MITSUBISHI

PROGRAMMABLE CONTROLLER

MELSEC-A

Programming Manual

type A81CPU

MITSUBISHI
ELECTRIC

REVISIONS

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

Print Date	Manual number	Revision
Sep., 1988	IB (NA) 66168-A	First edition

INTRODUCTION

Thank you for choosing the Mitsubishi MELSEC-A Series of General Purpose Programmable Controllers. Please read this manual carefully so that the equipment is used to its optimum. A copy of this manual should be forwarded to the end User.

CONTENTS

1. INTRODUCTION

This manual gives information on the performances, functions, instructions, etc. required for programming with the A81CPU PID control module.
The A81CPU is used for PID control applied to process control of flow rate, air flow, temperature, etc.
The PID control function are defined by the A81CPU parameters.

1.1 Features

The main features of the A81CPU are as follows:

(1) Optimum system can be configured.

The A81CPU is a building block type CPU module. Hence, an appropriate system can be configured by loading the required I/O and special function modules on the base unit in accordance with the control specifications.

(2) 32 programs

Up to 32 programs can be written in 32 program areas, each consisting of 250 steps.

(3) PID control of 64 loops

Max. 64 loops can be PID-controlled as the PID control parameter area is reserved for 64 loops. PID control data is defined by parameters.

(4) A variety of arithmetic operations

The A81CPU may be used as an arithmetic module because it has many arithmetic operation instructions, such as four operations, logarithm, square root and trigonometric function.

(5) Debugging by step run

Any program can be executed per instruction using the command from the GPP.

(6) Operation monitoring by the GPP

Devices PX, PY, PM (SP. PM), PT, PD (SP. PD), A can be monitored by the GPP.

(7) Control status monitoring by the AD57(S1)

PID control status can be monitored on a CRT or a plasma display by using the AD57(S1) CRT controller module.

(8) Clock function

Clock operation can be performed by the internal clock element in accordance with the specified clock data (year, month, day, hour, minute, second, day of the week).
Clock data can be read to the special registers.

(9) Use with the ACPU

The A81CPU may be used with the ACPU and the buffer memory and I/O of either CPU can be accessed by the other.

1.2 PID Control System

(1) PID control system

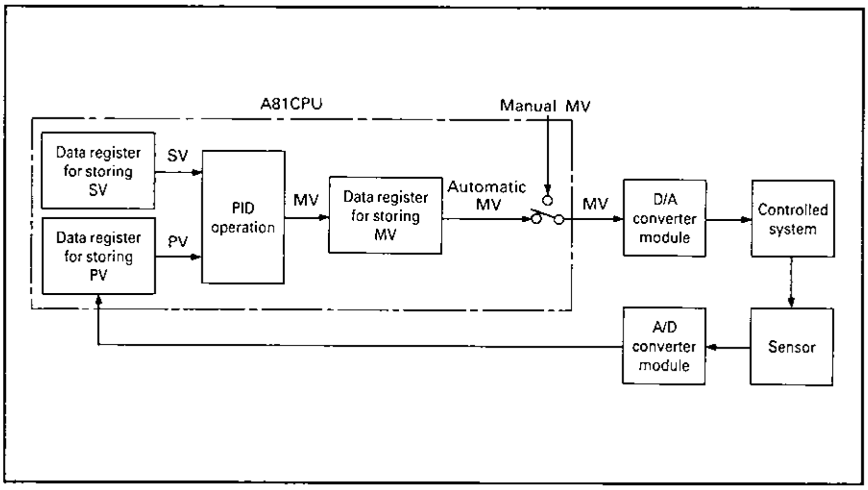


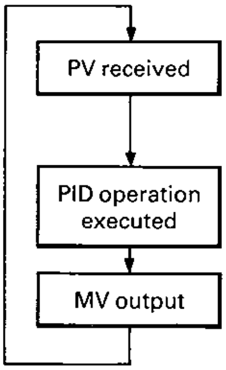
Fig. 1.1 PID Control System

REMARKS

SV, PV and MV in Fig. 1.1 indicate the following values:

- SV: Set value
- PV: Process value
- MV: Manipulated value

(2) PID control procedure



PV received ········ A sensor signal is received by the user program through the A/D converter module, converted into a % between −2.5 and 102.5, and stored in the PV data register.

PID operation executed ········ PID operation is executed using the SV, PV stored in the SV, PV data registers.

MV output ········ MV (−2.5 to 102.5%) obtained by the PID operation is converted into a value between −50 and 2050 by the user program. This value is converted into an analog value and output to the controlled system.

Fig. 1.2 PID Control Procedure

1.3 PID Operation

Allows two types of PID control; velocity type and process value derivative (incomplete derivative).

1.3.1 Operation method

(1) Velocity type operation

Calculates a manipulated value (MV) variation.
The actual MV is an accumulating total of MV variations calculated during sampling times.

(2) Process value derivative type operation

Performs calculations using a process value (PV) in the derivative term.
As the derivative term does not include an error, output is not changed suddenly by the derivative control action if the error varies due to the set value change.

1.3.2 Forward and reverse actions

(1) A forward or a reverse action is available for PID control.

(a) The reverse action decreases the PV to the SV when the MV increases.

(b) The forward action increases the PV to the SV when the MV decreases.

(2) Fig. 1.3 illustrates the forward and reverse acting processes by using the MV, PV and SV.

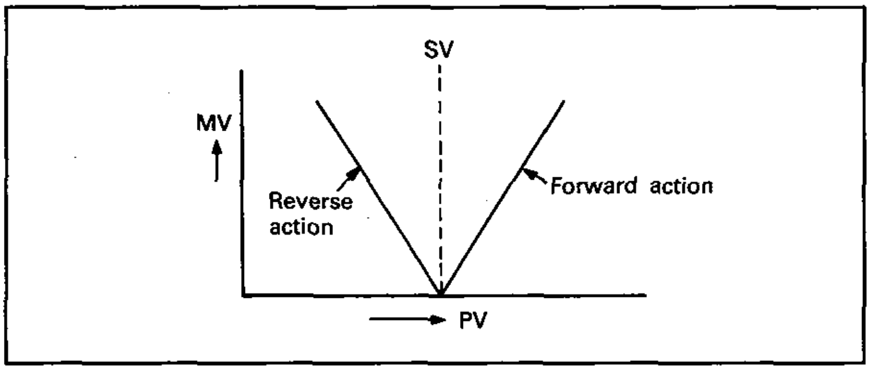


Fig. 1.3 Forward and Reverse Actions by MV, PV, SV

(3) Fig. 1.4 illustrates process control examples by the forward and reverse actions.

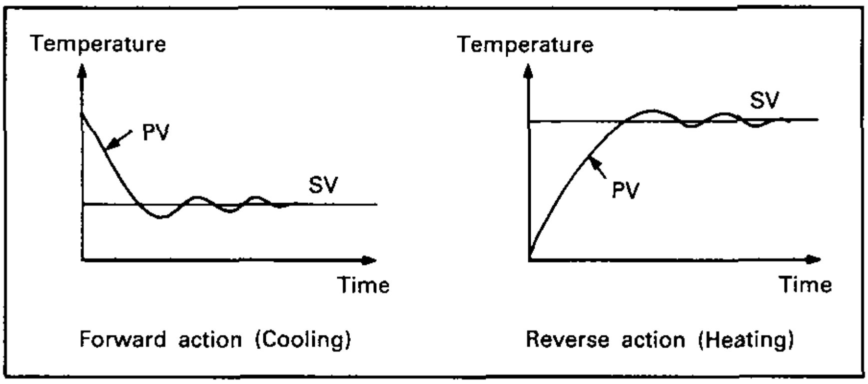


Fig. 1.4 Process Control Examples by Forward and Reverse Actions

1.3.3 Proportional control action (P control action)

(1) Provides a MV proportional to an error (difference between the set value(SV) and process value(PV)).

(2) The relation between the error (E) and MV is expressed as follows:

$$\boxed{MV = K_P \cdot E}$$

KP indicates a proportional constant which is referred to as proportional gain.

(3) Fig. 1.5 illustrates a proportional control action where the error is constant.

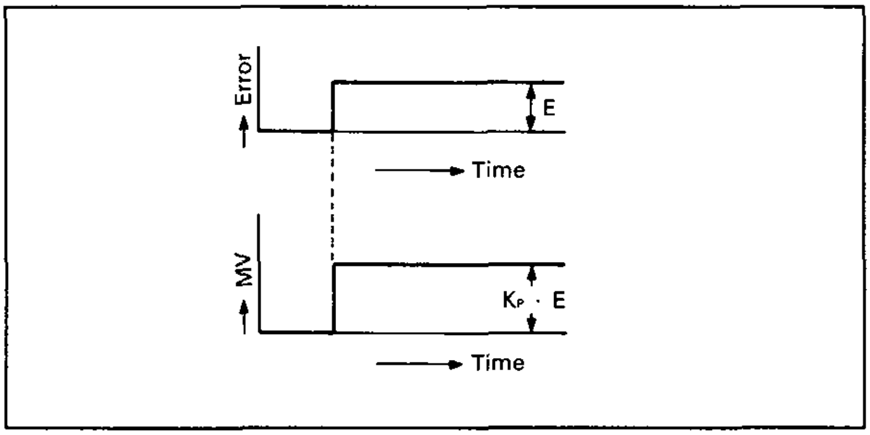


Fig. 1.5 Proportional Control Action Where Error Is Constant

(4) The MV varies between −2.50 and 102.50%.
The MV changes in proportion to KP for a given error.

(5) Disturbances in the system lead to offset errors.

1.3.4 Integral control action (I control action)

(1) Provides a steady change in MV for a given error, with the objective of eliminating the error.

(2) Integral time T_I indicates a period of time between an error occurring and the integral control action MV becoming equal to the proportional control action MV.
The smaller T_I is, the greater the integral control action works.

(3) Fig. 1.6 shows an integral control action where the error is constant.

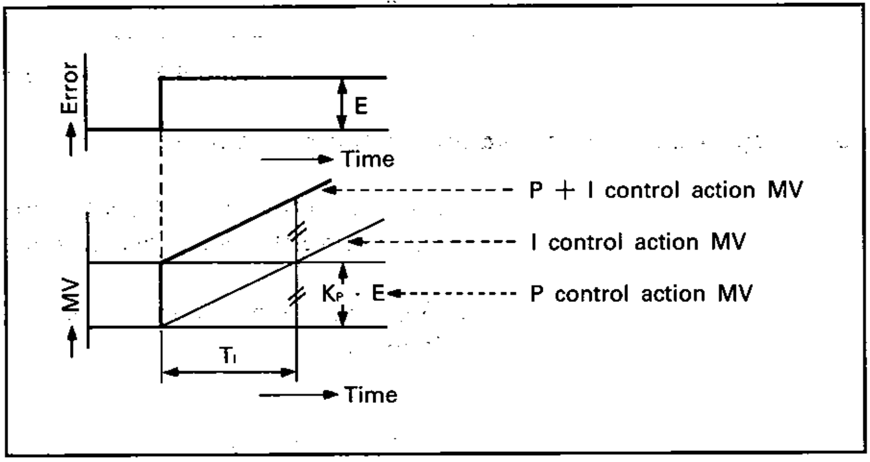


Fig. 1.6 Integral Control Action Where Error Is Constant

(4) The integral control action should be used with the proportional control action (as a PI control action) or with the proportional and derivative control actions (as a PID control action) and cannot be used independently.

1.3.5 Derivative control action (D control action)

(1) Stabilizes the system in response to rapid changes by adjusting the proportional bandwidth.

(2) Derivative time T_D indicates a period of time between an error occurring and the derivative control action MV becoming equal to the proportional control action MV.
The smaller T_D is, the greater the derivative control action works.

(3) Fig. 1.7 shows a derivative control action where the error is constant.

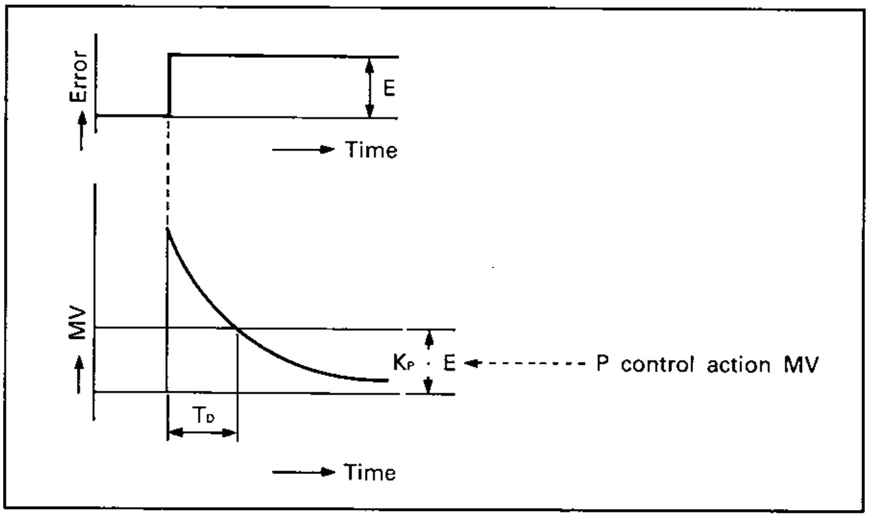


Fig. 1.7 Derivative Control Action Where Error Is Constant

(4) The derivative control action should be used with the proportional control action (as a PD control action) or with the proportional and integral control actions (as a PID control action) and cannot be used independently.

1.3.6 PID control action

(1) Exercises control using the MV obtained by the P, I and D control actions.

(2) Fig. 1.8 shows a PID control action where the error is constant.

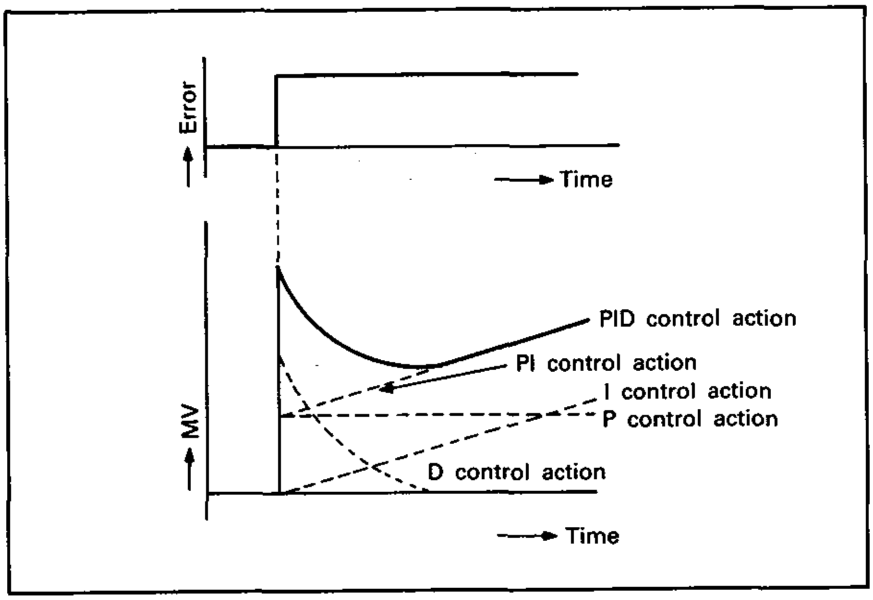


Fig. 1.8 PID Control Action Where Error Is Constant

2. A81CPU MODULE

This section gives performances, functions and handling procedures of the A81CPU.

2.1 Performances

Item			Performance
Operation processing	Control method		Repeated operation (Using stored program system)
	I/O control method		Direct mode
	Sampling time (s)		0.01 to 99.99
	Program loop monitoring (s)		5
	Allowable instantaneous power failure period (ms)		20
Number of instructions			65
Program capacity	Capacity (steps)		250 per program 8000 for all programs
	Number of programs		32 (No. 1 to 32)
PID control data	PID operation expression		Process value derivative type (incomplete derivative)
	Proportional constant (K_P)		0.01 to 100.00
	Integral time (T_I) (s)		1 to 32767
	Derivative time (T_D) (s)		0.00 to 300.00
	Set value (SV) setting range (%)		0.00 to 100.00
	Process value (PV) setting range (%)		0.00 to 100.00
	Input filter coefficient (α)		0.00 to 1.00
Device	Number of I/O points (PX/PY)		I + O = 512 (PX/PY100 to PX/PY2FF)
	Number of input points (PX) from PC CPU		64 (PX0 to PX3F)
	Number of output points (PY) to PC CPU		64 (PY40 to PY7F)
	Number of internal relays (PM) (points)		1024 (PM0 to PM1023)
	Timer (PT)	Number of points	128
		Specifications	10ms timer (PT0 to PT31)
			100ms timer (PT32 to PT127)
	Number of data registers (PD) (points)		1024 (PD0 to PD1023)
	Accumulator (A)	Number of points	3
		Specifications	1 bit (A0)
			16 bits (A1)
			Floating data (A2)
	Number of pointers (P) (points)		64 per program
	Number of special relays (SP. PM) (points)		512 (PM9000 to PM9511)
	Number of special registers (SP. PD) (points)		512 (PD9000 to PD9511)
Number of I/O points occupied with respect to PC CPU			128

Table 2.1 Performance List

2.2 Function List

Function	Description
Remote RUN/STOP	Allows remote RUN/STOP from external device (e.g. GPP, computer) with the RUN keyswitch in RUN position.
Program RUN/STOP	Allows program RUN/STOP from external device (e.g. GPP, computer, PC CPU) with the A81CPU in RUN mode.
PAUSE	Stops user program operations with the output (Y) status retained.
STEP-RUN	Executes the specified user program per instruction. Step-run may be executed in either of the two ways: • By specifying the loop count. • Per instruction.
LATCH	Retains device data if the A81CPU is switched off or reset or instantaneous power failure occurs 20ms or longer. Internal relays (PM0 to PM1023), timer (PT0 to PT127) coils/present values and data registers (PD0 to PD1023) can be latched.
CLOCK	Executes clock operation in the A81CPU. Clock data includes the year, month, day, hour, minute, second, and day of the week. Clock data can be read to special registers PD9095 to PD9098.

Table 2.2 Function List

2.3 Operation Processings

The A81CPU processings are shown in Fig. 2.1.

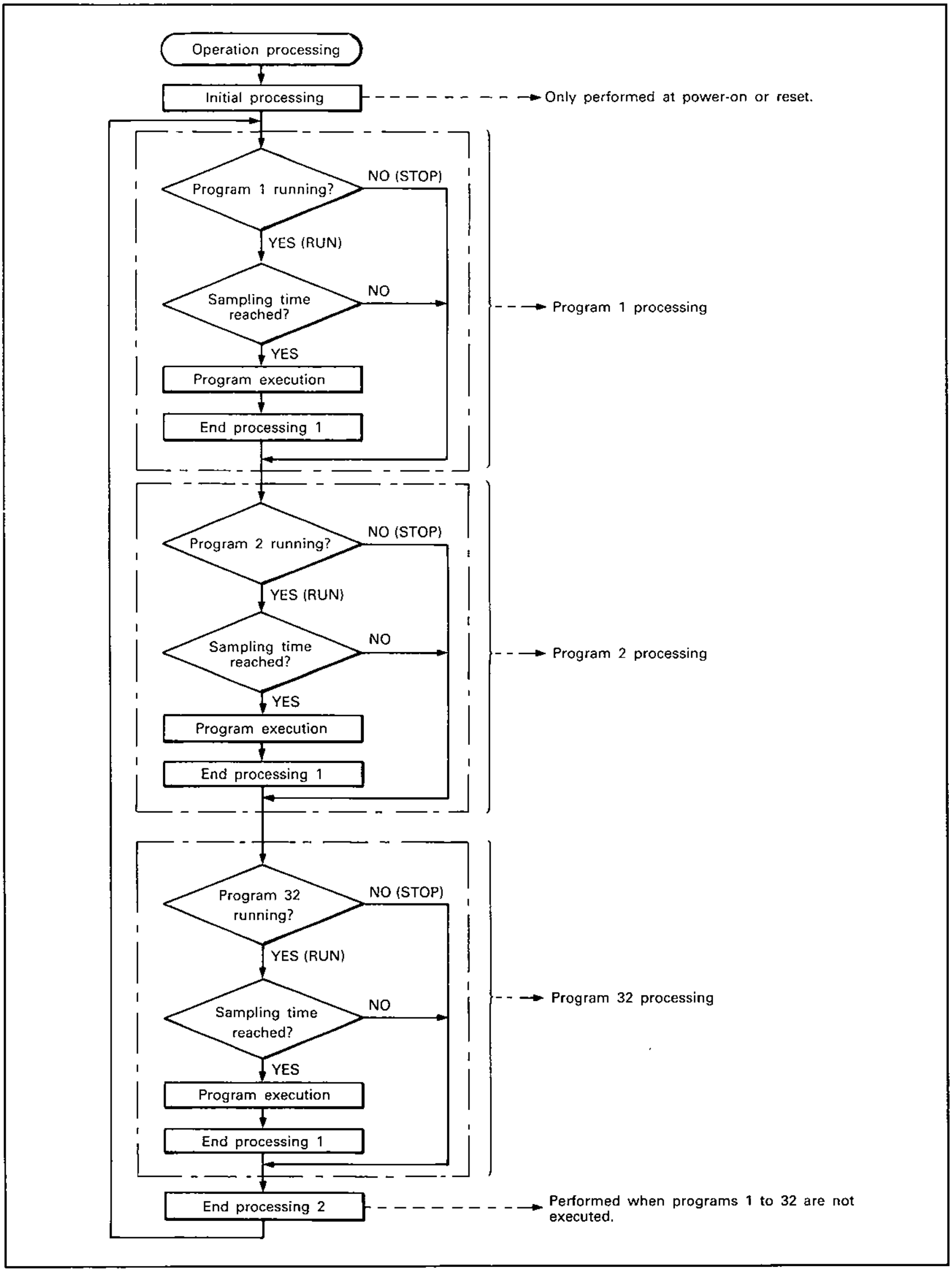


Fig. 2.1 Operation Processings

2.3.1 Repeated operation processing

(1) Repeated operation processing

Indicates that a sequence of operations are repeated.
The A81CPU repeats the processings of programs 1 to 32 as shown in Fig. 2.1.

(2) Stored program system

Sequentially reads and operates the required program stored in the corresponding user program area.
User programs are written by the GPP and stored to the user program areas.
The A81CPU reads the required program sequentially from the corresponding user program area and performs the repeated operation processing from step 0 to the END instruction.

(3) User program execution

(a) A user program is executed from step 0 to the END instruction if both of the following conditions are enabled:

1) Operation in RUN mode.

2) Preset sampling time (0.01s minimum) reached.

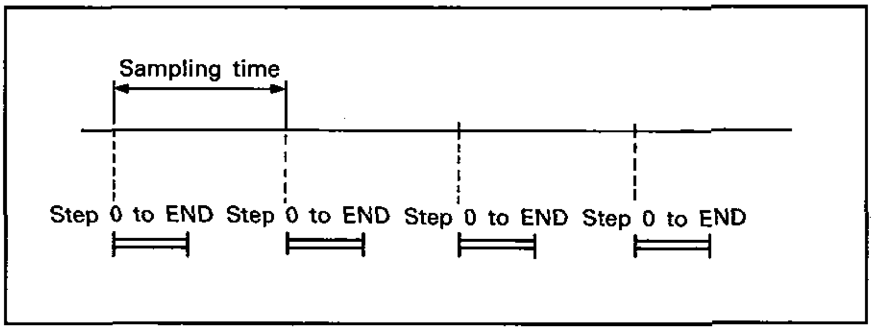
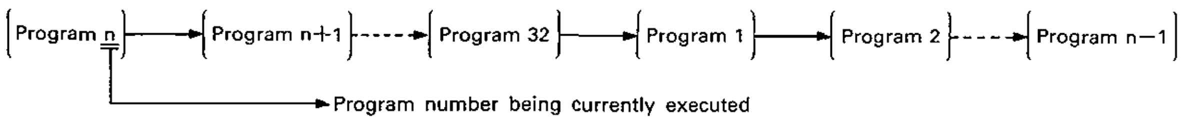


Fig. 2.2 Program Execution

(4) *Several program executions*

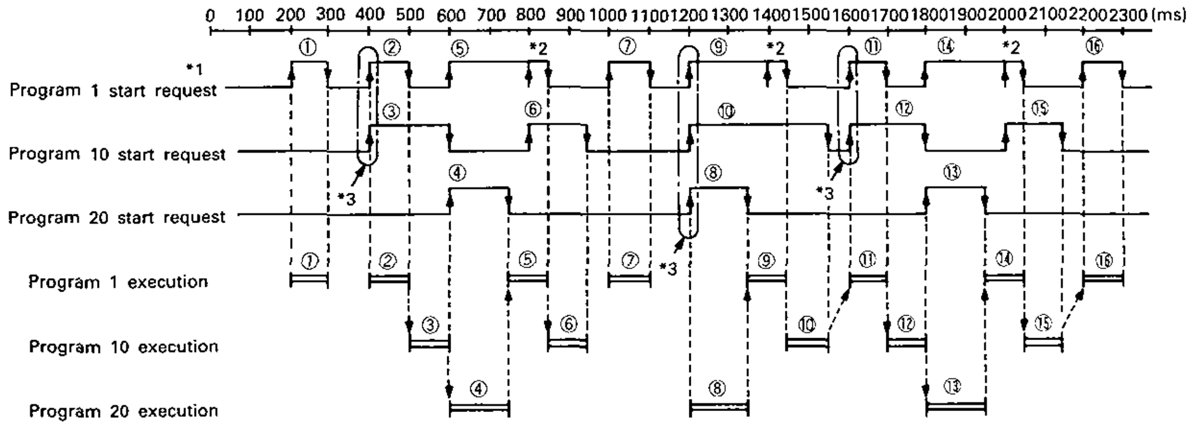
(a) Programs are executed in order of reaching the sampling time during processing of programs 1 to 32.

(b) The order of processing programs 1 to 32 remains unchanged as shown below if several programs reach their sampling times at the same time.



(c) A program may not be executed at the specified sampling time or the sampling time may be ignored depending on the sampling time and program processing time.
The example in Fig. 2.3 assumes that the sampling times and program processing times are as follows:

Program number	Processing Time	Sampling Time
1	100ms	200ms
10	100ms	400ms
20	150ms	600ms



*1: Any start request indicates a program execution start signal and is switched on when the sampling time is reached.
*2: Ignored if the start request is switched on.
*3: The order of processing depends on the current program being executed when the sampling time is reached.

Fig. 2.3 Several Program Executions

2.3.2 Initial processings

Before processing program 1, the A81CPU performs the following internal processings when the A81CPU is powered up or reset.

(1) Output module initialization

Resets the output module to switch all outputs off.

(2) Device clear

Clears the following devices (switched off or reset to zero).

(a) Special relays (PM9000 to PM9511)
(b) Special registers (PD9000 to PD9511)
(c) Accumulators (A0, A1, A2)

(3) I/O address assignment

Allocates I/O addresses to the I/O and special function modules loaded on the base unit.
For more information, see Section 3.3.

(4) I/O module data entry

Enters the types of the I/O and special function modules loaded on the base unit.

(5) Self-diagnosis

The A81CPU conducts self-checks when it is powered up or reset.
For further details, see the A81CPU User's Manual.

2.3.3 End processings

(1) There are the following types of end processings. (See Fig. 2.1)

(a) End processing 1 ······· Performed after a user program is executed. Terminates the current program processing and starts the the next program processing.

(b) End processing 2 ······ Performed when any user program is not executed.

(2) End processings 1 and 2 are performed as shown in Table 2.3.

○: indicates that the corresponding processing is processed.

End Processing	End Processing	
	1	2
Self-diagnosis Checks for fuse blown, battery power reduction, etc. For further details, see Section 4.10.	○	○
Data communication processing with computer link module Communicates with the computer link module (AJ71C24-S3, AD51E-S3) when data transfer is requested by the computer link module.		
Communication with GPP Communicates with the GPP when data transfer is requested by the GPP.		
Operation mode check Checks the following mode switching factors and switches mode. • RUN keyswitch setting • Operation mode switching request from the GPP • Remote RUN/STOP request from the computer, AD51E-S3		
Scan time timing Times the scan time of the program executed.	○	—
Loop monitoring timer reset Resets the loop monitoring timer.		

Table 2.3 End Processings 1 and 2

3. DEVICES

3.1 Device List

Device		Application Range	Data Access		Remarks	Refer To:
			A81CPU	PC CPU		
For communication with PC CPU	Input	PX00 to 3F (64 points)	Read	Write	(1) Accounted for as output (Y$_{(n+0)}$*0 to $_{(n+3)}$ F) by the PC CPU. (2) PX is expressed in hexadecimal.	Section 3.2.1
	Output	PY40 to 7F (64 points)	Write	Read	(1) Accounted for as input (X$_{(n+4)}$*0 to $_{(n+7)}$ F) by the PC CPU. (2) PY is expressed in hexadecimal.	
Dedicated to A81CPU	Input	PX/PY100 to 2FF (PX + PY = 512 points)	Read/write	Inaccessible	(1) PX/PY is expressed in hexadecimal.	Section 3.2.2
	Output					
Internal relay		PM0 to 1023 (1024 points)	Read/write	Inaccessible	(1) With latch function.	Section 3.3
Special relay		PM9000 to 9511 (512 points)	Read/write	Read/write (TO/FROM instruction)	(1) Special relays are accounted for as a buffer memory by the PC CPU and used in batches of 16 points.	Section 3.8
Data register		PD0 to 1023 (1024 points)	Read/write	Inaccessible	(1) Data is stored in floating format. (2) With latch function.	Section 3.4
Special register		PD9000 to 9511 (512 points)	Read/write	Read/write (TO/FROM instruction)	(1) Special registers have 16 bit locations. (2) Accounted for as a buffer memory by the PC CPU.	Section 3.9
Timer	10ms timer	PT0 to 31 (32 points)	Read/write	Inaccessible	(1) Up-timing retentive timer. (2) With latch function.	Section 3.5
	100ms timer	PT32 to 127 (96 points)				
Accumulator		A0 (1 point)	Read/write	Inaccessible	(1) For 1 bit data.	Section 3.6
		A1 (1 point)			(1) For 16 bit data	
		A2 (1 point)			(1) For floating data.	
Pointer		P□□00 to □□63 (2048 points)	—	—	(1) □□ = program number (01 to 32). (2) 64 points may be used in one program.	Section 3.7
Decimal constant	16 bits	K−32768 to 32767	—	—		
	32 bits	K±0.00001 to ±999900000				
Hexadecimal constant		0 to FFFF	—	—		

Table 3.1 Device List

*: n indicates the two most significant digits of the A81CPU head I/O address in the PC CPU system.

REMARKS

The A81CPU devices are headed by "P" so that they may be differentiated from the PC CPU devices.

3.2 Inputs/Outputs (PX/PY)

The A81CPU has the following inputs/outputs:

1) For communication with the PC CPU

2) For use with the A81CPU only

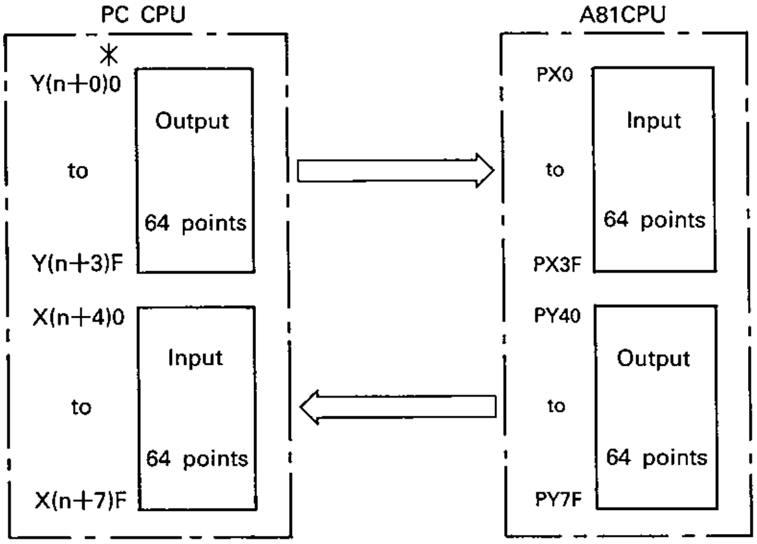
3.2.1 Inputs/outputs for communication with the PC CPU

(1) Used to transfer data between the PC CPU and A81CPU.

1) Input (PX) receives signals from the PC CPU.

2) Output (PY) transmits data to the PC CPU.

(2) 64 inputs and 64 outputs are configured as shown in Fig. 3.1.



1) *: n indicates the two most significant digits of the A81CPU head I/O address. (See Section 3.2.3.)
2) The A81CPU I/O addresses are between PX0 and 3F (inputs), PY40 and 7F (outputs).

Fig. 3.1 I/O (PX/PY) Configuration

(3) Some I/O addresses for communication with the PC CPU are pre-defined and others may be defined as appropriate. (See Table 3.2.)

I/O Number	Description
PX00 to PX1F	Program RUN/STOP request* (1) Gives an A81CPU RUN/STOP request from the PC CPU. • ON ········· RUN request • OFF ········· STOP request (2) Any program given a STOP request by the PC CPU remains stopped until a RUN request is given by the PC CPU. (3) For a correspondence between PX00 to 1F and programs, see Table 3.3.
PX20	PC ready (1) Switched on when the PC CPU switches on Y(n+2)0 to gain access to theA81CPU's PX/PY. (Address Y(n+2)0 in the PC CPU system corresponds to address PX20 in the A81CPU system.) • ON ········· The A81CPU's PX00 to 1F, 21 to 3D and PY40 to 7F are refreshed per 10ms. • OFF ········· PX00 to 1F, 21 to 3D data is all switched off and is not refreshed.
PX21 to PX3D	(1) Defined as appropriate by the user. (2) Used for communication with the PC CPU when PX20 is on.
PX3E, PX3F	Reserved
PY40 to PY5F	Program RUN request acceptance complete* (1) Corresponding PY is switched on when an A81CPU program is ready to start. (2) For a correspondence between PY40 to 5F and programs, see Table 3.3.
PY60	A81CPU hardware fault (1) Switched on to indicate an A81CPU hardware fault. Consult Mitsubishi representative. (2) When PY60 is switched on, the output status depends on the output reset switch setting. • Output reset switch OFF ········· Outputs retained. • Output reset switch ON ········· All outputs reset.
PY61 to PY7F	(1) Defined as appropriate by the user. (2) Used for communication with the PC CPU when PX20 is on.

Table 3.2 I/O for Communication with the PC CPU

*: For more information on program run/stop requests, see the A81CPU User's Manual.

IMPORTANT

PX3E, 3F are reserved for the A81CPU system and must not be switched on/off by the user from any program, computer, peripheral, etc.

Program No.	RUN/STOP Request		RUN/STOP Request Acceptance Complete	
	I/O address in PC CPU system	I/O address in A81CPU system	I/O address in PC CPU system	I/O address in A81CPU system
1	Y(n+0)0	PX00	X(n+4)0	PY40
2	Y(n+0)1	PX01	X(n+4)1	PY41
3	Y(n+0)2	PX02	X(n+4)2	PY42
4	Y(n+0)3	PX03	X(n+4)3	PY43
5	Y(n+0)4	PX04	X(n+4)4	PY44
6	Y(n+0)5	PX05	X(n+4)5	PY45
7	Y(n+0)6	PX06	X(n+4)6	PY46
8	Y(n+0)7	PX07	X(n+4)7	PY47
9	Y(n+0)8	PX08	X(n+4)8	PY48
10	Y(n+0)9	PX09	X(n+4)9	PY49
11	Y(n+0)A	PX0A	X(n+4)A	PY4A
12	Y(n+0)B	PX0B	X(n+4)B	PY4B
13	Y(n+0)C	PX0C	X(n+4)C	PY4C
14	Y(n+0)D	PX0D	X(n+4)D	PY4D
15	Y(n+0)E	PX0E	X(n+4)E	PY4E
16	Y(n+0)F	PX0F	X(n+4)F	PY4F
17	Y(n+1)0	PX10	X(n+5)0	PY50
18	Y(n+1)1	PX11	X(n+5)1	PY51
19	Y(n+1)2	PX12	X(n+5)2	PY52
20	Y(n+1)3	PX13	X(n+5)3	PY53
21	Y(n+1)4	PX14	X(n+5)4	PY54
22	Y(n+1)5	PX15	X(n+5)5	PY55
23	Y(n+1)6	PX16	X(n+5)6	PY56
24	Y(n+1)7	PX17	X(n+5)7	PY57
25	Y(n+1)8	PX18	X(n+5)8	PY58
26	Y(n+1)9	PX19	X(n+5)9	PY59
27	Y(n+1)A	PX1A	X(n+5)A	PY5A
28	Y(n+1)B	PX1B	X(n+5)B	PY5B
29	Y(n+1)C	PX1C	X(n+5)C	PY5C
30	Y(n+1)D	PX1D	X(n+5)D	PY5D
31	Y(n+1)E	PX1E	X(n+5)E	PY5E
32	Y(n+1)F	PX1F	X(n+5)F	PY5F

Table 3.3 Program RUN/STOP Requests, Request Acceptance Complete I/O Addresses

3.2.2 Inputs/outputs for use with the A81CPU only

Only used in the A81CPU programs and cannot be accessed by the PC CPU.

(1) Input is referred to as PX and output as PY, and their head addresses are fixed to PX/PY100.

(2) PX/PY ranges are 100 to 2FF.

3.3 I/O Addresses

I/O addresses indicate I/O module addresses for use in programs and are represented in hexadecimal.
I/O addresses may be determined by the A81CPU or PC system I/O assignment.

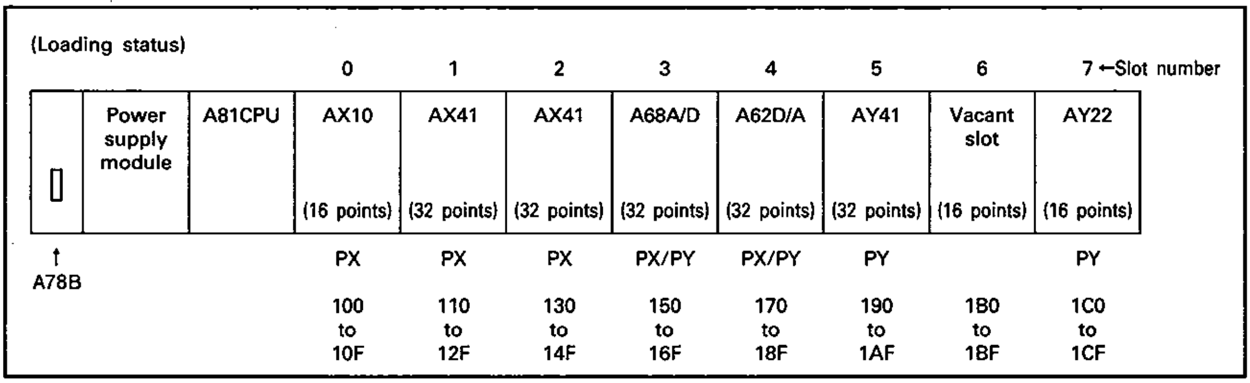
3.3.1 A81CPU independent system I/O addresses

(1) The A81CPU can only control eight slots of the A78B.

(2) I/O addresses are determined by the number of points occupied by the I/O and/or special function modules loaded on the A78B.

(3) Slot 0 always begins with PX/PY100 and the I/O address range is between 100 and 2FF.

(4) Assign 16 points to an vacant slot.



(Loading status)

			0	1	2	3	4	5	6	7 ←Slot number
↑ A78B	Power supply module	A81CPU	AX10 (16 points)	AX41 (32 points)	AX41 (32 points)	A68A/D (32 points)	A62D/A (32 points)	AY41 (32 points)	Vacant slot (16 points)	AY22 (16 points)
			PX	PX	PX	PX/PY	PX/PY	PY		PY
			100 to 10F	110 to 12F	130 to 14F	150 to 16F	170 to 18F	190 to 1AF	1B0 to 1BF	1C0 to 1CF

Fig. 3.2 A81CPU I/O Address Assignment Example

3.3.2 PC CPU system I/O addresses

The following should be noted when the A81CPU is used in the PC CPU system.

(1) Building block type PC CPU system

(a) The A78B is used as an extension base of the PC CPU.

(b) Connect the A78B to the last extension stage with respect to order of extension stage setting numbers (as opposed to order of extension cable connection).

(c) The A81CPU is accounted for as 128 I/O points in the PC CPU system.

(d) Set the A81CPU in PC CPU parameters as described below when making I/O assignment using the GPP:

1) First half slot ········ 64-point special function module
2) Second half slot ···· 64-point I/O module

(e) Any module loaded on the A78B is dedicated to the A81CPU and must be assigned in accordance with the A81CPU independent system I/O assignment.

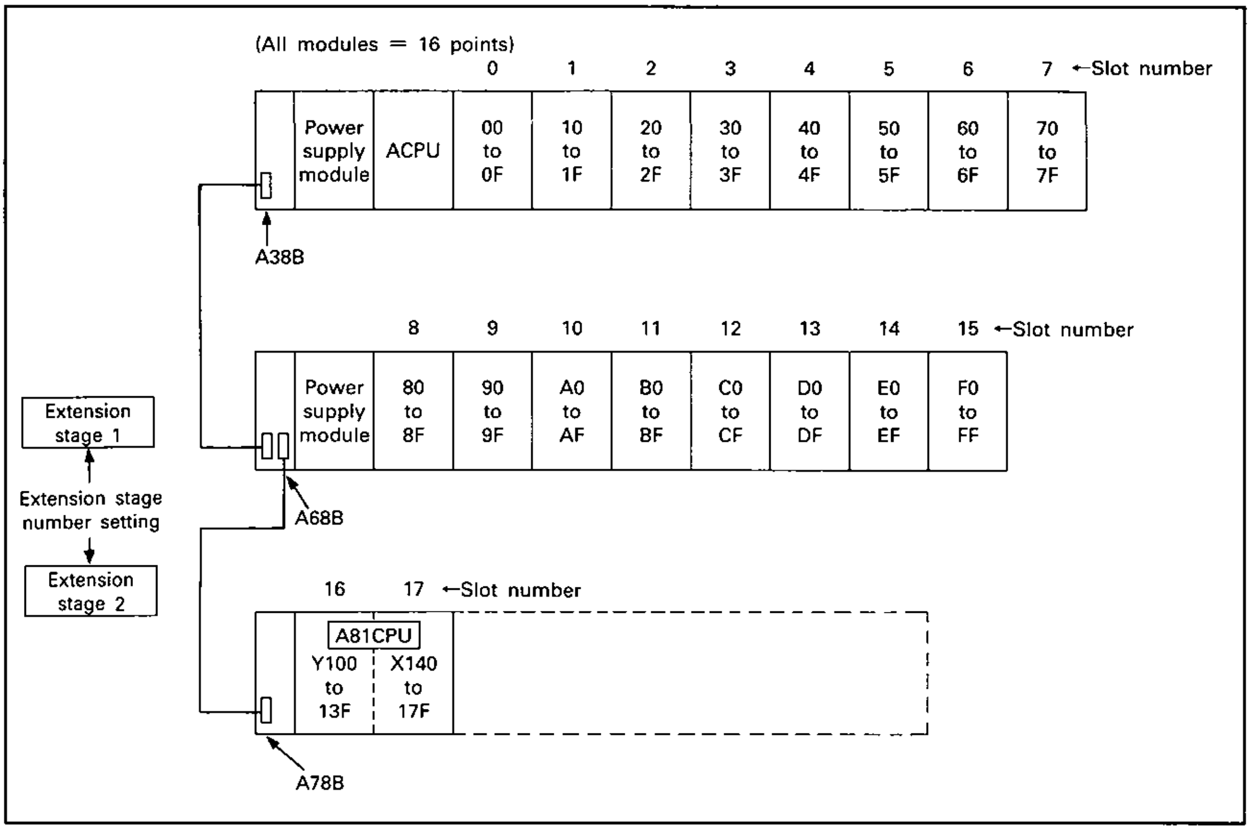


Fig. 3.3 I/O Address Assignment Example - A78B Set to the Last Extension Stage Number

(f) Notes on setting the A78B extension stage number to other than the last stage

1) When the A78B extension stage number is set to other than the last stage, assign the A78B I/O addresses independently of the modules loaded, on the assumption that two slots are occupied by a 128-point module and the other slots are vacant.

2) The A78B I/O addresses are allocated as follows in the PC CPU system:

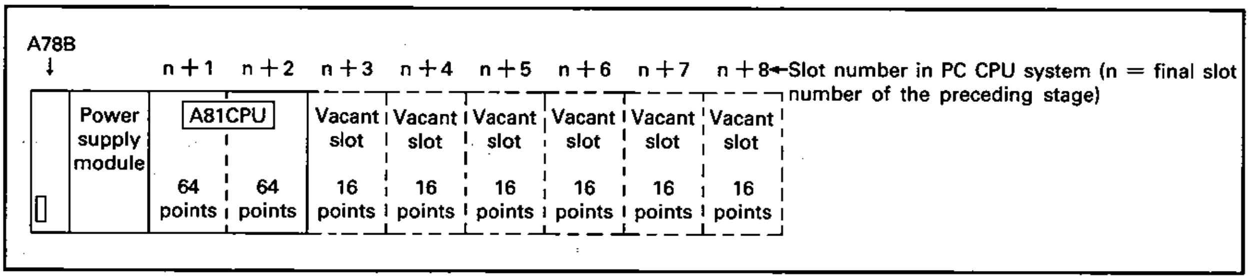


Fig. 3.4 A78B I/O Addresses in PC CPU System

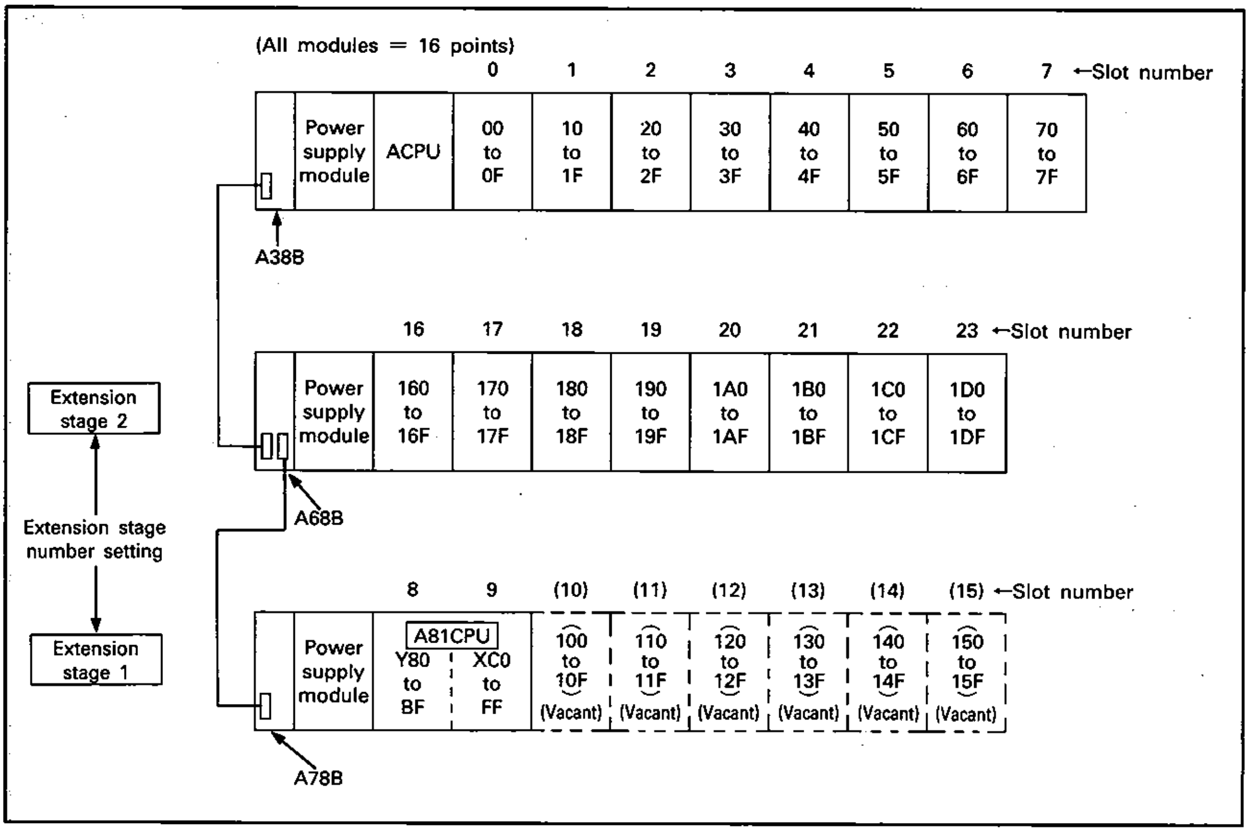


Fig. 3.5 I/O Address Assignment Example - A78B Set to Other Than the Last Extension Stage Number

(2) A0J2 type PC CPU system

(a) The A78B is used as an extension base of the A0J2CPU and cannot be used with the A65B, A68B.

(b) The A81CPU I/O addresses are always Y100 to 13F, X140 to 17F.

(c) Set the A78B extension stage number to stage 1.

(d) Any module loaded on the A78B is dedicated to the A81CPU and must be assigned in accordance with the A81CPU independent system I/O assignment.

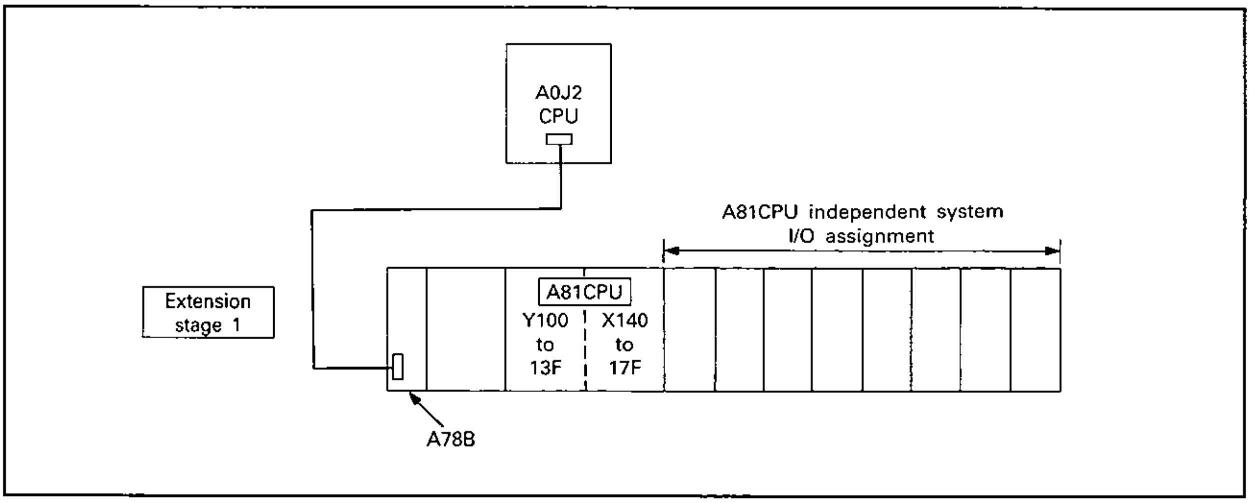


Fig. 3.6 I/O Address Assignment Example in A0J2CPU System

3.4 Internal Relay (PM)

A81CPU's auxiliary relay for ON/OFF data.

(1) Switched on/off per bit.

(2) Switched on/off in blocks of 16 bits for word data.

(3) All internal relays can be latched.

3.5 Data Register (PD)

Memory for storing A81CPU data.

(1) Stores data between $\pm 2.7 \times 10^{-27}$ and $\pm 9.2 \times 10^{18}$ in a floating-point format. For the floating-point format, see Section 6.1.3.

(2) All data registers can be latched.

3.6 Timer (PT)

Used in the A81CPU.

(1) Timer types

10ms and 100ms up-timing retentive timers

(2) Retentive timer

(a) Times the accumulative ON period of the timer coil.

(b) The timer coil is switched on/off by the $\boxed{\text{SET}}$ / $\boxed{\text{RST}}$ instruction and the present value is retained if the coil is switched off.

(3) Present value update timing

(a) 10ms timer ······ Counts the A81CPU's 10ms signals and updates the present value.

(b) 100ms timer ······ Counts the A81CPU's 10ms signals and updates the present value per 10 counts.

(4) Note on use of present value data

As there are no timer contacts, the control status should be changed in accordance with the operation result by executing a comparison instruction between the present value and other data.

(5) Timing range

(a) Timing range

0 to 32767 (10ms timer ······ 327.67 seconds, 100ms timer ······ 3276.7 seconds)

(b) Present value error

An excess of the timing range may result in the repetition of 32767, −32768, −32767 ······ −1, 0, 1 ······

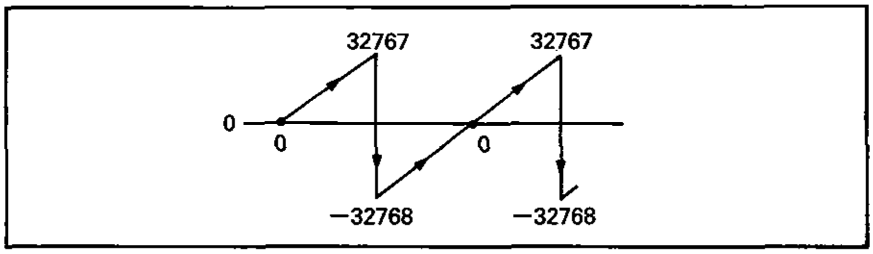


Fig. 3.7 Present Value Error

POINT

The timer times when step run is executed from the GPP with the A81CPU in STOP mode, and does not time when the A81CPU is in STOP or PAUSE mode.

(6) Clearing the present value

Execute the transfer, storage or other appropriate instruction to clear the present value or to change the data in the specified device.

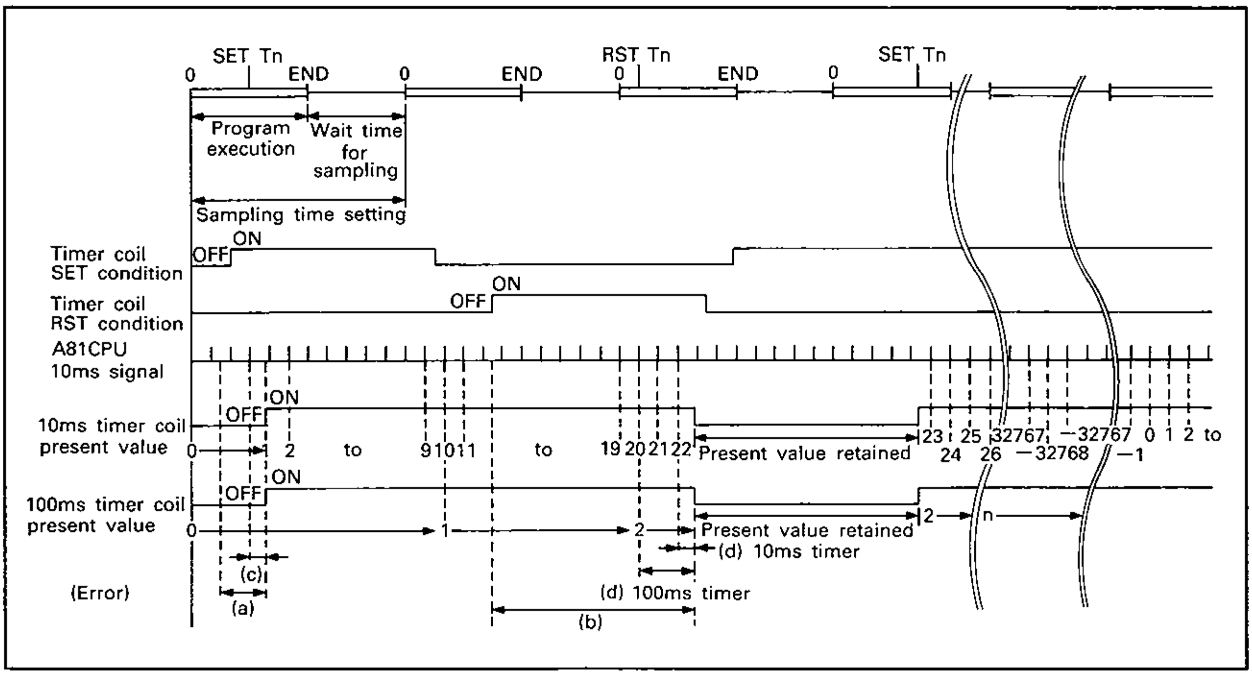


Fig. 3.8 Timing Chart

(7) Timer accuracies

Timer accuracies depend on the sampling time setting and coil ON/OFF timing as described below.

(a) Error between the timer coil ON timing and SET Tn instruction position in the program. (−1 sampling time)

(b) Error between the timer coil OFF timing and RST Tn instruction position in the program. (+1 sampling time)

(c) Error between timer coil ON timing and A81CPU 10ms signal. (−10ms)

(d) Error between timer coil OFF timing and A81CPU 10ms signal. (10ms timer ······ +10ms, 100ms timer ······ +100ms)

In consideration of (a) to (d), the overall accuracies are:

10ms timer ······ ±(sampling time ±10ms)
100ms timer ······ ±sampling time +100ms, −10ms

3.7 Accumulator

Data register for storing the operation result.

(1) Types

- For 1-bit data ······································ (A0)
- For 16-bit data ····································· (A1)
- For 32-bit floating-point data ···················· (A2)

(2) Application

Automatically stores the operation result when the corresponding instruction is executed or is used for operation and stores the result.

3.8 Pointer (P)

Indicates the destination of the branch instruction ($\boxed{\text{JC}}$, $\boxed{\text{JMP}}$, $\boxed{\text{CALL}}$).

(1) The pointer number must be specified when any branch instruction is executed.

(2) The same pointer number may be specified for several branch instructions.

(3) The same pointer number may be specified in any other program for the branch instruction in the program currently being executed.

(4) The same pointer number cannot be used at more than one location.

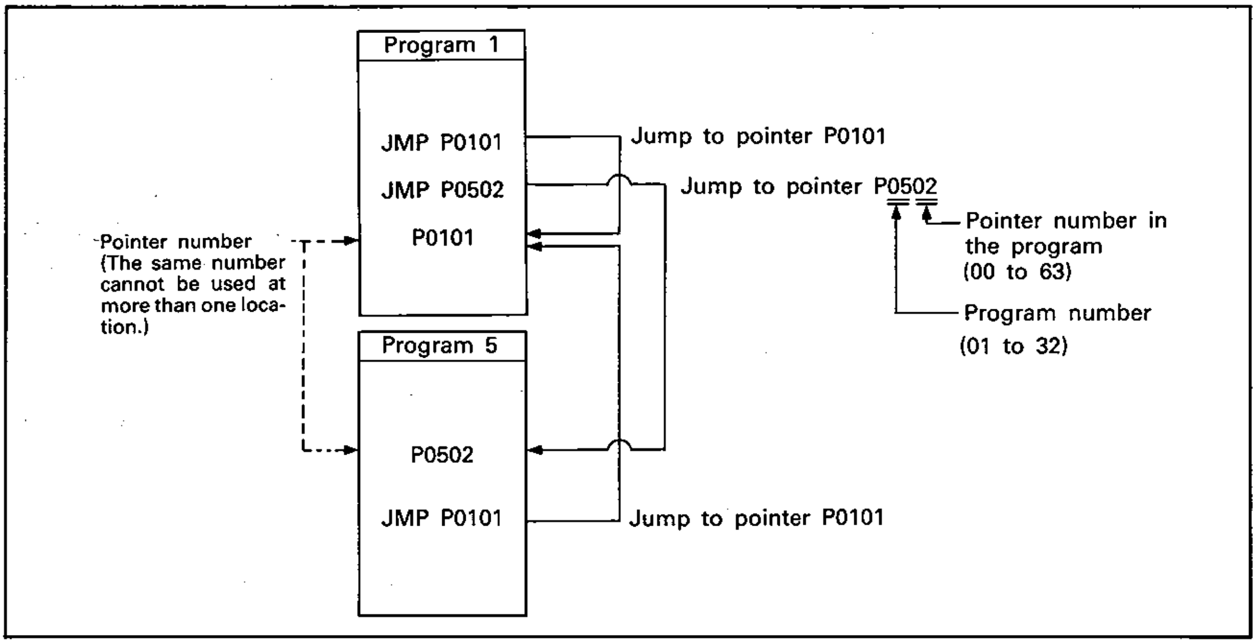


Fig. 3.9 Examples of Pointer Usage

3.9 Special Relays (SP.PM)

(1) The special relays are internal relays with special functions and the applications of some special relays are pre-defined as indicated in Table 3.4. Those marked * are only reset (switched off) by the program. Special relays not indicated in the table may be used as appropriate by the user.

User-defined area	PM9000 to PM9058
Pre-defined area	PM9059 to PM9099
User-defined area	PM9100 to PM9511

(2) The special relays are accessed by the TO / FROM instructions in the PC CPU program for communication with the PC CPU and are used in batches of 16 points as a buffer memory.

Number	Name	Description	Details
PM9059	Clock data set request	OFF: No processing ON: Data set request	Writes clock data from PD9095 to PD9098 to the clock devices after the END instruction is executed at the scan when M9059 is switched on.
PM9060	Clock data error	OFF: No error ON: Error	Switched on when a clock data (PD9095 to PD9098) error occurs.
PM9061	Clock data read request	OFF: No processing ON: Read request	Reads clock data in BCD to PD9095 to PD9098 when PM9061 is switched on.
PM9062	Program RUN/STOP request enable/disable flag	OFF: Disable ON: Enable	Enables program RUN/STOP request settings (PD9093, PD9094) from the computer, etc. when PM9062 is switched on.
PM9063	Data memory clear flag	OFF: No processing ON: Output clear	Clears all the data memory including the latched memory (other than special relays and special registers) in remote run mode from computer, etc. when PM9063 is on.
• PM9064	Fuse blown	OFF: Normal ON: Fuse blown in an I/O module	● Switched on when one or more output module fuses have blown. ● Reset when the A81CPU is reset.
* PM9065	AC DOWN detection	OFF: AC supply normal ON: AC is down	Switched on by an instantaneous power failure of within 20ms. Reset when the A81CPU is reset.
PM9066	Battery low	OFF: Normal ON: Battery low	Switched on when battery voltage drops below that specified. Switched off when battery voltage is restored.
• PM9067	Self-diagnostic error	OFF: No error ON: Error	Switched on by self-diagnosed error. Remains on if normal status is restored.

Table 3.4 Pre-Defined Special Relay List (Continue)

Number	Name	Description	Details
* PM9068 to PM9099	Self-diagnostic error (corresponding to program)	OFF: No error ON: Error	● Switched on by self-diagnosed program error. Remains on if normal status is restored. ● The error code, faulty step number, etc. are written to special registers PD9104 to 9199.

PM Number	Program Number	PM Number	Program Number
PM9068	1	PM9084	17
PM9069	2	PM9085	18
PM9070	3	PM9086	19
PM9071	4	PM9087	20
PM9072	5	PM9088	21
PM9073	6	PM9089	22
PM9074	7	PM9090	23
PM9075	8	PM9091	24
PM9076	9	PM9092	25
PM9077	10	PM9093	26
PM9078	11	PM9094	27
PM9079	12	PM9095	28
PM9080	13	PM9096	29
PM9081	14	PM9097	30
PM9082	15	PM9098	31
PM9083	16	PM9099	32

Table 3.4 Pre-Defined Special Relay List

3.10 Special Registers (SP.PD)

(1) The special registers are data registers with special functions and the applications of some special registers are pre-defined as indicated in Table 3.5. Those marked * are only reset (switched off) by the program. Special registers not indicated in the table may be used as appropriate by the user.

User-defined area	PD9000 to PD9092
Pre-defined area	PD9093 to PD9199
User-defined area	PD9200 to PD9511

(2) The special registers have 16 bit locations.

(3) The special registers are accessed by the TO / FROM instructions in the PC CPU program for communication with the PC CPU and are used in blocks of 16 points as a buffer memory.

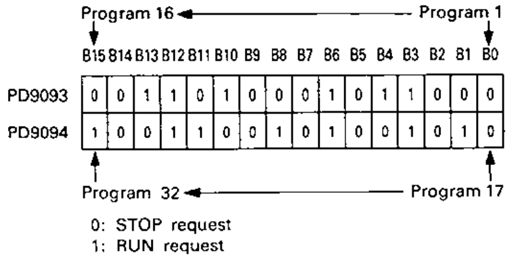
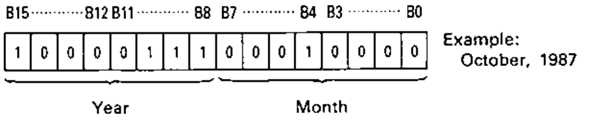
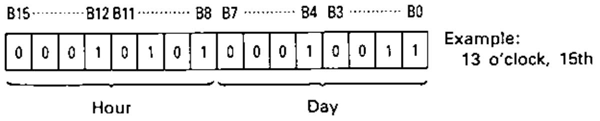
No.	Name	Data Stored	Details
PD9093 PD9094	Program RUN/STOP request	Bit map of program RUN/STOP requests	To specify a run/stop request for each program from the computer, etc. (Valid when PM9062 is on)  Program 16 ← Program 1 B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0 PD9093 0 0 1 1 0 1 0 0 0 1 0 1 1 0 0 0 PD9094 1 0 0 1 1 0 0 1 0 1 0 0 1 0 1 0 Program 32 ← Program 17 0: STOP request 1: RUN request
PD9095	Clock data	Clock data (Year, month)	Year (two last digits), month are written in BCD.  B15 ······ B12 B11 ······ B8 B7 ······ B4 B3 ······ B0 1 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 Year Month Example: October, 1987
PD9096	Clock data	Clock data (Day, hour)	Day, hour are written in BCD.  B15 ······ B12 B11 ······ B8 B7 ······ B4 B3 ······ B0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 1 1 Hour Day Example: 13 o'clock, 15th

Table 3.5 Pre-Defined Special Register List (Continue)

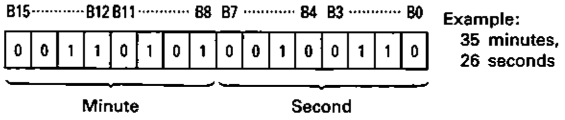
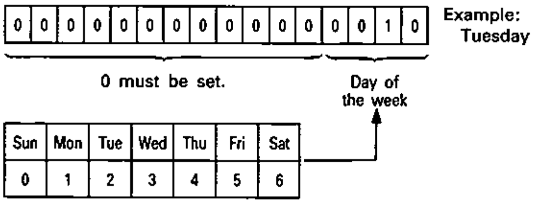
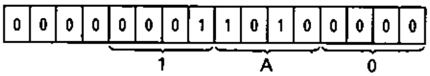
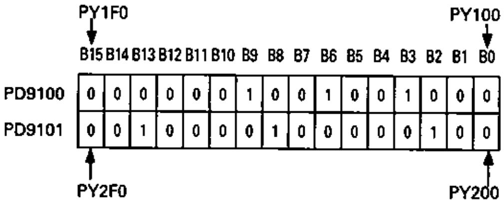
No.	Name	Data Stored	Details
PD9097	Clock data	Clock data (Minute, second)	Minute, second are written in BCD.  Example: 35 minutes, 26 seconds
PD9098	Clock data	Clock data (Day of the week)	Day of the week is written in BCD.  Example: Tuesday
PD9099	Fuse blown	Lowest module number location with blown fuse	Indicates the head I/O address of the lowest I/O module with blown fuse in hexadecimal. Example: The bit map is as shown below when the fuse of the output module at PY1A0 to 1AF is blown.  Cleared by resetting PD9100 and PD9101 to 0.
*PD9100 PD9101	Fuse blown module	Bit map of I/O modules with blown fuses	Indicates the output module numbers with blown fuses in blocks of 16 I/O points.  0: Fuse blown 1: Fuse normal
*PD9102	AC DOWN counter	AC DOWN count	1 is added each time the input voltage drops to 80% or less of rated during operation, and the value stored in BIN.
*PD9103	Self-diagnostic error	Self-diagnostic error number	● Records the self-diagnosed error number in BIN. ● The error number first stored is retained until reset with the exception of error codes 60 (fuse blown) and 61 (battery error) which are overwritten by the most recent error. ● For the error codes and definitions, see the error code list (Section 7.1).

Table 3.5 Pre-Defined Special Register List (Continue)

<table>
<tr><th>No.</th><th>Name</th><th>Data Stored</th><th>Details</th></tr>
<tr>
<td>* PD9104
to
PD9199</td>
<td>Self-diagnostic error (corresponding to program)</td>
<td>• Self-diagnosed error number
• Faulty step number
• Further error definition</td>
<td>
● The self-diagnosed error code and faulty step number are stored in BIN, and further error definition in hexadecimal per program.
● Data is not cleared if normal status is restored.
<table>
<tr><th>Program Number</th><th>Self-Diagnosed Error Number</th><th>Faulty Step Number</th><th>Further Error Definition</th></tr>
<tr><td>1</td><td>PD9104</td><td>PD9105</td><td>PD9106</td></tr>
<tr><td>2</td><td>PD9107</td><td>PD9108</td><td>PD9109</td></tr>
<tr><td>3</td><td>PD9110</td><td>PD9111</td><td>PD9112</td></tr>
<tr><td>4</td><td>PD9113</td><td>PD9114</td><td>PD9115</td></tr>
<tr><td>5</td><td>PD9116</td><td>PD9117</td><td>PD9118</td></tr>
<tr><td>6</td><td>PD9119</td><td>PD9120</td><td>PD9121</td></tr>
<tr><td>7</td><td>PD9122</td><td>PD9123</td><td>PD9124</td></tr>
<tr><td>8</td><td>PD9125</td><td>PD9126</td><td>PD9127</td></tr>
<tr><td>9</td><td>PD9128</td><td>PD9129</td><td>PD9130</td></tr>
<tr><td>10</td><td>PD9131</td><td>PD9132</td><td>PD9133</td></tr>
<tr><td>11</td><td>PD9134</td><td>PD9135</td><td>PD9136</td></tr>
<tr><td>12</td><td>PD9137</td><td>PD9138</td><td>PD9139</td></tr>
<tr><td>13</td><td>PD9140</td><td>PD9141</td><td>PD9142</td></tr>
<tr><td>14</td><td>PD9143</td><td>PD9144</td><td>PD9145</td></tr>
<tr><td>15</td><td>PD9146</td><td>PD9147</td><td>PD9148</td></tr>
<tr><td>16</td><td>PD9149</td><td>PD9150</td><td>PD9151</td></tr>
<tr><td>17</td><td>PD9152</td><td>PD9153</td><td>PD9154</td></tr>
<tr><td>18</td><td>PD9155</td><td>PD9156</td><td>PD9157</td></tr>
<tr><td>19</td><td>PD9158</td><td>PD9159</td><td>PD9160</td></tr>
<tr><td>20</td><td>PD9161</td><td>PD9162</td><td>PD9163</td></tr>
<tr><td>21</td><td>PD9164</td><td>PD9165</td><td>PD9166</td></tr>
<tr><td>22</td><td>PD9167</td><td>PD9168</td><td>PD9169</td></tr>
<tr><td>23</td><td>PD9170</td><td>PD9171</td><td>PD9172</td></tr>
<tr><td>24</td><td>PD9173</td><td>PD9174</td><td>PD9175</td></tr>
<tr><td>25</td><td>PD9176</td><td>PD9177</td><td>PD9178</td></tr>
<tr><td>26</td><td>PD9179</td><td>PD9180</td><td>PD9181</td></tr>
<tr><td>27</td><td>PD9182</td><td>PD9183</td><td>PD9184</td></tr>
<tr><td>28</td><td>PD9185</td><td>PD9186</td><td>PD9187</td></tr>
<tr><td>29</td><td>PD9188</td><td>PD9189</td><td>PD9190</td></tr>
<tr><td>30</td><td>PD9191</td><td>PD9192</td><td>PD9193</td></tr>
<tr><td>31</td><td>PD9194</td><td>PD9195</td><td>PD9196</td></tr>
<tr><td>32</td><td>PD9197</td><td>PD9198</td><td>PD9199</td></tr>
</table>
*: For more information on the error codes and further definitions stored, see the error code list (Section 7.1).
</td>
</tr>
</table>

Table 3.5 Pre-Defined Special Register List

3.11 Buffer Memory

(1) The special relays and special registers are accounted for as a buffer memory by the PC CPU.

(2) The special relays and special registers are accessed by the PC CPU using the FROM / TO instructions in the PC CPU sequence program.

(3) Buffer memory data is made up of 16 bits per address with addresses expressed in decimal.
Special relays are used in batches of 16 points with the head number being PM9000 or a multiple of 16.

A81CPU devices

Following the PC CPU addresses

	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
0	PD9000															
1	PD9001															
2	PD9002															
to	to															
510	PD9510															
511	PD9511															
512	PM 9015	PM 9014	PM 9013	PM 9012	PM 9011	PM 9010	PM 9009	PM 9008	PM 9007	PM 9006	PM 9005	PM 9004	PM 9003	PM 9002	PM 9001	PM 9000
513	PM 9031	PM 9030	PM 9029	PM 9028	PM 9027	PM 9026	PM 9025	PM 9024	PM 9023	PM 9022	PM 9021	PM 9020	PM 9019	PM 9018	PM 9017	PM 9016
to	to															
542	PM 9495	PM 9494	PM 9493	PM 9492	PM 9491	PM 9490	PM 9489	PM 9488	PM 9487	PM 9486	PM 9485	PM 9484	PM 9483	PM 9482	PM 9481	PM 9480
543	PM 9511	PM 9510	PM 9509	PM 9508	PM 9507	PM 9506	PM 9505	PM 9504	PM 9503	PM 9502	PM 9501	PM 9500	PM 9499	PM 9498	PM 9497	PM 9496

Fig. 3.10 Relation between Buffer Memory Addresses and Special Relays, Special Registers

4. MACRO FUNCTIONS

4.1 Macro Functions

Consist of the basic part (PID macro function) and optional functions, and are equivalent to an instrument which has input processing, PID control operation and output processing facilities. Normal PID control is provided by the basic part only. Required additional functions are separately available as options, e.g. square root extraction (√) used to linearize differential pressure input.

A wide variety of control functions may be used for various types of process control by adding optional functions to the basic part (PID macro function).

The macro functions for PID operation are referred to as PID macro functions which will be explained below.

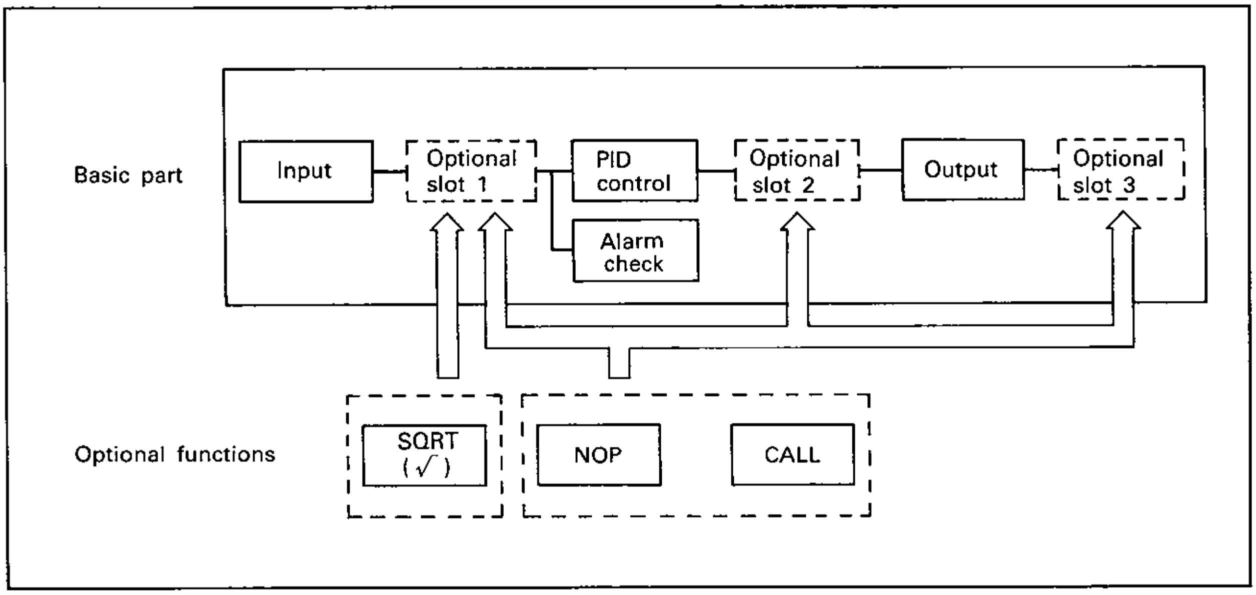


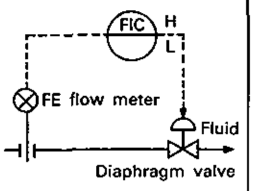
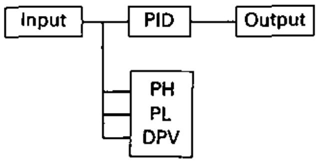
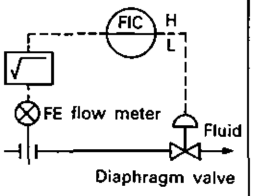
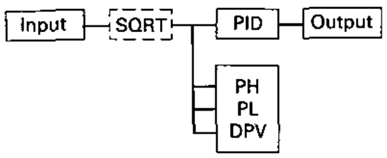
Fig. 4.1 PID Macro Function Configuration

(1) Correspondence between macro functions and loops

The maximum number of loops is 64 with one loop corresponding to one macro function.
Examples of using the macro functions for process control are shown below.
No. 1 indicates a simple PID controller consisting of the basic part only.
No. 2 indicates a PID controller preceded by square root extraction ($\sqrt{\ }$) using the optional function.
Both examples use one macro function, i.e. one loop.

REMARKS

A loop indicates the area processed by PID operation.
One loop consists of one PID operation area.

No.	Operation	Process	Macro Function
1	1 loop PID		PID function basic part 
2	1 loop PID+ linearization		PID function basic part+optional function 

(2) Optional function types

Any optional function can be specified by the peripheral equipment.
Any of the following optional functions may be selected for the corresponding optional slot.

(a) NOP

No operation.
Used to perform no processing in the optional slot.

(b) SQRT

Extracts the square root of accumulator (A2) data and stores the operation result to accumulator (A2). Used to linearize differential pressure input, etc.

(c) CALL

Calls the user program specified by the pointer. Used to perform operation other than square root extraction.
For example, by specifying CALL P3200, program 32 is executed beginning with pointer P3200.

(3) Optional slot 1 function

Uses optional functions to process PV.

(a) Before the processing of optional slot 1 is started, accumulator (A2) contains PV processed by the Input area.

(b) When the processing of the function in slot 1 is complete, the next processing is executed with the (A2) value used as PV.
PV in (A2) should be protected when specifying "CALL."

(4) Optional slot 2 function

Only used to specify "NOP."

(5) Optional slot 3 function

Used to output the operation result of any macro function.

(a) Before the processing of optional slot 3 is started, devices defined by the parameters contain MV, alarm, etc.

4.2 PID Macro Functions

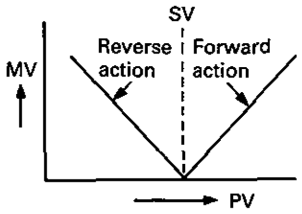
4.2.1 General operation

(1) PID algorithm

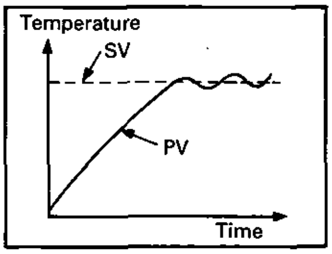
The A81CPU performs operation by velocity and process value derivative type operation expressions. The expressions are available in forward and reverse actions. The basic PID operation expressions are indicated below.

Description		Expression	Definition
Velocity, PV derivative type	Forward action	$\Delta MV_n = K_P\{(EV_n - EV_{n-1}) + \frac{\Delta T_S}{T_I} EV_n - \Delta D_n\}$ $\Delta D_n = \frac{T_D}{\Delta T_S + \alpha_D \cdot T_D}(2PV_{n-1} - PV_n - PV_{n-2}) - \frac{\alpha_D \cdot T_D}{\Delta T_S + \alpha_D \cdot T_D} \cdot \Delta D_{n-1}$ $MV_{PID} = \Sigma \Delta MV_n$ $EV_n = PV_n - SV$	ΔMV_n: Current output change rate EV_n: Current sampling error value EV_{n-1}: Previous sampling error value PV_n: Current sampling process value PV_{n-1}: Previous sampling process value PV_{n-2}: Process value two samples previous MV_{PID}: Manipulated value after PID operation SV: Set value ΔT_S: Sampling period K_P: Proportional gain T_I: Integral time T_D: Derivative time ΔD_n: Derivative term α_D: Derivative gain
	Reverse action	$\Delta MV_n = K_P\{(EV_n - EV_{n-1}) + \frac{\Delta T_S}{T_I} EV_n + \Delta D_n\}$ $\Delta D_n = \frac{T_D}{\Delta T_S + \alpha_D \cdot T_D}(2PV_{n-1} - PV_n - PV_{n-2}) + \frac{\alpha_D \cdot T_D}{\Delta T_S + \alpha_D \cdot T_D} \cdot \Delta D_{n-1}$ $MV_{PID} = \Sigma \Delta MV_n$ $EV_n = SV - PV_n$	

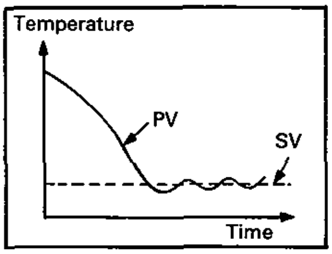
The reverse action increases the PV to the SV when the MV decreases.
The forward action decreases the PV to the SV when the MV increases.
The following figure illustrates the forward and reverse acting processes by using the MV, PV and SV.



Typical processes involving forward and reverse action controls are shown below:



Reverse action (Heating)



Forward action (Cooling)

Each loop may be set individually for forward or reverse action.

(2) Input processing

Reads PV, filters the data and transmits it to the PID operation area.
This processing is executed at the **Input** area shown in Fig. 4.1
The input processing area functions are shown in Fig. 4.2.

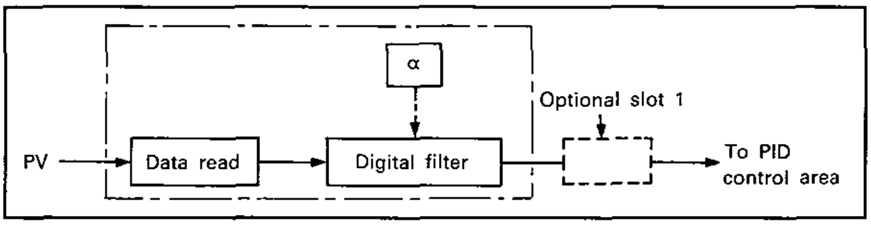


Fig. 4.2 Input Processing Area Functions

Data read

Reads PV data from the device specified in the parameter.

Digital filter

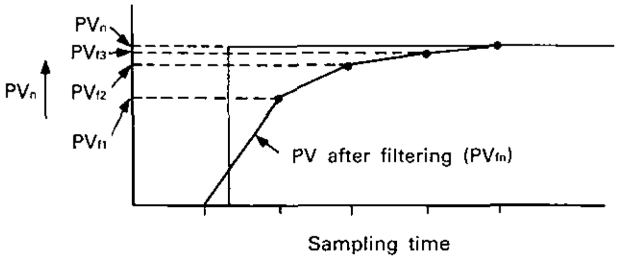
A filter is required because direct digital control (DDC), etc. may affect the noise control of input signals from sensors.
The A81CPU uses a first-order lag filter to remove high-frequency noise.
Filtered PV can be calculated as follows:

$$PV_{fn} = (PV_n \text{ input value}) + \alpha \ (PV_{fn-1} - PV_n \text{ input value})$$

where α = filter coefficient (0.00 to 1.00)
PV_{fn}, PV_{fn-1} = PV after filtering

In the above expression, filter coefficient should only be set in the parameter by the user.
α: corresponds to the first-order lag time constant.
The greater the filter coefficient, the longer the lag time.
PV is not filtered when the filter coefficient is 0.

Example: PVfn is as follows when the filter coefficient is 0.3.



$$PV_{f1} = PV_n + 0.3(0 - PV_n)$$
$$PV_{f2} = PV_n + 0.3(PV_{f1} - PV_n)$$
$$PV_{f3} = PV_n + 0.3(PV_{f2} - PV_n)$$

Related parameter

$\boxed{\alpha}$ .... Filter coefficient (0.00 to 1.00)

(3) Alarm check

Gives an alarm if the PV transmitted from the input processing area exceeds any of its predefined high limit, low limit and change rate.
This processing is executed at the Alarm check area shown in Fig. 4.1

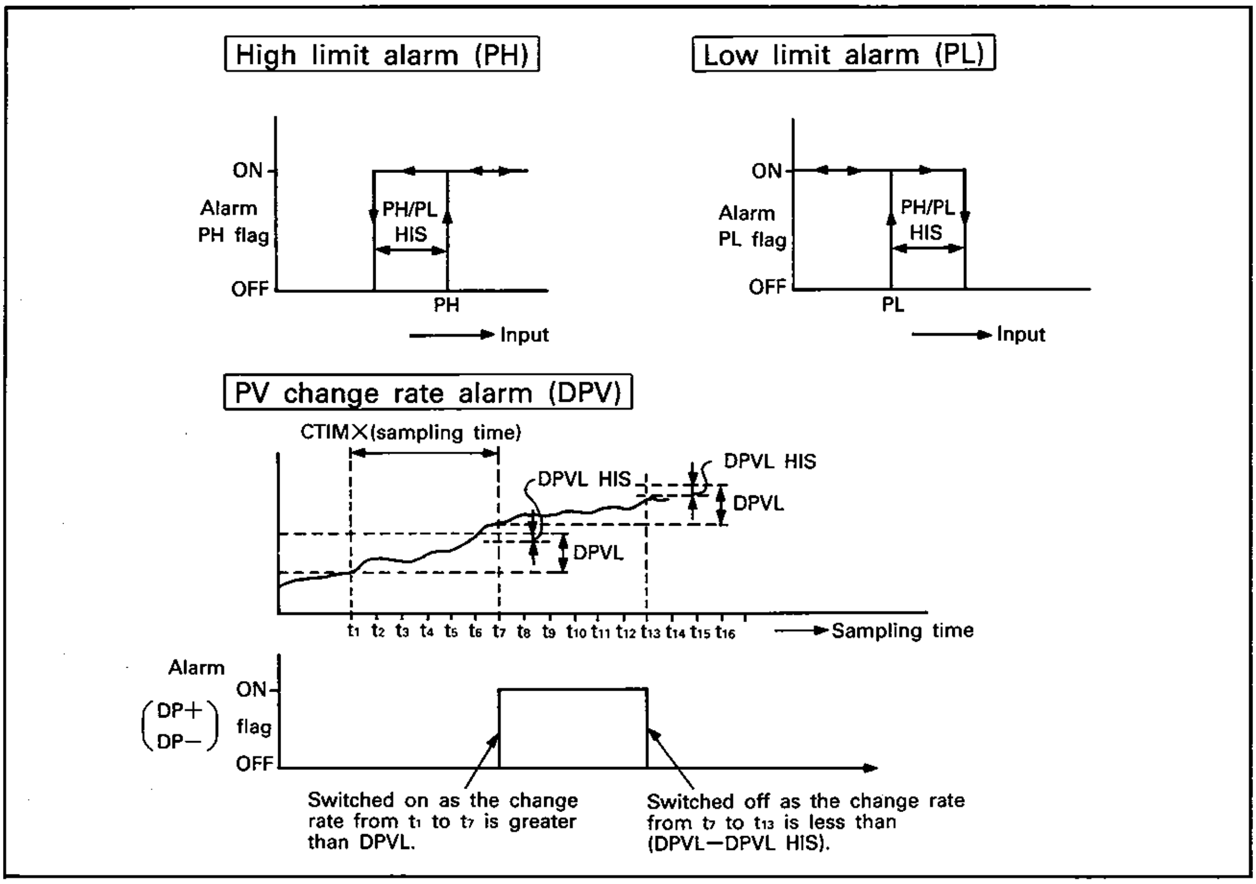


Fig. 4.3 Alarm Check Functions

High limit alarm PH

Has hysteresis as shown in Fig. 4.3.
By specifying the high limit alarm set value (PH) and high, low limit alarm hysteresis values (PH/PL HIS), the alarm PH flag is switched on/off in accordance with the input value to set/reset the internal relay PM set in ALARM parameter PH.

Related parameters

PH	High limit alarm set value (0.00 to 100.00%)
PH/PL HIS	High, low limit alarm hysteresis values (0.00 to 100.00%)
ALARM	Alarm (PM0 to PM1018) (The PM number depends on the device number set to "PV change rate alarm positive check".) Reads PV data from the device specified in the parameter.

Low limit alarm PL

Has hysteresis as shown in Fig. 4.3.
By specifying the low limit alarm set value (PL) and high, low limit alarm hysteresis values (PH/PL HIS), the alarm PL flag is switched on/off in accordance with the input value to set/reset the internal relay PM set in ALARM parameter PL.

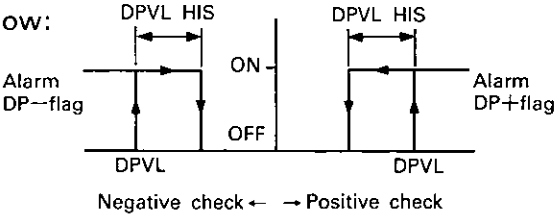
Related parameters

PL	Low limit alarm set value (0.00 to 100.00%)
PH/PL HIS	High, low limit alarm hysteresis values (0.00 to 100.00%)
ALARM	Alarm (PM0 to PM1019) (The PM number depends on the device number set to "PV change rate alarm positive check".)

PV change rate alarm DPL

Checks the change rate by comparing the PV input change with the PV change rate high limit set in the parameter (DPVL) in the specified duration.

The PV change rate is checked per (PV change rate check duration (CTIM)×sampling time). If the change rate is greater than DPVL, the internal relay PM specified in parameter DP+ (for positive check) or DP− (for negative check) is switched on. This internal relay is switched off if the PV change rate is less than (DPVL − PV change rate check hysteresis value (DPVL HIS)).
The internal relay remains on if the PV change rate is between DPVL and (DPVL−DPVL HIS).
This is illustrated below:



The change rate check may be specified in the PV change rate check direction parameter (POL) for any of positive, negative and both directions.
After the change rate check, PV input is written to the device specified in parameter PV.
This function may be used to check any sensor fault, wiring fault, sudden process change, etc.

Related parameters

DPVL	PV change rate alarm set value (0.00 to 100.00%)
DPVL HIS	PC change rate check hysteresis value (0.00 to 100.00%)
ALARM	Alarm (PM0 to PM1016)
POL	PV change rate check direction 0 ········· Positive check 1 ········· Negative check 2 ········· Both check
CTIM	PV change rate check duration (1 to 255 times)

(4) Output processing

Sets upper and lower limits for MV as calculated by the PID algorithm, processes and outputs MV, and gives alarm in accordance with the high, low limits and change rate. This processing is executed at the Output area shown in Fig. 4.1

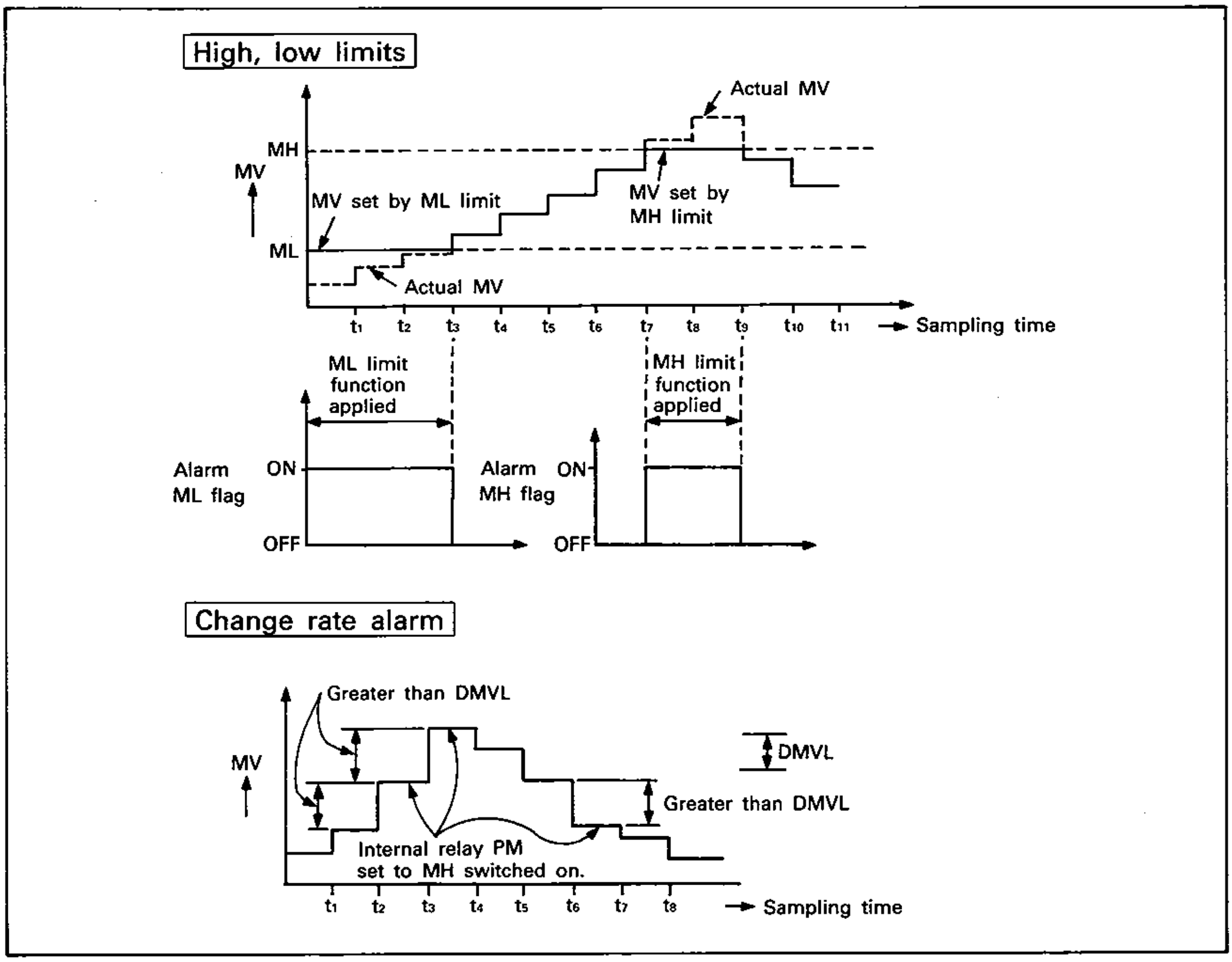


Fig. 4.4 Output Processing Area Functions

MV high limit

Give an alarm in automatic mode only.
The internal relay PM set in ALARM parameter MH is switched on if MV exceeds the MV high limit set in MV high limit parameter MH.
The internal relay PM is switched off when MV drops below the high limit in MH.

Related parameters

MH ························ MV high limit (−2.50 to 102.50%)
ALARM ····················· Alarm (PM0 to PM1022)
(The PM number depends on the device number set to "PV change rate alarm positive check".)

MV low limit

Give an alarm in automatic mode only.
MV is adjusted to the set value in MV low limit parameter ML and the internal relay PM set in ALARM parameter ML is switched on if MV drops below the MV low limit set in MV low limit parameter ML.
The PM is switched off when MV exceeds the low limit in ML.

Related parameters

MH	MV low limit (−2.50 to 102.50%)
ALARM	Alarm (PM0 to PM1023) (The PM number depends on the device number set to "PV change rate alarm positive check".)

MV change rate alarm

Gives an alarm in automatic mode only.

Related parameters

$DMV= | MV_n - MV_{n-1} |$

DMV=MV change rate
MV_n=current manipulated value
MV_{n-1}=previous manipulated value

The internal relay PM set in ALARM parameter DMV is switched on if the MV change rate (DMV) exceeds the value set in the MV change rate alarm parameter (DMVL).
The MV change rate alarm is valid in either of the positive and negative directions.

Related parameters

DMVL	MV change rate alarm value (0.00 to 100%)
ALARM	Alarm (PM0 to PM1021) (The PM number depends on the device number set to "PV change rate alarm positive check".)

Output

MV is checked and corrected by the MV high, low limit and/or MV change rate limit functions and is then written to the device specified in parameter MV.
In manual (M) mode, data is written from the device specified in manual MV parameter (MV MAN) to the device specified in parameter MV.

4.2.2 Operation mode

Determines MV for PID control.
Either of automatic and manual modes may be selected in accordance with the required control by writing the corresponding value by using the program to the data register (PD) specified by parameter setting.
(For parameter setting, see the SW0GHP-A81PC PID Control Software Package Operating Manual.)

(1) Automatic mode

Uses the PID operation result as MV for PID control.
MV is automatically defined by executing the macro function.

(2) Manual mode

Allows MV to be set by the peripheral device or in the user program for process control, independently of the PID operation result.
MV is changed manually by the user program or peripheral device. MV is specified by writing the required value by using the peripheral device or program to the data register (PD) selected by parameter setting.

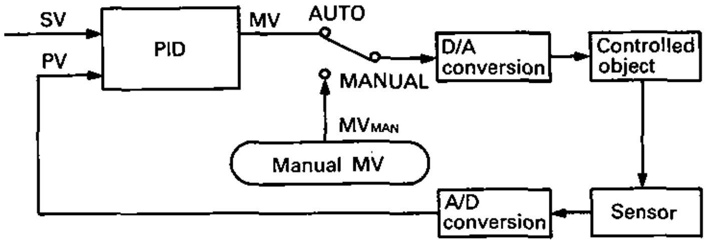


Fig. 4.5 Relation between Modes and I/O

POINT

(1) Automatic or manual mode must be selected before the loop designation instruction in the user program.

(2) The tracking function must be used with mode selection as control may be affected by sudden changes in MV. See Section 4.2.3.

4.2.3 Tracking function

(a) Prevents sudden MV output changes so that MV output is switched smoothly when operation is switched from automatic to manual mode or vice versa.

(b) Limits the MV change in the output processing area so that MV output is switched smoothly after switching from manual to automatic mode.

The tracking function includes both the bumpless function (a) and output limit function (b).

(1) Bumpless function

(a) Switching from manual to automatic mode

- Transfers the manual MV (data stored in the data register (PD) set in parameter MV_{MAN}) to the MV work area.

(b) Switching from automatic to manual mode

- Transfers MV from the MV work area to the MV register (data register (PD) set in parameter MV_{MAN}).
- Transfers PV to the SV area per sampling time during manual control.

(2) Output limit function

Limits the upper or lower limit of MV output by PID operation in automatic mode.
For further details, see Section 4.2.1 (4).
This function is only valid in automatic mode, and is invalid even in automatic mode when tracking function disable is specified in the parameter.

(3) PID macro function output process example

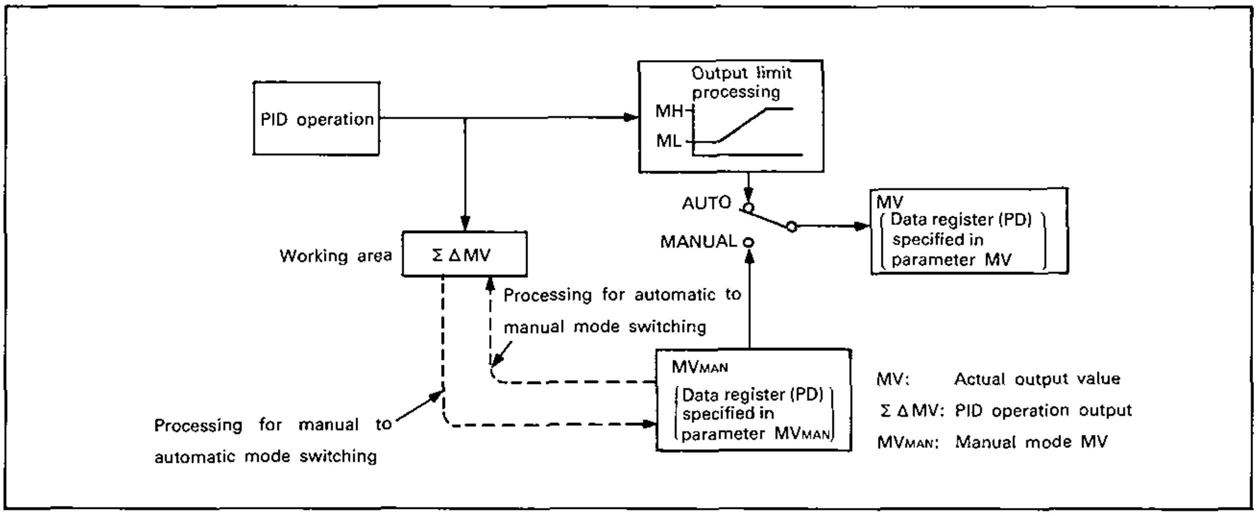


Fig. 4.6 PID Macro Function Output Process Example

4.2.4 Macro function parameters

No.	Symbol	Description	Setting Range	Accessibility	
				Read by "PRR"	Write by "PRW"
1	SV	Device for storing set value	PD0 to 1023	Yes	No
2	PV	Device for storing process value	PD0 to 1023	Yes	No
3	MV	Device for storing manipulated value	PD0 to 1023	Yes	No
4	MV_{MAN}	Device for storing manual manipulated value	PD0 to 1023	Yes	No
5	MODE	Device for storing mode switching	PD0 to 1023	Yes	No
6	—				
7	—				
8	ALARM	Device for setting alarm	PM0 to 1016	Yes	No
9	POL	PV change rate check direction	0: + 1: − 2: + & −	Yes	Yes
10	CTIM	PV change rate check duration	1 to 255	Yes	Yes
11	ACT	Action selection	0: Forward action, 1: Reverse action	Yes	Yes
12	PROG. NO.	Program number used with loop	1 to 32	Yes	Yes
13	—				
14	TR	Tracking function	0: Enable 1: Disable	Yes	Yes
15	—				
16	—				
17	PH	High limit alarm set value	0.00 to 100.00%	Yes	Yes
18	PL	Low limit alarm set value	0.00 to 100.00%	Yes	Yes
19	PH/PL HIS	High/low limit alarm hysteresis value	0.00 to 100.00%	Yes	Yes
20	DPVL	PV change rate alarm set value	0.00 to 100.00%	Yes	Yes
21	DPVL HIS	PV change rate check hysteresis value	0.00 to 100.00%	Yes	Yes
22	α	Filter coefficient	0.00 to 1.00	Yes	Yes
23	MH	MV high limit	−2.50 to 102.50%	Yes	Yes
24	ML	MV low limit	−2.50 to 102.50%	Yes	Yes
25	DMVL	MV change rate alarm set value	0.00 to 100.00%	Yes	Yes
26	EVL	Excessive error alarm set value	0.00 to 100.00%	Yes	Yes
27	KP	Proportional gain	0.01 to 100.00	Yes	Yes
28	TI	Integral time	0.01 to 32767.00s	Yes	Yes
29	TD	Derivative time	0.00 to 255.00s	Yes	Yes
30	αD	Derivative gain	0.00 to 1.00	Yes	Yes
31	—				
32	—				
33	—				
34	—				
35	—				
36	—				
37	—				
38	—				
39	—				
40	—				
41		Reads/writes $\Sigma \Delta MV$ of the specified loop No.	• PRR K⬚K41 ··· Stores specified loop No. $\Sigma \Delta MV$ to (A2). • PRW K⬚K41 ··· Stores data from (A2) to specified loop No. $\Sigma \Delta MV$.	Yes	Yes
42		Clears EV_{n-1}, PVf_{n-1}, PVf_{n-2}, $\Sigma \Delta MV$, ΔD_{n-1} of the specified loop No. to 0.	PRW K⬚K42	No	Yes

Table 4.1 Parameters

POINT

(1) The numbers (No.) in Table 4.1 are specified for the "PRR" and "PRW" instructions when parameters are accessed by the user programs.

(2) For numbers 1 to 5 and 8, a device number will be read by the "PRR" instruction. For other numbers, a set value will be read.

(3) Numbers 41 and 42 are not parameters. They are included in this table because they are used with the "PRR" and "PRW" instructions.

(1) SV (Set value)

(a) Devices PD0 to 1023 may be used.

(b) SV may be specified between 0.00 and 100.00%. Less than 0 is accounted for as 0.00 and more than 100 as 100.00.

(2) PV (Process value)

(a) Devices PD0 to 1023 may be used.

(b) PV may be specified between −2.50 and 102.50%. Less than −2.50 is accounted for as −2.50 and more than 102.50 as 102.50.

(3) MV (Manipulated value)

(a) Devices PD0 to 1023 may be used.

(b) MV output range may be specified between −2.50 and 102.50%.

(4) MV MAN (Manual manipulated value)

(a) Devices PD0 to 1023 may be used.

(b) MV MAN may be specified between −2.50 and 102.50%. Less than −2.50 is accounted for as −2.50 and more than 102.50 as 102.50.

(5) MODE (Mode switching)

(a) Devices PD0 to 1023 may be used.

(b) Specify 0 to select manual mode and 1 to select automatic mode. Mode is not switched if the value specified is other than 0 and 1.

(6) ALARM

Stores the check results of the Alarm check and Output areas.

(a) Devices PM0 to 1016 may be used.

(b) The following alarm check results are written to the group of eight devices headed by the specified device.

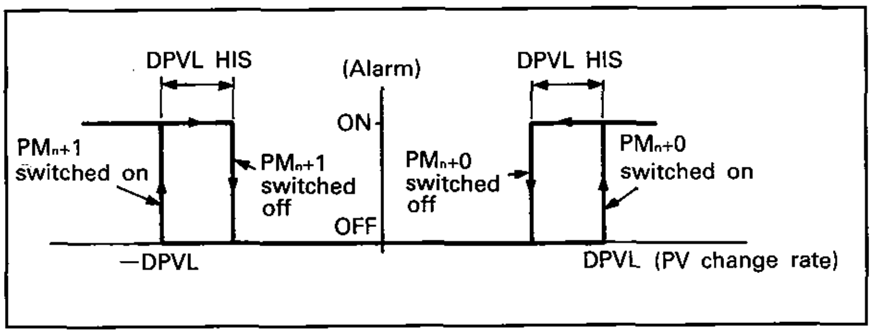
Device	Data
PM_n+0	PV change rate alarm positive area
PM_n+1	PV change rate alarm negative area
PM_n+2	PV high limit alarm
PM_n+3	PV low limit alarm
PM_n+4	Excessive error alarm
PM_n+5	MV change rate alarm
PM_n+6	MV high limit function
PM_n+7	MV low limit function

〔PV change rate alarm positive direction (PM$_n$+0), PV change rate alarm negative direction (PM$_n$+1)〕

Stores the result of PV change rate check made at intervals of the PV change rate check duration (CTIM).

1) PM$_n$+0 is switched on when the PV change rate changes greater than the PV change rate alarm setting (DPVL) in the positive direction.
   PM$_n$+0 is switched off when the PV change rate drops below (DPVL − DPVL HIS).

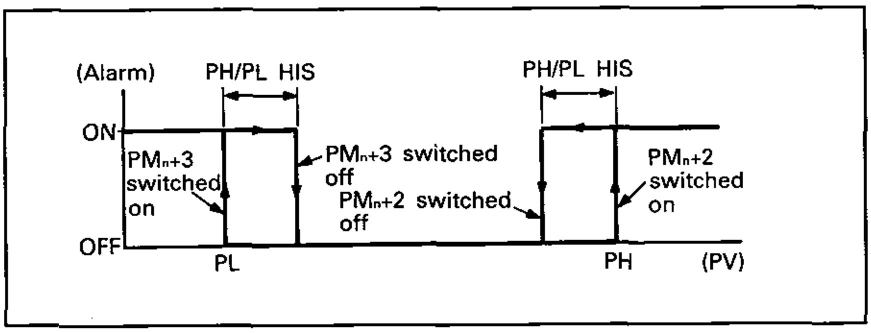
2) PM$_n$+1 is switched on when the PV change rate changes greater than DPVL in the negative direction.
   PM$_n$+1 is switched off when the PV change rate drops below (DPVL − DPVL HIS).



〔PV high limit alarm (PM$_n$+2), PV low limit alarm (PM$_n$+3)〕

1) PM$_n$+2 is switched on when PV exceeds the high limit alarm value (PH) and switched off when PV drops below (PH − PH/PL HIS).

2) PM$_n$+3 is switched on when PV drops below the low limit alarm value (PL) and switched off when PV exceeds (PL + PH/PL HIS).



〔Excessive error alarm ($PM_{n}+4$)〕

1) $PM_{n}+4$ is switched on when the error (PV-SV for forward action, SV-PV for backward action) is greater than the excessive error alarm set value (EVL).

2) $PM_{n}+4$ should be switched off by the user program.

〔MV change rate alarm ($PM_{n}+5$)〕

1) $PM_{n}+5$ is switched on when the MV change rate exceeds the MV change rate limit (DMVL).

2) $PM_{n}+5$ should be switched off by the user program.

〔MV high limit function ($PM_{n}+6$), MV low limit function ($PM_{n}+7$)〕

1) MV is adjusted to the MV high limit (MH) and $PM_{n}+6$ switched on when the operation result in the PID control area is greater than the MV high limit (MH).

2) MV is adjusted to the MV low limit (ML) and $PM_{n}+7$ switched on when the operation result in the PID control area is less than the MV low limit (ML).

3) $PM_{n}+6$ and $PM_{n}+7$ should be switched off by the user program.

(7) POL (PV change rate check direction)

0: Only checks the positive direction and stores the result to $PM_{n}+0$.
1: Only checks the negative direction and stores the result to $PM_{n}+1$.
2: Checks both directions and stores the results to $PM_{n}+0$ and $PM_{n}+1$.

(8) CTIM (PV change rate check duration)

(a) PV change rate is checked when the number of program executions reaches the value specified as CTIM.

(b) CTIM may be specified between 1 and 255.

(9) ACT (Action selection)

0: Forward action (NOR.)
1: Reverse action (REV.)

(10) TR (Tracking function enable/disable)

0: Valid
1: Invalid

(11) PH (High limit alarm set value)

Reference value for checking PM_n+2.

1) PH may be specified between 0.00 and 100.00%.

(12) PL (Low limit alarm set value)

Reference value for checking PM_n+3.

1) PL may be specified between 0.00 and 100.00%.

(13) PH/PL HIS (High/low limit alarm hysteresis values)

Prevents chattering by changing the reference values for switching on and off PM_n+2 and PM_n+3.

1) PH/PL HIS may be specified between 0.00 and 100.00%.

(14) DPVL (PV change rate alarm setting)

1) DPVL may be specified between 0.00 and 100.00%.

(15) DPVL HIS (PV change rate check hysteresis value)

Prevents chattering by changing the reference values for switching on and off PM_n+0 and PM_n+1.

1) DPVL HIS may be specified between 0.00 and 100.00%.

(16) α (Filter coefficient)

1) α may be specified between 0.00 and 1.00.

(17) MH (MV high limit)

1) MH may be specified between −2.50 and 102.50%.

(18) ML (MV low limit)

1) MH may be specified between −2.50 and 102.50%.

(19) DMVL (MV change rate alarm set value)

Reference value for checking PM_n+5.

1) DMVL may be specified between 0.00 and 100.00%.

(20) EVL (Excessive error alarm set value)

Reference value for checking PM_n+4.

1) EVL may be specified between 0.00 and 100.00%.

(21) KP (Proportional gain)

1) KP may be specified between 0.01 and 100.00.

(22) TI (Integral time)

1) TI may be specified between 0.01 and 32767.00 sec.

(23) TD (Derivative time)

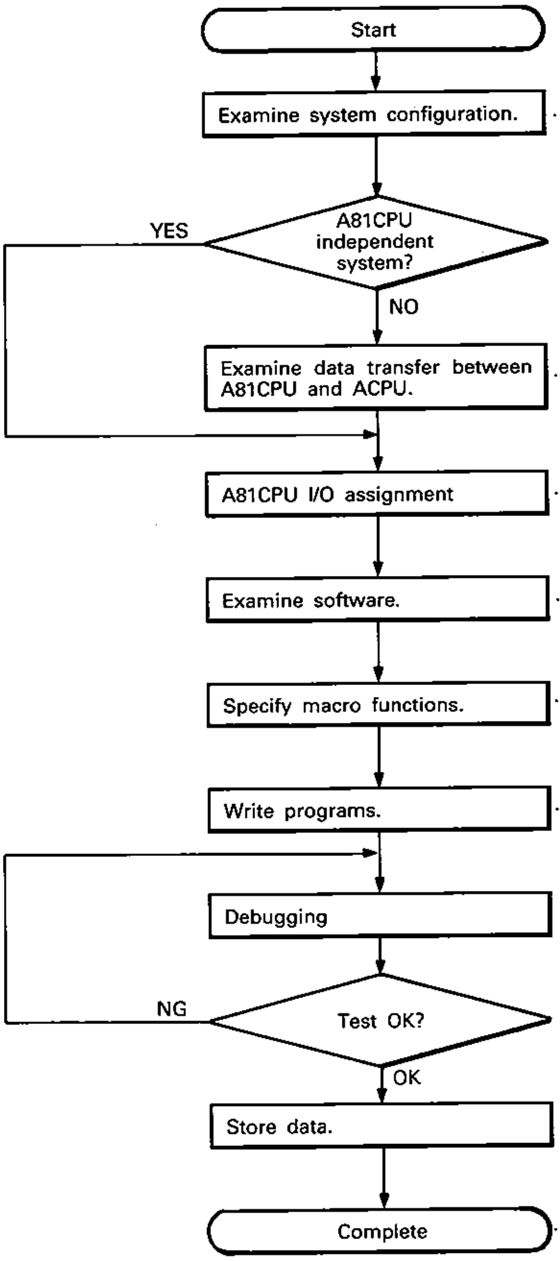
1) TD may be specified between 0.00 and 255.00 sec.

(24) αD (Derivative gain)

1) αD may be specified between 0.00 and 1.00.

5. PROGRAMS—GENERAL INFORMATION

5.1 Programming Procedure



Examine system configuration. ············ Consider whether the A81CPU is used independently or with the ACPU and determine the modules and I/O points used.

Examine data transfer between A81CPU and ACPU. ············ Determine A81CPU and ACPU software share, data transferred and its protocol.

A81CPU I/O assignment ············ Allocate the module I/O addresses to the A81CPU.

Examine software. ············ Determine the number of programs used, their capacities and macro functions.

Specify macro functions. ············ Specify the optional functions, parameters and sampling time.

Write programs. ············ Write programs from the peripheral to the A81CPU.

Complete ············ Write data onto disk.

POINT

In the A81CPU, PM0 to 1023, PD0 to 1023, PT0 to 127 and macro function work area (EV_{n-1}, PV_{fn-1}, PV_{fn-2}, $\Sigma \Delta MV$, ΔD_{n-1}) are battery backed. These areas should be initialized when starting to use the A81CPU.

(1) Clear PM0 to 1023, PD0 to 1023 and PT0 to 127 by the latch clear switch of the A81CPU.

(2) Clear the macro function work area by the PRW LOOP No. K42 instruction. For example, execute PRW ⊔ K5 ⊔ K41 to clear the macro function work area of loop 5.

5.2 Program Areas

5.2.1 Program area configuration

A capacity of 8000 steps is reserved for 32 program areas which are divided up as shown in Fig. 5.1. Up to 250 steps may be written to each area in order of program numbers, i.e. 1, 2, 3 ········· 32.

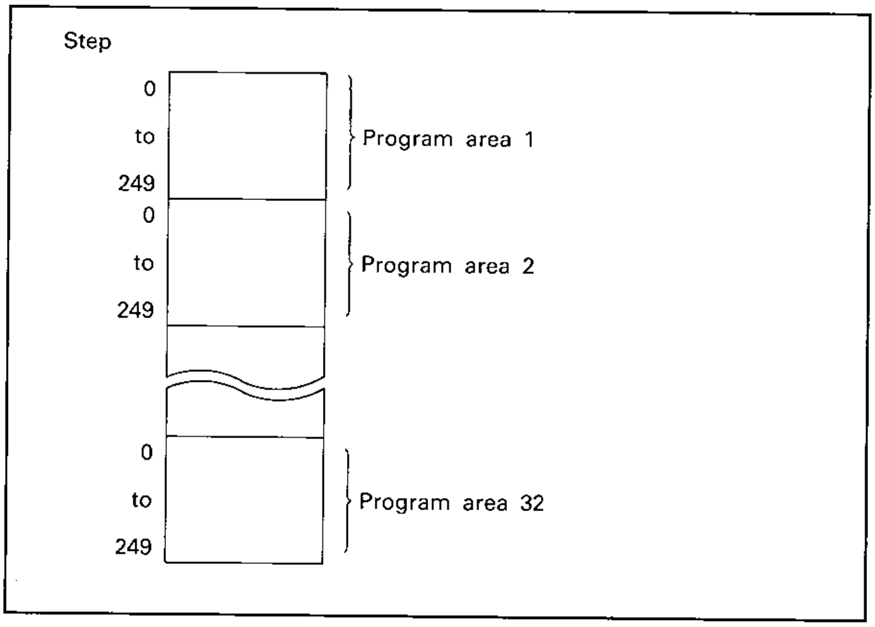


Fig. 5.1 Program Area Configuration

5.2.2 Program areas and operation processing

Several programs may be combined and processed as one program.

(1) Operation processing

A program start is effected when the sampling time is reached. When started, the program is executed from step 0 to the END instruction. After the END instruction, processing is held until the next program is started.

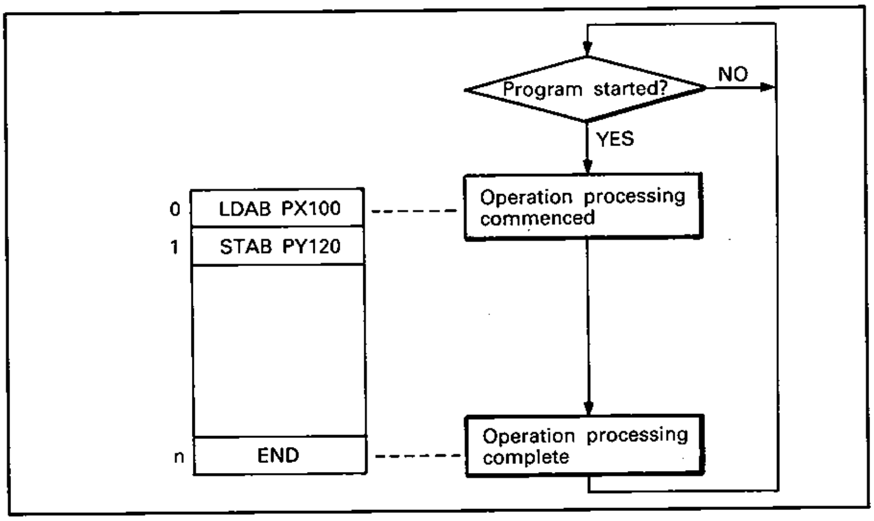
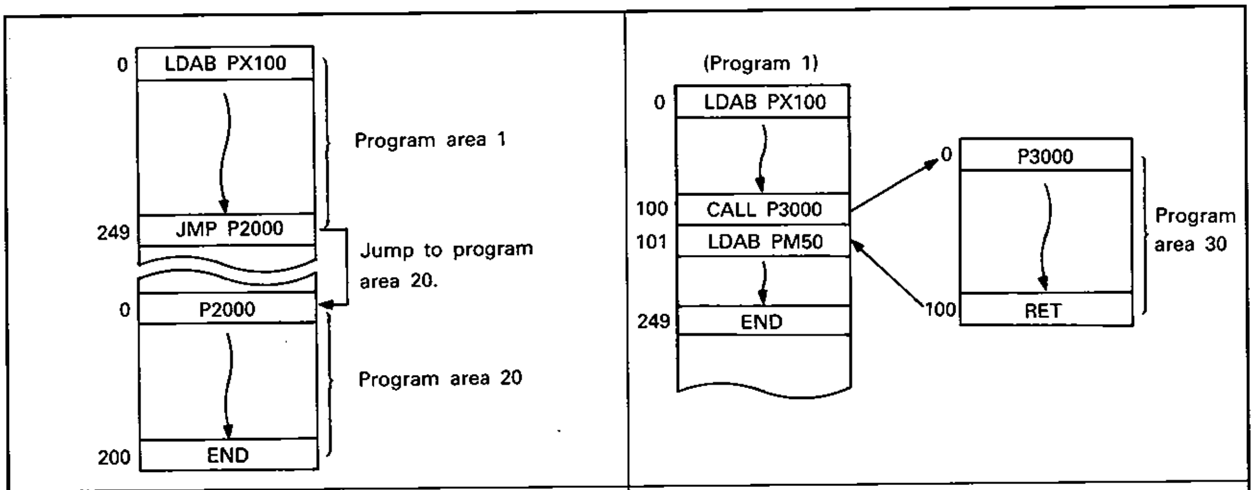


Fig. 5.2 Operation Processing

(2) Execution of one program in more than one program area

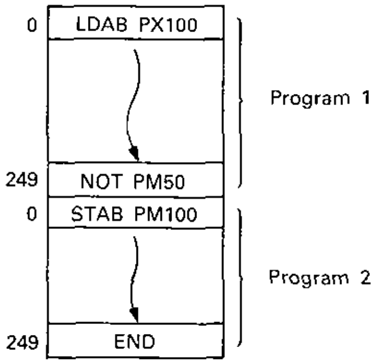
(a) Use a branch instruction (JMP, JC, CALL) to progress to another area as shown in Fig. 5.3.



The "JMP P2000" instruction in program area 1 causes execution to jump to pointer P2000 in program area 20. Operation processing is terminated by the start of program 1 when the "END" instruction is executed in program area 20.	The "CALL P3000" instruction in program area 1 causes execution to jump to pointer P3000 in program area 30. Operation processing is terminated by the start of program 1 when "END" is executed after execution is returned to program area 1 by the "RET" instruction in program area 30.

Fig. 5.3 Examples Using Branch Instructions

(b) The next program area may be used sequentially if the END instruction is not used in one program area as shown in Fig. 5.4.



The program in program area 2 is executed beginning with step 0 after step 249 of program area 1 as there is no "END" instruction in program area 1.
Operation processing is terminated by the start of the program in program area 1 when the "END" instruction is executed in program area 2.

Fig. 5.4 Sequentially Used Program Areas

POINT

The sampling time of the program executed sequentially should be set to 0 when one program is executed in more than one program area.

Setting the sampling time to other than 0 starts the corresponding program per sampling time.
For example, in Fig. 5.4, the sampling time of program area 2 should be set to 0 to execute programs 1 and 2 sequentially.

Program 2 is started and executed from step 0 to 249 at intervals of 1 second if the sampling time of program 2 is set to 1.00.

6. INSTRUCTIONS

6.1 Data Types

Data processed by operation includes bit data, word data and floating point data.

6.1.1 Bit data

Indicates the ON/OFF state as 1/0 in a bit device (PX, PY, PM, SP.PM, A0).

Example

LDAB PX100 ······ 1 is set to (A0) if PX100 is on and 0 is set to (A0) if PX100 is off.

6.1.2 Word data

Indicates 16-bit signed binary data or BCD (binary coded decimal) in a word device (SP.PD, T present value, A1) or 16 bit devices (PX, PY, PM, SP.PM).

POINT

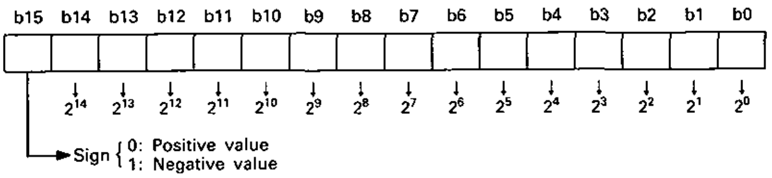
Any bit number specified must be a multiple of 16.

1) Input ·············· PX0, PX10, PX20 ····· PX2E0, PX2F0
2) Output ··········· PY0, PY10, PY20 ······ PY2E0, PY2F0
3) Internal relay ··· PM0, PM16, PM32 ··· PM992, PM1008
4) Special relay ··· PM9000, PM9016, PM9032 ················· PM480, PM496

(1) 16-bit signed binary data

−32768 to 32767 headed by a sign.

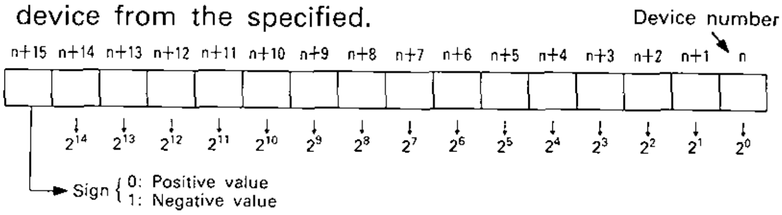
(a) When a word device is used, a sign is written to bit 15 (b15).



Example

		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
1234 stored in PD9200	PD9200	0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0
		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
−1234 stored in PD9200	PD9200	1	1	1	1	1	0	1	1	0	0	1	0	1	1	1	0

(b) When bit devices are used, a sign is written to the 16th bit device from the specified.



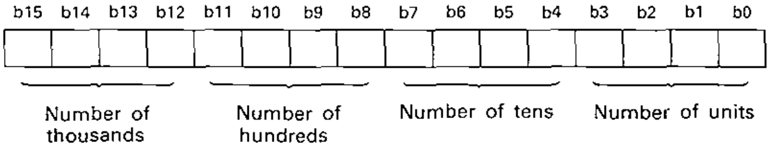
Example

	PM15	PM14	PM13	PM12	PM11	PM10	PM9	PM8	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
5432 stored in PM0 to 15	0	0	0	1	0	1	0	1	0	0	1	1	1	0	0	0
−5432 stored in PM0 to 15	1	1	1	0	1	0	1	0	1	1	0	0	1	0	0	0

(2) BCD data

Four BCD digits (0 to 9999) may be written.

(a) Word device used

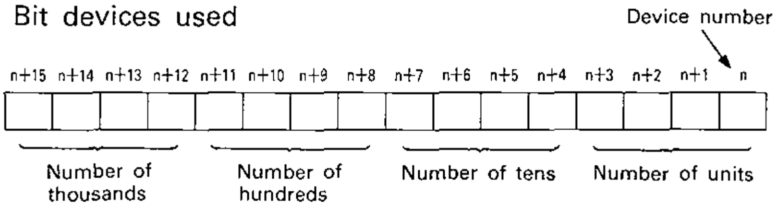


Example

1234 stored in PD9200 in BCD.

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PD9200	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0
	Number of thousands				Number of hundreds				Number of tens				Number of units			

(b) Bit devices used



Example

5432 stored in PM0 to 15 in BCD.

PM15	PM14	PM13	PM12	PM11	PM10	PM9	PM8	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0
0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0
Number of thousands				Number of hundreds				Number of tens				Number of units			

6.1.3 Floating-point data

Represents a fraction or a value outside the range −32768 to 32767 and may be specified between $\pm 2.7\times 10^{-20}$ and $\pm 9.2\times 10^{18}$. Any of the following instructions used with floating-point data converts the data format in accordance with the combination of devices used.

(1) Instructions which convert floating-point data into 16-bit binary data (MOV, TO, FROM)

  (a) Data between −32768 and 32767 is converted into 16-bit binary data.
  Any data outside the above range cannot be converted without fault as the floating point data is converted into binary 32 bits and the lower 16 bits are used.
  Any value outside the range −2147483648 and 2147483647 results in an operation error.

  (b) The fraction part of any floating-point data is omitted.

(Example)

MOV | PD0 | PD9200 causes 1234 to be transferred to PD9200 when 1234.5 exists in PD0.

PD0 1234.5 (Floating-point data)

(16-bit binary data)

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PD9200	0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0

(2) Instruction which converts floating-point data into 4 BCD digits (BCD)

  (a) Data between 0 and 9999 is converted into four BCD digits. Any value outside the above range results in an operation error.

  (b) The fraction part of any floating-point data is omitted.

(Example)

BCD | PD0 | PD9200 causes 5432 to be transferred to PD9200 when 5432.1 exists in PD0.

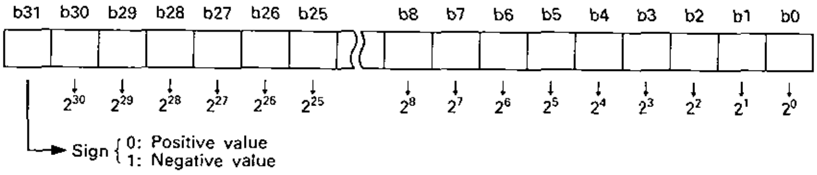
PD0 5432.1 (Floating-point data)

(BCD data)

	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PD9200	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0
	Number of thousands				Number of hundreds				Number of tens				Number of units			

(3) Instruction which converts floating-point data into 32-bit signed binary data (DTO)

(a) Converts data between −2147483648 and 2147483647 into 32-bit binary data.
Any data outside the above range results in an error.

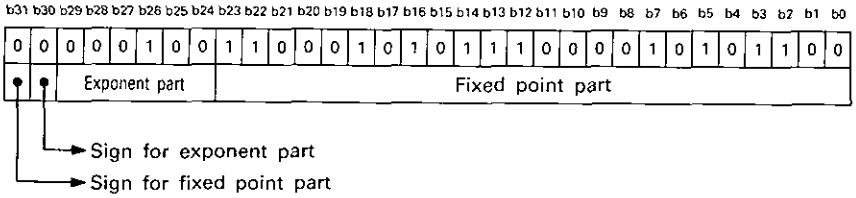


(4) Instruction which converts 32-bit signed binary data into floating-point data (DFRO)

(a) Converts data between −2147483648 and 2147483647 into floating point data.

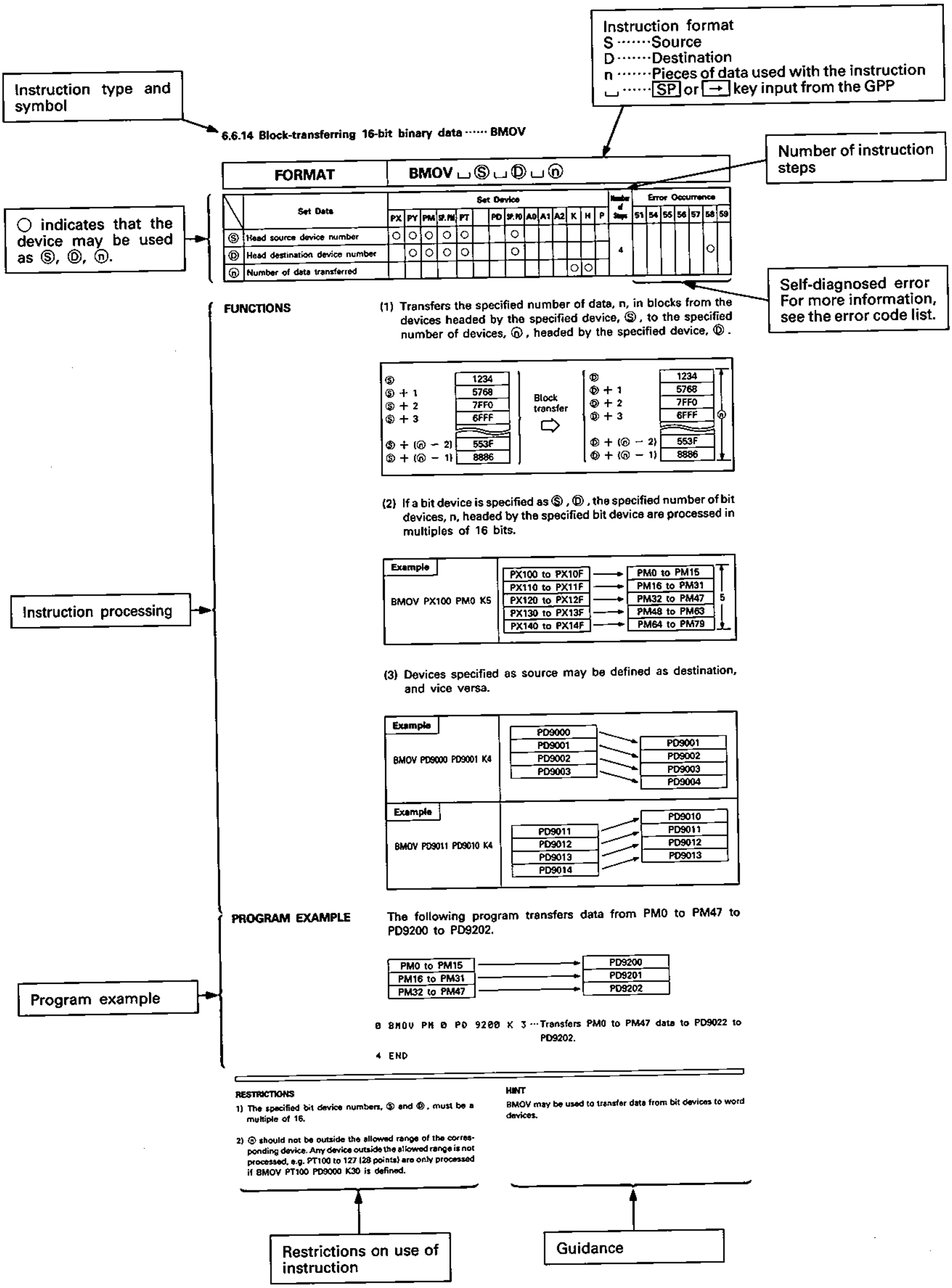
REMARKS

The floating-point data has 32 bit locations as shown below.



6.2 Guide to Sections 6.3 to 6.12

Sections 6.3 to 6.12 use the format below.



6.6.14 Block-transferring 16-bit binary data ······ BMOV

FORMAT	BMOV ␣ Ⓢ ␣ Ⓓ ␣ ⓝ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.M	PT		PD	SP.D	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Head source device number	○	○	○	○	○			○							4						○	
Ⓓ	Head destination device number		○	○	○	○			○														
ⓝ	Number of data transferred												○	○									

FUNCTIONS

(1) Transfers the specified number of data, n, in blocks from the devices headed by the specified device, Ⓢ, to the specified number of devices, ⓝ, headed by the specified device, Ⓓ.

(2) If a bit device is specified as Ⓢ, Ⓓ, the specified number of bit devices, n, headed by the specified bit device are processed in multiples of 16 bits.

(3) Devices specified as source may be defined as destination, and vice versa.

PROGRAM EXAMPLE

The following program transfers data from PM0 to PM47 to PD9200 to PD9202.

```
0 BMOV PM 0 PD 9200 K 3 ···Transfers PM0 to PM47 data to PD9022 to
                           PD9202.
4 END
```

RESTRICTIONS

1) The specified bit device numbers, Ⓢ and Ⓓ, must be a multiple of 16.

2) ⓝ should not be outside the allowed range of the corresponding device. Any device outside the allowed range is not processed, e.g. PT100 to 127 (28 points) are only processed if BMOV PT100 PD9000 K30 is defined.

HINT

BMOV may be used to transfer data from bit devices to word devices.

6.3 Logic Instructions

The logic instructions may be used for bit devices (PX, PY, PM, SP.PM), word devices (SP.PD) and word data (K, H).

Instruction	Description	Refer To
NOT	Complements the specified bit device data, Ⓢ, and stores to (A0). $\overline{\text{Ⓢ}}$ → (A0)	Section 6.3.1
WNOT	Complements the specified word device data or word data, Ⓢ, and stores to (A1). $\overline{\text{Ⓢ}}$ → (A1)	Section 6.3.2
AND	ANDs the specified bit device data, Ⓢ, and (A0) data and stores the operation result to (A0). (A0) ∧ Ⓢ → (A0)	Section 6.3.3
WAND	ANDs the specified word device data or word data, Ⓢ, and (A1) data and stores the operation result to (A1). (A1) ∧ Ⓢ → (A1)	Section 6.3.4
OR	ORs the specified bit device data, Ⓢ, and (A0) data and stores the operation result to (A0). (A0) ∨ Ⓢ → (A0)	Section 6.3.5
WOR	ORs the specified word device data or word data, Ⓢ, and (A1) data and stores the operation result to (A1). (A1) ∨ Ⓢ → (A1)	Section 6.3.6
XOR	EXCLUSIVE ORs the specified bit device data, Ⓢ, and (A0) data and stores the operation result to (A0). (A0) ⊻ Ⓢ → (A0)	Section 6.3.7
WXOR	EXCLUSIVE ORs the specified word device data or word data, Ⓢ, and (A1) data and stores the operation result to (A1). (A1) ⊻ Ⓢ → (A1)	Section 6.3.8

6.3.1 Complementing 1-bit data ······ NOT

FORMAT	NOT ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Bit device number complemented	○	○	○	○																	○	

FUNCTIONS

(1) Complements the specified bit device data, Ⓢ, and stores the result to accumulator (A0).

	Ⓢ	$\overline{Ⓢ}$→(A0)
$\overline{Ⓢ}$ ——→ (A0)	0	1
	1	0

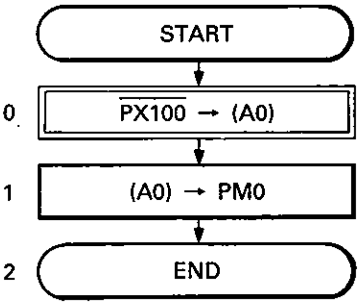
(2) The specified bit device data, Ⓢ, remains unchanged after the NOT instruction is executed.

REMARKS

The (A0) data is overwritten by the NOT execution result and therefore should be saved before NOT is executed if its data is required.

PROGRAM EXAMPLE

The following program complements PX100 ON/OFF data and stores the result to PM0.



```
0 NOT   PX 100·························Complements PX100 ON/OFF data and stores
                                       to (A0).
1 STAB PM 0····························Stores (A0) data to PM0.
2 END
```

6.3.2 Complementing 16-bit data …… WNOT

FORMAT	WNOT ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Word device number or constant complemented								○				○	○		1						○	

FUNCTIONS

(1) Complements the specified word device data or constant, Ⓢ, for 16 bits and stores the result to accumulator (A1).

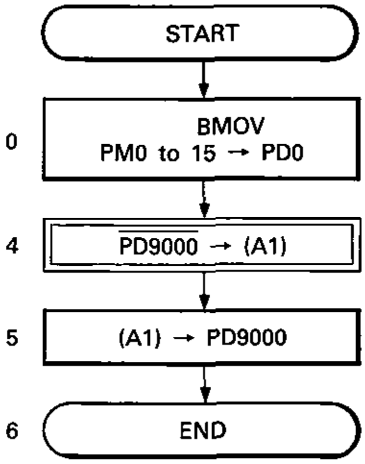
$\overline{\text{Ⓢ}}$ → (A1)	Before execution Ⓢ 0 0 0 0 1 1 1 1 0 1 0 1 0 1 0 1 ↓ $\boxed{\text{WNOT instruction}}$ After execution (A1) 1 1 1 1 0 0 0 0 1 0 1 0 1 0 1 0

(2) The specified word device data, Ⓢ, remains unchanged after the $\boxed{\text{WNOT}}$ instruction is executed.

REMARKS

The (A1) data is overwritten by the $\boxed{\text{WNOT}}$ execution result and therefore should be saved before $\boxed{\text{WNOT}}$ is executed if its data is required.

PROGRAM EXAMPLE The following program complements PM0 to 15 data and stores the result to PD9000.



```
0 BMOV PM 0    PD 9000   K 1···············Stores PM0 to 15 data to PD9000.
4 WNOT PD 9000 ······························Complements PD9000 data and
                                             stores the result to (A1).
5 STAW PD 9000 ······························Stores (A1) data to PD9000.
6 END
```

6.3.3 ANDing 1-bit data ······ AND

FORMAT	**AND ␣ Ⓢ**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Bit device number ANDed	○	○	○	○											1						○	

FUNCTIONS

(1) ANDs the specified bit device data, Ⓢ, and (A0) data, and stores the operation result to (A0).

	(A0)	Ⓢ	(A0)∧Ⓢ→(A0)
(A0)∧Ⓢ→(A0)	0	0	0
	0	1	0
	1	0	0
	1	1	1

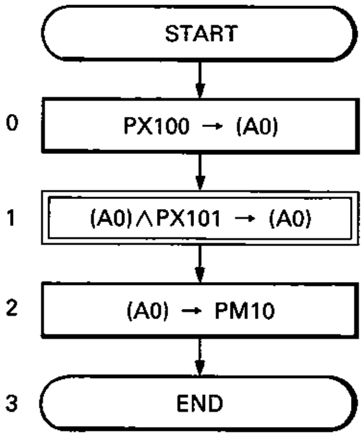
(2) The specified bit device data, Ⓢ, remains unchanged after the AND instruction is executed.

REMARKS

The (A0) data is overwritten by the AND execution result and therefore should be saved before AND is executed if the data is required.

PROGRAM EXAMPLE

The following program ANDs PX100 and PX101 data and stores the result to PM10.



```
0 LDAB PX 100··································Reads PX100 data to (A0).
1 AND  PX 101··································ANDs (A0) and PX101 data and
                                               stores the result to (A0).
2 STAB PM 10···································Stores (A0) data to PM10.
3 END
```

6.3.4 ANDing 16-bit data ······ WAND

FORMAT	WAND ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Word device number or constant ANDed								○				○	○		1						○	

FUNCTIONS

(1) ANDs the specified word device data or constant, Ⓢ, and (A1) data for all 16 bits and stores the operation result to (A1).

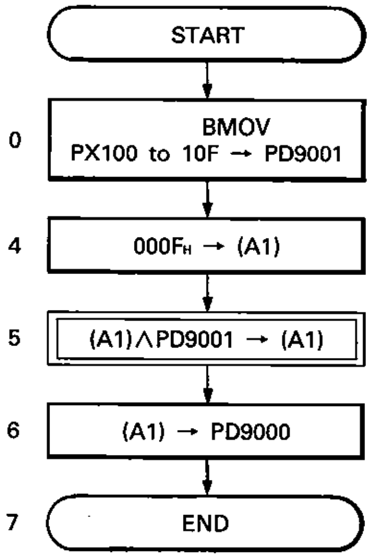
(A1) ∧ Ⓢ → (A1)	Before execution	(A1)	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
		Ⓢ	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
			↓ WAND instruction															
	After execution	(A1)	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

(2) The specified word device data, Ⓢ, remains unchanged after the WAND instruction is executed.

REMARKS

The (A1) data is overwritten by the WAND execution result and therefore should be saved before WAND is executed if the data is required.

PROGRAM EXAMPLE The following program stores PX100 to 10F data to PD9000.



```
0 BMOV PX 100     PD 9001    K 1·······Stores PX100 to data to PD9001.
4 LDAW   H 000F·····························Stores 000FH to (A1).
5 WAND PD 9001······························ANDs (A1) and PD9001 data.
6 STAW PD 9000······························Stores (A1) data to PD9000.
7 END
```

6.3.5 ORing 1-bit data …… OR

FORMAT	OR ⊔ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Bit device number ORed	○	○	○	○											1						○	

FUNCTIONS

(1) ORs the specified bit device data, Ⓢ, and (A0) data, and stores the operation result to (A0).

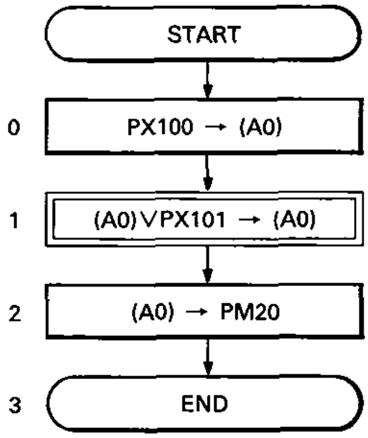
	(A0)	Ⓢ	(A0)∨Ⓢ→(A0)
(A0)∨Ⓢ→(A0)	0	0	0
	0	1	1
	1	0	1
	1	1	1

(2) The specified bit device data, Ⓢ, remains unchanged after the OR instruction is executed.

REMARKS

The (A0) data is overwritten by the OR execution result and therefore should be saved before OR is executed if the data is required.

PROGRAM EXAMPLE The following program ORs PX100 and PX101 data and stores the result to PM20.



```
0 LDAB PX 100 ······································Reads PX100 data to (A0).
1 OR   PX 101 ······································ORs (A0) and PX101 data and
                                                    stores the result to (A0).
2 STAB PM 20 ·······································Stores (A0) data to PM20.
3 END
```

6.3.6 ORing 16-bit data ······ WOR

FORMAT	WOR ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Word device number or constant ORed								○				○	○		1						○	

FUNCTIONS

(1) ORs the specified word device data or constant, Ⓢ , and (A1) data for all 16 bits and stores the operation result to (A1).

(A1) V Ⓢ → (A1)	Before execution	(A1)	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
		Ⓢ	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
			↓ WOR instruction
	After execution	(A1)	0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1

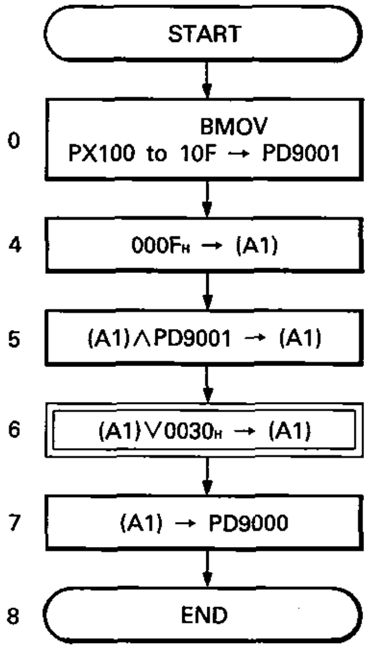
(2) The specified word device data, Ⓢ , remains unchanged after the WOR instruction is executed.

REMARKS

The (A1) data is overwritten by the WOR execution result and therefore should be saved before WOR is executed if the data is required.

PROGRAM EXAMPLE

The following program ORs PX100 to 10F and 0030HEX data and stores the result to PD9000.



Stores PX100 to 10F data to (A1).

```
0 BMOV PX 100   PD 9001    K  1 ·········Stores  PX100  to  10F  data  to
                                         PD9001.
4 LDAW   H 000F··························Stores 000FH to (A1).
5 WAND PD 9001···························ANDs (A1) and PD9001 data and
                                         stores PX100 to 10F data to (A1).
6 WOR    H 0030··························ORs (A1) and 0030HEX data.
7 STAW PD 9000···························Stores (A1) data to PD9000.
8 END
```

6.3.7 EXCLUSIVE ORing 1-bit data …… XOR

FORMAT	**XOR ␣ Ⓢ**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Bit device number EXCLUSIVE ORed	○	○	○	○											1						○	

FUNCTIONS

(1) EXCLUSIVE ORs the specified bit device data, Ⓢ, and (A0) data, and stores the operation result to (A0).

	(A0)	Ⓢ	(A0) ⊻ Ⓢ → (A0)
(A0)⊻Ⓢ→(A0)	0	0	0
	0	1	1
	1	0	1
	1	1	0

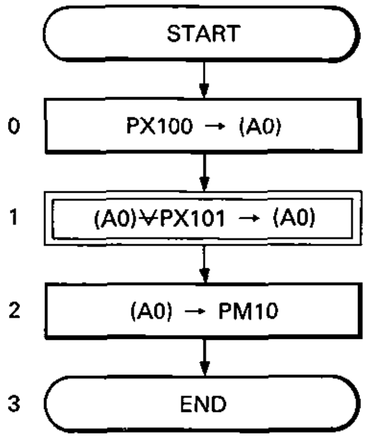
(2) The specified bit device data, Ⓢ, remains unchanged after the XOR instruction is executed.

REMARKS

The (A0) data is overwritten by the XOR execution result and therefore should be saved before XOR is executed if the data is required.

PROGRAM EXAMPLE

The following program EXCLUSIVE ORs PX100 and PX101 data and stores the result to PM10.



```
0 LDAB PX 100 ······································Stores PX100 data to (A0).
1 XOR  PX 101 ······································EXCLUSIVE ORs (A0) and PX101
                                                    data.
2 STAB PM 10  ······································Stores (A0) data to PM10.
3 END
```

6.3.8 ORing 16-bit data ······ WXOR

FORMAT	WXOR ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Word device number or constant EXCLUSIVE ORed								○				○	○		1						○	

FUNCTIONS

(1) EXCLUSIVE ORs the specified word device data or constant, Ⓢ, and (A1) data for all 16 bits and stores the operation result to (A1).

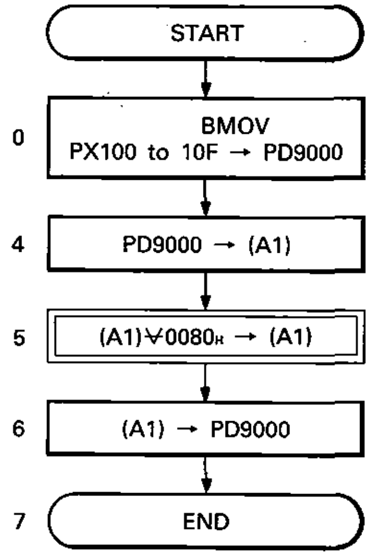
(A1) ⊻ Ⓢ → (A1)	Before execution	(A1)	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
		Ⓢ	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
			↓ WXOR instruction
	After execution	(A1)	0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0

(2) The specified word device data, Ⓢ, remains unchanged after the WXOR instruction is executed.

REMARKS

The (A1) data is overwritten by the WXOR execution result and therefore should be saved before WXOR is executed if the data is required.

PROGRAM EXAMPLE The following program transfers PX100 to 10F data to PD9000 and complements bit "b7" of PD9000.

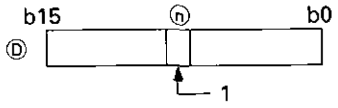
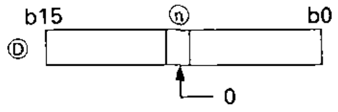


EXCLUSIVE ORs (A1) and 0080H data and complements bit "b7".

```
0 BMOV PX 100  PD 9000   K 1············Transfers PX100 to 10F data to
                                        PD9000.
4 LDAW PD 9000··························Stores PD9000 data to (A1).
5 WXOR  H 0080··························Complements bit "b7" of (A1).
6 STAW PD 9000··························Stores (A1) data to PD9000.
7 END
```

6.4 Bit Set/Reset Instructions

Used to set/reset the bit devices (PY, PM, SP.PM, PT) and word devices (SP.PD, A1).

Instruction	Description	Refer To
SET	Switches on the specified bit device, Ⓓ. 1 → Ⓓ	Section 6.4.1
RST	Switches off the specified bit device, Ⓓ. 0 → Ⓓ	Section 6.4.2
BSET	Switches on the specified bit, ⓝ, of the specified word device, Ⓓ. 	Section 6.4.3
BRST	Switches off the specified bit, ⓝ, of the specified word device, Ⓓ. 	Section 6.4.4

6.4.1 Setting the device ······ SET

FORMAT	SET ␣ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓓ	Device number set (switched on)		○	○	○	○										1						○	

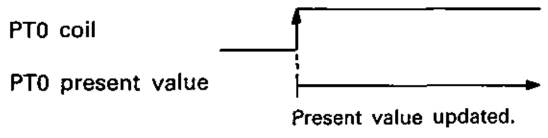
FUNCTIONS

(1) Switches on the specified bit device, Ⓓ.

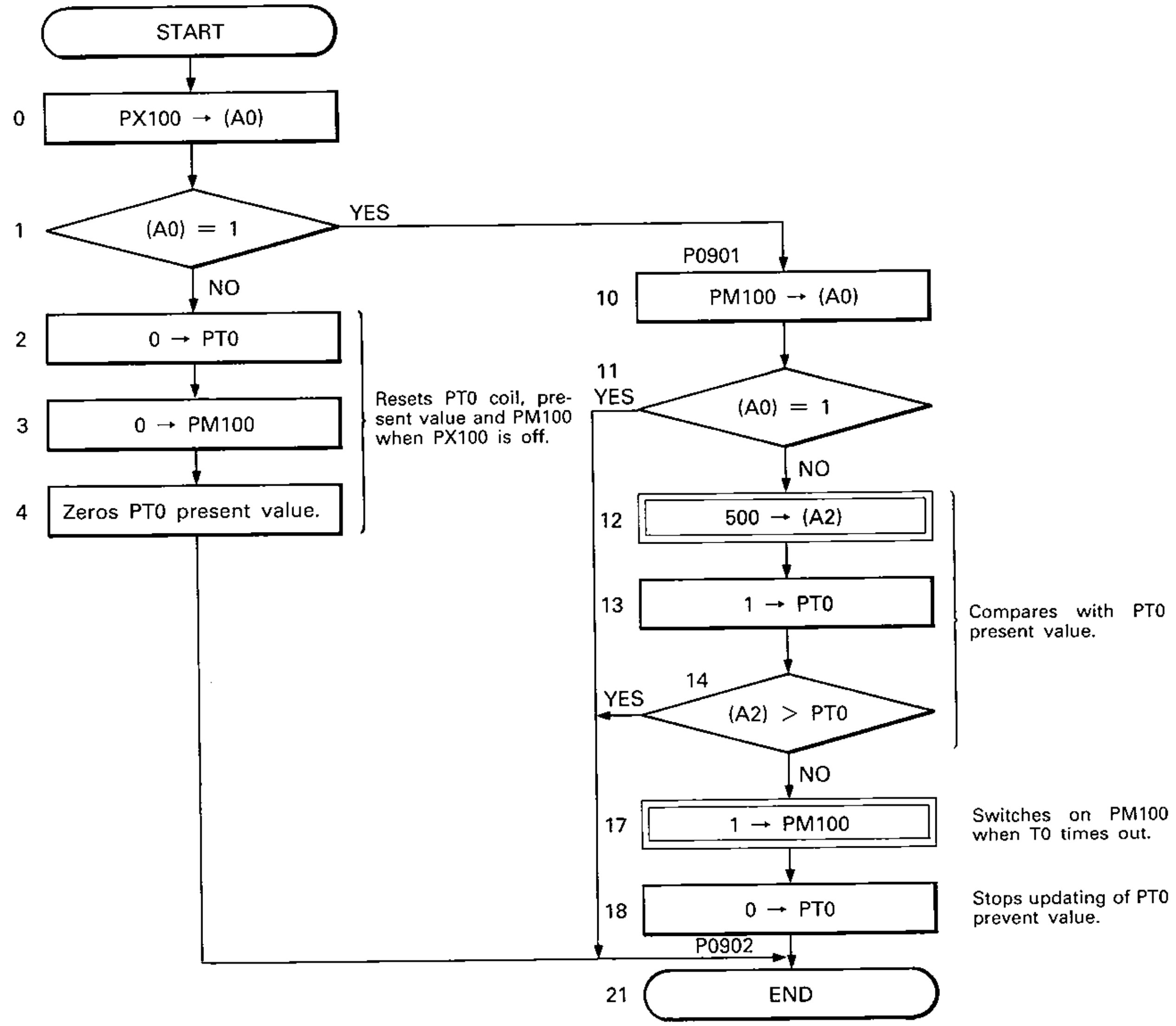
1 → Ⓓ

(2) If PT is specified as Ⓓ, the coil of that PT is switched on and the timer present value is updated. For further details, see Section 3.6.

SET PT 0



PROGRAM EXAMPLE The following program switches on the PT0 coil when PX100 is switched on and switches on PM100 five seconds later. (Program 9 used)



```
 0 LDAB PX 100··························Reads PX100 data to (A0).
 1 JC     P 0901························Judges the ON/OFF state of
                                        PX100.
 2 RST   PT 0···························Switches off PT0 coil.
 3 RST   PM 100·························Switches off PM100.
 4 MOV    K 0           PT 0············Zeroes PT0 present value.
 7 JMP    P 0902························Jumps to P0902.
 8 P     0901···························Pointer P0901.
10 LDAB PM 100··························Reads PM100 data to (A0).
11 JC     P 0902························Judges the ON/OFF state of
                                        PM100.
12 LDAW   K 500·························Sets 500 to (A1).
13 SET   PT 0···························Switches on PT0 coil.
14 GTAW PT 0····························Compares (A1) data with PT0
16 JMP    P 0902························value.
17 SET   PM 100·························Jumps to P0902.
18 RST   PT 0···························Switches on PM100.
19 P     0902···························Switches off PT0 coil.
21 END                                  Pointer P0902.
```

6

6.4.2 Resetting the device …… RST

FORMAT	RST ␣ Ⓓ

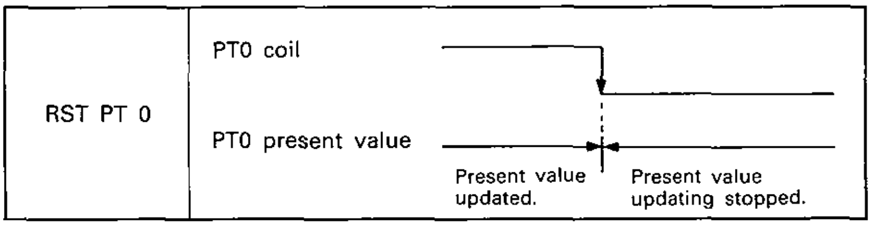
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓓ	Device number reset (switched off)		○	○	○	○										1						○	

FUNCTIONS

(1) Switches off the specified bit device, Ⓓ.

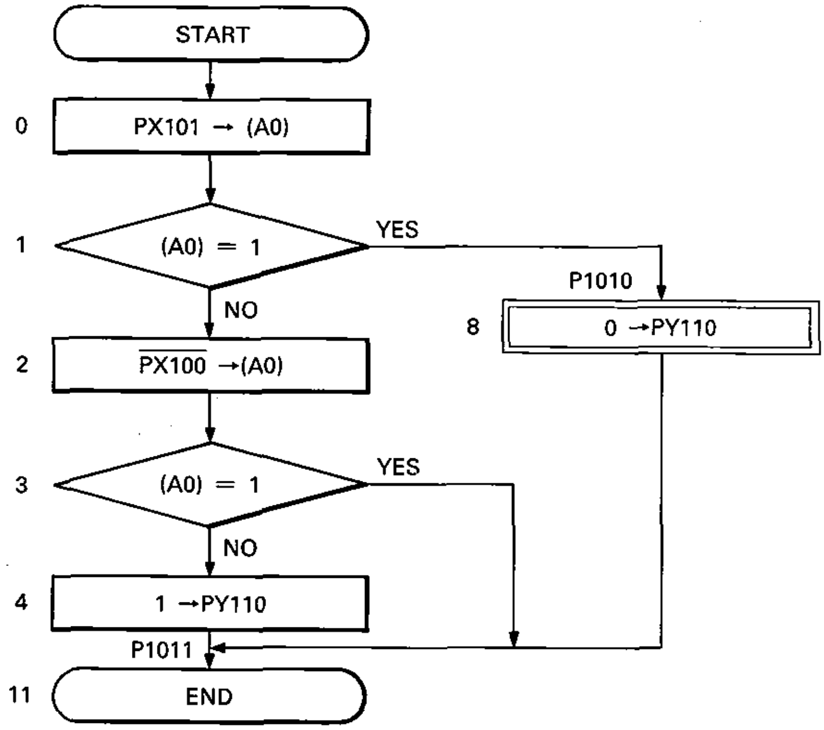
0 → Ⓓ

(2) If PT is specified in Ⓓ, the coil of that PT is switched off. The timer present value is retained if the PT coil is switched off.
For further details, see Section 3.6.



PROGRAM EXAMPLE

The following program switches on PY110 when PX100 is switched on and switches off PY110 when PX101 is switched on. (Program 10 used)



```
 0 LDAB PX 101 ···································Reads PX101 data to (A0).
 1 JC     P 1010 ·································Judges the ON/OFF state of
                                                  PX101.
 2 NOT  PX 100 ···································Complements PX100 data and
                                                  reads the result to (A0).
 3 JC     P 1011 ·································Judges the ON/OFF state of
                                                  PX100.
 4 SET  PY 110 ···································Switches on PY110.
 5 JMP    P 1011 ·································Jumps to pointer P1011.
 6 P      1010
 8 RST  PY 110 ···································Switches off PY110.
 9 P      1011 ··································Pointer P1011.
11 END
```

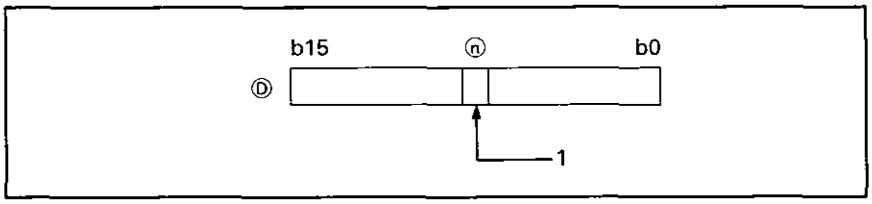
6.4.3 Setting the word device bit ...... BSET

FORMAT	BSET ␣ Ⓓ ␣ ⓝ

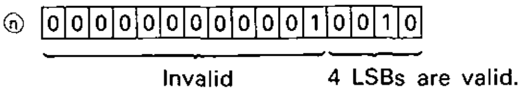
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓓ	Device number specified								○		○					3						○	
ⓝ	Bit number set (switched on)												○	○									

FUNCTIONS

(1) Switches on the specified bit, ⓝ, of the specified word device, Ⓓ.



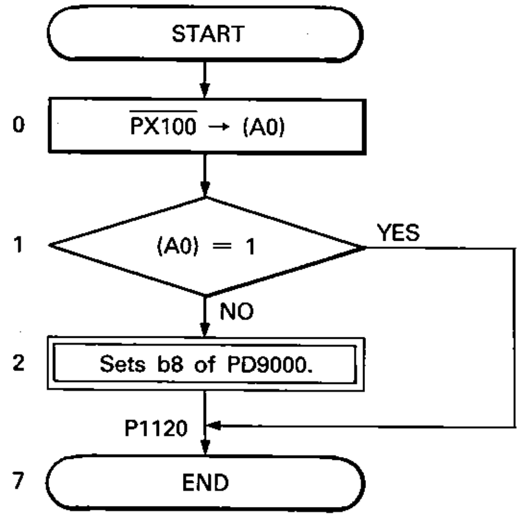
(2) ⓝ should be between 0 and 15.
Any ⓝ value over 15 is converted into a binary and its 4 least significant bits (LSB) are valid, e.g. 18 is regarded as 2.



PROGRAM EXAMPLE

The following program switches on b8 of PD9000 when PX100 is switched on. (Program 11 used)

		b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
BSET PD9000 K8	Before execution	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
								↓		BSET instruction							
	After execution	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	0



```
0 NOT  PX 100 ·······························Compensates PX100 data and
                                             reads the result to (A0).
1 JC    P 1120 ······························Judges the ON/OFF state of
                                             PX100.
2 BSET PD 9000   K 8 ························Switches on b8 of PD9000 when
                                             PX100 is switched on.
5 P     1120 ································Pointer P1120.
7 END
```

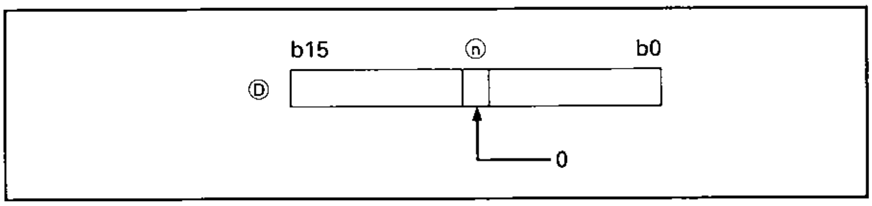
6.4.4 Resetting the word device bit ······ BRST

FORMAT	BRST ␣ Ⓓ ␣ ⓝ

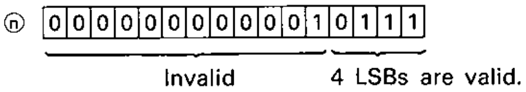
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓓ	Device number specified								○		○					1						○	
ⓝ	Bit number reset (switched off)												○	○									

FUNCTIONS

(1) Switches off the specified bit, ⓝ , of the specified word device, Ⓓ .

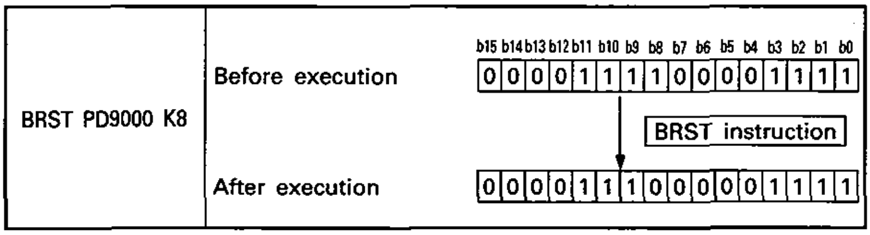


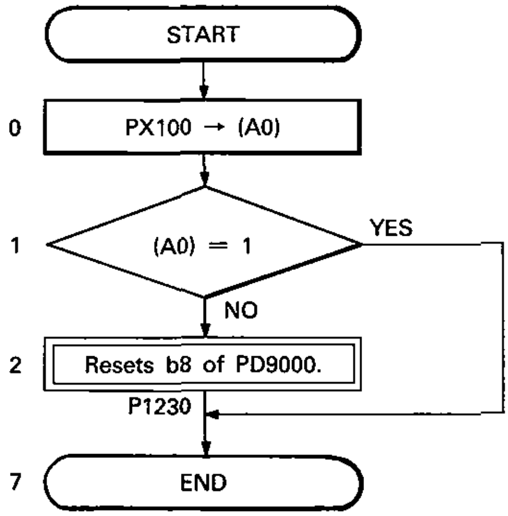
(2) ⓝ should be between 1 and 15.
Any ⓝ value over 15 is converted into a binary and its 4 least significant bits (LSB) are valid, e.g. 23 is regarded as 7.



PROGRAM EXAMPLE

The following program switches off b8 of PD9000 when PX100 is switched off. (Program 12 used)





```
0 LDAB PX 100······························Reads PX100 data to (A0).
1 JC     P 1230····························Judges the ON/OFF state of
                                           PX100.
2 BRST PD 9000     K 8·····················Switches off b8 of PD9000 when
                                           PX100 is switched off.
5 P     1230·······························Pointer P1230.
7 END
```

MEMO

6.5 BCD ←→ BIN Conversion Instructions

Converts 16-bit binary data or floating-point data into 4-digit BCD data and vice versa.

Instruction	Description	Refer To
BCD	Converts the specified 16-bit binary data, Ⓢ, into 4-digit BCD data and transfers the result to the specified device, Ⓓ. Ⓢ $\xrightarrow{\text{BCD conversion}}$ Ⓓ (16-bit binary data)	Section 6.5.1
	Converts the specified floating-point data, Ⓢ, into 4-digit BCD data and transfers the result to the specified device, Ⓓ. Ⓢ $\xrightarrow{\text{BCD conversion}}$ Ⓓ (Floating-point data)	Section 6.5.2
BIN	Converts the specified 4-digit BCD data, Ⓢ, into 16-bit binary data and transfers the result to the specified device, Ⓓ. Ⓢ $\xrightarrow{\text{BIN conversion}}$ Ⓓ (16-bit binary data)	Section 6.5.3
	Converts the specified 4-digit BCD data, Ⓢ, into floating-point data and transfers the result to the specified device, Ⓓ. Ⓢ $\xrightarrow{\text{BIN conversion}}$ Ⓓ (Floating-point data)	Section 6.5.4

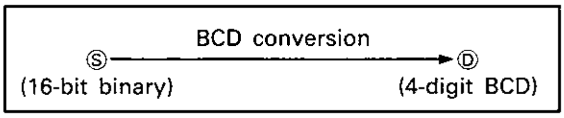
6.5.1 BCD conversion instruction (16-bit binary to 4-digit BCD) ······ BCD

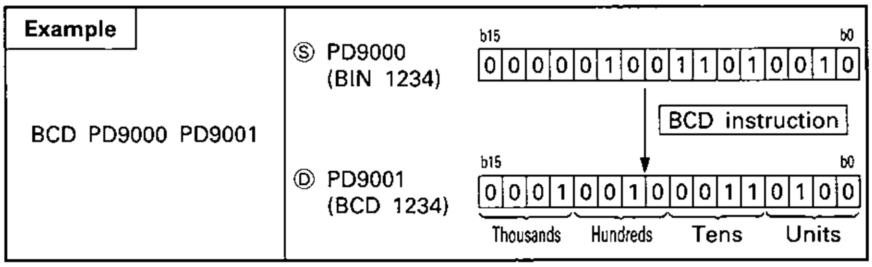
FORMAT	BCD ␣ Ⓢ ␣ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number containing BIN data	○	○	○	○	○			○							3						○	
Ⓓ	Device number for storing BCD data		○	○	○	○			○		○												

FUNCTIONS

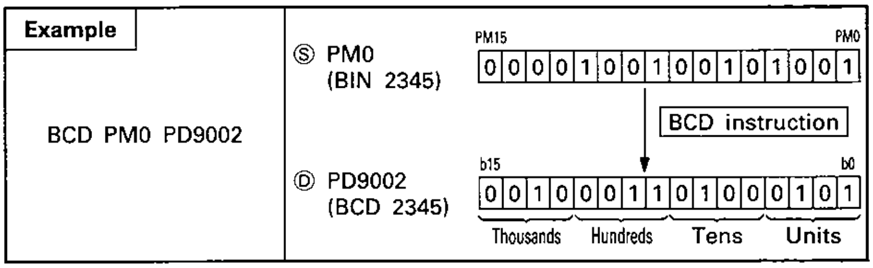
(1) Converts the specified 16-bit binary data (0 to 9999), Ⓢ, into BCD and transfers the result to the specified device, Ⓓ.




Example

BCD PD9000 PD9001

(2) If a bit device is specified as Ⓢ, 16 bits headed by the specified bit device are treated as binary data.


Example

BCD PM0 PD9002

RESTRICTIONS

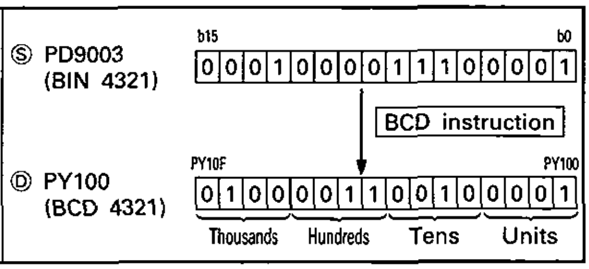
1) Any bit device (PX, PY, PM, SP.PM) number specified as Ⓢ and Ⓓ must be a multiple of 16.

2) Any value between 0 and 9999 may be converted into BCD.

(3) If a bit device is specified as Ⓓ, 4-digit BCD data is transferred to 16 bits headed by the specified bit device.

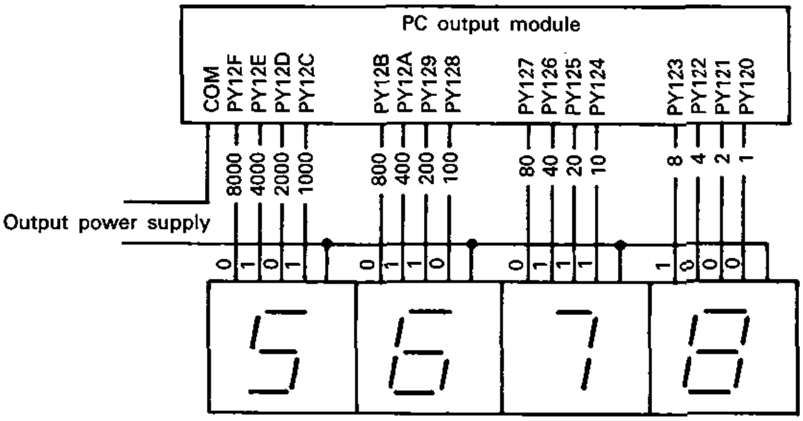
Example

BCD PD9003 PY100



PROGRAM EXAMPLE

The following program outputs the PT0 present value from PY120 to 12F to the BCD display.



7-segment display

```
0 BCD   PT 0        PY 120······Converts PT0 present value into BCD and
3 END                           outputs to Y120 to 12F.
```

HINT

16 points headed by PY□□0 are used when specifying a bit device as Ⓓ for the BCD instruction. To output data to the BCD display by the BCD instruction, cables should be wired so that the number of units indicated on the BCD display may be output to PY□□0 to □□3.

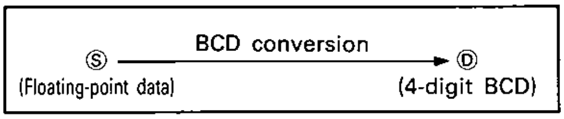
6.5.2 BCD conversion instruction (floating-point data to 4-digit BCD) ······ BCD

FORMAT	**BCD ␣ Ⓢ ␣ Ⓓ**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number containing floating-point data							○				○				3						○	
Ⓓ	Device number for storing BCD data		○	○	○	○			○		○												

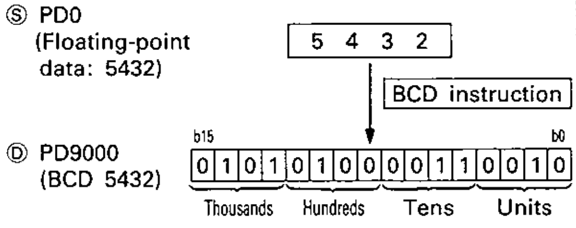
FUNCTIONS

(1) Converts the specified floating-point data (0 to 9999), Ⓢ , into BCD and transfers the result to the specified device, Ⓓ .



Example

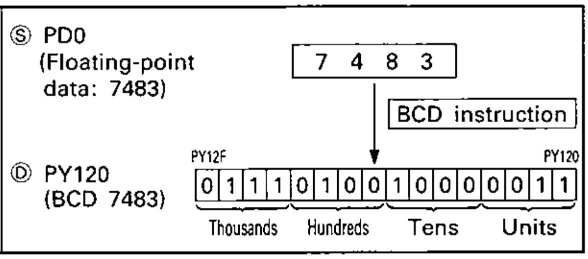
BCD PD0 PD9000



(2) If a bit device is specified as Ⓓ , 4-digit BCD data is transferred to 16 bits headed by the specified bit device.

Example

BCD PD0 PY120

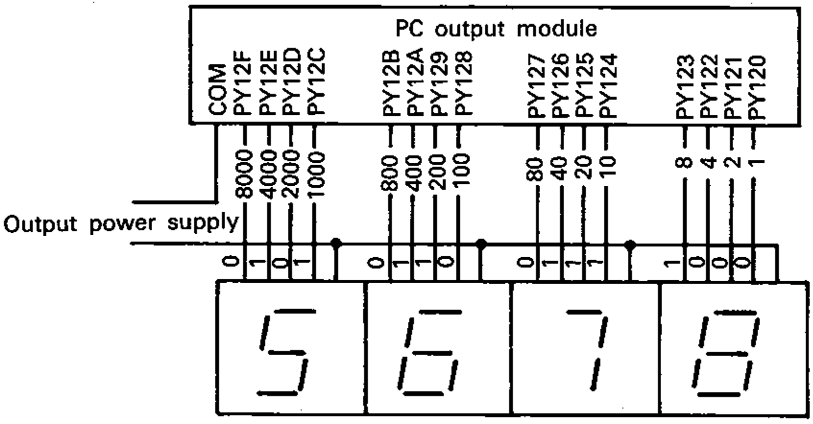


RESTRICTIONS

1) Any bit device (PX, PY, PM, SP.PM) number specified as S and Ⓓ must be a multiple of 16.

2) Any value between 0 and 9999 may be converted into BCD.

PROGRAM EXAMPLE The following program outputs the PD150 floating-point data from PY120 to 12F to the BCD display.



7-segment display

```
0 BCD   PD 150          PY 120············Outputs PD150 floating-point data
3 END                                     to Y120 to Y12F.
```

HINT

16 points headed by PY□□0 are used when specifying a bit device as Ⓓ for the BCD instruction. To output data to the BCD display by the BCD instruction, cables should be wired so that the number of units indicated on the BCD display may be output to PY□□0 to □□3.

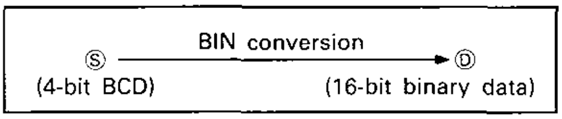
6.5.3 BIN conversion instruction (4-digit BCD to 16-bit binary data) ······ BIN

FORMAT	BIN ␣ Ⓢ ␣ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number containing BCD data	○	○	○	○	○			○							3						○	
Ⓓ	Device number for storing BIN data		○	○	○	○			○		○												

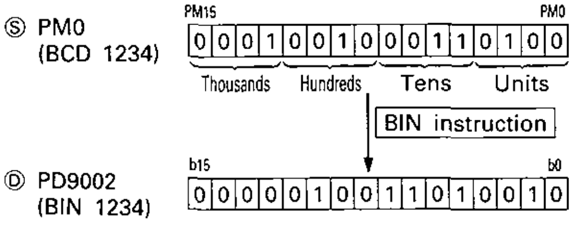
FUNCTIONS

(1) Converts the specified 4-digit BCD data (0 to 9999), Ⓢ , into binary data and transfers the result to the specified device, Ⓓ .



Example

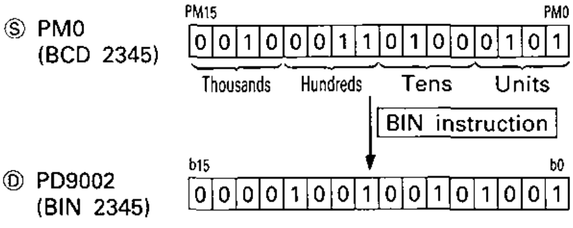
BIN PM0 PD9002



(2) If a bit device is specified as Ⓢ , 16 bits headed by the specified bit device are treated as 4-digit BCD data.

Example

BIN PM0 PD9002



RESTRICTIONS

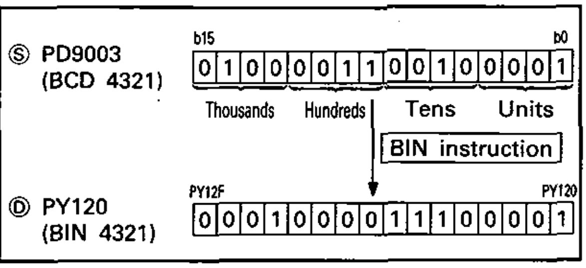
1) Any bit device (PX, PY, PM, SP.PM) number specified as Ⓢ and Ⓓ must be a multiple of 16.

2) Any 4-digit BCD value between 0 and 9999 may be converted into BIN.

(3) If a bit device is specified as Ⓓ, binary data is transferred to 16 bits headed by the specified bit device.

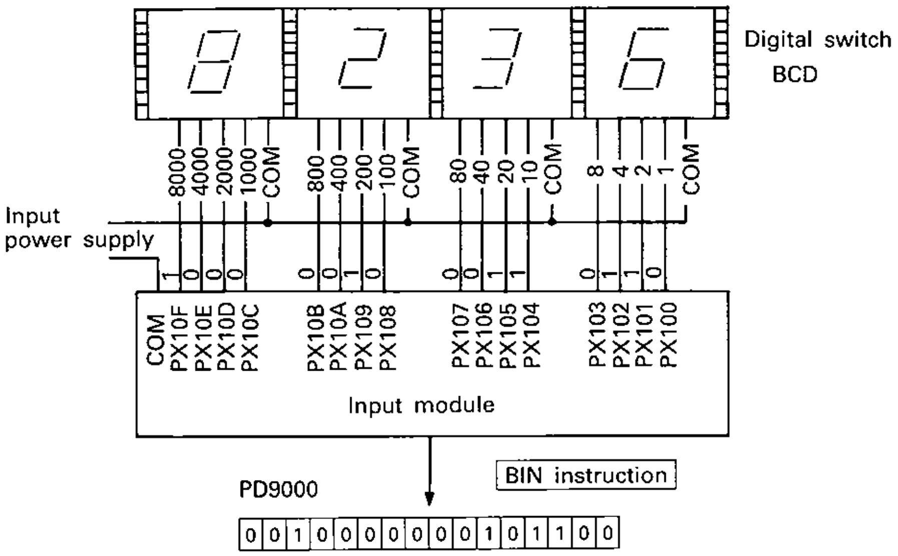
Example

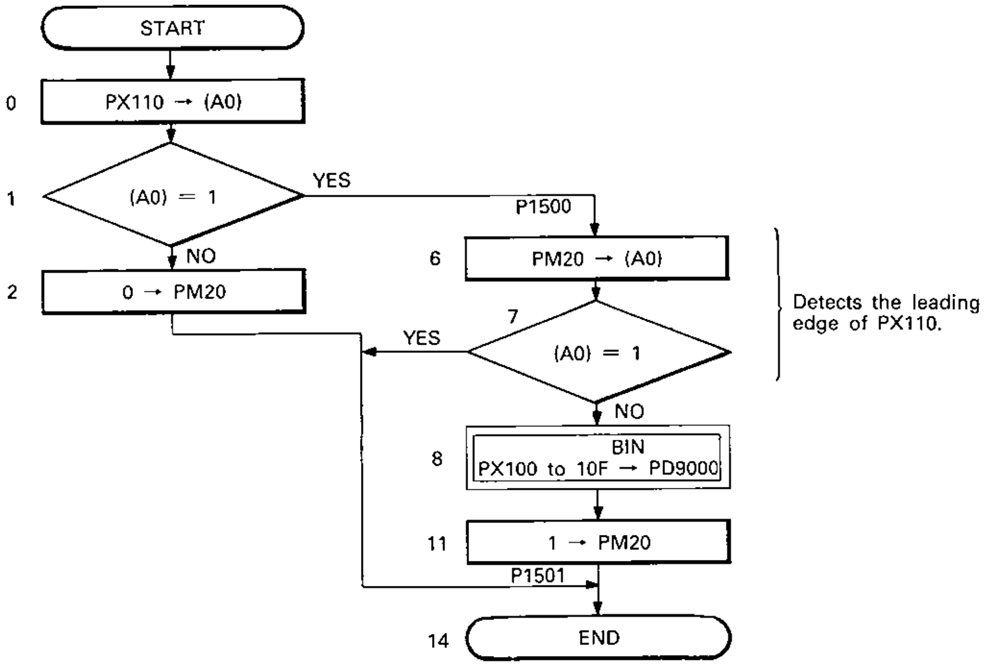
BIN PD9003 PY120



PROGRAM EXAMPLE

The following program converts BCD data of PX100 to 10F into BIN and stores the result to PD9000 when PX110 is switched on. (Program 15 used)





HINT

16 points headed by PX□0 are used when specifying a bit device as Ⓢ for the BIN instruction. To read the BCD code from the digital switch by the BIN instruction, cables should be wired so that the number of units indicated on the digital switch may be input from PX□0 to □3.

```
 0 LDAB PX 110 ...........................Reads PX110 data to (A0).
 1 JC     P 1500 .........................Judges the ON/OFF state of
                                          PX110.
 2 RST  PM 20 ............................Resets PM20.
 3 JMP    P 1501 .........................Jumps to pointer P1501.
 4 P    1500
 6 LDAB PM 20 ............................Reads PM20 data to (A0).
 7 JC     P 1501 .........................Judges the ON/OFF state of PM20.
 8 BIN  PX 100  PD 9000 ..................Converts BCD data of PX100 to
                                          10F into BIN and stores the result
                                          to PD9000.
11 SET  PM 20
12 P    1501
14 END
```

6.5.4 BIN conversion instruction (4-digit BCD to floating-point data) …… BIN

FORMAT	BIN ␣ Ⓢ ␣ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number containing BCD data	○	○	○	○	○			○							3						○	
Ⓓ	*Device number for storing floating-point data*							○				○											

FUNCTIONS

(1) Converts the specified 4-digit BCD data (0 to 9999), Ⓢ, into floating-point data and transfers the result to the specified device, Ⓓ.

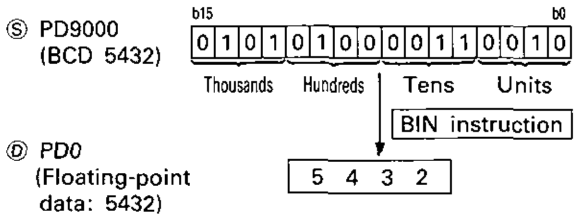
BIN conversion

Ⓢ ———————→ Ⓓ

(4-digit BCD) (Floating-point data)

Example

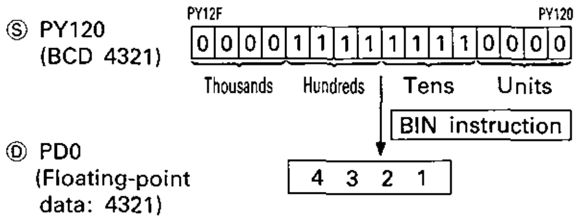
BIN PD9000 PD0



(2) If a bit device is specified as Ⓢ, 16 bits headed by the specified *bit device are treated as 4-digit BCD data.*

Example

BIN PY120 PD0



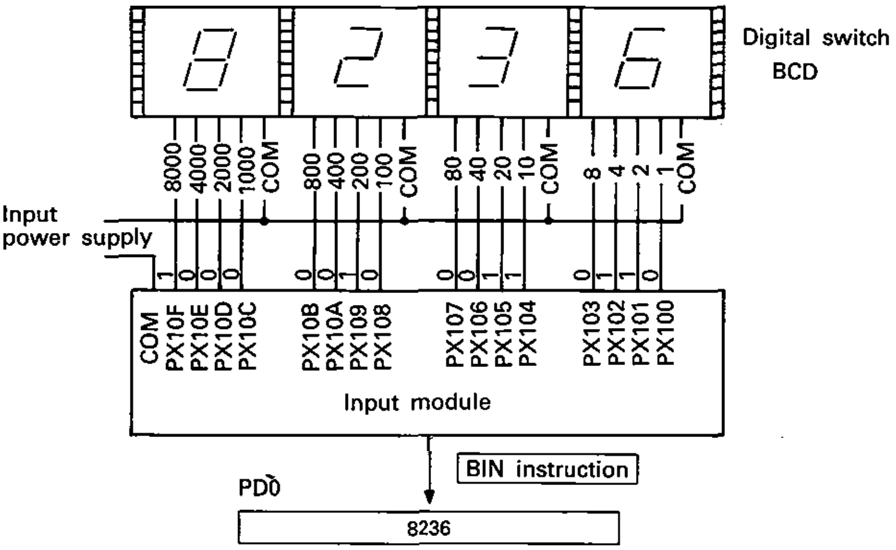
RESTRICTIONS

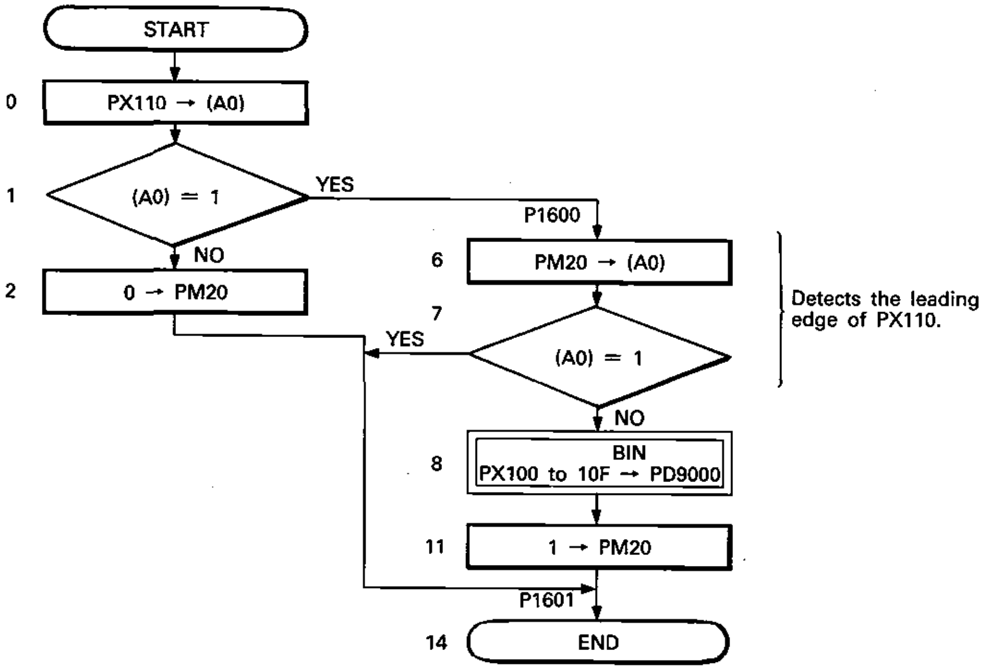
1) Any bit device (PX, PY, PM, SP.PM) number specified as Ⓢ and Ⓓ must be a multiple of 16.

2) Any 4-digit BCD value between 0 and 9999 may be converted into BIN.

PROGRAM EXAMPLE

The following program converts BCD data of PX100 to 10F into floating-point data and stores the result to PD0 when PX110 is switched on. (Program 16 used)





HINT

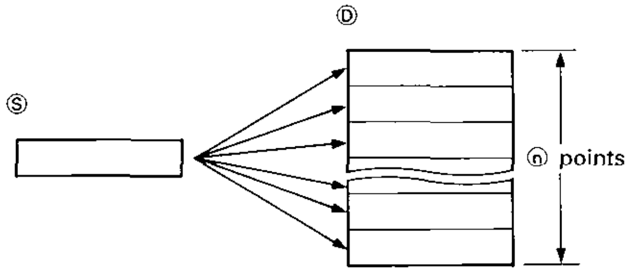
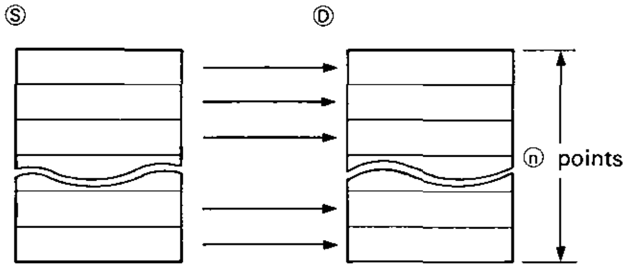
16 points headed by PX□0 are used when specifying a bit device as Ⓢ for the BIN instruction. To read the BCD code from the digital switch by the BIN instruction, cables should be wired so that the number of units indicated on the digital switch may be input from PX□0 to □3.

```
 0 LDAB PX 110 ·····························Reads PX110 data to (A0).
 1 JC      P 1600 ··························Judges the ON/OFF state of
                                            PX110.
 2 RST  PM 20 ······························Resets PM20.
 3 JMP     P 1601 ··························Jumps to pointer P1601.
 4 P       1600
 6 LDAB PM 20 ······························Reads PM20 data to (A0).
 7 JC      P 1601 ··························Judges the ON/OFF state of PM20.
 8 BIN  PX 100          PD 0 ···············Converts BCD data of PX100 to
                                            10F into BIN and stores the result
                                            to PD0.
11 SET  PM 20 ······························Sets PM20.
12 P       1601
14 END
```

6.6 Transfer Instructions

Used to process 1-bit data, 16-bit binary data and floating-point data. Instructions and device combinations used depend on the data processed.

Instruction	Description	Refer To
LDAB	Stores the specified bit device data, Ⓢ, to accumulator (A0). Ⓢ ——→ (A0)	Section 6.6.1
LDAW	Stores the specified word device data or constant, Ⓢ, to accumulator (A1). Ⓢ ——→ (A1)	Section 6.6.2
LDAF	Converts the specified device data or constant, Ⓢ, into floating-point data and stores the result to accumulator (A2). Ⓢ ——→ (A2)	Section 6.6.3
STAB	Transfers bit data from accumulator (A0) to the specified bit device, Ⓓ. (A0) ——→ Ⓓ	Section 6.6.4
STAW	Transfers word data from accumulator (A1) to the specified word device, Ⓓ. (A1) ——→ Ⓓ	Section 6.6.5
STAF	Transfers floating-point data from accumulator (A2) to the specified device, Ⓓ. (A2) ——→ Ⓓ	Section 6.6.6
MOV	Transfers data or constant from the specified device, Ⓢ, to the specified device, Ⓓ. Ⓢ ——→ Ⓓ Any of the following may be used as appropriate in accordance with the combination of data processed.	—
	(1) Transferring 1-bit data to 1 bit Transfers data from one bit device to the other.	Section 6.6.7
	(2) Transferring 16-bit data to 16 bits Transfers word device data or constant to 16 bit devices or a word device.	Section 6.6.8
	(3) Transferring floating-point data to 16 bits Transfers floating-point data to 16 bit devices or a word device.	Section 6.6.9
	(4) Transferring 16-bit data to floating-point data device. Transfers word device data or constant to a floating-point data device.	Section 6.6.10
	(5) Transferring floating-point data to floating-point data device. Transfers floating-point data to a floating-point data device.	Section 6.6.11

Instruction	Description	Refer To
FMOV	Transfers the specified device data or constant, Ⓢ, to the number of devices, ⓝ, headed by the specified device, Ⓓ.  Either of the following may be used in accordance with the combination of data processed:	—
	(1) Batch-transferring 16-bit data Transfers 16-bit binary data, word device data or constant in batches to 16 bit devices or a word device.	Section 6.6.12
	(2) Batch-transferring floating-point data Transfers floating-point data in batches to floating-point device.	Section 6.6.13
BMOV	Transfers data from the specified number of devices, ⓝ, headed by the specified device, Ⓢ, to the number of devices, ⓝ, headed by the specified device, Ⓓ.  Either of the following may be used in accordance with the combination of data processed:	—
	(1) Block-transferring 16-bit data Transfers 16-bit binary data or word device data in blocks to 16 bit devices or a word device.	Section 6.6.14
	(2) Block-transferring floating-point data Transfers floating-point data in blocks to a floating-point device.	Section 6.6.15

6.6.1 Transfer to accumulator (A0) ······ LDAB

FORMAT	LDAB ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓓ	Bit device number read to (A0)	○	○	○	○											1							

FUNCTIONS

(1) Stores the specified bit device data, Ⓢ, to accumulator (A0).

	Ⓢ	Ⓢ → (A0)
Ⓢ ——→(A0)	0	0
	1	1

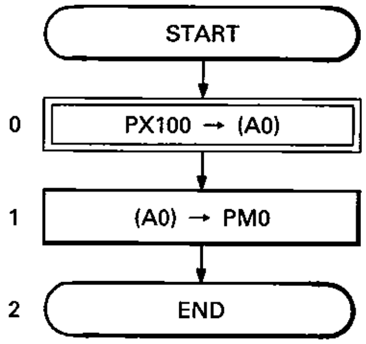
(2) The specified bit device data, Ⓢ, remains unchanged after the LDAB instruction is executed.

REMARKS

The (A0) data is overwritten by the LDAB execution result and therefore should be saved before LDAB is executed if the data is required.

PROGRAM EXAMPLE

The following program switches ON/OFF PM0 in accordance with the ON/OFF state of PX100.



```
0 LDAB PX 100························Reads PX100 data to (A0).
1 STAB PM 0·····························Stores (A0) data to PM0.
2 END
```

6.6.2 Transfer to accumulator (A1) ······ LDAW

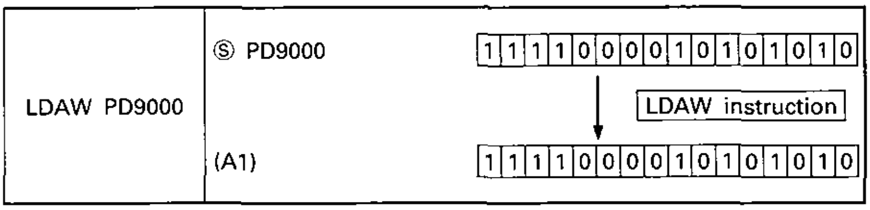
FORMAT	LDAW ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Word device number or constant read to (A1)					○			○				○	○		1							

FUNCTIONS

(1) Stores the specified word device data or constant, Ⓢ, to accumulator (A1).

Ⓢ → (A2)

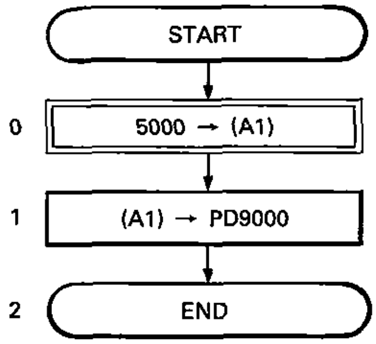


(2) The specified word device data, Ⓢ, remains unchanged after the LDAW instruction is executed.

REMARKS

The (A1) data is overwritten by the LDAW execution result and therefore should be saved before LDAW is executed if the data is required.

PROGRAM EXAMPLE The following program stores 5000 to PD9000.



```
0 LDAW  K 5000 ······································Reads data 5000 to (A1).
1 STAW PD 9000 ······································Stores (A1) data to PD9000.
2 END
```

6.6.3 Transfer to accumulator (A2) ······ LDAF

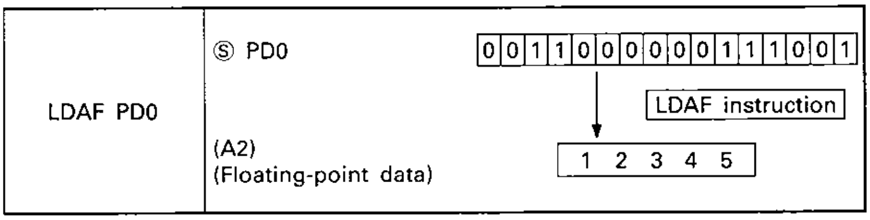
FORMAT	LDAF ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Floating-point device number or constant read to (A2)					○		○					○			1							

FUNCTIONS

(1) Converts the specified word device data or constant, Ⓢ, into floating-point data and stores the result to accumulator (A2).

Ⓢ → (A2)



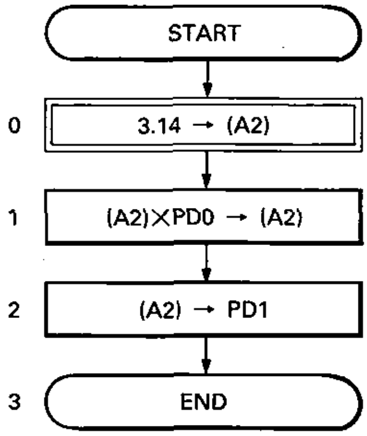
(2) The specified device data, Ⓢ, remains unchanged after the LDAF instruction is executed.

REMARKS

The (A2) data is overwritten by the LDAF execution result and therefore should be saved before LDAF is executed if the data is required.

PROGRAM EXAMPLE

The following program multiplies the PD0 value by 3.14 and stores the result to PD1.



```
0 LDAF   K 3.14 ·····································Stores constant 3.14 to (A2).
1 *      PD 0 ·······································Stores the multiplication result of
                                                     (A2) and PD0 data to (A2).
2 STAF PD 1··········································Stores (A2) data to PD1.
3 END
```

6.6.4 Transfer from accumulator (A0) ······ STAB

FORMAT	STAB ␣ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number for storing (A0) data		○	○	○											1						○	

FUNCTIONS

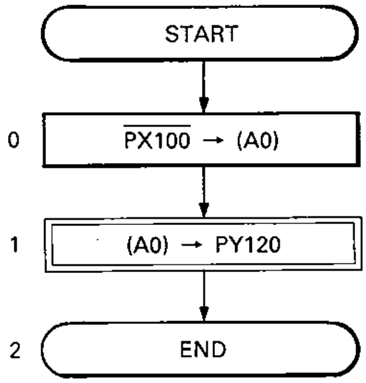
(1) Transfers data from accumulator (A0) to the specified bit device, Ⓓ.

	Ⓓ	(A0) → Ⓓ
(A0) → Ⓓ	0	0
	1	1

(2) The (A0) data remains unchanged after the STAB instruction is executed.

PROGRAM EXAMPLE

The following program switches on PY120 when PX100 is off and switches off PY120 when PX100 is on.



```
0 NOT   PX 100························Complements PX100 data and stores the result
                                        to (A0).
1 STAB PY 120························Transfers (A0) data to PY120.
2 END
```

6.6.5 Transfer from accumulator (A1) ······ STAW

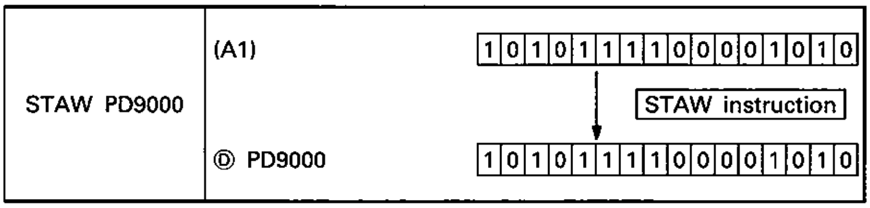
FORMAT	STAW ␣ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓓ	Device number for storing (A1) data					○			○							1						○	

FUNCTIONS

(1) Transfers data from accumulator (A1) to the specified word device, Ⓓ.

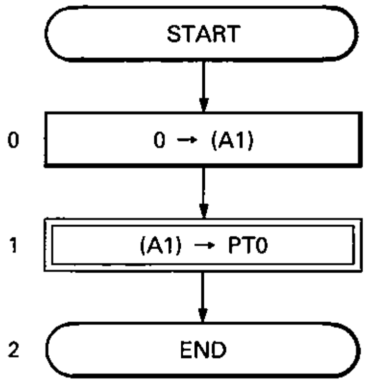
(A1) → Ⓓ



(2) The (A1) data remains unchanged after the STAW instruction is executed.

PROGRAM EXAMPLE

The following program zeroes PT0 present value.



```
0 LDAW  K 0·····························Stores constant 0 to (A1).
1 STAW  PT 0····························Transfers (A1) data to PT0.
2 END
```

6.6.6 Transfer from accumulator (A2) ······ STAF

FORMAT	STAF ␣ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓓ	Device number for storing (A2) data					○		○								1						○	

FUNCTIONS

(1) Transfers data from accumulator (A2) to the specified floating-point device, Ⓓ.

(A2) → Ⓓ

Example

STAF PD0

(A2) 5 3. 4 2

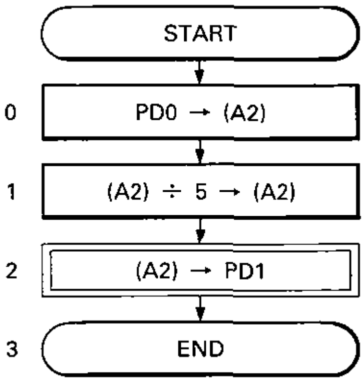
↓

Ⓓ PD0 5 3. 4 2

(2) If PT is specified as Ⓓ, the floating-point data is transferred from accumulator (A2) after it is converted into binary data.

PROGRAM EXAMPLE

The following program divides the PD0 data by 5 and stores the resultant quotient to PD1.



```
0  LDAF  PD  0·····························Reads PD0 data to (A2).
1  /      K  5·····························Divides (A2) data by 5.
2  STAF  PD  1·····························Transfers the result to PD1.
3  END
```

RESTRICTION

Any (A2) value between −32768 and 32767 may be transferred to PT.

6.6.7 Transferring 1-bit data to 1 bit device ...... MOV

FORMAT	**MOV ␣ Ⓢ ␣ Ⓓ**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Source bit device number	○	○	○	○					○						3							
Ⓓ	Destination bit device number		○	○	○					○													

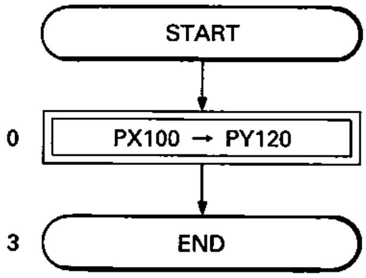
FUNCTIONS

(1) Transfers the specified bit device data, Ⓢ , to the specified bit device, Ⓓ .

Ⓢ ——→ Ⓓ
(1-bit data) (1-bit data)

PROGRAM EXAMPLE

The following program switches ON/OFF PY120 in accordance with the ON/OFF state of PX100.



```
0 MOV   PX 100          PY 120......Transfers data from PX100 to PY120.
3 END
```

RESTRICTION

(A0) cannot be specified as both Ⓢ and Ⓓ.

HINTS

1) The STAB instruction may be used if (A0) is specified as Ⓢ.

2) The LDAB instruction may be used if (A0) is specified as Ⓓ.

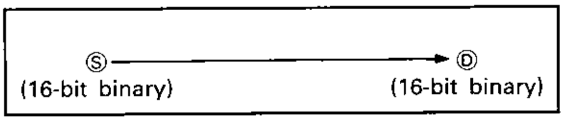
6.6.8 Transferring 16-bit data to 16 bits ······ MOV

FORMAT	MOV ⌴ Ⓢ ⌴ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Source device number or constant					○			○		○		○	○		3							
Ⓓ	Destination device number		○	○	○	○			○		○												

FUNCTIONS

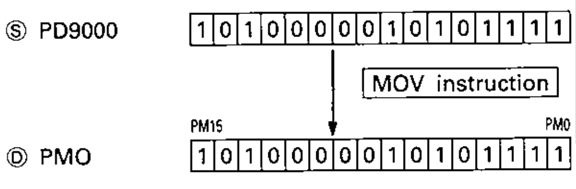
(1) Transfers the specified 16-bit data, Ⓢ, to the specified 16 bits, Ⓓ.



(2) If a bit device is specified as Ⓓ, 16 bits headed by the specified bit device are treated as binary data.

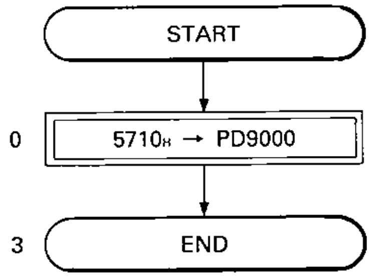
Example

MOV PD9000 PMO



PROGRAM EXAMPLE

The following program sets 5710H to PD9000.



```
0 MOV   H 5710  PD 9000······Sets 5710H to PD9000.
3 END
```

RESTRICTIONS

1) The bit device number, Ⓓ, must be a multiple of 16.

2) (A1) cannot be specified as both Ⓢ and Ⓓ.

6.6.9 Transferring floating-point data to 16 bits …… MOV

FORMAT	MOV ␣ Ⓢ ␣ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Source floating-point device number							○				○				3						○	
Ⓓ	Destination device number		○	○	○				○		○												

FUNCTIONS

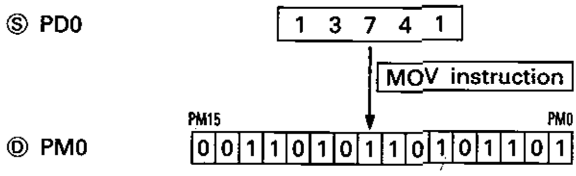
(1) Converts the specified floating-point device data, Ⓢ, into 16-bit binary data and transfers the result to the specified device, Ⓓ.

Ⓢ ——————→ Ⓓ
(Floating-point data) (16-bit binary data)

(2) If a bit device is specified as Ⓓ, 16 bits headed by the specified bit device are treated as binary data.

Example

MOV PD0 PM0



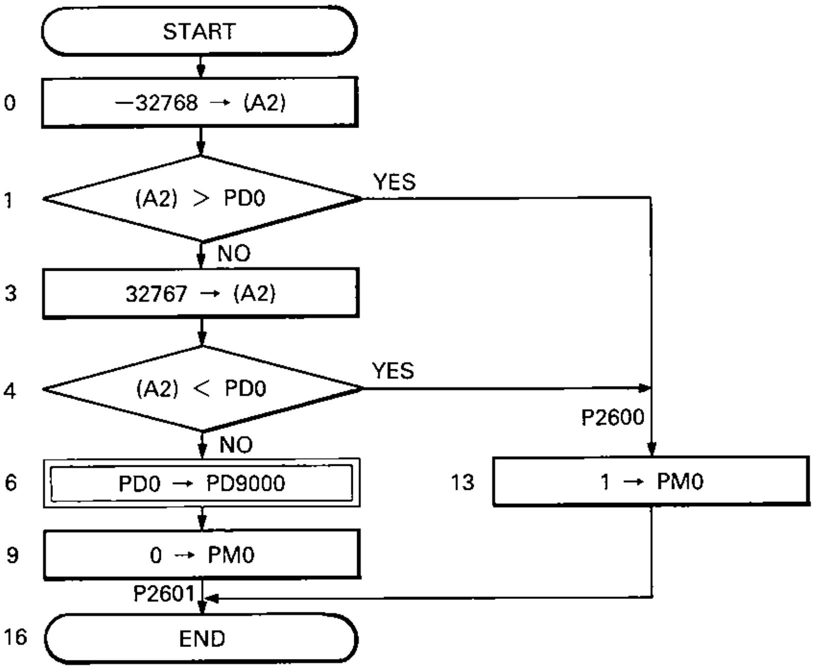
RESTRICTIONS

1) The specified device data, Ⓢ, between -32768 and 32767 may only be converted into 16-bit binary data and transferred to Ⓓ without any fault.

2) The bit device number, Ⓓ, must be a multiple of 16.

PROGRAM EXAMPLE

The following program converts the floating-point data of PD0 into 16-bit binary data and transfers the result to PD9000 if the PD0 data is between −32768 and 32767, and switches on PM0 if the PD0 data is outside the above range. (Program 26 used)



```
 0 LDAF  K-32768..................Sets −32768 to (A2).
 1 GTAF PD 0......................Compares PD0 data with (A2)
                                  data.
 2 JMP   P 2600...................Jumps to pointer P2600 if PD0
                                  data is less than (A2) data.
 3 LDAF  K 32767..................Sets 32767 to (A2) if PD0 data is
                                  greater than or equal to (A2) data.
 4 LTAF PD 0......................Compares PD0 data with (A2)
                                  data.
 5 JMP   P 2600...................Jumps to pointer P2600 if PD0
                                  data is greater than (A2) data.
 6 MOV  PD 0   PD 9000............Converts PD0 floating-point data
                                  into 16-bit binary and transfers
                                  the result to PD9000.
 9 RST  PM 0......................Switches off PM0.
10 JMP   P 2601...................Jumps to pointer P2601.
11 P    2600......................Pointer P2600
13 SET  PM 0......................Switches on PM0.
14 P    2601......................Pointer P2601
16 END
```

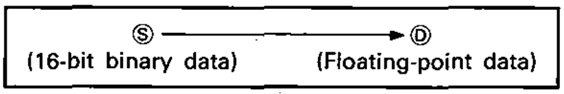
6.6.10 Transferring 16-bit data to floating-point data device ······ MOV

FORMAT	MOV ␣ Ⓢ ␣ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Source word device number or constant					○			○		○			○		3							
Ⓓ	Destination floating-point device number							○				○											

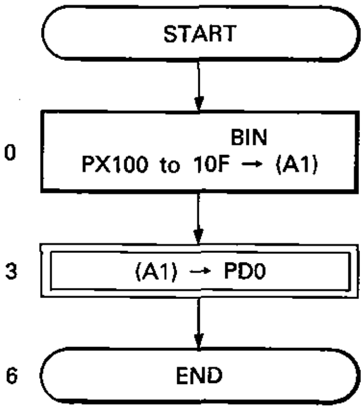
FUNCTIONS

(1) Converts the specified 16-bit binary data, Ⓢ, into floating-point data and transfers the result to the specified device, Ⓓ.



PROGRAM EXAMPLE

The following program converts 4-digit BCD data of PX100 to 10F into floating-point data and transfers the result to PD0.



```
0 BIN PX 100   A 1 ··············Converts BCD data of PX100 to 10F into BIN
                                 and stores the result to (A1).
3 MOV  A 1    PD 0 ··············Converts 16-bit binary data of (A1) into float-
6 END                            ing-point data and transfers the result to PD0.
```

RESTRICTIONS

Constant H (Hexdecimal) may be specified between 0 and FFFFH.

HINT

16-bit binary data in bit devices should be transferred to a word device by using the BMOV instruction before it is converted into floating-point data by the MOV instruction.

6.6.11 Transferring floating-point data to floating-point data device ······ MOV

FORMAT	MOV ␣ Ⓢ ␣ Ⓓ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Source floating-point device number							○				○	○			3							
Ⓓ	Destination floating-point device number							○				○											

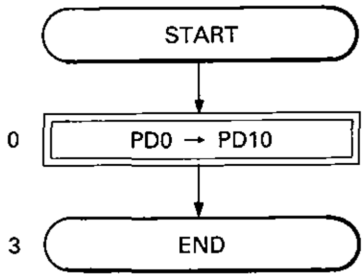
FUNCTIONS

(1) Transfers the specified floating-point data, Ⓢ, to the specified device, Ⓓ.

Ⓢ ——→ Ⓓ
(Floating-point data) (Floating-point data)

PROGRAM EXAMPLE

The following program transfers data from PD0 to PD10.



```
0 MOV   PD 0    PD 10·········Transfers PD0 data to PD10.
3 END
```

RESTRICTIONS

1) (A2) cannot be specified as both Ⓢ and Ⓓ.

2) Constant K may be specified between −9999000000 and 9999000000.

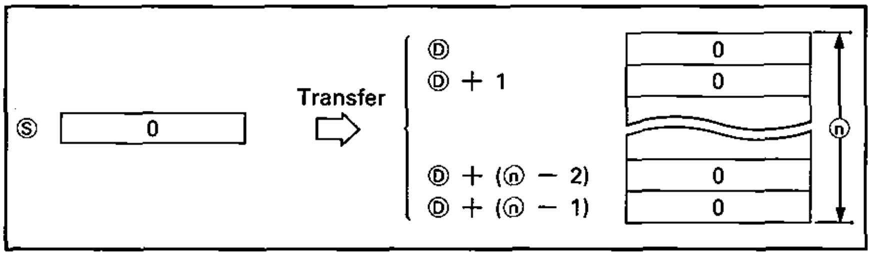
6.6.12 Batch-transferring 16-bit binary data ...... FMOV

FORMAT	FMOV ␣ Ⓢ ␣ Ⓓ ␣ ⓝ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device data or constant to be transferred	○	○	○	○	○			○		○		○	○		4						○	
Ⓓ	Head destination device number		○	○	○	○			○														
ⓝ	Number of data transferred												○	○									

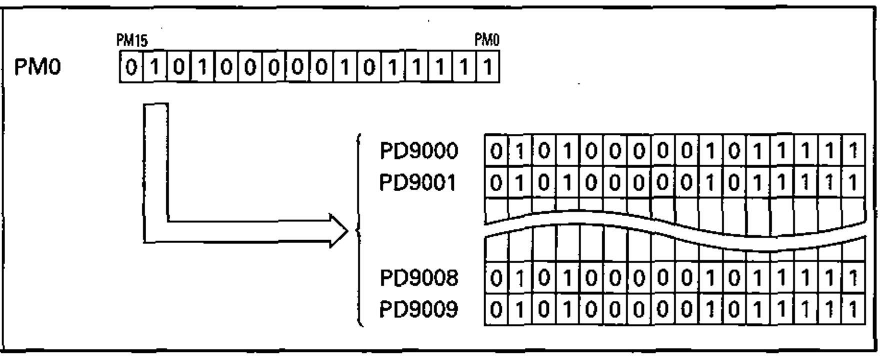
FUNCTIONS

(1) Transfers the specified 16-bit binary data, Ⓢ , to the specified number of devices, ⓝ , headed by the specified device, Ⓓ .



(2) If a bit device is specified as Ⓢ or Ⓓ , 16 bits headed by the specified bit device are treated as binary data.

FMOV PM0 PD9000 K10

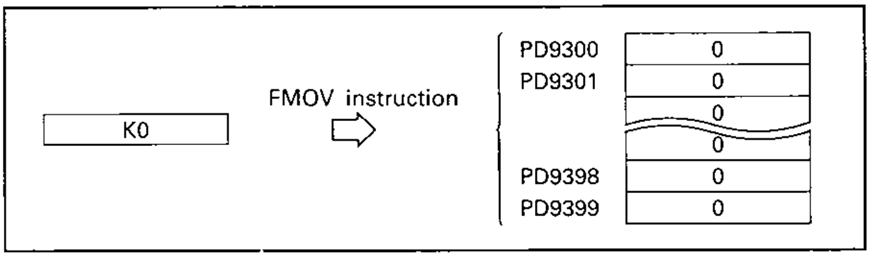


RESTRICTIONS

1) The specified bit devices, Ⓢ and Ⓓ , must be a multiple of 16.

2) ⓝ should not be outside the allowed range of the corresponding device. Any device outside the allowed range is not processed, e.g. PD9500 to 9511 (12 points) are only processed if FMOV K0 PD9500 K20 is defined.

PROGRAM EXAMPLE The following program clears PD9300 through PD9399.



```
0 FMOV   K 0    PD 9300   K 100
4 END
```

Transfers 0 to PD9300 through PD9399.

HINT

FMOV is useful for initializing several bit or word devices.

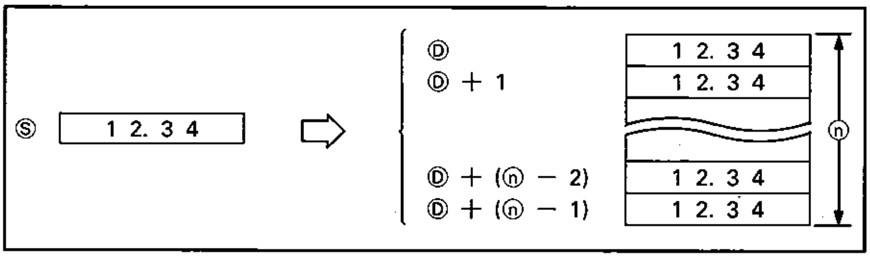
6.6.13 Batch-transferring floating-point data ······ FMOV

FORMAT	FMOV ␣ Ⓢ ␣ Ⓓ ␣ ⓝ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device data or constant to be transferred							○				○	○	○		4						○	
Ⓓ	Head destination device number							○															
ⓝ	Number of data transferred												○	○									

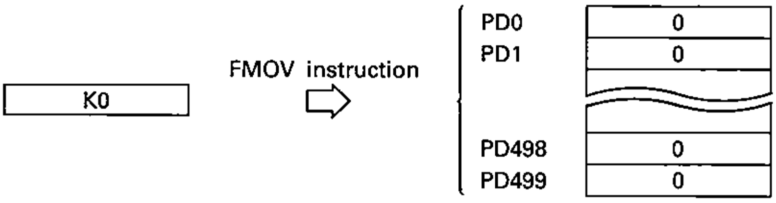
FUNCTIONS

(1) Transfers the specified floating-point data, Ⓢ, to the specified number of devices, n, headed by the specified device, Ⓓ.



PROGRAM EXAMPLE

The following program initializes PD0 through PD499.



```
0  FMOV    K0     PD  0     K  500 ·····Transfers 0 to PD0 through PD499.
4  END
```

RESTRICTION

ⓝ should not be outside the allowed range of the corresponding device. Any device outside the allowed range is not processed, e.g. PD1000 to 1023 (24 points) are only processed if FMOV K0 PD1000 K30 is defined.

HINT

FMOV is useful for initializing several data registers.

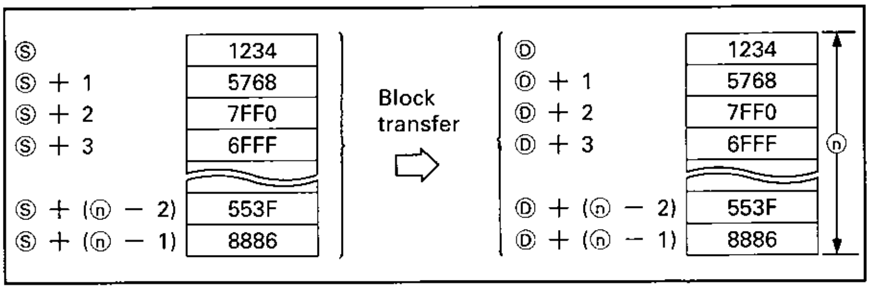
6.6.14 Block-transferring 16-bit binary data ······ BMOV

FORMAT	BMOV ⊔ Ⓢ ⊔ Ⓓ ⊔ ⓝ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Head source device number	○	○	○	○	○			○							4							
Ⓓ	Head destination device number		○	○	○	○			○													○	
ⓝ	Number of data transferred												○	○									

FUNCTIONS

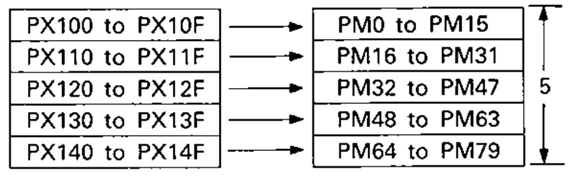
(1) Transfers the specified number of data, n, in blocks from the devices headed by the specified device, Ⓢ, to the specified number of devices, ⓝ, headed by the specified device, Ⓓ.



(2) If a bit device is specified as Ⓢ, Ⓓ, the specified number of bit devices, n, headed by the specified bit device are processed in multiples of 16 bits.

Example

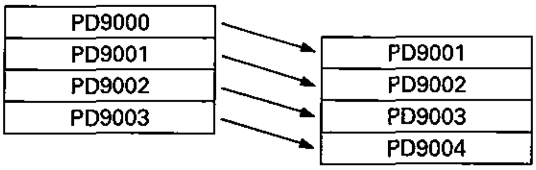
BMOV PX100 PM0 K5



(3) Devices specified as source may be defined as destination, and vice versa.

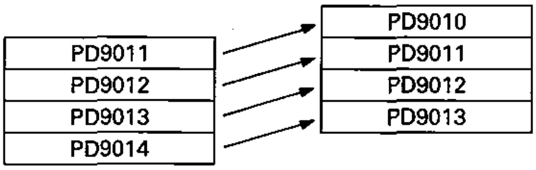
Example

BMOV PD9000 PD9001 K4



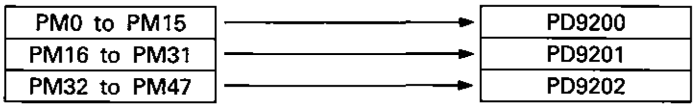
Example

BMOV PD9011 PD9010 K4



PROGRAM EXAMPLE

The following program transfers data from PM0 to PM47 to PD9200 to PD9202.



```
0 BMOV PM 0 PD 9200 K 3 ···Transfers PM0 to PM47 data to PD9022 to
                           PD9202.
4 END
```

RESTRICTIONS

1) The specified bit device numbers, Ⓢ and Ⓓ, must be a multiple of 16.

2) ⓝ should not be outside the allowed range of the corresponding device. Any device outside the allowed range is not processed, e.g. PT100 to 127 (28 points) are only processed if BMOV PT100 PD9000 K30 is defined.

HINT

BMOV may be used to transfer data from bit devices to word devices.

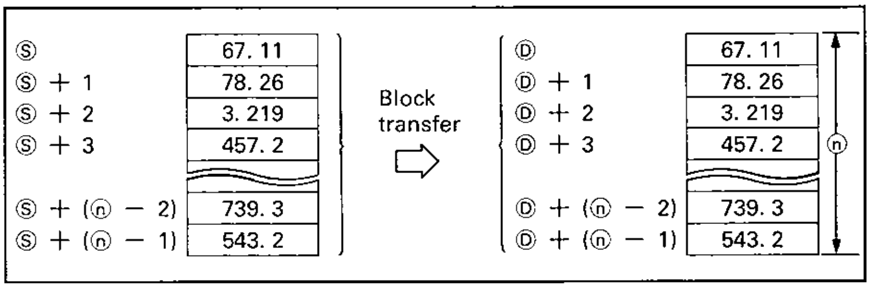
6.6.15 Block-transferring floating-point data …… BMOV

FORMAT	BMOV ␣ Ⓢ ␣ Ⓓ ␣ ⓝ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Head source device number							○								4							
Ⓓ	Head destination device number							○														○	
ⓝ	Number of data transferred												○	○									

FUNCTIONS

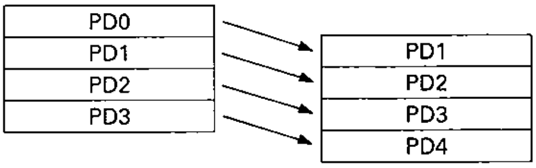
(1) Transfers the specified number of data, ⓝ, from the data registers (PD) headed by the specified PD, Ⓢ, to the specified number of devices, ⓝ, headed by the specified device, Ⓓ.



(2) Devices specified as source may be defined as destination, and vice versa.

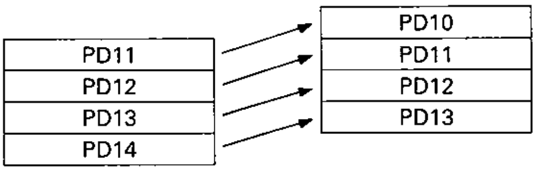
Example

BMOV PD0 PD1 K4



Example

BMOV PD11 PD10 K4

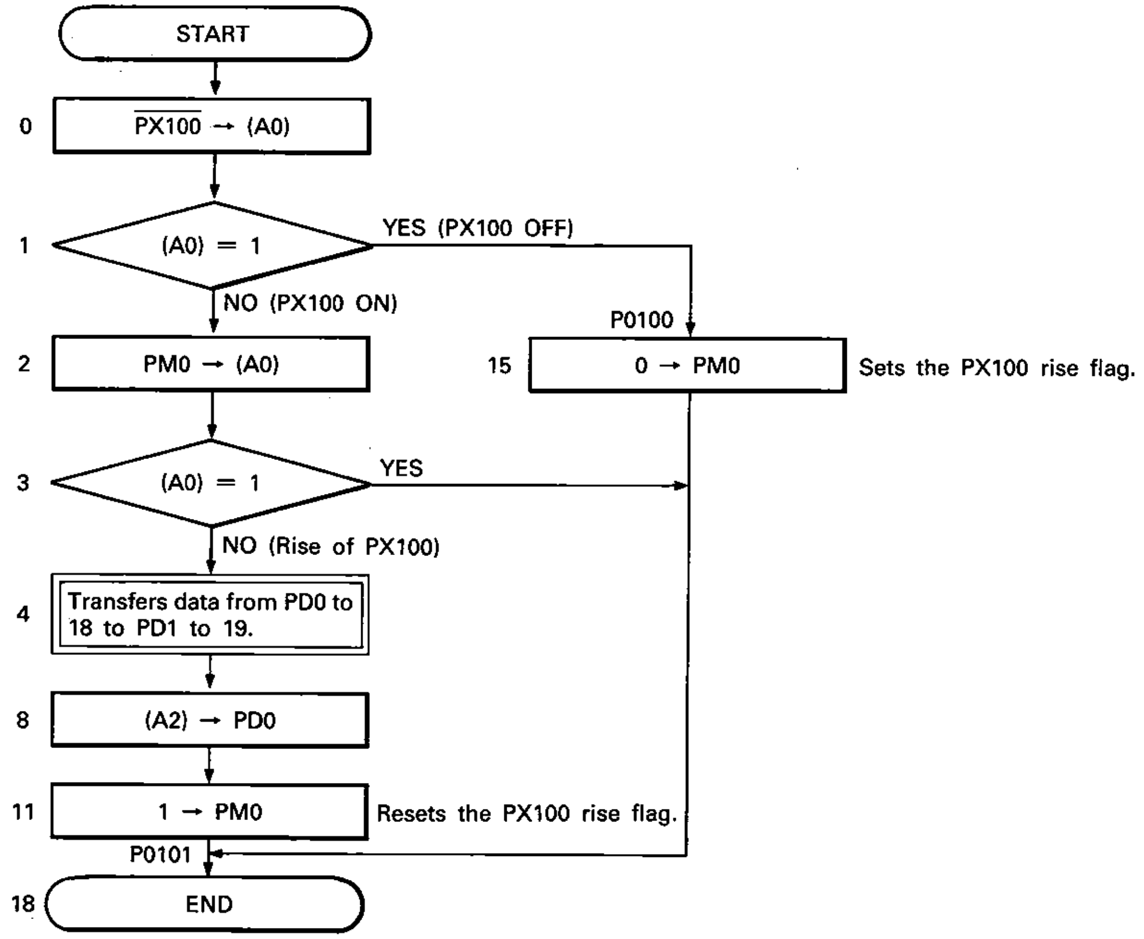


RESTRICTION

ⓝ should not be outside the allowed range of the corresponding device. Any device outside the allowed range is not processed, e.g. data is onlytransferred from PD0 through 23 (24 points) to PD1000 through 1023 if BMOV PD0 PD1000 K30 is defined.

PROGRAM EXAMPLE

The following program transfers data from PD0 to PD18 to PD1 to PD19 and stores (A2) data to PD0 when PX100 is switched on. (Program 1 used)



```
 0 NOT   PX 100 ·····································Reads PX100 data to (A0).
 1 JC     P 0100·····································Judges ON/OFF state of PX100.
 2 LDAB PM 0·········································Reads PM0 data to (A0).
 3 JC     P 0101·····································Judges ON/OFF state of PM0.
 4 BMOV PD 0   PD 0   K 19 ························Transfers data from PD0 to 18 to
                                                     PD1 to 19.
 8 MOV    A 2   PD 1 ································Transfers data from (A2) to PD0.
11 SET   PM 0········································Sets PM0 (PX100 rise flag).
12 JMP    P 0101·····································Jumps to P0101.
13 P     0100········································Pointer P0100.
15 RST   PM 0········································Resets PM0 (PX100 rise flag).
16 P     0101 ·······································Pointer P0101.
18 END
```

6.7 Buffer Memory Access Instructions

Used to access the buffer memory of the special function module loaded to the A81CPU's base unit.

Instruction	Description	Refer To
FROM	Reads 1-word data from the special function module buffer memory. Either of the following may be used in accordance with the device used to store the data read.	—
	(1) Reading data to word device Stores 16-bit binary data of buffer memory to a word device.	Section 6.7.1
	(2) Reading data to floating-point device Converts 16-bit binary data of buffer memory into floating-point data and stores the result to a floating-point device.	Section 6.7.2
DFRO	Reads 2-word data from the special function module buffer memory. Either of the following may be used in accordance with the device used to store the data read.	—
	(1) Reading data to word devices Stores 32-bit binary data of buffer memory to two word devices.	Section 6.7.3
	(2) Reading data to floating-point devices Converts 32-bit binary data of buffer memory into floating-point data and stores the result to floating-point devices.	Section 6.7.4
TO	Writes data to 1-word area of the special function module buffer memory. Either of the following may be used in accordance with the data written.	—
	(1) Writing 16-bit binary data Writes word device data or constant to 1-word area of the buffer memory.	Section 6.7.5
	(2) Writing floating-point data Converts floating-point data into 16-bit binary data and writes the result to 1-word area of the buffer memory.	Section 6.7.6
DTO	Writes data to 2-word area of the special function module buffer memory. Either of the following may be used in accordance with the data written.	—
	(1) Writing 32-bit binary data Writes 32-bit binary data from two word devices to 2-word area of the buffer memory.	Section 6.7.7
	(2) Writing floating-point data Converts floating;point data into 32-bit binary data and writes the result to 2-word area of the buffer memory.	Section 6.7.8

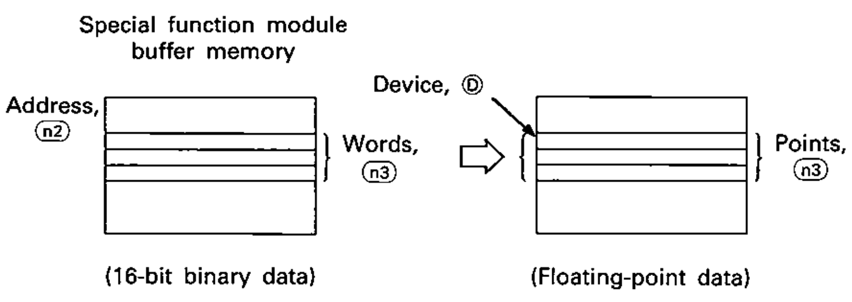
6.7.1 Reading data from special function module in blocks of 1 word (16-bit binary data to 16-bit binary data) …… FROM

FORMAT	**FROM ⌴ (n1) ⌴ (n2) ⌴ Ⓓ ⌴ (n3)**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(n1)	Two most significant digits of special function module head I/O number												○	○		5	○	○	○			○	
(n2)	Head address of buffer memory												○	○									
Ⓓ	Head device number for storing data read								○														
(n3)	Number of data read												○	○									

FUNCTIONS

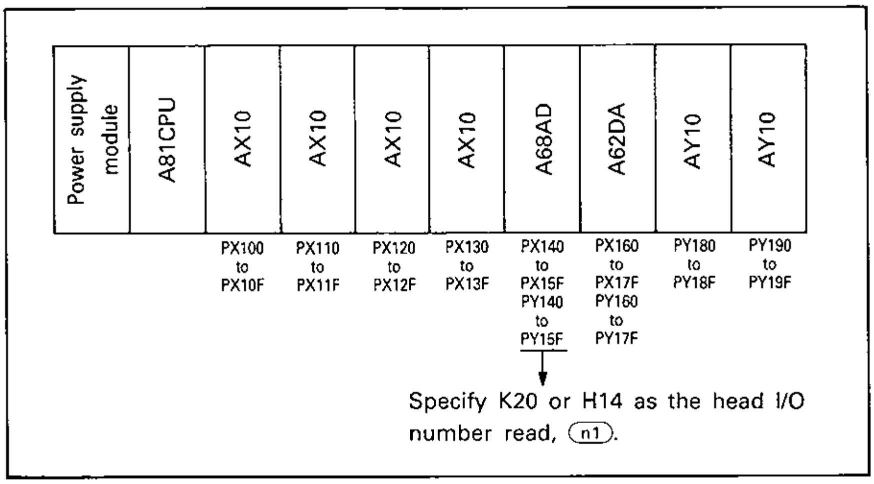
(1) Reads the number of words, (n3), from addresses headed by the specified address, (n2), of the buffer memory in the specified special function module, (n1), and stores the data to the devices headed by the specified device, Ⓓ.



RESTRICTION

(n3) should be within the allowed range of the specified device, Ⓓ, and that of the special function module buffer memory accessed.

(2) (n1) should be defined by the two most significant digits of the head I/O number assigned to the slot which accommodates the special function module.

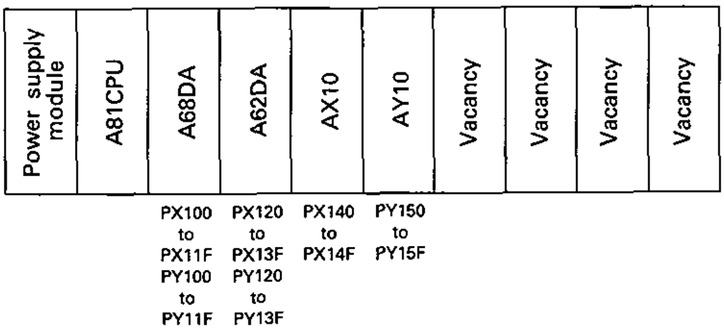


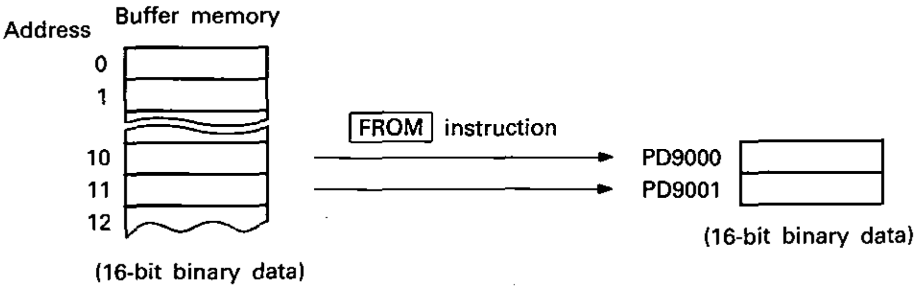
Specify K20 or H14 as the head I/O number read, (n1).

(3) 16-bit binary data is stored to the specified device, Ⓓ.

PROGRAM EXAMPLE

The following program reads 2 words to PD9000 from address 10 of the buffer memory in the A68AD loaded onto slot 0 of the main base unit.





```
0 FROM H 0010 K 10 PD 9000  K 2···Reads data from buffer memory
                                   addresses 10 and 11 to PD9000
                                   and PD9001.
5 END
```

HINTS

1) The DFRO instruction should be used if the buffer memory data is made up in blocks of 2 words.

2) Buffer memory data stored in blocks of 1 bit should be read as 16-bit binary data.

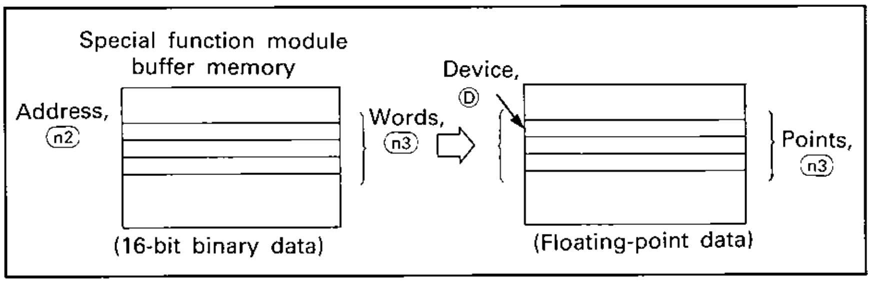
6.7.2 Reading data from special function module in blocks of 1 word (16-bit binary data to floating-point data) ······ FROM

FORMAT	FROM ⊔ (n1) ⊔ (n2) ⊔ Ⓓ ⊔ (n3)

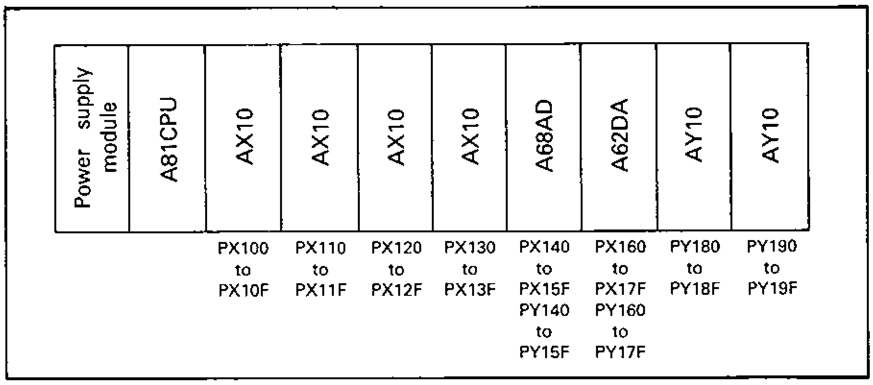
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(n1)	Two most significant digits of special function module head I/O number												○	○		5	○	○	○			○	
(n2)	Head address of buffer memory												○	○									
Ⓓ	Head device number for storing data read							○															
(n3)	Number of data read												○	○									

FUNCTIONS

(1) Reads the number of words, (n3), from addresses headed by the specified address, (n2), of the buffer memory in the specified special function module, (n1), and stores the data to the devices headed by the specified device, Ⓓ.



(2) (n1) should be defined by the two most significant digits of the head I/O number assigned to the slot which accommodates the special function module.



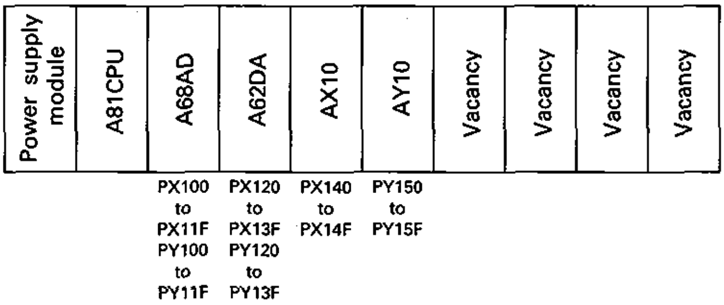
RESTRICTION

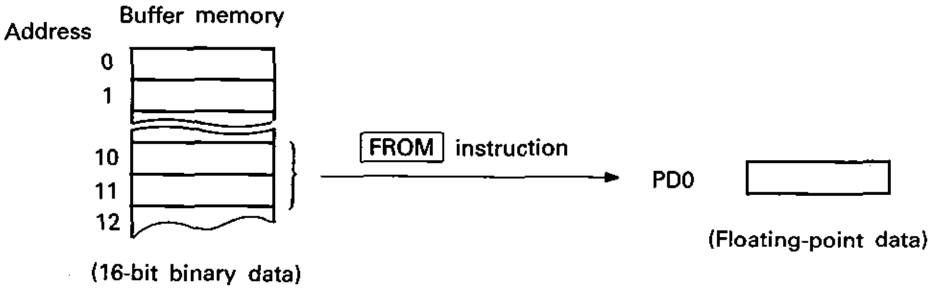
(n3) should be within the allowed range of the specified device, Ⓓ, and that of the special function module buffer memory accessed.

(3) 16-bit binary data in the buffer memory is converted into floating-point data and the result is stored to the specified device, Ⓓ.

PROGRAM EXAMPLE

The following program reads 2 words to PD9000 from address 10 of the buffer memory in the A68AD loaded onto slot 0 of the main base unit.





```
0 FROM H 0010  K 10  PD 0  K 2········Reads data from buffer memory
                                      addresses 10 and 11 to PD0.
5 END
```

HINT

The DFRO instruction should be used if the buffer memory data is made up in blocks of 2 words.

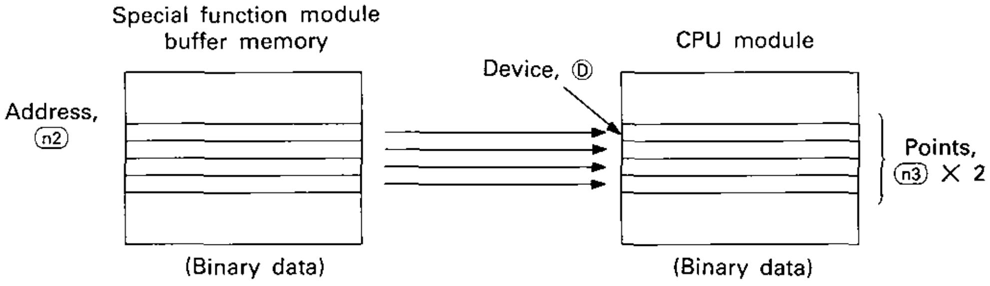
6.7.3 Reading data from special function module in blocks of 2 words (32-bit binary data to 32-bit binary data) ······ DFRO

FORMAT	DFRO ␣ (n1) ␣ (n2) ␣ Ⓓ ␣ (n3)

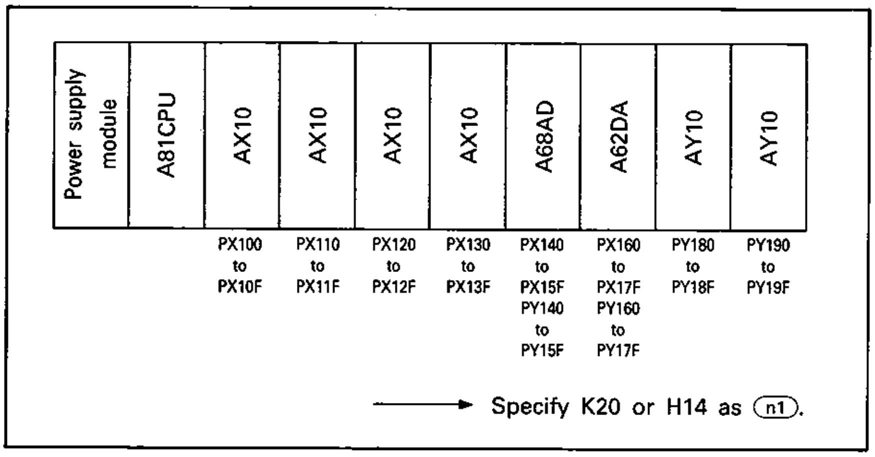
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(n1)	Two most significant digits of special function module head I/O number												○	○		5	○	○	○			○	
(n2)	Head address of buffer memory												○	○									
Ⓓ	Head device number for storing data read								○														
(n3)	Number of data read												○	○									

FUNCTIONS

(1) Reads the number of words, (n3) × 2, from addresses headed by the specified address, (n2), of the buffer memory in the specified special function module, (n1), and stores the data to the devices headed by the specified device, Ⓓ.



(2) n1 should be defined by the two most significant digits of the head I/O number assigned to the slot which accommodates the special function module.



→ Specify K20 or H14 as n1.

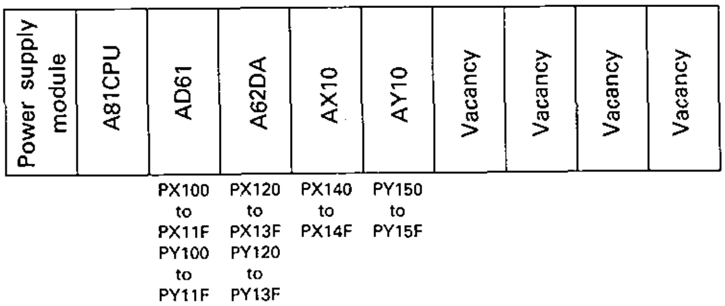
(3) Binary data is stored to the devices headed by the specified device, Ⓓ, in blocks of 2 devices.

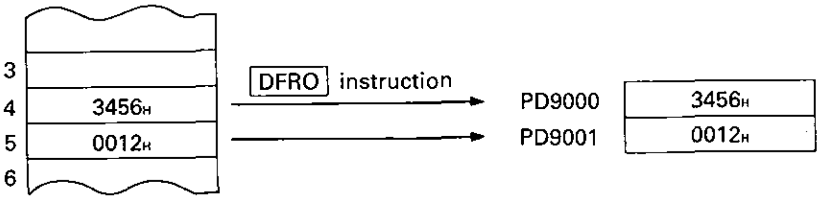
RESTRICTION

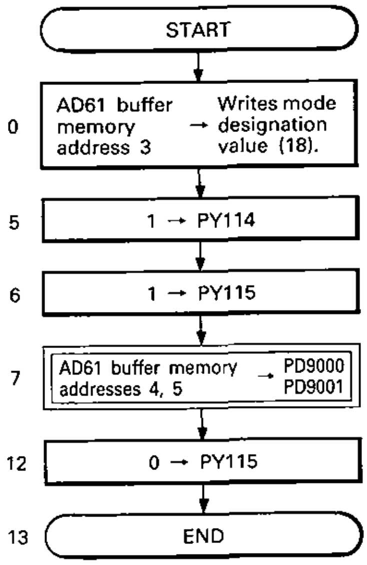
n3 should be within the allowed range of the specified device, Ⓓ, and that of the special function module buffer memory accessed.

PROGRAM EXAMPLE

The following program reads the CH1 present value to PD9000 and PD9001 from addresses 4 and 5 of the buffer memory in the AD61 loaded onto slot 0 of the main base unit.







............Switches on AD61 CH1 count enable.

............Switches on AD61 CH1 present value read request.

............Switches off AD61 CH1 present value read request.

```
 0 TO    H 0010  K 3  K 18  K 1·······Writes 2-phase input mode to
                                      address 3 of the buffer memory.
 5 SET   PY 114·······················Sets CH1 count enable PY114.
 6 SET   PY 115·······················Sets CH1 present value read re-
                                      quest PY115.
 7 DFRO  H 0010  K 4  PD 9000  K 1····Reads data from buffer memory
                                      addresses 4 and 5 to PD9000 and
                                      PD9001.
12 RST   PY 115·······················Resets CH1 present value read
                                      request PY115.
13 END
```

HINTS

DFRO is used to process 2-word data as binary.

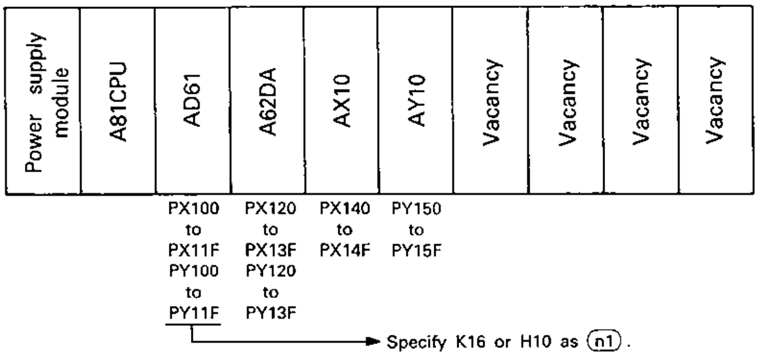
6.7.4 Reading data from special function module in blocks of 2 words (32-bit binary data to floating-point data) ······ DFRO

FORMAT	DFRO ␣ (n1) ␣ (n2) ␣ Ⓓ ␣ (n3)

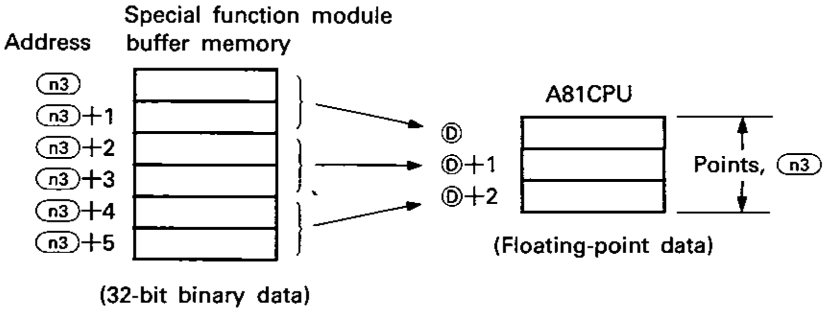
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(n1)	Two most significant digits of special function module head I/O number												○	○		5	○	○	○			○	
(n2)	Head address of buffer memory												○	○									
Ⓓ	Head device number for storing data read							○															
(n3)	Number of data read												○	○									

FUNCTIONS

(1) Reads data from addresses headed by the specified I/O address, (n1), of the special function module.
(n1) should be defined by the two most significant digits of the I/O address assigned to the special function module.



(2) 32-bit binary data stored in addresses headed by the specified address, (n2), is converted into floating-point data and the result is stored to the devices headed by the specified device, Ⓓ.



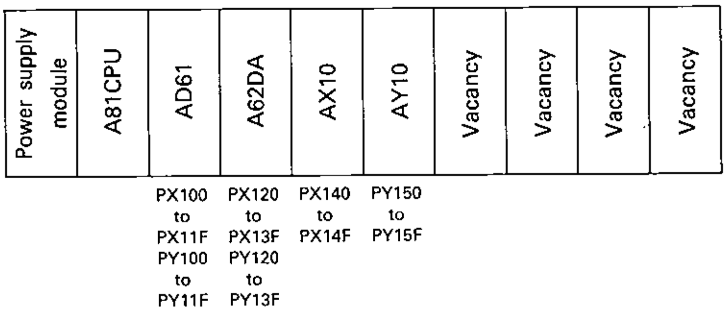
RESTRICTION

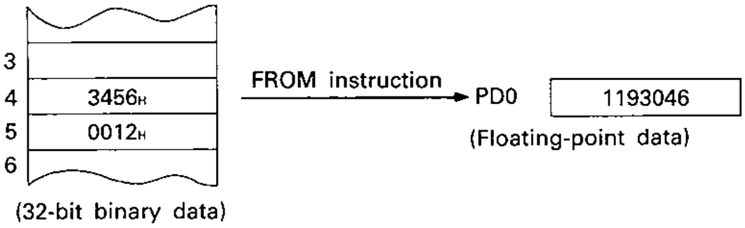
(n3) should be within the allowed range of the specified device, Ⓓ, and that of the special function module buffer memory accessed.

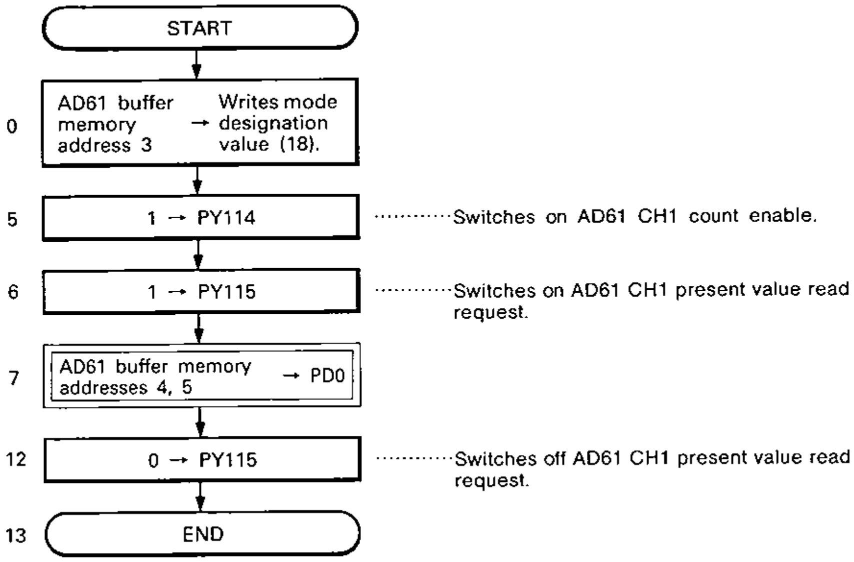
PROGRAM EXAMPLE

The following program reads the CH1 present value to PD0 from addresses 4 and 5 of the buffer memory in the AD61 loaded onto slot 0 of the main base unit.

Main base unit configuration







```
 0 TO     H 0010  K 3  K 18  K 1·······Writes 2-phase input mode to
                                       address 3 of the buffer memory.
 5 SET   PY 114·························Sets CH1 count enable PY114.
 6 SET   PY 115·························Sets CH1 present value read re-
                                       quest PY115.
 7 DFRO  H 0010  K 4  PD 0  K 1·······Reads data from buffer memory
                                       addresses 4 and 5 to PD0.
12 RST   PY 115 ························Resets CH1 present value read
                                       request PY115.
13 END
```

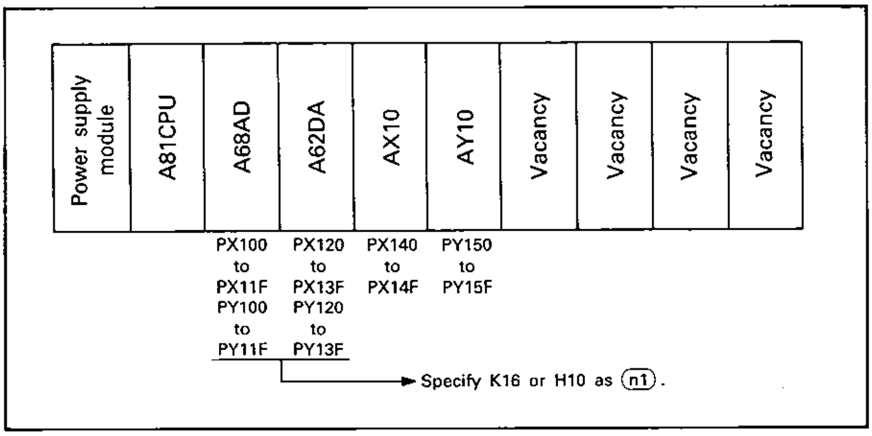
6.7.5 Writing data to special function module in blocks of 1 word (16-bit binary data to 16-bit binary data) ······ TO

FORMAT	TO ␣ (n1) ␣ (n2) ␣ (S) ␣ (n3)

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(n1)	Two most significant digits of special function module head I/O number												○	○		5	○	○	○			○	
(n2)	Head address of buffer memory												○	○									
(S)	Head device number containing data written								○		○		○	○									
(n3)	Number of data written												○	○									

FUNCTIONS

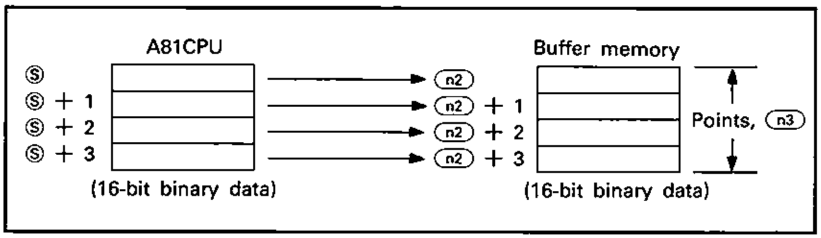
(1) Writes data to addresses headed by the specified I/O address, (n1), of the special function module.
(n1) should be defined by the two most significant digits of the I/O address assigned to the special function module.



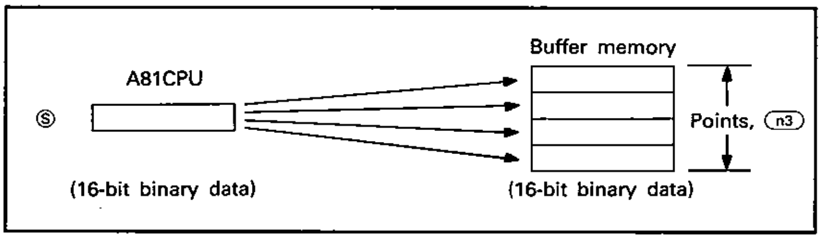
Specify K16 or H10 as (n1).

(2) The specified 16-bit binary data, Ⓢ, is written to the buffer memory addresses headed by the specified address, (n2).

(a) Ⓢ = device number



(b) Ⓢ = constant (K, H) or (A1)



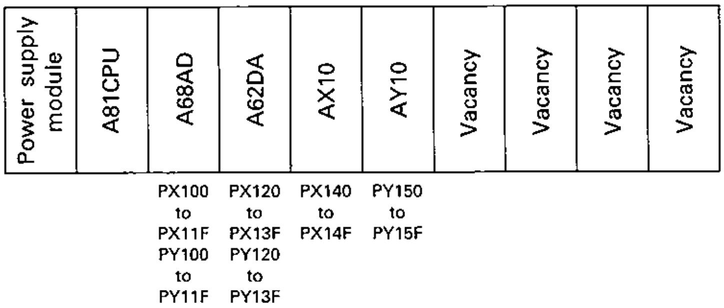
RESTRICTION

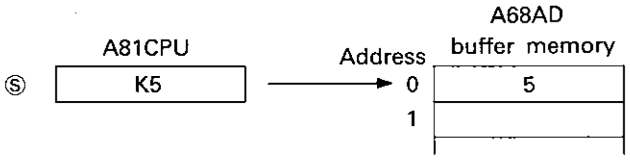
(n3) should be within the allowed range of the specified device, Ⓓ, and that of the special function module buffer memory accessed.

PROGRAM EXAMPLE

The following program writes 5 to address 0 (number of channels) of the buffer memory in the A68AD loaded onto slot 0 of the main base unit.

Main base unit configuration





```
0  TO   H 0010   K 0   K 5   K 1 ···········Writes 5 to buffer memory
                                             address 0.
5  END
```

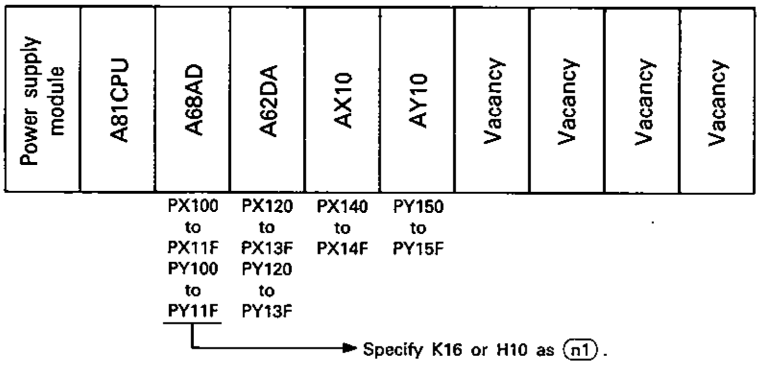
6.7.6 Writing data to special function module in blocks of 1 word (Floating-point data to 16-bit binary data) …… TO

FORMAT	TO ␣ (n1) ␣ (n2) ␣ Ⓢ ␣ (n3)

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(n1)	Two most significant digits of special function module head I/O number												○	○		5	○	○	○			○	
(n2)	Head address of buffer memory												○	○									
Ⓢ	Head device number containing data written or data written							○				○											
(n3)	Number of data written												○	○									

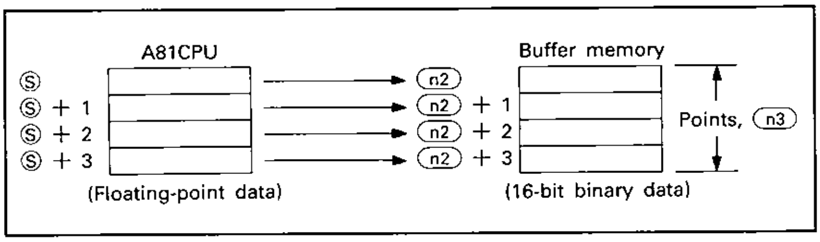
FUNCTIONS

(1) Writes data to addresses headed by the specified I/O address, (n1), of the special function module.
(n1) should be defined by the two most significant digits of the I/O address assigned to the special function module.

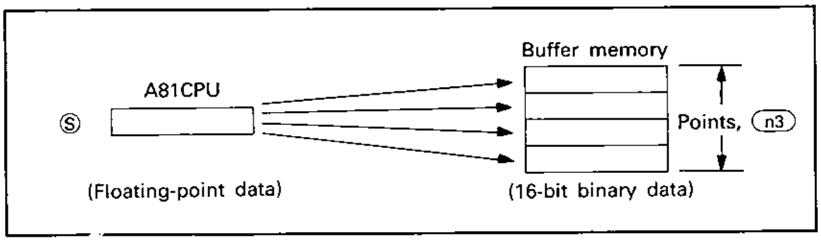


(2) The specified floating-point data, Ⓢ, is converted into 16-bit binary data and the result is written to the buffer memory addresses headed by the specified address, (n2).

(a) Ⓢ = device number



(b) Ⓢ = constant (K, H) or (A2)



RESTRICTION

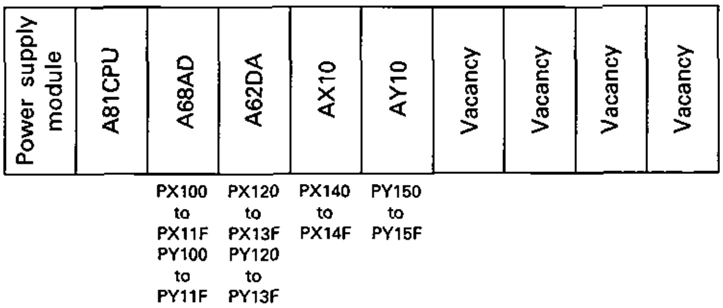
1) (n3) should be within the allowed range of the specified device, Ⓓ, and that of the special function module buffer memory accessed.

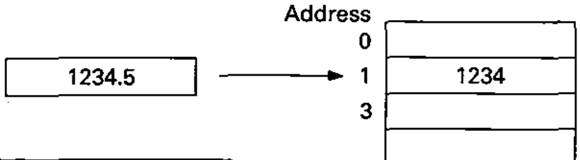
2) The specified device number data, Ⓢ, between −32768 and 32767 may only be converted into 16-bit binary data without any fault.

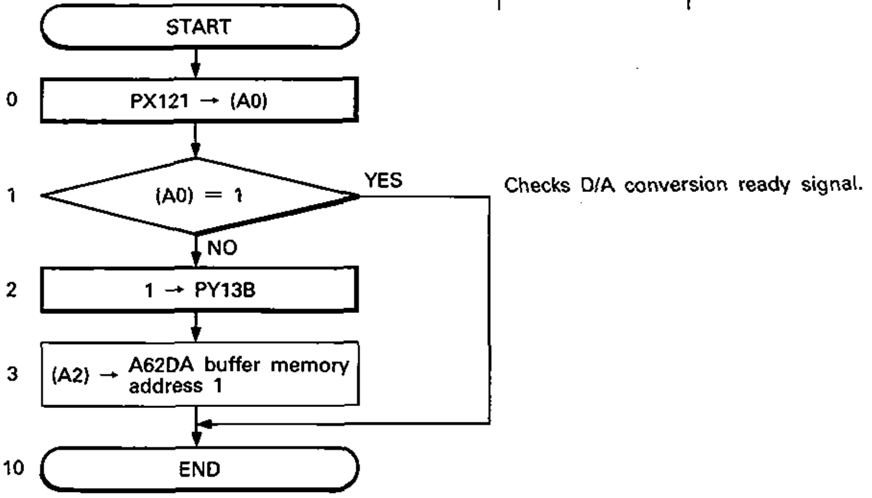
PROGRAM EXAMPLE

The following program writes floating-point data from (A2) to address 1 (CH2 digital value) of the buffer memory in the A62DA loaded onto slot 1of the main base unit. (Program 6 used)

Main base unit configuration







Checks D/A conversion ready signal.

```
0 NOT   PX 121·····································Reads D/A conversion ready
                                                   signal.
1 JC     P 0600····································Jumps to P0600 if D/A conversion
                                                   ready signal is off.
2 SET   PY 13B·····································Switches on output enable signal
                                                   PY13B.
3 TO     H 0012 K 1 A 2 K 1························Writes data from (A2) to buffer
                                                   memory address 1.
8 P     0600·······································Pointer P0600
10 END
```

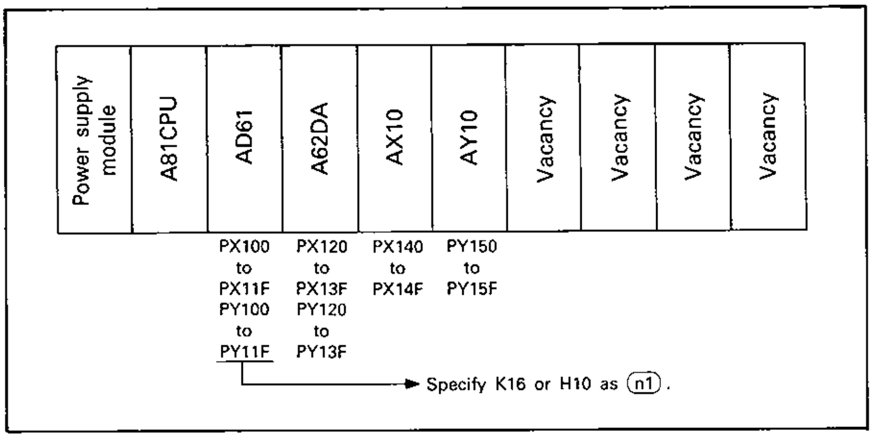
6.7.7 Writing data to special function module in blocks of 2 words (32-bit binary data to 32-bit binary data) ······ DTO

FORMAT	DTO ␣ (n1) ␣ (n2) ␣ Ⓢ ␣ (n3)

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(n1)	Two most significant digits of special function module head I/O number												○	○		5	○	○	○			○	
(n2)	Head address of buffer memory												○	○									
Ⓢ	Head device number containing data written								○														
(n3)	Number of data written												○	○									

FUNCTIONS

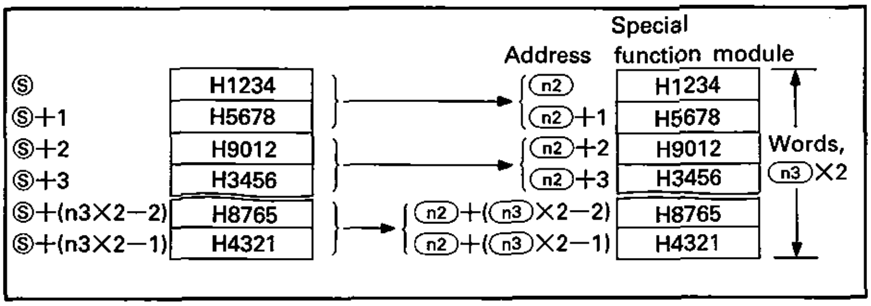
(1) Writes data to addresses headed by the specified I/O address, (n1), of the special function module.
(n1) should be defined by the two most significant digits of the I/O address assigned to the special function module.



RESTRICTION

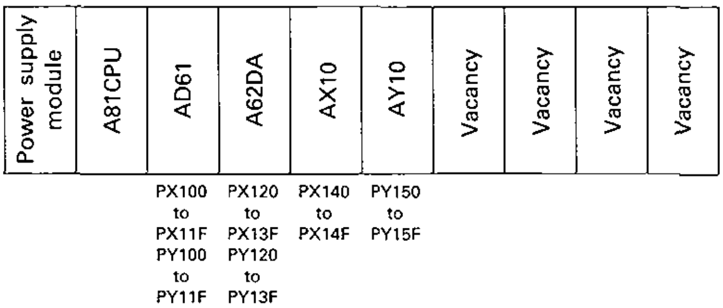
(n3) should be within the allowed range of the specified device, Ⓢ, and that of the special function module buffer memory accessed.

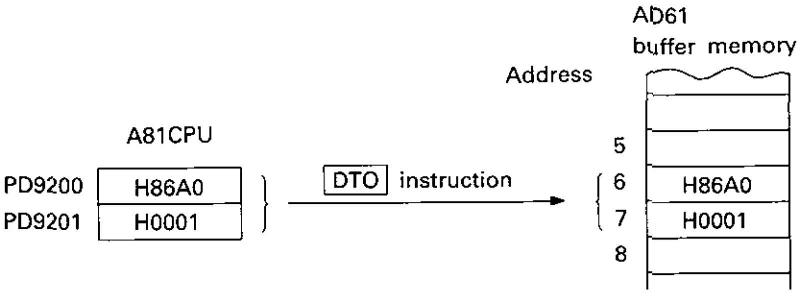
(2) The specified 32-bit binary data, Ⓢ, is written to the buffer memory 2-word area headed by the specified address, (n2).



PROGRAM EXAMPLE

The following program writes data from PD9000 and PD9001 to addresses 6 and 7 (CH1 set value) of the buffer memory in the AD61 loaded onto slot 0 of the main base unit.





```
 0  MOV  H 86A0  PD 9000························Sets 86A0H to PD9000.
 3  MOV  H 0001  PD 9001························Sets 0001H to PD9001.
 6  DTO  H 0010  K 6 PD 9000  K 1··········Writes data from PD9000, PD9001
                                            to buffer memory addresses 6, 7.
11  END
```

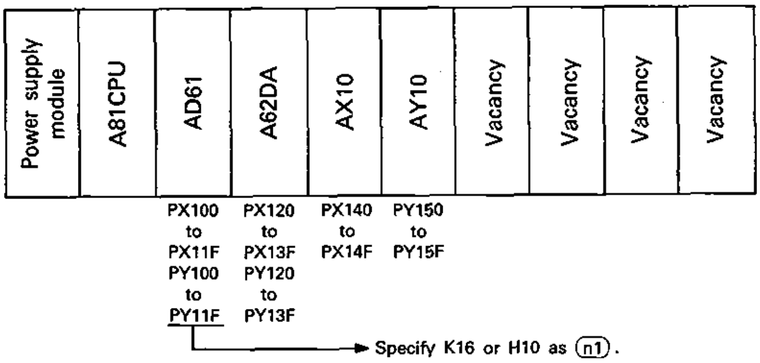
6.7.8 Writing data to special function module in blocks of 2 words (Floating-point data to 32-bit binary data) ······ DTO

FORMAT	DTO ␣ (n1) ␣ (n2) ␣ Ⓢ ␣ (n3)

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(n1)	Two most significant digits of special function module head I/O number												○	○		5	○	○	○			○	
(n2)	Head address of buffer memory												○	○									
Ⓢ	Head device number containing data written or data written							○				○	○										
(n3)	Number of data written												○	○									

FUNCTIONS

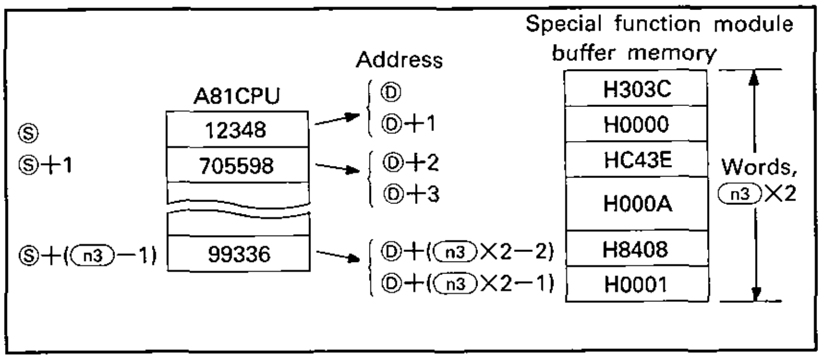
(1) Writes data to addresses headed by the specified I/O address, (n1), of the special function module.
(n1) should be defined by the two most significant digits of the I/O address assigned to the special function module.



Specify K16 or H10 as (n1).

(2) *The specified floating-point data,* Ⓢ, is converted into 32-bit binary data and the result is written to the buffer memory addresses headed by the specified address, n2.

(a) Ⓢ = device number

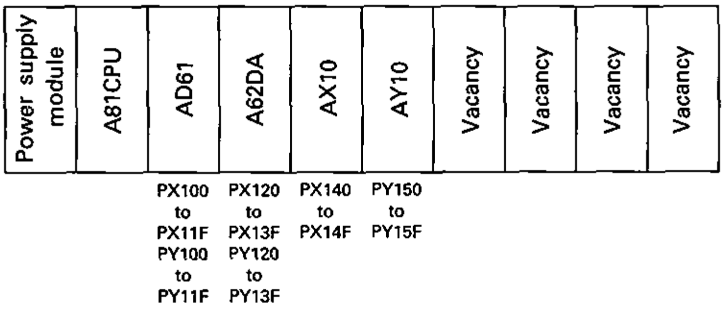


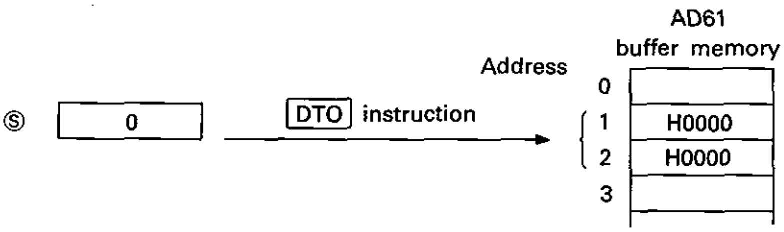
RESTRICTION

n3 should be within the allowed range of the specified device, Ⓓ, and that of the special function module buffer memory accessed.

PROGRAM EXAMPLE

The following program writes 0 to addresses 1 and 2 (CH1 preset value) of the buffer memory in the AD61 loaded onto slot 0 of the main base unit.

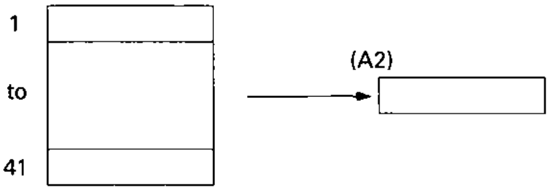
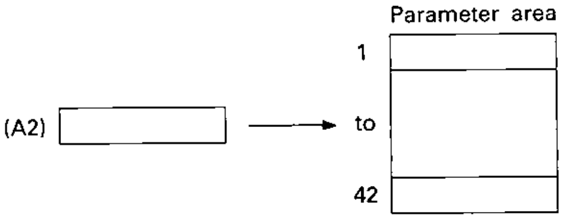




```
0 BMOV PX 100  PD 9000  K 1 ············Clears (A2) to 0.
4 LDAW PD 9000 ·························Stores (A2) data to buffer memory
                                        addresses 1, 2.
5 WXOR  H 0080
6 STAW PD 9000
7 END
```

6.8 Macro Function Parameter Read/Write Instructions

Used to read and write macro function parameters using the user program.

Instruction	Description	Refer To
PRR	Reads the macro function parameter to (A2) in accordance with the specified loop number and data number.  	Section 8.6.1
PRW	Writes (A2) data to the macro function parameter in accordance with the specified loop number and data number.  	Section 8.6.2

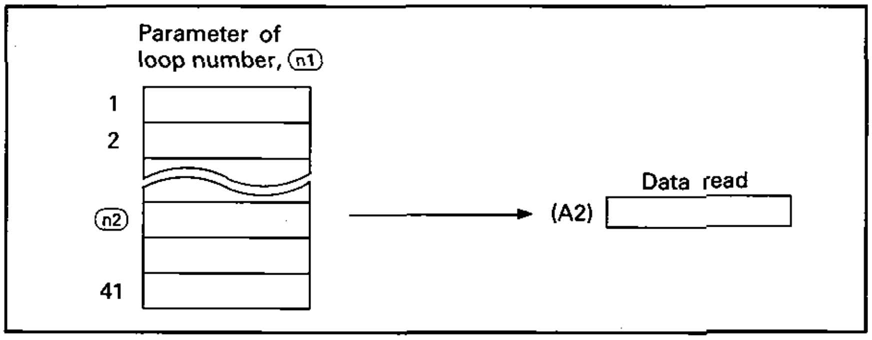
6.8.1 Reading the macro function parameter ······ PRR

FORMAT	PRR ␣ (n1) ␣ (n2)

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(n1)	Loop number read												○			3						○	
(n2)	Data number read												○										

FUNCTIONS

(1) Reads the macro function parameter to (A2) in accordance with the specified loop number (1 to 64), (n1), and data number (1 to 41), (n2).



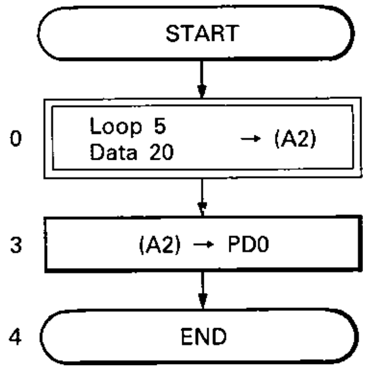
(2) When data 41 is specified, "ΔMV" of the specified loop is read to (A2).

Restrictions

1) The loop number specified as (n1) is between 1 and 64.

2) The data number specified as (n2) is between 1 and 41.

PROGRAM EXAMPLE The following program reads the PV change rate alarm set value (data 20) of loop 5 to PD0.



```
0 PRR  K 5   K 20··································Reads data 20 of loop 5 to (A2).
3 STAF PD 0 ············································Transfers (A2) data to PD0.
4 END
```

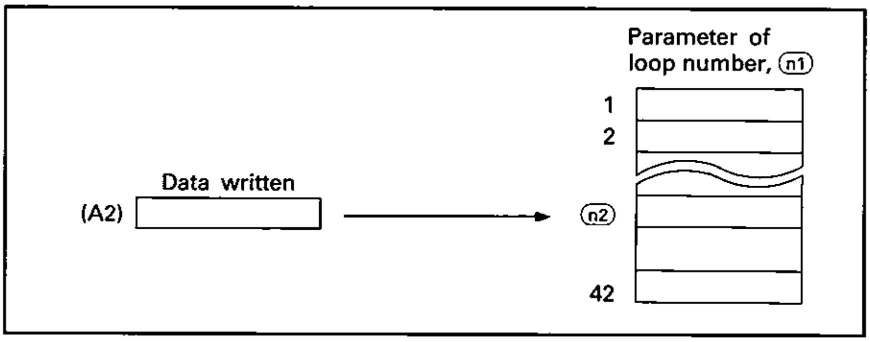
6.8.2 Writing the macro function parameter ······ PRW

FORMAT	PRW ␣ (n1) ␣ (n2)

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(n1)	Loop number written												○			3						○	
(n2)	Loop number written												○										

FUNCTIONS

(1) Writes the (A2) data to the macro function parameter in accordance with the specified loop number (1 to 64), (n1), and data number (1 to 42), (n2).



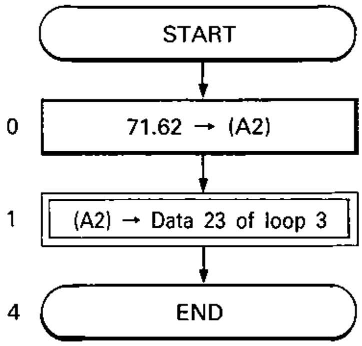
(2) When data 42 is specified, "EV$_{n-1}$, PVf$_{n-1}$, PVf$_{n-2}$, $\Sigma \Delta$MV, ΔD$_{n-1}$" of the specified loop is cleared to zero, independently of the A2 data.

Restrictions

1) The loop number specified as (n1) is between 1 and 64.

2) The data number specified as (n2) is between 1 and 42.

PROGRAM EXAMPLE The following program sets the MV high limit value (data 23) of loop 3 to 71.62%.



```
0 LDAF   K 71.62 ································Sets 71.62 to (A2).
1 PRW    K 3     K 23 ··························Transfers (A2) data to data 1 of
                                                 loop 3.
4 END
```

6.9 Comparison Instructions

Any of the comparison instructions compares accumulator A1 (16-bit BIN data) or A2 (32-bit BIN data) with the specified device to determine which instruction to be executed, the one at the next step or the step after the next.

Instruction	Description	Refer To
GTAW	Compares (A1) with the specified device, Ⓢ, and performs either of the following processings in accordance with the result. (A1)>Ⓢ→ Executes the instruction at the next step. (A1)≦Ⓢ→ Executes the instruction at the step after the next.	Section 6.9.1
GTAF	Compares (A2) with the specified device, Ⓢ, and performs either of the following processings in accordance with the result. (A2)>Ⓢ→ Executes the instruction at the next step. (A2)≦Ⓢ→ Executes the instruction at the step after the next.	Section 6.9.2
LTAW	Compares (A1) with the specified device, Ⓢ, and performs either of the following processings in accordance with the result. (A1)<Ⓢ→ Executes the instruction at the next step. (A1)≧Ⓢ→ Executes the instruction at the step after the next.	Section 6.9.3
LTAF	Compares (A2) with the specified device, Ⓢ, and performs either of the following processings in accordance with the result. (A2)<Ⓢ→ Executes the instruction at the next step. (A2)≧Ⓢ→ Executes the instruction at the step after the next.	Section 6.9.4
EQAW	Compares (A1) with the specified device, Ⓢ, and performs either of the following processings in accordance with the result. (A1)=Ⓢ→ Executes the instruction at the next step. (A1)≠Ⓢ→ Executes the instruction at the step after the next.	Section 6.9.5
EQAF	Compares (A2) with the specified device, Ⓢ, and performs either of the following processings in accordance with the result. (A2)=Ⓢ→ Executes the instruction at the next step. (A2)≠Ⓢ→ Executes the instruction at the step after the next.	Section 6.9.6

6.9.1 Data comparison with (A1) (>) …… GTAW

FORMAT	GTAW ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant compared with A1					○			○				○	○		2						○	

FUNCTIONS

(1) Compares accumulator (A1) with the specified device or constant, Ⓢ. The step executed depends on the comparison result as indicated below:

(A1) > Ⓢ	(A1) > Ⓢ	Executes the instruction at the next step.
	(A1) ≦ Ⓢ	Executes the instruction at the step after the next.

(2) The data in the specified device, Ⓢ, remains unchanged after the GTAW instruction is executed.

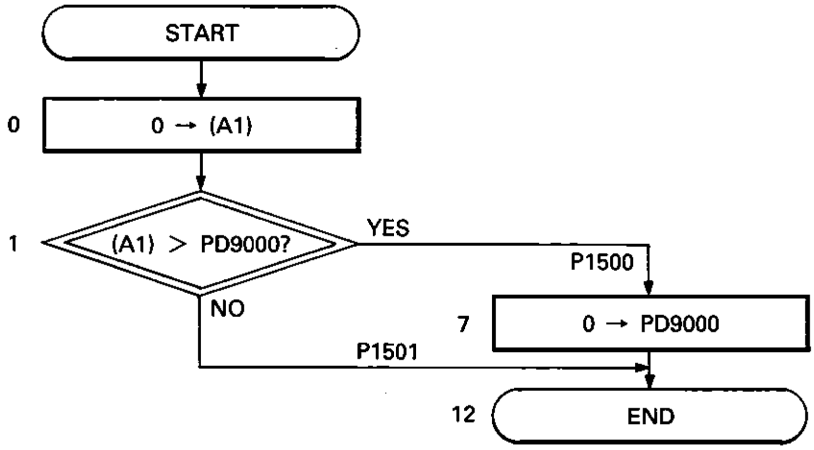
RESTRICTIONS

1) The instruction used at the step after the GTAW instruction should be of one step. An operation error will occur if the instruction used consists of two or more steps.

(2) The constant used with the GTAW instruction is between −32768 and 32767 or between 0000H and FFFFH.

PROGRAM EXAMPLE

The following program changes any negative value in PD9000 to 0. (Program 15 used)



```
 0 LDAW   K 0 ..............................................Stores 0 to (A1).
 1 GTAW  PD 9000 ...........................................Compares (A1) with PD9000.
 3 JMP    P 1500 ...........................................Jumps to pointer P1500 if PD9000
                                                            value is less than 0.
 4 JMP    P 1501 ...........................................Jumps to pointer P1501 if PD9000
                                                            value is greater than or equal to 0.
 5 P     1500
 7 MOV    K 0     PD 9000 ..................................Stores 0 to PD9000.
10 P     1501
12 END
```

6.9.2 Data comparison with (A2) (>) ······ GTAF

FORMAT	**GTAF ␣ Ⓢ**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant compared with (A2)					○		○					○	○		1						○	

FUNCTIONS

(1) Compares (A2) with the specified device or constant, Ⓢ. The step executed depends on the comparison result as indicated below:

(A2) > Ⓢ	(A2) > Ⓢ	Executes the instruction at the next step.
	(A2) ≦ Ⓢ	Executes the instruction at the step after the next.

(2) The data in the specified device, Ⓢ, remains unchanged after the GTAF instruction is executed.

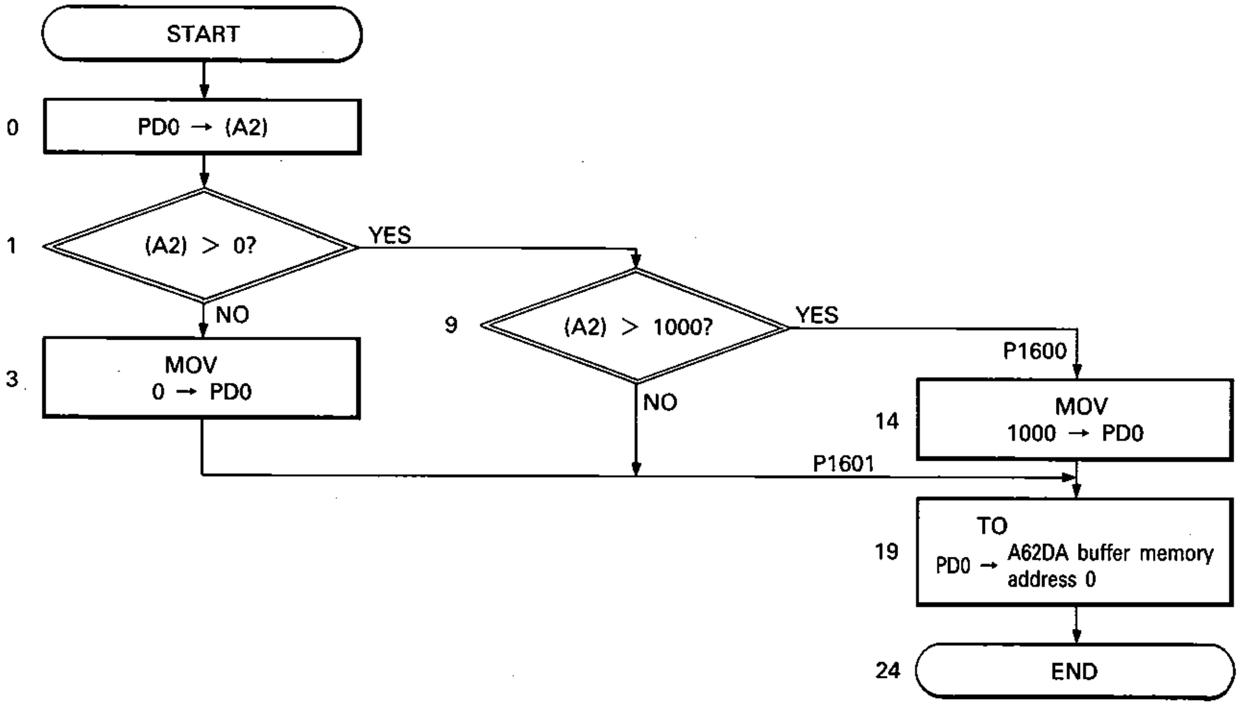
RESTRICTIONS

1) The instruction used at the step after the GTAF instruction should be of one step. An operation error will occur if the instruction used consists of two or more steps.

2) The data used with the GTAF instruction is between ± 2.7 × 10^{-20} and ± 9.2 × 10^{18}.
The constant specified during programming is between ± 1 ×10^{-9} and ± 9.999×10^{9} or between 0000_H and $FFFF_H$.

PROGRAM EXAMPLE

The following program changes any PD0 value less than 0 to 0 and any value greater than 1000 to 1000, and outputs the value from channel 1 of the A62DA. (Program 16 used)



```
 0 LDAF PD 0·········································Stores PD0 data to (A2).
 1 GTAF  K 0·········································Compares (A2) data with 0.
 2 JMP   P 1600······································Jumps to pointer P1600.
 3 MOV   K 0     PD 0································Stores 0 to PD0.
 6 JMP   P 1602······································Jumps to pointer P1602.
 7 P     1600
 9 GTAF  K 1000······································Compares (A2) data with 1000.
10 JMP   P 1601······································Jumps to pointer P1601 if the PD0
                                                     value is greater than 1000.
11 JMP   P 1602······································Jumps to pointer P1602 if the PD0
12 P     1601                                        value is less than or equal to 1000.
14 MOV   K 1000    PD 0······························Stores 1000 to PD0.
17 P     1602
19 TO    H 0018  K 0  PD 0  K 1······················Writes PD0 data to buffer memory
24 END                                               address 0.
```

6.9.3 Data comparison with (A1) (<) ······ LTAW

FORMAT	LTAW ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant compared with (A1)					○			○				○	○		2						○	

FUNCTIONS

(1) Compares (A1) with the specified device or constant, Ⓢ. The step executed depends on the comparison result as indicated below:

(A1) < Ⓢ	(A1) < Ⓢ	Executes the instruction at the next step.
	(A1) ≧ Ⓢ	Executes the instruction at the step after the next.

(2) The data in the specified device, Ⓢ, remains unchanged after the LTAW instruction is executed.

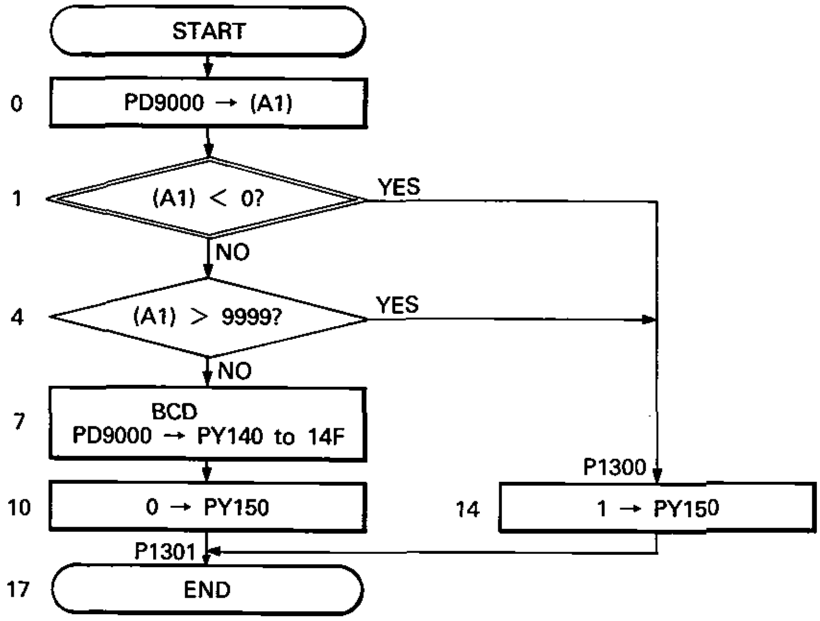
RESTRICTIONS

1) The instruction used at the step after the LTAW instruction should be of one step. An operation error will occur if the instruction used consists of two or more steps.

2) The constant used with the LTAW instruction is between −32768 and 32767 or between 0000H and FFFFH.

PROGRAM EXAMPLE

The following program converts the PD9000 value into BCD and outputs the result to PY140 to 14F if that value is between 0 and 9999, and switches on PY150 if that value is outside the above range. (Program 13 used)



```
 0 LDAW  PD 9000·····································Stores PD9000 data to (A1).
 1 LTAW   K 0········································Compares (A1) data with 0.
 3 JMP    P 1300·····································Jumps to pointer P1300 if PD9000
                                                      data is less than 0.
 4 GTAW   K 9999 ····································Compares (A1) data with 9999 if
                                                      PD9000 data is greater than or
                                                      equal to 0.
 6 JMP    P 1300 ····································Jumps to pointer P1300 if PD9000
                                                      data is greater than 9999.
 7 BCD   PD 9000     PY 140 ·························Converts PD9000 value into BCD
                                                      and outputs the result to PY140 to
                                                      14F if PD9000 value is between 0
                                                      and 9999.
10 RST   PY 150······································Switches off PY150.
11 JMP    P 1301·····································Jumps to pointer P1301.
12 P     1300
14 SET   PY 150······································Switches on PY150.
15 P     1301
17 END
```

6.9.4 Data comparison with (A2) (<) …… LTAF

FORMAT	**LTAF ␣ Ⓢ**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant compared with (A2)					○		○					○	○		1						○	

FUNCTIONS

(1) Compares (A2) with the specified device or constant, Ⓢ. The step executed depends on the comparison result as indicated below:

(A2) < Ⓢ	(A2) < Ⓢ	Executes the instruction at the next step.
	(A2) ≧ Ⓢ	Executes the instruction at the step after the next.

(2) The data in the specified device, Ⓢ, remains unchanged after the LTAF instruction is executed.

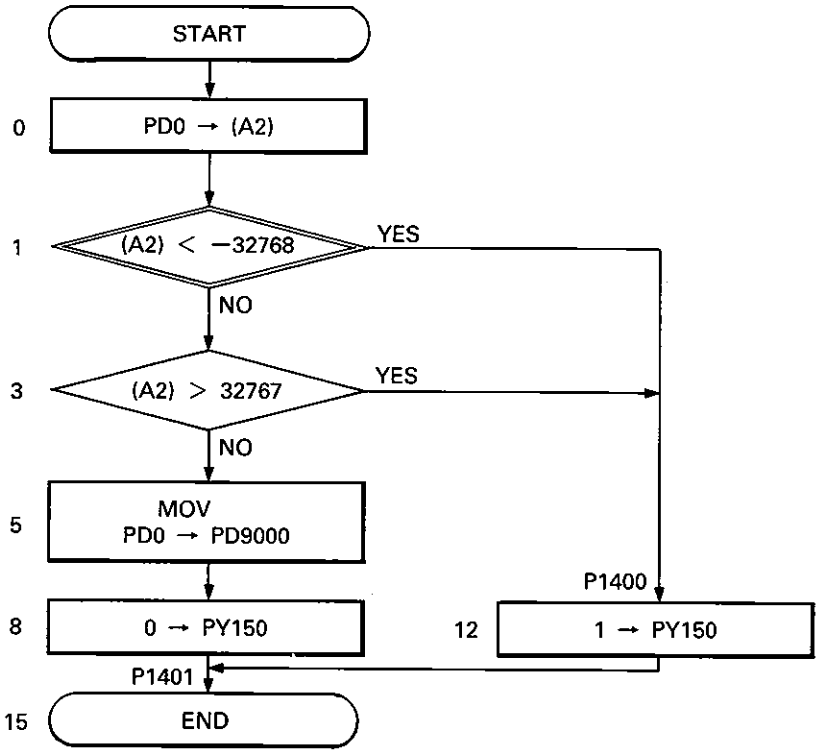
RESTRICTIONS

1) The instruction used at the step after the LTAF instruction should be of one step. An operation error will occur if the instruction used consists of two or more steps.

2) *The data used with the* LTAF *instruction is between* $\pm 2.7 \times 10^{-20}$ and $\pm 9.2 \times 10^{18}$.
The constant specified during programming is between $\pm 1 \times 10^{-9}$ and $\pm 9.999 \times 10^{9}$ or between 0000_H and $FFFF_H$.

PROGRAM EXAMPLE

The following program transfers the PD0 value to PD9000 and switches on PY150 if that value is between −32768 and 32767 if that value is outside the above range. (Program 14 used)



```
 0 LDAF PD 0·····························Stores PD0 data to (A2).
 1 LTAF  K-32768·························Compares (A2) data with −32768.
 2 JMP   P 1400··························Jumps to pointer P1400 if PD0
                                         data is less than −32768.
 3 GTAF  K 32767·························Compares (A2) data with 32767.
 4 JMP   P 1400··························Jumps to pointer P1400 if PD0
                                         data is greater than 32767.
 5 MOV  PD 0       PD 9000···············Transfers PD0 data to PD9000.
 8 RST  PY 150···························Switches off PY150.
 9 JMP   P 1401··························Jumps to pointer P1401.
10 P    1400
12 SET  PY 150···························Switches on PY150.
13 P    1401
15 END
```

6.9.5 Data comparison with (A1) (=) ······ EQAW

FORMAT	EQAW ␣ Ⓢ

	Set Data	Set Device															Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P			51	54	55	56	57	58	59
Ⓢ	Device number or constant compared with (A1)					○			○				○	○			2						○	

FUNCTIONS

(1) Compares (A1) with the specified device or constant, Ⓢ. The step executed depends on the comparison result as indicated below:

(A1) = Ⓢ	(A1) = Ⓢ	Executes the instruction at the next step.
	(A1) ≠ Ⓢ	Executes the instruction at the step after the next.

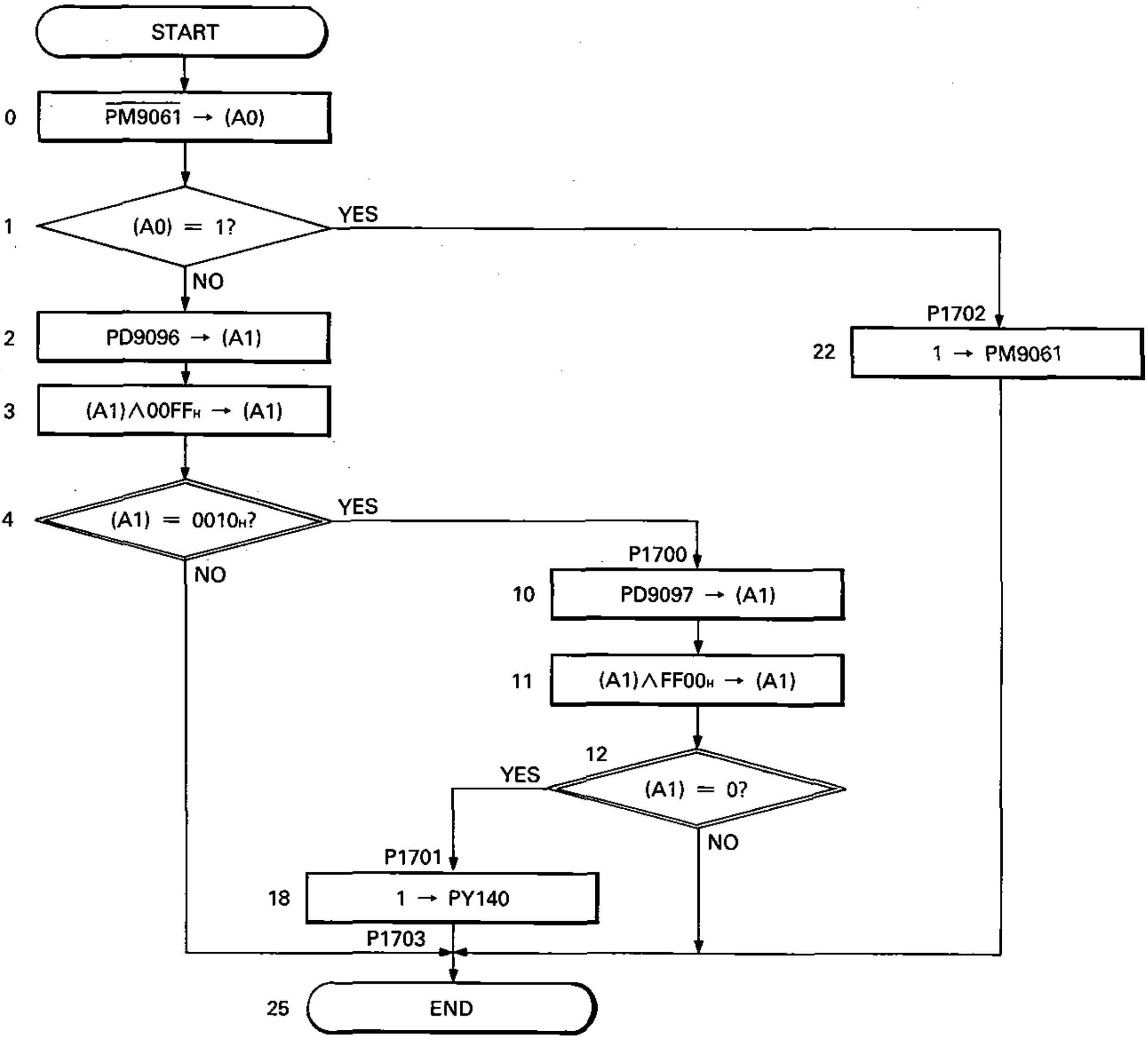
(2) The data in the specified device, Ⓢ, remains unchanged after the EQAW instruction is executed.

RESTRICTIONS

1) The instruction used at the step after the EQAW instruction should be of one step. An operation error will occur if the instruction used consists of two or more steps.

2) *The constant used with the* EQAW instruction is between −32768 and 32767 or between 0000_H and $FFFF_H$.

PROGRAM EXAMPLE The following program switches on PY140 when the clock reaches 10:00. (Program 17 used)



```
 0 NOT   PM 9061···················Complements PM9061 and stores
                                   the result to (A0).
 1 JC     P 1702···················Jumps to pointer P1702 if PM9061
                                   data is off.
 2 LDAW  PD 9096 ··················Stores PD9096 data to (A1).
 3 WAND   H 00FF ··················ANDs (A1) and 00FFH data and
                                   stores the result to (A1).
 4 EQAW   H 0010···················Compares (A1) data with 0010H.
 6 JMP    P 1700···················Jumps to pointer P1700 if (A1) =
                                   0010H.
 7 JMP    P 1703···················Jumps to pointer P1703 if (A1) =
                                   0010H.
 8 P       1700
10 LDAW  PD 9097 ··················Stores PD9097 data to (A1).
11 WAND   H FF00 ··················ANDs (A1) and FF00H data and
                                   stores the result to (A1).
12 EQAW   H 0000 ··················Compares (A1) data with 0.
14 JMP    P 1701 ··················Jumps to pointer P1701 if (A1) =
                                   0.
15 JMP    P 1703···················Jumps to pointer P1703 if (A1) =
                                   0.
16 P       1701
18 SET   PY 140  ··················Switches on PY140.
19 JMP    P 1703···················Jumps to pointer P1703.
20 P       1702
22 SET   PM 9061···················Switches on PM9061 (clock data
                                   read request).
23 P       1703
25 END
```

6.9.6 Data comparison with (A2) (=) …… EQAF

FORMAT	EQAF ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant compared with A2					○		○					○	○		1						○	

FUNCTIONS

(1) Compares (A2) with the specified device or constant, Ⓢ. The step executed depends on the comparison result as indicated below:

(A2) = Ⓢ	(A2) = Ⓢ	Executes the instruction at the next step.
	(A2) ≠ Ⓢ	Executes the instruction at the step after the next.

(2) The data in the specified device, Ⓢ, remains unchanged after the EQAF instruction is executed.

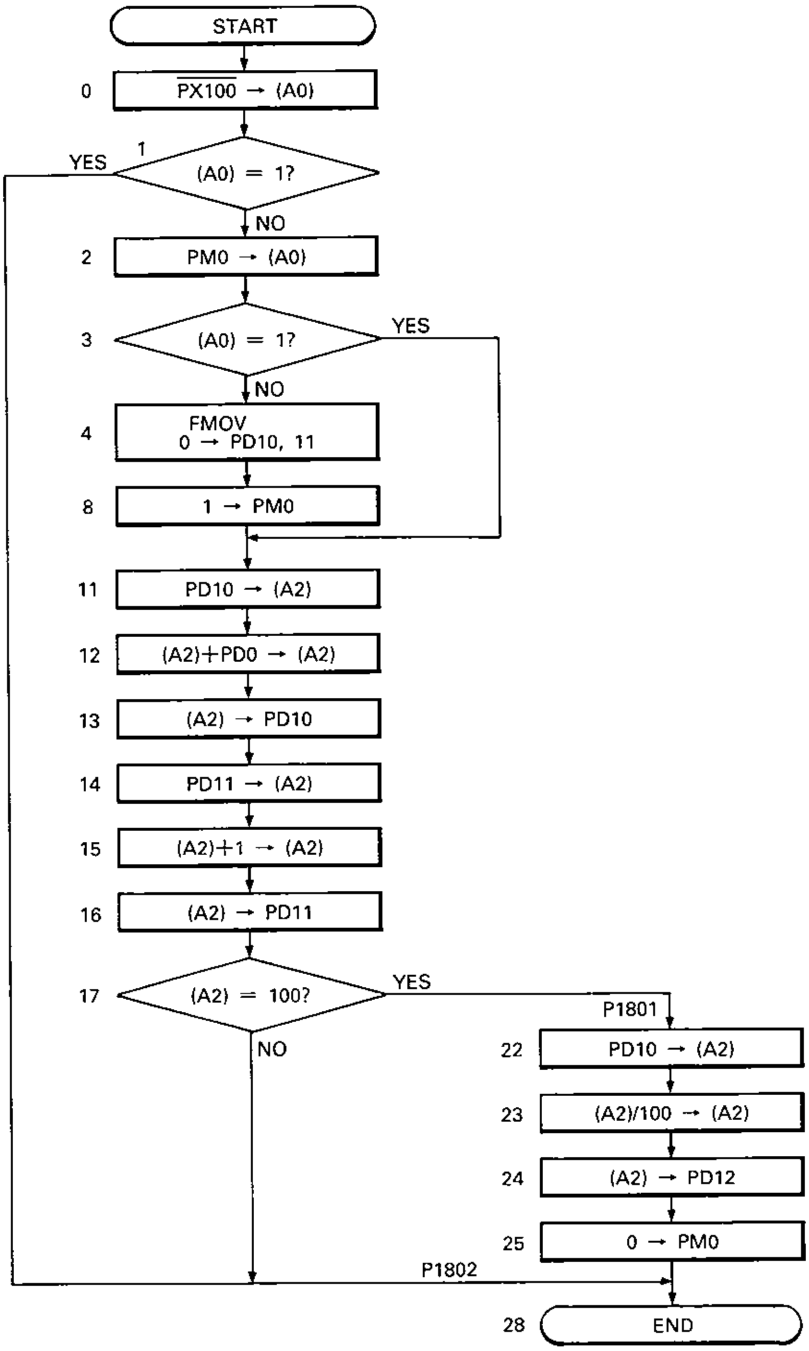
RESTRICTIONS

1) The instruction used at the step after the EQAF instruction should be of one step. An operation error will occur if the instruction used consists of two or more steps.

2) The data used with the EQAF instruction is between ± 2.7 $\times 10^{-20}$ and ± 9.2 $\times 10^{18}$.
The constant specified during programming is between ± 1 $\times 10^{-9}$ and 9.999$\times 10^{9}$ or between 0000H and FFFFH.

PROGRAM EXAMPLE

The following program switches on PX100, samples the PD0 value 100 times, and stores the mean value to PD12. (Program 18 used)



```
 0 NOT   PX 100 ······························Flips PX100 data and stores the
                                               result to (A0).
 1 JC     P 1802······························Jumps to pointer P1802 if PX100
                                               is off.
 2 LDAB  PM 0·································Stores PX100 data to (A0).
 3 JC     P 1800······························Jumps to pointer P1800 if PM0 is
                                               on.
 4 FMOV   K 0  PD 10      K 2·················Stores 0 to PD10, 11.
 8 SET   PM 0·································Switches on PM0.
 9 P     1800
11 LDAF  PD 10································Stores PD10 data to (A2).
12 +      PD 0································Adds (A2) and PD0 data and
                                               stores the result to (A2).
13 STAF  PD 10································Stores (A2) data to PD10.
14 LDAF  PD 11································Stores PD11 data to (A2).
15 +      K 1·································Adds 1 to (A2) data and stores the
                                               result to (A2).
16 STAF  PD 11································Stores (A2) data to PD11.
17 EQAF   K 100·······························Compares (A2) data against 100.
18 JMP    P 1801······························Jumps to pointer P1801 if (A2) =
                                               100.
19 JMP    P 1802······························Jumps to pointer P1802 if (A2) =
                                               100.
20 P     1801·································Stores PD10 data to (A2).
22 LDAF  PD 10
23 /      K 100·······························Divides (A2) data by 100 and
                                               stores the result to (A2).
24 STAF  PD 12································Stores (A2) data to PD12.
25 RST   PM 0·································Switches off PM0.
26 P     1802
28 END
```

6

6.10 Branch Instructions

Used to cause a branch, e.g. to jump within a program, to call another program.

Instruction	Description	Refer To
JMP	Causes an unconditional jump to a program step specified by the pointer.	Section 6.10.1
JC	Causes a jump to a program step specified by the pointer if the (A0) data is 1. Executes the program at the next step if the (A0) data is 0.	Section 6.10.2
CALL	Execute the subroutine program specified by the pointer.	Section 6.10.3
RET	Returns execution from the subroutine program to the previous program.	Section 6.10.4

6.10.1 Unconditional jump ······ JMP

FORMAT	JMP ␣ P ＊＊＊＊

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
P	Jump destination pointer number														○	1				○			

FUNCTIONS

(1) Unconditionally executes the program specified by the pointer.

Instruction	Result
JMP P ＊＊＊＊	Jump to the specified pointer.

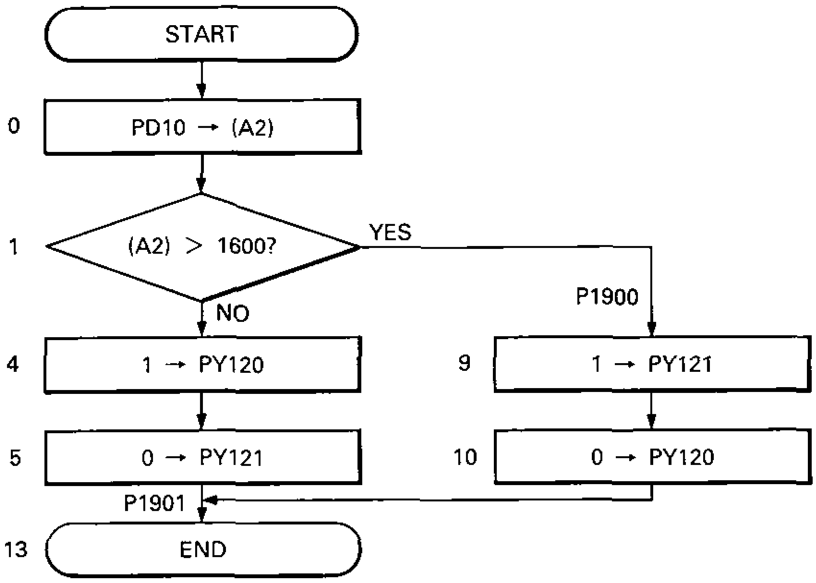
(2) A jump may be made between programs.

RESTRICTIONS

An error is flagged if the pointer specified by JMP P ＊＊＊＊ does not exist in the program.

PROGRAM EXAMPLE

The following program switches on PY121 if the PD10 value is greater than 1600 in program 1, and switches on PY120 if that value is equal to or less than 1600. (Program 19 used)



```
 0 LDAF PD 10 ··································Transfers PD10 data to (A2).
 1 GTAW  K 1600 ································Executes the next step if (A2) data
                                                is greater than 1600, and executes
                                                the step after the next if that data
                                                is less than or equal to 1600.
 3 JMP   P 1900 ································Jumps to pointer P1900 if (A2)
                                                data is greater than 1600.
 4 SET  PY 120 ·································Switches on PY120 if (A2) data is
                                                equal to or less than 1600.
 5 RST  PY 121 ·································Switches off PY121 if (A2) data is
                                                equal to or less than 1600.
 6 JMP   P 1901 ································Jumps to pointer P1901 if (A2)
                                                data is equal to or less than 1600.
 7 P    1900
 9 SET  PY 121 ·································Switches on PY121 if (A2) data is
                                                greater than 1600.
10 RST  PY 120 ·································Switches off PY120 if (A2) data is
                                                greater than 1600.
11 P    1901
13 END
```

6.10.2 Conditional jump ······ JC

FORMAT	JC ␣ P ✻✻✻✻

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
P	Jump destination pointer number														○	1				○			

FUNCTIONS

(1) Jumps to the specified pointer if the condition is enabled [(A0) = 1].

(2) Executes the instruction at the next step if the condition is disabled [(A0) = 0].

Instruction	Condition	Result
JC P ✻✻✻✻	(A0) = 1	Jumps to the specified pointer.
	(A0) = 0	Executes the instruction at the next step.

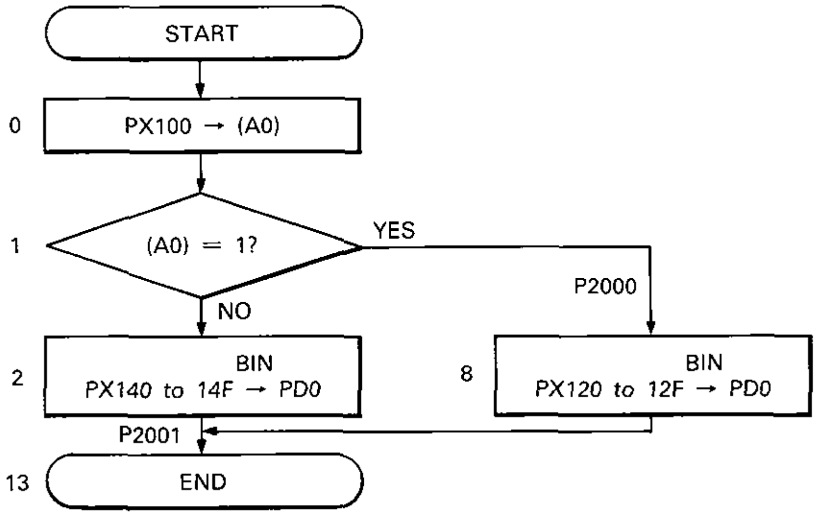
(2) A jump may be made between programs.

RESTRICTIONS

An error is flagged if the pointer specified by JC P ✻✻✻✻ does not exist in the program.

PROGRAM EXAMPLE

The following program transfers the PX120 to 12F data to PD0 if PX100 is on in program 1, and transfers the PX140 to 14F data to PD0 if PX100 is off. (PX120 to 12F and PX140 to 14F values in BCD. Program 20 used.)



```
0 LDAB PX 100··········································Transfers PX100 data to (A0).
1 JC    P 2000·········································Jumps to pointer P2000 if (A0)
                                                       data is 1 and executes the step
                                                       after the next if that data is not 1.
2 BIN  PX 140    PD 0··································Converts PX140 to 14F data into
                                                       BIN and stores the result to PD0.
5 JMP   P 2001 ········································Jumps to pointer P2001.
6 P      2000
8 BIN  PX 120    PD 0··································Converts PX120 to 12F data into
                                                       BIN and stores the result to PD0.
11 P     2001
13 END
```

6.10.3 Subroutine call/return ······ CALL/RET

FORMAT	CALL ⊔ P ✱✱✱✱/RET

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
P	Call destination pointer number														○	2/1				○			

FUNCTIONS

(1) The CALL instruction executes the subroutine program specified by the pointer.

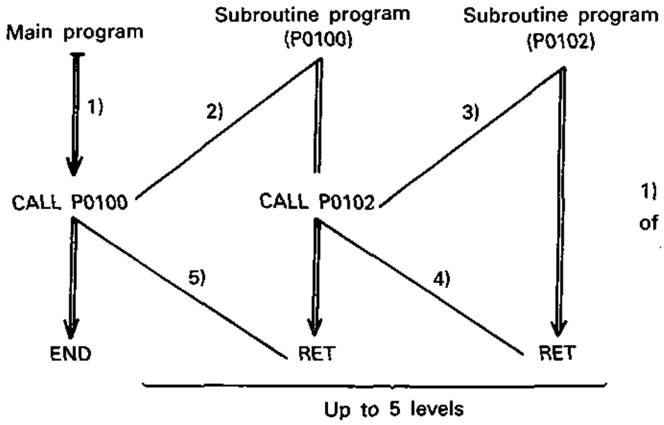
(2) The RET instruction indicates the end of the subroutine program.

(3) The RET instruction returns execution to the instruction at the step following the CALL instruction.

Instruction	Result	Number of Steps
CALL P ✱✱✱✱	Executes the subroutine program specified by the pointer.	2
RET	Terminates the subroutine program execution and executes the instruction at the step following the CALL instruction.	1

(4) The CALL instruction may be executed between programs.

(5) The CALL instructions may be nested up to a level of five. An operation sequence is as shown below when the CALL instructions are nested.



1) to 5) indicate the order of program execution.

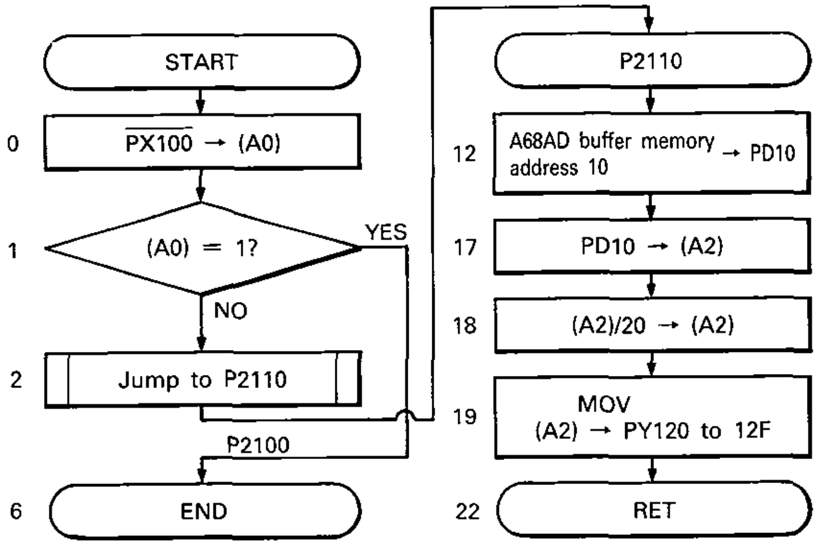
RESTRICTIONS

1) The subroutine program must be ended by the RET instruction. Otherwise an error will occur.

2) An error will be flagged if the CALL instructions are nested to six or more levels.

PROGRAM EXAMPLE

The following example performs no operation if PX100 is off, and reads a value from channel 1 of the A68AD, divides the value by 20 and outputs that value to PY120 to 12F if PX100 is on. (Program 21 used)



```
 0 NOT   PX 100 .................... Flips PX100 and transfers to (A0).
 1 JC     P 2100 ................... Jumps to pointer P2100 if PX100
                                     is off, and executes the next step if
                                     PX100 is on.
 2 CALL   P 2110 ................... Jumps to the subroutine program
                                     at pointer 2100 and executes the
                                     program.
 4 P     2100
 6 END
10 P     2110
12 FROM   H 0014  K 10  PD 10  K 1.. Reads the digital value from the
                                     AD68AD CH1 to PD10.
17 LDAF  PD 10 ..................... Stores PD10 data to (A2).
18 /      K 20 ..................... Divides (A2) data by 20 and stores
                                     the result to (A2).
19 MOV    A 2        PY 120 ........ Transfers (A2) data to PY120 to
                                     12F.
22 RET ............................. Returns to the instruction follow-
                                     ing the [CALL] instruction.
```

6.11 Operation Instructions

The operation instructions perform operations used for process control, etc., e.g. addition, subtraction, multiplication, division and trigonometric function, using accumulator (A2).

Instruction	Description	Refer To
+	Adds the (A2) data and the specified device data and stores the result to (A2). (A2) + Ⓢ → (A2)	Section 6.11.1
−	Subtracts the specified device data from the (A2) data and stores the result to (A2). (A2) − Ⓢ → (A2)	Section 6.11.2
*	Multiplies the (A2) data by the specified device data and stores the result to (A2). (A2) * Ⓢ → (A2)	Section 6.11.3
/	Divides the (A2) data by the specified device data and stores the result to (A2). (A2) / Ⓢ → (A2)	Section 6.11.4
PCT	Divides the (A2) data by the specified device data, multiplies the result by 100, and stores the final result to (A2). {(A2) / Ⓢ} × 100 → (A2)	Section 6.11.5
SQAT	Calculates the square root of the (A2) data and stores the result to (A2). $\sqrt{\ }$ (A2) → (A2)	Section 6.11.6
ABS	Calculates the absolute value of the (A2) data and stores the result to (A2). | (A2) | → (A2)	Section 6.11.7
SIN	Calculates the sine value of the (A2) data and stores the result to (A2). sin(A2) → (A2)	Section 6.11.8
COS	Calculates the cosine value of the (A2) data and stores the result to (A2). cos(A2) → (A2)	Section 6.11.9
TAN	Calculates the tangent value of the (A2) data and stores the result to (A2). tan(A2) → (A2)	Section 6.11.10
ASIN	Calculates the arc sine value of the (A2) data and stores the result to (A2). $\sin^{-1}$(A2) → (A2)	Section 6.11.11
ACOS	Calculates the arc cosine value of the (A2) data and stores the result to (A2). $\cos^{-1}$(A2) → (A2)	Section 6.11.12
ATAN	Calculates the arc tangent value of the (A2) data and stores the result to (A2). $\tan^{-1}$(A2) → (A2)	Section 6.11.13
EXP	Calculates the exponential function of the (A2) data and stores the result to (A2). $e^{(A2)}$ → (A2)	Section 6.11.14
LOG	Calculates the common logarithm of the (A2) data and stores the result to (A2). $\log_{10}$(A2) → (A2)	Section 6.11.15
LN	Calculates the natural logarithm of the (A2) data and stores the result to (A2). log e(A2) → (A2)	Section 6.11.16

6.11.1 Addition …… +

FORMAT	+ ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant added to (A2)					○		○					○			1						○	

FUNCTIONS

(1) Adds the (A2) data and the specified device data, Ⓢ, and stores the result to (A2).

(A2) + Ⓢ → (A2)

Before execution (A2) − 1 2 3 4 5
Ⓢ 8 7 6 5 0

⇩ + instruction

After execution (A2) 7 5 3 0 5

(2) The specified device data, Ⓢ, remains unchanged after the + instruction is executed.

REMARKS

The (A2) data is overwritten by the execution result of the + instruction. The (A2) data required should be saved before execution of the + instruction.

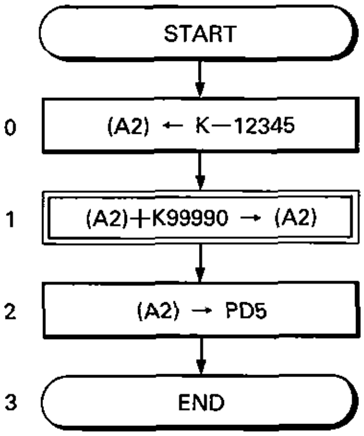
RESTRICTIONS

1) Data used with the + instruction is between ± 2.7 × 10^{-20} and ± 9.2 × 10^{18}.
The constant (K) specified during programming is between K ± 1 × 10^{-9} and K9.999 × 10^{9}.

2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

Example: If +K999999 is entered in the program, it changes to +K999900.

PROGRAM EXAMPLE The following program stores −12345 to (A2), adds the (A2) data and 99990, and stores the result to PD5.



```
0 LDAF  K-12345 ··································Stores −12345 to (A2).
1 +     K 99990 ··································Adds (A2) data and 99990 and
                                                  stores the result to (A2).
2 STAF PD 5 ······································Stores (A2) data to PD5.
3 END ············································Terminates program execution.
```

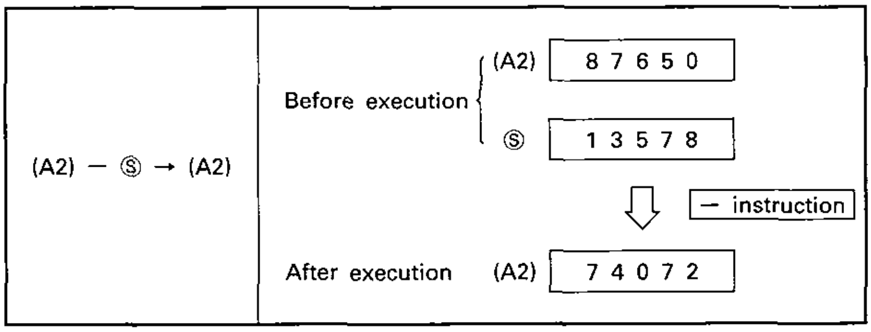
6.11.2 Subtraction …… −

FORMAT	− ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant subtracted from (A2)					○		○					○			1						○	

FUNCTIONS

(1) Subtracts the specified device data, Ⓢ, from the (A2) data and stores the result to (A2).



(2) The specified word device data, Ⓢ, remains unchanged after the $\boxed{-}$ instruction is executed.

REMARKS

The (A2) data is overwritten by the execution result of the $\boxed{-}$ instruction. The (A2) data required should be saved before execution of the $\boxed{-}$ instruction.

RESTRICTIONS

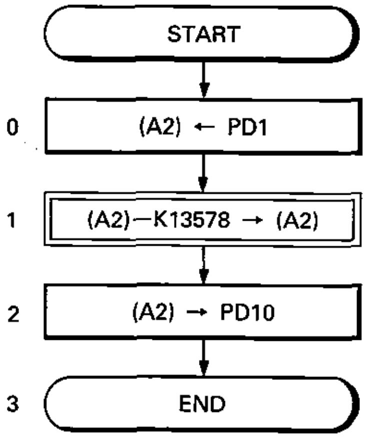
1) Data used with the $\boxed{-}$ instruction is between $\pm 2.7 \times 10^{-20}$ and $\pm 9.2 \times 10^{18}$.
The constant (K) specified during programming is between $K \pm 1 \times 10^{-9}$ and $K9.999 \times 10^{9}$.

2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

Example: If −K87654 is entered in the program, it changes to −K87650.

PROGRAM EXAMPLE

The following program stores the PD1 data (87650) to (A2), subtracts 13578 from the (A2) data, and stores the result to PD10.



```
0 LDAF  PD 1 ·······························Stores PD1 data (87650) to (A2).
1 -      K 13578 ·······················Subtracts 13578 from (A2) data
                                          and stores the result to (A2).
2 STAF  PD 10 ······························Stores (A2) data to PD10.
3 END
```

6.11.3 Multiplication …… *

FORMAT	* ⌴ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant for multiplying (A2)					○		○					○			1						○	

FUNCTIONS

(1) Multiplies the (A2) data by the specified device data, Ⓢ, and stores the result to (A2).

(A2) * Ⓢ → (A2)	Before execution (A2) 1 2 3 4 5 Ⓢ 3 4 5 6 ⇩ $\boxed{*}$ instruction After execution (A2) 4 2 6 6 4 3 2 0

(2) The specified device data, Ⓢ, remains unchanged after the $\boxed{*}$ instruction is executed.

REMARKS

The (A2) data is overwritten by the execution result of the $\boxed{*}$ instruction. The (A2) data required should be saved before execution of the $\boxed{*}$ instruction.

RESTRICTIONS

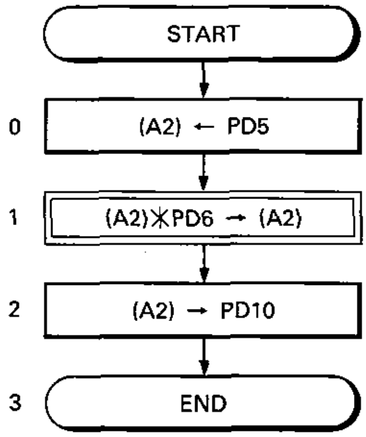
1) Data used with the $\boxed{*}$ instruction is between $\pm 2.7 \times 10^{-20}$ and $\pm 9.2 \times 10^{18}$.
The constant (K) specified during programming is between K $\pm 1 \times 10^{-9}$ and K9.999×10^{9}.

2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

Example: If *K123456 is entered in the program, it changes to *K123400.

PROGRAM EXAMPLE

The following program stores the PD5 data (12345) to (A2), multiplies the (A2) data by the PD6 data (3456), and stores the result to PD10.



```
0 LDAF PD 5·······································Stores PD5 data (12345) to (A2).
1 ●    PD 6·······································Multiplies (A2) and PD6 data
                                                  together and stores the result to
                                                  (A2).
2 STAF PD 10······································Stores (A2) data to PD10.
3 END
```

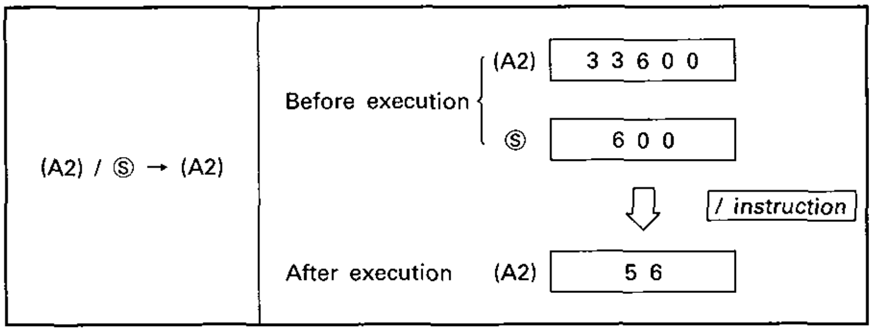
6.11.4 Division ······ /

FORMAT	/ ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant added to (A2)					○		○					○			1						○	

FUNCTIONS

(1) Divides the (A2) data by the specified device data, Ⓢ, and stores the result to (A2).



(2) The specified device data, Ⓢ, remains unchanged after the / instruction is executed.

REMARKS

The (A2) data is overwritten by the execution result of the / instruction. The (A2) data required should be saved before execution of the / instruction.

RESTRICTIONS

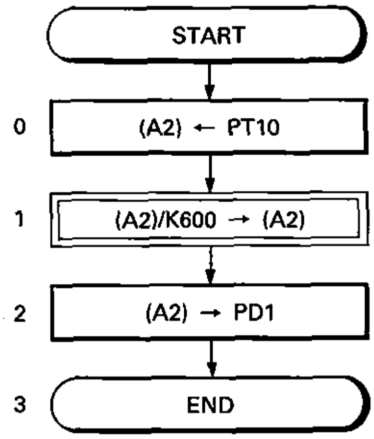
1) Data used with the / instruction is between $\pm 2.7 \times 10^{-20}$ and $\pm 9.2 \times 10^{18}$.
The constant (K) specified during programming is between K $\pm 1 \times 10^{-9}$ and K9.999×10^{9}.

2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

Example: If /K53456 is entered in the program, it changes to /K53450.

PROGRAM EXAMPLE

The following program stores the PT10 present value (33600) to (A2), divides the (A2) data by 600, and stores the result to PD1.



```
0 LDAF PT 10 ·····························Stores PT10 present value to (A2).
1 /       K 600···························Divides (A2) data by 56 and stores
                                          the result to (A2).
2 STAF PD 1·······························Stores (A2) data to PD1.
3 END
```

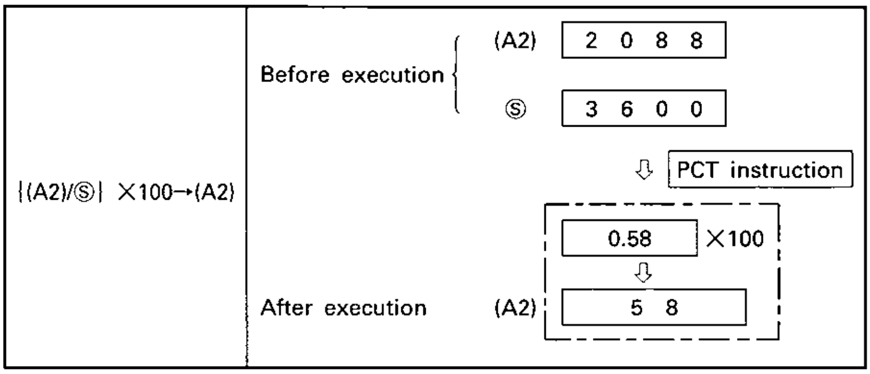
6.11.5 % operation ······ PCT

FORMAT	PCT ␣ Ⓢ

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant for performing % operation on (A2) data					○		○					○			1						○	

FUNCTIONS

(1) Performs % operation on the (A2) data by the specified device data, Ⓢ, (divides the (A2) data by data, Ⓢ, then multiplies the result by 100) and stores the result to (A2).



(2) The specified device data, Ⓢ, remains unchanged after the PCT instruction is executed.

REMARKS

The (A2) data is overwritten by the execution result of the PCT instruction. The (A2) data required should be saved before execution of the PCT instruction.

RESTRICTIONS

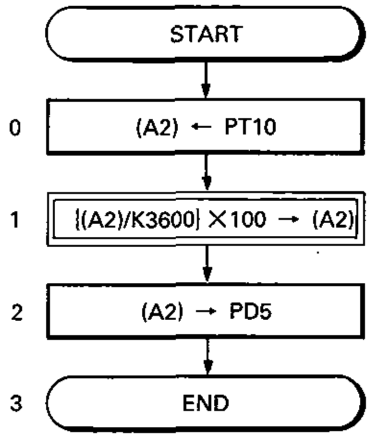
1) Data used with the PCT instruction is between ± 2.7 × 10^{-20} and ± 9.2 × 10^{18}.
The constant (K) specified during programming is between K ± 1 × 10^{-9} and K9.999 × 10^{9}.

2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

Example: If PCT K123456 is entered in the program, it changes to PCT K123400.

PROGRAM EXAMPLE

The following program stores the PT10 present value (2088) to (A2), performs % operation on the (A2) data by 3600, and stores the result to PD5.



```
0 LDAF PT 10 ······························Stores PT10 present value to (A2).
1 PCT    K 3600 ···························Performs % operation on (A2)
                                           data by 3600 and stores the result
                                           to (A2).
2 STAF PD 5 ·······························Stores (A2) data to PD5.
3 END
```

MEMO

6.11.6 Square root ...... SQRT

FORMAT	SQRT

<table>
<tr><th rowspan="2"></th><th rowspan="2">Set Data</th><th colspan="14">Set Device</th><th rowspan="2">Number of Steps</th><th colspan="7">Error Occurrence</th></tr>
<tr><th>PX</th><th>PY</th><th>PM</th><th>SP.PM</th><th>PT</th><th></th><th>PD</th><th>SP.PD</th><th>A0</th><th>A1</th><th>A2</th><th>K</th><th>H</th><th>P</th><th>51</th><th>54</th><th>55</th><th>56</th><th>57</th><th>58</th><th>59</th></tr>
<tr><td></td><td>Device for obtaining square root</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>○</td><td></td><td></td><td></td><td>1</td><td></td><td></td><td></td><td></td><td></td><td>○</td><td></td></tr>
</table>

FUNCTIONS

(1) Performs square root operation on the A2 data and stores the result to A2.

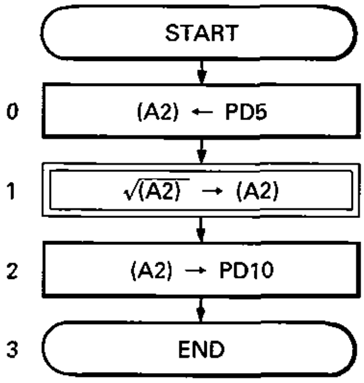
$\sqrt{(A2)} \rightarrow (A2)$	Before execution	(A2)	1 2 6 7 3 6
		⇩	SQRT instruction
	After execution	(A2)	3 5 6

REMARKS

The (A2) data may be overwritten by the execution result of the SQRT instruction. The (A2) data required should be saved before execution of the SQRT instruction.

PROGRAM EXAMPLE

The following program stores the PD5 data (126736) to (A2), performs square root operation on the (A2) data, and stores the result to PD10.



```
0 LDAF PD 5 ............................Transfers PD5 data (126736) to (A2).
1 SQRT .................................Performs square root operation on (A2) data
                                        and stores the result to (A2).
2 STAF PD 10............................Transfers (A2) data to PD10.
3 END
```

RESTRICTIONS

1) Data used with the SQRT instruction is between + 2.7 × 10^{-20} and + 9.2 × 10^{18}.
Negative data cannot be used.

6.11.7 Absolute value ······ ABS

FORMAT	ABS

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
	Device for obtaining absolute value											○				1							

FUNCTIONS

(1) Removes signs from the (A2) data and stores the absolute value to (A2).

|(A2)| → (A2)	Before execution (A2) −1 2 3 5 ⇩ ABS instruction After execution (A2) 1 2 3 5

REMARKS

The (A2) data is overwritten by the execution result of the ABS instruction. The (A2) data required should be saved before execution of the ABS instruction.

PROGRAM EXAMPLE

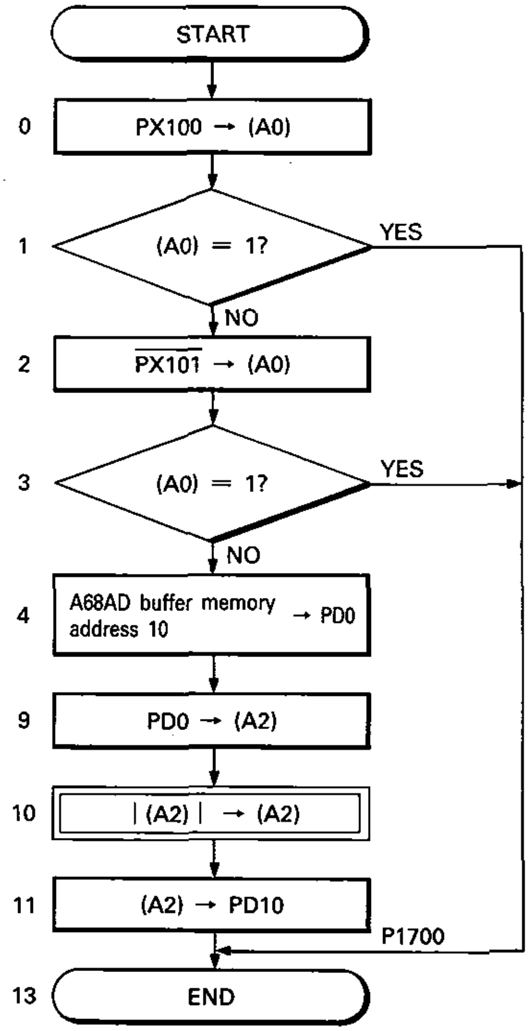
The following program reads the CH1 digital value from the A68AD, converts it into an absolute value, and stores the result to PD10. (Program 17 used)

Main base unit configuration

Power supply module	A81CPU	A68AD	A62DA	AX10	AY10	Vacancy	Vacancy	Vacancy	Vacancy
		PX100 to PX11F PY100 to PY11F	PX120 to PX13F PY120 to PY13F	PX140 to PX14F	PY150 to PY15F				

RESTRICTIONS

1) Data used with the ABS instruction is between ± 2.7 × 10^{-20} and ± 9.2 × 10^{18}.



```
 0 LDAB PX 100                              } Checks the A68AD operating
 1 JC    P 1700                             } status.
 2 NOT  PX 101                              } (Watch dog timer error, A/D con-
 3 JC    P 1700                             } version ready signal)
 4 FROM  H 0010  K 10  PD 0  K 1........Reads digital value from CH1 of
                                          the A68AD.
 9 LDAF PD 0..............................Transfers PD0 data to (A2).
10 ABS....................................Stores the (A2) absolute value to
                                          (A2).
11 STAF PD 10.............................Stores (A2) data to PD10.
12 P    1700
14 END
```

6.11.8 Sine ······ SIN

FORMAT	SIN

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
	Device for obtaining sine											○				1							

FUNCTIONS

(1) Performs sine operation on the (A2) data in radian ((π/180) × *angle) and stores the result to (A2).*

sin (A2) → (A2)	Before execution	(A2)	1. 5 7 0 7 9 6
		⇩	SIN instruction
	After execution	(A2)	0. 5

(2) Data used with the SIN instruction is between 0 and ± 2π. Any value outside this range must be divided by 2π and its remainder used for sine operation.

REMARKS

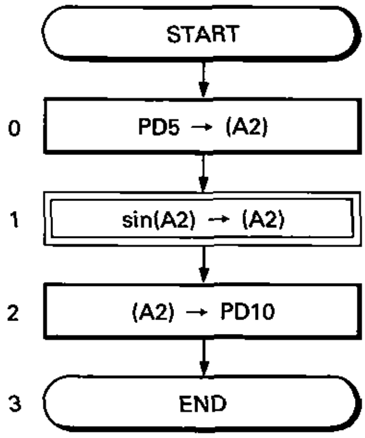
The (A2) data is overwritten by the execution result of the SIN instruction. The (A2) data required should be saved before execution of the SIN instruction.

RESTRICTIONS

1) Data used with the SIN instruction is between 0 and ± 2π.

PROGRAM EXAMPLE

The following program performs sine operation on the PD5 data (in radian) and stores the result to PD10.



```
0 LDAF PD 5 ·····························Transfers PD5 data to (A2).
1 SIN ···································Executes sine operation and
                                         stores the result to (A2).
2 STAF PD 10 ····························Transfers (A2) data to PD10.
3 END
```

6.11.9 Cosine ······ COS

FORMAT	**COS**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device for obtaining cosine											○				1							

FUNCTIONS

(1) Performs cosine operation on the (A2) data in radian ((π/180) × angle) and stores the result to (A2).

cos (A2) → (A2)	Before execution	(A2)	−0. 7 2 2 7 3 4
		⇩	COS instruction
	After execution	(A2)	0. 7 5

(2) Data used with the COS instruction is between 0 and ± 2π. Any value outside this range must be divided by 2π and its remainder used for cosine operation.

REMARKS

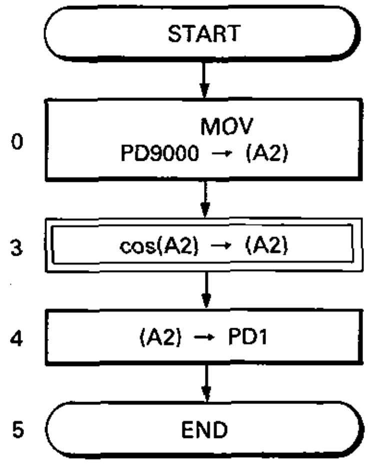
The (A2) data is overwritten by the execution result of the COS instruction. The (A2) data required should be saved before execution of the COS instruction.

RESTRICTIONS

1) Data used with the COS instruction is between 0 and ± 2π.

PROGRAM EXAMPLE

The following program performs cosine operation on the PD9000 data (in radian) and stores the result to PD1.



```
0 MOV  PD 9000         A 2················Transfers PD9000 data to (A2).
3 COS·····················································Executes cosine operation and
                                                          stores the result to (A2).
4 STAF PD 1·······································Transfers (A2) data to PD1.
5 END
```

MEMO

6.11.10 Tangent …… TAN

FORMAT	TAN

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
	Device for obtaining tangent											○				1						○	

FUNCTIONS

(1) Performs tangent operation on the (A2) data in radian ((π/180) × angle) and stores the result to (A2).

tan (A2) → (A2)	Before execution (A2) 2. 6 1 1 5 3 1 ⇩ TAN instruction After execution (A2) −0. 5 8 6

(2) Data used with the TAN instruction is between 0 and ± 2π. Any value outside this range must be divided by 2π and its remainder used for tangent operation.

REMARKS

The (A2) data is overwritten by the execution result of the TAN instruction. The (A2) data required should be saved before execution of the TAN instruction.

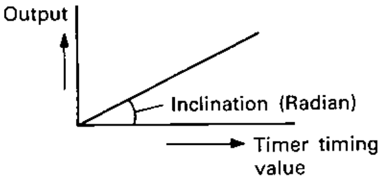
RESTRICTIONS

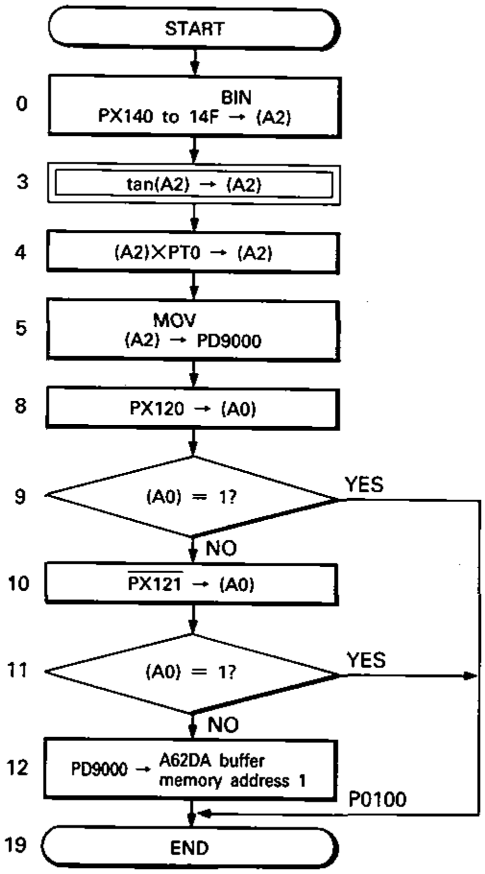
1) Data used with the TAN instruction is between 0 and ± 2π.

PROGRAM EXAMPLE

The following program provides output to the A62DA at intervals of elapsed time with the inclination constant. The inclination is defined by PX140 to 14F, timing is executed by timer PT0, the timer value and inclination are operated, and the result is output as a digital value.

Power supply module	A81CPU	A68AD	A62DA	AX10	AY10	Vacancy	Vacancy	Vacancy	Vacancy
		PX100 to PX11F PY100 to PY11F	PX120 to PX13F PY120 to PY13F	PX140 to PX14F	PY150 to PY15F				





```
 0 BIN   PX 140 ......................Transfers PX140 to 14F data to
                                      (A2).
 3 TAN ...............................Performs tangent operation using
                                      (A2) data as inclination and stores
                                      the result to (A2).
 4 *     PT 0 ........................Multiplies (A2) data and PT0 pre-
                                      sent value together and stores the
                                      output to (A2).
 5 MOV   A 2   PD 9000 ...............Converts (A2) data into 16-bit bin-
                                      ary and stores the result to
                                      PD9000.
 8 LDAB PX 120                       ⎫ Checks A62DA operating status.
 9 JC     P 0100                     ⎬ (Watch dog timer error, D/A con-
10 NOT  PX 121                       ⎪ version ready signal)
11 JC     P 0100                     ⎭
12 TO   H 0012  K 1  PD 9000  K 1 ....Writes digital value to CH1 of the
                                        A62DA.
17 P     0100
19 END
```

6.11.11 Arc sine ······ ASIN

FORMAT	ASIN

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
	Device for obtaining arc sine											○				1						○	

FUNCTIONS

(1) Performs arc sine operation on the (A2) data and stores the result to (A2) in radian.

$\sin^{-1}$ (A2) → (A2)	Before execution (A2) 0. 5 6 7 9 4 ⇩ ASIN instruction After execution (A2) 0. 6 0 4 (Unit: radian)

REMARKS

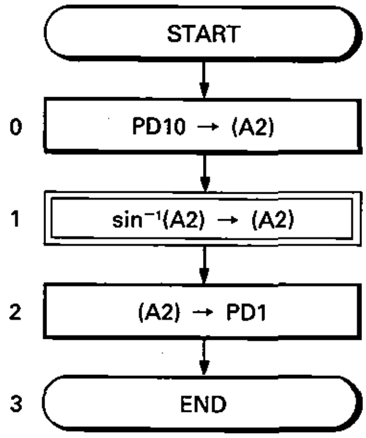
The (A2) data is overwritten by the execution result of the ASIN instruction. The (A2) data required should be saved before execution of the ASIN instruction.

RESTRICTIONS

1) Data used with the ASIN instruction is between 0 × 10^{-20} *and* ± 1.

PROGRAM EXAMPLE

The following program performs arc sine operation on the PD10 data and stores the result to PD1 (in radian).



```
0 LDAF PD 10·······································Transfers PD10 data to (A2).
1 ASIN·············································Performs arc sine operation and
                                                   stores the result to (A2).
2 STAF PD 1········································Transfers (A2) data to PD1.
3 END
```

6.11.12 Arc cosine ······ ACOS

FORMAT	ACOS

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
	Device for obtaining arc cosine											○				1						○	

FUNCTIONS

(1) Performs arc cosine operation on the (A2) data and stores the result to (A2) in radian.

$\cos^{-1}$ (A2) → (A2)	Before execution (A2) 1. 9 2 3 0 3 5 ⇩ ACOS instruction After execution (A2) −0. 3 4 5 (Unit: radian)

REMARKS

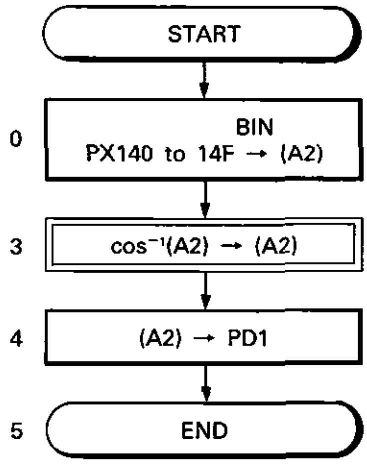
The (A2) data is overwritten by the execution result of the ACOS instruction. The (A2) data required should be saved before execution of the ACOS instruction.

RESTRICTIONS

1) Data used with the ACOS instruction is between 0×10^{-20} and ± 1.

PROGRAM EXAMPLE

The following program performs arc cosine operation on the value incoming from inputs PX140 to 14F and stores the result to PD1.



0 BIN PX 140 A 2 ······ Converts incoming BCD data from PX140 to 14F into BIN and transfers the result to (A2).

3 ACOS ······ Performs arc cosine operation and stores the result to (A2).

4 STAF PD 1 ······ Transfers (A2) data to PD1.

5 END

6.11.13 Arc tangent ······ ATAN

FORMAT	ATAN

Set Data	Set Device														Number of Steps	Error Occurrence						
	PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Device for obtaining arc tangent											○				1						○	

FUNCTIONS

(1) Performs arc tangent operation on the (A2) data and stores the result to (A2) in radian.

$\tan^{-1}$ (A2) → (A2)	Before execution (A2)	40. 3203
	⇩ ATAN instruction	
	After execution (A2)	1. 546

REMARKS

The (A2) data is overwritten by the execution result of the ATAN instruction. The (A2) data required should be saved before execution of the ATAN instruction.

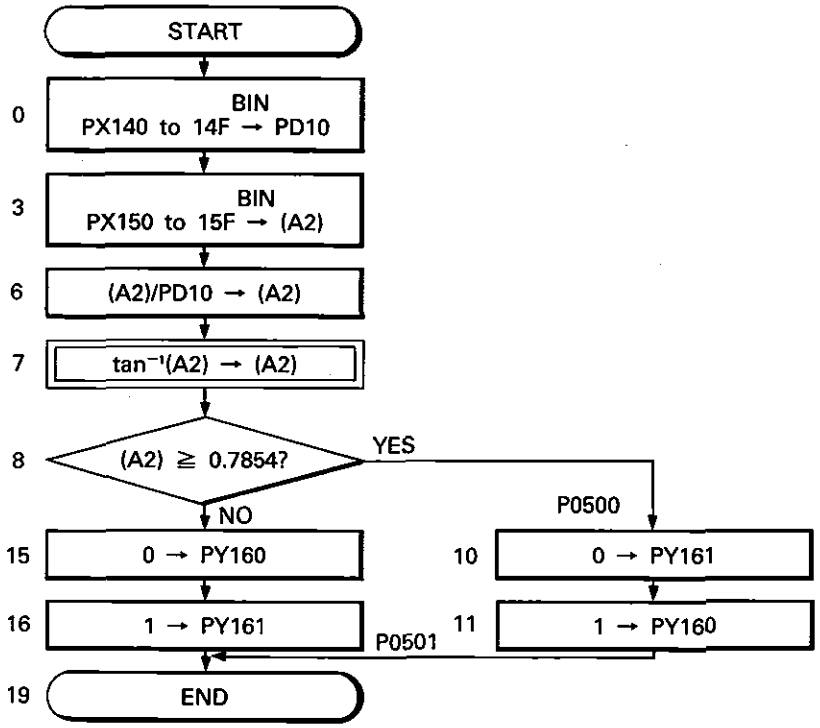
PROGRAM EXAMPLE

The following program converts incoming BCD values from PX140 to 14F and PX150 to 15F into BIN, divides the PX150 to 15F value by the PX140 to 14F and switches on PY160 if the inclination is π/4 radian or greater, and switches on PY161 if the inclination is less than π/4 radian. (Program 5 used)

Power supply module	A81CPU	A68AD	A62DA	AX10	AY10	Vacancy	Vacancy	Vacancy	Vacancy
		PX100 to PX11F PY100 to PY11F	PX120 to PX13F PY120 to PY13F	PX140 to PX14F	PY150 to PY15F				

RESTRICTIONS

1) Data used with the ATAN instruction is between 0 × 10^{-20} and ± 9.2 × 10^{18}.



```
 0 BIN    PX 140         PD 10    } Converts PX140 to 14F and PX150
 3 BIN    PX 150          A 2     } to 15F data into BIN and stores the
                                    results to PD10 and (A2), respectively.
 6 /      PD 10  ......................Divides (A2) data by PD10 data
                                    and stores the result to (A2).
 7 ATAN          ......................Calculates inclination by arc tangent operation.
 8 LTAF    K 0.7854 ...................Executes the step after the next if
                                    inclination is /4 radian or more,
                                    and executes the next step if inclination is less than /4 radian.
 9 JMP     P 0500 .....................Jumps to pointer P0500.
10 RST    PY 161                  } Switches PY161 off and PY160 on
11 SET    PY 160                  } if inclination is /4 radian or more.
12 JMP     P 0501......................Jumps to pointer P0501.
13 P      0500
15 RST    PY 160                  } Switches PY160 off and PY161 on
16 SET    PY 161                  } if inclination is less than /4 radian.
17 P      0501
19 END
```

6.11.14 Exponential function ······ EXP

FORMAT	EXP

	Set Device														Number of Steps	Error Occurrence						
Set Data	PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Device for obtaining exponential function											○				1						○	

FUNCTIONS

(1) Performs exponential function operation on the (A2) data which is used as an exponent to e, and stores the result (e^{A2}) to (A2).

$e^{(A2)} \rightarrow$ (A2)	Before execution (A2)	2. 5 4
	⇩	EXP instruction
	After execution (A2)	1 2. 3 7 9 6 7 1

REMARKS

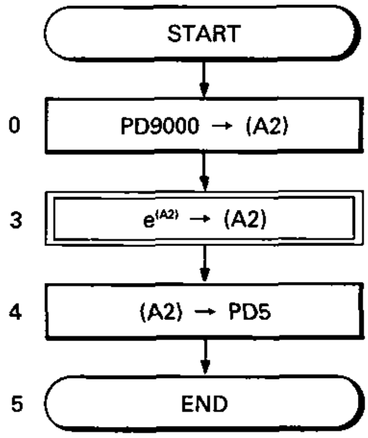
The (A2) data is overwritten by the execution result of the EXP instruction. The (A2) data required should be saved before execution of the EXP instruction.

RESTRICTIONS

1) Data specified in A2 is between —45.05845 and 43.6657. An operation error will be flagged if any value specified in A2 is outside that range.

PROGRAM EXAMPLE

The following program performs exponential function operation on the PD9000 data and stores the result to PD5.



```
0 MOV  PD 9000 ·····························Transfers PD9000 data to (A2).
3 EXP ······································Performs exponential function op-
                                            eration and stores the result to
                                            (A2).
4 STAF PD 5 ································Transfers (A2) data to PD5.
5 END
```

6.11.15 Common logarithm …… LOG

FORMAT	LOG

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
	Device for obtaining common logarithm											○				1						○	

FUNCTIONS

(1) Performs common logarithm operation on the (A2) data and stores the result to (A2).

LOG (A2) → (A2)	Before execution (A2)	363.1
	⇩	LOG instruction
	After execution (A2)	2.56

REMARKS

The (A2) data is overwritten by the execution result of the LOG instruction. The (A2) data required should be saved before execution of the LOG instruction.

PROGRAM EXAMPLE

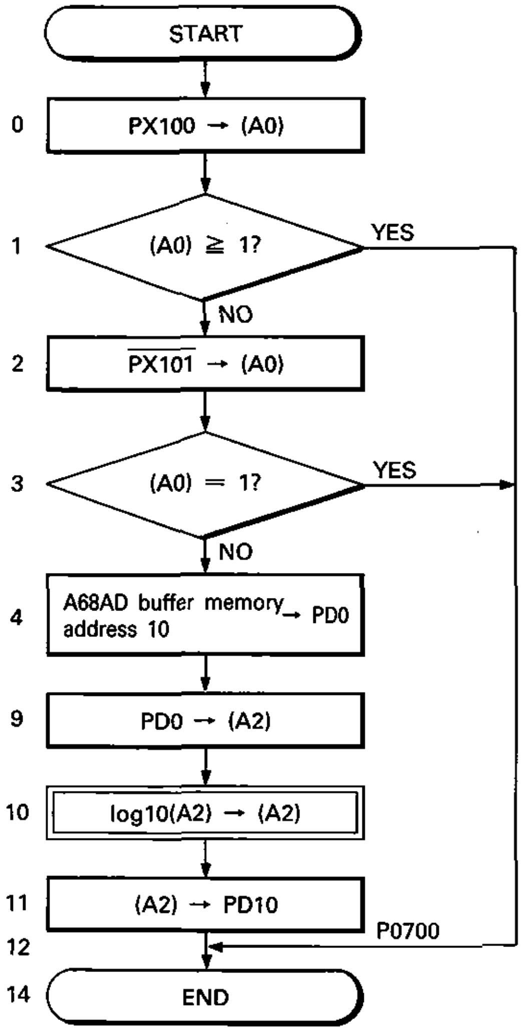
The following program performs common logarithm operation on the digital value read from CH1 of the A68AD and stores the result to PD10. (Program 7 used)

Main base unit configuration

Power supply module	A81CPU	A68AD	A62DA	AX10	AY10	Vacancy	Vacancy	Vacancy	Vacancy
		PX100 to PX11F PY100 to PY11F	PX120 to PX13F PY120 to PY13F	PX140 to PX14F	PY150 to PY15F				

RESTRICTIONS

1) Data used with the LOG instruction is between 2.7 × 10^{-20} *and 9.2 × 10^{18}.*



```
 0 LDAB PX 100                         Checks A68AD operating status.
 1 JC     P 0700                       (Watch dog timer error, D/A con-
 2 NOT  PX 101                         version ready signal)
 3 JC     P 0700
 4 FROM   H 0010   K 10   PD 0   K 1 ···Reads digital value from CH1 of
                                        the A68AD and stores the value to
                                        PD0.
 9 LDAF PD 0 ···························Transfers PD0 data to (A2).
10 LOG ·································Performs common logarithm op-
                                        eration on (A2) data and stores
                                        the result to (A2).
11 STAF PD 10···························Transfers (A2) data to PD10.
12 P    0700
14 END
```

6.11.16 Natural logarithm ······ LN

FORMAT	LN

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
	Device for obtaining natural logarithm											○				1							

FUNCTIONS

(1) *Performs natural logarithm operation on the (A2) data and stores the result to (A2).*

LN (A2) → (A2)	Before execution (A2) 1 1 0 0 ⇩ LN instruction After execution (A2) 7. 0 0 3

REMARKS

The (A2) data is overwritten by the execution result of the LN instruction. The (A2) data required should be saved before execution of the LN instruction.

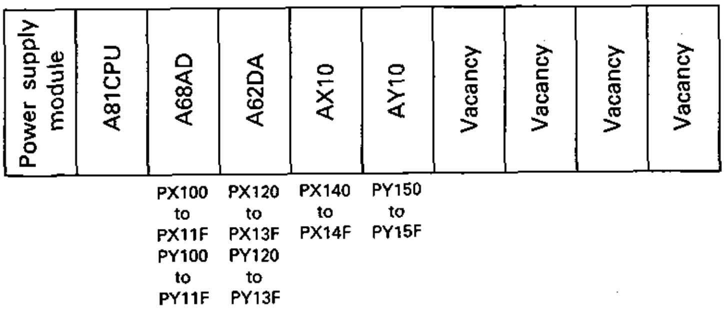
RESTRICTIONS

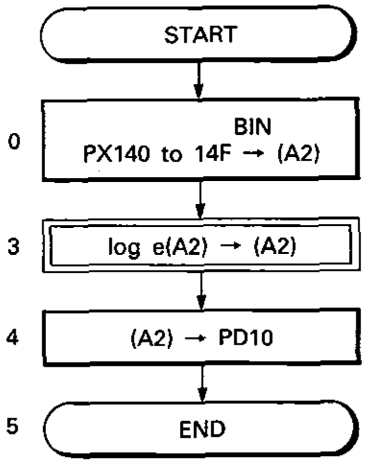
1) Data used with the LN instruction is between 2.7 × 10^{-20} and 9.2 × 10^{18}.

PROGRAM EXAMPLE

The following program converts incoming BCD data from PX140 to 14F into BIN, performs natural logarithm operation, and stores the result to PD10.

Main base unit configuration





```
0 BIN   PX 140   A 2 ..............Converts PX140 to 14F BCD data
                                   into BIN and stores the result to
                                   (A2).
3 LN ..............................Performs natural logarithm opera-
                                   tion on (A2) data and stores the
                                   result to (A2).
4 STAF PD 10.......................Transfers (A2) data to PD10.
5 END
```

MEMO

6.12 Special Instructions

Special instructions include magnitude comparison instructions, high and low limit instructions, alarm output instructions, etc.

Instruction	Description	Refer To
HS (High select)	Compares the (A2) and specified device data and stores higher data to (A2). If (A2) ≧ Ⓢ, (A2) → (A2). If (A2) < Ⓢ, Ⓢ → (A2).	Section 6.12.1
LS (Low select)	Compares the (A2) and specified device data and stores lower data to (A2). If (A2) ≦ Ⓢ, (A2) → (A2). If (A2) > Ⓢ, Ⓢ → (A2).	Section 6.12.2
HLM (High limit)	Stores the specified device data to (A2) if the (A2) data is greater than the specified device data. If (A2) > Ⓢ, Ⓢ → (A2).	Section 6.12.3
LLM (Low limit)	Stores the specified device data to (A2) if the (A2) data is less than the specified device data. If (A2) < Ⓢ, Ⓢ → (A2).	Section 6.12.4
NOP	Does nothing at the current step and progresses to the next step. (No operation)	Section 6.12.5
END	Written at the end of any program to declare the program end.	
HAL (High alarm)	Switches on alarm if the (A2) data becomes equal to or greater than the set value and switches off alarm if that data becomes less than (set value - hysteresis value).	Section 6.12.6
LAL (Low alarm)	Switches on alarm if the (A2) data becomes equal to or less than the set value and switches off alarm if that data becomes greater than (set value + hysteresis value).	Section 6.12.7
SAL (Set alarm)	Switches on alarm if the (A2) data is within the (set value + ON area) range and switches off alarm if that data is outside the above range.	Section 6.12.8
DISP	Displays the PID control status monitored on the CRT connected to the AD57.	Section 6.12.9
LOOP	Executes the specified macro function.	Section 6.12.10

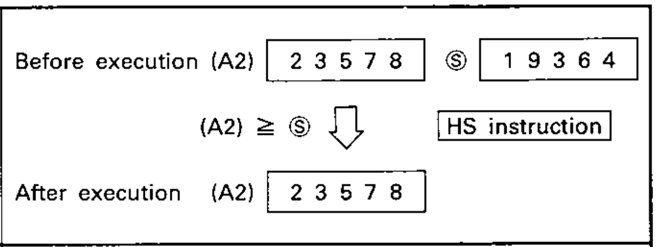
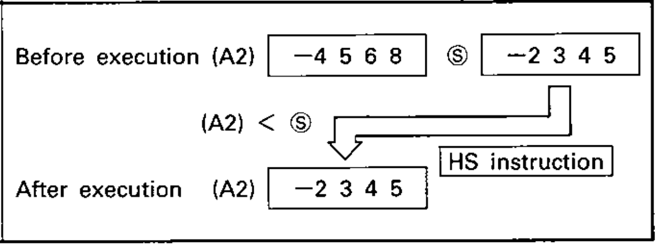
6.12.1 High select ······ HS

FORMAT	**HS ␣ Ⓢ**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant for high select					○		○					○			1						○	

FUNCTIONS

(1) Compares the (A2) data and the specified device data, Ⓢ, and stores higher data to (A2).

If (A2) ≧ Ⓢ, (A2) → (A2)	
If (A2) < Ⓢ, Ⓢ → (A2)	

(2) The word device data, Ⓢ, remains unchanged after the HS instruction is executed.

REMARKS

The (A2) data may be overwritten by the execution result of the HS instruction. The (A2) data required should be saved before execution of the HS instruction.

RESTRICTIONS

1) Data used with the HS instruction is between ± 2.7 × 10^{-20} and ± 9.2 × 10^{18}.
The constant (K) specified during programming is between K ± 1 × 10^{-9} and K9.999 × 10^{9}.

2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

Example: If HS K123456 is entered in the program, it changes to HS K123400.

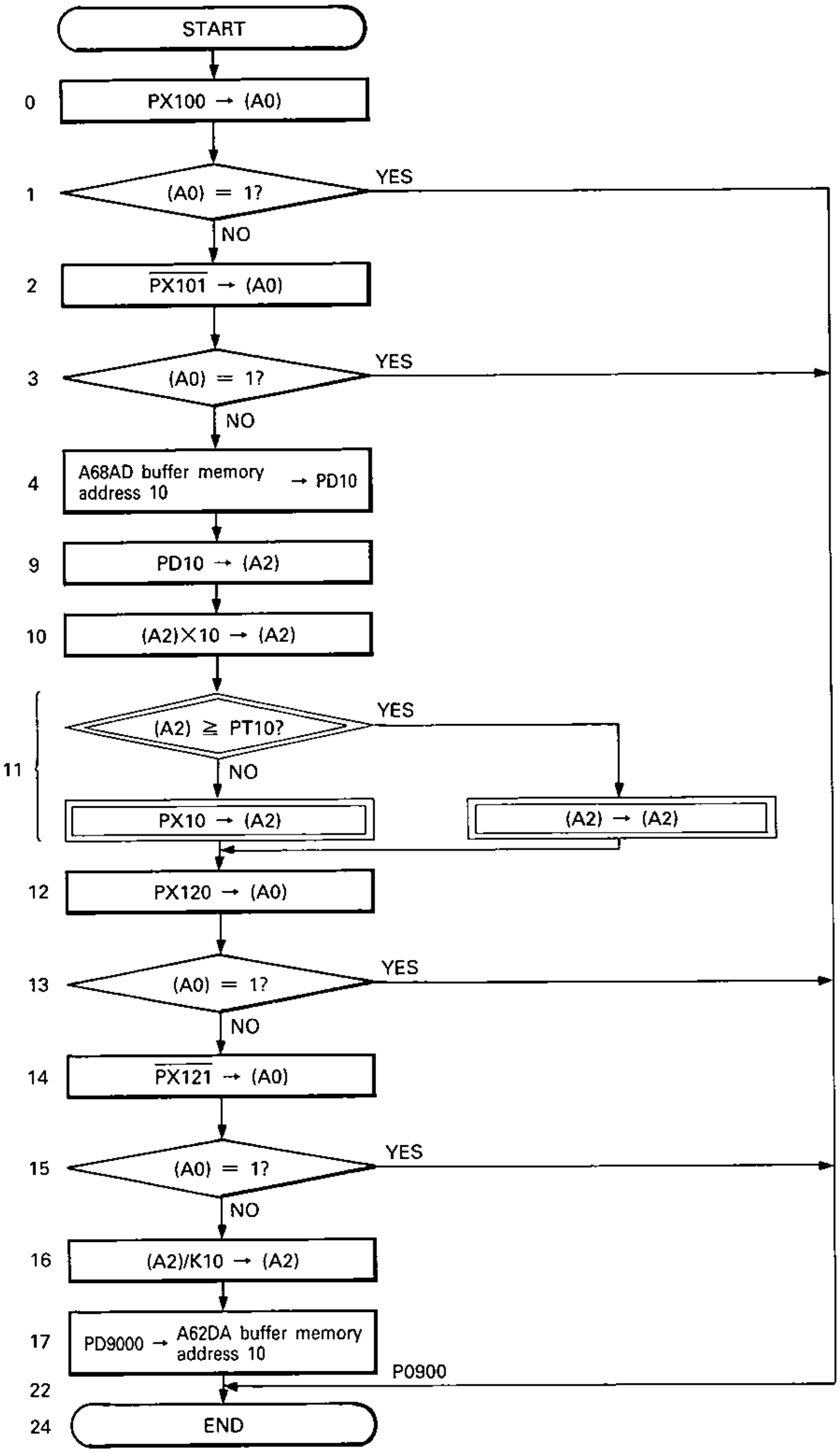
PROGRAM EXAMPLE

The following program compares the value read from the A68AD CH1 with the present value of timer PT0, outputs the CH1 value to the A62DA CH1 if CH1 PT0, and outputs the PT0 value and switches on PY100 if CH1 PT0. (Program 9 used)
(The A68AD digital value is incremented in proportion to the PT0 present value and the higher value is output to the A62DA if there is an incremental difference.)

Main base unit configuration

Power supply module	A81CPU	A68AD	A62DA	AX10	AY10	Vacancy	Vacancy	Vacancy	Vacancy
		PX100 to PX11F PY100 to PY11F	PX120 to PX13F PY120 to PY13F	PX140 to PX14F	PY150 to PY15F				

(It is assumed that the digital value is incremented by 1 as the timer present value is incremented by 1 seconds.)


START
0
PX100 → (A0)
1
(A0) = 1?
YES
NO
2
$\overline{\text{PX101}}$ → (A0)
3
(A0) = 1?
YES
NO
4
A68AD buffer memory address 10 → PD10
9
PD10 → (A2)
10
(A2)×10 → (A2)
11
(A2) ≧ PT10?
YES
NO
PX10 → (A2)
(A2) → (A2)
12
PX120 → (A0)
13
(A0) = 1?
YES
NO
14
$\overline{\text{PX121}}$ → (A0)
15
(A0) = 1?
YES
NO
16
(A2)/K10 → (A2)
17
PD9000 → A62DA buffer memory address 10
P0900
22
24
END

```
 0 LDAB PX 100                         ⎫ Checks the A68AD operating
 1 JC     P 0900                       ⎪ status.
 2 NOT  PX 101                         ⎪ (Watch dog timer error, A/D con-
 3 JC     P 0900                       ⎭ version ready signal)
 4 FROM   H 0010  K 10  PD 10  K 1·····Reads the CH1 digital value from
                                        the A68AD to PD10.
 9 LDAF PD 10··························Transfers PX10 data to (A2).
10 *      PT 10························Multiplies the read digital value
                                        by 10.
11 HS     PT 10························Stores (A2) data to (A2) if (A2) ≧
                                        PT0, and PT0 data to (A2) if (A2) <
                                        PT0.
12 LDAB PX 120                         ⎫ Checks the A62DA operating
13 JC     P 0900                       ⎪ status.
14 NOT  PX 121                         ⎪ (Watch dog timer error, A/D con-
15 JC     P 0900                       ⎭ version ready signal)
16 /      K 10·························Divides (A2) value by 10.
17 TO     H 0012 K 0  A 2   K 1········Writes the digital value to the
22 P    0900                            A62DA CH1.
24 END
```

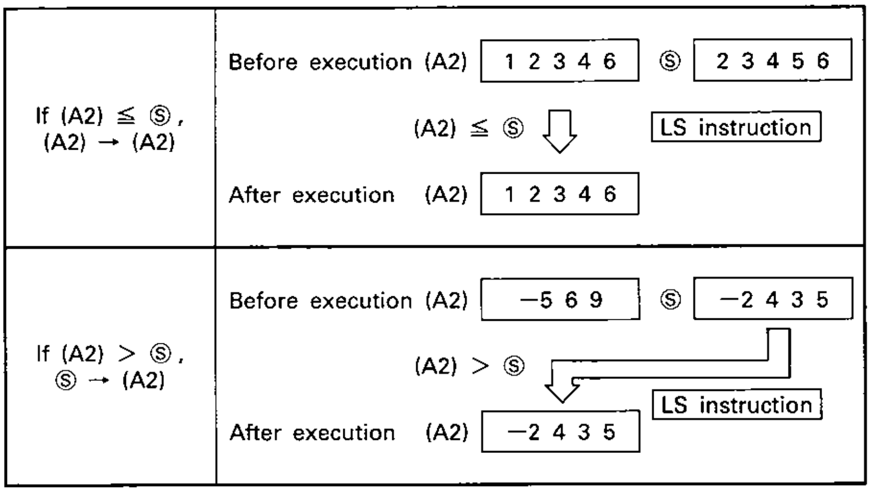
6.12.2 Low select ······ LS

FORMAT	**LS ␣ Ⓢ**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant for low select					○		○					○			1						○	

FUNCTIONS

(1) Compares the (A2) data and the specified device data, Ⓢ, and stores lower data to (A2).



(2) The word device data, Ⓢ, remains unchanged after the LS instruction is executed.

REMARKS

The (A2) data may be overwritten by the execution result of the LS instruction. The (A2) data required should be saved before execution of the LS instruction.

RESTRICTIONS

1) Data used with the LS instruction is between ± 2.7 × 10^{-20} and ± 9.2 × 10^{18}.
The constant (K) specified during programming is between K ± 1 × 10^{-9} and K9.999 × 10^{9}.

2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

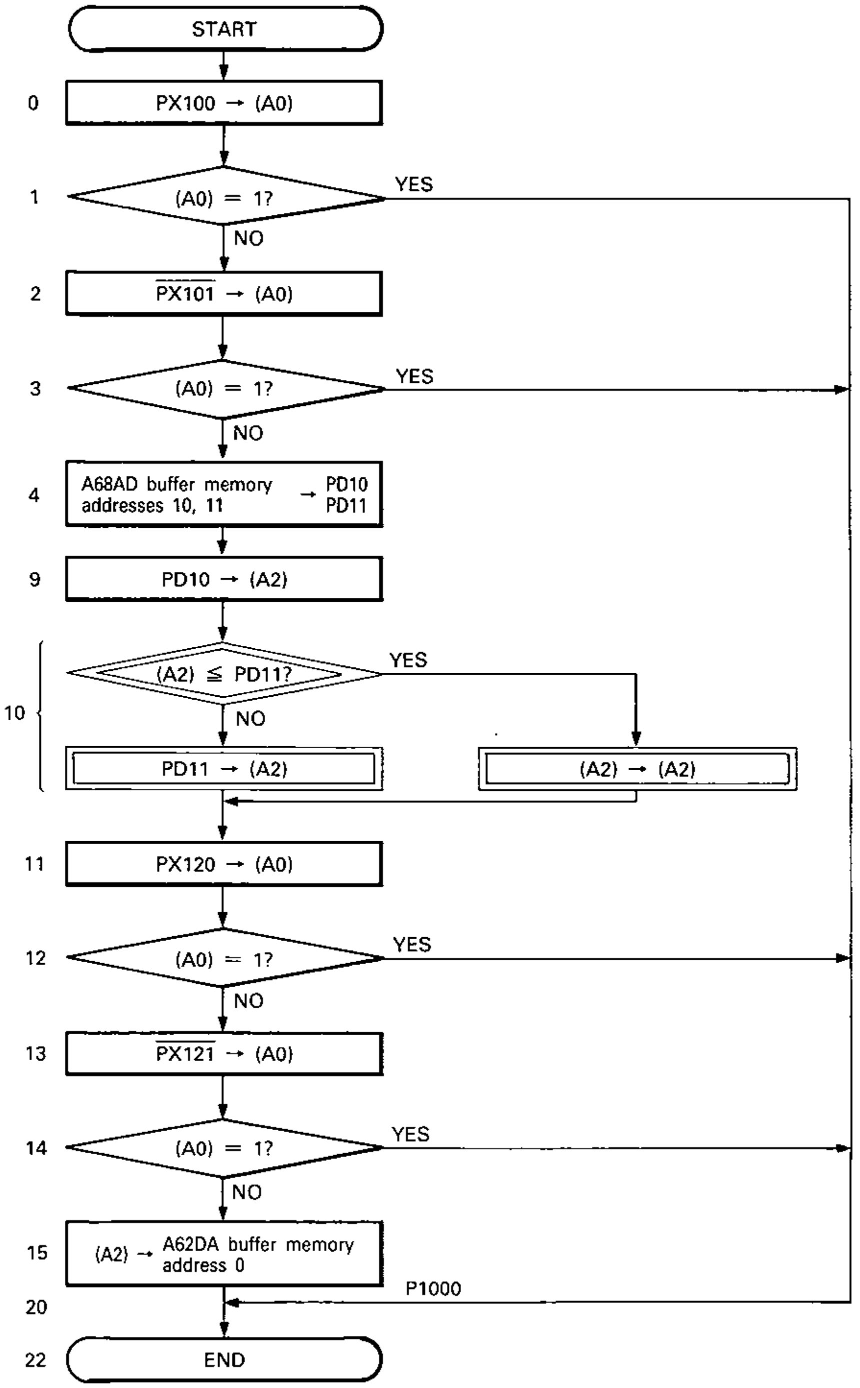
Example: If LS K145678 is entered in the program, it changes to LS K145600.

PROGRAM EXAMPLE

The following program compares the values read from the A68AD CH1 and CH2 and outputs the CH1 value to the A62DA CH1 if CH1 ≦ CH2 or the CH2 value if CH1 > CH2. (Program 10 used)

Main base unit configuration

Power supply module	A81CPU	A68AD	A62DA	AX10	AY10	Vacancy	Vacancy	Vacancy	Vacancy
		PX100 to PX11F PY100 to PY11F	PX120 to PX13F PY120 to PY13F	PX140 to PX14F	PY150 to PY15F				


START
0
PX100 → (A0)
1
(A0) = 1?
YES
NO
2
$\overline{PX101}$ → (A0)
3
(A0) = 1?
YES
NO
4
A68AD buffer memory addresses 10, 11 → PD10 PD11
9
PD10 → (A2)
10
(A2) ≦ PD11?
YES
NO
PD11 → (A2)
(A2) → (A2)
11
PX120 → (A0)
12
(A0) = 1?
YES
NO
13
$\overline{PX121}$ → (A0)
14
(A0) = 1?
YES
NO
15
(A2) → A62DA buffer memory address 0
P1000
20
22
END

```
 0 LDAB PX 100                       ⎫ Checks the A68AD operating
 1 JC     P 1000                     ⎪ status.
 2 NOT  PX 101                       ⎪ (Watch dog timer error, A/D con-
 3 JC     P 1000                     ⎭ version ready signal)
 4 FROM   H 0010   K 10   K 2 ······ Reads the CH1, CH2 digital values
                                     from the A68AD to PD10, 11.
 9 LDAF PD 10 ······················ Transfers PX10 data to (A2).
10 LS   PD 11 ······················ Stores (A2) data to (A2) if (A2) ≦
                                     PT11, and PD11 data if (A2) >
                                     PT11.
11 LDAB PX 120                       Checks the A62DA operating
12 JC     P 1000                     status.
13 NOT  PX 121                       (Watch dog timer error, A/D con-
14 JC     P 1000                     version ready signal)
15 TO     H 0012  K 0  A 2  K 1 ···· Writes the digital value to the
20 P    1000                         A62DA CH1.
22 END
```

MEMO

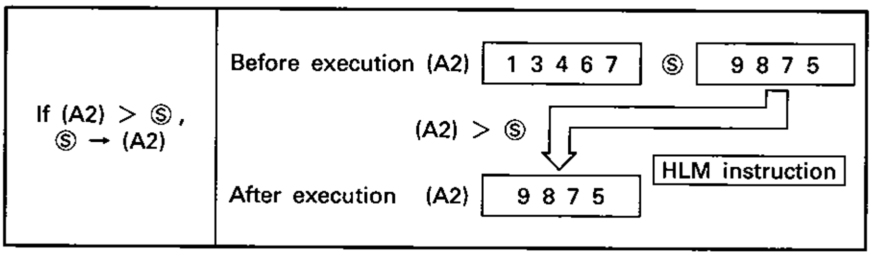
6.12.3 Clamping high limit value …… HLM

FORMAT	**HLM ␣ Ⓢ**

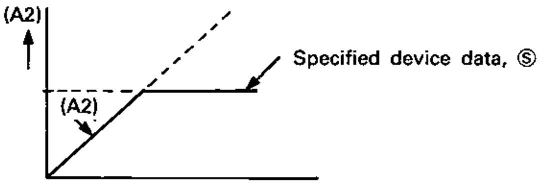
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant compared with (A2) data					○		○					○			1						○	

FUNCTIONS

(1) Compares the (A2) data and the specified device data, Ⓢ, and stores the specified device data to (A2) to limit the (A2) data if the (A2) data is greater than the specified device data.



(2) The (A2) data remains unchanged if the (A2) data is less than the specified device data, Ⓢ.



(3) The word device data, Ⓢ, remains unchanged after the HLM instruction is executed.

REMARKS

The (A2) data may be overwritten by the execution result of the HLM instruction. The (A2) data required should be saved before execution of the HLM instruction.

RESTRICTIONS

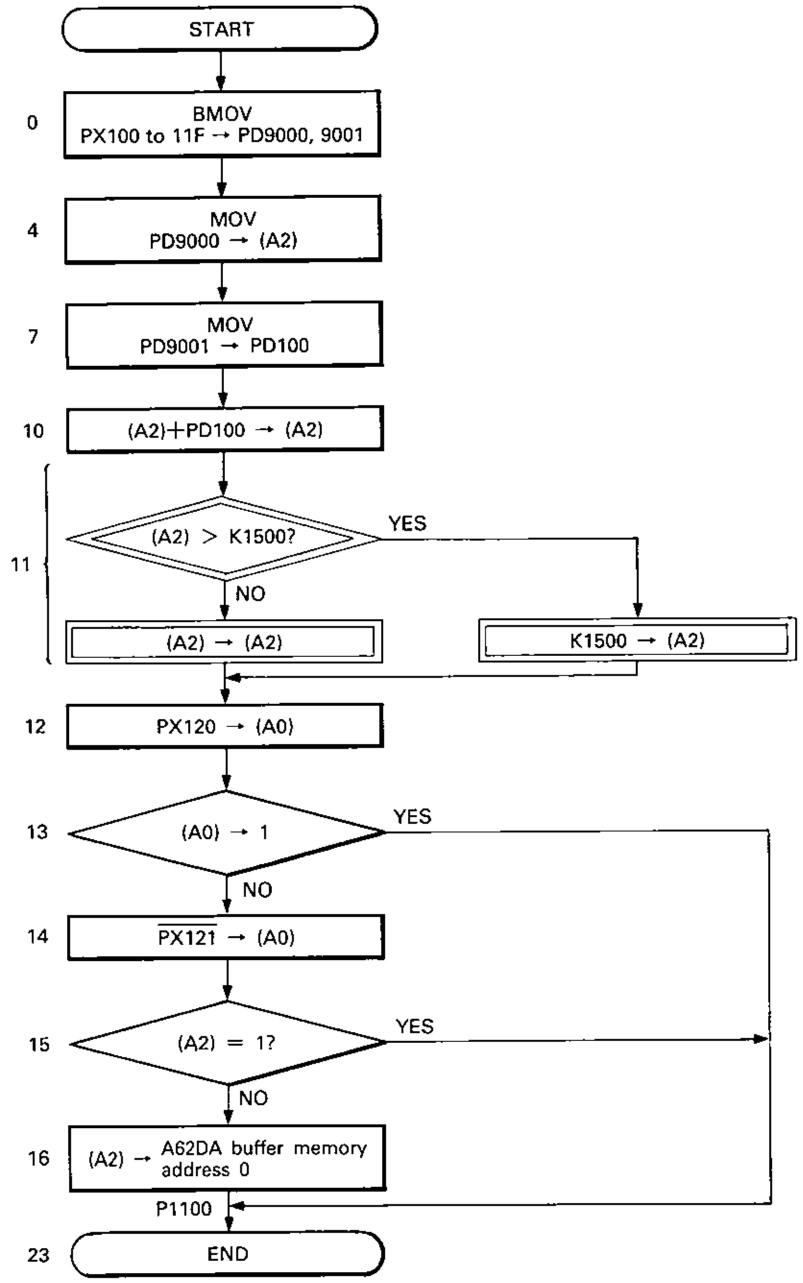
1) Data used with the HLM instruction is between $\pm 2.7 \times 10^{-20}$ and $\pm 9.2 \times 10^{18}$.
The constant (K) specified during programming is between $K \pm 1 \times 10^{-9}$ and $K9.999 \times 10^{9}$.

2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

Example: If HLM K976543 is entered in the program, it changes to HLM K976500.

PROGRAM EXAMPLE

The following program adds the incoming value from PX100 to 10F and the outgoing value from PX100 to 11F, limits the value at 1500 if it exceeds 1500, and outputs it to the A62DA. (Program 11 used)



```
 0 BMOV PX 100   PD 9000  K 2·············Stores PX100 to 10F data to
                                           PD9000 and PX110 to 11F data to
                                           PD9001.
 4 MOV  PD 9000        A 2················Stores PD9000 data to (A2).
 7 MOV  PD 9001      PD 100···············Stores PD9001 data to PD100.
10 +    PD 100····························Adds (A2) and PD100 data and
                                           stores the result to (A2).
11 HLM   K 1500 ··························The following steps are processed
12 LDAB PX 120                          ⎫  if (A2) data is equal to or greater
13 JC    P 1100                         ⎪  than 1500.
14 NOT  PX 121                          ⎪  Checks the A62DA operating
15 JC    P 1100                         ⎭  status.
                                           (Watch dog timer error, A/D con-
                                           version ready signal)
16 TO    H 0012  K 0   A 2   K 1·······Writes the digital value to the
21 P    1100                               A62DA CH1.
23 END
```

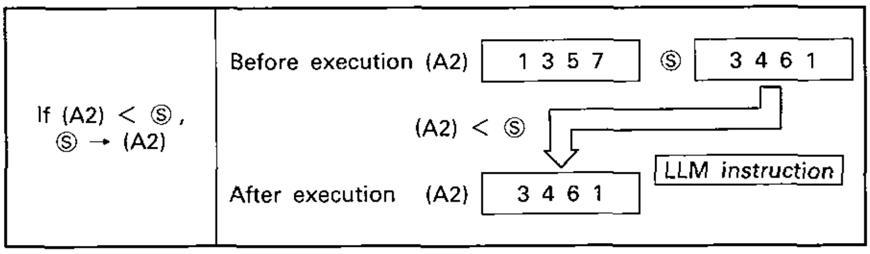
6.12.4 Clamping low limit value ...... LLM

FORMAT	LLM ␣ Ⓢ

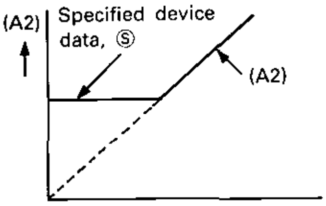
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
Ⓢ	Device number or constant compared with (A2) data					○		○					○			1						○	

FUNCTIONS

(1) Compares the (A2) data and the specified device data, Ⓢ, and stores the specified device data to (A2) to limit the (A2) data if the (A2) data is less than the specified device data.



(2) The (A2) data remains unchanged if the (A2) data is greater than the specified device data, Ⓢ.



(3) The word device data, Ⓢ, remains unchanged after the LLM instruction is executed.

REMARKS

The (A2) data may be overwritten by the execution result of the LLM instruction. The (A2) data required should be saved before execution of the LLM instruction.

RESTRICTIONS

1) Data used with the LLM instruction is between ± 2.7×10^{-20} and 9.2×10^{18}.
The constant (K) specified during programming is between K ± 1×10^{-9} and K9.999×10^{9}.

2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

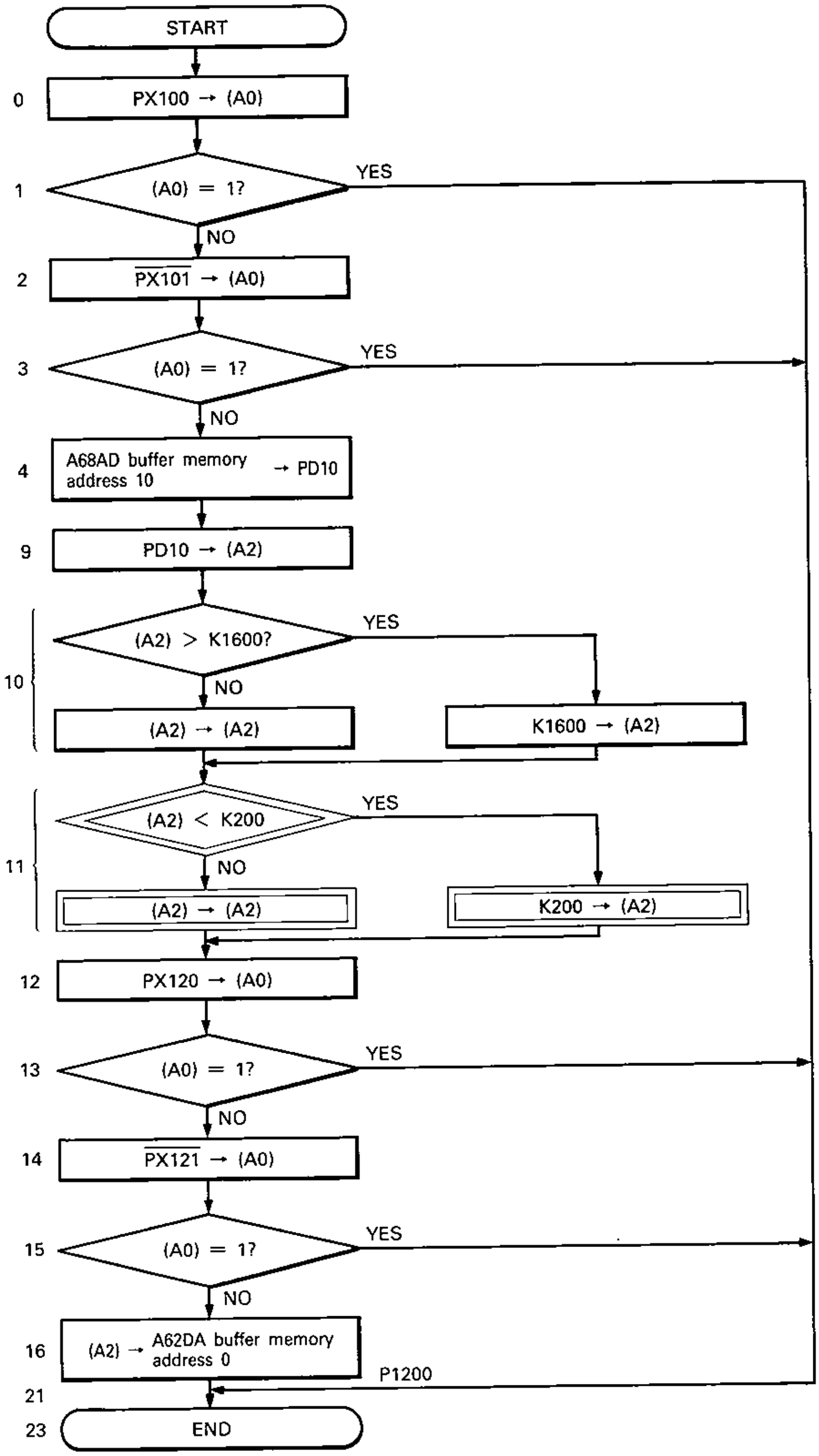
Example: If LLM K154366 is entered in the program, it changes to LLM K154300.

PROGRAM EXAMPLE

The following program limits the incoming digital value from the A68AD at 1600 if it exceeds 1600, limits at 200 if it drops below 200, and outputs it to the A62DA. (Program 12 used)

Main base unit configuration

Power supply module	A81CPU	A68AD	A62DA	AX10	AY10	Vacancy	Vacancy	Vacancy	Vacancy
		PX100 to PX11F PY100 to PY11F	PX120 to PX13F PY120 to PY13F	PX140 to PX14F	PY150 to PY15F				


START
0
PX100 → (A0)
1
(A0) = 1?
YES
NO
2
$\overline{\text{PX101}}$ → (A0)
3
(A0) = 1?
YES
NO
4
A68AD buffer memory address 10 → PD10
9
PD10 → (A2)
10
(A2) > K1600?
YES
NO
(A2) → (A2)
K1600 → (A2)
11
(A2) < K200
YES
NO
(A2) → (A2)
K200 → (A2)
12
PX120 → (A0)
13
(A0) = 1?
YES
NO
14
$\overline{\text{PX121}}$ → (A0)
15
(A0) = 1?
YES
NO
16
(A2) → A62DA buffer memory address 0
P1200
21
23
END

```
 0 LDAB PX 100                          ⎫ Checks the A68AD operating
 1 JC     P 1200                        ⎪ status.
 2 NOT  PX 101                          ⎪ (Watch dog timer error, A/D con-
 3 JC     P 1200                        ⎭ version ready signal)
 4 FROM   H 0010  K 10   PD 10    K 1····Reads the CH1 digital value from
                                          the A68AD to PD10.
 9 LDAF PD 10·····························Transfers PX10 data to (A2).
10 HLM    K 1600··························Limits at 1600 if the (A2) value
                                          exceeds 1600.
11 LLM    K 200···························Limits at 200 if the (A2) value
                                          becomes less than 200.
12 LDAB PX 120                          ⎫ Checks the A62DA operating
13 JC     P 1200                        ⎪ status.
14 NOT  PX 121                          ⎪ (Watch dog timer error, A/D con-
15 JC     P 1200                        ⎭ version ready signal)
16 TO     H 0012  K 0    A 2      K 1····Writes the digital value to the
21 P     1200                             A62DA CH1.
23 END
```

6.12.5 No operation …… NOP

FORMAT	NOP

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
																1							

FUNCTIONS

(1) A do-nothing instruction, having no influence on the previous operation.

(2) NOP is used to:

(a) Reserve a space for program debugging.
(b) Delete an instruction temporarily.

(3) All subsequent step numbers are corrected automatically if an instruction occupying more than one step is changed to NOP in the program already written.

6.12.6 Program end ······ END

FORMAT	END

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
																1							

FUNCTIONS

(1) Used to terminate a program and written at the end of the program.

(2) Program execution is terminated at the step of the END instruction.

(3) The program is started per sampling time specified by the user.

(4) Program execution continues up to the END instruction of the other program if the current program has no END instruction. For instance, if program 1 has no END instruction and program 2 has END, processing is performed up to program 2 after program 1 is started.
For further information on program execution, see Section 5.2.

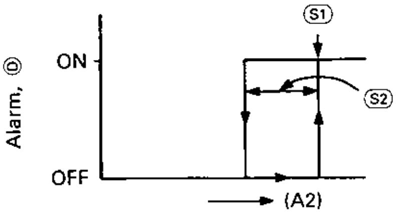
6.12.7 Alarm output at set value or greater ······ HAL

FORMAT	HAL ␣ (S1) ␣ (S2) ␣ Ⓓ

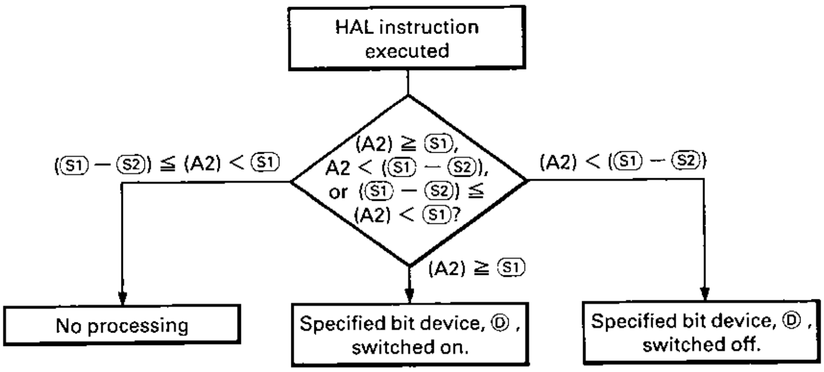
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(S1)	Device number or constant containing alarm set value							○					○			3						○	
(S2)	Device number or constant containing hysteresis value							○					○										
Ⓓ	Device number switched on to output alarm		○	○	○																		

FUNCTIONS

(1) Compares the (A2) data and the specified device data, (S1), and switches on the specified bit device, Ⓓ, if (A2) (S1). After switched on, the specified bit device, Ⓓ, is switched off when the (A2) data drops below ((S1) — specified hysteresis value, (S2)).



Sequence of instruction execution



(2) The ON/OFF state of the specified bit device, Ⓓ, is only checked when the HAL instruction is executed. The bit device state remains unchanged unless the instruction is executed.

Restrictions

1) Data used with the HAL instruction is between ± 2.7 × 10^{-20} and ± 9.2 × 10^{18}.
The constant (K) specified during programming is between K ± 1 × 10^{-9} and K ± 9.999 × 10^{9}.

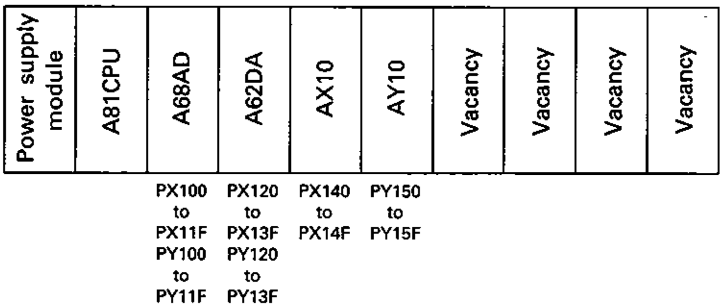
2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

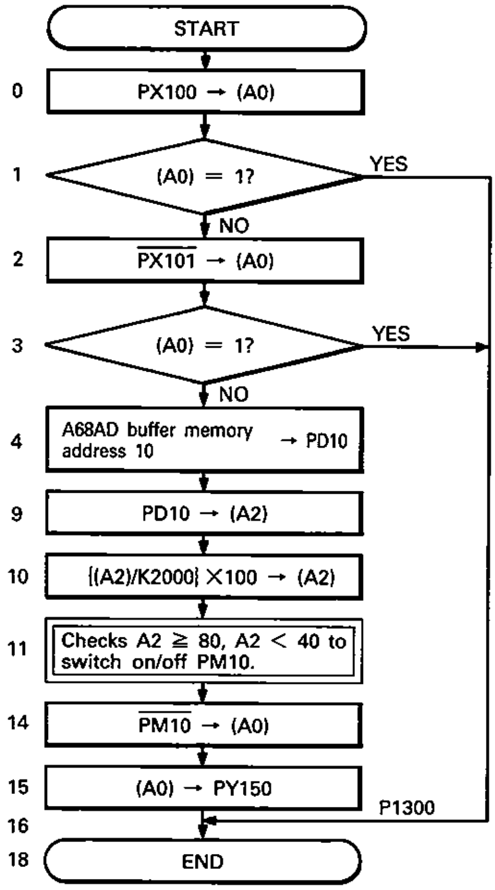
Example: If HAL K135467 PD10 PM50 is entered in the program, it changes to HAL K135400 PD10 PM50.

PROGRAM EXAMPLE

The following program reads a digital value from the A68AD, converts it into a % value, and switches off PY150 if that value reaches or exceeds 80 and switches on PY150 if the value becomes less than 40. (Program 13 used)

Main base unit configuration





```
 0 LDAB PX 100                          ) Checks  the  A68AD  operating
 1 JC     P 1300                        | status.
 2 NOT  PX 101                          | (Watch dog timer error, A/D con-
 3 JC     P 1300                        ) version ready signal)
 4 FROM   H 0010   K 10   PD 10   K 1···Reads the CH1 digital value from
                                          the A68AD to PD10.
 9 LDAF PD 10 ····························Transfers PD10 data to (A2).
10 PCT    K 2000··························Converts the read digital value
                                          into a % value and stores the
                                          result to (A2).
11 HAL    K 80     K 20   PM··············Switches PM10 on if (A2) ≧ 80
                                          and off if (A2) < 40.
14 NOT  PM 10·····························Flips PM10 data and transfers the
                                          result to (A0).
15 STAB PY 150····························Transfers (A0) data to PY150.
16 P    1300                              (Switches PY150 off if (A2) ≧ 80
18 END                                    and on if (A2) < 40.)
```

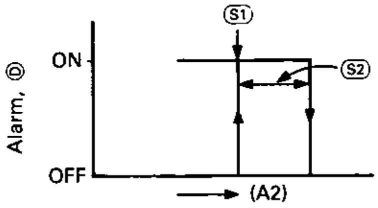
6.12.8 Alarm output at set value or less …… LAL

FORMAT	LAL ␣ (S1) ␣ (S2) ␣ Ⓓ

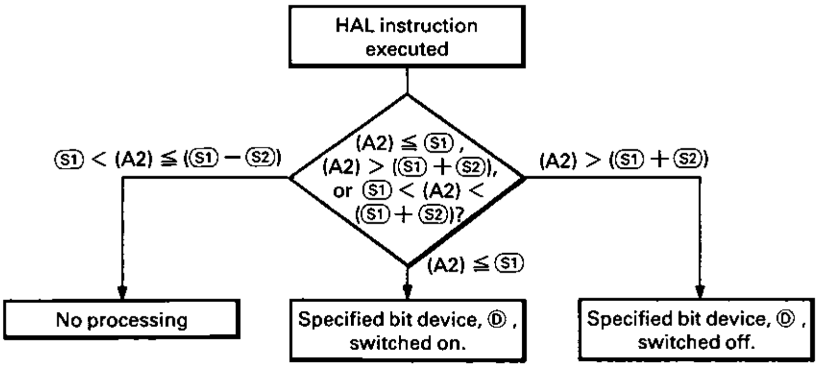
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(S1)	Device number or constant containing alarm set value							○					○			3							
(S2)	Device number or constant containing hysteresis value							○					○									○	
Ⓓ	Device number switched on to output alarm		○	○	○																		

FUNCTIONS

(1) Compares the (A2) data and the specified device data, (S1), and switches on the specified bit device, Ⓓ, if (A2) (S1). After switched on, the specified bit device, Ⓓ, is switched off when the (A2) data becomes greater than ((S1) + specified hysteresis value, (S2)).



Sequence of instruction execution



(2) The ON/OFF state of the specified bit device, Ⓓ, is only checked when the LAL instruction is executed. The bit device state remains unchanged unless the instruction is executed.

Restrictions

1) Data used with the LAL instruction is between ± 2.7 × 10^{-20} and ± 9.2 × 10^{18}.
The constant (K) specified during programming is between K ± 1 × 10^{-9} and K ± 9.999 × 10^{9}.

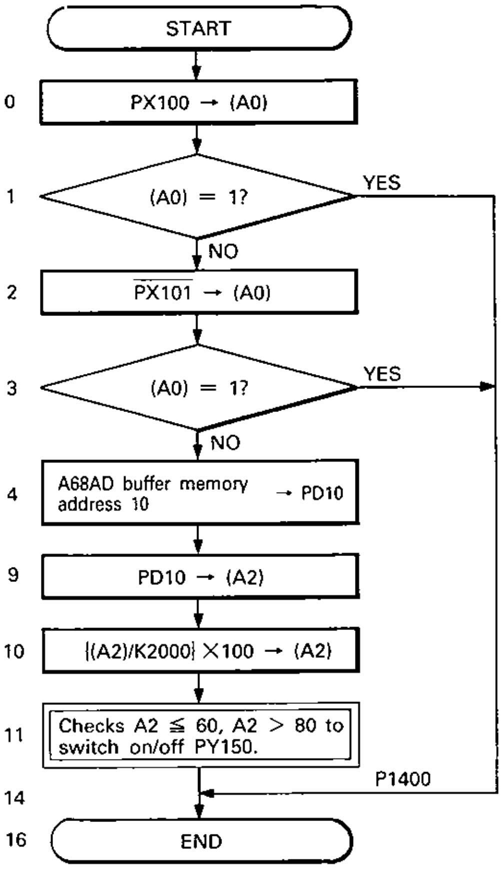
2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

Example: If LAL K256789 PD50 PY100 is entered in the program, it changes to LAL K256700 PD50 PY100.

PROGRAM EXAMPLE

The following program reads a digital value from the A68AD, converts it into a % value, and switches on PY150 if that value reaches or drops below 60 and switches off PY150 if it exceeds 80. (Program 14 used)

Power supply module	A81CPU	A68AD	A62DA	AX10	AY10	Vacancy	Vacancy	Vacancy	Vacancy
		PX100 to PX11F PY100 to PY11F	PX120 to PX13F PY120 to PY13F	PX140 to PX14F	PY150 to PY15F				



```
 0 LDAB PX 100                              Checks the A68AD operating
 1 JC     P 1400                            status.
 2 NOT  PX 101                              (Watch dog timer error, A/D con-
 3 JC     P 1400                            version ready signal)
 4 FROM   H 0010  K 10    PD 10    K 1 ·Reads the CH1 digital value from
                                            the A68AD to PD10.
 9 LDAF PD 10 .............................Transfers PD10 data to (A2).
10 PCT    K 2000...........................Converts the read value into a %
                                            value and stores the result to (A2).
11 LAL    K 60   K 20   PY 150 ...........Switches PY150 on if (A2) ≦ 60
14 P     1400                               and off if (A2) > 80.
16 END
```

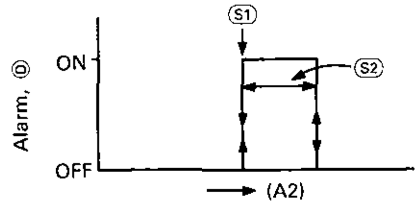
6.12.9 Alarm output at set value …… SAL

FORMAT	SAL ␣ (S1) ␣ (S2) ␣ Ⓓ

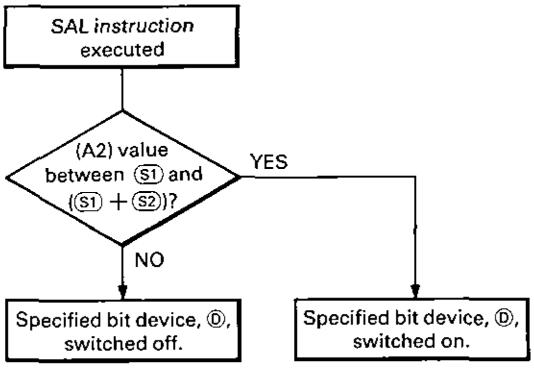
	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
(S1)	Device number or constant containing alarm set value							○					○			3						○	
(S2)	Device number or constant containing hysteresis value							○					○										
Ⓓ	Device number switched on to output alarm		○	○	○																		

FUNCTIONS

(1) Switches on the specified bit device, Ⓓ, if the (A2) data is between the specified device data, (S1), and ((S1) + hysteresis value, (S2)).
The specified bit device, Ⓓ, is switched off when the (A2) data is outside the above range.



Sequence of instruction execution



(2) The ON/OFF state of the specified bit device, Ⓓ, is only checked when the SAL instruction is executed. The bit device state remains unchanged unless the instruction is executed.

Restrictions

1) Data used with the SAL instruction is between ± 2.7 × 10^{-20} and ± 9.2 × 10^{18}.
The constant (K) specified during programming is between K ± 1 × 10^{-9} and K ± 9.999 × 10^{9}.

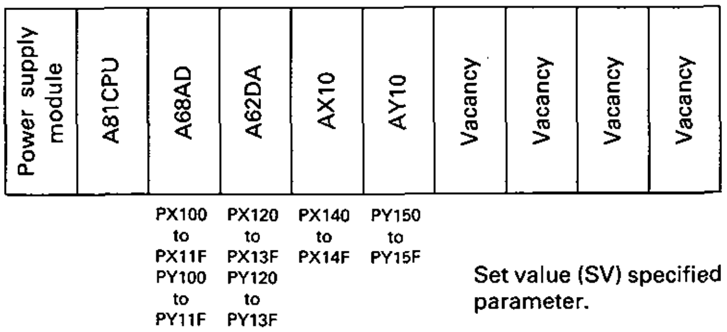
2) Any specified constant (K) outside the range −32768 to 32767 is set to 0 during programming with the exception of the four most significant digits.

Example: If SAL K567891 K25 PY150 is entered in the program, it changes to SAL K567800 K25 PY150.

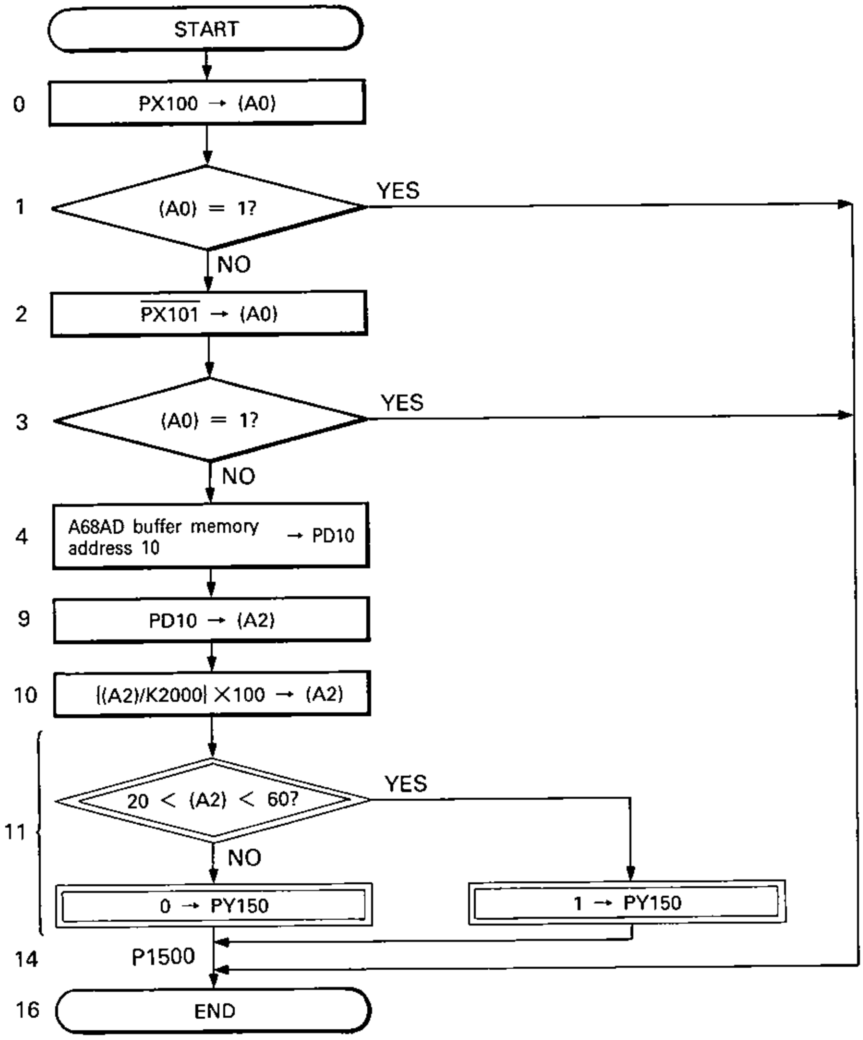
PROGRAM EXAMPLE

The following program reads a digital value from the A68AD, converts it into a % value, and outputs that value to the output module if it is between 20 and 60. (Program 15 used)

Main base unit configuration



Set value (SV) specified as PD1 by the parameter.


START
0
PX100 → (A0)
1
(A0) = 1?
YES
NO
2
$\overline{\text{PX101}}$ → (A0)
3
(A0) = 1?
YES
NO
4
A68AD buffer memory address 10 → PD10
9
PD10 → (A2)
10
[(A2)/K2000] ×100 → (A2)
11
20 < (A2) < 60?
YES
NO
0 → PY150
1 → PY150
14
P1500
16
END

```
 0 LDAB PX 100                         } Checks the A68AD operating
 1 JC     P 1500                       } status.
 2 NOT  PX 101                         } (Watch dog timer error, A/D con-
 3 JC     P 1500                       } version ready signal)
 4 FROM   H 0010  K 10   PD 10   K 1····Reads the CH1 digital value from
                                         the A68AD to PD10.
 9 LDAF PD 10 ···························Transfers PD10 data to (A2).
10 PCT    K 2000 ························Converts the read value into a %
                                         value and stores the result to (A2).
11 SAL    K 20      K 60      PY 150 ····Switches PY150 on if (A2) data is
14 P     1500                            between 20 and 60.
16 END
```

6.12.10 Monitoring by the AD57 ······ DISP

FORMAT	DISP

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
—	———										○					1			○			○	

FUNCTIONS

(1) Allows the PID control status to be monitored on the CRT connected to the AD57 CRT controller module by specifying the required screen number using 16-bit binary data stored in accumulator (A1).

DISP	(A1) 3 ↓ PID control status of loops 9 to 12 is monitored on the CRT connected to the AD57.

(2) The macro functions (PID control status) of 64 loops are displayed in groups of 4 loops.

(3) The PID control status is monitored in blocks of 4 loops in order of data numbers (1 to 16) stored in (A1). For example, the PID control status of loops 5 to 8 is monitored if the (A1) data is 2.

(A1) Data	Loop Numbers Monitored on CRT
1	1 to 4
2	5 to 8
3	9 to 12
to	to
15	57 to 60
16	61 to 64

(3) When the PID control status is monitored, the DISP instruction must be executed every scan because the CRT monitoring display is updated at the time when the DISP instruction is executed.

Restrictions

1) The A1 value used with the DISP instruction is between 1 and 16. Use of any other value will result in an error.

2) The error code 55 is displayed if the DISP instruction is executed in the absence of the AD57.

PROGRAM EXAMPLE

The following example displays the PID control status of loops 17 to 20 on the CRT connected to the AD57.

```
0
to                                    } PID control program.
n
n+1 LDAW K5·························Sets 5 to accumulator (A1).
n+2 DISP·····························Monitors loops 17 to 20.
n+3 END
```

6.12.11 Executing the macro function ...... LOOP

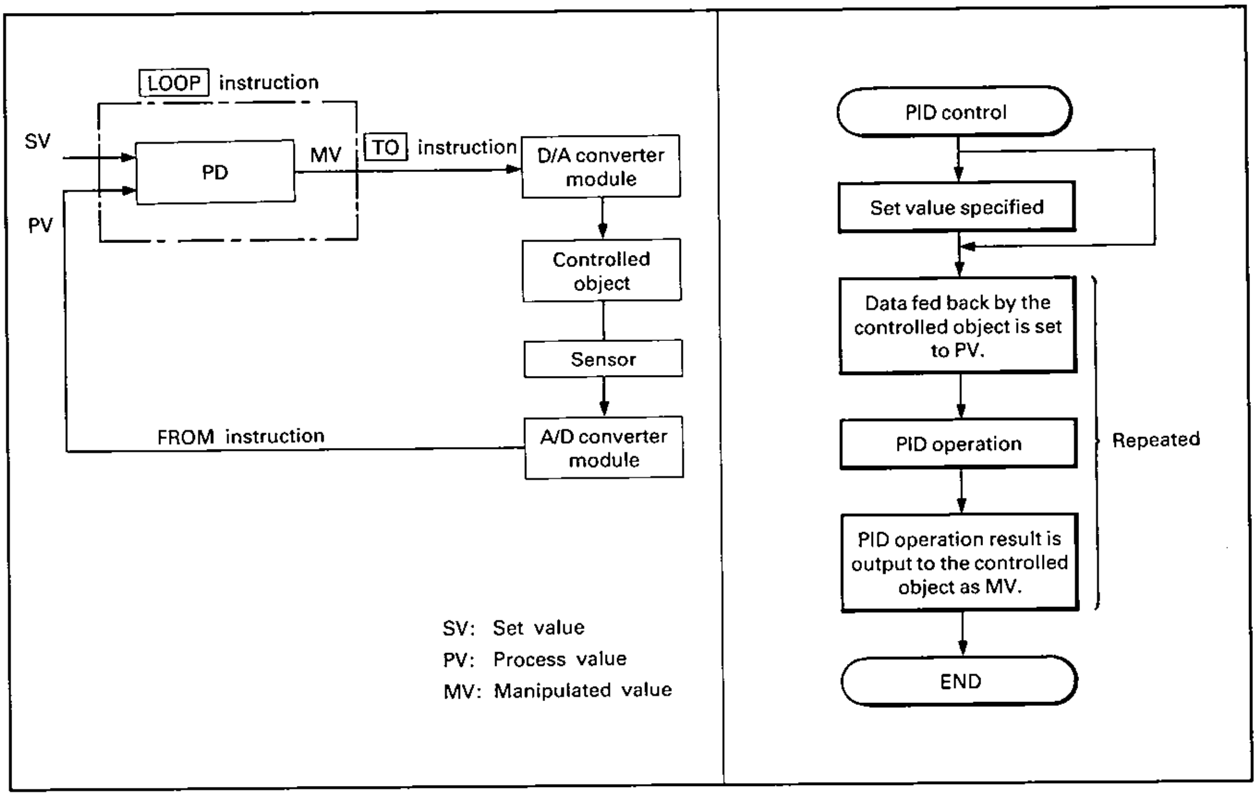
FORMAT	**LOOP ␣ ⓝ**

	Set Data	Set Device														Number of Steps	Error Occurrence						
		PX	PY	PM	SP.PM	PT		PD	SP.PD	A0	A1	A2	K	H	P		51	54	55	56	57	58	59
ⓝ	Loop number for execution of macro function												○			2						○	

FUNCTIONS

(1) Performs PID operation in accordance with the set value (SV) and process value (PV) of the specified loop number, n, and stores the result to the device for storing manipulated value (MV).

(2) The MV output range is defined by the macro function parameter between −2.50 and 102.50% in accordance with the MV high limit (MH) and low limit (ML).



Restrictions

1) n should be specified between 1 and 64.

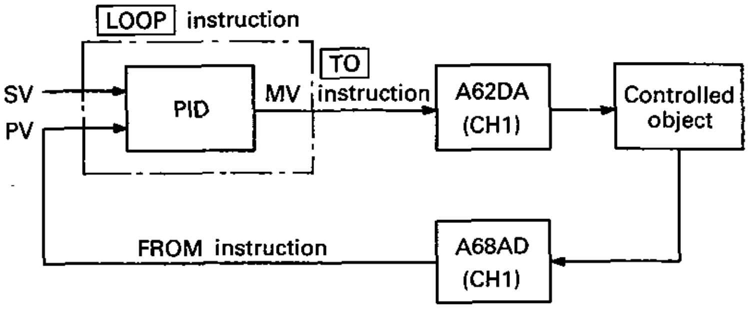
PROGRAM EXAMPLE

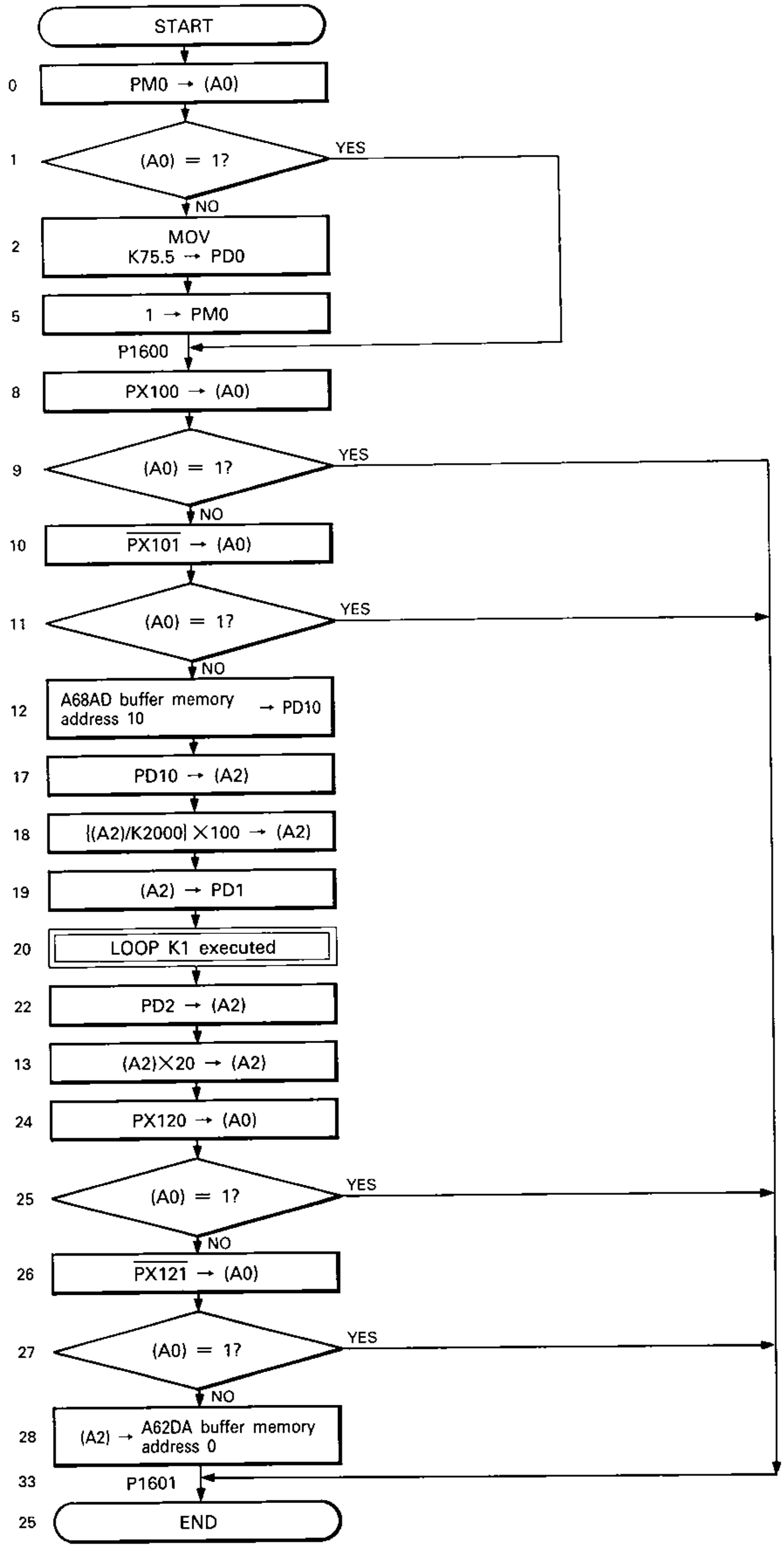
The following program reads PV from the controlled object via the A68AD CH1 and writes MV to the controlled object via the A62DA CH1.

Main base unit configuration

Power supply module	A81CPU	A68AD	A62DA	AX10	AY10	Vacancy	Vacancy	Vacancy	Vacancy
		PX100 to PX11F PY100 to PY11F	PX120 to PX13F PY120 to PY13F	PX140 to PX14F	PY150 to PY15F				

- SV: PD0
- PV: PD1
- MV: PX2
- Parameters already been defined.




START
0
PM0 → (A0)
1
(A0) = 1?
YES
NO
2
MOV
K75.5 → PD0
5
1 → PM0
P1600
8
PX100 → (A0)
9
(A0) = 1?
YES
NO
10
$\overline{PX101}$ → (A0)
11
(A0) = 1?
YES
NO
12
A68AD buffer memory address 10 → PD10
17
PD10 → (A2)
18
{(A2)/K2000} × 100 → (A2)
19
(A2) → PD1
20
LOOP K1 executed
22
PD2 → (A2)
13
(A2) × 20 → (A2)
24
PX120 → (A0)
25
(A0) = 1?
YES
NO
26
$\overline{PX121}$ → (A0)
27
(A0) = 1?
YES
NO
28
(A2) → A62DA buffer memory address 0
33
P1601
25
END

```
 0 LDAB PM 0
 1 JC    P 1600
 2 MOV   K 75.5        PD 0            } Sets SV to PD0.
 5 SET  PM 0
 6 P    1600
 8 LDAB PX 100
 9 JC    P 1601                        } Checks the A68AD operating
10 NOT  PX 101                           status.
11 JC    P 1601
12 FROM  H 0010 K 0  PD 10    K 1 ···Reads CH1 digital value from the
                                        A68AD to PD10.
17 LDAF PD 10                          } Converts the read digital value
18 PCT   K 2000                          into a % value and stores the
19 STAF PD 1                             result to PD1 (PV).
20 LOOP  K 1 ·····························Performs PID operation for loop 1.
22 LDAF PD 2                           } Converts PD2 value (MV) into a
23 ●     K 20                            digital value for analog output.
24 LDAB PX 120
25 JC    P 1601                        } Checks the A62DA operating
26 NOT  PX 121                           status.
27 JC    P 1601
28 TO    H 0012 K 10 PD 10    K 1 ···Writes the digital value to the
                                        A62DA CH1.
33 P    1601
35 END
```

7. ERROR CODES

When the A81CPU self-detects an error, the "RUN" LED flickers and the corresponding error code is displayed on the LED indicator and written to the special registers.

7.1 Error Code List

Error Code	CPU Status	Definition and Cause	Corrective Action	D9103 Content (BIN value)	Special Registers D9104 to 9199 Corresponding to Programs		
					Error code	Step number	Detail
49	Stop	CPU hardware fault.	Consult Mitsubishi representative.	49	–	–	–
50	Stop	END instruction is not executed in one program after 5 seconds.	Correct the faulty program.	50	–	–	–
51	Stop	Incorrect FROM and/or TO instruction execution. (1) Special function module control bus error.	Hardware fault (CPU, special function module and/or base unit). Consult Mitsubishi representative.	51	–	–	–
52	Stop	Two or more computer link modules are installed in one CPU system.	Correct system configuration.	52	–	–	–
53	Stop	Operation element fault.	CPU hardware fault, consult Mitsubishi representative.	53	–	–	–
54	Stop	No response from special function module after execution of FROM and/or TO instruction. (1) Possible hardware fault.	Consult Mitsubishi representative.	–	54	○	–
55	Stop	No special function module in I/O slot addressed by FROM and/or TO instruction.	Examine the program step indicated by the diagnostics and correct.	–	55	○	–
56	Stop	(1) No jump destination or several destinations specified for the CALL JMP or JC instruction. (2) RET instruction has been executed with no corresponding CALL instruction. (3) END instruction has been executed but no RET instruction executed after CALL instruction.	Examine the program step indicated by the diagnostics and correct.	–	56	○	–
57	Stop	An unrecognized instruction code is being used.	Examine the program step indicated by the diagnostics and correct.	–	57	○	–

Table 7.1 Error Code List (Continue)

Error Code	CPU Status	Definition and Cause	Corrective Action	D9103 Content (BIN value)	Special Registers D9104 to 9199 Corresponding to Programs		
					Error code	Step number	Detail
58	Faulty program only stopped.	(1) The result of BCD conversion has exceeded the specified range (9999). (2) Specified operation device range has been exceeded, e.g. overflow, underflow.	Examine the program step indicated by the diagnostics and correct.	–	58	○	○
59	(1) or (2) selected by parameter setting. (1) Faulty program only stopped. (2) Operation continued with the set data limited.	Invalid data specified for the PRW instruction.	Examine the program step indicated by the diagnostics and correct.	–	59	○	○
60	Run	Output module fuse blown.	Check the fuse blown indicator LED on output modules.	60	–	–	–
61	Run	(1) Battery voltage low. (2) Battery not connected.	(1) Change the battery. (2) Connect the battery.	61	–	–	–

Table 7.1 Error Code List

Further error definitions

For error codes "58" and "59" any of the following numbers is written to the special register in hexadecimal to indicate further definition of the error.

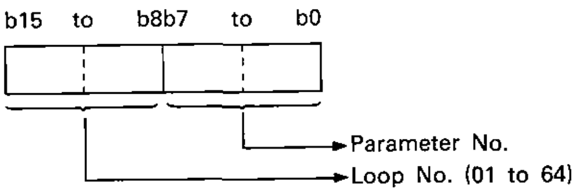
Error Code	Error Definition	
	Number	Description
58	01	Overflow Operation result is more than ±9.2 × 10^{18}.
	02	Underflow Operation result is less than ±2.7 × 10^{-20}.
	04	Negative or 0 operation Operation data is negative or 0.
	08	Division by 0 Division data is 0.
	0C	Operation data too great Operation data is too great.
	10	BCD conversion error BCD conversion error is more than 9999.
	20	Invalid data Invalid data specified for the corresponding instruction.
59	Loop No. + parameter No.	Parameter error Faulty loop and parameter numbers are written as shown below. 

Table 7.2 Further Error Definition List

7.2 Error Codes Displayed During Instruction Execution

Any of the following error codes may be displayed when the corresponding instruction is executed. These error codes are indicated in the Error Occurrence column of the corresponding instructions in Section 6. For full information on the symbols of the causes and corrective actions (A to L), see the next page.

Instruction	Cause and Corrective Action for Error Code						Remarks	Instruction	Cause and Corrective Action for Error Code						Remarks
	51	54	55	56	58	59			51	54	55	56	58	59	
AND					K			LM							
ABS								LS					F, K		
ASIN					I			LLM					F, K		
ACOS					I			LAL					F, K		
ATAN								LOOP							
BSET					K			MOV					E, K		
BRST					K			NOT					K		
BTST					K			NOP							
BCD					E, J, K			OR					K		
BIN					J, K			PRR					K		
BMOV					E, K			PRW					E, K	L	
CALL				D				PCT(%)					F, H, K		
COS								RST					K		
DISP	A	B	C		E, K			RET				D			
EQAW					F, K			SET					K		
EQAF					F, K			STAB					K		
EXP					I			STAW					K		
END								STAF					E, K		
FMOV					E, K			SQRT(√)					G		
FROM(DFROM)	A	B	C		K			SIN							
GTAW					F, K			SAL					F, K		
GTAF					F, K			TO(DTO)	A	B	C		E, K		
HS					F, K			TAN					E		
HLM					F, K			WNOT					K		
HAL					F, K			WAND					K		
JMP				D				WOR					K		
JC				D				WXOR					K		
LDAB					K			XOR					K		
LDAW					K			+					E, K		
LDAF					K			−					F, K		
LTAW					F, K			*					E, F, K		
LTAF					F, K			/					F, H, K		
LOG					G										

Error causes and corrective actions

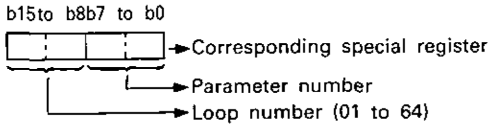
Symbol	Cause	Corrective Action
A	Incorrect FROM and/or TO instruction execution. (Special function module control bus error)	Hardware fault (special function module, A81CPU module and/or base unit). Consult Mitsubishi representative.
B	No response from special function module after execution of FROM and/or TO instruction. (Possible hardware fault)	*Consult Mitsubishi representative.*
C	No special function module in I/O slot addressed by FROM and/or TO instruction.	Examine the program step (stored in special registers D9104 to 9199 in groups of program numbers) indicated by the diagnostics and correct.
D	(1) No jump destination or several destinations specified for the CALL, JMP or JC instruction. (2) RET instruction has been executed with no corresponding CALL instruction executed. (3) END instruction has been executed but no RET *instruction executed after* CALL *instruction.* (4) CALL instruction nested to a level of five or more.	• Examine the program step (stored in special registers D9104 to 9199 in groups of program numbers) indicated by the diagnostics and correct. • The step of the END instruction is stored as a faulty step if the END instruction has been executed without executing the RET instruction.
E	Operation result is outside the range $\pm 9.2 \times 10^{18}$ (32-bit floating-point data) or −2147483647 to +2147483647 (32-bit binary data). ("01_H" is written to the special register for storing further error definition corresponding to the faulty program.)	Examine the program step (stored in special registers D9104 to 9199) indicated by the diagnostics and correct.
F	Operation result is outside the range $\pm 2.7 \times 10^{-20}$. ("02_H" is written to the special register for storing further error definition corresponding to the faulty program.)	
G	Value used with the SQRT ($\sqrt{\ }$), LOG, LM instruction is negative. ("04_H" is written to the special register for storing further error definition corresponding to the faulty program.)	
H	Division by 0 ("08_H" is written to the special register for storing further error definition *corresponding to the faulty program.)*	
I	Value used with the $\sin^{-1}$, $\cos^{-1}$, e^x instruction is outside the specified range. ($\sin^{-1}$, $\cos^{-1}$ must be between 0 and 1, e^x between −45.05845 and 43.66573.) ("$0C_H$" is written to the special register for storing further error definition corresponding to the faulty program.)	
J	BCD conversion result or BIN conversion source data is greater than 9999. ("10_H" is written to the special register for storing further error definition corresponding to the faulty program.)	
K	Invalid device (number) specified for the corresponding instruction. ("20_H" is written to the special register for storing further error definition corresponding to the faulty program.)	
L	Invalid data specified for the PRW instruction. (The following data is written to the special register for storing further error definition corresponding to the faulty program.) 	

Table 7.4 Error Causes and Corrective Actions

IMPORTANT

The components on the printed circuit boards will be damaged by static electricity, so avoid handling them directly. If it is necessary to handle them take the following precautions.

(1) Ground human body and work bench.

(2) Do not touch the conductive areas of the printed circuit board and its electrical parts with any non-grounded tools etc.

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE:MITSUBISHI DENKI BLDG MARUNOUCHI TOKYO 100 TELEX: J24532 CABLE MELCO TOKYO
NAGOYA WORKS : 1-14 , YADA-MINAMI 5 , HIGASHI-KU , NAGOYA , JAPAN

IB (NA) 66168-A (8809) MEE Printed in Japan