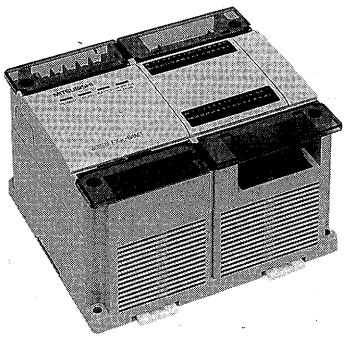
MITSUBISHI

PROGRAMMABLE CONTROLLERS

MELSEC-F

FX_{2C} SUPPLEMENTARY PROGRAMMING MANUAL

THE FX_{2C} PROGRAMMABLE CONTROLLER



FX2C Supplementary Programming Manual

Manual number: JY992D50201
Print reference: HI-HB-151-A9501 (SEN) (A1)
Manual revision: A
Date: January 1995

Foreword

- This manual contains text, diagrams and explanations which will guide the reader in the correct programming and operation of the PC.
- Before attempting to install or use the PC this manual should be read and understood.
- If in doubt at any stage of the installation of the PC 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 PC please consult the nearest Mitsubishi Electric distributor.
- This manual is subject to change without notice.

FAX BACK

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.

Fax numbers:
Mitsubishi Electric.....
America (708) 298-1834
Australia (02) 638 7072
Germany (0 21 02) 4 86-1 12
South Africa (0111) 444-8304
United Kingdom (01707) 278695

Your name: . . . . . .
. . . . . .
Your company: . . . . . .
. . . . . .
Your location: . . . . . .
. . . . . .

Please tick the box of your choice;

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 (PC)

This manual provides information for the use of the FX family of PCs. 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 fulfil 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 fulfil 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 co-ordinated manner in compliance to established safety practices. The operators should also be familiar with all 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.

Further References:

For further information regarding the installation and operation of the FX2C please reference the following manuals:

Manual:

FX Series Programmable Controllers Hardware Manual–

This manual describes the range of available units, their basic features and installation details.

Manual:

FX Series Programming Manual–

This manual describes the programming features of FX0, FX0N and FX series programmable controllers. All of the FX instructions detailed within this manual are applicable to the FX2C (with the exception of those changes detailed within this supplementary manual).

Manual:

FX Series Programmable Controller Handy Manual–

This manual describes the programming features of the FX series programmable controllers. All of the FX instructions detailed within this manual are applicable to the FX2C (with the exception of those changes detailed within this supplementary manual).

1. Introduction

The following information has been collated in to a small reference guide to the new software features of the FX2C. This introduction brings together all of the attached sheets and catalogs and references them. This should aid the reader in being able to identify where the information is that they require.

These new software features are in addition to a basic program set which was taken from the standard FX PC. If information is required on an existing feature, i.e. one which appears on the FX PC please look for further guidance in either the FX Handy Manual or the new combined FX, FX0N, FX0 Programming Manual.

1.1 New FX2C Instructions

Word Description	Mnemonic	Function Number	Sheet Reference Number	☆ Number of pages
ASCII to Hexadecimal Conversion	HEX	83	SUP 14-☆	2
Check Code Instruction	CCD	84	SUP 15-☆	2
Float Instruction	FLT	49	SUP 3-☆	2
Hexadecimal to ASCII Conversion	ASCI	82	SUP 13-☆	2
PID Control Loop	PID	88	SUP 16-☆	5
Search Instruction	SER	61	SUP 21-☆	2
Serial Communications	RS	80	SUP 12-☆	4
Sort Instruction	SORT	69	SUP 22-☆	3
Square Root Instruction	SQR	48	SUP 1-☆	2

1.2 Additional/Upgraded FX2C Instructions

Existing Mnemonic	Word Description	Sheet Reference Number	☆ Number of pages
ADD (Fnc 20)	The Add Instruction Used With The M8023 Float Operation Flag	SUP 8-☆	3
BCD (Fnc 18)	The Decimal To BCD Instruction Used With M8023 Float Operation Flag	SUP 19-☆	2
BIN (Fnc 19)	The BCD To Decimal Instruction Used With M8023 Float Operation Flag	SUP 20-☆	2
DIV (Fnc 23)	The Divide Instruction Used With The M8023 Float Operation Flag	SUP 11-☆	3
FLT (Fnc 49)	The Float Instruction Used With M8023, The Float Operation Flag	SUP 4-☆	3
HKY (Fnc 71)	The Hexadecimal Key Input Instruction Used For Hexadecimal Input	SUP 7-☆	3
HSZ (Fnc 55)	Zone Compare Instruction With Table Comparison Operation Flag M8130	SUP 5-☆	3
HSZ (Fnc 55)	Zone Compare Instruction Used For Pulse Control With Flag M8132	SUP 6-☆	3
MUL (Fnc 22)	The Multiply Instruction Used With The M8023 Float Operation Flag	SUP 10-☆	3
SMOV (Fnc 13)	The Shift Move Instruction Used With M8168	SUP 17-☆	2
SQR (Fnc 48)	The Square Root Instruction Used With M8023 Float Operation Flag	SUP 2-☆	2
SUB (Fnc 21)	The Subtract Instruction Used With The M8023 Float Operation Flag	SUP 9-☆	3
XCH (Fnc 17)	The Exchange Instruction Used As A Byte Swap Instruction	SUP 18-☆	2

1.3 Additional Explanation Sheets

Word Description	Sheet Reference Number	☆ Number of pages
An Introduction To Number Formats For The FX	SUP 23-☆	2
Detailed Explanation Of The Structure Of Floating Point	SUP 24-☆	2

1.4 Use of Old Programming Equipment with FX2C PCs

The new instructions in the FX2C cannot be directly accessed using old programming tools. However, certain 'standard' applied instructions can be programmed in conjunction with special auxiliary coils (M coils) to achieve the same 'effective instruction' as the new instructions. The following tables identify which version of peripherals will work directly with all features of the FX2C and which peripherals will need to use the special M coils.

PERIPHERALS TABLE			
Description	**Model No.**	**Old Version requiring use of auxiliary M coils**	**New version providing complete compatibility**
Hand Held Programmer	FX-10P-E	V 1.10	V 2.00 ⇨
Hand Held Programmer Cassette	FX-20P-MFXA-E	V 1.20	V 2.00 ⇨
Programming Software	FX-PCS/AT-EE	V 1.01	V 2.00 ⇨
	FX-A6GPP-E-KIT	V 1.00	V 2.00 ⇨
Data Access Unit	FX-10DU-E	V 1.10	V 2.00 ⇨
	FX-20DU-E	V 1.10	V 2.00 ⇨
	FX-30DU-E		V 1.00 ⇨
	FX-40DU-E(S)		V 1.00 ⇨
	FX-40DU-TK-ES		V 1.00 ⇨

EXISTING APPLIED INSTRUCTION AND SPECIAL M COIL COMBINATION TO ACHIEVE THE IDENTIFIED NEW FX2C INSTRUCTION					
Existing FX instruction			**Mimicked FX2C Instruction**		
Mnemonic	FNC Number	Additional M coils	Description	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	M8196	ACII to HEX conversion	HEX	83
FMOV	16	M8195	Sum check	CCD	84

1.5 High Speed Instructions

One of the major new features of the FX2C is the enhanced processing of both basic and applied instructions. A full listing of available instructions for each of the programmable controllers in the FX series and their associated execution times can be found at the back of this supplementary manual (pages ET–1 to 10).

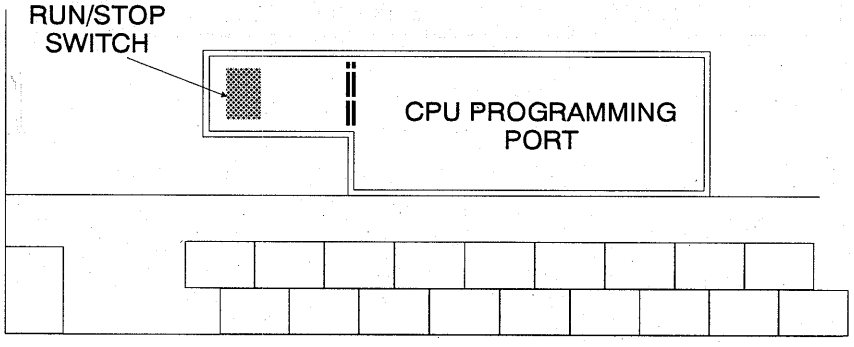
Under the FX2C listings, where applicable, the execution times for the second (2nd FNC) functions, i.e those 'standard applied instructions' which are modified by an additional special Mcoil to perform one of the new FX2C functions, have also been listed.

1.6 Functions with Minor Upgrades

Functions	Description of Change	Remark
EI	Now activates a pulse catch on the high speed inputs when not used for another high speed purpose.	Catch signaled with M8170 to M8175.
BMOV	Can now read and WRITE to file registers.	RAM file registers also possible.
HSCS	Now possible to activate an interrupt.	Uses I010 to I060 for output.
PLSY	Output pulses counted in a double word device.	Uses D8137,D8136
(D)FMOV, (D)MEAN, (D)ABSD	Double Word instructions are now possible.	–
DSW, SEGL, PR	Up to 2 instructions can be programmed in one program at the same time.	–

1.7 New RUN/STOP Switch

The FX2C has a built in RUN/STOP switch (for further information see the appropriate hardware manual). Certain models of the Japanese specification FX2 series units also have the same programming specification and the RUN/STOP switch as the FX2C. The location of the RUN/STOP switch for the Japanese specification units is as shown in the diagram below. The switch operates in parallel with the RUN input terminal.



MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE SQUARE ROOT INSTRUCTION (SQR, FNC 48)

Introduction

The FX PC family has a powerful set of applied instructions as well as the commonly used basic instructions. The applied instructions are versatile additions to the basic set and provide control over internal PC operations and external I/O. There are 10 groups of applied instructions, each with a different area of application, eg. arithmetic and logical operations.

Applied instructions 40 to 49 are concerned with the conversion between different data formats within the PC and are grouped under the heading of Data operations.

This group includes the Square Root instruction (SQR, FNC 48). This performs the mathematical operation

$$y = \sqrt{x}$$

on the given data and returns a truncated whole number for the answer.

The Square Root Instruction

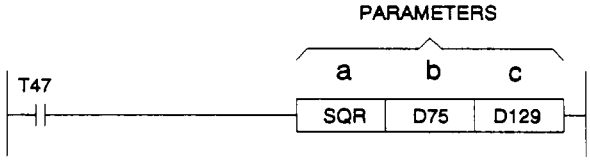
The Square Root instruction has three parameters:

a) the SQR statement itself

b) the specification of the operand i.e. the device or value on which square root is performed.

c) the specification of the result data's destination, i.e. where the result is to be put.

Shown below is the typical appearance of the SQR instruction in a ladder program, while the following table shows the same piece of ladder logic written in instruction format.



PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LD	T	47
1	SQR		48
		D	75
		D	129

It can be seen from the ladder program that when timer T47 turns ON, the square root instruction executes. It reads the contents of D75, calculates the square root and stores it in the data register D129. The following section looks at each parameter in greater detail:

Parameter a)- The SQR statement

The SQR statement can take four different forms:

I) SQR: The standard form of the instruction, taking 5 steps of program. While the SQR instruction is operating the result is continually calculated.

II) SQRP: The "P" suffix shows that SQR is being used in its PULSE format. The instruction will only calculate the result once when it is turned ON. SQRP also takes 5 program steps.

III) DSQR: This is similar to SQR, but the "D" prefix means DOUBLE word is used in the calculation for both the operand and the result, ie. 32 bits, equivalent to two consecutive data registers. DSQR takes 9 steps of program.

IV) DSQRP: This combines the DOUBLE word format with the PULSE operation to give a SQR instruction which calculates the square root of 32 bits of data when first turned ON. DSQRP takes 9 program steps.

Parameter b)- Operand

This identifies the location of the operand for the square root calculation. The data is either 16 or 32 bits (SQR, DSQR) and can be either a constant (K,H) or a data register (D). If the value is negative then the SQR instruction will generate an execution error.

Parameter c)- Destination

This is where the result of the calculation is placed. The data is always stored in a data register (D) as either 16 or 32 bits. The result is rounded down to the nearest whole number by ignoring the fractional part.

M8020 zero flag operates if the result is zero.

M8021 borrow flag operates if the result is rounded.

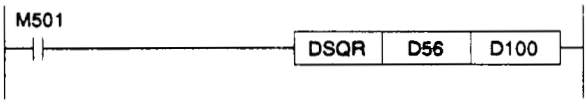
FX2

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE SQUARE ROOT INSTRUCTION (SQR, FNC 48)

Example Use of the Square Root Instruction

The diagram below shows the square root instruction being used to calculate the square root of the current contents of data register pair D57 and D56 and store the result in the data register pair D101 and D100. This occurs when the auxilliary relay M501 turns on.



The table below shows some example results of the square root instruction. Note that all values are rounded down to the nearest whole number. If the true root value is needed then it is recommended that the floating point version of the SQR instruction is used, see SUP 2.

OPERAND	ROOT	SQR RESULT
25	5.0	5
60	7.746	7
–236	15.36 i	ERROR
147	12.124	12

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE SQUARE ROOT INSTRUCTION (SQR, FNC 48) USED WITH M8023 FLOAT OPERATION FLAG

Introduction

The FX PC family has a powerful set of applied instructions as well as the commonly used basic instructions. The applied instructions are versatile additions to the basic set and provide control over internal PC operations and external I/O. There are 10 groups of applied instructions, each with a different area of application, eg. arithmetic and logical operations.

Applied instructions 40 to 49 are concerned with the conversion between different data formats within the PC and are grouped under the heading of Data operations.

This group includes the square root instruction (SQR, FNC 18). When used in conjunction with the float operation flag M8023 the SQR instruction calculate the square root of data from a floating point number or constant and gives the result as a floating point number.

The SQR Instruction with M8023

This form of the SQR instruction has three parts:

Part 1: Set M8023

a) setting the floating point operation flag.

Part 2: The SQR instruction (three parameters)

b) the SQR statement itself.

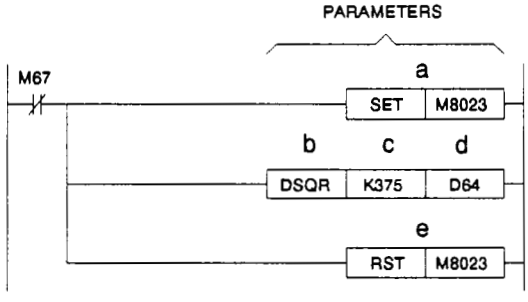
c) the specification of the operand i.e. the number which is to be square rooted.

d) the specification of the destination data i.e. to where the result of the calculation is to be written.

Part 3: Reset M8023

e) resetting the floating point operation flag.

Shown below is this format of the SQR instruction as a ladder diagram showing each of these features. The following table shows the same program written in instruction format.



PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LDI	M	67
1	SET	M	8023
3	DSQR		48
		K	375
		D	64
12	RST	M	8023

Looking at the ladder diagram we can see that when the auxiliary relay M67 is turned OFF, first the float operation flag is set to ON, then the SQR instruction executes. This results in the square root of the constant K375 being calculated as a floating point number. This value is then written to the data register pair D65 and D64. Finally the float operation flag is reset.

Parameter a)- Setting the float flag

The Float Operation flag M8023 must be SET. This flag changes the function of the SQR instruction and other math functions to operate on floating point numbers.

Parameter b)- The SQR statement

The SQR statement can only be DOUBLE word when used with the float operations flag:

I) DSQR: The basic format for this use of the SQR instruction, taking 9 steps of program memory. While the instruction is ON it continues to execute. The "D" prefix indicates DOUBLE word operation, i.e. two consecutive data registers (32 bits).

II) DSQRP: This is the PULSE format of the DSQR instruction. The instruction will execute once when it is first turned ON (i.e. rising edge execution) and operates on 32 bits of data.

Parameter c)- Source Data

This identifies the location of the value to be square rooted. The data can be either a constant (K,H) or held in data registers (D) and must be a floating point format number. Constants are converted to floating point automatically. If the number is a negative value then an execution error will occur.

FX2

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE SQUARE ROOT INSTRUCTION (SQR, FNC 48) USED WITH M8023 FLOAT OPERATION FLAG

Parameter d)- Destination Data

This identifies the location where the result of the calculation is to be written. The data will be written in floating point and the device must be a data register (D) pair.

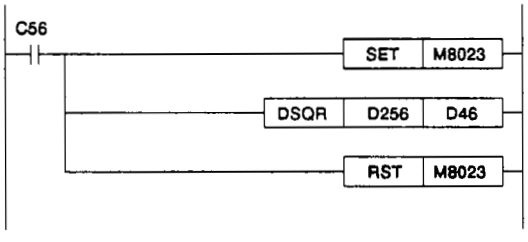
The zero flag M8020 will operate if the result is zero.

Parameter e)- Resetting the float flag

The Float Operation flag must be RESET to allow further functions to operate as normal. The flag can be left on if further operations requiring floating point calculations are needed. However, it is recommended that once all floating point functions are completed this flag should be switched off using RST.

Example Use of the SQR Instruction

The diagram below shows the SQR instruction calculating the square root of the data register pair D257 and D256. The result in floating point format is written to data register pair D47 and D46.



OPERAND	ROOT	DSQR RESULT
2.5×10^{5}	500.0	5.00×10^{2}
4.9×10^{-3}	0.07	7.00×10^{-2}
K15129	123	1.23×10^{2}
-4.096×10^{3}	64 i	ERROR

The table above shows some example results of the SQR instruction being used with the M8023 float operation flag.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE FLOAT INSTRUCTION (FLT, FNC 49)

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic set of PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. arithmetic and logical operations.

Applied instructions 40 to 49 are concerned with the conversion between different data formats within the PC and are grouped under the heading of Data operations.

This group includes the Float instruction (FLT, FNC 49). This provides the facility to convert data in the PC from integer format to data in floating point format (float).

The Float Instruction

The Float instruction has three parameters:

a) the FLT statement itself

b) the specification for the Source Data, ie. where the data in decimal format is taken from.

c) the specification for the converted data's Destination, i.e. where the converted data is to be put.

Shown below in Figure1 is the typical appearance of a Float instruction in a ladder program. While Figure 2 shows the same piece of ladder logic written in instruction format.

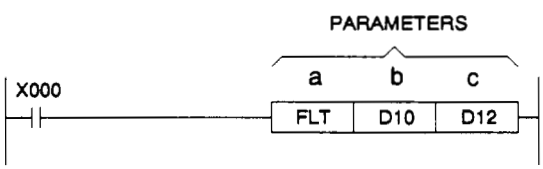


Figure 1: The Float instruction in ladder format.

PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LD	X	000
1	FLT		49
		D	10
		D	12

Figure 2: The Float instruction in instruction format.

Following Figure 1, when input X000 is turned on, the effect of the Float instruction is to read the contents of D10, convert this value in to a float value and put it into the Data Register pair D13 and D12. The following section explains each parameter in greater detail:

Parameter a)- The FLT statement

The FLT statement can take four different forms:

I) FLT: The standard form of the instruction, taking 5 steps of program. In the example shown in Figure 1, while X000 is turned on, the FLT instruction will operate.

II) FLTP: The "P" suffix shows that FLT is being used in its PULSE format. The instruction will only operate when X000 provides a rising edge signal as it turns on. FLTP also takes 5 program steps.

III) DFLT: This is similar to FLT, but the "D" prefix indicates the instruction is converting a DOUBLE word, ie. 32 bits, equivalent to two consecutive data registers. DFLT takes 9 steps of program.

IV) DFLTP: This combines the DOUBLE word format with the PULSE operation to give a FLT instruction which will convert 32 bits of data from the Source to the Destination when given a rising edge on its associated bit device, X000 in this case. DFLTP takes 9 program steps.

Parameter b)- Source Data

This identifies the location of the Source Data to be converted. The data is either 16 or 32 bits (FLT, DFLT) and must be held in a data register (D).

Parameter c)- Destination

This is where the converted Source Data is placed. The data is always stored in float format in a Data Registers (D) pair.

AN INTRODUCTION TO THE FLOAT INSTRUCTION (FLT, FNC 49)

Example Use of the Float Instruction

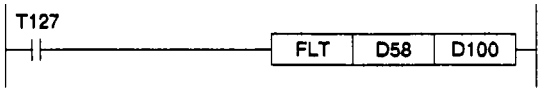


Figure 3: Example of the FLT instruction.

Figure 3 shows the Float instruction being used to convert the current contents of data register D58 and store the result in the data register pair D101 and D100 as a floating point number. This occurs when the timer contact T127 turns on.

Figure 4 illustrates the operation of this instruction giving an example of the contents of the data registers.

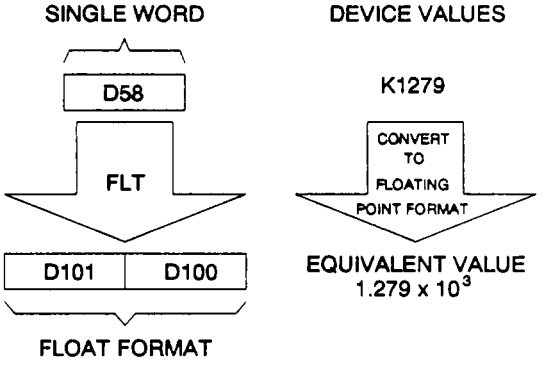


Figure 4: Operation of the FLT instruction.

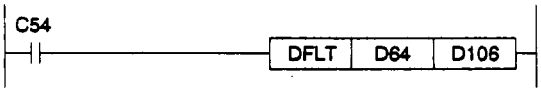


Figure 5: Example of the DFLT instruction.

Figure 5 shows the Float instruction being used to convert the current contents of data registers D64 and D65 as a double word and store the result in the data register pair D107 and D106 in float format. This occurs when the counter contact C54 turns on.

Figure 6 illustrates the operation of this instruction giving an example of the contents of the data registers.

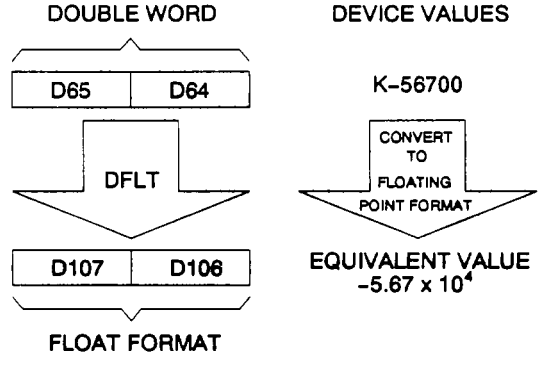


Figure 6: Operation of the DFLT instruction.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE FLOAT INSTRUCTION (FLT, FNC 49) USED WITH M8023, FLOAT OPERATION FLAG

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic set of PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. arithmetic and logical operations.

Applied instructions 40 to 49 are concerned with the conversion between different data formats within the PC and are grouped under the heading of Data operations.

This group includes the Float instruction (FLT, FNC 49). When used in conjunction with the Float Operation flag this provides the facility to convert data in the PC from floating point format (float) to data in decimal format.

The Float Instruction

The Float instruction has three parts:

Part 1: Set M8023
a) setting the floating point operation flag.

Part 2: the float instruction (three parameters)
b) the Float in statement itself

c) the specification for the decimal data location, ie. where the data in decimal format is placed.

d) the specification for the floating point data location, ie. where the floating point data to be converted is taken from.

Part 3: Reset M8023
e) resetting the floating point operation flag.

Shown below in Figure 1 is the format of a Float instruction with the float flag, in a ladder program. While Figure 2 shows the same piece of ladder logic written in instruction format.

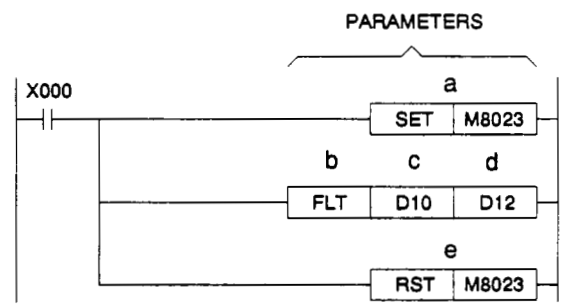


Figure 1: The Float instruction in ladder format.

PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LD	X	000
1	SET	M	8023
3	FLT		49
		D	10
		D	12
8	RST	M	8023

Figure 2: The Float instruction in instruction format.

Following Figure 1, when input X000 is turned ON, the float operation flag is set ON, the effect of the Float instruction is to read the contents of the data register pair D13 and D12, convert this float value into a decimal value and put it into data register D10. The float operation flag is then reset. The following section explains each parameter in greater detail:

Parameter a)– Setting the float flag

The Float Operation flag M8023 must be SET. This flag changes the function of the FLT instruction and math functions to operate on floating point numbers.

Parameter b)– The FLT statement

The FLT statement can take four different forms:

I) FLT: The standard form of the instruction, taking 5 steps of program. In the example shown in Figure 1, while X000 is turned on, the FLT instruction will operate converting the float value into decimal.

II) FLTP: The "P" suffix shows that FLT is being used in its PULSE format. The instruction will only operate when X000 provides a rising edge signal as it turns on. FLTP also takes 5 program steps.

III) DFLT: This is similar to FLT, but the "D" prefix indicates the instruction converts to a DOUBLE word, ie. 32 bits, equivalent to two consecutive data registers. DFLT takes 9 steps of program.

FX2

THE FLOAT INSTRUCTION (FLT, FNC 49) USED WITH M8023, FLOAT OPERATION FLAG

IV) DFLTP: This combines the DOUBLE word format with the PULSE operation to give a FLT instruction which will convert to 32 bit data when given a rising edge on its associated bit device, X000 in this case. DFLTP takes 9 program steps.

Parameter c)- Decimal Data

This identifies the location of the decimal value after the conversion. The data is either 16 or 32 bits (FLT, DFLT) and must be held in a data register (D).

Parameter d)- Float Data

This is where the floating point value is taken from before the conversion to decimal. The data is always stored in float format in a Data Registers (D) pair.

Parameter e)- Reset M8023

The Float Operation flag M8023 must be RESET to allow further functions to operate as normal. The flag can be left on if further operations requiring floating point calculations are needed. However it is recommended that once all floating point functions are completed this flag should be switched off using RST.

Example Use of the Float Instruction

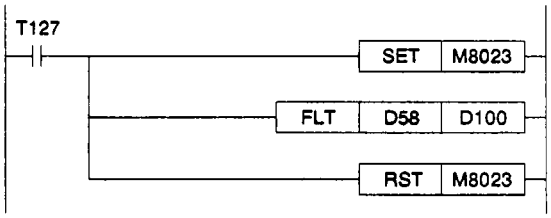


Figure 3: Example of the FLT instruction.

Figure 3 shows the Float instruction being used to convert the current contents of data register pair D101 and D100 as a float format number and store the result in data register D58 as a decimal value. This occurs when the timer contact T127 turns on.

Figure 4 illustrates the operation of this instruction giving an example of the contents of the data registers.

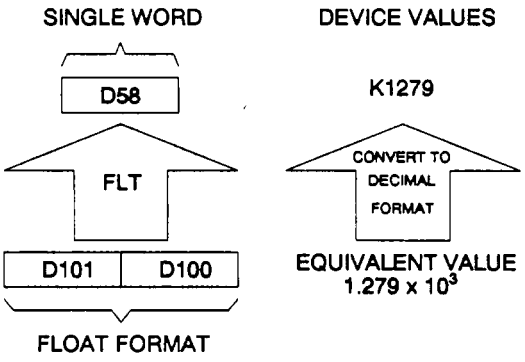


Figure 4: Operation of the FLT instruction.

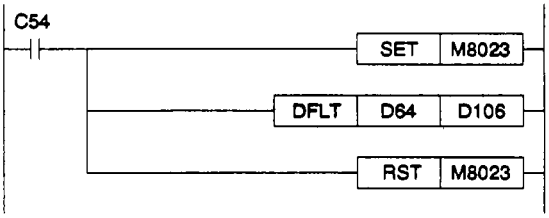


Figure 5: Example of the DFLT instruction.

Figure 5 shows the Float instruction being used to convert the current contents of the data register pair D107 and D106 as a float format number and store the result in data registers D64 and D65 as a double word in decimal. This occurs when the timer contact C54 turns on.

Figure 6 illustrates the operation of this instruction giving an example of the contents of the data registers.

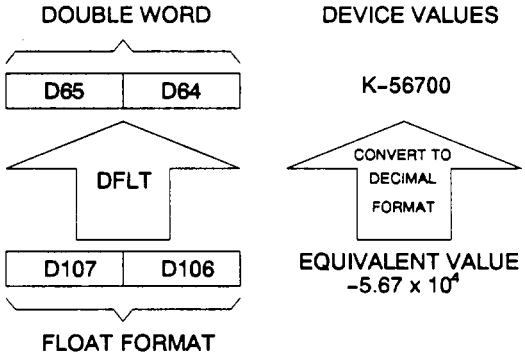


Figure 6: Operation of the DFLT instruction.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

ZONE COMPARE INSTRUCTION (HSZ, FNC 55) WITH TABLE COMPARISON OPERATION FLAG M8130

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic set of PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. arithmetic and logic.

Applied instructions 50 to 59 are all high speed instructions designed to perform independently of the program scan. This gives the group its name; High Speed instructions.

Included in the High Speed instructions is the high speed zone compare instruction (DHSZ, FNC 55). This instruction is used to monitor the value of a high speed counter (HSC). When the destination is set to special auxiliary relay M8130 the data stored in a table is compared and an output device specified in the table is changed accordingly. The table defines sequential DHSCS (FNC 53) and DHSCR (FNC 54) operations for the same counter.

The High Speed Zone Compare Instruction

The DHSZ instruction has 5 parameters:

a) the DHSZ statement itself

b) the table head address; the first data register of the comparison table.

c) the table length; the number of items in the table.

d) the test device; the high speed counter to be monitored.

e) the destination device; set to M8130 for this special function.

Shown below, in both ladder and instruction format, is the basic appearance of this form of the DHSZ instruction.

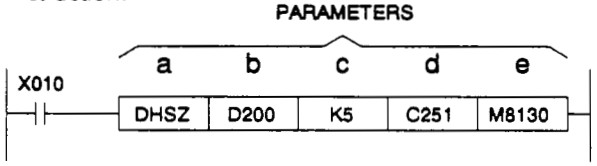


Figure 1: Example ladder for DHSZ instruction.

PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LD	X	10
1	DHSZ		55
		D	200
		K	5
		C	251
		M	8130

Figure 2: Example of DHSZ in instruction format

In the example given, each time the high speed counter C251 counts one pulse, the DHSZ instruction tests C251's current value with the value of the table entry; starting at D200 and having 5 entries. If the values match then the specified output is either set or reset and the comparison moves on to the next item in the table. When all items in the table have been matched (five in this example;K5) the table counter cycles back to the first item.

The following sections explain each parameter in more detail.

Parameter a) - The HSZ statement

This instruction has two forms but only one can be used.

I) HSZ: THIS FORMAT WILL NOT WORK.

Although it is possible to enter the instruction in this format the comparison will not give the expected results.
DO NOT use this format.

II) DHSZ: The "D" prefix indicates that this is the DOUBLE word format of the instruction. Only this format is valid because the instruction must be used with a high speed counter (all 32 bit). All word devices specified will be double word, i.e. 2 data registers or 32 bits of data will be used in the comparison. In this format the instruction takes 17 steps of program memory.

FX2

MITSUBISHI
PROGRAMMABLE CONTROLLERS

ZONE COMPARE INSTRUCTION (HSZ, FNC 55) WITH TABLE COMPARISON OPERATION FLAG M8130

Parameter b)- The table head address

This data register (D) specifies the first data register of the control table. The control table defines the values to be tested and the affect to the output when a match is made.

Parameter c)- The table length

The number of entries in the table is specified with a constant value (K,H). A maximum of 128 entries is allowed. A total of 4 data registers are used per entry. The following data must be defined for each entry in the table:

- The comparison value (Double Word).
- The output affected (Hexadecimal).
- The Set or Reset action.

ENTRY No.	COMPARISON VALUE LOWER,UPPER	OUTPUT DEVICE	SET/ RESET
0	D200,D201 K123	D202 H10 (=Y10)	D203 K1
1	D204,D205 K234	D206 H10	D207 K0
2	D208,D209 K345	D210 H23 (=Y23)	D211 K1
3	D212,D213 K456	D214 H23	D215 K0
4	D216,D217 K567	D218 H23	D219 K1

Length K5

Figure 3: Example control table; Start address D200, length K5.

The comparison value is a double word device using two consecutive data registers in the table.

Following this is the output device; specified in hexadecimal.

The last item in each entry indicates setting or resetting of the output device; K1 –set, K0 –reset.

Parameter d)- The test device

This is the high speed counter that is tested by the DHSZ instruction. Each time the counter counts, its new value is checked against the current entry in the table. When a match is made the specified output is either set or reset and the table current entry is move on to the next. Only the High Speed Counters can be used with this function; (C235 to C255).

Note: The correct HSC circuit must also be programmed.

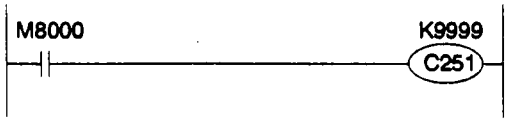


Figure 4: Minimum program required for HSC.

Parameter e)- The destination device

This parameter normally specifies the output devices to be set for the comparison. In this case the specification of special auxiliary relay M8130 indicates that a table is to be used to define the comparison values and the output devices affected.

There can only be 1 DHSZ instruction using the flag M8130 at any one time; i.e. only one switched on.

High Speed Table Comparison Operation

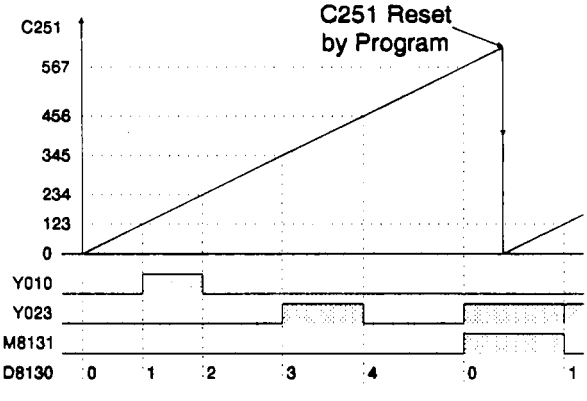


Figure 5: Operation of the outputs and the table counter D8130.

When M8130 is used with DHSZ special data register D8130 is defined as the current entry counter. As each comparison value is matched D8130 increments to the next entry number to be compared. Once all entries in the table have been matched the table counter returns to the beginning and the operation complete flag M8131 turns on.

If the counter is reset (by program or external pulse) and counts up to the first comparison value again then the operation complete flag M8131 is reset.

If the DHSZ instruction is switched off then D8130 is reset to 0 and M8131 is reset to off. All the output devices controlled by DHSZ will be left in their last state.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

ZONE COMPARE INSTRUCTION (HSZ, FNC 55) WITH TABLE COMPARISON OPERATION FLAG M8130

Important Notes

Only 6 high speed comparison functions can be active at any one time. High speed comparisons include DHSCS, DHSCR and all forms of DHSZ. Any number may be in the program but only 6 can be switched on.

When a high speed comparison is first switched on it will not be active until after the first END is executed. This is to allow the FX to initialize its internal comparison table. This will only be a problem if the high speed counter current value is close to the compare value at the time the instruction is switched on.

This instruction is interrupt processed and only operates when a signal is received at the HSC input. Therefore, changing the HSC current value or resetting the HSC from within the FX program will not be recognized until the next pulse is received at the HSC input.

The comparison values are checked in a sequential order. Take care to ensure the comparison values always increment or decrement, following the HSC operation.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

ZONE COMPARE INSTRUCTION (HSZ, FNC 55) USED FOR PULSE CONTROL WITH FLAG M8132

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic set of PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. arithmetic and logic.

Applied instructions 50 to 59 are all high speed instructions designed to perform independently of the program scan. This gives the group its name; High Speed instructions.

Included in the High Speed instructions is the high speed zone compare instruction (DHSZ, FNC 55). This instruction is used to monitor the value of a high speed counter (HSC). When the destination is set to special auxiliary relay M8132 the data stored in a table is compared and on a match the frequency defined in the table is used to control the pulse output frequency of the PLSY (FNC 57) instruction.

The High Speed Pulse Control Instruction

The DHSZ instruction in this format has 6 parameters:

a) the DHSZ statement itself

b) the table head address; the first data register of the comparison table.

c) the table length; the number of items in the table.

d) the test device; the high speed counter to be monitored.

e) the destination device; set to M8132 for this special function.

f) the DPLSY instruction; D8132 is used as the frequency control register.

Shown below, in both ladder and instruction format, is the basic appearance of this form of the DHSZ instruction.

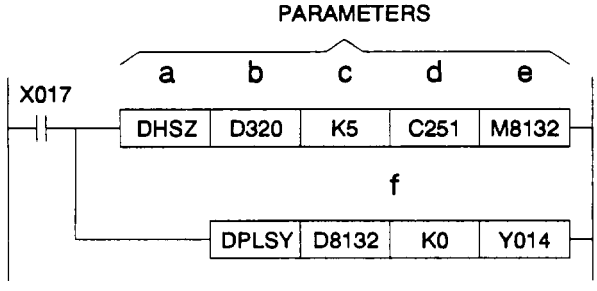


Figure 1: Example ladder for DHSZ instruction.

PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LD	X	10
1	DHSZ		55
		D	320
		K	5
		C	251
		M	8132
18	DPLSY		57
		D	8132
		K	0

Figure 2: Example of DHSZ in instruction format

In the example given, the DPLSY frequency is set to the first value in the table. Each time the high speed counter C251 counts one pulse, the DHSZ instruction tests C251's current value with the current value in the table; starting at D200 and having 5 entries.

If the values match then the comparison moves on to the next item in the table and the DPLSY frequency is set from this new entry until the new entry's comparison value is reached. When all items in the table have been matched (five in this example) the table counter cycles back to the first item.

The following sections explain each parameter in more detail.

Parameter a)- The HSZ statement

This instruction has two forms but only one can be used.

I) HSZ: THIS FORMAT WILL NOT WORK.

Although it is possible to enter the instruction in this format the comparison will not give the expected results.
DO NOT use this format.

II) DHSZ: The "D" prefix indicates that this is the DOUBLE word format of the instruction. Only this format is valid because the instruction must be used with a high speed counter (all 32 bit). All word devices specified will be double word, i.e. 2 data registers or 32 bits of data will be used in the comparison. This format of the instruction takes 17 program steps.

FX2

MITSUBISHI
PROGRAMMABLE CONTROLLERS

ZONE COMPARE INSTRUCTION (HSZ, FNC 55) USED FOR PULSE CONTROL WITH FLAG M8132

Parameter b)- The table head address

This data register (D) specifies the first data register of the control table. The control table defines the values to be tested and the affect when a match is made.

Parameter c)- The table length

The number of entries in the table is specified with a constant value (K,H); a maximum of 128 entries is allowed. A total of 4 data registers are used per entry. The following data must be defined for each entry in the table:

- The comparison value (Double Word).
- The output affected (Double Word).

ENTRY No.	COMPARISON VALUE	OUTPUT FREQUENCY (0 to 1000Hz)
	LOWER, UPPER	LOWER, UPPER
0	D320, D321 K20	D322, D323 K300
1	D324, D325 K600	D326, D327 K500
2	D328, D329 K700	D330, D331 K200
3	D332, D333 K800	D334, D335 K100

Length K5

Figure 3: Example control table;
Start address D320, length K5.

The comparison value is a double word device using two consecutive data registers in the table.

Following this is the pulse output frequency; this is the frequency that is set until the HSC matches the comparison value.

Parameter d)- The test device

This is the high speed counter that is tested by the DHSZ instruction. Each time the counter counts, its new value is checked against the current entry in the table. When a match is made the table current entry is moved on to the next and the PLSY frequency is set from the new table entry using D8132. Only the High Speed Counters can be used with this function; (C235 to C255).

Note: The correct HSC circuit must also be programmed.

Figure 4: Minimum program required for HSC C251

Parameter e)- The destination device

This parameter normally specifies the output devices to be set for the comparison. In this case the specification of special auxiliary relay M8132 indicates that a table is to be used to define the comparison values and set the pulse frequencies.

There can only be 1 DHSZ instruction using the flag M8132 at any one time; i.e. only one switched on.

High Speed Pulse Control Operation

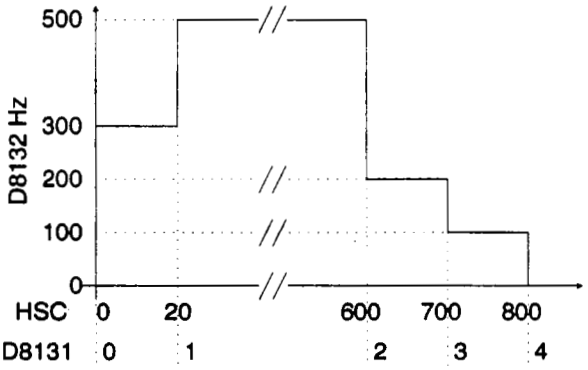


Figure 5: Graph showing the frequency set as the HSC reaches each value in the table.

When M8131 is used with DHSZ special data register D8131 is defined as the current entry counter. As each comparison value is matched D8131 increments to the next entry number to be compared.

The special data register D8132 is set to the frequency of the current entry. When this is used with the PLSY instruction the pulse output can be controlled.

To help monitor the process the data register pair D8134 and D8135 store the value of the current comparison value.

When the final comparison is made the completion flag M8133 is set on and the table counter cycles back to the first entry.

When the DHSZ instruction is switched off all values are reset; including the frequency output.

FX

MITSUBISHI
PROGRAMMABLE CONTROLLERS

ZONE COMPARE INSTRUCTION (HSZ, FNC 55) USED FOR PULSE CONTROL WITH FLAG M8132

Important Notes

Only 6 high speed comparison functions can be active at any one time. High speed comparisons include DHSCS, DHSCR and all forms of DHSZ. Any number may be in the program but only 6 can be switched on.

When a high speed comparison is first switched on it will not be active until after the first END is executed. This is to allow the FX to initialize its internal comparison table. This will only be a problem if the high speed counter current value is close to the compare value at the time the instruction is switched on.

This instruction is interrupt processed and only operates when a signal is received at the HSC input. Therefore, changing the HSC current value or resetting the HSC from within the FX program will not be recognized until the next pulse is received at the HSC input.

The comparison values are checked in a sequential order. Take care to ensure the comparison values always increment or decrement, following the HSC operation.

Because frequency output is commonly used to control active machinery it is recommended that the final entry in the table is set to K0,K0. This will ensure that the pulse output is stopped and the table counter will not return to the beginning. If this is done then monitoring special register pair D8134,D8134 for a value of K0 will indicate the end of the table.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE HEXADECIMAL KEY INPUT INSTRUCTION (HKY, FNC 71) USED FOR HEXADECIMAL INPUT

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic set of PC instructions and are used to provide control over internal PC operations and external I/O. They are separated into 10 groups, each group having a different emphasis, eg. arithmetic and logic.

Applied instructions 70 to 79 are grouped together under the heading of external Input/Output devices and provide facilities to pass information to and from external equipment and peripherals.

Included in these I/O instructions is the Hexadecimal KeY instruction (HKY, FNC 71). This instruction uses inputs and outputs in a multiplexing operations to read the key presses from a 16 key keypad. The number keys when pressed are recorded in sequence in a data register as a hexadecimal value and the six alpha keys, as well as being data inputs, signal a bit device to be set on for use as a special function key.

The Hexadecimal Key Input

The HKY input instruction has seven parameters:

a) setting the Hexadecimal mode flag.

b) the HKY statement itself.

c) the input head address; the first of the inputs to which the key signals are connected.

d) the output head address; the first of the outputs to which the multiplexing control signals are connected.

e) the numerical storage register; the data register(s) where the entered number is placed.

f) the function key head address; the first of the bit devices used for the function key operation.

g) resetting the hexadecimal mode flag.

Shown above is the typical appearance of an HKY instruction in both ladder and instruction format.

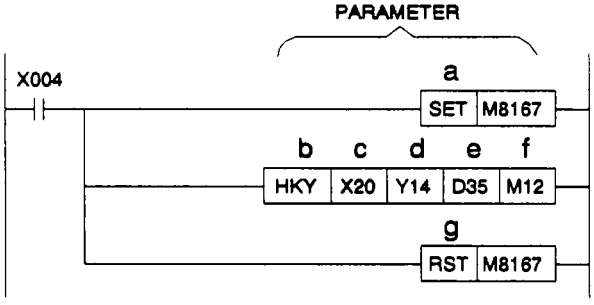


Figure 1: Ladder program of the basic format for the HKY instruction.

PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LD	X	4
1	SET	M	8167
3	HKY		71
		X	20
		Y	14
		D	35
		M	12
12	RST	M	8167

Figure 2: Instruction program of the basic format for the HKY instruction.

The above figures show the typical appearance of the hexadecimal keypad input function in instruction and ladder program formats, When executed using input X4 this program sets each of the outputs Y14 to Y17 on in turn and reads the inputs X20 to X23 to test for a key press. Either the number pressed (0 to 9) is shifted into the data register D35 or the appropriate function flag M12 to M18 is set for the alpha key pressed (A to F). The following section explains each parameter in greater detail.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE HEXADECIMAL KEY INPUT INSTRUCTION (HKY, FNC 71) USED FOR HEXADECIMAL INPUT

Parameter a)- Setting Hexadecimal Mode

The HKY hexadecimal mode key is needed to swithc the operation of the HKY instruction to interpret all 16 key as hexadeecimal data input.

Parameter b)- The HKY statement

The HKY statement can take two different forms:

I) HKY: The standard form of the instruction, taking 9 steps of program. When activated the function operates continuously using the other parameters given.

II) DHKY: This is similar to HKY, but the "D" prefix indicates the instruction stores the number in a DOUBLE word of data, ie. 32 bits, equivalent to two data registers.

Parameter c)- Input Head Address

This identifies the first of the four inputs used to read the data entered using the keypad. Only an input (X) can be specified for this parameter. A total of four consecutive inputs will be allocated.

Parameter d)- Output Head Address

This identifies the first of the four outputs used to scan the keys pressed on the keypad. Combined with the inputs, a grid is made and the outputs are sequential set and reset to trap a closed switch at one of the inputs.

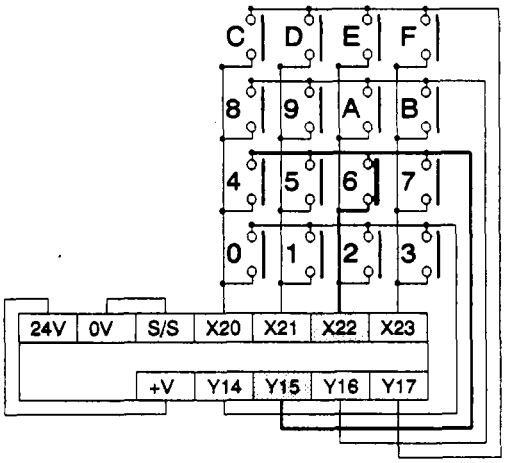


Figure 3: The Inputs and the outputs are connected to form the hexadecimal keypad.
When output Y15 is on key 6 is detected at input X22.

Each of the outputs are switched on in turn, activating one of the rows of keys. If a key is pressed then the corresponding input is also turned on and the key press can be identified. Only outputs (Y) can be specified for this parameter. A total of four outputs will be allocated.

Parameter e)- Numerical Storage Register

This identifies the data register (first of a pair if DOUBLE word is used) where the hexadecimal value entered on the keypad is stored. As each key is pressed the numbers 0 to 9 and the letters A to F are recorded in this data register and a number up to FFFFHEX (FFFFFFFFHEX for DOUBLE word) can be entered.

When a key is pressed the contents of the data register are shifted one place to the left (multiplied by 10HEX) and the new number is entered in the least significant digit (units) position. The following diagram shows this operation.

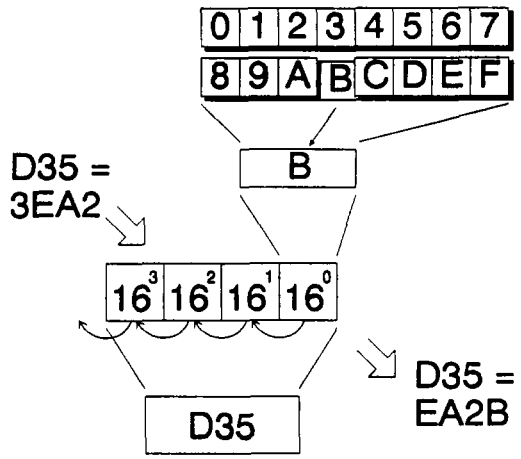


Figure 4: Number Key entry into the storage data register.

The number will overflow if the number of digits exceeds the maximum of 4 (8 if DOUBLE word).

THE HEXADECIMAL KEY INPUT INSTRUCTION (HKY, FNC 71) USED FOR HEXADECIMAL INPUT

Parameter f) – Function Key Head Address

This identifies the first of a group of 8 bit device used to record the function key presses and control signals.

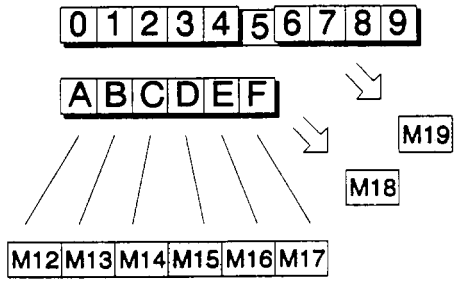


Figure 5: Function keys and key press flags.
M15: D has been pressed.
M19: A number (5) is now being pressed.

The first six correspond to the alpha keys pressed, the next is on while an alpha key is pressed and the last is on while a number key is pressed.

If a function key flag is already set and a different key is pressed the first flag is reset and the new key pressed identified by its function flag being set on.

After any key press has been sensed, the execution complete flag M8029 is set on.

Operation Speed

The HKY instruction is processed during each execution of the ladder program. This means that one full reading of the key pad will take 8 program scan to execute (i.e. one complete ON/OFF cycle for each output). After each output is set ON the inputs need to be processed to obtain the key signal. Therefore the response time of the HKY instruction is dependant upon the execution speed of the program and the limiting speed is determined by the response time of the inputs.

For fast programs it is possible that the inputs might not respond in time with the outputs because of input filtering delays. This can be controlled using constant scan mode and setting the scan time to more than 20ms (input filters are approx. 10ms ON/OFF).

If the normal scan time proves to be too slow then use of a timer interrupt is possible. When a timer interrupt routine is used it is necessary to refresh the inputs before execution and the outputs after execution of the HKY instruction. This is done using the REF instruction.

Note: It is also necessary to set the interrupt timer to be a little more than the input refresh time; 15ms or greater is recommended for normal inputs, 10ms for high speed inputs if the refresh time is reduced.

The following program is one example of how to do this.

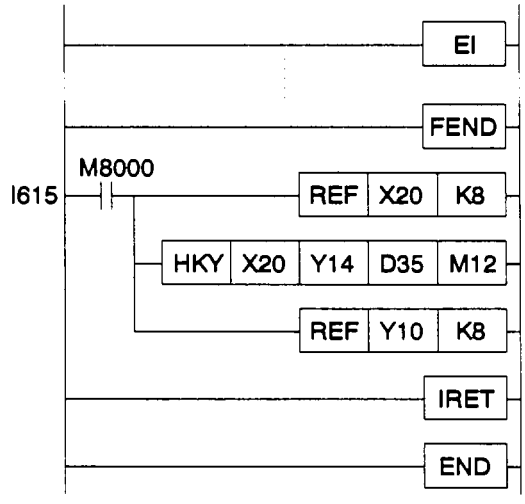


Figure 6: Using the HKY instruction in an interrupt routine to speed up the input response.

If the use of ordinary inputs is too slow then it is possible to use the high speed inputs X0 to X7 (if they are not already used for another purpose). With the filter time set to approx 7ms (REFF instruction) and used with the timer interrupt routine, as explained above, the interrupt processing time can be set to 10ms for the HKY instruction.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE ADD INSTRUCTION (ADD, FNC 20) USED WITH THE M8023 FLOAT OPERATION FLAG

Introduction

The FX PC family instruction set is divided into basic and applied instructions. The applied instructions provide powerful supplements to the basic set of PC instructions and give control over internal PC operations and external I/O. They are split into 10 groups, each group with a different application, eg. transfer of data within the PC.

Applied instructions 20 to 29 are concerned with Arithmetic and Logical Operations and are grouped under the heading of the same name.

This group includes the Add instruction (ADD, FNC 20). If Add is used with Auxiliary Relay M8023 (Float Operation Flag), then addition of floating point values is possible. M8023 must be turned on before the Add instruction is executed to allow this. This provides the facility to take floating point data from Data Registers, or Decimal and Hexadecimal constants and add similar data from a second source to it. The result is then stored in a destination device as a floating point value.

The Add Instruction with M8023

The floating point form of the Add instruction has three parts:

Part 1: Set M8023

a) turn on M8023 to enable floating point operation.

Part 2: The Add instruction.

b) the ADD statement itself

c) the specification for the first Source Data, ie. the form of the data.

d) the specification for the second Source Data.

e) the specification for the Destination Device, where the result will be stored.

Part 3: Reset M8023

f) turn off M8023 to disable floating point operation.

Figure 1 shows the typical appearance of a floating point Add instruction in a ladder program, while Figure 2 shows the same piece of ladder logic written in instruction format. This Add instruction would have the effect of adding the floating point value stored in Data Register pair D101 and D100 to the floating point value stored in D111 and D110 when Auxiliary Relay M100 is turned on. The floating point result would then be stored in D121 and D120. The following section explains each parameter in greater detail:

Parameter a)- Setting the float flag

The Float Operation flag M8023 must be SET to enable floating point calculations.

Parameter b)- The ADD statement

The Add instruction can only be used in its DOUBLE word format if floating point values are being used. This is because floating point values are stored in a pair of Data Registers. An error will occur if this form is not used. This gives two different forms for the instruction:

I) DADD: The standard form of the instruction, taking 13 steps of program. In the example shown in Figure 1, while M100 is turned on, the DADD instruction will operate.

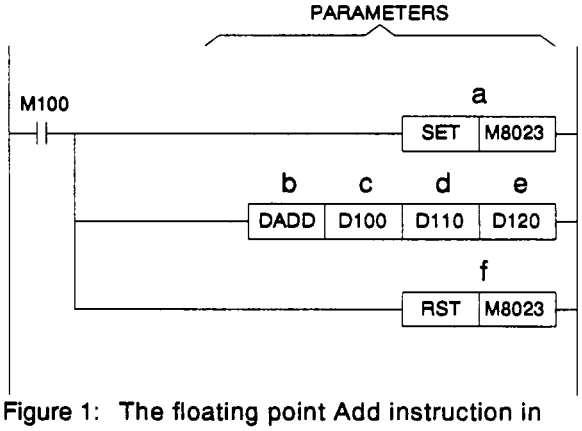


Figure 1: The floating point Add instruction in ladder format.

PROGRAM STEP NUMBER	INSTRUCTION PROGRAM		
0	LD	M	100
1	SET	M	8023
3	DADD		20
		D	100
		D	110
		D	120
16	RST	M	8023

Figure 2: The floating point Add instruction in instruction format.

FX2

THE ADD INSTRUCTION (ADD, FNC 20) USED WITH THE M8023 FLOAT OPERATION FLAG

II) DADDP: The "P" suffix shows that DADD is being used in its PULSE format. The instruction will only operate when M100 provides a rising edge signal as it turns on. DADDP also takes 13 program steps.

Parameter c)- Source Data 1

This identifies what form the Source Data takes.

The data can be one of the following: Decimal Values (K), Hexadecimal Values (H) or a pair of Data Registers (D) containing a floating point value. If K or H is used, then these values will be converted to floating point automatically.

Parameter d)- Source Data 2

This identifies the second Source Data, which can be one of these forms:

Decimal Values (K), Hexadecimal Values (H) or Data Registers (D) holding a floating point value. Again, if K or H is used, then these values will be converted to floating point automatically.

Parameter e)- Destination Devices

This identifies where the result will be stored. The Destination Devices can only be a pair of Data Registers, which store the sum as a floating point result.

Notes: The Destination Devices may be the same as one of the Source Devices. In this case, be careful to avoid unintentionally overwriting data.

The following combinations of Source Data are allowed:

Source Data 1	Source Data 2
K/H	K/H
Floating Point D	Floating Point D
K/H	Floating Point D
Floating Point D	K/H

Parameter f)- Resetting the float flag

The Float Operation Flag must be reset to allow further functions to operate as normal. The flag can be left on if further operations requiring floating point calculations are needed. However, it is recommended that once all floating point functions are completed this flag should be switched off using RST.

Example Use of the Add Instruction with M8023

Figure 3 shows a typical use of the floating point Add instruction. In this case, DADDP is being used to add a fixed offset, K1000, to the floating point value held in D11 and D10 when T1 times out . The floating point sum is then stored in Data Registers D1 and D0. Using the PULSE instruction format ensures that the instruction will only execute once after T1's contacts close. Notice how the floating point operation is enabled before the Add instruction with SET and disabled afterwards with ReSeT.

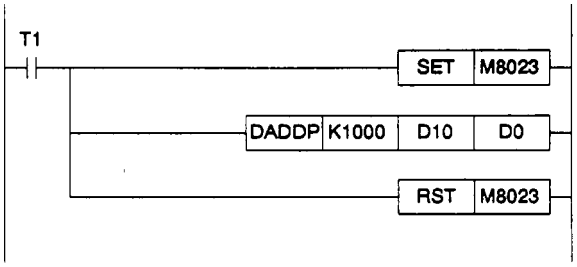


Figure 3: Example ladder diagram for the floating point Add instruction.

THE SUBTRACT INSTRUCTION (SUB, FNC 21) USED WITH THE M8023 FLOAT OPERATION FLAG

Introduction

The FX PC family instruction set is composed of two sets of instructions; basic and applied. Applied instructions give control over internal PC operations and external I/O and provide powerful supplements to the basic set of PC instructions. They are split into 10 groups, each group with a different application, eg. transfer of data within the PC.

Applied instructions 20 to 29 perform Arithmetic and Logical Operations and are grouped under the heading of the same name.

This group includes the Subtract instruction (SUB, FNC 21). If Subtract is used with Auxiliary Relay M8023 (Float Operation Flag), then subtraction of floating point values is possible. M8023 must be turned on before the Subtract instruction is executed to allow this. This provides the facility to take floating point data from Data Registers, or Decimal and Hexadecimal constants and subtract similar data from a second source from it. The result is then stored in a destination device as a floating point value.

The Subtract Instruction with M8023

The floating point form of the Subtract instruction has three parts:

Part 1: Set M8023

a) turn on M8023 to enable floating point operation.

Part 2: The Subtract instruction.

b) the SUB statement itself

c) the specification for the first Source Data, ie. the form of the data.

d) the specification for the second Source Data.

e) the specification for the Destination Device, where the result will be stored.

Part 3: Reset M8023

f) turn off M8023 to disable floating point operation.

Figure 1 shows the typical appearance of a floating point Subtract instruction in a ladder program, while Figure 2 shows the same piece of ladder logic written in instruction format. This Subtract instruction would have the effect of subtracting the floating point value stored in Data Register pair D21 and D20 from the floating point value stored in D11 and D10 when State S100 is turned on. The floating point result would then be stored in D31 and D30. The following section explains each parameter in greater detail:

Parameter a)- Setting the float flag

The Float Operation flag M8023 must be SET to enable floating point calculations.

Parameter b)- The SUB statement

The Subtract instruction can only be used in its DOUBLE word format if floating point values are being used. This is because floating point values are stored in a pair of Data Registers. An error will occur if this form is not used. This gives two different forms for the instruction:

I) DSUB: The standard form of the instruction, taking 13 steps of program. In the example shown in Figure 1, while S100 is turned on, the DSUB instruction will operate.

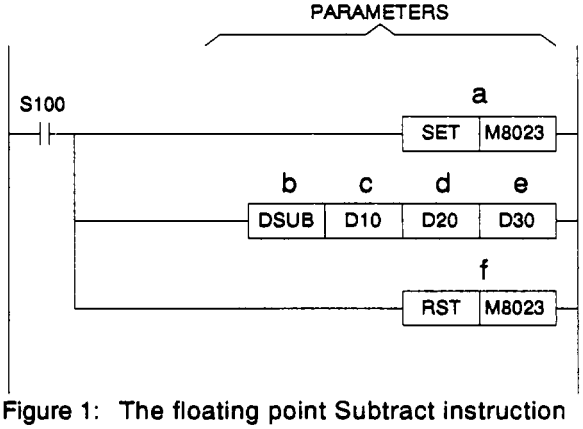


Figure 1: The floating point Subtract instruction in ladder format.

PROGRAM STEP NUMBER	INSTRUCTION PROGRAM		
0	LD	S	100
1	SET	M	8023
3	DSUB		21
		D	10
		D	20
		D	30
16	RST	M	8023

Figure 2: The floating point Subtract instruction in instruction format.

THE SUBTRACT INSTRUCTION (SUB, FNC 21) USED WITH THE M8023 FLOAT OPERATION FLAG

II) DSUBP: The "P" suffix shows that DSUB is being used in its PULSE format. The instruction will only operate when S100 provides a rising edge signal as it turns on. DSUBP also takes 13 program steps.

Parameter c)- Source Data 1

This identifies what form the Source Data takes.

The data can be one of the following: Decimal Values (K), Hexadecimal Values (H) or a pair of Data Registers (D) containing a floating point value. If K or H is used, then these values will be converted to floating point automatically.

Parameter d)- Source Data 2

This identifies the second Source Data, which can be one of these forms:

Decimal Values (K), Hexadecimal Values (H) or Data Registers (D) holding a floating point value. Again, if K or H is used, then these values will be converted to floating point automatically.

Parameter e)- Destination Devices

This identifies where the result will be stored. The Destination Devices can only be a pair of Data Registers, which store the result as a floating point value.

Notes: The Destination Devices may be the same as one of the Source Devices. In this case, take care to avoid accidentally overwriting data.

The following combinations of Source Data are allowed:

Source Data 1	Source Data 2
K/H	K/H
Floating Point D	Floating Point D
K/H	Floating Point D
Floating Point D	K/H

Parameter f)- Resetting the float flag

The Float Operation Flag must be reset to allow further functions to operate as normal. The flag can be left on if further operations requiring floating point calculations are needed. However, it is recommended that once all floating point functions are completed this flag should be switched off using RST.

Example Use of the Subtract Instruction with M8023

Figure 3 shows a typical use of the floating point Subtract instruction. For this example, DSUB is being used to subtract the floating point value held in D1 and D0 from that stored in D26 and D25 . The floating point result is then stored in Data Registers D26 and D25 again. This instruction will execute until C20's contacts close. Notice how the floating point operation is enabled before the Subtract instruction with SET and disabled afterwards with ReSeT.

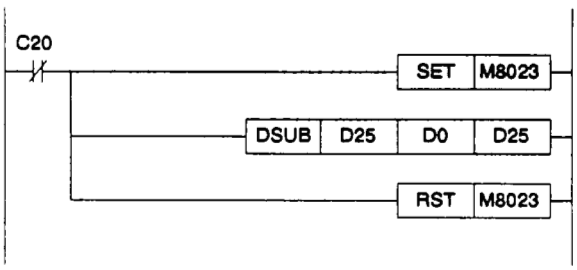


Figure 3: Example ladder diagram for the floating point Subtract instruction.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE MULTIPLY INSTRUCTION (MUL, FNC 22) USED WITH THE M8023 FLOAT OPERATION FLAG

Introduction

The FX PC family has both basic and applied instructions. Applied instructions control both internal PC operations and external I/O and are powerful supplements to the basic PC set of instructions. They are split into 10 groups, each with a different application, eg. transfer of data within the PC.

Applied instructions 20 to 29 perform Arithmetic and Logical Operations and are grouped together under the same heading .

This includes the Multiply instruction (MUL, FNC 22). If Multiply is used with Auxiliary Relay M8023 (Float Operation Flag), then multiplication of floating point values is possible. M8023 must be turned on before the Multiply instruction is executed to allow this. This provides the facility to take floating point data from Data Registers, or Decimal and Hexadecimal constants and multiply it by similar data from a second source. The result is then stored in a destination device as a floating point value.

The Multiply Instruction with M8023

The floating point form of the Multiply instruction has three parts:

Part 1: Set M8023

a) turn on M8023 to enable floating point operation.

Part 2: The Multiply instruction.

b) the MUL statement itself

c) the specification for the first Source Data, ie. the form of the data.

d) the specification for the second Source Data.

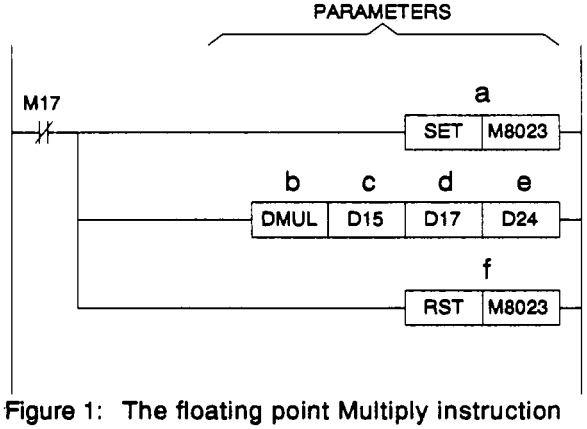


Figure 1: The floating point Multiply instruction in ladder format.

e) the specification for the Destination Device, where the result will be stored.

Part 3: Reset M8023

f) turn off M8023 to disable floating point operation.

Figure 1 shows the typical appearance of a floating point Multiply instruction in a ladder program, while Figure 2 shows the same piece of ladder logic written in instruction format. This Multiply instruction would have the effect of multiplying the floating point value stored in Data Register pair D16 and D15 by the floating point value stored in D18 and D17 when Auxiliary Relay M17 is turned off. The floating point result would then be stored in D25 and D24. Finally, the floating point operation is ended by resetting M8023. The following section explains each parameter in greater detail:

Parameter a)- Setting the float flag

The Float Operation flag M8023 must be SET to enable floating point calculations.

Parameter b)- The MUL statement

The Multiply instruction can only be used in its DOUBLE word format if floating point values are being used. This is because floating point values are stored in a pair of Data Registers. An error will occur if this form is not used. This gives two different forms for the instruction:

I) DMUL: The standard form of the instruction, taking 13 steps of program. In the example shown in Figure 1, while M17 is turned off, the DMUL instruction will operate.

PROGRAM STEP NUMBER	INSTRUCTION PROGRAM		
0	LDI	M	17
1	SET	M	8023
3	DMUL		22
		D	15
		D	17
		D	24
16	RST	M	8023

Figure 2: The floating point Multiply instruction in instruction format.

FX2

THE MULTIPLY INSTRUCTION (MUL, FNC 22) USED WITH THE M8023 FLOAT OPERATION FLAG

II) DMULP: The "P" suffix shows that DMUL is being used in its PULSE format. The instruction will only operate when M17 provides a falling edge signal as it turns off. DMULP also takes 13 program steps.

Parameter c)- Source Data 1

This identifies what form the Multiplicand, i.e. the value to be multiplied, takes.

The data can be one of the following: Decimal Values (K), Hexadecimal Values (H) or a pair of Data Registers (D) containing a floating point value. If K or H is used, then these values will be converted to floating point automatically.

Parameter d)- Source Data 2

This shows the form of the Multiplier, i.e. the value by which the Multiplicand is multiplied. This can be one of these forms:

Decimal Values (K), Hexadecimal Values (H) or Data Registers (D) holding a floating point value. Again, if K or H is used, then these values will be converted to floating point automatically.

Parameter e)- Destination Devices

This identifies where the result will be stored. The Destination Devices can only be a pair of Data Registers, which store the result as a floating point value.

Notes:

The result of the floating point version of the multiply instruction is different to the normal 32 bit Multiply instruction. The Destination Devices hold the complete result as a 32 bit floating point value, rather than a 64 bit result.

The Destination Devices may be the same as one of the Source Devices. In this case, take care to avoid accidentally overwriting data.

The allowed combinations of Source Data are shown in Figure 3.

Parameter f)- Resetting the float flag

The Float Operation Flag must be reset to allow further functions to operate as normal. The flag can be left on if further operations requiring floating point calculations are needed. However, it is recommended that once all floating point functions are completed this flag should be switched off using RST.

Source Data 1	Source Data 2
K/H	K/H
Floating Point D	Floating Point D
K/H	Floating Point D
Floating Point D	K/H

Figure 3: Allowed combinations of Source Data.

Example Use of the Multiply Instruction with M8023

Figure 4 shows a typical use of the floating point Multiply instruction. For this example, DMUL is being used to multiply the decimal value 250 by the floating point value held in D51 and D50. The floating point result is then stored in Data Registers D1 and D0. This instruction will execute when C20's contacts close. Additionally, this shows how the floating point operation is enabled before the Multiply instruction with SET and disabled afterwards with ReSeT.

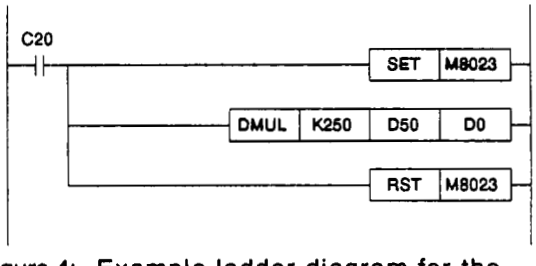


Figure 4: Example ladder diagram for the Multiply floating point instruction.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE DIVIDE INSTRUCTION (DIV, FNC 23) USED WITH THE M8023 FLOAT OPERATION FLAG

Introduction

The FX PC family has two sets of instructions; basic and applied. Applied instructions control internal PC operations and external I/O and are powerful supplements to the basic PC instructions. They are divided into 10 groups, each group with a different application, eg. transfer of data within the PC.

Applied instructions 20 to 29 perform Arithmetic and Logical Operations and are grouped together under this heading .

This includes the Divide instruction (DIV, FNC 23). If Divide is used with Auxiliary Relay M8023 (Float Operation Flag), then division of floating point values is possible. M8023 must be turned on before the Divide instruction is executed to allow this. This provides the facility to take floating point data from Data Registers, or Decimal and Hexadecimal constants and divide it by similar data from a second source. The result is then stored in a destination device as a floating point value.

The Divide Instruction with M8023

The floating point form of the Divide instruction has three parts:

Part 1: Set M8023

a) turn on M8023 to enable floating point operation.

Part 2: The Divide instruction.

b) the DIV statement itself

c) the specification for the first Source Data, ie. the form of the data.

d) the specification for the second Source Data.

e) the specification for the Destination Device, where the result will be stored.

Part 3: Reset M8023

f) turn off M8023 to disable floating point operation.

Figure 1 shows the typical appearance of a floating point Divide instruction in a ladder program, while Figure 2 shows the same piece of ladder logic written in instruction format. This Divide instruction would have the effect of dividing the floating point value stored in Data Register pair D29 and D28 by the floating point value stored in D31 and D30 when Auxiliary Relay M40 is turned off. The floating point result would then be stored in D41 and D40. Finally, the floating point operation is ended by resetting M8023. The following section explains each parameter in greater detail:

Parameter a)- Setting the float flag

The Float Operation flag M8023 must be SET to enable floating point calculations.

Parameter b)- The DIV statement

The Divide instruction can only be used in its DOUBLE word format if floating point values are being used. This is because floating point values are stored in a pair of Data Registers. An error will occur if this form is not used. This gives two different forms for the instruction:

I) DDIV: The standard form of the instruction, taking 13 steps of program. In the example shown in Figure 1, while M40 is turned off, the DDIV instruction will operate.

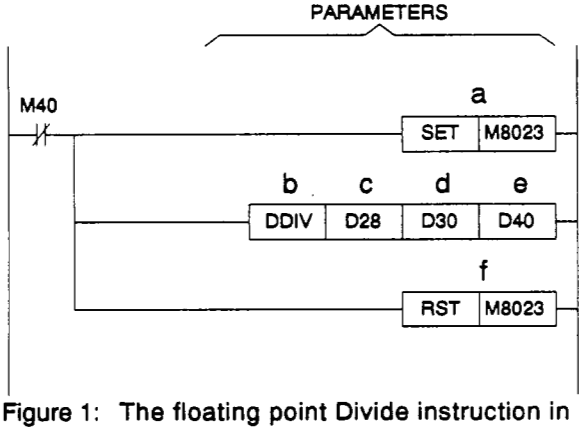


Figure 1: The floating point Divide instruction in ladder format.

PROGRAM STEP NUMBER	INSTRUCTION PROGRAM		
0	LDI	M	40
1	SET	M	8023
3	DDIV		23
		D	27
		D	30
		D	40
16	RST	M	8023

Figure 2: The floating point Divide instruction in instruction format.

FX2

THE DIVIDE INSTRUCTION (DIV, FNC 23) USED WITH THE M8023 FLOAT OPERATION FLAG

II) DDIVP: The "P" suffix shows that DDIV is being used in its PULSE format. The instruction will only operate when M40 provides a falling edge signal as it turns off. DDIVP also takes 13 program steps.

Parameter c)- Source Data 1

This identifies what form the Dividend, i.e. value to be divided, takes.

The data can be one of the following: Decimal Values (K), Hexadecimal Values (H) or a pair of Data Registers (D) containing a floating point value. If K or H is used, then these values will be converted to floating point automatically.

Parameter d)- Source Data 2

This shows the form of the Divisor, i.e. the value by which the dividend is divided. This can be one of these forms:

Decimal Values (K), Hexadecimal Values (H) or Data Registers (D) holding a floating point value. Again, if K or H is used, then these values will be converted to floating point automatically.

Parameter e)- Destination Devices

This identifies where the result will be stored. The Destination Devices can only be a pair of Data Registers, which store the result as a floating point value.

Notes:

The result of the floating point version of the divide instruction is different to the normal Divide instruction. The Destination Devices hold the complete result and there is no separation of the whole (Quotient) and fractional (Remainder) parts.

The Destination Devices may be the same as one of the Source Devices. In this case, take care to avoid accidentally overwriting data.

The allowed combinations of Source Data are shown in Figure 3.

Parameter f)- Resetting the float flag

The Float Operation Flag must be reset to allow further functions to operate as normal. The flag can be left on if further operations requiring floating point calculations are needed. However, it is recommended that once all floating point functions are completed this flag should be switched off using RST.

Source Data 1	Source Data 2
K/H	K/H
Floating Point D	Floating Point D
K/H	Floating Point D
Floating Point D	K/H

Figure 3: Allowed combinations of Source Data.

Example Use of the Divide Instruction with M8023

Figure 4 shows an example use of the floating point Divide instruction. For this example, DDIV is being used to divide the floating point value held in D1 and D0 by the decimal value 1000 . The floating point result is then stored in Data Registers D11 and D10. This instruction will execute when T0's contacts close. This also shows how the floating point operation is enabled before the Divide instruction with SET and disabled afterwards with ReSeT.

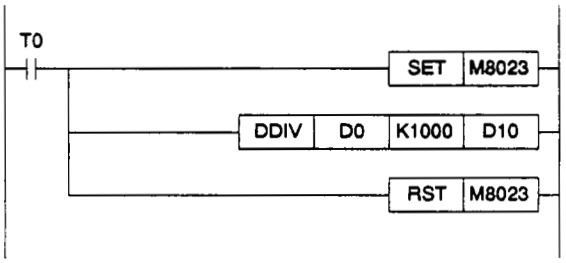


Figure 4: Example ladder diagram for the floating point Divide instruction.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE SERIAL COMMUNICATION INSTRUCTION (RS, FNC 80)

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic set of PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. transfer of data within the PC.

Applied instructions 80 to 89 are concerned with the communication of data to devices outside the FX plc and are grouped under the heading of FX Service Interfaces.

This group includes the Serial Communications instruction (RS, FNC 80). This provides the facility to send and receive data to/from a wide range of communication devices via the RS232 serial communications module; FX-232ADP.

The RS Instruction

The serial communications is controlled in four parts:

Part 1: Set Communication parameters

Part 2: The RS instruction (six parameters)

a) the data storage format.

b) the RS statement itself.

c) the transmit buffer Head Address.

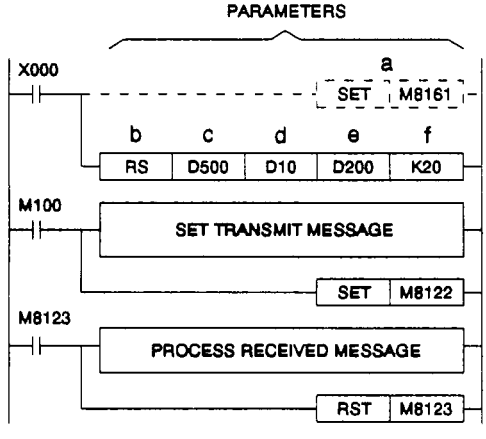
d) the transmit message length.

e) the receive buffer Head Address.

f) the receive message length.

Part 3: Transmit message

Part 4: Receive message



Part 1: Communication Parameters

Any serial communication protocol must first be configured to ensure full compatibility with the external communicating device. The communication protocol for the FX-232ADP is configured using special data register D8120; this can only be done while the RS instruction is inactive. The following table shows the bits of D8120 and their meaning for RS232 communications.

	DESCRIPTION	0	1
b0	DATA LENGTH	7 bits	8 bits
b1 b2	PARITY	(00) : No Parity (01) : Odd Parity (11) : Even Parity	
b3	STOP BITS	1 bit	2 bits
b4 b5 b6 b7	BAUD RATE (BPS)	(b7,b6,b5,b4) (0011) 300 (0100) 600 (0101) 1200 (0110) 2400 (0111) 4800 (1000) 9600 (1001) 19200	
b8	HEADER	NONE	USE D8124
b9	TERMINATOR	NONE	USE D8125
b10	HANDSHAKE	NONE	H/W§
b11 to b15	NOT USED	–	–

Special data register D8124 holds the value of the header byte, if one is selected. The default value is ASCII "STX" or 02HEX but this can be changed by the user before any communications begin.

Special data register D8125 holds the value of the terminator byte, if one is selected. The default value is ASCII "ETX" or 03HEX but this can be changed by the user before any communications begin.

§ If the peripheral communication device uses hardware handshaking then this mode should be selected. When selected, the DSR and DTR lines (pins 6 and 20) of the FX-232ADP are used to control the communication. See page 4 for terminal diagram.

AN INTRODUCTION TO THE SERIAL COMMUNICATION INSTRUCTION (RS, FNC 80)

Part 2: The RS instruction

Parameter a)

The data storage format can be either 16 bit or 8 bit mode. The 16 bit mode uses both the upper and lower bytes of the transmit and receive buffer areas and the 8 bit mode only uses the lower 8 bits. This is controlled by the special auxiliary relay M8161; ON for 8 bit mode.

BUFFER D200 K4 DATA "ABCD"

16 bit (default)

	High	Low
D200	"B"	"A"
D201	"D"	"C"

8 bit

	High	Low
D200		"A"
D201		"B"
D202		"C"
D203		"D"

Parameter b)

The RS statement, when active, indicates that communication is possible and the transmission and receival of data can occur. More than one RS instruction is permitted within a program but only one may be active at a time.

Parameter c)

The transmit buffer Head Address is the first data register or file register (D) of the transmit message area.

Parameter d)

The transmit message length is the length of the message to be transmitted. The value can be a constant (K) or if the message length will, vary a data register (D) can also be used. The value, if a data register, can be changed between transmit requests but not during transmission.

Parameter e)

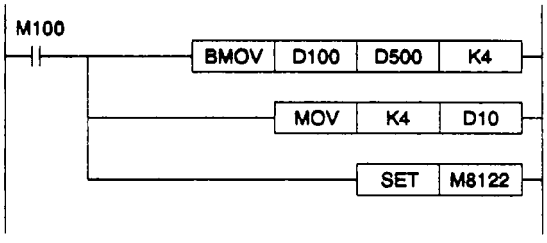
The receive buffer Head Address is the first data register (D) of the receive message area.

Parameter f)

The receive message length is the maximum length of the message that can be received. The value can be a constant (K) or if the message length will vary a data register (D) can also be used. The value, if a data register, can be changed between receipt of messages but not during receival.

Part 3: Transmit message

The transmission or sending of a message is controlled with auxiliary relay M8122.



First the data to be transmitted must be contained in the transmit buffer area. This can be done in one of two ways:

I) Copy or create the message into the message buffer area before transmitting using either MOV or BMOV instructions.

II) Change the RS instruction parameters to use the appropriate data register area where the message is held. A separate RS instruction will be needed for each message.

In the above example the data held in data registers D100 to D103 is copied to the transmit buffer area starting at D500. The message length of 8 bytes is then set by changing the value of the transmit message length using data register D10.

Once the data is defined and in the correct location the transmit flag M8122 can be set ON. The data will now be sent and the flag M8122 will be automatically reset once all data is transmitted.

It is recommended that the flag is set using a pulse signal; otherwise, after completion, the flag may be set back ON again by the program and the data sent again.

If a header and/or a terminator are used these will be automatically added to the message before transmission.

Transmit Counter

During sending, the special data register D8122 can be monitored to see the progress of the transmission. The value of D8122 starts at the full message length and decreases by one as each byte of data is transmitted.

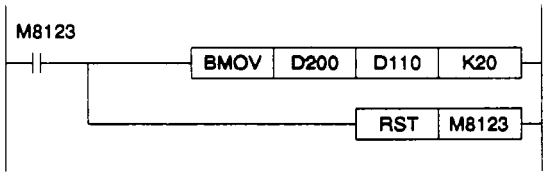
Note: Headers and terminators are not counted by D8122.

AN INTRODUCTION TO THE SERIAL COMMUNICATION INSTRUCTION (RS, FNC 80)

Part 4: Receive message

The receipt of a message is controlled by the RS instruction automatically. Once a full message has been received the data is stored in the receive buffer area and the special auxiliary relay M8123 is set ON.

If a header and/or a terminator are used then these will be removed automatically before the message is stored in the buffer area.



When M8123 comes ON, the data in the receive buffer should be processed and then, to free the area up for further messages, the flag should be reset to OFF. The flag will be automatically reset if the RS instruction is turned off.

In the example program above the receive message flag M8123 is monitored. When it becomes active all the data in the receive buffer is copied to a separate location and the receive flag is reset. After freeing the receive buffer area the data can be processed as appropriate.

Note: Data can not be sent while receiving a message. The receive in progress flag M8121 is set ON during message receival. The transmit flag M8122 may be set to ON at this time but the actual sending will be delayed until after the full message has been received.

Receive Counter

During receipt, the special data register D8123 can be monitored for the number of bytes received so far. After the full message is received its value will be the length of the message.

Headers and Terminators

In communications a way to find the beginning and end of a message ia often needed. This is usually done by adding indicators to the message known as headers and terminators. With the RS instruction it is possible to have an initial header byte and /or a final terminator byte automatically added to the massage.

These are selected by setting bits b8 and b9 of the communications parameters data register D8120.

During Transmit

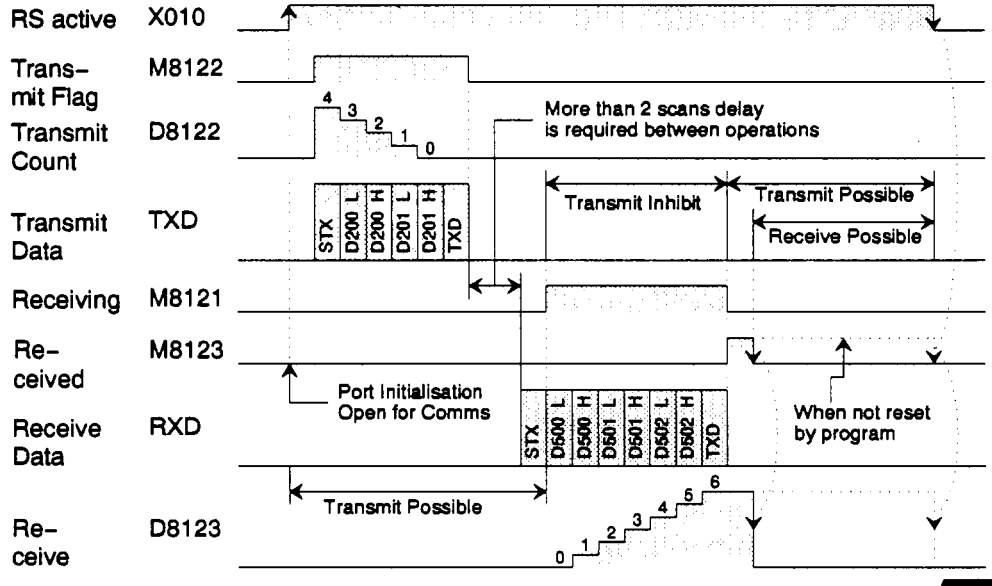
If a header is selected then the lower byte of special data register D8124 will be sent as the first byte of any message transmitted.

If a terminator is selected then the lower byte of special data register D8125 will be sent as the last byte of any message transmitted.

During Receive

If a header is selected then all data received will be ignored until the header byte is received.

If no header is used then the first byte received will be read as message data.



FX2

MITSUBISHI
PROGRAMMABLE CONTROLLERS

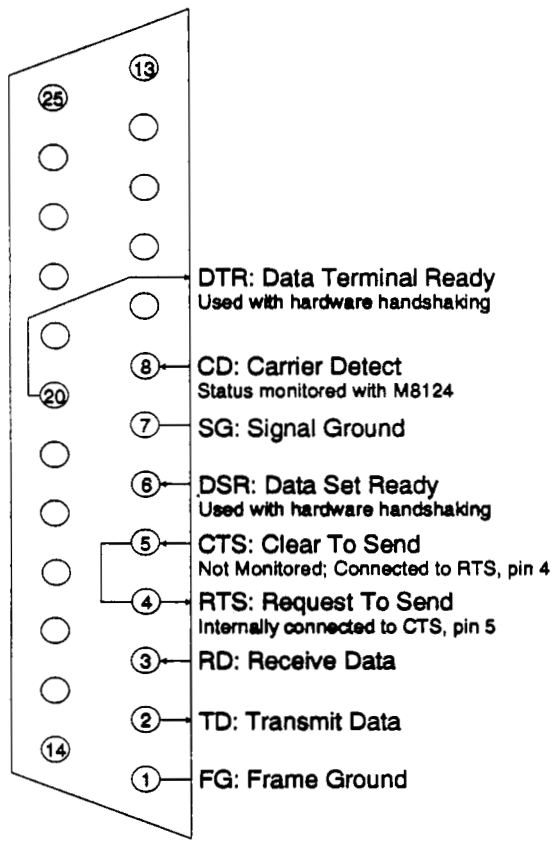
AN INTRODUCTION TO THE SERIAL COMMUNICATION INSTRUCTION (RS, FNC 80)

If a terminator is selected then, once in the process of reading, all data received will be read as message data until either the terminator byte is received or the receive message length is reached i.e. the receive buffer becomes full.

If no terminator is selected then reading will continue until the receive buffer is filled i.e. the message length must be received in full before the message is considered complete.

Once a full message has been received flag M8123 is set. All data received after this is ignored until this message received flag is cleared.

FX-232ADP Connector



MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE HEXADECIMAL TO ASCII CONVERSION INSTRUCTION (ASCI, FNC 82)

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. High speed operations.

Applied instructions 80 to 89 are mainly concerned with the communication of data to devices outside the FX plc and are grouped under the heading of FX Service Interfaces.

This group includes the ASCII conversion instruction (ASCI, FNC 82). This function is intended as a complement to the serial communications instruction (RS, FNC 80) and converts the hexadecimal values held in a data register into ASCII.

The ASCII Conversion Instruction

The ASCII conversion instruction has six parameters:

a) the data storage format.

b) the ASCI statement itself.

c) the source head address; where the hexadecimal data is stored.

d) the destination head address; where the converted ASCII characters will be stored.

e) the number of characters; the number of hexadecimal digits to convert to ASCII characters.

f) the data storage format reset.

Shown below is the typical appearance of an ASCI instruction in both ladder and instruction format.

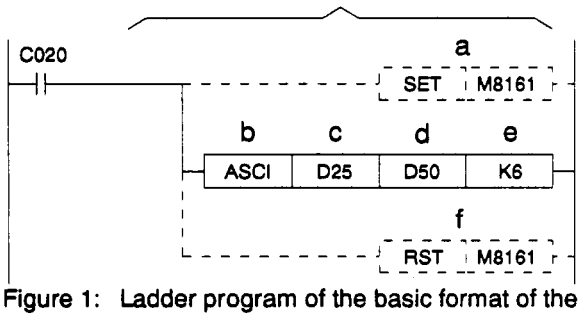


Figure 1: Ladder program of the basic format of the ASCI instruction.

PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LD	C	20
1	SET	M	8161
3	ASCI		82
		D	25
		D	50
		K	6
10	RST	M	8161

Figure 2: Instruction program of the basic format for the ASCI instruction.

The program shown in figures 1 and 2 will execute the conversion when the counter C20 reaches its preset value. While the ASCI instruction is active the six (K6) hexadecimal digits from data registers D25 and D26 are converted to ASCII characters and stored in data registers D50 thru D55 (8 bit format); each character being one byte.

The following section explains each parameter in more detail:

Parameter a)– The Data Storage Format

The data storage format can be either 16 bit or 8 bit mode. The 16 bit mode uses both the upper and lower bytes of the destination devices and the 8 bit mode only uses the lower 8 bits. This is controlled by the special auxiliary relay M8161; ON for 8 bit mode.

Note: The SET M8161 instruction is only needed if 8 bit mode is being used.

Parameter b)– The ASCI Statement

The ASCI instruction has two forms:

I) ASCI: The standard form of the instruction, taking 7 steps of program. In this form the ASCI instruction while active will continue to convert the source data to ASCII character.

II) ASCIP: This form is similar to the standard form and also takes 7 steps of program. The P suffix indicates Pulse format. When activated the ASCIP instruction will only perform the conversion once each time it is turned on.

FX2

AN INTRODUCTION TO THE HEXADECIMAL TO ASCII CONVERSION INSTRUCTION (ASCI, FNC 82)

Parameter c)- The Source Head Address

The value specified here identifies the first word device that contains the hexadecimal digits to be converted. If more than 4 digits are to be converted then the following word devices will also be read until all required digits are converted. The source address can be specified using:

Decimal Value (K), Hexadecimal Value (H), Timers (T), Counters (C), Data registers (D) or
Group bit devices; Inputs (KnX), Outputs (KnY), Auxiliary relays (KnM), States (KnS).

Parameter d)- The Destination Head Address

The value specified here identifies the first word device that will contain the ASCII characters. Each word device can contain 2 characters (2 bytes). The word devices following the Head Address will be used until all the characters have been stored. The destination address can be specified using:

Timers (T), Counters (C), Data registers (D) or
Group bit devices; Outputs (KnY), Auxiliary relays (KnM), States (KnS).

Parameter e)- The Number of Characters

The value specified here can only be a Decimal Value (K) or a Hexadecimal Value (H) and indicates the number of hexadecimal digits that will be converted and how many ASCII characters will be stored. The number of characters can be between 1 and 256 digits.

Parameter f)- The Data Storage Format Reset

The RST M8161 switches the data storage format back to the default of 16 bits.

Note: This parameter is only needed if this instruction uses 8 bit data storage format and there are other instructions in your program that use 16 bit data storage format.

Example Use of the ASCI Instruction

Using the program example in figures 1 and 2 the following diagram shows the results for both 16 bit format and 8 bit format.

Source Data D25 : 0ABC H
D26 : 6789 H

Destination D50 Digits K6

16 bit format

	High	Low
D50	"9"	"8"
D51	"A"	"0"
D52	"C"	"B"

(Default)

8 bit format

	High	Low
D50		"8"
D51		"9"
D52		"0"
D53		"A"
D54		"B"
D55		"C"

Figure 3: Graphic representation of both 16 and 8 bit data storage format.

ASCII Character Codes

The following table gives the ASCII character code in both hexadecimal and decimal for each of the hexadecimal digits that can be converted.

HEX DIGIT	ASCII		SYMBOL	HEX DIGIT	ASCII		SYMBOL
	HEX	DEC			HEX	DEC	
0	30	48	"0"	8	38	56	"8"
1	31	49	"1"	9	39	57	"9"
2	32	50	"2"	A	41	65	"A"
3	33	51	"3"	B	42	66	"B"
4	34	52	"4"	C	43	67	"C"
5	35	53	"5"	D	44	68	"D"
6	36	54	"6"	E	45	69	"E"
7	37	55	"7"	F	46	70	"F"

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE ASCII TO HEXADECIMAL CONVERSION INSTRUCTION (HEX, FNC 83)

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic set of PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. transfer of data within the PC.

Applied instructions 80 to 89 are concerned with the communication of data to devices outside the FX plc and are grouped under the heading of FX Service Interfaces.

This group includes the hexadecimal conversion instruction (HEX, FNC 83). This function is intended as a complement to the serial communications instruction (RS, FNC 80) and converts ASCII character codes held in data registers into hexadecimal values.

The HEX Conversion Instruction

The HEX conversion instruction has six parameters:

a) the data storage format.

b) the HEX statement itself.

c) the source head address; where the ASCII characters are stored.

d) the destination head address; where the converted hexadecimal data will be stored.

e) the number of characters; the number of ASCII characters to convert to hexadecimal digits.

f) the data storage format reset.

Shown below is the typical appearance of a HEX instruction in both ladder and instruction format.

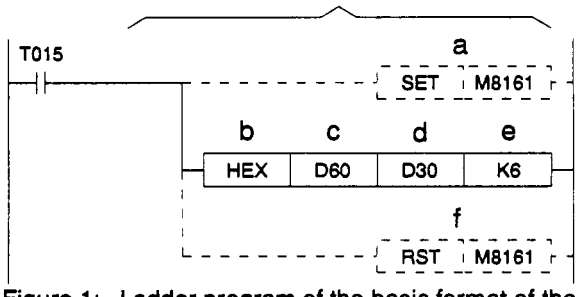


Figure 1: Ladder program of the basic format of the HEX instruction.

PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LD	T	15
1	SET	M	8161
3	HEX		83
		D	60
		D	30
		K	6
10	RST	M	8161

Figure 2: Instruction program of the basic format for the HEX instruction.

The program shown in figures 1 and 2 will execute the conversion when the counter T15 reaches its preset value. While the HEX instruction is active the six (K6) ASCII characters from data registers D60 thru D65 are converted to hexadecimal digits and stored in data registers D30 and D31 (8 bit format); two digits being one byte.

The following section explains each parameter in more detail:

Parameter a)- The Data Storage Format

The data storage format can be either 16 bit or 8 bit mode. The 16 bit mode uses both the upper and lower bytes of the source devices and the 8 bit mode only uses the lower 8 bits. This is controlled by the special auxiliary relay M8161; ON for 8 bit mode.

Note: The SET M8161 instruction is only needed if 8 bit mode is being used.

Parameter b)- The HEX Statement

The HEX instruction has two forms:

I) HEX: The standard form of the instruction, taking 7 steps of program. In this form the HEX instruction while active will continue to convert the source data to hexadecimal digits.

II) HEXP: This form is similar to the standard form and also takes 7 steps of program. The P suffix indicates Pulse format. When activated the HEXP instruction will only perform the conversion once each time it is turned on.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE ASCII TO HEXADECIMAL CONVERSION INSTRUCTION (HEX, FNC 83)

Parameter c)- The Source Head Address

The value specified here identifies the first word device that contains the ASCII characters to be converted. If more than 2 characters (2 bytes) are to be converted then the following word devices will also be read until all required characters are converted. The source address can be specified using:

Decimal Value (K), Hexadecimal Value (H), Timers (T), Counters (C), Data registers (D) or
Group bit devices; Inputs (KnX), Outputs (KnY), Auxiliary relays (KnM), States (KnS).

Parameter d)- The Destination Head Address

The value specified here identifies the first word device that will contain the hexadecimal digits. Each word device can contain 4 digits. The word devices following the Head Address will be used until all the digits have been stored. The destination address can be specified using:

Timers (T), Counters (C), Data registers (D) or
Group bit devices; Outputs (KnY), Auxiliary relays (KnM), States (KnS).

Parameter e)- The Number of Characters

The value specified here can only be a Decimal Value (K) or a Hexadecimal Value (H) and indicates the number of ASCII characters that will be converted and how many hexadecimal digits will be stored. The number of characters can be between 1 and 256 digits.

Parameter f)- The Data Storage Format Reset

The RST M8161 switches the data storage format back to the default of 16 bits.

Note: This parameter is only needed if this instruction uses 8 bit data storage format and there are other instructions in your program that use 16 bit data storage format.

Example Use of the HEX Instruction

Using the program example in figures 1 and 2 the following diagram shows the results for both 16 bit format and 8 bit format.

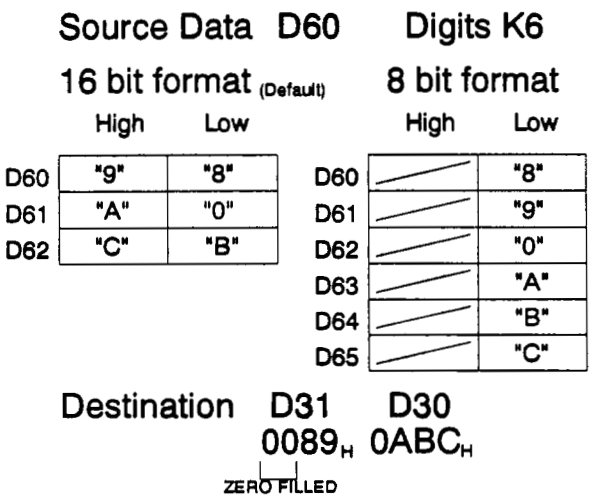


Figure 3: Graphic representation of both 16 and 8 bit data storage format.

ASCII Character Codes

The following table gives the ASCII character code in both hexadecimal and decimal for each of the hexadecimal digits that can be converted.

HEX DIGIT	ASCII		SYMBOL	HEX DIGIT	ASCII		SYMBOL
	HEX	DEC			HEX	DEC	
0	30	48	"0"	8	38	56	"8"
1	31	49	"1"	9	39	57	"9"
2	32	50	"2"	A	41	65	"A"
3	33	51	"3"	B	42	66	"B"
4	34	52	"4"	C	43	67	"C"
5	35	53	"5"	D	44	68	"D"
6	36	54	"6"	E	45	69	"E"
7	37	55	"7"	F	46	70	"F"

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE CHECK CODE INSTRUCTION (CCD, FNC 84)

Introduction

The FX PC family instruction set is divided into basic and applied instructions. The applied instructions provide powerful supplements to the basic set of PC instructions and give control over internal PC operations and external I/O. They are split into 10 groups, each group with a different application, eg. transfer of data within the PC.

Applied instructions 80 to 89 are concerned with the communication of data to devices outside the FX plc and are grouped under the heading of FX Service Interfaces.

This group includes the check code calculation instruciton (CCD, FNC 84). This function is intended as a complement to the serial communications instruction (RS, FNC 80) and calculates a sum check and parity check on the hexadecimal data contained in a group of data registers.

The Check Code Instruction

The CCD conversion instruction has six parameters:

a) the data storage format.

b) the CCD statement itself.

c) the source head address; where the data are stored.

d) the destination address; where the sum check value will be stored.

e) the number of characters; the number of bytes of data to sum.

f) the data storage format reset.

Shown below is the typical appearance of a CCD instruction in both ladder and instruction format.

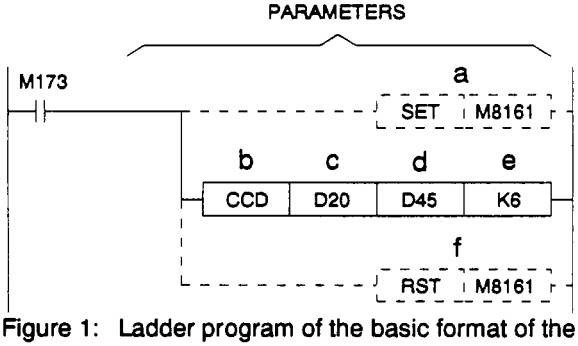


Figure 1: Ladder program of the basic format of the CCD instruction.

PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LD	M	173
1	SET	M	8161
3	CCD		83
		D	20
		D	45
		K	6
10	RST	M	8161

Figure 2: Instruction program of the basic format for the CCD instruction.

The program shown in figures 1 and 2 will execute the sum check and parity check when auxiliary relay M173 turns on. While the CCD instruction is active six (K6) bytes of data from data registers D20 thru D25 (8 bit format) are summed and the sum value and parity check are stored in data registers D45 and D46 respectivly.

The following section explains each parameter in more detail:

Parameter a)- The Data Storage Format

The data storage format can be either 16 bit or 8 bit mode. The 16 bit mode uses both the upper and lower bytes of the source devices and the 8 bit mode only uses the lower 8 bits. This is controlled by the special auxiliary relay M8161; ON for 8 bit mode.

Note: The SET M8161 instruction is only needed if 8 bit mode is being used.

Parameter b)- The CCD Statement

The CCD instruction has two forms:

I) CCD: The standard form of the instruction, taking 7 steps of program. In this form the CCD instruction will continue to calculate the sum and parity checks while active.

II) CCDP: This form is similar to the standard form and also takes 7 steps of program. The P suffix indicates Pulse format. When activated the CCDP instruction will only perform the calculation each time it is turned on.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE CHECK CODE INSTRUCTION (CCD, FNC 84)

Parameter c)- The Source Head Address

The value specified here identifies the first word device that contains the data to be summed. If more than 2 bytes are to be summed then the following word devices will also be read until all required data are used. The source address can be specified using:

Timers (T), Counters (C), Data registers (D) or Group bit devices; Inputs (KnX), Outputs (KnY), Auxiliary relays (KnM), States (KnS).

Parameter d)- The Destination Head Address

The value specified here identifies the first of two word devices that will contain the sum check value and the parity check value respectivly. The sum check is calculated on each byte (8 bits) and the result is in a 2byte word (16 bits). The parity is calculated as even parity on the bit patern for each byte. The destination address can be specified using:

Timers (T), Counters (C), Data registers (D) or Group bit devices; Outputs (KnY), Auxiliary relays (KnM), States (KnS).

Parameter e)- The Number of Characters

The value specified here can be a Decimal Value (K), a Hexadecimal Value (H) or a Data register (D) and indicates the number of bytes (8 bits) that will be summed. The number can be between 1 and 256 bytes.

Parameter f)- The Data Storage Format Reset

The RST M8161 switches the data storage format back to the default of 16 bits.

Note: This parameter is only needed if this instruction uses 8 bit data storage format and there are other instructions in your program that use 16 bit data storage format.

Example Use of the HEX Instruction

Using the program example in figures 1 and 2 the following diagram shows the results for both 16 bit format and 8 bit format.

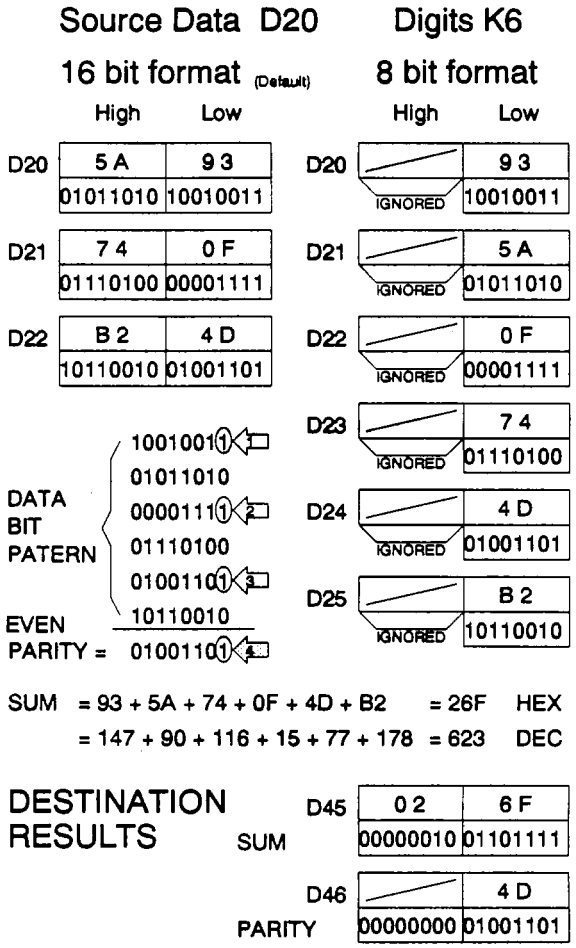


Figure 3: Calculations performed for the sum check and parity check.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE PID INSTRUCTION (PID, FNC 88)

Introduction

As control technology advances, the functions of a programmable controller are becoming more diverse. The FX family of PCs are following this trend by offering a PID control instruction (PID, FNC 88). This instruction is available on the FX2C and FX of V3.11 or later. It allows a user to define a function to control their system or process.

The PID instruction

Before explaining the operation of a PID function, it's a good idea to review the type of system or process it is often used to control. Generally, these are "closed loop" systems. At the simplest level, the system is kept under control by maintaining it at a set value. The set value relates to its most important parameter. The simplest example would be keeping the warmth of a room at a constant temperature. In the real world, usually several parameters need to be kept at a fixed value.

The name "closed loop" shows how the control is achieved. An Output Value is sent to the system by the controller. This is intended to keep the system at the Set Value. The controller also has a sensor to monitor how well the system is keeping to the Set Value. This provides feedback to the controller, which can adjust its Output Value accordingly to make sure the Set Value is maintained. So the feedback closes the loop in the system control diagram, shown in Figure 1.

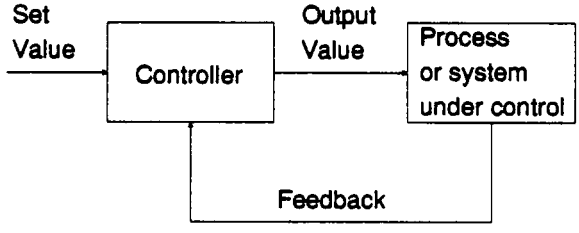


Figure 1: A closed loop system.

The PID instruction is at the heart of the control process, and is defined by the user to give the required control. It takes the Set Value and compares it to the actual value the system is currently at (Current Value). There is usually a difference between the two, called the "error" (ε). This error provides the raw material for the PID instruction to process. The result of the processing is a new Output Value which is sent to the system to bring it back to the Set Value.

The name "PID" represents the three different ways of processing the error to generate the new control value. These are:

- "P": Proportional control. This simply takes the error and multiplies it by a user defined value, K_P, to produce a new control value. This is always K_P times the error, hence the result is proportional.
- "I": Integral control. The control value produced is related to the value of the error over a given time period. Hence the error is integrated over the required period of time.
- "D": Derivative control. The value output by the controller is based on the rate of change of the error. This is the derivative of the error with respect to time.

This list shows that PID is rooted in mathematics, and calculus specifically. However, in general, it's not necessary to solve any complicated equations to set up the PID instruction. It is important to note that these three methods are usually combined to give the required control.

The PID instruction has the following parameters (See Figure 2):

a) The PID statement itself

b) Source Data 1, the required Set Value for the process.

c) Source Data 2, the Current Value of the process. This is the feedback to the controller.

d) Source Data 3. This is the head address of the bank of Data Registers which will be used to store the user's parameters for the PID instruction. These values determine the behavior of the PID control and hence the process.

e) Destination, the calculated Output Value sent to the process.

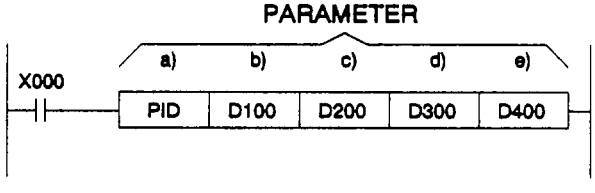


Figure 2: A typical PID instruction in ladder format.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE PID INSTRUCTION (PID, FNC 88)

PROGRAM STEP NUMBER	INSTRUCTION PROGRAM		
0	LD	X	000
1	PID		88
		D	100
		D	200
		D	300
		D	400

Figure 3: PID instruction in instruction format.

Each of these parameters are as follows:

Parameter a)- PID Statement

This is the only form of the PID instruction. It occupies 9 steps of program. In Figure 2, while X000 is ON, then the PID instruction will execute.

Parameter b)-Source Data 1: Set Value

The Set Value is contained in a single Data Register (16 bits).

Parameter c)- Source Data 2: Current Value

The Current Value of the process is also contained in a single Data Register.

Parameter d)- Source Data 3: User Defined Parameters Head Address

This Data Register is the first of 25 which hold the user defined parameters for the PID instruction. These parameters will determine the behavior of the control system built around the PID instruction. These parameters are given at the end of this sheet. The head address must be in the range D0-D975.

Parameter e)- Destination: Output Value

The Output Value calculated by the PID instruction is written to this Data Register.

Explanation of the PID instruction's User Defined Parameters

In total, the PID instruction uses 25 Data Registers to store the parameters set by the user. These are listed in Table 1 at the end of this sheet.

Mathematical equation used by the PID instruction

For reference, the equation used by this instruction is as follows:

$$\text{Output Value} = K_P\left\{\varepsilon + K_D\ T_D\frac{d\varepsilon}{dt} + \frac{1}{T_I}\int \varepsilon\ dt\right\}$$

Where:

- K_P= Proportional gain
- ε= Error
- K_D= Derivative gain
- T_D= Derivative time constant
- T_I = Integral time constant

Notes on the use of the PID instruction

- There are no restrictions on the number of PID instructions being used in a program. However, each PID instruction must use separate groups of Data Registers to avoid data conflicts inside the PC.
- Although the PID instruction can be used in interrupts, subroutines, step ladder or with jumps, care must be taken. In these situations, it is recommended to use the PID instruction in the way shown in Figure 4 .The choice of Data Registers and contacts is down to the user. Note however, that the Move instruction must always move K0 into DYYY+7. So if DYYY=D300, then DYYY+7=D307.

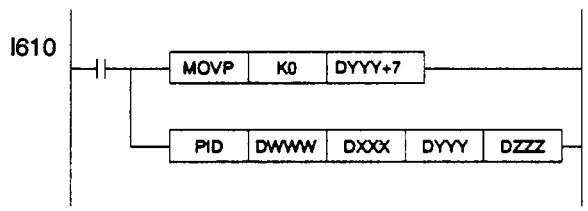


Figure 4: Ensuring correct PID operation. Note that an interrupt routine has only been chosen as one possible example.

- The Sampling Time (Ts) should be kept longer than the program scan time. If it is shorter, then an error will result. At this point, the Sampling Time will be made

AN INTRODUCTION TO THE PID INSTRUCTION (PID, FNC 88)

equal to the scan time and operation will continue.
In the case of timer interrupt operation, (I6XX–I8XX), the Sampling Time should be no shorter than the interrupt cycle time.

- The Sampling Time, T_S, is subject to an variation caused by the program scan. The maximum amount for this variation is T_S–(Program scan time+1ms)/+(Program scan time). This can be minimized by using the PID instruction from within a timed interrupt routine.
- The PID instruction includes alarms which allow abnormal process conditions to be signalled when they occur. These alarms can be enabled and disabled by the user. The points at which they are triggered are also specified by the user defined parameters.
- The PID instruction has certain error codes associated with it to help diagnose problems with its operation. These are stored in Data Register D8067. When an error occurs, it is flagged by M8067. Please see Table 2 at the end of this sheet for the list of error codes and their meanings.

Example use of the PID instruction

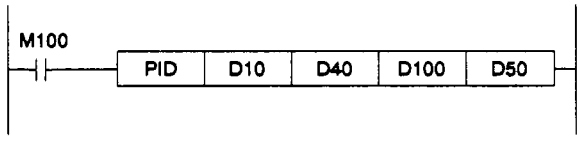


Figure 5: An example use of the PID instruction.

An example use of the PID instruction is shown in Figure 5. This shows a PID instruction which becomes active when M100 turns on. The Set Value is stored in D10, the Current Value is read from D40 and the Output Value is written to D50. This leaves D100 as the head address of the user defined parameters , i.e. D100–124 are used.

If the system being controlled was completely analog, then two special function blocks need to be added to the base unit to handle the signals to and from the PC. Figure 6 shows this.

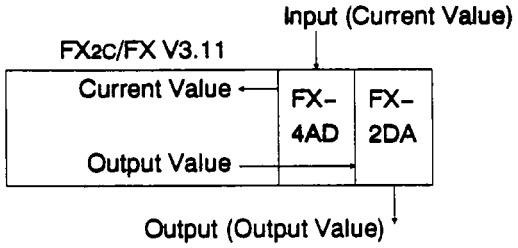


Figure 6: An example analog based PID system.

The first is an FX–4AD, which takes the analog Current Value from the process and converts it into an equivalent digital value. The base unit would use a FROM instruction to place this value into D40. An example of an analog Current Value could be a varying voltage signal.

Secondly, the Output Value must be converted from the number stored in D50 into an analog signal to be sent to the process. This can be handled by the FX–2DA special function block. An example of analog control would be a motorized valve whose position depends on a voltage level.

This example does not give any values for the user defined parameters as these will depend on the specific nature of the system under control.

Alternative method of providing a control output

The PWM instruction (PWM, FNC 58) can be used to produce a square wave output directly from a transistor base unit . This can have its duty cycle varied by the Output Value. Hence the on/off ratio of the square wave can be varied directly by the PID instruction. This avoids the need for a digital to analog output stage in some applications.

Setting the PID instruction user defined parameters

These values could be written directly to the Data Registers concerned (D100–D124 in this case). If the parameters are to be retained after power down, then remember to use the retentive Data Registers. Another approach would be to have the parameter values stored in the File Registers and use the BMOV instruction (BMOV, FNC 15) to write them to the required Data Registers. This would allow several sets of parameters to be stored and changed if necessary, under program control.

AN INTRODUCTION TO THE PID INSTRUCTION (PID, FNC 88)

Parameter Number		Name	Description	Setting Range
Source Data 3	+0	Sampling Time (T_S)	Interval between samples of the process' Current Value	1–32767 msec
	+1	Action direction/ Alarm control	Bit 0: 0: Forward direction. 1: Reverse direction.	N/A
			Bit 1: 0/1: Current Value change alarm OFF/ON.	N/A
			Bit 2: 0/1: Output Value change alarm OFF/ON.	N/A
			Bit 3–15: Reserved.	N/A
	+2	Input filter (α)	Setting value for input filter.	0–99%
	+3	Proportional gain (K_P)	Multiplying factor for proportional ("P") control.	1–32767%
	+4	Integral time constant (T_I)	Integral ("I") calculation multiplied by the inverse of this value. Selecting 0 for this value disables the integral control.	0–32767x100 msec
	+5	Derivative gain (K_D)	Multiplying factor for derivative ("D") control.	0–100%
	+6	Derivative time constant (T_D)	Derivative calculation multiplied by this value. Selecting 0 for this value disables the derivative control.	0–32767x10msec
	+7–+19	Reserved	N/A	N/A
	+20	Current Value change alarm preset value (increasing)	Alarm enabled by bit 1 of S3+1(Action direction) triggered when Current Value increases more than this value.	0–32767
	+21	Current Value change alarm preset value (decreasing)	Alarm enabled by bit 1 of S3+1(Action direction) triggered when Current Value decreases more than this value.	0–32767
	+22	Output Value change alarm preset value (increasing)	Alarm enabled by bit 2 of S3+1(Action direction) triggered when Output Value increases more than this value.	0–32767
	+23	Output Value change alarm preset value (decreasing)	Alarm enabled by bit 2 of S3+1(Action direction) triggered when Output Value decreases more than this value.	0–32767
	+24	Alarm output	Bit 0: Current Value change alarm(increasing)	N/A
			Bit 1: Current Value change alarm (decreasing)	N/A
			Bit 2: Output Value change alarm(increasing)	N/A
			Bit 3: Output Value change alarm (decreasing)	N/A

Table 1: List of user defined parameters for the PID instruction.

AN INTRODUCTION TO THE PID INSTRUCTION (PID, FNC 88)

Error code (Stored in D8067)	Meaning of error	Effect on execution of PID instruction
K6705	Devices specified for the PID instruction are not Data Registers.	Execution ceases
K6706	Data Registers have been specified outside the allowed range.	
K6730	Sampling time (T_S) is outside allowed range (T_S<0).	
K6732	Input filter value (α) is outside allowed range (α<0 or α>=100).	
K6733	Proportional gain (K_P) is outside allowed range (K_P<0).	
K6734	Integral time constant (T_I) is outside allowed range (T_I<0).	
K6735	Derivative gain (K_D) is outside allowed range (K_D<0 or K_D>=101).	
K6736	Derivative time constant (T_D) is outside allowed range (T_D<0).	
K6740	Sampling time (T_S) <= program scan time.	Sampling time is set equal to the scan time and execution continues.
K6742	Current Value change rate outside allowed limits (Δ Current Value <-32768 or >32767).	Data affected changes to maximum/minimum allowed value and operation continues.
K6743	Error outside allowed limits (ε<-32768 or ε>32767).	
K6744	Integral result outside allowed limits (Value outside range -32768 to 32767).	
K6745	Derivative gain over or differential value outside allowed range.	
K6746	Derivative result outside allowed limits (Value outside range -32768 to 32767).	
K6747	PID total result outside allowed limits (Value outside range -32768 to 32767).	

Table 2: List of PID instruction error codes and their meanings.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE SHIFT MOVE INSTRUCTION (SMOV, FNC 13) USED WITH M8168

Introduction

The FX PC family has a powerful set of applied instructions as well as the commonly used basic instructions. The applied instructions are versatile additions to the basic set and provide control over internal PC operations and external I/O. There are 10 groups of applied instructions, each with a different area of application, eg. arithmetic and logical operations.

Applied instructions 10 to 19 deal with the movement of data within the PC and are grouped under the category of Transfers.

This group includes the Shift Move instruction (SMOV, FNC 13). Shift Move provides a facility for handling blocks of numeric data. Usually these data are decimal numbers, but if SMOV is used with Auxiliary Relay M8168 on, then hexadecimal data can be manipulated. The instruction works in the following way:

The source device/devices (for this instruction) stores a hexadecimal number up to FFFF. Each digit of this number occupies a numbered location from 1 at the far right to 4 at the far left. A block of digits is defined by specifying the location of the first digit and the length of the block.

This block can then be shifted by defining a different position for the first digit, and a copy of the block is moved into the destination device(s) at the new position. Any digits already in the location to which the block is moved in the destination device(s) are overwritten. The other digits remain unchanged.

The Shift Move Instruction

The hexadecimal Shift Move instruction has two parts:

Part 1: Turn on M8168

a) An OUT instruction to turn on M8168.

Part 2: The SMOV statement (six parameters).

b) The SMOV statement itself

c) the specification for the Source Data, ie. where the block of digits to be shifted and moved is copied from.

d) the position number of the first digit of the block.

e) the number of digits in the block.

f) the destination device(s) for the block of digits.

g) the number of the new position for the shifted block in the destination device(s).

Shown in Figure 1 is the typical appearance of a hexadecimal Shift Move instruction in a ladder program, while Figure 2 shows the same piece of ladder logic written in instruction format.

The following section explains each parameter in greater detail:

Parameter a)- The SMOV statement

The SMOV statement can take four different forms:

I) SMOV: The standard form of the instruction, taking 11 steps of program. In the example shown in Figure 1, while X000 is turned on, the SMOV instruction will operate.

II) SMOVP: The "P" suffix shows that SMOV is being used in its PULSE format. The instruction will only operate when X000 provides a rising edge signal as it turns on. SMOVP also takes 11 program steps.

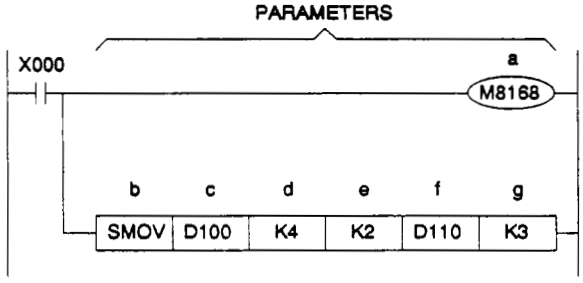


Figure 1: The Shift Move instruction in ladder format.

PROGRAM STEP NUMBER	INSTRUCTION PROGRAM		
0	LD	X	000
1	OUT	M	8168
3	SMOV		13
		D	100
		K	4
		K	2
		D	110

Figure 2: Shift Move instruction in instruction format.

FX2

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE SHIFT MOVE INSTRUCTION (SMOV, FNC 13) USED WITH M8168

Parameter b)- Source Device

This identifies what form the block of digits to be copied and moved takes and where it is located in the PC.The data can be in any of the following forms:

Grouped Bit Devices: Inputs (KnX), Outputs (KnY), Auxiliary Relays (KnM), States (KnS), or Timers (T), Counters (C), Data Registers (D) and Index Registers (V&Z).

Parameter c)- Position number of the first digit of the block.

This identifies the start of the block of digits in the Source Device and must be a value of 1 through 4.

Parameter d)- The number of digits in the block.

This defines the length of the block and is a value between 1 and 4.

Parameter e)- Destination Device.

The device(s) to which the block of digits will be copied and moved and where it/they are located. These may be:

Grouped Bit Devices: Outputs (KnY), Auxiliary Relays (KnM), States (KnS), or Timers (T), Counters (C), Data Registers (D) and Index Registers (V&Z).

Parameter f)- New Position

The number of the new position for the shifted block in the destination device(s). This is also a value between 1 and 4.

Example Use of the Shift Move Instruction with M8168.

The example in Figure 1 uses D100 as its source device, storing "FFE2", and D110 as the destination device, storing "2CD9".The Shift Move instruction would have this effect when X000 is turned on (see Figure 3);

- Copy the digits "FF" (block starting at position 4, 2 digits long) from the value "FFE2" in D100.
- Move the copied block to D110, at position 3. This overwrites the digits "CD" at position 3 with "FF"

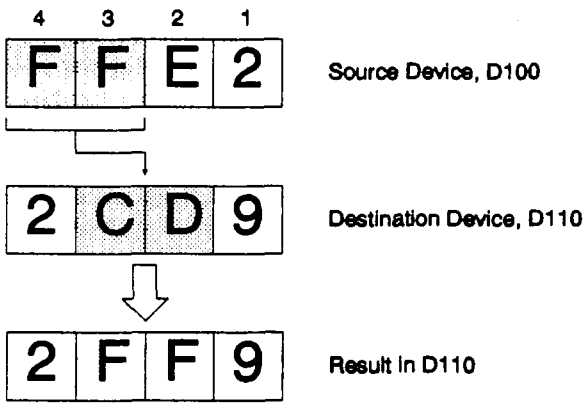


Figure 3: Example operation of the Shift Move instruction.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE EXCHANGE INSTRUCTION USED AS A BYTE SWAP FUNCTION

Introduction

The FX PC family has a powerful set of applied instructions as well as the commonly used basic instructions. The applied instructions are versatile additions to the basic set and provide control over internal PC operations and external I/O. There are 10 groups of applied instructions, each with a different area of application, eg. arithmetic and logical operations.

Applied instructions 10 to 19 deal with the movement of data within the PC and are grouped under the category of Transfers.

This group includes the Exchange instruction (XCH, FNC 17). Usually XCH is used to exchange the contents of two devices, but if used with Auxiliary Relay M8160 on, then XCH swap the data of the upper and lower bytes of the given device.

The Byte Swap Exchange Instruction

The Byte Swap Exchange instruction has five parameters:

a) setting the XCH byte swap flag.

b) the XCH statement itself.

c) the swap device; the device which is to have the data swapped.

d) repeat of the swap device; needed for this form of the XCH instruction.

e) resetting the XCH byte swap flag; to allow other XCH instructions to operate as normal.

The following figures show the typical appearance of this form of the XCH instruction in both ladder and instruction programing formats.

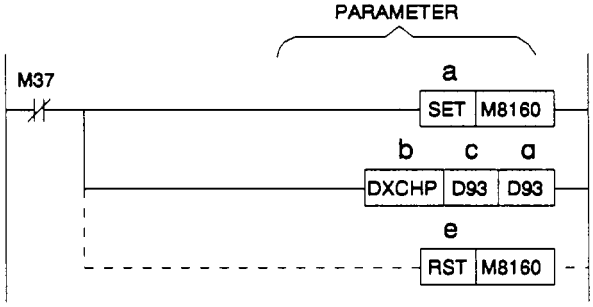


Figure 1: Ladder format of the byte swap form of the XCH instruction (using DOUBLE word and pulse formats).

PROGRAM STEPS	INSTRUCTION PROGRAM		
0	LDI	M	37
1	SET	M	8160
3	DXCHP		17
		D	93
		D	93
12	RST	M	8160

Figure 2: Instruction format of the byte swap form of the XCH instruction (using DOUBLE word and pulse formats).

When the instruction is executed by having M37 turn off the contents of data registers D93 and D94 are changed. The upper byte (8 bits) and the lower byte (8 bits) of each register are swapped. Because this instruction will perform this function each time the program executes it is advisable, for most applications, to use the pulse form of this instruction as in the example.

The following section explains each parameter in greater detail:

Parameter a)- Setting the byte swap flag

The special auxiliary relay M8160 is used to change the XCH instruction into a byte swap function. For this form of the XCH instruction it is necessary to have the XCH Byte Swap Flag on.

Parameter b)- The XCH statement

The SMOV statement can take four different forms:

I) XCH: The standard form of the instruction, taking 5 program steps. When the instruction is driven each byte of the data will be swapped every time the instruction is processed.

Note: The data will be swapped back during the following scan if the instruction is still being driven.

FX2

THE EXCHANGE INSTRUCTION USED AS A BYTE SWAP FUNCTION

II) XCHP: The "P" suffix shows that XCH is being used in its PULSE format. The instruction will execute the swap once when the driving signal is turned on and must be turned off and on again for the instruction to operate a second time. XCHP also takes 5 program steps.

Note: This is the recommended format for most applications, because data is prevented from being swapped back on the next scan.

III) DXCH: This is similar to XCH, but the 'D' prefix indicates the instruction will swap the data of a DOUBLE word, i.e. the bytes of each device of the double word will be swapped. This form of the instruction takes 9 program steps.

IV) DXCHP: this is a combination of the PULSE format and the DOUBLE word format; also taking 9 program steps. Each byte of the DOUBLE word will be swapped once when the instruction is executed. The data will not be swapped again until the instruction is switched off and on again.

Parameter c)- The Swap Device

This identifies the device that will have its upper and lower devices swapped. What ever the device type (or size) its value is considered to be a complete word (leading zeros will be added if necessary). Each byte (8 bits) of this word will be swapped and the data converted back to its original form.

Note: If DOUBLE word is used then each word is treated separately and will have their upper and lower bytes swapped independently.

The following devices are allowed:

Grouped Bit Devices: Outputs (KnY), Auxiliary Relays (KnM), States (KnS), or Timers (T), Counters (C), Data Registers (D) and Index Registers (V&Z; V only if using DOUBLE word).

Parameter d)- Repeat of Swap Device

This device MUST be a copy of the swap device above for this form of the XCH instruction. If a different device is specified then an execution error will occur (Error flag M8067).

Parameter e)- Resetting the XCH Byte Swap Flag

The XCH Byte Swap Flag must be reset if the XCH instruction is used in its normal form in another part of the program. If only the byte swap format of the XCH instruction is being used then this RST instruction is not needed.

Example Use of The Byte Swap Function

Following the example program in figures 1 and 2 the diagram below shows the results when the program is executed.

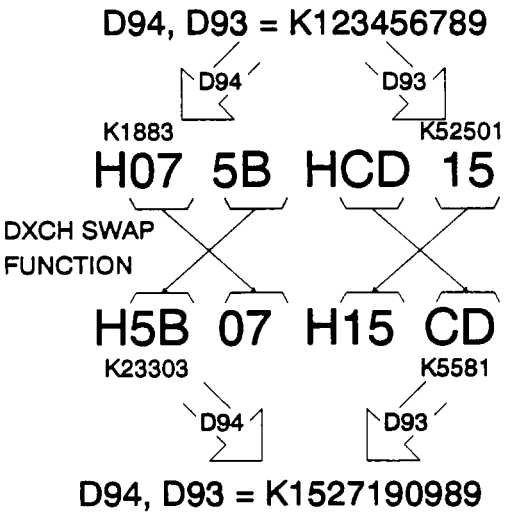


Figure 3: The DOUBLE data of D93 and D94 have each byte swapped. Each byte is shown here as two hexadecimal digits.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE DECIMAL TO BCD INSTRUCTION (BCD, FNC 18) USED WITH M8023 FLOAT OPERATION FLAG

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic set of PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. arithmetic and logical operations.

Applied instructions 10 to 19 are concerned with the movement of data within the PC and are grouped under the heading of Transfers.

This group includes the Decimal to BCD conversion instruction (BCD, FNC 18). When used in conjunction with the float operation flag M8023 the BCD instruction performs a conversion of data from floating point format to scientific format. Without any form of conversion floating point numbers are very difficult to interpret. This conversion is useful to view the contents of floating point numbers in a format that we can understand.

The BCD Instruction with M8023

This form of the BCD instruction has three parts:

Part 1: Set M8023

a) setting the floating point operation flag.

Part 2: The BCD instruction (three parameters)

b) the BCD statement itself.

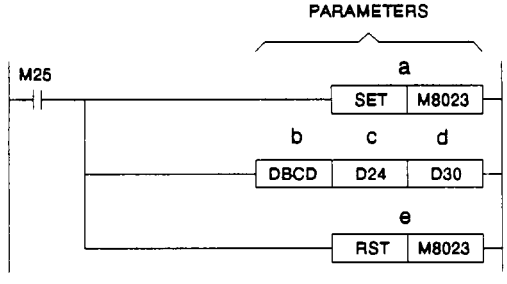
c) the specification of the Source Data i.e. from where the float value is read.

d) the specification of the destination data i.e. to where the scientific value is written.

Part 3: Reset M8023

e) resetting the floating point operation flag.

Shown below is this format of the BCD instruction as a ladder diagram showing each of these features. The following table shows the same program written in instruction format.



PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LD	M	25
1	SET	M	8023
3	DBCD		18
		D	24
		D	30
12	RST	M	8023

Looking at the ladder diagram we can see that when the auxiliary relay M25 is turned ON, first the float operation flag is set to ON, then the BCD instruction executes. This results in the contents of data register D25 and D24 being read as a floating point number and converted to scientific format. This value is then written to the data register pair D31 and D30.

Parameter a)- Setting the float flag

The Float Operation flag M8023 must be SET. This flag changes the function of the BCD instruction and math functions to operate on floating point numbers.

Parameter b)- The BCD statement

The BCD statement can be written in four ways:
Only DOUBLE WORD is valid for this format of the BCD instruction.

I) BCD: The basic format of the instruction, taking 5 steps of program memory. While the instruction is ON it continues to execute.
NOT VALID IN THIS FORMAT.

II) BCDP: The "P" suffix says the instruction is being used in PULSE mode. The instruction will execute once when it is first turned ON; i.e. rising edge execution.
NOT VALID IN THIS FORMAT.

III) DBCD: This is the same as the BCD format except the "D" prefix indicates DOUBLE words are converted, i.e. two consecutive data registers (32 bits) will be converted. DBCD takes 9 steps of program memory.

IV) DBCDP: This is a combination of the DOUBLE and the PULSE format of the BCD instruction. The instruction converts up to 32 bits of data only once after being switched on.

THE DECIMAL TO BCD INSTRUCTION (BCD, FNC 18) USED WITH M8023 FLOAT OPERATION FLAG

Parameter c)- Source Data

This identifies the location of the data to be converted. The data must be stored as a floating point number in a Data Registers (D) pair.

Parameter d)- Destination Data

This identifies the location where the converted data is to be written. The data will be written in Scientific format. The destination device must be a Data Register (D) pair.

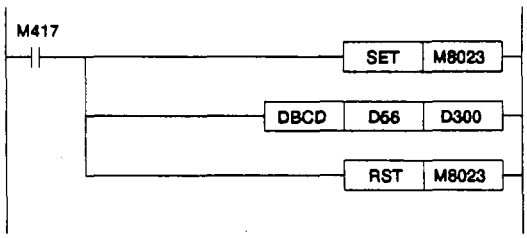
Scientific format separates the floating point number into mantissa and exponent. The mantissa is stored in the first data register and is truncated to four significant figures. The exponent is stored in the second data register and is adjusted to keep the mantissa a whole number.

Parameter e)- Resetting the float flag

The Float Operation flag must be RESET to allow further functions to operate as normal. The flag can be left on if further operations requiring floating point calculations are needed. However, it is recommended that once all floating point functions are completed this flag should be switched off using RST.

Example Use of the BCD Instruction

The diagram below shows the BCD instruction being used to convert the contents of data register pair D57 and D56 to scientific format in D300 and D301.



	Floating Point Value of D57,D56	Scientific Notation	
		D300	D301
Ex. 1	4.27594×10^{10}	4275	7
Ex. 2	-7.0×10^{-9}	-7000	-12
Speed of Light, C (m/s)	2.997924×10^{8}	2997	5
PI, Π	3.141592×10^{0}	3141	-3
Avogadro's Constant	6.022045×10^{23}	6022	20
Plank's Constant	6.626176×10^{-34}	6626	-37

Notice that the Exponent (D301) is 3 less than expected to keep the Mantissa (D300) a four digit whole number in the range ± 1,000 to 9,999 or 0.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

THE BCD TO DECIMAL INSTRUCTION (BIN, FNC 19) USED WITH M8023 FLOAT OPERATION FLAG

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic set of PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. arithmetic and logical operations.

Applied instructions 10 to 19 are concerned with the movement of data within the PC and are grouped under the heading of Transfers.

This group includes the Decimal to BCD conversion instruction (BIN, FNC 18). When used in conjunction with the float operation flag M8023 the BIN instruction performs a conversion of data from scientific format to floating point format . Without any form of conversion floating point numbers are very difficult to interpret. This conversion is useful to set the contents of floating point numbers from a format that we understand.

The BIN Instruction with M8023

This form of the BIN instruction has three parts:

Part 1: Set M8023

a) setting the floating point operation flag.

Part 2: The BIN instruction (three parameters)

b) the BIN statement itself.

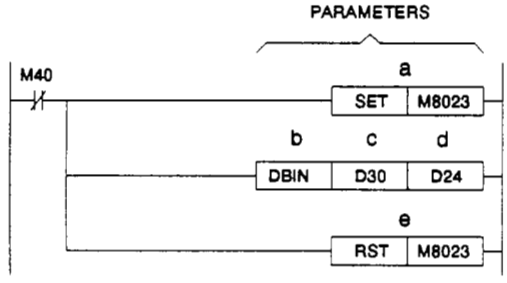
c) the specification of the Source Data i.e. from where the scientific value is read.

d) the specification of the destination data i.e. to where the float value is written.

Part 3: Reset M8023

e) resetting the floating point operation flag.

Shown below is this format of the BIN instruction as a ladder diagram showing each of these features. The following table shows the same program written in instruction format.



PROGRAM STEP No.	INSTRUCTION PROGRAM		
0	LDI	M	40
1	SET	M	8023
3	DBIN		19
		D	30
		D	24
12	RST	M	8023

Looking at the ladder diagram we can see that when the auxiliary relay M40 is turned ON, first the float operation flag is set to ON, then the BIN instruction executes. This results in the contents of data register D30 and D31 being read as a scientific format number and converted to floating point format. This value is then written to the data register pair D25 and D24.

Parameter a)– Setting the float flag

The Float Operation flag M8023 must be SET. This flag changes the function of the BIN instruction and math functions to operate on floating point numbers.

Parameter b)– The BIN statement

The BIN statement can only be DOUBLE word when used with the float operations flag:

I) DBIN: The basic format for this use of the instruction, taking 9 steps of program memory. While the instruction is ON it continues to execute. The "D" prefix indicates DOUBLE words are converted, i.e. two consecutive data registers (32 bits) will be converted.

II) DBINP: This is the PULSE format of the DBIN instruction. The instruction will execute once when it is first turned ON (i.e. rising edge execution) and converts 32 bits of data.

Parameter c)– Source Data

This identifies the location of the data to be converted. The data must be held in data registers and will be interpreted as scientific notation format.

Scientific notation format stores a number as mantissa and exponent. The mantissa is stored in the first data register and must have four significant figures.

Valid range for the mantissa: ± 1,000 to 9,999 or 0

FX2

THE BCD TO DECIMAL INSTRUCTION (BIN, FNC 19) USED WITH M8023 FLOAT OPERATION FLAG

The exponent is stored in the second data register and is adjusted to keep the mantissa a whole number.

Valid range for the exponent: –41 to +35

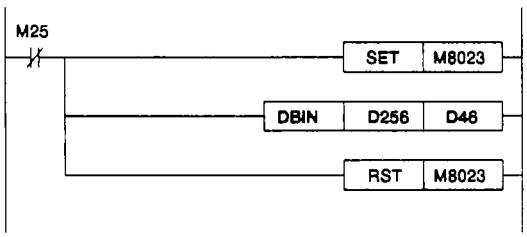
Parameter d)– Destination Data

This identifies the location where the converted data is to be written. The data will be written in floating point and the device must be a Data Register (D) pair.

Parameter e)– Resetting the float flag

The Float Operation flag must be RESET to allow further functions to operate as normal. The flag can be left on if further operations requiring floating point calculations are needed. However, it is recommended that once all floating point functions are completed this flag should be switched off using RST.

Example Use of the BIN Instruction



The diagram below shows the BIN instruction being used to convert scientific notation from D256 and D257 to floating point format in data register pair D47 and D46.

	Scientific Notation		Floating Point Value of D57,D56
	D46	D47	
Ex. 1	1593	15	1.593×10^{18}
Ex. 2	–6003	–10	-6.003×10^{-7}
Speed of Light, C (m/s)	2997	5	2.997×10^{8}
PI, Π	3141	–3	3.141×10^{0}
Electron Charge, Coulombs	1602	–22	1.602×10^{-19}
Gravity, g (m/s^2)	9807	–3	9.807×10^{0}

Notice that the Exponent (D47) is 3 less than expected to keep the Mantissa (D46) a four digit whole number in the range ± 1,000 to 9,999 or 0.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE SEARCH INSTRUCTION (SER, FNC 61)

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. High speed operations.

Applied instructions 60 to 69 are a handy set of instructions that simplify complex operations; they are grouped together under the name 'Useful instructions'.

This group includes the Search instruction (SER, FNC 61). This function searches a list of data devices for a specified value. The results given show: the number of items found, the position of the first and last item, and the largest and smallest values in the list.

The Search Instruction

The Search instruction has five parameters:

a) the SER statement itself

b) the head address of the list; the first device in which the data is stored.

c) the search value; the value looked for during the search.

d) the head address of the results list; where the results will be stored.

e) the list length; the number of items to search.

Show below is the typical appearance of a SER instruction in both instruction and ladder format.

PROGRAM STEP NUMBER	INSTRUCTION PROGRAM		
0	LD	X	17
1	SER		61
		D	130
		D	24
		D	35
		K	10

Figure 1: Instruction program of the basic format for the Search instruction.

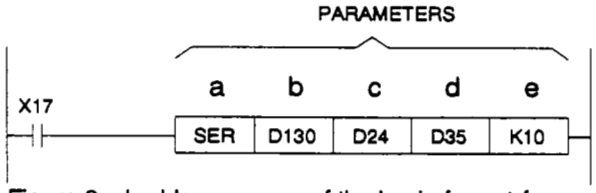


Figure 2: Ladder program of the basic format for the Search instruction.

The program would test each value in the list from D130 to D139 (list length K10) for a match with the contents of D24 when ever X17 is on. Any match found will be counted and the first and last occurrences recorded. In addition to this the maximum and minimum values in the list will also be found. The result would then be stored in the five consecutive data registers from D35 to D39. The following section explains each parameter in greater detail:

Parameter a)- The SER Instruction

The SER instruction can take four different forms:

I) SER: The standard form of the instruction, taking 9 steps of program. In this format the search is continually executed while the instruction is on.

II) SERP: The "P" suffix shows that SER is being used in its PULSE format. The instruction will only operate each time a rising edge signal turns the instruction on. SERP also takes 9 program steps.

III) DSER: This is similar to SER, but the "D" prefix indicates the instruction is using DOUBLE word data, ie. 32 bits, equivalent to two data registers each. All data for the instruction is in DOUBLE word format. The search list, the search value and the length will be in DOUBLE word format. The results will also be in DOUBLE word format; therefore 10 data devices will be used for the results. This format of the instruction takes 17 program steps.

IV) DSERP: This combines the DOUBLE word format with the PULSE operation to give a SER instruction which will test 32 bit data values when a rising edge switches the instruction on. DSERP takes 17 program steps.

FX2

AN INTRODUCTION TO THE SEARCH INSTRUCTION (SER, FNC 61)

Parameter b)- The Head Address of the List

The value specified here identifies the first item of the consecutive list of devices to be searched, i.e. what device type the list uses and where it is located in the PC. The data can be in any of the following forms: Grouped Bit Devices:

Inputs (KnX), Outputs (KnY), Auxiliary Relays (KnM), States (KnS) or Timers (T), Counters (C) and Data Registers (D).

Parameter c)- The Search Value

This identifies the type and location of the data to look for during the search. The following values are allowed:

Decimal Values (K), Hexadecimal Values (H), Grouped Bit Devices: Inputs (KnX), Outputs (KnY), Auxiliary Relays (KnM), States (KnS) or Timers (T), Counters (C), Data Registers (D) and Index Registers (V&Z).

Parameter d)- The Head Address of the Results List

This identifies where the result will be stored. A consecutive list of five devices (ten for DOUBLE word) will be used for the results containing the following:

How many of the search value are in the search list (zero if not found).

The position in the list of the first search value found (zero if not found).

The position in the list of the last search value found (zero if not found).

The position in the list of the smallest value found (last occurence if more than one found).

The position in the list of the largest value found (last occurence if more than one found).

The possible devices can be one of the following:

Grouped Bit Devices: Outputs (KnY), Auxiliary Relays (KnM), States (KnS) or Timers (T), Counters (C) and Data Registers (D).

Parameter e)- The List Length

The length of the list to search for the test value. The list length can be a maximum of 256 for 16 bit data devices and 128 for 32 bit (DOUBLE) data devices. The length can be specified using:

Decimal Values (K), Hexadecimal Values (H) or Data Registers (D).

Example Use of the Search Instruction

Using the program from the typical format (Figures 1 and 2), if the search value D24 has the value K100 and the search list is defined as follows

Length K10

Position	Search List	Search Value	Maxi-mum	Mini-mum
0	D130 = K100	Match		
1	D131 = K111			
2	D132 = K100	Match		
3	D133 = K98			
4	D134 = K123			
5	D135 = K66			MIN
6	D136 = K100	Match		
7	D137 = K95			
8	D138 = K210		MAX	

The results table will have the following values:

Results List	Contents	Meaning
D35	3	Number of Matches
D36	0	First match position
D37	6	Last match position
D38	5	Smallest value in list
D39	8	Largest value in list

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO THE SORT INSTRUCTION (SORT,FNC 69)

Introduction

The FX PC family instruction set is divided into two groups: basic and applied instructions. The applied instructions are powerful supplements to the basic PC instructions and are used to provide control over internal PC operations and external I/O. They are split into 10 groups, each group with a different emphasis, eg. High speed operations.

Applied instructions 60 to 69 are a handy set of instructions that simplify complex operations; they are grouped together under the name 'Useful instructions'.

This group includes the Sort instruction (SORT, FNC 61). This function sorts a table of data registers so that contents of one column are in numerical order. The other columns in the table follow the sorted data to maintain record continuity.

The Sort Instruction

The Sort instruction has six parameters:

a) the SORT statement itself

b) the head address of the table; the first device in which the data is stored.

c) the number of records; the number of devices in each column.

d) the number of fields; the number of columns in the table.

e) the head address of the sorted table; the first device in which the sorted data is placed.

f) the sort field; the column on which the sort is performed.

Show below is the typical appearance of a SORT instruction in both ladder and instruction format.

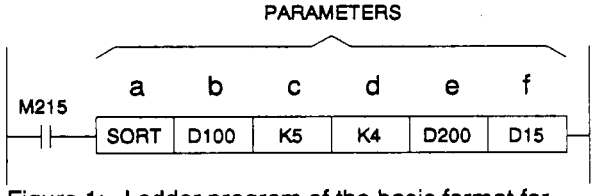


Figure 1: Ladder program of the basic format for the Sort instruction.

PROGRAM STEP NUMBER	INSTRUCTION PROGRAM		
0	LD	M	215
1	SORT		69
		D	100
		K	5
		K	4
		D	200
		D	15

Figure 2: Instruction program of the basic format for the Sort instruction.

The program defines the primary table from D100 as having 5 record and 4 fields. An example of this can be seen in figure 3. When M215 is switched on the table is sorted using the data in the field defined by D15. The new sorted table starts from D200; examples of which can be found in figures 4 and 5. The following section explains each parameter in greater detail:

Parameter a)- The SORT Instruction

The SORT instruction has only one form:

I) SORT: The standard form of the instruction, taking 11 steps of program. The SORT instruction can only be used once in a program.

Parameter b)- The Head Address of the Table

The value specified here identifies the first item of the table to be sorted, i.e. where it is located in the PC. The data can only be stored in Data Registers (D) and there must be enough space after this address to hold the complete table.

Parameter c)- The Number of Records (Nr)

This is the number of consecutive data registers that form each column of the table. There can be between 1 and a maximum of 32 records in a table. Only Decimal Values (K) or Hexadecimal Value (H) can be used to specify the number of records.

Parameter d)- The Number of Fields (Nf)

This is the number of columns that make up the table. Each one represents one type of data. There can be between 1 and a maximum of 6 fields in a table. Only Decimal Values (K) or Hexadecimal Value (H) can be used to specify the number of fields.

AN INTRODUCTION TO THE SORT INSTRUCTION (SORT,FNC 69)

Parameter e)- The Head Address of the Sorted Table

This identifies where the new table data will be stored. After the data is sorted it is placed in a new table that starts at this address. Only Data Registers (D) can be used to store the table data and there must be enough space after this address to hold the complete table.

Note: This address can be the same as the primary table address, however the data must not be changed during the sort. The operation complete flag M8029 is used to indicate the end of the sort operation.

Warning! If the primary table and the sorted table overlap (not the same address) then the data in the tables will become scrambled.

Parameter f)- The Sort Field

This indicates the field or column that is to be sorted. The value is the column number and must be valid for the current table specification. The sort field can be specified using:

Decimal Values (K), Hexadecimal Values (H) or Data Registers (D).

Example Use of the Sort Instruction

The table below shows an example of the primary table with the data already allocated to each record. The table is defined as in the example programs found in figures 1 and 2.

Nf →

Field Number	1	2	3	4
Record Number	ID Number	Height	Weight	Age
1	D100 1	D105 150	D110 45	D115 20
2	D101 2	D106 180	D111 50	D116 40
3	D102 3	D107 160	D112 70	D117 30
4	D103 4	D108 100	D113 20	D118 8
5	D104 5	D109 150	D114 50	D119 45

Nr ↓

Figure 3: Example of the primary table starting at address D100.

Using the program shown in figures 1 and 2 with the table data as defined in figure 3. If the sort field D15 has the value K2 then the results table will have the following values:

Nf →

Field Number	1	2	3	4
Record Number	ID Number	Height	Weight	Age
1	D200 4	D205 100	D210 20	D215 8
2	D201 1	D206 150	D211 45	D216 20
3	D202 5	D207 150	D212 50	D217 45
4	D203 3	D208 160	D213 70	D218 30
5	D204 2	D209 180	D214 50	D219 40

Nr ↓

Figure 4: Example of results table sorted on field 2, starting at address D200.

If the sort field D15 has the value K3 the results table will have the following values:

Nf →

Field Number	1	2	3	4
Record Number	ID Number	Height	Weight	Age
1	D200 4	D205 100	D210 20	D215 8
2	D201 1	D206 150	D211 45	D216 20
3	D202 2	D207 180	D212 50	D217 40
4	D203 5	D208 150	D213 50	D218 45
5	D204 3	D209 160	D214 70	D219 30

Nr ↓

Figure 5: Example of results table sorted on field 3, starting at address D200.

AN INTRODUCTION TO NUMBER FORMATS FOR THE FX

Introduction

The FX PC uses data devices to hold number information. The FX PC can store numbers in any of three formats:

a) Decimal (whole numbers)

b) Scientific notation

c) Floating Point

Decimal Format

The most common format used by PCs is Decimal numbers. This format stores the number as an integer i.e. there are no decimal places.

The decimal format for numbers allows the programmer to use either single word devices (16 bits) or double word devices (32 bits).

A single word device (e.g. D142) can store a number from –32768 up to +32767 in 1 unit steps.

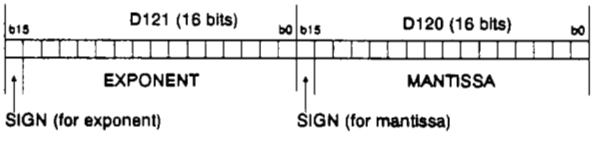
A double word device (e.g. D173 and D172) combines two data registers to give a possible range from –2147483648 up to +2147483647 in 1 unit steps.

This is sometimes not accurate enough and fractions are occasionally needed; particularly if extensive mathematical calculations involving divisions are used.

The Scientific Notation Format for Data

This is very closely related to the standard scientific notation. The number is stored in a format similar to scientific notation with a mantissa and an exponent.

The following diagram shows an example of the scientific data format.



The value of the number is equivalent to:

$$D120 \times 10^{D121}$$

For example, the speed of light C, is

299,792,458 m/s

or about

$$2997 \times 10^{5}\ m/s$$

Here the 2,998 would be the mantissa and the 5 would be the exponent.

The mantissa stores the value of the number but it is truncated to 4 significant figures with NO ROUNDING. It is also kept within the following limits:

± 1,000 to 9,999 or 0

The exponent is used to determine the number of decimal places the decimal point should be moved to give the actual number. For Scientific notation format the limits for the exponent are:

–41 to +35

Unfortunately, the FX can not use the scientific format for calculations. This means the scientific format must be converted to floating point to perform all math functions.

Floating Point Format for data

A number held in the floating point format allows both very big numbers and very small numbers to be stored and operated upon in the PC.

Using floating point format instead of the decimal format means that numbers that have fractional parts such as

PI, $\pi \approx 3.145927$

and numbers beyond the range of a double word device such as

Avogadro's constant $\approx 6.022 \times 10^{23} mol^{-1}$

can be used.

What Is Floating Point

Floating point is a method of storing numbers that require a whole and fractional part to be maintained. The speed of light C (299,792,458 m/s) could be stored in a double word but any multiplications would quickly give a number that is too big; which is why it would be better to use floating point format.

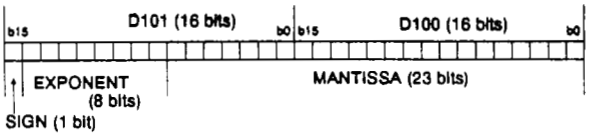
The difference between floating point in the PC and standard scientific notation is that the number base used in floating point is binary; the PC understands this better.

MITSUBISHI
PROGRAMMABLE CONTROLLERS

AN INTRODUCTION TO NUMBER FORMATS FOR THE FX

To further complicate our understanding both the mantissa and the exponent are stored together in a double word.

A Brief Look at the Float Structure



The sign bit is for the overall value; the exponent and the mantissa are held in binary. However, the actual interpretation of the bits is more complicated and further explanations are made in detail in sheet number FX-SDD007.

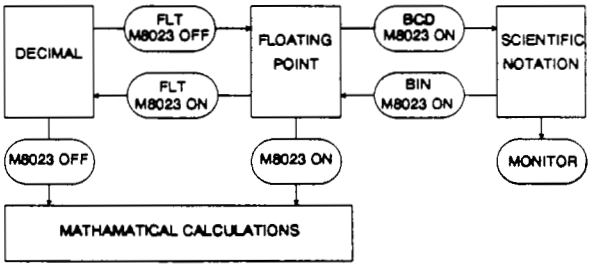
The whole pattern seems complex to our eyes and for this reason the FX also understands the scientific format of data.

Converting To Each Number Format

It is quite clear that each of these number formats are quite distinct in their construction and interpretation. We must have some method to convert between each type.

Within the FX there is a collection of functions designed to convert between each number format. This allows us to enter data in a format that we understand and then convert to the most appropriate format for the operation that we are doing.

The following figure depicts the conversion between each format and which functions should be used. To help find these functions the number of the associated data sheet explaining each function is listed below.



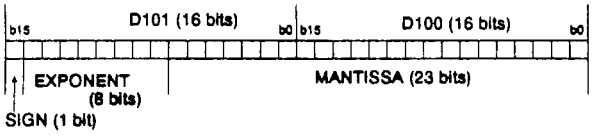
Operation		Sheet Number
FLT (M8023 OFF)		FX-SAD491
FLT (M8023 ON)		FX-SAD492
BCD (M8023 ON)		FX-SAT182
BIN (M8023 ON)		FX-SAT192
Mathematical Calculations on Decimal (M8023 OFF)	ADD	FX-SAL201
	SUB	FX-SAL211
	MUL	FX-SAL221
	DIV	FX-SAL231
	SQRT	FX-SAD481
Mathematical Calculations on Floating Point (M8023 ON)	ADD	FX-SAL202
	SUB	FX-SAL212
	MUL	FX-SAL222
	DIV	FX-SAL232
	SQRT	FX-SAD482

DETAILED EXPLANATION OF THE STRUCTURE OF FLOATING POINT

Introduction

To allow calculations with very small and very large numbers the FX PC uses floating point numbers. The format used in the FX PC has been taken from the recommendations of I.E.E.E (Institute of Electrical and Electronic Engineers) for the application of floating point in personal and micro computers.

Floating Point



The diagram above shows the bit map for a floating point number. The number is stored in two consecutive data registers and has a sign, mantissa and exponent.

In principal the actual number is calculated by

$$\pm \text{Mantissa} \times 2^{\text{Exponent}}$$

The multiplying factor is 2 because the values are held in binary.

Sign

The sign for floating point is only an indicator of the sign.

0 = positive value
1 = negative value

Exponent

The exponent is an 8 bit positive number that is interpreted to give an actual exponent of –126 to +127. This is achieved by subtracting the value 127 from the

E7	E6	E5	E4	E3	E2	E1	E0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

EXPONENT

actual value of the exponent.

Thus the exponent has a value of:

$$\left(E7 \times 2^7 + E6 \times 2^6 + \ldots + E0 \times 2^0\right) - 127$$

Mantissa

The mantissa is a 23 bit positive binary number. The values represent 23 binary places and the number is assumed to be preceded by a 1.

1.	A22	A21	A20	A19	...	A03	A02	A01	A00
2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}		2^{-20}	2^{-21}	2^{-22}	2^{-23}

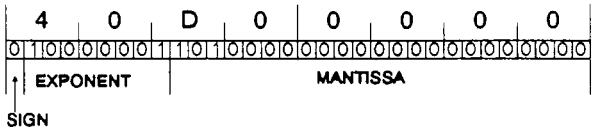
MANTISSA

Thus the mantissa has a value of:

$$1 \times 2^0 + A22 \times 2^{-1} + A21 \times 2^{-2} + \ldots + A0 \times 2^{-23}$$

Example

D101 = 16592 = 40D0 HEX
D100 = 0 = 0000 HEX



The sign is set to 0; positive.

The exponent is set to 10000001 which is equivalent to

$$\left(1 \times 2^7 + 0 \times 2^6 + \ldots + 1 \times 2^0\right) - 127$$
$$= \left(128 + 0 + \ldots + 1\right) - 127$$
$$= 2$$

The mantissa is set to 10100000000000000000000 which is equivalent to 1.101 binary or

$$1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + \ldots + 0 \times 2^{-23}$$
$$= 1 + \frac{1}{2} + \frac{0}{4} + \frac{1}{8} + \ldots + \frac{0}{8388608}$$
$$= 1.625$$

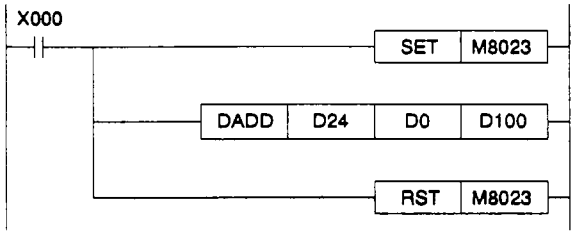
Therefore the number equals

$$+1.625 \times 2^2 = 6.5$$

DETAILED EXPLANATION OF THE STRUCTURE OF FLOATING POINT

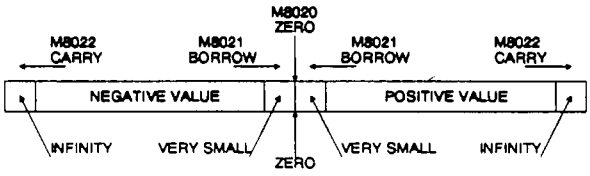
Floating Point In Action

To improve the accuracy of mathematical operations the FX math functions can use floating point numbers. This is done by setting special auxiliary relay M8023 float operation flag. The flag is reset after the floating point operation to allow normal mathematical calculations.



With normal mathematical operations when the result is out of range, flags are set to indicate the error. The same is true for floating point operations but the meaning of the flags is a little different.

The following shows the usage of the flags during floating point operations.



Flag	Float Operation
M8020 Zero	When result is zero
M8021 Borrow	Calculated result < smallest possible, but the result remains the smallest possible.
M8022 Carry	Calculated result > largest possible, but the result remains the largest possible.

The following table shows the ranges of valid values for floating point numbers.

Description	Sign	Exponent	Mantissa	Remark
Not A Number Not Used By FX	0 or 1	11111111 | 11111111	11111111111111111111111 | 00000000000000000000000	Exponent all 1's ± infinity (∞)
Normal Float	0 or 1	11111110 | 00000001	11111111111111111111111 11111111111111111111110 | 00000000000000000000001 00000000000000000000000	Largest Number $(\pm 3.403 \times 10^{38})$ Accuracy : 7 significant figures Smallest Number $(\pm 1.175 \times 10^{-38})$
Denormal Not Used By FX	0 or 1	00000000 | 00000000	11111111111111111111111 | 00000000000000000000001	Exponent all 0's
Zero	0 or 1	00000000	00000000000000000000000	All digits 0 (zero)

Execution Times

Basic Instructions

Mnemonic	Object Devices	Steps	Execution Time in μsec							
			FX0		FX0N		FX		FX2C	
			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	
LDI			3.2		3.2					
AND										
ANI										
OR										
ORI										
ANB	Not applicable	1	2.2		2.2		0.74		0.48	
ORB			2.0		2.0					
MPS										
MRD										
MPP										
MC	Nest level, M,Y	3	17	18.2	19.2	20.4	42.8	47.8	27	30
MCR	Nest level	2	6.0		6.2		40.4		19	
NOP	Not applicable	1	1.6		1.6		0.74		0.48	
END			410		470		960		700	
STL	S (see note 1)	1	4.2 + (8 × n)		6.4 + (6.8 × n)		39.1+ (21.1 × n)		25 + (13.5 × n)	
RET	Not applicable		8.0		12.4		40.5		20	

carried on over the page .....

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.

Mnemonic	Object Devices	Steps	Execution Time in μsec							
			FX0		FX0N		FX		FX2C	
			ON	OFF	ON	OFF	ON	OFF	ON	OFF
OUT	Y, M	1	3.2		3.2		0.74		0.48	
	S	2	7.0	7.2	7.0	7.2	50.0	48.1	26	24
	Special M	2	7.8	7.4	8.2	7.8	38.1	38.8	28	20
	T-K	3	21.8	19.4	25.2	21.0	72.4	52.6	45	34
	T-D	3	23.4	21.0	27.2	23.0	80.0	52.6	53	34
	C-K (16 bit)	3	14.6		17.8	15.6	67.9	40.3	42	25
	C-D (16 bit)	3	16.2		19.8	17.6	75.5	40.3	47	25
	C-K (32 bit)	5	12.5	6.0	16.0	8.6	82.3	40.3	51	25
	C-D (32 bit)	5	13.9	6.0	18.0	8.6	89.9	40.3	55	25
SET	Y, M	1	3.6	2.0	3.6	2.0	0.74		0.48	
	S	2	6.8	2.6	7.0	2.8	39.0	25.5	24	16
	S when used in an STL step (see note 1)		Not applicable				45.2+ (14.2 × n)	25.5	28 + (9 × n)	
	Special M	2	7.4	2.4	7.8	2.6	41.9	28.5	26	18
RST	Y, M	1	3.4	1.8	3.6	1.8	0.74		0.48	
	S	2	6.0	2.6	6.2	2.8	40.5	25.5	23	16
	Special M	2	7.4	2.4	7.8	2.6	41.8	28.9	26	18
	T, C	2	20.8	18.0	22.4	19.6	50.1	38.3	31	24
	D, V, Z and special D	3	10.0	2.8	9.2	3.0	35.5	25.5	22	16
PLS	Y, M	2	19.4		21.8		41.9	41.5	27	26
PLF	Y, M	2					42.7	40.6	26	25
P	0 TO 63	1	1.6		1.6		0.74		0.48	
I	I□□□	1								

Note 1 is located on the previous page.

Applied Instructions

Mnemonic	16/32 Bit	Execution Time in μsec									
		FX0		FX0N		FX			FX2C		
		ON	OFF	ON	OFF	ON	OFF	Pulse	ON	2nd FNC	Pulse
00 CJ	16	19.4	9.6	20.0	10.0	46.6	27.4	✔	29	–	✔
01 CALL	16	Not available				49.5	27.4	✔	31	–	✔
02 SRET	16					34.0			21	–	
03 IRET	*1	11.2		11.6		36.7			33	–	
04 EI	*1	6.4		7.8		62.6			34	–	
05 DI	*1					37.7			17	–	
06 FEND	*1	410		470		960			700	–	
07 WDT	16	9.2	5.6	9.8	7.0	35.9	25.1	✔	23	–	✔
08 FOR	*1	29.0		29.8		39.9			25	–	
09 NEXT	*1	12.4		12.4		29.1			19	–	
10 CMP	16	122.6	22.0	112.2	22.6	161.8	33.3	✔	49	–	✔
	32	129.2	30.4	118.6	31.6	189.0	39.9		57	–	
11 ZCP	16	140.0	25.0	128.4	25.8	186.9	33.3	✔	62	–	✔
	32	197.8	36.4	137.8	38.0	220.8	39.9		73	–	
12 MOV	16	46.2	18.0	47.2	18.4	78.4	33.3	✔	35	–	✔
	32	52.6	23.4	53.8	24.2	98.4	39.3		43	–	
13 SMOV	16	Not available				302.9	33.3	✔	170	–	✔
14 CML	16					74.0	33.3	✔	51	–	✔
	32					95.9	39.9		64	–	
15 BMOV *2	16	Not available		103.2 + (18.2n)	20.8	180.5 + (17.1n)	33.3	✔	118 + (10.6n)	–	✔

Mnemonic	16/32 Bit	Execution Time in μsec									
		FXg		FXon		FX			FX2C		
		ON	OFF	ON	OFF	ON	OFF	Pulse	ON	2nd FNC	Pulse
16 FMOV *2	16	Not available				107.6+ (5.3n)	33.3	✔	73+ (3.3n)	–	✔
	32					Not available			87+ (4.5n)	–	
17 XCH	16					90.3	33.3	✔	58	–	✔
	32					113.8	39.8		72	–	
18 BCD	16	63.6	18.0	64.4	18.4	130.9	33.3	✔	82	–	✔
	32	100.2	23.4	101.4	24.2	342.0	39.9		218	306	
19 BIN	16	64.4	18.0	66.2	18.4	135.4	33.3	✔	85	–	✔
	32	113.4	23.4	114.8	24.2	314.3	39.9		203	157	
20 ADD	16	69.4	21.0	70.8	21.6	115.5	33.3	✔	51	–	✔
	32	81.2	30.6	82.8	31.8	144.5	39.9		63	224	
21 SUB	16	69.8	21.0	71.6	21.6	116.6	33.3	✔	52	–	✔
	32	81.4	30.4	83.0	31.6	146.5	39.9		65	232	
22 MUL	16	89.4	21.0	91.0	21.6	133.4	33.3	✔	54	–	✔
	32	104.6	30.0	106.4	31.2	185.0	39.9		81	162	
23 DIV	16	119.2	21.0	120.8	21.6	139.5	33.3	✔	56	–	✔
	32	230.0	29.8	232.4	31.0	804.8	39.9		451	197	
24 INC	16	28.4	14.6	29.0	14.8	55.3	33.3	✔	26	–	✔
	32	33.4	17.0	34.2	17.4	65.4	34.4		29	–	
25 DEC	16	28.4	14.6	29.0	14.8	55.4	33.3	✔	26	–	✔
	32	33.6	17.0	34.4	17.4	65.1	34.4		29	–	
26 WAND	16	64.2	21.0	65.6	21.6	108.0	33.3	✔	67	–	✔
	32	73.0	29.4	74.6	30.6	135.4	39.9		83	–	
27 WOR	16	64.2	21.0	65.6	21.6	107.9	33.3	✔	67	–	✔
	32	73.0	29.4	74.6	30.6	135.5	39.9		83	–	
28 WXOR	16	64.2	21.0	65.6	21.6	106.5	33.3	✔	67	–	✔
	32	73.0	29.4	74.6	30.5	133.9	39.9		82	–	
29 NEG	16	Not available				55.1	33.3	✔	34	–	✔
	32					65.5	34.4		41	–	

See end of section for ✱ notes...

Mnemonic	16/32 Bit	Execution Time in μsec									
		FX0		FX0N		FX			FX2C		
		ON	OFF	ON	OFF	ON	OFF	Pulse	ON	2nd FNC	Pulse
30 ROR *3	16	Not available				91.9+ (3.0n)	33.3	✔	57+ (1.8n)	–	✔
	32					113.8+ (3.5n)	39.9		70+ (2.1n)	–	
31 ROL *3	16					91.9+ (3.0n)	33.3	✔	57+ (1.8n)	–	✔
	32					113.8+ (3.5n)	39.9		71+ (2.3n)	–	
32 RCR *3	16					99.0 + (1.4n)	33.3	✔	61 + (0.9n)	–	✔
	32					120.8+ (1.8n)	39.9		75+ (1.2n)	–	
33 RCL *3	16					99.0 + (1.4n)	33.3	✔	62 + (0.9n)	–	✔
	32					120.8+ (1.8n)	39.3		75+ (1.2n)	–	
34 SFTR *4	16	145.2+ (5.1n)	24.6	156.4+ (4.8n)	25.4	180.8+ (70.0n)	33.3	✔	172+ (42n)	–	✔
35 SFTL *4	16	150.6+ (5.1n)	24.2	162.4+ (4.8n)	25.0	180.8+ (70.0n)	33.3	✔	172+ (42n)	–	✔
36 WSFR *2	16	Not available				218.6+ (18.0n)	33.3	✔	147+ (11n)	–	✔
37 WSFL *2	16					218.6+ (18.0n)	33.3	✔	147+ (11n)	–	✔
38 SFWR	16					138.1	33.3	✔	87	–	✔
39 SFWL *5	16					143.1+ (6.8n)	33.3	✔	84	–	✔
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)	–	✔
	16 (S)	127.0+ (2.9n)		131.5+ (2.9n)		161.3+ (16.5n)			121+ (10.5n)		
	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)		
	16 (Y)	127.0+ (3.5n)		131.5+ (3.5n)							

See end of section for * notes...

Mnemonic	16/32 Bit	Execution Time in μsec									
		FXo		FX0N		FX			FXec		
		ON	OFF	ON	OFF	ON	OFF	Pulse	ON	2nd FNC	Pulse
41 DECO	16	881.4	20.6	932.0	21.4	114.8	28.8	✔	72	–	✔
42 ENCO	16	618.3	20.6	692.4	21.4	125.6	28.8	✔	79	–	✔
43 SUM	16	Not available				133.5	33.3	✔	84	–	✔
	32					196.6	39.9		123	–	
44 BON	16					168.9	33.3	✔	98	–	✔
	32					177.6	39.9		112	–	
45 MEAN *7	16					133.4+ (12.2n)	33.3	✔	84+ (7.7n)	–	✔
	32					Not available			105+ (9.8n)	–	
46 ANS	16					192.6	165.6		120 (off 110)	–	
47 ANR	16					86.5	25.5	✔	54	–	✔
48 SQR	16	Not available							208	–	✔
	32								220	344	
49 FLT	16								98	–	✔
	32								114	–	
50 REF *8	16	53.6	12.8	65.4+ (3.1n)	13.4	145.3+ (3.6n)	33.3	✔	67+ (3.2n)	–	✔
51 REFF *9	16	Not available				56.0+ (4.9n)	33.3	✔	35+ (3.1n)	–	✔
52 MTR	16					87.3	39.3		53 (off 26)	–	
53 HSCS *10	32	75.6	6.6	82.8	7.8	175.0	39.3		115	–	
54 HSCR *10	32									–	
55 HSZ *10	32	Not available				240.3	39.3		142	–	
56 SPD	*1					164.4	163.0		102	–	

See end of section for * notes...

Mnemonic	16/32 Bit	Execution Time in μsec									
		FX0		FX0S		FX			FX2C		
		ON	OFF	ON	OFF	ON	OFF	Pulse	ON	2nd FNC	Pulse
57 PLSY	16	189.4	10.0	212.4	21.4	154.5	173.6		101	–	
	32			223.4	31.4				115	–	
58 PWM	16	42.5	7.8	44.2	18.6	139.8	171.0		86	–	
59	16	Not available									
	32										
60 IST	16	766.0	322.4	212.4	21.4	272.9	33.3		153	–	
61 SER	16	Not available							147 + (12.5n)	–	✔
	32								168 + (17.4n)	–	✔
62 ABSD *11	16	Not available				141.4+ (61.4n)	33.3		91+ (35n)	–	
	32					Not available			110+ (43n)	–	
63 INCD	16					208.8	39.9		130 (off 26)	–	
64 TTMR	16					81.3	69.6		48	–	
65 STMR	16					176.6	167.8		106	–	
66 ALT	16	61.0	9.8	62.8	10.0	105.6	33.3	✔	66	–	✔
67 RAMP	16	248.6	82.6	250.4	84.6	181.8	134.5		113	–	
68 ROTC	16	Not available				232.5	209.1		144	–	
69 SORT	16	Not available							62 + (32.3n)	–	
	32								–	–	
70 TKY	16	Not available				245.7	33.3		153	–	
	32					229.1	39.9		145	–	
71 HKY	16					318.8	39.9		189	–	
	32					338.0	45.5		205	–	
72 DSW	16 – 1set					205.8	39.3		130	–	
	16 – 2sets					208.1			N.A.	–	

See end of section for ✱ notes...

(N.A. – Data Not Available At Time Of Publication)

Mnemonic	16/32 Bit	Execution Time in µsec									
		FX0		FX0N		FX			FX2C		
		ON	OFF	ON	OFF	ON	OFF	Pulse	ON	2nd FNC	Pulse
73 SEGD	16	Not available				142.1	33.3	✔	87	–	✔
74 SEGL	16 – 1set					209.7	33.3		127	–	
	16 – 2sets					246.9			148	–	
75 ARWS	16					285.0	163.0		169	–	
76 ASC	16					130.9	33.3		104	–	
77 PR	16 – printing					207.1	112.6		127	–	
	16 – ready					112.1			68	–	
78 FROM *12	16	Not available		120+ (400n)	26	170+ (406n)	45.0	✔	120+ (325n)	–	✔
	32			120+ (800n)	26	200+ (800n)			140+ (749n)	–	
79 TO *12	16			120+ (480m)	38	151+ (480n)	45.0	✔	106+ (384n)	–	✔
	32			120+ (950n)		200+ (936n)			140+ (749n)	–	
80 RS	16	Not available							132	–	
	32								132	–	
81 PRUN *13	16	Not available				137.1+ (53.5n)	33.3	✔	91+ (32n)	–	✔
	32					154.5+ (49.3n)			104+ (31n)	–	
82 ASCI	16	Not available							94 + (12n)	–	✔
83 HEX	16								95 + (23n)	–	✔
84 CCD	16								96 + (8n)	–	✔
85 VRRD	16	Not available				308.1	33.3	✔	209	–	✔
86 VRSC	16					319.1		✔	205	–	✔
87	16	Not available									
	32										
88	16										
	32										
89	16										
	32										

See end of section for * notes...

Mnemonic	16/32 Bit	Execution Time in μsec									
		FX0		FX0N		FX			FX2C		
		ON	OFF	ON	OFF	ON	OFF	Pulse	ON	2nd FNC	Pulse
90 MNET	16	Not available				643.9	25.5	✔	498	–	✔
91 ANRD	16					1,137	33.3	✔	846	–	✔
92 ANWR	16					1.387	470.9	✔	1,073	–	✔
93 RMST	16					948.8	950.0		691	–	
94 RMWR	16					2,214	33.3	✔	1,612	–	✔
	32					4,235	39.9		3,127	–	
95 RMRD	16					1,684	33.3	✔	1,254	–	✔
	32					3,168	39.9		2,414	–	
96 RMMN	16					1,589	33.3	✔	1,195	–	✔
97 BLK	16					672.4	669.3	✔	511	–	✔
98 MCDE	16					740.3	33.3	✔	554	–	✔
99	16	Not available									
	32										

✱1:

- These instructions require NO preliminary contact devices such as LD , AND, OR etc.

✱2:

- Where "n" is referred to this identifies the quantity of registers to be manipulated. "n" can be equal or less than 512.

✱3:

- Where "n" is referred to this identifies the quantity of bit devices to be manipulated. "n" can be equal or less than selected operating mode, i.e. if 32 bit mode is selected then "n" can have a value equal or less than 32.

✱4:

- Where "n" is referred to this identifies the quantity of bit devices to be manipulated. When an FX programmable controller 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 programmable controllers are involved in a parallel running function.

Device Specifications

The following tables list the devices applicable to the FX2C series.

ITEM	GENERAL SPECIFICATION	REMARK
Operation control method	Cyclic operation by stored program	
I\O control method	Batch processing (takes place after END instruction is executed)	Direct I/O control, refresh and input filter adjustment is available
Operation process time	Basic instructions 0.48 μs	Applied instructions; up to several 100 μs
Programming language	Relay symbolic language + Stepladder	SFC expression is possible
Program capacity/memory type	2K step RAM incorporated (standard)	Comment registration is possible (program memory is used). Alphanumerics (15 char/comment),
	Optional Memory cassettes max. 8K	10 steps/comment; program memory is reserved in units of 50 comments.
Number of instructions	Sequence (basic) instructions; 20, Stepladder instructions; 2, Applied instructions; 94	

New Special M coils and Special Data registers

M8023	Floating point operation flag
M8038	RAM file register area all clear
M8059	Interrupts I010,..,I060 disable
M8074	Use RAM file registers mode
M8121 to M8124	RS instruction control flags
M8130 to M8133	HSZ instruction New operation flags
M8160	XCH as byte swap function
M8161	ASCII 8 bit mode flag
M8162	High speed parallel run mode (only 2 words of data in each direction)
M8167, M8168	HKY and SMOV (respectively) use hexadecimal data
M8170 to M8175	Inputs X0 to X5 (respectively) pulse catch flags
M8181 to M8186	Alternative to I010,...,I060 (for use with old versions of programming tools)
M8190 to M8197	Convert existing functions (MOV, SMOV, RAMP, FMOV) to be interpreted as new functions (for use with old versions of programing tools)
M8198	Swap source and destination of BMOV (to allow write to file register with old versions of programming tools)
D8120 to D8125	RS instruction control registers
D8130 to D8135	HSZ instruction New operation registers
D8136, D8137	PLSY double word pulse count

Device Specification Table

ITEM		GENERAL SPECIFICATION			REMARK	
Input Spec.	DC input	24V DC, 7mA, opto-isolated			X0 to X377 max. (numbered in octal)	Maximum number of I/O points 256
Output Spec.	Transistor	Direct output from the PC is available at 30V DC, 0.1A per point, 0.8A max. per 4 point gang. Alternatively connection to termination blocks is possible.			Y0 to Y377 max. (numbered in octal)	
Auxiliary (internal) relay	General use	-			M0 to M499 (500 points)	Areas sizes and ranges can be changed by user edited parameter settings
	Latched	Automatically backed up by battery			M500 to M1535 (1036 points)	
	Special purpose	-			M8000 to M8255 (256 points)	
State relay	Initialization states	Can be used to start/initialize a sequence of STL program			S0 to S9 (10 points)	
	General use	-			S10 to S499 (490 points)	
	Latched	Automatically backed up by battery			S500 to S899 (400 points)	Areas, sizes and ranges can be changed by user edited parameter settings
	Annunciator	Automatically backed up by battery			S900 to S999 (100 points)	
Timer	100 msec	0.1 to 3,276.7 sec.			T0 to T199 (200 points)	
	10 msec	0.01 to 327.67 sec.			T200 to T245 (46 points)	
	1 msec - retentive	0.001 to 32.767 sec.		Automatically backed up by battery	T246 to T249 (4 points)	
	100 msec - retentive	0.1 to 3276.7 sec.			T250 to T255 (6 points)	
Counter	Up counter	16 bits (1 to 32,767 counts)		General use	C0 to C99 (100 points)	Areas, sizes and ranges can be changed by user edited parameter settings
				Backed by battery	C100 to C199 (100 points)	
	Up/Down counter	32 bits (-2,147,483,648 to +2,147,483,647 counts)		General use	C200 to C219 (20 points)	
				Backed up by battery	C220 to C234 (15 points)	
	High speed counter	32 bits up/down (-2,147,483,648 to +2,147,483,647 counts)		Backed up by battery	6 points of 1-phase counters (C235 to C240), 5 points of 1-phase counters, with external start and stop, (C241 to C245), 5 points of 2-phase counters (C246 to C250), 5 points of 2-phase, A-B phase type (C251 to C255), *see note 1*	
Data registers	General purpose data register	16 bits	Pair for 32 bit data register	General use	D0 to D199 (200 points)	Areas, sizes and ranges can be changed by user edited parameter settings
		16 bits		Backed up by EEPROM	D200 to D999 (800 points)	
	File registers	16 bits (in program memory) backed up by battery			D1000 to D2999 (2000 points)	
	RAM File regs	16 bits (in CPU memory) backed up by battery			D6000 to D7999 (2000 points)	Activated with M8074
	Special register	16 bits			D8000 to D8255 (256 points)	
	Index register	16 bits			V and Z (2 points)	
Pointer	For JUMP/CALL	Use with conditional jump (CJ, FNC00) applied instr.			P0 to P127 (128 points)	
	Interrupt	Triggered by inputs X000 to X005 and timer interrupt			I00□ to I80□ (9 points) [□ = 1 on trigger, □ = 0 on release]	
Nesting		Use with master control (MC) basic ladder instruction			Nest levels, N0 to N7 (8 points)	
Constant	Decimal	16 bits: -32,768 to +32,767			32 bits: -2,147,483,648 to +2,147,483,647	
	Hexadecimal	16 bits: 0 to $FFFF_H$			32 bits: 0 to $FFFFFFFF_H$	

Note 1: Not all high speed counters can be operated at the same time as they share the same special inputs; X000 to X007.

Under no circumstances will Mitsubishi Electric be liable or responsible for any consequential damage that may arise as a result of the installation and/or programming of the products associated with this manual.

All examples and diagrams shown in this manual are intended as an aid to understanding the text, not to guarantee operation. Mitsubishi Electric will accept no responsibility for actual use of the product based on these illustrative examples.

Owing to the very great variety of possible applications, users must satisfy themselves as to the suitability of each specific application.

FX_{2C} SUPPLEMENTARY PROGRAMMING MANUAL

THE FX_{2C} PROGRAMMABLE CONTROLLER

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE: MITSUBISHI DENKI BLDG MARUNOUCHI TOKYO 100 TELEX: J24532 CABLE MELCO TOKYO
HIMEJI WORKS: 840, CHIYODA CHO, HIMEJI, JAPAN

Effective MAR. 1995
Specifications are subject to change without notice.

JY992D50201B
HI-IB-151-B (9503) (SEN) (B1)