UDC 681.323.06: [681.323:62-523]

PROGRAMMING LANGUAGE FOR FLEX-PC N SERIES

Yukio Fukui
Yutaka Yatsuda
Masayoshi Matsumoto

1. FOREWORD

The FLEX-PC N Series is a PC with various functions which flexibly meet the hardware and software needs of each machine with "programmable controller (PC) that just fits the individual specifications of each machine" as one of its basic concepts. This article introduces the features and architecture of its flexible software multi-language and user macro functions.

The FLEX-PC N Series which has gained attention as a technique which can represent complex and sophisticated control contents more simply and has quickly incorporated and implemented the SFC (Sequential Function Chart), a high-level concept of the ladder diagram and other conventional PC programming which is being standardized by the IEC (International Electrotechnical Commission), is introduced here.

Fig. 1 Multi-language concept

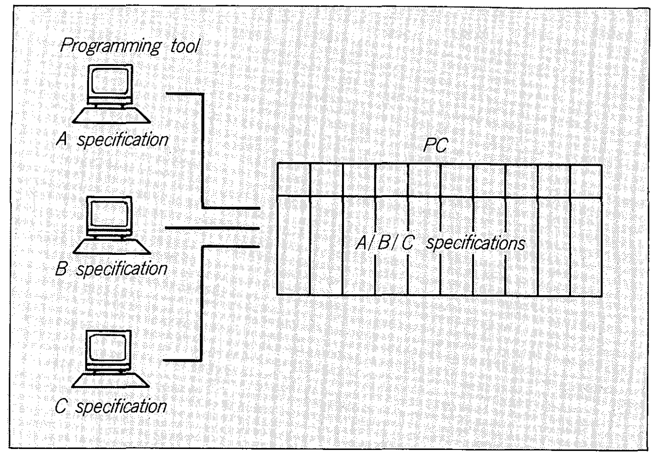
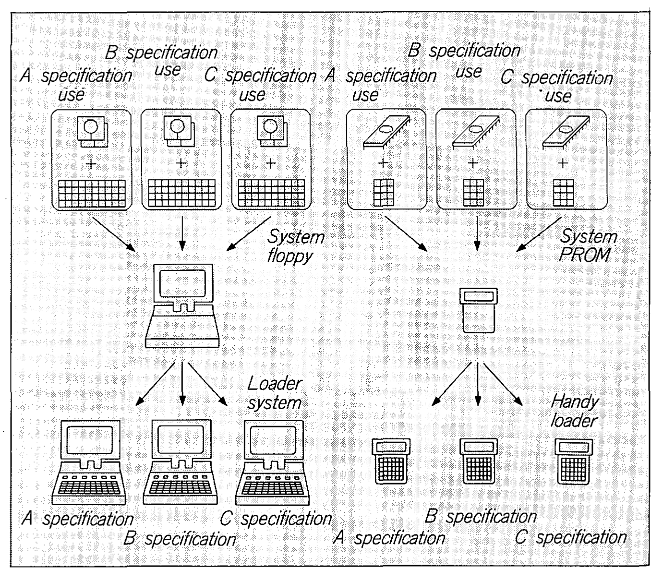


Fig. 2 Implementation by programming tool



2. MULTI-LANGUAGE

2.1 Development aim

The PC market is expanding and on the other hand, more than 60% of users have PCs from different manufacturers considering economic rationality, supply stability, and functions. The differences in the programming language and programming tool operability between PC manufacturers has now become a problem. To meet this problem, Fuji Electric has developed the FLEX-PC N Series as a PC capable of easily supplying PC and tool operability with thich users are already familiar with using. The function which deals with these multiple languages and tool operability is called "multi-language".

2.2 Overview

Multi-Language can efficiently supply PC and programming language and operability for each PC specificaion with which the user is familiar with using as shown in *Fig. 1*.

2.3 Implementation of multi-language

2.3.1 Implementation by programming tool

The programming tool implements multi-language as shown in *Fig. 2*.

The operating environments with which the user is familiar with using are (1) key arrangement, (2) key operation, (3) display screen, (4) message, etc.

Therefore, the basic hardware of the tool is standardized and the programming environment of each specification can be supplied by keyboard and system software.

Fig. 3 PC instruction

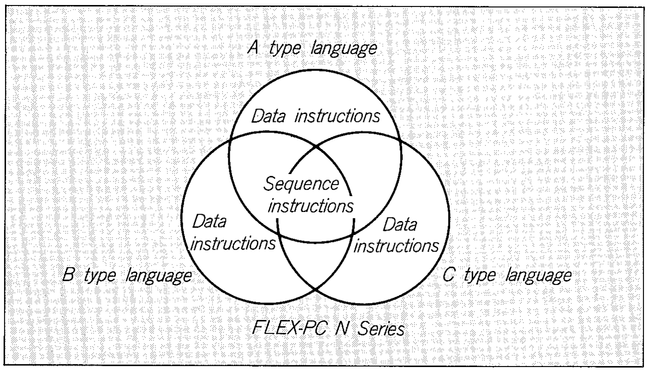


Fig. 4 Sequence instruction

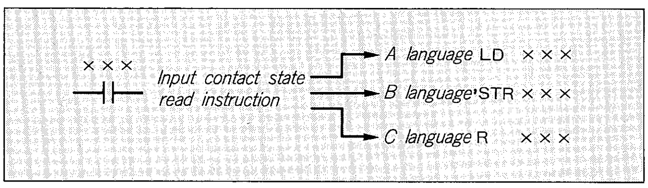
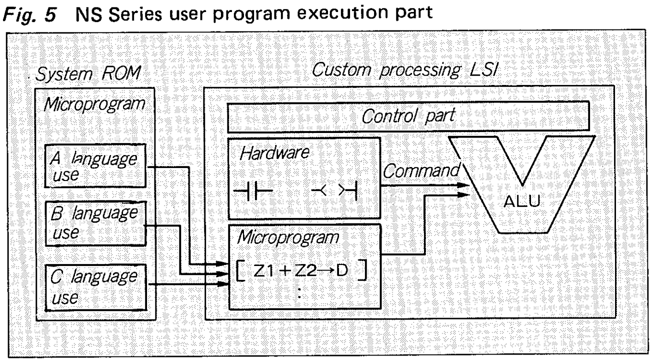


Fig. 5 NS Series user program execution part



With the LITE/N laptop type high-level loader, the programming tool for each hardware is obtained with the same hardware by replacing (simple replacement possible) only the system floppy disk and keyboard.

With the palm-size handy loader, the programming tool of each specification is obtained the same as LITE/N by changing only the internal system software ROM and the keyboard.

Both the high-level loader and handy loader have an additional function which offers an operating environment with which the user is familiar and improvement of ease of use as an original Fuji Electric tool.

2.3.2 Implementation by PC

The main factor of familiarity of users with PC seems to be with language (software), not with input/output form, etc. (hardware). Therefore, with the FLEX-PC N Series, the basic hardware is standardized and is efficiently applied to multi-language only with the hardware, the core part of program execution.

The problem in implementing a PC of each specification is how to implement the program execution operation of different languages.

This article uses the NS Series, the top of the line model of the FLEX-PC N Series, as an example.

PC instructions are roughly divided into two types as shown in *Fig. 3*.

(1) Sequence instruction:

Instructions in which operation is the same, such as an instruction that reads input on/off. Only the representation is different. (See *Fig. 4*.)

(2) Data instruction:

Instructions in which operation and representation are different, such as arithmetic operation.

With the NS Series, user program operation is executed at high-speed with our original custom LSI. For sequence instructions, the operation program is incorporated in the circuit of the custom LSI beforehand and for data instructions, the operation program is microprogrammed and is transferred to the custom LSI from the system ROM as shown in *Fig. 5*.

For sequence instruction, this implements multi-language by representation on the programming tool.

For data instructions, a data instruction operation microprogram for each language is provided in the system ROM in advance and is implemented by transferring a programming tool the same as the programming tool connected thereafter to the memory area (RAM) in the LSI. This transfer is performed when all the programs were deleted from the programming tool.

3. USER MACRO FUNCTION

3.1 Development aim

The FLEX-PC N Series user macro function is described beginning from paragraph 3.2. Its development aims were:

(1) Meet the following needs which were difficult to meet with the conventional general-purpose PC in the machine control equipment field:

(a) Application to high-speed processing and other individual machine specifications

(b) Meet the need for displaying independence and for discrimination as a man-machine interface and machine manufacturer

An I/O free location & selection function meets the need for an independent I/O configuration and type, which are different with each machine.

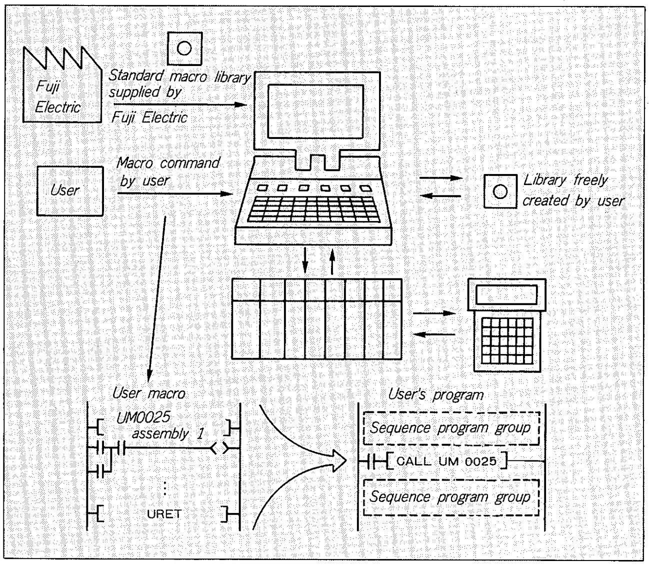
(2) Recent PC tend to have a large number of instructions which, with some PC, can exceed 300. As a result, the following problems are solved:

(a) Difference in how programming is carried out and how data instructions are used due to personal differences

(b) Program complexity and the difficulty of the service engineer to cope with trouble when it occurs

(3) An efficient programming function which allows easy reuse of programs can be realized

Fig. 6 User macro concept



3.2 Overview

User macros can be divided by execution format into the following two kinds:

(1) User macro by PC instruction combination
  (a) Development by user possible
  (b) Since operations are executed by custom processing LSI, as introduced in the preceding section, processing is fast.

(2) User macro by internal CPU microcomputer assembly program
  (a) No processing at PC instructions is possible.
  (b) User macros supplied by Fuji Electric as standard and user macros supplied on user request
  (c) User can develop macros faster and cheaper than developing a custom microcomputer.

The user macro concept is illustrated in *Fig. 6*. Once the necessary control program is created as a set of PC instructions and its name is registered, the user can handle it like a user custom data instruction thereafter. This offers the user the following advantages:

(1) Programs can be standardized.
(2) By macroizing debugged programs, subsequent debugging can be eliminated and debugging time can be substantially reduced.
(3) Program standardization also makes maintenance easy.

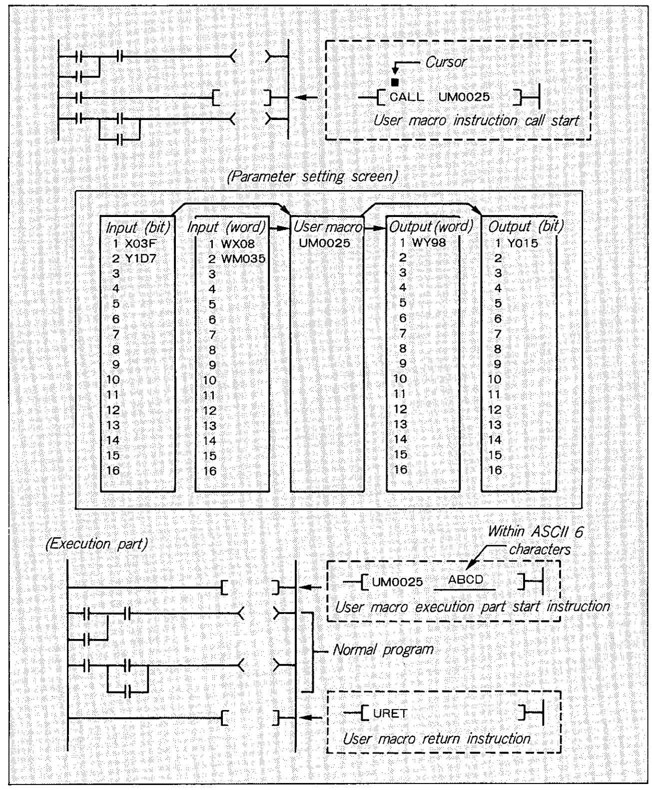
3.3 Handling by programming tool

The biggest feature of user macros is the ease with which they are handled by the LITE/N. This section introduces these.

3.3.1 User macro calling part program

FLEX-PC N Series user macros are calling part and execution part programs. With the FLEX-PC N Series, the input parameters passed from the calling part to the execution part and the output parameters of the executed result passed from the execution part to the calling part after execution can be easily programmed. When the user writes the user macro CALL instruction (calling part), a window like that shown in *Fig. 7* is opened automatically and parameters can be set.

Fig. 7 User macro programming



3.3.2 User macro execution part program

The user macro execution part program is a program which is inserted between the user macro execution part start instruction and user macro execution part return instruction as shown in *Fig. 7.*

The name of the macro can be given in the user macro execution part start instruction.

3.3.3 Loading of user macro from floppy disk

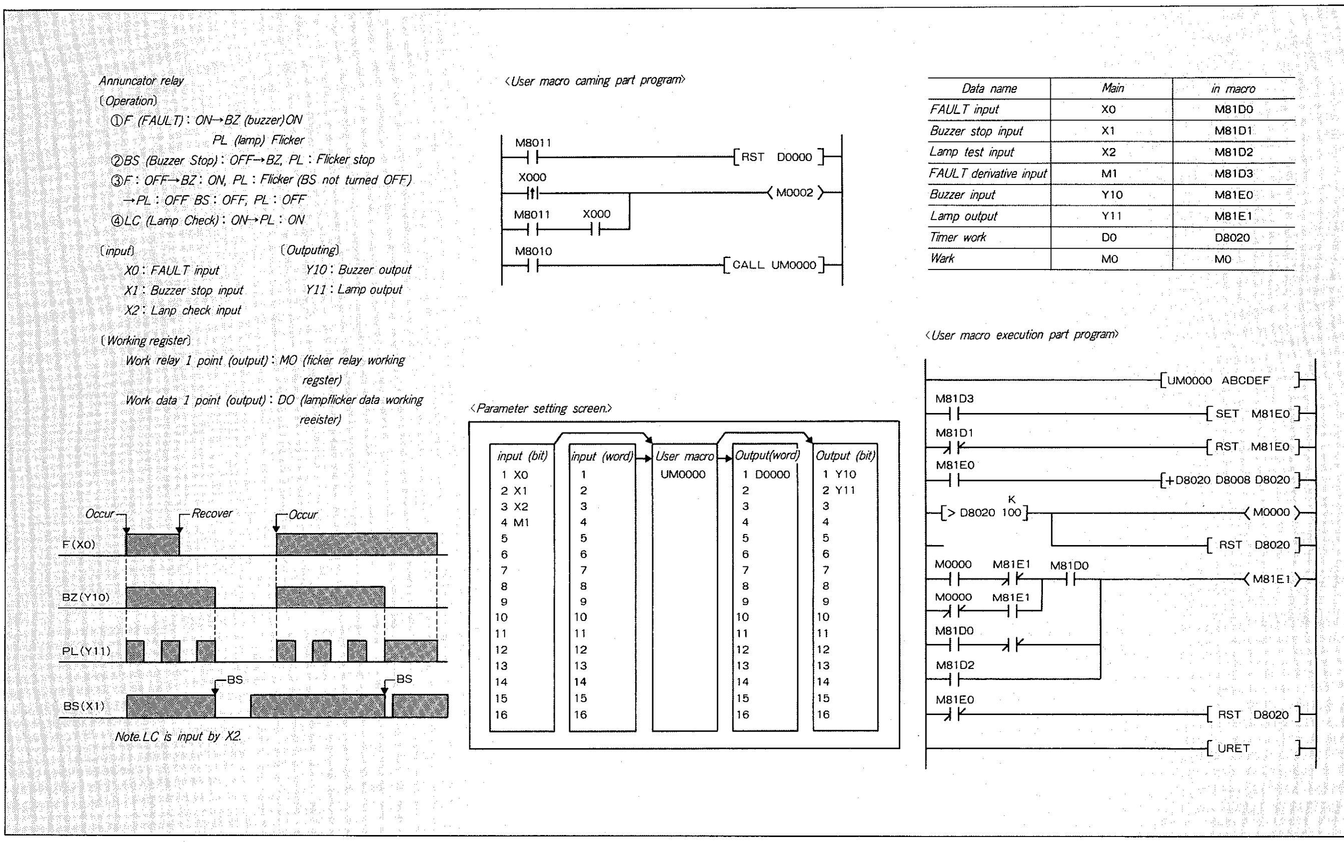
During normal programming, user macros created by the user can also be library edited to floppy disk, the same as the user macros supplied by Fuji Electric. The user macro CALL instruction is programmed and the execution part of the user macro used in the program is selected from the library and automatically loaded.

3.4 User macro program example

An annunciator relay program example is shown below as an example of the actual user program in *Fig. 8.*

An example which sets
FAULT input → X0
Buzzer stop input → X1
Lamp checking input → X2
FAULT differential input → M1
as input and
Buzzer output → Y10
Lamp output → Y11
Lamp flicker working register → D0

Fig. 8 User macro example



as output is shown in the figure. However, the user can easily use the annunciator function by simply setting the parameters matched to the PC system.

4. SEQUENTIAL FUNCTION CHART (SFC)

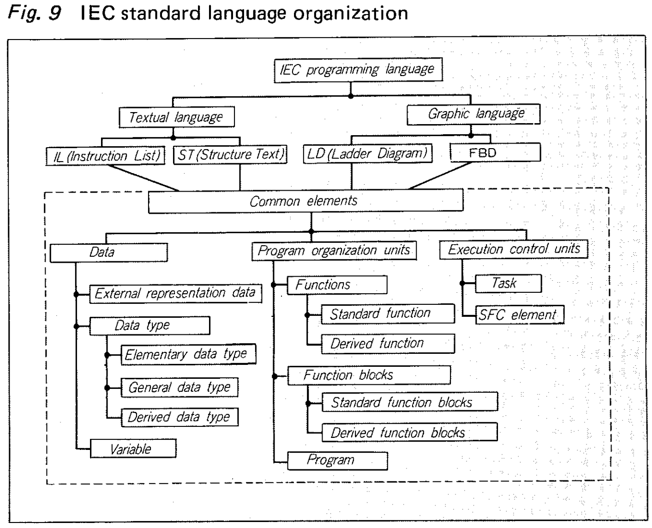
4.1 Overview

The existing ladder diagram and mnemonic language which approach the relay sequence have become widely popular in Japan as programmable controller (PC) programming languages. However, these languages are disadvantageous in that the control sequence and flow are difficult to understand. To overcome this disadvantage, from the beginning of development of the FUJILOG series, Fuji Electric created SC (Step Controller) instructions as an instruction language and made representation approaching actual control possible and received strong user support.

Recently, a new PC language element, SFC (Sequential Function Chart), has been attracting attention as a technique by which complex and advanced control contents can be represented more simply.

SFC was born in France and is GRAFCET, which has an enviable record of achievements in Europe, with improvements added to the nucleus. Advocated in 1977, GRAFCET is a graphic language used as a representation graph that applies the Petri Net concept which has been used since then in theoretical research, analysis, and simulation centered about researchers.

Fig. 9 IEC standard language organization



In 1977, the IEC (International Electrotechnical Commission) started standardization work encompassing PC control languages. In 1988, the working group (SC65A/WG6) summarized PC control languages, including SFC, as a standard plan. Today, JIS is proceeding with the establishment of a JIS standard based on this standard plan. In synchronization with this movement, Fuji Electric was rapidly to take up SFC and implemented it with the FLEX-PC. This is introduced here.

4.2 Features of SFC

SFC is defined as an execution sequence which associ-

Table 1 Characteristics of programming languages

Type	Language	Characteristic	
		Logic	Sequence
Textual type	Boolean algebra	○	
	IL (Instruction List)	○	
	ST (Structure Text)	○	
Graphic type	LD (Ladder Diagram)	◎	
	FBD (Function Block Diagram)	◎	
	Flow chart		◎
	GRAFCET, SFC element		◎
Table type	Decision table		◎

Fig. 10 Sequential control

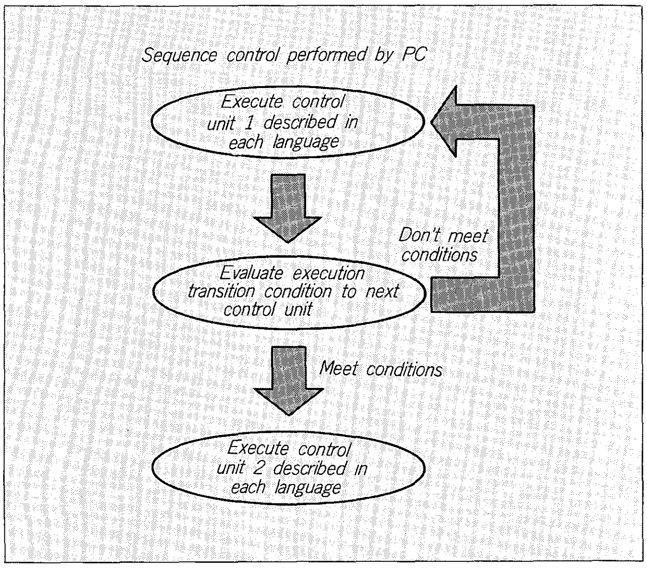
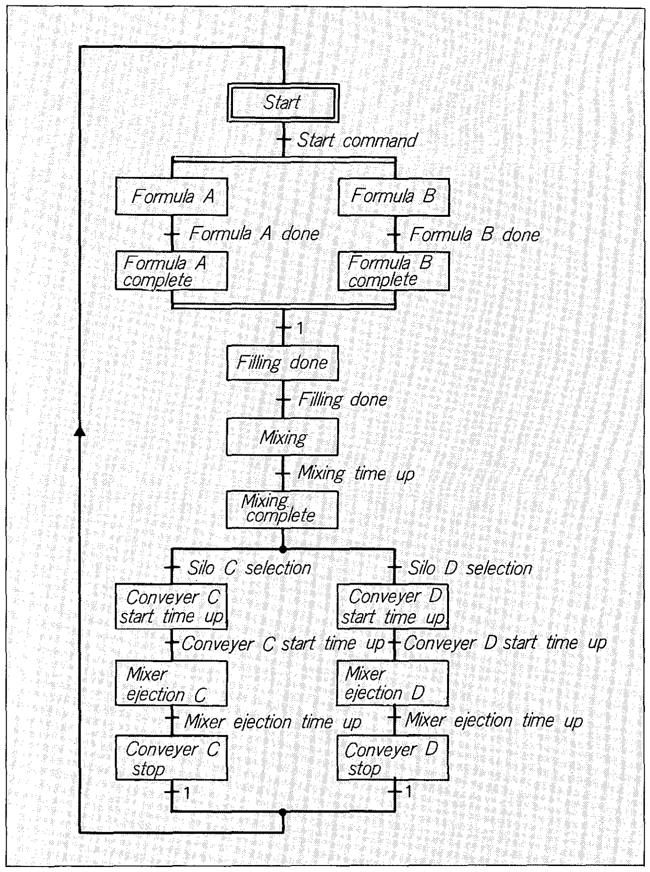


Fig. 11 Example of. description by SFC



ates simple partial processing described by ladder diagram and mnemonic in the previously mentioned IEC standard plan and common elements which can describe the execution conditions in graphic chart format (*Fig. 9*). It has a function which controls this (control step passing) (*Fig. 10*). Programs described by SFC can also be used in programs written in other languages and the characteristics of each extracted (*Table 1*).

Whereas a ladder diagram, etc. represents the signal combination logic by a circuit diagram, SFC represents the sequence of control by means of a chart. Therefore, control objective operation sequence and program correspondence is easy. This has such advantages as higher serviceability and easy understanding even by people who are not designers. In program development, it features easy design for construction by the top down method and application to software life cycle step work. An example of description by SFC is shown in *Fig. 11*.

With SFC, dividing a set of control contents into a number of steps and writing a program for each step is unnecessary. Therefore, there is no need to be aware of the execution timing of the entire program. This means that the independence of each step is high and a program can be handled like as a part in assembly work "parts-like program" and partial correction are easy.

5. SFC COMPONENT ELEMENTS

The SFC component elements are shown in *Fig. 12*.

5.1 Step

A step is one control unit. A step has two states: active state and inactive state. The state changes with establishment of the transition conditions before and after (above and below) the step. A unique operation output (action) is defined for each step. This action is taken only when the step is active. Steps which become active in the initial state of the sequence are called "initial steps" and are represented by a double line square.

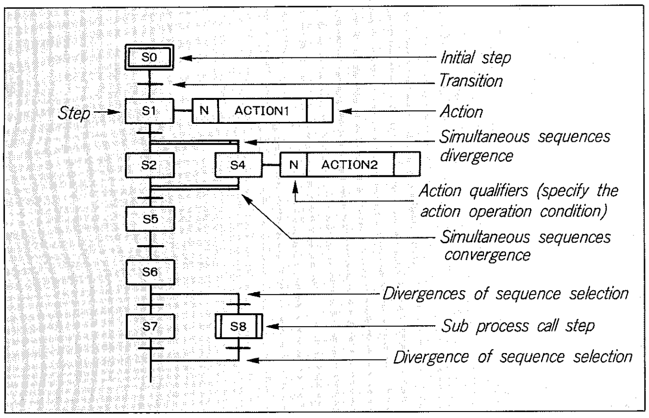
5.2 Transition

There is always one transition between steps. A transition is the condition for shifting from one step to the next step. When the preceding step is active, the transition condition is evaluated and when the condition is met, the active state shifts to the next step. The transition conditions are described by ladder diagram, mnemonic, etc.

5.3 Link

Link interconnects steps and shows the transition route. The direction is from top to bottom. Jump can be described by label.

Fig. 12 SFC component elements



5.4 Action

Action shows the operation output when the step became active. It is normally executed when the step is active. Action is described by ladder diagram, mnemonic, etc.

5.5 Basic operation

The basic operations of SFC are, first to activate the initial step and become the start of a set of sequences. Then the steps are passed in accordance with link and transition.

While a step is active, its action is normally executed and it is linked downward and its transition conditions are normally evaluated. A step remains active and does not move to the next step until this transition condition is met.

When the transition condition is met and the step linked to the link destination is inactive, the link destination step becomes active and the source step, which was active up to now, becomes inactive. At this time, the action of the link source step is reset. This series of operations appears to pass steps and eliminates the need for the interlock circuit that was necessary with the ladder diagram, etc.

With an SFC program, the flow of control is described as shown below.

(1) Divergence of sequence selection

Divergence of sequence selection evaluates all the transition conditions of the branch destination and selects the transition route which is met and proceeds with the pass. If multiple transitions are met at this time, the leftmost transition route is given priority and selected.

(2) Convergence of sequence selection

Convergence of sequence selection evaluates the transition condition directly below the step when the convergence source step is active and when the condition is met, passes to the convergence destination step.

(3) Simultaneous sequences divergence

Simultaneous sequence divergence evaluates the transition condition directly below the step when the divergence source step is active and when the condition is met, it activates all the divergence destination steps and passes them divergently.

(4) Simultaneous sequences convergence

Simultaneous sequences convergence does not evaluate the transition condition directly below the convergence destination step until all the convergence source (convergence wait) steps become active. Convergence passing is completed when the transition condition is met.

Table 2 General specifications of Fuji Electric SFC

Item	Specification (NS)
Processes	Max 256 (including sub processes)
Sub processes	Max 255
Steps	Max 1024
Transitions	Max 1024
Operation range/process	Within 256 steps and 256 transitions
Total divergences	Max 24
Nestings	Max 16 levels
Capacity/step	1 action
Capacity/action	Max 512 steps*
Action qualifier	N/S/R/L/D/P/SL/DS/SD (9 kinds)
Transition capacity	1 circuit (1 transition relay) and within 512 steps
User macro	Usable within an action
Concurrently executable processes	Max 256
Comments	Kanji comment: 14 characters, katakana symbol: 8 characters

< Note > *: Instruction step

6. SPECIFICATIONS AND FEATURES OF FUJI ELECTRIC SFC

6.1 Specifications

The general specifications of the Fuji Electric SFC are shown in *Table 2.*

6.1.1 Program configuration

With Fuji Electric SFC, one unit of the SFC program is called a "process" (*Fig. 5*). Up to 256 processes can be created. These 256 processes can be main processes or sub processes and have a tree structure (*Fig. 6*). Therefore, a program can be divided into function units by process so that system construction is structurally easy.

6.1.2 Operation

When the PC is started, the initial step of all the main programs is activated. Therefore, parallel processing in process units is possible. When the end of the process is an initial step or an ordianry step, the pass stops at the end step and execution of that action continues until the power is turned off.

This way of ending a step is effective at normal monitoring processes (steps), etc.

Parameter specification also makes it possible to restart a process from the state immediately before the previous stop. In one process, multiple steps can be activated at other than parallel processing and active steps can be overtaken and the same sub process can be called simultaneously.

Fig. 13 Process expression

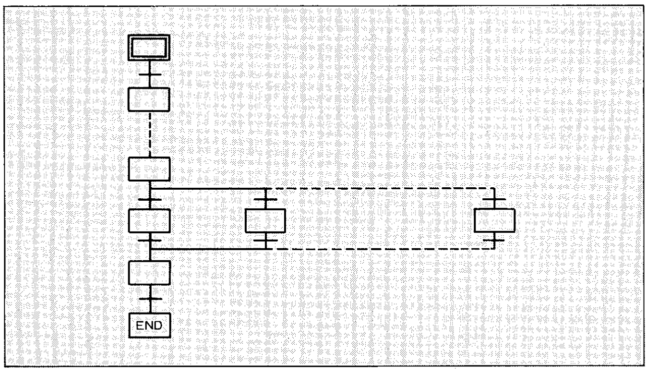


Fig. 14 Tree structure

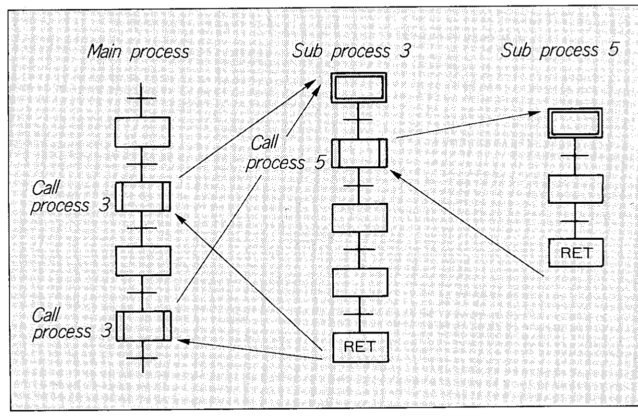


Table 3 Fuji Electric SFC representation elements

Symbol	Name	Function
S * * *	Initial step	One initial step is always necessary in a process. Action definition similar to a step is possible. (***: 000 to 3FF (H) 1023)
S * * *	Step	When a step becomes active, the action in that step is executed. Registered action: 1 (***: 000 to 3FF (H) 1023)
END S * * *	Process end step	Dummy step which signifies the conclusion of control (****: 000 to 3FF (H) 1023)
S***/PR**	Process call step	Dummy step that calls a sub process. (***: 000 to 3FF (H) 1023) Call process No. display: Switched (**: 00 to FF (H) 255)
RET S * * *	Process return step	A process return step is always necessary in a sub process. It is a dummy step that signifies the end of the sub process. (***: 000 to 3FF (H) 1023)
TN * * *	Transition	Step to step transition condition. Transition program: 1 circuit (1 relay) (***: 000 to 3FF (H) 1023)
	Divergence/Convergence of sequence selection	Executes one route from among the diverged routes
	Simultaneous sequence divergence/Convergence	Executes all divergence destinations simultaneously
S * * *	Jump	Connection start side in process. (***: 000 to 3FF (H) 1023)
S * * *	Joint	Connection receiving side in process (automatic generation)
	Link	Connection line

6.1.3 Representation

The following element representation has been added to the Fuji Electric SFC (*Table 3*), in addition to SFC element representation defined by the IEC standard plan.

(1) Sub process call step to permit a tree structure
(2) Process return step to return the sub process to the calling process
(3) Process end step which ends that process
(4) Jump/joint which perform looping and jump

The action description recommended by IEC should be connected to the right side of a step, but since the loader display is difficult to read, a display that shows the existence of an action at the right side of the step was added (*Fig. 7*).

6.1.4 Mixing with ladder circuit

As previously mentioned, SFC is superior from the standpoint of describing the control sequence, but does have one disadvantage in that describing control (manual operation system) of emergency stop and other interrupt processing and processing unrelated to the sequence, that is, logic combination only, is difficult. Therefore, with FLEX-PC, SFC can be mixed with the ladder circuit that is normally executed. The internal program configuration is shown in *Fig. 8*.

6.2 Functions of loader (peripheral device)

The loader function, which offers an SFC support software package for the FLEX-PC, has the following features.

6.2.1 Programming

Program registration, copying, and movement can be performed in process units. This made it possible to "parts-

Fig. 15 Existence of action

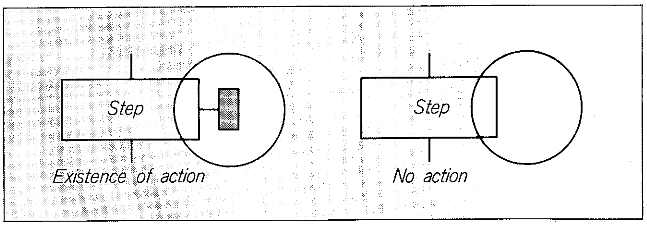
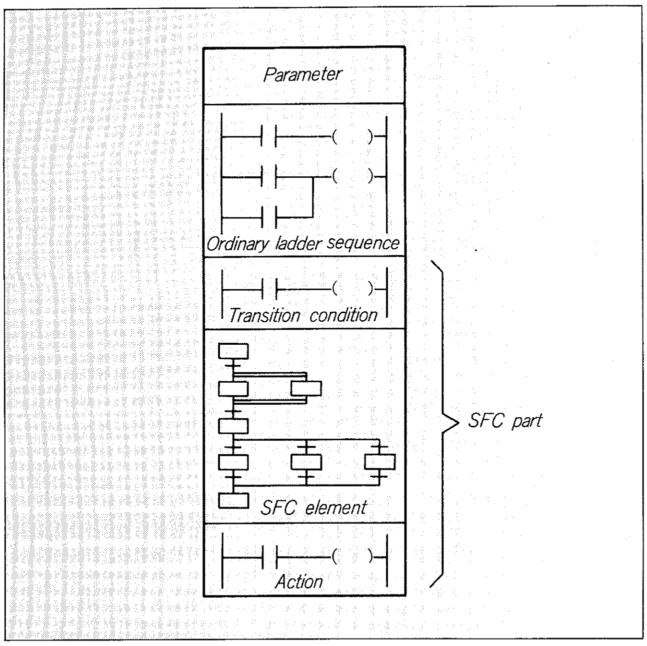


Fig. 16 Program configuration in PC



like processes". SFC program editing calls each screen by zooming so that it is performed top down in process, action, transition order and makes the user interface easy to use by means of an automatic correction intelligent function for incorrect programming during process editing and display point tracking type display switching, high-speed scrolling, etc. during process display.

To make the program easy to understand, Japanese language comments can be added for processes, steps, and transitions and symbols can be added for steps and Japanese language statements can be added in circuit units for actions.

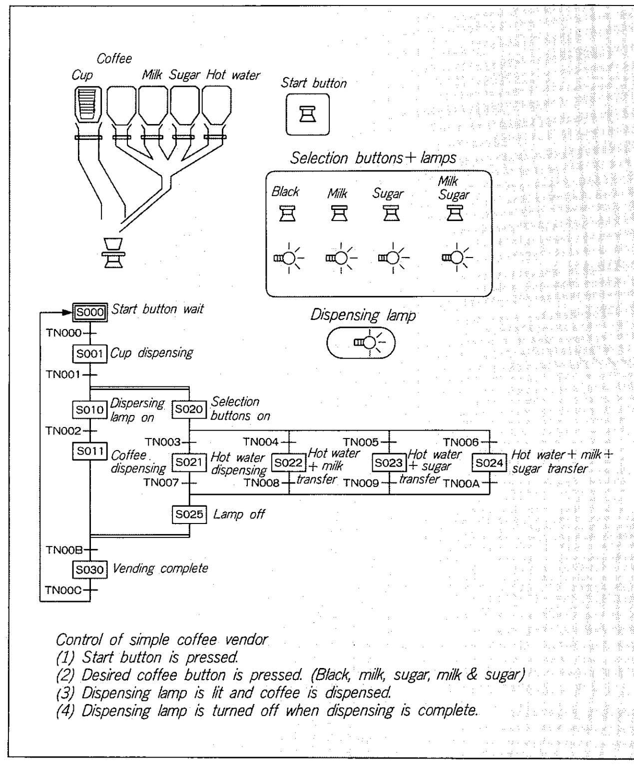
Since device search on the process display can be followed without deleting the search information from the screen, the search information does not have to be stored. This is effective especially at output coil contact usage state search.

Since a function that displays guidance for operand input when entering data instructions used in actions and transitions is provided, operability is good. A function that jumps the SFC program in textual form is also provided and is effective when trouble occurs.

6.2.2 Debugging function

One step execution, steps that do not exist in the process, transition mixed type device monitor and sampling trace, status latch, and a history function that allows

Fig. 17 SFC application example



thorough checking of the step passing situation after program execution make SFC debugging easy.

6.2.3 System

To implement these functions quickly, the SFC information data can be stored in the loader. The restrictions, which change with the model of FLEX-PC, are contained in software packages so that they can be dealt with flexibly.

7. ACTUAL APPLICATION EXAMPLE

An actual SFC application example is shown in *Fig. 9.*

8. CONCLUSION

The first half of this article introduced the multi language function and user macro function as flexible functions of the FLEX-PC N Series from the standpoint of software. These two functions are the industry's first attempt and pursue ease of use by the user. Further enhancement of the contents of these functions is planned. The last half of this article introduced SFC as a new PC language element. The advantages of SFC are noticeable in the specification, design, debugging, and maintenance processes. Especially, specification of the trouble point during troubleshooting is very simple compared to a ladder diagram. Moreover, because the program flow is read visually and is easy to understand and programming with few individual programmer errors is possible, SFC is expected to become popular in the future.

Expansion of the debugging and other functions and enhancement of trouble diagnosis and other RAS functions of the Fuji Electric SFC is planned and improvement of the user interface is being pursued further.