UDC 621 316.7

SOFTWARES FOR MICREX-F SERIES

Keiichi Tomisawa
Shin Hashimoto
Yo Kikuchi
Toshiyuki Koido
Choji Kikuchi

1 FOREWORD

Language and instruction set of programmable controller (hereinafter abbreviated as PC) are the first contact point with users and the most important for determining performance of PC's.

In particular, recent situations are:

(1) Diffusion of simplified and programless languages in personal computers, and

(2) Possibility of use of advanced functions, as a result of lowering of cost in semi-conductor products.

Also in the field of PC's the language is gradually changing into one that is optimum for describing the control system from one that is stiff due to restriction from hardware.

Fuji Electric has taken the lead by preparing, EPOL (MICREX-E Problem Oriented Language) that can be directly programmed from the control block diagram. In the MICREX-F series (hereinafter abbreviated as F series) the knowhows that we have acquired through our experience in both series of MICREX-E and FUJILOG-μ are built in and we have developed FPL (F-series Programming Language) which is more friendly with application side.

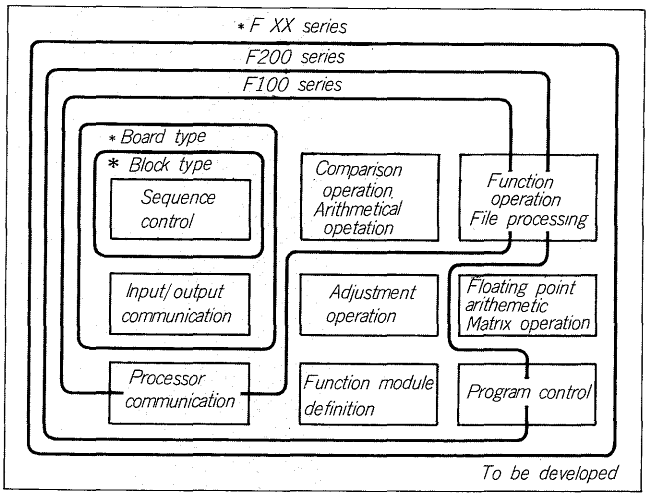
2 FEATURES

As a common language to all of F series, from instrument-image general-purpose sequencer to high-function controller in DDC and plant monitor and control level, FPL has the following features.

2.1 Hierarchy of functions

Fig. 1 shows the hierarchy of functions of FPL. Each processor has a set of instruction necessary in its function level, and the specifications of the instructions common to all equipment within the series. Also, as the group of loaders can support them in function-level of the loaders, the loaders can be utilized for all processors without depending utilized for all purpose without the necessity of depending on the type of processors.

Fig. 1 Hierarchy of F series command functions


* F XX series
F200 series
F100 series
* Board type
* Block type
Sequence control
Input/output communication
Processor communication
Comparison operation Arithmetical opetation
Adjustment operation
Function module definition
Function operation File processing
Floating point arithemetic Matrix operation
Program control
To be developed

2.2 Symbolic programming

Rudder diagrams and block diagrams which are generally used for describing controls are used for instruction language, and these instructions are carried out directly by LSI. In particular, since diagram positions and connecting informations are programmed, the original diagram can easily be reproduced from the program itself and this will become a program document by itself. This also means that it be comes easy to realize and build these support function in other equipment besides the program loader, and this widens the field of applications.

2.3 Moduled block

A moduled structure that can control in divided form the program and function modules (sub-routine) by each function blocks and processing units, is adopted for making the use of F-series much easier in comparison with the existing MICREX-E series, in particular in those concerned with starting timing and delivery and reception of parameters.

Also, as all data are moduled and filing instructions are intensified, data handling is made easier.

Table 1 List of instruction set

Classification	Designation	Symbol	F100 Series	F200 series
Sequence	a contact	—┤├—	○	○
	b contact	—┤/├—	○	○
	Coil	—( )—┤	○	○
	Set	—(S)—┤	○	○
	Reset	—(R)—┤	○	○
	Rise differential	—(↑)—┤	○	○
	Fall differential	—(↓)—┤	○	○
	Inversion	—/—	○	○
	Shift register	┬[SR]┤	○	○
	Step sequence	Sxx.xx Sxx.xx —┤├—( )┤	○	○
Timer	ON delay timer	—[TON]┤	○	○
	OFF delay timer	—[TOF]┤	○	○
	Integrating timer	—[S TMR R]┤	○	○
	Mono stay	—[MS]┤	○	○
	Mono stay (triggerable)	—[MR]┤	○	○
Counter	Counter	—[↑ CTR R]┤	○	○
	Down counter	—[↓ CTD R]┤	○	○
	Up-and-down counter	—[↑ ↓ UDCT]┤	○	○
	Ring counter	—[↑ RCTR R]┤	○	○
Arithmetical operation	Addition	┬[+]—	○	○
	Subtraction	┬[−]—	○	○
	Multiplication	┬[×]—	○	○
	Division	┬[÷]—	○	○
	Division (Rounding)	┬[÷R]—	○	○
	Root	—[√‾]—	○	○
	Absolute value	—[ABS]—	○	○
	Inversion sign	—[+/−]—	○	○
	Increment	—[+1]—	○	○
	Decrement	—[−1]—	○	○

Classification	Designation	Symbol	F100 series	F200 series
Comparison	>	┬[>]—	○	○
	≧	┬[>=]—	○	○
	=	┬[=]—	○	○
	≦	┬[<=]—	○	○
	<	┬[<]—	○	○
	≠	┬[≠]—	○	○
	File comparison	┬[REF]—		○
Logical operation	AND	┬[AND]—	○	○
	OR	┬[OR]—	○	○
	EOR	┬[EOR]—	○	○
	Inversion	—[INV]—	○	○
	Shift right logical	┬[SRL]—	○	○
	Shift left logical	┬[SLL]—	○	○
	Set bit	┬[SBIT]—		○
	Reset bit	┬[RBIT]—		○
	Test bit	┬[TBIT]—		○
Conversion	Binary/BCD	—[BCD]—	○	○
	BCD/binary	—[BIN]—	○	○
	Decode	—[DECO]—	○	○
	Encode	—[ENCO]—	○	○
	7 Segment	—[7SEG]—	○	○
	Individual count	—[BCNT]—	○	○
Transfer Function	sin	—[SIN]—		○
	cos	—[COS]—		○
	tan	—[TAN]—		○
	$\sin^{-1}$	—[ASIN]—		○
	$\cos^{-1}$	—[ACOS]—		○
	$\tan^{-1}$	—[ATAN]—		○
	Transfer	—[MOV]—	○	○
	Block transfer	┬[BT]—	○	○
	Digit transfer	┬[DT]—	○	○
	Superior digit transfer	—[MOVU]—	○	○
	Inferior digit transfer	—[MOVL]—	○	○
	Pattern clear	┬[PC]—		○
	Search	┬[SRCH]—		○
	Switch	┬[SW]—		○

Classification	Designation	Symbol	F100 series	F200 Series
Analog	Upper limit	┬[—⌐‾]—	○	○
	Lower limit	┬[_⌐—]—	○	○
	Upper and lower limits	┬[_⌐‾]—		○
	Dead zone	┬[⁄—⁄]—		○
	Bias	┬[⁄‾]—		○
	Filter	┬[FIL]—		○
	Differential	┬[DIF]—		○
	Integration	┬[INT]—		○
	Sample hold	—[HOLD]—		○
	Multipercent	┬[MLTP]—		○
	Divided percent	┬[DIVP]—		○
File	FIFO store (FILO)	┬[FFST]—	○	○
	FIFO load	┬[FIFO]—	○	○
	FILO load	┬[FILO]—	○	○
	File definition	┬[FILE]—	○	○
	File clear	┬[FLCL]—	○	○
	Selector	—[SEL]—	○	○
	Deselector	—[DSEL]—	○	○
	File read out	┬[RFIL]—		○
	File write in	┬[WFIL]—		○
	File information	—[FINF]—		○
Program control	Program entry	—[PROG]—	○	○
	Program end	—[PEND]—	○	○
	FM call	—[FM]—		○
	FM start	—[FMS]—		○
	FM end	—[FME]—		○
Branching	Skip	—[SKIP]—		○
	Skip end	—[SEND]—		○
	N skip	—[NSKP]—		○
	Jump	—[JMP]—	○	○
	Jump end	—[JEND]—	○	○
Loop	Loop	—[LOOP]—		○
	Continue	—[CONT]—		○
System command	PUSH	—[PUSH]—		○
	POP	—[POP]—		○
	LEA	—[LEA]—		○
	CHT	—[CHT]—		○

3 FUNCTION

Table 1 shows a list of instruction set for F100 series and F200 series.

The instructions used in sequence controls as contacts, relays, counters and timers, and step sequence instruction describing order control, status transition diagram and others are expressed in the rudder diagram.

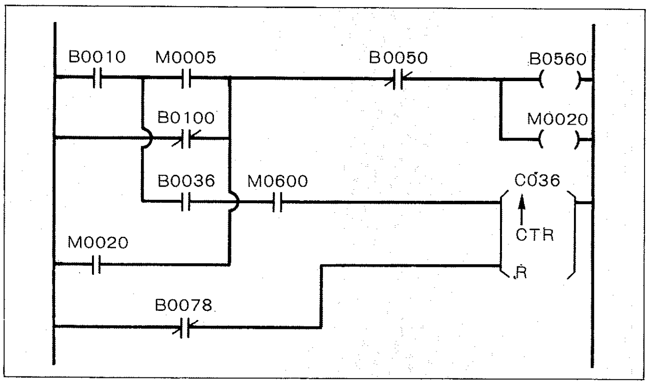
Various arithmetical operations, function computation and adjustment operation on PID and limiters that are mainly processed numerically are made out by block diagrams and line diagrams. The standard operational precision is in order of 8 digits BCD code (up to ± 79,999,999), and programming and debugging are easy.

3.1 Rudder diagram

An example of rudder diagram is shown in *Fig. 2*. This is programmed by a program loader in a form of diagram is and stored in the reagion of user's program in a form of symbolic expression. Therefore, when the rudder diagram is saved and loaded, positions of contacts and coils, and connection relationship can be reproduced perfectly. Also, in this expression, the rudder line intersections (AND conections of B0010 amd B0036 of the figure) will be made possible and the sequence that user wants to express can freely be made.

The execution of the rudder diagram is processed in high speed by a bit processor dedicated to process rudder diagram in a form of symbol.

Fig. 2 An example of rudder diagram expression

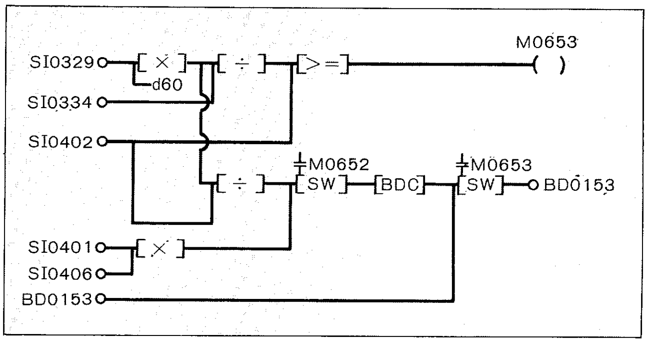


3.2 Block diagram

As one of the methods for expressing the numercial procesing, F series uses the block diagram method. This method can be handled with symbolic expression as in the case of rudder diagram mentioned above. *Fig 3* shows an example of description. In this method, as the principle, the input is either one or two, while the output is one, and the contact input used for switch instruction will be set to the upper part of the block (like M0652 of the figure).

Since the operation output can be directly designated as the input to the subsequent stage, the working memory need not be designated during the programming. By this, a program that proceeds along the effective processing flow can be faithfully expressed and, the processing speed can also be improved.

Fig. 3 An example of block diagram expression

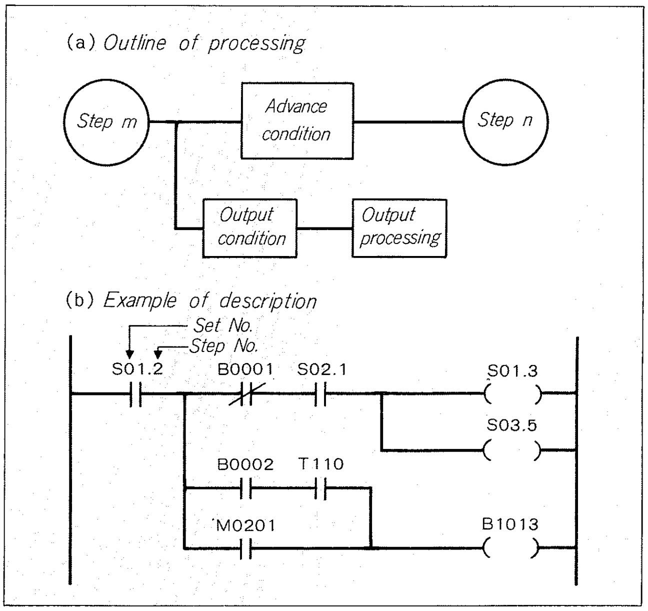


3.3 Step sequence

By using the step sequence command, a process advance type sequence operation can be realized. The step sequence operation, as shown in *Fig. 4* (a), controls the transition from mth step to the nth step under advance condition, and expresses the in-step processing with output condition and output processing. At this time, when the status has transited from the mth step. to nth step, the output of the mth step will be reset.

In F series, the above mentioned step sequence instruction can be programmed by the image of the rudder diagram, Fig. 4 (b) shows an example of the step sequence. As shown in the figure, a series of processing will be handled as one set as a processing unit and step sequences up to 100 sets can be made out in total. Also, among this set, the

Fig. 4 Step sequence processing


(a) *Outline of processing*

(b) *Example of description*

maximum of 100 step process can be controlled. Among each step, since the different set of the step can be designated as the next step and advance condition, complex systems having process branching and process waiting can efficiently be controlled.

The status of advance of the step is kept even during the failure of power supply for the system so that, when the power is restored, the equipment can be operated continuously. The advance condition of the step can be on-line displayed by the programming loader. The execution of the step sequence instructions are processed by bit processor in high speed.

3.4 Data modules

Both bit/word data of F series is divided into function units called data modules and managed

The following two can be cited as the features of the moduling.

(1) Programming that needs not use absolute address

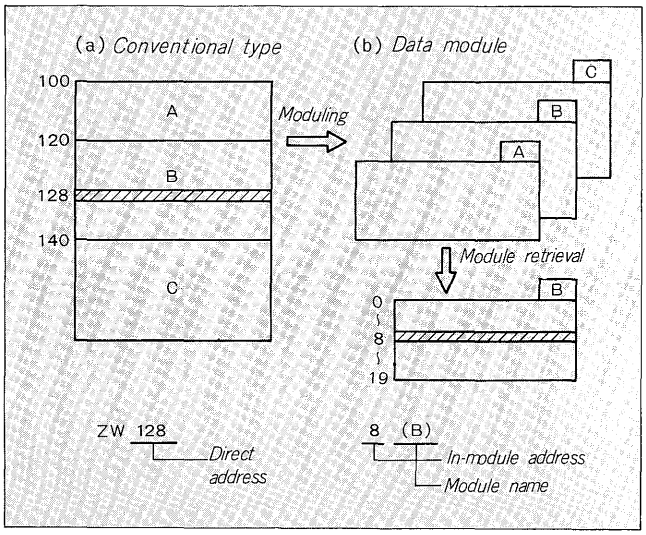
The mehod to access data is not through absolute address designation as in conventional cases but relative address designation in the module. Consequently, the programmer can assign and manage memories easily by being conscient of module name (or number) even if it is a large-scale system.

Fig. 5 shows a conceptual diagram of the data module.

(2) Perfectioning of the region over supervision (PAS)

In contrast of the fact that in conventional type of the region check is of hardware type (that is, depending on the fact that the memory is mounted or not), in F series, the region check is effectuated for module unit. Also, index address (indirect address designation) under program execution is checked by module units, so that the data destruction due to runaway of the application program can be limited to the minimum.

Fig. 5 Conceptual diagram of data module



3.5 Filing instruction

As the installation has come to be higher in functionality, not only the calculating function but also functions for controlling and processing large quantity of data have come to be necessary.

For this, F series has been improved in filing instruction as mentioned in the following.

(1) Large filing reagion

F series have a capacity of 4 k words (8 k words, if combined with that of data region) as filing region. The region can be defined freely by user as the maximum of 50 units of application file.

(2) Two-dimensional file definition

The standard type of the file for F200 series is two-dimensional file. File processing is, as shown in *Fig. 6*, effectuated by file definition instruction and file read out (or write in) instruction. The file definition instruction elaborates two-dimensional files of block size x no. of blocks by data type designated. Also, the file read out (write in) instruction designates the address within the file by the block no. and in-block no. and transfers those corresponding to the number of read out words to destination address (transfer read out data to the file).

(3) FIFO/FILO file processing

All files can be used not only as table files but also as FIFO (file in file out) and FILO (file in line out) files. By this, monitoring on the installation and tracking control can easily be made.

An example of use of FIFO file in automobile paint line is shown in *Fig. 7*

Fig. 6 File processing

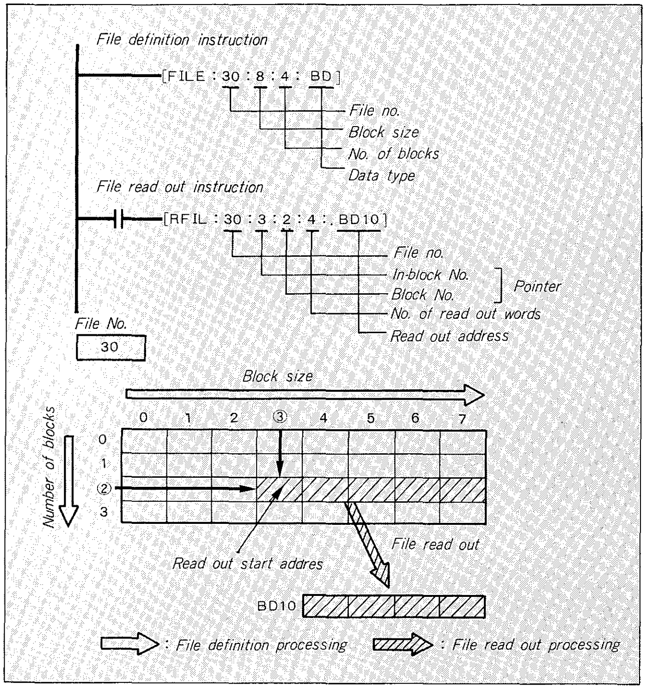


Fig. 7 Example of application of FIFO file

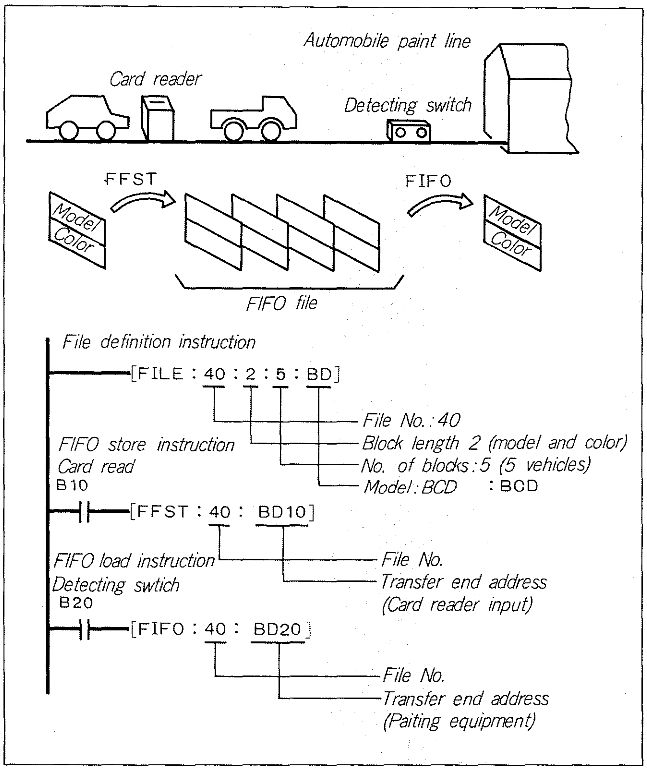
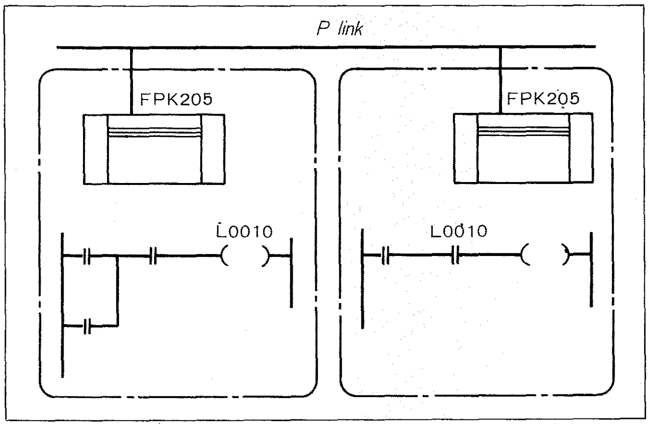


Fig. 8 P-link conceptual diagram



3.6 Input/output processing

The input/output region will be refreshed when the program scanning is over. So that, the execution of processing program and input/output data renewal are always synchronized. By this reason, the users need not worry about the change in input signals during the scanning of the program.

AND, in F200 series, a instruction that refreshes only the designated inputs and outputs (CHT instruction) is prepared. By this instruction, it was made possible to control processing of inputs and outputs that require high speed response such as constant cycle interruption and process interruption by the application.

By using the CHT instruction in the interruption program, high speed refreshing sychronized to interruption program can be effectuated.

3.7 P link processing

P link region is assigned to the memory of each capsule and when a capsule changes data of that region, by the function of P link interface, the P link region of other capsule will be changed into the same data. Consequently, the data assigned to P link, though they are in other capsule, can be accessed as if they are in their own capsule. As the result, reciprocal inter-lock and supervision in processor systems can easily be attained by the program.

Fig. 8 shows an example. The bit information written as L0010 in the rudder diagram of one FPK205 can be read out as L0010 in a rudder diagram of another FPK205.

4 PROGRAM LOADER

In order to exert fully the features of FPL symbolic language, two types of program loaders, namely, D10 and D20 are developed.

D10 handles rudder diagrams and line diagrams while D20 processes mainly rudder diagram and block diagrams. Programming is completed by elaborating the rudder diagram and block diagram on the screen. So that the debugging can easily be effectuated by the real time data displayed on the screen.

Also, as the loader interfaces are standardized and unified from the view point of both hardware and software, they can be commonly used without having to depend on the loader type or processor type.

Furthermore, besides the fact that two loaders can be used simultaneously, this loader is provided with many features as that it can be connected to either one of P capsule and T capsule. As for the details of these features, the readers are requested to refer to the separate article of this brochure under the tile of "MICREX-F series, Programming Tools".

5 SUMMARY

Together with the diffusion of PC's and expansion of their field of application, the importance of their programming method and set of instruction will become more and more important from now on.

The symbolic programming designated by EPL and set of instructions directly coupled with the applicaton and the documenting of the programs are thought to be showing the tendency of future current of PC's.

Furthermore, the description of system specification, or programming by description of control method and the method of elaboration of dialog with PC these we consider as urgent demand for objects of the next stage development and we intend to carry on with these developments in response to the demand of the time.