NEC

MOS INTEGRATED CIRCUIT

μPD17062

4-BIT SINGLE-CHIP MICROCONTROLLER CONTAINING PLL FREQUENCY SYNTHESIZER AND IMAGE DISPLAY CONTROLLER

The μPD17062 is a 4-bit CMOS microcontroller for digital tuning systems. The single-chip device incorporates an image display controller enabling a range of different displays, together with a PLL frequency synthesizer.

The CPU has six main functions: 4-bit parallel addition, logic operation, multiple bit test, carry-flag set/reset, powerful interrupt, and a timer.

The device contains a user-programmable image display controller (IDC) for on-screen displays. The different displays can be controlled with simple programs.

The device also has a serial interface function, many input/output (I/O) ports controlled by powerful I/O instructions, and 6-bit pulse width modulation (PWM) output for a 4-bit A/D converter and D/A converter.

FEATURES

- 4-bit microcontroller for digital tuning system
- Internal PLL frequency synthesizer: With prescaler μPB595
- 5 V ±10%
- Low-power CMOS
- Program memory (ROM): 8K bytes (16 bits × 3968 steps)
- Data memory (RAM): 4 bits × 336 words
- 6 stack levels
- 35 easy-to-understand instruction sets
- Support of decimal operations
- Instruction execution time: 2 μs (with an 8-MHz crystal)
- Internal D/A converter: 6 bits × 4 (PWM output)
- Internal A/D converter: 4 bits × 6
- Internal horizontal synchronizing signal counter
- Internal commercial power frequency counter
- Internal power-failure detector and power-on reset circuit
- Internal image display controller (IDC) (user-programmable)
  Number of characters in display: Up to 99 on a single screen
  Display configuration: 14 rows × 19 columns
  Number of character types: 120
  Character format: 10 × 15 dots (rimming possible)
  Number of colors: 8
  Character size: Four sizes in each of the horizontal and vertical dimensions
  Internal 1H circuit for preventing vertical deflection
- Internal 8-bit serial interface (One system with two channels: three-wire or two-wire)
- Interrupt input for remote-controller signals (with noise canceler)
- Many I/O ports
  Number of I/O ports : 15
  Number of input ports : 4
  Number of output ports: 8

The information in this document is subject to change without notice.

Document No. IC-3560
(O.D. No. IC-8937)
Date Published January 1995 P
Printed in Japan

© NEC Corporation 1995

ORDERING INFORMATION

Part number	Package
μPD17062CU-×××	48-pin plastic shrink DIP (600 mil)
μPD17062GC-×××	64-pin plastic QFP (14 × 14 mm)

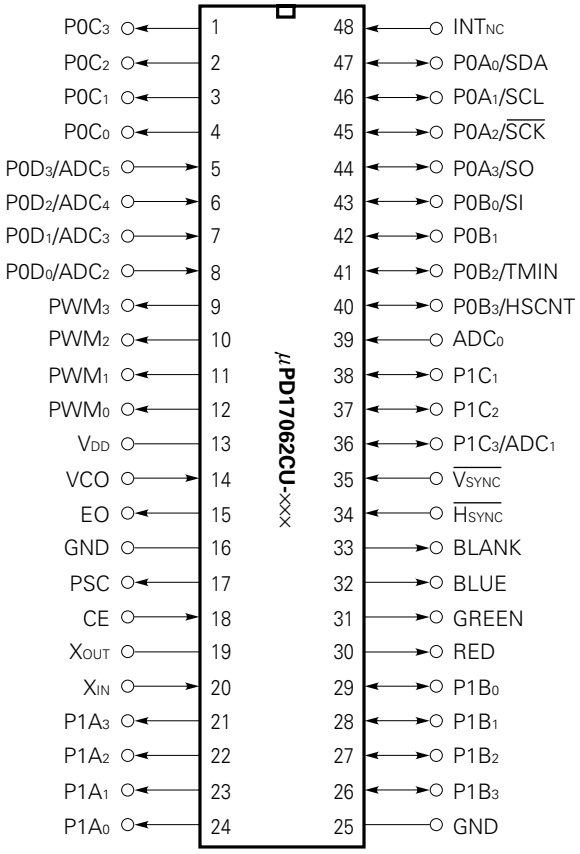
Remark ××× is the ROM code number.

FUNCTION OVERVIEW

Item	Function
ROM (program memory) capacity	3968 × 16 bits (masked ROM)
CROM (character ROM) capacity	1920 × 16 bits (included in ROM)
RAM (data memory) capacity	336 × 4 bits (including the area that can be used for VRAM)
VRAM (video RAM) capacity	224 × 4 bits (included in RAM)
Instruction execution time	2 μs (when the 8-MHz crystal is used)
Stack level	6 levels (stack operation possible)
Number of I/O ports	Number of input ports: 4
	Number of output ports: 8
	Number of I/O ports: 15
IDC (Image Display Controller)	Number of characters in display: Up to 99 on a single screen
	Display format: 10 × 15 dots, 14 rows × 19 columns
	Number of character types: 120 (user-programmable)
	Number of colors: 8
	Character size Vertical dimension : 1 to 4 times (can be set for each line) Horizontal dimension : 1 to 4 times (can be set for each character)
Serial interface	One system { Serial interface 0 (two-wire or I^2C bus compatible) Serial interface 1 (two-wire or three-wire)
D/A converter	6 bits × 4 (PWM output, withstand voltage of up to 12.5 V)
A/D converter	4 bits × 6 (successive-approximation converter by software)
Interrupt	4 channels { External interrupt : 2 channels Internal interrupt : 2 channels
Timer	1 channel (internal clock/zerocross input)
PLL frequency synthesizer	Scaling method : Pulse swallow method (VCO pin: Up to 40 MHz), external specialized two-modulus prescaler (μPB595, for example) Reference frequency : 6.25, 12.5, 25 kHz Charge pump : Error-out output Phase comparator : Capable of unlock detection by a program
Reset	Power-on reset
	Reset by the CE pin
	With power-failure detection function
Supply voltage	5 V ±10%

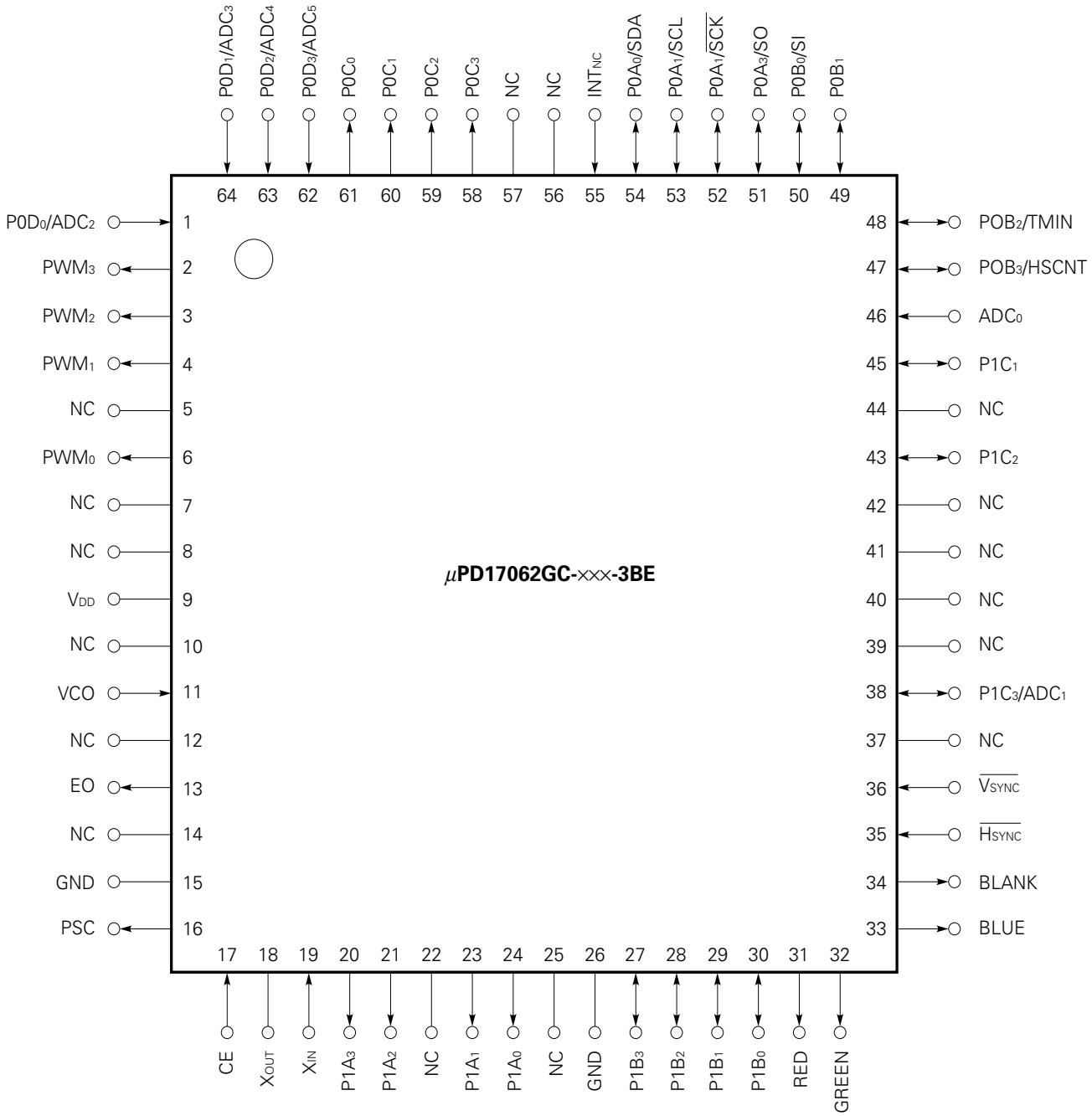
PIN CONFIGURATION (TOP VIEW)

48-pin plastic shrink DIP (600 mil)

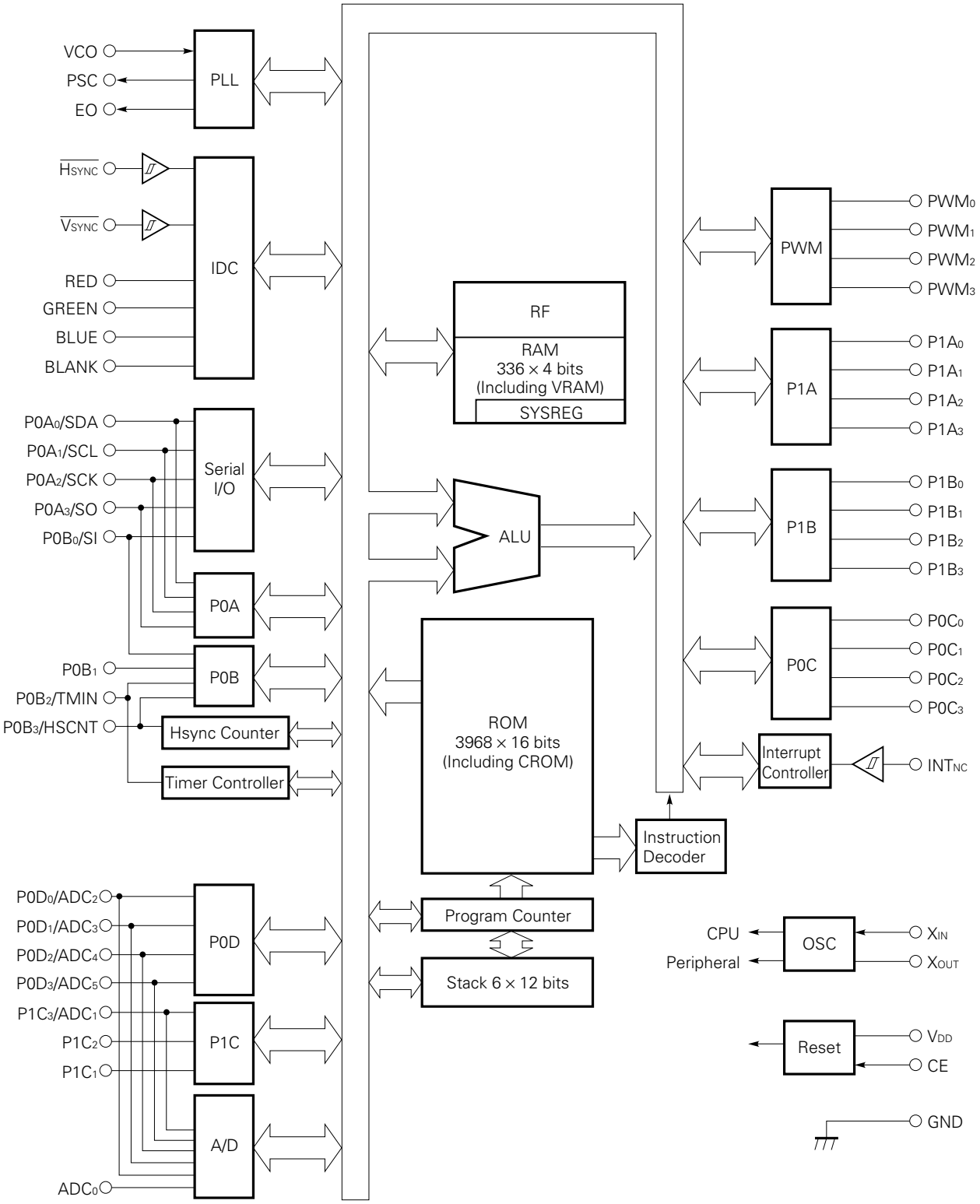


ADC0 to ADC5 : A/D converter input
BLANK : Blanking signal output
BLUE : Character signal output
CE : Chip enable
EO : Error out
GND : Ground
GREEN : Character signal output
HSCNT : Horizontal synchronizing signal counter input
$\overline{\text{HSYNC}}$: Horizontal synchronizing signal input
INTNC : Interrupt signal input
NC : No connection
PSC : Pulse swallow control output
PWM0 to PWM3 : Pulse width modulation output
P0A0 to P0A3 : Port 0A
P0B0 to P0B3 : Port 0B
P0C0 to P0C3 : Port 0C
P0D0 to P0D3 : Port 0D
P1A0 to P1A3 : Port 1A
P1B0 to P1B3 : Port 1B
P1C1 to P1C3 : Port 1C
RED : Character signal output
$\overline{\text{SCK}}$: Shift clock input/output
SCL : Shift clock input/output
SDA : Serial data input/output
SI : Serial data input
SO : Serial data output
TMIN : Timer event input
VCO : Local oscillation input
VDD : Main power supply
$\overline{\text{VSYNC}}$: Vertical synchronizing signal input
XIN : Clock oscillation
XOUT : Clock oscillation

64-pin plastic QFP (14 × 14 mm)



BLOCK DIAGRAM



CONTENTS

1. PINS

1.1 PIN FUNCTIONS

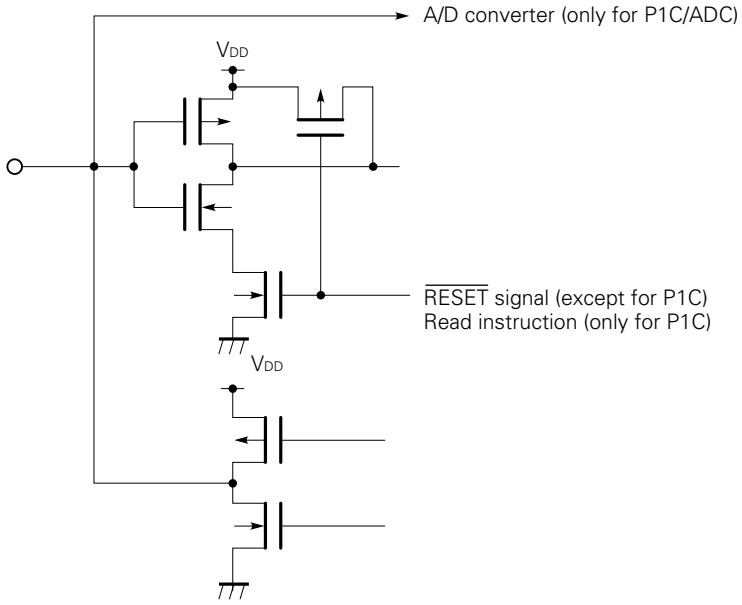
Pin No. DIP	Pin No. QFP (GC)	Symbol	Description	Output type	At power-on reset
1 | 4	58 | 61	$P0C_3$ | $P0C_0$	4-bit output port	CMOS push-pull	Undefined
5 | 8	62 | 1	$P0D_3/ADC_5$ | $P0D_0/ADC_2$	Input of port 0D and A/D converter • $P0D_3$ to $P0D_0$ 4-bit input port containing a pull-down resistor. • ADC_5 to ADC_2 Input of a 4-bit A/D converter, which is a software-based successive-approximation type. The reference voltage is V_{DD}.	—	Input
9 | 12	2 | 6	PWM_3 | PWM_0	Output of a 6-bit D/A converter. The output type is PWM. Output is done at a frequency of 15.625 kHz. The pin can also be used as a one-bit output port.	N-ch open drain	Undefined
13	9	V_{DD} V_{DD1} V_{DD0}	Supplies the power to the device. To enable all functions, 5 V ±10% is supplied. To operate only the CPU, 4 V is required. In the clock-stop state, the voltage can be reduced to 3.5 V. When the supply voltage increases from 0 V to 4 V, a power-on reset occurs and the program is started from address 0. Apply an identical voltage to all pins.	—	—
14	11	VCO	Inputs the signal obtained by dividing the local oscillation output by the specialized prescaler.	—	Input
15	13	EO	Outputs the PLL error signal. The signal is input through the external LPF to the local oscillation circuit.	CMOS tristate	Hi-z
16	15	GND GND2 GND1 GND0	Grounds the device. Connect all pins to ground.	—	—
17	16	PSC	Outputs the signal to switch the frequency division ratio of the specialized prescaler.	CMOS push-pull	Undefined
18	17	CE	Inputs the signal to select the device. To operate the PLL and IDC, set the input signal high. If the input signal is low, the device can be backed up with a low current drain by executing a stop instruction. When the input signal goes high, the device is reset and the program is started from address 0.	—	Input
19	18	X_{OUT}	Used to connect a crystal. An 8-MHz crystal is used.	—	—
20	19	X_{IN}		—	Input

Pin No. DIP	Pin No. QFP (GC)	Symbol	Description	Output type	At power-on reset
21 – 24	20 – 24	$P1A_3$ – $P1A_0$	4-bit output port. This N-ch open-drain output port has an intermediate withstand voltage.	N-ch open-drain	Undefined
26 – 29	27 – 30	$P1B_3$ – $P1B_0$	4-bit I/O port. Each bit can be set for input or output.	CMOS push-pull	Input
30 31 32	31 32 33	RED GREEN BLUE	Outputs the character data corresponding to R, G, and B of the IDC display. The output is active-high.	CMOS push-pull	Low level
33	34	BLANK	Outputs the blanking signal for cutting the video signal of the IDC display. The output is active-high.	CMOS push-pull	Low level
34	35	$\overline{HSYNC}$	Inputs the horizontal synchronizing signal of the IDC display. The input must be active-low.	—	Input
35	36	$\overline{VSYNC}$	Inputs the vertical synchronizing signal of the IDC display. The input must be active-low. The input signal can generate an interrupt.	—	Input
36 – 38	38 – 45	$P1C_3/ADC_1$ $P1C_2$ $P1C_1$	Input of port 1C and A/D converter • $P1C_3$ to $P1C_1$ 3-bit I/O port • ADC_1 Input of a 4-bit A/D converter	CMOS push-pull	Input
39	46	ADC_0	Input of a 4-bit A/D converter	—	Input
40 – 43 44 – 47	47 – 50 51 – 54	$P0B_3$/HSCNT $P0B_2$/TMIN $P0B_1$ $P0B_0$/SI $P0A_3$/SO $P0A_2/\overline{SCK}$ $P0A_1$/SCL $P0A_0$/SDA	Serial interface and input for port 0B, port 0A, horizontal synchronizing signal counter, and timer • $P0A_3$ to $P0A_0$ 4-bit I/O port. Each bit can be set for input or output. • $P0B_3$ to $P0B_0$ 4-bit I/O port. Each bit can be set for input or output. • HSCNT Inputs the count of the horizontal synchronizing signal. The input is self-biased. • TMIN Timer input. The pin inputs the commercial power to be used for the clock. • SI, SO, $\overline{SCK}$ Input/output for the three-wire serial interface • SI: Serial data input • SO: Serial data output • $\overline{SCK}$: Shift clock input/output • SDA, SCL Input/output for the two-wire serial interface • SCL: Serial clock input/output • SDA: Serial data input/output	N-ch open-drain ($P0A_1$, $P0A_0$) CMOS push-pull (Other than $P0A_1$ or $P0A_0$)	Input

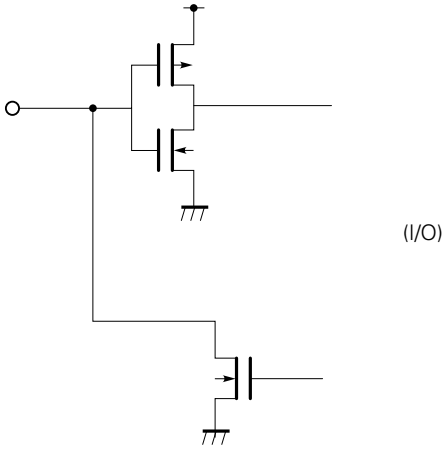
Pin No. DIP	Pin No. QFP (GC)	Symbol	Description	Output type	At power-on reset
48	55	INT_{NC}	Interrupt input. Contains the noise canceler. An interrupt can be generated at either the rising or falling edge of the input signal.	—	Input
—	5 6 7 8 10 12 14 22 25 37 39 40 41 42 44 56 57	NC	No connection. The pins are not connected to the internal circuit of the device. They can be used as desired.		

1.2 EQUIVALENT CIRCUITS OF THE PINS

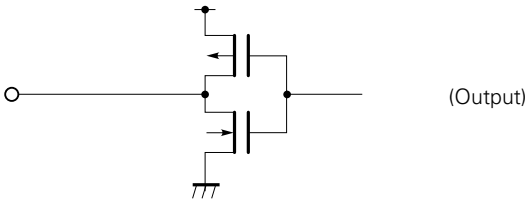
P0A ($P0A_3$/SO, $P0A_2/\overline{SCK}$)
P0B ($P0B_1$, $P0B_0$/SI)
P1B ($P1B_3$, $P1B_2$, $P1B_1$, $P1B_0$)
P1C ($P1C_3$/ADC_1, $P1C_2$, $P1C_1$)



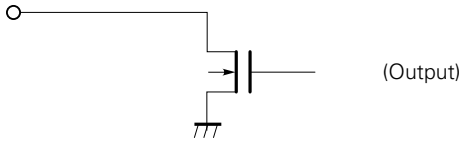
P0A ($P0A_1$/SCL, $P0A_0$/SDA)



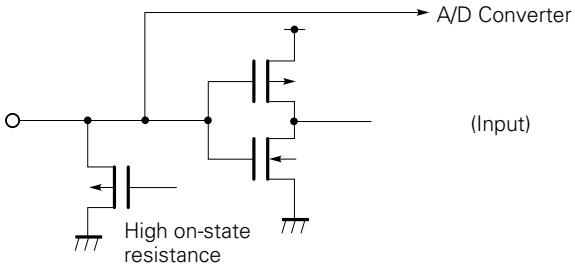
P0C ($P0C_3$, $P0C_2$, $P0C_1$, $P0C_0$)
RED, GREEN, BLUE, BLANK, PSC



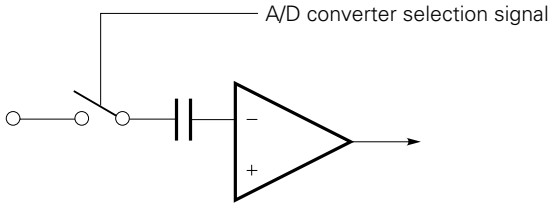
PWM (PWM_3, PWM_2, PWM_1, PWM_0)
P1A ($P1A_3$, $P1A_2$, $P1A_1$, $P1A_0$)



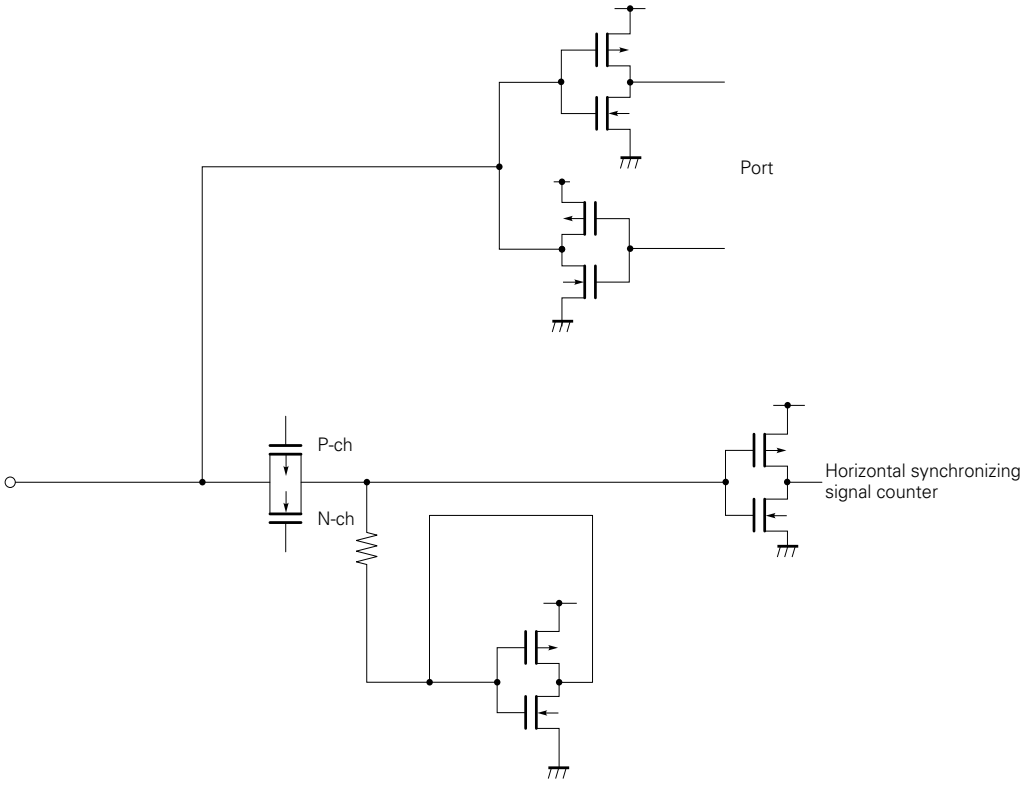
P0D ($P0D_3$/ADC_5, $P0D_2$/ADC_4, $P0D_1$/ADC_3, $P0D_0$/ADC_2)



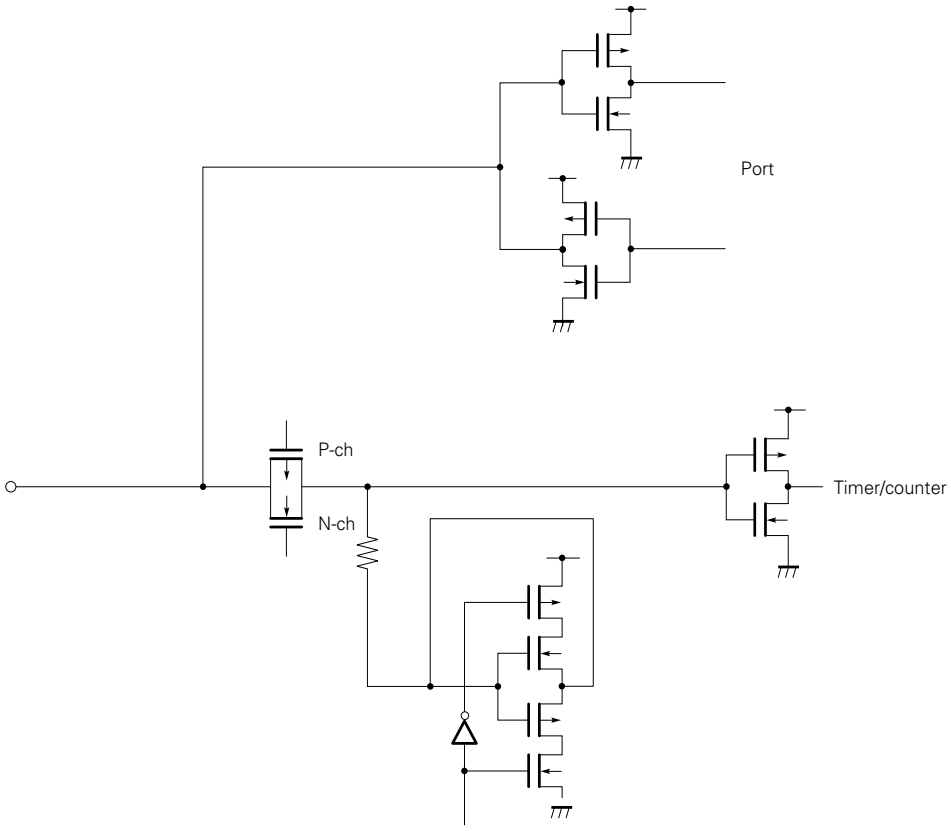
ADC_0



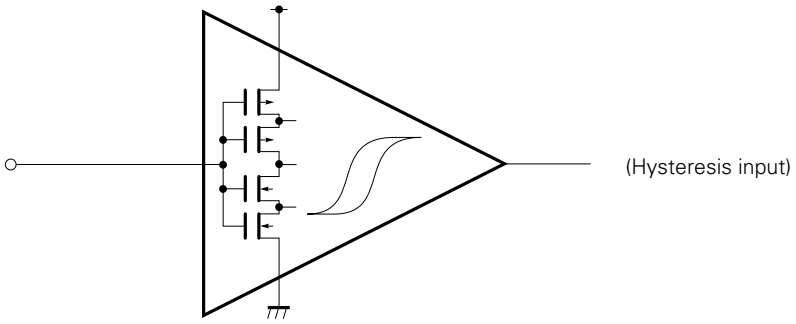
P0B3/HSCNT



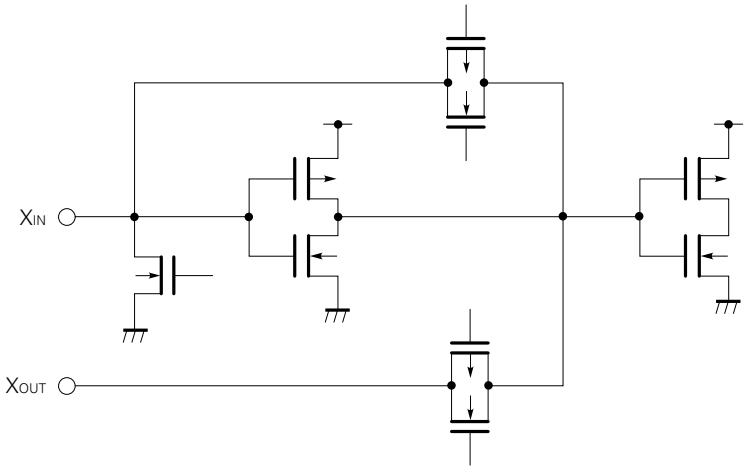
P0B2/TMIN



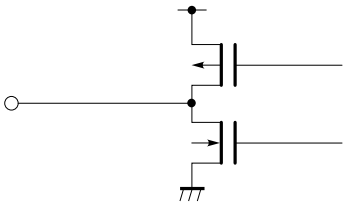
$\overline{\text{HSYNC}}$, $\overline{\text{VSYNC}}$, INTNC, CE



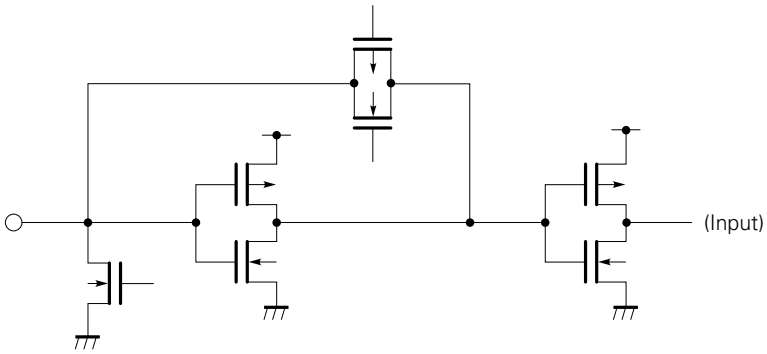
XOUT, XIN



EO



VCO



2. PROGRAM MEMORY (ROM)

Program memory stores the program to be executed by the CPU, as well as predetermined constant data.

2.1 CONFIGURATION OF PROGRAM MEMORY

Fig. 2-1 shows the configuration of program memory.

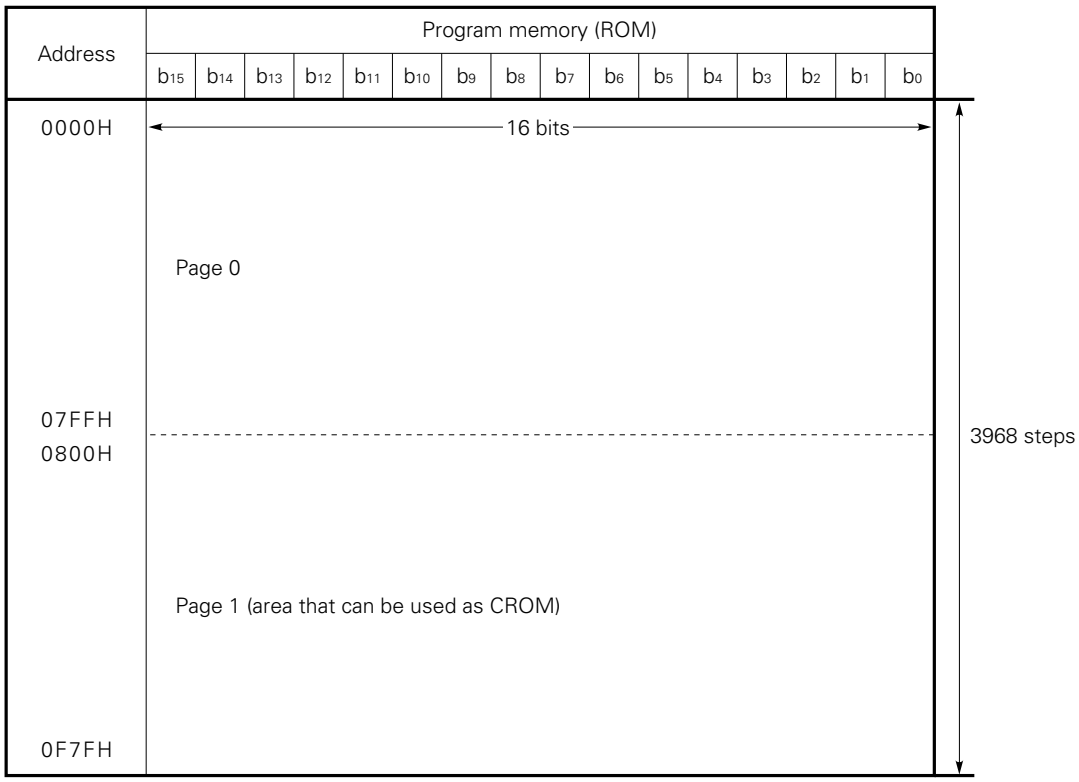
As shown in Fig. 2-1, the capacity of the program memory is 8K bytes (3968 × 16 bits).

Locations in program memory are addressed in units of 16 bits. The total address range is from 0000H to 0F7FH. Memory is divided into pages. The range of page 0 is from 0000H to 07FFH, while that of page 1 is from 0800H to 0F7FH.

The range from 0800H to 0F7FH can be used as the CROM (character ROM) area in which the display patterns for the IDC are stored. If this area is not used as CROM, it can be used as a program area.

The range from 0000H to 00FFH is a table reference area. The area is used by the JMP @AR, CALL @AR, MOVT, PUSH, and POP instructions.

Fig. 2-1 Configuration of Program Memory



2.2 FUNCTIONS OF PROGRAM MEMORY

Program memory has two basic functions:

(1) Program storage
(2) Constant data storage

A program is a set of instructions that control the CPU (Central Processing Unit: Device that actually controls the microcontroller). The CPU executes processing sequentially according to the instructions coded in the program. The CPU sequentially reads instructions from the program stored in program memory and executes processing according to each instruction.

Each instruction is one word, or 16 bits in length. A single instruction can thus be stored at a single address in program memory.

Constant data is predetermined data such as a display pattern. Constant data is read from program memory into a data buffer (DBF) in data memory (RAM) upon execution of the specialized MOVT instruction. This reading of constant data from memory is called table referencing.

Program memory is read-only storage that cannot be rewritten by the execution of an instruction. In this document, program memory and ROM (read-only memory) are synonymous.

2.3 PROGRAM FLOW

A program stored in program memory is usually executed one address at a time starting from address 0000H. If another program is to be executed upon some condition being satisfied, the program flow must be branched. To achieve this, the branch instruction (BR) is used.

If a single program is executed a number of times, the efficiency of the program memory is reduced. This problem can be solved by storing that program at a given location and calling it using the specialized CALL instruction. Such a program is called a subroutine while the usual program is called a main routine.

If a program is executed upon some condition being satisfied, independently of the current program flow, the interrupt function is used. If a predetermined condition is satisfied, the interrupt function transfers control to a specified address (vector address) irrespective of the current program flow.

These program flows are controlled by the program counter (PC), which specifies program memory addresses.

2.4 BRANCHING A PROGRAM

A program is branched by execution of the branch instruction (BR).

Fig. 2-2 illustrates the operation of the branch instruction.

Branch instructions (BR) are divided into two types. Direct branch instructions (BR addr) transfer control to a program memory address (addr) directly specified in its operand. Indirect branch instructions (BR @AR) transfer control to a program memory address specified in an address register (AR), described below.

See also **Chapter 3**.

2.4.1 Direct Branch

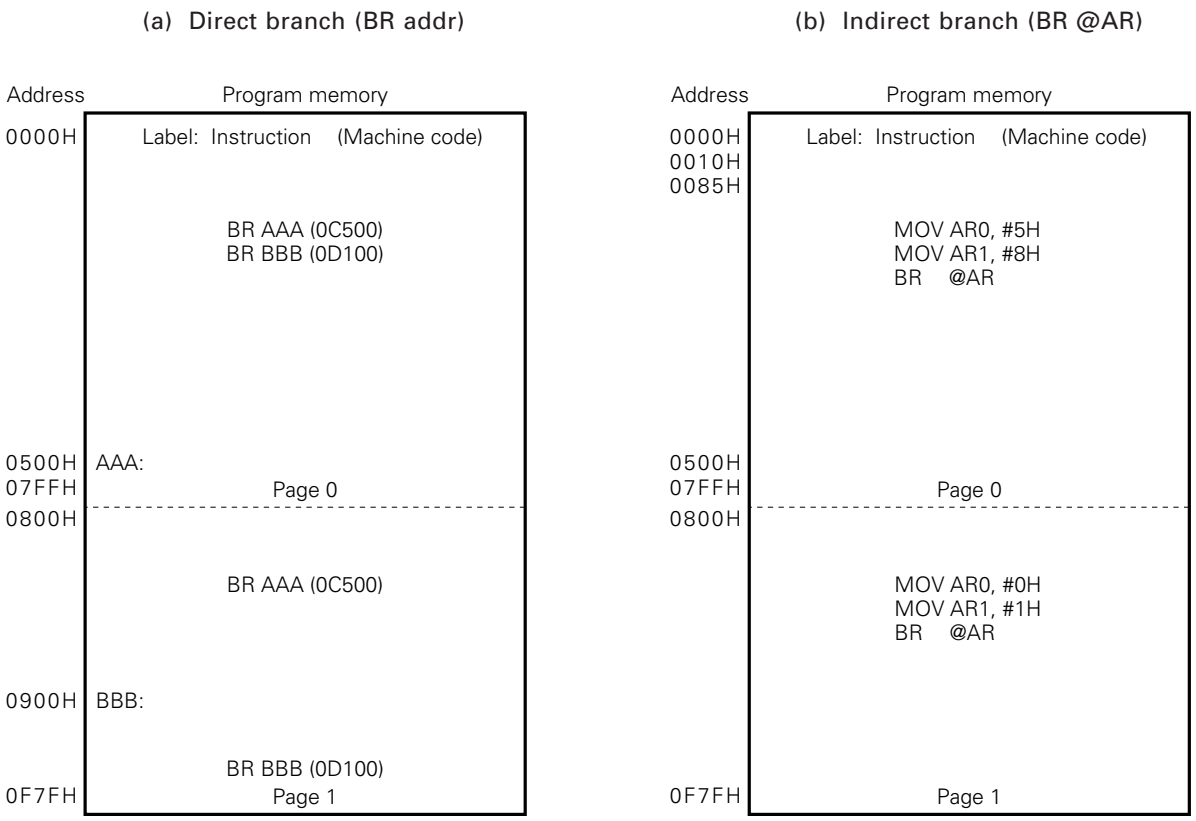
A direct branch instruction uses the least significant bit of the operation code and the 11 bits of its operand, 12 bits in total, to specify the destination program memory address. The destination of the direct branch instruction can be any address in program memory between 0000H and 0F7FH.

2.4.2 Indirect Branch

The indirect branch instruction uses the eight-bit data of an address register to specify the destination address. The destination of the indirect branch instruction is limited to addresses between 0000H and 00FFH.

See **Section 8.1**.

Fig. 2-2 Operation of Branch Instruction and Machine Code



Remark The machine code (16 bits) of the 17K series consists of five blocks, of one bit, four bits, three bits, four bits, and four bits. In this document, machine code is represented in these blocks so that it can be easily understood.

Example Machine code 0C500 → $\underset{1}{\underline{0}}\ \underset{4}{\underline{1100}}\ \underset{3}{\underline{101}}\ \underset{4}{\underline{0000}}\ \underset{4}{\underline{0000}}$

2.4.3 Notes on Debugging

Direct branch instructions to page 0 (addresses 0000H to 07FFH) and page 1 (addresses 0800H to 0F7FH) use different operation codes, as shown in Fig. 2-2.

The operation codes of the direct branch instructions to page 0 and page 1 are 0CH, and 0DH, respectively.

The difference arises because the direct branch instruction uses the addr operand, which is only 11 bits long, together with the least significant bit of the operation code, to specify the branch destination address.

When assembling a program, the 17K series assembler (AS17K) references a jump destination identified by a label and automatically converts the that instruction.

If the program is patched during debugging, the programmer must determine whether the branch destination is on page 0 or page 1 and convert the instruction into operation code 0CH or 0DH.

If address BBB in (a) of Fig. 2-2 is patched from 0900H to 0910H, for example, the machine code of the BR BBB instruction must be changed to 0D110.

2.5 SUBROUTINE

If a subroutine is executed, the specialized subroutine call instruction (CALL) and subroutine return instruction (RET, RETSK) are used.

Fig. 2-3 illustrates the operation of subroutine call.

Subroutine call instructions are divided into two types. The direct subroutine call instruction (CALL addr) calls the program memory address (addr) specified in its operand. The indirect subroutine call instruction (CALL @AR) calls the program memory address specified in an address register.

The RET or RETSK instruction is used to return control from a subroutine. The RET or RETSK instruction returns control to a program memory address next to the address at which the subroutine call instruction (CALL) was executed. Upon execution of the RETSK instruction, the first instruction after the return is executed as a no-operation instruction (NOP).

See also **Chapter 3**.

2.5.1 Direct Subroutine Call

The direct subroutine call instruction uses 11 bits of its operand to specify the program memory address to be called. If the direct subroutine call instruction is used, the destination, or the first address of the subroutine to be called, must be page 0 (addresses 0000H to 07FFH). The instruction cannot call a subroutine whose first address is in page 1 (addresses 0800H to 0F7FH).

The subroutine return instruction (RET, RETSK) can be in page 1. The CALL instruction can be in page 0 or page 1.

Examples 1. When the subroutine return instruction is in page 0

When the first address of the subroutine is in page 0, as shown in Fig. 2-4, the return address and return instruction can be in page 0 or page 1. When only the first address of the subroutine is in page 0, the CALL instruction can be used in either page. If the first address of the subroutine cannot be placed in page 0 because of programming restrictions, the method shown in example 2 can be used.

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

The branch instruction (BR) is placed in page 0, as shown in Fig. 2-4, and the desired subroutine (SUB1) is called via the BR instruction.

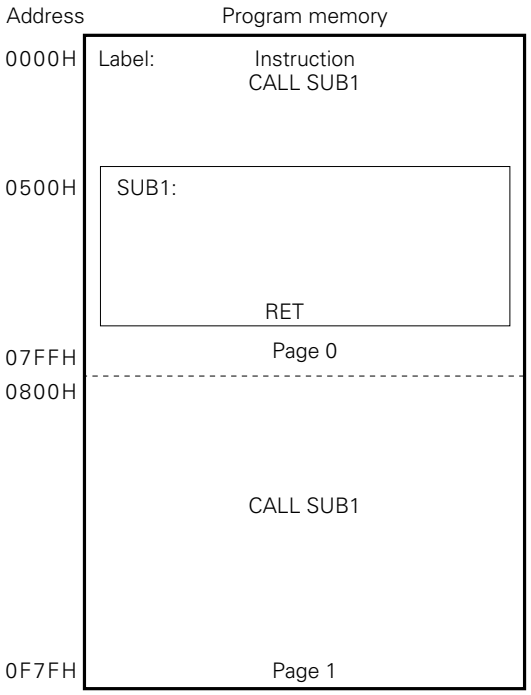
2.5.2 Indirect Subroutine Call

The indirect subroutine call instruction (CALL @AR) uses the 8-bit data in an address register (AR) to specify the address of a subroutine to be called. The instruction can call a subroutine from a program memory address between 0000H and 00FFH.

See **Section 8.1**.

Fig. 2-3 Operation of Subroutine Call Instruction

(a) Direct subroutine call (CALL addr)



(b) Indirect subroutine call (CALL @AR)

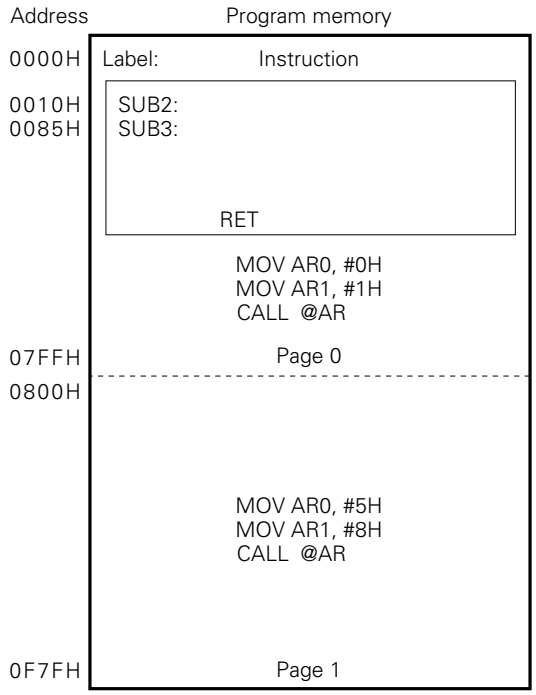
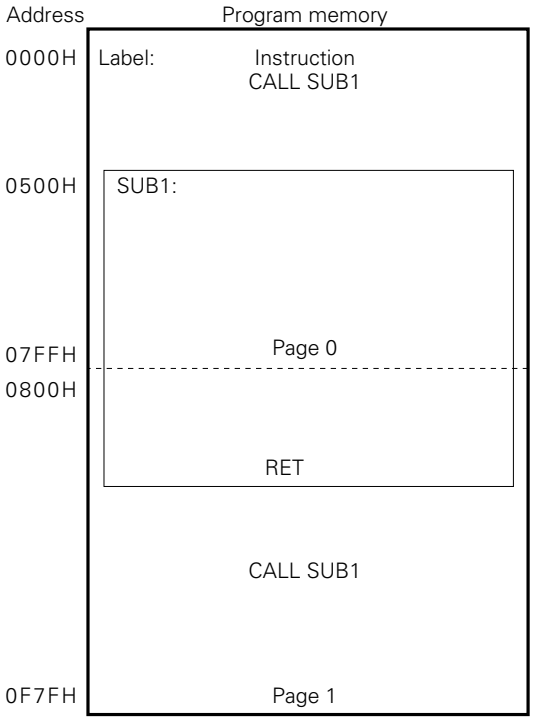
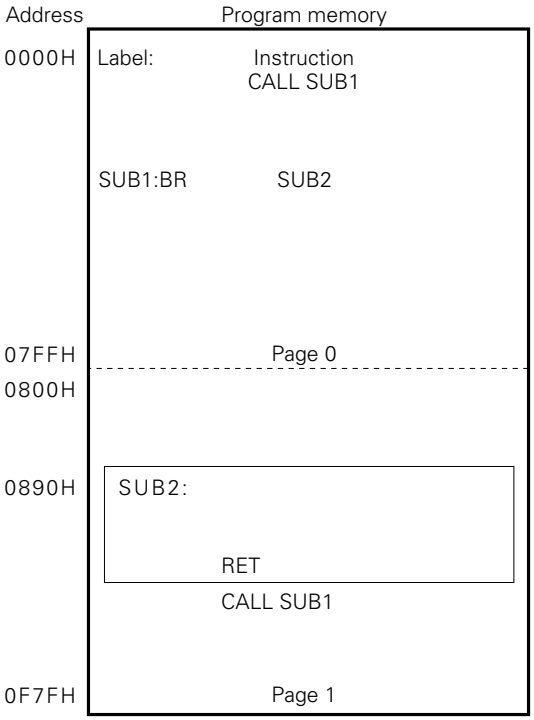


Fig. 2-4 Sample Uses of Subroutine Call Instruction

(a) If the subroutine return instruction is in page 1



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



2.6 TABLE REFERENCE

The table reference instruction is used to reference the constant data in program memory. If the MOVT DBF, @AR instruction is executed, data at the program memory address specified in an address register is placed in a data buffer (DBF).

Because each data item in program memory consists of 16 bits, the constant data placed in the data buffer by the MOVT instruction also consists of 16 bits (four words). Because the address register consists of eight bits, the MOVT instruction can reference a program memory address between 0000H and 00FFH.

When table referencing is executed, a single stack is used.

See **Sections 8.1** and **10.3**.

2.7 NOTES ON USING THE BRANCH INSTRUCTION AND SUBROUTINE CALL INSTRUCTION

The 17K series assembler (AS17K) detects an error if a program memory address (numeric address) is directly specified in the operand of the branch instruction (BR) or subroutine call instruction (CALL).

The assembler provides this function to minimize the number of bugs arising from program modification.

Examples 1. Instruction causing an error

```
; ①
        BR      0005H   ; The assembler detects the error.
; ②
        CALL    00F0H   ;
```

2. Instruction causing no error

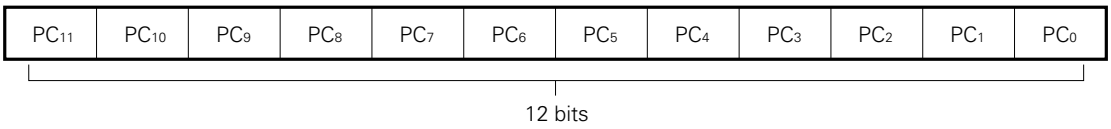
```
; ③
   LOOP1:               ; The BR or CALL instruction is executed for a label used in the
        BR      LOOP1   ; program.
; ④
   SUB1:                ;
        CALL    SUB1    ;
; ⑤
   LOOP2 LAB    0005H   ; As a label type, 0005H is assigned to LOOP2.
        BR      LOOP2   ;
; ⑥
        BR. LD. 0005H   ; The numeric value of the operand is converted to a label type.
                        ;       It is recommended that this method not be used to reduce
                        ;       the number of bugs.
```

For details, refer to the **AS17K User's Manual**.

3. PROGRAM COUNTER (PC)

The program counter addresses program memory or a program. It is a 12-bit binary counter.

Fig. 3-1 Program Counter



Normally, the program counter is incremented by 1 each time an instruction is executed. When a branch instruction or a subroutine call instruction is executed, however, the address specified in the operand field is loaded into the program counter. If a skip instruction has been executed, the address of the instruction following the skip instruction is specified, regardless of the contents of the skip instruction. If the specified address contains a skip condition, the instruction following the skip instruction is regarded as being a NOP instruction. That is, the NOP instruction is executed, and the address of the next instruction is specified.

If an interrupt request is accepted, one of addresses 1 to 4 (depending on the cause of the interrupt) is loaded into the PC.

If a power-on reset or a CE reset is performed, the program counter is reset to address 0.

Table 3-1 Vector Addresses upon Interrupt Occurrence

Priority	Interrupt cause	Vector address
1	INT_{NC} pin	4H
2	Internal timer	3H
3	$\overline{V_{SYNC}}$ pin	2H
4	Serial interface	1H

4. STACK

The stack is a register used to save an address returned by a program or the contents of the system register, described later, when a subroutine call occurs or an interrupt is accepted.

4.1 COMPONENTS

The stack consists of a stack pointer (SP), which is a 4-bit binary counter, six 13-bit address stack registers (ASRs), and two 3-bit interrupt stack registers.

4.2 STACK POINTER (SP)

The stack pointer is located at address 01H in the register file, and specifies an address stack register. The contents of the stack pointer are decremented by 1 whenever a push operation (CALL, MOVT, or PUSH instruction or interrupt acceptance) is performed, or incremented by 1 whenever a pop operation (RET, RETSK, RETI, MOVT, or POP instruction) is performed.

The high-order bit of the stack pointer is always set to 0. The stack pointer can indicate any of eight different values, 0H to 7H. However, 6H and 7H are not assigned to the stack.

Fig. 4-1 Structure of Stack Pointer

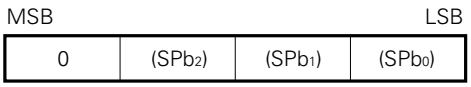


Table 4-1 Behavior of Stack Pointer

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

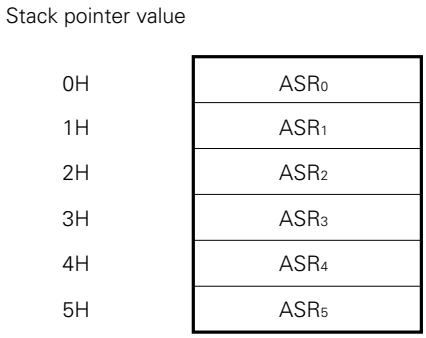
4.3 ADDRESS STACK REGISTERS (ASRs)

There are six address stack registers, each consisting of 13 bits. After a subroutine call instruction has been executed or an interrupt request accepted, the contents of the address stack register will contain a value that is equal to the contents of the program counter, plus one, or the return address. The contents of an address stack register are loaded into the program counter by executing a return instruction, after which control returns to the original program flow.

The address stack registers are used for both subroutine calls and interrupts. If two levels of the address stack registers are used for interrupts, the remaining four levels can be used for subroutine calls.

If a MOVT instruction is executed, an address stack register is used temporarily.

Fig. 4-2 Structure of Address Stack Registers



4.4 INTERRUPT STACK REGISTERS

There are two interrupt stack registers, each consisting of three bits, as shown in Fig. 4-3.

If an interrupt is accepted, the value of the two bits of the bank register (BANK) and the value of the one bit of the index-enable flag (IXE) in the system register (SYSREG), described later, are saved to an interrupt stack register. Once an interrupt return instruction (RETI) has been executed, the contents of the interrupt stack register are returned to the bank register and the index-enable flag of the system register.

Unlike the address stack registers, the interrupt stack registers contain no addresses specified by the stack pointer. As shown in Fig. 4-4, data is saved to an interrupt stack pointer each time an interrupt is accepted, the saved data being returned whenever an interrupt return instruction is executed. If accepted interrupts consist of more than two levels, the first level of data is pushed out. Thus, it must be saved by the program.

If a power-on reset is performed, the contents of the interrupt stack registers become undefined. Even if a CE reset is performed or a clock stop instruction is executed, however, the contents of the interrupt stack registers remain as is.

Fig. 4-3 Structure of Interrupt Stack Registers

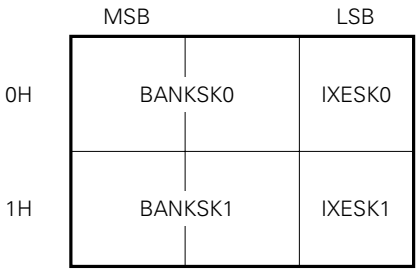
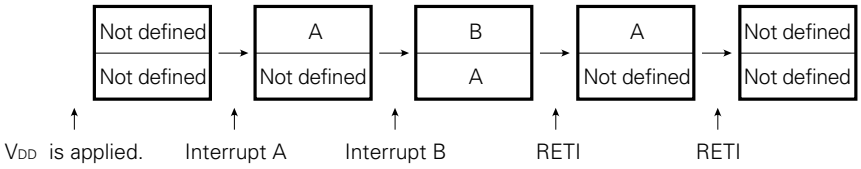


Fig. 4-4 Behavior of Interrupt Stack Registers



5. DATA MEMORY (RAM)

Data memory is used to store data for operations and control. Simply by executing an appropriate instruction, data can be written to and read from data memory at any time.

5.1 STRUCTURE OF DATA MEMORY

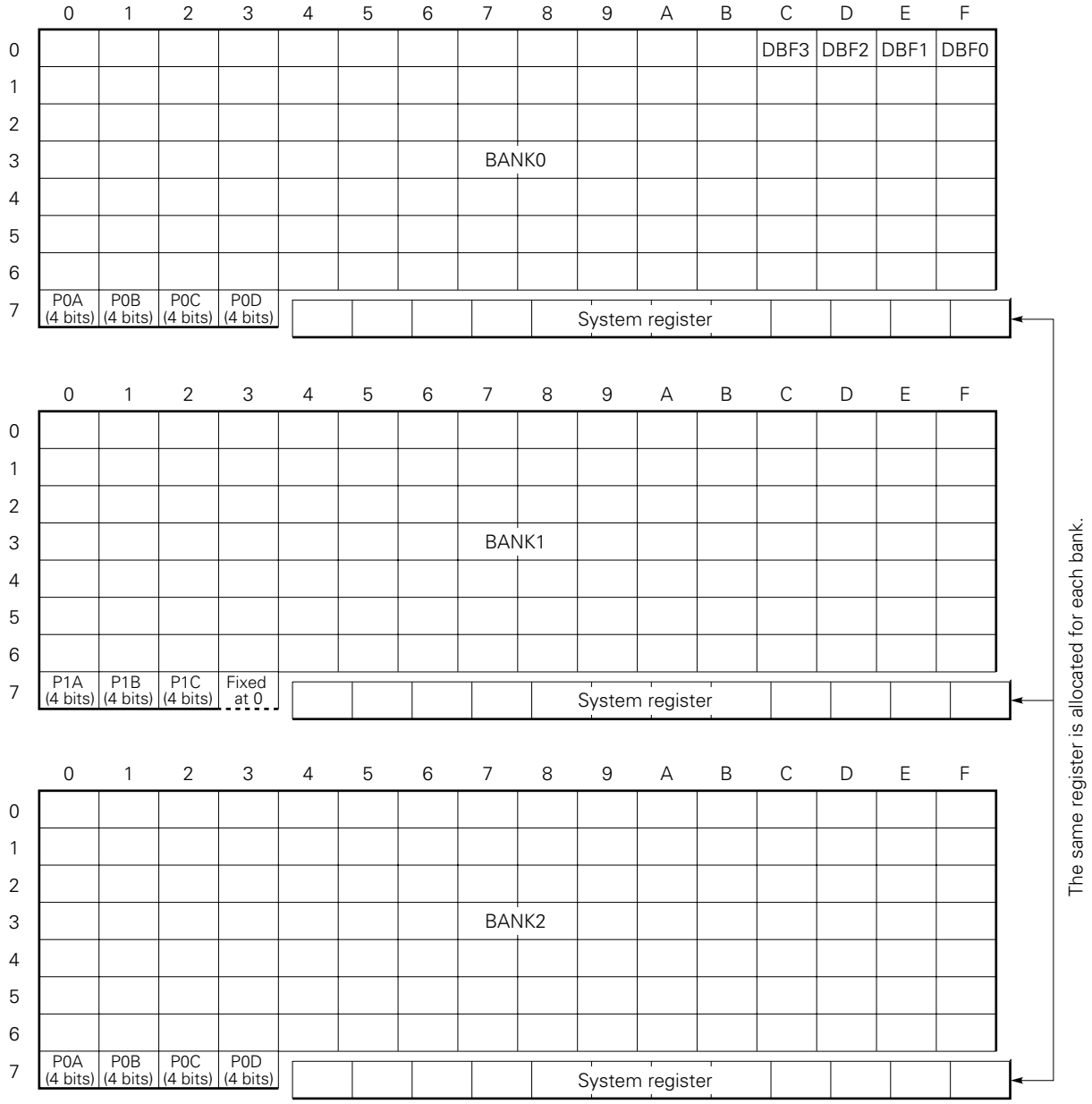
Fig. 5-1 shows the structure of data memory.

As shown in Fig. 5-1, data memory is divided into three units called banks. These three banks are called BANK0, BANK1, and BANK2.

In each bank, data is assigned an address in units of four bits. The high-order three bits are called the row address, while the low-order four bits are called the column address. For example, the data memory location having row address 1H and column address AH is referred to as the data memory location having address 1AH. One address consists of four bits of memory. These four bits are called a nibble.

Data memory is divided into the blocks described in **Sections 5.1.1** to **5.1.5**, according to function.

Fig. 5-1 Data Memory Structure



5.1.1 Structure of the System Register (SYSREG)

The system register consists of 12 nibbles, located at addresses 74H to 7FH in data memory. The system register is allocated regardless of the bank. That is, the system register is always located at addresses 74H to 7FH, regardless of the bank.

Fig. 5-2 shows the structure.

Fig. 5-2 Structure of the System Register

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

5.1.2 Structure of the Data Buffer (DBF)

The data buffer consists of four nibbles located at addresses 0CH to 0FH of BANK0 in data memory.

Fig. 5-3 shows the structure.

Fig. 5-3 Structure of the Data Buffer

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

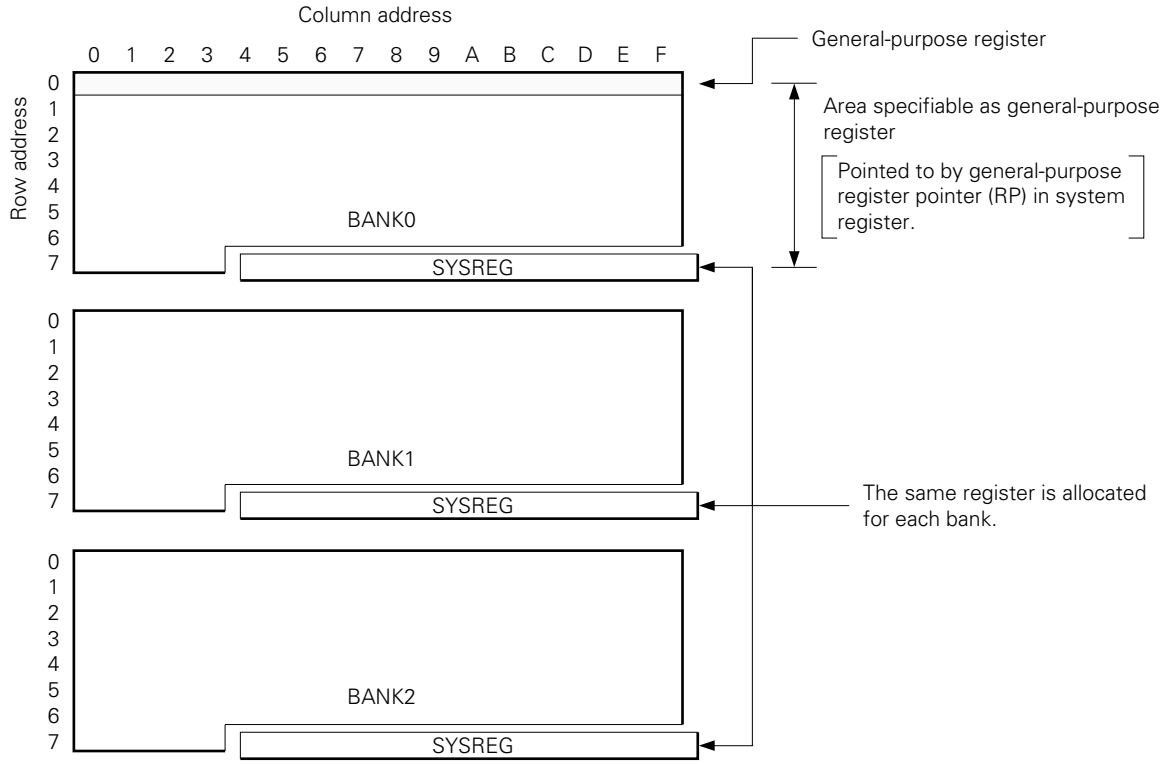
5.1.3 Structure of the General-Purpose Register (GR)

The general-purpose register consists of 12 nibbles, specified with an arbitrary row address, in data memory.

An arbitrary row address is specified using the general-purpose register pointer in the system register.

Fig. 5-4 shows the structure.

Fig. 5-4 Structure of the General-Purpose Register (GR)



5.1.4 Structure of Port Data Registers (port register)

The port registers consist of 12 nibbles at addresses 70H to 73H of the banks of data memory.

Fig. 5-5 shows the structure of the port registers.

As shown in Fig. 5-5, the same port registers are allocated in BANK0 and BANK2. Thus, the port registers actually consist of eight nibbles.

Fig. 5-5 Structure of Port Registers

Port register					
Address		70H	71H	72H	73H
Symbol	BANK0 BANK2	P0A	P0B	P0C	P0D
	BANK1	P1A	P1B	P1C	Fixed at 0

5.1.5 Structure of General-Purpose Data Memory

General-purpose data memory consists of that part of memory other than the system register and the port registers of data memory.

General-purpose data memory consists of a total of 336 words, with 112 words in each of BANK0 to BANK2.

5.1.6 Unmounted Data Memory

As shown in Fig. 5-6, nothing is assigned to bit 0 of address 72H in BANK1 of the port registers. For an explanation of this address, see **Section 5.3.2**.

5.2 FUNCTIONS OF DATA MEMORY

Data memory can be used to perform, with one instruction, a four-bit operation, comparison, decision, or transfer of the data in data memory and immediate data (arbitrary data) by executing one of the data memory manipulation instructions listed in Table 5-1.

If the general-purpose register is used, a four-bit operation, comparison, or transfer between data memory and the general-purpose register can be performed by a single instruction.

Examples are given below. See **Chapters 6** and **7** for details.

Example 1. Operation on data in data memory

```
; ①
MOV  35H, #0001B  ; Transfer (write) immediate data 0001B to data
                  ; memory address 35H in the currently selected bank.
; ②
ADD  76H, #0001B  ; Add immediate data 0001B to the contents of
                  ; data memory address 76H in the currently selected ;bank.
```

In instructions ① and ②, the currently selected bank is specified in the bank register of the system register. For an explanation of the bank register, see **Chapter 8**.
In ②, the instruction is for addition to the contents of data memory address 76H. Address 76H is part of the system register. Because the system register always exists regardless of the bank, the ADD instruction eventually adds 0001B to the contents of address 76H of the system register, regardless of the bank.

Remark For explanation of how to code instructions, see **Section 5.3.1**.

Example 2. Operation between data memory and the general-purpose register

Assume that the general-purpose register is allocated to row address 1H of BANK0.

```
; ①
ADD  7H, 36H      ; Add the contents of data memory address 36H in the
                  ; currently selected bank to the contents of the
                  ; general-purpose register location having column address
                  ; 7H, or address 17H of BANK0.
; ②
LD   7H, 36H      ; Transfer the contents of data memory address 36H to
                  ; the general-purpose register location having column
                  ; address 7H.
                  ; In this instruction, the general-purpose register
                  ; location is address 17H of BANK0.
```

The system register, data buffer, general-purpose register, and port registers can be manipulated in the same way as data memory by using the data memory manipulation instructions.

Sections 5.2.1 to **5.2.4** describe the functions of these registers.

5.2.1 Function of System Register (SYSREG)

The system register is used to control the CPU.

For example, the bank register shown in Fig. 5-2 is used to specify a data memory bank, while the general-purpose register pointer specifies the row address of the general-purpose register.

See **Chapter 8** for details.

5.2.2 Function of General-Purpose Register (GR)

The general-purpose register can be used both to perform operations on the data in data memory and to transfer data to and from data memory.

The bank and the row address for the general-purpose register are specified by the general-purpose register pointer in the system register. The general-purpose register pointer of the μPD17062 always specifies BANK0.

For example, if the general-purpose register pointer is set to 0, 16 nibbles at row address 0 of BANK0, or addresses 00H to 0FH of BANK0, are allocated as the general-purpose register.

Note that if the general-purpose register is used, transfer and arithmetic/logical instructions that involve the general-purpose register and immediate data cannot be executed. That is, the execution of a transfer or an arithmetic/logical instruction that involves the general-purpose register and immediate data requires that the general-purpose register be treated as data memory.

For example, assume that row address 0H of BANK0 is allocated as the general-purpose register (i.e., the value of the general-purpose register pointer is 0). In this case, if the currently selected bank is BANK0 (i.e., the value of the bank register is 0), executing ADD 00H, #1 increments by 1 the contents of address 00H of BANK0, which is allocated as the general register. However, if the currently selected bank is BANK1 (i.e., the value of the bank register is 1), executing ADD 00H, #1 increments by 1 the contents of address 00H of BANK1.

See **Chapter 6** for details.

5.2.3 Data Buffer (DBF)

The data buffer is used to store data to be transferred to a peripheral circuit, such as the reference voltage setting data for an A/D converter. It is also used to store data transferred from a peripheral circuit, such as input data for a serial interface.

See **Chapter 10** for details.

5.2.4 General-Purpose Port Data Registers (port registers)

Port registers are used both to store output data for general-purpose I/O ports and to read input data. The output of the pins assigned as an output port is determined by storing data into the port registers that correspond to those pins. The input status of those pins assigned as an input port can be detected by reading the contents of the port registers corresponding to those pins. Fig. 5-6 shows the correspondence between the port registers and ports (pins).

See **Chapter 15** for details.

Table 5-1 Data Memory Manipulation Instructions

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

Fig. 5-6 Correspondence Between Port Registers and Ports (Pins)

General-purpose port data register					Corresponding port	Pin	
Bank	Address	Symbol	Bit symbol			Symbol	Input or output
BANK0 BANK2	70H	P0A	b_3	P0A3	Port0A	$P0A_3$	Input and output (bit I/O)
			b_2	P0A2		$P0A_2$	
			b_1	P0A1		$P0A_1$	
			b_0	P0A0		$P0A_0$	
	71H	P0B	b_3	P0B3	Port0B	$P0B_3$	Input and output (bit I/O)
			b_2	P0B2		$P0B_2$	
			b_1	P0B1		$P0B_1$	
			b_0	P0B0		$P0B_0$	
	72H	P0C	b_3	P0C3	Port0C	$P0C_3$	Output
			b_2	P0C2		$P0C_2$	
			b_1	P0C1		$P0C_1$	
			b_0	P0C0		$P0C_0$	
	73H	P0D	b_3	P0D3	Port0D	$P0D_3$	Input
			b_2	P0D2		$P0D_2$	
			b_1	P0D1		$P0D_1$	
			b_0	P0D0		$P0D_0$	
BANK1	70H	P1A	b_3	P1A3	Port1A	$P1A_3$	Output
			b_2	P1A2		$P1A_2$	
			b_1	P1A1		$P1A_1$	
			b_0	P1A0		$P1A_0$	
	71H	P1B	b_3	P1B3	Port1B	$P1B_3$	Input and output (bit I/O)
			b_2	P1B2		$P1B_2$	
			b_1	P1B1		$P1B_1$	
			b_0	P1B0		$P1B_0$	
	72H	P1C	b_3	P1C3	Port1C	$P1C_3$	Input and output (group I/O)
			b_2	P1C2		$P1C_2$	
			b_1	P1C1		$P1C_1$	
			b_0	P1C0		–	
	73H	Fixed at 0					

5.3 NOTES ON USING DATA MEMORY

5.3.1 Addressing Data Memory

If the 17K series assembler is being used and a numeric representing a data memory address is specified directly in an operand of a data memory manipulation instruction, as shown in example 1, an error will occur.

This error occurs to facilitate the maintainability of programs and to reduce the number of causes of bugs when a program is modified. In this data sheet, however, real-address notation is used in the sample programs to make them easy to understand. When coding an actual program, refer to the assembler instruction manual.

Example 1.

Instructions that result in an error

```
; ①
MOV  2FH, #0001B       ; Address 2FH is specified directly.
; ②
MOV  0.2FH, #0001B     ; Address 2FH in BANK0 is specified directly.
```

Instructions that do not cause an error

```
; ③
M02F  MEM 0.2FH        ; Address 2FH of BANK0 is defined symbolically in
MOV   M02F, #0001B     ; M02F as a memory-type address.
; ④
MOV  .MD.2FH, #0001B   ; Address 2FH is converted into a memory-type
                       ; address by using .MD..  However, the use of this type of
                       ; instruction should be avoided to reduce the
                       ; likelihood of bugs arising.
```

Using an assembler pseudo instruction, namely the MEM instruction (symbol definition pseudo instruction), symbolically define a data memory address in advance.

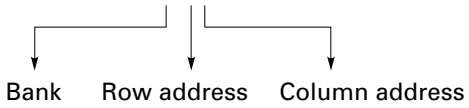
If a data memory address is defined symbolically, a data memory bank must also be specified, as shown in example 2.

This data memory bank specification is used when a data memory map is automatically created in the assembler.

Note that if a symbolically defined data memory address for BANK2 is used in the range of BANK1 in a program, as shown in example 2, the operation is performed in BANK1 data memory.

Example 2.

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




```
BANK1                   ;   Assembler built-in macro instruction  BANK ← 1
MOV   M1,       #0000B  ;   M1, M2, and M3 are defined symbolically in ① for different
MOV   M2,       #0000B  ;   banks, but are for BANK1 in this program.  Thus, all of these
MOV   M3,       #0000B  ;   three instructions write 0s to data memory address 15H in BANK1.
```

5.3.2 Notes on Using Unmounted Data Memory

As shown in Fig. 5-6, nothing is actually assigned to bit 0 (LSB) of address 72H of BANK1 of the port registers.

If a data memory manipulation instruction is executed for this address, the following operations are performed:

(1) Device behavior

If a read instruction is executed, a 0 is read.

Executing a write instruction results in no change.

(2) Assembler behavior

Normal assembly is performed.

No error occurs.

(3) Emulator (IE-17K) behavior

If a read instruction is executed, a 0 is read.

Executing a write instruction results in no change.

No error occurs.

6. GENERAL-PURPOSE REGISTER (GR)

The general-purpose register is allocated in data memory space, and is used to perform direct operations on the data in data memory and to transfer data to and from data memory.

6.1 STRUCTURE OF THE GENERAL-PURPOSE REGISTER

Fig. 6-1 shows the structure of the general-purpose register.

As shown in Fig. 6-1, 16 words (16 words × 4 bits) having the same row address in data memory space can be used as the general-purpose register.

The row address to be used as the general-purpose register can be specified using the general-purpose register pointer of the system register. The general-purpose register consists of seven bits. However, the high-order four bits are fixed to 0 so, within the data memory space, only row addresses 0H to 7H of BANK0 can be used as the general-purpose register.

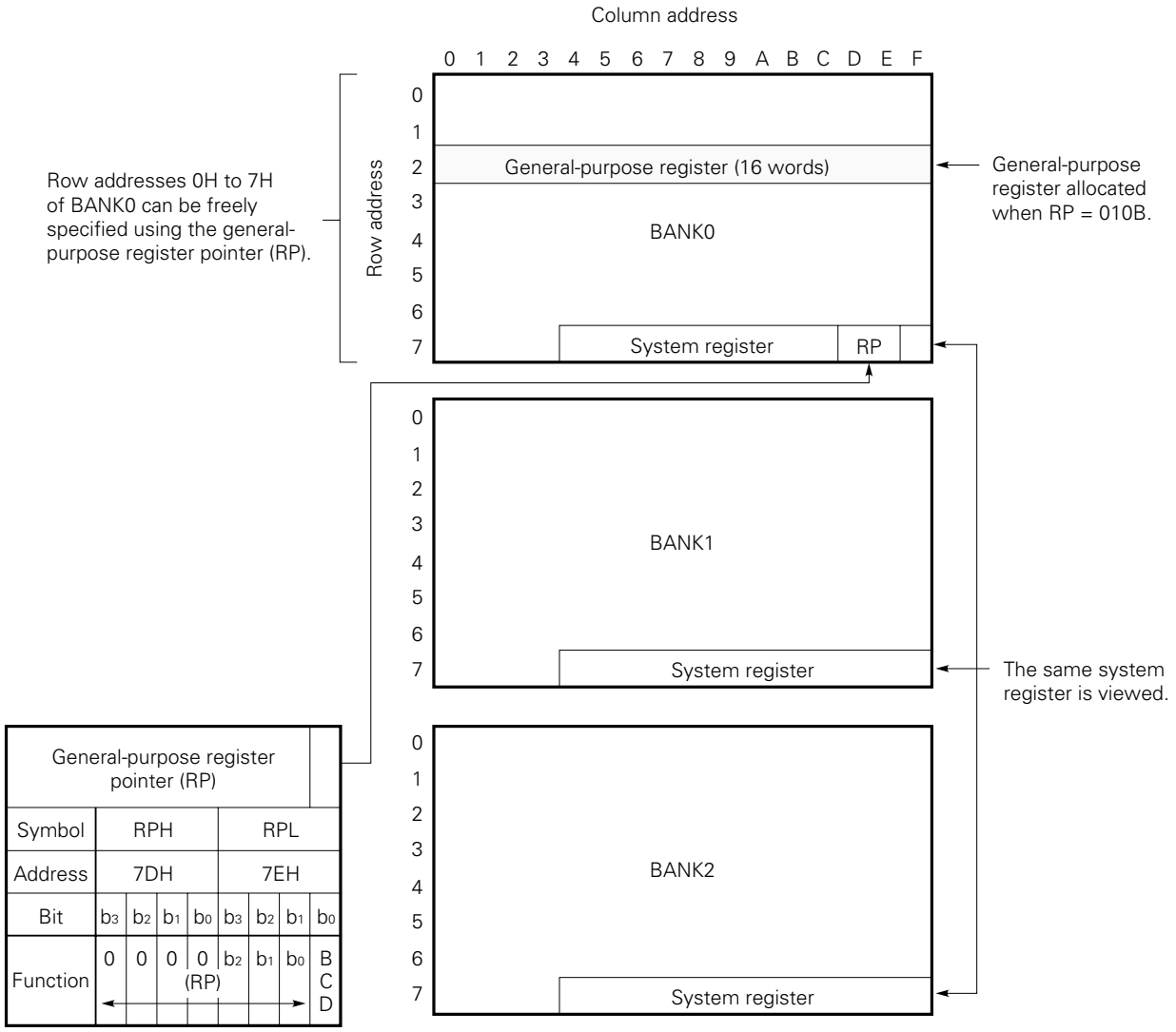
See **Section 8.6**.

6.2 FUNCTION OF THE GENERAL-PURPOSE REGISTER

The general-purpose register can be used to perform an operation or to transfer data between itself and data memory with the execution of a single instruction. The general-purpose register is allocated in data memory space. This enables an operation or transfer to be performed between data memory locations by the execution of a single instruction.

Like other data memory, the general-purpose register can be controlled using a data memory manipulation instruction.

Fig. 6-1 Structure of General-Purpose Register



6.3 ADDRESS GENERATION FOR GENERAL-PURPOSE REGISTER AND DATA MEMORY IN INDIVIDUAL INSTRUCTIONS

Table 6-1 lists the operation and transfer instructions that can be executed for the data in the general-purpose register and data memory.

Consider the following instruction:

ADD r, m ((r) ← (r) + (m))

Upon executing this instruction, the address of the general-purpose register is generated from the value of the general-purpose register pointer and the value specified in r, as shown in Table 6-2. Then, the contents of the general-purpose register specified by the generated address of the general-purpose register are added to the contents of the data memory location specified in m, the result being stored into the general-purpose register.

The address of the general-purpose register is generated, as described above, for each of the instructions listed in Table 6-1.

Table 6-1 Manipulation Instructions Executed between the General-Purpose Register and Data Memory

Instruction set	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) ∧ (m)
	OR	r, m	(r) ← (r) ∨ (m)
	XOR	r, m	(r) ← (r) ⊻ (m)
Transfer	LD	r, m	(r) ← (m)
	ST	m, r	(m) ← (r)
	MOV	@r, m	if MPE = 1: (MP, (r)) ← (m) if MPE = 0: (BANK, m_R, (r)) ← (m)
	MOV	m, @r	if MPE = 1: (m) ← (MP, (r)) if MPE = 0: (m) ← (BANK, m_R, (r))
Shift	RORC	r	Right shift, including a carry

Table 6-2 Address Generation for General-Purpose Register and Data Memory

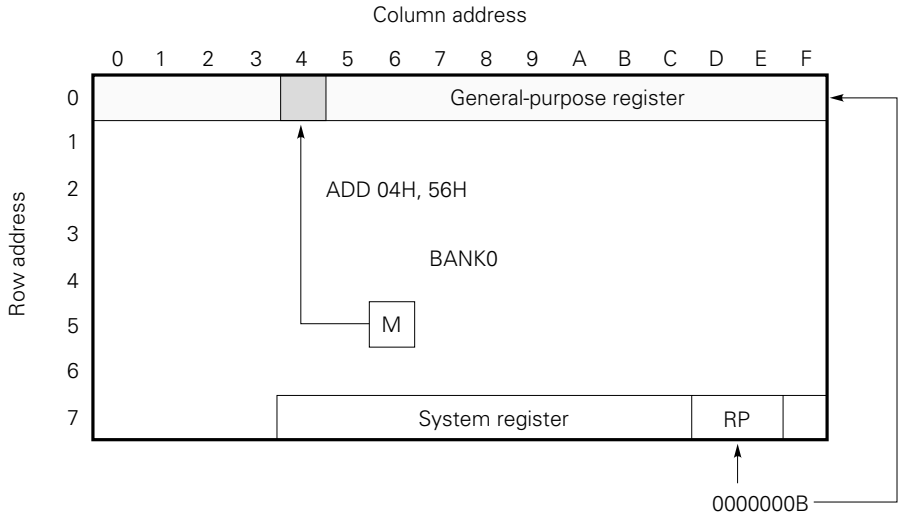
Instruction	Address	Generated address		
		Bank	Row address	Column address
ADD r, m	General-purpose register address specified in r	(0000B)	(RP)	r
	Data memory address specified in m	(BANK) (00××B)		m

Example 1. When BANK0 is selected

```
AND  RPL, #0001B   ; RP ← 0000000B; The general-purpose register is allocated in row
                   ; address 0H in BANK0.
ADD  04H, 56H      ;
```

Executing the above instruction adds the contents of address 04H of BANK0, part of the general-purpose register, to the contents of data memory address 56H, then stores the result into address 04H of the general-purpose register. See Fig. 6-2.

Fig. 6-2 Execution of Instructions in Example 1



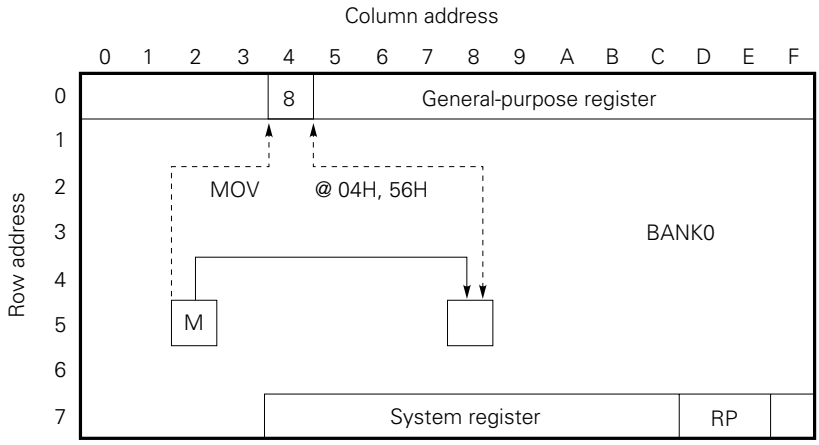
Example 2. When BANK0 is selected and MPE = 0 is specified

```
MOV  04H, #8          ; 04H ← 8
AND  RPL, #0001B      ; RP ← 0000000B; The general-purpose register is allocated in row
                      ; address 0H in BANK0.
MOV  @04H, 52H
```

Executing the above instruction transfers the contents of data memory address 52H to address 58H. The MOV @r, m instruction is called an indirect transfer of the general-purpose register contents. In this instruction, the contents of the general-purpose register address specified in r (8 in the above example) consist of the column address of data memory, and the row address specified in m (5 in the above example) is the row address of data memory. That is, the data memory address is 58H (see **Fig. 6-3**).

See **Section 8.5** for an explanation of the indirect transfer of the general-purpose register contents.

Fig. 6-3 Execution of Instructions in Example 2

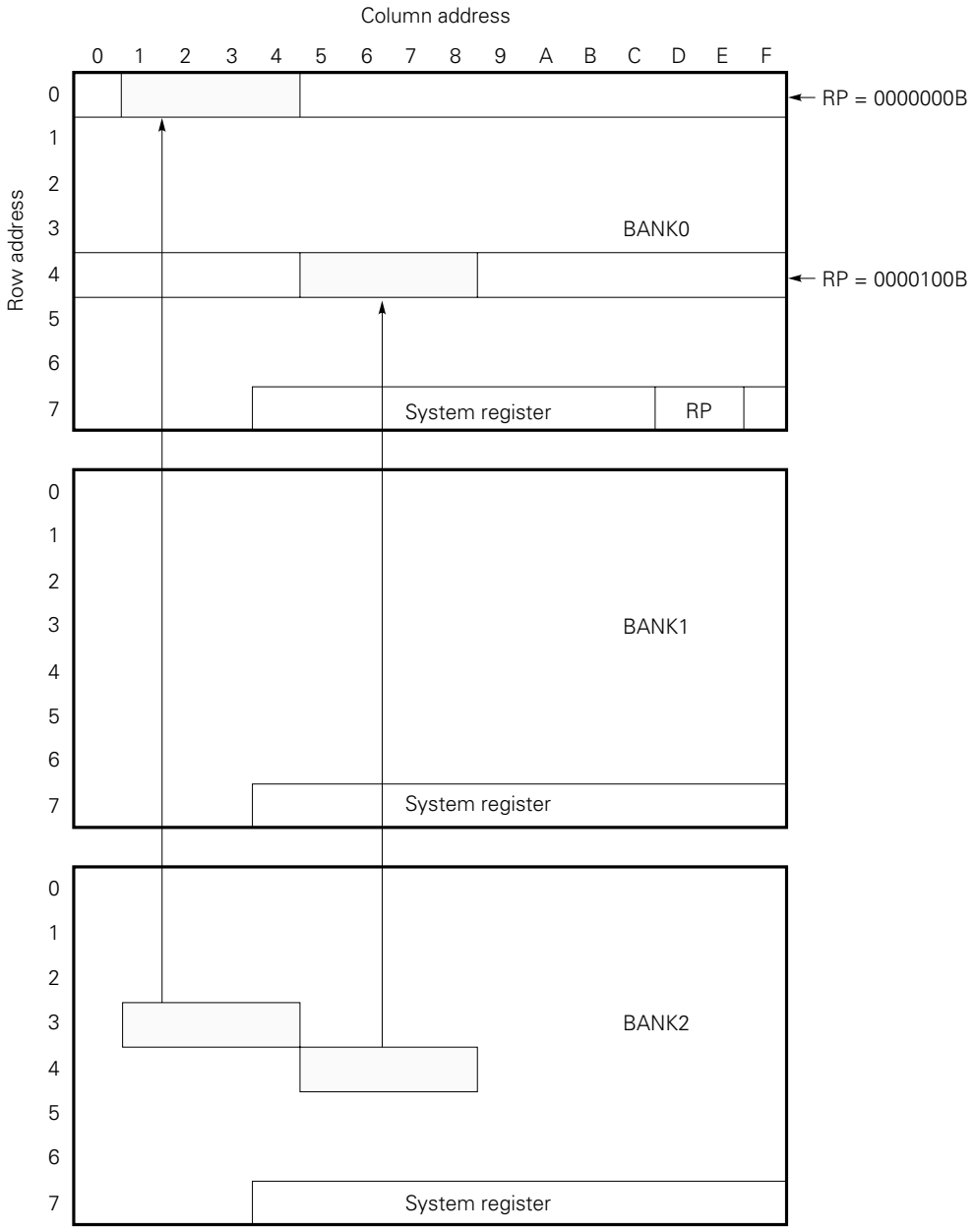


Example 3.

```
AND  RPL,    #0000B   ; RP ← 0000000B; The general-purpose register is  allocated in row
                      ; address 0H of BANK0.
MOV  BANK,   #0010B   ; BANK2
LD   01H,    31H
LD   02H,    32H
LD   03H,    33H
LD   04H,    34H
OR   RPL,    #1000B   ; RP ← 0000100B; The general-purpose register is allocated in row
                      ; address 4H of BANK0.
LD   05H,    45H
LD   06H,    46H
LD   07H,    47H
LD   08H,    48H
```

Example 3 shows a program that transfers eight words of data from BANK2 to BANK0 data memory in units of four words, as shown in Fig. 6-4. If the general-purpose register is allocated in a fixed row address, for example, only in row address 0 of BANK0, instructions are needed to transfer all of the eight words to the register and then store them into data memory. In contrast, if the row address of the general-purpose register is changed using the general-purpose register pointer as shown in example 3, the operation can be completed simply by executing a storage instruction.

Fig. 6-4 Execution of Instructions in Example 3



6.4 NOTES ON USING THE GENERAL-PURPOSE REGISTER

This section provides notes on using the general-purpose register, referring to the following example:

Example

```
AND   RPL,   #000B     ; RP ← 0000010B
OR    RPL,   #0100B    ;
MOV   BANK,  #0000B    ; BANK0
LD    04H,   32H
```

Executing the above instructions loads the contents of address 32H of BANK0 data memory into address 24H in the general-purpose register of BANK0.

In the above example, the general-purpose register is allocated in row address 2H of BANK0, so that the address of the general-purpose register specified in r in instruction LD r, m is address 24H of BANK0. The data memory address specified in m is address 32H of BANK0. (See **Fig. 6-5**.)

Note that it is necessary to code an actual data memory address, for example, 24H, as the value specified in r when using the assembler. In this case, only the low-order four bits are needed as the value for r, so the assembler ignores value 2H, which is a row address. Thus, executing instruction LD 24H, 32H produces the same result as executing the instruction in the above example.

If, when using the assembler, the address of the general-purpose register is specified directly in an operand of an instruction, as shown below, an error occurs.

Instruction that causes an error

```
LD    04H,    32H      ; The address of the general-purpose register is coded as 04.
```

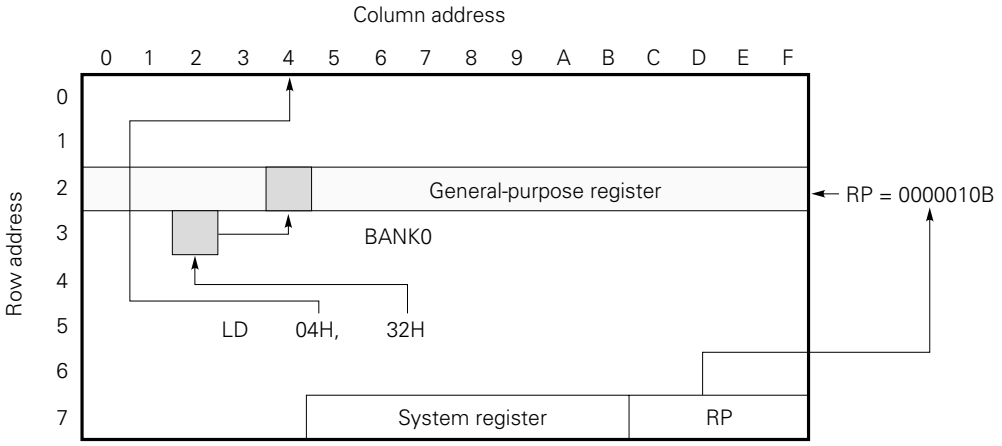
Most commonly used method

```
R1    MEM     0.04H    ;
M1    MEM     0.32H    ;   ① R1 and M1 are defined as memory-type addresses, and are
LD    R1,     M1       ;     assigned addresses 04H and 32H of BANK0, respectively.
```

Executing the following instructions produces the same result as executing the instructions in ① because R1 and R2 are assigned the same column address.

```
R2    MEM     0.34H
M1    MEM     0.32H
LD    R2,     M1
```

Fig. 6-5 Execution of the Above Example



Also, note the following when the general-purpose register is being used. No arithmetic/logical instructions are provided for the general-purpose register and immediate data. That is, the execution of an arithmetic/logical instruction that involves data memory allocated as the general-purpose register and immediate data requires that the data memory be treated as data memory rather than the general-purpose register.

7. ARITHMETIC LOGIC UNIT (ALU) BLOCK

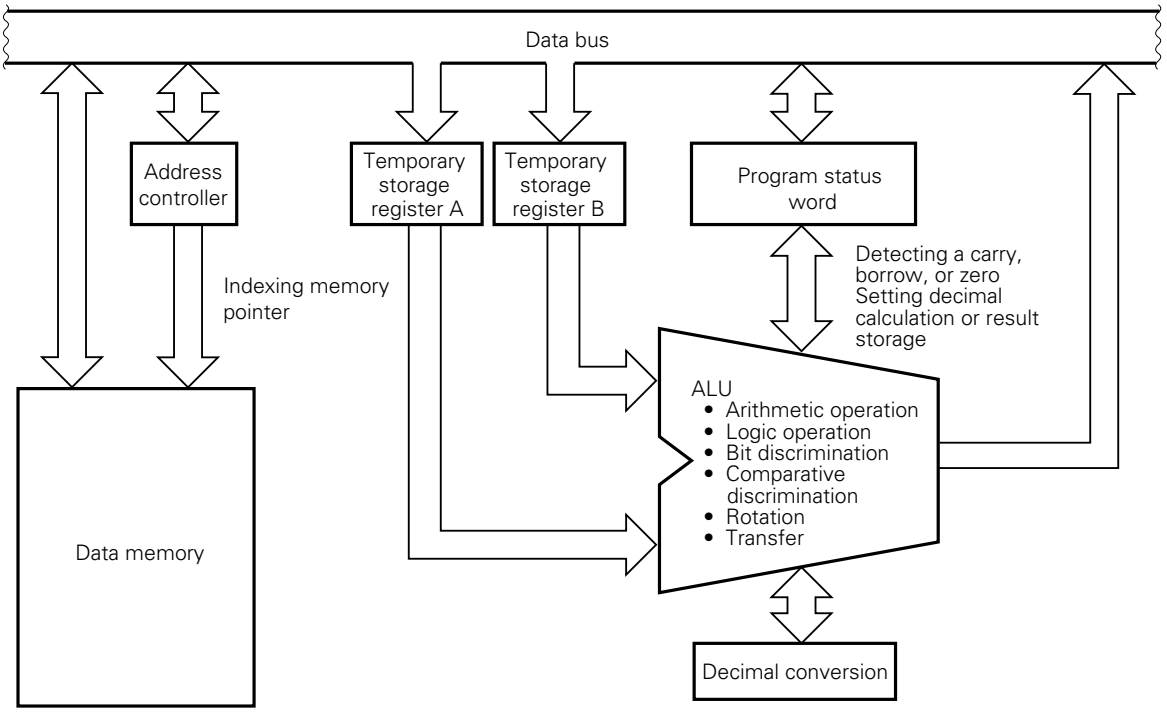
7.1 OVERVIEW

Fig. 7-1 is an overview of the ALU block.

As shown in Fig. 7-1, the ALU block consists of the ALU, temporary storage registers A and B, program status word, decimal conversion circuit, and data memory address controller.

The ALU performs arithmetic and logic operations on the 4-bit data in the data memory and performs discrimination, comparison, rotation, and transfer.

Fig. 7-1 Overview of the ALU Block



7.2 CONFIGURATION AND FUNCTIONS OF THE COMPONENTS OF THE ALU BLOCK

7.2.1 ALU

In response to a programmed instruction, the ALU performs 4-bit arithmetic or logic processing, bit discrimination, comparative discrimination, rotation, or transfer.

7.2.2 Temporary Storage Registers A and B

Temporary storage registers A and B temporarily hold the 4-bit data.

These registers are automatically used when an instruction is executed. They cannot be controlled by a program.

7.2.3 Program Status Word

A program status word controls the operation of the ALU and holds the status of the ALU.

For details of the program status word, see **Section 8.7**.

7.2.4 Decimal Conversion Circuit

If the BCD flag of the program status word is set to 1 when an arithmetic operation is executed, the decimal conversion circuit converts the results of the arithmetic operation to a decimal number.

7.2.5 Address Controller

The address controller specifies an address in data memory.

At the same time, the circuit also controls address modification by the index register or data memory row address pointer.

7.3 ALU OPERATIONS

Table 7-1 lists the operations performed by the ALU when instructions are executed.

Table 7-2 shows the data memory address modification by the index register and data memory row address pointer.

Table 7-3 lists the converted decimal data used in decimal operations.

Table 7-1 ALU Operations

ALU function	Instruction		Operation difference due to program status word (PSWORD)					Address modification	
			Value of the BCD flag	Value of the CMP flag	Operation	Operation of the CY flag	Operation of the Z flag	Index	Memory pointer
Addition	ADD	r, m	0	0	Binary operation The result is stored.	Set by a carry or borrow. Otherwise, the flag is reset.	Set if the operation result is 0000B. Otherwise, the flag is reset.	Provided	Not provided
		m, #n4							
	ADDC	r, m	0	1	Binary operation The result is not stored.		Retains the status if the operation result is 0000B. Otherwise, the flag is reset.		
		m, #n4							
Subtraction	SUB	r, m	1	0	Decimal operation The result is stored.		Set if the operation result is 0000B. Otherwise, the flag is reset.		
		m, #n4							
	SUBC	r, m	1	1	Decimal operation The result is not stored.		Retains the status if the operation result is 0000B. Otherwise, the flag is reset.		
		m, #n4							
Logic operation	OR	r, m	Optional (hold)	Optional (hold)	Not changed	Retains the previous state.	Retains the previous state.	Provided	Not provided
		m, #n4							
	AND	r, m							
		m, #n4							
	XOR	r, m							
		m, #n4							
Discrimination	SKT	m, #n	Optional (hold)	Optional (reset)	Not changed	Retains the previous state.	Retains the previous state.	Provided	Not provided
	SKF	m, #n							
Comparison	SKE	m, #n4	Optional (hold)	Optional (hold)	Not changed	Retains the previous state.	Retains the previous state.	Provided	Not provided
	SKNE	m, #n4							
	SKGE	m, #n4							
	SKLT	m, #n4							
Transfer	LD	r, m	Optional (hold)	Optional (hold)	Not changed	Retains the previous state.	Retains the previous state.	Provided	Not provided
	ST	m, r							
	MOV	m, #n4							
		@r, m							Provided
		m, @r							
Rotation	RORC	r	Optional (hold)	Optional (hold)	Not changed	Value of b_0 of the general-purpose register	Retains the previous state.	Not provided	Not provided

Table 7-2 Modification of the Data Memory Address and Indirect Transfer Address by the Index Register and Data Memory Row Address Pointer

IXE	MPE	General-purpose register address specified with r			Data memory address specified with m			Indirect transfer address specified with @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	m_R	(r)
0	1		Same as above			Same as above		MP		(r)
1	0		Same as above		BANK		m	BANK	m_R	
					Logical IX		OR	Logical IXH, IXM		OR (r)
1	1		Same as above			Same as above		MP		(r)

BANK : Bank register
IX : Index register
IXE : Index enable flag
IXH : Bits 10 to 8 of the index register
IXM : Bits 7 to 4 of the index register
IXL : Bits 3 to 0 of the index register
m : Data memory address specified with m_R and m_C
m_R : Data memory row address (high order)
m_C : Data memory column address (low order)
MP : Data memory row address pointer
MPE : Memory pointer enable flag
r : General-purpose register column address
RP : General-purpose register pointer
(×) : Contents addressed by ×

Table 7-3 Converted Decimal Data

Operation result	Hexadecimal addition		Decimal addition	
	CY	Operation result	CY	Operation result
0	0	0000B	0	0000B
1	0	0001B	0	0001B
2	0	0010B	0	0010B
3	0	0011B	0	0011B
4	0	0100B	0	0100B
5	0	0101B	0	0101B
6	0	0110B	0	0110B
7	0	0111B	0	0111B
8	0	1000B	0	1000B
9	0	1001B	0	1001B
10	0	1010B	1	0000B
11	0	1011B	1	0001B
12	0	1100B	1	0010B
13	0	1101B	1	0011B
14	0	1110B	1	0100B
15	0	1111B	1	0101B
16	1	0000B	1	0110B
17	1	0001B	1	0111B
18	1	0010B	1	1000B
19	1	0011B	1	1001B
20	1	0100B	1	1110B
21	1	0101B	1	1111B
22	1	0110B	1	1100B
23	1	0111B	1	1101B
24	1	1000B	1	1110B
25	1	1001B	1	1111B
26	1	1010B	1	1100B
27	1	1011B	1	1101B
28	1	1100B	1	1010B
29	1	1101B	1	1011B
30	1	1110B	1	1100B
31	1	1111B	1	1101B

Operation result	Hexadecimal subtraction		Decimal subtraction	
	CY	Operation result	CY	Operation result
0	0	0000B	0	0000B
1	0	0001B	0	0001B
2	0	0010B	0	0010B
3	0	0011B	0	0011B
4	0	0100B	0	0100B
5	0	0101B	0	0101B
6	0	0110B	0	0110B
7	0	0111B	0	0111B
8	0	1000B	0	1000B
9	0	1001B	0	1001B
10	0	1010B	1	1100B
11	0	1011B	1	1101B
12	0	1100B	1	1110B
13	0	1101B	1	1111B
14	0	1110B	1	1100B
15	0	1111B	1	1101B
–16	1	0000B	1	1110B
–15	1	0001B	1	1111B
–14	1	0010B	1	1100B
–13	1	0011B	1	1101B
–12	1	0100B	1	1110B
–11	1	0101B	1	1111B
–10	1	0110B	1	0000B
–9	1	0111B	1	0001B
–8	1	1000B	1	0010B
–7	1	1001B	1	0011B
–6	1	1010B	1	0100B
–5	1	1011B	1	0101B
–4	1	1100B	1	0110B
–3	1	1101B	1	0111B
–2	1	1110B	1	1000B
–1	1	1111B	1	1001B

Remark Correct decimal conversion is not possible in the shaded area.

7.4 NOTES ON USING THE ALU

7.4.1 Notes on Using the Program Status Word for Operations

After an arithmetic operation has been performed on the program status word, the operation result is held in the program status word.

The CY and Z flags of the program status word are usually set or reset according to the result of the arithmetic operation. If the arithmetic operation is performed on the program status word itself, the result of the operation is stored and a carry, borrow, or zero cannot be discriminated.

If the CMP flag is set, the result of the arithmetic operation is not stored and the CY and Z flags are set or reset as usual.

7.4.2 Notes on Performing Decimal Operations

A decimal operation can be carried out only when the operation result is within the following ranges:

(1) The result of addition is between 0 and 19 in decimal.
(2) The result of subtraction is between 0 and 9 or –10 and –1 in decimal.

If a decimal operation exceeding the above ranges is performed, the CY flag is set, resulting in a value greater than or equal to 1010B (0AH).

8. SYSTEM REGISTER (SYSREG)

"System register" is the generic name for those registers directly related to CPU control. System registers are allocated at addresses 74H-7FH in data memory and can be referenced regardless of the bank specification.

The system register types are as follows:

- Address register
- Window register
- Bank register
- Memory pointer enable flag
- Index register
- Data memory row address pointer
- General-purpose register pointer
- Program status word

Fig. 8-1 Configuration of System Register

Address	74H	75H	76H	77H	78H	79H	7AH	7BH	7CH	7DH	7EH	7FH
Register	System register											
	Address register (AR)				Window register (WR)	Bank register (BANK)	Index register (IX)			General-purpose register pointer (RP)		Program status word (PSWORD)
							Data memory row address pointer (MP)					
Symbol	AR3	AR2	AR1	AR0	WR	BANK	IXH	IXM	IXL	RPH	RPL	PSW
							MPH	MPL				
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	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	b_3 b_2 b_1 b_0	b_3 b_2 b_1 b_0
Data	0 0 0 0	0 0 0 0	← →	← →	← →	0 0	MPE 0 0 0	0 (IX) (MP)	← →	0 0 0 0	← →	BCD CMP CY Z IXE

8.1 ADDRESS REGISTER (AR)

The address register specifies a program memory address. It is located at addresses 74H-77H. The instructions used to manipulate the address register are indirect branch instructions (BR @AR, CALL @AR), the table reference instruction (MOVT), and stack manipulation instructions (PUSH, POP).

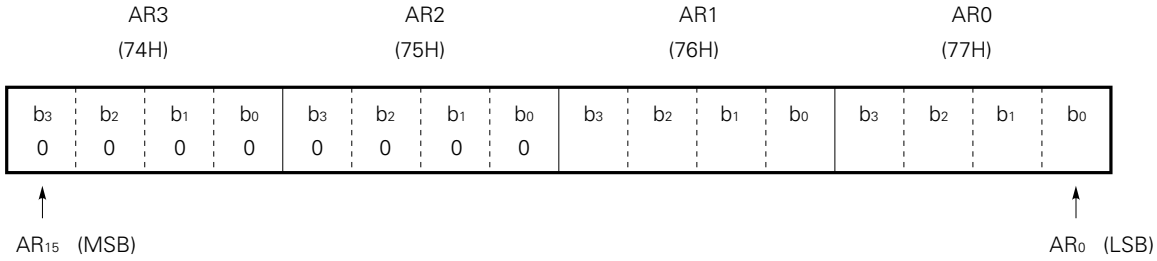
An indirect branch is a branch to the program memory address specified by the contents of the address register. Indirect branch instructions include BR @AR and CALL @AR.

Table reference is the transfer of the contents of the program memory address specified by the address register to the DBF of data memory (BANK0 0DH-0FH). This is done by executing a MOVT instruction.

Stacks are manipulated using the PUSH and POP instructions. The PUSH instruction stores the contents of the address register in the stack specified by the current stack pointer, and decrements the contents of the stack pointer by 1. The POP instruction increments the contents of the stack pointer by 1, and loads the contents of the stack specified by the current stack pointer into the address register.

AR3 and AR2 of μPD17062 are fixed at 0. Hence, the program address that can be specified by the address register is the 256 steps of 0000H-00FFH.

Fig. 8-2 Configuration of Address Register



8.2 WINDOW REGISTER (WR)

The window register is a 4-bit register, mapped to address 78H of the system register. It is used for data transfer together with the register file (RF), described later in this manual. All data in each register of the register file is manipulated via the window register.

Data transfer between the window register and the register file is achieved by execution of the exclusive PEEK WR, rf and POKE rf, WR instructions.

8.3 BANK REGISTER (BANK)

The bank register specifies a data memory bank.

The bank register contains BANK0 upon reset. The two high-order bits of address 79H are consistently set to 0.

Data memory is classified into three banks by the bank register. When a data memory manipulation instruction is executed, it acts on the data memory in the bank specified by the bank register.

For example, to manipulate BANK1 data memory with BANK0 set as the current bank, the bank must first be switched to BANK1 in the bank register.

However, system registers allocated to addresses 74H-7FH of data memory are not confined to the concept of banks. The same system registers exist at addresses 74H-7FH of all banks. Executing MOV 78H, #0 in BANK1 and MOV 78H, #0 in BANK2 both result in writing 0 to address 78H of the system register. Therefore, system register manipulation is not constrained to the concept of banks.

When an interrupt is accepted, BANK is saved.

Table 8-1 Specification of Data Memory Bank

Bank request (BANK)				Data memory bank
b_3	b_2	b_1	b_0	
0	0	0	0	BANK0
0	0	0	1	BANK1
0	0	1	0	BANK2
0	0	1	1	Not to be set

8.4 MEMORY POINTER ENABLE FLAG (MPE)

The MPE specifies whether to specify the row address for execution of the MOV @r, M and MOV M, @r instructions by the MPL, or to perform execution with the same address. When the MPE is set, the row address is specified by the MPL. When the MPE is reset, the instruction is executed with same row address.

However, the address specified by the MPL is the row address of the currently specified bank.

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

8.5.1 Configuration of Index Register and Data Memory Row Address Pointer

As shown in Fig. 8-1, the index register consists of 11 bits, including the three low-order bits, of 7AH (IXH) of the system register, 7BH, and 7CH (IXM, IXL). The index register is used to indirectly specify a data memory address.

The data memory row address pointer consists of 7 bits, including the three low-order bits of 7AH (MPH) and 7BH (MPL).

This means that the seven high-order bits of the index register and data memory row address pointer are shared.

The four high-order bits of the index register, i.e., the four high-order bits of the data memory row address pointer (7AH b_2-b_0, 7BH b_3), of μPD17062 are fixed at 0.

8.5.2 Functions of Index Register and Data Memory Row Address Pointer

When a data memory manipulation instruction is executed with the index enable flag (IXE) set to 1, the index register ORs the data memory bank/address specified by the instruction and the contents of the index register. Then, the index register executes the instruction in the data memory address indicated by the operation result (in other words, the real address).

When a general-purpose register indirect transfer instruction (MOV @r, m and MOV m, @r) is executed with the memory pointer enable flag set to 1, the data memory row address pointer executes the instruction, regarding the indirect address bank specified by the general-purpose register and row address as being the value of the data memory row address pointer.

Table 8-2 shows the modification of data memory and the indirect address by the index register and data memory row address pointer.

All data memories are subject to modification by the index register and data memory row address pointer.

The following instructions are not subject to modification by the index register.

INC	AR
INC	IX
MOVT	DBF, @AR
PUSH	AR
POP	AR
PEEK	WR, rf
POKE	rf, WR
GET	DBF, P
PUT	p, DBF
BR	addr
BR	@AR
RORC	r
CALL	addr
CALL	@AR
RET	
RETSK	
RETI	
EI	
DI	
STOP	0
HALT	h
NOP	

Table 8-2 Modification of Data Memory Address by Index Register and Data Memory Row Address Pointer

IXE	MPE	General-purpose register address specified by r R			Data memory address specified by 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)	m_R	(R)
0	1		Same as above			Same as above		(MP)		(R)
1	0		Same as above		BANK / Logical OR (IX)		m	(BANK) / Logical OR (IXH)	m_R / (IXM)	(R)
1	1		Same as above			Same as above		(MP)		(R)

		Address-modified instructions		
Addition/subtraction	ADD, ADDC, SUB, SUBC	r	m	
			m, #n4	
Logical operation	AND, OR, XOR	r	m	
			m, #n4	
Comparison	SKE, SKGE, SKLT, SKNE		m, #n4	
Discrimination	SKT, SKF		m, #n	
Transfer	LD, ST	r	m	
	MOV		m, #n4	
		@ r	m	Indirect transfer address

M ; Data memory address
(M) ; Contents of data memory address
m ; Data memory address excluding banks
m_R ; Data memory row address
R ; General-purpose register address
(R) ; Contents of general-purpose register address
r ; General-purpose register column address
RP ; General-purpose register pointer
(RP) ; Contents of general-purpose register address

BANK ; Bank register
(BANK) ; Contents of bank register
IX ; Index register
(IX) ; Contents of index register
IXH ; Bits b_{10}-b_8 of index register
IXM ; Bits b_7-b_4 of index register
IXL ; Bits b_3-b_0 of index register
MP ; Data memory row address pointer
(MP) ; Contents of data memory row address pointer

8.5.3 For MPE = 0 and IXE = 0 (Data Memory Not Modified)

As shown in Table 8-2, data memory addresses are not affected by the index register or data memory row address pointer.

Example 1. When the row address of the general-purpose register is 0 for BANK0

```
ADD    03H,    11H
```

When the above instruction is executed, the contents of general-purpose register 03H and data memory 11H are added and the result is stored in general-purpose register 03H. (See **Example 1** in **Fig. 8-3**).

Example 2. When the row address of the general-purpose register is 0 for BANK0

```
MOV    05H,    #8      ; 05H ← 8
MOV    @05H,   34H     ; Register indirect transfer
```

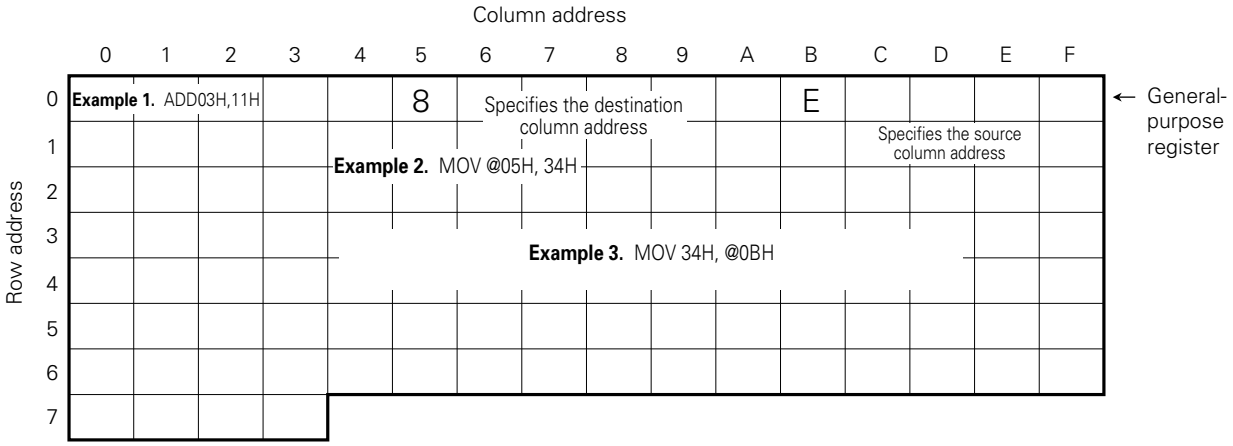
When the above instruction is executed, the contents of the data memory at address 34H are transferred to address 38H. This means that the MOV @ r, m instruction transfers the contents of data memory m to the same row address (in the above case, 3) as m and the column address (in the above case, 38H) specified by the contents (in the above case, 8) of general-purpose register r. (See **Example 2** in **Fig. 8-3**).

Example 3. When the row address of the general-purpose register is 0 for BANK0

```
MOV    0BH,    #0EH    ; 0BH ← 0EH
MOV    34H     @0BH    ; Register indirect transfer
```

When the above instruction is executed, the contents of the data memory are transferred from address 3EH to 34H. This means that the MOV m, @r instruction transfers the contents at the same row address (in the above case, 3) as data memory m and at the column address (in the above case, 3EH) specified by the contents (in the above case, 0EH) of general-purpose register r to m (See **Example 3** in **Fig. 8-3**). The (transfer) source and (transfer) destination are exactly opposite to those in example 2.

Fig. 8-3 Indirect Transfer of General-Purpose Register with MPE = 0 and IXE = 0



Address generation of example 2

MOV @r, m

05H 34H

	Bank	Row address	Column address
R	0	0	5
M	0	3	4
(@ r)	0	3	8
	Same as M		Contents of R

8.5.4 For MPE = 1 and IXE = 0 (Diagonal Indirect Transfer)

As shown in Table 8-2, the bank and row address of the data memory address in the indirect side specified by the general-purpose register are set to the value of the data memory row address pointer only when a general-purpose register indirect transfer instruction is executed.

Example 1. When the row address of the general-purpose register is 0 for BANK0

```
MOV    MPL,    #0101B    ; MP ← 00101B
MOV    MPH,    #1000B    ; MPE ← 1
MOV    05H,    #8        ; 05H ← 8
MOV    @05H,   34H       ; Register indirect transfer
```

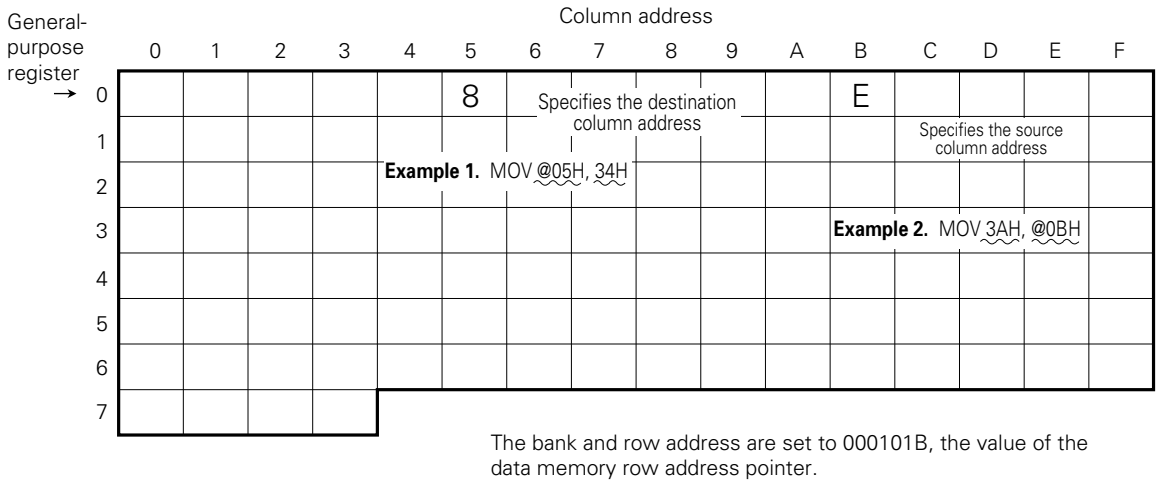
When the above instruction is executed, the contents of the data memory at address 34H are transferred to address 58H of data memory. This means that the MOV @r, m instruction at MPE = 1 transfers the contents of data memory m to the data memory whose bank and row addresses are the values of the data memory row address pointer (in the above example, BANK0, row address 5) and whose column address is specified (in the above case, 58H of BANK0) by general-purpose register r (in the above case, 8). (See **Example 1** in **Fig. 8-4**.)
Compared to MPE = 0 (**Example 2** in **Section 8.5.3**), the bank and row address of the data memory address in the indirect side specified by the general-purpose register can be specified by the data memory row address pointer (in **Example 2** of **Section 8.5.3**, the bank and row address in the indirect side are the same as those of m). Therefore, specifying MPE = 1 enables general-purpose register diagonal indirect transfer to be performed.
Similarly, the MOV m, @r instruction becomes as shown in Example 2.

Example 2. When the row address of the general-purpose register is 0 for BANK0

```
MOV    MPL,    #0101B    ; MP ← 00101B
MOV    MPH,    #1000B    ; MPE ← 1
MOV     0BH,   #0EH      ; 0BH ← 0EH
MOV     3AH,   @05H
```

(See **Example 2** in **Fig. 8-4**.)

Fig. 8-4 Indirect Transfer of General-Purpose Register with MPE = 1 and IXE = 0



Address generation of example 1

MOV @r, m
↓ ↓
05H 34H
MP = 00101B

	Bank	Row address	Column address
R	0	0	5
M	0	3	4
(@r)	0 0 0 0	1 0 1	8
	Value of MP		Contents of R

8.5.5 For MPE = 0 and IXE = 1 (Index Modification)

As shown in Table 8-2, when a data memory manipulation instruction is executed, the bank and row address of the data memory specified directly by the instruction are ORed with the index register. Then, the instruction is executed in the data memory address specified by the operation result (real address).

Example 1. When the row address of the general-purpose register is 0 for BANK0

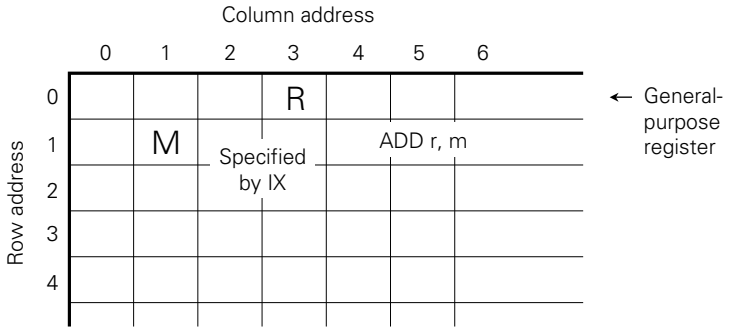
```
MOV   IXL,   #0010B   ; IX ← 000000010B
MOV   IXM,   #0000B   ; MPE ← 0
MOV   IXH,   #0000B   ;
OR    PSW,   #0001B   ; IXE ← 1
ADD   03H,   11H
```

When the above instruction is executed, the contents of the data memory at address 13H and the contents of the general-purpose register at address 03H are added and the result stored in the general-purpose register at address 03H.
This means that the ADD r, m instruction performs the OR operation on the address (in the above case, 11H of BANK0) specified by m and the index register value (in the above case, 000000010B), the result becoming the real address (in the above case, 13H of BANK0). Then, the instruction is executed at the real address. (See **Fig. 8-5**.)
Compared to IXE = 0 (**Example 1** in **Section 8.5.3**), the address of the data memory specified directly by the instruction is modified (OR operation) by the index register.

Example 2. To clear all bank data memories to 0

```
        MOV   IXL,   #0       ;
        MOV   IXM,   #0       ; IX ←  0
        MOV   IXH,   #0       ;
LOOP:
        OR    PSW,   #0001B   ; IXE ← 1
        MOV   00H,   #0       ; Sets data memory specified by IX to 0.
        INC   IX              ; IX ← IX + 1
        AND   PSW,   #1110B   ; IXE ← 0; IXE is not modified by IX because the address
                              ; is 7FH.
        SKT   IXM,   #0111B   ; Is row address 7 reached?
        BR    LOOP            ; LOOP if not 7
        ADD   IXM,   #1       ; Specifies the next bank without clearing row address 7.
        ADDC  IXH,   #0       ;
        SKF   IXM,   #1000B   ; Were banks cleared up to BANK2?
        SKT   IXH,   #0001B   ;
        BR    LOOP            ; LOOP unless cleared
```

Fig. 8-5 Data Memory Address Modification with IXE = 1



8.6 GENERAL-PURPOSE REGISTER POINTER (RP)

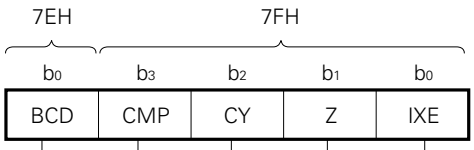
The general-purpose register pointer points to the bank and row address of the general-purpose register.

However, since RPH of the μPD17062 is fixed at 0, only RPL (3 bits) can be specified. This means that 0 to 7 can be specified as a register pointer. Hence, in the μPD17062, the row address of the general-purpose register can be specified anywhere within BANK0.

8.7 PROGRAM STATUS WORD (PSWORD)

The program status word consists of a flag that indicates the result of operation by the ALU in the CPU and a 5-bit flag that modifies the ALU function. PSWORD has a binary coded decimal (BCD) flag, compare (CMP) flag, carry (CY) flag, zero (Z) flag, and index enable (IXE) flag. Fig. 8-6 shows the functions of these flags.

Fig. 8-6 Configuration of PSWORD



Index enable flag	When this flag is set, index modification is enabled.
Zero flag	When the arithmetic operation result is other than 0, this flag is reset. The set condition differs according to the contents of the CMP flag. (1) When CMP = 0 The flag is set when the arithmetic operation result is 0. (2) When CMP = 1 The flag is set when the result of the arithmetic operation executed at Z = 1 is 0.
Carry flag	The carry flag is set when a carry occurs during the execution of an addition instruction or when a borrow occurs during the execution of a subtraction instruction. This flag is reset when neither carry nor borrow occurs. This flag is set when the least significant bit of the general-purpose register is 1, in which the RORC instruction is executed. The flag is reset when the bit is 0.
Compare flag	When this flag is set, the arithmetic operation result is not stored into data memory. The CMP flag is reset automatically when the SKT or SKF instruction is executed.
BCD flag	When this flag is set, all arithmetic operations are executed in decimal. When this flag is not set, all arithmetic operations are executed in binary.

9. REGISTER FILE (RF)

The register file is a group of registers that mainly control the CPU peripheral circuits. The register file has a capacity of 128 words × 4 bits. However, peripheral circuit addresses are actually allocated to the high-order 64 nibbles (00H-3FH) and addresses 40H-7FH of the currently selected bank of data memory to the low-order 64 nibbles (40H-7FH).

This means that 40H-7FH of each bank of data memory belongs to both the data memory address space and the register file address space.

In the assembler, the control register file is allocated to 80H-BFH.

Fig. 9-1 Configuration of Control Register (1/2)

Column Address / Row Address	Item	0	1	2	3	4	5	6	7
0 (8)Note	Register	IDCDMA enable register	Stack pointer (SP)						CE pin level judge register
	Symbol	0 0 IDCDMAEN 0	0 (SP2) (SP1) (SP0)						0 0 0 CE
	Read/ Write	R/W	R/W						R
1 (9)Note	Register		HSYNC-counter-gate control register	HSYNC-counter-gate judge register	PLL reference clock select register		INTNC mode select register		Basic timer 0 carry flip-flop judge register
	Symbol		0 0 HSCGT1 HSCGT0	HSCGOSTT 0 0 0	PLLRFCK3 PLLRFCK2 PLLRFCK1 PLLRFCK0		0 INTNCMD2 INTNCMD1 INTNCMD0		0 0 0 BTM0CY
	Read/ Write		R/W	R	R/W		R/W		R
2 (A)Note	Register		A/D converter control register	PLL-unlock-flip-flop judge register					Port 1C group I/O select register
	Symbol		ADCCH2 ADCCH1 ADCCH0 ADCCMP	0 0 0 PLLUL					0 0 0 P1CGIO
	Read/ Write		R/W	R					R/W
3 (B)Note	Register	IDC CROM bank register	IDC enable register	PLL-unlock-flip-flop sensibility select register			Port 1B bit I/O select register	Port 0B bit I/O select register	Port 0A bit I/O select register
	Symbol	0 0 0 CROMBNK	0 0 0 IDCEN	0 0 PLULSEN1 PLULSEN0			P1BBIO3 P1BBIO2 P1BBIO1 P1BBIO0	P0BBIO3 P0BBIO2 P0BBIO1 P0BBIO0	P0ABIO3 P0ABIO2 P0ABIO1 P0ABIO0
	Read/ Write	R/W	R/W	R /W			R/W	R/W	R/W

Note The number in parenthesis is the address used when the assembler (AS17K) is used.

Fig. 9-1 Configuration of Control Register (2/2)

8	9	A	B	C	D	E	F
Serial I/O0 mode select register	Timer 0 clock select register						Interrupt-level judge register
SIO0CH | SB | SIO0MS | SIO0TX	BTM0ZX | BTM0CK2 | BTM0CK1 | BTM0CK0						0 | INTVSYN | 0 | INTNC
R/W	R/W						R
Serial I/O0 wait control register							Interrupt edge selection register
SBACK | SIO0NWT | SIO0WRQ1 | SIO0WRQ0							0 | IEGVSYN | 0 | IEGNC
R/W							R/W
Serial I/O0 status judge register							Interrupt enable register
SIO0SF8 | SIO0SF9 | SBSTT | SBBSY							IPSIO0 | IPVSYN | IPBTM0 | IPNC
R							R/W
Serial I/O0 interrupt mode register	Serial I/O0 clock select register						Interrupt request register
0 | 0 | SIO0IMD1 | SIO0IMD0	0 | 0 | SIO0CK1 | SIO0CK0						IRQSIO0 | IRQVSYN | IRQBTM0 | IRQNC
R/W	R/W						R

Table 9-1 Peripheral Hardware Control Functions of Control Registers (1/5)

Peripheral hardware	Control register				Peripheral hardware control function			At reset		
	Register	Address	Read/write	b3 b2 b1 b0 Symbol	Function outline	Set value 0	Set value 1	Power On	STOP	CE
Stack	Stack pointer (SP)	01H	R/W	0	Fixed at 0			7	7	7
				(SP2)	Stack pointer (3 bits are valid.)					
				(SP1)						
				(SP0)						
Timer	Timer 0 clock select register	09H	R/W	BTM0ZX	On/off of zerocross circuit	No operation	Operation	0	0	*
				BTM0CK2	Base clock setting of basic timer 0 (internal/external)	Pulse for timer carry flop-flop set 0: 10 Hz (100 ms, internal) 1: 200 Hz (5 ms, internal) 2: 10 Hz (100 ms, internal) 3: 200 Hz (5 ms, internal) 4: f_{TMIN}/5 Hz (external) 5: 200 Hz (5 ms, internal) 6: f_{TMIN}/6 Hz (external) 7: 200 Hz (5 ms, internal)				
				BTM0CK1		Pulse for timer interrupt 0: 200 Hz (5 ms, internal) 1: 10 Hz (100 ms, internal) 2: 50 Hz (20 ms, internal) 3: 50 Hz (20 ms, internal) 4: 200 Hz (5 ms, internal) 5: f_{TMIN}/5 Hz (external) 6: 200 Hz (5 ms, internal) 7: f_{TMIN}/6 Hz (external)				
				BTM0CK0						
	Basic timer 0 carry flip-flop judge register	17H	R	0	Fixed at 0			0	1	1
				0						
				0						
				BTM0CY	Detects the carry flip-flop state	Reset	Set			
Interrupt	Interrupt-level judge register	0FH	R	0	Fixed at 0			0	0	0
				INTVSYN	Detects the V_{SYNC} pin state	Low level	High level			
				0	Fixed at 0					
				INTNC	Detects the INT_{NC} pin state	Low level	High level			
	INT_{NC} mode select register	15H	R/W	0	Fixed at 0			0	0	0
				INTNCMD2	Selects the pulse width of interrupt accept pulse width of the INT_{NC} pin	0: Accepts with edge 1: 200 μs 2: 400 μs 3: 2 ms 4: 4 ms				
				INTNCMD1						
				INTNCMD0						

Remark *: Retains the previous state.

Table 9-1 Peripheral Hardware Control Functions of Control Registers (2/5)

Peripheral hardware	Control register: Register	Address	Read/write	b3 b2 b1 b0 Symbol	Peripheral hardware control function: Function outline	Set value: 0	Set value: 1	At reset: Power On	At reset: STOP	At reset: CE
Interrupt	Interrupt edge select register	1FH	R/W	0	Fixed at 0			0	0	0
				IEGVSYN	Sets the interrupt issue edge (V_{SYNC})	Rising edge	Falling edge			
				0	Fixed at 0					
				IEGNC	Sets the interrupt issue edge (INT_{NC})	Rising edge	Falling edge			
	Interrupt permission register	2FH	R/W	IPSIO0	- Serial interface 0	Interrupt disabled	Interrupt enabled	0	1	1
				IPVSYN	- V_{SYNC} signal					
				IPBTM0	- Basic timer 0					
				IPNC	- INT_{NC} pin					
	Interrupt request register	3FH	R	IRQSIO0	- Serial interface 0	No interrupt request/ processing in progress	Interrupt request made	0	0	0
				IRQVSYN	- V_{SYNC} signal					
				IRQBTM0	- Basic timer 0					
				IRQNC	- INT_{NC} pin					
Pin	CE pin level judge register	07H	R	0	Fixed at 0			0	–	–
				0						
				0						
				CE	Detects the CE pin state	Low level	High Level			
PLL frequency synthesizer	PLL reference clock select register	13H	R/W	PLLRFCK3	Fixed at 1	2: 6.25 kHz 3: 12.5 kHz 6: 25 kHz F: Operation stopped (disabled state) 0, 1, 4, 5, 7-E: Setting disabled		F	F	*
				PLLRFCK2						
				PLLRFCK1						
				PLLRFCK0						
	PLL unlock flip-flop judge register	22H	R	0	Fixed at 0			0	*	*
				0						
				0						
				PLLUL	Detects the unlock flip-flop state	Locked state	Unlocked state			
	PLL unlock flip-flop sensibility select register	32H	R/W	0	Fixed at 0			0	0	*
				0						
				PLULSEN1	Sets the set delay time for the unlock flip-flop	0 0 1 1 1.25 to 1.5 μs / 3.5 to 3.75 μs / 0.25 to 0.5 μs / Disabled state 0 1 0 1				
				PLULSEN0						

Remark *: Retains the previous state.

Table 9-1 Peripheral Hardware Control Functions of Control Registers (3/5)

Peripheral hardware	Control register: Register	Ad-dress	Read/ write	b3 b2 b1 b0 Symbol	Peripheral hardware control function: Function outline	Set value: 0	Set value: 1	At reset: Power On	At reset: STOP	At reset: CE
A/D converter	A/D converter controll register	21H	R/W	ADCCH2	Selects the pin used as an A/D converter	0: AD0	1: AD1	0	0	0
				ADCCH1		2: AD2	3: AD3			
				ADCCH0		4: AD4	5: AD5			
						6, 7: Not to be set				
				ADCCMP	Detects the comparison result	$V_{IN} < V_{REF}$	$V_{IN} > V_{REF}$	**	*	*
General-purpose port	Port 1C group I/O select register	27H	R/W	0	Fixed at 0			0	0	0
				0						
				0						
				P1CGIO	Sets I/O of port 1C (group I/O)	Input	Output			
	Port 1B bit I/O select register	35H	R/W	P1BBIO3	$P1B_3$ pin	Input	Output	0	0	0
				P1BBIO2	$P1B_2$ pin					
				P1BBIO1	$P1B_1$ pin					
				P1BBIO0	$P1B_0$ pin					
	Port 0B bit I/O select register	36H	R/W	P0BBIO3	$P0B_3$ pin					
				P0BBIO2	$P0B_2$ pin					
				P0BBIO1	$P0B_1$ pin					
				P0BBIO0	$P0B_0$ pin					
	Port 0A bit I/O select register	37H	R/W	P0ABIO3	$P0A_3$ pin					
				P0ABIO2	$P0A_2$ pin					
				P0ABIO1	$P0A_1$ pin					
				P0ABIO0	$P0A_0$ pin					
Serial interface	Serial I/O0 mode select register	08H	R/W	SIO0CH	Sets the number of communication lines	2-wire method	3-wire method	0	0	0
				SB	Sets the communication method	Serial I/O method	I^2C bus method (only for 2-wire method)			
				SIO0MS	Sets master/slave	Master operation	Slave operation			
				SIO0TX	Sets the transfer direction	Reception	Transmission			
	Serial I/O0 wait control register	18H	R/W	SBACK	Sets and detects acknowledge (I^2C bus method)	Sets and detects 0 and 1		0	0	0
				SIO0NWT	Sets the wait permission	Permitted	Released			
				SIO0WRQ1	Sets the wait mode	0 0: No wait	0 1: Data wait			
				SIO0WRQ0		1 0: Acknowledge wait	1 1: Address wait			

Remark *: Retains the previous state. **: Indefinite

Table 9-1 Peripheral Hardware Control Functions of Control Registers (4/5)

Peripheral hardware	Control register				Peripheral hardware control function			At reset		
	Register	Address	Read/write	b3 b2 b1 b0 Symbol	Function outline	Set value 0	Set value 1	Power On	STOP	CE
Serial interface	Serial I/O0 status judge register	28H	R	SIO0SF8	Detects the contents of clock counter	Resets when the contents of the clock counter become 0 or 1	Resets when the contents of the clock counter become 8	0	0	0
				SIO0SF9		Resets when the contents of the clock counter become 0 or 1	Resets when the contents of the clock counter become 9			
				SBSTT	Detects the number of clocks (I²C bus method)	Sets up the start condition - 9th clock				
				SBBSY	Detects the start condition (I²C bus method)	Sets up the start condition - stop condition				
	Serial I/O0 interrupt mode register	38H	R/W	0	Fixed at 0			**	*	*
				0						
				SIO0IMD1	Sets the interrupt condition of serial interface 0	0 0: 7th clock	1 0: 8th clock			
				SIO0IMD0		0 1: 7th clock after occurrence of start condition	1 1: Stop condition			
	Serial I/O0 clock select register	39H	R/W	0	Fixed at 0			**	*	*
				0						
				SIO0CK1	Sets the internal clock of serial interface 0	0 0: 100 kHz	1 0: 500 kHz			
				SIO0CK0		0 1: 200 kHz	1 1: 1 MHz			
Horizontal synchronizing signal counter	HSYNC counter gate control register	11H	R/W	0	Fixed at 0			0	0	0
				0						
				HSCGT1	Controls the HSYNC counter gate	0 0: Gate close	1 0: 1.69 ms gate open			
				HSCGT0		0 1: Gate open	1 1: Not to be set			
	HSYNC counter gate judge register	12H	R	HSCGOSTT	Detects open/close of the HSYNC counter	Gate close	Gate open	0	–	–
				0	Fixed at 0					
				0						
				0						

Remark *: Retains the previous state. **: Indefinite

Table 9-1 Peripheral Hardware Control Functions of Control Registers (5/5)

Peripheral hardware	Control register				Peripheral hardware control function			At reset		
	Register	Address	Read/write	b3 b2 b1 b0 Symbol	Function outline	Set value 0	Set value 1	Power On	STOP	CE
IDC	IDC DMA enable register	00H	R/W	0	Fixed at 0			0	0	0
				0						
				IDCDMAEN	Sets the DMA mode permission	Not permitted	Permitted			
				0	Fixed at 0					
	IDC CROM bank register	30H	R/W	0	Fixed at 0			0	0	0
				0						
				0						
				CROMBNK	Selects the CROM bank	BANK0 (0800H-0BFFH)	BANK1 (0C00H-0F7FH)			
	IDC enable register	31H	R/W	0	Fixed at 0			0	0	0
				0						
				0						
				IDCEN	Turns the IDC display on/off	Display on	Display off			

9.1 IDCDMAEN (00H, b_1)

This flag must be set to enable the operation of IDC.

When the IDCDMAEN flag is set, the mode changes to DMA mode and IDC is enabled. In DMA mode, the instruction cycle is seen as 12 μs. For details, see **Chapter 20**.

00H

b_3	b_2	b_1	b_0
0	0	IDCDMAEN	0

0	DMA prohibited mode (instruction cycle = 2 μs)
1	DMA mode (instruction cycle = 12 μs)

9.2 SP (01H)

SP is a pointer that addresses the stack register.

01H

b_3	b_2	b_1	b_0
0	(SPb_2)	(SPb_1)	(SPb_0)

SP (stack pointer)

0	0	0	Level 6
0	0	1	Level 5
0	1	0	Level 4
0	1	1	Level 3
1	0	0	Level 2
1	0	1	Level 1
1	1	0	At reset
1	1	1	Not to be set

9.3 CE (07H, b_0)

CE is a flag for reading the CE pin level.

The flag indicates 1 when a high level signal is input to the CE pin, or 0 when a low level signal is input.

07H

b_3	b_2	b_1	b_0
0	0	0	CE

0	CE pin low level
1	CE pin high level

9.4 SERIAL INTERFACE MODE REGISTER (08H)

08H

b_3	b_2	b_1	b_0
SIO0CH	SB	SIO0MS	SIO0TX

Transmission/reception setting

0	2-wire bus mode : RX (reception) mode CH0 serial I/O mode : SI mode CH1 serial I/O mode : $P0A_3$ used as a general-purpose port
1	2-wire bus mode : TX (transmission) mode CH0 serial I/O mode : SO mode CH1 serial I/O mode : $P0A_3$ used as an SO pin

Setting of serial interface clock direction

0	2-wire bus mode : Slave operation Serial I/O mode : External clock operation
1	2-wire bus mode : Master operation Serial I/O mode : Internal clock operation

Setting of serial interface mode

0	Serial I/O mode
1	2-wire bus mode

Setting of serial interface channel

0	Selects CH0
1	Selects CH1

9.5 BTM0MD (09H)

09H

b_3	b_2	b_1	b_0
BTM0ZX	BTM0CK2	BTM0CK1	BTM0CK0

Time base setting

			TIMER INT		TIMER CARRY	
0	0	0	5 ms	Internal	100 ms	Internal
0	0	1	100 ms	Internal	5 ms	Internal
0	1	0	20 ms	Internal	100 ms	Internal
0	1	1	20 ms	Internal	5 ms	Internal
1	0	0	5 ms	Internal	$5/f_{TMR}$ s	External
1	0	1	$5/f_{TMR}$ s	External	5 ms	Internal
1	1	0	5 ms	Internal	$6/f_{TMR}$ s	External
1	1	1	$6/f_{TMR}$ s	External	5 ms	Internal

Zerocross setting

0	Zerocross off
1	Zerocross on

9.6 INTVSYN (0FH, b_2)

The INTVSYN flag is used for reading the vertical synchronous signal level. When a high level signal is input to the $\overline{V_{SYNC}}$ pin, the flag is set to 1. When a low level signal is input to the $\overline{V_{SYNC}}$ pin , the flag is reset to 0.

9.7 INTNC (0FH, b_0)

The INT_{NC} flag is used for reading the INT_{NC} pin state.

The flag indicates 1 when a high level signal is input to the INT_{NC} pin, and 0 when a low level signal is input to the INT_{NC} pin.

0FH

b_3	b_2	b_1	b_0
0	INTVSYN	0	INTNC

0	The INT_{NC} pin is low level.
1	The INT_{NC} pin is high level.

0	The $\overline{V_{SYNC}}$ pin is low level.
1	The $\overline{V_{SYNC}}$ pin is in the high level period.

9.8 HORIZONTAL SYNCHRONIZING SIGNAL COUNTER CONTROL (11H, 12H)

11H

b_3	b_2	b_1	b_0
HSCGT3	HSCGT2	HSCGT1	HSCGT0

Setting of horizontal synchronizing signal counter

0	0	Gate close
0	1	Gate open
1	0	Gate open (1.69 ms interval)
1	1	Not to be set

Both bits are fixed at 0.

12H

b_3	b_2	b_1	b_0
HSCGOSTT	0	0	0

Input confirmation of gate open/close of horizontal synchronizing signal counter

0	Gate close
1	Gate open

9.9 PLL REFERENCE MODE SELECTION REGISTER (13H)

13H

b_3	b_2	b_1	b_0
PLLRFCK3	PLLRFCK2	PLLRFCK1	PLLRFCK0

Reference frequency f_r setting

0	0	1	0	6.25 kHz
0	0	1	1	12.5 kHz
0	1	1	0	25 kHz
1	1	1	1	PLL disabled
0	1	1	1	Not to be set
1	0	1	0	
1	0	1	1	
1	1	1	0	

Fixed at 1

9.10 SETTING OF INT_{NC} PIN ACCEPTANCE PULSE WIDTH (15H)

15H

b_3	b_2	b_1	b_0
INTNCMD3	INTNCMD2	INTNCMD1	INTNCMD0

Setting of INT_{NC} pin acceptance pulse width

0	0	0	Edge (no noise canceler)
0	0	1	200 μs
0	1	0	400 μs
0	1	1	2 ms
1	0	0	4 ms

Fixed at 0

9.11 TIMER CARRY (17H)

17H

b_3	b_2	b_1	b_0
0	0	0	BTM0CY

Exclusive flag for reading timer carry
This flag is set according to the selected time base, and reset when the timer carry is read.

9.12 SERIAL INTERFACE WAIT CONTROL (18H)

18H

b_3	b_2	b_1	b_0
SBACK	SIO0NWT	SIO0WRQ1	SIO0WRQ0

Setting of wait timing

		2-wire bus mode	Serial I/O mode
0	0	Does not wait	Does not wait
0	1	Waits when the clock falls with the contents of the clock counter being 8	Waits when the contents of the clock counter become 9
1	0	Waits when the clock falls with the contents of the clock counter being 9	Waits when the contents of the clock counter become 9
1	1	Waits when the clock falls with the contents of the clock counter being 8 after detection of the start condition	Not to be set

Wait setting

0	Forced wait
1	Wait released

Acknowledgement at 2-wire bus mode

9.13 IEGNC (1FH)

The IEGNC flag is used for selecting the interrupt detection edge of the INT_{NC} pin and $\overline{V_{SYNC}}$ pin.

When the flag is set to 0, an interrupt occurs at a rising edge. When the flag is set to 1, an interrupt occurs at a falling edge.

1FH

b_3	b_2	b_1	b_0
0	IEGVSYN	0	IEGNC

0	Interrupt occurs at the rising edge of the INT_{NC} pin
1	Interrupt occurs at the falling edge of the INT_{NC} pin

0	Interrupt occurs at the rising edge of the $\overline{V_{SYNC}}$ pin
1	Interrupt occurs at the falling edge of the $\overline{V_{SYNC}}$ pin

9.14 A/D CONVERTOR CONTROL (21H)

21H

b_3	b_2	b_1	b_0
ADCCH2	ADCCH1	ADCCH0	ADCCMP

A/D converter input channel select

0	0	0	ADC_0 select
0	0	1	ADC_1 select, shared with $P1C_3$
0	1	0	ADC_2 select, shared with $P0D_0$
0	1	1	ADC_3 select, shared with $P1D_1$
1	0	0	ADC_4 select, shared with $P0D_2$
1	0	1	ADC_5 select, shared with $P0D_3$
1	1	0	No corresponding channel (not to be set)
1	1	1	

9.15 PLL UNLOCK FLIP-FLOP JUDGE REGISTER (22H)

22H

b_3	b_2	b_1	b_0
0	0	0	PLLUL

Detects the unlock flip-flop state

0	Unlock flip-flop = 0: PLL locked
1	Unlock flip-flop = 1: PLL unlocked

9.16 PORT1C I/O SETTING (27H)

27H

b_3	b_2	b_1	b_0
0	0	0	P1CGIO

P1C port I/O setting

0	$P1C_1$, $P1C_2$, $P1C_3$: input port
1	$P1C_1$, $P1C_2$, $P1C_3$: output port

9.17 SERIAL I/O0 STATUS REGISTER (28H)

28H

b_3	b_2	b_1	b_0
SIO0SF8	SIO0SF9	SBSTT	SBBSY

Busy condition detection

0	Detects the stop condition
1	Detects the start condition

Start condition detection

0	Resets when the contents of the clock counter become 9
1	Detects the start condition

9 clock detection

0	Resets when the contents of the clock counter become 0 or 1
1	Sets when the contents of the clock counter become 9

8 clock detection

0	Resets when the contents of the clock counter become 0 or 1
1	Sets when the contents of the clock counter become 8

9.18 INTERRUPT PERMISSION FLAG (2FH)

This flag is used to enable interrupt for each interrupt cause. When the flag is set to 1, interrupt is enabled. When the flag is set to 0, interrupt is disabled.

2FH

b_3	b_2	b_1	b_0
IPSIO0	IPVSYN	IPBTM0	IPNC

0	Interrupt from the INT_{NC} pin disabled
1	Interrupt from the INT_{NC} pin enabled

0	Interrupt from the clock timer disabled
1	Interrupt from the clock timer enabled

0	Interrupt from the $\overline{V_{SYNC}}$ pin disabled
1	Interrupt from the $\overline{V_{SYNC}}$ pin enabled

0	Interrupt from the serial interface disabled
1	Interrupt from the serial interface enabled

9.19 CROM BANK SELECTION (30H)

30H

b_3	b_2	b_1	b_0
0	0	0	CROMBNK

CROM address setting

0	CROM address 0800H-0BFFH
1	CROM address 0C00H-0F7FH

9.20 IDCEN (31H)

31H

b_3	b_2	b_1	b_0
0	0	0	IDCEN

0	IDC operation prohibited (display off)
1	IDC operation start (display on)

9.21 PLL UNLOCK FLIP-FLOP DELAY CONTROL REGISTER (32H)

32H

b_3	b_2	b_1	b_0
PLULSEN3	PLULSEN2	PLULSEN1	PLULSEN0

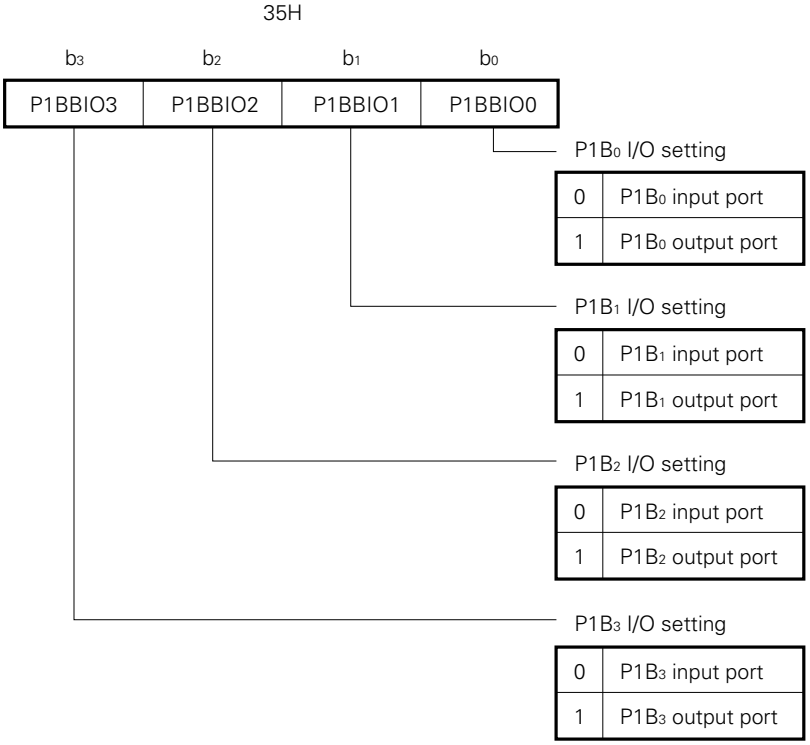
Setting of the delay time of the reference frequency f_r and divided frequency f_N required for setting the unlock flip-flop

0	0	1.25 to 1.5 μs or more
0	1	3.5 to 3.75 μs or more
1	0	0.25 to 0.5 μs or more
1	1	Unlock flip-flop disable (always set)

Fixed at 0

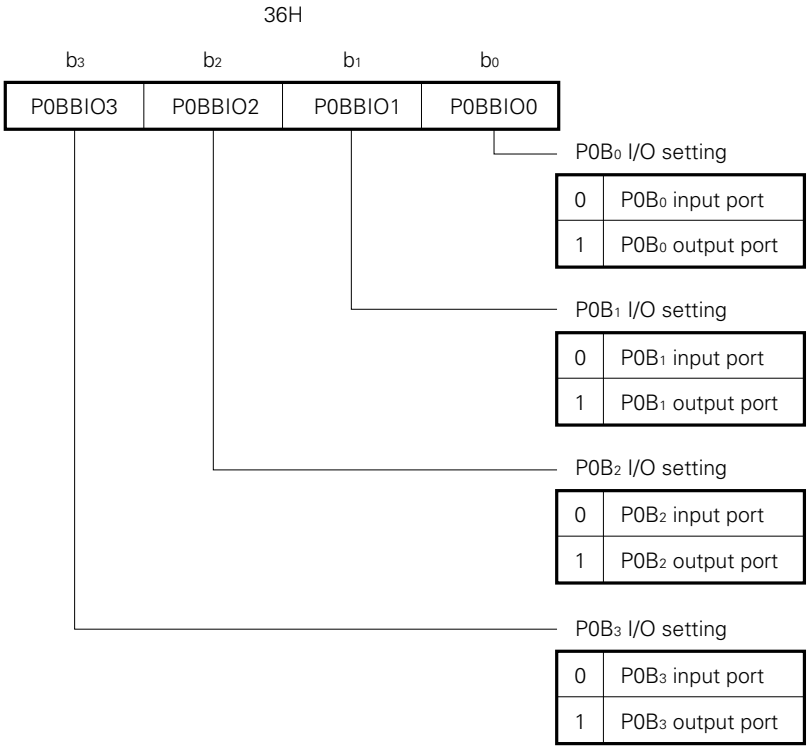
9.22 P1BBIOn (35H)

P1BBIOn specifies the PORT1B I/O. When P1BBIOn is set to 0, PORT1B becomes an input port. When P1BBIOn is set to 1, PORT1B becomes an output port.



9.23 P0BBIOn (36H)

P0BBIOn specifies the PORT0B I/O. When P0BBIOn is set to 0, PORT0B becomes an input port. When P0BBIOn is set to 1, PORT0B becomes an output port.



9.24 P0ABIOn (37H)

P0ABIOn specifies the PORT0A I/O. When P0ABIOn is set to 0, PORT0A becomes an input port. When P0ABIOn is set to 1, PORT0A becomes an output port.

37H

b_3	b_2	b_1	b_0
P0ABIO3	P0ABIO2	P0ABIO1	P0ABIO0

$P0A_0$ I/O setting

0	$P0A_0$ input port
1	$P0A_0$ output port

$P0A_1$ I/O setting

0	$P0A_1$ input port
1	$P0A_1$ output port

$P0A_2$ I/O setting

0	$P0A_2$ input port
1	$P0A_2$ output port

$P0A_3$ I/O setting

0	$P0A_3$ input port
1	$P0A_3$ output port

9.25 SETTING OF INTERRUPT REQUEST GENERATION TIMING IN SERIAL INTERFACE MODE (38H)

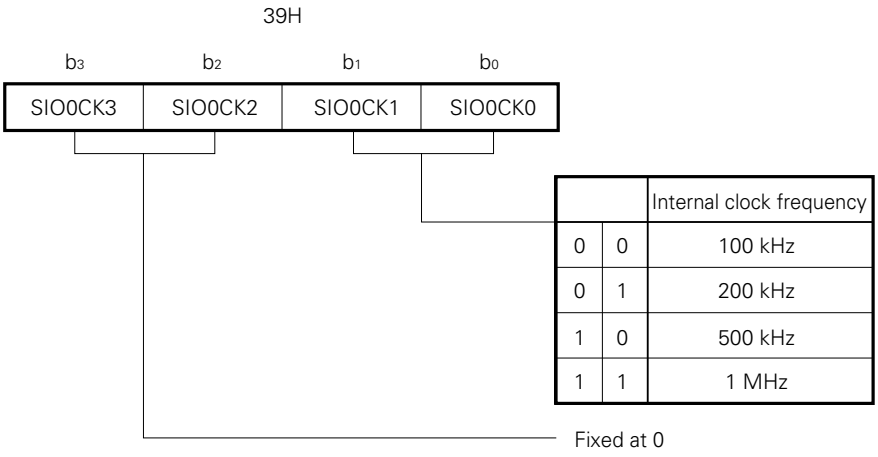
38H

b_3	b_2	b_1	b_0
SIO0IMD3	SIO0IMD2	SIO0IMD1	SIO0IMD0

Fixed at 0

		Function
0	0	Interrupt request generated at rising edge of the 7th bit of the shift clock
0	1	Interrupt request generated at rising edge of the 8th bit of the shift clock
1	0	Interrupt request generated at rising edge of the 7th bit of the shift clock immediately after the start condition is detected
1	1	Interrupt request generated when the stop condition is detected

9.26 SHIFT CLOCK FREQUENCY SETTING (39H)



		Internal clock frequency
0	0	100 kHz
0	1	200 kHz
1	0	500 kHz
1	1	1 MHz

9.27 IRQNC (3FH)

IRQNC is an interrupt request flag that indicates the interrupt request state.

When an interrupt request is generated, the flag is set to 1. When the request is accepted (interrupt is made), the flag is reset to 0.

The interrupt request flag can be read and written by the program. Hence, if 1 is written, an interrupt by software can be generated. If 0 is written, the interrupt hold status can be released. The IRQNC flag becomes 0 upon reset.

Flag name	Bit position	Interrupt source
IRQNC	b_0	INT_{NC} pin
IRQBTM0	b_1	Clock timer
IRQVSYN	b_2	$\overline{V_{SYNC}}$ pin
IRQSIO0	b_3	Serial interface

10. DATA BUFFER (DBF)

The data buffer is used to transfer data to and from peripheral hardware and to reference tables.

10.1 DATA BUFFER STRUCTURE

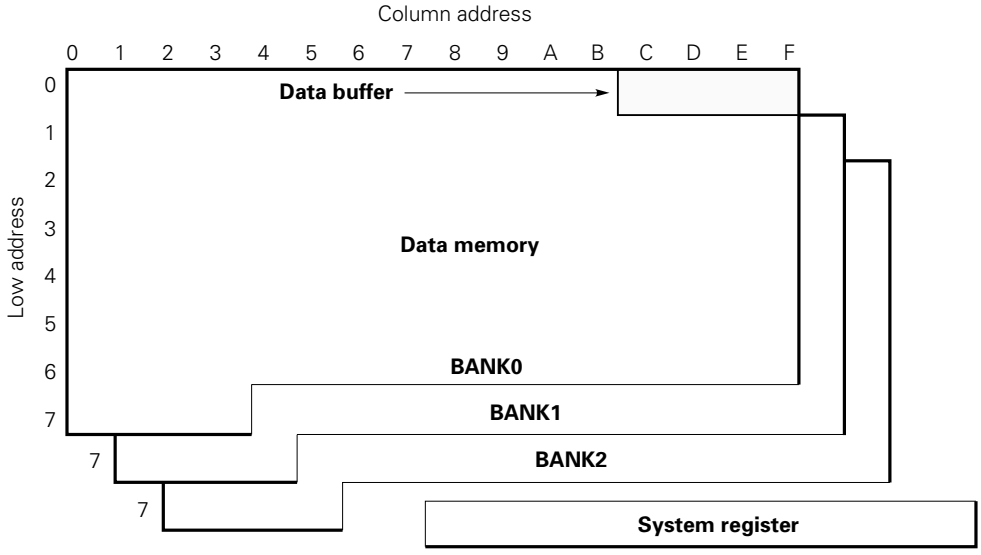
10.1.1 Mapping of Data Buffer to Data Memory

Fig. 10-1 shows how the data buffer is mapped to data memory.

As shown in Fig. 10-1, the data buffer is allocated to addresses 0CH to 0FH of data memory BANK0 and consists of 16 bits in a 4-word × 4-bit configuration.

Because the data buffer is mapped to data memory, it can be operated by data memory instructions.

Fig. 10-1 Data Buffer Map



10.1.2 Data Buffer Structure

Fig. 10-2 shows the data buffer structure.

As shown in Fig. 10-2, the data buffer consists of 16 bits. Bit b_0 of data memory address 0FH is the LSB, and bit b_3 of data memory address 0CH bit 3 is the MSB.

Fig. 10-2 Data Buffer Structure

Data memory	Address	0CH				0DH				0EH				0FH			
	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	DBF3				DBF2				DBF1				DBF0			
	Data	MSB															LSB
		Data															

10.2 FUNCTIONS OF DATA BUFFER

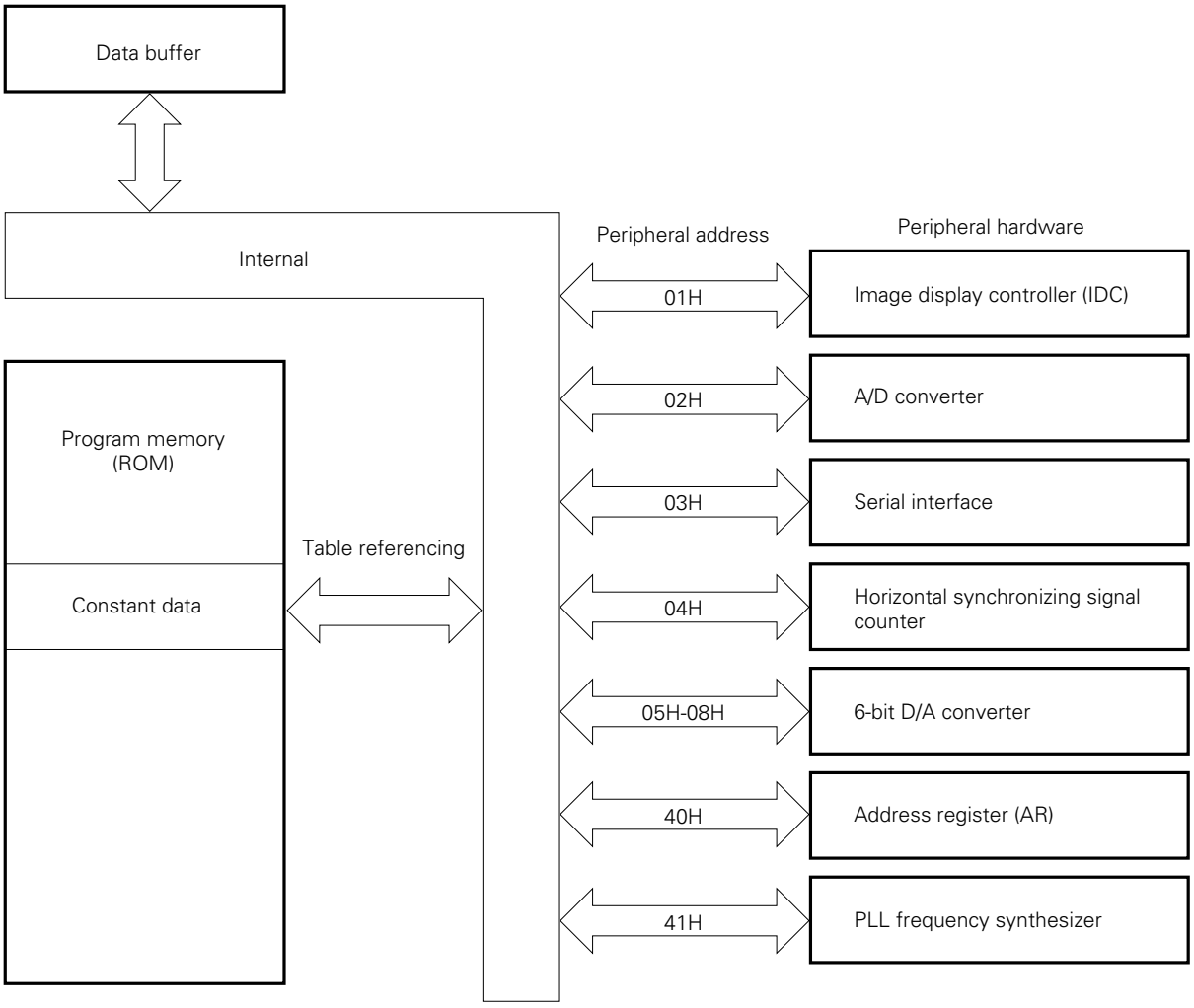
The data buffer provides the following two functions:

(1) Read constant data in program memory (to reference tables)
(2) Transfer data to and from peripheral hardware

Fig. 10-3 shows the relationship between the data buffer, peripheral hardware, and memory.

Table referencing is described in **Section 10.3**, and the peripheral hardware is described in **Sections 10.4** to **10.6**.

Fig. 10-3 Relationship Between Data Buffer, Peripheral Hardware, and Memory



10.3 DATA BUFFER AND TABLE REFERENCING

10.3.1 Table Referencing

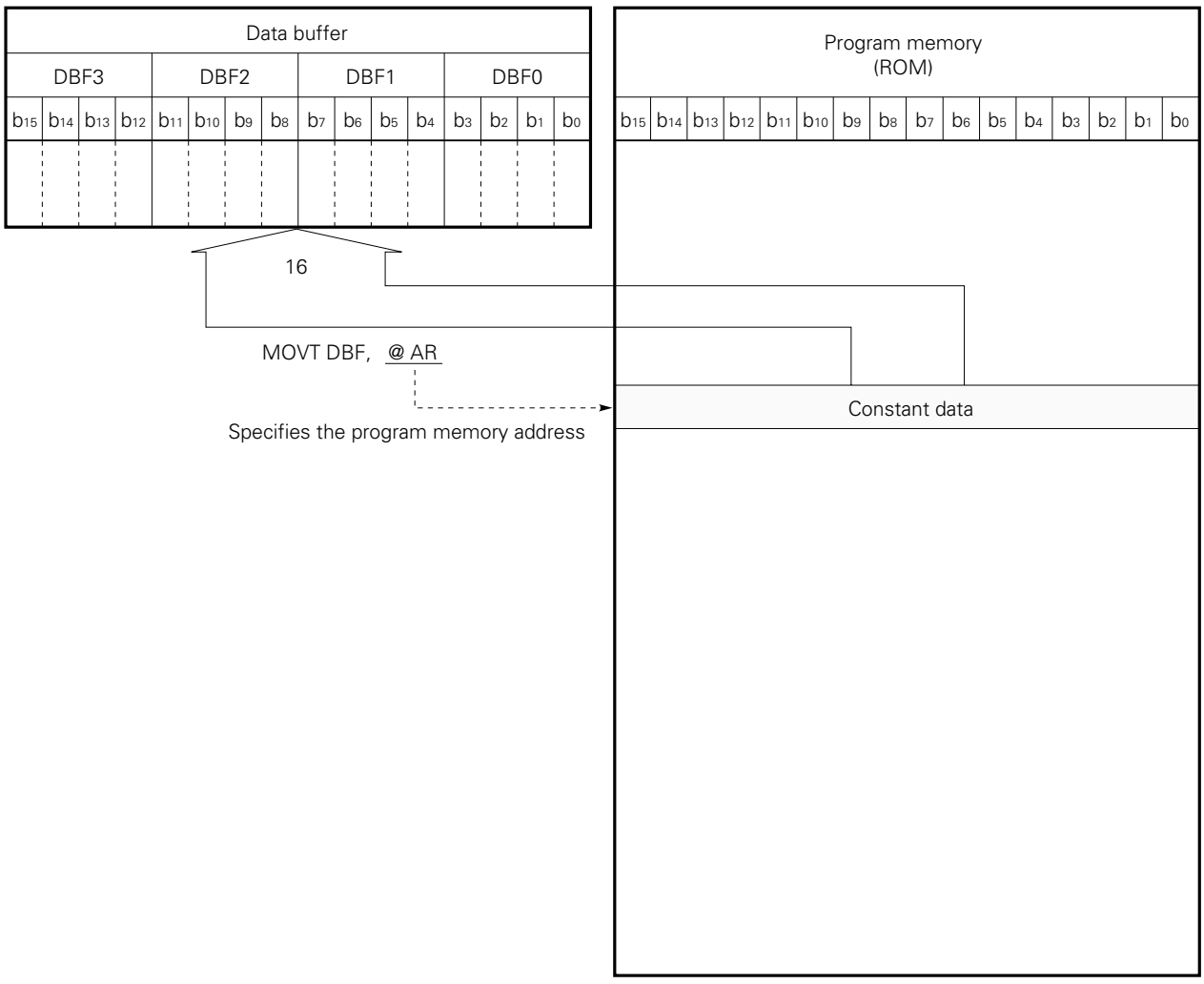
Tables are referenced by reading the constant data from program memory into the data buffer. This is done using the MOVT DBF, @AR instruction.

Therefore, if display data or other constant data is written to program memory in advance and a table reference instruction is executed, writing of a complex data conversion program is unnecessary.

The MOVT instruction is described below.

A example program is given in **Section 10.3.2**.

MOVT DBF, @AR ; Reads the contents of the program memory addressed by the address register into the data buffer as shown below.



When a table reference instruction is executed, the stack is used one level.

Because the address register (AR) has only eight valid bits, program memory available for table reference is limited to 256 steps from address 0000H to address 00FFH.

See also **Chapter 4** and **Section 8.1**.

10.3.2 Example Table Referencing Program

This section shows an example table referencing program.

Example

```
        P0A     MEM   0.70H       ;
        P0B     MEM   0.71H       ;
        P0C     MEM   0.72H       ;
        ORG     0000H
START :
        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    P0ABIO3, P0ABIO2, P0ABIO1, P0ABIO0
        SET4    P0BBIO3, P0BBIO2, P0BBIO1, P0BBIO0
        MOV     RPL,  #1110B      ; Sets general-purpose register to row address 7H of BANK0 .
        MOV     AR1,  #(.DL.DATA SHR 4 AND 0FH)
        MOV     AR0,  #(.DL.DATA SHR 0 AND 0FH)
                                  ; Sets address register to 0001H.
LOOP :
; ①
        MOVT    DBF,  @AR         ; Transfers the contents of the ROM specified by AR to data
                                  ; buffer.
; ②
        LD      P0A,  DBF2        ; Transfers the contents of data buffer to Port0A (70H),
        LD      P0B,  DBF1        ; Port0B(71H), and Port0C (72H) port data registers.
        LD      P0C,  DBF0
        ADD     AR0,  #1          ; Increments the contents of data register by one.
        ADDC    AR1,  #0
        SKNE    AR0,  #0CH        ; Writes 0 in AR0 when the value of AR0 reaches 0CH.
        MOV     AR0,  #0          ;
        BR      LOOP
```

This program sequentially reads the constant data stored at program memory addresses 0001H to 000CH into the data buffer (①) and outputs the data to Port0A, Port0B, and Port0C (②).

The constant data is left-shifted one bit. As a result, a high-level data is sequentially output to the Port0A, Port0B, and Port0C pins.

10.4 DATA BUFFER AND PERIPHERAL HARDWARE

10.4.1 How to Control Peripheral Hardware

The following peripheral hardware units transfer data via the data buffer:

- Image display controller
- A/D converter
- Serial interface
- Horizontal synchronizing signal counter
- 6-bit D/A converter
- Address register
- PLL frequency synthesizer

The peripheral hardware is controlled by setting the data in the peripheral hardware via the data buffer or reading its data.

Each peripheral hardware unit is provided with a data transfer register called a peripheral register. An address, called a peripheral address, is allocated to each peripheral hardware unit. Data transfer between the data buffer and peripheral hardware can be performed by executing a GET or PUT instruction (dedicated to the peripheral register) for the peripheral register.

The GET and PUT instructions are described below. The peripheral hardware and data buffer functions are listed in Table 10-1.

GET DBF, p; Reads the data of the peripheral register at address p into the data buffer.

PUT p, DBF; Writes the data of the data buffer to the peripheral register at address p.

There are three types of peripheral registers: Write/read (PUT/GET), write only (PUT), and read only (GET).

Device operation when a GET or PUT instruction is executed for a write only (PUT only) or read only (GET only) peripheral register is described below.

- When a read (GET) instruction is executed for a write only (PUT only) peripheral register, an undefined value is returned.
- When a write (PUT) instruction is executed for a read only (GET only), it has no effect.

Be careful when using a 17K series assembler and emulator.

For details, see **Section 10.6**.

Table 10-1 Peripheral Hardware and Data Buffer Functions

Peripheral hardware	Data buffer and data transfer peripheral register				Function		
	Name	Symbol	Peripheral address	PUT instruction/ GET instruction	Data buffer I/O bits	Valid bits	Explanation
Image display controller	IDC start position setting register	IDCORG	01H	PUT/GET	8	7	Sets the image display controller display start position.
A/D converter	A/D converter V_{REF} data register	ADCR	02H	PUT/GET	8	4	Sets the AD converter comparison voltage V_{REF}. $V_{REF} = \frac{x - 0.5}{16} \times V_{DD}$ (V) $1 \leq x \leq 15$
Serial interface	Presettable shift register	SIO0SFR	03H	PUT/GET	8	8	Sets the serial out data and reads the serial in data.
Horizontal synchronizing signal counter	HSYNC counter data register	HSC	04H	GET	8	6	Reads the value of the horizontal synchronizing signal counter.
6-bit D/A converter (PWM output) PWM_0 pin	PWM data register 0	PWMR0	05H	PUT/GET	8	7	Sets the D/A converter output signal duty. Duty $D = \frac{x + 0.75}{64}$ (%) $0 \leq x \leq 63$ Frequency f = 15.625 kHz
6-bit D/A converter (PWM output) PWM_1 pin	PWM data register 1	PWMR1	06H				
6-bit D/A converter (PWM output) PWM_2 pin	PWM data register 2	PWMR2	07H				
6-bit D/A converter (PWM output) PWM_3 pin	PWM data register 3	PWMR2	08H				
Address register	Address register	AR	40H	PUT/GET	16	16	Reads of writes data from or to the address register.
PLL frequency synthesizer	PLL data register	PLLR	41H	PUT/GET	16	16	Sets the PLL frequency synthesizer frequency division ratio.

10.4.2 Precautions When Transferring Data With Peripheral Registers

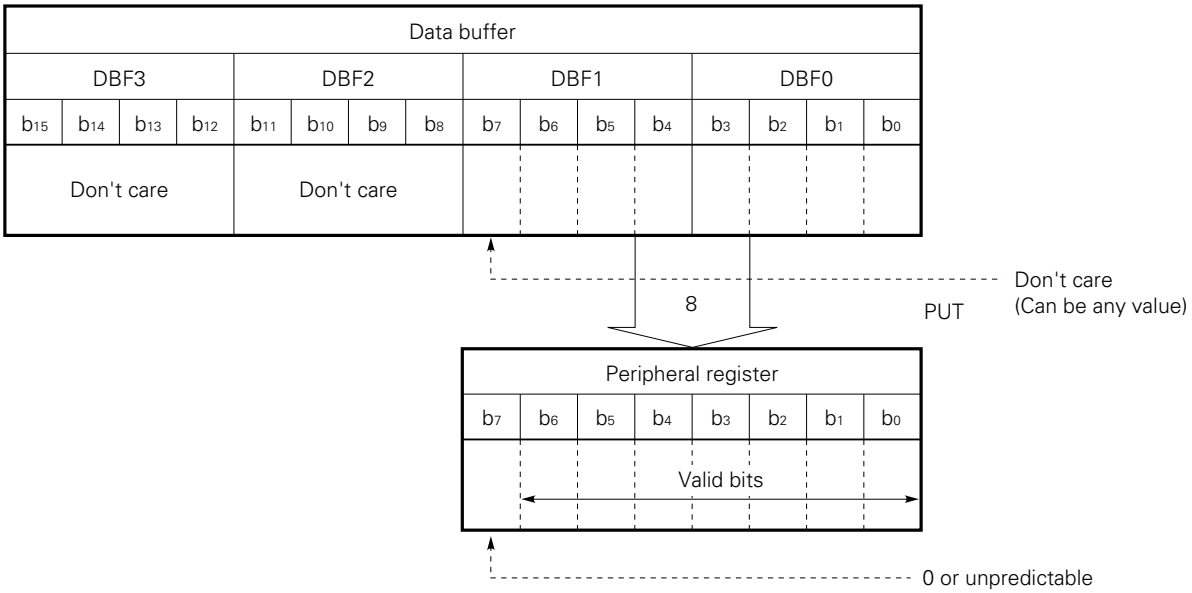
Data is transferred between the data buffer and peripheral registers in 8-bit or 16-bit units.

A PUT or GET instruction is executed for one instruction cycle (2 μs) even if the data is 16 bits long.

When 8-bit data transfer is performed but the peripheral register execution data is seven bits, for example, long one extra bit is added.

At data write, the status of this extra data is "Don't care" as shown in Example 1. At data read, the status of this extra data is "Unpredictable" as shown in Example 2.

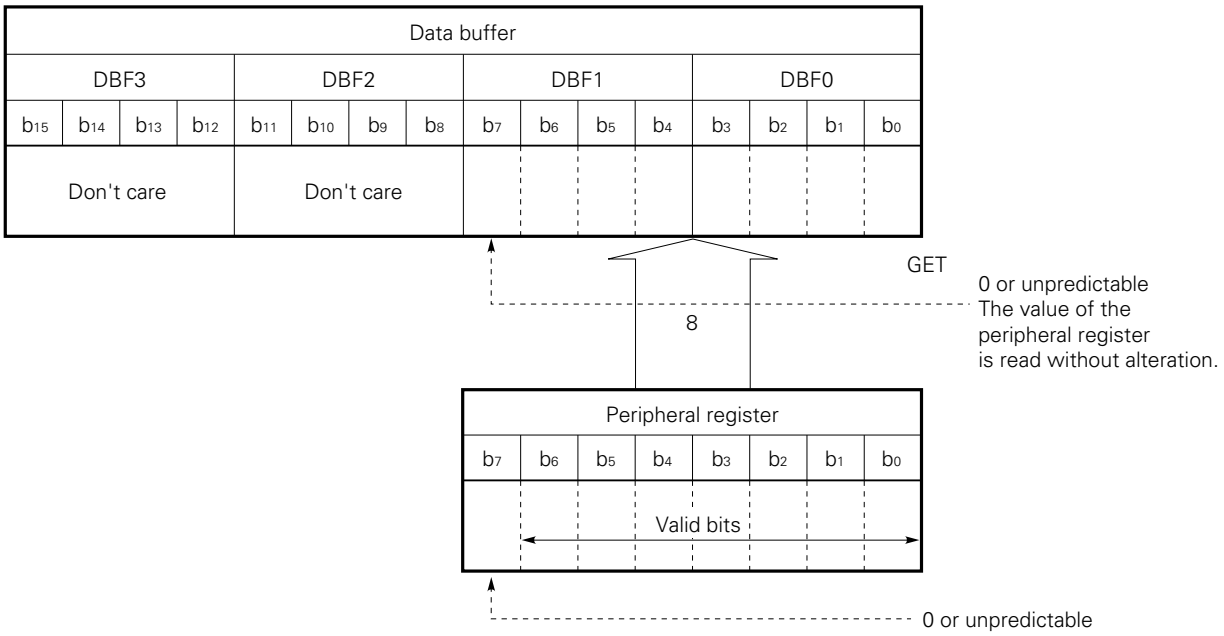
Example 1. PUT instruction (When the valid peripheral register bits are seven bits from bit_0 to bit_6.)



When 8-bit data is written to a peripheral register, the status of the eight high-order bits of the data buffer (contents of DBF3 and DBF2) is "Don't care".

Of the 8-bit data in the data buffer, the status of each bit that does not correspond to a valid bit in the peripheral register is "Don't care".

Example 2. GET instruction



When the 8-bit data of a peripheral register is read, the value of the eight high-order bits (DBF3 and DBF2) of the data register does not change.

Of the 8-bit data of the data register, each bit that is not a valid peripheral register bit becomes 0 or unpredictable. Whether the bit becomes 0 or unpredictable is decided in advance for each peripheral register.

10.4.3 State at Peripheral Register Reset

The valid bits of each peripheral register are reset as follows:

Reset	Valid bit state
Power-on	Unpredictable
Clock-stop	Previous state held
CE	Previous state held

10.5 Data Buffer and Peripheral Registers

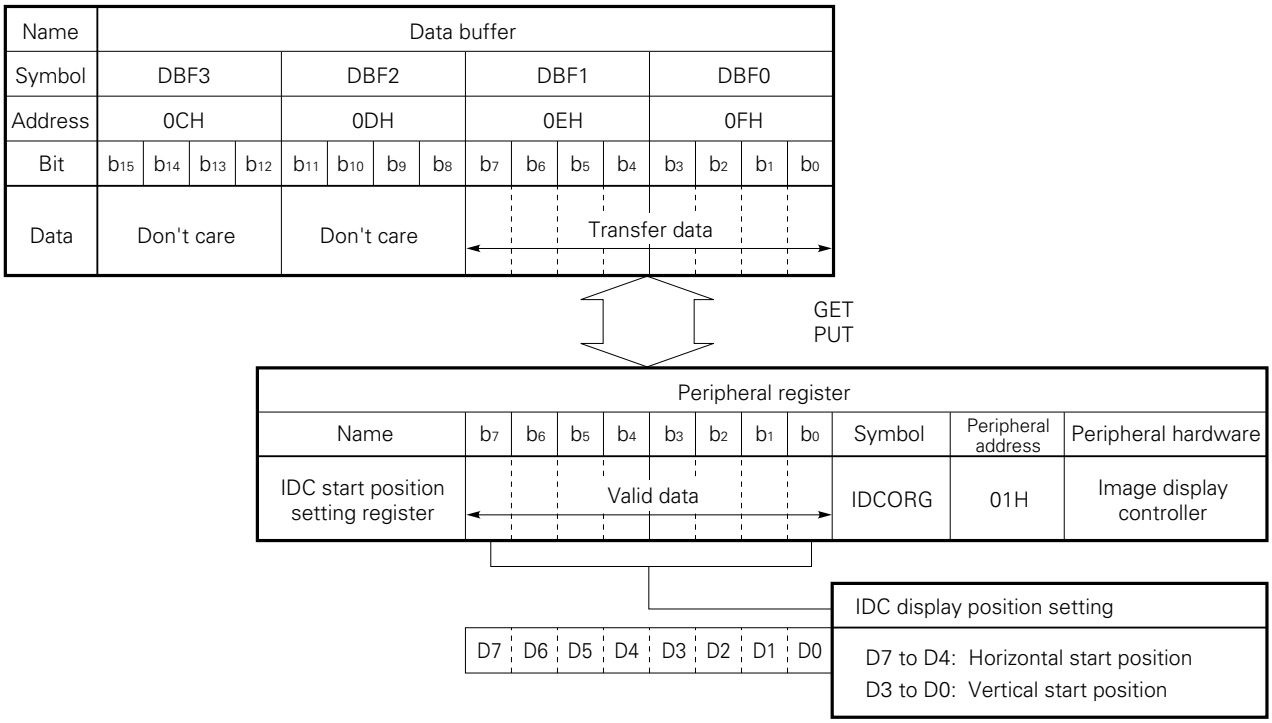
Sections 10.5.1 to **10.5.7** describe the data buffer and the peripheral registers.

10.5.1 IDC Start Position Setting Register

Fig. 10-4 shows the functions of the IDC start position setting register.

The IDC start position setting register sets the IDC display start position.

Fig. 10-4 IDC Start Position Register Functions



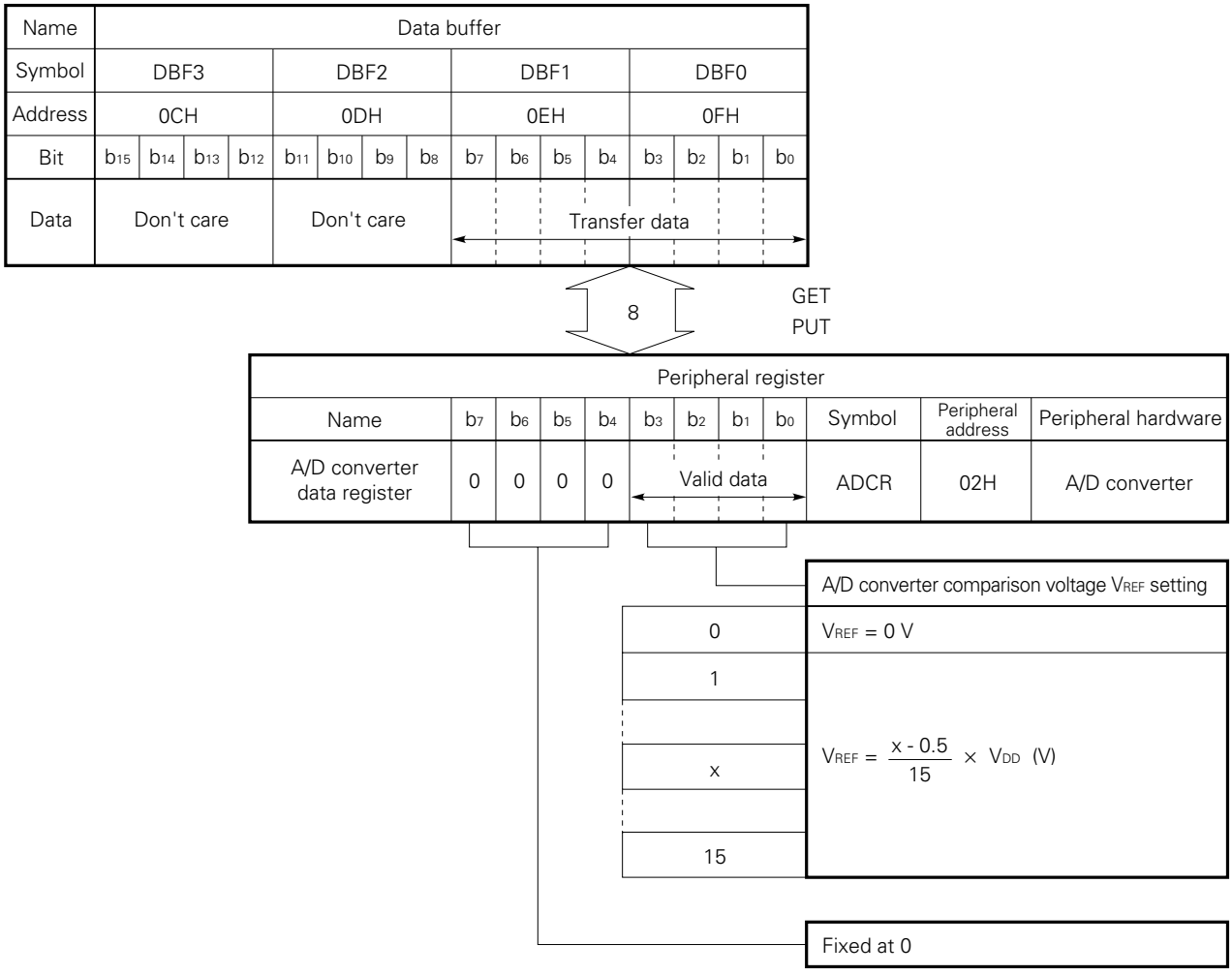
10.5.2 A/D Converter Data Register

Fig. 10-5 shows the functions of the A/D converter data register.

The A/D converter data register sets the A/D converter comparison voltage.

Because the A/D converter is a 4-bit converter, the four low-order bits of the A/D converter data register are valid.

Fig. 10-5 A/D Converter Data Register Functions

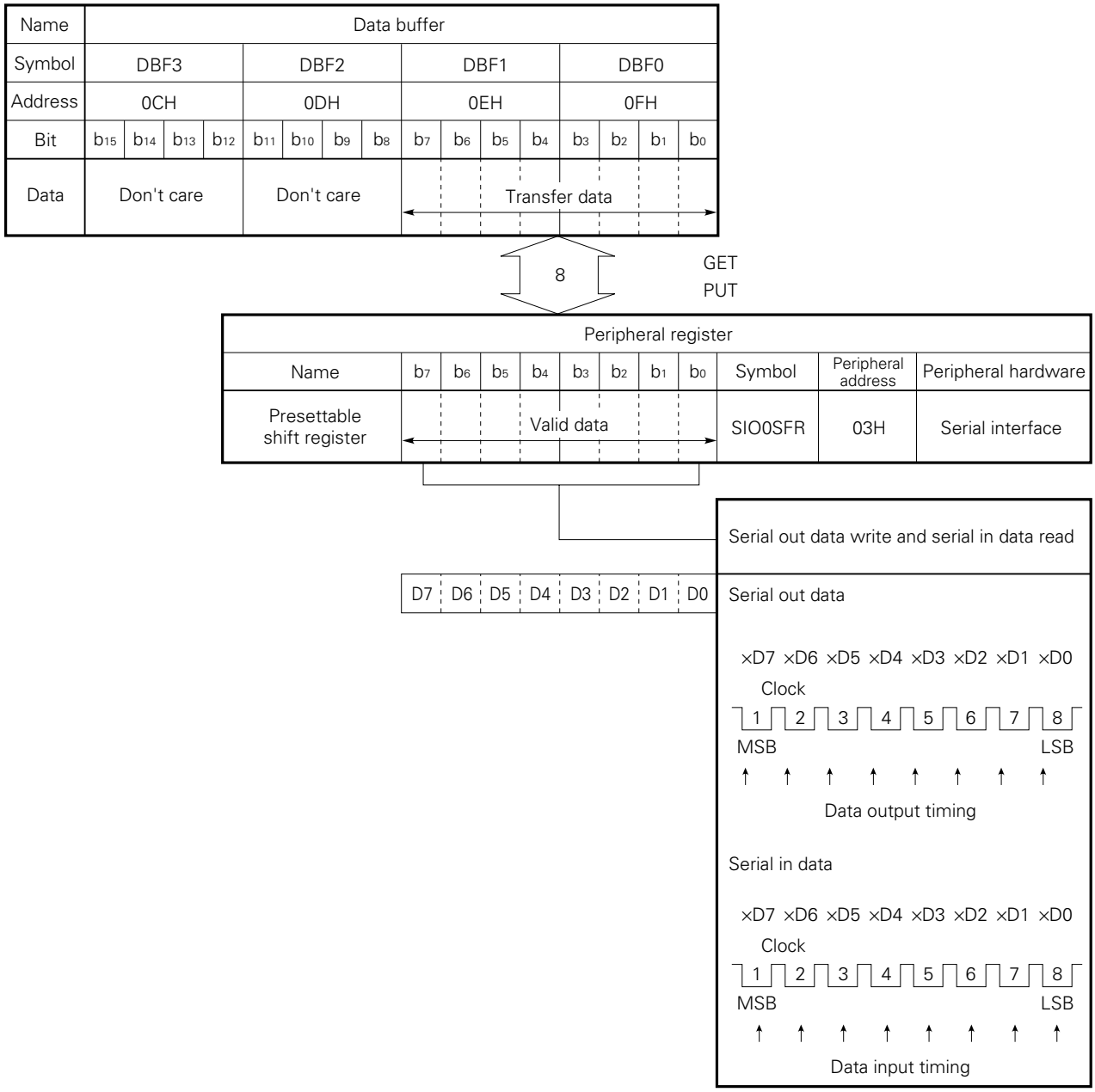


10.5.3 Presettable Shift Register

Fig. 10.6 shows the functions of the presettable shift register.

The presettable shift register writes the serial interface serial out data and reads the serial interface serial in data.

Fig. 10-6 Relationship between Presettable Shift Register and Data Buffer



Serial interface serial data is output while being shifted sequentially the data from the MSB (bit b_7) of the presettable shift register.

During serial data input, data is shifted sequentially from the LSB (bit b_0).

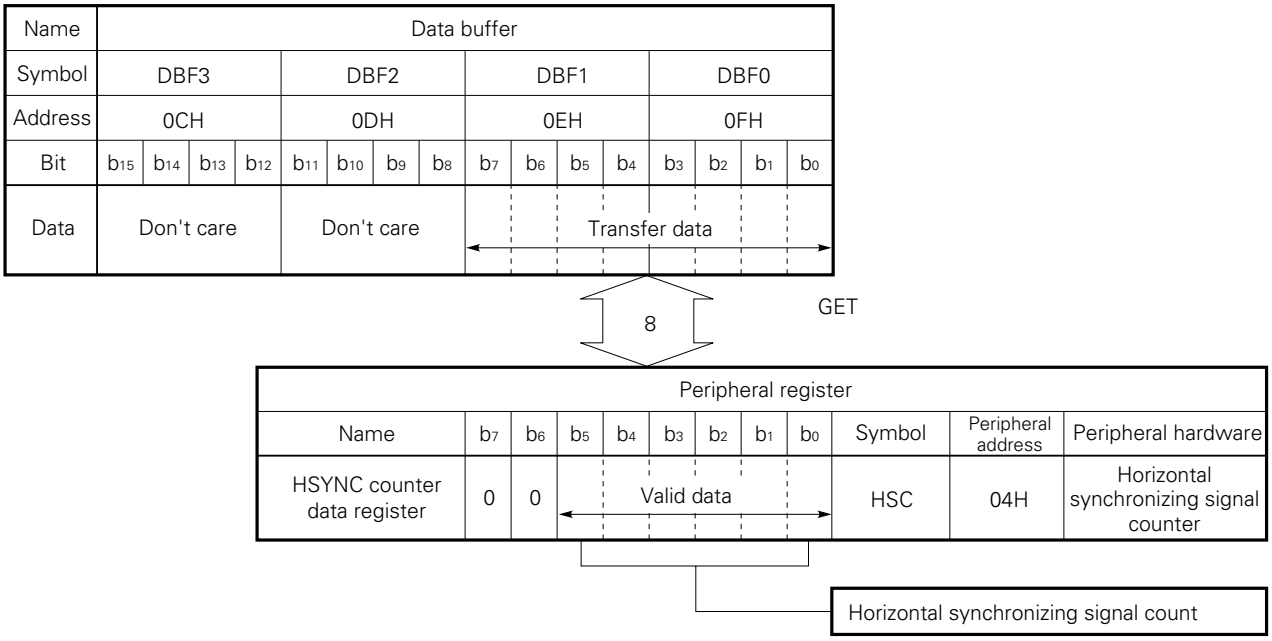
10.5.4 HSYNC Counter Data Register

Fig. 10.7 shows how the HSYNC counter data register functions .

The HSYNC counter data register reads the horizontal synchronizing signal count.

When the HSYNC counter data register reaches 3FH, it returns to 00H at the next input.

Fig. 10-7 HSYNC Data Register Functions



10.5.5 PWM Data Register

Fig. 10-8 shows how the PWM data register functions.

The PWM data register sets the duty cycle of the 6-bit D/A converter (PWM output) output.

The 6-bit D/A converter has four channels (pins PWM_3, PWM_2, PWM_1, and PWM_0). Because the duty cycle can be set independently for each channel, four independent PWM duty cycle registers are also provided.

Fig. 10-8 PWM Data Register Functions

Name	Data buffer															
Symbol	DBF3				DBF2				DBF1				DBF0			
Address	0CH				0DH				0EH				0FH			
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	Don't care				Don't care				Transfer data							

8

GET
PUT

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	0	Valid data							PWMR0	05H	PWM_0 pin
PWM1 data register	0								PWMR1	06H	PWM_1 pin
PWM2 data register	0								PWMR2	07H	PWM_2 pin
PWM3 data register	0								PWMR3	08H	PWM_3 pin

Set the PWM output duty of each pin.

0

x

63

$$\text{Duty } D = \frac{x + 0.75}{64} \ (\%)$$

Frequency f = 15.625 kHz

(f: PWM output repetition frequency)

0	Use PWM pins as D/A converter.
0	Use PWM as 1-bit output pin. Output contents of b_5.

10.5.6 Address Registers

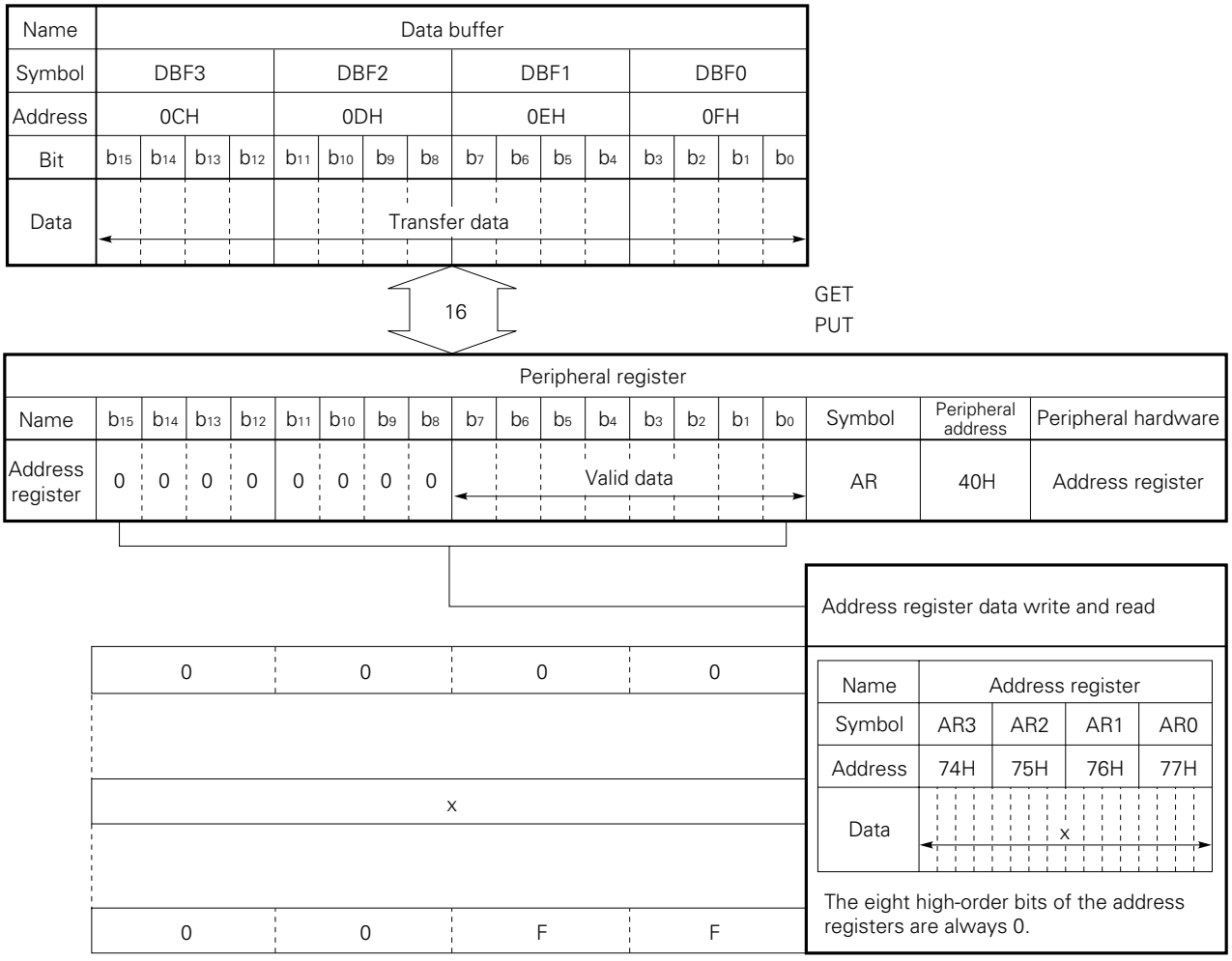
The address registers are mapped to addresses 74H to 77H in the system register (at data memory addresses 74H to 7FH). They are used for program memory address operations. See **Chapter 8**.

The address registers can be used to manipulate data directly with data memory operation instructions. They can also be used to transfer data via the data buffer as part of the peripheral hardware.

In other words, data can be read and written via the data buffer with PUT and GET instructions, as well as data memory operation instructions.

Fig. 10-9 shows the relationship between the address registers and the data buffer.

Fig. 10-9 Relationship Between Address Registers and Data Buffer

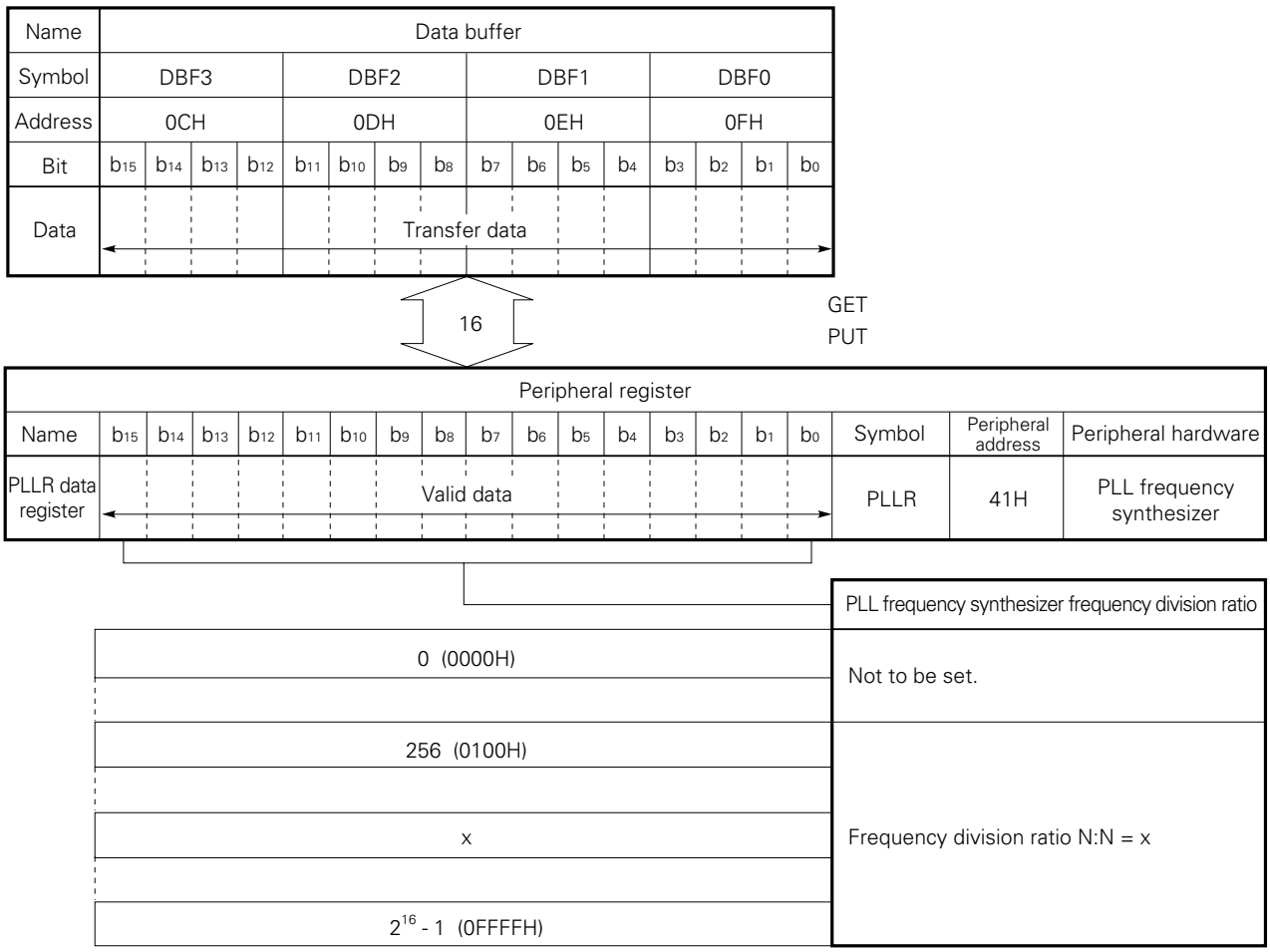


10.5.7 PLL Data Register

Fig. 10-10 shows how the PLL data register functions.

The PLL data register sets the frequency division ratio of the PLL frequency synthesizer. For the pulse swallow method, all 16 bits are valid, the 12 high-order bits are set in the program counter, and the remaining four low-order bits are set in the swallow counter.

Fig. 10-10 PLL Data Register



10.6 PRECAUTIONS WHEN USING DATA BUFFERS

10.6.1 Write Only, Read Only, and Unused Address Data Buffer Precautions

When the 17K series assembler and emulator are used for data transfer with peripheral hardware via the data buffer, note the following regarding unused peripheral addresses and write only (PUT only) and read only (GET only) peripheral registers.

(1) Device operation

Reading from a write only peripheral register returns an unpredictable value.
Writing to a read only register does not change its contents.
Reading from an unused address returns an unpredictable value. Writing to an unused address does not change its contents.

(2) When using an assembler

An instruction that reads from a write only register generates an error.
An instruction that writes to a read only register generates an error.
An instruction that reads from or writes to an unused address generates an error.

(3) When using an emulator (used to execute instructions by batch processing, etc.)

Reading from a write only register returns an unpredictable value and does not generate an error.
Writing to a read only register does not change its contents and does not generate an error.
Reading from an unused address returns an unpredictable value. Writing to an unused address does not change its contents and does not generate an error.

10.6.2 Peripheral Register Addresses and Reserved Words

When a 17K series assembler is used, no error is generated when peripheral address "p" is specified directly (with a numerical value) in PUT p, DBF or GET DBF, p as shown in Example 1.

However, to reduce program bugs, this method should be avoided.

Therefore, the peripheral addresses should be symbolically defined with symbol definition instructions (an assembler pseudo instructions), as shown in Example 2.

To simplify symbol definition, peripheral addresses are predefined in the assembler as reserved words.

Therefore, if reserved words are used, a program can be written without performing symbol definition, as shown in Example 3.

The reserved words of peripheral registers are shown in the Symbol field in Table 10-1 and the Symbol field in Figs. 10-4 to 10-10.

Example 1.

```
        PUT     02H,    DBF     ; The assembler does not generate an error if peripheral
        GET     DBF,    03H     ; addresses are directly specified by 02H and 03H.  How
                                ; ever, to reduce program bugs, this method should be
                                ; avoided.
```

2.

```
SIO0DATA    DAT     03H     ; Assigns SIO0DATA to 03H using a symbol definition
        PUT     SIO0DATA,   DBF     ; instruction.
```

3.

```
        PUT     SIO0SFR             ; If reserved word SIO0SFR is used, symbol definition is
                                    ; unnecessary.
```

11. INTERRUPT

An interrupt temporarily stops the program being executed in response to a request from the peripheral hardware (INT_{NC} pin, timer, $\overline{V_{SYNC}}$ pin or serial interface). The interrupt then branches the program flow to a predetermined address (vector address).

11.1 INTERRUPT BLOCK CONFIGURATION

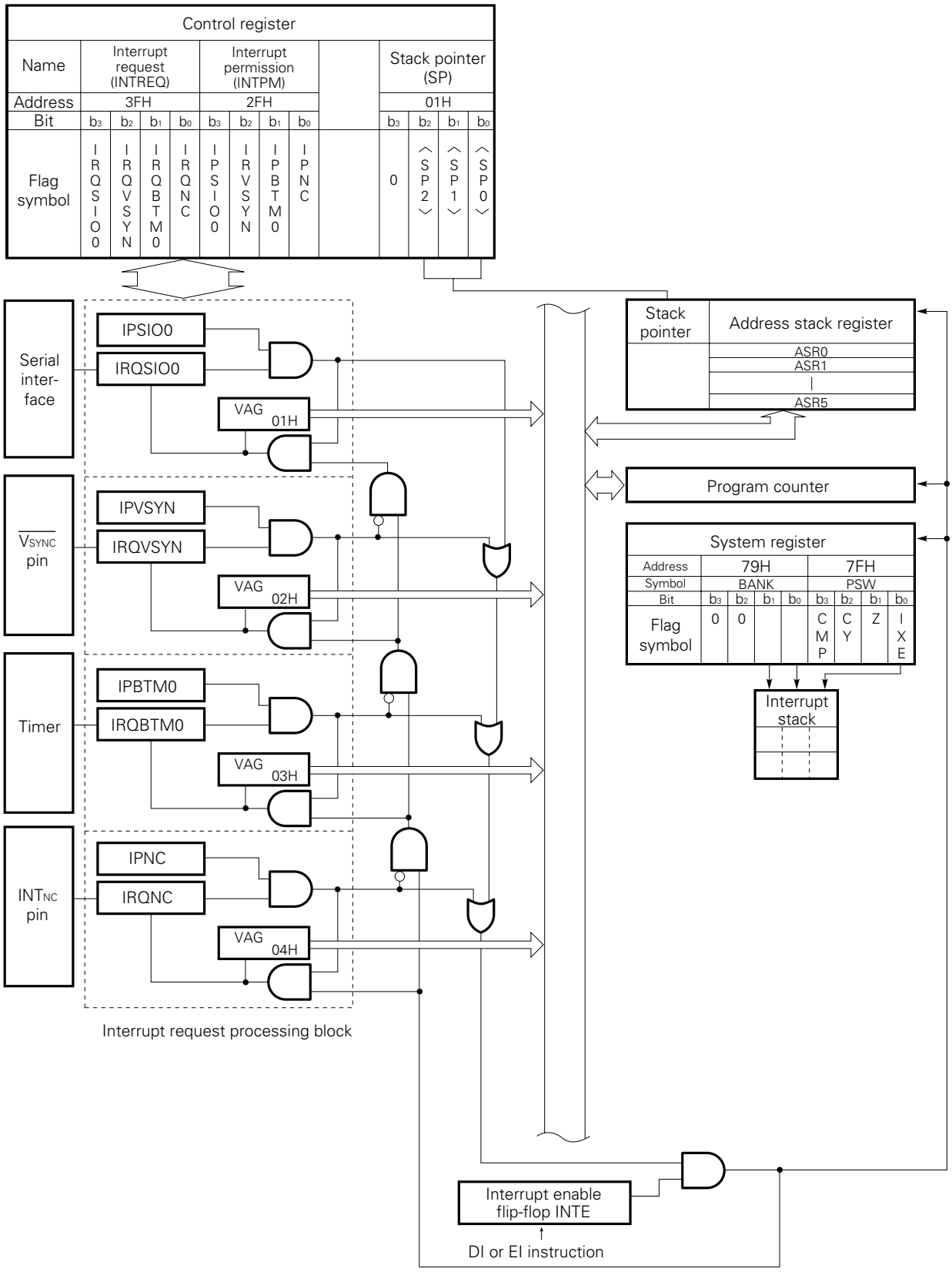
Fig. 11-1 shows the interrupt block configuration.

The interrupt block consists of the interrupt request control blocks, interrupt enable flip-flop (INTE), stack pointer, address stack register, program counter, and interrupt stack. The interrupt request control blocks control interrupt requests from the INT_{NC} pin, timer, $\overline{V_{SYNC}}$ pin, and serial interface. The interrupt enable flip-flop (INTE) sets all interrupt permissions. The stack pointer, address stack register, program counter, and interrupt stack are controlled when an interrupt is accepted.

The interrupt request processing block in the peripheral hardware consists of the IRQ×××flip-flop, IP××× flip-flop, and vector address generator (VAG). The IRQ××× flip-flop detects an interrupt request, the IP××× flip-flop sets an interrupt permission, and the vector address generator (VAG) specifies a vector address at interrupt acceptance.

The IRQ××× flip-flop and IP××× flip-flop correspond to the interrupt request flag and interrupt permission flag, respectively, in the control register in one-to-one ratio.

Fig. 11-1 Interrupt Block Configuration



11.2 INTERRUPT FUNCTION

The following peripheral hardware can use the interrupt function: the INTNC pin, timer, $\overline{\text{VSYNC}}$ pin, and serial interface.

If the peripheral hardware satisfies the specified condition (e.g., a falling edge is input to the INTNC pin), the interrupt function temporarily stops the program being executed and starts the exclusive processing program.

The interrupt signal sent from the peripheral hardware at this time is called an interrupt request. Outputting an interrupt signal can be expressed as "issuing an interrupt signal". The exclusive processing program for interrupts is called the interrupt processing routine.

When an interrupt is accepted, processing is branched to the program memory address (vector address) specified for each interrupt source. Each interrupt processing routine can be started from this vector address.

Processing for the interrupt function can be divided into the processing done before interrupt acceptance and the processing done after interrupt acceptance. First, the interrupt function operates until the interrupt request from the peripheral hardware is accepted. Then, after the interrupt is accepted, the interrupt function branches processing to the vector address and returns control to the program that was interrupted.

Sections 11.2.1 to **11.2.8** describe the functions of the blocks shown in Fig. 11-1.

11.2.1 Peripheral Hardware

There are four peripheral hardware interrupt functions: the INTNC pin, timer, $\overline{\text{VSYNC}}$ pin, and serial interface.

Interrupt request issuance conditions can be set for each type of peripheral hardware.

For example, the request issuance timing for the INTNC pin (rising or falling edge of the signal applied to the INTNC pin) can be selected.

See **Sections 11.3** to **11.7** for details of interrupt request issuance conditions for the peripheral hardware.

11.2.2 Interrupt Request Processing Block

An interrupt request processing block is provided for each type of peripheral hardware. This block controls interrupt request permits an interrupt, and generates the vector address at interrupt acceptance.

Sections 11.2.3 to **11.2.8** describe the flags of the interrupt request processing block.

11.2.3 Interrupt Request Flags (IRQ×××)

The interrupt request flags are set to 1 when an interrupt request is issued from the peripheral hardware. These flags are reset to 0 when the interrupt request is accepted.

Because the interrupt request flags correspond one-to-one to the flags in the interrupt request register, they can be read and written via the window register.

Writing a 1 via the window register has the same effect as issuing an interrupt request.

Once these flags are set, they are not reset until the corresponding interrupts are accepted or a 0 is written via the window register.

Even if two or more interrupt requests are issued together, the interrupt request flags corresponding to the unaccepted interrupts are not reset.

These flags are reset to 0 at power-on reset, clock stop, or CE reset.

11.2.4 Interrupt Permission Flags (IP×××)

The interrupt permission flags set interrupt permissions for various types of peripheral hardware.

If these flags are set to 1 and the corresponding interrupt request flags are also set, the corresponding interrupt requests are output.

Because these flags correspond one-to-one to the flags in the interrupt permission register of the control register, they are read and written via the window register.

These flags are reset to 0 at power-on reset, clock stop, or CE reset.

11.2.5 Vector Address Generator (VAG)

When interrupts from various types of peripheral hardware are accepted, the vector address generator generates the branch address (vector address) of the program memory for the source of the accepted interrupt.

Table 11-1 lists the vector addresses corresponding to the interrupt sources.

Table 11-1 Interrupt Vector Addresses

Interrupt source	Vector address
INT_{NC} pin	04H
Timer	03H
$\overline{V_{SYNC}}$ pin	02H
Serial interface	01H

11.2.6 Interrupt Enable Flip-Flop (INTE)

The interrupt enable flip-flop sets the interrupt permissions of all four types of interrupts.

If each interrupt request processing block outputs a 1 while this flip-flop is set to 1, a 1 is output from this flip-flop and an interrupt is accepted.

Even if a 1 is output from each interrupt request processing block while this flip-flop is reset to 0, an interrupt is not accepted.

To set or reset this flip-flop, use exclusive instructions EI (set) and DI (reset).

If the EI instruction is executed, this flip-flop is set when the instruction executed after the EI instruction is completed. If the DI instruction is executed, the flip-flop is reset during the DI instruction execution cycle.

If an interrupt is accepted while the interrupt enable flip-flop is set (EI state), this flip-flop is reset (DI state).

Even if the DI instruction is executed in the DI state or the EI instruction is executed in the EI state, the instruction is invalid.

This flag is reset (DI state) at power-on reset, clock stop, or CE reset.

11.2.7 Stack Pointer, Address Stack Register, and Program Counter

The return address of control returned from the interrupt processing routine is saved in the address stack register.

The stack pointer specifies one of six address stack registers (ASR0 to ASR5) to be used.

In other words, when an interrupt is accepted, the stack pointer value is reduced by 1, and the program counter value is saved in the address stack register indicated by the stack pointer. Next, if exclusive return instruction RETI is executed after the interrupt processing routine, the contents of the address stack register indicated by the stack pointer are returned to the program counter. At this time, the stack pointer value is increased by 1.

See also **Chapter 4**.

11.2.8 Interrupt Stack

The interrupt stack saves the contents of the bank register and index enable flag in the system register when an interrupt is accepted.

When the interrupt is accepted and the contents of the bank register and index enable flag are saved, the bank register and index enable flag in the system register are reset to 0.

The interrupt stack can save the contents of the bank registers and index enable flags of up to two levels. Therefore, the interrupt stack can issue multiple interrupts of up to two levels; for example, an interrupt can be accepted during execution of a routine that is processing another interrupt.

The contents of the interrupt stack are restored in the bank register and index enable flag in the system register by executing the RETI instruction. The RETI instruction is exclusively used to return control from the interrupt processing routine.

See also **Chapter 4**.

11.3 INTERRUPT ACCEPTANCE

11.3.1 Interrupt Acceptance and Priority

An interrupt is accepted as follows:

(1) When the interrupt conditions are satisfied (e.g., a rising edge is input to the INTNC pin), each type of peripheral hardware outputs the interrupt request signal to the interrupt request blocks.

(2) When an interrupt request block accepts an interrupt request signal from the peripheral hardware, it sets the corresponding IRQxxx flag to 1 (e.g., sets IRQNC for the INTNC pin).

(3) If an interrupt permission flag corresponding to an IRQxxx (e.g., IPNC flag for the IRQNC flag) is set 1 when each interrupt request flag is set, each interrupt request block outputs a 1.

(4) A signal output from each interrupt request block is input to the interrupt enable flip-flop via an OR circuit. This interrupt enable flip-flop is set to 1 by the EI instruction and reset by the DI instruction.
If a 1 is output from each interrupt request block while the interrupt enable flip-flop is set, a 1 is output from the interrupt enable flip-flop and the interrupt is accepted.

When the interrupt is accepted, the signal from the interrupt enable flip-flop is input to the interrupt request block via an AND circuit as shown in Fig. 11-1.
The interrupt request flag is reset by the signal input to each interrupt request block, and the vector address for each interrupt is output.
If a 1 is output from the interrupt request block at this time, the interrupt acceptance signal is not transferred to the next level. If two or more interrupt requests are issued together, they are accepted in the following sequence:

(DMA) > INTNC pin > timer > $\overline{\text{VSYNC}}$ pin > serial interface

This sequence is called the hardware priority.

Fig. 11-2 shows the interrupt acceptance flowchart.

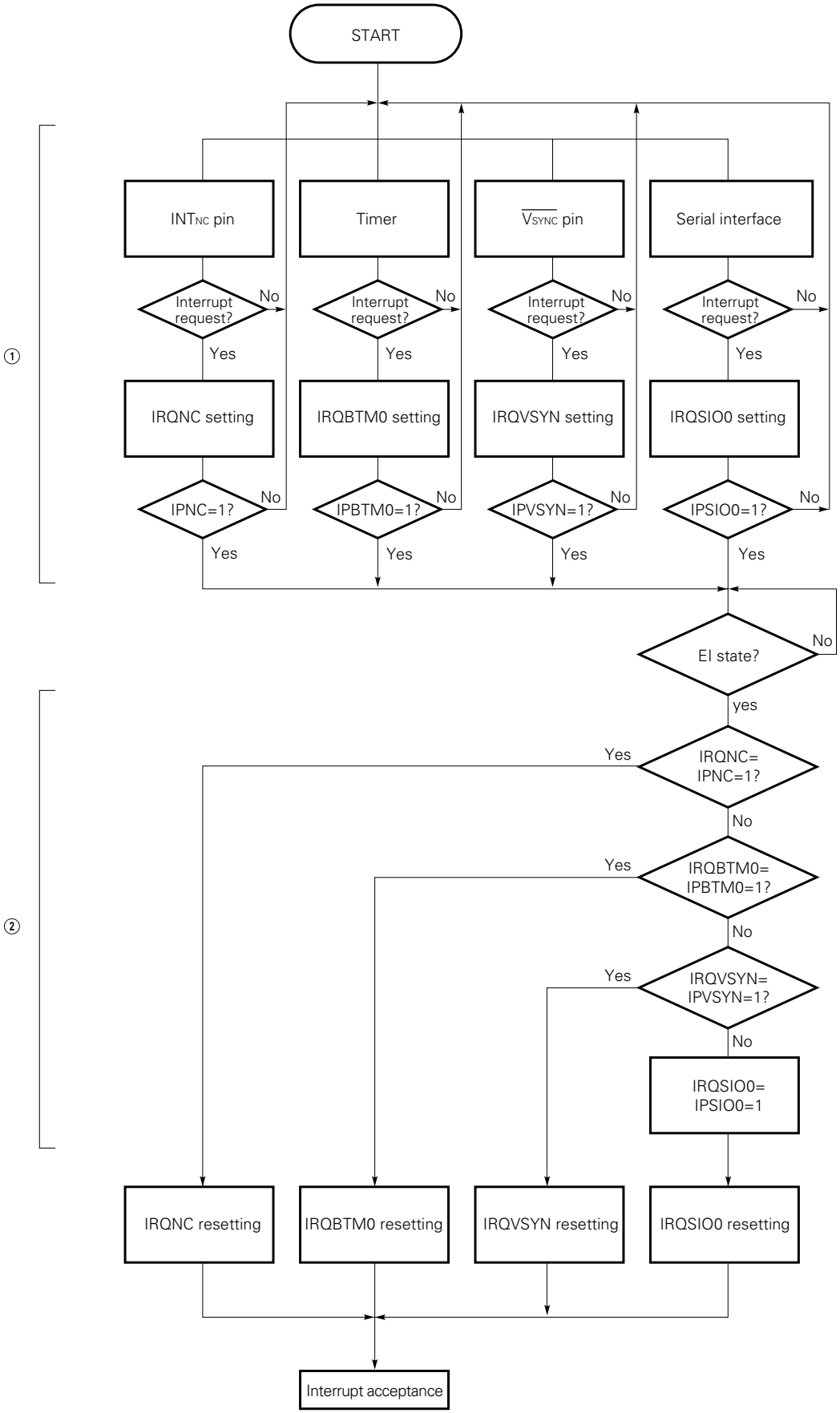
The processing in ① of Fig. 11-2 is always executed in parallel. If two or more interrupt requests are generated at the same time, the interrupt request flags are set at the same time.

On the other hand, the processing in ② is executed according to the priority given by the interrupt permission flags.

In other words, if an interrupt permission flag is not set, the interrupt from the interrupt source is not accepted. An interrupt with a high hardware priority can be inhibited by resetting the corresponding interrupt permission flag in the program.

This type of interrupt is called a maskable interrupt. For a maskable interrupt, an interrupt with a high hardware priority can be inhibited by the program; therefore, it is also called the software priority.

Fig. 11-2 Interrupt Acceptance Flowchart



11.3.2 Timing Chart at Interrupt Acceptance

Fig. 11-3 shows the timing chart at interrupt acceptance.

Fig. 11-3 (1) shows the timing chart of one interrupt.

The timing chart when an interrupt request flag is set to 1 is shown in (a) of (1). The timing chart when an interrupt permission flag is set to 1 is shown in (b) of (1).

In both cases, the interrupt is accepted when the interrupt request flag, interrupt enable flip-flop, and interrupt permission flag are all set.

If the flag or flip-flop that is set satisfies the skip conditions or the conditions for the first instruction cycle of the MOVT DBF or @AR instruction, the interrupt is accepted after execution of the skipped instruction (becomes NOP) or the second instruction cycle of the MOVT DBF or @AR instruction.

The interrupt enable flip-flop is set in the instruction cycle after the cycle in which the EI instruction is executed.

Fig. 11-3 (2) shows the timing chart when two or more interrupts are used.

If all interrupt permission flags are set when two or more interrupts are used, the interrupt with the highest hardware priority is accepted first. The program can be used to change the interrupt permission flags to change the hardware priority.

The interrupt cycle shown in Fig. 11-3 is a special cycle in which an interrupt request flag is reset, a vector address is specified, and the contents of the program counter are saved after an interrupt is accepted. The time required for an interrupt is equal to the time required for one instruction (2 μs, or 12 μs when the IDC is operating). See **Section 11.4** for details.

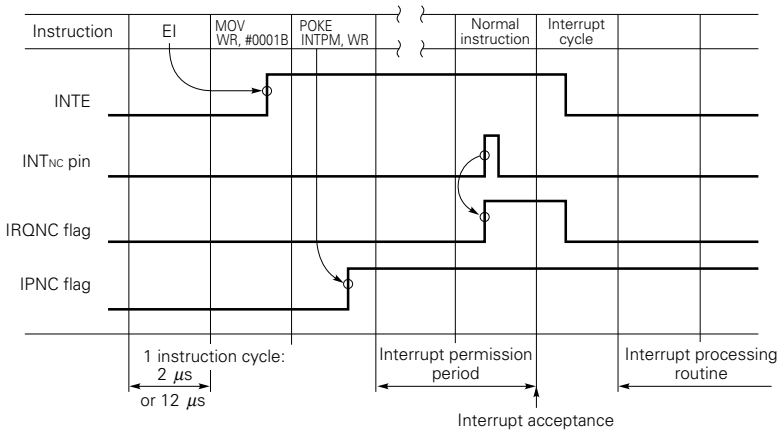
Because the interrupt request flag is set to 1 regardless of the EI instruction and interrupt permission flags, an interrupt request can be identified by detecting an interrupt request flag using the program.

Fig. 11-3 Interrupt Reception Timing Chart (1/2)

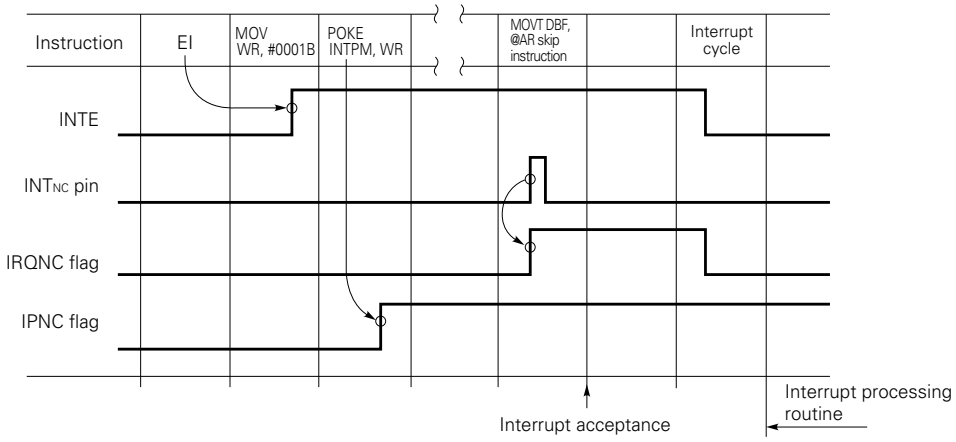
(1) When one interrupt (e.g., rising edge at the INTNC pin) is used

(a) When an interrupt mask time is not set by the interrupt permission flag

① When the MOVT instruction or a normal instruction that does not satisfy the skip conditions is executed at interrupt acceptance



② When the MOVT instruction or an instruction satisfying the skip conditions is executed at interrupt reception



(b) When an interrupt holding period is set by the interrupt permission flag

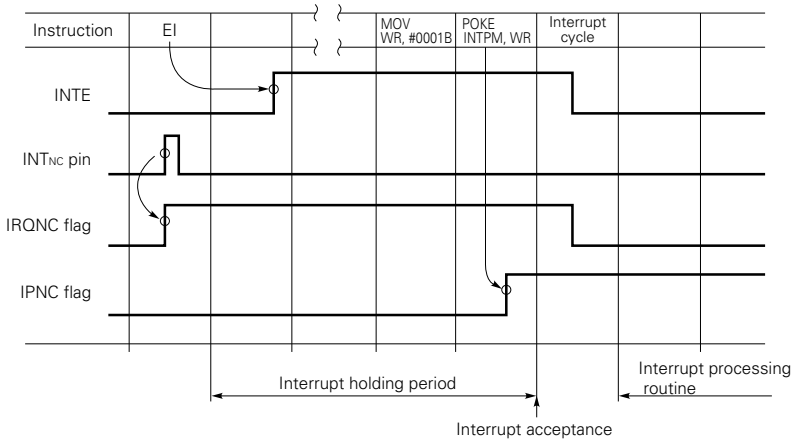
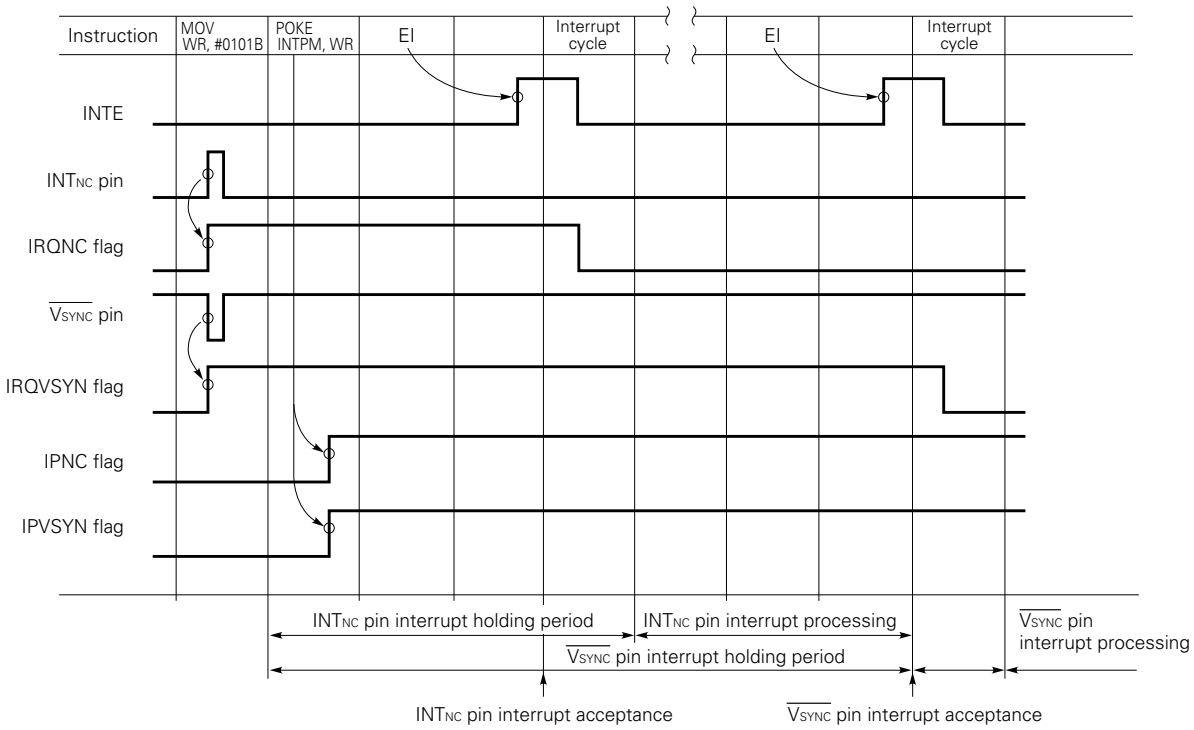


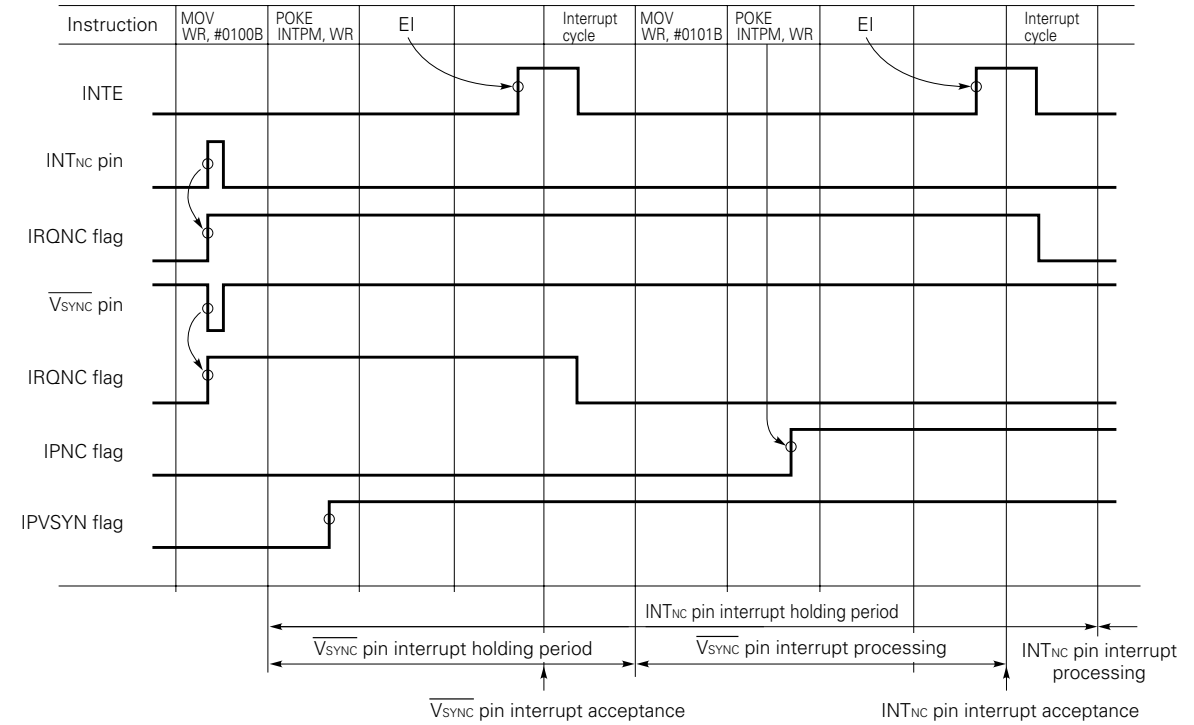
Fig. 11-3 Interrupt Acceptance Timing Chart

(2) When two or more interrupts (e.g., rising edge at the INTNC pin and falling edge at the $\overline{\text{VSYNC}}$ pin) are used

(a) Hardware priorities



(b) Software priorities



11.4 OPERATIONS AFTER INTERRUPT ACCEPTANCE

When an interrupt is accepted, the following processing sequence is executed:

(1) The interrupt enable flip-flop or interrupt request flag corresponding to the accepted interrupt is reset. In other words, a write protected state is set.

(2) The stack pointer value is decreased by 1.

(3) The contents of the program counter are saved in the address stack register indicated by the stack pointer. The contents of the program counter become the program memory address after the contents at interrupt acceptance. For a branch instruction, the contents become the branch destination address. For a subroutine call instruction, the contents become the called address. If a skip instruction satisfies the skip conditions, an interrupt is accepted after the next instruction is executed as the NOP instruction. Therefore, the contents of the program counter become the skipped address.

(4) The lower two bits of the bank register (BANK: address 79H) and the index enable flag (IXE: bit b_0 of address 7FH) are saved in the interrupt stack.

(5) The contents of the vector address generator corresponding to the accepted interrupt are transferred to the program counter. In other words, processing is branched to the interrupt processing routine.

The processing in (1) to (5) above is executed during one special instruction cycle (2 μs, or 12 μs when the IDC is operating) without normal instruction execution.

This instruction cycle is called the interrupt cycle. The processing from interrupt acceptance to branching to the corresponding vector address requires one instruction cycle.

11.5 RETURNING CONTROL FROM INTERRUPT PROCESSING ROUTINE

To return control from the interrupt processing routine to the processing executed at interrupt acceptance, use the exclusive RETI instruction. When the RETI instruction is executed, the following processing sequence is executed.

(1) The contents of the address stack register indicated by the stack pointer are restored in the program counter.

(2) The contents of the interrupt stack are restored in the lower two bits of the bank register or bit b_0 of the index enable flag.

(3) The stack pointer value is increased by 1.

The processing in (1) to (3) above is executed during one instruction cycle of the RETI instruction. The only difference between the RETI instruction and subroutine return instruction RET or RETSK is in the restoration of the contents of the bank register or index enable flag in (2) above.

11.6 INTERRUPT PROCESSING ROUTINE

An interrupt is accepted in a program area that permits interrupts regardless of the program being executed.

Therefore, to return control to the original program after interrupt processing, return the program to the state it is in when it is not processing an interrupt.

For example, if an arithmetic operation is performed during interrupt processing, the contents of the carry flag may differ from those before interrupt acceptance. This content change may cause a decision error in the program to which control has returned.

A system or control register that can at least operate within the interrupt processing routine should be saved or restored within the interrupt processing routine.

See **Section 11.9** for processing that permits an interrupt while another interrupt is being processed (multiple interrupts).

11.6.1 Save Processing

This section describes how to save the contents of registers using the interrupt routine as an example.

Only the contents of bank register and index enable flag of system registers are automatically saved by the hardware. Use the program to save another system register as described in the example if necessary.

The PEEK and POKE instructions can be used to save or restore the contents of the system registers or other registers as described in the example.

To save the contents of a register, a transfer instruction (LD r, LD m, ST m, or ST r) can be used in addition to PEEK and POKE. If a transfer instruction is used to save the contents of a register when the row address of the general-purpose register is not defined at interrupt acceptance, the data memory address is hard to specify.

If the general-purpose register address is not defined when the transfer instruction is used to save the contents of the general-purpose register, the address to be saved also becomes undefined. In this case, use of the general-purpose register should be fixed at least in the interrupt permission routine.

However, because the address of the register file controlled by the PEEK or POKE instruction is specified regardless of the contents of the general-purpose register and because addresses 40H-7FH of the register file overlap with the bank data memory, each system register can be saved only by specifying the bank.

In the example, the PEEK or POKE instruction is used to save the contents of the window register and general-purpose register pointer. Then, the general-purpose register is respecified to row address 07H of BANK0 and the ST instruction is used to save another system register.

Fig. 11-4 illustrates register content saving using the PEEK and POKE instructions.

11.6.2 Restoration Processing

This section describes an example of restoration.

To restore the contents of a register, reverse the procedures for register saving explained in **Section 11.6.1**.

Because an interrupt is always accepted in an interrupt permitted state (EI state), the EI instruction must be executed before the RETI instruction.

The EI instruction sets the interrupt enable flip-flop to 1 after the next RETI instruction is executed. Therefore, control is returned to the program before an interrupt is accepted, then the program enters an interrupt permitted state.

11.6.3 Notes on Interrupt Processing Routine

Note the following regarding the interrupt processing routine:

(1) Data saved by hardware

All bank registers and index enable flags are reset to 0 after being saved in the interrupt stack.

(2) Data saved by software

Data saved by software is not reset after being saved.

Program status words such as the BCD flag, compare flag, carry flag, zero flag, and memory pointer enable flags keep their preacceptance values. Initialize these program status words if necessary.

Example Saving the status in an interrupt processing routine

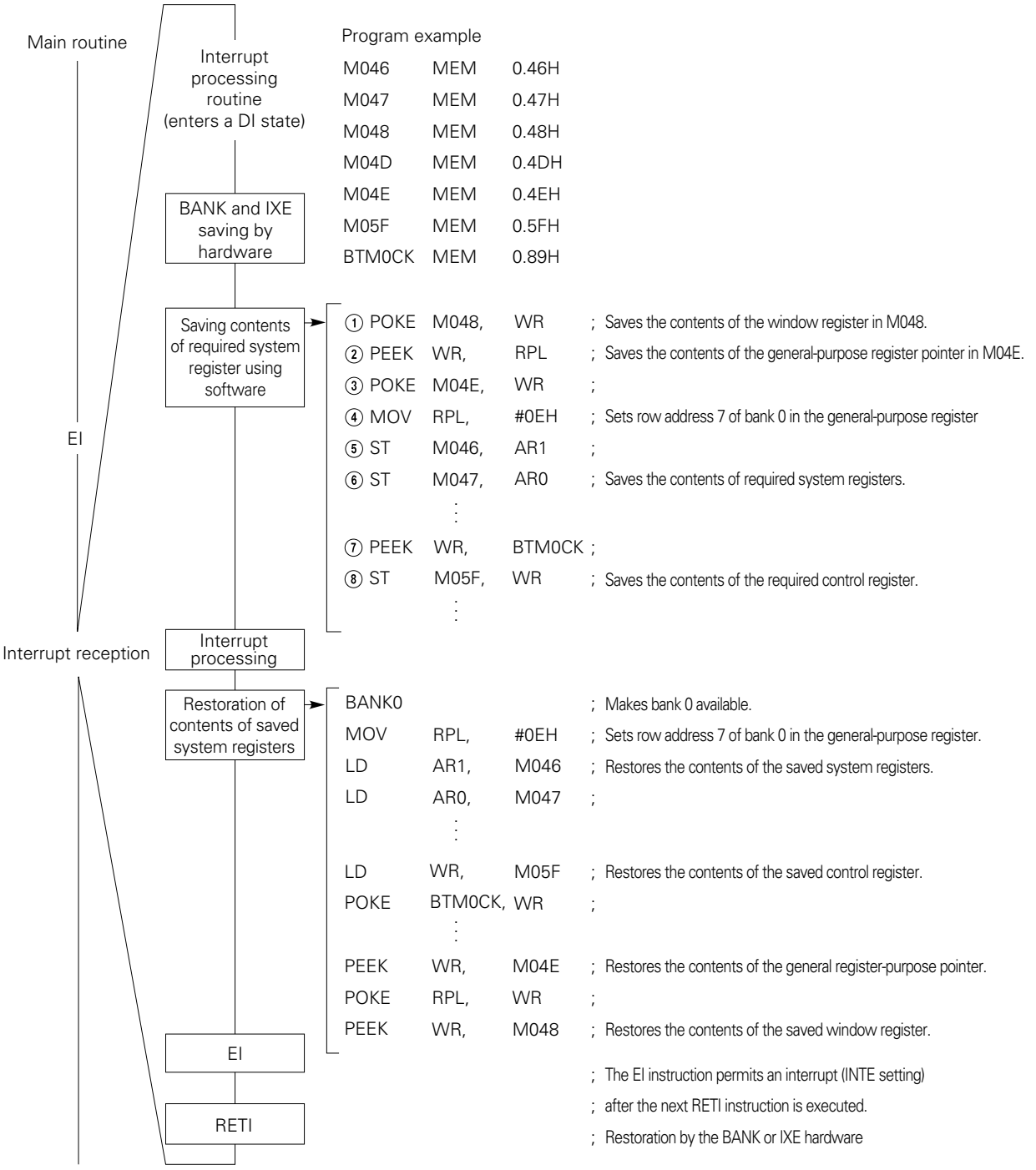
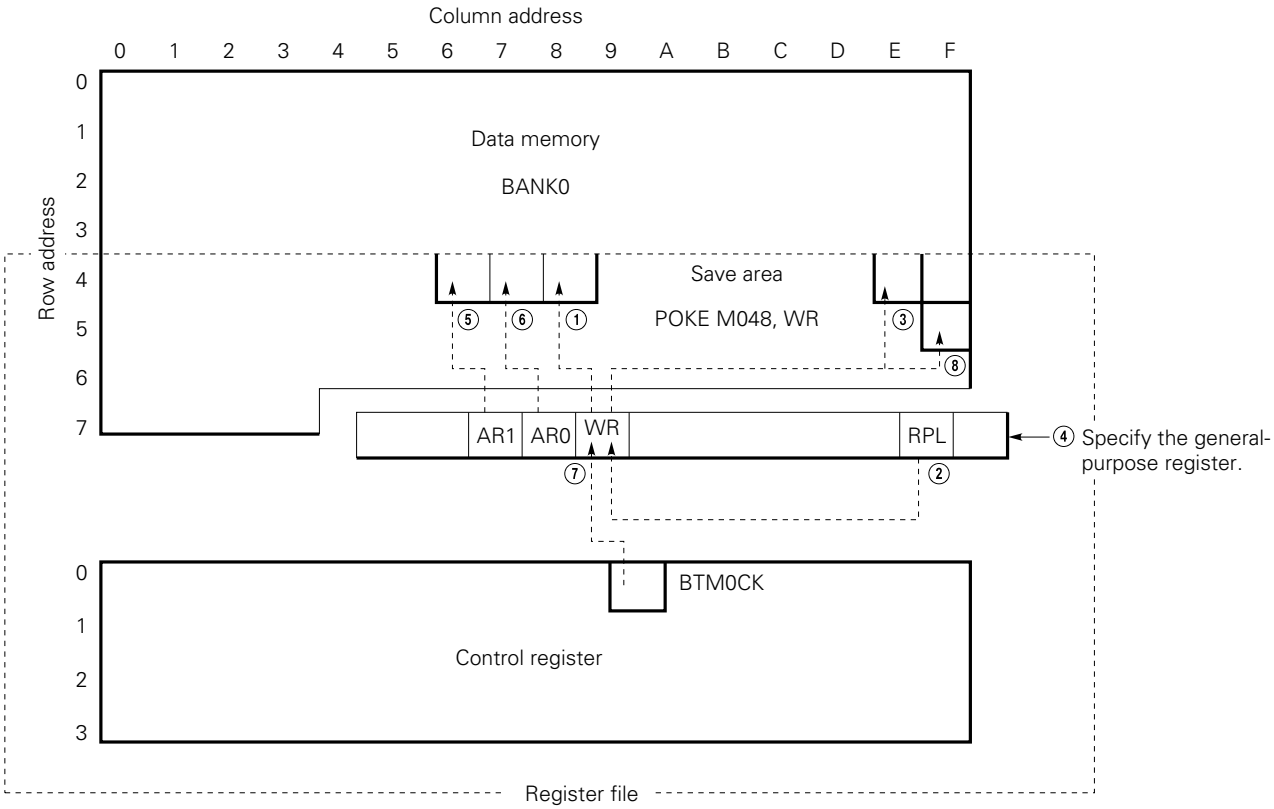


Fig. 11-4 Saving the System or Control Register Using the Window Register

Numbers ① to ⑧ correspond to the numbers in the program example.



11.7 EXTERNAL INTERRUPTS (INT_{NC} PIN, $\overline{V_{SYNC}}$ PIN)

There are two external interrupt sources: INT_{NC} and $\overline{V_{SYNC}}$.

An interrupt request is issued when a rising or falling edge is input to the INT_{NC} or $\overline{V_{SYNC}}$ pin.

11.7.1 Configuration

Fig. 11-5 shows the configurations of the INT_{NC} and $\overline{V_{SYNC}}$ interrupts.

As shown in Fig. 11-5, the INT_{NC} and $\overline{V_{SYNC}}$ signals are input to the INTNC or INTVSYN latch and to edge detectors.

The edge detectors output their respective interrupt request signals according to the inputs from the pin and the status of the IEGNC or IEGVSYN flip-flop.

The IEGNC flip-flop and IEGVSYN flip-flop correspond to the IEGNC flag and IEGVSYN flag, respectively, in the interrupt edge selection register (INTEDGE: address 1FH) of the control register.

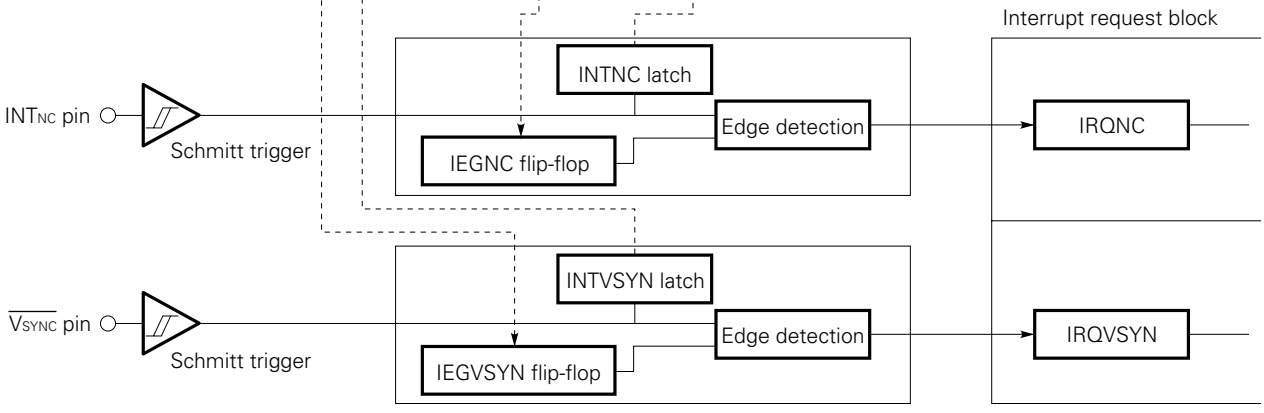
The INTNC latch and INTVSYN latch correspond to the INTNC flag and INTVSYN flag, respectively, in the interrupt-pin-level judge register (INTJDG: address 0FH) of the control register.

The Schmitt triggers at the INT_{NC} and $\overline{V_{SYNC}}$ inputs prevent pulses operations due to noise. These pins do not accept pulses of 1 μs or less.

A minimum pulse width can be set for the INT_{NC} pin. See **Section 9.10**.

Fig. 11-5 INT_0 Pin and INT_1 Pin Configurations

Control register								
Name	Interrupt edge select (INTEDGE)				Interrupt pin level judge (INTJDG)			
Address	1 FH				0 FH			
Bit	b_3	b_2	b_1	b_0	b_3	b_2	b_1	b_0
Flag symbol	0	IEGVSYN	0	IEGNC	0	INTVSYN	0	INTNC



11.7.2 Functions

An interrupt can be issued when either a rising or falling edge is input to the INT_{NC} or $\overline{V_{SYNC}}$ pin.

Use the IEGNC or IEGVSYN flag in the interrupt edge select register of the control register to select the rising or falling edge.

Table 12-2 shows the relationship between the IEGNC and IEGVSYN flags and the active edges of interrupt requests.

Note the following:

If the IEGNC or IEGVSYN flag is used to switch the interrupt request edge, an interrupt request signal may be issued at the moment of switching.

Suppose that the IEGNC flag is set to 0 (falling edge) and a high level is input to the INT_{NC} pin as shown in Table 11-3. At this time, if the IEGNC flag is set to 1, the edge detector determines that a rising edge has been input and therefore issues an interrupt request.

See **Section 11.2** for operations after interrupt request issuance.

Because the signals input to the INT_{NC} and $\overline{V_{SYNC}}$ pins are input to the INTNC and INTVSYN latches as shown in Fig. 11-5, the input signal levels can be detected by reading the INTNC and INTVSYN flags.

Because the INTNC and INTVSYN flags are set or reset regardless of interrupts, they can be used as 2-bit general-purpose input ports when the corresponding interrupt functions are not used.

If interrupt is not permitted, the flags can be used as general-purpose ports that can detect a rising or falling edge by reading the interrupt request flags (IRQNC or IRQVSYN). However, because the interrupt request flags are not automatically reset in this case, they must be reset by the program.

Table 11-2 IEGNC and IEGVSYN Flags and Interrupt Request Issuance Edges

Flag values		Active edges of interrupt request pins	
IEGNC	INTVSYN	INT_{NC} pin	$\overline{V_{SYNC}}$ pin
0	0	Rise	Rise
0	1	Rise	Fall
1	0	Fall	Rise
1	1	Fall	Fall

Table 11-3 Interrupt Request Issuance by IEGNC Flag Change

IEGNC or IEGVSYN flag change			INT_{NC} or $\overline{V_{SYNC}}$ pin	Whether interrupt request is issued	IRQNC flag
1 (Fall)	→	0 (Rise)	Low	Not issued	No change
			High	Issued	Set
1 (Rise)	→	0 (Fall)	Low	Issued	Set
			High	Not issued	No change

11.8 INTERNAL INTERRUPT (TIMER, SERIAL INTERFACE)

There are two types of internal interrupts: the timer interrupt and the serial interface interrupt.

11.8.1 Timer Interrupt

The timer interrupt function can issue interrupt requests at a specified time interval.

An interval of 100 ms, 20 ms, or 5 ms can be selected.

See **Chapter 12** for details.

11.8.2 Serial Interface Interrupt

The serial interface interrupt function can issue an interrupt request when a serial out or serial in operation terminates.

Therefore, interrupt requests are mainly issued by the serial clock.

See **Chapter 16** for details.

11.9 MULTIPLE INTERRUPTS

The multiple interrupt function is used to process interrupt C or D while another interrupt from source A or B is being processed as shown in Fig. 11-6.

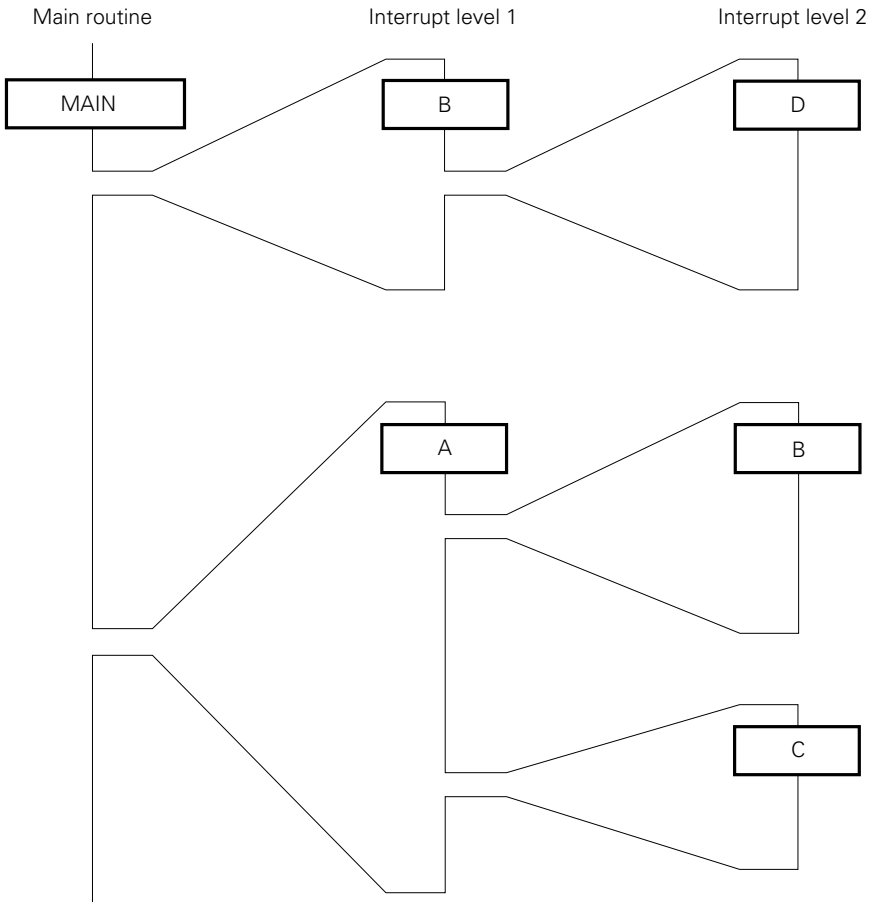
The interrupt depth at this time is called the interrupt level.

Note the following regarding the multiple interrupt function.

(1) Interrupt source priorities
(2) Interrupt level restriction by an interrupt stack
(3) Interrupt level restriction by the address stack register
(4) System or control register saving

See **Sections 11.9.1** to **11.9.4** for details.

Fig. 11-6 Example of Multiple Interrupts



11.9.1 Interrupt Source Priorities

When using the multiple interrupt function, the priorities of interrupt sources must be determined.

For example, if the interrupt sources are A, B, C, and D, the following priorities can be specified: A = B = C = D or A < B < C < D.

If A = B = C = D, the main routine always accepts interrupts A, B, C, and D. However, if interrupt C is accepted, interrupts A, B, and D are inhibited, making the multiple interrupt function unusable.

If the priorities are A < B < C < D, interrupt C should be processed with the first priority even if interrupt A or B is being processed. In this case, processing of interrupt D has the same priority as interrupt C.

The priorities can be set to hardware or software priorities by using the interrupt permission flags. **Section 11.3** describes the hardware and software priorities.

To determine priorities at multiple interrupts, interrupt sources A and B are assumed have no priority and source A is assumed to issue requests at 10 ms intervals. The interrupt processing time is assumed to be 4 ms. Source B is assumed to issue requests at 2 ms intervals. Lastly, the interrupt processing time is assumed to be 1 ms.

Under these conditions, if interrupt A is issued by an interrupt request from A while interrupt B is being processed, and the priorities of A and B are not determined, several interrupts from B will not be executed.

Because an interrupt is generally used for emergency processing, the A < B priority should be set in the program to prevent interrupt A while interrupt B is being processed and accept interrupt B while interrupt A is being processed.

When using the multiple interrupt function for non-emergency purposes, priorities need not be determined. However, if the number of existing interrupt sources exceeds the multiple interrupt level limit described in **Section 11.9.2** or **11.9.3**, be sure to determine priorities so that the interrupt level is not exceeded.

11.9.2 Interrupt Level Restriction by Interrupt Stack

The contents of the bank register of the system register and index enable flag are automatically saved in the interrupt stack.

Fig. 11-7 (a) shows the interrupt stack operation.

The contents of all bank registers and index enable flags are reset when they are saved in the interrupt stack.

Because there are two levels of interrupt stacks, if multiple interrupts of more than two levels are issued, the contents of the bank register and index enable flag are not restored normally as shown in Fig. 11-7 (b).

In other words, multiple interrupts of more than two levels cannot be used.

However, if the bank register and index enable flag are fixed in a main routine permits interrupts and multiple interrupts have clear priorities as shown in Fig. 11-8, multiple interrupts of two levels or more can be used by using subroutine return instruction RET.

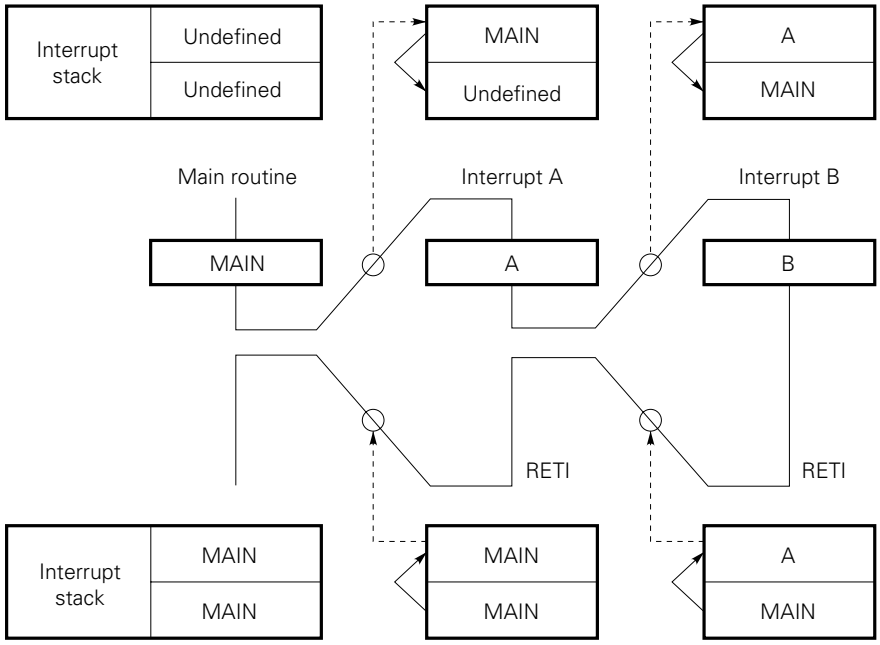
For multiple interrupts of more than two levels, operations of the device and emulator differ as shown in Figs. 11-8 and 11-9.

At interrupt stack, the device operation is the sweep-off type and the emulator operation is the rotation type.

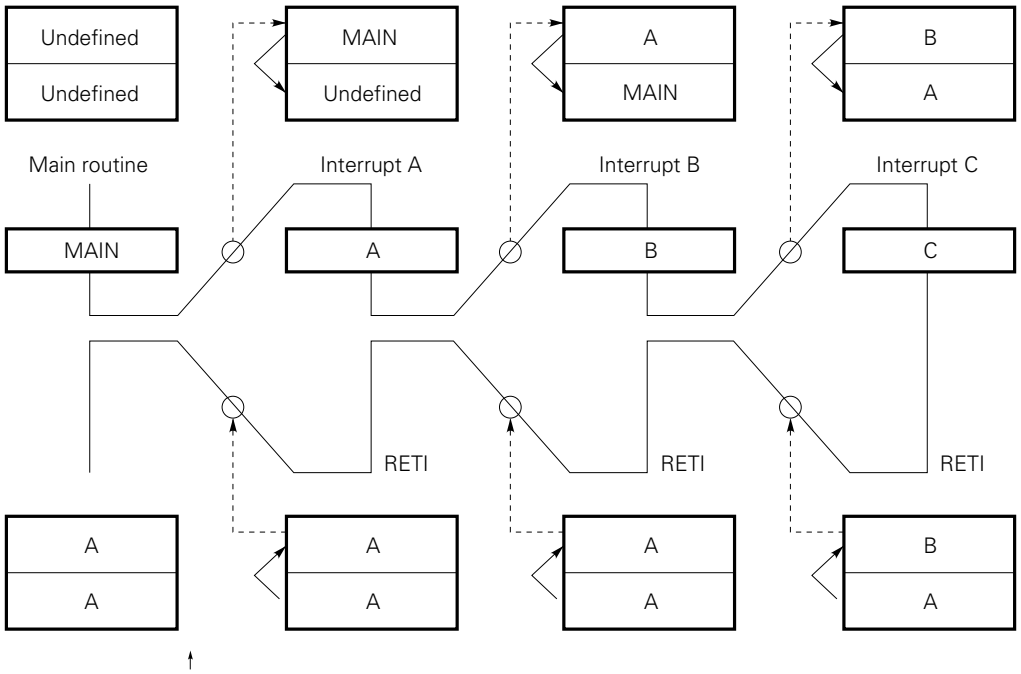
Use the RET instruction as the last restoration instruction when using multiple interrupts of more than two levels. RETI and RET instructions operate in the same manner except when restoring the contents of the interrupt stack.

Fig. 11-7 Interrupt Stack Operation at Multiple Interrupts

(a) Multiple level-2 interrupts

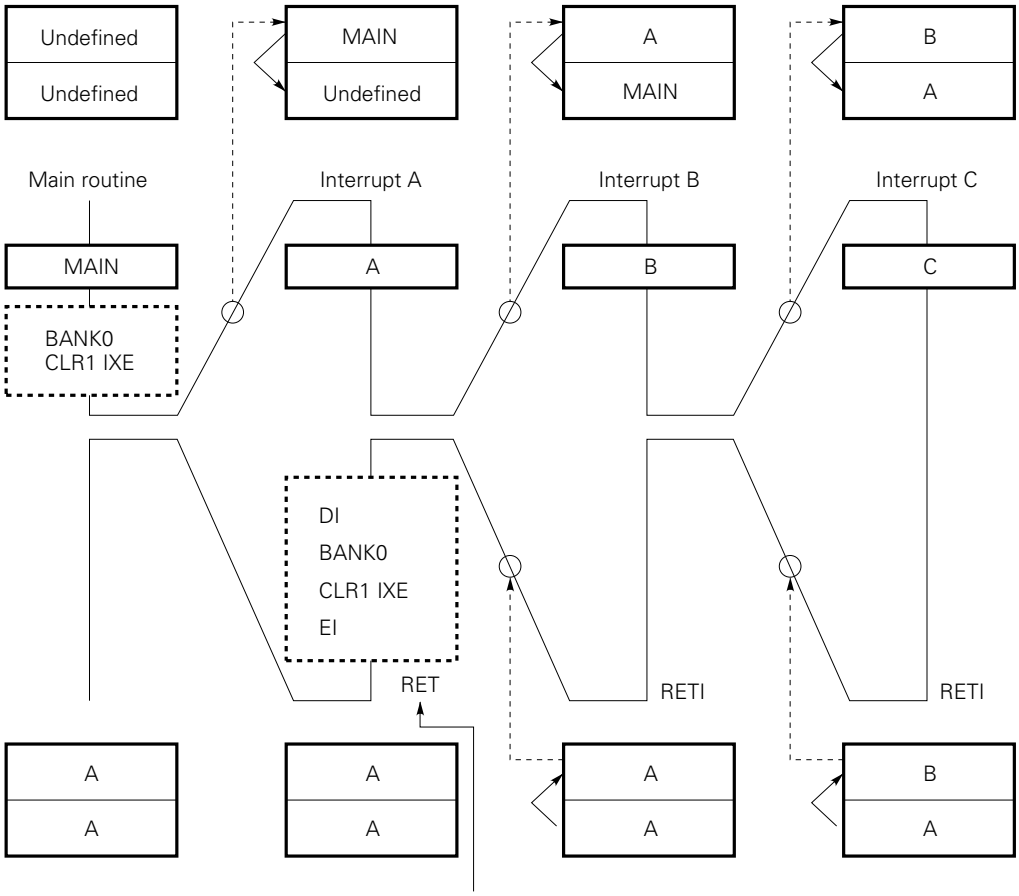


(b) Multiple level-3 interrupts



If control is returned to the main routine at this time, BANK and IXE of interrupt A are restored and the main routine operates abnormally.

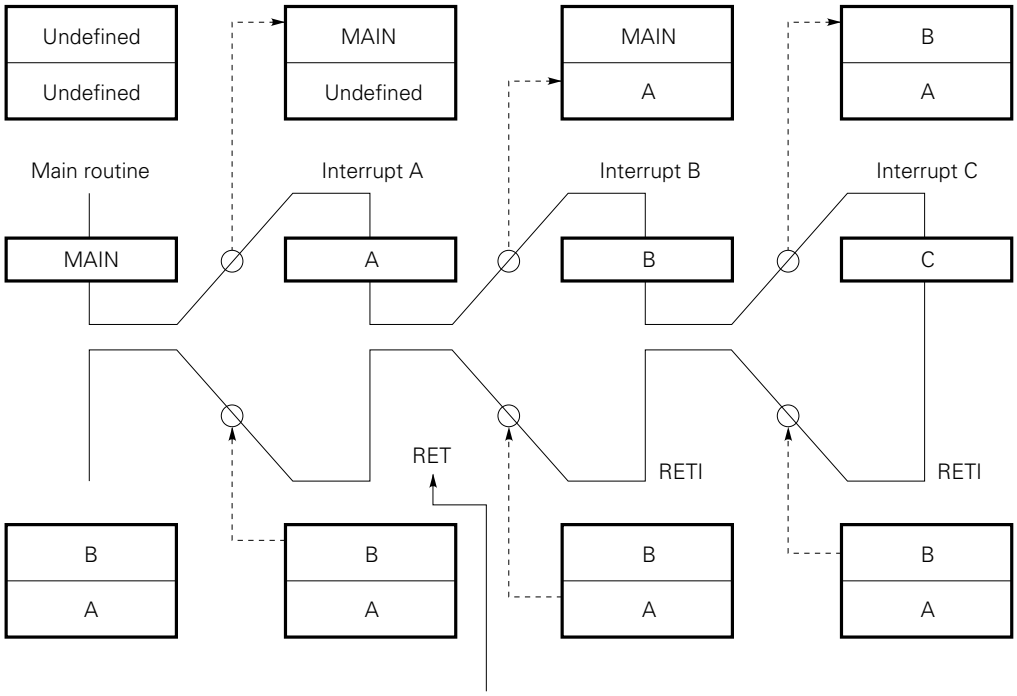
Fig. 11-8 Example of Using Multiple Level-3 Interrupts



To interrupt A, be sure to set a lower priority than interrupts B and C. Fix the bank register and index enable flag (BANK0 and IXE = 0 in this example) in the main routine that permits interrupt A. This processing enables the use of RET instructions for multiple interrupts of three levels after specifying the bank register and index enable flag of the main routine.

If the bank register and index enable flag at interrupt A are exactly the same as those of the main routine, the RETI instruction can be used. However, because the operation of the 17K series emulator differs as shown in Fig. 11-9, the RETI instruction cannot be used for debugging.

Fig. 11-9 Interrupt Stack Operation when 17K Series Emulator is Used



If the RETI instruction is used on the emulator, the contents of the bank register and index enable flag of interrupt B are restored.

11.9.3 Interrupt Level Restriction by Address Stack Register

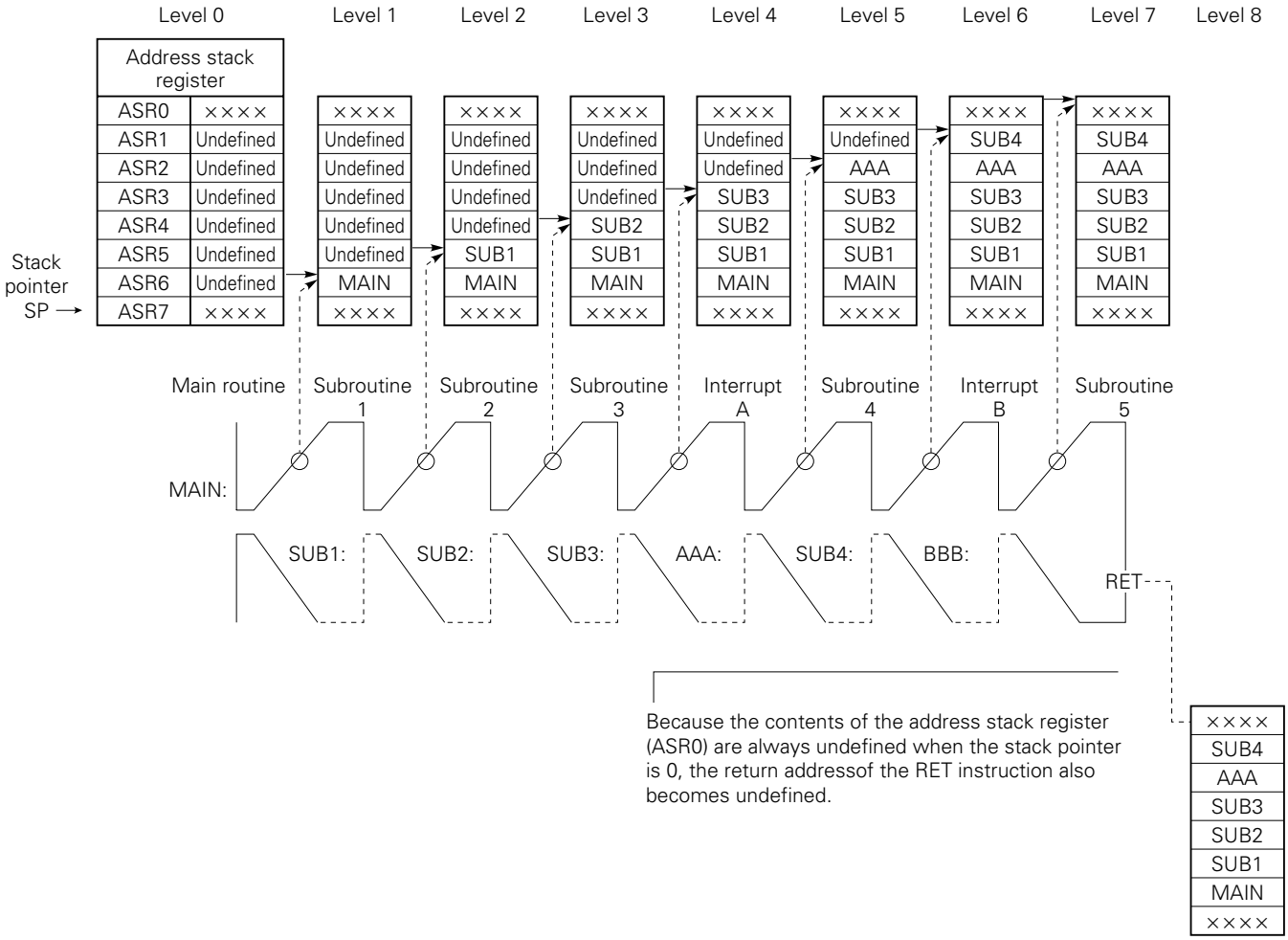
The return address at control return from interrupt processing is automatically saved in the address stack register.

The address stack register can use the six levels from ASR0 to ASR5 as described in **Chapter 4**. Because the interrupt sources are the INTNC pin, timer, $\overline{\text{VSYNC}}$ pin, and serial interface, the multiple interrupt level is unlimited when the address stack register is used only for interrupts.

However, because the address stack register is also used to save the return address at subroutine calling, multiple interrupt levels are limited according to the levels of the address stack register used for subroutine calling.

For example, if four levels are used for subroutine calling, only two levels of the multiple interrupts shown in Fig. 11-10 can be used.

Fig. 11-10 Address Stack Register Operation



11.9.4 Saving the Contents of System and Control Registers

The contents of system and control registers must be saved before using the multiple interrupt function. The contents of these registers change during interrupt processing.

An area must be obtained for these contents for each interrupt source.

An interrupt being accepted and interrupts with lower priorities must be inhibited, and interrupts with higher priorities must be permitted.

Because an interrupt with a high priority is an emergency interrupt, it should have first priority. Therefore, the contents of system and control registers should be saved after permitting an interrupt with a high priority.

The following example describes processing of the interrupt processing routine to enable an interrupt with a high priority and to save the contents of system and control registers:

Example **Example of permitting an interrupt and saving register contents at multiple interrupts**

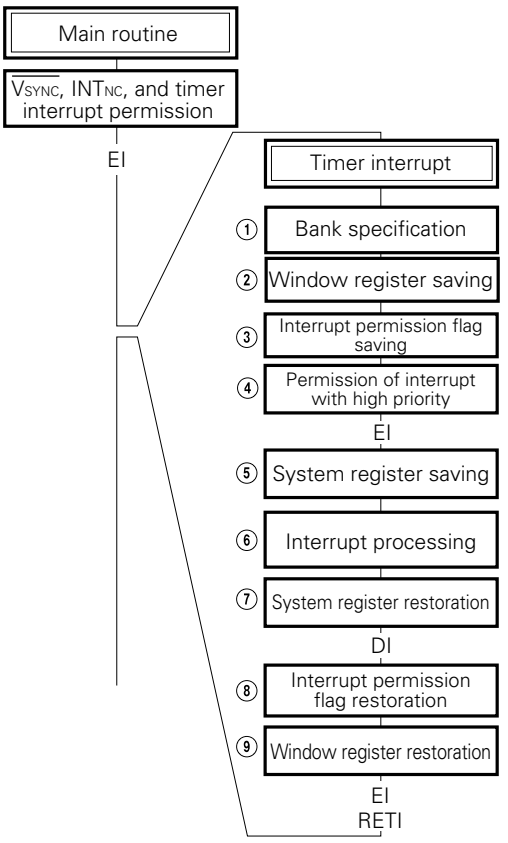
Use the INT_{NC} pin, $\overline{V_{SYNC}}$ pin, and timer interrupts with the following software priorities:

$\overline{V_{SYNC}}$ pin > timer > INT_{NC} pin

A timer interrupt is assumed to be accepted in the first level.

The figure below shows an example program and flowchart for this processing.

Flowchart **Program example**




```
MOV   WR,      #0111B   ; Permit timer, INTNC pin, and VSYNC pin interrupts
POKE  INTPM,   WR       ; using the interrupt permission register.
                        ; The contents of the bank register, index enable flag,
                        ; and program counter are automatically saved.

BANK1 (built-in macro)  ; Specifies that the data is saved in BANK1.

POKE  M1,      WR       ; Saves the window register.

PEEK  WR,      INTPM    ; Save the interrupt permission
POKE  M2,      WR       ; register in data memory M2.

MOV   WR,      #0100B   ; Permit a VSYNC pin interrupt using
POKE  INTPM,   WR       ; the interrupt permission register.

PEEK  WR,      RPL      ; Save the contents of system and
POKE  M3,      WR       ; control registers other than window
                        ; and interrupt permission registers.

PEEK  WR,      M3       ; Restore the contents of system
POKE  INTPM,   WR       ; and control registers other than window
                        ; and interrupt permission registers.

PEEK  WR,      M2       ; Restore the contents of the interrupt
POKE  INTPM,   WR       ; permission register.

PEEK  WR,      M1       ; Restores the contents of the window register.
```

In ①, specify the data memory bank containing the contents of the system register.

Because the bank becomes BANK0 when an interrupt is accepted, if the data is saved in BANK0, this instruction is not necessary.

In ②, save the contents of the window register in data memory M1.

Because the POKE instruction is used, the address of data memory M1 should be 40H or more. Because the window register is used as a work area for subsequent data saving, its contents must be saved first.

In ③, save the interrupt permission flags (IPNC, IPBMT0, and IPVSYN) set when interrupts are accepted. In this example, all INT_{NC} pin, $\overline{V_{SYNC}}$ pin, and timer interrupts must be permitted when control is returned to the main routine in this save operation. The priority of the timer interrupt is higher than that of the INT_{NC} pin. Therefore, if the timer interrupt is accepted while the INT_{NC} pin interrupt is being processed, control should be returned with the INT_{NC} pin interrupt inhibited.

In ④, permit $\overline{V_{SYNC}}$ interrupt with a lower priority than the timer interrupt. Then, use the EI instruction to permit all interrupts.

Because processing in ①, ②, ③, and ④ must be executed with an interrupt inhibited, the $\overline{V_{SYNC}}$ interrupt with the highest priority is also inhibited during this processing.

In ⑤ and ⑥, save and restore the contents of the system and control registers. At this time, interrupts with high priorities can be enabled.

If the contents of the registers are saved when a $\overline{V_{SYNC}}$ interrupt with a high priority is accepted, the contents of the system and control registers do not change when control is returned from $\overline{V_{SYNC}}$ interrupt processing.

In ⑦ and ⑧, return the contents of the interrupt permission flag and window register.

At this time, all interrupts should be inhibited.

If a timer interrupt is issued when the instruction in ⑦ that permits an interrupt is executed in an EI state, the contents of the window register in ⑧ are not restored but are saved again in ②. At this time, the contents of the window register cannot be restored.

12. TIMER

The timer functions are used to manage the time in creating programs.

12.1 TIMER CONFIGURATION

Fig. 12-1 shows the configuration of the timer.

The timer consists of two blocks, timer carry flip-flop (timer carry FF) block and timer interrupt block, as shown in Fig. 12-1.

The clock generation circuit, which specifies time intervals for the timer carry FF and timer interrupts, consists of an 8 MHz frequency divider, selector A, selector B, bias circuit, and a timer mode select register (BTM0CK at address 09H), which is a control register.

12.1.1 Timer Carry FF Block Configuration

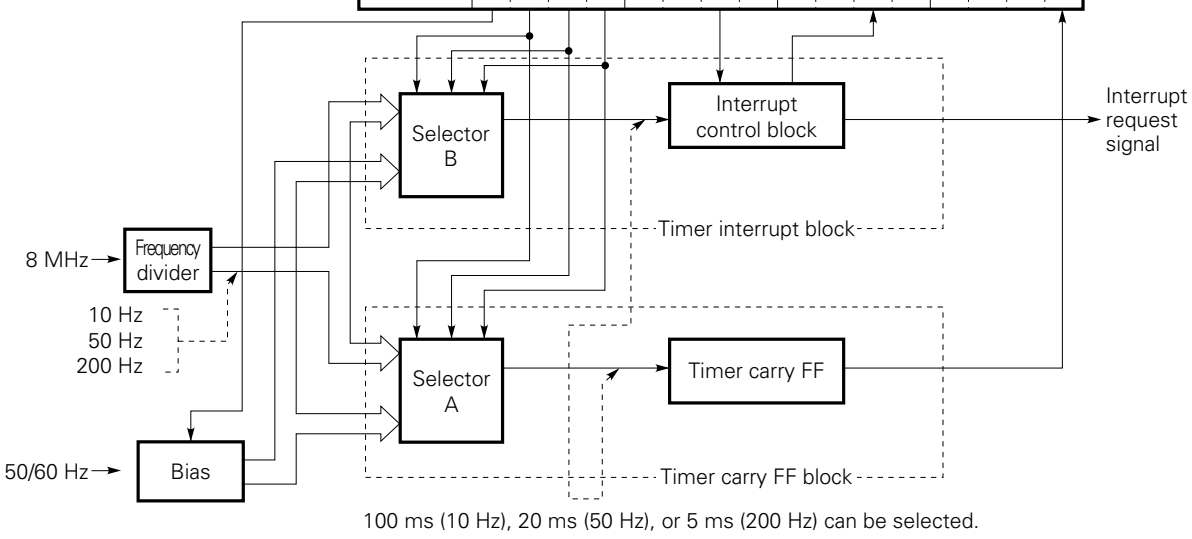
The timer carry FF block consists of selector A, timer carry FF, a timer carry FF judge register (BTM0CYJDG at address 17H), which is a control register, as shown in Fig. 12-1.

12.1.2 Timer Interrupt Block Configuration

The timer interrupt block consists of selector B, an interrupt control block, an interrupt permission register (INTPM at address 2FH), which is a control register, and an interrupt request register (INTREQ at address 3FH), as shown in Fig. 12-1.

Fig. 12-1 Timer Configuration

Control register																
Register	Timer mode select (BTM0CK)				Interrupt permission (INTPM)				Interrupt request (INTREQ)				Timer carry FF judge (BTM0CYJDG)			
Address	09H				2FH				3FH				17H			
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
Flag symbol	BTM0ZX	BTM0CK2	BTM0CK1	BTM0CK0	IPSIO0	IPVSYN	IPBTM0	IPNC	IRQSIO0	IRQVSYN	IRQBTM0	IRQNC	0	0	0	BTM0CY



100 ms (10 Hz), 20 ms (50 Hz), or 5 ms (200 Hz) can be selected.

12.2 TIMER FUNCTIONS

There are two timer functions, timer carry FF check and timer interrupt.

The timer carry FF check function performs time management by checking, by program, the state of the timer carry FF, which is set at constant intervals. The timer interrupt function performs time management by requesting an interrupt at constant intervals.

The timing at which the timer carry FF is set to 1 or the timer interrupt is requested is controlled by the timer interval set pulse output from selector A or B, respectively.

The timer interval set pulse can be specified as 10 Hz (100 ms), 50 Hz (20 ms), or 200 Hz (5 ms) by setting the appropriate data in the timer mode select register.

The timer mode select register is used to specify the time base mode (internal timer mode or external timer mode) for selectors A and B. The internal timer mode uses pulses generated by dividing the device's operating frequency (8 MHz). The external timer mode uses 50 or 60 Hz supplied at the $P0B_2$/TMIN pin.

The timer mode select register is again used to specify whether to divide the frequency of the pulse supplied at the $P0B_2$/TMIN pin by 5 or 6.

The timer interval set pulse is specified by combining the timer carry FF and timer interrupt.

Fig. 12-2 shows the relationships between the timer mode select register and timer interval set pulse.

In the internal timer mode, the timer interval set pulse is generated by dividing the device's operating frequency (8 MHz). If the frequency deviates from the correct value (8 MHz), the timer interval set pulse will also deviate at the same ratio.

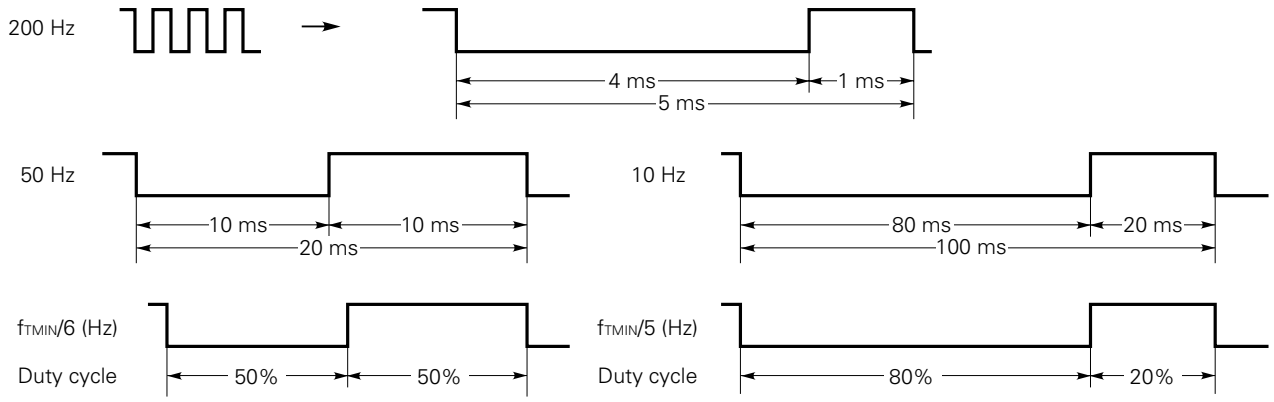
Fig. 12-2 Relationship Between the Timer Mode Select Register and Timer Interval Set Pulse

Control register				
Register	Timer mode select (BTM0CK)			
Address	09H			
Read/Write	R/W			
Bit	b_3	b_2	b_1	b_0
Flag symbol	BTM0ZX	BTM0CK2	BTM0CK1	BTM0CK0

			Selection of frequency (time) of the timer carry FF set pulse		Selection of frequency (time) of the timer interrupt pulse	
0	0	0	10 Hz (100 ms)	Internal timer	200 Hz (5 ms)	Internal timer
0	0	1	200 Hz (5 ms)	Internal timer	10 Hz (100 ms)	Internal timer
0	1	0	10 Hz (100 ms)	Internal timer	50 Hz (20 ms)	Internal timer
0	1	1	200 Hz (5 ms)	Internal timer	50 Hz (20 ms)	Internal timer
1	0	0	f_{TMIN}/5 Hz (5/f_{TMIN} s)	External timer	200 Hz (5 ms)	Internal timer
1	0	1	200 Hz (5 ms)	Internal timer	f_{TMIN}/5 Hz (5/f_{TMIN} s)	External timer
1	1	0	f_{TMIN}/6 Hz (6/f_{TMIN} s)	External timer	200 Hz (5 ms)	Internal timer
1	1	1	200 Hz (5 ms)	Internal timer	f_{TMIN}/6 Hz (6/f_{TMIN} s)	External timer

f_{TMIN} is the input frequency (50 or 60 Hz) at the $P0B_2$/TMIN pin.

0	Disables the bias circuit.
1	Enables the bias circuit.



12.3 TIMER CARRY FLIP-FLOP (TIMER CARRY FF)

The timer carry FF is set to 1 by the positive-going edge of the timer carry FF set pulse specified by the timer mode select register.

The content of the timer carry FF corresponds to the lowest bit (BTM0CY flag) of the timer carry FF judge register on a one-to-one basis, and when the timer carry FF is set to 1, the BTM0CY flag is also set to 1 at the same time.

The BTM0CY flag is reset to 0 by the PEEK instruction when it reads the content of the window register (Read & Reset).

When the BTM0CY flag is reset to 0, the timer carry FF is also reset to 0 at the same time.

Reading the BTM0CY flag by program can create a timer that operates at intervals of the time specified in the timer mode select register.

Section 12.3.1 gives an example of a program used to read the BTM0CY flag.

When using the timer carry FF, observe the following point.

A power-on reset disables the timer carry FF from being bet. It cannot be set until the PEEK instruction is issued to read the content of the BTM0CY flag.

0 is read in when the BTM0CY flag is read-accessed for the first time after a power-on reset. Once it is reset, the timer carry FF is set to 1 at intervals of the time specified in the timer mode select register.

The timer carry FF also controls the timing of a CE reset.

To put in another way, once the CE pin goes from a low to a high, a CE reset occurs at the same time the timer carry FF is set.

Therefore, reading the content of the BTM0CY flag at a reset (power-on or CE reset) enables a power failure check. See **Section 12.4** and **Chapter 14** for details.

Because the BTM0CY flag is a read-only flag, writing to it with the POKE instruction does not affect the operation of the device at all. However, an error is reported by the 17K series assembler.

12.3.1 Example of Using the Timer Based on the BTM0CY Flag

An example of a program follows.

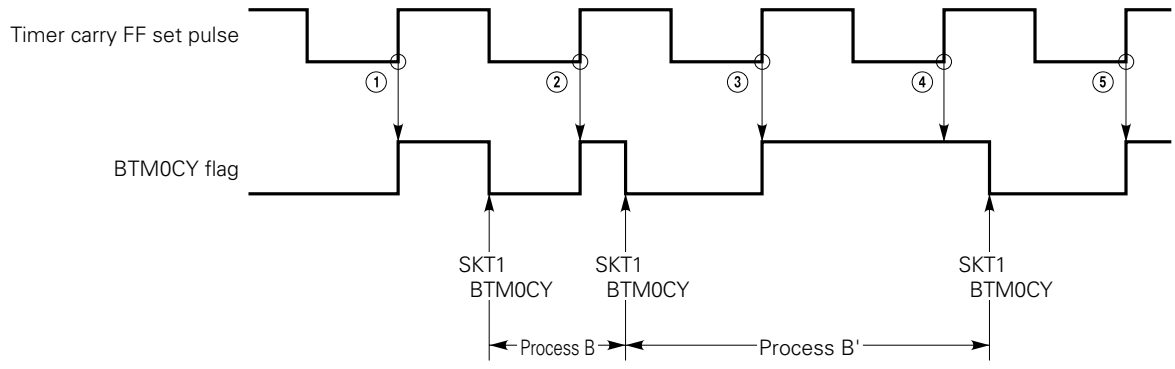
```
Example     INITFLG  NOT BTM0ZX, NOT BTM0CK2, NOT BTM0CK1, NOT BTM0CK0
                     ; Built-in macro
                     ; Specifies that the timer carry FF be set at intervals of 100 ms.
LOOP1:
    MOV  M1, #0110B
LOOP2:
    SKT1 BTM0CY      ; Built-in macro
                     ; Tests the BTM0CY flag.  Branches to NEXT if the flag is 0.
    BR   NEXT
    ADD  M1, #0100B  ; Adds 4 to data memory M1.
    SKT1 CY          ; Built-in macro
                     ; Tests the CY flag.
    BR   NEXT        ; Branches to NEXT if the flag is 0.
    Process A        ; Performs process A if the flag is 1.
    MOV  M1, #0110B
NEXT:
    Process B        ; Performs process B and branches to LOOP.
    BR   LOOP
```

This program performs process A at intervals of one second.
Note the following point (1) when creating this program.

(1) The time interval at which the BTM0CY flag is checked must be less than the time interval at which the timer carry FF is set to 1.

This is because if it takes 100 ms or longer to perform process B, it is impossible to detect when the timer carry FF is set, as shown in Fig. 12-3.

Fig. 12-3 BTM0CY Flag Check and Timer Carry FF



Because it takes long to perform process B' after it is detected that the BTM0CY flag is set at ②, it is impossible to detect when the BTM0CY flag is set at ③.

12.3.2 Timer Error Caused by the BTM0CY Flag

There are two types of timer error that can occur because of the BTM0CY flag. One type depends on the timing when the BTM0CY flag is checked, and the other type occurs when the timer carry FF setting interval is changed.

These types of timer error are detailed below.

(1) Timer error by BTM0CY flag check timing

As described in **Section 12.3.1**, the time interval at which the BTM0CY flag is checked must be less than the time interval at which the timer carry FF is set to 1.

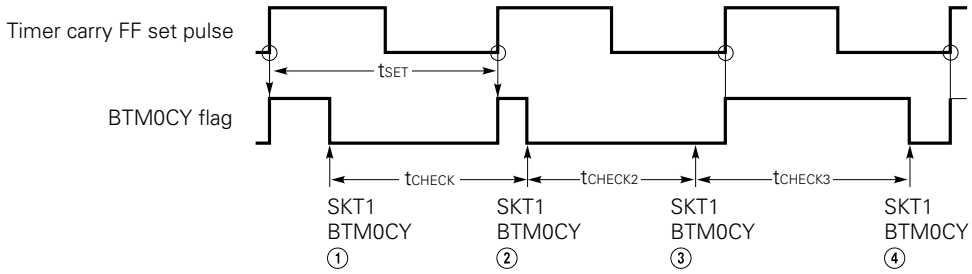
Suppose the time interval at which the BTM0CY flag is checked is t_{CHECK}, and the time interval (100 ms or 5 ms) at which timer carry FF is set is t_{SET}. The relationship between these two intervals must be as follows:

$$t_{CHECK} < t_{SET}$$

Under this condition, as shown in Fig. 12-4, the timer error that depends on the timing when the BTM0CY flag is checked is as follows:

$$0 < \text{error} < t_{CHECK}$$

Fig. 12-4 Timer Error That Depends on the Time Interval at Which the BTM0CY Flag Is Checked



As shown in Fig. 12-4, when the BTM0CY flag is checked at ②, it appears to be 1 and causes the timer to be updated. When it is checked at ③, it appears to be 0, and defers the updating of the timer until it is checked again at ④. In this case, the timer count is increased by t_{CHECK3}.

(2) Timer error that occurs when the timer carry FF setting time interval is changed

The timer carry FF setting time interval is specified by the BTM0CK2, BTM0CK1, and BTM0CK0 flags in the timer mode select register.

As shown in Fig. 12-1 and 12-2, the timer interval set pulse can be selected from 200 Hz, 10 Hz, and an external timer.

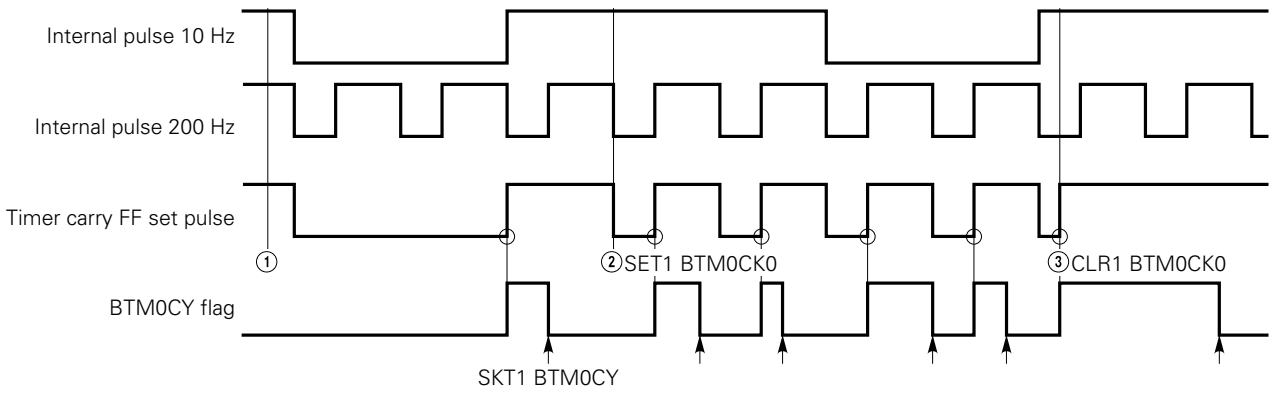
These three pulses operate independently.

Therefore, when the timer interval set pulse is switched using the BTM0CK2, BTM0CK1, or BTM0CK0 flag, a timer error occurs as shown in the following example.

Example

```
; ①
INITFLG   NOT BTM0ZX, NOT BTM0CK2, NOT BTM0CK1, NOT BTM0CK0
                    ; Built-n macro
  Process A         ; Specifies the timer carry FF set pulse as 10 Hz (100 ms).
; ②
SET1   BTM0CK0      ; Built-in macro
                    ; Specifies the timer carry FF set pulse as 200 Hz (5 ms).
  Process A
; ③
CLR1   BTM0CK0      ; Built-in macro
                    ; Specifies the timer carry FF set pulse as 10 Hz (100 ms).
```

With this coding, the timer carry FF set pulse is switched as shown below.



As shown above, when the timer carry FF setting time interval is switched, if a newly selected pulse goes low, it allows the BTM0CY flag to preserve its previous state (② in the figure). If the pulse goes high, it sets the BTM0CY flag to 1 (③ in the figure).

As shown in Fig. 12-5, if the timer carry FF setting time interval is switched, the timer error that occurs before the BTM0CY flag is set for the first time is as follows:

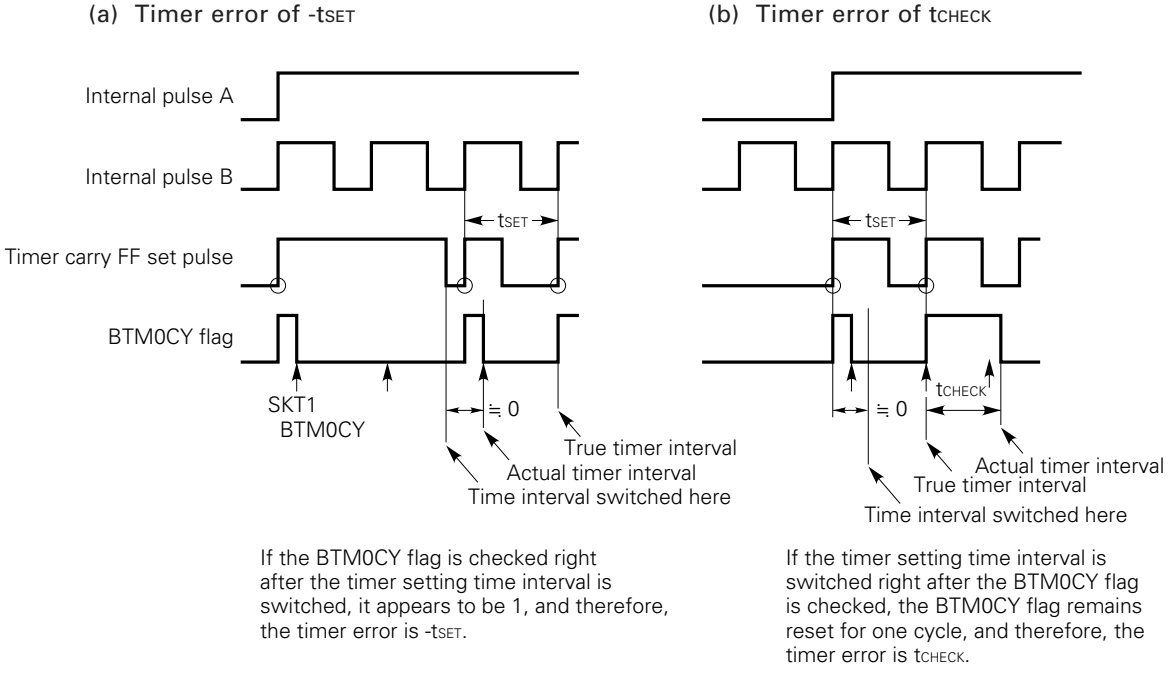
$-t_{SET} < \text{error} < t_{CHECK}$

where t_{SET} : Newly selected timer carry FF setting time interval
t_{CHECK} : Time interval at which the BTM0CY flag is checked

The internal pulses, 4 Hz, 10 Hz, 200 Hz, and 1 kHz, have a phase difference. However, this phase difference is less than a newly selected set pulse interval, and included in the timer error described above.

See **Section 12.6** for details about the phase difference of each pulse.

Fig. 12-5 Timer Error That Occurs When the Timer Carry FF Setting Time Interval Is Switched from A to B



12.4 CAUTIONS IN USING THE TIMER CARRY FF

The timer carry FF is used not only as a timer function but also as a reset sync signal at a CE reset. A CE rest occurs when the timer carry FF set pulse rises after the CE pin goes from a low to a high. Note the following points:

(1) The sum of the time used to update the timer and the time interval at which the BTM0CY flag is checked must be less than the timer carry FF setting time interval.

(2) If a created program needs a timer that operates at constant intervals regardless of a CE reset once a power-on reset occurs, the program must correct the timer at each CE reset.

(3) A check of the BTM0CY flag takes precedence over a reset sync signal for a CE reset. If both occur at the same time, a CE reset is delayed one cycle.

Sections **12.4.1** to **12.4.3** detail the above topics.

12.4.1 Timer Update Time and BTM0CY Flag Check Time Interval

As described in **Section 12.3.1**, the time interval t_{SET} at which the BTM0CY flag is checked must be less than the time interval at which the timer carry FF is set.

Even when the above requirement is satisfied, if the timer update process takes long, the timer process may not be performed correctly when a CE reset occurs.

To solve this problem, it is necessary to satisfy the following condition.

$$t_{CHECK} + t_{TIMER} < t_{SET}$$

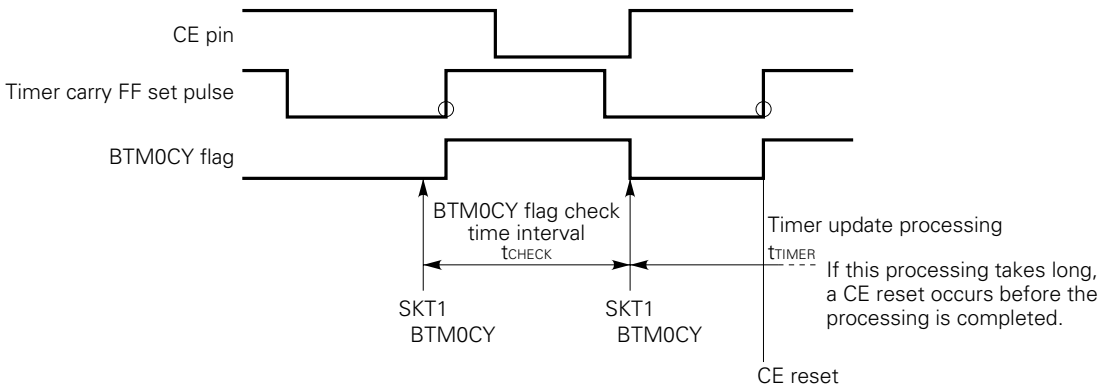
where t_{CHECK} : Time interval at which the BTM0CY flag is checked
t_{TIMER} : Timer update process time
t_{SET} : Time interval at which the timer carry FF is set

An example follows:

Example Timer update process and BTM0CY flag check time interval

```
START:                          ; Program address 0000H
      INITFLG  NOT BTM0ZX, NOT BTM0CK2, NOT BTM0CK1, NOT BTM0CK0
                                ; Built-in macro
                                ; Specifies the timer carry FF setting time interval as 100 ms.
TIMER:
      ; ①
      SKT1     BTM0CY flag ; Built-in macro
                           ; Tests the BTM0CY flag.
      BR       AAA         ; Branches to AAA, if 0.
      Timer update
      BR       TIMER
AAA:
      Process A
      BR       TIMER
```

The timing chart for the processing by the above program is shown below.



12.4.2 Correcting the Timer Carry FF at a CE reset

This section describes an example of correcting the timer at a CE reset.

If the timer carry FF is used both to check for power failure and as a timer, it is necessary to correct the timer at a CE reset, as explained in the following example.

The timer carry FF is reset to 0 at a power-on reset, and it is kept from being set until the BTM0CY flag is read-accessed using a PEEK instruction.

When the CE pin goes from a low to a high, a CE reset occurs in synchronization with the positive-going edge of the timer carry FF set pulse. At this point, the BTM0CY flag is set to 1 and becomes active.

Therefore, checking the state of the BTM0CY flag at a system reset (power-on reset or CE reset) can discriminate between a power-on reset and CE reset; if the flag is 0, it indicates a power-on reset, and if 1, it indicates a CE reset (power failure check).

A timer for ordinary time measurement must continue to operate even at a CE-reset.

Reading the BTM0CY flag for a power failure check could reset the BTM0CY flag to 0, thus losing a chance of detecting a set (1) state of the flag.

To skirt the above problem, it is necessary to update the timer for time measurement at a CE reset that occurs because of power failure.

See also **Section 14.6** for details about a power failure check.

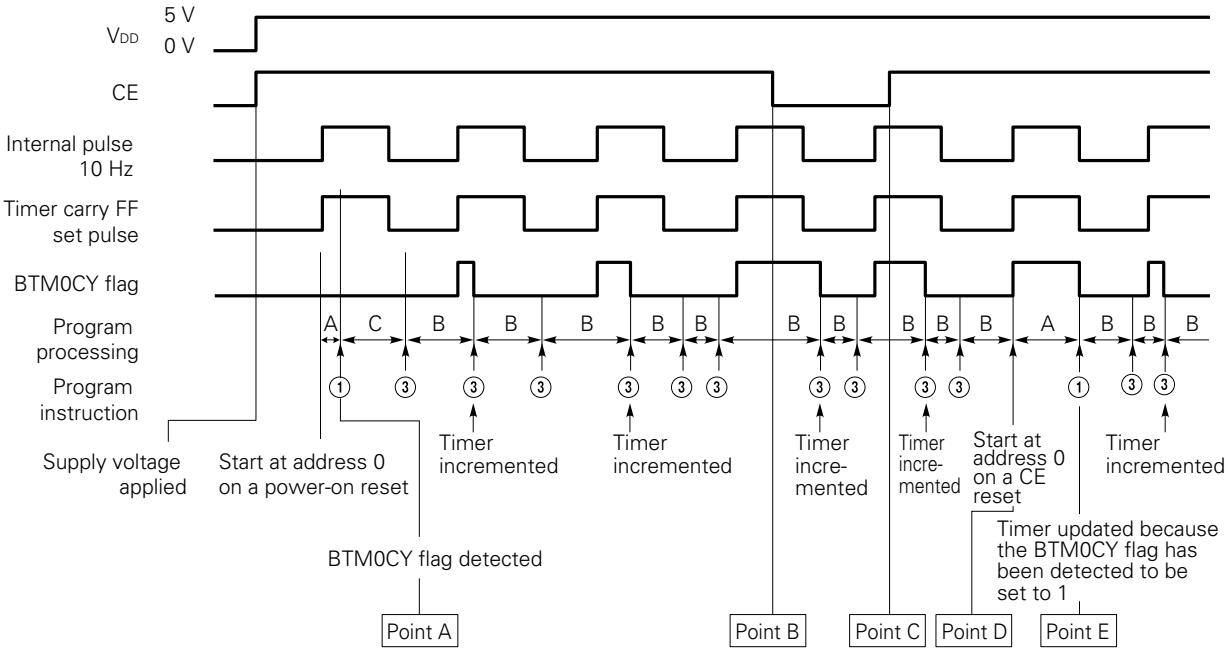
Example Correcting the timer at a CE reset

When using the timer carry FF for a power failure check and clock update

```
START:                          ; Program address 0000H
        Process A
        ; ①
        SKT1    BTM0CY          ; Built-in macro
                                ; Tests the BTM0CY flag.
        BR      INITIAL         ; Branches to INITIAL if 0 (power failure check)
BACKUP:
        ; ②
        Update the clock by 100 ms ; Correct the clock because of a backup (CE reset).
LOOP:
        ; ③
        Process B               ; While performing process B,
        SKF1    BTM0CY          ; tests the BTM0CY flag and updates the clock.
        BR      BACKUP
        BR      LOOP
INITIAL:
        INITFLG  NOT BTM0ZX, NOT BTM0CK2, NOT BTM0CK1, NOT BTM0CK0
                                ; Built-in macro
                                ; Because a power failure (power-on reset) is detected, the timer
                                ; carry FF setting time interval is set to 100 ms, and passes
                                ; control to process C.
        Process C
        BR      LOOP
```

Fig. 12-6 shows the timing chart for the above program.

Fig. 12-6 Timing Chart



As shown in Fig. 12-6, the positive-going edge of the internal 10 Hz pulse starts the program at 000H at a power-on reset.

When the BTM0CY flag is checked at point A, it appears to be reset to 0, thus indicating a power-on reset, because it is just after the power is turned on.

When a power-on reset occurs, process C is performed to specify the timer carry FF set pulse as 100 ms.

Because the timer carry FF was read-accessed once at point A, the BTM0CY flag is set to 1 at intervals of 100 ms.

Even when the CE pin goes low at point B and high at point C, the program continues updating the clock while performing process B unless a clock stop instruction has been executed.

Because the CE pin goes from a low to a high at point C, a CE reset occurs at point D, where the timer carry FF set pulse rises for the second time, thus starting the program at 0000H.

When the BTM0CY flag is checked at point E, a backup (CE reset) is detected because the BTM0CY flag is already set to 1.

As seen from the timing chart, if the clock is not updated by 100 ms at point E, the clock loses 100 ms each time a CE reset occurs.

If process A (power failure check) takes longer than 100 ms at point E, the program loses twice a chance of detecting when the BTM0CY flag is set; therefore, process A must be completed within 100 ms.

To put in another way, checking the BTM0CY flag for power failure must be performed before the timer carry FF is set after the program starts at 0000H.

12.4.3 If the BTM0CY flag is checked at the same time with a CE reset

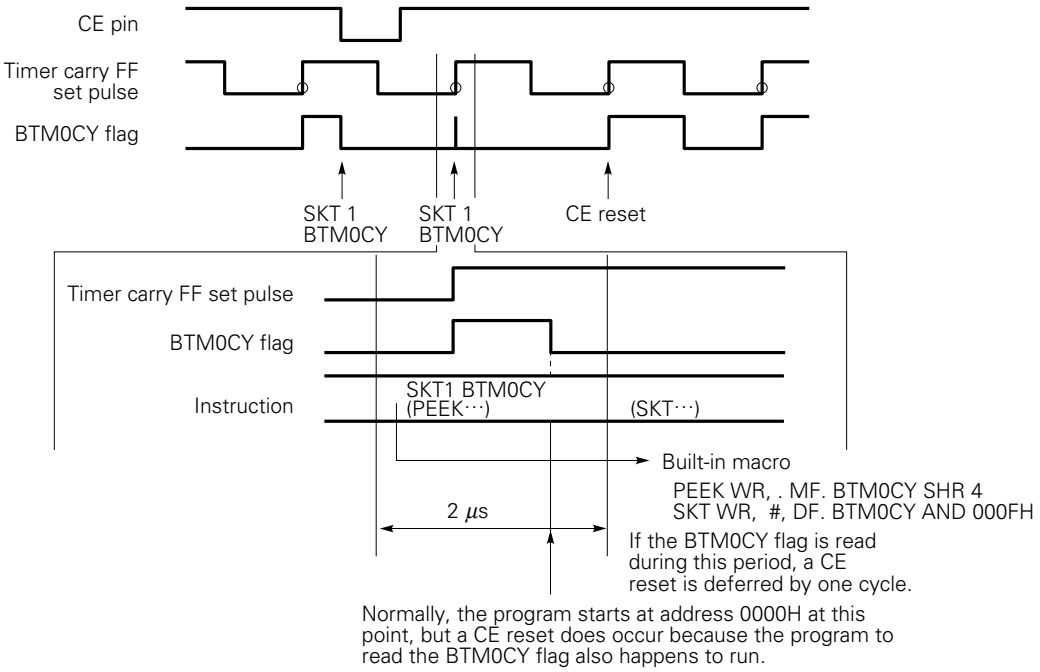
As described in **Section 12.4.2**, a CE reset occurs at the same time the BTM0CY flag is set to 1.

If the BTM0CY flag read instruction happens to occur at the same time a CE reset occurs, the BTM0CY flag read instruction takes precedence.

Once a CE pin goes from a low to a high, if the setting of the BTM0CY flag (at the positive-going edge of the timer carry FF set pulse) and a BTM0CY flag read instruction occur at the same time, a CE reset occurs next time the BTM0CY flag is set.

This operation is shown in Fig. 12-7.

Fig. 12-7 Operation That Occurs When a CE Reset and a BTM0CY Flag Read Instruction Coincide



So, if your program checks the BTM0CY flag cyclically and the BTM0CY flag check time interval coincides with the BTM0CY flag setting time interval, a CE reset will not occur forever.

Note the following point:

Because one instruction cycle is 2μs (1/500 kHz), a program that checks the BTM0CY flag once at every 500 instructions reads the BTM0CY flag at every 1 ms (2μs × 500).

Under this condition, whichever timer interval set pulse, 5 ms or 100 ms, is selected, a CE reset will not occur for ever, once the setting and checking of the BTM0CY flag occur at the same time.

To be specific, avoid creating a cyclic program that satisfies the following condition.

$$\frac{t_{SET} \times 500}{x} = n \text{ (n is any integer)}$$

where t_{SET} : BTM0CY flag setting time interval

x : BTM0CY flag read instruction cycle time x number of steps

In other words, the program should not contain x steps when the above calculation produces any integer.

The program shown below is an example of a program that meets the above condition. Do not creates such a program.

Example

```
  Process A
  INITFLG   NOT BTM0ZX, NOT BTM0CK2, NOT BTM0CK1, BTM0CK0
                    ; Built-in macro
                    ; Specifies the timer carry FF set pulse as 5 ms.
LOOP:
; ①
  SKT1  BTM0CY ; Built-in macro
  BR    BBB
AAA:
  496 steps
  BR    LOOP
BBB:
  496 steps
  BR    LOOP
```

Because the BTM0CY flag read instruction at ① in this program is executed at every 500 instructions, once the BTM0CY flag happens to be set at the timing of the instruction at ①, a CE reset will not occur forever.

In addition, because the instruction execution time is 12μs (1/83.33 kHz) during the operation of the IDC, do not create a cyclic program that meets the following condition.

$$\frac{t_{SET} \times 83.33}{x} = n \text{ (n is any integer)}$$

where t_{SET} : BTM0CY flag setting time interval

x : BTM0CY flag read instruction cycle time x number of steps

12.5 TIMER INTERRUPT

The timer interrupt function issues an interrupt request at the negative-going edge of the timer interrupt pulse specified in the timer mode select register.

The timer interrupt request corresponds to the IRQBTM0 flag in the interrupt request register on a one-to-one basis. When an interrupt is requested, the corresponding IRQBTM0 flag is set to 1.

In other words, when a timer interrupt request pulse falls, the IRQBTM0 flag is set to 1.

As described in **Chapter 11**, to use the timer interrupt function, it is necessary not only to issue an interrupt request but also to execute the EI instruction, which enables all interrupts, and enable the timer interrupt.

The timer interrupt is enabled by setting the IPBTM0 flag to 1 in the interrupt permission register.

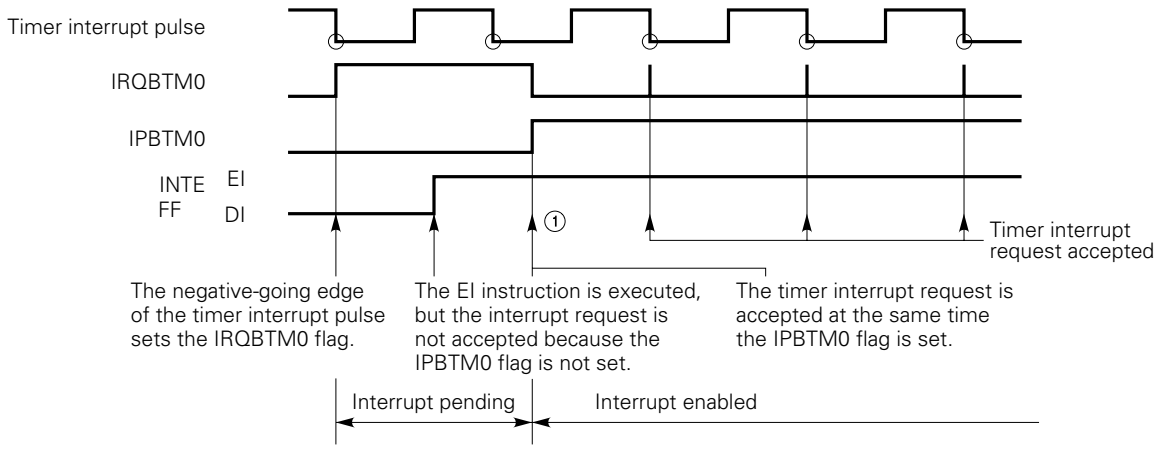
To put in another way, if the EI instruction has been executed, and the IPBTM0 flag is set to 1, an interrupt request is accepted when the IRQBTM0 flag is set to 1.

When a timer interrupt request is accepted, program control is passed to program memory address 0003H.

When the interrupt request is accepted, the IRQBTM0 flag is reset to 0.

Fig. 12-8 shows the relationship between the timer interrupt pulse and the IRQBTM0 flag.

Fig. 12-8 Relationship Between the Timer Interrupt Pulse and the IRQBTM0 Flag



At this point, note the following: Once the IRQBTM0 flag is set when a timer interrupt is disabled by the DI instruction or the IPBTM0 flag, the corresponding interrupt request is accepted immediately when the EI instruction is executed or the IPBTM0 flag is set.

In the above case, writing 0 to the IRQBTM0 flag can cancel the interrupt request.

Meanwhile, writing 1 to the IRQBTM0 flag amounts to issuing an interrupt request.

Accepting a timer interrupt request uses one level of stack.

When an interrupt request is accepted, the contents of the bank register and index enable flag are saved automatically.

A RETI instruction is used to return from an interrupt handling routine. This instruction is dedicated to use for this purpose.

See **Chapters 4** and **11** for details.

Sections 12.5.1 and **12.5.2** describe an example of using a timer interrupt and a timer interrupt error, respectively.

See **Chapter 11** for relationships with other types of interrupts (INT_{NC} pin, $\overline{V_{SYNC}}$ pin, and serial interface).

12.5.1 Example of Using a Timer Based on a Timer Interrupt

An example follows.

Example

```
        BR      AAA             ; Branches to AAA.
TIMER:                          ; Program address 0003H
        ADD     M1, #0001B      ; Add 1 to M1.
        SKT1    CY              ; Tests the CY flag.
        BR      BBB             ; Returns if no carry is generated.
        ┌──────────┐
        │ Process A │
        └──────────┘
BBB:
        EI
        RETI
AAA:
        INITFLG  NOT BTM0ZX, NOT BTM0CK2, NOT BTM0CK1, NOT BTM0CK0
                                ; Built-in macro
                                ; Specifies the timer interrupt pulse as 5 ms.
        MOV     M1, #0000B      ; Clears the content of M1 to 0.
        SET1    IPBTM0          ; Enables a timer interrupt.
        EI                      ; Enables all types of interrupts.
LOOP:
        ┌──────────┐
        │ Process B │
        └──────────┘
        BR      LOOP
```

This program performs process A at every 80 ms.

At this point, note the following: Accepting an interrupt request causes a DI state automatically, and the IRQBTM0 flag is set to 1 even in the DI state.

In other words, if process A takes 5 ms or longer, an interrupt request will be accepted immediately when a return is made by a RETI instruction, thus disabling process B from being performed.

12.5.2 Timer Interrupt Error

As explained in **Section 12.4**, an interrupt request is accepted each time the timer interrupt pulse goes low, provided that the interrupt is enabled.

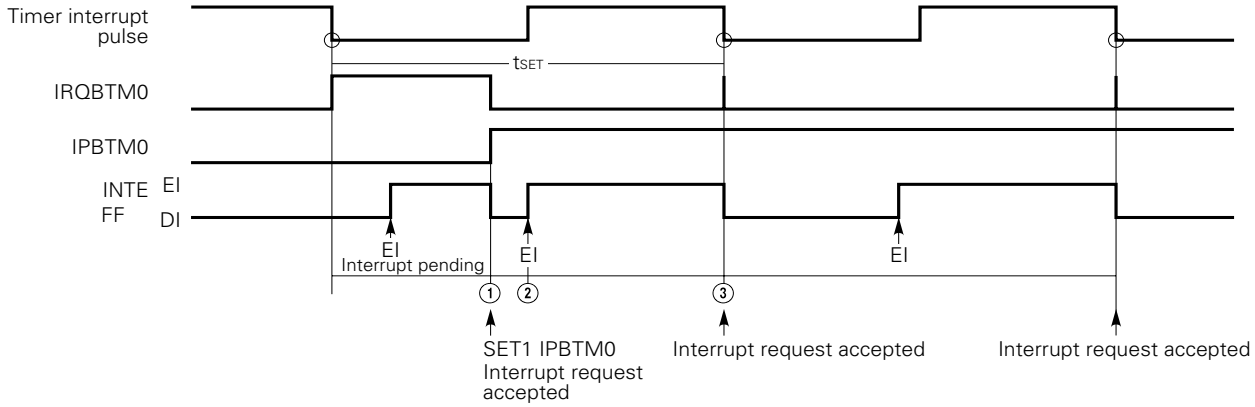
A timer error due to use of a timer interrupt occurs when:

(1) An interrupt request is accepted for the first time after the timer interrupt is enabled.
(2) An interrupt request is accepted for the first time after the timer interrupt pulse interval is switched.
(3) Writing to the IRQBTM0 flag occurs.

These timer error types are illustrated in Fig. 12-9.

Fig. 12-9 Timer Interrupt Error (1/2)

(a) When a timer interrupt is enabled



At point ①, an interrupt request is accepted immediately when the IPBTM0 flag is set to enable the timer interrupt.

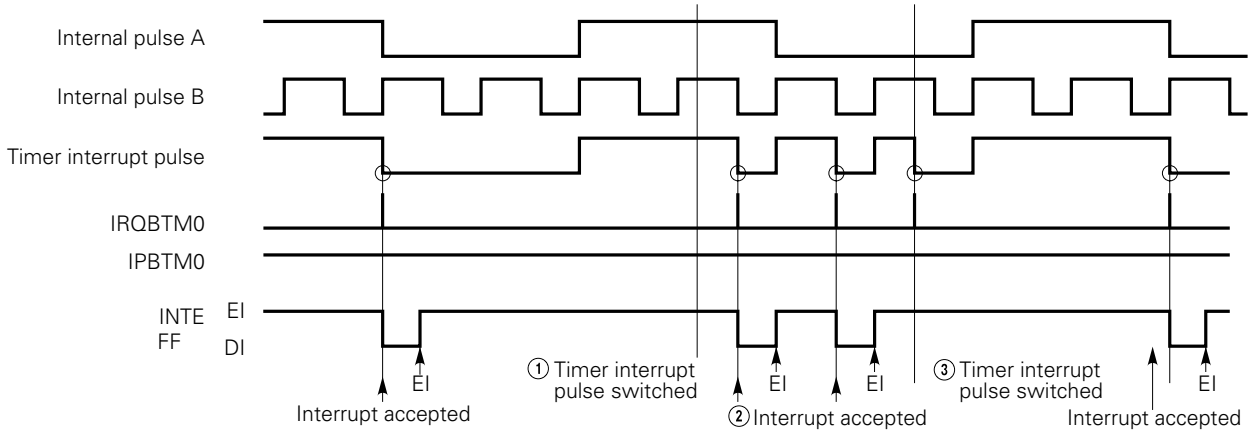
In the above case, timer error -tSET occurs.

If the EI instruction is executed at point ② to enable interrupts, an interrupt occurs at the negative-going edge of the timer interrupt pulse at point ③.

In the above case, the time error is: -tSET < timer error < 0

Fig. 12-9 Timer Interrupt Error (2/2)

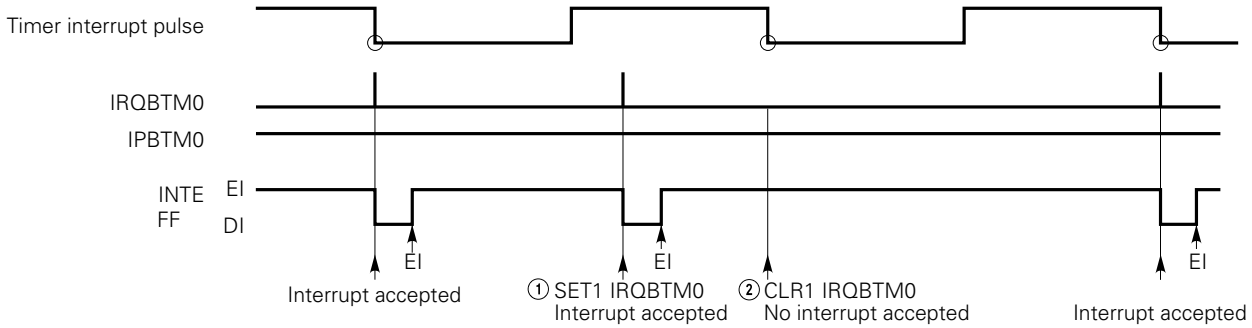
(b) When the timer interrupt pulse is switched



Although the timer interrupt pulse is switched to B at ①, no interrupt occurs because the timer interrupt pulse does not go low. Therefore, an interrupt occurs when the interrupt pulse goes low at point ②.

When the timer interrupt pulse is switched to A at point ③, the timer interrupt pulse goes low, and the interrupt request is accepted immediately.

(c) When the IRQBTM0 flag is manipulated



When the IRQBTM0 flag is set at point ①, an interrupt request is accepted immediately.
If the IRQBTM0 flag is reset at the same time the timer interrupt pulse goes low at point ②, the interrupt request is not accepted.

12.6 CAUTIONS IN USING THE TIMER INTERRUPT

In a program using a timer that operates at constant intervals once a power-on reset occurs, it is necessary to have the timer interrupt handling routine finish within that constant interval.

This is explained using an example.

Example

```
      BR      AAA          ; Branches to AAA after reset.
TIMER:                     ; Program address 0003H
      ADD     M1, #0100B   ; Adds 0100B to the content of M1.
      SKT1    CY           ; Performs clock processing if a carry occurs.
      BR      AAA
      ; ①
      Clock process
      EI
      RETI
AAA:
      INITFLG   NOT BTM0ZX, NOT BTM0CK2, NOT BTM0CK1, NOT BTM0CK0
                           ; Built-in macro
                           ; Specifies the timer interrupt time and timer carry FF setting time interval as 250
                             and 100 ms, respectively.
      SET1    IPBTM0       ; Built-in macro
      EI                   ; Enables the timer interrupt.
      Process A
      BR      AAA
```

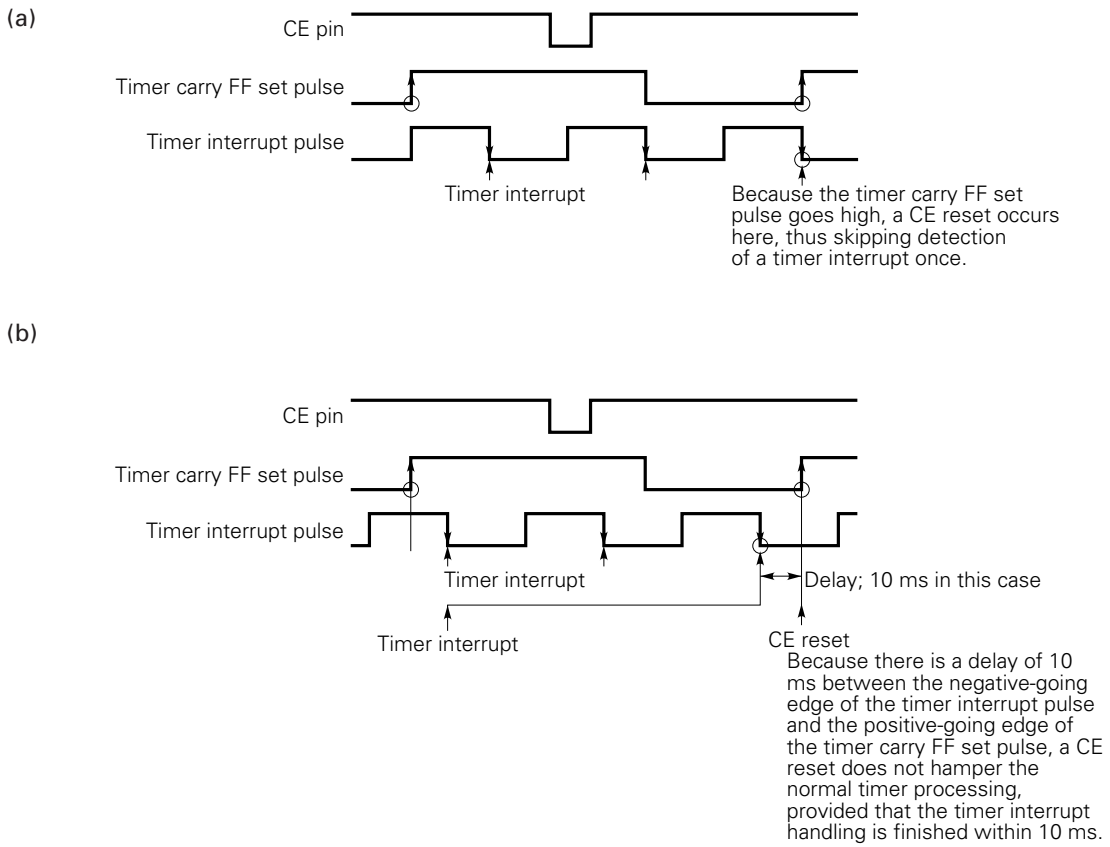
The program in this example performs the clock process ① at every one second while performing process A.

If the CE pin goes from a low to a high as shown in Fig. 12-10 (a), a CE reset occurs in synchronization with the positive-going edge of the timer carry FF set pulse. If the timer interrupt request happens to be issued simultaneously when the timer carry FF is set, a CE reset takes precedence. When the CE reset occurs, it resets the timer interrupt request (IRQBTM0 flag), thus skipping the timer process once.

In reality, however, to avoid skipping the timer process in the above example, a delay is provided between the negative-going edge of the timer carry FF set pulse and the negative-going edge of the timer interrupt pulse, as shown in Fig. 12-10 (b).

As shown at (2) in Fig. 12-10, restricting the clock process to within 10 ms can eliminate skipping of a timer interrupt that would otherwise be caused by a CE reset.

Fig. 12-10 Timing Chart



13. STANDBY

The standby function is intended to reduce the current drain of the device at backup.

13.1 STANDBY BLOCK CONFIGURATION

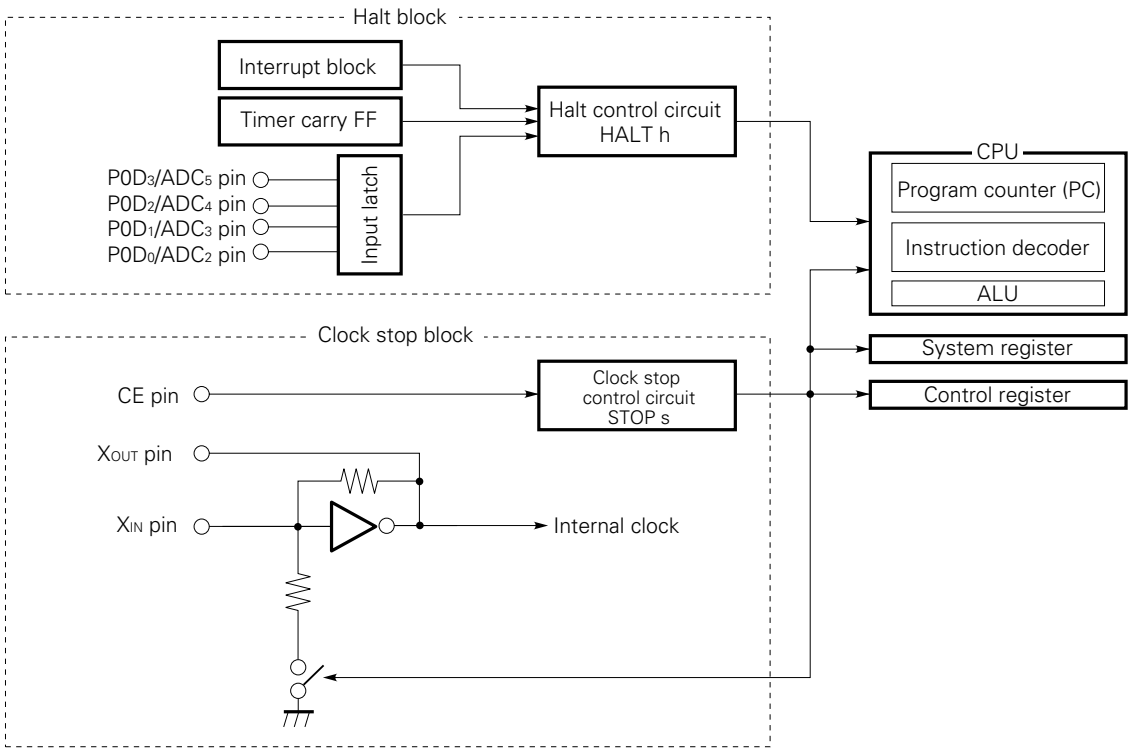
Fig. 13-1 shows the configuration of the standby block.

As shown in Fig. 13-1, the standby block is further divided into halt control and clock stop control blocks.

The halt control block consists of the halt control circuit, interrupt control block, timer carry FF, and the $P0D_0$/ADC_2 to $P0D_3$/ADC_5 pins. It controls the operation of the CPU (program counter, instruction decoder, and ALU block).

The clock stop control block controls the 8 MHz crystal oscillator, CPU, system register, and control register.

Fig. 13-1 Standby Block Configuration



13.2 STANDBY FUNCTION

The standby function stops the whole or part of the operation of the device to reduce its current drain.

The standby function is divided into halt and clock stop functions.

The halt function uses a dedicated instruction (HALT h instruction) to stop the CPU in order to reduce the required current drain.

The clock stop function uses a dedicated instruction (STOP s instruction) to stop the 8 MHz crystal oscillator in order to reduce the current drain in the device.

To use these functions, it is necessary to specify a device operation mode at the CE pin.

Section 13.3 explains the device operation mode specified at the CE pin.

Sections 13.4 and **13.5** describe the halt and clock stop functions.

Remark For the μPD17062, the operand s of the STOP s instruction must be 0000B. Therefore, the actual instruction is: STOP 0000B

13.3 DEVICE OPERATION MODE SPECIFIED AT THE CE PIN

The CE pin controls the following items according to the level and positive-going edge of its input signal.

(1) Whether to enable or disable the clock stop instruction
(2) Whether to reset the device

Sections 13.3.1 and **13.3.2** explain the above items, respectively.

13.3.1 Controlling Whether to Enable or Disable the Clock Stop Instruction

The clock stop instruction, STOP s, is effective only when the CE pin is at a low level.

If the STOP s instruction is executed when the CE pin is at a high level, it is treated as a no-operation (NOP) instruction.

13.3.2 Resetting the Device

Driving the CE pin from a low to a high can reset the device (CE reset).

There is another type of reset, which is a power-on reset. It occurs when supply voltage V_{DD} is turned on.

See **Chapter 14** for details.

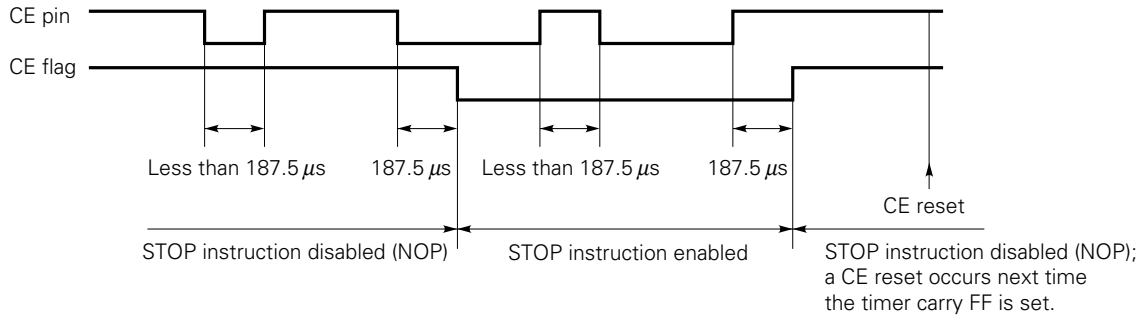
13.3.3 Signal Inputs to the CE Pin

The CE pin does not accept a high or low level with a duration of less than 187.5 μs to prevent malfunction due to noise.

The input level of a signal supplied to the CE pin is checked using the CE flag in the control register (bit b_0 at address 07H).

Fig. 13-2 shows the relationship between the input signal and CE flag.

Fig. 13-2 Relationship Between the Input Signal and CE Flag



13.4 HALT FUNCTION

The halt function stops the operation of the CPU clock by executing the HALT h instruction.

When the HALT h instruction is executed, the program stops at this instruction and rests there until the halt state is released.

In the halt state, the current drain in the device is reduced by the amount required by the CPU to operate.

The halt state can be released using the timer carry FF, interrupt, and key entry.

The operand h of the HALT h instruction specifies a condition (timer carry FF, interrupt, or key entry) to release the halt state.

The HALT h instruction is always effective regardless of the input level at the CE pin.

Sections 13.4.1 to **13.4.5** explain the halt state and halt release conditions.

13.4.1 Halt State

The CPU is entirely at a stop in the halt state.

In the halt state, the program completely stops at the HALT h instruction.

In the halt state, however, the peripheral hardware continues operating as it did before execution of the HALT h instruction.

13.4.2 Halt Release Conditions

Fig. 13-3 summarizes the release conditions.

As shown in Fig. 13-3, the halt release condition is 4-bit data specified in the operand h of the HALT h instruction.

The halt state is released when a condition specified as 1 in the operand h is satisfied.

Upon release of the halt state, the subsequent instructions after the HALT h instruction are executed sequentially.

If multiple release conditions are specified at a time, the halt state is released when only one of them is satisfied.

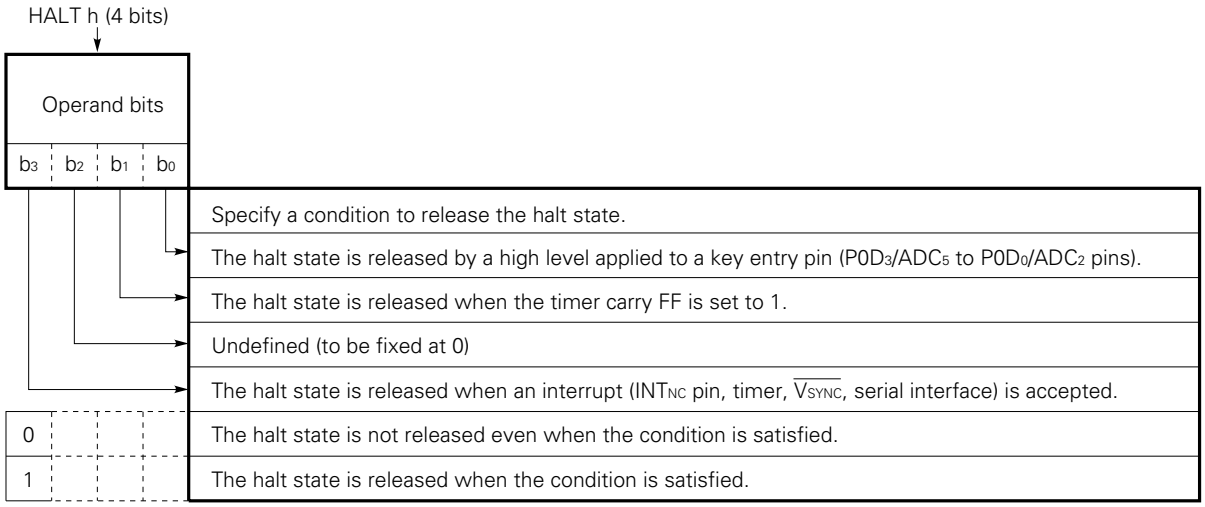
Also when a power-on or CE reset occurs, the halt state is released and the device is reset.

If the operand h is 0000B, no halt release condition is specified.

Under this condition, the halt state is released by resetting (power-on or CE reset) the device.

Sections 13.4.3 to **13.4.6** explain the timer carry FF, interrupt, and key entry as halt release conditions.

Fig. 13-3 Halt Release Conditions



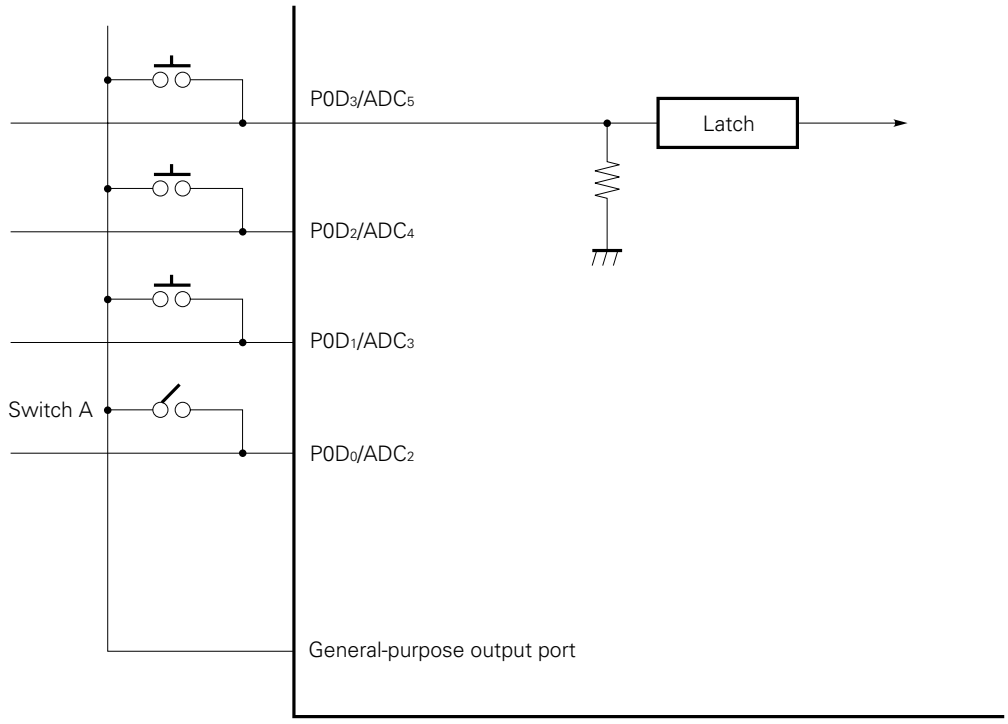
13.4.3 Halt Release by Key Entry

The HALT 0001B instruction specifies a key entry as a halt release condition.

If this condition is specified, the halt state is released when a high level is applied to one of the $P0D_0/ADC_2$ to $P0D_3/ADC_5$ pins.

Items (1) to (3) describe cautions to be taken in using a general-purpose output port as a key source signal and the $P0D_0/ADC_2$ to $P0D_3/ADC_5$ pins for an A/D converter.

(1) Cautions in using a general-purpose output port as a key source signal



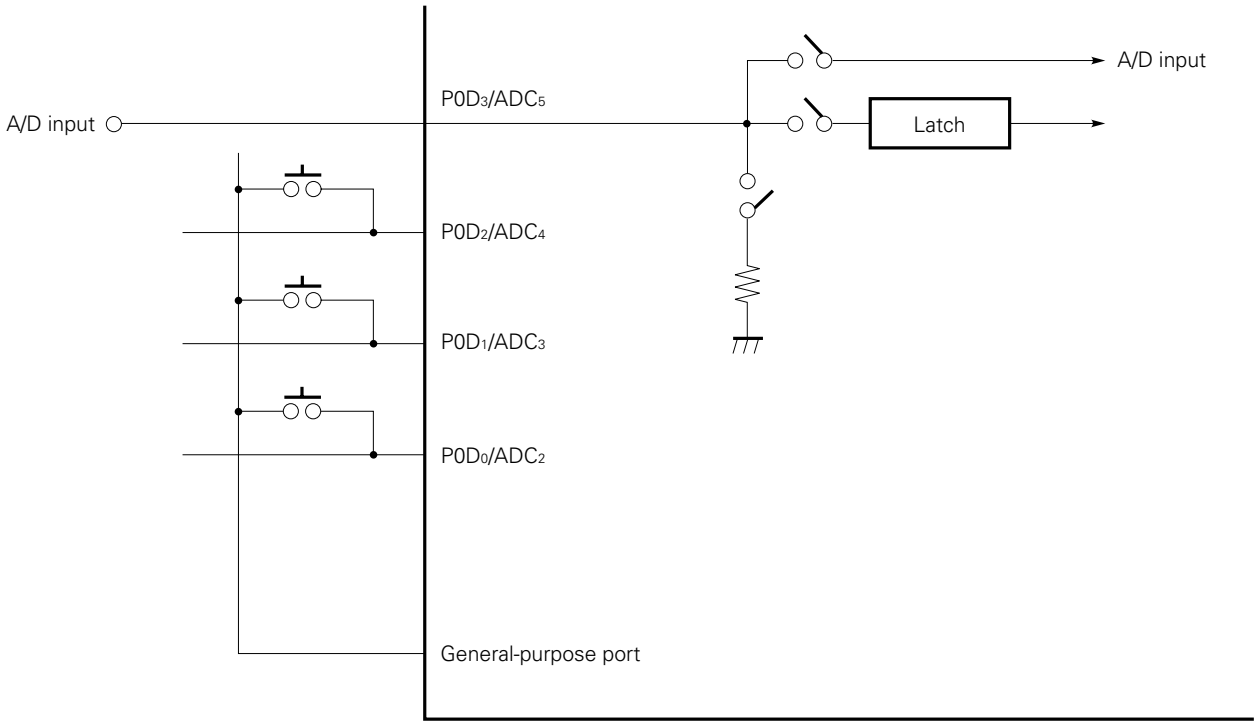
The HALT 0001B instruction must be executed after the general-purpose output port for key source signal input is raised to a high.

If an alternate switch, like switch A in the above figure, is used, a high level is always applied to the $P0D_0/ADC_2$ pin when the switch is kept closed, and causes the halt state to be released immediately.

Therefore, great care should be taken when an alternate switch is used.

The $P0D_0/ADC_2$ to $P0D_3/ADC_5$ pins are internally pulled down automatically.

(2) Cautions in using the P0D0/ADC2 to P0D3/ADC5 pins for an A/D converter



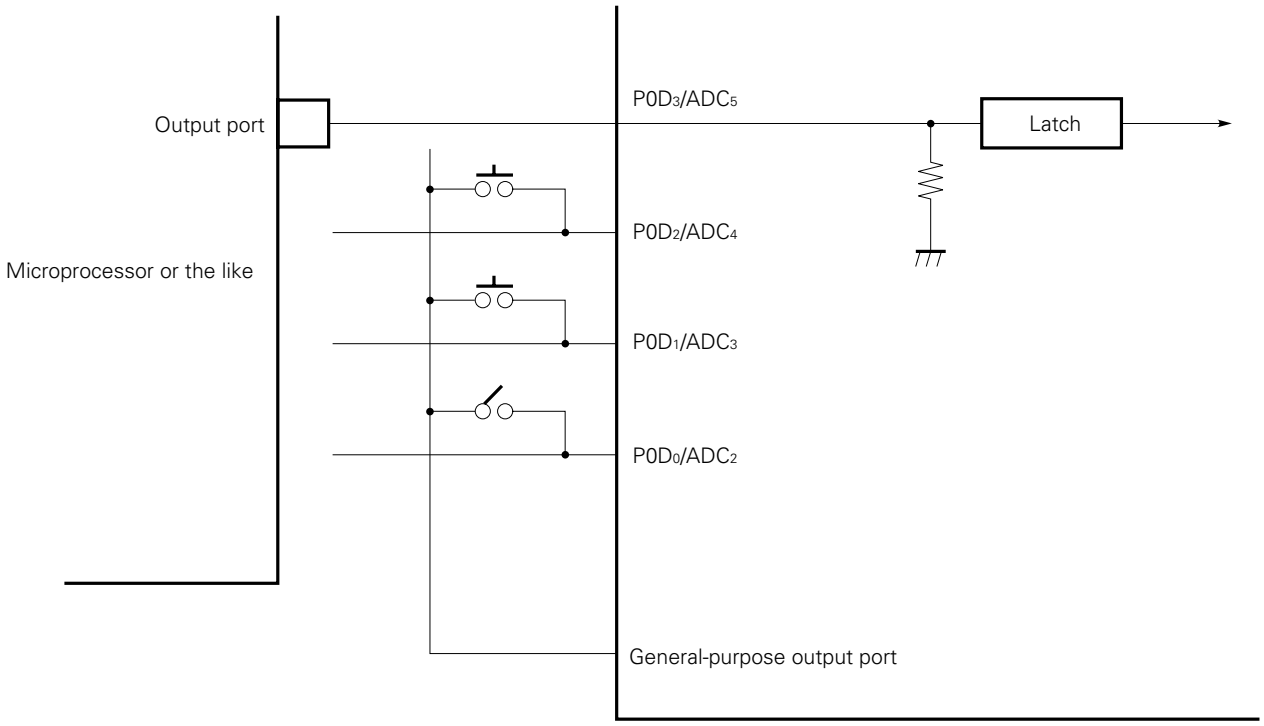
If one of the P0D0/ADC2 to P0D3/ADC5 pins is selected for an A/D converter (only one pin can be selected at one time), it is disconnected from the input latch and connected to the internal A/D converter input.

If a pin happens to be at a high level when it is selected for an A/D converter, the latch circuit is held at a high.

If the HALT 0001B instruction is executed under the above condition, the halt state is released immediately because the input latch is at a high.

To solve the above problem, specify the input port so that a low level is input to the A/D converter, before executing the HALT 0001B instruction.

(3) Alternative method to release the halt state



The P0D0/ADC2 to P0D3/ADC5 pins can be used a general-purpose input port with a built-in pull-down resistor.

This configuration of the P0D0/ADC2 to P0D3/ADC5 pins enables a microprocessor to be used to release the halt state as shown above.

13.4.4 Releasing the Halt State by the Timer Carry FF

The HALT 0010B instruction specifies the timer carry FF as a halt release condition.

If it is specified that the halt state is to be released according to the timer carry FF, the halt state is released immediately when the timer carry FF is set to 1.

The timer carry FF corresponds to the BTM0CY flag (bit b_0 at address 17H) in the control register on a one-to-one basis, and is set to 1 at constant intervals (5 or 100 ms).

Use of the timer carry FF can therefore release the halt state at constant intervals.

An example of using the HALT 0010B instruction follows:

Example

```
        HLTTMR    DAT 0010B       ; Defines a symbol.
        INITFLG   NOT BTM0ZX, NOT BTM0CK2, NOT BTM0CK1, NOT BTM0CK0
                                  ; Built-in macro
                                  ; Specifies the timer carry FF setting time interval as 100 ms.
LOOP1:
        MOV       M1, #0110B
LOOP2:
        HALT      HLTTMR          ; Specifies the timer carry FF as a halt release condition.
        SKT1      BTM0CY          ; Built-in macro
        BR        LOOP            ; Branches to LOOP2 if the BTM0CY flag is not set.
        ADD       M1, #0001B      ; Adds 0001B to the contents of M1.
        SKT1      CY              ; Built-in macro
        BR        LOOP2           ; Performs process A if there is a carry.
        ┌──────────┐
        │ Process A │
        └──────────┘
        BR        LOOP1
```

This sample program releases the halt state at intervals of 100 ms and perform process A at intervals of 1 s.

13.4.5 Releasing the Halt State by an Interrupt

The HALT 1000B instruction specifies an interrupt as halt release condition.

If it is specified that the halt state is to be released according to an interrupt, the halt state is released immediately when an interrupt request is accepted.

Four interrupt sources, INT_{NC} pin, timer, $\overline{V_{SYNC}}$, and serial interface, can be used as a condition to release the halt state.

It is necessary to program which interrupt source is to be used as a halt release condition, beforehand.

For an interrupt request to be accepted, besides issuing the interrupt request, it is necessary to enable all interrupts (EI instruction) and the interrupt that corresponds to the issued interrupt request (to set the interrupt permission flag).

If interrupts are not enabled, no interrupt request is accepted and therefore the halt state is not released, even if an interrupt request is issued.

If an interrupt request is accepted and the halt state is released, program control is passed to the vector address of the corresponding interrupt. After the required interrupt handling is finished, when the RET1 instruction is executed, program control is returned to the instruction just after the HALT instruction.

This is explained in the following example.

Example

```
    HLTINT      DAT 1000B       ; Defines a symbol.
START:                          ; Address 0000H
    BR          MAIN            ;
    NOP
INTTM:                          ; Timer interrupt vector address (0003H)
    BR          INTTIMER        ; Branches to INTTIMER (interrupt handling).
INT0:                           ; INTNC pin interrupt vector address (0004H)
    Process A                   ; Interrupt requested at the INTNC pin
    EI
    RETI
INTTIMER:
    Process B                   ; Timer interrupt handling
    EI
    RETI
MAIN:
    SET2        IPBTM0, IPNC    ; Built-in macro
                                ; Enables INTNC pin and timer interrupts.
    SET1        BTM0CK2         ; Built-in macro
LOOP:                           ; Specifies the timer interrupt time interval as 5 ms.
    Process C                   ; Main routine processing
    EI                          ; Enables all interrupts.
    HALT        HLTINT          ; Specifies an interrupt as a halt release condition.
    ; ①
    BR          LOOP
```

This sample program releases the halt state and performs process B when a timer interrupt request is accepted. When an interrupt request at the INTNC pin is issued, the program performs process A. It also performs process C each time the halt state is released.

If an INTNC pin interrupt is requested exactly at the same time with a timer interrupt during the halt state, the program performs process A for the INTNC pin interrupt, which has a higher hardware priority than the timer interrupt. When a RETI instruction is executed upon completion of process A, program control is returned to the BR LOOP instruction at ①, but this instruction will not be executed. Instead, the timer interrupt request is accepted immediately. The BR LOOP instruction is executed only after a RETI is executed upon completion of process B (timer interrupt handling).

13.5 CLOCK STOP FUNCTION

The clock stop function stops the operation of the 8 MHz crystal oscillator by executing the STOP s instruction.

The clock stop function can reduce the current drain of the μPD17062 by 10 μA (maximum).

The operand s of the STOP s instruction is 0000B.

This instruction is effective only when the CE pin is at a low level. If executed when the CE pin is at a high, the STOP s instruction is regarded as a no-operation instruction (NOP).

In other words, the STOP s instruction should be executed when the CE pin is at a low.

A CE reset is used to release the clock stop state.

Sections 13.5.1 to **13.5.3** describe the clock stop state, how to release the clock stop state, and cautions to be taken in using the clock stop instruction.

13.5.1 Clock Stop State

In the clock stop state, all operations of the device, including CPU and peripheral hardware operations, are stopped, because the crystal oscillator stops.

During the clock stop state, the power-failure detector does not operate even if the supply voltage V_{DD} is lowered to about 2.2 V. This makes possible a low-voltage data memory backup.

13.5.2 Releasing the Clock Stop State

The clock stop state is released by raising the level of the CE pin from a low to a high (CE reset) or by lowering the supply voltage V_{DD} of the device below 2.2 V, then increasing it to 4.5 V (power-on reset).

Figs. 13-4 and 13-5 show how the clock stop state is released by a CE reset and power-on reset, respectively.

Releasing the clock stop state using a power-on reset causes the power-failure detector to start operating.

Fig. 13-4 Releasing the Clock Stop State by a CE Reset

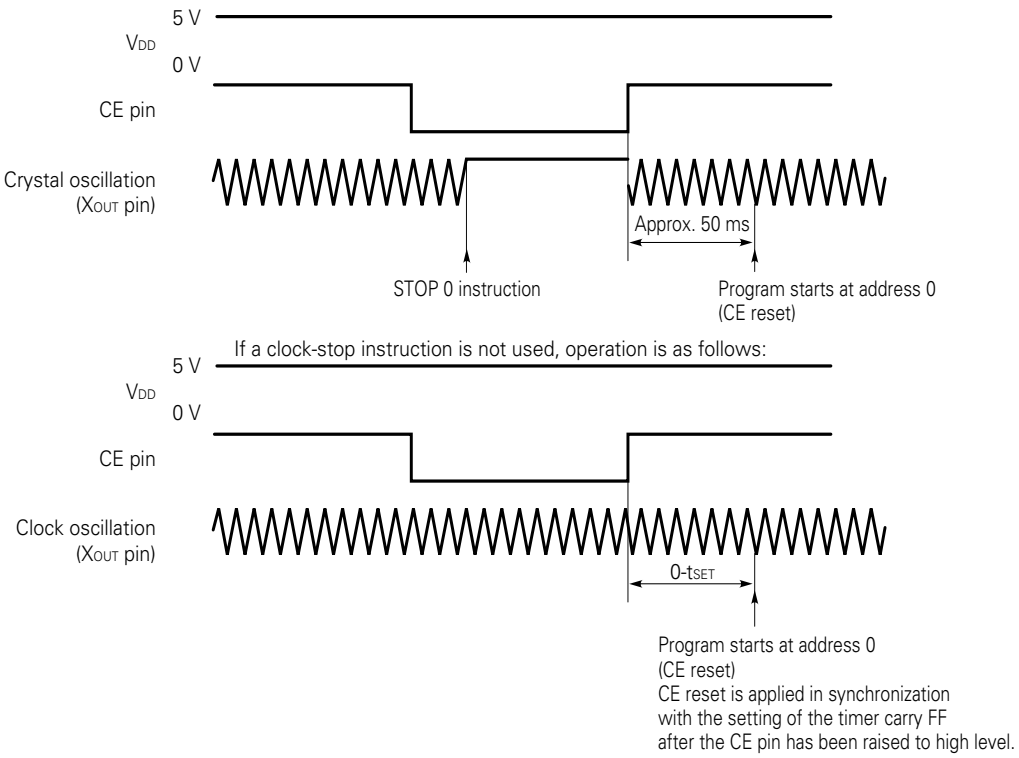
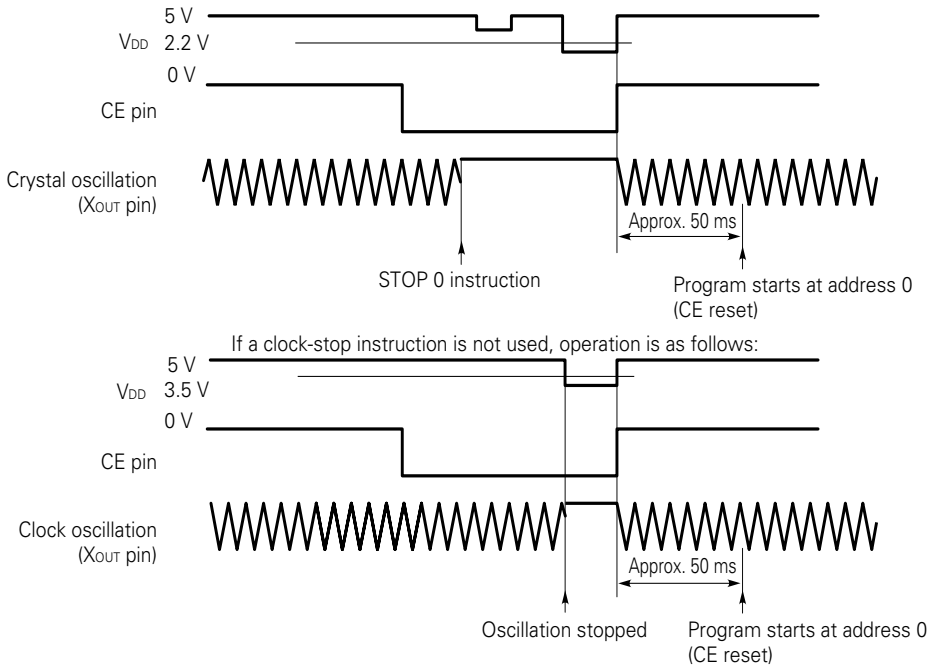


Fig. 13-5 Releasing the Clock Stop State by a Power-on Reset



13.5.3 Cautions in Using the Clock Stop Instruction

The clock stop instruction (STOP s) is effective only when the CE pin is at a low level.

To enable the clock stop state to be released, the program must therefore have a provision to handle when the CE pin happens to be at a high.

Such a provision is explained using the example below.

Example

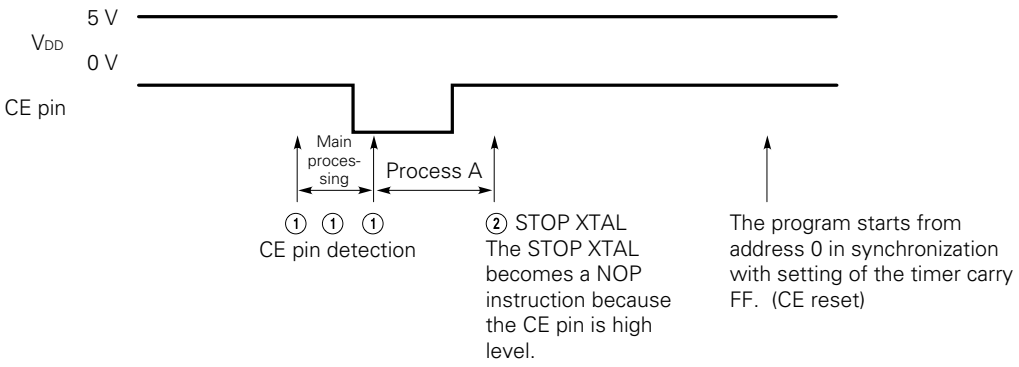
```
    XTAL        DAT 0000B     ; Defines a symbol for the clock stop condition.
CEJDG:
    ; ①
    SKF1        CE            ; Built-in macro
                              ; Checks the input level at the CE pin.
    BR          MAIN          ; Branches to the main process if the CE pin is high.
    ┌──────────┐
    │ Process A │             ; Processing performed when the CE pin is low
    └──────────┘
    ; ②
    STOP        XTAL          ; Stops the clock.
    ; ③
    BR          $ - 1
MAIN:
    ┌──────────┐
    │ Main process │
    └──────────┘
    BR CEJDG
```

The above program checks the CE pin at ①. If the CE pin is at a low, the clock stop instruction (STOP XTAL) at ② is executed after process A is finished.

If the CE pin goes high during execution of the STOP XTAL instruction at ② as shown below, the STOP XTAL instruction is treated as a no-operation (NOP). If the program does not contain the branch instruction (BR $ - 1) at ③, program control is passed to the main process, possibly resulting in a malfunction.

The program must always have a branch instruction at ③ or have a provision that can prevent a malfunction in the main process.

Even if the CE pin remains high, the branch instruction at ② allows a CE reset to occur next time the timer carry FF is set.



13.6 OPERATION OF THE DEVICE AT A HALT OR CLOCK STOP

13.6.1 State of Each Pin at a Halt and Clock Stop

Table 13-1 summarizes how the CPU and peripheral hardware behave during the halt or clock stop state.

During the halt state, execution of the CPU instructions is suspended, but the peripheral hardware operates normally, as described in Table 13-1.

During the clock stop state, on the other hand, all peripheral hardware is at a stop.

During the halt state, the control register that controls the operating state of the peripheral hardware works as usual (not initialized). During the clock stop state (when the STOP s instruction is executed), on the other hand, the control register is initialized to a specified value.

To put in another way, the peripheral hardware keeps operating as specified in the control register during the halt state. During the clock stop state, however, the peripheral hardware operates according to the initial value set in the control register.

See **Chapter 9** for the initial value for the control register.

Let's study the following example.

Example When the $P0A_0$/SDA and $P0A_1$/SCL pins of port 0A are specified as output ports, and the $P0A_2$/$\overline{SCK}$ and $P0A_3$/SO pins are used as a serial interface

```
HLTINT      DAT 1000B       ; Defines a symbol.
XTAL        DAT 0000B       ;
INITFLG     P0ABIO3, P0ABIO2, P0ABIO1, P0ABIO0
                            ; Built-in macro
; ①
SET2        P0A0, P0A1      ;
INITFLG     SIO0CH, NOT SB, SIO0MS, SIO0TX
                            ;
SET2        SIO0CK1, SIO0CK0
; ②
SET2        SIO0IMD1, SIO0IMD0
CLR1        IRQSIO0
SET1        IPSIO0
EI
; ③
SET1        SIO0NWT
; ④
HALT        HLTINT
; ⑤
STOP        XTAL
```

The above program outputs a high level from the $P0A_0$ and $P0A_1$ pins at ①, specifies a serial interface condition at ②, and starts serial communication at ③.

When the HALT instruction is executed at ④, the serial communication continues, and the halt state is released after a serial interface interrupt request is accepted.

If the STOP instruction at ⑤ is executed in place of the HALT instruction at ④, all flags in the control register set up at ①, ②, and ③ are initialized, and therefore, the serial communication is suspended and all pins of port 0A are specified as general-purpose input/output ports.

Table 13-1 Behavior of the Device at the Halt and Clock Stop States

Peripheral hardware	State			
	CE pin = high level		CE pin = low level	
	Halt	Clock stop	Halt	Clock stop
Program counter	Stops at the address of the HALT instruction.	The STOP instruction is invalid (NOP).	Stops at the address of the HALT instruction.	Initialized to 0000H and stops.
System register	Holds the previous state.		Holds the previous state.	InitializedNote.
Peripheral hardware register	Holds the previous state.		Holds the previous state.	Holds the previous state.
Control register	Holds the previous state.		Holds the previous state.	InitializedNote.
Timer	Operates normally.		Operates normally.	Stops operating.
A/D converter	Operates normally.		Operates normally.	Stops operating.
D/A converter	Operates normally.		Operates normally.	Stops operating.
Serial interface	Stops operating.		Stops operating.	Stops operating.
General-purpose input/output port	Operates normally.		Operates normally.	Works as input port.
General-purpose input port	Operates normally.		Operates normally.	Works as input port.
General-purpose output port	Operates normally.		Operates normally.	Holds the previous state.
IDC	Holds the same state as when the HALT instruction is executed.		Stops operating.	Stops operating.

Note See **Chapters 8** and **9** for the initial values.

13.6.2 Cautions in Processing of Each Pin During Halt or Clock Stop State

The halt function is intended to reduce the required current drain, for example, by allowing only the clock to operate. Meanwhile, the clock stop function is intended to reduce the required current drain by suspending all operations except preservation of data in memory. During the halt or clock stop state, therefore, it is necessary to reduce the required current drain as much as possible. Because the current drain varies with the state of each pin, it is necessary to take cautions listed in Table 13-2.

Table 13-2 State of Each Pin During the Halt or Clock Stop State and Cautions to Be Taken (1/2)

<table>
<tr><th colspan="2" rowspan="2">Pin function</th><th rowspan="2">Pin symbol</th><th colspan="2">State of each pin and cautions in processing</th></tr>
<tr><th>Halt state</th><th>Clock stop state</th></tr>
<tr><td rowspan="4">General-purpose input/output port</td><td>Port0A</td><td>P0A3/SO
P0A2/$\overline{SCK}$
P0A1/SCL
P0A0/SDA</td><td rowspan="5">The state that exists before the execution of the halt instruction continues.

(1) When the port is specified as output
If the pin pulled down externally during high-level output or pulled up externally during low-level output, the current drain increases.
Be careful especially for N-channel open-drain outputs (P0A1, P0A0, P1A3 to P1A0).

(2) When the port is specified as input
When the pin is floating, the current drain increases due to noise.

(3) Port 0D (P0D3/ADC7 to P0D0/ADC4)
Because the pin is already pulled down internally, the current drain will increase if it is pulled up externally. If the pin is selected for A/D converter, however, the internal pull-down resistor is disconnected.</td><td rowspan="5">These pins are specified as general-purpose input ports. All input ports except the P0A1/SCL and P0A0/SDA pins are designed so that even if they are floating externally, the current drain will not increase due to noise. For the P0A1/SCL and P0A0/SDA pins, an external pull-down resistor or pull-up resistor must be connected to keep the current drain from increasing.
Port 0D (P0D3/ADC5 to P0D0/ADC2) are pulled down internally.</td></tr>
<tr><td>Port0B</td><td>P0B3/HSCNT
P0B2/TMIN
P0B1
P0B0/SI</td></tr>
<tr><td>Port1B</td><td>P1B3
P1B2
P1B1
P1B0</td></tr>
<tr><td>Port1C</td><td>P1C3/ADC1
P1C2
P1C1</td></tr>
<tr><td>General-purpose input port</td><td>Port0D</td><td>P0D3/ADC5
P0D2/ADC4
P0D1/ADC3
P0D0/ADC2</td></tr>
<tr><td>General-purpose output port</td><td>Port0C</td><td>P0C3
P0C2
P0C1
P0C0</td><td>(4) Port0B (P0B3/HSCNT to P0B0/SI) and port1B (P1B3/P1B0)
When the P0B3/HSCNT pin operates as the HSYNC counter or when the P1B3 pin operates as an external timer input, the built-in self-bias circuit operates, resulting in an increase in the current drain.</td><td>These pins are specified as general-purpose ports.
The outputs are preserved. Therefore, if they are pulled down externally during high-level output or pulled up during low-level output, the current drain will increase.</td></tr>
</table>

Table 13-2 State of Each Pin During the Halt or Clock Stop State and Cautions to Be Taken (2/2)

Pin function	Pin symbol	State of each pin and cautions in processing	
		Halt state	Clock stop state
Interrupt	INT_{NC}	If the pin is floating, external noise causes the current drain to increase.	
IDC	RED GREEN BLUE BLANK $\overline{H_{SYNC}}$ $\overline{V_{SYNC}}$	The output pins remain in the state in which they were when the HALT instruction was executed. If the IDCEN flag is set, the current drain increases.	The IDC is disabled. Each pin behaves as follows: The current drain will not increase, even if the RED, GREEN, and BLUE pins output a low level, or the $\overline{H_{SYNC}}$ and $\overline{V_{SYNC}}$ pins are floating.
D/A converter	PWM_3 PWM_2 PWM_1 PWM_0	It is necessary to take the same cautions as for the general-purpose output port.	All pins output a low level.
A/D converter	ADC_0	The pin becomes floating.	
Clock oscillator	X_{IN} X_{OUT}	The current drain varies with the waveform of the oscillation output of the clock oscillator. The larger the amplitude, the current drain becomes lower. The oscillation amplitude of the oscillator varies depending on its crystal and load capacitance; evaluation is required.	The X_{IN} pin is internally pulled down, and the X_{OUT} pin outputs a high level.

14. RESET

The reset function is used to initialize device operation.

14.1 RESET BLOCK CONFIGURATION

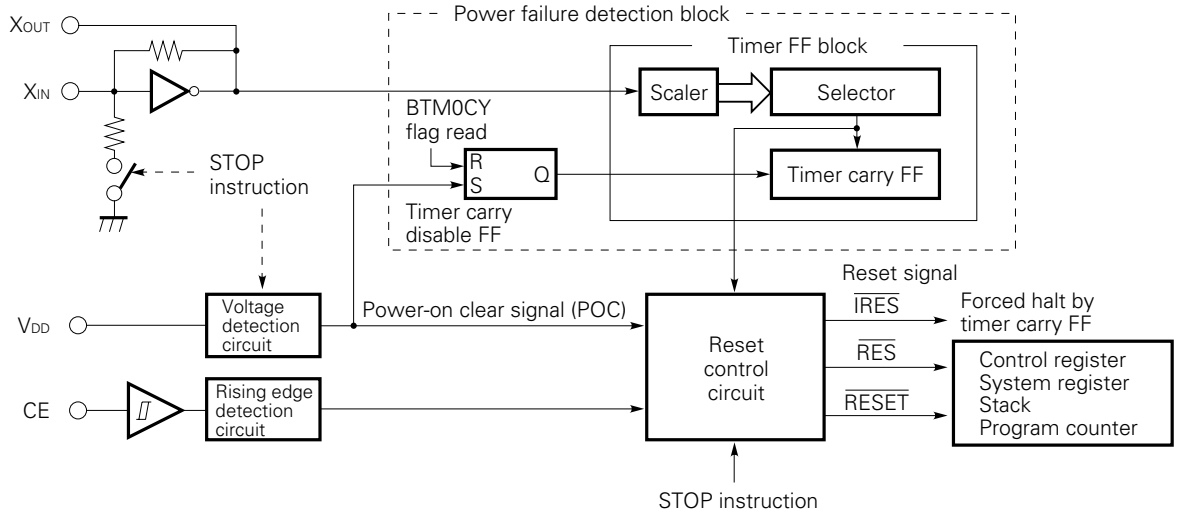
Fig. 14-1 shows the configuration of the reset block.

Device reset is divided into reset by turning on V_{DD} (power-on reset or V_{DD} reset), and reset by CE pin (CE reset).

The power-on reset block consists of a voltage detection circuit that detects the voltage applied to the V_{DD} pin, a power failure detection circuit, and a reset control circuit.

The CE reset block consists of a circuit that detects the rising edge of the signal input to the CE pin, and a reset control circuit.

Fig. 14-1 Reset Block



14.2 RESET FUNCTION

Power-on reset is applied when V_{DD} rises from a certain voltage, CE reset is applied when the CE pin rises from low level to high level.

Power-on reset initializes the program counter, stack, system register and control registers, and executes the program from address 0000H.

CE reset initializes the program counter, stack, system register and some control registers, and executes the program from address 0000H.

The main differences between power-on reset and CE reset are the operation of the control registers that are initialized and the power failure detection circuit described in **Section 14.6**.

Power-on reset and CE reset are controlled by reset signals $\overline{IRES}$, $\overline{RES}$, and $\overline{RESET}$ output from the reset control circuit in Fig. 14-1.

Table 14-1 shows the $\overline{IRES}$, $\overline{RES}$, and $\overline{RESET}$ signal and power-on reset and CE reset relationship.

The reset control circuit also operates when the clock-stop instruction (STOP) described in **Chapter 13** is executed.

Sections 14.3 and **14.4** describe CE reset and power-on reset, respectively.

Section 14.5 describes the relationship between CE reset and power-on reset.

Table 14-1 Relationship between Internal Reset Signal and Each Reset

Internal reset signal	Output signal			Contents controlled by each reset signal
	At CE reset	At power-on reset	At clock-stop	
$\overline{IRES}$	×	○	○	Forces the device into the halt state. The halt state is released by the setting of the timer carry FF.
$\overline{RES}$	×	○	○	Initializes some control registers.
$\overline{RESET}$	○	○	○	Initializes the program counter, stack, system register, and some control registers.

14.3 CE RESET

CE reset is executed by raising the CE pin from low level to high level.

When the CE pin rises to high level, the $\overline{\text{RESET}}$ signal is output and the device is reset in synchronization with the rising edge of the pulse used for the next setting of the timer carry FF.

When CE reset is applied, the $\overline{\text{RESET}}$ signal initializes the program counter, stack, system register, and some control registers to their initial value and executes the program from address 0000H.

For the initial values, see the relevant item.

CE reset operation is different when clock-stop is used and when it is not used.

These operations are described in **Sections 14.3.1** and **14.3.2**, respectively.

Section 14.3.3 describes the cautions at CE reset.

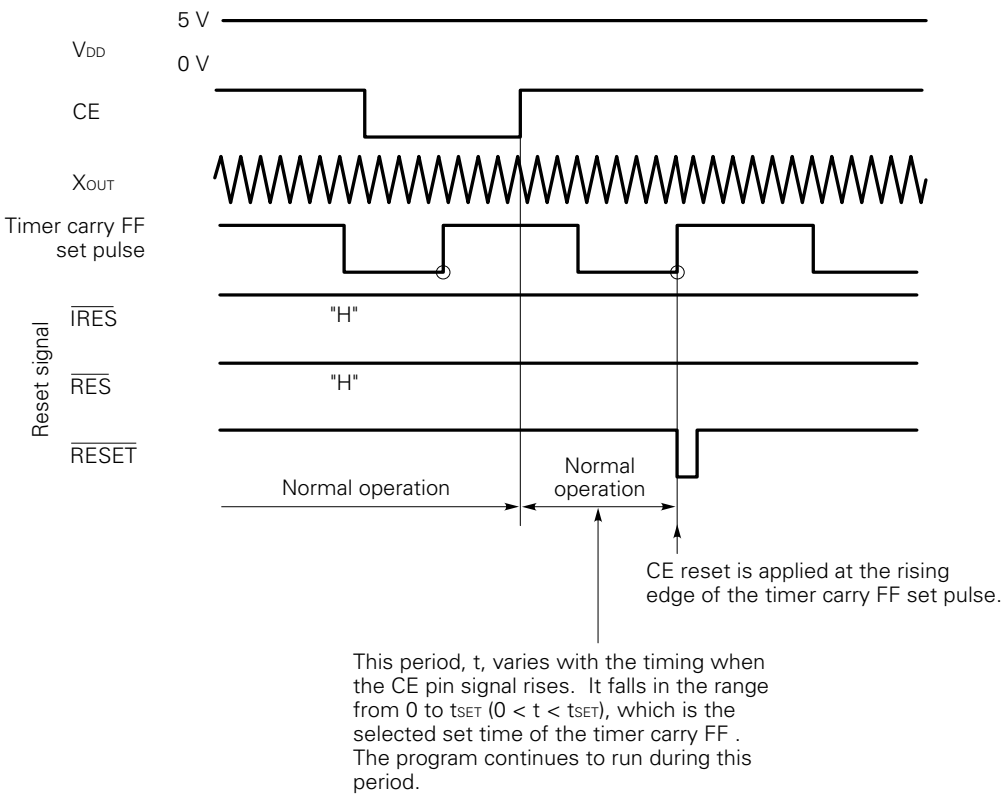
14.3.1 CE Reset When Clock-Stop (STOP Instruction) Not Used

Fig. 14-2 shows the reset operation.

When clock-stop (STOP instruction) is not used, the timer mode selection register of the control registers is not initialized.

Therefore, after the CE pin becomes high level, the $\overline{\text{RESET}}$ signal is output, and reset is applied at the rising edge of the timer carry FF set pulse (5 or 100 ms) .

Fig. 14-2 CE Reset Operation When Clock-Stop Not Used



14.3.2 CE Reset When Clock-Stop (STOP Instruction) Used

Fig. 14-3 shows the reset operation.

When clock-stop is used, the $\overline{\text{IRES}}$, $\overline{\text{RES}}$ and $\overline{\text{RESET}}$ signals are output at the time the STOP instruction is executed.

At this time, the $\overline{\text{RES}}$ signal initializes the timer mode selection register of the control registers to 0000B and sets the timer carry FF set signal to 100 ms.

Since the $\overline{\text{IRES}}$ signal is output continuously while the CE pin is low level, release by timer carry FF is forcibly halted.

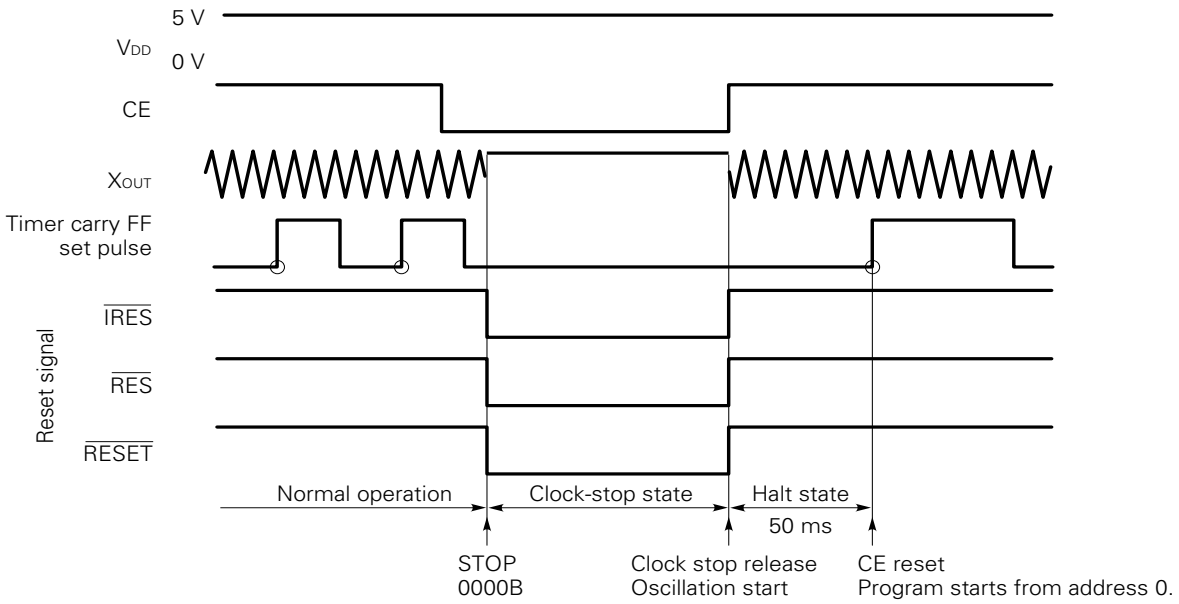
Since the clock itself stops, the device stops operating.

When the CE pin rises to high level, the clock-stop state is released and oscillation begins.

The $\overline{\text{IRES}}$ signal halts release by timer carry FF. When the timer carry FF set pulse rises after the CE pin rises, the halt state is released and the program starts from address 0.

Since the timer carry FF set pulse is initialized to 100 ms, CE reset is applied 50 ms after the CE pin rises to high level.

Fig. 14-3 CE Reset Operation When Clock-Stop Used



14.3.3 Cautions at CE Reset

When CE reset is used, careful attention must be given to points (1) and (2) below regardless of the instruction being executed.

(1) Time required for clock and other timer processing

When writing a clock program by using timer carry FF and timer interrupts, the program must end processing within a certain time.

For details, see **Sections 12.4** and **12.6**.

(2) Processing of data, flags, etc. used in the program

Care must be exercised when rewriting the contents of data, flags, etc. that cannot be processed by one instruction so that the contents, such as the last channel, do not change even when CE reset is applied.

Two examples are given below:

Example 1.

```
; ①
LCTUNE :
    Initial reception
      The channel indicated by
      the contents of M1 and
      M2 is received.          ; The last channel is received.

MAIN :                         ; Main processing
    Channel change             ; The changed channel is assigned to general-purpose
                               ; registers R1 and R2.
; ②
    ST      M1, R1             ; The last channel is rewritten.
; ③
    ST      M2, R2
    BR      MAIN
```

In this example, if the last channel is 12H, then the data memory contents of M1 and M2 will be 1H and 2H, respectively.

When CE reset is applied, the last channel (Channel 12) is received in ①.

When the channel is changed in the main processing, the changed channel is rewritten to M1 and M2 in ② and ③.

When the channel is changed to 04H, 0H and 4H are rewritten to M1 and M2 in ② and ③. If CE reset is applied after ②, the reset process runs without executing ③.

Since this results in the last channel being 02H, Channel 02 is received in ①.

This can be avoided by using a program shown in example 2.

Example 2.

```
; ④
    SKT1    FLG1                ; If FLG1 is set to 1,
    BR      LCTUNE
    ST      M1, R1              ; data is rewritten to M1 and M2 again.
    ST      M2, R2
    CLR1    FLG1
; ①
LCTUNE :
    Initial reception
      The channel indicated by
      the contents of M1 and
      M2 is received.           ; The last channel is received.

MAIN :                          ; Main processing
    Channel change              ; The changed channel is assigned to general-purpose
                                ; registers R1 and R2.
; ⑤
    SET1    FLG1                ; FLG1 is set while rewriting the last channel.
; ②
    ST      M1, R1              ; The channel is rewritten.
; ③
    ST      M2, R2
    CLR1    FLG1
    BR      MAIN
```

In this example, FLG1 is set when rewriting the last channel in ② and ③. This allows data to be rewritten in ④ again even if CE reset is applied in ③.

14.4 POWER-ON RESET

Power-on reset is executed by raising V_{DD} from a certain voltage (called the power-on clear voltage) or less.

When V_{DD} is less than the power-on clear voltage, the power-on clear signal (POC) is output from the voltage detection circuit shown in Fig. 14-1.

When the power-on clear signal is output, the crystal oscillation circuit stops and the device stops operating.

While the power-on clear signal is being output, the $\overline{IRES}$, $\overline{RES}$ and $\overline{RESET}$ signals are output.

When V_{DD} exceeds the power-on clear voltage, the power-on clear signal is dropped and crystal oscillation starts. At the same time, the $\overline{IRES}$, $\overline{RES}$ and $\overline{RESET}$ signals are also dropped.

Since the $\overline{IRES}$ signal halts release by timer carry FF, power-on reset is applied at the rising edge of the next timer carry FF set signal.

Since the $\overline{RESET}$ signal has initialized the timer carry FF set signal to 100 ms, 50 ms after V_{DD} exceeds the power-on clear voltage, reset is applied and the program starts from address 0.

This operation is shown in Fig. 14-4.

At power-on reset, the program counter, stack, system register and control registers are initialized when the power-on clear signal is output.

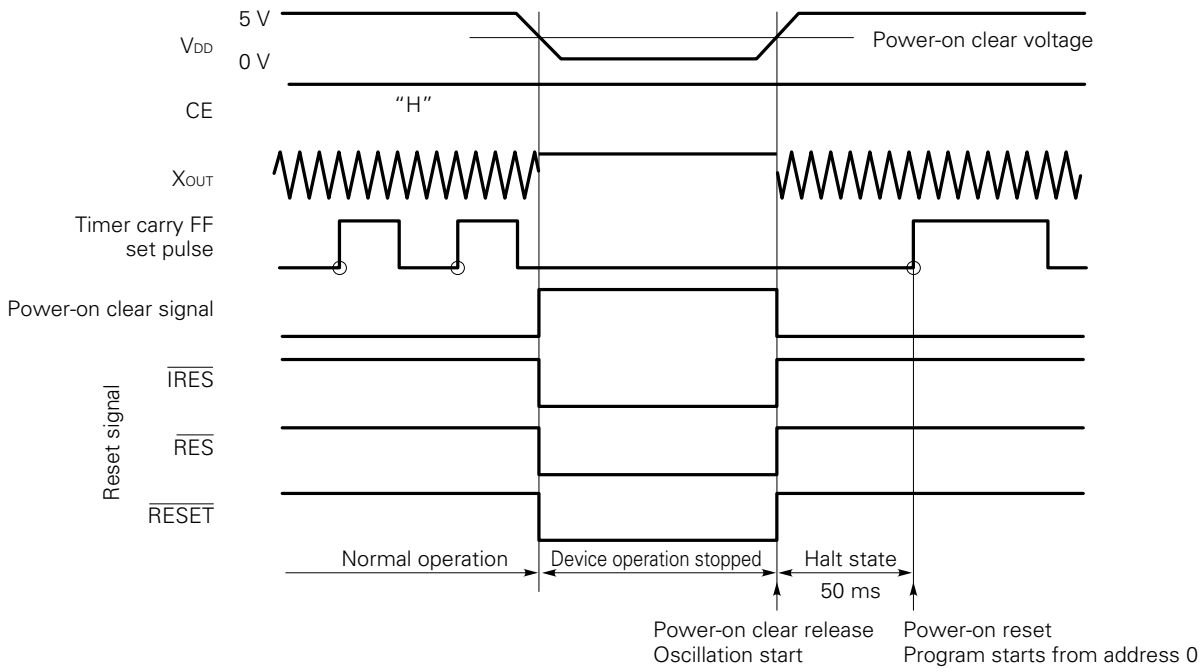
For the initial values, see the relevant items.

During normal operation, the power-on clear voltage is 3.5 V (rated value). In the clock-stop state, the power-on clear voltage becomes 2.2 V (rated value).

Sections 14.4.1 and **14.4.2** describe operation at this time.

Section 14.4.3 describes operation when V_{DD} rises from 0 V,

Fig. 14-4 Power-on Reset Operation



14.4.1 Power-on Reset at Normal Operation

Fig. 14-5 (a) shows power-on reset at normal operation.

As shown in Fig. 14-5 (a), when the V_{DD} drops below 3.5 V, the power-on clear signal is output and operation of the device stops regardless of the input level of the CE pin.

When V_{DD} then rises to 3.5 V or greater, after a 50 ms halt, the program starts from address 0000H.

Normal operation refers to the state in which the clock-stop instruction is not used. This also includes the halt state set by the halt instruction.

14.4.2 Power-on Reset in Clock-Stop State

Fig. 14-5 (b) shows power-on reset in the clock-stop state.

As shown in Fig. 14-5 (b), when V_{DD} drops below 2.2 V, the power-on clear signal is output and device operation stops.

However, since the device is in the clock-stop state, its operation apparently does not change.

When V_{DD} rises to 3.5 V or greater, after a 50 ms halt, the program starts from address 0000H.

14.4.3 Power-on Reset When V_{DD} Rises From 0 V

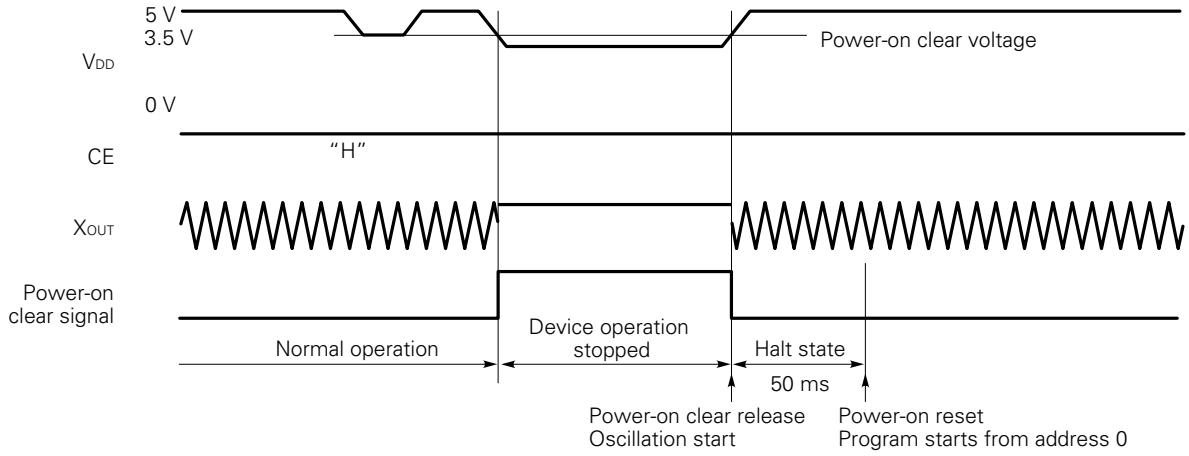
Fig. 14-5 (c) shows power-on reset when V_{DD} rises from 0 V.

As shown in Fig. 14-5 (c), the power-on clear signal is being output while V_{DD} is rising from 0 to 3.5 V.

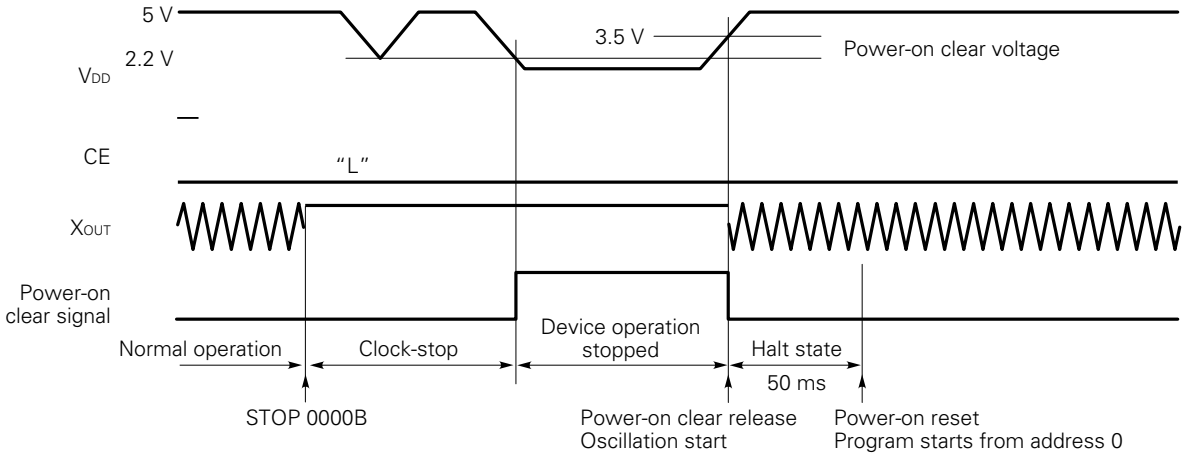
When V_{DD} rises above the power-on clear voltage, the crystal oscillation circuit starts and after a 50 ms halt, the program starts from address 0000H.

Fig. 14-5 Power-on Reset and V_{DD}

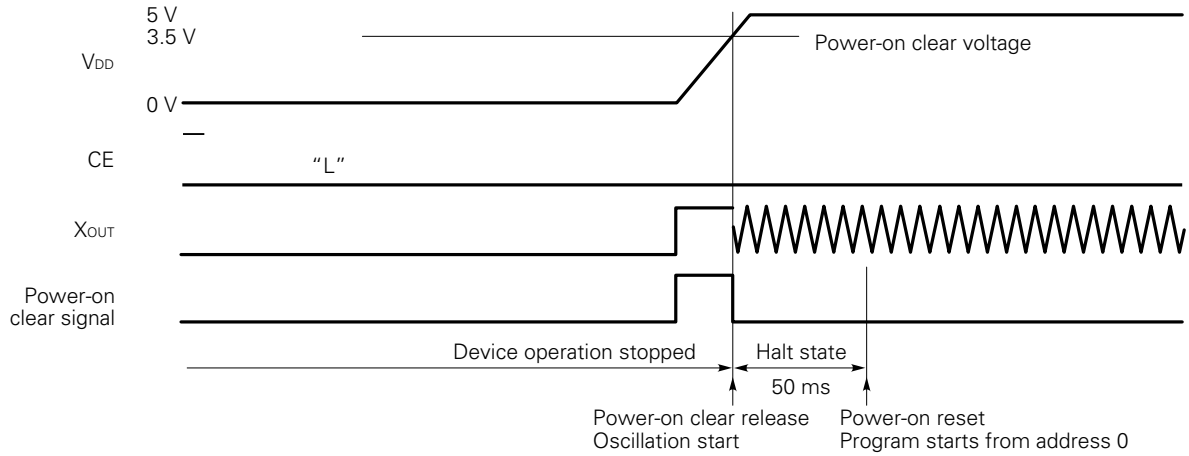
(a) During normal operation (including halt state)



(b) At clock-stop



(c) When V_{DD} rises from 0 V



14.5 RELATIONSHIP BETWEEN CE RESET AND POWER-ON RESET

When supply voltage is first turned on, power-on reset and CE reset may be applied simultaneously.

Sections 14.5.1 through **14.5.3** describe this reset operation.

Section 14.5.4 describes the cautions when supply voltage rises.

14.5.1 When V_{DD} Pin and CE Pin Rise Simultaneously

Fig. 14-6 (a) shows the reset operation. Power-on reset starts the program from address 0000H.

14.5.2 When CE Pin Raised in Forced Halt State Caused by Power-on Reset.

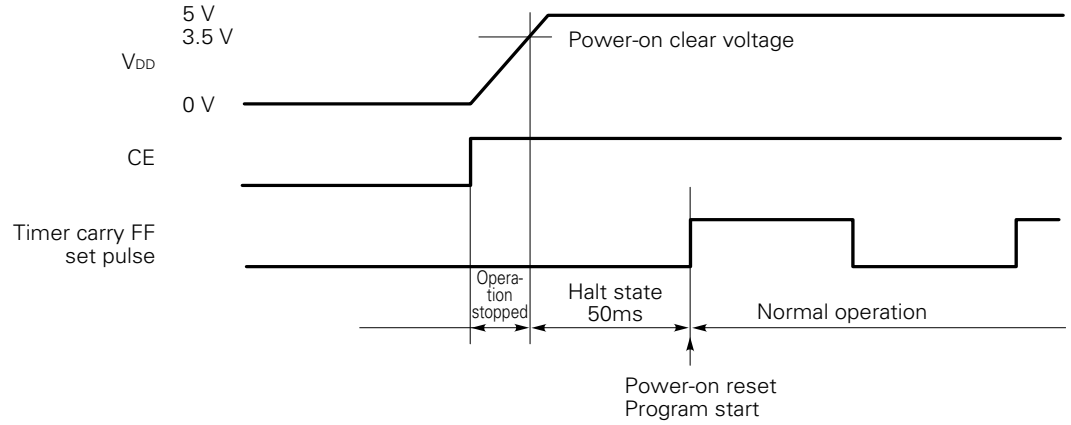
Fig. 14-6 (b) shows the reset operation. Power-on reset starts the program from address 0000H, as in **Section 14.5.1.**

14.5.3 When CE Pin Raised after Power-on Reset

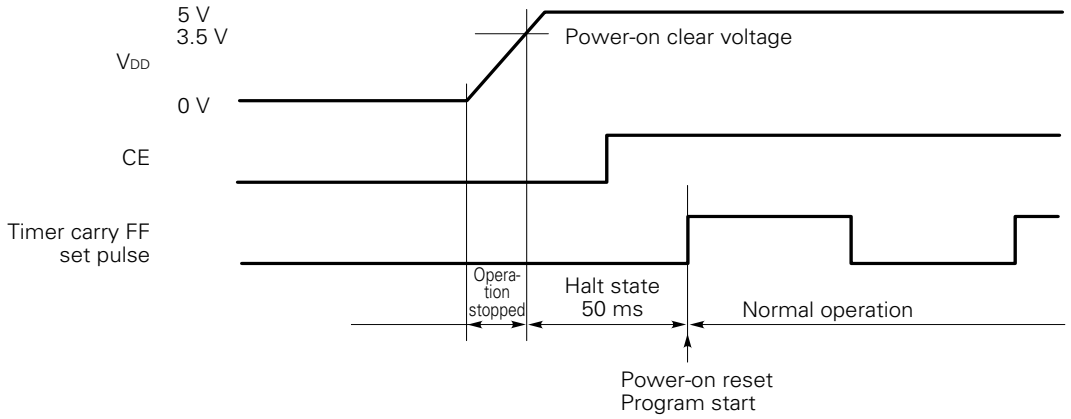
Fig. 14-6 (c) shows the reset operation. Power-on reset starts the program from address 0000H. CE reset restarts the program from address 0000H at the rising edge of the next timer carry FF set signal.

Fig. 14-6 Relationship Between Power-on Reset and CE Reset

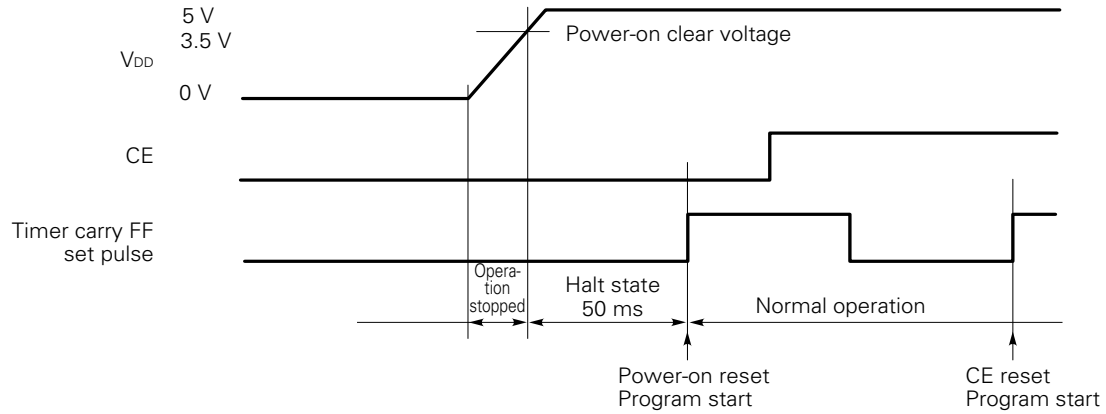
(a) When V_{DD} and CE pin raised simultaneously



(b) When CE pin raised in halt state



(c) When CE pin raised after power-on reset



14.5.4 Cautions When Supply Voltage Raised

When supply voltage is raised, careful attention must be given to points (1) and (2) below.

(1) When V_{DD} raised from power-on clear voltage

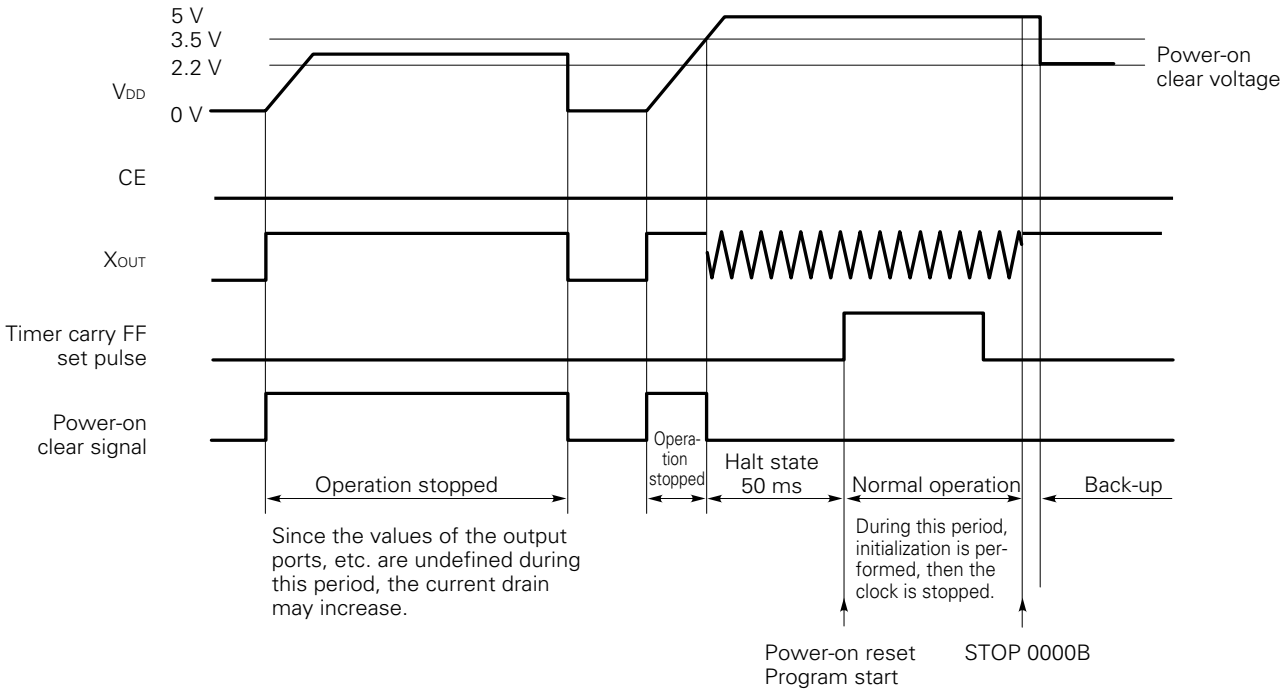
When V_{DD} is raised, it must be raised to 3.5 V or greater, once.

This is shown in Fig. 14-7.

As shown in Fig. 14-7, when a voltage under 3.5 V is applied when V_{DD} is turned on in a program that uses clock-stop to back up V_{DD} at 2.2 V, for example, the power-on clear signal continues to be output and the program does not run.

Since the device output port outputs an undefined value, the supply current increases, according to the situation, reducing the back-up time with a battery considerably.

Fig. 14-7 Caution When V_{DD} Raised



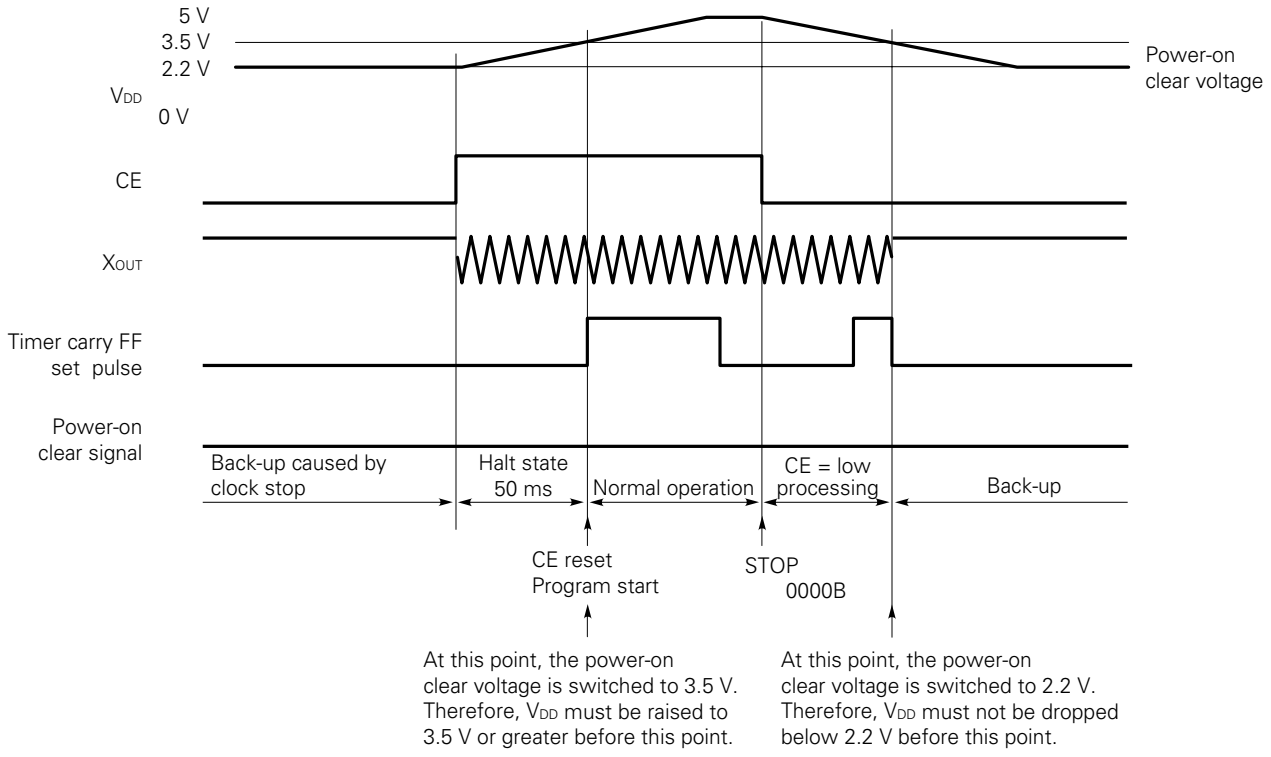
(2) At return from clock-stop state

When returning from the back-up state when clock-stop is used to back-up supply voltage at 2.2 V, V_{DD} must be raised to 3.5 V or greater within 50 ms after the CE pin becomes high level.

As shown in Fig. 14-8, return from the clock-stop state is performed by CE reset. Since the power-on clear voltage is switched to 3.5 V 50 ms after the CE pin is raised, if V_{DD} is not 3.5 V or greater at this time, power-on reset is applied.

The same caution is necessary when V_{DD} is dropped.

Fig. 14-8 Return from Clock-Stop State

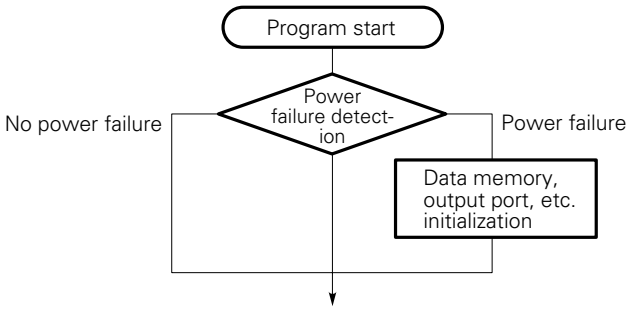


14.6 POWER FAILURE DETECTION

Power failure detection is used to judge whether the device is reset by turning on V_{DD} or by the CE pin, as shown in Fig. 14-9.

Since the contents of the data memory, output ports, etc. become "undefined" when V_{DD} is turned on, they are initialized by power failure detection.

Fig. 14-9 Power Failure Detection Flowchart



14.6.1 Power Failure Detection Circuit

As shown in Fig. 14-1, the power failure detection circuit consists of a voltage detection circuit and timer carry disable FF that is reset by the output (power-on clear signal) of the voltage detection circuit, and timer carry FF.

The timer carry disable FF is set to 1 by the power-on clear signal and is reset to 0 when a BTM0CY flag (address 17H, bit b_0) read instruction is executed.

When the timer carry disable FF is set to 1, the BTM0CY flag is not set to 1.

That is, when the power-on clear signal is output (at power-on reset), the program starts in the state in which the BTM0CY flag is reset and the setting disabled state is set until a BTM0CY read instruction is executed thereafter.

Once a BTM0CY read instruction is executed, the BTM0CY flag is set at each rising edge of the timer carry FF set pulse thereafter. When reset is applied to the device, the contents of the BTM0CY flag are monitored. If the BTM0CY flag has been reset to 0, power-on reset (power failure) is judged and if the BTM0CY flag has been set to 1, CE reset (no power failure) is judged.

Since the voltage that can detect a power failure is the same as the voltage applied by power-on reset, V_{DD} becomes 3.5 V at clock oscillation and 2.2 V at clock-stop.

Fig. 14-10 shows the BTM0CY flag state transition.

Fig. 14-11 shows timing chart and BTM0CY flag operation specified in Fig. 14-10.

Fig. 14-10 BTM0CY Flag State Transition

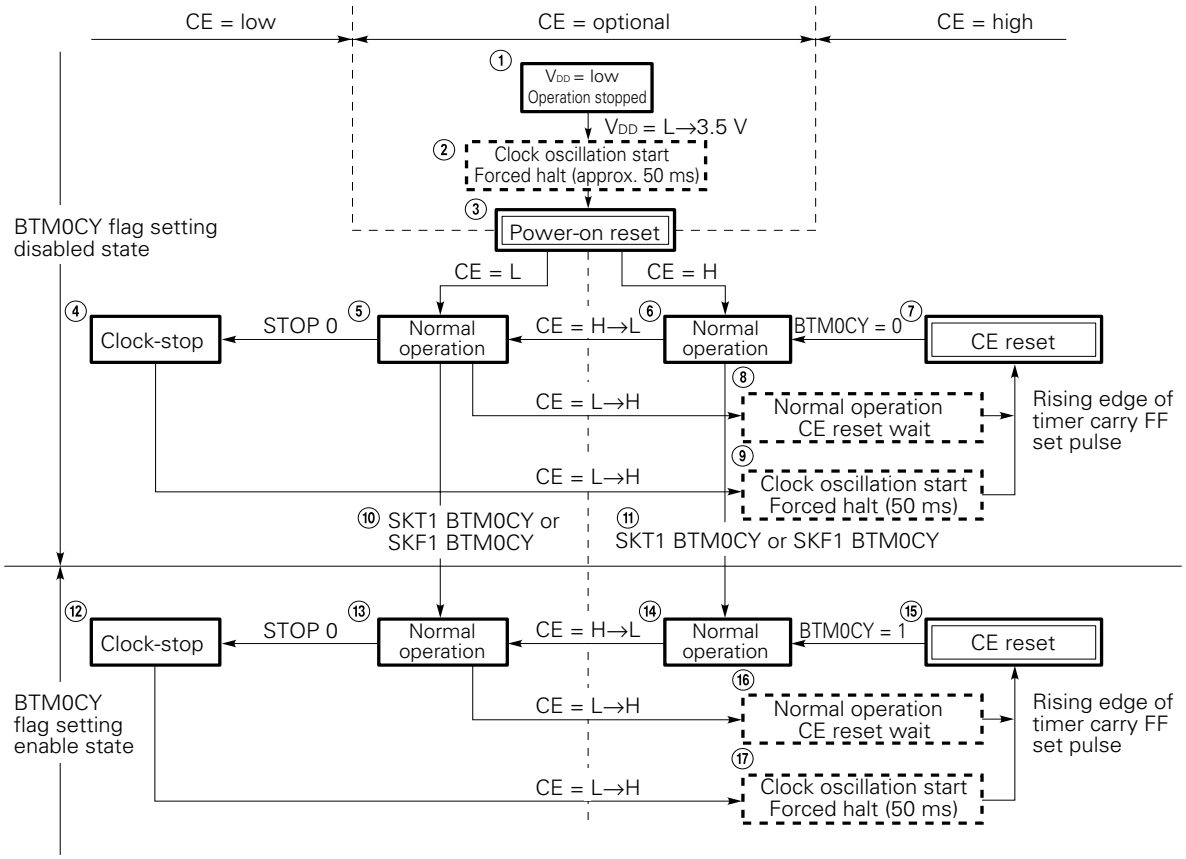
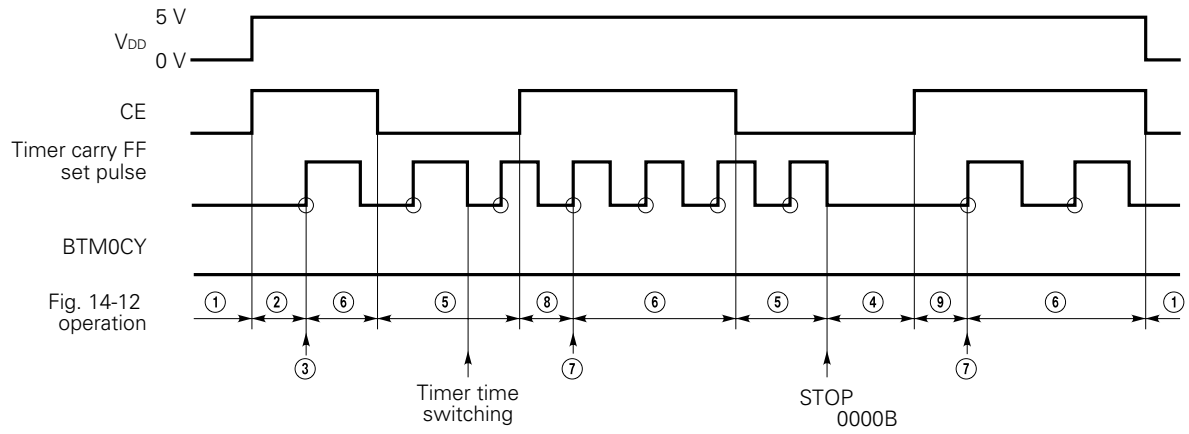
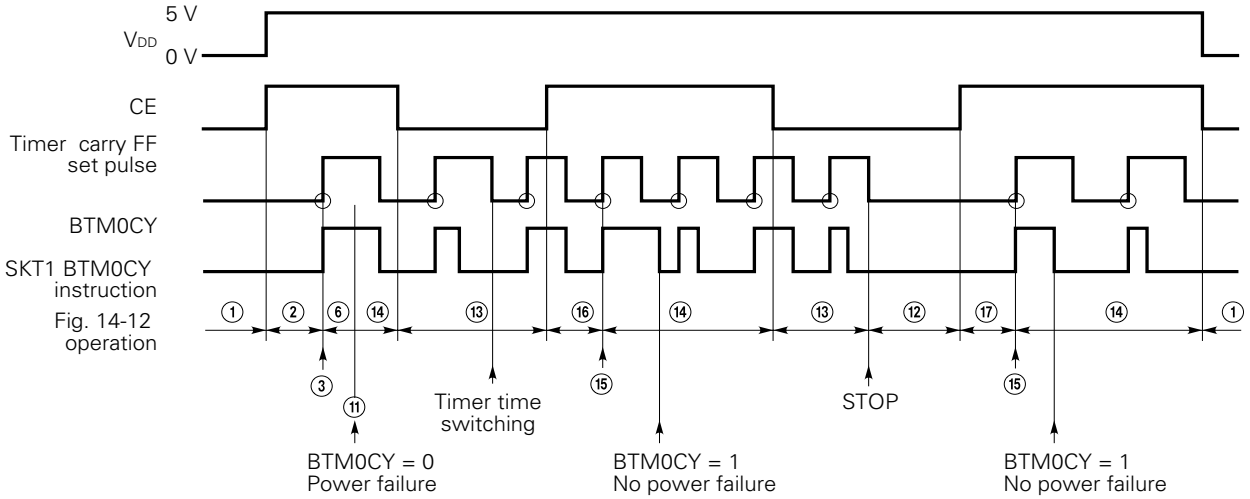


Fig. 14-11 BTM0CY Flag Operation

(a) When BTM0CY flag not detected even once (neither SKT1 BTM0CY nor SKF1 BTM0CY executed)



(b) When power failure detected with BTM0CY flag



14.6.2 Cautions at Power Failure Detection with BTM0CY Flag

When clock counting, etc. is performed with the BTM0CY flag, careful attention must be given to the following points.

(1) Clock updating

When writing a clock program by using the timer carry FF, the clock must be updated after a power failure. This is because the BTM0CY flag is reset to 0 and one clock count is lost by BTM0CY flag reading when a power failure is detected.

(2) Clock update processing time

When the clock is updated, its processing must end before the next rising edge of the timer carry FF set pulse.

This is because if the CE pin rises to high level during clock update processing, the clock update processing will not be executed up to the end and a CE reset will be applied.

For (1) and (2) above, see **Section 12.4.2.**

When processing is performed at a power failure, careful attention must be given to the following point.

(3) Power failure detection timing

When clock counting, etc. is performed with the BTM0CY flag, the flag must be read for power-failure detection before the next rising edge of the timer carry FF set pulse, after a program starts from address 0000H.

This is because when the timer carry FF set time is set to 5 ms, for instance, and power failure detection is performed 6 ms after the program starts, one BTM0CY flag is lost.

See **Section 12.4.2.**

As shown in the example on the next page, power failure detection and initialization must be performed within the timer carry FF set time.

This is because when the CE pin is raised and CE reset is applied during power failure processing and initialization, these processings are interrupted and a problem may occur.

When the timer carry FF set time is changed in initialization, an instruction that makes the change must be executed at the end of initialization.

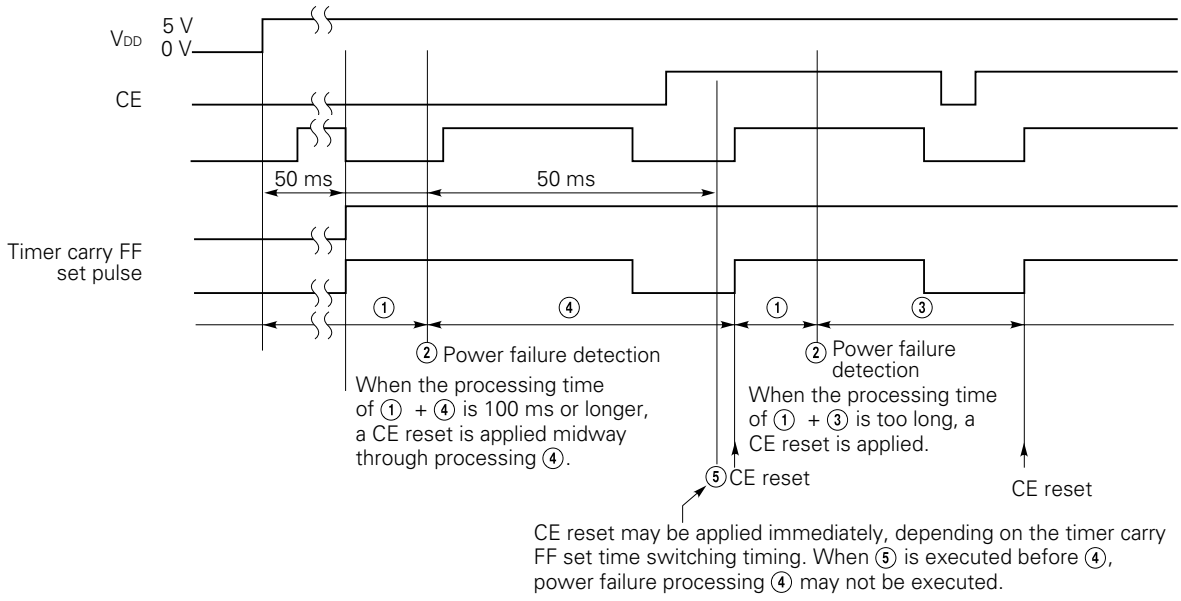
This is also because when the timer carry FF set time is switched before initialization as shown in the example on the next page, initialization by CE reset may not be executed up to the end.

Example

Sample program

```
START:                    ; Program address 0000H
    ; ①
    Reset processing      ;
    ; ②
    SKT1   BTM0CY         ; Power failure detection
    BR     INITIAL
BACKUP:
    ; ③
    Clock updating
    BR     MAIN
INITIAL:
    ; ④
    Initialization
    ; ⑤
    INITFLG  NOT BTM0ZX, NOT BTM0CK2, NOT BTM0CK1, BTM0CK0
                          ; Built-in macro
                          ; Sets timer carry FF set time to 5 ms.
MAIN:
    SKT1   BTM0CY
    BR     MAIN
    Clock updating
```

Operation example



15. GENERAL-PURPOSE PORT

A general-purpose port outputs a high level, low level, or floating signal to an external circuit and reads a high level or low level signal from an external circuit.

15.1 CONFIGURATION AND CLASSIFICATION OF GENERAL-PURPOSE PORT

Fig. 15-1 shows a block diagram of the general-purpose port.

Table 15-1 lists the classifications of general-purpose ports.

As shown in Fig. 15-1, the general-purpose port consists of Port0A (P0A) to Port1C (P1C), that set data according to addresses 70H to 73H (port register) in each bank of data memory. The same port register is mapped to both BANK0 and BANK2.

Each port consists of general-purpose port pins. For example, Port0A consists of pin $P0A_3$ to pin $P0A_0$.

As stated in Table 15-1, general-purpose ports are classified into I/O shared ports (I/O ports), input-only ports (input ports), and output-only ports (output ports).

I/O ports are classified into bit I/O ports which allow I/O to be specified in 1-bit (1-pin) units and group I/O ports in which I/O can be specified in 3-bit (3-pin) units .

Fig. 15-1 Block Diagram of General-Purpose Port

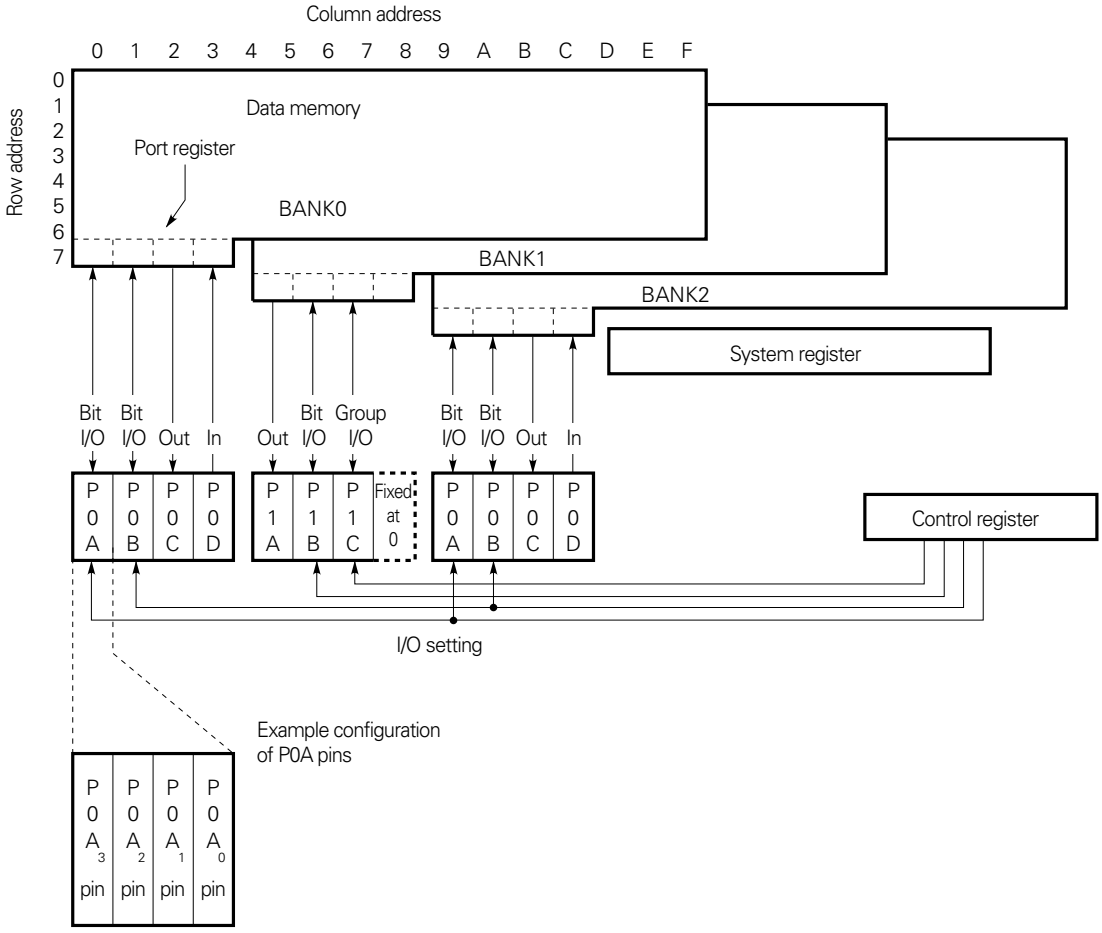


Table 15-1 Classification of General-Purpose Ports

Classification of general-purpose ports			Target ports	Data setting method
General-purpose ports	I/O shared port	Bit I/O	Port0A Port0B Port1B	Port register
		Group I/O	Port1C	Port register
	Input-only port		Port0D	Port register
	Output-only port		Port0C Port1A	Port register

15.2 FUNCTIONS OF GENERAL-PURPOSE PORTS

A general-purpose I/O port, set up either as a general-purpose output port or output port, outputs high level or low level signals from each corresponding pin by setting data in the port register accordingly.

A general-purpose I/O port, set up either as a general-purpose input port or input port, detects the level of the input signal applied to each corresponding pin by reading the contents of the port register.

General-purpose I/O ports are set either as an input ports or output ports, according to the contents of the control register for each port.

This enables I/O switching to be done by the program.

Since P0A to P0D and P1A to P1C are set as general-purpose ports at power-on reset, other pins that are used as peripheral hardware are set up independently according to the contents of the corresponding control register.

Sections 15.2.1 to **15.2.4** describe the functions of the port register and outline the functions of each port.

15.2.1 General-Purpose Port Data Register (Port Register)

The port register sets output data for each general-purpose port and reads input data.

Since the port register is mapped into data memory, it can be manipulated by all data memory manipulation instructions.

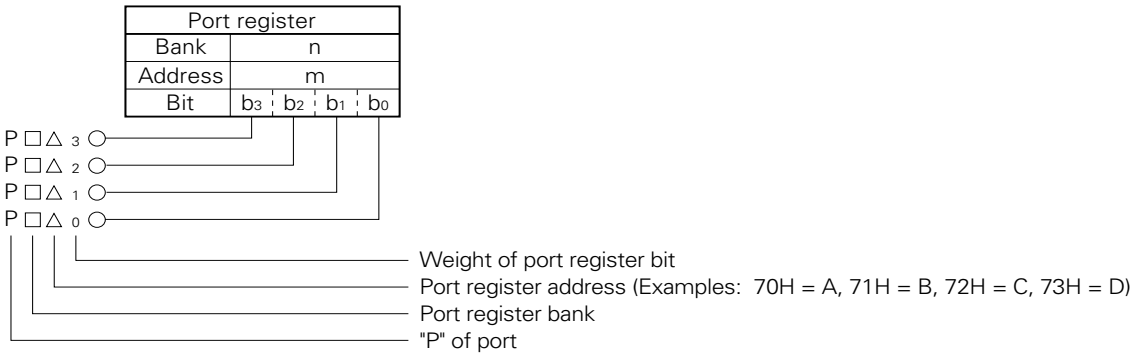
Fig. 15-2 shows the relationship between the port register and each corresponding pin.

Output for each pin is set by setting data in the port register for the pin set as a general-purpose output port.

The input state of each pin is detected by reading the contents of the port register for the pin set as a general-purpose output port.

Table 15-2 shows the relationship between each port (each pin) and the port register.

Fig. 15-2 Relationship Between Port Register and Each Pin



Reserved words are defined in the port register by the assembler.

Since reserved words are defined in flag units (bits), the assembler built-in macro instructions can be used.

Note that reserved words of data memory type are not defined in the port register.

15.2.2 General-Purpose I/O Ports (P0A, P0B, P1B, P1C)

The I/O of P0A is switched by the P0A bit I/O selection register (RF address 37H). The I/O of P0B is switched by the P0B bit I/O selection register (RF address 36H). The I/O of P1B is switched by the P1B bit I/O selection register (RF address 35H). And, the I/O of P1C is switched by the P1C group I/O selection register (RF address 27H).

The I/O data of P0A is set by P0A (data memory address: 70H of BANK0 or BANK2) of the port register. The I/O data of P0B is set by P0B (data memory address: 71H of BANK0 or BANK2) of the port register. The I/O data of P1B is set by P1B (data memory address: 71H of BANK1) of the port register. And, the I/O data of P1C is set by P1C (data memory address: 72H of BANK1) of the port register.

See Table 15-2.

For details, see **Section 15.3**.

15.2.3 General-Purpose Input Port (P0D)

P0D input data is read by P0D (data memory address: 73H of BANK0 or BANK2) of the port register.

See Table 15-2.

For details, see **Section 15.4**.

15.2.4 General-Purpose Output Ports (P0C, P1A)

(1) P0C, P1A

P0C and P1A output data is set by P0C (data memory address: 72H of BANK0 or BANK2) and P1A (data memory address: 70H of BANK1) of the port register.

See Table 15-2.

For details, see **Section 15.5**.

Table 15-2 Relationship between Each Port (Pin) and Port Register

Port	Pin		Data setting method				
			Port register (data memory)				
	Symbol	I/O	Bank	Address	Symbol	Bit symbol (reserved word)	
Port0A (P0A)	$P0A_3$	I/O (bit I/O)	BANK0 BANK2	70H	P0A	b_3	P0A3
	$P0A_2$					b_2	P0A2
	$P0A_1$					b_1	P0A1
	$P0A_0$					b_0	P0A0
Port0B (P0B)	$P0B_3$	I/O (bit I/O)		71H	P0B	b_3	P0B3
	$P0B_2$					b_2	P0B2
	$P0B_1$					b_1	P0B1
	$P0B_0$					b_0	P0B0
Port0C (P0C)	$P0C_3$	Output		72H	P0C	b_3	P0C3
	$P0C_2$					b_2	P0C2
	$P0C_1$					b_1	P0C1
	$P0C_0$					b_0	P0C0
Port0D (P0D)	$P0D_3$	Input		73H	P0D	b_3	P0D3
	$P0D_2$					b_2	P0D2
	$P0D_1$					b_1	P0D1
	$P0D_0$					b_0	P0D0
Port1A (P1A)	$P1A_3$	Output	BANK1	70H	P1A	b_3	P1A3
	$P1A_2$					b_2	P1A2
	$P1A_1$					b_1	P1A1
	$P1A_0$					b_0	P1A0
Port1B (P1B)	$P1B_3$	I/O (bit I/O)		71H	P1B	b_3	P1B3
	$P1B_2$					b_2	P1B2
	$P1B_1$					b_1	P1B1
	$P1B_0$					b_0	P1B0
Port1C (P1C)	$P1C_3$	I/O (group I/O)		72H	P1C	b_3	P1C3
	$P1C_2$					b_2	P1C2
	$P1C_1$					b_1	P1C1
	No target pins					b_0	Note

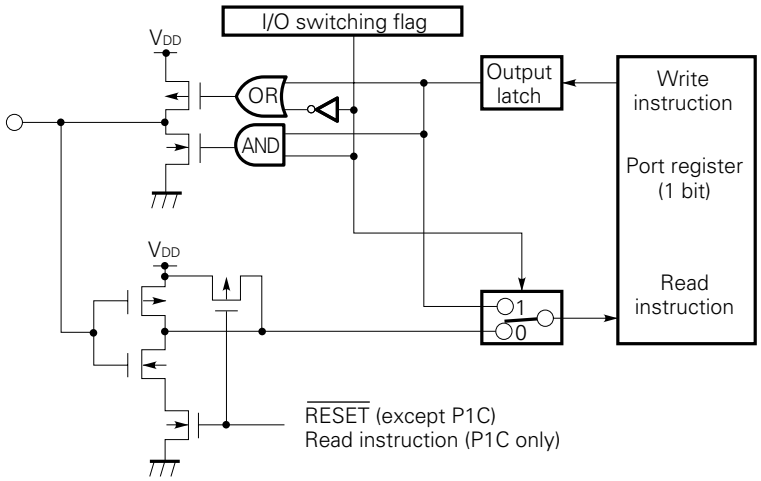
Note Nothing is mapped to b_0 of 72H. When b_0 is read, 0 is always read.

15.3 GENERAL-PURPOSE I/O PORTS (P0A, P0B, P1B, P1C)

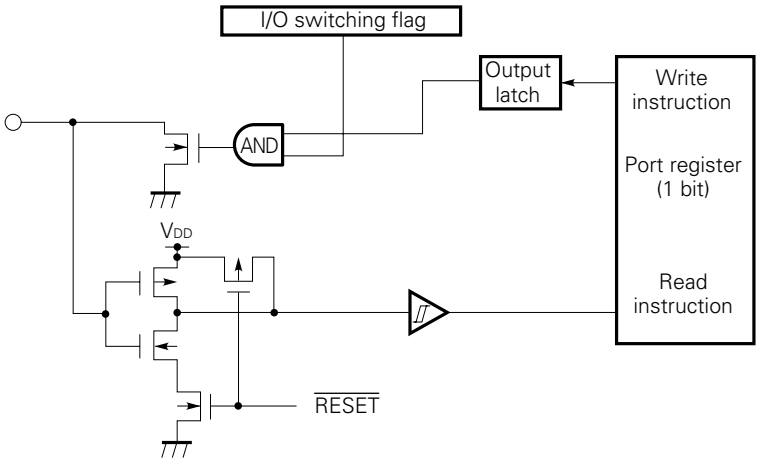
15.3.1 Configuration of I/O Ports

In the following, (1) to (3) explain the configuration of the I/O ports.

(1) P0A ($P0A_3$, $P0A_2$ pins)
P0B ($P0B_3$, $P0B_2$, $P0B_1$, $P0B_0$ pins)
P1B ($P1B_3$, $P1B_2$, $P1B_1$, $P1B_0$ pins)
P1C ($P1C_3$, $P1C_2$, $P1C_1$ pins)



(2) P0A ($P0A_1$, $P0A_0$ pins)



15.3.2 How to Use I/O Ports

An I/O port is set as an input or output port according to the contents of each I/O selection register of P0A, P0B, P1B, and P1C of the control register.

I/O of the bit I/O port (P0A, P0B, P1B) can be set in 1-bit (1-pin) units. I/O of the group I/O port (P0C) can be set in 3-bit (3-pin) units.

Output data is set and input data is read when a data write instruction or data read instruction is executed in the corresponding port register.

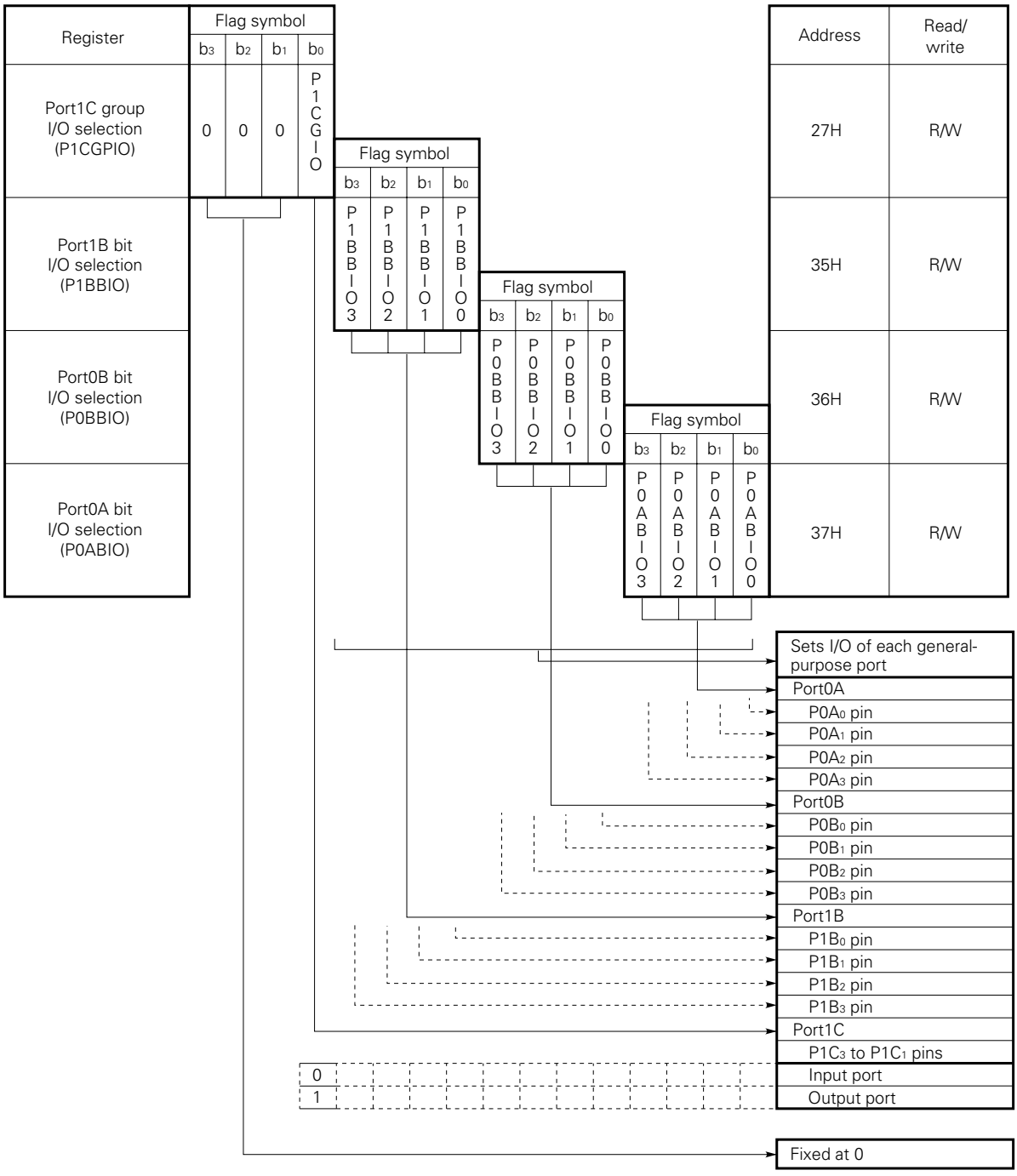
Section 15.3.3 describes the I/O selection register of each port.

Sections 15.3.4 and **15.3.5** explain the use of an input port and output port.

15.3.3 Port0A Bit I/O Selection Register (P0ABIO)
Port0B Bit I/O Selection Register (P0BBIO)
Port1B Bit I/O Selection Register (P1BBIO)
Port1C Group I/O Selection Register (P1CGPIO)

The Port0A bit I/O selection register sets I/O for each pin of P0A. The Port0B bit I/O selection register sets I/O for each pin of P0B. The Port1B bit I/O selection register sets I/O for each pin of P1B. The Port1C group I/O selection register sets I/O for each pin of P1C.

The following explains the configuration and functions.



At reset																	
	Power-on	0	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	0
	CE	↓	↓	↓	0	0	0	0	0	0	0	0	0	0	0	0	0

15.3.4 To Use an I/O Port (P0A, P0B, P1B, P1C) as an Input Port

Select the pin to be used as an input port by using the I/O selection register of each port.

P1C can be set to I/O in 3-bit (3-pin) units only.

The pin specified as an input port enters floating (Hi-Z) status and waits for the input of an external signal.

Input data can be read by executing an instruction to read the contents of the port register for each pin, for example, the SKT instruction.

When a high level signal is input to each pin of the port register, 1 is read. When a low level signal is input to each pin of the port register, 0 is read.

Upon the execution of an instruction, such as the MOV instruction, to write to the port register specified as an input port, the contents of the output latch are overwritten.

15.3.5 To Use an I/O Port (P0A, P0B, P1B, P1C) as an Output Port

Select the pin to be used as an output port by using the I/O selection register of each port.

P1C can be set to I/O in 3-bit (3-pin) units only.

The pin specified as an output port outputs the contents of the output latch from each pin.

Output data can be set by executing an instruction to write the contents of the port register for each pin, for example, the MOV instruction.

To output a high level signal to each pin, write 1. To output a low level signal to each pin, write 0.

The pin can be set to the floating state (Hi-Z) by specifying it as an input port.

Upon executing an instruction, such as the SKT instruction, for reading the port register specified as an output port, the contents of the output latch are read.

Note, however, that the contents of the output latch and the read contents may differ because the two pins, $P0A_1$ and $P0A_0$, are read without changing the pin state.

See **Section 15.3.6**.

15.3.6 Notes on Using I/O Ports ($P0A_1$ and $P0A_0$)

As shown in the example below, when pins $P0A_1$ and $P0A_0$ pins are used as output pins, the contents of the output latch may be overwritten.

Example:

```
INITFLG   NOT P0ABIO3, NOT P0ABIO2, P0ABIO1, P0ABIO0
                        ; Set the P0A1, P0A0 pins as output pins
INITFLG   NOT P0A3, NOT P0A2, P0A1, P0A0
; ①                     ; Output a high level signal to the P0A1 and P0A0 pins
CLR1      P0A1          ; Output a low level signal to the P0A1 pin
; Macro expansion
    AND .MF.P0A1 SHR 4, #.DF. (NOT P0A1 AND 0FH)
```

If the $P0A_0$ pin is externally pulled down to the low level upon execution of instruction ①, above, the CLR1 instruction overwrites the contents of the output latch of the $P0A_0$ pin with 0.

15.3.7 State of I/O Port (P0A, P0B, P1B, P1C) at Reset

(1) At power-on reset

All I/O ports are set as input ports.

Since the contents of the output latch are "indefinite", the output latch must be initialized by the program before the ports can be switched to output ports.

(2) At CE reset

All I/O ports are set as input ports.

The contents of the output latch are retained.

(3) At clock stop

All I/O ports are set as input ports.

The contents of the output latch are retained.

In I/O ports other than P1C, the $\overline{\text{RESET}}$ signal output at clock stop prevents the current drain from being increased by noise from the input buffer, as shown in **Section 15.3.1**.

(4) During the halt state

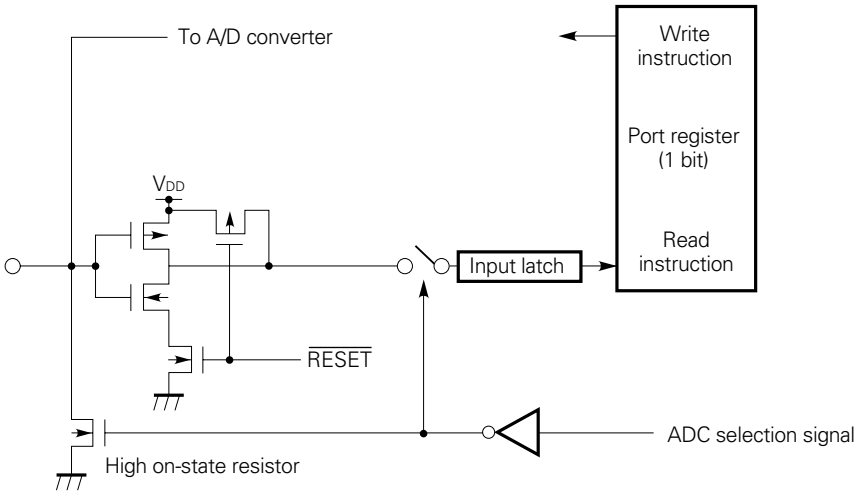
The previous state is retained.

15.4 GENERAL-PURPOSE INPUT PORT (P0D)

15.4.1 Configuration

The following explains the configuration of the input port.

(1) P0D (P0D$_3$, P0D$_2$, P0D$_1$, P0D$_0$ pins)



15.4.2 Example of Using Input Port (P0D)

Input data can be read by executing an instruction, such as the SKT instruction, to read the contents of the port register for each pin.

When a high level signal is input to each pin of the port register, 1 is read. When a low level signal is input to each pin of the port register, 0 is read.

When a write instruction, such as the MOV instruction, is executed, the port register remains as is.

15.4.3 Notes on Using Input Port (P0D)

P0D is internally pulled down when being used as a general-purpose port.

15.4.4 State of Input Port (P0D) upon Reset

(1) At power-on reset

All I/O ports are set as general-purpose input ports.

(2) At CE reset

All I/O ports are set as general-purpose input ports.

(3) At clock stop

All I/O ports are set as general-purpose input ports.

The $\overline{\text{RESET}}$ signal, output upon clock stop, prevents the current drain from increasing due to noise from the input buffer, as explained in **Section 15.4.1**.

P0D is pulled down internally.

(4) During halt state

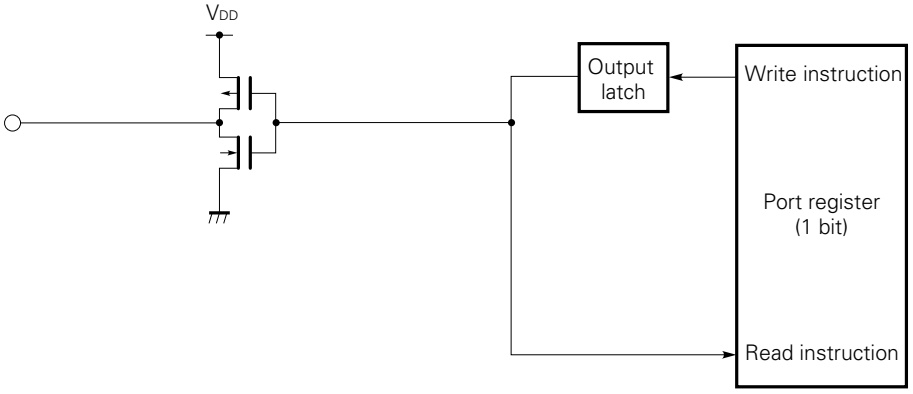
The previous state is retained.

15.5 GENERAL-PURPOSE OUTPUT PORTS (P0C, P1A)

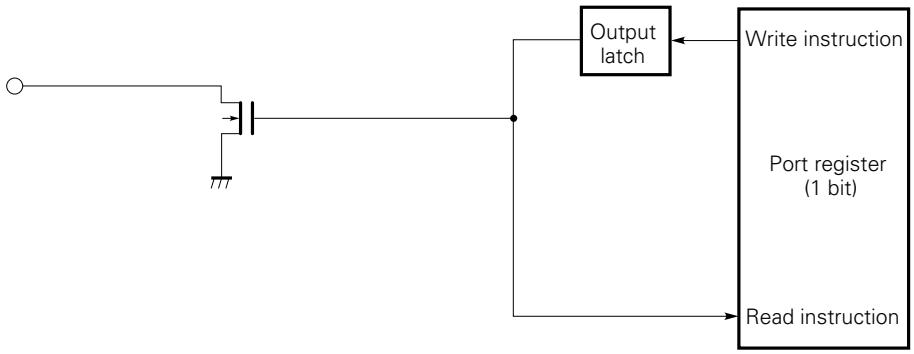
15.5.1 Configuration of Output Ports (P0C, P1A)

(1) and (2), below, show the configuration of the output ports.

(1) P0C ($P0C_3$, $P0C_2$, $P0C_1$, $P0C_0$ pins)



(2) P1A ($P1A_3$, $P1A_2$, $P1A_1$, $P1A_0$ pins)



15.5.2 Example of Using Output Ports (P0C, P1A)

The output ports output the contents of the output latch from each pin.

Output data can be set by executing an instruction, such as the MOV instruction, to write the contents of the port register for each pin.

To output a high level signal to each pin, write 1. To output a low level signal to each pin, write 0.

However, the $P1A_3$, $P1A_2$, $P1A_1$, and $P1A_0$ pins, because of the N-ch open-drain output, are set to the floating state (Hi-Z) when a high level signal is output.

Upon executing an instruction to read the port register, such as the SKT instruction, the contents of the output latch are read.

15.5.3 State of Output Ports (P0C, P1A) at Reset

(1) At power-on reset

The contents of the output latch are output.

Since the contents of the output latch are "indefinite", "indefinite" values are output until the output latch is initialized by the program.

(2) At CE reset

The contents of the output latch are output.

Since the contents of the output latch are retained, the output data remains as is upon a CE reset.

(3) At clock stop

The contents of the output latch are output.

Since the contents of the output latch are retained, the output data remains as is at clock stop.

Therefore, the output latch must be initialized by the program as necessary.

(4) During halt state

The contents of the output latch are output.

Since the contents of the output latch are retained, the output data remains as is during the halt state.

16. SERIAL INTERFACE

The μPD17062 has two sets of serial interface pins, channel 0 (CH0) and channel 1 (CH1), for exchanging data with an external unit.

The CH0 pin, which consists of two wires, SDA and SCL, can be operated in any of three modes, clock synchronous two-wire serial input, clock synchronous two-wire serial output, and two-wire bus[Note]. The SDA and SCL pins can be used as general-purpose ports when not being used as a serial interface.

The CH1 pin, which consists of three wires, $\overline{SCK}$, SO, and SI, can be operated in any of three modes, clock synchronous two-wire serial I/O input, clock synchronous two-wire serial I/O output, and three-wire serial I/O.

CH0 and CH1 cannot be operated at the same time. Which of pins CH0 and CH1 is used is specified with the SIO0CH flag (register file: 08H, b_3) of SMODE (Serial Interface Mode Register).

The two-wire hardware-supported bus mode is for the single master. Therefore, this mode does not support any arbitration function. Arbitration must be done by the software.

Note The two-wire bus mode can be used as an I^2C bus.

Table 16-1 External Pins for Serial Interface

CH	Pin name	Function		
		Two-wire bus mode	Serial I/O mode	Port I/O setting register
0	$P0A_0$/SDA	Serial data I/O	Serial data I/O	P0ABIO0
	$P0A_1$/SCL	Shift clock I/O	Shift clock I/O	P0ABIO1
1	$P0A_2/\overline{SCK}$	Cannot be used	Shift clock I/O	P0ABIO2
	$P0A_3$/SO		Serial data output	P0ABIO3
	$P0B_0$/SI		Serial data input	P0BBIO0

16.1 SERIAL INTERFACE MODE REGISTER

The serial interface mode register specifies the operation mode of the serial interface. This register sets up the channel to be used, the protocol, clock, and transmission/reception.

This register is mapped to address 08H in the register file.

All flags of this register are set to 0 at power-on reset.

Fig. 16-1 Configuration of Serial Interface Mode Register

Bit position	b_3	b_2	b_1	b_0
Flag name	SIO0CH	SB	SIO0MS	SIO0TX

Table 16-2 CH0 Operation Modes

Serial interface mode register			Port 0A I/O specification		SDA pin	SCL pin	Operation mode
SB	SIO0MS	SIO0TX	P0ABIO0	P0ABIO1			
0	0	0	0	0	SD-IN	CK-IN	Serial I/O-SI, EXT-CLK
0	0	0	0	1	SD-IN	OUT-PORT	Serial I/O-SI, INT-CLK(SOFT-CLK)
0	0	0	1	0	OUT-PORT	IN-PORT	1OUT-PORT+1IN-PORT
0	0	0	1	1	OUT-PORT	OUT-PORT	2OUT-PORT
0	0	1	×	0	SD-OUT	CK-IN	Serial I/O-SO, EXT-CLK
0	0	1	×	1	SD-OUT	OUT-PORT	Serial I/O-SO, INT-CLK(SOFT-CLK)
0	1	0	0	×	SD-IN	CK-OUT	Serial I/O-SI, INT-CLK
0	1	0	1	×	OUT-PORT	CK-OUT	CLK-OUT+1OUT-PORT
0	1	1	×	×	SD-OUT	CK-OUT	Serial I/O-SO, INT-CLK
1	0	0	0	0	SD-IN	CK-IN	BUS-SLAVE-RX
1	0	0	0	1	SD-IN	OUT-PORT	BUS-MASTER-RX(SOFT-CLK)
1	0	0	1	0	OUT-PORT	IN-PORT	1OUT-PORT+1IN-PORT
1	0	0	1	1	OUT-PORT	OUT-PORT	2OUT-PORT
1	0	1	×	0	SD-OUT	CK-IN	BUS-SLAVE-TX
1	0	1	×	1	SD-OUT	OUT-PORT	BUS-MASTER-TX(SOFT-CLK)
1	1	0	0	×	SD-IN	CK-OUT	BUS-MASTER-RX
1	1	0	1	×	OUT-PORT	CK-OUT	CLK-OUT+1OUT-PORT
1	1	1	×	×	SD-OUT	CK-OUT	BUS-MASTER-TX

Remark ×: Don't care

Table 16-3 CH1 Operation Modes

Serial interface mode register			Port 0A I/O specification			SI pin	$\overline{SCK}$ pin	SO pin	Operation mode
SB	SIO0MS	SIO0TX	P0ABIO2	P0ABIO3	P0BBIO0				
0	0	0	0	0	0	SD-IN	CK-IN	IN-PORT	Serial I/O-SI, EXT-CLK, 1IN-PORT
0	0	0	0	0	1	SD-IN	CK-IN	OUT-PORT	Serial I/O-SI, EXT-CLK, 1OUT-PORT
0	0	0	0	1	0	OUT-PORT	IN-PORT	IN-PORT	1OUT-PORT+2IN-PORT
0	0	0	0	1	1	OUT-PORT	IN-PORT	OUT-PORT	2OUT-PORT+1IN-PORT
0	0	0	1	0	0	SD-IN	OUT-PORT	IN-PORT	Serial I/O-SI, INT-CLK (SOFT-CLK), 1IN-PORT
0	0	0	1	0	1	SD-IN	OUT-PORT	OUT-PORT	Serial I/O-SI, INT-CLK (SOFT-CLK), 1IN-PORT
0	0	0	1	1	0	OUT-PORT	OUT-PORT	IN-PORT	2OUT-PORT + 1IN-PORT
0	0	0	1	1	1	OUT-PORT	OUT-PORT	OUT-PORT	3OUT-PORT
0	0	1	0	0	×	SD-IN	CK-IN	SD-OUT	Serial I/O-SI/SO, EXT-CLK
0	0	1	0	1	×	OUT-PORT	CK-IN	SD-OUT	Serial I/O-SO, EXT-CLK, 1OUT-PORT
0	0	1	1	0	×	SD-IN	OUT-PORT	SD-OUT	Serial I/O-SI/SO, INT-CLK (SOFT-CLK)
0	0	1	1	1	×	OUT-PORT	OUT-PORT	SD-OUT	Serial I/O-SO, INT-CLK (SOFT-CLK), 1OUT-PORT
0	1	0	×	0	0	SD-IN	CK-OUT	IN-PORT	Serial I/O-SI, INT-CLK, 1IN-PORT
0	1	0	×	0	1	SD-IN	CK-OUT	OUT-PORT	Serial I/O-SI, INT-CLK, 1OUT-PORT
0	1	0	×	1	0	OUT-PORT	CK-OUT	IN-PORT	CLK-OUT, 1OUT-PORT, 1IN-PORT
0	1	0	×	1	1	OUT-PORT	CK-OUT	OUT-PORT	CLK-OUT, 2OUT-PORT
0	1	1	×	0	×	SD-IN	CK-OUT	SD-OUT	Serial I/O-SI/SO, INT-CLK
0	1	1	×	1	×	OUT-PORT	CK-OUT	SD-OUT	Serial I/O-SO, INT-CLK, 1OUT-PORT
1	×	×	×	×	×	–	–	–	Not to be set

Remark ×: Don't care

16.1.1 SIO0CH

The SIO0CH flag is used to select the channel of the serial interface.

When the SIO0CH flag is set to 0, the serial interface hardware is connected to CH0. When the SIO0CH flag is set to 1, the serial interface hardware is connected to CH1.

The external pin of the unselected channels is used as a general-purpose port.

Table 16-4 Channel Setting of Serial Interface

SIO0CH	Channel to be selected
0	CH0
1	CH1

16.1.2 SB

The SB flag specifies the serial interface protocol.

When the SB flag is set to 0, serial I/O mode is specified. When the SB flag is set to 1, two-wire bus mode is specified.

Since CH1 does not support two-wire bus mode, the SB flag must be set to 0 when CH1 is used.

Table 16-5 Specification of Serial Interface Protocol

SB	Protocol
0	Serial I/O mode
1	Two-wire bus mode

16.1.3 SIO0MS

The SIO0MS flag specifies the serial interface clock to be used.

When the SIO0MS flag is set to 0, the external clock is selected. When the SIO0MS flag is set to 1, the internal clock is selected. When the internal clock is selected, its frequency is set by the shift clock frequency register (RF: 39H).

When the SIO0MS flag is set to 0 in two-wire bus mode, slave operation is specified. When the SIO0MS flag is set to 1 in two-wire bus mode, master operation is specified.

Table 16-6 SIO0MS Flag Functions

SIO0MS	Function
0	Two-wire bus mode : Slave operation Serial I/O mode : External clock operation
1	Two-wire bus mode : Master operation Serial I/O mode : Internal clock operation

16.1.4 SIO0TX

If the SIO0TX flag is set to 0 in two-wire bus mode, reception mode is specified. If the SIO0TX flag is set to 1 in two-wire bus mode, transmission mode is specified.

If the SIO0TX flag becomes 0 upon specifying CH0 serial I/O mode, SI mode (the SDA pin is in input mode) is specified. If the SIO0TX flag becomes 1 upon specifying CH0 serial I/O mode, SO mode (the SDA pin is in output mode) is specified.

When the CH1 serial I/O mode is specified, the SIO0TX flag specifies whether the SO pin is to be used as a serial interface. When the SIO0TX flag is set to 1, the SO pin is used as an SO pin. When the SIO0TX flag is set to 0, the SO pin is used as a general-purpose port.

Table 16-7 SIO0TX Flag Functions

SIO0TX	Function
0	Two-wire bus mode : RX (reception) mode CH0 serial I/O mode : SI mode CH1 serial I/O mode : $P0A_3$ is used as a general-purpose port
1	Two-wire bus mode : TX (transmission) mode CH0 serial I/O mode : SO mode CH1 serial I/O mode : $P0A_3$ is used as an SO pin

16.2 CLOCK COUNTER

The clock counter is a wrap around counter that counts the clock of the shift clock pin ($P0A_1$/SCL pin for CH0, $P0A_2$/$\overline{SCK}$ pin for CH1) of the currently selected serial interface. The clock counter counts the shift clock from 1 to 9 repeatedly. The initial value of the counter is 0. The counter is incremented by 1 each time the clock rising edge is detected. Once the counter has been incremented to 9, the counter is reset to 1, after which it is again incremented in the same way.

In the following cases, the clock counter is reset to 0.

(1) In two-wire bus mode

(a) At power-on reset
(b) When a STOP instruction is executed and the system is clock stopped
(c) When a start condition is detected
(d) When the serial interface operation mode is switched from two-wire bus mode to serial I/O mode

(2) In serial I/O mode

(a) At power-on reset
(b) When a STOP instruction is executed and the system clock is stopped
(c) When data is written into the wait register
(d) When the serial interface operation mode is switched from serial I/O mode to two-wire bus mode

Whether the contents of the clock counter became 8 or 9 can be tested in the software by the status register. A request to stop the clock in either transmission mode or reception mode in two-wire bus mode can be handled by the wait register.

16.3 STATUS REGISTER

The status register is a four-bit read-only register that retains the start and stop states in two-wire bus mode and the contents of the current clock counter.

Fig. 16-2 Configuration of Status Register

Bit position	b_3	b_2	b_1	b_0
Flag name	SIO0SF8	SIO0SF9	SBSTT	SBBSY

16.3.1 SBBSY (Serial Bus Busy) Flag

The SBBSY flag, mapped to b_0 (LSB) of the status register (RF: 28H), detects the busy signal in two-wire bus mode.

The SBBSY flag is valid only when two-wire bus mode is selected by the SB flag of the serial mode register. When the start condition is detected, the SBBSY flag is set to 1. When the stop condition is detected, the SBBSY flag is reset to 0.

When serial I/O mode is selected by setting the contents of the serial mode register, the SBBSY flag is reset to 0 and remains set to 0 until two-wire bus mode is selected.

This means that the SBBSY flag does not change in serial I/O mode.

When neither transmission nor reception is performed, testing of the SBBSY flag in two-wire bus mode enables the system to determine whether other devices are communicating.

16.3.2 SBSTT (Serial Bus Start Test) Flag

The SBSTT flag, mapped to b1 of the status register, detects the start condition in two-wire bus mode.

The SBSTT flag is valid only when two-wire bus mode is selected by setting the SB flag of the serial mode register. When the start condition is detected, the SBSTT flag is set to 1. When the contents of the clock counter become 9, the SBSTT flag is reset to 0.

16.3.3 SIO0SF9 (Serial I/O Shift 9 Clock) Flag

The SIO0SF9 flag, mapped to b_2 of the status register, is set to 1 when the contents of the clock counter become 9. When the contents of the clock counter become 0 or 1, the SIO0SF9 flag is reset to 0.

In master mode of two-wire bus mode, the contents of the flag that indicates whether the slave has returned an acknowledgement must be read after the SIO0SF9 flag becomes 1 but before the flag becomes 1 again.

The SIO0SF9 flag is not influenced by the contents of the serial mode register. This means that the SIO0SF9 flag is set when the contents of the clock counter become 9, even in serial I/O mode.

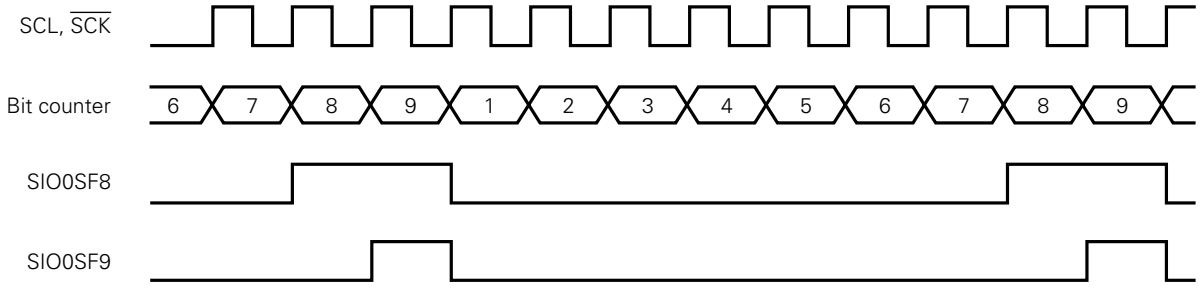
16.3.4 SIO0SF8 (Serial I/O Shift 8 Clock) Flag

The SIO0SF8 flag, mapped to b_3 of the status register, is set to 1 when the contents of the clock counter become 8. When the contents of the clock counter become 0 or 1, the SIO0SF8 flag is reset to 0.

An operation to read the presettable shift register must be performed while the SIO0SF8 flag is set to 1.

The SIO0SF8 flag is not influenced by the contents of the serial mode register.

Fig. 16-3 SIO0SF8 and SIO0SF9 Operations



16.4 WAIT REGISTER

The μPD17062 can set a state in which the serial interface hardware does not operate, even if a shift clock is input. This state is called wait mode and is set by the wait register.

The wait register consists of four bits; the SIO0WRQ0 flag, which specifies the timing to stop (wait) serial interface communication, SIO0WRQ1 flag, SIO0NWT flag, which indicates if whether the current state is waiting, and the SBACK flag, which indicates whether an acknowledgement is returned in two-wire bus mode.

The wait register, mapped to the register file, is manipulated by executing "PEEK" and "POKE" instructions via the window register.

All flags of the wait register are reset to 0 at power-on reset and when the system clock is stopped by executing a STOP instruction.

Fig. 16-4 Configuration of Wait Register

Bit position	b_3	b_2	b_1	b_0
Flag name	SBACK	SIO0NWT	SIO0WRQ1	SIO0WRQ0

16.4.1 SIO0WRQ1 and SIO0WRQ0 (Serial I/O Wait Request) Flag

The SIO0WRQ1 and SIO0WRQ0 flags reserve (specify) the timing at which the serial interface hardware is forced to wait. The μPD17062 expands the concept of waiting from slave operations in two-wire bus mode, to the transmission side in two-wire bus mode and internal clock operations in serial I/O mode.

During wait, the clock counter and the shift clock applied to the presettable shift register are disabled. This means that, during wait, the clock counter is not updated and the contents of the presettable shift register are not shifted, even if the level of the shift clock pin changes.

Table 16-8 Wait Timings

SIO0WRQ1	SIO0WRQ0	Wait mode	Two-wire bus mode	Serial I/O mode
0	0	No-wait	Does not wait.	Does not wait.
0	1	Data wait	Waits when the shift clock falls with the clock counter set to 8.	Waits with the shift clock in the high level state when the contents of the clock counter become 8.
1	0	Acknowledge wait	Waits when the shift clock falls with the clock counter set to 9.	Waits with the shift clock in the high level state when the contents of the clock counter become 9.
1	1	Address wait	Waits when the shift clock falls with the clock counter set to 8 after detection of the start condition.	Not to be set

(1) Slave operation wait in two-wire bus mode

When the timing specified by SIO0WRQ1 and SIO0WRQ0 is set, the SCL pin is switched to output mode and a low level signal is output.

If no-wait (SIO0WRQ1 = SIO0WRQ0 = 0) is specified, this operation is not performed.

Wait is released by writing 1 into the SIO0NWT flag of the wait register.

For example, if 1 is written into the SIO0NWT flag of the wait register while waiting with the data wait mode (SIO0WRQ1 = 0, SIO0WRQ0 = 1: Waits when the shift clock falls with the clock counter set to 8) specified, wait is released. When the shift clock falls with the clock counter set to 8 again, the slave operation waits again.

If communication has not started in slave operation, ordinary address wait mode (SIO0WRQ1 = SIO0WRQ0 = 1) is specified. In this wait mode, the slave operation waits when the shift clock falls, with the clock counter at set to 8, the first time after detection of the start condition. This means that, in this mode, the slave operation waits before the ninth clock (for acknowledgment of transmission of the slave address) rises. While the slave operation waits in this mode, the contents of the presettable shift register (PSR) are read to determine whether the address is mapped to the local station.

Testing the SIO0NWT flag enables the system to determine whether the slave operation is waiting.

(2) Master operation wait in two-wire bus mode

Master operation wait in two-wire bus mode incurs the interruption of transmission. In this mode, when the timing specified by SIO0WRQ1 and SIO0WRQ0 is set, the shift clock is fixed to the low level.

For example, testing the flag enables the system to determine whether the receiver has returned an acknowledgement while waiting with the acknowledge wait mode (SIO0WRQ1 = 1, SIO0WRQ0 = 0: Waits when the shift clock falls with the clock counter set to 9) specified. While waiting in this mode, data to be transmitted next can be set in the presettable shift register.

Wait is released by writing 1 into the SIO0NWT flag, in exactly the same way as for a slave operation. If no-wait is specified, this operation is not performed.

(3) Internal clock operation wait in serial I/O mode

This wait mode is almost the same as wait in two-wire bus mode. This wait also incurs the interruption of transmission. The only difference is that the master operation waits with the shift clock set to low level in two-wire bus mode while the internal clock operation waits with the shift clock set to the high level in serial I/O mode.

If serial I/O mode is specified, the clock counter is reset to 0 by performing a data write operation on the wait register. For this reason, if data wait mode is specified then acknowledge wait mode is respecified, the clock counter starts counting after being reset to 0 and the shift clock stops at the high level when the clock counter reaches 9.

This means that, in internal clock operation mode of serial I/O mode, writing data into the wait register resets the clock counter, after which transmission starts.

If no-wait is specified, the shift clock is output continuously.

(4) External clock operation wait in serial I/O mode

For external clock operation in serial I/O mode, update of the clock counter and shift of the presettable shift register are prohibited at the timing specified by SIO0WRQ1 and SIO0WRQ0. For example, if data wait mode is specified, the external clock waits when the clock falls with the clock counter set to 8. And, the clock counter is subsequently not updated by the shift clock input, nor does the presettable shift register shift data.

To enable data after waiting, 1 must be written into the SIO0NWT flag as usual. This means that the clock counter is reset to 0 by writing data into the wait register, after which wait is released.

16.4.2 SIO0NWT (Serial I/O No-Wait) Flag

Writing appropriate data into the SIO0NWT flag can both release wait and execute forced wait.

(1) Writing 0 into SIO0NWT

In this case, forced wait is executed. In other words, the clock being supplied to the clock counter and presettable shift register is disabled.

If the SIO0MS flag of the serial interface mode register is set to 1 at this time, shift clock operation stops in the current state.

(2) Writing 1 into SIO0NWT

In this case, wait is released. In other words, the clock being supplied to the clock counter and presettable shift register is enabled. If the SIO0MS flag of the serial interface mode register is set to 1 at this time, the shift clock operation resumes from the state existing immediately before waiting began.

16.4.3 SBACK (Serial Bus Acknowledge)

The operation of SBACK varies with the operation mode of the serial interface.

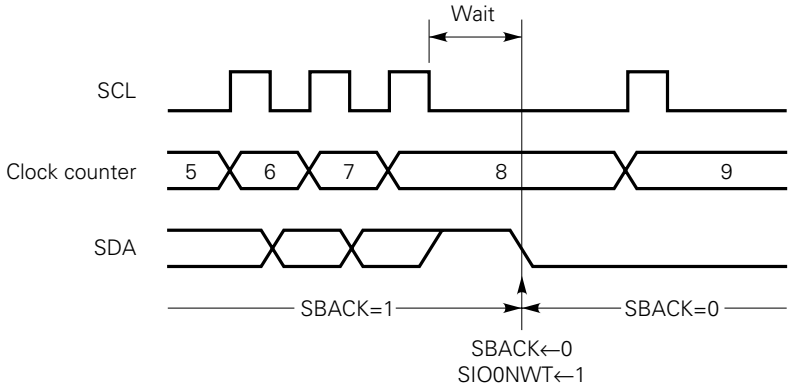
The following describes the operation of SBACK.

(1) For reception in two-wire bus mode (SIO0TX = 0)

In this case, the data set in the SBACK flag is automatically output to the SDA pin at the acknowledge output timing. The contents of the SBACK flag can be changed only by executing POKE instruction on the wait register. For this reason, to return acknowledgement continuously, simply write 0 into the SBACK flag. Then, 0 is automatically transmitted as an acknowledgement. This eliminates the need to manipulate the SBACK flag each time 1-byte data is received.

Data must be written to the SBACK flag after the 9th bit of the data has been set but before the 9th bit of the next data is set. Normally, wait is instigated at the falling edge of the 8th or 9th bit. Therefore, data should be written into the SBACK flag at this time.

Fig. 16-5 Timing of SBACK Rewriting during Wait



(2) For transmission in two-wire bus mode (SIO0TX = 1)

In this case, the contents of an acknowledgement received from the receiver side are set in the SBACK flag. This means that the acknowledge state of the receiver side can be determined simply by reading the contents of the SBACK flag.

This examining of the SBACK flag must be done after the 9th bit of 1-byte is set but before the 9th bit of the next data is set. Normally, waiting is instigated at the falling edge of the 9th bit. Therefore, the SBACK flag must be read at this time.

Data can be written into the SBACK flag by executing a POKE instruction, even during transmission.

(3) In serial I/O mode

In this case, the contents of the SBACK flag are not influenced by the shift clock. In other words, the SBACK flag is completely isolated from the serial interface. Hence, SBACK can be used as a 1-bit flag for data storage.

16.5 PRESETTABLE SHIFT REGISTER (PSR)

The presettable shift register is an 8-bit register. It outputs the contents of the most significant bit of the PSR to the serial data output pin ($P0A_0$/SDA pin for CH0, $P0A_3$/SO pin for CH1) synchronously with the falling edge of the clock signal on the shift clock pin ($P0A_1$/SCL pin for CH0, $P0A_2/\overline{SCK}$ pin for CH1) and reads the data of the serial data input pin ($P0A_0$/SDA pin for CH0, $P0B_0$/SI pin for CH1) into the least significant bit of the PSR synchronously with the rising edge of the clock.

In the wait state, the shift clock is not supplied to PSR. In other words, the PSR does not shift data in the wait state even if a clock (internal or external) is supplied to the shift clock pin (internally or externally) .

The operation of the PSR that is not in the wait state varies between two-wire bus mode and serial I/O mode.

Data is written to the PSR by the PUT instruction and read from the PSR by the GET instruction via the 8 low-order bits (data memory address: 0EH, 0FH) of DBF in data memory.

(1) PSR operation in two-wire bus mode

If two-wire bus mode is specified, the shift clock is supplied to the PSR only while the clock counter is set to 1 to 8. For example, to receive 9-bit data (8-bit data + 1-bit acknowledgement) in two-wire bus mode, the first 8 bits of data are read into the PSR. Then, the 9th bit is read into the SBACK flag of the wait register.

When the contents of the PSR are transmitted in two-wire bus mode, the contents of the PSR are output to the serial data pin while the clock counter is set to 1 to 8, and the contents of the SBACK flag are output while the clock counter is set to 9 (more precisely, between the fall of the 8th bit clock to the rise of the 9th bit clock).

The PSR operates as described above not only when using the hardware of the serial interface of the μPD17062 (also when using internal or external clock) but also when the clock is generated by the software with the port ($P0A_0$) also used as the shift clock pin set as an output port.

During transmission, data output to the SDA pin is again read into the PSR synchronously with the rise of the next shift clock. Therefore, in transmission also, once the shift clock has been output 8 times, data on the pin being transmitted is stored in the PSR. If no data conflict occurs during transmission, the data stored into the PSR will be exactly the same as that before the transmission. Hence, the data before transmission and PSR data after transmission can be compared to determine whether the data was transmitted normally.

The above explanation applies to a PSR that is not in the wait state, the PSR does not perform any shift operations.

(2) PSR operation in serial I/O mode

When serial I/O mode is specified, the shift clock supply to the PSR is not related to the contents of the clock counter. The PSR performs a shift operation according to the clock in the shift clock pin unless it is currently in the wait state.

The PSR does not perform any shift operations in the wait state. Hence, when the PSR is used merely for data storage, not as a serial interface, the PSR must be set to the wait state.

In serial I/O mode, data must be written to the PSR or read from the PSR when the shift clock is at the high level or in the wait state. If data is written or read at any other time, the PSR will not operate normally.

Normally, when the internal clock is used, wait should occur with the rise of the 8th bit clock, and the PSR should be manipulated during this wait state. When the external clock is used, the PSR should be manipulated while the high level of the shift clock is checked by the transmission side.

16.6 SERIAL INTERFACE INTERRUPT SOURCE REGISTER (SIO0IMD)

The interrupt source register (SIO0IMD) is a four-bit register that specifies when an interrupt is generated in the CPU during serial interface communication.

The SIO0IMD register is mapped to address 38H of the register file.

Fig. 16-6 shows the configuration of the SIO0IMD register. The register is not mapped to the two high-order bits of the SIO0IMD. If the two high-order bits of the SIO0IMD are read, 0 is read from each bit.

Fig. 16-6 Configuration of Serial Interface Interrupt Source Register (RF: 38H)

Bit position	b_3	b_2	b_1	b_0
Flag name	SIO0IMD3 (0)	SIO0IMD2 (0)	SIO0IMD1	SIO0IMD0

Table 16-9 Functions of Serial Interface Interrupt Source Register

SIO0IMD1	SIO0IMD0	Function
0	0	An interrupt request is generated when the 7th bit of the shift clock rises.
0	1	An interrupt request is generated when the 8th bit of the shift clock falls.
1	0	An interrupt request is generated at the rising edge of the 7th bit of the shift clock immediately after detection of the start condition.
1	1	An interrupt request is generated upon detection of the stop condition.

16.7 SHIFT CLOCK FREQUENCY REGISTER (SIO0CK)

The shift clock frequency register is a four-bit register for setting the frequency of the internal clock of the serial interface.

The shift clock frequency register is mapped to address 39H of the register file.

Fig. 16-7 shows the configuration of the shift clock frequency register. The register is not mapped to the two high-order bits of the shift clock frequency register. If the two high-order bits of the shift clock frequency register are read, 0 is read from each bit.

Fig. 16-7 Configuration of Shift Clock Frequency Register (RF: 39H)

Bit position	b_3	b_2	b_1	b_0
Flag name	SIO0CK3 (0)	SIO0CK2 (0)	SIO0CK1	SIO0CK0

Table 16-10 Internal Clock Frequencies of Serial Interface

SIO0CK1	SIO0CK0	Internal clock frequency
0	0	100 kHz
0	1	200 kHz
1	0	500 kHz
1	1	1 MHz

17. D/A CONVERTER

17.1 PWM PINS

The μPD17062 has 4 output pins for 6-bit PWM, which enables varying the duty cycle of the 15.625 kHz pulse signal in 64 steps. With this capability, attaching an external lowpass filter to the μPD17062 makes it function as a D/A converter. The PWM pins can also be used as 1-bit output ports.

When used as a D/A converter, the μPD17062 sets the D/A outputs in the output data latches, PWMRs. These latches, PWMR0, PWMR1, PWMR2, and PWMR3, are mapped at addresses 05H, 06H, 07H, and 08H, respectively. They can be read- and write-accessed via the DBF. Table 17-1 lists the correspondence between the PWMR addresses and the PWM pins.

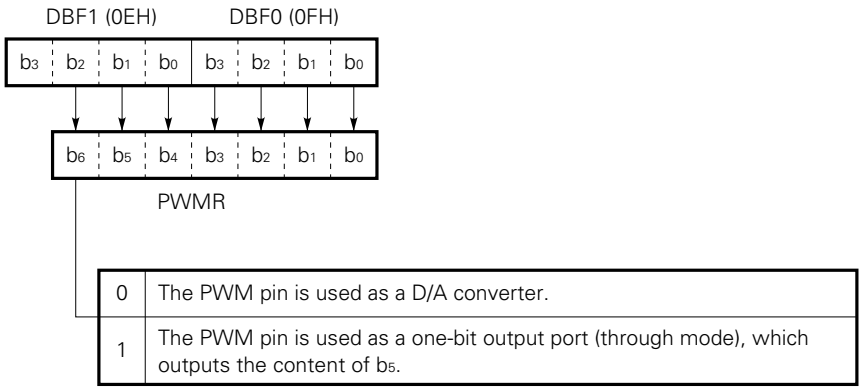
Table 17-1 PWMR Addresses and the Corresponding Pins

Peripheral equipment	Peripheral address	Corresponding pin
PWM0	05H	PWM_0
PWM1	06H	PWM_1
PWM2	07H	PWM_2
PWM3	08H	PWM_3

Each PWMR consists of 7 bits. Fig. 17-1 shows the configuration of the PWMR and its correspondence with the DBF. The highest bit of the PWMR specifies whether the PWM pin is to be used as a PWM output pin or an output port. The other six bits specify the output values of the PWM signal.

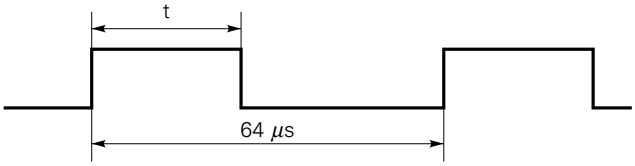
Fig. 17-2 shows the waveform output from the PWMR pin.

Fig. 17-1 PWMR Structure and the Corresponding DBF Bits



0	The PWM pin is used as a D/A converter.
1	The PWM pin is used as a one-bit output port (through mode), which outputs the content of b_5.

Fig. 17-2 Waveform Output from the PWM Pin



t = n + 0.75 (μs) (where n is a value specified in the PWMR)

18. PLL FREQUENCY SYNTHESIZER

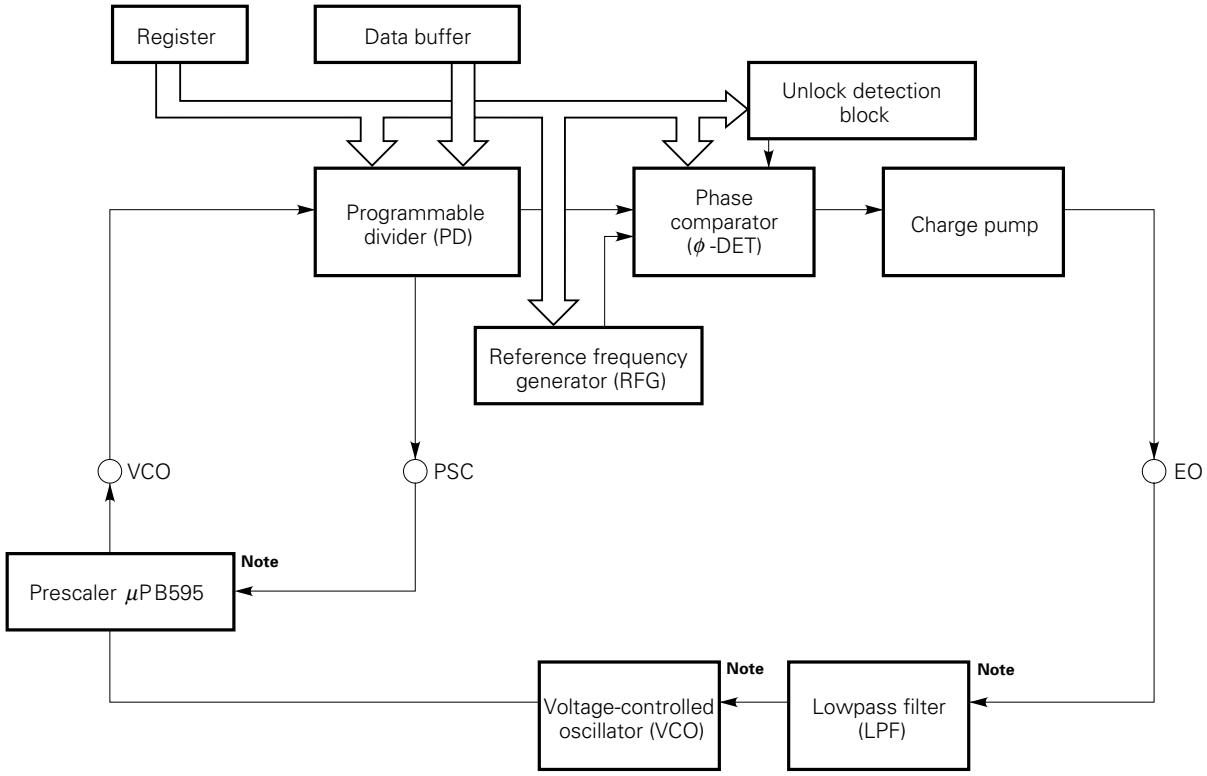
18.1 PLL FREQUENCY SYNTHESIZER CONFIGURATION

Fig. 18-1 is a block diagram of the PLL frequency synthesizer.

As shown in Fig. 18-1, the PLL frequency synthesizer consists of a programmable divider (PD), phase comparator (ϕ-DET), reference frequency generator (RFG), and charge pump. Strictly speaking, a PLL frequency synthesizer is configured by connecting these blocks with an external lowpass filter (LPF) and voltage-controlled oscillator (VCO).

See **Sections 18.3** to **18.5** for details of these blocks.

Fig. 18-1 PLL Frequency Synthesizer Block Diagram



Note External circuit

18.2 OVERVIEW OF EACH PLL FREQUENCY SYNTHESIZER BLOCK

The PLL frequency synthesizer receives an input signal at the VCO pin, divides its frequency in the programmable divider, and outputs the difference in phase between the divider output and the reference frequency from the EO pin.

The PLL frequency synthesizer works only when the CE pin is at a high level. It is disabled when the CE pin is at a low level. See **Section 18.6** for the disable mode of the PLL frequency synthesizer.

Items (1) to (4) briefly describe each block of the synthesizer.

(1) Programmable divider (PD)

The programmable divider divides the frequency of a signal input from the VCO pin. It uses NEC's proprietary pulse swallow method to divide a frequency. A division value is given through the data buffer (DBF).

See **Section 18.3**.

(2) Reference frequency generator (RFG)

The reference frequency generator generates a reference frequency that the phase comparator (ϕ-DET) uses for reference purposes.

A reference frequency can be selected using the PLL reference mode select register (at address 13H).

See **Section 18.4**.

(3) Phase comparator (ϕ-DET) and unlock detection block

The phase comparator compares the output signal of the programmable divider (PD) with a signal from the reference frequency generator (RFG) and outputs the phase difference between the signals. The phase comparator can also detect the PLL unlock state.

Detection of the PLL unlock state is controlled with the PLL unlock FF delay control register (at address 32H) and the PLL unlock FF judge register (at address 22H).

See **Section 18.5**.

(4) Charge pump

The charge pump directs the output signal of the phase comparator (ϕ-DET) to the EO pin as a high, low level, or floating output.

See **Section 18.5**.

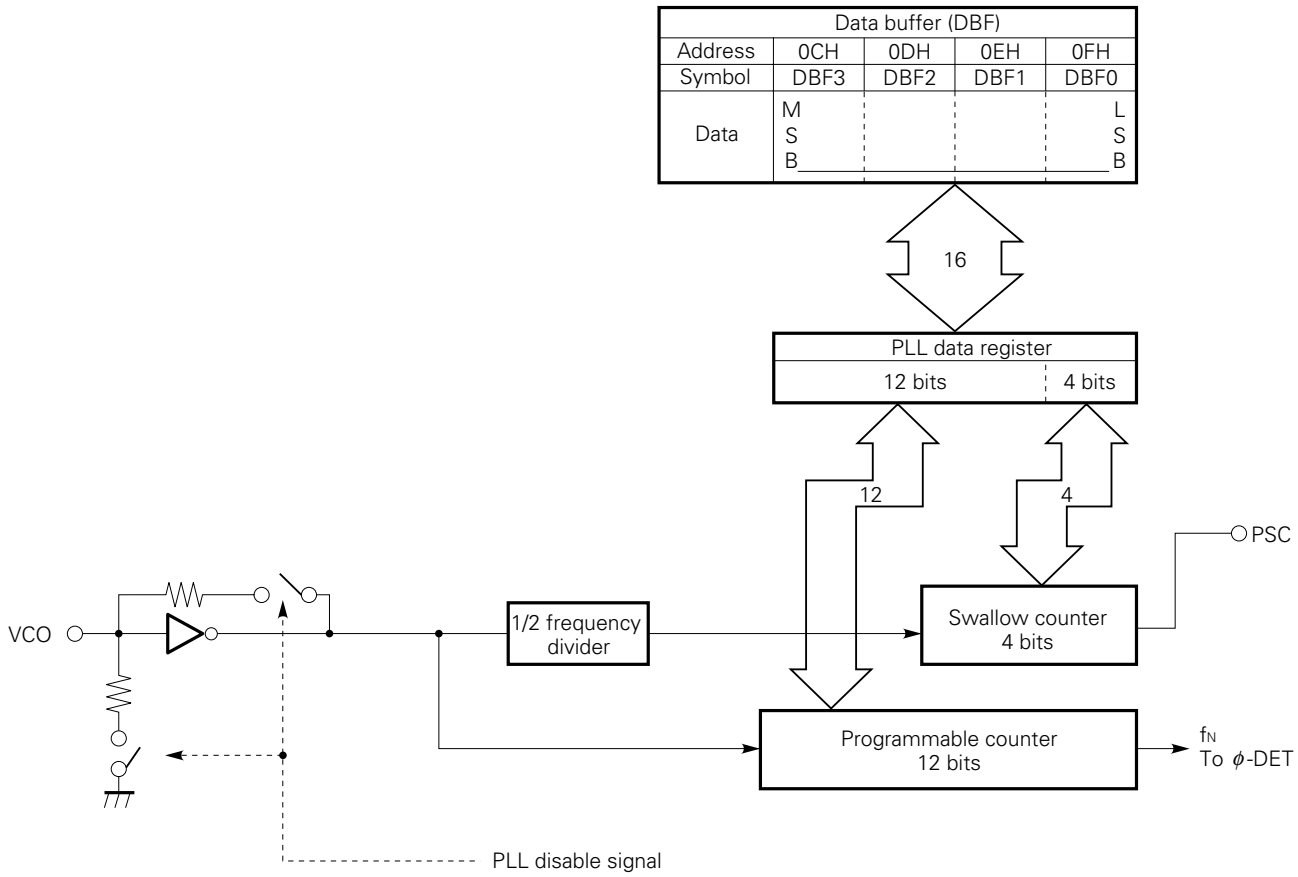
18.3 PROGRAMMABLE DIVIDER (PD) AND PLL MODE SELECT REGISTER

18.3.1 Programmable Divider Configuration

Fig. 18-2 shows the configuration of the programmable divider (PD).

As shown in Fig. 18-2, the programmable divider consists of a swallow counter and programmable counter.

Fig. 18-2 Programmable Divider Configuration



18.3.2 Programmable Divider (PD) and Data Buffer (DBF)

The programmable divider divides the frequency of an input signal at the VCO pin by the values specified in the swallow counter and programmable counter.

The swallow and programmable counters consist of a 4- and 12-bit binary downcounter, respectively.

The swallow and programmable counters are loaded with a division value by setting it in the PLL data register (PLLR, at address 41H) through the data buffer (DBF).

Writing to and reading from the PLL data register are performed with the "PUT PLLR,DBF" and "GET DBF,PLLR" instructions respectively.

A division value is called an N-value.

The following expression represents the frequency "f_N" of a signal generated in the programmable divider using the value N in the PLL data register (PLLR).

Pulse swallow method

$$f_N = \frac{f_{in}}{N}$$ (where N is 16 bits)

See **Section 18.7** for how to set the division value (N-value) for each frequency division method.

18.4 REFERENCE FREQUENCY GENERATOR (RFG)

18.4.1 Reference Frequency Generator (RFG) Configuration and Functions

Fig. 18-3 shows the configuration of the reference frequency generator.

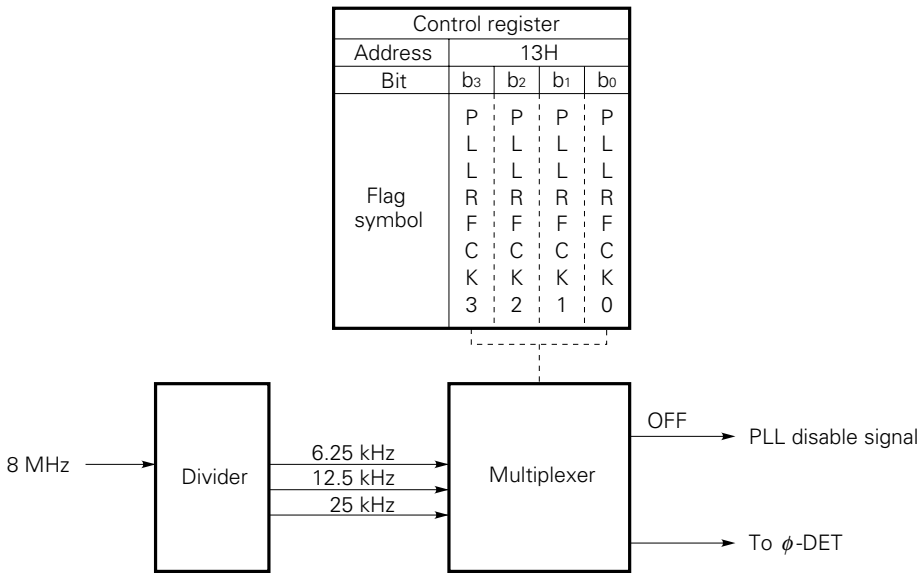
As shown in Fig. 18-3, the reference frequency generator divides the frequency of the clock oscillator (8 MHz) to generate the reference frequency "f_r" for the PLL frequency synthesizer.

The reference frequency f_r can be selected from 6.25, 12.5, and 25 kHz.

Selection of the reference frequency f_r is performed using the PLL reference mode select register (at address 13H).

Section 18.4.2 describes the configuration and functions of the PLL reference mode select register.

Fig. 18-3 Reference Frequency Generator (RFG) Configuration



18.4.2 PLL Reference Mode Select Register Configuration and Functions

Fig. 18-4 shows the configuration and functions of the PLL reference mode select register.

When the PLL reference mode select register selects the PLL disable mode, the VCO pin is pulled down internally, and the EO pin floats.

See **Section 18.6** for the PLL disable mode.

Fig. 18-4 PLL Reference Mode Select Register Configuration and Functions

Register	Flag symbol b_3	b_2	b_1	b_0	Address	Read/write
PLL reference mode select (PLRFMODE)	PLLRFCK3	PLLRFCK2	PLLRFCK1	PLLRFCK0	13H	R/W except PLLRFCK1, which is read-only

b_3	b_2	b_1	b_0	Specify the reference frequency f_r for the PLL frequency synthesizer.
0	0	1	0	6.25 kHz
0	0	1	1	12.5 kHz
0	1	1	0	25 kHz
1	1	1	1	PLL disable
0	1	1	1	Not to be set
1	0	1	0	
1	0	1	1	
1	1	1	0	

Fixed at 1

		b_3	b_2	b_1	b_0
Upon reset	Power-on	1	1	1	1
	Clock stop	1	1	1	1
	CE	Kept unchanged			

18.5 PHASE COMPARATOR (φ-DET), CHARGE PUMP, AND UNLOCK DETECTION BLOCK

18.5.1 Configuration of the Phase Comparator (φ-DET), Charge Pump, and Unlock Detection Block

Fig. 18-5 shows the configuration of the phase comparator (φ-DET), charge pump, and unlock detection block.

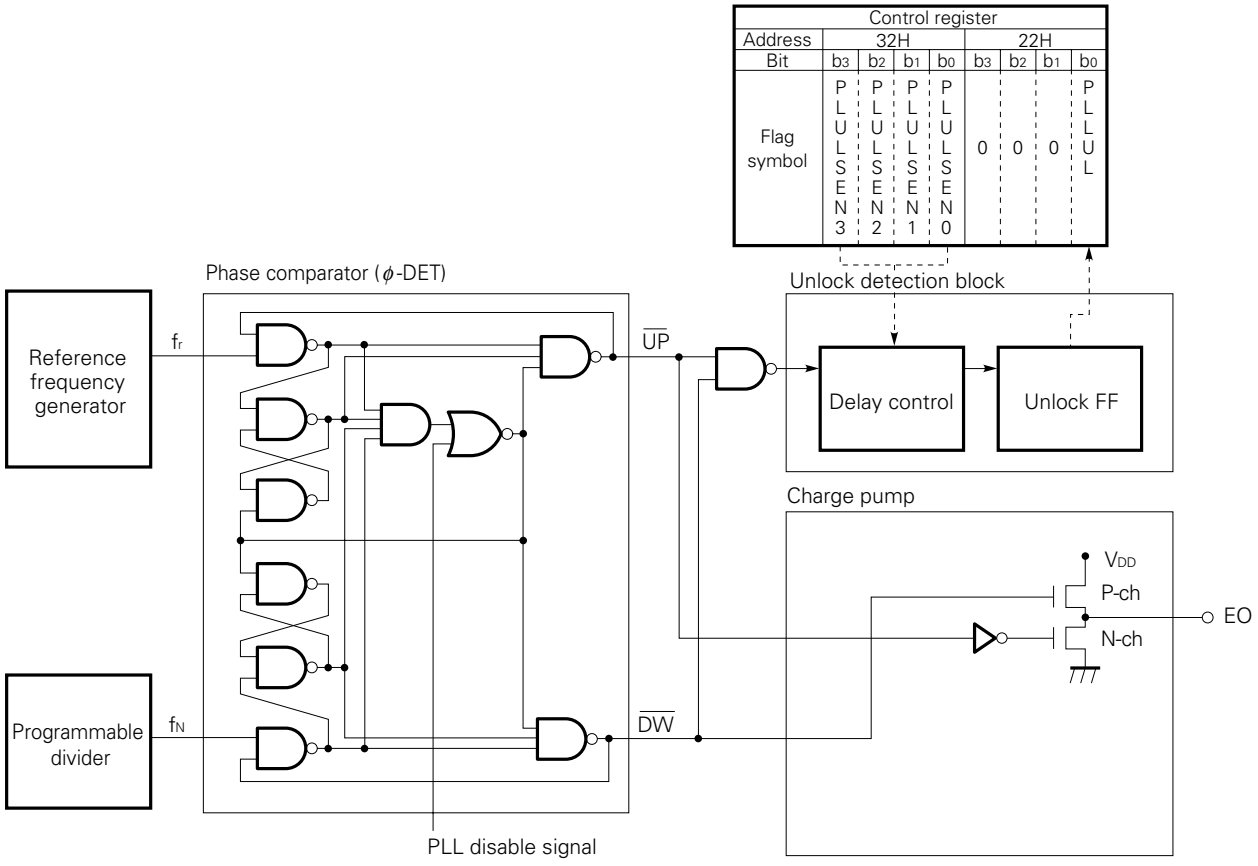
The phase comparator compares the phase of the output frequency "f_N" of the programmable divider (PD) with the reference frequency output "f_r" of the reference frequency generator, and outputs the up request signal ($\overline{UP}$) or down request signal ($\overline{DW}$).

The charge pump directs the output of the phase comparator to the error output pin (EO pin).

The unlock detection block consists of a delay control circuit and unlock flip-flop (FF). It detects the unlock state of the PLL frequency synthesizer.

Sections 18.5.2, **18.5.3**, and **18.5.4** explain the operation of the phase comparator, charge pump, and unlock detection block, respectively.

Fig. 18-5 Configuration of the Phase Comparator, Charge Pump, and Unlock Detection Block



18.5.2 Functions of the Phase Comparator (φ-DET)

As shown in Fig. 18-5, the phase comparator compares the phase of the output frequency "f_N" of the programmable divider (PD) and the phase of the reference frequency "f_r", and outputs the up request signal ($\overline{UP}$) or down request signal ($\overline{DW}$).

If the divider output frequency f_N is lower than the reference frequency f_r, the phase comparator outputs an up request. If f_N is higher than f_r, the phase comparator outputs a down request.

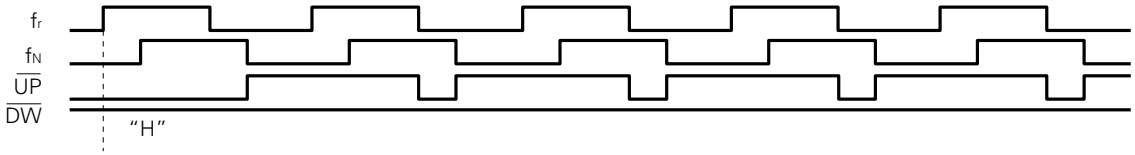
Fig. 18-6 shows the relationship among the reference frequency f_r, divider output frequency f_N, up request $\overline{UP}$, and down request $\overline{DW}$.

In the PLL disable mode, neither up nor down request is output.

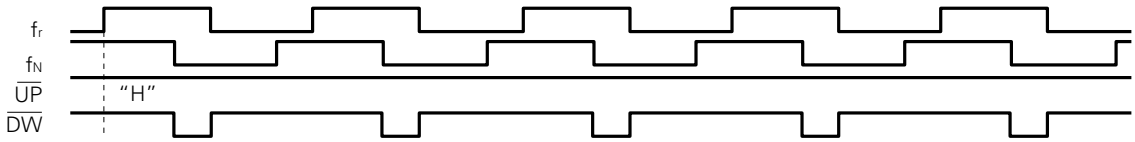
The up and down requests are directed to the charge pump and unlock detection block.

Fig. 18-6 Relationship among f_r, f_N, $\overline{UP}$, and $\overline{DW}$ Signals

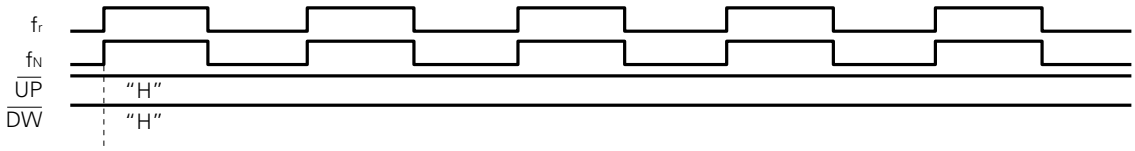
(1) When f_N is lagging behind f_r



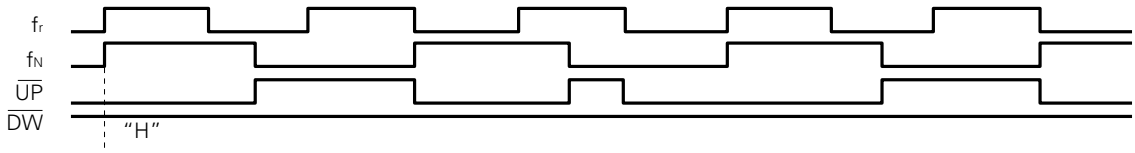
(2) When f_N is leading f_r



(3) When f_N is in phase with f_r



(4) When f_N is lower than f_r



18.5.3 Charge Pump

As shown in Fig. 18-5, the charge pump directs the up request signal ($\overline{UP}$) or down request signal ($\overline{DW}$) from the phase comparator (ϕ-DET) to the error output pin (EO) pin.

The relationships among the output at the error output pin, divider output frequency f_N, and reference frequency f_r are as follows:

Reference frequency f_r > divider output frequency f_N: Low level output
Reference frequency f_r < divider output frequency f_N: High level output
Reference frequency f_r = divider output frequency f_N: Floating

18.5.4 Unlock Detection Block

As shown in Fig. 18-5, the unlock detection block detects the unlock state of the PLL frequency synthesizer according to the up request signal ($\overline{UP}$) or down request signal ($\overline{DW}$) from the phase comparator (ϕ-DET).

Either the up or down request signal is low in the unlock state. So the unlock detection block detects this low signal as unlock state. When the unlock state is detected, the unlock flip-flop (FF) is set (1).

The state of the unlock FF is detected using the PLL unlock FF judge register (at address 22H).

The unlock FF is set at intervals of the then selected reference frequency f_r.

The unlock FF is reset when the PLL unlock FF judge register is read-accessed with a PEEK instruction.

The unlock FF must be checked at intervals greater than the period ($1/f_r$) of the reference frequency f_r.

The unlock delay control circuit controls whether to set the unlock FF, by delaying the up and down request signals output from the phase comparator.

If the delay becomes large, the unlock FF will not be set even if the phase difference between the divider output frequency f_N and reference frequency fr is large.

The delay is specified in the unlock delay control circuit using the PLL unlock FF delay control register (at address 32H).

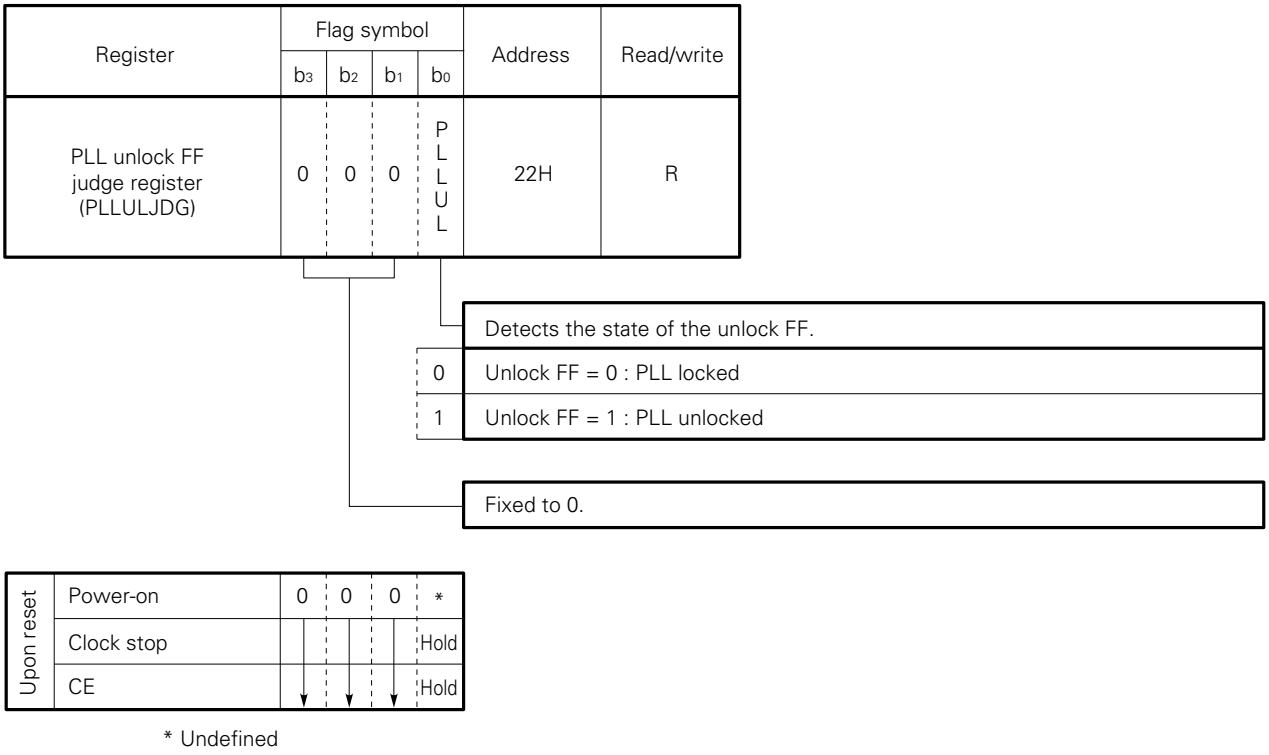
The following paragraphs describe the configuration and functions of the PLL unlock FF judge register and PLL unlock FF delay control register.

(1) PLL unlock FF judge register (PLLULJDG)

This register is a read-only register. It is reset when its content is read into a window register (WR) with a PEEK instruction.

Because the unlock FF is set at intervals of the period ($1/f_r$) of the reference frequency f_r, the content of this register must be read into the window register at intervals larger than the period of the reference frequency.

Fig. 18-7 Configuration and Functions of the PLL Unlock FF Judge Register (PLLULJDG)



Remark The PLLULJDG is reset when it is read-accessed with a PEEK instruction.

(2) PLL unlock FF delay control register (PLULSEN)

When the unlock FF disable mode is selected, the unlock FF remains set. So, note that if the PLL unlock FF judge register checks the unlock FF in the unlock FF disable mode, it always appears to be unlocked (PLLUL flag = 1).

Fig. 18-8 Configuration and Functions of the PLL Unlock FF Delay Control Register (PLULSEN)

Register	Flag symbol b_3	b_2	b_1	b_0	Address	Read/write
PLL unlock FF delay control (PLULSEN)	PLULSEN3	PLULSEN2	PLULSEN1	PLULSEN0	32H	R/W

Sets the delay time between the reference (f_r) and division frequency (f_N) signals, which is necessary to set the unlock FF.

0	0	1.25-1.5 μs or more
0	1	3.5-3.75 μs or more
1	0	0.25-0.5 μs or more
1	1	Unlock FF disabled (Always to be set)

Fixed to 0.

Upon reset					
	Power-on	0	0	0	0
	Clock stop			0	0
	CE			Hold	Hold

18.6 PLL DISABLE MODE

The PLL frequency synthesizer is disabled when the CE pin is at a low level. It is also disabled when the PLL reference mode select register (PLRFMODE, at address 13H) selects the PLL disable mode.

Table 18-1 summarizes how each block operates during the PLL disable mode.

Because the PLL reference mode select register is not initialized at a CE reset (its previous state is preserved), it returns to the previous state after the CE pin goes low (selecting the PLL disable mode) then back to a high.

If it is necessary to select the PLL disable mode at a CE reset, the PLL reference mode select register should be initialized by program.

The PLL frequency synthesizer is disabled at a power-on reset.

Table 18-1 Operation of Each Block During the PLL Disable Mode

Block	CE pin = low or PLRFMODE = 1111B
VCO pin	Pulled down internally
Programmable counter	Frequency division disabled
Reference frequency generator	Output disabled
Phase comparator	Output disabled
Charge pump	Error output pin floating

18.7 SETTING DATA FOR THE PLL FREQUENCY SYNTHESIZER

The following data is necessary to control the PLL frequency synthesizer.

(1) Reference frequency : f_r
(2) Division value : N

The following paragraphs explain how to set the PLL data.

(1) Setting reference frequency f_r

The reference frequency is specified according to the PLL reference mode select register.

(2) Calculating division value N

The division value N is calculated as follows:

$$N = \frac{f_{UCO}}{P \times f_r}$$

where f_{UCO} : Frequency input to the VCO pin
f_r : Reference frequency
P : Prescaler frequency division ratio

(3) Example of setting the PLL data

The following example shows how to specify the data required to receive channel 02 of the West Europe TV system, assuming that the prescaler used here is the μPB595 and that the frequency division ratio P is 8.

Receive frequency : 48.25 MHz
Reference frequency : 6.25 kHz
Intermediate frequency : 38.9 MHz

The division value N is calculated as follows:

$$N = \frac{f_{UCO}}{P \times f_r} = \frac{48250 + 38900}{8 \times 6.25} = 1743 \text{ (decimal)}$$

$$= 06CFH \text{ (hexadecimal)}$$

The PLL data register (PLLR, at address 41H) and PLL reference mode select register (PLRFMOD, at address 13H) are set with data as shown below.

PLLR			
0000	0110	1100	1111
0	6	C	F

PLRFMODE
0010
6.25 kHz

19. A/D CONVERTER

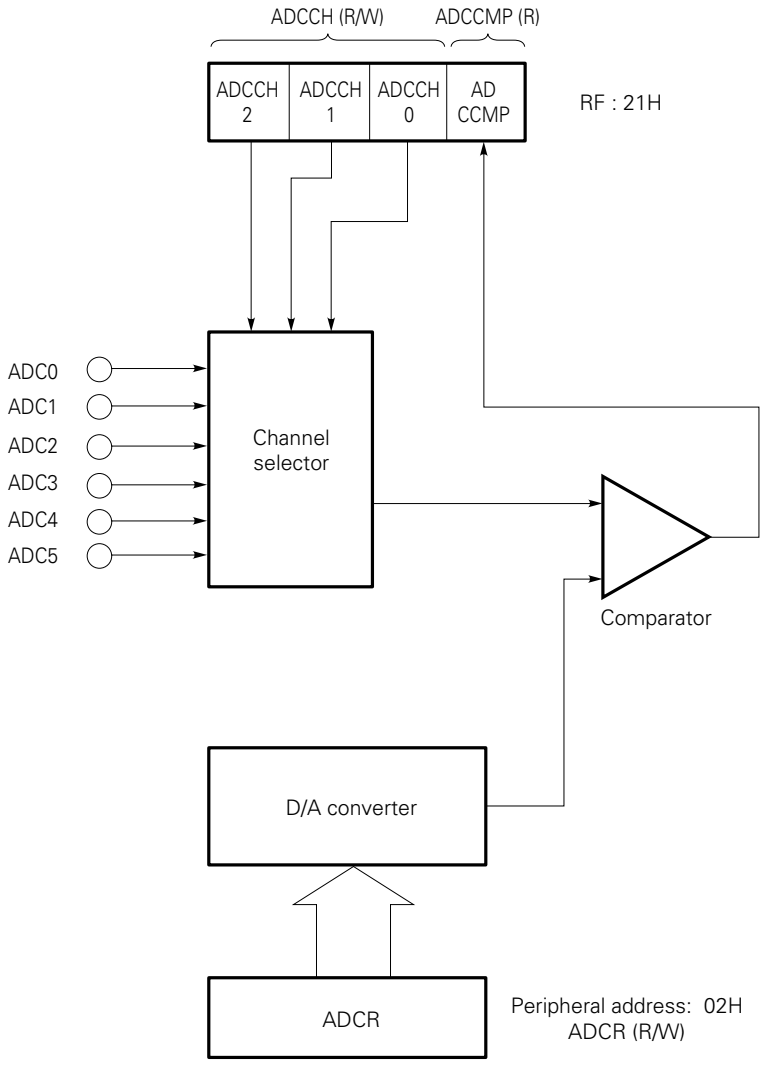
The μPD17062 contains a 4-bit program-controlled A/D converter that operates with a successive comparison method.

19.1 PRINCIPLE OF OPERATION

The A/D converter in the μPD17062 consists of a 4-bit resistor string-based D/A converter and comparator.

The D/A converter is set with data using a 4-bit register (ADCR) mapped at peripheral address 02H. The result of comparison is judged according to the ADCCMP flag in the register file.

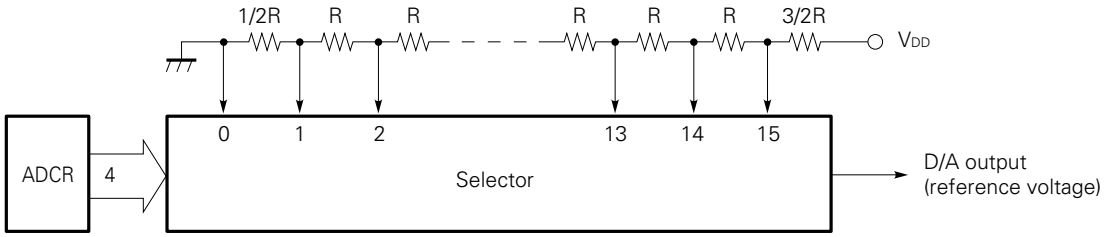
Fig. 19-1 A/D Converter Configuration



19.2 D/A CONVERTER CONFIGURATION

The D/A converter used in the A/D converter of the μPD17062 is a resistor string D/A converter consisting of 16 resistors connected in series between the V_{DD} and GND pins in which a voltage at each resistor connection point is selected. The configuration of the D/A converter is shown in Fig. 19-2.

Fig. 19-2 D/A Converter Configuration



With the configuration shown above, the D/A converter outputs a ground level when the ADCR is set with value 0000B. It also outputs $1/32 \times V_{DD}$ when the ADCR is set with 0001B. The following expression represents the reference voltage V_{REF} that the D/A converter outputs when the ADCR is set with value n (decimal).

$$V_{REF} = V_{DD} \times \frac{2n - 1}{32} \quad (\text{where } 15 \geq n \geq 1)$$

Table 19-1 D/A Converter Reference Voltage

Set data (ADCR)		Reference voltage (V_{REF})	
Hexadecimal	Binary	× V_{DD}	V_{DD} = 5 V
0	0000	0	0 [V]
1	0001	1/32	0.15625
2	0010	3/32	0.46875
3	0011	5/32	0.78125
4	0100	7/32	1.09375
5	0101	9/32	1.40625
6	0110	11/32	1.71875
7	0111	13/32	2.03125
8	1000	15/32	2.34375
9	1001	17/32	2.65625
A	1010	19/32	2.96875
B	1011	21/32	3.28125
C	1100	23/32	3.59375
D	1101	25/32	3.90625
E	1110	27/32	4.21875
F	1111	29/32	4.53125

19.3 REFERENCE VOLTAGE SETTING REGISTER (ADCR)

The ADCR is a 4-bit register to specify a reference voltage for the A/D converter. It is mapped at peripheral address 02H. Data is written to and read from the ADCR register through the data buffer using the "PUT" and "GET" instructions respectively. The data transfer between the ADCR and DBF is performed in 8-bit units although the ADCR is a 4-bit register. In other words, 8-bit data is transferred through the DBF1 (0EH) and DBF0 (0FH). To be specific, when the "GET DBF, ADCR" instruction is executed to read from the ADCR register, the content of the ADCR is sent to the DBF0 and 0000B is sent to the DBF1. Similarly, when the "PUT ADCR, DBF" instruction is executed, the data in the DBF0 is sent to the ADCR; the DBF1 may contain any data.

The ADCR is undefined at power-on. At a clock stop and CE reset, it retains the previous data.

19.4 COMPARISON REGISTER (ADCCMP)

The ADCCMP is a register that holds the output of a comparator which compares an input voltage at the ADC pin with the reference voltage (V_{REF}). It is mapped at bit b0 (LSB) of the register file at address 21H. The ADCCMP is a 1-bit read-only register; it cannot be written to. The PEEK instruction is executed to read data from the ADCCMP into a window register. At this point, the ADC pin select data is also read into the upper 3 bits of the window register.

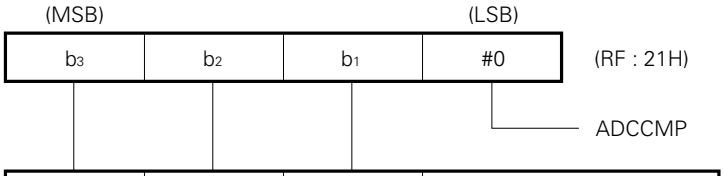
The window register will receive the ADCCMP content as follows:

ADCCMP = 0 when input voltage < reference voltage
ADCCMP = 1 when input voltage ≥ reference voltage

19.5 ADC PIN SELECT REGISTER (ADCCHn)

The ADCCHn register selects an A/D converter input pin. It is mapped at the upper 3 bits of the register file at address 21H. Table 19-2 lists the relationships between the ADCCHn and the actually selected pins.

Table 19-2 ADC Pin Selection



ADCCH2	ADCCH1	ADCCH0	Selected pin
0	0	0	ADC_0
0	0	1	$P1C_3/ADC_1$
0	1	0	$P0D_0/ADC_2$
0	1	1	$P0D_1/ADC_3$
1	0	0	$P0D_2/ADC_4$
1	0	1	$P0D_3/ADC_5$
1	1	0	No corresponding pin (do not set)
1	1	1	

When using $P1C_3/ADC_1$ as the A/D converter, specify the P1C as an input port.

The $P0D_0/ADC_2$ to $P0D_3/ADC_5$ pins are internally equipped with pull-down resistors, but the pull-down resistors are disconnected when they are selected as the A/D converter.

If P1C and P0D pins selected as the A/D converter are accessed as ports, "0" is read out.

19.6 EXAMPLE OF A/D CONVERSION PROGRAM

The following example shows an A/D conversion program based on the successive comparison method. The result of conversion is held in the DBF0.

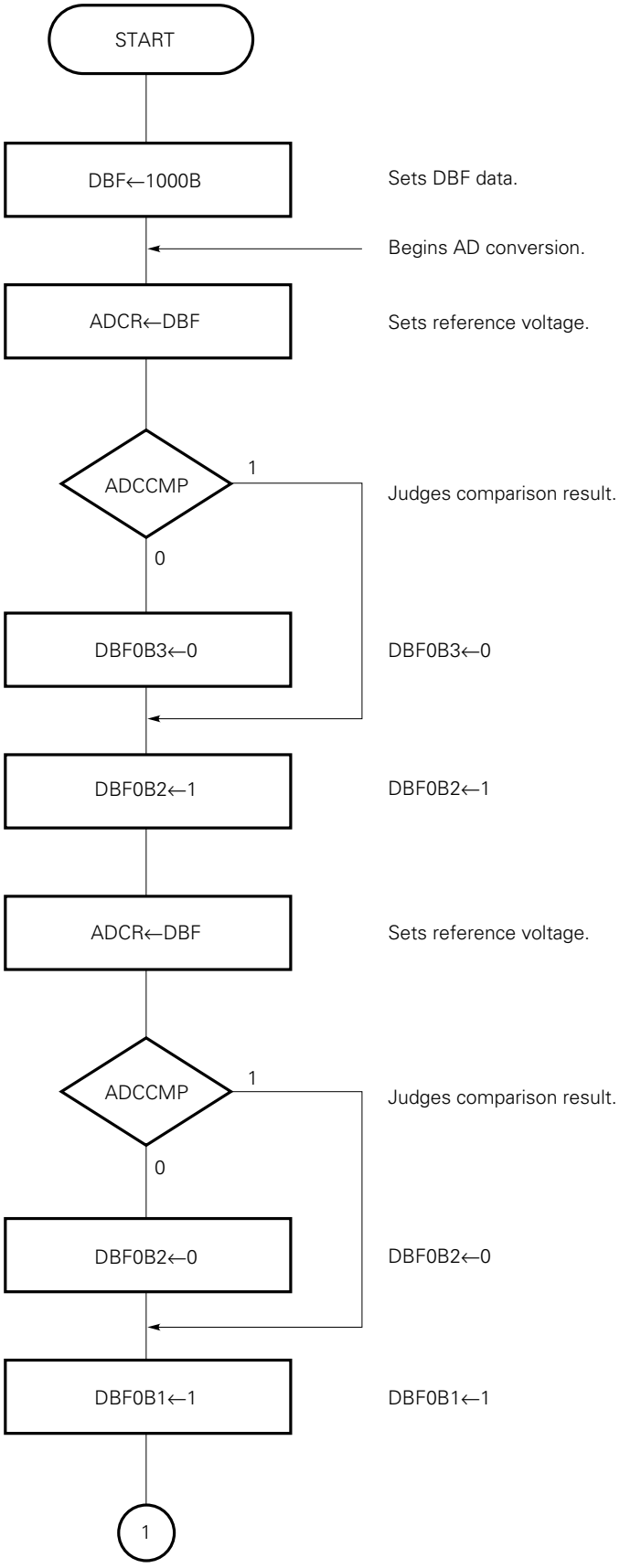
Sample program

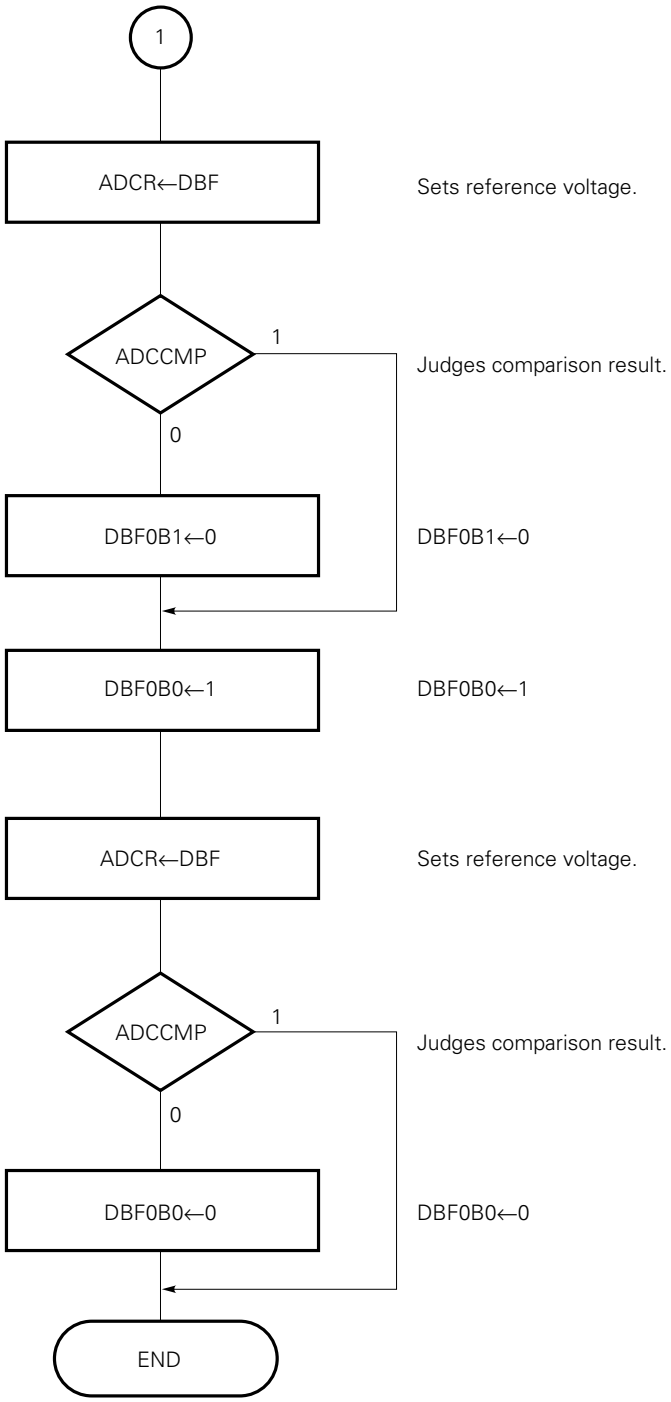
```
DBF0B3   FLG 0.0FH.3
DBF0B2   FLG 0.0FH.2
DBF0B1   FLG 0.0FH.1
DBF0B0   FLG 0.0FH.0

START:
        BANK0
        INITFLG   DBF0B3, NOT DBF0B2, NOT DBF0B1, NOT DBF0B0  ; Sets DBF data.
        PUT       ADCR, DBF                                   ; Sets reference voltage.
        SKT1      ADCCMP                                      ; Judges comparison result.
        CLR1      DBF0B3                                      ; DBF0B3 ← 0
        SET1      DBF0B2                                      ; DBF0B2 ← 1
        PUT       ADCR, DBF                                   ; Sets reference voltage.
        SKT1      ADCCMP                                      ; Judges comparison result.
        CLR1      DBF0B2                                      ; DBF0B2 ← 0
        SET1      DBF0B1                                      ; DBF0B1 ← 1
        PUT       ADCR, DBF                                   ; Sets reference voltage.
        SKT1      ADCCMP                                      ; Judges comparison result.
        CLR1      DBF0B1                                      ; DBF0B1 ← 0
        SET1      DBF0B0                                      ; DBF0B0 ← 1
        PUT       ADCR, DBF                                   ; Sets reference voltage.
        SKT1      ADCCMP                                      ; Judges comparison result.
        CLR1      DBF0B0                                      ; DBF0B0 ← 0
END:
```

Number of steps in the conversion loop : 17
Conversion time : 34 μs (not in DMA mode)
Conversion time : 204 μs (in DMA mode)

Flowchart


START
DBF←1000B
Sets DBF data.
Begins AD conversion.
ADCR←DBF
Sets reference voltage.
ADCCMP
1
Judges comparison result.
0
DBF0B3←0
DBF0B3←0
DBF0B2←1
DBF0B2←1
ADCR←DBF
Sets reference voltage.
ADCCMP
1
Judges comparison result.
0
DBF0B2←0
DBF0B2←0
DBF0B1←1
DBF0B1←1
1


1
ADCR←DBF
Sets reference voltage.
ADCCMP
1
Judges comparison result.
0
DBF0B1←0
DBF0B1←0
DBF0B0←1
DBF0B0←1
ADCR←DBF
Sets reference voltage.
ADCCMP
1
Judges comparison result.
0
DBF0B0←0
DBF0B0←0
END

20. IMAGE DISPLAY CONTROLLER

The image display controller (IDC) function indicates a channel number, volume of sound, time, and other information on a TV screen. The pattern of a display is user-programmable, and the display pattern definition is stored in the CROM area.

The pattern to be actually displayed is stored in VRAM, which is mapped at BANK1 and BANK2 in data memory.

20.1 SPECIFICATION OVERVIEW AND RESTRICTIONS

(1) Maximum number of characters that can be displayed on one screen: 97

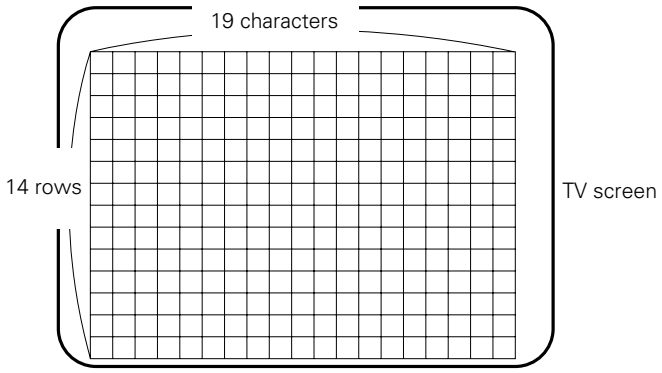
This specification applies when one control code is used per row. The maximum number of characters that can be displayed on one screen varies with the number of times control data is used.

Character size	Maximum number of display characters per row	Number of times that control data is used per row
Standard (minimum)	19	Up to 3
Double size	9	Up to 6
Triple size	5	Up to 5
Quadruple size	4	Up to 4

Using the control data three times per row amounts to that it is possible to specify the color three times per row.

(2) Variable display position range: 19 characters × 14 rows

The display area is defined for the TV screen as follows:

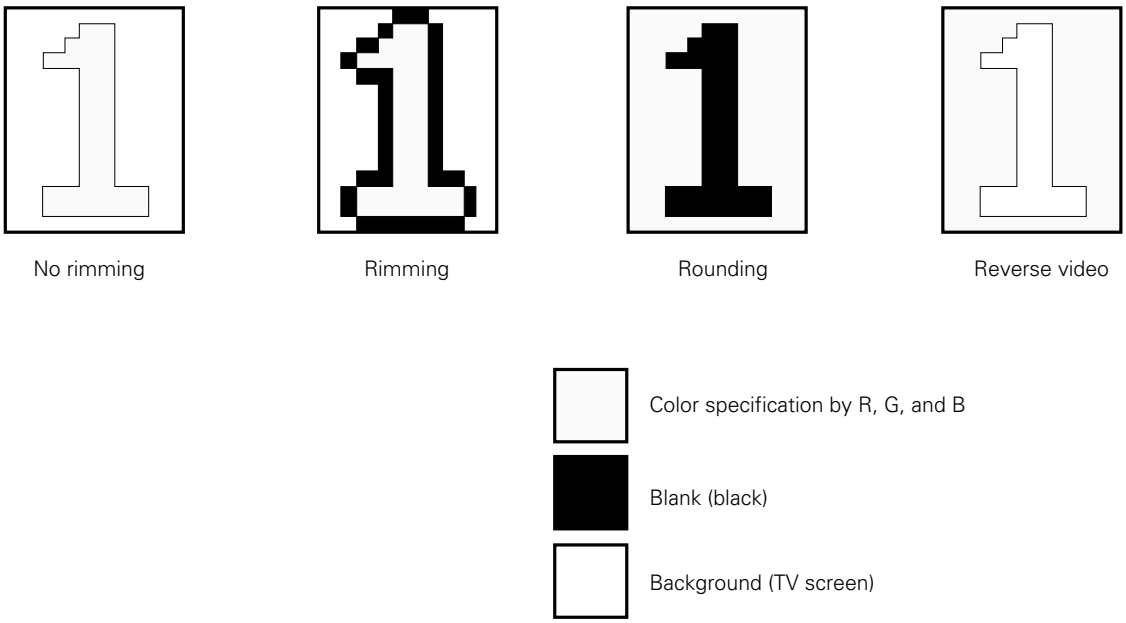


(3) Up to 8 colors (including black and white) can be specified for individual characters.

Independent specification of R, G, and B (using control data[Note])

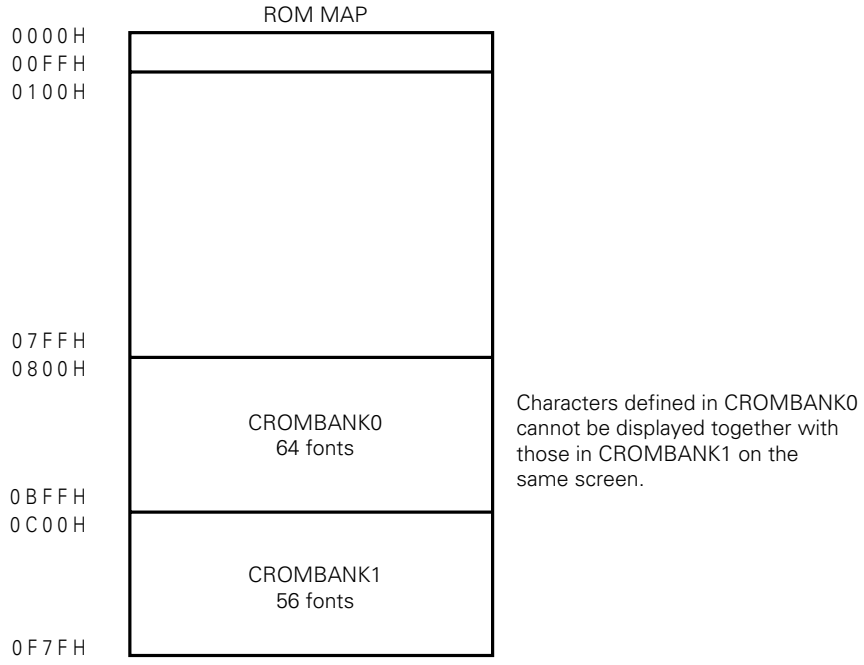
Note Up to three control data items can be specified per row.

(4) Rounding, rimming, and reverse video can be specified for individual characters.



(5) Number of fonts: 120 (user-programmable)

The number of fonts that can be displayed on one screen simultaneously is limited to within 64. Character pattern data is located in program memory (CROM), and up to 120 character patterns can be specified; however, up to 64 character patterns (in the same CROMBANK) can be displayed on one screen simultaneously.



(6) Up to 4 different character sizes, both vertical and horizontal, are available.
The same vertical character size is specified for all characters in a row, while the horizontal character size is specified for individual characters (according to the control data[Note 1]).

(7) The character bit configuration is 10 × 15 dots.
There is no gap between character positions.[Note 2]

(8) Character pattern data is allocated in program memory.
If there is only a small amount of character pattern data, the CROM area can also be used as a program area.

(9) Character data is allocated in the data memory space.
The character data can be transferred, read, and written in the same manner as ordinary data in data memory.

Notes 1. Up to three control data items can be specified per row.
2. Because there is no gap between character positions, kanji and other graphic images can be defined by combining two or more predefined characters.

20.2 DIRECT MEMORY ACCESS

The direct memory access (DMA) function transfers memory contents directly to peripheral equipment, without using the CPU.

In the μPD17062, the DMA mode is used to run the IDC.

The instruction cycle of the μPD17062 is 2 μs, but its apparent instruction cycle becomes 12 μs during the DMA mode. This does not mean that the actual instruction cycle becomes 12 μs, but means that data transfer for the IDC takes 10 μs (5 instruction cycles) and execution of an instruction takes one instruction cycle as usual. During DMA mode, instructions are executed at every five instruction cycles.

For the above reason, execution of one instruction takes 12 μs apparently when the IDC is being used. In a program in which a problem may occur if 12 μs and 2 μs instruction cycles are mixed, the IDC must be kept at a stop and the DMA mode can be specified only for critical sections of the program. In this case, during five instruction cycles for IDC data transfer, only the clock operates, and the μPD17062 does nothing.

During the DMA mode, the ROM address for five instructions out of the six is not pointed to by the program counter. Instead, it is pointed to by the CROM address pointer, and the RAM address is pointed to by the VRAM address pointer.

The DMA mode is controlled using the IDCDMAEN flag.

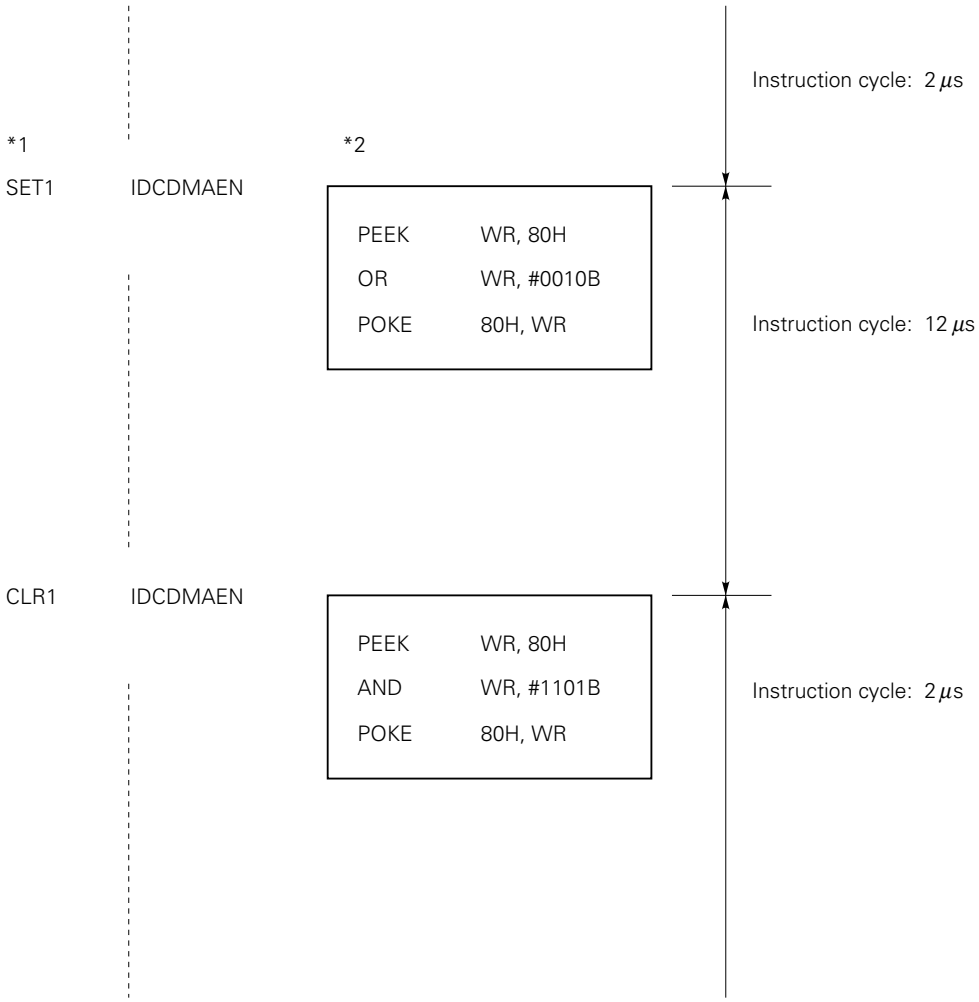
The IDCDMAEN flag is mapped at the register file. It is a one-bit flag that can be both read- and write-accessed. When this flag is set, a DMA request is accepted to begin the DMA mode in preference to any other interrupt request. When the IDCDMAEN flag is reset, no DMA request is accepted. If it is reset during the DMA mode, the DMA mode is terminated upon completion of the instruction that resets the flag.

Table 20-1 IDCDMAEN Flag

b_3	b_2	b_1	b_0	
0	0	IDCDMAEN	0	(RF 00H)

0	Does not use the DMA mode (instruction cycle: 2 μs)
0	Uses the DMA mode (instruction cycle: 12 μs).

Sample program



Remark The "SET1" or "CLR1" is not included in the μPD17062 instruction set. They are a built-in macro instruction of the 17K series assembler. They set or reset a one-bit flag. If they are written in a source program as shown at *1, they are expanded during assembly as shown at *2.

20.3 IDC ENABLE FLAG

The IDCEN (IDC enable) flag is manipulated to start IDC operations (turn on the display). The flag is mapped at the lowest bit (#0) of the register file at 31H.

Table 20-2 IDCEN Flag

b_3	b_2	b_1	b_0	
0	0	0	IDCEN	(RF 31H)

0	Turns off the display.
1	Turns on the display.

(1) Cautions in turning on the display

(a) The IDCEN flag must be set to 1 (begin displaying), when the vertical sync signal ($\overline{\text{Vsync}}$) is high (vertical flyback time: $\overline{\text{Vsync}}$ = low level) after the IDCDMAEN flag (RF, at 00H, #1) is turned on.

(b) Do not write data to VRAM, when the IDCEN flag is 1 (display turned on).

Sample program

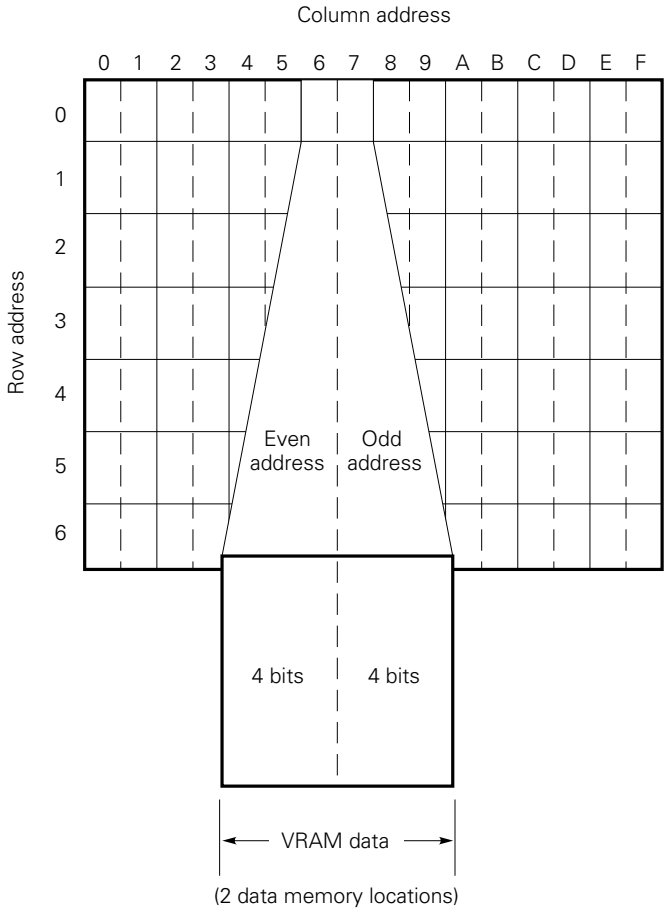
```
        SET1    IDCDMAEN        ; Sets the DMA mode.
        :
        CLR1    IDCEN           ; If the display is on when VRAM data is to be specified,
                                ; reset the IDCEN (turn off the display).
        :
        |  Sets data in VRAM.  |  ; Sets VRAM data.
        :
LOOP
        SKF1    INTVSYN         ; Makes sure Vsync = low level, and sets the IDCEN.
        BR      LOOP
        SET1    IDCEN           ; Turns on the display.
        :
```

20.4 VRAM

VRAM is the memory that holds data used to select a picture pattern that the IDC displays on a screen such as a TV screen. In the μPD17062, the VRAM data is allocated at BANK1 and BANK2 in data memory. One VRAM data item (8 bits) is held at two adjoining addresses (even and odd address).

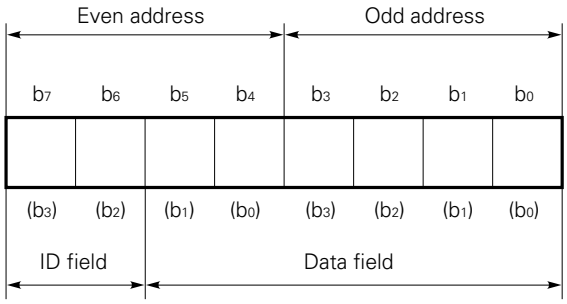
BANK1 and BANK2 are each mapped at 112 nibbles of data memory (total of 224 nibbles, or 224 × 4 bits). That is, up to 112 VRAM data items can be specified.

Fig. 20-1 VRAM Configuration



VRAM data consists of 8 bits. Of the 8 bits, the upper 2 bits are the ID field. The ID field indicates the type of VRAM data. The lower 6 bits are the data field. The data field contains the display data or control data.

Fig. 20-2 VRAM Data Configuration



20.4.1 ID Field

The ID field indicates the type of data in the data field.

The data field can hold the following three types of data.

(1) Character pattern select data
(2) Carriage return data
(3) Control data select data

Table 20-3 ID Field

ID field		Type of data in the data field
b_7	b_6	
0	0	Character pattern select data
0	1	Carriage return (return address) to BANK1 (from BANK1 to BANK1, and from BANK2 to BANK1)
1	0	Control data
1	1	Carriage return (return address) to BANK2 (from BANK2 to BANK2)

20.4.2 Character Pattern Select Data

The character pattern is the data of an image displayed on a screen such as a TV screen. It is allocated in the CROM area (at 0800H to 0F7FH) of program memory. The character pattern select data becomes part (b_9 to b_4) of a CROM address. In other words, the 6 bits of data in the data field indicate b_9 to b_4 of the CROM address.

CROM consists of BANK0 and BANK1. Note that even if the 6-bit VRAM data is the same, the CROM address varies according to the value of the CROMBNK (b_0 at 30H). If the data field contains a value of 0 (000000B), the CROM address is 080×H (10000000××××B) or 0C0×H (11000000××××B). Specifying the BANK of CROM selects 080×H or 0C0×H. Table 20-4 lists the CROM addresses that the VRAM character pattern select data actually points to.

Table 20-4 VRAM Data (Character Pattern Select Data) versus CROM Addresses

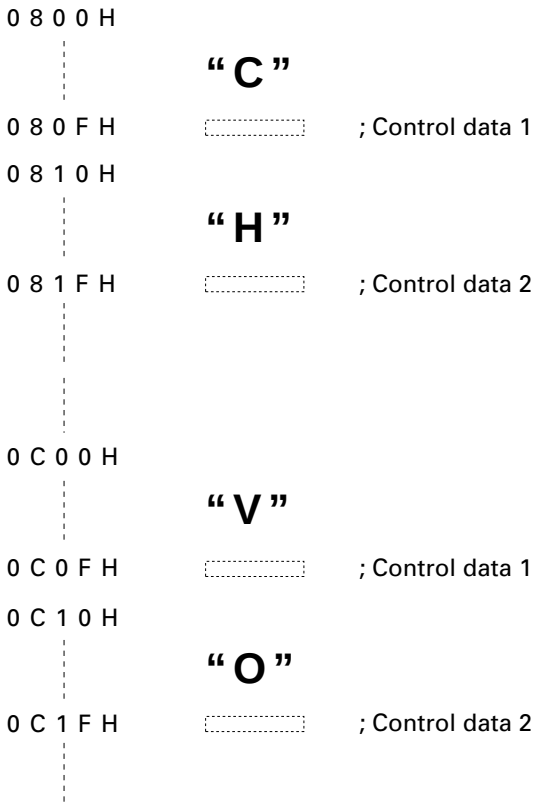
VRAM data (8 bits)	CROM address		VRAM data (8 bits)	CROM address	
	BANK0	BANK1		BANK0	BANK1
00H	0800H-080EH	0C00H-0C0EH	20H	0A00H-0A0EH	0E00H-0E0EH
01H	0810H-081EH	0C10H-0C1EH	21H	0A10H-0A1EH	0E10H-0E1EH
02H	0820H-082EH	0C20H-0C2EH	22H	0A20H-0A2EH	0E20H-0E2EH
03H	0830H-083EH	0C30H-0C3EH	23H	0A30H-0A3EH	0E30H-0E3EH
04H	0840H-084EH	0C40H-0C4EH	24H	0A40H-0A4EH	0E40H-0E4EH
05H	0850H-085EH	0C50H-0C5EH	25H	0A50H-0A5EH	OE50H-0E5EH
06H	0860H-086EH	0C60H-0C6EH	26H	0A60H-0A6EH	0E60H-0E6EH
07H	0870H-087EH	0C70H-0C7EH	27H	0A70H-0A7EH	0E70H-0E7EH
08H	0880H-088EH	0C80H-0C8EH	28H	0A80H-0A8EH	0E80H-0E8EH
09H	0890H-089EH	0C90H-0C9EH	29H	0A90H-0A9EH	0E90H-0E9EH
0AH	08A0H-08AEH	0CA0H-0CAEH	2AH	0AA0H-0AAEH	0EA0H-0EAEH
0BH	08B0H-08BEH	0CB0H-0CBEH	2BH	0AB0H-0ABEH	0EB0H-0EBEH
0CH	08C0H-08CEH	0CC0H-0CCEH	2CH	0AC0H-0ACEH	0EC0H-0ECEH
0DH	08D0H-08DEH	0CD0H-0CDEH	2DH	0AD0H-0ADEH	0ED0H-0EDEH
0EH	08E0H-08EEH	0CE0H-0CEEH	2EH	0AE0H-0AEEH	0EE0H-0EEEH
0FH	08F0H-08FEH	0CF0H-0CFEH	2FH	0AF0H-0AFEH	0EF0H-0EFEH
10H	0900H-090EH	0D00H-0D0EH	30H	0B00H-0B0EH	0F00H-0F0EH
11H	0910H-091EH	0D10H-0D1EH	31H	0B10H-0B1EH	0F10H-0F1EH
12H	0920H-092EH	0D20H-0D2EH	32H	0B20H-0B2EH	0F20H-0F2EH
13H	0930H-093EH	0D30H-0D3EH	33H	0B30H-0B3EH	0F30H-0F3EH
14H	0940H-094EH	0D40H-0D4EH	34H	0B40H-0B4EH	0F40H-0F4EH
15H	0950H-095EH	0D50H-0D5EH	35H	0B50H-0B5EH	0F50H-0F5EH
16H	0960H-096EH	0D60H-0D6EH	36H	0B60H-0B6EH	0F60H-0F6EH
17H	0970H-097EH	0D70H-0D7EH	37H	0B70H-0B7EH	0F70H-0F7EH
18H	0980H-098EH	0D80H-0D8EH	38H	0B80H-0B8EH	Not to be set
19H	0990H-099EH	0D90H-0D9EH	39H	0B90H-0B9EH	
1AH	09A0H-09AEH	0DA0H-0DAEH	3AH	0BA0H-0BAEH	
1BH	09B0H-09BEH	0DB0H-0DBEH	3BH	0BB0H-0BBEH	
1CH	09C0H-09CEH	0DC0H-0DCEH	3CH	0BC0H-0BCEH	
1DH	09D0H-09DEH	0DD0H-0DDEH	3DH	0BD0H-0BDEH	
1EH	09E0H-09EEH	0DE0H-0DEEH	3EH	0BE0H-0BEEH	
1FH	09F0H-09FEH	0DF0H-0DFEH	3FH	0BF0H-0BFEH	

Sample program

VRAM data

	0	1	2	3	4	5	6	7	8	9	A	B
0	8	0	0	0	0	1	4	0				
1												

CROM data



If the CROM data and VRAM data are specified as shown above, the display on the screen varies depending on the CROM bank.

The CROM bank is specified by CROMBNK (b_0 at 30H).

The following description applies to the above example.

(1) CROMBNK = 0

Display "CH" appears on the screen. The control data used in this case is "control data 1".

(2) CROMBNK = 1

Display "VO" appears on the screen. The control data used in this case is "control data 1".

20.4.3 Carriage Return Data

The term carriage return data refers to the data pointing to the address of the VRAM data that specifies the first character in a row on the screen.

The carriage return data specifies the end of a display row.

When carriage return data appears two times consecutively, it specifies the end of a screen.

There are two types of carriage return data; one type is a carriage return to BANK1, and the other is a carriage return to BANK2. The data in the ID field determines whether the data field indicates a carriage return to BANK1 or BANK2. If the ID field contains 01B, it indicates a carriage return to BANK1, and if the ID field contains 11B, it indicates a carriage return to BANK2.

The carriage return data consists of 6 bits, the upper 3 bits of which point to the row address of VRAM and the lower 3 bits of which point to the upper 3 bits of the VRAM column address. The lowest bit of the VRAM column address is fixed at 0. If the carriage return data is 010011B, therefore, the VRAM row address is 010B (2H), and the VRAM column address is 0110B (6H); namely, they mean the return data to 26H.

Fig. 20-3 Carriage Return Data Configuration

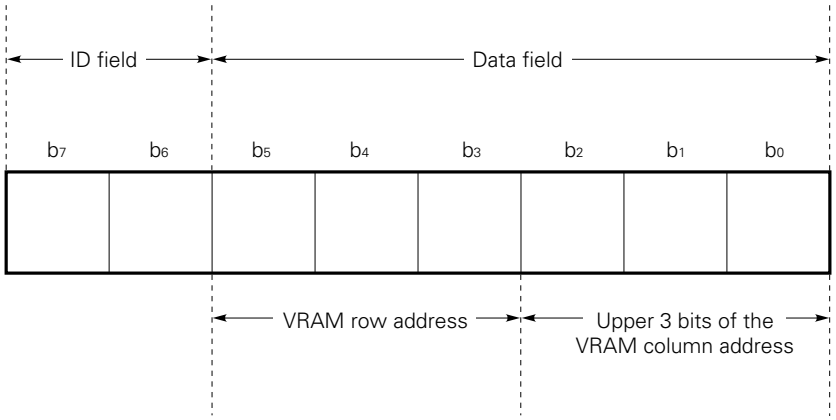


Fig. 20-4 Carriage Return Data (8 Bits Including the ID Field)

BANK1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	40		41		42		43		44		45		46		47	
1	48		49		4A		4B		4C		4D		4E		4F	
2	50		51		52		53		54		55		56		57	
3	58		59		5A		5B		5C		5D		5E		5F	
4	60		61		62		63		64		65		66		67	
5	68		69		6A		6B		6C		6D		6E		6F	
6	70		71		72		73		74		75		76		77	

BANK1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	C0		C1		C2		C3		C4		C5		C6		C7	
1	C8		C9		CA		CB		CC		CD		CE		CF	
2	D0		D1		D2		D3		D4		D5		D6		D7	
3	D8		D9		DA		DB		DC		DD		DE		DF	
4	E0		E1		E2		E3		E4		E5		E6		E7	
5	E8		E9		EA		EB		EC		ED		EE		EF	
6	F0		F1		F2		F3		F4		F5		F6		F7	

20.4.4 Control Data Select Data

The term control data refers to the data that specifies the character size, display position, and color of a character pattern on the screen. This data is held in CROM (at ××FH).

The control data select data is held in VRAM and selects control data in CROM.

The 6 bits of the data field correspond to b_9 to b_4 of the CROM address. Similarly to the pattern select data, the control data select data also requires that a CROM bank be specified. A CROM bank is specified using CROMBNK (b_0 at 30H) in the register file.

Fig. 20-5 Relationship between the Control Data and CROM Address

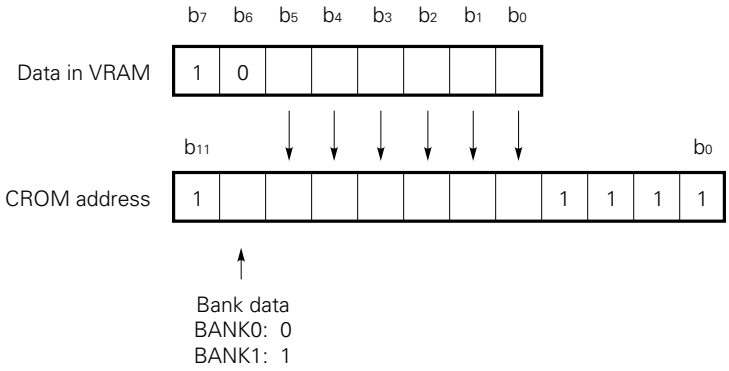


Table 20-5 VRAM Data (Control Data Select Data) versus CROM Addresses

VRAM data (8 bits)	CROM address		VRAM data (8 bits)	CROM address	
	BANK0	BANK1		BANK0	BANK1
80H	080FH	0C0FH	A0H	0A0FH	0E0FH
81H	081FH	0C1FH	A1H	0A1FH	0E1FH
82H	082FH	0C2FH	A2H	0A2FH	0E2FH
83H	083FH	0C3FH	A3H	0A3FH	0E3FH
84H	084FH	0C4FH	A4H	0A4FH	0E4FH
85H	085FH	0C5FH	A5H	0A5FH	0E5FH
86H	086FH	0C6FH	A6H	0A6FH	0E6FH
87H	087FH	0C7FH	A7H	0A7FH	0E7FH
88H	088FH	0C8FH	A8H	0A8FH	0E8FH
89H	089FH	0C9FH	A9H	0A9FH	0E9FH
8AH	08AFH	0CAFH	AAH	0AAFH	0EAFH
8BH	08BFH	0CBFH	ABH	0ABFH	0EBFH
8CH	08CFH	0CCFH	ACH	0ACFH	0ECFH
8DH	08DFH	0CDFH	ADH	0ADFH	0EDFH
8EH	08EFH	0CEFH	AEH	0AEFH	0EEFH
8FH	08FFH	0CFFH	AFH	0AFFH	0EFFH
90H	090FH	0D0FH	B0H	0B0FH	0F0FH
91H	091FH	0D1FH	B1H	0B1FH	0F1FH
92H	092FH	0D2FH	B2H	0B2FH	0F2FH
93H	093FH	0D3FH	B3H	0B3FH	0F3FH
94H	094FH	0D4FH	B4H	0B4FH	0F4FH
95H	095FH	0D5FH	B5H	0B5FH	0F5FH
96H	096FH	0D6FH	B6H	0B6FH	0F6FH
97H	097FH	0D7FH	B7H	0B7FH	0F7FH
98H	098FH	0D8FH	B8H	0B8FH	Not to be set
99H	099FH	0D9FH	B9H	0B9FH	
9AH	09AFH	0DAFH	BAH	0BAFH	
9BH	09BFH	0DBFH	BBH	0BBFH	
9CH	09CFH	0DCFH	BCH	0BCFH	
9DH	09DFH	0DDFH	BDH	0BDFH	
9EH	09EFH	0DEFH	BEH	0BEFH	
9FH	09FFH	0DFFH	BFH	0BFFH	

20.4.5 Cautions in Specifying VRAM Data

(1) Reset the IDCEN flag to 0 before specifying VRAM data.

(2) The VRAM data must begin at 00H in BANK1.

(3) Do not set VRAM data at 7×H in BANK1 or BANK2.

(4) Always set control data at the beginning of a screen. To prevent a program error, control data should be set at the beginning of each row. Otherwise, the previous control data remains effective.

(5) Data setting
   (a) The character pattern select data should be set at VRAM addresses sequentially starting at the lowest address so that the corresponding display begins at the upper left corner of the screen.
   (b) Control data can be used up to three times on each row.
   (c) The character pattern data that is modified by the control data begins after the control data select data. The horizontal start position data and vertical start position data correspond to only a character that follows immediately the control data select data; the other characters are output consecutively.
   (d) Always specify carriage return data at the end of a row.
   (e) Always specify two carriage return data items at the end of a screen.

20.5 CHARACTER ROM

The CROM (character ROM) consists of the IDC pattern data and control data. The CROM data shares the program memory with programs. The CROM area has a capacity of 2 Ksteps (1920 × 16 bits). An area not used as CROM is used as an ordinary program area.

The CROM area in ROM is at 0800H to 0F7FH. The CROM area is divided into BANK0 and BANK1. A concept of bank applies only to CROM. It does not apply to a program area. CROM BANK0 is 1 Kword at from 0800H to 0BFFH, and CROM BANK1 is 896 words at from 0C00H to 0F7FH.

The CROM bank is switched according to the CROMBNK flag (b_0 at 30H) in the register file.

Table 20-6 CROM Bank

CROMBNK flag	CROM bank	CROM address
0	BANK0	0800H-0BFFH
1	BANK1	0C00H-0F7FH

Remark The CROM bank should not be switched when the IDCEN flag is 1.

The register file at 30H can be read- and write-accessed, but the bits other than the CROMBNK flag (b_0) are always 0.

Because the CROM data is mapped in a program memory area, its size is 16 bits.

There are two types of CROM data.

(1) Character pattern data
(2) Control data

20.5.1 Character Pattern Data

The character pattern data is a character or graphic pattern. One character consists of 10 horizontal and 15 vertical dots. The corresponding character pattern data consists of 16 bits × 15 steps. The data of 10 horizontal dots corresponds to one CROM step. 15 steps at addresses ××0H to ××EH in CROM form one character pattern data item.

The structure of character pattern data varies according to whether the corresponding character has rimming.

Fig. 20-6 shows the configuration of the character pattern data.

The highest bit selects whether there is rimming. Set the bit to 0 when the character has no rimming, when the character has rimming, set the bit to 1.

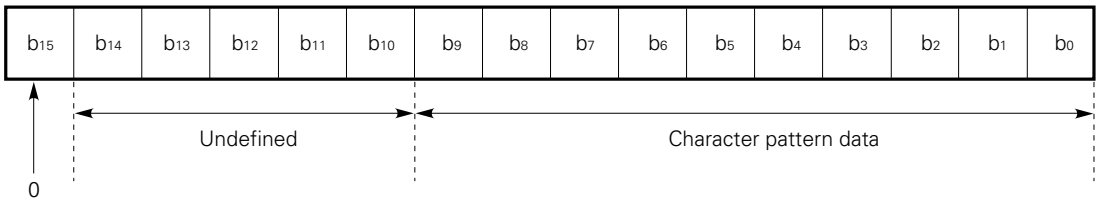
For a character with no rimming, the lower 10 bits indicate the dot image of the actually displayed character pattern. b_9 corresponds to the left section of the display, and b_0 to the right section. The bit that corresponds to a bright dot is 1, and the bit that corresponds to a dark dot is 0.

For a character with rimming, the character pattern data is 5 bits as shown in Fig. 20-6. At this point, two dots of the display pattern correspond to one character pattern data bit. This bit is combined with 10 rim data bits (rim data is specified in one-dot units) to form a character pattern for a character with rimming.

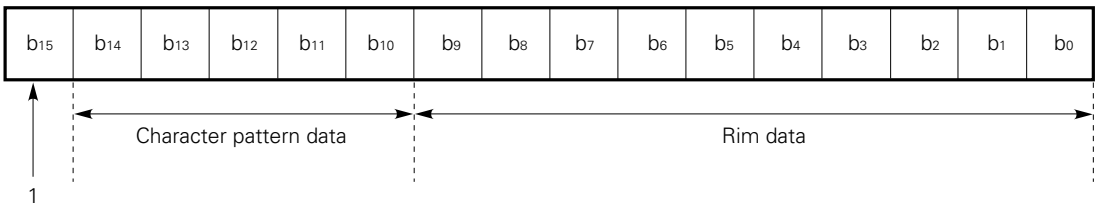
With the 17K series assembler, the DCP pseudo instruction can define a character pattern easily. Use of this instruction automatically generates the data shown in Fig. 20-6, regardless of whether there is rimming.

Fig. 20-6 Character Pattern Data Configuration

(a) Data for a character with no rimming



(b) Data for a character with rimming



If 2 is to be displayed, the character pattern is set as shown in Fig. 20-7. 0 and 1 in the pattern data correspond to □ and ■, respectively. In addition, the character size, position, and color are specified by the control data. Fig. 20-8 shows an example of the pattern of a character with rimming.

Fig. 20-7 Example of the Pattern of a Character with No Rimming

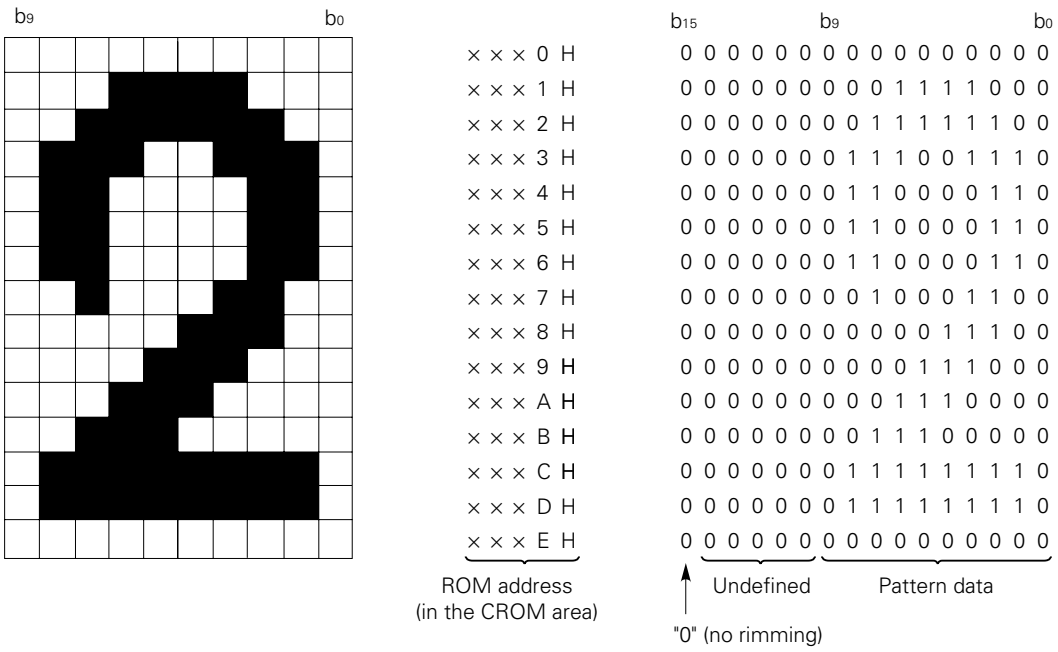
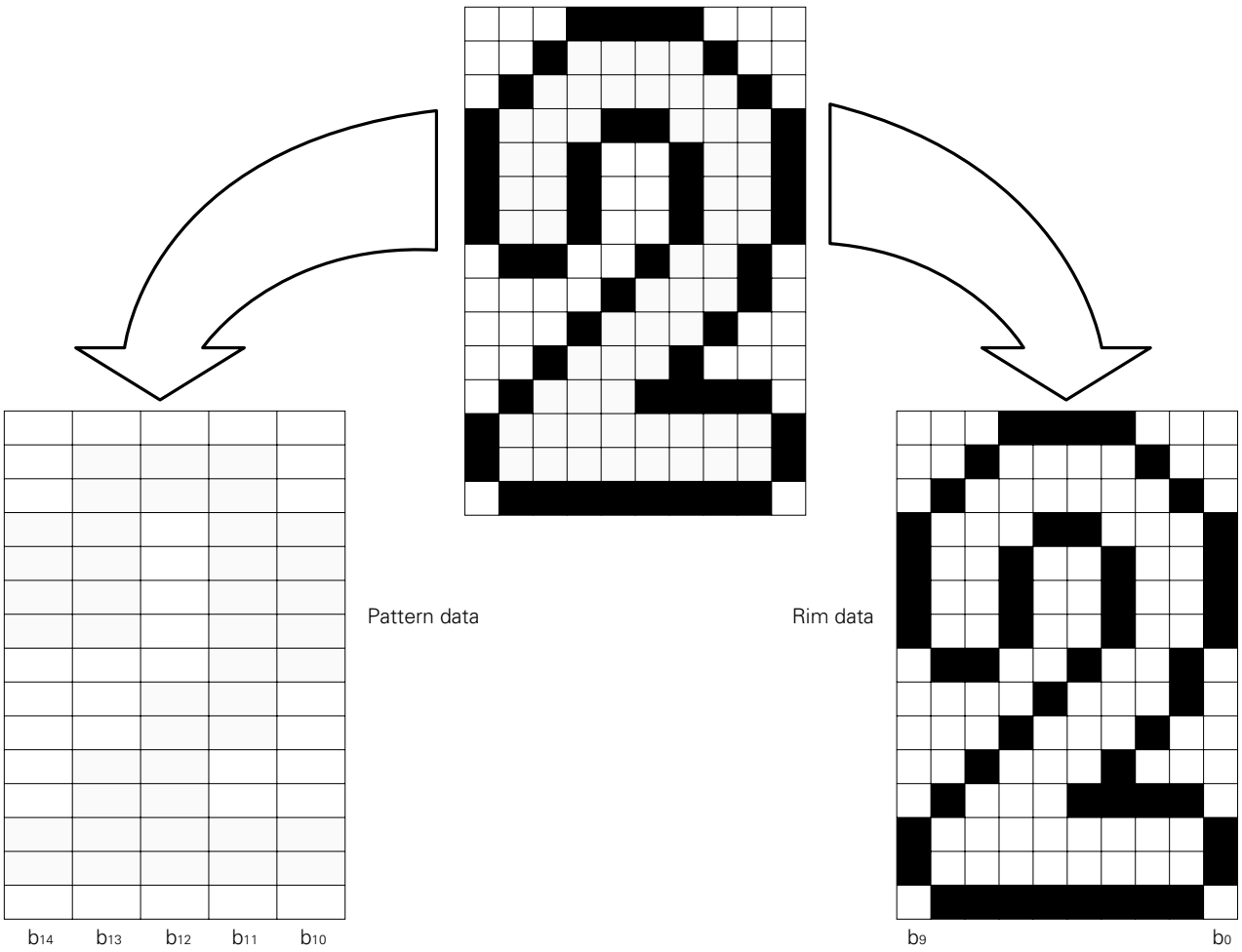


Fig. 20-8 Example of the Pattern of a Character with Rimming



	b15							b9							b0
× × × × 0 H	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
× × × × 1 H	1	0	1	1	1	0	0	0	1	0	0	0	0	1	0	0
× × × × 2 H	1	0	1	1	1	0	0	1	0	0	0	0	0	0	1	0
× × × × 3 H	1	1	1	0	1	1	1	0	0	0	1	1	0	0	0	1
× × × × 4 H	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	1
× × × × 5 H	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	1
× × × × 6 H	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	1
× × × × 7 H	1	0	0	0	1	0	0	1	1	0	0	1	0	0	1	0
× × × × 8 H	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	0
× × × × 9 H	1	0	0	1	1	0	0	0	0	1	0	0	0	1	0	0
× × × × A H	1	0	1	1	0	0	0	0	1	0	0	0	1	0	0	0
× × × × B H	1	0	1	1	0	0	0	1	0	0	0	1	1	1	1	0
× × × × C H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
× × × × D H	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
× × × × E H	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0

ROM address (in the CROM area)

Pattern data

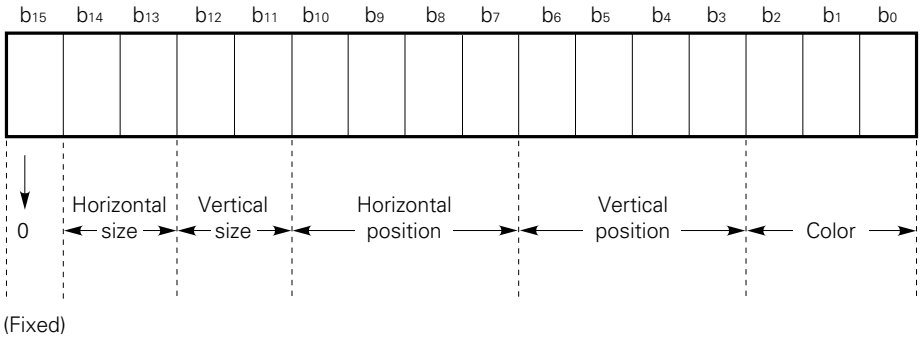
Rim data

"1" (rimming)

20.5.2 Control Data

The control data specifies the display position, size, and color of a character pattern. It is stored at ×××FH in the CROM area. One control data item consists of 16 bits. The highest bit is always 0. Fig. 20-9 shows the configuration of the control data.

Fig. 20-9 Control Data Configuration



The control data is provided between two character pattern data items. It has nothing to do with the character pattern data items at addresses before and after the control code. Any control data can be specified using data in VRAM.

(1) Horizontal size data (b_{14} and b_{13} of control data)

The horizontal size data determines the horizontal size of each image of a character. Each character has four image sizes (up to three sizes per row). Table 20-7 lists details of the horizontal size data.

Table 20-7 Horizontal Size Setting

Horizontal size data		Size	Horizontal width of a character	Maximum number of display characters per row
b_{14}	b_{13}			
0	0	Standard	2.5 μs	16
0	1	Double	5.0 μs	8
1	0	Triple	7.5 μs	5
1	1	Quadruple	10.0 μs	4

(2) Vertical size data (b_{12} and b_{11} of the control data)

The vertical size data determines the vertical size of each image of a character. Up to four sizes can be specified on each row. Table 20-8 lists details of the vertical size data.

The vertical size data specified at the beginning of a row is effective throughout that row. The vertical size data in any other control data for the same row is ignored.

Table 20-8 Vertical Size Setting

Vertical size data		Size	Vertical width of a character (interlace)	Maximum number of display characters in the vertical direction
b_{12}	b_{11}			
0	0	Standard	15H	12
0	1	Double	30H	6
1	0	Triple	45H	4
1	1	Quadruple	60H	3

(3) Horizontal position data (b_{10} to b_7 of the control data)

The horizontal position data specifies which of the 16 horizontal positions shown in Fig. 20-10 the display is to begin at. Although each row has 19 horizontal display positions, the display can start only at within 16 character positions from the left side of the screen.

The beginning of the row is specified with an absolute column number (column from 0 to 15 in Fig. 20-10). The horizontal position data consists of four bits of the control data, with b_{10} corresponding to the MSB and b_7 corresponding to the LSB, and therefore it takes a value from 0H to FH. Value 0H corresponds to column 0, and value FH to column 15.

The number of character positions left blank between two characters on the same row is specified by the horizontal position data. In other words, the position of the next character is specified by the number (in hexadecimal) of blank character positions after the current character position.

In Fig. 20-10, for example, the horizontal position data of A and that of C are 8H and 1H, respectively. If the control data of C is changed to 0, C is displayed in column 9 (at character position 9). If no control data is used after A, C is displayed also in column 9.

Remark The term number of character positions used in the above description applies when the horizontal size data is 00. If the horizontal size data is changed, the character positions are counted for the new horizontal size data. If the horizontal size data is a double size, for example, one row has only eight character positions.

(4) Vertical position data (b_6 to b_3 of the control data)

The vertical position data specifies which of the 12 rows (vertical positions) shown in Fig. 20-10 the display is to begin at. The vertical position data consists of four bits of the control data, with b_6 corresponding to the MSB and b_3 corresponding to the LSB, and it takes a value from 0H to DH. Value 0H corresponds to row 0, and value DH to column 13.

A character position at the beginning of the screen is specified with an absolute row number (row from 0 to 11 in Fig. 20-10). The number of rows left blank between two rows is specified by the vertical position data. In other words, the position of the next character is specified by the number (in hexadecimal) of blank rows after the current row.

In Fig. 20-10, for example, the vertical position data of A and that of B are 6H and 1H, respectively. Similarly, the vertical position data of D is 0H.

Remark The term number of rows used in the above description applies when the vertical size data is 00 (standard size). If the vertical size data is changed, the rows are counted based on the new vertical size data. If the vertical size data is a double size, for example, one screen has only six rows.

Fig. 20-10 Display Positions

	Column 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Column 15
Row 0																
Row 1																
Row 2																
Row 3																
Row 4																
Row 5																
Row 6									A		C					
Row 7																
Row 8				B												
Row 9					D											
Row 10																
Row 11																
Row 12																
Row 13																

(5) Color data (b_2 to b_0 of the control data)

The color data specifies the color of a display character. It is output from a specified output pin (R, G, or B pin). Table 20-9 lists the correspondence between the color data and the output pins.

Table 20-10 summarizes the relationships between the color data setting and output colors.

Table 20-9 Color Data

b_2	b_1	b_0
R	G	B

Table 20-10 Character Color

Color data			Character color
R	G	B	
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

20.5.3 Defining Display Patterns with an Assembler

With the 17K series assembler, the DCP pseudo instruction can be used to define display patterns easily. How to use the DCP pseudo instruction is described below.

(1) Instruction format

Symbol field	Mnemonic field	Operand field	Comment field
[Label:]	DCP	expression, 'display pattern'	[; comment]

(2) Explanation

(a) The expression takes value 0 or 1. The display pattern written in the second operand specifies whether to use rimming in the display pattern.

0 : No rimming
1 : Rimming

If the expression does not evaluate to 0 or 1, an error is reported.

(b) The display pattern definition consists of 10 characters and can include three different characters, O, #, and " " (blank).
If the display pattern is specified with any other character type or more then 10 characters, an error is reported. Each of these three character types corresponds to one dot and has the following meaning:

O : Bright dot
: Rimming
" " : Blank

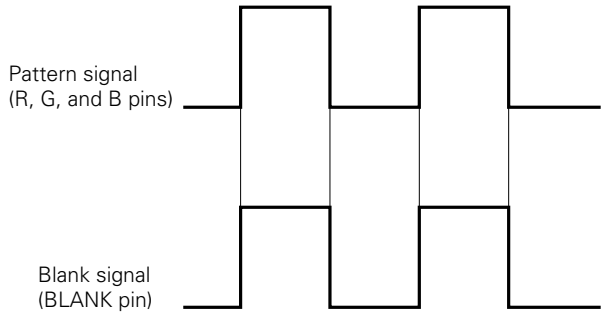
When the expression in the first operand evaluates to 0, the display pattern cannot include #.

20.6 BLANK, R, G, AND B PINS

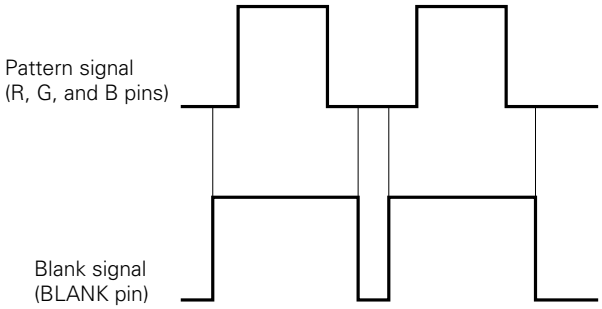
All these pins are CMOS push-pull output pins. They output an active-high signal. The BLANK pin outputs a signal to turn off a broadcasting picture. The R, G, and B pins output character pattern data. If rimming is not specified, the BLANK signal is the same as the character pattern signal (generated by ORing the R, G, and B signals). If rimming is specified, the BLANK signal output from the BLANK pin is a waveform enveloping the character pattern signal.

Fig. 20-11 IDC Output Waveform

(a) When rimming is not specified



(b) When rimming is specified

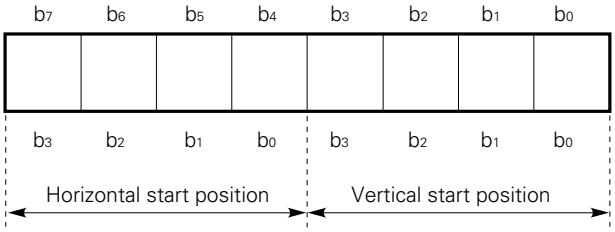


20.7 SPECIFYING THE DISPLAY START POSITION

IDC display start positions (upper left of the screen) can be specified by setting data in the IDC start position setting register. Up to 16 horizontal and vertical positions can be specified. In other words, the display position of the entire screen can be shifted. The IDC start position setting register consists of a 4-bit vertical start position setting register and a 4-bit horizontal start position setting register. The IDC start position setting register is mapped at peripheral address 01H. It can be read- and write-accessed using the GET and PUT instructions.

Note that the IDC start position setting register should not be written to when the IDCEN flag is 1.

Fig. 20-12 IDC Start Position Setting Register Configuration



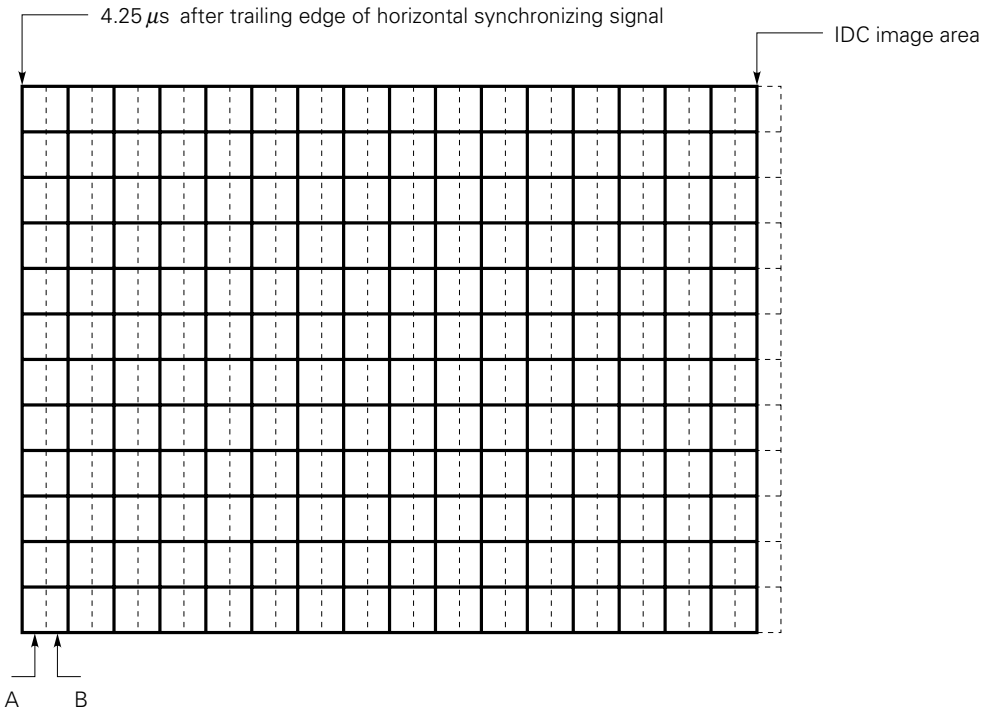
20.7.1 Horizontal Start Position Setting Register

If the horizontal start position setting register contains 0H, the horizontal start position is set 4.25 μs after the trailing edge of the horizontal sync signal. Each time the horizontal start position setting register is incremented by one, the horizontal start position shifts to the right by 250 ns; namely the following expression applies.

Horizontal start position = 4.25 μs + 250 ns × (horizontal start position setting data)

In Fig. 20-13, position A corresponds to the horizontal position setting data 0H. When the horizontal position setting data is changed to 1, the start position shifts to the right by 250 ns (one dot of the minimum-size character), that is position B. (The solid lines indicate the screen when the horizontal position data is 0, and the dotted lines, when the horizontal position data is 1.

Fig. 20-13 Horizontal Shifting



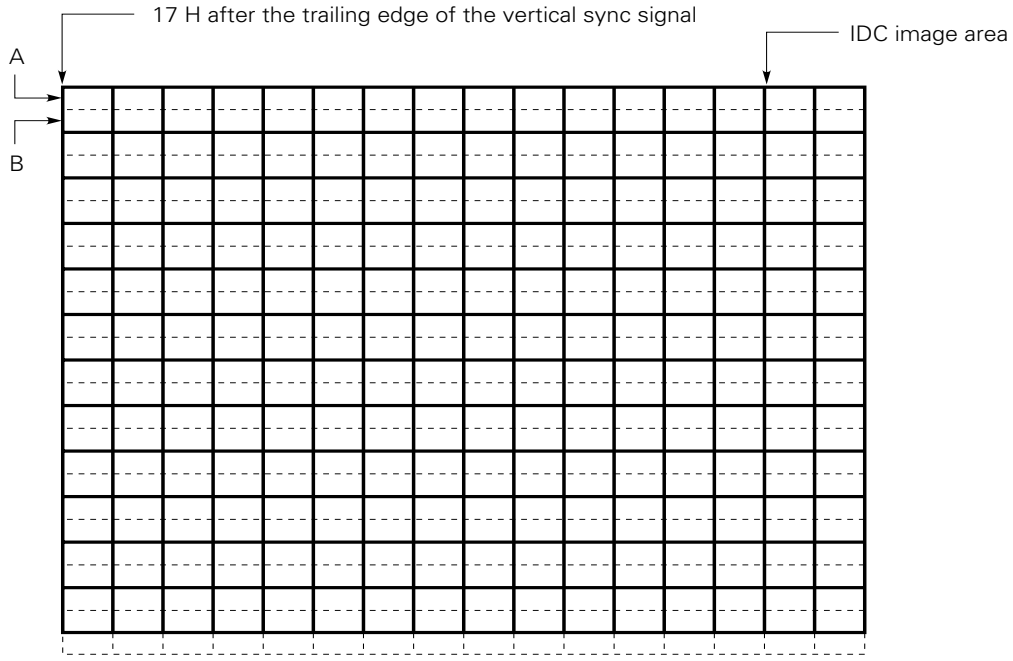
20.7.2 Vertical Start Position Setting Register

If the vertical start position setting register contains 0H, the vertical start position is set 17 H (interlace) after the trailing edge of the vertical sync signal. Each time the vertical start position setting register is incremented by one, the vertical start position shifts down by 1 H; namely the following expression applies.

Vertical start position = 17 H + 1 H × (vertical start position setting data)

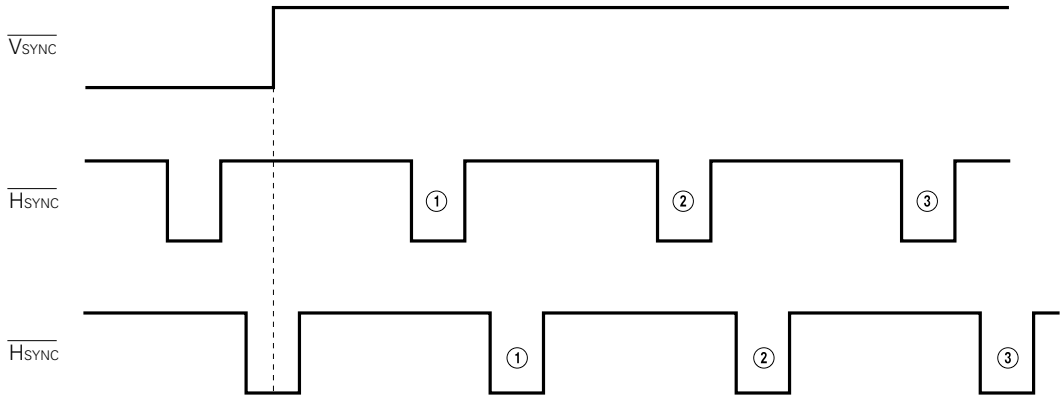
In Fig. 20-14, position A corresponds to the vertical position setting data 0H. When the vertical position setting data is changed to 1, the start position shifts down by 1 H, that is position B. (The solid lines indicate the screen when the vertical position setting data is 0, and the dotted lines, when the vertical position setting data is 1.

Fig. 20-14 Vertical Shifting



The vertical start position of the display character is determined by the vertical start position register. At this point, the vertical start position (number of horizontal scan lines) depends on the state of the $\overline{\text{VSYNC}}$ and $\overline{\text{HSYNC}}$ signals supplied to the μPD17062, as shown in Fig. 20-15. In other words, the first $\overline{\text{HSYNC}}$ signal that comes after the $\overline{\text{VSYNC}}$ signal rises is counted as 1 H.

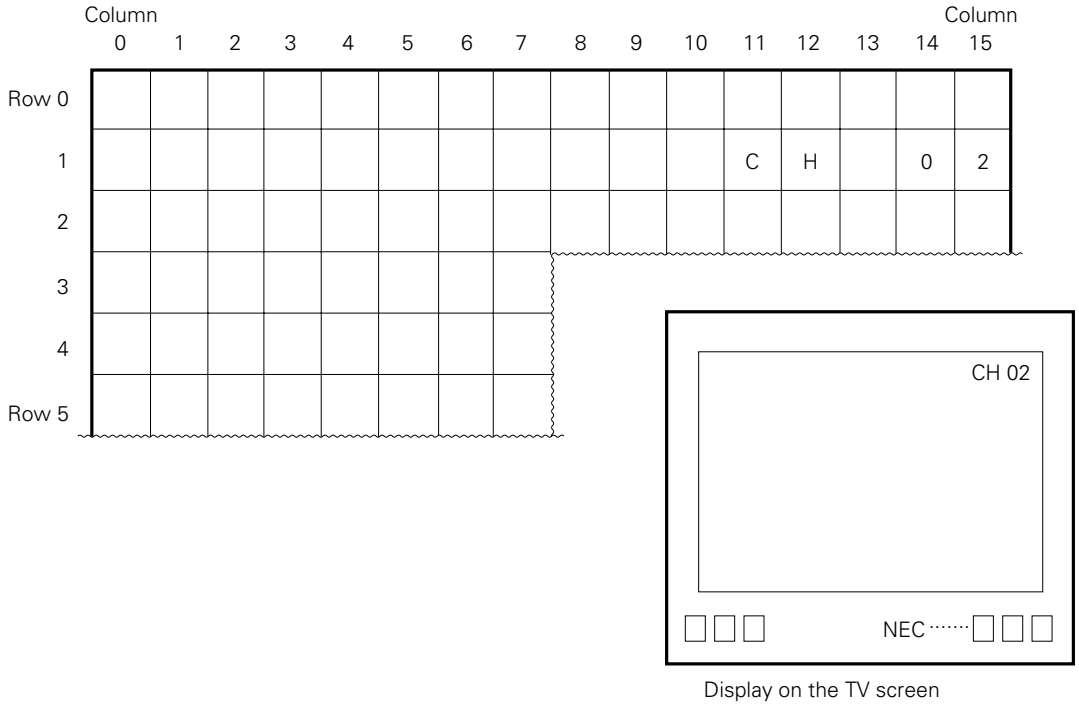
Fig. 20-15 Counting the Vertical Start Position



Each circled number corresponds to the number of scan lines.

20.8 SAMPLE PROGRAMS

The following sample program generates a display shown below.



Display on the TV screen

The RAM names of VRAM are defined as follows (tentative):

```
; ** RAM SET **
VRAM0   MEM   2.00H
VRAM1   MEM   2.01H
VRAM2   MEM   2.02H
VRAM3   MEM   2.03H
VRAM4   MEM   2.04H
VRAM5   MEM   2.05H
VRAM6   MEM   2.06H
VRAM7   MEM   2.07H
VRAM8   MEM   2.08H
VRAM9   MEM   2.09H
VRAMA   MEM   2.0AH
VRAMB   MEM   2.0BH
VRAMC   MEM   2.0CH
VRAMD   MEM   2.0DH
VRAME   MEM   2.0EH
VRAMF   MEM   2.0FH
        ⋮
```

VRAM map (BANK2)

	0	1	2	3	4	5	6	7
0	VRAM0	VRAM1	VRAM2	VRAM3	VRAM4	VRAM5	VRAM6	VRAM7
1	------	------	------	------	------	------	------	------
2	------	------	------	------	------	------	------	------

	8	9	A	B	C	D	E	F
0	VRAM8	VRAM9	VRAMA	VRAMB	VRAMC	VRAMD	VRAME	VRAMF
1	------	------	------	------	------	------	------	------
2	------	------	------	------	------	------	------	------

The sample program follows:

```
        Program start
                                    ; Performs initialization such as clearing RAM.
        ┌──────────────┐
        │Initialization│
        └──────────────┘
        SET1    IDCDMAEN            ; Selects the DMA mode.
        CLR1    IDCEN               ; Turns off the display.
;
        ; ** Channel display routine **
;
        CLR1    CROMBNK             ; Sets the CROM bank to 0.
;
        MOV     VRAM0, #1000B       ; Specifies control code 1.
        MOV     VRAM1, #0000B
;
        MOV     VRAM2, #0           ; Specifies display character data C.
        MOV     VRAM3, #0CH
;
        MOV     VRAM4, #0           ; Specifies display character data H.
        MOV     VRAM5, #0DH
;
        MOV     VRAM6, #1000B       ; Specifies control code 2.
        MOV     VRAM7, #0001B
;
        MOV     VRAM8, #0           ; Specifies display character data 0.
        MOV     VRAM9, #0
;
        MOV     VRAMA, #0           ; Specifies display character data 2.
        MOV     VRAMB, #2
;
        MOV     VRAMC, #0100B       ; CR (carriage return)
        MOV     VRAMD, #0000B
;
        MOV     VRAME, #0100B       ; CR (carriage return)
        MOV     VRAMF, #0000B
        ; ①
LOOP:
        SKF1    INTVSYN             ; Make sure Vsync = low level and turns on the display.
        BR      LOOP
        SET1    IDCEN               ; Turns on the display
        ⋮
```

At point ①, the contents of VRAM (BANK2) are as follows:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	8	0	0	C	0	D	8	1	0	0	0	2	4	0	4	0
1																

For this example, the contents of CROM are as follows:

```
                                CROM DATA
                                ; * * * * * * * * * * * * * * * * * * * * * * * * *
                                ; * * *          Image Display Controller data set      * * *
ROM ADDRESS                     ; * * * * * * * * * * * * * * * * * * * * * * * * *
    0 8 0 0                             ORG     0800H
                                        ; * * * * * * * * *
                                        ; * * *  0  * * *
                                        ; * * * * * * * * *
    0 8 0 0   0 0 0 0           DCP 0, '                '      ; "0"
    0 8 0 1   0 0 7 C           DCP 0, '     OOOOO      '
    0 8 0 2   0 0 F E           DCP 0, '   OOOOOOO      '
    0 8 0 3   0 1 C 7           DCP 0, ' OOO     OOO     '
    0 8 0 4   0 1 8 3           DCP 0, ' OO       OO     '
    0 8 0 5   0 1 8 3           DCP 0, ' OO       OO     '
    0 8 0 6   0 1 8 3           DCP 0, ' OO       OO     '
    0 8 0 7   0 1 8 3           DCP 0, ' OO       OO     '
    0 8 0 8   0 1 8 3           DCP 0, ' OO       OO     '
    0 8 0 9   0 1 8 3           DCP 0, ' OO       OO     '
    0 8 0 A   0 1 8 3           DCP 0, ' OO       OO     '
    0 8 0 B   0 1 8 3           DCP 0, ' OO       OO     '
    0 8 0 C   0 1 C 7           DCP 0, ' OOO     OOO     '
    0 8 0 D   0 0 F E           DCP 0, '   OOOOOOO      '
    0 8 0 E   0 0 7 C           DCP 0, '     OOOOO      '
                                        ; * *  CD1  * *      ; * * Control data 1 * *
    0 8 0 F   0 5 8 A           DW      0000010110001010B   ; Horizontal size = standard, and vertical size = standard
                                                            ; Horizontal position = column 11, and vertical position = row 1
                                                            ; Color = green (G), and rimming = no
```

```
                                        ; * * * * * * * * *
                                        ; * * *  1  * * *
                                        ; * * * * * * * * *
0810  0000        DCP 0, '               '    ; "1"
0811  0006        DCP 0, '        OO     '
0812  000E        DCP 0, '        OO     '
0813  001E        DCP 0, '      OOOO     '
0814  0076        DCP 0, '      OOOO     '
0815  00C6        DCP 0, '        OO     '
0816  0186        DCP 0, '        OO     '
0817  0006        DCP 0, '        OO     '
0818  0006        DCP 0, '        OO     '
0819  0006        DCP 0, '        OO     '
081A  0006        DCP 0, '        OO     '
081B  0006        DCP 0, '        OO     '
081C  0006        DCP 0, '        OO     '
081D  0006        DCP 0, '      OOOOOO   '
081E  0006        DCP 0, '      OOOOOO   '
                                        ; * *  CD2  * *      ; * * Control data 2 * *
081F  0082        DW      0000000010000010B   ; Horizontal size = standard, and vertical size = standard
                                              ; Horizontal position = column 1, and vertical position = row 0
                                              ; Color = green (G), and rimming = no
                                        ; * * * * * * * * *
                                        ; * * *  2  * * *
ROM ADDRESS                             ; * * * * * * * * *
0820  0000        DCP 0, '               '    ; "2"
0821  007C        DCP 0, '      OOOOO    '
0822  00FE        DCP 0, '     OOOOOOO   '
0823  01C7        DCP 0, '   OOO     OOO '
0824  0183        DCP 0, '   OOO       OO'
0825  0003        DCP 0, '             OO'
0826  0007        DCP 0, '            OOO'
0827  000E        DCP 0, '           OOO  '
0828  0038        DCP 0, '         OOO    '
0829  00E0        DCP 0, '        OOO     '
082A  01C0        DCP 0, '      OOO       '
082B  0180        DCP 0, '     OOO        '
082C  0180        DCP 0, '   OOO          '
082D  01FF        DCP 0, '   OOOOOOOOO    '
082E  01FF        DCP 0, '   OOOOOOOOO    '
                                        ; * *  CD2  * *
082F  0000        DW      0000000000000000B   ; NO USE
```

```
                                    ; * * * * * * * *
                                    ; * * *  3  * * *
                                    ; * * * * * * * *
0830  0000           DCP 0, '                '    ; "3"
0831  007C           DCP 0, '     OOOOO      '
0832  00FE           DCP 0, '   OOOOOOO      '
0833  01C7           DCP 0, ' OOO     OOO    '
0834  0183           DCP 0, ' OOO      OO    '
0835  0003           DCP 0, '           OO    '
                        :
                        :
                        :
                        :
                        :
                                    ; * * * * * * * *
                                    ; * * *  C  * * *
                                    ; * * * * * * * *
08C0  0000           DCP 0, '                '    ; "C"
08C1  007F           DCP 0, '     OOOOO      '
08C2  00FF           DCP 0, '   OOOOOOO      '
08C3  01C0           DCP 0, ' OOO     OOO    '
08C4  0180           DCP 0, ' OOO      OO    '
08C5  0180           DCP 0, ' OO             '
08C6  0180           DCP 0, ' OO             '
08C7  0180           DCP 0, ' OO             '
08C8  0180           DCP 0, ' OO             '
08C9  0180           DCP 0, ' OO             '
08CA  0180           DCP 0, ' OO             '
08CB  0180           DCP 0, ' OO       OO    '
08CC  01C0           DCP 0, ' OOO     OOO    '
08CD  00FF           DCP 0, '   OOOOOOO      '
08CE  007F           DCP 0, '     OOOOO      '
                                    ;
08CF  0000           DW      0000000000000000B      ; NO USE
```

```
                                        ; * * * * * * * *
                                        ; * * *  H  * * *
ROM ADDRESS                             ; * * * * * * * *
      0 8 D 0   0 0 0 0       DCP 0, '                '     ; "H"
      0 8 D 1   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D 2   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D 3   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D 4   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D 5   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D 6   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D 7   0 1 F F       DCP 0, '  OOOOOOOOO '
      0 8 D 8   0 1 F F       DCP 0, '  OOOOOOOOO '
      0 8 D 9   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D A   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D B   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D C   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D D   0 1 8 3       DCP 0, '  OO        OO '
      0 8 D E   0 1 8 3       DCP 0, '  OO        OO '
                                        ;
      0 8 D F   0 0 0 0       DW      0000000000000000B     ; NO USE
```

21. HORIZONTAL SYNC SIGNAL COUNTER

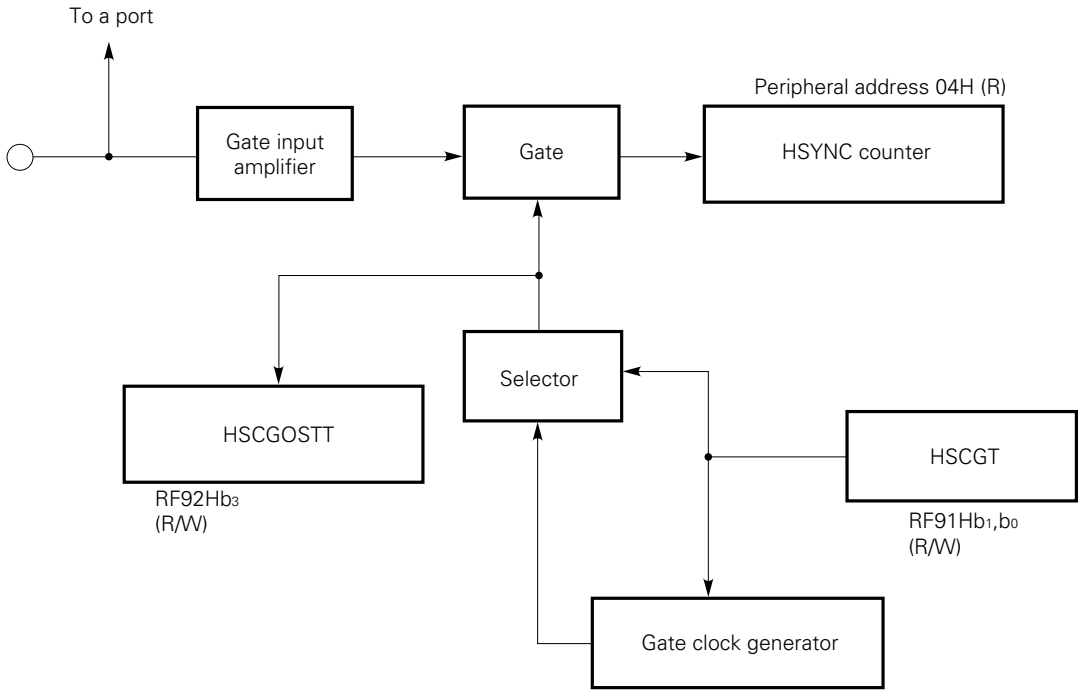
21.1 HORIZONTAL SYNC SIGNAL COUNTER CONFIGURATION

The horizontal sync signal counter counts the frequency of a horizontal sync signal for TV or similar equipment. When a TV broadcasting signal is received, a prescribed horizontal sync signal is output. Using this fact, the horizontal sync signal counter checks whether there is a broadcast station at a particular frequency.

The horizontal sync signal counter consists of a 6-bit HSYNC counter (HSC), gate clock generator, gate control register (HSCGT), gate input amplifier, and test gate open register (HSCGOSTT).

A signal supplied to the $P0B_3$/HSCNT pin is amplified by the self-biased input amplifier. The output of the amplifier passes through a gate which opens for a specific time interval specified by the gate control register. After passing through the gate, the amplifier output is counted in the 6-bit HSYNC counter. When the gate is closed, the HSYNC counter stops counting and sets 1 in the test gate open register. The HSYNC counter is a read-only register. Reading the HSYNC counter finds out how many pulses are counted when the gate is open. Dividing the number of pulses by the time during which the gate is open (1.69 ms) can obtain the frequency of the horizontal sync signal. The $P0B_3$/HSCNT pin is also used as an I/O port. It is assigned to the $P0B_3$ port. When it is used as a horizontal sync signal counter, the $P0B_3$ must be set as an input port. When it is used as a port, the HSCGT must be set with 0000B. When the $P0B_3$ is used as an input to the horizontal sync signal counter, it is read always as 0.

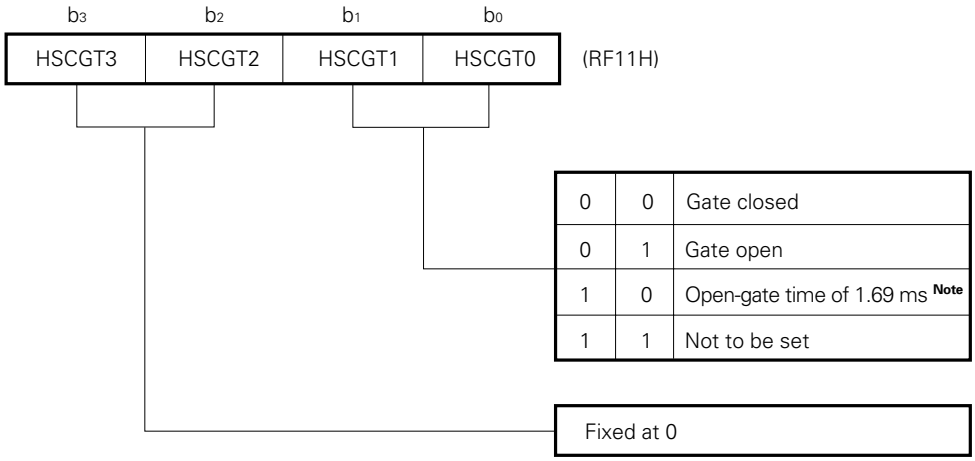
Fig. 21-1 Horizontal Sync Signal Counter Block Diagram



21.2 GATE CONTROL REGISTER (HSCGT)

The gate control register is a 2-bit register consisting of the HSCGT1 and HSCGT0 flags used to control the gate. It is mapped in the register file at 11H. The gate control register can be read- and write-accessed through the window register (system register) using the PEEK and POKE instructions, respectively.

The following modes can be set up using the gate control register.



Note The gate clock generator works only when this mode is selected.

21.2.1 Gate Closed Mode

In the gate closed mode, the gate is kept closed, disabling the HSC and gate clock generator from operating (the content of the HSYNC counter does not change). This mode also turns off the bias input to the horizontal sync signal counter, and therefore it should be selected when the port is used.

This mode is selected at a power-on reset and a clock stop.

21.2.2 Gate Open Mode

When the gate open mode is entered, the gate opens and causes the HSYNC counter to start counting the input signal after it is reset.

When the HSYNC counter overflows, it goes back to 0.

In this mode, the input pin is biased.

21.2.3 1.69 ms gate mode

When the 1.69 ms gate mode is entered, the HSYNC counter is reset and starts counting the input signal after 3.375/2 ms (with an error of 0 to 62.5 μs). The gate is kept open for 1.69 ms. The input pin is biased.

If the input signal is high when the gate is opened or closed, it is counted as one.

21.3 HSYNC COUNTER (HSC)

The HSYNC counter is mapped at peripheral address 04H. It is a 6-bit read-only binary counter. It can be read-accessed through the data buffer using the GET instruction.

When it overflows, the 6-bit HSYNC counter goes back to 00H.

The HSYNC counter is reset to 00H at a power-on reset and clock stop.

(1) Gate open bit (HSCGOSTT)

The HSCGOSTT is mapped at the MSB (b_3) of the register file at 12H. It is always high when the gate with the $\overline{\text{Hsync}}$ input is open. Note that when the 1.69 ms gate mode is selected, the HSCGOSTT becomes high when the input data is set even if there is no gate clock.

21.4 EXAMPLE OF USING THE HORIZONTAL SYNC SIGNAL

The following example is a program that uses the horizontal sync signal counter.

When the 1.69 ms gate is open

```
        :
        CLR1    P0BBIO3         ; Sets P0B3 in input mode.
        :
```

```
PEEK   WR, 0B6H
AND    WR, #0111B
POKE   0B6H, WR
```

```
LOOP:
        PEEK    WR, #92H        ; Makes sure that the gate is closed once.
        SKF     WR, #1000B
        BR      LOOP
        ;
        MOV     WR, #0010B      ; Selects the 1.69 ms gate mode.
        POKE    91H, WR
                :
LOOP2:
        PEEK    WR, #92H        ; Makes sure that the gate is closed.
        SKF     WR, #1000B
        BR      LOOP2
        GET     DBF, HSC        ; Reads the content of the HSYNC counter.
        :
```

22. INSTRUCTION SETS

22.1 OUTLINE OF INSTRUCTION SETS

b_{15} / b_{14}-b_{11} BIN	HEX	0		1	
0 0 0 0	0	ADD	r, m	ADD	m, #n4
0 0 0 1	1	SUB	r, m	SUB	m, #n4
0 0 1 0	2	ADDC	r, m	ADDC	m, #n4
0 0 1 1	3	SUBC	r, m	SUBC	m, #n4
0 1 0 0	4	AND	r, m	AND	m, #n4
0 1 0 1	5	XOR	r, m	XOR	m, #n4
0 1 1 0	6	OR	r, m	OR	m, #n4
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 s h		
1 0 0 0	8	LD	r, m	ST	m, r
1 0 0 1	9	SKE	m, #n4	SKGE	m, #n4
1 0 1 0	A	MOV	@r, m	MOV	m, @r
1 0 1 1	B	SKNE	m, #n4	SKLT	m, #n4
1 1 0 0	C	BR	addr (page 0)	CALL	addr (page 0)
1 1 0 1	D	BR	addr (page 1)	MOV	m, #n4
1 1 1 0	E			SKT	m, #n
1 1 1 1	F			SKF	m, #n

22.2 INSTRUCTIONS

Legend

AR	:	Address register
ASR	:	Address stack register pointed to by the stack pointer
addr	:	Program memory address (11 low-order bits)
BANK	:	Bank register
CMP	:	Compare flag
CY	:	Carry flag
DBF	:	Data buffer
h	:	Halt release condition
INTEF	:	Interrupt enable flag
INTR	:	Register automatically saved in the stack when an interrupt occurs
INTSK	:	Interrupt stack register
IX	:	Index register
MP	:	Data memory row address pointer
MPE	:	Memory pointer enable flag
m	:	Data memory address specified by m_R and m_C
m_R	:	Data memory row address (high-order)
m_C	:	Data memory column address (low-order)
n	:	Bit position (four bits)
n4	:	Immediate data (four bits)
PAGE	:	Page (Bits 12 and 11 of the program counter)
PC	:	Program counter
p	:	Peripheral address
p_H	:	Peripheral address (three high-order bits)
p_L	:	Peripheral address (four low-order bits)
r	:	General register column address
rf	:	Register file address
rf_R	:	Register file address (three high-order bits)
rf_C	:	Register file address (four low-order bits)
SP	:	Stack pointer
s	:	Stop release condition
WR	:	Window register
(×)	:	Contents of ×

22.3 LIST OF INSTRUCTION SETS

Instruction set	Mnemonic	Operand	Operation	Instruction code			
				Op code	Operand		
Add	ADD	r, m	(r) ← (r) + (m)	00000	m_R	m_C	r
		m, #n4	(m) ← (m) + n4	10000	m_R	m_C	n4
	ADDC	r, m	(r) ← (r) + (m) + CY	00010	m_R	m_C	r
		m, #n4	(m) ← (m) + n4 + CY	10010	m_R	m_C	n4
	INC	AR	AR ← AR + 1	00111	000	1001	0000
		IX	IX ← IX + 1	00111	000	1000	0000
Subtract	SUB	r, m	(r) ← (r) – (m)	00001	m_R	m_C	r
		m, #n4	(m) ← (m) – n4	10001	m_R	m_C	n4
	SUBC	r, m	(r) ← (r) – (m) – CY	00011	m_R	m_C	r
		m, #n4	(m) ← (m) – n4 – CY	10011	m_R	m_C	n4
Logical operation	OR	r, m	(r) ← (r) ∨ (m)	00110	m_R	m_C	r
		m, #n4	(m) ← (m) ∨ n4	10110	m_R	m_C	n4
	AND	r, m	(r) ← (r) ∧ (m)	00100	m_R	m_C	r
		m, #n4	(m) ← (m) ∧ n4	10100	m_R	m_C	n4
	XOR	r, m	(r) ← (r) ▽ (m)	00101	m_R	m_C	r
		m, #n4	(m) ← (m) ▽ n4	10101	m_R	m_C	n4
Test	SKT	m, #n	CMP ← 0, if (m) ∧ n = n, then skip	11110	m_R	m_C	n
	SKF	m, #n	CMP ← 0, if (m) ∧ n = 0, then skip	11111	m_R	m_C	n
Compare	SKE	m, #n4	(m) – n4, skip if zero	01001	m_R	m_C	n4
	SKNE	m, #n4	(m) – n4, skip if not zero	01011	m_R	m_C	n4
	SKGE	m, #n4	(m) – n4, skip if not borrow	11001	m_R	m_C	n4
	SKLT	m, #n4	(m) – n4, skip if borrow	11011	m_R	m_C	n4
Rotation	RORC	r	→ CY → $(r)_{b3}$ → $(r)_{b2}$ → $(r)_{b1}$ → $(r)_{b0}$ →	00111	000	0111	r
Transfer	LD	r, m	(r) ← (m)	01000	m_R	m_C	r
	ST	m, r	(m) ← (r)	11000	m_R	m_C	r
	MOV	@r, m	if MPE = 1: (MP, (r)) ← (m) if MPE = 0: (BANK, m_R, (r)) ← (m)	01010	m_R	m_C	r
		m, @r	if MPE = 1: (m) ← (MP, (r)) if MPE = 0: (m) ← (BANK, m_R, (r))	11010	m_R	m_C	r
		m, #n4	(m) ← n4	11101	m_R	m_C	n4
	MOVT	DBF, @AR	SP ← SP – 1, ASR ← PC, PC ← AR, DBF ← (PC), PC ← ASR, SP ← SP + 1	00111	000	0001	0000

Instruction set	Mnemonic	Operand	Operation	Instruction code			
				Op code	Operand		
Transfer	PUSH	AR	SP ← SP – 1, ASR ← AR	00111	000	1101	0000
	POP	AR	AR ← ASR, SP ← SP + 1	00111	000	1100	0000
	PEEK	WR, rf	WR ← (rf)	00111	rf_R	0011	rf_C
	POKE	rf, WR	(rf) ← WR	00111	rf_R	0010	rf_C
	GET	DBF, p	DBF ← (p)	00111	p_H	1011	p_L
	PUT	p, DBF	(p) ← DBF	00111	p_H	1010	p_L
Branch	BR	addr	PC_{10-0} ← addr, PAGE ← 0	01100	addr		
			PC_{10-0} ← addr, PAGE ← 1	01101			
		@AR	PC ← AR	00111	000	0100	0000
Sub-routine	CALL	addr	SP ← SP – 1, ASR ← PC, PC_{11} ← 0, PC_{10-0} ← addr	11100	addr		
		@AR	SP ← SP – 1, ASR ← PC, PC ← AR	00111	000	0101	0000
	RET		PC ← ASR, SP ← SP + 1	00111	000	1110	0000
	RETSK		PC ← ASR, SP ← SP + 1 and skip	00111	001	1110	0000
	RETI		PC ← ASR, INTR ← INTSK, SP ← SP + 1	00111	100	1110	0000
Interrupt	EI		INTEF ← 1	00111	000	1111	0000
	DI		INTEF ← 0	00111	001	1111	0000
Others	STOP	s	STOP	00111	010	1111	s
	HALT	h	HALT	00111	011	1111	h
	NOP		No operation	00111	100	1111	0000

22.4 BUILT-IN MACRO INSTRUCTIONS

The following macro instructions are built in the 17K series assembler (AS17K). For details, refer to the assembler user's guide.

Legend

flag n : FLG-type symbol
< > : An operand enclosed in < > is optional.

	Mnemonic	Operand	Operation	n
Built-in macro	SKTn	flag 1, ··· flag n	if (flag 1) to (flag n) = all "1", then skip	1 ≤ n ≤ 4
	SKFn	flag 1, ··· flag n	if (flag 1) to (flag n) = all "0", then skip	1 ≤ n ≤ 4
	SETn	flag 1, ··· flag n	(flag 1) to (flag n) ← 1	1 ≤ n ≤ 4
	CLRn	flag 1, ··· flag n	(flag 1) to (flag n) ← 0	1 ≤ n ≤ 4
	NOTn	flag 1, ··· flag n	if (flag n) = "0", then (flag n) ← 1 if (flag n) = "1", then (flag n) ← 0	1 ≤ n ≤ 4
	INITFLG	<NOT>flag 1, ··· <<NOT>flag n>	if description = NOT flag n, then (flag n) ← 0 if description = flag n, then (flag n) ← 1	1 ≤ n ≤ 4
	BANKn		(BANK) ← n	0 ≤ n ≤ 2

23. RESERVED SYMBOLS FOR ASSEMBLER

The reserved μPD17062 symbols for the assembler are listed below.

23.1 SYSTEM REGISTER

Symbol	Attribute	Value	Read/ write	Description
AR3	MEM	0.74H	R	Bits 15 to 12 of the address register
AR2	MEM	0.75H	R	Bits 11 to 8 of the address register
AR1	MEM	0.76H	R/W	Bits 7 to 4 of the address register
AR0	MEM	0.77H	R/W	Bits 3 to 0 of the address register
WR	MEM	0.78H	R/W	Window register
BANK	MEM	0.79H	R/W	Bank register
IXH	MEM	0.7AH	R	Bits 10 to 8 of the index register high
MPH	MEM	0.7AH	R	Bits 6 to 4 of the memory pointer
MPE	FLG	0.7AH.3	R/W	Memory pointer enable flag
IXM	MEM	0.7BH	R/W	Bits 7 to 4 of the index register
MPL	MEM	0.7BH	R/W	Bits 3 to 0 of the memory pointer
IXL	MEM	0.7CH	R/W	Bits 3 to 0 of the index register
RPH	MEM	0.7DH	R	Bits 6 to 3 of the register pointer
RPL	MEM	0.7EH	R/W	Bits 2 to 0 of the register pointer
PSW	MEM	0.7FH	R/W	Program status word
BCD	FLG	0.7EH.0	R/W	BCD operation 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

23.2 DATA BUFFER

Symbol	Attribute	Value	Read/ write	Description
DBF3	MEM	0.0CH	R/W	Bits 15 to 12 of the data buffer
DBF2	MEM	0.0DH	R/W	Bits 11 to 8 of the data buffer
DBF1	MEM	0.0EH	R/W	Bits 7 to 4 of the data buffer
DBF0	MEM	0.0FH	R/W	Bits 3 to 0 of the data buffer

23.3 PORT REGISTER

Symbol	Attribute	Value	Read/ write	Description
P0A3	FLG	0.70H.3	R/W	Bit 3 of port 0A
P0A2	FLG	0.70H.2	R/W	Bit 2 of port 0A
P0A1	FLG	0.70H.1	R/W	Bit 1 of port 0A
P0A0	FLG	0.70H.0	R/W	Bit 0 of port 0A
P0B3	FLG	0.71H.3	R/W	Bit 3 of port 0B
P0B2	FLG	0.71H.2	R/W	Bit 2 of port 0B
P0B1	FLG	0.71H.1	R/W	Bit 1 of port 0B
P0B0	FLG	0.71H.0	R/W	Bit 0 of port 0B
P0C3	FLG	0.72H.3	R/W	Bit 3 of port 0C
P0C2	FLG	0.72H.2	R/W	Bit 2 of port 0C
P0C1	FLG	0.72H.1	R/W	Bit 1 of port 0C
P0C0	FLG	0.72H.0	R/W	Bit 0 of port 0C
P0D3	FLG	0.73H.3	R[Note]	Bit 3 of port 0D
P0D2	FLG	0.73H.2	R[Note]	Bit 2 of port 0D
P0D1	FLG	0.73H.1	R[Note]	Bit 1 of port 0D
P0D0	FLG	0.73H.0	R[Note]	Bit 0 of port 0D
P1A3	FLG	1.70H.3	R/W	Bit 3 of port 1A
P1A2	FLG	1.70H.2	R/W	Bit 2 of port 1A
P1A1	FLG	1.70H.1	R/W	Bit 1 of port 1A
P1A0	FLG	1.70H.0	R/W	Bit 0 of port 1A
P1B3	FLG	1.71H.3	R/W	Bit 3 of port 1B
P1B2	FLG	1.71H.2	R/W	Bit 2 of port 1B
P1B1	FLG	1.71H.1	R/W	Bit 1 of port 1B
P1B0	FLG	1.71H.0	R/W	Bit 0 of port 1B
P1C3	FLG	1.72H.3	R/W	Bit 3 of port 1C
P1C2	FLG	1.72H.2	R/W	Bit 2 of port 1C
P1C1	FLG	1.72H.1	R/W	Bit 1 of port 1C

Note These are read-only ports. However, even if an output instruction is written, the assembler (IE-17K) does not generate an error message. Also, operation is not affected even if it is actually executed on the device.

23.4 REGISTER FILES

Symbol	Attribute	Value	Read/ write	Description
IDCDMAEN	FLG	0.80H.1	R/W	DMA enable flag
SP	MEM	0.81H	R/W	Stack pointer
CE	FLG	0.87H.0	R	CE pin status flag
SIO0CH	FLG	0.88H.3	R/W	SIO0 channel selection flag
SB	FLG	0.88H.2	R/W	SIO0 mode selection flag
SIO0MS	FLG	0.88H.1	R/W	SIO0 clock mode selection flag
SIO0TX	FLG	0.88H.0	R/W	SIO0 TX/RX selection mode
BTM0ZX	FLG	0.89H.3	R/W	Timer 0 interrupt mode selection flag
BTM0CK2	FLG	0.89H.2	R/W	Timer 0 carry FF mode selection flag
BTM0CK1	FLG	0.89H.1	R/W	Timer 0 carry FF mode selection flag
BTM0CK0	FLG	0.89H.0	R/W	Timer 0 carry FF mode selection flag
INTVSYN	FLG	0.8FH.2	R	Vsync pin status flag
INTNC	FLG	0.8FH.0	R	INT_{NC} pin status flag
HSCGT3	FLG	0.91H.3	R/W	Hsync counter mode selection flag (dummy: 0)
HSCGT2	FLG	0.91H.2	R/W	Hsync counter mode selection flag (dummy: 0)
HSCGT1	FLG	0.91H.1	R/W	Hsync counter mode selection flag
HSCGT0	FLG	0.91H.0	R/W	Hsync counter mode selection flag
HSCGOSTT	FLG	0.92H.3	R	Hsync counter gate open flag
PLLRFCK3	FLG	0.93H.3	R/W	PLL reference clock selection flag
PLLRFCK2	FLG	0.93H.2	R/W	PLL reference clock selection flag
PLLRFCK1	FLG	0.93H.1	R/W	PLL reference clock selection flag
PLLRFCK0	FLG	0.93H.0	R/W	PLL reference clock selection flag
INTNCMD3	FLG	0.95H.3	R/W	INT_{NC} pin status flag (dummy)
INTNCMD2	FLG	0.95H.2	R/W	INT_{NC} pin status flag
INTNCMD1	FLG	0.95H.1	R/W	INT_{NC} pin status flag
INTNCMD0	FLG	0.95H.0	R/W	INT_{NC} pin status flag
BTM0CY	FLG	0.97H.0	R	Timer 0 carry FF status flag
SBACK	FLG	0.98H.3	R/W	Serial bus acknowledge flag
SIO0NWT	FLG	0.98H.2	R/W	SIO0 no wait flag
SIO0WRQ1	FLG	0.98H.1	R/W	SIO0 wait request flag
SIO0WRQ0	FLG	0.98H.0	R/W	SIO0 wait request flag
IEGVSYN	FLG	0.9FH.2	R/W	Vsync interrupt edge selection flag
IEGNC	FLG	0.9FH.0	R/W	INTNC interrupt edge selection flag
ADCCH2	FLG	0.0A1H.3	R/W	A/D converter channel selection flag
ADCCH1	FLG	0.0A1H.2	R/W	A/D converter channel selection flag
ADCCH0	FLG	0.0A1H.1	R/W	A/D converter channel selection flag
ADCCMP	FLG	0.0A1H.0	R/W	A/D converter judge flag
PLLUL	FLG	0.0A2H.0	R	PLL unlock FF flag
P1CGIO	FLG	0.0A7H.0	R/W	Port 1C I/O selection flag

Symbol	Attribute	Value	Read/ write	Description
SIO0SF8	FLG	0.0A8H.3	R	SIO0 shift 8 clock flag
SIO0SF9	FLG	0.0A8H.2	R	SIO0 shift 9 clock flag
SBSTT	FLG	0.0A8H.1	R	Serial bus start test flag
SBBSY	FLG	0.0A8H.0	R	Serial bus busy flag
IPSIO0	FLG	0.0AFH.3	R/W	SIO0 interrupt permission flag
IPVSYN	FLG	0.0AFH.2	R/W	Vsync interrupt permission flag
IPBTM0	FLG	0.0AFH.1	R/W	Timer 0 interrupt permission flag
IPNC	FLG	0.0AFH.0	R/W	INTNC interrupt permission flag
CROMBNK	FLG	0.0B0H.0	R/W	CROM bank selection flag
IDCEN	FLG	0.0B1H.0	R/W	IDC enable flag
PLULSEN3	FLG	0.0B2H.3	R/W	PLL unlock time selection flag (dummy: 0)
PLULSEN2	FLG	0.0B2H.2	R/W	PLL unlock time selection flag (dummy: 0)
PLULSEN1	FLG	0.0B2H.1	R/W	PLL unlock time selection flag
PLULSEN0	FLG	0.0B2H.0	R/W	PLL unlock time selection flag
P1BBIO3	FLG	0.0B5H.3	R/W	P1B3 I/O selection flag
P1BBIO2	FLG	0.0B5H.2	R/W	P1B2 I/O selection flag
P1BBIO1	FLG	0.0B5H.1	R/W	P1B1 I/O selection flag
P1BBIO0	FLG	0.0B5H.0	R/W	P1B0 I/O selection flag
P0BBIO3	FLG	0.0B6H.3	R/W	P0B3 I/O selection flag
P0BBIO2	FLG	0.0B6H.2	R/W	P0B2 I/O selection flag
P0BBIO1	FLG	0.0B6H.1	R/W	P0B1 I/O selection flag
P0BBIO0	FLG	0.0B6H.0	R/W	P0B0 I/O selection flag
P0ABIO3	FLG	0.0B7H.3	R/W	P0A3 I/O selection flag
P0ABIO2	FLG	0.0B7H.2	R/W	P0A2 I/O selection flag
P0ABIO1	FLG	0.0B7H.1	R/W	P0A1 I/O selection flag
P0ABIO0	FLG	0.0B7H.0	R/W	P0A0 I/O selection flag
SIO0IMD3	FLG	0.0B8H.3	R/W	SIO0 interrupt mode selection flag (dummy: 0)
SIO0IMD2	FLG	0.0B8H.2	R/W	SIO0 interrupt mode selection flag (dummy: 0)
SIO0IMD1	FLG	0.0B8H.1	R/W	SIO0 interrupt mode selection flag
SIO0IMD0	FLG	0.0B8H.0	R/W	SIO0 interrupt mode selection flag
SIO0CK3	FLG	0.0B9H.3	R/W	SIO0 shift clock selection flag (dummy: 0)
SIO0CK2	FLG	0.0B9H.2	R/W	SIO0 shift clock selection flag (dummy: 0)
SIO0CK1	FLG	0.0B9H.1	R/W	Serial clock selection
SIO0CK0	FLG	0.0B9H.0	R/W	Serial clock selection
IRQSIO0	FLG	0.0BFH.3	R	SIO0 interrupt request flag
IRQVSYN	FLG	0.0BFH.2	R	Vsync interrupt request flag
IRQBTM0	FLG	0.0BFH.1	R	Timer 0 interrupt request flag
IRQNC	FLG	0.0BFH.0	R	INT_{NC} interrupt request flag

23.5 PERIPHERAL HARDWARE REGISTER

Symbol	Attribute	Value	Read/ write	Description
IDCORG	DAT	01H	R/W	IDC start position setting register
ADCR	DAT	02H	R/W	A/D-converter reference-voltage (V_{REF}) setting register
SIO0SFR	DAT	03H	R/W	SIO0 register
HSC	DAT	04H	R	Hsync counter data register
PWMR0	DAT	05H	R/W	PWM data register 0
PWMR1	DAT	06H	R/W	PWM data register 1
PWMR2	DAT	07H	R/W	PWM data register 2
PWMR3	DAT	08H	R/W	PWM data register 3
AR	DAT	40H	R/W	Address register
PLLR	DAT	41H	R/W	PLL data register
AR_EPA1	DAT	8040H	–	CALL/BR/MOVT instruction operand (EPA bit is on)
AR_EPA0	DAT	4040H	–	CALL/BR/MOVT instruction operand (EPA bit is off)

23.6 OTHERS

Symbol	Attribute	Value	Description
DBF	DAT	0FH	Fixed operand value for a PUT/GET/MOVT instruction
IX	DAT	01H	Fixed operand value for an INC instruction

24. ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS (T_a = 25 ±2 °C)

Parameter	Symbol	Rated value	Unit
Supply voltage	V_{DD}	–0.3 to +6.0	V
Input voltage	V_I	–0.3 to V_{DD} + 0.3	V
Output voltage	V_O	–0.3 to V_{DD} + 0.3 (excluding $P1A_3$ to $P1A_0$ and PWM_3 to PMW_0)	V
Output absorption current	I_O	10 (excluding P1A)	mA
Output withstand voltage	V_{BDS}	13 (P1A, PWM)	V
Operating temperature	T_{opt1}	–20 to +70	°C
	T_{opt2}	–40 to +85 (when IDC has stopped)	
Storage temperature	V_{stg}	–55 to +125	°C

RECOMMENDED OPERATION RANGE (T_a = –40 to +85 °C)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Supply voltage	V_{DD1}	T_a = –20 to +70 °C (when CPU, PLL, and IDC are operating)	4.5	5.0	5.5	V
	V_{DD2}	When CPU and PLL are operating (IDC is not operating)	4.5	5.0	5.5	V
	V_{DD3}	When only CPU is operating (PLL and IDC are not operating)	4.0	5.0	5.5	V
Data hold voltage	V_{DDR}	When crystal oscillation has stopped	3.0		5.5	V
Output withstand voltage	V_{BDS}	$P1B_3$-$P1B_1$	0.0		12.5	V
Input amplitude	V_{in1}	VCO	0.7		V_{DD}	V_{P-P}
Supply voltage rise time	t_{rise}	V_{DD} : 0 → 4.0 V			500	ms

AC CHARACTERISTICS (T_a = –40 to +85 °C, V_{DD} = 5 V ±10 %, RH ≤ 70 %)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Operating frequency	f_{in1}	VCO Sine wave input V_{in} = 0.7 V_{P-P}	0.7		20	MHz
	f_{in2}	TMIN	45		65	Hz
	f_{in3}	HSCNT	10		20	kHz
IDC jitter	IDC_G			4.0	8.0	ns
IDC horizontal start position	IDC_{HP}	From trailing edge of $\overline{H_{SYNC}}$		4.25		μs
IDC vertical start position	IDC_{VP}	From trailing edge of $\overline{V_{SYNC}}$		17		H

A/D CONVERTER CHARACTERISTICS (T_a = –40 to +85 °C, V_{DD} = 5 V ±10 %, RH ≤ 70 %)

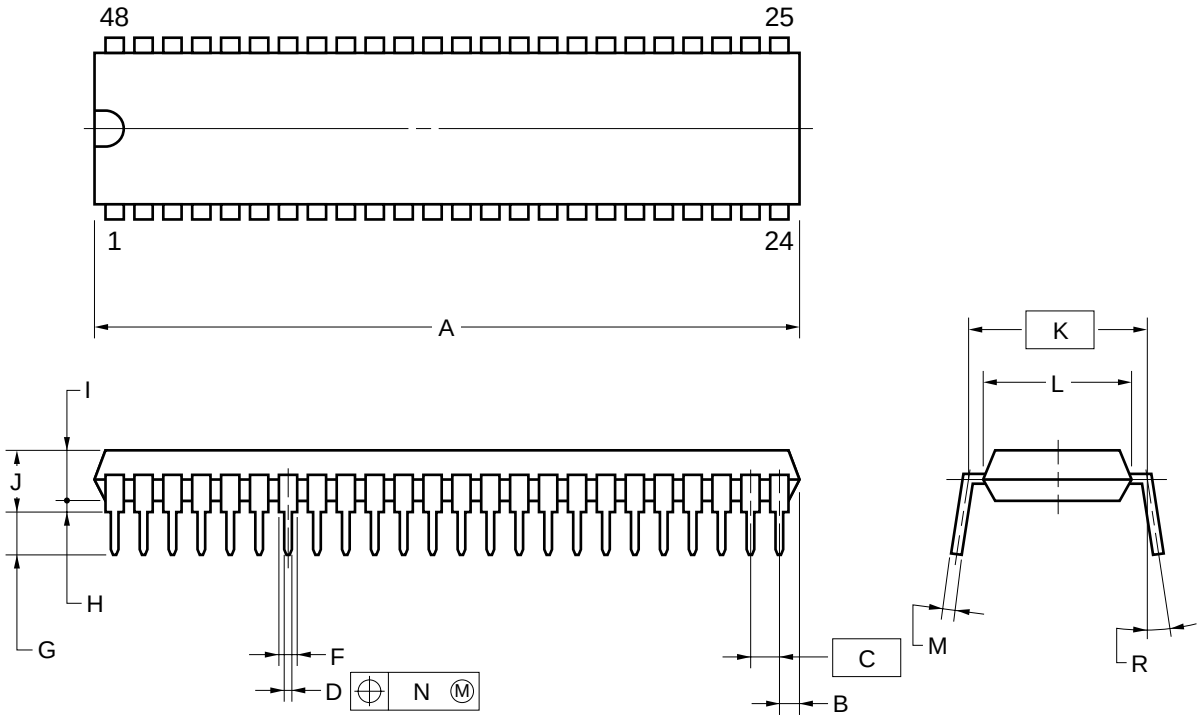
Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
A/D conversion resolution					4	bit
A/D conversion total error tolerance		T_a = –10 to +50 °C	±0.5		±1.0	LSB
A/D input impedance			1.0			MΩ

DC CHARACTERISTICS (T_a = –40 to +85 °C, V_{DD} = 5 V ±10 %, RH ≤ 70 %)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Supply voltage	V_{DD1}	T_a = –20 to +85 °C (when CPU, PLL, and IDC are operating)	4.5	5.0	5.5	V
	V_{DD2}	When CPU and PLL are operating (IDC is not operating)	4.5	5.0	5.5	V
	V_{DD3}	When only CPU is operating (PLL and IDC are not operating)	4.0	5.0	5.5	V
Supply current	I_{DD4}	When only CPU is operating, PLL and IDC are not operating, and HALT instruction is being used (20 instructions are executed at 5-ms intervals)		1.0	3.0	mA
Data hold voltage	V_{DDR}	When the timer FF power failure detection method is used When crystal oscillation has stopped	3.0		5.5	V
Data hold current	I_{DDR}	When crystal oscillation has stopped T_a = 25 °C		1.5	10	μA
High-level input voltage	V_{IH1}	P0A, P0B, P0D, P1B, P1C	0.7V_{DD}			V
	V_{IH2}	CE, INT_{NC}, $\overline{V_{SYNC}}$, $\overline{H_{SYNC}}$	0.8V_{DD}			mA
Low-level input voltage	V_{IL1}	P0A, P0B, P0D, P1B, P1C			0.3V_{DD}	V
	V_{IL2}	CE, INT_{NC}, $\overline{V_{SYNC}}$, $\overline{H_{SYNC}}$			0.2V_{DD}	mA
High-level output voltage	I_{OH1}	P0A, P0B, P0C, P1B, P1C, RED, GREEN, BLUE, BLANK V_{OH} = V_{DD} – 1 V	–1.0	–2.0		mA
Low-level output voltage	I_{OL1}	P0A, P0B, P0C, P1B, P1C, RED, GREEN, BLUE, BLANK V_{OH} = V_{DD} – 1 V	1.0	2.0		mA
	I_{OL4}	P1A V_{OL} = 1 V	15	22		mA
High-level input current	I_{IH1}	When P0D pull-down resistor is applied V_{IH} = V_{DD}	20	70	150	μA
	I_{IH2}	VCO V_{IH} = V_{DD}	0.1	0.8	1.3	mA
Output off leakage current	I_{IL1}	P1A, PWM V_{OH} = 12.5 V			0.5	μA
	I_{IL2}	EO V_{OH} = V_{DD}, V_{OL} = 0 V			±1	μA
Output withstand voltage	V_{BDS}	P1A, PWM			12.5	V

25. PACKAGE DRAWINGS

48PIN PLASTIC SHRINK DIP (600 mil)



NOTES

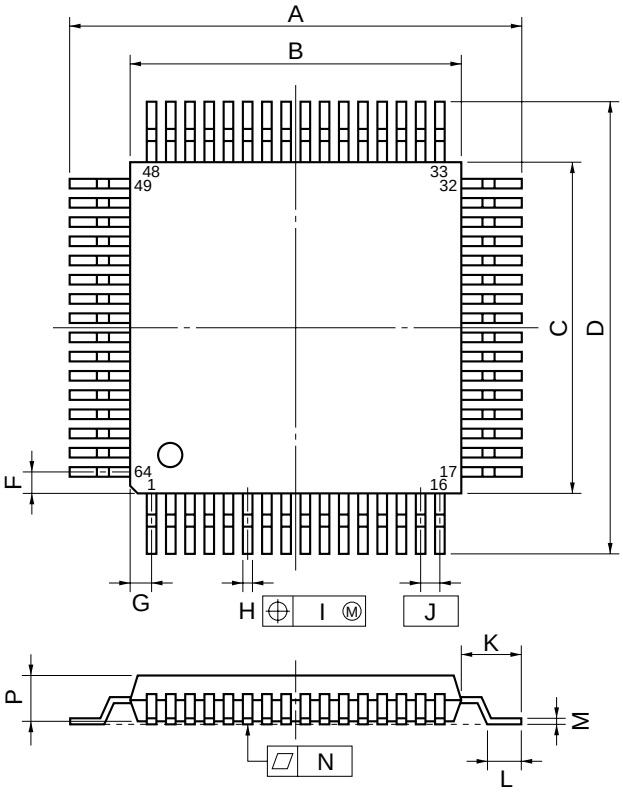
1) Each lead centerline is located within 0.17 mm (0.007 inch) of its true position (T.P.) at maximum material condition.

2) Item "K" to center of leads when formed parallel.

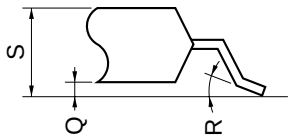
ITEM	MILLIMETERS	INCHES
A	44.46 MAX.	1.751 MAX.
B	1.78 MAX.	0.070 MAX.
C	1.778 (T.P.)	0.070 (T.P.)
D	0.50 ± 0.10	$0.020^{+0.004}_{-0.005}$
F	0.85 MIN.	0.033 MIN.
G	3.2 ± 0.3	0.126 ± 0.012
H	0.51 MIN.	0.020 MIN.
I	4.31 MAX.	0.170 MAX.
J	5.72 MAX.	0.226 MAX.
K	15.24 (T.P.)	0.600 (T.P.)
L	13.2	0.520
M	$0.25^{+0.10}_{-0.05}$	$0.010^{+0.004}_{-0.003}$
N	0.17	0.007
R	0~15°	0~15°

P48C-70-600B-1

64 PIN PLASTIC QFP (□14)



detail of lead end



NOTE

Each lead centerline is located within 0.13 mm (0.005 inch) of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS	INCHES
A	17.2±0.2	0.677±0.008
B	14.0±0.2	$0.551^{+0.009}_{-0.008}$
C	14.0±0.2	$0.551^{+0.009}_{-0.008}$
D	17.2±0.2	0.677±0.008
F	1.0	0.039
G	1.0	0.039
H	0.35±0.10	$0.014^{+0.004}_{-0.005}$
I	0.13	0.005
J	0.8 (T.P.)	0.031 (T.P.)
K	1.6±0.2	0.063±0.008
L	0.8±0.2	$0.031^{+0.009}_{-0.008}$
M	$0.15^{+0.10}_{-0.05}$	$0.006^{+0.004}_{-0.003}$
N	0.10	0.004
P	2.7	0.106
Q	0.125±0.075	0.005±0.003
R	5°±5°	5°±5°
S	3.0 MAX.	0.119 MAX.

S64GC-80-3BE-1

26. RECOMMENDED SOLDERING CONDITIONS

The conditions listed below shall be met when soldering the μPD17062.

For details of the recommended soldering conditions, refer to our document *SMD Surface Mount Technology Manual* (IEI-1207).

Please consult with our sales offices in case any other soldering process is used, or in case soldering is done under different conditions.

Table 26-1 Soldering Conditions for Surface-Mount Devices

μPD17062GC-×××-3BE: 64-pin plastic QFP (14 × 14 mm)

Soldering process	Soldering conditions	Symbol
Infrared ray reflow	Peak package's surface temperature: 235 °C Reflow time: 30 seconds or less (at 210 °C or more) Maximum allowable number of reflow processes: 1 Exposure limit Note: 7 days (20 hours of pre-baking is required at 125 °C afterward.)	IR35-207-1
VPS	Peak package's surface temperature: 215 °C Reflow time: 40 seconds or less (at 200 °C or more) Maximum allowable number of reflow processes: 2 Exposure limit Note: 7 days (20 hours of pre-baking is required at 125 °C afterward.) <Cautions> (1) Do not start reflow-soldering the device if its temperature is higher than the room temperature because of a previous reflow soldering. (2) Do not use water for flux cleaning before a second reflow soldering.	VP15-207-2
Partial heating method	Terminal temperature: 300 °C or less Heat time: 3 seconds or less (for each side of device)	–

Note Exposure limit before soldering after dry-pack package is opened.
Storage conditions: Temperature of 25 °C and maximum relative humidity at 65% or less

Caution Do not apply more than a single process at once, except for "Partial heating method."

Table 26-2 Soldering Conditions for Through Hole Mount Devices

μPD17062CU-×××: 48-pin plastic shrink DIP (600 mil)

Soldering process	Soldering conditions
Wave soldering (only for leads)	Solder temperature: 260 °C or less Flow time: 10 seconds or less
Partial heating method	Terminal temperature: 260 °C or less Heat time: 10 seconds or less

Caution In wave soldering, apply solder only to the lead section. Care must be taken that jet solder does not come in contact with the main body of the package.

APPENDIX DEVELOPMENT TOOLS

The following support tools are available for developing programs for the μPD17062.

Hardware

Name	Description
In-circuit emulator [IE-17K IE-17K-ET[Note 1] EMU-17K[Note 2]]	The IE-17K, IE-17K-ET, and EMU-17K are in-circuit emulators applicable to the 17K series. The IE-17K and IE-17K-ET are connected to the PC-9800 series (host machine) or IBM PC/AT™ through the RS-232-C interface. The EMU-17K is inserted into the extension slot of the PC-9800 series (host machine). Use the system evaluation board (SE board) corresponding to each product together with one of these in-circuit emulators. SIMPLEHOST™, a man machine interface, implements an advanced debug environment. The EMU-17K also enables user to check the contents of the data memory in real time.
SE board (SE-17002)	The SE-17002 is an SE board for the μPD17002 and μPD17062. It is used solely for evaluating the system. It is also used for debugging in combination with the in-circuit emulator.
Emulation probe (EP-17002CU)	The EP-17002CU is an emulation probe for the 48-pin shrink DIP (600 mil). It is used to connect the SE board and the target system.
Emulation probe (EP-17002GC)	The EP-17002GC is an emulation probe for the 64-pin QFP (14 × 14 mm). It is used with EV-9400GC-64[Note 3] to connect the SE board to the target system.
Conversion socket (EV-9200GC-64[Note 3])	The EV-9200GC-64 is a conversion socket used to connect the EP-17002GC to the target system.

Notes 1. Low-end model, operating on an external power supply

2. The EMU-17K is a product of IC Co., Ltd. Contact IC Co., Ltd. (Tokyo, 03-3447-3793) for details.

3. The EP-17002GC is supplied together with one EV-9200GC-64. A set of five EV-9200GC-64 is also available.

Software

Name	Description	Host machine	OS		Distribution media	Part number
17K series assembler (AS17K)	AS17K is an assembler applicable to the 17K series. In developing μPD17062 programs, AS17K is used in combination with a device file (AS17062).	PC-9800 series	MS-DOS™		5.25-inch, 2HD	μS5A10AS17K
					3.5-inch, 2HD	μS5A13AS17K
		IBM PC/AT	PC DOS™		5.25-inch, 2HC	μS7B10AS17K
					3.5-inch, 2HC	μS7B13AS17K
Device file (AS17062)	AS17062 is a device file for the μPD17062 . It is used together with the assembler (AS17K), which is applicable to the 17K series.	PC-9800 series	MS-DOS		5.25-inch, 2HD	μS5A10AS17062
					3.5-inch, 2HD	μS5A13AS17062
		IBM PC/AT	PC DOS		5.25-inch, 2HC	μS7B10AS17062
					3.5-inch, 2HC	μS7B13AS17062
Support software (SIMPLEHOST)	SIMPLEHOST, running on the Windows™, provides man-machine-interface in developing programs by using a personal computer and the in-circuit emulator.	PC-9800 series	MS-DOS	Windows	5.25-inch, 2HD	μS5A10IE17K
					3.5-inch, 2HD	μS5A13IE17K
		IBM PC/AT	PC DOS		5.25-inch, 2HC	μS7B10IE17K
					3.5-inch, 2HC	μS7B13IE17K

Remark The following table lists the versions of the operating systems described in the above table.

OS	Versions
MS-DOS	Ver. 3.30 to Ver. 5.00A[Note]
PC DOS	Ver. 3.1 to Ver. 5.0[Note]
Windows	Ver. 3.0 to Ver. 3.1

Note MS-DOS versions 5.00 and 5.00A and PC DOS Ver. 5.0 are provided with a task swap function. This function, however, cannot be used in these software packages.

[MEMO]

Cautions on CMOS Devices

① **Countermeasures against static electricity for all MOSs**

Caution When handling MOS devices, take care so that they are not electrostatically charged.

Strong static electricity may cause dielectric breakdown in gates. When transporting or storing MOS devices, use conductive trays, magazine cases, shock absorbers, or metal cases that NEC uses for packaging and shipping. Be sure to ground MOS devices during assembling. Do not allow MOS devices to stand on plastic plates or do not touch pins. Also handle boards on which MOS devices are mounted in the same way.

② **CMOS-specific handling of unused input pins**

Caution Hold CMOS devices at a fixed input level.

Unlike bipolar or NMOS devices, if a CMOS device is operated with no input, an intermediate-level input may be caused by noise. This allows current to flow in the CMOS device, resulting in a malfunction. Use a pull-up or pull-down resistor to hold a fixed input level. Since unused pins may function as output pins at unexpected times, each unused pin should be separately connected to the V_{DD} or GND pin through a resistor.

If handling of unused pins is documented, follow the instructions in the document.

③ **Statuses of all MOS devices at initialization**

Caution The initial status of a MOS device is unpredictable when power is turned on.

Since characteristics of a MOS device are determined by the amount of ions implanted in molecules, the initial status cannot be determined in the manufacture process. NEC has no responsibility for the output statuses of pins, input and output settings, and the contents of registers at power on. However, NEC assures operation after reset and items for mode setting if they are defined.

When you turn on a device having a reset function, be sure to reset the device first.

Caution This product contains an I^2C bus interface circuit.
When using the I^2C bus interface, notify its use to NEC when ordering custom code. NEC can guarantee the following only when the customer informs NEC of the use of the interface:
Purchase of NEC I^2C components conveys a license under the Philips I^2C Patent Rights to use these components in an I^2C system, provided that the system conforms to the I^2C Standard Specification as defined by Philips.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.
NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.
While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customer must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.
NEC devices are classified into the following three quality grades:
"Standard", "Special", and "Specific". The Specific quality grade applies only to devices developed based on a customer designated "quality assurance program" for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.

- Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots
- Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)
- Specific: Aircrafts, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

The quality grade of NEC devices in "Standard" unless otherwise specified in NEC's Data Sheets or Data Books. If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact NEC Sales Representative in advance.
Anti-radioactive design is not implemented in this product.

M4 94.11

SIMPLEHOST is a trademark of NEC Corporation.
MS-DOS and Windows are trademarks of Microsoft Corporation.
PC/AT and PC DOS are trademarks of IBM Corporation.