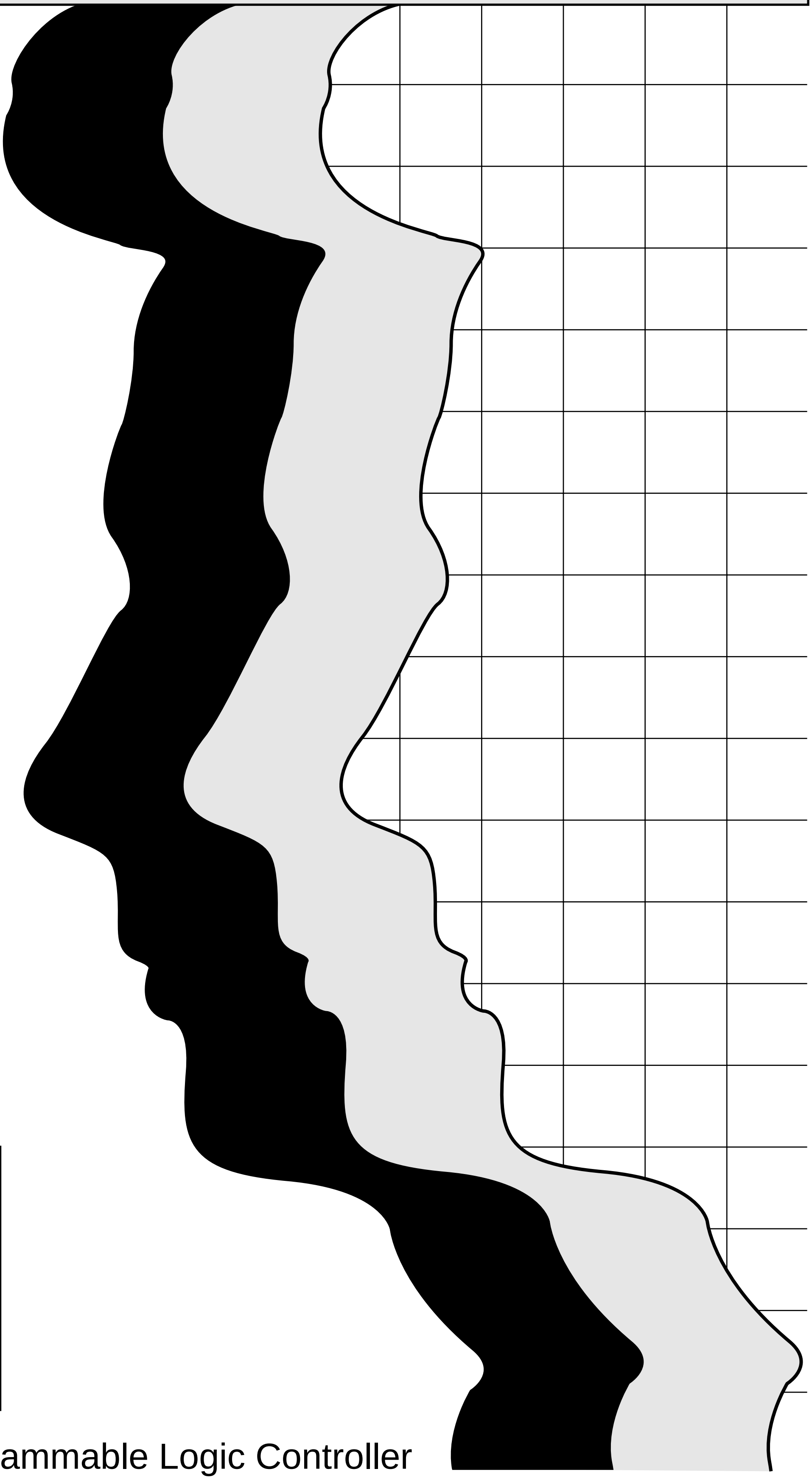
MITSUBISHI

type ACPU/QCPU-A (A Mode)(Fundamentals)

Programming Manual



Mitsubishi Programmable Logic Controller

SAFETY CAUTIONS

(You must read these cautions before using the product)

In connection with the use of this product, in addition to carefully reading both this manual and the related manuals indicated in this manual, it is also essential to pay due attention to safety and handle the product correctly.
The safety cautions given here apply to this product in isolation. For information on the safety of the PC system as a whole, refer to the CPU module User's Manual.
Store this manual carefully in a place where it is accessible for reference whenever necessary, and forward a copy of the manual to the end user.

REVISIONS

*The manual number is given on the bottom left of the back cover.

Print Date	*Manual Number	Revision
Feb., 1991	IB (NA) 66249-A	First edition
Sep., 1993	IB (NA) 66249-B	Addition of models A1SCPU, A2UCPU(S1), A3UCPU, A4UCPU, A2ACPU(S1)-F, A3ACPU-F Correction CONTENTS, Section 1, 1.1, 2.1, 2.3.1, 2.4, 2.4.2, 2.4.3, 2.5, 2.6, 2.6.1, to 2.6.3, 3.1 to 3.6, 3.6.2, 3.6.3, 3.7, 3.7.1 to 3.7.4, 3.8, 3.8.2, 3.9 to 3.11, 3.11.2, 3.12, 3.13, 3.13.1, 3.13.2, 3.14 to 3.17, 4, 4.2, 4.2.2, 4.2.3, 4.3, 4.3.2, 4.4, 4.4.1, 5, 5.1, 5.1.1 to 5.1.3, 5.2, 5.2.1, 5.3, 5.3.1, 5.3.2, 5.3.4, 6, 6.1 to 6.8, 6.8.2, 6.8.3, 6.9 to 6.13, 6.13.1, 6.14 to 6.17, 7, 8 Addition Section 5.2.3, 6.18, 9 to 11 Deletion For BEGINNERS
Sep., 1990	IB (NA) 66249-C	Correction Section 3.11.2, 3.13.2, 4.2.2
Dec., 1990	IB (NA) 66249-D	Chapter 9 added
May., 1991	IB (NA) 66249-E	Correction Section 3.6.3, 3.7.4, 3.13.1, 3.13.2, 4.4.1 A1SCPU added
Feb., 1992	IB (NA) 66249-F	Correction Section 3.16, 9.1, 9.6
Apr., 1992	IB (NA) 66249-G	Correction Section 3.5, 5.3.4, Chapter 8, Section 9.6
Aug., 1992	IB (NA) 66249-H	Correction Section 3.3
May., 1993	IB (NA) 66249-I	Correction A2UCPU(S1), A3UCPU, A4UCPU, A52GCPU, A373CPU added Section 6.18, 9.7, 9.8, Chapter 10, 11 added
Aug., 1994	IB (NA) 66249-J	Correction Section 2.4.1, 2.4.2, Chapter 11
May., 1997	IB (NA) 66249-K	Correction Section 3.1, 3.3, 3.4, 3.5, 3.6.2, 3.10, 3.11.2, 5.3.4, 10.4.1, IMPORTANT Addition Section 3.11.1, 5.3.4, 6.1, Chapter 11

This manual does not imply guarantee or implementation right for industrial ownership or implementation of other rights. Mitsubishi Electric Corporation is not responsible for industrial ownership problems caused by use of the contents of this manual.

Print Date	*Manual Number	Revision
Aug., 1998	IB (NA) 66249-L	Addition of models A1SJCPU-S3, A1SCPUC24-R2, A2SCPU(S1), A1SHCPU, A1SJHCPU, A2SHCPU(S1), A2ASCPU(S1), A1FXCPU Addition Section 4.5, Appendix 1 Correction CONTENTS, Section 1.1, 3.1, 4.3.1, Chapter 5, Section 6.9, 6.10, 6.11, 9.1, 9.3 to 9.12, 10.3 Deletion A2NCPU(P21/R21)-F, A2NCPU(P21/R21)-S1-F, A3NCPU(P21/R21)-F, A3N Board
Feb., 2000	IB (NA) 66249-M	The manual name is changed into ACPU/QCPU-A(A Mode) Programming Manual (Fundamentals). [Old name: ACPU Programming Manual (Fundamentals).] Addition of models Q02CPU-A, Q02HCPU-A, Q06HCPU-A, A2USHCPU-S1, A1SJHCPU-S8
Dec., 2000	IB (NA) 66249-N	Correction Section 3.1, 3.6.2, Chapter 8

This manual does not imply guarantee or implementation right for industrial ownership or implementation of other rights. Mitsubishi Electric Corporation is not responsible for industrial ownership problems caused by use of the contents of this manual.

CONTENTS

1. GENERAL DESCRIPTION

This manual describes the devices and the allocation procedures of input and output numbers which are required for creating programs for the MELSEC-A series programmable controllers.

For the operations which pertain to the A6GPP, refer to the A-series basic course (for SW4GP-GPPAEE) (IB-66412) in the A-SERIES TRAINING MANUAL.

1.1 CPU Types and Their Abbreviations Used in this Manual

The CPU types and their abbreviations used in this manual are as given in the table below.

Table 1.1 CPU Types and Their Abbreviations Used in this Manual

Abbreviation		CPU Type
An	A1	A1CPU(P21/R21)
	A2(-S1)	A2CPU(P21/R21), A2CPU(P21/R21)-S1
	A3	A3CPU(P21/R21)
AnN	A1N	A1NCPU(P21/R21)
	A2N(-S1)	A2NCPU(P21/R21), A2NCPU(P21/R21)-S1
	A3N	A3NCPU(P21/R21)
A3H		A3HCPU (P21/R21)
A3M		A3MCPU(P21/R21)
A3V		A3VCPU(P21/R21)
AnA	A2A(-S1)	A2ACPU(P21/R21), A2ACPU(P21/R21)-S1
	A3A	A3ACPU(P21/R21)
AnS	A1S	A1SCPU(S1), A1SCPUC24-R2, A1SJCPU, A1SJCPU-S3
	A2S	A2SCPU(S1)
AnSH	A1SH	A1SHCPU, A1SJHCPU(S8)
	A2SH	A2SHCPU(S1)
A0J2H		A0J2HCPU(P21/R21)
A2C		A2CCPU(P21/R21), A2CCPUDC24, A2CCPUC24(-PRF), A2CJCPU(S3)
A73		A73CPU(P21/R21)
A52G		A52GCPU(T21B)
AnU	A2U(-S1)	A2UCPU, A2UCPU-S1
	A3U	A3UCPU
	A4U	A4UCPU
A2AS	A2AS(-S1)	A2ASCPU, A2ASCPU-S1, A2ASCPU-S30
	A2USH-S1	A2USHCPU-S1
QCPU-A	Q02	Q02CPU-A
	Q02H	Q02HCPU-A
	Q06H	Q06HCPU-A
A1FX		A1FXCPU

POINT

This manual does not apply to the A0J2CPU(P23/R23).
For the instructions used with the A0J2CPU(P23/R23), refer to the A0J2CPU Programming Manual (IB-66057).

2. PROGRAMMING LANGUAGES AND OPERATIONS

This chapter describes the programming languages and the method of expression of numeric values which are applied to the writing of sequence programs.

2.1 Programming Languages

Two different methods, a graphic method and a dedicated instruction method, can be used for the programming of a PC CPU.

- The graphic method uses the relay symbolic language.*1
- The dedicated instruction method uses the logic symbolic language.*2

These languages can be used to write a sequence program.

The SFC language*3 (MELSAP II) as well as the relay symbolic language and the logic symbolic language can be used for programming.

- For the SFC language (MELSAP II), refer to the MELSAP II Programming Manual (IB-66361).

REMARKS

*1: When the A6GPP/A6PHP/A6HGP is used, set it to the "ladder mode".

*2: When the A6GPP/A6PHP/A6HGP/A7PU is used, set it to the "list mode".

*3: For the PC CPU types which can execute an SFC language, refer to the MELSAP II Programming Manual (IB-66361) Version B and later.

2.1.1 Relay symbolic language (ladder mode)

The relay symbolic language is based on the concept of relay control ladder operation and allows programming using expressions which are similar to relay sequence ladders.

(1) Ladder block

A ladder block is the smallest unit of sequence program operation. Each block begins at the bus on the left side and ends at the bus on the right side.

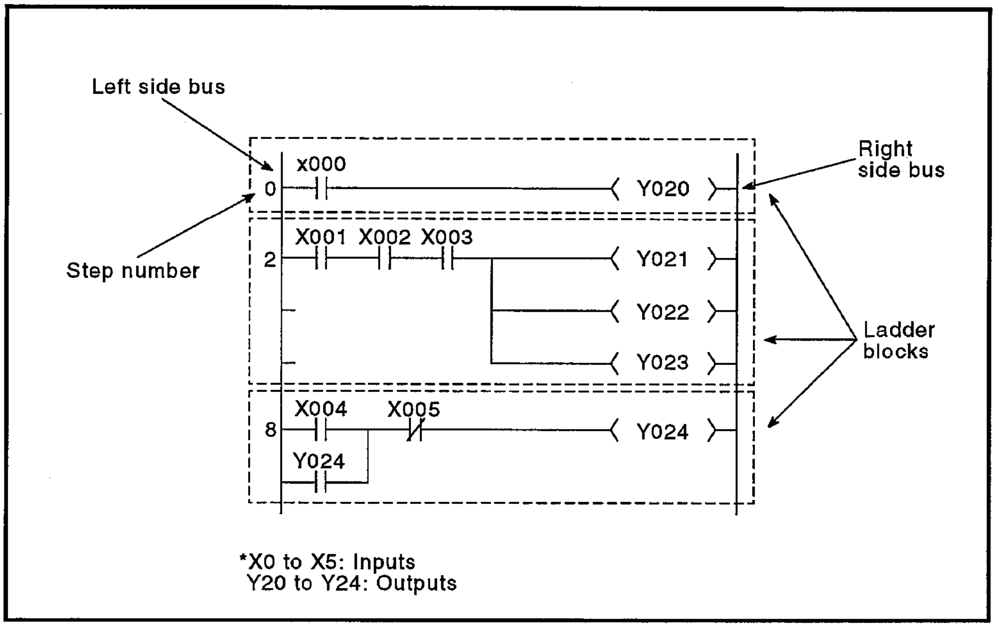
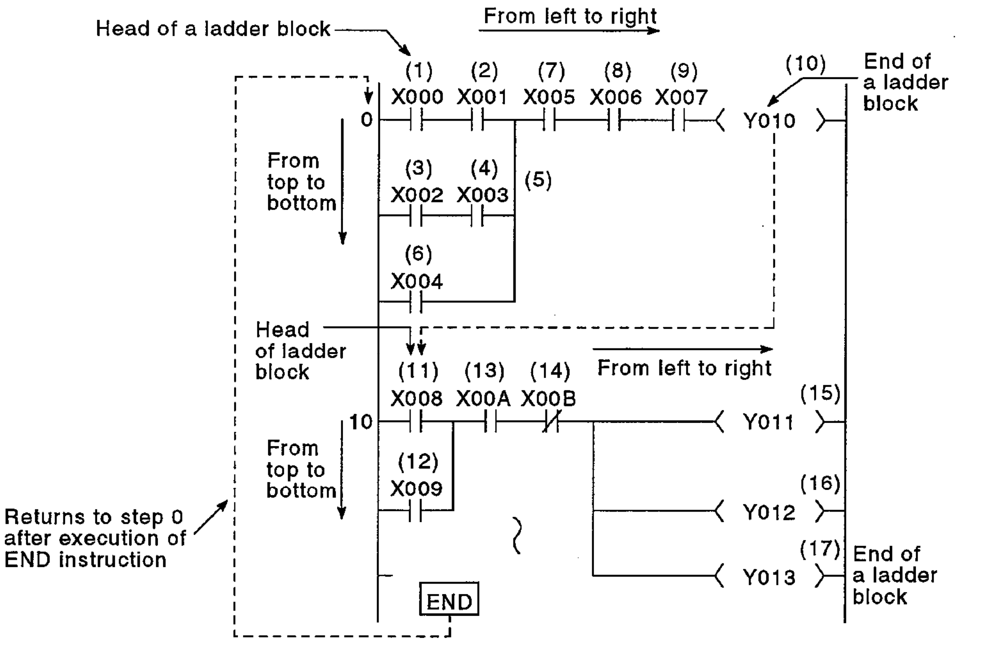


Fig. 2.1 Ladder Blocks

(2) Operation processing

Sequence program operation is executed repeatedly according to ladder blocks beginning with step 0 and finishing with the END instruction.
In each ladder block, operation begins with the left bus and proceeds to the right and from top to bottom.



*Numbers 1 to 17 indicate the order of operation.
END is not displayed on peripheral devices.

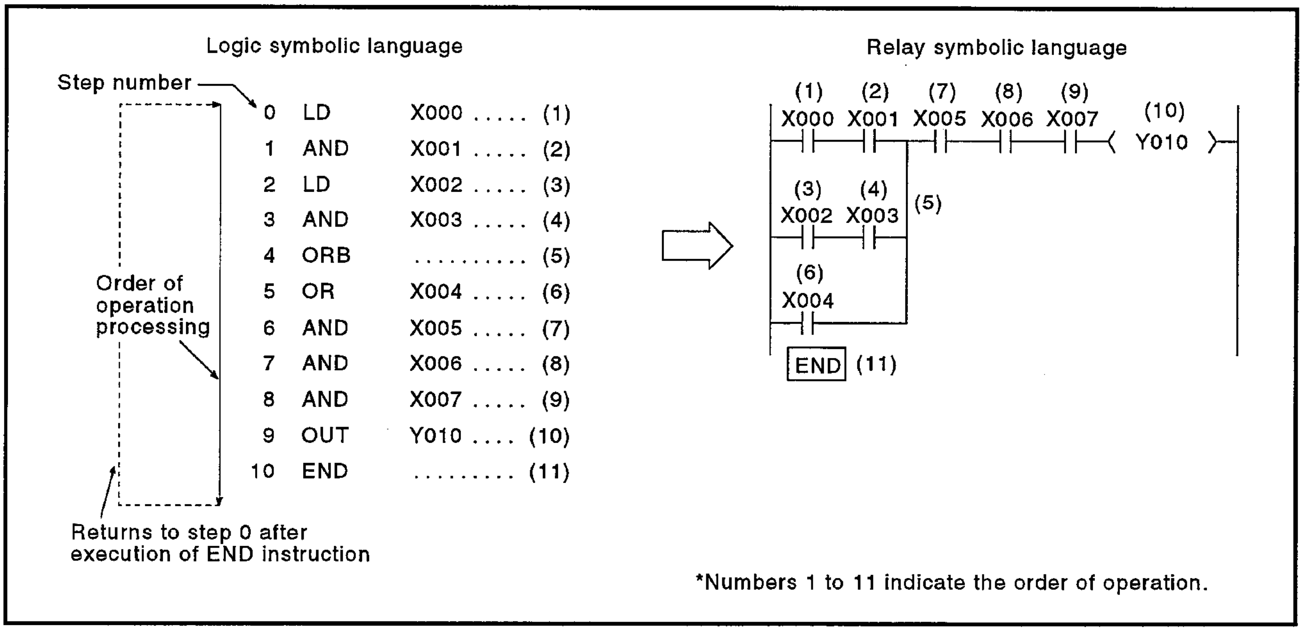
Fig. 2.2 Order of Operation Processing

2.1.2 Logic symbolic language (list mode)

The logic symbolic language uses dedicated instructions for writing contacts and coils (which are written in the relay symbolic language using symbols).

(1) Operation processing

Sequence program operation is executed repeatedly beginning with step 0 and finishing with the END instruction. After the END instruction execution, processing is executed again beginning with step 0.



Logic symbolic language

```
 0  LD    X000 ..... (1)
 1  AND   X001 ..... (2)
 2  LD    X002 ..... (3)
 3  AND   X003 ..... (4)
 4  ORB   .......... (5)
 5  OR    X004 ..... (6)
 6  AND   X005 ..... (7)
 7  AND   X006 ..... (8)
 8  AND   X007 ..... (9)
 9  OUT   Y010 .... (10)
10  END   ......... (11)
```

*Numbers 1 to 11 indicate the order of operation.

Fig. 2.3 Order of Operation Processing

2.2 Operation Processing of the PC CPU

The PC CPU employs a repetitive operation method using a stored program.

(1) Stored program

(a) The sequence program for operation processing is stored in the internal memory of the CPU.

(b) The CPU performs operation processing by reading the stored program by each instruction, and it controls status of devices according to the operation results.

(2) Repetitive operation method

The repetitive operation method repeats execution of a series of operations.
The PC CPU repeats the operation processing as shown below.

(a) The PC CPU executes the sequence program, stored in the internal memory, beginning with step 0 to the END instruction.

(b) After executing the END instruction, the PC CPU performs internal processings such as timer/counter update and self - diagnosis, and then, returns to step 0 of the sequence program.

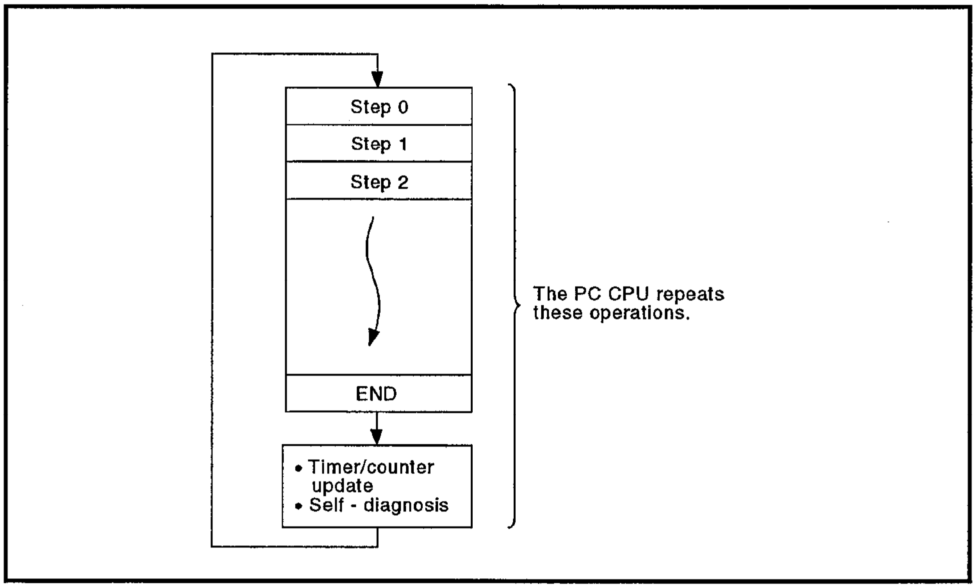


Fig. 2.4 Operation Processing of the PC CPU

REMARK

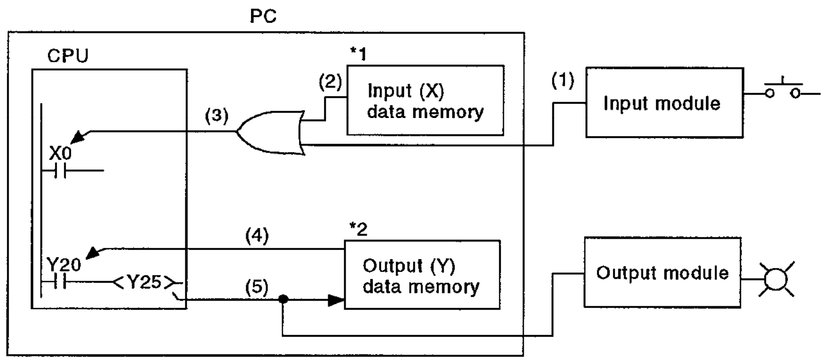
The series of steps from step 0 to the next step 0 or from an END instruction to the next END instruction is called a scan.
The scan time of the PC CPU is calculated as a total of the processing time of the sequence program (step 0 to END) and the internal processing time of the PC CPU.

2.3 Input and Output Processings

The input and output modules are controlled by either the direct mode or the refresh mode.
The control mode is determined by the type of CPU used.

2.3.1 Direct mode

The direct mode allows input signals to be received by the PC by each signal and used as input data.
The operation results are output by each result to the output data memory and output modules.



- When an input contact command is executed:
  The input data (1) of the input module and the input data (2) of the data memory are ORed. The operation result is used as the input data (3) for execution of the sequence program.
- When an output contact command is executed:
  The output data (4) is read from the data memory for execution of the sequence program.
- When an output OUT instruction is executed:
  The operation result (5) is output to the output module and stored in the output (Y) data memory.

Fig. 2.5 Flow of Input/Output Data in the Direct Mode

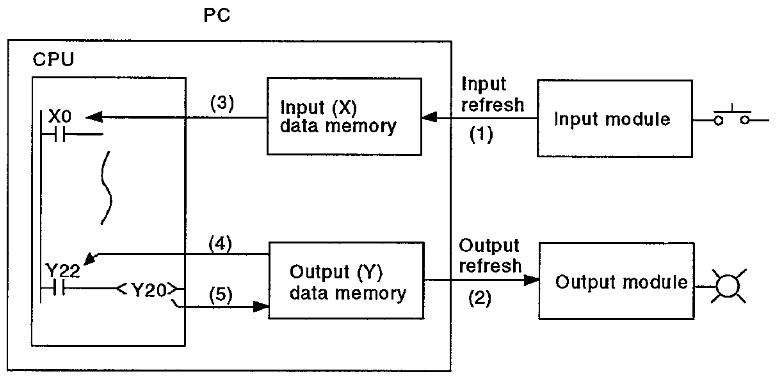
REMARKS

*1 The following operations turn ON/OFF the input (X) data memory.
Test operation with a peripheral device
Link refresh of the MELSECNET
Writing data from a computer link module

*2 The following operations turn ON/OFF the output (Y) data memory.
Execution of the OUT instruction of the sequence program
Test operation with a peripheral device
Writing data from a computer link module

2.3.2 Refresh mode

The refresh mode allows input signals to be stored in batch in the input data memory before execution of each scan.
The data in the input data memory is used for execution of the sequence program operation.
The operation results (Y) are output by each result to the output data memory. The data in the output data memory is output in batch to the output modules after execution of the END instruction.



- Input refresh
  Input data is read (1) in batch from the input module before execution of step 0 and stored in the input (X) data memory.
- Output refresh
  Data (2) in the output (Y) data memory is output in batch to the output module before execution of step 0.
- When an input contact command is executed:
  Input data is read (3) from the input (X) data memory and used for execution of the sequence program.
- When an output contact command is executed:
  Output data is read (4) from the output (Y) data memory and used for execution of the sequence program.
- When an output OUT instruction is executed:
  The operation result (5) is stored in the output (Y) data memory.

Fig. 2.6 Flow of Input/Output Data in the Refresh Mode

POINT

To access a part of input/output module memory area as in the direct mode, use the SEG (partial refresh) instruction. Refer to the ACPU Programming Manual (Common Instructions) (IB-66250) for details.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	o	x	x	x	x	o	x	o
Remark											

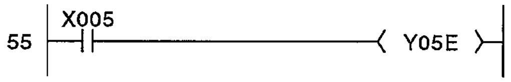
2.4 Response Lag

This section describes the lag between a change in the input module and its resultant change in the output module in the cases of direct mode and refresh mode.

2.4.1 In the direct mode

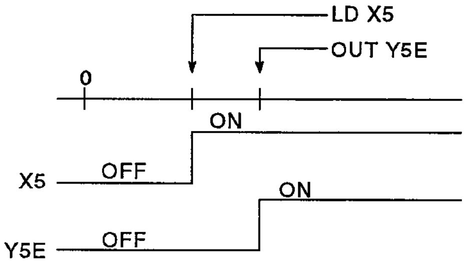
An output module change lags max. 1 scan behind the corresponding input module change as shown in Fig. 2.7.

Ladder example



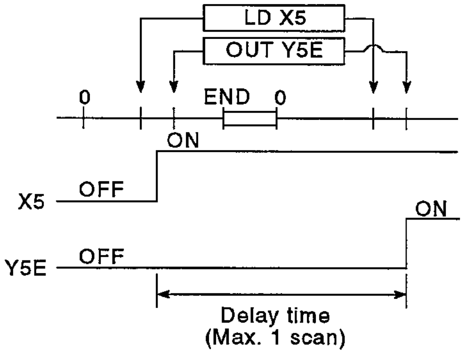
Output Y5E is switched on by input X5 switched on.

Earliest Y5E is switched on



Input X5 is switched on immediately before the operation of step 55 is executed. Y5E is switched on during the scan in which X5 has been switched on. The delay time is therefore minimal.

Latest Y5E is switched on



Input X5 is switched on after the operation of step 55 is executed. Y5E is switched on during the next scan. The delay time is therefore 1 scan max.

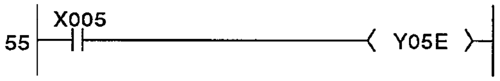
Fig. 2.7 Output Y Change to Corresponding Input X Change

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	x	o	o	o	o	o	o	o	o	o
Remark											

2.4.2 In the refresh mode

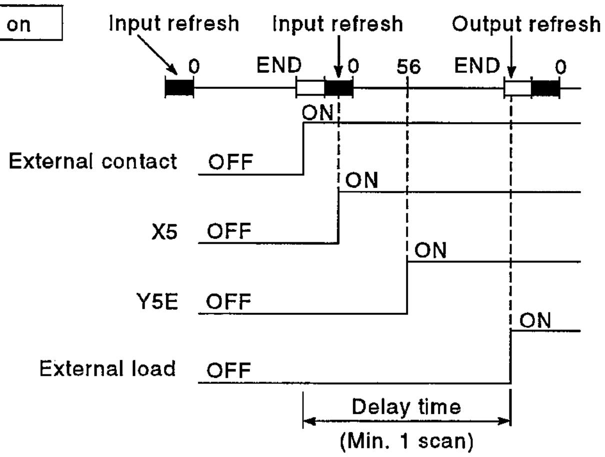
An output module change lags max. 2 scans behind the corresponding input module change as shown in Fig. 2.8.

Ladder example



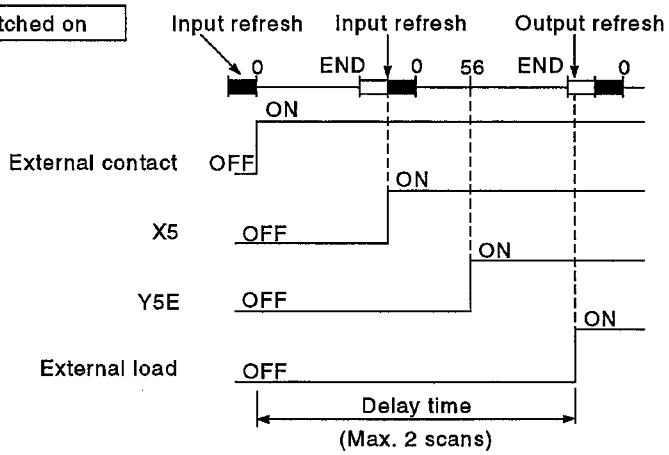
Output Y5E is switched on by input X5 switched on.

Earliest Y5E is switched on



The external contacts close immediately before the input refresh is made. X5 is switched on when the input refresh is mode. Y5E is switched on when the operation of step 56 is executed. The external load is switched on when the output refresh is mode after the [END] instruction is executed. The delay time (external contact change to external load change) is therefore 1 scan.

Latest Y5E is switched on



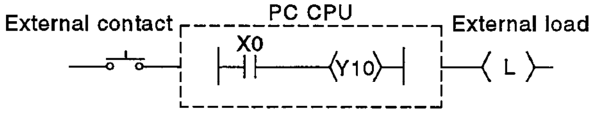
The external contacts close immediately after the input refresh is made. X5 is switched on when the next input refresh is mode. Y5E is switched on when the operation of step 56 is executed. The external load is switched on when the output refresh is mode after the [END] instruction is executed. The delay time (external contact change to external load change) is therefore 2 scans.

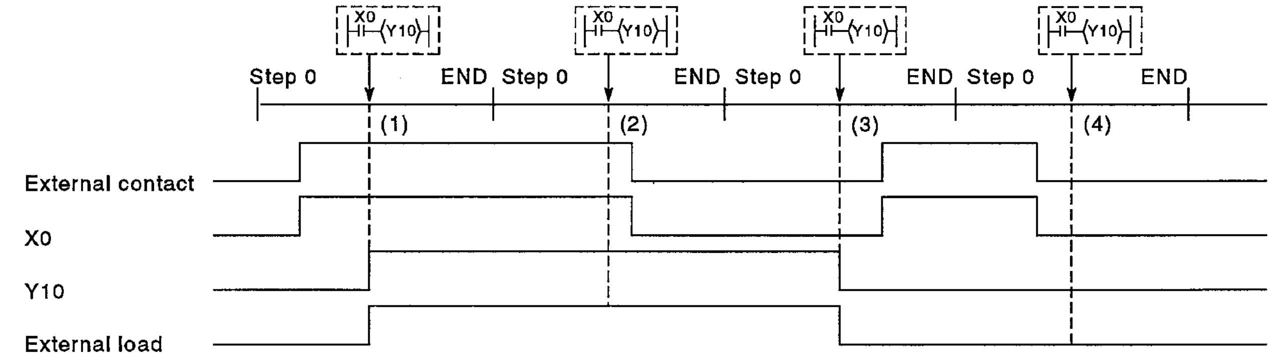
Fig. 2.8 Output Y Change to Corresponding Input X Change

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	x	o	o	x	x	x	x	o	x	o
Remark											

2.4.3 Response when direct and refresh modes are switched

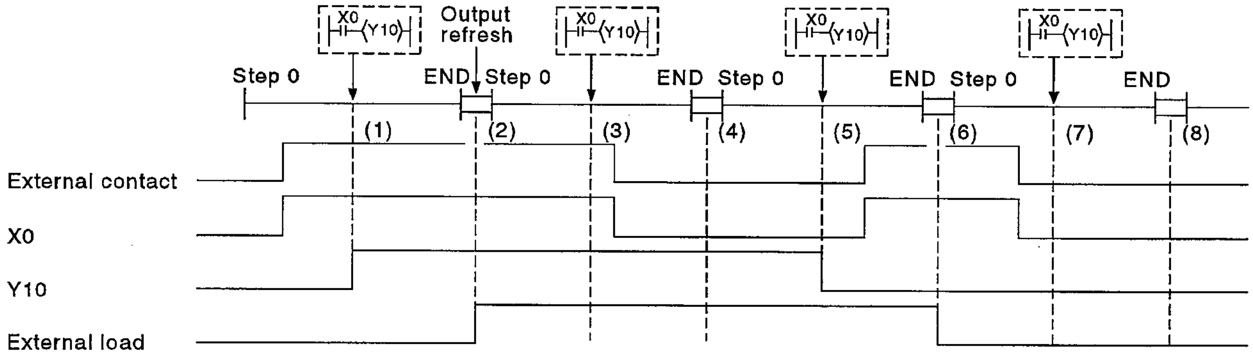
Some types of PC CPUs can handle input data in direct mode and output data in refresh mode, or vice versa.
This section describes the flow of ON/OFF data from the input module to the output module when the I/O control mode is switched from direct mode to refresh mode, or vice versa.
Figures 2.9 to 2.12 show ON/OFF timing of inputs and outputs when the program shown below is executed.





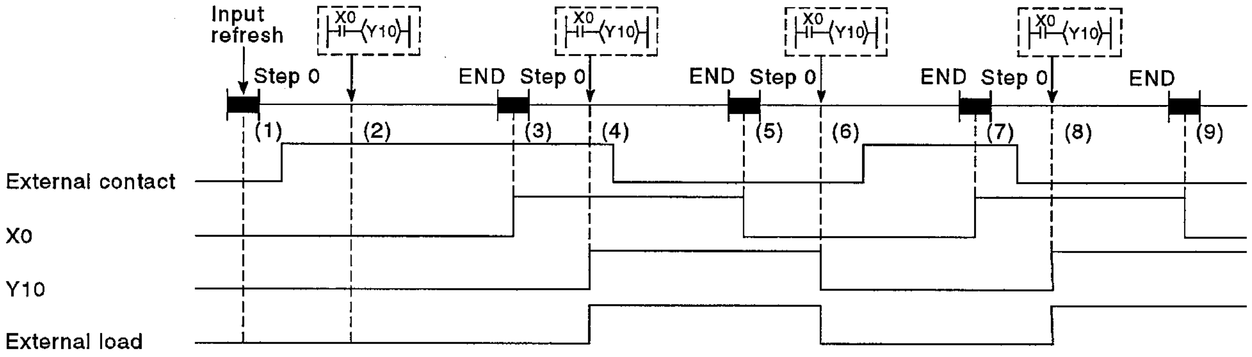
(1) : Since X0 is turned from OFF to ON, Y10 is turned ON.
(2) : Since X0 remains ON, Y10 remains ON.
(3) : Since X0 is turned OFF, Y10 is turned OFF.
(4) : Since X0 remains OFF, Y10 remains OFF.

Fig. 2.9 When Both Input and Output are in Direct Mode



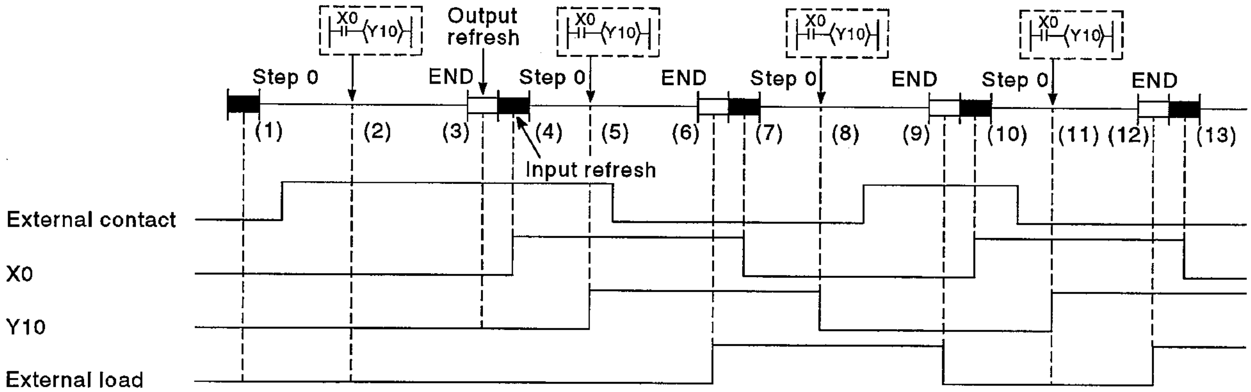
(1) : Since X0 is turned from OFF to ON, Y10 is turned ON.
(2) : The Y10 ON data is output to external load in refresh mode.
(3) : Since X0 remains ON, Y10 remains ON.
(4) : Since Y10 remains ON , external load remains ON.
(5) : Since X0 is turned OFF, Y10 is turned OFF.
(6) : The Y10 OFF data is output to external load in refresh mode.
(7) : Since X0 remains OFF, Y10 remains OFF.
(8) : Since Y10 remains OFF, external load remains OFF.

Fig. 2.10 When Input is in Direct Mode and Output is in Refresh Mode



(1) : Since the external contact is OFF when X0 is input in refresh mode, X0 remains OFF.
(2) : Since X0 remains OFF, Y10 remains OFF.
(3) : Since the external contact is ON when X0 is input in refresh mode, X0 is turned ON.
(4) : Since X0 is turned from OFF to ON, Y10 is turned ON.
(5) : Since the external contact is OFF when X0 is input in refresh mode, X0 is turned OFF.
(6) : Since X0 is turned OFF, Y10 is turned OFF.
(7) : Since the external contact is ON when X0 is input in refresh mode, X0 is turned ON.
(8) : Sine X0 is turned from OFF to ON, Y10 is turned ON.
(9) : Sine the external contact is OFF when X0 is input in refresh mode, X0 is turned OFF.

Fig. 2.11 When Input is in Refresh Mode and Output is in Direct Mode



(1) : Since the external contact is OFF when X0 is input in refresh mode, X0 remains OFF.
(2) : Since X0 is OFF, Y10 remains OFF.
(3) : The Y10 OFF data is output to external load in refresh mode.
(4) : Since the external contact is ON when X0 is input in refresh mode, X0 is turned ON.
(5) : Since X0 is turned OFF to ON, Y10 is turned ON.
(6) : The Y10 ON data is output to external load in refresh mode.
(7) : Since the external contact is OFF when X0 is input in refresh mode, X0 is turned OFF.
(8) : Since X0 is turned OFF, Y10 is turned OFF.
(9) : The Y10 OFF data is output to external load in refresh mode.
(10) : Since the external contact is ON when X0 is input in refresh mode, X0 is turned ON.
(11) : Since X0 is turned from OFF to ON, is turned ON.
(12) : The Y10 ON data is output to external load in refresh mode.
(13) : Since the external contact is OFF when X0 is input in refresh mode, X0 is turned OFF.

Fig. 2.12 When Both Input and Output are in Refresh Mode

Applicable CPU	All Types of CPUs
Remark	

2.5 Scan Time

(1) Scan time

Scan time refers to the period of time from the start of step 0 to the completion of the END instruction of a sequence program.
Length of scan time differs between scans and is determined by the number of instructions executed within one time of scan.

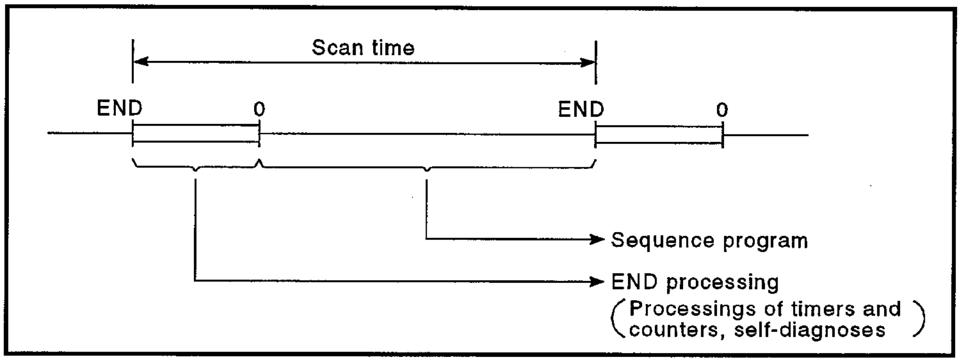


Fig. 2.13 Scan Time

(2) Storage of scan time data

(a) The PC stores scan times in special registers D9017 to 9019 in units of 10 ms.

1) Data stored in D9017 to 9019

- D9017........Minimum value of scan time
- D9018........Present value of scan time
- D9019........Maximum value of Scan time

2) Scan time accuracy

Scan time accuracy is ± 10 ms.
Hence, when the value in D9018 is 5, the actual scan time is between 40 and 60 ms.

3) D9017 to 9019 are not cleared and store the scan time if the WDT instruction is executed.

REMARKS

(1) In ladder monitor mode of the peripheral device, data of D9019 is displayed.

(2) The AnA and AnU stores present data of scan time (1 ms unit) in D9021.

(3) Scan time measurement may include errors when an interrupt program is used.
Refer to Section 5.1.3 for details.

Applicable CPU	All Types of CPUs
Remark	

2.6 Numeric Data Usable for Sequence Programs

Alphanumeric data used by the PC CPUs is represented by 0 (OFF) and 1 (ON), in the most basic form as binary data.
Other types of data such as hexadecimal data which represents binary data in units of 4 bits and BCD (binary-coded decimal) data are also usable.
Table 2.1 shows the representation of numeric values by the BCD, binary, hexadecimal, and decimal notations.

Table 2.1 Binary, Hexadecimal, BCD, and Decimal Notations

DEC (Decimal)	HEX (Hexadecimal)	BIN (Binary)	BCD (Binary-coded Decimal)
0	0	0	0
1	1	1	1
2	2	10	10
3	3	11	11
⋮	⋮	⋮	⋮
9	9	1001	1001
10	A	1010	1 0000
11	B	1011	1 0001
12	C	1100	1 0010
13	D	1101	1 0011
14	E	1110	1 0100
15	F	1111	1 0101
16	10	1 0000	1 0110
17	11	1 0001	1 0111
⋮	⋮	⋮	⋮
47	2F	10 1111	100 0111

(1) Input of numeric data from an external device to the PC CPU

To input numeric values using a digital switch or other external device to the PC CPU, use BCD (binary-coded decimal) data which allows data to be set in the form of decimal data.
Although the BCD data is binary-coded, it needs to be converted to binary data so that the PC CPU may execute correct operation.
A BIN instruction is provided to convert the BCD input data into binary data used for the operations by the PC CPU. It is recommended to write a program which converts numeric data to binary data. By using this conversion program, numeric data can be input from an external device without considering the BIN data.

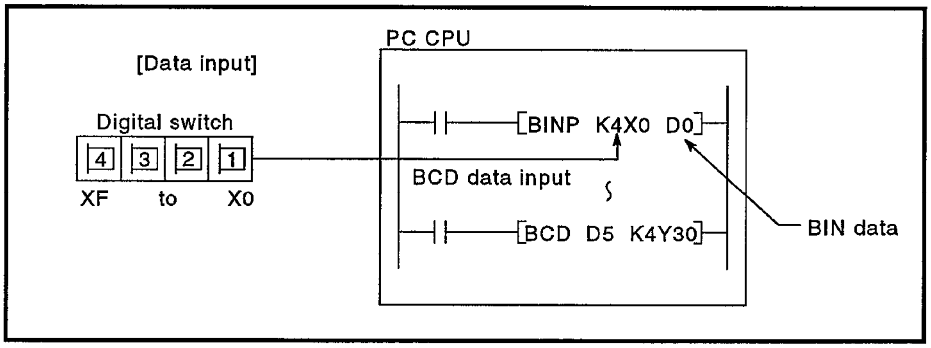


Fig. 2.14 Data Input to the PC CPU

(2) Output of numeric data from the PC CPU to an external device

To read operation results of the PC CPU with an external device, use a digital display.
The binary data used for operation by the PC CPU cannot be read correctly if it is output to a digital display directly.
A BCD instruction is provided to convert the binary data into the BCD data. It is recommended to write a program which converts numeric data to BCD data. By using this program, a display in decimal notation is possible.

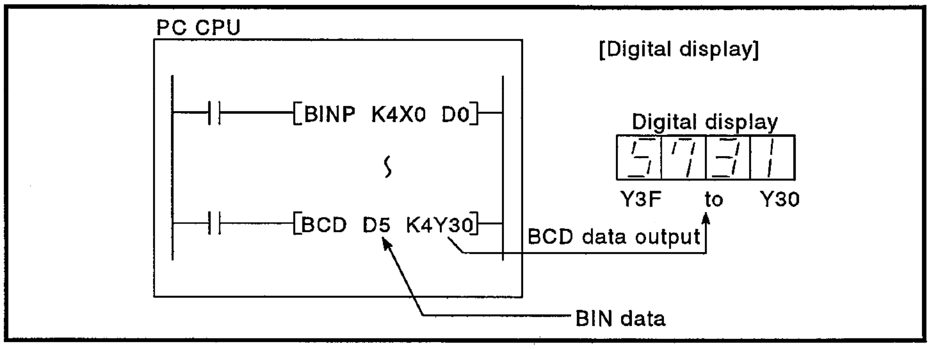


Fig. 2.15 Data Output from the PC CPU

Applicable CPU	All Types of CPUs
Remark	

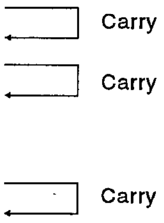
2.6.1 Binary notation

(1) Binary

Binary data is represented by 0 (OFF) and 1 (ON).
Decimal notation uses the numerals 0 through 9. When counting beyond 9, a 1 is placed in the 10s column and a 0 is placed in the 1s column to make the number 10.
In binary notation, the numerals 0 and 1 are used. A carry occurs after 1 and the number becomes 10 (decimal 2).
Table 2.2 gives a comparison between binary and decimal notations.

Table 2.2 Comparison between Binary and Decimal Notations

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011



(2) Representation of binary values

(a) The PC CPU uses 16-bit registers (data registers, link registers, etc.)
Each bit of a register is allocated with a 2^n value.
The most significant bit is used for the positive/negative judgment.
- If the most significant bit is 0: Positive
- If the most significant bit is 1: Negative

Fig. 2.16 gives numeric expression used for the registers of the PC CPU.

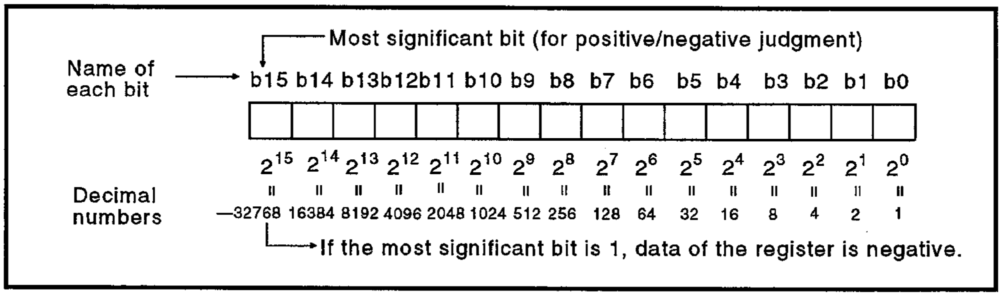


Fig. 2.16 Numeric Expression for Each Register of the PC CPU.

(b) Data usable with the PC CPU

According to the numeric expression shown in Fig. 2.16, numeric values from —32768 to 32767 can be expressed.
Therefore, values from —32768 to 32767 can be stored in the registers of the PC CPU.

Applicable CPU	All Types of CPUs
Remark	

2.6.2 Hexadecimal

(1) Hexadecimal notation

In hexadecimal notation, 4 binary bits are expressed in 1 digit.
If 4 binary bits are used in binary notation, 16 different values from 0 to 15 can be represented.
Since hexadecimal notation represents 0 to 15 in 1 digit, letters A to F are used to represent the numbers 10 to 15.
Then, a carry occurs after F.
Table 2.3 shows numeric representation by binary, hexadecimal, and decimal notations.

Table 2.3 Numeric Representation by Binary, Hexadecimal and Decimal Notations

Decimal	Hexadecimal	Binary
0	0	0
1	1	1
2	10	10
3	11	11
⋮	⋮	⋮
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	1 0000
17	11	1 0001
⋮	⋮	⋮
47	2F	10 1111

(2) Expression of hexadecimal values

The PC CPU uses 16-bit registers (data registers, link registers, etc.)
Data which can be stored in each register is within 0 to FFFFH range in hexadecimal notation.

Applicable CPU	All Types of CPUs
Remark	

2.6.3 BCD (Binary-coded decimal)

(1) BCD notation

BCD notation represents each decimal digit with 4 binary digits.
Though it uses 4-bit representation like hexadecimal notation, it does not use letters A to F.
Table 2.4 gives numeric representation by binary, BCD, and decimal notations.

Table 2.4 Numeric Representation by Binary, BCD, and Decimal Notations

Decimal	Binary	BCD (Binary-coded Decimal)
0	0	0
1	1	1
2	10	10
3	11	11
4	100	100
5	101	101
6	110	110
7	111	111
8	1000	1000
9	1001	1001
10	1010	1 0000
11	1011	1 0001
12	1100	1 0010

(2) Expression of BCD values

The PC CPU uses 16-bit registers (data registers, link registers, etc.)
Data which can be stored in each register is within 0 to 9999 range in BCD notation.

3. DEVICES

3.1 List of Devices

Table 3.1 shows the devices and their range of use which are usable with the PC CPU.
The usable range for devices marked by an asterisk (*) in Table 3.1 can be set or changed by setting parameters with a peripheral device.
For the parameter set range refer to Section 8 "PARAMETER SETTING".

Table 3.1 List of Devices

		Device	A1S, A1SJ-S3	A1SH, A1SJH(S8)	A1S-S1, A2S	A2SH	A2S-S1	A2SH-S1	A0J2H	A2C A52G	A1FX
		I/O device points *1	256 points	2048 points	512 points	2048 points	1024 points	2048 points	336 points *4	512 points	512 points
	X	Input *2	X, Y0 to FF (Total X and Y points: 256)		X, Y0 to 1FF (Total X and Y points: 512)		X, Y0 to 3FF (Total X and Y points:1024)		X, Y0 to 1FF *4 (Total X and Y points: 336)	X, Y0 to 1FF (Total X and Y points: 512)	X, Y0 to F1 (Total X and Y points: 224)
	Y	Output *2									
	M	Special relay	M9000 to M9255 (256 points)								
*		Internal relay	M0 to 999 (1000 points)						Setting can be changed using parameters (Total M, L, S: 2048 points) A1FX cannot use step relays.		
		Latch relay	L1000 to 2047 (1048 points)								
		Step relay	0 point								
	B	Link relay	B0 to 3FF (1024 points)								
*	T	100 ms timer	T0 to 199 (200 points)						Setting can be changed using parameters (Total of timers: 256 points)		
		10 ms timer	T200 to 255 (256 points)								
		100 ms retentive timer	0 point								
		1ms timer	—								
*	C	Counter	C0 to 255 (256 points)						Setting can be changed using parameters (Total of counters and interrupt counters: 256 points) A2C and A52G cannot use interrupt counters.		
		Interrupt counter *3	0 point								
			For interrupt programs (A2C and A52G cannot use interrupt counters.)								
	D	Data register	D0 to 1023 (1024 points)								
		Special register	D9000 to D9255 (256 points)								
	W	Link register	W0 to 3FF (1024 points)								
*	R	File register	0 point (Parameter setting: 1 to 4 k points)								
	A	Accumulator	A0, A1 (2 points)								
	Z	Index register	Z (1 point)								
	V		V (1 point)								
	N	Nesting	N0 to 7 (8 levels)								
	P	Pointer	P0 to 255 (256 points)								
	I	Interrupt pointer	I0 to I31 (32 points)							—	I0 to I5, I12, I13, I29 to I31, (11 points)
	K	Decimal constant	K-32768 to 32767 (16-bit instruction)								
			K-2147483648 to 2147483647 (32-bit instruction)								
	H	Hexadecimal constant	H0 to FFFF (16-bit instruction)								
			H0 to FFFFFFFF (32-bit instruction)								

*1 : Indicates the number of points that can be specified by input/output (X, Y).
Numbers which follow those allocated to I/O modules and special function modules may be allocated to remote I/O stations and MELSECNET/MINI-S3 in the data link system.

*2 : Indicates the I/O numbers and the number of I/O points that can actually control the I/O modules and special function modules.

*3 : Interrupt counters for an interrupt program should be set in unit of 8 points in the range from C0 to C255.
(Counters and interrupt counters for an interrupt program are numbered consecutively.)
Interrupt counters for counting the number of interrupts are fixed to C224 to C255.

*4 : Can be expanded to 480 points if an extension base unit is used.

Table 3.1 List of Devices

		Device	A2AS	A2AS(S1/S30), A2USH-S1	Q02, Q02H, Q06H
		I/O device points *1	8192 points	8192 points	
	X	Input *2	X, Y0 to 1FF (Total X and Y points: 512)	X, Y0 to 3FF (Total X and Y points: 1024)	X, Y0 to 1FFF (Total X and Y points: 4096)
	Y	Output *2			
*	M	Special relay	M9000 to M9255 (256 points)		
		Internal relay	M0 to 999 (1000 points) M2048 to 8191 (6144 points)	Setting can be changed using parameters (Total M, L, S: 8192 points)	
		Latch relay	L1000 to 2047 (1048 points)		
		Step relay	0 point		
	B	Link relay	B0 to 1FFF (8192 points)		
*	T	100 ms timer	T0 to 199 (200 points), T256 to 2047 (1848 points)*4	Setting can be changed using parameters (Total of timers: 2048 points)	
		10 ms timer	T200 to 255 (256 points)		
		100 ms retentive timer	0 point		
		1ms timer	—		0 point Usable with the ZHTIME instruction
*	C	Counter	C0 to 255 (256 points), C256 to 1023 (768 points)*5	Setting can be changed using parameters (Total of counters and interrupt counters: 1024 points)	
		Interrupt counter *3	0 point		
			For counting the number of interrupts		
	D	Data register	D0 to 8191 (8192 points)		
		Special register	D9000 to D9255 (256 points)		
	W	Link register	W0 to 1FFF (8192 points)		
*	R	File register	0 point (Parameter setting: 1 to 8 k points)		
	A	Accumulator	A0, A1 (2 points)		
	Z	Index register	Z, Z1 to Z6 (7 point)		
	V		V, V1 to V6 (7 point)		
	N	Nesting	N0 to 7 (8 levels)		
	P	Pointer	P0 to 255 (256 points)		
	I	Interrupt pointer	I0 to I31 (32 points)		
	K	Decimal constant	K-32768 to 32767 (16-bit instruction)		
			K-2147483648 to 2147483647 (32-bit instruction)		
	H	Hexadecimal constant	H0 to FFFF (16-bit instruction)		
			H0 to FFFFFFFF (32-bit instruction)		

*1 : Indicates the number of points that can be specified by input/output (X, Y). Numbers which follow those allocated to I/O modules and special function modules may be allocated to remote I/O stations and MELSECNET/MINI-S3 in the data link system.

*2 : Indicates the I/O numbers and the number of I/O points that can actually control the I/O modules and special function modules.

*3 : Interrupt counters for an interrupt program should be set in unit of 8 points in the range from C0 to C255.
(Counters and interrupt counters for an interrupt program are numbered consecutively.)
Interrupt counters for counting the interrupt occurrences are fixed to C224 to C255.

*4 : Timers T256 to T2047 are called extended timers.
To use extended timers, the number of points and the type of timers (100 ms timer, 10 ms timer, 100 ms retentive timer) to be used must be set for parameters and the setting of a setting value register is also necessary.

*5 : Counters C256 to C1023 are called extended counters.
To use extended counters, the number of points of counters must be set for parameters and the setting of a setting value register is also necessary.

Table 3.1 List of Devices

Device			A1 A1N	A2 A2N	A2-S1 A2N-S1	A3, A3N A3V, A73	A3H A3M
		I/O device points [1]	256 points	512 points	1024 points	2048 points	2048 points
	X	Input [2]	X, Y0 to FF (Total X and Y points: 256)	X, Y0 to 1FF (Total X and Y points: 512)	X, Y0 to 3FF (Total X and Y points: 1024)	X, Y0 to 7FF (Total X and Y points: 2048)	X, Y0 to 7FF (Total X and Y points: 2048)
	Y	Output [2]					
*	M	Special relay	M9000 to M9255 (256 points)				
		Internal relay	M0 to 999 (1000 points)			Setting can be changed using parameters (Total M, L, S: 2048 points)	
		Latch relay	L1000 to 2047 (1048 points)				
		Step relay	0 point				
	B	Link relay	B0 to 3FF (1024 points)				
*	T	100 ms timer	T0 to 199 (200 points)			Setting can be changed using parameters (Total of timers: 256 points)	
		10 ms timer	T200 to 255 (256 points)				
		100 ms retentive timer	0 point				
		1ms timer	—				
*	C	Counter	C0 to 255 (256 points)			Setting can be changed using parameters (Total of counters and interrupt counters: 256 points)	
		Interrupt counter [3]	0 point				
			For interrupt programs				For counting the number of interrupts
	D	Data register	D0 to 1023 (1024 points)				
		Special register	D9000 to D9255 (256 points)				
	W	Link register	W0 to 3FF (1024 points)				
*	R	File register	0 point (Parameter setting: 1 to 4 k points)			0 point (Parameter setting: 1 to 8 k points)	
	A	Accumulator	A0, A1 (2 points)				
	Z	Index register	Z (1 point)				
	V		V (1 point)				
	N	Nesting	N0 to 7 (8 levels)				
	P	Pointer	P0 to 255 (256 points)				
	I	Interrupt pointer	I0 to I31 (32 points)				
	K	Decimal constant	K-32768 to 32767 (16-bit instruction)				
			K-2147483648 to 2147483647 (32-bit instruction)				
	H	Hexadecimal constant	H0 to FFFF (16-bit instruction)				
			H0 to FFFFFFFF (32-bit instruction)				

[1] Indicates the number of points that can be specified by input/output (X, Y).
Numbers which follow those allocated to I/O modules and special function modules may be allocated to remote I/O stations and MELSECNET/MINI-S3 in the data link system.

[2] Indicates the I/O numbers and the number of I/O points that can actually control the I/O modules and special function modules.

[3] Interrupt counters for an interrupt program should be set in unit of 8 points in the range from C0 to C255.
(Counters and interrupt counters for an interrupt program are numbered consecutively.)
Interrupt counters for counting the interrupt occurrences are fixed to C224 to C255.

Table 3.1 List of Devices

		Device	A2A	A2A-S1	A3A	A2U	A2U-S1	A3U	A4U
		I/O device points *1	512 points	1024 points	2048 points	8192 points			
	X	Input *2	X, Y0 to 1FF (Total X and Y points: 512)	X, Y0 to 3FF (Total X and Y points: 1024)	X, Y0 to 7FF (Total X and Y points: 2048)	X, Y0 to 1FF (Total X and Y points: 512)	X, Y0 to 3FF (Total X and Y points: 1024)	X, Y0 to 7FF (Total X and Y points: 2048)	X, Y0 to FFF (Total X and Y points: 4096)
	Y	Output *2							
*	M	Special relay	M9000 to M9255 (256 points)						
		Internal relay	M0 to 999 (1000 points), M2048 to 8191 (6144 points)				Setting can be changed using parameters (Total M, L, S: 8192 points)		
		Latch relay	L1000 to 2047 (1048 points)						
		Step relay	0 point						
	B	Link relay	B0 to FFF (4096 points)			B0 to 1FFF (8192 points)			
*	T	100 ms timer	T0 to 199 (200 points), T256 to 2047 (1848 points)*4				Setting can be changed using parameters (Total of timers: 2048 points)		
		10 ms timer	T200 to 255 (256 points)						
		100 ms retentive timer	0 point						
		1ms timer	—						
*	C	Counter	C0 to 255 (256 points), C256 to 1023 (768 points)*5				Setting can be changed using parameters (Total of counters and interrupt counters: 1024 points)		
		Interrupt counter *3	0 point						
			For counting the number of interrupts						
	D	Data register	D0 to 6143 (6144 points)			D0 to 8191 (8192 points)			
		Special register	D9000 to D9255 (256 points)						
	W	Link register	W0 to FFF (4096 points)			W0 to 1FFF (8192 points)			
*	R	File register	0 point (Parameter setting: 1 to 8 k points)						
	A	Accumulator	A0, A1 (2 points)						
	Z	Index register	Z, Z1 to Z6 (7 points)						
	V		V, V1 to V6 (7 points)						
	N	Nesting	N0 to 7 (8 levels)						
	P	Pointer	P0 to 255 (256 points)						
	I	Interrupt pointer	I0 to I31 (32 points)						
	K	Decimal constant	K-32768 to 32767 (16-bit instruction)						
			K-2147483648 to 2147483647 (32-bit instruction)						
	H	Hexadecimal constant	H0 to FFFF (16-bit instruction)						
			H0 to FFFFFFFF (32-bit instruction)						

*1 Indicates the number of points that can be specified by input/output (X, Y). Numbers which follow those allocated to I/O modules and special function modules may be allocated to remote I/O stations and MELSECNET/MINI-S3 in the data link system.

*2 Indicates the I/O numbers and the number of I/O points that can actually control the I/O modules and special function modules.

*3 Interrupt counters for an interrupt program should be set in unit of 8 points in the range from C0 to C255.
(Counters and interrupt counters for an interrupt program are numbered consecutively.)
Interrupt counters for counting the interrupt occurrences are fixed to C224 to C255.

*4 Timers T256 to T2047 are called extended timers.
To use extended timers, the number of points and the type of timers (100 ms timer, 10 ms timer, 100 ms retentive timer) to be used must be set for parameters and the setting of a setting value register is also necessary.

*5 Counters C256 to C1023 are called extended counters.
To use extended counters, the number of points of counters must be set for parameters and the setting of a setting value register is also necessary.

Applicable CPU	**All Types of CPUs**
Remark	

3.2 Input (X) and Output (Y)

The input and output devices are used for data transaction between the PC CPU and external devices.
The input devices hold ON/OFF data sent from external devices to the input module. Input data is used by the program as contact data (N/O and N/C contacts) and as the source data for basic and application instructions.
The output devices are used to output operation results of the program from the output module to external devices.

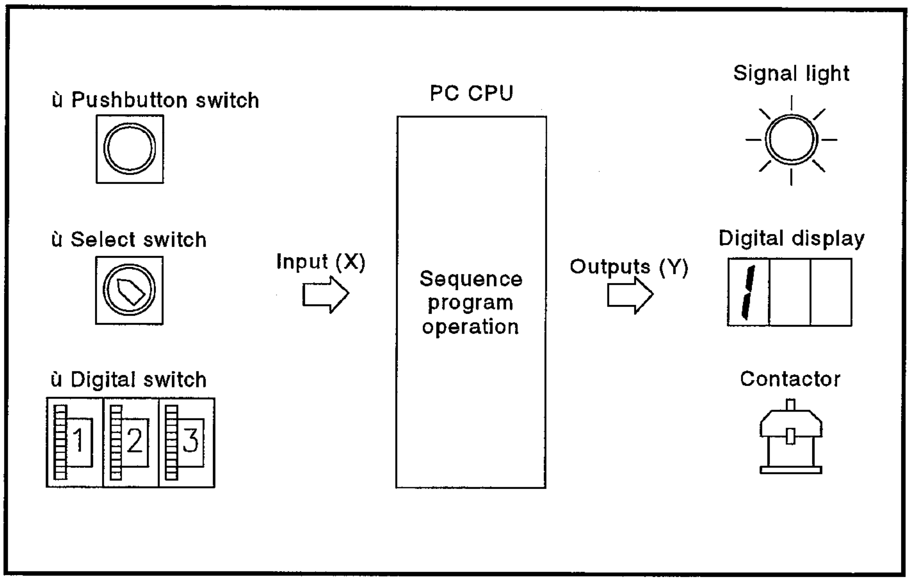


Fig. 3.1 Inputs (X) and Outputs (Y)

(1) Input X

(a) Inputs give commands and data from external devices (e.g. push buttons, select switches, limit switches, digital switches) to the PC.

(b) Regarding that one point of input incorporates a virtual relay Xn in the PC, the N/O contact and N/C contact of that Xn are used in the program.

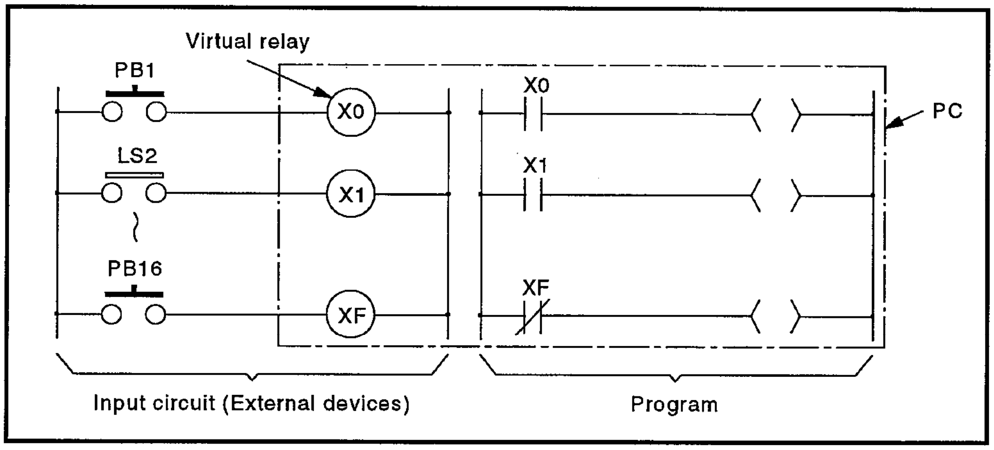


Fig. 3.2 Inputs (X)

(c) There is no restriction on the number of N/O contacts and N/C contacts of Xn used in the program.

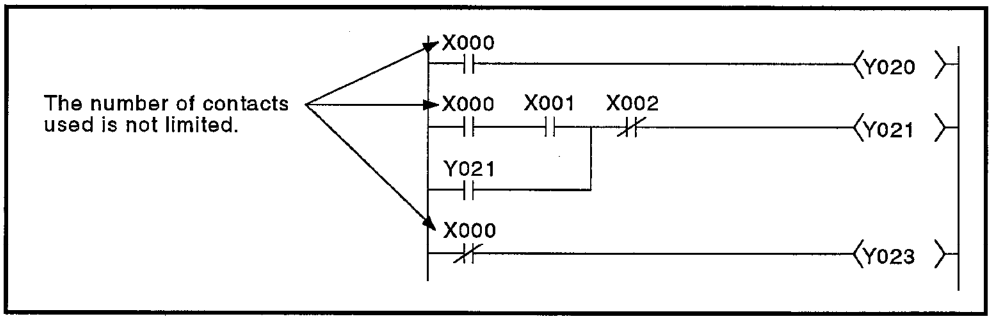


Fig. 3.3 Used in the Inputs (X)

(2) Output Y

(a) Outputs provide program control results to external devices (e.g. solenoids, magnetic switches, signal lamps, digital indicators).

(b) Outputs can be fetched to the outside as an equivalent to one N/O contact.

(c) There is no restriction on the number of N/O contacts and N/C contacts of Yn used in the program.

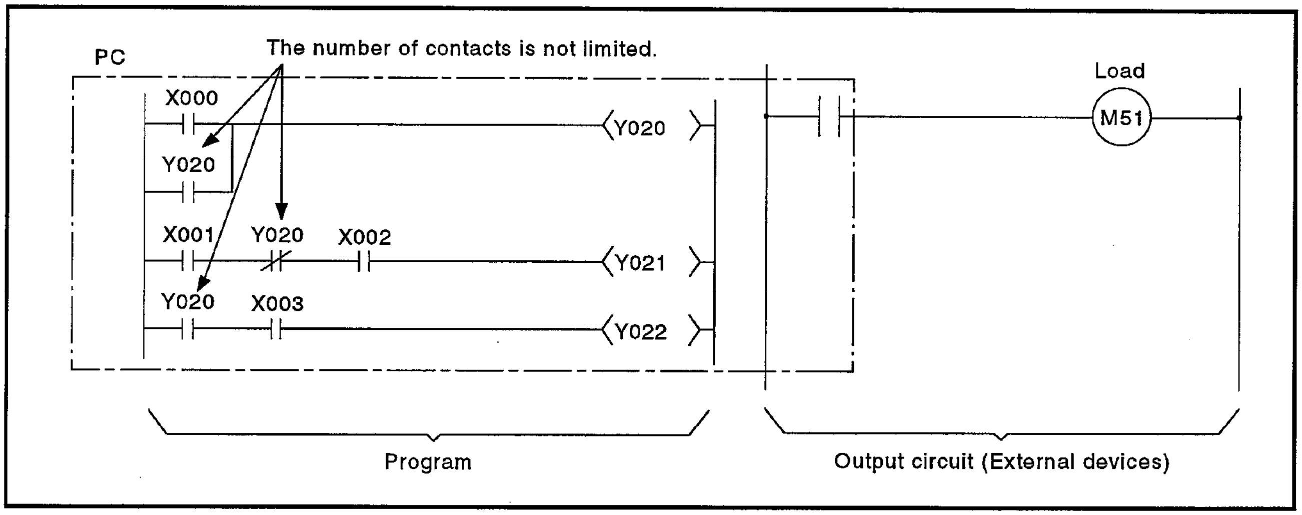

PC
The number of contacts is not limited.
X000
Y020
Y020
X001
Y020
X002
Y021
Y020
X003
Y022
Load
M51
Program
Output circuit (External devices)

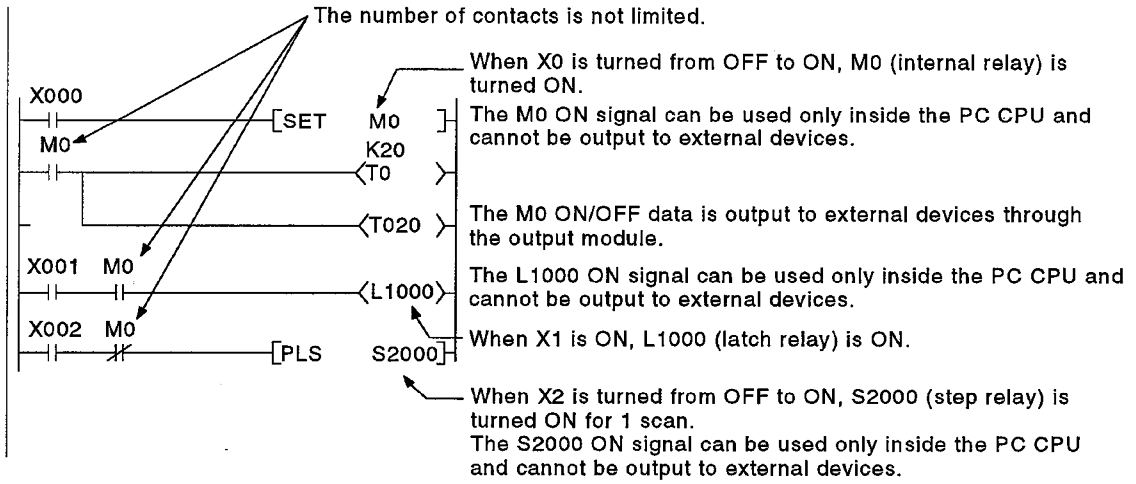
Fig. 3.4 Outputs (Y)

(d) The Y range which corresponds to the range to where the input modules are loaded and the range to where no module is loaded can be used for the internal relay M.
The Y range used for the internal relay M cannot be latched.

Applicable CPU	All Types of CPUs
Remark	

3.3 Internal Relays (M), Latch Relays (L), and Step Relays (S)

The internal relays, latch relays, and step relays are auxiliary relays used in the PC CPU.
The number of contacts (N/O and N/C) used in the sequence program is not limited.
The output (Y) devices are used to output operation results of the sequence program to external devices.



The number of contacts is not limited.

When X0 is turned from OFF to ON, M0 (internal relay) is turned ON.
The M0 ON signal can be used only inside the PC CPU and cannot be output to external devices.

The M0 ON/OFF data is output to external devices through the output module.

The L1000 ON signal can be used only inside the PC CPU and cannot be output to external devices.

When X1 is ON, L1000 (latch relay) is ON.

When X2 is turned from OFF to ON, S2000 (step relay) is turned ON for 1 scan.
The S2000 ON signal can be used only inside the PC CPU and cannot be output to external devices.

Fig. 3.5 Internal Relays, Latch Relays, and Step Relays

The number of internal relays, latch relays, and step relays is changed or set by parameters.

(1) Internal relays M, step relays S

Cannot be latched. Hence, all internal relays are switched off if the PC is switched on, reset or latch-cleared.

(2) Latch relay L

(a) Battery backed, i.e. operation results are retained if the PC is switched on or reset.

(b) Set the CPU to STOP, and latch-clear to switch off the latch relays from the external device.

Applicable CPU	All Types of CPUs
Remark	

3.4 Link Relays (B)

(1) Internal relay for use in a data link system.
When not using a data link, link relays can be used as internal relays.

(2) The ON/OFF data of the link relays used in a data link system can be read by switching them on/off as coils in the host (master, local station) and as contacts in other stations (master, local stations). Link relays thus allow ON/OFF data to be transferred between the master and local stations.

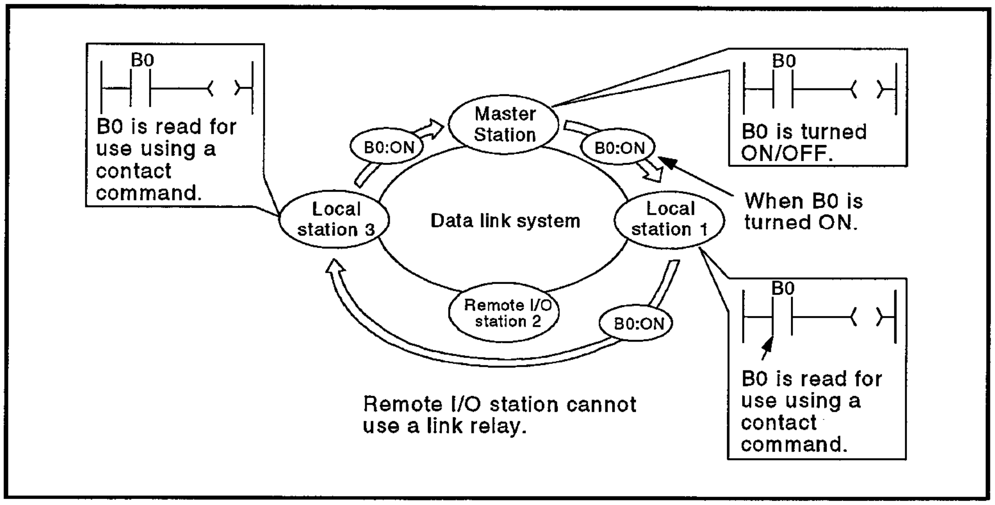


Fig. 3.6 Link Relay in a Data Link System

(3) The link range (range of link relays for use as coils in each station) must be set to the master station. Link relays outside the link range may be used as internal relays.

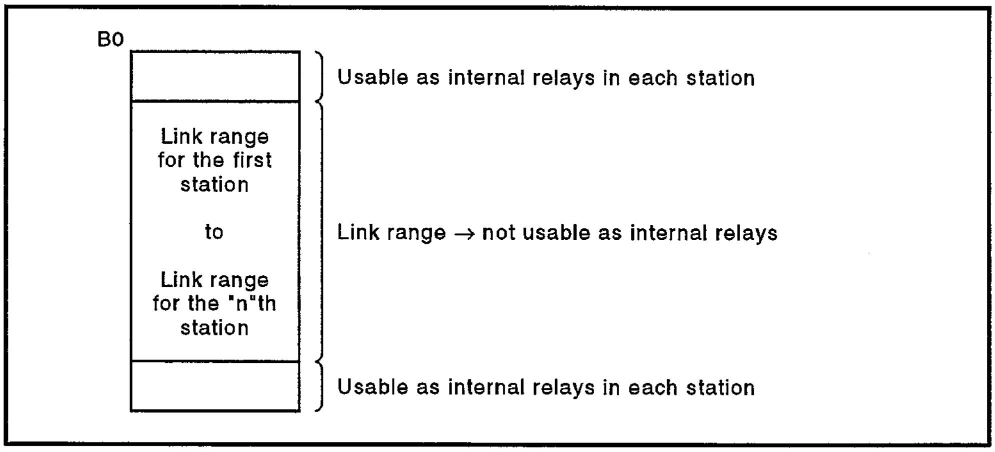
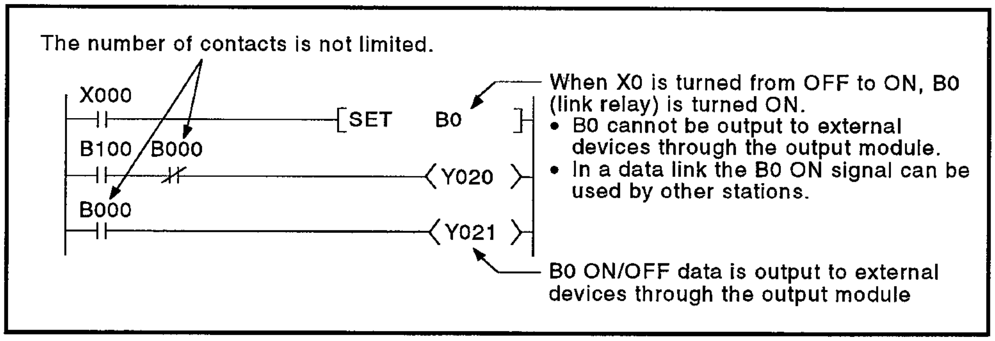

B0
Usable as internal relays in each station
Link range for the first station
to
Link range for the "n"th station
Link range → not usable as internal relays
Usable as internal relays in each station

Fig. 3.7 Assignment of Link Relays

(4) There is no restriction on the number of N/O contacts and N/C contacts of link relay used in the program.



The number of contacts is not limited.

When X0 is turned from OFF to ON, B0 (link relay) is turned ON.
- B0 cannot be output to external devices through the output module.
- In a data link the B0 ON signal can be used by other stations.

B0 ON/OFF data is output to external devices through the output module

Fig. 3.8 Processing of Link Relays

REMARK

For detail of the data link system, refer to the MELSECNET, MELSECNET/B Data Link Reference Manual (IB-66350).

Applicable CPU	All Types of CPUs
Remark	

3.5 Annunciators (F)

Annunciators are used with a fault detection program.
If an annunciator is set, a special relay is turned ON, and the annunciator number is stored in a special register.

(a) Special relay : M9009..... If an annunciator is set, M9009 is turned ON.

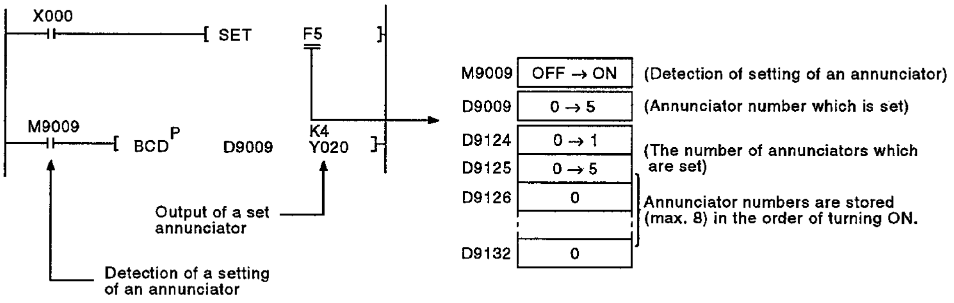
(b) Special registers : D9009..... Stores the annunciator number which is set first.
: D9124..... Stores the number of annunciators which are set.
: D9125 to D9132 ... Stores the annunciator numbers in order of setting.
(D9125 and D9009 store the same annunciator number.)

By using annunciators in a fault detection program, occurrence of fault and fault data (annunciator numbers) can be checked by monitoring M9009 and D9009.

Example

In the sequence program shown below, when F5 is set, M9009 is turned ON and "5" is stored in D9009.

(Fault detection program)


X000
SET
F5
M9009
BCD P
D9009
K4 Y020
Output of a set annunciator
Detection of a setting of an annunciator
M9009 OFF → ON (Detection of setting of an annunciator)
D9009 0 → 5 (Annunciator number which is set)
D9124 0 → 1
D9125 0 → 5 (The number of annunciators which are set)
D9126 0
D9132 0
Annunciator numbers are stored (max. 8) in the order of turning ON.

(1) How to set annunciators

(a) Annunciators can be set using the OUT (OUT F□) instruction or the SET (SET F□) instruction.

1) The OUT instruction sets/resets annunciators according to ON/OFF of an input condition. The OUT instruction is executed every scan. If annunciators are reset by the OUT instruction, data of M9009, D9009, and D9124 to 9132 does not change.

2) The SET instruction is executed at the leading edge (OFF to ON) of an input condition and sets annunciators.
(If the input condition is turned OFF, annunciators are held in set status.)
If a large number of annunciators are used, it is recommended to use the SET instruction instead of the OUT instruction to make the scan time short.

(b) When annunciators (F□) are set:

1) Annunciator numbers (F□) are stored in order of setting in D9125 to 9132. When data of D9124 is 0, the first set annunciator number is stored also in D9009. The number of set annunciators stored in D9124 increases by one each time an annunciator is set.

SET F50 SET F25 SET F99

D9009	0	50	50	50
D9124	0	1	2	3
D9125	0	50	50	50
D9126	0	0	25	25
D9127	0	0	0	99

2) Indication of annunciators varies with type of the CPU module as follows.

a) The CPU modules having an LED display on the front panel: The F number stored in D9009 is indicated on the LED display.

b) The CPU modules having an "ERROR" LED on the front panel: The "ERROR" LED flashes. (When it lights, it indicates an operation error.)

c) The A0J2H CPU module: The "RUN" LED flashes or lights according to the status of M9048, as follows.
When M9048 is ON: The "RUN" LED flashes.
When M9048 is OFF: The "RUN" LED remains lit.

POINTS

(1) If an annunciator is set when any annunciator is already set, the F number being indicated on the LED display does not change.
(2) If the number of set F numbers has become larger than 8, data of D9124, and D9125 to 9132 does not change.
However, data of the set F numbers is stored in the PC.

(2) How to reset annunciators

(a) Annunciators can be reset by the RST (RST F[ ]) instruction or the LEDR instruction.

1) Use the RST (RST F[ ]) instruction to reset the set annunciator numbers.

2) The LEDR instruction is used to reset the annunciator numbers stored in D9009 and D9125.
Fig. 3.9 is the examples of programs used to reset annunciators by the LEDR instruction.

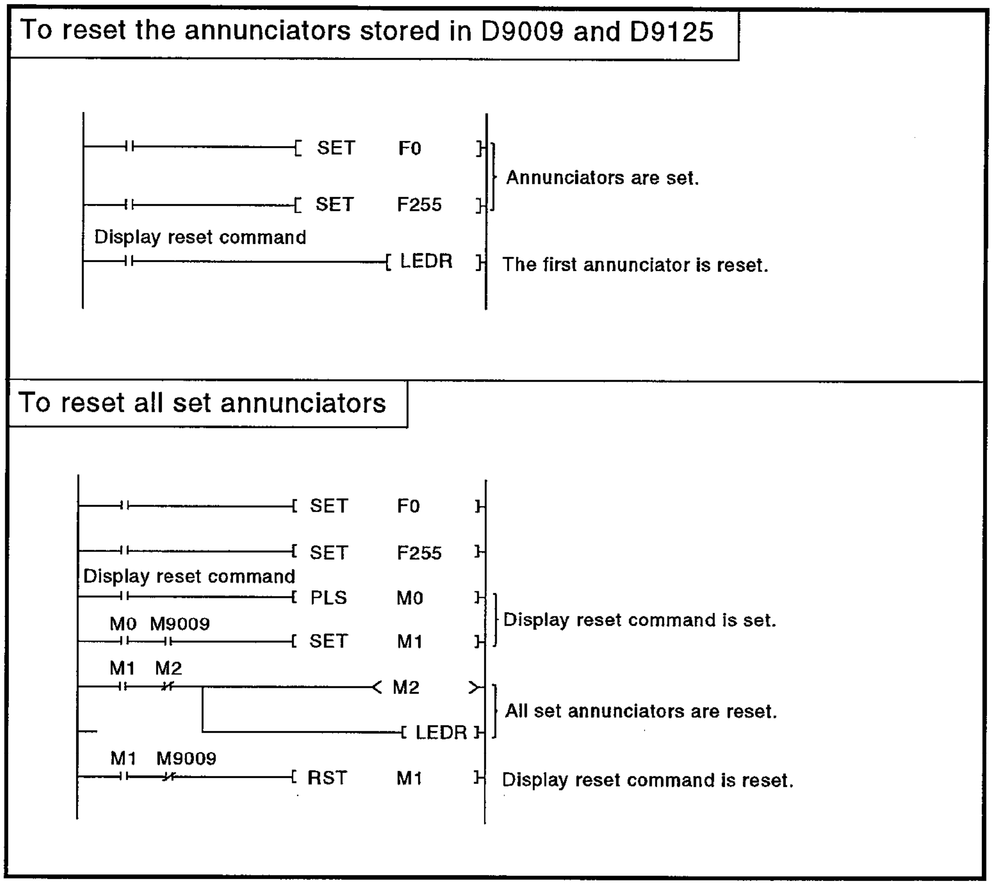

To reset the annunciators stored in D9009 and D9125
SET F0
SET F255
Annunciators are set.
Display reset command
LEDR
The first annunciator is reset.
To reset all set annunciators
SET F0
SET F255
Display reset command
PLS M0
M0 M9009
SET M1
Display reset command is set.
M1 M2
M2
LEDR
All set annunciators are reset.
M1 M9009
RST M1
Display reset command is reset.

Fig. 3.9 Programs to Reset Annunciators

POINTS

(1) If annunciators are set by an instruction other than the OUT or SET instruction, they function as if they are internal relays.
(Note that M9009 is not set, and annunciator numbers are not stored in D9009 and D9124 t D9132.)

(2) Even when the annunciator is reset by an OUT instruction, contents of M9009, D9009, and D9124 to D9132 do not change.
When the annunciator is set by an OUT instruction, reset the annunciator as follows:
1) Reset the annunciator by an OUT instruction.
2) Execute RST instruction (RST F[ ]) /LEDR instructions.

(b) When annunciators are reset by the RST F[ ] instruction:

1) The F numbers of reset annunciators are erased from D9125 to 9132. And then, data stored in the special registers and the F numbers stored in the PC are shifted forward. If the annunciator of which F number is stored in D9125 is reset, the data shifted forward and newly stored to D9125 is stored also in D9009. The number of set annunciators, stored in D9124, decreases one.

		SET F50	SET F25	SET F99	RST F25	RST F50
D9009	0	50	50	50	50	99
D9124	0	1	2	3	2	1
D9125	0	50	50	50	50	99
D9126	0	0	25	25	99	0
D9127	0	0	0	99	0	0

		SET F100	SET F126	RST F50	RST F25
D9009	50	50	50	25	99
D9124	8	8	8	8	8
D9125	50	50	50	25	99
D9126	25	25	25	99	15
D9132	210	210	210	100	120

2) If the CPU module is provided with an LED display, the F number stored in D9125 is indicated on the display. The F number newly stored in D9125 after reset is indicated on the LED display.

(Special registers)

D9009	50
D9124	3
D9126	50
D9125	25
D9127	99
⋮	

(LED display)

F 50

RST F50 →

D9009	25
D9124	2
D9125	25
D9126	99
D9127	0
⋮	

F 25

3) If data in D9124 has become 0 by resetting the F numbers, the "LED display" or the "ERROR" LED turns OFF.

POINT

To reset annunciators, the LEDR instruction or, if the CPU module is provided with an LED display, the "INDICATOR RESET" switch on the front panel may be used.

Applicable CPU	All Types of CPUs
Remark	

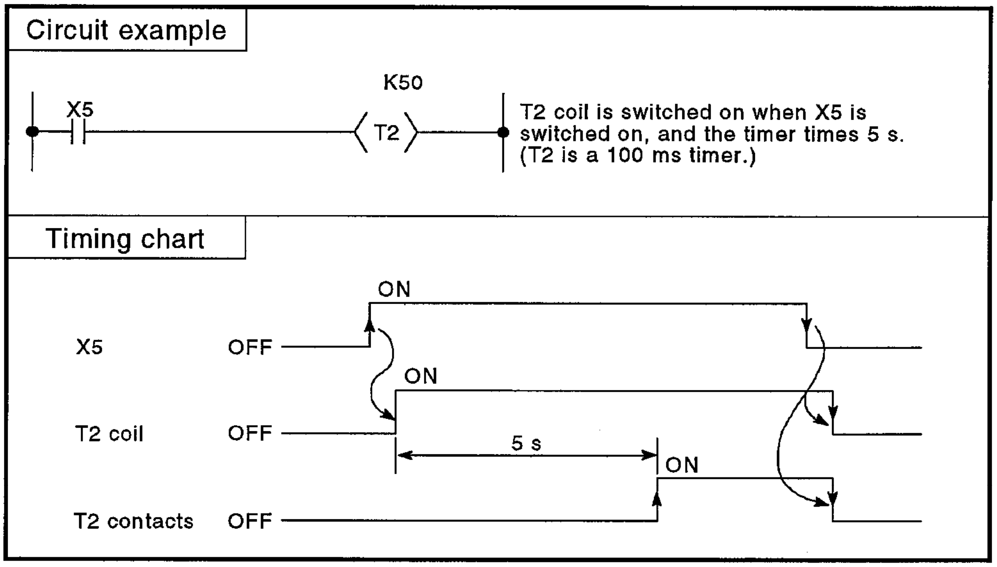
3.6 Timers (T)

The PC CPU uses up count timers.
An up count timer starts timing of the present value when its coil is turned ON. When the present value has become equal with its preset value (time out), the contact of the timer is turned ON.

3.6.1 100 ms timers, 10 ms timers, and 100 ms retentive timers

(1) 100 ms and 10 ms timers

(a) Timing of the present value starts when the timer coil is turned ON. When the timer coil is turned OFF, the present value becomes 0 and the contact is turned OFF.


Circuit example

T2 coil is switched on when X5 is switched on, and the timer times 5 s. (T2 is a 100 ms timer.)

Timing chart

Fig. 3.10 Timing Chart

(2) 100 ms retentive timer

(a) The 100 ms retentive timer measures the length of ON time of a timer coil.
When a timer coil is turned ON, timing of the present value starts.
When the coil is turned OFF, the present value and the ON/OFF status of the contact are retained.
When the coil is turned ON again, timing starts with the retained present value.

(b) Use the RST T[ ] instruction to clear the present value and to turn OFF the contact.

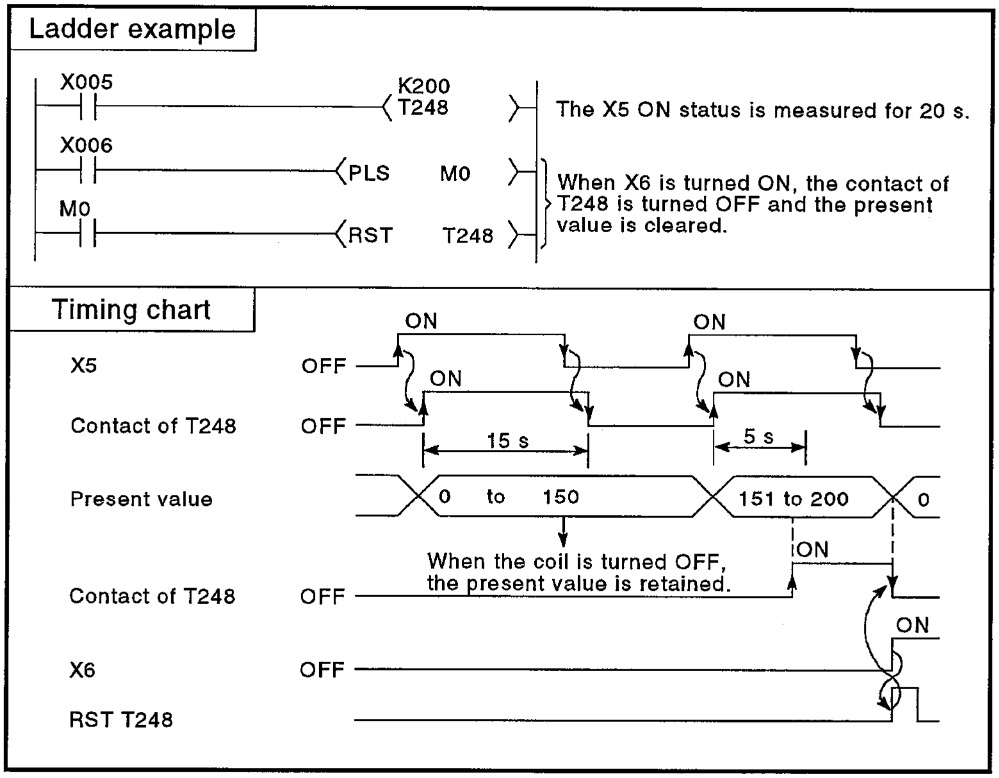

Ladder example
X005
K200
T248
The X5 ON status is measured for 20 s.
X006
PLS M0
M0
RST T248
When X6 is turned ON, the contact of T248 is turned OFF and the present value is cleared.
Timing chart
X5
OFF
ON
ON
Contact of T248
OFF
ON
ON
15 s
5 s
Present value
0 to 150
151 to 200
0
When the coil is turned OFF, the present value is retained.
ON
Contact of T248
OFF
ON
X6
OFF
RST T248

Fig. 3.11 Timing Chart

Applicable CPU	All Types of CPUs
Remark	

3.6.2 Processing and accuracy of timers

(1) Timer processing

A timer coil turned ON/OFF by execution of the OUT T□ instruction. The present value is updated and the contact is turned ON/OFF after execution of the END instruction.

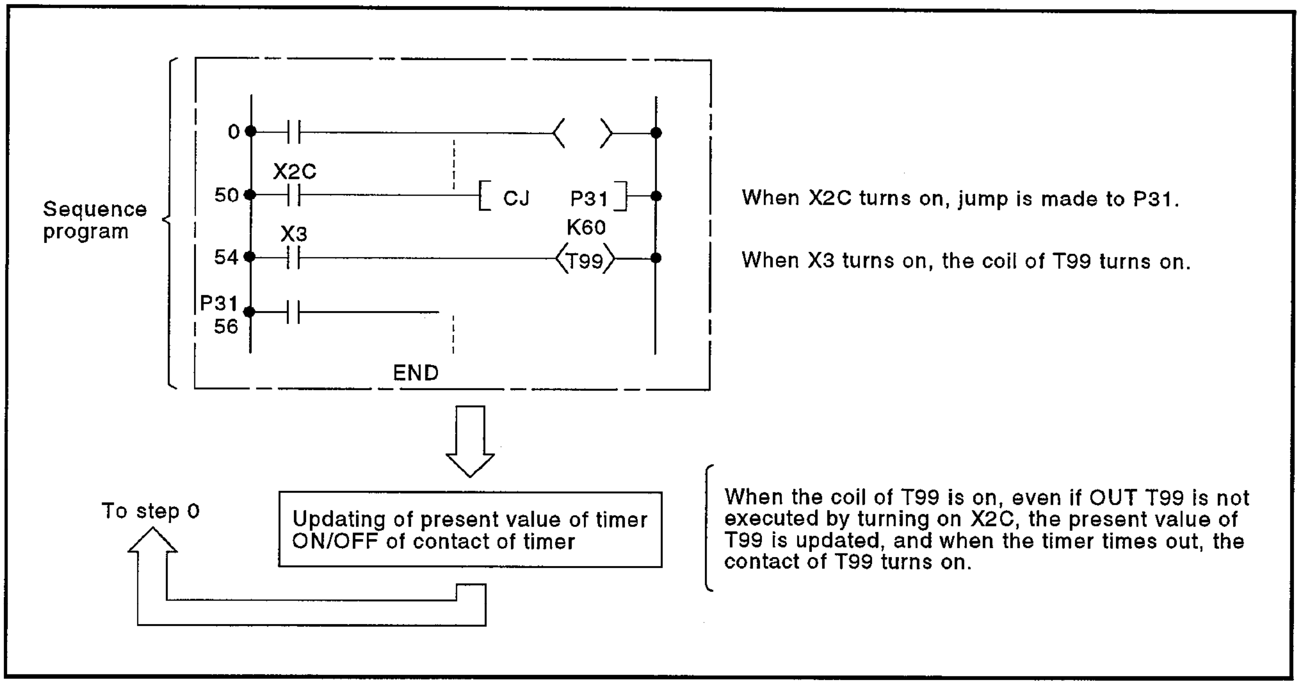
(a) When a timer coil is turned ON, the present value of the timer is updated after execution of the END instruction.
When the timer has timed out, the contact of the timer is turned ON.

1) When the timer coil is turned OFF, the 10 ms and 100 ms timers reset the present value to 0 and the contact is turned OFF after execution of the END instruction.

2) The 100 ms timer retains the present value and the ON/OFF status of the contact when the timer coil is turned OFF.

(b) When a timer is reset by execution of the RST T□ instruction, the present value is reset to 0 and the contact is turned OFF.

(c) If the OUT T□ instruction is skipped by use of the CJ instruction after a timer coil is turned ON, the timer coil status is held ON. Then, the present value is updated without execution of the OUT T□ instruction and the contact is turned ON when the timer times out.


Sequence program
0
X2C
50
CJ
P31
X3
K60
54
T99
P31
56
END
To step 0
Updating of present value of timer
ON/OFF of contact of timer

When X2C turns on, jump is made to P31.

When X3 turns on, the coil of T99 turns on.

When the coil of T99 is on, even if OUT T99 is not executed by turning on X2C, the present value of T99 is updated, and when the timer times out, the contact of T99 turns on.

Fig. 3.12 Timer Processing

POINT

When the preset value is "0", it is regarded as infinite and the timer does not time out.

(2) Present value update timing and accuracy in direct mode

(a) Timer accuracy depends on the timer and scan time.

Timer Type	Scan Time T	Accuracy
10 ms	T < 10 ms	+2 scan time to −10 ms
10 ms	T ≥ 10 ms	+2 scan time to −1 scan time
100 ms, 100 ms retentive	T < 100 ms	+2 scan time to −100 ms
100 ms, 100 ms retentive	T ≥ 100 ms	+2 scan to −1 scan time

(b) The following example indicates the present value update timing and accuracy with a 10 ms timer used in the program of 10 ms or more scan time.

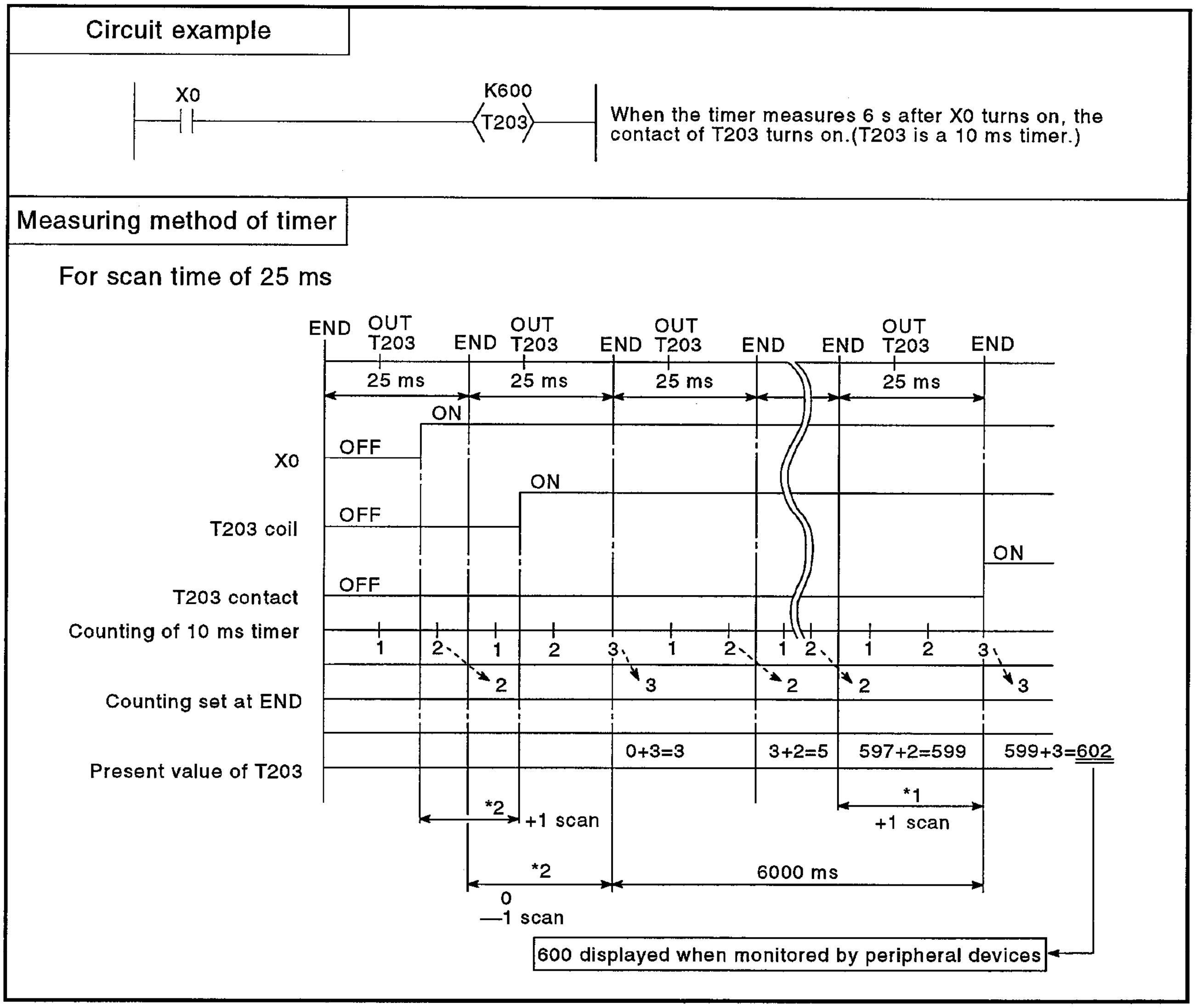


Fig. 3.13 Timer Timing

T203 time-out period includes the following errors:

*1: 10 ms timer error (+1 scan time)

*2: Error depending on timing of timer input continuity and location of the [OUT] T[ ] instruction in program (±1 scan time)

Accuracy is therefore $^{+2\text{ scan time}}_{-1\text{ scan time}}$ ($^{+0.05}_{-0.025}$ seconds in Fig. 3.13).

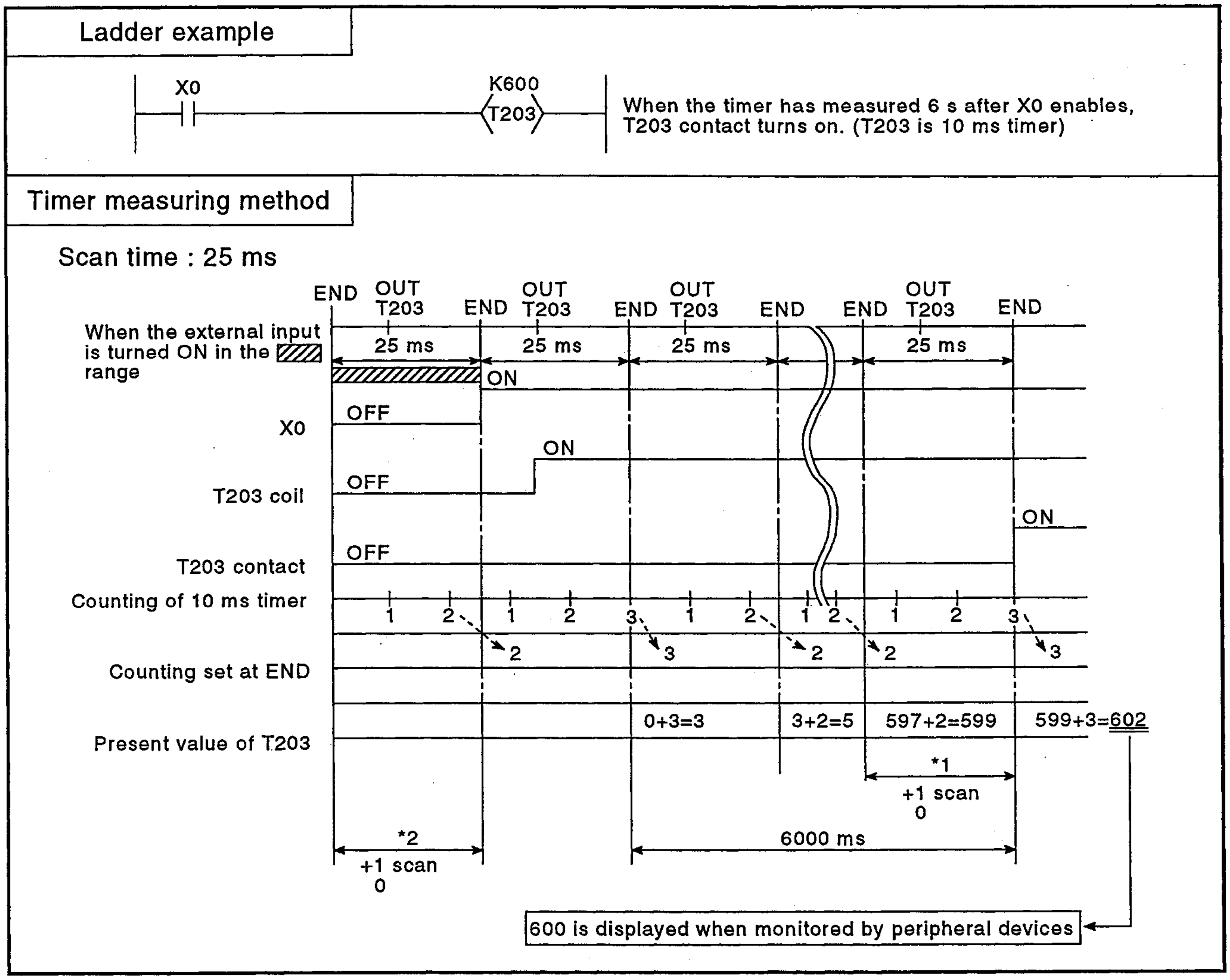
3) Contact status is updated only after the [END] instruction is processed regardless of the timer coil status during any scan.

(3) Update timing and accuracy in refresh mode

(a) Timer accuracy depends on the timer and scan time.

Timer Type	Scan Time T	Accuracy
10 ms	T < 10 ms	+2 scan time to –10 ms
10 ms	T ≥ 10 ms	+2 scan time to –1 scan time
100 ms, 100 ms retentive	T < 100 ms	+2 scan time to –100 ms
100 ms, 100 ms retentive	T ≥ 100 ms	+2 scan to –1 scan time

(b) The following example indicates the present value update timing and accuracy by using a 10 ms timer in a program of 10 ms or more scan time.



When the timer has measured 6 s after X0 enables, T203 contact turns on. (T203 is 10 ms timer)

Fig. 3.14 Timer Timing

T203 time-out period includes the following errors:
*1 : 10 ms timer error (+1 scan time)
*2 : Error depending on timing of timer input continuity and location of the [OUT] T[⁐] instruction in program (+1 scan time)

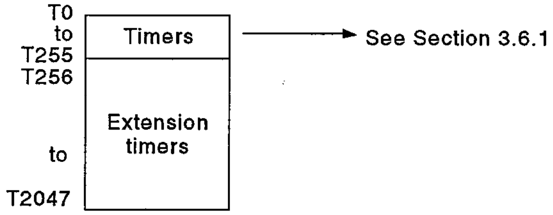
Accuracy is therefore +2 scan time (+0.05 s in Fig. 2.7)

(c) Contact status is updated only after the [END] instruction is processed, regardless of the timer coil status during any scan.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	x	x	x	o	o	o	x	x	x
Remark											

3.6.3 Extension timers

The AnA, A2AS, or AnU is provided with 2048 timers from T0 to T2047. Timers T256 to T2047 are allocated to extension timers.
Extension timers can be set as 100 ms, 10 ms, and 100 ms retentive timers.
Setting values for the extension timers are set indirectly with data registers, link registers, and file registers.



(1) Setting of extension timers
Use parameter setting to perform the following settings.

(a) Number of timers to be used
Set the number of timers to 257 or over.
Timers T256 and over are set for extension timers.
The default value is 256. Up to 2048 can be set.

(b) Purpose of extension timers
Set the extension timers for 100 ms, 10 ms, or 100 ms retentive timers.

(c) Set value devices (data registers, link registers, file registers)
Set value devices are used to store set values for extension timers.
The set value devices begin with a designated device number and are used through to the last device that corresponds to the last extension timer.

(2) Programming of extension timers

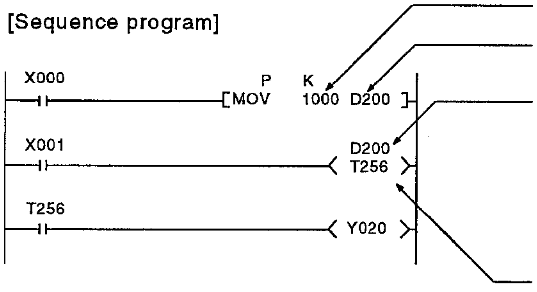
To use extension timers with a sequence program, use the [OUT] T□ instruction. (It is not necessary to key in setting value and devices for setting values.)

When an [OUT] T□ instruction is entered, the set value device number which corresponds to the designated timer number is displayed on the screen.

Example

To set the number of timers to be used at 512, all the extension timers to the 100 ms timers, and the set value device to D200:

[Sequence program]



Set value of T256

Set value device for T256

Set value device
Since they are set with parameters, it is not necessary to input them during programming.
Example) By inputting with the GPP function [⊣⊢] [T] [2] [5] [6] [GO], a device for setting T256 (D200) will be automatically displayed.

Extension timer

(3) Precautions

(a) To use extension timers, set the number of timers to 257 or over. The timer numbers which are not set in parameters cannot be written to the sequence program using contact commands or [OUT] T□ instructions.
The timer numbers which are not set in parameters can be used as data registers.

(b) To change the number of timers by parameter setting, do not set a number smaller than that designated by a contact command or an [OUT] T□ instruction.
If a timer number designated by an [OUT] T□ instruction is larger than the last timer number designated by parameter setting, measuring of the present value and turning ON/OFF of contacts is disabled.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	x	x	x	x	x	o	x	x	x
Remark											

3.6.4 1ms timer

With QCPU-A, a 1ms timer can be used in addition to the conventional high-speed timer (10ms) and low-speed timer (100ms).

(1) Usage

Adding "ZHTIME", a 1ms timer setting instruction, in a program enables the use of a 1ms timer. (The ZHTIME instruction must be written in the main program.)

The ZHTIME instruction is checked at startup and at switching from STOP to RUN. When this instruction exists in the main program, the 1ms timer can be used.

If the ZHTIME instruction does not exist in the main program, only the 100ms/10ms timer can be used, and the 1ms timer is disabled.

The number of occupied points is set as the total points of the 100ms timer, 10ms timer, retentive timer, and 1ms timer.

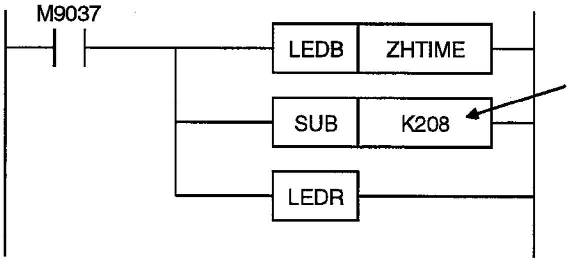
The area for the 1ms timer is reserved following that of the retentive timer.

Consequently, the constant specified with the ZHTIME instruction is designated as the device number following that of the retentive timer specified by parameters in the unit of 16 points.

(2) Use example of the ZHTIME instruction

The use example of the ZHTIME instruction is shown below.

Example) When the timer in 1ms is set at T208 and later:



Designate the device in the unit of 16 points.

(3) Accuracy of 1ms timer

The following table shows the accuracy of 1ms timer.

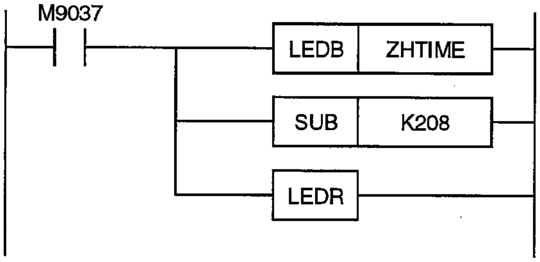
Timer type	Scan time	Accuracy
1ms	T < 1ms	+ 2 scan time to - 1 ms
	T ≥ 1ms	+ 2 scan time to - 1 scan time

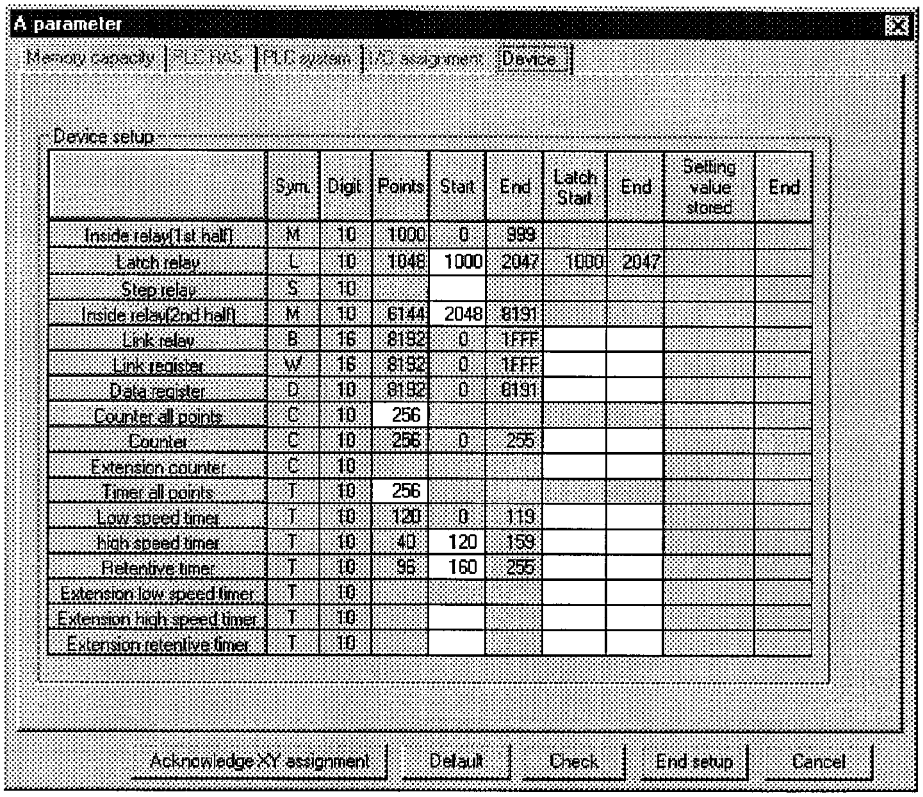
(4) Setting example

The following are the setting examples with and without the expansion timer:

(a) Setting example when the expansion timer is not used

Number of occupied points: 256 (100ms timer: 120 points, 10ms timer: 40 points, retentive timer: 48 points, 1ms timer: 48 points)



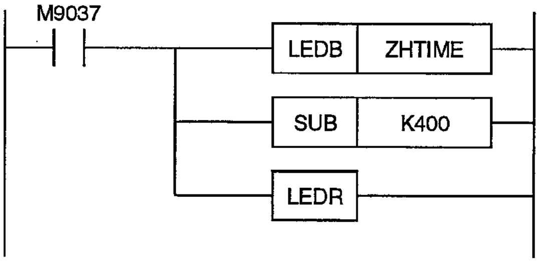


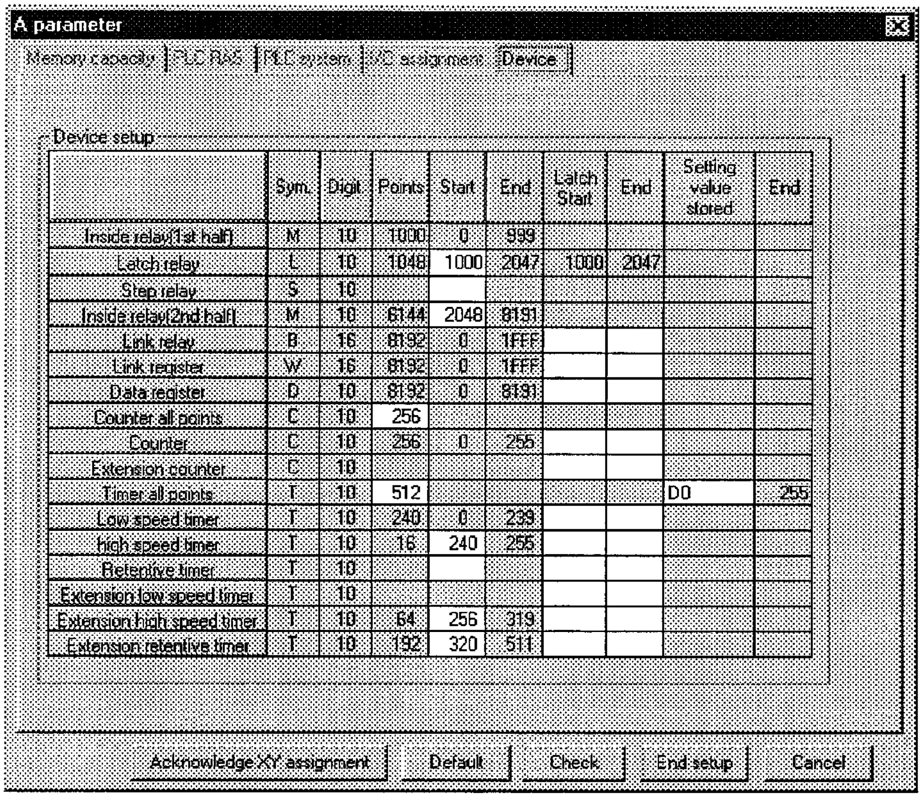
Device setup

	Sym.	Digit	Points	Start	End	Latch Start	End	Setting value stored	End
Inside relay(1st half)	M	10	1000	0	999				
Latch relay	L	10	1048	1000	2047	1000	2047		
Step relay	S	10							
Inside relay(2nd half)	M	10	6144	2048	8191				
Link relay	B	16	8192	0	1FFF				
Link register	W	16	8192	0	1FFF				
Data register	D	10	8192	0	8191				
Counter all points	C	10	256						
Counter	C	10	256	0	255				
Extension counter	C	10							
Timer all points	T	10	256						
Low speed timer	T	10	120	0	119				
high speed timer	T	10	40	120	159				
Retentive timer	T	10	96	160	255				
Extension low speed timer	T	10							
Extension high speed timer	T	10							
Extension retentive timer	T	10							

According to the setting above, the devices designated for the 100ms timer are T0 to T119, for the 10ms timer are T120 to T159, for the retentive timer are T160 to T207, and for the 1ms timer are T208 to T255.

(b) Setting example when the expansion timer is used
Number of occupied points: 512 (100ms timer: 240 points, 10ms timer: 80 points, retentive timer: 80 points, 1ms timer: 112 points)





	Sym.	Digit	Points	Start	End	Latch Start	End	Setting value stored	End
Inside relay(1st half)	M	10	1000	0	999				
Latch relay	L	10	1048	1000	2047	1000	2047		
Step relay	S	10							
Inside relay(2nd half)	M	10	6144	2048	8191				
Link relay	B	16	8192	0	1FFF				
Link register	W	16	8192	0	1FFF				
Data register	D	10	8192	0	8191				
Counter all points	C	10	256						
Counter	C	10	256	0	255				
Extension counter	C	10							
Timer all points	T	10	512					D0	255
Low speed timer	T	10	240	0	239				
high speed timer	T	10	16	240	255				
Retentive timer	T	10							
Extension low speed timer	T	10							
Extension high speed timer	T	10	64	256	319				
Extension retentive timer	T	10	192	320	511				

According to the setting above, the devices designated for the 100ms timer are T0 to T239, for the 10ms timer are T240 to T319, for the retentive timer are T320 to T399, and for the 1ms timer are T400 to T511.

POINTS

Note the following points to use the ZHTIME instruction.

(1) The ZHTIME instruction must be written in the main program.

(2) The ZHTIME instruction must be designated in the unit of 16 points.

(3) The number of occupied points designated in the timer setting by parameters should include those for the 1ms timer.

(4) When the range for the timer setting by parameters is between T256 and 2047, the initial device number to be used should be set at the item of the retentive timer berween T256 and 2047.
The 100ms timer should be used as the retentive timer.

Applicable CPU	All Types of CPUs
Remark	

3.7 Counters (C)

The PC CPU uses up count counters.
A up count counter counts out when the count value becomes equal with its preset value, and the contact of the counter is turned ON.

(1) Count processing

(a) A counter coil is turned ON by execution of the [OUT] C□ instruction. The present value is updated and the contact is turned ON/OFF after execution of the [END] instruction.

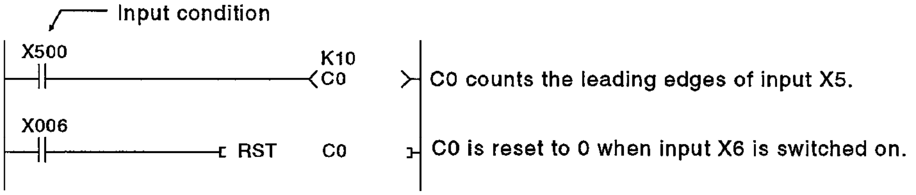
(b) The counter counts the leading edges of pulses driving its coil and counts once only when the coil is switched from off to on.

(2) Reset

(a) When the counter coil is switched on, the counter present value and contact status are updated after the [END] (FEND) instruction is executed.

(b) The count value is not cleared if the coil is switched off.
Use the [RST] C□ instruction to clear the count value and update the contact status.

Ladder example



C0 counts the leading edges of input X5.

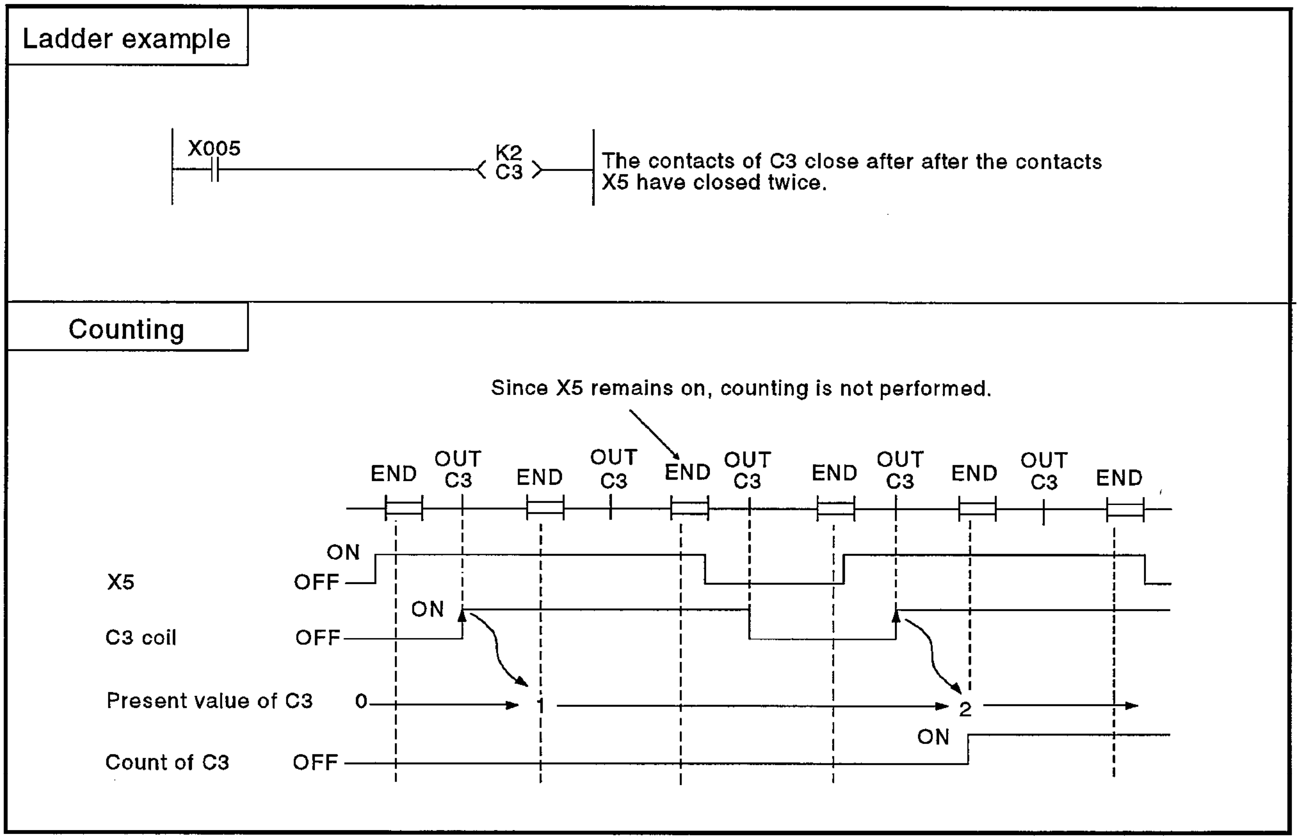
C0 is reset to 0 when input X6 is switched on.

Fig. 3.15 Count Ladder

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	o	x	x	x	x	o	x	o
Remark											

3.7.1 Count processing in direct mode

When input (X) signals are processed in direct mode, counting is executed at the leading edge of the input condition of the counter.



The contacts of C3 close after after the contacts X5 have closed twice.

Since X5 remains on, counting is not performed.

Fig. 3.16 Counter Counting

REMARK

For the maximum counting speed of the counter, refer to Section 3.7.3.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QUPU-A	A0J2H	A2C A52G	A73
	o	x	o	o	o	o	o	o	o	o	o
Remark											

3.7.2 Count processing in refresh mode

When input (X) signals are processed in refresh mode, counting is executed at the leading edge of the input condition which is received at refresh.

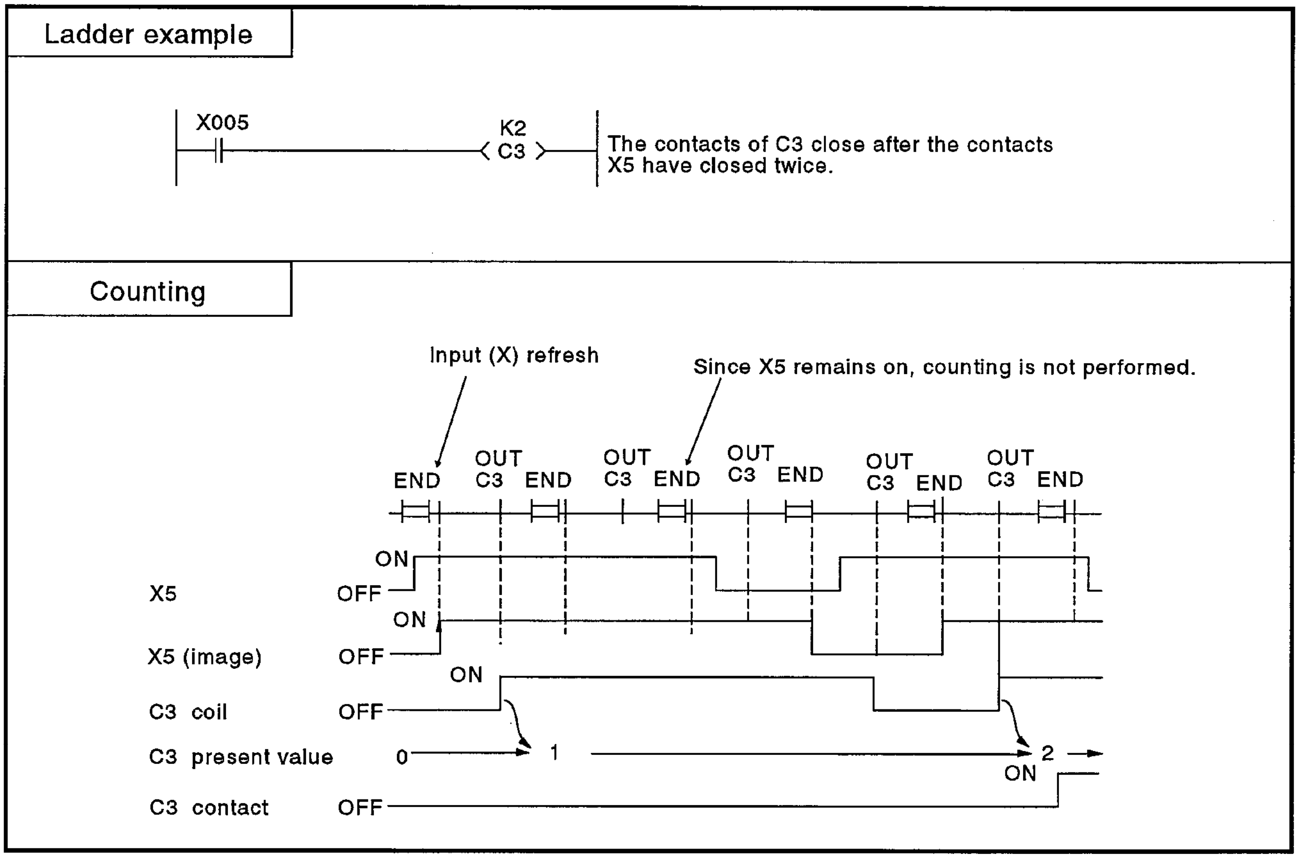


Fig. 3.17 Counter Counting

REMARK

For the maximum counting speed of the counter, refer to Section 3.7.3.

Applicable CPU	All Types of CPUs
Remark	

3.7.3 Maximum counting speed

The maximum counting speed of counters is determined by the length of scan time. Counting is possible only when the ON/OFF switching time of the input condition is longer than scan time.

$$\text{Maximum counting speed (Cmax.)} = \frac{n}{100} \times \frac{1}{ts} \text{(times/s)}$$

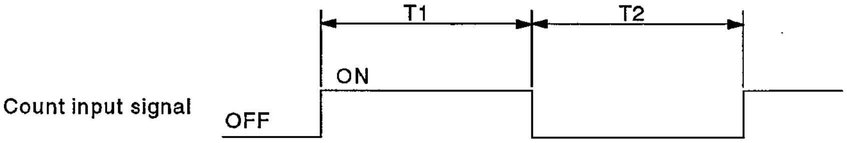
REMARK

where, n = duty (%)
ts = interrupt signal interval (s)

Duty is the ratio of the input signal's on time to off time as a percentage.

$$\text{If } T1 \leq T2 \quad n = \frac{T1}{T1 + T2} \times 100 \, [\%]$$

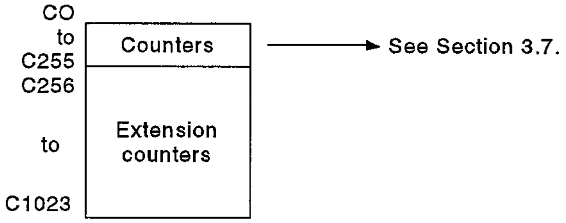
$$\text{If } T1 > T2 \quad n = \frac{T1}{T1 + T2} \times 100 \, [\%]$$



Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	x	x	x	o	o	o	x	x	x
Remark											

3.7.4 Extension counters

The AnA, A2AS, and AnU, QCPU-A is provided with 1024 counters from C0 to C1023.
Counters C256 to C1023 are allocated to extension counters.
Extension counters execute the same processings as those mentioned in Section 3.7.
Setting values for the extension counters are set indirectly with data registers, link registers, and file registers.



(1) Setting of extension counters
Use parameter setting to perform the following settings.

(a) Number of counters to be used

1) Set the number of counters to 257 or over.
Counters C256 and over are set for extension counters.

2) The default value is 256. Up to 1024 can be set.

(b) Set value devices (data registers, link registers, file registers)

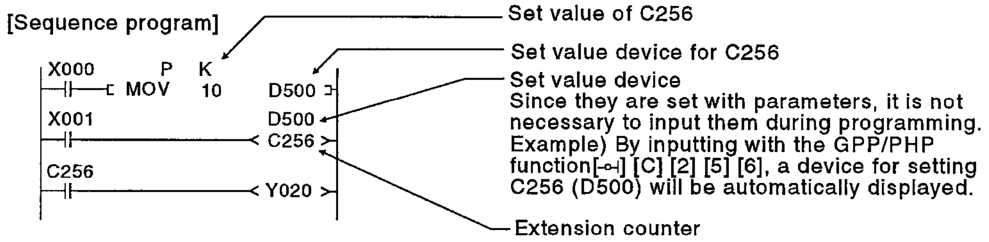
1) Set value devices are used to store set values for extension counters.

2) The set value devices begin with a designated device number and are used through to the last device which corresponds to the last extension counter.

(2) Programming of extension counters
To use extension counters with a sequence program, use the [OUT] C[ ] instruction. (It is not necessary to key in setting values and devices for setting values.)
When an [OUT] C[ ] instruction is entered, the set value device number which corresponds to the designated counter number is displayed on the screen.

Example

To set the number of counters to be used at 512, and the set value device to D500:


[Sequence program]
X000
P MOV
K 10
D500
X001
D500
C256
C256
Y020
Set value of C256
Set value device for C256
Set value device
Since they are set with parameters, it is not necessary to input them during programming.
Example) By inputting with the GPP/PHP function[-○-] [C] [2] [5] [6], a device for setting C256 (D500) will be automatically displayed.
Extension counter

(3) Precautions

(a) To use extension counters, set the number of counters to 257 or over.
The counter numbers which are not set in parameters cannot be written to the sequence program using contact commands or [OUT] C[ ] instructions.
The counter numbers which are not set in parameters can be used as data registers.

(b) To change the number of counters by parameter setting, do not set a number smaller than that designated by a contact command or an [OUT] C[ ] instruction.
If a counter number designated by an [OUT] C[ ] instruction is larger than the last counter number designated by parameter setting, measuring of the present value and turning ON/OFF of contacts is disabled.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	x	x	x	x	x	o	x	o
Remark											

3.8 Interrupt Counters (C)

The PC CPU can use two kinds of interrupt counters; those used within interrupt programs and those which count the number of interrupts.

3.8.1 Counters for interrupt programs

Counters used within interrupt programs are up count counters.
An up count counter counts out when the count value becomes equal with its preset value, and then, the contact of the counter is turned ON.

(1) Count processing

(a) A counter coil for an interrupt program is turned ON by execution of the [OUT] C□ instruction in an interrupt Program. The present value is updated and the contact is turned ON/OFF after execution of the IRET instruction.

(b) Counting is executed only when the input status of the [OUT] C□ instruction is turned from OFF to ON.
If the input status remains ON (or OFF), counting will not be executed.

(2) Reset

(a) The counted value is not cleared when the coil is turned OFF. Use the [RST] C□ instruction to clear the counted value and to turn OFF the contact.

(b) The present value and the contact of a counter is cleared when an [RST] instruction is executed to reset the counter.

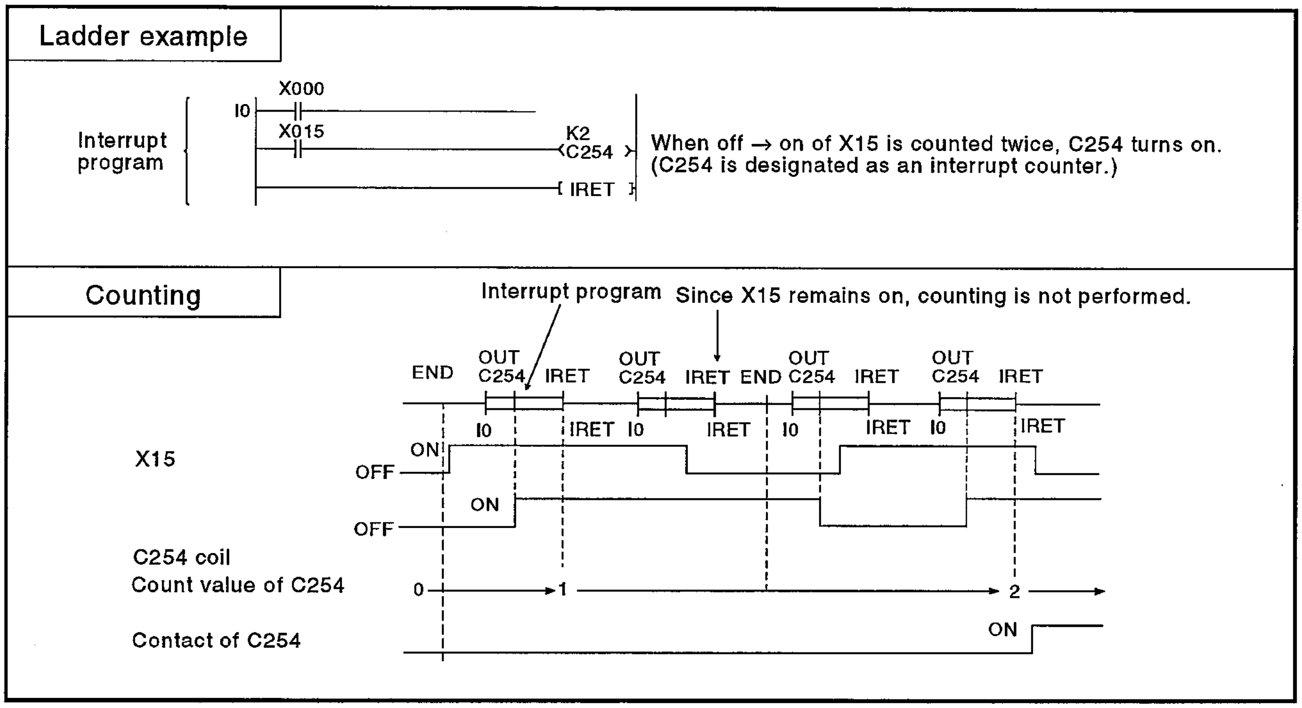


Fig. 3.18 Counting Method of Counter for Interruption

(3) Maximum counting speed

The maximum counting speed of the interrupt counter depends on the interrupt signal interval. Counting is only possible if the input condition is on for more than the interrupt signal interval.

$$\text{Maximum counting speed (Cmax.)} = \frac{n}{100} \times \frac{1}{ti} \text{ (times/s)}$$

REMARK

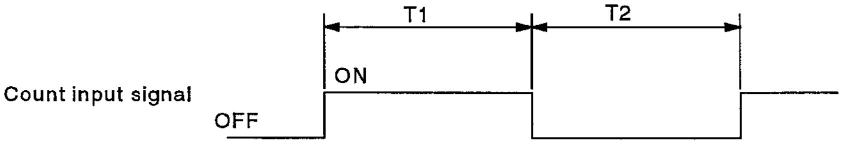
where, n = duty (%)

ti = interrupt signal interval (s)

Duty is the ratio of the input signal's on time to off time as a percentage.

$$\text{If } T1 \leq T2 \quad n = \frac{T1}{T1 + T2} \times 100\ [\%]$$

$$\text{If } T1 > T2 \quad n = \frac{T2}{T1 + T2} \times 100\ [\%]$$



Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	x	o	x	o	o	o	x	x	x
Remark											

3.8.2 Counters for counting the number of interrupts (Interrupt counters)

The interrupt counters are used to count the number of occurrences of interrupt.

(1) Count processing

(a) Counters C224 to C255 are used for the interrupt counters. When an interrupt occurs, the present value is updated and the contact is turned ON/OFF. The counter counts out when the number of interrupts becomes equal with its preset value, and then, the contact of the counter is turned ON. Contacts of the interrupt counters can be used freely in a sequence program.

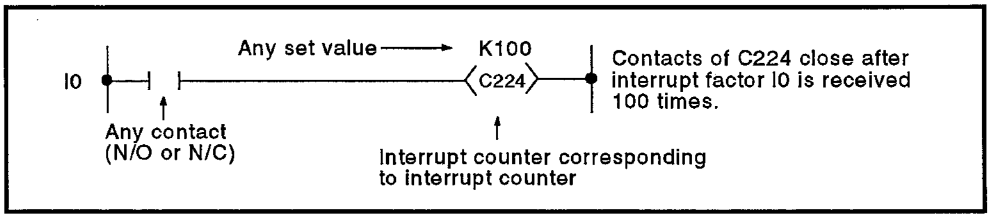
(b) The table below shows correspondence of interrupt pointers to interrupt counters. When an interrupt occurs at an interrupt pointer number, its corresponding counter executes counting. Refer to Section 3.16 for details of interrupt pointers.

Interrupt Pointer	Interrupt Counter	Interrupt Pointer	Interrupt Counter	Interrupt Pointer	Interrupt Counter	Interrupt Pointer	Interrupt Counter
I0	C224	I8	C232	I16	C240	I24	C248
I1	C225	I9	C233	I17	C241	I25	C249
I2	C226	I10	C234	I18	C242	I26	C250
I3	C227	I11	C235	I19	C243	I27	C251
I4	C228	I12	C236	I20	C244	I28	C252
I5	C229	I13	C237	I21	C245	I29	C253
I6	C230	I14	C238	I22	C246	I30	C254
I7	C231	I15	C239	I23	C247	I31	C255

(2) Setting of the interrupt counters
Perform the following settings to use counters C224 to C255 as interrupt counters.

1) Set interrupt pointers in units of point by parameter setting.

2) Insert the following program between the [FEND] instruction and the [END] instruction.


I0
Any set value
K100
C224
Any contact
(N/O or N/C)
Interrupt counter corresponding
to interrupt counter
Contacts of C224 close after
interrupt factor I0 is received
100 times.

3) Interrupt should be kept enabled by executing the [EI] instruction at the program head.

(3) Precautions

(a) It is impossible to execute counting with an interrupt counter and an interrupt program using one interrupt pointer.
An interrupt pointer set for an interrupt counter cannot be used to execute an interrupt program.

(b) If an interrupt occurs when any of the following processings is being executed count processing waits until the execution is completed. Count processing starts when the execution of each processing is completed.
If the same interrupt occurs again when any processing is being executed, count processing is executed only once.

- A sequence program instruction is being executed.
- The interrupt-disable section in the [END] processing is being executed. (Timer/counter update, etc.)
- An interrupt program is being executed.

(c) The maximum counting speed can be calculated using the longest processing time of the following:

- Instruction with the longest processing time present in the program
- END processing interrupt disable area ..... max. 2 ms
- Interrupt program processing time

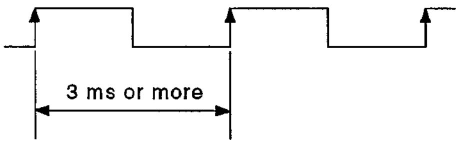
$$\text{Max. counting speed} = \frac{1}{\text{(max. processing time of the above)} + (500\ \mu\text{s} \times \text{number of interrupt counters})}\ \text{(PPS)}$$

Example:

The [End] processing time is 2 ms (0.002 s). If the max. instruction processing time is 0.3 ms a program is not written during run, there is no interrupt program, and two interrupt counters are used.

$$\text{Max. counting speed} = \frac{1}{0.002 + 0.0005 \times 2} \approx 333\ \text{(PPS)}$$

Hence, the highest speed pulse train which may be reliably read by the A3HCPU, with the above conditions is 333 pulses/s.



(d) If a large number of interrupt counters are used, the processing time of the sequence program becomes long, and a "WDT ERROR" sometimes occurs.
Decrease the number of interrupt counters or slow down the input pulse counting speed to avoid this error.

(e) The interrupt counters continue counting after count out.
Use an [RST] C□ instruction before the [FEND] instruction in the sequence program to reset the counter.

(f) Counter value of the interrupt counters can be read by use of the [MOV] instruction in the sequence program.

Applicable CPU	All Types of CPUs
Remark	

3.9 Data Registers (D)

(1) Data registers are used to store numeric data (-32768 to 32767 or 0000H to FFFFH) within the programmable controller.
Each data register consists of 16 bits which is the unit of data read and write.

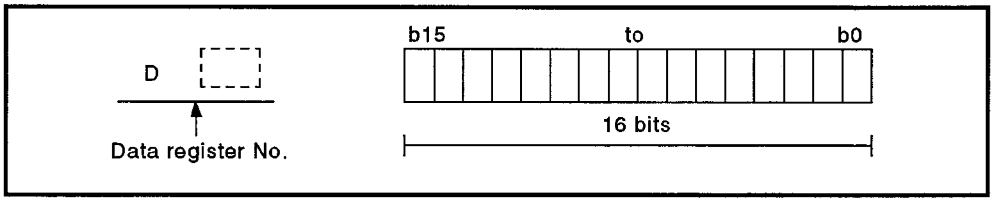


Fig. 3.19 Structure of a Data Register

(2) Use two data registers to handle 32-bit data.
The data register number designated by the 32-bit instruction holds the lower 16 bits and the designated data register number + 1 holds the higher 16 bits.

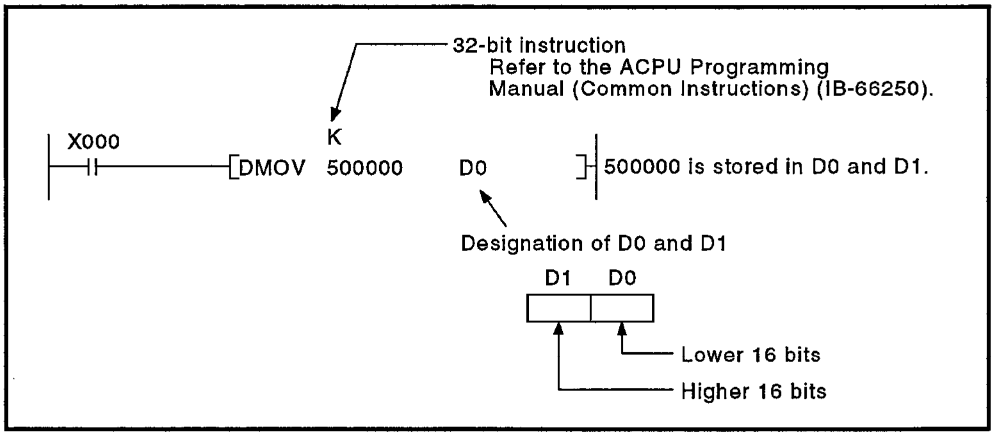

32-bit instruction
Refer to the ACPU Programming Manual (Common Instructions) (IB-66250).
X000
K
DMOV 500000 D0
500000 is stored in D0 and D1.
Designation of D0 and D1
D1
D0
Lower 16 bits
Higher 16 bits

Fig. 3.20 Programming of Data Registers when the 32-bit Instruction is Used

(3) Data stored in a register by use of the program is retained till it is replaced by a new data.

(4) If additional data registers are required, the registers mentioned below can be used as data registers.
- Unused timers (T) and counters (C)
- File registers (R) within the range set by parameters.
- Link registers (W) not used for data link.
- Index registers
- Accumulators

Applicable CPU	**All Types of CPUs**
Remark	

3.10 Link Registers (W)

(1) Link registers are used to store data for data link.
Each link register consists of 16 bits which is the unit of data read and write.

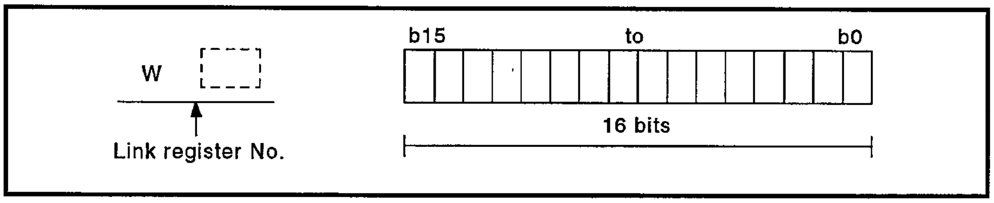


Fig. 3.21 Structure of a Link Register

(2) Use two link registers to handle 32-bit data.
The link register number designated by the 32-bit instruction holds the lower 16 bits and the designated link register number + 1 holds the higher 16 bits.

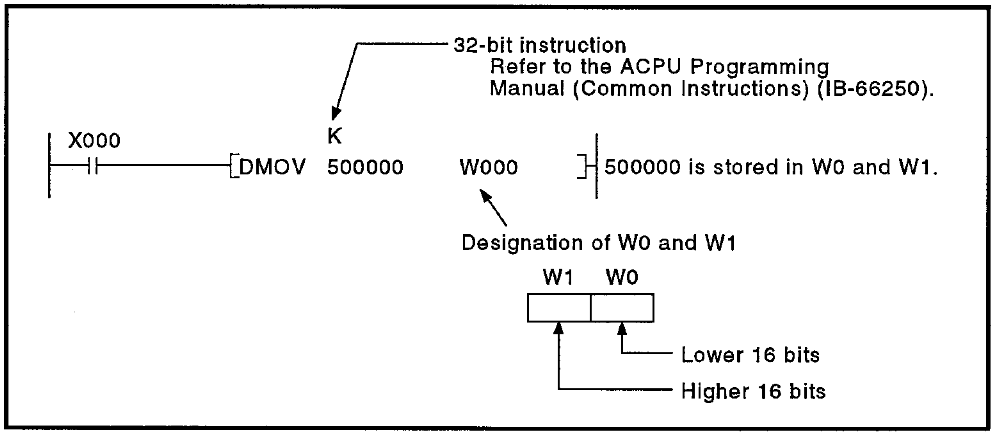

32-bit instruction
Refer to the ACPU Programming Manual (Common Instructions) (IB-66250).
X000
K
DMOV 500000 W000
500000 is stored in W0 and W1.
Designation of W0 and W1
W1 W0
Lower 16 bits
Higher 16 bits

Fig.3.22 Programming of Link Registers when the 32-bit Instruction is Used

(3) The data communication described below is possible by using link registers for data link.

(a) Data written to a station (master or local) can be read and used with other stations (local or master).
Using link registers, data communication is enabled from the master station to all local stations, from a local station to the master station, and between local stations.

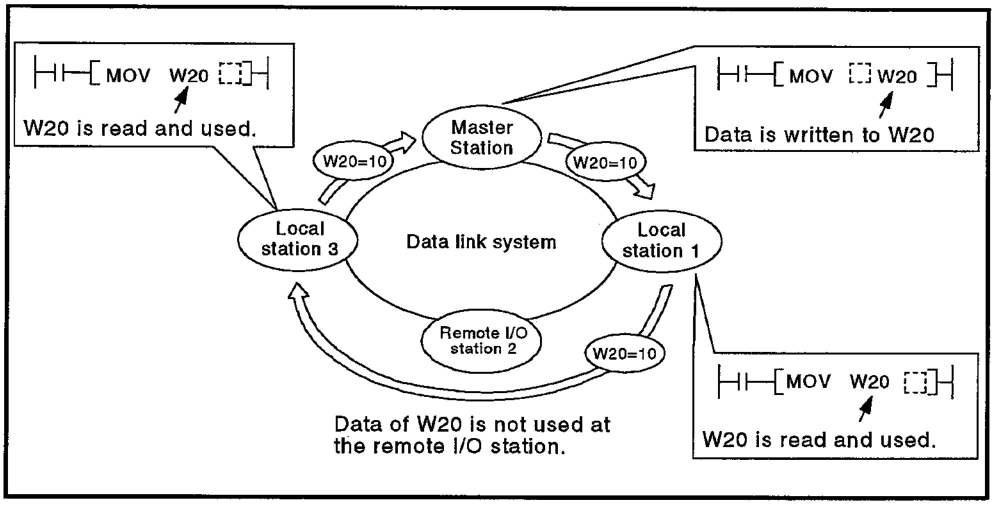


Fig. 3.23 Use of Link Registers with a Data Link System

(4) To use link registers for data link, it is necessary to set a link range (the range used for registers with each station) in the master station. The link register numbers not set within a link range can be used as data registers at each station.

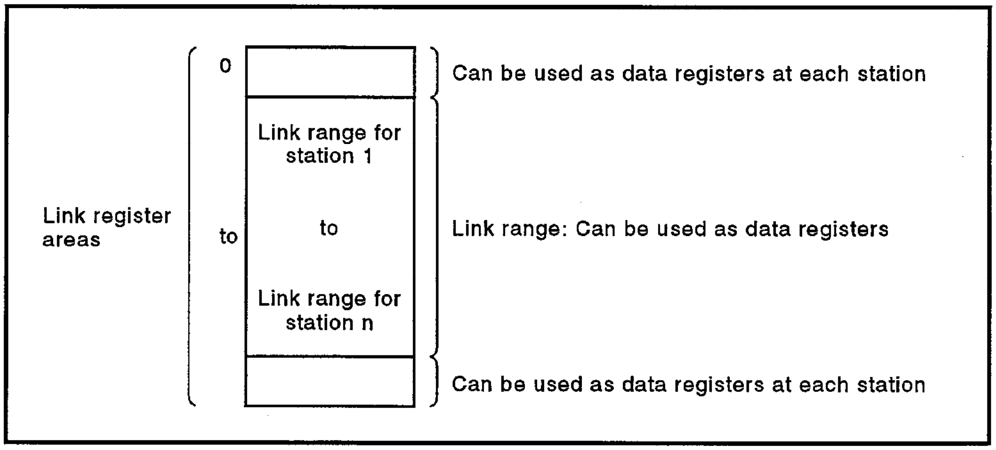

Link register areas
0
to
Link range for station 1
to
Link range for station n
Can be used as data registers at each station
Link range: Can be used as data registers
Can be used as data registers at each station

Fig. 3.24 Allocation of Link Registers

REMARKS

(1) For the data link system and link range setting, refer to the MELSECNET (II) Data Link System Reference Manual (IB-66263) and MELSECNET, MELSECNET/B Data Link System Reference Manual (IB-66350).

(2) To use link registers with remote I/O stations, use the RFRP/RTOP instructions at the master station.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	△*1	o	o	o	o	o	o	o	o	o
Remark	*1 : A1 and A1N are unusable.										

3.11 File Registers (R)

3.11.1 File registers

(1) File registers are used in an extended area of data registers.
The user memory area and sequencer CPU in the memory cassette is used for file registers.

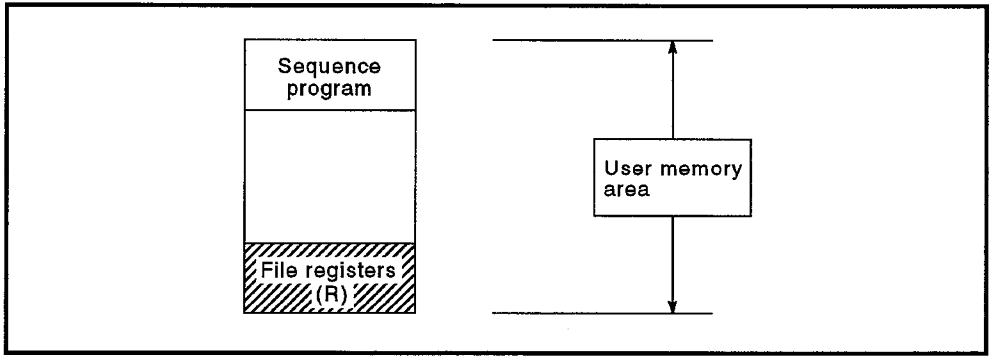


Fig. 3.25 File Register Area

(2) Each file register consists of 16 bits which is the unit of data read and write.

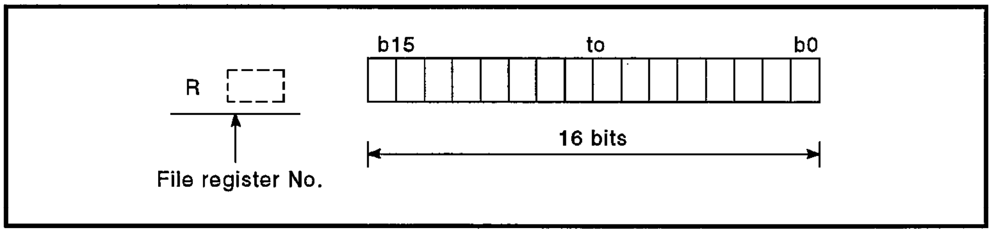


Fig. 3.26 Structure of a File Register

(3) Use two file registers to handle a 32-bit data.
The file register number designated by the 32-bit instruction holds the lower 16 bits and the designated file register number + 1 holds the higher 16 bits.

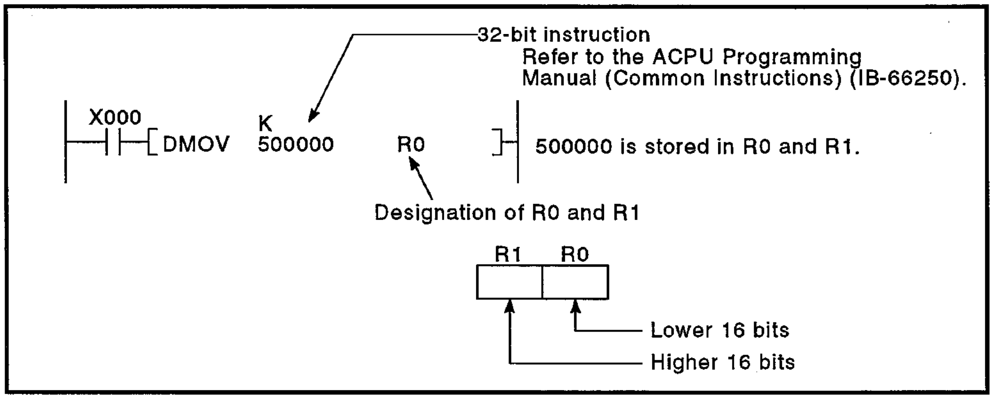

32-bit instruction
Refer to the ACPU Programming Manual (Common Instructions) (IB-66250).
X000
DMOV
K
500000
R0
500000 is stored in R0 and R1.
Designation of R0 and R1
R1
R0
Lower 16 bits
Higher 16 bits

Fig. 3.27 Programming of File Registers when the 32-bit Instruction is Used

(4) Clearing file register data

(a) Data stored in a file register is not cleared when the power supply is turned ON, or when the RESET switch is moved to "RESET" or "LATCH CLEAR".

(b) Use the FMOV(P) instruction and write "0" to clear a file register.

Example To clear file registers R0 to R1023 (1K):

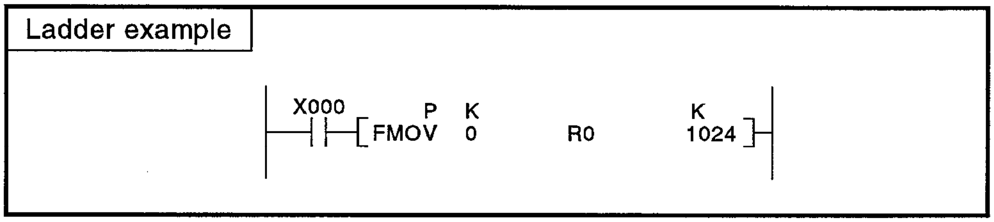
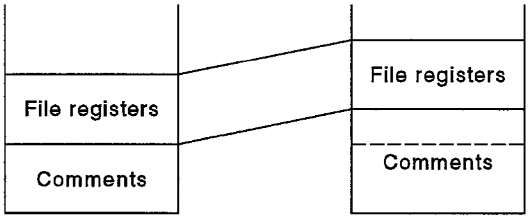


Fig. 3.28 A Ladder to Clear File Registers

(5) Note on setting the comment capacity

File registers are set in the area before the comments.
This means that if the comment capacity is changed, the location of the file registers changes and the stored data values will be incorrect.
Before changing the comment capacity, write the file register data to a peripheral device, then write this data back to the CPU after the comment capacity has been changed.



Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	x	x	x	o	o	o	x	x	x
Remark											

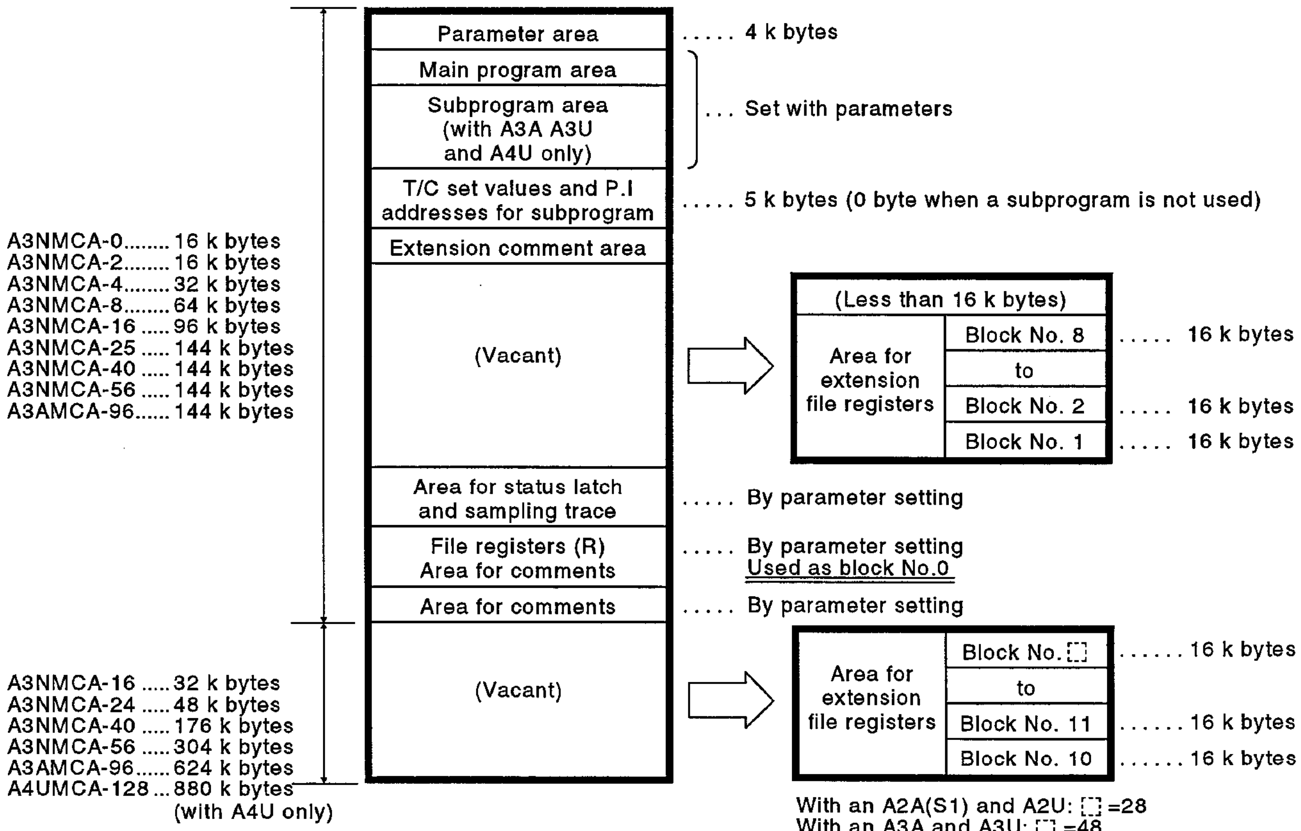
3.11.2 Extension file registers

Extension file registers are for extended use of data registers and allocated automatically to vacant (unused) areas in the memory cassette.
A total of 8192 points are set as 1 block. Up to 48 blocks can be used.
The number of blocks varies with type of memory cassette to be used and capacity of vacant memory areas and the sequencer CPU.

> **POINT**
>
> Extension file register can be used only when the memory capacity for registers is set with parameters.

(1) Allocation of extension file registers

(a) Extension file registers are allocated automatically to vacant areas in a memory cassette as shown below.
Vacant areas are divided into units of 16 k bytes, and each area is attached with a block number.
If capacity of a vacant area is less than 16 k bytes, such area cannot be used as an extension file register.


Parameter area
..... 4 k bytes
Main program area
Subprogram area (with A3A A3U and A4U only)
... Set with parameters
T/C set values and P.I addresses for subprogram
..... 5 k bytes (0 byte when a subprogram is not used)
Extension comment area
A3NMCA-0........ 16 k bytes
A3NMCA-2........ 16 k bytes
A3NMCA-4........ 32 k bytes
A3NMCA-8........ 64 k bytes
A3NMCA-16 ..... 96 k bytes
A3NMCA-25 ..... 144 k bytes
A3NMCA-40 ..... 144 k bytes
A3NMCA-56 ..... 144 k bytes
A3AMCA-96...... 144 k bytes
(Vacant)
(Less than 16 k bytes)
Area for extension file registers
Block No. 8
to
Block No. 2
Block No. 1
..... 16 k bytes
..... 16 k bytes
..... 16 k bytes
Area for status latch and sampling trace
..... By parameter setting
File registers (R) Area for comments
..... By parameter setting Used as block No.0
Area for comments
..... By parameter setting
A3NMCA-16 ..... 32 k bytes
A3NMCA-24 ..... 48 k bytes
A3NMCA-40 ..... 176 k bytes
A3NMCA-56 ..... 304 k bytes
A3AMCA-96...... 624 k bytes
A4UMCA-128 ... 880 k bytes (with A4U only)
(Vacant)
Area for extension file registers
Block No. □
to
Block No. 11
Block No. 10
...... 16 k bytes
...... 16 k bytes
...... 16 k bytes

With an A2A(S1) and A2U: □ =28
With an A3A and A3U: □ =48
With an A4U: □ =64

(b) The block numbers usable for extension file registers among block number 1 to 8 can be obtained as follows based on the capacity of vacant areas.

Capacity of vacant areas (k bytes)/16 = N (Truncation)
Usable block numbers: 0 to N

(c) The block numbers usable for extension file registers among block numbers 10 to 64 vary with type of memory cassette to be used.

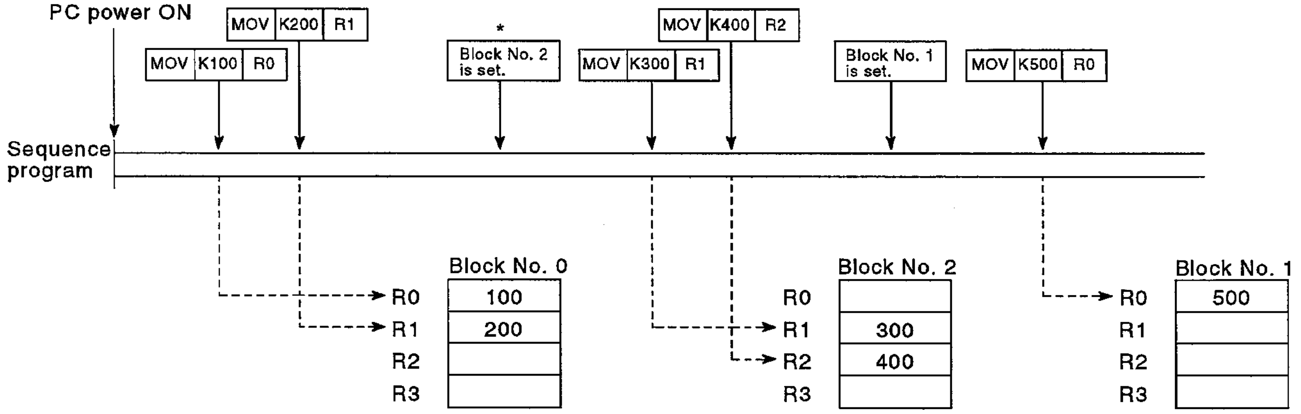
A3NMCA-16	10, 11
A3NMCA-24	10 to 12
A3NMCA-40	10 to 20
A3NMCA-56	10 to 28
A3AMCA-96....................	10 to 48
A4UMCA-128	10 to 64

POINT

When a value outside the usable block number range is designated when an A3NMCA-16, 24, 40, 56, A3AMCA-96, or A4UMCA-128 is used, the following will occur:

	Though No Error is Generated, Read Out Value will be Indefinite.	"OPERATION ERROR"
A3NMCA-16	—	12 and after
A3NMCA-24	13 to 28	29 and after
A3NMCA-40	21 to 28	29 and after
A3NMCA-56	—	29 and after
A3AMCA-96	—	49 and after
A4UMCA-128	—	65 and after

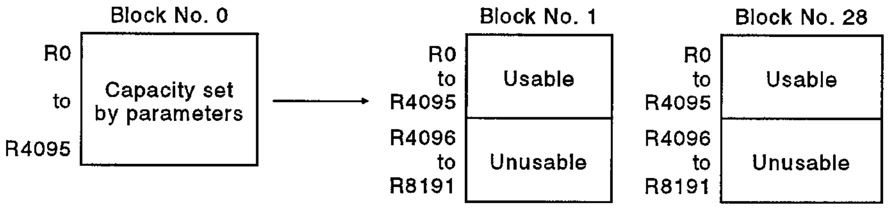
(2) Use of extension file registers
When data read/write is executed with extension file registers, use them as file registers (R).
Set the device numbers of extension file register with R0 to R8191 similarly as file registers are set.
Designate block numbers of extension file registers and file registers to be used.
To use the file registers set with parameters, designate block No. 0.
Block No. 0 is set when the PC CPU is powered on.



POINT

The device number usable for data read/write among the extension file register block numbers 1 through 64 are within the range of the file registers (block No. 0) set with parameters.

Example) If the file register area is set at 4 k with parameters:



If index registers (Z, V) are used for index qualification of file register (R) device numbers, read/write can be done with all device numbers from 0 to 8191.

(3) Precautions on the use of extension file registers

(a) To use extension file registers, set the file register capacity with parameters.
If the file register capacity is not set, extension file registers cannot be used.

(b) If sampling trace or status latch is executed when extension file registers are used, the file registers of which data is to be read are selected as follows.

- At sampling trace:
  When a step number is designated: The file registers of the block numbers designated when the END instruction is executed.
  When sampling time is designated: File registers vary with timing of sampling.
- At status latch:
  The file registers of the block numbers designated when the STATUS LATCH instruction is executed.

*: To set block No. 2, use the "RSET" instruction.
For details on the RSET instruction, refer to the Programming Manual (Dedicated Instructions) for the AnSHCPU/AnACPU/AnCCPU/QCPU-A (A Mode).

Applicable CPU	**All Types of CPUs**
Remark	

3.12 Accumulators (A)

(1) Accumulators are used to store the operation results of basic and application instructions.
For the basic and application instructions of which operation results are stored in accumulators, refer to the list of instructions in the ACPU Programming Manual (Common Instructions) (IB-66250).

(2) Each accumulator consists of 16 bits which is the unit of data read and write.

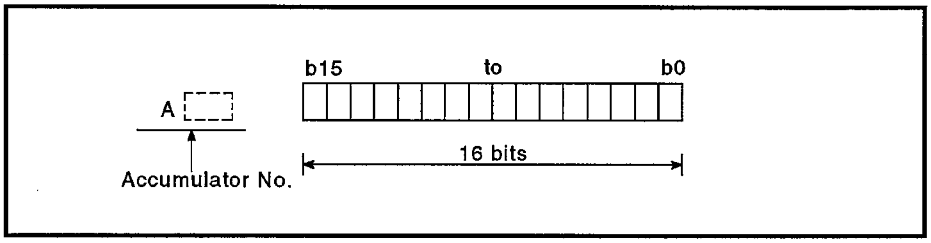


Fig. 3.29 Structure of an Accumulator

(3) Two accumulators (A0, A1) are provided.
Use two accumulators to handle 32-bit data.
Accumulator A0, when used with a 32-bit instruction, holds the lower 16 bits and A1 holds the higher 16 bits.

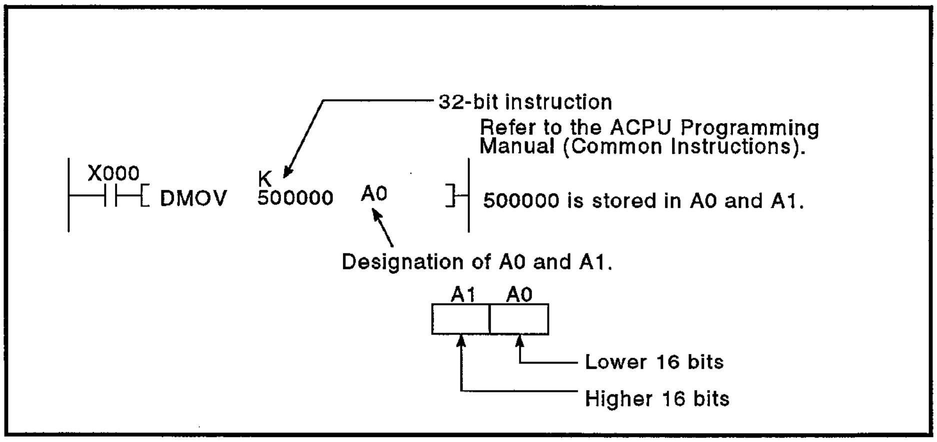

32-bit instruction
Refer to the ACPU Programming Manual (Common Instructions).
X000
DMOV
K
500000
A0
500000 is stored in A0 and A1.
Designation of A0 and A1.
A1
A0
Lower 16 bits
Higher 16 bits

Fig. 3.30 Programming of Accumulators when the 32-bit Instruction is Used

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	o	o	x	x	x	o	o	o
Remark											

3.13 Index Registers

3.13.1 Index registers (Z, V)

(1) Index registers are used to designate indirectly the devices (X, Y, M, L, S, B, F, T, C, D, W, R, K, H, and P) which are used with basic and application instructions.
This indirect designation of devices is disabled when those instructions which use bit devices (X, Y, M, L, S, B, F, T, and C) in units of each device for contacts, coils, etc. are used.

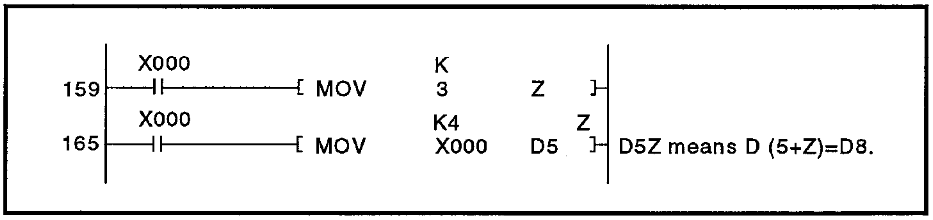


Fig. 3.31 Example of Indirect Designation Using Index Register

(2) Each index register consists of 16 bits which is the unit of data read and write.

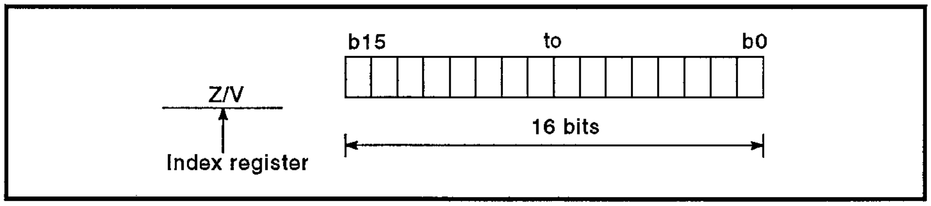


Fig. 3.32 Structure of an Index Register

(3) Two index registers (Z, V) are provided.
Index register Z, when used with a 32-bit instruction, holds the lower 16 bits and V holds the higher 16 bits.
A 32-bit instruction cannot be used to designate V.

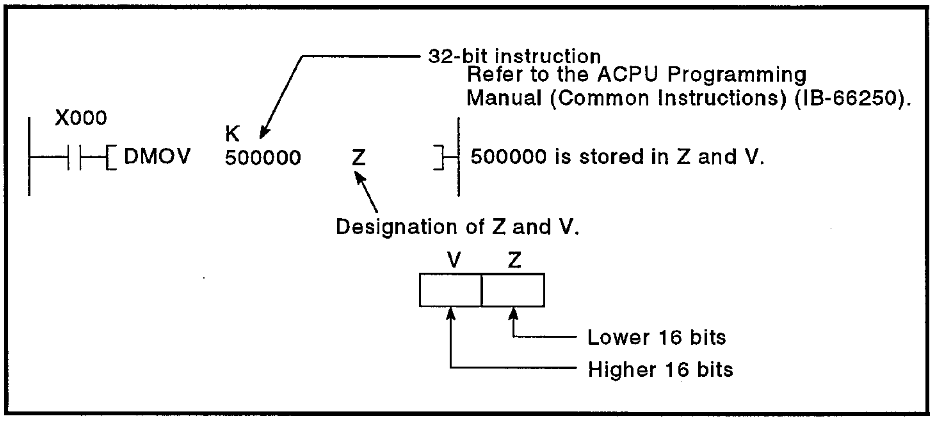
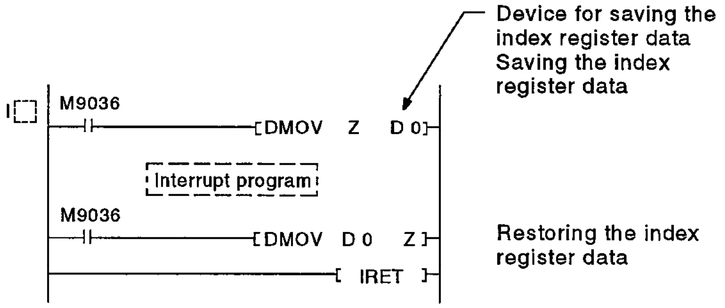

32-bit instruction
Refer to the ACPU Programming Manual (Common Instructions) (IB-66250).
X000
DMOV
K
500000
Z
500000 is stored in Z and V.
Designation of Z and V.
V
Z
Lower 16 bits
Higher 16 bits

Fig. 3.33 Programming of Index Registers when a 32-bit Instruction is Used

POINT

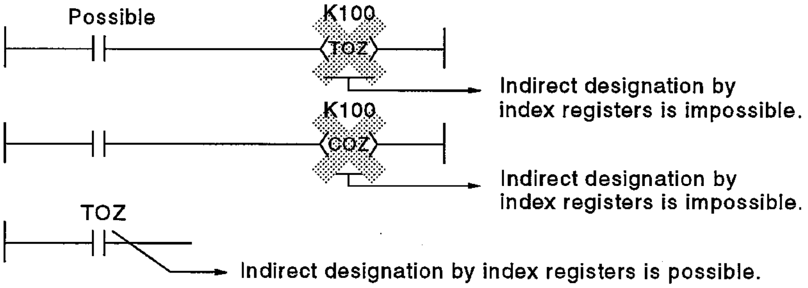
The CPU modules other than the AnA, A2AS, and AnU, QCPU-A do not save the index register data before the execution of an interrupt program.
When index register data has been changed during an interrupt program execution and when the index register data needs to be changed back to the state before the interrupt program execution at the completion of the interrupt program execution (IRET execution), make a program as follows:


Device for saving the index register data
Saving the index register data
I□
M9036
DMOV Z D 0
Interrupt program
M9036
DMOV D 0 Z
Restoring the index register data
IRET

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	x	x	x	o	o	o	x	x	x
Remark											

3.13.2 Index registers (Zn(Z, Z1 to Z6), Vn(V, V1 to V6))

(1) index registers are used to designate indirectly the devices (X, Y, M, L, S, B, F, T, C, D, W, R, K, H, and P) which are used with basic and application instructions.
This indirect designation of devices is enabled also when those instructions which use bit devices (X, Y, M, L, S, B, F, T, and C) in units of each device for contacts, coils, etc. are used.
However, when devices T and C are used with OUT T[ ] or OUT C[ ], indirect designation by using index registers is impossible.


Possible
K100
TOZ
Indirect designation by index registers is impossible.
K100
COZ
Indirect designation by index registers is impossible.
TOZ
Indirect designation by index registers is possible.

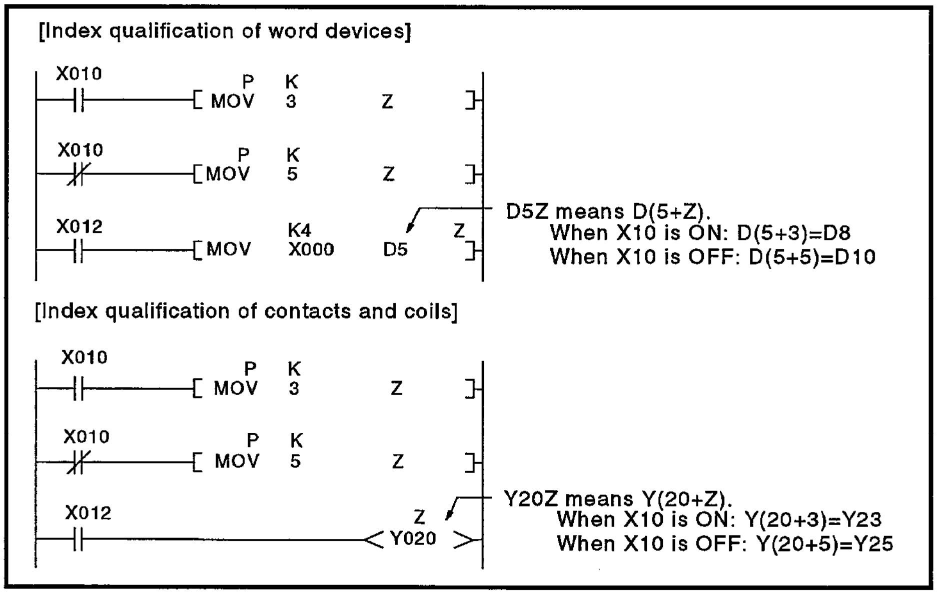

[Index qualification of word devices]
X010
P MOV K3 Z
X010
P MOV K5 Z
X012
MOV K4X000 D5 Z
D5Z means D(5+Z).
When X10 is ON: D(5+3)=D8
When X10 is OFF: D(5+5)=D10
[Index qualification of contacts and coils]
X010
P MOV K3 Z
X010
P MOV K5 Z
X012
Y020 Z
Y20Z means Y(20+Z).
When X10 is ON: Y(20+3)=Y23
When X10 is OFF: Y(20+5)=Y25

Fig. 3.34 Example of Indirect Setting with Index Registers

(2) Each index register consists of 16 bits which is the unit of data read and write.

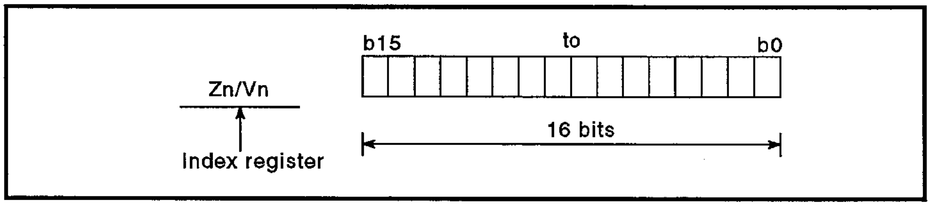


Fig. 3.35 Structure of an Index Register

(a) Two kinds of index registers Zn and Vn are provided. When a 32-bit instruction is used, a Zn holds the lower 16 bits and a Vn holds the higher 16 bits.
The following pairs of index registers are used with a 32-bit instruction.

1) Z and V
2) Z1 and V1
3) Z2 and V2
4) Z3 and V3
5) Z4 and V4
6) Z5 and V5
7) Z6 and V6

Since the Zn is regarded as the lower 16-bit device, the Vn cannot be used with a 32-bit instruction.

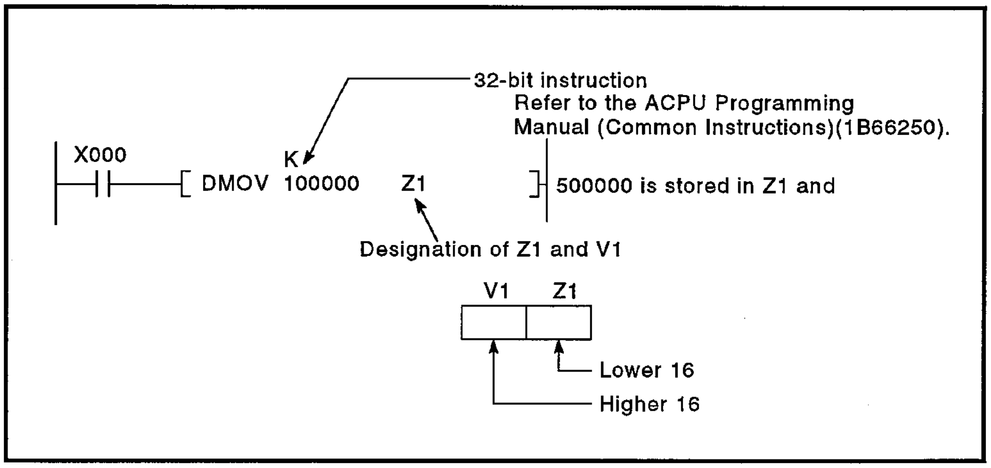
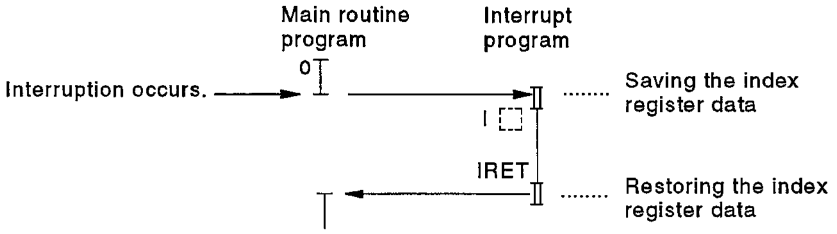

32-bit instruction
Refer to the ACPU Programming Manual (Common Instructions)(1B66250).
X000
K
DMOV 100000 Z1
500000 is stored in Z1 and
Designation of Z1 and V1
V1 Z1
Lower 16
Higher 16

Fig. 3.36 Programming of Index Registers when a 32-bit Instruction is Used

POINT

The AnA, A2AS, and AnU, QCPU-A modules save the index register data before the execution of an interrupt program.
Saved index register data is restored after the interrupt program execution is completed (IRET execution).
When index register data has been changed during an interrupt program execution, the index register data returns to that before the interrupt program execution at the completion of the interrupt program execution.


Main routine program
Interrupt program
0
Interruption occurs.
I □
IRET
.... Saving the index register data
.... Restoring the index register data

Applicable CPU	All Types of CPUs
Remark	

3.14 Nesting (N)

(1) Nesting devices are used with the master control instructions.

(2) The master control instructions are used to open and close the bus so that switching of ladders may be executed efficiently by the sequence program.
Each master control sequence begins with an MC instruction and ends with an MCR instruction.
(Refer to the ACPU Programming Manual (Common Instructions) (IB-66250) for details.)

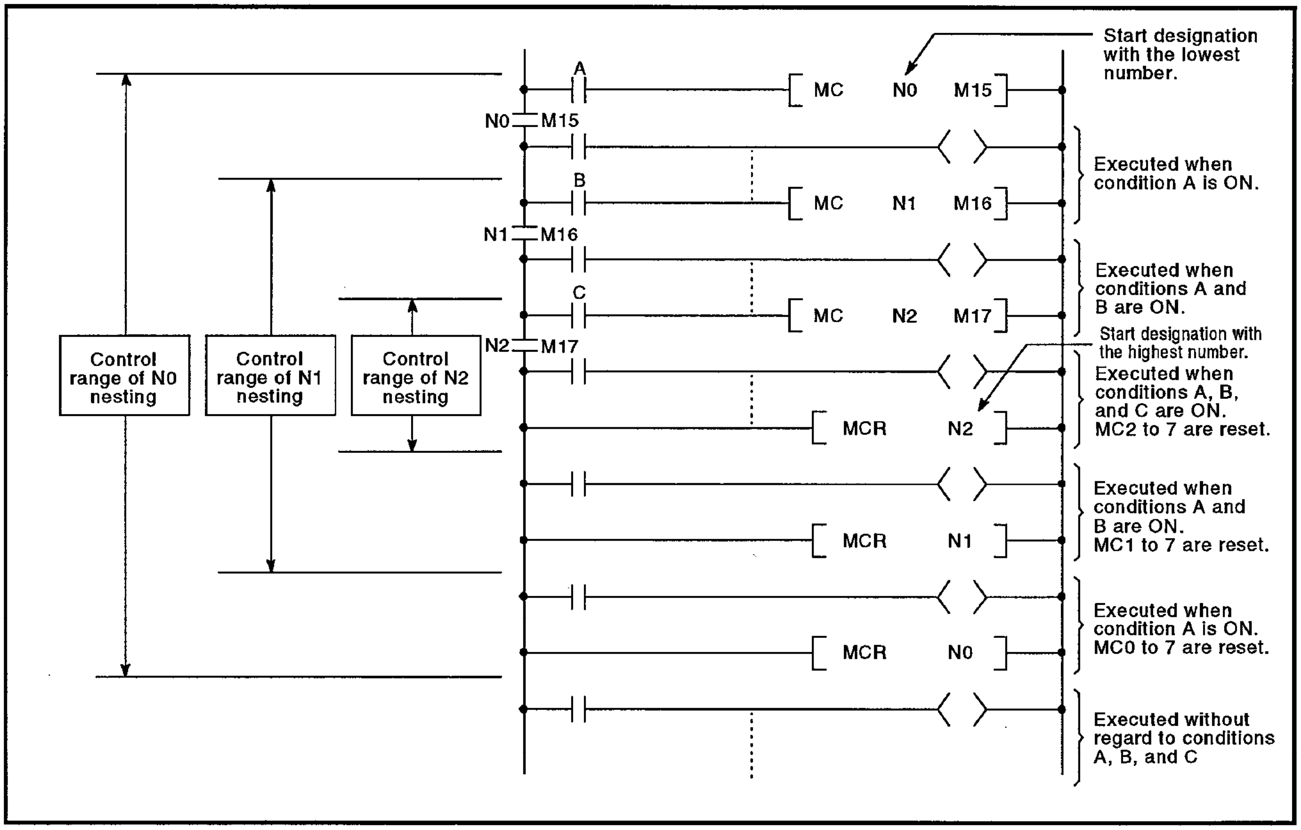

Start designation with the lowest number.
A
MC N0 M15
N0 M15
Executed when condition A is ON.
B
MC N1 M16
N1 M16
Executed when conditions A and B are ON.
C
MC N2 M17
Start designation with the highest number.
N2 M17
Executed when conditions A, B, and C are ON. MC2 to 7 are reset.
MCR N2
Control range of N0 nesting
Control range of N1 nesting
Control range of N2 nesting
Executed when conditions A and B are ON. MC1 to 7 are reset.
MCR N1
Executed when condition A is ON. MC0 to 7 are reset.
MCR N0
Executed without regard to conditions A, B, and C

Fig. 3.37 Nesting of Master Control

(3) When an MC instruction is OFF, the results of the operations from an MC to an MCR are as follows.

100 ms and 10 ms timers	Count value becomes 0. Coil and contact are turned OFF.
100 ms retentive timers and counters	Coils are turned OFF. Count value and contacts retain present status.
Devices used with an OUT instruction	All are turned OFF.
Devices used with SET, RST, SFT, basic, and application instructions	Present status is retained.

Applicable CPU	All Types of CPUs
Remark	

3.15 Pointers (P)

(1) Pointers are used with the jump instructions (CJ, SCJ, JMP, CALL) in two different ways as follows.

(a) Designation of the destination of jump (CJ, SCJ, JMP) and the head of destination (label).

(b) Designation of the destination of subroutine call (CALL, CALLP) and the head of the subroutine program (label).

(2) A label number cannot be used at more than one place. If used, an error will occur.

(3) P255 always designates the END step.

(a) Use P255 with a CJ, SCJ, or JMP instruction to jump to the END step.

(b) P255 cannot be used as a label.

(c) P255 cannot be used with the devices for the CALL(P) instruction.

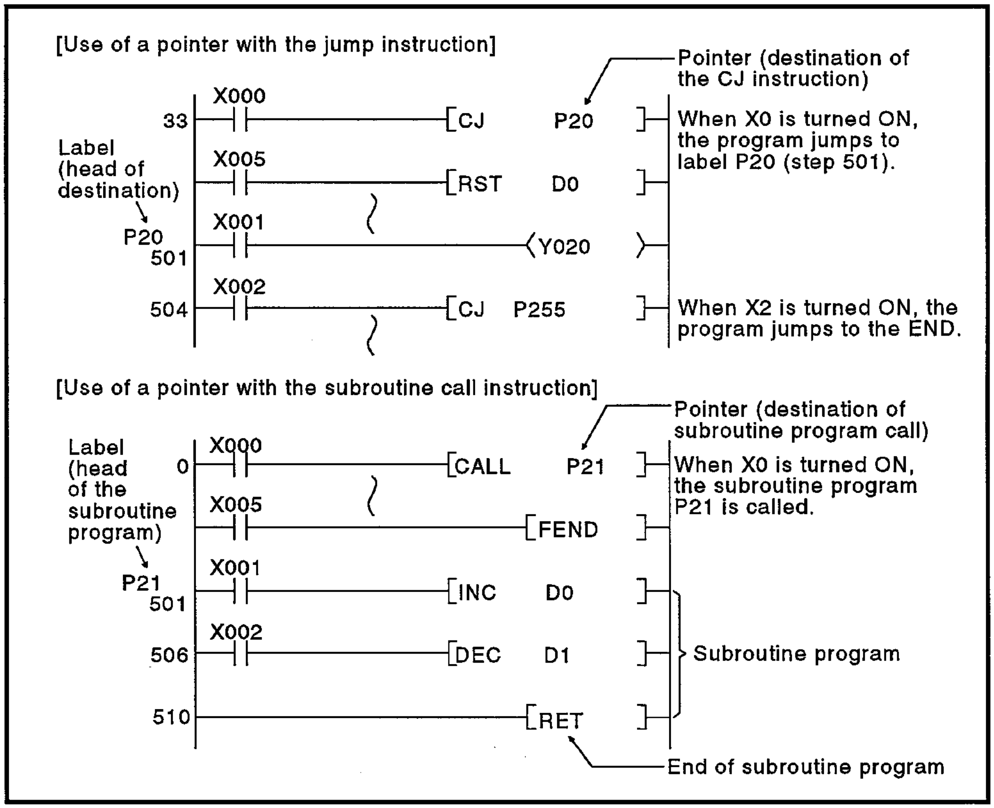

[Use of a pointer with the jump instruction]
Pointer (destination of the CJ instruction)
X000
33
CJ
P20
When X0 is turned ON, the program jumps to label P20 (step 501).
Label (head of destination)
X005
RST
D0
X001
P20
501
Y020
X002
504
CJ
P255
When X2 is turned ON, the program jumps to the END.
[Use of a pointer with the subroutine call instruction]
Pointer (destination of subroutine program call)
Label (head of the subroutine program)
X000
0
CALL
P21
When X0 is turned ON, the subroutine program P21 is called.
X005
FEND
X001
P21
501
INC
D0
X002
506
DEC
D1
Subroutine program
510
RET
End of subroutine program

Fig.3.38 Programming of Pointer with the Jump Instruction

(4) When a CHK instruction is used in the sequence program, P254 should be used as the label of the CHK instruction.
If a CHK instruction is not used, P254 can be used equally to P0 to P253.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	o	o	o	o	o	o	x	o
Remark											

3.16 Interrupt Pointers (I)

(1) Interrupt pointers are used as the label at the head of each interrupt program.
Each interrupt program begins with an interrupt pointer and ends with the IRET instruction.

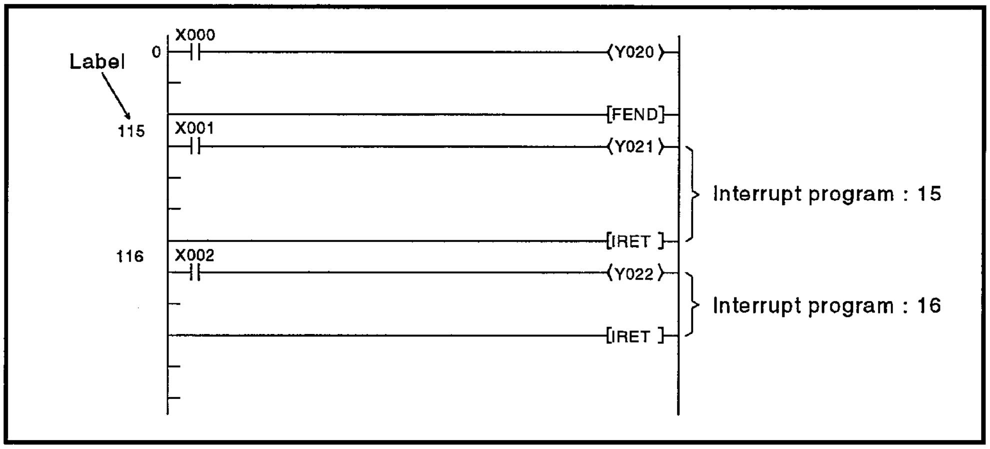


Fig. 3.39 Interrupt Pointer

(2) Table 3.2 shows interrupt pointers classified by purpose of use.

Table 3.2 Interrupt Pointers and Their Purpose of Use

Interrupt Pointer	Interrupt Factor		Interrupt Pointer	Interrupt Factor	
I0	Interrupt factor by the AI61 and A1SI61 process interrupt module	1st point	I16	(*1) Interrupt factor by the sequence start generator module	1st piece
I1		2nd point	I17		2nd piece
I2		3rd point	I18		3rd piece
I3		4th point	I19		4th piece
I4		5th point	I20		5th piece
I5		6th point	I21		6th piece
I6		7th point	I22		7th piece
I7		8th point	I23		8th piece
I8		9th point	I24	Unusable	
I9		10th point	I25		
I10		11th point	I26		
I11		12th point	I27		
I12		13th point	I28		
I13		14th point	I29	Interrupt factor by 40 ms by an internal timer	
I14		15th point	I30	Interrupt factor by 20 ms by an internal timer	
I15		16th point	I31	Interrupt factor by 10 ms by an internal timer	

REMARK

1) *1 : The sequence start generator module is a special function module which can send an interrupt start signal to the PC CPU. (excluding the AI61, A1SI61)
Pointers are allocated to the sequence start generator modules beginning with the one nearest to the PC CPU when the modules are installed to the base unit.

2) Priority of the interrupt factors is provided as follows.

High → I16 to I23, I0 to I15, I31, I30, I29 → Low
Priority

(3) When any of I29 to 31 are used with the program, interrupt programs are executed at every preset interrupt time. Fig. 3.40 shows the example of execution of interrupt programs when I29 to 31 are used.

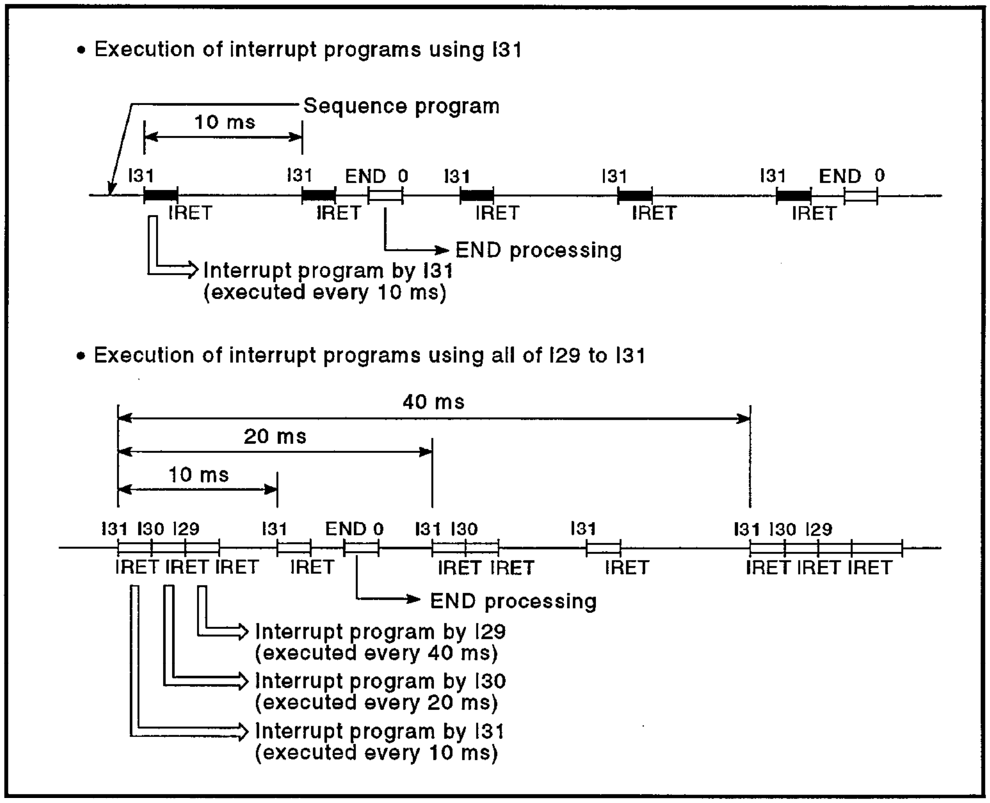


Fig. 3.40 Execution of Interrupt Programs Using I29 to I31

3.17 Special Relays and Special Registers

Special relays and special registers are the internal relays and data registers which are allocated to specific purposes in the PC CPU.
Special relays and special registers have the following purposes.

(1) PC CPU operation status check
The special relays and registers are used for the PC CPU operation status check as follows.

(a) Operation status check (RUN/STOP/PAUSE/STEP RUN)
(b) Error check by self-diagnosis .... Section 6.13
(c) Momentary power failure check .... Table 3.1
(d) Battery voltage check
(e) Operation error check........................... Section 6.13
(f) Execution status of a fault detection
program using annunciators .... Section 3.5
(g) Scan time check .... Section 2.5

(2) Timing contact
Some of the special relays vary in function according to the purpose of use in the sequence program.

(a) Normally ON/OFF flags .... Table 3.1
(b) RUN flag.. Table 3.1
(c) Initial processing flag (ON for 1 scan)................ Table 3.1
(d) Clock by every designated scan...................... Table 3.1
(e) Clock by every present time .... Table 3.1

(3) For commands to the PC CPU
The special relays and registers are used also to give commands to the PC CPU by the sequence program or with peripheral devices.

(a) PAUSE enable................................. Section 6.4
(b) Clock data read/write .... Section 6.9
(c) Execution of constant scan .... Section 6.1
(d) Execution of I/O module switching................. Section 6.10
(e) Execution of error check (fuse blow,
battery voltage, I/O module verify)................. Section 6.13

REMARKS

1) Special relay and special register numbers are allocated as follows.

(a) Special relays .... M9000 to M9255
(b) Special registers......... D9000 to D9255

2) For details of the special relays and registers which can be used with the PC CPU, refer to the ACPU Programming Manual (Common Instructions) (IB-66250).

Special relays and registers used for confirming the operating status and timing contact of the PC CPU are as given in Table 3.1 below.

Table 3.1 Applications of Special Relays and Registers

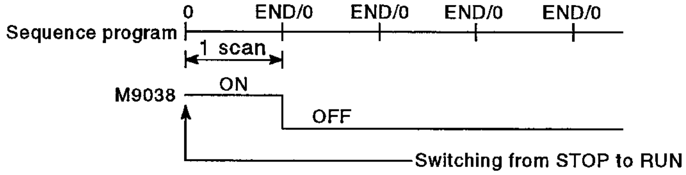
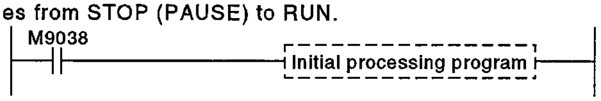
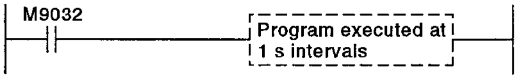
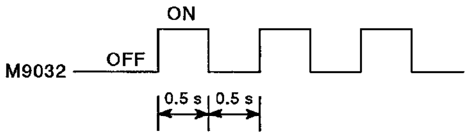
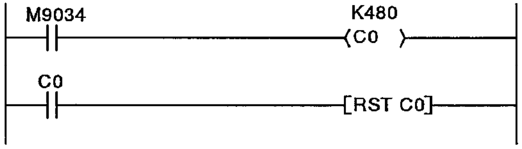
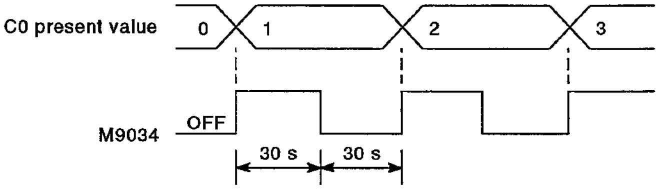
Item	Special Relay	Special Register	Applications and Contents
Initial processing flag (1 scan ON)	M9038	—	(1) M9038 relays turns ON for 1 scan when the PC CPU state switches from STOP (PAUSE) to RUN.  0 END/0 END/0 END/0 END/0 Sequence program 1 scan M9038 ON OFF Switching from STOP to RUN (2) By using M9038, a sequence program to be executed only once can be created without using a PLS instruction when the CPU state switches from STOP (PAUSE) to RUN. 
Clock at designated interval	M9030 to M9034	—	(1) M9030 to M9034 relays switch ON and OFF at designated intervals when the CPU is in the RUN state. (They hold the ON or OFF state when the CPU is in the STOP state.) • M9030 : 0.1 s clock • M9031 : 0.2 s clock • M9032 : 1 s clock • M9033 : 2 s clock • M9034 : 1 min clock (2) By using M9030 to M9034 relays, a program executed at designated intervals can be created and the time measurement can be done without using timers. • Program executed at 1 s intervals   • Measurement of 8 hours   C0 present value 0 1 2 3 M9034 OFF 30 s 30 s

Table 3.1 Applications of Special Relays and Registers (continued)

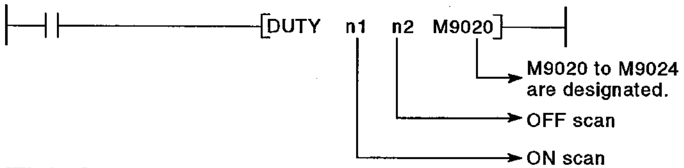
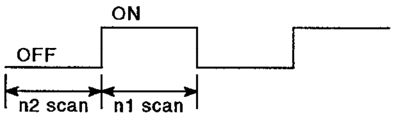
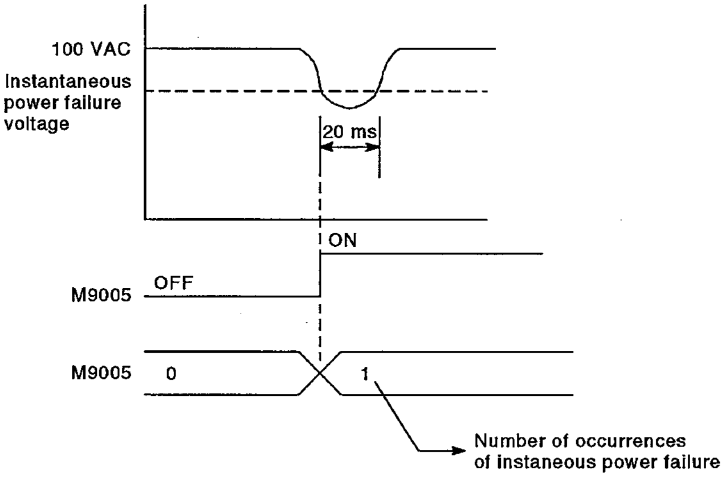
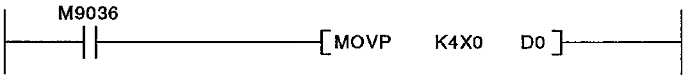
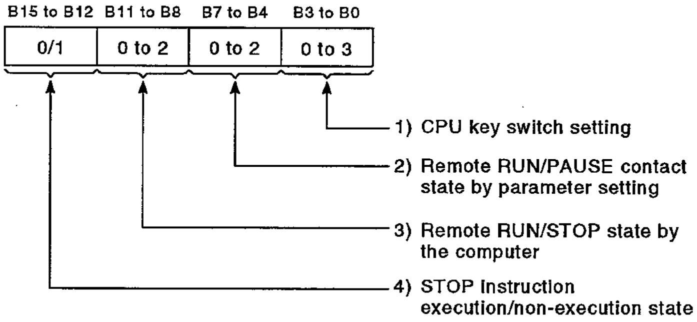
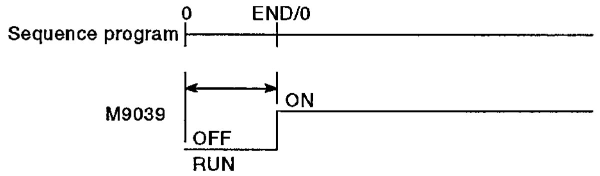
Item	Special Relay	Special Register	Applications and Contents
Clock by designated scan	M9020 to M9024	—	(1) M9020 to M9024 relays switch ON and OFF by designated scans. Scans are designated with a DUTY instruction. [Sequence program]  DUTY n1 n2 M9020 M9020 to M9024 are designated. OFF scan ON scan [Timing]  (2) Timing starts with the OFF state when the power is turned ON or reset. (3) By setting the n1 scan to 0, the clock can be terminated.
Detection of instantaneous power failure	M9005	D9005	(1) M9005 relay is used for detecting the occurrence of instantaneous power failure of 20 ms or less in the CPU. It remains ON when an instantaneous power failure is detected. When M9005 is ON, the sequence program operation continues. (2) When an instantaneous power failure is detected, D9005 (AC DOWN detection) data increases by 1.  100 VAC Instantaneous power failure voltage 20 ms ON OFF M9005 M9005 0 1 Number of occurrences of instaneous power failure (3) This can be used for checking the stability of the power supply being used.
Always OFF flag	M9037	—	(1) M9037 relay is always OFF when the power is ON. (2) This can be used to disable execution temporarily for debugging, etc.
Always ON flag	M9036	—	(1) M9036 relay is always ON when the power is ON. (2) This can be used to create a program which is executed only once after the power is turned ON. 

Table 3.1 Applications of Special Relays and Registers (continued)

Item	Special Relay	Special Register	Applications and Contents
Confirmation of operating state	—	D9015	(1) D9015 is used to store the CPU operating state.  B15 to B12 B11 to B8 B7 to B4 B3 to B0 0/1 0 to 2 0 to 2 0 to 3 1) CPU key switch setting 2) Remote RUN/PAUSE contact state by parameter setting 3) Remote RUN/STOP state by the computer 4) STOP instruction execution/non-execution state **Data Stored to 1) to 3)**: 0: RUN; 1: STOP; 2: PAUSE; 3: STEP RUN **Data Stored to 4)**: 0: STOP instruction not executed; 1: STOP instruction executed (2) This can be used to check the cause of stop when the PC CPU's RUN key switch is set in the RUN position and the CPU does not enter the RUN state.
RUN flag	M9039	—	(1) M9039 relay turns ON at the 2nd scan of the sequence program when the RUN key switch is in the RUN position. 

4. ALLOCATION OF I/O NUMBERS

This section gives the input and output number allocation procedures for performing communications with the input/output modules and special function modules by using the building-block type CPU, A0J2H, A2CCPU, and A1FXCPU.

The I/O number allocation for the A73 is the same as that for the building-block type CPUs.
However, since the installation procedure for the main base unit is different from that for the building-block type CPU, refer to the following manuals.

- A73 : A73CPU Reference Manual (IB-66233)

Since the I/O number allocation for the A52G varies according to the operation mode (remote I/O mode, proximity I/O mode, composite mode), refer to the following manual:

- A52GCPU (T21B) Reference Manual (IB-66420)

4.1 I/O Numbers

The input and output numbers are used with the sequence program for data input from the input modules and for data output to the output modules.
The input and output numbers are expressed in three hexadecimal digits.
Fig. 4.1 shows the example of the I/O numbers when all of the I/O modules are of the 16-point type.

Power supply module	CPU module	X [0] [0] [0]	X [0] [1] [0]	X [0] [2] [0]	Y [0] [3] [0]	Y [0] [4] [0]
		to	to	to X2C to	to	to
		X [0] [0] [F]	X [0] [1] [F]	X [0] [2] [F]	Y [0] [3] [F]	Y [0] [4] [F]
		input 16 points	input 16 points	input 16 points	output 16 points	output 16 points

Fig. 4.1 Example of the I/O Numbers

REMARK

When a peripheral device is used for programming, the I/O numbers can be input in two digits.

I/O number		Input by a peripheral device
X010	→	X10
Y020	→	Y20

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	x	o	o	o	o	o	x	x	x
Remark											

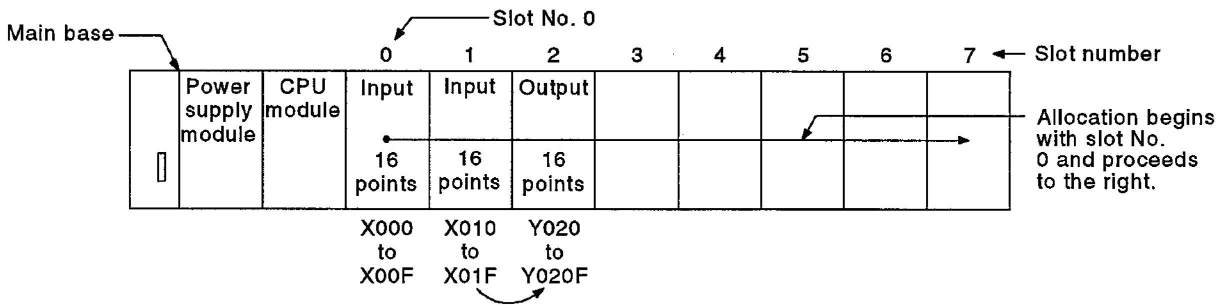
4.2 I/O Number Allocation of the Building-block Type CPUs

This section gives the I/O number allocation procedure of the building-block type CPUs.

4.2.1 Basics of the I/O number allocation

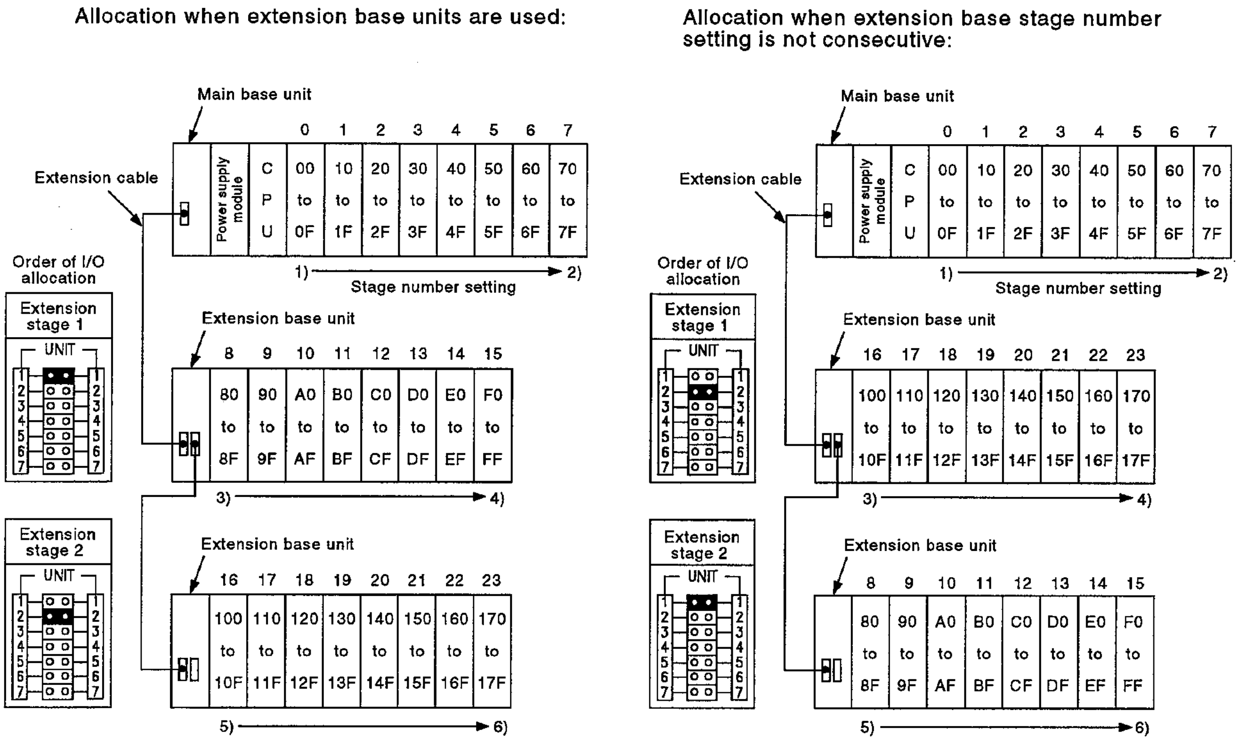
The I/O number allocation should be performed when the PC CPU is powered on or reset.
Designate the I/O numbers, which are allocated as follows, with the sequence program.

(1) The I/O numbers are allocated beginning with slot No. 0 of the main base unit (right next to the CPU module) to the right.
Input modules are allocated with X□□□, and output modules are allocated with Y□□□ in consecutive numbers.



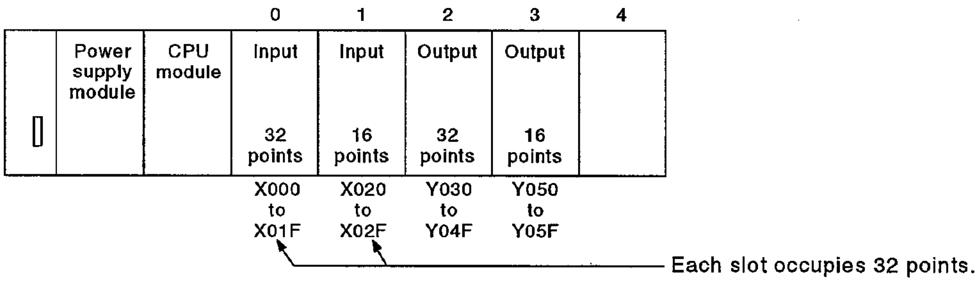
Inputs (X) and outputs (Y) are allocated in consecutive numbers.

(2) If extension base units are used, the allocation at the 1st extension base unit begins with the number which immediately follows the last number allocated to the main base unit.
Allocation of extension base units should not follow the order of connection of extension cables but follow the order of stage number setting of extension base units.

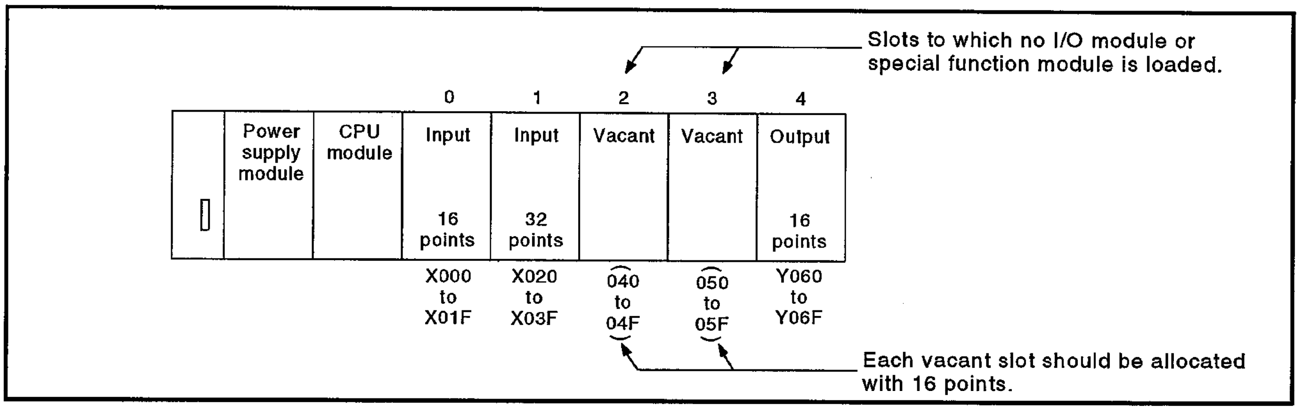


* The I/O numbers are allocated in the order of 1), 2), 3), 4), 5), and 6).
The I/O number allocation examples use the 16-point modules.

(3) Each module occupies the number of I/O numbers which is equal to the number of the I/O points of the module.
For example, when a 32-point input module is loaded to slot No. 0 of the main base unit, 32 points of I/O numbers from X00 to X1F are occupied.



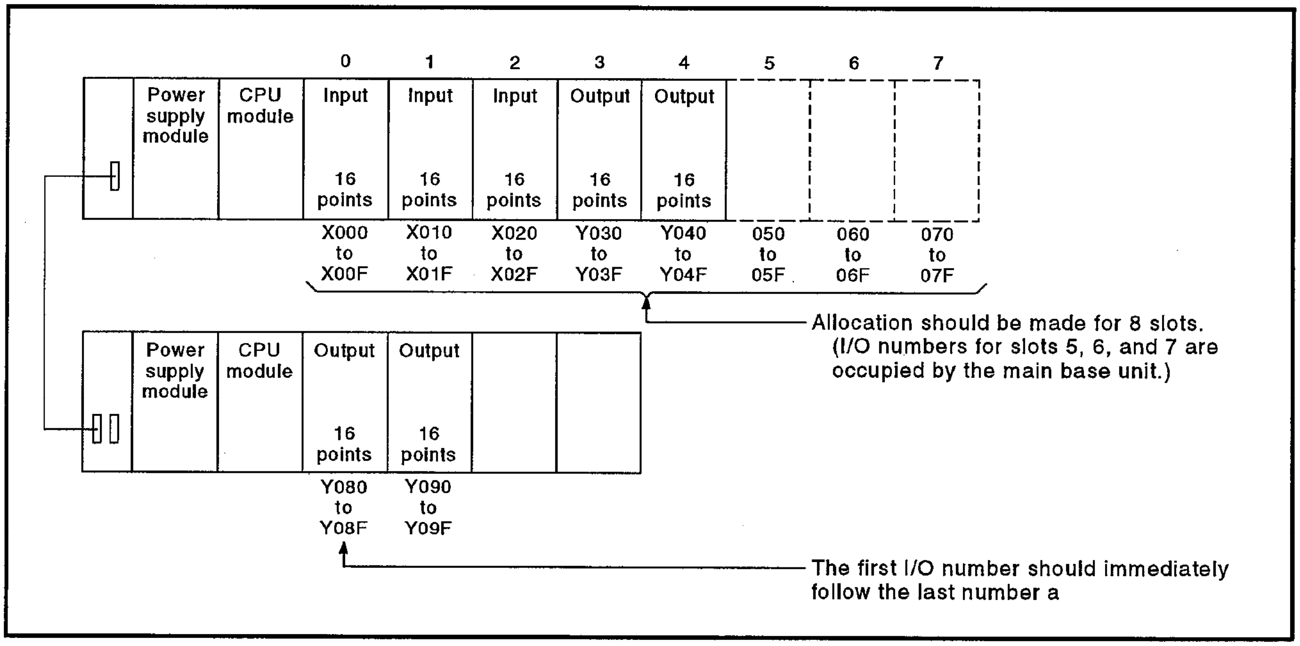
(4) Each vacant slot to which no I/O module or special function module is loaded should be allocated with 16 points.



Slots to which no I/O module or special function module is loaded.

Each vacant slot should be allocated with 16 points.

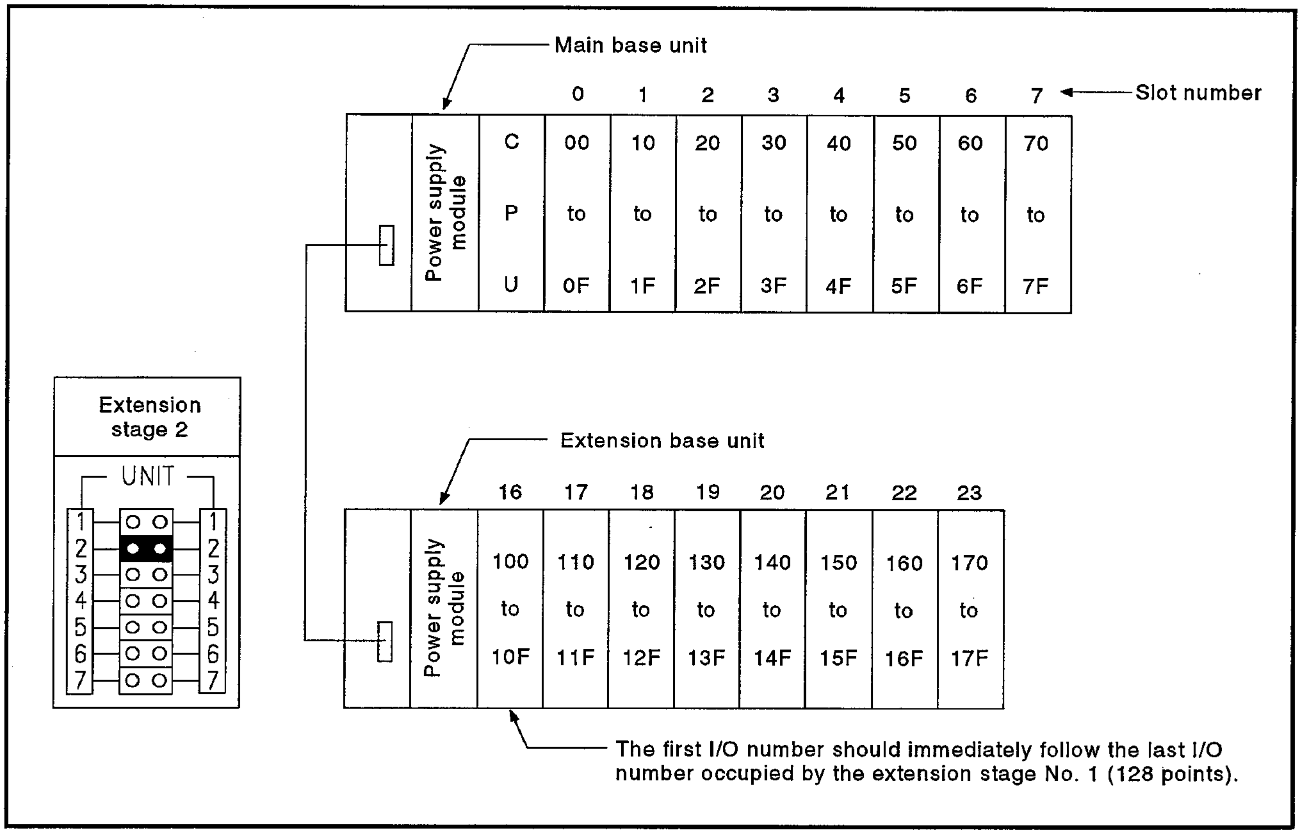
(5) Each of the main and extension base units should be regarded to have 8 slots to which I/O numbers are allocated.
For example, allocate the I/O numbers for 8 slots to a 5-slot main base unit. The I/O number allocation of the next (extension) base unit should begin with the number which immediately follows the last number allocated to the main base unit.



Allocation should be made for 8 slots. (I/O numbers for slots 5, 6, and 7 are occupied by the main base unit.)

The first I/O number should immediately follow the last number a

(6) If setting of the extension stage numbers is not consecutive (stage numbers are skipped), the I/O numbers should be allocated considering that the I/O numbers obtained as "the number of skipped stages x 8 (slots) x 16 (points)" are already occupied.



The first I/O number should immediately follow the last I/O number occupied by the extension stage No. 1 (128 points).

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	o	o	o	o	o	x	x	x
Remark											

4.2.2 I/O allocation using peripheral devices

The building-block type CPUs can control I/O modules and special function modules without the I/O allocation using peripheral devices.
The I/O allocation using peripheral devices has the following features.

(1) Functions of the I/O allocation using peripheral devices

(a) When a 5-slot base unit is used, the number of I/O points to be allocated to additional 3 slots can be set at 0 points so that other I/O numbers may be efficiently used.

(b) The number of I/O points can be reserved to adopt modules other than the 16-point module to allow system expansion.

(c) Allocated I/O numbers are held unchanged when I/O modules or special function modules other than the 16-point module are removed for repair or replacement.

(2) Basics of the I/O allocation using peripheral devices

(a) The number of allocation points per slot of the main base unit or extension base unit is provided as follows.

Number of Allocation Points			
Vacant Slot	**Input Module**	**Output Module**	**Special Function Module**
0	—	—	—
16	16	16	16
32	32	32	32
48	48	48	48
64	64	64	64

(b) The number of I/O points allocated to a specific slot by using a peripheral device is given priority over the number of I/O points occupied by a module to be loaded in the slot.

1) If the number of I/O points allocated to a slot using a peripheral device is smaller than that occupied by a module loaded to the slot, the number of I/O points usable with the module decreases.
For example, when a 32-points input module is loaded in a slot to which 16 points for the input module are allocated using a peripheral device, the latter 16 points occupied by the input module cannot be used.

2) If the number of I/O points allocated to a slot is larger than that occupied by the I/O module loaded to the slot, the excessive points are set as dummy points.

3) If a slot in which an I/O module is loaded is set as a vacant slot, the loaded module becomes unusable.

(c) The I/O allocation of the slots to which I/O allocation using a peripheral device has not been set is executed according to the number of points occupied by the modules loaded in the slots.

(3) Precautions

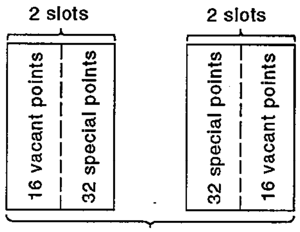
(a) The I/O allocation of a slot in which a special function module is loaded must correspond exactly to that of the loaded module.
If there is any discrepancy between the I/O allocation of the slot and that of the loaded module, an error occurs.

1) A11VC..............Output: 16 points

2) AI61.................Special: 32 points

3) AG62................Input: Number of setting points

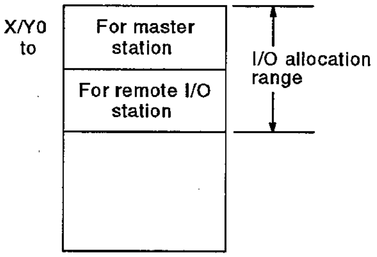
4) 2-slot module.....Set with 16 vacant points and 32 special points as shown below.

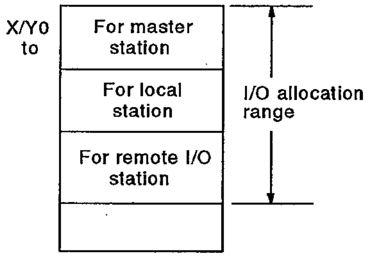


Also refer to the User's Manual for the special function module to be used.

(b) When the MELSECNET data link system is used, perform the I/O allocation as follows.

1) If the I/O allocation is to be set at a master station, the master station and all the remote I/O station need I/O allocation.





2) If the I/O allocation is to be set at a local station, only the local stations need I/O allocation.

3) If the I/O allocation is to be set at a composite I/O module (such as A42XY), perform the I/O allocation for the output module.

POINT

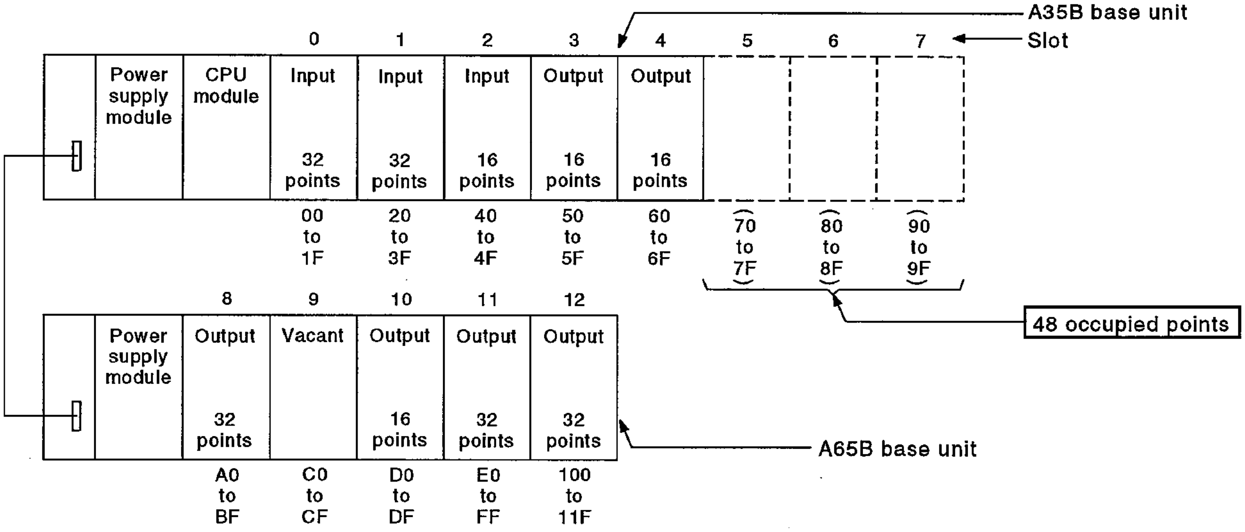
The following restrictions are provided to the last slot of the 7th extension stage of the A3CPU module.
1) If an input module is loaded, the Y addresses cannot be used for internal relays.
2) If the I/O allocation using a peripheral device is set including some differences from that of the module to be loaded, do not load the module to the slot. If it is loaded, input and output will beincorrectly executed.
3) If the X and Y addresses are transferred within a data link system, do not load a module to the slot.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	o	o	o	o	o	x	x	x
Remark											

4.2.3 Example of I/O number allocation

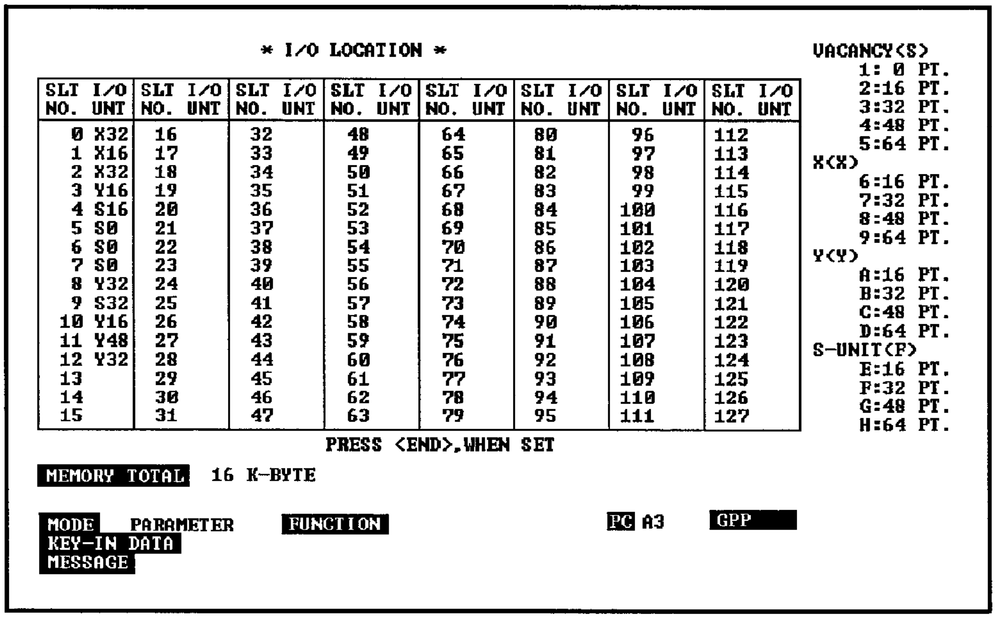
The following are examples of I/O number allocation when a building-block type CPU is loaded with I/O modules and when a peripheral device is used for I/O allocation.

(1) I/O allocation when I/O modules are loaded



(2) I/O allocation using a peripheral device

(a) Example of I/O allocation



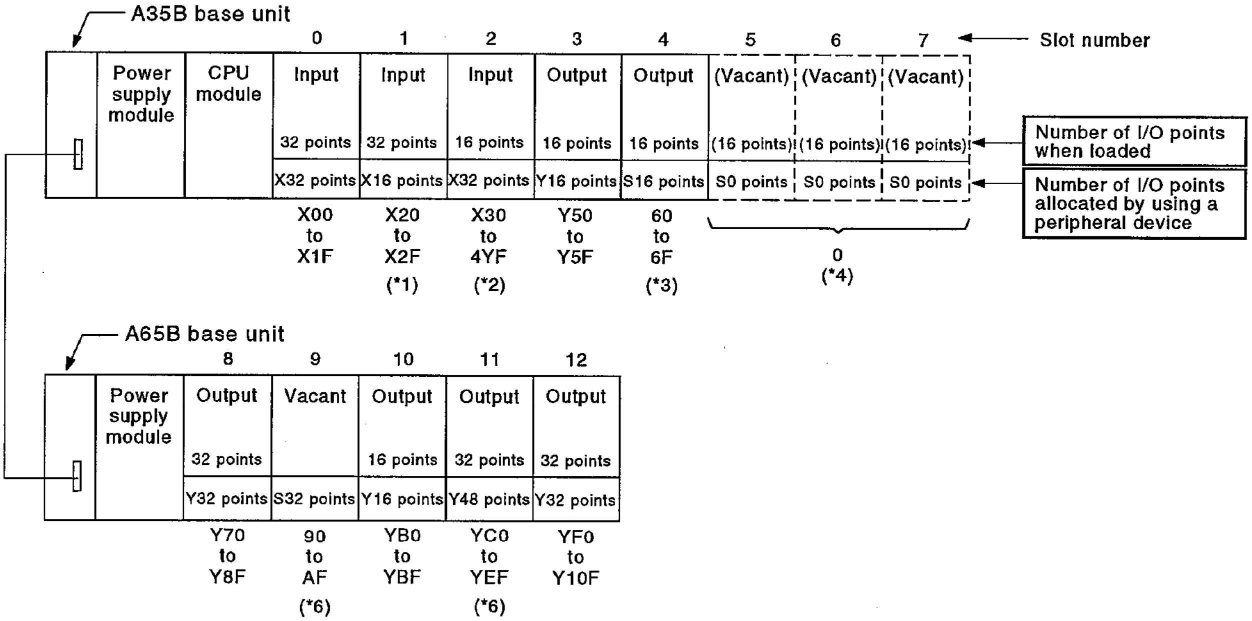
```
                        * I/O LOCATION *                                   VACANCY(S)
                                                                              1: 0 PT.
 SLT  I/O|SLT  I/O|SLT  I/O|SLT  I/O|SLT  I/O|SLT  I/O|SLT  I/O|SLT  I/O      2:16 PT.
 NO.  UNT|NO.  UNT|NO.  UNT|NO.  UNT|NO.  UNT|NO.  UNT|NO.  UNT|NO.  UNT      3:32 PT.
   0  X32| 16     | 32     | 48     | 64     | 80     |  96    |112           4:48 PT.
   1  X16| 17     | 33     | 49     | 65     | 81     |  97    |113           5:64 PT.
   2  X32| 18     | 34     | 50     | 66     | 82     |  98    |114         X(X)
   3  Y16| 19     | 35     | 51     | 67     | 83     |  99    |115           6:16 PT.
   4  S16| 20     | 36     | 52     | 68     | 84     | 100    |116           7:32 PT.
   5  S0 | 21     | 37     | 53     | 69     | 85     | 101    |117           8:48 PT.
   6  S0 | 22     | 38     | 54     | 70     | 86     | 102    |118           9:64 PT.
   7  S0 | 23     | 39     | 55     | 71     | 87     | 103    |119         Y(Y)
   8  Y32| 24     | 40     | 56     | 72     | 88     | 104    |120           A:16 PT.
   9  S32| 25     | 41     | 57     | 73     | 89     | 105    |121           B:32 PT.
  10  Y16| 26     | 42     | 58     | 74     | 90     | 106    |122           C:48 PT.
  11  Y48| 27     | 43     | 59     | 75     | 91     | 107    |123           D:64 PT.
  12  Y32| 28     | 44     | 60     | 76     | 92     | 108    |124         S-UNIT(F)
  13     | 29     | 45     | 61     | 77     | 93     | 109    |125           E:16 PT.
  14     | 30     | 46     | 62     | 78     | 94     | 110    |126           F:32 PT.
  15     | 31     | 47     | 63     | 79     | 95     | 111    |127           G:48 PT.
                                                                              H:64 PT.
                          PRESS <END>,WHEN SET

 MEMORY TOTAL  16 K-BYTE

 MODE  PARAMETER    FUNCTION                         PC A3    GPP
 KEY-IN DATA
 MESSAGE
```

Fig. 4.2 Example of I/O Allocation Using a Peripheral Device

(b) I/O numbers allocated by using a peripheral device


A35B base unit
Slot number
Number of I/O points when loaded
Number of I/O points allocated by using a peripheral device
A65B base unit

A35B base unit

		0	1	2	3	4	5	6	7
Power supply module	CPU module	Input	Input	Input	Output	Output	(Vacant)	(Vacant)	(Vacant)
		32 points	32 points	16 points	16 points	16 points	(16 points)	(16 points)	(16 points)
		X32 points	X16 points	X32 points	Y16 points	S16 points	S0 points	S0 points	S0 points
		X00 to X1F	X20 to X2F	X30 to 4YF	Y50 to Y5F	60 to 6F		0	
			(*1)	(*2)		(*3)		(*4)	

A65B base unit

	8	9	10	11	12
Power supply module	Output	Vacant	Output	Output	Output
	32 points		16 points	32 points	32 points
	Y32 points	S32 points	Y16 points	Y48 points	Y32 points
	Y70 to Y8F	90 to AF	YB0 to YBF	YC0 to YEF	YF0 to Y10F
		(*6)		(*6)	

REMARKS

*1 : Since only 16 points are allocated, the latter 16 input points are unusable.

*2 : Since 32 points are allocated, points from 40 to 4F are set as dummy points.

*3 : Since 16 points (S) for a vacant slot are allocated, these points cannot be used for output.

*4 : Since 0 point (S) for a vacant slot is allocated, these 3 slots are not allocated with I/O points.

*5 : Since 32 points (S) for a vacant slot are allocated, 32 vacant points are set.

*6 : Since 48 points are allocated, points from E0 to EF are set as dummy points.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	x	x	x	x	x	x	o	x	x
Remark											

4.3 I/O Allocation of the A0J2HCPU

The A0J2HCPU can be connected with up to 8 I/O modules.
The following describes the allocation of I/O numbers for the A0J2H system.

4.3.1 Basics of I/O allocation

(1) I/O modules for the A0J2H

(a) All the I/O modules for the A0J2H occupy 64 points per module.
Inputs (X) and outputs (Y) are allocated as follows.
1) Inputs (X) First-half 32 points
2) Outputs (Y) Second-half 32 points

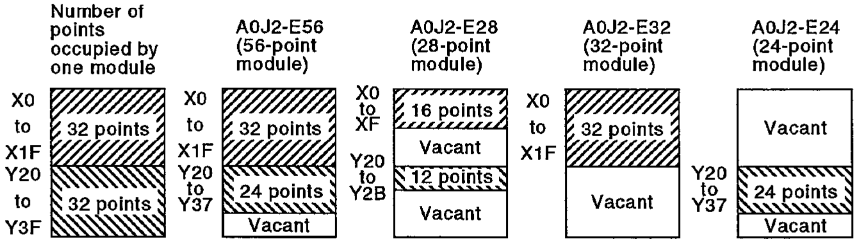
(b) The I/O numbers are allocated in the order of I/O module numbers of the I/O modules for the A0J2H.
The I/O module numbers do not need to be set in the order of connection of the I/O modules for the A0J2H.
Use caution not to use duplicate I/O module numbers.
The I/O module numbers can be set at 0 to 7.

(c) The head I/O number of the I/O modules for the A0J2H is provided as follows.

Setting	0	1	2	3	4	5	6	7
Head input number	X00	X40	X80	XC0	X100	X140	X180	X1C0
Head output number	Y20	Y60	YA0	YE0	Y120	Y160	Y1A0	Y1E0

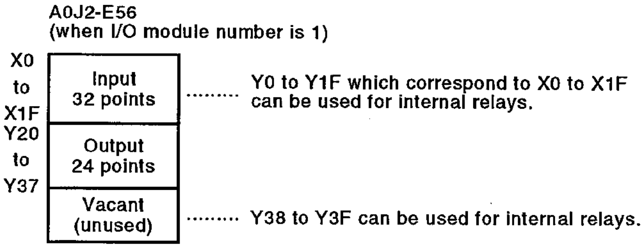
Example

When the I/O module setting number is 0, the I/O numbers of each I/O module are allocated as follows.



* The ▨ area is used for input and the ▧ area is used for output.

(d) The output numbers which correspond to allocated input numbers and unused numbers can be used for internal memory addresses. However, if an input-only module is used, the output numbers which correspond to allocated input numbers and unused numbers cannot be used for internal memory addresses.


A0J2-E56
(when I/O module number is 1)
X0 to X1F
Input 32 points
Y0 to Y1F which correspond to X0 to X1F can be used for internal relays.
Y20 to Y37
Output 24 points
Vacant (unused)
Y38 to Y3F can be used for internal relays.

(2) Special function modules for the A0J2H

(a) All the special function modules for the A0J2H occupy 64 points per module. The first-half 32 points are used for both input (X) and output (Y). However, the I/O numbers usable with the sequence program vary with each special function module. Refer to the User's Manual for respective special function module for usable I/O numbers and their purposes.

(b) The head I/O number of the special function modules for the A0J2 is provided according to the I/O module numbers as follows.

Setting	0	1	2	3	4	5	6	7
Head input number	X00	X40	X80	XC0	X100	X140	X180	X1C0
Head output number	Y00	Y40	Y80	YC0	Y100	Y140	Y180	Y1C0

(c) The special function modules can use the second-half 32 points for internal relays.

(3) Extension base unit

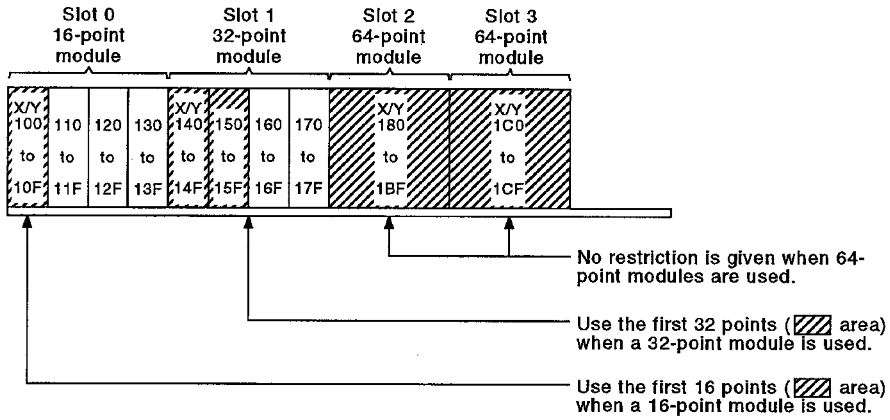
(a) If an extension base unit is used, slot 0 to 3 can be used.

(b) Slots 0 to 3 correspond to I/O module numbers 4 to 7. The I/O numbers of slot 0 always begin with X/Y100.

(c) Slot 1 occupies 64 points without regard to the number of I/O modules and vacant slots.
If 64-point I/O modules are not used (16- or 32-point modules are used), the following occurs.

1) When a 16-point module is used, the latter 48 points become unused. The A0J2H regards those 48 points as a wrap-around of the 16-point module. (I/O points are duplicated for every 16 addresses)
For example, if either of addresses 110, 120, or 130 (see Fig. 4.2 below) is accessed, the A0J2H regards that address 100 is accessed.

2) When a 32-point module is used, the latter 32 points become unused. The A0J2H regards those 32 points as a wrap-around of the 32-point module.
For example, if address 160 (see Fig. 4.2 below) is accessed, the A0J2H regards that address 140 is accessed.



* When modules other than 64-point I/O modules are used, I/O numbers other than actual I/O numbers (▨ area) cannot be used with the sequence program.

Fig. 4.2 I/O Allocation of Extension Base Unit

POINT

With an extension base unit, the A0J2H I/O modules and special function modules can be used up to 4 modules in total.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	x	x	x	x	x	x	o	x	x
Remark											

4.3.2 Example of I/O number allocation

The following figures show the examples of I/O allocation of the A0J2H.

(1) Fig. 4.3 shows an example of I/O allocation when the A0J2H I/O modules and special function modules are used.

I/O Module No.	I/O Number	
	I/O Module	Special Function Module
0	X00 to X1F Y20 to Y3F	X/Y00 to X/Y1F
1	X40 to X5F Y60 to Y7F	X/Y40 to X/Y5F
2	X80 to X9F YA0 to YBF	X/Y80 to X/Y9F
3	XC0 to XDF YE0 to YFF	X/YC0 to X/YDF
4	X100 to X11F Y120 to Y13F	X/Y100 to X/Y11F
5	X140 to X15F Y160 to Y17F	X/Y140 to X/Y15F
6	X180 to X19F Y1A0 to Y1BF	X/Y180 to X/Y19F
7	X1C0 to X1DF Y1E0 to Y1FF	X/Y1C0 to X/Y1DF

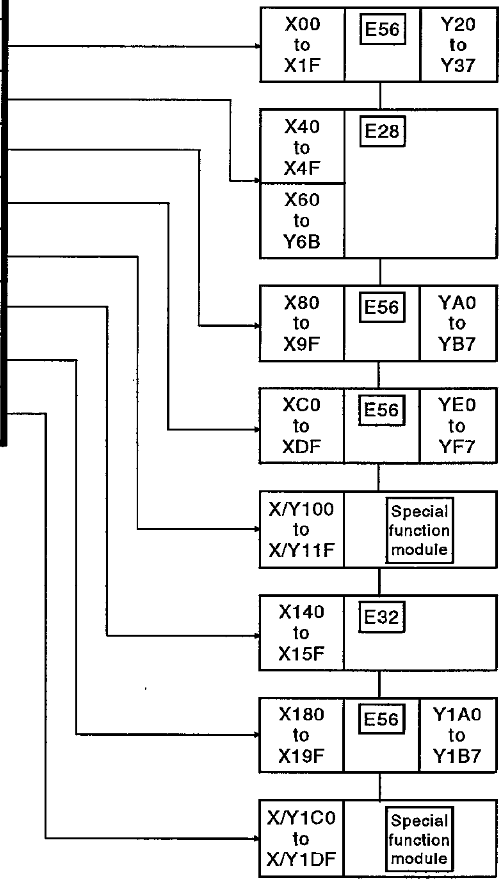
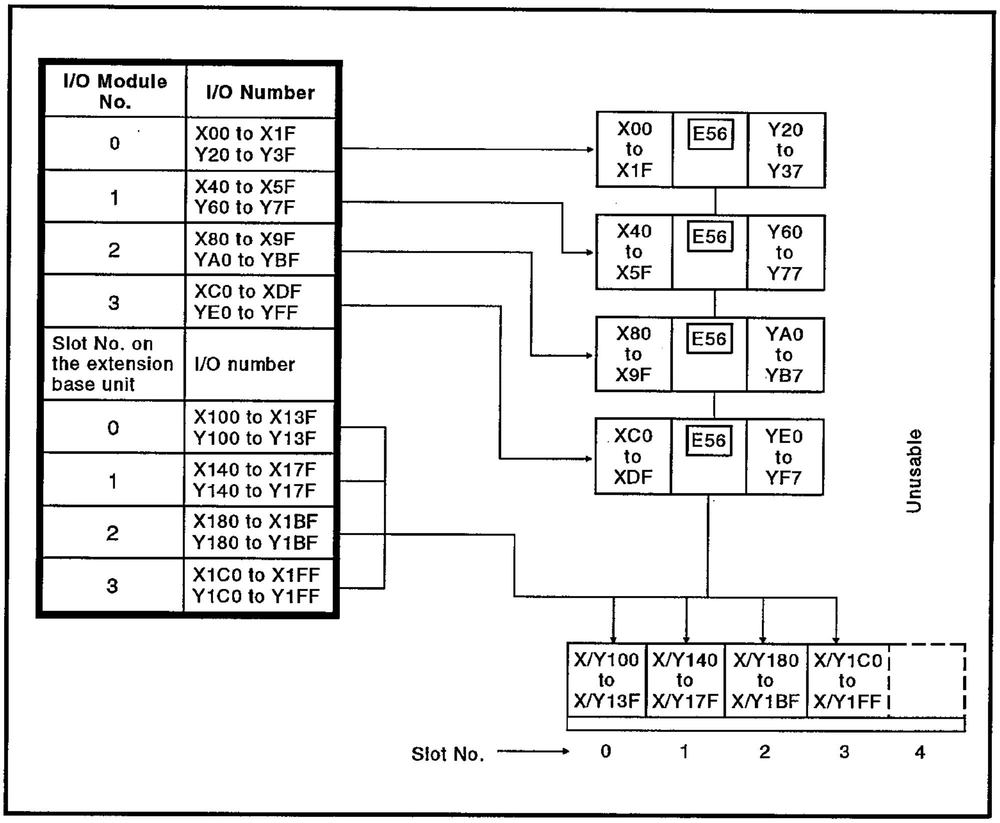


Fig. 4.3 Example of I/O Allocation (1) (with the A0J2 I/O modules and special function modules)

(2) Fig. 4.4 shows an example of I/O allocation when the A0J2H I/O modules and an extension base unit are used.



I/O Module No.	I/O Number
0	X00 to X1F Y20 to Y3F
1	X40 to X5F Y60 to Y7F
2	X80 to X9F YA0 to YBF
3	XC0 to XDF YE0 to YFF
Slot No. on the extension base unit	I/O number
0	X100 to X13F Y100 to Y13F
1	X140 to X17F Y140 to Y17F
2	X180 to X1BF Y180 to Y1BF
3	X1C0 to X1FF Y1C0 to Y1FF

Fig. 4.4 Example of I/O Allocation (2) (with an extension base unit)

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	x	x	x	x	x	x	x	△*1	x
Remark	*1: Only the A2C is applicable.										

4.4 I/O Allocation of the A2CCPU

The A2CCPU controls communication with I/O modules and remote terminals by I/O numbers.
The following describes the allocation of I/O numbers for the A2C system.

4.4.1 Basics of I/O allocation

The I/O numbers are allocated with X/Y0 to X/Y1FF in the order of station numbers set with the station number setting switch on each module. Note that the I/O number allocation is not executed in the order of connection of the I/O modules and remote terminal modules.

(1) Station number and I/O number allocation
Each station is allocated with 8 points. Stations 1 through 64 can be set. Table 4.1 shows correspondence between station number setting and I/O number allocation.

Table 4.1 Correspondence between Station Numbers and I/O Numbers

Station No.	I/O Number	Station No.	I/O Number	Station No.	I/O Number	Station No.	I/O Number
1	X/Y0 to 7	17	X/Y80 to 87	33	X/Y100 to 107	49	X/Y180 to 187
2	X/Y8 to F	18	X/Y88 to 8F	34	X/Y108 to 10F	50	X/Y188 to 18F
3	X/Y10 to 17	19	X/Y90 to 97	35	X/Y110 to 117	51	X/Y190 to 197
4	X/Y18 to 1F	20	X/Y98 to 9F	36	X/Y118 to 11F	52	X/Y198 to 19F
5	X/Y20 to 27	21	X/YA0 to A7	37	X/Y120 to 127	53	X/Y1A0 to 1A7
6	X/Y28 to 2F	22	X/YA8 to AF	38	X/Y128 to 12F	54	X/Y1A8 to 1AF
7	X/Y30 to 37	23	X/YB0 to B7	39	X/Y130 to 137	55	X/Y1B0 to 1B7
8	X/Y38 to 3F	24	X/YB8 to BF	40	X/Y138 to 13F	56	X/Y1B8 to 1BF
9	X/Y40 to 47	25	X/YC0 to C7	41	X/Y140 to 147	57	X/Y1C0 to 1C7
10	X/Y48 to 4F	26	X/YC8 to CF	42	X/Y148 to 14F	58	X/Y1C8 to 1CF
11	X/Y50 to 57	27	X/YD0 to D7	43	X/Y150 to 157	59	X/Y1D0 to 1D7
12	X/Y58 to 5F	28	X/YD8 to DF	44	X/Y158 to 15F	60	X/Y1D8 to 1DF
13	X/Y60 to 67	29	X/YE0 to E7	45	X/Y160 to 167	61	X/Y1E0 to 1E7
14	X/Y68 to 6F	30	X/YE8 to EF	46	X/Y168 to 16F	62	X/Y1E8 to 1EF
15	X/Y70 to 77	31	X/YF0 to F7	47	X/Y170 to 177	63	X/Y1F0 to 1F7
16	X/Y78 to 7F	32	X/YF8 to FF	48	X/Y178 to 17F	64	X/Y1F8 to 1FF

(2) Occupied points and station numbers

(a) When the number of I/O points of I/O modules and remote terminal modules is 8 or more, one module occupies several consecutive station numbers.
Those station numbers occupied by a module cannot be set to other modules.

(b) Set the station numbers of I/O modules and remote terminal modules so that the numbers may be consecutive.
When modules which occupy 8 or more points per module are used, set the head station number of the station numbers occupied by the module as the station number of the module.

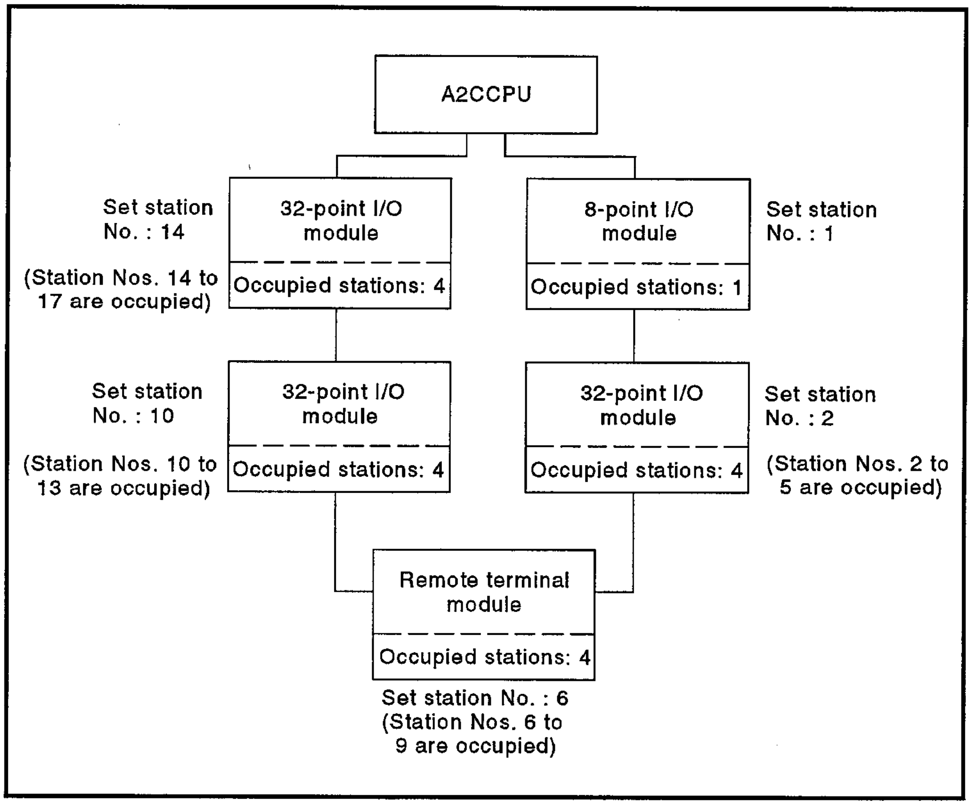


Fig. 4.5 Example of Station Number Setting

REMARK

For the I/O modules and remote terminal modules usable with the A2CCPU, refer to the A2CCPU (P21/R21), A2CCPU-DC24V, A2CCPUC24(-PRF), A2CJCPU(S3) User's Manual (IB-66545).

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	o	x	x	x	x	x	x	x	x
Remark											

4.5 I/O Number Assignment for A1FXCPU

This chapter describes I/O number assignment made to transfer data between the A1FXCPU and extension modules/extension blocks.

"Inputs (X)" are used to import data from the extension modules/extension blocks to the A1FXCPU, and "outputs (Y)" are used to output data from the A1FXCPU to the extension modules/extension blocks.

I/O numbers are addresses of the inputs/outputs built in the A1FXCPU and the extension modules/extension blocks.

The number of input/output points that may be controlled by the A1FXCPU is 242 (built in A1FXCPU: 14 input points/4 output points, extension modules/extension blocks: 224 points).

However, one special module or special block occupies 8 points.
Hence, when special modules/special blocks are used, the number of points available for extension modules/extension blocks is found by:

224 points -8 x (number of special modules/special blocks)

- The A1FXCPU contains 14 input points and 4 output points and occupies X0 to XD as inputs and Y10 to Y13 as outputs.
Therefore, extension modules/extension blocks use X/Y20 to X/YFF.

4.5.1 I/O number assignment

When switched on or reset by the RUN/STOP switch, the A1FXCPU makes the following I/O number assignment.
When writting a sequence program, specify the I/O numbers assigned in accordance with the following items.

(1) I/O number assignment

(a) I/O numbers are assigned to the extension module/extension block connected on the right-hand side of the A1FXCPU, starting with X/Y20.
Numbers X□□□ are assigned to the inputs of extension modules/ extension blocks and Y□□□ to their outputs.

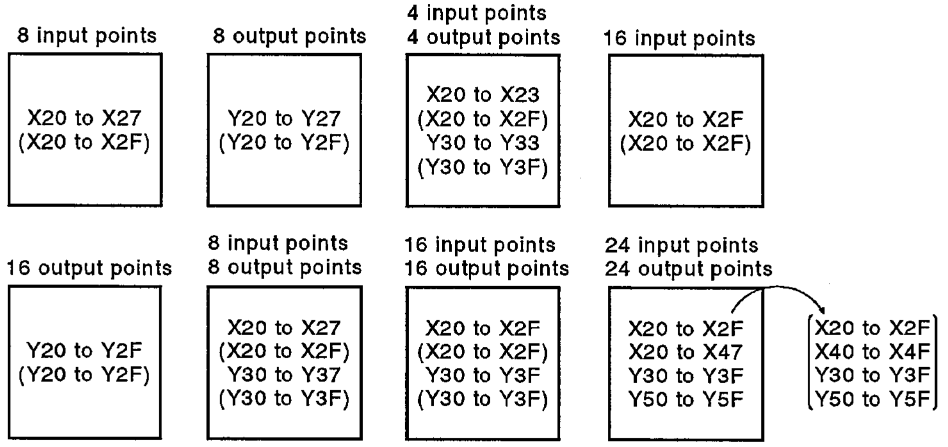
(b) I/O numbers are assigned in hexadecimal.

(c) Inputs/outputs atart at X/Yn0.
The I/O numbers of each module are indicated below.

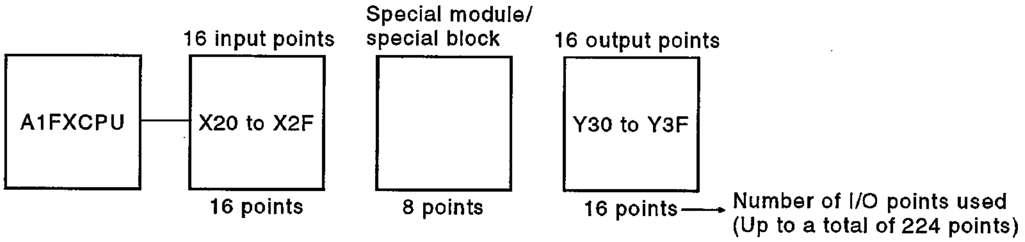
Number of I/O Points of Extension Module/Extension Block	I/O Numbers
8 input points	Xn0 to Xn7 (Xn8 to XnF must not be used)
8 output points	Yn0 to Yn7 (Yn8 to YnF are handled as internal relays)*1
4 input points, 4 output points	Xn0 to Xn3 (An4 to XnF must not be used) Y[n+1]0 to Y[n+1]3 (Y[n+1]4 to Y[n+1]F are handled as internal relays)*1
16 input points	Xn0 to XnF
16 output points	Yn0 to YnF
8 input points, 8 output points	Xn0 to Xn7 (Xn8 to XnF must not be used) Y[n+1]0 to Y[n+1]7 (Y[n+1]8 to Y[n+1]F are handled as internal relays) *1
16 input points, 16 output points	Xn0 to XnF, Y[n+1]0 to Y[n+1]7
24 input points, 24 output points	Xn0 to XnF, X[n+2]0 to X[n+2]7 (X[n+2]8 to X[n+2]F must not be used) Y[n+1]0 to Y[n+1]F, Y[n+3]0 to Y[n+3]7 (Y[n+3]8 to Y[n+3]F are handled as internal relays) *1

*1:Can be switched on/off in the sequence program but cannot be provided to the outside.

For example, I/O numbers are as follows when an extension module/extension block is connected on the right-hand side of the A1FXCPU. I/O numbers in parentheses are occupied by each extension module/extension block.

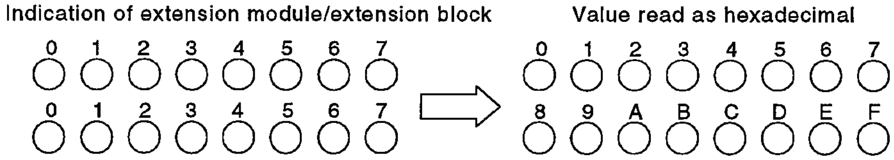


(d) One special module/special block occupies 8 points but does not use I/O numbers.
Hence, when special modules/special blocks are used, skip them over when setting the I/O numbers.

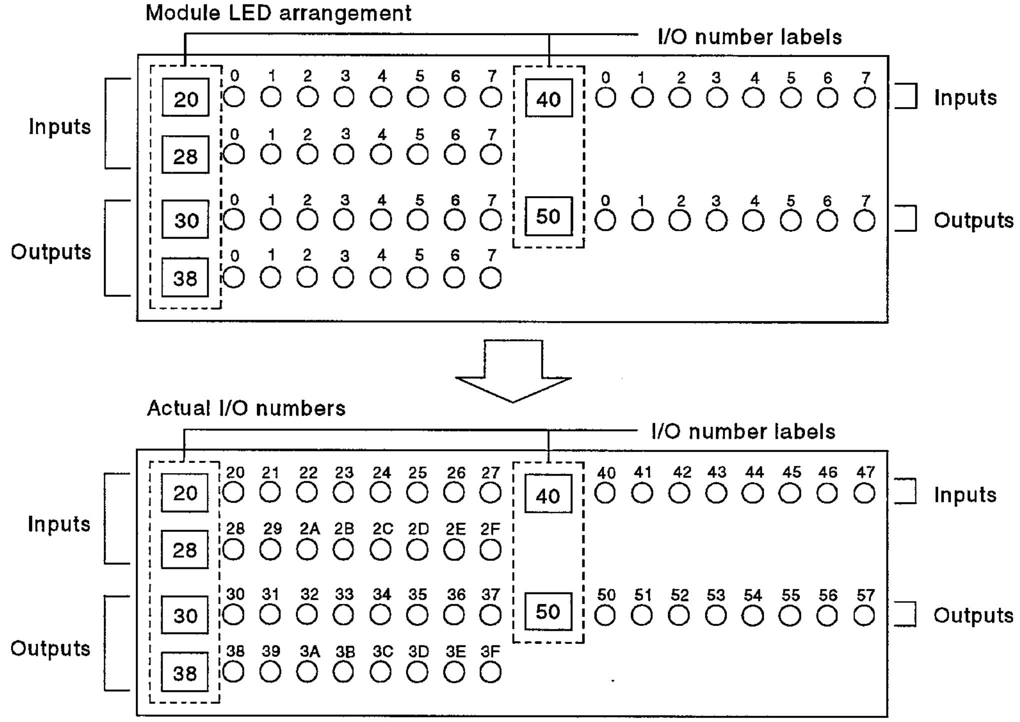


REMARKS

The LED indication of the extension module/extension block is in octal.
When using the A1FXCPU to control the extension module/extension block, read the octal of the LED indication as hexadecimal.


Indication of extension module/extension block
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
Value read as hexadecimal
0 1 2 3 4 5 6 7
8 9 A B C D E F

When a 48-point extension module (24 input points, 24 output points) is connected next to the A1FXCPU.



5. PROGRAM STRUCTURE

(1) Classification of programs

Table 5.1 shows the classification of the programs usable with the PC CPU.

Table 5.1 PC CPU Types and Programs

o : Applicable, x : Not applicable, — : Not provided

PC CPU Type	Main Program					Sub Program				
	Sequence Program		Microcomputer Program			Sequence Program		Microcomputer Program		
	Main Routine/ Subroutine Program	Interrupt Program	Utility Program	User-Created Micro-computer Program	SFC Program	Main Routine/ Subroutine Program	Interrupt Program	Utility Program	User-Created Micro-computer Program	SFC Program
A2C	o	—	o	o	o	—	—	—	—	—
A52G										
A1S(S1)		o								
A1SJ-S3										
A1SJH(S8)										
A1SH										
A2S(S1)										
A2SH(S1)										
A1FX										
A2AS (S1/S30) A2USH-S1			x	—						
Q02, Q02H										
Q06H						o	o			
A0J2H			o	o	—	—	—			
A1										
A1N										
A2										
A2N(S1)					o					
A2A(S1)			x	—						
A2U(S1)										
A3			o	o	—	o	o	—	o	—
A73					o					
A3H										
A3M										
A3N			x	—					—	
A3A										
A3U										
A4U										

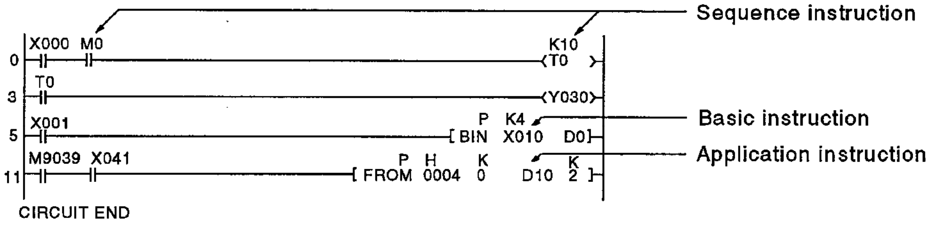
(2) Program capacity

Program capacity is the capacity of memory in which main programs and sub-programs are stored. Program capacity varies with type of PC CPU. Use parameter setting to fix program capacity.
(Refer to Chapter 8 for the default value and setting range of program capacity.)

Applicable CPU	All Types of CPUs
Remark	

5.1 Sequence Program

A sequence program consists of a main routine program, a subroutine program, and interrupt programs, and runs with the PC CPU sequence instructions, basic instructions, and applied instructions.



Sequence programs are classified into main routine programs, subroutine program, and interrupt programs.
There is no restrictions on the order of creation of the subroutine program and interrupt programs.
A subroutine program and interrupt programs can be contained in the sequence between an [FEND] instruction and an [END] instruction without dividing the program area into the subroutine program area and the interrupt program area.

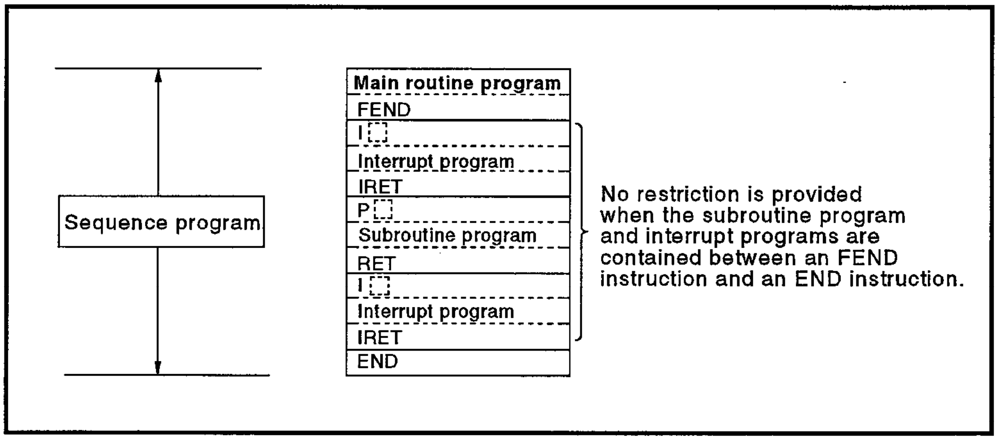

Sequence program
Main routine program
FEND
I□
Interrupt program
IRET
P□
Subroutine program
RET
I□
Interrupt program
IRET
END
No restriction is provided when the subroutine program and interrupt programs are contained between an FEND instruction and an END instruction.

Fig. 5.1 Order of Creation of the Subroutine Program and Interrupt Programs

POINT

The subsequence program for the PC CPU for which a subprogram can be created has the same structure as the main sequence program.

REMARK

Refer to the ACPU Programming Manual (Common Instructions) (IB-66250) for the sequence instructions, basic instructions, and application instructions.

Applicable CPU	All Types of CPUs
Remark	

5.1.1 Main routine program

The main routine program is stored in the head portion of the sequence program area and executed regularly in every scan.
If the main program is not stored, the PC CPU cannot execute not only the main program but also other programs.

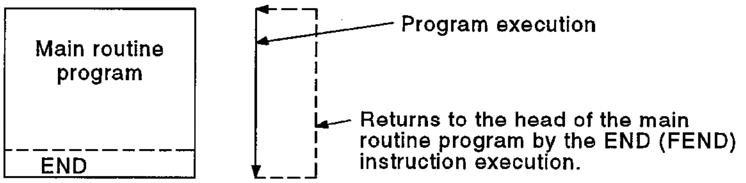

Main routine program
END
Program execution
Returns to the head of the main routine program by the END (FEND) instruction execution.

Fig. 5.2 Execution of the Main Routine Program

5.1.2 Subroutine program

The subroutine programs are executed when they are called by the main routine program with a CALL instruction.

(1) Purpose of the subroutine programs

(a) To execute a program for a specific processing more than once within one scan.

(b) To execute a program only when a specific condition is established.

(2) Structure of the subroutine programs

(a) The subroutine program must be created after the main routine program and after an FEND instruction and before an END instruction.

(b) Attach a P[ ] (pointer) to the head step of each subroutine program.
One pointer number cannot be used twice in the main routine program or the subroutine programs.

(c) Enter the RET instruction in the last step of each subroutine program.

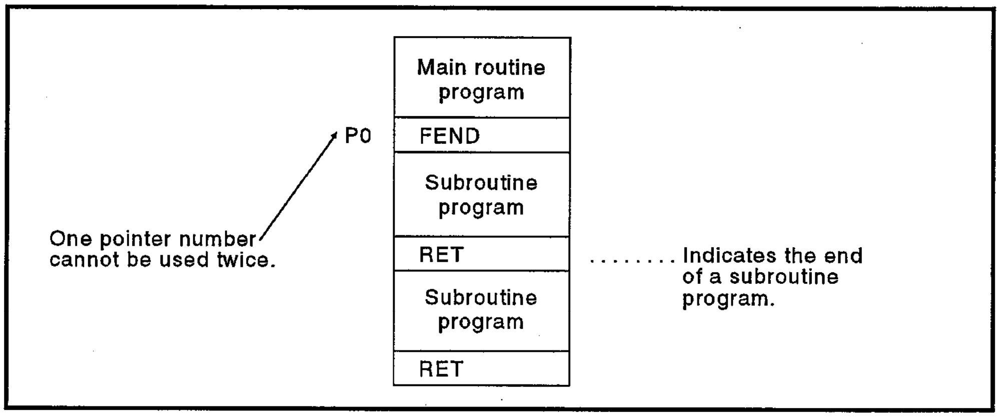

Main routine program
FEND
P0
Subroutine program
One pointer number cannot be used twice.
RET
.... Indicates the end of a subroutine program.
Subroutine program
RET

Fig. 5.3 Structure of Subroutine Programs

(3) Exection of the subroutine programs
Execution of the subroutine programs is controlled by turning ON/OFF the input of the CALL instruction.

(a) When the input condition is established : Specified subroutine programs are executed.

(b) When the input condition is not established : The subroutine programs are not executed, and the instruction right after the CALL instruction is executed.

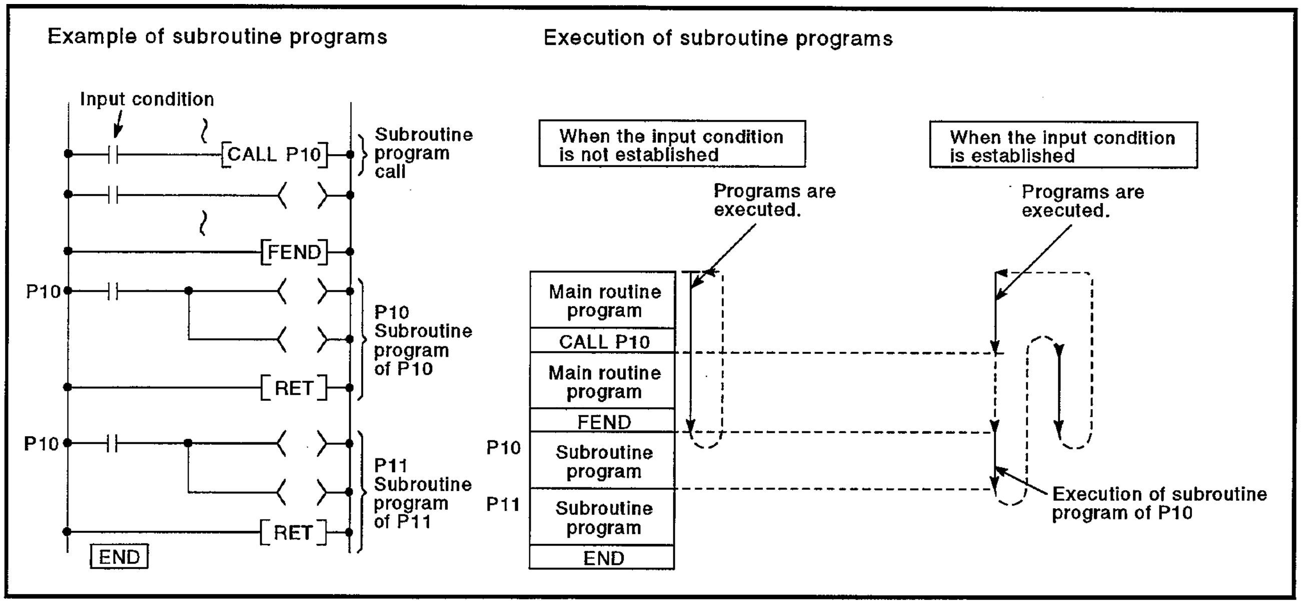

Example of subroutine programs
Input condition
CALL P10
Subroutine program call
FEND
P10
P10 Subroutine program of P10
RET
P11 Subroutine program of P11
END
Execution of subroutine programs
When the input condition is not established
When the input condition is established
Programs are executed.
Programs are executed.
Main routine program
CALL P10
Main routine program
FEND
P10
Subroutine program
P11
Subroutine program
END
Execution of subroutine program of P10

Fig. 5.4 Execution of the Subroutine Programs

(4) Nesting the subroutine programs
Nesting in which a subroutine program is called by another subroutine program can be done up to 5 levels.
Nesting of 6 or more levels is unusable.

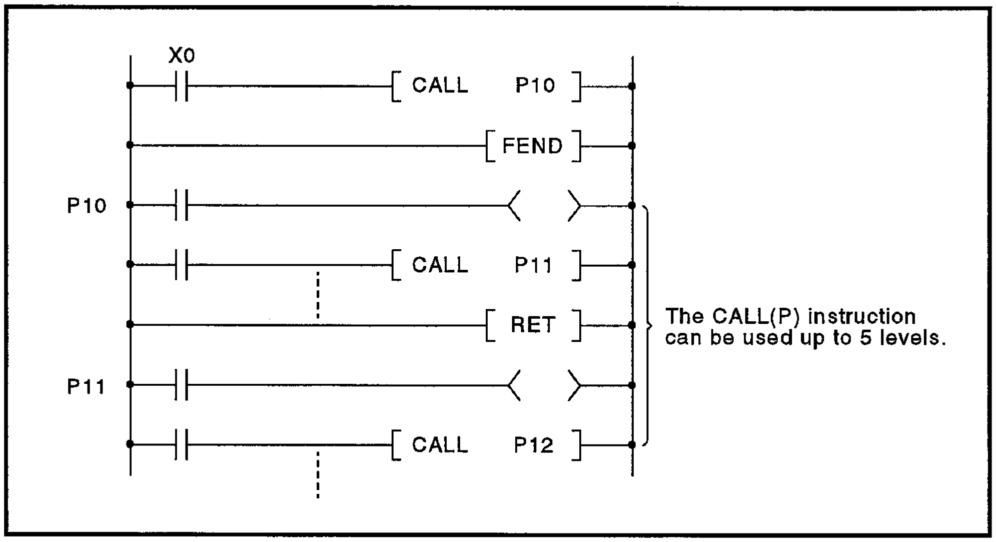


Fig. 5.5 Nesting of the Subroutine Programs

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	o	o	o	o	o	o	x	o
Remark											

5.1.3 Interrupt programs

Interrupt programs are executed only when interrupt factors occur. Each interrupt program can be created for each device number specified by interrupt points I0 to I31.

(1) Structure of the interrupt programs

(a) The interrupt programs must be created after the main routine program and after a FEND instruction and before a END instruction.

(b) Attach an I (interrupt pointer) to the head step of each interrupt program.

(c) Enter the IRET instruction in the last step of each interrupt program.

(2) Execution of the interrupt programs

(a) Use the EI instruction to enable interrupt.

1) An interrupt program is not executed if an interrupt factor occurs before execution of an EI instruction. The interrupt program which corresponds to an interrupt factor which occurs after execution of an EI instruction is executed.

2) An interrupt program is not executed if an interrupt factor occurs in the STOP state. The interrupt program which corresponds to an interrupt factor which occurs after execution of an EI instruction in the RUN state is executed.

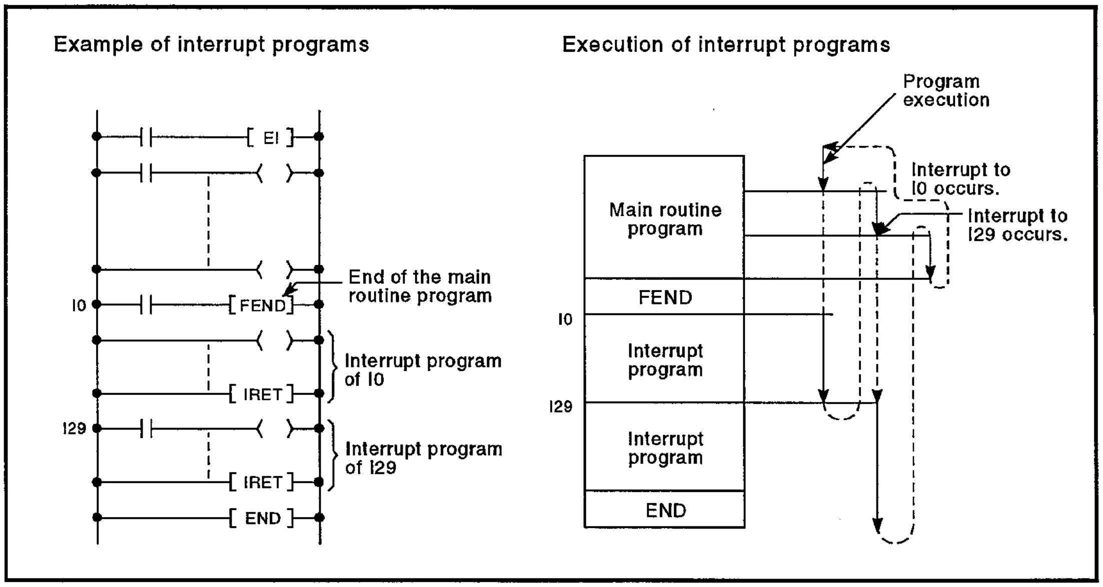


Fig. 5.6 Execution of Interrupt Programs

(b) When an interrupt factor occurs, the interrupt program specified by the interrupt pointer number that corresponds to the factor is executed.
Interrupt programs are executed according to the following conditions.

1) When multiple interrupts occur:
When multiple interrupt factors occur simultaneously, interrupt programs are executed according to the priority given to the interrupt pointer numbers.
When an interrupt program is being executed, the next priority interrupt program is executed when the currently executed interrupt program is finished.

2) When an instruction which disables interrupt (DI) is being executed:
Interrupt is disabled when a FROM/TO instruction or a CHG instruction is being executed.

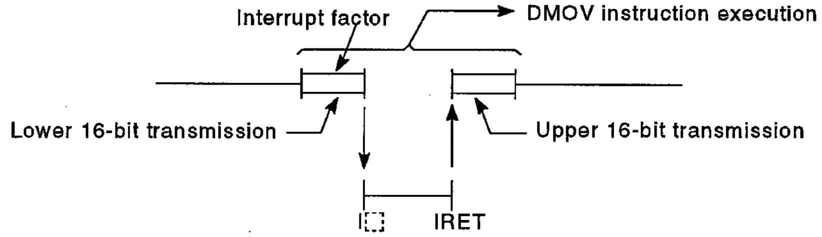
i) When an interrupt factor occurs when a FROM/TO instruction is being executed, the corresponding interrupt program is executed when read/write specified by the FROM/TO instruction execution is completed.

ii) When an interrupt factor occurs when a CHG instruction is being executed, the corresponding interrupt program is executed when the CHG instruction execution is completed.

3) When an instruction other than that mentioned above is being executed:
Interrupt programs are executed when a sequence instruction, a basic instruction, or an applied instruction other than those mentioned above is being executed.

Example

When an interrupt factor occurs when the lower 16-bit data is being transmitted by a 32-bit data transmission (DMOV) instruction, the DMOV instruction execution is suspended when transmission of the lower 16-bit data is completed, and an interrupt program is executed.
When the interrupt program exection is completed, the upper 16-bit data is transmitted by executing the DMOV instruction.


Interrupt factor
DMOV instruction execution
Lower 16-bit transmission
Upper 16-bit transmission
I□
IRET

If data to be used with an interrupt program is stored by use of the main routine/subroutine program, use the EI and DI instructions so that an interrupt program may not be executed until the main or subroutine program completes execution of the data store operation.

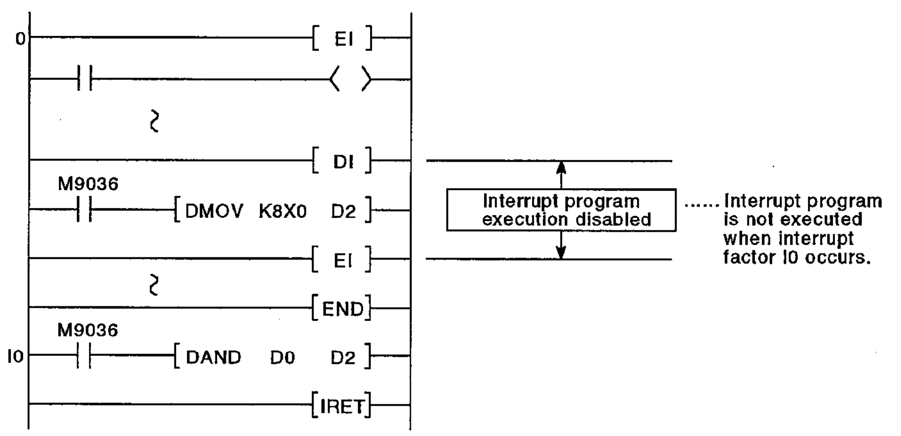

0
EI
DI
M9036
DMOV K8X0 D2
EI
END
M9036
I0
DAND D0 D2
IRET
Interrupt program execution disabled
...... Interrupt program is not executed when interrupt factor I0 occurs.

Fig. 5.7 Setting an Interrupt Program Execution Disable Section

4) Real-time interrupt
Real-time interrupt specified by interrupt pointer numbers I29,I30, and I31 is executed in two different ways according to type of PC CPU as follows.

An: When an interrupt factor occurs during link refresh, the interrupt program is executed after link refresh is completed.

Other than An: When an interrupt factor occurs during link refresh, link refresh is suspended and the interrupt program is executed.

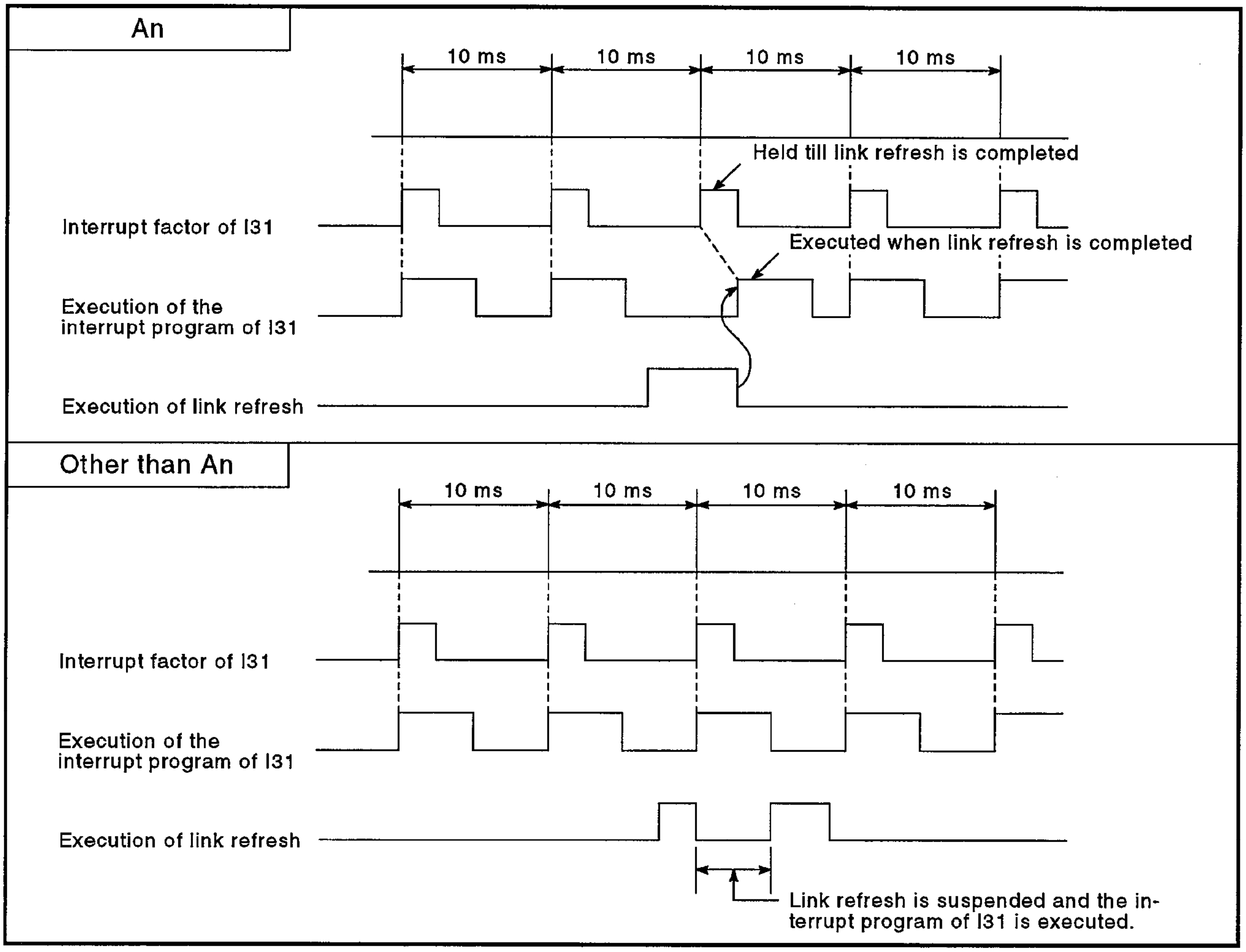

An
10 ms
10 ms
10 ms
10 ms
Held till link refresh is completed
Interrupt factor of I31
Executed when link refresh is completed
Execution of the interrupt program of I31
Execution of link refresh
Other than An
10 ms
10 ms
10 ms
10 ms
Interrupt factor of I31
Execution of the interrupt program of I31
Execution of link refresh
Link refresh is suspended and the interrupt program of I31 is executed.

Fig. 5.8 Timing of Real-time Interrupt

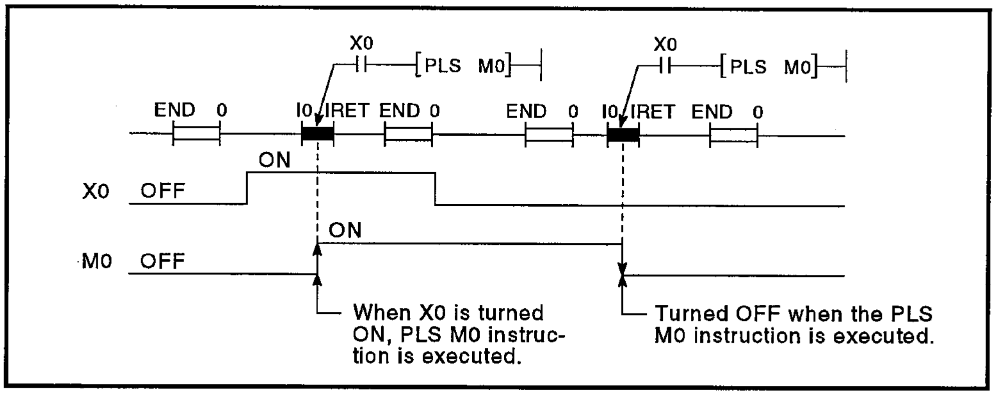
5) Interrupt during END processing

i) If an interrupt factor occurs during END processing of monitoring of other station in a data link system, the interrupt program is executed when monitoring of other stations is completed.

ii) If an interrupt factor occurs during the wait time of the END instruction in constant scan, the interrupt program that corresponds to the factor is executed.

(3) Restrictions on program creation

(a) Devices which are turned ON by the PLS instruction in an interrupt program remain ON until the same interrupt program is executed again.



When X0 is turned ON, PLS M0 instruction is executed.

Turned OFF when the PLS M0 instruction is executed.

(b) When an interrupt program is being executed, interrupt is disabled (DI).
Use caution not to execute an EI or a DI instruction during execution of an interrupt program.

(c) If an interrupt factor occurs when the main program and the subprograms are being executed alternately, the interrupt program is executed as follows.

1) If an interrupt program included in the main program is the same as that included in the sub-program, the interrupt program included in the currently executed program is executed.

2) If one of the following interrupt factors occurs during execution of the main program or the sub-program which does not contain an interrupt program, the following occurs.

- I0 to I23 : An operation error occurs and the PC CPU stops.
- I29 to I31 : Ignored.

(d) Timers cannot be set in the interrupt programs.
If a timer is used in an interrupt program, the program operation may often be disordered. For example, the coil of a timer is OFF when the contact of the timer is ON, or the current value becomes equal with the set value.

(e) Timing of the internal clock of the PC CPU is sometimes suspended by execution of an interrupt program. The timer current value scan time and constant scan time are delayed, as follows, every time an interrupt program is executed.
Check the processing time of each instruction, and create an interrupt program whose processing time is not more than 8 ms.

1) $0 < t < 8$	Not delayed.
2) $8 \leq t < 20$	0 or 10 ms delay depending on timing
3) $20 \leq t < 30$	10 or 20 ms delay depending on timing

(t: Interrupt program processing time)

Type of interrupt program specified by the I number, which causes the timer current value scan time and constant scan time to be delayed, differs with type of PC CPU.

1) A1, A1N, A2, A2N, A3N, A73, A0J2H	I0 to I31
2) A3	None
3) A3H, A3M	I29 to I31

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o*2	o*2	o	o*2	o*2	△*1*3	o*3	o*3	o	o	o*2
Remark	*1: Applicability of the AnA depends on the version. (Refer to the MELSAP II Programming Manual Version E and later.) *2: Refer to page 5-1 for applicable microcomputer programs. *3: Only the SFC programs are applicable.										

5.2 Microcomputer Programs

The microcomputer programs are written in a machine language (machine codes) which can be used with the PC CPU.

(1) Classification

The microcomputer programs are classified as the utility program, SFC program and the user-created microcomputer program.

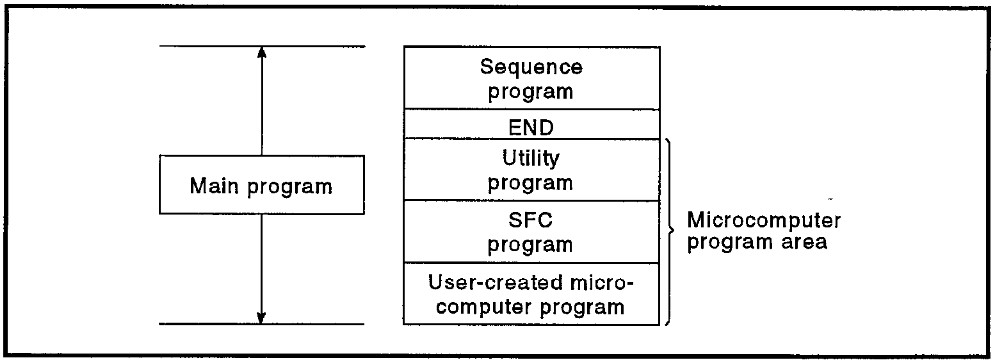
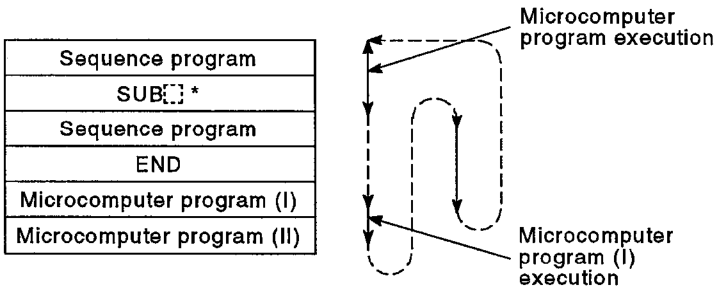

Main program
Sequence program
END
Utility program
SFC program
User-created micro-computer program
Microcomputer program area

Fig. 5.9 Structure of the Microcomputer Programs

(2) Execution

The microcomputer programs are executed when they are called by the SUB instruction in the sequence program. The head address (hexadecimal) of the microcomputer program to be executed is specified by the SUB instruction.


Sequence program
SUB[ ] *
Sequence program
END
Microcomputer program (I)
Microcomputer program (II)
Microcomputer program execution
Microcomputer program (I) execution

* SUB[ ] specifies the head address of the microcomputer program (I).

Fig. 5.10 Execution of the Microcomputer Program

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	○	○	○	○	○	△*1	△*1	△*1	○	○	○
Remark	*1: Only the SFC programs are applicable.										

5.2.1 Utility program

The utility program is provided by Mitsubishi and enables PID control, trigonometric function operations, and other control and operations which cannot be executed by the sequence program.

(1) Storage of the utility program
The utility program must be stored in the microcomputer program area before it is used.
Use the system floppy disk which contains the utility program to store it in the microcomputer program area.
Refer to the Instruction Manual for the utility program to be used for detail.

(2) Precautions

(a) The utility program must be stored at the head address (address 0) in the microcomputer program area.
If the head address of the utility program is other than 0, the utility program cannot be executed.

(b) The utility program cannot be stored in the sub-program microcomputer program area.

5.2.2 User-created microcomputer program

The user-created microcomputer program is written by a user in a machine language which can be used with the PC CPU.
Refer to the ACPU Programming Manual (Common Instructions) (IB-66250) for the specification of the machine language and the creation and storage procedure of the microcomputer program.

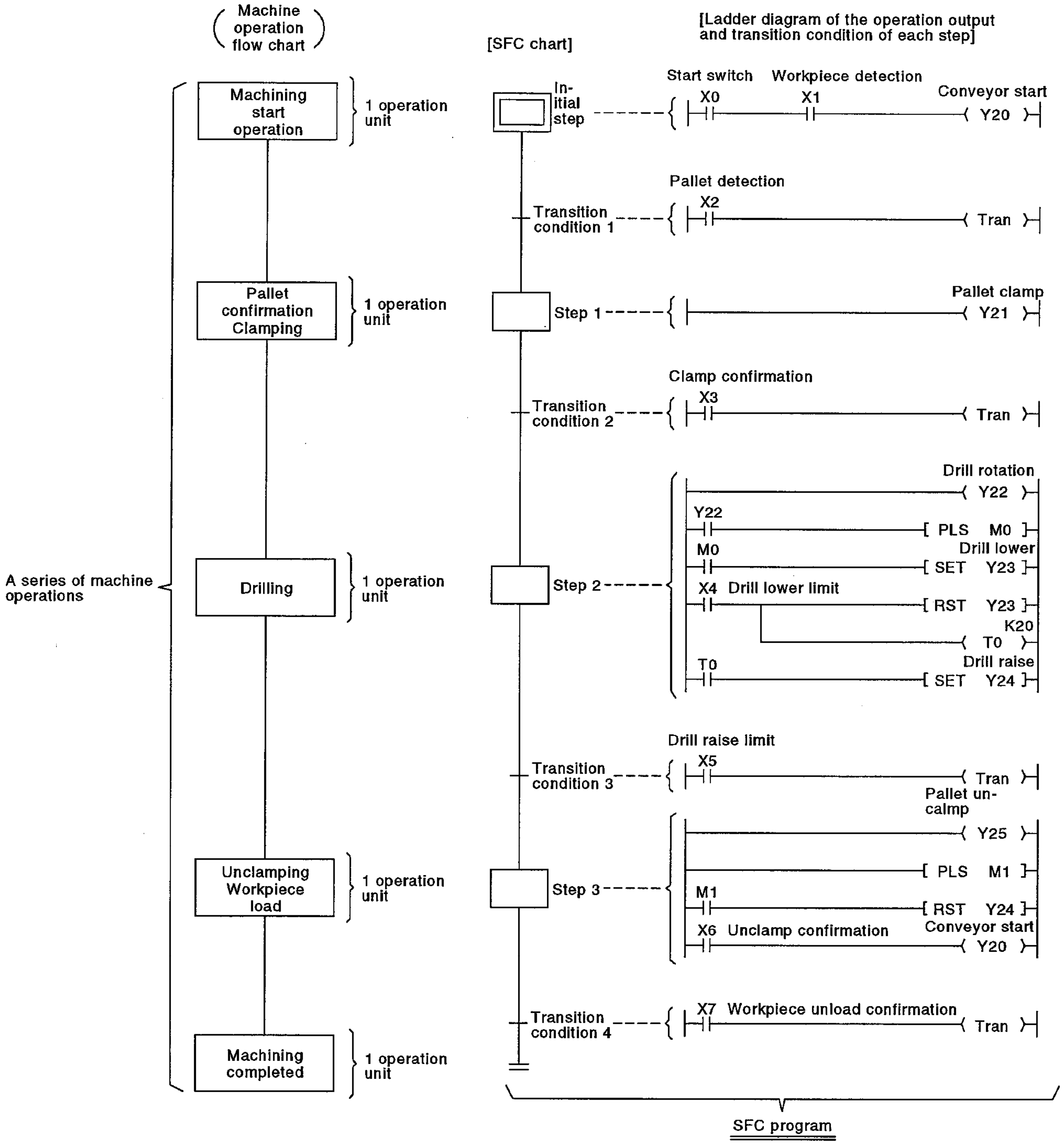
5.2.3 SFC program

(1) What is SFC?
"SFC" is the abbreviation for "Sequential Function Chart" which is a format for describing the control specifications such as program execution sequence and conditions in the form of steps which comprise a series of control operations.

(2) SFC program
In an SFC program, each unit operation of a series of machine operation is described as one step, and a ladder diagram is used for programming each control operation in each step.

REMARK

For the PC CPU types which can execute an SFC program, refer to the MELSAP II Programming Manual (IB-66361) Version B and later.


Machine operation flow chart
[SFC chart]
[Ladder diagram of the operation output and transition condition of each step]
A series of machine operations
Machining start operation
1 operation unit
Initial step
Start switch
Workpiece detection
X0
X1
Conveyor start
Y20
Pallet detection
Transition condition 1
X2
Tran
Pallet confirmation Clamping
1 operation unit
Step 1
Pallet clamp
Y21
Clamp confirmation
Transition condition 2
X3
Tran
Drilling
1 operation unit
Step 2
Drill rotation
Y22
Y22
PLS M0
M0
Drill lower
SET Y23
X4 Drill lower limit
RST Y23
K20
T0
T0
Drill raise
SET Y24
Drill raise limit
Transition condition 3
X5
Tran
Pallet un-calmp
Unclamping Workpiece load
1 operation unit
Step 3
Y25
PLS M1
M1
RST Y24
X6 Unclamp confirmation
Conveyor start
Y20
Transition condition 4
X7 Workpiece unload confirmation
Tran
Machining completed
1 operation unit
SFC program

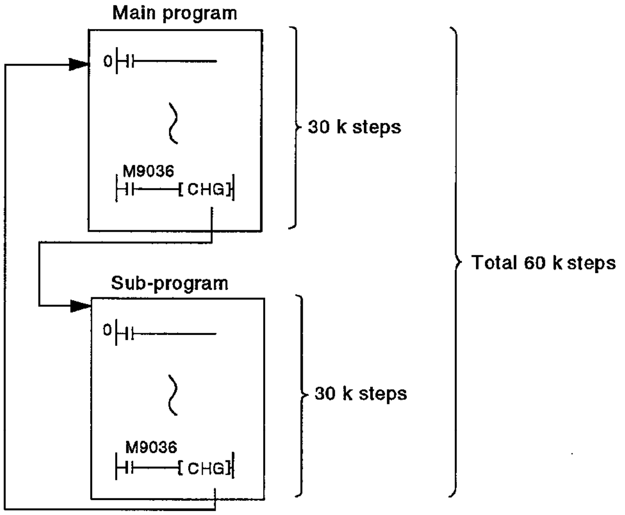
Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	△*1	x	o	o	△*1	△*1	△*1	x	x	o
Remark	*1 : Only the A3, A3N, A3A, A3U, and A4U, Q06H are applicable.										

5.3 How to Use the Sub-Programs

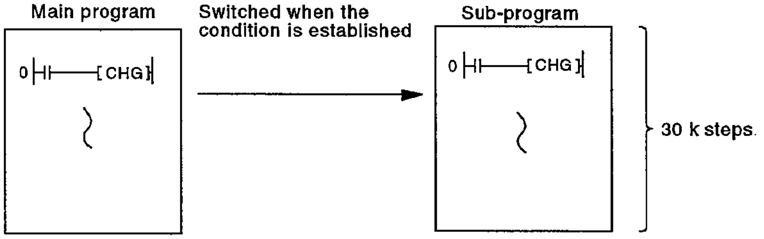
When an A3, A3N, A3H, A3M, A3V, A3A, A3U, A4U, Q06H, or A73 is used, the main and sub program operations can be switched by using a CHG instruction.
[For details of the CHG instruction, refer to the ACPU Programming Manual (Common Instructions) (IB-66250).]

A sub-program can be executed in the form of "serial execution" or "parallel execution".

- Serial execution: This is used when the program capacity (30 k steps) is insufficient.
  By switching the main program and subprogram alternately with a CHG instruction, a program which is larger than 30 k steps can be executed.



- Parallel execution : This is used when the program needs to be switched according to the object of control with one system. Programs stored in the main and subprogram areas are switched by a CHG instruction according to the object of control.



Since the execution condition of the CHG instruction and the configuration of the instruction execution/non-execution result storage memory area differ with the CPU type used, the procedure of use of a sub-program accordingly differs.

(1) The CHG instruction execution condition is provided in the following two different timings:

(a) Execution at leading edge: Executed when the condition is turned from OFF to ON.

(b) Execution during ON : Executed while the condition is ON.

(2) The instruction execution/non-execution result storage area is used with the operating system of the PC CPU for the determination of execution of the PLS and ⬚P instructions in the next scan.

Table 5.2 gives the correspondence between the CPU type and the CHG instruction execution conditions and the instruction execution/non-execution result storage memory area.

Table 5.2 Correspondence between the CPU Type, CHG Instruction Execution Conditions, and the Instruction Execution/Non-Execution Result Storage Memory Area

CPU Type	CHG Instruction Execution Condition		Instruction Execution/Non-Execution Result Storage Area		Refer to
	Execution at Leading Edge	Execution During ON	Main/sub Common	Main/sub Independent	
A3	○	—	○	—	Section 5.3.1
A3N	○	—	—	—	Section 5.3.2
A3V					
A73					
A3H	—	○	—	○	Section 5.3.3
A3M					
A3A					
A3U					
A4U					
Q06H					

POINTS

(1) Devices except pointers (P), interrupt points (I), timers (T), and counters (C) are used in common to the main and sub-programs.
(a) Pointers (P) and interrupt pointers (I) can be designated independently with the main and sub-programs.
(b) Timer (T) numbers and counter (C) numbers used with the main programs cannot be used with the sub-programs.

(2) Refer to the ACPU Programming Manual (Common Instructions) (IB-66250) for the operations of the PLS instruction, counters, and ⬚ P instruction when the CHG instruction is used to switch the operation between the main programs and the sub-programs.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	△*1	x	x	x	x	x	x	x	x	x
Remark	*1 : Only the A3 is applicable.										

5.3.1 When the CHG instruction is executed at the leading edge of the input and the execution/non-execution result storage memory is used commonly with the main programs and the sub-programs

When the CHG instruction is executed at the leading edge of the input and the execution/non-execution result storage memory is used commonly with the main programs and the sub-programs, the operation processing is performed as follows.

(1) The results of execution/non-execution of the main (sub) program instructions stored in the memory are cleared when the main (sub) program is switched to the sub (main) program by the CHG instruction.

(2) When the main (sub) program is switched to the sub (main) program, and then switched again to the main (sub) program, and when an instruction of which execution conditions vary according to the result of the last execution/non-execution of the instruction is executed, the result of the last execution/non-execution of the instruction with the same program is required. By setting M9050 special relay to ON before execution of the CHG instruction, the result of execution/non-execution of instructions stored in the memory area is transferred to the save area. Therefore, the result of execution/non-execution is not cleared though the data in the memory area is cleared by execution of the CHG instruction.

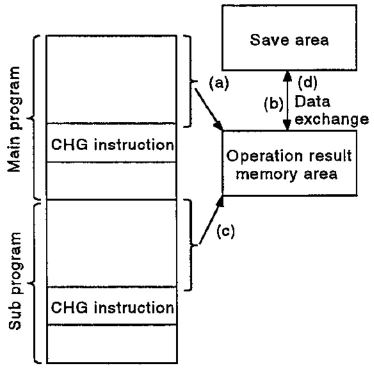
(3) When M9050 is ON

(a) The main program is run with the repeatedly triggered instruction executed in accordance with its input condition and the edge triggered instruction executed in accordance with its input condition and previous operation result.

↓ [CHG] instruction executed

(b) The data in the execution/non-execution result storage memory area is exchanged with that in the save area.

↓



(c) The subprogram is run with the repeatedly triggered instruction executed in accordance with its input condition and the edge triggered instruction executed in accordance with its input condition and previous operation result.

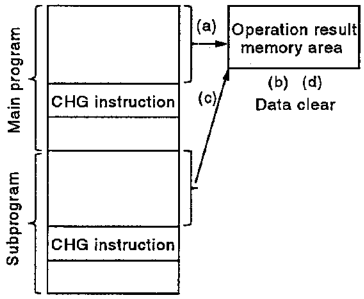
↓ [CHG] instruction executed

(d) The data in the execution/ non-execution result storage memory area is exchanged with that in the save area.

↓
Return to step (a).

(4) When M9050 is OFF

(a) The main program is run with the repeatedly triggered instruction executed in accordance with its input condition and the edge triggered instruction executed in accordance with its input condition and previous operation result.



↓ [CHG] instruction executed

(b) The data in the execution/ non-execution result storage memory area is cleared.

↓

(c) The subprogram is run with the repeatedly triggered instruction executed in accordance with its input condition and the edge triggered instruction executed in accordance with its input condition and previous operation result.

↓ [CHG] instruction executed

(d) The data in the execution/ non-execution result storage memory area is cleared.

↓
Return to step (a).

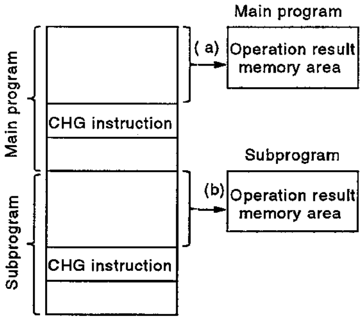
Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	x	x	o*3	o*2	△*1*3	△*1*3	△*1*3	x	x	o*2
Remark	*1 : Only the A3N, A3A, A3U, and A4U, Q06H are applicable. *2 : Refer to Section 5.3.2. *3 : Refer to Section 5.3.3.										

5.3.2 When the CHG instruction is executed at the leading edge of the input and the execution/non-execution result storage memory is sued independently with the main programs and the sub-programs

When the CHG instruction is executed at the leading edge of the input and the execution/non-execution result storage memory is used independently with the main programs and the subprograms, the operation processing is performed as follows.

(a) The main program is run with the repeatedly triggered instruction executed in accordance with its input condition and the edge triggered instruction executed in accordance with its input condition and previous operation result.

↓ [CHG] instruction executed



(b) The subprogram is run with the repeatedly triggered instruction executed in accordance with its input condition and the edge triggered instruction executed in accordance with its input condition and previous operation result.

↓ [CHG] instruction executed

Return to step (a).

5.3.3 When the CHG instruction is executed when the input is ON and the execution/non-execution result storage memory is used independently with the main programs and the sub-programs

When the CHG instruction is executed when the input is ON and the execution/non-execution result storage memory is used independently with the main programs and the sub-programs, the operation processing is performed as follows.

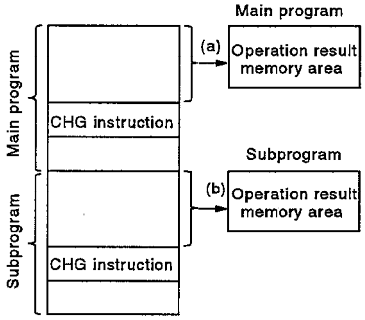
(a) The main program is run with the repeatedly triggered instruction executed in accordance with its input condition and the edge triggered instruction executed in accordance with its input condition and previous operation result.

↓ [CHG] instruction executed

(b) The subprogram is run with the repeatedly triggered instruction executed in accordance with its input condition and the edge triggered instruction executed in accordance with its input condition and previous operation result.

↓ [CHG] instruction executed

Return to step (a).



Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	△*1	x	o	o	△*1	△*1	△*1	x	x	o
Remark	*1: Only the A3, A3N, A3U, A4U and Q06H are applicable.										

5.3.4 Notes on write during run

(1) When a main program and sub program have been created, it is possible to modify or re-write the sub (or main) program while the main (or sub) program is being executed. However, if a CHG instruction is executed at the main (sub) program and a switch made to the sub (main) program during transfer of a program to the CPU from a peripheral device, or when the CPU is modifying pointer (P) or index pointer (I) addresses, correct operation of the sub (main) program will not be possible, and for this reason processing that prohibits the execution of CHG instructions is executed.
However, depending on the combination of the CPU type and the GPP function software package used at the peripheral device, these notes may not apply.

The table below shows whether or not these notes apply with each combination of CPU type and GPP function software package.

GPP Function Software Package	Applicable CPU	
	A3A, A3U, A4U, Q06H	CPU Other than Those to Left
• SW4GP-GPPAEE (software version R or higher) • SW□IVD-GPPA (□ = 1 or higher)	o	x
Software packages other than above	x	x

o : Notes do not apply (write during run is possible even during execution of a CHG instruction)
x : Notes apply (processing to prohibit execution of the CHG instruction when write during run is executed is performed)

When special relays M9051, M9056, M9057 are used as the execution condition for the CHG instruction, the execution status of the CHG instruction can be confirmed.

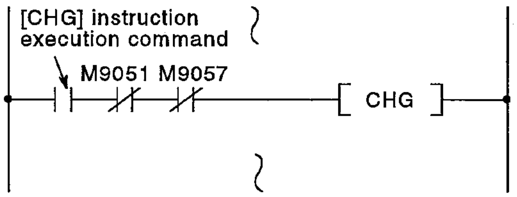
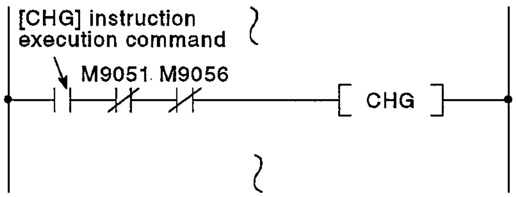
	Main Sequence Program	Subsequence Program
Ladder example		
Special relays	M9051 ..... Switched on during main (sub) sequence program transfer to the CPU. Automatically switched off when the transfer is complete. M9056 ..... Switched on on completion of the main sequence program transfer to the CPU. Automatically switched off on completion of pointer (P) or interrupt pointer (I) address storage. M9057 ..... Switched on on completion of the subsequence program transfer to the CPU. Automatically switched off on completion of pointer (P) or interrupt pointer (I) address storage.	

Fig. 5.11 Interlocking the [CHG] Instruction Execution Conditions

(2) The main and sub programs are rewritten as shown below.

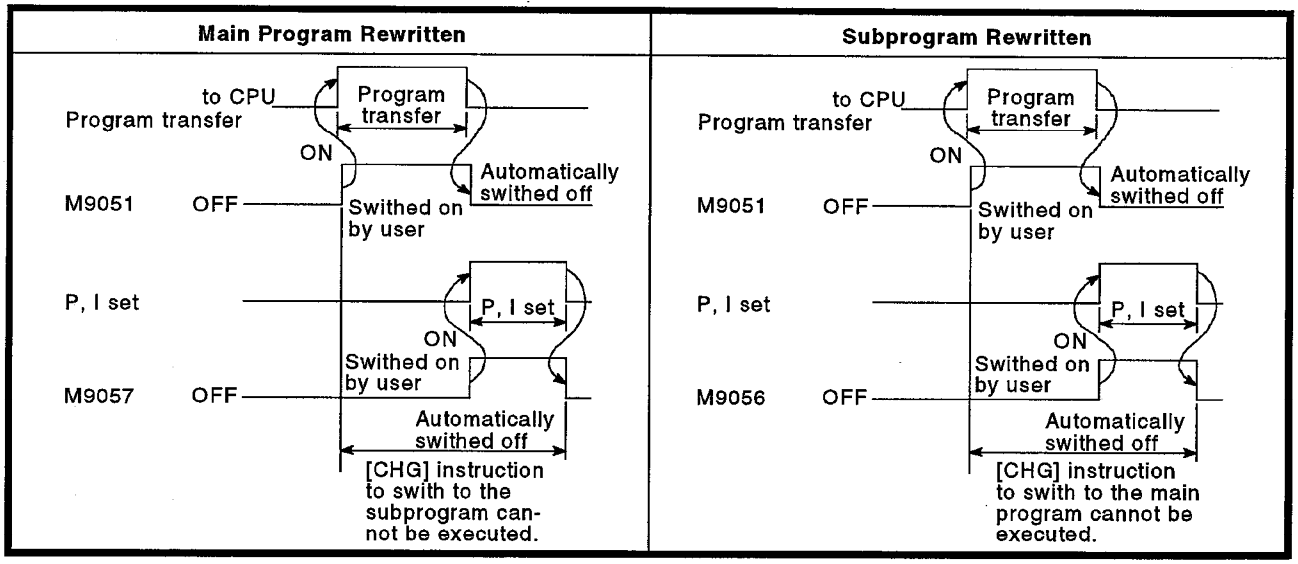
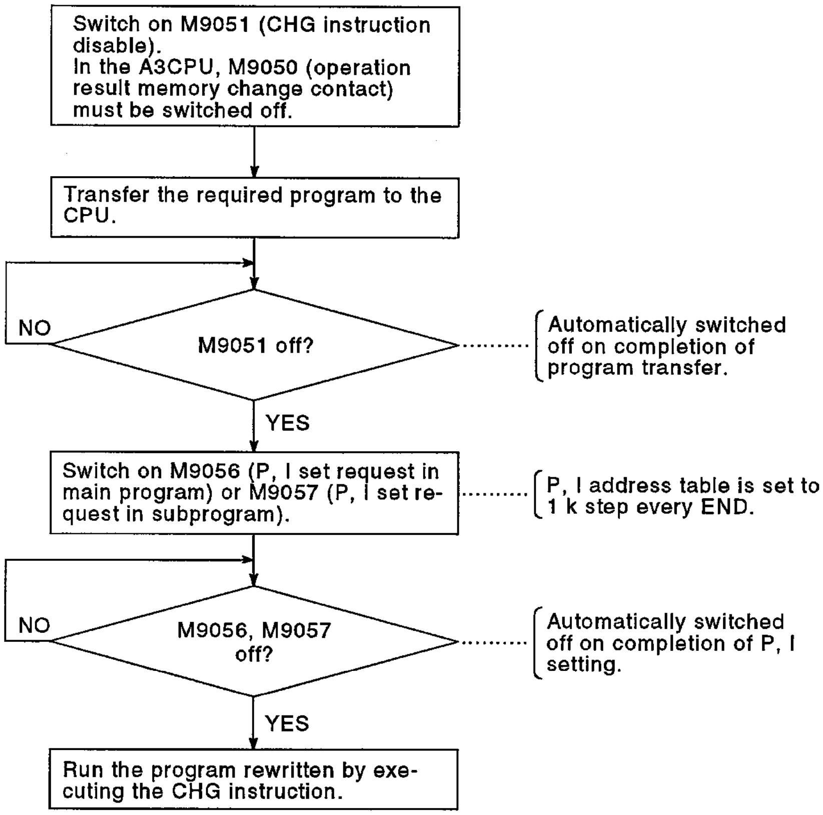


Fig. 5.12 Program Rewrite Timing Chart

(3) Main (sub) program rewrite procedure


Switch on M9051 (CHG instruction disable).
In the A3CPU, M9050 (operation result memory change contact) must be switched off.
Transfer the required program to the CPU.
NO
M9051 off?
Automatically switched off on completion of program transfer.
YES
Switch on M9056 (P, I set request in main program) or M9057 (P, I set request in subprogram).
P, I address table is set to 1 k step every END.
NO
M9056, M9057 off?
Automatically switched off on completion of P, I setting.
YES
Run the program rewritten by executing the CHG instruction.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	△*1	x	o	o	△*1	△*1	△*1	x	x	o
Remark	*1: Only the A3, A3N, A3U, A4U and Q06H are applicable.										

5.3.5 Notes on writing subprogram

(1) When there is an interrupt program, the same interrupt programs must be written in the main and subsequence programs using the same interrupt pointer numbers.

(2) A timer and a counter cannot be used with the same device number in the main and subsequence programs.
If the timer and counter numbers are the same, the RST T/C instruction may be used in the main (sub) sequence program when the OUT T/C instruction exists in the (sub) sequence program because the RST T/C instruction resets the T/C present value on completion of its execution.

(3) Any interrupt is disabled while the CHG instruction is being executed. Hence, an interrupt program is only executed after the CHG instruction is executed if the corresponding interrupt factor occurs.

(4) The pointer (P) indicating the destination of the branch instruction (CJ, SCJ, CALL, JMP)may be used with the same numbers in both the main and subsequence programs.

6. FUNCTIONS

Table 6.1 gives principal functions of the CPU module.

Table 6.1 Table of Functions

Function	Description
Constant scan	• The sequence program is executed at constant intervals regardless of scan time.
Latch (power-failure backup)	• Contents of devices are retained when the power is cut OFF, when the module is reset, or when a momentary power failure over 20 ms occurred. • Devices L, B, T, C, D, and W can be latched.
Remote RUN/STOP	• Remote RUN/STOP can be instructed from a peripheral device when the RUN keyswitch is set to RUN.
PAUSE	• The CPU operation is stopped retaining the output (Y) state.
Status latch	• Contents of all devices at the time when the status latch condition is established are stored in the status latch area in a memory cassette. • The contents of devices stored in the status latch area can be monitored with a peripheral device.
Sampling trace	• Operating state of specified devices is sampled at specified intervals and the result of sampling is stored in the sampling trace area in a memory cassette. • Data stored in the sampling trace area can be monitored with a peripheral device.
Offline switch	• Devices (Y, M, L, S, F, B) used with the OUT instruction can be separated from the sequence program operation.
Step run	• The sequence program stops after execution of each instruction. • Step run operation is classified into two different methods as follows. • By specified loop count • By each instruction
Clock	• The clock operation is enabled in the CPU module. • The clock data is controlled in year, month, date, hour, minute, second, and day of the week. • The clock data can be stored in special registers D9025 to D9028.
Online I/O module replacement	• The I/O modules can be replaced when the CPU is running (power ON).
Interrupt processing	• When an interrupt factor occurs, the corresponding interrupt program is executed with priority over the sequence program.
Comment	• Comments of meaning and purposes are provided for each device and printed or monitored with a peripheral device to help recognition of ladder blocks.
Watchdog timer (10 to 2000 ms)	• An internal timer used for hardware or software faults of the PC CPU. Setting value is variable.
Microcomputer mode	• The utility programs and the user-created microcomputer programs are stored in the microcomputer program area and are called by the sequence program for various control operations and processings.
Self-diagnosis	• The CPU checks itself and detects and indicates occurrences of errors or stops CPU operations.
Output setting at STOP → RUN	• The state of output (Y) when operation state is switched from STOP to RUN.
Entry code registration	• Read and write of the programs (main/sub-programs and parameters) and comments from the PC CPU are disabled.
Print title registration	• Comments for the system names and program names are registered and printed.
Annunciator display mode	• Comments and the F numbers are displayed on the LED indicators when the annunciators are set.
ERROR LED priority setting	• Whether the ERROR LED at an error occurrence is lit or unlit is set.

	A1S(S1), A1SJ-S3, A1SJH(S8), A1SH, A2S(S1), A2SH(S1), A1FX	A1	A2(S1)	A3	A1N	A2N(S1)	A3N, A73	A3V	A3H, A3M	A2A(S1)	A2AS(S1/S30) A2U, A2USH-S1	Q02, Q02H, Q06H	A3A, A3U, A4U	A0J2H	A2C, A52G	Refer to
	O	—	—	—	O	O	O	O	O	O			O	O	O	Sec. 6.1
	O	O	O	O	O	O	O	O	O	O			O	O	O	Sec. 6.2
	O	O	O	O	O	O	O	O	O	O			O	O	O	Sec. 6.3
	O	O	O	O	O	O	O	O	O	O			O	O	O	Sec. 6.4
	O	—	O	O	—	O	O	O	O	O			O	O	O	Sec. 6.5
	O	—	O	O	—	O	O	O	O	O			O	O	O	Sec. 6.6
	O	O	O	O	O	O	O	O	—	—			—	O	O	Sec. 6.7
	—	O	O	O	O	O	O	O	O	O			O	—	—	Sec. 6.8
	O	—	—	—	O	O	O	O	—	O			O	—	△*1	Sec. 6.9
	—	—	—	—	O	O	O	—	—	O			O	—	—	Sec. 6.10
	O	O	O	O	O	O	O	O	O	O			O	O	—	Sec. 5.1.13
	O	O	O	O	O	O	O	O	O	O			O	O	O	Sec. 6.11
	O	O	O	O	O	O	O	O	200 ms fixed					O	O	Sec. 6.12
	O	O	O	O	O	O	O	O	O	—			—	O	O	Sec. 5.2
	O	O	O	O	O	O	O	O	O	O			O	O	O	Sec. 6.13
	O	O	O	O	O	O	O	O	O	O			O	O	O	Sec. 6.14
	O	O	O	O	O	O	O	O	O	O			O	O	O	Sec. 6.15
	O	O	O	O	O	O	O	O	O	O			O	O	O	Sec. 6.16
	—	—	—	O	—	—	O	O	O	—			O	—	—	Sec. 6.17
	O	—	—	—	—	—	—	—	—	O			O	O	O	Sec. 6.18

*1: Only the A2CCPUC24(-PRF) and A52GCPU(-T21B) have the clock function.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	x	o	o	o	o	o	o	o	o	o
Remark											

6.1 Constant Scan

(1) Constant scan
Scan time varies with the number of instructions executed in each scan. The constant scan function executes a sequence program at constant intervals.
By use of this function, a sequence program can be executed maintaining constant scan time.

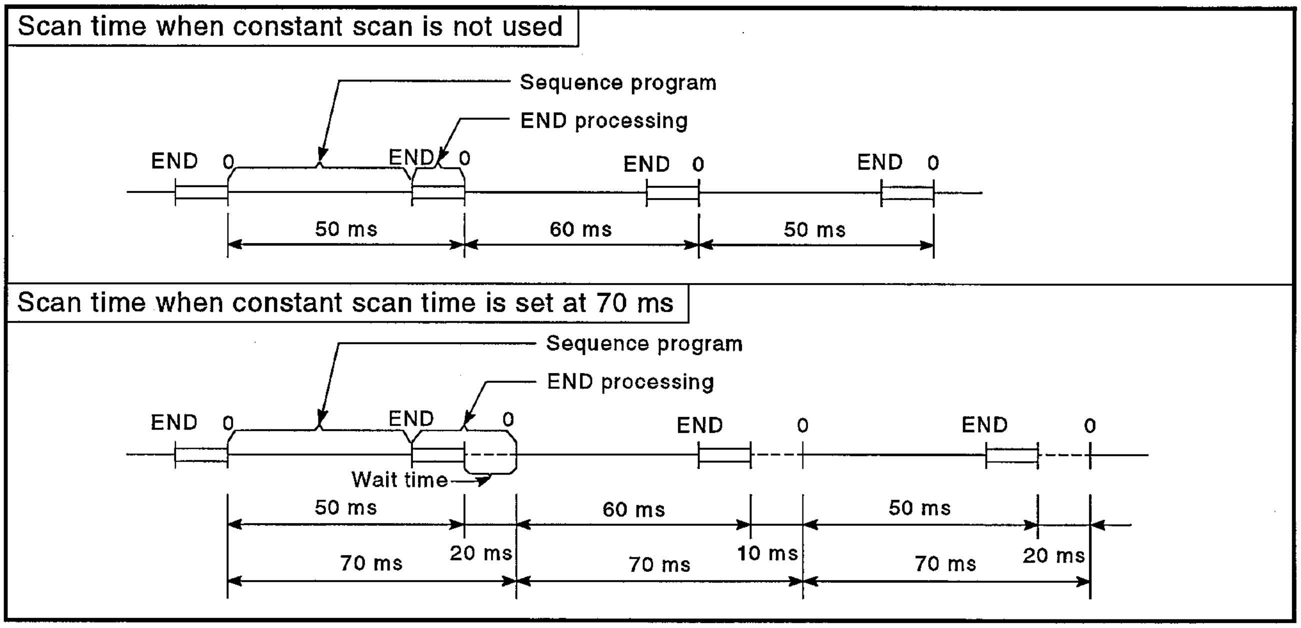
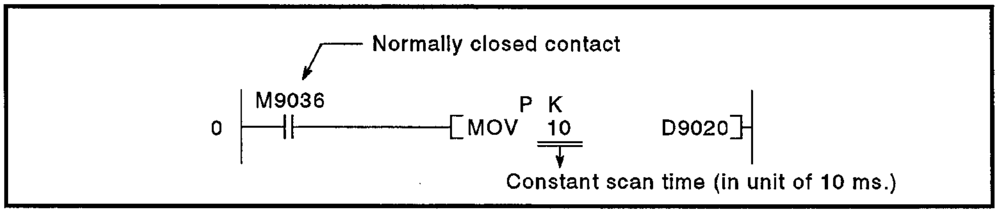


Fig. 6.1 Constant Scan Operation

(2) Setting constant scan time

(a) Constant scan time should be set with special register D9020.
Constant scan begins when the constant scan time is set.
Content of D9020 is cleared when power is turned ON or when the CPU is reset.
To begin constant scan when power is turned ON or when the CPU is reset, write the following program at the beginning of the sequence program.



(b) If scan time of a sequence program is longer than set constant scan time, the set constant scan time is ignored and the sequence program is executed according to its scan time.

(c) Set a value greater than the maximum sequence program scan time for the constant scan setting.
If the sequence program scan time is longer than the setting for constant scan, processing will be performed in accordance with the sequence program scan time and the constant scan function will not operate normally.

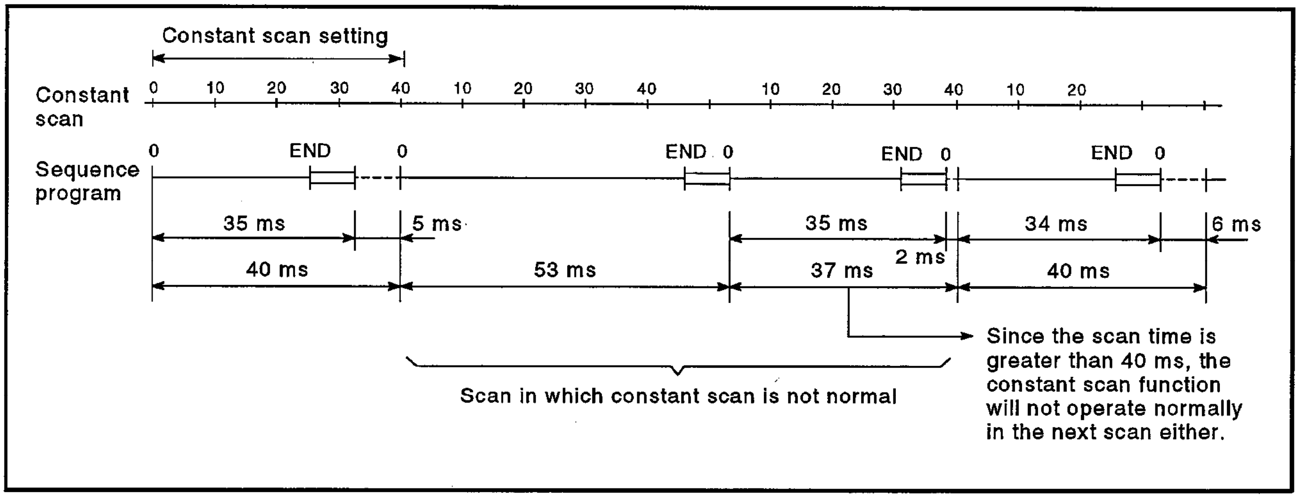

Constant scan setting
Constant scan
0
10
20
30
40
10
20
30
40
10
20
30
40
10
20
Sequence program
0
END
0
END
0
END
0
END
0
35 ms
5 ms
40 ms
53 ms
35 ms
2 ms
37 ms
34 ms
6 ms
40 ms
Since the scan time is greater than 40 ms, the constant scan function will not operate normally in the next scan either.
Scan in which constant scan is not normal

Fig. 6.2 Operation when Scan Time is Longer than Constant Scan Time

(d) During the interval between execution of the sequence program END instruction and the start of the next scan, sequence program processing is stopped (each device retains the status it had before execution of the END instruction.)
However, when an interrupt cause occurs, the interrupt program is executed.

(3) Notes on constant

(a) The relationship between the set value for constant scan and the set time for WDT (watchdog timer) is given by the expression below.
If the set value is longer than the WDT set time, a WDT error occurs.

(Constant scan set value) ≤ (WDT set value) — 1

Applicable CPU	**All Types of CPUs**
Remark	

6.2 Retaining Device Data (Latch Function)

Data of all devices is cleared to the initial value (bit devices: OFF, word devices: 0) when the PC CPU is turned ON, when the RUN keyswitch is turned to RESET, or when a momentary power failure of more than 20 ms occurs.

The latch function is used to retain device data when the PC CPU is turned ON, when the RUN keyswitch is turned to RESET, or when a momentary power failure of more than 20 ms occurs.

The sequence program operation is not influenced by the latch function.

(1) Purpose

The latch function is used, when continuous control is being performed, to retain data such as number of products, number of defectives and addresses even when a momentary power failure of more than 20 ms occurs and to enable continuous control.

(2) Devices usable with the latch function

(a) The following devices are usable.

1) Latch relays

2) Link relays

3) Timers

4) Counters

5) Data registers

6) Link registers

(b) Latch range is set with parameters on a peripheral device. Refer to Chapter 8 for the latch range of each device.

POINT

Device data within the latch range is retained by the battery (A6BAT) attached to the CPU module or the memory cassette.

(1) If the sequence program operation is performed with the sequence program stored in a ROM, a backup battery is necessary to enable the latch function.

(2) If the battery connector is uncoupled from the CPU module or the memory cassette when the CPU module or the memory cassette is being turned OFF, device data within the latch range is destroyed.

(3) Clearing device data within the latch range

(a) To clear device data within the latch range and to set the initial data, execute "LATCH CLEAR". When LATCH CLEAR is executed, device data out of the latch range is also cleared as follows.

1) Y, M/L/S, F, B............. Turned OFF.

2) Special relays
M9000 to M9255......... Retained.

3) T,C............................. Contacts and coils are turned OFF and current value becomes 0.

4) D, Z, V, W, A...............Become 0.

5) R................................ Retained.

6) Special registers
D9000 to D9255.......... Retained.

(b) The method of latch clear differs with type of CPU.
For the method of latch clear, refer to the User's Manual for the PC CPU to be used.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	o	o	o	o	o	o	△*1	o
Remark	*1 : Remote RUN/STOP by a computer is disabled.										

6.3 PC CPU RUN/STOP with a Peripheral Device (Remote RUN/STOP)

The RUN keyswitch is used to RUN/STOP the PC CPU.
Remote RUN/STOP is executed with a peripheral device, remote RUN contacts, or computers with the RUN keyswitch set in the RUN position.

(1) Purpose

Remote RUN/STOP is used in the following cases.

(a) The PC CPU is located out of reach of operators.

(b) To RUN/STOP the PC CPU mounted in a control panel with a device located outside the control panel.

(2) Remote RUN/STOP operation

The sequence program operation of remote RUN/STOP is as follows.

(a) Remote STOP........ The sequence program operation stops after execution of the END instruction.

(b) Remote RUN.......... When remote RUN is executed when the sequence program operation is in the stop state caused by remote STOP, the sequence program operation begins with step 0.

(3) Remote RUN/STOP procedures

The following procedures are used to enable remote STOP/RUN.

(a) Using the remote RUN contact
Remote RUN/STOP can be controlled by turning the remote RUN contact ON/OFF.

1) When the remote RUN contact is OFF, remote RUN is enabled.

2) When the remote RUN contact is ON, remote STOP is enabled.

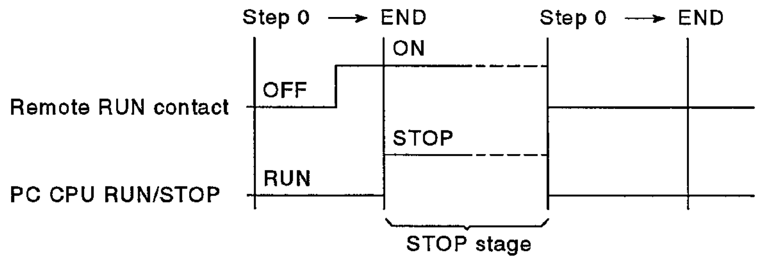


Fig. 6.3 RUN/STOP Timing when the Remote RUN Contact is Used

POINT

Use parameter setting to set the remote RUN contacts.
Refer to Chapter 8 for the remote RUN contacts that can be set.

(b) Using a peripheral device or a computer
Remote RUN/STOP can be controlled with a peripheral device or a computer as shown below.

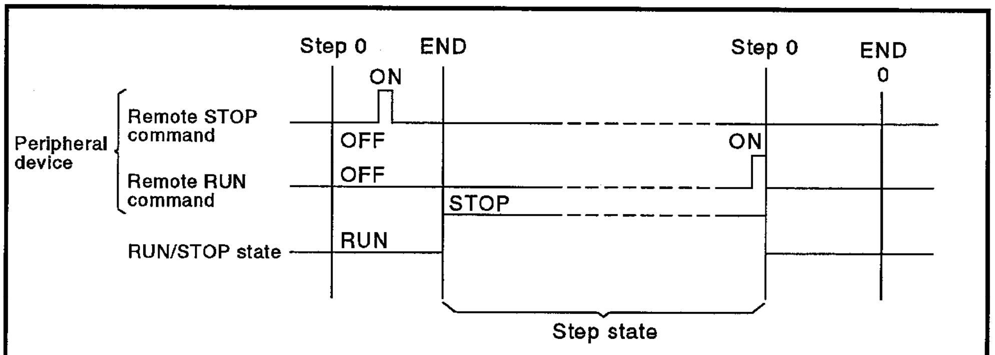


Fig. 6.4 RUN/STOP Timing when a Peripheral Device or a Computer is Used

(4) Precautions

Since priority is given to the STOP state, the following points should be considered.

(a) When remote STOP is executed with a remote RUN contact, a peripheral device, or a computer, the PC CPU goes into the STOP state.

(b) To return the PC CPU from the STOP state caused by remote STOP to the RUN state, set all the external factors (remote RUN contacts, peripheral devices, computers, etc.) which were used for remote STOP to the RUN state.

REMARK

The RUN and STOP states are described as follows.

- RUN state.............. The sequence program is executed repeatedly from step 0 to the END instruction.
- STOP state........... The sequence program operation is being stopped. All outputs (Y) are turned OFF.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	o	o	o	o	△	△	△	o
Remark	△ : PAUSE with the RUN keyswitch is disabled.										

6.4 Stopping the Sequence Program Operation Retaining the State of Outputs (PAUSE)

The PAUSE function is used to stop the sequence program operation while retaining the ON/OFF state of all outputs (Y).
The following methods are used to enable PAUSE.

1) Using the RUN keyswitch
2) Using the remote PAUSE contact
3) Using the GPP/HGP/PHP

(1) Purpose

The PAUSE function is useful when outputs (Y) need to remain ON when the PC CPU is stopped.

(2) Using the RUN keyswitch

(a) When M9040 is turned ON when the RUN keyswitch is being set in the PAUSE position, the PAUSE state contact (M9041) is turned ON after execution of the END instruction of the next scan. When the END instruction of the scan after the scan in which the PAUSE state contact is turned ON is executed, the PC CPU goes into PAUSE and stops the sequence program operation.

(b) Either set the RUN keyswitch in the RUN position or turn M9040 OFF with a peripheral device to restart the sequence program operation.

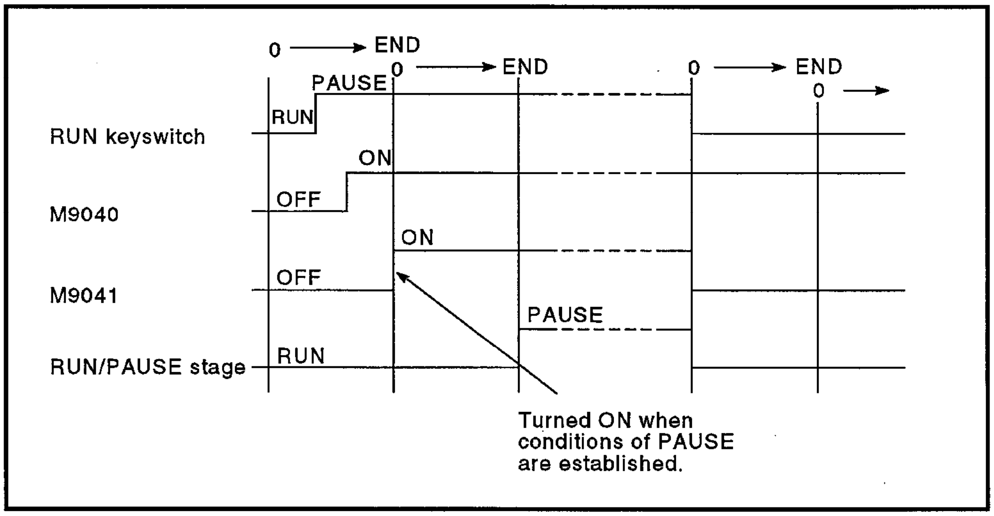


Fig. 6.5 PAUSE Timing when the RUN Keyswitch is Used.

(3) Using the remote PAUSE contact

(a) When the remote PAUSE contact and the PAUSE enable flag (M9040) are turned ON, the PAUSE state contact (M9041) is turned ON after execution of the END instruction of the scan. When the END instruction of the scan after the scan in which the PAUSE state contact is turned ON is executed, the PC CPU goes into PAUSE and stops the sequence program operation.

(b) Either turn OFF the remote PAUSE contact or turn M9040 OFF with a peripheral device to restart the sequence program operation.

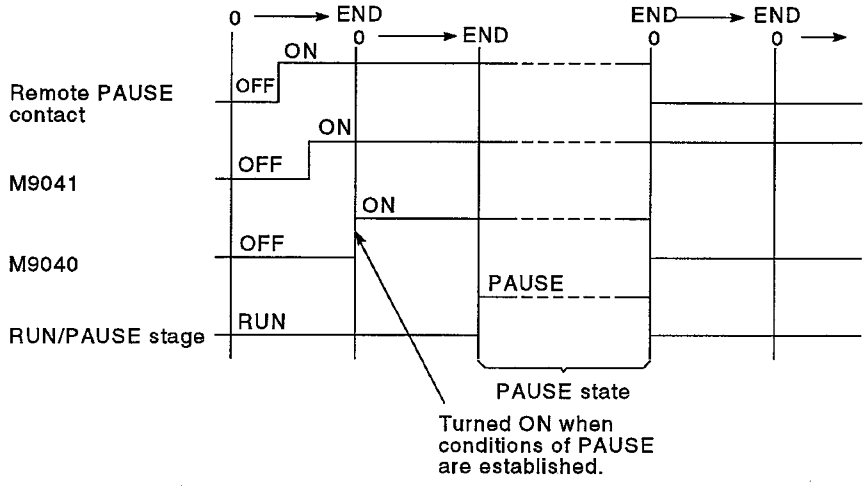


Fig. 6.6 PAUSE Timing when the Remote PAUSE Contact is Used.

POINT

Use parameter setting to set the remote PAUSE contacts.
Refer to Chapter 8 for the remote PAUSE contacts that can be set.

(4) Using a peripheral device

(a) The PAUSE state contact (M9041) is turned ON after execution of the END instruction of the scan in which a remote PAUSE command from a peripheral device is received.
When the END instruction of the scan after the scan in which the PAUSE state contact is turned ON is executed, the PC CPU goes into PAUSE and stops the sequence program operation.

(b) When a remote RUN command from a peripheral device is received, the PC CPU restarts the sequence program operation beginning with step 0.

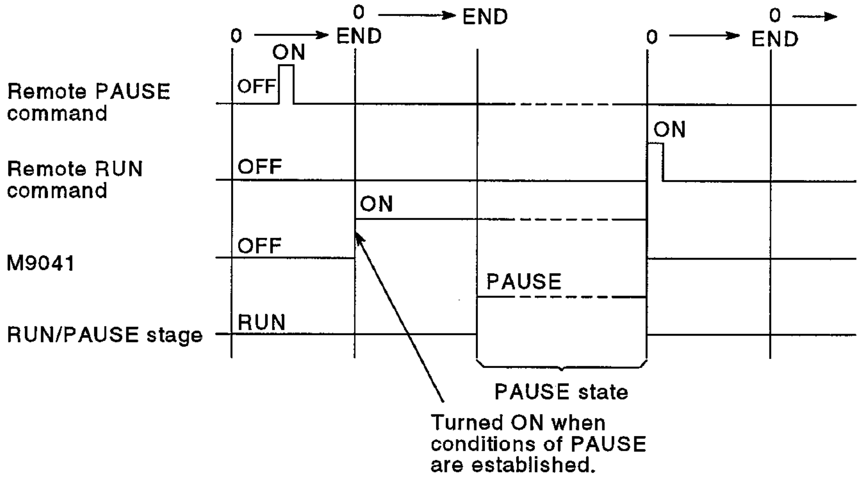
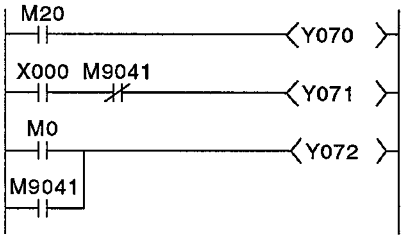


Fig. 6.6 PAUSE Timing when the Remote PAUSE Contact is Used.

POINT

To set the state of output (Y) to ON or OFF in the PAUSE state, use the PAUSE state contact (M9041) as interlock.



In the PAUSE state, the ON/OFF state of M20 determines the ON/Off state of Y70.

Set to OFF in the PAUSE state.

Set to ON in the PAUSE state.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	△*1	o	o	o	o	o	o	o	o	o
Remark	*1 : A1 and A1N are unusable.										

6.5 Retaining Device Data when a Specific Condition is Established (Status Latch)

Device data at a specific moment cannot be checked by monitoring with a peripheral device.
If the status latch function is used, data of all devices when the SLT instruction is executed is transmitted and stored to preset status latch area.
Data stored by status latch can be monitored with the peripheral device.

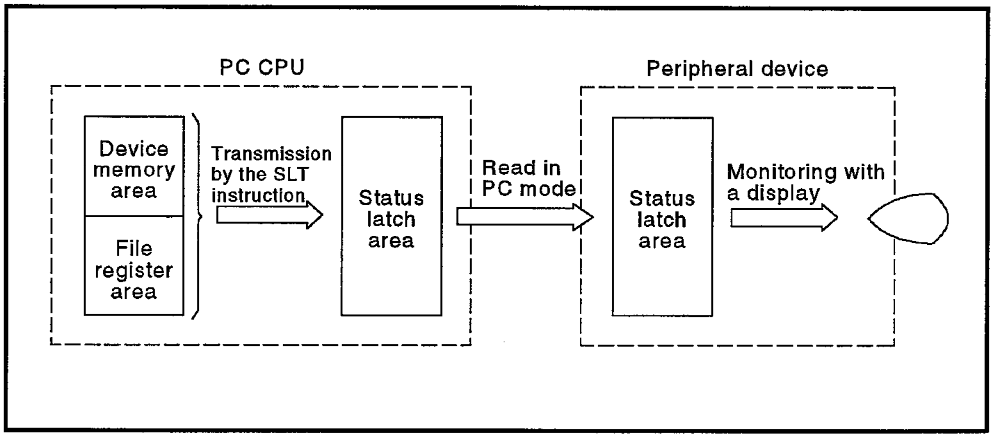


Fig. 6.8 Status Latch Processing

(1) Purpose

Status latch is useful if device data when an error occurs needs to be stored and monitored.
It is also useful to check the cause of occurrence of an error by setting the SLT instruction to be executed when a condition of error is established during the sequence program operation.

(2) Status latch processing

(a) The following data is stored in the status latch area when the SLT instruction is executed.

1) Device memory
X, Y, M, L, S, F, B........... ON/OFF data
T,C................................ Contact and coil ON/OFF data and current values
D,W,A,Z,V....................... Stored data

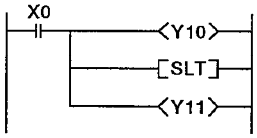
2) File registers (R)............ Stored data

(b) Data is stored in the status latch area when the SLT instruction is executed.
If there are outputs which are turned ON/OFF or if device data is stored using the same input condition, data stored before execution of the SLT instruction differs from that stored after execution of the SLT instruction.

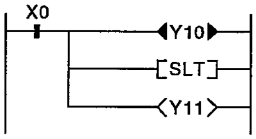
Example

If there are programs, before and after an SLT instruction, which are turned ON/OFF by the same input, the ON and OFF states of the outputs are displayed differently between them.

[Example]



[Display when status latch data is monitored]



Since Y10 is ON when the SLT instruction is executed, it is displayed as ON.

Since Y11 is ON when the SLT instruction is executed, it is displayed as ON.

(3) Precautions

Scan time increases each time an SLT instruction is executed.
Set the time of the WDT and constant scan considering the time increase due to execution of the SLT instruction.
Refer to the ACPU Programming Manual (Common Instructions) (IB-66250) for the execution time of the SLT instruction.

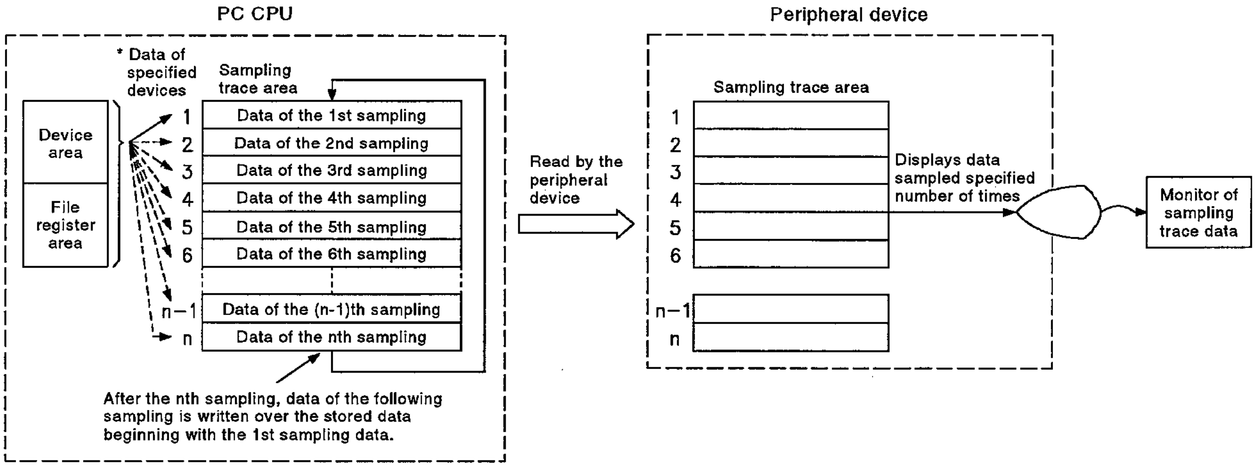
POINT

Use parameter setting to set the devices used for status latch.
Refer to Chapter 8 for the setting of the devices used for status latch.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	△*1	○	○	○	○	○	○	○	○	○
Remark	*1 : A1 and A1N are unusable.										

6.6 Sampling Device Data at Constant Intervals (Sampling Trace)

Changes of the ON/OFF status of bit devices and word device data cannot be checked by monitoring with a peripheral device.
Sampling trace is used to sample and store data of specified devices to the sampling trace area at constant intervals.
Data stored in the sampling trace area is sampled specified number of times and latched when the STRA instruction is executed.
Data stored in the sampling trace area can be monitored with the peripheral device.


PC CPU
* Data of specified devices
Sampling trace area
Device area
File register area
1
2
3
4
5
6
n−1
n
Data of the 1st sampling
Data of the 2nd sampling
Data of the 3rd sampling
Data of the 4th sampling
Data of the 5th sampling
Data of the 6th sampling
Data of the (n-1)th sampling
Data of the nth sampling
After the nth sampling, data of the following sampling is written over the stored data beginning with the 1st sampling data.
Read by the peripheral device
Peripheral device
Sampling trace area
1
2
3
4
5
6
n−1
n
Displays data sampled specified number of times
Monitor of sampling trace data

* When the STRA instruction is executed, data of specified devices is sampled specified number of times and stored to the sampling trace area and latched.

Fig. 6.9 Sampling Trace Processing

(1) Purpose

Sampling trace is useful to check data of specified devices sampled at constant intervals. This is effective to save time when correcting a sequence program.

(2) Devices usable with sampling trace

The following devices are usable.

(a) Bit devices (X, Y, M, L, S, F, B, and contacts and coils of T and C)

(b) Word devices (Current value of T and C, and D, W, R, A, Z, V)

(3) Number of times of sampling

The number of times of sampling is set for the total number of times and for that after the STRA instruction execution.

(a) Setting the total number of times sets the sampling trace area. This can be set by 128 times within the range of 0 to 1024 times.

(b) Setting the number of times after the STRA instruction execution enables completion of sampling trace and data latch after the STRA instruction execution.
This can be set by 128 times within the range of 0 to 1024 times.

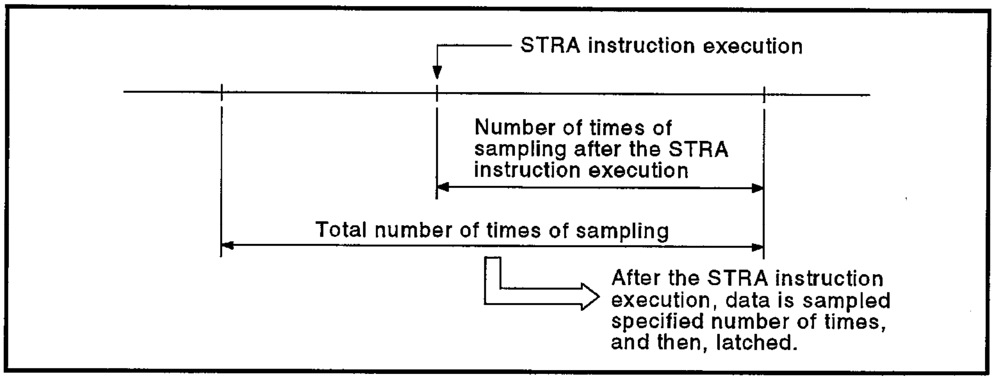

STRA instruction execution
Number of times of sampling after the STRA instruction execution
Total number of times of sampling
After the STRA instruction execution, data is sampled specified number of times, and then, latched.

Fig. 6.10 Number of Times of Sampling

(c) Setting of the number of times of sampling is provided as follows.

$$\left(\begin{array}{l}\text{Number of times of sampling after}\\ \text{the STRA instruction execution}\end{array}\right) \leq \left(\begin{array}{l}\text{Total number of}\\ \text{times of sampling}\end{array}\right) \leq \text{1024 times}$$

(4) Sampling interval

The sampling interval sets the timing of sampling. The following two kinds of settings are used.

(a) Sampling by END
Sampling trace is executed after execution of every END instruction of the sequence program.

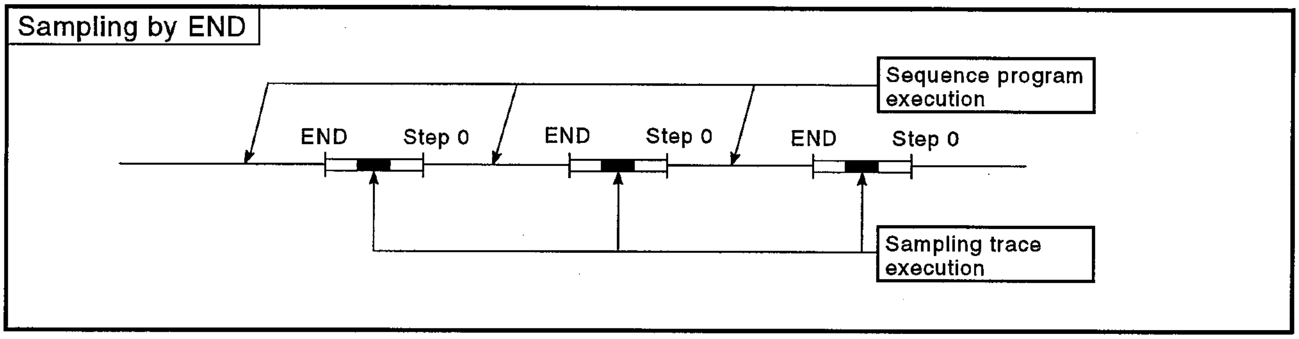


Fig. 6.11 Sampling Trace Execution by Every END

(b) Sampling at specified intervals
Sampling trace is executed every 10 x n ms.
(Sampling trace is executed also when other instructions of the sequence program are being executed.)

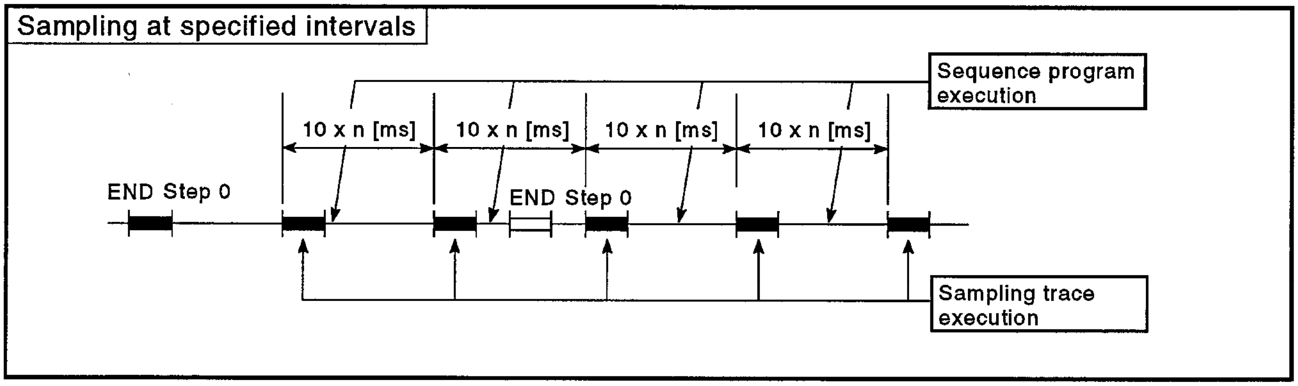
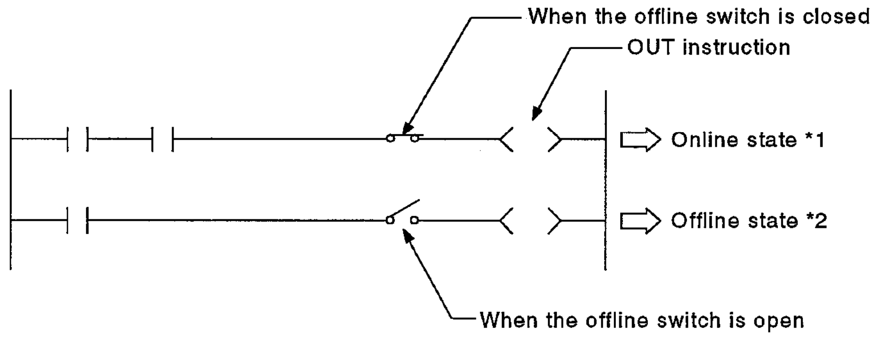


Fig. 6.12 Sampling Trace at Specified Intervals

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	o	o	o	o	x	x	x	o	o	o
Remark											

6.7 Forced ON/OFF of the OUT Instruction with a Peripheral Device in the RUN Stare (Offline Switch)

When the PC CPU is in the RUN state (when the sequence program operation is being executed), the devices used with the OUT instruction of the sequence program cannot be turned ON/OFF with a peripheral device using the test function.
The offline switch function is used to separate the OUT instruction execution from the sequence program operation and enables turning ON and OFF of the devices used with the OUT instruction with a peripheral device using the test function even when the PC CPU is in the RUN state.
The offline switch function is useful to check operation of the output modules and wiring between the output modules and peripheral devices using the test function on a peripheral device when the sequence program operation is being executed.


When the offline switch is closed
OUT instruction
Online state *1
Offline state *2
When the offline switch is open

*1...Online state is established when the offline switch is closed.
Devices used with the OUT instruction are turned ON/OFF by the sequence program operation.
*2...Offline state is established when the offline switch is open. The OUT instruction execution is separated from the sequence program operation.
(Devices are not turned ON/OFF by the sequence program operation.)

Fig. 6.13 Offline State and Online State

(1) Devices usable with the offline switch function

The following devices used with the OUT instruction can be used with the offline switch function.

(a) Outputs (Y)

(b) Internal relays (M)

(c) Latch relays (L)

(d) Step relays (S)

(e) Link relays (B)

(f) Annunciators (F)

(2) State of devices on offline state

State of devices in offline state (when the offline switch is open) is as follows.

(a) State of devices before the offline switch function is used is retained.

(b) If forcible set/reset of devices is executed with a peripheral device in offline state, the forcibly set/reset state of devices is retained.

POINTS

(1) Devices in offline state cannot be turned ON/OFF by the sequence program operation.
If devices are set to offline in a test run of the CPU, reset them online by disabling the offline switch function after the test run.

(2) Devices reset from offline to online can be turned ON/OFF by the sequence program operation.
To set the devices online, check the input condition of the OUT instruction and make sure that there is no problem when they are reset online.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	o	x	o	o	o	o	o	x	x	o
Remark	*1 : Only the AnN is applicable.										

6.8 Step Operation

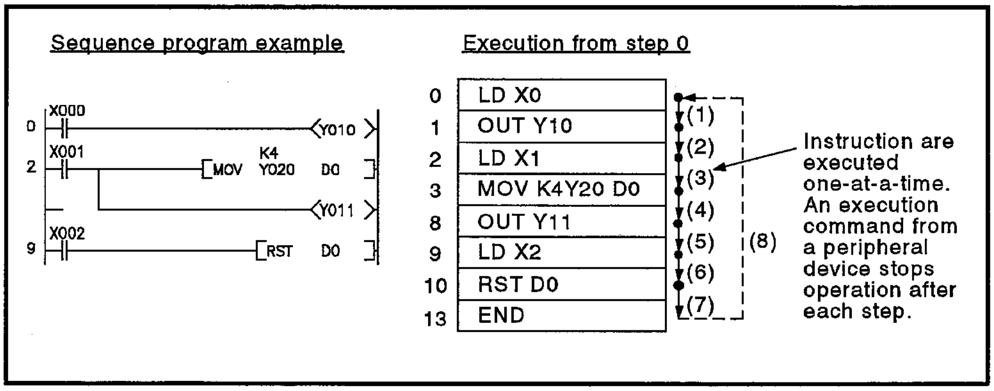
Sequence program steps can be executed one-at-a-time, beginning with a specified step.
Step operation can be used to confirm sequence program operation and contents of each device while the sequence program is being monitored for debugging or other purpose.

6.8.1 Step operation (I)

(1) Contents of step operation (I)

There are two methods used for step operation (I):

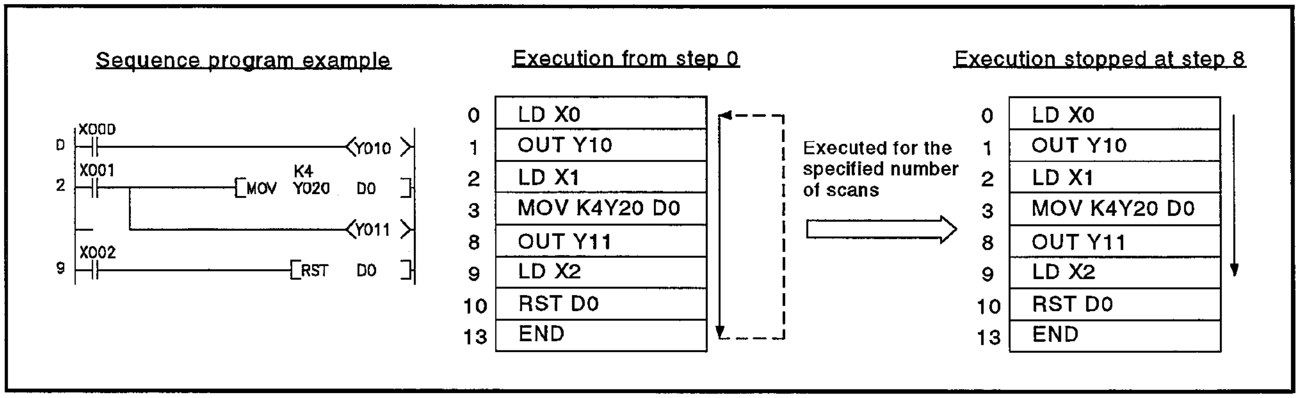
(a) Step operation of each instruction
Instructions are executed one-at-a-time from the current step.
Operation steps after each execution.
It can be used when each device state is confirmed at the execution of each instruction.


Sequence program example
0
X000
Y010
2
X001
MOV
K4
Y020
D0
Y011
9
X002
RST
D0
Execution from step 0
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
Instruction are executed one-at-a-time. An execution command from a peripheral device stops operation after each step.

0	LD X0
1	OUT Y10
2	LD X1
3	MOV K4Y20 D0
8	OUT Y11
9	LD X2
10	RST D0
13	END

Fig. 6.14 Step Operation by Each Instruction

(b) Step operation by specifying a loop count.
Step operation is executed for the specified loop count (1 through 1 to 32767) beginning with step 0 or the current step and stops at the specified step.



0	LD X0
1	OUT Y10
2	LD X1
3	MOV K4Y20 D0
8	OUT Y11
9	LD X2
10	RST D0
13	END

Fig. 6.15 Step Operation by Specifying the Loop Count

(2) Method of step operation

Use the following procedure to execute step operation:

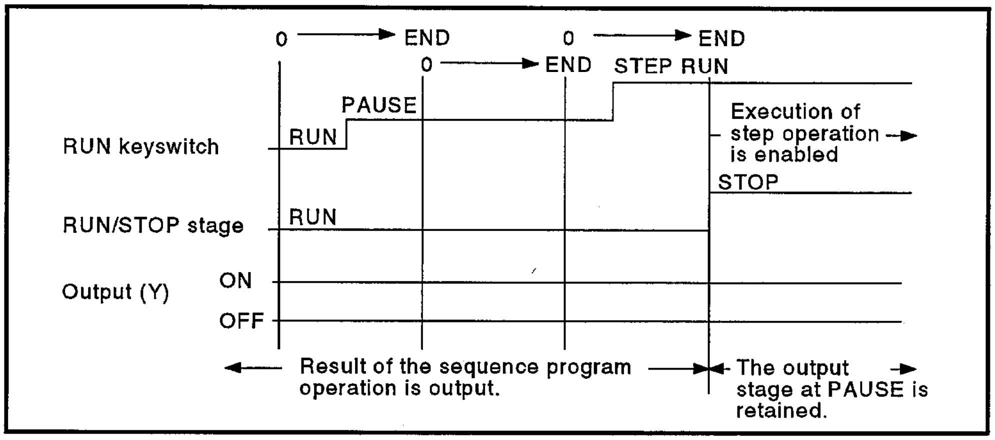
(a) Set the RUN keyswitch on the PC CPU to the "STEP RUN" position.

(b) Execute step operation with the GPP/PHP/HGP.
Refer to the Operating Manual of the peripheral device for the step operation.

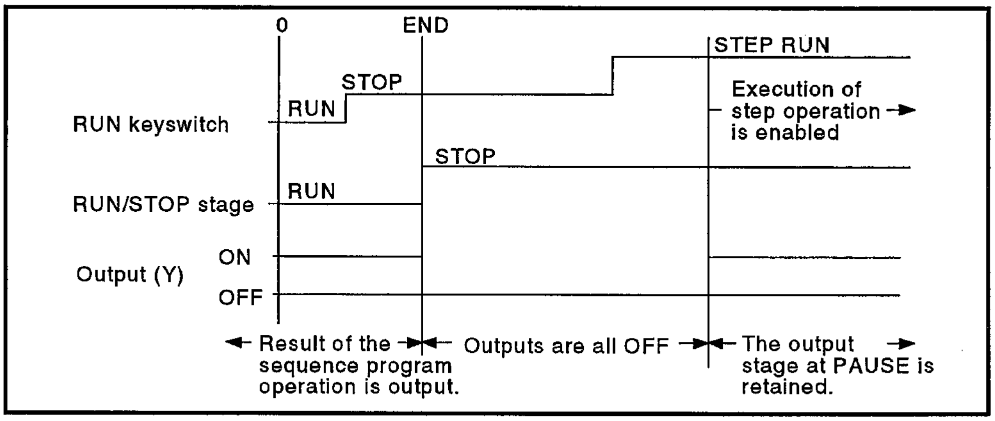
(3) Output (Y) state at step operation

The state of output (Y) differs with the method of setting the RUN key switch to the STEP RUN position.

(a) Operation : RUN → PAUSE → STEP RUN
When the RUN keyswitch is set to the STEP RUN position, the output ON/OFF state is retained, and the operation is stopped. The step operation executed with the GPP/PHP/HGP uses the state of ON/OFF of output (Y) at stop.


0 → END
0 → END
0 → END
STEP RUN
PAUSE
RUN
RUN keyswitch
Execution of step operation is enabled
STOP
RUN/STOP stage
RUN
Output (Y)
ON
OFF
Result of the sequence program operation is output.
The output stage at PAUSE is retained.

(b) Operation : RUN → STOP → STEP RUN
After storing all output in the internal memory, all outputs (Y) are turned OFF and the operation is stopped.


0
END
STEP RUN
STOP
RUN
RUN keyswitch
Execution of step operation is enabled
STOP
RUN/STOP stage
RUN
Output (Y)
ON
OFF
Result of the sequence program operation is output.
Outputs are all OFF
The output stage at PAUSE is retained.

(4) Processing of timers and special timing clocks at step operation

Processing of timer and special timing clocks (M9030 to M9034) in the sequence program is as follows.

(a) Timers

- 10 ms timer....................10 ms is added every 1 scan execution.
- 100 ms timer..................100 ms is added every 10 scan executions.

(b) Special timing clocks

- M9030 (0.1 s clock).......Turned ON/OFF every 5 scans.
- M9031 (0.2 s clock).......Turned ON/OFF every 10 scans.
- M9032 (1 s clock)..........Turned ON/OFF every 50 scans.
- M9033 (2 s clock)..........Turned ON/OFF every 100 scans.
- M9034 (1 min clock)......Turned ON/OFF every 3000 scans.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	x	x	x	x	x	o	o	o	x	x	x
Remark											

6.8.2 Step operation (II)

(1) Contents of step operation (II)

Step operation (II) can be executed with type AnA, A2AS, and AnU CPUs.
There are five methods used for step operation :

(a) Step operation of each instruction (Same as the step operation (I))
Instructions are executed one-at-a-time from the current step.
Operation stops after each execution.
It can be used to confirm each device state of each instruction execution.

(b) Step operation with step interval and loop count specification.
Instructions are executed one-at-a-time every specified step interval (1 to 99 s).
Operation is executed for the specified loop count (1 to 32767) and stops.

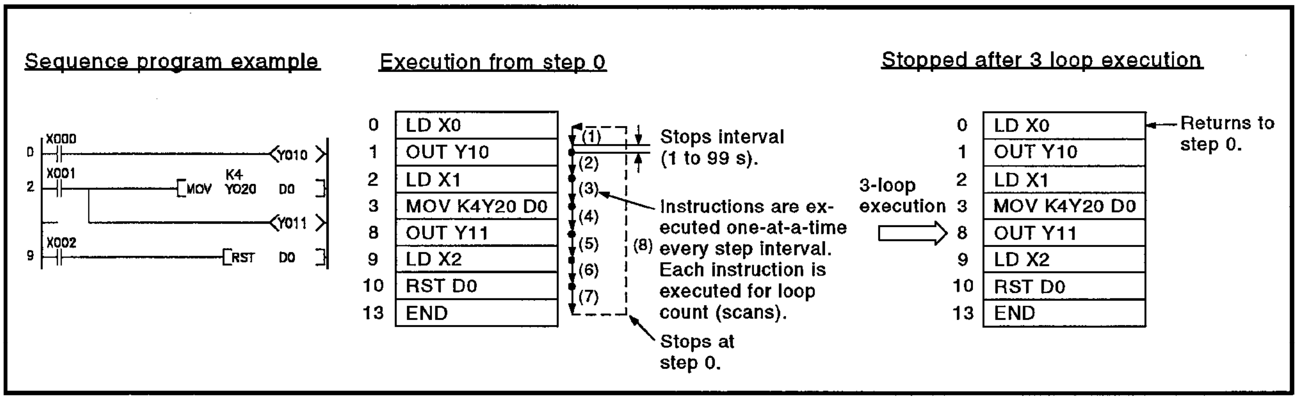
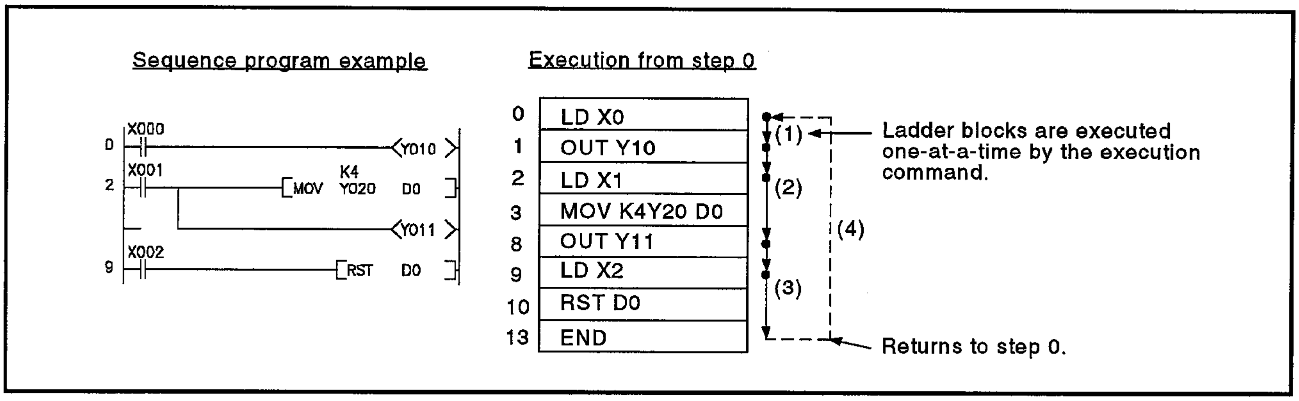


Fig. 6.16 Step Operation by Specifying Step Interval Loop Count

(c) Step operation by ladder block
Ladder blocks are executed one-at-a-time from the current step. Operation stops after each ladder.
It can be used to confirm each device state at each ladder block execution.

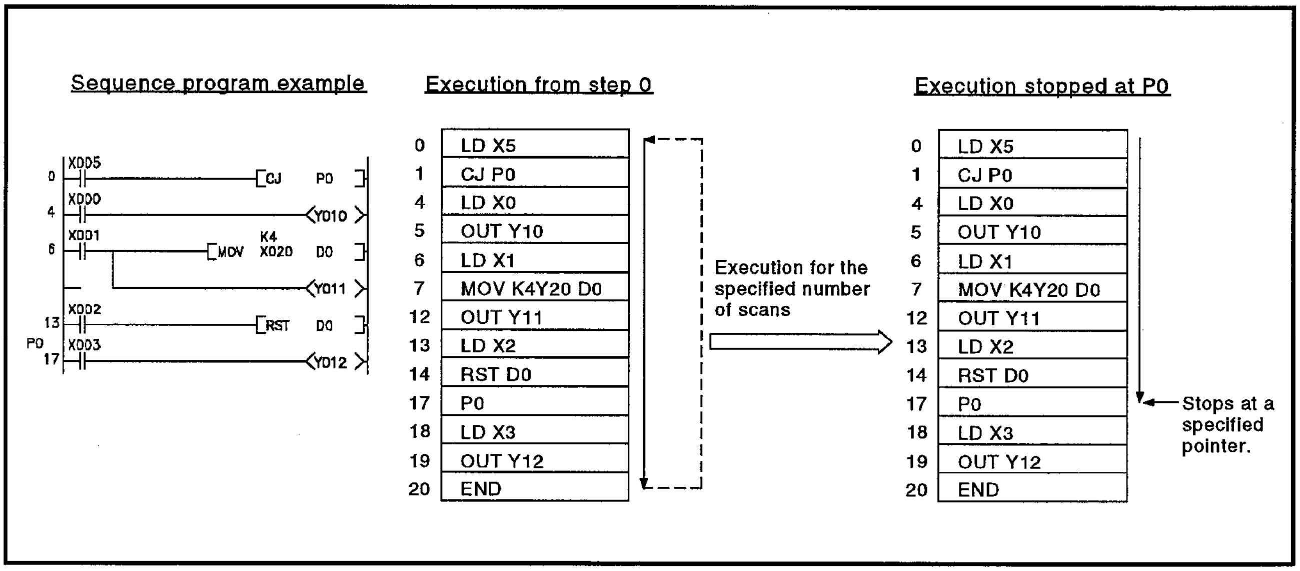


```
0   LD X0
1   OUT Y10
2   LD X1
3   MOV K4Y20 D0
8   OUT Y11
9   LD X2
10  RST D0
13  END
```

Fig. 6.17 Step Operation by Ladder Block

(d) Step operation by specifying loop count and break points
Step operation is executed for the specified loop count (1 to 9999) and stops at a specified break point.
There are two methods used for specifying a break point.

1) Step operation using a label (Pn) as a break point
This is the method of setting the pointer (Pn) that is set in the sequence program as the label of the break point.
A maximum of 4 places can be specified as the break point.



```
0   LD X5
1   CJ P0
4   LD X0
5   OUT Y10
6   LD X1
7   MOV K4Y20 D0
12  OUT Y11
13  LD X2
14  RST D0
17  P0
18  LD X3
19  OUT Y12
20  END
```

Fig. 6.18 Step Operation by Specifying Loop Count and Label

2) Step operation using a specified step as a break point.
This is the method of setting a step of a sequence program as a break point.
A maximum of 4 places can be specified as the break point.
A total of 4 places can be specified as break point when using both of above-mentioned 1) and 2).

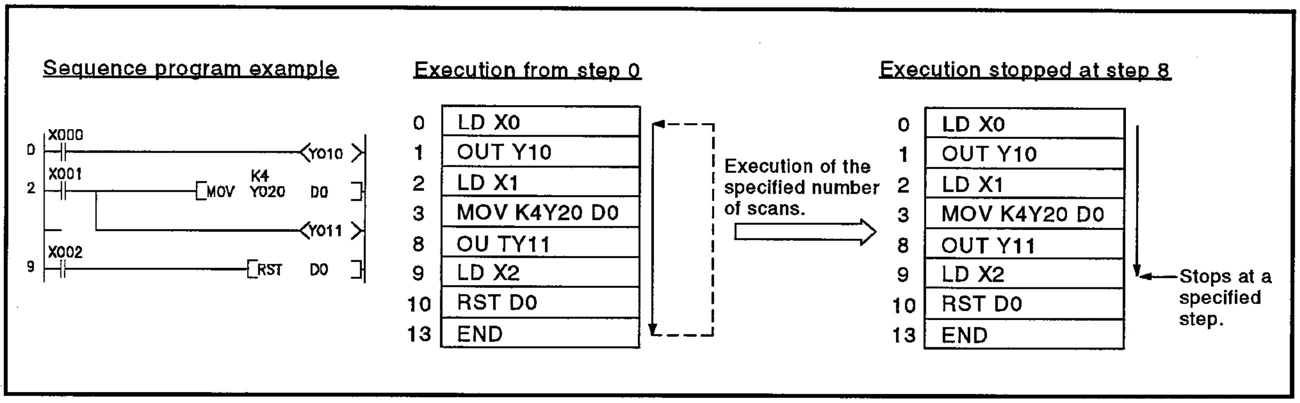
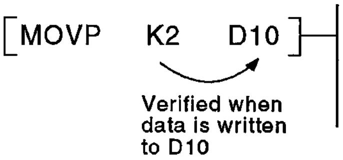


Fig. 6.19 Step Operation by Specifying Loop Count and Step Number

(e) Step operation according to device state
Operation stops when data of a specified bit device or a word device equals the set data.
Verify of data with the set data is executed when data is written to the specified device.

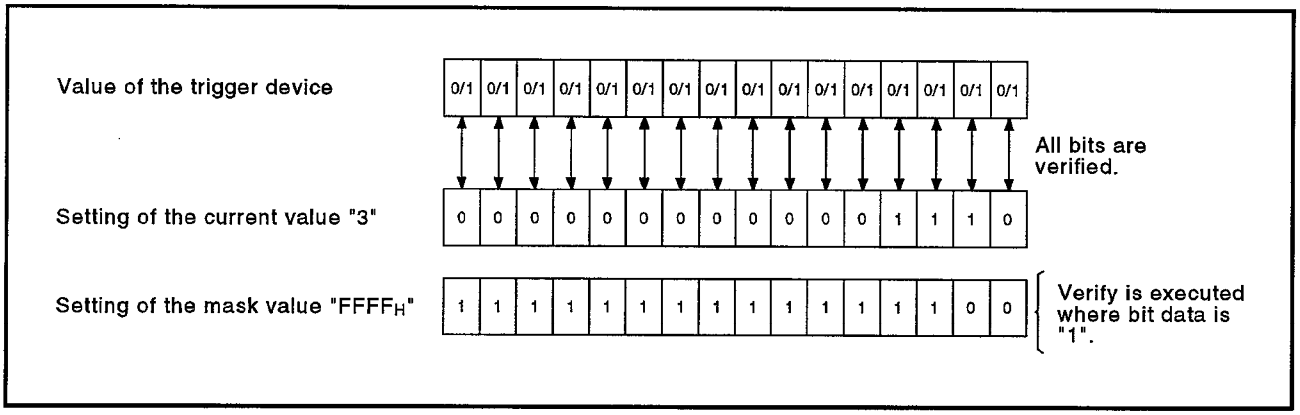


Specified device	: D10
Access	: Write
Current value	: 2
Mask value	: FFFF$_H$

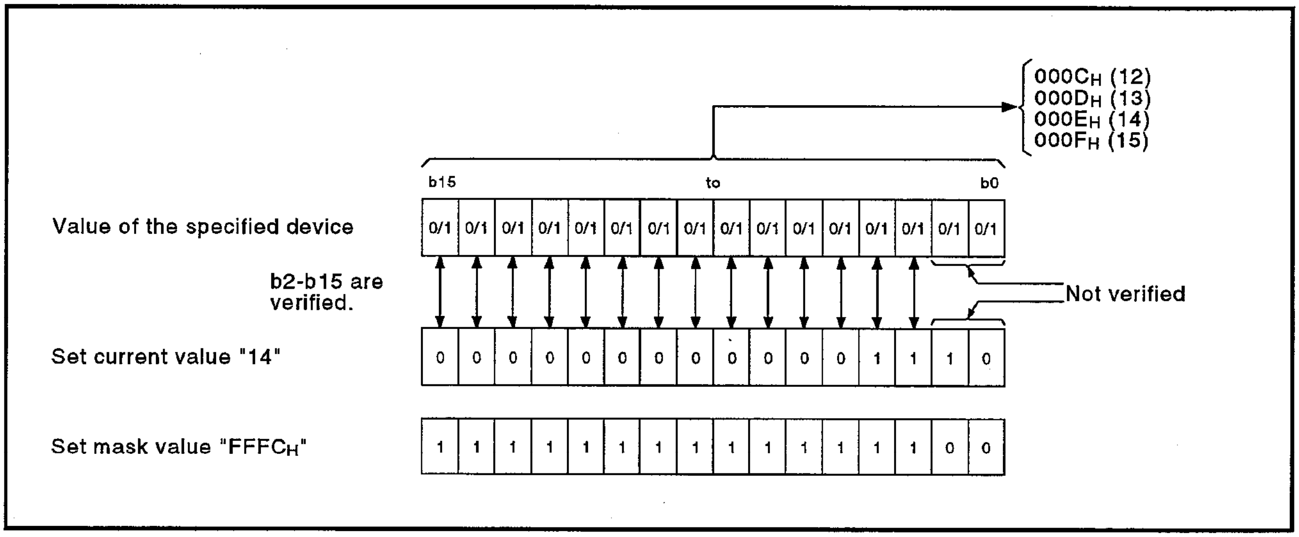
Number of the Trigger Devices	Device that can be Specified	Device State	Device Data
1 point	All bit devices	ON → OFF ON ← OFF	—
	All word devices	—	*1Current value & Mask value

*1 Current value & Mask value.
The following example describes the relation between the set values of the current value, the mask value and step operation.
The bit of which current value is verified with the trigger device is set with the mask value.
When the bit of the mask value is 1, the bit is verified, and when it is 0, the bit is not verified.

1) When the current value is "3" and the mask value is "FFFF"
When the mask value is "FFFF" , step operation is stopped when all bits of the current value and the trigger device become equal with each other.



2) When the current value is "14" and the mask value is "FFFC$_H$"
When the mask value is "FFFC" , step operation is stopped when the bit data except 2 lower bits of the current value and the trigger device value become equal with each other.
Lower 2 bits of the trigger device are ignored.
When the value of the trigger devices becomes "12", "13", "14" or "15", step operation is stopped.



POINT

When status latch is being executed by a specified device, step operation according to break specification of a device condition can not be executed.

(2) Method of step operation

Use the following procedure to execute step operation:

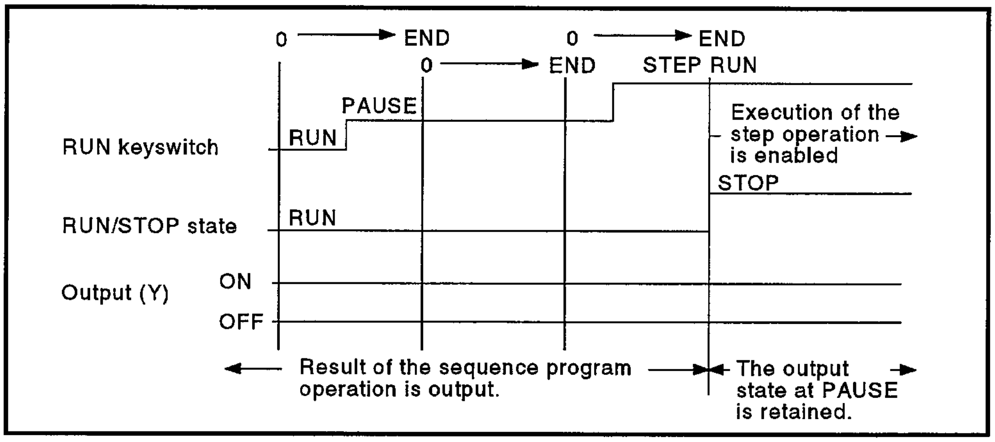
(a) Set the RUN keyswitch on the PC CPU to the STEP RUN position.

(b) Execute the step operation with the GPP/PHP/HGP.
Refer to the Operating Manual of a peripheral device for step operation.

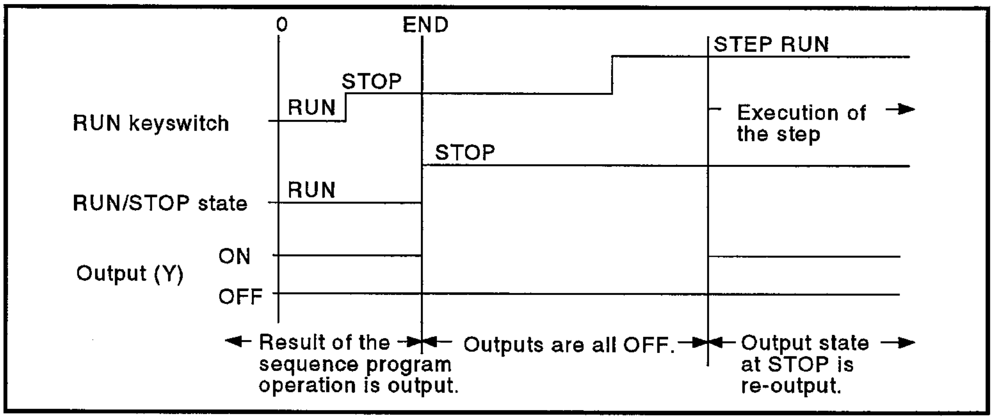
(3) Output (Y) state during step operation

The state of output (Y) differs with the method of setting the RUN keyswitch to the STEP-RUN position.

(a) Operation : RUN → PAUSE → STEP RUN
When the RUN keyswitch is set to the STEP RUN position, the ON/OFF state of all outputs (Y) are retained and the operation is stopped.
Step operation with the GPP/PHP/HGP is executed by using the ON/OFF state of output (Y) at the time of stop.


0 → END
0 → END
0 → END
STEP RUN
PAUSE
RUN
Execution of the step operation is enabled
STOP
RUN keyswitch
RUN/STOP state
RUN
Output (Y)
ON
OFF
Result of the sequence program operation is output.
The output state at PAUSE is retained.

(b) Operation :RUN → STOP → STEP RUN
After storing the state of all outputs (Y) in the internal memory, all outputs (Y) are turned OFF, operation is stopped.



(4) Processing of timers and special timing clocks when step operation is executed

Processing of timers in the sequence program and special timing clock (M9030 to M9034) is as follows.

(a) Timers

- 10 ms timer..................10 ms is added every 1 scan execution.
- 100 ms timer................100 ms is added every 10 scans execution.

(b) Special timing clock

- M9030 (0.1 s clock).......Turned ON/OFF every 5 scans.
- M9031 (0.2 s clock).......Turned ON/OFF every 10 scans.
- M9032 (1 s clock)..........Turned ON/OFF every 50 scans.
- M9033 (2 s clock)..........Turned ON/OFF every 100 scans.
- M9034 (1 min clock).... Turned ON/OFF every 3000 scans.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	o	x	o	o	o	o	o	x	x	o
Remark	*1 : Only the AnN is applicable.										

6.8.3 Precautions for step operation

The following describes the precautions for step operation.

(1) Loop count is counted after execution of the step specified as a stop step in the step operation with the loop count specification.
Therefore, when the step specified as a stop step has not been executed due to the CJ instruction, loop count is not counted.

(2) When the RUN keyswitch is moved from the STEP RUN position into the STOP position or from the RUN position into the STOP position, the state of outputs before STOP is retained in the internal memory of the CPU module.
Therefore, when the RUN keyswitch is moved from the STOP position into the STEP RUN position or from the STOP position into the RUN position, the outputs retained in the internal memory of the CPU module are re-output before executing the operation.
Move the keyswitch from the STOP position into the STEP RUN position or the RUN position after reset in order to disable output of the outputs in the internal memory of the CPU module at the STOP state.

(3) The next step of the NEXT instruction step can not be set to a stop step in the step operation by the loop count specification.
If the next step of the NEXT instruction is set to a stop step in the step operation by the loop count specification, an error occurs and the PC CPU stops.
At that time, the error message "CAN'T EXECUTE(P)" is displayed.
In the case of the operation by each instruction, this setting does not cause an error.

(4) When the operation processing is stopped with the step operation, I/O refresh is executed.
The PC CPU using the refresh mode receives or outputs signals during the operation.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	o	x	o	x	o	o	o	o	x	△*1	o
Remark	*1: Usable with the A2CCPUC24(-PRF) and A52GCPU(-T21B).										

6.9 Clock Function

Some PC CPUs have the clock function inside of the CPU module.
It can be used for time management, because clock data can be read with a sequence program.
Clock operation by the clock function is continued with a battery when the PC CPU is powered off or at momentary power failure more than 20 ms.

POINT

Clock operation can not be continued if the following is done when the PC CPU is powered OFF or at momentary power failure more than 20 ms.

- The connector of the battery is uncoupled.
- The memory cassette is removed from the CPU module.

(1) Clock data
Clock data is the data comprised of year, month, day, hour, minute, second, and day of the week used by the clock element installed inside the PC CPU.

Data Name	Description	
Year	The lower 2 digits of the Christian Era	
Month	1 to 12	
Day	1 to 31 (A leap year is distinguished automatically)	
Hour	0 to 23 (Controlled 24 hours)	
Minute	0 to 59	
Second	0 to 59	
Day of the week	0	Sunday
	1	Monday
	2	Tuesday
	3	Wednesday
	4	Thursday
	5	Friday
	6	Saturday

(2) Precision
Precision of the clock element depends on the operating ambient temperature as shown below.

Ambient Temperature (°C)	Precision (Week by Error in Seconds)	
	Except AnA and AnU	AnA and AnU
+55	+15.5	within ± 8
+25	+2.75	within ± 15
0	+6.5	within ± 7

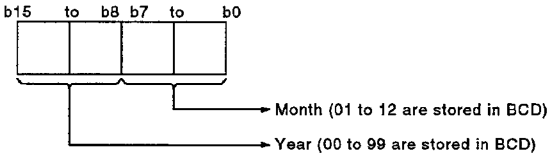
(3) Special relays and special registers for reading and writing clock data. The following describes the special relays and special registers used for setting data and reading clock data for clock operation.
The devices enclosed in square brackets [ ] are used for the A2CCPUC24(-PRF) and A52GCPU(-T21B).

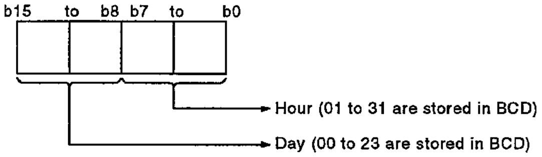
(a) Special relays used for the clock function

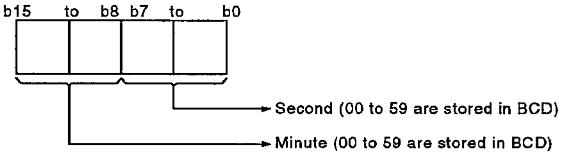
Device	Name	Description
M9025 [M9073]	Clock data set request	• Clock data is written to the special register (D9025 to D9028) for the clock operation. • After the END instruction is executed in a scan in which M9025 is turned ON the clock data stored in D9025 to D9028 is written to the clock element.
M9026 [M9074]	Clock data error	• Error is detected when a clock data is set. • Turned ON when each data is not BCD data.
M9027[*1]	Clock data display	• Clock data is displayed on the LED display on the front of the PC CPU. • When M9027 is turned ON, clock data is displayed on the LED display on the front of the CPU module.
M9028 [M9076]	Clock data read request	• Clock data is read to the special registers (D9025 to D9028). • When M9028 is turned ON, clock data is read to D9025 to D9028 after execution of the END instruction.

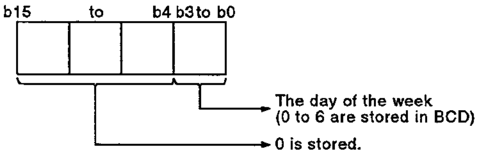
[*1] Usable with A3A A73 and A3N.

(b) Special registers used clock data

<table>
<tr><th>Device</th><th>Name</th><th>Description</th></tr>
<tr><td>D9025
[D9073]</td><td>Clock data (year and month)</td><td>• Year and Month are as follows.

b15 to b8 b7 to b0
Month (01 to 12 are stored in BCD)
Year (00 to 99 are stored in BCD)

</td></tr>
<tr><td>D9026
[D9074]</td><td>Clock data (day and hour)</td><td>• Day and Hour are as follows.

b15 to b8 b7 to b0
Hour (01 to 31 are stored in BCD)
Day (00 to 23 are stored in BCD)

</td></tr>
<tr><td>D9027
[D9075]</td><td>Clock data (minute and second)</td><td>• Minute and Second are as follows.

b15 to b8 b7 to b0
Second (00 to 59 are stored in BCD)
Minute (00 to 59 are stored in BCD)

</td></tr>
<tr><td>D9028
[D9076]</td><td>Clock data (day of the week)</td><td>• The day of the week is as follows.


• Setting of the day of the week is as follows.
<table>
<tr><td>Day of the week</td><td>Sun.</td><td>Mon.</td><td>Tues.</td><td>Wed.</td><td>Thur.</td><td>Fri.</td><td>Sat.</td></tr>
<tr><td>Data stored</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr>
</table></td></tr>
</table>

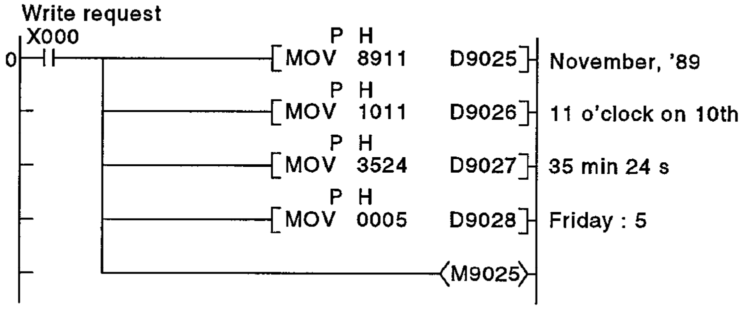
(4) Writing clock data to the clock element

(a) Use the following procedure to write clock data to the clock element.

1) Store clock data to D9025 to D9028 in the BCD code by use of the sequence program or a peripheral device.

2) Turn M9025 ON by use of the sequence program or a peripheral device, and write the data of D9025 to D9028 to the clock element.

(b) Writing example of the clock data by use of the sequence program.



(c) When writing clock data with a peripheral device, use the test mode. Refer to the Operation Manual of the peripheral device for turning ON/OFF the special relays and writing data to the special registers with the test mode.

POINTS

(1) The CPU is shipped without clock data being correctly set. When using the clock function, write clock data to the clock element.

(2) When correcting some parts of clock data, it is necessary to write all data again to the clock element.

(3) If illegal clock data is written to the clock element, the clock operation can not be executed normally.

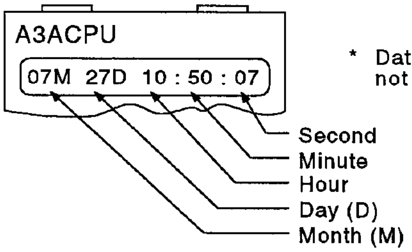
Example:

13 (month) 32 (day)

(5) Reading clock data

(a) When reading clock data to D9025 to D9028, turn M9028 ON by use of the sequence program or a peripheral device.

(b) The PC CPU which has a 16 digit LED indicator on the front can display clock data (month, day, hour, minute, second). To display clock data on the LED indicator, turn M9027 ON. Clock data has the lower priority of order for display when an error occurs it is displayed, and clock data is not displayed.



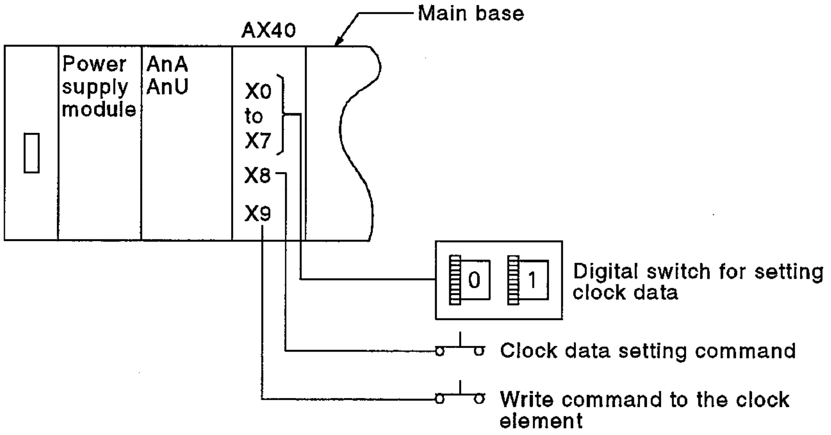
* Data of year and day of the week are not displayed.

(6) When an AnA, A2AS, AnU and QCPU-A is used, data of a specified word device can be written to, and read from the clock element with a dedicated instruction.
When clock data is set with peripheral digital switches, use a dedicated instruction.

Example

A program for setting clock data using peripheral digital switches is as shown below.

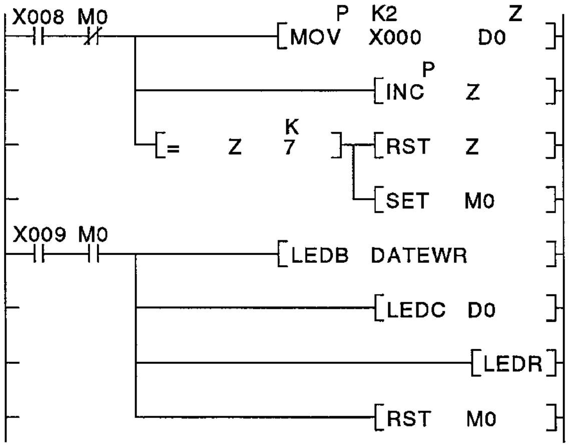
System configuration
The system configuration for setting clock data.


AX40
Main base
Power supply module
AnA
AnU
X0
to
X7
X8
X9
0
1
Digital switch for setting clock data
Clock data setting command
Write command to the clock element

Clock data storage device
Devices used to store clock data are as follows.

D0	Year
D1	Month
D3	Day
D4	Hour
D5	Second
D6	Day of the week

Program example
The following is a program to write clock data to the clock element.



Clock data is written to D0 to D6.

When setting of clock data is completed, M0 (clock data setting completion flag) is set, and Z is reset.

Writing clock data

(7) Handling the year 2000

Year 2000 is a leap year, so February 29 comes after February 28.
The AnSHCPU automatically fixes data by the clock element in the CPU module, so the user does not have to manually set the date to the clock element.
The year only contains the last two digits of the year. Therefore, when the clock data is read from the PC CPU and used in the sequence control, the year data may have to be fixed using a sequence program depending on the usage.

Year 1999 → "99"

Year 2000 → "00"

When comparison is made only with the last two digits of the year read, the year 2000 and consecutive years are considered older than the year 1999.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	x	x	x	x	o	△*2	x	x	x	o
Remark	*1 : Only the AnN is applicable. *2 : Only the AnU is applicable.										

6.10 I/O Module Replacement During Online

When I/O modules or special-function modules are replaced when the PC CPU is in the ON state, the "UNIT VERIFY ERROR" occurs.
And the operation of the PC CPU stops or the I/O numbers become different from set numbers.
The I/O module replacement during online is to replace I/O modules without causing the "UNIT VERIFY ERROR" when the the PC CPU is in the ON state.

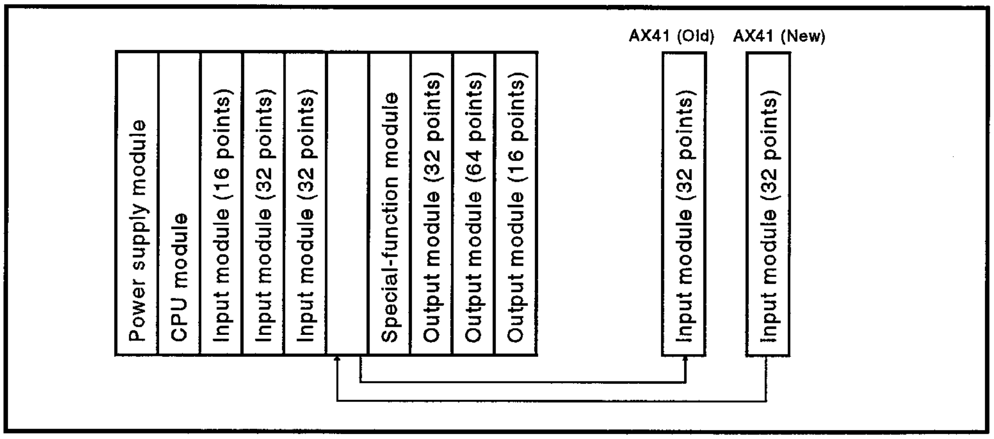


Fig. 6.20 I/O Module Replacement During Online

(1) Purpose
The I/O module replacement during online can be used while continuing the control with the sequence program and allow I/O modules to be replaced without causing "UNIT VERIFY ERROR"

(2) Operation procedure

(a) Use the following procedure to replace an I/O module during online:

1) Set the upper 2 digits of the head I/O number displayed in 3 digits of the I/O module to be replaced into D9094 (Replacement I/O head I/O number storage register).
Example : Head I/O number 070 → Set number: H07
Head I/O number 170 → Set number: H17

2) Set M9094 (I/O replacement flag).

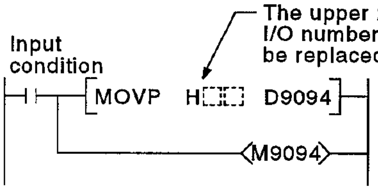
3) Replace the specified I/O module.

4) Reset M9094 (I/O replacement flag)

(b) Data can be set for D9094 and M9094 can be turned ON/OFF by sequence program or a peripheral device.

(a) Using the sequence program

The specified I/O module can be replaced with the following sequence program when the PC CPU is in the RUN state.



The upper 2 digits of the head I/O number of the I/O module to be replaced are specified.

* This sequence program can be written with a peripheral device to the PC CPU in the RUN state.

PROCEDURE

(1) Turn ON the input condition, and set the replacing module number to D9094. M9094 is turned ON.
(2) Replace the specified I/O module.
(3) Turn the input condition OFF. M9094 is turned OFF. (Operation of the replaced module starts again.)

(b) Using a peripheral device

The I/O module specified with the test mode of a peripheral device can be replaced. The PC CPU may be in either the RUN state, the STOP state or the PAUSE state.

PROCEDURE

(1) Connect the peripheral device to the PC CPU.
(2) Set the upper 2 digits of the head I/O number of the I/O module to be replaced to D9094 with the test mode.
(3) Set (turn ON) M9094 with the test mode.
(4) Replace the specified I/O module.
(5) Reset (turn OFF) M9094 with the test mode. (Operation of the replaced module starts.)

(3) Each module executes processings as follows according to the ON/OFF state of special relay M9094.

M9094	Module	Processing Contents in the CPU "RUN" State
ON	CPU	1) The operation state is stored to the image memory area of X or Y in the CPU when the first END is processed after M9094 is turned ON. After that, the corresponding input or output module is not accessed.
		2) After M9094 is turned ON, the operation processing of the corresponding input module is executed (ON or OFF) of the image memory area of X.
		3) After M9094 is turned ON, the result of the execution processing of the CPU is stored in the image memory area of Y by the operation of the corresponding output module.
		4) After M9094 is turned ON, verify of module and fuse blown check are not executed for all modules.
	Replaced module	1) The operation state at the first END instruction processing after turning ON M9094 is retained and the operation is stopped.
	Other modules	1) Normal operation. The module which executes output according to data of the input module to be replaced operates in the condition that the input data is fixed.
OFF	CPU	1) The input module executes operation processing with data of the image memory area of X in the first END processing after M9094 is turned OFF. The corresponding input module is accessed.
		2) An output module outputs data (ON or OFF) of the image memory area of Y in the first END processing after M9094 is turned OFF. The corresponding output module is accessed.
		3) After turning M9094 OFF, verify of module and fuse blown check are restarted for all modules.
	Replaced module	1) Access is restarted with data of the image memory area of the CPU after the END processing after M9094 is turned OFF.
	Other modules	1) Normal operation.

* Replacement with a peripheral device (force M9094 ON/OFF) while the CPU is in the STOP state is the same as mentioned above.
Since all outputs are turned OFF, all processing stops.
Output resumes in the RUN state.

(4) Precautions

NO.	Item	Precautions
1.	Replacement of special function module	They cannot be replaced online. Turn OFF the power and replace.
2.	Replaceable I/O module	1) Only the I/O modules which have the same number of I/O points can be substituted. (When the number of I/O points is different, errors may occur.)
		2) Only the modules specified with special register D9094 can be replaced. If module that is not specified is replaced, I/O misoperation of the devices connected with the module may occur because the module being replaced is still operating.
3.	Keyswitch operation to the RUN and the STOP position	Set the keyswitch before replacement. Do not operate it until replacement is completed. If it is operated while replacement, the operation processing of the CPU is changed and I/O misoperation is caused.
4.	Operation of D9094	1) When the power to the CPU is turned ON, HFF is set, and when the power is turned OFF, the data memory is cleared. Data is overwritten by specifying the upper 2 digits of the head I/O number.
		2) As the setting range and the set value of D9094 is not checked in the CPU, set the I/O number correctly and replace.
5.	Prohibition of the test operation with a peripheral device	After turning special relay M9094 ON, do not execute the test operation with a peripheral device. If the test operation is executed, the content of the relay enters the image memory area, and when M9094 is turned OFF, it is output.
6.	Safe measures for replacement when the CPU is in the RUN state	1) When the special relay M9094 is turned ON, the replacement module is stopped and when the relay is turned OFF, it is operated with data of the image memory area. Check the operation of the device after the replacement of the module. Turn M9094 OFF, and restart the operation.
		2) When a peripheral device which receives output is a display module such as the A6FD, safe measures are not needed. When a peripheral device is operated, provide interlock by use of the sequence program or turn off the device.
7.	In case a power failure more than 20 ms occurs (CPU is started initially) while the module is removed	1) Turn OFF the power supply, and load the replacement module. The "UNIT VERIFY ERR" error message appears, if a module is loaded in the ON state.
		2) If I/O numbers are allocated, the I/O numbers are not changed. This can prevent wrong input and output.
		3) If I/O numbers are not allocated, when the CPU module is started initially, the vacant slot allocated automatically as vacant 16 points. (Refer to Section 6.4.1.) When the removed I/O module has more than 16 points, the I/O numbers will be changed, and wrong input and output may occur. Therefore load the module in the OFF state.

Applicable CPU	All Types of CPUs
Remark	

6.11 Device Comments

The Name and application of a device can be assigned when programming. The device name and application are called comments.

(1) Purpose of a comment

If a comment is registered, it can be used as follows.

(a) A comment can be attached to a device and displayed when the program is monitored.
The devices used in the program are displayed with name and application, so that it is convenient when corrections are to be made to the program.

(b) A comment can be attached to the sequence program and can be printed.
Sequence program, device names and applications are printed at the same time, so that a program can be understood easily.

(c) The A6FD can display the comment data of the PC CPU by use of the sequence program.
The A6FD can display error contents by use of the fault detection ladder with annunciators.

(2) Number of comments and storage area

Number of comments that can be stored in the PC CPU and the storage area are as shown in Table 6.2.

Table 6.2 Number of Comment and Storage Area

	A2S(S1), A1SJ-S3 A1S(S1), A2C A0J2H, A52G	A1 A1N	A1SJH, A1SH A2SH(S1), A1FX	A2(S1), A3, A3H, A2A(S1), A3A, A73, A4U A2N(S1), A3N, A3M, A2U(S1), A3U, A2AS(S1/S30), A2USH-S1, Q02, Q02H, Q06H
Number of comments	0 to 1600*	0 to 128* (F0 to F127)	0 to 3648 (In unit of 84)	0 to 4032 (In unit of 64)
Storage area	Memory area in the CPU module			User memory area in the memory cassette

*0 to 4032 comments can be created with a peripheral device.

(3) Devices to which a comment can be added

(a) Bit devices
Input (X)/output (Y)
Internal relay (M)
Latch relay (L)
Step relay (S)
Link relay (B)
Annunciator (F)
Timer (T)
Counter (C)

(b) Word devices
Data register (D)
Link register (W)
File register (R)

(c) Others
Pointer (P)
Interrupt pointer (I)

POINTS

(1) Comments of the special relay and the special register are stored on a system floppy disk.
User can not create comment for the special relay and the special register.

(2) The comment of input (X) and output (Y) can not be created with the same I/O number.

(3) All devices' (except the input and output devices) comments can be created with the same device number independently in the main program and the sub-program.

(4) Creating a comment

(a) A maximum of 15 characters can be used for the comment of 1 device.

(b) Alphanumeric characters (upper and a lower cases) and special symbols can be used to create a comment.

Applicable CPU	**All Types of CPUs**
Remark	

6.12 Watchdog Timer

(1) Watchdog timer

The watchdog timer is an internal timer of a programmable controller to detect the error of hardware and a sequence program.
Default value is set for 200 ms.

(2) Reset watchdog timer

Before step 0 is executed (after the END processing is executed), a programmable controller resets the watchdog timer.
When a programmable controller is operating normally and the END instruction is executed within the set value in a sequence program, the watchdog timer does not time out.
When the END instruction has not been executed within the set value due to an error in the programmable controller or the scan time of the sequence program was too long, the watchdog timer times out.

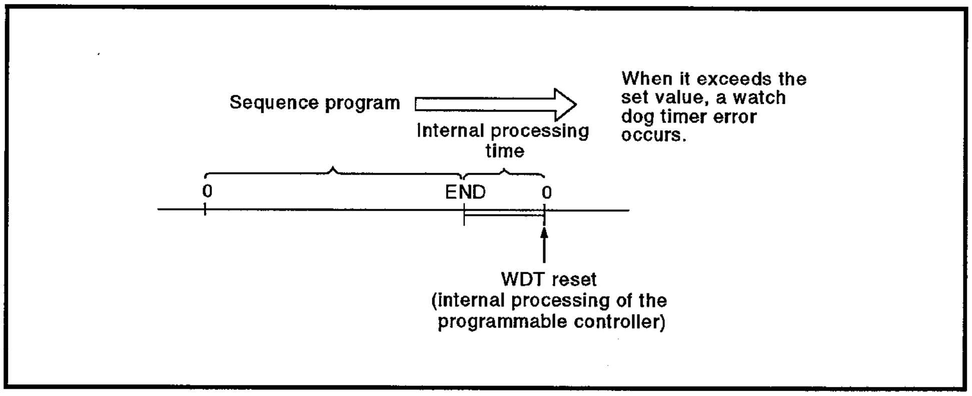

Sequence program
Internal processing time
When it exceeds the set value, a watch dog timer error occurs.
0
END
0
WDT reset (internal processing of the programmable controller)

Fig. 6.21 Reset of the Watchdog Timer

(3) Processing when the watchdog timer has timed out

When scan time exceeds the set value of the watchdog timer, a watchdog timer error occurs, and the programmable controller operates as follows.

(a) Outputs of the programmable controller are all turned OFF.

(b) The RUN LED on the front of the CPU module goes out or flashes.

(c) M9008 is turned ON, and an error code is stored to D9008.

REMARKS

1) The setting time of the watchdog timer can be changed by the parameter setting with the peripheral device.
   But when A3H, A3M, AnA, A2AS, AnU and QCPU-A are used, the setting time of the watchdog timer can not be changed.
   Refer to Chapter 8 for details on the parameter setting.
2) The watchdog timer can be reset with the WDT instruction in the sequence program.
   Scan time is not reset and is measured till the END instruction.

Applicable CPU	All Types of CPUs
Remark	

6.13 Self-diagnosis Function

The self-diagnosis is the function that diagnoses an error in the PC CPU itself.

(1) Self-diagnosis timing

The self-diagnosis is executed when the PC CPU is powered on or reset and when each instruction including the END instruction is executed.

(a) At power on or reset
Executability of operation by the PC CPU is diagnosed.

(b) At the execution of each instruction*1
When the operation of each instruction of a sequence program cannot be executed normally, an error occurs.

(c) At the execution of the END instruction
The diagnosis (watchdog, I/O module verify and fuse blown) that does not influence the operation of the sequence program is executed.

(2) Operation mode at error detection

There are two methods of operation mode, one of which stops the operation of the PC CPU and the other continues operation when an error is detected by the self-diagnosis.
There is an option, by parameter setting, to stop the operation when an error occurs in the mode that continues the operation.
(Refer to Section 6.13.1.)

(a) When an error which stops operation is detected by the self-diagnosis, operation is stopped and all outputs (Y) are turned OFF.

(b) When an error which continues operation is detected, only the part of the program where the error occurred is not executed, the next step and remaining program is executed.
When an I/O module verify error is detected, operation is continued with the I/O addresses before the detection of the error.

(3) Confirmation of error contents

When an error is detected, M9008 (self-diagnosis error) is turned ON and an error code is stored to D9008 (self-diagnosis error).
Confirm error contents, especially in the continue mode, to prevent the PC CPU or a machine system from misoperating.
Error contents detected by the self-diagnosis are as shown in Table 6.3.
Refer to the User's Manual of the PC CPU on the self-diagnosis contents that can be used in each PC CPU.

REMARKS

(1) Two different states mentioned in the "state of the CPU" and the "state of RUN LED" columns in Table 6.3 can be switched by the setting with a peripheral device.

(2) The message on the LED display is as shown below when the error contents of the error is related with the CHK instruction in the Operation Check Error.

[<CHK> ERROR□□□]

Displays an error code in 3 digits.

(3) *1 : When an error is detected during a sequence program execution, an operation error flag (M9010, M9011) is set, and the error step of the instruction with which the error is detected is stored in an error step storage register (D9010, D9011).
When an operation error (D9008 : 50) occurs with an AnA, AnU or QCPU-A the error step number can be checked by monitoring D9010 and D9011.

Table 6.3 Self-diagnosis is Contents and Error Messages

Diagnosis Contents		Diagnosis Timing	State of the CPU	State of the RUN LED	LED Message (A3N, A3A, A3U, A4U)
Memory error	Instruction code check	At the execution of each instruction	Stop	Flicker	INSTRCT. CODE ERR
	Parameter setting check	At power on or reset When {STOP/PAUSE} is change into {RUN/STEP-RUN}			PARAMETER ERROR
	Without the END instruction	When M9056 or M9057 is in the ON state When switching from {STOP/PAUSE} to {RUN/STEP-RUN}			MISSING END INS.
	Instruction execution is disabled	At the execution of each instruction [CJ] [SCJ] [JMP] [CALL(P)] [FOR-NEXT] When switching from {STOP/PAUSE} to {RUN/STEP-RUN}			CAN, T EXECUTE(P)
	Format (CHK instruction) check	When switching from {STOP/PAUSE} to {RUN/STOP-RUN}			CHK FORMAT ERR.
	Instruction execution is disabled	At interruption occurrence When switching {STOP/PAUSE} to {RUN/STEP-RUN}			CAN, T EXECUTE (I)
	Without a memory cassette	At power on or reset			CASSETTE ERROR
CPU error	RAM check	At power on or reset When M9084 is in the ON state during STOP	Stop	Flicker	RAM ERROR
	Operation ladder check	At power on reset			OPE. CIRCUIT ERR.
	Watchdog error monitor	At the execution of the END instruction			WDT ERROR
	The END instruction is not executed	At the execution of the END instruction			END NOT EXECUTE
	Main CPU check	Usually			WDT ERROR
I/O error	I/O module verify	At the execution of the END instruction (It is not checked when M9084 or M9094 is in the ON state.)	Stop Operation	Flicker ON	UNIT VERIFY ERR.
	Fuse blown	At the execution of the END instruction (It is not checked when M9084 or M9094 is in the ON state.)			FUSE BREAK OFF.
Special-function module error	Control bus check	At the execution of the FROM and the TO instructions	Stop	Flicker	CONTROL-BUS ERR.
	Special-function module error	At the execution of the FROM and the TO instructions			SP. UNIT DOWN
	Link module error	At power on or reset When switching {STOP/PAUSE} to {RUN/STEP-RUN}			LINK UNIT ERROR
	I/O interrupt error	At interruption occurrence			I/O INT. ERROR
	Special-function module allocation error	At power on or reset When switching {STOP/PAUSE} to {RUN/STEP-RUN}			SP. UNIT LAY. ERR.
	Special-function module error	At the execution of the FROM and the TO instructions	Stop Operation	Flicker ON	SP. UNIT ERROR
	Link parameter error	At power on or reset When switching {STOP/PAUSE} to {RUN/STEP-RUN}	Operation	ON	LINK PARA. ERROR
Battery	Battery voltage drop	Regularly (It is not checked when M9084 is in the ON state.)	Operation	ON	BATTERY ERROR
* Operation check error		At the execution of each instruction	Stop Operation	Flicker ON	OPERATION ERROR

Applicable CPU	All Types of CPUs
Remark	

6.13.1 Operation mode when an error occurs

The PC CPU can be set to stop or continue operation when the following errors occur.
Stopping or continuing the operation is set with parameters.

(a) Operation error

(b) I/O module verify error

(c) Fuse blown

(d) Special-function module error

(1) Default value of the operation mode when an error occurs

The default value (initial value) of the operation mode and the state of the PC CPU when an error occurs are as shown in Table 6.4.

Table 6.4 Operation Mode at Time of Error

Error		CPU Status					
		Operation	RUN LED		Special Relay Switched On	Special Register for Storing Data	Self-check Error Number (D9008)
		Default Value	NO "ERROR" LED CPU	"ERROR" on the LED of other CPU			
Operation error	Sequence program error, e.g. the value to be converted into BCD is greater than 0 to 9999 (or 0 to 99999999).	Continue	Flicker	On	M9010 M9011	D9010 D9011	50
I/O module verify error	Any I/O module status detected is different from that at power on (e.g. 32-point module change).	Stop	Flicker	A3: Off A3N, A3H: Flicker	M9002	D9002	31
Fuse blow error	An output module fuse has blown.	Continue	Flicker	On	M9000	D9000	32
Function module error	FROM/TO Instruction has been executed to the slot without any special function module.	Stop	Flicker	A3: Off A3N, A3H: Flicker	M9010 M9011	D9010 D9011	46

Applicable CPU	**All Types of CPUs**
Remark	

6.14 Setting of the Output (Y) State when Switching from STOP to RUN

When the operation state is switched from RUN to STOP, outputs (Y) in the RUN state are stored to the PC CPU.
Whether re-outputting outputs (Y) when the operation state is switched from STOP to RUN or outputting them after executing the operation can be set with the parameter.

(1) Re-output .... The operation of a sequence program is executed after outputting the outputs (Y) state which were stored before the operation state was switched to STOP.

(2) Output after executing... the operation After clearing all outputs (Y) and executing the operation of a sequence program, outputs (Y) are output.

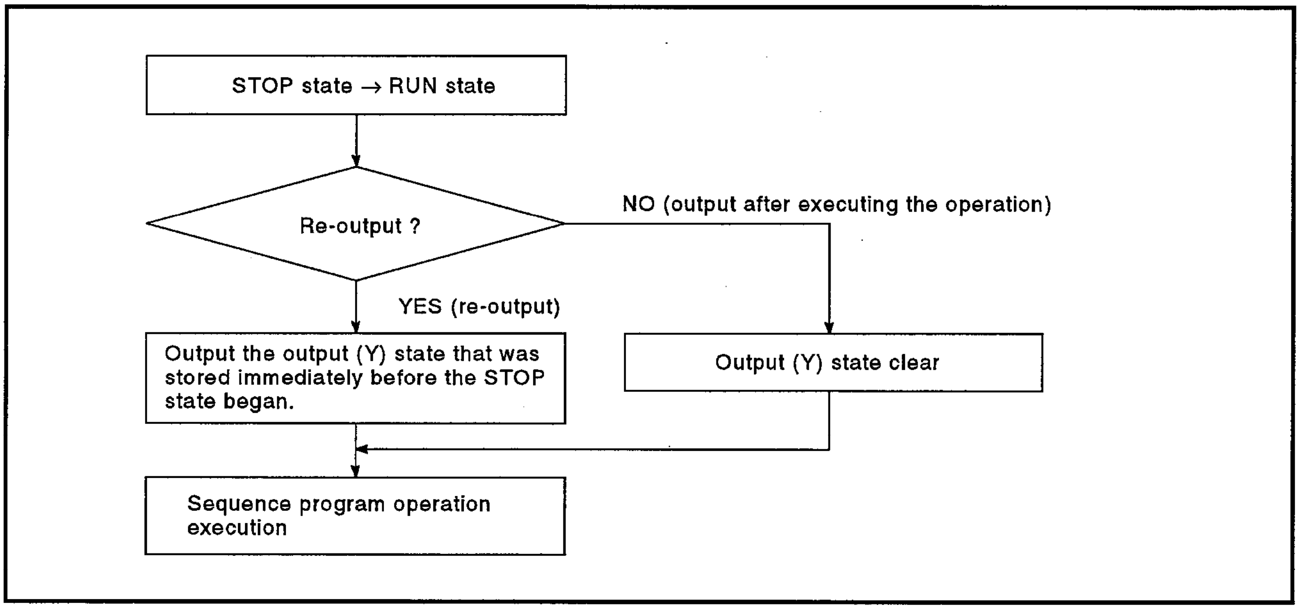

STOP state → RUN state
Re-output ?
NO (output after executing the operation)
YES (re-output)
Output the output (Y) state that was stored immediately before the STOP state began.
Output (Y) state clear
Sequence program operation execution

Fig. 6.22 Processing when the Operation STOP State is Switched from RUN

Applicable CPU	All Types of CPUs
Remark	

6.15 Registration of the Entry Code

The entry code is used to prohibit programs and comments in the PC CPU from being read or rewritten with a peripheral device.

(1) Read/write from the PC CPU to which the entry code is registered.
In case the key word is registered, parameters, main/subprograms and comments can not be read or written from the PC CPU to a peripheral device unless the entry code is entered to the peripheral device.

(2) Registration and cancellation of the entry code
A maximum of 6 digits in hexadecimal (0 to 9, A to F) can be used to set the entry code.
The entry code is registered or canceled with parameter setting.

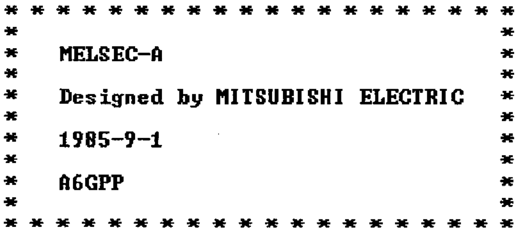
Applicable CPU	All Types of CPUs
Remark	

6.16 Registration of the Print Title

The print title is the comment of the system name and the program name used with the sequence program.

(1) Purpose

If the print title is registered, it can be printed.
The printed print title can be used for the cover of a sequence program.



```
* * * * * * * * * * * * * * * * * * * * *
*                                       *
*   MELSEC-A                            *
*                                       *
*   Designed by MITSUBISHI ELECTRIC     *
*                                       *
*   1985-9-1                            *
*                                       *
*   A6GPP                               *
*                                       *
* * * * * * * * * * * * * * * * * * * * *
```

(2) Setting the print title

(a) A maximum of 128 characters (32 characters x 4 lines) can be used to make the print title.

(b) The print title is set by parameters with a peripheral device.

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	△*1	x	o	o	△*1	△*1	x	x	x	o
Remark	*1 : Only the A3, A3N, A3A, A3U, and A4U are applicable.										

6.17 Display Mode Setting of Annunciators (In case of the CPU module with the LED indicator of 16 characters)

As for the PC CPU that has the LED indicator of 16 characters on the front of the CPU module, the LED indicator displays the annunciator number stored in D9009 when the annunciator (F) is turned ON.
At this time, in case a comment has been added to the annunciator, the following setting is enabled with a parameter.
(In case a comment has not been added to the annunciator, the setting with comment is disabled.)

(1) Without comment : Only the annunciator number is displayed on the LED indicator.

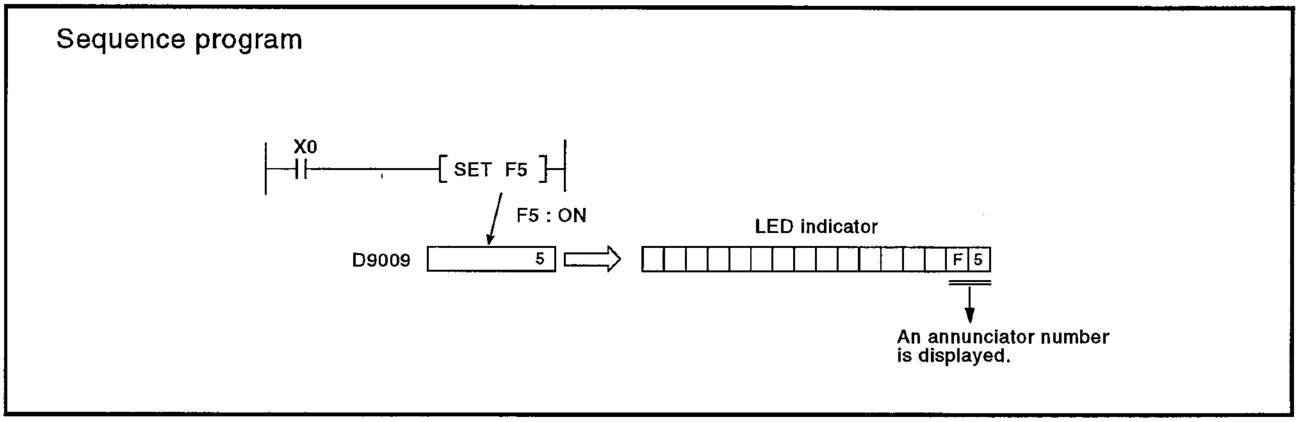


Fig. 6.23 Without Comment

(2) With comment : An annunciator number and a comment are displayed alternately on the LED indicator every 2 s.

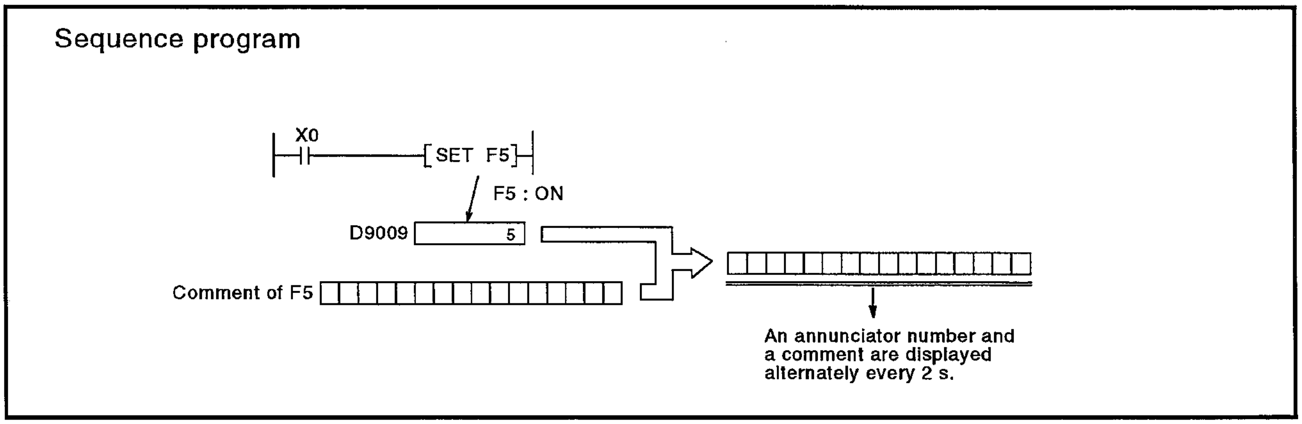

Sequence program
X0
SET F5
F5 : ON
D9009
5
Comment of F5
An annunciator number and a comment are displayed alternately every 2 s.

Fig. 6.24 With Comment

Applicable CPU	AnS AnN AnSH	An	A1FX	A3H A3M	A3V	AnA	AnU A2AS	QCPU-A	A0J2H	A2C A52G	A73
	△*1	x	o	x	x	o	o	o	o	o	o
Remark	*1 : AnN is unusable.										

6.18 ERROR LED Indication Priority Setting

The following functions are available by changing the ERROR LED indication priority setting:

(a) Unnecessary error indications among those given in Table 6.5 can be avoided.
For example, it is possible to set the ERR. LED not to be lit even when an annunciator is turned ON.
This feature is not available with the errors that stop the sequence program operation.

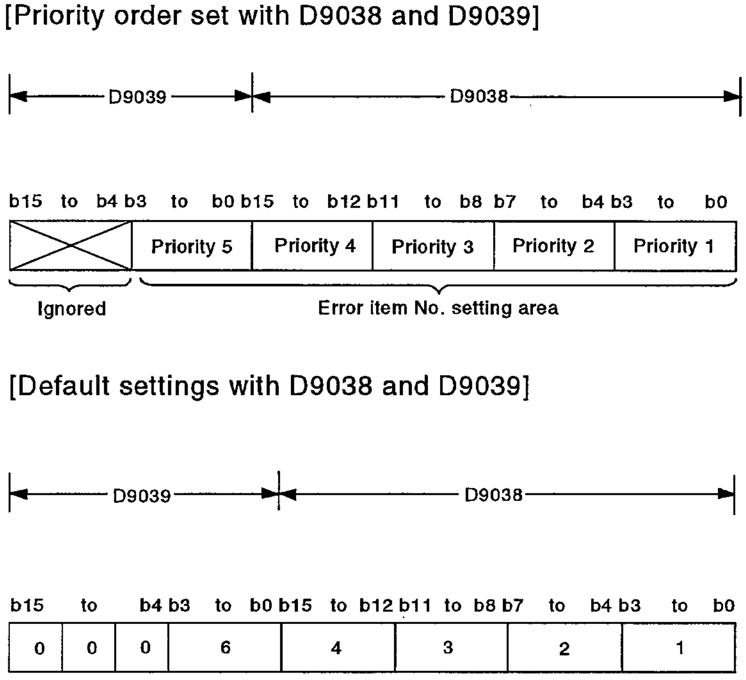
(b) It is possible to reset an annunciator by using a LEDR instruction.
By setting the annunciator to the first priority, it is possible to reset the annunciator (F) by using a LEDR instruction regardless of other error contents.
(The LEDR instruction cannot be used for resetting the annunciator (F) when an error whose ERR. LED indication priority is higher than the annunciator occurs.)

(1) Default priority settings for the ERR. LED indication are as given in Table 6.5.

Table 6.5 Error Indication Priority

Priority	Error Contents	Error Item No.	ERROR LED State at an Error Occurrence
High	Error with which the operation stops unconditionally	—	Lit
↑	I/O module verify Fuse blown	1	
	Special module error Operation error Link parameter error SFC parameter error SFC operation error	2	
	CHK instruction execution	3	Unlit
Low	Annunciator (F) on	4	Flicker
	Battery error	6	Lit

(2) Changing the priority
Change the ERR. LED indication priority by designating an error item number given in Table 6.5 with D9038 and D9039 (LED indication priority storage registers).
The priority order and default settings (settings by the PC CPU initial processing) to be set with D9038 and D9039 are as given in Table 6.25.



[Priority order set with D9038 and D9039]

b15 to b4	b3 to b0	b15 to b12	b11 to b8	b7 to b4	b3 to b0
☒	Priority 5	Priority 4	Priority 3	Priority 2	Priority 1

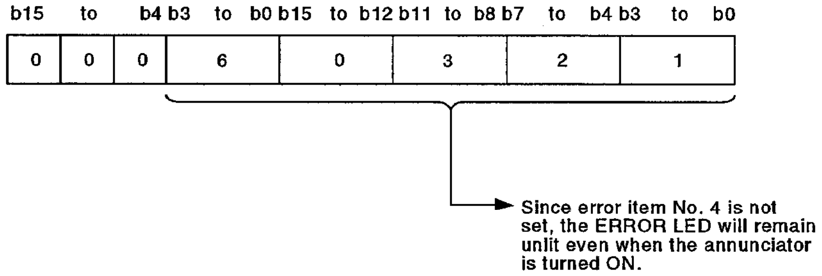
[Default settings with D9038 and D9039]

b15	to	b4	b3 to b0	b15 to b12	b11 to b8	b7 to b4	b3 to b0
0	0	0	6	4	3	2	1

Fig. 6.25 Priority Settings with D9038 and D9039

POINTS

(1) To set the ERROR LED to be unlit when an error given in Table 6.5 occurs, set the error item No. in D9038 or D9039 to 0.
Example: To set the ERROR LED to remain unlit when the annunciator is turned ON, change error item No. 4 to 0.



b15	to	b4	b3 to b0	b15 to b12	b11 to b8	b7 to b4	b3 to b0
0	0	0	6	0	3	2	1

Since error item No. 4 is not set, the ERROR LED will remain unlit even when the annunciator is turned ON.

(2) Even though the ERROR LED is set to remain unlit, M9008 (CPU error flag) is turned ON and the error code is stored in the CPU error register.

Applicable CPU	**All Types of CPUs**
Remark	

7. METHOD OF DATA COMMUNICATION WITH SPECIAL-FUNCTION MODULE

The following describes the method of reading data to the PC CPU from a special-function module and writing data to a special-function module from the PC CPU.

(1) Special-function modules

Special-function modules are used to deal with the analog data and high-speed pulses in the PC CPU which can not be processed only with the I/O modules.
For example, analog data is converted to digital data through an analog-digital converter module of the special-function module to be used with the PC CPU.
The special-function modules have memory (buffer memory) in which data received from the outside and data output to the outside is stored.

(2) Read/write of data from the PC CPU

Read and write data with the FROM/TO instruction.
When the FROM/TO instruction is executed, data stored in the buffer memory of the special-function module is read, or data is written to the buffer memory when the FROM/TO instruction is executed.

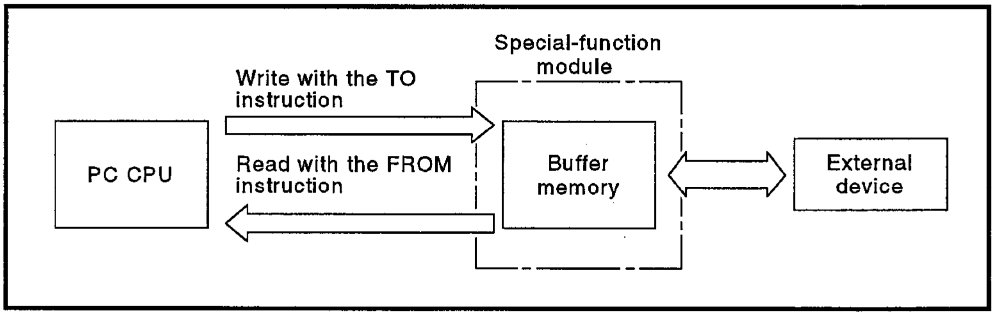


Fig. 7.1 Data Communication with A Special-Function Module

REMARKS

(1) Refer to the ACPU Programming Manual (Common Instructions) (IB-66250) for details on the FROM/TO instruction.

(2) Refer to the manual of the special-function module for details on the buffer memory of the special-function module.

8. PARAMETER SETTING

(1) Parameters are used to specify the allocation of the user memory area in the CPU module and the ranges of use of various functions.
Parameter data is stored in the head 3 k bytes of the user memory area.

(2) The default value has been arranged for parameter data as shown in Table 8.1.
Parameter data utilize the default values.

(3) Parameter data can be changed within the setting range shown in Table 8.1 according to the purpose.
Execute parameter setting with a peripheral device.
Refer to the Operating Manual of each peripheral device for the operation of parameter setting.

REMARKS

(1) The conversion from the setting unit to the number of bytes used with the main sequence program and the subsequence program is as shown in Table 8.1.

Item	Setting Unit	Number of Bytes
Main sequence program capacity	1 k step	2 k bytes
Subsequence program capacity		
File register capacity	1 k points	2 k bytes
Comment capacity	64 points	1 k bytes
Sampling trace memory capacity	128 times	1 k bytes

(2) As for the comment capacity, additional 1 k bytes are included in the displayed value, because 1 k bytes are added automatically when the comment capacity is set with a peripheral device.

Table 8.1 List of Parameter Setting Ranges

Item				Default Value	Setting Unit	Setting Range			
						A1 A1N	A1S A1S-S1 A1SJ-S3 A1SH A1SJH(S8) A0J2H	A2 A2N A2S A2SH A1FX	A2-S1 A2N-S1 A2S-S1 A2SH-S1
Common parameters	Allocation of user memory area	Sequence program memory capacity		6 k steps	1 k step	1 to 6 k steps	1 to 8 k steps	1 to 14 k steps	
		Subsequence program memory capacity		—	1 k step	—	—		
		File register capacity		—	1 k point	—	1 to 4 k points		
		Comment capacity		None	128 points/ 64 points	128 points (fixed)	0 to 4032 points		
		Status latch	Memory capacity	None		—	0/8 to 16 k bytes		
			Data memory				None/Provided		
			File register				None/Provided		
		Sampling trace	Memory capacity	—	128 times	—	0/8 k bytes		
			Device setting				Device number		
			Execution condition				Every scan Every hour		
			Sampling count				0 to 1024 times		
		Microcomputer program capacity		None	2 k bytes	0 to 10 k bytes	0 to 14 k bytes	0 to 26 k bytes	
	Latch (power failure backup) range setting	Link relay (B)		L1000 to 2047 only	1 point	B0 to 3FF			
		Timer (T)				T0 to 255			
		Counter (C)				C0 to 255			
		Data register (D)				D0 to 1023			
		Link register (W)				W0 to 3FF			

o : Setting possible x : Setting impossible

											Peripheral Device Usable for Setting *	
	A3 A3N	A73	A3H A3M	A2C A52G	A2A A2U A2AS	A2A-S1 A2U-S1 A2AS-S1 A2AS-S30 A2USH-S1	A3A A3U	A4U	Q02 Q02H	Q06H	PU	Other than PU
	1 to 30 k steps			1 to 8 k steps	1 to 14 k steps		1 to 30 k steps		1 to 28 k steps	1 to 30 k steps	o	o
	1 to 30 k steps			—	—		1 to 30 k steps	1 to 30 k steps (up to 3 programs can be set)	—	1 to 30 k steps	o	o
	1 to 8 k points			1 to 4 k points	1 to 8 k points		0 to 8 k points				o	o
	0 to 4032 points										x	o
	0/8 k to 24 k bytes			0/8 to 16 k bytes	—						x	o
	None/Provided											
	None/Provided			None/ Provided								
	0/8 k bytes				—						x	o
	Device number											
	Every scan Every hour											
	0 to 1024 times											
	0 to 58 k bytes			0 to 14 k bytes	0 to 26 k bytes		0 to 58 k bytes				x	o
					AnA : B0 to BFFF, AnU : B0 to B1FFF				B0 to B1FFF		o	o
					T0 to T255, T256 to T2047							
					C0 to C255, C256 to C1024							
					AnA : D0 to D6143, AnU: D0 to D8191				D0 to D8191			
					AnA : W0 to WFFF, AnU : W0 to W1FFF				W0 to W1FFF			

* : "PU" and "Other than PU" in the Peripheral Devices Usable for Setting column correspond to the following devices:
PU : A8PUE
Other than PU : Devices which are not using GPP function software package and which can set the CPU type by PC type setting.

Table 8.1 List of Parameter Setting Ranges (continued)

Item			Default Value	Setting Unit	Setting Range				
					A1 A1N	A1S A1S-S1 A1SJ-S3 A1SH A1SJH(S8) A0J2H	A2 A2N A2S A2SH A1FX	A2-S1 A2N-S1 A2S-S1 A2SH-S1	
Common parameters	Link range setting	Number of link stations	None	16 points	1 to 64				
		Input (X)			X0 to FF	X0 to FF A0J2H, A1S-S1: X0 to 1FF	X0 to 1FF	X0 to 3FF	
		Output (Y)			Y0 to FF	Y0 to FF A0J2H, A1S-S1: Y0 to 1FF	Y0 to 1FF	Y0 to 3FF	
		Link relay (B)			B0 to 3FF				
		Link register (W)		1 point	W0 to 3FF				
	Internal relay (M), latch relay (L), and step relay (S) setting		M0 to 999 L1000 to 2047	1 point	M/L/S 0 to 2047 M, L, and S are continuous numbers				
	Watchdog timer setting		200 ms	10 ms	10 ms to 2000 ms				
	Timer setting	T0 to T255	100 ms : T0 to 199 10 ms : T200 to 255	8 points	256 points for 100 ms, 10 ms, and retentive timers Timers must be continuous numbers				
	Counter setting	Interrupt counter setting	None	8 points	256 points for counters and interrupt counters Counters must be continuous numbers.				

o : Setting possible   x : Setting impossible

											Peripheral Device Usable for Setting *	
	A3 A3N	A73	A3H A3M	A2C A52G	A2A A2U A2AS	A2A-S1 A2U-S1 A2AS-S1 A2AS-S30 A2USH-S1	A3A A3U	A4U	Q02 Q02H	Q06H	PU	Other than PU
											x	o
	X0 to 7FF			X0 to 1FF	X0 to 1FF	X0 to 3FF	X0 to 7FF	X0 to FFF				
	Y0 to 7FF			Y0 to 1FF	Y0 to 1FF	Y0 to 3FF	Y0 to 7FF	Y0 to FFF				
					B0 to 1FFF							
					W0 to 1FFF							
					M/L/S 0 to 8191 M, L, and S are continuous numbers						o	o
			200 ms fixed	100 ms to 2000 ms	200 ms fixed						o	o
											o	o
			Interrupt count counters (C224 to 225) setting	—	Interrupt count counters (C224 to 255) setting						o	o

* : "PU" and "Other than PU" in the Peripheral Devices Usable for Setting column correspond to the following devices:
PU : A8PUE
Other than PU : Devices which are not using GPP function software package and which can set the CPU type by PC type setting.

Table 8.1 List of Parameter Setting Ranges (continued)

Item			Default Value	Setting Unit	Setting Range				
					A1 A1N	A1S A1S-S1 A1SJ-S3 A1SH A1SJH(S8) A0J2H	A2 A2N A2S A2SH A1FX	A2-S1 A2N-S1 A2S-S1 A2SH-S1	
Common parame-ters	I/O number allocation	Input (X) module	None	16 points	0 to 64 points				
		Output (Y) module							
		Special func-tion module							
		Vacant slot							
	Remote RUN/PAUSE contact setting		None	1 point	X0 to FF	X0 to FF A0J2H, A1S-S1 : X0 to 1FF	X0 to 1FF	X0 to 3FF	
	Operation mode at error occurrence	Fuse blown	Continue	—	Stop/Continue				
		I/O verify error	Stop						
		Operation error	Continue						
		Special function module check error	Stop						
	Annunciator display mode		—	—	—				
	STOP → RUN display mode		Operation state before STOP is output.	—	Output of state before STOP or after operation execution				
	Print title registration		None	—	128 characters using all keys on MELSAP				
	Entry code registration		None	—	Hexadecimal (0 to 9, A to F) Max. 6 digits				
Parameter for A2C	Remote terminal setting	Total number of slave stations	—	—	—				
		Protocol							
		Head station number							
		Mode setting							
Parameter for A3H ad A3M	I/O control setting		—	—	—				

o : Setting possible x : Setting impossible

											Peripheral Device Usable for Setting *	
	A3 A3N	A73	A3H A3M	A2C A52G	A2A A2U A2AS	A2A-S1 A2U-S1 A2AS-S1 A2AS-S30 A2USH-S1	A3A A3U	A4U	Q02 Q02H	Q06H	PU	Other than PU
				—	0 to 64 points Module type can be entered.						x	o
	X0 to 7FF			X0 to 1FF	X0 to 1FF	X0 to 3FF	X0 to 7FF	X0 to FFF			x	o
											x	o
	Alternate display of F number / F number with comment			—	—		Alternate display of F number / F number with comment		—		x	o
											x	o
											x	o
											o	o
				1 to 64	—						x	o
				MINI standard / No protocol								
				1 to 61								
				0: Automatic online return enabled 1: Automatic online return disabled 2: Transmission stop at online fault								
	Direct/refresh setting for input and output independently			—							o	o

* : "PU" and "Other than PU" in the Peripheral Devices Usable for Setting column correspond to the following devices:
PU : A8PUE
Other than PU : Devices which are not using GPP function software package and which can set the CPU type by PC type setting.

Table 8.1 List of Parameter Setting Ranges (continued)

Item			Default Value	Setting Unit	Setting Range				
					A1 A1N	A1S A1S-S1 A1SJ-S3 A1SH A1SJH(S8) A0J2H	A2 A2N A2S A2SH A1FX	A2-S1 A2N-S1 A2S-S1 A2SH-S1	
Parameters for the AnA, and AnU	Extension timer	T256 to T2047	—	—	—				
		Number of points in use							
		Set value device							
	Extension counter	Number of points in use	—	—	—				
		Set value device							
	MELSECNET -II link range setting	Number of link stations	—	—	—				
		Link relay							
		Link register							
	MELSECNET/ MINI / MELSECNET/ MINI-S3	Number of modules supported	—	—	—				
		Head I/O number							
		Type registration							
		Received data							
		Send data							
		Retry count							
		FROM/TO response designation							
		Faulty station data clear designation							
		Faulty station detection							
		Error No.							
		Total number of remote stations							
		Transmission condition setting in line fault							

o : Setting possible x : Setting impossible

										Peripheral Device Usable for Setting *	
	A3 A3N	A73	A3H A3M	A2C A52G	A2A A2U A2AS	A2A-S1 A2U-S1 A2AS-S1 A2AS-S30 A2USH-S1	A3A A3U	A4U	Q02 Q02H	Q06H	PU	Other than PU
					1792 points for 100 ms, 10 ms, and retentive timers Timers must be continuous numbers				1792 points for 100 ms, 10 ms, retentive timer and 1 ms. Timers must be continuous numbers		o	o
					0 to 2048 points							
					D, W, and R when the number of points exceeds 256							
					0 to 1024 points						o	o
					D, W, and R when the number of points exceeds 256							
					0 to 64 stations						x	o
					B0 to BFFF							
					W0 to WFFF							
					0 to 8 modules						x	o
					o to 1F0	0 to 3F0	0 to 7F0	0 to FF0				
					MINI, MINI-S3							
					X, M, L, B, T, C, D, W, and R are not provided. (Bit devices are set in 16-point units.)							
					Y, M, L, B, T, C, D, W, and R are not provided. (Bit devices are set in 16-point units.)							
					0 to 32 times							
					Priority to link/CPU							
					Hold/Clear							
					M, L, B, T, C, D, W, and R are not provided. (Bit devices are set in 16-point units.)							
					T, C, D, W, R							
					0 to 64 stations							
					Test message / OFF data hold (send data)							

* : "PU" and "Other than PU" in the Peripheral Devices Usable for Setting column correspond to the following devices:

PU	: A8PUE
Other than PU	: Devices which are not using GPP function software package and which can set the CPU type by PC type setting.

9. CONFIGURATION OF USER MEMORY AREA

This chapter gives the configuration of the user memory area of the PC CPU.

(1) The following programs can be stored in the sequence program area of the PC CPU:

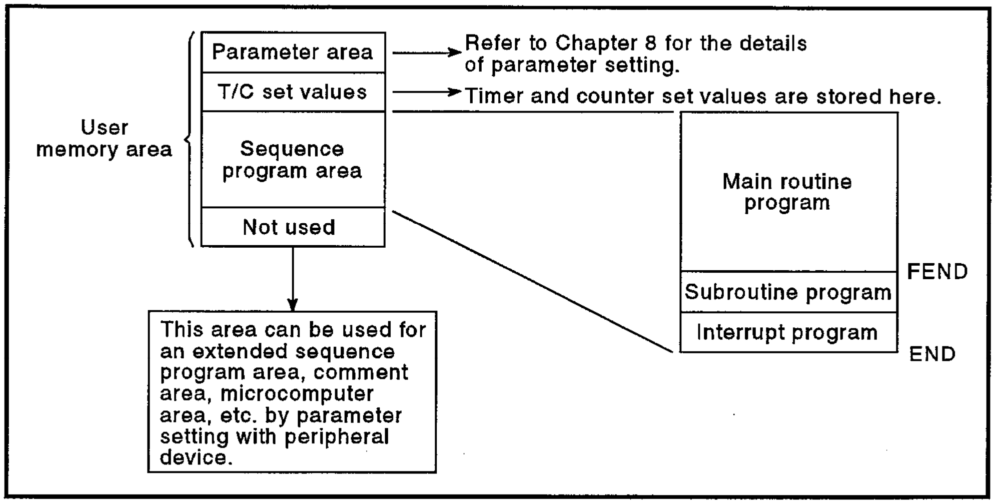

User memory area
Parameter area
T/C set values
Sequence program area
Not used
Refer to Chapter 8 for the details of parameter setting.
Timer and counter set values are stored here.
Main routine program
FEND
Subroutine program
Interrupt program
END
This area can be used for an extended sequence program area, comment area, microcomputer area, etc. by parameter setting with peripheral device.

Fig. 9.1 Configuration of the Sequence Program Area

(2) The user memory area can be used for a sequence program area, comment area, file register area, etc. by parameter setting with peripheral device.

(a) When parameter setting is not done
A parameter area, T/C set value area, sequence program area of 6 k steps are allocated beginning with the head of the user memory area.

1) Parameter area
The area where the parameters given in Table 8.1 are stored.

2) T/C set value area
The area where the timer and and counter set values used with the sequence program are stored.

3) Sequence program area
The area where the main routine programs, subroutine programs, and interrupt programs are stored. (Refer to Section 5.1 for the details of the main routine programs, subroutine programs, and interrupt programs.)

(b) When parameter setting is done

1) A parameter area, T/C set value area, sequence program area, and sub-program area are allocated beginning with the head of the user memory area.

2) Comments and file registers set with parameters given in Table 8.1 are allocated beginning with the end of the user memory area.

9.1 A1CPU and A1NCPU

Configuration of the user memory area of the A1 and A1NCPUs is the same regardless of the memory type (IC-RAM, EPROM, E^{2}PROM) installed. For the operation using a ROM, parameters, T/C set values, sequence program, and microcomputer program in the user memory area can be stored to the ROM.

(1) When parameter setting is not done
The user memory area is set with defaults and has a configuration as shown in Fig. 9.2.

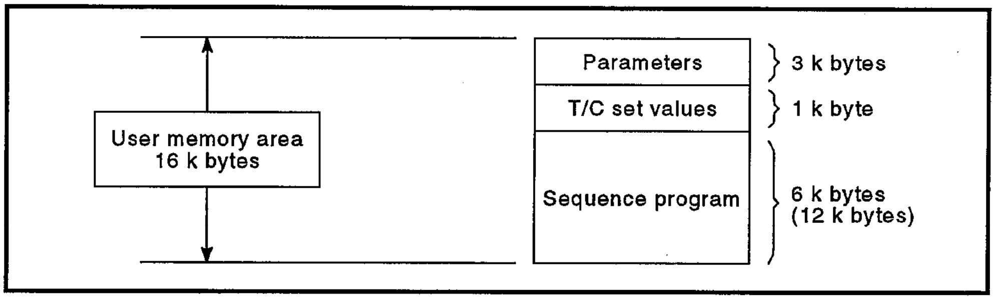


Fig. 9.2 Cofiguration of the User Memory Area

(2) When parameter setting is done

(a) The sequence program area can be changed to the microcomputer program area by parameter setting.(User-created microcomputer programs and utility programs can be stored in the microcomputer program area.)

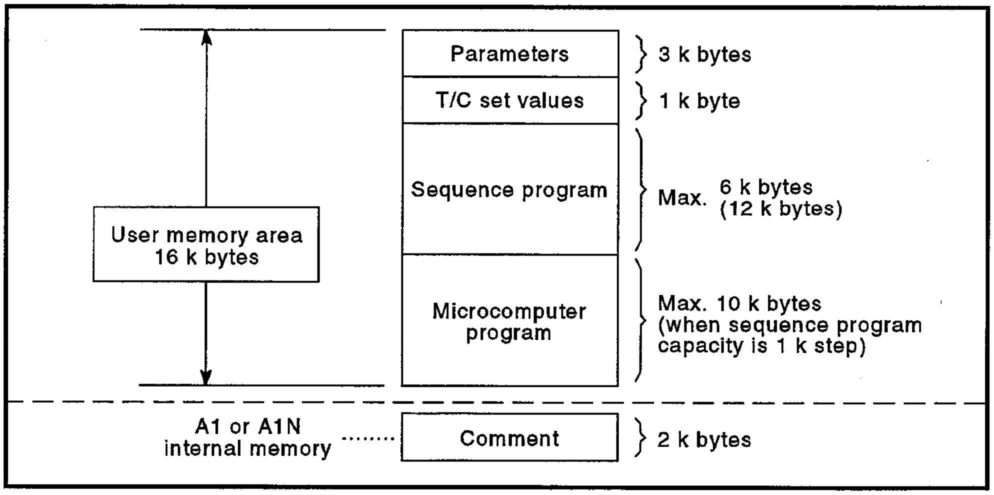


FIg. 9.3 Configuration of the User Memory Area and Comment Area

POINTS

(1) Comments are stored in the comment area (internal memory) of the A1 or A1N. The 16 k byte user memory area shown in Fig. 9.3 will not be reduced even when a comment area is set by parameter setting.

(2) Comments, which are created by using a peripheral device, of 124 points from F0 to F123 can be stored in the comment area of the A1 or A1N.

9.2 A2CPU(S1) and A2NCPU(S1)

Configuration of the user memory area of the A2(S1) and A2N(S1)CPUs varies according to the memory cassette type.
However, when parameter setting is not done, the user memory area is set with defaults regardless of the memory type and 16 k bytes are allocated.

(1) When parameter setting is not done
Configuration of the user memory area when parameter setting is not done is as shown in Fig. 9.4.

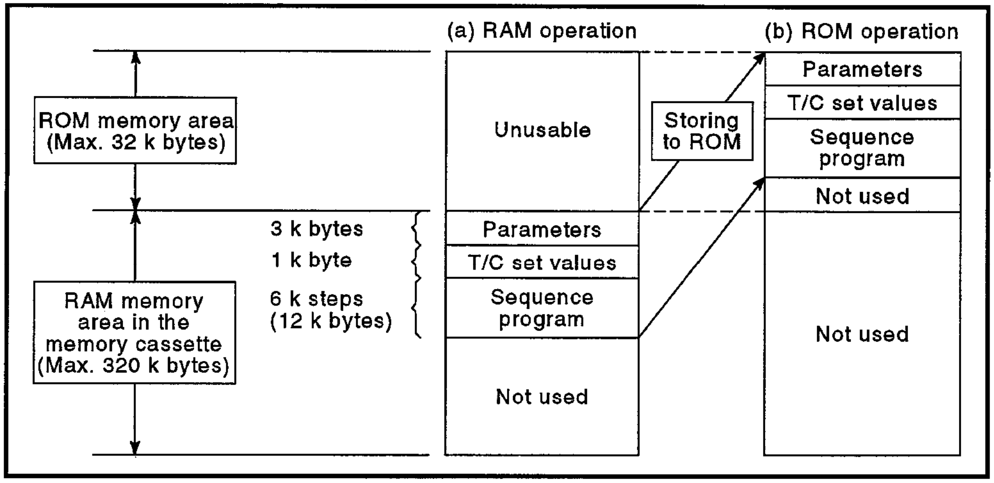


Fig. 9.4 Configuration of the User Memory Area

(2) When parameter setting is done
A maximum of 144 k bytes of the user memory area can be used by allocating it as shown in Fig. 9.5. An area exceeding 144 k bytes can be used as extension file registers by using the SW0GHP-UTLP-FN1 utility program.

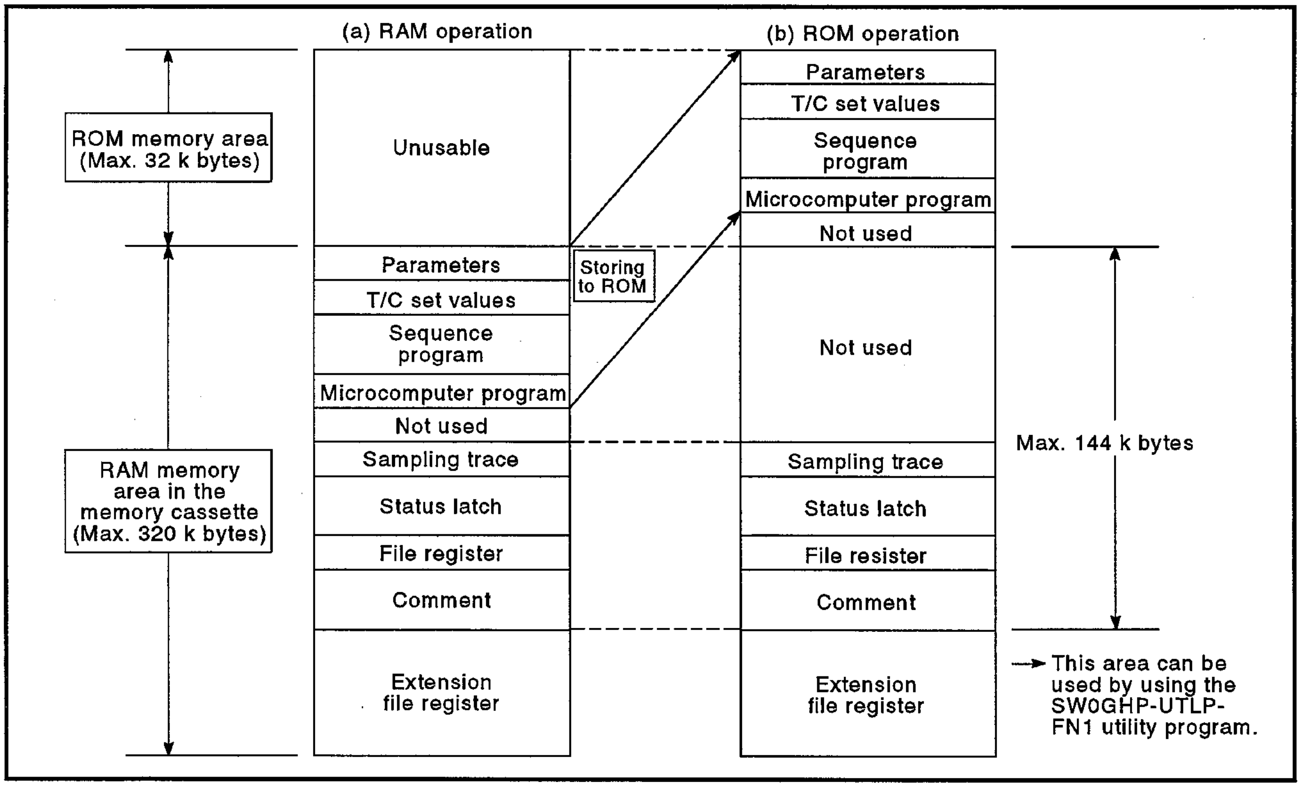


Fig. 9.5 Configuration of the User Memory Area

9.3 A0J2HCPU, A2CCPU, A52GCPU, A1SCPU, and A1SJCPU-S3

The A0J2HCPU, A2CCPU, A52GCPU, A1SCPU and A1SJCPU-S3 have a 32 k byte user memory area.

(1) When parameter setting is not done
Configuration of the user memory area when parameter setting is not done is as shown in Fig. 9.6.

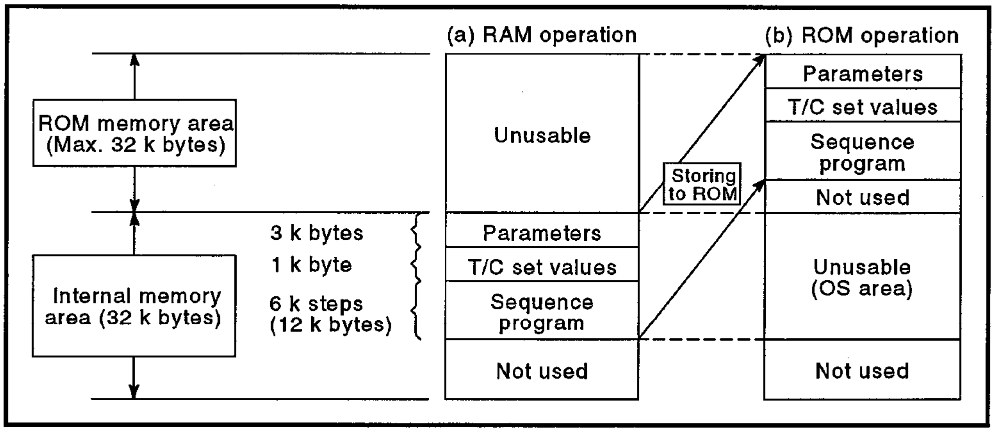


Fig. 9.6 Configuration of the User Memory Area

(2) When parameter setting is done
A maximum of 32 k bytes of the memory area can be used by allocating it as shown in Fig. 9.7.

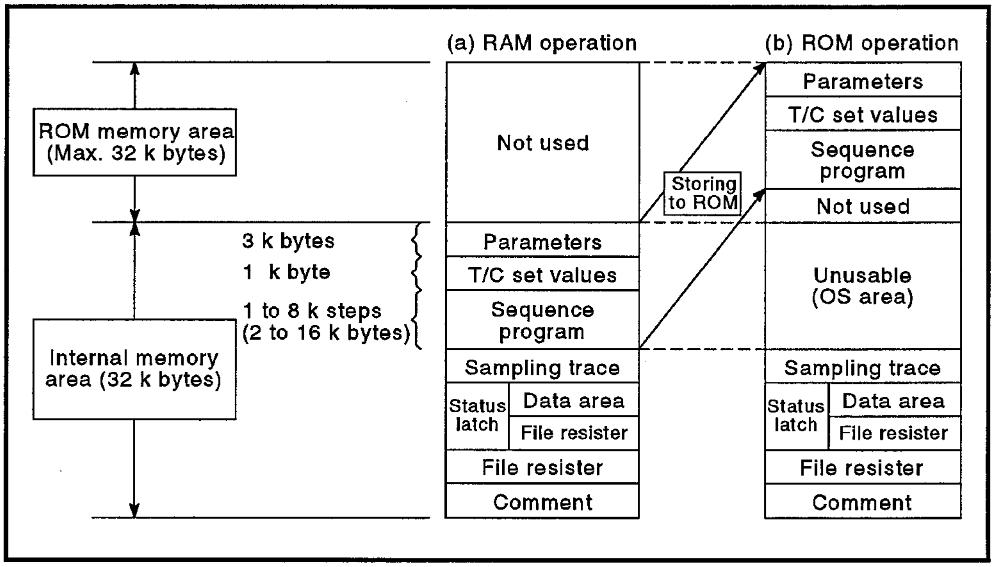


Fig. 9.7 Configuration of the User Memory Area

9.4 A2SCPU(S1)

The A2S and the A2S-S1 have user memory area of 64 k bytes and 192 k bytes, respectively.

(1) When parameter setting is not changed

Configuration of the user memory area when parameter setting is not changed is as shown in Fig. 9.8.

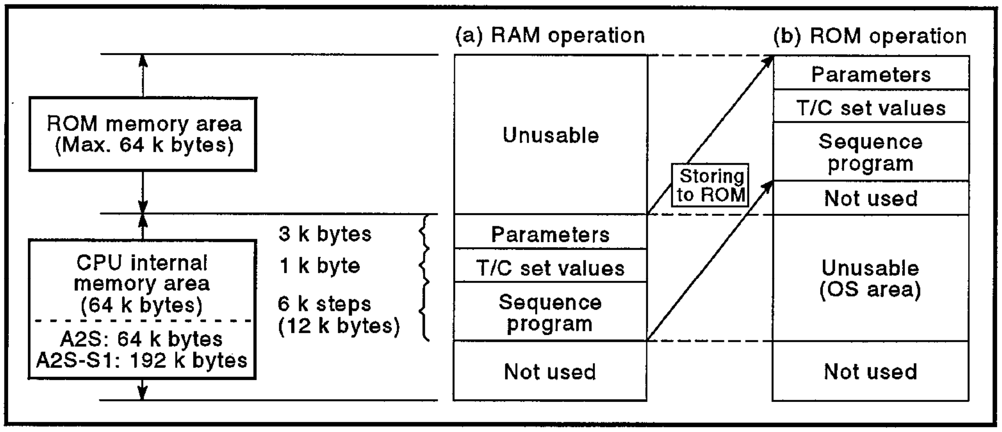
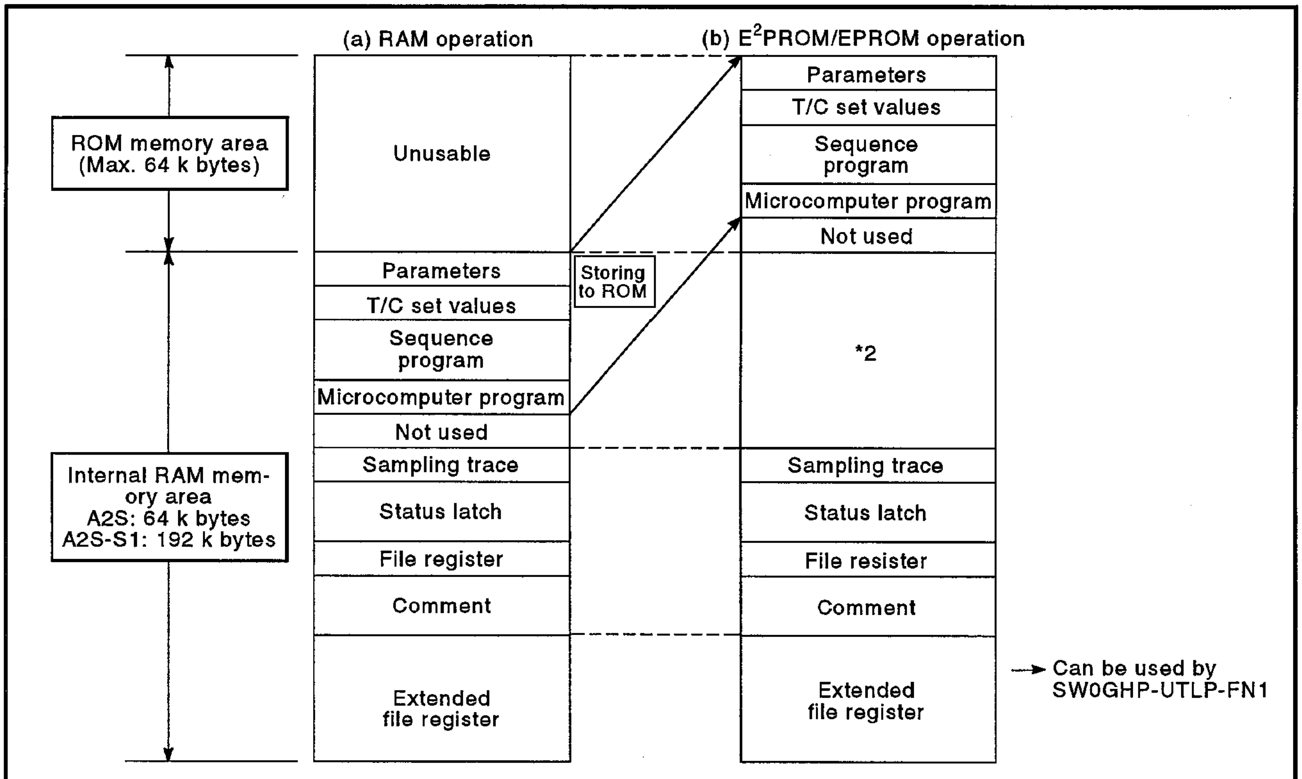


Fig. 9.8 Configuration of the User Memory Area

(2) When parameter setting is changed

By changing the parameter setting, it is possible to use the user memory area (64 k bytes, 192 k bytes) by allocating it as shown in Fig. 9.9.



*2 : E²PROM operation ..... Since this area is used by the system, the area is not usable.
EPROM operation .... The area is not used area and can be used for extended file register.

Fig. 9.9 Configuration of the User Memory Area

9.5 A1SHCPU(S8), A1SJHCPU and A2HCPU(S1)

The A1SH, A1SJH and A2SH, and the A2SH-S1 have user memory area of 64 k bytes and 192 k bytes, respectively.

(1) When parameter setting is not changed

Configuration of the user memory area when parameter setting is not changed is as shown in Fig. 9.10.

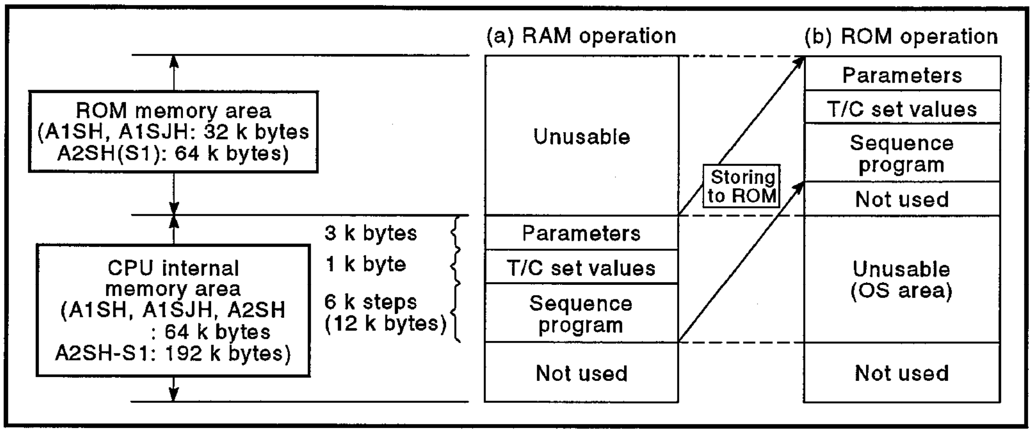


Fig. 9.10 Configuration of the User Memory Area

(2) When parameter setting is changed

By changing the parameter setting, it is possible to use the user memory area (64 k bytes, 192 k bytes) by allocating it as shown in Fig. 9.11.

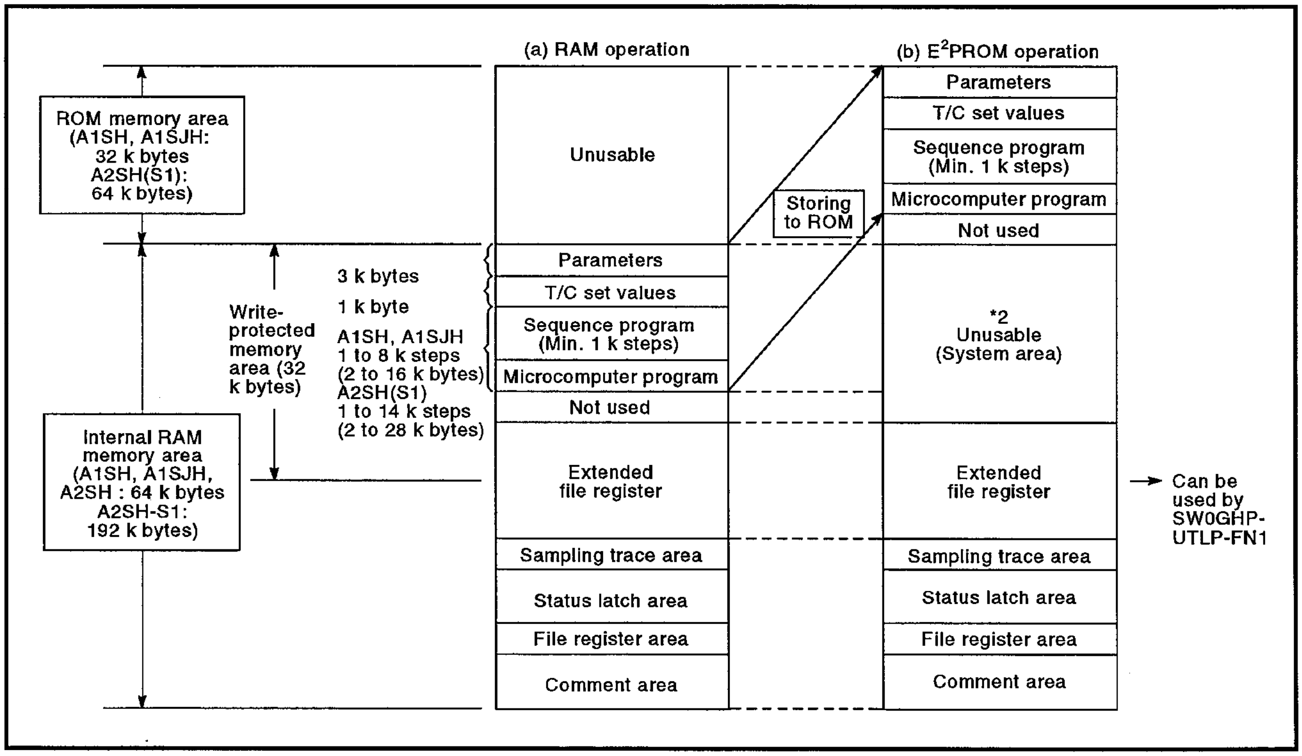


Fig. 9.11 Configuration of the User Memory Area

9.6 A1FXCPU

The A1FX has user memory area of 64 k bytes.

(1) When parameter setting is not changed

Configuration of the user memory area when parameter setting is not changed is as shown in Fig. 9.12.

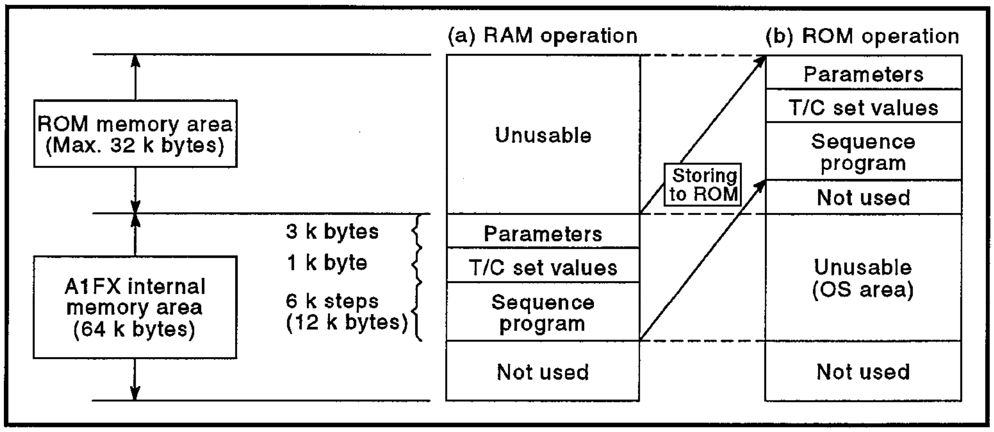


Fig. 9.12 Configuration of the User Memory Area

(2) When parameter setting is changed

By changing the parameter setting, it is possible to use the user memory area (64 k bytes) by allocating it as shown in Fig. 9.13.

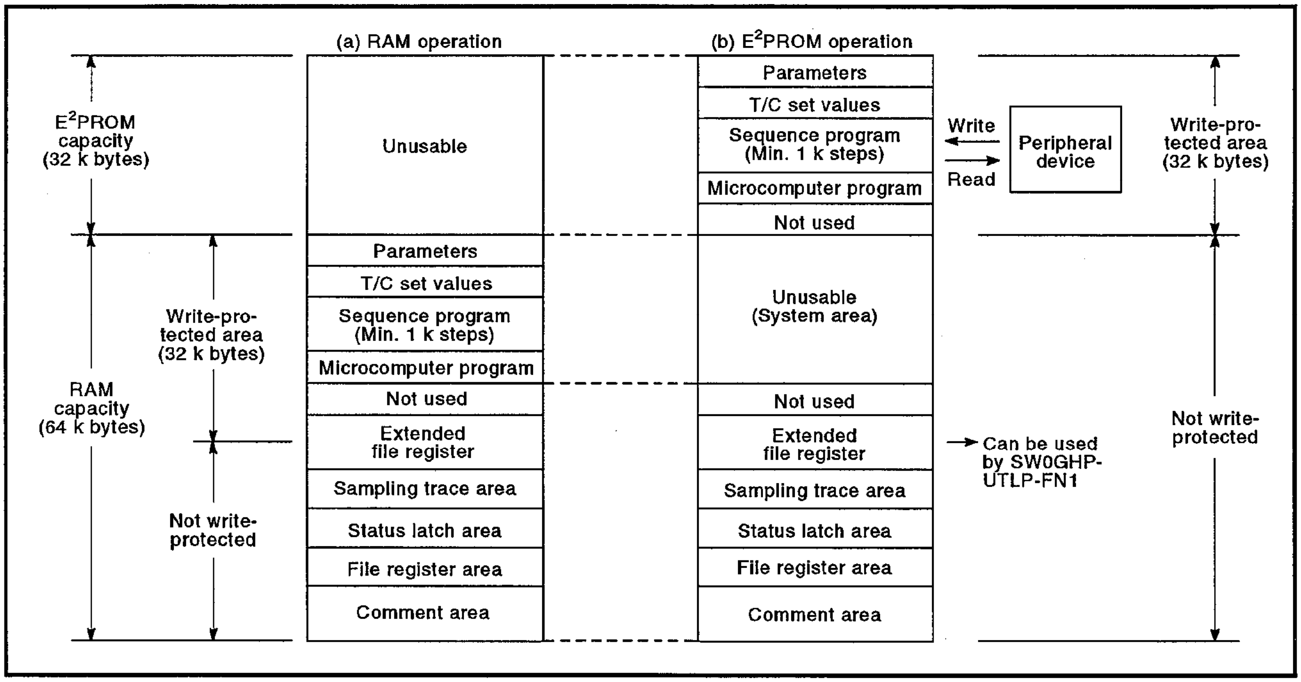


Fig. 9.13 Configuration of the User Memory Area

9.7 A2ASCPU, A2ASCPU-S1, A2ASCPU-S30, and A2USHCPU-S1

The A2AS, and the A2AS-S1, A2AS-S30 and A2USH-S1 have user memory area of 64 k bytes and 256 k bytes, respectively.

(1) When parameter setting is not changed

Configuration of the user memory area when parameter setting is not changed is as shown in Fig. 9.14.

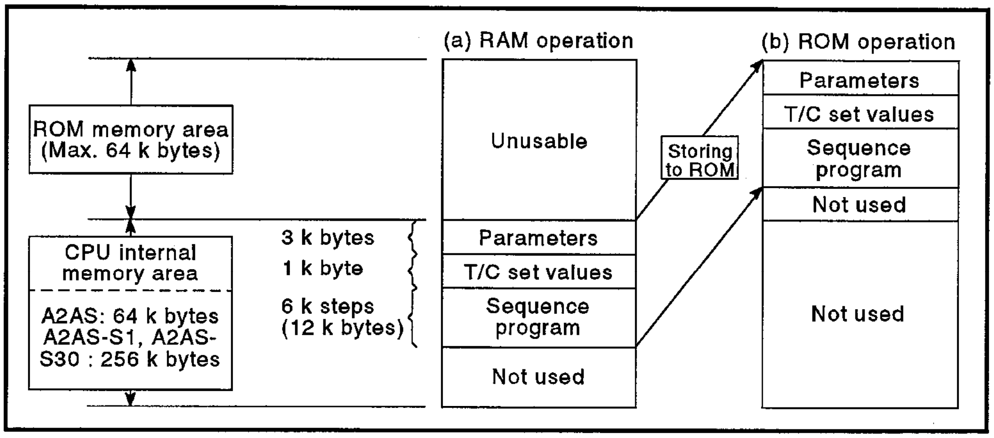
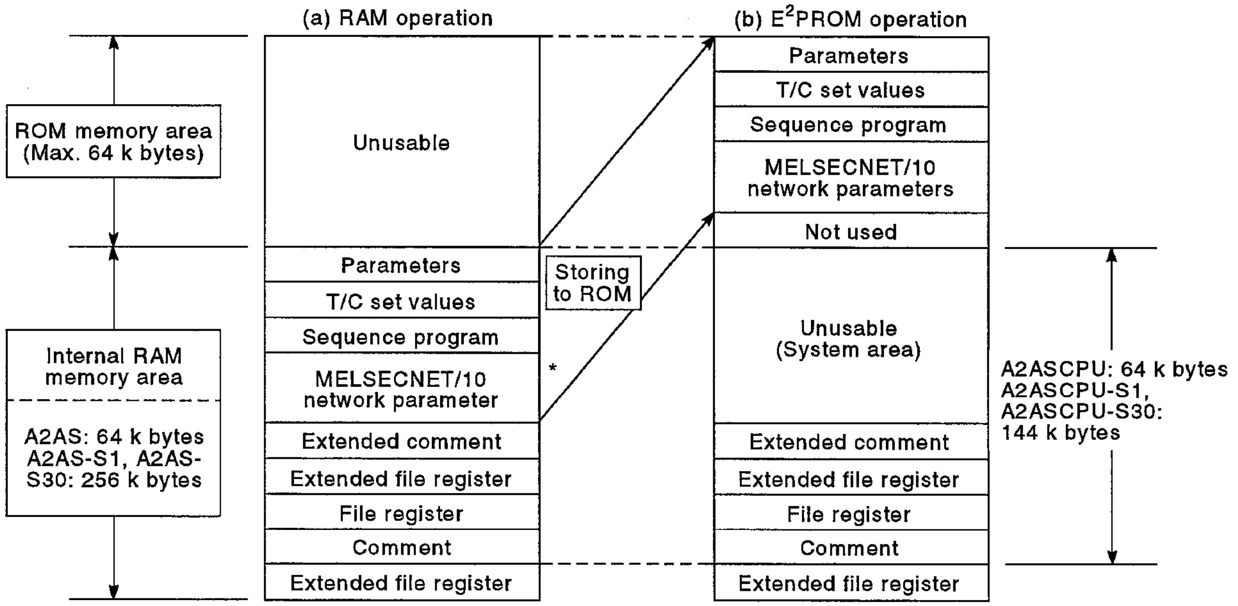


Fig. 9.14 Configuration of the User Memory Area

(2) When parameter setting is changed

By changing the parameter setting, it is possible to use the user memory area (64 k bytes, 256 k bytes) by allocating it as shown in Fig. 9.15.



* : When MELSECNET(II) data link system is constructed using the GPP function software package which is compatible to AnU, 2 k bytes (equivalent to 1 k step) are occupied for link paramaeter area.

Fig. 9.15 Configuration of the User Memory Area

9.8 A3CPU, A3NCPU, A3VCPU, A73CPU, and A3HCPU

Configuration of the user memory area of the A3CPU, A3NCPU, A3VCPU, A73CPU, and A3HCPU varies according to the memory cassette type. However, when parameter setting is not done, the user memory area is set with defaults regardless of the memory type and 16 k bytes are allocated.

(1) When parameter setting is not done
Configuration of the user memory area when parameter setting is not done is as shown in Fig. 9.16.

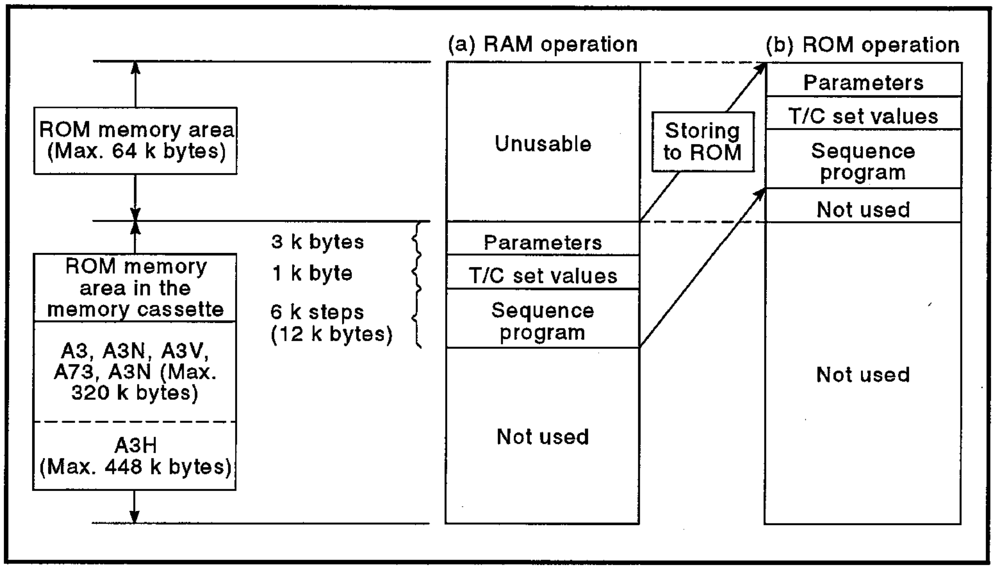


Fig. 9.16 Configuration of the User Memory Area

(2) When parameter setting is done
A maximum of 144 k bytes of the user memory area can be used by allocating it as shown in Fig. 9.17.
Unused areas can be used as extension file registers by using the SW0GHP-UTLP-FN1 utility program.

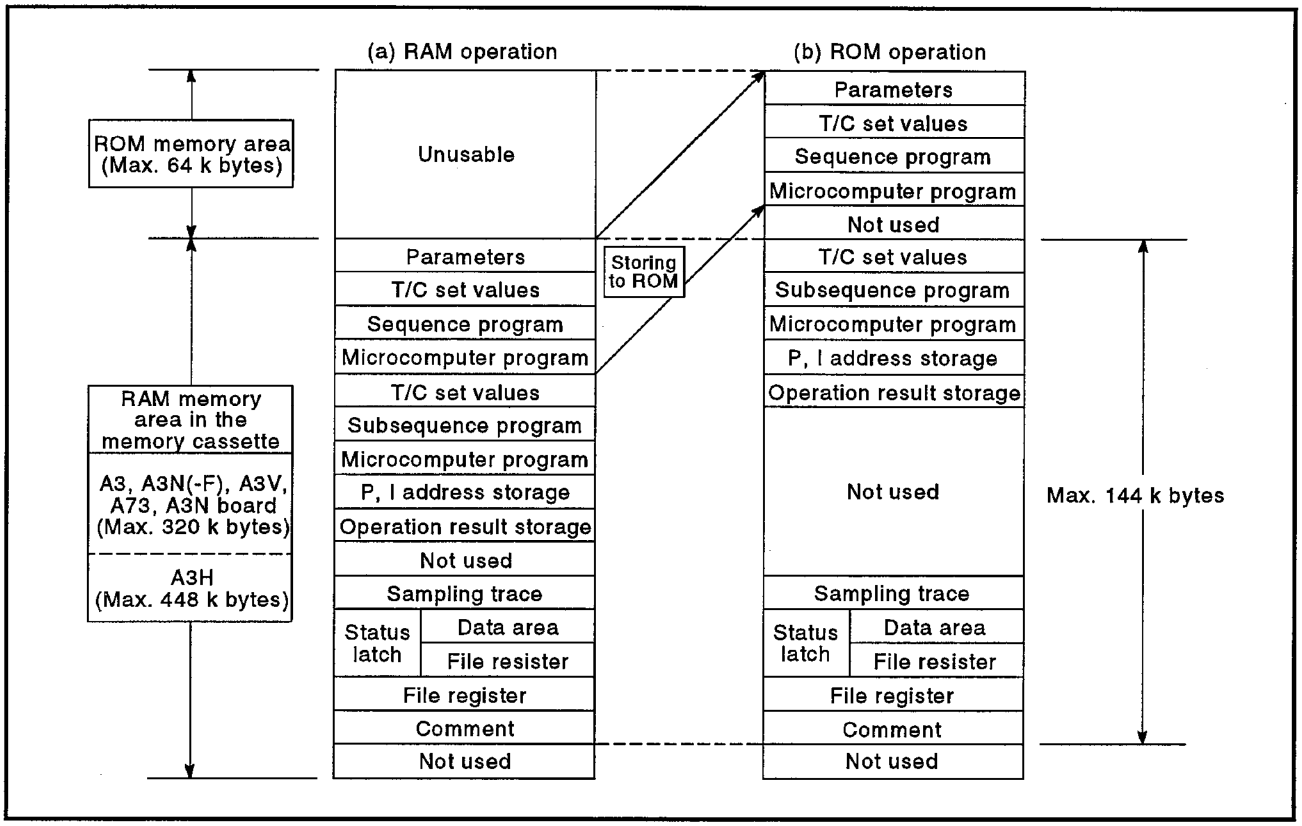


Fig. 9.17 Configuration of the User Memory Area

9.9 A3MCPU

Configuration of the user memory area of the A3MCPU varies according to the memory cassette type. However, when parameter setting is not done, the user memory area is set with defaults regardless of the memory type and 16 k bytes are allocated.

(1) When parameter setting is not done
Configuration of the user memory area when parameter setting is not done is as shown in Fig. 9.18.

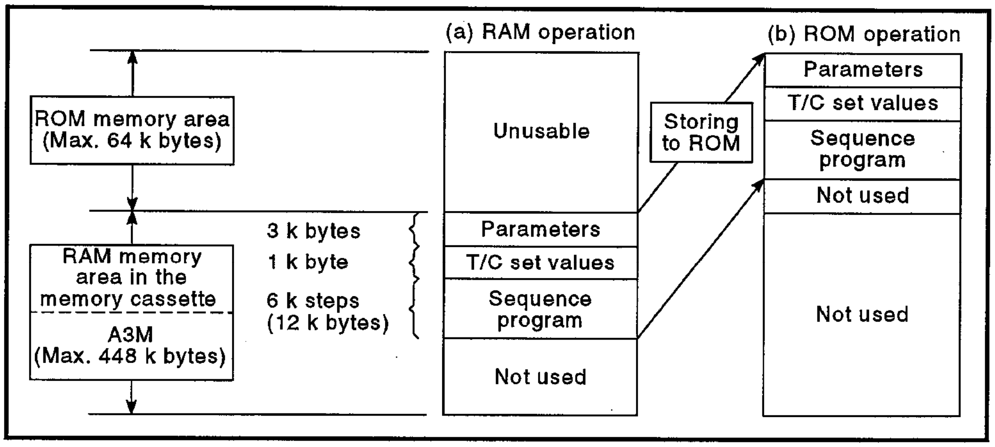
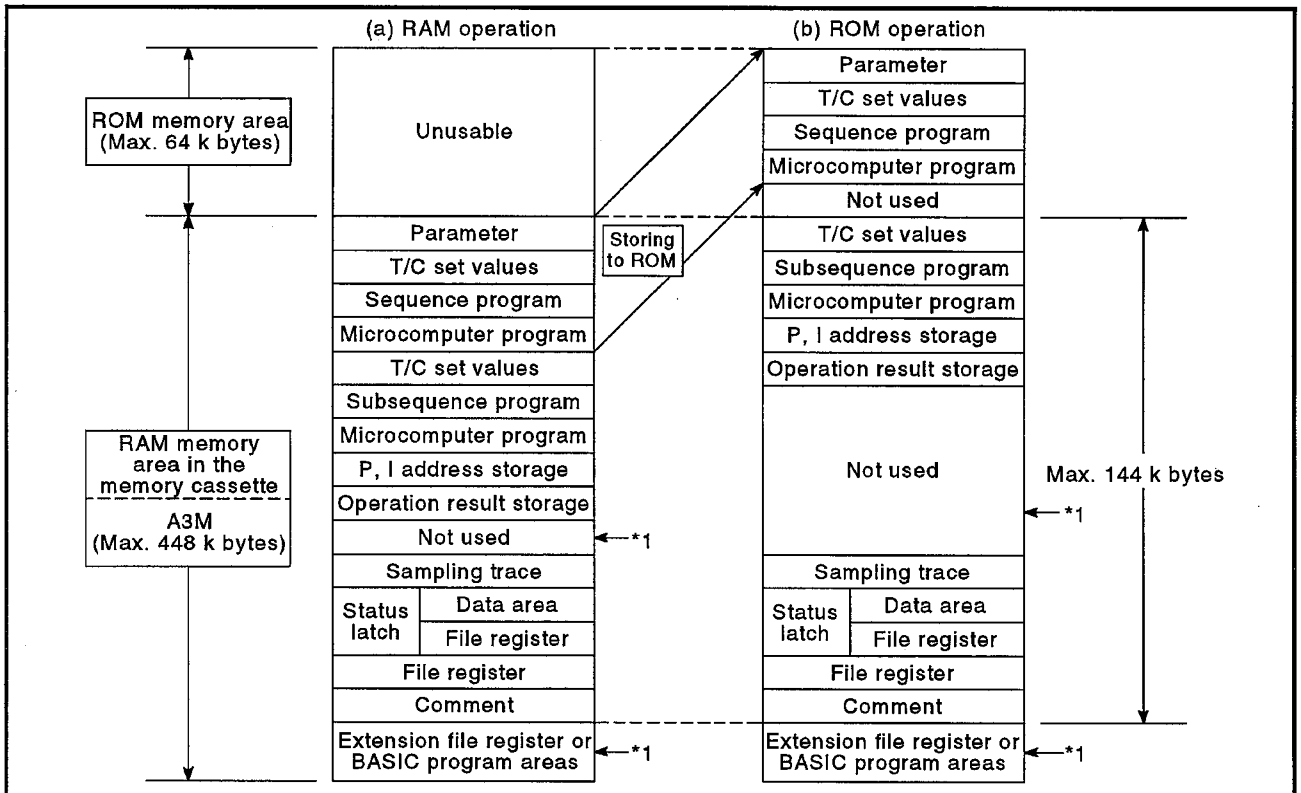


Fig. 9.18 Configuration of the User Memory Area

(2) When parameter setting is done
A maximum of 144 k bytes of the user memory area be used by allocating it as shown in Fig. 9.19.
Areas exceeding 144 k bytes can be used as extension file registers or BASIC program areas.



*1 : This area can be used as extension file registers by using the SW0GHP-UTLP-FN1 utility program.

Fig. 9.19 Configuration of the User Memory Area

9.10 A2ACPU(S1) and A3ACPU

Configuration of the user memory area of the A2A(S1) and A3ACPUs varies according to the memory cassette type. However, when parameter setting is not done, the user memory area is set with defaults regardless of the memory type and 16 k bytes are allocated.

(1) When parameter setting is not done
Configuration of the user memory area when parameter setting is not done is as shown in Fig. 9.20.

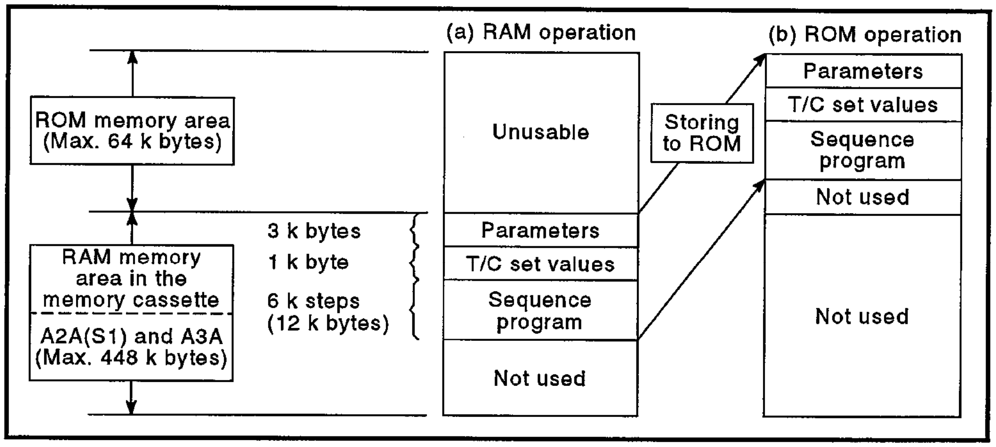
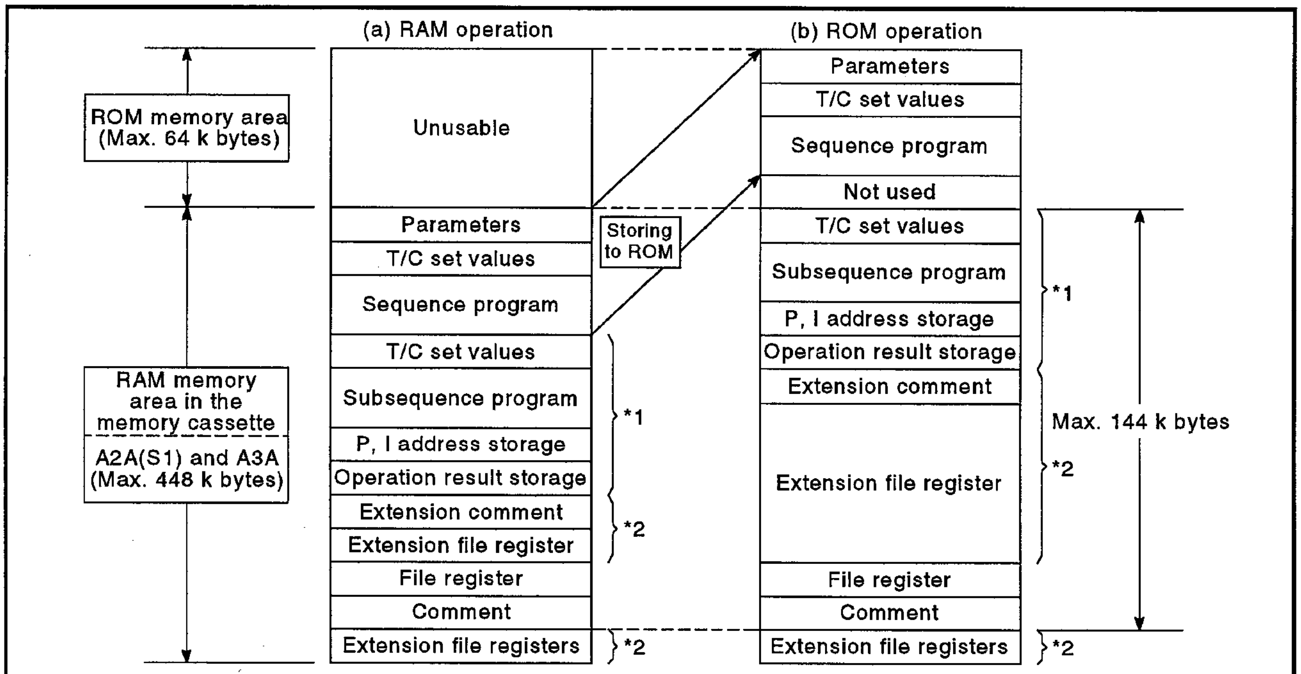


Fig. 9.20 Configuration of the User Memory Area

(2) When parameter setting is done
A maximum of 448 k bytes of the user memory area can be used by allocating it as shown in Fig. 9.21.



*1 : These areas can be set only with the A3A.
*2 : Refer to Section 3.11.2 for the details of extension file registers.

Fig. 9.21 Configuration of the User Memory Area

9.11 A2UCPU(S1) and A3UCPU

Configuration of the user memory area of the A2U(S1) and A3UCPUs varies according to the memory cassette type. However, when parameter setting is not done, the user memory area is set with defaults regardless of the memory type and 16 k bytes are allocated.

(1) When parameter setting is not done
Configuration of the user memory area when parameter setting is not done is as shown in Fig. 9.22.

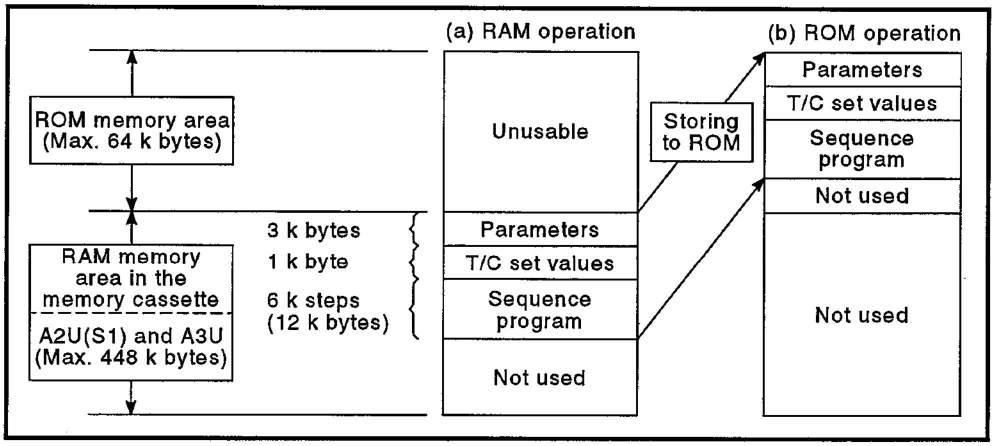
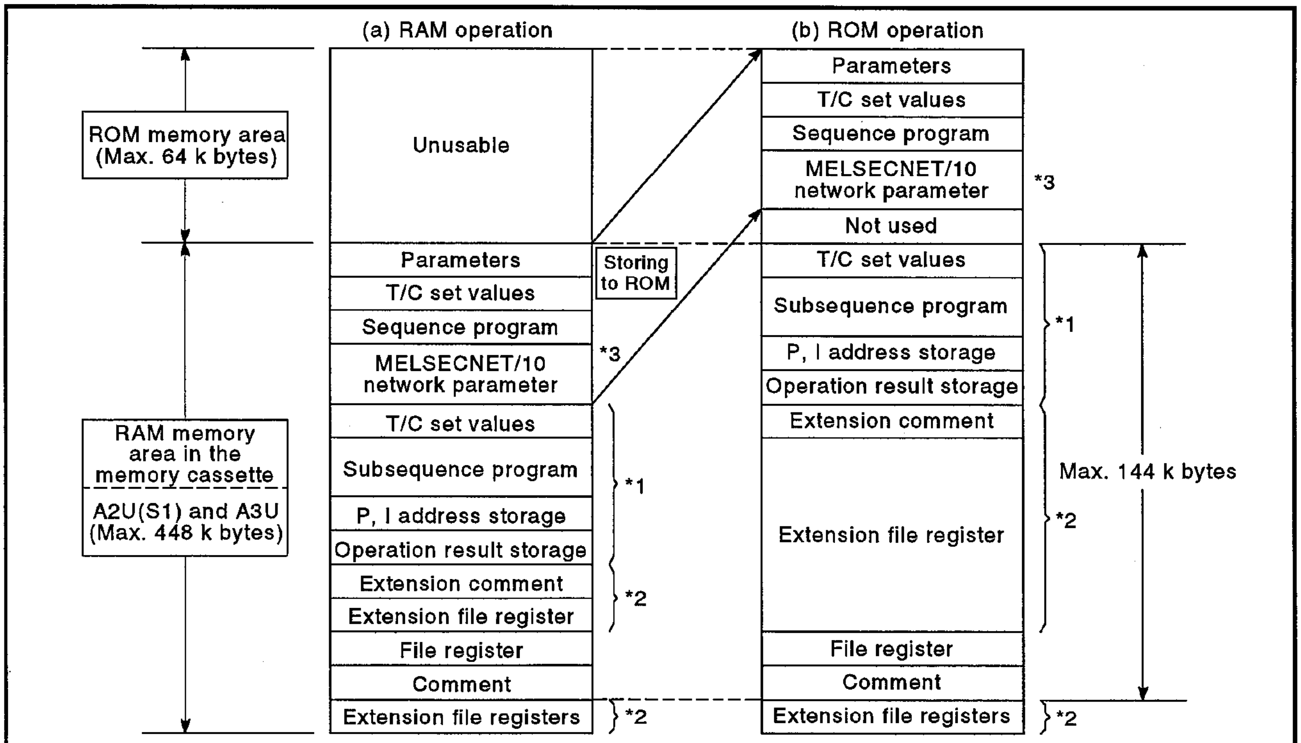


Fig. 9.22 Configuration of the User Memory Area

(2) When parameter setting is done
A maximum of 448 k bytes of the user memory area can be used by allocating it as shown in Fig. 9.23.



*1 : These areas can be set only with the A3U.
*2 : Refer to Section 3.11.2 for the details of extension file registers.
*3 : Refer to Section 11 for allocation of the MELSECNET/10 network parameters.
When MELSECNET(II) data link system is constructed using the GPP function software package which is compatible to AnU, 2 k bytes (equivalent to 1 k step) are occupied for link paramaeter area.

Fig. 9.23 Configuration of the User Memory Area

9.12 A4UCPU

Configuration of the user memory area of the A4UCPU varies according to the memory cassette type. However, when parameter setting is not done, the user memory area is set with defaults regardless of the memory type and 16 k bytes are allocated.
Sub-programs 2 and 3 can be set by using an A3AMCA-96 or A4UMCA-128 memory cassette with the A4U. Sub-programs 2 and 3 cannot be set by using a memory cassette other than those mentioned above.

(1) When parameter setting is not done
Configuration of the user memory area when parameter setting is not done is as shown in Fig. 9.24.

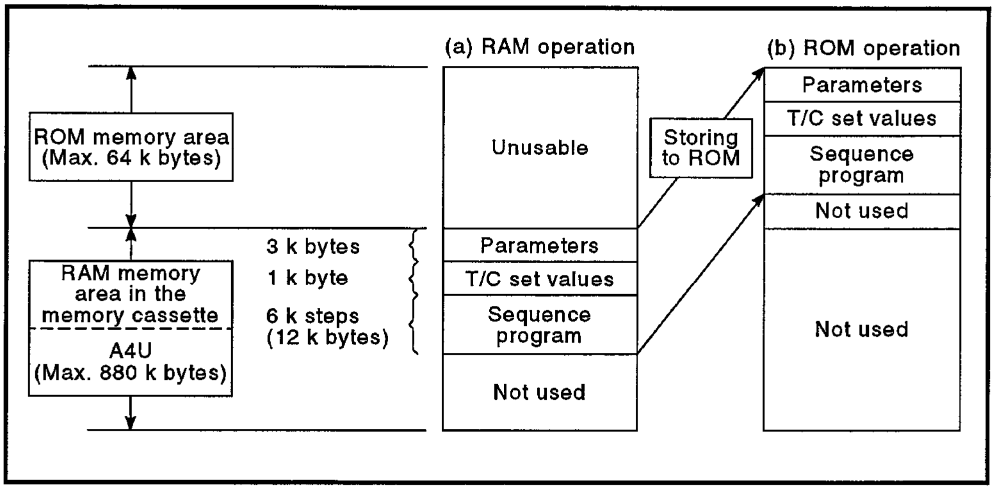
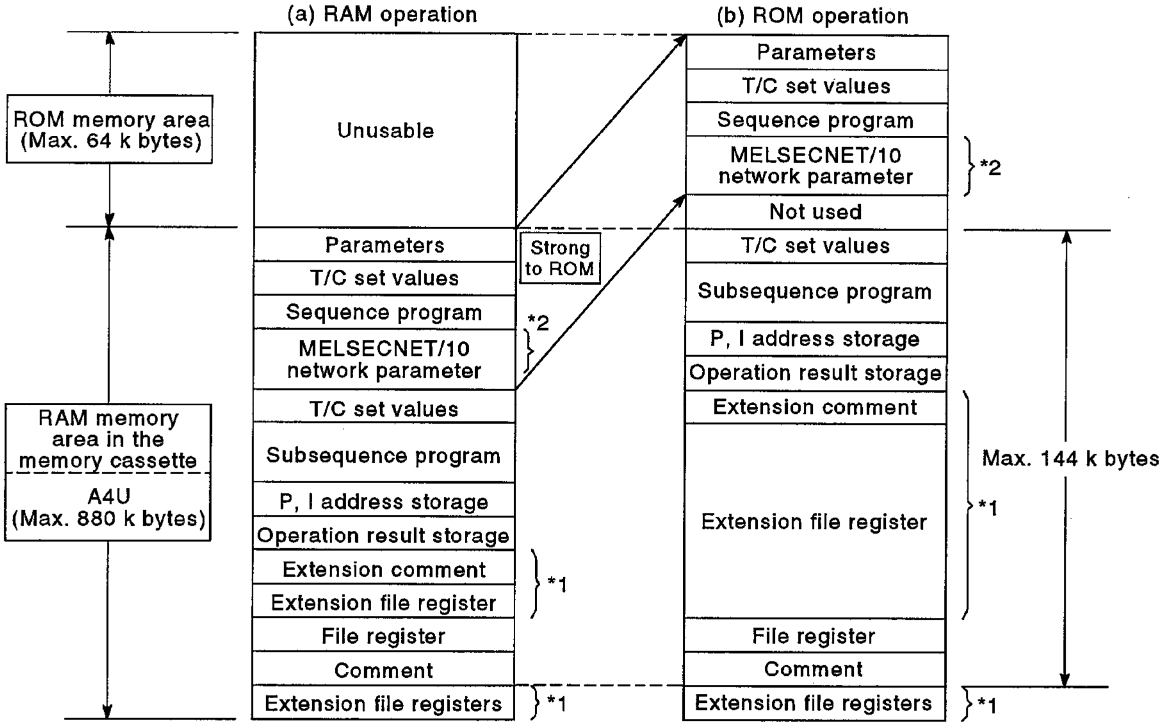


Fig. 9.24 Configuration of the User Memory Area

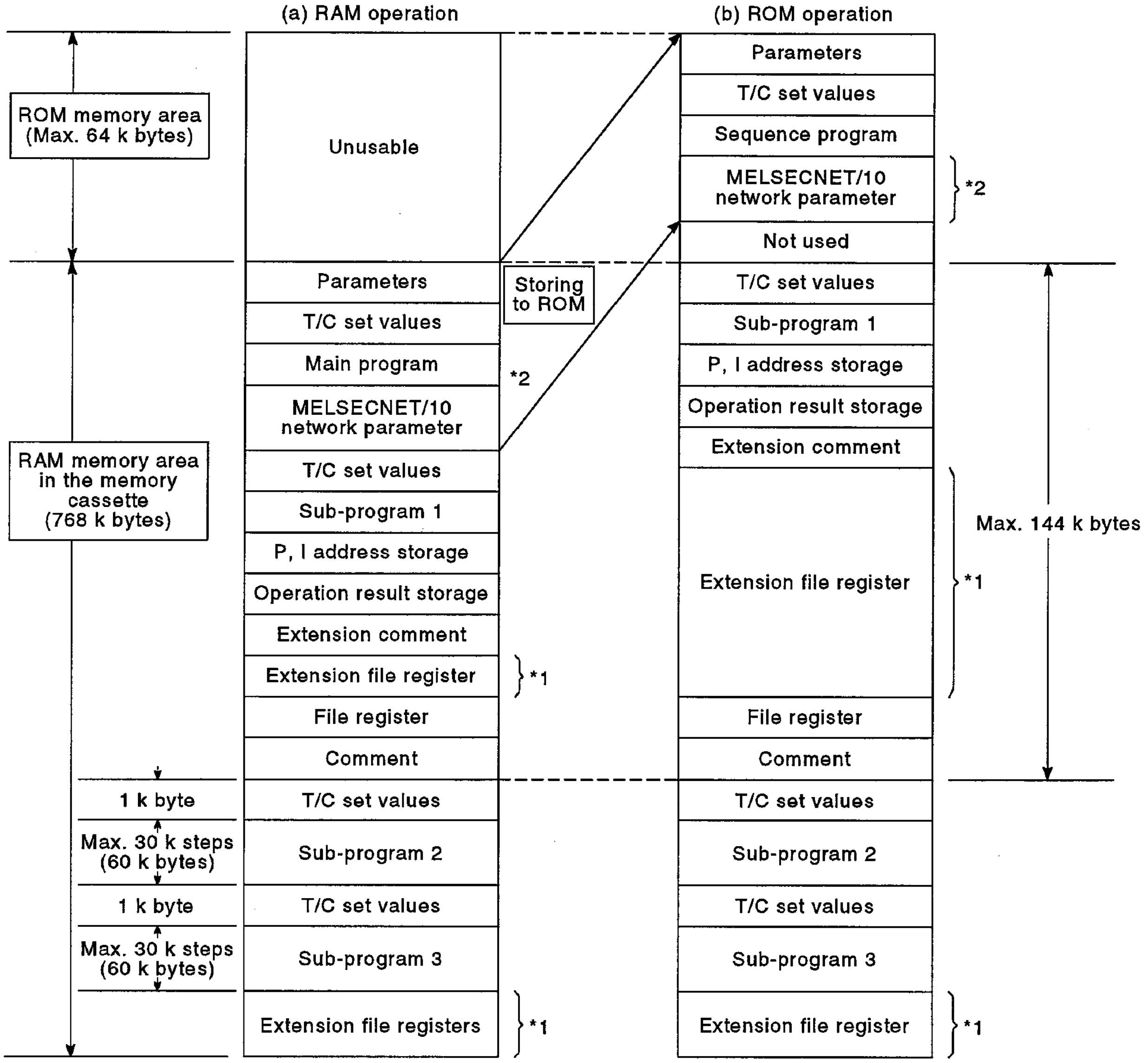
(2) When parameter setting is done (When sub-programs 2 and 3 are not used)
A maximum of 880 k bytes of the user memory area can be used by allocating it as shown in Fig. 9.25.



*1 : Refer to Section 3.11.2 for the details of extension file registers.
*2 : Refer to Chapter 11 for the allocation of the MELSECNET/10 network parameters.
When MELSECNET(II) data link system is constructed using the GPP function software package which is compatible to AnU, 2 k bytes (equivalent to 1 k step) are occupied for link paramaeter area.

Fig. 9.25 Configuration of the User Memory Area

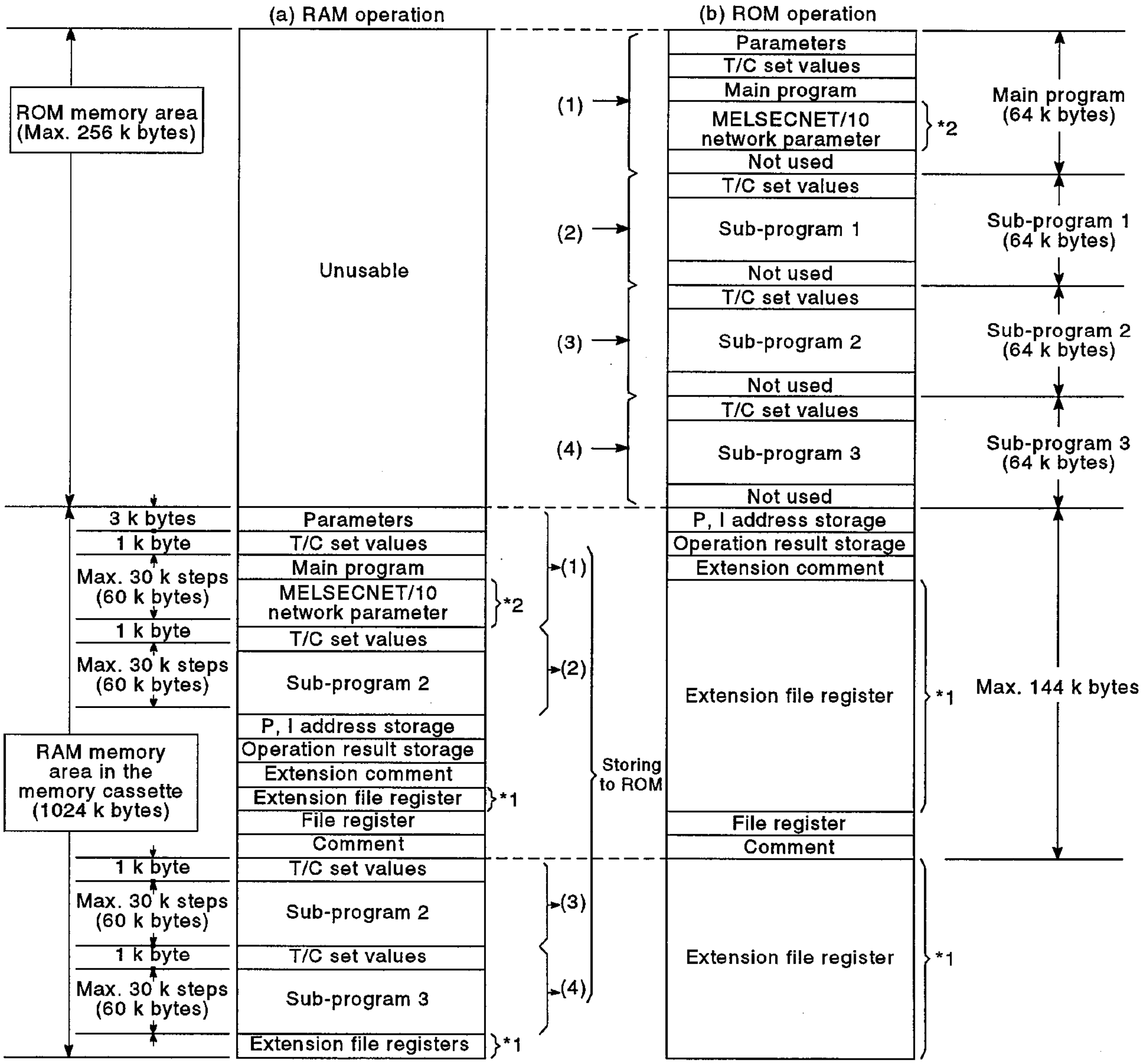
(3) When parameter setting is done (When sub-programs 2 and 3 are set with an A3AMCA-96 memory cassette)
When parameter setting is done with an A3AMCA-96, a maximum of 624 k bytes of the user memory area can be used by allocating it as shown in Fig. 9.26.
Sub-programs 2 and 3 are allocated beginning with the last block of the extension file registers.



*1 : Refer to Section 3.11.2 for the details of extension file registers.
*2 : Refer to Chapter 11 for the allocation of the MELSECNET/10 network parameters.
When MELSECNET(II) data link system is constructed using the GPP function software package which is compatible to AnU, 2 k bytes (equivalent to 1 k step) are occupied for link paramaeter area.

Fig. 9.26 Configuration of the User Memory Area

(4) When parameter setting is done (When sub-programs 2 and 3 are set with an A4UMCA-128 memory cassette)
When parameter setting is done with an A4UMCA-128, a maximum of 880 k bytes of the user memory area can be used by allocating it as shown in Fig. 9.27.
Sub-programs 2 and 3 are allocated by a RAM operation beginning with the last block (block No. 64) of the extension file registers.
To perform a ROM operation, the main programs and sub-programs (1 to 3) whose capacity has been set need to be stored in the ROM. (EPROM: 256 k ROM is used)



*1 : Refer to Section 3.11.2 for the details of extension file registers.
*2 : Refer to Chapter 11 for the allocation of the MELSECNET/10 network parameters.
When MELSECNET(II) data link system is constructed using the GPP function software package which is compatible to AnU, 2 k bytes (equivalent to 1 k step) are occupied for link paramaeter area.

Fig. 9.27 Configuration of the User Memory Area

9.13 Q02CPU-A, Q02HCPU-A and Q06HCPU-A

The Q02CPU-A, Q02HCPU-A and Q06HCPU-A have user memory area of 144 k bytes respectively.

(1) When parameter setting is not done
Configuration of the user memory area when parameter setting is not done is as shown in Fig. 9.28.

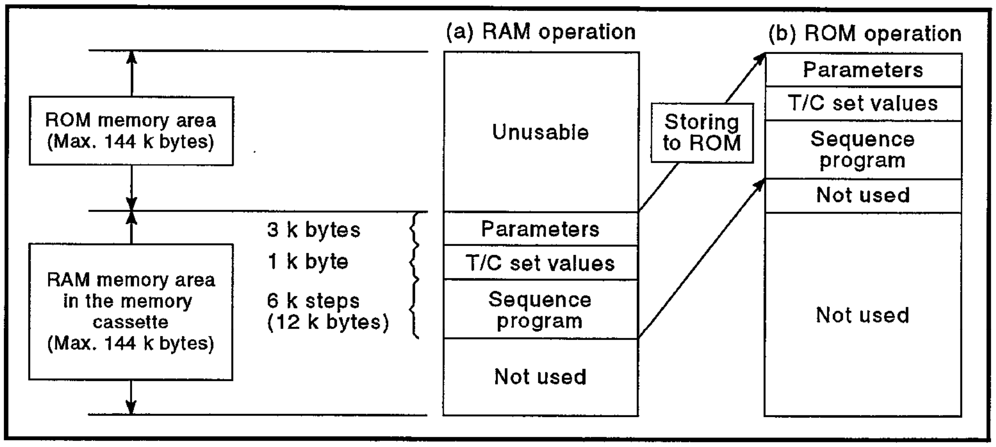
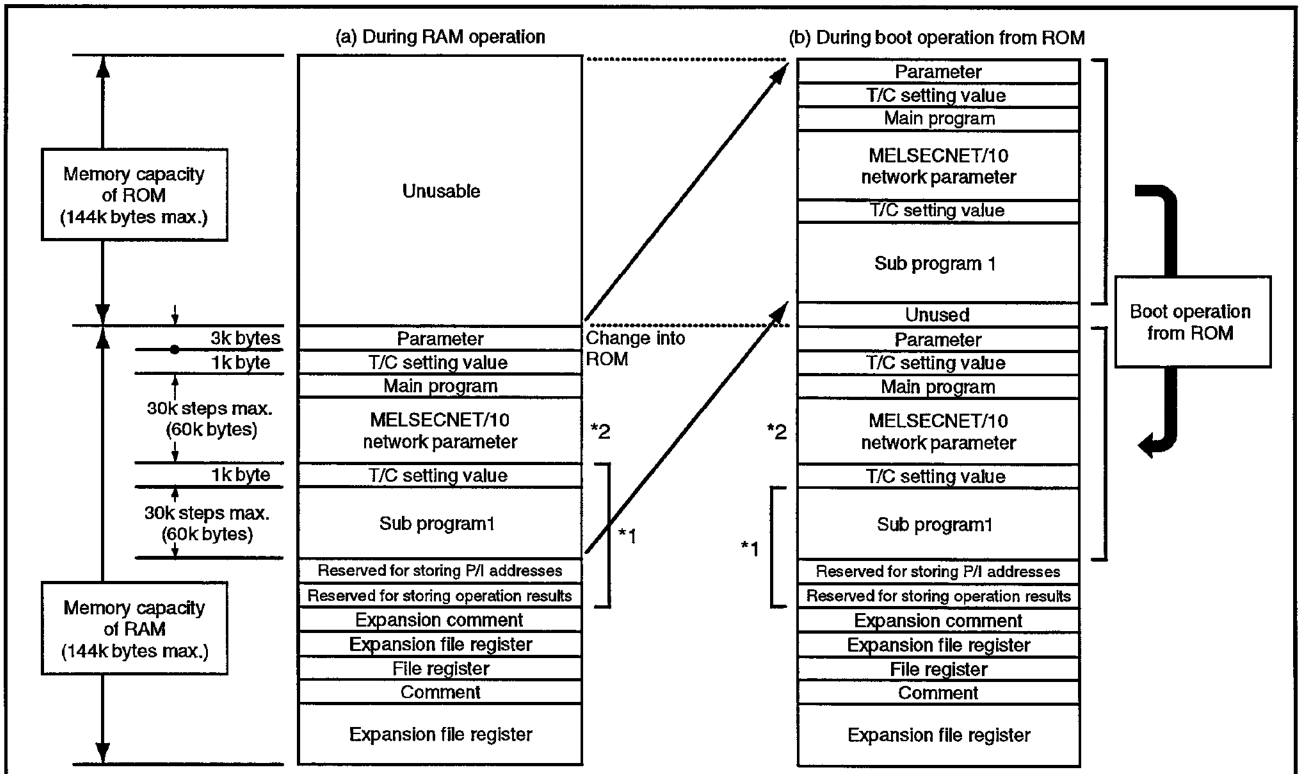


Fig. 9.28 Configuration of the User Memory Area

(2) When parameter setting is done
By changing the parameter setting, it is possible to use the user memory area (144 k bytes) by allocating it as shown in Fig. 9.29.



*1 : These areas can be set only with the Q06HCPU-A.
*2 : Refer to Section 11 for allocation of the MELSECNET/10 network parameters.
When MELSECNET(II) data link system is constructed using the GPP function software package which is compatible to AnU, 2 k bytes (equivalent to 1 k step) are occupied for link parameter area.

Fig. 9.29 Configuration of the User Memory Area

10. CONFIGURATION OF USER MEMORY AREA (WHEN AN SFC PROGRAM IS USED)

When an SFC program is used, an SFC program memory area is added to the user memory area mentioned in Chapter 9.
Memory areas other than the SFC program area conform to the memory area configuration mentioned in Chapter 9.

10.1 SFC Program Memory Area

The SFC program memory area is classified into SFC program area, SFC program work area, and step trace area.
Refer to the MELSAP-II Programming Manual (IB-66361) for the details of the SFC program memory area.

(1) SFC program area
The SFC programs are stored in the microcomputer program area.
When setting parameters, set the total capacity of the sequence program area and microcomputer program area (SFC program + utility program + user-created microcomputer program) to a value smaller than the main program capacity of the PC CPU to be used.

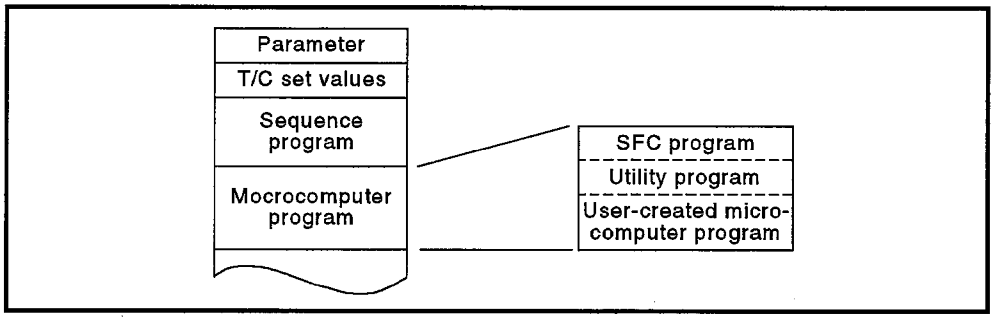


Fig. 10.1 Allocation of the SFC Program

(2) SFC program work area and step trace area *

(a) The SFC program work area is used by the OS to execute an SFC program, and 4 k bytes are required in the user memory area.
(The SFC program work area is not accessible by the user.)
When the SFC program work area cannot occupy 4 k bytes, the SFC program execution is disabled.

(b) A step trace area can be set in the 0 to 12 k byte range and is used for executing step tracing with peripheral devices.

POINT

* : Allocation of the SFC program work area and step trace area is determined by the CPU type and memory cassette type to used.
Refer to Sections 10.2 to 10.4 for details.

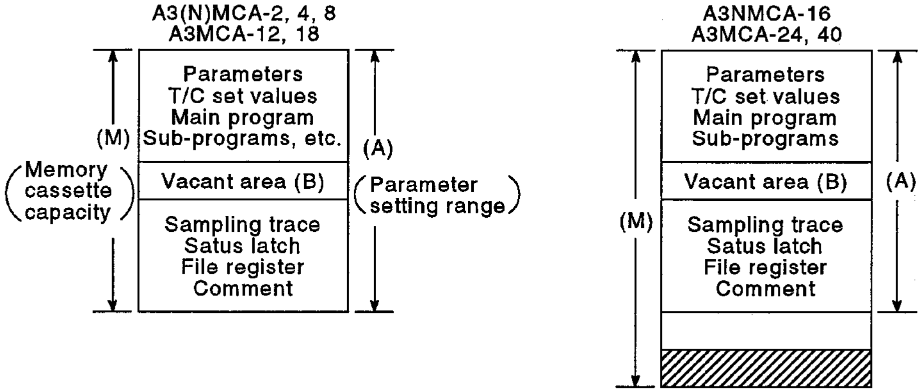
10.2 A2NCPU(S1) and A3NCPU

When the A2N(S1) and A3N are used, the feasibility of SFC program execution depends on the vacant capacity in the memory cassette used.
The following gives the feasibility of SFC program execution depending on the vacant capacity in the memory cassette used and the allocation of the SFC program work area and step trace area.

POINTS

(1) * : The feasibility of SFC program execution when the A2N(S1) or A3N is used is determined by the version of the CPU module. For the details of the CPU versions which are compatible with the SFC program, refer to the MELSAP-II Programming Manual (IB-66361) (Version: B and later).

(2) Refer to Section 10.1 for the allocation of the SFC program memory area.



Memory Cassette type	Memory Cassette Capacity (M)	Parameter Setting Range (A)	Vacant Area Capacity (B)	SFC Program Execution Feasibility	Available Stap Trace Area Capacity
A3(N)MCA-2 A3(N)MCA-4 A3(N)MCA-8 A3MCA-12 A3MCA-18	16 k bytes 32 k bytes 64 k bytes 96 k bytes 144 k bytes	16 k bytes 32 k bytes 64 k bytes 96 k bytes 144 k bytes	16 k bytes or more *1	Excution possible	12 k bytes
			4 to15 k bytes		[(4 to 15) -4] k bytes
			3 k bytes or less	Excution impossible	————
A3NMCA-16 A3NMCA-24 A3NMCA-40	128 k bytes 192 k bytes 320 k bytes	96 k bytes 144 k bytes 144 k bytes	16 k bytes or more *1	Excution possible	12 k bytes
			4 to15 k bytes		[(4 to 15) -4] k bytes
			3 k bytes or less		12 k bytes* (▨ area is used)

*1 : When the extension file registers are used by using the SW0GHP-UTLP-FN1 utility program, extension file register No.1 cannot be used.
*2 : When the extension file registers are used by using the SW0GHP-UTLP-FN1 utility program, extension file register No. 10 (▨ area) cannot be used.

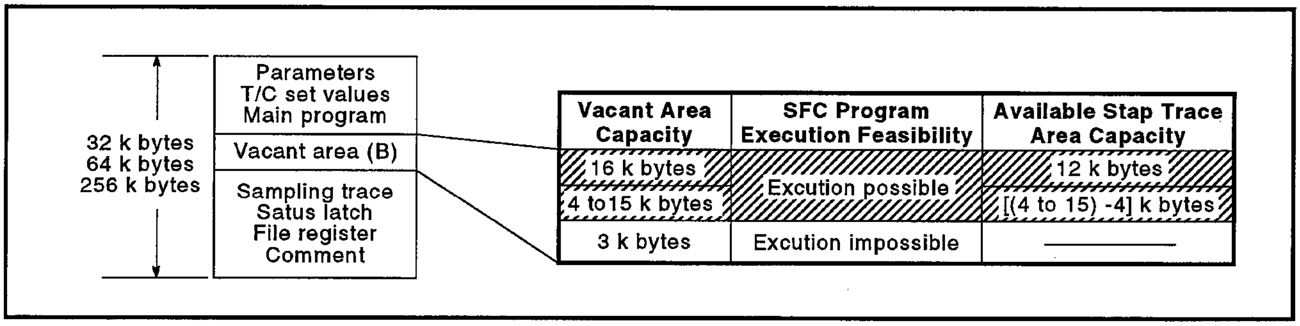
Fig. 10.2 Allocation of the SFC Program Memory Area

10.3 Compact Type CPU

The SFC program work area and step trace area are allocated in the following user memory area.

Type CPU	User Memory Area
A0J2H, A2C, A52G, A1S, A1S-S1, A1SJ-S3	32 k bytes
A1SH, A1SJH, A2SH, A1FX, A2S, A2AS	64 k bytes
A2S-S1, A2SH-S1	192 k bytes
A2AS-S1, A2AS-S30, A2AS-S60	256k bytes

The following gives the feasibility of SFC program execution depending on the vacant capacity in the user memory area and the allocation of the SFC program work area and step trace area.



Vacant Area Capacity	SFC Program Execution Feasibility	Available Stap Trace Area Capacity
16 k bytes	Excution possible	12 k bytes
4 to15 k bytes		[(4 to 15) -4] k bytes
3 k bytes	Excution impossible	————

Fig 10.3 SFC Program Memory Area

10.4 AnACPU* and AnUCPU

When the AnA, or AnU is used, the feasibility of SFC program execution depends on the vacant capacity in the memory cassette used and setting of extension comment capacity.

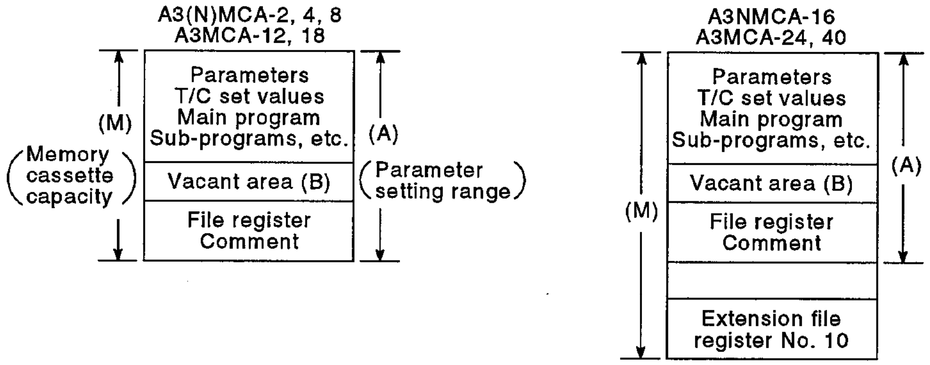
POINTS

(1) * : The feasibility of SFC program execution when the AnA is used is determined by the version of the CPU module.
For the details of the CPU versions which are compatible with the SFC program, refer to the MELSAP-II Programming Manual (IB-66361) (Version: B and later).

(2) Refer to Section 10.1 for the allocation of the SFC program memory area.

10.4.1 Allocation when extension comment capacity is not set

The feasibility of SFC program execution depends on the vacant capacity in the memory cassette used.
The following gives the feasibility of SFC program execution depending on the vacant capacity in the memory cassette used and the allocation of the SFC program work area and step trace area.



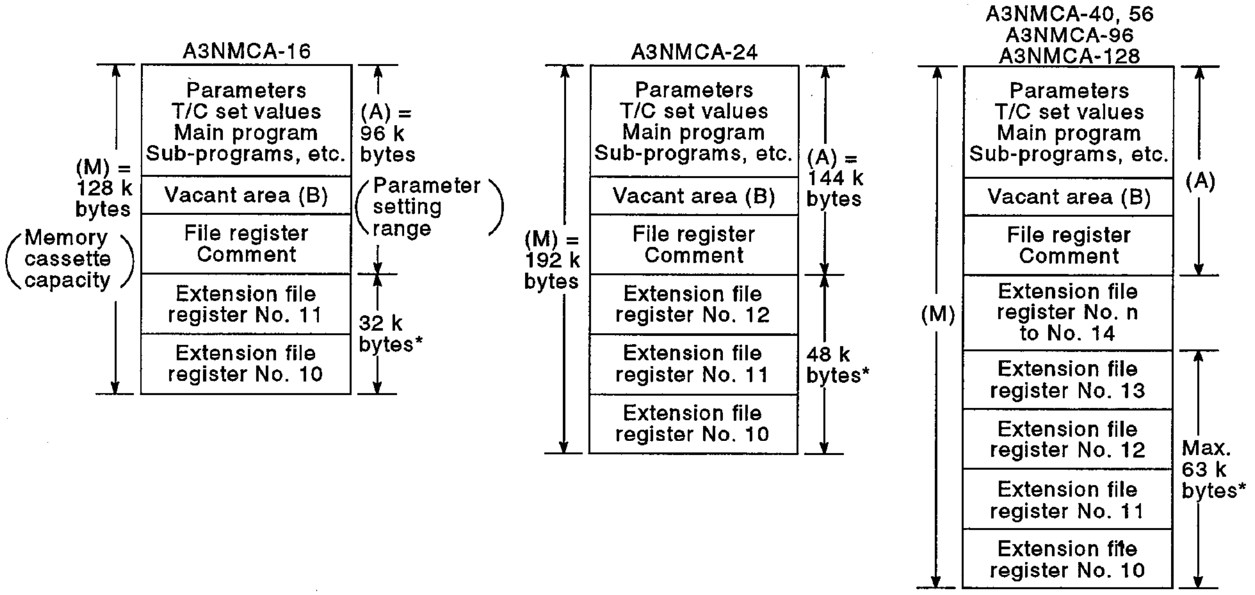
Memory Cassette Type	Memory Cassette Capacity (M)	Parameter Setting Range (A)	Vacant Area Capacity (B)	SFC Program Execution Feasibility	Available Stap Trace Area Capacity
A3(N)MCA-2 A3(N)MCA-4 A3(N)MCA-8 A3MCA-12	16 k bytes 32 k bytes 64 k bytes 96 k bytes	16 k bytes 32 k bytes 64 k bytes 96 k bytes	16 k bytes or more	Excution possible	0 to 12 k bytes (Extension file register No. 1 unavailable)
			4 to15 k bytes		0 to [(4 to 15) -4] k bytes
			3 k bytes or less	Excution impossible	————
A3NMCA-16	128 k bytes	96 k bytes	16 k bytes or more	Excution possible	0 to 12 k bytes (Extension file register No. 1 unavailable)
			15 k bytes or less		0 to 12 k bytes (Extension file register No. 10 unavailable
A3NMCA-18	144 k bytes	144 k bytes	32 k bytes or more	Excution possible	0 to 12 k bytes (Extension file register No. 2 unavailable)
			16 to 31 k bytes	Excution impossible	————
			4 to 15 k bytes	Excution possible	0 to [(4 to 15) -4] k bytes
			3 k bytes or less	Excution impossible	————
A3NMCA-24 A3NMCA-40 A3NMCA-56	192 k bytes 320 k bytes 448 k bytes	144 k bytes	32 k bytes or more	Excution possible	0 to 12 k bytes (Extension file register No. 2 unavailable)
			16 to 31 k bytes	Excution impossible	————
			15 k bytes or less	Excution possible	0 to 12 k bytes (Extension file register No. 10 unavailable)

Fig. 10.4 Allocation of the SFC Program Memory Area

10.4.2 Allocation when extension comment capacity is set

Allocation when the parameter setting capacity contains vacant area capacity when the extension comment capacity is set conforms to the allocation when the extension comment capacity is not set. (Refer to Section 10.4.1.) When the capacity after setting the extension comment capacity exceeds the parameter setting capacity, extension comments will be stored in extension file register Nos. 10 and after when the following memory cassettes are used.
The feasibility of SFC program execution and the allocation of the SFC program work area and step trace area depend on the vacant area capacity in the memory capacity less the extension comment capacity.

• A3NMCA-16 • A3NMCA-24 • A3NMCA-40
• A3NMCA-56 • A3AMCA-96[*1] • A4UMCA-128[*2]
*1: Compatible with A3ACPU and AnUCPU only.
*2: Compatible with A4UCPU only.



Memory Cassette Type	Memory Cassette Capacity (M)	Parameter Setting Range (A)	Vacant Area Capacity Less Extension Comment Capacity	SFC Program Execution Feasibility	Available Stap Trace Area Capacity
A3NMCA-16	128 k bytes	96 k bytes	16 k bytes or more	Excution possible	0 to 12 k bytes (Extension file register No. 1 unavailable)
			4 to 15 k bytes		
			3 k bytes or less	Excution impossible	————
A3NMCA-24 A3NMCA-40 A3NMCA-56 A3AMCA-96 A3UMCA-128	192 k bytes 320 k bytes 448 k bytes 768 k bytes 1024 k bytes	144 k bytes	32 k bytes or more	Excution possible	0 to 12 k bytes (Extension file register No. 2 unavailable)
			16 to 31 k bytes	Excution impossible	————
			4 to 15 k bytes	Excution possible	0 to [(4 to 15) -4] k bytes
			3 k bytes or less	Excution impossible	————

* : Extension comments are stored in extension file register Nos. 10 and the following areas according to set capacity.
Examples:
When the extension comment capacity is 20 k bytes, extension comments are stored in extension file register Nos. 10 and 11.
Areas used for the extension comments cannot be used as extension file registers.

Fig. 10.5 Allocation of the SFC Program Area

11. ALLOCATION OF THE MELSECNET/10 NETWORK PARAMETERS

To use a MELSECNET/10 data link system with an AnUCPU, A2ASCPU and QCPU-A, it is necessary to set the MELSECNET/10 network parameters.

(1) MELSECNET/10 network parameter capacity
The MELSECNET/10 network parameters need to be set with each MELSECNET/10 link module.
The set network parameters are stored in a range of 64 k bytes from the head of the memory area of the memory cassette.
Because of this, when MELSECNET/10 network parameters are set with an A3U or A4U, the main program setting range is reduced by 30 k steps.

$$\text{Memory capacity 64 k bytes} \geq \left(\begin{array}{l} \text{Parameter area (3 k bytes)} \\ \quad + \\ \text{T/C setting area (1 k byte)} \\ \quad + \\ \text{Main program area (2 to 60 k bytes)} \\ \quad + \\ \text{Network parameters (2 to 16 k bytes)} \end{array} \right)$$

(2) The capacity used for network parameters varies depending on the settings made.

Network parameter setting capacity
o : Must be set △ : Set if necessary x : Setting not necessary

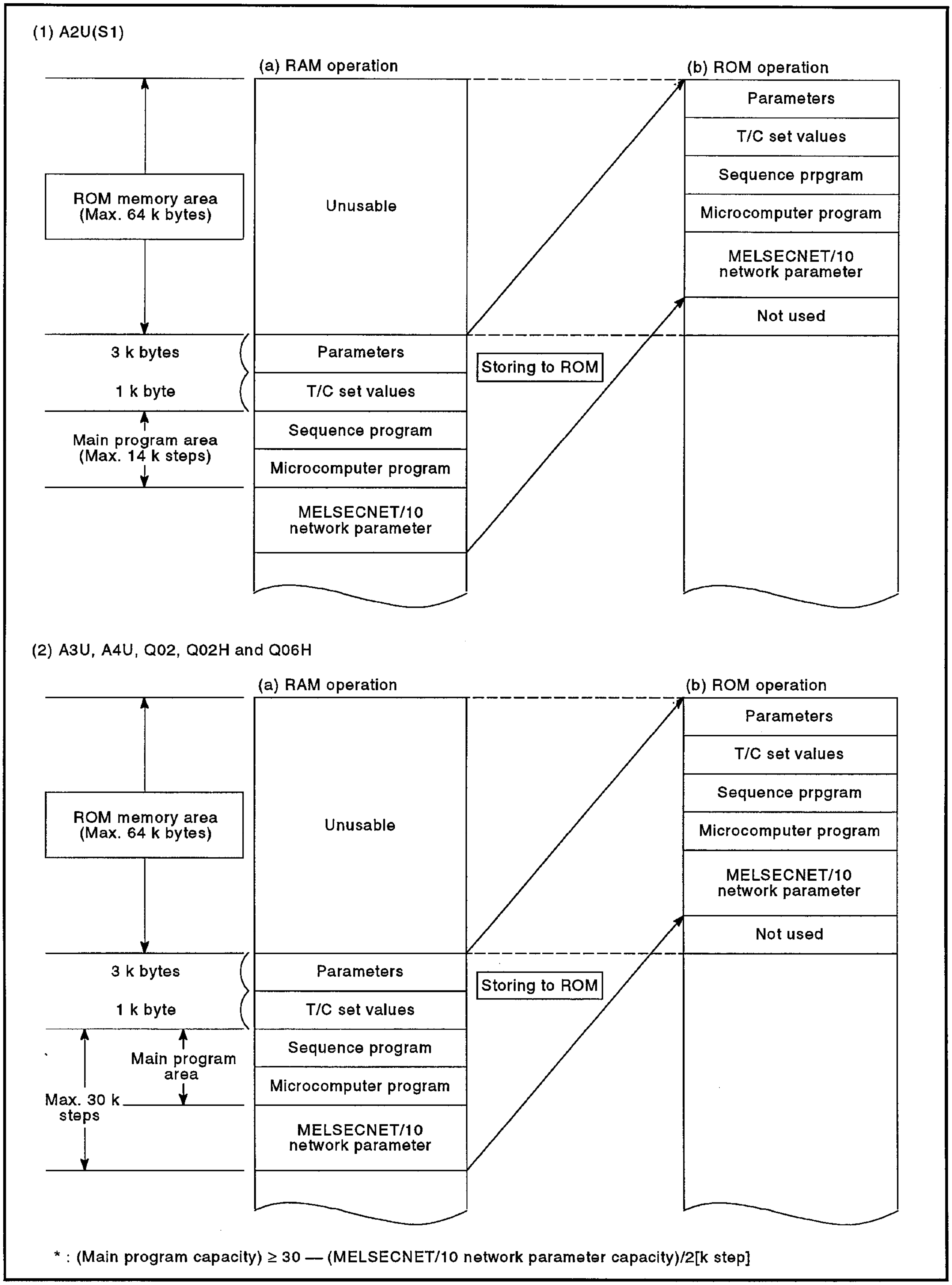
Item	Control Station	Normal Station
Internal data (30 bytes)	o	o
Routing parameters (390 bytes)	△	△
Transfer parameters for data link (246 bytes)	△	△
Common parameters (2164 bytes/module) *1	o	x
Refresh parameters (92 bytes/module) *2	o	o
Station-specific parameters (1490 bytes/module)	△	△

*1 For a remote master station, the common parameters occupy 2722 bytes.
*2 If for example the number of network modules mounted is two, the refresh parameters occupy 92 x 2 = 184 bytes.

For the network parameter capacity, secure an area on the basis of the total of each of the settings in 2 k byte units.

Total Capacity of Each Setting	Network Parameter Setting Capacity
30 to 2048 bytes	2 k bytes
2049 to 4096 bytes	4 k bytes
4097 to 6144 bytes	6 k bytes
6145 to 8192 bytes	8 k bytes
8193 to 10240 bytes	10 k bytes
10241 to 12288 bytes	12 k bytes
12289 to 14336 bytes	14 k bytes
14337 to 16384 bytes	16 k bytes

(3) Range to be stored to the ROM for the ROM operation
When the memory contents are stored to a ROM, the MELSECNET/10 netowork parameters are also stored the ROM.
Use an EPROM which can store parameters, T/C set values, main programs, and MELSECNET/10 network parameters.



* : (Main program capacity) ≥ 30 — (MELSECNET/10 network parameter capacity)/2[k step]

Fig 11.1 Allocation of the MELSECNET/10 Link Parameters

APPENDIX

APPENDIX 1 Differences between A3NCPU and A3N board

Refer to	Item	A3NCPU	A3N Board
2.4.2	Refresh method	o	x
3.7.2	Count processing during refreshing	o	△ *1
4.2.2	I/O allocation using peripheral devices	o	x
6.10	Changing I/O modules in online	o	x
6.17	Setting the display mode for annunciator	o	x

*1 Applied only for A7LMS-F

Error Message	Description	Course of Action
CIRCUIT CONTINUATION ERROR.	The return ladder is not correct.	Ladder the return.
COMMAND ERROR.	Method of specification is incorrect.	Look at HELP Information and specify correctly.
	There is an incorrect code in the sequence program.	Delete the incorrect step using list mode.
COMMAND ERROR. ERROR STEP=%S.	When writing a parameter + main, main program or sub-program in PC mode, the step indicated by "Error Step\XXX" has an incorrect code.	Using step mode, check the following, then edit or delete the error step. Did you use a "W" command with a CPU that has no link card in the A0J or in the X,Y device range? Otherwise, the program is irregular.
COMMENT CAPACITY NOT SET.	The comment capacity parameter has not been set.	Set the comment capacity with Parameter Setup.
DATA NOT FOUND.	When reading out file maintenance data, the data file is smaller than the parameter capacity.	Set the parameter correctly.
	When recording items for the Ladder Mode Monitor, you selected Device Memory or Status Latch, but the required data is not in the AT's memory.	Read out the required data in PC mode or in File Maintenance.
DATA OVERFLOWS ONE SCREEN. PRESS [ENTER].	The SFC Device Search found more devices than it can display on one screen.	Press [ENTER] to see the next screen.
DEVICE NUMBER OUT OF RANGE.	The number specified for the device is out of range.	Set the device number to an in-range value.
DRIVE NOT READY.	When writing to the floppy disk, there is no disk inserted in the destination drive.	Insert a disk in the destination drive.
EXECUTABLE DURING BLOCK STOP ONLY.	You selected the test functions Restart or without first stopping the block.	Execute Continue or One Cycle Execute only when the block is stopped.
FILE NOT FOUND.	Your search selection has no file.	Change your selection.
FLOPPY DISK ERROR.	You tried to access a drive that has no disk inserted.	Insert a floppy disk or change the drive number.
	You tried to write a write-protected disk.	Switch the disk's write protect tab to write-enable.
	Disk is unusable.	Change the disk.
INCORRECT BRANCH/ CONNECTION.	A branch or converge on the SFC chart does not follow the language format values.	Correct the branch or converge in accordance with the language format rules.
INCORRECT DATA.	A block containing no set steps has been recorded for Step Trace.	Set a base step for the block.
INCORRECT DATA NAME.	The sampling trace, status latch or device memory data name has been incorrectly specified.	Enter a correct data name. You may use alphanumeric characters and "-"," _". (Up to 8 characters.) You must begin the name with an alphabetic character (A-Z).

Error Message	Description	Course of Action
INCORRECT EXECUTION POSITION.	You selected Zoom while not at a step or transition condition.	Move the cursor to a step or transition consition before selecting Zoom.
	When deleting a branch or converge on the SFC Chart, the chart symbol specified by the cursor is different from the function key diagram symbol.	Press a function key with the same diagram symbol as the one specified by the cursor.
INCORRECT JUMP DESTINATION.	You specified a Jump out of or into a branched area.	Please follow SFC language format.
INCORRECT OPERATION.	The order of your settings is incorrect.	Input settings in correct order.
INCORRECT SYSTEM NAME.	When performing an intra-disk file copy (source and destination are the same floppy disk), you specified the same system name for source and destination.	Change the destination system name.
INCORRECT TERMINATION.	The bottom edge of the SFC diagram is not a Jump or End step.	Enter a Jump or End step on the bottom edge.
INVALID KEY.	You pressed an improper key.	Follow the explanations on the screen, consult HELP or read the manual and press a proper key.
JUMP DESTINATION NOT FOUND.	When writing the SFC diagram, you created a Jump node with a non-existent destination step.	Write the Jump destination step before writing in the Jump transition.
LADDER NOT ENTERED.	You selected an unsaved ladder for program copy.	Select a saved ladder.
LADDER OVERFLOW.	The ladder has more than 8 ANB or ORB commands in a row, or more than 9 LD commands in a row.	Edit the ladder to correct the error condition.
MICROCOMPUTER PROGRAM CAPACITY NOT SET.	You tried to create a program in SFC mode without setting the microcomputer capacity.	Set the microcomputer capacity with Parameter setting.
NOT BROKEN.	You tried to perform the test function Restart or Cancel Break when there was no break set.	Set a break point or blockbreak.
NO 'END' COMMAND.	A transition condition or output directive program created with the List language has no End statement.	Add an End statement to the end of the transition condition or output directive program.
NOT ENOUGH MEMORY.	The remaining memory capacity on the floppy disk is too small to perform the requested write operation.	Insert a new disk and re-execute the write operation. If there is unneeded data on the disk, delete it to free up space.
NOT USABLE FOR SFC-STEP.	You entered a command or label that is not alowed in output directives (commands MC, MCR, CJ, SCJ, JMP, FEND, RET, IRET, CHK, CHG or labels P,I).	Enter an acceptable command for an output directive.
NUMBER NOT ENTERED.	You selected an entry number that does not contain an entered program for Macro Recall.	Select an entry number that contains an entered program.
OR LADDER EXCEEDS 24 LINES AND OVERFLOWS DISPLAY AREA.	You tried to display a ladder block that is more than 25 lines long.	Verify via List Display. When you verify via Ladder Display, the block is truncated to 24 lines.
OUT OF SETTING RANGE.	Capacity setting is out of range. Specified step number is out of range.	Set capacity in-range. Specify step number in-range.
PARAMETER ERROR.	You tried to record a Sampling Trace, but the PC Sampling Trace capacity has not been set.	Set PC Sampling Trace to "Y" with Parameter Setting.

Error Message	Description	Course of Action
PARAMETER SETTING MISMATCH.	When trying to read or write in PC mode, settings in parameter memory do not correspond.	Correct the appropriate parameters.
	During parameter verification, the contents do not correspond.	
	You tried to display Sampling Trace without executing Sampling Trace from Parameters.	Select "Y" for Sampling Trace from Parameters.
PC CAPACITY NOT SET.	You executed a write in PC mode, but the PC capacity parameter has not been set.	Set the PC capacity with Parameter Setting.
PC COMMUNICATION ERROR.	PC power is off.	Turn the PC power on.
	Cable or connection is bad.	Re-connect carefully with the correct cable type.
	The PC was reset during transmission.	Re-start the transmission from the AT.
	The RUN LED began to flash during transmission.	Correct the cause of the flashing, reset the PC, and re-start the transmission from the AT.
	The AT issued a Remote Run when the PC was obeying a Remote Stop issued by the computer link unit.	Cancel the Remote Stop issued by the computer link unit and issue a Remote Run from the AT.
PC SELECTION ERROR.	The PC type specified in the initial data and the PC actually connected are different.	Re-display the initialization setting screen and enter the correct PC type.
PC TYPE MISMATCH.	The PC type specified by parameters does not match the PC actually connected.	Re-set the parameter to the connected PC type.
PRINTER NOT READY.	You tried to execute a printout, but the printer is not connected.	Connect the printer.
PROGRAM NOT FOUND.	Your search selection has no program.	Change your selection.
RANGE SETTING ERROR.	The specified range is outside of the acceptable range.	Specify a proper range.
SELECT "DETAILED DISPLAY".	You tried to write a diagram while it was in skeleton diagram display mode.	Set the display to "detailed" mode.
SELECT "DISPLAY WITH COMMENT".	You selected Create Comments when the display was not in "include comments" mode.	Change the display to include comment 1 or comment 2.
SELECT "DISPLAY WITHOUT COMMENT".	You tried to write, Insert or Delete to a circuit in Zoom mode when the display was in "With comments" mode.	Change the display to "Without comments" mode.
SET PC TO STEP-RUN.	You selected a test function (F1 to F7) when the PC was not in Step Run mode.	Set the PC to Step Run mode and re-select the test function.
SFC DIAGRAM ERROR.	There is a ladder entry that does not follow SFC grammar.	Re-enter following the SFC grammar.
STEP NUMBER ALREADY EXISTS.	In statement mode, you selected a step that already has a program.	Check the program and choose a correct step number.
SYSTEM NAME NOT FOUND.	The read origin system name does not exist on the specified drive.	View the directory to confirm the system name and re-enter.
THE PC CHANNEL AND PC NUMBER ARE INCORRECT.	The operation is prohibited for the currently set PC Channel and PC Number, or you tried to set the Remote I/O port while it was connected.	Consult HELP or the manual to avoid prohibited operations.

Error Message	Description	Course of Action
WRITE TO ROM IMPOSSIBLE.	You executed a write in PC mode,but the RAM/ROM switch in the PC memory cassette was set to ROM.	Set the RAM/ROM switch in the PC memory cassette to RAM.
WRITE-IN ERROR.	You executed "write" (download) in PC mode when the PC memory cassette RAM/ROM switch was set to ROM.	Change the memory cassette RAM/ROM switch to RAM.
	When writing a file, you exceeded the disk capacity.	Delete unnecessary files to increase free space on the disk.

Information Message	Description
COMPLETE. STEP NUMBER CHANGED.	When a step number has been changed.
COPYING TO THE RAM DISK.	You chose PC mode or ladder mode from the menu.
EXECUTING.	When a Read, Write or Verify (to/from the PC, floppy disk or ROM) is in progress, or when a circuit conversion is in progress.
LADDER END.	In ladder display, when the last part of the program has been read out.
MONITOR STARTED.	When monitoring is in progress (Monitor function).
MONITOR STOPPED.	In ladder monitor, a trigger stop has been encountered, or you pressed ESC.
OPERATION COMPLETE.	When a Read, Write or Verify (to/from the PC, floppy disk or ROM) has completed or when a circuit conversion is complete.
PLEASE SELECT PC TYPE.	When you select a PC type during system setting.
PRESS [CONV.] (SHIFT + F4) TO DELETE A LADDER BLOCK.	When a circuit block is deleted.
PROCESSING.	Internal management in progress.
READOUT PC MEMORY.	It is necessary to read from the PC.
SAMPLING STARTED.	In Sampling Trace mode, the PC has begun sampling.
SAMPLING STOPPED.	In Sampling Trace mode, you pressed ESC during sampling execution to abort sampling.
STOPPED.	A function has been aborted in progress by the user.
VERIFYING.	When a verify is in progress.
WRITE SETTING DATA TO PC.	It is necessary to write setup data to the PC.

WARRANTY

Please confirm the following product warranty details before starting use.

1. Gratis Warranty Term and Gratis Warranty Range

If any faults or defects (hereinafter "Failure") found to be the responsibility of Mitsubishi occurs during use of the product within the gratis warranty term, the product shall be repaired at no cost via the dealer or Mitsubishi Service Company. Note that if repairs are required at a site overseas, on a detached island or remote place, expenses to dispatch an engineer shall be charged for.

[Gratis Warranty Term]

The gratis warranty term of the product shall be for one year after the date of purchase or delivery to a designated place.

Note that after manufacture and shipment from Mitsubishi, the maximum distribution period shall be six (6) months, and the longest gratis warranty term after manufacturing shall be eighteen (18) months. The gratis warranty term of repair parts shall not exceed the gratis warranty term before repairs.

[Gratis Warranty Range]

(1) The range shall be limited to normal use within the usage state, usage methods and usage environment, etc., which follow the conditions and precautions, etc., given in the instruction manual, user's manual and caution labels on the product.

(2) Even within the gratis warranty term, repairs shall be charged for in the following cases.

1. Failure occurring from inappropriate storage or handling, carelessness or negligence by the user. Failure caused by the user's hardware or software design.
2. Failure caused by unapproved modifications, etc., to the product by the user.
3. When the Mitsubishi product is assembled into a user's device, Failure that could have been avoided if functions or structures, judged as necessary in the legal safety measures the user's device is subject to or as necessary by industry standards, had been provided.
4. Failure that could have been avoided if consumable parts (battery, backlight, fuse, etc.) designated in the instruction manual had been correctly serviced or replaced.
5. Failure caused by external irresistible forces such as fires or abnormal voltages, and Failure caused by force majeure such as earthquakes, lightning, wind and water damage.
6. Failure caused by reasons unpredictable by scientific technology standards at time of shipment from Mitsubishi.
7. Any other failure found not to be the responsibility of Mitsubishi or the user.

2. Onerous repair term after discontinuation of production

(1) Mitsubishi shall accept onerous product repairs for seven (7) years after production of the product is discontinued. Discontinuation of production shall be notified with Mitsubishi Technical Bulletins, etc.

(2) Product supply (including repair parts) is not possible after production is discontinued.

3. Overseas service

Overseas, repairs shall be accepted by Mitsubishi's local overseas FA Center. Note that the repair conditions at each FA Center may differ.

4. Exclusion of chance loss and secondary loss from warranty liability

Regardless of the gratis warranty term, Mitsubishi shall not be liable for compensation to damages caused by any cause found not to be the responsibility of Mitsubishi, chance losses, lost profits incurred to the user by Failures of Mitsubishi products, damages and secondary damages caused from special reasons regardless of Mitsubishi's expectations, compensation for accidents, and compensation for damages to products other than Mitsubishi products and other duties.

5. Changes in product specifications

The specifications given in the catalogs, manuals or technical documents are subject to change without prior notice.

6. Product application

(1) In using the Mitsubishi MELSEC programmable logic controller, the usage conditions shall be that the application will not lead to a major accident even if any problem or fault should occur in the programmable logic controller device, and that backup and fail-safe functions are systematically provided outside of the device for any problem or fault.

(2) The Mitsubishi general-purpose programmable logic controller has been designed and manufactured for applications in general industries, etc. Thus, applications in which the public could be affected such as in nuclear power plants and other power plants operated by respective power companies, and applications in which a special quality assurance system is required, such as for Railway companies or National Defense purposes shall be excluded from the programmable logic controller applications.

Note that even with these applications, if the user approves that the application is to be limited and a special quality is not required, application shall be possible.

When considering use in aircraft, medical applications, railways, incineration and fuel devices, manned transport devices, equipment for recreation and amusement, and safety devices, in which human life or assets could be greatly affected and for which a particularly high reliability is required in terms of safety and control system, please consult with Mitsubishi and discuss the required specifications.

type ACPU/QCPU-A (A Mode)(Fundamentals)

Programming Manual

MODEL	ACPU(FUNDA.)-P-E
MODEL CODE	13J740
IB(NA)-66249-N(0012)MEE	

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : MITSUBISHI DENKI BLDG MARUNOUCHI TOKYO 100-8310 TELEX : J24532 CABLE MELCO TOKYO
NAGOYA WORKS : 1-14 , YADA-MINAMI 5 , HIGASHI-KU, NAGOYA , JAPAN

When exported from Japan, this manual does not require application to the Ministry of International Trade and Industry for service transaction permission.

Specifications subject to change without notice.