DATA SHEET

NEC
ELECTRON DEVICE

MOS INTEGRATED CIRCUIT
μPD17005

DIGITAL TUNING SYSTEM HARDWARE BUILT-IN 4-BIT SINGLE-CHIP MICROCONTROLLER

μPD17005 is a 4-bit single-chip CMOS micro controller which contains digital tuning system hardware.

17K architecture is used for CPU, data and memory manipulations and various types of operations, and peripheral hardware control can be performed directly by one instruction.

Peripheral hardware devices include a prescaler which operates up to 150 MHz, PLL frequency synthesizer, LPF (Low Pass Filter) amplifier, and frequency counter for digital tuning in addition to various types of input/output ports, LCD controller/driver, A/D converter, D/A converter (PWM output), and clock generator ports.

Consequently, a high performance digital tuning system with a variety of functions can be constructed using only one chip. μPD17005 is pin-compatible with μPD17003A and its memory size (ROM) is reduced. One-time PROM version μPD17P005 is available as μPD17005, and μPD17P005 can be used for program evaluation of μPD17005 at small volume production.

FEATURES

- Using 17K architecture
- Program memory (ROM)
  16K bytes (7932 steps x 16 bits)
- General purpose data memory (RAM)
  432 nibble (432 words x 4 bits)
- Instruction execution time
  4.44 μs (using 4.5 MHz quarts oscillator)
- Decimal operation enabled
- Table reference enabled
- Built-in PLL frequency synthesizer hardware
  Dual modules prescaler (150 MHz Max.), programmable divider, phase comparator, charge pump, and LPF amplifier
- Various types of peripheral hardware
  General purpose input/output ports, LCD controller/driver, serial interface, A/D converter, D/A converter (PWM output), clock generator, ports, and frequency counter
- Various types of interrupt
  External interrupt: 2 channels
  Internal interrupt: 3 channels
- Power On Reset, resetting by a CE pin, and built-in blackout detection circuit
- CMOS low power consumption
- Power supply voltage 5 V ± 10 %

ORDERING INFORMATION

Order Code	Package	Quality Grade
μPD17005GF-xxx-3B9	80-pin plastic QFP (14 x 20)	Standard

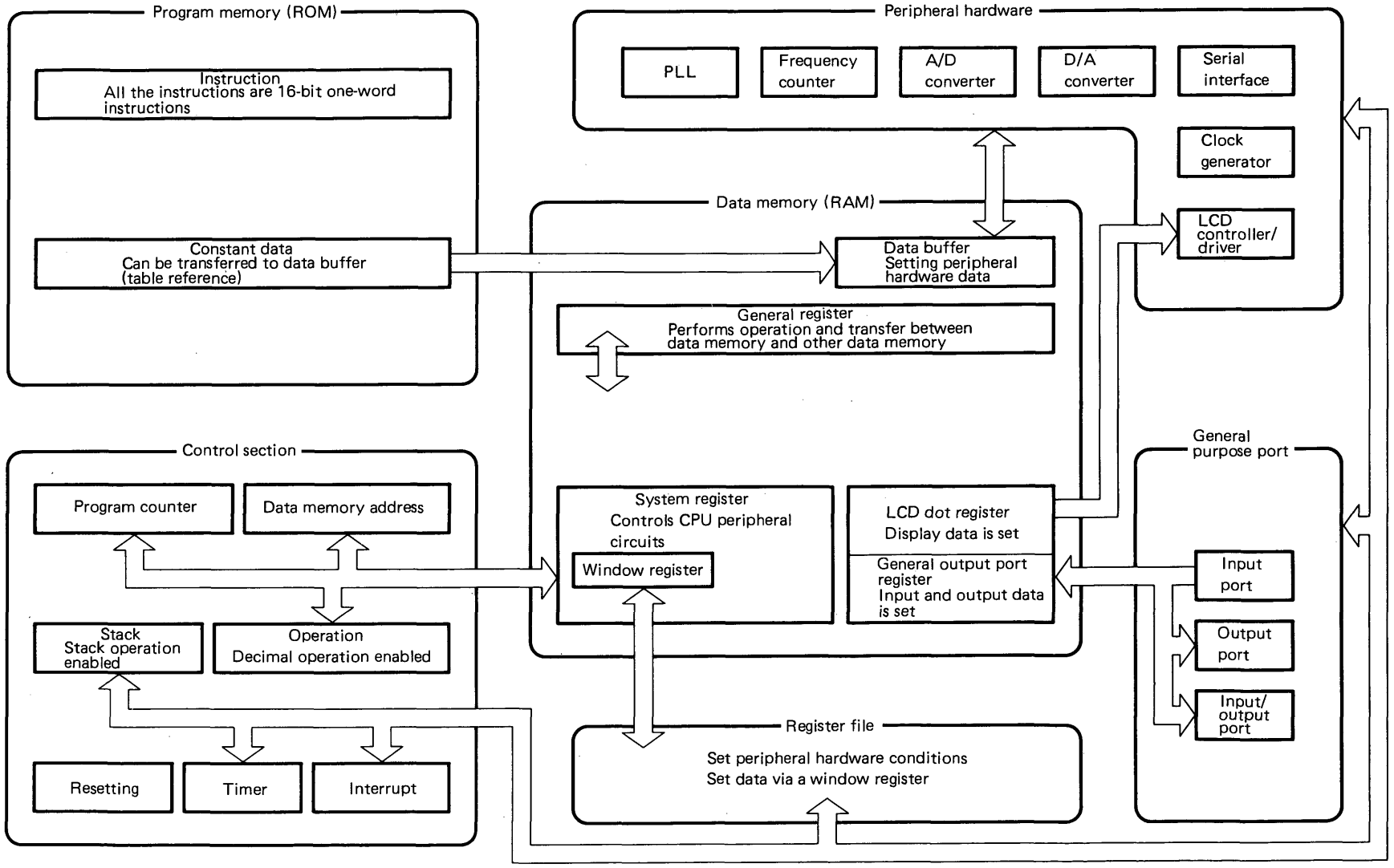
Please refer to "Quality grade on NEC Semiconductor Devices" (Document number IEI-1209) published by NEC Corporation to know the specification of quality grade on the devices and its recommended applications.

Document No. IC-2451
(O. D. No. IC-7915)
Date Published December 1990 P
Printed in Japan

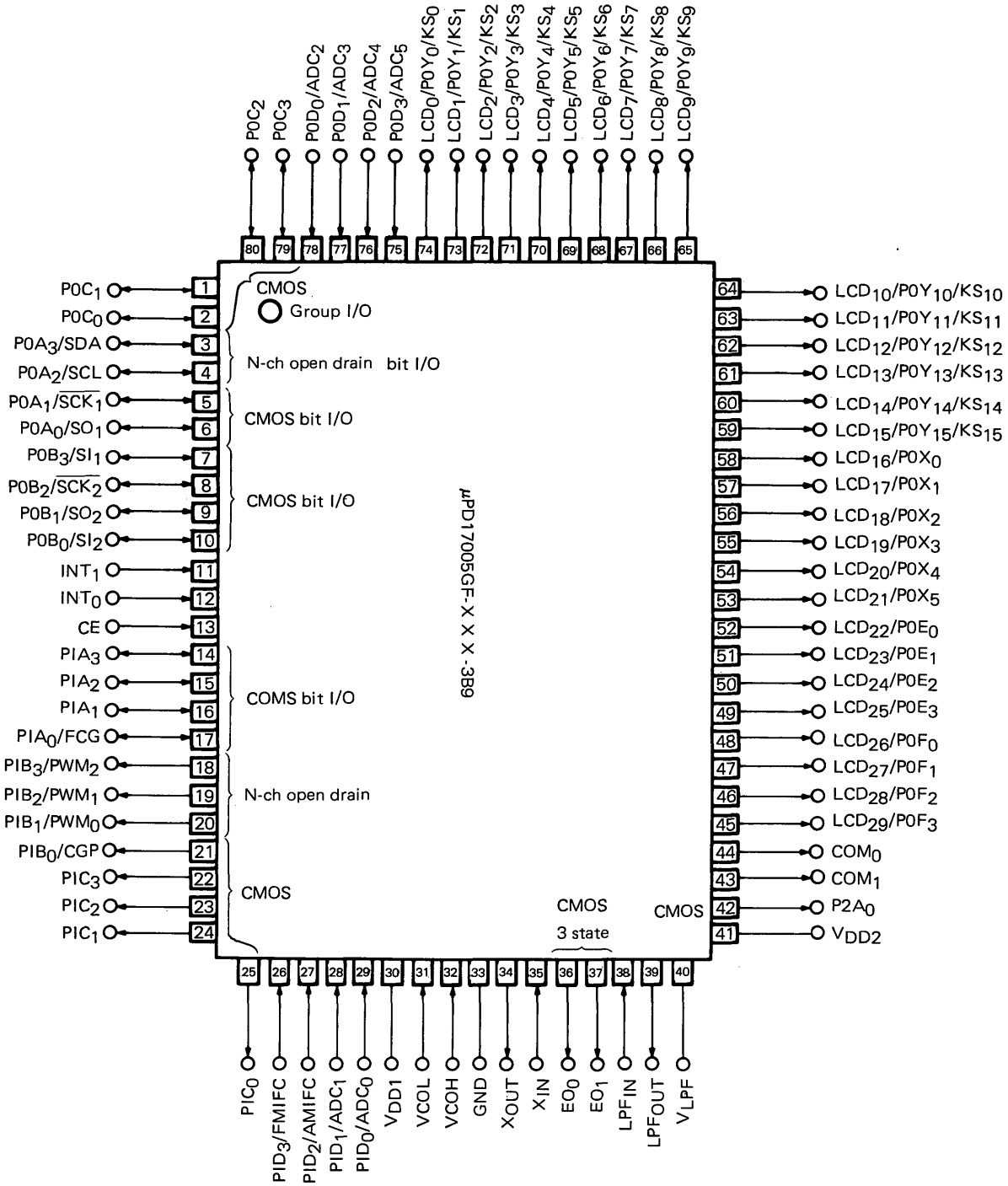
μPD17005 FUNCTION OUTLINE

Item	Function
Program memory (ROM)	• 16K bytes (7932 steps x 16 bits) Table reference area: up to 7932 steps
General data memory (RAM)	• 432 nibble (432 words x 4 bits) Data buffer : 4 nibbles General register : 16 nibbles
System register	• 12 nibbles
Register file	• 33 nibbles (control register)
General port register (including LCD dot data register)	• 24 nibbles
Instruction execution time	• 4.44 μs (using 4.5 MHz quarts oscillator)
Stack level	• 7 levels (stack operation enabled)
General purpose port	• Input/output port : 16 • Input ports : 8 • Output ports : 9 (+30: LCD segment pin)
Clock generator port (CGP)	• 1 VDP (Variable Duty Pulse) and SG (Signal Generator) functions
LCD controller/driver	• 30 segments, 2 common 1/2 duty, 1/2 bias, frame frequency 250 Hz, driving voltage V_{DD}, segment pin used also for key source: 16 ports All of the 30 ports can be used as output ports (4 ports, 4 ports, 6 ports, and 16 ports can be set independently)
Serial interface	• Two types (3 channels) 8-bit 3-wire system: 2 channels 8-bit 2-wire system: 1 channel
D/A converter	• 8 bits x 3 (PWM output and output resisting pressure 16 V Max.)
A/D converter	• 6 bits x 6 (consecutive comparison method by software)
Interrupt	• 5 channels (maskable interrupt) External interrupt : 2 channels (INT_0 pin and INT_1 pin) Internal interrupt : 3 channels (timer, serial interface 1, and frequency counter)
Timer	• Two types Timer carry FF (1, 5, 100, 250 ms) Timer interrupt (1, 5, 100, 250 ms)
Reset	• Power On Reset (at power supply connection) • Resetting by CE pin (CE pin Low → High) • Blackout detection function

Item		Function
PLL frequency synthesizer	Division method	• 2 types Direct division method (VCOL pin 20 MHz Max.) Pulse swallow method (VCOL pin 40 MHz Max.) (VCOH pin 150 MHz Max.)
	Reference frequency	• 12 types are selected by the program 1, 1.25, 2.5, 3, 5, 6.25, 9, 10, 12.5, 25, 50, 100 kHz
	Charge pump	• Two independent error output
	Phase comparator	• Unlocking can be detected by a program Unlocking FF delay time can be selected
	LPF amplifier	• CMOS operation amplifier output resisting pressure 16 V Max.
Frequency counter		• Frequency test $P1D_3$/FMIFC pin 5-15 MHz $P1D_2$/AMIFC pin 0.1-1 MHz • External gate width test $P1A_1$/FCG pin
Power supply voltage		5 V ±10 %
Package		80-pin plastic QFP (14 x 20)


Program memory (ROM)
Instruction
All the instructions are 16-bit one-word instructions
Constant data
Can be transferred to data buffer (table reference)
Peripheral hardware
PLL
Frequency counter
A/D converter
D/A converter
Serial interface
Clock generator
LCD controller/ driver
Data memory (RAM)
Data buffer
Setting peripheral hardware data
General register
Performs operation and transfer between data memory and other data memory
Control section
Program counter
Data memory address
Stack
Stack operation enabled
Operation
Decimal operation enabled
Resetting
Timer
Interrupt
System register
Controls CPU peripheral circuits
Window register
LCD dot register
Display data is set
General output port register
Input and output data is set
General purpose port
Input port
Output port
Input/ output port
Register file
Set peripheral hardware conditions
Set data via a window register

PIN CONFIGURATION (Top View)



CONTENTS

1. PIN FUNCTIONS

1.1 EXPLANATION ON EACH PIN FUNCTION

Pin No.	Pin symbol	Input/ Output	Output mode	Pin name	Function
79 80 1 2	$P0C_3$ $P0C_2$ $P0C_1$ $P0C_0$	Input/ output	CMOS Push-Pull	Port 0C	4-bit general purpose output port. Can be specified as an input or output port in 4-bit units (group I/O). Input/output is specified by the P0CGPIO register (address 27H) of a register file. The P0C register (address 27H of BANK0) of the port register is used for reading input data and setting output data. At Power On Reset, Clock Stop instruction execution, or CE Reset, these pins are specified as input ports.
3 4 5 6	$P0A_3$/SDA $P0A_2$/SCL $P0A_1/\overline{SCK_1}$ $P0A_0/SO_1$	Input/ Output	N-ch open drain CMOS Push-Pull	Port 0A	Used as a 4-bit general purpose input/output port and also for serial interface. A general purpose input/output port and serial interface is switched by the SIO1MODE register (address 08H) and SIO2MODE register (address 02H) of the SIO1MODE register of the register file. (1) **When the pin is used as a 4-bit general purpose** input/output port The port can be specified as an input or output port in bit units (bit I/O). Input or output is specified by the P0ABIO register (address 35H) of the register file. The P0A register (address 70H of BANK0) is used for reading input data and output data and setting the port register. Since $P0A_3$/SDA, and $P0A_2$/SCL pins are N-ch open drain output, pull-up resistance is required in the external section. (2) **When the pins are used for serial interface** Two types of serial interfaces are available, serial interface 1 and serial interface 2 including Port 0B (pin numbers 7 to 10). Serial interface 1 and serial interface 2 can be used concurrently. Two channels of a 2-wire system and 3-wire system can be used for serial interface 1 and one channel of a 3-wire system can be used for serial interface 2. When using serial interface 1, specify pins from the SIO1MODE register of the register file and when using serial interface 2, specify pins using SIO2MODE register. The function of each pin is listed below.

<table>
<tr><th>Pin No.</th><th>Pin symbol</th><th>Input/ Output</th><th>Output mode</th><th>Pin name</th><th>Function</th></tr>
<tr><td>3
4
5
6</td><td>P0A$_3$/SDA
P0A$_2$/SCL
P0A$_1$/$\overline{SCK_1}$
P0A$_0$/SO$_1$</td><td>Input
Output</td><td>N-ch open drain
COMS Push-Pull</td><td>Port 0A</td><td>
<table>
<tr><th>Pin name</th><th>Function</th><th colspan="2">Operating mode</th></tr>
<tr><td>P0A$_3$/SDA</td><td>Data input/ output</td><td rowspan="2">2-wire</td><td rowspan="5">Serial interface 1</td></tr>
<tr><td>P0A$_2$/SCL</td><td>Clock input/ output</td></tr>
<tr><td>P0A$_1$/$\overline{SCK_1}$</td><td>Clock input/ output</td><td rowspan="3">3-wire</td></tr>
<tr><td>P0A$_0$/SO$_1$</td><td>Data output</td></tr>
<tr><td>P0B$_3$/SI$_1$</td><td>Data input</td></tr>
<tr><td>P0B$_2$/$\overline{SCK_2}$</td><td>Clock input/ output</td><td rowspan="3">3-wire</td><td rowspan="3">Serial interface 2</td></tr>
<tr><td>P0B$_1$/SO$_2$</td><td>Data output</td></tr>
<tr><td>P0B$_0$/SI$_2$</td><td>Data input</td></tr>
</table>
Since pins P0A$_3$/SDA and P0A$_2$/SCL are N-ch open drain, Pull-Up resistance is required externally.
At Power On Reset, Clock Stop instruction execution, and CE Reset, all of these pins are specified as input ports of general purpose input/output ports.
</td></tr>
<tr><td>7
8
9
10</td><td>P0B$_3$/SI$_1$
P0B$_2$/$\overline{SCK_2}$
P0B$_1$/SO$_2$
P0B$_0$/SI$_2$</td><td>Input
Output</td><td>CMOS Push-pull</td><td>Port 0B</td><td>Used for 4-bit general purpose input/output ports and also for serial interface
The SIO1MODE register (address 08H) or SIO2MODE register (address 02H) of the register file are used for switching the function as general purpose input/output port to serial interface or vice versa.
(1) When using the pins as 4-bit general purpose input/output ports
The pins can be specified as input or output ports in bit units (bit I/O).
Input or output is specified by the P0BBIO register (address 35H) of the register file.
The P0B register (address 71H of BANK0) of the port register is used for reading input data and setting output data.</td></tr>
</table>

Pin No.	Pin symbol	Input/ Output	Output mode	Pin name	Function
7 8 9 10	$P0B_3/SI_1$ $P0B_2/\overline{SCK_2}$ $P0B_1/SO_2$ $P0B_0/SI_2$	Input Output	CMOS Push-pull	Port 0B	(2) **When the pins are used for serial interface** Two types of serial interface can be used including Port 0A (addresses 3 to 6), serial interface 1 and serial interface 2. See the explanation on Port 0A for the function of each pin. At Power On Reset, Clock Stop instruction execution, and CE Reset, all of these pins are specified as input ports of general purpose input/output ports.
11 12	INT_1 INT_0	Input	—	Interrupt	External interrupt request input pin. An interrupt request is issued from the input signal rising edge or falling edge of the input signal added to the pin. A rising edge and a falling edge can be specified by the INTEDGE register (address 1FH) of the register file using INT_0 pin and INT_1 pin independently. Even if an interrupt request is issued, interrupt cannot be accepted unless it is permitted (maskable interrupt). Types of interrupt permission include permission of all the interrupts by the EI instruction and permission of the interrupt of each INT_0 pin and INT_1 pin. Permission of interrupt for each pin is specified by the INTPM2 register (address 2FH) of the register file. When interrupt is permitted and when an interrupt request is issued, the interrupt is accepted. When interrupt is accepted. When interrupt is accepted, control of the program is passed to address 0005H in the case of interrupt by the INT_0 pin and address 0004H in the case of interrupt by the INT_1 pin. When interrupts for both INT_0 pin and INT_1 pin are allowed and when interrupts for both pins are issued, priority is given to the interrupt by INT_0 pin. Even if an interrupt is not permitted, the issuing of an interrupt request can be checked using the INTREQ2 register (address 3FH) of the register file. When an interrupt function is not used, the input level of each pin can be detected by the INTJDG register (address 0FH) of the register file, and the pin can be used as a general purpose input port. At Power On Reset, Clock Stop Instruction execution, or CE Reset, the interrupt permission and interrupt requests are reset.

Pin No.	Pin symbol	Input/ Output	Output mode	Pin name	Function
13	CE	Input	–	Chip Enable	Input pins for device operation selection signal and reset signal. Device operation selection is to select the operation of the PLL frequency synthesizer and standby status as described below. (1) **Device operation selection** When the CE pin is at a High level, the PLL frequency synthesizer section can be operated. When the CE pin is at a Low level, the PLL frequency synthesizer section sets to a Disable state (operation prohibited) automatically in the device internal section. When the CE pin is at a Low level, the operation of quartz oscillation circuits in the internal section and CPU can be stopped by executing a Clock Stop instruction and data memory can be kept under a low consumption current (15 μA or less) (at CE pin = High level, the Clock Stop instruction operates as the NOP instruction). At execution of a Clock Stop instruction, the LCD controller/driver is set to a Display Off mode (LCD_0-LCD_{29}, COM_0, COM_1 pin are Low level output) and general purpose input-output ports (Port 0A, Port 0B, Port 0C, and Port 1A) are used as input ports. (2) **Reset signal input** When the CE pin is changed from a Low level to High level, the device is reset by synchronizing with the Timer Carry FF of the internal section (CE Reset). When the device is reset, the program starts from address 0. In this case, the general purpose input/output ports are used as input ports. Since four types of internal Timer Carry FF, 1, 5, 100, and 250 ms can be selected, the time elapsing from when the pin is changed from the Low level to High level until the device is reset can be selected. However, if a Clock Stop instruction has been executed, the device is reset about 100 ms after the CE pin is changed to a High level. This pin does not accept a Low level or High level of less than 110-165 μs to prevent operation error due to noise.

Pin No.	Pin symbol	Input/ Output	Output mode	Pin name	Function
13	CE	Input	—	Chip Enable	By using the CEJDG register (address 07H) of the register file, the input signal level of this pin can be detected. In this case also, the contents of the CEJDG register do not change at a Low level or High level of less than 110-165 μs. Shumit Trigger input with hysterisis feature is used for this pin. Note that a voltage higher than that of V_{DD} pin must not be supplied at power connection.
14 15 16 17	$P1A_3$ $P1A_2$ $P1A_1$ $P1A_0$/FOG	Input Output	CMOS Push-Pull	Port 1A	Used as a 4-bit general purpose input/output port and also as an external gate counter ($P1A_0$/FCG pin). The switching between the general purpose input/output port and an external gate counter is performed by the IFCMODE register (address 12H) of the register file. (1) **When the port is used as a 4-bit general purpose input/output port** The port can be specified as an input or output port in bit units (bit I/O). Input or output is specified by the P1ABIO register (address 35H) of the register file. The P1A register (address 70H of BANK1) of the port register is used for reading input data and setting output data. (2) **When the port is used as an external gate counter (FCG) ($P1A_0$/FCG pin)** The counter counts the time from one rising edge to the next rising edge of the signal sent to the $P1A_0$/FCG pin. A reference frequency (1 kHz, 100 kHz, 900 kHz) of the internal section is counted by a 16-bit counter. The external gate counter is specified by the IFCMODE register (address 12H) and IFCCONT register (address 23H) of the register file. The PIA_0/FCG pin must be specified as the input port by the P1ABI0 register (address 35H). Since the IFCMODE register and IFCCONT register control the frequency counter ($P1D_3$/FMIFC and $P1D_2$/AMIFC pins) and a clock generator port ($P1B_0$/CGP pin) also, an external gate counter, frequency counter, and a lock generator port cannot be used concurrently. At Power On Reset, execution of a Clock Stop instruction, and CE Reset, all of these pins are specified for input ports of the general input/output ports.

<table>
<tr><th>Pin No.</th><th>Pin symbol</th><th>Input/ Output</th><th>Output mode</th><th>Pin name</th><th>Function</th></tr>
<tr><td>18
19
20
21</td><td>P1B$_3$/PWM$_2$
P1B$_2$/PWM$_1$
P1B$_1$/PWM$_0$
P1B$_0$/CGP</td><td>Output</td><td>N-ch open drain
CMOS Push-Pull</td><td>Port 1B</td><td>
Used as a 4-bit general output port, D/A converter (P1B$_2$/PWM$_2$, P1B$_2$/PWM$_1$, P1B$_1$/PWN$_0$ pins), and a clock generator port (P1B$_0$/CGP pin).
The PWMMODE register (address 13H) of the register file is used for switching the general output port, D/A converter and a clock generator port.
(1) When the port is used as a 4-bit general purpose output port
The P1B register (address 71H of BANK1) of the port register is used for setting output data.
The pins PIB$_3$/PWM$_2$, PIB$_2$/PWM$_1$, and PIB$_0$/PWM$_0$ require Pull-Up resistance for N-ch open drain output.
(Resisting pressure 16 V Max.)
(2) When the port is used as a D/A converter (PMW output) (pins P1B$_3$/PWM$_2$, P1B$_2$/PWM$_1$, and P1B$_1$/PWM$_0$)
Each of pins P1B$_3$/PWM$_2$, P1B$_2$/PWM$_1$, and P1B$_1$/PWM$_0$ can output an independent signal. A pulse width modulation (PWM) method is used as the output method, the frequency is 878.9 Hz (225 kHz/256) and duty is 0.25/256-255.25/256. (256 stages)
The duty must be set below the PWMRO-PWMR2 registers (addresses from 05H to 07H) via a data buffer.
<table>
<tr><th>Function</th><th>Frequency</th><th>Duty</th></tr>
<tr><td>D/A converter</td><td>878.9 Hz</td><td>$\frac{0.25+X}{256}$ x 100 %
X = 0 – 255</td></tr>
</table>
These three pins are N-ch open drain output and the resisting pressure is 16 V Max.
(3) When the port is used as a clock generator port (CGP) (P1B$_0$/CGP pin)
The P1B$_0$/CGP pin is set to a CGP mode by the PWMMODE register (address 13H) and IFCMODE register (address 12H) of the register file. Two functions are available for a CGP mode, VDP (Variable Duty Pulse) and SG (Signal Generator). The VDP function produces output in 64 stages, duty 2/67-65/67 at frequency 269 Hz.
</td></tr>
</table>

<table>
<tr><th>Pin No.</th><th>Pin symbol</th><th>Input/ Output</th><th>Output mode</th><th>Pin name</th><th>Function</th></tr>
<tr><td>18
19
20
21</td><td>P1B$_3$/PWM$_2$
P1B$_2$/PWM$_1$
P1B$_1$/PWM$_0$
P1B$_0$/CGP</td><td>Output</td><td>N-ch open drain
CMOS Push-Pull</td><td>Port 1B</td><td>The SG function produces output by dividing with the value of 4-130 (64 stages) using frequency 18 kHz as the reference frequency.
Both the VDP and SG functions set data as follows using the CGPR register (address 20H) via a data buffer.
<table><tr><th>Function</th><th>Frequency</th><th>Duty</th></tr><tr><td>VDP</td><td>269 Hz</td><td>$\frac{2+X}{67}$ x 100 %
X = 0 – 63</td></tr><tr><td>SG</td><td>$\frac{18}{2(2+X)}$ kHz
X = 0 – 63</td><td>50 %</td></tr></table>At Power On Reset or execution of Clock Stop instruction, these pins are specified as general purpose output ports.
At Power On Reset, undefined data is output.
At execution of a Clock Stop instruction, the value of the general purpose output port is retained. At CE reset, the statuses (general purpose output port, A/D converter, CGP) which are set at that time are retained.</td></tr>
<tr><td>22
23
24
24</td><td>P1C$_3$
P1C$_2$
P1C$_1$
P1C$_0$</td><td>Output</td><td>CMOS Push-Pull</td><td>Port 1C</td><td>4-bit general purpose output port. Output data is set via the P1C register (address 72H of BANK1) of the port register.
At Power On Reset, undefined data is output.
At execution of a Clock Stop instruction or CE reset, the value which was output previously is kept.</td></tr>
<tr><td>26
27
28
29</td><td>P1D$_3$/FMIFC
P1D$_2$/AMIFC
P1D$_1$/ADC$_1$
P1D$_0$/ADC$_0$</td><td>Input</td><td>—</td><td>Port 1D</td><td>Used as a 4-bit general purpose input port, frequency counter (pins P1D$_3$/FMIFC and P1D$_2$/AMIFC), and also A/D converter (pins P1D$_1$/ADC$_1$ and P1D$_0$/ADC$_0$).
The IFCMODE register (address 12H) of the register file is used for switching the general purpose input port and A/D converter.
The ADCCH register (address 14H) of the register file is used for switching the general input port and A/D converter.
(1) When the port is used as a 4-bit general purpose input port
The P1D register (address 73H of BANK1) of the port register is used for reading input data.</td></tr>
</table>

Pin No.	Pin symbol	Input/ output	Output mode	Pin name	Input pin Function
26 27 28 29	$P1D_3$/FMIFC $P1D_2$/AMIFC $P1D_1$/ADC_1 $P1D_0$/ADC_0	Input	–	Port 1D	(2) **When the port is used as a frequency counter ($P1D_3$/FMIFC and $P1D_2$/AMIFC)** Using the IFCMODE register of the register file, pins $P1D_3$/FMIFC and $P1D_2$/AMIFC can be used as frequency test pins. The following frequencies can be tested.

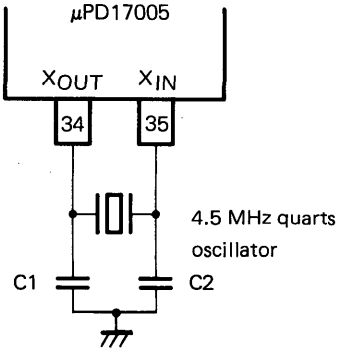
Input pin	Input frequency	Input oscillation
$P1D_3$/FMIFC	5-15 MHz	0.3 V_{p-p}
	10.5-10.9 MHz	0.06 V_{p-p}
$P1D_2$/AMIFC	0.1-1 MHz	0.3 V_{p-p}
	0.44-0.46 MHz	0.05 V_{p-p}

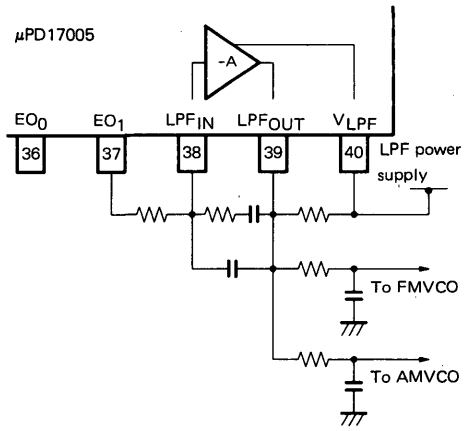
As the test method, the frequency input within the gate time (1 ms, 4 ms, 8 ms, open) is counted by a 16-bit counter. However, the value divided by 2 is counted for the $P1D_3$/FMIFC pin.
At termination of the test (when the gate is closed), an interrupt request can be issued.
These functions can be used at detection of broadcast station by counting the intermediate frequency. When the port is used as a frequency counter, cut the direct current section of the input signal with a condenser because an alternate current amplifier is used for input. The pin which was selected is used as an intermediate electric potential (about 1/2 V_{DD}). Pins which are not selected can be used as a general purpose input port.
The alternate current amplifier must be initialized by a program as required because it is not set to Disabled (prohibited state) even if the CE pin (pin number 13) is set to a Low level (if the amplifier is operating, the current consumed may increase the noise factor).
Since the IFCMODE register also specifies an external gate counter ($P1A_0$/FCG pin) and clock generator port ($P1B_0$/CGP pin), the frequency counter, external gate counter, and clock generator port cannot be used concurrently.

Pin No.	Pin Symbol	Input/ output	Output mode	Pin name	Function
26 27 28 29	$P1D_3$/FMIFC $P1D_2$/AMIFC $P1D_1/ADC_1$ $P1D_0/ADC_0$	Input	–	Port 1D	(3) **When the port is used as an A/D converter (pins $P1D_1/ADC_1$ and $P1D_0/ADC_0$)** The port can be used as an A/D converter of 6 bits by the ADCCH register (address 14H) of the register file. The A/D converter can use six channels by switching pins POD_3/ADC_5 to POD_0/ADC_2 (pin numbers from 75 to 78) in addition to pins $P1D_1/ACD_1$ and $P1D_0/ACD_0$. A consecutive comparison type is used as the conversion method and the reference voltage is created by dividing power supply voltage V_{DD} using the R string method. At Power On Reset or execution of a Clock Stop instruction, all these pins are specified as a general purpose input port. At CE reset, the statuses (general purpose input port, frequency counter, and A/D converter) set at that time are retained.
30 41	V_{DD1} V_{DD2}	–	–	Power supply	Device power supply pin Voltage of 5 V ± 10 % is supplied at operation of CPU and peripheral functions. When only CPU is operating the voltage can be reduced to 3.5 V. When the CE pin (pin number 13) is at a Low level and when a Clock Stop instruction is executed, oscillation of the quartz oscillator stops and a data set backup state is set. During the clock stop state, the voltage can be reduced to 2.2 V. When the voltage rises from 0 V to 4.5 V or when the voltage rises to 4.5 V again after decreasing to a degree less than 3.5 V (less than 2.2 V at clock stop), Power On Reset is performed for the device. When Power On Reset is performed, the peripheral circuits, system registers, and register files are initialized and the program starts from address 0. The time spent from the voltage 0 V to 4.5 V must be within 500 ms. Resetting by a CE pin (CE Pin Reset) is also available in addition to Power On Reset described above for resetting a device. Since the values of timer carry FF if the register file differs between Power On Reset and CE Reset, blackout can be detected by detecting the timer carry FF. A voltage higher than that of the V_{DD} pin must not be supplied to all the pins other than V_{DD} pins (V_{DD1} and V_{DD2}). In particular, care is necessary when the V_{DD} pin and the CE pin are started simultaneously.

<table>
<tr><th>Pin No.</th><th>Pin Symbol</th><th>Input/ output</th><th>Output mode</th><th>Pin name</th><th>Function</th></tr>
<tr><td>30
41</td><td>V_{DD1}
V_{DD2}</td><td>–</td><td>–</td><td>Power supply</td><td>Latch-Up may occur. The V_{DD1} pin and V_{DD2} pin must be connected to an electrical potential.
The V_{DD2} pin is used to supply power to quartz oscillation circuits (pins X_{IN} and X_{OUT}), error out circuits (pins EO_0 and EO_1), and low path filter circuits (pin LPF_{IN}). Pin V_{DD1} is used for supplying power to other sections.</td></tr>
<tr><td>31
32</td><td>VCOL
VCOH</td><td>Input</td><td>–</td><td>Local oscillation Low input

Local oscillation High input</td><td>Used for inputting local oscillation (VCO) frequency of PLL.
A direct division method (MF mode) and pulse swallow method (HF mode and VHF mode) are available as division methods and the method is specified by the PLLMODE register (address 21H) of the register file. The input pin, input frequency and division ratio by each division method are as follows.
<table>
<tr><th>Division method</th><th>Input pin</th><th>Input frequency (MHz)</th><th>Input voltage (V_{p-p})</th><th>Division ratio</th></tr>
<tr><td>Direct division</td><td>VCOL</td><td>0.5–30</td><td>0.3</td><td>16–$2^{16}-1$</td></tr>
<tr><td>Pulse swallow (HF)</td><td>VCOL</td><td>5–40</td><td>0.3</td><td>256–$2^{16}-1$</td></tr>
<tr><td>Pulse swallow (VHF)</td><td>VCOH</td><td>9–150</td><td>0.3</td><td>256–$2^{16}-1$</td></tr>
</table>
Since alternate current amplifier is used for input of these pins, the direct current section of the input signal must be cut using a condenser.
The pin specified by the PLLMODE register is used as an intermediate electrical potential (about 1/2 V_{DD}). Pins which are not specified are pulled down in the internal section of the device.
When PLL is disabled or when the CE pin is at a Low level, these pins are pulled down in the internal section of the device. At Power On Reset or execution of a Clock Stop instruction, a PLL Disabled state is set. At CE reset, the state specified by the PLLMODE register is set.</td></tr>
<tr><td>33</td><td>GND</td><td>–</td><td>–</td><td>Ground</td><td>Ground pin of the device</td></tr>
</table>

Pin No.	Pin symbol	Input/ output	Output mode	Pin name	Function
34 35	X_{OUT} X_{IN}	Output Input	CMOS —	Quartz oscillator	Quartz oscillator connection pin Connects a 4.5 MHz quarts oscillator as shown below.  The values of C1 and C2 are determined by the quartz oscillator which is used. When the values of C1 and C2 are increased to values which are too high, the oscillation activation feature may deteriorate or current consumption may increase. In general, the adjustment range of a trimmer condenser for oscillation frequency adjustment increases when the oscillator is connected to the X_{IN} pin. However, the quartz oscillator which is actually used, including oscillation stabilizer, must be used for evaluation. An oscillation frequency cannot be adjusted accurately because of the problem at capacity, etc., if a probe is connected to the X_{OUT} pin or X_{IN} pin. Consequently, the frequency must be tested while testing the LCD driving wave form (125 Hz) or VCO oscillation frequency. Since the reference frequency of the timer of the internal section or PLL is used by dividing 4.5 MHz, if the value is shifted from 4.5 MHz, the values of the timer and reference frequency also shift in the same proportion.
36 37	EO_0 EO_1	Output	CMOS 3 states	Error out	Used as charge pump output pins of a PLL frequency synthesizer. When the value producing by dividing the local oscillation (VCO) frequency which is input to the VCOL pin (pin number 31) or VCOH pin (pin number 32) is higher than the reference frequency, a High level is output from these pins and when the value is lower than the reference frequency, a Low level is output. When the values match, floating occurs.

Pin No.	Pin symbol	Input/ output	Output mode	Pin name	Function
36 37	EO_0 EO_1	Output	CMOS 3 states	Error out	A PLL frequency synthesizer can be structured by adding output of these pins to VCO (Voltage Controlled Oscillator) via LPF (Low Pass Filter). Either of the pins EO_1 and EO_2 can be used because the same signal is output. At a PLL Disabled state, these pins are set floating. That is, when the CE pin (pin number 13) is at a Low level or at Power On Reset, floating occurs. The PLL frequency synthesizer can detect a PLL unlocked state by the PLLULJDG register (address 05H) of the register file. Four types of time (0.5 μs, 1 μs, 2 μs, and Disable) can be selected as the delay time for detecting the PLL unlocated state using the PLULDLY register (address 15H) of the register file.
38 39 40	LPF_{IN} LPF_{OUT} V_{LPF}	Input Output —	— N-ch open drain —	LPF amplifier	Pins for a built-in CMOS operation amplifier for LPF (Low Pass Filter). Examples of an internal equivalent circuit of each pin and application of circuit are shown below.  Pull-Up resistance is required for the LPF_{OUT} pin because of N-ch open drain output. The resisting pressure is 16 V Max. A voltage higher than that of the LPF_{OUT} pin must be supplied to the V_{LPF} pin (16 V Max). At a PLL Disabled State, the LPF_{IN} pin is pulled up in the device internal section.

Pin No.	Pin symbol	Input/ output	Output mode	Pin name	Function
42	$P2A_0$	Output	CMOS Push-Pull	Port 2A	1-bit general purpose output port. Output data is set via the P2A register (address 70H of BANK2) of the port register. At Power On Reset, undefined data is output. At execution of a Clock Stop instruction or CE reset, the value which was output previously is kept.
43 44	COM_1 COM_0	Output	CMOS 3-value output	Common signal	Common signal output pins of the LCD controller/driver. The duty, bias, frame frequency, and driving voltage of the LCD controller/driver are 1/2, 1/2, 250 Hz, and V_{DD} respectively. Display of up to 60 dots can be performed by the matrix with pins $LCD_0/P0Y_0/KS_0$-$LCD_{29}/P0F_3$. Three types of voltages, 0, 1/2 V_{DD}, and V_{DD} are output from these pins. The light of the dot from which a potential difference of $\pm V_{DD}$ is produced between these pins and pins $LCD_0/P0Y_0/KS_0$-$LCD_{29}/P0F_3$ comes on. When a Display Off mode is set by the LCDMODE register (address 10H of the LCDMODE register of the register file), a Low level is output at Power On Reset or execution of a Clock Stop instruction. At CE Reset, the state is kept if the mode is a Display On mode.
45 | 48 49 | 52 53 | 58 59 | 74	$LCD_{29}/P0F_3$ | $LCD_{26}/P0F_0$ $LCD_{25}/P0E_3$ | $LCD_{22}/P0E_0$ $LCD_{21}/P0X_5$ | $LCD_{16}/P0X_0$ $LCD_{15}/P0Y_{15}/KS_{15}$ | $LCD_0/P0Y_0/KS_0$	Output	CMOS Push-Pull	LCD segment signal	Used for segment signal output (pins $LCD_{29}/P0F_3$-$LCD_0/P0Y_0/KS_0$) of a LCD controller/driver, key source signal output (pins $LCD_{15}/P0Y_{15}/KS_{15}$-$LCD_0/POY_0/KS_0$), and also as a general output port ($LCD_{29}/P0F_3$- $LCD_0/P0Y_0/KS_0$). The LCDMODE register (address 10H) and LCDPORT register (address 11H) are used for outputting segment signals and key source signals, and switching general purpose output ports. (1) **When the pins are used for segment signal output of a LCD controller/driver (pins $LCD_{29}/P0F_3$-$LCD_0/P0Y_0/KS_0$)** The duty, bias, and frame frequency (segment signal output 125 Hz) of a LCD controller/driver are 1/2, 1/2, and 250 Hz respectively. Display of up to 60 dots is enabled by using a matrix of these segment signal output pins, the COM_0 pin, and COM_1 pin (numbers 43 and 44). The light of the dot from which a potential difference of $\pm V_{DD}$ is produced between these segment signal output pins, and COM_0 and COM_1 pins comes on.

Pin No.	Pin symbol	Input/ output	Output mode	Pin name	Function
45 | 48	$LCD_{29}/P0F_3$ | $LCD_{26}/P0F_0$	Output	CMOS Push-Pull	LCD segment signal	Display data of an LCD controller/driver is set via LCD dot registers (addresses 60H to 6EH of BANK0). Data can also be set by LCD group registers (addresses 08H to 0FH) via a data buffer. A display On mode and Display Off mode of a LCD controller/ driver is set by the LCDMODE register of the register file. In Display Off mode, these segment signal output pins output a Low level. However, for pins which are specified for a general purpose output port by the LCDPORT register of the register file, data of the output port is output regardless of the display mode, On or Off. Sixteen pins from $LCD_{15}/P0Y_{15}/KS_{15}$ to $LCD_0/P0Y_0/KS_0$ are also used for key source signal output of a key matrix as described in (2) and an LCD segment signal and a key source signal can be output concurrently.
49 | 52	$LCD_{25}/P0E_3$ | $LCD_{22}/P0E_0$				
53 | 58	$LCD_{21}/P0X_5$ | $LCD_{16}/P0X_0$				
59 | 74	$LCD_{15}/P0Y_{15}/KS_{15}$ | $LCD_0/P0Y_0/KS_0$				(2) **When the pins are used as a key source signal of a key matrix (pins $LCD_{15}/P0Y_{15}/KS_{15}$-$LCD_0/P0Y_0/KS_0$)** Using the LCDMODE register of the register file, sixteen pins from $LCD_{15}/P0Y_{15}/KS_{15}$ to $LCD_0/P0Y_0/KS_0$ can be used as a key source output signal. A key source signal is output with a LCD segment signal in time sharing mode (key source signal output time 220 μs). When a key source signal is used, pins $P0D_3/ADC_5$ to $P0D_0/ADC_2$ (pin numbers 75 to 78) are used as the return signal input pins. Consequently, a key matrix of 16 key sources and 4 key input (up to 64) can be structured. A key source signal is output every 4 ms. Output data of a key source signal is set by the KSR register (address 42H) via a data buffer. When the LCD controller/ driver is in Display Off mode (segment signal output = Low level) and when these pins are specified for a general purpose output port, a key source signal is not output.

Pin No.	Pin symbol	Input/ output	Output mode	Pin name	Function
45 | 48 49 | 52 53 | 58 59 | 74	$LCD_{29}/P0F_3$ | $LCD_{26}/P0F_0$ $LCD_{25}/P0E_3$ | $LCD_{22}/P0E_0$ $LCD_{21}/P0X_5$ | $LCD_{16}/P0X_0$ $LCD_{15}/P0Y_{15}/KS_{15}$ | $LCD_0/P0Y_0/KS_0$	Output	CMOS Push-Pull	LCD segment signal	(3) **When the pins are used for a general purpose output port** Each pin can be specified for an output port as listed in the following table using the LCDPORT register (address 11H) of the register file.

Pin number	Pin name	Port name	Number bit
45 | 48	$LCD_{29}P0F_3$ | $LCD_{26}P0F_0$	Port 0F	4 bits
49 | 52	$LCD_{25}/P0E_3$ | $LCD_{22}/P0E_0$	Port 0E	4 bits
53 | 58	$LCD_{21}/P0X_5$ | $LCD_{16}/P0X_0$	Port 0X	6 bits
59 | 74	$LCD_{15}/P0Y_{15}/KS_{15}$ | $LCD_0/P0Y_0/KS_0$	Port 0Y	16 bits

Port 0F, Port 0E, Port 0X, and Port 0Y can be specified as general purpose output ports individually. Pins which are not specified for a general purpose output port can be used as LCD segment signal output pins.

Output data of each output port is set as listed below.

<table>
<tr><th>Pin No.</th><th>Pin symbol</th><th>Input/ output</th><th>Output mode</th><th>Pin name</th><th>Function</th></tr>
<tr>
<td>45
|
48

49
|
52

53
|
58

59
|
74</td>
<td>$LCD_{29}/P0F_3$
|
$LCD_{26}/P0F_0$

$LCD_{25}/P0E_3$
|
$LCD_{22}/P0E_0$

$LCD_{21}/P0X_5$
|
$LCD_{16}/P0X_0$

$LCD_{15}/P0Y_{15}/KS_{15}$
|
$LCD_0/P0Y_0/KS_0$</td>
<td>Output</td>
<td>CMOS Push-Pull</td>
<td>LCD sigment signal</td>
<td>
<table>
<tr><th>Port Name</th><th>Setting output data</th></tr>
<tr><td>Port 0F</td><td>• P0F register (Address 6DH of BANK0) Used also for the LCDD13 register of the LCD dot register</td></tr>
<tr><td>Port 0E</td><td>• P0E register (Address 6BH of BANK0) Used also for the LCDD11 register of the LCD dot register</td></tr>
<tr><td>Port 0X</td><td>• P0XH and P0XL registers (Addresses 69H and 68H of BANK0) Used also for the LCDD9 and LCDD8 registers of the LCD dot register
• Set by the P0X group register (0CH) via a data buffer</td></tr>
<tr><td>Port 0Y</td><td>• Set by a P0Y group register (42H) via a data buffer</td></tr>
</table>
At Power On Reset or execution of Clock Stop instruction, all of these pins are specified for segment signal output and set to a Display Off mode.
Consequently a Low level is output from all these pins.
At CE Reset, the statuses (segment signal output, key source signal output, and general purpose output port) which are set at that time are retained.
</td>
</tr>
<tr>
<td>75
76
77
78</td>
<td>$P0D_3/ADC_5$
$P0D_2/ADC_4$
$P0D_1/ADC_3$
$P0D_0/ADC_2$</td>
<td>Input</td>
<td>(Pull-Down Input with resistance)</td>
<td>Port 0D</td>
<td>Used for a 4-bit general purpose input port and also LCD segment key source signal return input, and also A/D converter input. The ADCCH register (address 14H) of the register file is used for switching the general purpose port and A/D converter.
The pins $P0D_3/ADC_5$ to $P0D_0/ADC_2$ contain pull-down resistance so that they can be used as key return signal input pins of a key matrix.
(1) When the pins are used for general purpose input ports
Input data is read via the P0D register (address 72H of BANK0) of the port register.</td>
</tr>
</table>

Pin No.	Pin symbol	Input/ output	Output mode	Pin name	Function
75 76 77 78	$P0D_3/ADC_5$ $P0D_2/ADC_4$ $P0D_1/ADC_3$ $P0D_0/ADC_2$	Input	(Pull-Down Input with resistance)	Port 0D	When pins are used for a general input port, the built-in pull down resistance is always set to ON. (2) **When the pins are used for key source signal return input of an LCD segment** When an LCD segment pin is used for key source, the built-in pull down resistance is set to ON only during output of a key source signal (220 μs) and the resistance is set to OFF during output of an LCD segment signal. The signals which were input to these pins during output of key source signals are fetched as key input data. Consequently, these pins must be used when a LCD segment signal output is used as the key source signal. (3) **When pins are used as an A/D converter** By the ADCCH register (address 14H) of the register file, the port can be used as a 6-bit A/D converter. A consecutive comparison method by a program is used as the A/D converter conversion method and the reference voltage is created by dividing power supply voltage V_{DD} using the R string method. An A/D converter can be used by switching six channels, pins $P1D_1/ADC_1$ and $P1D_0/ADC_0$ (pin numbers 28 and 29) in addition to pins from $P0D_3/ADC_5$ to $P0D_0/ADC_2$. The channel used is specified by the ADCCH register of the register file. The other five channels which are not specified for the A/D converter can be used as a general purpose input port. For the built-in pull-down resistance, only the pin which was set is set to OFF when it is set to A/D converter input by the ADCCH register. At Power On Reset or execution of a Clock Stop instruction, the pins are specified for a general purpose input port. At CE Reset, the status (general purpose input port, LCD segment key source, return input, and A/D converter) which are set at that point are retained.

1.2 NOTES ON USING A GENERAL PURPOSE PORT

1.2.1 Port Register Data Set

The port registers (registers P0A to P2A) on data memory are used for reading input data or setting output data of each of the ports, Port 0A, Port 0B, Port 0C, Port 0D, Port 1A, Port 1B, Port 1C, Port 1D, and Port 2A.

In this case, the $P0A_3$ pin of Port 0A corresponds to the highest bit of port register P0A and the $P0A_0$ pin corresponds to the lowest bit.

These apply also to Port 0B, Port 0C, Port 0D, Port 1A, Port 1B, Port 1C, Port 1D, and Port 2A. Output data of Port 0E, Port 0F, Port 0X, and Port 0Y is set by the LCD group register via the LCD dot register or a data buffer on the data memory.

1.2.2 Input/output Ports (Port 0A, Port 0B, Port 0C, and Port 1A)

(1) When each port is specified as an input port

By executing an instruction (the address of the port register is specified for m of SKT m, #i, or ADD r, m) for reading the contents of each port register in the data memory, the status of each port pin is used as the value of the port register.

When an instruction (specified for r of MOV m, #i or ADD r, m) for writing data to each port register is executed, the value is written to the output data latch circuit.

(2) When each port is specified as an output port

When an instruction for writing data to each port register is executed, the value is written to the output data latch circuit and is output from each pin.

When an instruction for reading the contents of each port register is executed, the content of output data latch are used as the value of the port register. However, for pins $P0A_3$/SDA and $P0A_2$/SCL, the pin status is read as it is when the contents of the port register are read and the status may be different from the output data.

At Power On Reset, CE Reset, or execution of a Clock Stop instruction, all of these pins are set for input ports.

Since the contents of the output data latch circuit are undefined at Power On Reset, a Write instruction must be executed for the port register before setting data to the output port. Otherwise, undefined data is output. At CE Reset or execution of a Clock Stop instruction, the contents of the output data latch circuit do not change.

1.2.3 Output Ports (Port 1B, Port 1C, Port 0F, Port 0E, Port 0X, and Port 0Y)

An output port is used for writing the value of the port register to the output data latch circuit by executing an instruction for writing data in a port register and outputting data from each pin.

When a Read instruction is executed for a port register value, the port register value is set as the status of the output data latch circuit.

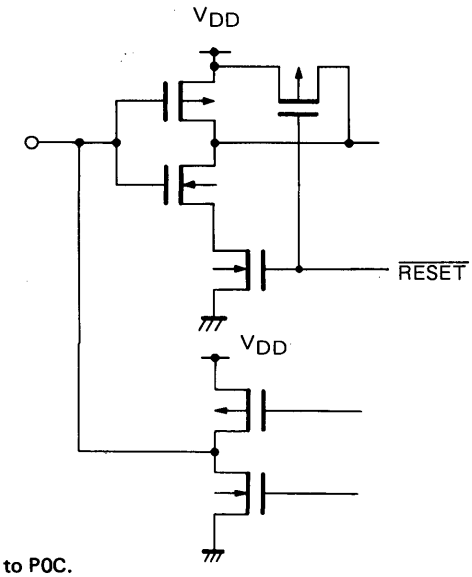
At Power On Reset, undefined data is output.

At CE Reset, the previous output data is kept at execution of a Clock Stop instruction. However, Port 0E, Port 0F, Port 0X, and Port 0Y output a Low level automatically at Power On Reset and at execution of a Clock Stop instruction.

1.3 PIN EQUIVALENT CIRCUITS

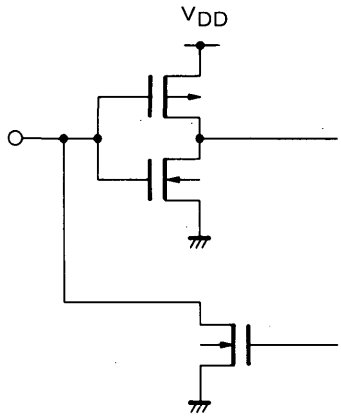
1.3.1 P0A ($P0A_1/\overline{SCK_1}$, $P0A_0/SO_1$)
P0B ($P0B_3/SI_1$, $P0B_2/SCK_2$, $P0B_1/SO_2$, $P0B_0/SI_2$)
P0C ($P0C_3$, $P0C_2$, $P0C_1$, $P0C_0$) (*1)
P1A ($P1A_3$, $P1A_2$, $P1A_1$, and $P1A_0$)

(Input/output)



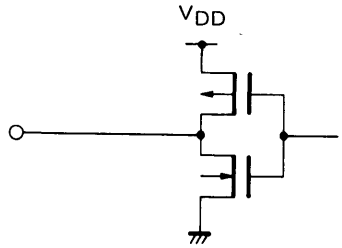
*1: The $\overline{RESET}$ signal is not provided to P0C.

1.3.2 P0A ($P0A_3$/SDA and $P0A_2$/SCL) (Input/output)

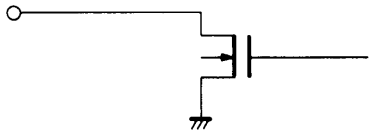


1.3.3 P1B ($P1B_0$/CGP)
P1C ($P1C_3$, $P1C_2$, $P1C_1$, and $P1C_0$)
P2A ($P2A_0$)
$LCD_0/P0Y_0/KS_0$ to $LCD_{29}/P0F_3$

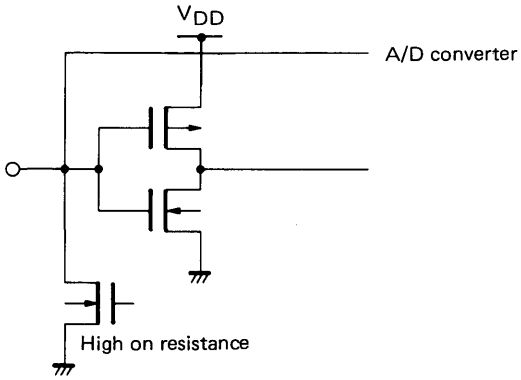
(Output)



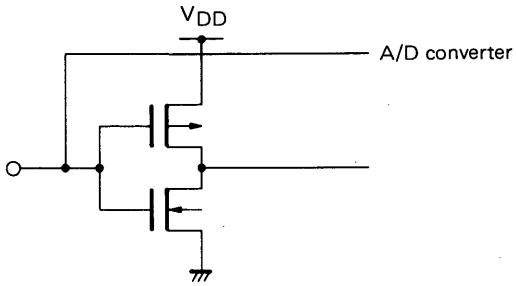
1.3.4 P1B ($P1B_3/PWM_2$, $P1B_2/PWM_1$, and $P1B_1/PWM_0$) (Output)



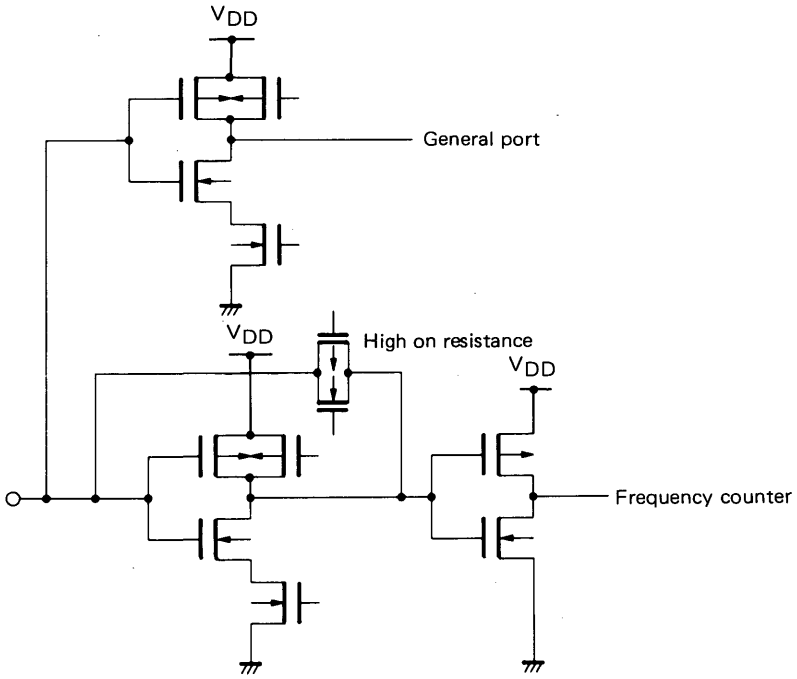
1.3.5 P0D ($P0D_3/ADC_5$, $P0D_2/ADC_4$, $P0D_1/ADC_3$, and $P0D_0/ADC_2$) (Input)



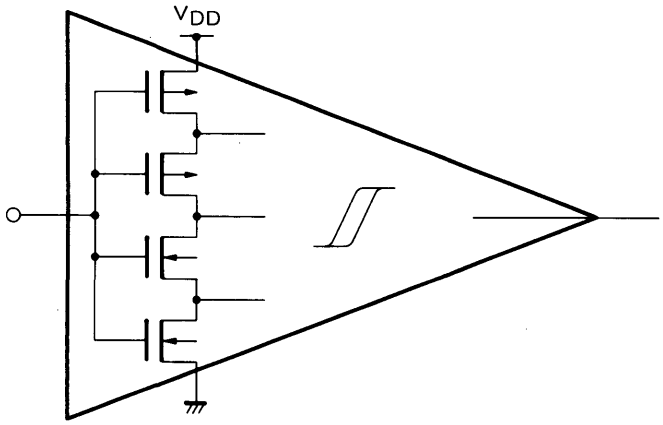
1.3.6 P1D ($P1D_1/ADC_1$ and $P1D_0/ADC_0$) (Input)



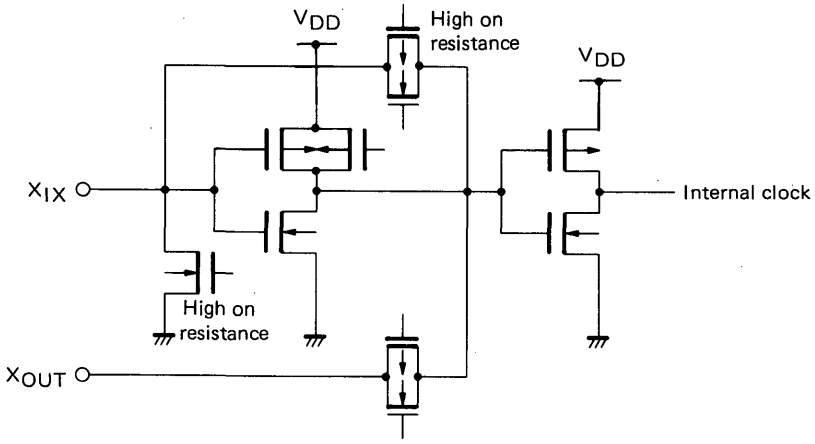
1.3.7 P1D ($P1D_3$/FMIFC, and $P1D_2$/AMIFC) (Input)



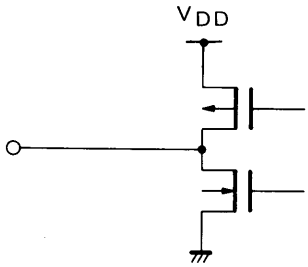
1.3.8 CE, INT_1, INT_0 (Schumit Trigger Input)



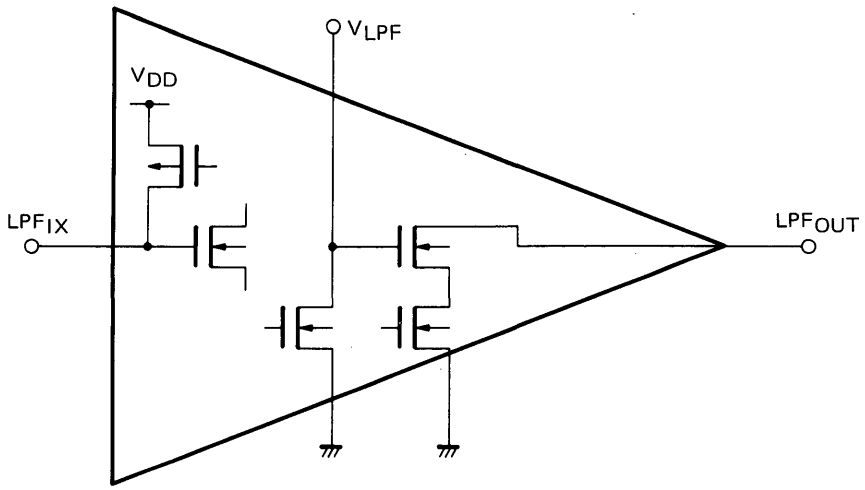
1.3.9 X_{OUT} (Output) and X_{IN} (Input)



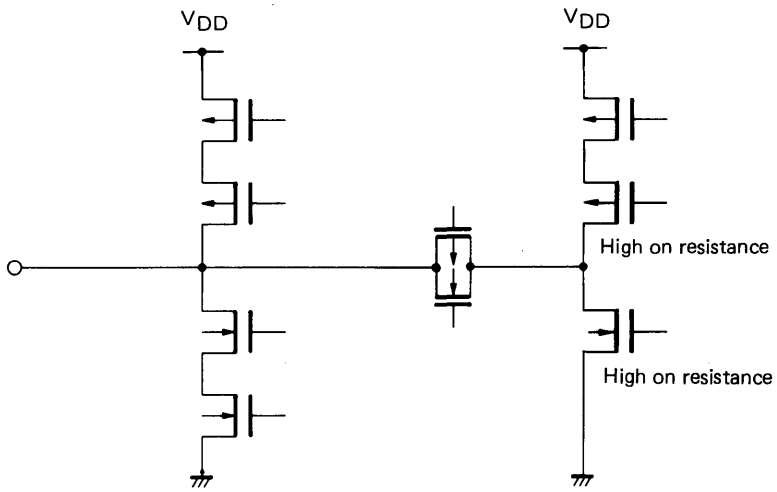
1.3.10 $E0_1$, $E0_0$ (Output)



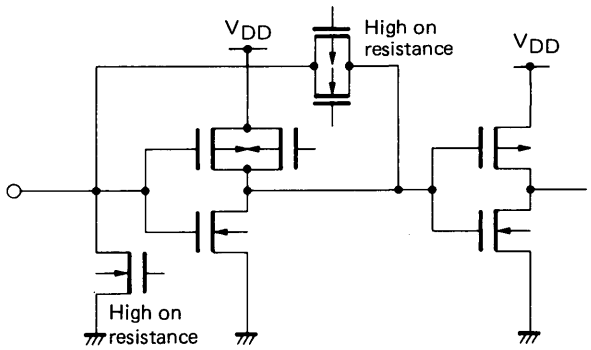
1.3.11 LPF_{IN} (Input), LPF_{OUT} (Output), and V_{LPF}



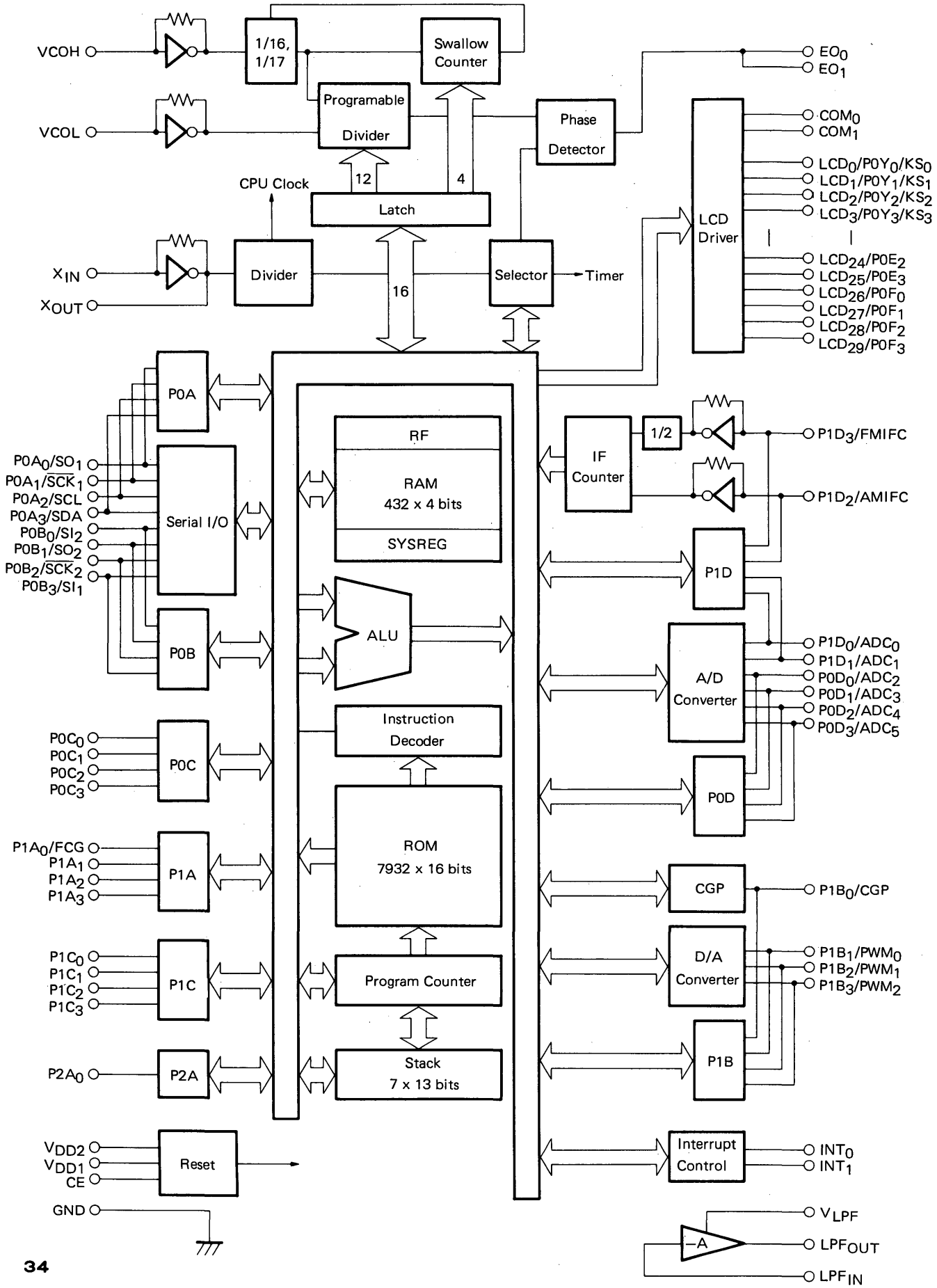
1.3.12 COM_1, COM_0 (Output)



1.3.13 VCOH, VCOL (Input)



2. BLOCK DIAGRAM



3. PROGRAM MEMORY (ROM)

Program memory contains "programs" executed by CPU and "constant data" which is predefined.

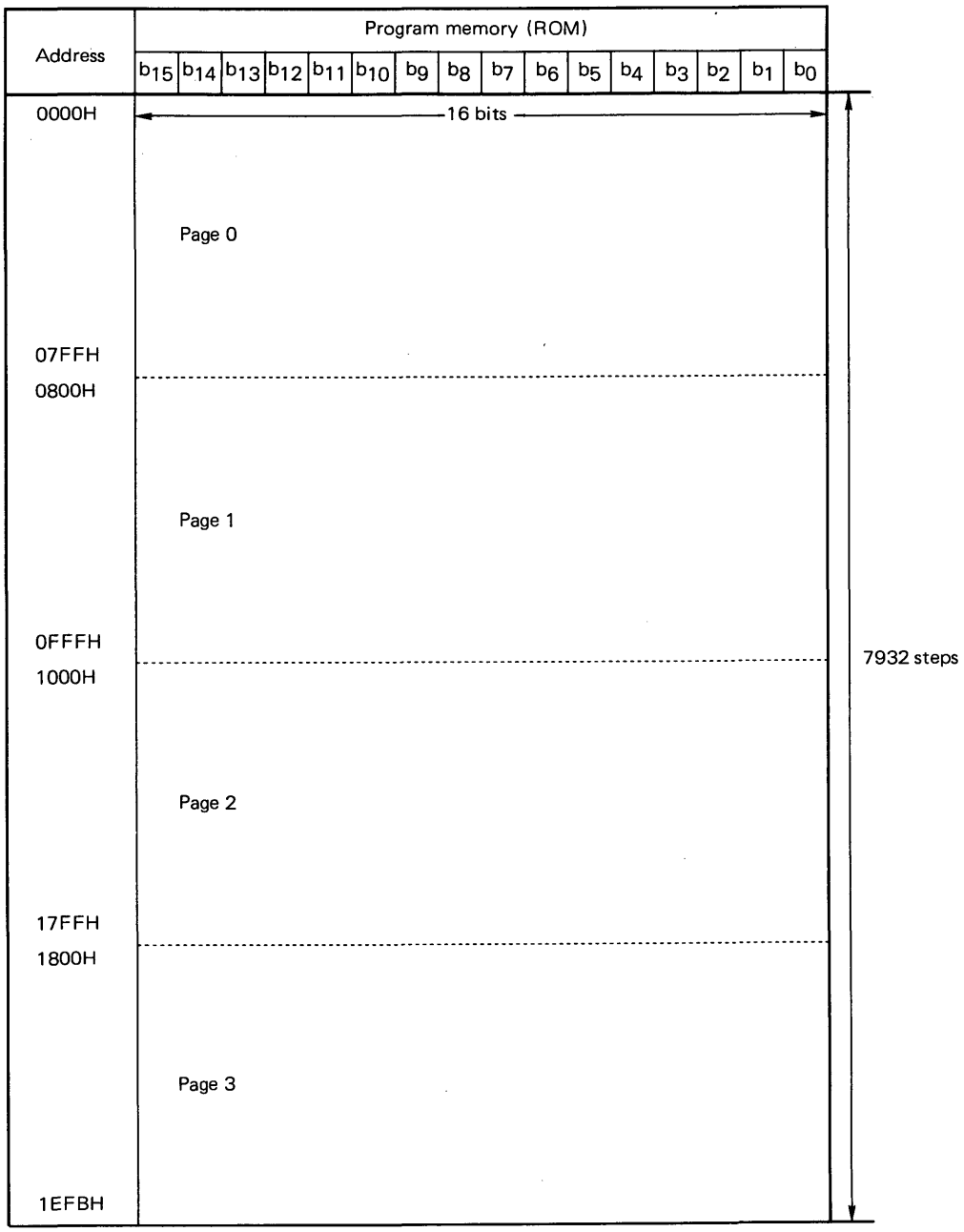
3.1 STRUCTURE OF PROGRAM MEMORY

Fig. 3-1 shows the structure of program memory.

As shown in Fig. 3-1, program memory consists of 7932 steps x 16 bits.

"Addresses" are assigned to program memory in 16-bit units and addresses comprise those from 0000H to 1EFBH. The memory space is divided into three pages: page 0 is from 0000H to 07FFH, page 1 from 0800H to 0FFFH, page 2 from 1000H to 17FFH, and page 3 from 1800H to 1EFBH.

Fig. 3-1 Structure of program memory



3.2 PROGRAM MEMORY FUNCTIONS

The following two major functions are provided by program memory.

(1) Storing programs.
(2) Storing constant data.

A program is a collection of "instructions" which operate CPU (Central Processing Unit: controls a micro controller) and CPU executes processing sequentially according to the "instructions" written in the program. That is, CPU reads "instructions" sequentially from the program stored in the program memory and executes processing according to each "instruction".

Since an "instruction" is a "one-word instruction" of 16-bit length, one instruction can be stored in each address of the program memory.

Constant data is predefined data such as display pattern. Constant data in the program memory can be read into a data buffer (DBF) of the data memory (RAM) by using the MOVT instruction, which is a dedicated instruction. The reading of constant data in the program memory is called "table referencing".

Since the program memory is Read Only Memory, the contents cannot be rewritten by an "instruction". Consequently, the program memory and ROM (Read Only Memory) are used synonymously.

3.3 PROGRAM FLOW

A program stored in the program memory is executed in address units from address 0000H. To execute a different program step according to the condition, the program flow must be branched. In this case, a branch instruction (BR) is used.

When the same program is to be executed repeatedly, the efficiency of the program memory deteriorates if many copies of the same program are used. In this case, by storing the program in one section and calling the program using the CALL instruction, the program can be executed repeatedly.

This program is called a "subroutine". Programs which are executed normally as opposed to subroutines are called "main routines".

To execute a program when a condition is satisfied regardless of the program flow, an interrupt function is used. When the condition is satisfied, the interrupt function can branch control to the predetermined address (called a vector address) regardless of the current program flow.

The program flow which was described above is controlled by a program counter (PC) which specifies the addresses of the program memory.

3.4 PROGRAM BRANCHING

A branch instruction (BR) is used for branching control to a program.

Fig. 3-2 shows the operation of a branch instruction.

Two types of branch instructions (BR) are available, direct branch instruction (BR addr) which directly branches control to the specified program memory address (addr), and indirect branch instruction (BR @AR) which branches control to the program memory address specified by the contents of the address register (AR).

See **4, "Program Counter (PC)"** also.

3.4.1 Direct Branching

A direct branch instruction specifies a program memory address of the branching destination using 13 bits; the two low-order bits of the operation code and 11 bits of the operation of the instruction. Consequently, the branching addresses of the direct branch instruction are all of the addresses of the program memory, 0000H to 1EFBH.

3.4.2 Indirect Branching

An indirect branch instruction specifies the address of the branching destination according to the 13-bit data of the address register. Consequently, the branching destination addresses of an indirect branch instruction are limited to those from 0000H to 1EFBH.

See Section **9.3, "Address Registers".**

3.4.3 Notes on Debugging

As shown in Fig. 3.2, the operation code used for a direct branch instruction differs depending on the page: page 0 (addresses 0000H to 07FFH), page 1 (addresses 0800H to 0FFFH), page 2 (address 1000H to 17FFH), and page 3 (address 1800H to 1EFBH).

The operation code for direct branch instruction to page 0 is "0CH" and the operation code to page 1 is "0DH". The operation code for page 2 is "0EH" and the operation code for page 3 is "0FH".

This is because the two low-order bits of the operation code is used as the branching destination address as there are only 11 bits of the operand "addr" of a direct branching instruction.

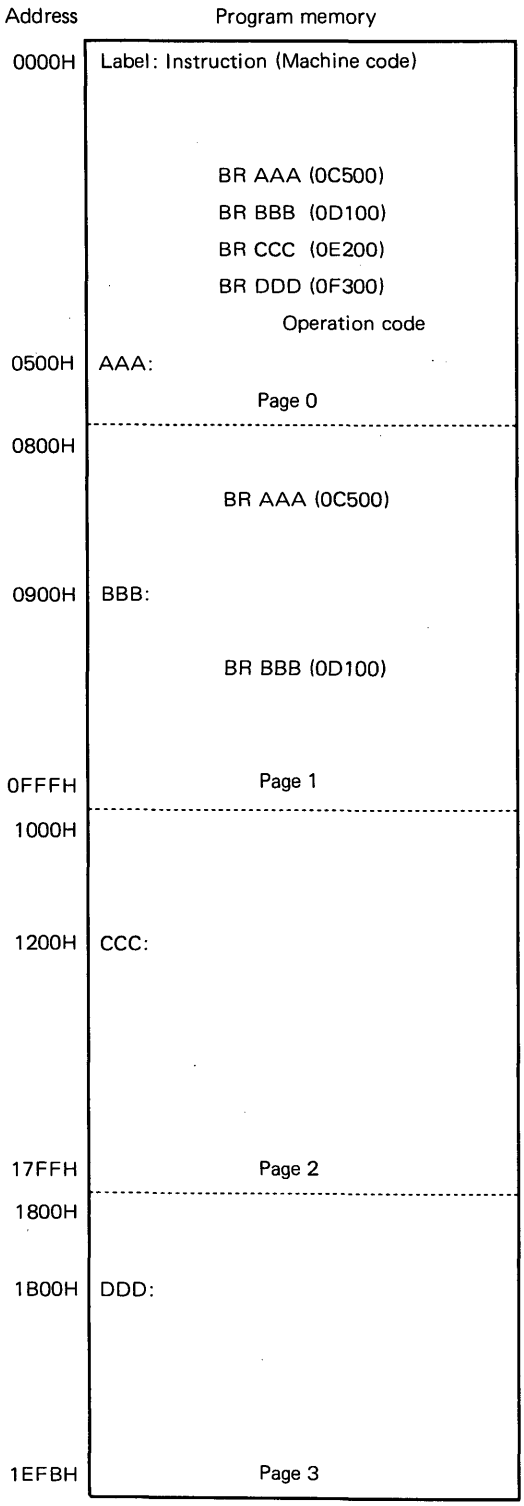
The operation code is converted automatically by referencing the jump destination specified by the label by the Assembler if Assembler (AS17K) of 17K series is used at assembling.

When patch modification is performed at debugging, the programmer must determine the page number 0, 1, 2, or 3 as the branch destination, and specify 0CH, 0DH, 0EH, or 0FH as the operation code corresponding to the page number.

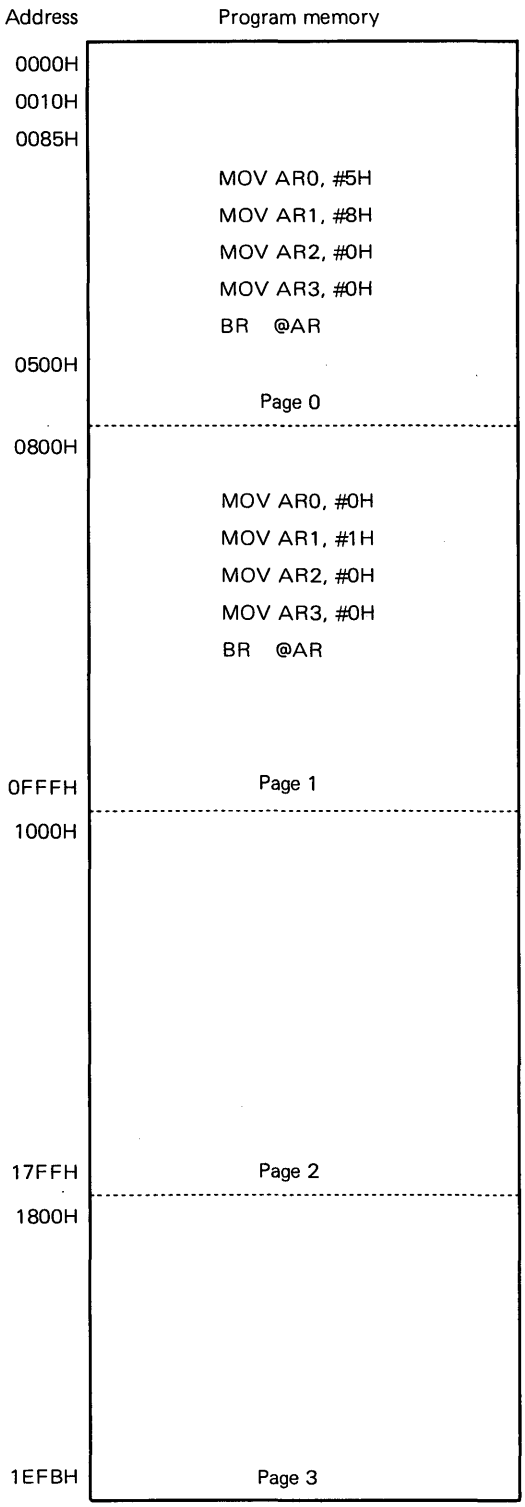
For instance, to perform patch modification of address 0900H of BBB to address 0910H in (1) of Fig. 3-2, enter "0D110" as the machine code of the "BR BBB" instruction.

Fig. 3-2 Operation of a branch instruction and machine codes

(a) Direct branching (BR addr)



(b) Indirect branching (BR @AR)



3.5 SUBROUTINE

A subroutine call instruction (CALL) and subroutine return instructions (RET and RETSK) which are dedicated instructions are used for subroutines.

Fig. 3.3 shows operation of subroutine call.

Subroutine call instructions include a direct subroutine call instruction (CALL addr) which calls the program memory address (addr) specified by the operation of the instruction, and a direct subroutine call instruction (CALL @AR) which calls the program memory address specified by the contents of the address register.

The RET instruction and RETSK instruction are used for return instructions from a subroutine. By executing the RET or RETSK instruction, control can be passed to the program memory address following the address from which the subroutine call instruction (CALL) was executed. In this case, the RETSK instruction executes the first instruction as a No Operation (NOP) instruction.

See also **4, "Program Counter (PC)".**

3.5.1 Direct Subroutine Call

A direct subroutine call specifies a program memory address as the call destination using an instruction operand of 11 bits. Consequently, when a direct subroutine call is used, the calling address, that is, the first address of the subroutine must be set within page 0 (addresses from 0000H to 07FFH). The subroutine whose first address is stored in page 1, 2, or 3 cannot be called.

However, a subroutine return instruction (RET or RETSK) can be put in page 1, 2, or 3. The CALL instruction can be stored in any page 0, 1, 2, or 3.

Example 1:

When a return instruction from a subroutine is stored in page 0.

As shown in Fig. 3.4, if the first address of the subroutine is in page 0, the return instruction can also be stored either in page 0 or page 1.

As long as the first address of the subroutine is in page 0, the CALL statement can be used without page concept.

However, when the first address of the subroutine cannot be stored in page 0, the technique described in Example 2 is useful.

Example 2:

When the first address of the subroutine is stored in page 1.

As shown in Fig. 3.4, a branch instruction (BR) is set in page 0 and the actual subroutine (SUB1) is called via the BR instruction.

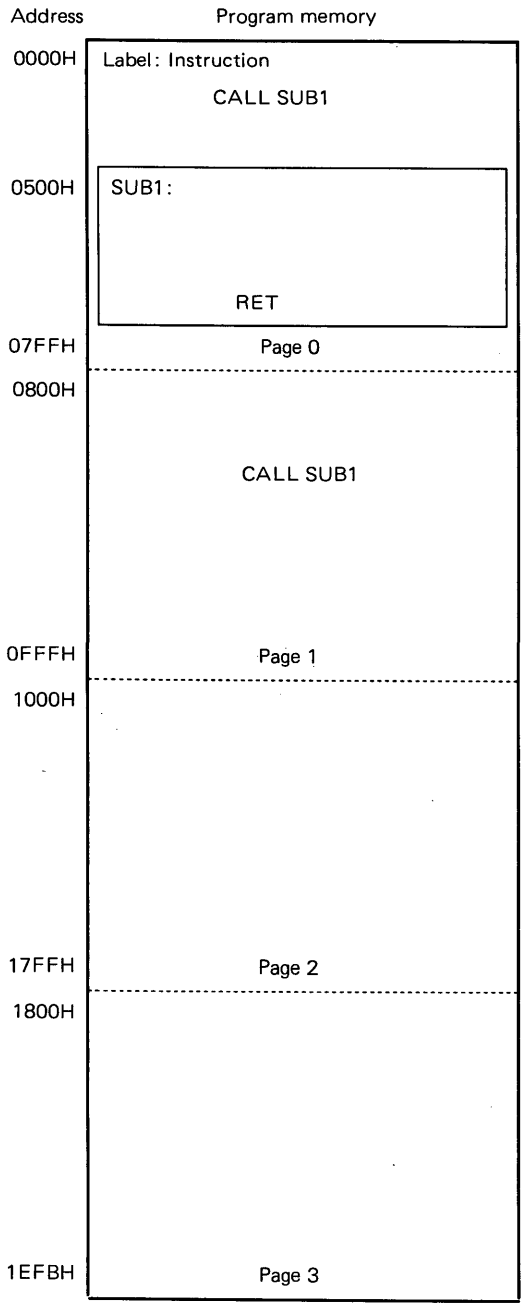
3.5.2 Indirect Subroutine Call

An indirect subroutine call instruction (CALL @AR) specifies the address of the subroutine call destination from the 13-bit data of the address register (AR). Consequently, program memory addresses from 0000H to 1EFBH can be called by an indirect subroutine.

See Section **9.3, "Address Register (AR)".**

Fig. 3-3 Operation of a subroutine call instruction

(a) Subroutine call (CALL addr)



(b) Indirect subroutine call (CALL @AR)

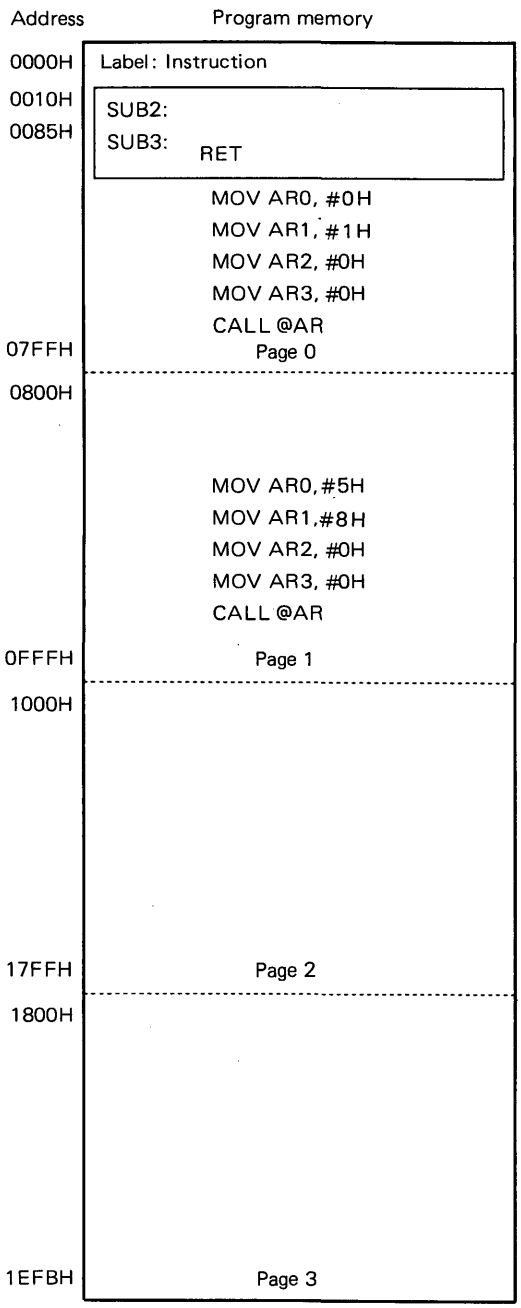
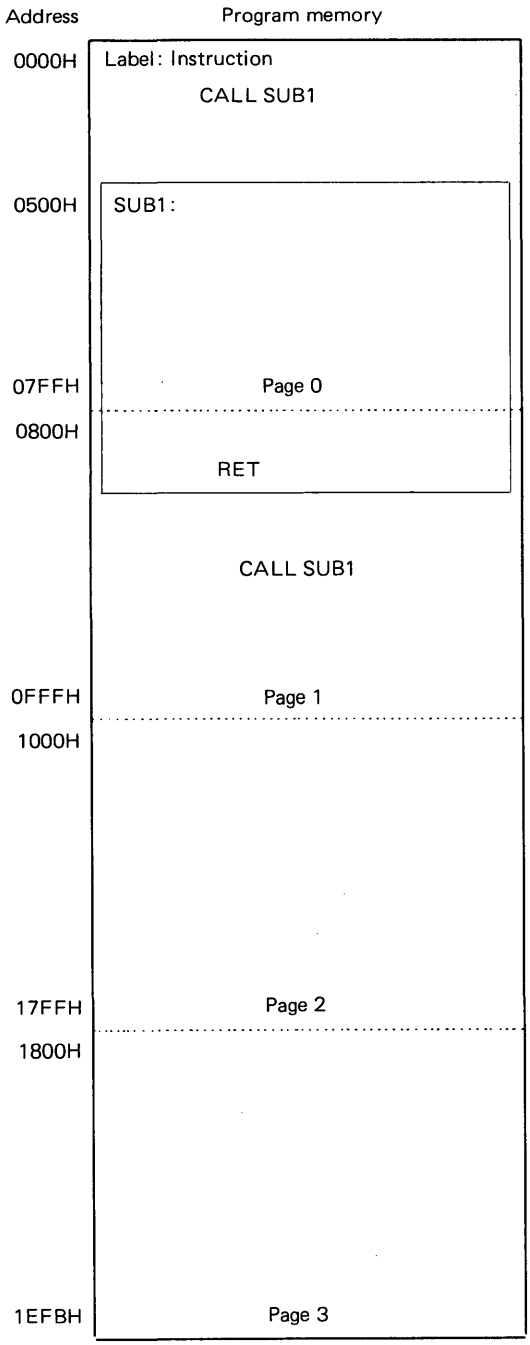
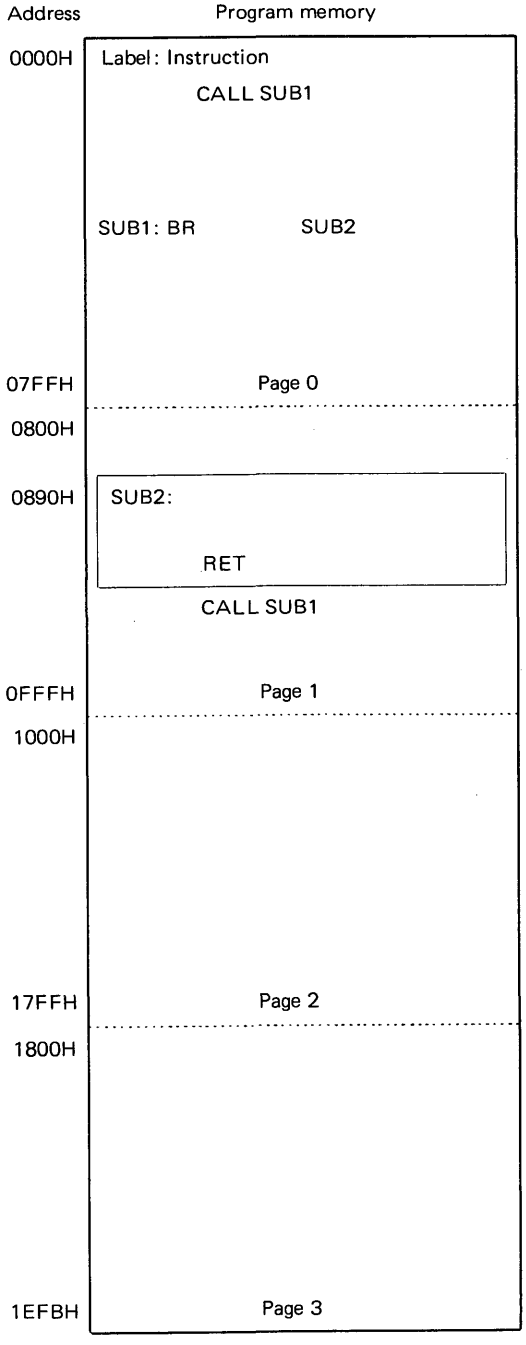


Fig. 3-4 Subroutine call instruction utilization examples

(a) When the subroutine return instruction is stored in page 1



(b) When the first address of the subroutine is stored in page 1



3.6 TABLE REFERENCING

Table referencing is used when constant data in program memory is referenced. When the MOVT DBF or @AR instruction is executed, contents of the program memory address specified by the address register are stored in the data buffer (DBF).

Since the contents of the program memory consist of 16 bits, the constant data stored in the data buffer by the MOVT instruction consists of 16 bits (4 words). Since the address register consists of 13 bits, the program memory addresses from 0000H to 1EFBH can be referenced by the MOVT instruction.

When table referencing is performed, one level of stack is used.

See Section **9.3 "Address Register"** and Section **11.3, "Data Buffer and Table Referencing".**

3.7 NOTES ON USING A BRANCHING INSTRUCTION AND A SUBROUTINE CALL INSTRUCTION

If a direct program memory address (address by a numeric value) is specified as the operand of a branch instruction (BR) or subroutine call (CALL), an error occurs when Assembler (AS17K) of 17K series is used as shown in Example 1.

This is incorporated in Assembler to reduce the debugging necessary at program fix, etc.

Example 1:

When an error occurs

```
;  ①
        BR      0005H       ;  An error occurs at Assembler
;  ②
        CALL    00F0H       ;
```

Example 2:

When an error does not occur

```
;  ③
      LOOP1:                ;  A label is used in the program and the BR or CALL instruction is executed
        BR        LOOP1     ;  using the label.
;  ④
      SUB1:                 ;
        CALL      SUB1      ;
;  ⑤
      LOOP2  LAB  0005H     ;  0005H is allocated to LOOP2 as the label type.
        BR        LOOP2     ;
;  ⑥
        BR.LD.    0005H     ;  The numeric value of the operand is converted to the label type.
                            ;  This method is not recommended for reducing debugging factors.
```

Refer to the **AS17K User's Manual** for details.

4. PROGRAM COUNTER (PC)

A program counter specifies a program memory address.

4.1 STRUCTURE OF A PROGRAM COUNTER

A program counter consists of a binary counter of 13 bits as shown in Fig. 4-1.

Fig. 4-1 Structure of a program counter

MSB LSB

PC_{12}	PC_{11}	PC_{10}	PC_9	PC_8	PC_7	PC_6	PC_5	PC_4	PC_3	PC_2	PC_1	PC_0

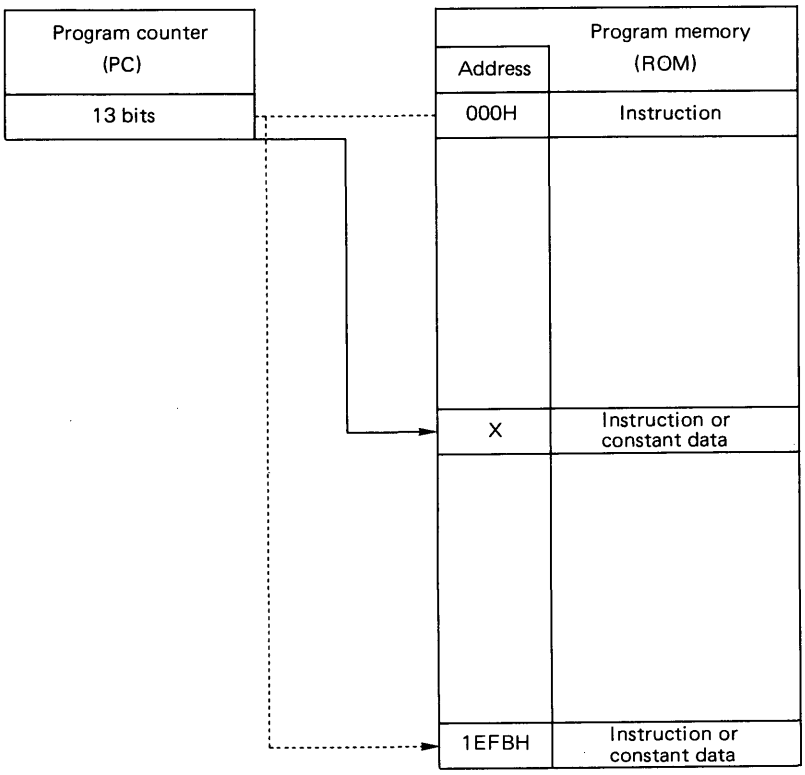
4.2 FUNCTIONS OF A PROGRAM COUNTER

As shown in Fig. 4-2, a program counter specifies the address of the instruction which is to be executed, or constant data which is to be used by a number of instructions, or constant data item written in the program memory.

Normally, the value is incremented by one whenever an instruction is executed. When a branching instruction (BR), a subroutine call instruction (CALL), a return instruction (RET, RETSK, or RETI), or a table referencing instruction (MOVT) is executed or when interrupt is accepted, the specified value is stored and the instruction at the address is executed.

Sections **4.2.1** to **4.2.6** describe the operation of a program counter when each statement is executed.

Fig. 4-2 Functions of a program counter



4.2.1 Execution of a Branch Instruction

Branch instructions include a direct branch instruction (BR addr), which directly specifies a branch destination, and an indirect branch instruction (BR @AR), which specifies the destination according to the contents of the address register (AR).

For a direct branch instruction (BR addr), the value specified by the operand (low-order 11 bits) of the instruction is stored as shown in Fig. 4-3. In this case, although there are only 11 bits for the operand of the instruction, the entire program memory area (0000H to 0EFBH) can be used for direct branching because the low-order 2 bits of the operation code (high-order 5 bits) of the instruction correspond to bit 11 (b_{11}) and bit 12 (b_{12}) of the program counter. The low-order 2 bits of the operation code of the instruction is determined by Assembler (AS17K) automatically.

For an indirect branch instruction, the contents of the address register described later are stored as shown in Fig. 4-3. In this case, addresses 0000H to 1EFBH of the program memory can be used for indirect branching.

4.2.2 Execution of a Subroutine Call Statement (CALL) and a Subroutine Return Statement (RET or RETSK)

Subroutine call instructions include a direct subroutine call instruction (CALL addr), and an indirect subroutine call instruction (CALL @AR), which specifies the call destination according to the content of the address register (AR) and which is described later.

At execution of a direct subroutine call instruction (CALL addr), the value specified by the operand of the instruction is stored as shown in Fig. 4-3. Since the high-order 2 bits of the program counter is set to 0 in this case, addresses 0000H to 07FFH of the program memory can be called by the direct subroutine call instruction.

As shown in Fig. 4-3, at execution of an indirect subroutine call instruction (CALL @AR), the content of the address register which is described later is stored. In this case, addresses from 0000H to 1EFBH of the program memory can be called by the indirect subroutine call instruction.

At execution of a subroutine return instruction (RET or RETSK), the content of the address stack register (ASR) specified by the stack pointer (SP) is stored. That is, the return address from the subroutine is stored.

4.2.3 Execution of a Table Reference Instruction (MOVT)

As shown in Fig. 4-3, the contents of the address register are stored when a table reference instruction (MOVT DBF, @AR) is executed. A table reference instruction calls the contents (16 bits) of the program memory in the address specified by the address register to a data buffer. At this time, addresses 0000H to 1EFBH of the program memory can be accessed for table reference.

A table reference instruction also stores the address following the table reference instruction execution address in the program counter after calling the contents of the memory. By this operation, program control is passed to the address following the table reference instruction. In this case, caution is necessary because one level of stack is used. Two instruction cycles (8.89 μs) are required for execution of one table reference instruction.

4.2.4 Acceptance of Interrupt and Execution of an Interrupt Return Instruction (RETI instruction)

When interrupt is accepted, the vector address specified by each interrupt is stored in the program counter as shown in Fig. 4-3. Table 4-1 lists vector addresses for each interrupt.

When an interrupt return instruction (RETI) is executed, the contents of the address stack register specified by the stack pointer which is described later are stored in the program counter. That is, the return address from interrupt processing is stored.

See **12, "Interrupt"** also.

4.2.5 Execution of a Skip Instruction

At execution of a Skip instruction (SKT, SKF, or SKE instruction), the address following the Skip instruction is stored in the program counter regardless of the contents of the skip condition. At execution of a subroutine return skip instruction (RETSK) also, the contents of the address stack register (ASR) specified by the stack pointer are stored in the program counter.

In this case, if the instruction which was executed is to be skipped according to the condition (and always in the case of RETSK), the instruction following the skip instruction is executed as a No Operation (NOP) instruction. Consequently, the same number of instructions are executed even if the following instruction is skipped when a Skip instruction is executed.

4.2.6 Resetting

At Power On Reset (V_{DD} = Low → High) or CE Reset (CE = Low → High), the contents of the program counter are reset to 0000H. Consequently, the program is executed from address 0 in this case.

At execution of a Clock Stop instruction (STOPs instruction), the program stops at the address where the instruction was executed. Since the Clock Stop instruction is released when the CE pin is changed from Low to High, the program is executed from address 0 after Clock Stop is released. A Clock Stop instruction is accepted only when the CE pin is at a Low level, and when the CE pin is at a High level, a No Operation Instruction (NOP) is used.

See **15, "Resetting"** also.

Fig. 4-3 Specifying a program counter in each instruction

Instruction \ Program counter		Contents of the program counter												
		b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
BR addr	Page 0	0	0	Operand of the instruction (addr)										
	Page 1	0	1											
	Page 2	1	0											
	Page 3	1	1											
CALL addr		0	0	Operand of the instruction (addr)										
BR @AR CALL @AR MOVT DBF, @AR		Contents of the address register												
RET RETSK RETI		Address stack register specified by the stack pointer (SP) (Return address)												
When interrupt is accepted		Vector address of each interrupt												
Power On Reset or CE Reset		0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-1 Interrupt vector addresses

Sequence	Interrupt factor	Vector address
1	INT_0 pin	0005H
2	INT_1 pin	0004H
3	Timer	0003H
4	Serial interface 1	0002H
5	Frequency counter	0001H

4.3 NOTES ON PROGRAM COUNTER OPERATION

As described before, a program counter specifies a program memory address. Addresses 0000H to 1EFBH are available for program memory while addresses which can be specified by a program counter are 0000H to 1FFFH.

This is because a program counter consists of 13 bits.

In this case, if the last address (address 1EFBH) of the program memory is neither a branch instruction (BR) nor a return instruction (RET, RETSK, or RETI), program memory does not exist in the following address (address 1EFCH) specified by the program counter.

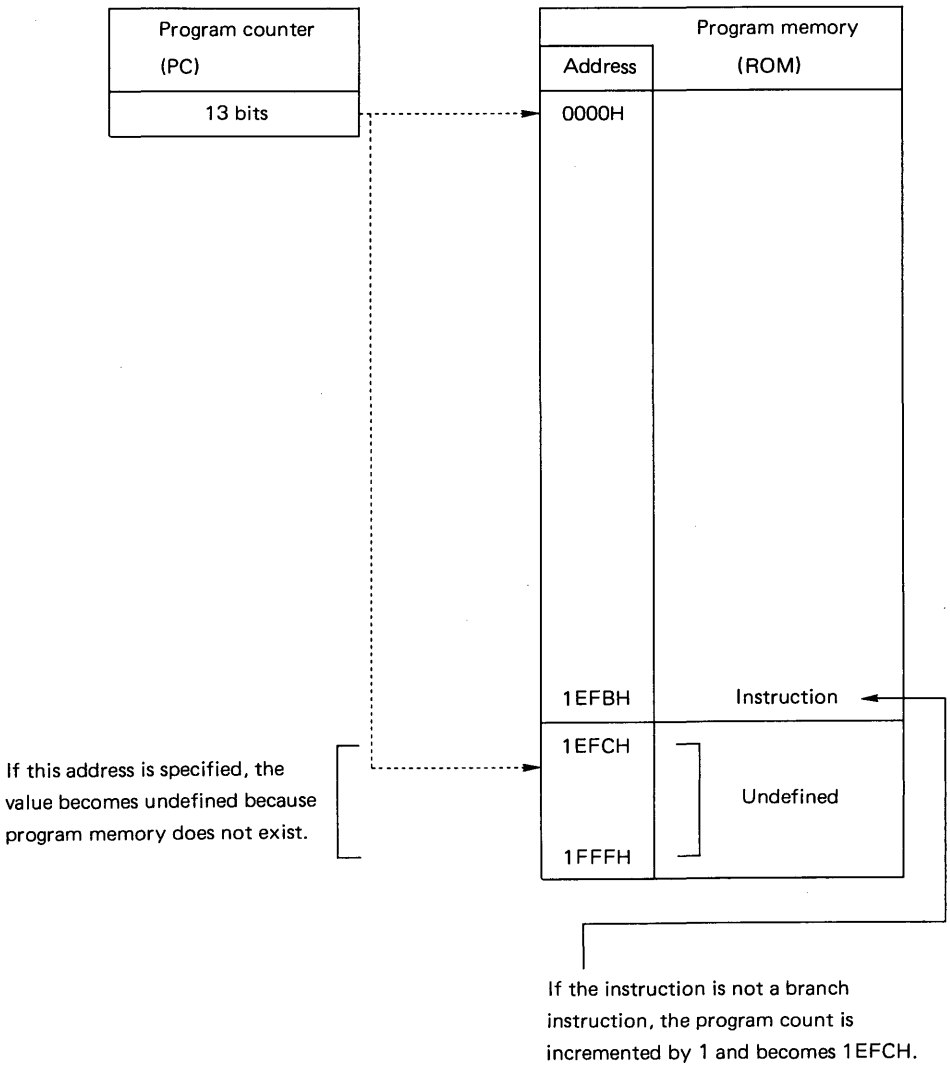
Since the contents of the addresses where program memory does not exist (addresses from 1EFCH to 1FFFH) are "undefined" as shown in Fig. 4-4, the instruction cannot be executed correctly.

Therefore, the following points must be noted.

(1) When the value of the program counter is 1EFBH, that is, when an instruction is written in 1EFBH, the instruction must be a branch instruction (BR).

(2) An instruction which causes the program counter to reach a value between 1EFCH and 1FFFH, that is, an instruction which branches control to an address between 1EFCH and 1FFFH, must not be used. However, if an instruction which passes controls to an address between 1EFCH and 1FFFH is used, the Assembler (AS17K) causes an error.

Fig. 4-4 Notes on program counter operation (PC)



5. STACK

Stack is a register for saving a program return address or the contents of a system register (which is described later) when a subroutine is called or when interrupt is accepted.

5.1 STRUCTURE OF STACK

Fig. 5-1 shows the structure of Stack.

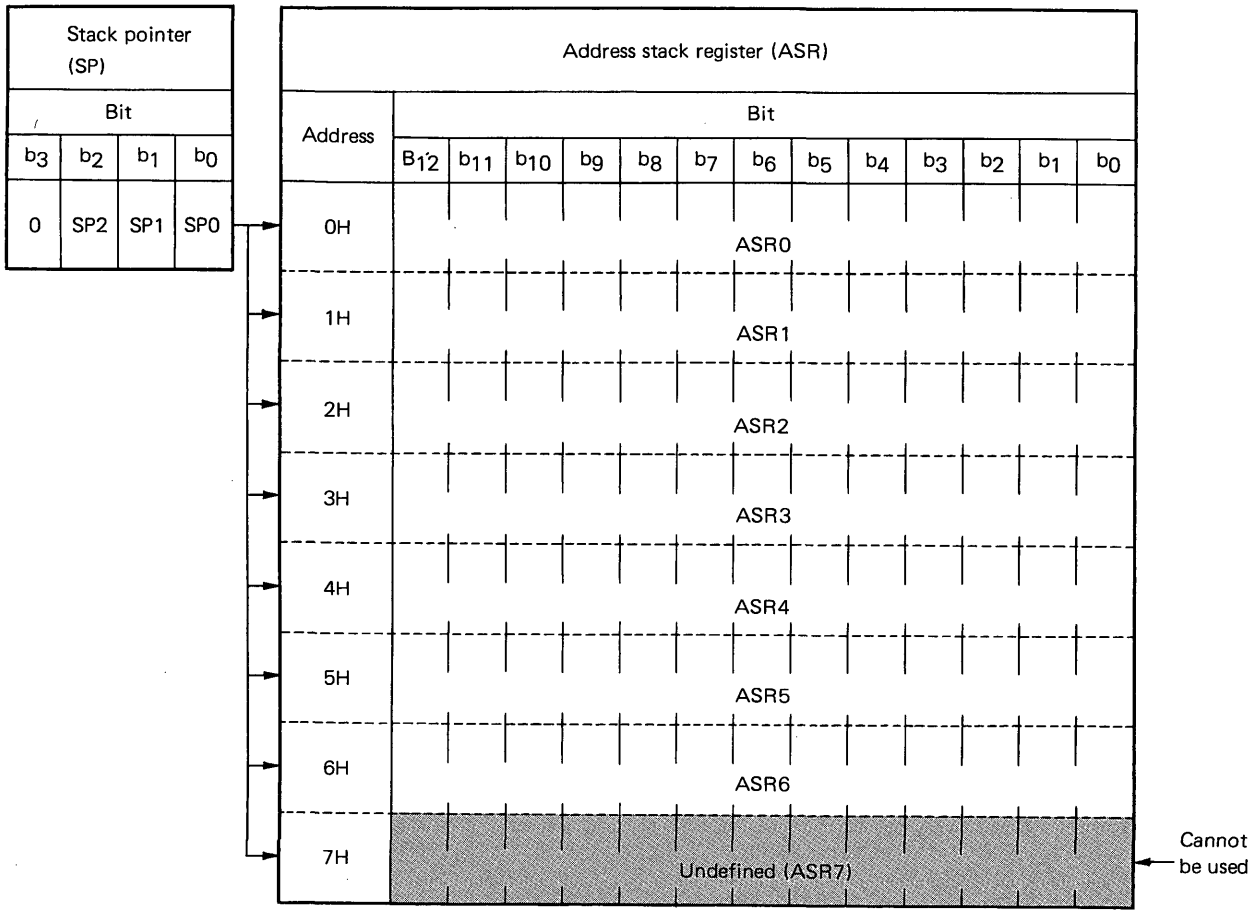
As shown in Fig. 5-1, Stack consists of a stack pointer, address stack registers and interrupt stack registers (INTSK).

A stack pointer consists of a binary counter of four bits. The high-order bit is always set to "0" and the pointer actually operates as a binary counter of 3 bits.

Although address stack registers comprise eight registers ASR0 to ASR7 each of which contains 13 bits, ASR7 does not actually exist as a register, and in effect, there are seven stack registers ASR0 to ASR6 of 13 bits.

Interrupt stack registers include four bank stack registers (BANKSK) each of which contains 4 bits and status stack registers (PSWSK) each of which contains 4 bits. However, no meanings are attached to the values of the high-order 2 bits of the bank stack register and the high-order 3 bits of the status stack register.

Fig. 5-1 Structure of a stack



Interrupt stack register (INTSK)								
Address	Bank stack (BANKSK)				Status stack (PSWSK)			
	Bits							
	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
0H	–	–	BANKSK0		–	–	–	IXE SK0
1H	–	–	BANKSK1		–	–	–	IXE SK1
2H	–	–	BANKSK2		–	–	–	IXE SK2
3H	–	–	BANKSK3		–	–	–	IXE SK3

5.2 FUNCTIONS OF A STACK

A stack saves an address returned from a subroutine or interrupt routine, or the contents of a system register (SYSREG) when a subroutine is called, or when an interrupt is accepted.

When a subroutine call instruction (CALL addr or CALL @AR) is executed, the program memory address following the subroutine call instruction execution address, that is, a return address, is saved in the address stack registers (ASR0 to ASR7). When a subroutine return instruction (RET or RETSK) is next executed, the return address which was saved in the address stack register is set in the program counter.

In addition to the operation described above, the contents of the system registers (low-order 2 bits of the bank register and Index Enable Flag; 3 bits in total) are saved in the interrupt stack register at interrupt acceptance.

At execution of a table reference instruction also (MOVT DBF, @AR), a stack is used.

A stack can be manipulated by a stack manipulation instruction (POP AR or PUSH AR).

Sections **5.3** to **5.5** describe the function of stack pointers, address stack registers, and interrupt stack registers.

5.3 STACK POINTER (SP)

A stack pointer is a 3-bit binary counter used for specifying which address stack register is to be used among the eight address stack registers (ASR0 to ASR7).

5.3.1 Structure of a Stack Pointer

The structure of a stack pointer is shown below.

A stack pointer is stored in address 01H of a control register.

See **10, "Register File"**, for control registers.

Name	Flag symbol				Address	Read/Write
	b_3	b_2	b_1	b_0		
Stack pointer SP	0	SP2	SP1	SP0	01H	R/W

Specifies the address of the address stack register (ASR)	
0 0 0	Address 0 (ASR0)
0 0 1	Address 1 (ASR1)
0 1 0	Address 2 (ASR2)
0 1 1	Address 3 (ASR3)
1 0 0	Address 4 (ASR4)
1 0 1	Address 5 (ASR5)
1 1 0	Address 6 (ASR6)
1 1 1	Address 7 (ASR7)

Set to "0"

Resetting					
	Power On	0	1	1	1
	Clock Stop	↓	1	1	1
	CE	↓	1	1	1

5.3.2 Operation of a Stack Pointer

As listed in Table 5-1, a stack pointer is decremented by 1 at execution of a subroutine call instruction (CALL addr or CALL @AR), the first instruction cycle of a reference instruction (MOVT DBF, @AR), or a stack manipulation instruction (PUSH AR) or interrupt acceptance. The pointer is incremented by 1 at execution of a subroutine return instruction (RET or RETSK), the second instruction cycle of a table reference instruction (MOVT DBF, @AR), a stack manipulation instruction (POP AR), or an interrupt return instruction (RETI).

Table 5-1 Operation of a stack pointer (SP)

Instruction	Stack value pointer
CALL addr CALL @AR MOVT DBF, @AR PUSH AR At interrupt acceptance	SP–1
RET RETSK MOVT DBF, @AR POP AR RETI	SP+1

See Section **5.6, "Stack Operation at Execution of Each Instruction"** for the actual operation performed at execution of each instruction.

Since a stack pointer is a binary counter of 3 bits as described below, the value can be between 0H and 7H. However, since there are actually only seven address stack registers, ASR0 to ASR6, caution is necessary when the stack pointer value becomes 7H. See Section **5.7, "Stack Nesting Level, PUSH Instruction, and POP Instruction".**

Since a stack pointer is located in a register file, the value can be directly read or written using the PEEK instruction or POKE instruction. In this case, although the value of the stack pointer changes, it does not influence the value of the address stack register.

At Power On Reset (V_{DD} = Low → High), execution of a Clock Stop, or CE Reset (CE = Low → High), the value of the stack pointer is 7H.

5.4 ADDRESS STACK REGISTER (ASR)

An address stack register saves a return address of the program after execution of a subroutine, interrupt processing routine, or table reference.

The register saves the program return address produced by incrementing the program counter value by 1 at execution of a subroutine call instruction (CALL addr or CALL @AR) or the first instruction cycle of a table reference instruction (MOVT DBF, @AR), or interrupt acceptance.

At execution of the "PUSH AR" instruction, which is a stack manipulation instruction, the content of the address register is saved in the address stack register.

The address stack register to which the return address is saved (which one of ASR0 to ASR7 is to be used) is the address stack register specified by the value created by decrementing the stack pointer value by 1 at execution of the instruction.

The contents of the address stack register specified by stack pointer are returned to the program counter at execution of a subroutine return instruction (RET or RETSK), interrupt return instruction (RETI), or second instruction cycle of a table reference instruction (MOVT DBF, @AR).

At execution of a stack manipulation instruction, "POP AR", the value of the address stack register specified by the stack pointer is returned to the address register.

See Section **5.6, "Stack Operation at Execution of Each Statement"** for the actual operation performed at execution of each statement.

Although there are eight address stack registers, ASR0 to ASR7, no register exists in ASR7. Therefore, caution is necessary for a subroutine call or interrupt which exceeds seven levels. See Section **5.7, "Stack Nesting Level, PUSH Instruction, and POP Instruction"** and Section **12.9, "Multi-interrupt".**

At Power On Reset (V_{DD} = Low → High), the contents of the address stack register become undefined. At CE Reset (CE = Low → High) or at execution of a Clock Stop Instruction, the previous contents are retained.

5.5 INTERRUPT STACK REGISTER

When interrupt is accepted, an interrupt stack register saves the low-order 2 bits of the bank register (BANK) in the system register and Index Enable flag (IXE) of 1 bit.

When an interrupt return instruction (RETI) is next executed, content of the interrupt stack register are returned to the bank register and Index Enable flag.

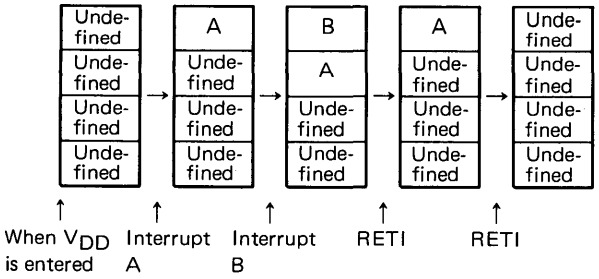
There is no address specified by a stack pointer for an interrupt stack register as an address stack register. An interrupt stack register saves data whenever an interrupt is accepted as shown in Fig. 5-2 and returns data whenever an interrupt return instruction (RETI) is executed. When an interrupt exceeding level 4 is accepted, the first data is purged. Therefore, the data must be saved by a program.

At Power On Reset (V_{DD} = Low → High), the contents of the stack register are undefined.

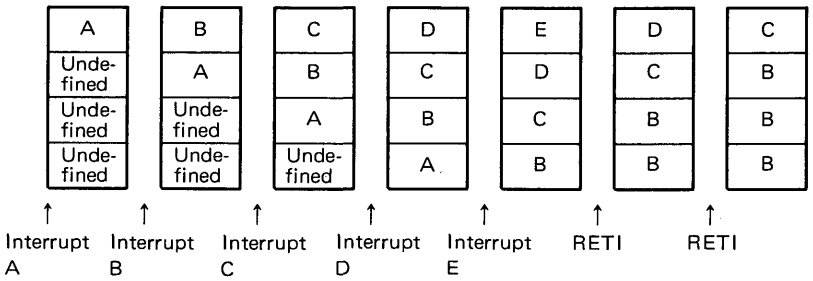
At CE Reset (CE = Low → High) or execution of a Clock Stop instruction, the previous status is retained.

Fig. 5-2 Operation of an interrupt stack register)

(a) When the interrupt does not exceed level 4



(b) When the interrupt exceeds level 4



5.6 STACK OPERATION AT EXECUTION OF EACH STATEMENT (SUBROUTINE, TABLE REFERENCE, AND INTERRUPT)

Sections **5.6.1** to **5.6.3** describe stack operation at execution of each instruction.

5.6.1 Subroutine Call Instruction (CALL) and Return Instructions (RET and RETSK)

Table 5-2 indicates operation of a stack pointer, address stack register, and program counter when a subroutine call instruction or return instruction is executed.

Table 5-2 Operation of a program counter when a subroutine is used

Instruction	Operation
CALL addr	① The program counter value is incremented by 1 ② The stack pointer value is decremented by 1 ③ The program counter value is saved in the address stack register specified by the stack pointer ④ The value specified by the operand (addr) of the instruction is transferred to the program counter
RET RETSK	① The value of the address stack register specified by the stack pointer is returned to the program counter ② The stack pointer value is incremented by 1

At execution of the RETSK instruction, the first instruction after return is a No Operation instruction (NOP).

Fig. 5-3 shows the operation example. In Fig. 5-3, there is a CALL instruction in address 100H of the main routine for calling a subroutine at address 30H and, in address 35H, a CALL instruction for calling a subroutine at address 50H.

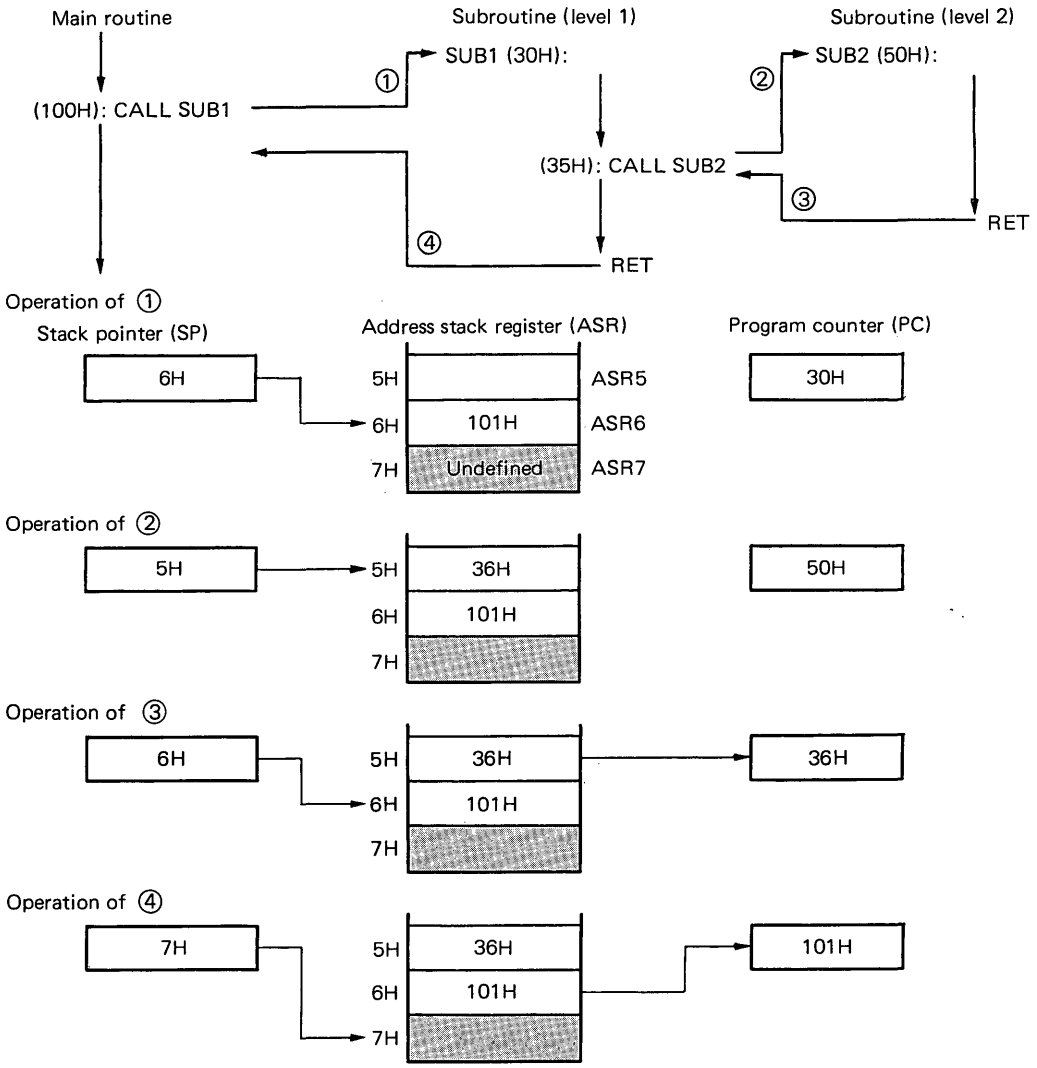
The subroutine which starts from address 30H is called a "level 1" subroutine and the subroutine which starts from address 50H is called a "level 2" subroutine. Arrows in the diagram indicate the program flow.

If 7H is assumed as the value of the stack pointer immediately preceding the execution of the instruction at address 100H, the program counter is set to 101H by execution of the CALL statement at address 100H, and the stack pointer value becomes 6H after being decremented by 1.

As a result, 101H which is the return address from the level 1 subroutine is saved in address 6H (ASR6) of the address stack register and 30H which is the operand of the CALL instruction is transferred to the program counter.

When the CALL instruction at address 35H is executed, the stack pointer value becomes 5H after being decremented by 1, 36H which is a return address from the level 2 subroutine is saved in address 5H (ASR5) of the address stack register, and 50H which is the operation of the CALL instruction is transferred to the program counter. When the RET instruction is executed in the level 2 subroutine the contents of address 5H (ASR5) of the address stack register are returned to the program counter and the stack pointer value becomes 6H after being incremented by 1. When the RET instruction of the level 1 subroutine is executed, 101H which is the return address of the main routine is returned to the program counter and the value of the stack pointer becomes 7H after being incremented by 1.

Fig. 5-3 Example of stack operation when a subroutine is called



5.6.2 Table Reference Instruction (MOVT DBF, @AR)

Table 5-3 lists operation performed at execution of the table reference instruction.

Table 5-3 Operation performed at execution of the table reference instruction

Instruction		Operation
MOVE DBF, @AR	First instruction cycle	① The program counter value is incremented by 1 ② The stack pointer value is decremented by 1 ③ The program counter value is saved in the address stack register specified by the stack register ④ The address register value is transferred to the program counter
	Second instruction cycle	⑤ The contents of the program memory specified by the program counter are transferred to a data buffer ⑥ The address stack register value specified by the stack pointer is returned to the program counter ⑦ The stack pointer value is incremented by 1

Fig. 5-4 shows the operation example.

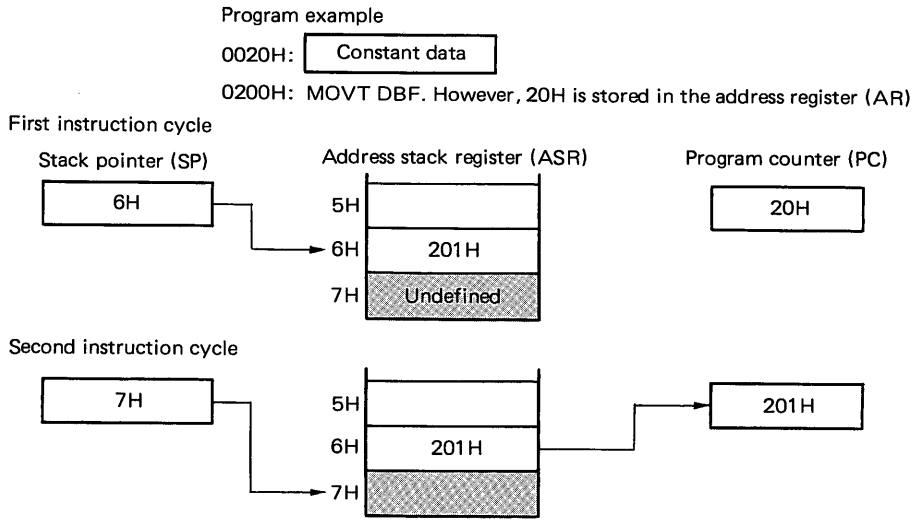
In Fig. 5-4, a table reference instruction is stored in address 200H, the program memory address which stores the constant data to be referenced is 20H, the value of the stack pointer immediately preceding execution of the "MOVT DBF, @AR" instruction of address 200H is 7H.

When the "MOVT DBF, @AR" instruction at address 200H is executed, the stack pointer value becomes 6H after being decremented by 1 at the first instruction cycle, 201H which is the address following the "MOVT DBF, @AR" instruction is saved in address 6H of the address stack register, and the program memory address which contains the constant data is transferred to the program counter.

The address 20H is specified by an address register.

Constant data of address 20H, which is the content of the program counter is transferred to the data buffer at the second instruction cycle and 201H which is the content of the address stack register is returned to the program counter. The stack pointer value becomes 7H after being incremented by 1.

Fig. 5-4 Example of stack operation at table reference



5.6.3 Interrupt Acceptance and Return Instruction (RETI Instruction)

Table 5-4 shows operation performed at interrupt acceptance and execution of a return instruction.

Table 5-4 Operation of program counter performed at interrupt operation

Instruction	Operation
At interrupt acceptance	① The program counter (PC) value is incremented by 1 However, when the instruction at interrupt acceptance is a branch instruction (BR) or subroutine call instruction (CALL), the branching address or address of the program memory to be called is set. ② The stack pointer value is decremented by 1. ③ The program counter value of the address stack register specified by the stack pointer is saved. ④ The values of the bank register (BANK) and Index Enable flag are saved in the interrupt stack. ⑤ The vector address is transferred to the program counter.
RETI	① The interrupt stack value is returned to the bank register and Index Enable flag. ② The value of the address stack register specified by the stack pointer is returned to the program counter. ③ The value of the stack pointer is incremented by 1.

Fig. 5-5 shows the operation example.

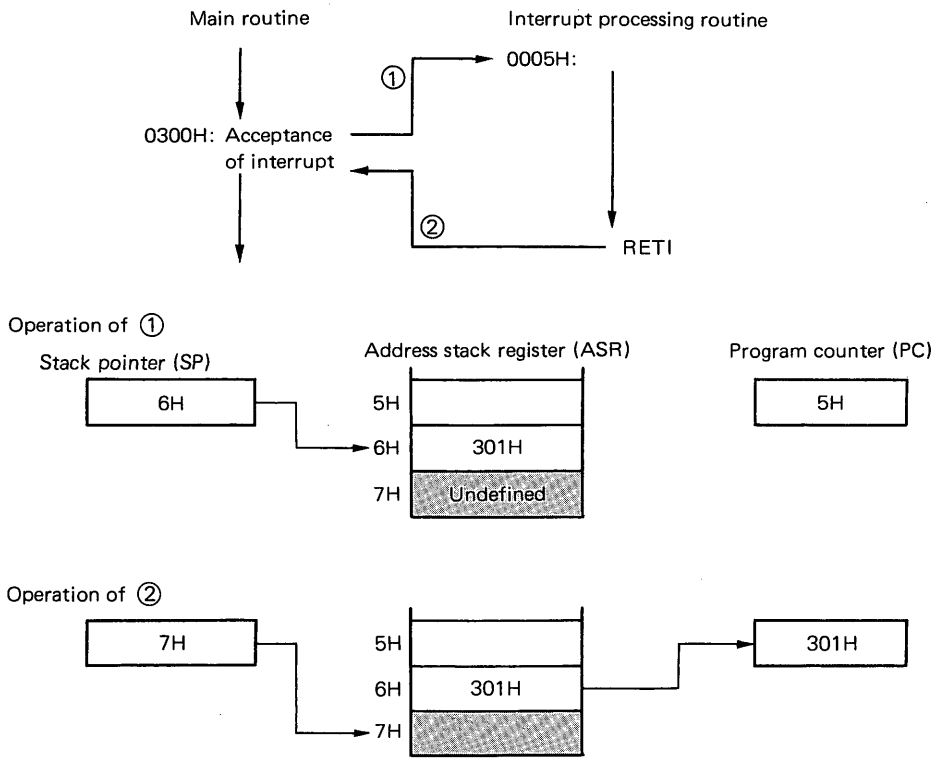
In Fig. 5-5, the stack pointer value is 7H and interrupt by the INT_0 pin is accepted during execution of an instruction at address 300H.

In this case, the stack pointer value becomes 6H after being decremented by 1 at execution of the instruction at address 300H, 301H which was to be executed next is saved at address 6H (ASR6) of the address stack register, and the low order 2 bits of the bank register and 1 bit of Index Enable flag are saved in the stack register. Interrupt vector address 0005H of the INT_0 pin is transferred to the program counter and an instruction at address 0005H is executed.

When the return instruction (RETI) is executed in the interrupt processing routine, the contents of the interrupt stack register are returned to the bank register and Index Enable flag. The content of the address stack register 301H is returned to the program counter and the stack pointer value becomes 7H after being incremented by 1.

See **12, "Interrupt"** for interrupt operation.

Fig. 5-5 Example of stack operation performed at interrupt



5.7 STACK NESTING LEVEL, PUSH INSTRUCTION, AND POP INSTRUCTION

A stack pointer operates as a 3-bit binary counter which is simply incremented or decremented by 1 by a subroutine call instruction or return instruction.

Consequently, if the CALL instruction or MOVT instruction is executed or if interrupt is accepted when the stack pointer value is 0H, the stack pointer value becomes 7H after being decremented by 1. In this case, the return address from the subroutine or interrupt processing routine or address register value is written in ASR7 which is address 7H of the address stack register.

Since ASR7 which is address 7H of the address stack register does not actually exist, the return address and address register value cannot be written.

Consequently, when a return instruction is executed and when the stack pointer value is 7H, the content of ASR7 which is address 7H of the address stack register is transferred to the program counter.

Since the content read from ASR7 which is address 7H of the address stack register is "undefined", program control is not returned normally.

This problem can be solved by saving the address stack register value using the PUSH instruction or POP instruction.

Table 5-5 shows operation of the PUSH instruction and POP instruction.

Table 5-5 Operation of the PUSH instruction and POP instruction

Instruction	Operation
POP AR	① The address stack register value specified by the stack pointer is transferred to the address register ② The stack pointer value is incremented by 1
PUSH AR	① The stack pointer value is decremented by 1 ② The address register value is transferred to the address stack register specified by the stack pointer

Fig. 5-6 shows the operation example.

In Fig. 5-6, the CALL instruction which calls the level 7 subroutine starting from address 30H is stored at address 10H of the level 6 subroutine, and the CALL instruction which calls the level 8 subroutine starting from address 50H is stored at address 35H.

The arrows in the diagram indicate the program flow.

In this example, the value of the stack pointer immediately before execution of address 10H is 1H. By executing the CALL instruction of address 10H, the stack pointer value becomes 0H after being decremented by 1. Level 7 subroutine return address, 11H, is saved in the address 0H of the address stack register. The CALL instruction operand, 30H, is transferred to the program counter.

When the POP instruction in the level 7 subroutine is executed, the stack pointer value becomes 1H after being incremented by 1 and the content of the address 0H of the address stack register, 11H, is transferred to the address register.

The stack pointer value becomes 0H after being decremented by 1 when the CALL instruction of address 35H is executed, address 36H which is the return address of the subroutine at level 8 is saved in address 0H of the address stack register. The operation of the CALL instruction, 50H, is transferred to the program counter.

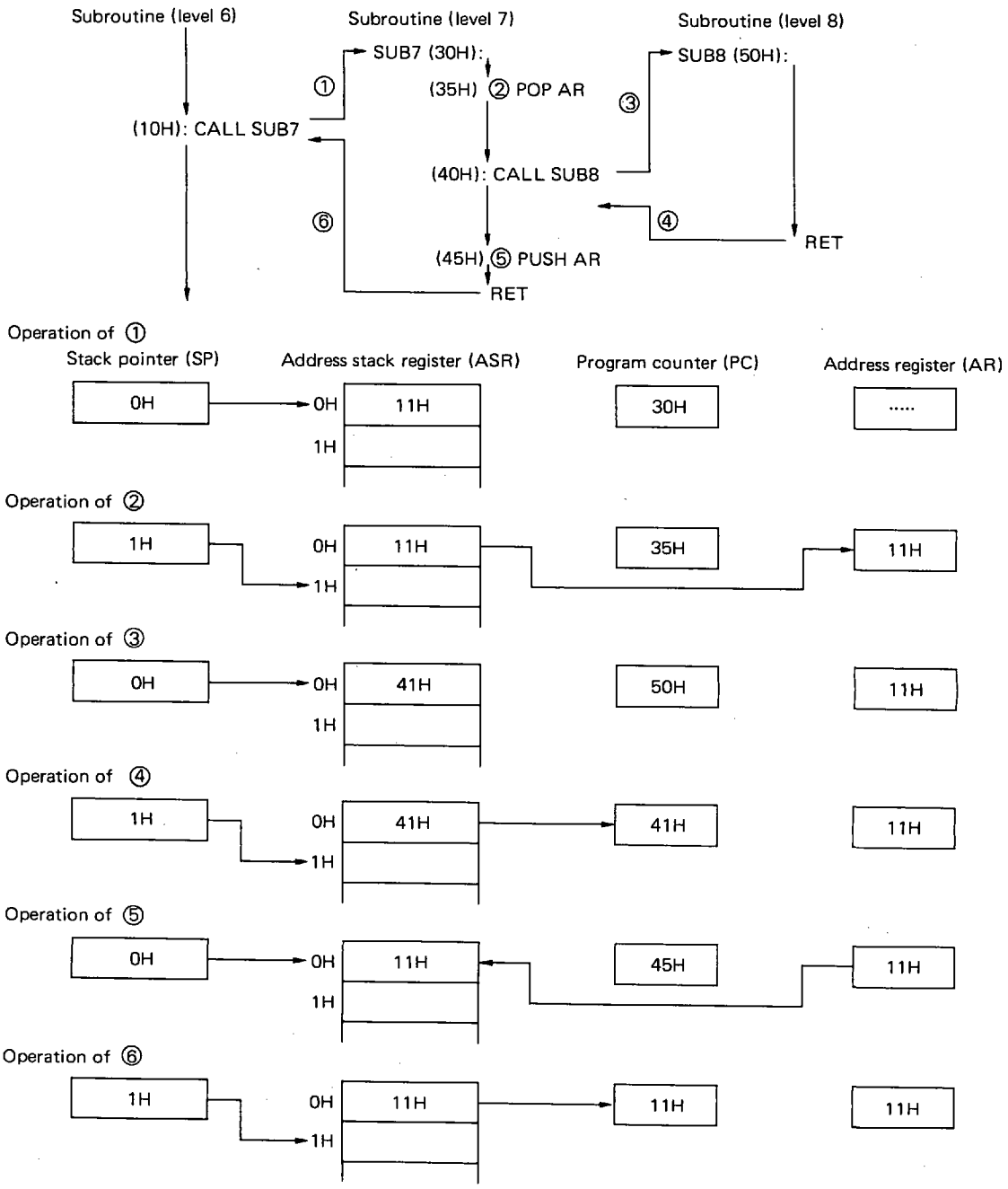
When the RET instruction is executed in the level 8 subroutine, the content of address 0H of the address stack register, 36H, is returned to the program counter, and stack pointer value becomes 1H after being incremented by 1.

When the PUSH instruction of the level 7 subroutine is executed, the stack pointer becomes 0H after being decremented by 1, address 11H which is the content of the address register, that is, the return address to the level 6 subroutine is transferred to address 0H of the address stack register.

When the RET instruction in the level 7 subroutine is executed, the content of address 0H of the address stack register, which is 11H, is returned to the program counter and the stack pointer value becomes 1H after being incremented by 1.

In the method described above, eight levels are defined as the stack nesting levels.

Fig. 5-6 Example of stack operation by the PUSH and POP instructions



6. DATA MEMORY (RAM)

Data memory is used for storing data such as operation and control. Data can be written or read using various instructions.

6.1 STRUCTURE OF DATA MEMORY

Fig. 6-1 shows the structure of data memory.

As shown in Fig. 6-1, data memory is divided into four sections in the unit called "bank". The four banks are called BANK0, BANK1, BANK2, and BANK3.

In each bank, addresses are assigned to data in 4-bit units. The high-order 3 bits are called "row address" and the low-order 4 bits are called "column address". For instance, data memory whose row address is 1H and whose column address is 0AH is called data memory of address 1AH. One address consists of memory of 4 bits and this is called "1 nibble".

Data memory is classified into the blocks described in Sections **6.1.1** to **6.1.6** according to the function.

6.1.1 Structure of a System Register (SYSREG)

A system register consists of 12 nibbles allocated to addresses from 74H to 7FH of the data memory. A system register is allocated regardless of the bank. That is, the same system register exists in addresses 74H to 7FH in any bank.

Fig. 6-2 shows the structure.

6.1.2 Structure of a Data Buffer (DBF)

A data buffer consists of 4 nibbles allocated to address 0CH to 0FH of data memory BANK0.

Fig. 6-3 shows the structure.

6.1.3 Structure of a General Register (GR)

A general register consists of 16 nibbles which are specified by any row addresses of the data memory.

The row addresses are specified by the general register pointer (RP) in the system register (SYSREG).

Fig. 6-4 shows the structure.

6.1.4 Structure of an LCD Segment Dot Data Register (LCD Dot Register)

This register consists of 16 nibbles allocated to addresses from 60H to 6FH of data memory BANK0. Fig. 6-5 shows the structure.

As shown in Fig. 6-5, since no data is assigned to address 6FH, the register actually consists of 15 nibbles.

6.1.5 Structure of a Port Data Register (Port Register)

A port data register consists of 16 nibbles which are allocated to addresses from 70H to 73H of each data memory bank.

Fig. 6-6 shows the structure.

As shown in Fig. 6-6, since no data is assigned to addresses from 71H to 73H of BANK2 and address from 70H to 73H of BANK3, the register actually consists of 9 nibbles.

6.1.6 Structure of General Data Memory

General data memory consists of a section defined by excluding a system register, LCD dot register, and port register from data memory. The memory consists of 432 nibbles in total; 96 nibbles of BANK0, and 112 nibbles each of BANK1 and BANK2, BANK3.

6.1.7 Data Memory which is Not Installed

As shown in Figs. 6-5 and 6-6, no data is assigned to address 6FH of the LCD dot register, addresses from 71H to 73H of the BANK2 of the port register, and addresses from 70H to 73H of the BANK3 of the port register.

See Section **6.3.2, "Notes on the data memory which is not installed"** for these addresses.

Fig. 6-1 Structure of data memory

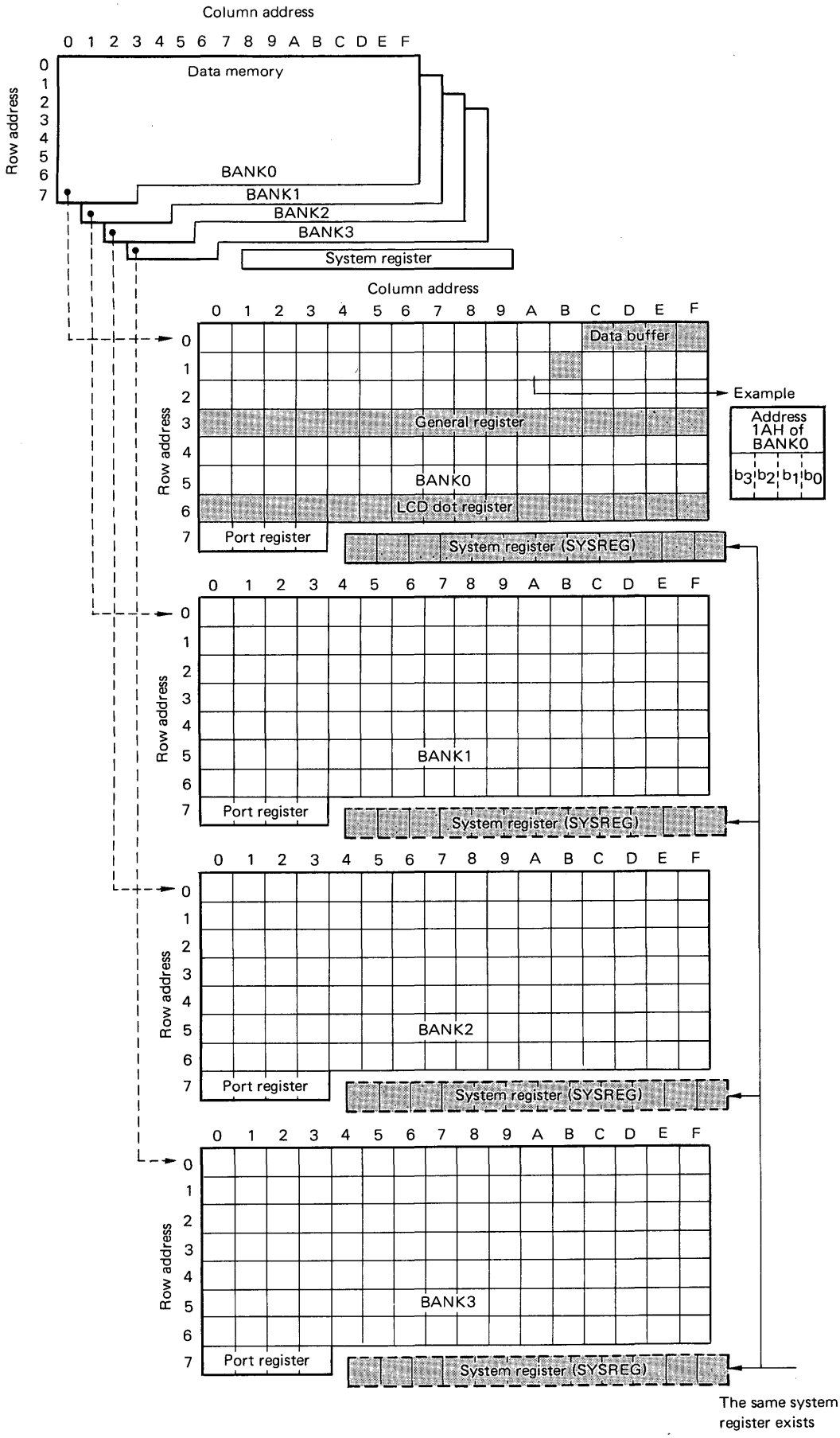


Fig. 6-2 Structure of a system register

System register (SYSREG)												
Address	74H	75H	76H	77H	78H	79H	7AH	7BH	7CH	7DH	7EH	7FH
Name (symbol)	Address register (AR)				Window register (WR)	Bank register (BANK)	Index register (IX) / Data memory row address pointer (MP)			General residence pointer (RP)		Program status word (PSWORD)

Fig. 6-3 Structure of a data buffer

Data buffer (DBF)				
Address	0CH	0DH	0EH	0FH
Symbol	DBF3	DBF2	DBF1	DBF0

Fig. 6-4 Structure of a general register (GR)

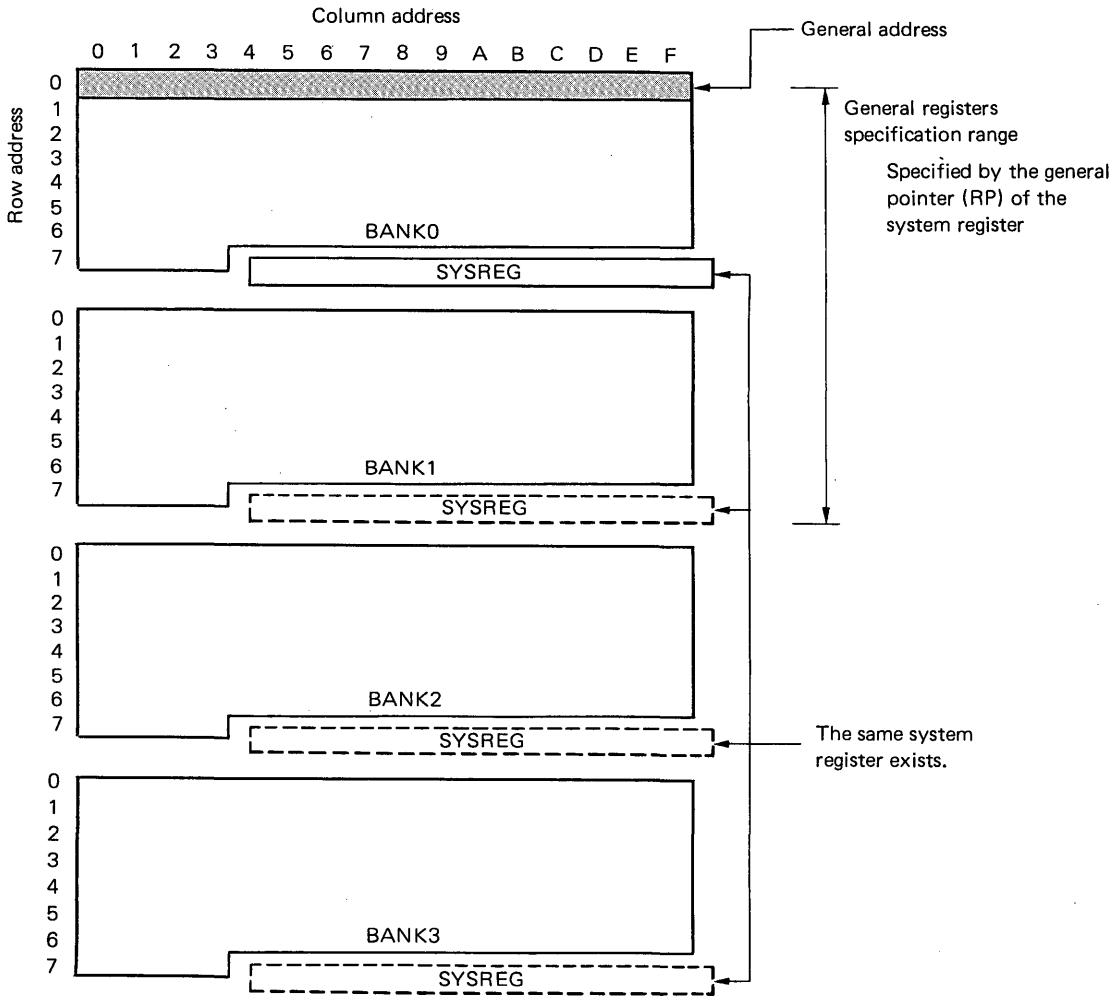


Fig. 6-5 Structure of an LCD dot register

LCD dot register																
Address	60H	61H	62H	63H	64H	65H	66H	67H	68H	69H	6AH	6BH	6CH	6DH	6EH	6FH
Symbol	LCDD0	LCDD1	LCDD2	LCDD3	LCDD4	LCDD5	LCDD6	LCDD7	LCDD8	LCDD9	LCDD10	LCDD11	LCDD12	LCDD13	LCDD14	

No data is assigned.
This area cannot be
used as data memory

Fig. 6-6 Structure of a port register

Port register					
Address		70H	71H	72H	73H
Symbol	BANK0	P0A	P0B	P0C	P0D
	BANK1	P1A	P1B	P1C	P1D
	BANK2	P2A	No data is assigned. This area cannot be used as data memory.		
	BANK3	No data is assigned. This area cannot be used as data memory.			

6.2 FUNCTIONS OF DATA MEMORY

Data memory can perform 4-bit operation, comparison, checking, and transfer with one instruction between data in data memory and immediate data (any data) by executing a data memory manipulation instruction as listed in Table 6-1.

By using a general register, 4-bit operation, comparison and transfer can be performed by one instruction.

Examples are shown below. See **7, "General Register (GR)"** and **8, "ALU"** for details.

Example 1: Operation of data memory

```
; ①
        MOV  35H,  #0001B ;  Transfers (writes) immediate data 0001B to the data memory of address
                          ;  35H of the bank which is selected at that time.
; ②
        ADD  76H,  #0001B ;  Adds immediate data 0001B to the data memory of address 76H of the
                          ;  bank which is selected at that time.
```

For both ① and ②, the selected bank is specified by the bank register in the system register. See Chapter 9, "System Register (SYSREG)" for a bank register.

② is an addition instruction for the data memory of address 76H and address 76H is also a system register. Since a system register exists regardless of the bank, this instruction adds 0001B to the system register 76H regardless of the bank.

Example 2: Operation of data memory and a general register

When the general register (GR) is stored in row address 1H of BANK0

```
; ①
        ADD      7H, 36H ;  Adds contents of the data memory at address 36H of the bank selected at
                         ;  that time to the contents of the general register whose column address is
                         ;  7H, that is, address 17H of BANK0.
; ②
        LD       7H, 36H ;  Transfers the contents of address 36H of the data memory to the general
                         ;  register whose column address is 7H. In this case, the general register is
                         ;  address 17H of BANK0.
```

A system register, data buffer, general register, LCD dot register, and port register can be manipulated by data memory manipulation instructions as data memory.

Sections **6.2.1** to **6.25** describe the functions.

6.2.1 Function of a System Register (SYSREG)

A system register controls the CPU.

For instance, the bank register shown in Fig. 6-2 specifies a bank of the data memory and the general register pointer (RP) specifies a row address of the general register.

See **9, "System Register (SYSREG)"** for details.

6.2.2 Function of a General Register (GR)

A general register is used for performed operation or data transfer with data memory.

A bank and row address of a general register are specified by the general register pointer on the system register.

For instance, if the general register pointer is set to 0, 16 nibbles of row address 0 of BANK0, that is, addresses 00H to 0FH of BANK0 are specified as a general register.

Note that a transfer instruction or operation instruction between a general register and immediate data is not allowed when a general register is used. That is, when transfer or operation is performed between a general register and immediate data, the general register must be handled as data memory.

For instance, when the general register is in row address 0H of BANK0 (register pointer is 0) and when the bank which is currently selected is BANK0 (bank register is 0), the content of the address 00H of BANK0 specified as the general register is incremented by 1 at execution of "ADD 00H, #1". If this instruction is executed when the bank currently selected is BANK1 (bank register is 1), the content of address 00H of BANK1 is incremented by 1.

See **7, "General Register (GR)"** for details.

6.2.3 Data Buffer (DBF)

A data buffer is used for storing data to be transferred to peripheral circuits such as PLL division ratio, and data sent from peripheral circuits such as input data of serial interface.

See **11, "Data Buffer"**, for details.

6.2.4 LCD Segment Dot Data Register (LCD Dot Register)

An LCD dot register is used for setting display data of LCD controller/driver. By setting data to an LCD dot register, display ON/Off data can be set to each segment of the LCD controller/driver. Fig. 6-7 shows the relationship between an LCD controller/driver and each segment.

See **23, "LCD Controller/Driver"** for details.

6.2.5 General Purpose Port Data Register (Port Register)

A port register sets output data of each general purpose input/output port or reads input data. By setting data in the port register corresponding to the pins set as output port, output of each pin is set. By reading the port register corresponding to the pins set as an input port, the input state of each pin can be detected. Fig. 6-8 shows the relationship between a port register and each port (each pin).

See **17, "General Purpose Port"** for details.

Table 6-1 Data memory manipulation instructions

Function		Instruction
Operation	Addition	ADD ADDC
	Subtraction	SUB SUBC
	Logical	AND OR XOR
Comparison		SKE SKGE SKLT SKNE
Transfer		MOV LD ST
Checking		SKT SKF

Fig. 6-7 Relationship between bits of an LCD register and each segment of an LCD controller/driver

LCD register			Relationship between segments and bits		
Address	Symbol	Bit	Common pin		Segment pin
			COM_1 (43)	COM_0 (44)	
6EH	LCDD14	b_3	b_3		$LCD_{29}/P0F_3$ (45)
		b_2		b_2	
		b_1	b_1		$LCD_{28}/P0F_2$ (46)
		b_0		b_0	
6DH	LCDD13	b_3	b_3		$LCD_{27}/P0F_1$ (47)
		b_2		b_2	
		b_1	b_1		$LCD_{26}/P0F_0$ (48)
		b_0		b_0	
6CH	LCDD12	b_3	b_3		$LCD_{25}/P0E_3$ (49)
		b_2		b_2	
		b_1	b_1		$LCD_{24}/P0E_2$ (50)
		b_0		b_0	
6BH	LCDD11	b_3	b_3		$LCD_{23}/P0E_1$ (51)
		b_2		b_2	
		b_1	b_1		$LCD_{22}/P0E_0$ (52)
		b_0		b_0	
6AH	LCDD10	b_3	b_3		$LCD_{21}/P0X_5$ (53)
		b_2		b_2	
		b_1	b_1		$LCD_{20}/P0X_4$ (54)
		b_0		b_0	
69H	LCDD9	b_3	b_3		$LCD_{19}/P0X_3$ (55)
		b_2		b_2	
		b_1	b_1		$LCD_{18}/P0X_2$ (56)
		b_0		b_0	
68H	LCDD8	b_3	b_3		$LCD_{17}/P0X_1$ (57)
		b_2		b_2	
		b_1	b_1		$LCD_{16}/P0X_0$ (58)
		b_0		b_0	
67H	LCDD7	b_3	b_3		$LCD_{15}/P0Y_{15}/KS_{15}$ (59)
		b_2		b_2	
		b_1	b_1		$LCD_{14}/P0Y_{14}/KS_{14}$ (60)
		b_0		b_0	
66H	LCDD6	b_3	b_3		$LCD_{13}/P0Y_{13}/KS_{13}$ (61)
		b_2		b_2	
		b_1	b_1		$LCD_{12}/P0Y_{12}/KS_{12}$ (62)
		b_0		b_0	
65H	LCDD5	b_3	b_3		$LCD_{11}/P0Y_{11}/KS_{11}$ (63)
		b_2		b_2	
		b_1	b_1		$LCD_{10}/P0Y_{10}/KS_{10}$ (64)
		b_0		b_0	
64H	LCDD4	b_3	b_3		$LCD_9/P0Y_9/KS_9$ (65)
		b_2		b_2	
		b_1	b_1		$LCD_8/P0Y_8/KS_8$ (66)
		b_0		b_0	
63H	LCDD3	b_3	b_3		$LCD_7/P0Y_7/KS_7$ (67)
		b_2		b_2	
		b_1	b_1		$LCD_6/P0Y_6/KS_6$ (68)
		b_0		b_0	
62H	LCDD2	b_3	b_3		$LCD_5/P0Y_5/KS_5$ (69)
		b_2		b_2	
		b_1	b_1		$LCD_4/P0Y_4/KS_4$ (70)
		b_0		b_0	
61H	LCDD1	b_3	b_3		$LCD_3/P0Y_3/KS_3$ (71)
		b_2		b_2	
		b_1	b_1		$LCD_2/P0Y_2/KS_2$ (72)
		b_0		b_0	
60H	LCDD0	b_3	b_3		$LCD_1/P0Y_1/KS_1$ (73)
		b_2		b_2	
		b_1	b_1		$LCD_0/P0Y_0/KS_0$ (74)
		b_0		b_0	

( ) : Pin number

Fig. 6-8 Relationship between a port register and each point (pin)

General purpose port data register					Port	Pin		
Bank	Address	Symbol	Bit	symbol		Number	Symbol	Input/Output
BANK0	70H	P0A	b_3	P0A3	Port 0A	3	$P0A_3$	Input/Output (bit I/O)
			b_2	P0A2		4	$P0A_2$	
			b_1	P0A1		5	$P0A_1$	
			b_0	P0A0		6	$P0A_0$	
	71H	P0B	b_3	P0B3	Port 0B	7	$P0B_3$	Input/Output (bit I/O)
			b_2	P0B2		8	$P0B_2$	
			b_1	P0B1		9	$P0B_1$	
			b_0	P0B0		10	$P0B_0$	
	72H	P0C	b_3	P0C3	Port 0C	79	$P0C_3$	Input/Output (group I/O)
			b_2	P0C2		80	$P0C_2$	
			b_1	P0C1		1	$P0C_1$	
			b_0	P0C0		2	$P0C_0$	
	73H	P0D	b_3	P0D3	Port 0D	75	$P0D_3$	Input
			b_2	P0D2		76	$P0D_2$	
			b_1	P0D1		77	$P0D_1$	
			b_0	P0D0		78	$P0D_0$	
BANK1	70H	P1A	b_3	P1A3	Port 1A	14	$P1A_3$	Input/Output (bit I/O)
			b_2	P1A2		15	$P1A_2$	
			b_1	P1A1		16	$P1A_1$	
			b_0	P1A0		17	$P1A_0$	
	71H	P1B	b_3	P1B3	Port 1B	18	$P1B_3$	Output
			b_2	P1B2		19	$P1B_2$	
			b_1	P1B1		20	$P1B_1$	
			b_0	P1B0		21	$P1B_0$	
	72H	P1C	b_3	P1C3	Port 1C	22	$P1C_3$	Output
			b_2	P1C2		23	$P1C_2$	
			b_1	P1C1		24	$P1C_1$	
			b_0	P1C0		25	$P1C_0$	
	73H	P1D	b_3	P1D3	Port 1D	26	$P1D_3$	Input
			b_2	P1D2		27	$P1D_2$	
			b_1	P1D1		28	$P1D_1$	
			b_0	P1D0		29	$P1D_0$	
BANK2	70H	P2A	b_3	P2A3	– – –	No related pin Cannot be used for data memory either		
			b_2	P2A2				
			b_1	P2A1				
			b_0	P2A0	Port 2A	42	$P2A_0$	Output
	71H	– – –	b_3	– – –	No address is assigned Cannot be used for data memory either			
			b_2					
			b_1					
			b_0					
	72H	– – –	b_3	– – –				
			b_2					
			b_1					
			b_0					
	73H	– – –	b_3	– – –				
			b_2					
			b_1					
			b_0					
BANK3	70H	– – –	b_3	– – –	No address is assigned Cannot be used for data memory either			
			b_2					
			b_1					
			b_0					
	71H	– – –	b_3	– – –				
			b_2					
			b_1					
			b_0					
	72H	– – –	b_3	– – –				
			b_2					
			b_1					
			b_0					
	73H	– – –	b_3	– – –				
			b_2					
			b_1					
			b_0					

6.3 NOTES ON USING DATA MEMORY

6.3.1 Data Memory Address Specification

When Assembler (AS17K) of 17K is used, and when a data memory address is coded directly using a numeric value as the operand of the data memory manipulation instruction, an error occurs.

This is incorporated in the Assembler to reduce necessary debugging at program fix, etc.

Example 1:

When an error occurs

```
;  ①
        MOV    2FH,  #0001B ;  Specifies address 2FH directly
;  ②
        MOV  0.2FH,  #0001B ;  Specifies address 2FH of BANK0 directly
```

When an error does not occur

```
;  ③
        M02F  MEM  0.2FH      ;  Defines a symbol in M02F using address 2FH of BANK0 as the
        MOV  M02F  #0001B     ;  memory type.
;  ④
        MOV .MD .2FH, #0001B  ;  Converts address 2FH to a memory type using .MD.
                              ;  However, this method must be avoided to reduce necessary debugging.
```

Consequently, the data memory address symbol must be defined in advance using the MEM instruction (symbol definition pseudo instruction) which is an Assembler pseudo instruction.

As shown in Example 2, a bank of the data memory must also be defined for data memory symbol definition. This is used for creating a data memory map automatically in Assembler.

If the data memory for which symbol definition is set in BANK2 is used in the range of BANK1 in the program as shown in Example 2, data memory of BANK1 is manipulated.

Example 2:

```
M1  MEM  0.15H ;
M2  MEM  1.15H ;    Symbol definition pseudo instruction
M3  MEM  2.15H ;
```

Bank — Row address

Column address

```
BANK1                ;  Assembler built-in macro instruction
                     ;  BANK ← 1
MOV  M1,  #0000B ;
MOV  M2,  #0000B ;   Although symbol definition of M1, M2, and M3 are set in another bank in 1,
MOV  M3,  #0000B ;   these three instructions write 0 in the data memory of address 15H of BANK1
                     because BANK1 is specified in the program.
```

6.3.2 Notes on Data Memory which is Not Installed

As shown in Figs. 6-5 and 6-6, no address is assigned to address 6FH of an LCD dot register, addresses of 71H to 73H of address BANK2 of a port register, and addresses of 70H to 73H of BANK3 of a port register.

In this case, if a data memory manipulation instruction is executed for these addresses, the following operation is performed.

(1) Device operation

When a Read instruction is executed, "0" is read.

Even if a Write instruction is executed, no change is made.

(2) Assembler (AS17K) operation

Normally, data is assembled.

An error does not occur.

(3) Emulator (IE-17K) operation

When a Read instruction is executed, "0" is read.

Even if a Write instruction is executed, no change is made.

An error does not occur.

7. GENERAL REGISTER (GR)

A general register is stored in data memory and is used for direct operation and for transfer performed with data memory.

7.1 STRUCTURE OF A GENERAL REGISTER

Fig. 7-1 shows the structure of a general register.

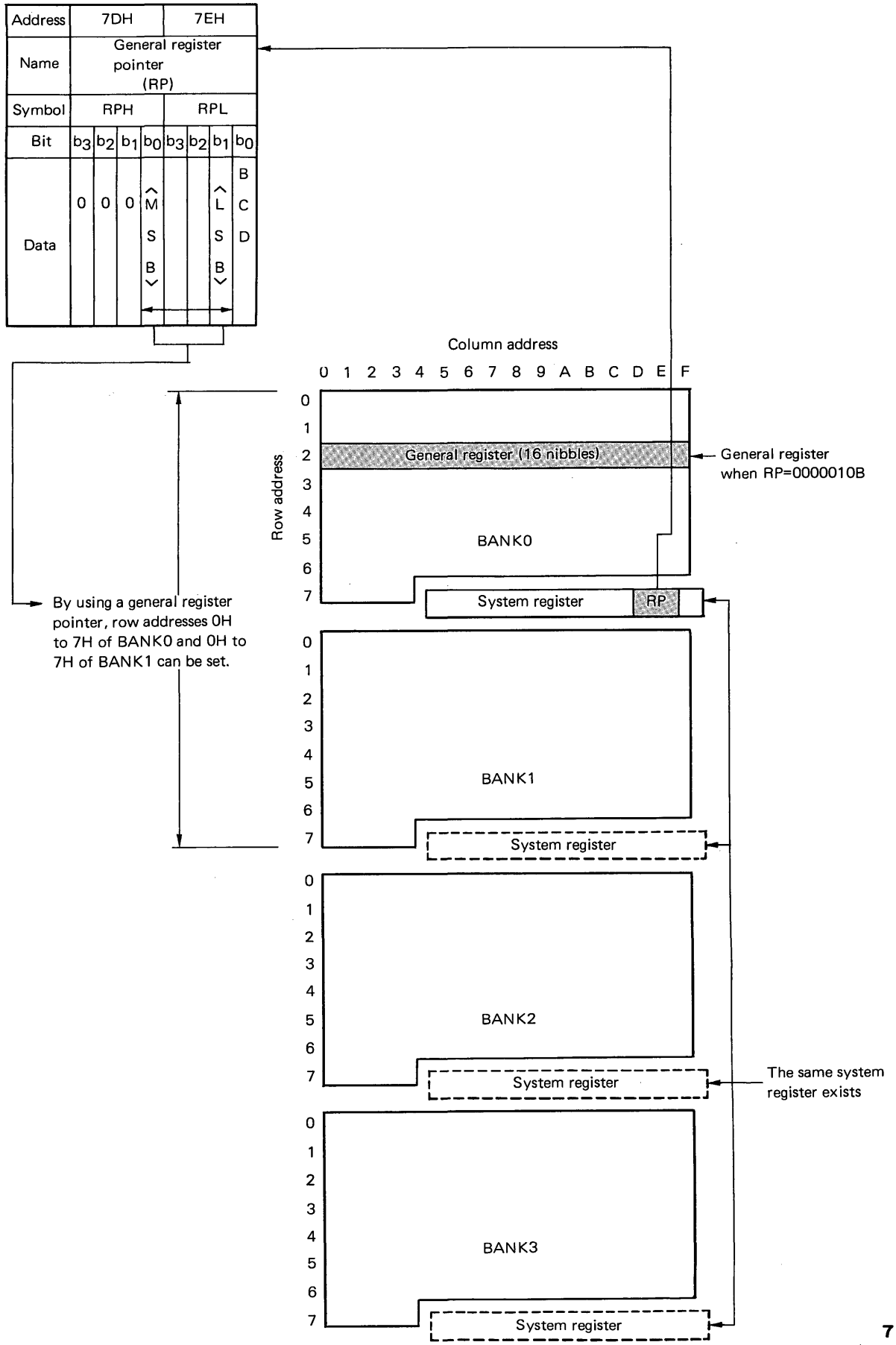
As shown in Fig. 7-1, 16 nibbles (16 words x 4 bits) which are row addresses of the data memory can be used as a general register.

The row addresses which are used are set by a general register pointer of the system register.

Since 4 bits can be used as a general register pointer, row addresses from 0H to 7H of BANK0 and row addresses from 0H to 7H of BANK1 can be used as general registers.

See Section **9.7, "General Register Pointer (RP)"** also.

Fig. 7-1 Structure of a general register



7.2 FUNCTION OF A GENERAL REGISTER

By using a general register, operation and transfer can be performed by one instruction between data memory and a general register.

Since a general register is stored in data memory, operation and transfer between one data memory and another data memory can be performed by just one instruction.

Since a general register is stored in data memory, the register can be controlled by data memory manipulation instructions in the same way as other data memory.

Section **7.3** describes operation of data memory manipulation instructions.

7.3 GENERAL REGISTER IN EACH INSTRUCTION AND DATA MEMORY ADDRESS GENERATION AND OPERATION

Table 7-1 lists operation and transfer instructions used between general register and data memory.

Table 7-2 shows a general register and data memory address generation.

For example, address "R" of the general register specified by the ADD r, m instruction is generated by the content of the register pointer and the value specified by the operand r of the instruction as shown in Table 7-2.

Data memory address "M" specified by this instruction is generated by the content of the bank register and operand m.

Consequently, "R", the content of the general register specified by the general register address R and "(M)", the content of the data memory specified by the data memory address M are added and the result is stored in the general register.

Addresses of a general register are generated in the same way as shown in Table 7-1 for other instructions also. Examples 1, 2, and 3 show the operation examples.

Table 7-1 Manipulation instructions between a general register and data memory

Instruction group	Instruction	Operation
Addition	ADD r, m	(R) ← (R) + (M)
	ADDC r, m	(R) ← (R) + (M) + (CY)
Subtraction	SUB r, m	(R) ← (R) − (M)
	SUBC r, m	(R) ← (R) − (M) − (CY)
Logical operation	AND r, m	(R) ← (R) AND (M)
	OR r, m	(R) ← (R) OR (M)
	XOR r, m	(R) ← (R) XOR (M)
Transfer	LD r, m	(R) ← (M)
	ST m, r	(M) ← (R)
	MOV @r, m	[MP, (R)] ← (M) or [m, (R)] ← (M)
	MOV m, @r	M← [MP, (R)] or M ← [H, (R)]
Shift	RORC r	Right shift including (CY)

Table 7-2 General register and data memory address generation

Instruction	Address contents	Address generated											
		Symbol	Bank				Row address			Column address			
			b_3	b_2	b_1	b_0	b_2	b_1	b_0	b_3	b_2	b_1	b_0
ADD r, m	Address of the general register specified by r	R	(RP)							r			
	Address of the data memory specified by m	M	(BANK)				m						

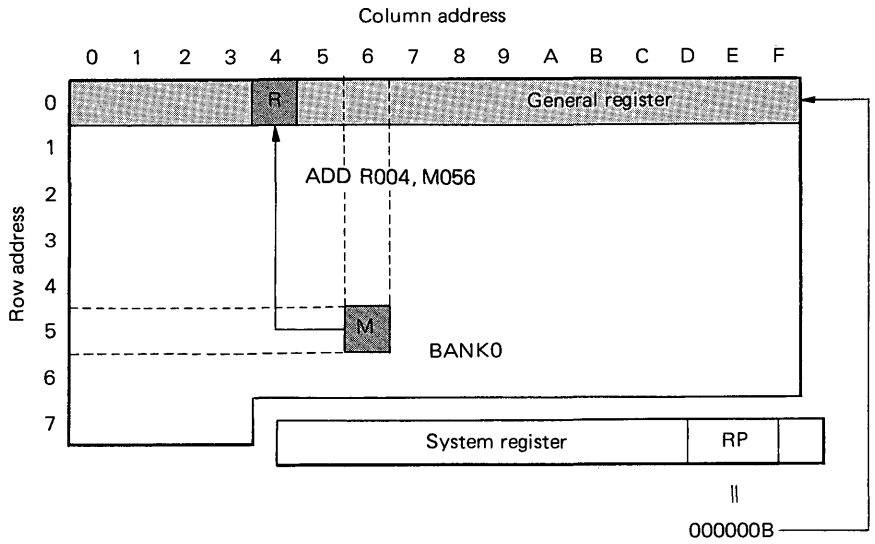
Example 1: Operation of data memory and a general register

(1) When the bank of the data memory and the bank of the general register are the same
The bank is BANK0 and the general register is stored in row address 0H of BANK0.

```
R004  MEM  0.04H ;  Symbol definition
M056  MEM  0.56H ;
ADD   R004, M056 ;  Addition of data memory and general register
```

If the above instruction is executed, the content of R004 (address 04H of BANK0) which is the general register, and the content of data memory M056 (address 56H) are added and the result is stored in general register R004 (04H) as shown in Fig. 7-2.

Fig. 7-2 Example of operation of data memory and a general register

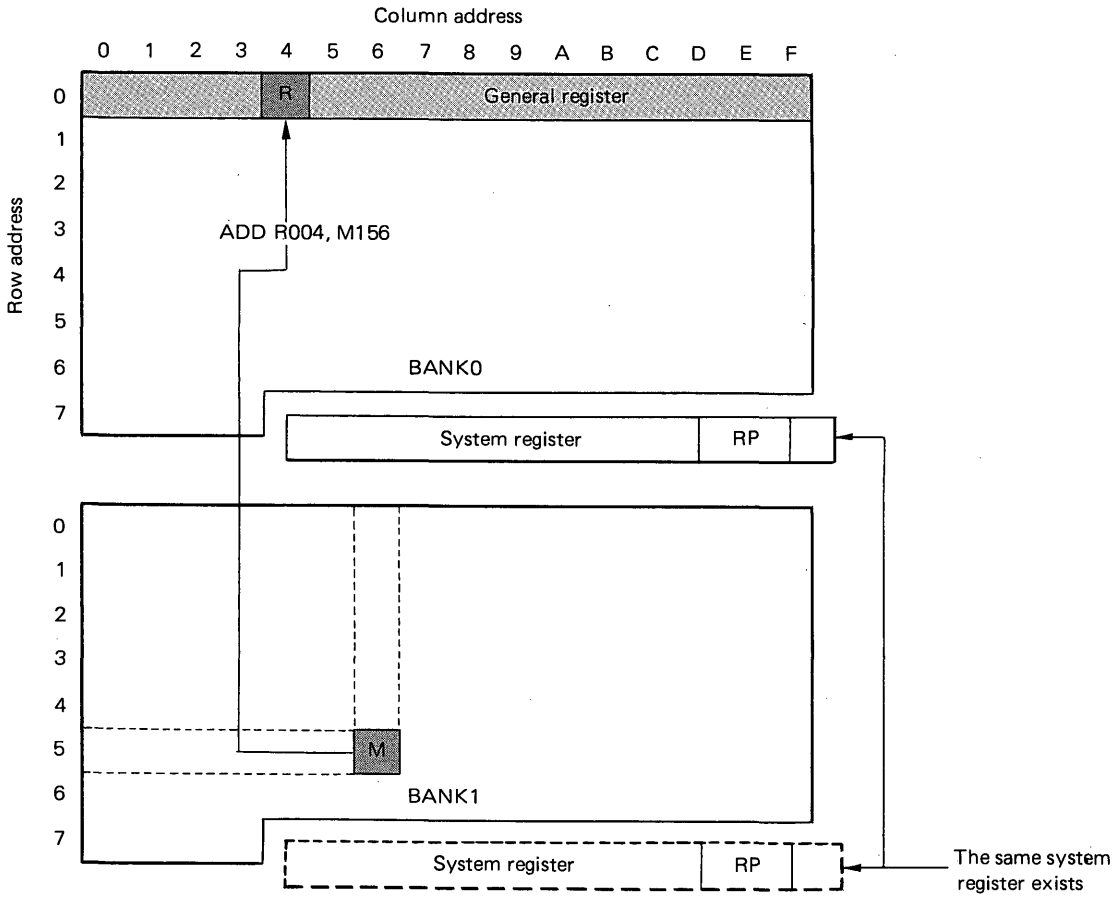


(2) When the bank of the data memory and the bank of the general register are different
BANK1 is selected and the general register is stored in row address 0H of BANK0.

```
R004  MEM  0.04H  ;  Symbol definition
M156  MEM  1.56H  ;
BANK1             ;  Assembler (AS17K) built-in macro instruction
ADD   R004, M156  ;  Addition of data memory and a general register
```

When the above instruction is executed, the content of register R004 (address 04H of BANK0) and the content of data memory M156 (address 56H of BANK1) are added and the result is stored in general register R004 (04H). That is, although BANK1 is selected, data memory of BANK1 and data memory of BANK0 are added by using only one instruction because the general register is stored in BANK0.

Fig. 7-3 Operation example between data memory and a general register

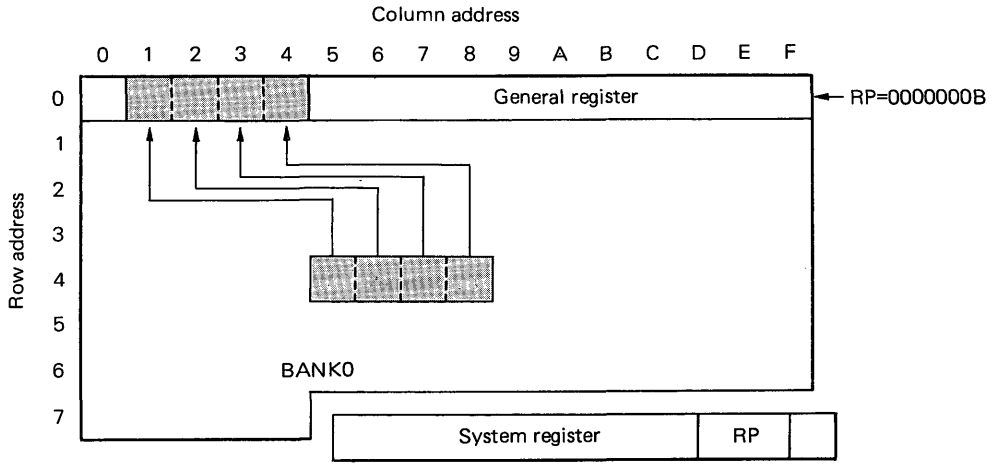


Example 2: Transfer of data to general registers
BANK0 is selected and the general register is in row address 0H of BANK0.

```
R001  MEM  0.01H ;  Symbol definition
R002  MEM  0.02H ;
R003  MEM  0.03H ;
R004  MEM  0.04H ;
M045  MEM  0.45H ;
M046  MEM  0.46H ;
M047  MEM  0.47H ;
M048  MEM  0.48H ;
LD    R001, M045
LD    R002, M046
LD    R003, M047
LD    R004, M048
```

As shown in Fig. 7-4, the above program transfers the contents of data memory, M045, M046, M047, and M048 (address 45H, 46H, 47H, and 48H) to general registers, R001, R002, R003, and R004 (addresses 01H, 02H, 03H, and 04H) respectively.

Fig. 7-4 Example of transfer to general registers



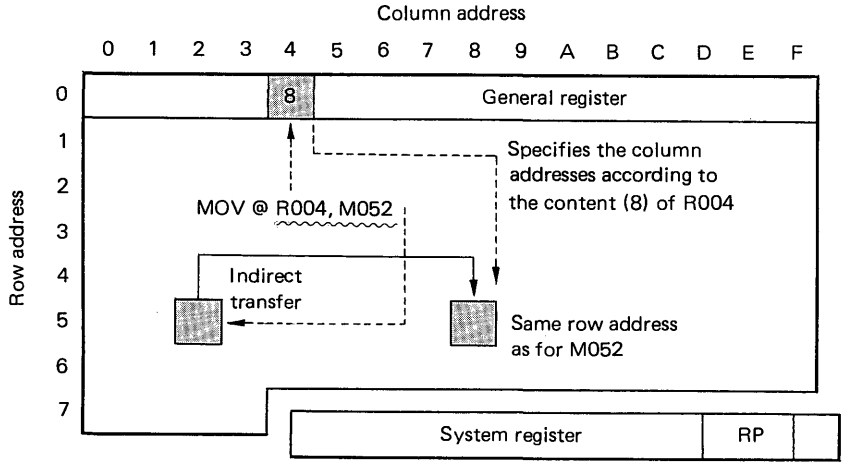
Example 3: General register indirect transfer
BANK0 is selected and the general register is row address 0H of BANK0.

```
R004  MEM   0.04H ;
M052  MEM   0.52H ;
MOV   R004, #8    ; (R004) ← 8
MOV   @R004, M052 ; Transfer between general registers
```

When the above instruction is executed, the content of data memory M052 (address 52H) is transferred to data memory (in this example, address 58H) of indirect transfer destination.
The MOV @r, m instruction is called general register indirect transfer and it transfers the content of the data memory whose address is specified by m to the data memory whose address is specified by @r (called indirect address).
In this case, the data memory address (indirect address) of the indirect transfer destination specified by @r is the same row address (5H in the above example) as the address of the data memory specified by m. The content of the general register specified by r (in the above example, 8, the content of address 04H) is used as the column address of the data memory, that is, address 58H becomes the data memory address of the indirect transfer destination.

See Section **9.6, "Index Register (IX) and Data Memory Row Address Pointer (MP: Memory Pointer)"**, for general register indirect transfer.

Fig. 7-5 Example of general register indirect transfer



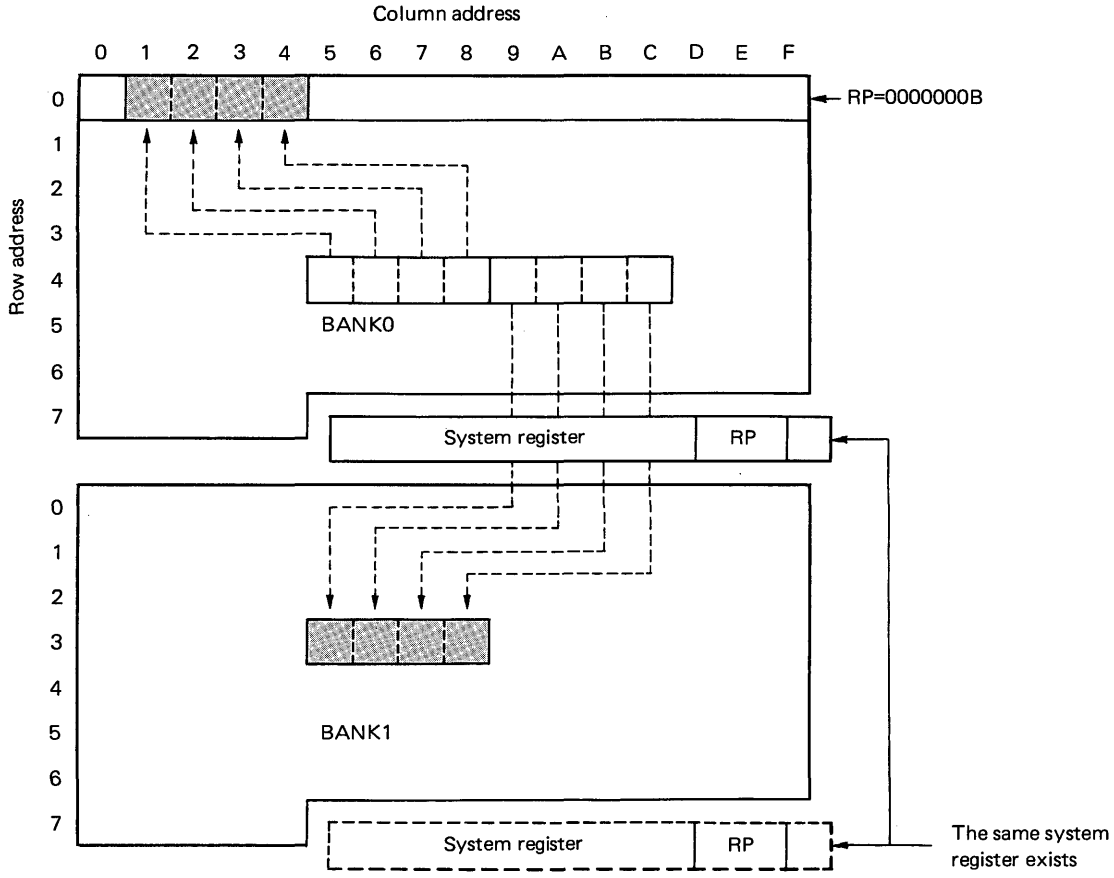
Example 4: Changing row address of general register BANK0 is selected and the general register is stored in row address 0H of BANK0.

```
R001  MEM  0.01H ;  Symbol definition
R002  MEM  0.02H ;
R003  MEM  0.03H ;
R004  MEM  0.04H ;
R005  MEM  0.05H ;
R006  MEM  0.06H ;
R007  MEM  0.07H ;
R008  MEM  0.08H ;
M045  MEM  0.45H ;
M046  MEM  0.46H ;
M047  MEM  0.47H ;
M048  MEM  0.48H ;
M049  MEM  0.49H ;
M04A  MEM  0.4AH ;
M04B  MEM  0.4BH ;
M04C  MEM  0.4CH ;
LD    R001, M045
LD    R002, M046
LD    R003, M047
LD    R004, M048
;  ①
MOV   RPH,  #0001B ;  Transfers 0001011B to the general register pointer. That is, 0001011B is set in row
MOV   RPL,  #0110B ;  address 3H of BANK1.
LD    R005, M049
LD    R006, M04A
LD    R007, M04B
LD    R008, M04C
```

As shown in Fig. 7-6, the above instruction transfers 4 nibbles each of the contents of the data memory range M045 to M04C which consists of 8 nibbles in BANK0 to BANK0 and BANK1. In this case, when the general register is fixed and when only 0 exists in BANK0, an instruction for storing data in the data memory after transferring all the 8 nibbles to the general register is required in the program as shown below.
However, by changing the row address of the general register using a general register pointer as shown in the above instruction, the operation can be terminated by the LD instruction only.

```
M135  MEM  1.35H ;  Symbol definition
M136  MEM  1.36H ;
M137  MEM  1.37H ;
M138  MEM  1.38H ;
LD    R005, M049
LD    R006, M04A
LD    R007, M04B
LD    R008, M04C
BANK1             ;  Assembler (AS17K) built-in macro instruction BANK ← 1
ST    M135, R005
ST    M136, R006
ST    M137, R007
ST    M138, R008
```

Fig. 7-6 Example of changing row addresses of a general register



7.4 NOTES ON USING A GENERAL REGISTER

Sections **7.4.1** to **7.4.4** describe notes on using a general register.

7.4.1 General Register Address Specification

As shown below, when general register address is specified directly to the operand of the instruction in Assembler (AS17K) of 17K series, an error occurs. This error routine is incorporated for reducing debugging factors at program fixing etc., in the same way as for data memory.

Case where an error occurs:

```
LD    04H,  32H   ; A general register address and a data memory address are entered directly using
                  ; numeric values.
```

Case where an error does not occur:

```
R004  MEM   0.04H ; Symbol of address 04H is defined in R004 as the memory type.
M032  MEM   0.32H ;
LD    R004, M032  ;
```

7.4.2 Row Addresses of a General Register

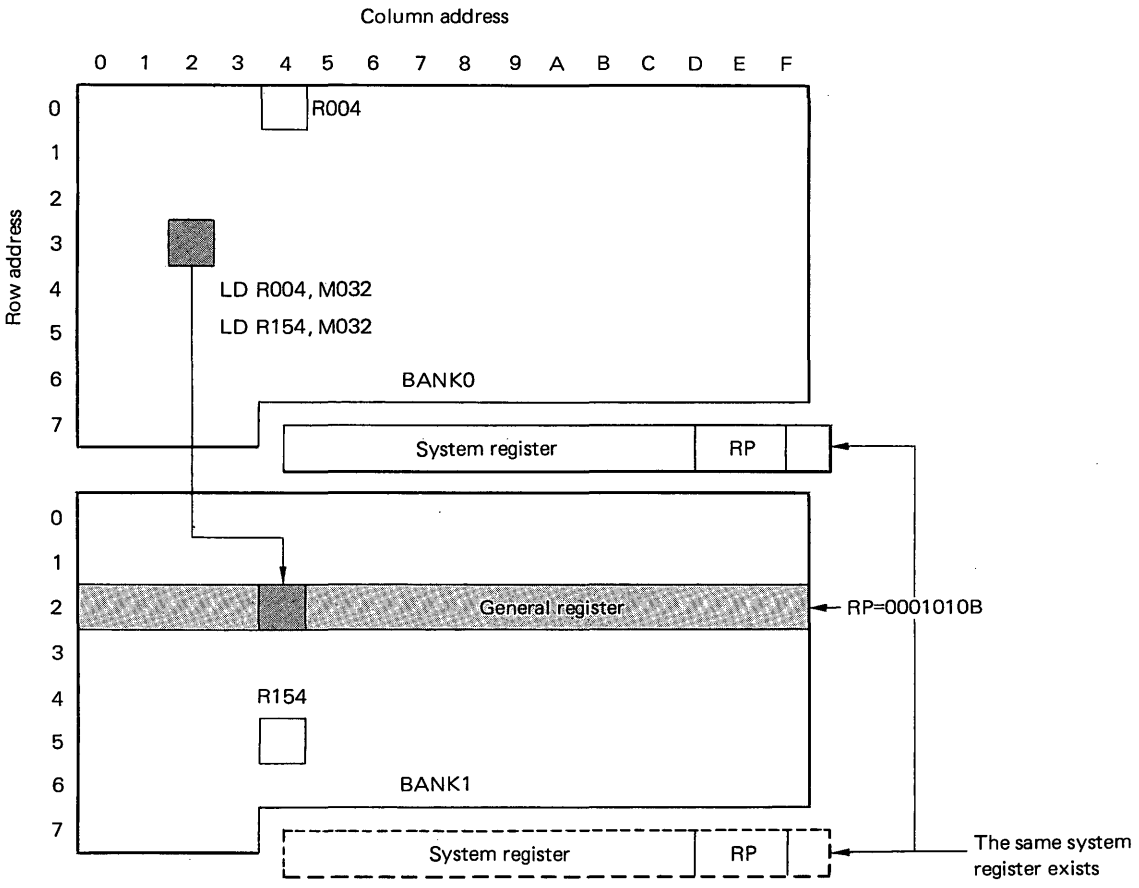
Since a row address of a general register is determined by a general register pointer, the bank of the address specified by operand "r" of the instruction and the row address are ignored.

Example: General register row address specification example

```
R004  MEM  0.04H   ;
R154  MEM  1.54H   ;
M032  MEM  0.32H   ;
MOV   RPH, #0001B ;
MOV   RPH, #0100B ;  RP ← 0001010B
;  ①
LD    R004, M032
;  ②
LD    R154, M032
```

When the above instruction is executed, both ① and ② transfer the content of data memory M032 (address 32H of BANK0) to address 24H of BANK1 which is the general register.
That is, if a general register address is specified in instructions ① and ②, the banks of R004 and R154 and row address are ignored and only address 4H of the column address becomes valid.

Fig. 7-7 General register row address specification example



7.4.3 Operation of a General Register and Immediate Data

No operation instruction between a general register and immediate data is available when a general register is used. That is, to execute an operation instruction between the data memory specified in the general register and immediate data, the data memory must be handled as a general register instead of data memory.

Example: Operation example of a general register and immediate data
BANK0 is selected.

```
  R125  MEM   1.25H    ;
  M005  MEM   0.05H    ;
;  ①
  MOV   RPH,  #0001B ;  Sets the general register to row address 2H of BANK1.
  MOV   RPL,  #0100B ;
;  ②
  ADD   R125, #3     ;
;  ③
  ADD   M005, #3     ;
```

In the above instructions, ② adds immediate data ③ to the data memory of address 25H of BANK0 and ③ adds data memory 3 of address 05H of BANK0.
That is, the general register is set in row address 2H of BANK1 in ① , and R125 which is the operand of the instruction ② is handled as data memory, not a general register.
Consequently, to perform addition to address 25H of BANK1 which is a general register in instruction ② , the program must be coded as follows:

```
BANK1              ;  Assembler (AS17K) built-in macro instruction
ADD   R125, #3
```

7.4.4 Operation to Data Memory which cannot Be Specified for a General Register

Since data memory BANK0 and data memory BANK1 can be specified for a general register, the bank must be switched for operation to data memory of BANK2 and data memory of BANK3 as shown in Example 1.

In this case, a window register in the system register can be used by specifying the system register and the general register.

Example 2 shows this.

Example 1:

Add the content of address 33H of BANK0 to the content of address 33H of BANK2.

```
R000  MEM  0.00H   ;
R033  MEM  0.33H   ;
M233  MEM  2.33H   ;
MOV   RPH, #0000B ;
MOV   RPL, #0000B ;
BANK2
LD    R000, M233   ;
BANK0
ADD   R000, R033   ;
BANK2
ST    M233, R000   ;
```

Example 2:

Example where a window register is used

```
R000  MEM  0.00H   ;
R033  MEM  0.33H   ;
M233  MEM  2.33H   ;
; ①
MOV   RPH, #0000B ;
MOV   RPL, #1110B ;
; ②
LD    WR,  R033    ;
BANK2
; ③
ADD   WR,  M233    ;
ST    M233, WR     ;
```

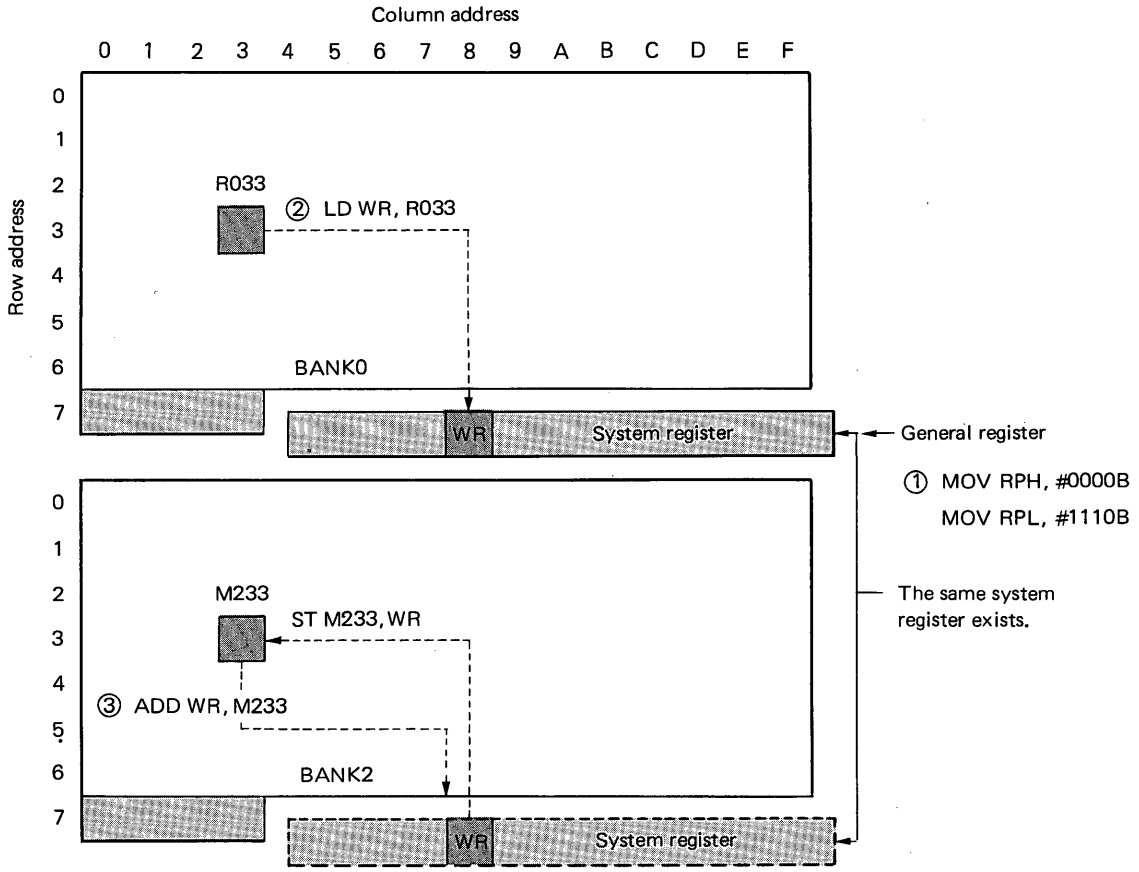
When program shown in Example ② are executed, row address 7H of BANK0, that is, system register is specified for the general register in ①.

In instruction ②, the content of data memory R033 (address 33H of BANK0) is transferred to the window register.

In this case, since the window register of BANK0 is a system register, the same content is also viewed from address 78H of BANK2 as shown in Fig. 7-7. That is, the window register which is the data memory in BANK2 can also be a general register.

Consequently, the result same as that of Example ① can be produced by executing an addition instruction in ③.

Fig. 7-8 Example of the operation performed using a window register



8. ALU (ARITHMETIC LOGIC UNIT) BLOCK

ALU performs 4-bit data arithmetic operation, logical operation, bit checking, comparison checking, and rotation processing.

8.1 STRUCTURE OF AN ALU BLOCK

Fig. 8-1 shows the structure of an ALU block.

As shown in Fig. 8-1, an ALU block consists of an ALU main unit which performs 4-bit data processing, temporary storage registers A and B which are the peripheral circuits of ALU, status flip/flop which controls the ALU status, and decimal compensating circuit used when decimal operation is used.

The status flip/flop consists of zero flag FF, carry flag FF, compare flag FF, and BCD flag FF as shown in Fig. 8-2.

The status flip/flop corresponds to a zero flag (Z), carry flag (CY), compare flag (CMP), and BCD flag (BCD) of a program status word (PSWORD: addresses 7EH and 7FH) in the system register on a one to one basis.

Fig. 8-1 Structure of an ALU block

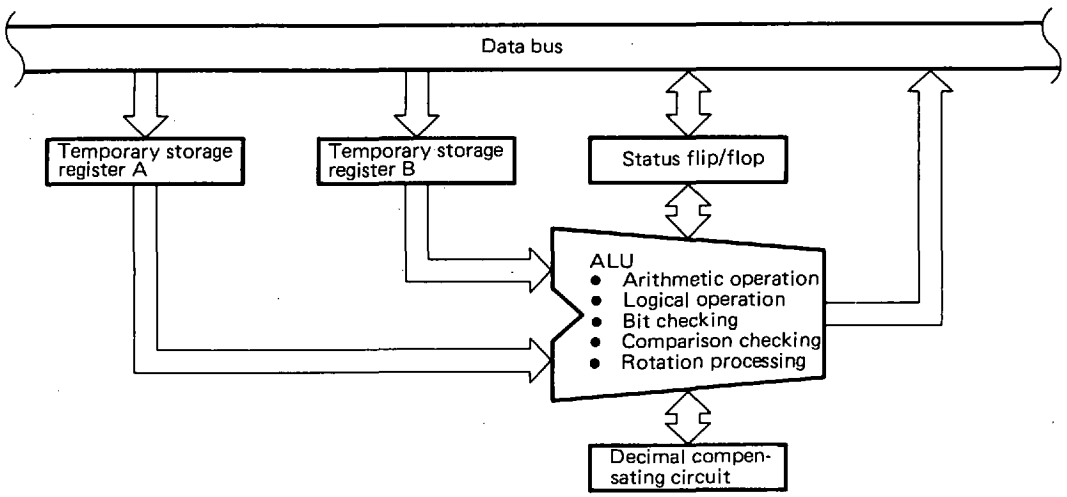
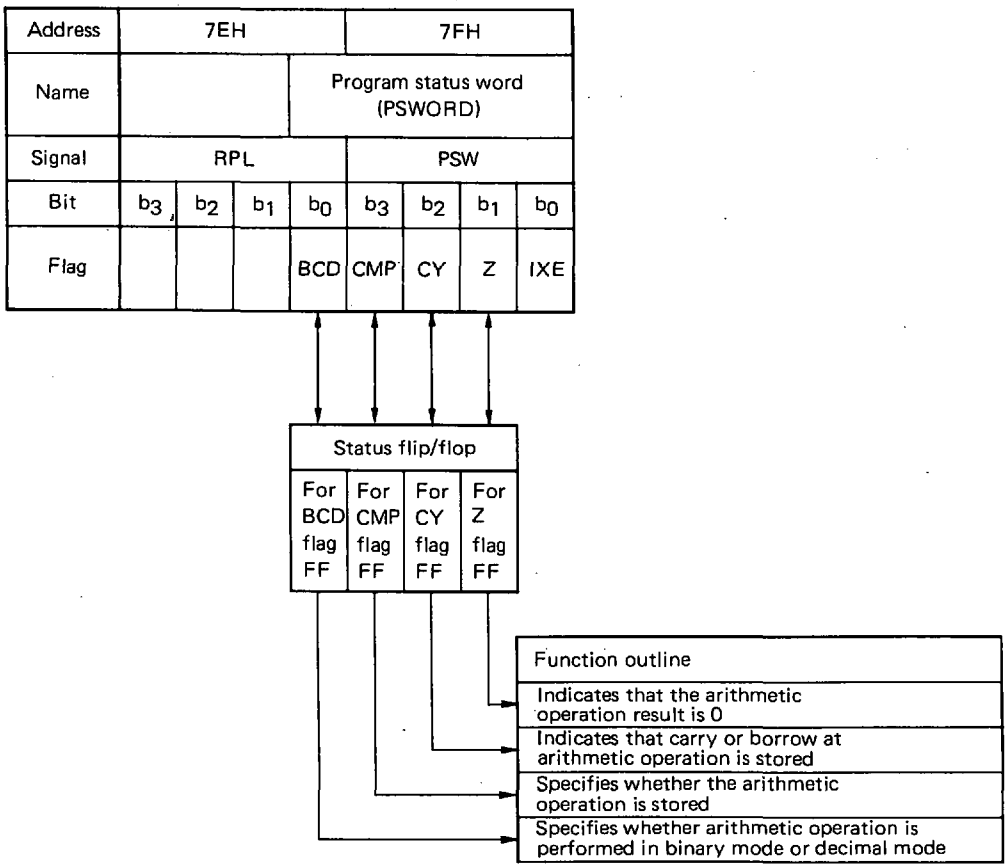


Fig. 8-2 Function of a program status word



8.2 FUNCTION OF AN ALU BLOCK

An ALU block performs arithmetic operation, logical operation, bit checking, comparison checking, and rotation processing according to the instruction specified by the program.

Table 8-1 lists each of the operation, checking, and rotation processing instructions.

By executing each instruction listed in Table 8-1, operation, checking, or rotation processing of 4 bit units or decimal arithmetic operation of one column can be executed by one instruction.

8.2.1 Function of ALU

Arithmetic operations include addition and subtraction.

Arithmetic operation between the content of a general register and the content of data memory or arithmetic operation between the content of data memory and immediate data can be performed.

An operation of 4 bits in binary mode and operation of one bit in decimal mode are possible.

Logical operations include a logical product (Logical AND), logical sum (Logical OR), and logical exclusive sum (Logical Exclusive OR).

Logical operation can be performed between the content of a general register and data memory or between the content of data memory and immediate data.

Bit checking checks bits '0' or '1' in the 4-bit data of the data memory.

Comparison checking compares the content of data memory and immediate data and performs checking based on "equal to", "not equal to", "greater than", and "less than".

Rotation processing shifts 4-bit data of the general register by 1 bit to the direction of low-order bits. (Rotate clockwise)

See Sections **8.3** to **8.6** for details.

8.2.2 Functions of Temporary Registers A and B

Temporary registers A and B are required for processing 4-bit data collectively and temporarily store data which is to be processed and data which is used for processing data.

These registers are automatically used at execution of an instruction and cannot be controlled by a program.

8.2.3 Function of Status Flip/Fop

Status flip/flop stores the status of ALU operation control and data which was processed.

Since status flip/flop corresponds to each flag of a program status word (PSWORD) of a system register on a one to one basis, status flip/flop can be operated concurrently by operating the system register.

Each flag of a program status word is described below.

(1) Z flag

The Z flag is set (1) when the result of the arithmetic operation is 0000B and reset (0) when the result is other than 0000B. However, the condition in which the flag is set (1) varies depending on the CMP flag status as described in (a) and (b).

(a) CMP flag = 0

If the operation result is 0000B, the flag is set (1) and if the result is not 0000B, the flag is reset (0).

(b) CMP flag = 1

If the operation result is 0000B, the previous status is retained and if the result is not 0000B, the flag is reset (0).

(2) CY flag

If Carry or Borrow occurs as a result of the arithmetic operation, the flag is set (1) and if Carry or Borrow does not occur, the flag is reset (0).

When operation is performed together with Carry or Borrow, the content of the lowest bit is transferred to the CY flag. When rotation processing (rotation clockwise) is performed, the content of the CY flag is transferred to the highest bit (b_3) of the data and the content of the lowest bit (b_0) of the data is transferred to the CY flag.

The flag content remains unchanged in operations other than arithmetic operation and rotation processing.

(3) CMP flag

When the CMP flag is set (1), the result of the arithmetic operation executed is not stored in the general register or data memory.

When a bit checking instruction is executed, the CMP flag is reset (0).

This flag does not influence comparison checking, logical operation, or rotation processing.

(4) BCD flag

When a BCD flag is set (1), all the arithmetic operations are performed in decimal mode.

When the flag is reset (0), arithmetic operations are performed in binary mode.

This flag does not influence logical operation, bit checking, comparison checking, and rotation checking. The values of these flags (Z, CY, CMP, and BCD) can be changed by operating the program status word (PSWORD) directly.

In this case the status flip/flop which corresponds to the flag which was changed also changes accordingly.

8.2.4 Function of a Decimal Compensating Circuit

When the BCD flag is set (1) at arithmetic operation, the arithmetic operation result is converted to decimal digits by a decimal compensating circuit.

See Section **8.3, "Arithmetic Operation (Addition and Subtraction of Binary and Decimal Data)"** for decimal operation and decimal compensating circuit.

Table 8-1 ALU processing instructions (1/3)

ALU function		Instruction	Operation	Explanation
Arithmetic operation	Addition	ADD r, m	(R) ← (R) + (M)	Adds the contents of general data and data memory and stores the result to a general register.
		ADD m, #i	(M) ← (M) + i	Adds the contents of data memory and immediate data and stores the result in data memory.
		ADDC r, m	(R) ← (R) + (M) + (CY)	Adds the contents of general register and data memory together with CY flag and stores the result in the general register.
		ADDC m, #i	(M) ← (M) + i + (CY)	Adds the contents of data memory and immediate data together with CY flag and stores the result in data memory.
	Subtraction	SUB r, m	(R) ← (R) − (M)	Subtracts the content of data memory from the content of a general register and stores the result in the general register.
		SUB m, #i	(M) ← (M) − i	Subtracts immediate data from the content of data memory and stores the result in data memory.
		SUBC r, M	(R) ← (R) − (M) − (CY)	Subtracts the content of data memory and CY flag from the content of the general register and stores the result in the general register.
		SUBC m, #i	(M) ← (M) − i − (CY)	Subtracts immediate data and CY flag from the content of data memory and stores the result in data memory.
Logical operation	Logical sum	OR r, m	(R) ← (R) OR (M)	Executes OR for the contents of the general register and data memory and stores the result in the general register.
		OR m, #i	(M) ← (M) OR i	Executes OR for the contents of the data memory and immediate data and stores the result in data memory.
	Logical product	AND r, m	(R) ← (R) AND (M)	Executes AND for the contents of general register and data memory and stores the result in the general register.
		AND m, #i	(M) ← (M) AND i	Executes AND for the contents of data memory and immediate data and stores the result in data memory.
	Logical exclusive sum	XOR r, m	(R) ← (R) XOR (M)	Executes XOR for the eontents of the general register and data memory and stores the result in the general register.
		XOR m, #i	(M) ← (M) XOR i	Executes XOR for the contents of the data memory and immediate data and stores the result in data memory.

Table 8-1 ALU processing instruction (2/3)

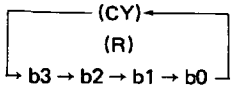
ALU function		Instruction	Operation	Explanation
Bit checking	True	SKT m, #n	SKIP if (M) $\geqq$ n	If all the bits specified by n are True (1) among the contents of the data memory, processing is skipped. The result is not stored.
	False	SKF m, #n	SKIP if (M) $\leqq$ (n XOR 1111B)	If all the bits specified by n are False (0) among the contents of the data memory, processing is skipped. The result is not stored.
Comparison checking	Equal to	SKIP if (M) = i	SIPT if (M) = i	Skips processing when the content of the data memory is equal to the immediate data. The result is not stored.
	Not equal to	SKNE m, #i	SKIP if (M) = i	Skips processing when the content of the data memory is not equal to the immediate data. The result is not stored.
	Greater than	SKGE m, #i	SKIP if (M) $\geqq$ i	Skips processing when the content of the data memory is greater than the immediate data. The result is not stored.
	Less than	SKLT m, #i	SKIP if (M) < i	Skips processing when the content of the data memory is less than the immediate data. The result is not stored.
Rotation	Clockwise rotation	RORC r		The content of the general register is rotated clockwise together with the CY flag and the result is stored in the general register.

Table 8-1 ALU processing instruction (3/3)

	Operation difference according to the program status word (PSWORD)						
ALU function	Value of BCD flag	Value of CMP flag	Operation	Operation of CY flag	Operation of Z flag	Qualification by IXE flag	Others
Arithmetic operation	0	0	Binary operation The result is stored.	Set occurred in Carry or Borrow Reset if Carry or Borrow does not occur	Set when the operation result is 0000B. Reset when the result is not 0000B.	Yes	
	0	1	Binary operaton The result is not stored.		The status is maintained when the operation result is 0000B. Reset when the result is not 0000B.		
	1	0	Decimal operation The result is stored.		Set when the operation result is 0000B. Reset when the result is not 0000B.		
	1	1	Decimal operation The result is not stored.		The status is maintained when the operation result is 0000B. Reset when the result is not 0000B.		
Logical operation	Any value (maintained)	Any value (maintained)	Unchanged	Previous status maintained	Previous status maintained	Yes	
Bit checking	Any value (maintained)	Reset	Unchanged	Previous status maintained	Previous status maintained	Yes	
Comparison checking	Any value (maintained)	Any value (maintained)	Unchanged	Previous status maintained	Previous status maintained	Yes	
Rotation	Any value (maintained)	Any value (maintained)	Unchanged	Value of general register b_0	Previous status maintained	No	

8.2.5 ALU Block Processing Procedure

If an arithmetic operation instruction, logical operation instruction, bit checking instruction, comparison checking instruction, or rotation processing instruction is executed in a program, the data to be operated upon or checked and data for processing or being processed are stored in temporary storage registers A and B respectively.

The data to be processed is the content of the general register whose address is specified by the first operation of each statement or content of the data memory. The data consists of 4 bits.

The data to be processed is the content of the general register or data memory whose address is specified by the first operand of each instruction.

The data for processing data is the content of the data memory whose address is specified by the second operation of each instruction or immediate data which is directly specified by the second operand. The data consists of 4 bits.

For instance, in the following addition instruction, the data to be processed is the content of the general register whose address is specified by the first operand r and the data used for processing is the content of the data memory whose address is specified by the second operand m.

```
ADD r, m
    |  └── Second operand
    └──── First operand
```

In the following addition instruction, the data to be processed is the content of the data memory whose address is specified by m and the data used for processing is the immediate data specified by #i.

```
ADD m, #i
```

Rotation processing instruction, RORCr, requires only the data to be processed because the processing method is predetermined and the data is the content of the general register whose address is specified by r.

The data stored in temporary storage registers A and B is used for execution, in ALU, by an arithmetic operation, logical operation, bit checking, comparison checking, or rotation processing according to the instruction.

When the instruction which was executed is an arithmetic operation, logical operation, or rotation processing, the data processed by ALU is stored in the general register or data memory specified by the first operand, and processing is terminated according to the result of processing in ALU.

When the executed instruction is for bit checking or comparison checking, the following instruction in the program is skipped (the next instruction is executed as a No Operation Instruction (NOP)) and operation is terminated.

The following items (1) to (4) must be noted for the operation of an ALU block.

(1) An arithmetic operation is influenced by status flip/flop, that is, the CMP flag and BCD flag of the program status word.
(2) A logical operation is not influenced by the CMP flag nor the BCD flag of the program status word. It does not influence the Z flag and CY flag.
(3) Bit checking resets the CMP flag of the program status word.
(4) When the IXE flag of the program status word is set (1), an arithmetic operation, logical operation, bit checking and comparison checking receive address qualification by the index register (IX).

See **9, "System Register (SYSREG)"** for address qualification by index registers.

8.3 ARITHMETIC OPERATION (ADDITION AND SUBTRACTION IN BINARY MODE OR DECIMAL MODE)

As listed in Table 8-1, arithmetic operation instruction includes an addition instruction, "ADD", and a subtraction instruction, "SUB".

The "ADDC instruction" which adds data together with Carry and the "SUBC instruction" which subtracts data together with Borrow are also available.

Instructions, "ADD", "ADDC", "SUB", and "SUBC" are classified into those for addition and subtraction between a general register and data memory, and those for addition and subtraction between data memory and immediate data. This is determined by the value coded in the operand of each instruction.

If the operand is "r, m" the instruction is for addition or subtraction between a general register and data memory and if the operand is "m, #i", the instruction is for addition or subtraction between data memory and immediate data.

Arithmetic operation instructions are influenced by a program status word.

A binary operation or decimal operation is performed according to the BCD flag of the program status word and the CMP flag can be used for indicating that the operation result is not to be stored.

When an Index Enable flag is set, the address is qualified by the index register. See **9, "System Register (SYSREG)"** for address qualification by an index register.

Sections **8.3.1** to **8.3.4** describe each arithmetic operation instruction and program status word (PSWORD). Section **8.3.5** provides notes on using arithmetic operations.

8.3.1 Addition and Subtraction when CMP Flag=0 and BCD Flag=0

Binary addition or subtraction are performed.

The result is stored in a general register or data memory.

The CY flag is set (1) when the operation result exceeds 1111B (Carry occurred) or the result is less than 0000B (Borrow occurred). In other cases, the flag is reset.

When the operation result is 0000B, the Z flag is set (1) regardless of the occurrence of Carry or Borrow and if the result is not 0000B, the flag is reset (0).

Example 1:

```
MOV  R1,  #1111B  ;  Transfers 1111B to general register R1
MOV  M1,  #0001B  ;  Transfers 0001B to data memory M1
ADD  R1,  M1      ;  Adds R1 and M1
```

In this case, R1+M1 is calculated as follows.

```
  1111B  ... Contents of R1
+0001B   ... Contents of M1
-------
1 0000B
~
Carry
```

The addition result, 0000B, is written to the content of R1 and the CY flag is set (1). The content of M1 remains unchanged. Since the operation result is 0000B, the Z flag is set (1). When the contents of R1 and M1 are added, the CY flag is reset (0) unless Carry is output.

Example 2:

```
MOV  M1,  #1010B  ;  Transfers 1010B to data memory M1
ADD  M1,  #0101B  ;  Adds immediate data 0101B to M1
```

In this case, M1+0101B is calculated as follows.

```
  1010B ... Contents of M1
 +0101B ... Immediate data
-------
 0 1111B
 ~
 Carry
```

Consequently, 1111B is written to the content of M1 and the CY flag and Z flag are reset.

Example 3:

```
MOV  R1,  #1000B  ;  Writes 1000B to general register R1.
MOV  M1,  #1111B  ;  Writes 1111B to data memory M1.
;  ①
ADD  M1,  #0001B  ;  Adds immediate data 0001B to M1.
;  ②
ADDC R1, M1       ;  Adds R1 and M1 together with Carry.
```

In ① the calculation is performed as follows.

```
  1111B ... Content of M1
 +0001B ... Immediate data
-------
 1 0000B
 ~
 Carry
```

Consequently, 0000B is written to M1 and the CY flag and Z flag are set (1).

In ②, the following calculation is performed.

```
  1000B ... Content of R1
  0000B ... Content of M1
+    1   ... Content of CY flag
-------
 0 1001B
 ~
 Carry
```

That is, at execution of the ADDC instruction, addition is performed together with the content of the CY flag and the CY flag is rewritten according to the Carry output performed as a result.

Example 4:

```
MOV  R1,  #0000B  ;  Writes 0000B to general register R1.
MOV  M1,  #1000B  ;  Writes 1000B to data memory M1.
SUB  R1,  M1      ;  Subtract M1 from R1.
```

In this case, R1-M1 is calculated as follows.

```
  0000B ... Content of R1
 −1000B ... Content of M1
 ------
 1 1000B
  Borrow
```

In this case, subtraction result 1000B is written to R1. Since Borrow occurred, the CY flag is set (1).
Carry occurring by an addition instruction and Borrow occurring by a subtraction instruction are managed by the same CY flag.

Example 5:

```
  MOV  R1,  #0000B  ;
  MOV  M1,  #0000B  ;
;  ①
  SUB  M1,  #0001B  ;
;  ②
  SUBC R1,  M1      ;
```

In this case, ① and ② are calculated as follows.

```
①  0000B ... Content of M1
  −0001B ... Immediate data
  ------
  1 1111B
   Borrow
```

```
②  0000B ... Content of R1
    1111B ... Content of M1
  −    1  ... Content of CY flag
  ------
  1 0000B
   Borrow
```

Consequently, the operation results are as follows;
R1=0000B, MI=1111B, CY flag=1, and Z flag=1.

8.3.2 Addition and Subtraction when CMP Flag=1 and BCD Flag=0

Binary addition or subtraction is performed.

However, since the CMP flag is set (1), the operation result is not stored in the general register or data memory.

When Carry or Borrow occurs because of the operation result, the CY flag is set (1) and if Carry or Borrow does not occur, the CY flag is reset (0).

If the operation result is 0000B, the previous state of Z flag is maintained and if the result is not 0000B, the flag is reset (0).

Example 1:

```
   MOV  PSW, #1000B  ;  Sets the CMP flag (writes to the program status word).
   MOV  R1,  #1111B  ;
   MOV  M1,  #1111B  ;
;  ①
   ADD  R1,  M1      ;
;  ②
   SUB  R1,  M1      ;
   MOV  PSW, #1010B  ;  Sets the CMP flag and Z flag.
;  ③
   SUB  R1,  M1      ;
```

In this case, the following calculation is performed in instruction ①.

```
  1111B ... Content of R1
 +1111B ... Content of M1
 ------
 11110B
 ~
 Carry
```

However, since the CMP flag is set (1), the operation result is not stored in R1. Since Carry occurs, the CY flag is set (1). Since the operation result is not 0000B, the Z flag is reset. In ②, since the content of R1 and M1 are the same as for ①, the CY flag is reset (0). Since the operation result is 0000B, the previous status, 0, is maintained for the Z flag.

Operation of ③ is the same as for ②. However, status ① is maintained for the Z flag because the Z flag has been set (1) in advance.

When the CMP flag is set (1), the arithmetic operation result is not stored and as only the statuses of the CY flag and Z flag change, the flag is useful for comparing data of 5 bits or more.

Example 2:

```
   MOV  PSW, #1010B      ;  Sets (1) the CMP flag and Z flag.
;  ①
   SUB  M1,  #0001B(1H) ;
;  ②
   SUBC M2,  #0010B(2H) ;
;  ③
   SUBC M3,  #0011B(3H) ;
```

Since the CMP flag has been set (1) in this case, the operation result is not stored. Consequently, the contents of M1, M2, and M3 do not change even if instructions ①, ②, and ③ are executed.
Since the Z flag is set first (1), the Z flag is kept set (1) if all the operation results of ①, ②, and ③ are 0000B and the flag is reset (0) if at least one of the operation results is not 0000B.
The CY flag is set when the contents of 12 bits of M3, M2, and M1, are less than 001100100001B (321H).
Consequently, 12-bit data of M3, M2 and M1 and 12-bit data of 321H can be compared by testing the Z flag and CY flag at termination of instructions ①, ②, and ③.
The results are shown below.

```
If Z=1, CY=0,   M3, M2, M1=321H
          ↑
       Always 0
If Z=0, CY=0,   M3, M2, M1 > 321H
If Z=0, CY=1,   M3, M2, M1 < 321H
```

In example 2, contents of the general register and contents of data memory can be compared by using the SUB r, m and SUBC r, m instructions.

8.3.3 Addition and Subtraction when CMP Flag=0 and BCD Flag=1

Decimal arithmetic operation is performed.

The operation result is stored in the general register or data memory.

If the operation result exceeds 1001B (9D) or is less than 0000B (0D), the CY flag is set (1), and if the operation result is between 0000B (0D) and 1001B (9D), the CY flag is reset (0).

The Z flag is set (1) when the operation result is 0000B and reset (0) when the operation result is not 0000B (0D).

The decimal operation is performed by converting the result of the operation performed in binary mode to decimal data using a decimal compensating circuit. See Table 8-2 for the binary/decimal conversion.

To execution a decimal operation correctly, the following conditions must be satisfied.

(1) The result of addition is 0-19D.

(2) The result of subtraction is 0-9D or −10D to −1.

Value 0-19D is the value taking the CY flag into consideration and indicates $\underset{\sim}{0}$, 0000B-$\underset{\sim}{1}$, 0011B in binary mode.
CY CY

Value−10D to −1 indicates $\underset{\sim}{1}$, 0110B-$\underset{\sim}{1}$, and 1111B in the same way.
CY CY

When decimal operation is performed outside of the conditions described above, the CY flag is set (1) and data higher than 1010B (0AH) is output as the operation result.

Table 8-2 Decimal compensation data

Operation output data		Data output after decimal compensation			
		Addition		Subtraction	
CY flag	Data	CY flag	Data	CY flag	Data
0	0000B	0	0000B	0	0000B
0	0001B	0	0001B	0	0001B
0	0010B	0	0010B	0	0010B
0	0011B	0	0011B	0	0011B
0	0100B	0	0100B	0	0100B
0	0101B	0	0101B	0	0101B
0	0110B	0	0110B	0	0110B
0	0111B	0	0111B	0	0111B
0	1000B	0	1000B	0	1000B
0	1001B	0	1001B	0	1001B
0	1010B	1	0000B	1	1100B
0	1011B	1	0001B	1	1101B
0	1100B	1	0010B	1	1110B
0	1101B	1	0011B	1	1111B
0	1110B	1	0100B	1	1100B
0	1111B	1	0101B	1	1101B
1	0000B	1	0110B	1	1110B
1	0001B	1	0111B	1	1111B
1	0010B	1	1000B	1	1100B
1	0011B	1	1001B	1	1101B
1	0100B	1	1110B	1	1110B
1	0101B	1	1111B	1	1111B
1	0110B	1	1100B	1	0000B
1	0111B	1	1101B	1	0001B
1	1000B	1	1110B	1	0010B
1	1001B	1	1111B	1	0011B
1	1010B	1	1100B	1	0100B
1	1011B	1	1101B	1	0101B
1	1100B	1	1010B	1	0110B
1	1101B	1	1011B	1	0111B
1	1110B	1	1100B	1	1000B
1	1111B	1	1101B	1	1001B

Example 1:

```
  MOV  M1,  #0111B (7)  ;
  MOV  RPL, #0001B      ;  Sets BCD flag (the BCD flag is allocated in b0 of system RPL0).
  MOV  PSW, #0000B      ;  Resets the CMP flag, CY flag, and Z flag.
;  ①
  ADD  M1,  #1001B (9)  ;  7 + 9
;  ②
  SUB  M1,  #0111B (7)  ;  6 – 7
```

In this case, ① is calculated as follows.

```
  0111B  ... Content of M1
+1001B   ... Immediate data
 10000B  ... Binary addition result
 Carry
   ↓          Converts data based on the binary/decimal compensation shown in Table 8-2
 10110B  ... M1 storage data
 Carry
```

That is, the CY flag is set and 0110B (6) is stored in M1. If the CY flag is assumed to have a weighting of 10, decimal operation of 7 + 9 = 16 is assumed.
In ②, the following calculation is carried out.

```
  0110B  ... Content of M1
–0111B   ... Immediate data
 11111B  ... Binary subtraction result
 Borrow
   ↓          Binary decimal compensation
 11001B  ... M1 storage data
```

That is, since 6 is stored in M1 in ①, operation of 6 – 7 was performed, producing 9 as the result and the CY flag is set.

Example 2:

```
MOV  M1,  #0101B (5) ;
MOV  M2,  #0110B (6) ;
MOV  M3,  #0111B (7) ;
MOV  RPL, #0001B     ;  Sets (1) BCD flag.
MOV  PSW, #0000B     ;  Resets (0) the CMP flag, CY flag, and Z flag.

;  ①
SUB  M1,  #0111B (7) ;
;  ②
SUBC M2,  #0110B (6) ;
;  ③
SUBC M3,  #0101B (5) ;
```

In this case, ①, ②, and ③ are calculated as follows.

```
①  0101B  ... Content of M1
  –0111B  ... Immediate data
  ------
   11110B
   Borrow
     |          Binary/decimal compensation
     ↓
   10000B (8) ... M1 storage data
   Borrow

②  0110B  ... Content of M2
  –0110B  ... Immediate data
  ------
   11111B  ... CY flag
   Borrow
     |          Binary/decimal compensation
     ↓
   11001B (9) ... M2 storage data
   Borrow

③  0111B  ... Content of M3
  –0101B  ... Immediate data
  ------
   00001B  ... CY flag
   Borrow
     |          Binary/decimal compensation
     ↓
   00001B (1) ; M3 storage data
```

That is, immediate data 567 is subtracted from 765 which is stored in M3, M2, and M1, producing 198 as the result.

Example 3:

```
  MOV   M1,  #1001B  ;
  MOV   RPL, #0001B  ;  Sets (1) the BCD flag.
  MOV   PSW, #0000B  ;  Resets (0) the CMP flag, CY flag, and Z flag.
; ①
  ADDC  M1,  #1010B  ;
; ②
  ADDC  M1,  #1010B  ;
```

In this case, ① is calculated as follows.

```
  1001B (9)  ... Content of M1
 +1010B (10) ... Immediate data
 1 0011B ....... CY flag
 Carry
   ↓                 Binary/decimal conversion
 1 1001B ....... Operation result
 Carry
```

That is, operation 9 + 10 = 9 is performed. If the CY flag is taken into account, decimal operation 9 + 10 = 19 is assumed. However, in ②, the following calculation is carried out.

```
  1001B (9)  ... Content of M1
 +1010B (10) ... Immediate data
 1 0100B ....... CY flag
 Carry
   ↓                 Binary decimal conversion
 1 1110B ....... Operation result
 Carry
```

That is, since the CY flag has been set (1), the operation result exceeded 19 and correct decimal operation cannot be performed.

8.3.4 Addition and Subtraction when CMP Flag=1 and BCD Flag=1

A decimal arithmetic operation is performed.

The operation result is not stored in the general register or data memory.

That is, operation when CMP flag = 1 and operation when BCD flag = 1 are performed concurrently.

Example:

```
   MOV   RPL, #0001B   ;  Sets (1) the BCD flag.
   MOV   PSW, #1010B   ;  Sets (1) the CMP flag and Z flag and resets (0) the CY flag.
;  ①
   SUB   M1,  #0001B   ;
;  ②
   SUBC  M2,  #0010B   ;
;  ③
   SUBC  M3,  #0011B   ;
```

In this case, the contents of 12 bits in M3, M2, and M1 and immediate data 321 can be compared in decimal mode in ①, ②, and ③.

8.3.5 Notes on Using Arithmetic Operations

Note that the result of an arithmetic operation is stored in the program status word when an arithmetic operation is performed for a program status word.

That is, although the CY flag and Z flag in the program status word are normally reset by the arithmetic operation result, the arithmetic operation result is stored if an arithmetic operation is performed for the program status word itself and thus, Carry, Borrow, or Zero cannot be checked.

However, when the CMP flag is set (1), the CY flag and Z flag are set or reset normally since no arithmetic operation result is stored.

The examples are shown below.

Example 1:

```
MOV   PSW, #0110B
ADD   PSW, #1010B
```

In this case, the results are calculated as follows.

```
  0110B ... Content of PSW
+1010B ... Immediate Data
1 0000B
 Carry
```

Although the CY flag and Z flag must be set as a result, the operation result 0000B is stored in PSW since the CMP flag is "0".

Example 2:

```
MOV   PSW, #1010B
ADD   PSW, #1000B
```

In this case, the results are calculated as follows.

```
  1010B ... Content of PSW
+1000B ... Immediate data
1 0010B
 Carry
```

Since the CMP flag is set (1), operation result 0010B is not stored in the PSW.
Consequently, the CY flag of PSW is set (1) and the Z flag is reset (0). That is, 1100B is stored in PSW.

8.4 LOGICAL OPERATION

As shown in Table 8-1, logical sum (Logical OR), logical product (Logical AND), and logical exclusive sum (Logical Exclusive OR) are allowed for logical operations. Table 8-3 lists truth values of the logical sum, logical product, and logical exclusive sum.

Logical operation instructions are classified into these three types and the "OR instruction", "AND instruction", and "XOR instruction" are used respectively.

The "OR", "AND", and "XOR" instructions are classified into logical operations between a general register and data memory, and logical operations between data memory and immediate data. The type is determined by the value "r, m" or "m, #i" entered in the operand of the instruction in the same way as for the arithmetic operation.

Logical operation is not influenced by the BCD flag or CMP flag of the program status word.

The CY flag and Z flag do not impose any influence on the logical operation.

When the Index Enable flag is set (1), address qualification is performed from the index register. See **9, "System Register (SYSREG)"** for address qualification by an index register.

Sections **8.4.1** and **8.4.2** describe logical sum, logical product, and logical exclusive sum.

Table 8-3 Truth value of logical operation

Logical sum A OR B = C			Logical product A AND B = C			Logical exclusive sum A XOR B = C		
A	B	C	A	B	C	A	B	C
0	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	1	1
1	0	1	1	0	0	1	0	1
1	1	1	1	1	1	1	1	0

8.4.1 Logical Sum (Logical OR)

A logical sum instruction performs the OR operation of 4-bit data according to the truth values listed in Table 8-3.

Example:

```
  MOV  R1,  #1010B  ;
  MOV  M1,  #1001B  ;
;  ①
  OR   R1,  M1      ;
;  ②
  OR   M1,  #1100B  ;
```

In this case, operation of ① is performed as follows.

```
   1010B ... Content of R1
OR 1001B ... Content of M1
   1011B ... Operation result
```

Consequently, 1011B is stored in R1.

Operation of ② is performed as follows.

```
   1001B ... Content of M1
OR 1100B ... Immediate data
   1101B ... Operation result
```

Consequently, 1101B is stored in M1.

Logical sum is useful for setting (1) the data memory contents in 1-bit, 2-bit, 3-bit, or 4-bit units.

8.4.2 Logical Product (Logical AND)

A logical product instruction performs an AND operation of 4-bit data according to the truth values listed in Table 8-3.

Example:

```
  MOV  R1,  #1010B  ;
  MOV  M1,  #1001B  ;
;  ①
  AND  R1,  M1      ;
;  ②
  AND  M1,  #1100B  ;
```

In this case, operation of ① is performed as follows.

```
   1010B ... Content of R1
AND 1001B ... Content of M1
   1000B
```

As a result, 1000B is stored in R1.
Operation of ② is performed as follows.

```
   1001B ... Content of M1
AND 1100B ... Immediate data
   1000B
```

As a result, 1000B is stored in M1.

Logical product is useful for resetting the data memory contents in 1-bit, 2-bit, 3-bit, or 4-bit units.

8.4.3 Logical Exclusive Sum (Logical Exclusive OR)

A logical exclusive sum instruction performs XOR operation of 4-bit data according to the truth values listed in Table 8-3.

Example:

```
  MOV  R1,  #1010B  ;
  MOV  M1,  #1001B  ;
;  ①
  XOR  R1,  M1      ;
;  ②
  XOR  M1,  #1100B  ;
```

In this case, operation of ① is performed as follows.

```
    1010B ... Content of R1
XOR 1001B ... Content of M1
    0011B
```

As a result, 0011B is stored in R1.

Operation of ② is performed as follows.

```
    1001B ... Content of M1
XOR 1100B ... Immediate data
    0101B
```

As a result, 0101B is stored in M1.

Logical exclusive sum is useful for inverting the data memory contents in 1-bit, 2-bit, 3-bit, or 4-bit units.

8.5 BIT CHECKING

As listed in Table 8-1, bit checking methods are classified into True bit checking (1) and False bit checking (0).

The "SKT instruction" and "SKF instruction" are used for True bit (1) checking and False bit (0) checking.

The "SKT" and "SKF" instructions can be used for data memory only.

Bit checking is not influenced by the BCD flag of a program status word. The CY flag and Z flag do not impose any influence. However, the CMP flag is reset (0) if the "SKT" or "SKF" instruction is executed.

When an Index Enable flag is set (1), address qualification is performed by the index register. See **9, "System Register (SYSREG)"** for address qualification by an index register.

Sections **8.5.1** and **8.5.2** describe True bit (1) checking and False bit (0) checking.

8.5.1 Checking a True (1) Bit

True bit (1) checking instruction "SKT m, #n" checks whether the bits specified by n among the four bits in the data memory are "True (1)". When all the bits specified by n are "True (1)", the following instruction is skipped.

Example:

```
    MOV  M1,  #1011B  ;
;   ①
    SKT  M1,  #1011B  ;
    BR   A
    BR   B
;   ②
    SKT  M1,  #1101   ;
    BR   C
    BR   D
```

In this case, bits b_3, b_1, and b_0 of M1 are checked in ① and control is branched to B because all the bits are True (1).

In 2 , bits b_3, b_2, and b_0 of M1 are checked in ② , control is branched to C because bit b_2 of M1 is False (0).

8.5.2 Checking a False (0) Bit

False bit (0) checking instruction "SKF m, #n" checks whether the bits specified by n among the four bits in the data memory are False (0). When all the bits specified by n are "False (0)", the following instruction is skipped.

Example:

```
    MOV  M1,  #1001B  ;
;   ①
    SKF  M1,  #0110B  ;
    BR   A
    BR   B
;   ②
    SKF  M1,  #1110B  ;
    BR   C
    BR   D
```

In ①, bits b_3 and b_1 of M1 are checked. Since all the bits are False (0), control is branched to B.

In ②, bits b_3, b_2, and b_1 of M1 are checked. Since bit b_3 of M1 is True (1), control is branched to C.

8.6 COMPARISON CHECKING

As shown in Table 8-1, comparison checking involves the checking of "equal to", "not equal to", "greater than", and "less than". The conditions, "equal to", "not equal to", "greater than", and "less than" are checked by the "SKE instruction", "SKNE instruction", "SKGE instruction", and "SKLT instruction", respectively.

The "SKE", "SKNE", "SKGE", and "SKLT" instructions can only perform comparison checking between data memory and immediate data.

Consequently, when comparison checking is performed between a general register and data memory, a subtraction instruction is executed using the CMP flag and Z flag of a program status word (PSWORD). See Section **8.3, "Arithmetic operation (addition and subtraction in binary mode and decimal mode)".**

Comparison checking is influenced by the BCD flag and CMP flag of the program status word (PSWORD). The CY flag and Z flag do not impose any influence.

When an Index Enable flag is set (1), address qualification is performed by the index register. See **9, "System Register (SYSREG)"** for address qualification by an index register.

Sections **8.6.1** to **8.6.4** describe instructions for checking "equal to", "not equal to", "greater than", and "less than".

8.6.1 Checking "Equal to"

"Equal to" checking instruction "SKE m, #i" checks whether the contents of data memory and immediate data are "equal".

When the contents of the data memory and immediate data are "equal", the following instruction is skipped.

Example:

```
   MOV  M1,  #1010B  ;
;  ①
   SKE  M1,  #1010B  ;
   BR   A
   BR   B
;  ②
   SKE  M1,  #1000B  ;
   BR   C
   BR   D
```

In this case, since the contents of M1 and immediate data 1010B are equal in ①, control is branched to B.
In ②, since the contents of M1 and 1000B of immediate data are not equal, control is branched to C.

8.6.2 Checking "Not Equal to"

"Not equal to" checking instruction "SKNE m, #i" checks whether the contents of data memory and immediate data are "not equal".

When contents of the data memory and immediate data are "not equal", the following instruction is skipped.

Example:

```
   MOV  M1,  #1010B  ;
;  ①
   SKNE M1,  #1000B  ;
   BR   A
   BR   B
;  ②
   SKNE M1,  #1010B  ;
   BR   C
   BR   D
```

In ①, since the contents of M1 and immediate data 1000B are not equal, control is branched to B.
In ②, since the contents of M1 and immediate data 1010B are equal, control is branched to C.

8.6.3 Checking "Greater Than"

"Greater than" checking instruction "SKGE m, #i" compares the contents of data memory and immediate data. When the contents of the data memory is "greater than" or "equal to" the immediate data, the following instruction is skipped.

Example:

```
   MOV  M1,  #1000B  ;
;  ①
   SKGE M1,  #0111B  ;
   BR   A
   BR   B
;  ②
   SKGE M1,  #1000B  ;
   BR   C
   BR   D
;  ③
   SKGE M1,  #1001B  ;
   BR   E
   BR   F
```

In this case, the content of M1 which is 1000B is "greater than" immediate data in ①, "equal to" immediate data in ②, and "less than" the immediate data in ③, so that control is branched to B, D, and E respectively.

8.6.4 Checking "Less Than"

"Less than" checking instruction "SKLT m, #i" compares the contents of data memory and immediate data. When the content of the data memory is "less than" the immediate data, the following instruction is skipped.

Example:

```
   MOV  M1,  #1000B  ;
;  ①
   SKLT M1,  #1001B  ;
   BR   A
   BR   B
;  ②
   SKLT M1,  #1000B  ;
   BR   C
   BR   D
;  ③
   SKLE M1,  #0111B  ;
   BR   E
   BR   F
```

In this case, since the content of M1, which is 1000B, is determined as "less than", "equal to", and "greater than" the immediate data in ①, ②, and ③ respectively, control is passed to B, C, and E in each case.

8.7 ROTATION PROCESSING

Rotation processing includes clockwise rotation processing and counterclockwise rotation processing.

The "RORC instruction" is used for clockwise rotation processing.

The "RORC instruction" can be used for general registers only.

Clockwise rotation processing by the "RORC instruction" is not influenced by the BCD flag and CMP flag of the program status word. The Z flag does not impose any influence.

Address qualification by the index register (IX) is not performed for the "RORC instruction" even if an Index Enable flag is set (1).

Although there is no special instruction for counterclockwise rotation processing, the processing can be performed by the "ADDC instruction", which is an addition instruction.

Sections **8.7.1** and **8.7.2** describe clockwise rotation processing and counterclockwise rotation processing.

8.7.1 Clockwise Rotation Processing

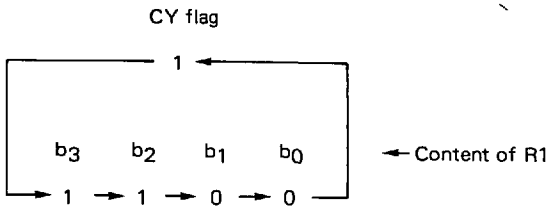
Clockwise rotation processing "RORC r" rotates the content of a general register by one bit in the direction of the low-order bits.

In this case, the content of the CY flag is written to the highest-order bit b_3 of the general register and the content of the lowest-order bit b_0 is written to the CY flag.

Example 1:

```
MOV   PSW,  #0100B ;  Sets (1) the CY flag.
MOV   R1,   #1001B
RORC  R1           ;
```

In this case, the following processing is performed.



That is, the content is rotated clockwise as follows; CY flag → b_3, b_3 → b_2, b_2 → b_1, b_1 → b_0, b_0 → CY flag.

Example 2:

```
MOV   PSW,  #0000B ;  Resets (0) the CY flag
MOV   R1,   #1000B ;
MOV   R2,   #0100B ;
MOV   R3,   #0010B ;
RORC  R1
RORC  R2
RORC  R3
```

In this case, the above program rotates the 12-bit data of R1, R2, and R3 clockwise.

8.7.2 Counterclockwise Rotation Processing

Counterclockwise rotation processing can be performed by using an addition instruction, "ADDC r, m".

Example:

```
MOV   RSW, #0000B ;  Resets (0) the CY flag.
MOV   R1,  #1000B ;
MOV   R2,  #0100B ;
MOV   R3,  #0010B ;
ADDC  R3,  R3
ADDC  R2,  R2
ADDC  R1,  R1
```

In this case, the above program rotates the 12-bit data of R1, R2, and R3 counterclockwise.

9. SYSTEM REGISTER (SYSREG)

A system register is used for directly controlling the CPU and is stored in data memory.

9.1 STRUCTURE OF A SYSTEM REGISTER

Fig. 9-1 shows the location of a system register in data memory.

As shown in Fig. 9-1, a system register is stored in data memory addresses 74H to 7FH. The same system register is stored in addresses 74H to 7FH in any bank.

Since a system register is stored in data memory, it can be manipulated by a data memory manipulation instruction.

A system register can be specified as a general register.

Fig. 9-2 shows the structure of a system register.

As shown in Fig. 9-2, a system register consists of seven types of registers.

Address register (AR)
Window register (WR)
Bank register (BANK)
Index register (IX)
Data memory row address pointer (MP)
General register pointer (RP)
Program status word (PSWORD)

Fig. 9-1 Location of a system register in data memory

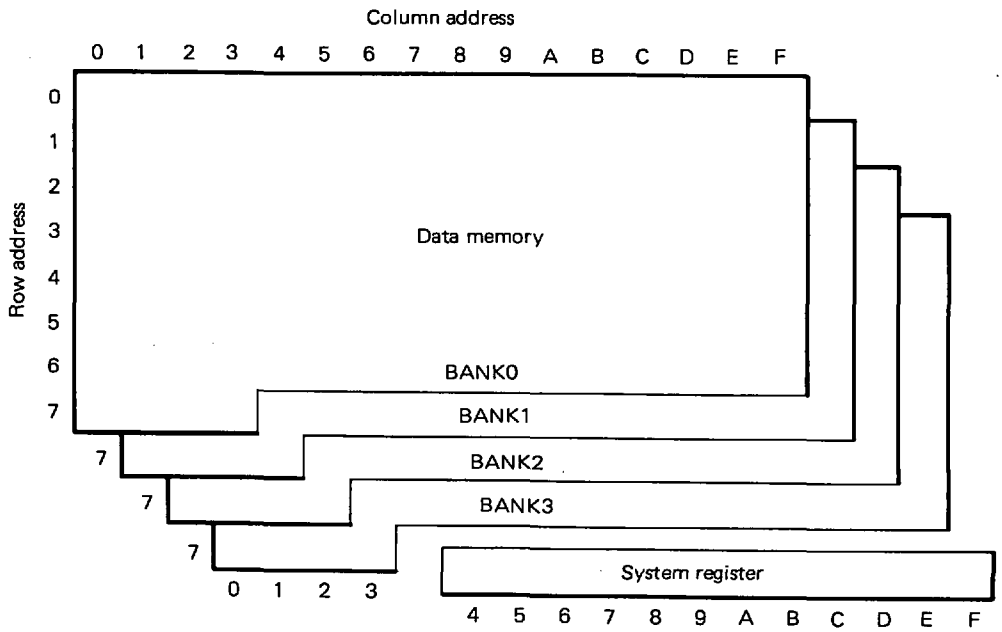
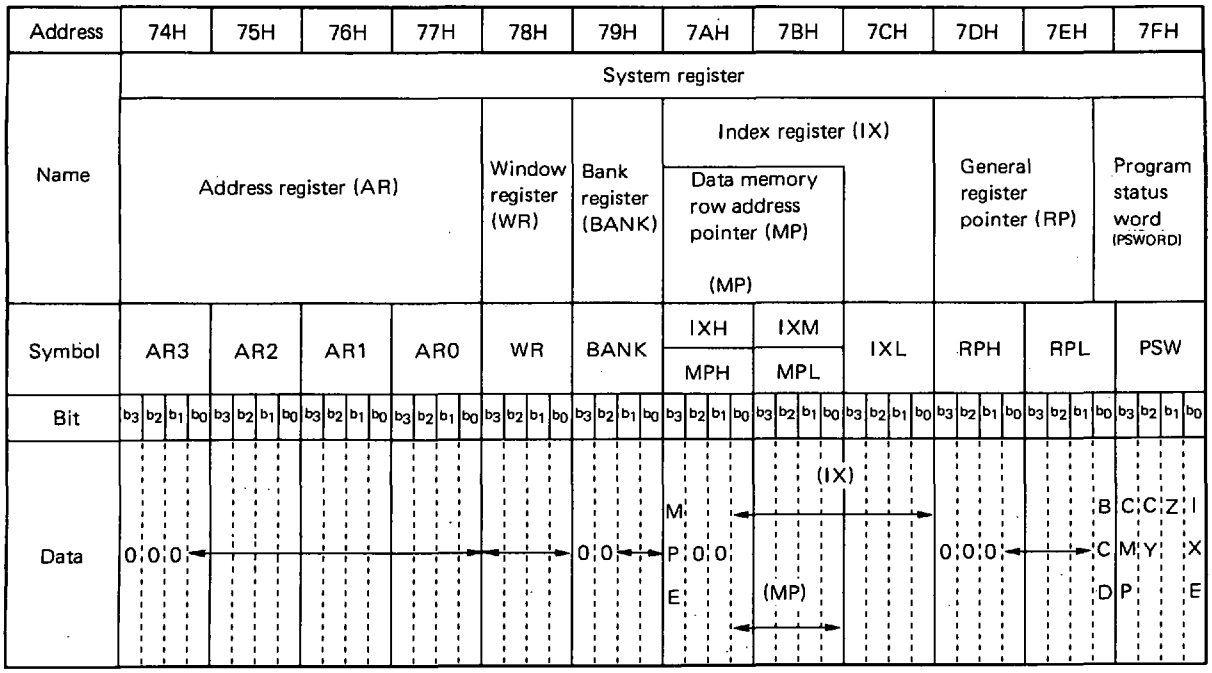


Fig. 9-2 Structure of a system register



9.2 FUNCTION OF A SYSTEM REGISTER

9.2.1 Function of Each Register

The function of each register of a system register is described below.

See Sections **9.3** to **9.8** for details of the function of each register.

(1) Address register (AR)

Specifies indirectly a program memory address.

(2) Window register (WR)

Used for data transfer with a register file.

(3) Bank register (BANK)

Specifies a data memory bank.

(4) Index register (IX)

Used for address qualification of data memory.

(5) Data memory row address pointer (MP)

Specifies a row address at general register indirect transfer.

(6) General register pointer (RP)

Specifies addresses of a bank and row of a general register.

(7) Program status word (PSWORD)

Sets operation and transfer instruction conditions.

9.2.2 System Register Manipulation Instructions

Since a system register is stored in data memory, it can be controlled by any data memory manipulation instruction. The following special instructions are made available for an address register and index register of the system register.

INC AR: Increments the content of the address register by "1".
Since the number of valid bits of an address register is 13, the value is incremented by "1" whenever an instruction is executed and when the value reaches 1FFFH, 0000H is set as the next value.

INC IX: Increments the content of an index register by "1".
Since the number of valid bits of an index register is 9, the value is incremented by "1" whenever an instruction is executed and when the value reaches 1FFH, 000H is set as the next value.

9.3 ADDRESS REGISTER (AR)

9.3.1 Structure of an Address Register

Fig. 9-3 shows the structure of an address register.

Fig. 9-3 Structure of an address register

Address		74H				75H				76H				77H			
Name		Address register (AR)															
Symbol		AR3				AR2				AR1				AR0			
Bit		b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
Data		0	0	0	MSB												LSB
At resetting	Power On	0				0				0				0			
	CE	0				0				0				0			
	STOP	0				0				0				0			

Power ON : At Power On Reset
CE : At CE Reset
STOP : At execution of a Clock Stop instruction

As shown in Fig. 9-3, an address register consists of 16 bits of the system register address 74H to 77H (AR3 to AR0). However, since the high-order 13 bits are always set to 0, the register actually functions as a 13-bit register.

At Power On Reset, CE Reset, or execution of a Clock Stop instruction, all the 16 bits are reset to 0.

9.3.2 Function of an Address Register

An address register specifies the program memory address at execution of an indirect branch instruction (BR @AR), indirect subroutine call instruction (CALL @AR), table reference instruction (MOVT DBF, @AR), or stack manipulation instruction (PUSH AR, POP AR).

Sections **9.3.3** to **9.3.6** describe the operation at execution of each instruction.

A special instruction (INC AR) is provided to the address register for incrementing the value by 1 at a time. By using this instruction, the data in the address register can be incremented in 13-bit units. When the content of the address register is 1EFBH, the content of the register can be changed to 0000H by executing the "INC AR" instruction.

However, data stored in addresses from 0000H to 1EFBH become valid.

9.3.3 Table Reference Instruction (MOVT DBF, @AR)

By executing the "MOVT DBF, @AR" instruction, constant data (16 bits) of the program memory address specified by the content of the address register can be read to the data buffer (DBF: addresses from 0CH to 0FH of BANK0) of the data memory.

Constant data of the program memory stored in addresses from 0000H to 1EFBH can be read to a data buffer.

See also Section **11.2.4, "Data Buffer and Table Reference"**.

Example:

```
Address     Label
000FH       DATA1:   16-bit constant data
                     MOV   AR0,  #0FH  ;  Writes 0FH in AR0.
                     MOV   AR1,  #0H   ;  Writes 0H in AR1.
                     MOV   AR2,  #0H   ;  Writes 0H in AR2.
                     MOV   AR3,  #0H   ;  Writes 0H in AR3.
                     MOVT  DBF,  @AR   ;  Reads constant data in address 000FH of program
                                       ;  memory to a data buffer.
```

9.3.4 Stack Manipulation Instruction (PUSH AR, POP AR)

By executing the "PUSH AR" instruction, the stack pointer can be decremented by 1 and the content of the address register (AR) can be stored in the address stack register indicated by the stack pointer.

By executing the "POP AR" instruction, the content of the address stack register specified by the stack pointer can be transferred to the address register and the stack pointer can be incremented by 1.

See **5, "Stacking".**

9.3.5 Indirect Branch Instruction (BR @AR)

By executing the "BR @AR" instruction, control can be branched to the program memory address specified by the content of the address register.

Example

```
MOV   AR0, #0FH  ;  Writes 0FH in AR0.
MOV   AR1, #0H   ;  Writes 0H in AR1.
MOV   AR2, #0H   ;  Writes 0H in AR2.
MOV   AR3, #0H   ;  Writes 0H in AR3.
BR    @AR        ;  The program passes control to 000FH.
```

Since high-order 3 bits of the address register are always set to zeros, the values are always zeros even if a Write statement is executed.

9.3.6 Indirect Subroutine Call Instruction (CALL @AR)

By executing the "CALL @AR" instruction, the subroutine of the program memory address specified by the content of the address register can be called.

Example 1:

```
Address     Label
000FH       SUB:    Subroutine processing
                    RET

                MOV   AR0,  #0FH  ;  Writes 0FH in AR0.
                MOV   AR1,  #0H   ;  Writes 0H in AR1.
                MOV   AR2,  #0H   ;  Writes 0H in AR2.
                MOV   AR3,  #0H   ;  Writes 0H in AR3.
                CALL  @AR         ;  Calls the subroutine in address 000FH.
```

In Example 1, the address used for indirect subroutine call is specified by the "MOV" statement.
This method deteriorates the program memory usage efficiency when subroutines are frequently called indirectly.
The use of the "POP" and "PUSH" instructions and table reference is recommended as shown in Example 2.

Example 2:

```
SUBENTRY:
        DI
        POP     AR
        MOVT    DBF, @AR
        INC     AR
        PUSH    AR
        EI
        PUT     AR, DBF
        BR      @AR
SUB1    :
SUB2    :
MAIN    :
        CALL    SUBENTRY
        DW      .DL. SUB1
        CALL    SUBENTRY
        DW      .DL. SUB2
```

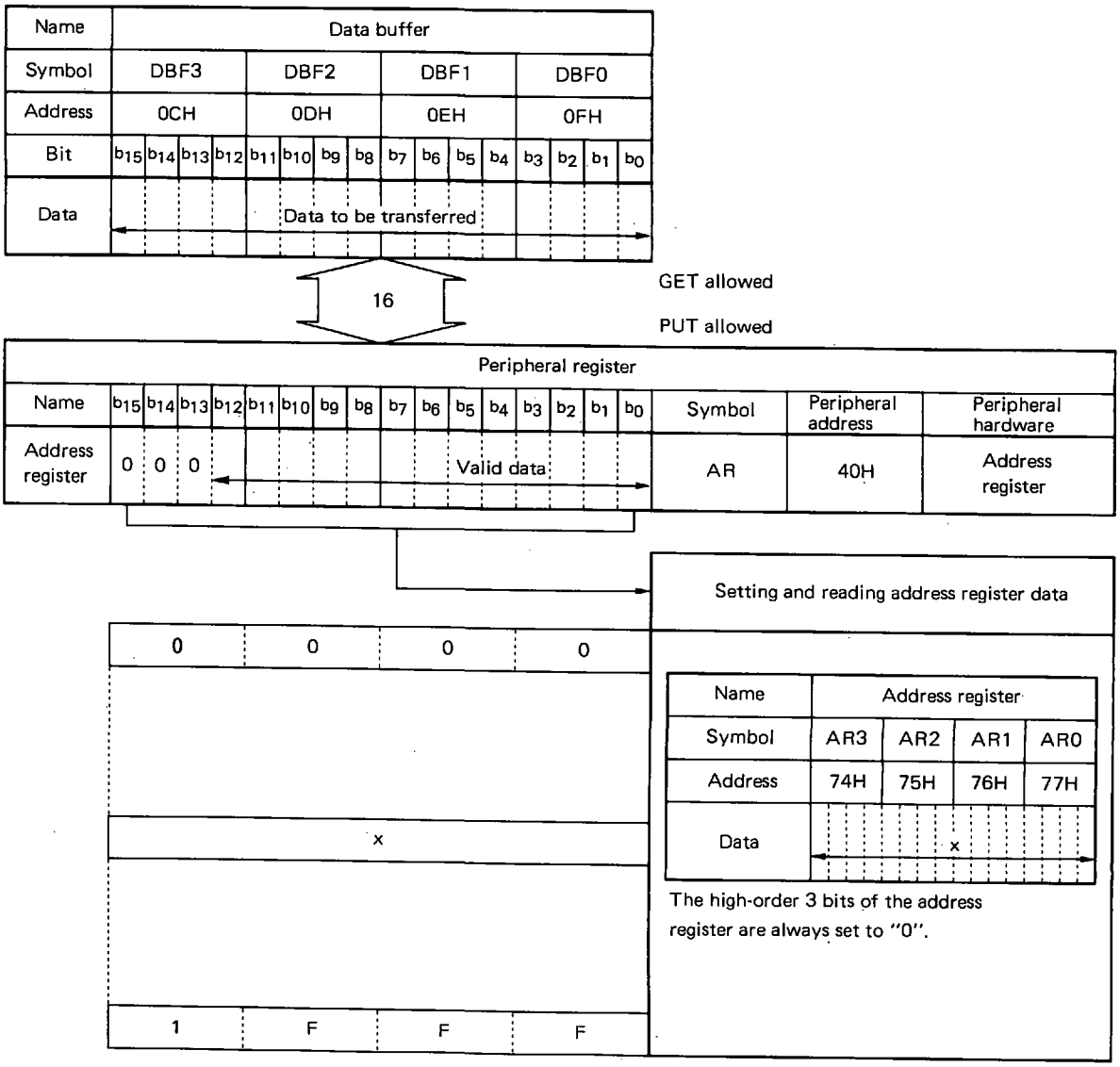
9.3.7 Address Register and Data Buffer

Data in an address register can be manipulated directly using a data memory manipulation instruction, and the data can also be transferred via a data buffer as a part of peripheral hardware.

That is, data in an address register can be read and written via a data buffer using the "PUT" and "GET" instructions in addition to data memory manipulation instructions.

The relationship between an address register and a data buffer is shown below.

See **11, "Data Buffer"** for details of a data buffer.



9.4 WINDOW REGISTER (WR)

9.4.1 Structure of a Window Register

Fig. 9-4 shows the structure of a window register.

Fig. 9-4 Structure of a window register

Address		78H			
Name		Window register (WR)			
Symbol		WR			
Bit		b_3	b_2	b_1	b_0
Data		MSB			LSB
At resetting	Power On	Undefined			
	CE	Retaining the previous status			
	STOP				

As shown in Fig. 9-4, a window register consists of 4 bits of system register 78H.

At Power On Reset, the content is undefined and at CE Reset or execution of a Clock Stop instruction, the previous status is retained.

9.4.2 Function of a Window Register

A window register is used for transfer of data to a register file (RF) which is described later.

Special instructions, "PEEK WR, rf" and "POKE rf, WR" are used for transfer of data to a register file.

Sections **9.4.3** and **9.4.4** describe operation performed when each instruction is executed.

See **10, "Register File"** also.

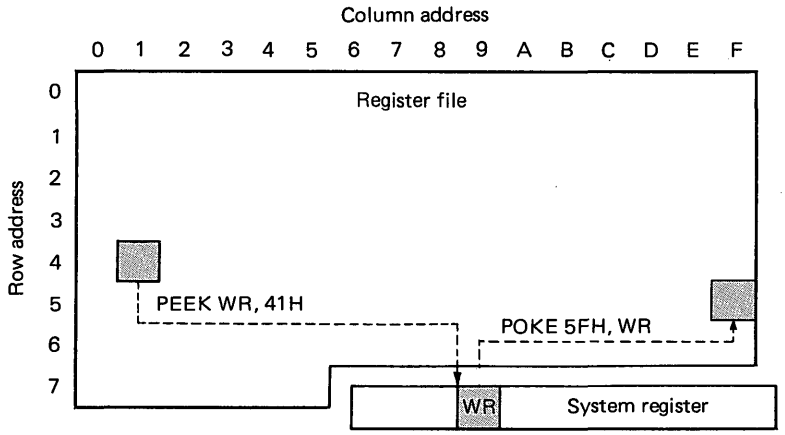
9.4.3 PEEK WR, rf Instruction

As shown in Fig. 9-5, by executing the PEEK WR, rf instruction (rf: register file address), the contents of the register file whose address is specified by rf can be transferred to a window register.

9.4.4 POKE rf, WR Instruction

As shown in Fig. 9-5, by executing the POKE rf, WR instruction, the contents of the window register can be transferred to the register file whose address is specified by rf.

Fig. 9-5 Operation of PEEK and POKE instructions



9.5 BANK REGISTER (BANK)

9.5.1 Structure of a Bank Register

Fig. 9-6 shows the structure of a bank register.

Fig. 9-6 Structure of a bank register

Address		79H			
Name		Bank register (BANK)			
Symbol		BANK			
Bit		b_3	b_2	b_1	b_0
Data		0	0	^ MSB v	^ LSB v
At resetting	Power On	0			
	CE	0			
	STOP	0			

As shown in Fig. 9-6, a bank register consists of 4 bits of system register 79H (BANK). However, the high-order 2 bits are always set to 0.

At Power On Reset, CE Reset, or execution of a Clock Stop instruction, the register is reset to 0.

9.5.2 Function of a Bank Register

A bank register switches a bank of data memory.

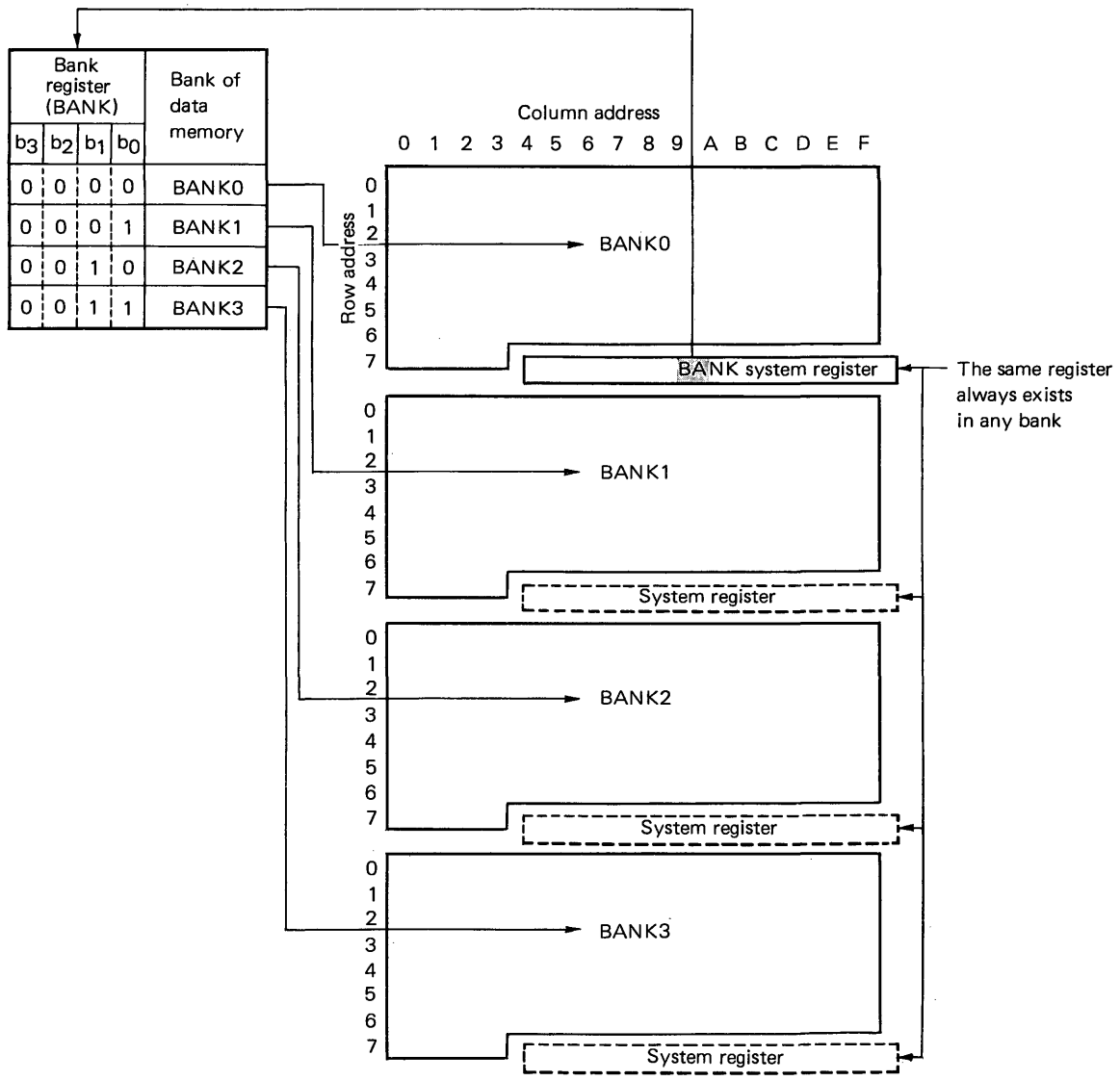
Fig. 9-7 shows the bank register values and specification of a bank of data memory.

As shown in Fig. 9-7, a data memory is divided into four banks according to the bank register, and when a data memory manipulation instruction is executed, the data memory specified by the bank register is manipulated.

Consequently, to manipulate data memory of BANK1 when BANK0 is currently specified, the bank must be changed to BANK1 by writing 0001H to the bank register.

In this case, there is no bank concept for the system registers located in addresses from 74H to 7FH of the data memory and the same system register exist in addresses 74H to 7FH in all the banks. In this case, even if "MOV BANK, #0" instruction is executed in BANK1 or "MOV BANK, #0" instruction is executed in BANK2, as a result, 0 is written in the bank register (BANK: address 78H). That is, when a bank register is manipulated, the status of the bank is not related.

Fig. 9-7 Data memory bank specification



The "BANKn" instruction ($0 \leq n \leq 3$) is provided in the 17K series Assembler as an intrinsic macro instruction for manipulating a bank.

An example of a bank and data memory operation is shown below.

Example:

```
M000    MEM     0.00H   ; Symbol definition
M100    MEM     1.00H   ;
BANK0                   ; Equivalent to MOV BANK, #0000B
MOV     M000,   #0101B  ; Writes 0101B in address 00H of BANK0
BANK1                   ; Equivalent to MOV BANK, #0001B
MOV     M100,   #0101B  ; Writes 0101B in address 00H of BANK1
MOV     M000,   #0101B  ; Writes 0101B in address 00H of BANK1
                        ; In this case, although data memory M000 is defined as BANK0 at symbol
                        ; definition, it is used as the bank which is specified at that time at program
                        ; execution.
```

9.6 INDEX REGISTER (IX) AND DATA MEMORY ROW ADDRESS POINTER (MP: MEMORY POINTER)

9.6.1 Structure of an Index Register and a Data Memory Row Address Pointer

Fig. 9-8 shows the structure of an index register and data memory row address pointer.

Fig. 9-8 Structure of an index register and a data memory row address pointer

Address		7AH				7BH				7CH				7EH				7FH			
Name		Index register (IX) / Memory pointer (MP)																Program status word (PSWORD)			
Symbol		IXH / MPH				IXM / MPL				IXL								PSW			
Bit		b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
Data		MPE	0	0	^MSB<				IX				^LSB<								IXE
									^LSB<												
							MP														
At resetting	Power On	0				0				0											0
	CE	0				0				0											0
	STOP	0				0				0											0

As shown in Fig. 9-8, an index register consists of 11 bits comprising the low-order 3 bits (IXH) of system register 7AH, 7BH, and 7CH (IXM and IXL) and an Index Enable flag which is the lowest bit of 7FH (PSW).

A data memory row address pointer (memory pointer) consists of an 8-bit data memory row address pointer comprising the low-order 3 bits of 7AH (MPH), and 7BH (MPL), and a data memory row address pointer enable flag (memory pointer enable flag: MPE) which is the highest bit of 7AH (MPH).

That is, the high-order 7 bits of the index register and the data memory row address pointer are shared.

However, the high-order 2 bits of the index register and data memory row address pointer (bits b_2 and b_1 of 7AH) are always set to "0".

At Power On Reset, CE Reset, or execution of a Clock Stop instruction, all the registers are reset to "0".

9.6.2 Functions of an Index Register and Data Memory Row Address Pointer

The function of an index register and data memory row address pointer are described in sections (1) and (2).

(1) Index register

An index register qualifies the bank and address of the data memory specified by the instruction according to the contents of the index register when a data memory manipulation instruction is executed.

However, address qualification becomes valid only when an Index Enable flag is set by the index register.

For address qualification, the bank and address of the data memory are manipulated by the contents of the index register and OR operation, and the execution is performed for the data memory of the address (called an actual address) specified by the operation result.

Address qualification by an index register is effective for all the data memory manipulation instructions.

An index register is not effective for the following instructions.

MOVT	DBF, @AR	BR	addr	INC	AR	EI
PEEK	WR, rf	BR	@AR	INC	IX	DI
POKE	rf, WR	CALL	addr	RORC	r	
GET	DBF, p	CALL	@AR	STOP	s	
PUT	p, DBF	RET		HALT	h	
PUSH	AR	RETSK		NOP		
POP	AR	RETI				

(2) Data memory row address pointer

A data memory row address pointer qualifies the address of the indirect transfer destination with the content of the data memory row address pointer when a general register indirect transfer instruction (MOV @r, m or MOV m, @r) is executed.

However, address qualification by a data memory row address pointer is effective only when a data memory row address pointer enable flag (memory pointer enable flag) is set (1).

Address qualification is effective only for general register indirect transfer instructions.

Address qualification is performed by replacing the addresses of the bank and row of the indirect transfer destination with the content of data memory row address pointer.

Table 9-2 shows data memory address qualification and indirect transfer address qualification by an index register and data memory row address pointer.

Sections **9.6.3** to **9.6.6** describe each operation of address qualification of data memory by an index register data and data memory row address pointer.

Table 9-2 Data memory address qualification by an index register and data memory row address pointer

IXE	MPE	General register address specified by r R			Data memory address specified m M			Indirect transfer address specified by @r @R		
		Bank	Row address	Column address	Bank	Row address	Column address	Bank	Row address	Column address
		b_3 b_2 b_1 b_0	b_2 b_1 b_0	b_3 b_2 b_1 b_0	b_3 b_2 b_1 b_0	b_2 b_1 b_0	b_3 b_2 b_1 b_0	b_3 b_2 b_1 b_0	b_2 b_1 b_0	b_3 b_2 b_1 b_0
0	0	(RP)		r	(BANK)		m	(BANK)	mH	(R)
0	1		As above			As above		(MP)		(R)
1	0		As above		(BANK)	m		(BANK)	mH	
					Logical OR			Logical OR		(R)
					(IX)			(IXH)	(IXM)	
1	1		As above			As above		(MP)		(R)

	Group of instructions whose addresses are qualified		
Addition and subtraction	ADD, ADDC, SUB, SUBC	r	m
			m. #i
Comparison	AND, OR, XOR	r	m
			m. #i
Logic	SKE, SKGE, SKLT, SKNE		m. #i
Checking	SKT, SKF		m. #n
Transfer	LD, ST	r	m
	MOV		m. #i
		@r	m
			Indirect transfer address

M	:	Data memory address
(M)	:	Data memory address contents
m	:	Data memory address excluding bank
mH	:	Data memory row address
mL	:	Data memory column address
BANK	:	Bank register
(BANK)	:	Bank register contents
IX	:	Index register
(IX)	:	Index register contents
IXH	:	Index register bits b_{10}-b_8
IXM	:	Index register bits b_7-b_4
IXL	:	Index register bits b_3-b_0
MP	:	Data memory row address pointer
(MP)	:	Data memory row address pointer contents
R	:	General register address
(R)	:	General register address contents
r	:	General register column address
RP	:	General register pointer
(RP)	:	General register pointer contents

9.6.3 When MPE=0 and IXE=0 (Without Data Memory Qualification)

As shown in Table 9-2, data memory addresses are not influenced by either an index register nor by a data memory row address pointer.

(1) Data memory manipulation instruction

Example:

BANK0 is specified and the register is in row address 0.

```
R003   MEM   0.03H
M061   MEM   0.61H
ADD    R003, M061
```

When the above instruction is executed, the contents of general register R003 and data memory M061 are added, and the result is stored in general register R003 as shown in Fig. 9-9.

(2) General register indirect transfer

Example 1:

BANK0 is specified and the general register is set in row address 0.

```
R005   MEM   0.05H
M034   MEM   0.34H
MOV    R005, #8     ; R005 ← 8
MOV    @R005, M034  ; Register indirect transfer
```

When the above instruction is executed, the contents of data memory are transferred to address 38H of the data memory as shown in Fig. 9-9.

That is, the "MOV @r, m" instruction transfers the contents of the data memory specified as m to the data memory of the indirect address specified by @r.

The contents of the general register whose bank address and row address are the same as m (row address 3 of BANK0 in the above example) and whose column address is specified by r (8 in the above example) are used for the indirect transfer address. That is, 38H is used in the above example.

Example 2:

BANK0 is specified and the general register is set in row address 0.

```
R00B   MEM   0.0BH
M034   MEM   0.34H
MOV    R00B, #0EH   ; R00B ← 0EH
MOV    M034, @R00B  ; Register indirect transfer
```

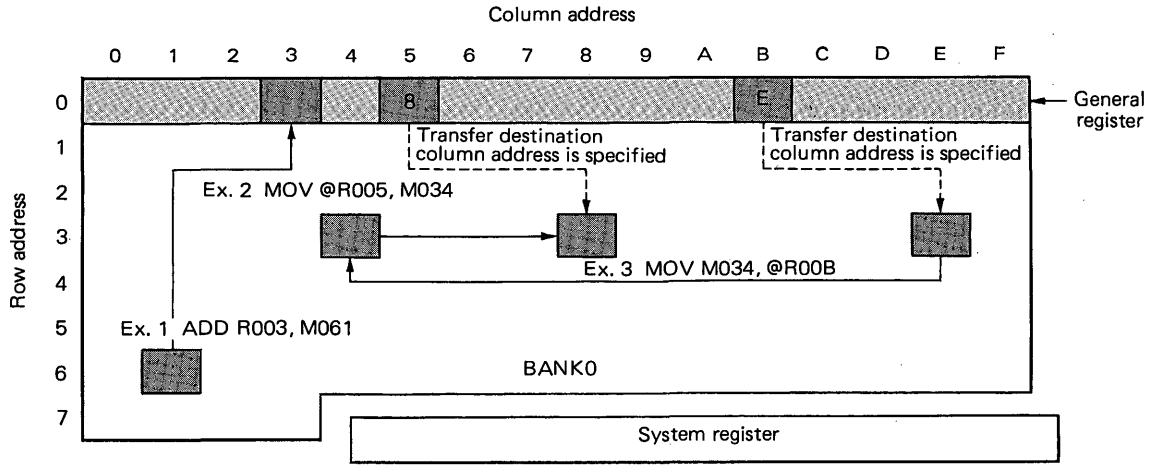
When the above instruction is executed, the contents of the data memory of address 3EH are transferred to data memory M034.

That is, the "MOV m, @r" instruction transfers the contents of the data memory of the indirect address specified by @r to the data memory specified by m.

The contents of the general register whose bank and row addresses are the same as m (row address 3 of BANK0) and whose column address is specified by r (0EH in the above example) are used as the indirect transfer address 3EH.

This example shows the case where the data memory address source (transfer source) and destination (transfer destination) are switched from those shown in Example 1.

Fig. 9-9 Operation example when IXE=0 and MPE=0



Address generation of the example shown above

ADD R003, M061

		Bank	Row address	Column address
Data memory address	M	0000	110	0001
General register address	R	0000	000	0011

Address generation of Example 1

MOV @R005, M034

		Bank	Row address	Column address
Data memory address	M	0000	011	0100
General register address	R	0000	000	0101
Indirect transfer address	@R	0000 (Same as M)	011 (Same as M)	1000 (Contents of R)

9.6.4 When MPE=1 and IXE=0 (Diagonal Indirect Transfer)

As shown in Table 9-2, the bank and row address of the indirect transfer address specified by @r are used as the data memory row address pointer value only when a general register indirect transfer instruction (MOV @r, m or MOV m, @r) is executed.

Example 1:

BANK0 is specified and the general register is set in row address 0.

```
R005    MEM     0.05H
M034    MEM     0.34H
MOV     MPL,    #0110B ;  MP ← 00110B
MOV     MPH,    #1000B ;  MPE ← 1
MOV     R005,   #8     ;  R005 ← 8
MOV     @R005,  M034   ;  Register indirect transfer
```

When the above instruction is executed, the contents of data memory M034 are transferred to address 68H of the data memory as shown in Fig. 9-10.

That is, if the "MOV @r, m" instruction is executed when MPE=1, the contents of the data memory specified by m are transferred to the data memory of the indirect address specified by @r.

In this case, the content of the general register whose bank and row address are the data memory row address pointer value (BANK0 and row address 6 in the above example), and whose column address specified by r (8 in the above example), used as the indirect address specified by @r.

That is, in the above example, the address is 68H of BANK0.

In comparison with the Example 1, MPE=0 in Section 9.6.3, the difference is that the bank and row address of the indirect address specified by @r are specified by the data memory row address pointer (in Example 1 in Section 9.6.3, the bank and row address of the indirect address are the same, which is m).

Consequently, by specifying MPE=1, general register indirect transfer can be performed in the diagonal direction.

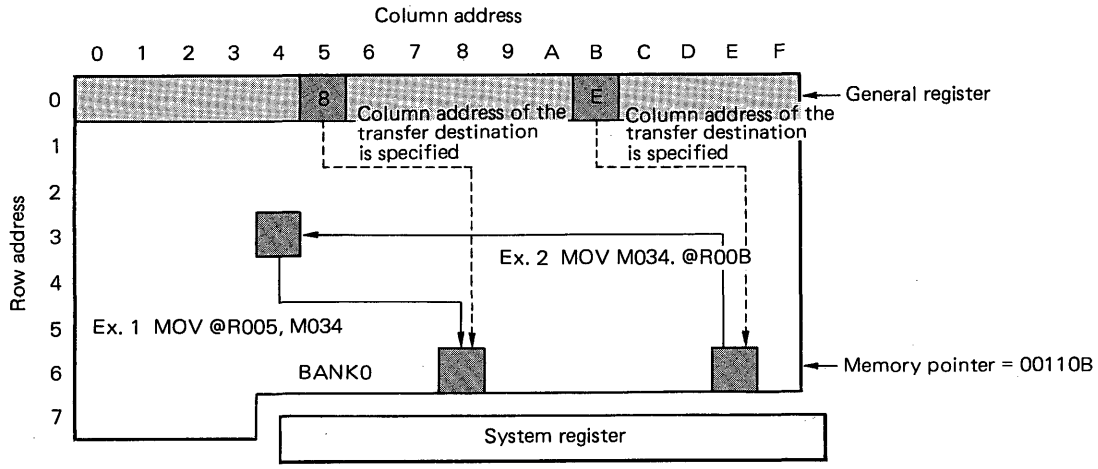
Example 2:

BANK0 is specified and the general register is set in row address 0.

```
R00B    MEM     0.0BH
M034    MEM     0.34H
MOV     MPL,    #0110B ;  MP ← 00110B
MOV     MPH,    #1000B ;  MPE ← 1
MOV     R00B,   #0EH   ;  R00B ← 0EH
MOV     M034,   @R00B
```

When the above statement is executed, the contents of the data memory of address 6EH are transferred to the data memory M034 as shown in Fig. 9-10.

Fig. 9-10 Operation example when MPE=1 and IXE=1



Address generation of Example 1

MOV R005, M034

		Bank	Row address	Column address
Data memory address	M	0000	011	0100
General register address	R	0000	000	0101
Indirect transfer address	@R	0000	110	1000
		Same as MP		Contents of R

Address generation of Example 2

MOV M034, @R00B

		Bank	Row address	Column address
Data memory address	M	0000	011	0100
General register address	R	0000	000	1011
Indirect transfer address	@R	0000	110	1110
		Same as MP		Contents of R

9.6.5 When MPE=0 and IXE=1 (Data Memory Address Index Qualification)

As shown in Table 9-2, addresses of the entire data memory and banks specified directly by operation "m" of the instruction are qualified by the index register when a data memory manipulation instruction is executed.

When a general register indirect transfer instruction (MOV @r, m or MOV m, @r) is executed, the bank and row address of the indirect transfer address specified by @r are also qualified by the index register.

For address qualification, the contents of the data memory address and index register are manipulated by OR operation, and an instruction for the data memory address (called real address) specified by the operation result is executed.

The example is shown below.

Example 1:

BANK0 is specified and the general register is set in row address 0.

```
R003    MEM     0.03H
M061    MEM     0.61H
MOV     IXL,    #0010B ;  IX ← 00010010010B
MOV     IXM,    #1001B ;
MOV     IXH,    #0000B ;
OR      PSW,    #0001B ;  IXE ← 1
ADD     R003,   M061
```

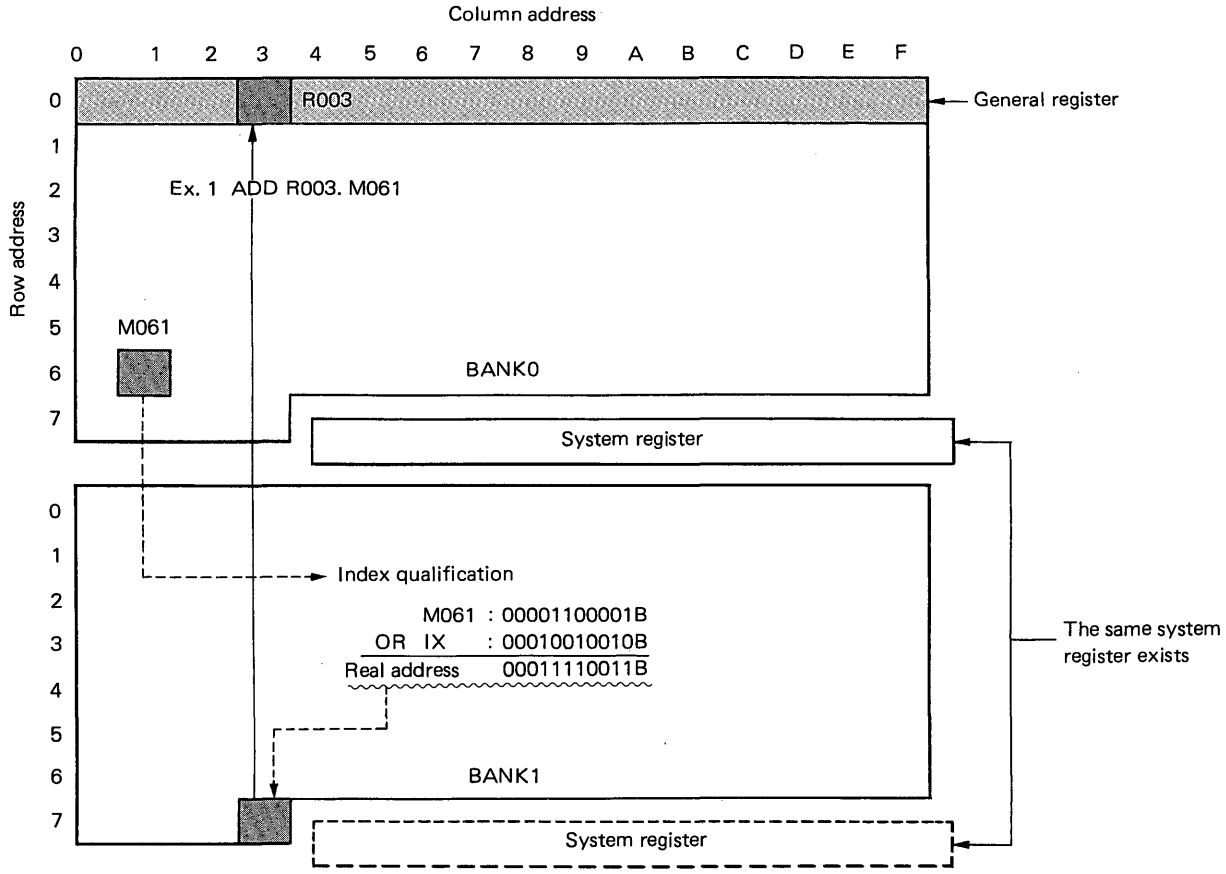
When the above instruction is executed, the contents of the data memory (real address) of address 73H of BANK1 and the contents of general register R003 (address 03H of BANK0) are added and the result is stored in general register R003.

That is, when the "ADD r, m" instruction is executed, the data memory address (address 61H of BANK0 in the above example) specified by "m" is qualified by the index register.

For qualification, the content of address 61H of BANK0 which is the address of data memory M061 (binary 00001100001B) and the index register value (00010010010B) are manipulated by OR operation, and the instruction is executed for the real address (address 73H of BANK1) using the result 00011110011B as the real address.

The address of the data memory directly specified by operand "m" of the instruction is qualified (OR operation) by the index register. Compare this case with that when IXE=0 (Example shown in Section **9.6.3**).

Fig. 9-11 Operation example when IXE=1 and MPE=0



Address generation of Example 1

ADD R003, M061

		Bank	Row address	Column address
Data memory address	M	0000	110	0001
General register address	R	0000	000	0011
Index qualification — M061		0000	110	0001
		BANK	m	
Index qualification — IX		0001	001	0010
		IHX	IXM	IXL
Index qualification — Real address (OR operation)		0001	111	0011

An instruction is executed for this address

Example 2:

General register indirect transfer

BANK0 is specified and the general register is set in row address 0.

```
R005   MEM    0.05H
M034   MEM    0.34H
MOV    IXL,   #0001B ;  IX ← 000100000001B
MOV    IXM,   #1000B ;
MOV    IXH,   #0000B ;
OR     PSW,   #0001B ;  IXE ← 1
MOV    R005,  #8     ;  R005 ← 8
MOV    @R005, M034   ;  Register indirect transfer
```

When the above instructions are executed, the contents of address 35H of data memory BANK1 are transferred to address 38H of data memory BANK1 as shown in Fig. 9-12.

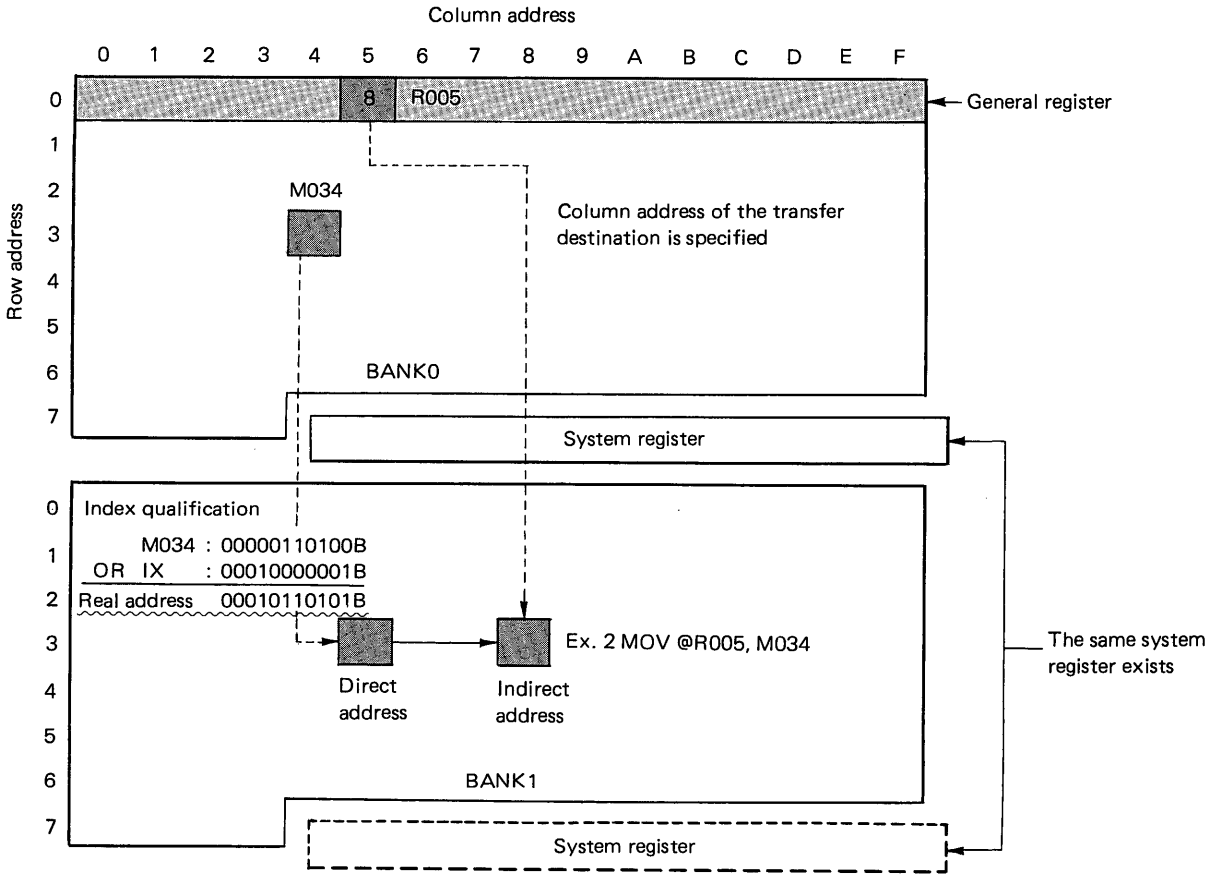
That is, if the "MOV @r, m" instruction is executed when IXE=1, the data memory address (direct address) specified by "m" is qualified by the contents of the index register, and the bank and row address of the indirect address specified by "@r" are also qualified by the index register.

For address qualification, all the direct bank and row address and column addresses specified by "m" are qualified, and the indirect bank and row addresses which are specified by @r are also qualified.

That is, in the above example, the direct address is 35H of BANK1 and indirect address is 38H of BANK1.

Consequently, the bank and row addresses of the direct address specified by "m" and column address are qualified by the index register, and general register indirect transfer is performed to the same row address as the data memory address which was qualified (in Example 2 in Section **9.6.3,** direct addresses are not qualified). Compare this with Example 2 IXE=0 in Section **9.6.3.**

Fig. 9-12 General register indirect transfer operation example when MPE=0 and IXE=1



Example 3:

Contents of the entire bank memory are cleared to 0.

```
M000    MEM     0.00H
MOV     IXL,    #0      ;  (IX) ← 0
MOV     IXM,    #0      ;
MOV     IXH,    #0      ;
LOOP:
OR      PSW,    #0001B  ;  IXE ← 1
MOV     M000,   #0      ;  Sets the data memory specified (IX) to 0.
INC     IX              ;  (IX) ← (IX) + 1
AND     PSW,    #1110B  ;  IXE ← 0: Since the address of IXE is 7FH, the address is not qualified by (IX).
SKT     IXM,    #0111B  ;  If row address 7 is detected
BR      LOOP            ;  If the address is not 7, LOOP
ADD     IXM,    #1      ;  Specify row address 0 of the next bank (row address 7 is not cleared)
ADDC    IXH,    #0      ;
SKT     IXM,    #1000B  ;  Was data memory up to BANK3 cleared?
SKF     IXH,    #0001B  ;
BR      LOOP            ;  If not cleared, LOOP
```

Example 4:

Array processing

As shown in Fig. 9-13, 8-bit data A is defined in one dimension as shown below.

DIM A(N) $(0 \leq N \leq 15)$

In this case, the following instructions must be executed to perform the operation,

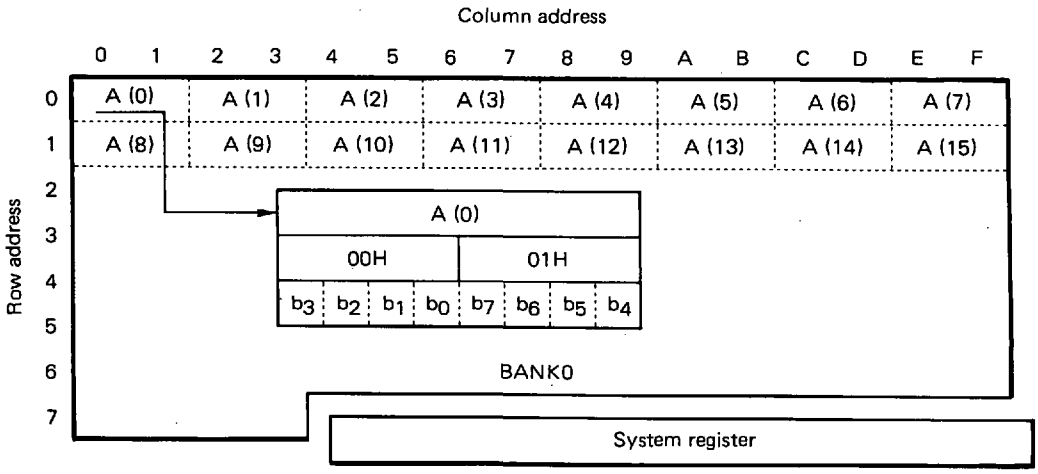
$A(N) = A(N) + 4 \quad (0 \leq N \leq 15)$

BANK0 is specified and the general register is set in row address 7.

```
M000    MEM     0.00H
M001    MEM     0.01H
MOV     IXH,    #0      ;  (IX) ← N
MOV     IXM,    #0      ;
MOV     IXL,    #N      ;
ADD     IXL,    IXL     ;  (IX) ← (IX) x 2
ADDC    IXM,    IXM     ;  Since the array element is 8 bits, the data memory address qualified by the
ADDC    IXH,    IXH     ;  index register is shifted.
OR      PSW,    #0001B  ;  IXE ← 1
ADD     M000,   #4      ;  Value 4 is added to data memory M000 and data memory M001 whose addresses
ADDC    M001,   #0      ;  are qualified by (IX). That is, value 4 is added to the 8-bit array specified by
                        ;  A(N).
```

To specify N of array A(N), N must be specified for the index register as shown in the above example.

Fig. 9-13 Operation example when IXE=1 and MPE=0 (array processing)



9.6.6 When MPE=1 and IXE=1

As shown in Table 9-2, all the data memory addresses directly specified by operand "m" of a data memory manipulation instruction are qualified by an index register.

By using a general register indirect transfer instruction, the direct address specified by "m" is qualified by an index register and the indirect address specified by "@r" is specified by the data memory row address pointer.

Example 1:

BANK0 is specified and the general register is set in row address 0.

```
R005    MEM     0.05H
R034    MEM     0.34H
MOV     IXL,    #0001B ;  (IX) ← 00010000001B
MOV     IXM,    #1000B ;  (MP) ← 0001000B
MOV     IXH,    #0000B ;
MOV     R005,   #8     ;  R005 ← 8
OR      IXH,    #1000B ;  MPE ← 1
OR      PSW,    #0001B ;  IXE ← 1
MOV     @R005,  M034   ;  Register indirect transfer
```

When the above instruction is executed, the contents of address 35H of BANK1 of the data memory are transferred to address 08H of BANK1 of the data memory.

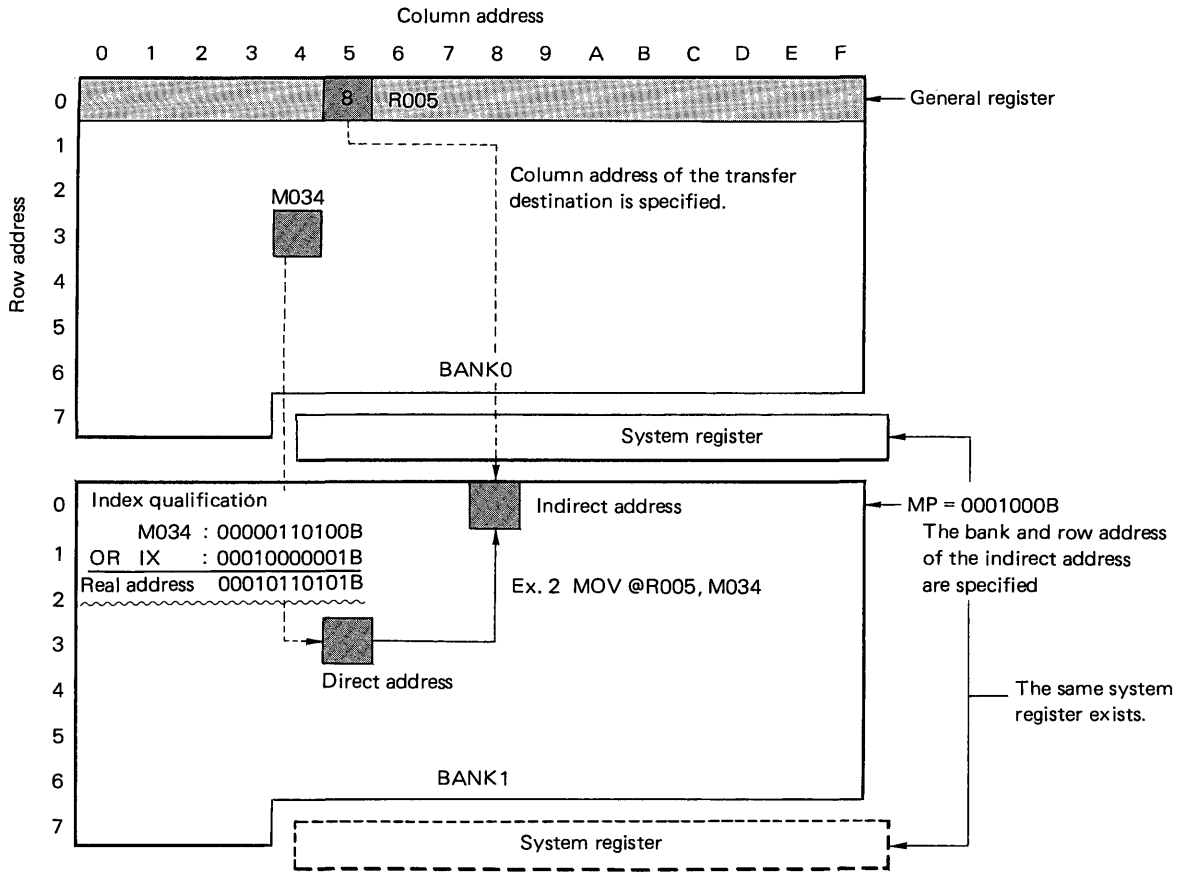
That is, if the "MOV @r, m" instruction is executed when MPE=1 and IXE=1, the data memory address (direct address) specified by "m" is qualified by the contents of the index register, and the indirect address specified by "@r" is qualified by the contents of the data memory row address pointer.

For direct address qualification, all of the bank, row address, and column address of the data memory address specified by "m" are manipulated by OR operation with the contents of the index register, and for indirect address specification, the bank and row address become the contents of the data memory row address pointer.

That is, in the above example, the direct address is 35H of BANK1 and the indirect address is address 08H of BANK1.

In this case, the bank and row address of the indirect address specified by "@r" are specified by the contents of the data memory row address pointer (in Example 2 in Section **9.6.5**, an indirect address qualified by the index region; compare this example with Example 2 when MPE=0 and IXE=1 in Section **9.6.5.**)

Fig. 9-14 General register indirect transfer operation example when MPE=1 and IXE=1



9.7 GENERAL REGISTER POINTER (RP)

9.7.1 Structure of a General Register Pointer

Fig. 9-15 shows the structure of a general register pointer.

Fig. 9-15 Structure of a general register pointer

Address		7DH				7EH			
Name		General register pointer (RP)							
Symbol		RPH				RPL			
Bit		b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
Data		0	0	0	MSB			LSB	BCD
At Resetting	Power On	0				0			
	CE	0				0			
	STOP	0				0			

As shown in Fig. 9-15, a general register pointer consists of 7 bits; 4 bits of address 7DH (RPH) of the system register and the high-order 3 bits of address 7EH (RPL).

However, since the high-order 3 bits of address 7DH are always set to 0, the low order 4 bits (the lowest bit of address 7DH and the high-order 3 bits of address 7EH) can be used.

At Power On Reset, CE Reset, or execution of a Clock Stop instruction, all the addresses are cleared to 0.

9.7.2 Function of a General Register Pointer

A general register pointer is used for specifying a general register in data memory.

Sixteen nibbles, which have the same row address as the data memory can be specified for a general register.

Consequently, the row address to be used is specified by the general register pointer as shown in Fig. 9-16.

Since the valid number of bits of a general register pointer are 4 bits, data memory row addresses 0H to 7H of HANK0 and 0H to 7H of BANK1 can be used for a general register. That is, the entire data memory except for BANK2 can be specified as general registers.

When data memory is specified as a general register, the operation and transfer between the general register and data memory can be performed.

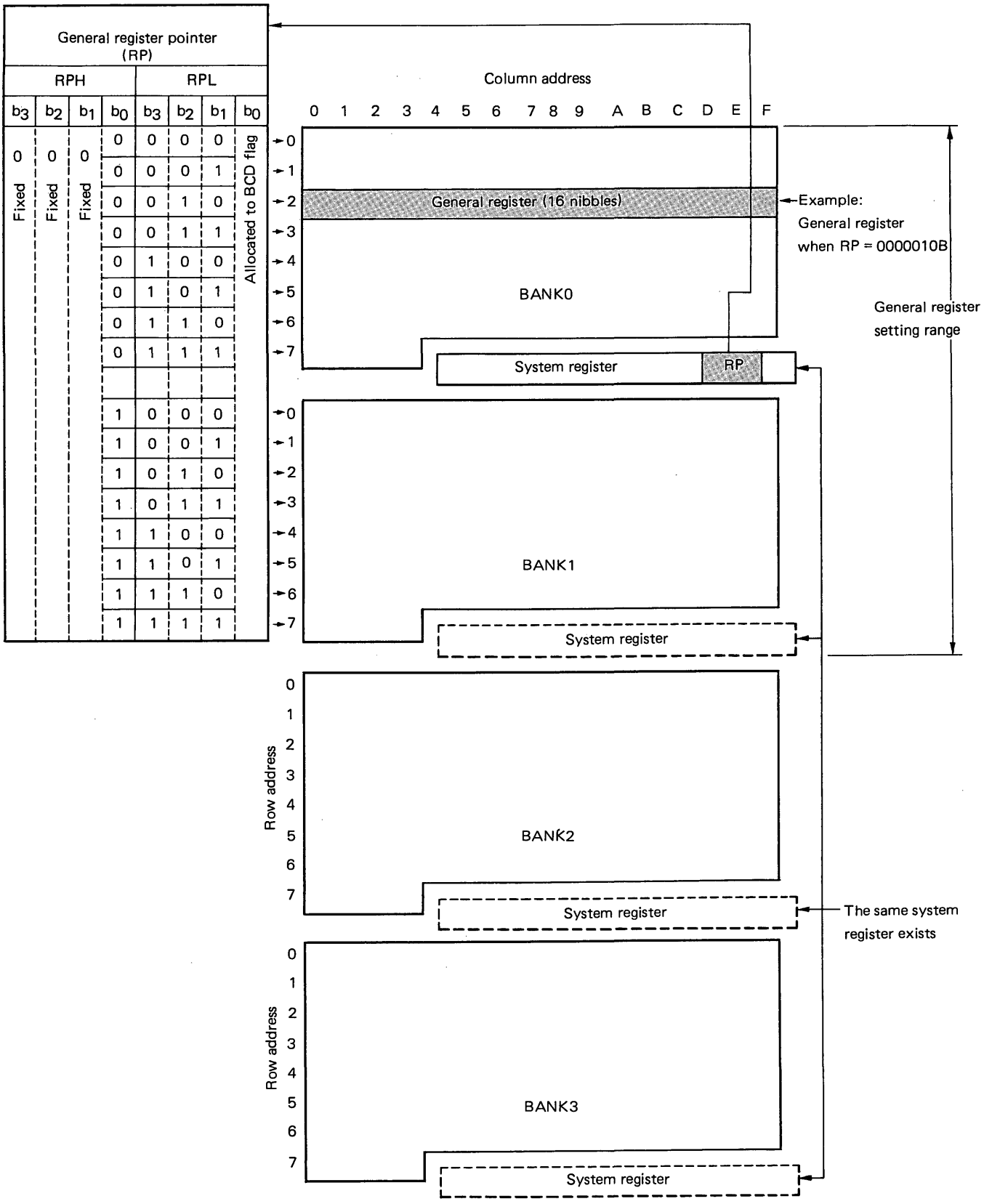
For instance, when the following instruction is executed, addition or transfer can be performed between the general register whose address is specified by operand "r" of the instruction, and data memory whose address is specified by "m".

ADD r, m or

LD r, m

See **7, "General Register (GR)"** for details.

Fig. 9-16 Structure of a general register



9.8 PROGRAM STATUS WORD (PSWORD)

9.8.1 Structure of a Program Status Word

Fig. 9-17 shows the structure of a program status word

Fig. 9-17 Sturcture of a program status word

Address		7EH				7FH			
Name		(RP)			Program status word (PSWORD)				
Symbol		RPL				RSW			
Bit		b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
Data					BCD	CMP	CY	Z	IXE
At Resetting	Power On	0				0			
	CE	0				0			
	STOP	0				0			

As shown in Fig. 9-17, a program status word consists of 5 bits; the lowest bit of address 7EH (RPL) of the system register and 4 bits of address 7FH (PSW).

Each bit of the program status word functions differently; they are a BCD flag (BCD), Compare flag (CMP), Carry flag (CY), Zero flag (Z), and Index Enable flag (IXE).

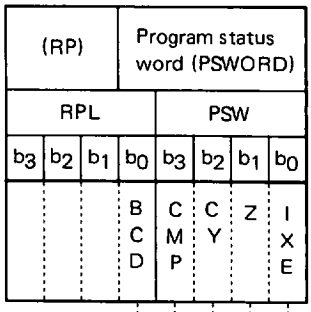
At Power On Reset, CE Reset, or execution of a Clock Stop instruction, all the above bits are cleared to 0.

9.8.2 Function of a Program Status Word

A program status word is a register for setting the conditions of an operation and transfer instruction in ALU, and for indicating the operation result.

Table 9-3 outlines the function of each flag of a program status word.

Table 9-3 Function outline of each flag of a program status word



Flag name	Function
Index Enable flag (IXE)	Flag for qualifying the data memory address at execution of a data memory manipulation instruction 0 : Not qualified 1 : Qualified
Zero flag (Z)	Flag for indicating that the arithmetic operation result is 0. Since the statuses of 0 and 1 are different depending on the contents of the Compare flag, caution is necessary (see Table 9-4).
Carry flag (CY)	Flag which indicates Carry or Borrow after execution of an addition instruction or subtraction instruction. Reset (0) when there is no Carry or Borrow. Set (1) when there is Carry or Borrow. This bit is also used as a shift bit of the "RORC r" instruction.
Compare flag (CMP)	Flag for not storing the arithmetic operation result to data memory or a general register. 0 : The result is stored 1 : The result is not stored
BCD flag (BCD)	Flag for performing arithmetic operation in decimal mode 0 : Binary operation 1 : Decimal operation

9.8.3 Index Enable Flag (IXE)

An IXE flag is used for qualifying the address of data memory at execution of a data memory manipulation instruction.

When an IXE flag is set (1), the contents of the data memory address and index register (IX) specified by the instruction are manipulated by OR operation, and the instruction is executed for the data memory which uses the OR operation result as the real address.

See Section **9.6, "Index Register (IX) and Data Memory Row Address Pointer (MP: Memory Pointer)".**

9.8.4 Zero Flag (Z) and Compare Flag (CMP)

A Z flag indicates that the result of the arithmetic operation is 0, and a CMP flag is used for not storing the arithmetic operation result to data memory or to a general register.

Table 9-4 shows the setting and resetting conditions of Z flag and CMP flag statuses.

Table 9-4 Statuses of a Compare flag (CMP) and conditions of setting and resetting of a Zero flag (Z)

Condition	Status of Z flag	
	When the CMP flag is 0.	Whan the CMP flag is 1.
At Power On Reset, CE Reset, or execution of a Clock Stop instruction.	Reset	Set together with the CMP flag.
When "0" is directly written to the Z flag by a data memory manipulation instruction.	Reset	Reset
When the result of the arithmetic operation is not "0".	Reset	Reset
When "1" is directly written to the Z flag by a data memory manipulation instruction.	Set	Set
When the result of the arithmetic operation is "0" and neither Carry nor Borrow has occurred.	Set	The previous status of the Z flag is retained.
When the result of the arithmetic operation is "0" and Carry or Borrow has occurred.	Set	The previous status of the Z flag is retained.

The Z flag and CMP flag are used for comparing the contents of a general register with the contents of data memory. See **8, "ALU"** for details.

9.8.5 Carry Flag (CY)

The CY flag is set (1) when Carry or Borrow occurs and reset (0) when neither Carry nor Borrow occurs.

When the "RORC r" instruction (the contents of the general register specified by r are shifted to the right by 1 bit) is executed, the value of the CY flag immediately before execution of the instruction is shifted to the highest bit of the general register and the lowest bit is shifted to the CY flag.

The CY flag is useful for skipping the next instruction when Carry or Borrow occurs.

See **8, "ALU"** for details.

9.8.6 BCD Flag (BCD)

The BCD flag is used for performing arithmetic operation in decimal mode.

By setting (1) the BCD flag, all the arithmetic operations are performed in decimal mode.

See **8, "ALU"** for details.

9.8.7 Notes on Arithmetic Operation

When performing arithmetic operations (addition and subtraction) for a program status word (PSW), the following points must be noted.

When arithmetic operation is performed for a program status word, the result of the operation is stored.

The example is shown below.

Example:

```
MOV  PSW,  #0001B
ADD  PSW,  #1111B
```

When the above instruction is executed, it is assumed that the CY flag which is bit b_2 of PSW is set (1) since Carry occurs. However, 0000B is stored in PSW because the result of the arithmetic operation is 0000B.

9.9 NOTES ON USING SYSTEM REGISTERS

9.9.1 Reserved Words of System Registers

Since a system register is stored in data memory, all the data memory manipulation instructions can be used. In this case, since a direct data memory address can be specified in an operation of an instruction, symbol definition must be set for the data memory address in advance when Assembler (AS17K) of 17K series is used as shown in Example 1.

Although the system register is also used as data memory, symbol definition is set for the register as a "reserved word" in Assembler (AS17K) because it contains a specific function, unlike ordinary general purpose data memory.

A reserved word of a system register is assigned in addresses 74H to 7FH and is defined by the one of the symbols (AR3, AR2, ... PSW) shown in Fig. 9-2, "Structure of a system register".

If a reserved word is used, symbol definition is not required as shown in Example 2.

See **26, "μPD17005 Reserved Words"** also.

Example 1:

```
MOV   34H,  #0101B ;  When data memory address 34H or 76H is specified in the operand, an error
MOV   76H,  #1010B ;  occurs in the Assembler.
M037  MEM   0.37H  ;  For general purpose data memory, symbol definition is required for the data
MOV   M037, #0101B ;  memory address using the MEM pseudo instruction.
```

Example 2:

```
MOV   AR1,  #1010B ;  If AR1 (address 6H), which is a reserved word is used, symbol definition is not
                   ;  required.
                   ;  Reserved word AR1 is defined as "AR1 MEM 0.76H".
```

When Assembler (AS17K) is used, the following macro instructions are incorporated in the Assembler internal section as flag type symbol manipulation instructions.

SETn : Sets "1" in the flag.
CLRn : Resets "0" in the flag.
SKTn : Skips if all the flags are set to "1".
SKFn : Skips if all the flags are set to "0".
NOTn : Inverts the flag.
INITFLG: Initializes the flag.

Consequently, by using these built-in macro instructions, data memory can be used as a flag as shown in Example 3.

Since the functions of a program status word are classified in bit units (flag units), a reserved word is defined for each bit. The reserved words include MPE, BCD, CMP, CY, Z, and IXE.

By using these flag type reserved words, built-in macro instructions can be used directly as shown in Example 4.

Example 3:

```
F0003   FLG 0.00H.3 ;  Flag type symbol definition
SET1    F0003       ;  Built-in macro
        Macro expansion
        OR    .MF.F0003 SHR 4, #.DF.F0003 AND 000FH
                    ;  Sets bit b3 of address 00H of BANK0.
```

Example 4:

```
SET1    BCD         ;  Built-in macro
        Macro expansion
        OR    .MF.BCD SHR 4, #.DF.BCD AND 000FH
                    ;  Sets the BCD flag
                    ;  BCD is defined by "BCD FLG 0.7EH.0".
CLR2    Z, CY       ;  Flag of the same address
        Macro expansion
        AND   .MF.Z SHR 4, #.DF.(NOT(Z OR CY)AND 000FH)
CLR2    Z, BCD      ;  Flag of different address
        Macro expansion
        AND   .MF.Z SHR 4, #.DF. (NOT Z AND 000FH)
        AND   .MF.BCD SHR 4, #.DF. (NOT BCD AND 000FH)
```

9.9.2 Handling of System Registers which are Always Set to "0"

For data (see Fig. 9-2) which is always set to "0" among the system registers, cautions are necessary for Emulator operation and Assembler operation.

The cautions are described in Sections (1), (2), and (3).

(1) Device operation

When the register is set to "0", no influence is imposed even if a Write instruction is executed for the data. When a Read instruction is executed, "0" is set.

(2) When using a 17K series Emulator (IE-17K)

When an instruction for writing "1" to the data which is always set to "0" is executed, an error occurs. That is, when the following instructions are executed, the Emulator (IE-17K) generates an error.

Example 1:

```
MOV   BANK,  #0100B ;  Writes 1 in bit b3 which is always set to "0"
```

Example 2:

```
MOV   IXL,   #1111B ;
MOV   IXM,   #1111B ;
MOV   IXH,   #0001B ;
ADD   IXL,   #1     ;
ADDC  IXM,   #0     ;
ADDC  IXH,   #0     ;
```

However, the Emulator (IE-17K) does not generate an error even if the "INC AR" or "INC IX" instruction is executed when all the valid bits are set to "1" as shown in Example 2. This is because if the "INC" instruction is executed when all the valid bits of the address register and index register are set to "1", all these bits are set to "0" by the instruction.

The Emulator (IE-17K) does not generate an error either even if "1" is written to the data which is always set to "0", as long as the register is an address register as shown in Examples 1 and 2.

(3) When using 17K series Assembler (AS17K)

An error is not output even if an instruction for writing "1" in the data which is always set to "0" is executed.

That is, if the following instruction which is shown in Example 1 is executed, an Assembler error does not occur and an Emulator error occurs when the instruction is executed under the Emulator (IE-17K).

```
MOV  BANK,  #0100B
```

The error does not occur in Assembler (AS17K) because the data memory address used for the data memory manipulation operation cannot be checked when register indirect transfer is performed.

Consequently, an Assembler error occurs only when the following built-in macro instruction is used under Assembler (AS17K).

Value 4 or a higher value is used for "n" in the "BANKn" built-in macro instruction.

This is because the "BANKn" instruction is to be used for manipulating a bank register of a system register.

10. REGISTER FILE (RF)

A register file is used for setting conditions of peripheral hardware.

10.1 STRUCTURE OF A REGISTER FILE

10.1.1 Structure of a Register File

Fig. 10-1 shows the structure of a register file.

As shown in Fig. 10-1, the register file consists of 128 nibbles (128 words x 4 bits).

The addresses are assigned in 4-bit units for a register file in the same way as for data memory and 128 nibbles in total are used; row addresses 0H to 7H and column addresses 00H to 0FH.

Addresses from 00H to 3FH are called a control register.

10.1.2 Register File and Data Memory

Fig. 10-2 shows the relationship between a register file and data memory.

As shown in Fig. 10-2, addresses from 40H to 7FH of a register file overlap those of data memory.

That is, the same memory as addresses from 40H to 7FH of the data memory bank which is selected at that time exists in addresses from 40H to 7FH of the register file.

For instance, when the bank which is currently selected is BANK0 addresses 40H to 7FH of the register file are equal to addresses from 40H to 7FH of BANK0 of data memory. When BANK1 is selected, addresses 40H to 7FH of the register file are equal to addresses from 40H to 7FH of BANK1 of the data memory.

Fig. 10-1 Structure of a register file

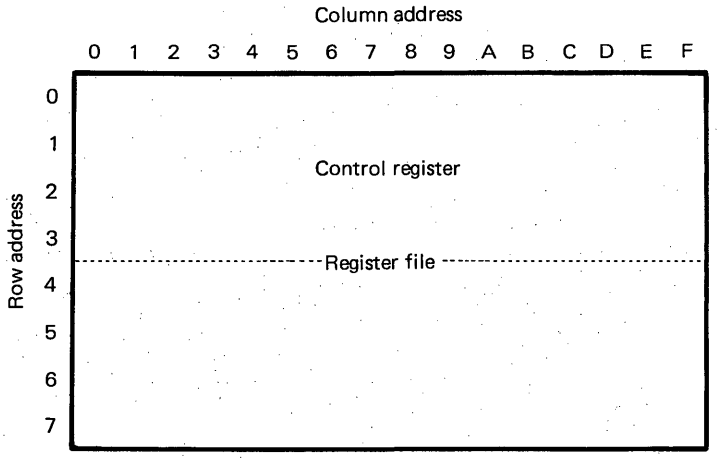
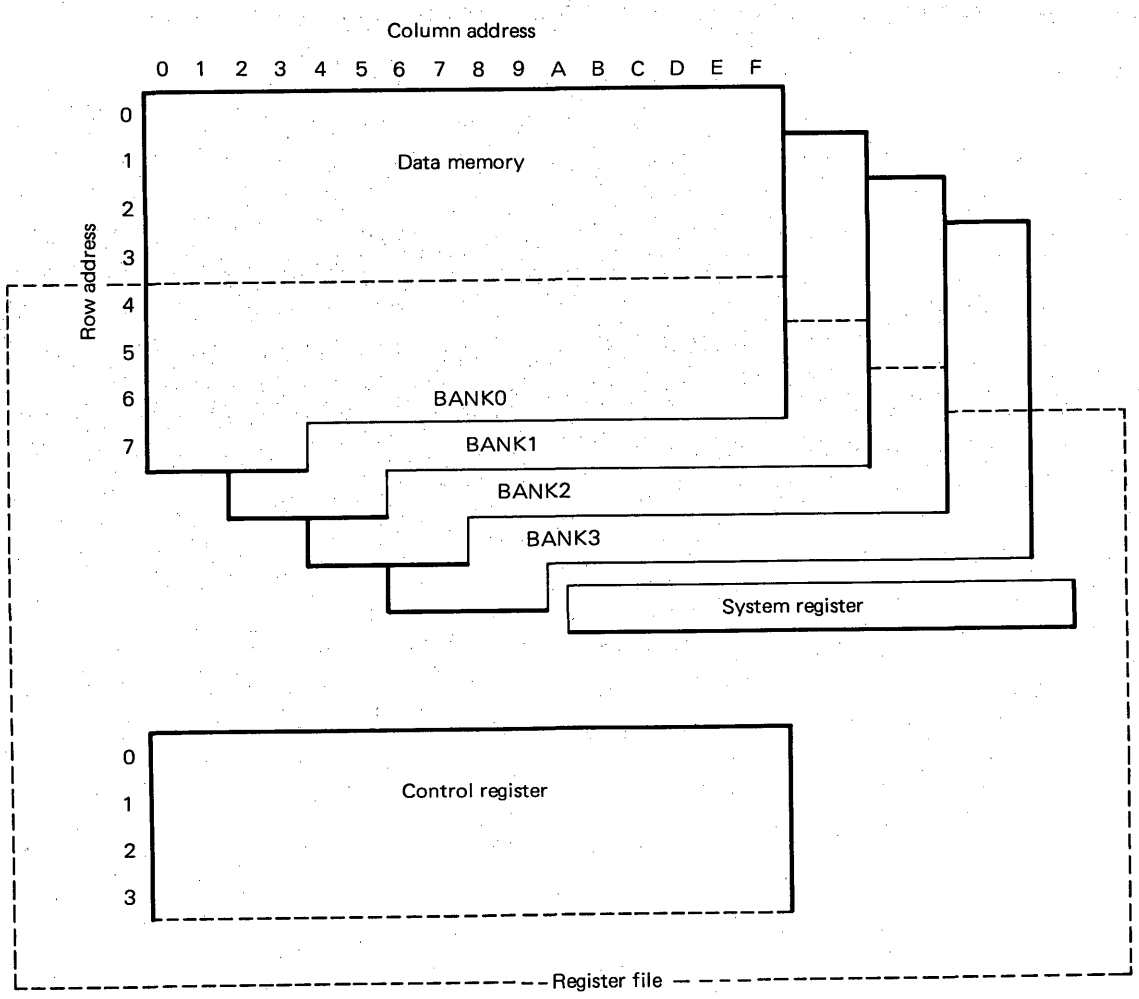


Fig. 10-2 Relationship between a register file and data memory



10.2 FUNCTION OF A REGISTER FILE

10.2.1 Function of a Register File

A register file is a control register mainly used for setting the condition of peripheral hardware.

This control register is located in a control register (addresses from 00H to 3FH) in a register file.

Since the remaining register file addresses (addresses from 40H to 7FH) overlap with data memory, the function is the same as for the ordinary data memory except that it can be manipulated by register file manipulation instructions, "PEEK" and "POKE" as described in Section **10.2.3.**

10.2.2 Function of a Control Register

Conditions of the following peripheral hardware are set by a control register.

See Section **10.3** for details of peripheral hardware and control register.

- Stack
- Timer
- Interrupt
- CE pin
- PLL frequency synthesizer
- General purpose port
- A/D converter
- D/A converter
- Clock generator port
- Serial interface
- Frequency counter
- LCD controller/driver

10.2.3 Register File Manipulation Instructions

A window register (WR: address 78H) in a system register is used for writing data to and reading data from a register file.

The following specific instructions are used for writing and reading data.

PEEK WR, rf: Reads into WR the data of the register file specified by rf.
POKE rf, WR: Writes data of WR to the register file whose address is specified by rf.

The utilization example is shown below.

Example:

```
RF00   MEM   0.80H ; Symbol definition
RF1F   MEM   0.9FH ; For symbol definition, addresses of 00H-3FH of the register file must be defined
RF53   MEM   0.53H ; as addresses from 80H to BFH of BANK0.
RF6D   MEM   0.6DH ; See Section 10.4, "Notes On Using a Register File" for details.
BANK0
① PEEK  WR,   RF00 ;
② POKE  RF1F, WR   ;
③ PEEK  WR,   RF53 ;
④ POKE  RF6D, WR   ;
BANK1
⑤ PEEK  WR,   RF00 ;
⑥ POKE  RF1F, WR   ;
⑦ PEEK  WR,   RF53 ;
⑧ POKE  RF6D, WR   ;
```

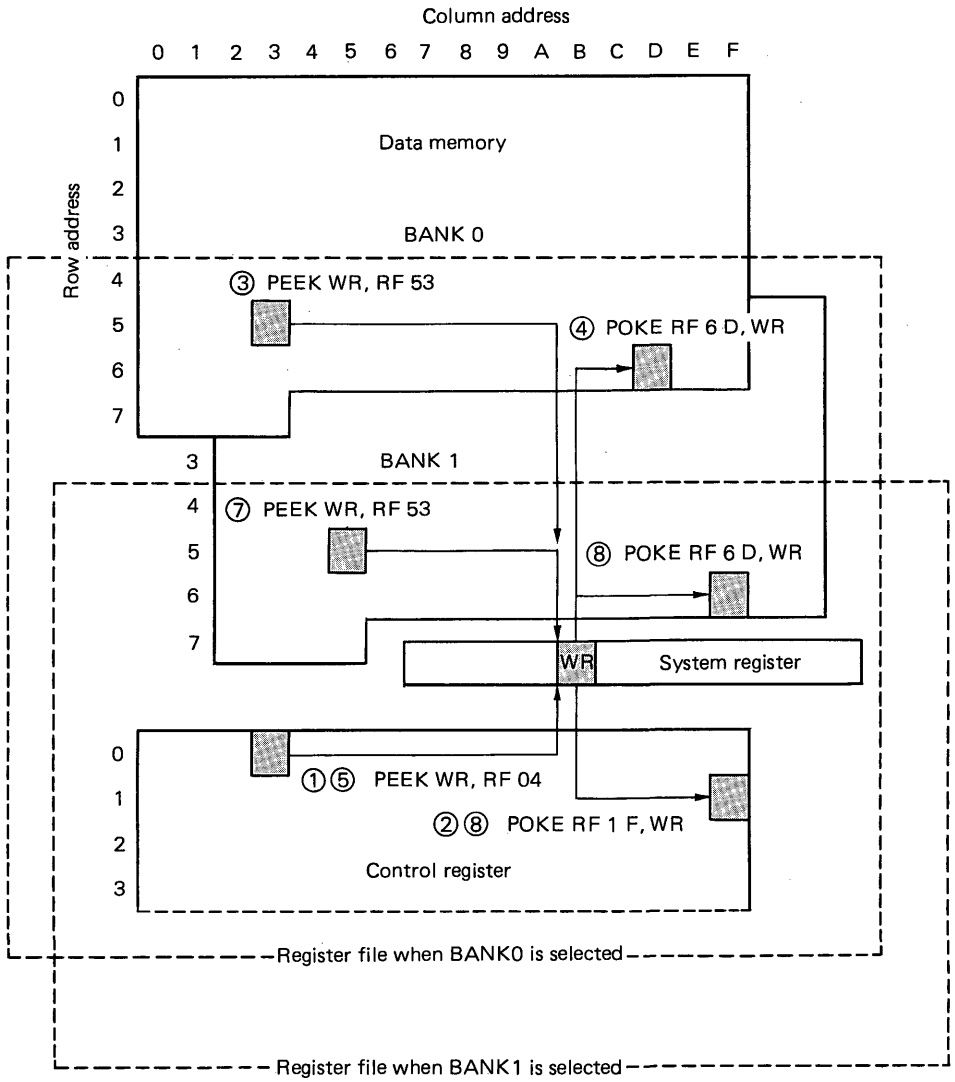
Fig. 10-3 shows the operation example.

As shown in Fig. 10-3, when the "PEEK WR, rf" or "POKE rf, WR" instruction is executed for a control register (addresses from 00H to 3FH), the contents of the register file whose address is specified by "rf" are written to or read for the window register only.

Since addresses from 40H to 7FH of the register file overlap those of the data memory, the instruction is executed for data memory address "rf" of the bank which is currently set when the "PEEK WR, rf" or "POKE rf, WR" instruction is executed.

Addresses from 40H to 7FH of a register file can also be manipulated by an ordinary memory manipulation instruction.

Fig. 10-3 Access to a register file by the PEEK or POKE instruction



10.3 CONTROL REGISTER

10.3.1 Structure of a Control Register

Fig. 10-4 shows the structure of a control register.

As shown in Fig. 10-4, a control register consists of 64 nibbles (64 words x 4 bits), which are addresses 00H to 3FH of the register file.

However, only 33 nibbles are actually used. The remaining 31 nibbles are unused registers and Read and Write operations are inhibited.

Each control register has an attribute of one nibble. Four types of attributes are available, Read/Write Enable (R/W), Read Only (R), Write Only (W), and Reset at Reading (R & Reset).

No change is made when Write operation is performed for a Read Only (R or R & Reset) register.

When a Write Only (W) register is read, an "undefined" value is read.

Among the four-bit data in one nibble, the contents of the bits which are always set to "0" are maintained at "0" when Read operation is performed and "0" is also maintained when Write operation is performed.

When the contents of the 31 nibbles, which are unused registers, are read, undefined values are read and when Write operation is performed, no change is made.

Cautions on using Assembler (AS17K) are required when unused registers, Write Only registers (W), and Read Only registers (R) are manipulated. See Section **10.4, "Notes on Using a Register File"** for details.

10.3.2 Peripheral Hardware Control Function of a Control Register

See Table 10-1 for the outline of each peripheral hardware control function of a control register.

See **11** to **24** for the functions of control registers.

Fig. 10-4 Structure of a control register (1/2)

Column address / Row address	Item	0	1	2	3	4	5	6	7
0 (8)	Name		Stack pointer SP	Serial I/02 mode selection (SI02MODE)		IF counter gate judgment (IFCGJDG)	PLL unlocking FF judgment (PLLULJDG)	A/D converter Compare judgment (ADCJDG)	CE pin level judgment (CEJDG)
	Symbol		SP3 SP2 SP1 SP0	SIO2TS SIO2HIZ SIO2CK1 SIO2CK0		0 0 0 IFCG	0 0 0 PLLUL	0 0 0 ADCCMP	0 0 0 CE
	Read/Write		R/W	R/W		R	R&Reset	R	R
1 (9)	Name	LCD mode selection (LCDMODE)	LCD port selection (LCDPORT)	IF counter mode selection (IFCMODE)	PWM mode selection (PWMMODE)	A/D converter channel control (ADCCH)	PLL unlocking FF delay control (PLULDLY)	Key input judgment (KEYJDG)	Timer carry FF judgment (TMCYJDG)
	Symbol	0 0 KSEN LCDEN	P0YON P0XON P0EON P0FON	IFCMD1 IFCMD0 IFCCK1 IFCCK0	PWM2ON PWM1ON PWM0ON CGPON	ADCCH3 ADCCH2 ADCCH1 ADCCH0	PLULDLY3 PLULDLY2 PLULDLY1 PLULDLY0	0 0 0 KEYJ	0 0 0 TMCY
	Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R&Reset	R&Reset
2 (A)	Name		PLL mode selection (PLLMODE)		IF counter control (IFCCONT)				Port OC group I/O selection (P0CGPIO)
	Symbol		PLLMD3 PLLMD2 PLLMD1 PLLMD0		0 0 IFCSTRT IFCRES				0 0 0 P0CGIO
	Read/Write		R/W		W				R/W
3 (B)	Name		PLL reference mode select (PLRFMODE)				Port A bit I/O selection (P1ABIO)	Port OB bit I/O selection (P0BBIO)	Port OA bit I/O selection (P0ABIO)
	Symbol		PLLRFMD3 PLLRFMD2 PLLRFMD1 PLLRFMD0				P11ABIO3 P11ABIO2 P11ABIO1 P11ABIO0	P0BBIO3 P0BBIO2 P0BBIO1 P0BBIO0	P0ABIO3 P0ABIO2 P0ABIO1 P0ABIO0
	Read/Write		R/W				R/W	R/W	R/W

Fig. 10-4 Structure of a control register (2/2)

8	9	A	B	C	D	E	F
serial I/01 mode selection (SIO1MODE)	Timer mode selection (TMMODE)						Interrupt pin level judgment (INTJDG)
SIO1CH: SB, SIO1MS, SIO1TX	TMMD3, TMMD2, TMMD1, TMMD0						0, 0, INT1, INT0
R/W	R/W						R
Serial I/01 weight control (SIO1WT)							Interrupt judgment selection (INTEDGE)
SBACK, SIO1NWT, SIO1WRQ1, SIO1WRQ0							0, 0, IEG1, IEG0
R/W							R/W
Serial I/01 status judgment (SIO1STUS)						Interrupt permission 1 (INTPM1)	Interrupt permission 2 (INTPM2)
SIO1SF8, SIO1SF9, SBSTT, SBBSY						0, 0, 0, IPIFC	IPSIO1, IPTM, IP1, IP0
R/W						R/W	R/W
Serial I/01 interrupt mode (SIO1INT)	Serial I/01 clock selection (SIO1CLK)					Interrupt request 1 (INTREQ1)	Interrupt request 2 (INTREQ2)
SIO1IMD3, SIO1IMD2, SIO1IMD1, SIO1IMD0	SIO1CK3, SIO1CK2, SIO1CK1, SIO1CK0					0, 0, 0, IRQIFC	IRQSIO1, IRQTM, IRQ1, IRQ0
R/W	R/W					R/W	R/W

Table 10-1 Outline of peripheral hardware control functions of control register (1/4)

Peripheral hardware	Control register: Name	Address	Read/Write	b3 b2 Symbol b1 b0	Peripheral hardware control function: Function outline	Value set	At Reset: Power on	STOP	CE
Stack	Stack pointer SP	01H	R/W	(SP3)	0		7	7	7
				(SP2)	Stack pointer				
				(SP1)					
				(SP0)					
Timer	Timer mode selection (TMMODE)	09H	R/W	TMMD3	Setting timer interrupt time	0 0 1 1 100 ms 250 ms 5 ms 1 ms 0 1 1 1	0	0	Retained
				TMMD2					
				TMMD1	Setting timer carry FF time	0 0 1 1 100 ms 250 ms 5 ms 1 ms 0 1 1 1			
				TMMD0					
	Timer carry FF judgment (TMCYJDG)	17H	Read & Reset	0			0	1	1
				0					
				0					
				TMCY	Timer carry FF detection	0 : FF reset 1 : FF set			
Interrupt	Interrupt edge selection (INTEDGE)	1FH	R/W	0			0	0	0
				0					
				IEG1	INT1 pin Setting interrupt	0 : Upload 1 : Download			
				IEG0	INT0 pin issuing edge				
	Interrupt permission 1 (INTPM1)	2EH	R/W	0			0	0	0
				0					
				0					
				IPIFC	IF counter	0 : Prohibited 1 : Permitted			
	Interrupt permission 2 (INTPM2)	2FH	R/W	IPSIO1	Serial interface 1				
				IPTM	Timer				
				IP1	INT_1 pin				
				IP0	INT_0 pin				
	Interrupt request 1 (INTREQ1)	3EH	R/W	0			0	0	0
				0					
				0					
				IRQIFC	IF counter	0 : Not requested 1 : Requested Reset when interrupt is accepted			
	Interrupt request 2 (INTREQ2)	3FH	R/W	IRQSIO1	Serial interface 11				
				IRQTM	Timer				
				IRQ1	INT_1 pin				
				IRQ0	INT_0 pin				
Pin	CE pin level judgment (CEJDG)	07H	R	0			–	–	–
				0					
				0					
				CE	CE pin	0 : Low level 1 : High level			
	Interrupt pin level judgment (INTJDG)	0FH	R	0					
				0					
				INT1	INT_1 pin				
				INT0	INT_0 pin				

Detects interrupt request

Detects the stotus

Table 10-1 Outline of peripheral hardware control functions of control register (2/4)

Peripheral hardware	Control register: Name	Address	Read/Write	b3 b2 Symbol b1 b0	Peripheral hardware control function: Function outline	Value set	At Reset: Power On	STOP	CE
PLL frequency synthesizer	PLL unlocking FF jubgment (PLLULJDG)	05H	R	0 0 0 PLLUL	Detects the unlocking FF status	0 : Lock 1 : Unlock Read & Reset	0	Retained	Retained
	PLL unlocking FF delay control (PLULDLY)	15H	R/W	PLULDLY3 PLULDLY2 PLULDLY1 PLULDLY0	0 0 Sets unlocking FF set delay time	0 0 1 1 1 μs 2 μs 0.5 μs Disable 0 1 0 1	0	0	Retained
	PLL mode selection (PLLMODE)	21H	R/W	PLLMD3 PLLMD2 PLLMD1 PLLMD0	0 0 Sets the PLL division method	0 0 1 1 Disable MF VHF HF 0 1 0 1	0	0	0
	PLL reference mode selection (PLRFMODE)	31H	R/W	PLLRFMD3 PLLRFMD2 PLLRFMD1 PLLRFMD0	Sets PLL reference frequency	0 : 1.25 1 : 2.5 2 : 5 3 : 10 4 : 6.25 5 : 12.5 6 : 25 7 : 50 8 : 3 9 : A : B : Setting prohibited C : 1 D : 9 E : 100 F : OFF	F	F	Retained
A/D convertar	A/D convertar channel selection (ADCCH)	14H	R/W	ADCCH3 ADCCH2 ADCCH1 ADCCH0	0 Selects pin used as A/D converter	0 : AD_0 1 : AD_1 2 : AD_2 3 : AD_3 4 : AD_4 5 : AD_5 6 : 7 : Input port	7	7	7
	A/D converter Compare judgment (ADCJDG)	06H	R	0 0 0 ADCCMP	Datects A/D converter comparison result	0 : $V_{REF} > V_{ADCIN}$ 1 : $V_{REF} < V_{ADCIN}$	Undefined	Retained	Retained
General purpose port	Port OC group I/O selection (P0CGPIO)	27H	R/W	0 0 0 P0CGIO	$P0C_3$ Input/output setting of $-P0C_0$ pins (four pins concurrently)	0 : Input 1 : Output	0	0	0
	Port 1A bit I/O selection (P1ABIO)	35H	R/W	P1ABIO3 P1ABIO2 P1ABIO1 P1ABIO0	$P1A_3$ pin $P1A_2$ pin $P1A_1$ pin $P1A_0$ pin				
	Port OB bit I/O selection (P0BBIO)	36H	R/W	P0BBIO3 P0BBIO2 P0BBIO1 P0BBIO0	$P0B_3$ pin $P0B_2$ pin $P0B_1$ pin $P0B_0$ pin Sets input/output (each pin)	0 : Input 1 : Output	0	0	0
	Port OA bit I/O selection (P0ABIO)	37H	R/W	P0ABIO3 P0ABIO2 P0ABIO1 P0ABIO0	$P0A_3$ pin $P0A_2$ pin $P0A_1$ pin $P0A_0$ pin				
D/A converter	PWM mode selection (PWMMODE)	13H	R/W	PWM2ON PWM1ON PWM0ON CGPON	PWM_2 pin PWM_1 pin PWM_0 pin Set as a D/A converter CGP pin is set as CGP	0 : General purpose output port 1 : D/A converter 0 : General purpose output port 1 : CGP	0	0	Retained

Table 10-1 Outline of peripheral hardware control functions of control register (3/4)

Peripheral hardware	Control register: Name	Address	Read/Write	b3 b2 Symbol b1 b0	Peripheral hardware control function: Function outline	Value set	At Reset: Power On	STOP	CE
Serial interface	Serial I/O2 mode selection (SIO2MODE)	02H	R/W	SIO2TS	Sets the starting of serial interface 2	0 : No operation 1 : Start	0	0	0
				SIO2HIZ	Sets $P0B_1/SO_2$ pin	0 : General purpose port 1 : Serial count			
				SIO2CK1	Sets clock of serial interface 2	0 0 1 1 0 External section 75 kHz 150 kHz 450 kHz 0 1 0 1			
				SIO2CK0					
	Serial I/O1 mode selection (SIO1MODE)	08H	R/W	SIO1CH	Sets 2-wire system or 3-wire system	0 0 1 1 SBI SB SIO1 Setting Prohibited 0 1 0 1	0	0	0
				SB	Sets 2-wire system				
				SIO1MS	Sets clock direction	0 : External clock 1 : Internal clock			
				SIO1TX	Sets input/output	0 : Input 1 : Output			
	Serial I/O1 wait control (SIO1WT)	18H	R/W	SBACK	Sets and detect t 2-wire system SB acknowledgement	Setting and detection of 0 and 1	0	0	0
				SIO1NWT	Sats and detects wait permission	0 : Permitted 1 : Release			
				SIO1WRQ1	Sets wait timing of serial interface 1	0 0 1 1 No wait 8 clocks 9 clocks SB 8 clocks 0 1 0 1			
				SIO1WRQ0					
	Serial I/O1 status judgment (SIO1STUS)	28H	R/W	SIO1SF8	Detects clock counter of serial interface 1	Set by clock counter 8, reset by 0 or 1	0	0	0
				SIO1SF9		Set by clock counter 9, reset by 0 or 1			
				SBSTT	Detects 2-wire system clock count	Set until start condition 9th clock			
				SBBSY	Detects start and stop conditions of 2-wire system SB	Set until start condition – stop condition			
	Serial I/O1 interrupt mode (SIO1INT)	38H	R/W	SIO1IMD3	0	0 0 7th clock 8th clock 0 1	Undefined	Retained	Retained
				SIO1IMD2	0				
				SIO1IMD1	Sets serial interface 1 interrupt condition	1 7th clock after the start 1 Stop 0 condition 1 condition			
				SIO1IMD0					
	Serial I/O1 clock selection (SIO1CLK)	39H	R/W	SIO1CK3	0		Undefined	Retained	Retained
				SIO1CK2	0				
				SIO1CK1	Sets internal clock of serial interface 1	0 0 1 1 75 kHz 150 kHz 225 kHz 450 kHz 0 1 0 1			
				SIO1CK0					
Frequency counter	IF counter selection (IFCGJDG)	04H	R	0			0	–	–
				0					
				0					
				IFCG	Detects the opening and closing of frequency counter gate	0 : Closed 1: Opened			
	IF counter mode selection (IFCMODE)	12H	R/W	IFCMD1	Sets frequency counter mode	0 0 1 1 CGP FMIF AMIF FCG 0 1 0 1	0	0	Retained
				IFCMD0					
				IFCCK1	Sets frequency counter gate time	0 1 ms 0 4 ms 1 8 ms 1 Opened 0 1 kHz 1 100 kHz 1 900 kHz 1			
				IFCCK0					
	IF countar control (IFCCONT)	23H	W	0			0	0	Retained
				0					
				IFCSTRT	Specifies count start of frequency counter	0 : NOP instruction 1 : Start			
				IFCRES	Specifies data reset of frequency counter	0 : NOP instruction 1 : Reset			

Table 10-1 Outline of peripheral hardware control functions of control registers (4/4)

Peripheral hardware	Control register: Name	Address	Read/Write	b3 b2 Symbol b1 b0	Peripheral hardware control function: Function outline	Value set	At Reset: Power On	STOP	CE
LCD driver	LCD mode selection (LCDMODE)	10H	R/W	0			0	0	Retained
				0					
				KSEN	Sets key source signal output	0 : Key source OFF 1 : Key source ON			
				LCDEN	Setting LCD display output	0 : Display OFF 1 : Display ON			
	LCD port selection (LCDPORT)	11H	R/W	P0YON	Pins $P0Y_0$ to $P0Y_{15}$ } Each pin is set as a general purposs output port	0 : LCD segment 1 : General purpose output port	0	0	Retained
				P0XON	Pins $P0X_0$ to $P0X_5$				
				P0EON	Pins $P0E_0$ to $P0E_3$				
				P0FON	Pins $P0F_0$ to $P0F_3$				
	Key input judgment Sets K (KEIJDG)	16H	Read & Reset	0			0	0	0
				0					
				0					
				KEYJ	Detects LCD key source input latch	0 : Without latch 1 : With latch			

10.4 NOTES ON USING A REGISTER FILE

10.4.1 Notes on Control Register Manipulation (Write Only, Read Only, and Unused Registers)

Cautions are necessary on the use of the 17K series Assembler (AS17K) and Emulator (IE-17K) as described below when a Write Only register (W), Read Only register (R), and unused control registers (addresses 00H to 3FH of the register file) are utilized.

(1) Device operation

When a Write Only register is read, an "undefined value" is read.

No change is made even if Write operation is performed for a Read Only register.

When an unused register is read, an "undefined value" is read and no change is made even if Write operation is performed.

(2) When using Assembler (AS17K)

An "error" occurs in the instruction for reading the Write Only register.

An "error" occurs in the instruction for writing data to the Read Only register.

When data is written to or read from an unused register, an "error" occurs for the instruction used for reading or writing data.

(3) When using an Emulator (IE-17K) (manipulated in batch processing, etc.)

When Read operation is performed for a Write Only register, an "undefined" value is read. No "error" occurs.

When Write operation is performed for a Read Only register, no change is made. No "error" occurs.

When Read operation is performed for an unused register, an "undefined value" is read, and no change is made even if Write operation is performed. No "error" occurs.

10.4.2 Register File Symbol Definition and Reserved Word

An "error" occurs when 17K series Assembler (AS17K) is used and when a register file address is directly entered, using a numeric value, to operand "rf" of the "PEEK WR, rf" or "POKE rf, WR" instruction.

Consequently, the address of the register file must be defined as a symbol in advance as shown in Example 1.

Example 1:

When an error occurs

```
PEEK   WR,  02H ;
POKE   2H,  WR  ;
```

When an error does not occur

```
RF51   MEM  0.51H ; Symbol definition
PEEK   WR,  RF51  ;
```

The following points must be noted in this case.

When the symbol of the control register (addresses from 00H to 3FH) is defined as a data address type, it must be defined as addresses from 80H to BFH of BANK0 as shown in Example 2.

Since a control register in a register file can be manipulated via a window register, it is used for generating an "error" in Assembler when it is manipulated by any instructions other than the "PEEK" and "POKE" instructions.

However, symbol definition is possible for register files (addresses from 40H to 7FH) which overlap the data memory without any changes.

Example 2:

```
RF51    MEM    1.51H  ;  Register file which overlaps the data memory
RF02    MEM    0.82H  ;  Control register

BANK0
PEEK    WR,    RF51   ;  RF51 is used as the data memory of "address 51H of BANK0".
PEEK    WR,    RF02   ;  RF02 is used as address 02H of the control register.

BANK1
PEEK    WR,    RF51   ;  RF51 is used as the data memory of "address 51H of BANK1"
PEEK    WR,    RF02   ;  RF02 is used as address 02H of the control register.
```

When Assembler (AS17K) is used, the following macro instructions are incorporated in Assembler as flag type symbol manipulation instructions.

SETn : Sets "1" in the flag.
CLRn : Resets the flag to "0".
SKTn : Skips when all the flags are set to "1".
SKFn : Skips when all the flags are set to "0".
NOTn : Inverts the flag.
INITFLG: Initializes the flag.

By using these built-in macro instructions, the contents of the register file can be manipulated in bit units.

Since many flags of a control register are manipulated in bit units, a "reserved word" is defined in advance as a flag type symbol under Assembler (AS17K).

However, no flag type reserved word is available for a stack pointer. The stack pointer reserved word is defined by "SP" as data memory.

10.4.3 Notes on Using Assembler (AS17K) Built-in Macro Instructions

The points described in Sections (1) and (2) must be noted when Assembler built-in macro instructions are used for control registers.

(1) Built-in macro instructions for a stack pointer

As described in Section **10.4.2**, flag manipulation instructions cannot be used through a reserved word because no flag type reserved word is defined for a stack pointer.

(2) Built-in macro instructions for a Write Only register

No flag built-in flag manipulation macro instructions can be used for a Write Only register.

If the "SETn" built-in macro instruction is used for a Write Only register as shown in the example below, the contents of the register file are read to a window register.

In this case, the value read in the window register becomes undefined (if Read operation is performed for a Read Only register file, the value becomes undefined), and an undefined value is written to the bit which is not specified by the "SETn" instruction.

In this case, Assembler (AS17K) generates an error. The IF counter control register (IFCCONT: address 23H) is available as a Write Only register; Assembler built-in macro instructions must not be used for the IF counter control register (IFCRES flag and IFCSTRT flag).

Example:

```
SET1    IFCRES ; Sets (1) IFCRES (flag type symbol for indicating bit b0 of the IF counter control
               ; register).
Macro expansion
; ①
        PEEK  WR,   23H     ; Reads the contents of address 23H of the control register into WR.
; ②
        OR    WR,   #0001B  ; Sets (1) bit b0 of WR.
; ③
        POKE  23H,  WR      ; Writes the content of WR to address 23H of the control register.
```

When the above instructions are executed at point ①, bit b_1 (IFCSTRT flag) of WR becomes undefined and as a result, bit b_1 (IFCSTRT flag) of WR which is written by ③ remains undefined also.

However, in Assembler (AS17K), an error occurs in ①.

11. DATA BUFFER (DBF)

A data buffer is used for transferring data with peripheral hardware and reading table reference data.

11.1 STRUCTURE OF A DATA BUFFER

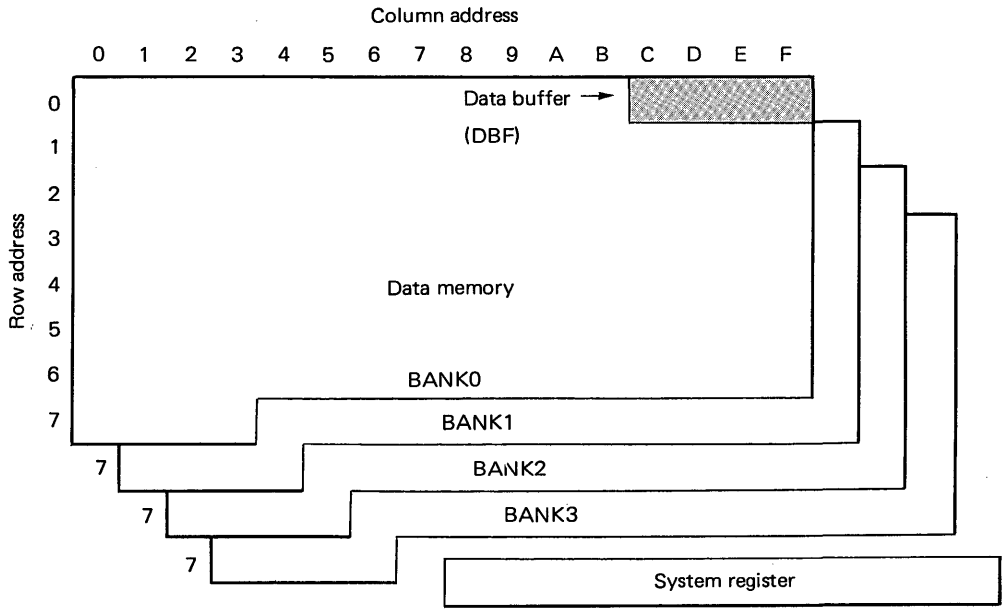
11.1.1 Location of a Data Buffer in Data Memory

Fig. 11-1 shows the location of a data buffer in data memory.

As shown in Fig. 11-1, a data buffer (DBF) is allocated in the addresses from 0CH to 0FH of BANK0 of data memory and consists of 16 bits; 4 words x 4 bits.

Since data buffer is located in data memory, it can be manipulated by data memory manipulation instructions.

Fig. 11-1 Location of a data buffer



11.1.2 Structure of a Data Buffer

Fig. 11-2 shows the structure of a data buffer.

As shown in Fig. 11-2, a data buffer consists of 16 bits using bit b_0 of data memory address 0FH of the data as LSB and bit b_3 of data memory address as MSB.

Fig. 11-2 Structure of a data buffer

Data memory	Address	0 CH				0 DH				0 EH				0 FH			
	Bit	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
Data buffer	Bit	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
	Symbol	DBF 3				DBF 2				DBF 1				DBF 0			
	Data	^ M S B v				Data								^ L S B v			

11.2 Function of a Data Buffer

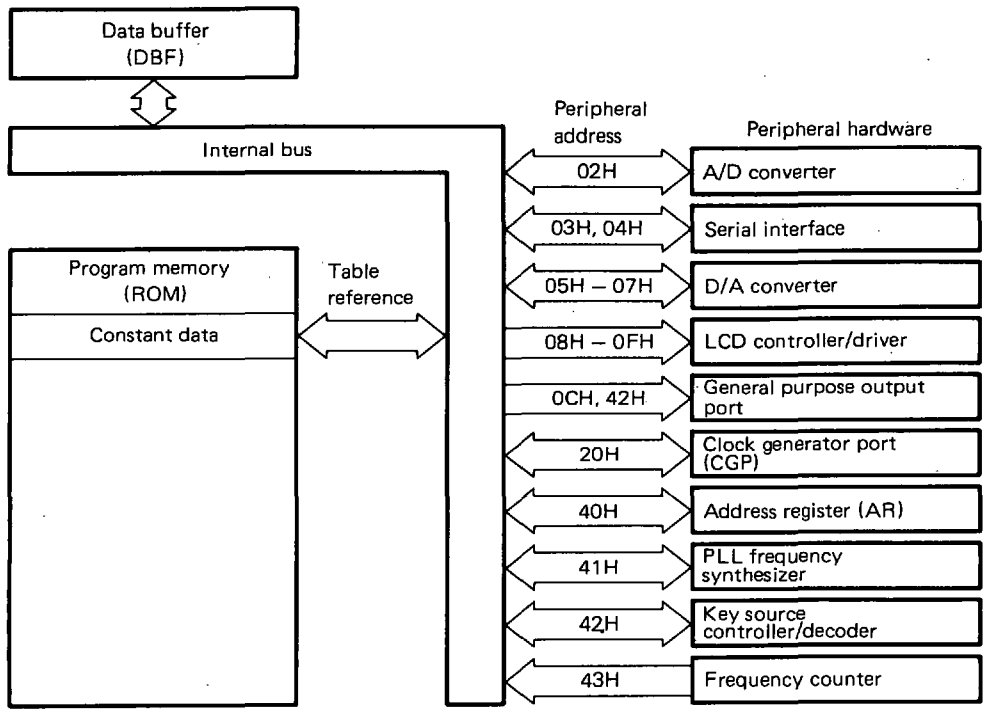
A data buffer has two functions as described in (1) and (2).

(1) Reads constant data in the program memory (table reference)
(2) Transfers data with peripheral hardware

Fig. 11-3 shows the relationship between data buffer, peripheral hardware, and table reference.

Sections **11.3** and **11.4** describe table reference, and the relationship with each peripheral hardware device, respectively.

Fig. 11-3 Relationship between a data buffer, peripheral hardware, and table reference



11.3 DATA BUFFER AND TABLE REFERENCE

11.3.1 Table Reference Operation

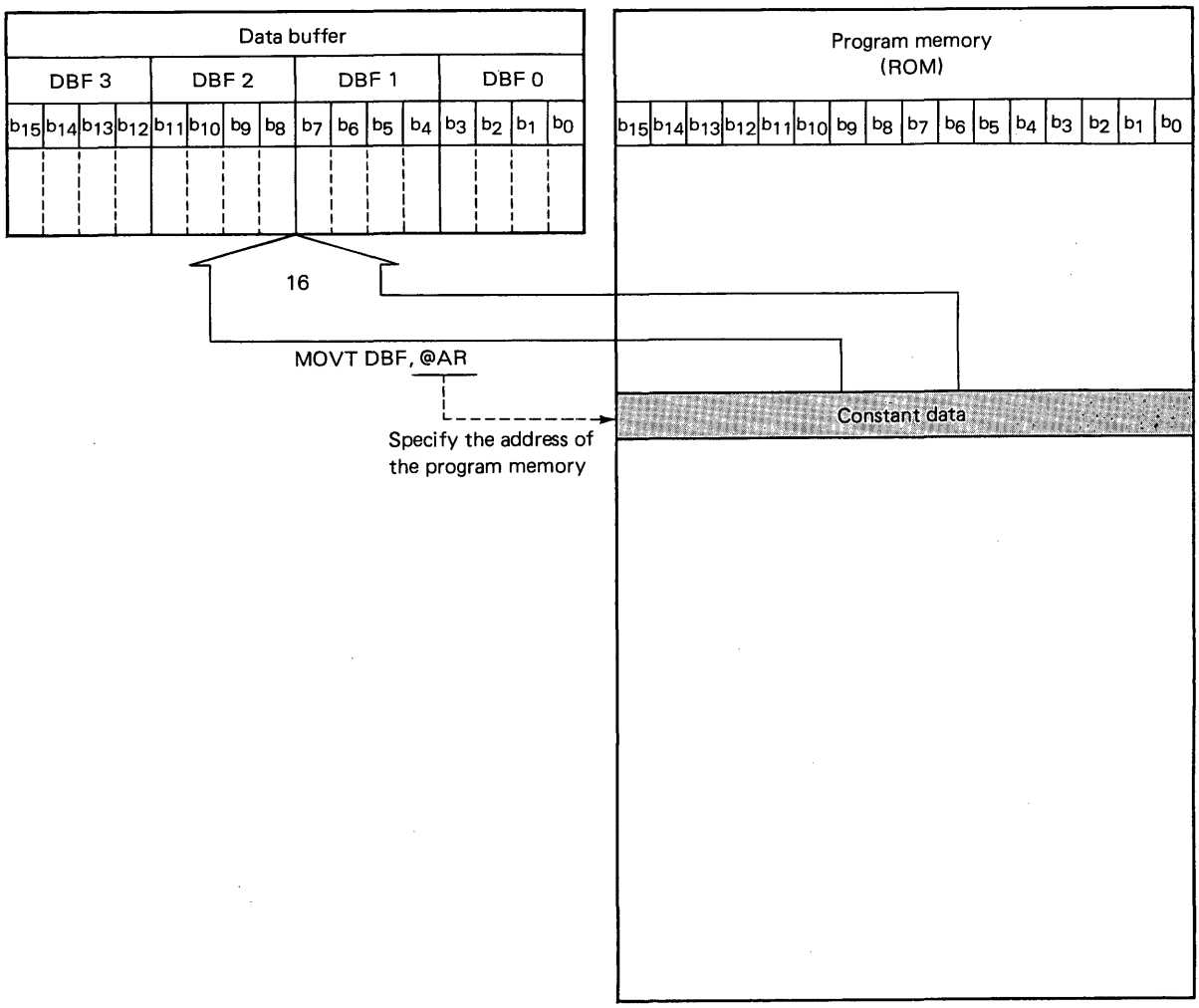
For table reference, constant data in program memory can be read into a data buffer by using the "MOVT DBF, @AR" instruction.

Consequently, a complicated data conversion program is no longer required if constant data such as display data and division value of PLL frequency synthesizer is written to program memory and the data in the table is referenced whenever required.

The "MOVT" instruction is described below.

Section **11.3.2** shows the program example.

MOVT DBF, @AR ; Reads the contents of the program memory whose address is specified by the contents of the address register into a data buffer as shown below.



At execution of a table reference instruction, one level of stack is used.

Since 13 bits of the address register (AR) are valid, 7932 steps of program memory addresses 0000H to 1EFBH can be used for table reference.

See **5, "Stack"** and Section **9.3 "Address Register (AR)"** also.

11.3.2 Examples of Table Reference Programs

Examples 1, 2, and 3 show table reference program examples.

Example 1:

```
M000    MEM    0.00H ;
P0A     MEM    0.70H ;
P0B     MEM    0.71H ;
P0C     MEM    0.72H ;

START:                ;  Program address 0000H
  BR    MAIN
DATA:
  DW    0001H         ;  Constant data
  DW    0002H         ;
  DW    0004H         ;
  DW    0008H         ;
  DW    0010H         ;
  DW    0020H         ;
  DW    0040H         ;
  DW    0080H         ;
  DW    0100H         ;
  DW    0200H         ;
  DW    0400H         ;
  DW    0800H         ;
MAIN:
  BANK0               ;  Built-in macro
  SET4  P0ABI03, P0ABI02, P0ABI01, P0ABI00
  SET4  P0BBI03, P0BBI02, P0BBI01, P0BBI00
  SET1  P0CGIO
  MOV   M000, #0
  MOV   RPH,  #000B   ;  Sets the general register in row
  MOV   RPL,  #1110B  ;  address 7H of BANK0.
LOOP:
  MOV   AR3,  #(.DL.DATA SHR 12 AND 0FH)
  MOV   AR2,  #(.DL.DATA SHR 8 AND 0FH)
  MOV   AR1,  #(.DL.DATA SHR 4 AND 0FH)
  MOV   AR0,  #(.DL.DATA SHR 0 AND 0FH)
                      ;  Sets address register (AR) in 0001H.
; ①
  MOVT  DBF,  @AR     ;  Transfers the ROM value specified by the content of AR to the data buffer.
; ②
  LD    P0A,  DBF2    ;  Transfers the data buffer values to port data registers, Port0A (70H),
  LD    P0B,  DBF1    ;
  LD    P0C,  DBF0    ;  Port0B (71H), and Port0C (72H).
  ADD   M000, #1      ;  Increments the content of the address register by 1.
  ADD   AR0,  M000
  ADDC  AR1,  #0
  ADDC  AR2,  #0
  ADDC  AR3,  #0
```

```
    SKNE   M000,  #0CH  ; Writes 0 in M000 when the value of M000 becomes 0CH.
    MOV    M000,  #0    ;
    BR     LOOP
```

When the above program is executed, constant data stored in program memory addresses from 0001H to 000CH is read into data buffers sequentially in ① and output to Port 0A, Port 0B, and Port 0C, in ②.

Since the values which are shifted to the left by 1 bit are stored as constant data, High level is output to each of the pins of Port 0A, Port 0B, and Port 0C sequentially.

In Example 1, the starting address on the program memory which stores constant data is set in the address register using the "MOV" instruction.

As a result, when the "MOV" instruction is used, the starting address of each constant data must be set in the address register many times when a large amount of constant data is to be stored.

Consequently, the program shown in Example 2 is useful when the number of steps may increase as a result of many "MOV" statements or when the program is to managed under a common routine.

Example 2:

```
    M000   MEM    0.00H ;
START:
    BR     MAIN          ;
DATAFETCH:
    DI                   ;
    POP    AR            ; Reads the content of the address stack register to the address register.
    ADD    AR0,   M000   ; In this case, the stack pointer indicates the MAIN routine return
    ADDC   AR1,   #0     ; address.
    ADDC   AR2,   #0     ;
    ADDC   AR3,   #0     ; Shifts by the constant data address specified by the content of M000.
    MOVT   DBF,   @AR    ; Reads constant data.
    EI
    RET                  ; Returns to the MAIN routine.
DATA1:
    CALL   DATAFETCH     ; Calls the common processing routine.
    DW     0123H         ; In this case, the address of DATA1+1 is saved in the address stack
    DW     4567H         ; register.
           :
    DW     89ABH         ;
DATA2:
    CALL   DATAFETCH     ; Calls the common processing routine.
    DW     1357H         ; In this case, the address of DATA2+1 is saved in the address stack
    DW     2468H         ; register.
           :
    DW     9BDFH         ;
MAIN
    BANK0                ; Built-in macro
    SET4   P0ABI03, P0ABI02, P0ABI01, P0ABI00
    SET4   P0BBI03, P0BBI02, P0BBI01, P0BBI00
    SET1   P0CGI0
    MOV    M000, #0
    MOV    RPH,  #0000B ; Sets the general register to row
    MOV    RPL,  #1110B ; address 7H of BANK0.
```

```
LOOP:
  CALL   DATA1          ;  Reads the value of constant data DATA1 specified by the content of M000.
  LD     P0A,  DBF2     ;  Transfers data buffer values to port registers, Port0A (70H), Port0B (71H), and
  LD     P0B,  DBF1     ;  Port0C (72H).
  LD     P0C,  DBF0
  CALL   DATA2          ;  Reads the value of constant data DATA2 specified by the content of M000.
  LD     P0A,  DBF2     ;  Transfers the data buffer values to port data registers,
  LD     P0B,  DBF1     ;  Port0A (70H), Port0B (71H), and Port0C (72H).
  LD     P0C,  DBF0     ;  Port0C (72H).
  ADD    M000, #1       ;
  SKNE   M000, #0CH     ;  Writes 0 in M000 when the content of M000
  MOV    M000, #0       ;  becomes 0CH.
  BR     LOOP
```

In Example 2, two levels of stacks are required for executing the "CALL" instruction twice, and the "POP" and "MOVT" instructions.

As shown in Example 3, the "CALL" instruction has to be executed only once. In this case also, two levels of stacks are required for the "MOVT" instruction.

Example 3:

```
DATAFETCH:
  DI                    ;
  POP    AR             ;  Reads the content of the address stack register to the address register.
  MOVT   DBF,  @AR      ;  Transfers the constant data storage address to the data buffer.
  INC    AR             ;  Stores the MAIN routine return address.
  PUSH   AR             ;
  PUT    AR,   DBF      ;  Transfers the constant data storage address to the address register.
  ADD    AR0,  M000     ;  Shifts by the constant data address specified by the content of
  ADDC   AR1,  #0       ;  M000.
  ADDC   AR2,  #0       ;
  ADDC   AR3,  #0       ;
  MOVT   DBF,  @AR      ;  Reads the constant data.
  EI
  RET                   ;  Returns control to the MAIN routine.
DATA1:
  DW     0123H          ;  Constant data.
                        ;
DATA2:
  DW     1357H          ;  Constant data.
                        ;
MAIN:
LOOP:
  CALL   DATAFETCH      ;
  DW     .DL.DATA1      ;
  LD     P0A, DBF2      ;
  CALL   DATA2          ;
  DW     .DL.DATA2      ;
  LD     P0A, DBF2      ;
  BR     LOOP
```

11.4 BUFFER DATA AND PERIPHERAL HARDWARE

11.4.1 Peripheral Hardware Control Method

Peripheral hardware devices used for data transfer performed via data buffers are listed below.

- A/D converter
- Serial interface
- D/A converter
- LCD controller/driver
- Output port
- Clock generator port (CGP)
- PLL frequency synthesizer
- Key source controller/decoder
- Frequency counter
- Address register (AR)

Each peripheral hardware device is controlled by sending data to the device or reading data from the device via a data buffer.

A data transfer register (called a peripheral register) is available for each peripheral hardware device and an address (called a peripheral address) is assigned to each peripheral register.

By executing the specific instructions, "GET" and "PUT" for a peripheral hardware register, data transfer between a data buffer and each peripheral hardware device is enabled.

The "GET" and "PUT" instructions are described below. Table 11-1 lists the functions of peripheral hardware devices and data buffers.

GET DBF, p : Reads data of the peripheral register addressed by p into a data buffer.
PUT p, DBF : Sets data of the data buffer to the peripheral register addressed by p.

Peripheral registers include a Read/Write register (PUT/GET), Write Only register (PUT), and Read Only register (GET).

In this case, if the "GET" and "PUT" instruction are executed for a Write Only (PUT only) and Read Only (GET only) peripheral registers, the following results are produced in terms of the device.

- When a Read (GET) instruction is executed for a Write Only (PUT only) peripheral register, an undefined value is read.
- When a Write (PUT) instruction is executed for a Read Only (GET only) peripheral register, no influence is given.

However, cautions are necessary when the 17K series Assembler (AS17K) or Emulator (IE-17K) is used. See Section **11.5, "Notes on Using a Data Buffer"** for details.

See **16** to **24** for details of peripheral registers.

11.4.2 Notes on Data Transfer with Peripheral Registers

Data transfer between a data buffer and each peripheral register is performed in 8-bit units or 16-bit units.

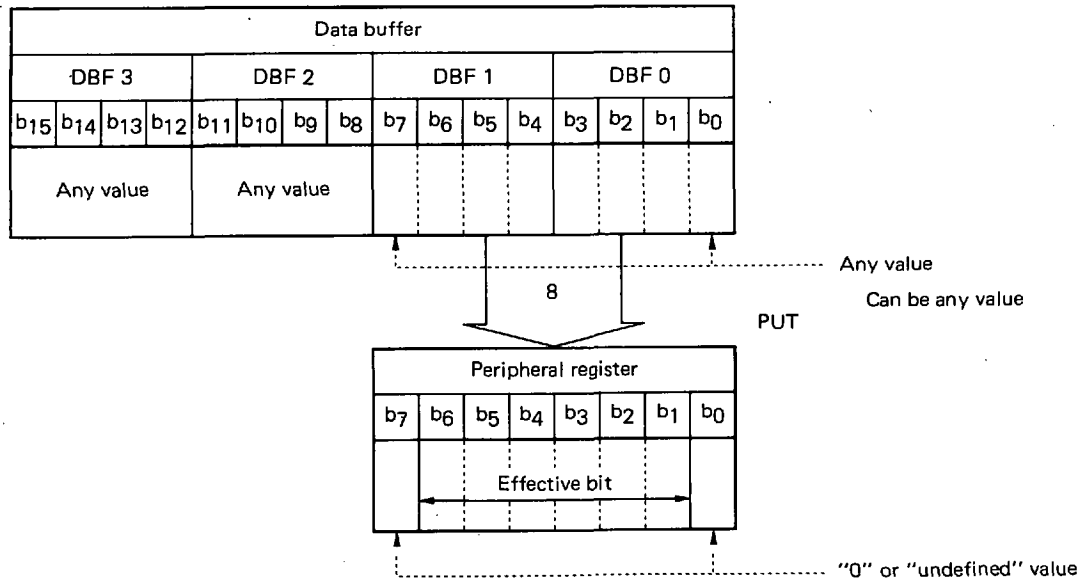
In this case, an execution time of only one instruction (4.44 μs) is required for the "PUT" or "GET" instruction even if the data unit is 16 bits.

When the effective data bit length of the peripheral register is 7 bits and when 8-bit data transfer is performed, 1 bit is handled as excess data.

This excess data becomes "any (any value)" data at Write operation and becomes an "undefined value" at Read operation as shown in Examples 1 and 2.

Example 1:

At execution of the "PUT" instruction (the effective bits of the peripheral register are 6 bits, bit b_1 to bit b_6)

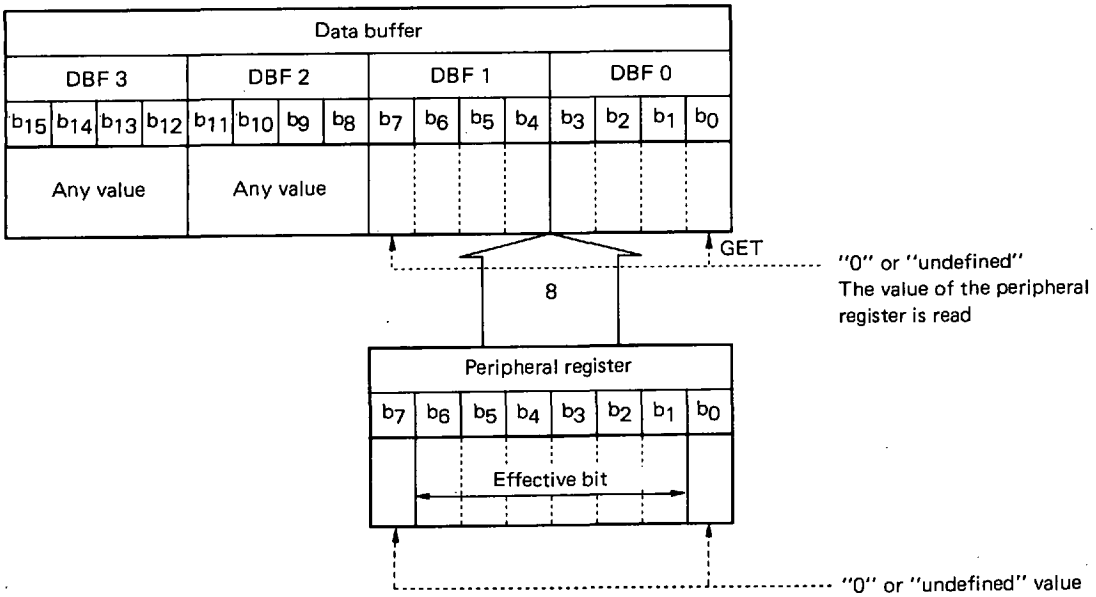


When 16-bit data is written to a peripheral register, the high-order 8 bits of the data buffer (contents of DBF3 and DBF2) are handled as "any value" (can be any value).

Each bit which does not correspond to any of the effective bits of the peripheral register among the 8-bit data of the data buffer is handled as "any value".

Example 2:

At excecution of the "GET" instruction



When the 8-bit data of the peripheral register is read, the value of the high-order 8 bits of the data buffer (DBF3 and DBF2) remains unchanged.

Of the 8-bit data of the data buffer, each bit which is not an effective bit of the peripheral register is determined to be "0" or "undefined". The value, "0" or "undefined" is determined by the peripheral register in advance.

11.4.3 Statuses when Peripheral Registers are reset

The effective bits of each peripheral register are set as follows at resetting.

Reset	Status of the effect bit
Power On	Undefined
Clock Stop	The previous status is retained
CE	The previous status is retained

Table 11-1 Relationship between peripheral hardware and data buffer (1/2)

Peripheral hardware		Peripheral register used for data transfer with data buffer			
		Name	Symbol	Peripheral address	PUT/GET instruction
A/D converter		A/D converter register	ADCR	02H	PUT/GET
Serial interface	Serial interface 2 (SIO2)	Presetable shift register 2	SIO2SFR	03H	PUT/GET
	Serial interface 1 (SB, SBI, SIO1)	Presetable shift register 1	SIO1SFR	04H	
D/A converter (PWM output)	PWM_0 pin	PWM data register 0	PWMR0	05H	PUT/GET
	PWM_1 pin	PWM data register 1	PWMR1	06H	
	PWM_2 pin	PWM data register 2	PWMR2	07H	
LCD controller/ driver	LCD segment group 0	LCD segment group data register 0	LCDR0	08H	PUT
	LCD segment group 1	LCD segment group data register 1	LCDR1	09H	
	LCD segment group 2	LCD segment group data register 2	LCDR2	0AH	
	LCD segment group 3	LCD segment group data register 3	LCDR3	0BH	
	LCD segment group 4	LCD segment group data register 4	LCDR4	0CH	
	LCD segment group 5	LCD segment group data register 5	LCDR5	0DH	
	LCD segment group 6	LCD segment group data register 6	LCDR6	0EH	
	LCD segment group 7	LCD segment group data register 7	LCDR7	0FH	
Output port	Port 0X	P0X group data register	P0X	0CH	PUT
	Port 0Y	P0Y group data register	P0Y	42H	PUT/GET
Clock generator port (CGP)		CGP data register	CGPR	20H	PUT/GET
Address register (AR)		Address register	AR	40H	PUT/GET
PLL frequency synthesizer		PLL data register	PLLR	41H	PUT/GET
Key source controller/decoder		Key source data register	KSR	42H	PUT/GET
Frequency counter		IF counter data register	IFC	43H	GET

Table 11-1 Relationship between peripheral hardware and data buffer (2/2)

Function		
Data buffer input/output bit count	Effective Number of bits	Outline
8	6	Sets the comparative voltage REF data of the A/D converter $V_{REF} = \frac{x - 0.5}{64} \times V_{DD}, 1 \leq x \leq 63$
8	8	Sets Serial Out data and reads Serial In data.
8	8	Sets duty of the D/A converter output signal Duty D $= \frac{x + 0.25}{256} \times 100\%, 0 \leq x \leq 255$ Frequency f = 878.9 Hz
8	7	LCD segment group 0
	4	LCD segment group 1
	7	LCD segment group 2
	7	LCD segment group 3
	7	LCD segment group 4
	3	LCD segment group 5
	7	LCD segment group 6
	7	LCD segment group 7
		Sets display data of each group 0: Display ON 1: Display OFF
8	8	Sets output data of Port 0X 0: Low level 1: High level
16	16	Sets output data of Port 0Y 0: Low level 1: High level
8	7	Sets frequency of SG function Frequency f $= \frac{18}{2\,(2 \times x)}$ kHz and Sets duty of the VDP function Duty D $= \frac{x + 2}{67}, 0 \leq x \leq 63$
16	16	Data transfer with address register
16	16	Sets PLL division value (N value)
16	16	Sets output data of the key source signal
16	16	Reads the discrete value of the frequency counter

11.5 NOTES ON USING DATA BUFFERS

11.5.1 Notes on Data Buffer Manipulation of a Write Only Register, Read Only Register, and Unused Addresses

The following cautions are necessary for an unused peripheral address, a Write Only peripheral register (PUT only), and Read Only peripheral register (GET only) on the use of the 17K series Assembler (AS17K) and Emulator (IE-17K) in terms of device operation when data transfer is performed with peripheral hardware via a data buffer.

(1) Device operation

When a Write Only register is read, an "undefined value" is read.

No change is made even if Write operation is performed for a Read Only register.

When an unused address is read, an "undefined value" is read and no change is made even if Write operation is performed.

(2) When using Assembler (AS17K)

An "error" occurs in the instruction for reading the Write Only register.

An "error" occurs in the instruction for writing data to the Read Only register.

An "error" occurs in the instructions for reading and writing the unused address.

(3) When using an Emulator (IE-17K) (an instruction is executed in batch processing, etc.)

When a Write Only register is read, an "undefined value" is read.

No change is made even if Write operation is performed for a Read Only register.

When an unused address is read, an "undefined" value is read and no change is made even if a Write operation is performed. No "error" occurs.

11.5.2 Addresses of Peripheral Registers and Reserved Words

As shown in Example 1, an "error" does not occur even if peripheral address "p" which is specified by the "PUT p, DBF" instruction or "GET DBF, p" instruction is specified directly (using a numeric value) when 17K series Assembler (AS17K) is used.

However, this method is not recommended for program debugging.

Consequently, a symbol must be defined for peripheral addresses using a symbol definition pseudo instruction, which is an Assembler pseudo instruction as shown in Example 2.

To simplify symbol definition, a peripheral address is defined in Assembler (AS17K) as a "reserved word".

By using the reserved word, a program can be written without symbol definition as shown in Example 3.

See the column "Symbol" in Table 11-1 for reserved words of peripheral registers.

See also **26, "μPD17005 Reserved Words"** for reserved words.

Example 1:

```
PUT   02H,  DBF   ; Even if a peripheral address is specified directly by 02H or 03H, an error does
GET   DBF,  03H   ; not occur in Assembler. However this may increase the likelihood of program
                  ; bugs occurring.
```

Example 2:

```
PLLDATA DATA  41H ; Assigns PLLDATA to 04H using a symbol definition pseudo
PUT     PLLDATA, DBF ; instruction.
```

Example 3:

```
PUT   PLLR, DBF   ; Symbol definition is not required if reserved word "PLLR" is used.
```

12. INTERRUPT

An interrupt passes program control to a predetermined address (called a vector address) after stopping the program which is currently being executed according to the request from peripheral hardware (INT_0 pin, INT_1 pin, timer, serial interface 1, or frequency counter).

12.1 STRUCTURE OF AN INTERRUPT BLOCK

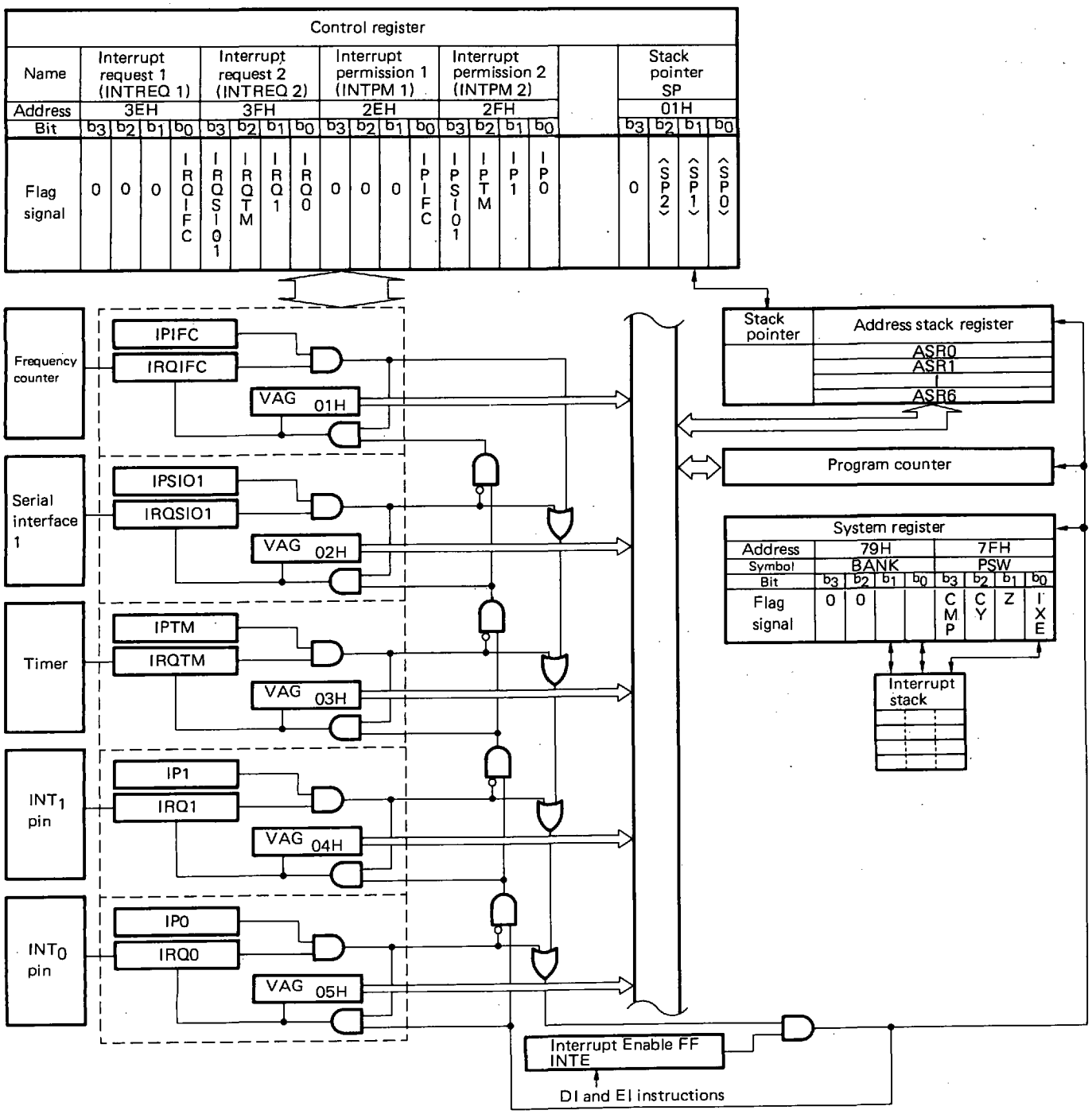
Fig. 12-1 shows the structure of an interrupt block.

As shown in Fig. 12-1, an interrupt block consists of the following: INT_0 pin, INT_1 pin, timer, serial interface 1, each "interrupt request control block" which controls an interrupt request output from each peripheral hardware device of the frequency counter, "Interrupt Enable flip/flop (INTE)" which sets all the interrupt enable flags, and "stack pointer", "address stack register", "program counter", and "interrupt counter" which are controlled when interrupt is accepted.

An "interrupt request processing block" of each peripheral hardware device consists of "flip/flop (IRQxxx) which detects each interrupt request", "flip/flop (IPxxx) which sets each Interrupt Enable", and "vector address generator (VAG)" which specifies a vector address when interrupt is accepted.

IRQxxx flip/flop and IPxxx flip/flop correspond to each flag of interrupt request 1, interrupt request 2, interrupt permission 1, and interrupt permission 2 on a one to one basis.

Fig. 12-1 Structure of an interrupt block



12.2 INTERRUPT FUNCTION

The interrupt function can be used by the INT_0 pin, INT_1 pin, timer, serial interface 1, and frequency counter.

The interrupt function is used for executing a specific program by interrupting a program which is currently being executed when these peripheral hardware devices satisfy a specified condition (for instance, a download signal is added to the INT_0 pin).

In this case, the interrupt signal from the peripheral hardware is called an "interrupt request" and the outputting of the interrupt signal is referred to as "issuing of an interrupt request". A specific interrupt processing program is called an "interrupt processing" routine.

When an interrupt is accepted, control is branched to a program memory address (vector address) predetermined for each interrupt factor. Consequently, each interrupt processing can be started from this vector address.

The interrupt function is classified into the processing up to acceptance of the interrupt and processing after acceptance of the interrupt. That is, the function is classified into the function up to acceptance of the interrupt request issued from each hardware device, and the function for branching control to the vector address after acceptance of the interrupt, and returning control to the program which was being executed before the interrupt was issued.

Sections **12.2.1** to **12.2.5** describe the function of each block which was shown in Fig. 12-1.

12.2.1 Peripheral Hardware

Peripheral hardware devices which have the interrupt function are the INT_0 pin, INT_1 pin, timer, serial interface 1 and frequency counter.

A condition of issuing an interrupt request can be set for each peripheral hardware.

For instance, for the INT_0 pin, the selection condition can be made as to whether a request is issued by the rising edge of the signal added to the INT_0 pin or by the falling edge of the signal added to the INT_0 pin.

See Sections **12.3** to **12.7** for details of each peripheral hardware interrupt request issuing condition.

12.2.2 Interrupt Request Processing Block

An interrupt request processing block is set in each peripheral hardware device and the block generates a vector address at the presence or absence of interrupt request, interrupt permission, or interrupt acceptance.

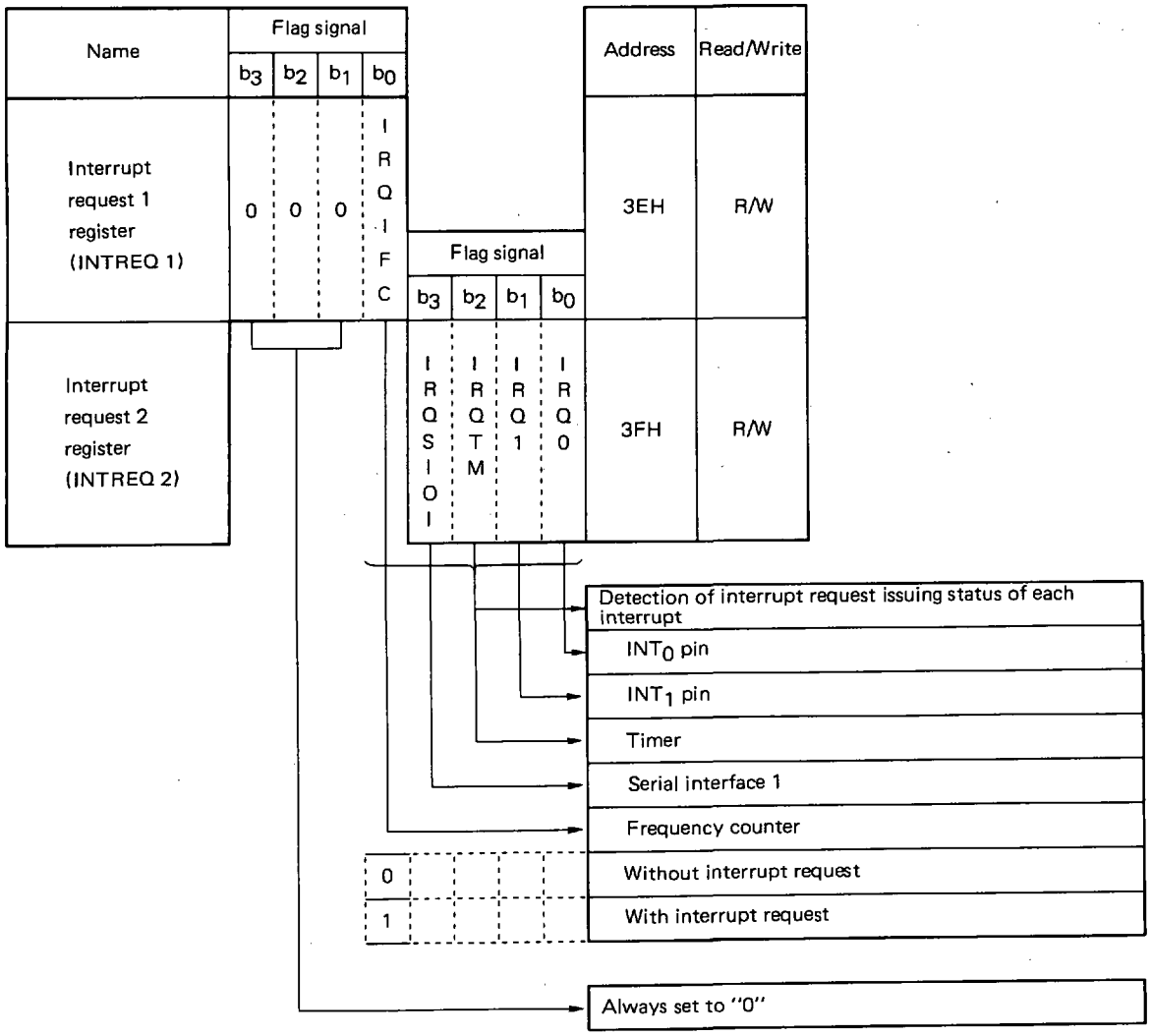
Sections **12.2.3** and **12.2.4** describe each flag of an interrupt request processing block.

12.2.3 Interrupt Request Flag (IRQXXX)

Since the IRQXXX flag corresponds to each flag of the interrupt request 1 register and interrupt request 2 register of the control register on a one to one basis, Read and Write operations are allowed via a window register.

Sections (1) and (2) show the structure and functions.

(1) Structure and functions of an interrupt request 1 register and interrupt request 2 register



At Resetting									
	Power On	0	0	0	0	0	0	0	0
	Clock Stop				0	0	0	0	0
	CE				0	0	0	0	0

(2) Interrupt request flag function

Each interrupt request flag (IRQxxx) is set (1) when an interrupt request is issued from each peripheral hardware device and is reset (0) when the interrupt is accepted.

By detecting these interrupt request flags (IRQxxx) when interrupt is not permitted, each interrupt request issuing status can be detected.

When "1" is written via window register, the same processing as for the case where an interface request is issued is performed.

Once this flag is set, the flag is not reset until a corresponding interrupt is accepted or "0" is written via a window register.

Even if a number of interrupt requests are issued simultaneously, an interrupt request flag is not reset for the interrupt which is not accepted.

12.2.4 Interrupt Permission Flag (IPxxx)

Since the IPxxx flag corresponds to each flag of the interrupt permission 1 register and interrupt permission 2 register of the control register, Read and Write operations can be performed via a window register.

The structure and functions are described in items (1) and (2).

(1) Structure and functions of the interrupt permission 1 register and interrupt permission 2 register

The structure and functions are described below.

Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
Interrupt permission 1 register (INTPM1)	0	0	0	IPIFC	2EH	R/W
Interrupt permission 2 register (INTPM2)	IPSIOI	IPTM	IP1	IP0	2FH	R/W

Flag signal

b_3	b_2	b_1	b_0	Sets permission of each interrupt
				INT_0 pin
				INT_1 pin
				Timer
				Serial interface 1
				Frequency counter
0				Prohibited
1				Permitted

Always set to "0".

At Resetting	b_3	b_2	b_1	b_0
Power	0	0	0	0
Clock Stop	0	0	0	0
CE	0	0	0	0

(2) Function of an interrupt permission flag

Each interrupt flag sets the interrupt permission of each peripheral hardware device.

To accept an interrupt, the interrupt must be permitted by each permission flag, a corresponding interrupt request must be issued, and the "EI" statement (permission for all the interrupts) must be executed.

12.2.5 Vector Address Generator (VAG)

When interrupt of each peripheral hardware device is accepted, the branch address (vector address) of the program memory for the interrupt factor which was accepted is generated.

Table 12-1 lists the vector address for each interrupt factor.

Table 12-1 Vector addresses for interrupt factors

Interrupt factor	Vector address
INT_0 pin	05H
INT_1 pin	04H
Timer	03H
Serial interface 1	02H
Frequency counter	01H

12.2.6 Interrupt Enable Flip/Flop (INTE)

Interrupt Enable flip/flop sets all the five types of interrupts.

When this flip/flop is set (1) and when "1" is output from each interrupt request processing block, "1" is output from this flip/flop and the interrupt is accepted.

Interrupt is not accepted even if "1" is output from each interrupt request processing block when this flip/flop is reset (0).

The specific "EI" instruction (set) and "DI" instruction (reset) are used for setting and resetting flip/flop.

When the "EI" instruction is executed, this flip/flop is set at completion of the instruction which was executed following the "EI" instruction, and when the "DI" instruction is executed, the flip/flop is reset during the "DI" instruction execution cycle.

When the interrupt is accepted in the state (EI state) where the Interrupt Enable flip/flop is set, this flip/flop is reset (DI state) when the interrupt is accepted.

Even if the "DI" instruction is executed "in the DI state" or even if the "EI" instruction is executed "in the EI state", no influence is imposed.

At Power On Reset, Clock Stop, or CE Reset, this flag is reset (DI state).

12.2.7 Stack Pointer, Address Stack Register, and Program Counter

An address stack register saves an address returned from the interrupt processing routine.

An address pointer specifies the address stack register to be used among the seven registers (ASR0 to ASR6).

That is, if interrupt is accepted, the value of the stack pointer is decremented by 1 and the value of the program counter is saved in the address stack register specified by the stack pointer. When the "RETI" instruction, which is a specific return instruction, is executed after execution of the interrupt processing routine, the content of the address stack register specified by the stack pointer is returned to the program counter and the stack pointer value is incremented by 1.

See **4, "Stack"** also.

1.2.2.8 Interrupt Stack

An interrupt stack saves the contents of the bank register and index enable flag at acceptance of interrupt.

When interrupt is accepted and the bank register and Index Enable flag are saved, the bank register and Index Enable flag in the system register are reset (0).

An interrupt stack can save up to four levels of the contents of bank register and Index Enable flags.

Consequently, multiple interrupts of up to four levels can be performed such as in the case in which another interrupt is accepted in an interrupt processing routine.

The content of the interrupt stack is returned to the bank register and Index Enable flag of the system register by executing the "RETI" instruction which is a specific return instruction used for an interrupt processing routine.

See also **4, "Stack"**.

12.3 INTERRUPT ACCEPTANCE OPERATION

12.3.1 Interrupt Accept Operation and Priority

Operation up to acceptance of interrupt is described below.

(1) If the interrupt condition is satisfied (for instance, the falling signal is input in the INT_0 pin), each peripheral hardware device outputs an interrupt request signal to each interrupt request blcok.

(2) When an interrupt request signal is accepted from each peripheral hardware device, the corresponding IRQxxx flag (for instance IRQ0 flag if the device is INT_0 pin) is set (1) to each interrupt request blcok.

(3) If an interrupt permission flag (IPxxx) corresponding to each IRQxxx flag, for instance IP0 flag if the flag is IRQ0 flag, is set (1) when each interrupt request flag (IRQxxx) is set, "1" is output from each interrupt request block.

(4) The signal output from each interrupt request block is input to an Interrupt Enable flip/flop via an OR circuit. This Interrupt Enable flip/flop is set (1) by the "EI" instruction and reset by the "DI" instruction.
If "1" is output from each interrupt request block when an Interrupt Enable flip/flop is set, "1" is output from the Interrupt Enable flip/flop and the interrupt is accepted.

When interrupt is accepted, output of the Interrupt Enable flip/flop is input to each interrupt request block via the AND circuit as shown in Fig. 12-1.

An interrupt request flag (IRQxxx) is set by the signal input to each interrupt request block and the vector address for each interrupt is output.

In this case, since the interrupt acceptance signal is not conveyed to the next stage if "1" is output from the interrupt request block, interrupts are accepted in the following priority sequence when a number of interrupt requests are issued simultaneously.

INT_0 pin > INT_1 pin > timer > serial interface 1 > frequency counter

This priority sequence is called "hardware priority sequence".

Fig. 12-2 shows the flow chart of the interrupt acceptance operation.

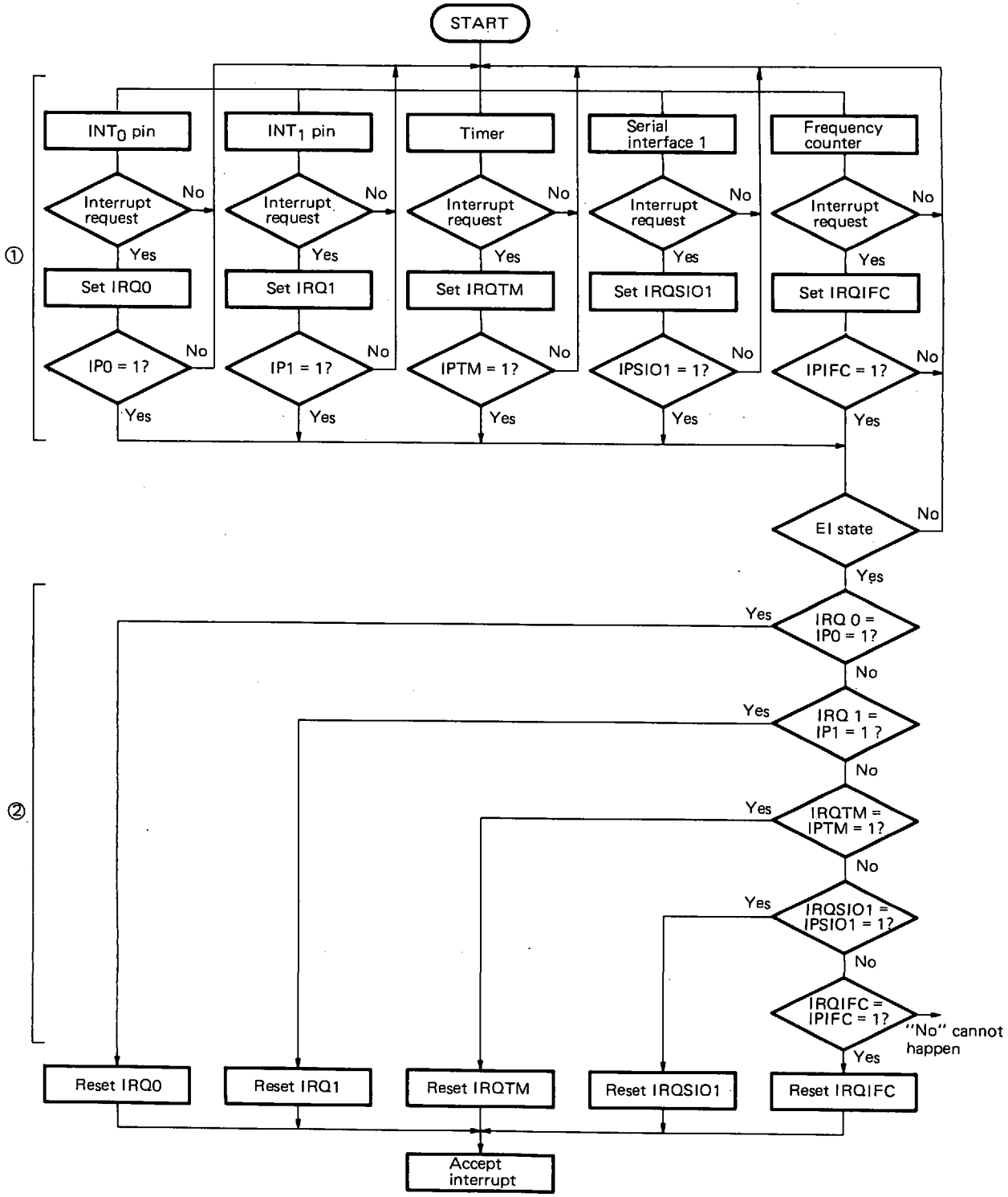
Since processing ① shown in Fig. 12-2 is always executed concurrently, each interrupt request flag (IRQxxx) is set concurrently when a number of interrupt requests are issued concurrently.

However, processing ② is executed according to the priority sequence by each interrupt permission flag.

That is, interrupt for the interrupt factor is not accepted unless the interrupt permission flag (IPxxx) is set. Since an interrupt permission flag (IPxxx) can be set or reset, interrupt of high hardware priority can be prohibited by resetting the interrupt permission flag.

Interrupt by an interrupt permission flag is called "maskable interrupt". Since the interrupt of high hardware priority can be prohibited by a program, maskable interrupt is also called "software priority sequence".

Fig. 12-2 Flow chart of interrupt acceptance operation



12.3.2 Timing Chart when Interrupt is Accepted

Fig. 12-3 shows a timing chart when interrupt is accepted.

The timing chart shown in (1) of Fig. 12-3 is a timing chart by interrupt of one type.

(a) of (1) shows the timing chart when an interrupt request flag (IRQxxx) is set (1) at the end, and (b) of (1) shows the timing chart when an interrupt permission flag (IPxxx) is set (1) at the end.

In either case, interrupt is accepted when all of the interrupt request flag (IRQxxx), interrupt enable flip flop, and interrupt permission flag (IPxxx) are set.

When the flag or flip/flop which was set last satisfies the first instruction cycle of "MOVT DBF, @AR" instruction or skipping condition, interrupt is accepted after execution of the second instruction cycle of the "MOVT DBF, @AR" instruction or the instruction which was skipped (NOP) is executed.

Interrupt Enable flip/flop is set by the instruction cycle following execution of the "EI" instruction.

(2) in Fig. 12-3 shows the timing chart when a number of interrupts are used.

When a number of interrupts are used and all the interrupt permission flags (IPxxx) are set, the interrupt which is given priority by the hardware is accepted first. However, the hardware priority can be changed by manipulating the interrupt permission flag using a program.

The "interrupt cycle" shown in Fig. 12-3 is a special cycle for resetting the interrupt request flag, specifying a vector address, and saving a program counter after the interrupt is accepted and requires execution time of one instruction (4.44 μs). See Section **12.4, "Operation After Interrupt Is Accepted"** for details.

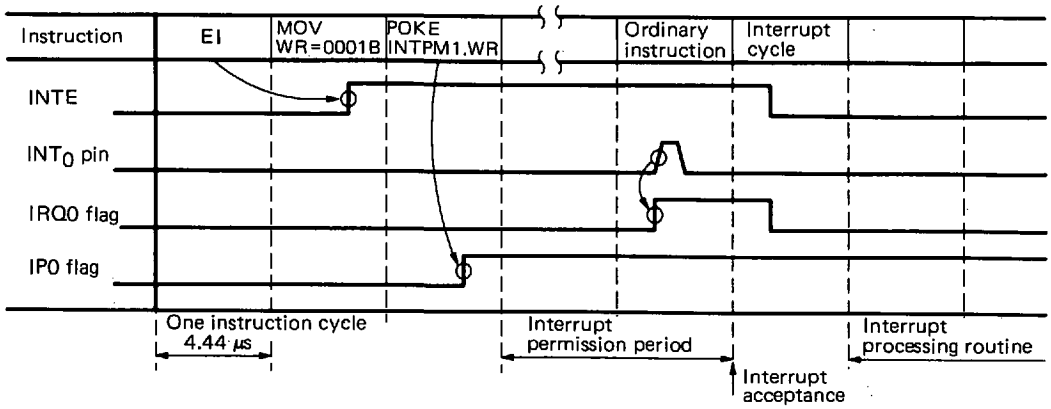
Since an interrupt request flag is set (1) by the peripheral hardware interrupt request regardless of the EI instruction, and interrupt permission flag, the presence or absence of an interrupt request can be detected by detecting the interrupt request flag (IRQxxx) by a program.

Fig. 12-3 Interrupt acceptance timing chart (1/2)

(1) When one type of interrupt
(example: rising of the INT_0 pin) is used

(a) When there is no interrupt mask time by an interrupt permission flag (IPxxx)

① Ordinary instruction which is not the "MOVT" instruction or which does not satisfy the skip condition when interrupt is accepted



② "MOVT" instruction or "instruction which satisfies the skip condition" when interrupt is accepted

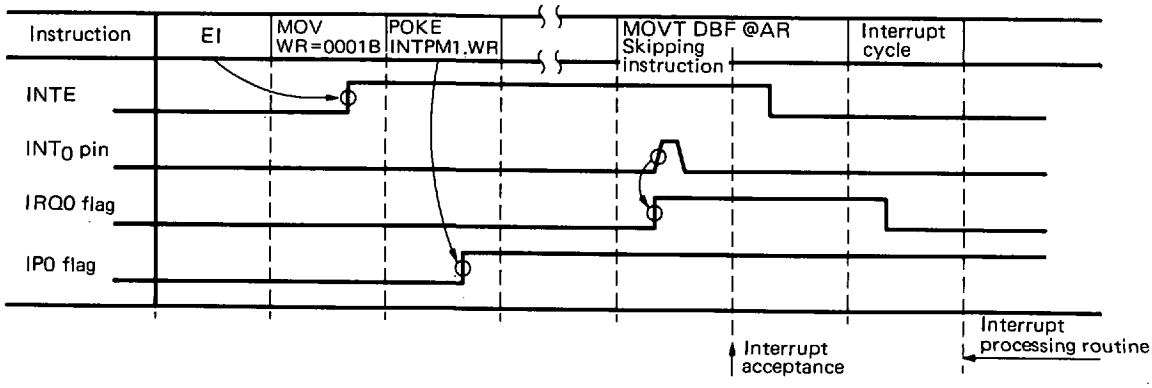


Fig. 12-3 Interrupt acceptance timing chart (2/2)

(b) When there is an interrupt hold period by an interrupt permission flag

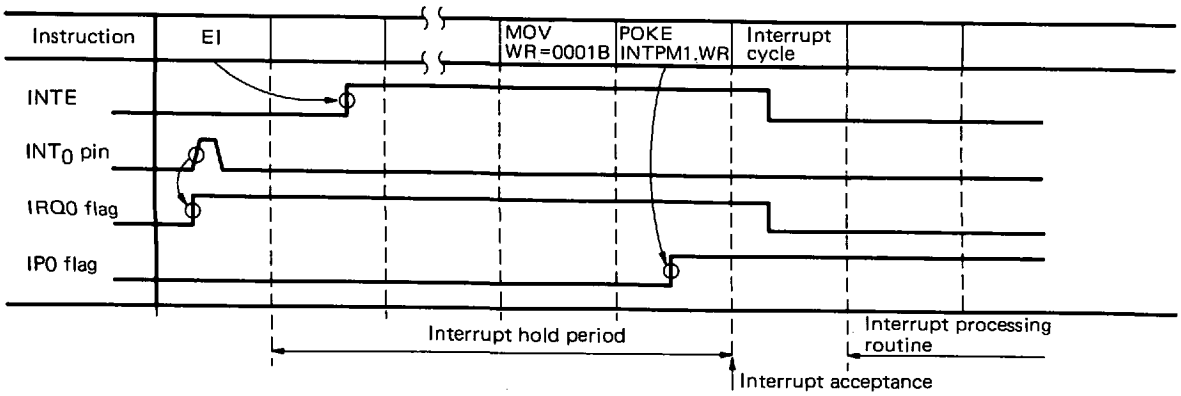
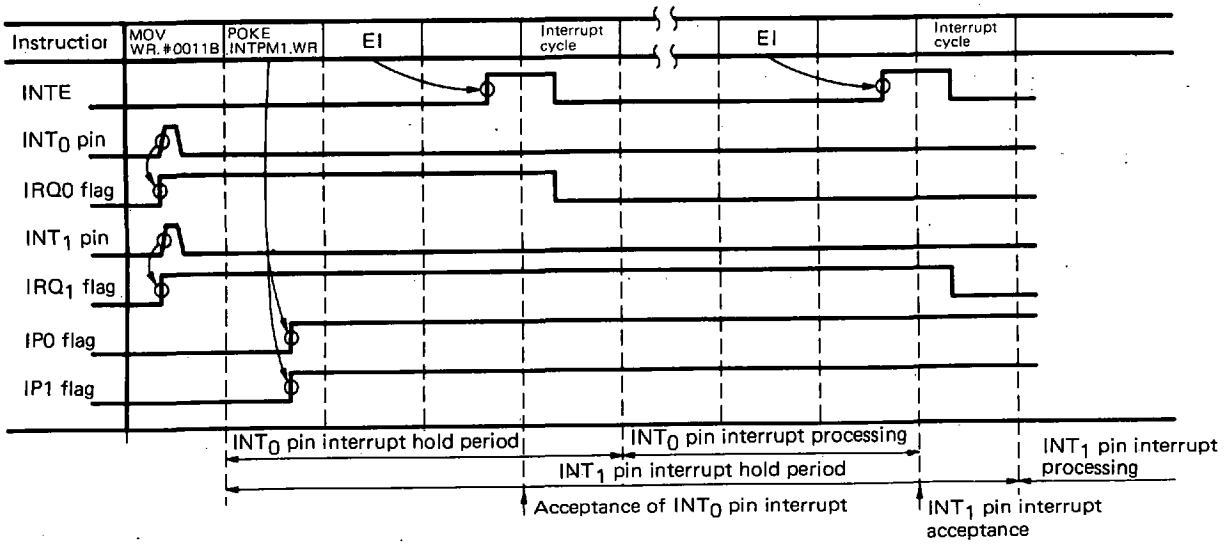


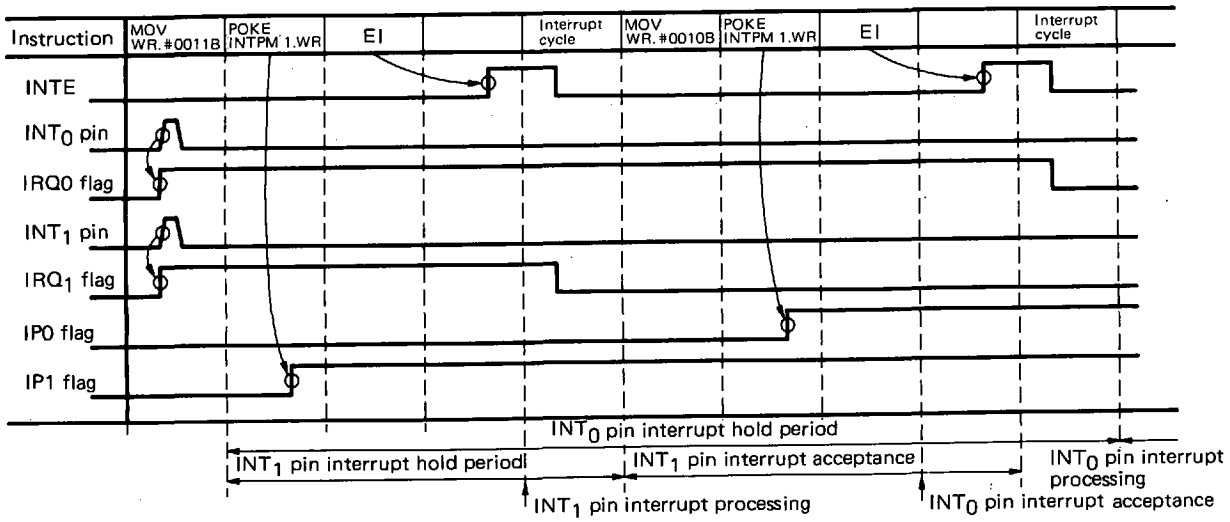
Fig. 12-3 Interrupt acceptance timing chart (2/2)

(2) When a number of interrupts are used
(example: two types; INT_0 pin and INT_1 pin)

(a) Hardware priority



(b) Software priority



12.4 OPERATION AFTER ACCEPTANCE OF INTERRUPT

When interrupt is accepted, the following processing units are executed automatically and sequentially.

(1) Set interrupt enable flip/flop and the interrupt request flag (IRQxxx) corresponding to the interrupt request which was accepted.
That is an interrupt prohibited state is set.

(2) Decrement the content of the stack pointer by 1.

(3) Save the content of the program counter to the address stack register specified by the stack pointer. In this case, the content of the program counter is the next program memory address when the interrupt is accepted.

For instance, if the instruction is a branch instruction, the address of the branch destination is used and if the instruction is a subroutine call instruction, the address which was called is used. When the skip condition is satisfied by the skip instruction, interrupt is accepted after the next instruction is executed as the "NOP" instruction. Consequently, the content of the program counter is the address which was skipped.

(4) Back up the low-order 2 bits of the bank register (BANK: address 79H) and Index Enable flag (IXE: bit b_0 of address 7FH) in the interrupt stack.

(5) Transfer the content of the vector address generator corresponding to the interrupt which was accepted to the program counter.
That is, control is branched to the interrupt processing routine.

Processing units described in (1) to (5) are executed during one special instruction cycle (4.44 μs) which does not involve execution of ordinary instructions. This instruction cycle is called an "interrupt cycle".

That is, the time of one instruction cycle is required from accept of interrupt to branching of control to the corresponding vector address.

12.5 RETURN PROCESSING FROM AN INTERRUPT PROCESSING ROUTINE

Use a specific instruction, "RETI", to return control from an interrupt processing routine to the processing which is to be performed when the interrupt is accepted.

When the "RETI" instruction is executed, the following processing units are executed automatically and sequentially.

(1) Return the content of the address stack register specified by the stack pointer to the program counter.

(2) Return the content of the interrupt stack to the low-order 2 bits and Index Enable flag (IXE: bit 0 of address 7FH) of the bank register (BANK: address 79H).

(3) Increment the content of the stack pointer by 1.

Processing units from (1) to (3) are processed within one instruction cycle where the "RETI" instruction is executed.

The "RETI" instruction and subroutine return instructions, "RET" and "RETSK" are different only in the return operations of the bank register described in (2) and index enable flag.

12.6 INTERRUPT PROCESSING ROUTINE

Interrupt is accepted, regardless of the program which is being executed at that time, when the interrupt request is issued, as long as interrupt is permitted in the program area.

Consequently, to return control to the program after execution of interrupt processing, the status must be reset so that no execution of interrupt processing appears to have happened.

For instance, when an arithmetic operation is executed during interrupt processing, the content of the Carry flag (CY) may be changed from the value before the interrupt was accepted, so that the program makes an incorrect decision after the return.

Consequently, backup and return operations at least are required in the interrupt processing routine for system registers and control registers which may be manipulated within the interrupt processing routine.

See Section **12.9, "Multiple Interrupt"** for the processing performed when another interrupt is permitted (multiple interrupt) during interrupt processing.

12.6.1 Backup Processing

This section describes an example of backup processing in interrupt routine.

Since only a bank register (BANK) and an Index Enable flag are backed up automatically by the hardware among system registers, other system registers must be backed up by the program, if required.

As shown in the program example, the "POKE" instruction and "PEEK" instruction are useful for backing up of system registers and performing return processing.

A transfer instruction (LD, r, m or ST m, r) can also be used instead of the "PEEK" and "POKE" instruction. However, if a transfer instruction is used for the backup method when the row addresses of the general registers are not fixed at acceptance of interrupt, the data memory address cannot be specified easily.

If a transfer instruction is used for backing up the general register itself, the address which is backed up is not fixed because the general register address is not fixed. Consequently, the general register must be fixed at least during execution of the interrupt permission routine.

However, since the addresses of register files which are controlled by the "PEEK" instruction or "POKE" instruction are specified regardless of the contents of the general registers and the addresses of the register files, 40H to 7FH, overlap the data memory of the bank which is currently selected, each system register can be backed up by specifying the bank only.

In the example, the general register is specified again in row address 07H of BANK1 after the window register and register pointer (RPH, RPL) are backed up the "PEEK" and "POKE" instructions, and then another system register is backed up by the "ST" instruction.

Fig. 12-4 shows a backup operation example by the "PEEK" and "POKE" instructions.

12.6.2 Return Processing

This section describes an example of return processing.

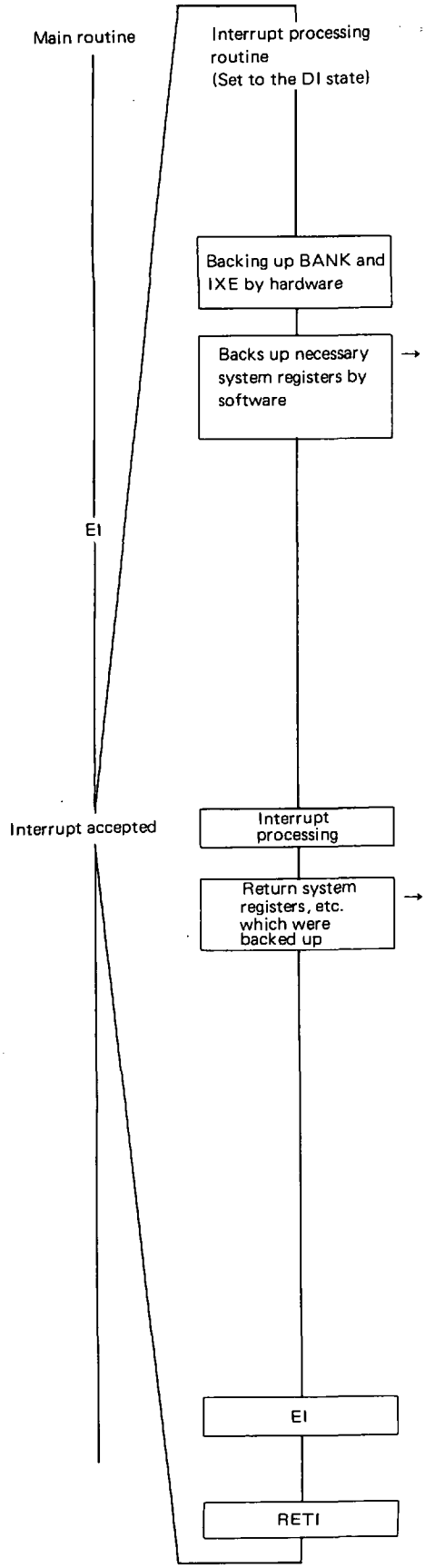
Return processing is the reverse operation of the backup processing described in Section **12.6.1.**

Since the state is an "interrupt permitted state" (EI state) when interrupt is accepted, the "EI" instruction must be executed before executing the "RETI" instruction.

The "EI" instruction sets (1) Interrupt Enable flip/flop after execution of the "RETI" instruction. Consequently, the state is set to an "interrupt permitted state" after control is returned to the program which was being executed before the interrupt was accepted.

Example:

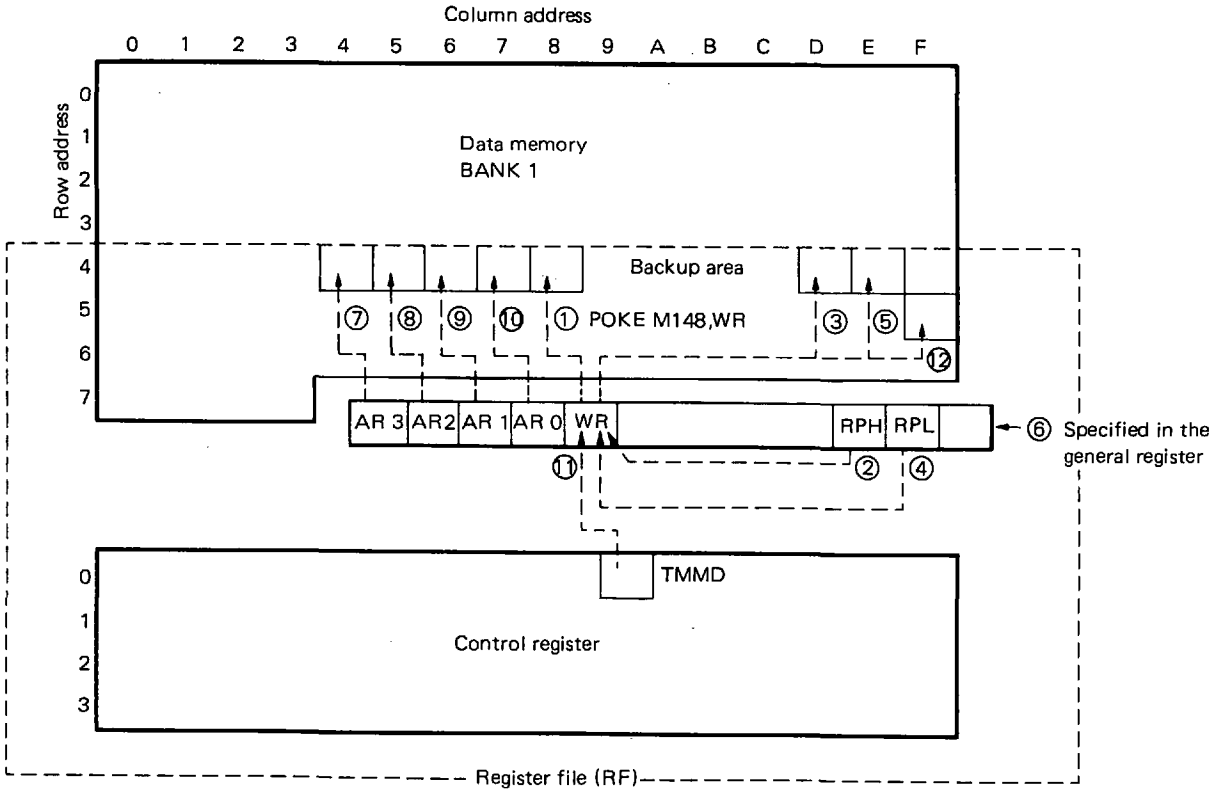
Method of backing up states in an interrupt processing routine



Program example

```
M144    MEM   1.44H
M145    MEM   1.45H
M146    MEM   1.46H
M147    MEM   1.47H
M148    MEM   1.48H
M14D    MEM   1.4DH
M14E    MEM   1.4EH
M15F    MEM   1.5FH
TMMD    MEM   0.89H

    BANK1                ; Sets the bank to 1.
(1) POKE  M148, WR       ; Backs up the content of the window register to M148.
(2) PEEK  WR,RPH         ;
(3) POKE  M14D,WR        ; Backs up the contents of the general
                         ; register pointer to M14D and M14E.
(4) PEEK  WR,RPL         ;
(5) POKE  M14E,WR        ;
(6) MOV   RPH, #1        ; Sets the general register to row address
                         ; 7 of bank 1.
    MOV   RPL, #0EH      ;
(7) ST    M144,AR3       ; Backs up necessary system registers.
(8) ST    M145,AR2       ;
(9) ST    M146,AR1       ;
(10) ST   M147,AR0       ;
(11) PEEK WR,TMMD        ; Backs up necessary control registers.
(12) ST   M15F,WR        ;

BANK1                    ; Sets bank to 1.
MOV   RPH,1              ; Sets the general register to row address
                         ; 7 of bank 1.
MOV   RPL,0EH            ;
LD    AR3,M144           ; Returns the system registers which were backed up.
LD    AR2,M145           ;
LD    AR1,M146           ;
LD    AR0,M147           ;

LD    WR,M15F            ; Returns the control registers which were backed up.
POKE  TMMD,WR            ;
PEEK  WR,M14D            ; Returns general register pointers.
POKE  RPH,WR             ;
PEEK  WR,M14E            ;
POKE  RPL,WR             ;
PEEK  WR,M148            ; Returns window registers which were backed up.

                         ; Interrupt is permitted by the EI instruction
                         ; (setting INTE) when the next "RETI"
                         ; instruction is executed.

                         ; Returns BANK and IXE by hardware.
```

Fig. 12-4 Backup operation of system registers and control registers performed using a window register



12.6.3 Notes on Interrupt Processing Routine

The following points must be noted for an interrupt processing routine.

(1) Data which is backed up by hardware

Bank registers and Index Enable flags are reset to "0" after being backed up in an interrupt stack.

(2) Data which was saved by software

Data which was backed up by software is not reset after backup.

In particular, program status words including a BCD flag (BCD), compare flag (CMP), carry flag (CY), zero flag (Z), and memory pointer enable flag (MPE) must be initialized as required, because the values which were set before the interrupt was accepted, are retained.

12.7 EXTERNAL (INT_0 PIN AND INT_1 PIN) INTERRUPT

Two types of external interrupt pins are available, the INT_0 pin (pin number 12) and INT_1 pin (pin number 13). An interrupt request is issued according to the rising edge or falling edge of the signal sent to these pins.

12.7.1 Structure of External Interrupt

Fig. 12-5 shows the structures of the INT_0 pin and INT_1 pin.

As shown in Fig. 12-5, the signals input from the INT_0 and INT_1 pin are input to each edge detection circuit after being input to the INT_0 latch and INT_1 latch, respectively.

The edge detection circuit outputs an interrupt request signal according to the signal input from each pin and input of flip/flop of IEG0 and IEG1.

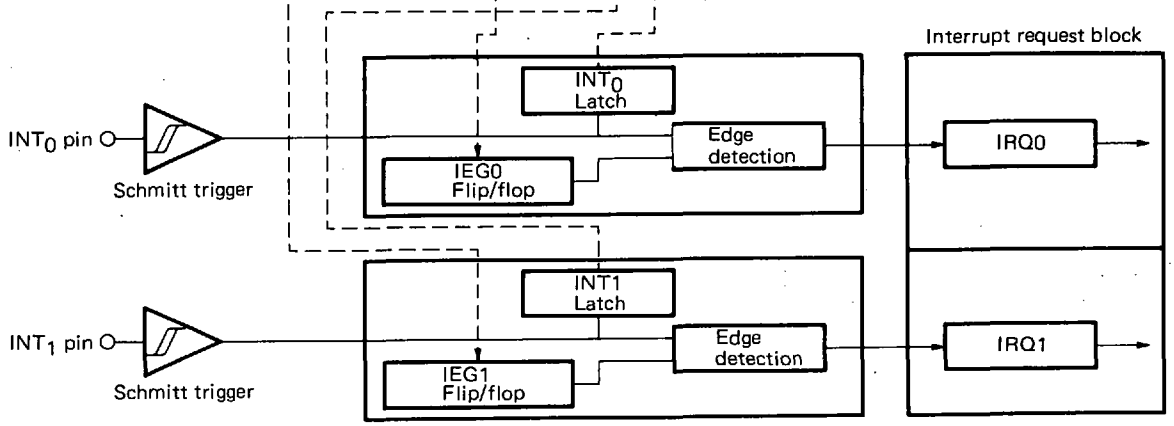
Flip/flop of IEG0 and flip/flop of IEG1 correspond to flag IEG0 and flag IEG1 of the low-order 2 bits of the interrupt edge selection register (INTEDGE: address 1FH) of the control register on a one to one basis.

The INT_0 and INT_1 latches correspond to the flags INT_0 and INT_1 of the low-order 2 bits of the interrupt pin level judgment register (INTJDG: address 0FH) of the control register on a one to one basis.

Since pins INT_0 and INT_1 operate incorrectly due to noise, Schmitt trigger input is applied and consequently, pulse input less than 1 μs is not accepted.

Fig. 12-5 Structures of the INT_0 pin and INT_1 pin

Control register								
Name	Interrupt edge selection (INTEDGE)				Interrupt pin level judgment (INT JDG)			
Address	1 FH				0 FH			
Bit	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
Flag signal	0	0	IEG1	IEG0	0	0	INT1	INT0



12.7.2 External Interrupt Function

The INT_0 pin and INT_1 pin issue an interrupt request according to the rising or falling edge added to each pin.

The IEG0 and IEG1 flag which are the low-order 2 bits of the interrupt edge selection register (INTEDGE: address 1FH) of the control register are used for selecting a rising edge or falling edge.

For the signals input to the INT_0 pin and INT_1 pin, the signal interrupt levels which were input by the INT_0 flag and INT_1 flag of the interrupt pin level judgment register (address 0FH) can be detected regardless of the interrupt as shown in Fig. 12-5.

12.7.3 Structure and Function of Interrupt Edge Selection Register (INTEDGE)

An interrupt edge selection register sets an input signal edge (rising or falling edge) which issues interrupt requests of INT_0 pin and INT_1 pin, which are external interrupts.

The structure and functions are described below.

Table 12-2 shows the relationship between the IEG0 and IEG1 flags and interrupt request issuing edges.

Name	Flag signal b_3	b_2	b_1	b_0	Address	Read/Write
Interrupt edge selection register (INTEDGE)	0	0	IEG1	IEG0	1FH	R/W

- IEG1 (b_1), IEG0 (b_0): Sets input signal edge for issuing an interrupt request of each pin
  - IEG0 (b_0): INT_0 pin (pin number 12)
  - IEG1 (b_1): INT_1 pin (pin number 11)
    - 0: Rising edge
    - 1: Falling edge
- b_3, b_2: Always set to "0"

At Reset	b_3	b_2	b_1	b_0
Power On	0	0	0	0
Clock Stop			0	0
CE			0	0

Table 12-2 IEG0 and IEG1 flags and interrupt request issuing edge

Value of each flag		Interrupt request issuing edge of each pin	
IEG0	IEG1	INT_0 pin	INT_1 pin
1	1	Falling	Falling
1	0	Falling	Rising
0	1	Rising	Falling
0	0	Rising	Rising

When the interrupt request issuing edge is switched by the IEG0 flag and IEG1 flag, the interrupt request signal may be issued as soon as the edge is switched.

For instance, assume that the IEG0 flag is set to "1" (falling edge) and a High level is input from the INT_0 pin as shown in Table 12-3. If the IEG0 flag is reset at this time, the edge detection circuit determines that an edge is input and issues an interrupt request. Because of this, caution is necessary.

See Section **12.2, "Interrupt functions"** for the operation performed after an interrupt is issued.

Table 12-3 Issuing of interrupt requests due to the change to the IEG flag

Change to the IEG0 and IEG1 flags	State of the INT_0 pin and INT_1 pin	Issuing of an interrupt request	Status of the IRQ flag
1 → 0 (Falling) (Rising)	Low level	Not issued	Status retained
	High level	Issued	Set
0 → 1 (Falling) (Rising)	Low level	Issued	Set
	High level	Not issued	Status retained

12.7.4 Structure and Functions of an Interrupt Pin Level Judgment Register

By reading interrupt pin level judgment register INT_0 and INT_1 flag, the signal level input to the INT_0 pin and INT_1 pin can be detected.

Since the INT_0 flag and INT_1 flag are set and reset regardless of the interrupt, they can be used as a 2-bit general purpose input port when an interrupt function is not used.

If interrupt is not permitted, the pins can be used as general purpose ports for detecting a rising or falling edge by reading the interrupt request flags (IRQ0, IRQ1).

However, since the interrupt request flags (IRQ0 and IRQ1) are not reset automatically, they must be reset by the program.

The structure and function are shown below.

Name	Flag signal b_3	b_2	b_1	b_0	Address	Read/Write
Interrupt pin level judgment register (INTJDG)	0	0	INT1	INT0	0FH	R

- Detects the level input to each pin
  - INT_0 pin
  - INT_1 pin
    - 0: Low level is input
    - 1: High level is input
- Always fixed to "0"

At Reset	b_3	b_2	b_1	b_0
Power On	0	0	–	–
Clock Stop	↓	↓	–	–
CE	↓	↓	–	–

12.8 INTERNAL (TIMER, SERIAL INTERFACE 1, FREQUENCY COUNTER) INTERRUPT

Three types of internal interrupt are available; timer, serial interface 1, and frequency counter.

12.8.1 Timer Interrupt

Timer interrupt can issue an interrupt request at regular intervals.

Four times can be selected, 250 ms, 100 ms, 5 ms, and 1 ms.

See **13, "Timer Function"** for details.

12.8.2 Interrupt of Serial Interface 1

Interrupt of serial interface 1 can issue an interrupt request at Serial Out or Serial In operation termination.

A serial clock is used for issuing the interrupt request.

See **21, "Serial Interface"** for details.

12.8.3 Frequency Counter

Frequency counter interrupt can issue an interrupt request at termination of counting operation.

See **22, "Frequency Counter (FC)"** for details.

12.9 MULTIPLE INTERRUPT

Multiple interrupt is the method of interrupt in which interrupt processing of other interrupt factors C and D are performed during interrupt processing of interrupt factors A and B as shown in Fig. 12-6.

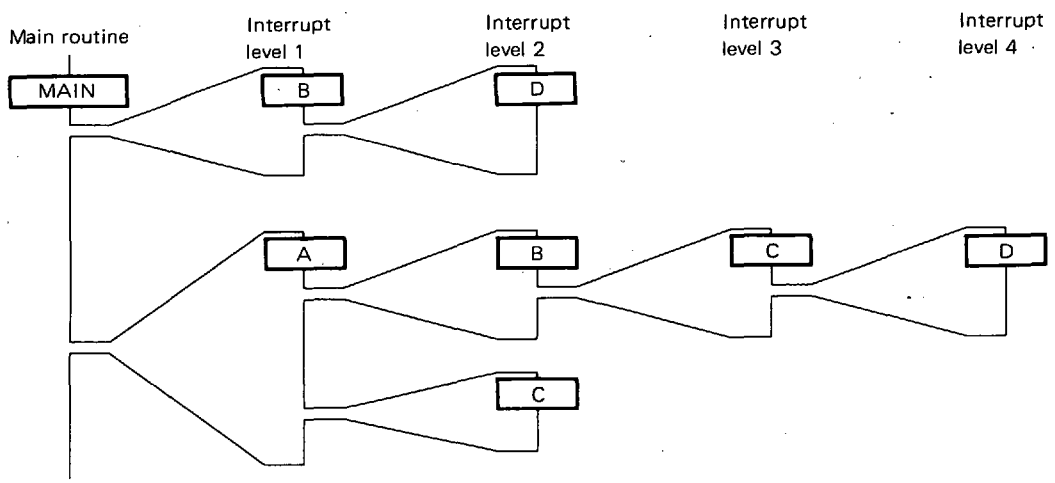
The multiplicity of the interrupt is called an interrupt level.

When multiple interrupt is used, the following points must be noted.

(1) Priority of interrupt factors
(2) Limit of the interrupt level by the interrupt stack
(3) Limit of the interrupt level by the address stack register (ASR)
(4) Backing up of system registers and control registers

Sections **12.9.1** to **12.9.4** describe the items (1) to (4) described above.

Fig. 12-6 Example of multiple interrupt



12.9.1 Priority of Interrupt Factors

When multiple interrupt is used, the priority of interrupt factors must be determined.

For instance, the priority of interrupt factors, A, B, C, and D can be A = B = C = D or A < B < C < D.

However, when the priority is A = B = C = D, interrupts of A, B, C, and D are always accepted in the main routine.

Since interrupts of A, B, and D are prohibited in this case, if interrupt of C is accepted, then the meaning of multiple interrupt is lost.

When the priority is A < B < C < D, interrupt of C must be processed first even if interrupt of A or B is being processed, and interrupt of D must be processed first even if interrupt of C is being processed.

The priority set by the hardware which is described by Section **12.3, "Interrupt Acceptance Operation"** or priority set by software using the interrupt permission flag (IPxxx) can be used for the priority described above.

The priority must be determined in multiple interrupt for the following reason. Considering interrupt factors A and B, factor A issues a request every 10 ms and the interrupt processing requires 4 ms and factor B issues a request every 2 ms and the processing requires 1 ms.

Assume that no priority is set for A and B. If interrupt processing of A is executed according to the interrupt request A during interrupt processing of B, interrupt processing of B is not performed for a number of times.

In general, since interrupt is used for urgent processing, in the case described above, a program is required to prohibit interrupt processing of A during interrupt processing of B by setting priority of A < B, and accepting interrupt of B during processing of interrupt A.

When multiple interrupt is used without any urgency, priority is not always required. However, when the number of interrupt factors exceeds the limit of the multiple interrupt levels as described in Sections **12.9.2** and **12.9.3**, the priority should be set so as not to exceed the limit.

12.9.2 Limit of the Interrupt Levels by an Interrupt Stack

Contents of a bank register of the system register and Index Enable flag are automatically backed up in the interrupt stack.

Fig. 12-7 (a) shows operation of an interrupt stack.

As soon as being backed up in an interrupt stack, the bank register and Index Enable flag are reset.

Since four level of interrupt stacks area available, the bank register and Index Enable flag cannot be returned normally if the number of multiple interrupt levels exceeds 4 as shown in (b) of Fig. 12-7.

That is, multiple interrupt exceeding 4 levels cannot be used.

However, when the bank register and Index Enable flag are fixed in the main routine for which interrupt is permitted, and when the priority is clearly defined, multiple interrupt of more than four levels is enabled by using a subroutine return instruction, "RET" instruction.

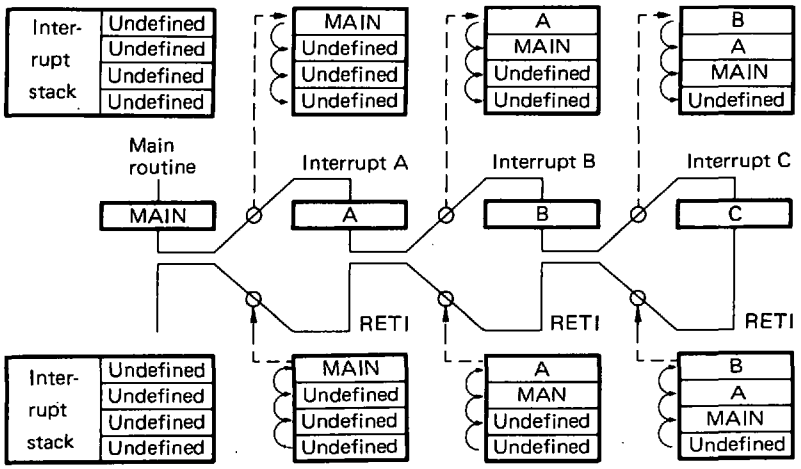
In multiple interrupt exceeding 4 levels, caution is necessary because the device and Emulator operate differently as shown in Figs. 12-8 and 12-9.

That is, the device operation of the interrupt stack is of a "one-off type" and the Emulator (IE-17K) operation is of "rotation type".

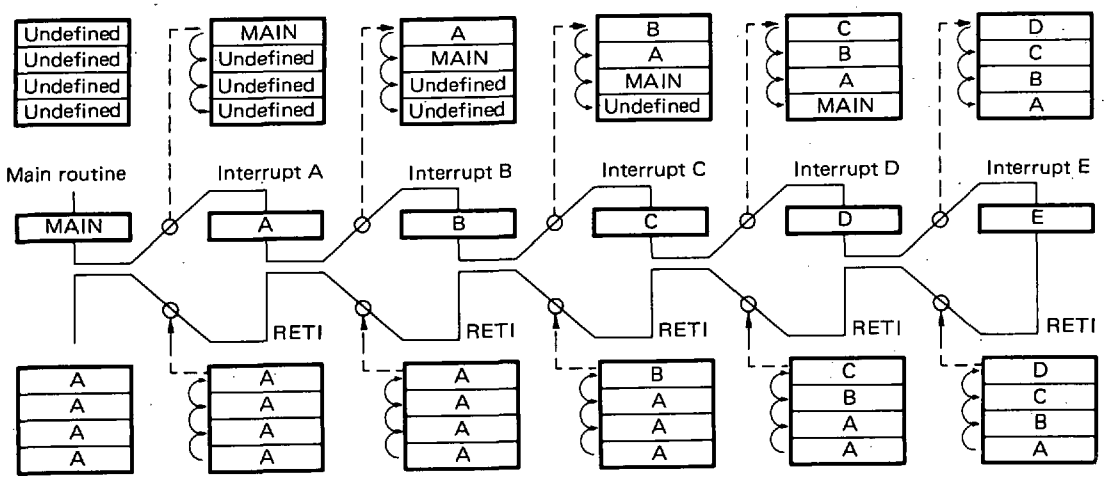
Consequently, the "RET" instruction must be used for the last return instruction when multiple interrupt exceeding 4 levels is used. The "RETI" instruction and "RET" instruction perform identical processing other than stack for turn processing.

Fig. 12-7 Interrupt stack operation at multiple interrupt

(a) Level 3 multiple interrupt

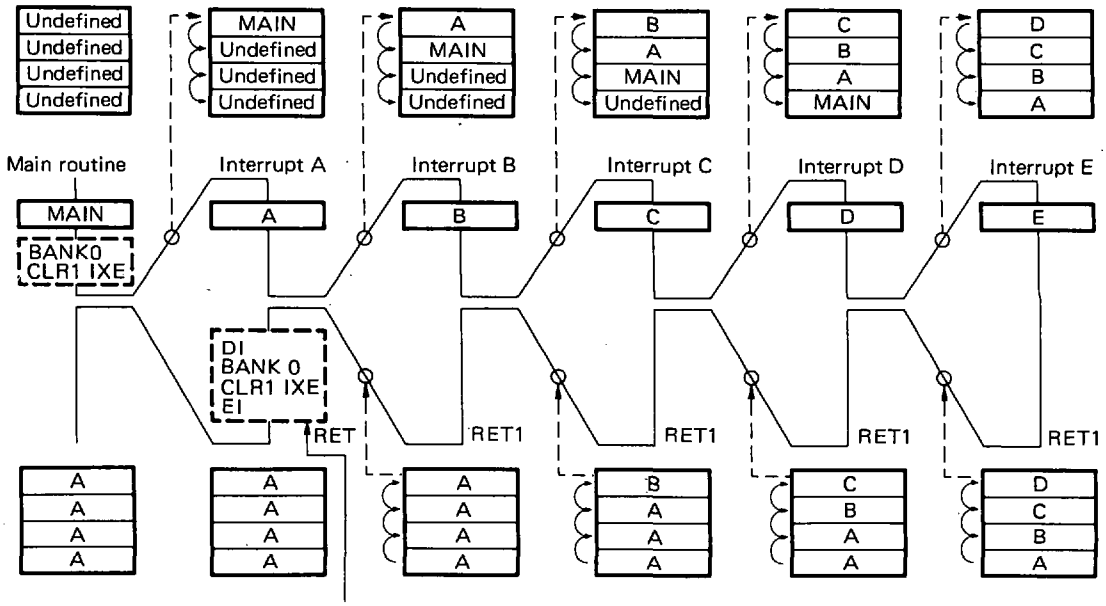


(b) Level 5 multiple interrupt



The BANK and IXE of interrupt A are returned if control is returned to the main routine at this point, and the operation of the main routine is not performed normally.

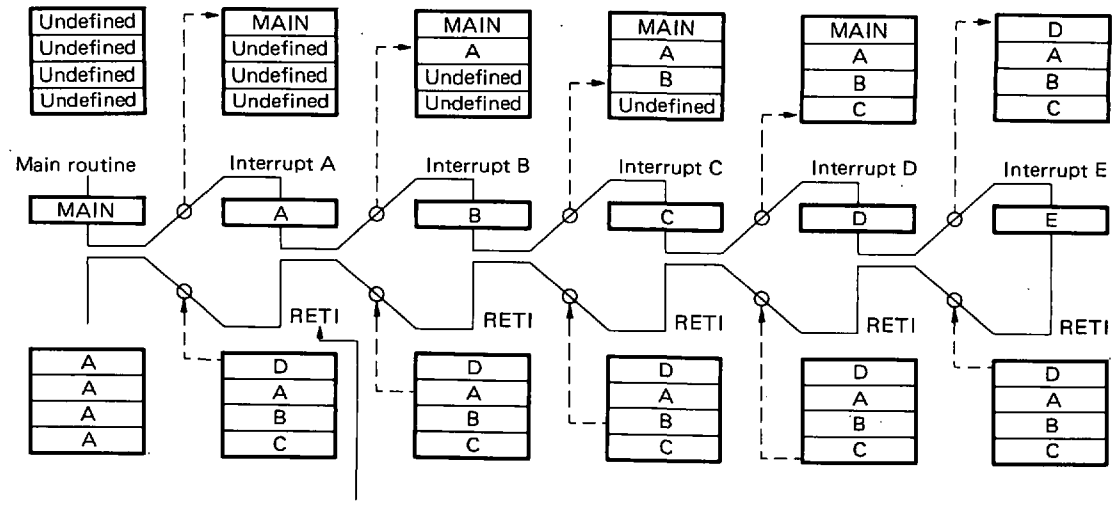
Fig. 12-8 Example of using multiple interrupt of level 3 or more



If the priority of A is set lower than B, C, and D and if the bank register and Index Enable flag of the main routine which allows interrupt A are fixed (in this example BANK0, IXE=0), multiple interrupt of 5 levels is enabled by using the "RET" instruction after specifying the bank register and Index Enable flag of the main routine at termination of interrupt processing of A.

When the bank register and Index Enable flag of interrupt A are the same as those of the main routine, the "RETI" instruction can be used. However, debugging cannot be performed by the "RETI" instruction because the operation in the Emulator of 17K series is different as shown in Fig. 12-9.

Fig. 12-9 Interrupt stack operation when 17K series Emulator (IE-17K) is used



If the "RETI" instruction is used by the Emulator (IE-17K), the contents of the bank register (BANK) and Index Enable flag (IXE) of interrupt D are returned.

12.9.3 Limit of Interrupt Levels by an Address Stack Register

An address returned from interrupt processing is saved in the address stack register automatically. Seven levels of address stack registers, ASR0 to ASR6, can be used as described in **5, "Stack"**. Since five interrupt factors, INT_0 pin, INT_1 pin, timer, serial interface, and frequency counter, are available, there is no limit for multiple interrupt levels when an address stack register is used.

However, since an address stack register is also used for backing up a return address at subroutine call, the levels of the multiple interrupt are limited to the number of levels of the address stack registers.

For instance, when 4 levels are used in a subroutine call as shown in Fig. 12-10, only up to 3 levels can be used for multiple interrupt.

However, when the numbers of levels of the subroutine call and multiple interrupt exceeds 7, the program can be written using stack manipulation instructions, "PUSH" and "POP".

The example shows how address stack registers are backed up using the "PUSH" and "POP" instructions.

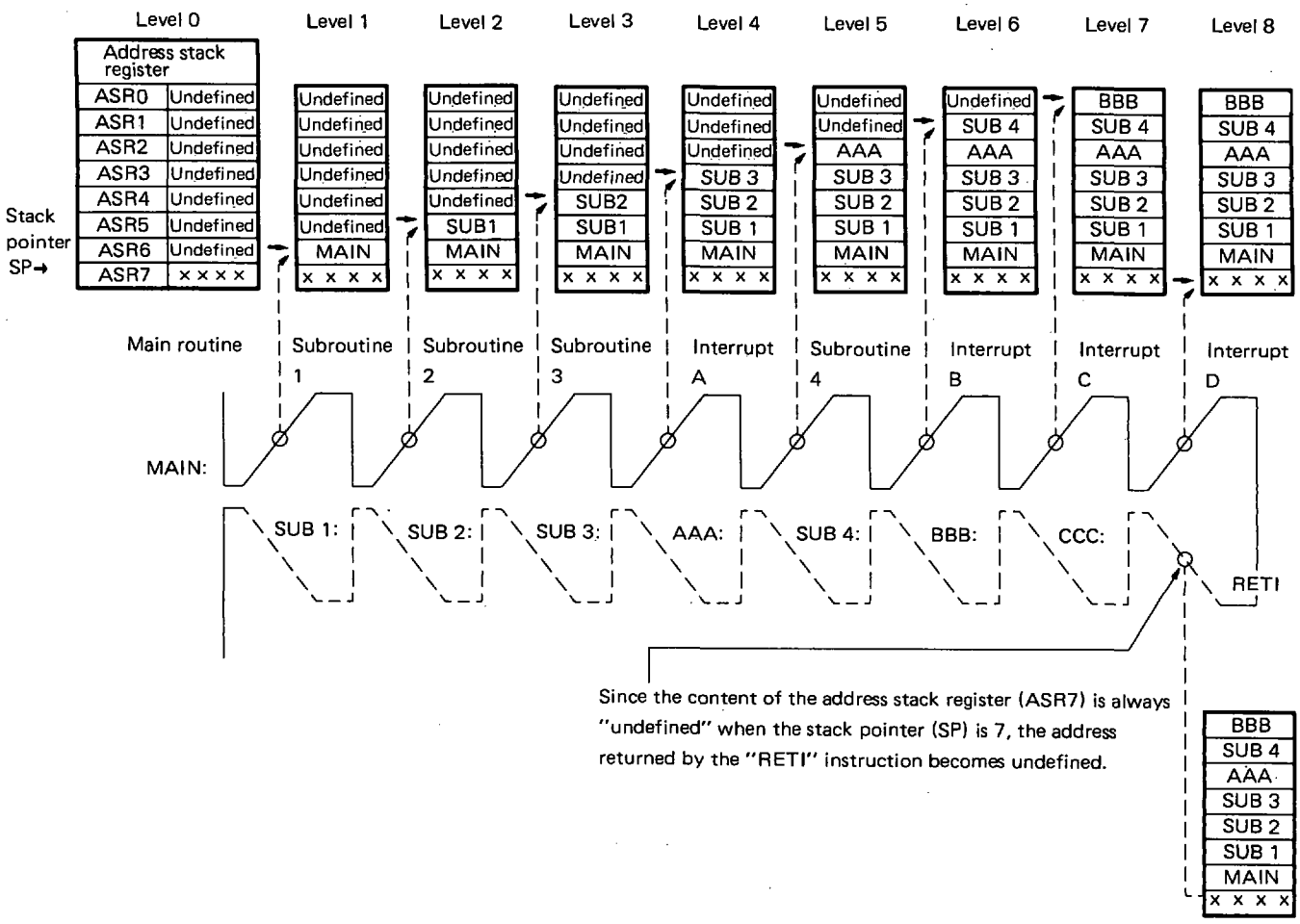
See also Section **5.7, "Stack Nesting Levels and PUSH Instruction and POP Instruction"**.

The following points must be noted when the content of an address stack register is backed up and returned using the "PUSH" instruction and "POP" instruction.

An address stack register must be backed up when the number of nesting levels exceeds 7.

Normally, backup processing is performed during interrupt processing at low priority.

Fig. 12-10 Operation of address stack registers (ASR)



Example:

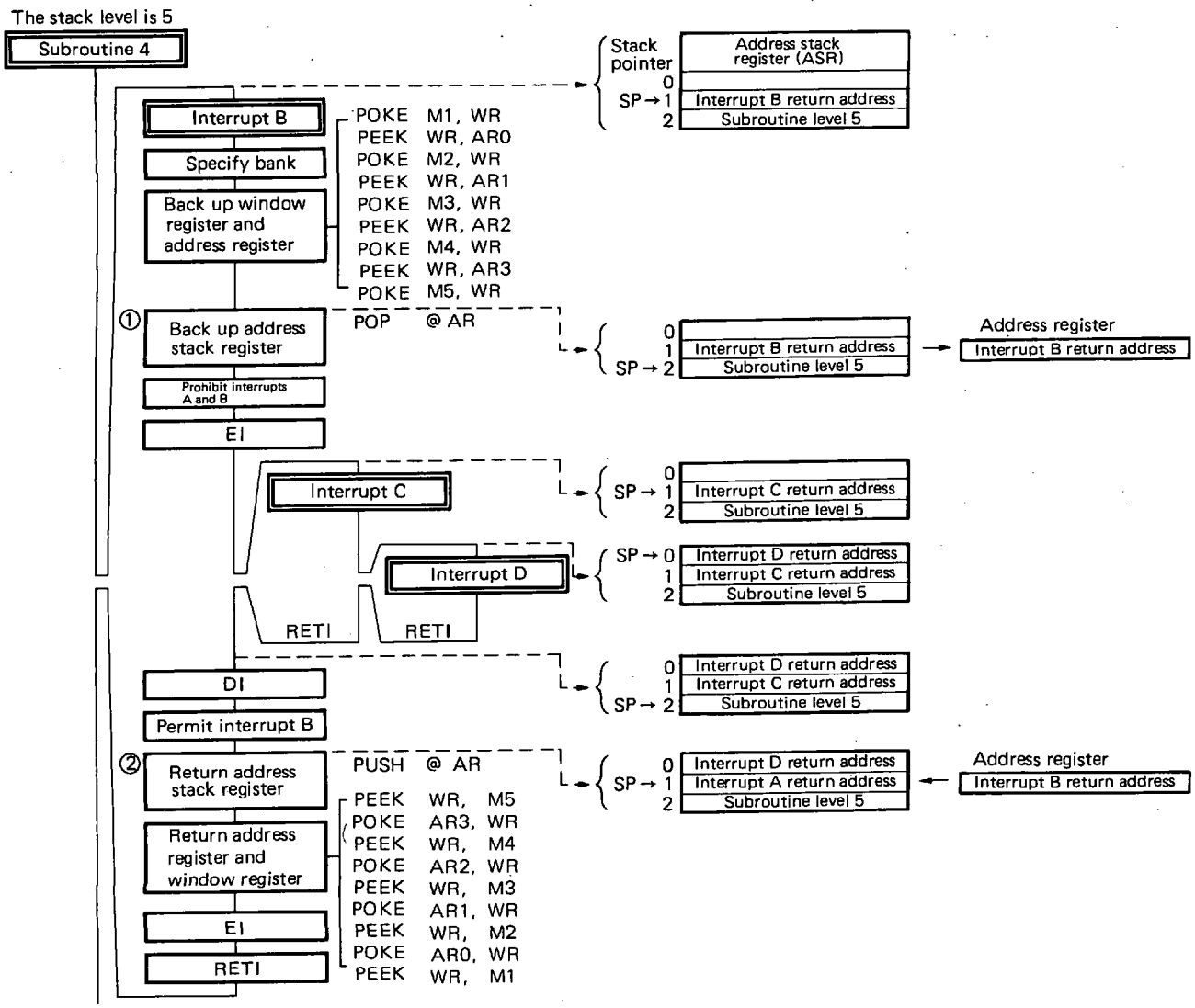
Saving address stack registers using the PUSH and POP instructions

The operation flow chart is shown below.

The program of subroutine 4 (stack level 5) shown in Fig. 12-10 is executed, interrupt factors B, C, and D are permitted, and the priority sequence is D > C > B.

Normally, if multiple interrupts are accepted in the sequence of B → C → D as shown in Fig. 12-10, overflow occurs in the address stack register when interrupt D is accepted.

Consequently, the contents of the address stack registers are backed up in processing of interrupt B as shown below.



In this example, since the return address of interrupt B is backed up in the address register using the "POP" instruction of ①, there are two remaining interrupt levels.

That is, the level of the address stack register for interrupt C and interrupt D are allocated.

Since the return address of interrupt B sent from the address register is returned to the address stack register by the "PUSH" instruction of ②, the "RETI" instruction from interrupt B is validated.

12.9.4 Backing Up System Registers and Control Registers

When multiple interrupt is used, the contents of the system register and control register which change during interrupt processing must be backed up in advance.

The backup area for the contents must be the location for each interrupt factor.

In addition, interrupts with priority equal to or lower than that of the interrupt currently accepted must be prohibited and interrupts with higher priority must be permitted.

Since interrupt with higher priority indicates higher urgency, interrupts with the highest priority must be permitted.

Consequently, the backing up of contents of system registers and control registers should be performed following the "processing which permits an interrupt of higher priority".

The example below shows processing of "permission of interrupt with higher priority" and "saving contents of system registers and control registers" in an interrupt processing routine.

Example:

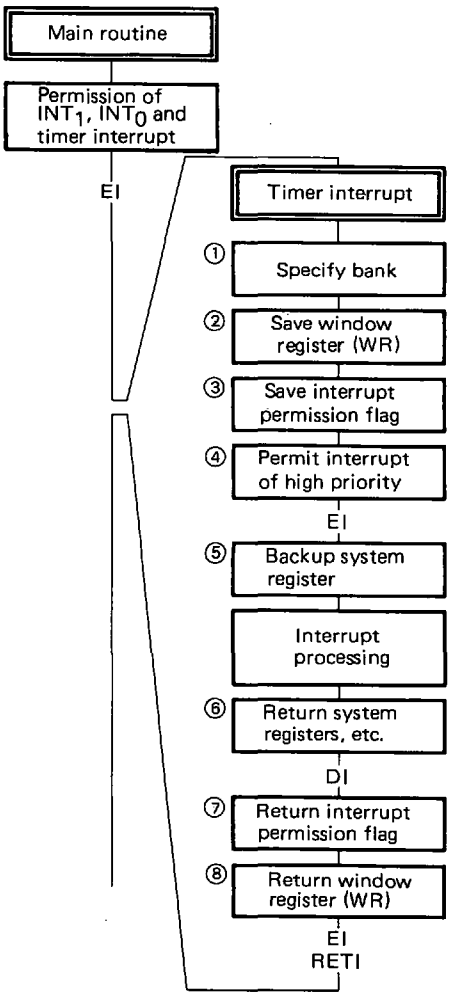
Permission of interrupt and backup processing in multiple interrupt

The INT_0 pin, INT_1 pin, and timer interrupt are used in the following priority (software priority).

INT_1 pin > timer > INT_0 pin

Timer interrupt is accepted at level 1. The program example and flow chart are shown below.

Flow chart



Program example

```
MOV WR. #0111B      ; Interrupt byttimer, INT0 pin, and INT1 pin are
POKE INTPM 1. WR    ; permitted by the interrupt permission 1 register

The contents of the bank register (BANK), Index Enable flag
(IXE), and program counter (PC) are backed up automatically.

BANK 1              ; Specify BANK1 as the bank of the memory
    (Built-in macro); used for backing up data.
POKE M 1. WR        ; The window register is saved.

PEEK WR. INTPM 1    ; The interrupt permission 1 register is
POKE M2. WR         ; saved in M2 of the data memory.
MOV WR. #0010B      ; Interrupt of INT1 pin is permitted by
POKE INTPM 1. WR    ; the interrupt permission 1 register.

PEEK WR. RPH        ; System control registers and control registers
POKE M 3. WR        ; other than the window register (WR) and interrupt
      ⋮             ; permission 1 register are backed up.

      ⋮

PEEK WR, M3         ; System registers and control registers other
POKE INTPM 1, WR    ; than the window register (WR) and interrupt
                    ; permission 1 register are returned.
PEEK WR, M2         ; Interrupt permission 1 register is returned.
POKE INTPM 1, WR    ;
PEEK WR, M1         ; Window register is returned.
```

In ①, specify the contents of the bank of the data memory used for backing up system registers, etc.
Since BANK0 is determined as the bank if interrupt is accepted, this instruction is not required if the memory used for backing up data is BANK0.
In ②, back up the content of the window register in M1 of the data memory.
Since the "POKE" instruction is used in this case, the address of data memory M1 must be 40H or higher. Since the window register is used as a work area for backing up subsequent data, it must be backed up first.
In ③, save the interrupt permission flags (IP0, IP1, IPTM flags) used when interrupt is accepted.
For this backup, for instance, when control is returned to the main routine in this case, all the interrupts by the INT_0 pin, INT_1 pin, and timer must be permitted. In this case, the interrupt by the INT_0 pin must be prohibited and control must be returned when a timer interrupt is accepted during INT_0 pin interrupt processing because the priority of timer interrupt is higher than that of INT_0 pin interrupt.
In ④, permit interrupt of the INT_1 pin with higher priority than that of the timer interrupt and subsequently, permit all the interrupts using the "EI" instruction.
In ①, ②, ③, and ④, save and return system registers and control registers. In this case, an interrupt by the INT_1 pin with the highest priority can be permitted.
System registers and control registers are backed up and returned by ⑤ and ⑥. In this case, interrupts with higher priority can be permitted.
By performing the same backup processing when an interrupt of the INT_1 pin with higher priority is accepted, the contents of the system registers and control registers can be maintained unchanged when control is returned from the INT_1 pin interrupt processing.
In ⑦ and ⑧ return the interrupt permission flag and window register.
In this case, all the interrupts must have been permitted.
If the instruction in ⑦ which permits an interrupt is executed in the "EI" state and if a timer interrupt request has been issued, the window register is saved in ② without returning the window register in ⑧ and as a result, the content of the window register cannot be returned.

12.10 NOTES ON USING INTERRUPT

An error occurs in assembler AS17K when a POKE command is used for the interrupt request flag. Therefore, an error also occurs when assembler built-in macrocommands SETn, CLRn, NOTn, and INITFLG are used to prevent that other interrupt request are canceled when a POKE command is used for the interrupt request flag.

An example is described below. (In this case, assume that no assemble error occurs when a POKE command is used.)

Example 1:

Polling is performed for the INT_1 pin and serial interface 1 using the timer interrupt.

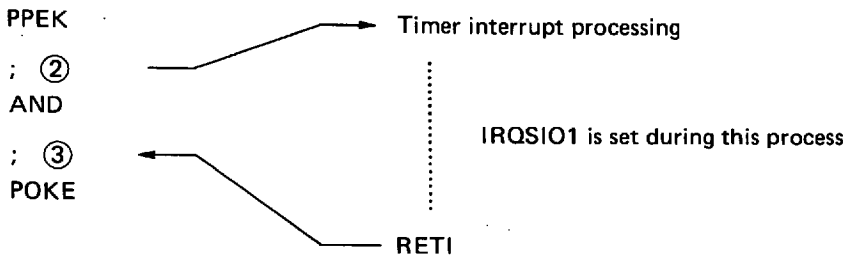
```
        SET1    IPTM
        EI
          :
          :
MAIN:
          :
          :
        SKF1    IRQ1
        BR      NEXT
INT1:
        Processing of INT1 pin
        ; ①
        CLR1    IRQ1
NEXT:
        SKF1    IRQSIO1
        BR      MAIN
SIO1:
        Processing of SIO1
        CLR1    IRQSIO1
        BR      MAIN
```

Instructions of the above program ① are expanded as follows.

```
        PEEK    WR, .MF.IRQ1 SHR 4
        ; ②
        AND     WR, #.DF. (NOT IRQ1) AND 0FH
        ; ③
        POKE    .MF.IRQ1 SHR 4, WR
```

If a timer interrupt occurs during execution of instruction in ② and an interrupt request of serial interface 1 is issued (IRQSIO1 is set) during timer interrupt processing as shown below, the instruction of ③ is executed when control is returned from the timer interrupt and as a result the IRQSIO1 flag is reset.

In particular, an interrupt request of serial interface 1 sets serial communication to a wait state once it is issued and the interrupt request is not issued any more.



The POKE command for each interrupt request flag is thus inhibited using an assembler.
To poll the interrupt request flag assuming that other interrupt requests are canceled, follow example 2 below.

Example 2:
Setting one interrupt request flag

```
SETIRQ1  MAC    IRQFLG
PEEK     WR,.MF.IRQFLG SHR 4
OR       WR,#.DF.IRQFLG AND 0FH
DW       (00111B SHL 11) OR (.DF.(IRQFLG AND 0F00H)) OR
         (0010B SHL 4) OR (.DF.(IRQFLG SHR 4) AND 0FH
              ; Defines the POKE command using a DW command.
         ENDM
```

The macrocommands described above are added to a device file (AS17005) as an "IRQ, MAC" file. The macrocommands that are supplied using the "IRQ, MAC" file are as follows:

.SETIRQn (Corresponds to internal macrocommand SETn.)
.CLRIRQn (Corresponds to internal macrocommand CLRn.)
.NOTIRQn (Corresponds to internal macrocommand NOTn.)
.INITIRQ (Corresponds to internal macrocommand INITFLG.)

The above macrocommands can be directly used when the "IRQ, MAC" file is included in a source module.
For how to use the "IRQ, MAC" file, see the device file (AS17005) user manual.

13. TIMER FUNCTION

The timer function is used for time management during the program creation.

13.1 CONFIGURATION

Fig. 13-1 shows the timer configuration.

The timer consists of a timer carry flip-flop (timer carry FF) block and timer interrupt block as shown in Fig. 13-1.

The clock generator circuit that sets the timer carry flip-flop and timer interrupt times consists of a 4.5 MHz frequency divider, selector A, selector B, and timer mode select register (TMMD, address 09H) in the control register.

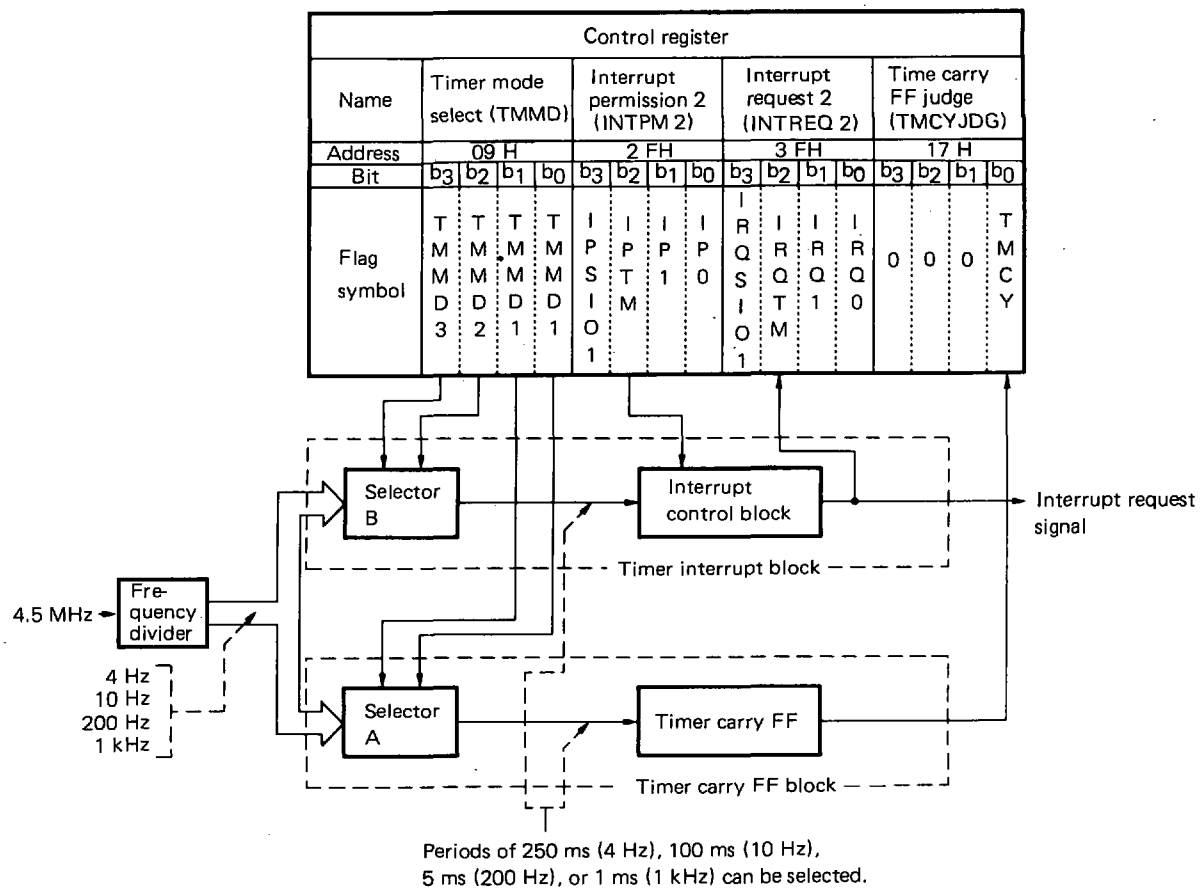
13.1.1 Timer Carry Flip-Flop Block Configuration

The timer carry flip-flop block consists of a selector A, timer carry flip-flop, and timer carry flip-flop judge register (TMCYJDG, address 17H) in the control register as shown in Fig. 13-1.

13.1.2 Timer Interrupt Block Configuration

The timer interrupt block consists of a selector B, interrupt control block, and interrupt permission 2 register (INTPM 2, address 2FH) and interrupt request 2 register (INTREQ 2, address 3FH) in the control register as shown in Fig. 13-1.

Fig. 13-1 Timer configuration



13.2 FUNCTIONS

The timer has timer carry flip-flop detection and timer interrupt functions.

The time is managed during the timer carry flip-flop detection when the status of the timer carry flip-flop that is set periodically is detected using a program. The time is managed during timer interrupt when an interrupt is made periodically.

The timing at which the timer carry flip-flop is set (1) and the timer interrupt is issued is controlled using a timer time setting pulse from selectors A and B. A timer time setting pulse of 4 Hz (250 ms), 10 Hz (100 ms), 200 Hz (5 ms), and 1 kHz (1 ms) can be selected when data is sent to the timer mode select register (TMMD, address 09H). The timer time setting pulse can also be selected independently using a timer carry flip-flop and timer interrupt.

The configuration and functions of the timer mode select register are described in Section 13.2.1. Fig. 13-2 shows the timer time setting pulse waveforms. The timer time set pulse is generated by dividing a device operating frequency of 4.5 MHz, so it is also shifted proportionally when the 4.5 MHz frequency is shifted.

13.2.1 Timer Mode Select Register Configuration and Functions

The timer mode select register sets the time of an internal timer (timer carry flip-flop and timer interrupt). The timer carry flip-flop and timer interrupt time intervals can be set independently. The configuration and functions are shown below.

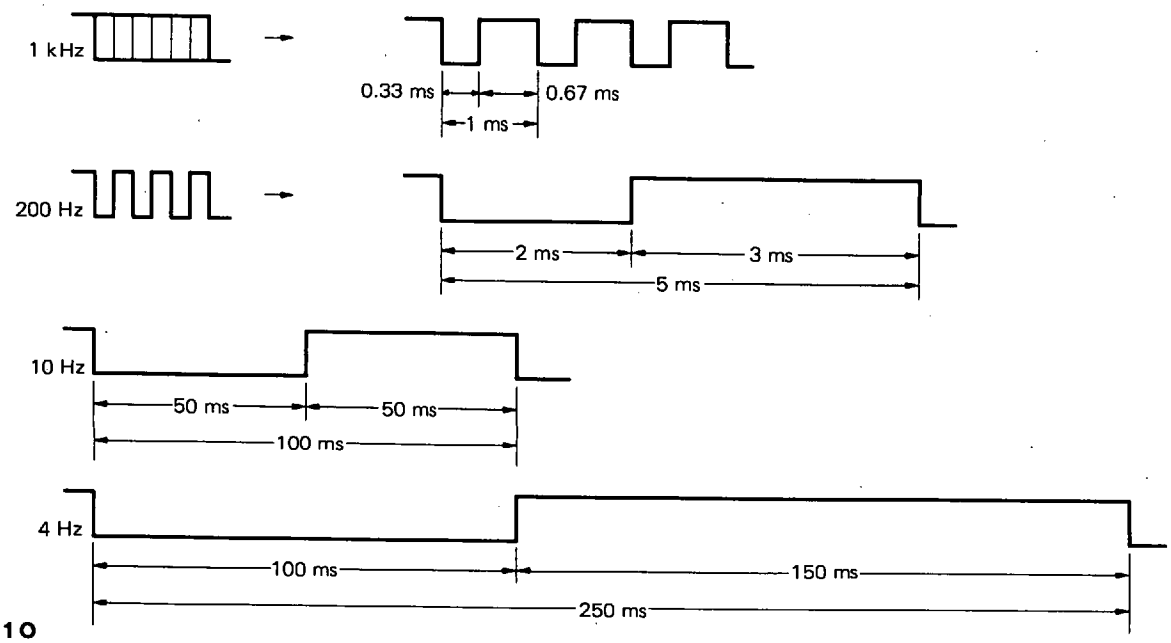
Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
Timer mode select register (TMMODE)	TMMD3	TMMD2	TMMD1	TMMD0	09H	R/W

b_1	b_0	Sets the timer carry flip-flop time interval.
0	0	100 ms
0	1	250 ms
1	0	5 ms
1	1	1 ms

b_3	b_2	Sets the timer interrupt time interval.
0	0	100 ms
0	1	250 ms
1	0	5 ms
1	1	1 ms

Reset		b_3	b_2	b_1	b_0
	Power on	0	0	0	0
	Clock stop	0	0	0	0
	CE	Former state is held.			

Fig. 13-2 Timer time set pulse waveforms



13.3 TIMER CARRY FLIP-FLOP

The timer carry flip-flop is set at the leading edge of a timer carry flip-flop setting pulse that is set (1) using the low-order two bits (TMMD1 and TMMD0 flags) of a timer mode select register (TMMD, address 09H).

The timer carry flip-flop information corresponds to the least significant bit (TMCY flag) of a timer carry flip-flop judge register (TMCYJDG, address 17H) at a ratio of 1 to 1. When the timer carry flip-flop is set (1), the TMCY flag is set (1) at the same time.

The TMCY flag is reset (0) when information is read in the window register using a PEEK command (Read & Reset). When the TMCY flag is reset (0), the timer carry flip-flop is also reset (0). A timer of the time set using the timer mode select register can be created by reading the TMCY flag using a program. Pay attention to the following when using the timer carry flip-flop:

The timer carry flip-flop is in set inhibit mode when the supply voltage is turned on (during V_{DD} reset) and cannot be set until the TMCY flag information is read using a PEEK command.

A logical "0" is always read when the TMCY flag is read after power on reset. After that, data is set (1) corresponding to the time set using the timer mode select register.

The timer carry flip-flop also controls the timing of the reset (CE reset) employing a CE pin. When the CE pin is changed from low to high, the CE pulse is reset in synchronization with the timing when the timer carry flip-flop is set. Therefore, a power failure can be detected when the TMCY flag information is read during system reset (power on reset and CE reset). For more information, see Section **13.4, "Cautions when Using Timer Carry Flip-Flop"** and Section **15, "Reset"**.

The TMCY flag is a read only flag, so it is not influenced during the POKE command write operation. However, an error occurs when an assembler (AS17K) of the 17K series is used. For more details, see Section **10.4, "Cautions when Using Register File"**.

13.3.1 Timer Carry Flip-Flop Judge Register Configuration and Functions

The timer carry flip-flop judge register detects the timer carry flip-flop status of an internal timer. The configuration and functions are shown below:

Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
Timer carry flip-flop judge register (TMCYJDG)	0	0	0	TMCY	17H	R & Reset

TMCY	
Detects the timer carry flip-flop status.	
0	The timer carry flip-flop is not set.
1	The timer carry flip-flop is set.

Fixed at "0".

Reset	b_3	b_2	b_1	b_0
Power on	0	0	0	0
Clock stop				1
CE				1

The TMCY flag is set corresponding to the time set using a timer mode select register (TMMODE). This flag is detected through the window register using a PEEK command. The value when the TMCY flag is set at that time is transferred to the window register. The TMCY flag is then reset (Read & Reset).

The TMCY flag is set to "0" during the power on reset and is set to "1" during the CE reset and the CE reset after clock stop command execution. This flag can thus be used as a power failure detection flag.

The TMCY flag is not reset until the PEEK command is executed after V_{DD} voltage is turned on. When the PEEK command is executed, the flag is set corresponding to the time set using the timer mode select register.

13.3.2 Timer Using TMCY Flag

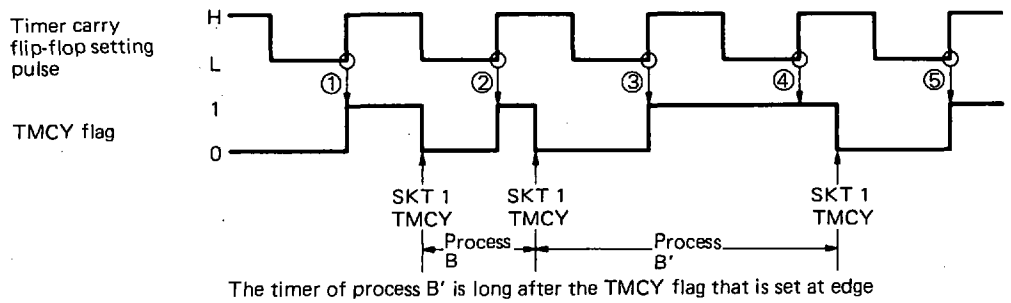
A program example is shown below.

```
Example: INITFLG NOT TMMD3, NOT TMMD2, NOT TMMD1, TMMD0
                    ; Built-in macroinstruction
                    ; The timer carry flip-flop set time is set to 250 ms.
LOOP:
  SKT1   TMCY       ; Built-in macroinstruction
                    ; The TMCY flag is tested. The flag branches into NEXT when it is "0".
  BR     NEXT
  ADD    M1, #0100B ; Numerical value 4 is added to data memory M1.
  SKT1   CY         ; Built-in macroinstruction
                    ; The CY flag is tested.
  BR     NEXT       ; The CY flag branches into NEXT when it is "0".
  [Process A]       ; The CY flag branches into process A it is "1".
NEXT:
  [Process B]       ; The CY flag branches into LOOP after process B is executed.
  BE     LOOP
```

Process A in the above program is executed every second. Pay attention to the following during the program creation:

(1) The TMCY flag detection time must be shorter than the time for which the timer carry flip-flop is set (1) because the timer carry flip-flop setting status is lost when the time of process B exceeds 250 ms as shown in Fig. 13-3.

Fig. 13-3 TMCY flag detection and timer carry flip-flop



The timer of process B' is long after the TMCY flag that is set at edge (2) is detected, so the status of the TMCY flag that is set at edge (3) is lost.

13.3.3 Timer Error Using TMCY Flag

Two errors can occur when using a TMCY flag, a TMCY flag detection time error and an error when the timer carry flip-flop setting time is altered.

The two errors are described below.

(1) TMCY flag detection time error

The TMCY flag detection time must be shorter than the time for which the timer carry flip-flop is set (1) as described in Section **13.3.1.**

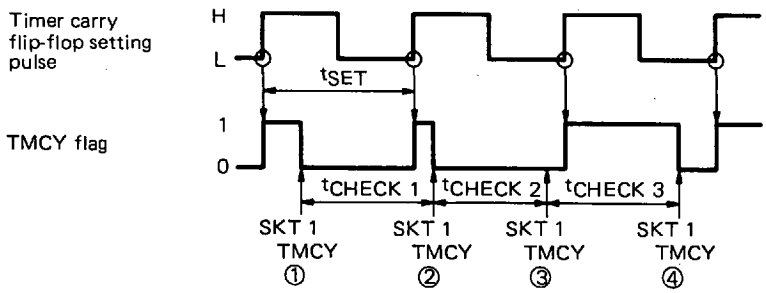
Assume that the interval of the time at which the TMCY flag is detected is t_{CHECK} and that the interval (250 ms, 100 ms, 5 ms, or 1 ms) of the time at which the timer carry flip-flop is set is t_{SET}. The relation shown below is required.

$$t_{CHECK} < t_{SET}$$

The timer error when the TMCY flag is detected as shown in Fig. 13-4 is as follows:

$$0 < \text{error} < t_{CHECK}$$

Fig. 13-4 TMCY flag detection time error



The TMCY flag is set to "1" when it is detected at edge ② as shown in Fig. 13-4. The timer is thus updated. The TMCY flag is set to "0" when it is detected at edge ③.

Therefore, the timer is not updated until the flag is detected again at edge ④. The time set in the timer is thus extended by t_{CHECK3}.

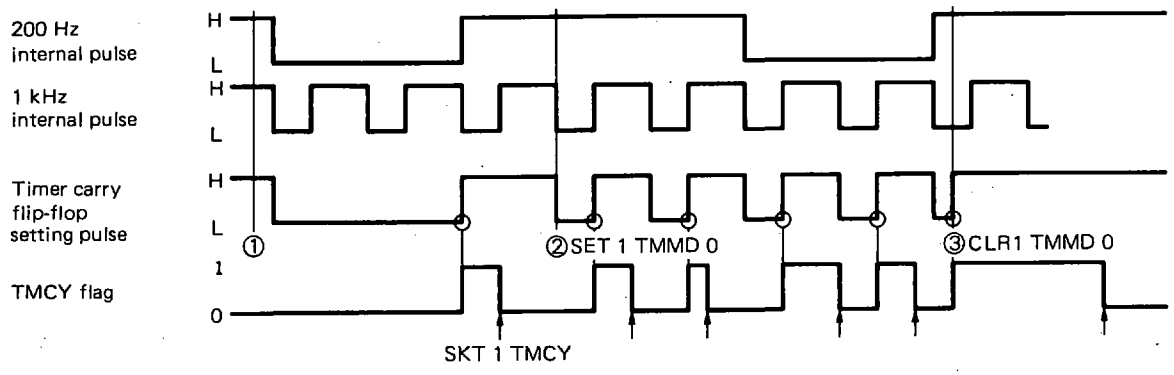
(2) Error when timer carry flip-flop setting time is altered

The timer carry flip-flop setting time is set using the TMMD1 and TMMD0 flags of a timer mode select register. Four timer time setting pulses (1 kHz, 200 Hz, 10 Hz, and 4 Hz) can be selected as shown in Figs. 13-1 and 13-2. The four pulses are activated independently. When the timer time setting pulses are selected using the TMMD1 and TMMD0 flags, errors occur as shown below.

Example:

```
; ①
  INITFLG NOT TMMD3, NOT TMMD2, TMMD1, NOT TMMD0
                    ; Built-in macroinstruction
                    ; The timer carry flip-flop setting pulse is set to 200 Hz (5 ms).
  Process A
; ②
  SET1    TMMD0     ; Built-in macroinstruction
                    ; The timer carry flip-flop setting pulse is set to 1 kHz (1 ms).
  Process A
; ③
  CLR1    TMMD0     ; Built-in macroinstruction
                    ; The timer carry flip-flop setting pulse is set to 200 Hz (5 ms).
```

The timer carry flip-flop setting pulse at that time is selected as shown below.



The TMCY flag holds the former status (see ② in the figure) when a selected pulse falls by selecting the timer carry flip-flop setting time as shown in the figure above. The TMCY flag is set (1) when a selected pulse rises (see ③ in the figure).

The 4 Hz (250 ms) and 10 Hz (100 ms) pulses are selected as in the 200 Hz (5 ms) and 1 kHz (1 ms) pulses shown in the example above. The error which occurs when the timer carry flip-flop setting time is selected before the first TMCY flag is set as shown in Fig. 13-5 is as follows:

$-t_{SET} < \text{error} < t_{CHECK}$

where,

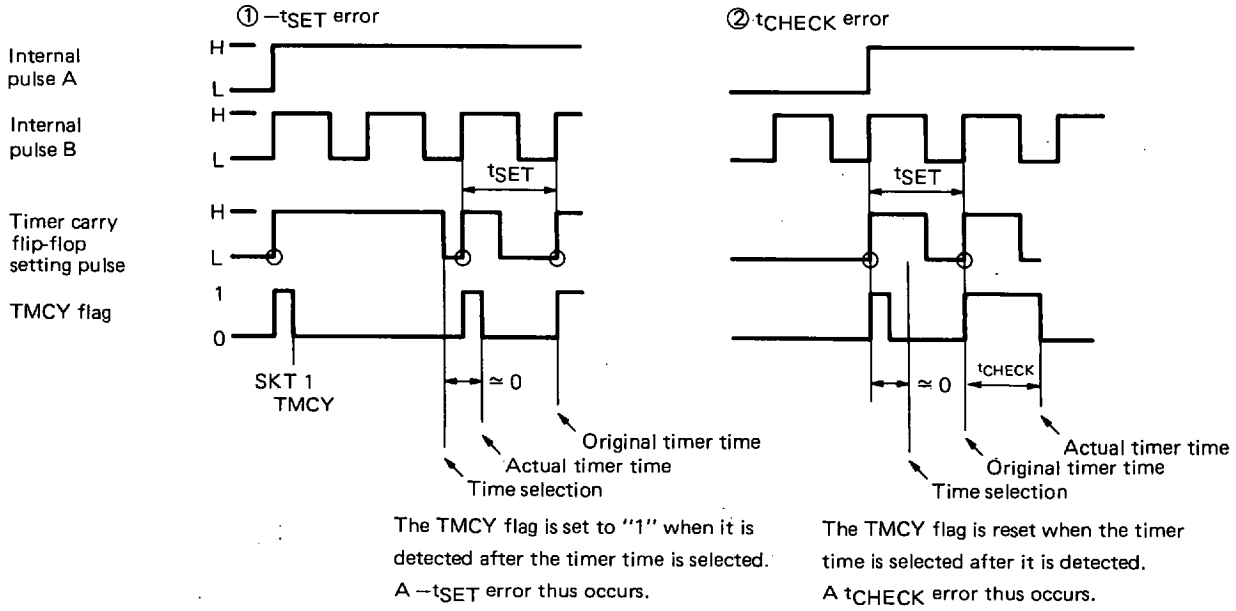
t_{SET} : Selected timer carry flip-flop setting time

t_{CHECK} : TMCY flag detection time

The 4 Hz, 10 Hz, 200 Hz, and 1 kHz internal pulses have the corresponding phase difference. This phase difference is less than the selected pulse time, so it is contained in the error above.

For more information on the pulse phase difference, see Section **13.6, "Cautions during Timer Interrupt"**.

Fig. 13-5 Errors when timer carry flip-flop setting time is altered from A to B



13.4 CAUTIONS WHEN USING TIMER CARRY FLIP-FLOP

The timer carry flip-flop is used for a reset sync signal during reset employing a CE pin (CE reset) as well as a timer function. The CE pulse is reset when the next timer carry flip-flop setting pulse rises after the CE pin is changed from low to high. Pay attention to the following:

(1) The sum of the timer updating time and TMCY flag detection time interval must be less than the timer carry flip-flop setting time.

(2) A timer must be corrected for every CE reset when a program for which the timer operates at all times is created after supply voltage is turned on (power on reset) irrespective of CE resetting.

(3) The TMCY flag detection has priority over a reset sync signal during the CE reset. If they fall on the same time, the CE reset is delayed one operation.

For more details of Steps (1), (2), and (3), see Section **13.4.1** through **13.4.3.**

13.4.1 Timer Updating Time and TMCY Flag Detection Time Interval

As described in **13.3.1**, the TMCY flag detection time interval t_{SET} must be less than the time set in the timer carry flip-flop. The timer may not operate normally during the CE reset when the timer updating time is long even if the TMCY flag detection time interval is shorter.

Therefore, the conditions below must be satisfied.

$t_{CHECK} + t_{TIMER} < t_{SET}$

where,

t_{CHECK} : TMCY flag detection time interval
t_{TIMER} : Timer updating time
t_{SET} : Timer carry flip-flop setting time

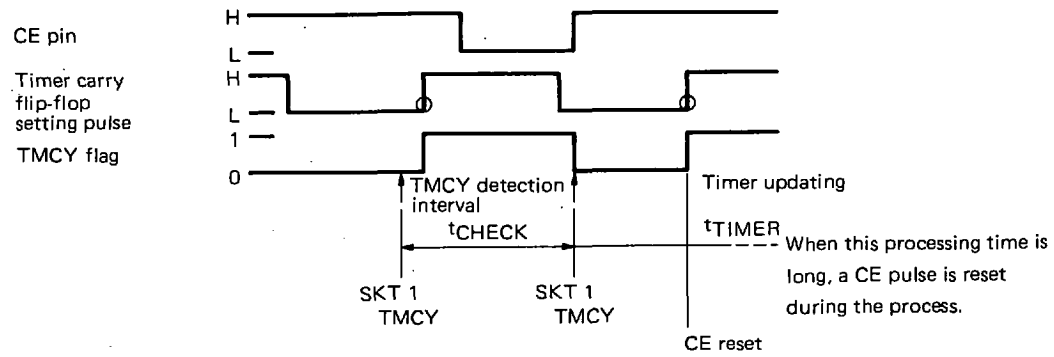
An example is shown below.

Example:

Timer updating and TMCY flag detection time interval

```
START:                  ; Program address 0000H
    INITFLG NOT TMMD3, NOT TMMD2, NOT TMMD1, NOT TMMD0
                        ; Built-in macroinstruction
                        ; The timer carry flip-flop setting time is set to 100 ms.
TIMER:
    ; ①
        SKT 1  TMCY     ; Built-in macroinstruction
                        ; The TMCY flag is tested.
        BR     AAA      ; The TMCY flag branches into AAA when it is "0".
        [Timer updating]
        BR     TIMER
AAA:
        [Process A]
        BR     TIMER
```

The timing chart for the above program is shown below.



13.4.2 Timer Carry Flip-Flop Correction During CE Reset

A timer correction example during the CE reset is described below.

Assume that the timer carry flip-flop is used for power failure detection and a clock timer when the timer correction is required during the CE reset.

The timer carry flip-flop is reset (0) when the supply voltage is turned on (power on reset) and is in set inhibit mode until the TMCY flag is read using a PEEK command.

When the CE pin is changed from low to high, the CE pulse is reset in synrhonization with the leading edge of a timer carry flip-flop setting pulse. The TMCY flag is then set (1), starting.

When the TMCY flag status is detected during the system reset (power on reset and CE reset), a power on reset occurred if the TMCY flag is "0". A CE reset occurred if it is "1" (Power failure detection).

The clock timer operation at that time must be continued during the CE reset. However, the TMCY flag is reset (0) when it is read to detect the power failure. The status in which the TMCY flag is set is thus skipped. Consequently, the clock timer must be updated when the TMCY flag status is determined to be a CE reset owing to power failure detection. The timer updating example is shown below.

For more information on the power failure detection, see Section **15.6, "Power Failure Detection".**

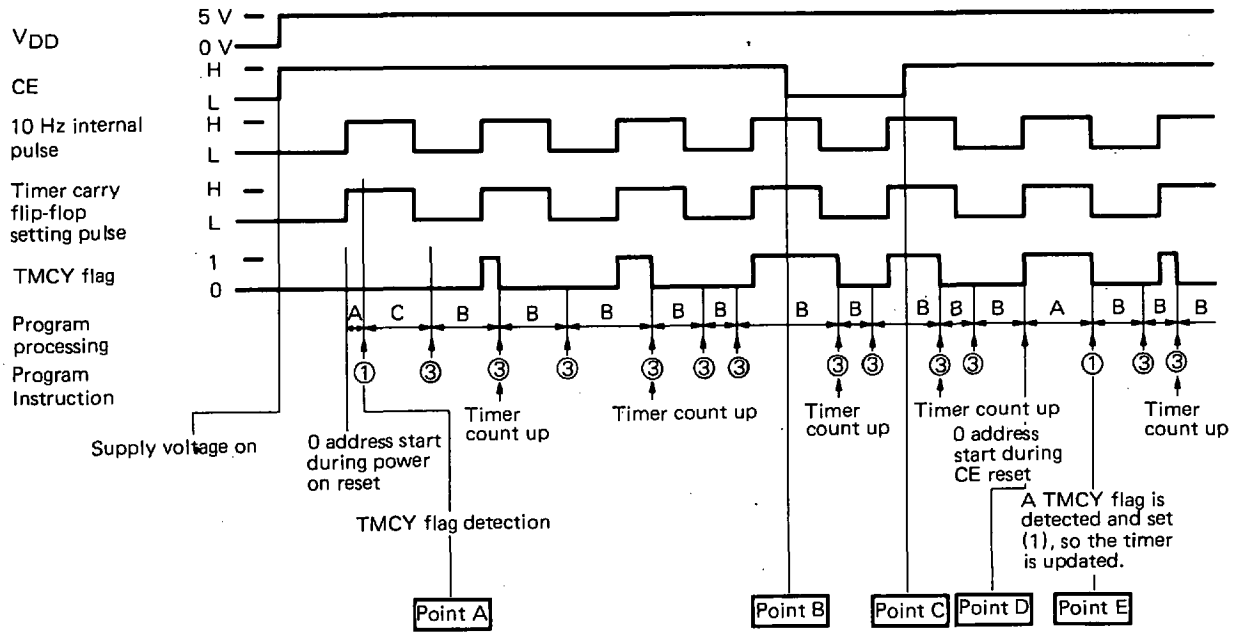
Example: Timer correction during CE reset

When the timer carry flip-flop detects power failure and updates a timer

```
START:                      ; Address 0000H
        Process A
  ; ①
        SKT1    TMCY        ; Built-in macroinstruction
                            ; The TMCY flag is tested.
        BR      INITIAL     ; The TMCY flag branches into INITIAL when it is "0" (power failure
                            ; detection).
BACKUP:
  ; ②
        100 ms timer        ; The timer is corrected owing to backup
        updating            ; (CE reset).

LOOP:
  ; ③
        Process B           ; Process B is executed.
        SKF1    TMCY        ; The TMCY flag is tested, and the timer is updated.
        BR      BACKUP
        BR      LOOP
INITIAL:
        INITFLG NOT TMMD3, NOT TMMD2, NOT TMMD1, NOT TMMD0
                            ; Built-in macroinstruction
                            ; The timer carry flip-flop setting time is set to 100 ms to execute process C
                            ; owing to power failure (power on reset).
        Process C
        BR      LOOP
```

The timing chart for the above program is shown in Fig. 13-6.

Fig. 13-6 Timing chart



When supply voltage V_{DD} is first applied, a program is started from address 0000H during 10 Hz internal pulse rise as shown in Fig. 13-6. The TMCY flag is reset (0) during the power on sequence when it is detected at point A. The TMCY flag status at that time is determined to be power failure (power on reset). Therefore, process C is executed, and the timer carry flip-flop setting pulse is set to 100 ms.

The TMCY flag information is read once at point A, so the TMCY flag is set (1) every 100 ms after that.

The program counts up the timer while executing process B, unless a clock stop command is executed, even if the CE pin goes low at point B and goes high at point C. When the CE pin rises from low to high at point C, the CE pulse is reset at the leading edge (point D) of the next timer carry flip-flop setting pulse and the program is started from address 0000H. When the TMCY flag is detected at point E, the TMCY flag status is determined to be a backup (CE reset) because the TMCY flag is set (1).

If the timer is not incremented 100 ms at point E, it is delayed 100 ms every time a CE pulse is reset. The TMCY flag setting is skipped twice if process A execution exceeds 100 ms when power failure is detected at point E. Process A must thus be executed in less than 100 ms.

Timer carry flip-flop setting pulses of 250 ms, 5 ms, and 1 ms are also selected in the same manner as the above.

The TMCY flag must be less than the timer carry flip-flop setting time owing to the power failure detection after the program is started from address 0000H.

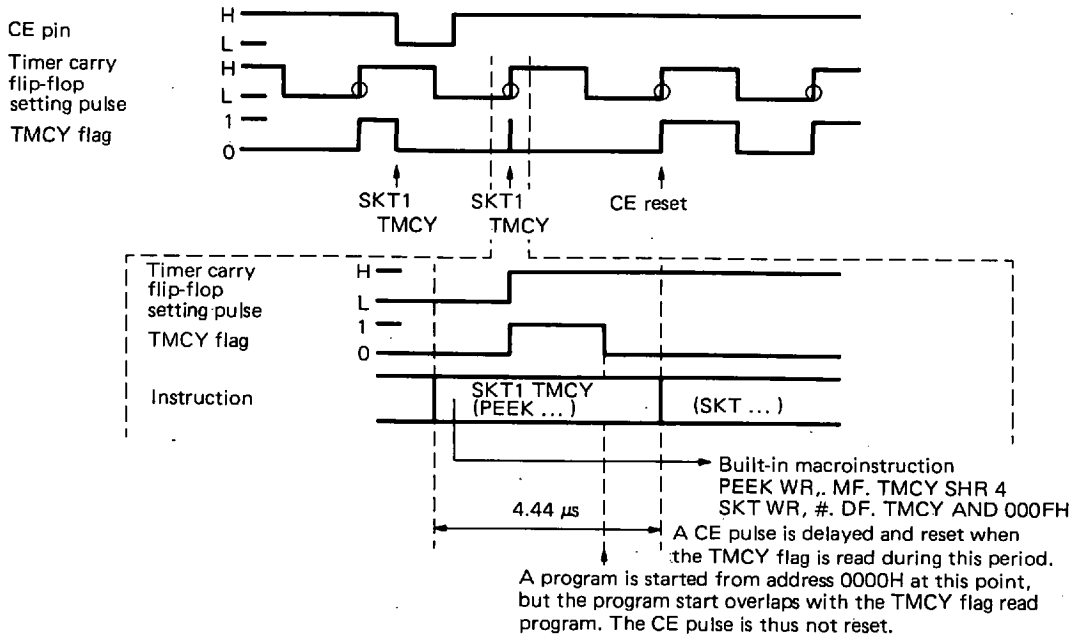
13.4.3 When TMCY Flag Detection and CE Reset Overlap

As described in Section **13.4.2**, a CE pulse is reset simultaneously when the TMCY flag is set (1). If the TMCY flag read instruction and CE reset overlap, the TMCY flag read instruction has priority.

If the next TMCY flag setting (timer carry flip-flop setting pulse rising) overlaps with the TMCY flag read instruction when the CE pin is changed from low to high, the CE pulse is reset at the timing at which the next TMCY flag but one is set.

This operation is shown in Fig. 13-7.

Fig. 13-7 CE reset and TMCY flag read overlap



The CE pulse may never be reset when the system is programed so that the TMCY flag is detected periodically and the TMCY flag detection time interval coincides with the TMCY flag setting time. Therefore, note to the following.

One instruction cycle is 4.44 μs (1/225 kHz). For example, the program for which a TMCY flag is detected every 225 instructions reads the TMCY flag every 1 ms (4.44 μs x 225). The CE pulse is never reset when the TMCY flag is set and detected at the same time even if a timer time setting pulse of 1 ms, 5 ms, 100 ms, or 250 ms is selected.

Do not write periodic programs that satisfy the conditions below.

$$\frac{t_{SET} \times 225}{X} = n \text{ (n: Natural number)}$$

where,

t_{SET} : TMCY flag setting time

X : Periodic X-step of TMCY flag read instruction

In this case, do not write an X-step program in which natural number n exists.

A program example that satisfies the conditions is shown below. Never create such a program.

Example:

```
        Process A
        INITFLG NOT TMMD3, NOT TMMD2, TMMD1, TMMD0
                        ; Built-in macroinstruction
                        ; The timer carry flip-flop setting pulse is set to 1 ms.
LOOP:
  ; ①
        SKT1    TMCY    ; Built-in macroinstruction
        BR      BBB
AAA:
        221 Steps
        BR      LOOP
BBB:
        221 Steps
        BR      LOOP
```

In this example, TMCY flag read instruction ① is repeated every 225 instructions. Therefore, the CE pulse cannot be reset when a TMCY flag is set at the timing of instruction ①.

13.5 TIMER INTERRUPT

An interrupt request is issued at the trailing edge of a timer interrupt pulse that is set using the high-order two bits (TMMD3 and TMMD2 flags) of a timer mode select register (TMMD, address 09H) during timer interrupt.

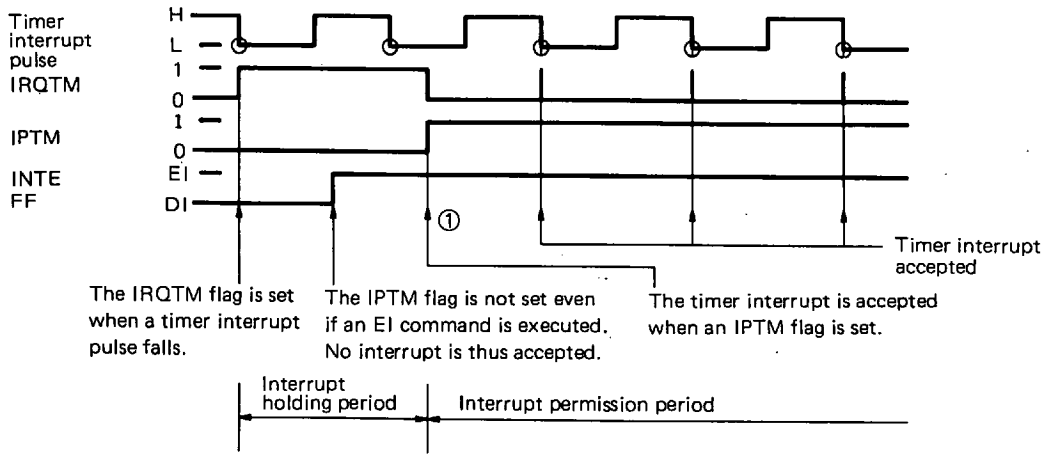
The timer interrupt request corresponds to the IRQTM flag of an interrupt request 2 register (INTREQ2, address 3FH) at a ratio of 1 to 1. When the timer interrupt request is issued, an IRQTM flag is set (1). When the timer interrupt pulse falls, the IRQTM flag is set (1).

As described in Section **12, "Interrupt"**, an EI command indicating all interrupt permission instructions must be executed and the timer interrupt permission must be set and the interrupt request issued during the timer interrupt.

The timer interrupt is permitted by setting (1) the IPTM flag of an interrupt permission 2 register (INTPM2, address 2FH). Consequently, the EI command is executed during the timer interrupt. The timer interrupt is accepted if the IRQTM flag is set (1) when the IPTM flag is set (1). When the timer interrupt is accepted, the program flow changes to program memory address 0003H. The IRQTM flag is reset (0) when the timer interrupt is accepted.

Fig. 13-8 shows the relation between the timer interrupt pulse and IRQTM flag.

Fig. 13-8 Relation between timer interrupt pulse and IRQTM flag



Notice that the timer interrupt can be accepted during the ET command execution and IPTM flag setting if the IRQTM flag is set when the timer interrupt is inhibited using a DI command or IPTM flag as shown at point (1) of Fig. 13-8.

In this case, the interrupt request can be canceled by writing "0" in the IRQTM flag. Writing "1" in the IRQTM flag causes the same result as if the interrupt request is issued. When the timer interrupt is accepted, a stack of one level is used. The bank register (BANK, address 79H) and index enable flag (IXE, bit b_0 of address 7FH) information are saved automatically at that time.

A dedicated RET1 command is used for the return from an interrupt processing routine.

For more details, see Section **5, "Stack,"** and Section **12, "Interrupt"**.

A timer interrupt example and error are described in Sections **13.5.1** and **13.5.2.** For more information on the relation between the timer interrupt and other interrupts (INT_0 pin, INT_1 pin, serial interface 1, and frequency counter), see Section **12, "Interrupt"**.

13.5.1 Timer Using Timer Interrupt

A timer interrupt example is shown below.

Example:

```
        BR    AAA          ; The program branches into AAA.
TIMER:                     ; The program address 0003H
        ADD   M1, #0001B   ; 1 is added to M1.
        SKT1  CY           ; The CY flag is tested.
        BR    BBB          ; Returned if no carry appears.
        [ Process A ]
BBB:
        EI
        RETI
AAA:
        INITFLG TMMD3, NOT TMMD2, NOT TMMD1, NOT TMMD0
                           ; Built-in macroinstruction
                           ; The timer interrupt pulse is set to 5 ms.
        MOV   M1, #0000B   ; The M1 information is cleared to "0".
        SET1  IPTM         ; The timer interrupt is permitted.
        EI                 ; All interrupts are permitted.
LOOP:
        [ Process B ]
        BR    LOOP
```

In the above program, process A is executed every 80 ms.

Notice that the system automatically enters the DI state when an interrupt is accepted and that the IRQTM flag is set (1) even if the system is in DI state.

When process A exceeds 5 ms, an interrupt is immediately accepted even if the system is returned using an RETI command. At that time, process B cannot be executed.

13.5.2 Timer Interrupt Error

As described in Section **13.5,** the interrupt is accepted every time a timer interrupt pulse falls if the timer interrupt is permitted.

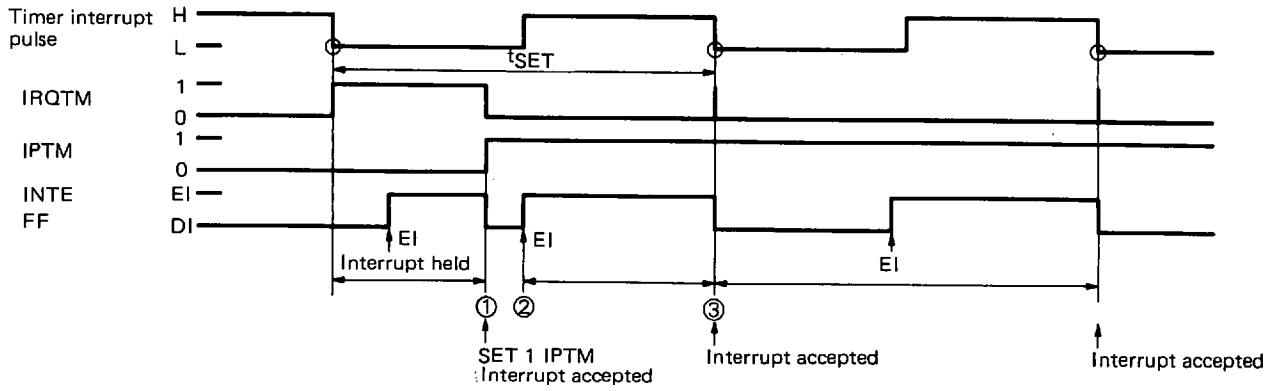
Therefore, the timer interrupt error occurs in the following cases:

(1) The first interrupt is accepted while timer interrupt is permitted.
(2) The first interrupt is accepted while timer interrupt pulse time is being altered.
(3) Data is written in the IRQTM flag.

Possible operating errors are shown in Fig. 13-9.

Fig. 13-9 Timer interrupt error (1/2)

(a) Timer interrupt is permitted.

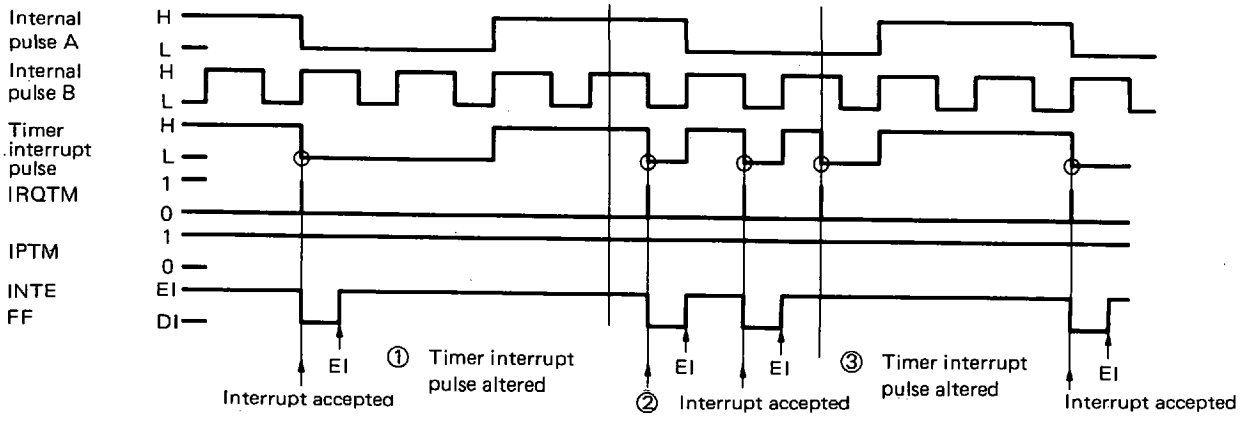


When the IPTM flag is set and the timer interrupt is permitted at point ①, an interrupt is accepted immediately. A possible error at that time is $-t_{SET}$.

An interrupt is established at the trailing edge of a timer interrupt pulse at point ③ when it is permitted at point ② using an EI command. A possible error at that time is as follows:

$-t_{SET} < \text{error} < 0$

(b) Timer interrupt pulse is selected

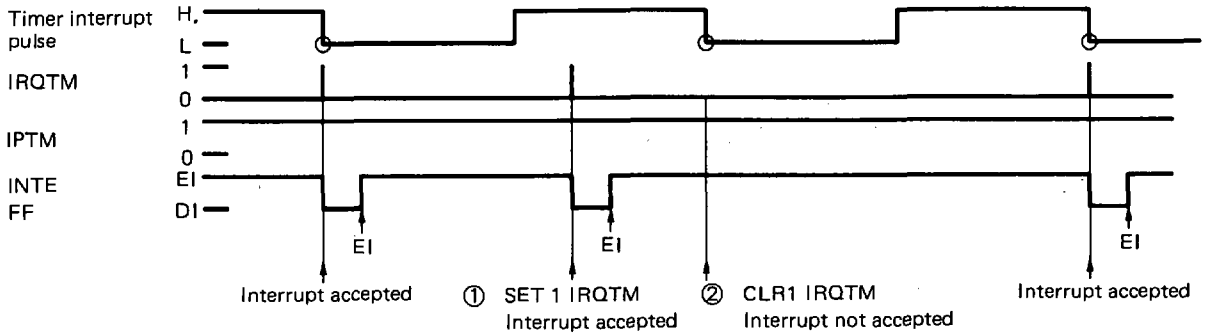


The timer interrupt pulse does not fall even if it is altered to B at point ①, so the interrupt is accepted at next point ②.

The timer interrupt pulse falls when it is altered to A at point ③, so the interrupt is accepted immediately.

Fig. 13-9 Timer interrupt error (2/2)

(c) IRQTM flag is activated.



An interrupt is accepted immediately when the IRQTM flag is set at point ①.

An interrupt is not accepted when the IRQTM flag reset overlaps with the fall of the timer interrupt pulse.

13.6 CAUTIONS DURING TIMER INTERRUPT

The timer interrupt must be processed within a fixed time when a program (e.g., timer program) for which the timer operates at all times is created after the supply voltage is supplied (power on reset) during the timer interrupt.

A timer interrupt example is shown below.

Example:

```
        BR    AAA          ; The program branches into AAA after reset.
TIMER:                     ; The program address 0003H
        ADD   M1, #0100B   ; 0100B is added to the M1 information.
        SKT1  CY           ; The timer is processed if a carry appears.
        BR    EI_RETI
  ; ①
        Timer processing
EI_RETI:
        EI
        RETI
AAA:
        INITFLG NOT TMMD3, NOT TMMD2, NOT TMMD1, NOT TMMD0
                           ; Built-in macroinstruction
                           ; The timer interrupt time is set to 250 ms, and the timer carry flip-flop
                           ; setting time to 100 ms.
        SET1  IPTM         ; Built-in macroinstruction
                           ; The timer interrupt is permitted.
        EI
        Process A
        BR    AAA
```

In the example above, timer processing ① is executed every second while executing process A.

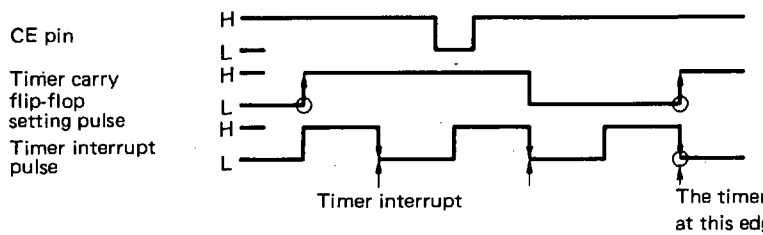
When the CE pin is changed from low to high as shown in Fig. 13-10 (a), a CE pulse is reset in synchronization with the rising of a timer carry flip-flop setting pulse. If the timer interrupt request issue overlaps with the timer carry flip-flop setting, the CE reset has priority. The timer interrupt request (IRQTM flag) is reset when the CE pulse is reset. Therefore, one timer cycle is skipped.

To prevent the timer interrupt from being skipped, a delay is provided at the trailing edges of timer carry flip-flop setting and timer interrupt pulses as shown in Fig. 13-10 (b). As a result, no timer interrupt is skipped when a CE pulse is reset by executing the timer processing within 10 ms.

Timer carry flip-flop and timer interrupt time setting pulses of 4 Hz (250 ms), 10 Hz (100 ms), 200 Hz (5 ms), and 1 kHz (1 ms) can be set independently. This enables the time difference shown in Fig. 13-11 and Table 13-1 to be set. The timer interrupt must be processed within the delay time of a pulse shown in Fig. 13-11 when it is required during the CE reset.

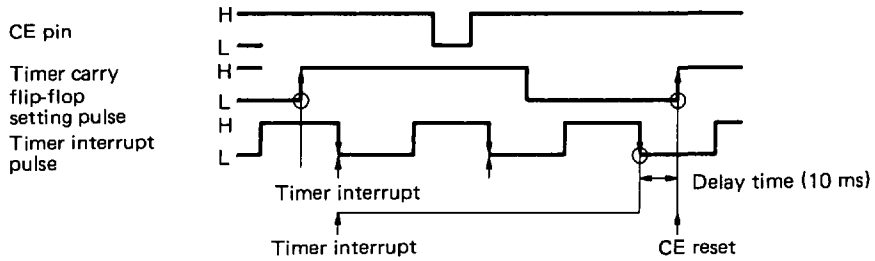
Fig. 13-10 Timing chart

(a)



The timer carry flip-flop setting pulse rises at this edge, so a CE pulse is reset and one timer interrupt is skipped.

(b)



A 10 ms delay occurs between the trailing edge of a timer interrupt pulse and the leading edge of a timer carry flip-flop setting pulse. The timer can thus be processed normally during the CE reset if the timer interrupt is processed within 10 ms.

Fig. 13-11 Time difference between timer carry flip-flop setting pulse and timer interrupt pulse

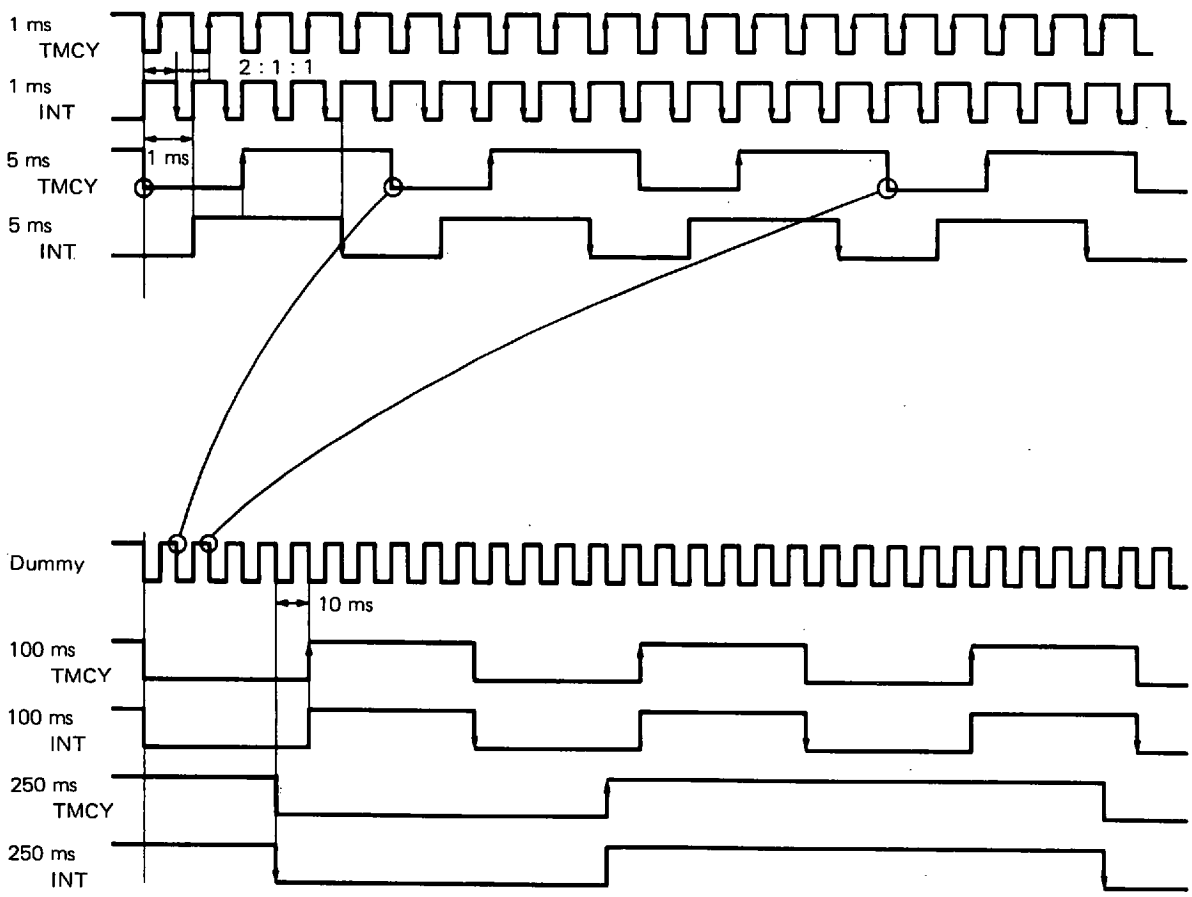
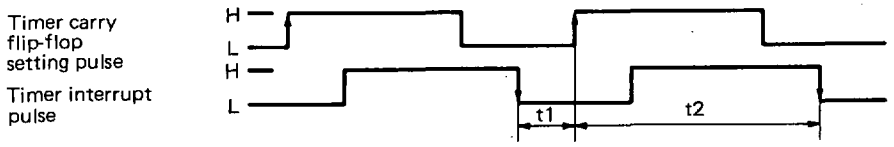


Table 13-1 Time difference between trailing edge of timer carry flip-flop pulse and leading edge of timer interrupt pulse

Internal pulse		Minimum time difference (See the figure below.)	
Timer carry	Timer interrupt	t1	t2
1 ms	1 ms	666 μs	333 μs
1 ms	5 ms	333 μs	666 μs
1 ms	100 ms	333 μs	666 μs
1 ms	250 ms	333 μs	666 μs
5 ms	1 ms	333 μs	666 μs
5 ms	5 ms	3 ms	2 ms
5 ms	100 ms	2 ms	3 ms
5 ms	250 ms	2 ms	3 ms
100 ms	1 ms	333 μs	666 μs
100 ms	5 ms	1 ms	4 ms
100 ms	100 ms	50 ms	50 ms
100 ms	250 ms	10 ms	40 ms
250 ms	1 ms	333 μs	666 μs
250 ms	5 ms	1 ms	4 ms
250 ms	100 ms	40 ms	10 ms
250 ms	250 ms	100 ms	150 ms



14. STANDBY

The standby function is used to reduce the power consumption of a backup device.

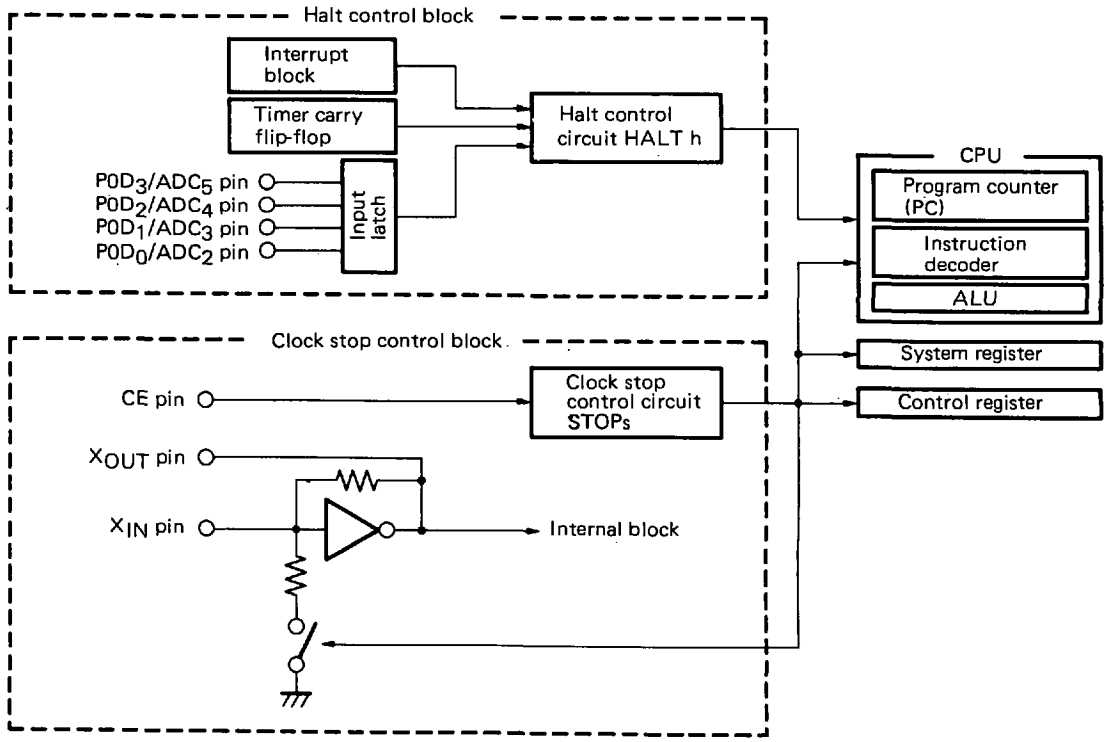
14.1 STANDBY BLOCK CONFIGURATION

Fig. 14-1 shows the standby block configuration. As shown in Fig. 14-1, the standby block consists of a halt control block and clock stop control block.

The halt control block consists of a halt control circuit, an interrupt control block, a timer carry flip-flop, and key input pins ($P0D_0/ADC_2$ pin (pin 78) to $P0D_3/ADC_5$ pin (pin 75)), controlling the operation of the CPU (consisting of program counter, instruction decoder, and ALU block).

The clock stop control block controls a 4.5 MHz crystal oscillator, CPU, system register, and control register using a clock stop control circuit.

Fig. 14-1 Standby block configuration



14.2 STANDBY FUNCTIONS

The standby function reduces the power consumption of the device by stopping part or all of the device operation.

The standby function is divided into halt and clock stop functions. The halt function reduces the power consumption of the device by stopping the CPU operation using a dedicated HALT h command. The clock stop function reduces the power consumption of the device by stopping the 4.5 MHz oscillator using a dedicated STOP s command.

In addition to the halt and clock stop functions, the device operating mode is also set using a CE pin. The CE pin is used to control the PLL frequency synthesizer and to reset the device. This function is defined to be one of the standby functions in controlling the PLL frequency synthesizer.

The device operating mode control using the CE pin is described in Section **14.3.** The halt and clock stop functions are described Sections **14.4** and **14.5.**

14.3 DEVICE OPERATING MODE USING CE PIN

The CE pin controls the functions below in the input level and at the leading edge of an external input signal.

(1) PLL frequency synthesizer operation

(2) Clock stop command validity

(3) Device reset

Steps (1), (2), and (3) above are described below.

14.3.1 PLL Frequency Synthesizer Operation Control

The PLL frequency synthesizer can be activated only when the CE pin is high. The PLL frequency synthesizer automatically enters the PLL disable mode when the CE pin is low. In the PLL disable mode, VCOH and VCOL pins are pulled down and EO_0 and EO_1 pins enter the floating state. An internal operational amplifier is also disabled in the PLL disable mode. At that time, the LPF_{IN} pin is pulled up internally, and the LPF_{OUT} pin (N-channel open drain output) is turned off. The PLL frequency synthesizer can be disabled using a program even if the CE pin is high.

14.3.2 Clock Stop Command Validity Control

The clock stop command (i.e., STOP s command) is valid only when the CE pin is low. The STOP s command executed when the CE pin is high is processed as a nonoperation command (NOP).

14.3.3 Device Reset

A device can be reset (CE reset) when the CE pin is changed from low to high. Power on reset when supply voltage V_{DD} is supplied is possible in addition to the CE reset.

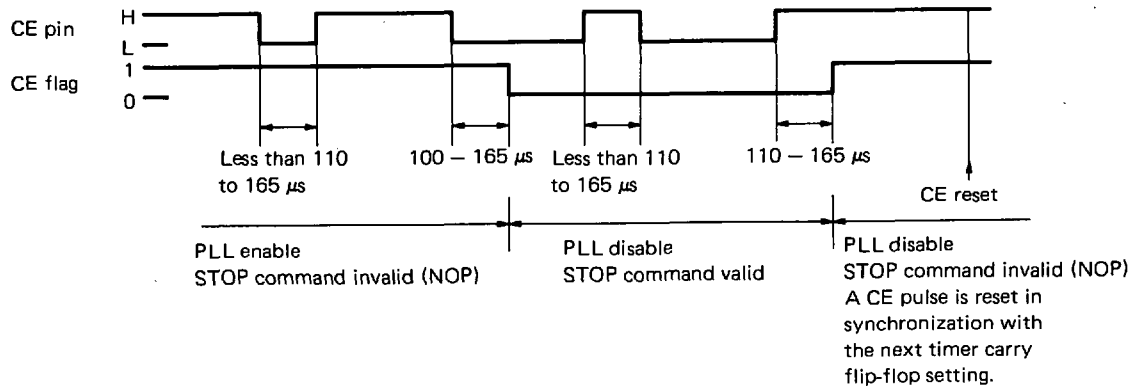
For more details, see Section **15, "Reset"**.

14.3.4 Signal Input to CE Pin

The CE pin does not accept a low or high signal of less than 110 to 165 μs to prevent malfunction due to noise. The input level of a signal that is input to the CE pin can be detected using a CE flag (bit b_0 of address 07H) of the control register.

Fig. 14-2 shows the relation between the input signal and CE flag.

Fig. 14-2 Relation between input signal and CE flag



14.3.5 CE Pin Level Judge Register Configuration and Functions

The CE pin level judge register detects the input signal level at a CE pin.

The configuration and functions are shown below.

Name	Flag symbol				Address	Read/Write
	b_3	b_2	b_1	b_0		
CE pin level judge register (CEJDG)	0	0	0	CE	07H	R

Bit	Function
CE	Detects the level of a signal that is input to the CE pin.
0	A low signal is input.
1	A high signal is input.
0	Fixed at "0".

		b_3	b_2	b_1	b_0
Reset	Power on	0	0	0	–
	Clock stop	↓	↓	↓	–
	CE	↓	↓	↓	–

The CE flag does not change when set to a low or high level for less than 110 to 165 μs.

14.4 HALT FUNCTIONS

The halt function stops the CPU operating clock by executing a HALT h command. A program stops when the HALT h command is executed. The program remains stopped until the halt mode is canceled. Therefore, the power consumption of a device that is in halt mode is reduced proportionally to the CPU operating current. The halt mode is canceled by a timer carry flip-flop, interrupt, and key entry. The cancel conditions for the timer carry flip-flop, interrupt, and key entry are specified by operand h of the HALT h command. The HALT h command is valid irrespective of the input level at a CE pin.

The halt mode and halt cancel conditions are described in Sections **14.4.1** through **14.4.6.**

14.4.1 Halt Mode

All CPU operations stop in halt mode. The program execution is stopped using a HALT h command. However, the peripheral hardware continues the operation that was set before HALT h command execution.

For more information on the peripheral hardware operation, see Section **14.6, "Device Operation during Halt and Clock Stop".**

14.4.2 Halt Cancel Conditions

Fig. 14-3 shows the halt cancel conditions.

As shown in Fig. 14-3, the halt cancel conditions are set using four-bit data that is specified by operand h of a HALT h command. The halt mode is canceled when the conditions specified by "1" for operand h is satisfied. When the halt mode is canceled, the command following the HALT h command is executed. When two or more cancel conditions are set at that time, the halt mode is canceled if only one condition is satisfied.

When a device is reset (power on reset or CE reset), the halt mode is canceled to execute each reset.

No cancel conditions can be set when 0000B is set to halt cancel condition "h". When the device is reset (power on reset or CE reset), the halt mode is canceled.

The halt cancel conditions of a timer carry flip-flop, interrupt, and key entry are described in Sections **14.4.3** through **14.4.5.** An example when two or more cancel conditions are set is described in Section **14.4.6.**

Fig. 14-3 Halt cancel conditions

HALT h (4-bit)

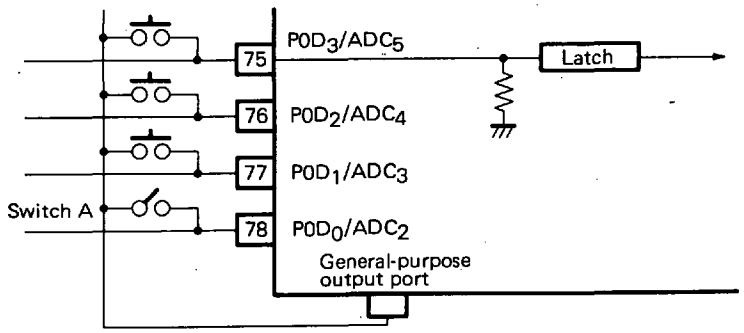
Operand bit				
b_3	b_2	b_1	b_0	
				Sets the cancel conditions of the halt mode.
				Canceled when a high signal is input to key input pins ($P0D_3/ADC_5$ to $P0D_0/ADC_2$).
				Canceled when the timer carry flip-flop is set (1).
				Undefined (fixed at "0").
				Canceled when interrupts (INT_0 pin, INT_1 pin, timer, serial interface 1, and frequency counter) are accepted.
0				Not canceled even if the conditions are satisfied.
1				Canceled when the conditions are satisfied.

14.4.3 Halt Cancel Using Key Entry

The halt cancel conditions entered using a key are set by a HALT 0001B command. When the halt cancel conditions are set, the halt mode is canceled if a high signal is input to one of pins $P0D_0/ADC_2$ through $P0D_3/ADC_5$.

When the key source signal is used in common with an LCD segment signal and the key input pin is used as an A/D converter, pay attention to Items (1) through (4) below.

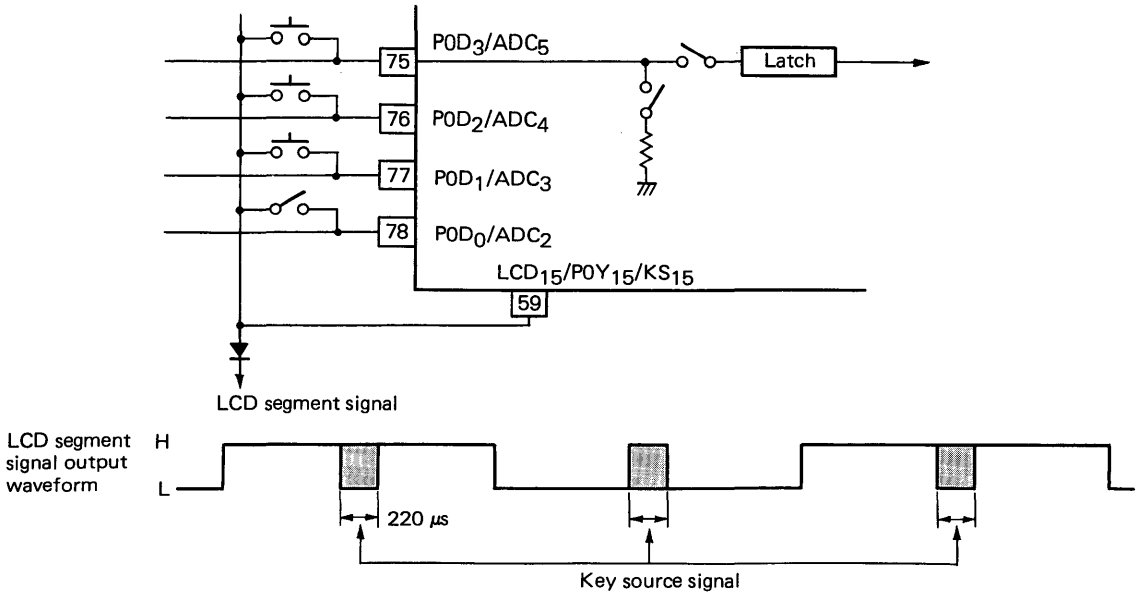
(1) General-purpose output port used as key source signal



A HALT 0001B command is executed after the general-purpose output port for a key source signal is set high. When switch A (i.e., alternating switch) shown in the figure above is used at that time, a high signal is fed to the $P0D_0/ADC_2$ pin at all times while switch A is closed. Therefore, the halt mode is canceled immediately. Pay attention when the alternating switch is used.

When the general-purpose output port is used as a key source signal, the KSEN flag (bit b_1 of address 10H) of a control register is set to "0". Pins $P0D_0/ADC_2$ through $P0D_3/ADC_5$ are automatically pulled down at that time.

(2) LCD segment signal output used in common with key source signal output



A HALT 0001B command is executed after the key source signal output data is set. When the key source signal output data is "0", the halt mode is not canceled even if a high LCD segment signal is input.

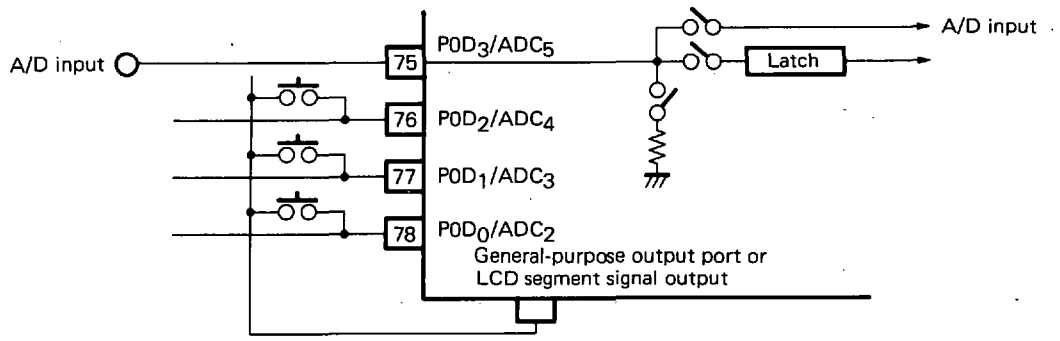
When the LCD segment signal output is used in common with the key source signal output, the KSEN flag (bit b_1 of address 10H) of a register file is set (1).

The key source signal data (key source output pin) is set via the data buffer using a key source data register (KSR).

An internal key latch circuit latches data when the LCD segment signal and key source signal outputs are used in common while a key source signal is output. The latch circuit is also isolated from the external device while an LCD segment signal is output.

The internal pull-down resistor is turned on only when a key source signal is output.

(3) Pins $P0D_0/ADC_2$ through $P0D_3/ADC_5$ used as A/D converter

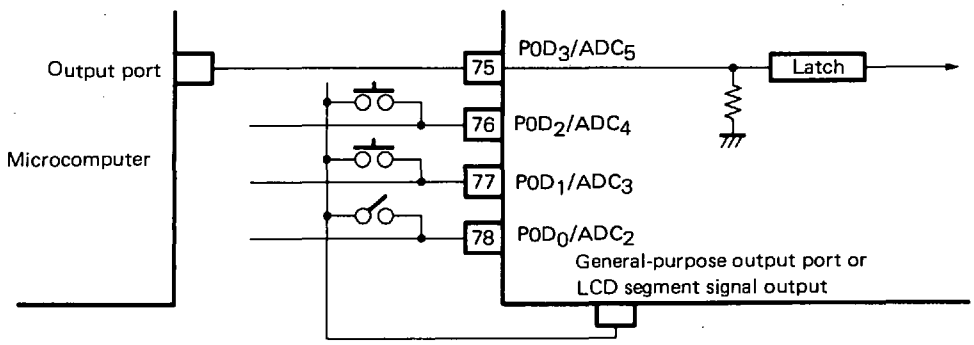


When pins $P0D_0/ADC_2$ through $P0D_3/ADC_5$ are selected as an A/D converter, the selected pin (only one pin can be selected at the same time) is isolated from the input latch and connected to the internal A/D converter input. The latch circuit is held high if a high signal is input to the pin that was selected as the A/D converter.

When a HALT 0001B command is executed while the latch circuit remains high, the halt mode is canceled immediately because the input latch is high.

Do not execute a HALT 0001B command while the latch circuit remains high.

(4) Others



Pins $P0D_0/ADC_2$ through $P0D_3/ADC_5$ can also be used as a general-purpose input port with pull-down resistors. The halt mode can also be canceled using other microcomputers as shown in the figure above.

14.4.4 Halt Cancel Using Timer Carry Flip-Flop

The halt mode cancel using a timer carry flip-flop is set by a HALT 0010B command. When the halt cancel is set, the halt mode is canceled every time the timer carry flip-flop is set (1).

As described in Section **13, "Timer"**, the timer carry flip-flop corresponds to the TMCY flag (bit b_0 of address 17H) of a control register at a ratio of 1 to 1 and is set (1) for every fixed time (1 ms, 5 ms, 100 ms, and 250 ms). Therefore, the halt mode can be canceled for every fixed time.

A halt cancel example is shown below.

Example:

```
HLTTMR  DAT  0010B  ;  Symbol definition
INITFLG NOT TMMD3, NOT TMMD2, NOT TMMD1, NOT TMMD0
                   ;  Built-in macroinstruction
                   ;  The timer carry flip-flop setting time is set to 250 ms.
LOOP:
        HALT   HLTTMR     ;  A timer carry flip-flop is set as the halt cancel condition.
        SKT1   TMCY       ;  Built-in macroinstruction
        BR     LOOP       ;  The program branches into LOOP if a TMCY flag is not set.
        ADD    M1, #0100B ;  0100B is added to the M1 information.
        SKT1   CY         ;  Built-in macroinstruction
        BR     LOOP       ;  Process A is executed if a carry occurs.
        ┌──────────┐
        │ Process A │
        └──────────┘
        BR     LOOP
```

In the above example, the halt mode is canceled every 250 ms and process A is executed every second.

14.4.5 Halt Cancel Using Interrupt

The halt mode cancel using an interrupt is set by a HALT 1000B command. When the halt mode cancel is set, the halt mode is canceled every time the interrupt is accepted.

As described in Section **12, "Interrupt"**, the interrupt has five factors, INT_0 pin, INT_1 pin, timer, serial interface 1, and frequency counter.

The interrupt factor used to cancel the halt mode must be specified in advance using a program.

An interrupt can be accepted when all interrupts (EI command) and each interrupt (interrupt permission flag is set) are permitted in addition to the interrupt request issued from each interrupt source. Consequently, an interrupt cannot be accepted when it is not permitted even if an interrupt request is issued. The halt mode is not canceled at that time either.

When the halt mode is canceled during the interrupt acceptance, the program flow changes to the vector address of each interrupt. When an RETI command is executed after interrupt processing, the program flow is returned to the command next to HALT.

A halt cancel example is shown on the next page.

Example:

```
        HLTINT DAT 1000B    ; Symbol definition
START:                      ; The program address 0000H
        BR     MAIN         ;
        NOP
        NOP
INTTM:                      ; Timer interrupt vector address (0003H)
        BR     INTTIMER     ; The program branches into timer interrupt INTTIMER.
        NOP
INT0:                       ; INT0 pin interrupt vector address. (0005H)
        [ Process A ]       ; INT0 pin interrupt
        EI
        RETI
INTTIMER:
        [ Process B ]       ; Timer interrupt
        EI
        RETI
MAIN:
        SET2 IPTM,   IP0    ; Built-in macroinstruction
        SET2 TMMD3, TMMD2   ; Built-in macroinstruction
LOOP:                       ; The timer interrupt time interval is set to 1 ms.
        [ Process C ]       ; Main routine processing
        EI                  ; All interrupts are permitted.
        HALT  HLTINT        ; The halt cancel is set using an interrupt.
  ; ①
        BR     LOOP
```

In the above program, the halt mode is canceled when a timer interrupt is accepted. Process B is then executed. Process A is executed when an INT_0 pin interrupt is accepted.

Process C is executed every time the halt mode is canceled.

When the INT_0 pin and timer interrupt requests are issued at the same time in halt mode, process A of the INT_0 pin with a higher hardware priority level is executed.

A program is returned to BR LOOP command ① when an RETI command is executed after process A is executed. However, the BR LOOP command is not executed, and the timer interrupt is accepted. The BR LOOP command is executed when an RETI command is executed after process B (i.e., timer interrupt) execution.

14.4.6 Several Cancel Conditions Set at the Same Time

When two or more halt cancel conditions are set at the same time, the halt mode is canceled if only one requirement is satisfied. An example when several cancel conditions are satisfied at the same time is shown below.

Example 1:

```
        HLTINT  DAT 1000B
        HLTTMR  DAT 0010B
        HLTKEY  DAT 0001B
START:
        BR    START
        NOP
        NOP
        NOP
        NOP
INT0:                          ; INT0 pin interrupt vector address (0005H)
        Process A              ; INT0 pin interrupt
        EI
        RETI
TMRUP:                         ; Timer carry flip-flop processing
        Process B
        RET
KEYDEC:                        ; Key entry
        Process C
        RET
START:
        MOVT  DBF, @AR         ; Key source output data (see the table) is set in the key source
                               ; data register (KSR).
        PUT   KSR, DBF
        SET2  KSEN, LCDEN      ; Built-in macroinstruction
                               ; LCD segment signal and key source signal outputs are used in common.
        SET2 TMMD1, TMMD0      ; Built-in macroinstruction
                               ; The timer carry flip-flop setting time is set to 1 ms.
        SET1  IP0              ; Built-in macroinstruction
                               ; An INT0 pin interrupt is permitted.
        EI
LOOP:
        HALT HLTINT OR HLTTMR OR HLTKEY
                               ; An interrupt, timer carry flip-flop, and key entry are set as halt
                               ; cancel conditions.
        SKF1  TMCY             ; Built-in macroinstruction
                               ; TMCY flag detection
        CALL  TMRUP            ; Timer carry flip-flop processing during setting (1)
        SKT1  KEYJ             ; Built-in macroinstruction
                               ; Key entry latch detection
        CALL  KEYDEC           ; Key entry during latch
        BR    LOOP
```

In example 1 above, three halt cancel conditions, an INT_0 pin interrupt, 1 ms timer carry flip-flop, and key entry are set.

If the halt mode is canceled using an interrupt, a vector address is detected. If it is canceled using a timer carry flip-flop, a TMCY flag is detected. If it is canceled using a key entry, a KEYJ flag is detected.

Pay attention to the following when using two or more cancel conditions:

(1) All cancel conditions that were set are detected when the halt mode is canceled.
(2) The cancel conditions are detected in accordance with the priority level.

For steps (1) and (2), pay attention when the START or later program in example 1 is as shown in example 2 below.

Example 2:

```
START:
        SET4  PIC3, PIC2, PIC1, PIC0 ; The general-purpose output port is used as a key source signal.
        SET2  TMMD1, TMMD0
        SET1  IP0
        EI
LOOP:
        HALT  HLTINT OR HLTTMR OR HLTKEY
        SKF4  P0D3, P0D2, P0D1, P0D0 ; The key entry is detected.
        BR    KEYDEC
        SKF1  TMCY
        CALL  TMRUP
        BR    LOOP
KEYDEC:                              ; Key entry
        [Process C]
        BR    LOOP
```

Assume that the timer carry flip-flop is set (1) immediately after the halt mode is canceled by entering the key in the program of example 2 above. The program then executes a HALT command again after the key entry is executed. The timer carry flip-flop remains set when the HALT command is executed, so the halt mode is canceled immediately.

However, a high-level pulse of 100 ms is usually input during the key entry, so the program branches into the key entry. The timer carry flip-flop thus cannot be detected precisely. Do not create the program shown in example 2 if the program has a high timer priority level and uses a timer carry flip-flop.

14.5 CLOCK STOP FUNCTION

The clock stop function stops the 4.5 MHz crystal oscillator by executing a STOP s command (clock stop state). The device power consumption is therefore reduced to 15 μA (max). For more information on the power consumption, see Section **14.7, "Power Consumption in Halt and Clock Stop Modes"**.

"0000B" is specified for operand "s" of a STOP s command. The STOP s command is valid when the CE pin (pin 13) is low. The STOP s command cannot be executed (nonoperation command (NOP)) while the CE pin is high. The STOP s command must be executed only when the CE pin is low.

The clock stop mode is canceled when the CE pin is changed from low to high (CE reset).

The clock stop mode, clock stop mode cancel, and cautions when using clock stop command are described in Sections **14.5.1** through **14.5.3.**

14.5.1 Clock Stop Mode

In clock stop mode, the crystal oscillator is stopped. Consequently, all CPU and peripheral hardware operations also stop.

For more information on the CPU and peripheral hardware operation, see Section **14.6, "Device Operation in Halt and Clock Stop Modes"**.

In clock stop mode, the power failure detector circuit does not operate even if supply voltage V_{DD} is decreased to 2.2 V. Low-voltage data memory can thus be backed up. For more information on the power failure detector circuit, see Section **15, "Reset"**.

14.5.2 Clock Stop Mode Cancel

The clock stop mode is canceled when the CE pin is changed from low to high (CE reset) or when supply voltage V_{DD} is increased to 4.5 V after it is first decreased to 2.2 V (power on reset).

Figs. 14-4 and 14-5 show the clock stop cancellation during the CE reset and power on reset. The power failure detector circuit operates when the clock stop mode is canceled during the power on reset.

For more information on the power on reset, see Section **15.4, "Power On Reset"**.

Fig. 14-4 Clock stop cancel during CE reset

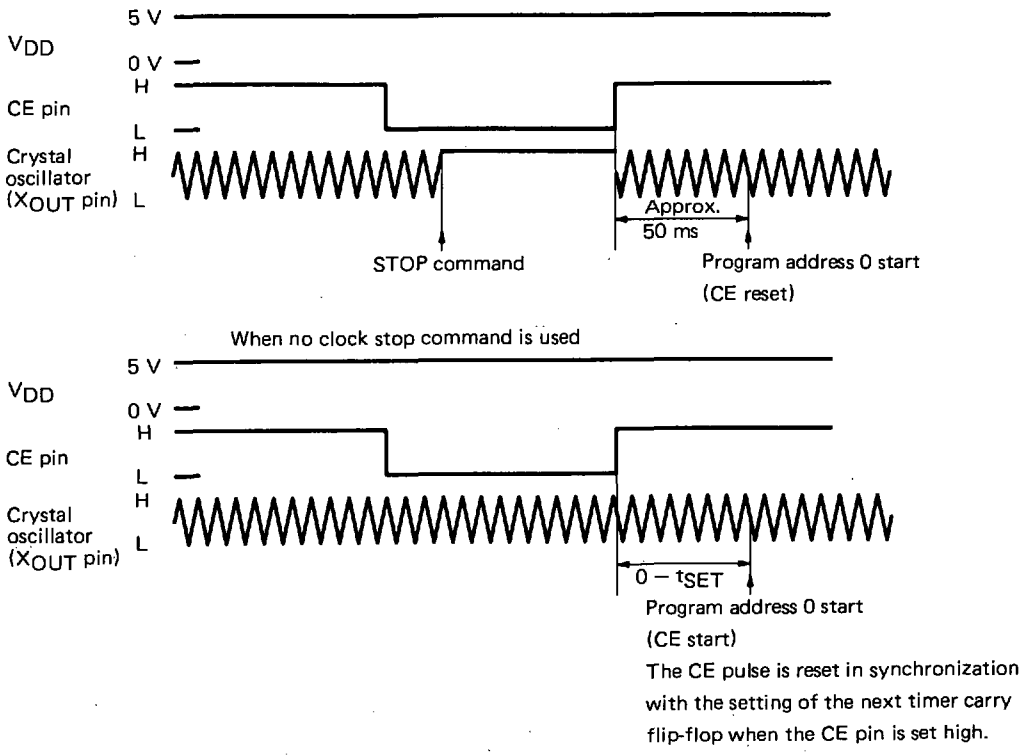
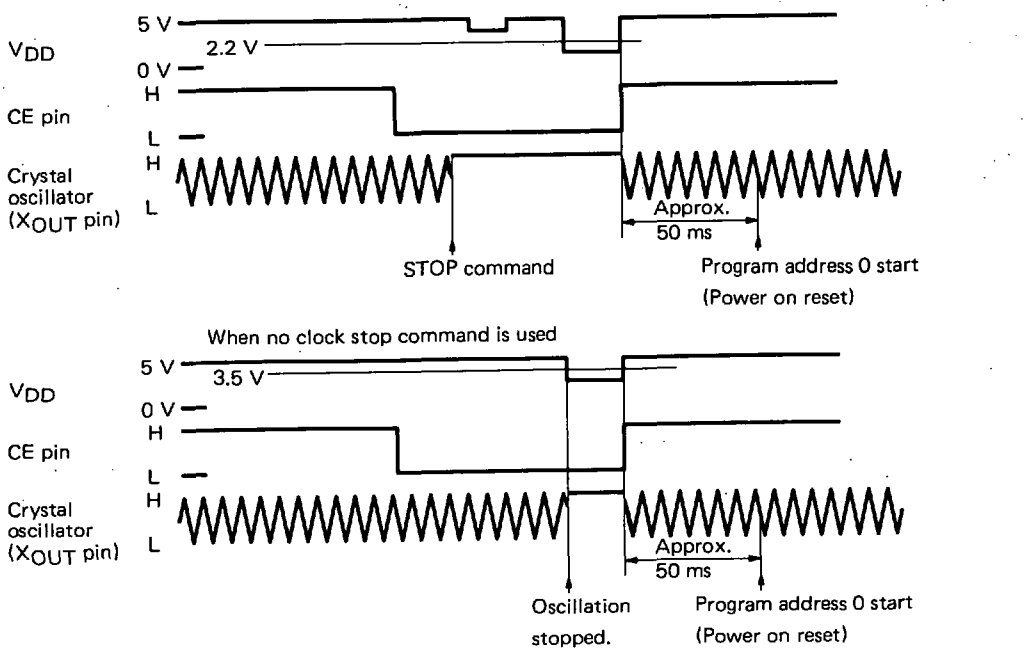


Fig. 14-5 Clock stop cancel during power on reset



14.5.3 Cautions when Using Clock Stop Command

A clock stop command (STOP s command) is valid when the CE pin is low. Therefore, the processing when the CE pin is high must be programmed in advance.

A program example is shown below.

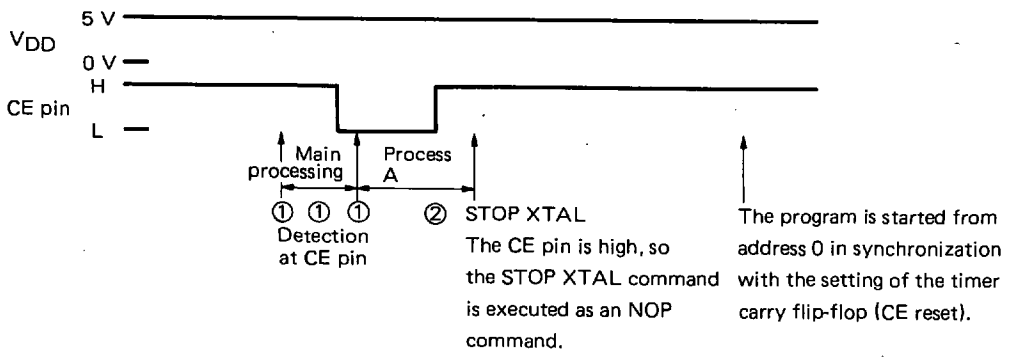
Example:

```
        XTAL  DAT   0000B  ; Symbol definition for clock stop conditions
CEJDG:
; ①
        SKF1  CE           ; Built-in macroinstruction
                           ; The input level at the CE pin is detected.
        BR    MAIN         ; The program branches into the main processing when the CE pin is high.
        [Process A]        ; Processing for CE = Low
; ②
        STOP  XTA          ; Clock stop
; ③
        BR    $-1
MAIN:
        [Main processing]
        BR    CEJDG
```

In the above example, the input level at the CE pin is detected in parameter ①. If the CE pin is low, a STOP XTAL clock stop command in item ② is executed after process A is executed. The STOP XTAL command is executed as a nonoperation command (NOP) when the CE pin is set high during the STOP XTAL command execution in parameter ② as shown in the figure below.

If there is no BR $-1 branch command in parameter ③, the program branches into the main processing. This may cause a malfunction. Consequently, a branch command must be inserted in the program as shown in parameter ③ or a program must be created in advance so that no malfunction occurs after the program branches into the main processing.

When a branch command is used as shown in parameter ③, the CE pin remains high, but a CE pulse is reset in synchronization with the setting of the next timer carry flip-flop.



14.6 DEVICE OPERATION IN HALT AND CLOCK STOP MODES

Table 14-1 shows the CPU and peripheral hardware operation in halt and clock stop modes.

In halt mode, the command execution stops, but all peripheral hardware operates normally as shown in Table 14-1. In clock stop mode, all peripheral hardware operations stop.

A control register that controls the peripheral hardware operation operates (is not initialized) as usual in halt mode. The control register is initialized at a specified value in clock stop mode (during the STOP s command execution). In other words, the peripheral hardware continues the operation that is set in the control register in halt mode, operating in accordance with the control register that is initialized at a specified value in clock stop mode.

For more information on the initialized control register value, see Section **10, "Register File (RF)".**

An operation example is shown below.

Example:

The $P0A_3$/SDA and $P0A_2$/SCL pins of port 0A are set to the output port, and the $P0A_1/\overline{SCK_1}$ and $P0A_0/SO_1$ pins are used as a serial interface.

```
        HLTINT  DAT   1000B  ;  Symbol definition
        XTAL    DAT   0000B  ;
        INITFLG P0ABIO3, P0ABIO2, P0ABIO1, P0ABIO0
                               ;  Built-in macroinstruction
;  ①
        SET2    P0A3,   P0A2
        INITFLG SI01CH, NOT SB, SIO1MS, SIO1TX
                               ;
        SET2    SI01CK1, SI01CK0
;  ②
        SET2    SI01IMD1, SI01IMD0
        CLR1    IRQSI01
        SET1    IPSI01
        EI
;  ③
        SET1    SI01NWT
;  ④
        HALT    HLTINT
;  ⑤
        STOP    XTAL
```

In the above example, a high signal is output from the $P0A_3$ and $P0A_2$ pins in parameter ①, the conditions of serial interface 1 are set in parameter ②, and the serial communication is started in parameter ③.

When a HALT command in parameter ④ is executed, the serial communication is continued and the halt mode is canceled when a serial interface 1 interrrupt is accepted.

When a STOP command in parameter ⑤ is executed instead of the HALT command, all flags of the control registers that are set in parameters ①, ②, and ③ are initialized.

The serial communication is thus interrupted, and all pins of port 0A are set to the general-purpose input port.

Table 14-1 Device operation in halt and clock stop modes

Peripheral hardware	State			
	CE pin = high		CE pin = low	
	Halt mode	Clock stop mode	Halt mode	Clock stop mode
Program counter	Stops at the HALT command address.	STOP command is invalid (NOP).	Stops at the HALT command address.	Initialized at 0000H and stops.
System register	Held.		Held.	Initialized (see **Note**).
Peripheral register	Held.		Held.	Held.
Control register	Held		Held.	Initialized (see **Note**).
Timer	Operates normally.		Operates normally.	Stops.
PLL frequency synthesizer	Operates normally.		Disable (including internal operational amplifier)	Stops.
A/D converter	Operates normally.		Operates normally.	Stops.
D/A converter	Operates normally.		Operates normally.	Stops.
Serial interface	Operates normally.		Operates normally.	Stops.
Frequency counter	Operates normally.		Operates normally.	Stops.
LCD controller/driver	Operates normally.		Operates normally.	Stops.
Key source controller/ decoder	Operates normally.		Operates normally.	Stops.
General-purpose input/ output port	Operates normally.		Operates normally.	Input port
General-purpose input port	Operates normally.		Operates normally.	Input port
General-purpose output port	Operates normally.		Operates normally.	Held.

Note: For more details of the initialized value, see Section 9, **"System Register"** and Section 10, **"Register File"**.

14.7 POWER CONSUMPTION IN HALT AND CLOCK STOP MODES

14.7.1 Power Consumption in Halt Mode

Fig. 14-6 shows power consumption I_{DD} in halt mode.

The power consumption in (1), (2), (3), and (4) shown in Fig. 14-6 employs the programs below. As shown in Fig. 14-6, the less halt cancel count reduces the power consumption.

(1) Program 1

No HALT command is used.

Example:

```
NOP
BR   $-1
```

(2) Program 2

A 5 ms timer interrupt is set as the halt cancel condition. Twenty commands (about 90 μs) are executed every time the halt mode is canceled.

Example:

```
        HALTINT  DAT  1000B
        BR       LOOP
TMINT:                          ;  Address 0003H
        NOP ┐
        NOP │
         :  │  17 commands
         :  │
        NOP │
        EI  ┘
        RETI
LOOP:
        SET1   TMMD3
        CLR1   TMMD2
        SET1   IPTM
        EI
        HALT   HLTINT
        BR     $-1
```

(3) Program 3

A 100 ms timer interrupt is set as the halt cancel condition. Twenty commands are executed every time the halt mode is canceled.

Example:

```
        HALTINT  DAT  1000B
        BR       LOOP
TMINT:                          ;  Address 0003H
        NOP  ┐
        NOP  │
          :  │   17 commands
          :  │
        NOP  │
        EI   ┘
        RETI
LOOP:
        CLR2   TMMD3,  TMMD2
        SET1   IPTM
        EI
        HALT   HLTINT
        BR     $-1
```

(4) Program 4

Nothing is set as the halt cancel condition.

Example:

```
HLTNORLS  DAT  0000B
HALT  HLTNORLS
```

Power consumption I_{DD} shown in Fig. 14-6 is measured under the conditions described below.

- PLL disable (Low-pass filter amplifier disable)
- Frequency counter disable
- The sine wave with frequency f_{IN} of 4.5 MHz and input amplitude V_{IN} of V_{DD} from a reference signal generator is input to the X_{IN} pin.
- All pins that are set for output are open.
- All pins that are set to the input port are pulled down using a 47 kΩ resistor (not including LPF_{IN} and X_{IN} pins).

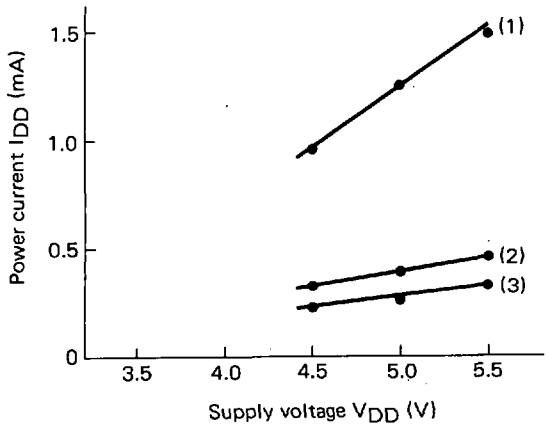
14.7.2 Power Consumption in Clock Stop Mode

Fig. 14-7 shows power consumption I_{DD} in clock stop mode. The power consumption shown in Fig. 14-7 is measured under the conditions described below.

- All pins that are set for output are open.
- All pins that are set for input are pulled down using a 47 kΩ resistor (not including LPF_{IN} and X_{IN} pins).
- A crystal oscillator is connected (the oscillation stops).

Fig. 14-6 Power consumption in halt mode (Reference)

(a) I_{DD} VS V_{DD} (T_a = 25 °C)



(b) I_{DD} VS T_a (V_{DD} = 5.5 V)

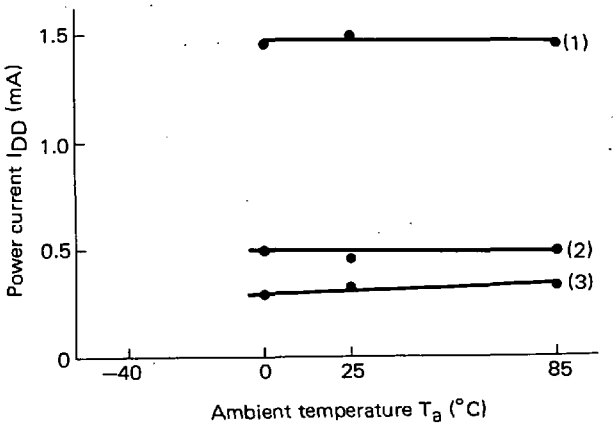
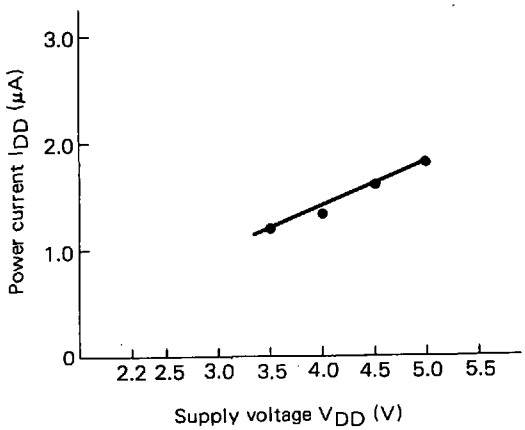
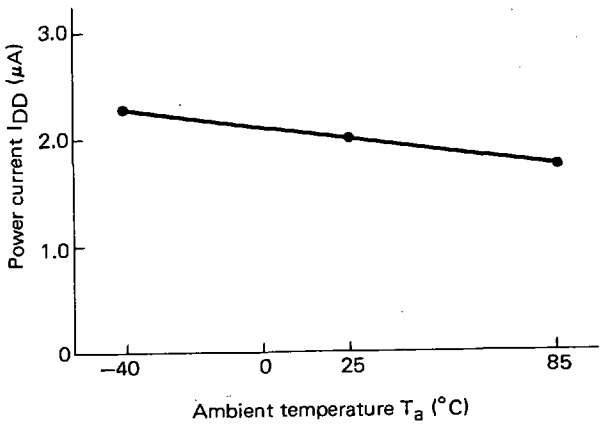


Fig. 14-7 Power consumption in clock stop mode (Reference)

(a) I_{DD} VS V_{DD} (T_a = 25 °C)



(b) I_{DD} VS T_a (V_{DD} = 5.5 V)



14.7.3 Cautions During Pin Processing in Halt and Clock Stop Modes

The halt function is used to reduce the power consumption when only a timer operates, and the clock stop function to reduce the power consumption when only data memory is held.

The power consumption must thus be reduced in halt and clock stop modes as much as possible.

Notice that the power consumption may significantly vary depending on the signal level at each pin as shown in Table 14-2.

Table 14-2 Pin state in halt and clock stop modes

Pin		Symbol	Pin state	
			Halt mode	Clock stop mode
General-purpose input/output port	Port 0A	$P0A_3$/SDA $P0A_2$/SCL $P0A_1/\overline{SCK}_1$ $P0A_0/SO_2$	The mode preceding the halt is held. (1) Ports are specified for output pins. The power consumption increases when the output pins are externally pulled down while a high signal is output or when they are externally pulled up while a low signal is output. Note to the N-channel open drain output (at $P0A_3$, $P0A_2$, and $P1B_3$ through $P1B_0$ pins). (2) Ports are specified for input pins (not including ports 1A and 1D). The power consumption increases owing to noise when the ports are in the floating state. (3) Port 0C ($P0D_3/ADC_5$ through $P0D_0/ADC_2$) The power consumption increases when the pins are pulled up externally because a pull-down resistor is incorporated. The pull-down resistor at the pin that is set to an A/D converter is turned off. (4) Port 1D ($P1D_3$/FMIFC through $P1D_0/ADC_0$) Port 1A ($P1A_3$/FCG through $P1A_0$) When the $P1D_3$/FMIFC and $P1D_2$/AMIFC pins are used as an IF counter, the internal amplifier operates and the power consumption increases. The IF counter is not automatically disabled even if the CE pin is set low. Therefore, initialize using a program as required. The power consumption does not increase owing to noise even if the pins at ports 1D and 1A are in the floating state when they are specified for the general-purpose input port.	The pins at all ports are specified for the general-purpose input port. All input ports other than port 0C ($P0C_3$ through $P0C_0$) do not increase the power consumption due to noise even if they are in the floating state. Port 0C ($P0C_3$ through $P0C_0$) must be pulled down or up externally to prevent the power consumption from increasing owing to noise. Port 0D ($P0D_3/ADC_5$ through $P0D_0/ADC_2$) is pulled down internally.
	Port 0B	$P0B_3/SI_1$ $P0B_2/\overline{SCK}_2$ $P0B_1/SO_2$ $P0B_0/SI_2$		
	Port 0C	$P0C_3$ $P0C_2$ $P0C_1$ $P0C_0$		
	Port 1A	$P1A_3$/FCG $P1A_2$ $P1A_1$ $P1A_0$		
General-purpose input port	Port 0D	$P0D_3/ADC_5$ $P0D_2/ADC_4$ $P0D_1/ADC_3$ $P0D_0/ADC_2$		
	Port 1D	$P1D_3$/FMIFC $P1D_2$/AMIFC $P1D_1/ADC_1$ $P1D_0/ADC_0$		
General-purpose output port	Port 1B	$P1B_3/PWM_2$ $P1B_2/PWM_1$ $P1B_1/PWM_0$ $P1B_0$/CGP		The pins are specified for the general-purpose output port. The output information is held. The power consumption increases when the pins are pulled down externally while a high signal is output or when they are pulled up while a low signal is output.
	Port 1C	$P1C_3$ $P1C_2$ $P1C_1$ $P1C_0$		
	Port 2A	$P2A_0$		
Interrupt		INT_1 INT_0	The power consumption increases owing to noise when the port is in the floating state.	

Pin	Symbol	Pin state	
		Halt mode	Clock stop mode
LCD segment	$LCD_{29}/P0F_3$ | $LCD_{26}/P0F_0$ $LCD_{25}/P0E_3$ | $LCD_{22}/P0E_0$ $LCD_{21}/P0X_5$ | $LCD_{16}/P0X_0$ $LCD_{15}/P0Y_{15}/KS_{15}$ | $LCD_0/P0Y_0/KS_0$	Pay the same attention as in the above general-purpose ports when this pin is used as a general-purpose output port. The transistor switch is always on when a key source signal is output. The power consumption increases via port 0D (internal pull-down resistor) when logical "1" is output as key source data.	All pins are specified for the LCD segment signal output, outputting a low signal (display off).
PLL frequency synthesizer	VCOL VCOH LPF_{IN} LPF_{OUT} V_{LPF} EO_0 EO_1	The power consumption increases during the PLL operation. When the PLL is disabled, each pin is as follows: VCOL, VCOH; Pulled down internally. LPF_{IN}; Pulled up internally. EO_0, EO_1; Floating The PLL is disabled automatically when the CE pin is set low.	The PLL is disabled. Each pin is as follows: VCOL, VCOH; Pulled down internally. LPF_{IN}; Pulled up internally. EO_0, EO_1; Floating
Crystal oscillator	X_{IN} X_{OUT}	The power consumption varies depending on the oscillation waveform of a crystal oscillator. A high oscillation amplitude decreases the power consumption. The oscillation amplitude varies according to a crystal oscillator or load capacitor.	The X_{IN} pin is pulled down internally, and the X_{OUT} pin outputs a high signal.

15. RESET

The reset function is used to initialize the device operation.

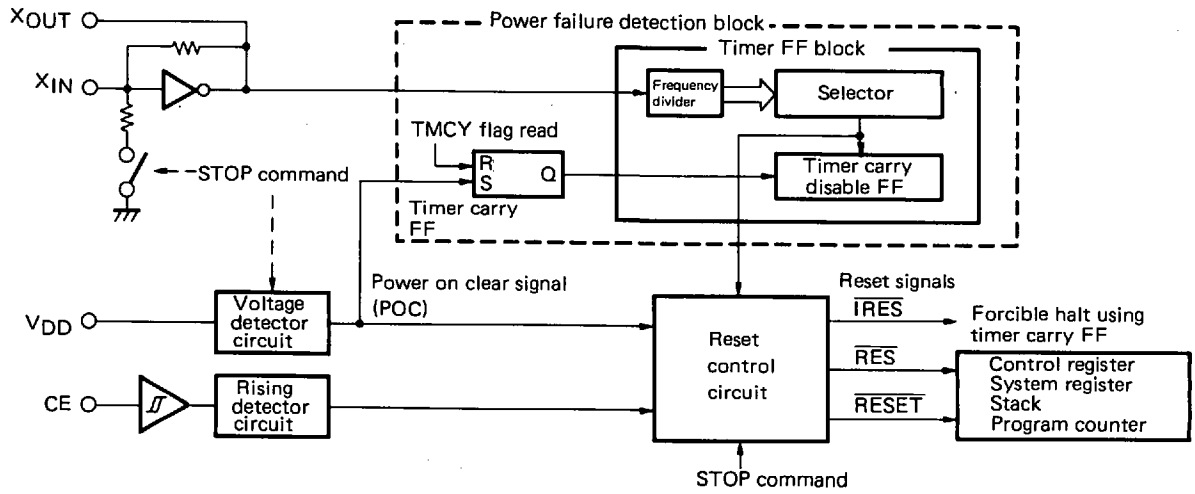
15.1 RESET BLOCK CONFIGURATION

Fig. 15-1 shows the reset block configuration.

The device reset is classified into a supply voltage V_{DD} on reset (power on reset or V_{DD} reset) and a CE pin reset (CE reset).

The power on reset block consists of a voltage detector circuit that detects the voltage input from the V_{DD} pin, a power failure detector circuit, and a reset control circuit. The CE reset block consists of a circuit that detects the rise of an input signal at the CE pin and a reset control circuit.

Fig. 15-1 Reset block configuration



15.2 RESET FUNCTION

A power on reset occurs when supply voltage V_{DD} rises from less than a fixed voltage, and a CE reset occurs when the CE pin rises from low to high.

The power on reset function is used to initialize a program counter, stack, system register, and control register and to execute the program from address 0000H. The CE reset function is used to partially initialize a program counter, stack, system register, and control register and to execute the program from address 000H.

The difference between the power on and CE resets lies in the operations of an initialized control register and power failure detector circuit (described in Section **15.6**).

The power on reset and CE reset are controlled using reset signals $\overline{IRES}$, $\overline{RES}$, and $\overline{RESET}$ that are output from the reset control circuit shown in Fig. 15-1.

Table 15-1 shows the relation between the $\overline{IRES}$, $\overline{RES}$, and $\overline{RESET}$ signals, and power on reset and CE reset.

The reset control circuit operates when a clock stop command (STOP s) described in Section **14, "Standby"** is executed. The CE reset and power on reset are described in Sections **15.3** and **15.4.**

Section **15.5** describes the relation between the CE reset and power on reset.

Table 15-1 Relation between internal reset signal and each reset

Internal reset signal	Output signal			Control information
	CE reset	Power on reset	Clock stop	
$\overline{\text{IRES}}$	X	O	O	Sets the device forcibly to halt mode. The halt mode is canceled when a timer carry flip-flop is set.
$\overline{\text{RES}}$	X	O	O	Partially initializes a control register.
$\overline{\text{RESET}}$	O	O	O	Partially initializes a program counter, stack, system register, and control register.

15.3 CE RESET

The CE reset occurs when the CE pin rises from low to high. When the CE pin rises, a $\overline{\text{RESET}}$ signal is output in synchronization with the leading edge of the next timer carry flip-flop setting pulse. The device is then reset.

When the CE reset occurs, a program counter, stack, system register, and control register are partially initialized using the $\overline{\text{RESET}}$ signal and the program is executed from address 0000H. For more information on the initial value, see each section.

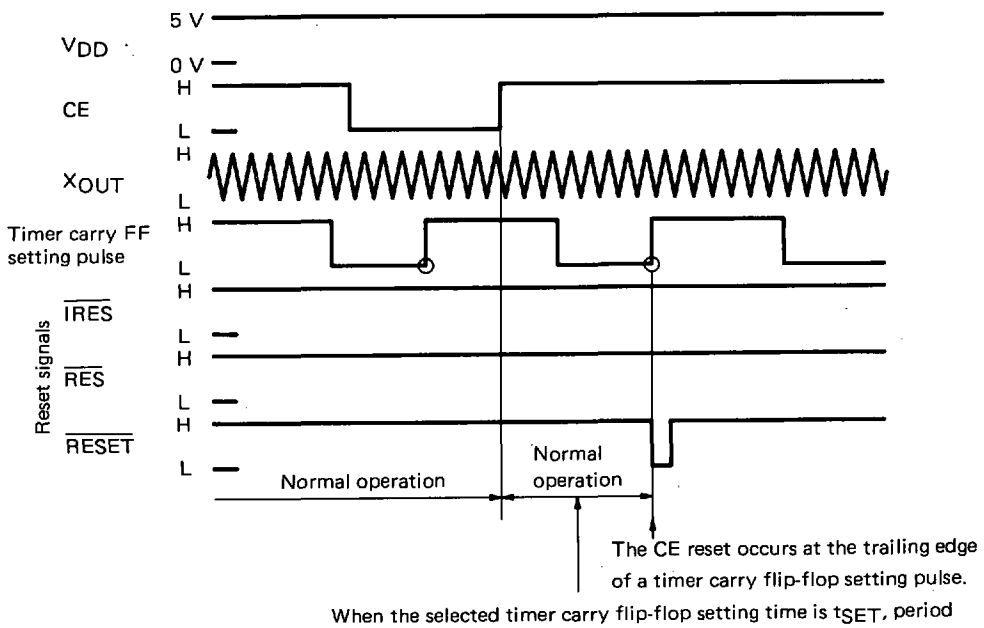
The CE reset operation varies depending on whether a clock stop command is used or not (see Sections **15.3.1** and **15.3.2**). Section **15.3.3** describes cautions during the CE reset.

15.3.1 CE Reset when Clock Stop (STOP) Command is Not Used

Fig. 15-2 shows the CE reset operation.

A timer mode select register in the control register is not initialized when no clock stop (STOP) command is used. A $\overline{\text{RESET}}$ signal is output at the leading edge of a selected timer carry flip-flop setting pulse of 1 ms, 5 ms, 100 ms, or 250 ms after the CE pin is set high. The device is then reset.

Fig. 15-2 CE reset when clock stop command is not used



When the selected timer carry flip-flop setting time is t_{SET}, period "t" becomes $0 < t < t_{SET}$ at the leading edge of a CE pulse. The program operation is continued during this period.

15.3.2 CE Reset when Clock Stop (STOP) Command is Used

Fig. 15-3 shows the CE reset operation.

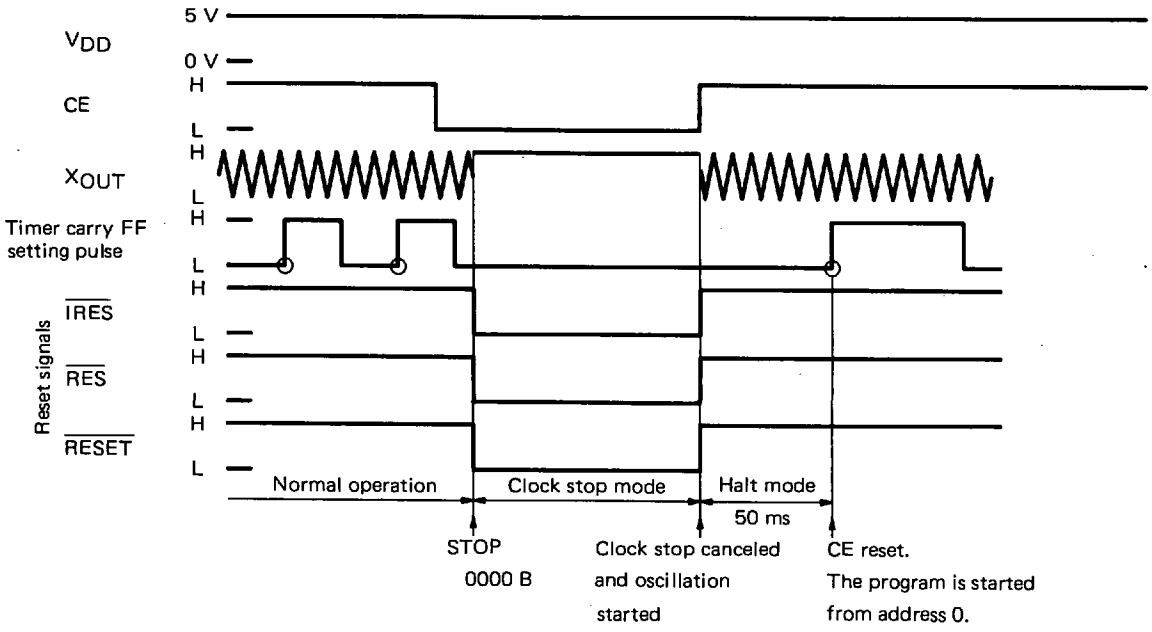
$\overline{IRES}$, $\overline{RES}$, and $\overline{RESET}$ signals are output during STOP s command execution when a clock stop command is used. At that time, a timer mode select register in the control register is initialized as 0000B using the $\overline{RES}$ signal. A timer carry flip-flop setting signal is thus specified as 100 ms.

The $\overline{IRES}$ signal is output continuously while the CE pin is low, so the forcible cancel halt mode is entered using a timer carry flip-flop. However, the clock is stopped, so the device is also stopped.

The clock stop mode is canceled when the CE pin rises from low to high. Oscillation is then started.

The timer carry flip-flop cancel halt mode is entered using the $\overline{IRES}$ signal at that time. Therefore, the halt mode is canceled at the leading edge of a timer carry flip-flop setting pulse after the CE pin rises. The program is then started from address 0. Since the timer carry flip-flop setting pulse is initialized as 100 ms, the CE reset occurs 50 ms after the CE pin rises.

Fig. 15-3 CE reset when clock stop command is used



15.3.3 Cautions During CE Reset

The CE reset occurs irrespective of execution commands, so pay attention to the following:

(1) Timer processing time

The timer program must be created within a fixed time using a timer carry flip-flop or timer interrupt.

For more details, see Section **13.4, "Cautions when Using Timer Carry Flip-Flop"**, and Section **13,6, "Cautions during Timer Interrupt"**.

(2) Program data and flag processing

The information of the data and flag (e.g., security code) that cannot be processed using one command must not be changed during the CE reset. Note when rewriting the security code information.

A security code example is shown below.

Example 1:

```
START:

        Key entry                    ; Security code key entry wait
          R1 ← Key A information     ; The information of the pressed key is substituted for general
          R2 ← Key B information       registers R1 and R2.

SCCHK:
; ①
        SET2    CMP, Z               ; Security code check
        SUB     R1, M1
        SUB     R2, M2
        SKT1    Z
        BR      NOOPERATION          ; The program does not operate if the security code differs.
MAIN:                                ; Main processing

        Key entry                    ; When the key for rewriting the security code is
          R3 ← Key C information     ; pressed, the key information is substituted for
          R4 ← Key D information     ; R3 and R4.

; ②
        ST      M1, R3               ; Security code rewriting
; ③
        ST      M2, R4
        BR      MAIN
```

Assume that the security code is "12H" in example 1 above. The information of data memory M1 and M2 becomes "1H" and "2H".

When a CE pulse is reset, the key entry information and security code "12H" are compared in parameter ①. If they are the same, the program is processed normally. When a security code is altered during main processing, the altered code is rewritten into M1 and M2 in parameters ② and ③.

If the security code is altered into "34H", codes "3H" and "4H" are written into M1 and M2 in parameters ② and ③. The program is reset without executing parameter ③ when the CE reset occurs during the parameter ② execution. The security code thus becomes "32H" and causes a trouble. A program example that prevents this trouble is shown below.

Example 2:

```
START:

        Key entry                          ; Security code key entry wait
          R1 ← Key A information           ; The information of the pressed key is substituted for general
          R2 ← Key B information           ; registers R1 and R2.

; ④
        SKT1    FLG1                       ; If the FLG1 flag is "1",
        BR      SCCHK
        ST      M1, R3                     ; Codes are rewritten in M1 and M2.
        ST      M2, R4
        CLR1    FLG1
SCCHK:
; ①
        SET2    CMP, Z                     ; Security code check
        SUB     R1, M1
        SUB     R2, M2
        SKT1    Z
        BR      NOOPERATION                ; The program does not operate if the security code differs.
MAIN:                                      ; Main processing

        Key entry                          ; When the key for rewriting the security code is
          R3 ← Key C information           ; pressed, the key information is substituted for
          R4 ← Key D information           ; R1 and R2.

; ⑤
        SET1    FLG1                       ; An FLG1 flag is set while the security code is rewritten.
; ②
        ST      M1, R3                     ; Security code rewriting
; ③
        ST      M2, R4
        CLR1    FLG1
        BR      MAIN
```

In example 2 above, an FLG1 flag is set when the security code is rewritten in parameters ② and ③. A code is rewritten in parameter ④ even if the CE reset occurs in parameter ③.

15.4 POWER ON RESET

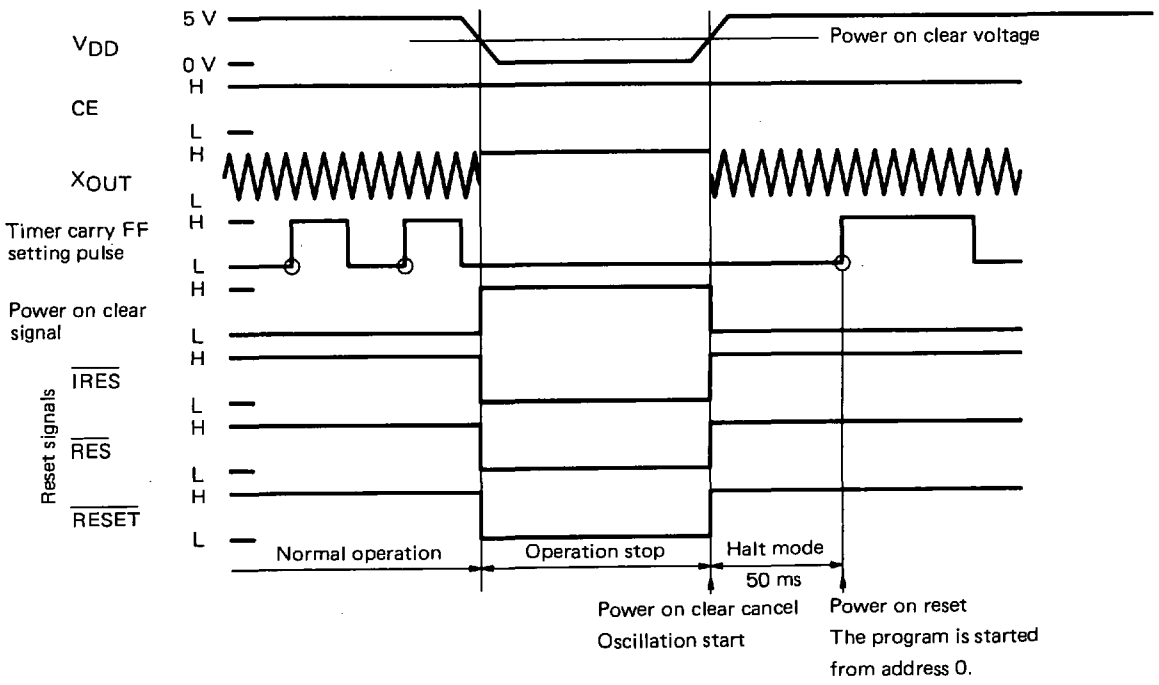
The power on reset occurs when supply voltage V_{DD} rises from less than a fixed voltage (power on clear voltage). A power on clear signal (POC) is output from the voltage detector circuit shown in Fig. 15-1 when supply voltage V_{DD} does not exceed the power on clear voltage. When the power on clear signal is output, a crystal oscillator is stopped and the device is stopped. $\overline{IRES}$, $\overline{RES}$, and $\overline{RESET}$ signals are also output while the power on clear signal is output. The power on clear signal is turned off when supply voltage V_{DD} exceeds the power on clear voltage. The $\overline{IRES}$, $\overline{RES}$, and $\overline{RESET}$ signals are also turned off when the crystal oscillator operates.

The timer carry flip-flop cancel halt mode is entered using the $\overline{IRES}$ signal. The power on reset therefore occurs at the leading edge of the next timer carry flip-flop setting signal. The timer carry flip-flop setting signal is initialized as 100 ms using the $\overline{RESET}$ signal. Consequently, the system is reset 50 ms after supply voltage V_{DD} exceeds the power on clear voltage. The program is then started from address 0. The power on reset operation is shown in Fig. 5-4.

A program counter, stack, system register, and control register during power on reset are initialized when the power on clear signal is output. For more information on the initial value, see each section.

The power on clear voltage is 3.5 V (specification) in normal operating mode, and 2.2 V (specification) in clock stop mode. The power on reset in normal operating and clock stop modes is described in Sections **15.4.1** and **15.4.2.** The operation when supply voltage V_{DD} rises from 0 V is described in Section **15.4.3.**

Fig. 15-4 Power on reset operation



15.4.1 Power On Reset in Normal Operating Mode

The power on reset in normal operating mode is shown in Fig. 15-5 (a).

As shown in Fig. 15-5 (a), a power on clear signal is output when supply voltage V_{DD} becomes less than 3.5 V irrespective of the input level at the CE pin. The device is then stopped. The program is started from address 0000H in halt mode 50 ms after supply voltage V_{DD} exceeds 3.5 V again.

In normal operating mode, no clock stop command is used. The normal operating mode also includes the halt mode in which a halt command is executed.

15.4.2 Power on Reset in Clock Stop Mode

The power on reset in clock stop mode is shown in Fig. 15-5 (b).

As shown in Fig. 15-5 (b), a power on clear signal is output when supply voltage V_{DD} becomes less than 2.2 V. The device is then stopped. However, the device operation does not change visually because the device is in clock stop mode.

The program is started from address 0000H in halt mode 50 ms after supply voltage V_{DD} exceeds 3.5 V.

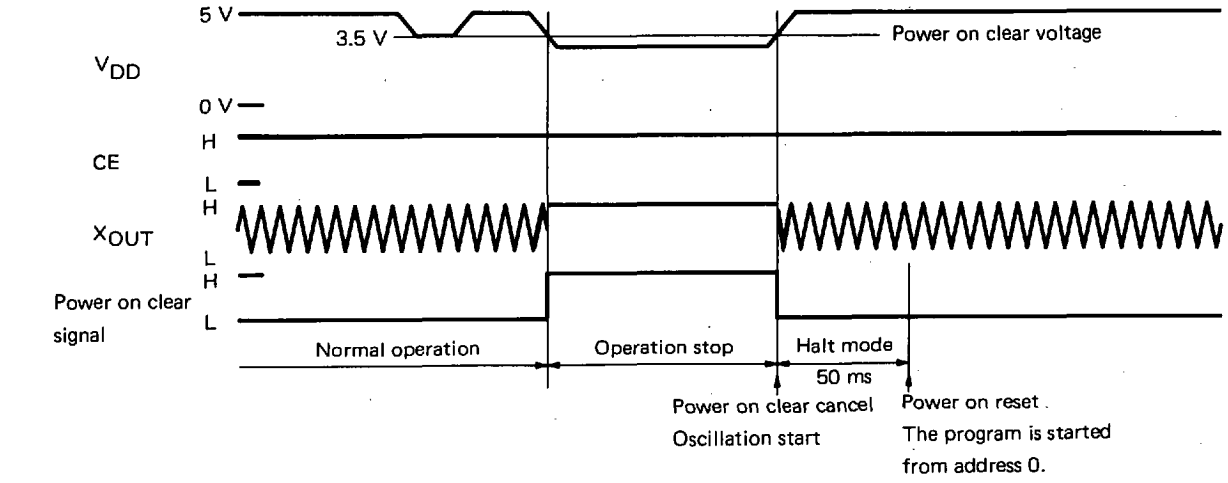
15.4.3 Power on Reset when Supply Voltage V_{DD} Rises from 0 V

The power on reset operation is shown in Fig. 15-5 (c).

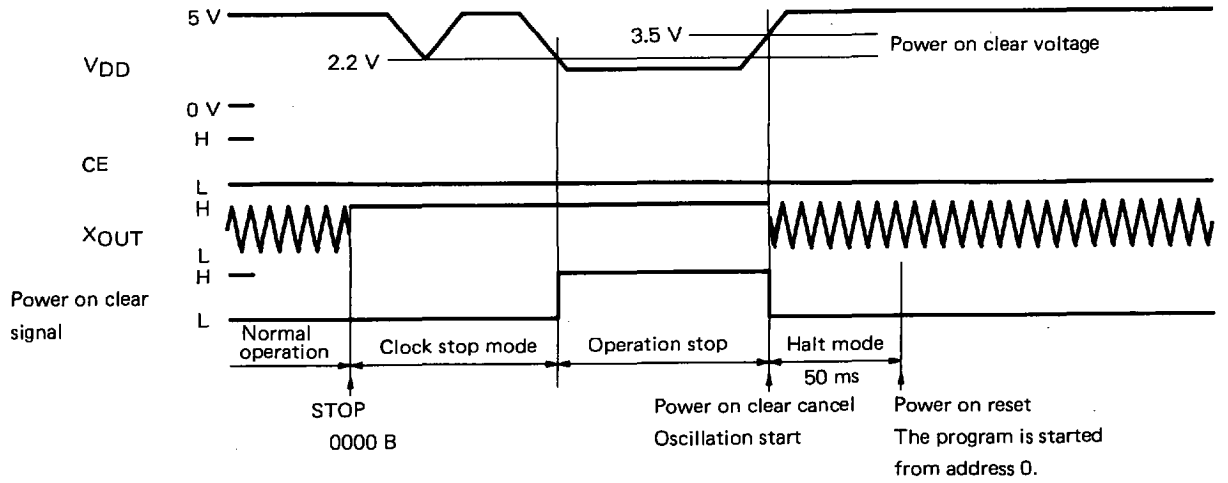
As shown in Fig. 15-5 (c), a power on clear signal is output until supply voltage V_{DD} rises from 0 V to 3.5 V. A crystal oscillator operates when supply voltage V_{DD} exceeds the power on clear voltage. The program is started from address 0000H in halt mode after 50 ms.

Fig. 15-5 Power on reset and supply voltage V_{DD}

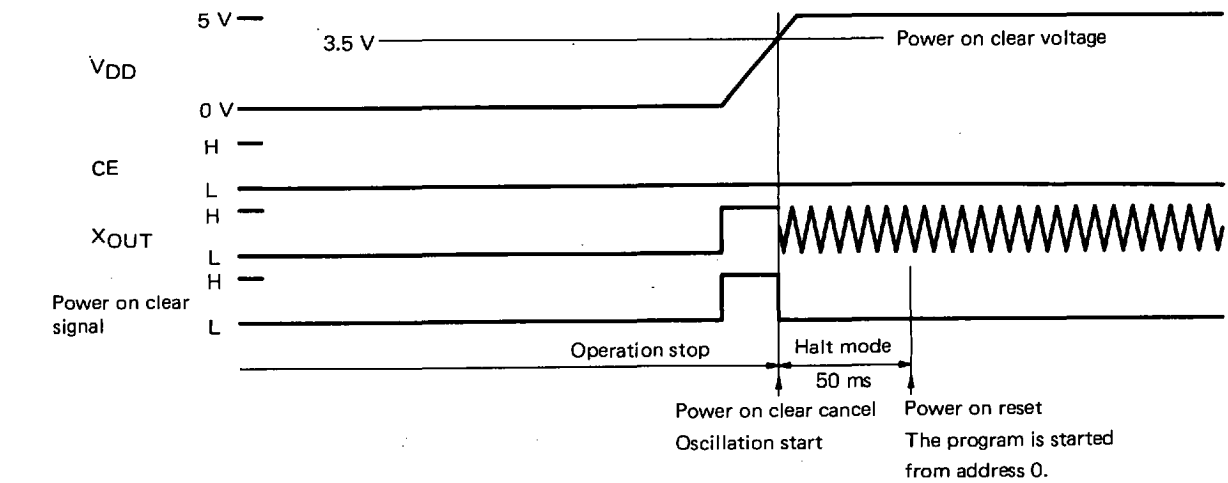
(a) Normal operating mode (including halt mode)



(b) Clock stop mode



(c) Supply voltage V_{DD} rising from 0 V



15.5 RELATION BETWEEN CE RESET AND POWER ON RESET

The power on reset and CE reset may occur at the same time when supply voltage V_{DD} is turned on first. The reset operation at that time is described in Sections **15.5.1** through **15.5.3.** Cautions during the rise of supply voltage V_{DD} are described in Section **15.5.4.**

15.5.1 V_{DD} and CE Pins Rising at the Same Time

The operation when the V_{DD} and CE pins rise at the same time is shown in Fig. 15-6 (a).

The power on reset program is then started from address 0000H.

15.5.2 CE Pin Rising in Power on Reset Halt Mode

The operation when the CE pin rises in halt mode is shown in Fig. 15-6 (b).

The power on reset program is then started from address 0000H as in Section **15.5.1.**

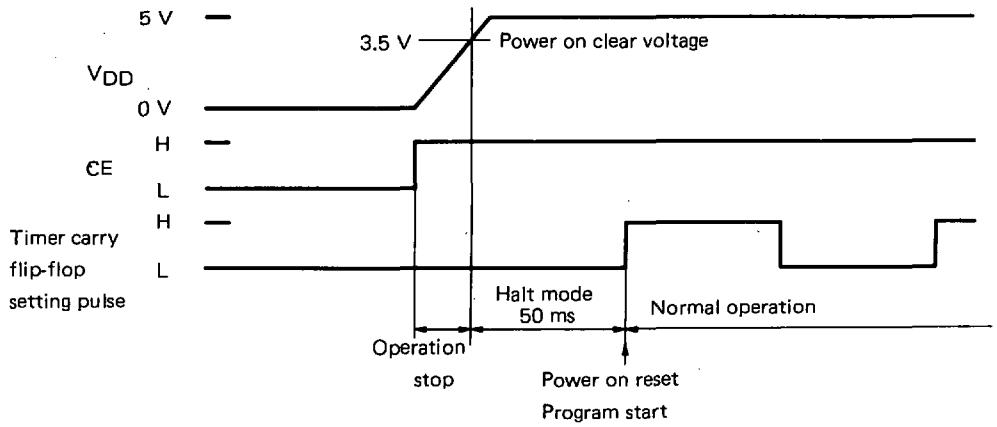
15.5.3 CE Pin Rising after Power on Reset

The operation when the CE pin rises after power on reset is shown in Fig. 15-6 (c).

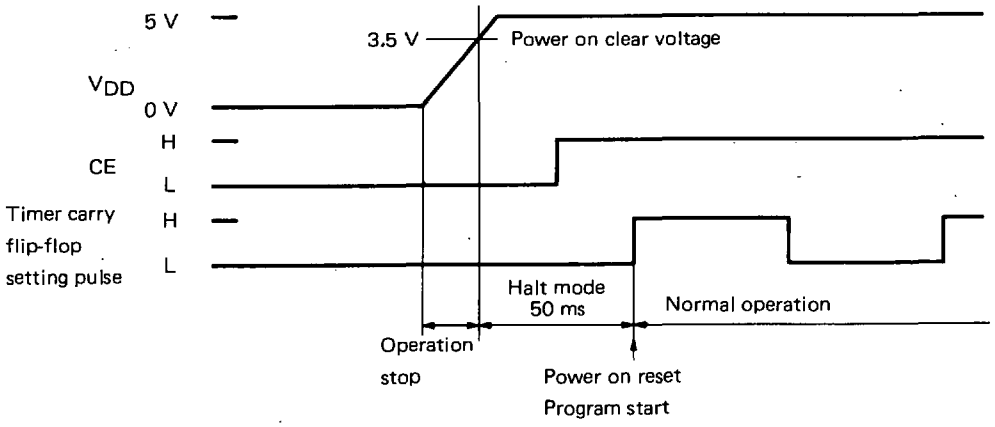
The power on reset program is then started from address 0000H, then restarted from address 0000H at the leading edge of the next timer carry flip-flop setting signal during the CE reset.

Fig. 15-6 Relation between power on reset and CE reset

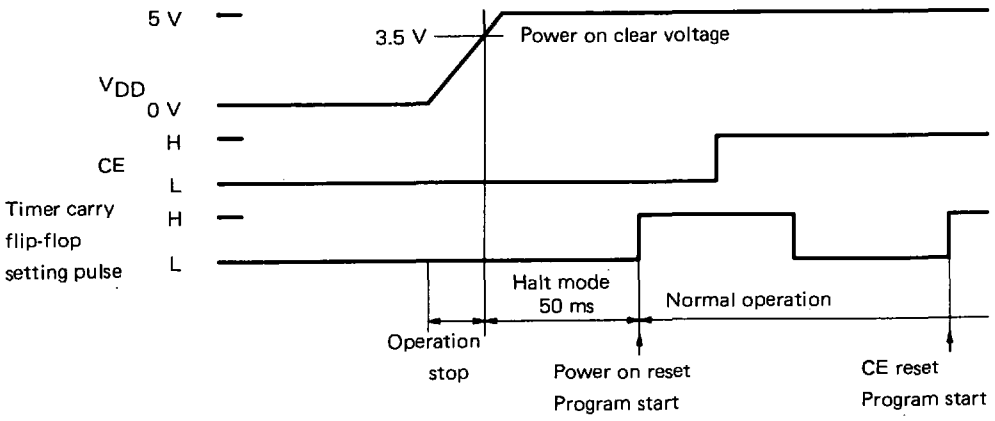
(a) V_{DD} and CE pins rising at the same time



(b) CE pin rising in halt mode



(c) CE pin rising after power on reset



15.5.4 Cautions During Rise of Supply Voltage V_{DD}

Pay attention to the following when supply voltage V_{DD} rises.

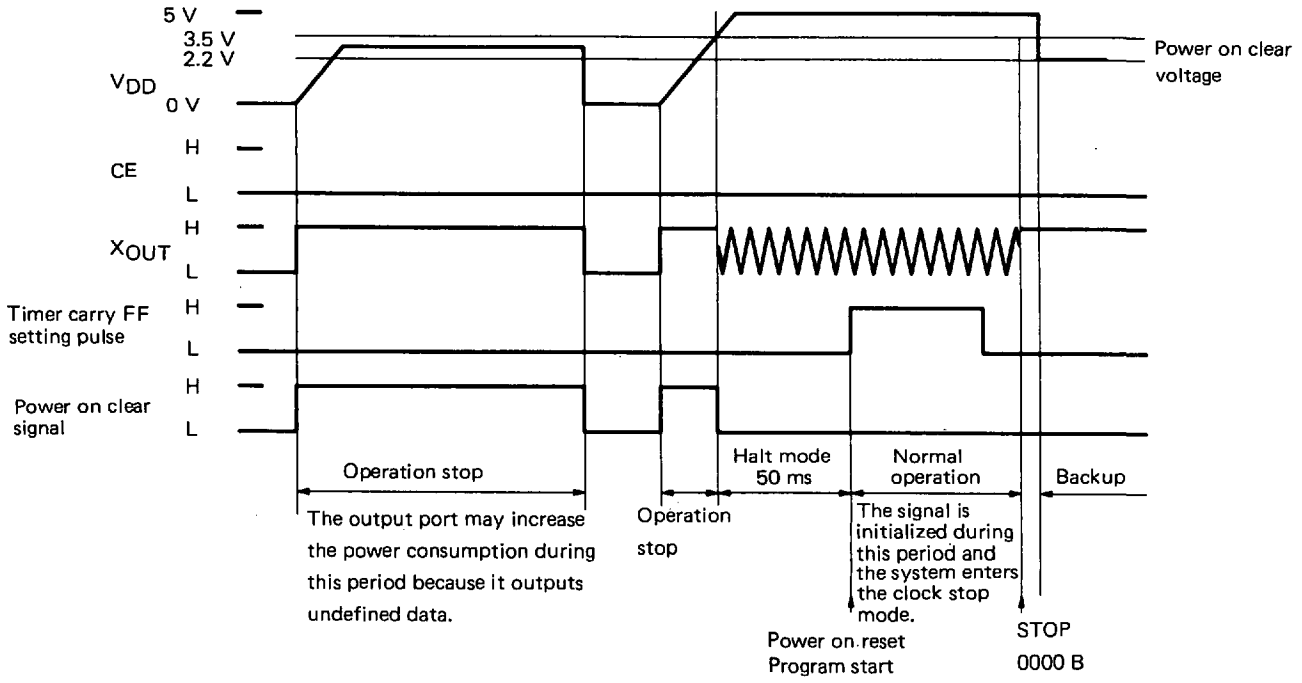
(1) Supply voltage V_{DD} rising from less than power on clear voltage

Supply voltage V_{DD} must first exceed 3.5 V when it rises. The cautions during the rise of the supply voltage are shown in Fig. 15-7.

As shown in Fig. 15-7, a power on clear signal is output if only a voltage of less than 3.5 V is applied when supply voltage V_{DD} is turned on in the program for which the supply voltage is backed up at 2.2 V using a clock stop command. The program then does not operate.

The device output port at that time may increase the power consumption because it outputs an undefined value. The backup time significantly decreases during the battery backup.

Fig. 15-7 Cautions during rise of supply voltage V_{DD}

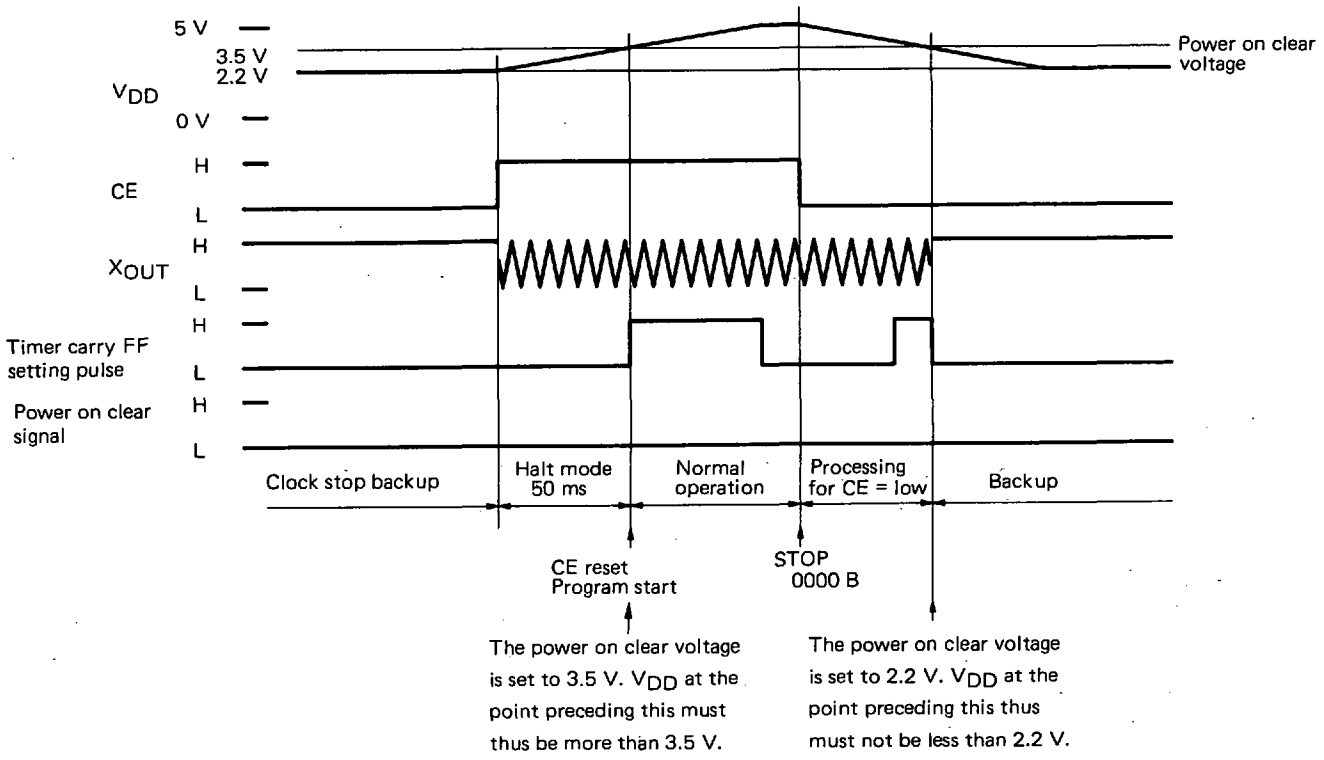


(2) Return from clock stop mode

To return from the backup state when supply voltage V_{DD} is backed up at 2.2 V using a clock stop command, the supply voltage must exceed 3.5 V within 50 ms after the CE pin is set high.

As shown in Fig. 15-8, the system is returned from the clock stop mode during the CE reset. A power on clear voltage is set to 3.5 V 50 ms after the CE pin rises. A power on reset occurs supply voltage V_{DD} does not exceed 3.5 V when the power on clear voltage is set to 3.5 V. Pay the same attention as the above when supply voltage V_{DD} becomes less than 3.5 V.

Fig. 15-8 Return from clock stop mode



15.6 POWER FAILURE DETECTION

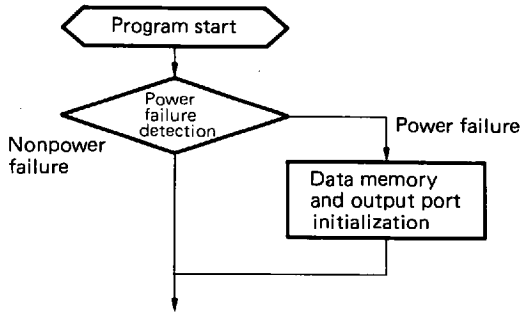
As shown in Fig. 15-9, the power failure function is used to determine whether supply voltage V_{DD} is turned on or the device is reset using a CE pin during the device reset.

The data memory or output port information is "undefined" when the supply voltage is turned on. This information is thus initialized using the power failure function.

This power failure detection consists of the TMCY flag detection employing a power failure detector circuit and the data memory information detection (RAM judge).

The power failure detector circuit and the power failure detection employing a TMCY flag are described in Sections **15.6.1** and **15.6.2.** The RAM judge power failure detection is described in Sections **15.6.3** and **15.6.4.**

Fig. 15-9 Power failure detection flowchart



15.6.1 Power Failure Detector Circuit

As shown in Fig. 15-1, the power failure detector circuit consists of a voltage detector circuit, a timer carry disable flip-flop that is reset using the voltage detector circuit output (power on clear signal), and a timer carry flip-flop. The timer carry disable flip-flop is set (1) using a power on clear signal and reset (0) when a TMCY flag (bit b_0 of address 17H) read command is executed.

A TMCY flag cannot be set (1) when the timer carry disable flip-flop is set (1). When the power on clear signal is output (during power on reset), the program is started with the TMCY flag reset. The TMCY flag is in set inhibit mode until the TMCY flag read command is executed.

The TMCY flag is set every time a timer carry flip-flop setting pulse rises when the TMCY flag read command is executed. When the TMCY flag information is detected during the device reset, a power on reset (power failure) occurs if the TMCY flag is reset ("0"). A CE reset (nonpower failure) occurs if it is set ("1").

The voltage at which power failure can be detected is the same as a power on reset voltage. Therefore, supply voltage V_{DD} becomes 3.5 V during crystal oscillation, and it becomes 2.2 V in clock stop mode.

Fig. 5-10 shows the TMCY flag state transition. Fig. 5-11 shows the timing chart of Fig. 5-10 and the TMCY flag operation.

Fig. 15-10 TMCY flag state transition

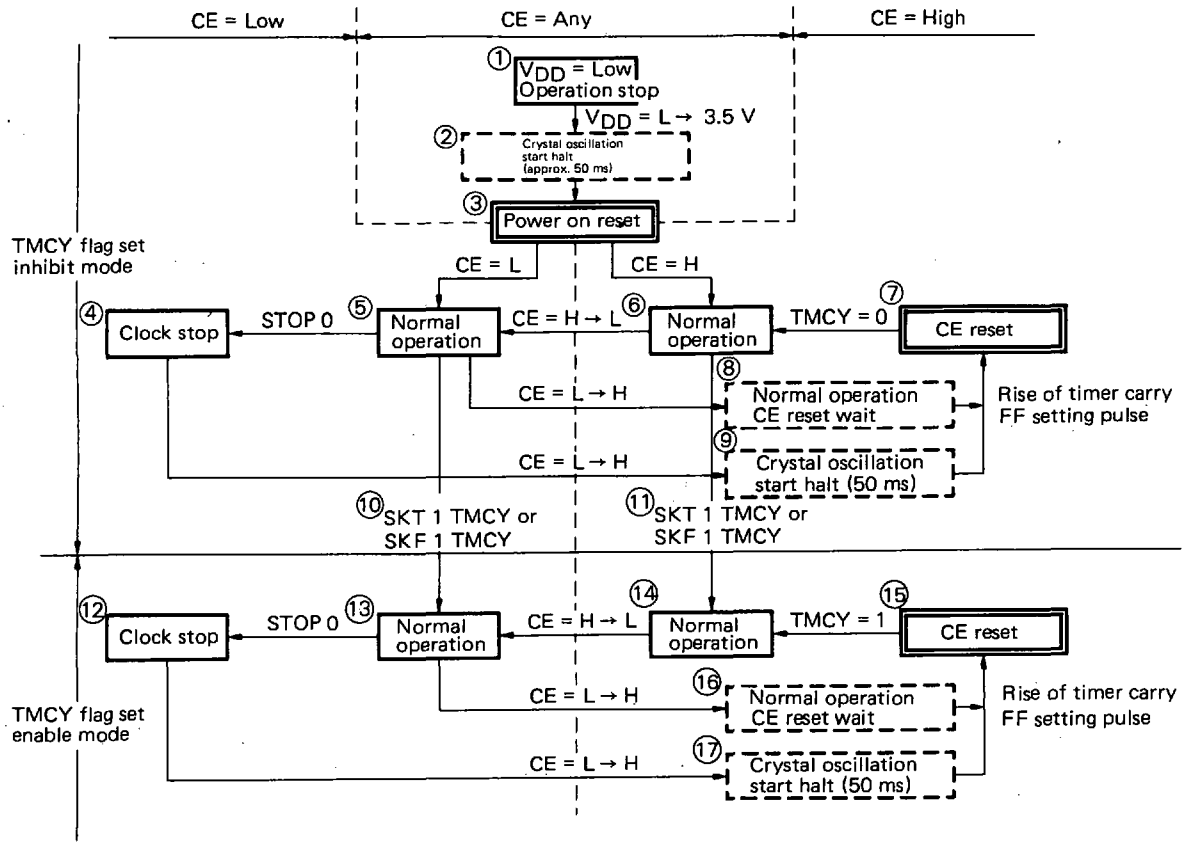
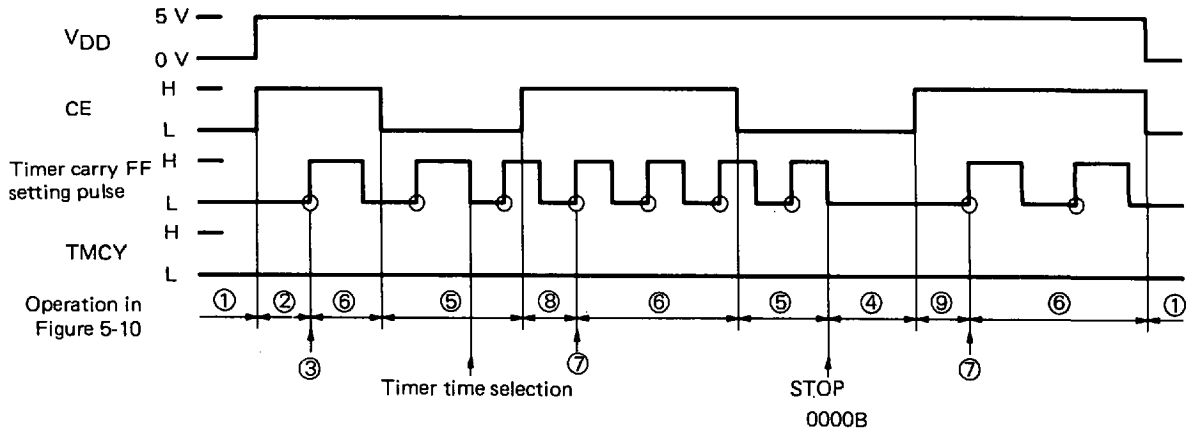
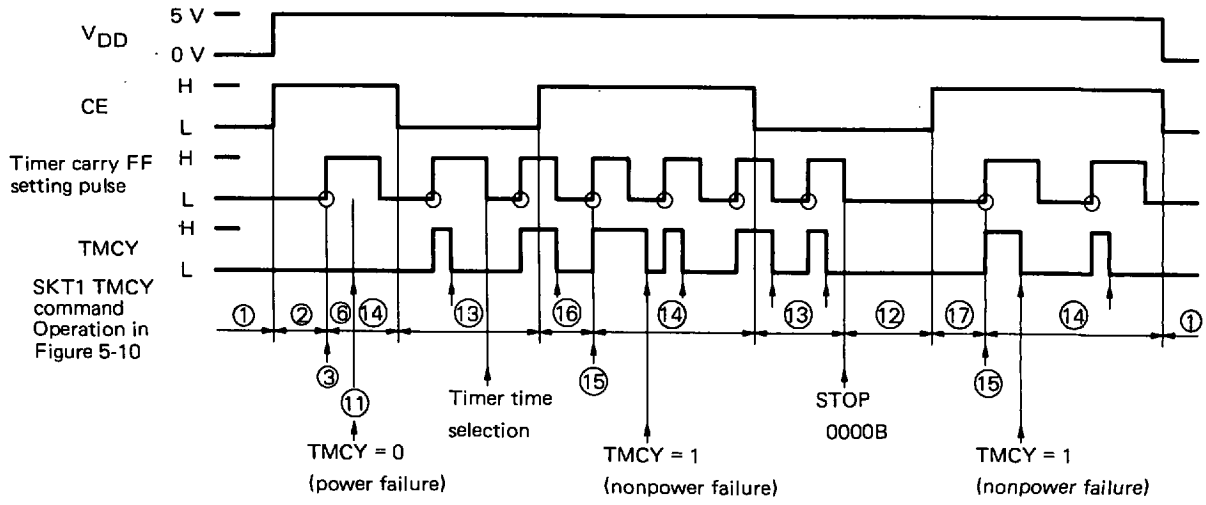


Fig. 5-11 TMCY flag operation

(a) TMCY flag not detected
(SKT1 TMCY or SKF1 TMCY flag not executed)



(b) Power failure detected using TMCY flag



15.6.2 Cautions when Power Failure is Detected Using TMCY Flag

Pay attention to the following when a timer is counted using the TMCY flag:

(1) Timer updating

A timer must be updated after power failure detection when the timer program is created using a timer carry flip-flop. If the timer is not updated, a TMCY flag is reset (0) and one timer count is skipped because the TMCY flag is read during power failure detection.

(2) Timer updating time

A timer must be updated before the next timer carry flip-flop setting pulse rises because a CE reset occurs before the timer updating is completed when the CE pin is set high during timer updating.

For more details of Steps (1) and (2) above, see Section **13.4.2, "Timer Carry Flip-Flop Correction during CE Reset".**

Pay attention to the following for processing during power failure:

(3) Power failure detection timing

A power failure detection TMCY flag must be read from when the program is started from address 0000H until the next timer carry flip-flop setting pulse rises when the timer count is made using a TMCY flag. One TMCY flag is skipped when the timer carry flip-flop setting time is set to 5 ms and when the power failure is detected 6 ms after the program is started. For more information, see Section **13.4.2, "Timer Carry Flip-Flop Correction during CE Reset".**

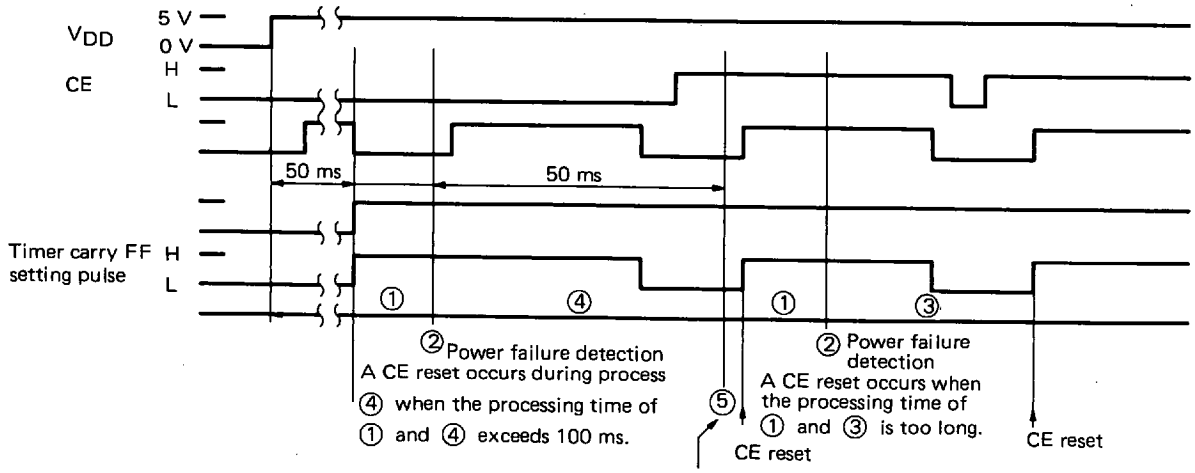
Power failure detection and initialization must be executed within the timer carry flip-flop setting time. The power failure detection and initialization may be interrupted halfway when a CE reset occurs with the CE pin set high during power failure detection and initialization. Use only one alteration command during the last initialization to alter the timer carry flip-flop setting time during initialization. The initialization may not be completed owing to the CE reset when the timer carry flip-flop setting time is set before initialization.

Example:

Program example

```
START:                          ; The program address 0000H
; ①
    Processing during reset     ;
; ②
    SKT1    TMCY                ; Power failure detection
    BR      INITIAL
BACKUP:
; ③
    Timer updating
    BR      MAIN
INITIAL:
; ④
    Initialization
; ⑤
    INITFLG TMMD3, TMMD2, TMMD1, NOT TMMD0
                                ; Built-in macroinstruction
                                ; The timer carry flip-flop setting time is set to 5 ms.
MAIN:
    SKT 1   TMCY
    BR      MAIN
    Timer updating
```

Operation example



15.6.3 RAM Judge Power Failure Detection

The RAM judge function detects the power failure by determining whether the data memory information at a specified address has the specified value during the device reset. A power failure detection program example is shown below.

The data memory information when supply voltage V_{DD} is turned on is "undefined". The RAM judge function detects the power failure by comparing the "undefined" and "specified" values. Therefore, the power failure may be detected incorrectly as described in Section **15.6.4, "Cautions during RAM Judge Power Failure Detection"**. However, the RAM judge function has the advantage that a low supply voltage can be backed up as compared with when the power failure is detected using a power failure detector circuit, as shown in Table 15-2.

Program example of RAM judge power failure detection

```
        M000    MEM     0.00H
        M00F    MEM     0.0FH
        M060    MEM     0.60H
        M06F    MEM     0.6FH
        DATA1   DAT     1010B
        DATA2   DAT     0101B
        DATA3   DAT     0110B
        DATA4   DAT     1001B
START:
        SET1    CMP             ;  The CMP flag is set.
        SET1    Z               ;  The Z flag is set.
        SUB     M000,   DATA1   ;  M000 = DATA1
        SUB     M00F,   DATA2   ;  M00F = DATA2
        SUB     M060,   DATA3   ;  M060 = DATA3
        SUB     M06F,   DATA4   ;  M06F = DATA4
        SKF1    Z               ;  The program branches into BACKUP.
        BR      BACKUP
INITIAL:
        MOV     M000,   DATA1
        MOV     M00F,   DATA2
        MOV     M060,   DATA3
        MOV     M06F,   DATA4
        [ Initialization ]
        BR      MAIN
BACKUP:
        [ Backup ]
MAIN:
        [ Main processing ]
```

Table 15-2 Power failure detector circuit and RAM judge power failure detection

	Power failure detector circuit		RAM judge	
Data holding voltage (in clock stop mode)	Actual value	Specification	Actual value	Specification
	1 – 2 V	2.2 V	0 – 1 V	2.0 V
Other	No malfunction		Malfunction included	

15.6.4 Cautions During RAM Judge Power Failure Detection

The data memory when supply voltage V_{DD} is turned on is "undefined". Therefore, pay attention to the following:

(1) Compared data

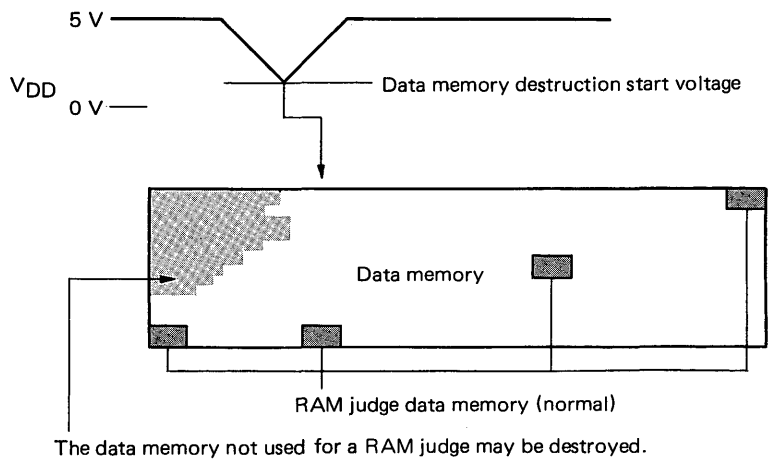
When the bit count of data memory that is compared using a RAM judge function is set to "n", the probability that the data memory coincides with the compared data is $(1/2)^n$ when supply voltage V_{DD} is turned on. This is determined to be a backup at the probability of $(1/2)^n$ during the RAM judge power failure detection. This probability can be lowered by comparing more data bits. The data memory information when supply voltage V_{DD} is turned on often becomes the same value (e.g., "0000B" or "1111B"). Value "1010B" or "0110B" is thus used as the compared data.

(2) Cautions in program

When supply voltage V_{DD} rises from the voltage at which data memory destruction is begun as shown in Fig. 15-12, the compared data memory is normal but sections other than the data memory may be destroyed. This is determined to be a backup during the RAM judge power failure detection.

Therefore, take care that the program is not hung up even if the data memory is destroyed.

Fig. 15-12 V_{DD} and data memory destruction



16. PLL FREQUENCY SYNTHESIZER

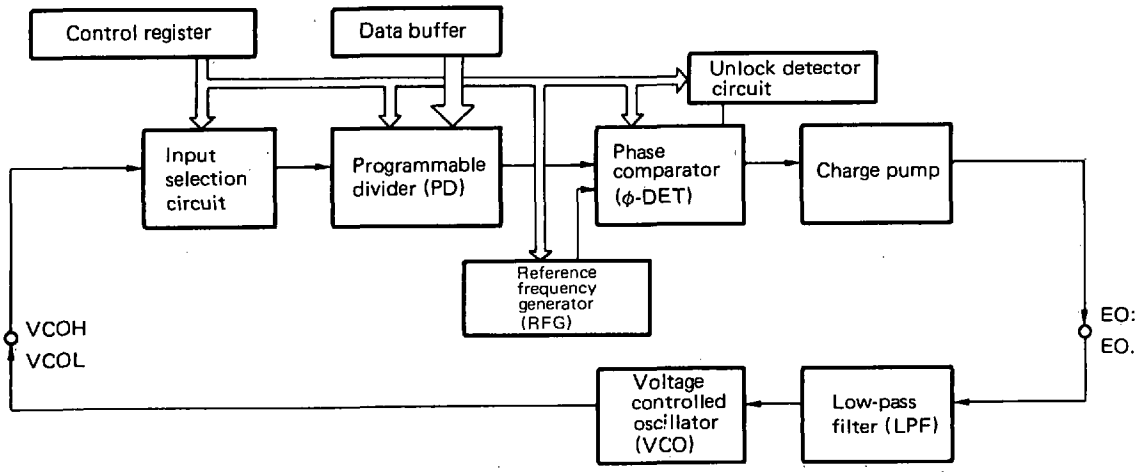
The phase locked loop (PLL) frequency synthesizer is used to lock medium frequency (MF), high frequency (HF), and very high frequency (VHF) signals to a fixed frequency using a phase difference comparison system.

16.1 PLL FREQUENCY SYNTHESIZER CONFIGURATION

Fig. 16-1 shows the PLL frequency synthesizer block diagram.

As shown in Fig. 16-1, the PLL frequency synthesizer consists of an input selection circuit, programmable divider (PD), phase comparator (ϕ-DET), reference frequency generator (RFG), and charge pump. These blocks are connected to an external low-pass filter (LPF) and voltage controlled oscillator (VCO). The PLL frequency synthesizer also has an internal CMOS operational amplifier so that it can be used as an external low-pass filter amplifier.

Fig. 16-1 PLL frequency synthesizer block diagram



16.2 PLL FREQUENCY SYNTHESIZER FUNCTIONS

The PLL frequency synthesizer divides the frequency of a signal from the VCOH pin (pin 32) or VCOL pin (pin 31) using a programmable divider and outputs the phase difference between the divided frequency and reference frequency from EO_1 and EO_0 pins.

The PLL frequency synthesizer operates only when the CE pin is high; it enters the disable mode when the CE pin is low. For more information on the PLL disable mode, see Section **16.7.** Sections **16.2.1** through **16.2.6** describe each block function.

16.2.1 Input Selection Circuit

The input selection circuit selects the pin to which the signal output from an external voltage controlled oscillator is input. A VCOH or VCOL pin is selected as the input pin using a PLL mode select register (RF address 21H) (see Section **16.3**).

16.2.2 Programmable Divider

The programmable divider divides the frequency of a signal from the VCOH or VCOL pin at the frequency division ratio that is set using a program.

A direct frequency division system or pulse swallow system can be selected using a PLL mode select register. The frequency division value is set via the data buffer using a PLL data register (peripheral address 41H) (see Section **16.3**).

16.2.3 Reference Frequency Generator

The reference frequency generator produces the reference frequency that is compared using a phase comparator. Twelve reference frequencies can be selected using a PLL reference mode select register (RF address 31H) (see Section **16.4**).

16.2.4 Phase Comparator and Unlock Detector Circuit

The phase comparator compares the frequency-divided signal output from a programmable divider and the signal from a reference frequency generator and outputs the phase difference.

The unlock detector circuit detects the PLL unlock state. The PLL unlock state is detected using a PLL unlock flip-flop delay control register (RF address 15H) and PLL unlock flip-flop judge register (RF address 05H) (see Section **16.5**).

16.2.5 Charge Pump

The charge pump outputs the signal from a phase comparator from the EO_1 and EO_0 pins as high, low, and floating ouput signals (see Section **16.5**).

16.2.6 Low-Pass Filter (LPF) Amplifier

The low-pass filter amplifier is a CMOS operational amplifier that is used for an external low-pass filter (see Section **16.6**).

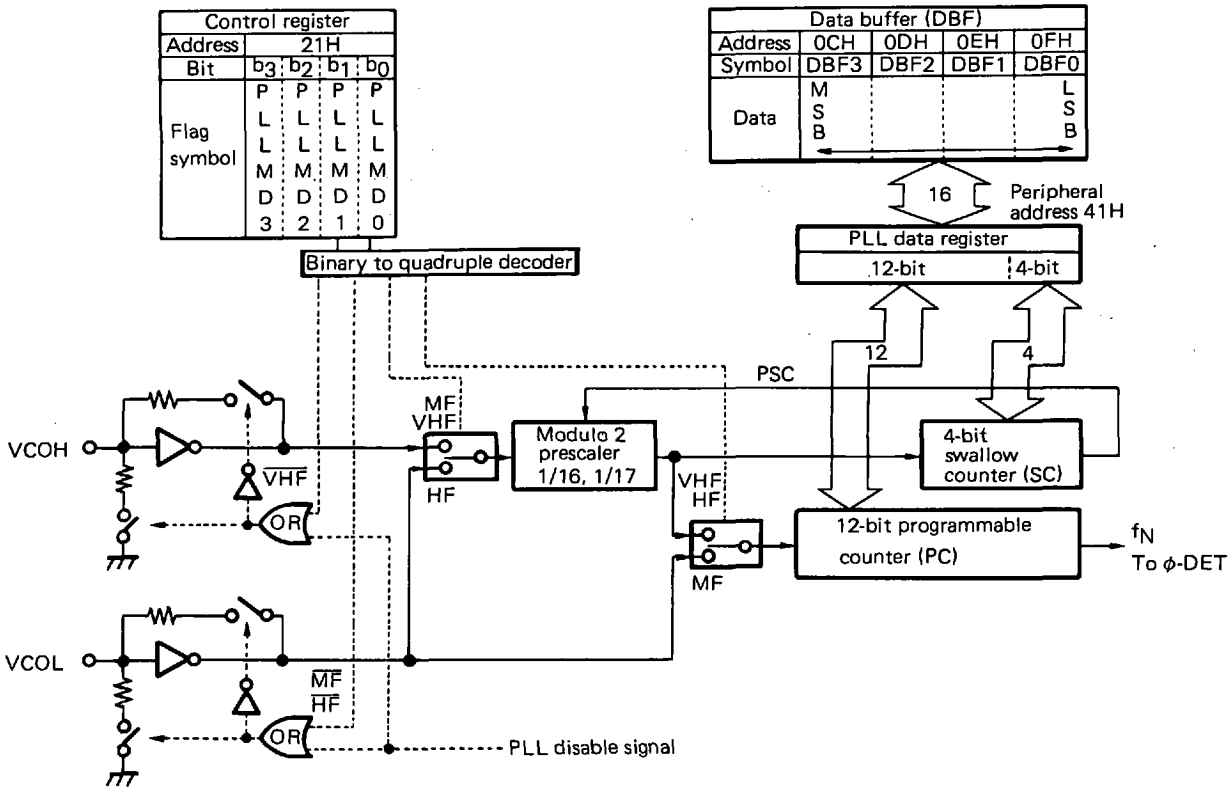
16.3 INPUT SELECTION CIRCUIT AND PROGRAMMABLE DIVIDER

16.3.1 Input Selection Circuit and Programmable Divider Configuration

Fig. 16-2 shows the input selection circuit and programmable divider configuration.

As shown in Fig. 16-2, the input selection circuit consists of a VCOH pin, VCOL pin, and two input amplifiers. The programmable divider consists of a modulo two prescaler, swallow counter (SC), programmable counter (PC), and frequency division selection switch.

Fig. 16-2 Input selection circuit and programmable divider configuration



16.3.2 Input Selection Circuit and Programmable Divider Functions

The input selection circuit and programmable divider selects the input pin and frequency division system of a PLL frequency synthesizer.

A VCOH or VCOL pin can be selected as the input pin, and a direct frequency division system or pulse swallow system can be selected as the frequency division system.

The programmable divider divides a frequency in accordance with the value that is set in a swallow counter and programmable counter. Table 16-1 shows the input pins (VCOH and VCOL) and frequency division systems. The input pin used and the frequency division system are selected using a PLL mode select register. The configuration and functions of the PLL mode select register are described in Section **16.3.3.**

The frequency division value of the programmable divider is set via the data buffer using a PLL data register.

Section **16.3.4** describes the programmable divider and PLL data register.

Table 16-1 Input pin and frequency division system

Frequency division system	Pin used	Input frequency (MHz)	Input amplitude (V_{p-p})	Possible frequency division value
Direct frequency division (MF)	VCOL	0.5 to 30	0.3	16 to $2^{12} - 1$
Pulse swallow (HF)	VCOL	5 to 40	0.3	256 to $12^{16} - 1$
Pulse swallow (VHF)	VCOH	9 to 150	0.3	256 to $2^{16} - 1$

16.3.3 PLL Mode Select Register (PLLMODE) Configuration and Functions

The PLL mode select register sets the frequency division system and input pin of a PLL frequency synthesizer. The PLL mode select register configuration and functions are shown below. Steps (1) through (4) below describe the frequency division outline.

PLL mode select register configuration and functions

Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
PLL mode select register (PLLMODE)	PLLMD3	PLLMD2	PLLMD1	PLLMD0	21H	R/W

PLLMD3	PLLMD2	Sets the frequency division system of a PLL frequency synthesizer.
0	0	VCOL and VCOH pin disable
0	1	Direct frequency division (VCOL pin MF mode)
1	0	Pulse swallow (VCOH pin VHF mode)
1	1	Pulse swallow (VCOL pin HF mode)

Fixed at "0".

		PLLMD3	PLLMD2	PLLMD1	PLLMD0
Reset	Power on	0	0	0	0
	Clock stop			0	0
	CE			Held	Held

(1) Direct frequency division system (MF)

The VCOL pin is used, and the VCOH pin is pulled down. The direct frequency division system divides the frequency in only a programmable counter.

(2) Pulse swallow system (HF)

The VCOL pin is used, and the VCOH pin is pulled down. The pulse swallow system divides the frequency in a swallow counter and programmable counter.

(3) Pulse swallow system (VHF)

The VCOH pin is used, and the VCOL pin is pulled down. The pulse swallow system divides the frequency in a swallow counter and programmable counter.

(4) VCOL and VCOH pin disable

VCOL and VCOH pins are pulled down internally. However, a phase comparator, reference frequency generator, and charge pump operate. The VCOL and VCOH pin disable mode thus differs in operation from the PLL disable mode (described later).

16.3.4 Programmable Divider and PLL Data Register

The programmable divider divides the frequency of a signal from the VCOH and VCOL pins in accordance with the value that is set in a swallow counter and programmable counter. The swallow counter consists of a four-bit binary down counter, and the programmable counter consists of a twelve-bit binary down counter. The frequency division value of the swallow counter and programmable counter is set via the data buffer using a PLL data register (PLLR, address 41H).

The PLL data register data is set and read using "PUT PLLR, DBF" and "GET DBF, PLLR" commands. The frequency division value is called "value N".

The relation between the PLL data register and data buffer is described below. For more details of the frequency division value (value N) setting in each frequency division system, see Section **16.8.**

(1) PLL data register and data buffer

The PLL data register to data buffer relation is described below.

In the direct frequency division system, the high-order 12 bits are valid. In the pulse swallow system, all 16 bits are valid. The 12 bits in the direct frequency division system are set in a program counter. The high-order 12 bits in the pulse swallow system are set in a program counter, and the low-order 4 bits are set in a swallow counter.

(2) Relation between frequency division value "N" and frequency division output frequency of programmable Divider

Value "N" that is set in a PLL data register and output signal frequency "f_N" that is frequency divided using a programmable divider are shown below. For more information, see Section **16.8.**

(a) Direct frequency division (MF)

$$f_N = \frac{f_{in}}{N}$$

where N is 12 bits.

(b) Pulse swallow system (HF and VHF)

$$f_N = \frac{f_{in}}{N}$$

where N is 16 bits.

Name	Data buffer															
Symbol	DBF 3				DBF 2				DBF 1				DBF 0			
Address	0 CH				0 DH				0 EH				0 FH			
Bit	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Data	Transfer data															

16

GET and PUT commands can be entered.

Peripheral register																		
Name	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Symbol	Peripheral address	Peripheral hardware
PLL data register	Valid data																PLLR	41 H	PLL frequency synthesizer

Sets the frequency division ratio of a PLL frequency synthesizer.

Direct frequency division system	0	Any	Setting inhibition
	I		
	15 (00 FH)	Any	
	16 (010 H)	Any	Frequency division ratio N: N = x
	I		
	X	Any	
	I		
	$2^{12} - 1$ (FFFH)	Any	
Pulse swallow system	0		Setting inhibition
	I		
	255 (00 FFH)		
	256 (0100 H)		Frequency division ratio N: N = x
	I		
	X		
	I		
	$2^{16} - 1$ (FFFFH)		

16.4 REFERENCE FREQUENCY GENERATOR

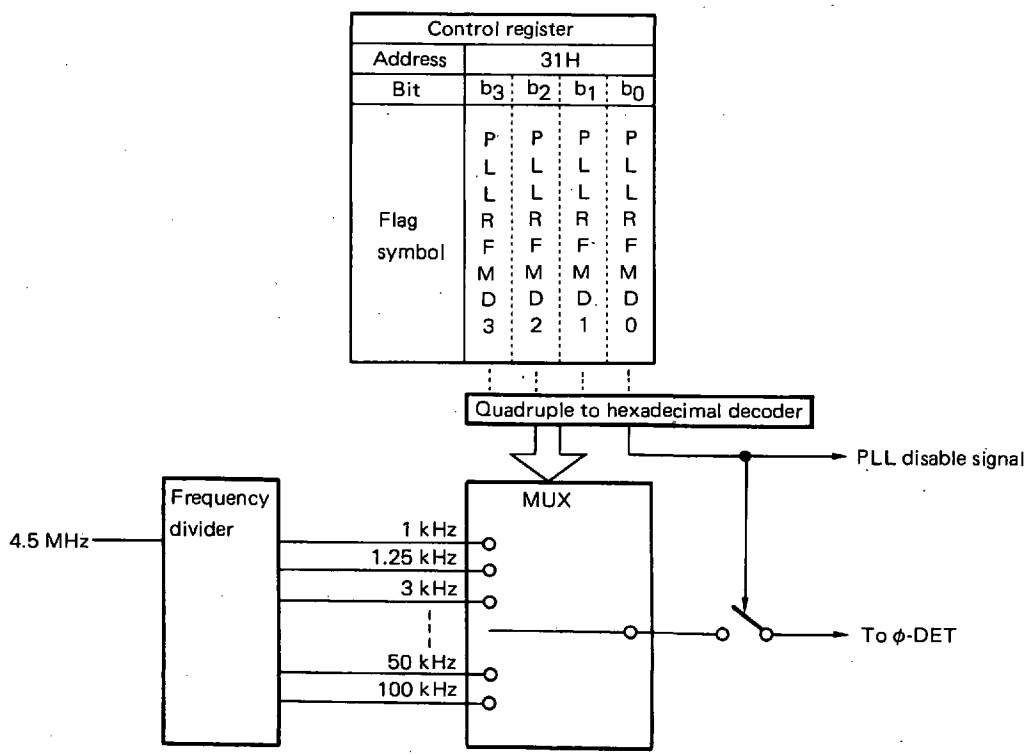
16.4.1 Reference Frequency Generator Configuration and Functions

Fig. 16-3 shows the reference frequency generator configuration.

As shown in Fig. 16-3, the reference frequency generator divides a crystal oscillation frequency of 4.5 MHz and generates reference frequency f_r of a PLL frequency synthesizer.

Twelve reference frequencies (1, 1.25, 2.5, 3, 5, 6.25, 9, 10, 12.5, 25, 50, 100 kHz) can be selected using a PLL reference mode select register. The PLL reference mode select register is described in Section **16.4.2.**

Fig. 16-3 Reference frequency generator (RFG) configuration



16.4.2 PLL Reference Mode Select Register Configuration and Functions

The configuration and functions are shown below.

Name	Flag symbol				Address	Read/Write
	b_3	b_2	b_1	b_0		
PLL reference mode select register (PLRFMODE)	PLLRFMD3	PLLRFMD2	PLLRFMD1	PLLRFMD0	31H	R/W

				Sets reference frequency f_r of a PLL frequency synthesizer.
0	0	0	0	1.25 kHz
0	0	0	1	2.5 kHz
0	0	1	0	5 kHz
0	0	1	1	10 kHz
0	1	0	0	6.25 kHz
0	1	0	1	12.5 kHz
0	1	1	0	25 kHz
0	1	1	1	50 kHz
1	0	0	0	3 kHz
1	0	0	1	Setting inhibition
1	0	1	0	Setting inhibition
1	0	1	1	Setting inhibition
1	1	0	0	1 kHz
1	1	0	1	9 kHz
1	1	1	0	100 kHz
1	1	1	1	PLL disable

Reset					
	Power on	1	1	1	1
	Clock stop	1	1	1	1
	CE	Held			

The VCOH and VCOL pins are pulled down internally when the PLL disable mode is set using the PLL reference mode select register. The EO_1 and EO_0 pins are also set floating, and the LPF_{IN} pin of an internal CMOS operational amplifier is pulled up internally. For more details of the PLL disable, see Section **16.7.**

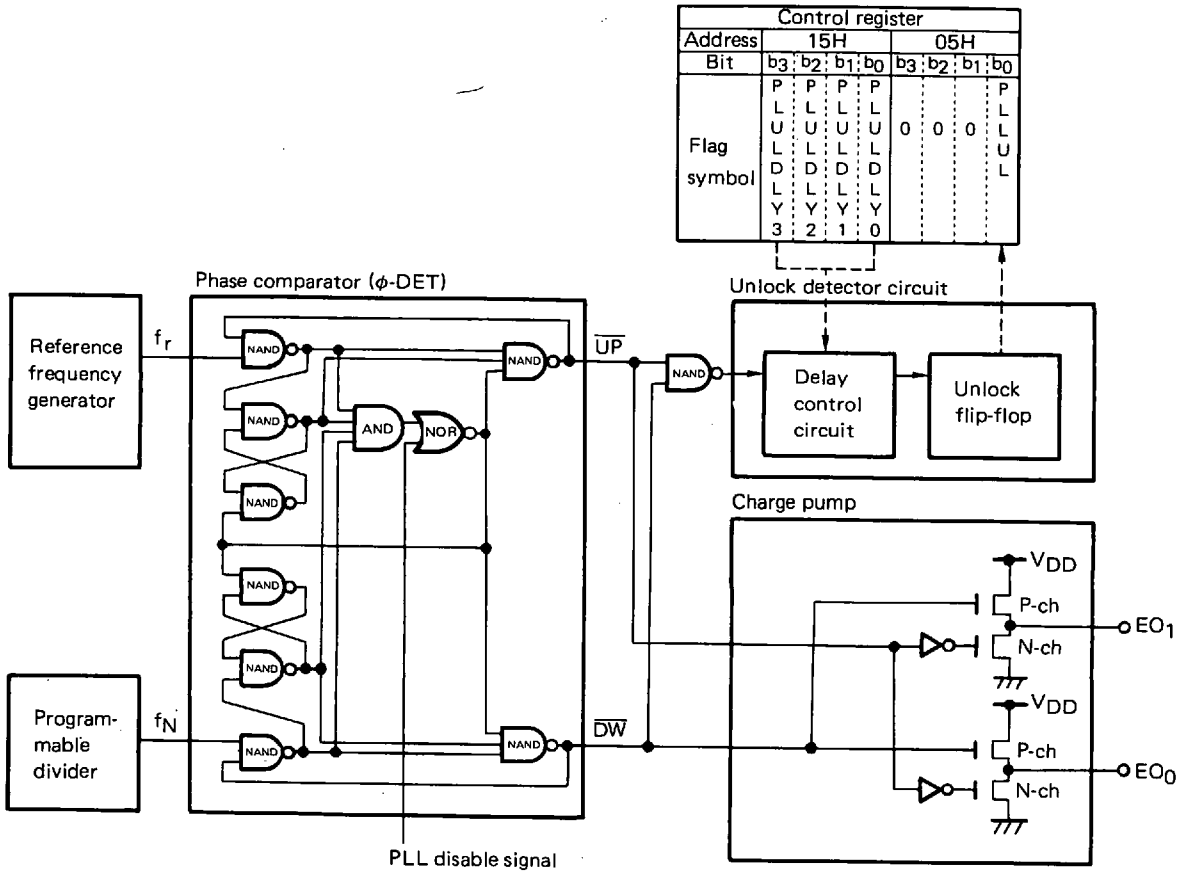
16.5 PHASE COMPARATOR (φ-DET), CHARGE PUMP, AND UNLOCK DETECTOR CIRCUIT

16.5.1 Phase Comparator, Charge Pump, and Unlock Detector Circuit Configuration

Fig. 16-4 shows the phase comparator, charge pump, and unlock detector circuit configuration. The phase comparator compares the phase of frequency division output "f_N" from a programmable divider and that of reference frequency output "f_r" from a reference frequency generator and outputs up request (UP) and down request (DW) signals. The charge pump outputs the output of the phase comparator from error output pins (EO_1 and EO_0 pins). The unlock detector circuit consisting of a delay control circuit and unlock flip-flop detects the unlock state of a PLL frequency synthesizer.

The phase comparator, charge pump, and unlock detector circuit operations are described in Sections **16.5.2** through **16.5.4.**

Fig. 16-4 Phase comparator, charge pump, and unlock detector circuit configuration



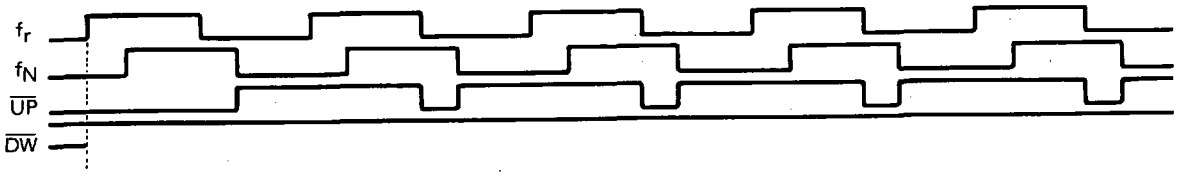
16.5.2 Phase Comparator Functions

As shown in Fig. 16-4, the phase comparator compares the phase of frequency division output "f_N" from a programmable divider and that of reference frequency output "f_r" from a reference frequency generator and outputs up request and down request signals. The up request signal is output if divided frequency f_N is lower than reference frequency f_r. The down request signal is output if the former is higher than the latter.

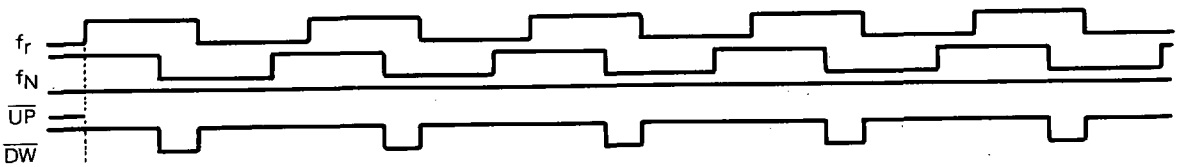
Fig. 16-5 shows the reference frequency (f_r), divided frequency (f_N), up request signal, and down request signal. In PLL disable mode, the up and down request signals are output. The up and down request signals are input to the charge pump and unlock detector circuit.

Fig. 16-5 Relation between f_r, f_N, $\overline{UP}$, and $\overline{DW}$ signals

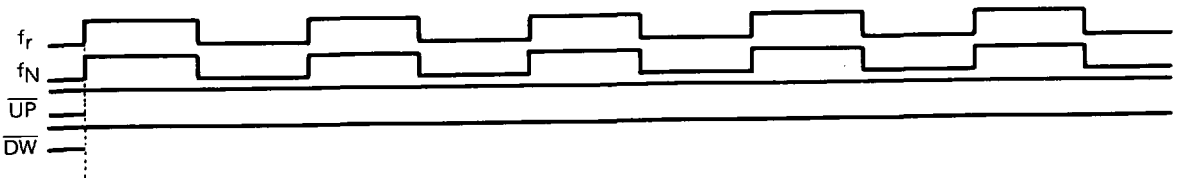
(a) f_N is phase delayed with respect to f_r.



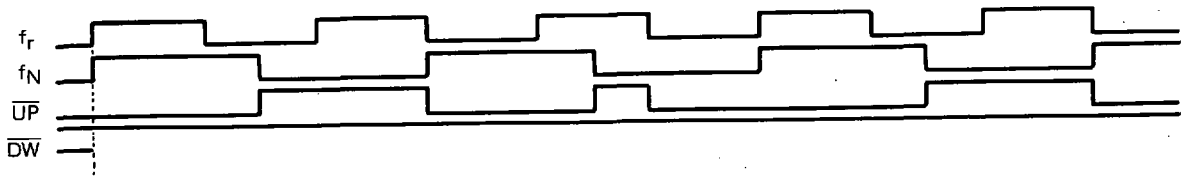
(b) f_N is phase advanced with respect to f_r.



(c) f_N is in phase with f_r.



(d) f_N is lower than f_r.



16.5.3 Charge Pump

As shown in Fig. 16-4, the charge pump outputs the up and down request signals from a phase comparator from error output pins (EO_1 and EO_0 pins).

The relation between the error output pin output, divided frequency f_N, and reference frequency f_r is shown below.

Reference frequency f_r > Divided frequency f_N: Low level output
Reference frequency f_r < Divided frequency f_N: High level output
Reference frequency f_r = Divided frequency f_N: Floating

16.5.4 Unlock Detector Circuit

As shown in Fig. 16-4, the unlock detector circuit detects the unlock state of a PLL frequency synthesizer using the up and down request signals from a phase comparator.

The up and down request signals cause a low signal to be output when the PLL frequency synthesizer is in unlock state. Therefore, the low signal output enables the detection of the unlock state. An unlock flip-flop (FF) is set (1) in unlock state. The unlock FF state is detected using a PLL unlock flip-flop judge register. An unlock flip-flop is set in accordance with the period of reference frequency f_r selected at that time. The unlock flip-flop is also reset when the PLL unlock flip-flop judge register information is read using a PEEK command. This unlock flip-flop must thus be detected at a period longer than period $1/f_r$ of reference frequency f_r.

The unlock delay control circuit controls the unlock flip-flop setting state by delaying the up and down request signals from a phase comparator. If the up and down request signals are delayed significantly, the unlock flip-flop is not set even if the divided frequency f_N and reference frequency f_r phases are shifted considerably. The delay time of the unlock delay control circuit is set using the PLL unlock flip-flop delay control judge register.

The configuration and functions of the PLL unlock flip-flop judge register and PLL unlock flip-flop delay control register are described in Sections **16.5.5** and **16.5.6.**

16.5.5 PLL Unlock Flip-Flop Judge Register (PLLULJDG)

Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
PLL unlock flip-flop judge register (PLLULJDG)	0	0	0	PLLUL	05H	R & Reset

	Detects the unlock flip-flop state.
0	Unlock FF = 0: PLL lock state
1	Unlock FF = 1: PLL unlock state

Fixed at "0".

Reset						
	Power on	0	0	0	Not held	
	Clock stop				Held	
	CE	↓	↓	↓	Held	

The PLL unlock flip-flop judge register that is a read only register is reset when the register information is read in a window register using a PEEK command. The unlock flip-flop is set at a period of reference frequency f_r. Therefore, this register must be read at a period longer than period $1/f_r$ of a reference frequency when it is read in the window register.

16.5.6 PLL Unlock Flip-Flop Delay Control Register (PLULDLY)

Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
PLL unlock flip-flop delay control register (PLULDLY)	0	0	PLULDLY1	PLULDLY0	15H	R/W

Sets the delay time of reference frequency f_r and divided frequency f_N required to set the unlock flip-flop.

0	0	0.9 to 1.0 μs or more
0	1	1.9 to 2.0 μs or more
1	0	0.45 to 0.55 μs or more
1	1	Unlock flip-flop disable (Always set.)

Fixed at "0".

Reset					
	Power on	0	0	0	0
	Clock stop			0	0
	CE			Held	Held

When the unlock flip-flop disable mode is set, the unlock flip-flop remain set at all times. Consequently, when the PLL unlock state is detected using a PLL unlock flip-flop judge register, the unlock state is always held (the PLLUL flag is set to "1").

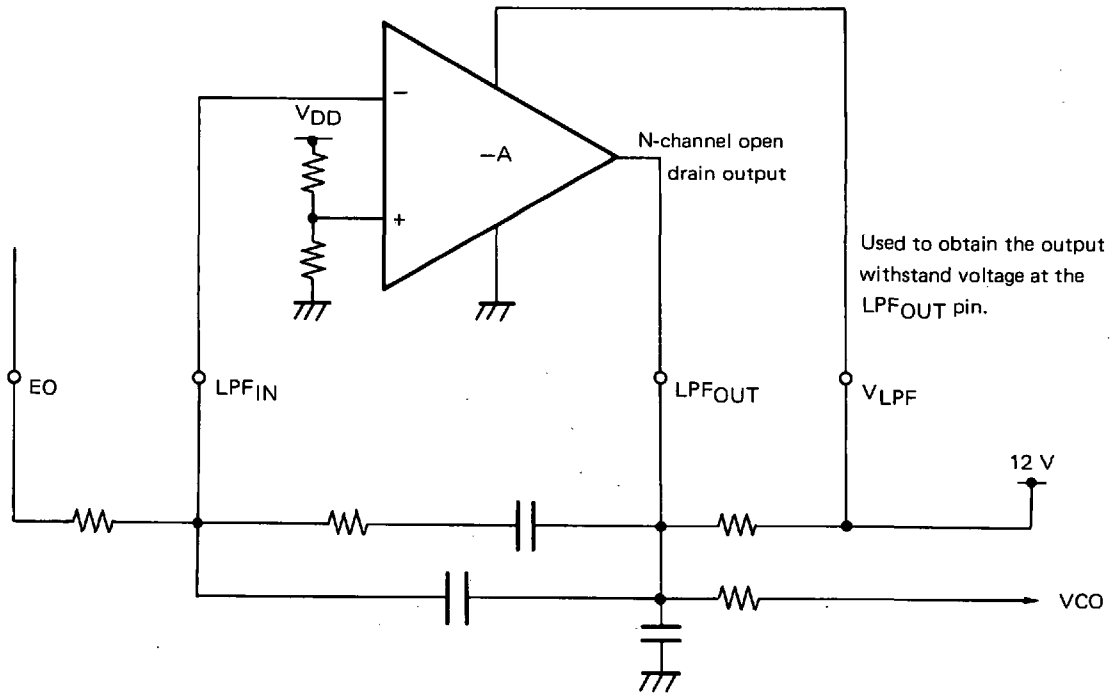
16.6 LOW-PASS FILTER OPERATIONAL AMPLIFIER

16.6.1 Operational Amplifier Configuration and Functions

Fig. 16-6 shows the operational amplifier and low-pass filter configuration.

As shown in Fig. 16-6, the noninverting input terminal of the operational amplifier is kept at an intermediate potential. The V_{LPF} pin is used to obtain the output withstand voltage (16 V, max) at the LPF_{OUT} pin. A voltage higher than the voltage that is applied to the LPF_{OUT} pin must thus be supplied to the V_{LPF} pin.

Fig. 16-6 Operation amplifier and low-pass filter configuration



16.7 PLL DISABLE MODE

A PLL frequency synthesizer is stopped (disabled) while the CE pin (pin 13) is low. The synthesizer is also stopped when the PLL disable mode is set using a PLL reference mode select register.

Table 16-2 shows the block operation in each PLL disable mode. Only the VCOL and VCOH pins are pulled down internally when the VCOL and VCOH pin disable mode is set using a PLL mode select register. The other blocks operate at that time.

The PLL reference mode select register and PLL mode select register are not initialized during the CE reset (the former state is held). Therefore, the former state is returned when the CE pin is set high after the CE pin is set low and the PLL disable mode is entered. Initialize using a program to enter the PLL disable mode during the CE reset. The PLL disable mode is also entered during the power on reset.

Table 16-2 Block operation in each PLL disable mode

Mode / Block	CE pin = Low (PLL disable)	CE pin = High	
		PLRFMODE = 1111B (PLL disable)	PLLMODE = 0000B (VCOH, VCOL disable)
VCOL and VCOH pins	Pulled down internally.	Pulled down internally.	Pulled down internally.
Programmable counter	Frequency division stops.	Frequency division stops.	Operates.
Reference frequency generator	Output stops.	Output stops.	Operates.
Phase comparator	Output stops.	Output stops.	Operates.
Charge pump	Error output pin is set to floating state.	Error output pin is set to floating state.	Operates (low signal is usually output because there is no input).
Operational amplifier	LPF_{IN} pin is pulled down internally.	LPF_{IN} pin is pulled down internally.	Operates.

16.8 USE OF PLL FREQUENCY SYNTHESIZER

The data below is required to control a PLL frequency synthesizer.

(1) Frequency division system : Direct frequency division (MF) and pulse swallow (HF and VHF)
(2) Pin used : VCOL and VCOH pins
(3) Reference frequency : f_r
(4) Frequency division value : N

Setting the PLL data in each frequency division system (MF, HF, and VHF) is described in Sections **16.8.1** through **16.8.3.**

16.8.1 Direct Frequency Division System

(1) Frequency division system selection

The direct frequency division system is selected using a PLL mode select register.

(2) Pin used

The VCOL pin can operate when the direct frequency division system is selected.

(3) Reference frequency f_r setting

The reference frequency is set using a PLL reference mode select register.

(4) Frequency division value N calculation

The frequency division value is calculated as follows:

$$N = \frac{f_{VCOL}}{f_r}$$

where

f_{VCOL}: Input frequency at VCOL pin
f_r : Reference frequency

(5) PLL data setting example

The data used to receive the MW-band broadcasting station below is set as follows:

Receive frequency : 1422 kHz (MW band)
Reference frequency : 9 kHz
Intermediate frequency : +450 kHz

Frequency division value N is given by

$$N = \frac{f_{VCOL}}{f_r} = \frac{1422 + 450}{9} = 208 \text{ (decimal)}$$

$$= 0D0H \text{ (hexadecimal)}$$

Data is set in a PLL data register (PLLR, address 41H), PLL mode select register (PLLMODE, address 21H), and PLL reference mode select register (PLRFMOD, address 31H) as shown below.

PLLR			
0 0 0 0	1 1 0 1	0 0 0 0	Any
0	D	0	

PLLMODE	PLRFMODE
0 0 0 1	1 1 0 1
MF	9 kHz

16.8.2 Pulse Swallow System (HF)

(1) Frequency division system selection

The pulse swallow system is selected using a PLL mode select register.

(2) Pin used

The VCOL pin can operate when the pulse swallow system is selected.

(3) Reference frequency f_r setting

The reference frequency is set using a PLL reference mode select register.

(4) Frequency division value N calculation

The frequency division value is calculated as follows:

$$N = \frac{f_{VCOL}}{f_r}$$

where

f_{VCOL}: Input frequency at VCOL pin

f_r : Reference frequency

(5) PLL data setting example

The data used to receive the SW-band broadcasting station below is set as follows:

Receive frequency : 25.50 MHz (SW band)

Reference frequency : 5 kHz

Intermediate frequency : +450 kHz

Frequency division value N is given by

$$N = \frac{f_{VCOL}}{f_r} = \frac{25500 + 450}{5} = 5190 \text{ (decimal)}$$

$$= 1446\text{H (hexadecimal)}$$

Data is set in a PLL data register (PLLR, address 41H), PLL mode select register (PLLMODE, address 21H), and PLL reference mode select register (PLRFMOD, address 31H) as shown below.

PLLR			
0 0 0 1	0 1 0 0	0 1 0 0	0 1 1 0
1	4	4	6

PLLMODE	PLRFMODE
0 0 1 1	0 0 1 0
HF	5 kHz

16.8.3 Pulse Swallow System (VHF)

(1) Frequency division system selection

The pulse swallow system is selected using a PLL mode select register.

(2) Pin used

The VCOH pin can operate when the pulse swallow system is selected.

(3) Reference frequency f_r setting

The reference frequency is set using a PLL reference mode select register.

(4) Frequency division value N calculation

The frequency division value is calculated as follows:

$$N = \frac{f_{VCOH}}{f_r}$$

where

f_{VCOH}: Input frequency at VCOH pin

f_r : Reference frequency

(5) PLL data setting example

The data used to receive the FM-band broadcasting station below is set as follows:

Receive frequency : 100.0 MHz (FM band)

Reference frequency : 25 kHz

Intermediate frequency : +10.7 MHz

Frequency division value N is given by

$$N = \frac{f_{VCOH}}{f_r} = \frac{100.0 + 10.7}{0.025} = 4428 \text{ (decimal)}$$

$$= 114CH \text{ (hexadecimal)}$$

Data is set in a PLL data register (PLLR, address 41H), PLL mode select register (PLLMODE, address 21H), and PLL reference mode select register (PLRFMOD, address 31H) as shown below.

PLLR			
0 0 0 1	0 0 0 1	0 1 0 0	1 1 0 0
1	1	4	C

PLLMODE	PLRFMODE
0 0 1 0	0 1 1 0
VHF	25 kHz

16.9 STATE DURING RESET

16.9.1 State During Power on Reset

The PLL reference mode register is initialized to 1111B, so the PLL disable mode is entered.

16.9.2 State During Clock Stop

The PLL disable mode is entered when the CE pin is set low.

16.9.3 State During CE Reset

(1) CE reset from clock stop

The PLL reference mode register is initialized to 1111B during clock stop, so the PLL disable mode is entered.

(2) CE reset during nonclock stop

The PLL reference mode register holds the former state. The former setting mode is thus entered when the CE pin is set high.

16.9.4 State in Halt Mode

The setting mode is held when the CE pin is high.

17. GENERAL-PURPOSE PORT

The general-purpose port outputs a high, low, or floating signal to the external circuit and reads a high or low signal from the external circuit.

17.1 GENERAL-PURPOSE PORT CONFIGURATION AND FUNCTIONS

Fig. 17-1 shows the general-purpose port block diagram and Table 17-1 shows the type of a general-purpose port.

As shown in Fig. 17-1, the general-purpose port consists of port 0A (P0A) through port 2A (P2A) that set data in accordance with addresses 70H through 73H (port register) of each data memory bank, port 0F (P0F) and port 0E (P0E) that set data in accordance with addresses 6BH and 6DH of data memory bank 0, and port 0Y (P0Y) and port 0X (P0X) that set data via a data buffer (DBF).

Each port consists of general-purpose port pins. For example, port 0A consists of $P0A_3$ through $P0A_0$ pins.

As shown in Table 17-1, each general-purpose port is classified as a combined input and output port (input/output port), input only port (input port), or output only port (output port). The input/output port is also classified as a bit I/O port that can specify the input or output in one-bit (one-pin) unit or a group I/O port that can specify the input or output in four bits (four pins).

Fig. 17-1 General-purpose port block diagram

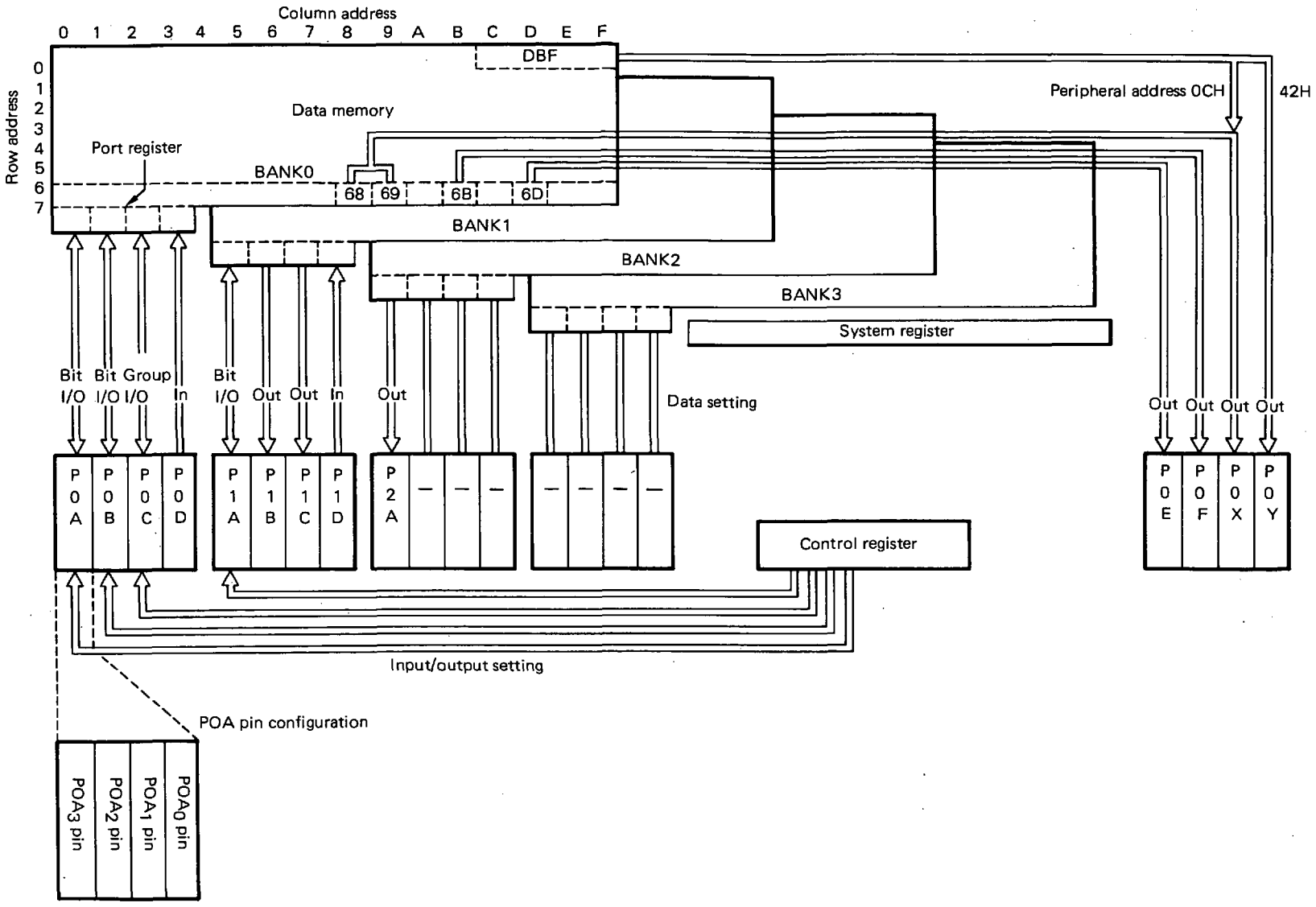


Table 17-1 General-purpose port classification

General-purpose port			Port	Data setting
General-purpose port	Combined input and output port	Bit I/O	Port 0A Port 0B Port 1A	Port register
		Group I/O	Port 0C	Port register
	Input only port		Port 0D Port 1D	Port register
	Output only port		Port 1B Port 1C Port 2A	Port register
			Port 0E Port 0F	Port register (combined with LCD dot register)
			Port 0X	Peripheral register
			Port 0Y	

17.2 GENERAL-PURPOSE PORT FUNCTIONS

The general-purpose input/output port that is set as a general-purpose output port or output port outputs a high or low signal from the corresponding pin when data is set to the port register or port group register.

The general-purpose input/output port that is set as a general-purpose input port or input port can detect the level of an input signal fed to the corresponding pin when the port register information is read. The input and output ports of the general-purpose input/output port are selected using a control register corresponding to each port (i.e., the input or output can be selected using a program).

The P0A through P0D, P1A through P1D, and P2A pins are set to the general-purpose port during the power on reset. The pins that are used in combination with other peripheral hardware are thus set independently using a corresponding control register. Since the P0E, P0F, and P0Y pins are set to the LCD segment signal output pin during the power on reset, they are specified independently using a corresponding control register when used as general-purpose output ports.

The port register functions, port group register functions, and each port function are described in Sections **17.2.1** through **17.2.5.**

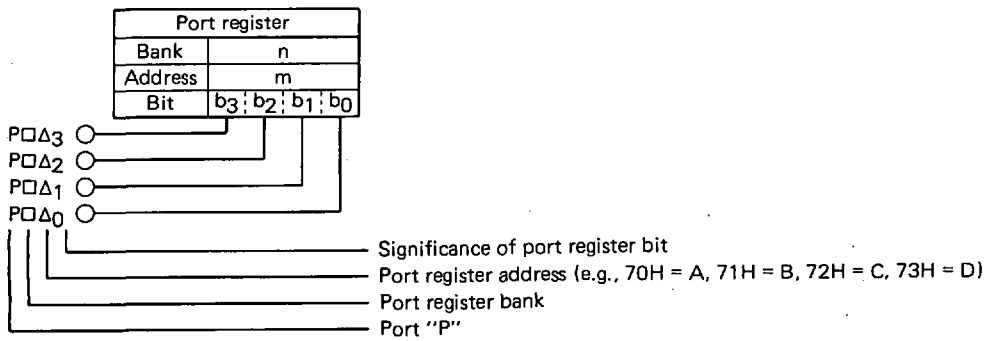
17.2.1 General-Purpose Port Data Register (Port Register)

The port register sets the general-purpose port output data or reads its input data. The port register is allocated in data memory, so it can be activated using data memory commands.

Fig. 17-2 shows the relation between the port register and the corresponding pins. Each pin output is set when data is set to the port register corresponding to the pin that is set to a general-purpose output port. The input state at each pin is detected when the port register corresponding to the pin that is set to a general-purpose input port is read.

The relation between each port (pin) and the port register is shown in Table 17-2.

Fig. 17-2 Port register to pin relation



A reserved word is defined in the assembler (AS17K) of the port register. This reserved word is defined in flags (bits), so an assembler built-in macroinstruction can be used. Notice that a data memory-type reserved word is not defined in the port register.

The P0E, P0F, P0X, and P0Y pins are used in combination with the LCD segment signal output. Therefore, the port registers are used in combination with LCD dot registers. The LCD dot registers are also allocated in data memory, so they can be handled as in the port registers.

17.2.2 Port Group Register

The port group register sets the POX and POY output data. The port group register is controlled via a data buffer (DBF) using "PUT" and "GET" commands. The output data can be set 6 bits or 16 bits at a time using the port group register.

17.2.3 General-Purpose Input/Output Ports (POA, POB, POC, and P1A)

The POA, POB, POC, and P1A inputs or outputs are selected using a POA bit I/O select register (RF address 37H), POB bit I/O select register (RF address 36H), POC group I/O select register (RF address 27H), and P1A bit I/O select register (RF address 35H).

The POA, POB, POC, and P1A input or output data is set using port POA (data memory address, BANK0 address 70H) of a port register, port POB (data memory address, BANK0 address 71H), port POC (data memory address, BANK0 address 72H), and P1A (data memory address, BANK0 address 70H) (see Table 17-2). For more information, see Section **17.3.**

17.2.4 General-Purpose Input Ports (POD and P1D)

The POD and P1D input data is read using port POD (data memory address, BANK0 address 73H) of a port register and P1D (data memory address, BANK1 address 73H) (see Table 17-2). For more information, see Section **17.4.**

17.2.5 General-Purpose Output Ports (P1B, P1C, P2A, POE, POF, POX, and POY)

(1) P1B, P1C, and P2A

The P1B, P1C, and P2A output data is set using port P1B (data memory address, BANK1 address 71H) of a port register, port P1C (data memory address, BANK1 address 72H), and port P2A (data memory address, BANK2 address 70H) (see Table 17-2). For more information, see Section **17.5.**

(2) POE, POF, POX, and POY

Ports POE, POF, POX, and POY usually operate as LCD segment signal output pins. The POE, POF, POX, and POY outputs are selected using an LCD port select register (RF address 11H).

The POE and POF output data is set using a POE register (combined with LCD dot register LCDD13; data memory address, BANK0 address 6BH) and POF register (combined with LCD dot register LCD11; data memory address, BANK0 address 6BH). The POX output data is set using a POXL register (combined with LCD dot register LCDD8; data memory address, BANK0 address 68H) and POXH register (combined with LCD dot register LCDD11; data memory address, BANK0 address 69H) or is set via the data buffer using a POX group data register (combined with an LCD group data register; peripheral address 0CH).

The POY output data is set via the data buffer using a POY group data register (combined with a key source data register; peripheral address 42H) (see Table 17-2). For more information, see Section **17.6.**

Table 17-2 Port (pin) to port register relation (1/2)

Port	Pin			Data setting					
				Port register (data memory)					Remarks
	NO.	Symbol	Input/ output	Bank	Address	Symbol	Bit symbol (reserved word)		
Port0A (P0A)	3	$P0A_3$	Input/ output (bit I/O)	BANK0	70H	P0A	b_3	P0A3	
	4	$P0A_2$					b_2	P0A2	
	5	$P0A_1$					b_1	P0A1	
	6	$P0A_0$					b_0	P0A0	
Port0B (P0B)	7	$P0B_3$	Input/ output (bit I/O)		71H	P0B	b_3	P0B3	
	8	$P0B_2$					b_2	P0B2	
	9	$P0B_1$					b_1	P0B1	
	10	$P0B_0$					b_0	P0B0	
Port0C (P0C)	79	$P0C_3$	Input/ output (group I/O)		72H	P0C	b_3	P0C3	
	80	$P0C_2$					b_2	P0C2	
	1	$P0C_1$					b_1	P0C1	
	2	$P0C_0$					b_0	P0C0	
Port0D (P0D)	75	$P0D_3$	Input		73H	P0D	b_3	P0D3	
	76	$P0D_2$					b_2	P0D2	
	77	$P0D_1$					b_1	P0D1	
	78	$P0D_0$					b_0	P0D0	
Port1A (P1A)	14	$P1A_3$	Input/ output (bit I/O)	BANK1	70H	P1A	b_3	P1A3	
	15	$P1A_2$					b_2	P1A2	
	16	$P1A_1$					b_1	P1A1	
	17	$P1A_0$					b_0	P1A0	
Port1B (P1B)	18	$P1B_3$	Output		71H	P1B	b_3	P1B3	
	19	$P1B_2$					b_2	P1B2	
	20	$P1B_1$					b_1	P1B1	
	21	$P1B_0$					b_0	P1B0	
Port1C (P1C)	22	$P1C_3$	Output		72H	P1C	b_3	P1C3	
	23	$P1C_2$					b_2	P1C2	
	24	$P1C_1$					b_1	P1C1	
	25	$P1C_0$					b_0	P1C0	
Port1D (P1D)	26	$P1D_3$	Input		73H	P1D	b_3	P1D3	
	27	$P1D_2$					b_2	P1D2	
	28	$P1D_1$					b_1	P1D1	
	29	$P1D_0$					b_0	P1D0	
Port2A (P2A)	No pin			BANK2	70H	P2A	b_3	P2A3	Not allocated.. Cannot be used as data memory.
							b_2	P2A2	
							b_1	P2A1	
	42	$P2A_0$	Output				b_0	P2A0	
					71H	---	b_3 b_2 b_1 b_0	---	Not allocated.. Cannot be used as data memory.
					72H	---	b_3 b_2 b_1 b_0	---	
					73H	---	b_3 b_2 b_1 b_0	---	

Table 17-2 Port (pin) to port register relation (2/2)

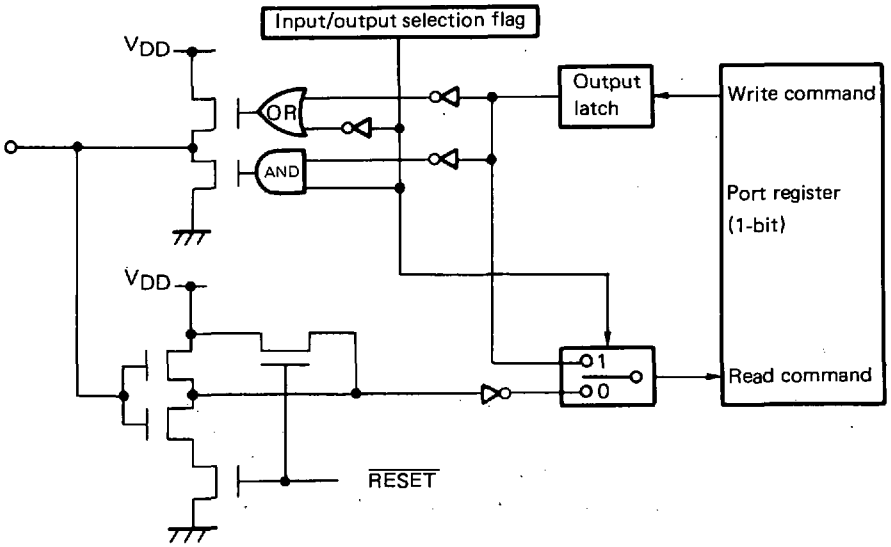
Port	Pin			Data setting							
				Port register (data memory)				Port group register (Peripheral register)			
	No.	Symbol	Input/ output	Bank	Address	Symbol	Bit symbol (reserved word)	Peripheral address	Symbol (reserved word)	Bit
				BANK3	70H	---	b_3 ---	Not allocated. Cannot be used as data memory.		
							b_2			
							b_1			
							b_0			
					71H	---	b_3 ---			
							b_2			
							b_1			
							b_0			
					72H	---	b_3 ---			
							b_2			
							b_1			
							b_0			
					73H	---	b_3 ---			
							b_2			
							b_1			
							b_0			
Port0E (P0E)	49	$P0E_3$	Output	BANK1	6BH (Combined with LCDD11.)	P0E	b_3 P0E3			
	50	$P0E_2$					b_2 P0E2			
	51	$P0E_1$					b_1 P0E1			
	52	$P0E_0$					b_0 P0E0			
Port0F (P0F)	45	$P0F_3$	Output		6DH (Combined with LCDD13.)	P0F	b_3 P0F3			
	46	$P0F_2$					b_2 P0F2			
	47	$P0F_1$					b_1 P0F1			
	48	$P0F_0$					b_0 P0F0			
Port0X (P0X)	53	$P0X_5$	Output		69H (Combined with LCDD9.)	P0XH	b_3 P0XH3	0CH (Combined with LCDR4.)	P0X	b_7
	54	$P0X_4$					b_2 P0XH2			b_6
	55	$P0X_3$					b_1 P0XH1			b_5
	56	$P0X_2$					b_0 P0XH0			b_4
	57	$P0X_1$			68H (Combined with LCDD8.)	P0XL	b_3 P0XL3			b_3
	58	$P0X_0$					b_2 P0XL2			b_2
	No pin						b_1 P0XL1			any
							b_0 P0XL0			
Port0Y (P0Y)	59	$P0Y_{15}$	Output					42H (Combined with KSR.)	P0Y	b_{15}
	60	$P0Y_{14}$								b_{14}
	61	$P0Y_{13}$								b_{13}
	|	|							|
	72	$P0Y_2$								b_2
	73	$P0Y_1$								b_1
	74	$P0Y_0$								b_0

17.3 GENERAL-PURPOSE INPUT/OUTPUT PORTS (P0A, P0B, P0C, AND P1A)

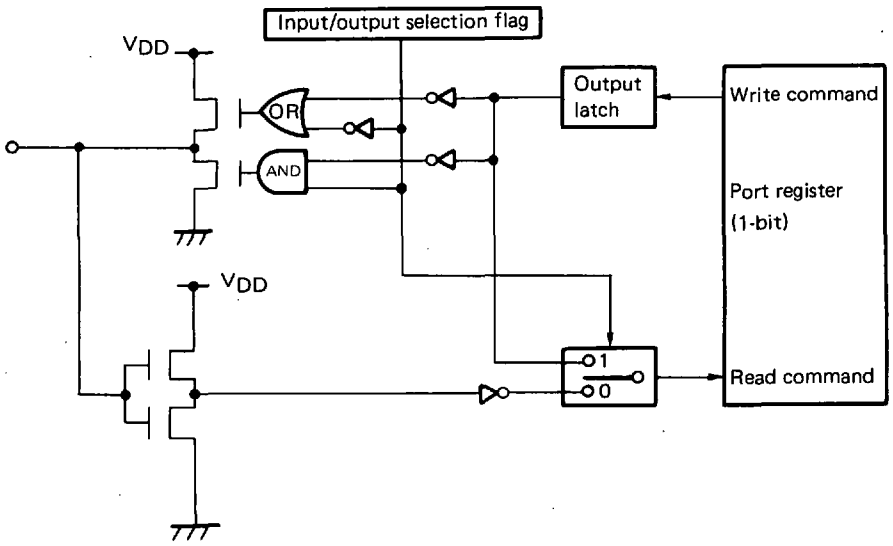
17.3.1 Input/Output Port Configuration

The input/output port configuration is shown below.

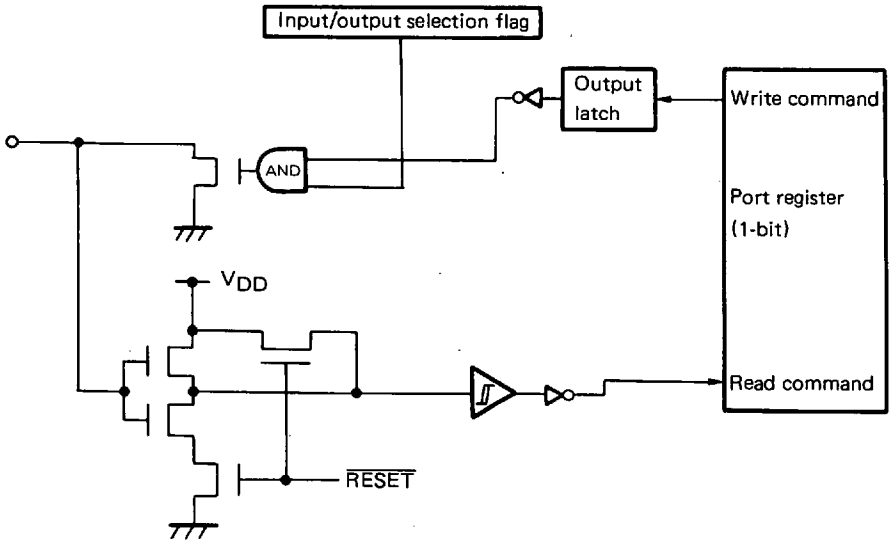
(1) P0A ($P0A_1$ and $P0A_0$ pins)
P0B ($P0B_3$, $P0B_2$, $P0B_1$, and $P0B_0$ pins)
P1A ($P1A_3$, $P1A_2$, $P1A_1$, and $P1A_0$ pins)



(2) P0C ($P0C_3$, $P0C_2$, $P0C_1$, and $P0C_0$ pins)



(3) P0A (P0A$_3$ and P0A$_2$ pins)



17.3.2 Use of Input/Output Port

The input and output at the input/output port are set using P0A, P0B, P0C, and P1A I/O select registers of a control register. The input and output at bit I/O ports (P0A, P0B, and P1A) can be set in one-bit units. The input and output at a group I/O port (P0C) can be set in four bits.

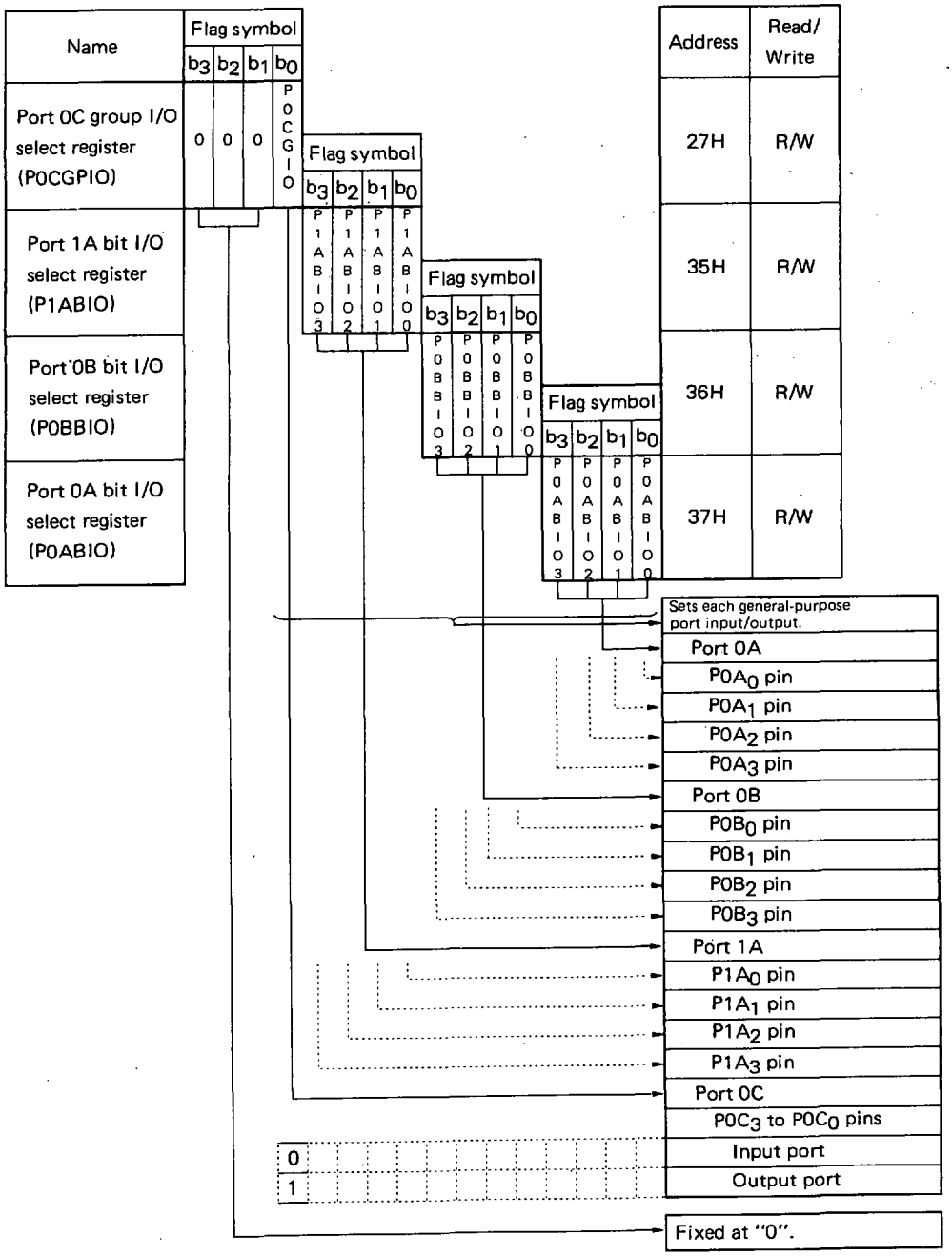
The output data is set when data is written in the corresponding port register. The input data is read when a data read command is executed.

The port I/O select register is described in Section **17.3.3.** Sections **17.3.4** and **17.3.5** describe how to use the input/output port as input and output ports.

17.3.3 Port 0A Bit I/O Select Register (P0ABIO)
Port 0B Bit I/O Select Register (P0ABIO)
Port 1A Bit I/O Select Register (P1ABIO)
Port 0C Group I/O Select Register (P0CGPIO)

The port 0A bit I/O, port 0B bit I/O, port 1A bit I/O, and port 0C group I/O select registers set each P0A, P0B, P1A, and P0C pin input/output.

The configuration and functions are shown below.



Reset	Power on	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Clock stop				0	0	0	0	0	0	0	0	0	0	0	0
	CE				0	0	0	0	0	0	0	0	0	0	0	0

17.3.4 Input/Output Ports (P0A, P0B, P0C, and P1A) Used as Input Port

A pin that is used as input is selected using a port I/O select register. The input and output at port P0C can be set in four bits.

The pin that is specified as an input port enters the floating (high impedance) state and the system waits until an external signal is input. The input data is read when a command (e.g., "SKT" command) that reads the information of a port register corresponding to each pin is executed. Logical "1" is read when a high signal is input to each pin, and logical "0" is read when a low signal is input. The output latch information is rewritten when a write command (e.g., "MOV" command) is executed for the port register specified as an input port.

17.3.5 Input/Output Ports (P0A, P0B, P0C, and P1A) Used as Output Port

A pin that is used as output is selected using a port I/O select register. The input and output at port P0C can be set in four bits.

The pin that is specified as an output port outputs the output latch information from each pin. The output data is set when a command (e.g., "MOV" command) that is written in the information of a port register corresponding to each pin is executed. Logical "1" is written when a high signal is output to each pin, and logical "0" is written when a low signal is output. When the port register is specified as an input port, the floating state can be entered. The output latch information is read when a read command (e.g., "SKT" command) is executed for the port register specified as an output port.

The $P0A_3$ and $P0A_2$ pin states are read as they are, so the output latch information may differ from the read information. For more information, see Section **17.3.6.**

17.3.6 Cautions when Using Input/Output Ports ($P0A_3$ and $P0A_2$ Pins)

The output latch information may be rewritten when the $P0A_3$ and $P0A_2$ pins are used as an output as described below.

Example:

```
INITFLG  P0ABIO3, P0ABIO2, NOT P0ABIO1, NOT P0ABIO0
                ; The P0A3 and P0A2 pins are set to the output.
INITFLG  P0A3, P0A2, NOT P0A1, NOT P0A0
                ; A high signal is output to the P0A3 and P0A2 pins.
; ①
CLR1   P0A3    ; A low signal is output to the P0A3 pin.
Macroexpansion
AND    .MF.  P0A3 SHR 4, #. DF. (NOT P0A3 AND 0FH)
```

The output latch information at the $P0A_2$ pin is rewritten in "0" using a "CLR1" command if the $P0A_2$ pin is pulled low externally when a command in parameter ① above is executed.

17.3.7 Input/Output Port (P0A, P0B, P0C, and P1A) State During Reset

(1) State during power on reset

All is specified for an input port.

The output latch information is "undefined", so it must be initialized using a program as required before selected to an output port.

(2) State during CE reset

All is specified for an input port.

The output latch information is held.

(3) State during clock stop

All is specified for an input port.

The output latch information is held.

As described in Section **17.3.1,** input/output ports other than P0C prevent the power consumption from increasing owing to the noise in an input buffer using a $\overline{\text{RESET}}$ signal output during the clock stop. The power consumption may increase owing to external noise when the P0C input/output port is in floating state during the clock stop. Pull up or pull down the P0C port externally as required.

(4) State in halt mode

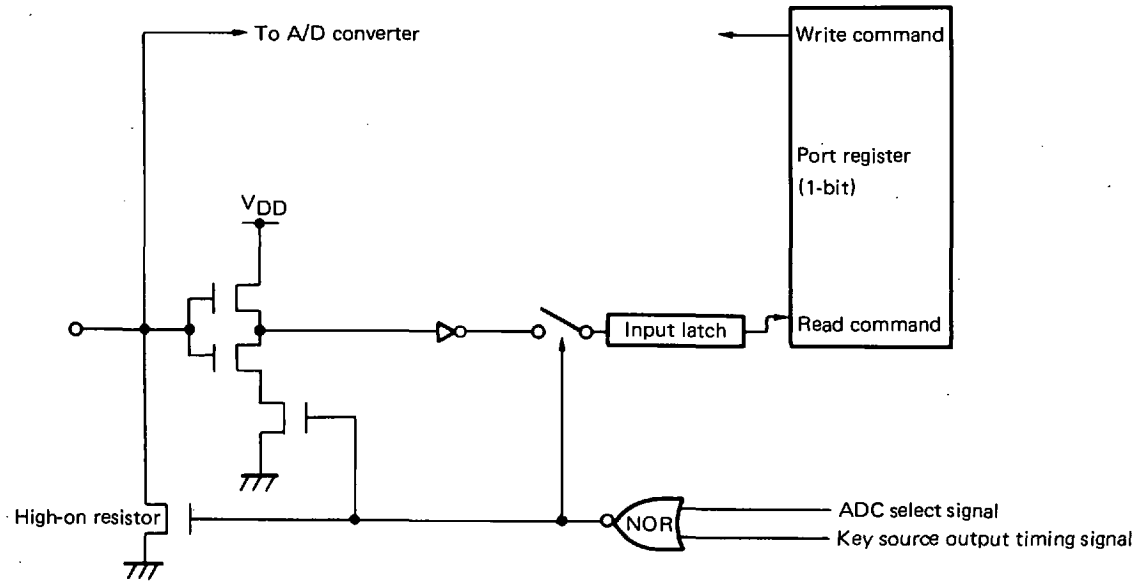
The former state is held.

17.4 GENERAL-PURPOSE INPUT PORTS (P0D AND P1D)

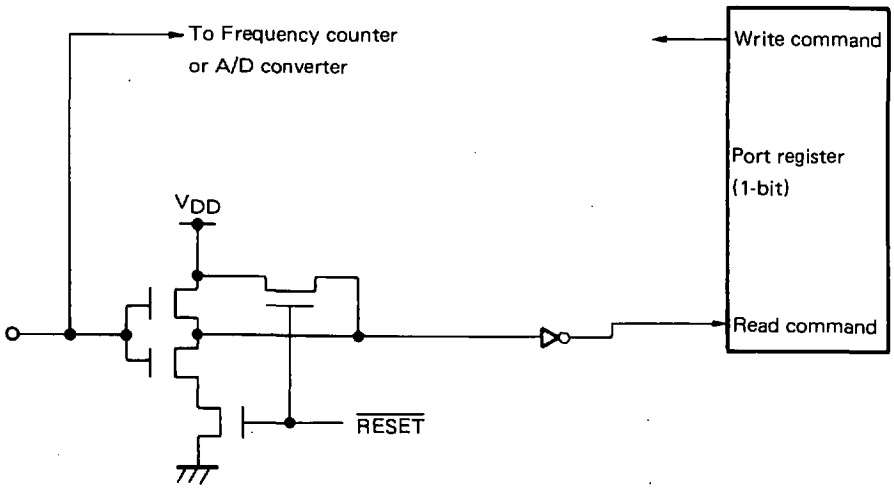
17.4.1 Input Port Configuration

The input port configuration is shown below.

(1) P0D ($P0D_3$, $P0D_2$, $P0D_1$, and $P0D_0$ pins)



(2) P1D ($P1D_3$, $P1D_2$, $P1D_1$, and $P1D_0$ pins)



17.4.2 Use of Input Ports (P0D and P1D)

The input data is read when a command (e.g., "SKT" command) that reads the information of a port register corresponding to each pin is executed. Logical "1" is read when a high signal is input to each pin, and logical "0" is read when a low signal is input. Nothing changes even if a write command (e.g., "MOV" command) is executed for the port register.

17.4.3 Cautions when Using Input Port (P0D)

The P0D input is pulled down internally when it is used as a general-purpose port.

17.4.4 Input Port (P0D and P1D) State During Reset

(1) State during power on reset

All is specified for a general-purpose input port.

(2) State during CE reset

All is specified for a general-purpose input port.

(3) State during clock stop

All is specified for a general-purpose input port.

A $\overline{\text{RESET}}$ signal is output during the clock stop. Therefore, the P1D input port prevents the power consumption from increasing owing to the noise in an input buffer as described in Section **17.4.1.** The P0D input port is pulled down internally.

(4) State in halt mode

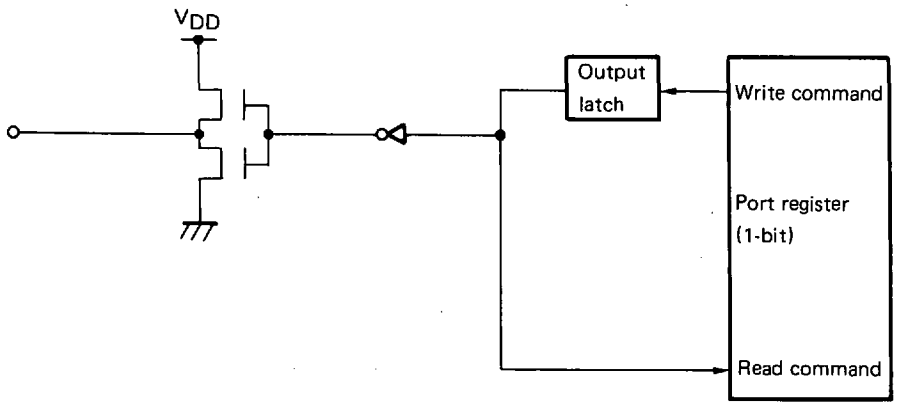
The former state is held.

17.5 GENERAL-PURPOSE OUTPUT PORTS (P1B, P1C, AND P2A)

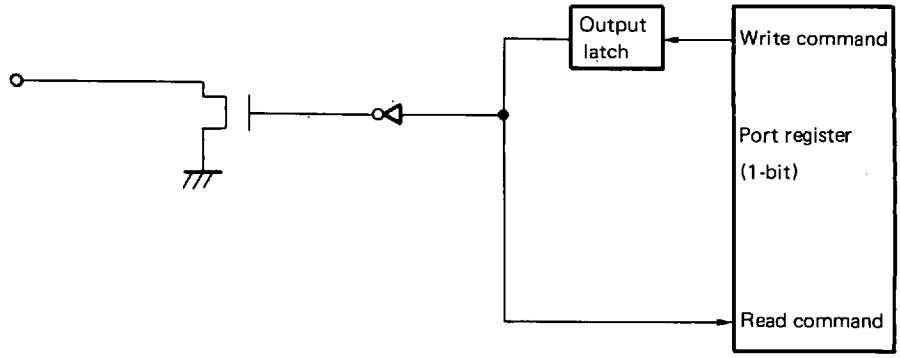
17.5.1 Output Port (P1B, P1C, and P2A) Configuration

The output port configuration is shown below.

(1) P1B ($P1B_0$ pin)
P1C ($P1C_3$, $P1C_2$, $P1C_1$, and $P1C_0$ pins)
P2A ($P2A_0$ pin)



(2) P1B ($P1B_3$, $P1B_2$, and $P1B_1$ pins)



17.5.2 Use of Output Ports (P1B, P1C, and P2A)

The output port outputs the output latch information from each pin.

The output data is set when a command (e.g., "MOV" command) that is written in the information of a port register corresponding to each pin is executed. Logical "1" is written when a high signal is output to each pin, and "0" is written when a low signal is output. The $P1B_3$, $P1B_2$, and $P1B_1$ pins are output at an N-channel open drain. The floating state is thus entered when a high signal is output. The output latch information is read when a port register read command (e.g., "SKT" command) is executed.

17.5.3 Output Port (P1B, P1C, and P2A) State During Reset

(1) State during power on reset

The output latch information is output.

The output latch information is "undefined". An "undefined" value is thus output for a fixed time (until the output latch information is initialized using a program).

(2) State during CE reset

The output latch information is output.

The output latch information is held, so no output data changes during the CE reset.

(3) State during clock stop

The output latch information is output.

The output latch information is held, so no output data changes during the clock stop. Initialize the output latch information using a program as required.

(4) State in halt mode

The output latch information is output.

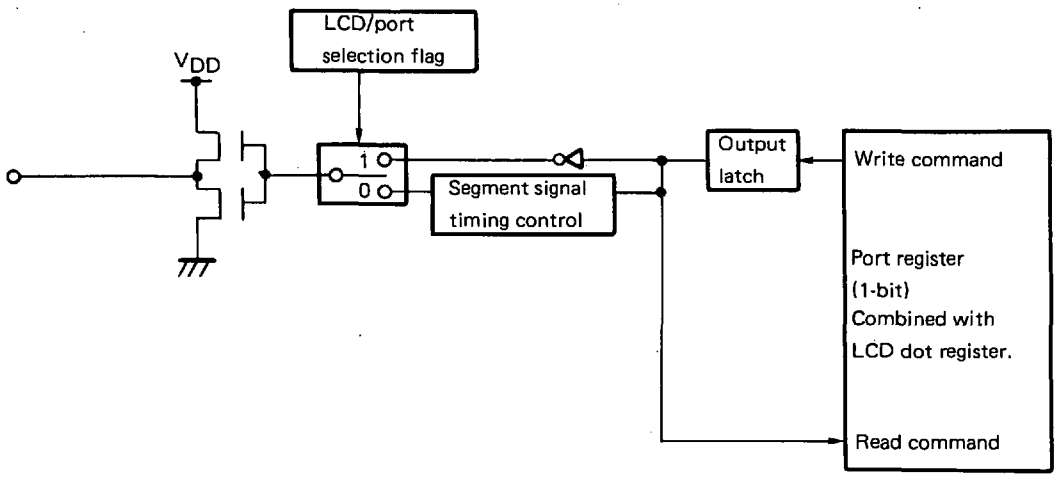
The output latch information is held, so no output data changes in halt mode.

17.6 GENERAL-PURPOSE OUTPUT PORTS (P0E, P0F, P0X, and P0Y)

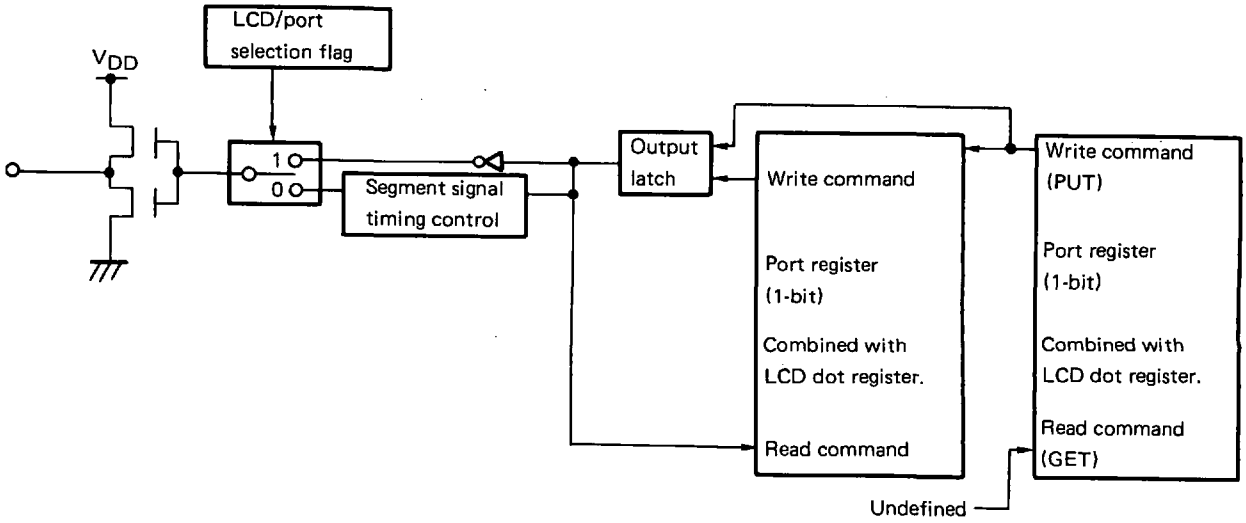
17.6.1 Output Port (P0E, P0F, P0X, and P0Y) Configuration

The output port (P0E, P0F, P0X, and P0Y) configuration is shown below.

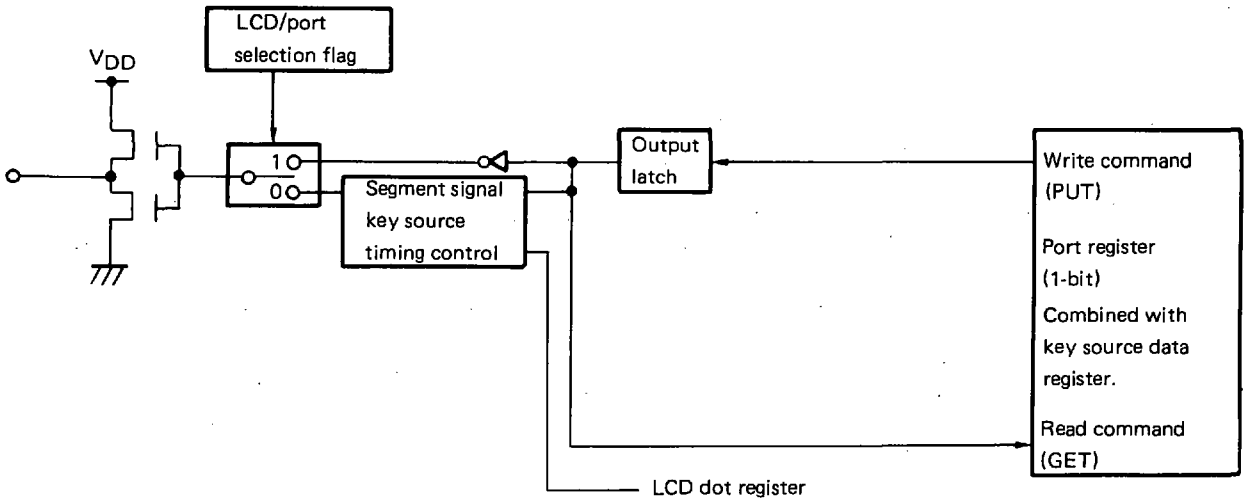
(1) P0E ($P0E_3$, $P0E_2$, $P0E_1$, and $P0E_0$ pins)
P0F ($P0F_3$, $P0F_2$, $P0F_1$, and $P0F_0$ pins)



(2) P0X ($P0X_5$ through $P0X_0$ pins)



(3) P0Y ($P0Y_{15}$ through $P0Y_0$ pins)



17.6.2 Use of Output Ports (P0E, P0F, P0X, and P0Y)

The P0E, P0F, P0X, and P0Y pins are set as LCD segment signal outputs during the power on reset. Therefore, the output port used is selected using an LCD port select register. Each P0E, P0F, P0X, and P0Y port can be selected independently. The pin that is not set as an output port using the LCD port select register can be used as an LCD segment signal output.

Setting the P0E, P0F, P0X, and P0Y output data is described in Sections **17.6.3** through **17.6.5.** The configuration and functions of the LCD port select register and port group register are described in Sections **17.6.6** and **17.6.7.**

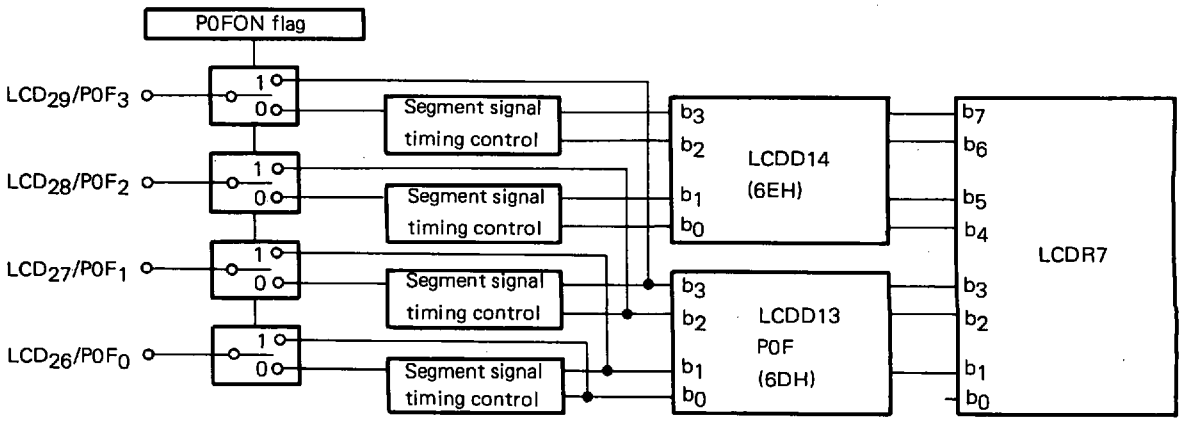
17.6.3 P0E and P0F Data Setting

The P0E and P0F output data is set when a command (e.g., "MOV" command) that is written in the information of a port register corresponding to each pin is executed. Logical "1" is written when a high signal is output to each pin, and logical "0" is written when a low signal is output. The output latch information is read when a port register read command (e.g., "SKT" command) is executed.

Fig. 17-3 shows the relation between the P0F port register, LCD dot register, and LCD group register. As shown in Fig. 17-3, LCD dot register LCDD14 can be used as general-purpose data memory when the P0F output port is used. The high-order three bits of the P0F output port change when data is set to LCD group register LCDR7. The P0E port also changes.

For more information, see Fig. 23-6, **"Relation between LCD Display Dot, P0E through P0Y, Key Source Output, and Data Setting Register"** in Section **23, "LCD Controller/Driver".**

Fig. 17-3 Relation between POF port register, LCD dot register, and LCD group register



17.6.4 P0X Data Setting

The P0X output data is set using a port register or port group register.

A command (e.g., "MOV" command) that is written in the information of port registers (P0XH and P0XL) corresponding to each pin is executed when the port register is used. Logical "1" is written when a high signal is output to each pin, and logical "0" is written when a low signal is output. The output latch information is read when a port register read command (e.g., "SKT" command) is executed.

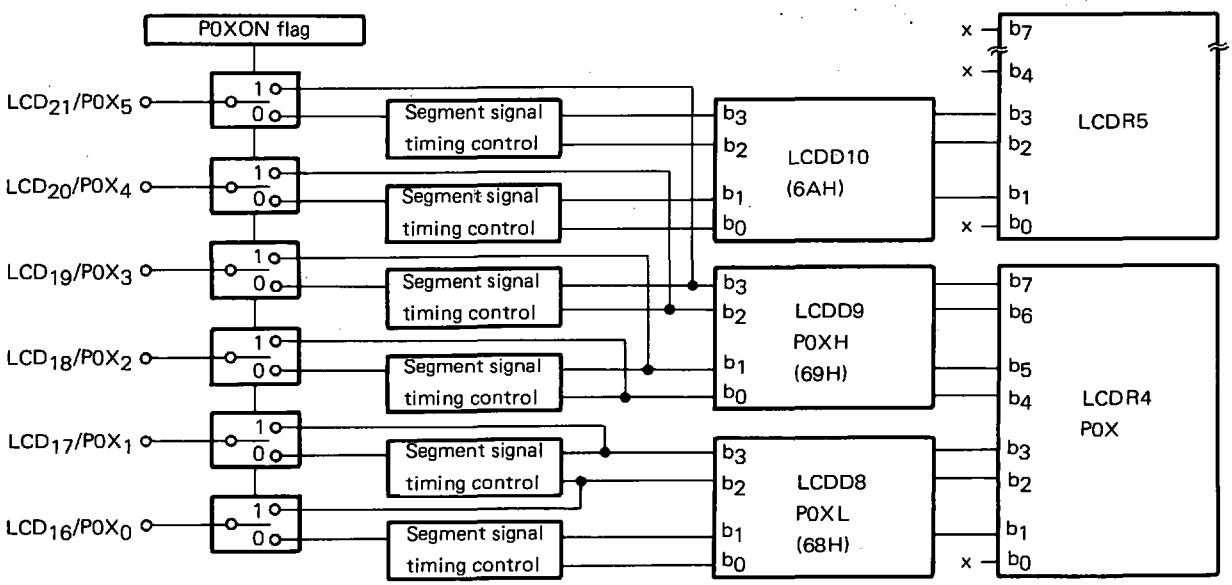
A "PUT P0X, DBF" command that is written in the information of a port group register (P0X) corresponding to each pin is executed when the port group register is used. A "undefined value" is read when a "GET DBF, P0X" command that reads the port group register (P0X) information is executed. When data is set using the port group register, logical "1" is written when a high signal is output to each pin. Logical "0" is written when a low signal is output.

Fig. 17-4 shows the relation between the P0X port register, port group register, LCD dot register, and LCD group register.

As shown in Fig. 17-4, LCD dot register LCDD10 can be used as general-purpose data memory when the P0X port is used.

For more information, see Fig. 23-6, "Relation between LCD Display Dot, P0E through P0Y, Key Source Output, and Data Setting Register" in Section **23, "LCD Controller/Driver".**

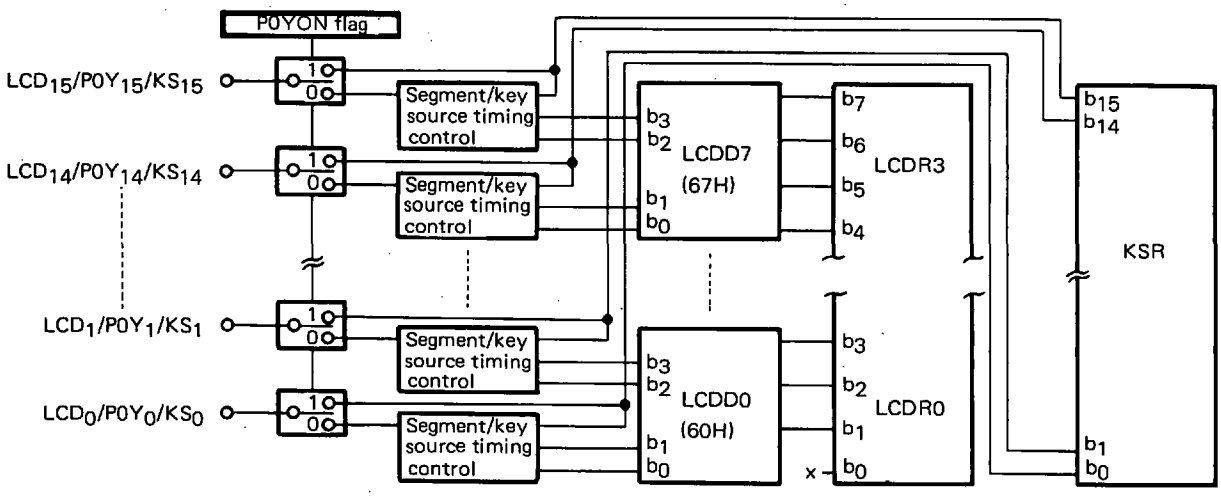
Fig. 17-4 Relation between P0F port register, port group register, LCD dot register, and LCD group register



17.6.5 P0Y Data Setting

The P0Y output data is set when a "PUT P0Y, DBF" command that is written in the information of a port group register (P0Y, combined with key source data register) corresponding to each pin is executed. The output latch information is read when a "GET DBF, P0Y" command that reads the port group register information is executed. Logical "1" is written when a high signal is output to each pin, and logical "0" is written when a low signal is output.

Fig. 17-5 Relation between P0Y port register, LCD dot register, and LCD group register



17.6.6 LCD Port Select Register (LCDPORT)

The LCD port select register sets the LCD segment signal output pin for a general-purpose output port.

The LCD port select register configuration and functions are shown below.

Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
LCD port select register (LCDPORT)	P0YON	P0XON	P0EON	P0FON	11H	R/W

	Sets the LCD segment signal output pin for a general-purpose output port.
P0FON	$LCD_{26}/P0F_0$ – $LCD_{29}/P0F_3$ pin (port 0F)
P0EON	$LCD_{22}/P0E_0$ – $LCD_{25}/P0E_3$ pin (port 0E)
P0XON	$LCD_{16}/P0X_0$ – $LCD_{21}/P0X_5$ pin (port 0X)
P0YON	$LCD_0/P0Y_0/KS_0$ – $LCD_{15}/P0Y_{15}/KS_{15}$ pin (port 0Y)
0	LCD segment signal output
1	General-purpose output port

Reset		b_3	b_2	b_1	b_0
	Power on	0	0	0	0
	Clock stop	0	0	0	0
	CE	0	0	0	0

Port 0F, port 0E, port 0X, and port 0Y can be set independently for a general-purpose output port. The pin that is not set for the general-purpose output port operates as an LCD segment signal output.

The $LCD_0/P0Y_0/KS_0$ through $LCD_{15}/P0Y_{15}/KS_{15}$ pins can be used in combination with an LCD segment signal output and key source signal output. The LCD segment and key source signals are not output when the pins are set to the general-purpose output port.

17.6.7 Port 0X (P0X) Group Register
Port 0Y (P0Y) Group Register

The P0X and P0Y group register functions are shown below.

The P0X and P0Y group data registers set the P0X ($P0X_0$ through $P0X_5$ pins) and P0Y ($P0Y_0$ through $P0Y_{15}$ pins) output data. The P0X and P0Y group registers can also set 6- and 16-bit output data at a time.

(1) P0X group register functions

Name	Data buffer			
Symbol	DBF 3	DBF 2	DBF 1	DBF 0
Address	0 CH	0 DH	0 EH	0 FH
Bit	b_{15} b_{14} b_{13} b_{12} b_{11} b_{10} b_9 b_8		b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0	
Data	Any	Any	Transfer data	

8

GET is undefined data.
A PUT command can be entered.

Peripheral register										
Name	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Symbol	Peripheral address	Peripheral hardware
P0X group register	Valid data						–	–	P0X	0 CH	Port 0 X

Any	
Sets the port 0X output data.	
$LCD_{16}/P0X_0$ pin	
$LCD_{17}/P0X_1$ pin	
$LCD_{18}/P0X_2$ pin	
$LCD_{19}/P0X_3$ pin	
$LCD_{20}/P0X_4$ pin	
$LCD_{21}/P0X_5$ pin	
0	Low level output
1	High level output

Port 0X can set the output data using a P0X group register (peripheral address 0CH), and P0XH and P0XL port registers (data memory addresses, BANK0 69H and 68H). The P0XH and P0XL register (port register) data corresponding to the overlapped data bit also becomes the same value when data is set to the P0X group register (peripheral register).

(2) P0Y group register functions

Name	Data buffer			
Symbol	DBF 3	DBF 2	DBF 1	DBF 0
Address	0 CH	0 DH	0 EH	0 FH
Bit	b_{15} b_{14} b_{13} b_{12}	b_{11} b_{10} b_9 b_8	b_7 b_6 b_5 b_4	b_3 b_2 b_1 b_0
Data	Transfer data			

16

GET and PUT commands can be entered.

Peripheral register			
Name	Symbol	Peripheral address	Peripheral hardware
P0Y group register (Valid data)	P0Y	42 H	Port 0 Y

	Sets the port 0Y output data.
	$LCD_0/P0Y_0/KS_0$ pin
	$LCD_1/P0Y_1/KS_1$ pin
	$LCD_2/P0Y_2/KS_2$ pin
	$LCD_3/P0Y_3/KS_3$ pin
	$LCD_4/P0Y_4/KS_4$ pin
	$LCD_5/P0Y_5/KS_5$ pin
	$LCD_6/P0Y_6/KS_6$ pin
	$LCD_7/P0Y_7/KS_7$ pin
	$LCD_8/P0Y_8/KS_8$ pin
	$LCD_9/P0Y_9/KS_9$ pin
	$LCD_{10}/P0Y_{10}/KS_{10}$ pin
	$LCD_{11}/P0Y_{11}/KS_{11}$ pin
	$LCD_{12}/P0Y_{12}/KS_{12}$ pin
	$LCD_{13}/P0Y_{13}/KS_{13}$ pin
	$LCD_{14}/P0Y_{14}/KS_{14}$ pin
	$LCD_{15}/P0Y_{15}/KS_{15}$ pin
0	Low level output
1	High level output

Port 0Y is used in combination with a key source signal output pin. The P0Y group register (peripheral address 42H) is therefore used in combination with a key source data register (peripheral address 42H) described later. The P0Y group register sets the port 0Y output data when the $LCD_0/P0Y_0/KS_0$ through $LCD_{15}/P0Y_{15}/KS_{15}$ pins are specified for an output port and sets the key source signal output when they are specified for a key source signal output pin.

17.6.8 Output Port (P0E, P0F, P0X, and P0Y) State During Reset

(1) State during power on reset

The output ports are set to an LCD segment signal output pin to output a low signal.

The output latch information is undefined, so undefined data is output when the output latch is set to the output ports. Initialize using a program as required.

(2) State during CE reset

The output ports are set to an LCD segment signal output pin to output a low signal.

The output latch information is held, so the former state is held when the output latch is set to the output ports.

(3) State during clock stop

The output ports are set to an LCD segment signal output pin to output a low signal.

The output latch information is held, so the former state is held when the output latch is set to the output ports.

(4) State in halt mode

The output latch information is output.

The output latch information is held, so no output data changes in halt mode.

18. A/D CONVERTER (ADC)

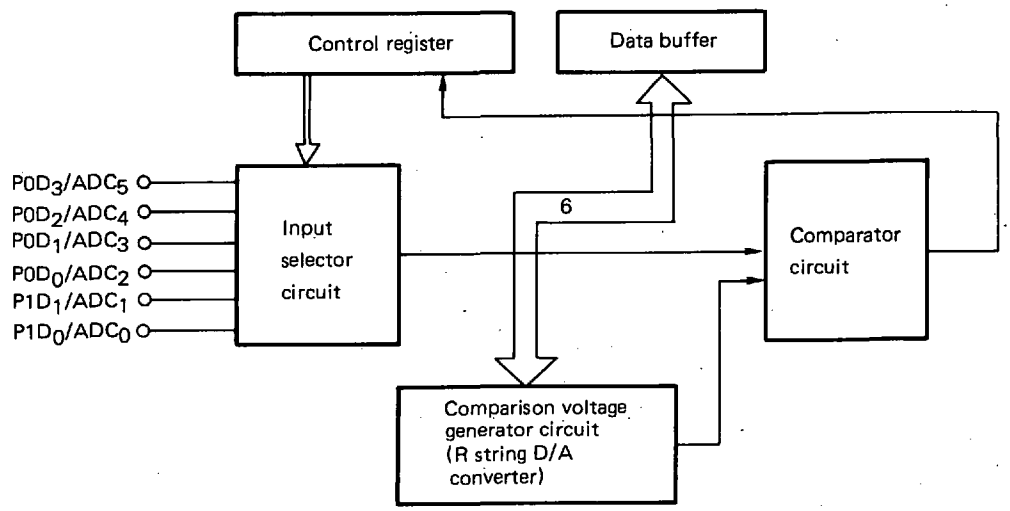
The A/D converter is used to load the external analog voltage as a digital signal.

18.1 A/D CONVERTER CONFIGURATION

Fig. 18-1 shows the A/D converter block diagram.

As shown in Fig. 18-1, the A/D converter consists of an input selector circuit, comparison voltage generator circuit, and comparator circuit.

Fig. 18-1 A/D Converter block diagram



18.2 A/D CONVERTER FUNCTIONS

The A/D converter compares the input voltage at $P0D_3/ADC_5$ through $P1D_0/ADC_0$ pins and the internal comparison voltage and outputs the compared voltage in "true (1)" or "false (0)". This compared result can be used as a successive comparison A/D converter when it is determined by software.

Each block function is described below. For more details, see Sections **18.3** through **18.5.**

18.2.1 Input Selector Circuit

The input selector circuit selects which $P0D_3/ADC_5$ through $P1D_0/ADC_0$ pins should be used. These pins are selected using an A/D converter channel select register (RF address 14H). Only one pin can be used at the same time.

For more details, see Section **18.3.**

18.2.2 Comparison Voltage Generator Circuit

The comparison voltage generator circuit generates a comparison voltage that is compared with the input voltage.

The comparison voltage is generated using an R string D/A converter. For more details, see Section **18.4.**

18.2.3 Comparator Circuit

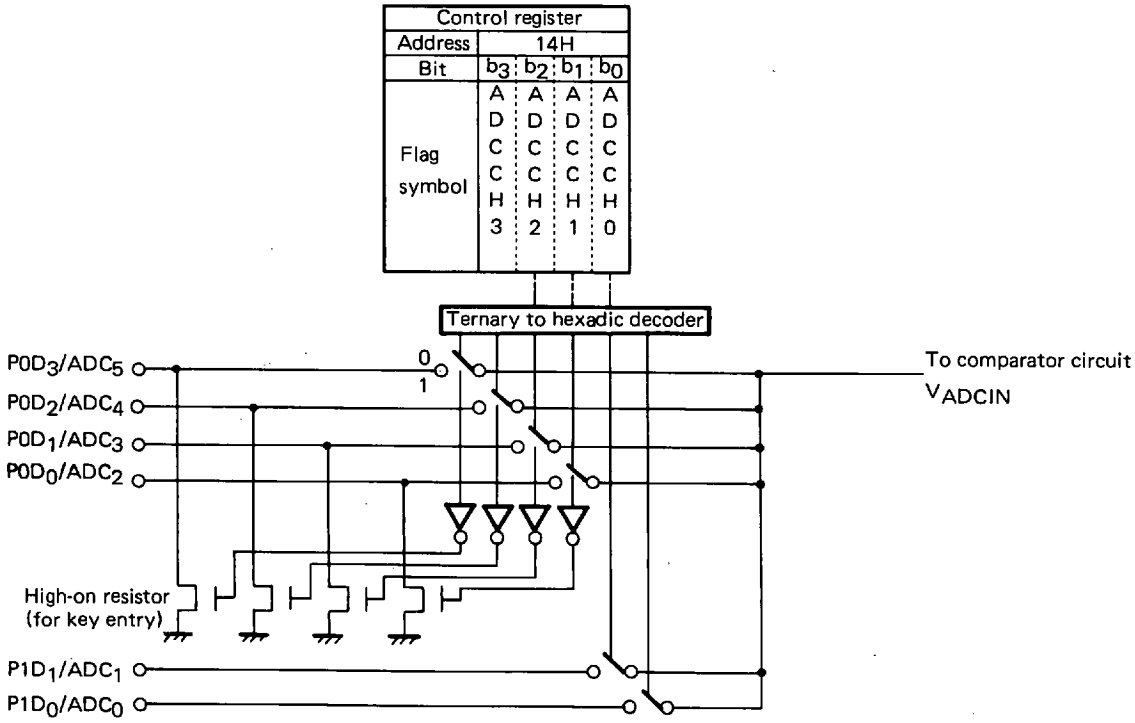
The comparator circuit compares the input voltage and internal comparison voltage. The compared voltage is detected using an A/D converter compare judge register (RF address 06H). For more details, see Section **18.5.**

18.3 INPUT SELECTOR CIRCUIT

18.3.1 Input Selector Circuit Configuration

Fig. 18-2 shows the input selector circuit configuration.

Fig. 18-2 Input selector circuit configuration



18.3.2 Input Selector Circuit Functions

The input selector circuit selects the pin to be used by an A/D converter channel select register. Only one pin can be used as an A/D converter at the same time. The pin that is not selected for the A/D converter can be used as a general-purpose input port.

Port 0D ($P0D_3/ADC_5$ through $P0D_2/ADC_2$ pins) has an internal pull-down resistor. The pull-down resistor is turned off when the pins are selected using the A/D converter channel select register. The pull-down resistor remains on when the pins are not selected.

The A/D converter channel select register configuration and functions are described below.

18.3.3 A/D Converter Channel Select Register (ADCCH)

The A/D converter channel select register sets the pin used as an A/D converter.

The A/D converter channel select register configuration and functions are shown below.

Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
A/D converter channel select register (ADCCH)	ADCCH3	ADCCH2	ADCCH1	ADCCH0	14H	R/W

Sets the pin used as an A/D converter.			
0	0	0	$P1D_0/ADC_0$ pin
0	0	1	$P1D_1/ADC_1$ pin
0	1	0	$P0D_0/ADC_2$ pin
0	1	1	$P0D_1/ADC_3$ pin
1	0	0	$P0D_2/ADC_4$ pin
1	0	1	$P0D_3/ADC_5$ pin
1	1	0	No A/D converter is used. (General-purpose input port)
1	1	1	No A/D converter is used. (General-purpose input port)

Fixed at "0".

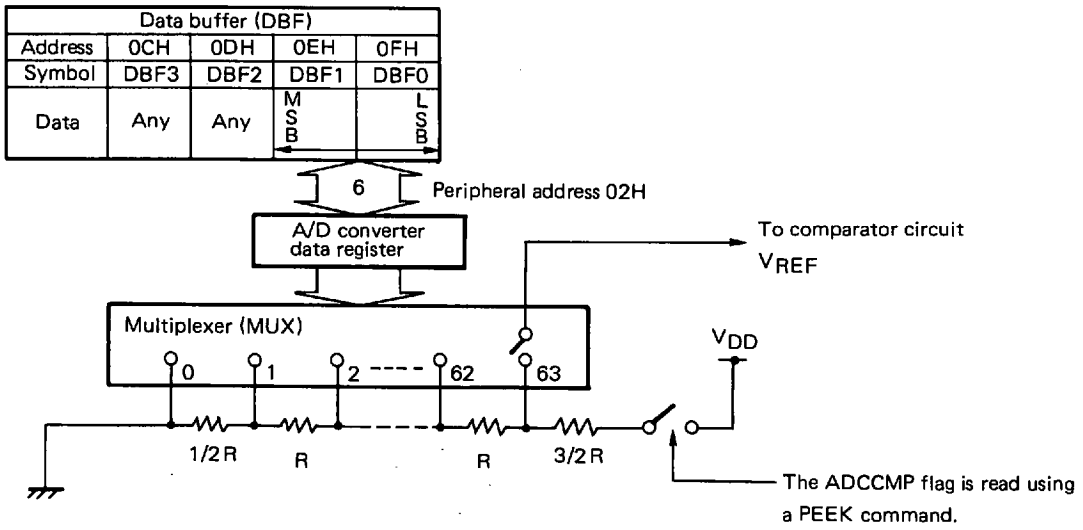
Reset					
	Power on	0	1	1	1
	Clock stop		1	1	1
	CE	↓	Held		

18.4 COMPARISON VOLTAGE GENERATOR CIRCUIT

18.4.1 Comparison Voltage Generator Circuit Configuration

Fig. 18-3 shows the comparison voltage generator circuit configuration.

Fig. 18-3 Comparison voltage generator circuit configuration



18.4.2 Comparison Voltage Generator Circuit Functions

The comparison voltage generator circuit selects a multiplexer (MUX) using the 6-bit data that is set in an A/D converter data register (ADCR, peripheral address 02H), generating the comparison voltage using an R string D/A converter. Up to the 64-step comparison voltage can be set in accordance with an R string system. Power to the R string system is the same as device supply voltage V_{DD}.

The voltage applied to an R string resistor is supplied only when an A/D converter compare judge register (RF address 06H) described later is detected. The comparison voltage is compared with the voltage that is input from each pin to a comparator circuit.

The A/D converter data register configuration and functions are described in Section **18.4.3.** Table 18-1 shows the comparison voltage list.

18.4.3 A/D Converter Data Register (ADCR) Configuration and Functions

The A/D converter data register sets the comparison voltage of an A/D converter.

The A/D converter data register consists of six bits. The low-order six bits of a data buffer thus become valid.

Name	Data buffer			
Symbol	DBF 3	DBF 2	DBF 1	DBF 0
Address	0 CH	0 DH	0 EH	0 FH
Bit	b_{15} b_{14} b_{13} b_{12}	b_{11} b_{10} b_9 b_8	b_7 b_6 b_5 b_4	b_3 b_2 b_1 b_0
Data	Any	Any	Transfer data	

8

GET and PUT commands can be entered.

Peripheral register										
Name	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Symbol	Peripheral address	Peripheral hardware
A/D converter data register	0	0	Valid data						ADCR	02 H	A/D converter

Sets comparison voltage V_{REF} of an A/D converter.

0	$V_{REF} = 0$ V
1	$V_{REF} = \frac{x - 0.5}{64} \times V_{DD}$ V
x	
63	

Fixed at "0".

Table 18-1 A/D converter data register setting value and comparison voltage

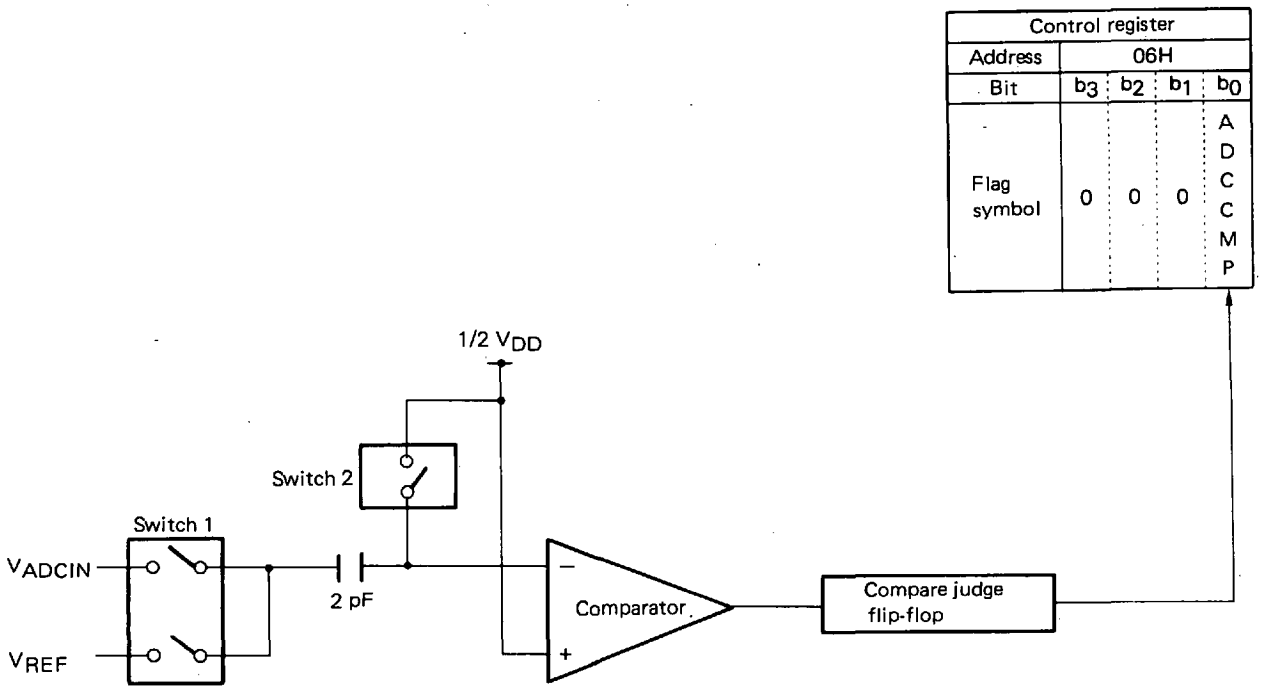
ADCR setting data		Comparison voltage		ADCR setting data		Comparison voltage	
Decimal (DEC)	Hexadecimal (HEX)	Logical voltage Unit: x V_{DD} V	V_{DD} = 5 V Unit: V	Decimal (DEC)	Hexadecimal (HEX)	Logical voltage Unit: x V_{DD} V	V_{DD} = 5 V Unit: V
0	00H	0	0	32	20H	31.5/54	2.461
1	01H	0.5/64	0.039	33	21H	32.5/64	2.539
2	02H	1.5/64	0.117	34	22H	33.5/64	2.617
3	03H	2.5/64	0.195	35	23H	34.5/64	2.695
4	04H	3.5/64	0.273	36	24H	35.5/64	2.773
5	05H	4.5/64	0.352	37	25H	36.5/64	2.852
6	06H	5.5/64	0.430	38	26H	37.5/64	2.930
7	07H	6.5/64	0.508	39	27H	38.5/64	3.008
8	08H	7.5/64	0.586	40	28H	39.5/64	3.086
9	09H	8.5/64	0.664	41	29H	40.5/64	3.164
10	0AH	9.5/64	0.742	42	2AH	41.5/64	3.242
11	0BH	10.5/64	0.820	43	2BH	42.5/64	3.320
12	0CH	11.5/64	0.898	44	2CH	43.5/64	3.398
13	0DH	12.5/64	0.977	45	2DH	44.5/64	3.477
14	0EH	13.5/64	1.055	46	2EH	45.5/64	3.555
15	0FH	14.5/64	1.133	47	2FH	46.5/64	3.633
16	10H	15.5/64	1.211	48	30H	47.5/64	3.711
17	11H	16.5/64	1.289	49	31H	48.5/64	3.789
18	12H	17.5/64	1.367	50	32H	49.5/64	3.867
19	13H	18.5/64	1.445	51	33H	50.5/64	3.945
20	14H	19.5/64	1.523	52	34H	51.5/64	4.023
21	15H	20.5/64	1.602	53	35H	52.5/64	4.102
22	16H	21.5/64	1.680	54	36H	53.5/64	4.180
23	17H	22.5/64	1.758	55	37H	54.5/64	4.258
24	18H	23.5/64	1.836	56	38H	55.5/64	4.336
25	19H	24.5/64	1.914	57	39H	56.5/64	4.414
26	1AH	25.5/64	1.992	58	3AH	57.5/64	4.492
27	1BH	26.5/64	2.070	59	3BH	58.5/64	4.570
28	1CH	27.5/64	2.148	60	3CH	59.5/64	4.648
29	1DH	28.5/64	2.227	61	3DH	60.5/64	4.727
30	1EH	29.5/64	2.305	62	3EH	61.5/64	4.805
31	1FH	30.5/64	2.383	63	3FH	62.5/64	4.883

18.5 COMPARATOR CIRCUIT

18.5.1 Comparator Circuit Configuration

Fig. 18-4 shows the comparator circuit configuration.

Fig. 18-4 Comparator circuit configuration



18.5.2 Comparator Circuit Functions

The comparator circuit compares input voltage V_{ADCIN} at each pin and internal comparison voltage V_{REF} and outputs the compared voltage to a compare judge flip-flop. The compare judge flip-flop can be detected by reading an ADCCMP flag of the A/D converter compare judge register. The ADCCMP flag is set when input voltage V_{ADCIN} exceeds comparison voltage V_{REF}. It is reset when input voltage V_{ADCIN} does not exceed comparison voltage V_{REF}.

The comparator circuit comparison is performed when the ADCCMP flat is read. Switches 1 and 2 are activated for comparison when the ADCCMP flag is read using a PEEK command. One comparison time of the A/D converter corresponds to one command execution time (4.44 μs).

The A/D converter compare judge register configuration and functions are described in Section **18.5.3.**

18.5.3 A/D Converter Compare Judge Register (ADCJDG)

The A/D converter compare judge register compares input voltage V_{ADCIN} and comparison voltage V_{REF} of an A/D converter and detects the compared voltage.

The A/D converter compare judge register configuration and functions are shown below.

Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
A/D converter compare judge register (ADCJDG)	0	0	0	ADCCMP	06H	R

b_0 (ADCCMP)	
	Sets the compared voltage of an A/D converter.
0	$V_{ADCIN} < V_{REF}$
1	$V_{ADCIN} > V_{REF}$

Fixed at "0".

		b_3	b_2	b_1	b_0
Reset	Power on	0	0	0	Not held
	Clock stop				Held
	CE				Held

18.6 A/D CONVERTER PERFORMANCE

The A/D converter performance is shown in the table below.

Item	Performance
Resolution	1LSB
Input voltage range	$0 - V_{DD}$
Quantization error	$\pm\frac{1}{2}$LSB
Overrange	$\frac{62.5}{64} \times V_{DD}$
Offset, gain, and nonlinearity errors	$\pm\frac{3}{2}$LSB (see **Note**)

Note: Includes a quantization error.

18.7 USE OF A/D CONVERTER

18.7.1 Comparison with One Comparison Voltage

A program example is shown below.

Example:

Input voltage V_{ADCIN} and comparison voltage V_{REF} at ADC_0 pin are detected. The program branches into AAA when V_{ADCIN} exceeds V_{REF}. It branches into BBB when V_{REF} exceeds V_{ADCIN}.

```
INIT:
    ADCR7   FLG     0.0EH.3 ;  DUMMY
    ADCR6   FLG     0.0EH.2 ;  DUMMY
    ADCR5   FLG     0.0EH.1 ;  Each data buffer is defined as an ADCR data setting flag.
    ADCR4   FLG     0.0EH.0
    ADCR3   FLG     0.0FH.3
    ADCR2   FLG     0.0FH.2
    ADCR1   FLG     0.0FH.1
    ADCR0   FLG     0.0FH.0
    INITFLG NOT ADCCH3, NOT ADCCH2, NOT ADCCH1, NOT ADCCH0
                            ;  The P1D0/ADC0 pin is set to an A/D converter.
START:
    INITFLG NOT ADCR3, NOT ADCR2, NOT ADCR1, NOT ADCR0
    INITFLG NOT ADCR7, NOT ADCR6,       ADCR5, NOT ADCR4
    PUT     ADCR, DBF   ;  31.5/64 VDD is set to comparison voltage VREF.
    SKT1    ADCCMP      ;  The ADCCMP flag is detected.
    BR      AAA         ;  The program branches into AAA if it is false.
    BR      BBB         ;  The program branches into BBB if it is true.
```

18.7.2 Successive Comparison During Binary Search

The A/D converter can compare only one comparison voltage during one comparison. A successive comparison program must be executed to convert the input voltage into a digital signal. The processing time of the successive comparison program that varies depending on the input voltage may not be undesirable when viewed from other processing programs.

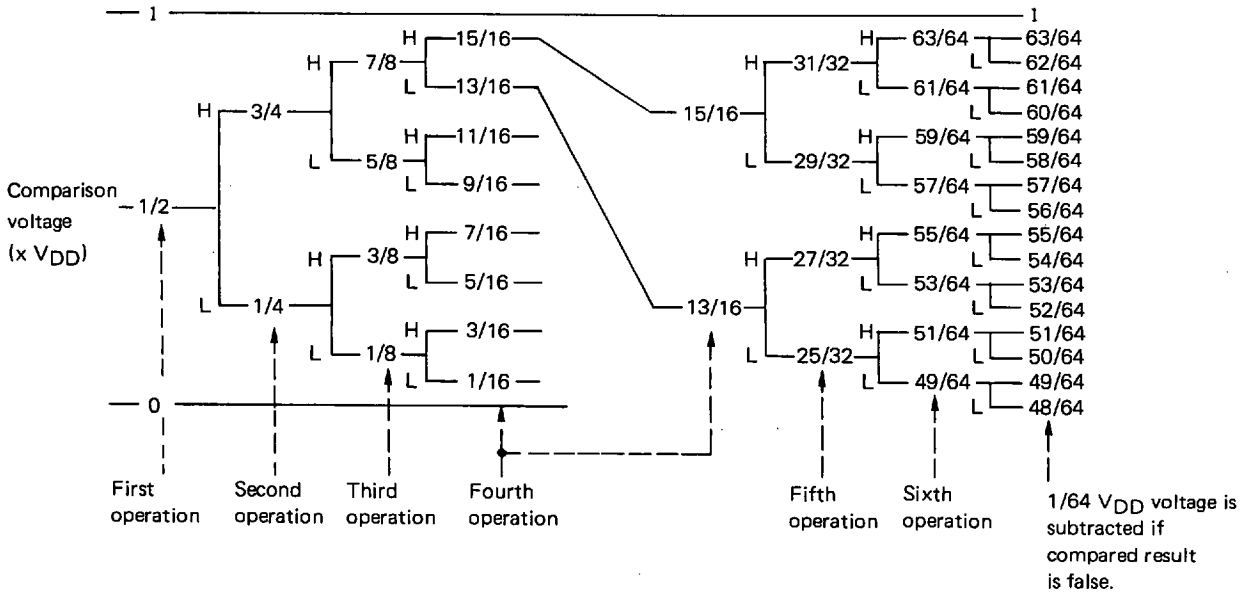
Therefore, a binary search method is convenient as described below.

(1) Binary search concept

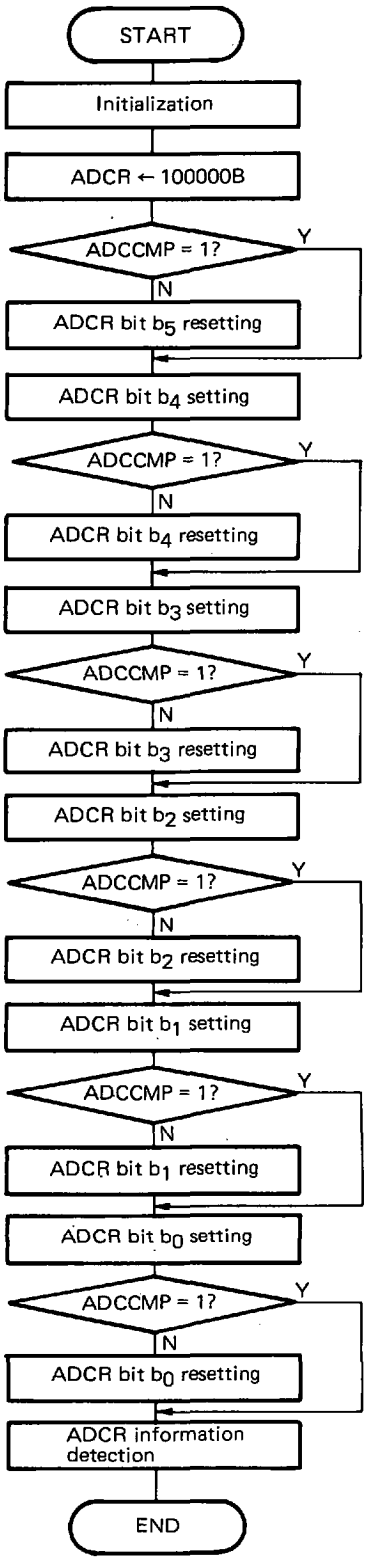
The binary search concept is described below.

A 1/2 V_{DD} voltage is first set to the comparison voltage. A 1/4 V_{DD} voltage is applied if the compared result is true (a high signal is input). A 1/4 V_{DD} voltage is subtracted and compared if it is false (a low signal is input).

1/8 V_{DD} and 1/16 V_{DD} through 1/64 V_{DD} voltages are compared sequentially in the same manner as the above. If the compared result is false when the sixth operation is completed, a 1/64 V_{DD} voltage is subtracted. The conversion is then completed.



(2) Binary search flowchart



: Sets the pin used.

: Sets a 1/2 V_{DD} voltage to the comparison voltage.

: Detects the compared result.

: Subtracts a 1/2 V_{DD} voltage if the compared result is "0".

: Applies a 1/4 V_{DD} voltage and sets it to the comparison voltage if the compared result is "0" or "1".

: Detects the compared result.

: Subtracts a 1/4 V_{DD} voltage if the compared result is "0".

: Applies a 1/8 V_{DD} voltage and sets it to the comparison voltage if the compared result is "0" or "1".

: Detects the compared result.

: Subtracts a 1/8 V_{DD} voltage if the compared result is "0".

: Applies a 1/16 V_{DD} voltage and sets it to the comparison voltage if the compared result is "0" or "1".

: Detects the compared result.

: Subtracts a 1/16 V_{DD} voltage if the compared result is "0".

: Applies a 1/32 V_{DD} voltage and sets it to the comparison voltage if the compared result is "0" or "1".

: Detects the compared result.

: Subtracts 1/32 V_{DD} voltage if the compared result is "0".

: Applies a 1/64 V_{DD} voltage and sets it to the comparison voltage if the compared result is "0" or "1".

: Detects the compared result.

: Subtracts a 1/64 V_{DD} voltage if the compared result is "0".

: Terminates the conversion if the compared result is "1".

(3) Binary search program example

(a) For short conversion time

```
INIT:
    ADCR7   FLG   0.0EH.3 ;  DUMMY
    ADCR6   FLG   0.0EH.2 ;  DUMMY
    ADCR5   FLG   0.0EH.1 ;  Each data buffer bit is defined as an ADCR data setting flag.
    ADCR4   FLG   0.0EH.0
    ADCR3   FLG   0.0FH.3
    ADCR2   FLG   0.0FH.2
    ADCR1   FLG   0.0FH.1
    ADCR0   FLG   0.0FH.0
    INITFLG NOT ADCCH3, NOT ADCCH2, NOT ADCCH1, NOT ADCCH0
                          ;  The P1D0/ADC0 pins are set in an A/D converter.

START:
    INITFLG NOT ADCR3, NOT ADCR2, NOT ADCR1, NOT ADCR0
    INITFLG NOT ADCR7, NOT ADCR6,      ADCR5, NOT ADCR4
    PUT     ADCR, DBF   ;  Sets a 31.5/64 VDD voltage to the comparison voltage.
    SKT1    ADCCMP      ;  Detects ADCCMP.
    CLR1    ADCR5       ;  Subtracts a 32/64 VDD voltage if the compared result is "0".
    SET1    ADCR4       ;  Applies a 16/64 VDD voltage.
    PUT     ADCR, DBF
    SKT1    ADCCMP      ;  Detects ADCCMP.
    CLR1    ADCR4       ;  Subtracts a 16/64 VDD voltage if the compared result is "0".
    SET1    ADCR3       ;  Applies a 8/64 VDD voltage.
    PUT     ADCR, DBF
    SKT1    ADCCMP      ;  Detects ADCCMP.
    CLR1    ADCR3       ;  Subtracts a 8/64 VDD voltage if the compared result is "0".
    SET1    ADCR2       ;  Applies a 4/64 VDD voltage.
    PUT     ADCR, DBF
    SKT1    ADCCMP      ;  Detects ADCCMP.
    CLR1    ADCR2       ;  Subtracts a 4/64 VDD voltage if the compared result is "0".
    SET1    ADCR1       ;  Applies a 2/64 VDD voltage.
    PUT     ADCR, DBF
    SKT1    ADCCMP      ;  Detects ADCCMP.
    CLR1    ADCR1       ;  Subtracts a 2/64 VDD voltage if the compared result is "0".
    SET1    ADCR0       ;  Applies a 1/64 VDD voltage.
    PUT     ADCR, DBF
    SKT1    ADCCMP      ;  Detect ADCCMP.
    CLR1    ADCR0       ;  Subtracts a 1/64 VDD voltage if the compared result is "0".
Program step            :  31 steps
Execution step          :  31 steps
A/D conversion time     :  137.8 μs
```

(b) For short program step count

```
INIT:
    WORKR1  MEM    0.01H  ;
    WORKR0  MEM    0.00H  ;
    INITFLG NOT ADCCH3 NOT ADCCH2, NOT ADCCH1, NOT ADCCH0
                          ; The P1D0/ADC0 pins are set to an A/D converter.
START:
    MOV     DBF1,    #0010B
    MOV     DBF0,    #0000B
    MOV     WORKR1,  #0110B
    MOV     WORKR0,  #0000B
    CLR1    CY
LOOP:
    RORC    WORKR1
    RORC    WORKR0
    SKF1    CY
    BR      END
    PUT     ADCR,    DBF   ; Sets a 31.5/64 VDD voltage to the comparison voltage.
    SKT1    ADCCMP         ; Detects ADCCMP.
    BR      BBB
AAA:                       ; Increase the comparison voltage.
                           ; If the compared result is "1".
    OR      DBF1,    WORKR1
    OR      DBF0,    WORKR0
    BR      LOOP
BBB:                       ; Decreases the comparison voltage if the compared result is "0".
    EOR     DBF1,    WORKR1
    EOR     DBF0,    WORKR0
    BR      LOOP
END:
```

Program step : 20 steps
Execution step : 65 steps
A/D conversion time : 288.9 μs

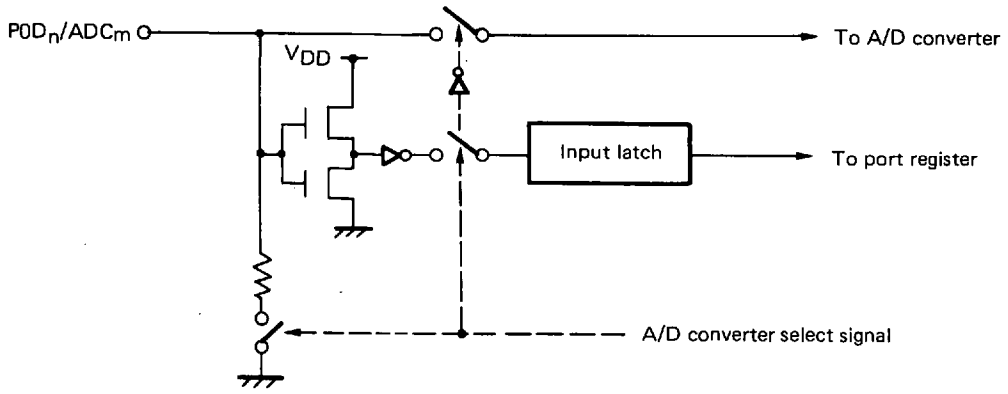
18.8 CAUTIONS WHEN USING A/D CONVERTER

The halt mode may not be entered when the $P0D_3/ADC_5$ through $P0D_2/ADC_2$ pins are used as an A/D converter and when the halt cancel is set by entering the key because the input port latch is isolated from the pins that is are set in the A/D converter as described in Section **14.4, "Halt Functions".**

Fig. 18-5 shows the relation between the $P0D_3/ADC_5$ through $P0D_2/ADC_2$ pins and the input latch.

As shown in Fig. 18-5, the input latch is held in high ("1") level if a high signal is input to the input pin when the A/D converter is set using an A/D converter select signal. Therefore, a high signal is determined to be input to this pin even if the halt cancel is set using a key when the input latch is held. The halt mode is then canceled immediately.

Fig. 18-5 Relation between $P0D_3/ADC_5$ through $P0D_2/ADC_2$ pins and input latch



18.9 STATE DURING RESET

18.9.1 State During Power on Reset

The $P0D_3/ADC_5$ through $P0D_0/ADC_2$, $A1D_1/ADC_1$, and $P1D_0/ADC_0$ pins are set in the general-purpose input port.

18.9.2 State During Clock Stop

The $P0D_3/ADC_5$ through $P0D_0/ADC_2$, $P1D_1/ADC_1$, and $P1D_0/ADC_0$ pins are set in the general-purpose input port.

18.9.3 State During CE Reset

The pin that is set in an A/D converter is held.

19. D/A CONVERTER (DAC)

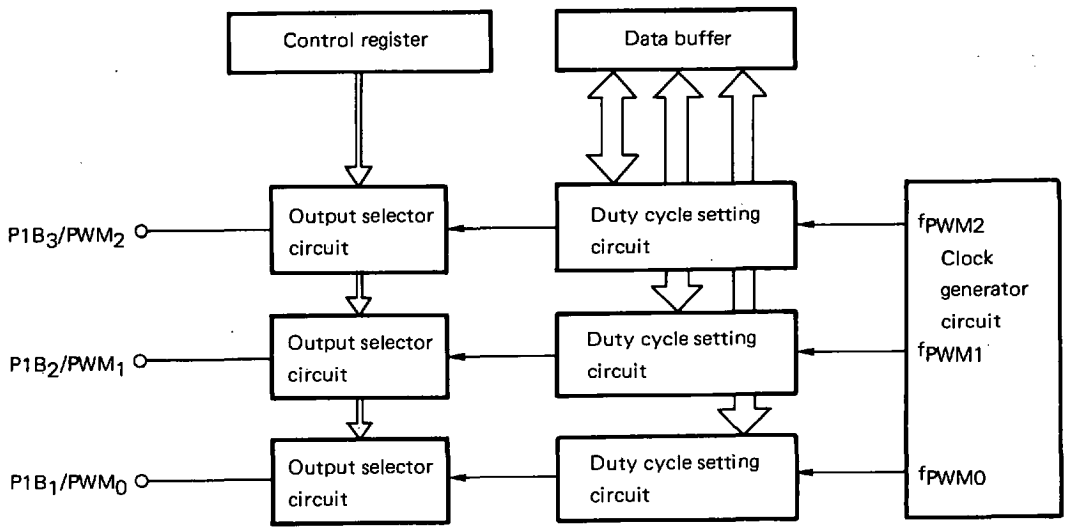
The D/A converter outputs a signal in accordance with a pulse width modulation (PWM) system in which the duty cycle can changed. The D/A converter can also convert the digital signal into an analog signal using an external low-pass filter.

19.1 D/A CONVERTER CONFIGURATION

Fig. 19-1 shows the D/A converter block diagram.

As shown in Fig. 19-1, the D/A converter consists of input selector circuits for each pin, duty cycle setting circuits, and a clock generator circuit.

Fig. 19-1 D/A converter block diagram



19.2 D/A CONVERTER FUNCTIONS

Each D/A converter pin outputs a duty variable signal independently. The output frequency is 878.9 Hz. The duty cycle can be changed in 256 steps.

Each block function is described below.

19.2.1 Input Selector Circuits

The input selector circuits set whether each pin should be used as a general-purpose output port or A/D converter. Each pin is selected using a PWM mode select register (RF address 13H). For more details, see Section **19.3.**

19.2.2 Duty Cycle Setting Circuits

The duty cycle setting circuits output a 256-step duty variable signal. The duty cycle at each pin is set independently via the data buffer using a PWM data register (peripheral addresses 05H, 06H, and 07H). For more details, see Section **19.4.**

19.2.3 Clock Generator Circuit

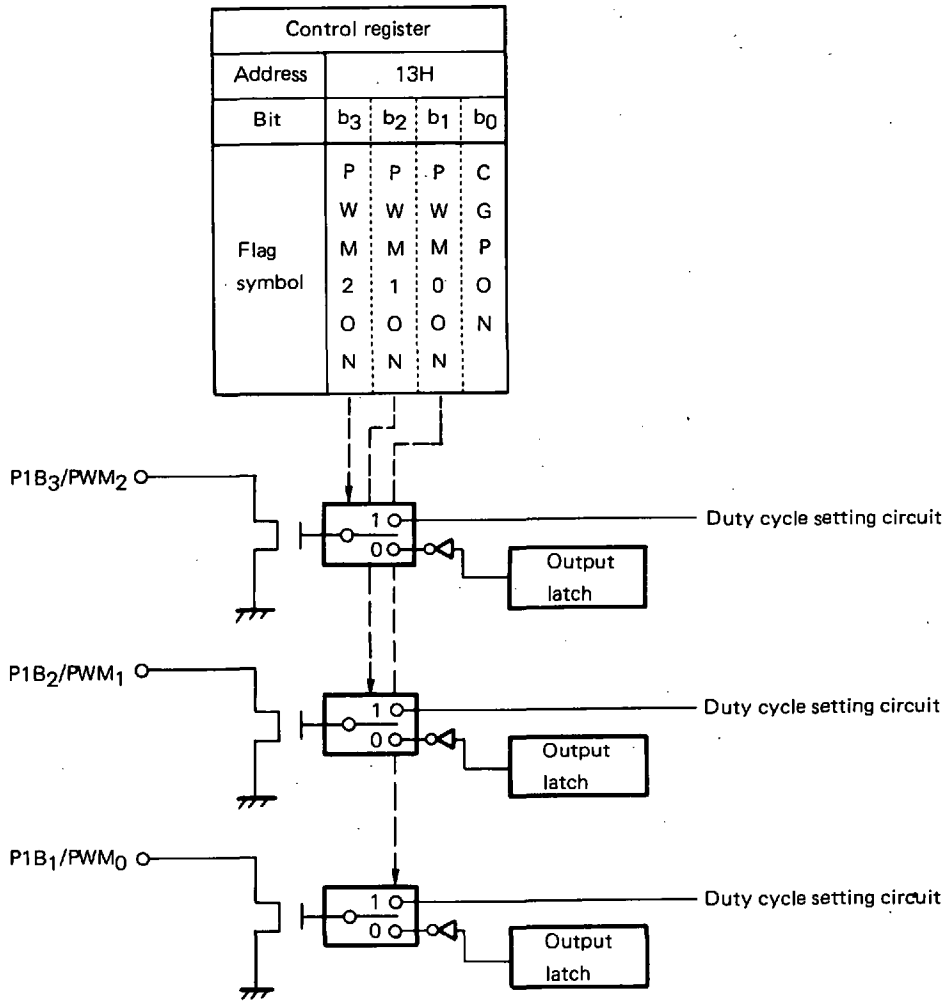
The clock generator circuit generates a reference clock pulse that sets the duty cycle. Generated clock frequency f_{PWM} is 255 kHz. For more details, see Section **19.5.**

19.3 OUTPUT SELECTOR CIRCUITS

19.3.1 Output Selector Circuit Configuration

Fig. 19-2 shows the output selector circuit configuration.

Fig. 19-2 Output selector circuit configuration



19.3.2 Output Selector Circuit Functions

The output selector circuits set whether a general-purpose output port or D/A converter should be used. The general-purpose output port and D/A converter at each pin can be set independently using PWM2ON, PWM1ON, and PWM0ON PWM mode select register flags.

The $P1B_3/PWM_2$ through $P1B_1/PWM_0$ pins require and external pull-up resistor because of the N-channel open drain output.

The PWM mode select register configuration and functions are described in Section **19.3.3.**

19.3.3 PWM Mode Select Register Configuration and Functions

The PWM mode select register sets the pin that is used as a D/A converter (PWM output) and clock generator port (CGP).

The PWM mode select register configuration and functions are shown below.

For more information on the clock generator port, see Section **20, "Clock Generator Port (CGP)"**.

Name	Flag symbol				Address	Read/Write
	b_3	b_2	b_1	b_0		
PWM mode select register (PWMMODE)	PWM2ON	PWM1ON	PWM0ON	CGPON	13H	R/W

CGPON	Sets the $P1B_0$/CGP pin as a CGP function.
0	General-purpose output port
1	CGP function

PWM2ON	PWM1ON	PWM0ON	Sets the pin that is used as a D/A converter (PWM output).
			$P1B_1/PWM_0$ pin
			$P1B_2/PWM_1$ pin
			$P1B_3/PWM_2$ pin
0			General-purpose output port
1			D/A converter

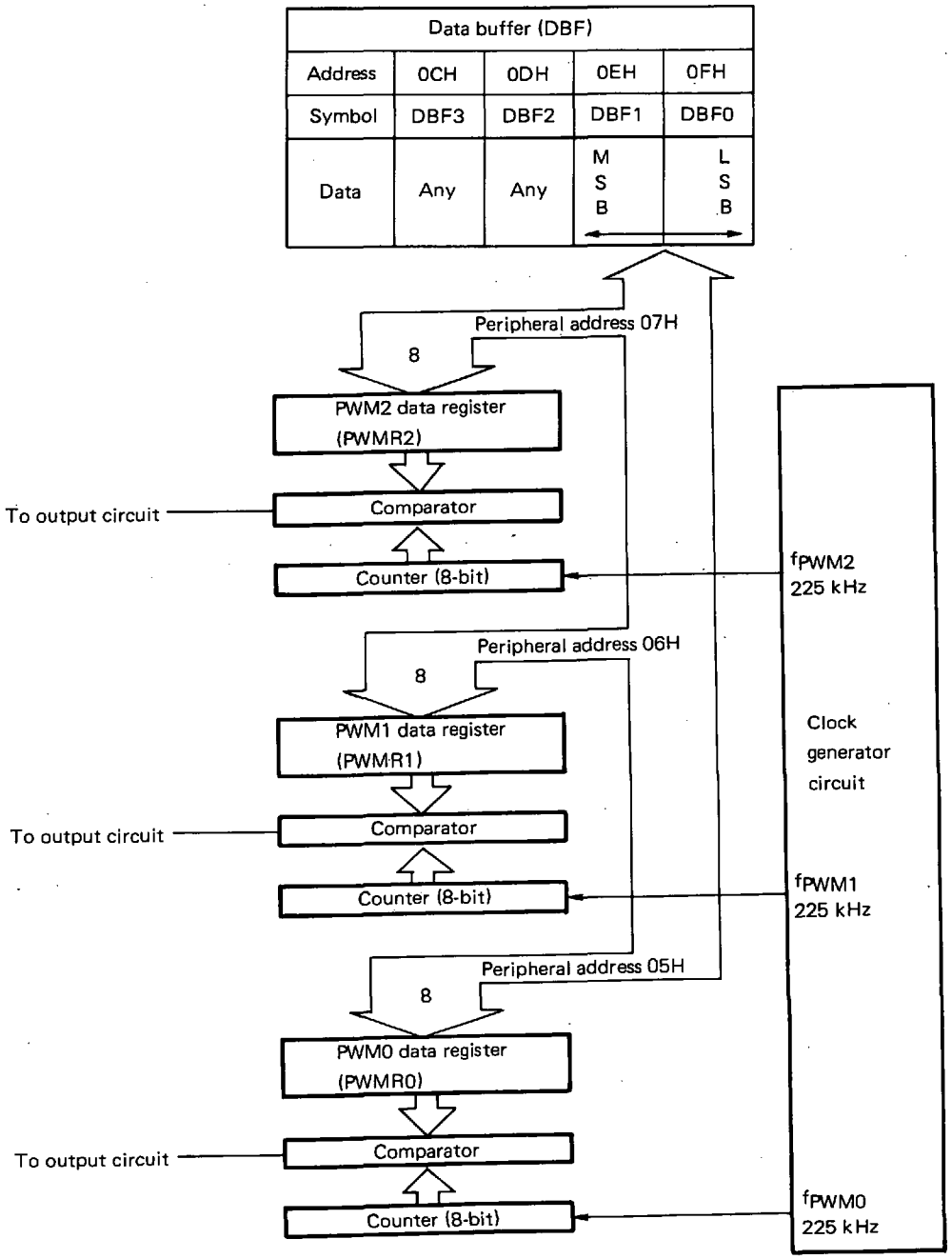
Reset					
	Power on	0	0	0	0
	Clock stop	0	0	0	0
	CE	Held			

19.4 DUTY CYCLE SETTING CIRCUITS AND CLOCK GENERATOR CIRCUIT

19.4.1 Duty Cycle Setting Circuit and Clock Generator Circuit Configuration

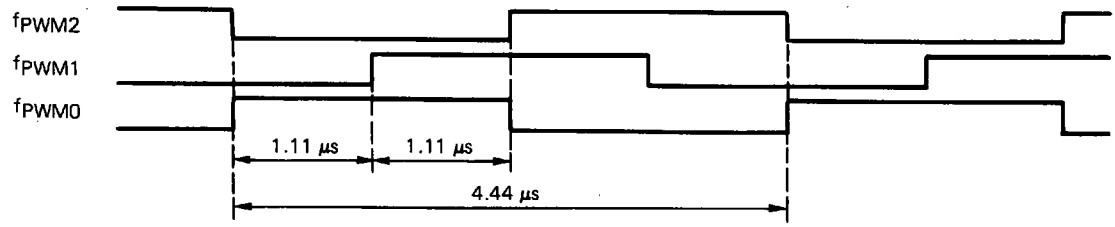
Fig. 19-3 shows the duty cycle setting circuit and clock generator circuit configuration.

Fig. 19-3 Duty cycle setting circuit and clock generator circuit configuration



19.4.2 Clock Generator Circuit Functions and Operation

The clock generator circuit outputs the reference clocks (f_{PWM2}, f_{PWM1}, and f_{PWM0}) that set the duty cycle of each output signal (PWM_2, PWM_1, and PWM_0 pins). The f_{PWM2}, f_{PWM1}, and f_{PWM0} reference clock output frequencies are 225 kHz (4.44 μs). The phase difference between the reference clocks is as shown below.



19.4.3 Duty Cycle Setting Circuit Functions and Operation

The duty cycle setting circuits compare the values that are set in PWM data registers PWM2, PWM1, and PWM0 and the f_{PWM2}, f_{PWM1}, and f_{PWM0} reference clock values that are counted using eight-bit counters. A high signal is output if the PWM data register values are high, and a low signal is output if they are low.

Assume that the values set in the PWM data registers are "x". The duty cycle is given by the expression below.

$$\text{Duty cycle: } D = \frac{x + 0.25}{256} \times 100\ \%$$

where 0.25 indicates an offset value. A high signal is output even if x is "0".

The reference clock output frequency is 225 kHz, so the output signal frequency and period are as shown below.

$$\text{Frequency: } f = \frac{225\ \text{kHz}}{256} = 878.9\ \text{Hz}$$

$$\text{Period} \quad : t = \frac{256}{225\ \text{kHz}} = 1137.8\ \mu\text{s}$$

Data can be set independently in the PWM data registers via the data buffer. Independent duty signals can be output from each pin.

The PWM data register configuration and functions, and the relation between the output waveforms at each pin and their duty cycle are described in Sections **19.4.4** and **19.4.5.**

19.4.4 PWM Data Register Configuration and Functions

The PWM data register functions are shown below. The PWM data registers set the duty cycle of a D/A converter (PWM output) output signal.

Name	Data buffer			
Symbol	DBF 3	DBF 2	DBF 1	DBF 0
Address	0 CH	0 DH	0 EH	0 FH
Bit	b_{15} b_{14} b_{13} b_{12}	b_{11} b_{10} b_9 b_8	b_7 b_6 b_5 b_4	b_3 b_2 b_1 b_0
Data	Any	Any	Transfer data	

8

GET and PUT commands can be entered.

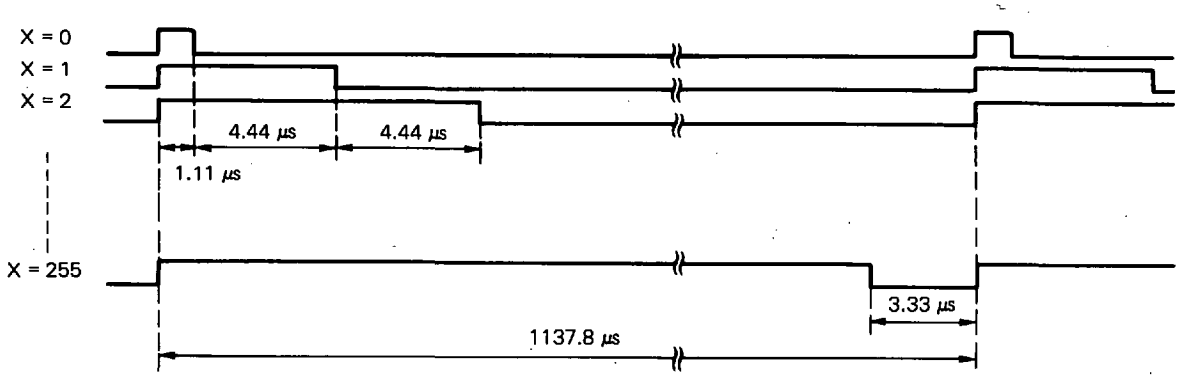
Peripheral register										
Name	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Symbol	Peripheral address	Peripheral hardware
PWM0 data register	Valid data								PWMR 0	05 H	PWM_0 pin
PWM1 data register									PWMR 1	06 H	PWM_1 pin
PWM2 data register									PWMR 2	07 H	PWM_2 pin

	Sets the duty cycle of a PWM output signal at each pin.
0	Duty cycle $D = \frac{x + 0.25}{256} \times 100\ \%$
x	Frequency: $f = \frac{225}{256}$ kHz
255	= 878.9 Hz

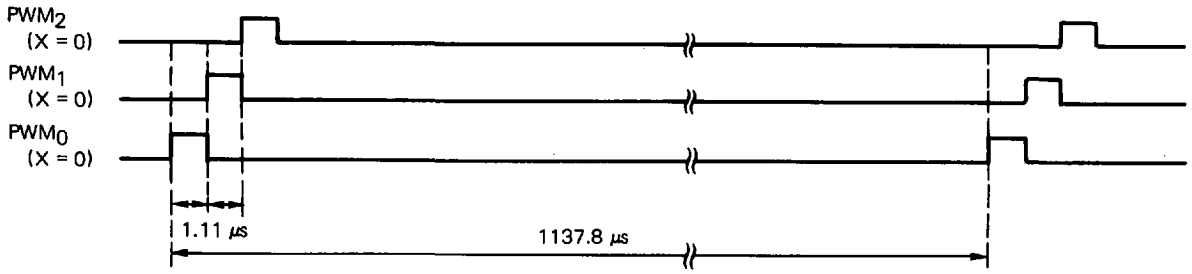
19.4.5 Relation between D/A Converter Output Waveforms and Pins

The relation between the duty cycle and output waveform is shown in step (1), and the output waveforms at each pin in step (2).

(1) Duty cycle and output waveform



(2) Output waveforms at pins



19.5 STATE DURING RESET

19.5.1 State During Power on Reset

The $P1B_3/PWM_2$ through $P1B_1/PWM_0$ pins are specified for a general-purpose output port. The output value becomes "undefined". The PWM data register values become "undefined".

19.5.2 State During Clock Stop

The $P1B_3/PWM_2$ through $P1B_1/PWM_0$ pins are specified for a general-purpose output port. The output value becomes "the former output latch information". The PWM data register values hold the former value.

19.5.3 State During CE Reset

The $P1B_3/PWM_2$ through $P1B_1/PWM_0$ pins hold the former output state. The pin that is used as a D/A converter holds the PWM output state.

19.5.4 State in Halt Mode

The $P1B_3/PWM_2$ through $P1B_1/PWM_0$ pins hold the former output state. The pin that is used as a D/A converter hold the PWM output state.

20. CLOCK GENERATOR PORT (CGP)

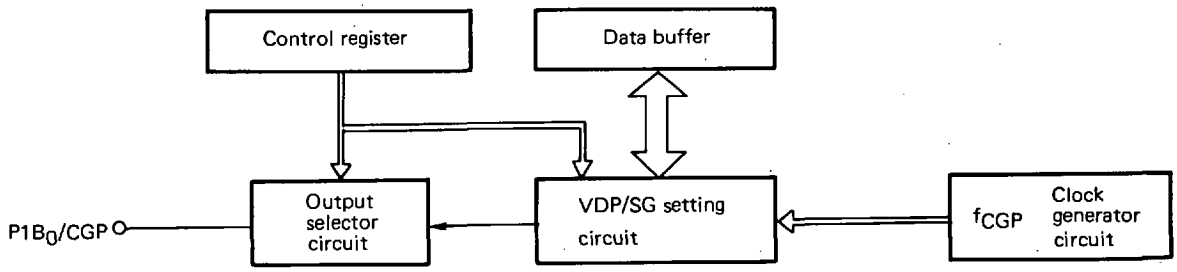
The clock generator port outputs a signal in accordance with a variable duty pulse (VDP) system that can change the duty cycle and a signal generator system that can change the frequency.

20.1 CLOCK GENERATOR PORT CONFIGURATION

Fig. 20-1 shows the clock generator port block diagram.

As shown in Fig. 20-1, the clock generator port consists of a pin input selector circuit, VDP/SG setting circuit, and clock generator circuit.

Fig. 20-1 Clock generator port block diagram



20.2 CLOCK GENERATOR PORT FUNCTIONS

The clock generator port outputs a duty variable signal (VDP function) or frequency variable signal (SG function) from the $P1B_0$/CGP pin. The VDP function can change the duty cycle in 64 steps. The SG function can change the frequency in 64 steps.

Each block function is described below.

The clock generator port is used in combination with a frequency counter described later and hardware. The clock generator port and frequency counter cannot be used at the same time. For more details, see Section **20.7.**

20.2.1 Input Selector Circuit

The input selector circuit sets whether the $P1B_0$/CGP pin should be used as a general-purpose output port or clock generator port. The $P1B_0$/CGP pin is selected using a PWM mode select register (RF address 13H). For more details, see Section **20.3.**

20.2.2 VDP/SG Setting Circuit

The VDP/SG setting circuit selects VDP and SG functions, outputs a duty variable signal when the VDP function is used, and outputs a frequency variable signal when the SG function is used. The duty and frequency variable signals when using the VDP and SG functions are set via the data buffer using a CGP data register (peripheral address 20H).

For more details, see Section **20.4.**

20.2.3 Clock Generator Circuit

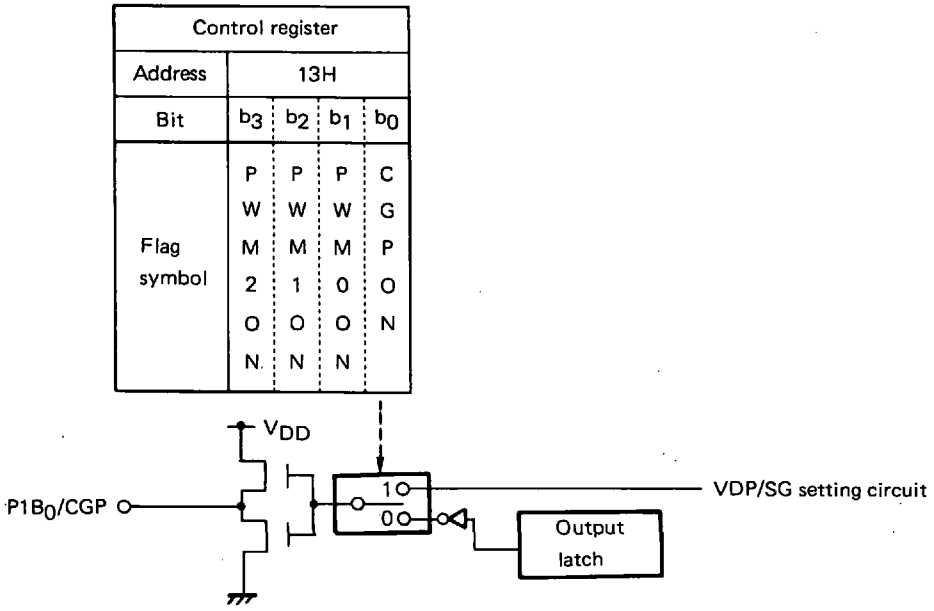
The clock generator circuit generates a reference clock pulse that sets the VDP duty variable and SG frequency variable signals. Generated clock frequency f_{CGP} is 18 kHz. For more details, See Section **20.4.**

20.3 OUTPUT SELECTOR CIRCUIT

20.3.1 Output Selector Circuit Configuration

Fig. 20-2 shows the output selector circuit configuration.

Fig. 20-2 Output selector circuit configuration



20.3.2 Output Selector Circuit Functions

The output selector circuit selects a general-purpose output port and clock generator port. The general-purpose output port and clock generator port are selected using the CGPON flag of a PWM mode select register. The PWM mode select register configuration and functions are described in Section **20.3.3.**

20.3.3 PWM Mode Select Register Configuration and Functions

The PWM mode select register sets the pin that is used as a D/A converter and clock generator port. The PWM mode select register configuration and functions are shown below. For more information on the D/A converter, see Section **19, "D/A Converter (DAC)".**

Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
PWM mode select register (PWMMODE)	PWM2ON	PWM1ON	PWM0ON	CGPON	13H	R/W

CGPON — Sets the $P1B_0$/CGP pin as a CGP function.

0	General-purpose output port
1	CGP function

PWM2ON, PWM1ON, PWM0ON — Sets the pin that is used as a D/A converter (PWM output).

	PWM2ON	PWM1ON	PWM0ON
	$P1B_3/PWM_2$ pin	$P1B_2/PWM_1$ pin	$P1B_1/PWM_0$ pin
0	General-purpose output port		
1	D/A converter		

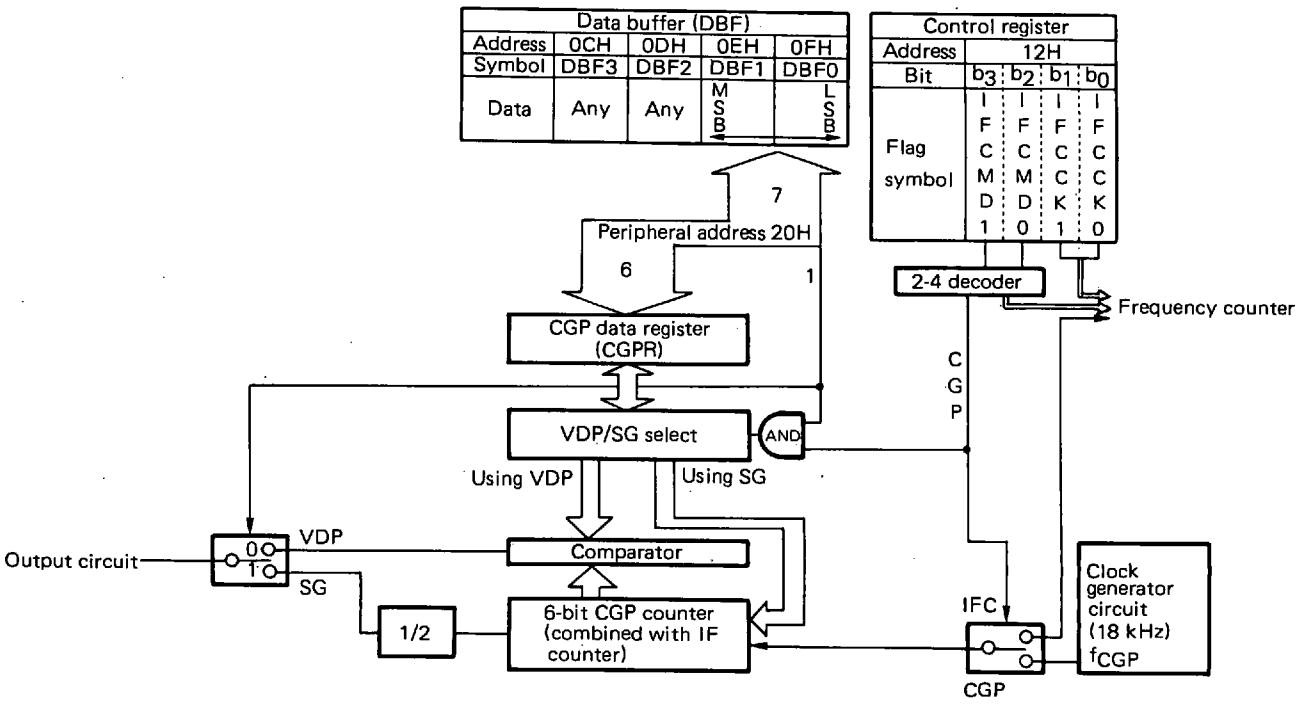
		b_3	b_2	b_1	b_0
Reset	Power on	0	0	0	0
	Clock stop	0	0	0	0
	CE	Held			

20.4 VDP/SG SETTING CIRCUIT AND CLOCK GENERATOR CIRCUIT

20.4.1 VDP/SG Setting Circuit and Clock Generator Circuit Configuration

Fig. 20-3 shows the VDP/SG setting circuit and clock generator circuit configuration.

Fig. 20-3 VDP/SG setting circuit and clock generator circuit configuration



20.4.2 Clock Generator Circuit Functions and Operation

The clock generator circuit outputs a reference clock pulse (f_{CGP}) that sets the VDP duty variable and SG frequency variable signals. The output frequency is 18 kHz.

20.4.3 VDP/SG Setting Circuit Functions and Operation

The VDP/SG setting circuit selects the VDP and SG functions, and sets the VDP duty variable and SG frequency variable signals.

The 6-bit counter (CGP counter) of the VDP/SG setting circuit is used in combination with an IF counter described later. The clock generator port or frequency counter function is selected using an IF counter mode select register (RF address 12H).

The operation when using the VDP and SG functions is described below. Data is set via the data buffer in the CGP data register. The IF counter mode select register configuration and functions are described in Section **20.4.4**, the CGP data register configuration and functions in Section **20.4.5**, and the output waveforms when using the VDP and SG functions in Section **20.4.6**. The CGP data register setting value, and VDP duty cycle and SG frequency list are described in Section **20.4.7**.

(1) VDP function

The value that is set in a CGP data register is compared with the reference clock (f_{CGP}) value that is counted using a 6-bit counter when the VDP function is used. A high signal is output if the CGP data register value is high, and a low signal is output if it is low.

Assume that the value set in the CGP data register is "x". The duty cycle is given by the expression below.

$$\text{Duty: } D_{VDP} = \frac{x+2}{67} \times 100\ \%$$

where 2 indicates the offset value. A high signal is output even if x is "0".

The reference clock output frequency is 18 kHz, so output signal frequency f_{VDP} and period t_{VDP} are as shown below.

$$\text{Frequency: } f_{VDP} = \frac{18\ \text{kHz}}{67} = 268.7\ \text{Hz}$$

$$\text{Period: } t_{VDP} = \frac{67}{18\ \text{kHz}} = 3722.2\ \mu\text{s}$$

(2) SG function

A signal is output if the value set in a CGP data register becomes "0" by counting the reference clock (f_{CGP}) using a 6-bit counter (CGP counter) when the SG function is used.

Assume that the value set in the CGP data register is "x". The output frequency is given by the expression below.

$$\text{Frequency: } f_{SG} = \frac{18}{2\,(x+2)}\ \text{kHz}$$

where 2 indicates the offset value. The frequency is divided even if x is "0".

A 1/2 frequency divider is used at that time, so the duty cycle becomes 50 % as shown below.

$$\text{Duty cycle: } D_{SG} = 50\ \%$$

20.4.4 IF Counter Mode Select Register (IFCMODE) Configuration and Functions

The IF counter mode select register sets frequency counter (IF counter and external gate counter) and clock generator port functions.

The IF counter mode select register configuration and functions are shown below.

Name	Flag symbol b_3	b_2	b_1	b_0	Address	Read/Write
IF counter mode select register (IFCMODE)	IFCMD1	IFCMD0	IFCCK1	IFCCK0	12H	R/W

Sets the gate time of an IF counter and the reference frequency of an external gate counter.

IFCCK1	IFCCK0	Gate time of IF counter	Reference frequency of external gate counter
0	0	1 ms	1 kHz
0	1	4 ms	100 kHz
1	0	8 ms	900 kHz
1	1	Open	0 kHz

Sets the IF counter, external gate counter (FCG), and clock generator port functions.

IFCMD1	IFCMD0	
0	0	Clock generator port (CGP)
0	1	IF counter (FMIFC)
1	0	IF counter (AMIFC)
1	1	External gate counter (FCG)

Reset		IFCMD1	IFCMD0	IFCCK1	IFCCK0
	Power on	0	0	0	0
	Clock stop	0	0	0	0
	CE	Held			

The frequency counter and clock generator port functions cannot be used at the same time.

IFCMD1 and IFCMD0 flags are specified as "0" when the clock generator port is used. The CGPON flag of an output selector circuit must also be set after the IFCMD1 and IFCMD0 flags are specified as "0".

20.4.5 CGP Data Register Configuration and Functions

The CGP data register configuration and functions are shown below.

The CGP data register selects the VDP and SG functions and sets the VDP duty variable and SG frequency variable signals.

Name	Data buffer			
Symbol	DBF 3	DBF 2	DBF 1	DBF 0
Address	0 CH	0 DH	0 EH	0 FH
Bit	b_{15} b_{14} b_{13} b_{12}	b_{11} b_{10} b_9 b_8	b_7 b_6 b_5 b_4	b_3 b_2 b_1 b_0
Data	Any	Any	Transfer data	

8

GET and PUT commands can be entered.

Peripheral register

Name	b_7 b_6 b_5 b_4 b_3 b_2 b_1	b_0	Symbol	Peripheral address	Peripheral hardware
CGP data register	Valid data	0	CGPR	20 H	Clock generator port (CGP)

Fixed at "0".

Selects the SG and VDP functions.

0	VDP function
1	SG function

Sets the VDP duty variable and SG frequency variable signals.

0	VDP function Duty cycle: $D = \frac{x+2}{67} \times 100\ \%$ Frequency: $f = 269$ Hz
x	SG function Frequency: $f = \frac{18}{2(x+2)}$ kHz Duty cycle: $D = 50\ \%$
63	

Used in combination.

Peripheral register

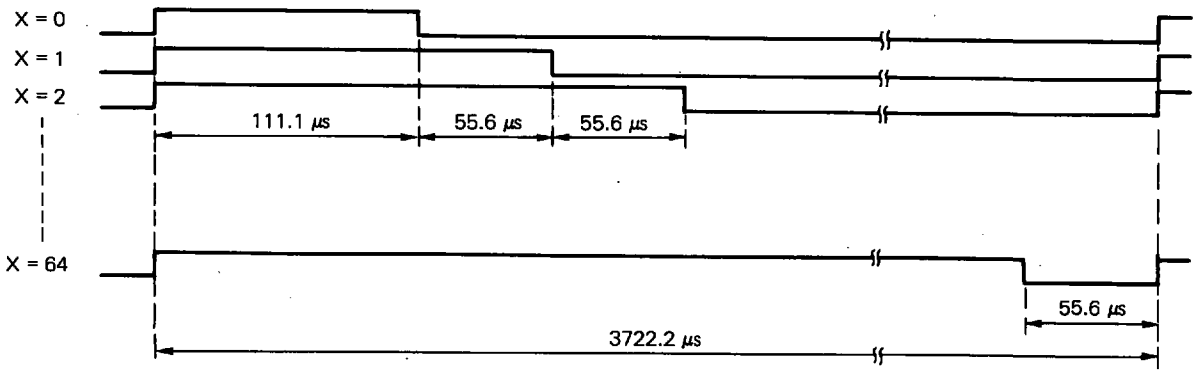
Name	b_{15} b_{14} b_{13} b_{12} b_{11} b_{10} b_9 b_8 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0
IF counter data register	Valid data

The CGP data register is used in combination with the high-order 6 bits of the IF counter data register in a frequency counter described later. Consequently, the frequency counter does not operate normally if data is set in the CGP data register (using a PUT command) when it is used. For more details, see Section **20.7.**

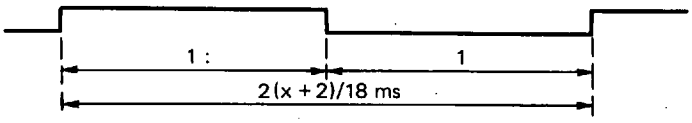
20.4.6 Output Waveform when Using VDP and SG Functions

The relation between the duty cycle and output waveform when using the VDP function is shown in step (1), and the output waveform when using the SG function in step (2).

(1) Duty cycle and output waveform when using VDP function

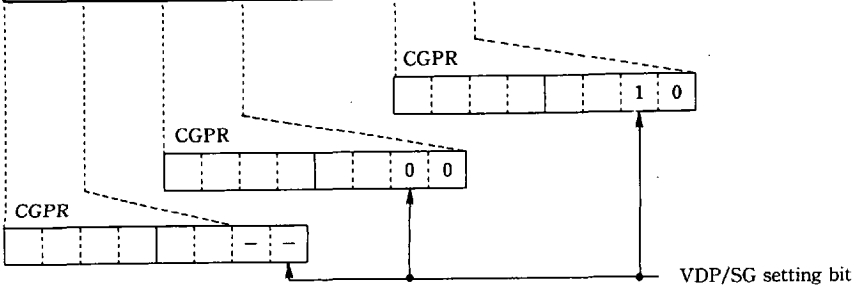


(2) Output waveform when using SG function



20.4.7 CGP Data Register (CGPR) Setting Value, and VDP Duty Cycle and SG Frequency List

CGPR setting data (high-order six bits)		VDP duty cycle		SG frequency		CGPR setting data (high-order six bits)		VDP duty cycle		SG frequency	
Decimal (DEC)	Hexa-decimal (HEX)	Hexa-decimal		Hexa-decimal	(Hz)	Decimal (DEC)	Hexa-decimal (HEX)	Hexa-decimal		Hexa-decimal	(Hz)
0	00H	00H	2/67	02H	4500.000	32	20H	80H	34/67	82H	264.706
1	01H	04H	3/67	06H	3000.000	33	21H	84H	35/67	86H	257.143
2	02H	08H	4/67	0AH	2250.000	34	22H	88H	36/67	8AH	250.000
3	03H	0CH	5/67	0EH	1800.000	35	23H	8CH	37/67	8EH	243.243
4	04H	10H	6/67	12H	1500.000	36	24H	90H	38/67	92H	236.842
5	05H	14H	7/67	16H	1285.714	37	25H	94H	39/67	96H	230.769
6	06H	18H	8/67	1AH	1125.000	38	26H	98H	40/67	9AH	225.000
7	07H	1CH	9/67	1EH	1000.000	39	27H	9CH	41/67	9EH	219.512
8	08H	20H	10/67	22H	900.000	40	28H	A0H	42/67	A2H	214.286
9	09H	24H	11/67	26H	818.182	41	29H	A4H	43/67	A6H	209.302
10	0AH	28H	12/67	2AH	750.000	42	2AH	A8H	44/67	AAH	204.545
11	0BH	2CH	13/67	2EH	692.308	43	2BH	ACH	45/67	AEH	200.000
12	0CH	30H	14/67	32H	642.857	44	2CH	B0H	46/67	B2H	195.652
13	0DH	34H	15/67	36H	600.000	45	2DH	B4H	47/67	B6H	191.489
14	0EH	38H	16/67	3AH	562.500	46	2EH	B8H	48/67	BAH	187.500
15	0FH	3CH	17/67	3EH	529.412	47	2FH	BCH	49/67	BEH	183.673
16	10H	40H	18/67	42H	500.000	48	30H	C0H	50/67	C2H	180.000
17	11H	44H	19/67	46H	473.684	49	31H	C4H	51/67	C6H	176.471
18	12H	48H	20/67	4AH	450.000	50	32H	C8H	52/67	CAH	173.077
19	13H	4CH	21/67	4EH	428.571	51	33H	CCH	53/67	CEH	169.811
20	14H	50H	22/67	52H	409.091	52	34H	D0H	54/67	D2H	166.667
21	15H	54H	23/67	56H	391.304	53	35H	D4H	55/67	D6H	163.636
22	16H	58H	24/67	5AH	375.000	54	36H	D8H	56/67	DAH	160.714
23	17H	5CH	25/67	5EH	360.000	55	37H	DCH	57/67	DEH	157.895
24	18H	60H	26/67	62H	346.154	56	38H	E0H	58/67	E2H	155.172
25	19H	64H	27/67	66H	333.333	57	39H	E4H	59/67	E6H	152.542
26	1AH	68H	28/67	6AH	321.429	58	3AH	E8H	60/67	EAH	150.000
27	1BH	6CH	29/67	6EH	310.345	59	3BH	ECH	61/67	EEH	147.541
28	1CH	70H	30/67	72H	300.000	60	3CH	F0H	62/67	F2H	145.161
29	1DH	74H	31/67	76H	290.323	61	3DH	F4H	63/67	F6H	142.857
30	1EH	78H	32/67	7AH	281.250	62	3EH	F8H	64/67	FAH	140.625
31	1FH	7CH	33/67	7EH	272.727	63	3FH	FCH	65/67	FEH	138.462



20.5 USE OF CLOCK GENERATOR PORT

How to use the VDP and SG functions are described below.

20.5.1 VDP Function

A program example of the VDP function is shown below.

As shown in the example, a $P1B_0$/CGP pin must be set for the CGP output using a "SET1 CGPON" command after data is set in the CGP data register. An undefined signal is output if the CGP data register information is undefined (during the power on reset) when the "SET1 CGPON" command is executed before data is set in the CGP data register.

Example:

A signal with a duty cycle of 10/67 is output.

```
VDPDUTY DAT 20H          ; Defines data with a duty cycle of 10/67 using a VDP function.
INITFLG NOT IFCMD1, NOT IFCMD0, NOT IFCCK1, NOT IFCCK0
                         ; Sets a six-bit counter in CGP.
MOV DBF1, (VDPDUTY SFR 4) AND 000FH
MOV DBF0, VDPDUTY AND 000FH
PUT CGPR, DBF            ; Sets a VDP function and duty cycle in the CGP data register.
SET1 CGPON               ; Sets a P1B0/CGP pin for the CGP output.
                         ; Execute this command after data is set in the CGP data register.
```

20.5.2 SG Function

A program example of the SG function is shown below.

As in the VDP function above, a $P1B_0$/CGP pin must be set for the CGP output using a "SET1 CGPON" command after data is set in the CGP data register. An undefined signal is output if the CGP data register information is undefined (during the power on reset) when the "SET1 CGPON" command is executed before data is set in the CGP data register.

Example

A signal with a frequency of 900 Hz is output.

```
SGFRQ  DATA 22H          ; Sets data with a frequency of 900 Hz using an SG function.
INITFLG NOT IFCMD1, NOT IFCMD0, NOT IFCCK1, NOT IFCCK0
                         ; Sets a six-bit counter in CGP.
MOV DBF1, (SGFRQ SFR 4) AND 000FH
MOV DBF0, SGFRQ AND 000FH
PUT CGPR, DBF            ; Sets an SG function and frequency in the CGP data register.
SET1 CGPON               ; Sets a P1B0/CGP pin for the CGP output.
                         ; Execute this command after data is set in the CGP data register.
```

20.6 STATE DURING RESET

20.6.1 State During Power on Reset

The $P1B_0$/CGP pin is specified for a general-purpose output port because a CGPON flag is reset.

The latch value at an output port is "undefined", so undefined data is output. The CGP data register value becomes "undefined".

20.6.2 State During Clock Stop

The $P1B_0$/CGP pin is specified for a general-purpose output port because a CGPON flag is reset.

The latch value at an output port becomes "the former output latch information". The latch value is then output. The CGP data register value holds the former value.

20.6.3 State During CE Reset

The $P1B_0$/CGP pin holds the former output state.

20.6.4 State in Halt Mode

The $P1B_0$/CGP pin holds the former output state.

20.7 CAUTIONS WHEN USING CLOCK GENERATOR PORT

The 6-bit counter that sets the duty cycle (VDP function) and frequency (SG function) of the clock generator port is used in combination with a frequency counter described later. The clock generator port and frequency counter cannot thus be used at the same time.

The operation described in Section **20.7.1** is performed when data is set in an IF counter mode select register and IF counter data register (peripheral address 43H) during clock generator port operation.

The operation described in Section **20.7.2** is performed when data is set in an IF counter mode select register and CGP data register during frequency counter operation.

20.7.1 When Used as Clock Generator Port

(1) IFCMD1 and IFCMD0 flags of IF counter mode select register are set

When logical values other than "0" are written in the IFCMD1 and IFCMD0 flags, the $P1B_0$/CGP pin holds the output level when data is set. The CGP is then stopped.

The CGP operation is started when the IFCMD1 and IFCMD0 flags are reset to "0".

(2) IF counter data register is used

The CGP operation is not influenced during read (GET) and write (PUT).

"Undefined" data is read during read, and nothing is changed during write.

The IF counter data register is a read only peripheral register. Write no data in the IF counter data register.

20.7.2 When Used as Frequency Counter

(1) IFCMD1 and IFCMD0 flags of IF counter mode select register are set

When logical "0" is written in the IFCMD1 and IFCMD0 flags, the $P1B_0$/CGP pin operates as a CGP data register when data is set.

The CGPON flag of a PWM mode select register must also be set to operate as the CGP data register.

The frequency counter operation is continued when the former value is set again to the IFCMD1 and IFCMD0 flags. However, no correct input data is obtained at that time. No frequency is counted while the CGP operation is selected.

(2) CGP data register is used

The frequency counter operation is not influenced during read (GET).

The frequency counter does not operate normally during write (PUT).

"Undefined" data is read during read.

Data is written in the high-order 6 bits of the frequency counter during write.

21. SERIAL INTERFACE

The serial interface is used for serial data transfer in 8 bits with an external device.

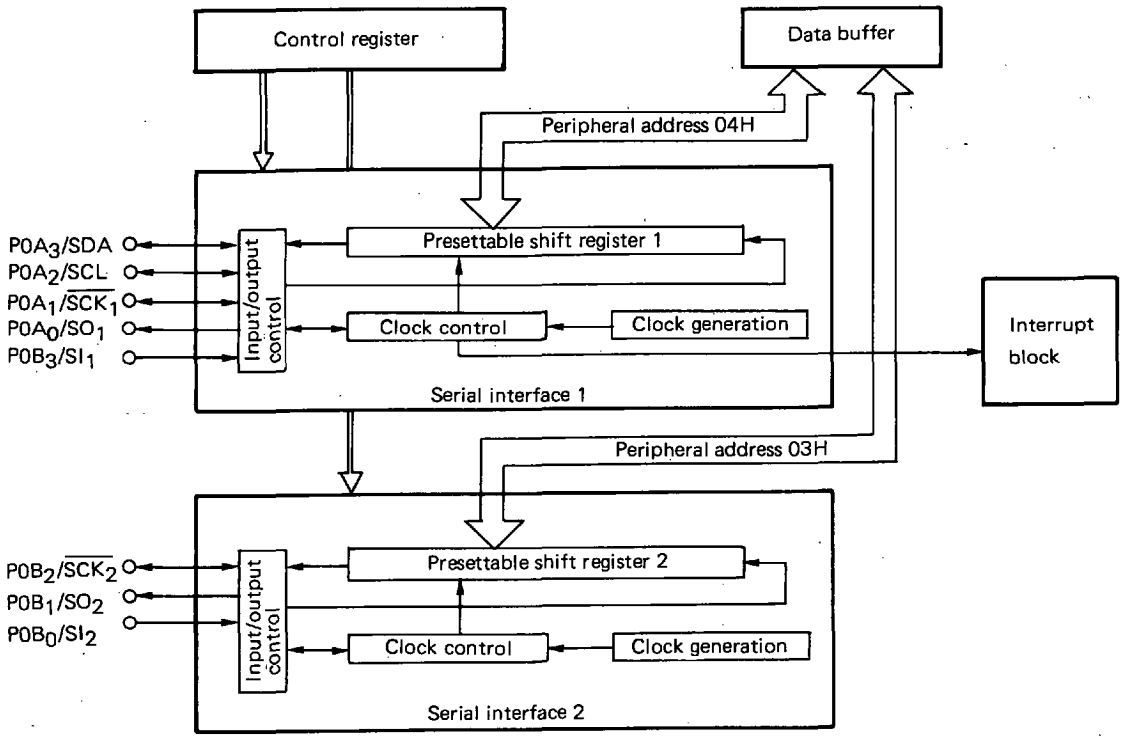
21.1 SERIAL INTERFACE CONFIGURATION

Fig. 21-1 shows the block diagram of this serial interface.

As shown, the serial interface is composed of two systems: serial interface 1 (SIO1) and serial interface 2 (SIO2).

The serial interface 1 and serial interface 2 are composed respectively of an input/output control block, presettable shift register, clock control and clock generation blocks.

Fig. 21-1 Serial interface block diagram



21.2 OUTLINE OF FUNCTIONS OF SERIAL INTERFACE

Table 21-1 shows the serial interface classification and communication systems.

As shown in Table 21-1, the serial interface is classified into the serial interface 1 (SIO1) and serial interface 2 (SIO2).

The serial interface 1 and serial interface 2 can be used at the same time. The serial interface 1 permits use of 2-wire system and 3-wire system. The 2-wire system uses $P0A_3$/SDA pin and $P0A_2$/SCL pin, whereas the 3-wire system uses $P0A_1/\overline{SCK_1}$ pin, $P0A_0/SO_1$ pin and $P0B_3/SI_1$ pin.

The communication system for the 2-wire system is selectable between the serial pulse system and serial I/O system.

The serial interface 2 permits use of 3-wire system only, and the pins used are $P0B_2/\overline{SCK_2}$, $P0B_1/SO_2$, and $P0B_0/SI_2$. The serial I/O communication system is used.

The serial interface 1 performs control by the control register's serial I/O1 mode select register (RF address 08H), serial I/O1 wait control register (RF address 18H), serial I/O1 status judge register (RF address 28H), serial I/O1 interrupt mode register (RF address 38H) and serial I/O1 clock select register (RF address 39H).

The serial interface 2 performs control by the control register's serial I/O2 mode select register (RF address 02H).

Setting of serial data and reading of serial in data of the serial interface 1 and serial interface 2 are performed respectively by the presettable shift register 1 (peripheral address 04H) and presettable shift register 2 (peripheral address 03H) via data buffer.

Sections **21.3** thru **21.12** explain the serial interface 1, and Sections **21.13** thru **21.21** explains the serial interface 2.

Table 21-1 Serial interface classification and communication system

	Classification by hardware	No. of communication wires	Communication system	Pin used
Serial interface	Serial interface 1 (SIO1)	2-wire system	Serial bus system	$P0A_3$/SDA $P0A_2$/SCL
			Serial I/O system	
		3-wire system	Serial I/O system	$P0A_1/\overline{SCK_1}$ $P0A_0/SO_1$ $P0B_3/SI_1$
	Serial interface 2 (SIO2)	3-wire system	Serial I/O system	$P0B_2/\overline{SCK_2}$ $P0B_1/SO_2$ $P0B_0/SI_2$

21.3 CONFIGURATION OF SERIAL INTERFACE 1 (SIO1)

Fig. 21-2 shows the block diagram of the serial interface 1.

As shown in Fig. 21-2, the shift clock control section of the serial interface 1 is composed of a clock input/output pin block, clock generation block, wait control block, clock count block, start/stop detection block and interrupt control block.

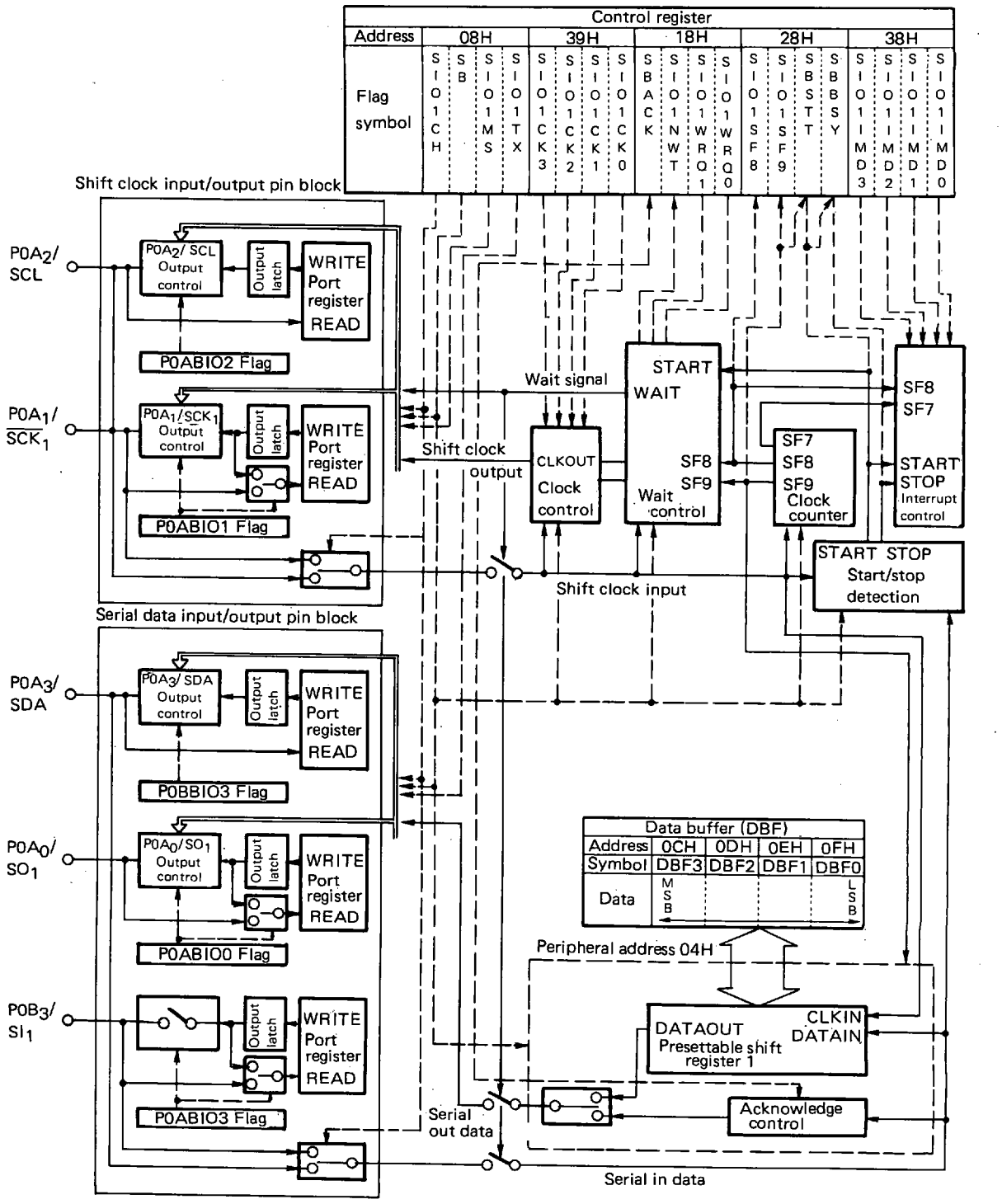
The serial data control section is composed of a serial data input/output pin block, presettable shift register and acknowledge block.

These blocks are controlled by the flags of the control register.

Writing of data into and reading of data from the presettable shift register are performed via the data buffer.

The functions of each block are outlined in Section **21.4.**

Fig. 21-2 Serial interface 1 block diagram



21.4 OUTLINE OF FUNCTIONS OF SERIAL INTERFACE 1

The serial interface 1 permits selection of 2-wire system and 3-wire system channels as shown in Table 21-1.

The 2-wire system uses two pins of $P0A_3$/SDA and $P0A_2$/SCL, and the 3-wire system uses three pins of $P0A_1$/SCK_1, $P0A_0$/SO_1 and $P0B_3$/SI_1.

The 2-wire system permits selection of communication system between serial bus system and serial I/O system, whereas the 3-wire system permits use of the serial I/O system only.

The serial bus system and serial I/O system permits selection of internal clock (master) operation and external clock (slave) operation, and also the reception (RX) operation and transmission (TX) operation are selectable.

The serial bus system permits serial communication between multiple devices to be performed using two wires.

Sections **21.4.1** thru **21.4.9** outlines the functions of each block indicated in Fig. 21-2.

For the details of each block, see Sections **21.5** thru **21.10.**

21.4.1 Shift Clock Input/Output Pin Block

This block is used for selecting the shift clock input/output pin. This selection of the shift clock input/output pin is performed by the serial I/O1 mode select register.

See Section **21.5.**

21.4.2 Serial Data Input/Output Pin Block

This block is used to select the shift clock input/output pin. This selection of the shift clock input/output pin is performed by the serial I/O1 mode select register.

See Section **21.5.**

21.4.3 Clock Generation Block

This block selects the clock frequency of the shift clock, and also controls the shift clock output timing. Selection of the clock frequency is performed by the serial I/O1 clock select register.

See Section **21.6.**

21.4.4 Clock Counter

The clock counter counts the number of the rising edges of the clocks output from the shift clock output pin, and issues signal at 7th clock (SF7 signal), 8th clock (SF8 signal) and 9th clock (SF9 signal).

These signals are used to control the wait (pause) and interruption of the serial communication.

The SF8 and SF9 signals can be detected by the serial I/O1 status judge register.

See Section **21.7.**

21.4.5 Start/Stop Detection Block

This block detects the start and stop conditions for the serial bus system.

This block does not operate when the serial I/O system is used.

The start and stop conditions can be detected using SBSTT and SBBSY flags of the serial I/O1 status judge register.

See Section **21.7.**

21.4.6 Presettable Shift Register (PSR1)

This is a shift register which sets the serial out data and stores the serial in data.

This register performs shift operation to input or output data by the clock input of the shift clock input pin.

Setting of the output data and reading of input data are performed via the data buffer.

See Section **21.8.**

21.4.7 Wait Control Block

This block controls the wait (pause) and wait cancel (communication operation) of serial communication. Setting of the wait condition is performed by SIO1WRQ1 and SIO1WRQ0 flags of the serial I/O1 wait control register, and setting of wait cancel is performed by SIO1NWT flag.

See Section **21.9.**

21.4.8 Acknowledge Block

This block controls the response signal (acknowledge) when the serial bus system is used. This block does not operate when the serial I/O system is used.

Setting and reading of the acknowledge are performed by SBACK flag of the serial I/O1 wait control register.

See Section **21.9.**

21.4.9 Interrupt Control Block

This block issues an interrupt request corresponding to the signal from the clock counter and start/stop detection block.

The interrupt request issuing condition is controlled by SIO1IMD3 thru SIO1IMD0 flags of the serial I/O1 interrupt mode register.

See Section **21.10.**

21.5 SHIFT CLOCK AND SERIAL DATA INPUT/OUTPUT PIN CONTROL BLOCK

The clock and data input/output control block controls the communication method of the serial interface 1 (serial bus system, serial I/O system), pins used (2-wire system, 3-wire system) and sending and receiving operations.

These are controlled by the serial I/O1 mode select register.

The configuration and function of the serial I/O1 mode select register are explained in Section **21.5.1.**

The setting status of each pin by the serial I/O1 mode select register is explained in Section **21.5.2.**

21.5.1 Configuration and Function of Serial I/O1 Mode Select Register

The configuration and function of the serial I/O1 mode select register are shown below.

Name	Flag b_3	b_2	b_1	b_0	Address	Read/Write
Serial I/O1 mode select register (SIO1MODE)	SIO1CH	SB	SIO1MS	SIO1TX	08H	R/W

SIO1TX — Setting of the serial input/output of $P0A_3$/SDA pin of 2-wire system and $P0A_0/SO_1$ pin of 3-wire system. (Selection of reception "RX" and transmission "TX" operations)

	2-wire system ($P0A_3$/SDA pin)	3-wire system ($P0A_0/SO_1$ pin)
0	Serial input (Hi-Z): RX operation	General purpose port
1	Serial output: TX operation	Serial output: TX operation

SIO1MS — Setting of the direction of shift clock

	Serial bus system	Serial I/O system
0	Slave operation (External clock input)	External clock input
1	Master operation (Internal clock output)	Internal clock output

SB — Selection of serial bus system or serial I/O system

0	Serial I/O system
1	Serial bus system (Do not set SIO1CH flag to 1.)

SIO1CH — Selection of 2-wire system or 3-wire system

0	2-wire system
1	3-wire system (Do not set SB flag to 1.)

Reset	b_3	b_2	b_1	b_0
Power ON	0	0	0	0
Clock stop	0	0	0	0
CE	0	0	0	0

21.5.2 Setting of Each Pin by Serial I/O1 Mode Select Register

Table 21-2 shows the setting of each pin by the serial I/O1 mode select register.

As shown, setting of each pin requires operation of the input/output set flag of each pin.

Table 21-2 Setting of pins by serial I/O1 mode select register

SIOMODE							Pin						
b_3	b_2		b_1		b_0			Input/output setting flag of each pin					
SIO1CH	SB	Communication system	SIO1MS	Direction of clock	SIO1TX	Serial input/output	Pin symbol	POABIO3	POABIO2	POABIO1	POABIO0	POBBIO3	Pin setting status
0	0	2-wire serial I/O			0	Input (Reception)	$P0A_3$/SDA	0					Serial input (Hi-Z)
								1					General purpose output port
					1	Output (Transmission)		0					Serial output
								1					
			0	External			$P0A_2$/SCL		0				External clock (Hi-Z)
									1				General purpose output port
			1	Internal					0				Internal clock
									1				
							$P0A_1$/SCK_1						General purpose input/output port
							$P0A_0$/SO_1						General purpose input/output port
							$P0B_3$/SI_1						General purpose input/output port
0	1	2-wire serial bus			0	Input (Reception)	$P0A_3$/SDA	0					Serial input (Reception: Hi-Z)
								1					General purpose output port
					1	Output (Transmission)		0					Serial output (Transmission)
								1					
			0	External (slave)			$P0A_2$/SCL		0				External clock (slave)
									1				General purpose output port
			1	Internal (master)					0				Internal clock (Master)
									1				
							$P0A_1$/SCK_1						General purpose input/output port
							$P0A_0$/SO_1						General purpose input/output port
							$P0B_3$/SI_1						General purpose input/output port
							$P0A_3$/SDA						General purpose input/output port
							$P0A_2$/SCL						General purpose input/output port
1	0	3-wire serial I/O	0	External			$P0A_1$/SCK_1			0			External clock
										1			General purpose output port
			1	Internal						0			Internal clock
										1			
					0	Input (Reception)	$P0A_0$/SO_1				0		General purpose input/output port
											1		General purpose output port
					1	Output (Transmission)					0		Serial output
											1		
							$P0B_3$/SI_1					0	Serial input
												1	General purpose output port
1	1						"Setting inhibited"						

21.6 CLOCK GENERATION BLOCK

The clock generation block controls clock generation and clock output timing when the internal clock is used (master operation).

The frequency f_{SC} of internal clock is set by the serial I/O1 clock select register.

The shift clock issued by the clock generation block is valid only during the master operation (SIO1MS=1).

The shift clock is output continuously until the serial communication is waited by the wait condition to be mentioned later.

Section **21.6.1** explains the configuration and function of the serial I/O1 clock select register.

Sections **21.6.2** and **21.6.3** show the clock output waveform and generation timing for each communication system.

21.6.1 Configuration and Function of Serial I/O1 Clock Select Register

The configuration and function of the serial I/O1 clock select register are shown below.

Name	Flag b_3	Flag b_2	Flag b_1	Flag b_0	Address	Read/Write
Serial I/O1 clock select register (SIO1 CLK)	SIO1CK3	SIO1CK2	SIO1CK1	SIO1CK0	39H	R/W

Setting of internal shift clock frequency fsc of serial interface 1.	
0	0	75 kHz
0	1	150 kHz
1	0	225 kHz
1	1	450 kHz

Fixed at "0".

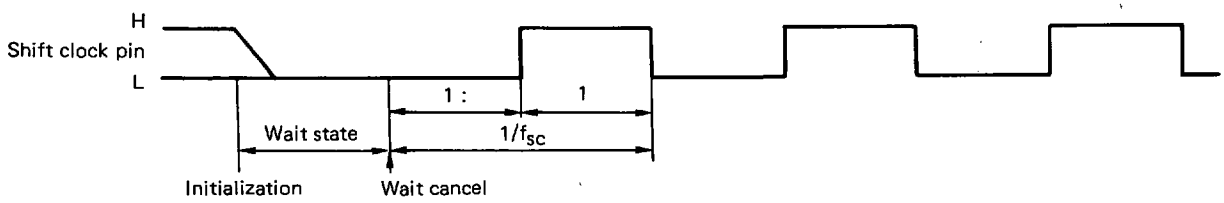
Reset				
	Power ON	0	0	Undefined
	Clock stop			Held
	CE			Held

21.6.2 Shift Clock Generating Timing for Serial Bus System

(1) When canceling wait status from initialized state

The initialized state means the state where the master operation with serial bus system is selected.

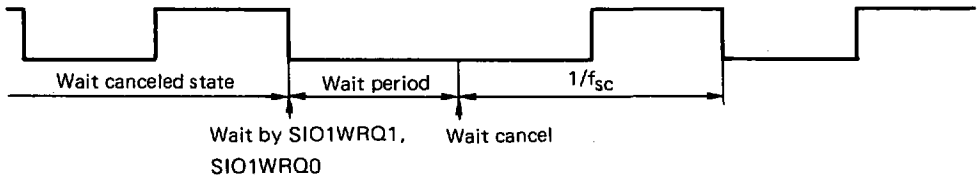
During the wait state, the shift clock pin is kept at low level.



(2) When wait operation is executed

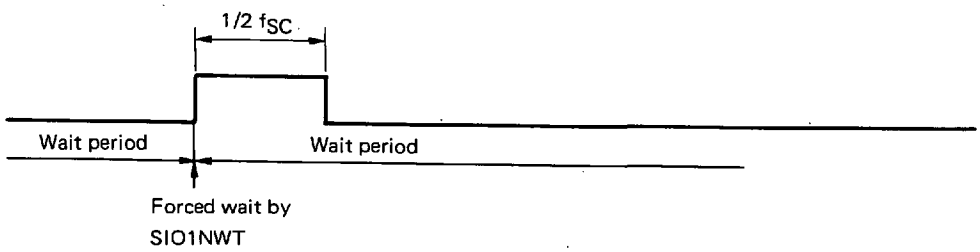
For the wait operation, see Section **21.9.**

(a) Ordinary wait by SIO1WRQ0 and SIO1WRQ1 flags



(b) Forced wait during a wait

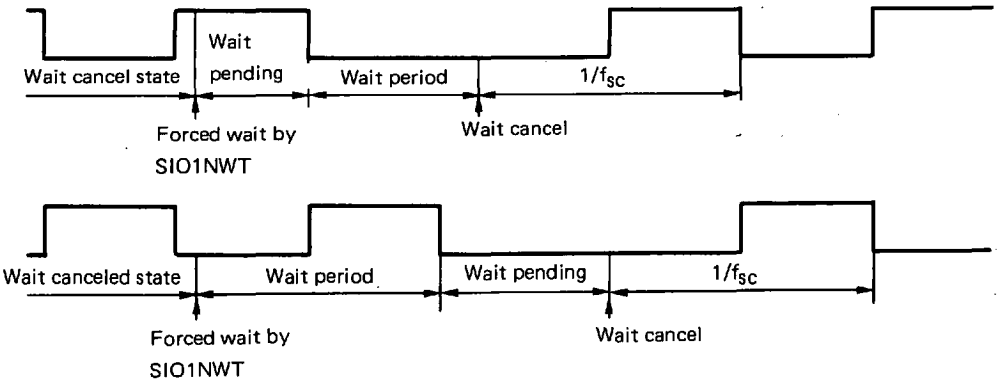
In this case, one clock pulse is output. (but, the clock counter and shift register stop their operation)



(c) Forced wait during wait cancel

The wait state is set at the falling edge of the clock coming next to the one where the forced wait was effected.

When the forced wait is effected, the clock counter and presettable shift register 1 stop their operation. When the forced wait is effected when the clock pin is held at low level, the clock counter and presettable shift resister 1 operate for one pulse.



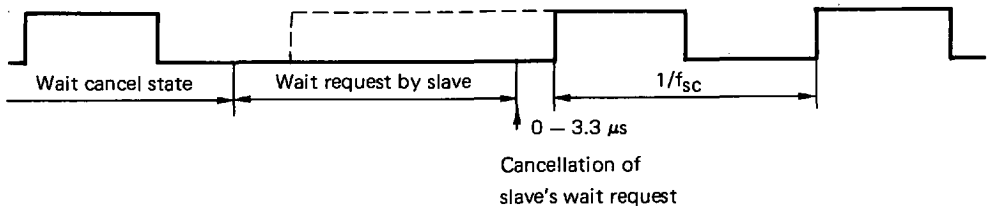
(d) Wait cancel during wait cancel state

No change occurs.

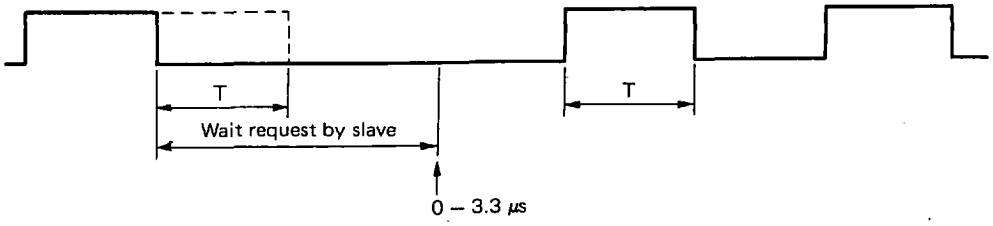
(e) When wait request is made by slave clock during wait cancel state

A clock is output 0 to 3.3 μs after the wait request by the slave is canceled.

- When the master is held at low level.



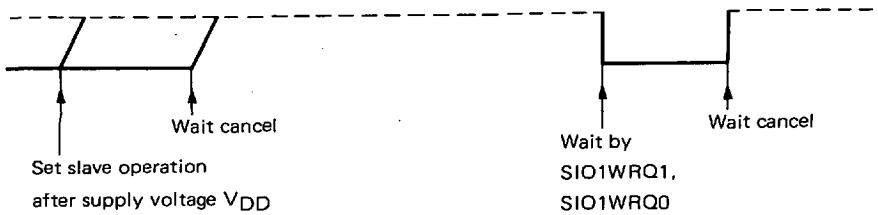
- When the master is held at high level



(3) When slave operation (external clock) is excuted.

The shift clock pin is undefined at the first setting up slave operating after supply voltage V_{DD}.

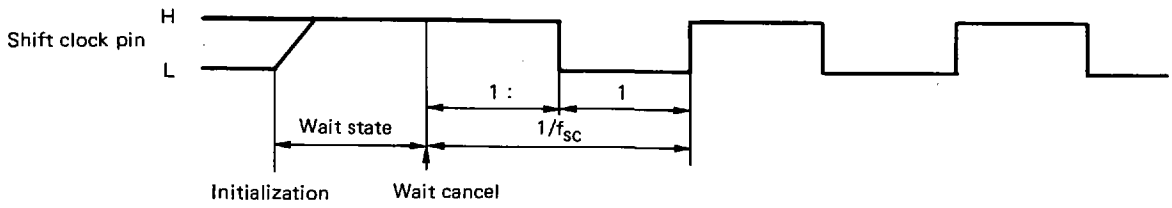
At this time if the shift clock pin is Hi-Z and input low level of external clock, it is fixed at low level.



21.6.3 Shift Clock Generation Timing for Serial I/O System

(1) Wait cancel from initialized state

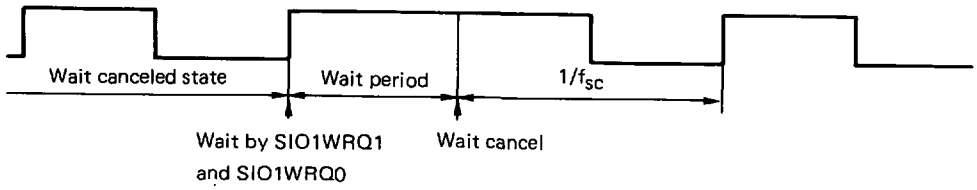
The initialized state means a state where the internal clock operation with serial I/O system is selected. During the wait state, the shift clock pin is held at high level.



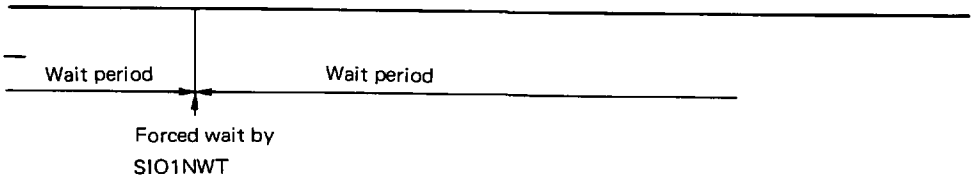
(2) When wait operation is effected

For the wait operation, see Section **21.9.**

(a) Ordinary wait by SIO1WRQ0 and SIO1WRQ1 flags

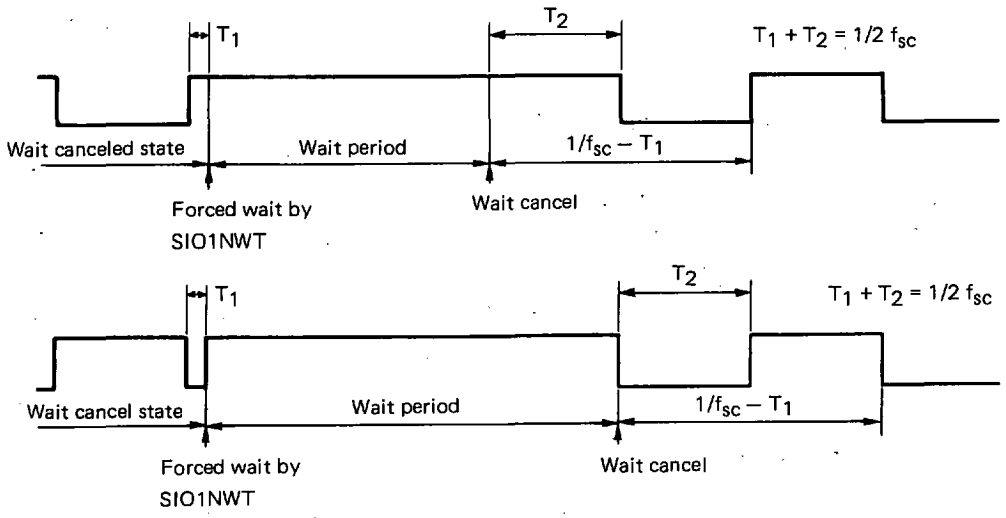


(b) Forced wait during a wait



Then the clock counter is reset.

(c) Forced wait during wait cancel



(d) Wait cancel effected during wait canceled state

In this case, no change occurs in the clock output waveform. Note that, however, the clock counter is reset.

21.7 CLOCK COUNTER AND START/STOP DETECTION BLOCK

The clock counter is a wraparound counter which counts the number of clocks issued at the shift clock pin ($P0A_2$/SCL pin for 2-wire system, and $P0A_1/\overline{SCK}_1$ pin for 3-wire system) of the shift clock pin selected.

The clock counter directly reads the state of the clock pin, hence it cannot discriminate whether the clock is issued internally or applied from the external source.

The clock counter does not operate during a wait of serial communication.

The contents of the clock counter can be detected via the SIO1SF8 and SIO1SF9 flags of the serial I/O1 status judge register, but cannot be read directly by a program.

The subsequent Sections **21.7.1** thru **21.7.4** explain the configuration and function of the serial I/O1 status judge register, operation of the clock counter and resetting of the clock counter.

The start/stop detection block is a block which is used to detect the start condition and stop condition when the serial bus system is used.

The start condition and stop condition can be detected by the SBSTT and SBBSY flags of the serial I/O1 status judge register.

The Section **21.7.5** indicates the operation of the SBSTT and SBBSY flags.

21.7.1 Configuration and Function of Serial I/O1 Status Judge Register

The serial I/O1 status judge register detects the clock counter of the serial interface 1 and the start/stop condition of the serial bus system.

The configuration and function of this register are shown below.

Name	Flag b_3	b_2	b_1	b_0	Address	Read/Write
Serial I/O1 status judge register (SIO1 STUS)	SIO1SF8	SIO1SF9	SBST	SBSY	28H	R

SBSY

Detection of start and stop conditions of serial bus system	
Serial bus system	Serial I/O system
0 Resetting (0) occurs if stop condition is detected.	"0" is held.
1 Setting (1) occurs if start condition is detected.	

SBST

Detection of clock counter and start condition of serial bus system	
Serial bus system	Serial I/O system
0 Resetting (0) occurs at the falling edge of the clock counter is "9".	The previous value is held.
1 Setting (1) occurs if start condition is detected.	

SIO1SF9

Detection of clock counter of serial interface 1
0 Resetting (0) occurs if clock counter turns to "0" or "1".
1 Setting (1) occurs if clock counter turns to "9".

SIO1SF8

Detection of clock counter of serial interface 1
0 Resetting (0) occurs if clock counter turns to "0" or "1".
1 Setting (1) occurs if clock counter turns to "0".

Reset	b_3	b_2	b_1	b_0
Power ON	0	0	0	0
Clock stop	0	0	0	0
CE	0	0	0	0

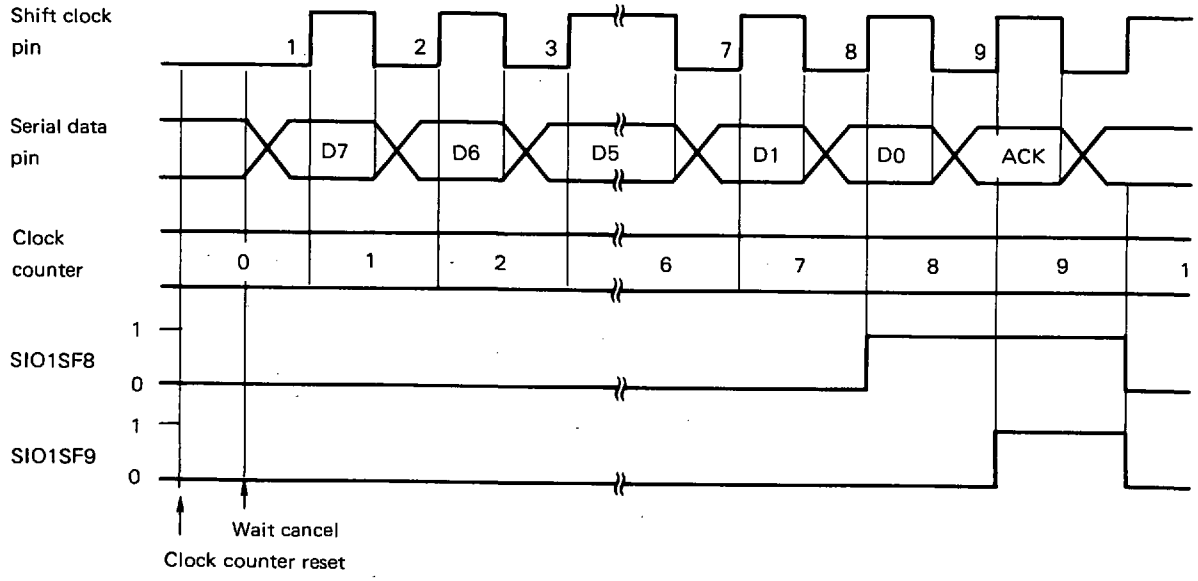
21.7.2 Operation of Clock Counter when Serial Pulse System is Used

The operation of the clock counter is explained below.

The initial value of the clock counter is "0". The counter value is increases by 1 upon each detection of rising edge of waveform at the clock pin. When counted to "9", the counter is reset and then counting begins with "1".

Flags SIO1SF8 and SIO1SF9 detect the state which causes the clock counter to count "8" and "9", respectively. These flags operate irrespective of the master (internal clock), slave (external clock), reception and transmission operations.

Fig. 21-3 Operation of clock counter when serial bus system is used



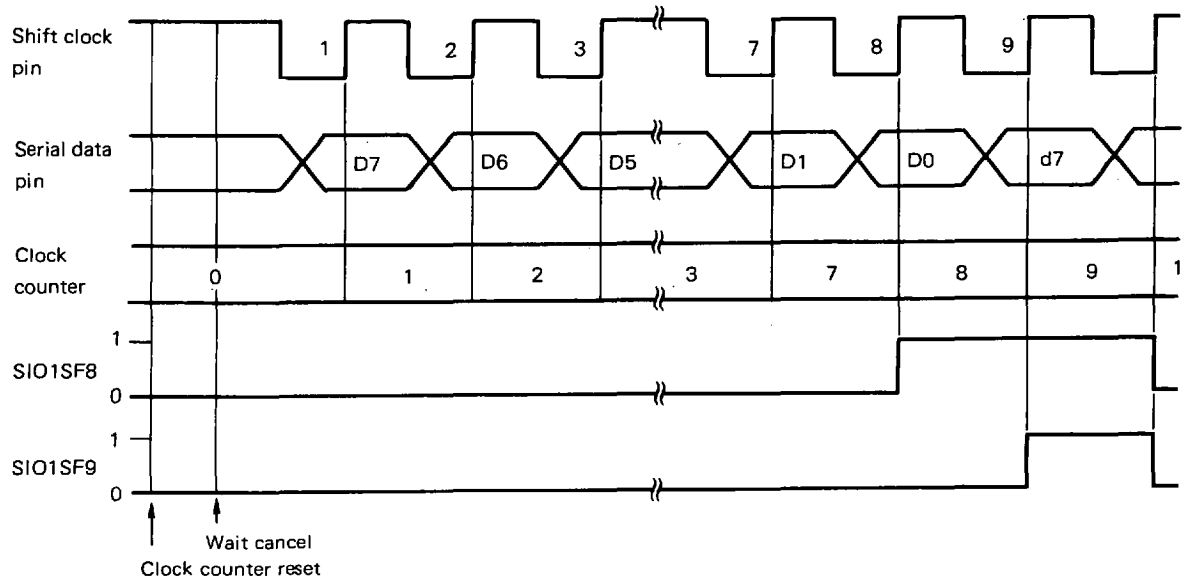
21.7.3 Operation of Clock Counter when Serial I/O System is Used

The operation of the clock counter is explained below.

The initial value of the clock counter is "0". The counter value is increases by 1 upon each detection of falling edge of waveform at the clock pin. When counted to "9", the counter is reset and then counting begins with "1".

Flags SIO1SF8 and SIO1SF9 detect the state which causes the clock counter to count "8" and "9", respectively. These flags operate irrespective of the master (internal clock), slave (external clock), reception and transmission operations.

Fig. 21-4 Operation of clock counter when serial I/O system is used



21.7.4 Clock Counter Reset (0) Condition

(1) 2-wire system serial bus

(a) Power ON
(b) When clock stop instruction is executed
(c) When start condition is detected
(d) When communication system is switched from 2-wire serial bus system to 2-wire or 3-wire serial I/O system
(e) CE reset

(2) 2-wire or 3-wire serial I/O system

(a) Power ON
(b) When clock stop instruction is executed
(c) When data write operation is effected to serial I/O1 wait control register
(d) When communication system is switched from 2-wire or 3-wire serial I/O system to 2-wire serial bus system
(e) CE reset

21.7.5 Operation of SBSTT and SBBSY Flags

The operation of the SBSTT and SBBSY flags is explained below.

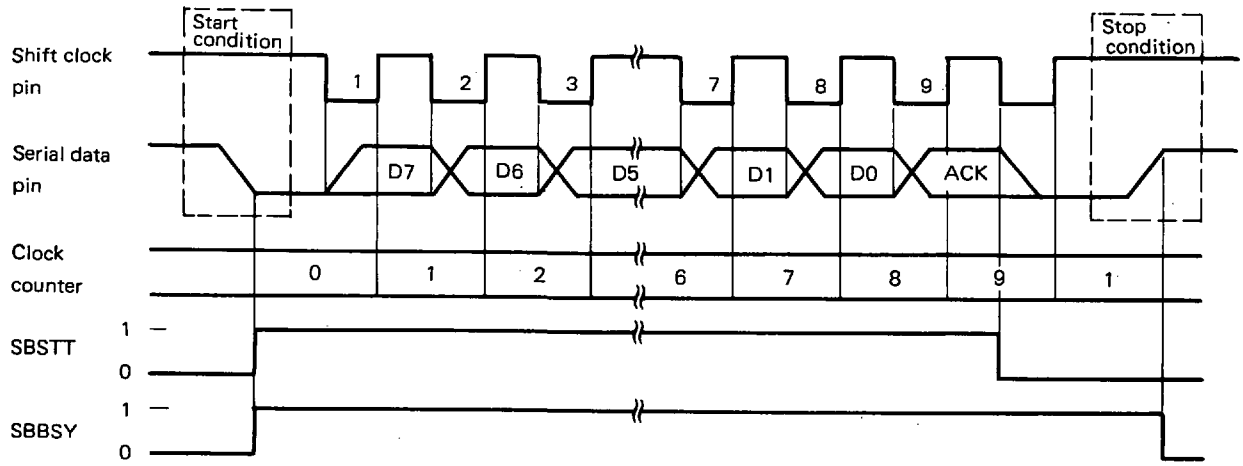
The SBSTT and SBBSY flags operate only when the serial bus system is used.

The communication state of other station can be detected by detecting these flags.

These flags operate irrespective of the master, slave, reception, transmission, during wait, and wait cancel states.

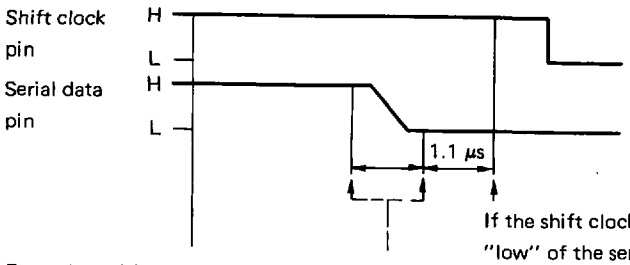
When the serial I/O system is used, the value "0" is held.

Fig. 21-5 Operation of SBSTT and SBBSY flags



The start condition and stop condition are fetched at the timing shown in (1) and (2) below.

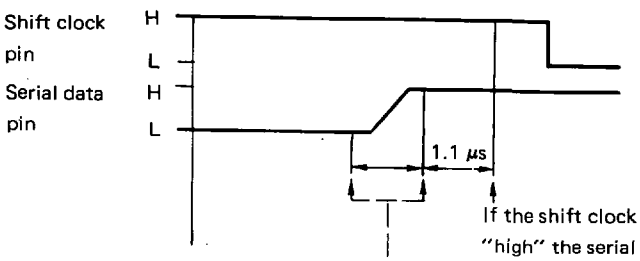
(1) Start condition fetch timing



Detection of falling edge of serial data pin level

If the shift clock pin is at high level 1.1 μs after detecting "low" of the serial data pin level, this is regarded as the start condition.

(2) Stop condition fetch timing



If the shift clock pin is at high level 1.1 μs after detecting "high" the serial data pin level, this is regarded as the stop condition.

Detection of rising edge of serial data pin level

21.8 PRESETTABLE SHIFT REGISTER (PSR)

The presettable shift register 1 is an 8-bit shift register for writing serial out data and reading serial in data.

Data writing to and data reading from the presettable shift register are performed by "PUT" instruction and "GET" instruction via the data buffer.

Section **21.8.1** explains the configuration of the presettable shift register and its relation to the data buffer.

The data shift operation of the presettable shift register 1 is performed in synchronization with the clock applied to the shift clock pin selected at that time ($P0A_2$/SCL pin for 2-wire system, and $P0A_1/\overline{SCK_1}$ pin for 3-wire system).

With the serial bus system, the most significant bit (MSB) of the presettable shift register 1 is output (transmission operation) to the serial data pin in synchronization with the falling edge of the shift clock, while the data of the serial data pin is read in to the least significant bit (LSB) of the presettable shift register 1 in synchronization with the rising edge of the clock.

The Sections **21.8.2** and **21.8.3** show operation of the register and precautions when using the serial bus system and serial I/O system.

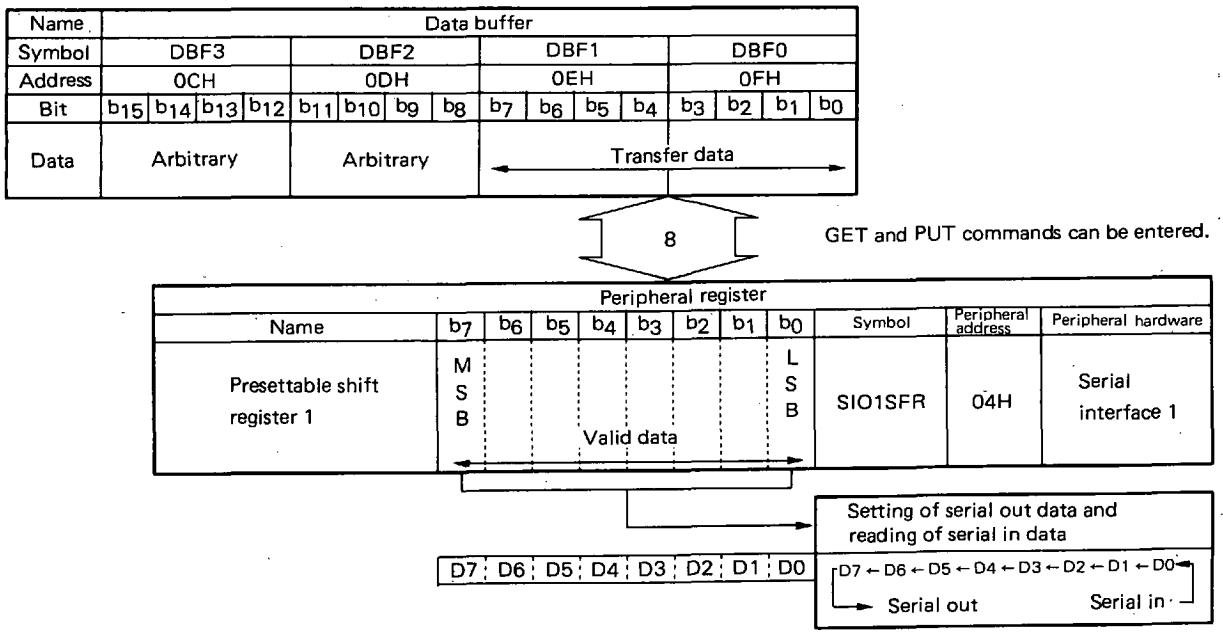
Section **21.8.4** explains precautions in writing data into and reading data from the presettable shift register 1.

During the wait state, the presettable shift register performs no data shift operation.

For details of the operation when each serial communication system is used, see Section **21.11.**

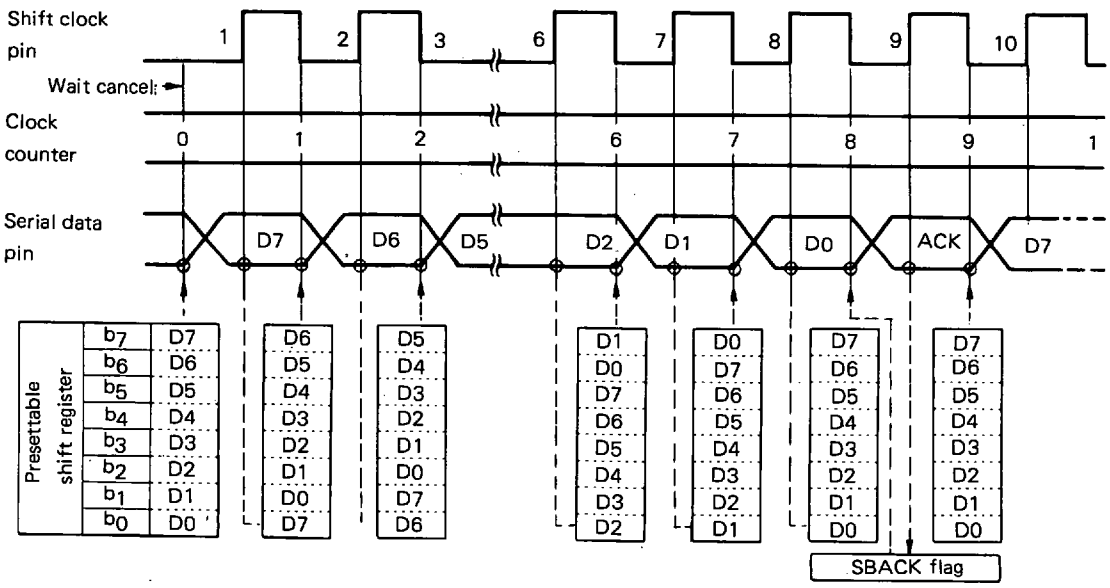
21.8.1 Configuration of Presettable Shift Register 1 and Its Relation with Data Buffer

The configuration of the presettable shift register 1 and its relation with the data buffer are shown below.



21.8.2 Operation of Presettable Shift Register 1 when Using Serial Bus System

Explained below is the data shift operation when the serial bus system is used.

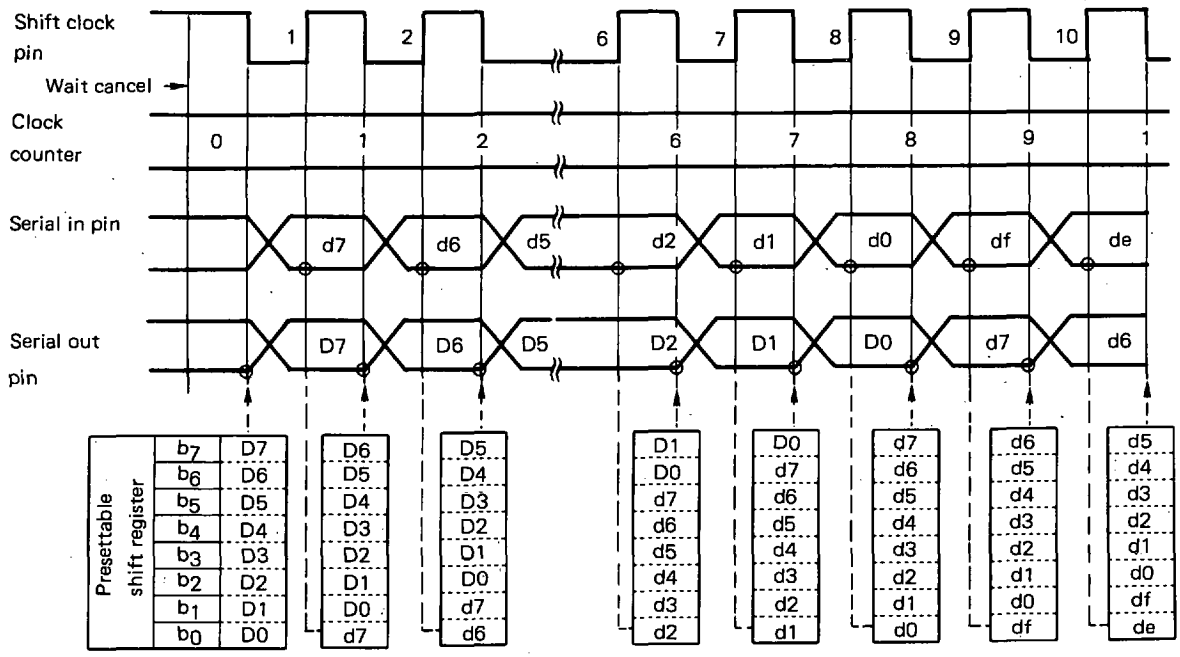


Data shift operation in reception and transmission

Serial bus system	
Receiving operation	Transmitting operation
The status of $P0A_3$/SDA is shifted from LSB at the rising edge of shift clock pin waveform for entering data. No output is made. The content of SBACK flag is output at the falling edge of shift clock when the clock counter value is 8. No operation is performed during the wait state.	The MSB of data is shifted to $P0A_3$/SDA pin at the falling edge of the shift clock pin waveform. The status of $P0A_3$/SDA is entered from LSB at the rising edge of the shift clock pin. The status of $P0A_3$/SDA pin is entered to SBACK flag at the rising edge of the shift clock when the clock counter turns to 9. No operation is performed during the wait state.

21.8.3 Operation with Serial I/O System

The data shift operation is explained below when serial I/O system is used.



Data shift operation in reception and transmission

Serial bus system	
Receiving operation	Transmitting operation
The status of $P0A_3$/SDA TERMINAL ($P0B_3/SI_1$ pin for 3-wire system) is shifted from LSB at the rising edge of the shift clock pin waveform for entering data. No output is made.	MSB of data is shifted to $P0A_3$/SDA pin ($P0A_0/SO_1$ pin for 3-wire system) at the falling edge of the shift clock pin waveform. The status of $P0A_3$/SDA pin ($P0B_3/SI_1$ pin for 3-wire system) is entered from LSB at the rising edge of the shift clock pin waveform.

21.8.4 Precautions when Setting and Reading Data

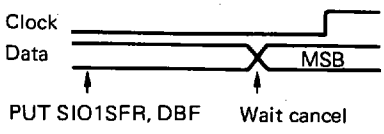
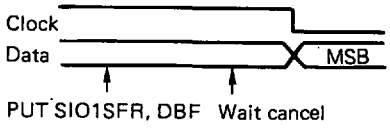
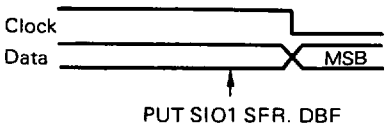
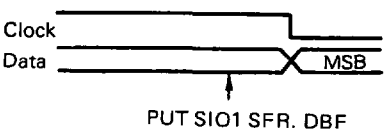
Setting of data to the presettable shift register 1 is performed by "PUT SIO1SFR, DBF" instruction.

Data reading is performed by "GET DBF, SIO1SFR" instruction.

Data setting and reading must be performed during the wait state. During the wait canceled state, this data setting and reading may fail depending on the status of the shift clock pin.

The data setting and reading timing and precautions are explained below.

Table 21-3 Data reading (GET) and data writing (PUT) operations of presettable shift register 1 and precautions

<table>
<tr><th colspan="2">Status when executing PUT/GET</th><th>Status of shift clock pin</th><th>Serial bus system</th><th>Serial I/O system</th></tr>
<tr><td rowspan="2">Wait state</td><td>Reading (GET)</td><td rowspan="2">When using serial bus system: Fixed at low level
When using serial I/O system: Fixed at high level</td><td>Normal read</td><td>Normal read</td></tr>
<tr><td>Writing (PUT)</td><td>Normal write
The content of MSB is output when the wait is canceled next (Transmission operation)
 
</td><td>Normal write
The content of MSB is output at the falling edge of shift clock pin waveform when the wait is canceled next. (Transmission operation)
 
</td></tr>
<tr><td rowspan="3">Wait cancel state</td><td rowspan="2">Reading (GET)</td><td>Low level</td><td>Normal reading</td><td>Normal reading</td></tr>
<tr><td>High level</td><td>Normal reading</td><td>Unable to read normally, or content of PSR1 is destroyed.</td></tr>
<tr><td>Writing (PUT)</td><td>High level</td><td>Normal writing
The content of MSB is output upon execution of PUT instruction.
The clock counter is not reset.
 
</td><td>Normal writing
The content of MSB is output upon execution of PUT instruction.
The clock counter is not reset.
 
</td></tr>
<tr><td>Low level</td><td>Unable to write normally, or content of PSR1 is destroyed.</td><td>Unable to write normally, or content of PSR1 is destroyed.</td></tr>
</table>

21.9 WAIT BLOCK AND ACKNOWLEDGE BLOCK

The wait block controls the pause (wait) and cancel of communication of the serial interface 1.

The acknowledge block outputs and detects the response signal when the serial bus system is used.

The wait block and acknowledge block are controlled by the serial I/O1 wait control register.

Section **21.9.1** explains the configuration and function of the serial I/O1 wait control register.

Sections **21.9.2** thru **21.9.4** outline the wait operation, explain the wait operation and precautions in each communication system, and Section **21.9.5** explains the acknowledge block.

21.9.1 Configuration and Function of Serial I/O1 Wait Control Register

The configuration and function of the serial I/O1 wait control register are shown below.

Name	Flag b_3	Flag b_2	Flag b_1	Flag b_0	Address	Read/Write
Serial I/O wait control register (SIO1WT)	SBACK	SIO1NWT	SIO1WRQ1	SIO1WRQ0	18H	R/W

Setting of wait condition

SIO1WRQ1	SIO1WRQ0	Name	2-wire serial bus system	Serial I/O system
0	0	No wait	Setting inhibition	No wait is effected.
0	1	Data wait	Wait is effected at the falling edge of shift clock when clock counter is "8".	Wait is effected at the rising edge of shift clock when clock counter is "8".
1	0	Acknowledge wait	Wait is effected at the falling edge of shift clock when clock counter is "9".	Wait is effected a the rising edge of shift clock when clock counter is "9".
1	1	Address wait	Wait is effected at the falling edge of shift clock when clock counter is "8" after detecting the start condition.	Setting inhibition.

Setting of wait and detection of wait state

	When written into flag	When reading flag
0	Forced wait is effected.	Wait is effected by the condition of SIO1WRQ1, 0 flags.
1	Wait is canceled (serial communication starts).	Serial communication

Setting and detection of response signal (acknowledge) of serial bus system

	Serial bus system		Serial I/O system
	When receiving (SIO1TX = 0)	When sending (SIO1TX = 1)	
0	"0" is output as acknowledge	Slave acknowledge is detected. Acknowledge is "0".	No operation. This one bit can be used as 1-bit flag, etc.
1	"1" is output as acknowledge	Slave acknowledge is detected. Acknowledge is "1".	

Reset	b_3	b_2	b_1	b_0
Power ON	0	0	0	0
Clock stop	0	0	0	0
CE	0	0	0	0

21.9.2 Outline of Wait Operation

The wait state means a state where the clock generation block, presettable shift register 1, etc. stop their operation, and the serial communication is suspended.

When the wait state if canceled, serial communication operation is started.

Wait state is canceled by writing "1" into SIO1NWT flag.

When "1" is written into the SIO1NWT flag, the internal clock is output to the shift clock output pin (when master is operating), and the presettable shift register 1 and clock counter start operation.

When the condition set by SIO1WRQ0 and SIO1WRQ1 flags is satisfied, the wait cancel state turns into the wait state. In this case, the SIO1NWT flag is reset (0) automatically.

The operation status of serial communication can be known by detecting the content of SIO1NWT flag while the wait is canceled.

After starting the serial communication by writing "1" to SIO1NWT flag, the data can be read or set by detecting the SIO1NWT flag turning to "0".

Note that however, there is a certain time difference between turning "0" of SIO1NWT flag and actual creation of wait state.

This means that correct data setting and reading may fail if data setting (PUT instruction) or data reading (GET instruction) is executed to the presettable shift register 1 during the wait canceled state. See Section **21.8.**

Writing of "0" to SIO1NWT flag during the wait cancel state causes the wait state to be established. This is called as "forced wait".

When using the forced wait with the serial bus system, attention should be made to the time difference between writing of "0" to SIO1NWT flag and actual creation of the wait status.

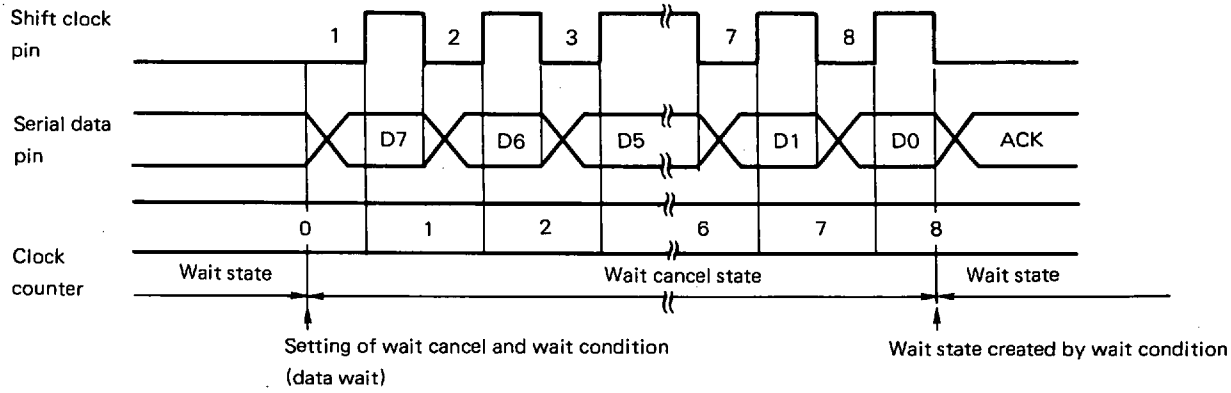
When the master of the serial bus system is operating, writing of "0" to SIO1NWT flag causes one shift clock pulse to be output. Note that, in such a case, the clock counter and presettable shift register 1 are not operating.

When the serial I/O system is used, writing of "1" to SIO1NWT flag during the wait cancel state, the clock counter is reset (0).

21.9.3 Wait Operation and Precautions when Serial Bus System is Used

(1) Wait operation when serial bus system is used

The data wait operation (SIO1WRQ1=0, SIO1WRQ0=1) with serial bus system is explained below.



When the wait state is canceled, the serial data is output (sending operation) at the time of cancellation, and the wait cancel state continues until the condition set by SIO1WRQ1 and SIO1WRQ0 flags is met.

When the wait condition is met, the shift clock pin is turned low, thus stopping operation of the clock counter and preset shift register 1.

Note that data will not be set correctly if data writing the presettable shift register 1 is performed during a wait cancel state with "low" shift clock pin level.

If data is written into the presettable shift register 1 during the period where the wait is canceled and shift clock pin is at high level, then the content of MSB is output to the serial data output pin at the falling edge of the shift clock coming next to the one when the "PUT" instruction is executed.

If forced wait is effected while the wait is canceled, the wait state is created at the falling edge of the clock coming after the clock at which "0" is written into SIO1NWT flag.

If wait cancellation is effected during a wait cancel state, no change will be caused.

Note that one pulse of shift clock will be issued if forced wait is effected during a wait.

When using the serial bus system, do not set the data wait condition (SIO1WRQ1=0, SIO1WRQ0=1) in succession.

This means that, if wait is canceled by setting twice the data wait condition in succession, the wait state is created at the moment the second wait is canceled.

Accordingly, after setting a data wait, it is necessary to set newly a different wait condition.

When using the serial bus system as shown in the subsequent paragraphs (2) and (3), attention should be paid to the existence of a period during which the SIO1NWT flag state does not agree with the actual communicating operation.

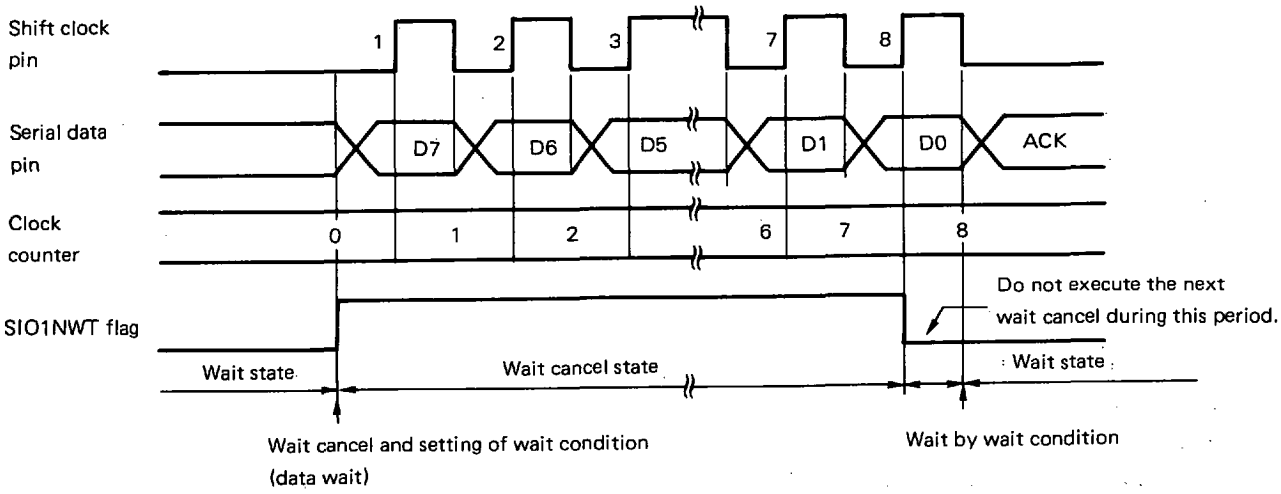
(2) Normal wait operation with serial bus system

With serial bus system, communication is put into a wait at the falling edge of the shift clock if the wait condition is met by SIO1WRQ1 and SIO1WRQ0 flags.

In this case, the SIO1NWT flag is reset at the rising edge which precedes the falling edge by 1/2 clock as shown below.

For example when the acknowledge is output, this indicates that the stop condition may be formed if the next wait cancel is set immediately after turning "0" of the SIO1NWT flag.

To avoid this, the data must be read or written after "0" is read from the SIO1NWT flag and then the low level is detected at the shift clock pin.



(3) Forced wait operation with serial bus system

If "0" is written into SIO1NWT flag during the wait cancel state of serial bus system, a forced wait will result at the falling edge of the next clock.

Accordingly, like the case (1) above, the next wait must be canceled and soon after detecting the low level at the shift clock pin.

When the master is operating for reception, one shift clock pulse will be output when forced wait is effected during a wait.

Attention should be taken to this point when setting the response signal shown in Section **21.9.5.**

(4) Wait request by slave

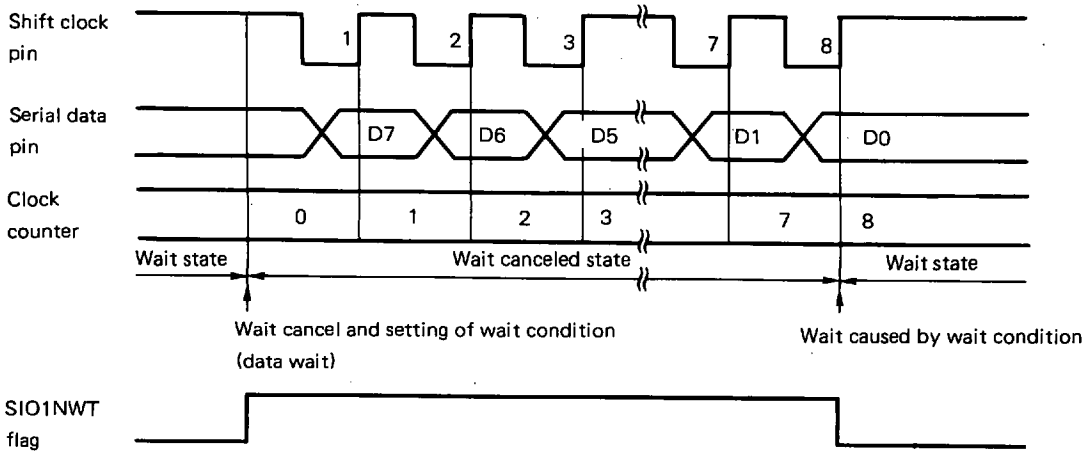
When the master is operating, the SIO1NWT flag will be reset to "0" if the shift clock output pin level is set low by force from an external source while it is issuing high level signal.

In such a case, the SIO1NWT flag is set ("1") at the time when the wait request by slave is canceled, and the operation is resumed.

21.9.4 Wait Operation when Using Serial I/O System and Precautions

(1) Wait operation with serial I/O system

An example of data wait operation (SIO1WRQ1=0, SIO1WRQ0=1) with serial I/O system is explained below.



When the wait is canceled, the serial data is output (sending operation) at the falling edge of the next clock pulse, and the wait canceled state is maintained until the condition set by SIO1WRQ1 and SIO1WRQ0 flags is met.

When the wait condition is met, the shift clock pin level is set high, and this stops operation of the clock counter and presettable shift register 1.

Note that correct data will not be set if data write of the presettable shift register 1 is performed while the wait is canceled and the shift clock pin level is low.

If data is written into the presettable shift register 1 while the wait is canceled and the shift clock pin level is high, the content of MSB will be output to the serial data output pin at the falling edge of the shift clock coming next to the one when the "PUT" instruction is executed.

If a forced wait is effected while a wait is canceled, the wait state will be created at once upon writing in of "0" to the SIO1NWT flag.

Note that the clock counter will be reset if wait is canceled again while a wait is already canceled.

21.9.5 Acknowledge Block and Operation

The acknowledge block operates only with the serial bus system.

This block is used to output the response signal in receiving operation with serial bus system and to detect the response signal in sending operation.

While receiving, the content of SBACK flag is output to the serial data pin at the falling edge of the shift clock when the clock counter is "8". (The serial data pin change the output port automatically.)

While receiving, the data once set to the SBACK flag will be held afterward.

When sending, the state of the serial data pin is read into the SBACK flag at the rising edge of the shift clock when the clock counter is "9". (The serial data pin change the input port automatically.)

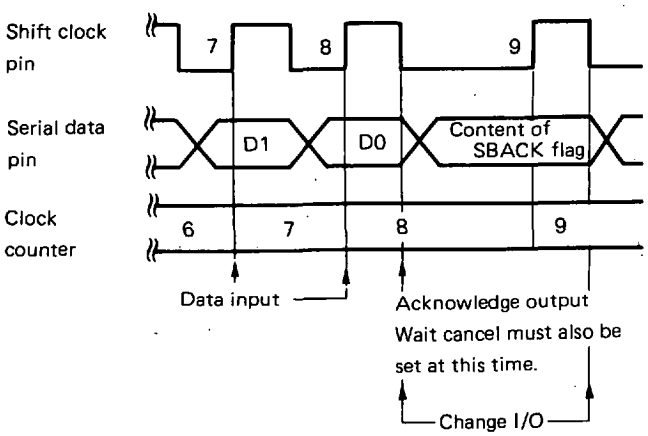
The acknowledge output operation and input operation are shown below.

When receiving, the setting of acknowledge (setting of SBACK flag) must be performed simultaneously with cancellation of wait (setting of SIO1NWT flag).

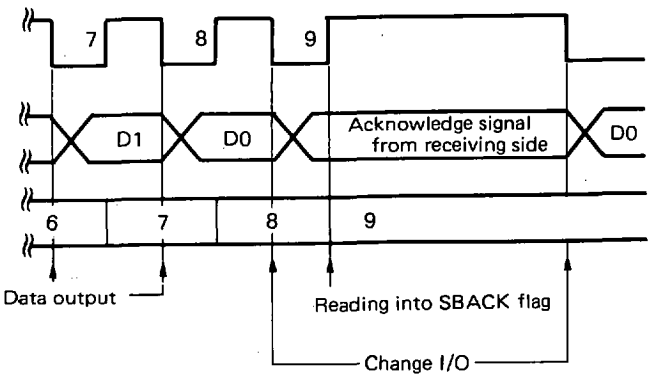
The reason is that SBACK flag and SIO1NWT flag are allocated in the same register address, and setting of the SBACK flag alone also causes the SIO1NWT flag to be set at the same time. In this case, if the serial communication is kept in the wait state at this time, this setting of both flags results in the execution of a forced wait during the wait state, and this causes one extra shift clock pulse to be issued.

When the serial I/O system is used, the SBACK flag can be used as the one-bit general purpose flag.

Receiving operation



Sending operation



When the acknowledge, Hi-Z output or input, the acknowledge wait (wait is effected at the falling edge of shift clock when clock counter is "9") must be set certainly.

21.10 INTERRUPT CONTROL BLOCK

The interrupt control block is issues interrupt request of serial interface 1, and the serial I/O interrupt mode register sets the condition for issuing this interrupt request.

If the interrupt request issue condition is met, IRQSIO1 flag of the interrupt request 2 register (RF address 3FH) is set (turned to "1").

Section **21.10.1** shows the configuration and function of the serial I/O1 interrupt mode register.

Sections **21.10.2** and **21.10.3** show the interrupt request issue timing of each communication system.

21.10.1 Configuration and Function of Serial Interface 1 Interrupt Mode Register

The function of each flag of the serial interface 1 interrupt mode register is shown below.

Do not re-write these flags during serial communication (while SIO1NWT flag is "1"). The flag must be re-written after writing "0" into SIO1NWT flag, or when the SIO1NWT flag is "0".

Because interrupt request may be issued as soon as the mode is changed to during serial communication.

Name	Flag b_3	Flag b_2	Flag b_1	Flag b_0	Address	Read/Write
Serial I/O1 interrupt mode register (SIO1INT)	SIO11MD3	SIO11MD2	SIO11MD1	SIO11MD0	38H	R/W

SIO11MD3	SIO11MD2	Setting of interrupt request issue condition: Serial bus system	Setting of interrupt request issue condition: Serial I/O system
0	0	Rising edge of shift clock when clock counter turns to "7".	Rising edge of shift clock when clock counter turns to "7". *1
0	1	Rising edge of shift clock when clock counter turns to "8".	Rising edge of shift clock when clock counter turns to "8". *2
1	0	Rising edge of shift clock when clock counter turns "7" after start condition is detected. *3	No interrupt request is issued.
1	1	When stop condition is detected. *4	No interrupt request is issued.

SIO11MD1, SIO11MD0: Fixed to "0"

Reset		SIO11MD3	SIO11MD2	SIO11MD1	SIO11MD0
	Power ON	0	0	Undefined	
	Clock stop				Held
	CE				Held

*1 This mode is changed to and so interrupt request is issued when clock counter is "7".

*2 This mode is changed to and so interrupt request is issued when clock counter is "8".

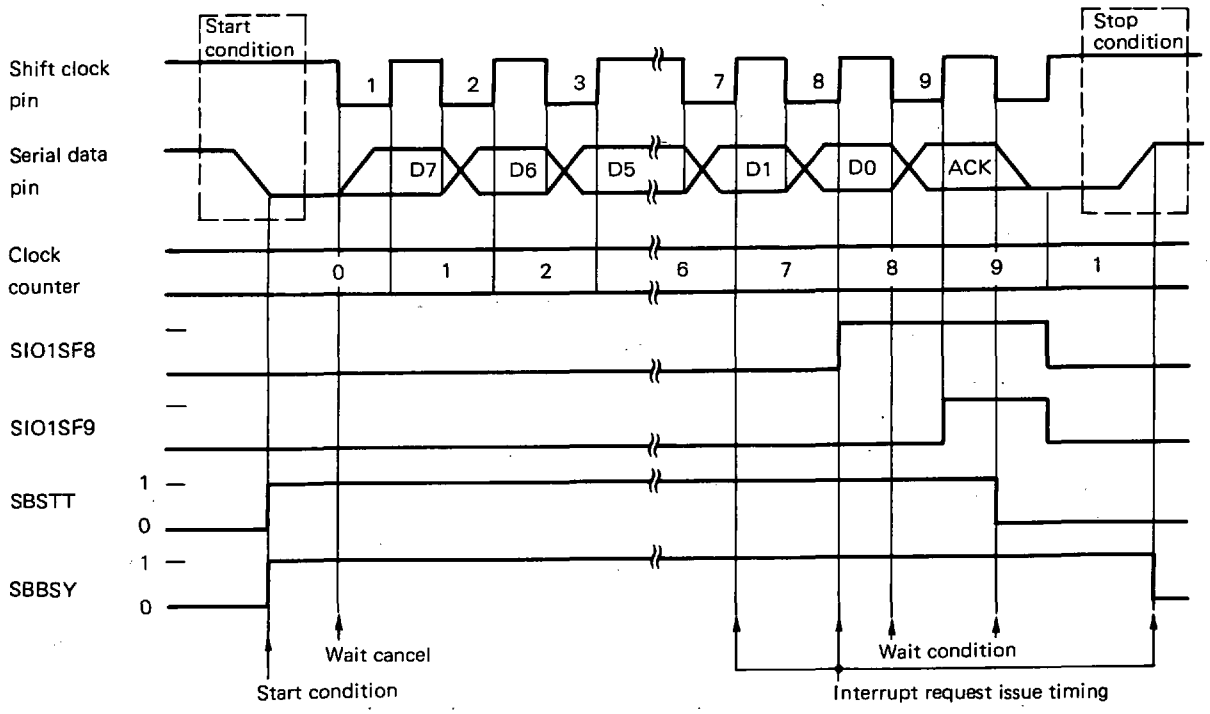
*3 When SBSTT flag is "1" and clock counter is "7", this mode is changed to and so interrupt request is issued.

*4 After stop condition is issued, this mode is change to and so interrupt request is issued.

21.10.2 Interrupt Request Issue Timing with Serial Bus System

The interrupt request issue timing for the serial bus system is indicated below.

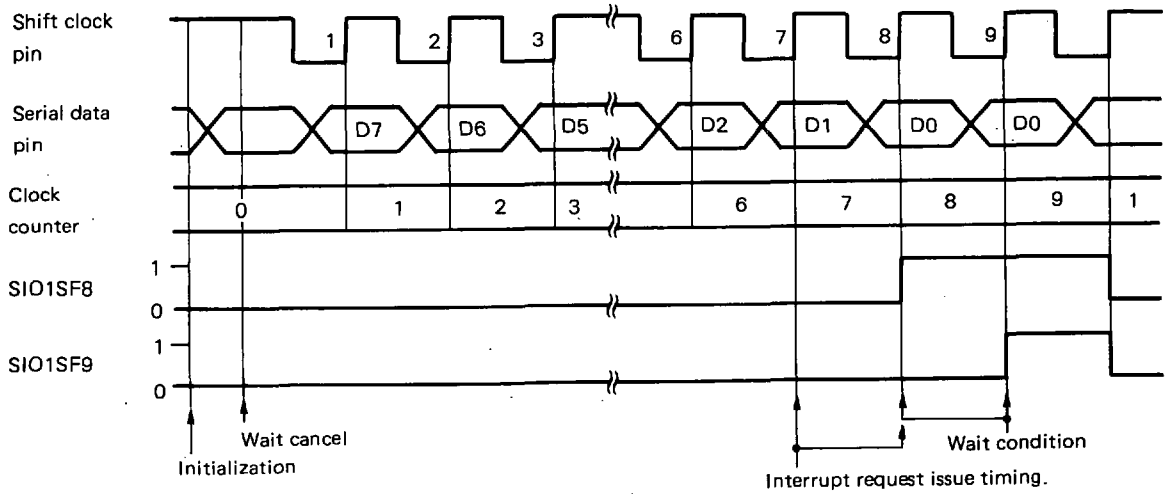
Fig. 21-6 Interrupt request issue timing for serial bus system



21.10.3 Interrupt Request Issue Timing for Serial I/O System

The interrupt request issue timing for serial I/O system is shown below.

Fig. 21-7 Interrupt request issue timing for serial I/O system



21.11 HOW TO USE SERIAL INTERFACE 1

21.11.1 Usage of 2-wire Serial Bus System

The serial bus system is selected by setting SIO1CH flag and SB flag to "0" and "1", respectively.

The serial bus system uses $P0A_3$/SDA pin and $P0A_2$/SCL pin.

Fig. 21-8 shows the input/output block and communication method of the serial bus system.

Table 21-8 shows the function and operation of each pin and control register when the serial bus system is used.

As shown in Fig. 21-8 and Table 21-4, the serial bus system involves master operation and slave operation, and each operation permits transmission (TX) and reception (RX).

The master and slave operation are selected by SIO1MS flag, while reception and transmission are selected by SIO1TX flag.

During the master operation, the internal shift clock is output from $P0A_2$/SCL pin. If transmission mode is selected, data is output from $P0A_3$/SDA pin at the falling edge of the shift clock. If reception mode is selected, the status of $P0A_3$/SDA pin is entered into the presettable shift register 1 at the rising edge of the shift clock.

In both the master and slave operation modes, the serial communication start condition and stop condition can be detected by SBSTT and SBBSY flags.

The start condition and stop condition are usually output from the master, and this output is controlled by program (each pin is controlled as general purpose output port).

In the slave operation mode, the external clock is waited with $P0A_2$/SCL pin set in the floating state. If transmission is selected, data is output from $P0A_3$/SDA pin at the falling edge of the shift clock. If reception is selected, the status of $P0A_3$/SDA pin is entered into the presettable shift register 1 at the rising edge of the clock applied to $P0A_2$/SCL pin.

During the receive operation in master or slave mode, acknowledge signal is output for each of 8-bit data is received.

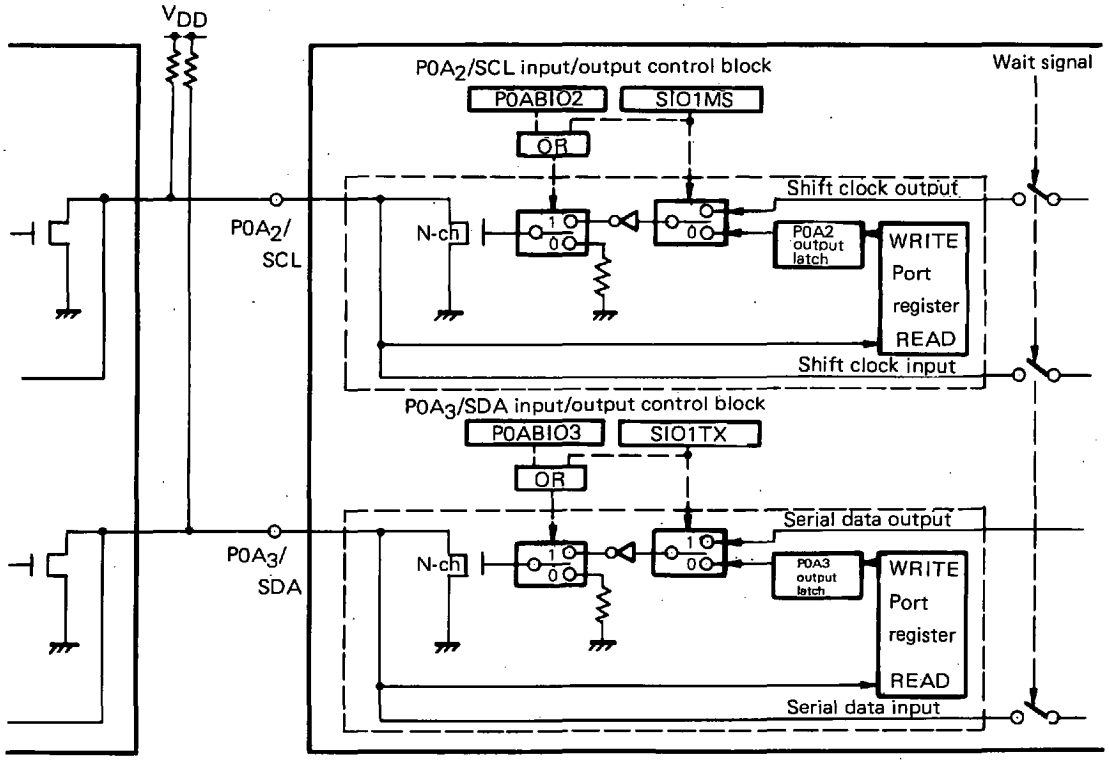
During the transmit operation in master or slave mode, acknowledge is detected for each of 8-bit data is transmitted.

The $P0A_3$/SDA pin and $P0A_2$/SCL pin are use the N-ch open drain output, and if either the master or slave issues low level, the communication line is set at low level.

The $P0A_3$/SDA pin or $P0A_2$/SCL pin, when it reads the value output to the pin, reads in directly the "status of the pin at that moment".

Fig. 21-8 Input/output block and communication method of 2-wire serial bus system

Input/output block



Communication method

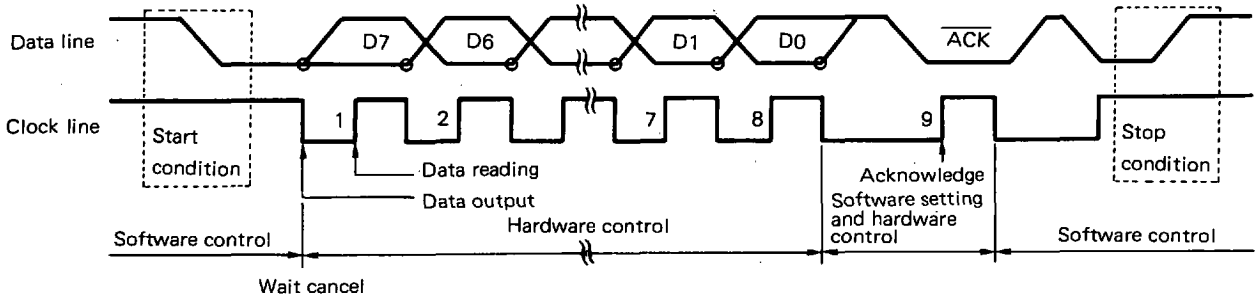


Table 21-4 Outline of operation of 2-wire serial bus system (1/2)

Item \ Operation mode		2-wire serial bus system			
		Slave operation		Master operation	
		Reception (RX) (SIO1TX = 0	Transmission (TX) SIO1TX = 1	Reception (RX) SIO1TX = 0	Transmission (TX) SIO1TX = 1
Pin setting state	$P0A_3$/SDA	**P0ABIO3 = 0** Floating In wait for external data input **P0ABIO3 = 1** General purpose output port Content of output latch is output. Usually P0ABIO3 is set "0".	Irrespective of the value of P0ABIO3, content of PSR1 is output at the falling edge of external clock.	**P0ABIO3 = 0** Floating In wait for external data input **P0ABIO3 = 1** General purpose output port Content of output latch is output. Usually P0ABIO3 is set "0".	Irrespective of the value of P0ABIO3, content of PSR1 is output at the falling edge of external clock.
	$P0A_2$/SCL	**P0ABIO2 = 0** Floating In wait for external data input **P0ABIO2 = 1** General purpose output port Content of output latch is output. Usually P0ABIO2 is set "0".	**P0ABIO2 = 0** Floating In wait for external data input **P0ABIO2 = 1** General purpose output port Content of output latch is output. Usually P0ABIO2 is set "0".	Irrespective of the value of P0ABIO2, internal shift clock is output.	Irrespective of the value of P0ABIO2, internal shift clock is output.
Clock counter operation		Increment at the rising edge of SCL pin			

Table 21-4 Outline of operation of 2-wire serial bus system (2/2)

Operation mode / Item	2-wire serial bus system			
	Slave operation		Master operation	
	Reception (RX) SIO1TX = 0	Transmission (TX) SIO1TX = 1	Reception (RX) SIO1TX = 0	Transmission (TX) SIO1TX = 1
Operation of presettable shift register 1 (PSR1)	**Output** No output **Input** Data of SDA pin is entered by shifting from LSB at each rising edge of SCL pin waveform.	**Output** Data is output to SDA pin by shifting from MSB at each falling edge of SCL pin waveform. **Input** Data of SDA pin is entered by shifting from LSB at each rising edge of SCL pin waveform.	**Output** No output **Input** Data of SDA pin is entered by shifting from LSB at each rising edge of SCL pin waveform	**Output** Data is output to SDA pin by shifting from MSB at each falling edge of SCL pin waveform. **Input** Data of SDA pin is entered by shifting from LSB at each rising edge of SCL pin waveform.
Wait operation	Serial communication starts with writing of "1" to SIO1NWT. SIO1NWT is reset to "0" by the conditions set in SIO1WRQ1 and SIO1WRQ0.			
	SIO1NWT = 0 Low level is forcibly output from SCL pin. SDA pin is floating. **SIO1NWT = 1** In wait for external clock with SCL pin set floating. SDA pin is set floating and data of SDA pin is entered to PSR1 at the rising edge of SCL pin waveform.	**SIO1NWT = 0** Low level is forcibly output from SCL pin. SDA pin is floating. **SIO1NWT = 1** In wait for external clock with SCL pin set floating. Content of PSR1 is output to SDA pin at the falling edge of SCL pin waveform.	**SIO1NWT = 0** Low level is forcibly output from SCL pin. SDA pin is floating. **SIO1NWT = 1** In wait for external clock with SCL pin set floating. SDA pin is set floating, and data of SDA pin is enterd to PSR1 at the rising edge of SCL pin waveform.	**SIO1NWT = 0** Low level is forcibly output from SCL pin. SDA pin is floating. **SIO1NWT = 1** In wait for external clock with SCL pin set floating. Content of PSR1 is output to SDA pin at the falling edge of SCL pin waveform.
Acknowledge	When clock counter value is "8", the content of ACK flag is output from SDA pin at the falling edge of SCL pin waveform.	The status of SDA pin is written into ACK flag at the rising edge of SCL pin waveform when clock counter value turns to "9".	When clock counter value is "8", the content of ACK flag is output from SDA pin at the falling edge of SCL pin waveform.	The status of SDA pin is written into ACK flag at the rising edge of SCL pin waveform when clock counter value turns to "9".

21.11.2 Usage of 2-wire Serial I/O System

The 2-wire serial I/O system is selected by setting SIO1CH flag and SB flag to "0" and "0", respectively.

The 2-wire serial I/O system uses $P0A_3$/SDA pin and $P0A_2$/SCL pin.

Fig. 21-9 shows the input/output block and communication method of the 2-wire serial I/O system.

Table 21-5 shows the function and operation of each pin and control register when the 2-wire serial I/O system is used.

As shown in Fig. 21-9 and Table 21-5, the 2-wire serial I/O system involves internal clock (master) operation and external clock (slave) operation, and each operation permits transmission (TX) and reception (RX).

The master and slave operation are selected by SIO1MS flag, while reception and transmission are selected by SIO1TX flag.

During the master operation, the internal shift clock is output from $P0A_2$/SCL pin. If transmission mode is selected, data is output from $P0A_3$/SDA pin at the falling edge of the shift clock. If reception mode is selected, the status of $P0A_3$/SDA pin is entered into the presettable shift register 1 at the rising edge of the shift clock.

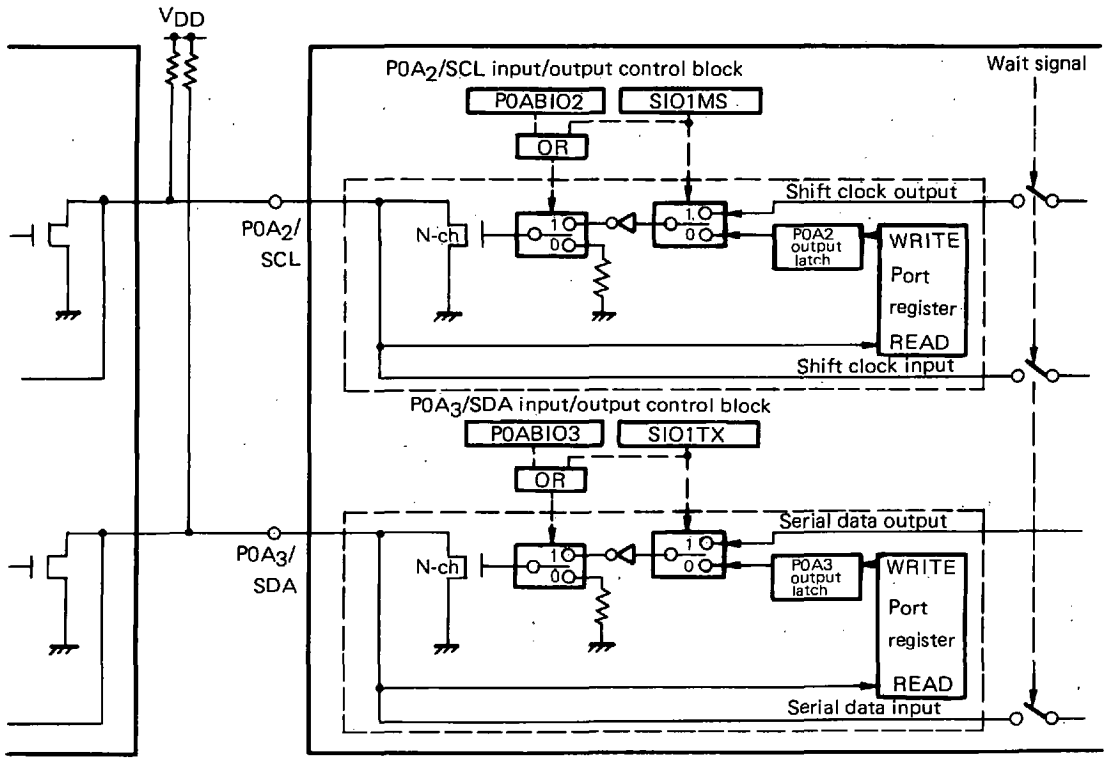
In the slave operation mode, the external clock is waited with $P0A_2$/SCL pin set in the floating state. If transmission is selected, data is output from $P0A_3$/SDA pin at the falling edge of the shift clock. If reception is selected, the status of $P0A_3$/SDA pin is entered into the presettable shift register 1 at the rising edge of the clock applied to $P0A_2$/SCL pin.

The $P0A_3$/SDA pin and $P0A_2$/SCL pin are use the N-ch open drain output, and if either the master or slave issues low level, the communication line is set at low level.

The $P0A_3$/SDA pin or $P0A_2$/SCL pin, when it reads the value output to the pin, reads in directly the "status of the pin at that moment".

Fig. 21-9 Input/output block and communication method of 2-wire serial I/O system

Input/output block



Communication method

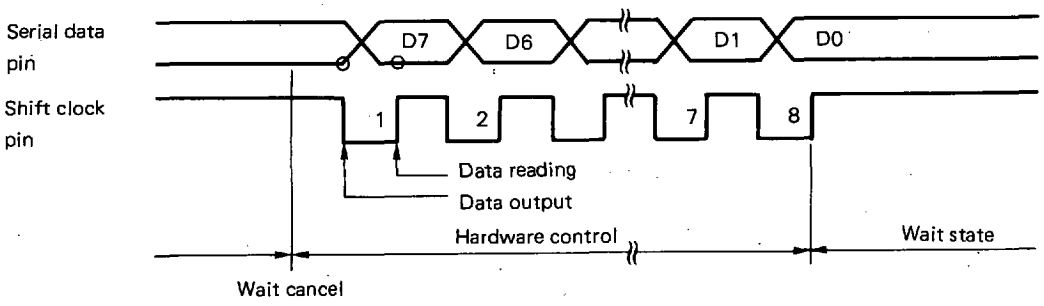


Table 21-5 Outline of operation of 2-wire serial I/O system

Item \ Operation mode		2-wire serial bus system			
		Slave operation		Master operation	
		Reception (RX) SIO1TX = 0	Transmission (TX) SIO1TX = 1	Reception (RX) SIO1TX = 0	Transmission (TX) SIO1TX = 1
Pin setting state	$P0A_3$/SDA	**P0ABIO3 = 0** Floating In wait for external data input **P0ABIO3 = 1** General purpose output port Content of output latch is output. Usually P0ABIO3 is set "0".	Irrespective of the value of P0ABIO3, content of PSR1 is output at the falling edge of external clock.	**P0ABIO3 = 0** Floating In wait for external data input **P0ABIO3 = 1** General purpose output port Content of output latch is output. Usually P0ABIO3 is set "0".	Irrespective of the value of P0ABIO3, content of PSR1 is output at the falling edge of external clock.
	$P0A_2$/SCL	**P0ABIO2 = 0** Floating In wait for external data input **P0ABIO2 = 1** General purpose output port Content of output latch is output. Usually P0ABIO2 is set "0".	**P0ABIO2 = 0** Gloating In wait for external data input **P0ABIO2 = 1** General purpose output port Content of output latch is output. Usually P0ABIO2 is set "0".	Irrespective of the value of P0ABIO2, internal shift clock is output.	Irrespective of the value of P0ABIO2, internal shift clock is output.
Clock counter operation		Increment at the rising edge of SCL pin			
Operation of presettable shift register 1 (PSR1)		**Output** No output **Input** Data of SDA pin is entered by shifting from LSB at each rising edge of SCL pin waveform.	**Output** Data is output to SDA pin by shifting from MSB at each falling edge of SCL pin waveform. **Input** Data of SDA pin is entered by shifting from LSB at each rising edge of SCL pin waveform.	**Output** No output **Input** Data of SDA pin is entered by shifting from LSB at each rising edge of SCL pin waveform.	**Output** Data is output to SDA pin by shifting from MSB at each falling edge of SCL pin waveform. **Input** Data of SDA pin is entered by shifting from LSB at each rising edge of SCL pin waveform.
Wait operation		Serial communication starts with writing of "1" to SIO1NWT. SIO1NWT is reset to "0" by the conditions set in SIO1WRQ1 and SIO1WRQ0.			
		SIO1NWT = 0 Low level is forcibly output from SCL pin. SDA pin is floating. **SIO1NWT = 1** In wait for external clock with SCL pin set floating. SDA pin is set floating, and data of SDA pin is entered to PSR1 at the rising edge of SCL pin waveform.	**SIO1NWT = 0** Low level is forcibly output from SCL pin. SDA pin is floating. **SIO1NWT = 1** In wait for external clock with SCL pin set floating. Content of PSR1 is output to SDA pin at the falling edge of SCL pin waveform.	**SIO1NWT = 0** Low level is forcibly output from SCL pin. SDA pin is floating. **SIO1NWT = 1** In wait for external clock with SCL pin set floating. SDA pin is set floating, and data of SDA pin is entered to PSR1 at the rising edge of SCL pin waveform.	**SIO1NWT = 0** Low level is forcibly output from SCL pin. SDA pin is floating. **SIO1NWT = 1** In wait for external clock with SCL pin set floating. Content of PSR1 is output to SDA pin at the falling edge of SCL pin waveform.

21.11.3 Usage of 3-wire Serial I/O System

The 3-wire serial I/O system is selected by setting SIO1CH flag and SB flag to "0" and "0", respectively.

The 3-wire serial I/O system uses $P0A_1/\overline{SCK_1}$, $P0A_0/SO_1$, and $P0B_3/SI_1$ pins.

Fig. 21-10 shows the input/output block and communication method of the 3-wire serial I/O system.

Table 21-6 shows the function and operation of each pin and control register when the 3-wire serial I/O system is used.

As shown in Fig. 21-10 and Table 21-6, the 3-wire serial I/O system involves internal clock (master) operation and external clock (slave) operation, and each operation permits transmission (TX) and reception (RX).

The master and slave operation are selected by SIO1MS flag, while reception and transmission are selected by SIO1TX flag.

During the master operation, the internal shift clock is output from $P0A_1/\overline{SCK_1}$ pin. If transmission mode is selected, data is output from $P0A_0/SO_1$ pin at the falling edge of the shift clock.

During the master operation, serial data input is not related to the transmission or reception mode. The status of $P0B_3/SI_1$ pin is entered to the presettable shift register 1 at the rising edge of the shift clock. In this case, however, the $P0B_3/SI_1$ pin must be set at the input pin.

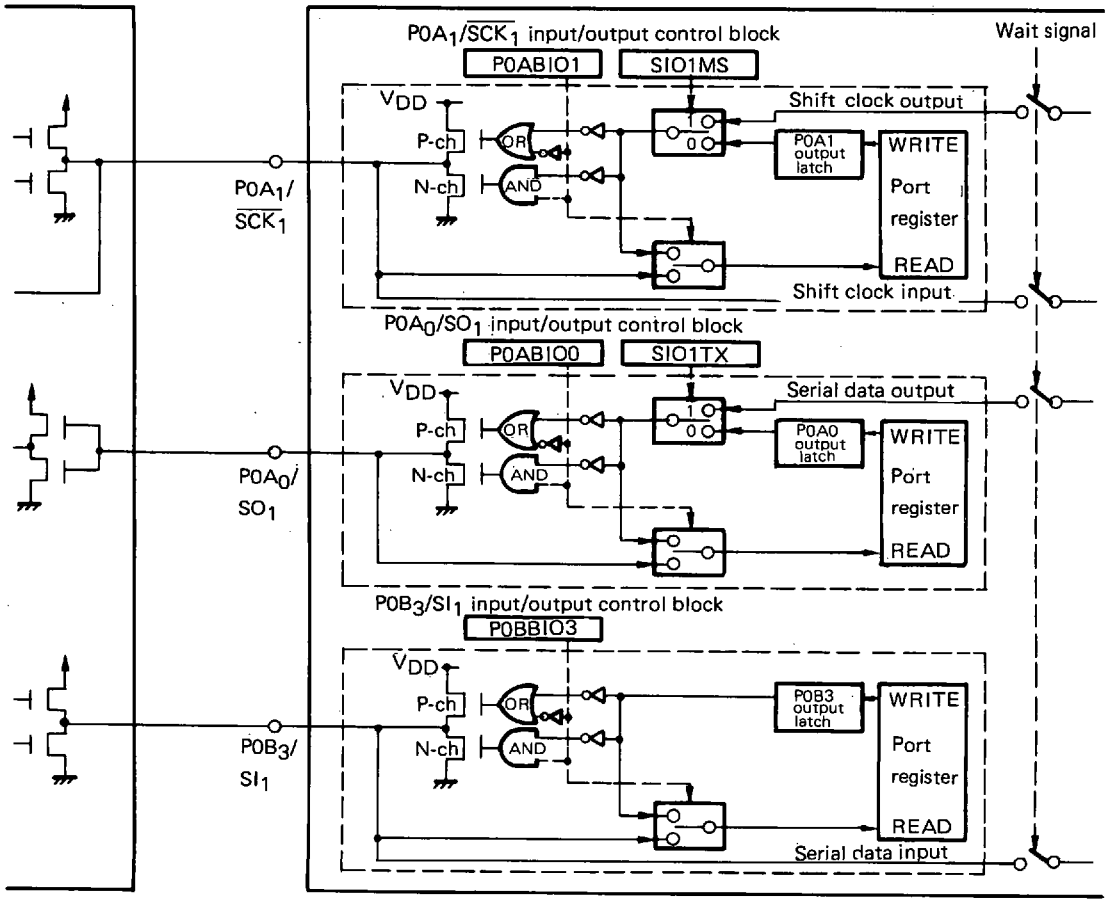
In the slave operation mode, the external clock is waited with $P0A_1/\overline{SCK_1}$ pin set in the floating state. If transmission is selected, data is output from $P0A_3$/SDA pin at the falling edge of the shift clock.

During slave operation, serial data input is not related to the transmission or reception mode. The status of $P0B_3/SI_1$ pin is entered to the presettable shift register 1 at the rising edge of the shift clock. In this case, however, the $P0B_3/SI_1$ pin must be set at the input pin.

The $P0A_1/\overline{SCK_1}$ pin or $P0A_0/SO_1$ pin, when it reads the value output to the pin, reads in directly the "status of the output latch at that moment".

Fig. 21-10 Input/output block and communication method of 3-wire serial I/O system

Input/output block



Communication method

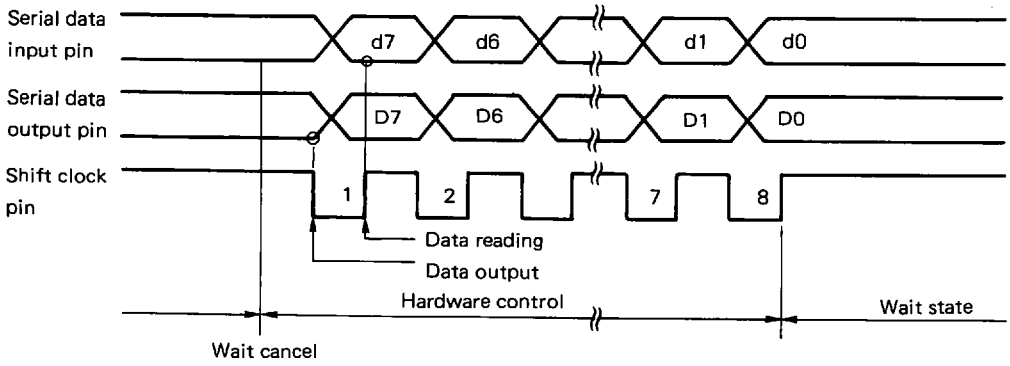


Table 21-6 Outline of operation of 3-wire serial I/O system (1/2)

Item \ Operation mode		2-wire serial bus system			
		Slave operation SIO1MS = 0		Master operation SIO1MS = 1	
		Reception (RX) SIO1TX = 0	Transmission (TX) SIO1TX = 1	Reception (RX) SIO1TX = 0	Transmission (TX) SIO1TX = 1
Setting of each pin	$P0A_1/\overline{SCK_1}$	**P0ABIO1 = 0** Floating In wait for input of external clock **P0ABIO1 = 1** General purpose output port Content of output latch is output. Usually P0ABIO1 is set "0".	**P0ABIO1 = 0** Floating In wait for input of external clock **P0ABIO1 = 1** General purpose output port Content of output latch is output. Usually P0ABIO1 is set "0".	Internal shift clock is output irrespective of setting of P0ABIO01.	Internal shift clock is output irrespective of setting of P0ABIO1.
	$P0A_0/SO_1$	**P0ABUO0 = 0** General purpose output port Floating **P0ABIO0 = 1** General purpose output port Content of output latch is output.	Content of PSR1 is output at the falling edge of external clock waveform irrespective of the setting of P0ABILO.	**P0ABIO0 = 0** General purpose input port Floating **P0ABIO0 = 1** General purpose output port Content of output latch is output.	Content of PSR 1 is output at the falling edge of internal shift clock waveform irrespective of the setting of P0ABIO0.
	$P0B_3/SI_1$	**P0BBIO3 = 0** Floating In wait for input of external data **P0BBIO3 = 1** General purpose output port Content of output latch is output. Usually P0BBIO3 is set "0".	**P0BBIO3 = 0** Floating In wait for input of external data **P0BBIO3 = 1** General purpose output port Content of output latch is output. Usually P0BBIO3 is set "0".	**P0BBIO3 = 0** Floating In wait for input of external data **P0BBIO3 = 1** General purpose output port Content of output latch is output. Usually P0BBIO3 is set "0".	**P0BBIO3 = 0** Floating In wait for input of external data **P0BBIO3 = 1** General purpose output port Content of output latch is output. Usually P0BBIO3 is set "0".
Clock counter operation		Increment at falling edge of $\overline{SCK_1}$ pin waveform			

Table 21-6 Outline of operation of 3-wire serial I/O system (2/2)

Item \ Operation mode	2-wire serial bus system			
	Slave operation SIO1MS = 0		Master operation SIO1MS = 1	
	Reception (RX) SIO1TX = 0	Transmission (TX) SIO1TX = 1	Reception (RX) SIO1TX = 0	Transmission (TX) SIO1TX = 1
Operation of presettable shift register 1 (PSR1)	**Output** No output **Input** Data of SI_1 is entered by shifting from LSB at each rising edge of SCK_1 pin waveform.	**Output** Data is output to SO_1 pin by shifting from MSB at each falling edge of SCK_1 pin waveform. **Input** Data of SI_1 is entered by shifting from LSB at each rising edge of SCK_1 pin waveform.	**Output** No output **Input** Data of SI_1 is entered by shifting from LSB at each rising edge of SCK_1 pin waveform.	**Output** Data is output to SO_1 pin by shifting from MSB at each falling edge of SCK_1 pin waveform. **Input** Data of SI_1 is entered by shifting from LSB at each rising edge of SCK_1 pin waveform.
Wait operation	Serial communication starts with writing of "1" to SIO1NWT. SIO1NWT is reset to "0" by the condition set in SIO1WRQ1 and SIO1WRQ0.			
	SIO1NWT = 0 SCK_1 pin is floating. SO_1 pin is general-purpose port. SI_1 pin is floating. **SIO1NWT = 1** SCK_1 pin is in wait for input of external clock. Data of SI_1 pin is entered to PSR1 at rising edge of SCK_1 pin. pin.	**SIO1NWT = 0** SCK_1 pin is floating. SO_1 pin is general-purpose port. SI_1 pin is floating. **SIO1NWT = 1** SCK_1 pin is in wait for input of external clock. Content of PSR1 is output to SO_1 pin at falling edge of SCK_1 pin. Data of SI_1 pin is entered to PSR1 at rising edeg of SCK_1 pin.	**SIO1NWT = 0** SCK_1 pin is at high level. SO_1 pin is general-purpose port. SI_1 pin is floating. **SIO1NWT = 1** SCK_1 pin outputs internal shift clock. SCK_1 pin is in wait for input of external clock. Data of SI_1 pin is entered to PSR1 at rising edge of SCK_1 pin.	**SIO1NWT = 0** SCK_1 pin is at high level. SO_1 pin holds status. SI_1 pin is floating. **SIO1NWT = 1** SCK_1 pin outputs internal shift clock. Content of PSR1 is output to SO_1 pin at falling edge of SCK_1 pin waveform. Data of SI_1 pin is entered to PSR1 at rising edge of SCK_1 pin waveform.

21.12 SERIAL INTERFACE 1 RESET STATUS

21.12.1 Power-ON Reset

$P0A_3$/SDA to $P0A_0/SO_1$ pin and $P0B_3/SI_1$ pins are all set at the general input port (floating output). The value of the presettable shift register 1 is undefined.

21.12.2 Clock Stop

$P0A_3$/SDA to $P0A_0/SO_1$ pin and $P0B_3/SI_1$ set at the general input port (floating output). The presettable shift register 1 retains the previous value.

21.12.3 CE Reset

$P0A_3$/SDA to $P0A_0/SO_1$ pin and $P0B_3/SI_1$ terminal set at the general input port (floating output). The presettable shift register retains the previous value.

21.12.4 Halt Status

The input/output pin holds the status which is set at that time.

In this case, clock output stops upon execution of "HALT" instruction if the internal clock is used (master operation).

When using the internal clock, it is therefore necessary to execute "HALT" instruction when communication has been terminated.

However, if clock is applied forcibly from an external source, the serial interface 1 functions even though the internal clock is established.

When the external clock is used (slave operation mode), the operation will be continued even though "HALT" instruction is executed.

Note that, when using the halt cancel by interruption of the serial interface 1, use of the internal clock is inhibited due to the aforementioned reason.

21.13 Configuration of Serial Interface 2 (SIO2)

Fig. 21-11 shows the block diagram of the serial interface 2.

As shown in Fig. 21-11, the shift clock control section of the serial interface 2 is composed of a clock input/output pin block, clock generation block, wait control block, and clock count block.

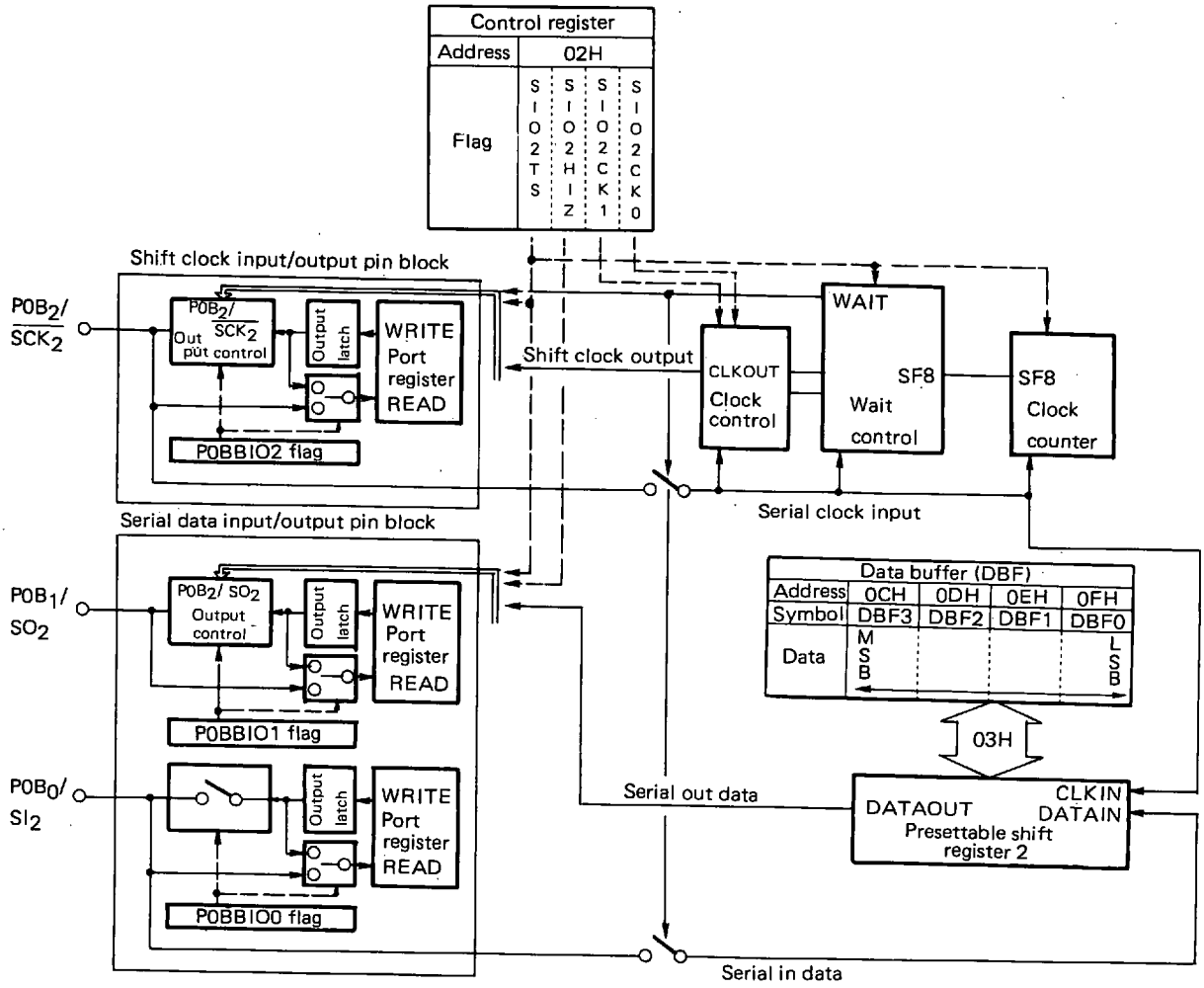
The serial data control section is composed of a serial data input/output pin block and presettable shift register 2.

These blocks are controlled by the flags of the control register.

Writing of data into and reading of data from the presettable shift register 2 are performed via the data buffer.

The functions of each block are outlined in Section **21.14.**

Fig. 21-11 Serial interface 2 block diagram



21.14 OUTLINE OF FUNCTION OF SERIAL INTERFACE 2

The serial interface 2 permits use of 3-wire serial I/O system as shown in Table 21-1.

The serial interface 2 uses $P0B_2/SCK_2$ pin, $P0B_1/SO_2$ pin and $P0B_0/SI_2$ pin.

The serial interface 2 permits selection of internal clock and external clock, and also permits selection of the reception and transmission operations.

Sections **21.14.1** thru **21.14.6** indicate the functions of blocks of the serial interface 2.

For details of each block, see Sections **21.15** thru **21.19.**

21.14.1 Shift Clock Input/Output Pin Block

This block is used for selecting the shift clock input/output pin. This selection of the shift clock input/output pin is performed by the serial I/O2 mode select register. See Section **21.15.**

21.14.2 Serial Data Input/Output Pin Block

This block is used to select the shift clock input/output pin. This selection of the shift clock input/output pin is performed by the serial I/O2 mode select register. See Section **21.15.**

21.14.3 Clock Generation Block

This block selects the clock frequency of the shift clock, and also controls the shift clock output timing. Selection of the clock frequency is performed by the serial I/O2 clock select register.

See Section **21.16.**

21.14.4 Clock Counter

The clock counter counts the number of the rising edges of the clocks output from the shift clock output pin, and issues signal at 8th clock (SF8 signal).

The SF8 signal is used to put the serial communication into a wait (pause).

See Section **21.17.**

21.14.5 Presettable Shift Register 2 (PSR2)

This is a shift register which sets the serial out data and stores the serial in data.

This register performs shift operation to input or output data by the clock input of the shift clock input pin.

Setting of the output data and reading of input data are performed via the data buffer.

See Section **21.18.**

21.14.6 Wait Control Block

This block controls the wait (pause) and wait cancel (communication operation) of serial communication.

Wait cancel of serial communication is performed by the serial I/O2 mode select register.

See Section **21.19.**

21.15 SHIFT CLOCK AND SERIAL DATA INPUT/OUTPUT CONTROL BLOCK

The shift clock and serial data input/output control block control the setting of pins of the serial interface 2 and sending and receiving operations.

These are controlled by the serial I/O2 mode select register.

The configuration and function of the serial I/O2 mode select register are explained in Section **21.15.1.**

The setting status of each pin by the serial I/O2 mode select register is explained in Section **21.15.2.**

21.15.1 Configuration and Function of Serial I/O2 Mode Select Register

The configuration and function of the serial I/O2 mode select register are explained below.

SIO1CK1 and SIO2CK0 flags select between internal clock and external clock, and also set the frequency of internal clock.

For the clock, see Section **21.16.**

SIO2TS flag sets the wait and wait cancel state of the serial interface 2.

For the wait operation, see Section **21.19.**

Name	Flag b_3	b_2	b_1	b_0	Address	Read/Write
Serial I/O2 mode register (SIO2 MODE)	SIO2TS	SIO2HIZ	SIO2CK1	SIO2CK0	02H	R/W

Setting of input/output clock frequency of serial interface 2		
0	0	External clock (slave)
0	1	75 kHz (Master)
1	0	150 kHz (Master)
1	1	450 kHz (Master)

	Setting of $P0B_1/SO_2$ pin as serial out pin
0	General purpose input/output port
1	Serial out

	Setting of serial input/output start of serial interface 2
0	Serial input/output does not operate.
1	Serial input/output starts.

Reset					
	Power ON	0	0	0	0
	Clock stop	0	0	0	0
	CE	0	0	0	0

21.15.2 Setting of Each Pin by Serial I/O2 Mode Select Register

Table 21-7 shows the setting of the pins by the serial I/O2 mode select register.

As shown in Table 21-7, setting of each pin also requires handling of the input/output setting flags.

When using $P0B_1/SO_2$ pin as serial out pin, $P0B_1/SO_2$ pin must be set as the output port by the Port 0B bit I/O select register (P0BBIO).

Similarly, $P0B_0/SI_2$ pin must be set as input port.

When using the external clock, $P0B_2/\overline{SCK}_2$ pin must be set as the general purpose input port. It must be set as output port when using the internal clock.

Table 21-7 Setting of pins by serial I/O2 mode select register (SIO2MODE)

SIO2MODE						Pin				
Communication system	b_2 SIO2HIZ	Setting of serial output	b_1 SIO2CK1	b_0 SIO2CK0	Clock direction	Pin symbol	Input/output setting flag P0BBIO2	P0BBIO1	P0BBIO0	Pin setting status
3-wire serial I/O			0	0	External clock	$P0B_2/\overline{SCK}_2$	0			During wait: General purpose input port During wait cancel: External clock input
			0	1	Internal clock		1			During wait: General purpose output port During wait cancel: General purpose output port
			1	0			0			During wait: General purpose input port During wait cancel: General purpose input port
			1	1			1			During wait: High level output During wait cancel: Internal clock output
	0	General purpose port				$P0B_1/SO_2$		0		General purpose input port
								1		General purpose output port
	1	Serial output						0		General purpose input port
								1		Serial output
						$P0B_0/SI_2$			0	Serial input
									1	General purpose output port

21.16 CLOCK GENERATION BLOCK

The clock generation block controls the clock generation and clock output timing when the internal clock is used (master operation mode).

The internal clock frequency fSC is set by SIO2CK1 and SIO2CK0 flags of the serial I/O2 mode select register.

The shift clock is output until the value of the clock counter, to be mentioned later, reaches "8".

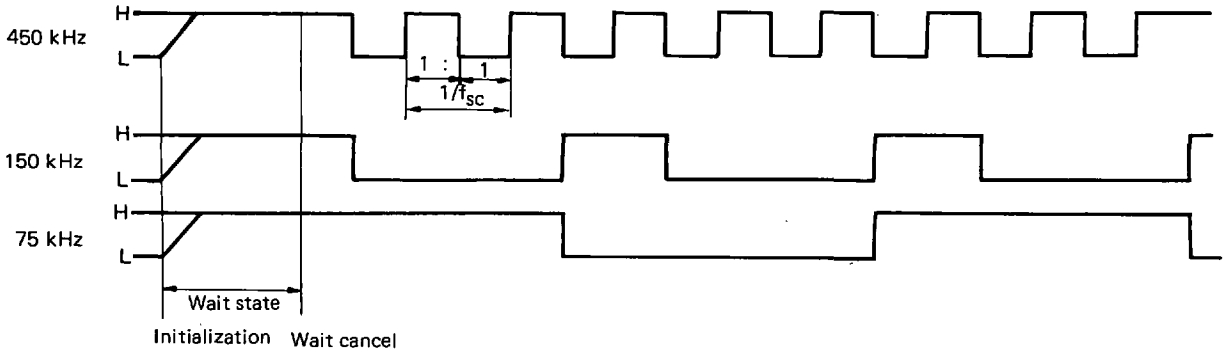
Section **21.16.1** shows the clock output waveform and generation timing.

21.16.1 Internal Shift Clock Generation Timing

(1) Wait cancel from initialization state

The initialization state indicates the state where the internal clock operation mode is selected and "high" level is output to $P0B_2/\overline{SCK}_2$ pin which is set as output pin.

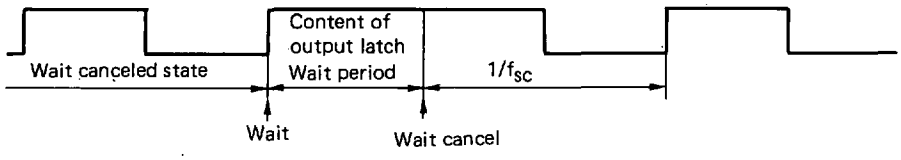
During the wait state, "High" level is output to the shift clock pin.



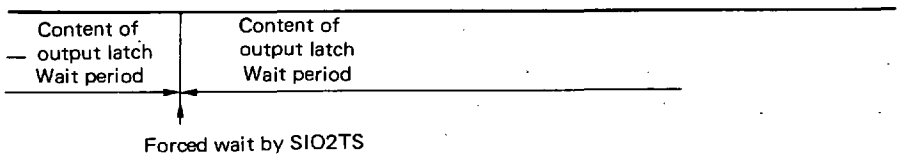
(2) When wait operation is performed

For the details of wait operation, see Section **21.19.**

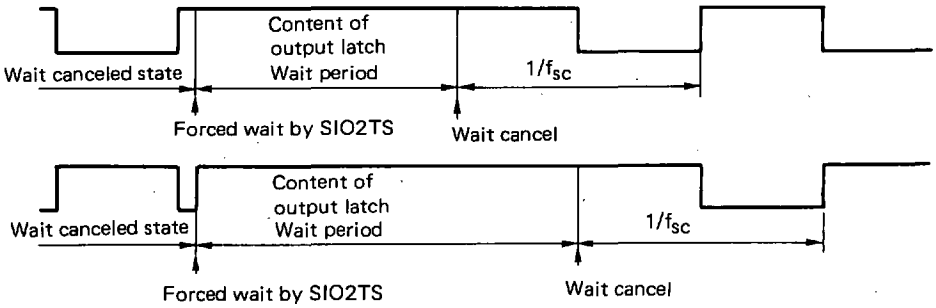
(a) Ordinary wait with clock counter reached "8"



(b) Forced wait during a wait



(c) Forced wait during wait canceled state



(d) Wait cancel during wait canceled state

No change occurs in the clock output waveform. The clock counter is not reset.

(e) When clock frequency change and wait cancel are effected at the same time

The setting of clock frequency and cancellation of wait are performed by the register of the same address, and cancellation of wait (setting of SIO2TS flag) and changing of the clock frequency can be performed by single instruction.

If wait cancellation and clock frequency change are performed at the same time, the same state is resulted as the wait canceled state from the initialization state mentioned in item (1) above.

21.17 CLOCK COUNTER

The clock counter counts the number of shift clocks output from or input to the shift clock pin ($P0B_2/\overline{SCK_2}$ pin).

The clock counter directly reads the status of shift clock pin, and it is unable to discriminate between internal clock and external clock.

The clock counter does not operate during the wait of serial communication.

The serial communication is turned to wait state at the rising edge of the shift clock waveform when the clock counter is "8".

The content of clock counter cannot be read directly by a program.

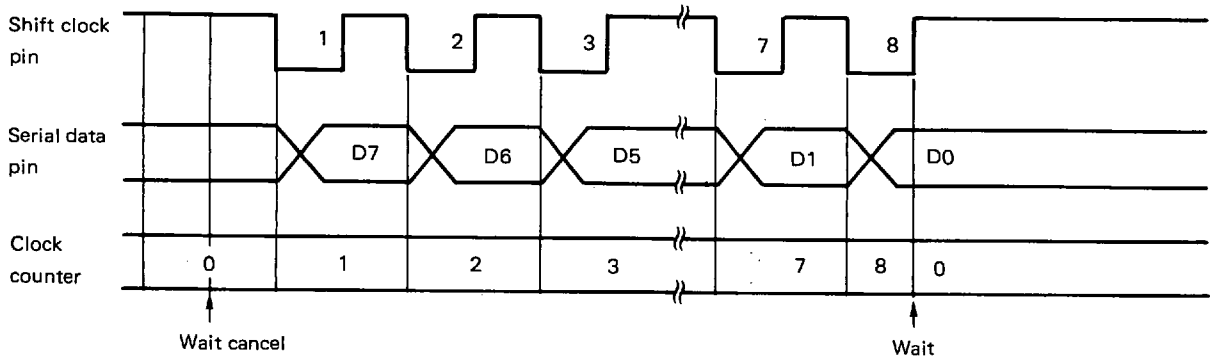
Sections **21.17.1** and **21.17.2** explain the clock counter operation and its reset condition.

21.17.1 Clock Counter Operation

The operation of clock counter is shown below.

The initial value of the clock counter is "0", and counter value increments (+1) upon each detection of the falling edge of the clock pin waveform. When counted up to "8", the counter is reset to "0" by the rising edge of next shift clock. The serial communication is put to wait state at the time the clock counter is reset to "0".

Fig. 21-12 Clock counter operation



21.17.2 Clock Counter Reset (0) Condition

The clock counter resetting conditions are listed below:

(1) Power ON
(2) Execution of clock stop instruction
(3) Writing of "0" into SIO2TS flag
(4) Rising of shift clock when wait is canceled and clock counter is "8".
(5) CE reset

21.18 PRESETTABLE SHIFT REGISTER (PSR2)

The presettable shift register 2 (PSR2) is an 8-bit shift register which is used to set the serial out data and read the serial in data.

Setting (writing) of data to and reading of data from the presettable shift register 2 are performed respectively by "PUT" instruction and "GET" instruction.

Section **21.18.1** shows the configuration of the presettable shift register 2 and its relation to the data buffer.

The data shift operation of the presettable shift register 2 is performed in synchronization with the clock applied to the shift clock pin ($P0B_2/\overline{SCK_2}$ pin).

The content of the most significant bit of the presettable shift register is output to the serial data pin in synchronization with the falling edge of the shift clock.

The data of the serial data pin is read into the least significant bit of the presettable shift register 2 in synchronization with the rising edge of the clock waveform.

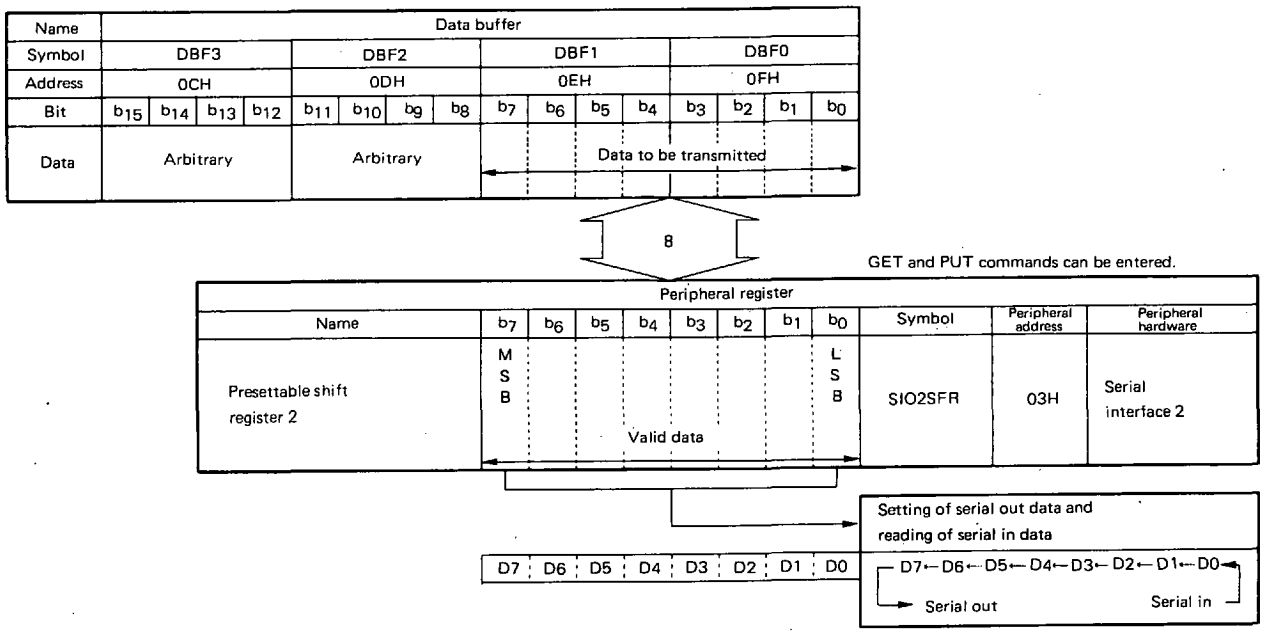
Section **21.18.2** shows the operation and precautions concerning this shift register.

Section **21.18.3** shows precautions concerning data writing into and data reading from the presettable shift register 2.

During the wait state, the presettable shift register 2 does not perform data shift operation.

21.18.1 Configuration of Presettable Shift Register 2 and Its Relation with Data Buffer

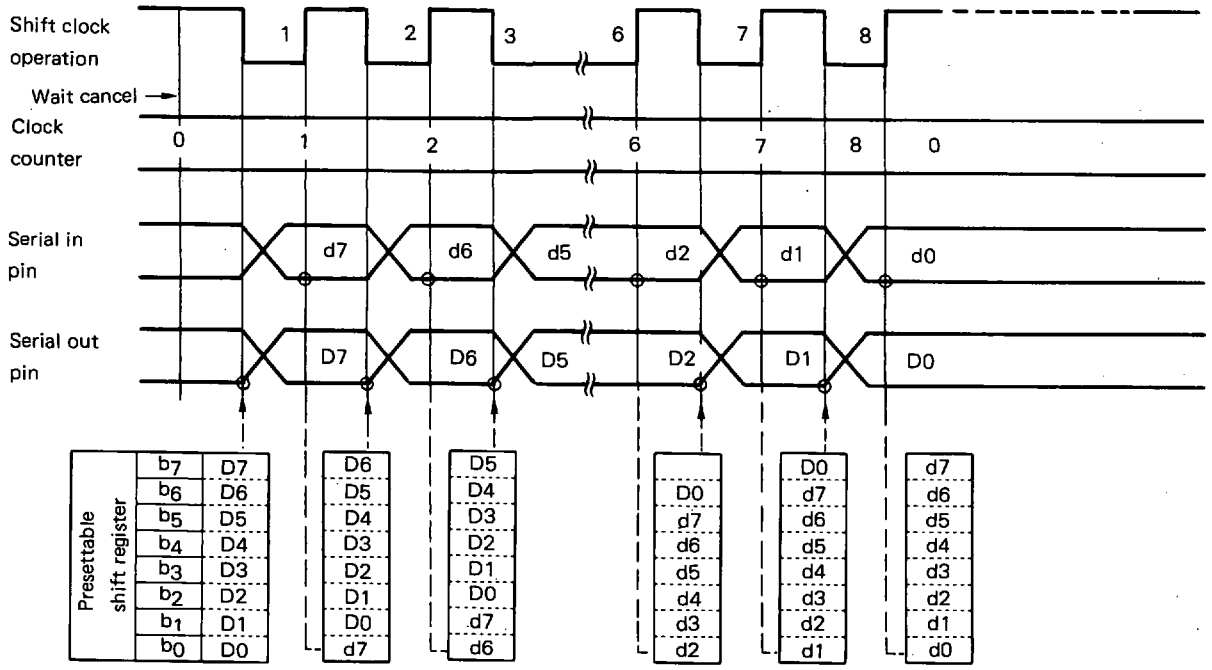
The configuration of the presettable shift register 2 and its relation to the data buffer are shown below.



21.18.2 Operation of Presettable Shift Register 2

The operation is shown below.

Fig. 21-13 Data shift operation of presettable shift register 2



Data shift operation of presettable shift register 2

Serial I/O system	
Serial input operation	Serial output operation
The status of $P0B_0/SI_2$ is entered by shifting from LSB at the rising edge of shift clock pin waveform. If the $P0B_0/SI_2$ pin is set as input port, the content of output latch is entered.	The data is output to $P0B_1/SO_2$ pin by shifting from MSB at the falling edge of shift clock pin waveform. If the $P0B_1/SO_2$ pin is set as input port, or if SIO2HIZ flag is "0", then no serial output is provided.

21.18.3 Precautions in Data Setting and Data Reading

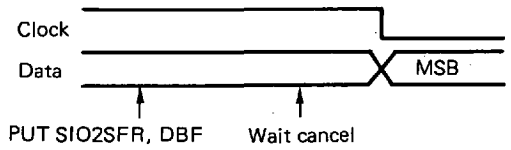
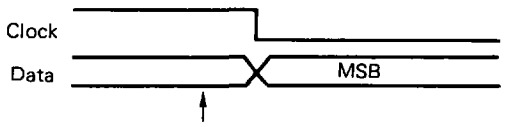
Data writing into the presettable shift register 2 is performed by "PUT SIO2SFR, DBF" instruction.

Reading of data is performed by "GET DBF, SIO2SFR" instruction.

Data setting and data reading must be performed while the wait status exists. During the wait cancel, data setting and data carrying may fail depending on the status of the shift clock pin.

Table 21-8 shows the timing of data setting and data reading and precautions.

Table 21-8 Data read (GET) and data read (PUT) operations of presettable shift register 2 and precautions

<table>
<tr><th colspan="2">Status when executing PUT/GET</th><th>Status of shift clock pin</th><th>Presettable shift register 2 (PSR2)</th></tr>
<tr><td rowspan="2">Wait state</td><td>Reading (PUT)</td><td>With external clock: Floating</td><td>Normal reading</td></tr>
<tr><td>Writing (PUT)</td><td>With internal clock: Value of output latch Usually "high" level is used.</td><td>Normal writing
The content of MSB is output as data when the wait is canceled next and shift clock pin waveform falls.
 
</td></tr>
<tr><td rowspan="3">Wait cancel state</td><td rowspan="2">Reading (GET)</td><td>"Low" level</td><td>Normal reading</td></tr>
<tr><td>"High"level</td><td>Normal reading fails, and content of PSR2 is destroyed.</td></tr>
<tr><td>Writing (PUT)</td><td>"High" level</td><td>Normal writing
The content of MSB is output as data upon execution of PUT instruction. Clock counter is not reset.
 
</td></tr>
<tr><td>"Low" level</td><td>Normal writing fails, and content of PSR2 is destroyed.</td></tr>
</table>

21.19 WAIT BLOCK

The wait block controls pause (wait) and cancel of communication of the serial interface 2. This control is performed by the SIO2TS flag.

Section **21.19.1** shows the wait operation and precautions.

21.19.1 Wait Operation and Precautions

The wait state means a state where the clock generation block, presettable shift register 2, etc. stop their operation, and the serial communication is suspended.

When the wait state if canceled, serial communication operation is started.

Wait state is canceled by writing "1" into SIO2TS flag.

When "1" is written into the SIO2TS flag, the internal clock is output to the shift clock output pin (master operation mode), and the presettable shift register 2 and clock counter start operation.

When the clock counter is "8" and shift clock rises, the wait cancel state turns into the wait state. In this case, the SIO2TS flag is reset (0) automatically.

The operation status of serial communication can be known by detecting the content of SIO2TS flag while the wait is canceled.

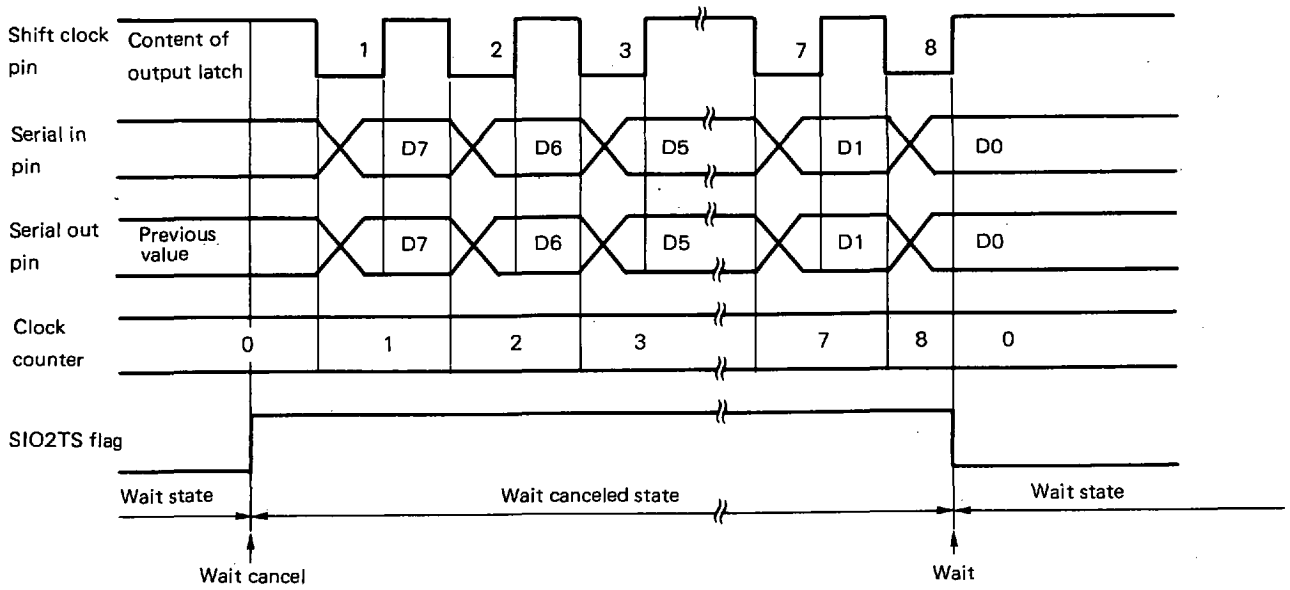
After starting the serial communication by writing "1" to SIO2TS flag, the data can be read or set by detecting the SIO2TS flag turning to "0".

This means that correct data setting and reading may fail if data setting (PUT instruction) or data reading (GET instruction) is executed to the presettable shift register 2 during the wait canceled state. See Section **21.18.3 "Precautions in data setting and data reading".**

Writing of "0" to SIO2TS flag during the wait cancel state causes the wait state to be established. This is called as "forced wait".

An example of wait operation is shown below.

Example of wait operation



When wait is canceled, the serial data is output at the falling edge of the next clock, and the flag turns into the wait canceled state.

When eight shift clock pulses are entered, the value of the output latch (usually "high" level) is output from the shift clock pin, and this causes the operation of the clock counter and presettable shift register 2 to be stopped.

Note that correct data will not be set if data writing to and data reading from the presettable shift register 2 are attempted while the wait is in the canceled state and the shift clock pin is at "high" level.

If data is written into the presettable shift register 2 while the wait is in the canceled state and the shift clock pin is at "low" level, the content of MSB will be output to the serial data output pin at the time when "PUT" instruction is executed.

If forced wait is effected during the wait canceled state, a wait state is resumed upon writing of "0" into SIO2TS flag.

21.20 USAGE OF SERIAL INTERFACE 2

Fig. 21-14 shows the input/output blocks and communication method of the serial interface 2.

Table 21-9 shows the operation of each mode of the serial interface 2.

As shown in Fig. 21-14 and Table 21-9, there are internal clock operation mode and external clock operation mode, and each permits transmission and reception.

Master and slave operation modes are selected by SIO2CK1 and SIO2CK0 flags.

Reception and transmission are set according to the pins used.

In the master operation mode, $P0B_2/\overline{SCK_2}$ pin outputs internal clock. In this case, however, the $P0B_2/\overline{SCK_2}$ pin must be set as output port.

In the slave operation mode, $P0B_2/\overline{SCK_2}$ pin is set in the floating state for receiving external clock. In this case, however, the $P0B_2/\overline{SCK_2}$ pin must be set as input port.

Serial data is output from $P0B_1/SO_2$ pin at the falling edge of the shift clock irrespective of the internal clock or external clock. In this case, however, $P0B_1/SO_2$ pin must be set as output port, and SIO2HIZ flag be set.

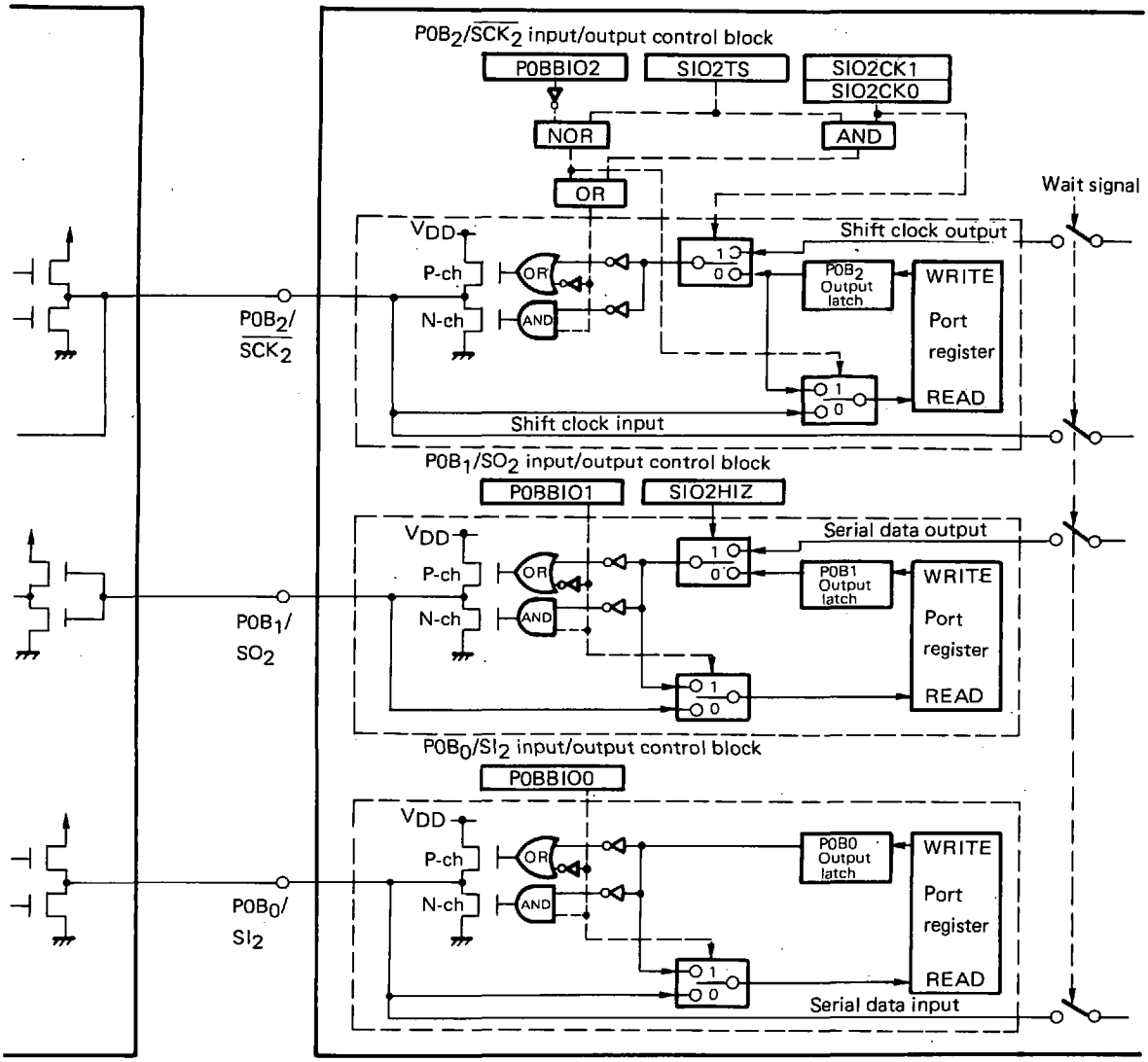
Serial data is input to the presettable shift register 2 as the status of $P0B_0/SI_2$ pin at the rising edge of the shift clock irrespective of the internal clock or external clock.

$P0B_2/\overline{SCK_2}$ pin reads "the current status of output latch" during a wait, or reads "the status of the current pin" during a wait cancel.

$P0B_1/SO_2$ pin reads "the current status of output latch".

Fig. 21-14 Input/output block of serial interface 2 and communication method

Input/output block



Communication method

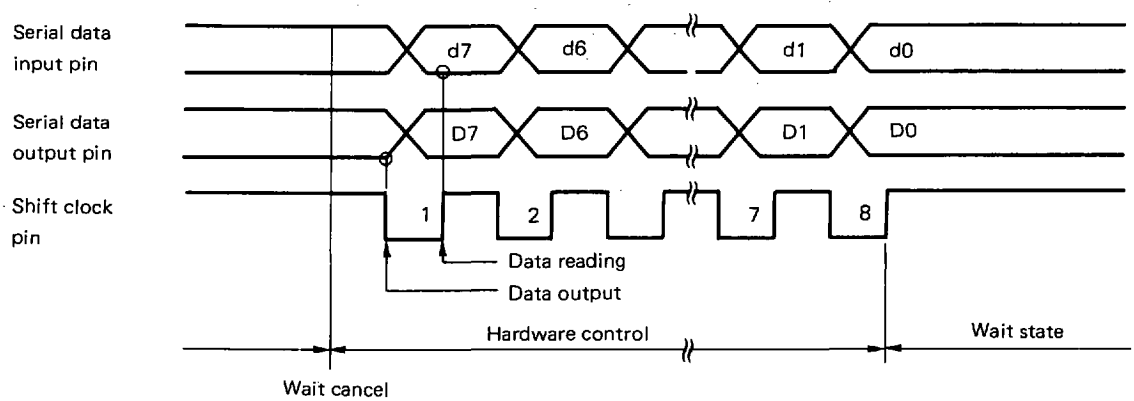


Table 21-9 Operation of each mode of serial interface 2

Operation mode / Item		3-wire serial I/O system			
		Slave operation SIO2CK1 = SIO2CK0 = 0		Master operation SIO2CK1 = SIO2CK0 = 0	
		During wait state	During wait canceled state	During wait state	During wait canceled state
Pin set state	$P0B_2/\overline{SCK_2}$	**P0BBIO2 = 0** Floating General purpose input port **P0BBIO2 = 1** General purpose output port Content of output latch Normally set 0 for P0BBIO2.	**P0BBIO2 = 0** Floating External clock input **P0BBIO2 = 1** General purpose output port	**P0BBIO2 = 0** Floating General purpose input port **P0BBIO2 = 1** General purpose output port Content of output latch Normally set 1 for P0BBIO2.	**P0BBIO2 = 0** Floating General purpose input port **P0BBIO2 = 1** Internal clock output
	$P0B_1/SO_2$	**SIO2HIZ = 0** **P0BBIO1 = 0** General purpose input port Floating **P0BBIO1 = 1** General purpose output port Content of output latch	**SIO2HIZ = 1** **P0BBIO1 = 0** General purpose input port Floating **P0BBIO1 = 1** Serial data output	**SIO2HIZ = 0** **P0BBIO1 = 0** General purpose input port Floating **P0BBIO1 = 1** General purpose output port Content of output latch	**SIO2HIZ = 1** **P0BBIO1 = 0** General purpose input port Floating **P0BBIO1 = 1** Serial data output
	$P0B_0/SI_2$	**P0BBIO0 = 0** Floating Waiting for input of external data **P0BBIO0 = 1** General purpose output port Content of output latch Normally set 0 for P0BBIO0.			
Clock counter operation		Increment at falling edge of $\overline{SCK_2}$			
Presettable shift register 2 (PSR2) operation		**Output** SIO2HIZ = 1 Output from SO_2 pin by shifting from MSB at each falling edge of $\overline{SCK_2}$ pin. SIO2HIZ = 0 No output **Input** Data of SI_2 pin is shifted from LSB and entered at each rising edge of $\overline{SCK_2}$ pin irrespective of P0BBIO0. However, if P0BBIO0 = 1, the content of output latch is output at SI_2 pin			
Wait operation		Serial communication starts with writing of "1" into SIO2TS. SIO2TS is reset to "0" at each rising edge of the the shift clock when the clock counter is "8". Operation of each pin are described above.			

21.21 RESET STATUS OF SERIAL INTERFACE 2

21.21.1 Power ON

$P0B_2/\overline{SCK_2}$ thru $P0B_0/SI_2$ pins are all set as general purpose input port.

The value of the presettable shift register 2 is undefined.

21.21.2 Clock Stop

$P0B_2/\overline{SCK_2}$ thru $P0B_0/SI_2$ pins are all set as general purpose input port.

The presettable shift register holds the previous value.

21.21.3 CE Reset

$P0B_2/\overline{SCK_2}$ thru $P0B_0/SI_2$ pins are all set as general purpose input port.

The presettable shift register 2 holds the previous value.

21.21.4 Halt State

The input/output pin holds the status which is set currently.

If the internal clock is in use (master operation mode), the clock output stops at the time when "HALT" instruction is executed.

Accordingly, "HALT" instruction must be executed upon termination of communication if it is to be used when the internal clock is being used. If clock is input forcibly from outside, the function of the serial interface 2 becomes operative even if the internal clock is being used already.

When the external clock is used (slave operation mode), the operation of the serial interface 2 continues even if "HALT" instruction is executed.

22. FREQUENCY COUNTER (FC)

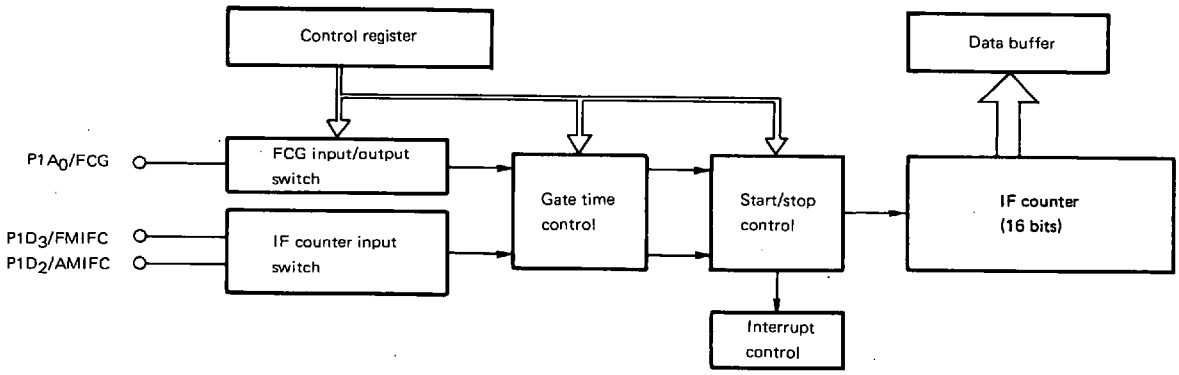
The frequency counter (FC) is used to count the intermediate frequency (IF) of the tuner and detection of pulse width of external signal.

22.1 CONFIGURATION OF FREQUENCY COUNTER

Fig. 22-1 shows the block diagram of the frequency counter.

As shown in Fig. 22-1, the frequency counter consists of an FCG input/output switch block, IF counter input switch block, gate time control block, start/stop control block and count block.

Fig. 22-1 Frequency counter block diagram



22.2 OUTLINE OF FUNCTION OF FREQUENCY COUNTER

The frequency counter possesses the IF counting function for counting the frequency of external input signal and the frequency counter function for external gate signal (FCG).

The FCG function detects the pulse width of external input signal.

The IF counter function counts the frequency entered to $P1D_3$/FMIFC pin or $P1D_2$/AMIFC pin using a 16-bit counter for a fixed time (1 ms, 4 ms, 8 ms).

The FCG function counts the frequency of the internal clock (1 kHz, 100 kHz, 900 kHz) during the period from the rising edge of the signal applied to $P1A_0$/FCG pin to the next rising edge using a 16-bit counter.

For the IF counter and FCG functions, see Sections **22.5** and **22.6.**

The following Sections **22.2.1** to **22.2.4** show the outline of function of each block.

The frequency counter uses the common hardware as the clock generator port mentioned in Section **20**, hence simultaneous use of the frequency counter and clock generator port is inhibited. For details, see Section **22.8 "Usage Precautions of Frequency Counter".**

22.2.1 Counter Switch Block and FCG Switch Block

The IF counter switch block is used to set whether $P1D_3$/FMIFC pin and $P1D_3$/AMIFC pin are to be used as general purpose input port or IF counter function.

The FCG input/output switch block is used to set whether $P1A_0$/FCG pin is used as general purpose input/output port or FCG function.

Switching of the general purpose port, IF counter function and FCG function is performed by the IF counter mode select register (RF address 12H).

For further details, see Section **22.3.**

22.2.2 Gate Time Control Block

The gate time control block controls the frequency counting time by the IF counter mode select register.

The operation of this control block when the IF count function and FCG function are used is outlined in the following Items (1) and (2).

For details, see Section **22.3.**

(1) When IF counter function is used

The IF counter mode select register of this gate time control block sets the internal gate time (1 ms, 4 ms, 8 ms) for counting the frequency applied to FMIFC or AMIFC pin.

(2) When FCG function is used

The IF counter mode select register sets the internal frequency (1 kHz, 100 kHz, 900 kHz) for counting the external gate time (from a leading edge of to the next leading edge of the signal applied to FCG pin).

22.2.3 Start/Stop Control Block

The start/stop control block controls start and stop of the frequency counting operation.

This control is performed by the IF counter control register (RF address 23H) and IF counter gate judge register (RF address 04).

This control block issues an interrupt request upon closure of the internal gate.

For details, see Section **22.4.**

22.2.4 IF Counter

The IF counter counts the input frequency using 16-bit binary counter when the IF counter function or FCG function is used.

When reading out the value of the IF counter data register (peripheral address 43H), this IF counter reads it via the data buffer.

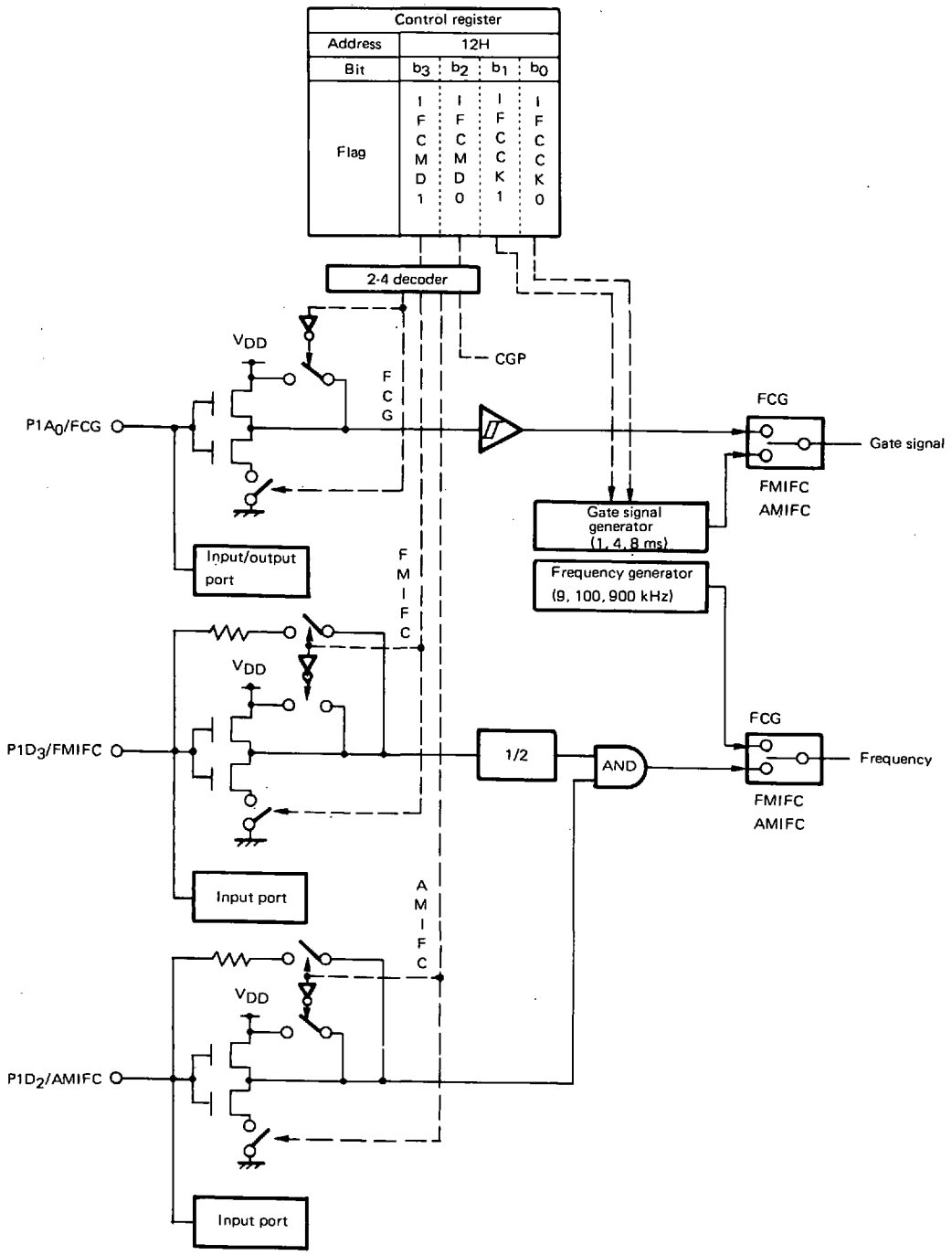
For details, see Section **22.4.**

22.3 INPUT/OUTPUT SWITCH BLOCK AND GATE TIME CONTROL BLOCK

22.3.1 Configuration of Input/Output Switch Block and Gate Time Control Block

Fig. 22-2 shows the configuration of the IF counter input switch, external gate counter input/output switch and gate time control blocks.

Fig. 22-2 Configuration of output switching blocks



22.3.2 Function of Input/Output Switching Block

The input/output switching block selects whether each pin is to be used as general purpose input/output port or as a frequency counter.

This switching between general purpose input/output port and frequency counter is performed by IFCMD1 and IFCMD0 flags of the IF counter mode select register.

Section **22.3.4** shows the configuration and function of the IF counter mode select register.

When using the external gate counter, the $P1A_0$/FCG pin must be set as input port.

22.3.3 Function of Gate Time Control Block

The gate time setting block sets the gate time (counting time) when IF count function if selected, or the count frequency when FCG function is selected.

Setting of the gate time and count frequency is performed by IFCCK1 and IFCCK0 flags of the IF counter mode select register.

Section **22.3.4** shows the configuration and function of the IF counter mode select register.

22.3.4 Configuration and Function of IF Counter Mode Select Register

The IF counter mode select register determines whether IF counter function or FCG function is to be used.

The configuration and function of this register are explained below.

The frequency counter serves also as the clock generator port, and this register also performs selection of the clock generator port.

Name	Flag b_3	b_2	b_1	b_0	Address	Read/Write
If counter mode select register (IFCMODE)	IFCMD1	IFCMD0	IFCCK1	IFCCK0	12H	R/W

Setting of IF counter gate time and the reference frequency of FCG			
		IF counter gate time	Reference frequency of FCG
0	0	1 ms	1 kHz
0	1	4 ms	100 kHz
1	0	8 ms	900 kHz
1	1	Open	0 kHz

		Selecting function of IF counter, FCG and clock generator port (CGP)
0	0	Clock generator port (CGP)
0	1	IF counter (FMIFC)
1	0	IF counter (AMIFC)
1	1	External gate counter (FCG)

Reset	Power ON	0	0	0	0
	Clock stop	0	0	0	0
	CE	Held			

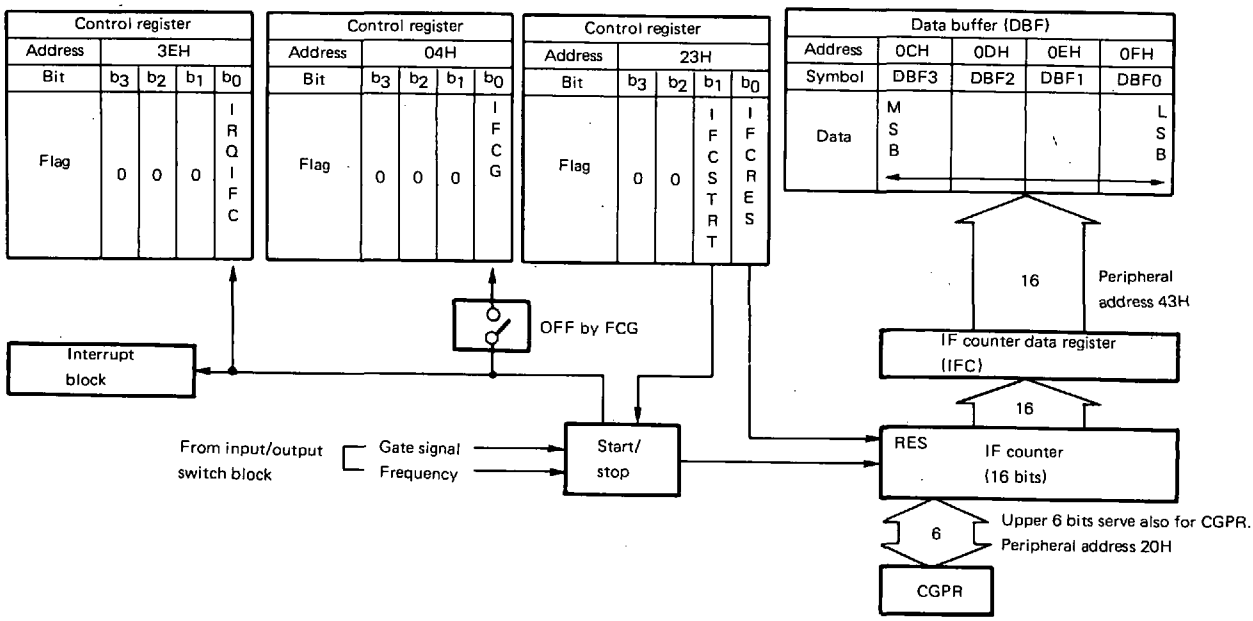
Note that the IF counter function, FCG function and CGP function cannot be used at the same time.

22.4 START/STOP CONTROL BLOCK AND IF COUNTER

22.4.1 Configuration of Start/Stop Control Block and Counter

Fig. 22-3 shows the configuration of the start/stop control block and counter.

Fig. 22-3 Configuration of start/stop control block and counter



22.4.2 Function of Start/Stop Control Block

The start/stop control block performs setting of count start and detection of count end of the frequency counter.

Counting is started by IFCSTRT flag of the IF counter control register.

End of counting is detected by IFCG flag of the IF counter gate judge register or by IRQIFC flag of interrupt request 2 register (RF address 3EH).

Sections **22.4.3** and **22.4.4** explain the operation of this control block in the IF counter function and FCG function mode.

Sections **22.4.7** and **22.4.8** explain the configuration and function of the IF counter gate judge register and IF counter control register.

22.4.3 Gate Operation in IF Counter Function Mode

(1) Gating time selected as 1, 4, or 8 ms

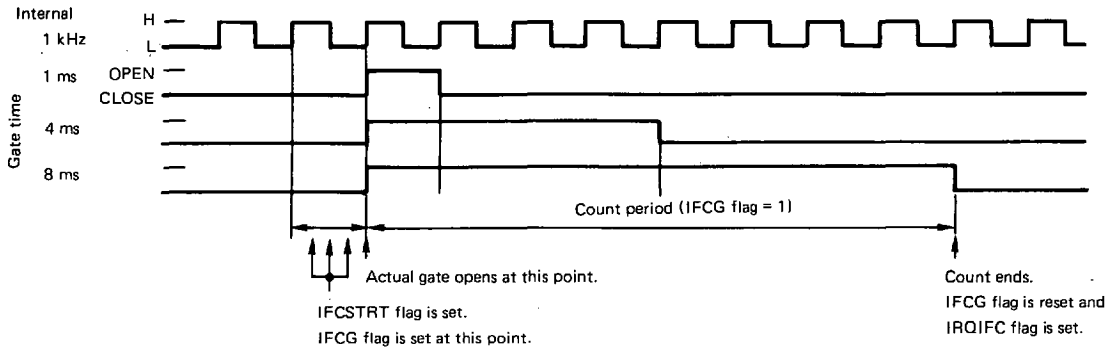
As shown below, the gate is opened for 1 ms, 4 ms, and 8 ms period beginning with the leading edge of the internal 1 kHz signal after setting (1) IFCSTRT flag.

During the open period of this gate, the frequency entered from the pin is counted by a 16-bit counter.

When the gate closes, IFCG flag is reset, while IRQIFC flag is set.

IFCG flag is automatically set (1) at the time when IFCSTRT flag is set.

IRQIFC flag is reset when an interrupt is received and when "0" is written.

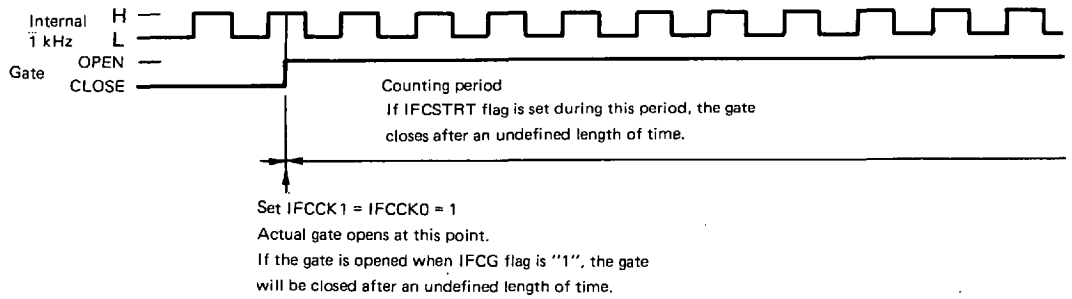


(2) Gate time selected as "open"

If "open" is selected by IFCCK1 or IFCCK0 flag, as shown below, the gate will open at the time such a selection is effected.

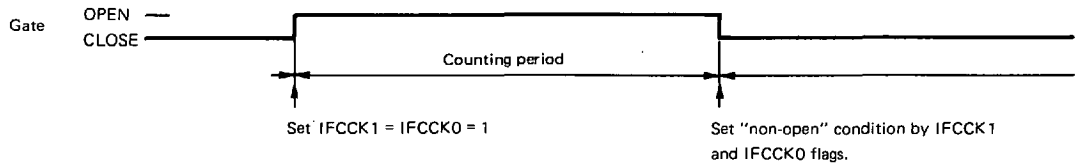
If count start is set by IFCSTRT flag during the open period of this gate, the gate will be closed at an undefined length of time.

When selecting the "open" mode for the gate time, do not set (1) IFCSTRT flag. Resetting of the counter by IFCRES flag is allowable.



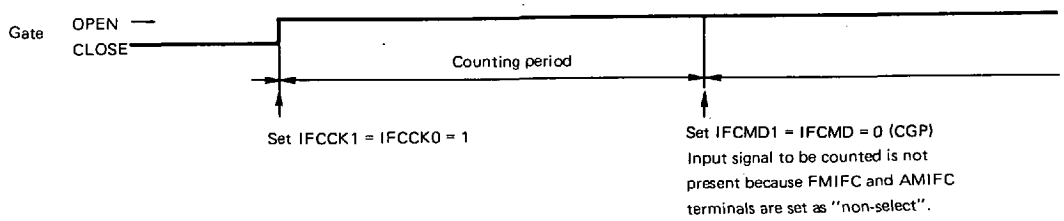
When "open" is selected for gating time, the open/close of the gate can be controlled in one of the following two methods (a) and (b).

(a) Setting again the "non-open" condition for the gate time by IFCCK1 and IFCCK0 flags.



(b) Setting the pin in use as "non-select" by IFCMD1 and IFCMD0 flags

This method allows the gate to be kept "open", and stops counting by inhibiting input from the pin.



22.4.4 Gating Operation in FCG Mode

The gate is opened for the period from the leading edge of the signal entered to the pin after setting (1) IFCSTRT flag to the next leading edge.

While this gate is open, the internal frequency (1 kHz, 100 kHz, 900 kHz) is counted by a 16-bit counter.

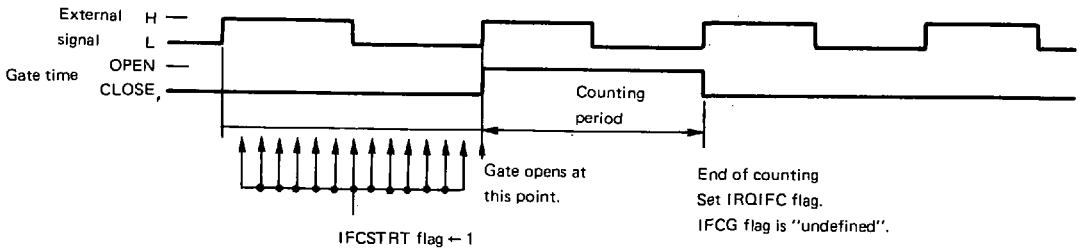
When the gate closes, IRQIFC flag is set.

The IRQIFC flag is reset when an interrupt is accepted and when "0" is written.

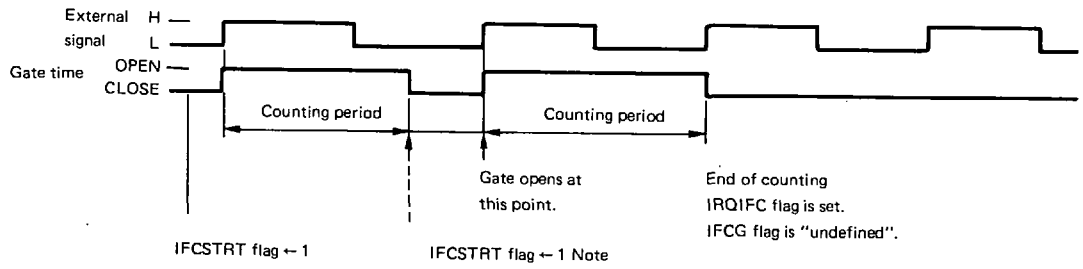
This means that setting of IFCSTRT flag alone does not cause automatic resetting of IRQIFC flag. It is therefore necessary to reset the flag by the program when starting the counting operation.

Note that IFCG flag is automatically set (1) when IFCSTRT flag is set, but it will not be reset when the gate is closed.

In the FCG function mode, it is unable to detect open/close state of the gate by IFCG flag.



Resetting and starting while the gate is open



22.4.5 Function and Operation of 16-Bit Counter

The 16-bit counter counts up the frequency entered within the gating time.

The 16-bit counter is reset by writing "1" into IFCRES flag of the IF counter control register.

The 16-bit counter turns to 0000H after counting up to FFFFH, and continues counting.

The upper 6 bits of the 16-bit counter serves also the clock generator port function, and it is unable to use the frequency counter and clock generator port at the same time.

The following Items (1) and (2) show the operation of this counter in FCG function mode.

The value of the IF counter data register is read via the data buffer.

Section **22.4.6** shows the configuration and function of the IF counter data register.

(1) IF counter function mode

The frequency applied to $P1D_3$/FMIFC pin or $P1D_2$/AMIFC pin is counted while the gate is open.

The frequency applied to $P1D_3$/FMIFC pin is divided to 1/2 before counting.

The relationship between the count value "x (HEX)" and input frequency (f_{FMIFC}, f_{AMIFC}) is shown below.

FMIFC

$$f_{FMIFC} = \frac{x(DEC)}{t_{GATE}} \times 2 \text{ (kHz)}$$

t_{GATE} : Gate time (1 ms, 4 ms, 8 ms)

AMIFC

$$f_{AMIFC} = \frac{x(DEC)}{t_{GATE}} \text{ (kHz)}$$

t_{GATE} : Gate time (1 ms, 4 ms, 8 ms)

(2) FCG function mode

The internal frequency is counted while the gate is opened by the signal applied to $P1A_0$/FCG pin. The relationship between the counted value "x (HEX)" and the input signal gate width (t_{GATE}).

$$t_{GATE} = \frac{x(DEC)}{f_r} \text{ (ms)}$$

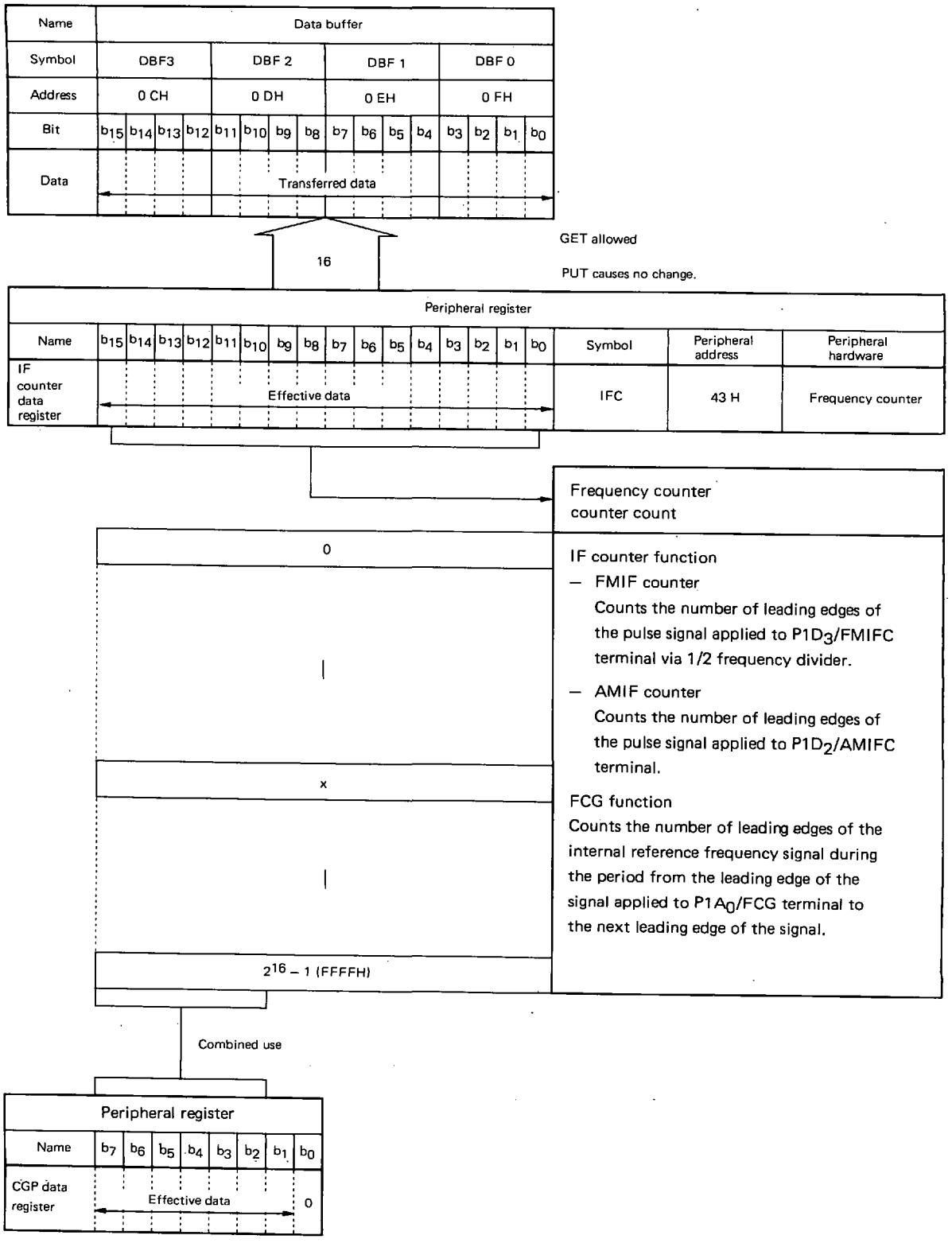
f_r : Internal frequency (1 kHz, 100 kHz, 900 kHz)

22.4.6 IF Counter Data Register

The function of the IF counter data register is explained below.

The IF counter data register reads the counted value of the frequency counter.

When the IF counter data register counts up to FFFFH, it turns to 0000H upon application of the next input, and counting is continued again.



Note that erroneous counting will result if data is set into the CGP data register while the frequency counter is operating because the upper 6 bits of the IF counter data register serves also as CGP data register.

That is, simultaneous use for the frequency counter function and CGP function is not allowed.

See the Section **22.8, "Precautions in Use of Frequency Counter".**

22.4.7 IF Counter Control Register (IFCCONT)

The IF counter control register controls start of counting operation by the frequency counter (IF counter and external gate counter), and also controls resetting of the 16-bit counter.

The configuration and function of the register are shown below.

Name	Flag b_3	b_2	b_1	b_0	Address	Read/Write
IF counter control register (IFCCONT)	0	0	IFCSTRT	IFCRES	23H	W

IFCRES: Controls counting of IF counter and external gate counter.

0	No change is caused.
1	Counter is reset.

IFCSTRT: Sets starting of counting operation of IF counter and external gate counter.

0	No change occurs.
1	Counting operation is started.

b_2: Fixed to "0".

		b_3	b_2	b_1	b_0
Reset	Power ON	0	0	0	0
	Clock stop			0	0
	CE			Held	

The IF counter control register is controlled by writing the content of the window register using a "POKE" instruction.

If the content is read into the window register using a "PEEK" instruction, "undefined value" will be entered.

In this case, if either one flag only is set using a SETn instruction which is a macro instruction built in the assembler (AS17K), then an error will be caused because the other flag is "undefined". For details, see the Section **22.8.3, "Precautions in use of SETn built-in macro instruction for IFCRES flag and IFCSTRT flag".**

22.4.8 IF Counter Gate Judge Register (IFCGJDG)

The IF counter gate judge register detects the open/close of the gate of the frequency counter (IF counter function mode only).

The configuration and function of the register are shown below.

Name	Flag b_3	b_2	b_1	b_0	Address	Read/Write
IF counter gate judge register (IFCGJDG)	0	0	0	IFCG	04H	R

Detects open/close of the gare

Detects open/close of the gate of frequency counter.

	IF counter function mode	External gate counter (FCG) function
0	This flag is set "1" when "1" is set to IFCSTRT flag, and the set status is maintained until the gate time terminates.	This flag is set "0" after a certain period of time (the gate time of IF counter function) irrespective of the input applied to $P1A_0$/FCG terminal.
1		

Fixed to "0"

Reset	b_3	b_2	b_1	b_0
Power ON	0	0	0	0
Clock stop				—
CE				—

Do not attempt to enter the content of the IF counter data register (IFCR) into the data buffer while the IFCG flag is set "1" (gate is open).

Open/close of the gate in the FCG function mode cannot be effected by the IFCG flag. The open/close operation of the external gate counter must be controlled by IRQIFC flag.

22.5 USAGE OF IF COUNTER FUNCTION

Sections **22.5.1** thru **22.5.3** explain usage of hardware, program example and counting error of the IF counter function.

22.5.1 IF Counter Hardware Usage Method

Fig. 22-4 shows the block diagram for using $P1D_3$/FMIFC pin and $P1D_2$/AMIFC pin.

Table 22-1 shows the range of frequencies allowed to be applied to $P1D_3$/FMIFC pin and $P1D_2$/AMIFC pin.

The IF counter serves as an input pin having a built-in AC amplifier, as shown in Fig. 22-4, and the DC component of the input signal must be cut off by the condenser C.

Switch SW is turned ON and the voltage of each pin rises to approx. 1/2 V_{DD} if the $P1D_3$/FMIFC pin and $P1D_2$/AMIFC pin are selected as the IF counter function.

In this case, if the pin voltage is not risen to the intermediate voltage, the AC amplifier is put out of the normal operating range, and this may result in erroneous IF counting operation.

To prevent this error, a sufficiently long wait time should be set before starting the counting operation after specifying each pin as IF counter.

Fig. 22-4 Functional block diagram of IF counting operation of each pin

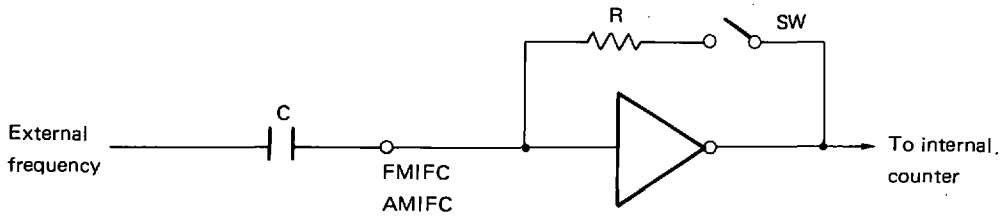


Table 22-1 IF counter input frequency range

Input pin	Permissible input frequency	Input signal amplitude
$P1D_3$/FMIFC	5 – 15 MHz	0.3 V_{p-p}
	10.5 – 10.9 MHz	0.06 V_{p-p}
$P1D_2$/AMFIC	0.1 – 1 MHz	0.3 V_{p-p}
	0.44 – 0.46 MHz	0.05 V_{p-p}

22.5.2 Program Example of IF Counter Function

A programming example of the IF counter function is shown below.

As shown in the example, it is necessary to put a wait time between the execution of the instruction for setting $P1D_3$/FMIFC pin or $P1D_2$/AMIFC pin to the IF counter and the start of the counting operation.

This wait time is necessary to ensure normal operation of the built-in AC amplifier when each pin is selected as IF counter, as has been explained in Section **22.5.1.**

Example:

Counting frequency by $P1D_3$/FMIFC pin (Gate time: 8 ms)

```
        INITFLG NOT IFCMD1, IFCMD0, IFCCK1, NOT IFCCK0
                                ; FMIFC pin is selected and gate time is set as 8 ms.
        [ Wait ]                ; Built-in AC amplifier stabilizing time
        IFC_RES_AND_START ;     Counter reset and start
                                ; (IFCRES and IFCSTRT flags are set) Note
                                ; Macro definition is used because SET2 instruction is not usable.
    LOOP:
        SKT1    IFCG            ; Detection of open/close of gate
        BR      READ            ; Branch to READ: if gate closes
        [ Processing A ]        ; Do not enter the data of IF counter with this processing A.
                                ; Do not set data to CGPR.
        BR      LOOP
    READ:
        GET  DBF,  IFC          ; Reads the value of IF counter data register into data buffer.
```

Note: See the Section **22.8.3, "Precautions in using SETn built-in macro instruction for IFCRES flag and IFCSTRT flag".**

22.5.3 Error of IF Counter

The IF counter error involves the gate time error and counting error. These errors are explained in the following Items (1) and (2).

(1) Gate time error

The gate time of the IF counter is created by dividing the frequency 4.5 MHz. If this 4.5 MHz frequency is deviated by "+x" ppm, then a deviation of "−x" ppm occurs in the gate time.

(2) Counting error

The IF counter counts the number of falling edges of the input signal to know the frequency. If the pin is at the "low" level when the gate is open, one pulse is counted additionally. When the gate is closing, however, counting is not affected by the pin status.

That is, the counting error is "+1, −0".

22.6 USAGE OF FCG FUNCTION

The following Sections **22.6.1** and **22.6.2** explain a program example and error of the FCG function.

22.6.1 Program Example of FCG Function

A programming example of FCG function is shown below. The FCG function opens/closes the gate by using IRQIFC flag.

This open/close of the gate is detected by using an interrupt by IRQIFC flag or by reading the status with a program.

Example:

Internal frequency 100 kHz

```
INITFLG IFCMD1, IFCMD0, NOT IFCCK1, IFCCK0
                        ; Selects FCG function, and sets internal frequency at 100 kHz.
    CLR1   IRQIFC
    IFC_RES_AND_START ; Counter reset and start
                        ; (IFCRES and IFCSTRT flags are set) Note
                        ; Macro definition is used because the use of SETn is not allowed.
LOOP:
    SKF1   IRQIFC       ; Detects open/close status of gate.
    BR     READ         ; Branch to READ: if gate is closed.
    [Processing A]      ; Do not attempt to enter the data of IF counter with this processing A.
                        ; Do not set data to CGPR.
    BR     LOOP
READ:
    GET    DBF,   IFC   ; The value of IF counter data register is entered.
```

Note: See the Section **22.8.3, "Precautions in using SETn built-in macro instruction for IFCRES flag and IFCSTRT flag".**

22.6.2 Error of External Gate Counter

The external gate counter error involves the internal frequency error and counting error. These errors are explained in the following Items (1) and (2).

(1) Internal frequency error

The internal frequency of the external gate counter is created by dividing the frequency 4.5 MHz. If this 4.5 MHz frequency is deviated by "+x" ppm, then a deviation of "+x" ppm occurs in the internal frequency.

(2) Counting error

The external gate counter counts the number of falling edges of the input signal to know the frequency. If the pin is at the "low" level when the gate is open (when the pin input rises), one pulse is counted additionally. When the gate is closing (at the leading edge of the next pin), however, counting is not affected by the pin status.

That is, the counting error is "+1, −0".

22.7 RESET STATUS

22.7.1 Power ON Resetting

$P1D_3$/FMIFC pin and $P1D_2$/AMIFC pin are specified as general purpose input ports.

$P1A_0$/FCG pin is specified as general purpose input/output port.

22.7.2 Clock Stop

P1D_3/FMIFC pin and P1D_2/AMIFC pin are specified as general purpose input ports.

P1A_0/FCG pin is specified as general purpose input/output port.

22.7.3 CE Resetting

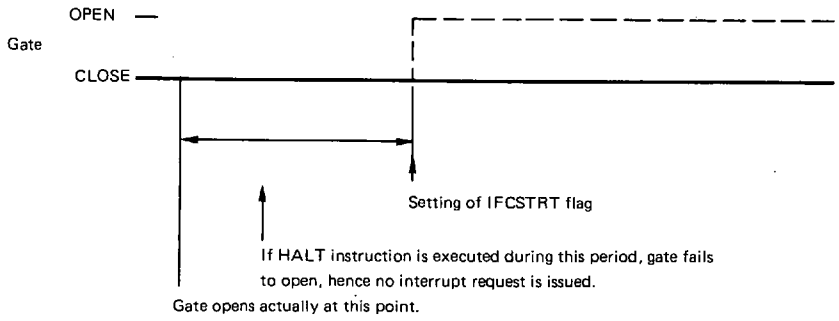
P1D_3/FMIFC pin, P1D_2/AMIFC pin and P1A_0/FCG pin hold their previous status.

22.7.4 Halt Status

P1D_3/FMIFC pin, P1D_2/AMIFC pin and P1A_0/FCG pin hold their most previous status.

When canceling the halt state by the interruption of the frequency counter, the following precaution must be observed: If "HALT" instruction is executed within the period beginning with the start of the counting operation by IFCTRT flag and ending with the actual opening of the gate, then the gate will fail to open.

To avoid this error, "HALT" instruction must be executed after placing a wait of more than 1 ms when the IF counter function is used. When using FCG function, the "HALT" instruction must be executed after conforming that the FCG pin has risen from "low" to "high" level.



22.8 PRECAUTIONS IN USING FREQUENCY COUNTER

The frequency counter serves also as the clock generator port.

Accordingly, the clock generator port and frequency counter cannot be used at the same time.

If the data of the IF counter mode select register and IF counter data register is processed while the frequency counter is being used as a clock generator port, the operation shown in Section **22.8.1** is executed.

If the data of the IF counter mode select register and CGP data register (peripheral address 20H) is processed while being used as the frequency counter, the operation as shown in Section **22.8.2** is performed.

As explained in Section **22.4.7**, IFCRES flag and IFCSTRT flag do not permit use of SETn instruction which is an assembler (AS17K) incorporating macro instruction. This is explained in Section **22.8.3.**

22.8.1 Usage as Clock Generator Port

(1) When IFCMD1 and IFCMD0 flags of IF counter mode select register are operated

If any value other than "0" is written into IFCMD1 and IFCMD0 flags, P1B_0/CGP pin holds the output level at the time of data setting, and stops its CGP operation.

When resetting both flags of IFCMD1 and IFCMD0 flags to "0" again, it starts the CGP operation.

(2) When IF counter data register is operated

The CGP operation is not affected by none of GET (reading) and PUT (writing).

When reading is executed, "undefined value" will be entered, and when writing is executed, nothing will be changed.

The IF counter data register is a peripheral register provided specially for reading, and no writing operation is allowed.

22.8.2 When Using as Frequency Counter

(1) When IFCMD1 and IFCMD0 flags of IF counter mode select register are operated

If "0" is written into IFCMD1 and IFCMD0 flags, $P1B_0$/CGP pin performs the CGP data register operation specified at the data set time.

This CGP operation, however, also requires CGPON flag of the PWM mode select register to be set.

If the previous value is set to IFCMD1 and IFCMD0 again, the frequency counter continues its operation, but the accuracy of the input data is not guaranteed.

The frequency counter does not count frequency while the CGP operation mode is selected.

(2) When CGP data register is operated

Execution of GET (reading) does not affect the frequency counting operation.

Execution of PUT (writing) prevents normal frequency counting operation.

When reading, "undefined" value will be entered.

When writing, a value is written into the upper 6 bits of the frequency counter.

22.8.3 Precautions in Using SETn Incorporating Macro Instruction for IFCRES Flag and IFCSTRT Flag

The IFCRES flag and IFCSTRT flag are located in the write-only register, and "undefined" value is read when these flags are read.

In this case, if the "SETn" instruction which is an assembler (AS17K) incorporating macro instruction is used to set either flag, an "undefined" value is entered to the other flag.

```
SET1  IFCRES
      PPEK  WR,  MF.IFCRES  SHR 4      ;  The value of WR is "undefined".
      OR    WR,  #.DF.IFCRES AND 0FH ;  Bit b0 of WR only is set.
      POKE  .MF. 1FCRES SHR 4, WR      ;  Undefined value is entered to IFCSTRT (bit b1).
```

If SETn instruction is used for IFCRES flag and IFCSTRT flag, the assembler (AS17K) generates an error.

When operating the IFCRES flag and IFCSTRT flag, the following macro definition is recommended.

```
IFCW    MEM  0.0A3H
;
IFC_RES_AND_START  MACRO
        MOV  WR, #0011B
        POKE  IFCM, WR
ENDM
;
IFC_RES                MACRO
        MOV  WR, #0001B
        POKE  IFCW, WR
ENDM
;
IFC_START              MACRO
        MOV  WR, #0010B
        POKE  IFCW, WR
ENDM
```

The macro instructions shown above are attached as "IFCSET.LIB" file (D17005.DEV).

Accordingly, the above-mentioned macro instruction can be used directly by including this IFCSET.LIB file in the source program.

23. LCD CONTROLLER/DRIVER

The Liquid Crystal Display (LCD) controller/driver permits LCD display of maximum 60 dots by combining the segment signal output and common signal output.

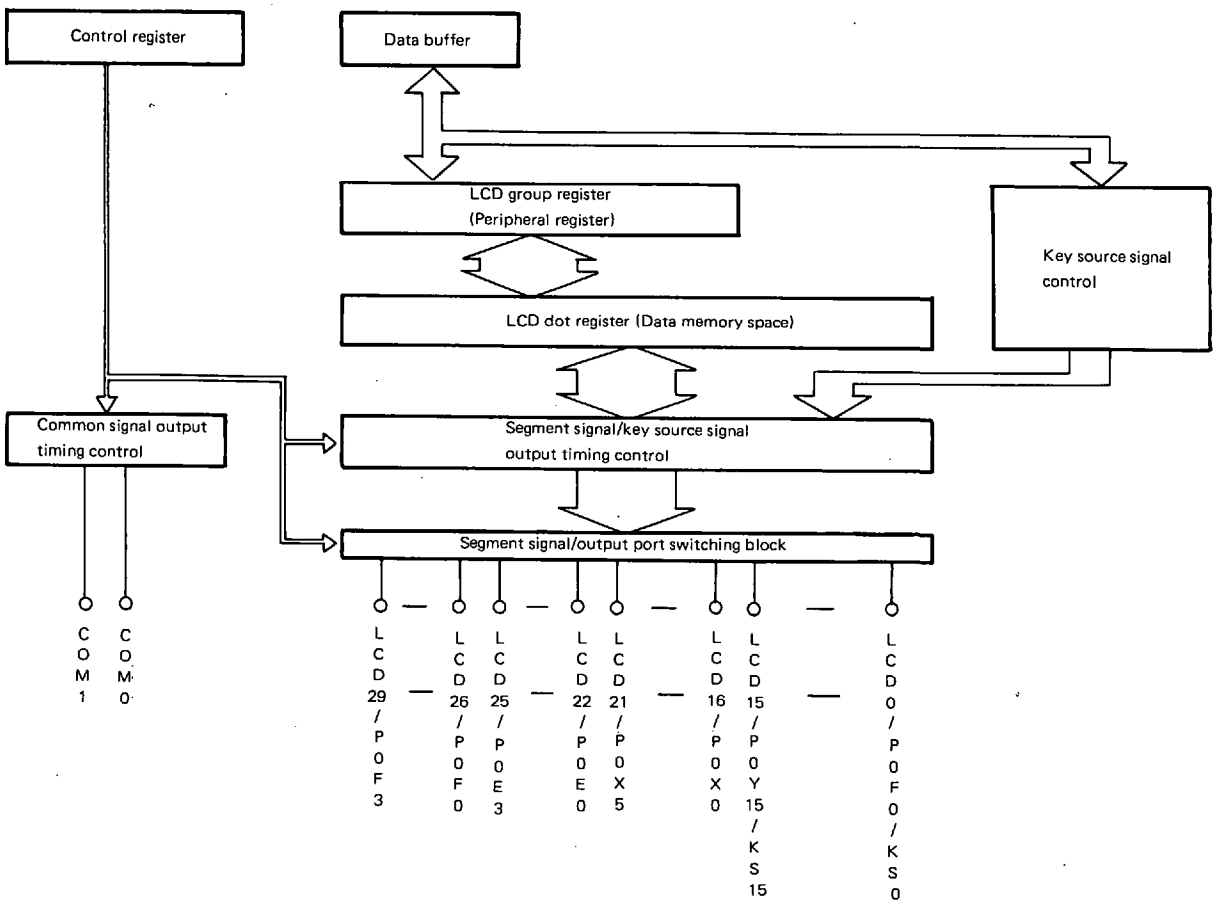
23.1 CONFIGURATION OF LCD CONTROLLER/DRIVER

Fig. 23-1 shows a block diagram of the LCD controller/driver.

As shown in Fig. 23-1, the LCD controller/driver consists of a common signal timing generation block, segment signal/key source signal timing generation block, segment signal/output port switching block, LCD dot data register, LCD group data register and key source signal control block.

Section **23.2** outlines the functions of each block.

Fig. 23-1 LCD controller/driver block diagram



23.2 OUTLINE OF FUNCTION OF LCD CONTROLLER/DRIVER

The LCD controller/driver controls display of maximum 60 dots through combined use of the common signal output pins (COM_1, COM_0 pins) and segment signal output pins ($LCD_{29}/P0F_3$ – $LCD_0/P0Y_0/KS_0$ pins).

Fig. 23-2 shows the relationship between the common signal output pins, segment signal output pins and displayed dots.

As shown in Fig. 23-2, one segment line is capable of displaying 2 dots at the intersecting points with COM_1 and COM_0 pins.

The 1/2 duty, 1/2 bias drive system is used, and the source voltage V_{DD} is used as the drive voltage.

The segment signal output pins ($LCD_{29}/P0F_3$ – $LCD_0/P0Y_0/KS_0$ pins) can be used as the general purpose output ports.

When using as a general purpose output port, Port 0F
($LCD_{29}/P0F_3$ – $LCD_{26}/P0F_0$ pins), Port 0E
($LCD_{25}/P0E_3$ – $LCD_{22}/P0E_0$ pins), Port 0X
($LCD_{21}/P0X_5$ – $LCD_{16}/P0X_0$ pins) and Port 0Y
($LCD_{15}/P0Y_{15}/KS_{15}$ – $LCD_0/P0Y_0/KS_0$ pins) can be used independently.

Among the segment signal output pins, $LCD_{15}/P0Y_{15}/KS_{15}$ – $LCD_0/P0Y_0/KS_0$ pins can be used as the key source signal output pins.

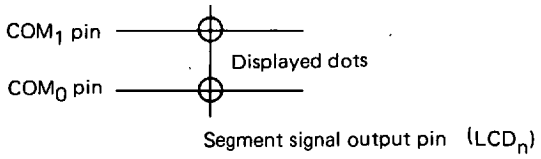
The key source signal output is output to the LCD segment signal output on the time sharing basis.

For the general purpose output port, refer also to Section **17, "General Purpose Port".**

For the key source signal output, refer also to Section **24, "Key Source Controller/Decoder".**

The following Sections **23.2.1** thru **23.2.6** outline the functions of the blocks of the LCD controller/driver.

Fig. 23-2 Common signal output, segment signal output and displayed dots



23.2.1 LCD Dot Register

The LCD dot register sets the dot data for LCD dots.

The LCD dot register is allocated on the data memory, hence it can be controlled by any data memory control instruction.

When using the segment signal output pin as general purpose output port, the output data must be set.

For details, see Section **23.3.**

23.2.2 LCD Group Register

The LCD group register sets the dot data for LCD dots.

The LCD group register sets the data via the data buffer.

When data is set in the LCD group register, the value of the corresponding LCD dot register changes at the same time.

When using the segment signal output pin as general purpose output port, the output data must be set.

In this case, too, the value of the corresponding LCD dot register changes as the data is set to the LCD group register.

For details, see section **23.3.**

23.2.3 Common Signal Output Timing Control Block

The common signal output timing control block controls the common signal output timing of COM_1 pin and COM_0 pin.

When no LCD display is specified, "low" level output is issued.

Whether LCD display is used or not is determined by the LCD mode select register (RF address 10H).

For details, see Section **23.4.**

23.2.4 Segment Signal/key Source Signal Output Timing Control Block

The segment signal/key source signal output timing control block control the segment signal output timing of

$LCD_{29}/P0F_3$ pin – $LCD_0/P0Y_0/KS_0$ pin.

When no LCD display is specified, "low" level output is issued.

Whether the LCD display is used or not is determined by the LCD mode select register.

This control block controls the timing of the key source signal and segment signal output from $LCD_{15}/P0Y_{15}/KS_{15}$ pin – $LCD_0/P0Y_0/KS_0$ pin.

Whether the key source signal is to be used or not is determined by the LCD mode select register.

For details, see Section **23.4.**

23.2.5 Segment Signal/general Purpose Port Switching Block

The segment signal/general purpose port switching block determines whether each segment signal output pin is to be used for LCD display (segment signal output) or used a general purpose port.

Switching between the segment signal output port and general purpose output port is performed by the LCD port select register (RF address 11H).

For details, see Section **23.4.**

23.2.6 Key Source Signal Output Control Block

The key source signal output control block is used to set the key source signal output data to be output from $LCD_{15}/P0Y_{15}/KS_{15}$ pin – $LCD_0/P0Y_0/KS_0$ pin.

Setting of the key source signal output data is performed by the key source data register (peripheral address 42H) via the data buffer.

The key source data register sets also the output data of Port 0Y.

When using the key source signal, $P0D_3/ADC_5$ (No. 75 pin) – $P0D_0/ADC_2$ pin (No. 78 pin) are used as the key input pins.

For details, see **24, "Key Source Controller/Decoder".**

23.3 LCD DOT REGISTER AND LCD GROUP REGISTER

The LCD dot register and LCD group register set the ON/OFF data of each display dot.

23.3.1 Configuration of LCD Dot Register

Fig. 23-2 shows the arrangement of the LCD dot register on the data memory.

Fig. 23-3 shows the configuration of the LCD dot register.

Fig. 23-2 Arrangement of LCD dot register on data memory

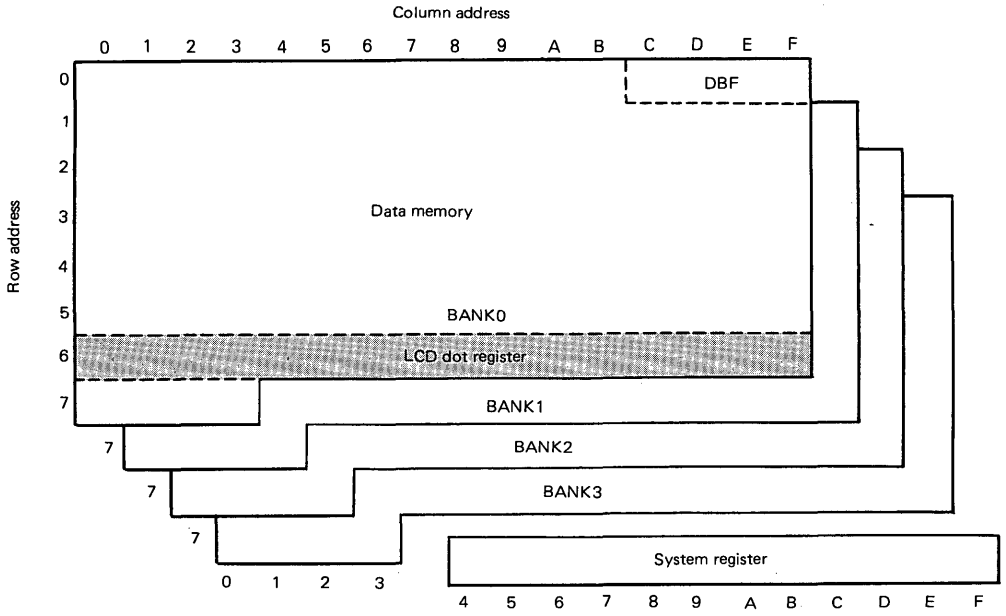
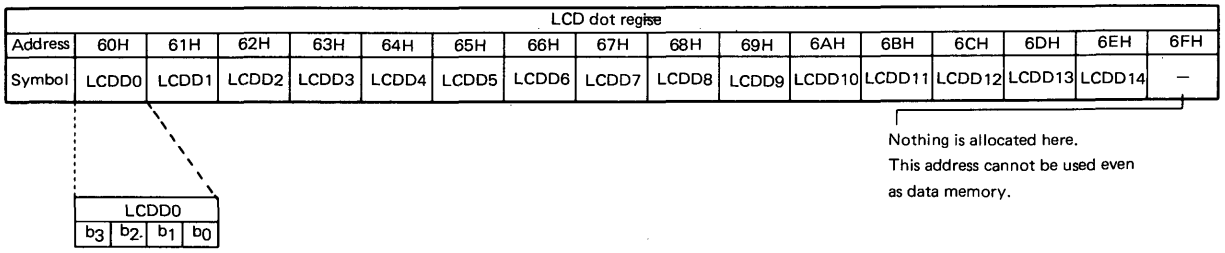


Fig. 23-3 Configuration of LCD dot register



23.3.2 Functions of LCD Dot Register

Fig. 23-4 shows the relationship between one nibble (4 bits) of the LCD dot register and LCD display dot.

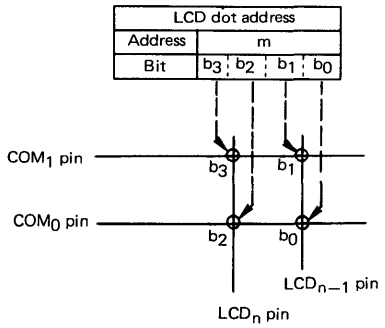
As shown in Fig. 23-4, four dots of display data (ON/OFF data) can be set by one nibble of the LCD dot register.

The LCD display dot set as "1" is lit ON, while the dot set as "0" is OFF.

The LCD dot register sets the output data when the segment signal output pin is used as output port.

Fig. 23-6 shows the relationship between each LCD dot register and the LCD display dot which turns ON or OFF corresponding to the display data.

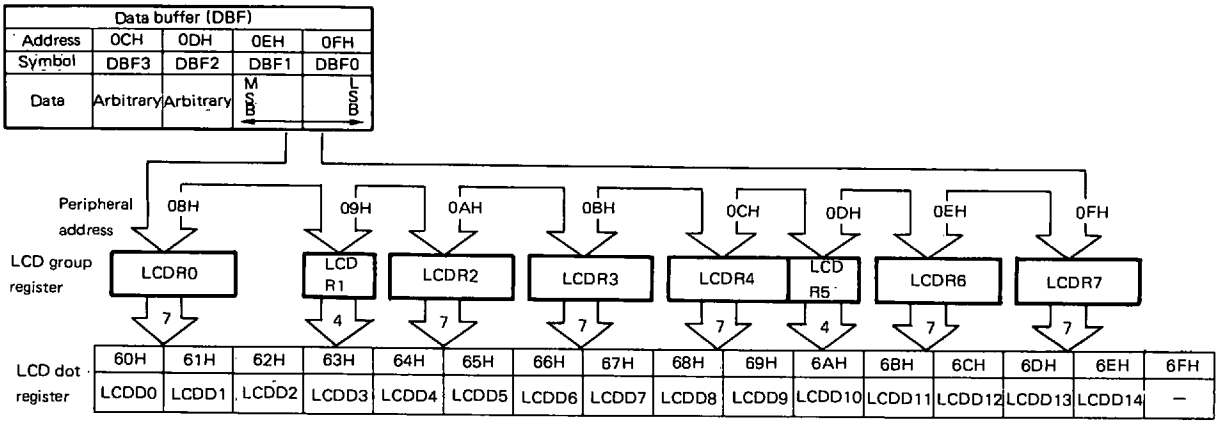
Fig. 23-4 Relation between one nibble of LCD dot register and LCD display dot



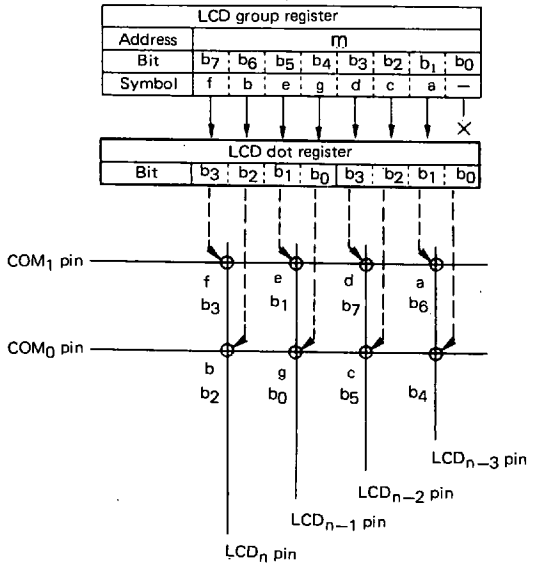
23.3.3 Configuration of LCD Group Register

Fig. 23-5 shows the configuration of the LCD group register and its relation to the LCD dot register.

Fig. 23-5 Configuration of LCD group register and its relation with LCD dot register



Relation between LCD group register and LCD display dot



23.3.4 Function of LCD Group Register

The LCD group register sets the ON/OFF data of the LCD display dot in the same way as the LCD dot register.

As shown in Fig. 23-5, the LCD group register sets the data in units of seven dots or four dots via the data buffer.

That is, when "PUT LCDRn, DBF" instruction is executed, the LCD display data of the group specified by "n" ($0 \leqq n \leqq 7$) is set.

In this case, the value of the corresponding LCD dot register changes at the same time when the "PUT LCDRn, DBF" instruction is executed.

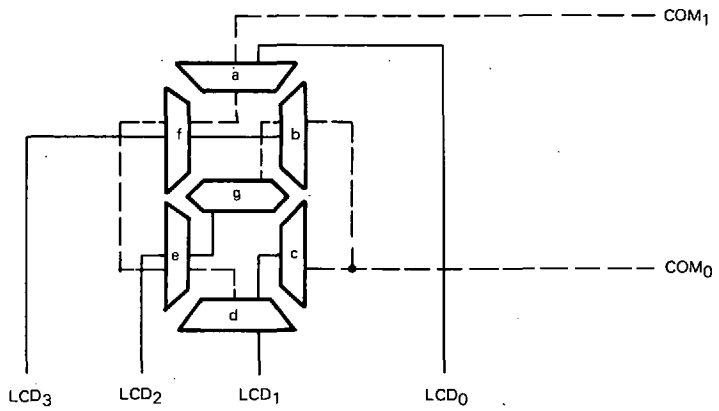
In other words, the 7-dot display data can be set by single instruction if the LCD group register is used.

The LCD dot register sets the output data when the segment signal output pin is used as the output port.

Section **23.3.5** shows the relationship between the LCD group register and data buffer.

Fig. 23-6 shows the relation between each LCD group register and the LCD display dot which is turned ON or OFF correspondingly.

The LCD group register is capable of setting 7-dot display data by single instruction, hence it can be used conveniently for 7-segment display, etc. using the wiring shown below.



23.3.5 LCD Group Register

The configuration and function of each LCD group register are shown below.

The LCD group register sets the ON and OFF dots of LCD display for each group.

The LCD group register permits setting of ON/OFF of maximum seven dots at the same time.

Name	Data buffer															
Symbol	DBF 3				DBF 2				DBF 1				DBF 0			
Address	0 CH				0 DH				0 EH				0 FH			
Bit	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
Data	Arbitrary				Arbitrary				Transfer data							

8

GET is undefined data

PUT allowed

Peripheral register											
Name		b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Symbol	Peripheral address	Peripheral hardware
LCD group register	0	Effective data							—	LCDR 0	08 H	LCD segment group 0
LCD group register	1	—				—		—	—	LCDR 1	09 H	" 1
LCD group register	2	Effective data							—	LCDR 2	0 AH	" 2
LCD group register	3	Effective data							—	LCDR 3	0 BH	" 3
LCD group register	4	Effective data							—	LCDR 4	0 CH	" 4
LCD group register	5	—	—	—	—	Effective data			—	LCDR 5	0 DH	" 5
LCD group register	6	Effective data							—	LCDR 6	0 EH	" 6
LCD group register	7	Effective data							—	LCDR 7	0 FH	" 7

Arbitrary	
ON/OFF of each LCD display dot (segment) is set	
Segment a	
Segment c	
Segment d	
Segment g	
Segment e	
Segment b	
Segment f	
0	OFF
1	ON

For the relation between segments a-g and each dot, see Fig. 23-6.

(MEMO)

Fig. 23-6 Relationship between LCD display dot, Port 0E – Port 0Y, key source output and each data setting register (1/2)

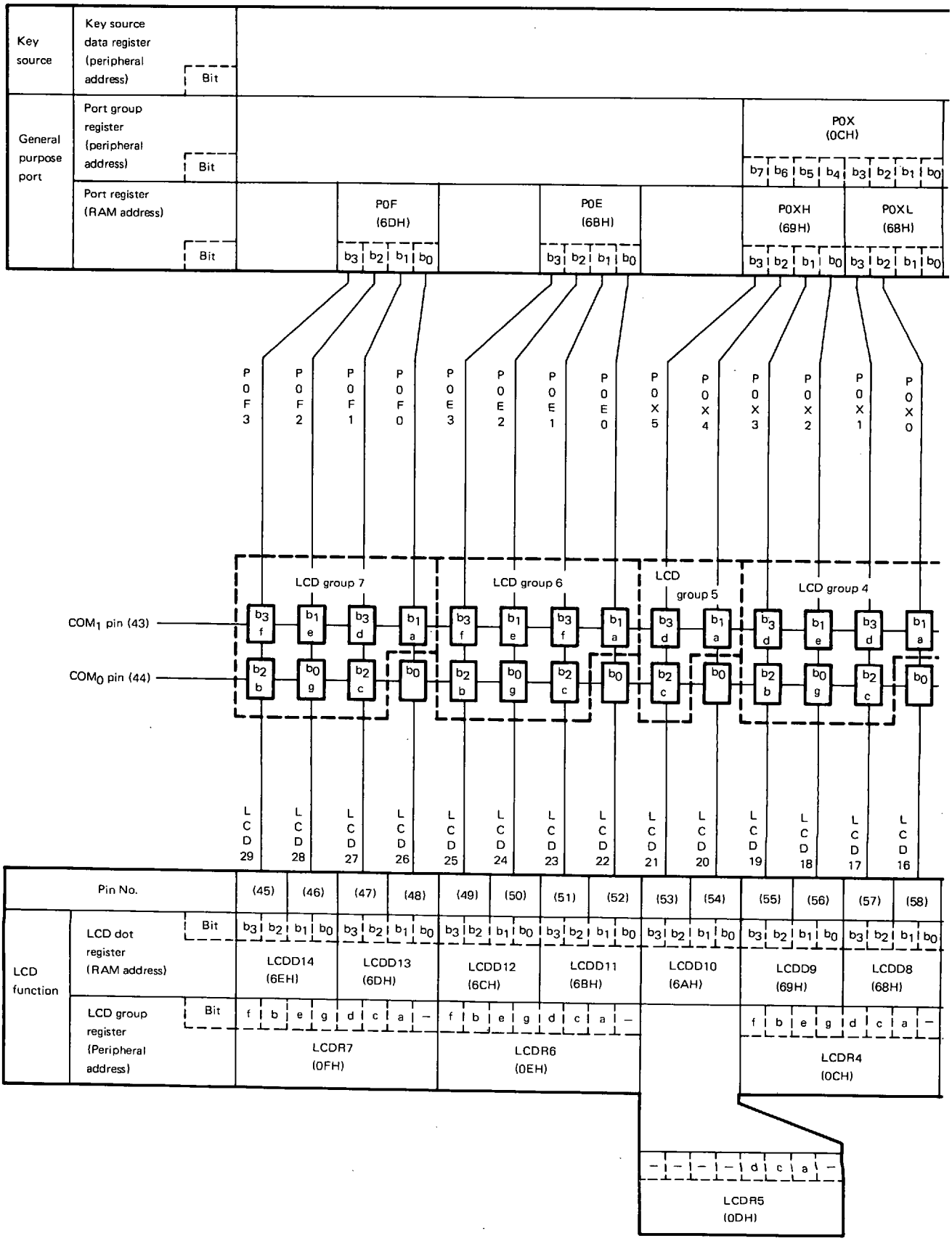
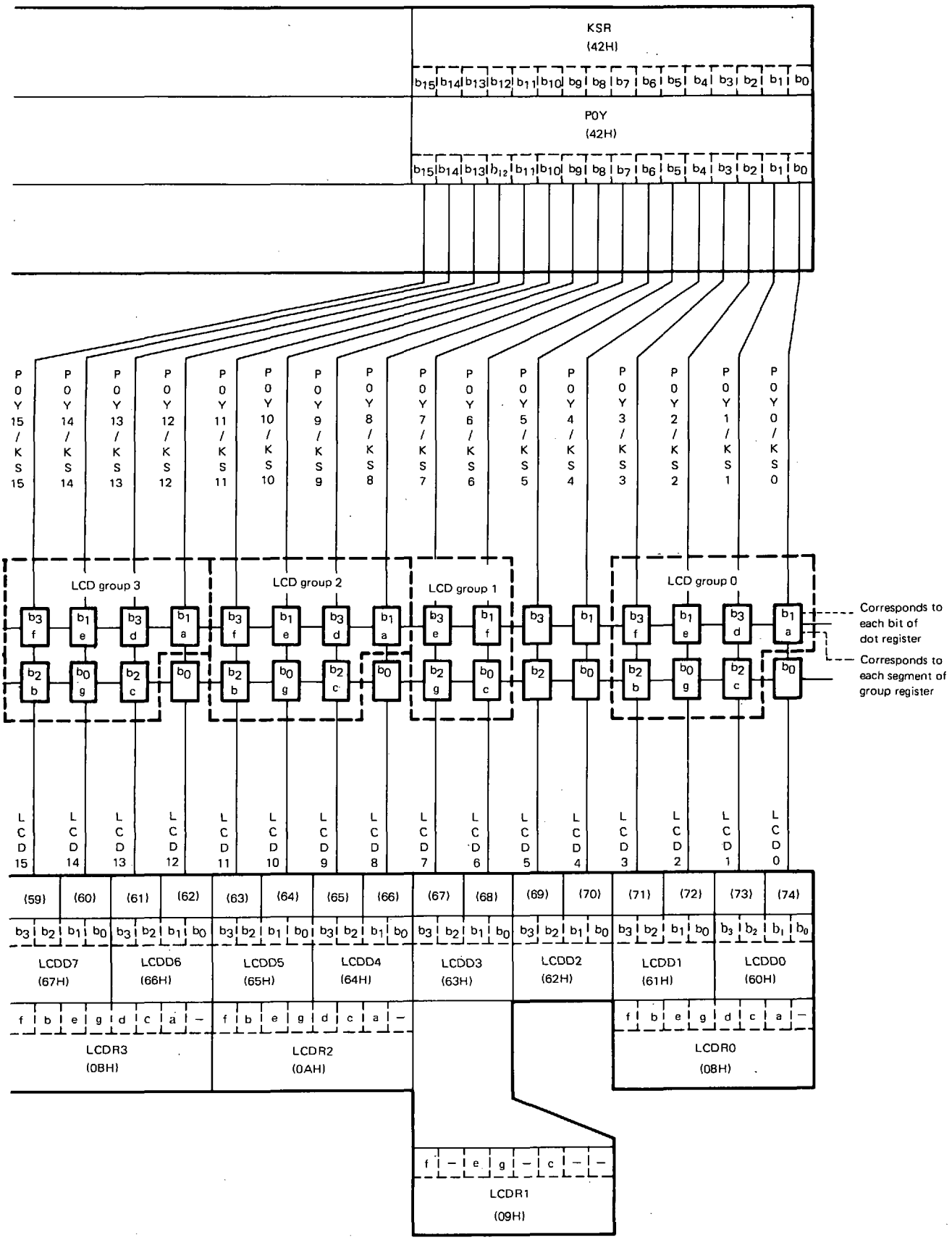


Fig. 23-6 Relationship between LCD display dot, Port 0E – Port 0Y, key source output and each data setting register (2/2)

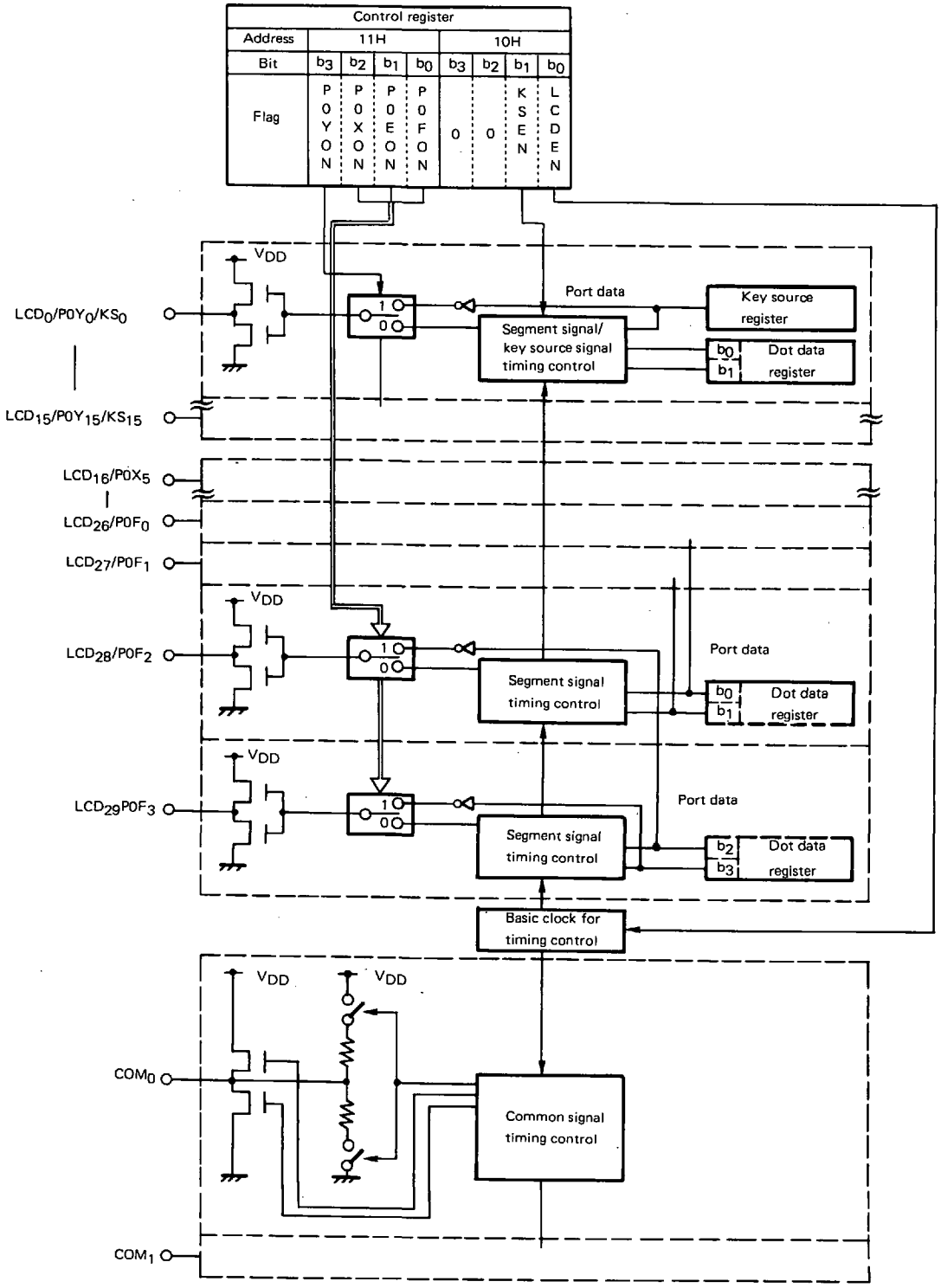


23.4 OUTPUT TIMING CONTROL BLOCK AND SEGMENT/PORT SWITCHING BLOCK

23.4.1 Configuration of Output Timing Control Block and Segment/port Switching Block

Fig. 23-7 shows the configuration of the common signal and segment signal/key source signal output timing control block and segment signal/general-purpose output port switching block.

Fig. 23-7 Configuration of timing control block and port switching block



23.4.2 Function of Segment Signal/general-Purpose Output Port Switching Block

The segment signal/general-purpose output port switching block determines whether each pin is used as the segment signal output port or as the general-purpose output port according to the P0YON – P0FON flags of the LCD port select register.

Each pin is set as general-purpose output port when the corresponding flag is "1".

The segment pin which is not selected as general-purpose output port permits LCD display.

$LCD_{15}/P0Y_{15}/KS_{15}$ pin – $LCD_0/P0Y_0/KS_0$ pin permit simultaneous output of both segment signal and key source signal, but priority is given to the output of the port for which Port 0Y is selected.

For the general-purpose output port, see Section **17, "General-Purpose Port".**

Section **23.4.4** shows the function and configuration of the LCD port select register.

23.4.3 Function of Output Timing Control Block

The output timing control block controls the timing of the LCD display common signal and segment signal and the timing of the key source signal and segment signal when the key source controller/decoder is used.

The common signal and segment signal are output when LCDEN flag of the LCD mode select register is "1".

This means that all the LCD display can be turned out by using the LCDEN flag.

When the LCD display is OFF, the common signal and segment signal are kept at "low" level.

The key source signal is output when KSEN flag of the LCD mode select register is "1".

That is, the key source signal can be handled by the KSEN flag.

Section **23.4.5** shows the configuration and function of the LCD mode select register.

Section **23.4.6** shows the output waveforms of the common signal and segment signal.

For the key source controller/decoder, see Section **24, "Key Source Controller/Decoder".**

23.4.4 LCD Port Select Register (LCDPORT)

The LCD port select register select the LCD segment signal output pin as the general-purpose output port.

The configuration and function of this register are shown below.

Name	Flag				Address	Read/Write
	b_3	b_2	b_1	b_0		
LCD port select register (LCDPORT)	P0YON	P0XON	P0EON	P0FON	11H	R/W

	Select LCD segment signal output pin as general-purpose output port.
P0FON = 1	$LCD_{26}/P0F_0$ – $LCD_{29}/P0F_3$ pin (Port 0F)
P0EON = 1	$LCD_{22}/P0E_0$ – $LCD_{25}/P0E_3$ pin (Port 0E)
P0XON = 1	$LCD_{16}/P0X_0$ – $LCD_{21}/P0X_5$ pin (Port 0X)
P0YON = 1	$LCD_0/P0Y_0/KS_0$ – $LCD_{15}/P0Y_{15}/KS_{15}$ pin (Port 0Y)
0	LCD segment signal output
1	General-purpose output port

Reset	Power ON	0	0	0	0
	Clock stop	0	0	0	0
	CE	0	0	0	0

23.4.5 LCD Mode Select Register

The LCD mode select register controls ON/OFF of all the LCD display dots and the key source signal.

The configuration and function of this register are shown below.

Name	Flag b_3	b_2	b_1	b_0	Address	Read/Write
LCD mode and select register (LCDMODE)	0	0	KSEN	LCDEN	10H	R/W

LCDEN — Determines ON/OFF of all the LCD display dots

0	Display OFF (All the segment output and common output pins are at "low" level)
1	Display ON

KSEN — Determines key source signal output

0	Key source OFF
1	Key source ON

b_3 — Fixed to "0"

		b_3	b_2	b_1	b_0
Reset	Power ON	0	0	0	0
	Clock stop			0	0
	CE			0	0

23.4.6 Common Signal and Segment Signal Output Waveform

Figs. 23-8 and 23-9 show the output waveforms of common signal and segment signal.

Fig. 23-8 shows the waveform which is output when key source signal is not being output, while Fig. 23-9 shows the waveform which is output while the key source signal is being output.

As shown in Fig. 23-8, the LCD driver outputs the signal of frame frequency 250 Hz on the 1/2 duty and 1/2 bias basis (voltage balance method).

The common signals are output in 3 different voltage levels (0, 1/2 V_{DD}, V_{DD}) from COM_1 and COM_0 pins, and these 2 signals have phase difference of 1/4 each other.

In other words, the common signal is set at the center voltage of 1/2 V_{DD}, and it varies between +1/2 V_{DD} and −1/2 DD. This display method is called the 1/2 bias drive system.

The segment signal is output from each segment signal output pin in 2 levels (0, V_{DD}), and their phase difference corresponds the display dots (A and B).

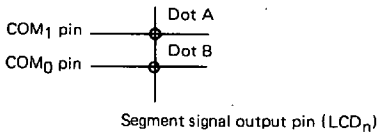
The 4 phase differences are shown in (1) thru (4) in the following Figure as four combinations of ON and OFF of dots A and B arranged to turn ON and OFF 2 display dots (A and B) by single segment pin.

Dots A and B light if the potential difference between common signal and segment signal reaches V_{DD}.

In other words, each of the dots A and B lights at the duty of 1/2, and the lighting frequency is 250 Hz.

This display system is called the 1/2 duty driving system, and the frequency is called the frame frequency.

Fig. 23-8 Output waveforms of common signal and segment signal (when key source signal is not output)



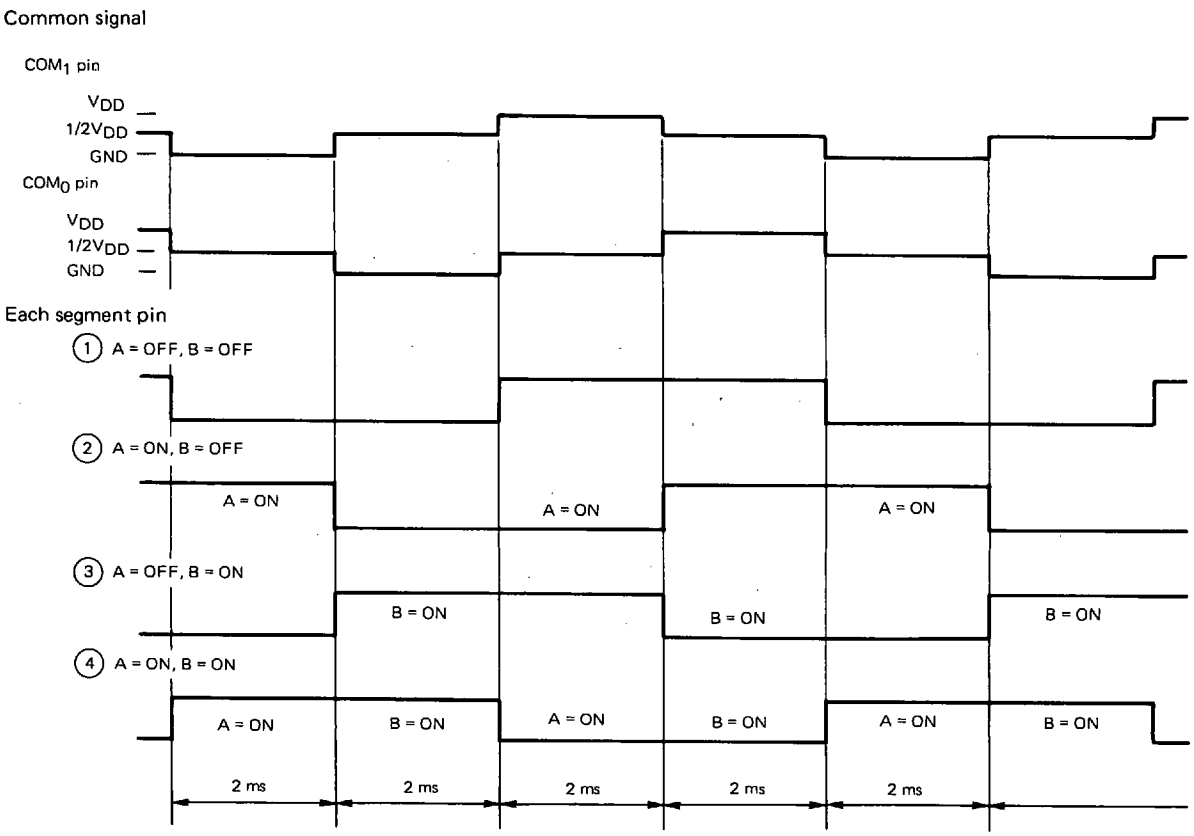
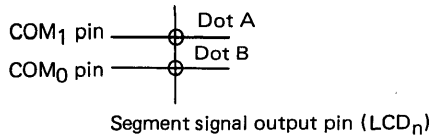
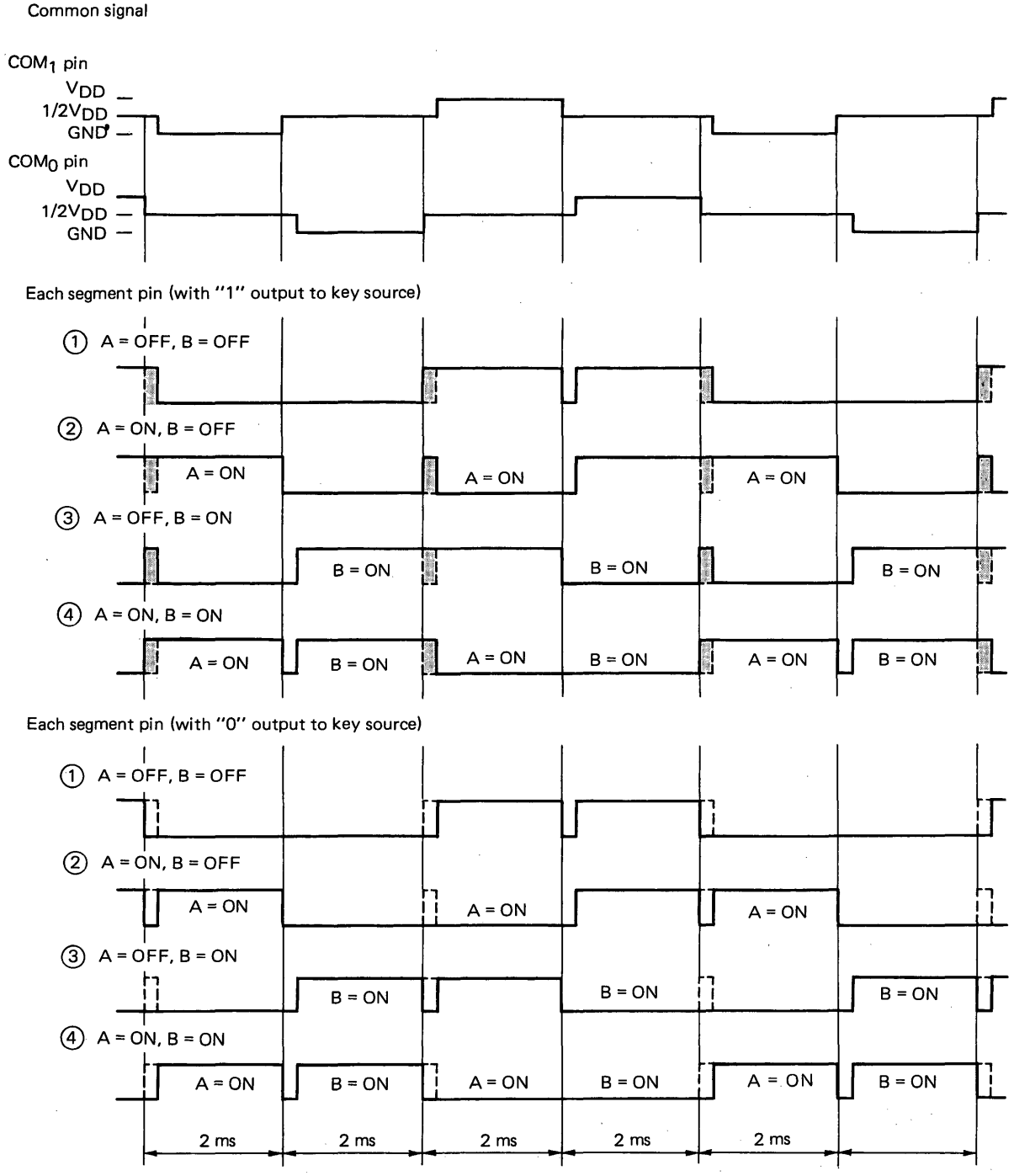


Fig. 23-9 Output waveforms of common signal and segment signal (when key source signal is output)





23.5 USAGE OF LCD CONTROLLER/DRIVER

Fig. 23-10 shows an wiring example of the LCD panel. Shown below is an example of program for lighting up the 7-segment display units by LCD0 – LCD3 pins.

Example:

```
PMNO    MEM   0.01H    ;  Preset memory number and BK data storage area
CH      FLG   0.0FH.1  ;  The LSB of DBF is defined by symbol as "CH" display flag.
LCDDATA:               ;  Display table data
;                                        b3 b2 b1 b0 b3 b2 b1 b0
                                                    ;  Corresponding to LCD dot register
;                                        f  b  e  g  d  c  a  —
                                                    ;  Corresponding to LCD group register
    DW  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  B;  BLANK
    DW  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  B;  1
    DW  0  0  0  0  0  0  0  0  0  1  1  1  1  0  1  0  B;  2
    DW  0  0  0  0  0  0  0  0  0  1  0  1  1  1  1  0  B;  3
    DW  0  0  0  0  0  0  0  0  1  1  0  1  0  1  0  0  B;  4
    DW  0  0  0  0  0  0  0  0  1  0  0  1  1  1  1  0  B;  5
    DW  0  0  0  0  0  0  0  0  1  0  1  1  1  1  1  0  B;  6
    DW  0  0  0  0  0  0  0  0  1  1  0  0  0  1  1  0  B;  7
    DW  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  0  B;  8
    DW  0  0  0  0  0  0  0  0  1  1  0  1  1  1  1  0  B;  9
    DW  0  0  0  0  0  0  0  0  1  1  1  0  1  1  1  0  B;  A
    DW  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  0  B;  B
    DW  0  0  0  0  0  0  0  0  1  0  1  0  1  0  1  0  B;  C
    DW  0  0  0  0  0  0  0  0  1  1  1  0  1  1  1  0  B;  D
    DW  0  0  0  0  0  0  0  0  1  0  1  1  1  0  1  0  B;  E
    DW  0  0  0  0  0  0  0  0  1  0  1  1  0  0  1  0  B;  F

    INITFLG NOT P0YON, NOT P0XON, NOT P0EON, NOT P0FON
    MOV   RPH,   #0000B
    MOV   RPL,   #1110B

    ; MOV  AR3,   #.DL.  (LCDDATA SHR 12)  AND 000FH
    ; MOV  AR2,   #.DL.  (LCDDATA SHR 8)   AND 000FH
    MOV   AR1,   #.DL.  (LCDDATA SHR 4)   AND 000FH
    MOV   AR0,   #.DL.  (LCDDATA SHR 0)   AND 000FH
    ADD   AR0,   PMNO
    ADDC  AR1,   #0
    ADDC  AR2,   #0
    ADDC  AR3,   #0
    MOVT  DBF,   @AR
    SKGE  PMNO,  #0AH
    SET1  CH
    PUT   LCDR0, DBF
    SET1  LCDEN
```

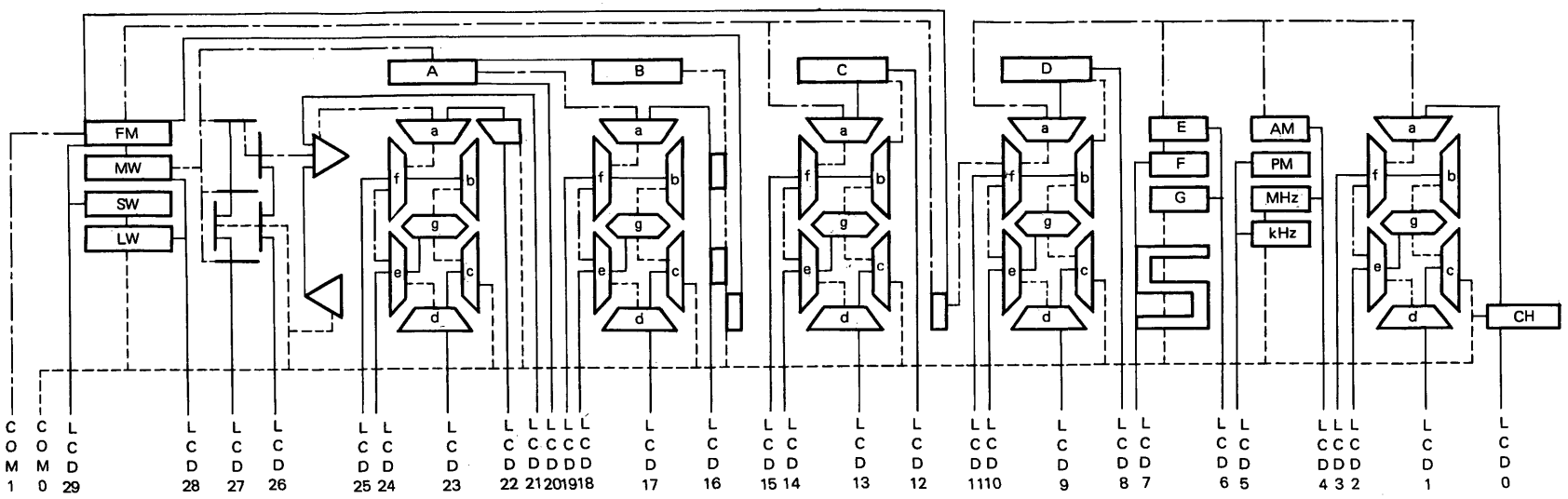


Fig. 23-10 Wiring example of LCD panel

Segment pins, comon pins and corresponding LCD panel displays

Segment pins / Common pins		LCD29	LCD28	LCD27	LCD26	LCD25	LCD24	LCD23	LCD22	LCD21	LCD20	LCD19	LCD18	LCD17	LCD16	LCD15	LCD14	LCD13	LCD12	LCD11	LCD10	LCD9	LCD8	LCD7	LCD6	LCD5	LCD4	LCD3	LCD2	LCD1	LCD0
		45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74
COM_1	43	FM	MW			f	e	d	a	▷	A	f	e	d	a	f	e	d	a	f	e	d	a	F	E	PM	AM	f	e	d	a
COM_0	44	SW	LW			b	g	c		◁	B	b	g	c	:	b	g	c	C	b	g	c	D		G	kHz	MHz	b	g	c	CH

23.6 RESET STATUS

23.6.1 Power ON

$LCD_{29}/P0F_3$ thru $LCD_0/P0Y_0/KS_0$ pins are specified as LCD segment signal output pins which output "low" level potential (display OFF).

COM_1 and COM_0 pins output "low" level potential.

23.6.2 Clock Stop

$LCD_{29}/P0F_3$ thru $LCD_0/P0Y_0/KS_0$ pins are specified as LCD segment signal output pins which output "low" level potential (display OFF).

COM_1 and COM_0 pins output "low" level potential.

23.6.3 CE Reset

Among $LCD_{29}/P0F_3$ thru $LCD_0/P0Y_0/KS_0$ pins, those specified as segment signal output pin output the segment signal, and those specified as general purpose port hold the current output value.

COM_1 and COM_0 pins output common signal.

23.6.4 Halt Status

Among $LCD_{29}/P0F_3$ thru $LCD_0/P0Y_0/KS_0$ pins, those specified as segment signal output pin output the segment signal, and those specified as general purpose port hold the current output value.

COM_1 and COM_0 pins output common signal.

24. KEY SOURCE CONTROLLER/DECODER

The key source controller/decoder composes maximum 64 key matrixes by outputting key source signal time-divided with LCD segment signal output.

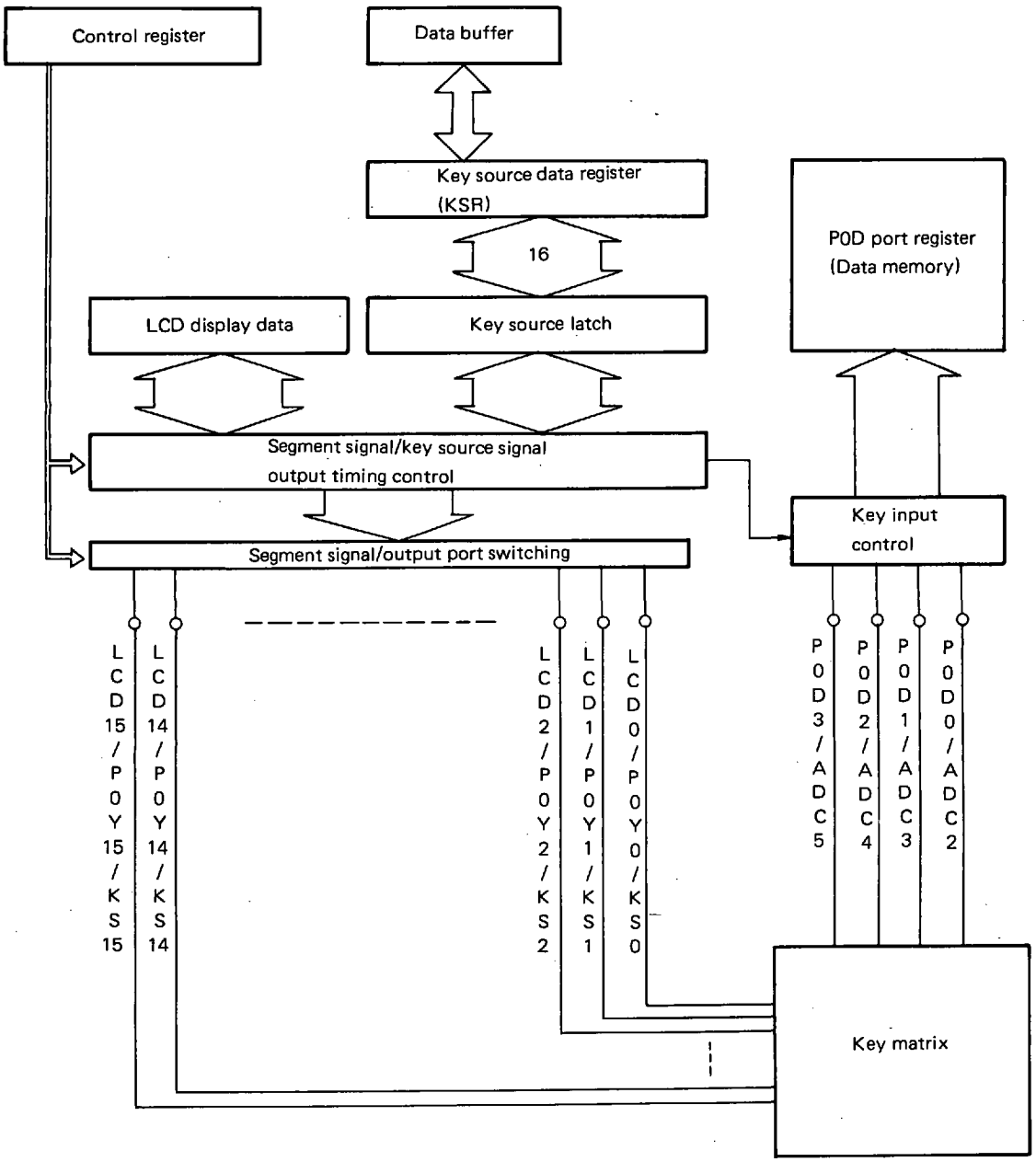
24.1 CONFIGURATION OF KEY SOURCE CONTROLLER/DECODER

Fig. 24-1 shows the configuration of the key source controller/decoder.

As shown in Fig. 24-1, the key source controller/decoder is composed of a segment signal/key source signal timing control block, segment signal/output port switching block, key source data register, key input control block and POD port register.

Section **24.2** outlines the function of each block.

Fig. 24-1 Configuration of key source controller/decoder



24.2 OUTLINE OF FUNCTIONS OF KEY SOURCE CONTROLLER/DECODER

The key source controller/decoder permits maximum 64 key matrixes to be composed by the key source signal output pins ($LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_0/P0Y_0/KS_0$ pins) and key input pins ($P0D_3/ADC_5$ thru $P0D_0/ADC_2$ pins).

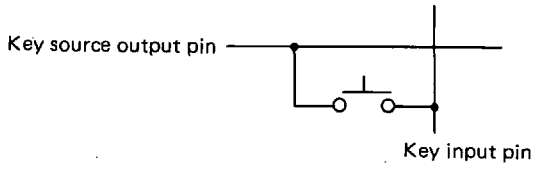
Fig. 24-2 shows an example of composition of the key matrix.

$LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_0/P0Y_0/KS_0$ pins serve also as the LCD segment signal output pins.

Accordingly, the key source signal is output on time division basis with the LCD segment signal output.

The following Sections **24.2.1** to **24.2.6** outline the function of each block of the key source controller/decoder.

Fig. 24-2 Key matrix composition example



24.2.1 Segment Sgnal/key Source Signal Output Timing Control Block

The segment signal/key source signal output timing control block controls the output timing of the key source signal and segment signal from $LCD_{29}/P0F_3$ thru $LCD_0/P0Y_0/KS_0$ pins.

Whether or not to use the key source signal is selected by the LCD mode select register.

The key source signal is not output if no LCD display is used. In such a case, these pins are kept at the "low" level.

Whether or not to use LCD display is selected by the LCD mode select register.

For details, see Section **24.3.**

24.2.2 Segment Signal/general-Purpose Port Switching Block

The segment signal/general-purpose port switching block determines whether the $LCD_{29}/P0F_3$ thru $LCD_0/P0Y_0/KS_0$ pins are to be used for LCD display (for outputting segment signal) or to be used as general-purpose output port.

This switching is performed by the LCD port select register.

When using the key source signal, it is necessary to set the $LCD_{29}/P0F_3$ thru $LCD_0/P0Y_0/KS_0$ pins for outputting the LCD segment signal.

For details, see Section **24.3.**

24.2.3 Key Source Data Register (KSR)

The key source data register sets the key source output data of the pin from which the key source signal is to be output.

The key source data register sets the data via the data buffer.

If data is set to the key source data register, the $LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_0/P0Y_0/KS_0$ pins output the key source data.

The key source data register also sets the output data when the $LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_0/P0Y_0/KS_0$ pins are used as general purpose output port.

In this case, too, the corresponding pins output the port output data if data is set in the key source data register.

For details, see Section **24.4.**

24.2.4 Key Input Control Block and P0D Data Register

The key input control block detects the key input signal entered to the $P0D_3/ADC_5$ thru $P0D_0/ADC_2$ pins in synchronization with the output timing of the key source signal.

When output of the key source signal is required from the $LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_0/P0Y_0/KS_0$ pins, the $P0D_3/ADC_5$ thru $P0D_0/ADC_2$ pins must be used as the key input pins.

Reading of the key input data is performed by the P0D data register (address BANK0, 73H) on the data memory.

The $P0D_3/ADC_5$ thru $P0D_0/ADC_2$ pins serve also as the A/D converter pins. Attention should be paid when using these pins for A/D converter.

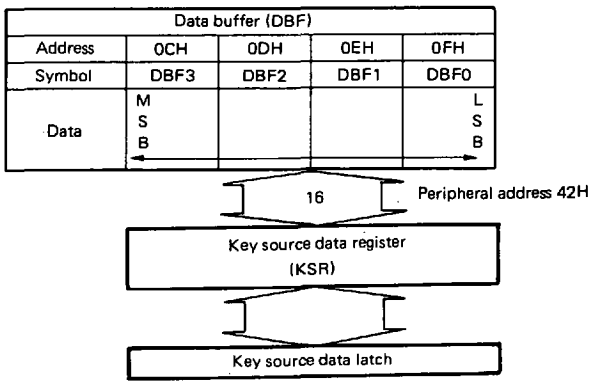
For details, see Section **24.5.**

24.3 KEY SOURCE DATA SETTING BLOCK

24.3.1 Configuration of Key Source Data Setting Block

Fig. 24-3 shows the configuration of the key source data setting block.

Fig. 24-3 Configuration of key source data setting block



24.3.2 Functions of Key Source Data Setting Block

The key source data setting block sets the key source data to be output from $LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_0/P0Y_0/KS_0$ pins.

The key source data is set by the key source data register (KSR: peripheral address 42H) via the data buffer.

Each bit of the key source data register corresponds each of the $LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_0/P0Y_0/KS_0$ pins, and sets the key source data of each pin.

The pins corresponding to the "1" bits of the key source data register output "high" level potential as the key source signal, while the pins corresponding to the "0" bits output "low" level potential.

For the output timing, see Section **24.4.**

When the $LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_0/P0Y_0/KS_0$ pins are used as general-purpose output port, this block sets the output data.

In this case, the data setting register is called the P0Y group data register (P0Y: peripheral address 42H). The peripheral address is not changed, and the register name only is changed.

Section **24.3.3** shows the configuration and function of the key source data register.

Refer also to Fig. 23-6 of Section **23, "LCD Controller/Driver".**

24.3.3 Configuration and Function of Key Source Data Register (KSR)

The configuration and function of the key source data register are shown below.

Name	Data buffer			
Symbol	DBF 3	DBF 2	DBF 1	DBF 0
Address	0 CH	0 DH	0 EH	0 FH
Bit	b_{15} b_{14} b_{13} b_{12}	b_{11} b_{10} b_9 b_8	b_7 b_6 b_5 b_4	b_3 b_2 b_1 b_0
Data	Data to be transferred			

16

GET allowed

PUT allowed

Peripheral register																	
Name	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	Symbol	Peripheral address	Peripheral hardware
Key source data register	Effective data																KSR	42 H	Key source controller/ decoder

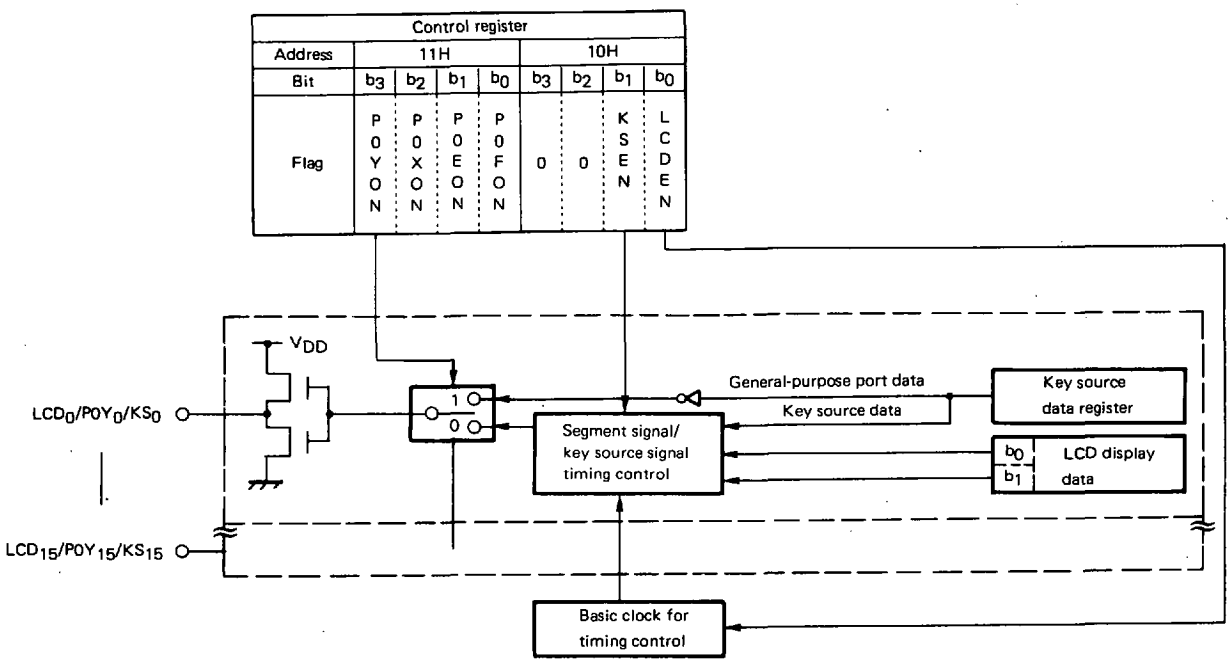
	Key source signal output pins are set
	$LCD_0/P0Y_0/KS_0$ pin
	$LCD_1/P0Y_1/KS_1$ pin
	$LCD_2/P0Y_2/KS_2$ pin
	$LCD_3/P0Y_3/KS_3$ pin
	$LCD_4/P0Y_4/KS_4$ pin
	$LCD_5/P0Y_5/KS_5$ pin
	$LCD_6/P0Y_6/KS_6$ pin
	$LCD_7/P0Y_7/KS_7$ pin
	$LCD_8/P0Y_8/KS_8$ pin
	$LCD_9/P0Y_9/KS_9$ pin
	$LCD_{10}/P0Y_{10}/KS_{10}$ pin
	$LCD_{11}/P0Y_{11}/KS_{11}$ pin
	$LCD_{12}/P0Y_{12}/KS_{12}$ pin
	$LCD_{13}/P0Y_{13}/KS_{13}$ pin
	$LCD_{14}/P0Y_{14}/KS_{14}$ pin
	$LCD_{15}/P0Y_{15}/KS_{15}$ pin
0	Key source signal is not output
1	Key source signal is output

24.4 OUTPUT TIMING CONTROL BLOCK AND SEGMENT/PORT SWITCHING BLOCK

24.4.1 Configration of Output Timing Control Block and Segment/port Switching Block

Fig. 24-4 shows the configuration of the common signal and segment signal/key source signal output timing control block and segment signal/general-purpose output port switching block.

Fig. 24-4 Configuration of timing control block and port switching block



24.4.2 Function of Segment Signal/General-Purpose Output Port Switching Block

The segment signal/general-purpose output port switching block determines whether each pin ($LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_0/P0Y_0/KS_0$ pins) is to be used to output the segment signal or to be used as the general purpose output port (Port 0Y) by the P0YON flag of the LCD port select register.

The pin is set as general purpose output port when P0YON flag is "1".

When outputting the key source signal from the $LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_0/P0Y_0/KS_0$ pins, the P0YON flag must be set "0".

That is, priority is given to port output if Port 0Y is selected.

For the general-purpose output port, see Section **17, "General-Purpose Port".**

24.4.3 Functions of Output Timing Control Block

The output timing control block controls the timing of the key source signal and segment signal.

The LCD segment signal is output when the LCDEN flag of the LCD mode select register is "1".

In this case, all the LCD displays can be turned out by the LCDEN flag. With all the LCD displays turned OFF, the segment signal output is kept "low", and no key source signal is output.

To output the key source signal, the LCDEN flag must be "1".

The key source signal is output when the KSEN flag of the LCD mode select register is "1".

That is, the KSEN flag determines whether or not to use the key source signal.

To output the key source signal, therefore, the P0YON flag must be "0" and also both LCDEN flag and KSEN

flag must be "1".

Section **24.4.4** shows the configuration and function of the LCD mode select register.

Section **24.4.5** shows the output waveform of the key source signal and segment signal.

As for the relationship among the LCD common signal, segment signal and key source signal, see Section **23, "LCD Controller/Decoder".**

24.4.4 Configuration and Function of LCD Mode Select Register

The LCD mode select register controls ON/OFF of the LCD display and sets the output of the key source signal.

The configuration and function of the register are shown below.

Name	Flag b_3	Flag b_2	Flag b_1	Flag b_0	Address	Read/Write
LCD mode select register (LCDMODE)	0	0	KSEN	LCDEN	10H	R/W

b_0 (LCDEN): Sets ON/OFF of all LCD displays
- 0: Display OFF (All the segment output and common output pins are at "low" level.)
- 1: Display ON

b_1 (KSEN): Sets the output of key source signal
- 0: Key source OFF
- 1: Key source ON

b_2: Fixed at "0"

Reset	b_3	b_2	b_1	b_0
Power ON	0	0	0	0
Clock stop			0	0
CE			0	0

24.4.5 Output Waveforms of Segment Signal and Key Source Signal

Fig. 24-5 shows the key source signal and segment signal output waveforms.

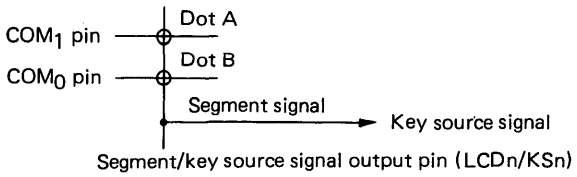
As shown in Fig. 24-5, the key source signal output is time-shared with the segment signal output.

The key source signal is output for 220 μs, and output interval is 4 ms.

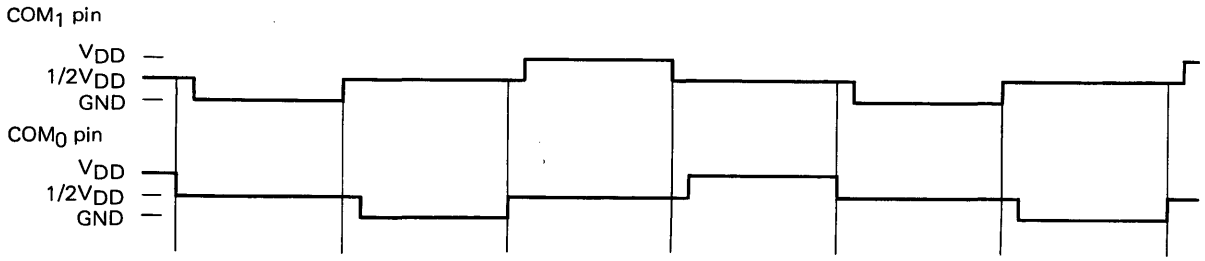
That is, "high" level potential is output for 220 μs at intervals of 4 ms from the pins which correspond to the "1" bits in the key source data register, while "low" level is output for 220 μs at intervals of 4 ms from the pins which correspond to the "0" bits of the key data source register.

When key source signal output is specified (KSEN flag = 1), the pins (LCD_{29}/$P0Y_3$ thru LCD_{16}/$P0X_0$) which do not issue key source also issue the waveform as shown in Fig. 24-5. In this case, the key source data "0" waveform is output.

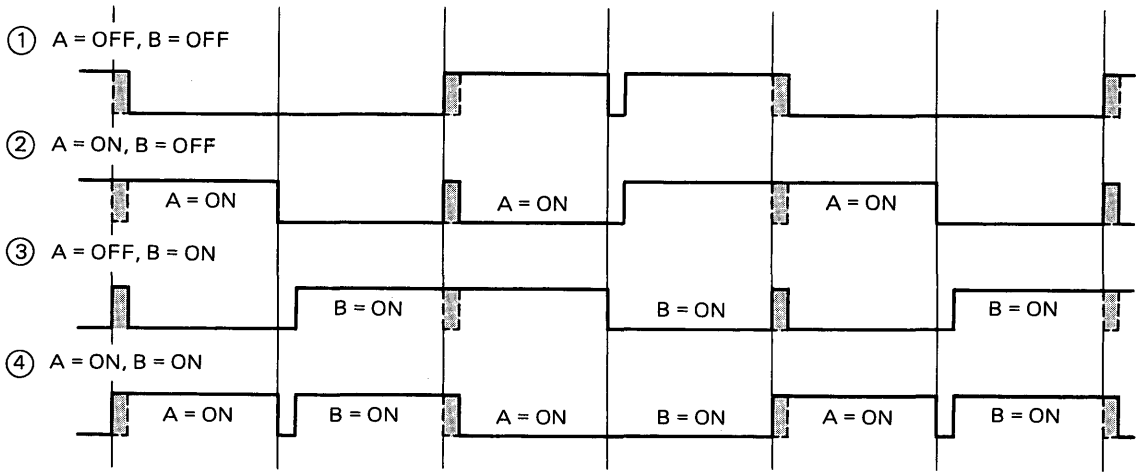
Fig. 24-5 Output waveforms of key source signal and segment signal



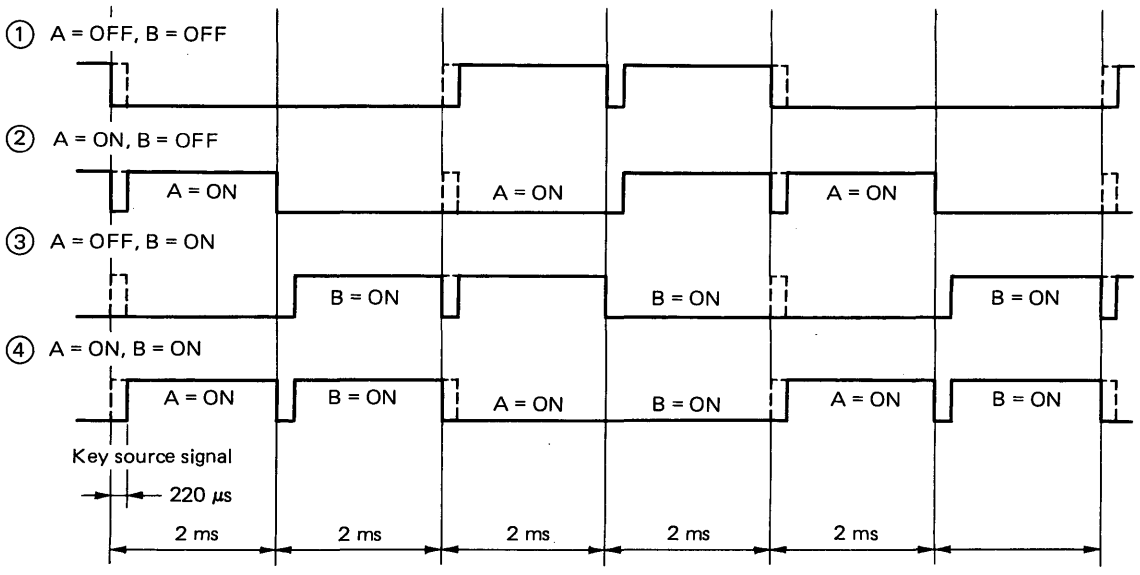
Common signal



Each segment pin (with "1" output to key source)



Each segment pin (with "0" output to key source)

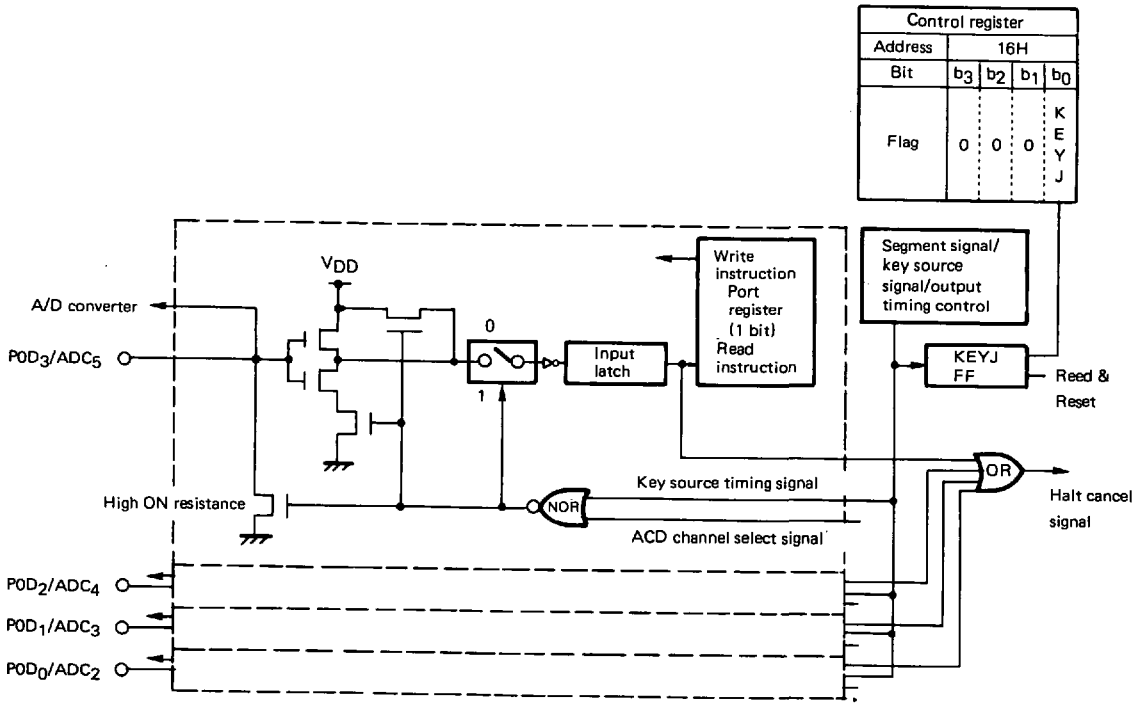


24.5 KEY INPUT CONTROL BLOCK

24.5.1 Configuration of Key Input Control Block

Fig. 24-6 shows the configuration of the key input control block.

Fig. 24-6 Configuration of key input control block



24.5.2 Functions of Key Input Control Block

The key input control block controls the timing for reading the key input signal from the $P0D_3/ADC_5$ thru $P0D_0/ADC_2$ pin, and also performs reading of the key input data.

Fig. 24-7 shows the key source signal and key input timing.

As shown in Fig. 24-7, the pull-down resistance built-in each of the $P0D_3/ADC_5$ thru $P0D_0/ADC_2$ pins is OFF while the display data is being output from the LCD segment, and is ON only during the 220 μs period where the key source signal is being output.

During this 220 μs of key source signal output period, the input signal of each key input pin is connected to the input latch.

Accordingly, the signal input to each key input pin can be detected during this 220 μs key source signal output period.

Fig. 24-8 shows the timing chart of the key source signal, key input signal and key input data (P0D data register).

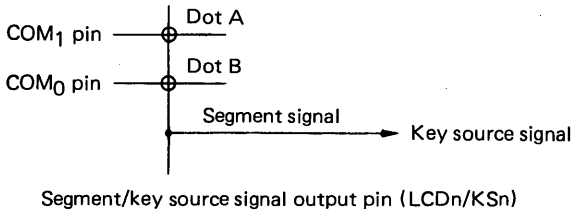
Whether or not the key source signal is output is detected by the KEYJ flag of the key input judge register (address 16H).

The KEYJ flag is set after the output of the key source signal for 220 μs. It is reset when data is entered into the key source data register and when the content of KEYJ flag is read.

Accordingly, the KEYJ flag is detected after outputting the key source signal data to the key source data register, and key input can be fetched by detecting the status of each key input pin after the KEYJ flag turned to "1".

Section **24.5.3** shows the configuration and function of the key input judge register.

Fig. 24-7 Key source signal and key input timing



Segment pin (with "1" output to key source; A=ON, B=OFF)

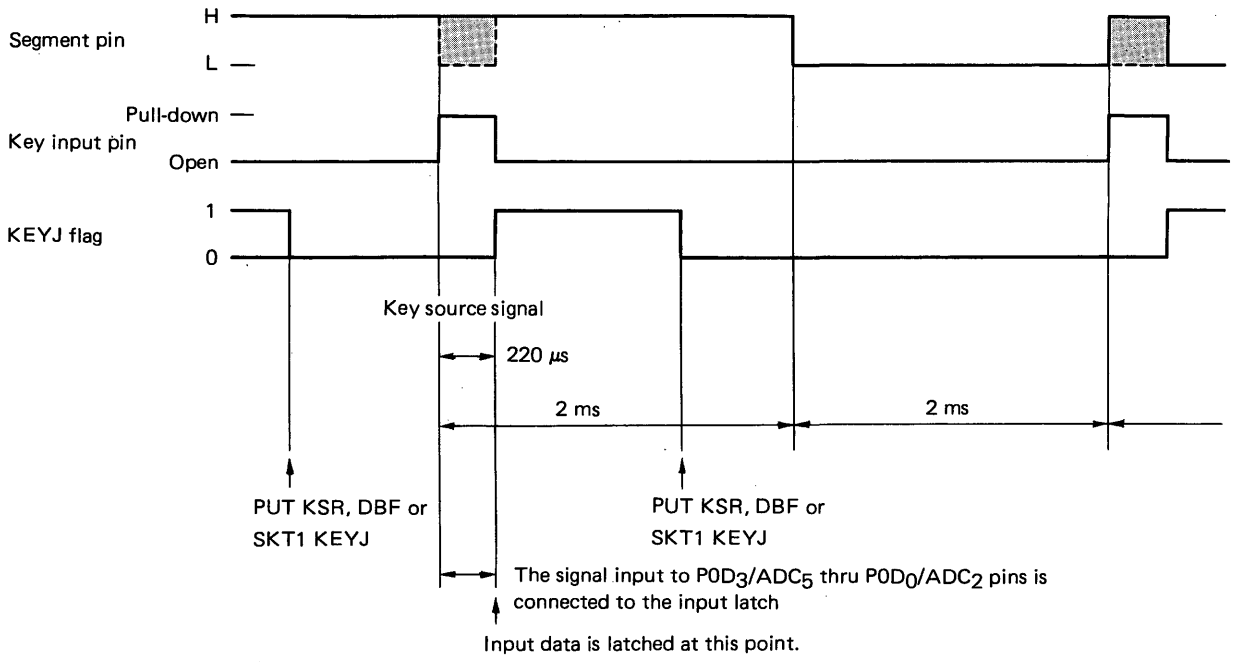
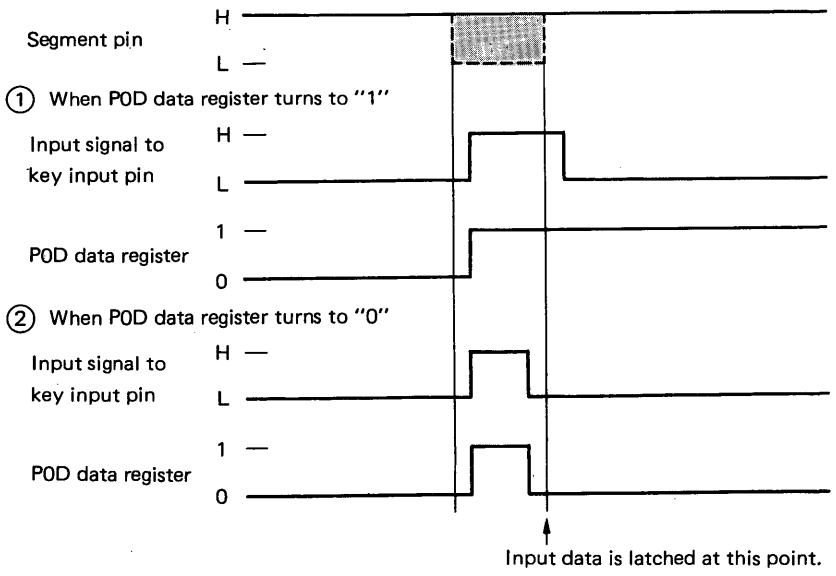


Fig. 24-8 Timing chart for the key source signal, key input signal and key input data (POD data register)



24.5.3 Configuration and Function of Key Input Judge Register

The key input judge register detects whether the key input is latched or not when the LCD segment signal output pin is used as the key source signal output pin.

The configuration and function of the register are shown below.

Name	Flag b_3	Flag b_2	Flag b_1	Flag b_0	Address	Read/Write
Key input judge register (KEYJDG)	0	0	0	KEYJ	16H	R & Reset

Detects whether key input signal is latched	
0	Key latch not provided
1	Key latch provided

Fixed at "0"

Reset	b_3	b_2	b_1	b_0
Power ON	0	0	0	0
Clock stop				0
CE				0

The key source signal data is set by putting the content of the data buffer into the key source data register by "PUT" instruction.

When the key source signal output data is set into the register via the data buffer by the "PUT" instruction, the KEYJ flag is set (turned to "0").

The KEYJS flag is reset (turned to "0") when the data is read into the window register by "PEEK" instruction. (Read & Reset)

24.6 USAGE OF KEY SOURCE CONTROLLER/DECODER

24.6.1 Key Matrix Composition Method

Fig. 24-9 shows an example composition of the key matrix.

As shown in Fig. 24-9, the key matrix permits use of maximum 64 keys.

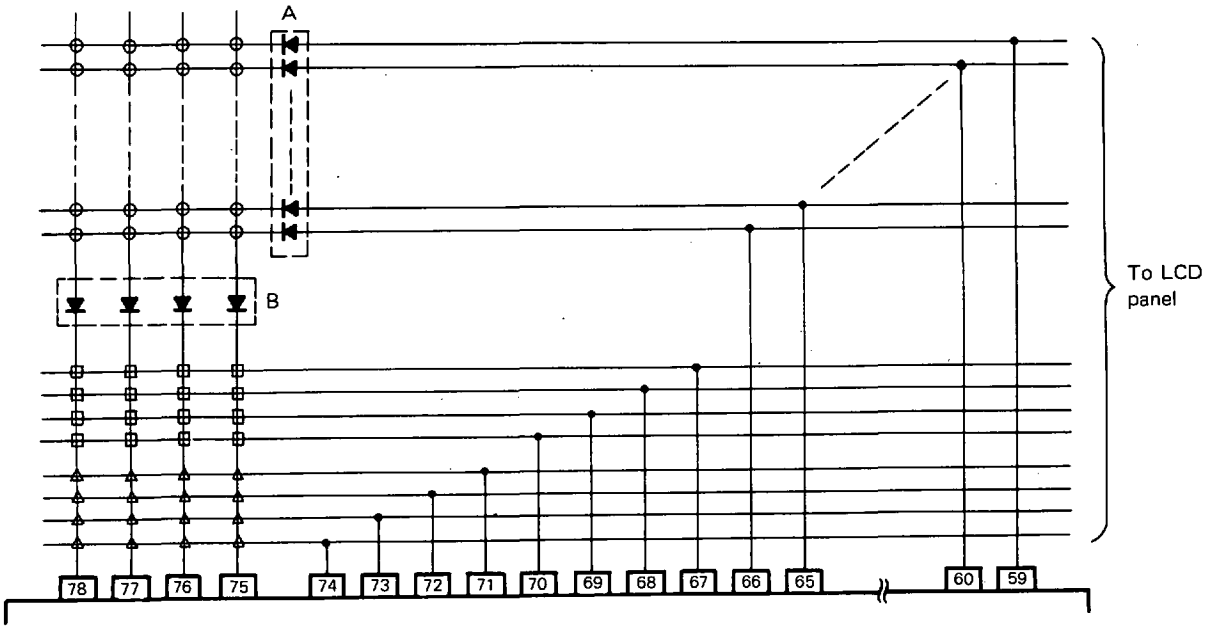
The key source signal output pin issues the LCD segment signal at the same time, and when using the momentary switch, it is necessary to use diode "A" for preventing reverse flow of the LCD segment signal.

Diodes "B" and "C" are used to prevent the key source signal to flow through a sneak path.

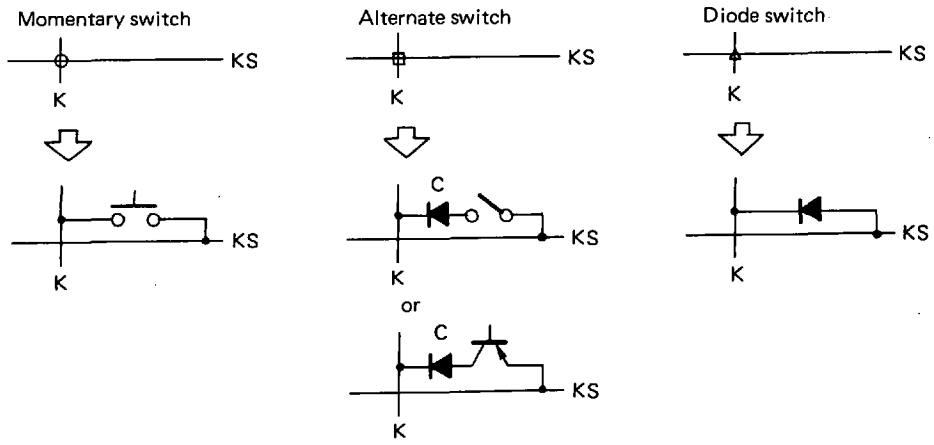
The transistor switch should be a PNP type transistor.

The following Item (1) explains precautions for using the NPN type transistor.

Items (2) thru (4) explains the precautions to be observed when there is no diode A, B, or C.

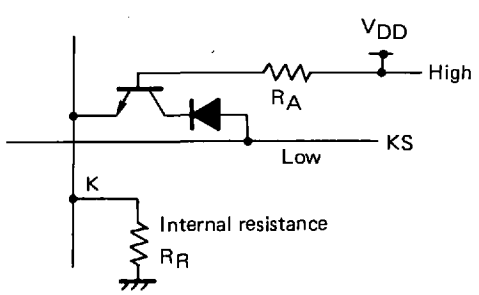


Composition of each switch



(1) Precautions when using NPN type transistor switch

When using NPN-type transistors to compose a transistor switch, attention should be taken because correct reading of the "low" level signal may fail in the following case.



If "high" level potential is being applied to the transistor base while KS is kept at "low" level in the circuit as shown at left, the voltage VK is applied to K.

$$V_K = \frac{R_B}{R_A + R_B} \times (V_{DD} - V_{EE})$$

In this case, KS is kept "low", and hence the "low" level potential is applied to K.

But, the voltage applied to K varies with the value of R_A and R_B as shown in the above-mentioned equation.

This means that the "low" level may not be entered depending on the value of R_A and R_B.

(2) Precautions when there is no diode A

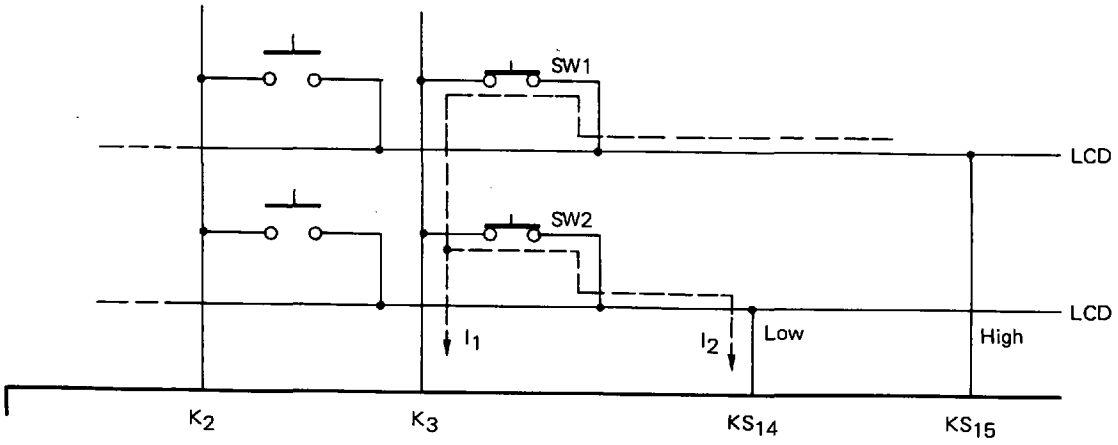
An example of circuit without diode A is shown below.

As shown, switch SW1 and SW2 are ON, and KS_{15} issues "high" level potential while KS_{14} issues "low" level.

If diode A is not provided, current I_1 and I_2 will flow as shown through the paths indicated by dotted lines.

This current I_2 disturbs the "high" level potential at KS_{15} and the "low" level potential at KS_{14}, hence the key input data of K_3 is read incorrectly.

Besides, if KS_{15} and KS_{14} are used as the LCD segment signal output, this sneak current also disturbs normal ON and OFF of the LCD display.



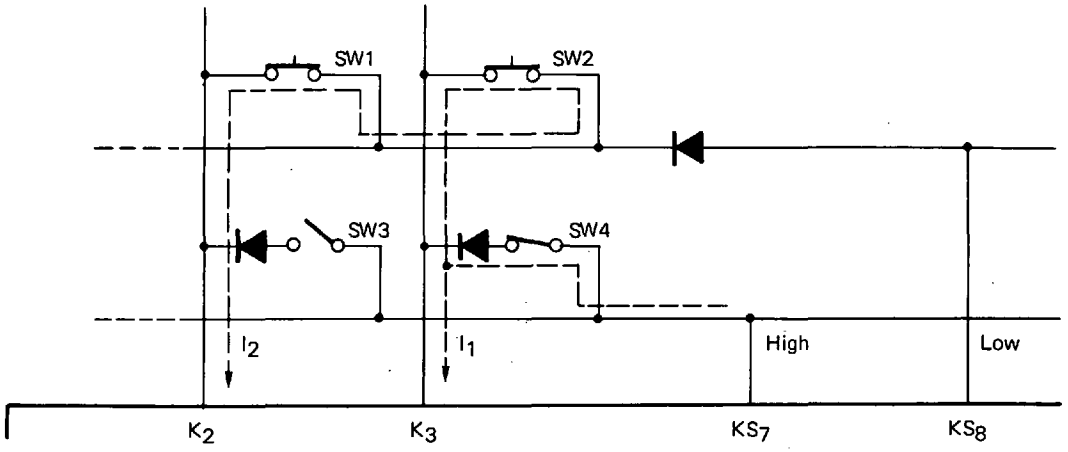
(3) Precautions when there is no diode B

Shown below is an example of circuit having no diode B.

Assume that switches SW1, SW2 and SW4 are ON, and "high" level potential is being issued from KS_7.

If there is no diode B, the current I_1 and I_2 will flow through the path as indicated by dotted lines.

This sneak current causes the status of SW3 to be erroneously regarded as "ON" due to application of "high" level input though this switch SW3 is actually kept "OFF".



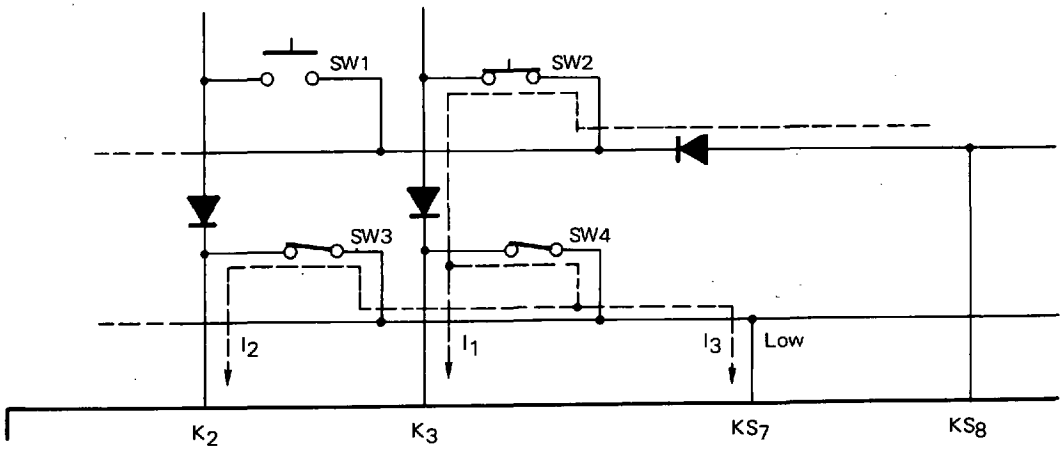
(4) Precautions when there is no diode C

Shown below is an example of circuit without diode C. Assume that switches SW2, SW3 and SW4 are ON, and "high" level potential is being output from KS_8.

If diode C is not provided, the current I_1, I_2 and I_3 will flow through the path indicated by dotted lines.

This sneak current causes the status of SW1 to be erroneously regarded as "ON" due to the application of "high" level input though this switch SW1 is actually kept "OFF".

Besides, the current I_3 prevents the "high" level potential from being output normally from KS_8.



24.6.2 Fetching Key Input Data of Single Key Source Line

An example of program is shown below.

Example:

To read the alternate switch of $LCD_{15}/P0Y_{15}/KS_{15}$ thru $LCD_8/P0Y_8/KS_8$ pins into data memory location M00-M07

```
INIT:
    M00    MEM  0.00H      ;
    P0D    MEM  0.73H      ;  Port 0D data register
    CLR1   P0YON           ;  LCD15/P0Y15/KS15 thru LCD0/P0Y0/KS0 pins are set in the LCD
                           ;  segment
    SET2   KSEN, LCDEN     ;  All the LCD displays are turned ON, and use of the key source signal
                           ;  is set.
    MOV    DBF3, #0000B
    MOV    DBF2, #0001B
    MOV    DBF1, #0000B
    MOV    DBF0, #0000B
    MOV    IXH,  #0000B
    MOV    IXM,  #0000B
    MOV    IXL,  #0000B
    MOV    RPH,  #0000B
    MOV    RPL,  #0000B
KSCAN:
    PUT    KSR,  DBF       ;
LOOP:
    SKF1   KEYJ            ;
    BR     KCHECK
    ┌──────────┐
    │Processing A│         ;  Processing is waited until key input is
    │BR  LOOP  │           ;  latched (4 ms).
    └──────────┘
KCHECK:
    MOV    RPL,  #1110B
    SET1   IXE
    ST     M00,  P0D
    CLR1   IXE
    MOV    RPL,  #0000B
    INC    IX
    ADD    DBF2, DBF2      ;  The value of key source data is updated, and
    ADDC   DBF3, DBF3      ;  key scan is performed again.
    SKT1   CY              ;
    BR     KSCAN
END:
```

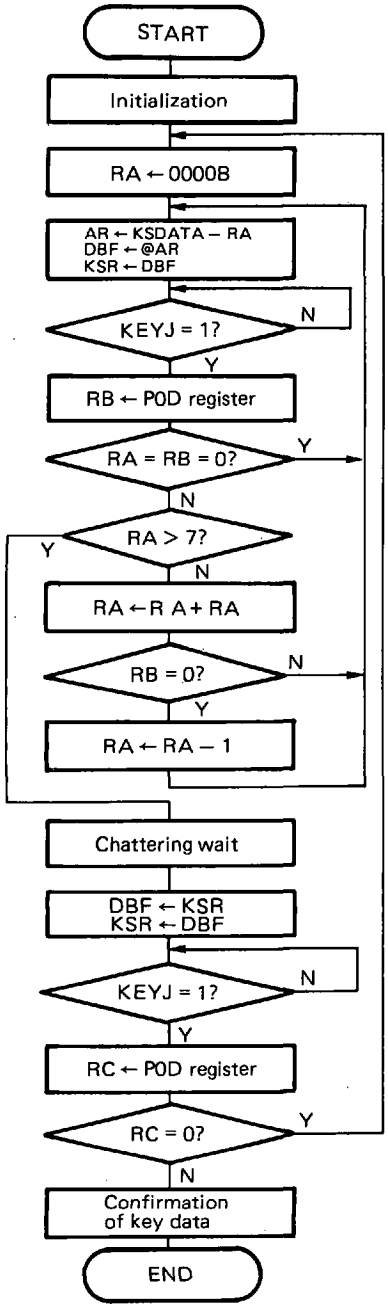
24.6.3 How to Fetch Momentary Switch by Binary Search

The key source controller/decoder requires 4 ms to fetch the key data of single key source signal line.

That is, fetching the key data of 16 key source signals requires 64 ms. Accordingly, use of the binary search method as shown in the following Items (1) and (2) is recommended.

(1) Flowchart

When KS_7-KS_0 are used as key source signal of momentary switch



: Setting of key source controller

: Offset address of the table containing key source data is set at RA.

: Key source data of offset address specified by RA is output.

: Processing is waited until data is latched by key input latch. (4 ms)

: Key input data is saved to RB.

: IF key input data and RA are "0", no key input is provided, hence all key data are fetched again.

: If RA is greater than "7", program enters the chattering wait after the data for one key source has been fetched.

: If RA is smaller than "7", program updates RA and continues the binary search.

: 4 ms chattering can be performed even if this chattering wait is not provided.

: Program enters the key input data defined by the binary search again into RC for confirmation.

: If RC = 0, program regards the state as chattering, and fetches data from the beginning.

: Key input data before chattering is stored into RB, data after chattering is stored into RC, and key source data is stored into RA.

Example of table data for binary search

Shift address (RA)	Table data (key source data)															
	b_{15}	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
0000B									■	■	■	■	■	■	■	■
0001B									■	■	■	■				
0010B													■	■		
0011B											■	■				
0100B															■	
0101B													■			
0110B											■					
0111B									■							
1000B																■
1001B															■	
1010B														■		
1011B													■			
1100B												■				
1101B											■					
1110B										■						
1111B									■							

(2) Program example

```
KSDATA:
        DW   0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 B
        DW   0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 B
        DW   0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 B
        DW   0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 B
        DW   0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 B
        DW   0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 B
        DW   0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 B
        DW   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 B
        DW   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 B
        DW   0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 B
        DW   0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 B
        DW   0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 B
        DW   0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 B
        DW   0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 B
        DW   0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 B
        DW   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 B
INIT:
        RA      MEM   0.0AH     ;  General-purpose work register
        RB      MEM   0.0BH     ;  General-purpose work register
        RC      MEM   0.0CH     ;  General-purpose work register
        P0D     MEM   0.73H     ;  Port 0D data register
        CLR1    P0YON           ;  Sets LCD15/P0Y15/KS15 thru LCD0/P0Y0/KS0 pins in LCD segment.
        SET2    KSEN, LCDEN     ;  Turns ON all the LCD displays, and sets the use of key source signal.
START:
        MOV     RA,   #000B     ;
```

```
KSCAN:
        MOV     AR3,    #.DL. (KADATA SHR 12) AND 000FH
        MOV     AR2,    #.DL. (KADATA SHR 8)  AND 000FH
        MOV     AR1,    #.DL. (KADATA SHR 4)  AND 000FH
        MOV     AR0,    #.DL. (KADATA SHR 0)  AND 000FH
        MOV     RPL,    #1110B
        ADD     AR0,    RA
        ADDC    AR1,    #0
        ADDC    AR2,    #0
        ADDC    AR3,    #0
        MOVT    DBF,    @AR
        PUT     KSR,    DBF
LOOP1:
        SKF1    KEYJ
        BR      KCHECK
        ┌──────────┐
        │Processing A│        ; 4 ms
        │BR  LOOP1 │
        └──────────┘
KCHECK:
        MOV     RPL,    #0000B
        LD      RB,     P0D
    SKNE    RA,     #0000B ; If there is no key input,
    SKE     RB,     #0000B ;
    BR      $+2
    BR      START          ; Branch to START.
    SKLT    RA,     #1000B ; After completing key scanning for one key source,
    BR      LASTCHK        ; Branch to LASTCHK
    ADD     RA,     RA     ; Updates the value of RA and performs key scanning again.
    SKNE    RB,     0000B
    ADD     RA,     #0001B
    BR      KSCAN
LASTCHK:
    SKNE    RB,     #0000B ; If there is no key input after terminating key scanning for one key source,
    BR      START          ; program regards this status as chattering, and executes the processing
    ┌──────────────┐
    │Chattering wait│      ; again from the beginning.
    └──────────────┘
    GET     DBF,    KSR    ; Fetches the key input after placing a wait of chattering time.
    PUT,    KSR,    DBF
LOOP2:
    SKF1    KEYJ           ;
    BR      KEYDEC
    ┌──────────┐
    │Processing B│         ; 4 ms
    │BR  LOOP2 │
    └──────────┘
KEYDEC:
    LD      RC,     P0D    ; If there is no key input after chattering wait,
    SKNE    RC,     #0000B ;
    BR      START          ; Program executes again the process from the beginning.
END:                       ; At this point, key source data is stored into RA, and key input data is
                           stored to RB and RC.
```

24.7 RESET STATUS

24.7.1 Power ON

$LCD_{29}/P0F_3$ thru $LCD_0/P0Y_0/KS_0$ pins are specified for outputting the LCD segment signal, and issue "low" level potential (display OFF), hence the key source signal is issued at "low" level.

24.7.2 Clock Stop

$LCD_{29}/P0F_3$ thru $LCD_0/P0Y_0/KS_0$ pins are specified for outputting the LCD segment signal, and issue "low" level potential (display OFF), hence the key source signal is issued at "low" level.

24.7.3 CE Resetting

If the key source signal is being issued, the current output data is held.

24.7.4 Halt Status

If the key source signal is being issued, the current output data is held.

If key input is specified as the halt cancel condition, the halt status is canceled when "high" level is applied to $P0D_3/ADC_5$ thru $P0D_0/ADC_2$ pins.

When the key controller is in use, the halt status can be canceled only by the "high" level entered within the period of 220 μs where the key source data is being output.

When canceling the halt status by the key input using the key source controller, do not use the $P0D_3/ADC_5$ thru $P0D_0/ADC_2$ pins for A/D converter.

For the canceling method of the halt status by key input, see Section **14.4 "Halt Function"**.

25. μPD17005 INSTRUCTIONS

25.1 INSTRUCTION SET

b_{15} / b_{14} b_{13} b_{12} b_{11}		0		1	
0 0 0 0	0	ADD	r, m	ADD	m, #i
0 0 0 1	1	SUB	r, m	SUB	m, #i
0 0 1 0	2	ADDC	r, m	ADDC	m, #i
0 0 1 1	3	SUBC	r, m	SUBC	m, #i
0 1 0 0	4	AND	r, m	AND	m, #i
0 1 0 1	5	XOR	r, m	XOR	m, #i
0 1 1 0	6	OR	r, m	OR	m, #i
0 1 1 1	7	INC INC MOVT BR CALL RET RETSK EI DI RETI PUSH POP GET PUT PEEK POKE RORC STOP HALT NOP	AR IX DBF, @AR @AR @AR AR AR DBF, p p, DBF WR, rf rf, WR r 0 h		
1 0 0 0	8	LD	r, m	ST	m, r
1 0 0 1	9	SKE	m, #i	SKGE	m, #i
1 0 1 0	A	MOV	@r, m	MOV	m, @r
1 0 1 1	B	SKNE	m, #i	SKLT	m, #i
1 1 0 0	C	BR	addr (Page 0)	CALL	addr (Page 0)
1 1 0 1	D	BR	addr (Page 1)	MOV	m, #i
1 1 1 0	E	BR	addr (Page 2)	SKT	m, #n
1 1 1 1	F	BR	addr (Page 3)	SKF	m, #n

25.2 LIST OF INSTRUCTIONS

Legends:

M	:	Data memory address
m	:	Data memory address excluding bank
m_H	:	Data memory row address
m_L	:	Data memory column address
R	:	General register address
r	:	General register column address
RP	:	General register pointer
RF	:	Register file
rf	:	Register file address
rf_H	:	Register file address (upper 3 bits)
rf_L	:	Register file address (lower 3 bits)
AR	:	Address register
IX	:	Index register
IXE	:	Index enable flag
DBF	:	Data buffer
WR	:	Window register
MP	:	Data memory row address pointer
MPE	:	Memory pointer enable flag
PE	:	Peripheral register
p	:	Peripheral address
p_H	:	Peripheral address (upper 3 bits)
p_L	:	Peripheral address (lower 4 bits)
PC	:	Program memory counter
SP	:	Stack pointer
STACK	:	Stack value indicated by stack pointer
$STACK_{PC}$	:	Program counter value indicated by stack pointer
BANK	:	Bank register
$(ROM)_{PC}$	:	Program memory data indicated by program memory counter
INTEF	:	Interrupt enable flag
SGR	:	Program memory segment register
i	:	Immediate data (4 bigs)
n	:	Bit position (4 bits)
addr	:	Program memory address (11 bits)
CY	:	Carry flag
c	:	Carry
b	:	Borrow
h	:	Halt canceling condition
[ ]	:	Data memory or register address
( )	:	Data memory or register value

Instruction group	Mnemonic	Operand	Operation	Machine code			
				Operation code			
Add instruction	ADD	r, m	(R) ← (R) + (M)	00000	m_H	m_L	r
		m, #i	(M) ← (M) + i	10000	m_H	m_L	i
	ADDC	r, m	(R) ← (R) + (M) + (CY)	00010	m_H	m_L	r
		m, #i	(M) ← (M) + i + (CY)	10010	m_H	m_L	i
	INC	AR	(AR) ← (AR) + 1	00111	000	1001	0000
		IX	(IX) ← (IX) + 1	00111	000	1000	0000
Subtract instruction	SUB	r, m	(R) ← (R) − (M)	00001	m_H	m_L	r
		m, #i	(M) ← (M) − i	10001	m_H	m_L	i
	SUBC	r, m	(R) ← (R) − (M) − (CY)	00011	m_H	m_L	r
		m, #i	(M) ← (M) − i − (CY)	10011	m_H	m_L	i
Compare instruction	SKE	m, #i	(M) − i, skip if zero	01001	m_H	m_L	i
	SKGE	m, #i	(M) − i, skip if not borrow	11001	m_H	m_L	i
	SKLT	m, #i	(M) − i, skip if borrow	11011	m_H	m_L	i
	SKNE	m, #i	(M) − i, skip if not zero	01011	m_H	m_L	i
Logical instruction	AND	m, #i	(M) ← (M) AND i	10100	m_H	m_L	i
		r, m	(R) ← (R) AND (M)	00100	m_H	m_L	r
	OR	m, #i	(M) ← (M) OR i	10110	m_H	m_L	i
		r, m	(R) ← (R) OR (M)	00110	m_H	m_L	r
	XOR	m, #i	(M) ← (M) XOR i	10101	m_H	m_L	i
		r, m	(R) ← (R) XOR (M)	00101	m_H	m_L	r
Transfer instruction	LD	r, m	(R) ← (M)	01000	m_H	m_L	r
	ST	m, r	(M) ← (R)	11000	m_H	m_L	r
	MOV	@r, m	if MPE= 1 : [(MP), (R)] ← (M) if MPE= 0 : [(m_H), (R)] ← (M)	01010	m_H	m_L	r
		m, @r	if MPE= 1 : (M) ← [(MP), (R)] if MPE= 0 : (M) ← [(m_H), (R)]	11010	m_H	m_L	r
		m, #i	(M) ← i	11101	m_H	m_L	i
	MOVT	DBF, @AR	($STACK_{PC}$) ← (PC), (PC) ← (AR) (DBF) ← $(ROM)_{PC}$, (PC) ← ($STACK_{PC}$)	00111	000	0001	0000
	PUSH	AR	(SP) ← (SP) − 1, ($STACK_{PC}$) ← (AR)	00111	000	1101	0000
	POP	AR	(AR) ← ($STACK_{PC}$), (SP) ← (SP) + 1	00111	000	1100	0000
	PEEK	WR, rf	(WR) ← (RF)	00111	rf_H	0011	rf_L
	POKE	rf, WR	(RF) ← (WR)	00111	rf_H	0010	rf_L
	GET	DBF, p	(DBF) ← (PE)	00111	p_H	1011	p_L
	PUT	p, DBF	(PE) ← (DBF)	00111	p_H	1010	p_L
Decision instruction	SKT	m, #n	if $(M)_n$= all "1", then skip	11110	m_H	m_L	n
	SKF	m, #n	if $(M)_n$= all "0", then skip	11111	m_H	m_L	n

Instruction group	Mnemonic	Operand	Operation	Machine code			
				Operation code			
Branch instruction	BR	addr	(PC) ← addr, (PC)$_{=12, \neq 11}$ ← 00	01100	addr (11 bits)		
			(PC) ← addr, (PC)$_{=12, \neq 11}$ ← 01	01101			
			(PC) ← addr, (PC)$_{=12, \neq 11}$ ← 10	01110			
			(PC) ← addr, (PC)$_{=12, \neq 11}$ ← 11	01111			
		@AR	(PC) ← (AR)	00111	000	0100	0000
Shift	RORC	r	(CY) ← (R)$_{b3}$ → (R)$_{b2}$ → (R)$_{b1}$ → (R)$_{b0}$ → (CY)	00111	000	0111	r
Subroutine instruction	CALL	addr	(SP) ← (SP) − 1, (STACK$_{PC}$) ← ((PC) + 1), (PC)$_{\neq 11}$ ← 0, (PC) ← addr	11100	addr (11 bits)		
		@AR	(SP) ← (SP) − 1, (STACK$_{PC}$) ← ((PC) + 1), (PC) ← (AR)	00111	000	0101	0000
	RET		(PC) ← (STACK$_{PC}$), (SP) ← (SP) + 1	00111	000	1110	0000
	RETSK		(PC) ← (STACK$_{PC}$), (SP) ← (SP) + 1, skip	00111	001	1110	0000
	RETI		(PC), (BANK), (IXE) ← (STACK), (SP) ← (SP) + 1	00111	100	1110	0000
Interrupt	EI		INTEF ← 1	00111	000	1111	0000
	DI		INTEF ← 0	00111	001	1111	0000
Others	STOP	0	stop clock if CE = low	00111	010	1111	0000
	HALT	h	halt	00111	011	1111	h
	NOP		No operation	00111	100	1111	0000

25.3 ASSEMBLER (AS17K) BUILT-IN MACRO INSTRUCTION

Legend

flag : One of flagl-flagn
flagl-flagn : Flag names indicated by reserved words
n : Number
< > : Omission allowed

	Mnemonic	Operand	n	Operation
	SKTn	flag1, ···flagn	1 ≦n≦ 4	if (flag1) − (flagn) = all " 1 ", then skip
	SKFn	flag1, ···flagn	1 ≦n≦ 4	if (flag1) − (flagn) = all " 0 ", then skip
	SETn	flag1, ···flagn	1 ≦n≦ 4	(flag1) − (flagn) ← 1
	CLRn	flag1, ···flagn	1 ≦n≦ 4	(flag1) − (flagn) ← 0
	NOTn	flag1, ···flagn	1 ≦n≦ 4	if (flag) = " 0 ", then (flag) ← 1 , if (flag) = " 1 ", then (flag) ← 0
	INITFLG	<NOT>flag1, ···<NOT>flagn	n= 4	if description = NOT flag, (flag) ← 0 if description = flag, (flag) ← 1
	BANKn		0 ≦n≦ 3	(BANK) ← n

26. μPD17005 RESERVED WORDS

26.1 LIST OF RESERVED WORDS

26.1.1 System Register (SYSREG)

Reserved word	Type	Address	Read/ Write	Function
AR3	MEM	0.74H	R/W	Address register bit b_{15}-b_{12}
AR2	MEM	0.75H	R/W	Address register bit b_{11}-b_8
AR1	MEM	0.76H	R/W	Address register bit b_7-b_4
AR0	MEM	0.77H	R/W	Address register bit b_3-b_0
WR	MEM	0.78H	R/W	Window register
BANK	MEM	0.79H	R/W	Bank register
IXH	MEM	0.7AH	R/W	Index register high
MPH	MEM	0.7AH	R/W	Memory pointer high
MPE	FLG	0.7AH.3	R/W	Memory pointer enable flag
IXM	MEM	0.7BH	R/W	Index register middle
MPL	MEM	0.7BH	R/W	Memory pointer low
IXL	MEM	0.7CH	R/W	Index register low
RPH	MEM	0.7DH	R/W	General register pointer high
RPL	MEM	0.7EH	R/W	General register pointer low
PSW	MEM	0.7FH	R/W	Program status word
BCD	FLG	0.7EH.0	R/W	BCD flag
CMP	FLG	0.7FH.3	R/W	Compare flag
CY	FLG	0.7FH.2	R/W	Carry flag
Z	FLG	0.7FH.1	R/W	Zero flag
IXE	FLG	0.7FH.0	R/W	Index enable flag

26.1.2 Data Buffer (DBF)

Reserved word	Type	Address	Read/ Write	Function
DBF3	MEM	0.0CH	R/W	DBF bit b_{15}-b_{12}
DBF2	MEM	0.0DH	R/W	DBF bit b_{11}-b_8
DBF1	MEM	0.0EH	R/W	DBF bit b_7-b_4
DBF0	MEM	0.0FH	R/W	DBF bit b_3-b_0

26.1.3 LCD Dot Data Register

Reserved word	Type	Address	Read/ Write	Function
LCDD0	MEM	0.60H	R/W	LCD data register
LCDD1	MEM	0.61H	R/W	LCD data register
LCDD2	MEM	0.62H	R/W	LCD data register
LCDD3	MEM	0.63H	R/W	LCD data register
LCDD4	MEM	0.64H	R/W	LCD data register
LCDD5	MEM	0.65H	R/W	LCD data register
LCDD6	MEM	0.66H	R/W	LCD data register
LCDD7	MEM	0.67H	R/W	LCD data register
LCDD8	MEM	0.68H	R/W	LCD data register
LCDD9	MEM	0.69H	R/W	LCD data register
LCDD10	MEM	0.6AH	R/W	LCD data register
LCDD11	MEM	0.6BH	R/W	LCD data register
LCDD12	MEM	0.6CH	R/W	LCD data register
LCDD13	MEM	0.6DH	R/W	LCD data register
LCDD14	MEM	0.6EH	R/W	LCD data register

26.1.4 General Port Register

Reserved word	Type	Address	Read/ Write	Function
P0A3	FLG	0.70H.3	R/W	Port 0A bit b_3
P0A2	FLG	0.70H.2	R/W	Port 0A bit b_2
P0A1	FLG	0.70H.1	R/W	Port 0A bit b_1
P0A0	FLG	0.70H.0	R/W	Port 0A bit b_0
P0B3	FLG	0.71H.3	R/W	Port 0B bit b_3
P0B2	FLG	0.71H.2	R/W	Port 0B bit b_2
P0B1	FLG	0.71H.1	R/W	Port 0B bit b_1
P0B0	FLG	0.71H.0	R/W	Port 0B bit b_0
P0C3	FLG	0.72H.3	R/W	Port 0C bit b_3
P0C2	FLG	0.72H.2	R/W	Port 0C bit b_2
P0C1	FLG	0.72H.1	R/W	Port 0C bit b_1
P0C0	FLG	0.72H.0	R/W	Port 0C bit b_0
P0D3	FLG	0.73H.3	R	Port 0D bit b_3
P0D2	FLG	0.73H.2	R	Port 0D bit b_2
P0D1	FLG	0.73H.1	R	Port 0D bit b_1
P0D0	FLG	0.73H.0	R	Port 0D bit b_0
P0XL3	FLG	0.68H.3	R/W	Port 0X bit b_1
P0XL2	FLG	0.68H.2	R/W	Port 0X bit b_0
P0XL1	FLG	0.68H.1	R/W	Dummy
P0XL0	FLG	0.68H.0	R/W	Dummy
P0XH3	FLG	0.69H.3	R/W	Port 0X bit b_5
P0XH2	FLG	0.69H.2	R/W	Port 0X bit b_4
P0XH1	FLG	0.69H.1	R/W	Port 0X bit b_3
P0XH0	FLG	0.69H.0	R/W	Port 0X bit b_2
P0E3	FLG	0.6BH.3	R/W	Port 0E bit b_3
P0E2	FLG	0.6BH.2	R/W	Port 0E bit b_2
P0E1	FLG	0.6BH.1	R/W	Port 0E bit b_1
P0E0	FLG	0.6BH.0	R/W	Port 0E bit b_0
P0F3	FLG	0.6DH.3	R/W	Port 0F bit b_3
P0F2	FLG	0.6DH.2	R/W	Port 0F bit b_2
P0F1	FLG	0.6DH.1	R/W	Port 0F bit b_1
P0F0	FLG	0.6DH.0	R/W	Port 0F bit b_0
P1A3	FLG	1.70H.3	R/W	Port 1A bit b_3
P1A2	FLG	1.70H.2	R/W	Port 1A bit b_2
P1A1	FLG	1.70H.1	R/W	Port 1A bit b_1
P1A0	FLG	1.70H.0	R/W	Port 1A bit b_0

Reserved word	Type	Address	Read/ Write	Function
P1B3	FLG	1.71H.3	R/W	Port 1B bit b_3
P1B2	FLG	1.71H.2	R/W	Port 1B bit b_2
P1B1	FLG	1.71H.1	R/W	Port 1B bit b_1
P1B0	FLG	1.71H.0	R/W	Port 1B bit b_0
P1C3	FLG	1.72H.3	R/W	Port 1C bit b_3
P1C2	FLG	1.72H.2	R/W	Port 1C bit b_2
P1C1	FLG	1.72H.1	R/W	Port 1C bit b_1
P1C0	FLG	1.72H.0	R/W	Port 1C bit b_0
P1D3	FLG	1.73H.3	R/W	Port 1D bit b_3
P1D2	FLG	1.73H.2	R/W	Port 1D bit b_2
P1D1	FLG	1.73H.1	R/W	Port 1D bit b_1
P1D0	FLG	1.73H.0	R/W	Port 1D bit b_0
P2A3	FLG	2.70H.3	R/W	Port 2A bit b_3
P2A2	FLG	2.70H.2	R/W	Port 2A bit b_2
P2A1	FLG	2.70H.1	R/W	Port 2A bit b_1
P2A0	FLG	2.70H.0	R/W	Port 2A bit b_0

26.1.5 Register File (Control Register)

Reserved word	Type	Address	Read/ Write	Function
SP	MEM	0.81H	R/W	Stack pointer
SIO2TS	FLG	0.82H.3	R/W	SIO_2 start flag
SIO2HIZ	FLG	0.82H.2	R/W	SO_2/$P0B_1$ select flag
SIO2CK1	FLG	0.82H.1	R/W	SIO_2 clock select bit b_1
SIO2CK0	FLG	0.82H.0	R/W	SIO_2 clock select bit b_0
IFCG	FLG	0.84H.0	R	IF counter gate status flag
PLLUL	FLG	0.85H.0	R	PLL unlock FF flag
ADCCMP	FLG	0.86H.0	R	ADC judge flag
CE	FLG	0.87H.0	R	CE pin status flag
SIO1CH	FLG	0.88H.3	R/W	SIO_1 mode select flag
SB	FLG	0.88H.2	R/W	SB/SBI select flag
SIO1MS	FLG	0.88H.1	R/W	SIO_1 clock mode select flag
SIO1TX	FLG	0.88H.0	R/W	SIO_1 TX/RX select flag
TMMD3	FLG	0.89H.3	R/W	Timer interrupt mode select flag
TMMD2	FLG	0.89H.2	R/W	Timer interrupt mode select flag
TMMD1	FLG	0.89H.1	R/W	Timer carry FF mode select flag
TMMD0	FLG	0.89H.0	R/W	Timer carry FF mode select flag
INT1	FLG	0.8FH.1	R	INT_1 pin status flag
INT0	FLG	0.8FH.0	R	INT_0 pin status flag
KSEN	FLG	0.90H.1	R/W	Key source decoder enable flag
LCDEN	FLG	0.90H.0	R/W	LCD driver enadle flag
P0YON	FLG	0.91H.3	R/W	Port 0Y enable flag
P0XON	FLG	0.91H.2	R/W	Port 0X enable flag
P0EON	FLG	0.91H.1	R/W	Port 0E enable flag
P0FON	FLG	0.91H.0	R/W	Port 0F enable flag
IFCMD1	FLG	0.92H.3	R/W	IF counter mode select flag
IFCMD0	FLG	0.92H.2	R/W	IF counter mode select flag
IFCCK1	FLG	0.92H.1	R/W	IF counter clock select flag
IFCCK0	FLG	0.92H.0	R/W	IF counter clock select flag
PWM2ON	FLG	0.93H.3	R/W	PWM2 enable flag
PWM1ON	FLG	0.93H.2	R/W	PWM1 enable flag
PWM0ON	FLG	0.93H.1	R/W	PWM0 enable fl g
CGPON	FLG	0.93H.0	R/W	CGP enable flag
ADCCH3	FLG	0.94H.3	R/W	AD mode select flag (Dummy : 0)
ADCCH2	FLG	0.94H.2	R/W	AD mode select flag
ADCCH1	FLG	0.94H.1	R/W	AD mode select flag
ADCCH0	FLG	0.94H.0	R/W	AD mode select flag

Reserved word	Type	Address	Read/ Write	Function
PLULDLY3	FLG	0.95H.3	R/W	PLL unlock time select flag (Dummy : 0)
PLULDLY2	FLG	0.95H.2	R/W	PLL unlock time select flag (Dummy : 0)
PLULDLY1	FLG	0.95H.1	R/W	PLL unlock time select flag
PLULDLY0	FLG	0.95H.0	R/W	PLL unlock time select flag
KEYJ	FLG	0.96H.0	R	Key input judge flag
TMCY	FLG	0.97H.0	R	Timer carry FF status flag
SBACK	FLG	0.98H.3	R/W	SB acknowledge flag
SIO1NWT	FLG	0.98H.2	R/W	SIO_1 not wait flag
SIO1WRQ1	FLG	0.98H.1	R/W	SIO_1 wait mode flag
SIO1WRQ0	FLG	0.98H.0	R/W	SIO_1 wait mode flag
IEG1	FLG	0.9FH.1	R/W	INT_1 interrupt edge select flag
IEG0	FLG	0.9FH.0	R/W	INT_0 interrupt edge select flag
PLLMD3	FLG	0.0A1H.3	R/W	PLL mode select flag (Dummy : 0)
PLLMD2	FLG	0.0A1H.2	R/W	PLL mode select flag (Dummy : 0)
PLLMD1	FLG	0.0A1H.1	R/W	PLL mode select flag
PLLMD0	FLG	0.0A1H.0	R/W	PLL mode select flag
IFCSTRT	FLG	0.0A3H.1	W	IF counter start flag
IFCRES	FLG	0.0A3H.0	W	IF counter reset flag
P0CGIO	FLG	0.0A7H.0	R/W	Port 0C I/O select flag
SIO1SF8	FLG	0.0A8H.3	R	SIO_1 clock counter status flag
SIO1SF9	FLG	0.0A8H.2	R	SIO_1 clock counter status flag
SBSTT	FLG	0.0A8H.1	R	SB start condition status flag
SBBSY	FLG	0.0A8H.0	R	SB start/stop condition status flag
IPIFC	FLG	0.0AEH.0	R/W	IF counter interrupt permission flag
IPSIO1	FLG	0.0AFH.3	R/W	SIO_1 interrupt permission flag
IPTM	FLG	0.0AFH.2	R/W	Timer interrupt permission flag
IP1	FLG	0.0AFH.1	R/W	INT_1 interrupt permission flag
IP0	FLG	0.0AFH.0	R/W	INT_0 interrupt permission flag
PLLRFMD3	FLG	0.0B1H.3	R/W	PLL reference clock select flag
PLLRFMD2	FLG	0.0B1H.2	R/W	PLL reference clock select flag
PLLRFMD1	FLG	0.0B1H.1	R/W	PLL reference clock select flag
PLLRFMD0	FLG	0.0B1H.0	R/W	PLL reference clock select flag
P1ABIO3	FLG	0.0B5H.3	R/W	$P1A_3$ I/O select flag
P1ABIO2	FLG	0.0B5H.2	R/W	$P1A_2$ I/O select flag
P1ABIO1	FLG	0.0B5H.1	R/W	$P1A_1$ I/O select flag
P1ABIO0	FLG	0.0B5H.0	R/W	$P1A_1$ I/O select flag
P0BBIO3	FLG	0.0B6H.3	R/W	$P0B_3$ I/O select flag
P0BBIO2	FLG	0.0B6H.2	R/W	$P0B_2$ I/O select flag
P0BBIO1	FLG	0.0B6H.1	R/W	$P0B_1$ I/O select flag
P0BBIO0	FLG	0.0B6H.0	R/W	$P0B_0$ I/O select flag

Reserved word	Type	Address	Read/ Write	Function
P0ABIO3	FLG	0.0B7H.3	R/W	$P0A_3$ I/O select flag
P0ABIO2	FLG	0.0B7H.2	R/W	$P0A_2$ I/O select flag
P0ABIO1	FLG	0.0B7H.1	R/W	$P0A_1$ I/O select flag
P0ABIO0	FLG	0.0B7H.0	R/W	$P0A_0$ I/O select flag
SIO1IMD3	FLG	0.0B8H.3	R/W	SIO_1 interrupt mode select flag (Dummy : 0)
SIO1IMD2	FLG	0.0B8H.2	R/W	SIO_1 interrupt mode select flag (Dummy : 0)
SIO1IMD1	FLG	0.0B8H.1	R/W	SIO_1 interrupt mode select flag
SIO1IMD0	FLG	0.0B8H.0	R/W	SIO_1 interrupt mode select flag
SIO1CK3	FLG	0.0B9H.3	R/W	SIO_1 shift clock select flag (Dummy : 0)
SIO1CK2	FLG	0.0B9H.2	R/W	SIO_1 shift clock select flag (Dummy : 0)
SIO1CK1	FLG	0.0B9H.1	R/W	SIO_1 shift clock select flag
SIO1CK0	FLG	0.0B9H.0	R/W	SIO_1 shift clock select flag
IRQIFC	FLG	0.0BEH.0	R/W	IF counter interrupt request flag
IRQSIO1	FLG	0.0BFH.3	R/W	SIO_1 interrupt request flag
IRQTM	FLG	0.0BFH.2	R/W	Timer interrupt request flag
IRQ1	FLG	0.0BFH.1	R/W	INT_1 interrupt request flag
IRQ0	FLG	0.0BFH.0	R/W	INT_0 interrupt request flag

26.1.6 Peripheral Hardware Address

Reserved word	Type	Address	Read/ Write	Function
DBF	DAT	0FH	R/W	Data buffer address of GET/PUT instruction
IX	DAT	01H	R/W	Index register address of INC instruction
ADCR	DAT	02H	R/W	A/D converter VREF data register
SIO2SFR	DAT	03H	R/W	SIO_2 presettable shift register
SIO1SFR	DAT	04H	R/W	SIO_1 presettable shift register
PWMR0	DAT	05H	R/W	PWM0 data register
PWMR1	DAT	06H	R/W	PWM1 data register
PWMR2	DAT	07H	R/W	PWM2 data register
LCDR0	DAT	08H	W	LCD group data register 0
LCDR1	DAT	09H	W	LCD group data register 1
LCDR2	DAT	0AH	W	LCD group data register 2
LCDR3	DAT	0BH	W	LCD group data register 3
LCDR4	DAT	0CH	W	LCD group data register 4
P0X	DAT	0CH	W	Port 0X data register
LCDR5	DAT	0DH	W	LCD group data register 5
LCDR6	DAT	0EH	W	LCD group data register 6
LCDR7	DAT	0FH	W	LCD group data register 7
CGPR	DAT	20H	R/W	CGP data register
AR	DAT	40H	R/W	Address register address of GET/PUT/PUSH/CALL/BR/ MOVT/INC
PLLR	DAT	41H	R/W	PLL data register
KSR	DAT	42H	R/W	Key source data register
P0Y	DAT	42H	R/W	Port 0Y data register
IFC	DAT	43H	R	IF counter data register

27. ELECTRICAL CHARACTERISTICS

27.1 ABSOLUTE MAXIMUM RATINGS (Unless otherwise specified, $T_a = 25 \pm 2$ °C

Source Voltage	V_{DD}		−0.3 – +6.0	V
Input Voltage	V_I		−0.3 – V_{DD} + 0.3	V
Output Voltage	V_O	Excluding $P1B_1 - P1B_3$, $P0A_2$, $P0A_3$ and LPF_{OUT}	−0.3 – V_{DD} + 0.3	V
Output Withstand Voltage	V_{BDS1}	$P1B_1 - P1B_3$, LPF_{OUT}	18.0	V
Output Withstand Voltage	V_{BDS2}	$P0A_2$, $P0A_3$	V_{DD} + 0.3	V
High-Level Output Current	I_{OH}	One pin	−12	mA
		All pins	−20	mA
Low-Level Output Current	I_{OL}	One pin	12	mA
		All pins	20	mA
Operating Temperature	T_{opt}		−40 – +85	°C
Storage Temperature	T_{stg}		−55 – +125	°C

27.2 RECOMMENDED OPERATING CONDITIONS

CHARACTERISTICS	SYMBOL	MIN.	TYP.	MAX.	UNIT	CONDITION
Source Voltage	V_{DD1}	4.5	5.0	5.5	V	PLL and CPU are operating
Source Voltage	V_{DD2}	3.5	5.0	5.5	V	PLL is OFF and CPU is operating
Data Holding Voltage	V_{DDR}	2.2		5.5	V	Quartz oscillator OFF
Source Voltage Rise Time	t_{rise}			500	ms	$V_{DD} = 0 \rightarrow 4.5$ V
Input Amplitude	V_{in1}	0.5		V_{DD}	V_{p-p}	VCOL, VCOH
Input Amplitude	V_{in2}	0.5		V_{DD}	V_{p-p}	AMIFC, FMIFC
Output Withstand Voltage	V_{BDS}			16.0	V	$P1B_1 - P1B_3$, LPF_{OUT}
Operating Temperature	T_{opt}	−40		+85	°C	

27.3 DC CHARACTERISTICS (Unless otherwise specified, T_a = −40 to +85 °C, V_{DD} = 4.5 to 5.5 V)

CHARACTERISTICS	SYMBOL	STANDARD VALUE				CONDITION
		MIN.	TYP.	MAX.	UNIT	
Source Voltage	V_{DD1}	4.5	5.0	5.5	V	CPU and PLL are operating
Source Voltage	V_{DD2}	3.5	5.0	5.5	V	CPU is operating and PLL is OFF
Source Current	I_{DD1}		1.2	2.4	mA	CPU is operating and PLL is OFF. X_{IN} pin Sine wave input (f_{in} = 4.5 MHz, V_{in} = V_{DD}), Ta = 25 °C
Source Current	I_{DD2}		0.45	0.90	mA	CPU is operating, PLL is OFF, and HALT instruction is used (20 instructions executed per 1 ms). X_{in} pin Sine wave input (f_{in} = 4.5 MHz, V_{in} = V_{DD}). T_a = 25 °C
Data Holding Voltage	V_{DDR1}	3.5		5.5	V	Power failure detection by timer FF, quartz oscillator oscillating
Data Holding Voltage	V_{DDR2}	2.2		5.5	V	Power failure detection by timer FF, quartz oscillator not oscillating
Data Holding Voltage	V_{DDR3}	2.0		5.5	V	Data memory (RAM) holding
Data Holding Current	I_{DDR1}		2	15	μA	Quartz oscillator not oscillating T_a = 25 °C
Data Holding Current	I_{DDR2}		2	10	μA	Quartz oscillator not oscillating V_{DD} = 5.0 V, T_a = 25 °C
Intermediate Level Output Voltage	V_{OM1}	2.3	2.5	2.7	V	COM_0 COM_1 V_{DD} = 5 V
High Level Input Voltage	V_{IH1}	0.8V_{DD}		V_{DD}	V	$P0A_0$ – $P0A_3$, $P0B_0$ – $P0B_3$, $P0C_0$ – $P0C_3$, $P1A_0$ – $P1A_3$, $P1D_0$ – $P1D_3$ CE, INT_0, INT_1
High Level Input Voltage	V_{IH2}	0.6V_{DD}		V_{DD}	V	$P0D_0$ – $P0D_3$
Low Level Input Voltage	V_{IL}	0		0.2V_{DD}	V	$P0A_0$ – $P0A_3$, $P0B_0$ – $P0B_3$, $P0C_0$ – $P0C_3$, $P0D_0$ – $P0D_3$, $P1A_0$ – $P1A_3$, $P1D_0$ – $P1D_3$, CE, INT_0, INT_1
High Level Output Current	I_{OH1}	−1.0	−5.0		mA	$P0A_0$, $P0A_1$, $P0B_0$ – $P0B_3$, $P0C_0$ – $P0C_3$, $P1A_0$ – $P1A_3$, $P1C_0$ – $P1C_3$, $P1B_0$, $P2A_0$ V_{OH} = V_{DD} − 1 V
High Level Output Current	I_{OH2}	−1.0	−4.0		mA	LCD_0 – LCD_{29}, EO_0, EO_1 V_{OH} = V_{DD} − 1 V
Low Level Output Current	I_{OL1}	1.0	7.0		mA	$P0A_0$ – $P0A_3$, $P0B_0$ – $P0B_3$, $P0C_0$ – $P0C_3$, $P1A_0$ – $P1A_3$, $P1C_0$ – $P1C_3$, $P1B_0$, $P2A_0$ V_{OL} = 1 V
Low Level Output Current	I_{OL2}	1.0	3.5		mA	LCD_0 – LCD_{29}, EO_0, EO_1, V_{OL} = 1 V
Low Level Output Current	I_{OL3}	1.0	2.0		mA	$P1B_1$ – $P1B_3$, V_{OL} = 1 V
Low Level Output Current	I_{OL4}	1.0	10.0		mA	$P0A_2$, $P0A_3$, V_{OL} = 1 V

CHARACTERISTICS	SYMBOL	STANDARD VALUE				CONDITION
		MIN.	TYP.	MAX.	UNIT	
High Level Input Current	I_{IH1}	0.1	0.8		mA	VCOH pull-down, $V_{IH} = V_{DD}$
High Level Input Current	I_{IH2}	0.1	0.8		mA	VCOL pull-down, $V_{IH} = V_{DD}$
High Level Input Current	I_{IH3}	0.1	1.3		mA	X_{IN} pull-down, $V_{IH} = V_{DD}$
High Level Input Current	I_{IH4}	0.05	0.13	0.30	mA	$P0D_0$-$P0D_3$ pull-down, $V_{IH} = V_{DD}$
Output Off Leak Current	I_{L1}			500	nA	$P0A_2$, $P0A_3$, $V_{OH} = V_{DD}$
Output Off Leak Current	I_{L2}			500	nA	$P1B_1 - P1B_3$, LPF_{OUT}, $V_{OH} = 16$ V
Output Off Leak Current	I_{L3}			±100	nA	EO_0, EO_1, $V_{OH} = V_{DD}$, $V_{OL} = 0$ V

27.4 AC CHARACTERISTICS (Unless otherwise specified, T_a = −40 to +85 °C, V_{DD} = 4.5 to 5.5 V)

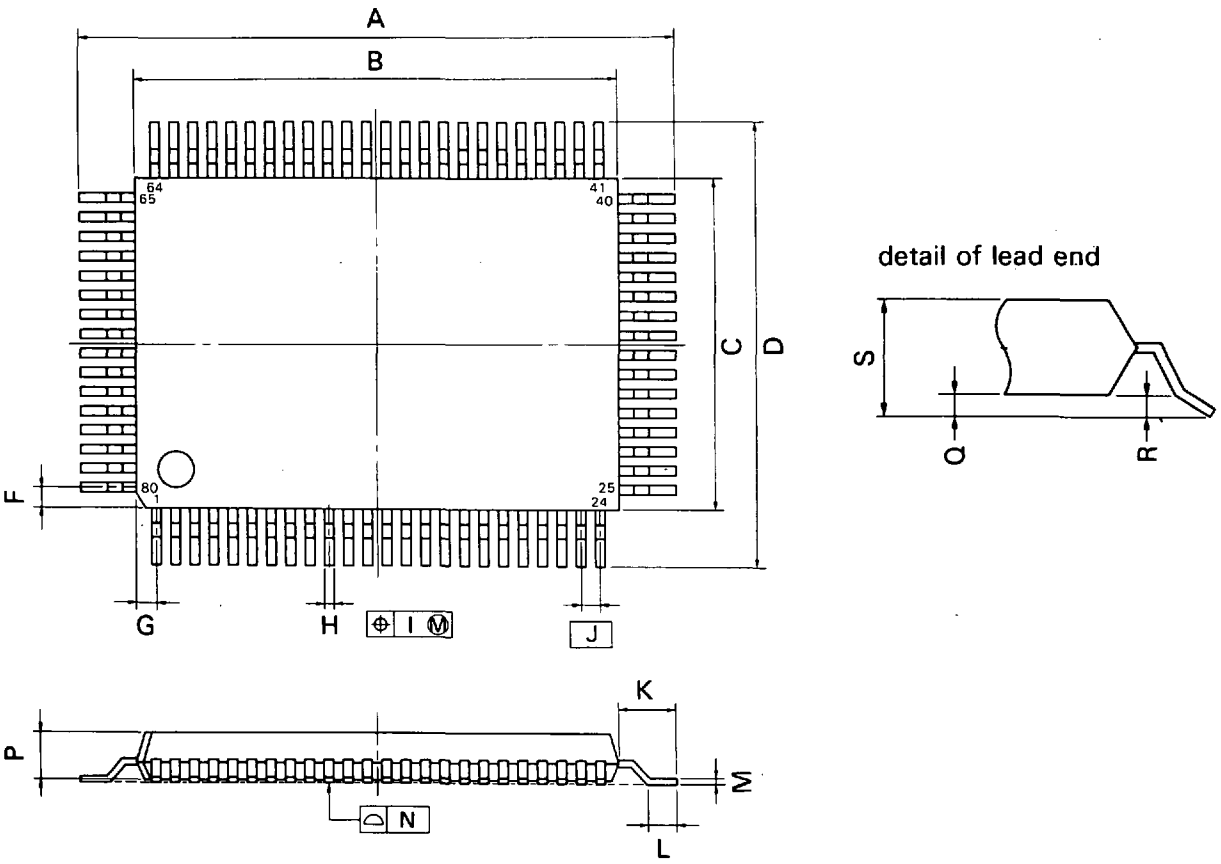
CHARACTERISTICS	SYMBOL	STANDARD VALUE				CONDITION
		MIN.	TYP.	MAX.	UNIT	
Operating Frequency	f_{in1}	0.5		30	MHz	VCOL MF mode, sice wave input V_{in} = 0.3 V_{p-p}
Operating Frequency	f_{in2}	5		40	MHz	VCOL HF mode, sice wave input V_{in} = 0.3 V_{p-p}
Operating Frequency	f_{in3}	9		150	MHz	VCOH, sine wave input V_{in} = 0.3 V_{p-p}
Operating Frequency	f_{in4}	0.1		1	MHz	AMIFC, sine wave input V_{in} = 0.3 V_{p-p}
Operating Frequency	f_{in5}	0.44		0.46	MHz	AMIFC, sine wave input V_{in} = 0.05 V_{p-p}
Operating Frequency	f_{in6}	5		15	MHz	FMIFC, sine wave input V_{in} = 0.3 V_{p-p}
Operating Frequency	f_{in7}	10.5		10.9	MHz	FMIFC, sine wave input V_{in} = 0.06 V_{p-p}
AD Converting Resolution				6	bit	
Total Error of AD Conversion			±1	±1.5	LSB	T_a = −10 to +50 °C

27.5 REFERENCE CHARACTERISTICS

CHARACTERISTICS	SYMBOL	STANDARD VALUE				CONDITION
		MIN.	TYP.	MAX.	UNIT	
Source Current	I_{DD3}		15		mA	CPU and PLL are operating VCOH sine wave input f_{in} = 150 MHz, V_{in} = 0.5 V_{p-p} V_{DD} = 5 V, T_a = 25 °C
High Level Output Current	I_{OH4}		−0.2		mA	COM_0, COM_1, $V_{OH} = V_{DD} - 1$ V
Intermediate Level Output Current	I_{OM1}		−20		μA	COM_0, COM_1, $V_{OM} = V_{DD} - 1$ V
Intermediate Level Output Current	I_{OM2}		20		μA	COM_0, COM_1, V_{OM} = 1 V
Low Level Output Current	I_{OL5}		0.2		mA	COM_0, COM_1, V_{OL} = 1 V

28. PACKAGE DIMENSION

80 PIN PLASTIC QFP (14×20)



S80GF-80-3B9

NOTE

Each lead centerline is located within 0.15 mm (0.006 inch) of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS	INCHES
A	$23.2^{\pm 0.4}$	$0.913^{+0.017}_{-0.016}$
B	$20^{\pm 0.2}$	$0.787^{+0.009}_{-0.008}$
C	$14^{\pm 0.2}$	$0.551^{+0.009}_{-0.008}$
D	$17.2^{\pm 0.4}$	$0.677^{\pm 0.016}$
F	1.0	0.039
G	0.8	0.031
H	$0.35^{\pm 0.10}$	$0.014^{+0.004}_{-0.005}$
I	0.15	0.006
J	0.8 (T.P.)	0.031 (T.P.)
K	$1.6^{\pm 0.2}$	$0.063^{\pm 0.008}$
L	$0.8^{\pm 0.2}$	$0.031^{+0.009}_{-0.008}$
M	$0.15^{+0.10}_{-0.05}$	$0.006^{+0.004}_{-0.003}$
N	0.15	0.006
P	2.7	0.106
Q	$0.1^{\pm 0.1}$	$0.004^{\pm 0.004}$
R	$0.1^{\pm 0.1}$	$0.004^{\pm 0.004}$
S	3.0 MAX.	0.119 MAX.

29. RECOMMENDED SOLDERING CONDITIONS

The following conditions (see table below) must be met when soldering this product.

Please consult with our sales offices in case other soldering process is used, or in case soldering is done under different conditions.

TYPES OF SURFACE MOUNT DEVICE

For more details, refer to our document "SMT MANUAL" (IEI-1207).

μPD17005GF-3B9

Soldering process	Soldering conditions	Symbol
Infrared ray reflow	Peak package's surface temperature: 220 °C or below, Reflow time: 30 seconds or below (200 °C or higher), Number of reflow process: 1, Exposure limit*: 2 days (16 hours pre-backing is required at 125 °C afterwards)	IR20-162
VPS	Peak package's surface temperature: 215 °C or below, Reflow time: 40 seconds or below (200 °C or higher), Number of reflow process: 1, Exposure limit*: 2 days (16 hours pre-backing is required at 125 °C afterwards)	VP15-162
Wave soldering	Solder temperature: 260 °C or below, Flow time: 10 seconds or below, Number of flow process: 1, Exposure limit*: 2 days (16 hours pre-backing is required at 125 °C afterwards)	WS60—162
Partial heating method	Terminal temperature: 300 °C or below, Flow time: 10 seconds or below, Exposure limit*: None	

*: Exposure limit before soldering after dry-pack package is opened.
Storage conditions: 25 °C and relative humidity at 65% or less.

Note: Do not apply more than a single process at once, except for "partial heating method".

APPENDIX A DIFFERENCE BETWEEN μPD17003A and μPD17005

(1) Hardware

Item	μPD17003A	μPD17005
ROM (x 16 bits)	3836 Page 0 (0000H – 07FFH) Page 1 (0800H – 0EFBH)	7932 Page 0 (0000H – 07FFH) Page 1 (0800H – 0FFFH) Page 2 (1000H – 17FFH) Page 3 (1800H – 1EFBH)
RAM (x 4 bits)	320 (BANK0 – BANK2)	432 (BANK0 – BANK3)
Program counter	Consists of 12 bits.	Consists of 13 bits.
Address stack register	Consists of 12 bits.	Consists of 13 bits.
Address register	Low-order eight bits are valid. High-order eight bits are fixed in "0".	Low-order 13 bits are valid. High-order three bits are fixed in "0". (Valid data is 0000H to 1EFBH that are in the whole area of ROM.)

(2) Software

Item	μPD17003A	μPD17005
BR command operation code	The BR command sent to page 0 is 0CH. The BR command sent to page 1 is 0DH.	The BR command sent to page 0 is 0CH. The BR command sent to page 1 is 0DH. The BR command sent to page 2 is 0EH. The BR command sent to page 3 is 0FH.
Interrupt request flag	An error occurs during assemble when a flag operation built-in macrocommand is used. Each flag is as follows: IRQIFC IRQSIO1 IRQTM IRQ1 IRQ0	An error occurs during assemble when a flag operation built-in macrocommand is issued. Each flag is as follows: IRQIFC IRQSIO1 IRQTM IRQ1 IRQ0

(3) Development tool

Item		μPD17003A	μPD17005
Hardware	SE board	SE-17003	SE-17005
	Emulation probe	EP-17003GF	
Software	Device file	AS17003	AS17005
	Macrolibrary	• IFCSET.LIB • IRQ.MAR (An interrupt request flag is activated using this library.)	• IFCSET.LIB • IRQ.MAC (An interrupt request flag is activated using this library.)

APPENDIX B DEVELOPMENT TOOL

The following support tools are available for development of the μPD17005.

Hardware					
Name		Description			Order name
In-circuit emulator (IE-17K)		IE-17K is an evaluation in-circuit emulator which can be used commonly for the 17K series. For program development of μPD17005, this IE-17K and SE-17005, which is a system evaluation board, must be used in combination. IE-17K is provided for operation on RAM basis, and addition/modification of programs can be performed at once on the console by simply connecting a console to IE-17K. SIMPLEHOST™ will help develop advanced application programs.			IE-17K
SE board (SE-17005)		SE-17005 is a system evaluation board of μPD17005, and may be used singly or in combination with IE-17K.			SE-17005
Probe (EP-17003GF)		EP-17005GF is a probe for connecting SE-17005 and target system.			EP-17003GF
Receptacle (EV-9200G-80)		EV-9200G-80 is a socket for connecting EP-17003GF and tartet.			EV-9200G-80
One-time PROM (μPD17P005 Note)		μPD17P005 is an one-time PROM version of μPD17005 for its program evaluation and small quantity production.			μPD17P005GF-3B9
Software					
Name		Description	Host machine	OS	Order name
Assembler	Assembler body (AS17K)	AS17K is the main body of assembler to be used commonly for 17K series. This AS17K are used in combination with the device file (AS17005).	PC-9801 series IBM PC/AT™	MS-DOS™ Ver 2.11 Ver 3.1 PC DOS™ Ver 3.1	MS-DOS version μS5A10AS17K (5" 2HD) μS5A13AS17K (3.5" 2HD) PC DOS version μS7B11AS17K (5" 2D)
	Device file (AS17005)	AS17005 is used in combination with AS17K for assembling the programs of μPD17005.			MS-DOS version μS5A10AS17005 (5" 2HD) μS5A13AS17005 (3.5" 2HD) PC DOS version μS7B11AS17005 (5" 2D)
Support software (SIMPLEHOST)		SIMPLEHOST is a software which provides a man-machine interface on MS-WINDOW™ when developing programs by using IE-17K and personal computer.		MS-WINDOWS	MS-DOS version μS5A10IE17K (5" 2HD) μS5A13IE17K (3.5" 2HD) PC DOS version μS7B10IE17K* (5" 2HC)

*: Under development

MS-DOS™ and MS-WINDOWS™ are trademarks of Microsoft Corporation.
IBM PC/AT™ and PC DOS™ are trademarks of IBM Corporation.
SIMPLEHOST™ is trademark of NEC Corporation.

(MEMO)