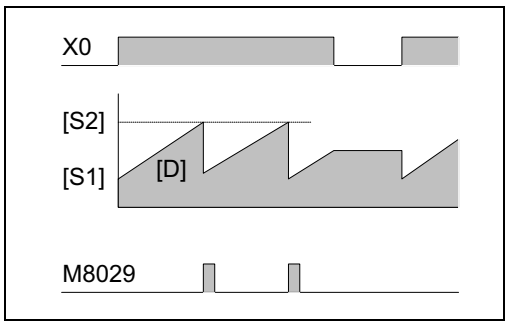


Operation:

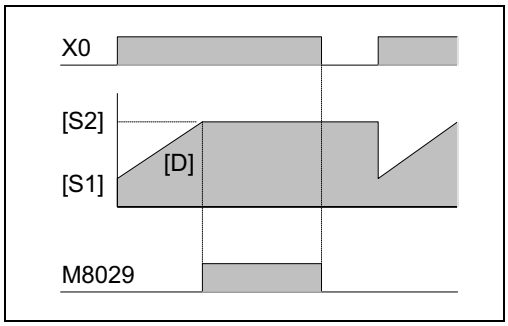
The RAMP instruction varies a current value (D) between the data limits set by the user (S1and S2). The 'journey' between these extreme limits takes n program scans. The current scan number is stored in device D+1. Once the current value of D equals the set value of S2the execution complete flag M8029 is set ON.
The RAMP instruction can vary both increasing and decreasing differences between S1and S2.

Points to note:

a) FX 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 S2the 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 re-driven 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 S1and 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 × D8039.

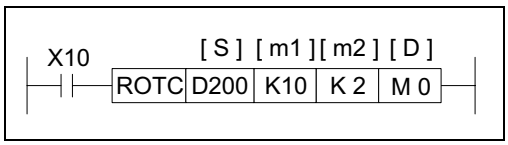
The RAMP instruction may also be used with special M flags M8193 and M8194 to mimic the operation of the SER (FNC 61) and RS (FNC 80) respectively when being programmed on older versions of programming peripherals. See page 1-5 for more details.

5.7.9 ROTC (FNC 68)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands				Program steps
		S	m1	m2	D	
ROTC FNC 68 (Rotary table control)	Controls a rotary tables movement is response to a requested destination/ position	D Note: uses 3 consecu-tive devices S+1≤ m1	K, H ⊠ Note: m1= 2 to 32,767 m1≥ m2	K, H ⊠ Note: m2= 0 to 32,767	Y, M, S Note: uses 8 consecu-tive devices	ROTC: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

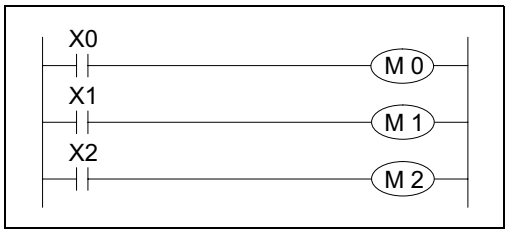
The ROTC instruction is used to aid the tracking and positional movement of the rotary table as it moves to a specified destination.

Points to note:

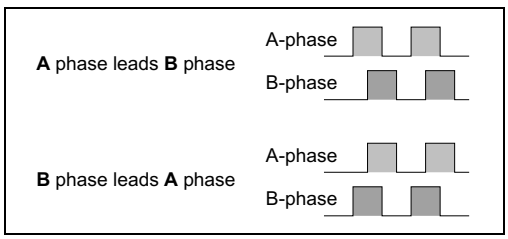
a) This instruction has many automatically defined devices. These are listed on the right of this page.

b) The ROTC instruction may only be used **ONCE**.

c) The ROTC instruction uses a built in 2-phase counter to detect both movement direction and distance travelled. Devices D+0and D+1 are used to input the phase pulses, while device D+2is 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 travelled 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

d) When the 'zero point' input (D+2) is received the contents of device S+0is 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

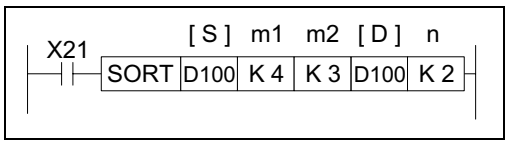
instruction:
S+1=3 $\times$ 50 = 150 (station 3's position in encoder pulses from the zero point)
S+2=7 $\times$ 50 = 350 (station 7's position in encoder pulses from the zero point)
m1= 400 (total number of encoder pulses per rev)
The rotary table is required approach the destination station at a slow speed starting from 1.5 stations before the destination. Therefor;
m2= 1.5 $\times$ 50 = 75 slow speed distance either side of the destination station (in encoder pulses)

5.7.10 SORT (FNC 69)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands					Program steps
		S	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 ⊠ Note: m1= 1 to 32 m2= 1 to 6		D ⊠	K, H D ⊠ Note: n = 1 to m2	SORT: 11 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Operation Complete M8029
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



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 nis 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:

a) When a sort occurs each record is sorted in to ascending order based on the data in the selected sort field n.

b) 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.

c) 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 m1times.

d) 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.

e) 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
RECORD (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

Table1st table sort when n= 2

		FIELD (m2)		
		1	2	3
RECORD (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

2nd table sort when n=1

		FIELD (m2)		
		1	2	3
RECORD (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

MEMO

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.8 External FX I/O Devices - Functions 70 to 79

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

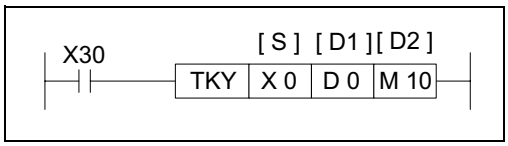
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.8.1 TKY (FNC 70)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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:

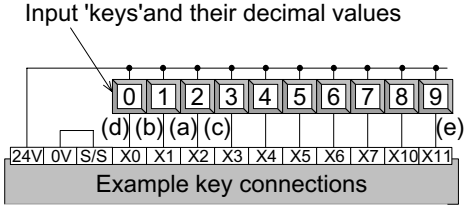
a) When a source device becomes active its associated destination (bit) device D2 also becomes active. This destination device will remain active until another source device is operated. Each source device maps directly to its own D2 device, i.e. S+0 maps to D2+0, S+7 maps to D2+7 etc. These in turn, map directly to decimal values which are then stored in the destination data devices specified by D1.

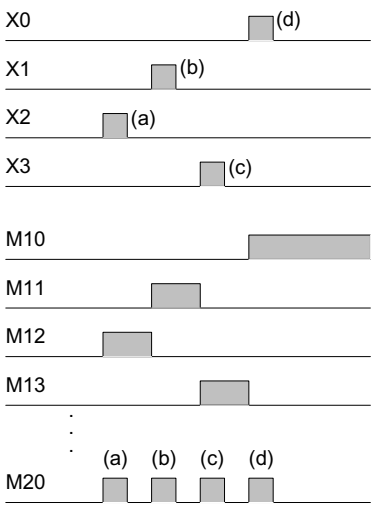
b) One source device may be active at any one time. The destination device D2+10 is used to signify that a key (one of the 10 source devices) has been pressed. D2+10 will remain active for as long as the key is held down. When the TKY instruction is active, every press of a key adds that digit to the stored number in D1. When the TKY is OFF, all of the D2 devices are reset, but the data value in D1 remains intact.

c) 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 D1and 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.



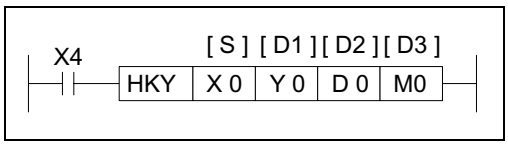


5.8.2 HKY (FNC 71)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands				Program steps
		S	D1	D2	D3	
HKY FNC 71 (Hexadeci-mal key input)	Multiplexes inputs and outputs to create a numeric keyboard with 6 function keys	X, Note: uses 4 consecu-tive devices	Y, Note: uses 4 consecu-tive devices	T, C, D, V, Z Note: uses 2 consecutive devices for 32 bit operation	Y, M, S Note: uses 8 consecu-tive devices	HKY: 9 steps DHKY: 17 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Operation Complete M8029
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)		



Operation 1 - Standard:

This instruction creates a multiplex of 4 outputs (D_1) 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.

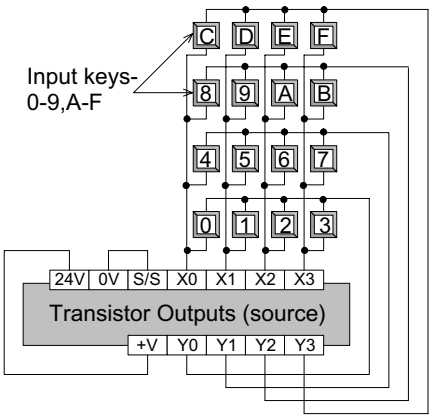
Points to note:

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 D_2. Activation of any of these keys causes bit device D_{3+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 D_{3+0} to D_{3+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 D_{3+6} to turn ON for the duration of that key press.

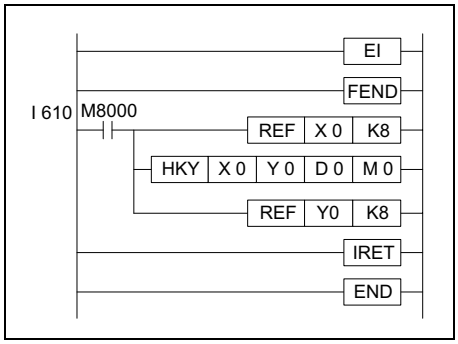
c) In all key entry cases, when two or more keys are pressed, only the key activated first is effective. When the pressing of a key is sensed the M8029 (execution complete flag) is turned ON. When the HKY instruction is OFF, all D_3 devices are reset but data value D_2 remains intact.

d) When the HKY instruction is used with 16 bit operation, D_2 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 D_2 and D_{2+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.



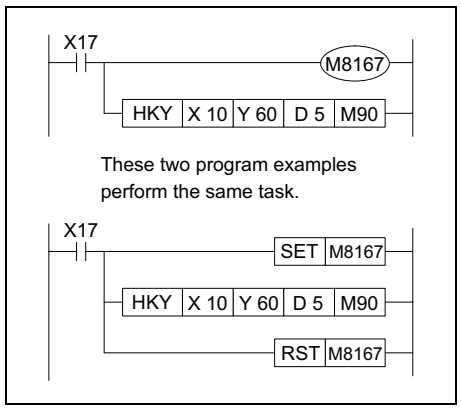
e) The HKY instruction may only be used **ONCE**.

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:

(Applicable units: FX(2C) and FX2N)
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.

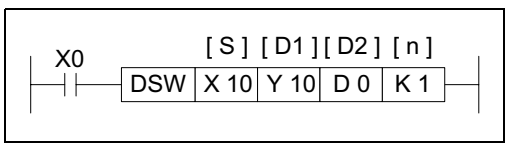


5.8.3 DSW (FNC 72)

FX0(S) FX0N FX FX(2C) FX2N(C)

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 consecu-tive 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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Operation Complete M8029
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)		



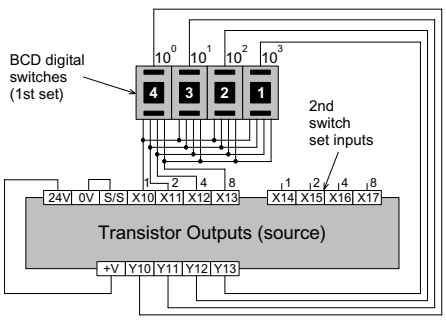
Operation:

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:

a) 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.

Continued on next page...



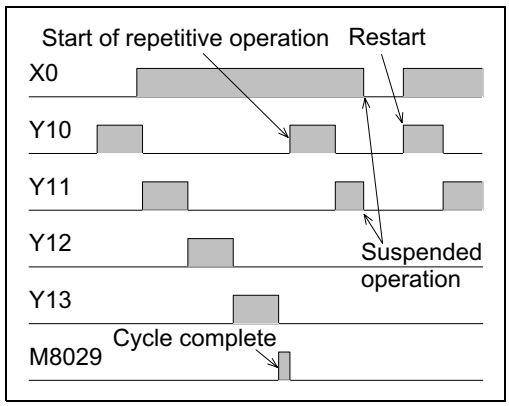
b) When n= 2, two sets of switches are read. This configuration requires 8 consecutive inputs taken from the head address specified in operand S. The data from the first set of switches, i.e. those using the first 4 inputs, is read into data device D_2. The data from the second set of switches (again 4 digits) is read into data device D_{2+1}.

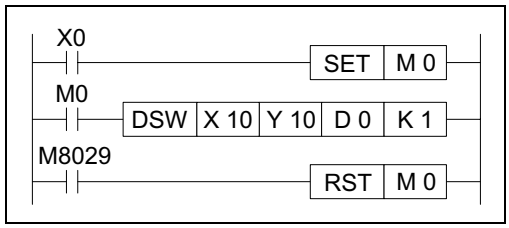
c) The outputs used for multiplexing (D_1) are cycled for as long as the DSW instruction is driven. After the completion of one reading, the execution complete flag M8029 is set. The number of outputs used does **not** depend on the number of switches n.

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.

f) The DSW instruction may be used **ONCE** on FX controllers with CPU versions lower than 3.07. FX units with CPU ver 3.07 or greater and all FX_{2C} units can operate a maximum of **TWO** DSW instructions.



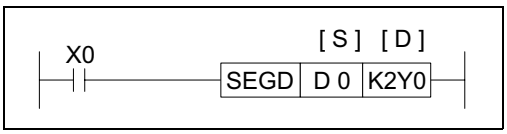


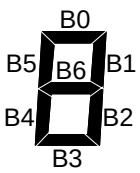
5.8.4 SEGD (FNC 73)

$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Zero M8020
$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$	$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$	$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$		





It can be seen that B7 is NOT used. Hence B7 of the destination device D will always be OFF,

Operation:

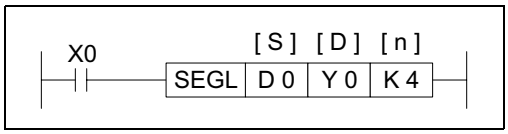
A single hexadecimal digit (0 to 9, A to F) occupying the lower 4 bits of source device S is decoded into a data format used to drive a seven segment display. A representation of the hex digit is then displayed. The decoded data is stored in the lower 8 bits of destination device D. The upper 8 bits 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.

5.8.5 SEGL (FNC 74)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Operation Complete M8029
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



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.

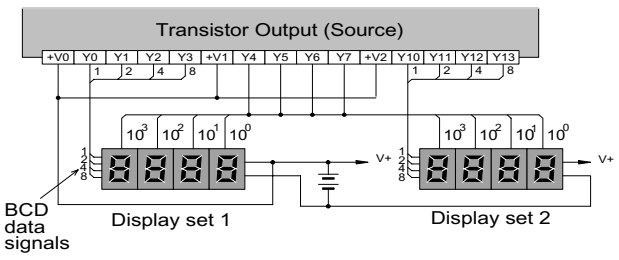
Points to note:

a) 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.

Note: A single set of strobe signals are always used regardless of the number of display sets.



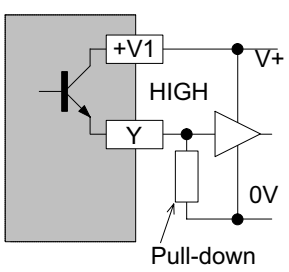
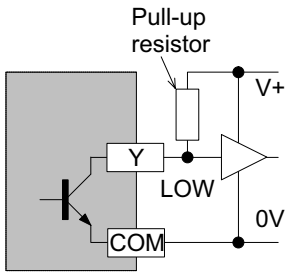
In this example it has been assumed that the seven segment displays accept data HIGH inputs and latch when a HIGH signal is received

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 SEGL instruction may be used **ONCE** on FX controllers with CPU versions lower than 3.07. FX units with CPU ver 3.07 or greater and all FX2C units 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
PLC Logic		Source output 	Sink output 
		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'
Seven segment Display logic	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:

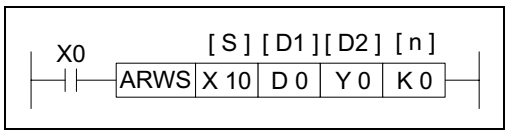
PLC Logic	Seven segment display logic		n	
	Data Logic	Strobe logic	1 display set	2 display sets
Positive (Source)	Positive (High)	Positive (High)	0	4
Negative (Sink)	Negative (Low)	Negative (Low)		
Positive (Source)	Positive (High)	Negative (Low)	1	5
Negative (Sink)	Negative (Low)	Positive (High)		
Positive (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)		

5.8.6 ARWS (FNC 75)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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 consecu-tive devices	T, C, D, V, Z Note: data is stored in a decimal format	Y Note: uses 8 consecu-tive devices	K, H ⊠ Note: n= 0 to 3,	ARWS: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

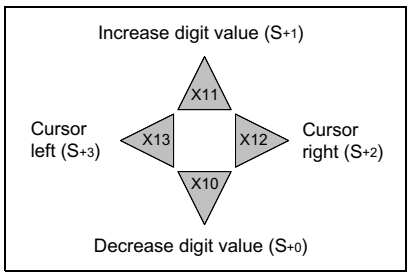


Operation:

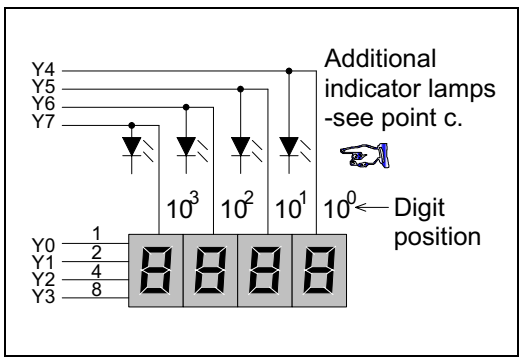
This instruction displays the contents of a single data device D1on a set of 4 digit, seven segment displays. The data within D1is actually in a standard decimal format but is automatically converted to BCD for display on the seven segment units. Each digit of the displayed number can be selected and edited. The editing procedure directly changes the value of the device specified as D1.

Points to note:

a) The data stored in destination device D1can have a value from the range 0 to 9,999 (decimal), i.e. 4 digit data. Each digits 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.



b) 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.



c) 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.

d) The parameter n has the same function as parameter n of the SEGL instruction - please seepage5-86, 'Selectingthecorrectvalueforoperand n'. Note: as the ARWS instruction only controls one set of displays only values of 0 to 3 are valid for n.

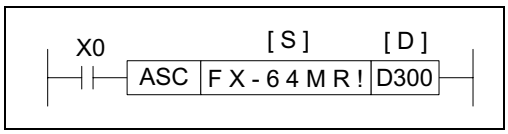
e) The ARWS instruction can be used **ONCE**. This instruction should only be used on transistor output PLC's.

5.8.7 ASC (FNC 76)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		S	D1	
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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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	58 (X)	46 (F)
D301	36 (6)	2D (-)
D302	4D (M)	34 (4)
D303	21 (!)	52 (R)

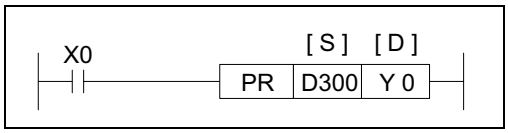
Note: ASCII Char **cannot** be entered from a hand held programmer.

5.8.8 PR (FNC 77)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		S	D1	
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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Operation Complete M8029
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)		



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).

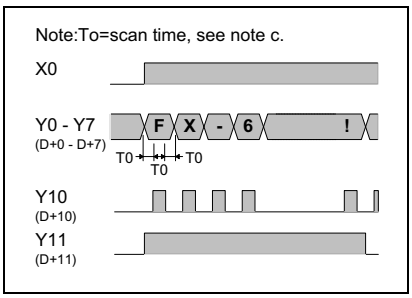
Points to note:

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.

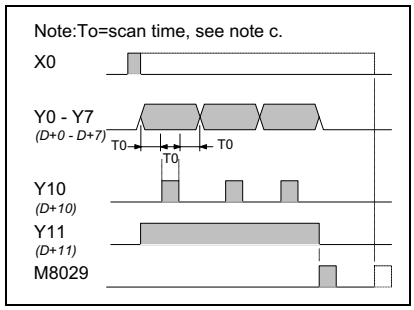
b) The PR instruction may be used **ONCE** on FX units fitted with CPU versions earlier than 3.07. FX units with CPU ver3.07 or later and all FX2C units can operate **TWO** PR instructions. This instruction should only be used on transistor output PLC's. The PR instruction will not automatically repeat its operation unless the drive input has been turned OFF and ON again.

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.

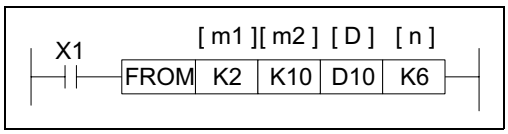


5.8.9 FROM (FNC 78)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands				Program steps
		m1	m2	D	n	
FROM FNC 78 (FROM)	Read data from the buffer memories of attached special function blocks	K, H ⊠ Note: m1= 0 to 7	K, H ⊠ Note: m2 = FX(2C) 0 to 31, FX2N 0 to 32767	KnY, KnM, KnS, T, C, D, V, Z	K, H ⊠ Note: 16 bit op: n= 1 to 32 32 bit op: n= 1 to 16	FROM, FROMP: 9 steps DFROM, DFROMP: 17 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

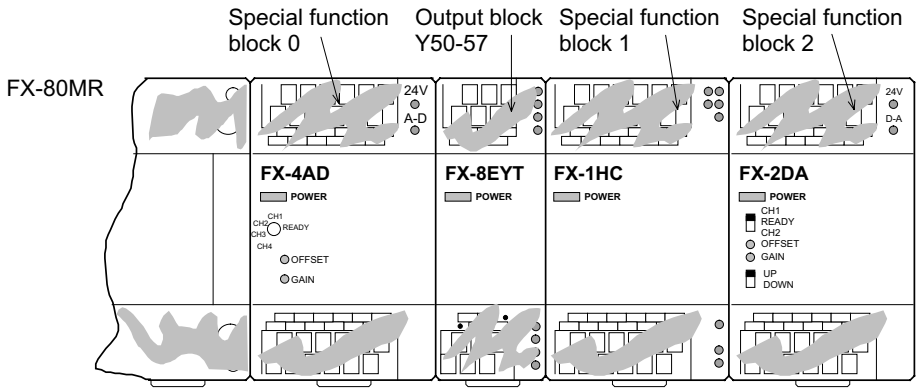


Operation:

The FROM instruction reads n words of data starting from the buffer memory address m2 of the special function block with the logical block position specified as m1, The read data is stored in the PLC at head address D for n word devices.

Points to note:

a) All special function blocks which are addressable with the FROM/TO instructions are connected to the extension bus on the right hand side of the PLC. Each special function block can be inserted at any point within the chain of extended units (as long as the system configuration rules are not broken). Each special function block is consecutively addressed from 0 to 7 beginning with the one closest to the base unit.



b) Each special function unit has different buffer memory registers. These often have a dedicated use for each individual unit. Before any reading or writing of data is undertaken ensure that the correct buffer memory allocations for the unit used are known.
m2: This defines the head address of the (special function blocks) buffer memories being accessed. m2 may have a value from the range 0 to 31.
n: This identifies the number of words which are to be transferred between the special function block and the PLC base unit. n may have a value of 1 to 31 for 16 bit operation but a range of 1 to 16 is available for 32 bit operation.

c) The destination head address for the data read FROM the special function block is specified under the D operand; and will occupy n further devices.

d) This instruction will only operate when the drive input is energized.

e) Users of FX PLC's have the option of allowing interrupts to occur immediately, i.e. during the operation of the FROM/TO instructions or to wait until the completion of the current FROM/TO instruction. This is achieved by controlling the special auxiliary flag M8028. The following table identifies certain points associated with this control and operation.

Interruption Disabled	Interruption Enabled
M8028 = OFF	M8028 = ON
Jumps called by interrupt operation are delayed until the completion of the data transfer of the FROM/TO instruction	Jumps called by interrupt operation occur immediately
A small delay of (800m +200) µsec can be expected in the worst case. Note: m = the number of 32 bit words	Data transfer will resume upon return from the interrupt program. This may not be desirable if a FROM/TO instruction has been programmed within the called interrupt
Ensures that FROM/TO instructions included in an interrupt program will not interact with others elsewhere	M8028 should only be used when a very short delay is required in applications where timing and accuracy's are important

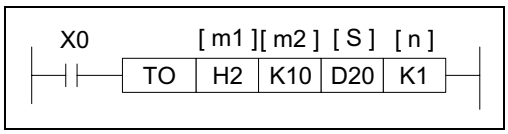
Users of FX0N have no option for interruption of the FROM/TO instructions and hence always operate in a mode equivalent to having M8028 switched OFF.

5.8.10 TO (FNC 77)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands				Program steps
		m1	m2	S	n	
TO FNC 79 (TO)	Writes data to the buffer memories of attached special function blocks	K, H ⊠ Note: m1= 0 to 7	K, H ⊠ Note: m2 =FX(2C) 0 to 31, FX2N 0 to 32767	K,H, KnX, KnY, KnM, KnS, T, C, D, V, Z	K, H ⊠ Note: 16 bit op: n= 1 to 32 32 bit op: n= 1 to 16	TO, TOP: 9 steps DTO, DTOP: 17 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The TO instruction writes n words of data to the head buffer memory address m2 of the special function block with the logical block position specified in m1. The written data is taken from the PLC's head address S for n word devices.

Points to note:

All points are the same as the FROM instruction (see previous page) except point c) which is replaced by the following:

a) The source head address for the data written TO the special function block is specified under the S operand.

MEMO

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.9 External FX Serial Devices - Functions 80 to 89

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

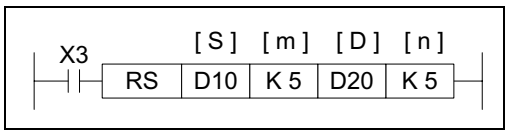
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.9.1 RS (FNC 80)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands				Program steps
		S	m	D	n	
RS FNC 80 (Serial Communications instruction)	Used to control serial communications from/to the programmable controller	D (including file registers)	K, H, D ☒ m = 1 to 256, FX2N 1 to 4096.	D	K, H, D ☒ m = 1 to 256, FX2N 1 to 4096	RS: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)		Transmit Delay M8121 Trans Request M8122 Data Received M8123



Operation:

This instruction performs the direct control of communications over FX and FX0N communication adapters which connect to the left hand port of the Main Processing Unit, i.e. FX0N-232ADP, FX-232ADP etc.

Points to note:

a) This instruction has many automatically defined devices. These are listed in the boxed column to the right of this page.

b) 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:
1) Transmit buffer - before transmission occurs, i.e. before M8122 is set ON
2) Receive buffer - after a message has been received and before M8123 is reset.

c) Data cannot be sent while a message is being received, the transmission will be delayed - see M8121.

d) More than one RS instruction can be programmed but only one may be active at any one time.

Assigned devices

Data devices:

D8120 - Contains the configuration parameters for communication, i.e. Baud rate,Stop bits etc. Full details over the page

D8122 - Contains the current count of the number of remaining bytes to be sent in the currently transmitting message.

D8123 - Contains the current count of the number of received bytes in the 'incoming' message.

D8124 - Contains the ASCII code of the character used to signify a message header - default is 'STX', 02 HEX.

D8125 - Contains the ASCII code of the character used to signify a message terminator -default is 'ETX', 03 HEX.

Operational flags:

M8121 - This flag is ON to indicate a transmission is being delayed until the current receive operation is completed.

M8122 - This flag is used to trigger the transmission of data when it is set ON.

M8123 - This flag is used to identify (when ON) that a complete message has been received.

M8124 - Carrier detect flag. This flag is for use with FX and FX2C Main Processing Units. It is typically useful in modem communications

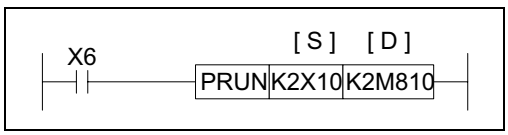
M8161 - 8 or 16 bit operation mode ON = 8 bit mode where only the lower 8 bits in each source or destination device are used, i.e. only one ASCII character is stored in one data register OFF = 16bit mode where all of the available source/ destination register is used, i.e. two ASCII characters are stored in each data register.

5.9.2 RUN (FNC 81)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		S	D	
PRUN FNC 81 (Parallel run)	Used to control the FX parallel link adapters: FX2-40AW/AP	KnX, KnM	KnY, KnY	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.		

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

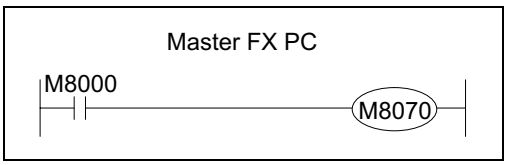


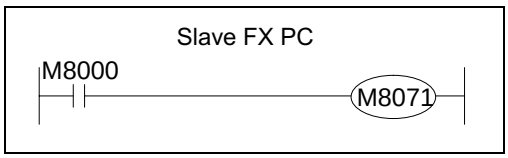
Operation:

This instruction is used with the FX parallel link adapters. It 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 FX PLC's. The programs shown opposite should be inserted into the appropriate FX PLC's programs.





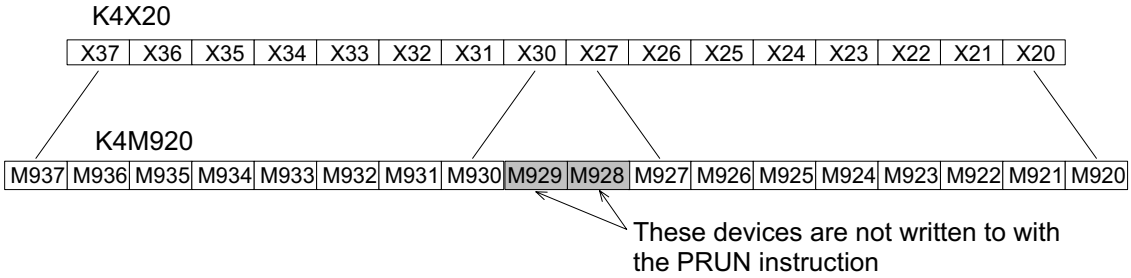
Once the station flags have been set, they can only be cleared by either forcibly resetting them when the FX PLC is in STOP mode or turning the power OFF and ON again.

b) During automatic communications the following data is 'swapped' between the Master and Slave PLC's.

Master station		Communication direction	Slave Station	
	Bit Data		Bit Data	
M8070 = ON	M800 to M899 (100 points)	→	M800 to M899 (100 points)	M8071 = ON
	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)	

Continued...

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.



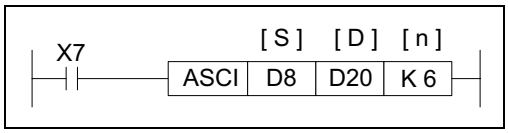
d) For more information please see page 9-6.

5.9.3 ASCI (FNC 82)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands			Program steps
		S	D	n	
ASCI 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 ✉	ASCI, ASCIP: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



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) AB**CD**H,**EF26**H. Taking the first n hexadecimal characters (digits) from the right (in this case n= 6) and converting them to ASCI 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.

Source (S)		Data
D9	b12-15	A
	b8-11	B
	b4-7	**C**
	b0-3	**D**
D8	b12-15	**E**
	b8-11	**F**
	b4-7	**2**
	b0-3	**6**

Destination (D)		ASCII Code HEX	ASCII Code DEC	Symbol
D20	b8-15	43	67	'C'
	b0-7	44	68	'D'
D21	b8-15	45	69	'E'
	b0-7	46	70	'F'
D22	b8-15	32	50	'2'
	b0-7	36	54	'6'

The ASCI instruction can be used with the M8161, 8 bit/16bit mode flag. The effect of this flag is exactly the same as that detailed on page 10-20. The example to the right 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).

ASCII Character Codes

The table below identifies the usable hexadecimal digits and their associated ASCII codes

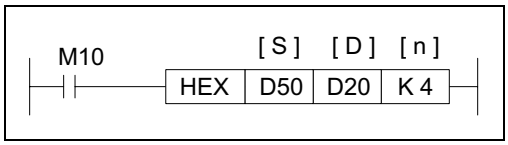
HEX Character		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ASCII Code	HEX	30	31	32	33	34	35	36	37	38	39	41	42	43	44	45	46
	DEC	48	49	50	51	52	53	54	55	56	57	65	66	67	68	69	70
Character Symbol		'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'A'	'B'	'C'	'D'	'E'	'F'

5.9.4 HEX (FNC 83)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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 f r om the destination head address.
If M8161 had been OFF, then the contents of D20 would read CAB1H.

Source (S)		ASCII Code HEX	ASCII Code DEC	Symbol
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:

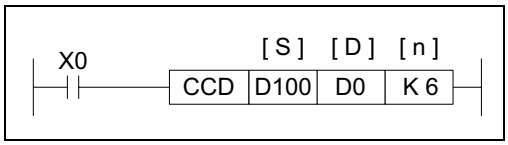
If an attempt is made to access an ASCII Code (HEX or Decimal) which falls outside of the ranges specified in the table on previous page, the instruction is not executed. Error 8067 is flagged in data register D8004 and error 6706 is identified in D8067. Care should be taken when using the M8161 flag, 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.

5.9.5 CCD (FNC 84)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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:

a) The SUM of the data stack is stored at destination D while the Parity for the data stack is stored at D_{+1}.

b) 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).

c) This instruction can be used with the 8 bit/ 16 bit mode flag M8161. The following results will occur under these circumstances. See page 10-20 for more details about M8161.

M8161=OFF									
Sourse (S)		Bit patterm							
D100 H	FF	1	1	1	1	1	1	1	1
D100 L	FF	1	1	1	1	1	1	1	1
D101 H	FF	1	1	1	1	1	1	1	1
D101 L	00	0	0	0	0	0	0	0	0
D102 H	F0	1	1	1	1	0	0	0	0
D102 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									
Sourse (S)		Bit patterm							
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***							

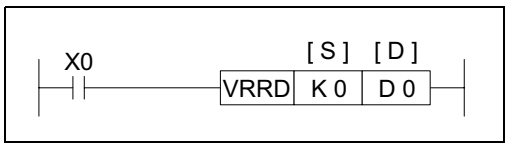
It should be noted that when M8161 is OFF 'n' represents the number of consecutive bytes checked by the CCD instruction. When M8161 is ON only the lower bytes of 'n' consecutive words are used.
The 'SUM' is quite simply a summation of the total quantity of data in the data stack. The Parity is checked vertically through the data stack as shown by the shaded areas.

5.9.6 VRRD (FNC 85)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		S	D	
VRRD FNC 85 (Volume read)	Reads an analog value from 1 of 8 volume inputs on the FX-8AV	K, H Note: S= 0 to 7 corresponding to the 8 available volumes on the FX-8AV	KnY, KnM, KnS T, C, D, V, Z	VRRD, VRRDP: 5 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The identified volume (S) on the FX-8AV is read as an analog input. The analog data is in an 8 bit format, i.e. values from 0 to 255 are readable. The read data is stored at the destination device identified under operand D.

Note:

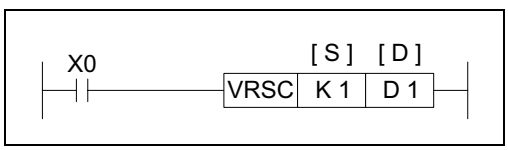
The FX-8AV volume 'inputs' are able to be read in two formats, a) as an analog value and b) as an 11 (0 to 10) position rotary switch. The second use is described in the VRSC instruction (FNC 86).

5.9.7 VRSD (FNC 86)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		S	D	
VRSC FNC 86 (Volume scale)	Reads the set position value, 0 to 10, from volume inputs on the FX-8AV	K, H Note: S= 0 to 7 corresponding to the 8 available volumes on the FX-8AV	KnY, KnM, KnS T, C, D, V, Z	VRSC, VRSCP: 5 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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.

Note:

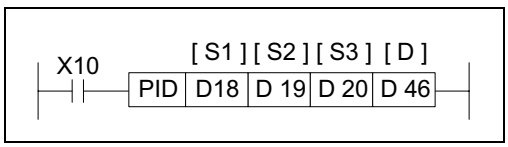
The FX-8AV volume 'inputs' are able to be read in two formats, a) as a 11 (0 to 10) position rotary switch and b) as an analog value. The second use is described in the VRRD instruction (FNC 85).

5.9.8 PID (FNC 88)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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⊠ Note: S1 and S2 use a single data register		D⊠ Note: S3 uses 25 consecutive data registers	D⊠ Note: D uses a single data register	PID: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

This instruction takes a current value (S_2) and compares it to a predefined set value (S_1). 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 S_3+0 through S_3+24.

Points to note:

a) Every PID application is different. There will be a certain amount of “trial and error” necessary to set the variables at optimal levels.

b) The PID instruction is **only** available on FX and FX2C Main Processing Units fitted with **CPU versions 3.11 or greater**.

c) On FX2N MPUs a Pre-tuning feature is available that can quickly provide initial values for the PID process. Refer to page 10-28 for more details.

d) 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.

e) 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 (S_1) 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.

f) 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.

g) 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 $PV_{nf} > SV$

$$\Delta MV = K_P\left\{(EV_n - EV_{(n-1)}) + \frac{T_S}{T_I}EV_n + D_n\right\}$$

$$EV_n = PV_{nf} - SV$$

$$D_n = \frac{T_D}{T_S + K_D \cdot T_D}(-2PV_{nf-1} + PV_{nf} + PV_{nf-2}) + \frac{K_D \cdot T_D}{T_S + K_D \cdot T_D} \cdot D_{n-1}$$

$$MV_n = \sum \Delta MV$$

Reverse $SV > PV_{nf}$

$$\Delta MV = K_P\left\{(EV_n - EV_{n-1}) + \frac{T_S}{T_I}EV_n + D_n\right\}$$

$$EV_n = SV - PV_{nf}$$

$$D_n = \frac{T_D}{T_S + K_D \cdot T_D}(2PV_{nf-1} - PV_{nf} - PV_{nf-2}) + \frac{K_D \cdot T_D}{T_S + K_D \cdot T_D} \cdot D_{n-1}$$

$$MV_n = \sum \Delta MV\Delta$$

$PV_{nf} = PV_n + \alpha(PV_{nf-1} - PV_n)$

EV_n = the current Error Value
EV_{n-1} = the previous Error Value
SV = the Set Point Value (S_1)
PV_n = the current Process Value (S_2)
PV_{nf} = the calculated Process Value
PV_{nf-1} = the previous Process Value
PV_{nf-2} = the second previous Process Value
ΔMV = the change in the Output Manipulation Values
MV_n = the current Output Manipulation Value (D)
D_n = the Derivative Value
D_{n-1} = the previous Derivative Value
K_P = the Proportion Constant
α = the Input Filter
T_S = the Sampling Time
T_I = the Integral Time Constant
T_D = the Time Derivative Constant
K_D = the Derivative Filter Constant

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 (S_3+1, b0)

The Forward operation is the condition where the Process Value, PV_{nf}, 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 PV_{nf}.

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, ΔMV will always have a value. Ideally, a system that is 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 S_3+0 through S_3+6 must be set by the user.

Parameter S3 + P	Parameter name/function	Description		Setting range
S3+0	Sampling time **TS**	The time interval set between the reading the current Process Value of the system (PV_{nf})		1 to 32767 msec
S3+1	Action - reaction direction and alarm control	b0	Forward operation(0), Reverse operation (1)	Not applicable
		b1	Process Value (PV_{nf}) alarm enable, OFF(0)/ ON(1)	
		b2	Output Value (MV) alarm enable, OFF(0)/ON(1)	
		b3 - 15	Reserved	
S3+2	Input filter α	Alters the effect of the input filter.		0 to 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 (PV_{nf}). This is the **P** part of the PID loop.		1 to 32767%
S3+4	Integral time constant **TI**	This is the **I** 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 **P** part of the loop. Selecting 0 (zero) for this parameter disables the **I** effect.		(0 to 32767) x 100 msec
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 (PV_{nf})		1 to 100%
S3+6	Derivative time constant **TD**	This is the **D** 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 **P** part of the loop. Selecting 0 (zero) for this parameter disables the **D** effect.		(0 to 32767) x 10 msec
S3+7 to 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 to 32767
S3+21	Process Value, minimum value		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		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 **(Read Only)**	b0	High limit exceeded in Process Value (PVnf)	Not applicable
		b1	Below low limit for the Process Value (PVnf)	
		b2	Excessive positive change in Output Value (MV)	
		b3	Excessive negative change in Output Value (MV)	
		b4 - 15	Reserved	

See **Initial values for PID loops** for basic guidance on initial PID values; page 5-114.
See page 10-24 for additional parameters available with FX2N MPUs.

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
	S_3 +3 (**K_P**)	S_3+ 4 (**T_I**)	S_3 + 6 (**T_D**)	
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_n, and the Previous Error Value, EV_{n-1}. 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_{nf} 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 T_I actually effects how fast the total integral correction is applied. The smaller T_I is, the bigger effect the integral will have.

Note: The T_I value is set in data register S_3+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_D, T_S, 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_D: If the value of T_D 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_D**, the derivative gain. The action of **K_D** 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_D value is set in Data register S_3+6. Setting zero for this variable disables the Derivative effect.

Effective use of the input filter α S_3+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 ± 32767.
Recommended initial settings:

T_S = Should be equal to the total program scan time or a multiple of that scan time, i.e. 2 times, 5 times, etc.
α = 50%
K_P = 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_I = This should ideally be 4 to 10 times greater than the T_D time
K_D = 50%
T_D = 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_D will be set to a quick or small time, this should however never be less than T_S. A slower reacting system will require the T_D duration to be longer. A beginning value can be T_D twice the value of T_S.

Care should be taken when adjusting PID variables to ensure the safety of the operator and avoid damage to the equipment.

On FX2N MPUs pre-tuning feature is available that can quickly provide initial values for the PID process. Refer to page 10-28 for more details.

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_P 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 S_3+0 to S_3+6. The PID instruction will be activated when M4 is On.

From the PID instruction at the bottom of the ladder, S_1 = D200; S_2 = D201; S_3 = D500; and D or MV = D525.

D500: Ts = 500 ms

D501: Forward Operation, Alarms Not Enabled

D502: Input Filter = 50%

D503: K_P = 75%

D504: T_I = 4000 ms

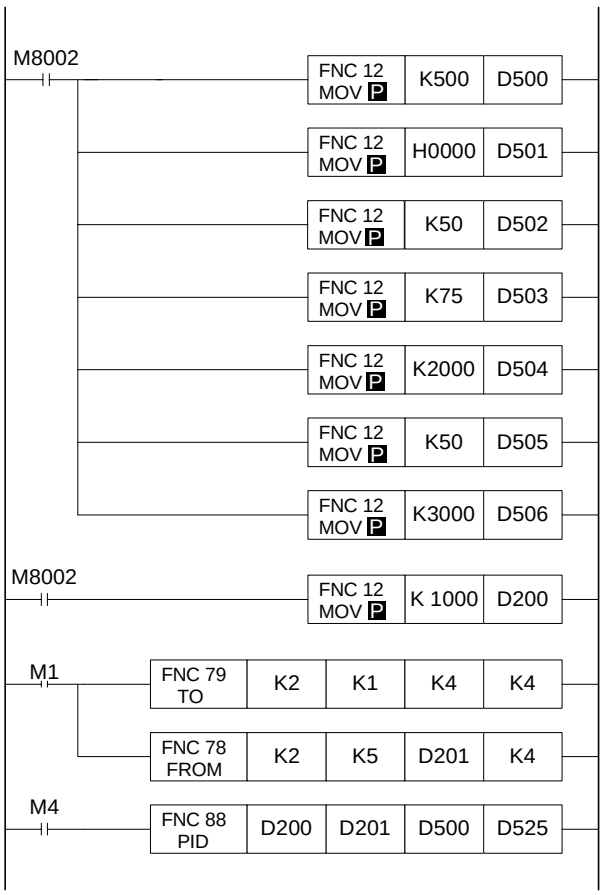
D505: K_D = 50%

D506: T_D = 1000 ms

D200: Set Point = 1000

D201: PV_{nf} (an analog input value)

Begin the PID instruction
D525: PID Output Value



MEMO

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.10 External F2 Units - Functions 90 to 99

Contents:

Please note: All of the instructions in this section reference chapter, Assigning System Devices for further information.

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_1, S_3 or for lists/tabled devices D_{3+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.

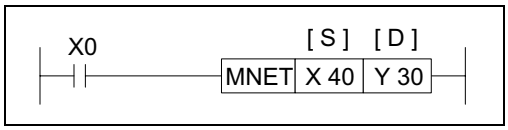
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.10.1 MNET (FNC 90)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		S	D	
MNET FNC 90 (F-16NT/NP Melsec net mini)	Used to control the F series net mini module - use with an FX2-24EI	X ⊠ Note: uses 8 consecutive devices	Y ⊠ Note: uses 8 consecutive devices	MNET, MNETP: 5 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The MNET instruction is used for communicating bit status signals between an FX PLC and an F-16NP/NT Melsec Net Mini interface.

The head address I/O numbers for both S and D are determined by the position of the FX2-24EI (connected to the F-16NT/NP) within the FX PLC's expansion chain. The devices specified for S and D can be used directly within the FX users program.

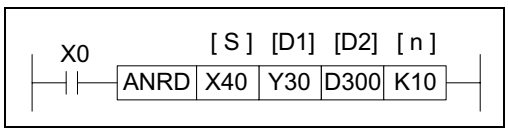
For more information please see page 9-3.

5.10.2 ANRD (FNC 91)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands				Program steps
		S1	D2	D2	n	
ANRD FNC 91 (F2-6A Analog read)	Used to read the F series analog module - use with an FX2-24EI	X ⊠ Note: uses 8 consecutive devices	Y ⊠ Note: uses 8 consecutive devices	KnY, KnM, KnS, T, C, D, V, Z	K, H ⊠ Note: n= 10 to 13	ANRD, ANRDP: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The ANRD instruction is used to read input channel n of an F2-6A analog module. The read analog value is stored at destination device D2.

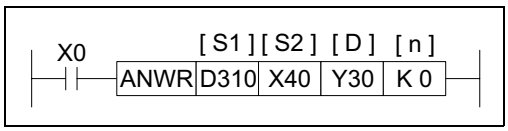
The head address I/O numbers for both Sand D1 are determined by the position of the FX2-24EI (connected to the F2-6A) within the FX PLC's expansion chain. The analog data stored at the destination device D2 is in an 8 bit format. The operand nis used to specify which analog channel is being read, i.e. 10 to 13. For more information please see page 9-4.

5.10.3 ANWR (FNC 92)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands				Program steps
		S1	D2	D2	n	
ANWR FNC 92 (F2-6A Analog write)	Used to write to the F series analog module - use with an FX2-24EI	KnY, KnM, KnS, T, C, D, V, Z	X ⊠ Note: uses 8 consecu-tive devices	Y ⊠ Note: uses 8 consecu-tive devices	K, H ⊠ Note: n= 0 or 1	ANWR, ANWRP: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

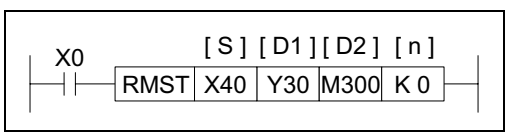
The ANWR instruction is used to write output data to channel n of an F2-6A analog module. The written analog value is stored in source device S1.
The head address I/O numbers for both S2 and Dare determined by the position of the FX2-24EI (connected to the F2-6A) within the expansion chain.
The analog data to be written is stored at the source device S1 in an 8 bit format.
The operand nis used to specify which analog channel is being written to, i.e. 0 or 1.
For more information please see page 9-4.

5.10.4 RMST (FNC 93)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands				Program steps
		S1	D1	D2	n	
RMST FNC 93 (F2-32RM RM start)	Used to start the F series CAM module - use with an FX2-24EI	X ⊠ Note: uses 8 consecu-tive devices	Y ⊠ Note: uses 8 consecu-tive devices	Y, M, S Note: uses 8 consecu-tive devices	K, H ⊠ Note: n= 0 or 1	RMST: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The RMST instruction is used to start the operation of an F2-32RM under FX control and to monitor the current status of the F2-32RM.

Points to note:

a) The head address I/O numbers for both S2 and Dare determined by the position of the FX2-24EI (connected to the F2-6A) within the FX PLC's expansion chain.

b) The operand nis used to specify which F2-32RM program is active, i.e. 0 or 1.

c) Operand D_2 stores the F_2-32RM status information.

	Status of bit device D_2+m	
	ON	**OFF**
D_{2+0}	BANK 1 (program 1) selected	BANK 0 (program 0) selected
D_{2+1}	-	Normally OFF
D_{2+2}	START	STOP
D_{2+3}	1.0 degree steps	0.5 degree steps
D_{2+4}	Normally ON	-
D_{2+5}	Clockwise operation (CW)	Counter-clockwise operation (CCW)
D_{2+6}	Normal operation - No error	Hardware Error
D_{2+7}	Normal operation - No error	Software Error

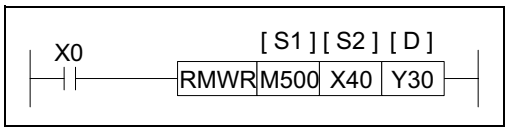
d) For more information please see page 9-4.

5.10.5 RMMR (FNC 94)

$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$

Mnemonic	Function	Operands			Program steps
		S_1	S_2	D	
RMWR FNC 94 (F_2-32RM RM write)	Disables outputs of the F series CAM module -use with an FX_2-24EI	Y, M, S Note: 16 bit operation uses 16 devices, 32 bit mode uses 32 consecutive devices.	X Note: uses 8 consecutive devices	Y ⊠ Note: uses 8 consecutive devices	RMWR,RMWRP :7 steps DRMWR, DRMWRP: 13 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
$FX_{0(s)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$	$FX_{0(s)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$	$FX_{0(s)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$



Operation:

This instruction sends output disable data to an F_2-32RM programmable CAM switch from an FX PLC.

The head address I/O numbers for both S_2 and Dare determined by the position of the FX_2-24EI (connected to the F_2-32RM) within the FX PLC's expansion chain.

The operand S_1 is the head address of either 16 or 32 source bits. The source bits map directly over F_2-32RM outputs Y0 to Y37 (numbered in octal)

When a source device is turned ON at the FX PLC the F_2-32RMs associated devices is disabled.

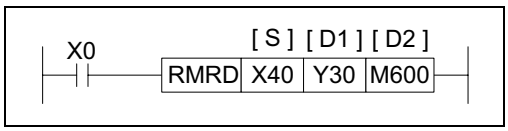
For more information please see page 9-4.

5.10.6 RMRD (FNC 95)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands			Program steps
		S	D1	D2	
RMRD FNC 95 (F2-32RM RM read)	Reads output status of F series CAM module -use with an FX2-24EI	X ⊠ Note: uses 8 consecu-tive devices	Y ⊠ Note: uses 8 consecu-tive devices	Y, M, S Note: 16 bit operation uses 16 devices, 32 bit mode uses 32 consecutive devices.	RMRD,RMRDP : 7 steps DRMRD, DRMRDP : 13 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

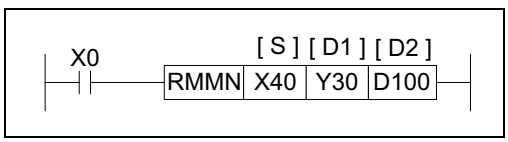
This instruction reads the current status of the outputs of an F2-32RM programmable CAM switch to an FX PLC.
The head address I/O numbers for both Sand D1 are determined by the position of the FX2-24EI (connected to the F2-32RM) within the expansion chain.
Output statuses of Y0-Y17 (16 bit operation) or Y0-Y37 (32 bit operation are read from the F2-32RM. The read data is mapped directly over the FX destination devices (head address D2). The D2devices will retain their last status even when the RMRD instruction is turned OFF. For more information please see page 9-4.

5.10.7 RMMN (FNC 96)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands			Program steps
		S	D1	D2	
RMMN FNC 96 (F2-32RM RM monitor)	Monitors data states of F series CAM module -use with an FX2-24EI	X ⊠ Note: uses 8 consecu-tive devices	Y ⊠ Note: uses 8 consecu-tive devices	KnY, YnM, YnS T, C, D, V, Z	RMMN, RMMNP: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Switch #4	Operand	Example value	Data type
ON	D2	350	angle (degrees)
OFF		830	speed (r.p.m)

Operation:

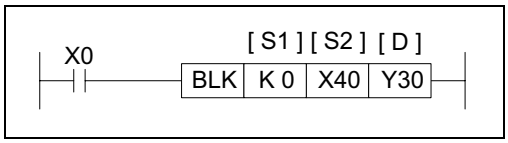
The RMMN instruction is used to read speed (r.p.m.) or current angular position from the F2-32RM to an FX PLC. There it is stored at the device specified by operand D2. The decision for which data is read, i.e. speed or position is made by the F2-32RM through its hardware switch #4. The head address I/O numbers for both S and D1 are deter-mined by the position of the FX2-24EI (connected to the F2-32RM) within the FX PLC's expansion chain.
For more information please see page 9-4.

5.10.8 BLK (FNC 97)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
BLK FNC 97 (F2-30GM Block)	Identifies a block number to the F2-30GM - use with an FX2-24EI	K, H KnX, KnY, KnM, KnS T, C, D, V, Z	X ⊠ Note: uses 8 consecutive devices	Y ⊠ Note: uses 8 consecutive devices	BLK, BLKP: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

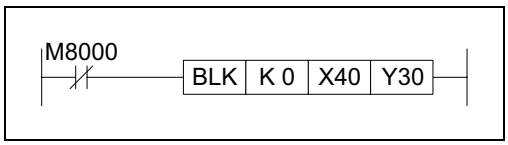
The BLK instruction is used to designate a block number (S1)to an F2-30GM pulse output unit through a controlling FX PLC.

Points to note:

a) The head address I/O numbers for both Sand D1 are determined by the position of the FX2-24EI (connected to the F2-30GM) within the FX PLC's expansion chain.

b) Effective block numbers which can be specified for S1 are 0 to 31 (in decimal). Data which is to be used for S1 cannot be in a BCD format, i.e data should not be read directly in (using KnX for example) from BCD thumbwheels.

c) When an F2-30GM is used and the BLK instruction is not required, the program opposite is needed to ensure the FX2-24EI recognizes and operates correctly for the connected F2-30GM.



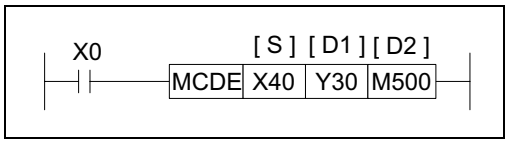
d) For more information please see page 9-5.

5.10.9 MCDE (FNC 98)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands			Program steps
		S	D1	D2	
MCDE FNC 98 (F2-30GM Machine code)	Reads a set codes from the F2-30GM - use with an FX2-24EI	X ⊠ Note: uses 8 consecutive devices	Y ⊠ Note: uses 8 consecutive devices	Y, M, S ⊠ Note: uses 64 consecutive (numbered in octal) devices	MCDE, MCDEP: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

This instruction allows an FX PLC to read the machine codes of an F2-30GM. There are 64 machine code points in the F2-30GM. These are numbered in octal, i.e. 0 to 77 and are prefixed by M.

Points to note:

a) The head address I/O number for both Sand D1 is determined by the position of the FX2-24EI (connected to the F2-30GM) within the expansion chain.

b) When the F2-30GM operates a machine code (one of the M0 to M77 devices), an associated destination device (D2) is activated. D2 devices directly map to the machine codes of the F2-30GM, i.e. when the F2-30GM activates M7, D2+7 (in octal) at the FX PLC also activates. If the data used in the sample instruction above is used and M77 at the F2-30GM is ON - so will (D2+77 an octal addition) M577 at the FX PLC be ON (while the MCDE instruction is active).

c) To make the associate numbering easy between FX and F2-30GM it is recommended that the lower two digits of the head address used to specify D2 are zero, i.e.'☆00' when S or M devices are used in the FX PLC.

d) For more information please see page 9-5.

Applied Instructions:

$FX_{0(S)}$	FX_{0N}	FX	$FX_{(2C)}$	$FX_{2N(C)}$

5.11 Floating Point 1 & 2 - Functions 110 to 129

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

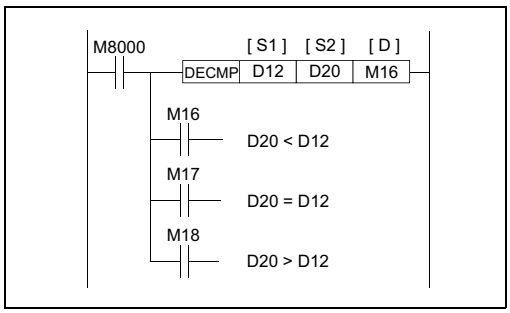
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.11.1 ECMP (FNC 110)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The data of S_1 is compared to the data of S_2. The result is indicated by 3 bit devices specified with the head address entered as D. The bit devices indicate:

S_2 is less than < S_1 - bit device D is ON
S_2 is equal to = S_1 - bit device D_{+1} is ON
S_2 is greater than > S_1 - bit device D_{+2} is ON

Points to note:

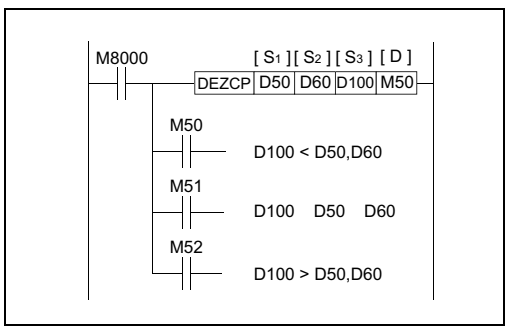
The status of the destination devices will be kept even if the ECMP instruction is deactivated.
Full algebraic comparisons are used: i.e. -1.79×10^{27} is smaller than 9.43×10^{-15}

5.11.2 EZCP (FNC 111)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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). **Note:** S_1 must be less than S_2			Y, M, S Note: 3 consecutive devices are used.	DEZCP, DEZCPP: 13 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

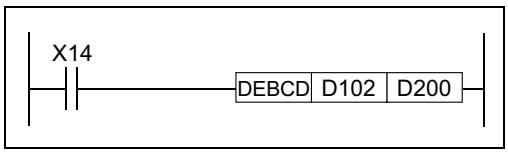
The operation is the same as the ECMP instruction except that a single data value (S_3) is compared to a data range (S_1 - S_2).
S_3 is less than S_1 and S_2 - bit device D is ON
S_3 is between S_1 and S_2 - bit device D_{+1} is ON
S_3 is greater than S_2 - bit device D_{+2} is ON

5.11.3 EBCD (FNC 118)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

Converts a floating point value at S into separate mantissa and exponent parts at D and D+1 (scientific format).

Points to note:

a) The instruction must be double word format. The destinations D and D+1 represent the mantissa and exponent of the floating point number respectively.

b) To provide maximum accuracy in the conversion the mantissa D will be in the range 1000 to 9999 (or 0) and the exponent D+1 corrected to an appropriate value.

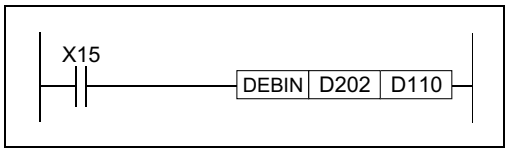
c) E.g. S= 3.4567 $\times$ 10 $^{-5}$ will become D= 3456, D+1 = -8

5.11.4 EBIN (FNC 119)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

Generates a floating point number at D from scientific format data at source S.

Points to note:

a) 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.

b) 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.

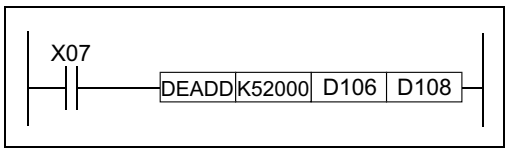
c) E.g. S= 5432, S+1 = 12 will become D= 5.432 x 10 ^{9}

5.11.5 EADD (FNC 120)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Zero M8020 Borrow M8021 Carry M8022
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)		



Operation:

The floating point values stored in the source devices S_1 and S_2 are algebraically added and the result stored in the destination device D.

Points to note:

a) The instruction must use the double word format; i.e., **D**EADD or **D**EADDP. 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.

b) If a constant K or H is used as source data, the value is converted to floating point before the addition operation.

c) The addition is mathematically correct: i.e., $2.3456 \times 10^{2} + (-5.6 \times 10^{-1}) = 2.34 \times 10^{2}$

d) 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.

e) 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.

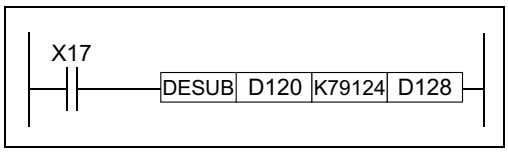
For more information about the format of floating point number refer to page 4-46.

5.11.6 EAUB (FNC 121)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		Zero M8020 Borrow M8021 Carry M8022



Operation:

The floating point value of S2 is subtracted from the floating point value of S1and the result stored in destination device D.

Points to note:

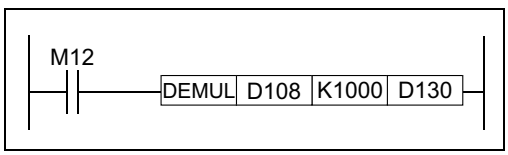
All points of the EADD instruction apply, except that a subtraction is performed.
See page 5-122.

5.11.7 EMUL (FNC 122)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
EMUL FNC 122 (Floating Point Mul-tiplication)	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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The floating point value of S1is 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:

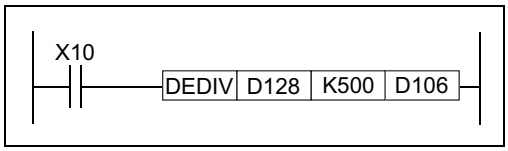
Point a, b, c and d of the EADD instruction apply, except that a multiplication is performed.
See page 5-122.

5.11.8 EDIV (FNC 123)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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.
See page 5-122.

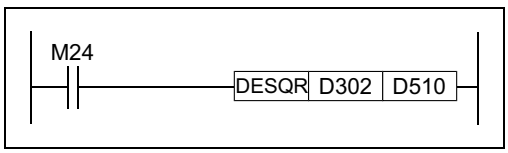
- If S2 is 0 (zero) then a divide by zero error occurs and the operation fails.

5.11.9 ESQR (FNC 127)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Zero M8020
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)		



Operation:

A square root is performed on the floating point value of Sand 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.
See page 5-122.

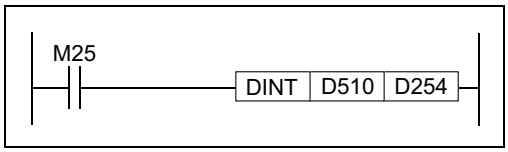
- If S is negative then an error occurs and error flag M8067 is set ON.

5.11.10 INT (FNC 129)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands		Program steps
		S	D	
INT FNC 129 (Float to Integer)	Converts a number from floating point format to decimal format	D - must be in floating point number format (always 32 bits).	D - decimal format for INT, INTP - 16 bits for DINT, DINTP - 32 bits	INT, INTP: 5 steps DINT, DINTP: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Zero M8020 Borrow M8021 Carry M8022
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)		



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:

a) 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.

b) This instruction is the inverse of the FLT instruction. (See page 5-49)

c) 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.

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.12 Trigonometry - FNC 130 to FNC 139

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

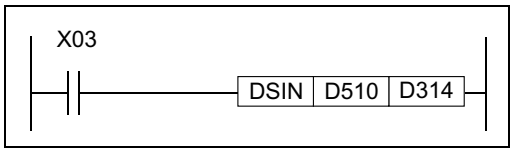
⌧ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.12.1 SIN (FNC 130)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Contents:

This instruction performs the mathematical SIN operation on the floating point value in S. The result is stored in D.

Points to note:

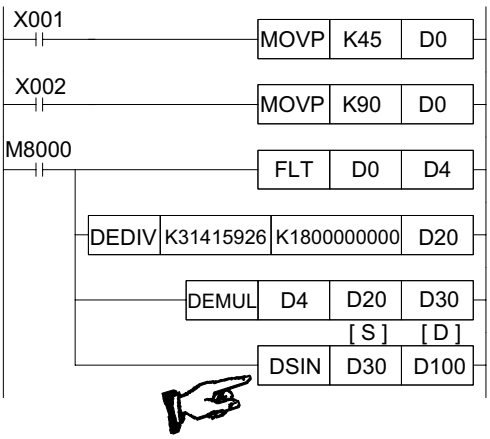
a) The instruction must use the double word format: i.e., **D**SIN or **D**SINP. All source and destination data will be double word; i.e., uses two consecutive data registers to store the data (32 bits).
The source data is regarded as being in floating point format and the destination is also in floating point format.

b) The source value must be an angle between 0 to 360 degrees in radians; i.e.,

$$0° \leq S < 360°$$

Radian Angles

Below is an program example of how to calculate angles in radians using floating point.



K45 degrees to D0

K90 degrees to D0

Convert D0 to float in D4,D5

Calculate π in radians (π/180)
Store as a float in D20,D21

Calculate angle in radians in D30,D31
(deg° × π/180 = rads)

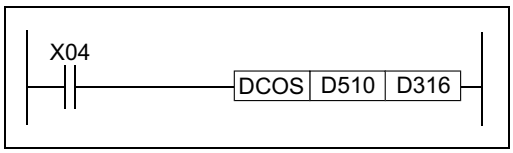
Calculate SIN of angle in D100

5.12.2 COS (FNC 131)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Contents:

This instruction performs the mathematical COS operation on the floating point value in S. The result is stored in D.

Points to note:

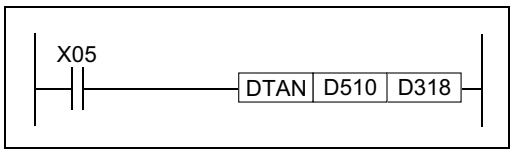
All the points for the SIN instruction apply, except that COS is calculated.
See page 5-127.

5.12.3 TAN (FNC 132)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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.
See page 5-127.

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.13 Data Operations 2 - FNC 140 to FNC 149

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

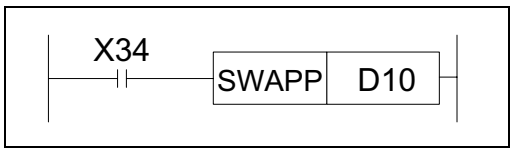
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.13.1 SWAP (FNC 147)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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 (see page 5-21).

Points to note:

a) In single word (16 bit) operation the upper and lower byte of the source device are exchanged.

b) 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

c) 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.

MEMO

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.14 Real Time Clock Control - FNC 160 to FNC 169

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

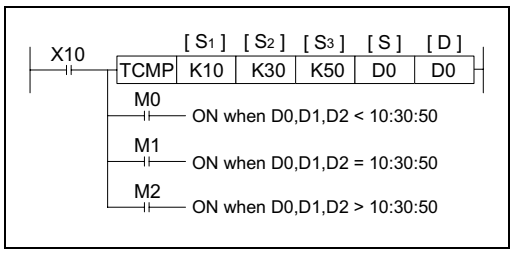
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.14.1 TCMP (FNC 160)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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, TCMPP: 11 steps
					Note: 3 consecutive devices are used.		

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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:

a) The status of the destination devices is kept, even if the TCMP instruction is deactivated.

b) The comparison is based on the time value specified in the source devices.
- The valid range of values for S1and S+0 is 0 to 23 (Hours).
- The valid range of values for S2and S+1 is 0 to 59 (Minutes).
- The valid range of values for S3and S+2 is 0 to 59 (Seconds).

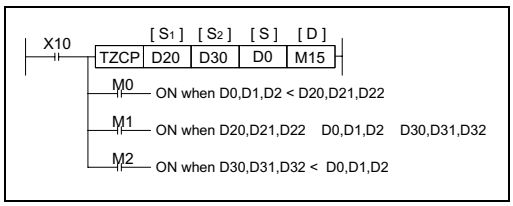
c) The current time of the real time clock can be compared by specifying D8015 (Hours), D8014 (Minutes) and D8013 (Seconds) as the devices for S1, S2 and S3 respectively.

5.14.2 TZCP (FNC 161)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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.			Y, M, S	TZCP, TZCPP: 9 steps
		Note: 3 consecutive devices are used for all				

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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:

a) The status of the destination devices is kept, even if the TCMP instruction is deactivated.

b) The comparison is based on the time value specified in the source devices.

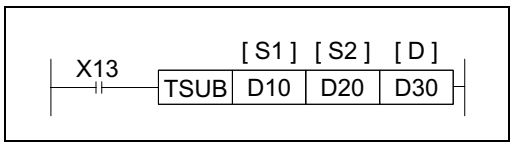
- The valid range of values for S1and S+0 is 0 to 23 (Hours).
- The valid range of values for S2and S+1 is 0 to 59 (Minutes).
- The valid range of values for S3and S+2 is 0 to 59 (Seconds).

5.14.3 TADD (FNC 162)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Zero M8020 Carry M8022
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)		



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 added to the time value in S2, 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.

S1		S2		D
D10: 10 hours		D20: 30 hours		D30: 13 hours
D11: 30 mins	+	D21: 10 mins	=	D31: 41 mins
D12: 27 secs		D22: 49 secs		D32: 16 secs
10:30:29		03:10:49		13:41:16

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.

S1		S2		D
D10: 10 hours		D20: 18 hours		D30: 13 hours
D11: 17 mins	+	D21: 12 mins	=	D31: 41 mins
D12: 29 secs		D22: 34 secs		D32: 16 secs
10:17:29		18:12:34		04:30:03
				M8022 ON

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.

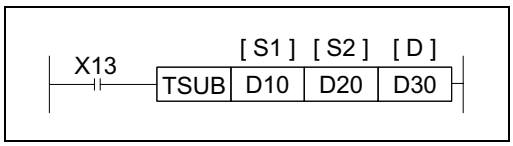
d) The same device may be used as a source (S1 or S2) and destination device. In this case the addition is continually executed; the destination value changing each program scan. To prevent this from happening, use the pulse modifier or an interlock program.

5.14.4 TSUB (FNC 163)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Mnemonic	Function	Operands			Program steps
		S1	S2	D	
TSUB FNC 163 (Time Subtrac-tion)	Subtracts one time value from another to give a new time	T, C, D Note: 3 consecutive devices are used.			TSUB, TSUBP: 7 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION					FLAGS	Zero M8020 Borrow M8021
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)		



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.

S1		S2		D
D10: 10 hours		D20: 3 hours		D30: 7 hours
D11: 30 mins	-	D21: 10 mins	=	D31: 19 mins
D12: 27 secs		D22: 49 secs		D32: 38 secs
10:30:27		03:10:49		07:19:38

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.

S1		S2		D
D10: 10 hours		D20: 18 hours		D30: 13 hours
D11: 17 mins	-	D21: 12 mins	=	D31: 41 mins
D12: 29 secs		D22: 34 secs		D32: 16 secs
10:17:29		18:12:34		16:04:55
				M8021 ON

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.

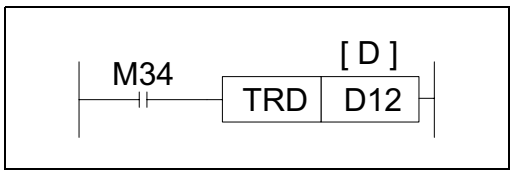
d) The same device may be used as a source (S1 or S2) and destination device. In this case the subtraction is continually executed; the destination value changing each program scan. To prevent this from happening, use the pulse modifier or an interlock program.

5.14.5 TRD (FNC 166)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Contents:

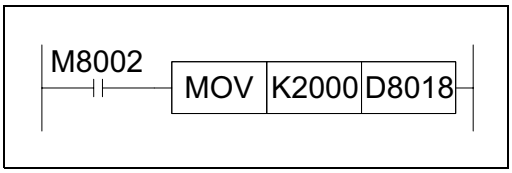
The current time and date of the real time clock are read and stored in the 7 data devices specified by the head address D.

The 7 devices are set as follows:

Device	Meaning	Values		Device	Meaning
D8018	Year	00-99	⇒	D+0	Year
D8017	Month	01-12	⇒	D+1	Month
D8016	Date	01-31	⇒	D+2	Date
D8015	Hours	00-23	⇒	D+3	Hours
D8014	Minutes	00-59	⇒	D+4	Minutes
D8013	Seconds	00-59	⇒	D+5	Seconds
D8019	Day	0-6 (Sun-Sat)	⇒	D+6	Day

Points to note:

The year is read as a two digit number. This can be change to a 4 digit number by setting D8018 to 2000 during the first program scan; see following program extract.



If this is done then the clock year should not be used during the first scan as it will be a two digit number before the instruction and a value of 2000 after the instruction until the END instruction executes. After the first scan the year is read and written as a 4 digit number.

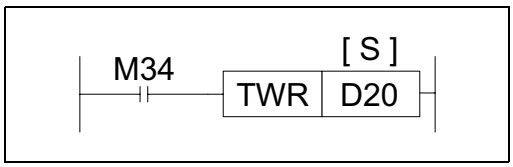
The FX-10DU-E, FX-20DU-E and FX-25DU-E only support a 2 digit year.

5.14.6 TWR (FNC 167)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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
S+0	Year	00-99	⇒	D8018	Year
S+1	Month	01-12	⇒	D8017	Month
S+2	Date	01-31	⇒	D8016	Date
S+3	Hours	00-23	⇒	D8015	Hours
S+4	Minutes	00-59	⇒	D8014	Minutes
S+5	Seconds	00-59	⇒	D8013	Seconds
S+6	Day	0-6 (Sun-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.

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.15 Gray Codes - FNC 170 to FNC 179

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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.

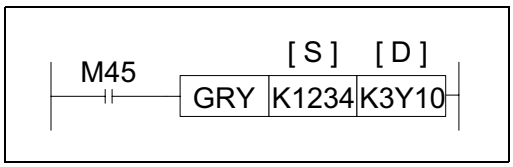
⌧ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.15.1 GRY (FNC 170)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The binary integer value in S is converted to the GRAY CODE equivalent and stored at D.

Points to Note:

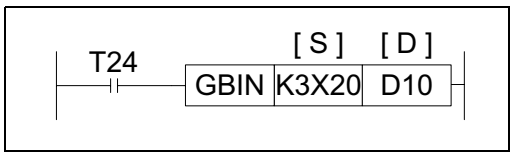
The nature of gray code numbers allows numeric values to be quickly output without the need for a strobing signal. For example, if the source data is continually incremented, the new output data can be set each program scan.

5.15.2 GBIN (FNC 171)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)	FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



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.

MEMO

Applied Instructions:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

5.16 Inline Comparisons - FNC 220 to FNC 249

Contents:

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_1, S_3 or for lists/tabled devices D_{3+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☆☆☆ - 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.

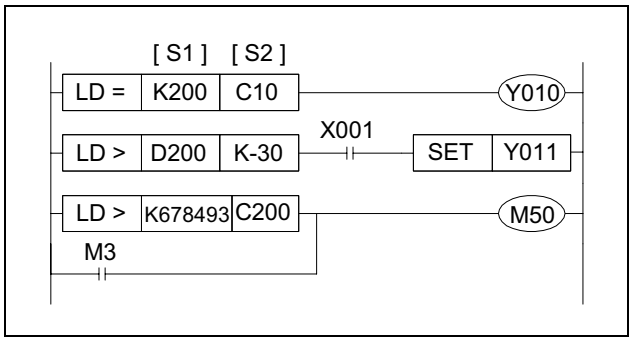
✉ - An operand which cannot be indexed, i.e. The addition of V or Z is either invalid or will have no effect to the value of the operand.

5.16.1 LD compare (FNC 224 to 230)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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	K,H, KnX, KnY, KnM, KnS, T, C, D, V, Z	LD❒: 5 steps DLD❒: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The value of S1 and S2 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.
(See page 2-3 for LD instruction)

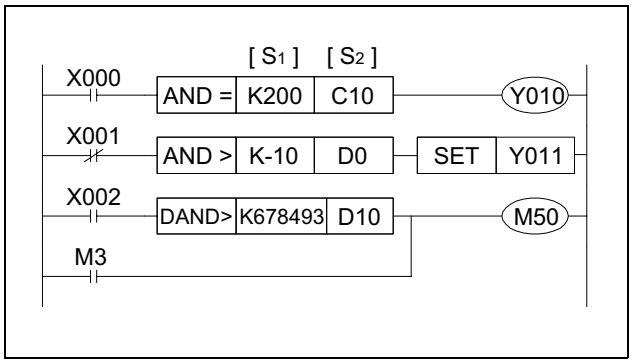
FNC No.	Mnemonic		Active when	Inactive when
	16 bit	32 bit		
224	LD =	DLD =	S1 = S2	S1 ≠ S2
225	LD >	DLD >	S1 > S2	S1 ≤ S2
226	LD <	DLD <	S1 < S2	S1 ≥ S2
228	LD <>	DLD <>	S1 ≠ S2	S1 = S2
229	LD ≤	DLD ≤	S1 ≤ S2	S1 > S2
230	LD ≥	DLD ≥	S1 ≥ S2	S1 < S2

5.16.2 AND compare (FNC 232 to 238)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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	K,H, KnX, KnY, KnM, KnS, T, C, D, V, Z	AND❒: 5 steps DAND❒: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The value of S_1 and S_2 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.
(See page 2-6 for AND instruction)

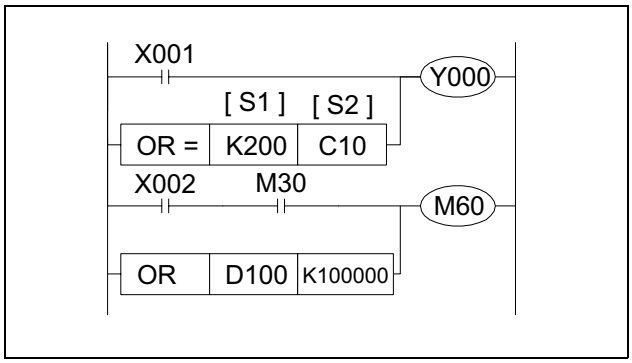
FNC No.	Mnemonic		Active when	Inactive when
	16 bit	32 bit		
232	AND =	DAND =	$S_1 = S_2$	$S_1 \neq S_2$
233	AND >	DAND >	$S_1 > S_2$	$S_1 \leq S_2$
234	AND <	DAND <	$S_1 < S_2$	$S_1 \geq S_2$
236	AND <>	DAND <>	$S_1 \neq S_2$	$S_1 = S_2$
237	AND ≤	DAND ≤	$S_1 \leq S_2$	$S_1 > S_2$
238	AND ≥	DAND ≥	$S_1 \geq S_2$	$S_1 < S_2$

5.16.3 OR compare (FNC 240 to 246)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

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	K,H, KnX, KnY, KnM, KnS, T, C, D, V, Z	OR❒: 5 steps DOR❒: 9 steps

PULSE-P					16 BIT OPERATION					32 BIT OPERATION				
FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)	FX0(s)	FX0N	FX	FX(2C)	FX2N(C)



Operation:

The value of S_1 and S_2 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.
(See page 2-7 for OR instruction)

FNC No.	Mnemonic		Active when	Inactive when
	16 bit	32 bit		
240	OR =	DOR =	$S_1 = S_2$	$S_1 \neq S_2$
241	OR >	DOR >	$S_1 > S_2$	$S_1 \leq S_2$
242	OR <	DOR <	$S_1 < S_2$	$S_1 \geq S_2$
244	OR <>	DOR <>	$S_1 \neq S_2$	$S_1 = S_2$
245	OR ≤	DOR ≤	$S_1 \leq S_2$	$S_1 > S_2$
246	OR ≥	DOR ≥	$S_1 \geq S_2$	$S_1 < S_2$

MEMO

Chapter Contents

6. Diagnostic Devices

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The following special devices are used by the PLC to highlight the current operational status and identify any faults or errors that may be occurring. There are some variations in the application of these devices to members of the FX PLC family, these are noted where appropriate.
The Internal diagnostic devices consist of both auxiliary (M) coils and data (D) registers.
Often there is a correlation between both M and D diagnostic devices for example M8039 identifies that the PLC is in constant scan mode but D8039 contains the value or length of the set constant scan.

Devices unable to be set by user:

Any device of type M or D that is marked with a "(✗)" cannot be set by a users program. In the case of M devices this means the associated coil cannot be driven BUT all contacts can be read. And for data devices (D) new values cannot be written to the register by a user BUT the register contents can be used in a data comparison.

Default Resetting Devices:

- Certain devices reset to their default status when the PLC is turned from OFF to ON. These are identified by the following symbol "(✍)".

Symbol summary:

- ✗ not able to be set by user
- ✍ automatically reset to default at power ON.
- ✍R Also reset to default when CPU is switched to RUN.
- ✍S Also reset to default when CPU is switched to STOP.

6.1 PLC Status (M8000 to M8009 and D8000 to D8009)

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

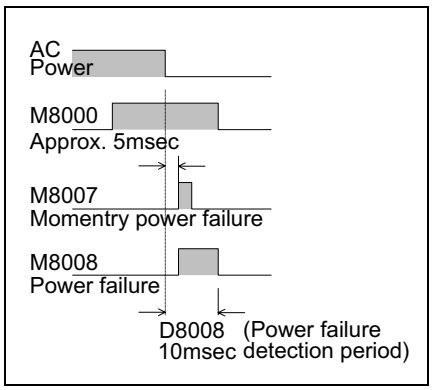
Diagnostic Device	Operation
M8000 (✗) RUN monitor NO contact	
M8001 (✗) RUN monitor NC contact	
M8002 (✗) Initial pulse NO contact	
M8003 (✗) Initial pulse NC contact	
M8004 (✗) Error occurrence	ON when one or more error flags from the range M8060 to M8067 are ON
M8005 Battery voltage Low	On when the battery voltage is below the value set in D8006
M8006 Battery error latch	Latches the battery Low error
M8007 Momentary power failure	
M8008 Power failure	Power loss has occurred
M8009 24V DC Down	Power failure of 24V DC service supply

Diagnostic Device	Operation
D8000 (✍) Watchdog timer	FX, FX2C: 100ms FX0, FX0S, FX0N, FX2N: 200ms See note 1
D8001 (✗) PLC type and version	20Vvv FX0(S), FX0N, FX, FX2C version V.vv 24Vvv: FX2N version V.vv
D8002 (✗) Memory capacity	0002: 2K steps 0004: 4K steps 0008: 8K steps (see also D8102)
D8003 (✗) Memory type	00H = RAM, 01H = EPROM, 02H = EEPROM, 0AH = EEPROM (protected) 10H = MPU memory
D8004 (✗) Error number M☆☆☆☆	The contents of this register ☆☆☆☆ identifies which error flag is active, i.e. if ☆☆☆☆ = 8060 identifies M8060
D8005 Battery voltage	E.g. 36 = 3.6 volts
D8006 Low battery voltage	The level at which a battery voltage low is detected
D8007 Power failure count	The number of time a momentary power failure has occurred since power ON.
D8008 Power failure detection	The time period before shut down when a power failure occurs (default 10ms)
D8009 24V DC failed device	Lowest device affected by 24V DC power failure

For symbol key see page 6-1.

Note 1:

- The contents of this register can be changed by the user. Settings in 1 msec steps are possible. The value should be set greater than the maximum scan time (D8012) to ensure constant scan operation.



General note:

- When the power supply used is 200V AC, the power down detection period is determined by the value of D8008. This can be altered by the user within the allowable range of 10 to 100msec.

6.2 Clock Devices (M8010 to M8019 and D8010 to D8019)

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Diagnostic Device	Operation
M8010	Reserved
M8011 (✗) 10 msec clock pulse	Oscillates in 10 msec cycles
M8012 (✗) 100 msec clock pulse	Oscillates in 100 msec cycles
M8013 (✗) 1 sec clock pulse	Oscillates in 1 sec cycles
M8014 (✗) 1 min clock pulse	Oscillates in 1 min cycles

Diagnostic Device	Operation
D8010 (✗) Present scan time	Current operation cycle / scan time in units of 0.1 msec
D8011 (✗) Minimum scan time	Minimum cycle/ scan time in units of 0.1 msec
D8012 (✗) Maximum scan time	Maximum cycle/ scan time in units of 0.1 msec (waiting time for constant scan mode is not included)

The following devices apply to FX2N(C) PLC's and to FX0N, FX and FX2C PLC's when a real time clock cassette is installed.

M8015 Time setting	When ON - clock stops, ON ➪ OFF restarts clock
M8016 Register data	When ON D8013 to 19 are frozen but clock continues
M8017 Min. rounding	When pulsed ON set RTC to nearest minute
M8018 (✗) RTC available	When ON Real Time Clock is installed
M8019 Setting error	Clock data has been set out of range

D8013 Seconds	Seconds data for use with an RTC cassette (0 - 59) *(FX0(S), FX0N see note 2)*
D8014 Minute data	Minute data for use with an RTC cassette (0-59)
D8015 Hour data	Hour data for use with an RTC cassette (0-23)
D8016 Day data	Day data for use with an RTC cassette (1-31)
D8017 Month data	Month data for use with an RTC cassette (1-12)
D8018 Year data	Year data for use with an RTC cassette (0-99)
D8019 Weekday data	Weekday data for use with an RTC cassette (0-6)

For symbol key see page 6-1.

Note 2:

- For FX0, FX0S PLC's and FX0N PLC's not fitted with a RTC, the register D8013 represents the value read from the first setting 'pot' in msec, range (0 to 255).

6.3 Operation Flags

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Diagnostic Device	Operation
M8020 (✗) Zero	Set when the result of an ADD (FNC 20) or SUB (FNC 21) is "0"
M8021 (✗) Borrow	Set when the result of a SUB (FNC 21) is less than the min. negative number
M8022 (✍) Carry	Set when 'carry' occurs during an ADD (FNC 20) or when an overflow occurs as a result of a data shift operation
M8023 (✍) Float operation flag *(FX2C only)*	Set to enable floating point operations. This flag can be used with BCD, BIN, ADD, SUB, MUL,DIV, SQR and FLT applied instructions
M8024	Reserved
M8025 *(Not FX0(S), FX0N)*	When ON HSC (FNC 53 - 55) instructions are processed even when the external HSC reset input is activated
M8026 *(Not FX0(S), FX0N)*	RAMP (FNC 67) hold mode
M8027 *(Not FX0(S), FX0N)*	PR (FNC 77) 16 element data string
M8028	FX0(S), FX0N: Change timers T32 to T55 to 10ms type FX, FX2C, FX2N: enable interrupts during FROM/ TO
M8029 (✗) Instruction execution complete	Set on the completion of operations such as DSW (FNC 72), RAMP (FNC 67) etc.

Diagnostic Device	Operation
D8020 *(FX0/FX0S/ FX0N only)*	Input filter setting for devices X000 to X007 default is 10 msec, (0-15)
D8020 *(FX2N(C) only)*	Input filter setting for devices X000 to X017 default is 10 msec, (0-15)
D8021 *(FX0/FX0S only)*	Input filter setting for devices X010 to X017 default is 10 msec, (0-15)
D8022 -D8027	Reserved
D8028 (✗)	Current value of the Z index register
D8029 (✗)	Current value of the V index register

For symbol key see page 6-1.

General note regarding input filters

- The settings for input filters only apply to the main processing units which use 24V DC inputs. AC input filters are not adjustable.

6.4 PLC Operation Mode (M8030 to M8039 and D8030 to D8039)

FX0(S) FX0N FX FX(2C) FX2N(C)

Diagnostic Device	Operation
M8030 (✍) Battery LED OFF *(Not FX0(S), FX0N)*	Battery voltage is low but BATT.V LED not lit
M8031 (✍) Non-latch memory all clear	Current device settings are reset at next END, i.e. contacts, coils and current data values for Y, M, S, T, C and D devices respectively. Special devices which have default settings are refreshed with those defaults
M8032 (✍) Latch memory all clear	
M8033 (✍) Memory hold in 'stop' mode	The device statuses and settings are retained when the PLC changes from RUN to STOP and back into RUN
M8034 (✍) All outputs disable	All of the physical switch gear for activating outputs is disabled. However, the program still operates normally.
M8035 (✍S) Forced operation mode	By using forced operation mode, i.e.M8035 is turned ON, it is possible to perform remote RUN/STOP or pulsed RUN/ STOP operation. Please see Chapter 10 for example operation
M8036 (✍S) Forced RUN signal	
M8037 (✍S) Forced STOP signal	
M8038(✍) RAM file registers clear (FX(2C) only)	Used to clear the contents of the 2000 RAM file registers of any data
M8039 (✍) Constant scan mode	When ON the PLC executes the user program within a constant scan duration. The difference between the actual end of the program operation and the set constant scan duration causes the PLC to 'pause'.

Diagnostic Device	Operation
D8030 (✗) *(FX0N only)*	Value read from first setting "pot" in msec, (0 to 255)
D8030 *(Not FX0(S), FX0N)*	This register is used as the 3rd storage device when Z or V has been selected as the destination for the SPD instruction (FNC 56)
D8031 (✗) *(FX0N only)*	Value read from second setting "pot" in msec, (0 to 255)
D8032 -D8038	Reserved
D8039 (✍) Constant scan duration	This register can be written to by the user to define the duration of the constant scan. Resolutions of 1msec are possible. This register has a default setting 0 msec which will be initiated during power ON.

For symbol key see page 6-1.

6.5 Step Ladder (STL) Flags (M8040 to M8049 and D8040 to D8049)

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Diagnostic Device	Operation
M8040 (✍) STL transfer disable	When ON STL state transfer is disabled
M8041 (✍S) Transfer start	When ON STL transfer from initial state is enabled during automatic operation (ref. IST FNC 60)
M8042 (✍) Start pulse	A pulse output is given in response to a start input (ref. IST FNC 60)
M8043 (✍S) Zero return complete	On during the last state of ZERO RETURN mode (ref. IST FNC 60)
M8044 (✍S) Zero point condition	ON when the machine zero is detected (ref. IST FNC 60)
M8045 (✍) All output reset disable	Disables the 'all output reset' function when the operation mode is changed (ref. IST FNC 60)
M8046 (✗) STL state ON	ON when STL monitoring has been enabled (M8047) and there is an active STL state
M8047 (✍) Enable STL monitoring	When ON D8040 to D8047 are enabled for active STL step monitoring
M8048 (✗) Annunciator ON *(Not FX0(S), FX0N)*	ON when Annunciator monitoring has been enabled (M8049) and there is an active Annunciator flag
M8049 (✍) Enable Annunciator monitoring *(Not FX0(S), FX0N)*	When ON D8049 is enabled for active Annunciator state monitoring

Diagnostic Device	Operation
D8040 (✗) Lowest active STL step	Up to 8 active STL states, from the range S0 to S899, are stored in D8040 to D8047 in ascending numerical order. (Updated at END)
D8041 (✗) 2nd active STL state	
D8042 (✗) 3rd active STL state	
D8043 (✗) 4th active STL state	
D8044 (✗) 5th active STL state	
D8045 (✗) 6th active STL state	
D8046 (✗) 7th active STL state	
D8047 (✗) 8th active STL state	
D8048	Reserved
D8049 (✗) Lowest active Annunciator *(Not FX0(S), FX0N)*	Stores the lowest currently active Annunciator from the range S900 to S999 (Updated at END)

For symbol key see page 6-1.

General note:

- All STL states are updated when the END instruction is executed.

6.6 Interrupt Control Flags (M8050 to M8059 and D8050 to D8059)

FX0(S) FX0N FX FX(2C) FX2N(C)

Diagnostic Device	Operation
M8050 (✍) I00❑ disable	When the EI (FNC 04) instruction is driven in the user program, all interrupts are enabled unless the special M devices noted here are driven ON. In that case for each special M coil that is ON, the associated interrupt is disabled, i.e. will not operate. Note ❑❑ denotes all types of that interrupt
M8051 (✍) I10❑ disable	
M8052 (✍) I20❑ disable	
M8053 (✍) I30❑ disable	

Diagnostic Device	Operation
D8050 -D8059	Reserved

The following assignments only refer to FX, FX2C FX2N(C) PLC's

M8054 (✍) I40❑ disable	When the EI (FNC 04) instruction is driven in the user program, all interrupts are enabled unless the special M devices noted here are driven ON. In that case for each special M coil that is ON, the associated interrupt is disabled, i.e. will not operate. Note ❑❑ denotes all types of that interrupt
M8055 (✍) I50❑ disable	
M8056 (✍) I6❑❑ disable	
M8057 (✍) I7❑❑ disable	
M8058 (✍) I8❑❑ disable	
M8059(✍) I010 to I060 disabled as a single group *(FX(2C), FX2N only)*	

The following assignments only refer to FX0, FX0S and FX0N PLC's

M8054	Reserved
M8055	Reserved
M8056 (✍) X0 pulse catch	When the leading edge of a pulse is received at an input from the range X0 to X3 the associated M device detailed here is set ON. By resetting the same device within the user program the next pulse occurrence will again set the M coil ON. Hence, fast input pulses are 'caught' and stored. This operation continues regardless of any EI (FNC04) and DI (FNC 5) instructions. For FX, FX2C, FX2N operation see page 6-12
M8057 (✍) X1 pulse catch	
M8058 (✍) X2 pulse catch	
M8059 (✍) X3 pulse catch	

For symbol key see page 6-1.

6.7 Error Detection Devices (M8060 to M8069 and D8060 to D6069)

FX0(S) FX0N FX FX(2C) FX2N(C)

Diagnostic Device	Operation		
	Detection	PROG.E LED	PLC STATUS
M8060 (✗) I/O configuration error *(Not FX0(S), FX0N)*	While the PLC is in RUN	OFF	RUN
M8061 (✗) PLC hardware error		Flash	STOP
M8062 (✗) PC/HPP communication error *(Not FX0(S), FX0N)*	When a signal from the HPP is received	OFF	RUN
M8063(✗)(✍R) Parallel link/ ADP error *(Not FX0(S), FX0N)*	When paired stations signal is received or ADP communications are in error		
M8064 (✗) Parameter error	When the program is changed (PLC in STOP) and when a program is transferred (PLC in STOP)	Flash	STOP
M8065 (✗) Syntax error			
M8066 (✗) Program error			
M8067(✗)(✍R) Operation error	While in PLC is in RUN	OFF	RUN
M8068 (✍) Operation error latch			
M8069 (✍) I/O bus error *(Not FX0(S), FX0N)*	See note 4	-	-

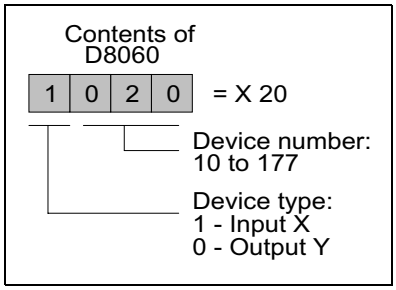
Diagnostic Device	Operation
D8060 (✗) *(Not FX0(S), FX0N)*	The first I/O number of the unit or block causing the error - See note 3
D8061 (✗)	Error code for hardware error - See also appropriate PLC hardware manual
D8062 (✗) *(Not FX0(S), FX0N)*	Error code for PC/HPP Communications error -See appropriate error code table
D8063(✗)(-R) *(Not FX0(S), FX0N)*	Error code for parallel link error - See also hardware manual. Also identification of ☆☆☆ADP communication error - See also FX-485PC-IF users manual
D8064 (✗)	Error code identifying parameter error - See appropriate error code table
D8065 (✗)	Error code identifying syntax error - See appropriate error code table
D8066 (✗)	Error code identifying program construction error See appropriate error code table
D8067(✗)(✍R)	Error code identifying operation error. See appropriate error code table
D8068 (✍)	Operation error step number latched
D8069(✗)(✍R)	Step numbers for found errors corresponding to flags M8065 to M8067

For symbol key see page 6-1.

- Please see the following page for the notes referenced in this table.

Note 3:

- If the unit or block corresponding to a programmed I / O number is not actually loaded, M8060 is set to ON and the first device number of the erroneous block is written to D8060.



Note 4:

- An I/O bus check is executed when M8069 is turned ON. If an error occurs, error code 6103 or 6104 is written to D8069 and M8061 is turned ON.

General note:

- HPP refers to Handy programming panel.

6.8 Link And Special Operation Devices (M8070 to M8099 and D8070 to D8099)

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Diagnostic Device	Operation
M8070 (✍R)	Driven when the PLC is a master station in a parallel link application
M8071 (✍R)	Driven when the PLC is a slave station in a parallel link application
M8072 (✗)	ON while the PLC is operating in a parallel link
M8073 (✗)	ON when M8070/ M8071 are incorrectly set during parallel link operations
M8074 *(FX(2C) only)*	Activate RAM file registers
M8075 -M8098	Reserved
M8099 (✍)	High speed free timer operation

→

Diagnostic Device	Operation
D8070 (✗)	Parallel link watchdog time - 500 msec
D8071 - D8098	Reserved
D8099	Free ring timer, range: 0 to 32,767 in units of 0.1 msec (for use in measuring input pulse durations)

For symbol key see page 6-1.

6.9 Miscellaneous Devices (M8100 to M8119 and D8100 to D8119)

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Diagnostic Device	Operation
M8109 (✗)	Output refresh error

Diagnostic Device	Operation
D8102 (✗) Memory Capacity	0002: 2K steps 0004: 4K steps 0008: 8K steps 0016: 16K steps
D8109 (✗)	Output refresh error device number; 0, 10, 20, etc.

6.10 Communication Adapter Devices, i.e. 232ADP, 485ADP (M8120 to M8129 and D8120 to D8129)

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Diagnostic Device	Operation
M8120	FX0N only setting data backup flag for 485 network
M8121(✗)(✍R)	RS- Data transmission delayed
M8122 (✍R)	RS- Data transmission flag
M8123 (✍R)	Finished receiving data
M8124 (FX/FX2C only)	RS- Carrier detection flag
M8125	Reserved
M8126	RS485 - global flag
M8127	RS485 - On Demand hand-shake flag
M8128	RS485 - On Demand error flag
M8129	RS485 - On Demand Byte/ Word flag, ON = Byte, OFF= Word

Diagnostic Device	Operation
D8120	Communications format
D8121	Local station number for 485 data network
D8122(✗)(✍R)	RS- Amount of remaining data to be transmitted
D8123(✗)(✍R)	RS - Amount of data already received
D8124	RS - Data header, default STX (02H)
D8125	232ADP - Data terminator, default ETX (03H)
D8126	Reserved
D8127	RS485 - On Demand head device register
D8128	RS485 - On Demand data length register
D8129	RS485 data network ‘time-out’ timer value

6.11 High Speed Zone Compare Table Comparison Flags (M8130 to M8139 and D8130 to D8139)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Diagnostic Device	Operation
M8130	Selects comparison tables to be used with the HSZ instruction
M8131 (✗)(✍)	Identifies when the HSZ comparison table has been processed.
M8132	Selects the use of the PLSY instruction with the HSZ comparison tables
M8133 (✗)(✍)	Identifies when the HSZ comparison table (when used with the PLSY instruction) has been processed.
M8134- M8139	Reserved

Diagnostic Device	Operation
D8130 (✗)(✍)	Contains the number of the current record being processed in the HSZ comparison table
D8131 (✗)(✍)	Contains the number of the current record being processed in the HSZ comparison table when the PLSY operation has been enabled
D8132 (✗)(✍)	Contains the source (output pulse frequency) data for the PLSY instruction when used with the HSZ comparison table
D8133	Reserved
D8134 D8135 (✗) (✍)	Contains a copy of the value for the current comparison when the HSZ comparison table and combined PLSY output are used. This data is only available in 32 bit or double word format.
D8136 D8137 (✗) (✍)	Contains the total number of pulses that have been output using the PLSY (or PLSR) instruction. This data is only available in 32 bit or double word format
D8138 - D8139	Reserved
D8140 D8141 (✗) (✍) *(FX2N(C) only)*	Contains the total number of pulses that have been output to Y0 using the PLSY or PLSR instructions. This data is only available in 32 bit or double word format.
D8142 D8143 (✗) (✍) *(FX2N(C) only)*	Contains the total number of pulses that have been output to Y1 using the PLSY or PLSR instructions. This data is only available in 32 bit or double word format.

For symbol key see page 6-1

6.12 Miscellaneous Devices (M8160 to M8199)

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Diagnostic Device	Operation
M8160	Selection of XCH operation to swap bytes in a single data word
M8161	Selection of 8 bit operations for applied instructions ASC, RS, ASCI, HEX, CCD
M8162	High speed mode for the PRUN instruction, 2 data words Read/write only
M8163 -M8166	Reserved
M8167	Selection of hexadecimal input mode for the HKY instruction
M8168	Selection of BCD mode for use with the SMOV instruction
M8169	Reserved
M8170 (✍R) X0 pulse catch	When the leading edge of a pulse is received at an input from the range X0 to X5 the associated M device detailed here is set ON. By resetting the same device within the user program the next pulse occurrence will again set the M coil ON. Hence, fast input pulses are 'caught' and stored. This operation requires the EI (FNC04) instruction to be active. For details of FX0,FX0S and FX0N operation see page 6-7
M8171 (✍R) X1 pulse catch	
M8172 (✍R) X2 pulse catch	
M8173 (✍R) X3 pulse catch	
M8174 (✍R) X4 pulse catch	
M8175 (✍R) X5 pulse catch	
M8176 -M8179	Reserved

Diagnostic Device	Operation
M8180	Reserved
M8181 (I010)	These special M coils can be entered into the HSCS (FNC 53) and HSCR (FNC 54)instructions instead of the associated Interrupt Pointers when using old version of programming software or hand held programmers. More details on this can be found in the 'Introduction' to this manual.
M8182 (I020)	
M8183 (I030)	
M8184 (I040)	
M8185 (I050)	
M8186 (I060)	
M8187 -M8189	Reserved
M8190 (+ MOV = SQR)	These special M coils modify the instruction noted in the brackets as '+ ☆☆☆' to perform the same task as the instruction mnemonic identified as '= ☆☆☆'. This is to allow old versions of programming software or hand held programmers to access the higher functions introduced on the FX CPU ver 3.07 and the FX2C units. More details on this can be found in the 'Introduction' to this manual.
M8191 (+ MOV = FLT)	
M8192 (+SMOV=SORT)	
M8193 (+ RAMP = SER)	
M8194 (+ RAMP = RS)	
M8195 (+ FMOV = CCD)	
M8196 (+FMOV = ASCI)	
M8197 (+FMOV = HEX)	
M8198	When this device is set ON it reverses the operation of the Source and Destination devices specified in the instruction. More details on this can be found in the 'Introduction' to this manual.
M8199	Reserved

For symbol key see page 6-1.

6.13 Index Registers (D8180 to D8199)

FX0(S) FX0N FX FX(2C) FX2N(C)

Diagnostic Device	Operation
D8180 (✗)	Reserved
D8181 (✗)	
D8182 (✗)	Z1 index register
D8183 (✗)	V1 index register
D8184 (✗)	Z2 index register
D8185 (✗)	V2 index register
D8186 (✗)	Z3 index register
D8187 (✗)	V3 index register
D8188 (✗)	Z4 index register
D8189 (✗)	V4 index register

Diagnostic Device	Operation
D8190 (✗)	Z5 index register
D8191 (✗)	V5 index register
D8192 (✗)	Z6 index register
D8193 (✗)	V6 index register
D8194 (✗)	Z7 index register
D8195 (✗)	V7 index register
D8196 (✗)	Reserved
D8197 (✗)	
D8198 (✗)	
D8199 (✗)	

6.14 Up/Down Counter Control (M8200 to M8234 and M8200 to D8234)

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Diagnostic Device	Operation
M8200 - M8234 (✍)	When M8☆☆☆ is operated, counter C☆☆☆ functions as a down counter. When M8☆☆☆ is not operated the associated counter operates as an up counter

Diagnostic Device	Operation
D8200 -D8234	Reserved

For symbol key see page 6-1.

6.15 High Speed Counter Control (M8235 to M8255 and D8235 to D8255)

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Diagnostic Device	Operation
M8235 -M8245 (✍)	When M8☆☆☆ is operated, the 1 phase high speed counter C☆☆☆ functions as a down counter. When M8☆☆☆ is not operated the associated counter operates as an up counter. The available counters depends upon the PLC type.
M8246 - M8255 (✗)(✍)	When M8☆☆☆ is operated, the 2 phase high speed counter C☆☆☆ functions as a down counter. When M8☆☆☆ is not operated the associated counter operates as an up counter. The available counters depends upon the PLC type.

Diagnostic Device	Operation
D8235 -D8255	Reserved

For symbol key see page 6-1.

6.16 Error Code Tables

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Error Detection Device	Stored Error Number	Associated Meaning	Action
D8061 **PLC Hardware error**	0000	No error	Check the cable connection between the programming device and the PLC
	6101	RAM error	
	6102	Operation circuit error	
	6103	I/O bus error (M8069 = ON)	
	6104	Extension unit 24V failure (M8069=ON)	
	6105	Watch Dog Timer error	Program execution time has exceeded the WDT time value set in D8000.

Error Detection Device	Stored Error Number	Associated Meaning	Action
D8062 **PC/HPP communications error**	0000	No error	Check the cable connection between the programming device and the PLC
	6201	Parity/ overrun/ framing error	
	6202	Communications character error	
	6203	Communication data sum check error	
	6204	Data format error	
	6205	Command error	

Error Detection Device	Stored Error Number	Associated Meaning	Note
D8063 **Serial communication errors**	0000	No error	Check both power and communications connections Special note regarding the 485 network: Because these errors are not transmitted through the network, they must be monitored by the unit acting as Master to the network
	6301	Parity/ overrun/ framing error	
	6302	Comms character error	
	6303	Comms data sum check error	
	6304	Comms data format error	
	6305	Command error 485 Network - received command other than GW (global) when station number was FF	
	6306	Watchdog timer error	
	6312	Parallel link character error	
	6313	Parallel link data sum check error	
	6314	Parallel link data format error	

Error Detection Device	Stored Error Number	Associated Meaning	Action
D8064 **Parameter error**	0000	No error	STOP the PLC, select the parameter mode, set the correct data
	6401	Program sum check error	
	6402	Memory capacity setting error	
	6403	Latched device area setting error	
	6404	Comment area setting error	
	6405	File register area setting error	
	6406 - 6408	Reserved	
	6409	Other setting error	

Error Detection Device	Stored Error Number	Associated Meaning	Action
D8065 **Syntax error**	0000	No error	During programming, each instruction is checked as it is entered. If a syntax error is detected, re-enter the instruction correctly
	6501	Incorrect instruction/ device symbol/ device number combination	
	6502	No timer or counter coil before setting value	
	6503	1) No setting value following either a timer or a counter coil 2) Insufficient number of operands for an applied instruction	
	6504	1) The same label number is used more than once 2) The same interrupt input or high speed counter input is used more than once	
	6505	Device number is outside the allowable range	
	6506	Invalid applied instruction	
	6507	Invalid P assignment	
	6508	Invalid I assignment	
	6509	Other error	
	6510	MC nesting (N) number error	
	6511	Interrupt and high speed counter assigned inputs overlap	

Error Detection Device	Stored Error Number	Associated Meaning	Action
D8066 **Circuit error**	0000	No error	A circuit error occurs if a combination of instructions is incorrect or badly specified. Select programming mode and correct the identified error.
	6601	LD and LDI is used continuously 9 or more times in succession	
	6602	1) No LD/ LDI instruction. Unauthorized use of the LD / LDI, AND / ANI instructions 2) The following instructions are not connected to the active bus line: STL, RET, MCR, (P)ointer, (I)nterrupt, EI, DI, SRET, IRET, FOR, NEXT, FEND and END 3) When MPP is missing	
	6603	MPS is used continuously more than 12 times	
	6604	Unauthorized use of the MPS/ MRD/ MPP instructions	
	6505	1) A single STL branch drives 9 or more parallel circuits 2) MC/ MCR or (I)nterrupts are designated within an STL state 3) RET has not been designated or is designated out of an STL state	
	6606	1) No (P)ointer/ (I)nterrupt 2) No SRET/ IRET 3) An (I)nterrupt/ SRET or IRET has been designated within the main body of the program 4) STL/ RET/ MC or MCR have been designated within either a subroutine or an interrupt routine	
	6607	1) Unauthorized use of FOR - NEXT. 6 or more levels have been designated 2) The following instructions have been designated within a FOR -NEXT loop: STL/ RET/ MC/ MCR/ IRET/ SRET/ FEND or END	
	6608	1) Unauthorized MC/ MCR relationship 2) Missing MCR N0 3) SRET/ IRET or an (I)nterrupt has been designated within an MC/ MCR block	
	6609	Other error	

Continued on next page...

Error Detection Device	Stored Error Number	Associated Meaning	Action
D8066 **Circuit error** *(FX2N(C) only)*	6610	LD, LDI is used continuously 9 or more times in succession	A circuit error occurs if a combination of instructions is incorrect or badly specified. Select programming mode and correct the identified error.
	6611	Number of LD/LDI instructions is fewer than ANB/ORB instructions	
	6612	Number of LD/LDI instructions is more than ANB/ORB instructions	
	6613	MPS is used continuously more than 12 times	
	6614	MPS instruction missing	
	6515	MPP instruction missing	
	6616	Unauthorized use of the MPS/ MRD/ MPP instructions; possible coil missing	
	6617	One of the following instructions is not connected to the active bus line: STL, RET, MCR, (P)ointer, (I)nterrupt, EI, DI, SRET, IRET, FOR, NEXT, FEND and END	
	6618	STL/ RET/ MC or MCR programmed within either a subroutine or an interrupt routine	
	6619	Invalid instruction programmed within a FOR - NEXT loop: STL/ RET/ MC/ MCR/ I/ IRET/ SRET	
	6620	FOR - NEXT nesting exceeded	
	6621	Unmatched number of FOR and NEXT instructions	
	6622	NEXT instruction not found	
	6623	MC instruction not found	
	6624	MCR instruction not found	
	6625	A single STL branch drives 9 or more parallel circuits	
	6626	Invalid instruction programmed within an STL - RET block: MC/ MCR/ I/ IRET/ SRET	
	6627	RET instruction not found	
	6628	I/ SRET/ IRET incorrectly programmed within main program body	
	6629	P or I label not found	
	6630	SRET or IRET not found	
	6631	SRET programmed in invalid location	
	6632	IRET programmed in invalid location	

Error Detection Device	Stored Error Number	Associated Meaning	Action
D8067 **Operation error**	0000	No error	These error occur during the execution of an operation. When an operation error occurs, STOP the PLC enter programming ode and correct the fault. Note: operation errors can occur even when the syntax or circuit design is correct, e.g. D500Z is a valid statement within an FX PLC. But if Z had a value of 100, the data register D600 would be attempted to be accessed. This will cause an operation error as there is no D600 device available.
	6701	1) No jump destination for CJ or CALL instructions 2) A label is designated in a block that comes after the END instruction 3) An independent label is designated in a FOR-NEXT loop or a subroutine	
	6702	6 or more CALL instructions have been nested together	
	6703	3 or more interrupts have been nested together	
	6704	6 or more FOR - NEXT loops have been nested together	
	6705	An incompatible device has been specified as an operand for an applied instruction	
	6706	A device has been specified outside of the allowable range for an applied instruction operand	
	6707	A file register has been accessed which is outside of the users specified range	
	6708	FROM/ TO instruction error	
	6709	Other error, i.e. missing IRE/ SRET, unauthorized FOR - NEXT relationship	
D8067 **PID Operation error**	6730	Sampling time T_S (T_S<0 or >32767)	The identified parameter is specified outside of its allowable range Execution ceases PID instruction must be reset before execution will resume
	6732	Input filter value α (α<0 or >=101)	
	6733	Proportional gain K_P (K_P<0 or >32767)	
	6734	Integral time constant T_I (T_I<0 or >32767)	
	6735	Derivative gain K_D (K_D<0 or >=101)	
	6736	Derivative time constant T_D (T_D<0 or >32767)	
	6740	Sampling time TS is less than the program scan time.	TS is set to program scan time - Execution will continue.
	6742	Current value Δ exceeds its limits	Data affected resets to the nearest limit value. For all errors except 6745, this will either be a minimum of -32768 or a maximum of +32767. Execution will continue., but user should reset PID instruction.
	6743	Calculated error ε exceeds its limits	
	6744	Integral result exceeds its limits	
	6745	Derivative gain over, or differential value exceeds allowable range	
	6746	Derivative result exceeds its limits	
	6747	Total PID result exceeds its limits	
	6750	SV - PV_{nf} < 150, or system is unstable (SV - PV_{nf} has wide, fast variations)	The error fluctuation is outside the normal operation limits for the PID instruction. Execution ceases. PID instruction must be reset.
	6751	Large Overshoot of the Set Value	
	6752	Large fluctuations during Autotuning Set Process	

MEMO

Chapter Contents

7. Execution Times And Instructional Hierarchy

FX0(S) FX0N FX FX(2C) FX2N(C)

7.1 Basic Instructions

Mnemonic	Object Devices	Steps	Execution Time in μsec									
			FX0, FX0S		FX0N		FX (< Ver 3.07)		FX (> Ver 3.07), FX2C		FX2N(C)	
			ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF
LD	X,Y,M,S,T,C and special M	1	3.4		3.4		0.74		0.48		0.08	
LDI			3.2		3.2							
AND												
ANI												
OR												
ORI												
LDP	X,Y,M,S,T,C	1	Function Not Available								43.2	
LDF												
ANP											37.4	
ANF												
ORP												
ORF												
ANB	Not applicable	1	2.2		2.2		0.74		0.48		0.08	
ORB			2.0		2.0							
MPS												
MRD												
MPP												
INV			Function Not Available									
MC	Nest level, M,Y	3	17	18.2	19.2	20.4	42.8	47.8	27	30	24.8	27.5
MCR	Nest level	2	6.0		6.2		40.4		19		20.8	
NOP	Not applicable	1	1.6		1.6		0.74		0.48		0.08	
END			410		470		960		700		508	
STL	S (see note 1)	1	4.2 + 8n		6.4 + 6.8n		39.1 + 21.1n		25 + 13.5n		27.3 + 12.6n	
RET	Not applicable		8.0		12.4		40.5		20		21.6	

carried on over the page.....

Mnemonic	Object Devices	Steps	Execution Time in μsec									
			FX0, FX0S		FX0N		FX (< Ver 3.07)		FX (> Ver 3.07), FX2C		FX2N(C)	
			ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF
OUT	Y, M	1	3.2		3.2		0.74		0.48		0.08	
	S	2	7.0	7.2	7.0	7.2	50.0	48.1	26	24	24.4	24.3
	Special M	2	7.8	7.4	8.2	7.8	38.1	38.8	28	20	0.16	
	T-K	3	21.8	19.4	25.2	21.0	72.4	52.6	45	34	42.3	37.4
	T-D	3	23.4	21.0	27.2	23.0	80.0	52.6	53	34	42.2	37.2
	C-K (16 bit)	3	14.6		17.8	15.6	67.9	40.3	42	25	25.5	24.9
	C-D (16 bit)	3	16.2		19.8	17.6	75.5	40.3	47	25	25.3	25.0
	C-K (32 bit)	5	12.5	6.0	16.0	8.6	82.3	40.3	51	25	25.3	24.9
	C-D (32 bit)	5	13.9	6.0	18.0	8.6	89.9	40.3	55	25	25.2	24.9
SET	Y, M	1	3.6	2.0	3.6	2.0	0.74		0.48		0.08	
	S	2	6.8	2.6	7.0	2.8	39.0	25.5	24	16	23.7	17.2
	S when used in an STL step (see note 1)		Function Not Available				45.2+ 14.2n	25.5	28 + 9n	16	27.3+ 12.6n	17.2
	Special M	2	7.4	2.4	7.8	2.6	41.9	28.5	26	18	0.16	
RST	Y, M	1	3.4	1.8	3.6	1.8	0.74		0.48		0.08	
	S	2	6.0	2.6	6.2	2.8	40.5	25.5	23	16	23.1	17.3
	Special M	2	7.4	2.4	7.8	2.6	41.8	28.9	26	18	0.16	
	T, C	2	20.8	18.0	22.4	19.6	50.1	38.3	31	24	27	25
	D, V, Z and special D	3	10.0	2.8	9.2	3.0	35.5	25.5	22	16	21.9	17.1
PLS	Y, M	2	19.4		21.8		41.9	41.5	27	26	0.32	
PLF	Y, M	2					42.7	40.6	26	25	0.32	
P	0 TO 63	1	1.6		1.6		0.74		0.48		0.08	
I	Iooo	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.

7.2 Applied Instructions

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

Mnemonic	16/32 Bit	Execution Time in μsec													
		FX0, FX0S		FX0N		FX(< Ver 3.07)			FX(> Ver 3.07), FX2C				FX2N(C)		
		ON	OFF	ON	OFF	ON	OFF	P	ON	OFF	2nd FNC ON	P	ON	OFF	P
00 CJ	16	19.4	9.6	20.0	10.0	46.6	27.4	✓	29	8.8	-	✓	29.0	6.4	✓
01 CALL	16	Function Not Available				49.5	27.4	✓	31	8.8	-	✓	32.2	6.4	✓
02 SRET	16					34.0			21		-		21.2		
03 IRET	*1	11.2		11.6		36.7			33		-		18.1		
04 EI	*1	6.4		7.8		62.6			34	8.8	-		55.8		
05 DI	*1					37.7			17	8.8	-		18.5		
06 FEND	*1	410		470		960			700		-		508		
07 WDT	16	9.2	5.6	9.8	7.0	35.9	25.1	✓	23	8.8	-	✓	26.3	6.4	✓
08 FOR	*1	29.0		29.8		39.9			25		-		27.6		
09 NEXT	*1	12.4		12.4		29.1			19		-		5.2		
10 CMP	16	122.6	22.0	112.2	22.6	161.8	33.3	✓	49	8.8	-	✓	87.6	6.4	✓
	32	129.2	30.4	118.6	31.6	189.0	39.9		57	12.2	-		91.9	6.4	
11 ZCP	16	140.0	25.0	128.4	25.8	186.9	33.3	✓	62	8.8	-	✓	103.2	6.4	✓
	32	197.8	36.4	137.8	38.0	220.8	39.9		73	12.2	-		108.9	6.4	
12 MOV	16	46.2	18.0	47.2	18.4	78.4	33.3	✓	35	8.8	-	✓	1.52	1.52	✓
	32	52.6	23.4	53.8	24.2	98.4	39.3		43	12.2	-		1.84	1.84	
13 SMOV	16	Function Not Available				302.9	33.3	✓	170	8.8	-	✓	155.2	6.4	✓
14 CML	16					74.0	33.3	✓	51	8.8	-	✓	51.4	6.4	✓
	32					95.9	39.9		64	12.2	-		55.9	6.4	
15 MOV *2	16	Function Not Available		103.2 + 18.2n	20.8	180.5 + 17.1n	33.3	✓	118+ 10.6n	8.8	-	✓	97.0+ 1.7n	6.4	✓
16 FMOV *2	16	Function Not Available				107.6 +5.3n	33.3	✓	73+ 3.3n	8.8	-	✓	69.1+ 2.8n	6.4	✓
	32					Function Not Available			87+ 4.5n	12.2	-		73.2+ 5.2n	6.4	
17 XCH	16					90.3	33.3	✓	58	8.8	-	✓	57.2	6.4	✓
	32					113.8	39.8		72	12.2	-		64.0	6.4	
18 BCD	16	63.6		64.4	18.4	130.9	33.3	✓	82	8.8	-	✓	37.9	6.4	✓
	32	100.2		101.4	24.2	342.0	39.9		218	12.2	306		57.6	6.4	
19 BIN	16	64.4		66.2	18.4	135.4	33.3	✓	85	8.8	-	✓	32.4	6.4	✓
	32	113.4		114.8	24.2	314.3	39.9		203	12.2	157		44.5	6.4	

See end of section for * notes...

Mnemonic	16/32 Bit	Execution Time in μsec													
		FX0, FX0S		FX0N		FX(< Ver 3.07)			FX(> Ver 3.07), FX2C				FX2N(C)		
		ON	OFF	ON	OFF	ON	OFF	P	ON	OFF	2nd FNC ON	P	ON	OFF	P
20 ADD	16	69.4		70.8	21.6	115.5	33.3	✓	51	8.8	-	✓	27.6	6.4	✓
	32	81.2		82.8	31.8	114.5	39.9		63	12.2	224		28.9	6.4	
21 SUB	16	69.8		71.6	21.6	116.6	33.3	✓	52	8.8	-	✓	27.6	6.4	✓
	32	81.4		83.0	31.6	146.5	39.9		65	12.2	232		28.9	6.4	
22 MUL	16	89.4		91.0	21.6	113.4	33.3	✓	54	8.8	-	✓	25.2	6.4	✓
	32	104.6		106.4	31.2	185.0	39.9		81	12.2	162		31.4	6.4	
23 DIV	16	119.2		120.8	21.6	139.5	33.3	✓	56	8.8	-	✓	32.0	6.4	✓
	32	230.0		232.4	31.0	804.8	39.9		451	12.2	197		36.4	6.4	
24 INC	16	28.4		29.0	14.8	55.3	33.3	✓	26	8.8	-	✓	18.8	6.4	✓
	32	33.4		34.2	17.4	65.4	34.4		29	12.2	-		20.2	6.4	
25 DEC	16	28.4		29.0	14.8	55.4	33.3	✓	26	8.8	-	✓	18.9	6.4	✓
	32	33.6		34.4	17.4	65.1	34.4		29	12.2	-		20.0	6.4	
26 WAND	16	64.2		65.6	21.6	108.0	33.3	✓	67	8.8	-	✓	23.4	6.4	✓
	32	73.0		74.6	30.6	135.4	39.9		83	12.2	-		24.7	6.4	
27 WOR	16	64.2		65.6	21.6	107.9	33.3	✓	67	8.8	-	✓	23.5	6.4	✓
	32	73.0		74.6	30.6	135.5	39.9		82	12.2	-		24.7	6.4	
28 WXOR	16	64.2		65.6	21.6	106.5	33.3	✓	67	8.8	-	✓	23.5	6.4	✓
	32	73.0		74.6	30.5	133.9	39.9		82	12.2	-		25.0	6.4	
29 NEG	16	Function Not Available				55.1	33.3	✓	34	8.8	-	✓	35.3	6.4	✓
	32					65.5	34.4		41	12.2	-		38.4	6.4	
30 ROR ∗3	16					91.9+ 3.0n	33.3	✓	57+ 1.8n	8.8	-	✓	61.7	6.4	✓
	32					113.8 +3.5n	39.9		70+ 2.1n	12.2	-		65.3	6.4	
31 ROL ∗3	16					91.9+ 3.0n	33.3	✓	57+ 1.8n	8.8	-	✓	61.2	6.4	✓
	32					113.8 +3.5n	39.9		71+ 2.3n	12.2	-		65.2	6.4	
32 RCR ∗3	16					99.0+ 1.4n	33.3	✓	61+ 0.9n	8.8	-	✓	66.3+ 2.2n	6.4	✓
	32					120.8 +1.8n	39.9		75+ 1.2n	12.2	-		69.7+ 2.6n	6.4	
33 RCL ∗3	16					99.0+ 1.4n	33.3	✓	62+ 0.9n	8.8	-	✓	65.8+ 2.2n	6.4	✓
	32					120.8 +1.8n	39.3		75+ 1.2n	12.2	-		69.5+ 2.6n	6.4	
34 SFTR ∗4	16	145.2 +5.1n	24.6	156.4 +4.8n	25.4	180.8 + 70.0n	33.3	✓	172+ 42n	8.8	-	✓	107+ 53.8n	6.4	✓
35 SFTL ∗4	16	150.6 +5.1n	24.2	162.4 +4.8n	25.0	180.8 + 70.0n	33.3	✓	172+ 42n	8.8	-	✓	105+ 53.8n	6.4	✓

See end of section for ∗ notes...

Mnemonic	16/32 Bit	Execution Time in μsec													
		FX0, FX0S		FX0N		FX(< Ver 3.07)			FX(> Ver 3.07), FX2C				FX2N(C)		
		ON	OFF	ON	OFF	ON	OFF	P	ON	OFF	2nd FNC ON	P	ON	OFF	P
36 WSFR ✻2	16	Function Not Available				218.6 + 18.0n	33.3	✓	147+ 11n	8.8	-	✓	126+ 11.7n	6.4	✓
37 WSFL ✻2	16					218.6 + 18.0n	33.3	✓	147+ 11n	8.8	-	✓	125+ 11.7n	6.4	✓
38 SFWR ✻5	16					138.1	33.3	✓	87	8.8	-	✓	83.9	6.4	✓
39 SFRD ✻5	16					143.1 +6.8n	33.3	✓	84	8.8	-	✓	80.2	6.4	✓
40 ZRST ✻6	16(D)	57.2+ 1.6n	12.8	65.0+ 1.6n	13.2	161.3 +3.2n	39.9	✓	121+ 2n	8.8	-	✓	77+ 1.7n	6.4	✓
	16(S)	127.0 +2.9n		131.5 +2.9n		161.3 + 16.5n			121+ 10.5n		-		83+ 11.1n		
	16(C)	64.4+ 3.3n		70.7+ 3.3n							-				
	16(T)	65.2+ 3.3n		71.5+ 3.3n							-				
	16(M)	127.0+ 0.08n		131.5 +3.5n		161.3+ 13.5n			121+ 8.7n		-		89.2+ 9.4n		
	16(Y)	127.0 +3.5n		131.5 +3.5n							-				
41 DECO	16	881.4	20.6	932.0	21.4	114.8	28.8	✓	72	8.8	-	✓	76.0	6.4	✓
42 ENCO	16	618.3	20.6	692.4	21.4	125.6	28.8	✓	79	8.8	-	✓	81.8	6.4	✓
43 SUM	16	Function Not Available				133.5	33.3	✓	84	8.8	-	✓	72.8	6.4	✓
	32					196.6	39.9		123	12.2	-		94.6	6.4	
44 BON	16					168.9	33.3	✓	98	8.8	-	✓	78.2	6.4	✓
	32					177.6	39.9		112	12.2	-		82.3	6.4	
45 MEAN Q7	16					133.4+ 12.2n	33.3	✓	84+ 7.7n	8.8	-	✓	83.8+ 3.4n	6.4	✓
	32					Function Not Available			105+ 9.8n	12.1	-		90.9+ 6.7n	6.4	
46 ANS	16					192.6	165.6		120	110	-		100.8	96.2	
47 ANR	16					86.5	25.5	✓	54	8.8	-	✓	37.7	6.4	✓
48 SQR	16	Function Not Available							208	8.8	-	✓	150.2	6.4	✓
	32								220	12.2	344		154.8	6.4	
49 FLT	16								98	8.8	-	✓	66.8	6.4	✓
	32								114	12.2	-		66.8	6.4	
50 REF ✻8	16	53.6	12.8	65.4+ 3.1n	13.4	145.3 +3.6n	33.3	✓	62+ 3.2n	8.8	-	✓	99.6+ 0.6n	6.4	✓

See end of section for ✻ notes...

Mne-monic	16/32 Bit	Execution Time in μsec													
		FX0, FX0S		FX0N		FX(< Ver 3.07)			FX(> Ver 3.07), FX2C				FX2N(C)		
		ON	OFF	ON	OFF	ON	OFF	P	ON	OFF	2nd FNC ON	P	ON	OFF	P
51 REFF ∗9	16	Function Not Available				56.0+ 4.9n	33.3	✓	35+ 3.5n	8.8	-	4	65.3+ 1.7n	6.4	✓
52 MTR	16					87.3	39.3		53	26	-		39.1	23.6	
53 HSCS ∗10	32	75.6	6.6	82.8	7.8	175.0	39.3		115	12.2	-		87.8	6.4	
54 HSCR ∗10	32									12.2	-		88.6	6.4	
55 HSZ ∗10	32	Function Not Available				240.3	39.3		142	12.2	-		100.6	6.4	
56 SPD	∗1					164.4	163.0		102	101	-		80.2	80.2	
57 PLSY	16	189.4	10.0	212.4	21.4	154.5	173.6		101	136	-		85.0	73.3	
	32			223.4	31.4				115	136	-		86.6	75.8	
58 PWM	16	42.5	7.8	44.2	18.6	139.8	171.0		86	101	-		70.4	73.3	
59 PLSR	16	Function Not Available											122.6	87.5	
	32												125.6	90.5	
60 IST	16	766.0	322.4	212.4	21.4	272.9	33.3		153	8.8	-		114.3	6.4	
61 SER ∗14	16	Function Not Available							147+ 12.5n	29	-	4	129.2 +8.6n	22.9	✓
	32								168+ 17.4n	29	-		147+ 9.0n	22.9	
62 ABSD ∗11	16	Function Not Available				141.4+ 61.4n	33.3		91+ 35n		-		91.8+ 20.2n	6.4	
	32					Function Not Available			110+ 43n		-		97.5+ 21.5n	6.4	
63 INCD	16					208.8	39.9		130	26	-		110.5	19.5	
64 TTMR	16					81.3	69.6		48	43	-		54.9	44.9	
65 STMR	16					176.6	167.8		106	104	-		84.4	84.4	
66 ALT	16	61.0	9.8	62.8	10.0	105.6	33.3	✓	66	8.8	-	4	50.1	6.4	✓
67 RAMP	16	248.6	82.6			181.8	134.5		113	83	-		98.1	81.6	
68 ROTC	16	Function Not Available				232.5	209.1		144	130	-		118.4	107.2	
69 SORT ∗15	16	Function Not Available							62+ 32.3 m1	23	-		50.5	19.5	

See end of section for ∗ notes...

Mnemonic	16/32 Bit	Execution Time in μsec													
		FX0, FX0S		FX0N		FX(< Ver 3.07)			FX(> Ver 3.07), FX2C				FX2N(C)		
		ON	OFF	ON	OFF	ON	OFF	P	ON	OFF	2nd FNC ON	P	ON	OFF	P
70 TKY	16	Function Not Available				245.7	33.3		153	23	-		97.2	22.2	
	32					229.1	39.9		145	23	-		98.7	22.2	
71 HKY	16					318.8	39.9		189	29	-		92.2	27.4	
	32					338.0	45.5		205	29	-		65.0	6.4	
72 DSW	16-1 set					205.8	39.3		130	30	-		92.2	27.4	
	16-2 sets					208.1			NA	30	-		NA	NA	
73 SEGD	16	Function Not Available				142.1	33.3	✓	87	8.8	-	4	65.0	6.4	✓
74 SEGL	16-1 set					209.7	33.3		127	30	-		105.9	26.5	
	16-2 sets					246.9			148		-		NA	NA	
75 ARWS	16					285.0	163.0		169	8.8	-		134.4	22.1	
76 ASC	16					130.9	33.3		104	8.8	-		49.5	6.4	
77 PR	16-printing					207.1	112.6		127	58	-		114.8	88.5	
	16-ready					112.1			68		-		88.0		
78 FROM *12	16	Function Not Available		120+400n	26	170+406n	45.0	✓	120+325n	14	-	4	97+487n	6.4	✓
	32			120+800n	26	200+800n			140+640n		-		99+962n	6.4	
79 TO *12	16			120+480n	38	151+480n	45.0	✓	106+384n	14	-	4	94+557n	6.4	✓
	32			120+950n	38	200+936n			140+749n		-		96+1099n	6.4	
80 RS	16	Function Not Available		125.5	20.5	Function Not Available			132	20	-		117.6	18.0	
81 PRUN *13	16	Function Not Available				137.1+53.5n	33.3	✓	91+32n	8.8	-	4	65.6+17.0n	6.4	✓
	32					154.5+49.3n			104+34n	12.2	-		67.0+17.7n	6.4	
82 ASCI	16	Function Not Available		115+9.7n	22.3	Function Not Available			94+12n	8.8	-	4	88.2+10.8n	6.4	✓
83 HEX	16			115+22.9	22.1				95+23n	8.8	-	4	89.7+20.0n	6.4	✓
84 CCD	16			115+11.7n	24.4				96+8n	8.8	-	4	90.5+4.8n	6.4	✓
85 VRRD	16	Function Not Available				308.1	33.3	✓	209	21	-	4	209.7	27.3	✓
86 VRSC	16					319.1	33.3	✓	205	21	-	4	202.4	27.3	✓
87	16	Function Not Available													
	32														

See end of section for * notes...

Mnemonic	16/32 Bit	Execution Time in μsec													
		FX0, FX0S		FX0N		FX(< Ver 3.07)			FX(> Ver 3.07), FX2C				FX2N(C)		
		ON	OFF	ON	OFF	ON	OFF	P	ON	OFF	2nd FNC ON	P	ON	OFF	P
88 PID	16	Function Not Available							407	109	-		155.0	89.0	
89	16	Function Not Available													
	32														
90 MNET	16	Function Not Available				643.9	25.5	✓	Function Not Available						
91 ANRD	16					1,137	33.3	✓							
92 ANWR	16					1.387	470.9	✓							
93 RMST	16					948.8	950.0		691	692	-		Function Not Available		
94 RMWR	16					2,214	33.3	✓	1,612	8.8	-	4			
	32					4,235	39.9		3,127	12.2	-				
95 RMRD	16					1,684	33.3	✓	1,254	8.8	-	4			
	32					3,168	39.9		2,414	12.2	-				
96 RMMN	16					1,589	33.3	✓	1,195	8.8	-	4			
97 BLK	16					672.4	669.3	✓	Function Not Available						
98 MCDE	16					740.3	33.3	✓							
99	16	Function Not Available													
	32														
110 ECMP	32	Function Not Available											104.4	6.4	✓
111 EZCP	32												124.5	6.4	✓
118 EBCD	32	Function Not Available											106.9	6.4	✓
119 EBIN	32												81.3	6.4	✓
120 EADD	32												117.4	6.4	✓
121 ESUB	32												117.4	6.4	✓
122 EMUL	32												96.4	6.4	✓
123 EDIV	32												100.4	6.4	✓

See end of section for ∗ notes...

Mne-monic	16/32 Bit	Execution Time in μsec													
		FX0, FX0S		FX0N		FX(< Ver 3.07)			FX(> Ver 3.07), FX2C				FX2N(C)		
		ON	OFF	ON	OFF	ON	OFF	P	ON	OFF	2nd FNC ON	P	ON	OFF	P
127 ESQR	32	Function Not Available											152.1	6.4	✓
128													Function Not Available		
129 INT	16												67.5	6.4	✓
	32												70.4	6.4	
130 SIN	32												199.5	6.4	✓
131 COS	32												262.5	6.4	✓
132 TAN	32												425.3	6.4	✓
147 SWAP	16	Function Not Available											36.1	6.4	✓
	32												41.2	6.4	
160 TCMP	16	Function Not Available											134.2	6.4	✓
161 TZCP	16												140.2	6.4	✓
162 TADD	16												118.8	6.4	✓
163 TSUB	16												109.4	6.4	✓
164	16												Function Not Available		
165	16														
166 TRD	16												46.2	6.4	✓
167 TWR	16												112.0	6.4	✓
168	16												Function Not Available		
169	16														
170 GRY	16												102.5	6.4	✓
	32												107.1	6.4	
171 GBIN	16												103.4	6.4	✓
	32												107.5	6.4	
224-230 LD❑	16	Function Not Available											1.52		
	32												1.84		
232-238 AND❑	16												1.52		
	32												1.84		
240-246 OR❑	16												1.52		
	32												1.84		

See end of section for * notes...

*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. When an FX PLC is used "n" can be equal or less than 1024. However, when FX0 and FX0N controllers are used "n" can be equal or less than 512.

*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. For FX0 and FX0N there should be no more than 4 simultaneously active instructions. However, FX can have 6 simultaneously active 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 FX PLC’s 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.

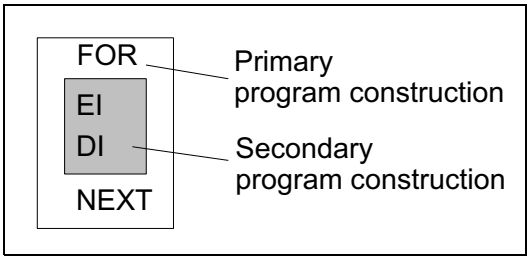
*15:

- 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.

7.3 Hierarchical Relationships Of Basic Program Instructions

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The following table identifies an 'inclusive relationship'. This means the secondary program construction is included within the complete operating boundaries of the primary program construction, e.g.:



Primary Program Construction	Secondary program construction							
	MC-MCR	**CJ - P**	**EI - DI**	**FOR - NEXT**	**STL - RET**	**P - SRET**	**I - IRET**	**FEND - END**
MC - MCR	✓ - 8 nest levels	✓	✓	✓	✓	✗ - (6608)	✗ - (6608)	✗ - (6608)
CJ - P	✓	✓	✓	✓	✓	●	●	✗ - (6701)
EI - DI	✓	✓	✓	✓	✓	✓	✓	✓①
FOR - NEXT	✗ -(6607)	✓	✓	✓ - 5 nest levels	✗ - (6607)	✗ - (6607)	✗ - (6607)	✗ - (6607)
STL - RET	✗ - (6605)	●	✓	✓ - (within 1 STL step)	✓	✗ - (6605)	✗ - (6605)	✗ - (6605)
P - SRET	✗ - (6606)	✓	✓	✓	✗ - (6606)	✗ - (6606)	✗ - (6606)	✗ - (6709)
I - IRET	✗ - (6606)	✓	✓	✓	✗ - (6606)	✗ - (6606)	✗ - (6606)	✗ - (6606)
FEND - END	✓	✓	✓	✓	●	✓	✓	✓②
0 - FEND	✓	✓	✓	✓	✓	✗ - (6606)	✗ - (6606)	✓②
0 - END (no FEND)	✓	✓	✓	✓	✓	✗ - (6606)	✗ - (6606)	✓②

✓ : Instruction combination is acceptable - for restrictions see appropriate note

✗ : Instruction combination is not allowed - bracketed number is the error code

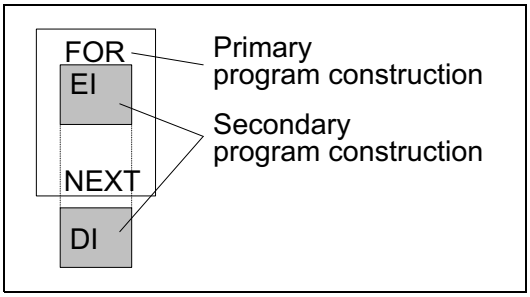
● : Instruction combination is not recommended for use even though there is no operational error

The combination of instructions with an 'inclusive relationship' is allowable. However please be aware of the following exceptions:

1) MC-MCR and STL-RET constructions cannot be used within FOR-NEXT loops, P-SRET or I-IRET subroutines.

2) Program flow may not be discontinued by using any of the following methods while inside MC-MCR, FOR-NEXT, P-SRET, I-IRET program constructions, i.e. using interrupts (I), IRET, SRET, FEND or the END instruction is not allowed.

The following table identifies an 'overlap-ping relationship'. This means the secondary program construction starts within the complete operating boundaries of the primary program construction but finishes outside of the primary construction, e.g.:

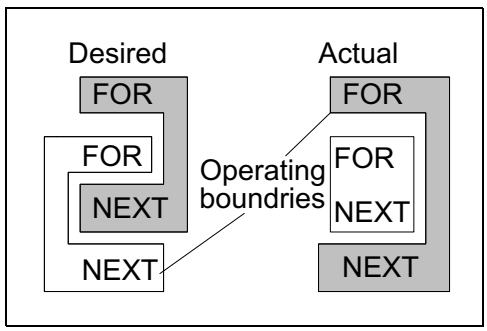


Primary Program Construction	Secondary program construction							
	MC-MCR	CJ - P	EI - DI	FOR - NEXT	STL - RET	P - SRET	I - IRET	FEND - END
MC - MCR	●	●	✓	✗ - (6607)	✗ - (6605)	✗ - (6606)	✗ - (6606)	✗ - (6608)
CJ - P	●	●	✓	●	●	●	●	✓
EI - DI	✓	✓	✓	✓	✓	✓	✓	✓
FOR - NEXT	✗ - (6607)	●	✓	✓¬	✗ - (6601)	✗ - (6607)	✗ - (6607)	✗ - (6607)
STL - RET	✗ - (6605)	●	✓	✗ - (6607)	✓	✗ - (6606)	✗ - (6606)	✗ - (6605)
P - SRET	✗ - (6608)	●	✓	✗ - (6607)	✗ - (6605)	✗ - (6606)	✗ - (6606)	✗ - (6709)
I - IRET	✗ - (6606)	●	✓	✗ - (6607)	✗ - (6606)	✗ - (6606)	✗ - (6606)	✗ - (6606)
FEND - END	✗ - (6608)	✗ - (6601)	✓①	✗ - (6607)	✗ - (6605)	✗ - (6709)	✗ - (6709)	✓②
0 - FEND	✗ - (6608)	✓	✓	✗ - (6607)	✗ - (6605)	✗ - (6709)	✗ - (6606)	✓②
0 - END (no FEND)	✗ - (6608)	✗ - (6601)	✓①	✗ - (6607)	✗ - (6605)	✗ - (6709)	✗ - (6606)	✓②

① Enters a state as if the DI instruction was missing. An error is not generated.

② The first occurrence of either an FEND or the END instruction takes priority. This would then end the program scan prematurely.

③ The sequence will not process as expected, e.g.:

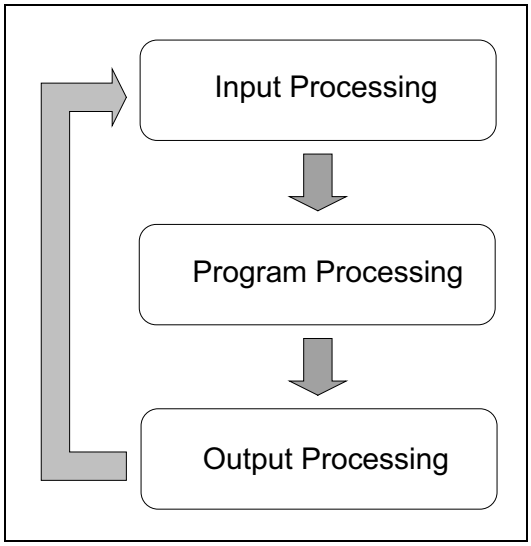


7.4 Batch Processing

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

This is the system used by all members of the FX family of PLC's. The basic concept is that there are three stages to any program scan. In other words, every time the program is processed form start to end the following sequence of events occurs:

Input processing:



All of the current input statuses are read in to a temporary memory area; sometimes called an image memory. The PLC is now ready for the next program processing.....

Program processing:

All of the updated inputs are checked as the program is processed. If the new input statuses change the status of driven outputs, then these are noted in the image memory for the.....

Output processing:

The new, current statuses of the outputs which have just be processed are physically updated, i.e relays are turned ON or OFF as required. The program scan starts again.....

The system is known as 'Batch processing' because all of the inputs, program operation and finally the outputs are processed as batches.

7.5 Summary of Device Memory Allocations

The memory allocations of the programmable are very complex, but from a users point of view there are three main areas:

a) The Program Memory:

This memory area holds all of the data regarding: parameters, sequence program, constant values K and H, pointer information for P and I devices, nest level information, file register contents/allocations and also the program comment area.

- This memory area is latched either by battery backup or by use of EEPROM program management (dependent on the PLC being used). Any data stored in this area is kept even when the PLC is powered down. The duration and reliability of the data storage is dependent upon the condition of the battery or EEPROM being used to perform the backup process.

b) Data Memory

This memory area contains, as the title suggests, all of the data values associated with: data registers (normal and special), Index registers, current timer values, retentive timer values (if available) and current counter values.

- All of the devices which are designated as being latched (including retentive timers) are backed up in a similar method to the one mentioned under point a).
- Index registers and special data registers (D8000 to D8255) operate in the specified manner under the following circumstances:

Circumstance	Reaction
PLC's power is turned OFF	All data is cleared
PLC's power is turned ON	Certain devices are reset to their defaults see chapter 6
PLC is switched from STOP to RUN	Certain devices are reset to their defaults see chapter 6
PLC is switched from RUN to STOP	

- All other devices such as current values of non latched data registers, timers and counters behave in the following manner:

Circumstance	Reaction
PLC's power is turned OFF	All data is cleared
PLC's power is turned ON	
PLC is switched from STOP to RUN	No change
PLC is switched from RUN to STOP	Cleared (unless special M coil M8033 is active)

c) Bit Memory

This memory area contains the contact status of all inputs, outputs, auxiliary relays, state coils, timers and counters.

- All of the devices which are designated as being latched (including retentive timers) are backed up in a similar method to the one mentioned under point a).
- Special auxiliary relays (M8000 to M8255) act in a similar way to the special data registers mentioned under point b).
- All other devices are subject to the same changes as the current values of data registers, timers and counter (see the last point and table under section b).

Summary

Memory type	Power		PLC	
	OFF	OFF ➢ ON	STOP ➢ RUN	RUN ➢STOP
All devices backed by battery	Not changed			
Special M and D devices (8000 to 8255) and index registers V and Z	Cleared	Default	Not changed	
All other devices	Cleared		Not changed	Cleared
			Not changed when M8033 is set	

7.6 Limits Of Instruction Usage

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

7.6.1 Instructions Which Can Only Be Used Once In The Main Program Area

The following instructions can only be used once in the main program area. For PLC applicability please check either the detailed explanations of the instructions or the instruction execution tables list earlier.

- Instructions which can only be used once are:

FNC 52 MTR	FNC 60 IST	FNC 70 TKY
FNC 57 PLSY	FNC 61 SORT	FNC 71 HKY
FNC 58 PWM	FNC 62 ABSD	FNC 72 DSW
FNC 59 PLSR	FNC 63 INCD	FNC 74 SEGL
	FNC 68 ROTC	FNC 75 ARWS

- Only one of either FNC 57 PLSY or FNC 59 PLSR can be programmed at once. Both instructions can not be present in the same active program.

7.6.2 Instructions Which Are Not Suitable For Use With 110V AC Input Units

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

When using 110V AC input units certain operations, functions and instructions are not recommended for use due to long energize/de-energize (ON/OFF) times of the 110V input devices.

- Program operations not recommended for use are:
  - Interrupt routines
  - High speed counters
- Instructions not recommended for use are:

FNC 51 REFF	FNC 68 ROTC	FNC 72 DSW
FNC 52 MTR	FNC 70 TKY	FNC 75 ARWS
FNC 56 SPD	FNC 71 HKY	

Note: although selected FX0S units have 110V AC inputs, the instructions mentioned above are not present in the CPU of the unit and hence cannot be used anyway

Chapter Contents

8. PLC Device Tables

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

8.1 Performance Specification Of The FX0 And FX0S

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: 1.6 to 3.6 µs Applied instructions: several 10's to 100 µs	
Programming language		Relay symbolic language + step ladder	Step ladder can be used to produce an SFC style program
Program capacity		800 steps	Provided by built in EEPROM memory
Number of instructions		Basic sequence instructions: 20 Step ladder instructions: 2 Applied instructions: 35	A Maximum 50 applied instructions are available including all variations
I/O configuration		Max total I/O set by Main Processing Unit	
Auxiliary relay (M coils)	General	512 points	M0 to M511
	Latched	16 points (subset)	M496 to M511
	Special	56 points	From the range M8000 to M8255
State relays (S coils)	General	64 points	S0 to S63
	Initial	10 points (subset)	S0 to S9
Timers (T)	100 msec	Range: 0 to 3,276.7 sec 56 points	T0 to T55
	10 msec	Range: 0 to 327.67 sec 24 points	T32 to T55 when special M coil M8028 is driven ON
Counters (C)	General	Range: 1 to 32,767 counts 16 points	C0 to C13 Type: 16 bit up counter
	Latched	2 points(subset)	C14 to C15 Type: 16 bit up counter
High speed counters (C)	1 phase	Range: -2,147,483,648 to +2,147,483,647 counts FX0: Select upto four 1 phase counters with a combined counting frequency of 5kHz or less. Alternatively select one 2 phase or A/B phase counter with a counting frequency of 2kHz or less. FX0S: When multiple 1-phase counters are used the sum of the frequencies must be equal or less than 14kHz. Only 1, 2 phase high speed counter may be used at any one time. When 2 phase counters are in use the maximum counted speeds must be equal or less than 14kHz, calculated as (2 ph counter speed 5 number of counted edges) + 1 ph counter speeds.	C235 to C238 (note C235 is latched) 4 points
	1 phase c/w start stop input		C241(latched), C242 and C244 (latched) 3 points
	2 phase		C246, C247 and C249 (all latched) 3 points
	A/B phase		C251, C252 and C254 (all latched) 3 points

continued over the page....

Item		Specification	Remarks
Data registers (D)	General	32 points	D0 to D31 Type:16 bit data storage register pair for 32 bit device
	Latched	2 points (subset)	D30 to D31 Type:16 bit data storage register pair for 32 bit device
	Externally adjusted	Range: 0 to 255 1 point	D8013 Data is entered indirectly through the external setting potentiometer
	Special	27 points (inclusive of D8013)	From the range D8000 to D8255 Type: 16 bit data storage register
	Index	2 points	V and Z Type: 16 bit data storage register
Pointers (P)	For use with CALL	64 points	P0 to P63
	For use with interrupts	4 points	I00❑ to I30❑ (rising trigger ❑ = 1, falling trigger ❑ = 0)
Nest levels		8 points for use with MC and MCR	N0 to N7
Constants	Decimal K	16 bit: -32,768 to +32,767 32 bit: -2,147,483,648 to +2,147,483,647	
	Hexadeci-mal H	16 bit: 0000 to FFFF 32 bit: 00000000 to FFFFFFFF	

8.2 Performance Specification Of The FX0N

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: 1.6 to 3.6 µs Applied instructions: several 10's to 100 µs	
Programming language		Relay symbolic language + step ladder	Step ladder can be used to pro-duce an SFC style program
Program capacity		2000 steps	Provided by built in EEPROM memory
Number of instructions		Basic sequence instructions: 20 Step ladder instructions: 2 Applied instructions: 42	A Maximum 59 applied instructions are available including all varia-tions
I/O configuration		Max hardware I/O configuration points 128, dependent on user selection (Max. software addressable Inputs 128, Outputs 128)	
Auxiliary relay (M coils)	General	512 points	M0 to M511
	Latched	128 points (subset)	M384 to M511
	Special	65 points	From the range M8000 to M8255

continued over the page....

Item		Specification	Remarks
State relays (S coils)	Latched	128 points	S0 to S127
	Initial	10 points (subset)	S0 to S9
Timers (T)	100 msec	Range: 0 to 3,276.7 sec 63 points	T0 to T62
	10 msec	Range: 0 to 327.67 sec 31 points	T32 to T62 when special M coil M8028 is driven ON
	1 msec	Range: 0 to 32.767 sec 1 point	T63
Counters (C)	General	Range: 1 to 32,767 counts 32 points	C0 to C31 Type: 16 bit up counter
	Latched	16 points (subset)	C16 to C31 Type: 16 bit up counter
High speed counters (C)	1 phase	Range: -2,147,483,648 to +2,147,483,647 counts Select upto four 1 phase counters with a combined counting frequency of 5kHz or less. Alternatively select one 2 phase or A/B phase counter with a counting frequency of 2kHz or less. Note all counters are latched	C235 to C238 4 points
	1 phase c/w start stop input		C241, C242 and C244 3 points
	2 phase		C246, C247 and C249 3 points
	A/B phase		C251, C252 and C254 3 points
Data registers (D)	General	256 points	D0 to D255 Type: 16 bit data storage register pair for 32 bit device
	Latched	128 points (subset)	D128 to D255 Type: 16 bit data storage register pair for 32 bit device
	File	1500 points	D1000 to D2499 set by parameter in 3 blocks of 500 program steps Type: 16 bit data storage register
	Externally adjusted	Range: 0 to 255 2 points	Data is move from external setting potentiometers to registers D8030∗ and D8031 (∗ D8013 is used when no RTC is fitted)
	Special	42 points (inclusive of D8013, D8030 and D8031)	From the range D8000 to D8255 Type: 16 bit data storage register
	Index	2 points	V and Z Type: 16 bit data storage register
Pointers (P)	For use with CALL	64 points	P0 to P63
	For use with interrupts	4 points	I00❑ to I30❑ (rising trigger ❑ = 1, falling trigger ❑ = 0)
Nest levels		8 points for use with MC and MCR	N0 to N7
Constants	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	

8.3 Performance Specification Of The FX (CPU versions 2.0 to 3.06)

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: 0.74 μs Applied instructions: several 10's to 100 μs	
Programming language		Relay symbolic language + step ladder	Step ladder can be used to produce an SFC style program
Program capacity		2000 steps built in	Expandable to 8000 steps using additional memory cassette
Number of instructions		Basic sequence instructions: 20 Step ladder instructions: 2 Applied instructions: 87	A Maximum 119 applied instructions are available including all variations
I/O configuration		Max hardware I/O configuration points 256, dependent on user selection (Max. software addressable Inputs 128, Outputs 128)	
Auxiliary relay (M coils)	General	1024 points	M0 to M1023
	Latched	524 points (subset)	M500 to M1023
	Special	256 points	From the range M8000 to M8255
State relays (S coils)	General	1000 points	S0 to S999
	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)	C219 to C234 Type: 16 bit up/down counter

continued over the page....

Item		Specification	Remarks
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
Data registers (D)	General	512 points	D0 to D511 Type: 16 bit data storage register pair for 32 bit device
	Latched	312 points (subset)	D200 to D511 Type: 16 bit data storage register pair for 32 bit device
	File	2000 points	D1000 to D2999 set by parameter in 4 blocks of 500 program steps Type: 16 bit data storage register
	Special	256 points	From the range D8000 to D8255 Type: 16 bit data storage register
	Index	2 points	V and Z Type: 16 bit data storage register
Pointers (P)	For use with CALL	64 points	P0 to P63
	For use with interrupts	6 input points and 3 timers	I00❑ to I50❑ and I6☆☆ to I8☆☆ (rising trigger ❑=1, falling trigger ❑=0, ☆☆=time in msec)
Nest levels		8 points for use with MC and MCR	N0 to N7
Constants	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	

8.4 Performance Specification Of The FX (CPU versions from 3.07) And FX2C (all versions)

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: 0.48 μs Applied instructions: several 10's to 100 μs	
Programming language		Relay symbolic language + step ladder	Step ladder can be used to produce an SFC style program
Program capacity		2000 steps built in	Expandable to 8000 steps using additional memory cassette
Number of instructions		Basic sequence instructions: 20 Step ladder instructions: 2 Applied instructions: 96	A Maximum 119 applied instructions are available including all variations
I/O configuration		Max hardware I/O configuration points 255, dependent on user selection (Max. software addressable Inputs 255, Outputs 255)	
Auxiliary relay (M coils)	General	1536 points	M0 to M1535
	Latched	1024 points (subset)	M500 to M1535
	Special	256 points	From the range M8000 to M8255
State relays (S coils)	General	1000 points	S0 to S999
	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)	C219 to C234 Type: 16 bit up/down counter

continued over the page....

Item		Specification	Remarks
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
Data registers (D)	General	1000 points	D0 to D999 Type: 16 bit data storage register pair for 32 bit device
	Latched	800 points (subset)	D200 to D999 Type: 16 bit data storage register pair for 32 bit device
	File registers	2000 points	D1000 to D2999 set by parameter in 4 blocks of 500 program steps Type: 16 bit data storage register
	RAM file registers	2000 points	D6000 to D7999 active when special relay M8074 is active Type: 16 bit data storage register
	Special	256 points	From the range D8000 to D8255 Type: 16 bit data storage register
	Index	2 points	V and Z Type: 16 bit data storage register
Pointers (P)	For use with CALL	128 points	P0 to P127
	For use with interrupts	6 input points, 3timers and 6 counters	I00❑ to I50❑ and I6☆☆ to I8☆☆ (rising trigger ❑=1, falling trigger ❑=0, ☆☆=time in msec)
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	
	Floating Point	32 bit: 0, $\pm 1.175 \times 10^{-38}$, $\pm 3.403 \times 10^{38}$ (Not directly enterable)	

8.5 Performance Specification Of The FX2N(C)

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: 0.08 μs Applied instructions: several 10's to 100 μs	
Programming language		Relay symbolic language + step ladder	Step ladder can be used to produce an SFC style program
Program capacity		8000 steps built in	Expandable to 16000 steps using additional memory cassette
Number of instructions		Basic sequence instructions: 20 Step ladder instructions: 2 Applied instructions: 125	A Maximum 125 applied instructions are available
I/O configuration		Max hardware I/O configuration points 255, dependent on user selection (Max. software addressable Inputs 255, Outputs 255)	
Auxiliary relay (M coils)	General	3072 points	M0 to M3071
	Latched	2572 points (subset)	M500 to M3071
	Special	256 points	From the range M8000 to M8255
State relays (S coils)	General	1000 points	S0 to S999
	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)	C219 to C234 Type: 16 bit up/down counter

Continued over the page....

Item		Specification	Remarks
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
Data registers (D)	General	8000 points	D0 to D7999 Type: 16 bit data storage register pair for 32 bit device
	Latched	7800 points (subset)	D200 to D7999 Type: 16 bit data storage register pair for 32 bit device
	File registers	7000 points	D1000 to D7999 set by parameter in 14 blocks of 500 program steps Type: 16 bit data storage register
	Special	256 points	From the range D8000 to D8255 Type: 16 bit data storage register
	Index	16 points	V0 to V7 and Z0 to Z7 Type: 16 bit data storage register
Pointers (P)	For use with CALL	128 points	P0 to P127
	For use with interrupts	6 input points, 3 timers, 6 counters	I00❑ to I50❑ and I6☆☆ to I8☆☆ (rising trigger ❑=1, falling trigger ❑=0, ☆☆=time in msec)
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	
	Floating Point	32 bit: 0, $\pm 1.175 \times 10^{-38}$, $\pm 3.403 \times 10^{38}$ (Not directly enterable)	

Chapter Contents

9. Assigning System Devices

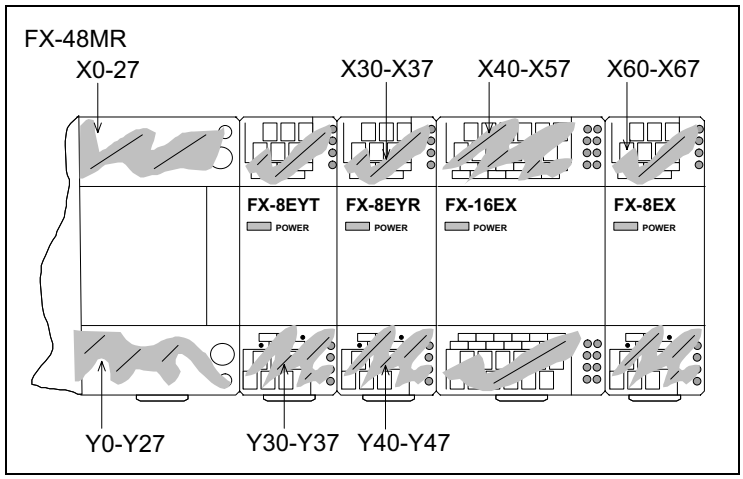
FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

9.1 Addressing Extension Modules

Most of the FX family of PLC's have the ability to connect additional discreet I/O and/or special function modules. To benefit from these additional units the user must address each block independently.

Addressing Additional Discrete I/O

This type of I/O is the standard input and output modules. As each extension block or powered extension unit is added to the system they assume the next available ad-dresses. Hence, the units closest to the base unit will have the lowest I/O numbers or addresses. I/O numbers are always counted in octal. This means from 0 to 7 and 10 to 17 etc. Within a users program the additional addresses are used as normal. Discreet I/O can be added at the users discretion as long as the rules of system configuration for each PLC type are obeyed. This information can be found in the appropriate hardware manual.



For easy use and identification, each additional I/O unit should be labeled with the appropriate I/O numbers using the provided number labels.

Caution when using an FX system with FX-8ER, FX-24MR units

- When an FX-8ER or an FX-24MR are used an additional 8 points (as 4 inputs, 4 outputs) of I/O must be allowed for. This is because both units split blocks of 8 inputs and 8 outputs to obtain a physical 4 input/ 4 output configuration. Hence, an FX-8ER unit actually occupies 8 input points and 8 output points even though there are only 4 physical inputs and 4 physical outputs.

Addressing Special Function Blocks

Special function blocks are allocated a logical 'station/block number' from 0 to 7. This is used by the FROM/TO instructions to directly access each independent special function module. The lower the 'station/block number' is, the closer to the base unit it can be found. Special function blocks can be added at the users discretion but the rules of configuration for each type of PLC must be obeyed at all times. The configuration notes can be found in the appropriate hardware manual for each programmable controller.

9.2 Using The FX2-24EI With F Series Special Function Blocks

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The FX2-24EI allows an FX base unit to be directly connected to an one of the following F series special function blocks:

a) The F-16NT/NP, a Melsec Net Mini interface

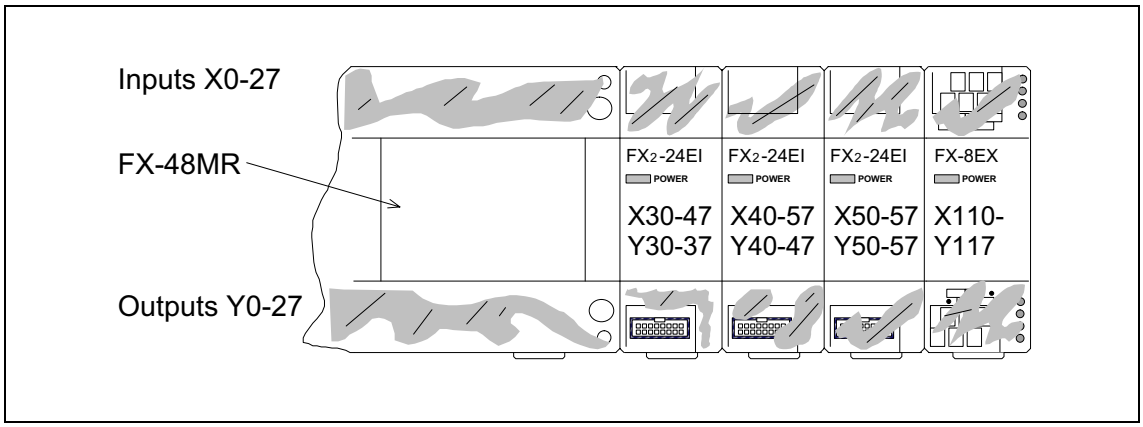
b) The F2-6A, a combine analog 4 input and 2 output unit

c) The F2-32RM, a programmable CAM sequencer

d) The F2-30GM, a pulse train positioning unit

One 24EI unit can control one F series special function unit.

- The '24EI' units are added to an FX system in the same manner as an additional discreet I/O module. Each 24EI occupies 16 input points and 8 output points, the diagram below illustrates this point. Experienced users may notice that the example shown in the diagram below would require too much power from the CPU unit (FX-48MR in this case) based on the I/O count.



This is not the case. The FX2-24EI is really a communications module. It does not directly drive any discreet I/O. For the sake of power calculations it can be assumed that it only occupies 8 standard I/O points. This means that the configuration shown in the diagram actually occupies 32 points (three FX2-24EI's and one FX-8EX) of I/O for the sake of power consumption calculations BUT actually occupies 80 points of addressable I/O.

Connection Of Earth Points When Using F Series Special Function Blocks

- When using the F series special function blocks the special function blocks earth terminal should be connected to the [SG] terminal of the FX CPU unit. However, when using the F-16NP/NT unit the [SG] terminal of the F-16NP/NT should on **NO** account be connected to the [SG] terminal of the FX. The function of these terminals are completely different and are hence **NOT** compatible.

9.2.1 Using the FX2-24EI With A F-16NP/NT

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The F-16NP/NT's operational I/O numbers (addresses) are based upon the position of the associated FX2-24EI within the users FX system. The diagram below shows how moving the position of the FX2-24EI used alters the addresses used by the F-16NP/NT, see EX. 1A to 1C. For installation, wiring and operational details of the F-16NP/NT please see the units dedicated manual. That manual will show examples of the F-16NP/NT being used and in each case the I/O numbers used to address it are those fixed by the F series PLC's. These I/O addresses will be replaced by those described in the previous paragraph, i.e. the FX devices assigned to the FX2-24EI being used.

Worked example:

The following tabular example identifies the correspondence between FX and F2 systems.

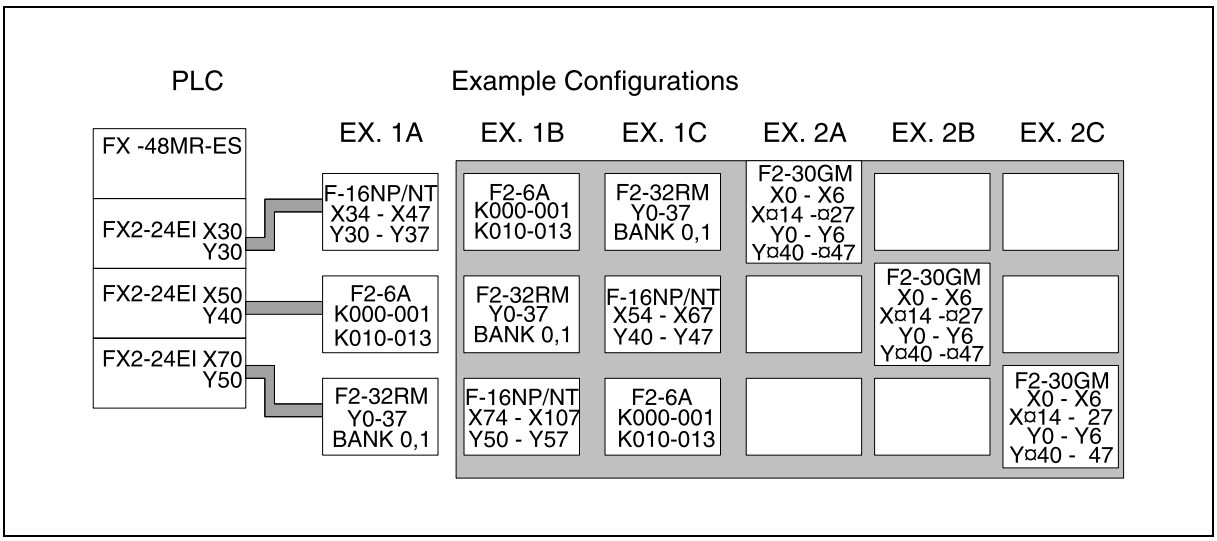
FX System Setup	F2 System Setup	Remark
X5**4**	X41**4**	Input data correct
X5**5**	X41**5**	Input data incorrect
X5**6**	X41**6**	Output data correct
X5**7**	X41**7**	Output data incorrect
X6**0** to X6**7**	X42**0** to X42**7**	Input to PLC
Y4**0** to Y4**7**	Y440 to Y44**7**	Output from PLC

The FX system used is similar to that shown in the diagram EX. 1C. The F2 system uses the second expansion port, i.e. the X400 port of the F series PLC.

Applicable FX Applied Instructions:
(Not applicable to FX2C Main Processing Units or FX units with CPU's greater than version 3.06)

- FNC 90, MNET - used to read and write the Net Mini information

Example configurations of FX2-24EI's and F series special function blocks



9.2.2 Using the FX2-24EI With A F2-6A

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The F2-6A's operational address is based upon the position of the associated FX2-24EI within the users FX system. However, the I/O channel numbers are not affected by this operational address. The I/O channel will always remain as K000 to K001 and K010 to K013. The diagram on the previous page shows how moving the position of the FX2-24EI used alters the operational address but NOT the channel number used by the F2-6A, see EX. 1A to 1C.
For details on using the F2-6A see the units users manual. Please remember when reading the manual, that it has been written for use on an F2 system and hence all of the I/O addresses stated are for such an F2 setup.

F2-6A Channel identification:

- Analog output channels are - K000 and K001 (2 points)
- Analog input channels are - K010, K011, K012 and K013 (4 points)

Applicable FX Applied Instructions:
(Not applicable to FX2C Main Processing Units or FX units with CPU's greater than version 3.06)

- FNC 91, ANRD - used to read the analog data in to the FX
- FNC 92, ANWR - used to write the data from the FX to the F2-6A for output

9.2.3 Using the FX2-24EI With A F2-32RM

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The F2-32RM's operational address is based upon the position of the associated FX2-24EI within the users FX system. However, the "32RM's" I/O numbers are will always remain the same, i.e not be affected by the FX's operational address. The 32RM's I/O numbers would remain as Y0 to Y37, and Bank 0, 1. The diagram on the previous page shows how moving the position of the FX2-24EI used, alters the operational address but NOT the I/O numbers used by the F2-32RM, see EX. 1A to 1C.
For details on using the F2-32RM see the units users manual. Please remember when reading the manual, that it has been written for use on an F2 system and hence all of the I/O addresses stated are for such an F2 setup.

Applicable FX Applied Instructions:
(applicable to all FX and FX2C Main Processing Units)

- FNC 93, RMST - used to output a start code from the FX to the F2-32RM
- FNC 94, RMWR - used to disable outputs on the F2-32RM
- FNC 95, RMRD - used to read the ON/OFF status of the F2-32RM
- FNC 96, RMMN - used to monitor speed and current position of the F2-32RM

9.2.4 Using the FX2-24EI With A F2-30GM

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The F2-30GM's operational address is based upon the position of the associated FX2-24EI within the users FX system. However, the I/O numbers are not directly affected by this operational address. The I/O numbers can be selected by the user. The diagram on the page 9-3 shows how moving the position of the FX2-24EI used alters the operational address but NOT the I/O numbers used by the F2-30GM, see EX. 2A to 2C. Where the symbol § is used in the diagram, this can be replaced by the numbers '0', '4' or '5'. It is recommended that for simplification the user favors '0'.
For details on using the F2-30GM see the units users manual. Please remember when reading the manual, that it has been written for use on an F2 system and hence all of the I/O addresses stated are for such an F2 setup.

Worked example:

The following tabular example identifies the correspondence between FX and F2-30GM systems.

F2-30GM System Setup	Comments	FX System Setup
X❋14 to X❋27	Turn On –	X54 to X67
Y❋40 to Y❋47	Are Turned On BY –	Y40 to Y47
X0 to X6	No correspondence	N/A
Y0 to Y6	No correspondence	N/A

The FX system used is similar to that shown in the diagram EX. 2B from page 9-3.

Applicable FX Applied Instructions:
(Not applicable to FX2C Main Processing Units or FX units with CPU's greater than version 3.06)

- FNC 97, BLK - used to designate a block number within the F2-30GM
- FNC 98, MCDE - used to read an M code number from the F2-30GM

9.3 Parallel Link Adapters

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The FX parallel link adapters provide a means of direct communication between two FX PLC's. There are two models of parallel link adapter providing two different communication mediums:

a) Fiber-optic link - FX2-40AP
Transmission distance: 50m (164 ft)
Fiber-optic: F-OFC-M10 - length 10m (32.8 ft)
F-OFC-M30 - length 30m (98.4 ft)
F-OFC-M50 - length 50m (164 ft)
Note all of the above fiber-optic cables come with connector CA9104AP fitted.

b) Wire link (twisted pair) - FX2-40AW
Transmission distance: 10m (32.8 ft)
Connect like terminals together, i.e [SA] on unit 1 to [SA] on unit 2.
Repeat for [SB] and [SG]. Finally also connect the [SG] terminal of each unit to the [SG] terminal of the local FX PLC.

Special System Devices:

- M8070 - When this is ON the FX PLC is designated 'Master'.
  M8071 - When this is ON the FX PLC is designated 'Slave'.
  M8072 - When this is ON there is communication between stations.
  M8073 - When this is ON there is a 'Master/Slave' designation error.
  M8063 - When this is on there is a link error - see error tables for further details

Applicable FX Applied Instructions:

- FNC 81, PRUN - Identifies which input devices are used in the data exchange.

Transmission Devices:

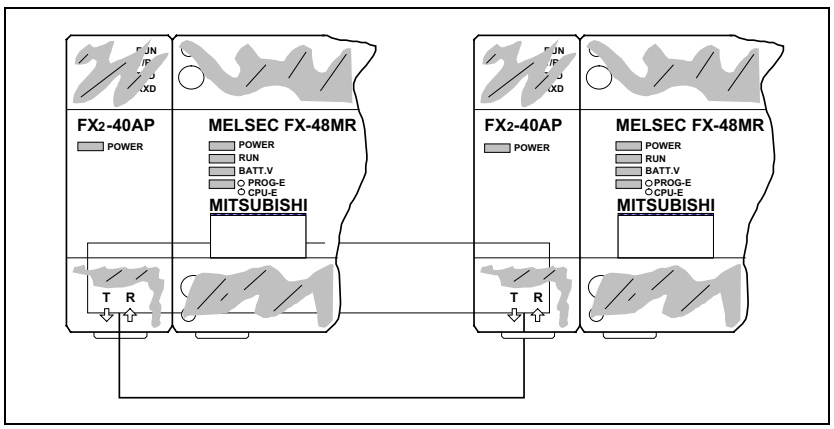
Master ➤ Slave, Slave ➤ Master

Bit devices -100 points (M800 to 899)	Bit devices - 100 points (M900 to 999)
Word devices - 10 points (D490 to 499)	Word devices - 10 points (D500 to 509)

Communication Time:

- 70 msec + (master and slave station cycle times)

General System Layout:



9.4 Real Time Clock Function

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

When one of the real time clock (RTC) memory cassettes is used with either and FX or an FX0N, the real time clock function of that PLC is then automatically enabled.
The time data of the RTC cassette is battery backed. This means when the PLC is turned OFF the time data and settings are not lost or corrupted. The duration or storage life of the time data is dependent upon the condition of the battery. The operation of the RTC clock circuits consumes negligible currents compared with the RAM memory (applicable to FX only).
The real time clock cassettes have a worst case accuracy of ± 45 seconds per month at an ambient temperature of 25°C. The calendar function of the RTC cassettes caters for leap years during the period 1980 through 2079.

FX PLC's Pre Version 2.0

- These PLC's do not support the RTC function.

Available RTC Cassettes:

RTC Cassette	Remark
FX-RTC	Real time clock function only
FX-RAM-8C	Real time clock function + 8K RAM (Note: the FX0N cannot use this cassette, because RAM program storage is not guaranteed)
FX-EEPROM-4C	Real time clock function + 4K EEPROM (Note the FX0N can only use 2K of the EEPROM)

Function Registers And Control Flags:

- Please see page 6-3 for a list of the available clock devices.
- Please see section 5.14 (page 5-138) for RTC supporting functions with FX2N.

General Use Of Memory Cassettes

- FX and FX2C Main Processing units can use any of the available memory cassette types. FX0N units can only use EEPROM or EPROM type Memory Cassettes and then only a maximum of 2k steps can be accessed.

- It is necessary to install a FX0N-40BL battery to maintain the RTC time with FX0N controllers.

- The FX2N has a real time clock built in to the MPU. Although it is possible to use the RTC cassettes for memory, it is not necessary to use them to obtain the RTC function.

9.4.1 Setting the real time clock

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

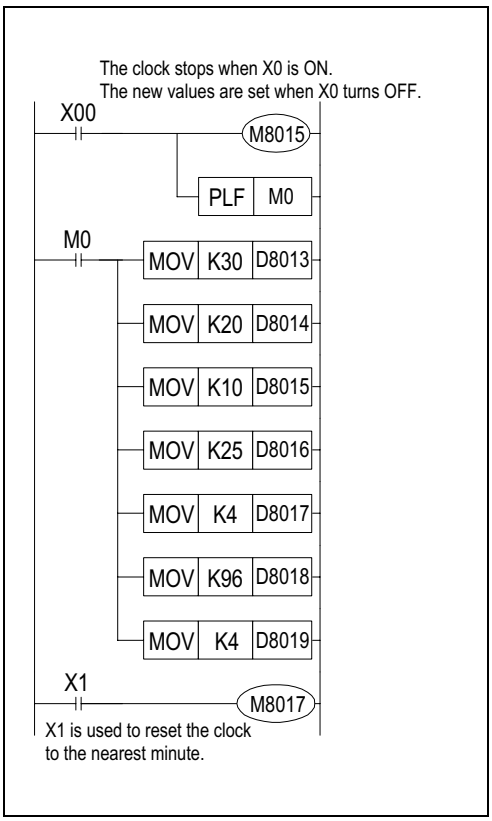
The RTC can be set using the special data registers and control flags as follows:

Device Number	Function	Range
D8013	Seconds	0 to 59
D8014	Minutes	0 to 59
D8015	Hours	0 to 23
D8016	Date	1 to 31 (correct for current Month)
D8017	Month	1 to 12
D8018	Year	00 to 99 (1980 to 2079)
D8019	Day of Week	0 to 6 (Sunday to Saturday

Device Number	Comments
M8015 Time setting	Set ON to stop the clock. When the clock is stopped the time values can be reset. The clock restarts when the flag is reset to OFF.
M8016 Register Hold	The clock data in the data registers is held. The clock still runs. Use this to pause the data to read the current time.
M8017 Minute Rounding	When on rounds the time up or down to the nearest minute.
M8018 Clock Available	Automatically set to indicate the RTC is available.
M8019 Setting Error	ON when the values for the RTC are out of range.

These devices are used as shown in the program on the right.

Note: The FX2N(C) has special instructions that simplify the setting and use of the RTC.
See section 5.14 for more details.



Chapter Contents

10. Points Of Technique

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

10.1 Advanced Programming Points

The FX family of programmable controllers has a very easy to learn, easy to use instruction set which enables simple programs to perform complex functions. This chapter will point out one or two useful techniques while also providing the user with valuable reference programs.

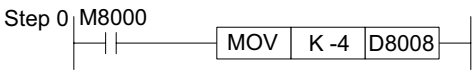
If some of these techniques are applied to user programs the user must ensure that they will perform the task or operation that they require. Mitsubishi Electric can take no responsibility for user programs containing any of the examples within this manual.

Each program will include a brief explanation of the system. Please note that the method of 'how to program' and 'what parameters are available' for each instruction will not be discussed. For this information please see the relevant, previous chapters.

10.2 Users of DC Powered FX Units

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

When using DC powered FX programmable controllers, it is necessary to add the following instructions to the beginning of the installed program:



Explanation:

With AC powered FX programmable controllers, the power break detection period can be adjusted by writing the desired detection period to the special data register D8008.
However, in the case of DC powered units this detection period must be set to 5 msec.

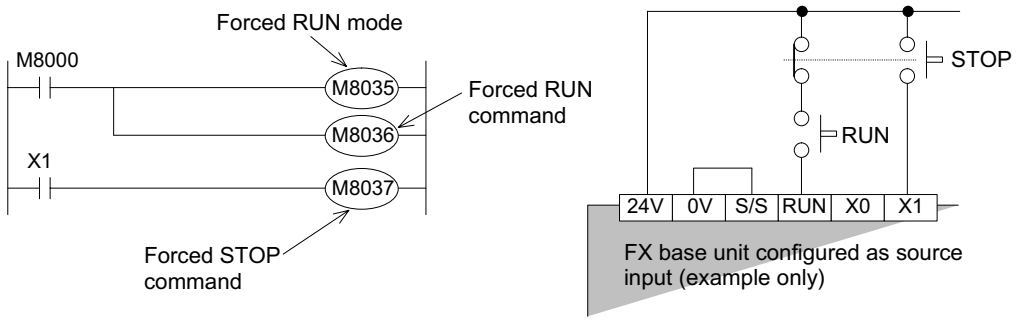
This is achieved by moving the value of -4 into D8008. Failure to do this could result in inputs being missed during the DC power 'drop'.

10.3 Using The Forced RUN/STOP Flags

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

10.3.1 A RUN/STOP push button configuration

The FX programmable controller has a single RUN terminal. When power is applied to this terminal the PLC changes into a RUN state, i.e. the program contained is executed. Consequently when there is no power 'on' the RUN terminal the PLC is in a STOP state. This feature can be utilized to provide the FX PLC with an external RUN/STOP - push button control. The following PLC wiring and program addition are required.



Explanation:

Pressing the RUN push button sets the PLC into the RUN state. This means M8000 is ON. Following the program, M8000 activates both M8035 and M8036. These two special auxiliary devices set the PLC in to forced RUN mode. Releasing the RUN push button would normally return the PLC to the STOP state, but because the two auxiliary coils, M8035 and 36 are ON, the PLC remains in RUN. To stop the, PLC pressing the STOP push button drives an input ON and consequently M8037 turns ON. This then automatically forces OFF both M8035 and 36 and resets itself. Hence, the PLC is in its STOP status and awaits the cycle to begin again.

Input priority:

- The STOP input is only processed after the programs END statement has been reached - this is because the physical input used, i.e. an X device is normally updated and processed at that time. Therefor, the RUN input is given priority when both RUN and STOP inputs are given simultaneously.
- To give priority to the STOP input and provide a 'safer' system, some form of mechanical/ circuitry interlock should be constructed between both RUN and STOP inputs. A very simple example is shown in the wiring diagram above.

- For push-button control to operate correctly, the user must set the RUN/STOP switch on FX2C and FX2N(C) units to the STOP position.
- FX2N(C) units do not have a RUN terminal. One of the inputs X0 to X17 (X0 to X7 for FX2N-16M) on the MPU should be configured as a RUN terminal in the parameter settings.

10.3.2 Remote RUN/STOP control

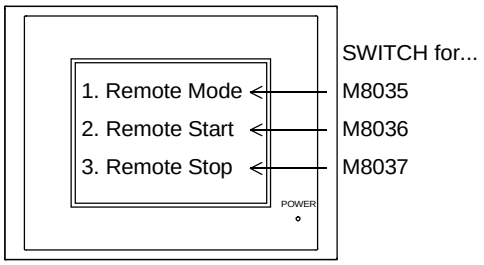
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The FX family of programmable controllers can be controlled, i.e. switched into RUN or STOP modes and have devices monitored by use of intelligent external control devices.
These includes such items as computers, the Mitsubishi FX data access units and Graphic Operator Terminals.
The following example utilizes a graphic FX-DU unit:

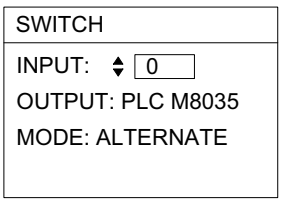
Explanation:

The programmable controller needs no special wiring or additional programming for this example.

The only condition required is that the PLC would not normally be in a RUN state, i.e., there is no connection to the RUN terminal and the RUN/STOP switch on PLC's that have one is set in the STOP position.



The DU should be programmed with 'SWITCH' devices driving the three special M codes M8035,36 and 37. By activating the 'SWITCH' devices for M8035 and M8036 the PLC can be switched into a RUN state, while driving the 'SWITCH' device M8037 will put the PLC into a STOP state.



Example 'SWITCH' device setting opposite.
Use an 'Alternate' switch for M8035 and M8036 and use a 'Momentary' switch for M8037.
(see DU operation manual for SWITCH operation and programming)

Note: While M8035 and M8036 are ON the MPU can not be changed to STOP mode using the RUN terminal or RUN/STOP switch. Either set M8037 ON, or reset M8035 and M8036, to return to the normal operating state.

Range of Mitsubishi graphic FX-DU units:

FX-25DU-E - a 4 line text/graphic unit.
FX-30DU-E - a 4 line text/graphics display unit with membrane style keypad.
FX-40DU-TK-E - a 7 line, touch key, text/graphics display unit with numeric keypad.
FX-50DU-TK(S)-E - a 15 line, touch key, color text/graphics display unit.
F940GOT-SWD/LWD-E - a 15 line, touch key, color text/graphics advanced display unit.

FX2N(C) Remote STOP

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

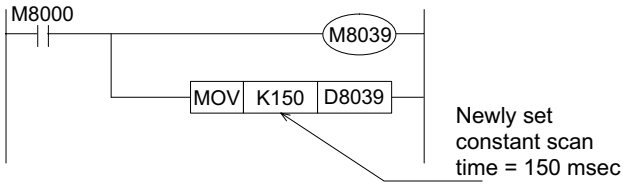
With FX2N(C) units, even if the RUN terminal or RUN/STOP switch is in the RUN position, it is still possible to do a remote STOP by forcing M8037 ON.

Return to RUN by resetting M8037.

10.4 Constant Scan Mode

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Some times the timing of operations can be a problem, especially if some co-ordination is being attempted with a second control system. In cases like this it is very useful to fix the PLC's scan time. Under normal conditions the PLC's scan time will vary from one scan to the next. This is simply because the natural PLC scan time is dependent on the number of and type of the active instructions. As these are continually changing between program scans the actual scan time is also a varying. Hence, by using the additional program function identified below, the PLC's scan time can be fixed so that it will be the same duration on every program scan. The actual scan duration is set by writing a scan time in excess of the current longest scan duration to special data register D8039 (in the example the value K150 is used). If the PLC scans the program quicker than the set scan time, a 'pause' will occur until the set scan duration is reached.
This program example should be placed at the beginning of a users program.

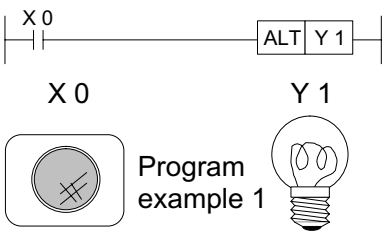


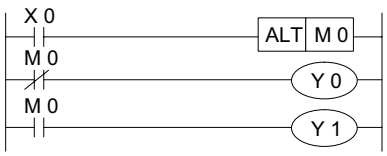
10.5 Alternating ON/OFF States

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

It is often useful to have a single input control or toggle a situation. A basic, yet typical example is the switching ON/OFF of a Light. This can be easily achieved by using standard ladder program to load an input and switch an output. However, this system requires an input which is latchable. If basic ladder steps are used to latch the program then it soon becomes complex and prone to mis-programming by the user. Using the ALT instruction to toggle the ON/OFF (SET/RESET, START/STOP, SLOW/FAST) state is much simpler, quicker and more efficient.

Explanation:





Program example 2

Pressing the momentary push button X1 once will switch the lamp ON. Pressing the push button for a second time will cause the lamp to turn OFF. And if the push button is again pressed for a third time, the lamp is turned ON again and so the toggled status continues.
The second program shown identifies a possible motor interlock/control, possibly a start/stop situation.

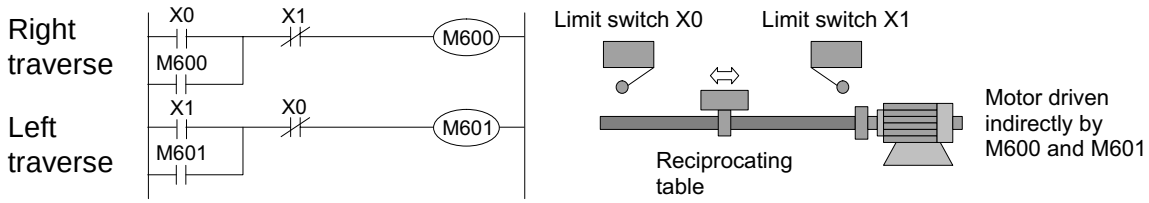
10.6 Using Battery Backed Devices For Maximum Advantage

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Battery backed devices retain their status during a PLC power down. These devices can be used for maximum advantage by allowing the PLC to continue from its last operation status just before the power failure.

For example: A table traverse system is operating, moving alternatively between two limit switches. If a PLC power failure occurs during the traversing the machine will stop.
Ideally, once the PLC regains its power the system should continue from where it left off, i.e. if the movement direction was to the left before the power down, it should continue to the left after the restoration of the power.

Explanation:



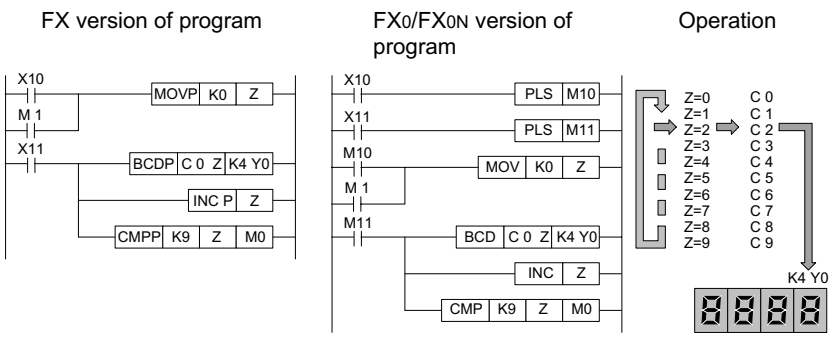
The status of the latched devices (in this example FX M coils M600 and M601) is retained during the power down. Once the power is restored the battery backed M coils latch themselves in again, i.e. the load M600 is used to drive M600.

10.7 Indexing Through Multiple Display Data Values

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Many users unwarily fall in to the trap of only using a single seven segment display to display only a single data value. This very simple combination of applied instructions shows how a user can 'page' through multiple data values displaying each in turn.

Explanation:

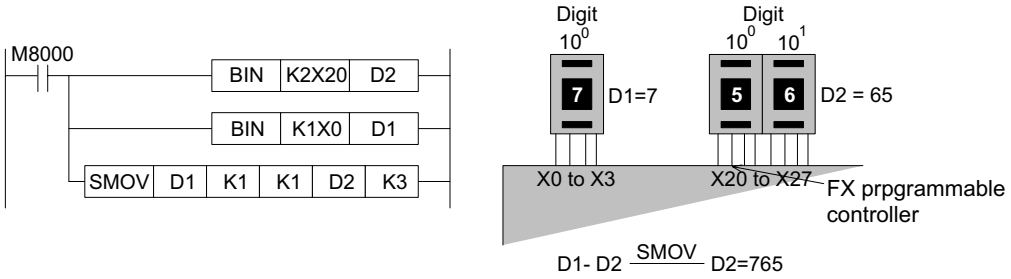


The contents of 10 counters are displayed in a sequential, 'paged' operation.
The paging action occurs every time the input X11 is received.
What actually hap-pens is that the index register Z is continually incremented until it equals 9. When this happens the comparison instruction drives M1 ON which in turn resets the current value of Z to 0 (zero). Hence, a loop effect is created with Z varying between fixed values of 0 and 9 (10 values). The Z value is used to select the next counter to be displayed on the seven segment display.
This is because the Z index modifier is used to offset the counter being read by the BCD output instruction.

10.8 Reading And Manipulating Thumbwheel Data

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Data can be easily read into a programmable controller through the use of the BIN instruction. When data is read from multiple sources the data is often stored at different locations. It may be required that certain data values are combined or mixed to produce a new value. Alternatively, a certain data digit may need to be parsed from a larger data word. This kind of data handling and manipulation can be carried out by using the SMOV instruction. The example below shows how two data values (a single digit and a double digit number) are combined to make a final data value.



Explanation:

The two BIN instructions each read in one of the data values. The first value, the single digit stored in D1, is combined with the second data value D2 (currently containing 2 digits). This is performed by the SMOV instruction. The result is that the contents of D1 is written to the third digit of the contents of D2. The result is then stored back into register D2.

10.9 Measuring a High Speed Pulse Input

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

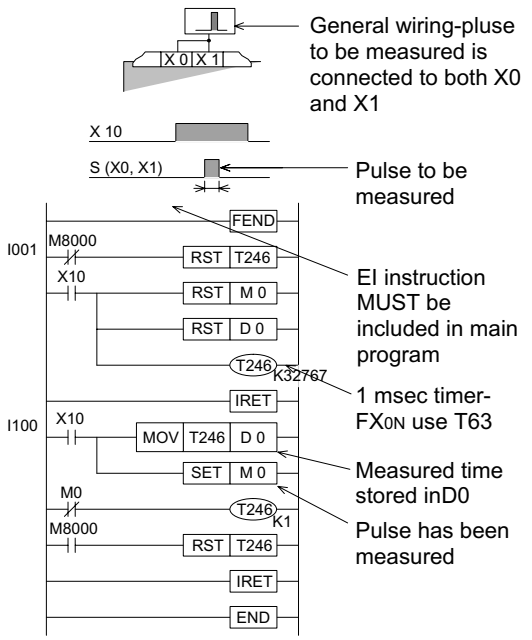
10.9.1 A 1 msec timer pulse measurement

Some times due to system requirements or even as a result of maintenance activities it is necessary to 'find out' how long certain input pulses are lasting for. The following program utilizes two interrupt routines to capture a pulse width and measure it with a 1 msec timer. The timer used in the ample is one of the FX timers. However, T63 on the FX0N would be used for a similar situation on that PLC.

Explanation:

The 1 msec timer T246 is driven when interrupt I001 is activated. When the input to X1 is removed the current value of the timer T246 is moved to data register D0 by interrupt program I100. The operation complete flag M0 is then set ON.

Note: X10 acts as an enable/disable flag.



10.9.2 A 0.1 msec timer pulse measurement

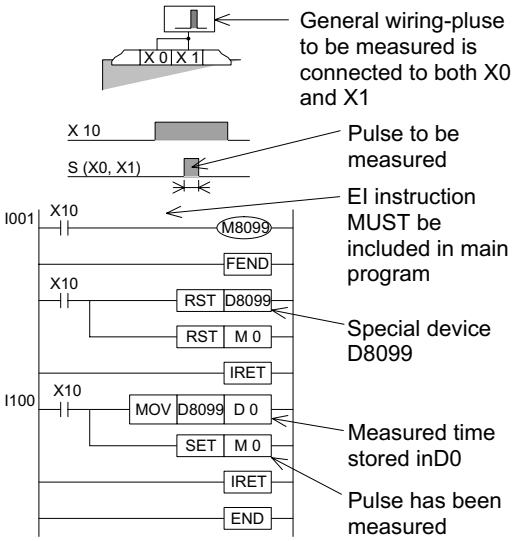
FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

This is a very accurate measuring process for pulse inputs. The use of a standard timer is not accurate enough in this case as the highest resolution is 1msec. Therefor, this example shows how the special high accuracy devices M8099 and D8099 are used to capture the 0.1 msec resolution pulse data.

Explanation:

The incoming pulse is captured between two interrupt routines. These routines operate independently of each other, one on the rising edge of the pulse input and one on the falling edge of the same input. During the pulse input the contents of special register D8099 are continually moved into data register D0. Once the pulse has completed the contents of D0 can be viewed at leisure.

Please note for this high speed/accuracy mode to be active for D8099, the corresponding special auxiliary bit device M8099 must be driven ON in the main program.



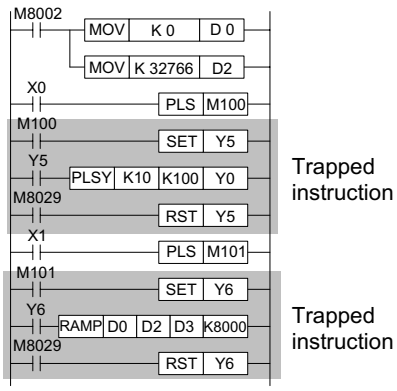
10.10 Using The Execution Complete Flag, M8029

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

Some of the applied instructions take more than one program scan to complete their operation. This makes identification of the current operating state difficult. As an aid to the programmer, certainappliedinstructionsidentify theircompletionbysettinganoperationcompleteflag, M8029. Because this flag can be used by several different instructions at the same time, a method similar to the following should be used to trap the M8029 status at each of the instructions using it:

Explanation:

The M8029 'trapping' sequence takes advantage of the batch refresh of the FX family of PLC's. As the program scan passes each instruction using M8029 the status of M8029 changes to reflect the current status of the instruction. Hence, by immediately resetting (or setting) the drive flag for the instruction the current operational status of the instruction is trapped. So when the batch refresh takes place only the completed instructions are reset. The example above uses a pulse to set the drive flags so that it is easy to monitor and see when each instruction finishes (if the instructions are continuously driven it will be difficult to see when they finish!).



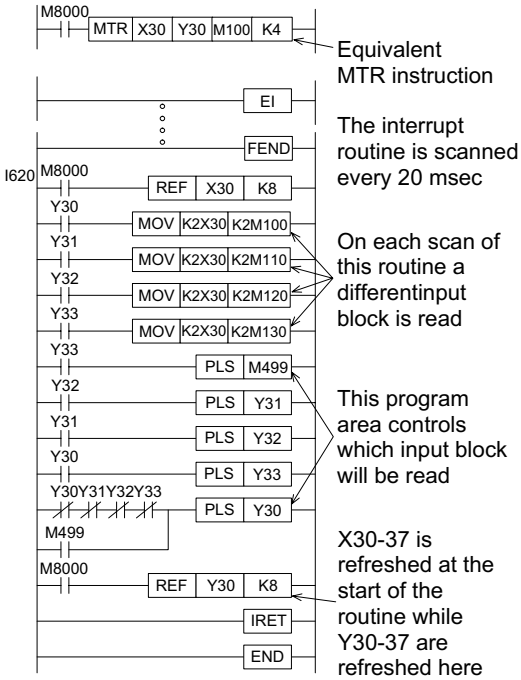
10.11 Creating a User Defined MTR Instruction

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

For users who want to have the benefits of the MTR instruction for FX users who want to specify more than one MTR area, this user defined MTR function will be very useful.

Explanation:

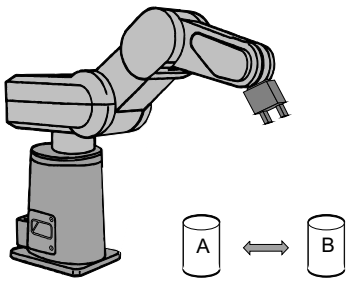
The main control of this program rests in the timer interrupt I620. This interrupt triggers every 20msec regardless of what the main program is doing. On each interruption one bank of the user defined matrix is read. The program simply consists of reading the inputs triggered by each of the multiplexed outputs.
The read data is then stored in sequential sets of auxiliary registers.
Each MOV instruction reads a new bank of multiplexed inputs.
The equivalent MTR instruction is shown immediately before the 'user defined' MTR.
See the MTR instruction on page 5-54 for more details.



10.12 An Example System Application Using STL And IST Program Control

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The following illustration shows a simple 'pick and place' system utilizing a small robotic arm. The zero point has been de-fined as the uppermost and left most position accessible by the robot arm.



A normal sequence of events

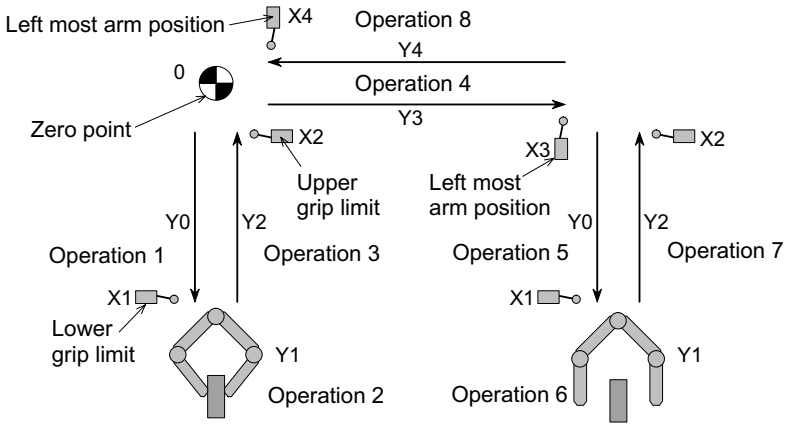
A product is carried from point 'A' to point 'B' by the robot arm. To achieve this operation the following sequence of events takes place:
Initial position: the robot arm is at its zero point.

1) The Robots grip is lowered to it lowest limit
   - output Y0: ON, input X1: ON, output Y0: OFF.
2) The grip clamped around the product at point A
   - output Y1: ON.

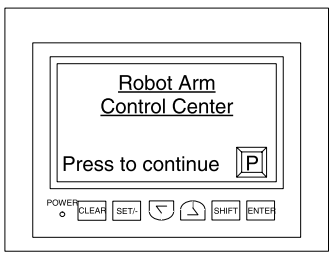
3) The grip, now holding the product, is raised to its upper limit
   - output Y2: ON, input X2: ON, output Y2: OFF.
4) The robot arm traverses to its right most position
   - output Y3: ON, input X3: ON, output Y3: OFF.
5) The grip and product are lowered to the bottom limit
   - output Y0: ON, input X1: ON, output Y0: OFF.
6) The grip is unclamped and the product is released at point B
   - output Y1: OFF.
7) The grip is retrieved back to its upper limit
   - output Y0: ON, input X2: ON, output Y0: OFF.
8) The arm traverses back to its zero point by moving to the left most limit
   - output Y4: ON, input X4: ON, output Y4: OFF.

The cycle can then start again.

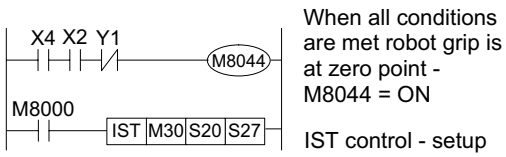
System parameters



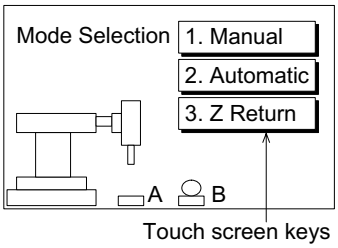
1) Double solenoid valves are used to control the up (Y2)/down (Y0) and right (Y3)/left (Y4) motion.
2) A single solenoid valve is used for the clamp (Y1)/unclamp operation.
3) The system uses an FX-40DU-TK to interface with the operator.
   The FX-40DU-TK is a touch screen data access unit.



This example uses the IST instruction (FNC 60) to control the operation mode of the robot arm. The program shown opposite identifies how the IST instruction is written into the main program.



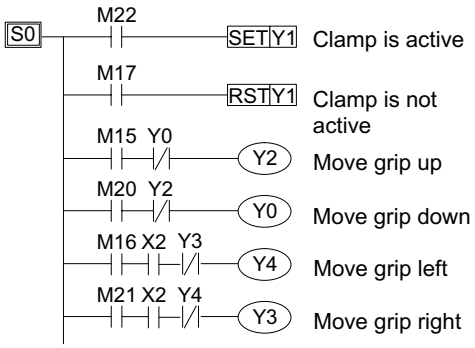
When the IST instruction is used there are 5 selectable modes which access three separate programs. This example has the following programs associated with its modes. Each mode is selected through the FX-40DU-TK. The screen shown opposite is the initial mode menu. Each of the menu options causes a screen jump to the selected mode. Menu options 1 and 3 also set ON auxiliary devices M30 and M31 respectively.
The active bits then trigger a screen change to the selected mode. Please note 'Automatic' has three further modes which are selected from a following screen/display.



An example DU screen design

Manual Mode:

In this mode ALL operations of the robot arm are controlled by the operator. An operation or movement is selected by pressing the corresponding option on the DUs screen (see below). These options then trigger DU SWITCH objects which drive associated auxiliary relays within the programmable controller. The SWITCH objects should be set to momentary so that they only operate when the key is pressed.



The status of the clamping action could be identified by two INDICATOR (SCR) functions on the DU unit. They could be monitoring the ON and OFF status of the clamp output Y1. Hence, when the clamp was ON a single black box opposite the ON button could appear. When the clamp is OFF the box would appear in front of the OFF button. At any one time only one box would be active.

Key assignment for DU screen opposite:

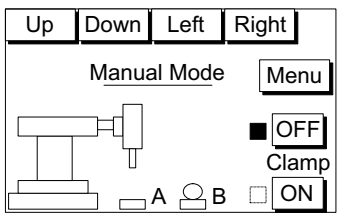
Up = M15 Down = M20

Left = M16 Right = M21

Clamp ON = M22

Clamp OFF = M17

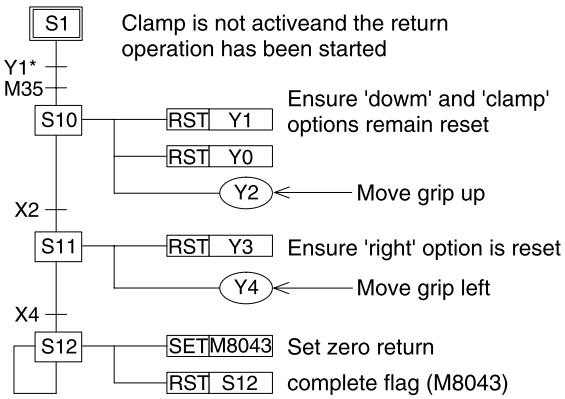
Menu = reset M30



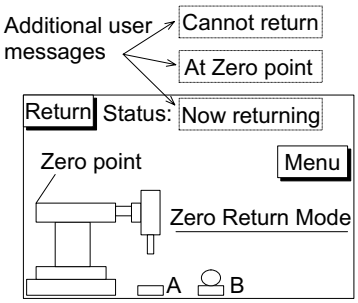
Once manual operation is completed the operator can return to the main mode selection screen by touching the 'Menu' key. This causes the manual mode bit flag, M30, to be reset. Once M30 is reset the DU screen then changes back to the desired mode selection screen.

Zero Return Mode

This mode fulfills an initialization function by returning the robot arm to a known position. Once 'Z Return' has been selected from the mode selection screen the bit device M35 is ON. At this point the DU screen changes to the 'zero return' screen.
The actual zero return operation will then start when the 'Return' push button is pressed (activating M25) and the robots grip is not active, i.e. Y1 is OFF (on the STL flow diagram opposite Y1 OFF is shown as Y1*).



The DU unit could be used to report back the status of the current returning operation. The example screen shown opposite uses 3 variable messages to indicate this status. The messages could be text strings stored in the PLC which are read and displayed by the DUs ASCII option.



Once the zero point has been returned to, the operator would also return to the mode selection screen. This is achieved by pressing the 'Menu' touch key. This then resets the zero return bit device M31 which allows the DU screen change to take place.

Key assignment for DU screen above:
Return = M25
Menu = reset M31

Automatic Mode

Under this option there are three further mode selections. The available modes are:

Step Mode:

- The automatic program is stepped through - operation by operation, on command by the user pressing the 'Start' button.

Cycle Mode:

- The automatic program is processed for one complete operational cycle. Each cycle is initiated by pressing the 'Start' button. If the 'Stop' button is pressed, the program is stopped immediately. To resume the cycle, the 'Start' button is pressed again.

Automatic Mode:

- A fully automatic, continuously cycling mode. The modes operation can be stopped by pressing the 'stop' button. However, this will only take effect after completion of the current cycle.

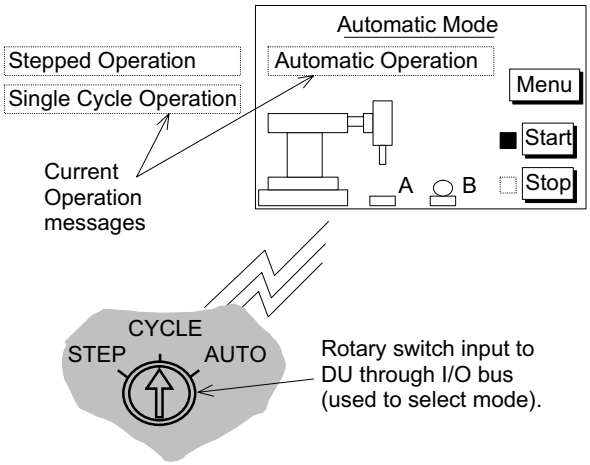
In this example these three modes are selected by an external rotary switch. The rotary switch is not connected to the PLC but to the I/O bus on the rear of the DU unit.
The use of the rotary switch means that the selected modes are mutually exclusive in their operation. For an operator friendly environment the currently selected mode is displayed on the DU screen (again this could be by use of the DUs ASCII function).
The start/ stop controls are touch keys on the DU screen. When a mode is selected the input received at the DU unit momentarily activates one of the following auxiliary relays:
Rotary switch:
position 1 'Step' - Step operation: DU input I0, controls bit device M32 position 2 'Cycle' - Single cycle operation:
DU input I1, controls bit device M33 position 3 'Auto' - Automatic operation: DU input I2, controls bit device M34

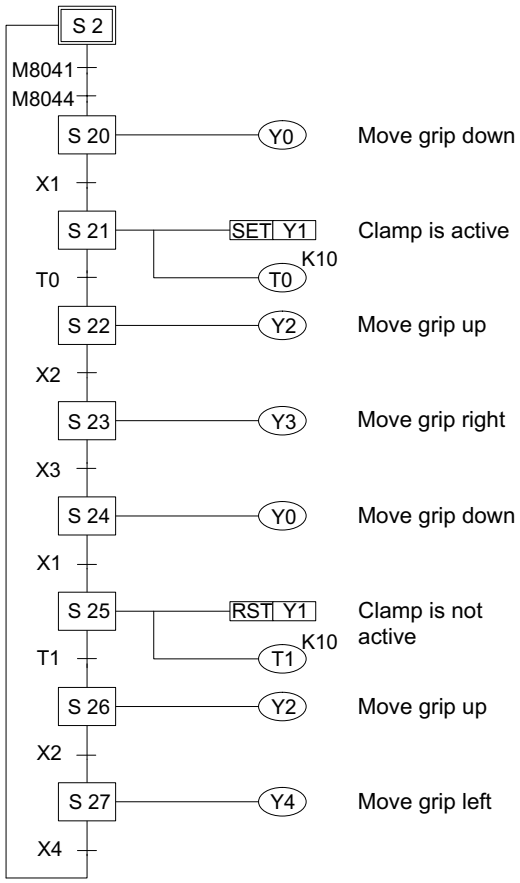
Key assignment for DU screen above:
Start = M36
Stop = M37

The program run in all three mode choices is shown opposite. As noted earlier, the 'Step' mode will require an operator to press the 'Start' key to start each new STL block. This could be viewed as an additional transfer condition between each state. However, the user is not required to program this as the IST instruction controls this operation automatically.

The 'Cycle' mode will process the program from STL step S2, all the way through until STL step S2 is encountered again. Once more the IST instruction ensures that only one cycle is completed for each initial activation of the 'Start' input.

Finally as suggested by the name, 'Auto' mode will continuously cycle through the program until the 'Stop' button is pressed. The actual halting of the program cycling will occur when the currently active cycle is completed.





Points of interest:

a) Users of the IST instruction will be aware that only one of the operation modes should be active at one time. In this example program the isolation of 'Manual' and 'Zero return' modes by the use of separate DU control screens, and the use of a rotary switch to isolate the three automatic modes achieves this objective. Alternatively all of the operation modes could be selected by a rotary switch.

b) For users who would like to test this example using simulator switches (i.e., without using a data access unit) the appropriate program changes are noted next to the full program listing later in this section. Alternatively, the original program could be used with all of the input conditions being given by forcing ON the contacts with a programming device e.g. a hand held programmer, Medoc etc.

c) Special flags used in this program are:

- M8040: State transfer inhibit
  - Manual mode: Always ON.
    Zero return and Cycle modes: Once the 'Stop' input is given the current state is retained until the 'Start' input is received.
    Step mode: This flag is OFF when the 'Start' input is ON. At all other times M8040 is ON, this enables the single STL step operation to be achieved.
    Auto mode: M8040 is ON initially when the PLC is switched into RUN. It is reset when the 'Start' input is given.
- M8041: State transfer start
  - Manual and Zero return modes: This flag is not used.
    Step and Cycle modes: This flag is only active while the 'Start' input is received.
    Auto mode: The flag is set ON after the 'Start' input is received. It is reset after the 'Stop' input is received.
- M8042: Start pulse
  - This is momentarily active after the 'Start' input is received.
- M8043: Zero return complete
  - This is a user activated device which should be controlled within the users program.
- M8044: At Zero position/ condition
  - This is a user activated device which should be controlled within the users program.

Full program listing:

0	LD	X	4	35	STL	S	1	72	STL	S	21
1	AND	X	2	36	LD	M	35	73	SET	Y	1
2	ANI	Y	1	37	RST	M	8043	74	OUT	T	0
3	OUT	M	8044	39	ANI	Y	1			K	10
5	LD	M	8000	40	SET	S	10	77	LD	T	0
6	IST		60	42	STL	S	10	78	SET	S	22
		M	30	43	RST	Y	1	80	STL	S	22
		S	20	44	RST	Y	0	81	OUT	Y	2
		S	27	45	OUT	Y	2	82	LD	X	2
13	STL	S	0	46	LD	X	2	83	SET	S	23
14	LD	M	8044	47	SET	S	11	85	STL	S	23
15	OUT	M	8043	49	STL	S	11	86	OUT	Y	3
17	LD	M	22	50	RST	Y	3	87	LD	X	3
18	SET	Y	1	51	OUT	Y	4	88	SET	S	24
19	LD	M	17	52	LD	X	4	90	STL	S	24
20	RST	Y	1	53	SET	S	12	91	OUT	Y	0
21	LD	M	15	55	STL	S	12	92	LD	X	1
22	ANI	Y	0	56	SET	M	8043	93	SET	S	25
23	OUT	Y	2	58	RST	S	12	95	STL	S	25
24	LD	M	20		(RET)*			96	RST	Y	1
25	ANI	Y	2	60	STL	S	2	97	OUT	T	1
26	OUT	Y	0	61	LD	M	8041			K	10
27	LD	M	16	62	RST	M	8043	100	LD	T	1
28	AND	X	2	64	AND	M	8044	101	SET	S	26
29	ANI	Y	3	65	SET	S	20	103	STL	S	26
30	OUT	Y	4	67	STL	S	20	104	OUT	Y	2
31	LD	M	21	68	OUT	Y	0	105	LD	X	2
32	AND	X	2	69	LD	X	1	106	SET	S	27
33	ANI	Y	4	70	SET	S	21	108	STL	S	27
34	OUT	Y	3					109	OUT	Y	4
	(RET)*							110	LD	X	4
	↑			This instruction returns the program flow to STL step S2.				→	**111**	**OUT**	**S**	**2**
*: Instructions in ( ) are not necessary									113	RET		
necessary									114	END		

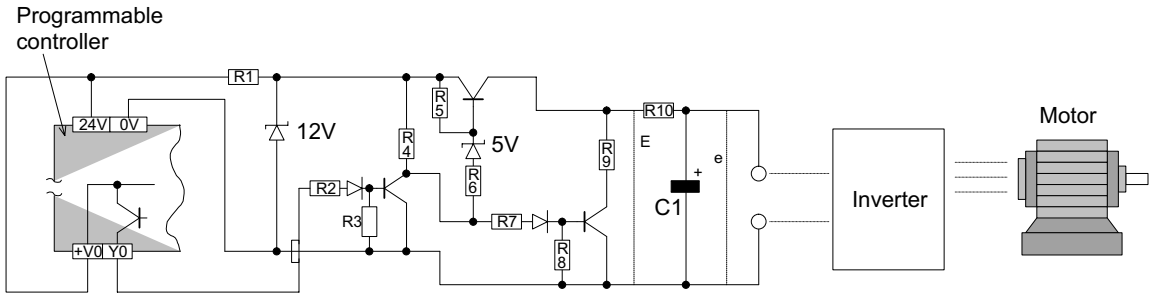
Program options:

6	IST		60	17	LD	**X**	**12**	27	LD	**X**	**6**
		X	**20**	19	LD	**X**	**7**	31	LD	**X**	**11**
		S	20	21	LD	**X**	**5**	36	LD	**X**	**25**
		S	27	24	LD	**X**	**10**				

10.13 Using The PWM Instruction For Motor Control

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The PWM instruction may be used directly with an inverter to drive a motor. If this configuration is used the following ripple circuit will be required between the PLC's PWM output and the inverters input terminals.



Circuit configuration for a PLC with source outputs

Key to component values:

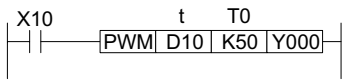
R1 - 510 Ω (1/2 W) R2 - 3.3kΩ (1/2 W)

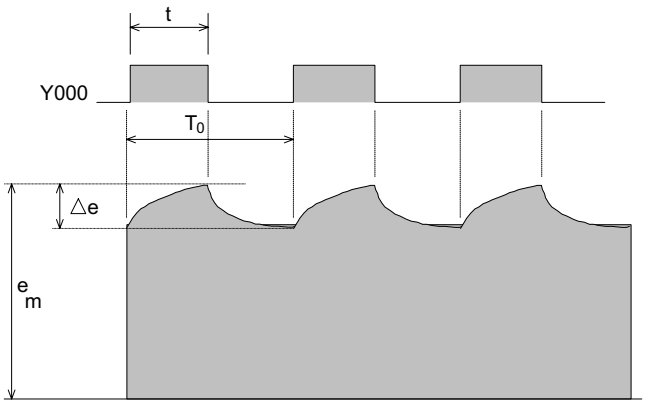
R3 to R8 - 1kΩ (1/4 W) R9 - 22 Ω (1/4 W)

R10 - variable dependent on configuration. In this example 1kΩ (1 W)

C1 - 470 μF

Note: the values of R10 and C1 are dependent on the system configuration.





Establishing system parameters and values

It is assumed that the input impedance of the inverter is of a high order. Having established this, the values of C1 and R10 are calculated to give τ a time result (in msec) approximately 10 times bigger than the value used for T0 in the PWM instruction:

τ = R10 (kΩ) ÅL C1 (μF)

During this calculation the value of R10 must be vastly greater than the value of R9. In the example, R9 is equal to 22Ω, where as R10 is equal to 1kΩ. This proportion is approximately 1:50 in favor of R10.

The maximum output voltage (to the inverter) including ripple voltage, can be found by using the following equation:

$$e_m \approx E \frac{t}{T_0}$$

Where:

e_m = Maximum output voltage

E= pulse (square wave) output voltage (see circuit on the previous page)

t = PWM pulse duration (see previous page for reference)

T_0 = PWM cycle time for pulse (see previous page for reference)

The average output voltage (to the inverter) including ripple voltage, can be found by using the following equation:

$$\frac{\Delta e}{e} \approx \frac{T_0 - t}{\tau} \leq \frac{T_0}{\tau}$$

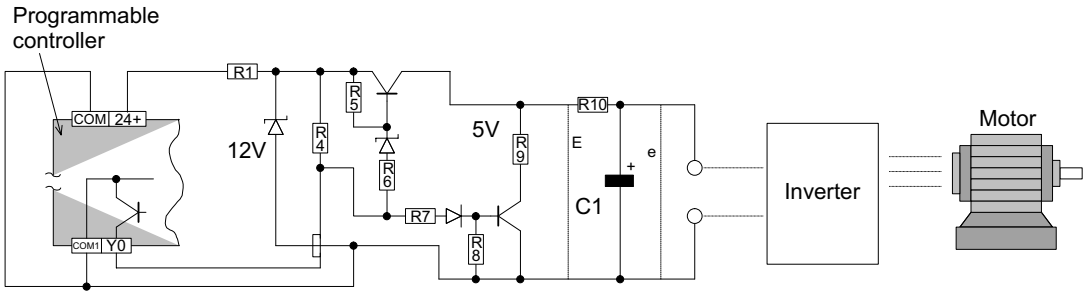
Where:

Δe = the voltage value of the ripple
e = ripple output voltage
T_0 = PWM cycle time for pulse
t = PWM pulse duration
τ = ripple circuit delay

See previous page for references.

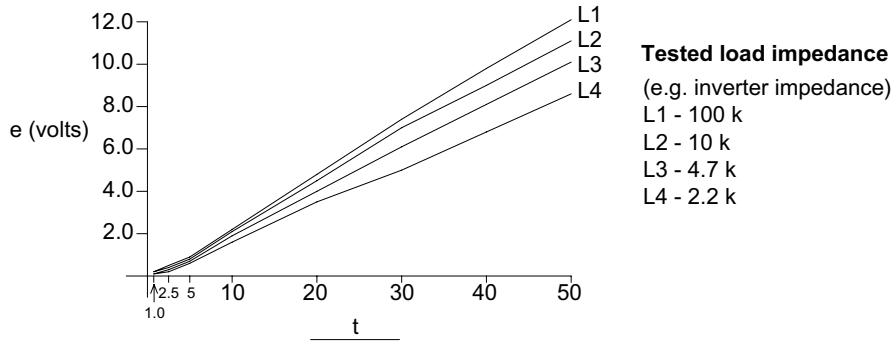
Operation

Once the system configuration has been selected and the ripple circuit has been built to suit, the motor speed may be varied by adjusting the value of 't' in the PWM instruction.
The larger the value of 't' the faster the motor speed will rotate. However, this should be balanced with the knowledge that the faster the output signal changes the greater the ripple voltage will be. On the other hand a slowly changing output signal will have a more controlled, yet smaller ripple effect. The speed of the signal change is determined by the size of C1. A large capacitive value for C1 would give a smaller ripple effect as charge is stored and released over a longer time period.



Circuit configuration for a PLC with sink outputs.
The component values are the same as stated previously

The following characteristics were noticed when the identified circuit was tested
The PWM instruction had T_0 set to K50. The value for t was varied and also the load impedance was varied to provide the following characteristics graph (see over page).



The duration of the T_0, time base also affects the ripple voltage. This can be clearly seen in the next set of test data:

PWM parameter setting			Measured ripple voltage
t	T_0	t / T_0	
100	200	0.5	1.27V
50	100		668mV
25	50		350mV
10	20		154mV
5	10		82mV

The behavior of the Sink switched circuit detailed above will be similar to that of the Source switched circuit detailed earlier.

10.14 Communication Format

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

10.14.1 Specification of the communication parameters:

Items such as baud rates, stop bits and parities must be identically set between the two communicating devices. The communication parameters are selected by a bit pattern which is stored in data register D8120.

D8120				
	Description		Bit (bn)status	
			0 (OFF)	1 (ON)
b0	Data length		7 bits	8 bits
b1 b2	Parity (b2, b1)		(00) : No parity (01) : Odd parity (11) : Even parity	
b3	Stop bits		1 bit	2bits
b4 b5 b6 b7	Baud rate - bps		(b7, b6, b5, b4) (0011): 300 bps (0100): 600 bps (0101): 1200 bps (0110): 2400 bps	(b7, b6, b5, b4) (0111): 4800 bps (1000): 9600 bps (1001): 19200 bps
b8	Header character		None	D8124, Default : STX (02H)
b9	Terminator character		None	D8125, Default : ETX (03H)
b10 b11 b12	Communication Control (see timing diagrams page 10-20 onwards)		No Protocol (b12, b11, b10) (0, 0, 0) : RS Instruction is not being used (RS232C interface) (0, 0, 1) : Terminal mode -RS232C interface (0, 1, 0) : Interlink mode - RS232C interface (FX2N V2.00 or above) (0, 1, 1) : Normal mode 1- RS232C, RS485(422) interfaces (RS485 FX2N(C) only) (1, 0, 1) : Normal Mode 2 - RS232C interface (FX only)	
			Computer Link (b12, b11, b10) (0, 0, 0) : RS485(422) interface (0, 1, 0) : RS232C interface	
b13	FX-485 Network	Sum Check	No Check	Added automatically
b14		Protocol	No protocol	Dedicated Protocol
b15		Protocol	Format 1	Format 4

General note regarding the use of Data register D8120:

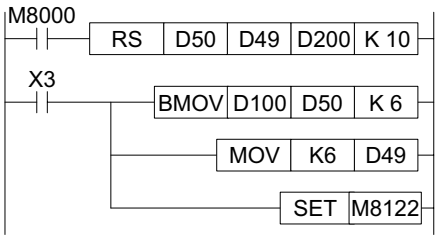
This data register is a general set-up register for all ADP type communications. Bits 13 to 15 in the 232ADP units should not be used. When using the FX-485 network with 485ADP units bits 13 to 15 should be used instead of bits 8 to 12.

10.14.2 Header and Terminator Characters

The header and terminator characters can be changed by the user to suit their requirements. The default setting for the header stored in D8124 is 'STX' (or 02H)and the terminator default setting stored in D8125 is 'ETX' (or 03H).
The header and terminator characters are automatically added to the 'send' message at the time of transmission. During a receive cycle, data will be ignored until the header is received. Data will be continually read until either the termination character is received or the receive buffer is filled. If the buffer is filled before the termination character is received then the message is considered incomplete.
If no termination character is used, then reading will continue until the receive data buffer is full. Only at this point will a message have been accepted and complete. There is no further buffering of any communications, hence if more data is sent than the available destination buffer size then the excess will be lost once the buffer is full.
It is therefore very important to specify the receive buffer length the same size as the longest message to be received.

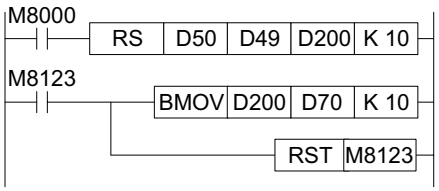
Events to complete a transmission:

The RS instruction should be set up and active.
The data to be transmitted should be moved into the transmission data buffer. If a variable is being used to identify the message length in the RS instruction this should be set to the new message length. The send flag M8122 should then be SET ON. This will automatically reset once the message has been sent. Please see the example program right.



Events encountered when receiving a message:

The RS instruction should be set up and active.
Once data is being received and an attempt is made to send out data, the special M flag M8121 is set ON to indicate the transmission will be delayed. Once the 'incoming' message is completely received the message received flag M8123 is set ON. At the same time if M8121 was ON it is automatically reset allowing further messages (delayed or otherwise) to be transmitted.



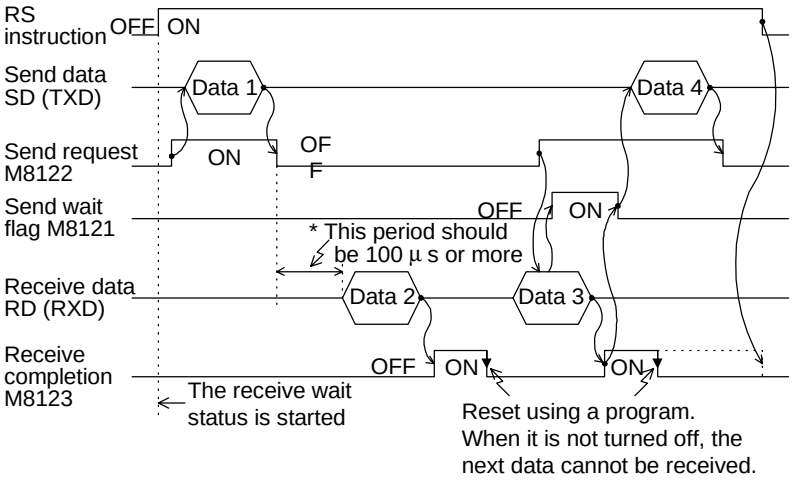
It is advisable to move the received data out of the received data buffer as soon as possible. Once this is complete M8123 should be reset by the user. This is then ready to send a message or to await receipt of a new message.

10.14.3 Timing diagrams for communications:

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)★

1) No Handshaking D8120 (b12, b11, b10) = (0, 0, 0)

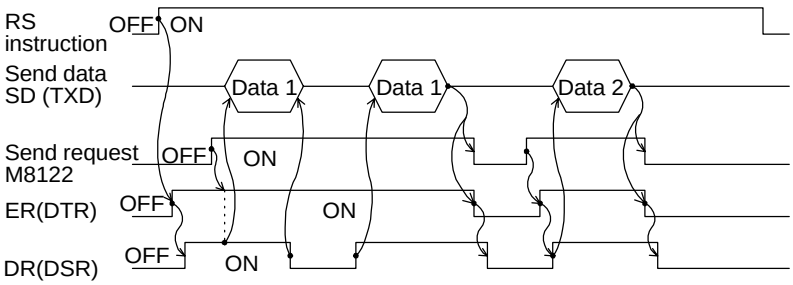
☆FX2N below version 2.0



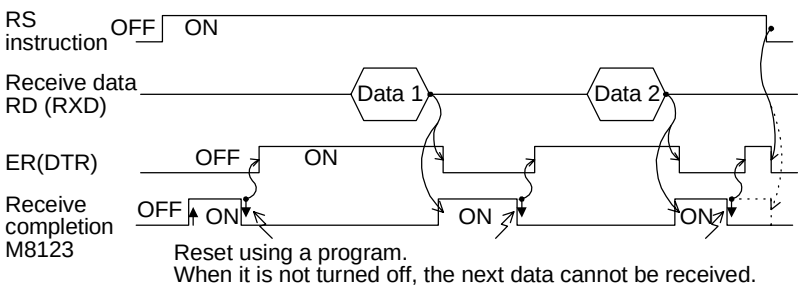
*When using FX0N, FX, FX2C this should be 2 x scan time or more

2) Terminal mode D8120 (b12, b11, b10) = (0, 0, 1)

a) Send Only



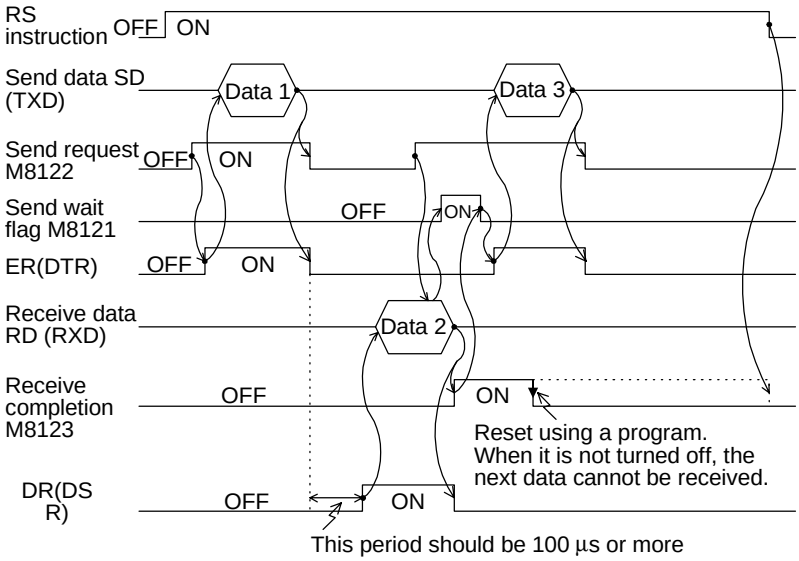
b) receive only



3) Normal Mode 1 D8120 (b12, b11, b10) = (0, 1, 1)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

☆FX2N below V2.00.

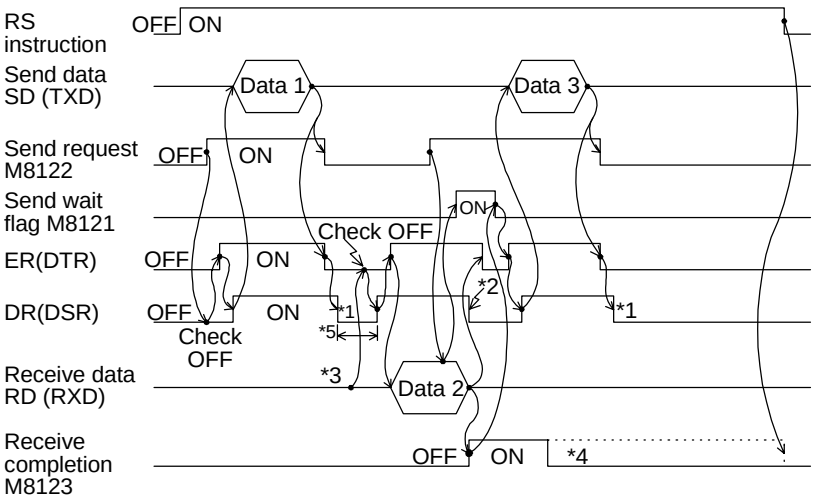


* When using FX0N, FX, FX2C this period should be 2x scan time or more.

4) Normal Mode 2 D8120 (b12, b11, b10) = (1, 0, 1)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

☆FX2N below V2.00

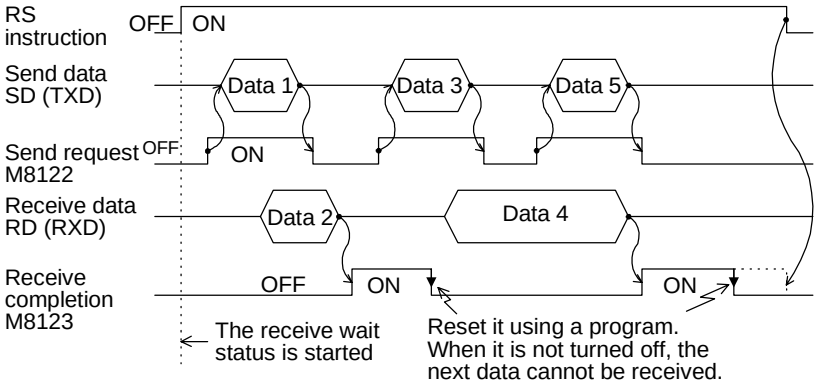


FX2N (V2.00 or above) Communications

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

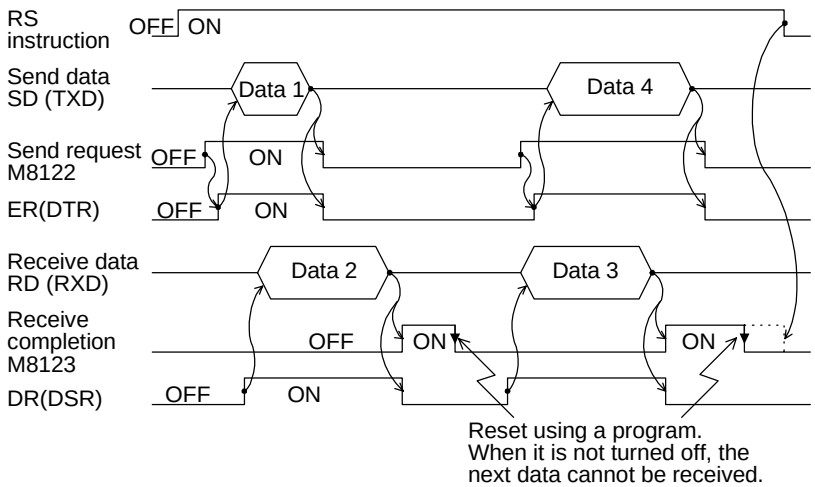
In the FX2N V2.00 or above and FX2NC, full duplex communication is performed.

1) No Hardware Handshaking D8120 (B12, b11, b10) = (0,0,0)



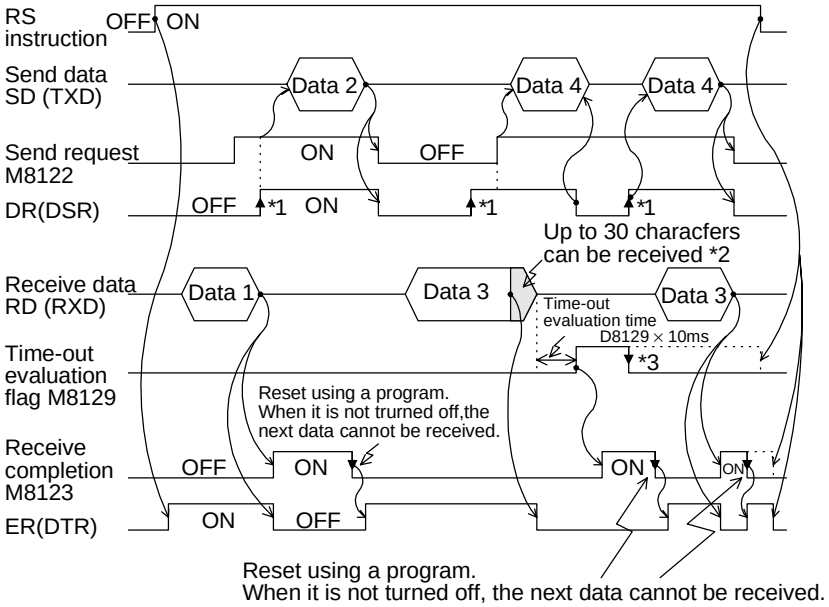
2) Terminal Mode
   The control line and transmission sequence are identical to those in the FX, on page

3) Normal Mode 1 D8120 (b12, b11, b10) = (0, 1, 1)



4) Interlink Mode D8120 (b12, b11, b10) = (0, 1, 0)

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)



10.14.4 8 bit or 16 bit communications.

This is toggled using the Auxiliary relay M8161. When this relay is OFF 16 bit communications takes place. This actually means that both bytes of a 16 bit data device are used in both the transmission and the receipt of messages. If the M8161 device is activated then 8 bit mode is selected. In this mode only the lower 8 bits (or byte) is used to perform the transmission-receiving actions. The toggling of the M8161 device should only occur when the RS instruction is not active, i.e. it is OFF.
When a buffer area is specified in the RS instruction it is important to check whether 8 or 16bit mode has been selected, i.e. a buffer area specified as D50 K3 would produce the following results.....

16 bit mode - M8161 = OFF		
Data register	High byte	Low byte
D50	X	F
D51		0

8 bit mode - M8161 = ON		
Data register	High byte	Low byte
D50		F
D51		X
D52		0

General note regarding hardware:

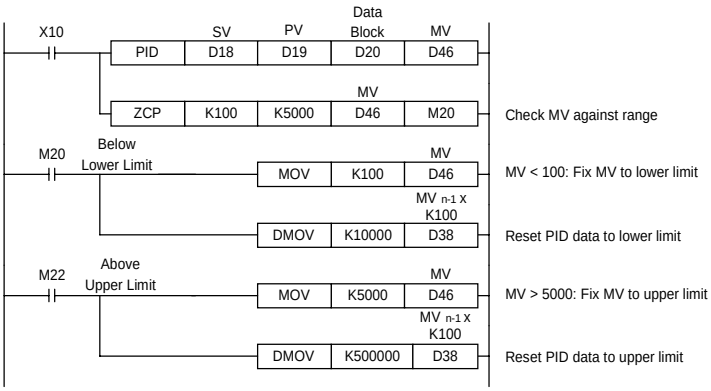
Information regarding pin outs of the respective ADP special function blocks can be found along with wiring details in the appropriate hardware manuals.

10.15 PID Programming Techniques

FX0(S) | FX0N | FX | FX(2C) | FX2N(C)

10.15.1 Keeping MV within a set range

In the reserved registers of the PID data block S_3+18 and S_3+19 form a double word device that contains the previous MV x K100. The following program uses this to keep MV under control when it exceeds the operating limits.
Example Program to keep MV in the range K100 to K5000



If data registers are used to hold the limit values, it is possible to use a MUL instruction instead of the DMOV. E.g. When D50 is upper limit use: MUL D50 K100 D38 because the result of MUL is already a double word DMUL is not needed.
Resetting (S_3+19, S_3+18) in this way prevents runaway, which occurs if only MV is changed.

10.15.2 Manual/Automatic change over

In order to switch from automatic (PID) control to manual control and back to automatic it is necessary for the PID process to perform 'Manual Tracking'. Although the FX PID instruction does not have a manual tracking feature there are two methods that can be used to make the switch from manual back to automatic as trouble free as possible.
To understand the reason for the two methods the following should be noted. The PID instruction sets its initial output value based on the initial value of the output register.
When the PID instruction is switched on it can only do P as it has only 1 data reading. On the first reading the current value of the output register is used as ΔMV. Thereafter the previous output value is used (stored in S_3+18, S_3+19).

After the next reading PI can be calculated and from the third reading full PID is performed.

Please see section 5.98, PID (FNC 88), for the complete equations.

Method 1

It is recommended that if manual to auto switching is desired that the PID instruction is switched off during manual operation and the operator controls the value of the MV register (the Output Value). When returning to auto mode, the PID instruction is switched on again and uses the last MV input by the operator during the first PID calculation. After 3 readings full PID will be operating and the process should be under control quickly. (Assuming that manual control did not cause a move too far from the Set Point.)

Method 2

During manual operations the PID instruction is kept running but the calculated MV is ignored; instead the operator controls MV. In order to prevent the PID instruction from running out of control the MV value set by the operator should be fed in to the MV_{n-1} registers of the PID data block in the same way as for MV range control earlier (i.e. Set S_3+18, S_3+19 to MV x 100). When switching back to PID control the internal values of the PID instruction are already set and full PID operation starts immediately.

10.15.3 Using the PID alarm signals

Included as part of the data block there are four alarm values. These set the maximum positive and negative change that should occur to MV and PV. The PID alarm signals are used to warn of the system going out of control.
When the system is starting from cold it is usually not good to include the Derivative numbers of the in the calculation; the changes to PV are large and the Derivative introduces too much correction. Also, if the system starts to move rapidly away from the SV then sometimes the use of D can over correct and cause chasing.
By having an 'alarm' flag for the change in PV and MV it is possible to monitor the state of the system and adjust the PID parameters to appropriate settings.
When the system is close to the SP the changes in PV (and MV) should be minimal.
In this situation using full PID is very useful in keeping the system close to the SP. (Full PID is appropriate).
However, if the conditions change (e.g. opening a refrigerator door, adding ingredients to a mixture, cold start, etc.) the system reacts. In some cases (especially cold start) the reaction is too much for the D to be useful (PI or sometimes just P only is better). In these cases the alarm flags can be used to change to PI control until the system returns to a more stable condition, when full PID can then be used.
Basically, rather than use actual values of the PV to determine the change over point from PI to PID (or PID to PI), use the size of the change in PV (or MV). This means changes to the Set Point do not require different ranges for the PI - PID change over point (at least, in theory).

10.15.4 Other tips for PID programming

- It is recommended that an input value for PV is read before the PID is activated. Otherwise, the PID will see a big change from 0 to the first value and calculate as if a big error is occurring.
- The PID instruction is not interrupt processed. It is scan dependent and as such the sampling can not occur faster the FX scan time. It is recommended that T_S is set to a multiple of the program scan time.
- To keep timing errors to a minimum it is recommended that constant scan is used.
- To improve sampling rates it is possible to put the PID instruction inside a timer interrupt routine.
- It is better to have the PID only perform P until the input value (PV) reaches the working range.
- When setting up it is a good idea to monitor the input and output of the PID instruction and check that they are about the expected values.
- If the PID system is not operating properly check the error flags for PID errors (D8067).

10.16 Additional PID functions

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

The following parameter table gives the additional parameters available with FX2N(C) MPUs. These are:

- S3+1 bit 4: Pre-tuning operation flag.
- S3+1 bit 5: Output Value range limit flag.
- S3+22: Output Value upper limit.
- S3+23: Output Value lower limit.

Parameter *S3* + P	Parameter name/function	Description		Setting range
S3+1	Action-reaction direction and alarm control	b0	Forward operation(0), Reverse operation (1)	Not applicable
		b1	Process Value (*S2*) change alarm OFF(0)/ON(1)	
		b2	Output Value (*MV*) change alarm OFF(0)/ON(1)	
		b3	Reserved	
		b4	Activate pre-tuning (auto resets on completion)	
		b5	Output Value *(MV)* range limit OFF(0)/ON(1)	
		b6-15	Reserved	
S3+22	Output Value, maximum positive change alarm	Active when *S3*+1, b2 is set ON.	This is an alarm for the quantity of positive change which can occur in one PID scan. If the Output Value *(MV)* exceeds this value, bit *S3*+24, b2 is set	0 to 32767
	Output Value, Upper limit restriction	Active when *S3*+1, b5 is set ON.	This is an upper limit for the Output Value *(MV)*. During operation the PID instruction restricts the output so that it does not exceed this limit.	-32768 to 32767
S3+23	Output Value, maximum negative change alarm	Active when *S3*+1, b2 is set ON.	This is an alarm for the quantity of negative change which can occur in one PID scan. If the Output Value *(MV)* falls below this value, bit *S3*+24, b3 is set.	0 to 32767
	Output Value, Lower limit restriction	Active when *S3*+1, b5 is set ON.	This is a lower limit for the Output Value *(MV)*. During operation, the PID instruction restricts the output so that it does not fall below this limit.	-32768 to 32767

For the full list of other parameters refer to page 5-102.

Note: S3+1 b2 and b5 should not be active at the same time. Only one value each is entered into the data registers S3+22 and S3+23.

10.16.1 Output Value range control (S_3+1 b5)

Bit 5 of parameter S3+1, when ON, activates S3+22 and S3+23 to be upper and lower limits for the output value (MV).
This feature restricts the output value to the specified limits; in effect, this automatically performs the same operation as that described in section 10.15.1.

10.17 Pre-tuning operation

FX0(S)	FX0N	FX	FX(2C)	FX2N(C)

10.17.1 Variable Constants

The Pre-tuning operation can be used to automatically set values for the following variables:

- The direction of the process; Forward or Reverse (S_3+1, bit 0)
- The proportional gain constant; K_P (S_3+3)
- The integral time constant; T_I (S_3+4)
- The derivative time constant; T_D (S_3+6)

Setting bit 4 of S_3+1 starts the pre-tuning process. Before starting, set all values that are not set by the pre-tuning operation: the sample time, Ts (S_3+0); the input filter α (S_3+2); the Derivative gain, K_D (S_3+5); the Set Point, SV (S_1); and any alarm or limit values, (S_3+20-23).

The Pre-tuning operation measures how fast the system will correct itself when in error. Because the P, I, and D equations all react with differing speed, the initial error must be large so that effective calculations can be made for each type of equation. The difference in values between SP and PV_{nf} must be a minimum of 150 for the Pre-tuning to operate effectively. If this is not the case, then please change SV to a suitable value for the purpose of pre-tuning.

The system keeps the output value (MV) at the initial value, monitoring the process value until it reaches one third of the way to the Set Point. At this point the pre-tuning flag (bit 4) is reset and normal PID operation resumes. SV can be returned to the normal setting without turning the PID command Off.

During the course of normal operation, the Pre-tuning will NOT automatically set new values if the SV is changed. The PID command must be turned Off, and the Pre-Tuning function restarted if it is necessary to use the Pre-tune function to calculate new values.

- Caution: The Pre-tuning can be used as many times as necessary. Because the flag resets, the set bit can be turned On again and new values will be calculated. If the system is running an oven heater and the SV is reduced from 250 to 200 C, the temperature must drop below 200 or the “Forward/Reverse” flag will be set in the wrong direction. In addition, the system error value must be large for the pre-tune variable calculations to work correctly.

- Note: Set the sampling time to greater than 1 second (1000 ms) during the pre-tuning operation. It is recommended that the sampling time is generally set to a value much greater than the program scan time.

- Note: The system should be in a stable condition before starting the pre-tuning operation. An unstable system can cause the Pre-tuning operation to produce invalid results. (e.g. opening a refrigerator door, adding ingredients to a mixture, cold start, etc.)
- Note: Even though Pre-tuning can set the above mentioned variables, additional logic may be needed in the program to "scale" all operating values to those capable of being processed by the special function devices being used.

10.18 Example Autotuning Program

The following programming code is an example of how to set up the Pre-Tuning function.

•

D500: SV = 500

D502: MV = 1800, initial value

D510: T_S, S_3+0 = 3000

D511: S_3+1, Bits 0-3 and 5-15 Off, Bits 4 and 5 On. Bit 4 = Pre-Tune Function
Bit 5 = MV Range Limit

D512: Input Filter, S_3+2 = 70%

D515: K_D , S_3+5 = 1800, initial value

D532: MV Max, S_3+22 = 2000

D533: MV Min, S_3+23 = 0

Pulse M1 to turn On PID command

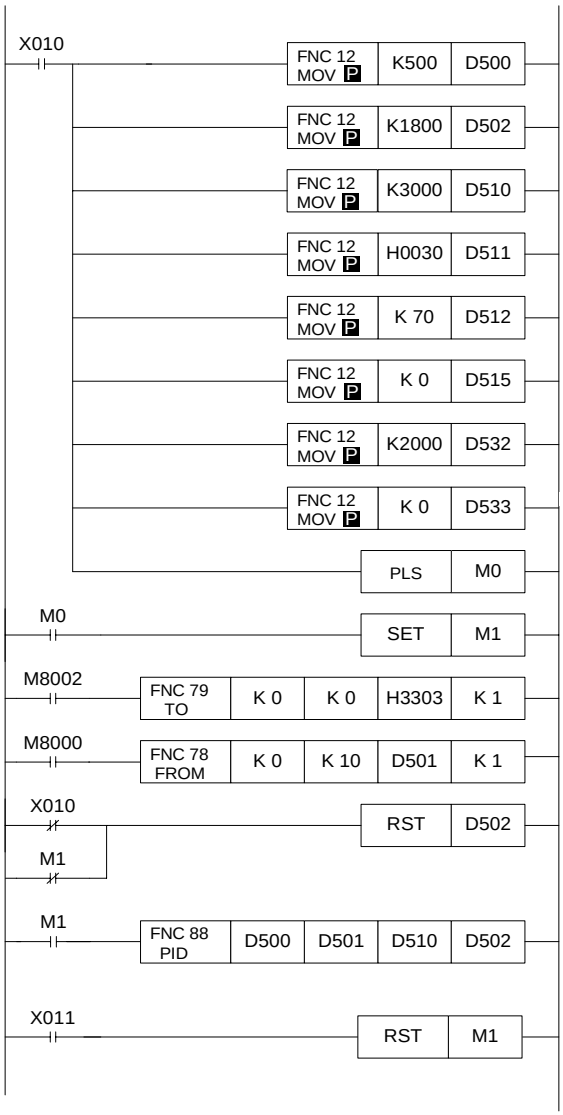
Send setting to Special Function Block

Read data from Special Function Block

Reset Output data when PID command is Off

PID Instruction Command Line

Turn Off PID Instruction



Chapter contents

11. Index

11.1 Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

11.2 ASCII Character Codes

Table 11.1:

ASCII code table (HEX)		Higher bit						
		1	**2**	**3**	**4**	**5**	**6**	**7**
Lower bit	**0**	Not accessible	(SP)	0	@	P	@	p
	1		!	1	A	Q	a	q
	2		“	2	B	R	b	r
	3		#	3	C	S	c	s
	4		$	4	D	T	d	t
	5		%	5	E	U	e	u
	6		&	6	F	V	f	v
	7		‘	7	G	W	g	w
	8		(	8	H	X	h	x
	9		)	9	I	Y	i	y
	A		*	:	J	z	j	z
	B		+	;	K	[	k	{
	C		,	<	L		l	|
	D		-	=	M	]	m	}
	E		.	>	N	(SP)	n	~
	F		/	?	O	_	o	C_R

Note:
(SP) = Space,
C_R = Carriage Return

11.3 Applied Instruction List

	Memonic	Fnc	Page	FX2N(C)	FX (CPU ver 3.07 or greater) and FX2C	FX (CPU ver 2.0 to 3.06)	FX0N	FX0,FX0S
A	ABSD	62	5-70					
	ADD	20	5-25					
	ALT	66	5-73					
	ANDq	232-238	5-148					
	ANR	47	5-47					
	ANRD	91	5-111					
	ANS	46	5-47					
	ANWR	92	5-112					
	ARWS	75	5-87					
	ASC	76	5-88					
	ASCI	82	5-98					
B	BCD	18	5-22					
	BIN	19	5-22					
	BLK	97	5-115					
	BMOV	15	5-20					
	BON	44	5-45					
C	CALL	01	5-7					
	CCD	84	5-100					
	CJ	00	5-5					
	CML	14	5-19					
	CMP	10	5-17					
	COS	131	5-128					
D	DEC	25	5-29					
	DECO	41	5-43					
	DI	05	5-9					
	DIV	23	5-28					
	DSW	72	5-83					
E	EADD	120	5-121					
	EBCD	118	5-120					
	EBIN	119	5-120					
	ECMP	110	5-119					
	EDIV	123	5-123					
	EI	04	5-9					
	EMUL	122	5-122					
	ENCO	42	5-44					
	ESQR	127	5-123					
	ESUB	121	5-122					
	EZCP	111	5-119					
F	FEND	06	5-11					
	FLT	49	5-49					
	FMOV	16	5-21					
	FOR	08	5-13					
	FROM	78	5-90					

	Memonic	Fnc	Page	FX2N(C)	FX (CPU ver 3.07 or greater) and FX2C	FX (CPU ver 2.0 to 3.06)	FX0N	FX0,FX0S
G	GBIN	171	5-143					
	GRY	170	5-143					
H	HEX	83	5-99					
	HKY	71	5-82					
	HSCR	54	5-56					
	HSCS	53	5-55					
	HSZ	55	5-57					
I	INC	24	5-29					
	INCD	63	5-71					
	INT	129	5-124					
	IRET	03	5-9					
	IST	60	5-67					
L	LDq	224-230	5-147					
M	MCDE	98	5-116					
	MEAN	45	5-46					
	MNET	90	5-111					
	MOV	12	5-18					
	MTR	52	5-54					
	MUL	22	5-27					
N	NEG	29	5-31					
	NEXT	09	5-13					
O	ORq	240-246	5-149					
P	PID	88	5-102					
	PLSR	59	5-63					
	PLSY	57	5-61					
	PR	77	5-89					
	PRUN	81	5-96					
	PWM	58	5-62					
R	RAMP	67	5-73					
	RCL	33	5-36					
	RCR	32	5-36					
	REF	50	5-53					
	REFF	51	5-53					
	RMMN	96	5-114					
	RMRD	95	5-114					
	RMST	93	5-112					
	RMWR	94	5-113					
	ROL	31	5-35					
	ROR	30	5-35					
	ROTC	68	5-75					
	RS	80	5-95					

	Memonic	Fnc	Page	FX2N(C)	FX (CPU ver 3.07 or greater) and FX2C	FX (CPU ver 2.0 to 3.06)	FX0N	FX0,FX0S
S	SEGD	73	5-84					
	SEGL	74	5-85					
	SER	61	5-69					
	SFRD	39	5-40					
	SFTL	35	5-37					
	SFTR	34	5-37					
	SFWR	38	5-39					
	SIN	130	5-127					
	SMOV	13	5-18					
	SORT	69	5-77					
	SPD	56	5-60					
	SQR	48	5-48					
	SRET	02	5-8					
	STMR	65	5-72					
	SUB	21	5-26					
	SUM	43	5-45					
	SWAP	147	5-131					
T	TADD	162	5-137					
	TAN	132	5-128					
	TCMP	160	5-135					
	TKY	70	5-81					
	TO	79	5-91					
	TRD	166	5-139					
	TSUB	163	5-138					
	TTMR	64	5-72					
	TWR	167	5-140					
	TZCP	161	5-136					
V	VRRD	85	5-101					
	VRSC	86	5-101					
W	WAND	26	5-30					
	WDT	07	5-12					
	WOR	27	5-30					
	WSFL	37	5-38					
	WSFR	36	5-38					
	WXOR	28	5-31					
X	XCH	17	5-21					
Z	ZCP	11	5-17					
	ZRST	40	5-43					

PROGRAMMING MANUAL

THE FX SERIES OF PROGRAMMABLE CONTROLLER
(FX_0, FX_{0S}, FX_{0N}, FX, FX_{2C}, FX_{2N}, FX_{2NC})

MITSUBISHI ELECTRIC CORPORATION
HEAD OFFICE: MITSUBISHI DENKI BLDG MARUNOUCHI TOKYO 100-8310 TELEX: J24532 CABLE MELCO TOKYO
HIMEJI WORKS: 840, CHIYODA CHO, HIMEJI, JAPAN

JY992D48301J
(MEE 9911)

Effective Nov. 1999
Specification are subject
to change without notice.