NEC
ELECTRON DEVICE

# MOS INTEGRATED CIRCUIT
# μPD75238

## 4 BIT SINGLE-CHIP MICROCOMPUTER

The μPD75238 is a single-chip microcomputer which contains a CPU capable of 1-, 4-, and 8-bit data processing, ROM, RAM, and I/O ports. In addition, it contains a fluorescent display tube (FIP′) controller/driver, A/D converter, clock timer, timer/pulse generator capable of 14-bit PWM output, serial interface, and vectored interrupt function.

In comparison with the μPD75217, the μPD75238 has larger ROM and RAM capacity and has been enhanced in such peripheral facilities as the display function of the FIP controller/driver, I/O ports, A/D converter, serial interface.

The μPD75238 finds best use in such applications as timer/tuner of VCRs from advanced type to common type, configuration of one-chip system control microcomputer, advanced CD player, advanced microwave ovens, etc.

With the μPD75238, the μPD75P238, which is a PROM product, and various development tools including IE-75001-R and assemblers are available. They can be used for evaluation during system development and small-volume production.

### FEATURES

- Mass-storage built-in ROM and RAM
  - Program memory (ROM) : 32K × 8
  - Data memory (RAM) : 1K × 4
- I/O port: 64 lines (excluding pins dedicated to FIP)
- Minimum instruction execution time: 0.67 μs (at 6.0 MHz)
- Instruction execution time specification function to allow a wide range of operating voltages
- Programmable FIP controller/driver contained
  - Number of segments: 9 to 24 segments
  - Number of digits : 9 to 16 digits
- 8-bit A/D converter: 8 channels
- Enhanced timer/counter function: 5 channels
- 8-bit serial interface: 2 channels
- Application-oriented interrupt functions
- PROM version device: μPD75P238

### ORDERING INFORMATION

| Part number | Package | Quality grade |
|---|---|---|
| μPD75238GJ-×××-5BG | 94-pin plastic QFP (20 × 20 mm) | Standard |

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

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

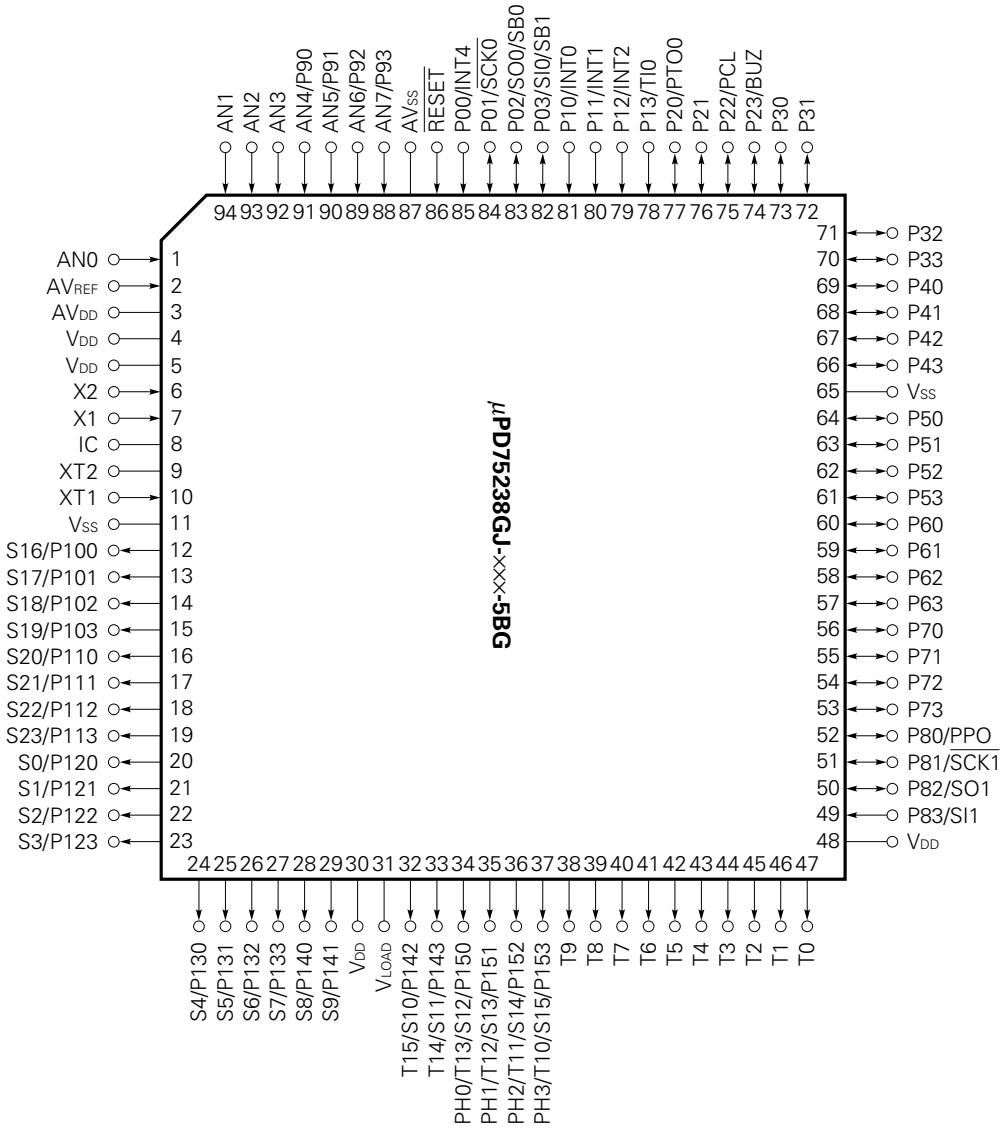
Document No. IC-2777A
(O.D. No. IC-8177A)
Date Published February 1993 P
Printed in Japan

**Major changes in this revision are indicated by stars (★) in the margins.**

## FUNCTIONS

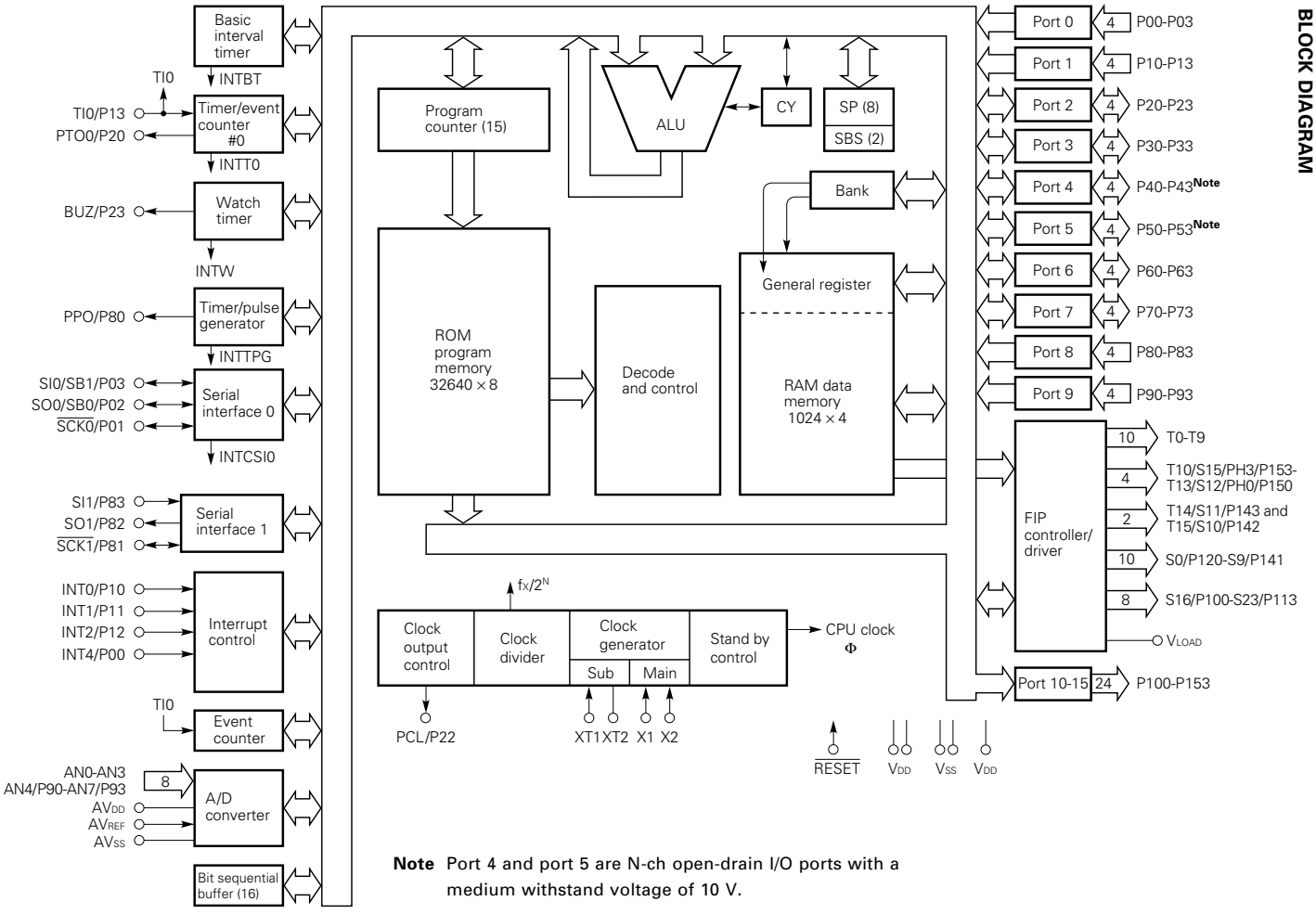
| Item | Function |
|---|---|
| On-chip memory | ROM: 32640 × 8 bits, RAM: 1024 × 4 bits |
| I/O lines<br>(Excluding pins dedicated to FIP) | 64 lines<br>• Input : 16 lines<br>• I/O : 24 lines<br>• Output : 24 lines |
| Instruction cycle | • 0.67 μs/1.33 μs/2.67 μs/10.7 μs (at 6.0 MHz)<br>• 0.95 μs/1.91 μs/3.82 μs/15.3 μs (at 4.19 MHz)<br>• 122 μs (at 32.768 kHz) |
| Fluorescent display tube (FIP) controller/driver | • Number of segments : 9 to 24 segments<br>• Number of digits : 9 to 16 digits<br>• Dimmer function : 8 levels<br>• Pull-down resistors provided by mask option<br>• Key scan interrupt generator |
| Timer/counter | 5 channels<br>• Basic interval timer : Usable as watchdog timer<br>• Timer/event counter<br>• Clock timer : With buzzer output function<br>• Timer/pulse generator : With 14-bit PWM output function<br>• Event counter |
| Serial interface | 2 channels<br>• SBI or 3-wire mode<br>• 3-wire mode |
| Interrupt | • Allows multiple hardware interrupts.<br>• External interrupts: 3<br>  • Detection of both edges<br>  • Detection edge programmable (with noise elimination)<br>  • Detection edge programmable<br>• External test input : 1<br>  • Rising edge detection<br>• Internal interrupts : 5<br>  • Timer/pulse generator<br>  • Timer/event counter<br>  • Basic interval timer<br>  • Serial interface #0<br>  • For key scanning<br>• Internal test inputs: 2<br>  • Clock timer<br>  • Serial interface #1 |
| System clock oscillator | • Main system clock : 6.0 MHz, 4.19 MHz<br>• Subsystem clock : 32.768 kHz, standard |
| Mask option | • High-voltage port : Pull-down resistor or open-drain output<br>• Ports 4 and 5 : Pull-up resistor<br>• Port 7 : Pull-down resistor |
| Operating temperature | -40 to +85 °C |
| Operating voltage | 2.7 to 6.0 V (Data held in standby mode: 2.0 to 6.0 V) |
| Package | 94-pin plastic QFP (20 × 20 mm) |

## PIN CONFIGURATION




**Caution Be sure to supply power to the $AV_{DD}$, $V_{DD}$, $V_{SS}$, and $AV_{SS}$ pins (pins 3, 4, 5, 11, 30, 48, 65, and 87).**

**Remark** IC: Internally connected pin (to be grounded)




**Note** Port 4 and port 5 are N-ch open-drain I/O ports with a medium withstand voltage of 10 V.

## CONTENTS

# 1. PIN FUNCTIONS

## 1.1 PORT PINS (1/2)

| Pin name | I/O | Also used as | Function | 8-bit I/O | When reset | I/O[Note 1] circuit type |
|---|---|---|---|---|---|---|
| P00 | I | INT4 | 4-bit input port (port 0). For P01 to P03, pull-up resistors can be provided by software in units of 3 bits. | × | Input | Ⓑ |
| P01 | | $\overline{SCK0}$ | | | | Ⓕ- A |
| P02 | | SO0/SB0 | | | | Ⓕ- B |
| P03 | | SI0/SB1 | | | | M - C |
| P10 | I | INT0 | With noise elimination function | × | Input | Ⓑ- C |
| P11 | | INT1 | 4-bit input port (port 1). Pull-up resistors can be provided by software in units of 4 bits. | | | |
| P12 | | INT2 | | | | |
| P13 | | TI0 | | | | |
| P20 | I/O | PTO0 | 4-bit I/O port (port 2). Pull-up resistors can be provided by software in units of 4 bits. | × | Input | E - B |
| P21 | | – | | | | |
| P22 | | PCL | | | | |
| P23 | | BUZ | | | | |
| P30[Note 2] | I/O | – | Programmable 4-bit I/O port (port 3). Input/output can be specified bit by bit. Pull-up resistors can be provided by software in units of 4 bits. | × | Input | E - C |
| P31[Note 2] | | – | | | | |
| P32[Note 2] | | – | | | | |
| P33[Note 2] | | – | | | | |
| P40-P43[Note 2] | I/O | – | N-ch open-drain 4-bit I/O port (port 4). A pull-up resistor can be provided bit by bit (mask option). Withstand voltage is 10 V in open-drain mode. | ○ | High level (when a pull-up resistor is provided) or high impedance | M |
| P50-P53[Note 2] | I/O | – | N-ch open-drain 4-bit I/O port (port 5). A pull-up resistor can be provided bit by bit (mask option). Withstand voltage is 10 V in open-drain mode. | | High level (when a pull-up resistor is provided) or high impedance | M |
| P60 | I/O | – | Programmable 4-bit I/O port (port 6). Input/output can be specified bit by bit. Pull-up resistors can be provided by software in units of 4 bits. | ○ | Input | E - C |
| P61 | | – | | | | |
| P62 | | – | | | | |
| P63 | | – | | | | |
| P70 | I/O | – | 4-bit I/O port (port 7). A pull-down resistor can be provided bit by bit (mask option). | | $V_{SS}$ level (when a pull-down resistor is provided) or high impedance | V |
| P71 | | – | | | | |
| P72 | | – | | | | |
| P73 | | – | | | | |

**Notes 1.** The circuits enclosed in circles have a Schmitt-triggered input.

**2.** An LED can be driven directly.

### 1.1 PORT PINS (2/2)

| Pin name | I/O | Also used as | Function | 8-bit I/O | When reset | I/O[Note] circuit type |
|---|---|---|---|---|---|---|
| P80 | I/O | PPO | 4-bit input port (port 8). | × | Input | A |
| P81 | I/O | $\overline{SCK1}$ | | | | Ⓕ |
| P82 | I/O | SO1 | | | | E |
| P83 | I | SI1 | | | | Ⓑ |
| P90 | I | AN4 | 4-bit input port (port 9) | × | Input | Y - A |
| P91 | | AN5 | | | | |
| P92 | | AN6 | | | | |
| P93 | | AN7 | | | | |
| P100 | O | S16 | P-ch open-drain, 4-bit high-voltage output port. Pull-down resistors can be provided (mask option). | ❍ | $V_{LOAD}$ level (when pull-down resistor to $V_{LOAD}$ is provided), $V_{SS}$ level (when pull-down resistor to $V_{SS}$ is provided), or high-impedance | I - F |
| P101 | | S17 | | | | |
| P102 | | S18 | | | | |
| P103 | | S19 | | | | |
| P110 | O | S20 | P-ch open-drain, 4-bit high-voltage output port. Pull-down resistors can be provided (mask option). | | | |
| P111 | | S21 | | | | |
| P112 | | S22 | | | | |
| P113 | | S23 | | | | |
| P120 | O | S0 | P-ch open-drain, 4-bit high-voltage output port. Pull-down resistors can be provided (mask option). | ❍ | $V_{LOAD}$ level (when pull-down resistor to $V_{LOAD}$ is provided) or high-impedance | I - C |
| P121 | | S1 | | | | |
| P122 | | S2 | | | | |
| P123 | | S3 | | | | |
| P130 | O | S4 | P-ch open-drain, 4-bit high-voltage output port. Pull-down resistors can be provided (mask option). | | | |
| P131 | | S5 | | | | |
| P132 | | S6 | | | | |
| P133 | | S7 | | | | |
| P140 | O | S8 | P-ch open-drain, 4-bit high-voltage output port. Pull-down resistors can be provided (mask option). P142 and P143 can drive LED directly. | ❍ | | |
| P141 | | S9 | | | | |
| P142 | | S10/T15 | | | | |
| P143 | | S11/T14 | | | | |
| P150 | O | S12/T13/PH0 | P-ch open-drain, 4-bit high-voltage output port. Pull-down resistors can be provided (mask option). LED can be driven directly. | | | |
| P151 | | S13/T12/PH1 | | | | |
| P152 | | S14/T11/PH2 | | | | |
| P153 | | S15/T10/PH3 | | | | |
| PH0 | O | S12/T13/P150 | P-ch open-drain, 4-bit high-voltage output port. Pull-down resistors can be provided (mask option). | × | | |
| PH1 | | S13/T12/P151 | | | | |
| PH2 | | S14/T11/P152 | | | | |
| PH3 | | S15/T10/P153 | | | | |

**Note** The circuits enclosed in circles have a Schmitt-triggered input.

### 1.2 NON-PORT PINS (1/2)

| Pin name | I/O | Also used as | Function | | When reset | I/O circuit type |
|---|---|---|---|---|---|---|
| T0-T9 | O | – | Output pins for FIP controller/ driver. Allows pull-down resistor to be provided (mask option). | High-voltage, large-current output for digit output | $V_{LOAD}$ level (when pull-down resistor to $V_{LOAD}$ is provided) or high-impedance | I - C |
| T10/S15-T13/S12 | | PH3/P153-PH0/P150 | | High-voltage, large-current output usable for digit/segment output as well. Any unused pins can be used for port H. Usable for port 15 in static mode. | | |
| T14/S11 | | P143 | | High-voltage, large-current output usable for digit/segment output as well. Usable for port 14 in static mode. | | |
| T15/S10 | | P142 | | | | |
| S0-S3 | | P120-P123 | | High-voltage output for segment output. Usable for port 12 to port 14 in static mode. | | |
| S4-S7 | | P130-P133 | | | | |
| S8 | | P140 | | | | |
| S9 | | P141 | | | | |
| S16-S19 | | P100-P103 | | High-voltage output for segment output. Usable for port 10 and port 11 in static mode. | $V_{LOAD}$ level (when pull-down resistor to $V_{LOAD}$ is provided), $V_{SS}$ level (when pull-down resistor to $V_{SS}$ is provided), or high-impedance | I - F |
| S20-S23 | | P110-P113 | | | | |

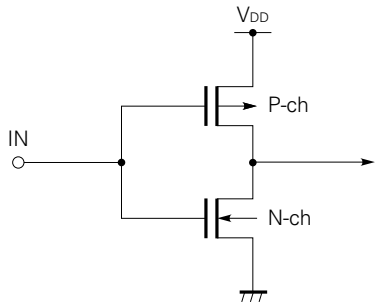
### 1.2 NON-PORT PINS (2/2)

| Pin name | I/O | Also used as | Function | | When reset | I/O[Note] circuit type |
|---|---|---|---|---|---|---|
| TI0 | I | P13 | External event pulse input for timer/event counter #0 and event counter #1. | | – | Ⓑ - C |
| PTO0 | O | P20 | Timer/event counter output | | Input | E - B |
| PCL | O | P22 | Clock output | | Input | E - B |
| BUZ | O | P23 | Fixed frequency output (for buzzer or system clock trimming) | | Input | E - B |
| $\overline{SCK0}$ | I/O | P01 | Serial clock I/O | | Input | Ⓕ - A |
| SO0/SB0 | I/O | P02 | Serial data output or serial bus I/O | | Input | Ⓕ - B |
| SI0/SB1 | I/O | P03 | Serial data input or serial bus I/O | | Input | M - C |
| INT4 | I | P00 | Edge detection vectored interrupt input (Either a rising or falling edge is detected.) | | – | Ⓑ |
| INT0 | I | P10 | Edge detection vectored interrupt input (The edge to be detected is selectable.) | Synchronous | – | Ⓑ - C |
| INT1 | | P11 | | Asynchronous | | |
| INT2 | I | P12 | Edge detection testable input (An rising edge is detected.) | Asynchronous | – | Ⓑ - C |
| $\overline{SCK1}$ | I/O | P81 | Serial clock I/O | | Input | Ⓕ |
| SO1 | O | P82 | Serial data output | | Input | E |
| SI1 | I | P83 | Serial data input | | Input | Ⓑ |
| AN0-AN3 | I | – | Analog input to A/D converter | | – | Y |
| AN4-AN7 | | P90-P93 | | | | Y - A |
| $AV_{DD}$ | – | – | Power supply for A/D converter | | – | – |
| $AV_{REF}$ | I | – | A/D converter reference voltage input | | – | Z |
| $AV_{SS}$ | – | – | A/D converter reference GND | | – | – |
| X1, X2 | I | – | Connection to a crystal/ceramic resonator for main system clock generation. When external clock is used, it is input to X1, and its inverted signal is input to X2. | | – | – |
| XT1 | I | – | Connection to a crystal resonator for subsystem clock generation. When external clock is used, it is input to XT1, and XT2 is left open. | | – | – |
| XT2 | – | | | | | |
| $\overline{RESET}$ | I | – | System reset input | | – | Ⓑ |
| PPO | O | P80 | Timer/pulse generator pulse output | | Input | – |
| $V_{DD}$ (3 pins) | – | – | Positive power supply | | – | – |
| $V_{SS}$ (2 pins) | – | – | GND potential | | – | – |
| $V_{LOAD}$ | – | – | Pull-down resistor connection for the FIP controller/driver, or power supply | | – | – |

**Note** The circuits enclosed in circles have a Schmitt-triggered input.

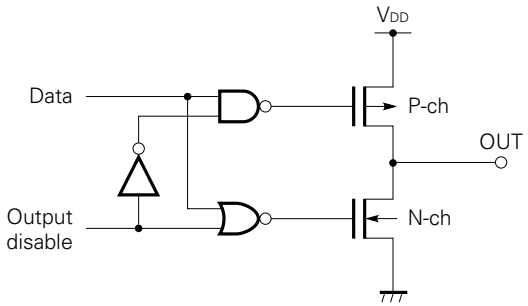
### 1.3 PIN INPUT/OUTPUT CIRCUITS (1/4)

**Type A**



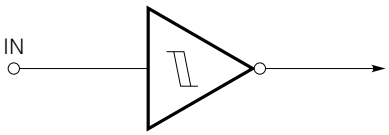

CMOS input buffer

**Type D**



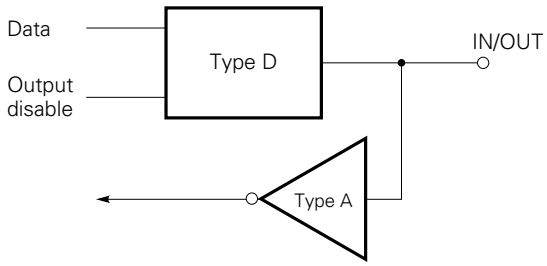

Push-pull output which can be set to high-impedance output (off for both P-ch and N-ch)

**Type B**



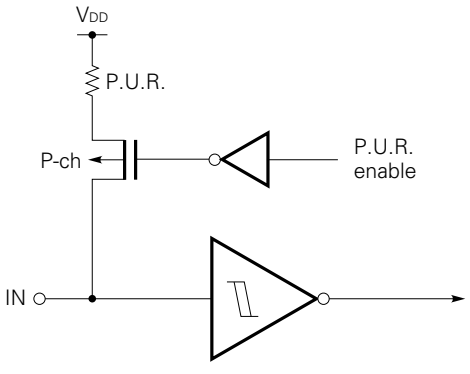

Schmitt trigger input with hysteresis

**Type E**



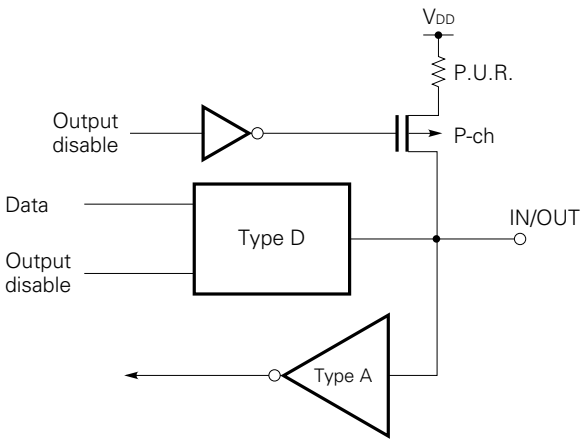

I/O circuit consisting of a push-pull output of type D and an input buffer of type A

**Type B - C**




P.U.R.: Pull-Up Resistor
Schmitt trigger input with hysteresis

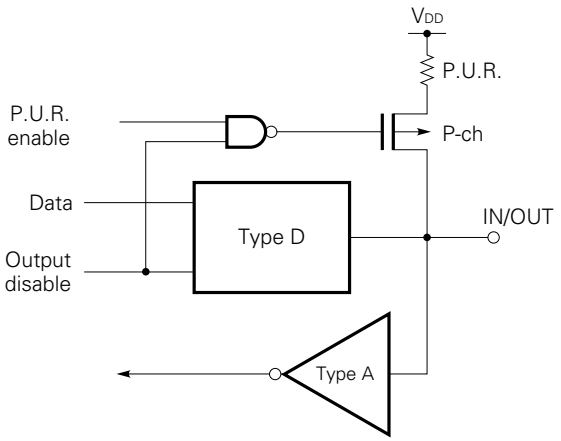
**Type E - B**




P.U.R.: Pull-Up Resistor

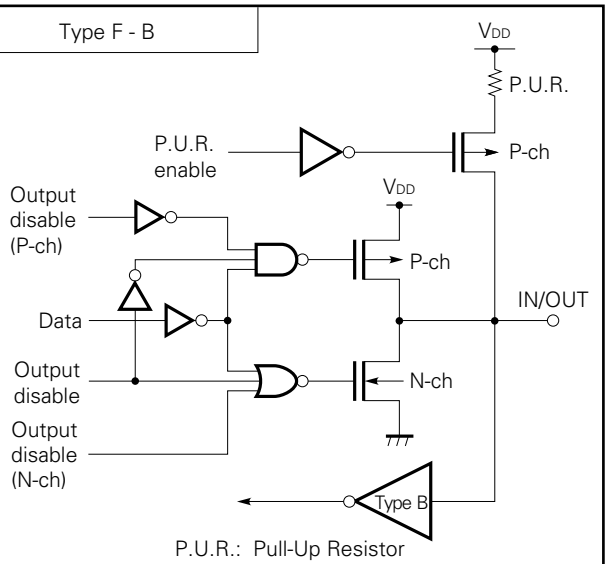
### 1.3 PIN INPUT/OUTPUT CIRCUITS (2/4)

**Type E - C**



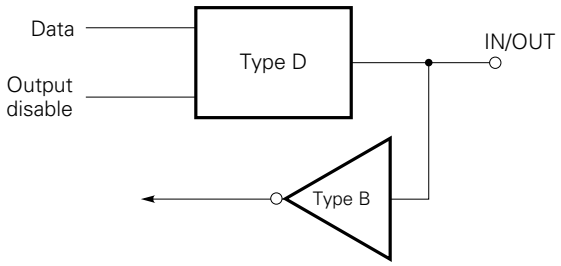

P.U.R.: Pull-Up Resistor

**Type F - B**



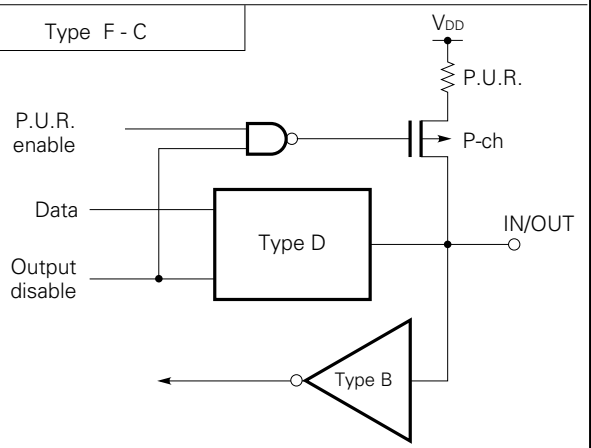

P.U.R.: Pull-Up Resistor

**Type F**



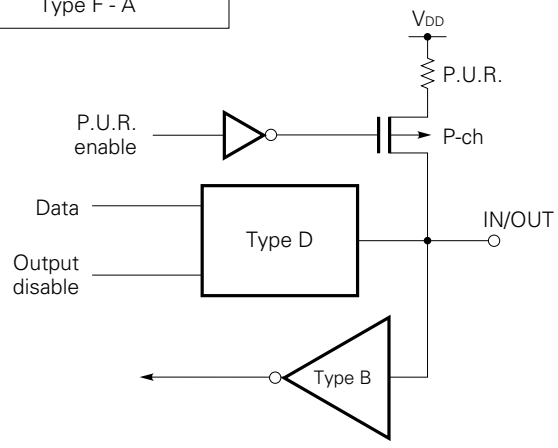

I/O circuit consisting of a push-pull output of type D and a Schmitt-triggered input of type B

**Type F - C**



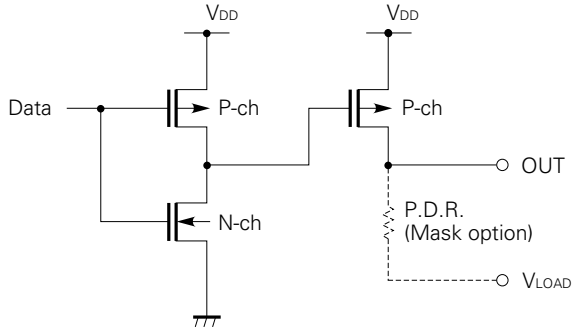

P.U.R.: Pull-Up Resistor

**Type F - A**




P.U.R.: Pull-Up Resistor

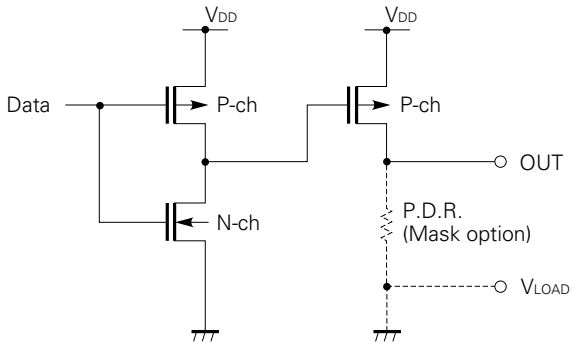
**Type I - C**




P.D.R.: Pull-Down Resistor

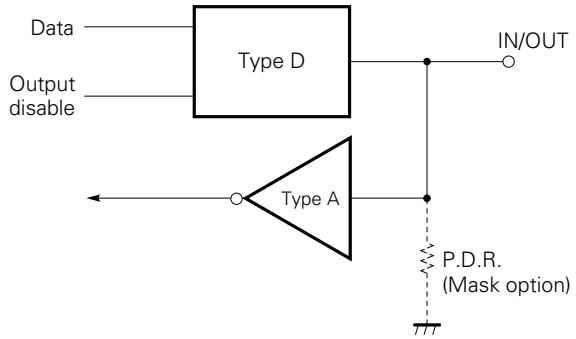
### 1.3 PIN INPUT/OUTPUT CIRCUITS (3/4)

Type I - F



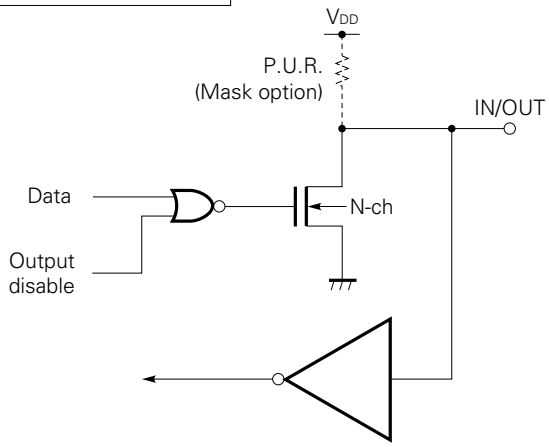

P.D.R.: Pull-Down Resistor

Type V



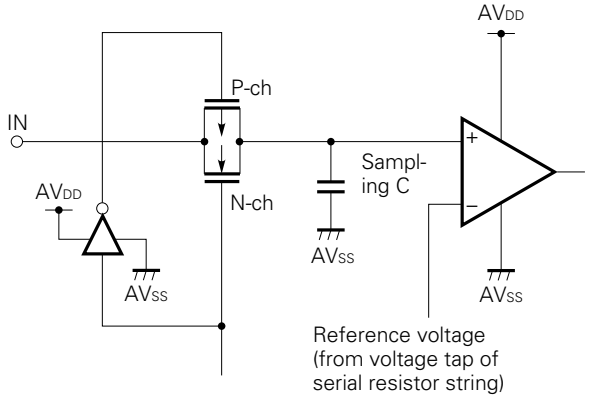

P.D.R.: Pull-Down Resistor

Type M

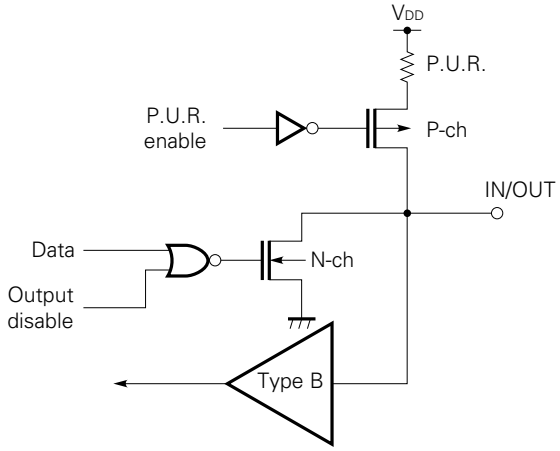



P.U.R.: Pull-Up Resistor

Type Y

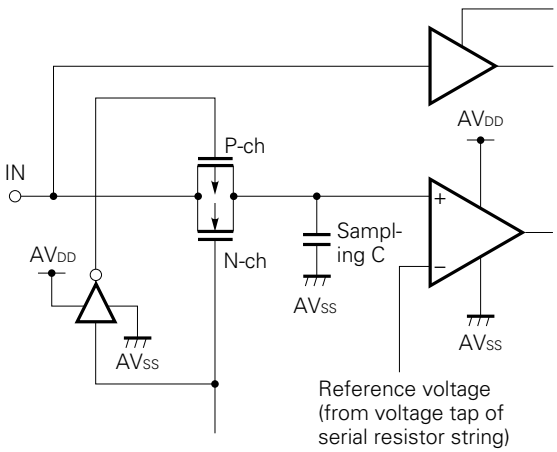



Type M - C

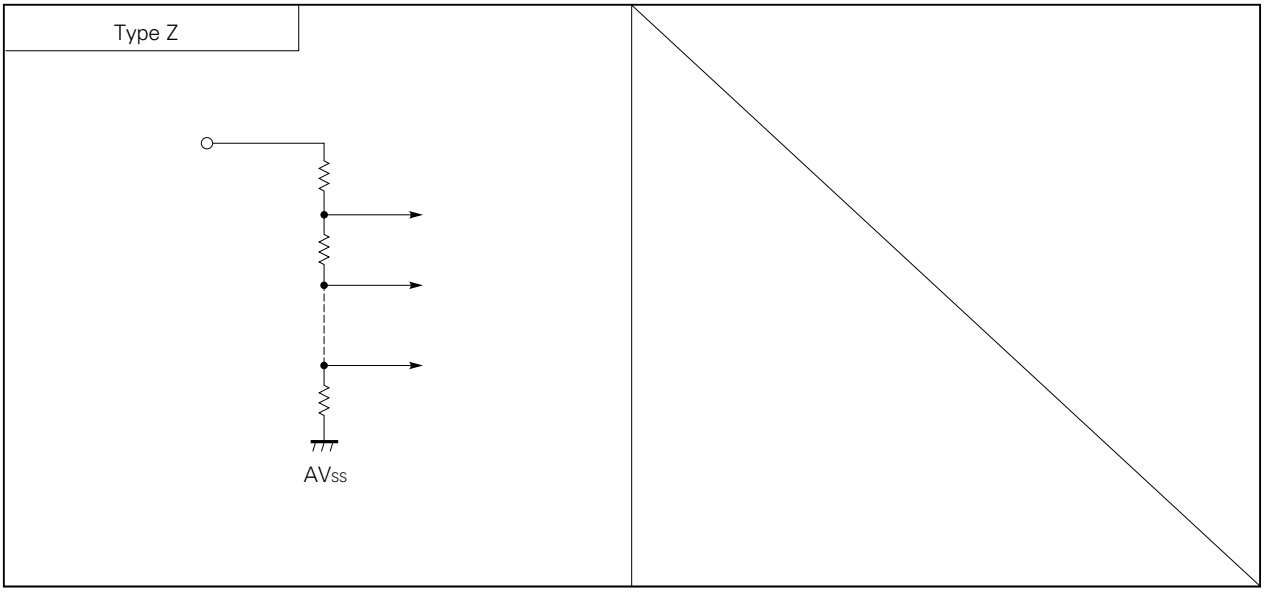



P.U.R.: Pull-Up Resistor

Type Y - A



### 1.3 PIN INPUT/OUTPUT CIRCUITS (4/4)



### 1.4 CONNECTION OF UNUSED μPD75238 PINS

| Pin name | Recommended connection |
|---|---|
| P00/INT4 | To be connected to $V_{SS}$ |
| P01/$\overline{SCK0}$ | To be connected to $V_{SS}$ or $V_{DD}$ |
| P02/SO0/SB0 | |
| P03/SI1/SB1 | |
| P10/INT0-P12/INT2 | To be connected to $V_{SS}$ |
| P13/TI0 | |
| P20/PTO0 | Input state : To be connected to $V_{SS}$ or $V_{DD}$<br>Output state: To be left open |
| P21 | |
| P22/PCL | |
| P23/BUZ | |
| P30-P33 | |
| P40-P43 | |
| P50-P53 | |
| P60-P63 | |
| P70-P73 | |
| P80/PPO | To be connected to $V_{SS}$ |
| P81/$\overline{SCK1}$ | |
| P82/SO1 | |
| P83/SI1 | |
| P90/AN4-P93/AN7 | |
| P100/S16-P103/S19 | To be left open |
| P110/S20-P113/S23 | |
| P120-P123 | |
| P130-P133 | |
| P140-P143 | |
| P150-P153 | |
| AN0-AN3 | To be connected to $V_{SS}$ |
| $AV_{REF}$ | |
| $AV_{DD}$ | To be connected to $V_{DD}$ |
| $AV_{SS}$ | To be connected to $V_{SS}$ |
| XT1 | To be connected to $V_{SS}$ or $V_{DD}$ |
| XT2 | To be left open |
| $V_{LOAD}$ | To be connected to $V_{SS}$ |

## 2. ARCHITECTURE AND MEMORY MAP OF THE μPD75238

The μPD75238 has three architectural features:

(a) Data memory bank configuration
(b) General register bank configuration
(c) Memory-mapped I/O

Each of these features is explained below.

### 2.1 DATA MEMORY BANK CONFIGURATION AND ADDRESSING MODES

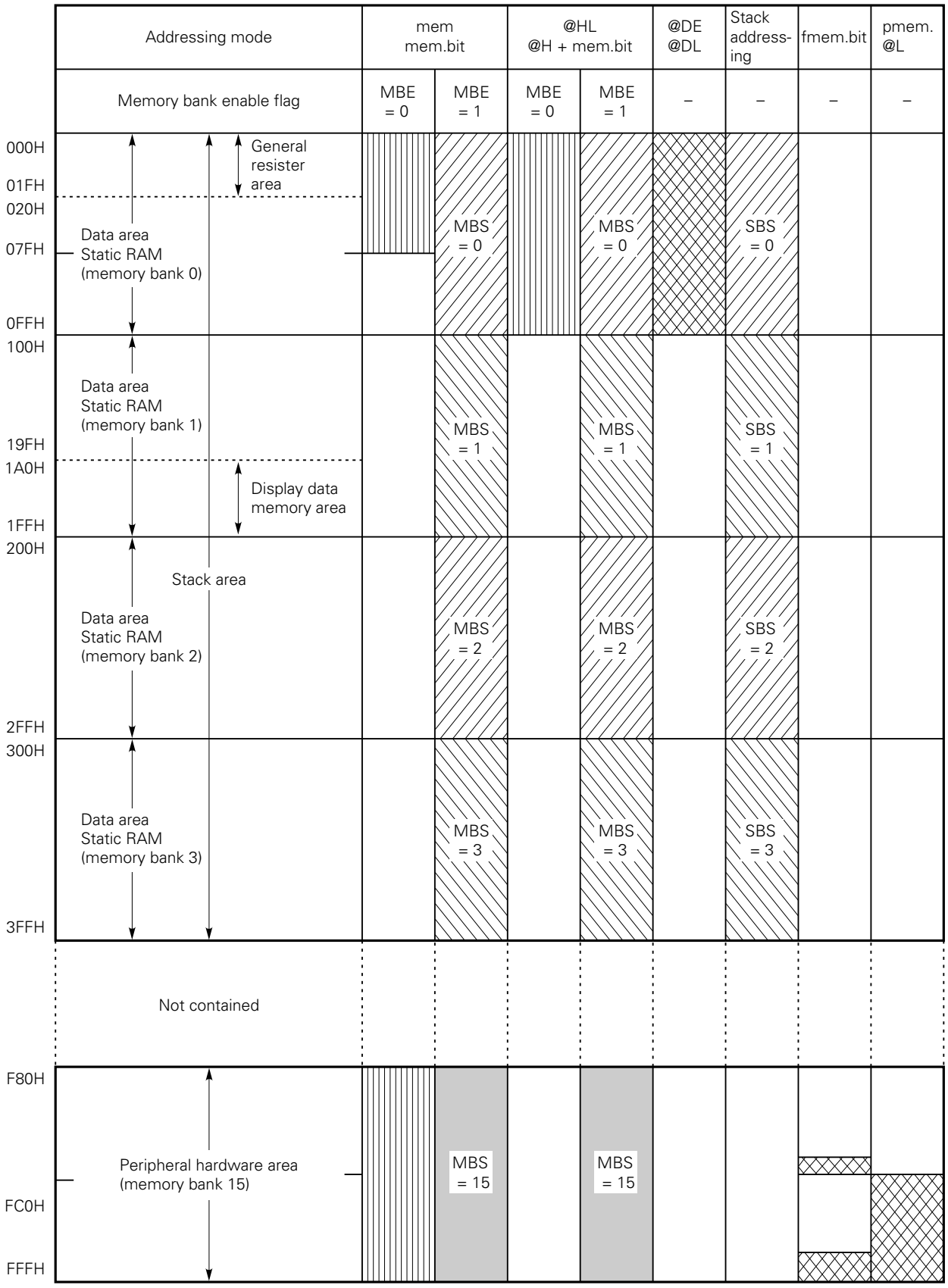
As shown in Fig. 2-1, the data memory space of the μPD75238 contains a static RAM (928 words × 4 bits) at addresses 000H to 19FH and 200H to 3FFH, a display data memory (96 words × 4 bits) at addresses 1A0H to 1FFH, and peripheral hardware (such as I/O ports and timers) at addresses F80H to FFFH. To address a 12-bit address in this data memory space, the μPD75238 uses such a memory bank configuration that the low-order eight bits are specified with an instruction directly or indirectly, and the high-order four bits are used to specify a memory bank (MB).

To specify a memory bank (MB), a memory bank enable flag (MBE) and memory bank select register (MBS) are contained, allowing the addressing indicated in Fig. 2-1 and Table 2-1. (The MBS is a register used to select a memory bank, and can be set to 0, 1, 2, 3, or 15. The MBE is a flag used to determine whether a memory bank selected using the MBS register is to be enabled. The MBE is automatically saved or restored at the time of interrupt processing or subroutine processing, so that it can be freely set in interrupt processing and subroutine processing.)

In addressing data memory space, the MBE is usually set to 1 (MBE = 1), and the static RAM in the memory bank specified by the MBS is operated. However, the MBE = 0 mode or the MBE = 1 mode can be selected for each step of program processing for more efficient programming.

| | Applicable program processing |
|---|---|
| MBE = 0 mode | • Interrupt processing<br>• Processing that repeats internal hardware and static RAM operations<br>• Subroutine processing |
| MBE = 1 mode | • Usual program processing |

**Fig. 2-1 Data Memory Organization and Addressing Range of Each Addressing Mode**




**Remark** — : Don't care

**Table 2-1 Addressing Modes**

| Addressing mode | Representation format | Specified address |
|---|---|---|
| 1-bit direct addressing | mem.bit | Bit specified by bit at the address specified by MB and mem. In this case:<br>When MBE = 0 and mem = 00H-7FH, MB = 0<br>When MBE = 0 and mem = 80H-FFH, MB = 15<br>When MBE = 1, MB = MBS |
| 4-bit direct addressing | mem | Address specified by MB and mem. In this case:<br>When MBE = 0 and mem = 00H-7FH, MB = 0<br>When MBE = 0 and mem = 80H-FFH, MB = 15<br>When MBE = 1, MB = MBS |
| 8-bit direct addressing | | Address specified by MB and mem (mem: even address). In this case:<br>When MBE = 0 and mem = 00H-7FH, MB = 0<br>When MBE = 0 and mem = 80H-FFH, MB = 15<br>When MBE = 1, MB = MBS |
| 4-bit register indirect addressing | @HL | Address specified by MB and HL. In this case, MB = MBE• MBS |
| | @HL+<br>@HL– | Address specified by MB and HL. In this case, MB = MBE• MBS.<br>HL+ automatically increments the L register after addressing.<br>HL- automatically decrements the L register after addressing. |
| | @DE | Address specified by DE in memory bank 0 |
| | @DL | Address specified by DL in memory bank 0 |
| 8-bit register indirect addressing | @HL | Address specified by MB and HL.<br>In this case, MB = MBE• MBS. Bit 0 of the L resister is ignored. |
| Bit manipulation addressing | fmem.bit | Bit specified by bit at the address specified by fmem. In this case:<br>fmem = FB0H-FBFH (interrupt-related hardware)<br>fmem = FF0H-FFFH (I/O port) |
| | pmem.@L | Bit specified by the low-order 2 bits of the L register at the address specified by the high-order 10 bits of pmem and the high-order 2 bits of the L register.<br>In this case, pmem = FC0H-FFFH |
| | @H+mem.bit | Bit specified by bit at the address specified by MB, H, and the low-order 4 bits of mem.<br>In this case, MB = MBE• MBS |
| Stack addressing | – | Address specified by SP in memory bank 0, 1, 2, and 3 selected by SBS |

As summarized in Table 2-1, the μPD75238 allows both direct and indirect addressing in data memory manipulation for 1-bit data, 4-bit data, and 8-bit data, so that very efficient and simple programming can be performed.

## 2.2 GENERAL REGISTER BANK CONFIGURATION

The μPD75238 contains four register banks, each consisting of eight general registers: X, A, B, C, D, E, H, and L. These registers are mapped to addresses 00H to 1FH in memory bank 0 of the data memory. (See **Fig. 2-2.**) To specify a general register bank, a register bank enable flag (RBE) and a register bank select register (RBS) are contained. The RBS is a register used to select a register bank, and the RBE is a flag used to determine whether a register bank selected using the RBS is to be enabled. The register bank (RB) enabled at instruction execution is determined as RB = RBE• RBS

As indicated in Table 2-2, the μPD75238 enables the user to create programs in a very efficient manner by selecting a register bank from the four register banks, depending on whether the processing is normal processing or interrupt processing. (The RBE is automatically saved and set at the time of interrupt processing, and is automatically restored upon completion of interrupt processing.)

**Table 2-2 Recommended Use of Register Banks with Normal Routines and Interrupt Routines**

| | |
|---|---|
| Normal processing | Use register banks 2 and 3 with RBE = 1. |
| Single interrupt processing | Use register bank 0 with RBE = 0. |
| Dual interrupt processing | Use register bank 1 with RBE = 1.<br>(In this case, the RBS needs to be saved and restored.) |
| Multiple (triple or more) interrupt processing | Save and restore the registers with PUSH or POP. |

The general registers allow transfers, comparisons, arithmetic/logical operations, and increments and decrements not only on a 4-bit basis, but also on an 8-bit basis with the XA, HL, DE, and BC register pairs. In this case, the register pairs of the register bank that has the inverted value of bit 0 of a register bank specified by RBE• RBS can be specified as XA', HL', DE', and BC', thus providing eight 8-bit registers. (See **Fig. 2-3**.)

**Fig. 2-2 General Register Configuration (4-Bit Processing)**

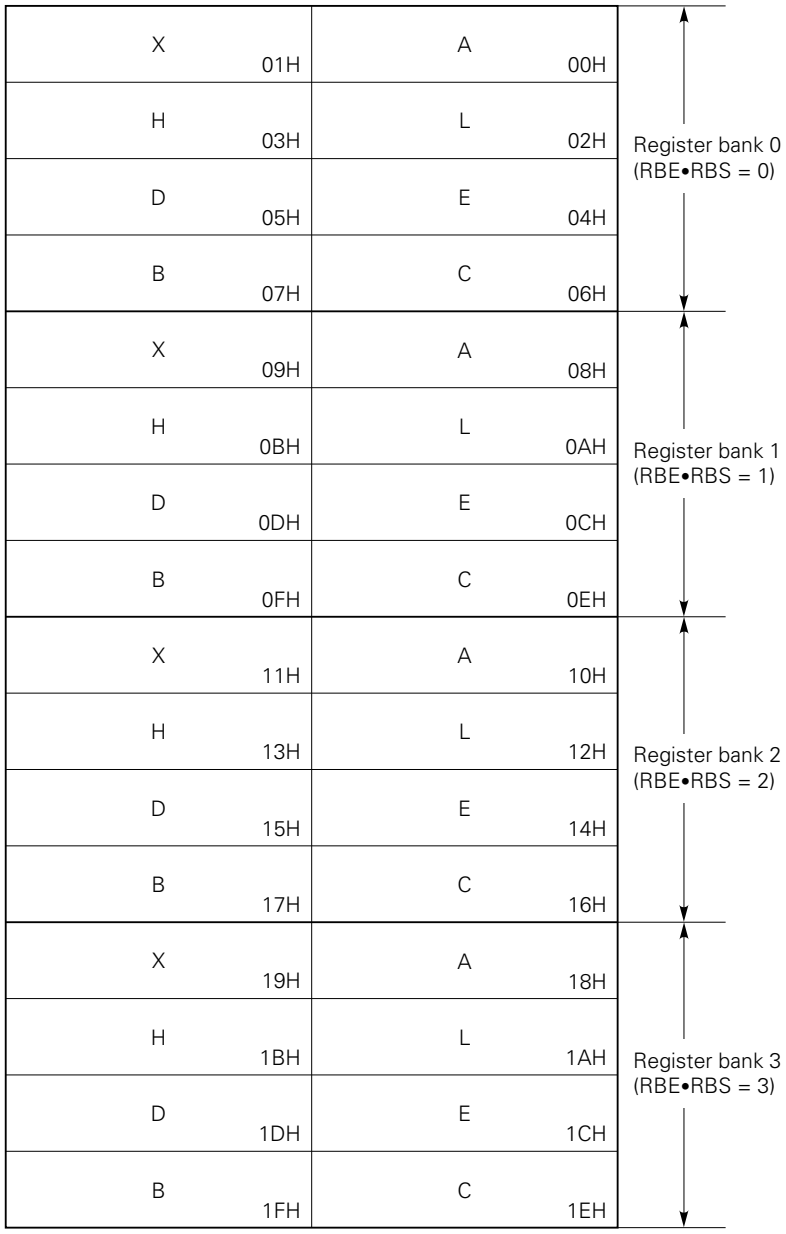
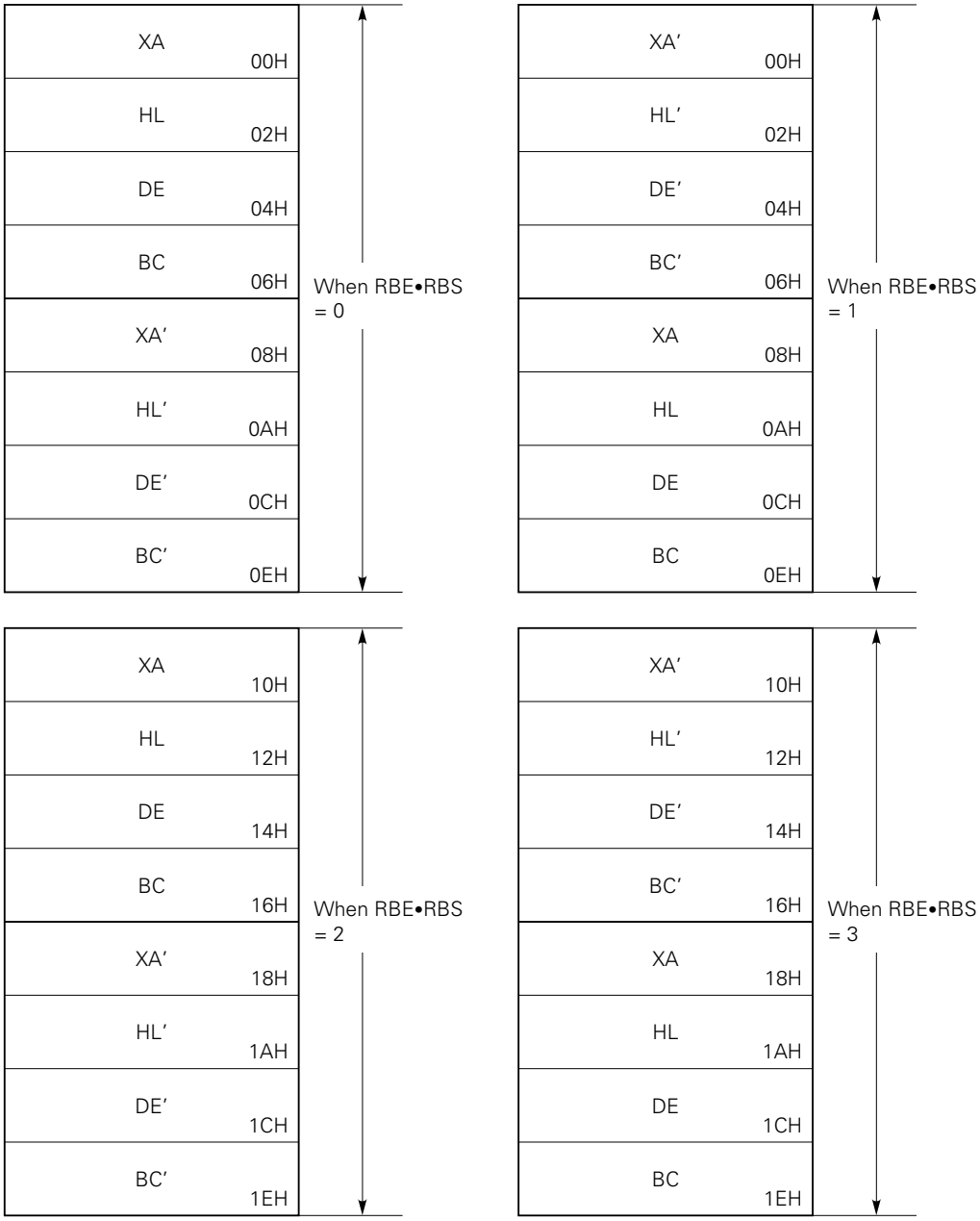


**Fig. 2-3 General Register Configuration (8-Bit Processing)**



### 2.3 MEMORY-MAPPED I/O

The μPD75238 employs memory-mapped I/O, which maps peripheral hardware such as timers and I/O ports to addresses F80H to FFFH in the data memory space as shown in Fig. 2-1. This means that there is no particular instruction to control peripheral hardware, but all peripheral hardware is controlled using memory manipulation instructions. (Some mnemonics for hardware control are available to make programs readable.)

To manipulate peripheral hardware, the addressing modes listed in Table 2-3 can be used.

The display data memory, key scan registers, and port H mapped to addresses 1A0H to 1FFH are to be manipulated by specifying memory bank 1.

**Table 2-3 Addressing Modes Applicable to Peripheral Hardware Mapped to Addresses F80H to FFFH**

| | Applicable addressing mode | Applicable hardware |
|---|---|---|
| Bit manipulation | Direct addressing mode specifying mem.bit with MBE = 0 or (MBE = 1, MBS = 15) | All hardware allowing bit manipulation |
| | Direct addressing mode specifying fmem.bit regardless of MBE and MBS setting | IST0, IST1, MBE, RBE, IE×××, IRQ×××, PORTn.0-3 |
| | Indirect addressing mode specifying pmem.@L regardless of MBE and MBS setting | PORTn. |
| 4-bit manipulation | Direct addressing mode specifying mem with MBE = 0 or (MBE = 1, MBS = 15) | All hardware allowing 4-bit manipulation |
| | Register indirect addressing mode specifying @HL with (MBE = 1, MBS = 15) | |
| 8-bit manipulation | Direct addressing mode specifying mem (even address) with MBE = 0 or (MBE = 1, MBS = 15) | All hardware allowing 8-bit manipulation addressing |
| | Register indirect addressing mode specifying @HL (with the L register containing an even number) with (MBE = 1, MBS = 15) | |

Table 2-4 summarizes the I/O map of the μPD75238.

The items in Table 2-4 have the following meanings:

- Symbol: Name representing the address of incorporated hardware, which can be coded in the operand field of an instruction
- R/W : Indicates whether the hardware allows read/write operation.
  - R/W: Both read and write operations possible
  - R : Read only
  - W : Write only
- Number of manipulatable bits:
  - Indicates the number of bits that can be processed in hardware manipulation
- Bit manipulation addressing:
  - Bit manipulation addressing applicable in hardware bit manipulation

**Table 2-4 μPD75238 I/O Map (1/4)**

| Address | Hardware name (symbol) b3 | b2 | b1 | b0 | R/W | Number of manipulatable bits 1 bit | 4 bits | 8 bits | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| F80H | Stack pointer (SP) | | | | R/W | – | – | ○ | | Bit 0 is always set to 0. |
| F82H | Resister bank select register (RBS) | | | | R Note 1 | – | ○ | ○ | | Note 2 ★ |
| F83H | Memory bank select register (MBS) | | | | | – | ○ | | | |
| F84H | Stack bank select register (SBS) | | | | R/W | – | ○ | – | | Bits 2 and 3 are always set to 0. |
| F85H | Basic interval timer mode register (BTM) | | | | W | Δ | ○ | – | mem.bit | Only bit 3 allows bit manipulation. |
| F86H | Basic interval timer (BT) | | | | R | – | – | ○ | | |
| F88H | Display mode register (DSPM) | | | | W | – | ○ | – | | |
| F89H | Dimmer select register (DIMS) | | | | W | – | ○ | – | | |
| F8AH | KSF | Digit select register (DIGS) | | | R/W | Δ | ○ | – | mem.bit | Only bit 3 allows a bit test. |
| F90H | Timer pulse generator mode register (TPGM) | | | | W | Δ | Δ | ○ | mem.bit | Only bit 3 allows bit manipulation. |
| F94H | Timer pulse generator modulo register L (MODL) | | | | R/W | – | – | ○ | | |
| F96H | Timer pulse generator modulo register H (MODH) | | | | R/W | – | – | ○ | | |
| F98H | Clock mode register (WM) | | | | W | – | – | ○ | | |
| FA0H | Timer/event counter 0 mode register (TM0) | | | | W | Δ | – | ○ | | Only bit 3 allows bit manipulation. |
| | | | | | | – | – | | | |
| FA2H | TOE0 | | | | W | ○ | – | – | | |
| FA4H | Timer/event counter 0 count register (T0) | | | | R | – | – | ○ | | |
| FA6H | Timer/event counter 0 modulo register (TMOD0) | | | | W | – | – | ○ | | |
| FA8H | Event counter mode register (TM1) | | | | W | Δ | – | ○ | | Only bit 3 allows bit manipulation. |
| | | | | | | – | – | | | |
| FABH | Gate control register (GATEC) | | | | W | – | ○ | – | | |
| FACH | Count register (T1) | | | | R | – | – | ○ | | |

**Notes 1.** For the SEL instruction, these registers are both readable and writable.

**2.** Can be operated separately as the RBS and MBS during 4-bit manipulation. Can also be operated as the BS during 8-bit manipulation.

**Table 2-4 μPD75238 I/O Map (2/4)**

| | Address | Hardware name (symbol) b3 | b2 | b1 | b0 | R/W | Number of manipulatable bits 1 bit | 4 bits | 8 bits | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | FB0H | IST1 | IST0 | MBE | RBE | R/W | ○ | ○ | ○ | fmem.bit | |
| | | Program status word (PSW) | | | | | – | – | | | |
| ★ | FB2H | Interrupt priority select register (IPS) | | | | W | ○ | ○ | – | | |
| | FB3H | Processor clock control register (PCC) | | | | W | ○ | ○ | | | |
| | FB4H | INT0 mode register (IM0) | | | | W | – | ○ | – | | Bit 2 is always set to 0. |
| | FB5H | INT1 mode register (IM1) | | | | W | – | ○ | | | Bits 1, 2, and 3 are always set to 0. |
| ★ | FB7H | System clock control register (SCC) | | | | W | ○ | – | – | | Only bits 0 and 3 allow bit manipulation. |
| | FB8H | IE4 | IRQ4 | IEBT | IRQBT | R/W | ○ | ○ | – | fmem.bit | |
| | FB9H | | | | EOT | R/W | ○ | ○ | | | |
| | FBAH | | | IEW | IRQW | R/W | ○ | ○ | – | | |
| | FBBH | IEKS | IRQKS | IETPG | IRQTPG | R/W | ○ | ○ | | | |
| | FBCH | | IRQT1 | IET0 | IRQT0 | R/W | ○ | ○ | – | | |
| | FBDH | | | IECSI0 | IRQCSI0 | R/W | ○ | ○ | | | |
| | FBEH | IE1 | IRQ1 | IE0 | IRQ0 | R/W | ○ | ○ | – | | |
| | FBFH | | | IE2 | IRQ2 | R/W | ○ | ○ | | | |
| | FC0H | Bit sequential buffer 0 (BSB0) | | | | R/W | ○ | ○ | ○ | | |
| | FC1H | Bit sequential buffer 1 (BSB1) | | | | R/W | ○ | ○ | | | |
| | FC2H | Bit sequential buffer 2 (BSB2) | | | | R/W | ○ | ○ | ○ | | |
| | FC3H | Bit sequential buffer 3 (BSB3) | | | | R/W | ○ | ○ | | | |
| | FC8H | | | CSIM11 | CSIM10 | W | – | – | ○ | | |
| ★ | FC9H | CSIE1 | | | | W | ○ | – | | | |
| | FCCH | Serial I/O shift register 1 (SIO1) | | | | R/W | – | – | ○ | | |

**Table 2-4 μPD75238 I/O Map (3/4)**

| Address | Hardware name (symbol) b3 | b2 | b1 | b0 | R/W | Number of manipulatable bits 1 bit | 4 bits | 8 bits | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| FD0H | Clock output mode register (CLOM) | | | | W | – | ○ | – | | |
| FD4H | Static mode register B (STATB) | | | | W | – | – | ○ | | |
| FD6H | Static mode register A (STATA) | | | | W | – | – | ○ | | |
| FD8H | SOC | EOC | | | R/W | Δ | – | ○ | | ADM is write only during 8-bit manipulation. ★ |
| | A/D conversion mode register (ADM) | | | | | – | – | | | |
| FDAH | SA register (SA) | | | | R | – | – | ○ | | |
| FDCH | Pull-up resistor specification register group A (POGA) | | | | W | – | – | ○ | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| FE0H | Serial operation mode register (CSIM0) | | | | W | – | – | ○ | | CSIM0 is write only during 8-bit manipulation. |
| | CSIE0 | COI | WUP | | R/W | ○ | ○ | | mem.bit | |
| FE2H | CMDD | RELD | CMDT | RELT | R/W | ○ | – | – | mem.bit | |
| | SBI control register (SBIC) | | | | | | | | | |
| | BSYE | ACKD | ACKE | ACKT | | | | | | |
| FE4H | Serial I/O shift register 0 (SIO0) | | | | R/W | – | – | ○ | | |
| FE6H | Slave address register (SVA) | | | | W | – | – | ○ | | |
| FE8H | PM33 | PM32 | PM31 | PM30 | W | – | – | ○ | | |
| | Port mode register group A (PMGA) | | | | | | | | | |
| | PM63 | PM62 | PM61 | PM60 | | | | | | |
| FECH | – | PM2 | – | – | W | – | – | ○ | | |
| | Port mode register group B (PMGB) | | | | | | | | | |
| | PM7 | – | PM5 | PM4 | | | | | | |

**Table 2-4 μPD75238 I/O Map (4/4)**

| Address | Hardware name (symbol) b3 | b2 | b1 | b0 | R/W | Number of manipulatable bits 1 bit | 4 bits | 8 bits | Bit manipulation addressing | Remarks |
|---|---|---|---|---|---|---|---|---|---|---|
| FF0H | Port 0 (PORT0) | | | | R | ○ | ○ | – | fmem.bit pmem.@L | |
| FF1H | Port 1 (PORT1) | | | | R | ○ | ○ | | | |
| FF2H | Port 2 (PORT2) | | | | R/W | ○ | ○ | – | | |
| FF3H | Port 3 (PORT3) | | | | R/W | ○ | ○ | | | |
| FF4H | Port 4 (PORT4) | | | | R/W | ○ | ○ | ○ | | |
| FF5H | Port 5 (PORT5) | | | | R/W | ○ | ○ | | | |
| FF6H | Port 6 (PORT6) | | | | R/W | ○ | ○ | ○ | | |
| FF7H | Port 7 (PORT7) | | | | R/W | ○ | ○ | | | |
| FF8H | Port 8 (PORT8) | | | | R | ○ | ○ | – | | |
| FF9H | Port 9 (PORT9) | | | | R | ○ | ○ | | | |
| FFAH | Port 10 (PORT10) | | | | W | ○ | ○ | ○ | | |
| FFBH | Port 11 (PORT11) | | | | W | ○ | ○ | | | |
| FFCH | Port 12 (PORT12) | | | | W | ○ | ○ | ○ | | |
| FFDH | Port 13 (PORT13) | | | | W | ○ | ○ | | | |
| FFEH | Port 14 (PORT14) | | | | W | ○ | ○ | ○ | | |
| FFFH | Port 15 (PORT15) | | | | W | ○ | ○ | | | |
| 1A0H+4n | Display data memory: S16-S23 (n = 0 to 15) | | | | R/W | ○ | ○ | ○ | mem.bit | |
| 1A1H+4n | | | | | R/W | ○ | ○ | | | |
| 1BEH | Key scan register (KS2) | | | | R/W | ○ | ○ | ○ | | |
| 1BFH | | | | | R/W | ○ | ○ | | | |
| 1C0H+4n | Display data memory: S0-S7 (n = 0 to 15) | | | | R/W | ○ | ○ | ○ | | |
| 1C1H+4n | | | | | R/W | ○ | ○ | | | |
| 1C2H+4n | Display data memory: S8-S15 (n = 0 to 15) | | | | R/W | ○ | ○ | ○ | | |
| 1C3H+4n | | | | | R/W | ○ | ○ | | | |
| 1FCH | Key scan register (KS0) | | | | R/W | ○ | ○ | ○ | | |
| 1FDH | | | | | R/W | ○ | ○ | | | |
| 1FEH | Key scan register (KS1) | | | | R/W | ○ | ○ | ○ | | |
| 1FFH | Port H (PORTH) | | | | R/W | ○ | ○ | ○ | | |

# 3. INTERNAL CPU FUNCTIONS

## 3.1 PROGRAM COUNTER (PC): 15 BITS

The program counter is a 15-bit binary counter for holding program memory address information.

**Fig. 3-1 Program Counter Format**

| PC14 | PC13 | PC12 | PC11 | PC10 | PC9 | PC8 | PC7 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Note that the reset start address must be set within a space of 16K bytes (0000H to 3FFFH). This is because a $\overline{\text{RESET}}$ input sets the low-order six bits of program memory address 0000H in PC13 to PC8, and the contents of address 0001H in PC7 to PC0 for initialization.

## 3.2 PROGRAM MEMORY (ROM): 32640 WORDS × 8 BITS

The program memory is a mask-programmable ROM with a configuration of 32640 words×8 bits for storing programs, table data, and so forth.

Program memory is addressed by the program counter. Table data can be referenced using the table reference instruction (MOVT).

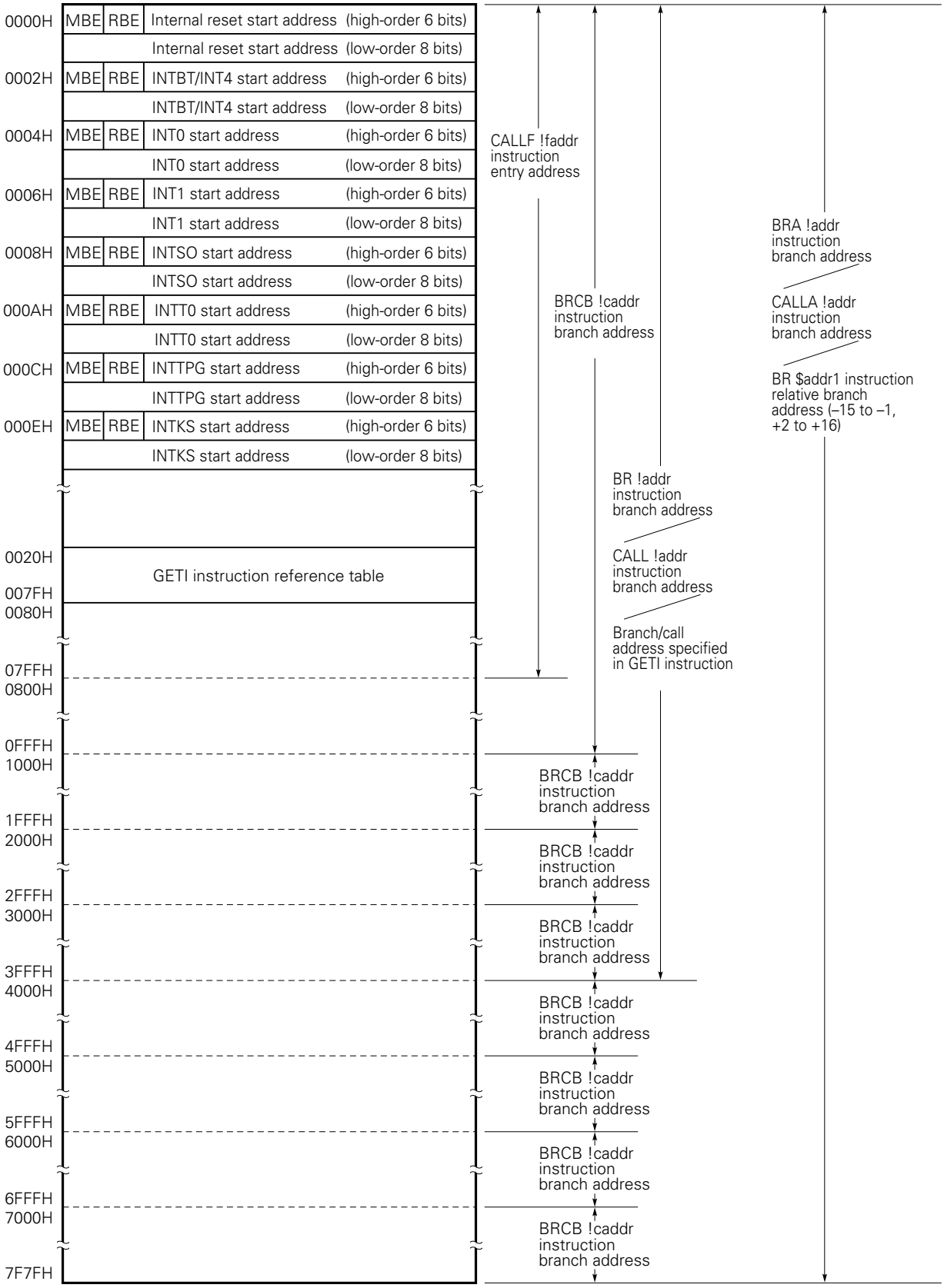
Fig. 3-2 shows the allowable branch address ranges for the branch instructions and subroutine call instructions. The whole-space branch instruction (BRA !addr1) and the whole-space call instruction (CALLA !addr1) allow a direct branch throughout the whole space 0000H to 7F7FH. The relative branch instruction (BR $addr) allows a branch to addresses (PC - 15 to PC - 1 and PC + 2 to PC + 16) regardless of block boundaries.

The program memory is located at addresses 0000H to 7F7FH containing the following specially assigned addresses. (All areas excluding 0000H and 0001H can be used as normal program memory.)

- 0000H to 0001H
  Vector table for holding the RBE and MBE setting values and program start address at the time of a $\overline{\text{RESET}}$ input. A reset start can be performed at an arbitrary address within a 16K-byte space (0000H to 3FFFH).
- 0002H to 000FH
  Vector address table for holding the RBE and MBE setting values and program start address at the time of each vectored interrupt occurrence. Interrupt processing can be started at an arbitrary address within a 16K-byte space (0000H to 3FFFH).
- 0020H to 007FH
  Table area referenced by the GETI instruction[Note]

**Note** The GETI instruction can represent an arbitrary 2-byte or 3-byte instruction or two 1-byte instructions in 1 byte, thus reducing the number of program bytes. (See **Section 8.1**.)

**Fig. 3-2 Program Memory Map**




**Caution The start address of an interrupt vector shown above consists of 14 bits. So, the start address must be set within a 16K-byte space (0000H to 3FFFH).**

**Remark** In addition to the above, the BR PCDE and BR PCXA instructions can cause a branch to an address with only the low-order 8 bits of the PC changed.

### 3.3 DATA MEMORY (RAM)

The data memory consists of a static RAM and peripheral hardware.

The static RAM consists of an area of 768 words × 4 bits in memory banks 0, 2, and 3, an area of 160 words × 4 bits in memory bank 1, and an area of 96 words × 4 bits in memory bank 1, which is used also as display data memory. The RAM is used for storing processing data, and is also used as stack memory for subroutine and interrupt execution.

To particular memory addresses, general registers, display data memory, and peripheral hardware (various registers) are mapped. Data in these areas are manipulated using general register manipulation instructions and memory manipulation instructions (See **Fig. 2-1**).

As a stack area, all addresses in memory banks 0, 1, 2, and 3 (000H to 3FFH) can be used.

The data memory has a configuration of 4 bits per address, but allows 8-bit memory manipulation instructions to be used for 8-bit oriented manipulation, and also allows bit manipulation instructions to be used for bit-by-bit manipulation. Even addresses are to be specified for 8-bit manipulation instructions.

Fig. 3-4 shows the organization of the display data memory area (1A0H to 1FFH).

**Fig. 3-3 Data Memory Map**

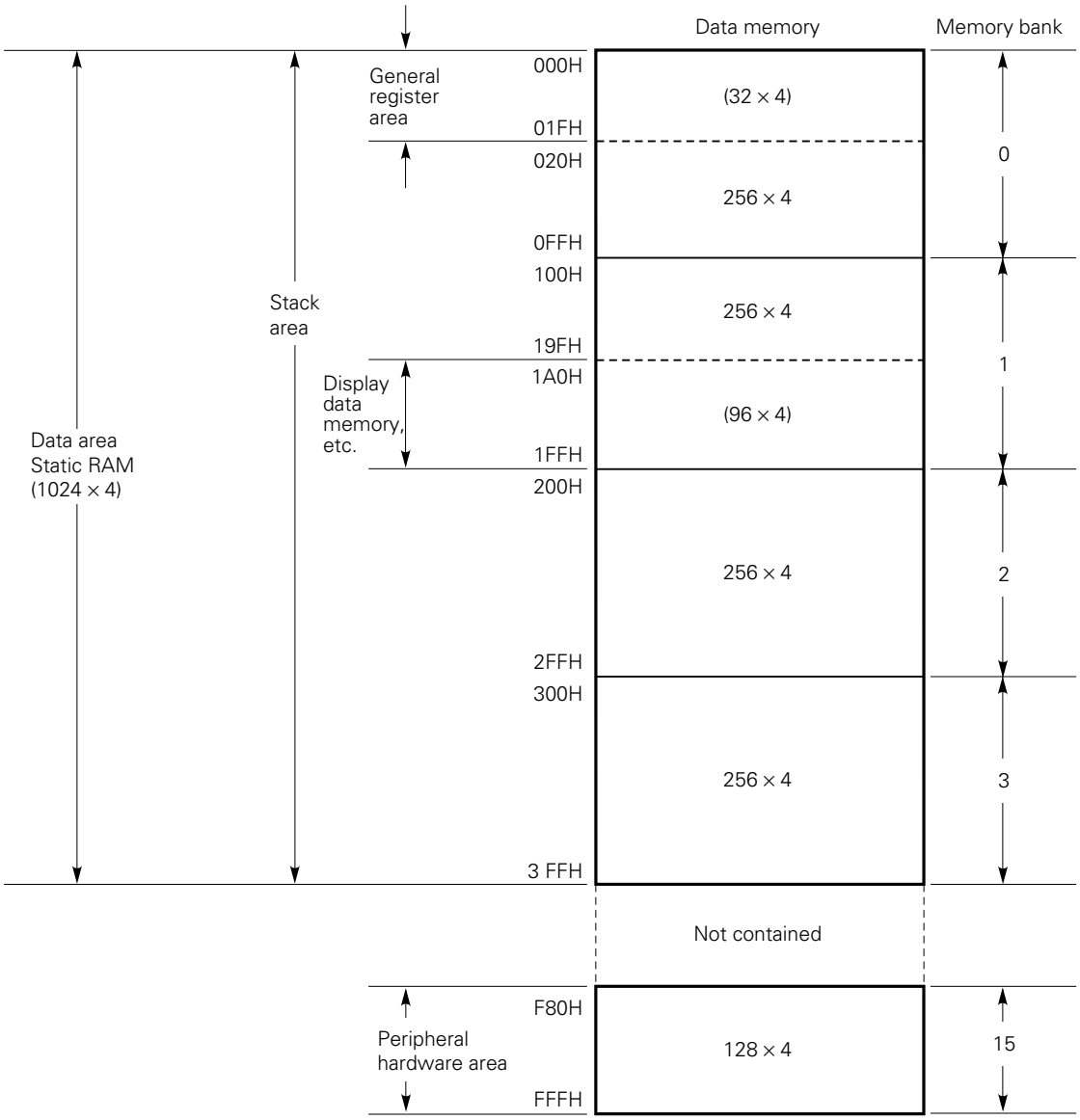


**Fig. 3-4 Display Data Memory Configuration**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1A1H | 1A0H | 1C3H | 1C2H | 1C1H | 1C0H |
| | | 1A3H | 1A2H | 1C7H | 1C6H | 1C5H | 1C4H |
| | | 1A5H | 1A4H | 1CBH | 1CAH | 1C9H | 1C8H |
| | | 1A7H | 1A6H | 1CFH | 1CEH | 1CDH | 1CCH |
| | | 1A9H | 1A8H | 1D3H | 1D2H | 1D1H | 1D0H |
| | | 1ABH | 1AAH | 1D7H | 1D6H | 1D5H | 1D4H |
| | | 1ADH | 1ACH | 1DBH | 1DAH | 1D9H | 1D8H |
| | | 1AFH | 1AEH | 1DFH | 1DEH | 1DDH | 1DCH |
| | | 1B1H | 1B0H | 1E3H | 1E2H | 1E1H | 1E0H |
| | | 1B3H | 1B2H | 1E7H | 1E6H | 1E5H | 1E4H |
| | | 1B5H | 1B4H | 1EBH | 1EAH | 1E9H | 1E8H |
| | | 1B7H | 1B6H | 1EFH | 1EEH | 1EDH | 1ECH |
| | | 1B9H | 1B8H | 1F3H | 1F2H | 1F1H | 1F0H |
| | | 1BBH | 1BAH | 1F7H | 1F6H | 1F5H | 1F4H |
| | | 1BDH | 1BCH | 1FBH | 1FAH | 1F9H | 1F8H |
| | | 1BFH | 1BEH(KS2) | 1FFH(PORTH) | 1FEH(KS1) | 1FDH | 1FCH(KS0) |
| Number of manipulatable bits | 1 bit | ○ | ○ | ○ | ○ | ○ | ○ |
| | 4 bits | ○ | ○ | ○ | ○ | ○ | ○ |
| | 8 bits | ○ | | ○ | | ○ | |

**Remarks 1.** KS0, KS1, and KS2 are key scan registers.

**2.** PORTH is the high-voltage, high-current output port, and is also used for digit output.

### 3.4 GENERAL REGISTERS: 8 × 4 BITS × 4 BANKS

The general registers are mapped to particular addresses in data memory. Four banks of registers are provided, with each bank consisting of eight 4-bit registers (B, C, D, E, H, L, X, A).

The register bank (RB) to be enabled at the time of instruction execution is determined by RB = RBE• RBS: (RBS = 0 to 3)

Each general register allows 4-bit manipulation. In addition, BC, DE, HL, or XA serves as a register pair for 8-bit manipulation. DL also makes a register pair as well as DE and HL; these three register pairs can be used as data pointers.

A general register area can be addressed and accessed as normal RAM, regardless of whether it is used as a register.

**Fig. 3-5 General Register Format**

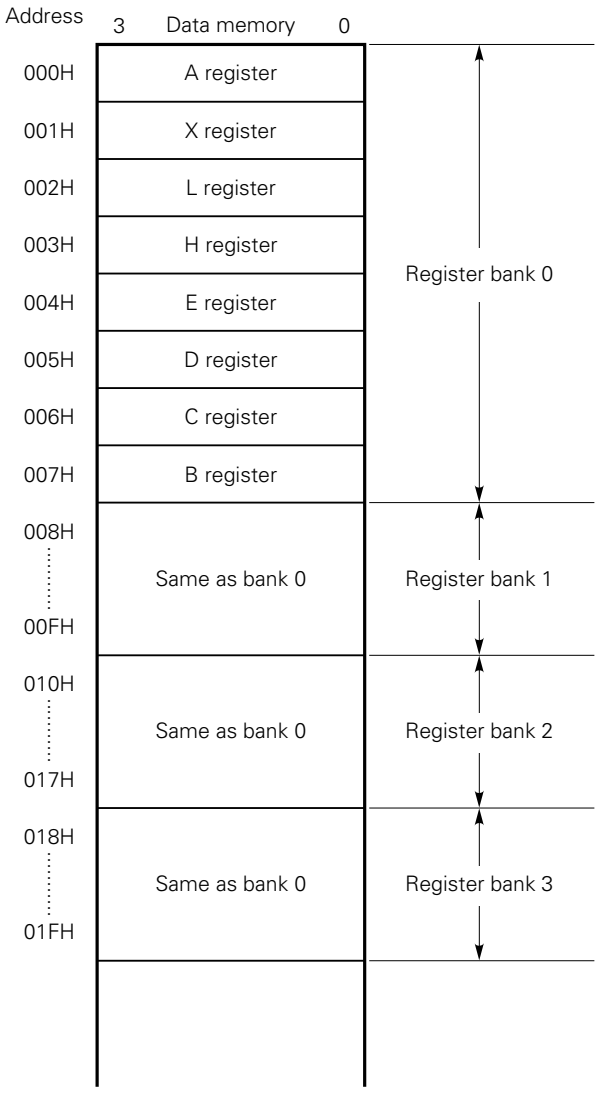
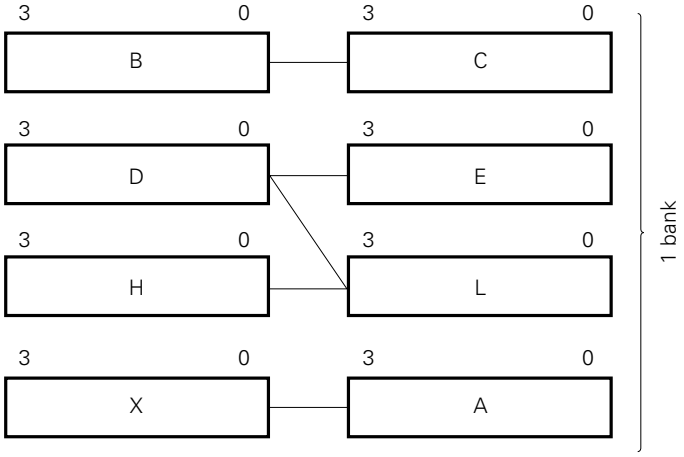



**Fig. 3-6 Register Pair Format**

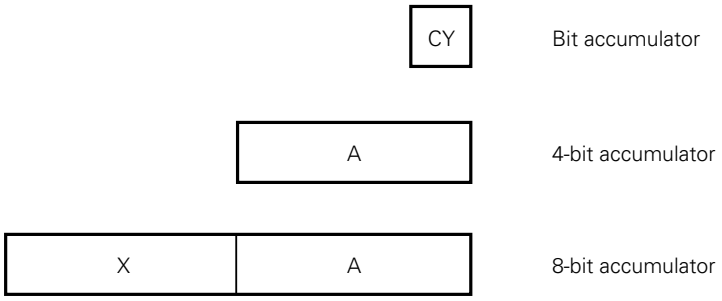


### 3.5 ACCUMULATORS

In the μPD75238, the A register and the XA register pair function as accumulators. The A register is mainly used for 4-bit data processing instructions, and the XA register pair is mainly used for 8-bit data processing instructions.

For a bit manipulation instruction, the carry flag (CY) functions as a bit accumulator.

**Fig. 3-7 Accumulators**




### 3.6 STACK POINTER (SP) AND STACK BANK SELECT REGISTER (SBS)

The μPD75238 uses static RAM as stack memory (LIFO scheme), and the 8-bit register holding the start address of the stack area is the stack pointer (SP).

The stack area is located at addresses 000H to 3FFH in memory banks 0, 1, 2, and 3. Either of the memory banks is selected according to the value of the 4-bit SBS.

The SP is decremented before a write (save) operation to stack memory, and is incremented after a read (restoration) operation from stack memory. The SBS is set with a 4-bit memory manipulation instruction. Note that the high-order two bits are always set to 00.

Fig. 3-9 and 3-10 show data saved to and restored from stack memory in these stack operations.

To place the stack area at a given location, the SP can be initialized with an 8-bit memory manipulation instruction, and the SBS can be initialized with a 4-bit memory manipulation instruction. Both can be read from as well.

**Table 3-1 Stack Area to Be Selected by the SBS**

| SBS | | Stack area |
|---|---|---|
| SBS1 | SBS2 | |
| 0 | 0 | Memory bank 0 |
| 0 | 1 | Memory bank 1 |
| 1 | 0 | Memory bank 2 |
| 1 | 1 | Memory bank 3 |

When the SP is initialized to 00H, a stack operation starts at the high-order address (nFFH) of memory bank (n: n = 0, 1, 2, or 3) specified with the SBS.

A stack area must be within the memory bank specified with the SBS. If a stack operation exceeds address n00H, the operation returns to address nFFH of the same bank. Stacking beyond memory bank boundaries is enabled only by resetting the SBS.

A $\overline{\text{RESET}}$ signal occurrence causes the contents of the SP and the SBS to be undefined, so that the SP must always be initialized to a desired value at the start of the program.

**Fig. 3-8 Stack Bank Select Register Format**

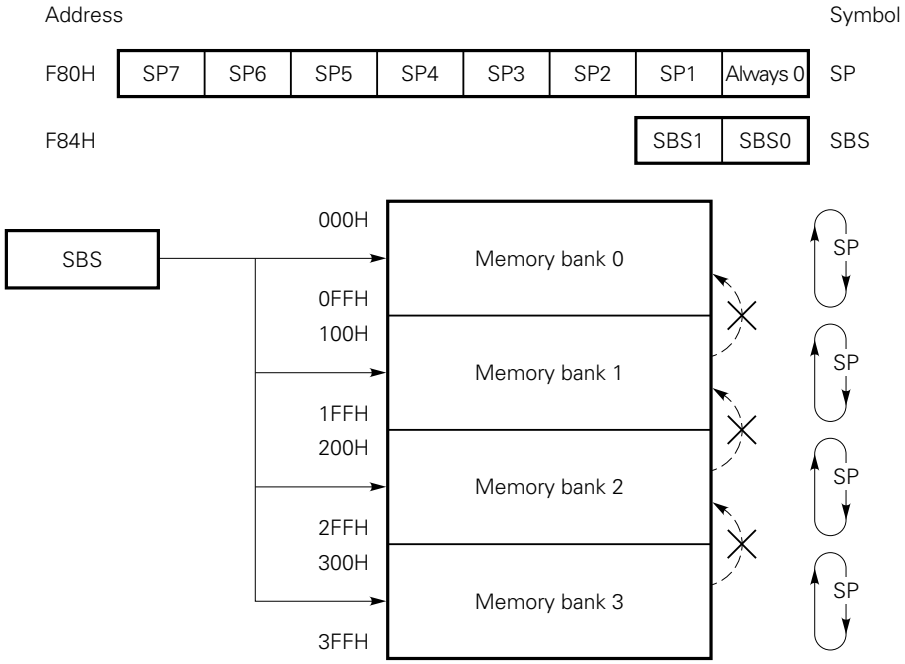


**Fig. 3-9 Data Saved to Stack Memory**

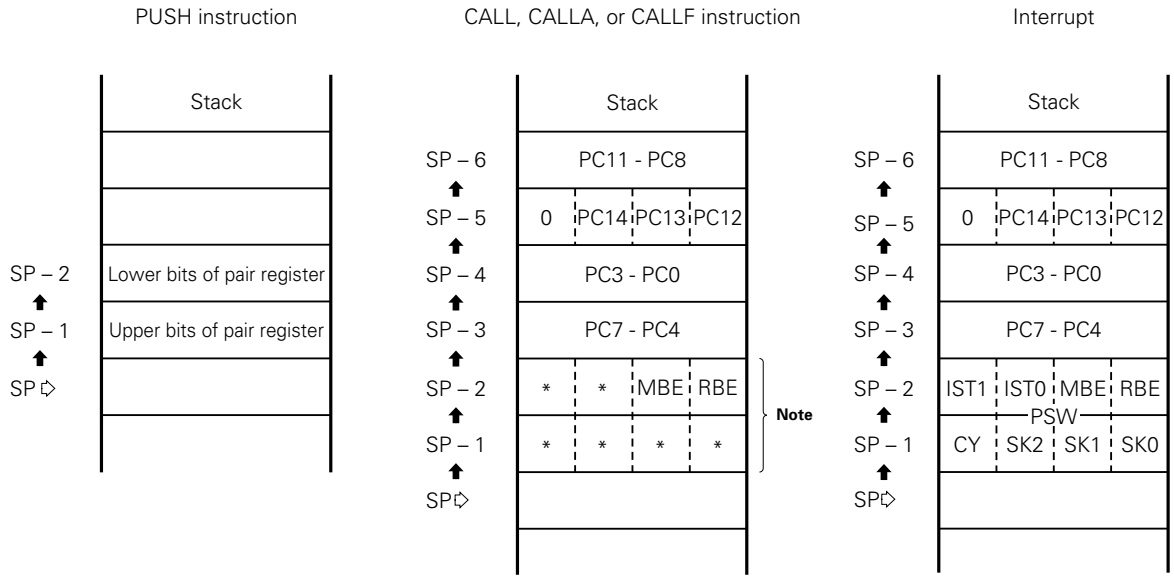
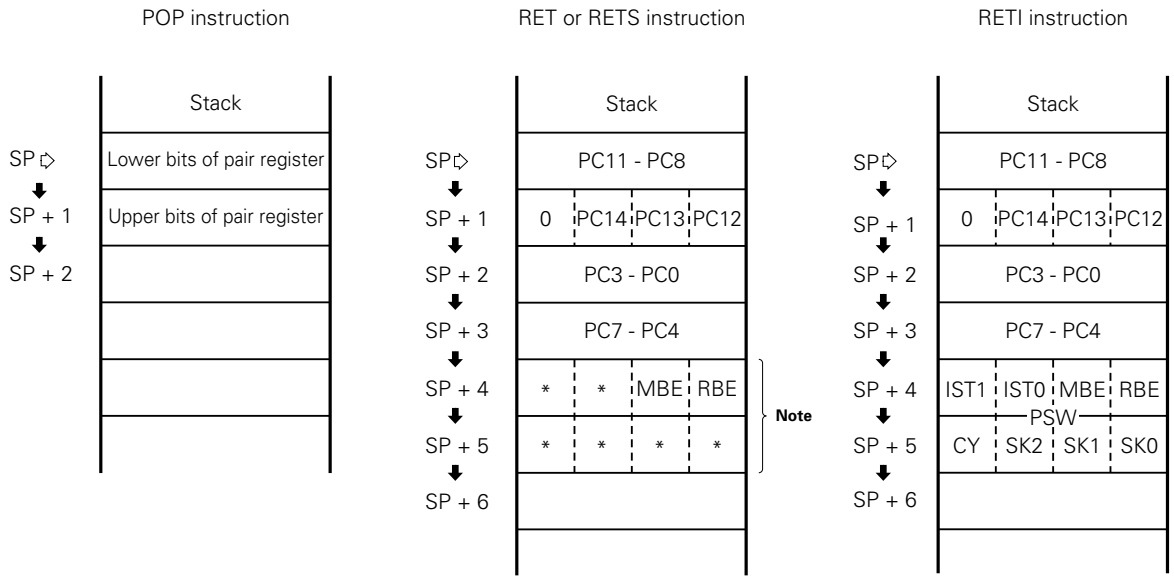



**Fig. 3-10 Data Restored from Stack Memory**




**Note** A PSW other than the MBE or RBE is not saved/restored.

**Remark** Data marked with * is undefined.

### 3.7 PROGRAM STATUS WORD (PSW): 8 BITS

The program status word (PSW) consists of various flags closely associated with processor operations.

The PSW is mapped to addresses FB0H and FB1H in the data memory space. The four bits at address FB0H can be manipulated with a memory manipulation instruction. Address FB1H cannot be manipulated with a normal data memory manipulation instruction.

**Fig. 3-11 Program Status Word Format**

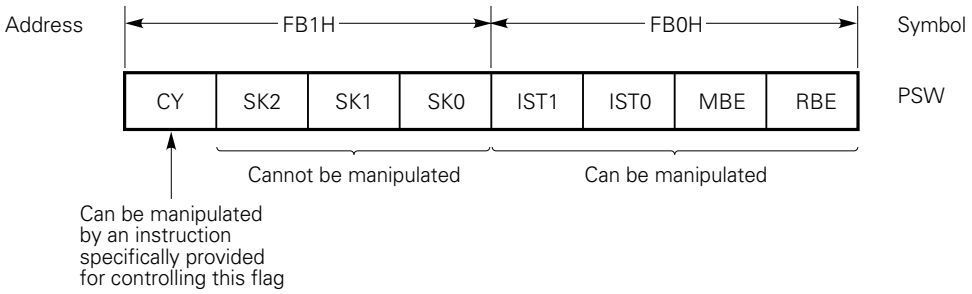



**Table 3-2 PSW Flags Saved/Restored in Stack Operation**

| | | Saved/restored flag |
|---|---|---|
| Save | When CALL, CALLA, or CALLF instruction is executed | MBE and RBE are saved. |
| | When hardware interrupt occurs | All PSW bits are saved. |
| Restore | When RET or RETS instruction is executed | MBE and RBE are restored. |
| | When RETI is executed | All PSW bits are restored. |

**(1) Carry flag (CY)**

The carry flag is a 1-bit flag used to store overflow or underflow occurrence information when an arithmetic operation with a carry (ADDC, SUBC) is executed.

The carry flag also has the function of a bit accumulator, and therefore can be used to store the result of a Boolean operation performed on the CY and bit at a specified data memory bit address.

The carry flag is manipulated using special instructions, independently of the other PSW bits.

A $\overline{\text{RESET}}$ signal occurrence causes the carry flag to be undefined.

**Table 3-3 Carry Flag Manipulation Instructions**

| | Instruction (mnemonic) | Carry flag operation/processing |
|---|---|---|
| Instruction dedicated to carry flag manipulation | SET1 CY<br>CLR1 CY<br>NOT1 CY<br>SKT CY | Sets CY to 1.<br>Clears CY to 0.<br>Inverts the contents of CY.<br>Skips if CY is set to 1. |
| Bit transfer instruction | MOV1 mem*.bit CY<br>MOV1 CY,mem*.bit | Transfers the contents of CY to a specified bit.<br>Transfers the contents of a specified bit to CY. |
| Bit Boolean instruction | AND1 CY,mem*.bit<br>OR1 CY,mem*.bit<br>XOR1 CY,mem*.bit | ANDs, ORs, or XORs CY with the contents of a specified bit, then sets the result in CY. |
| Interrupt handling | Interrupt execution | Saves CY and all other PSW bits to stack memory in parallel. |
| | RETI | Restores CY together with the other PSW bits from stack memory. |

**Remark** mem*.bit represents the following three addressing modes:

- fmem.bit
- pmem.@L
- @H+mem.bit

**(2) Skip flags (SK2, SK1, SK0)**

The skip flags are used to store skip status, and are automatically set or reset when the CPU executes an instruction.

The user cannot directly manipulate these flags as operands.

**(3) Interrupt status flag (IST1, IST0)**

The interrupt status flag is a 2-bit flag used to store the status of processing being performed. (For detailed information, see **Table 5-3**.)

**Table 3-4 Information Indicated by the Interrupt Status Flag**

| IST1 | IST0 | Status of processing being performed | Processing and interrupt control |
|---|---|---|---|
| 0 | 0 | Status 0 | Normal program processing is being performed.<br>Any interrupts are acceptable. |
| 0 | 1 | Status 1 | A lower- or higher-priority interrupt is being serviced.<br>Higher-priority interrupts are acceptable. |
| 1 | 0 | Status 2 | A higher-priority interrupt is being serviced.<br>No interrupts are acceptable. |
| 1 | 1 | — | Not to be set |

The interrupt priority control circuit (see **Fig. 5-1**) checks this flag to control multiple interrupts.
The contents of the IST1 and IST0 are saved as part of the PSW to stack memory if an interrupt is accepted, then are automatically set to a one-step higher status. The RETI instruction restores the contents present before an interrupt occurs.
The interrupt status flag can be manipulated using a memory manipulation instruction, and the status of processing being performed can be changed by program control.

**Caution The user must always disable interrupts with the DI instruction before manipulating this flag, and must enable interrupts with the EI instruction after manipulating this flag.**

**(4) Memory bank enable flag (MBE)**

The memory bank enable flag is a 1-bit flag used to specify the address information generation mode for the high-order four bits of a 12-bit data memory address.

When the MBE is set to 1, the data memory address space is expanded, allowing all data memory space to be addressed.

When the MBE is reset to 0, the data memory address space is fixed, regardless of MBS setting. (See **Fig. 2-1**.)

A $\overline{\text{RESET}}$ input automatically initializes the MBE by setting the MBE to the content of bit 7 at program memory address 0.

In vectored interrupt processing, the MBE is automatically set to the content of bit 7 in the vector address table for servicing the interrupt.

Usually, the MBE is set to 0 in interrupt processing, and static RAM in memory bank 0 is used.

**(5) Register bank enable flag (RBE)**

The register bank enable flag is a 1-bit flag used to determine whether to expand the general register bank configuration.

When the RBE is set to 1, a set of general registers can be selected from register banks 0 to 3, depending on the setting of the register bank select register (RBS).

When the RBE is reset to 0, register bank 0 is always selected as general registers, regardless of the setting of the RBS.

A $\overline{\text{RESET}}$ input automatically initializes the RBE by setting the RBE to the content of bit 6 at program memory address 0.

When a vectored interrupt occurs, the RBE is automatically set to the content of bit 6 in the vector address table for servicing the interrupt. Usually, the RBE is set to 0 in interrupt processing. Register bank 0 is used for 4-bit processing, and register banks 0 and 1 are used for 8-bit processing.

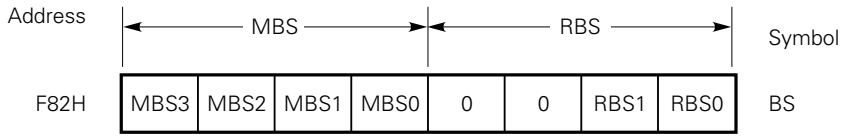
### 3.8 BANK SELECT REGISTER (BS)

The bank select register consists of a register bank select register (RBS) and memory bank select register (MBS), which specify a register bank and memory bank to be used, respectively.

The RBS and MBS are set using the SEL RBn instruction and SEL MBn instruction, respectively.

The contents of BS can be saved to or restored from a stack area eight bits at a time by using the PUSH BS/POP BS instruction.

**Fig. 3-12 Bank Select Register Format**




**(1) Memory bank select register (MBS)**

The memory bank select register is a 4-bit register used to store the high-order four bits of a 12-bit data memory address. The contents of this register specify a memory bank to be accessed. Memory banks 0, 1, 2, 3, and 15 can be specified.

The MBS is set with the SEL MBn instruction (n = 0, 1, 2, 3, 15)

Fig. 2-1 shows the range of addressing using MBE and MBS settings.

A $\overline{\text{RESET}}$ input initializes the MBS to 0.

**(2) Register bank select register (RBS)**

The register bank select register specifies a register bank to be used as general registers; a register bank can be selected from register banks 0 to 3.

The RBS is set with the SEL RBn instruction (n = 0 to 3).

A $\overline{\text{RESET}}$ input initializes the RBS to 0.

**Table 3-5 Register Bank to Be Selected with the RBE and RBS**

| RBE | RBS | | | | Register bank |
|---|---|---|---|---|---|
| | 3 | 2 | 1 | 0 | |
| 0 | 0 | 0 | × | × | Bank 0 is always selected. |
| 1 | 0 | 0 | 0 | 0 | Bank 0 is selected. |
| | | | 0 | 1 | Bank 1 is selected. |
| | | | 1 | 0 | Bank 2 is selected. |
| | | | 1 | 1 | Bank 3 is selected. |

Always 0

**Remark** ×: Don't care

# 4. PERIPHERAL HARDWARE FUNCTIONS

## 4.1 DIGITAL I/O PORTS

The μPD75238 employs memory-mapped I/O, enabling all I/O ports to be mapped to data memory space.

**Fig. 4-1 Data Memory Address Assigned to Digital Port**

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FF0H | P03 | P02 | P01 | P00 | PORT 0 |
| FF1H | P13 | P12 | P11 | P10 | PORT 1 |
| FF2H | P23 | P22 | P21 | P20 | PORT 2 |
| FF3H | P33 | P32 | P31 | P30 | PORT 3 |
| FF4H | P43 | P42 | P41 | P40 | PORT 4 |
| FF5H | P53 | P52 | P51 | P50 | PORT 5 |
| FF6H | P63 | P62 | P61 | P60 | PORT 6 |
| FF7H | P73 | P72 | P71 | P70 | PORT 7 |
| FF8H | P83 | P82 | P81 | P80 | PORT 8 |
| FF9H | P93 | P92 | P91 | P90 | PORT 9 |
| FFAH | P103 | P102 | P101 | P100 | PORT 10 |
| FFBH | P113 | P112 | P111 | P110 | PORT 11 |
| FFCH | P123 | P122 | P121 | P120 | PORT 12 |
| FFDH | P133 | P132 | P131 | P130 | PORT 13 |
| FFEH | P143 | P142 | P141 | P140 | PORT 14 |
| FFFH | P153 | P152 | P151 | P150 | PORT 15 |

**(1) Configurations of digital I/O ports**

Fig. 4-2 to Fig. 4-11 show the configurations of the ports.

**(2) I/O mode setting**

The I/O mode of each I/O port is set by the port mode register as shown in Fig. 4-12.

Each port functions as an input port when the corresponding bit of the port mode register is set to 0, and functions as an output port when the same corresponding bit is set to 1.

An 8-bit memory manipulation instruction is used to set port mode register group A or B.

A $\overline{\text{RESET}}$ input clears all bits of each port mode register to 0. This means that the output buffers are set off, and all ports are placed in the input mode.

**(3) Operation of digital I/O ports**

When an instruction is executed, the operation of the port and pins depends on the I/O mode setting, as listed in Table 4-1.

**Table 4-1 I/O Port Operations by I/O Instructions**

| | Input mode (corresponding bit in the mode register is 0) [Output buffer is off] | Output mode (corresponding bit in the mode register is 1) [Output buffer is on] |
|---|---|---|
| When a 1-bit test instruction, 1-bit input instruction, or 4-/8-bit input instruction is executed | Receives data on certain pins. | Receives the contents of the output latch. |
| When a 4-/8-bit output instruction is executed | Transfers data in the accumulator to the output latch. | Outputs data in the accumulator to output pins. |
| When a 1-bit output instruction[Note] is executed | The contents of the output latch are undefined. | Changes the output pin state according to the instruction. |

**Note** Instructions such as SET1/CLR1/MOV1 PORTn.bit, CY

**Fig. 4-2 Configuration of Ports 0, 1, and 8**

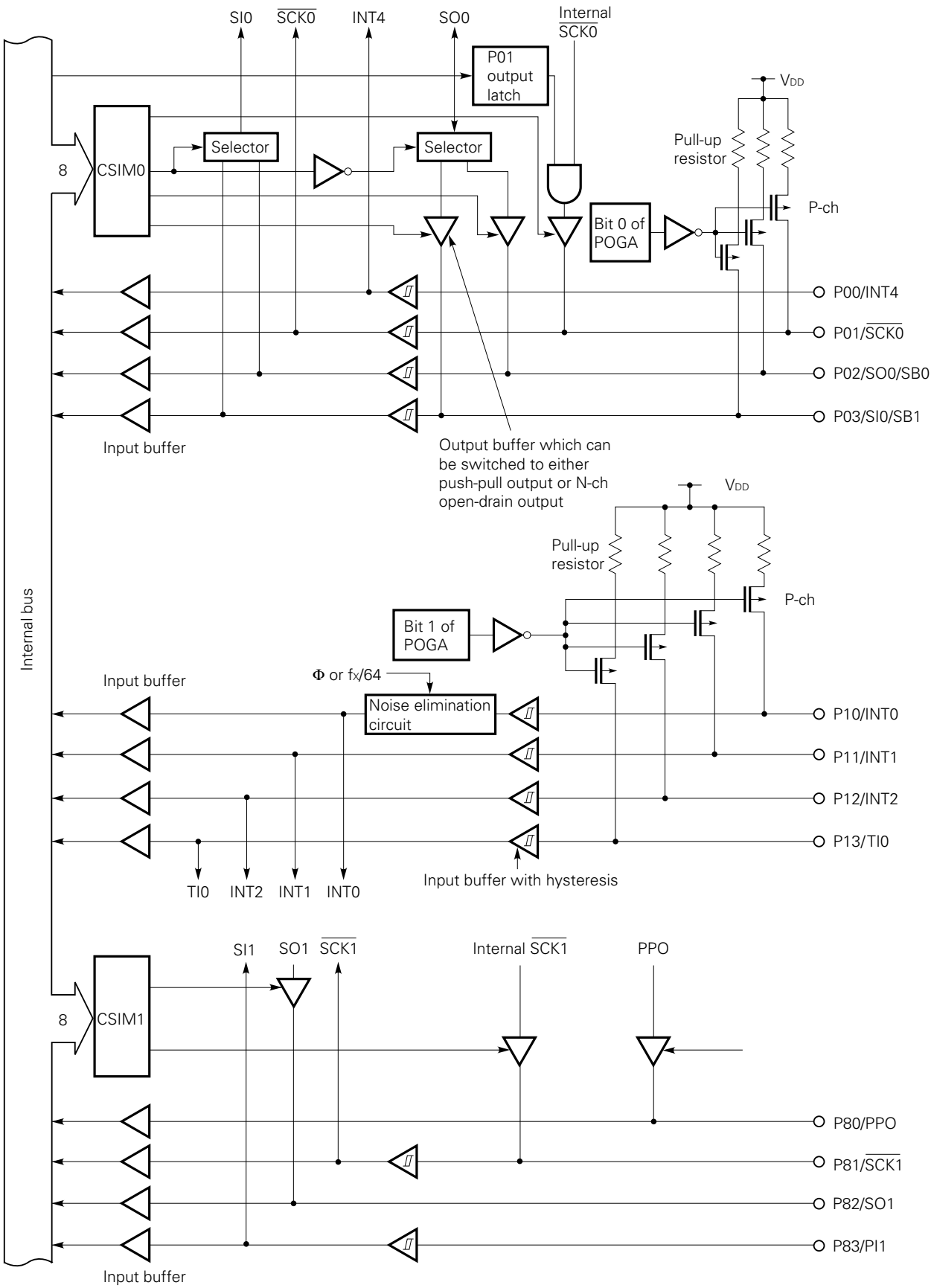


**Fig. 4-3 Configuration of Ports 3n and 6n (n = 0 to 3)**

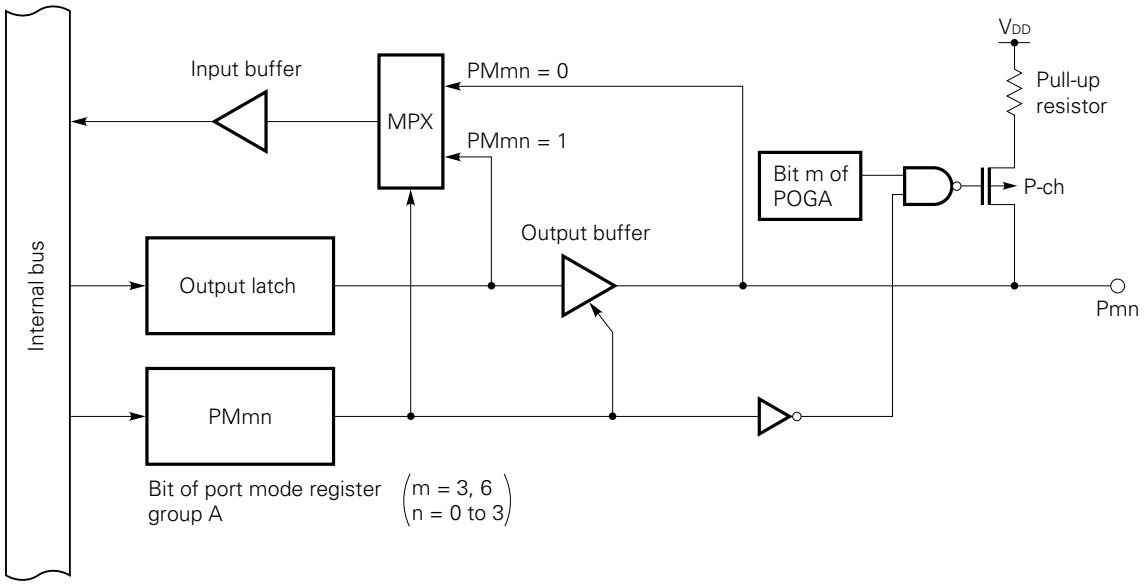



**Fig. 4-4 Configuration of Port 2**

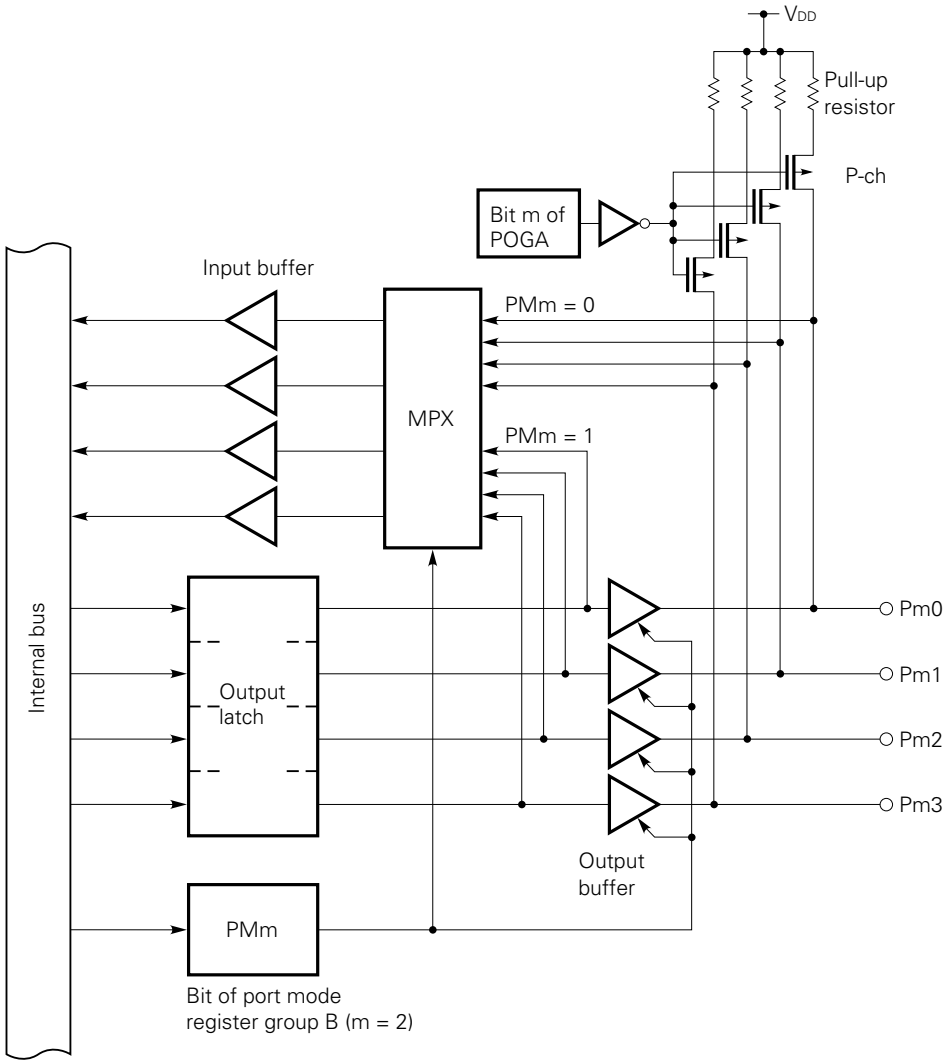


**Fig. 4-5 Configuration of Ports 4 and 5**

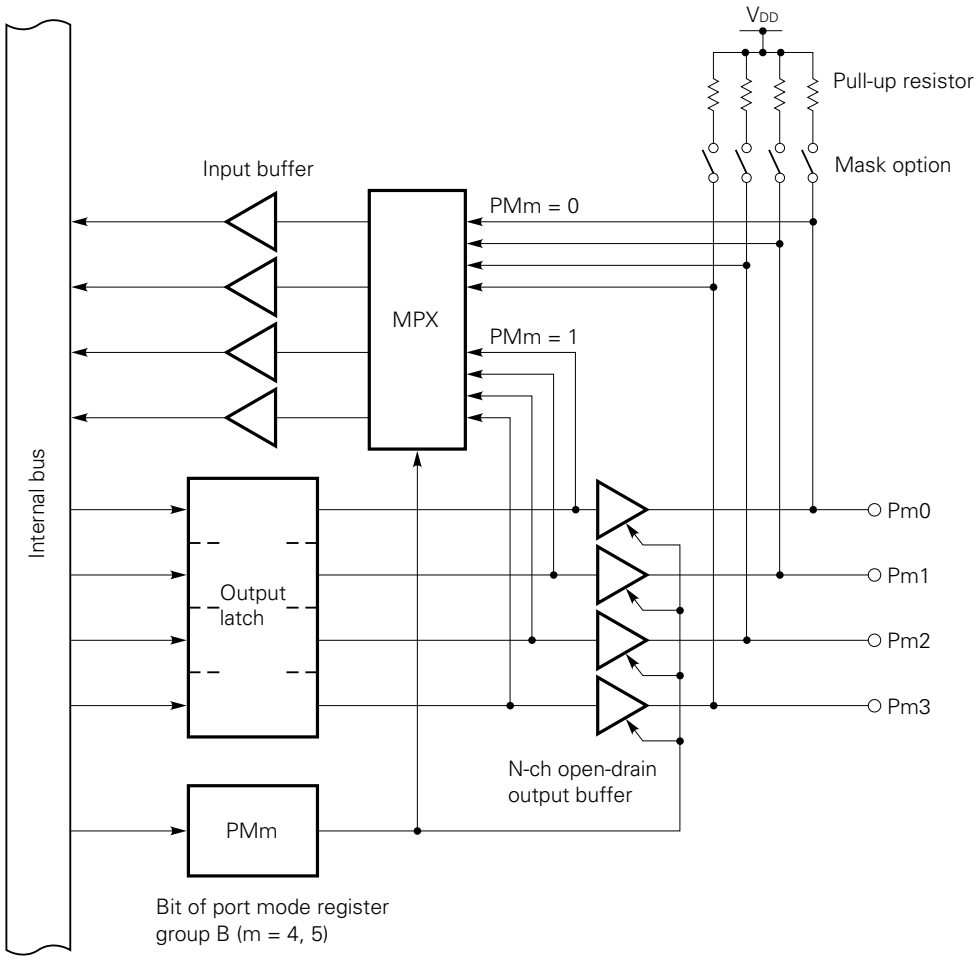


**Fig. 4-6 Configuration of Port 7**

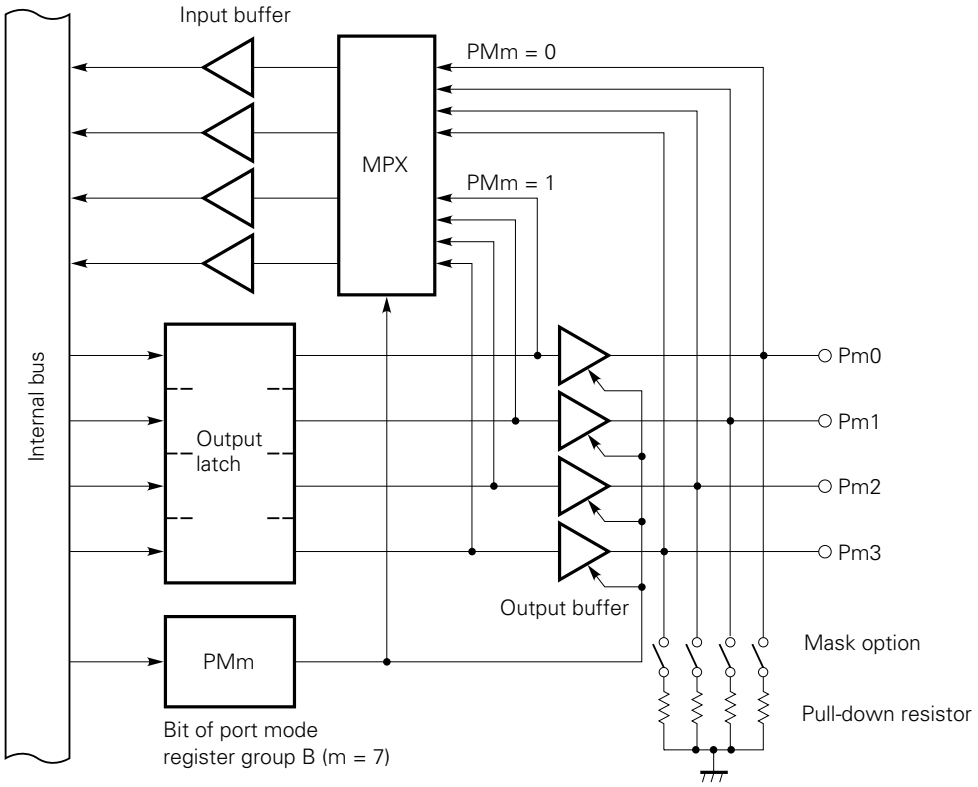



**Fig. 4-7 Configuration of Port 9**

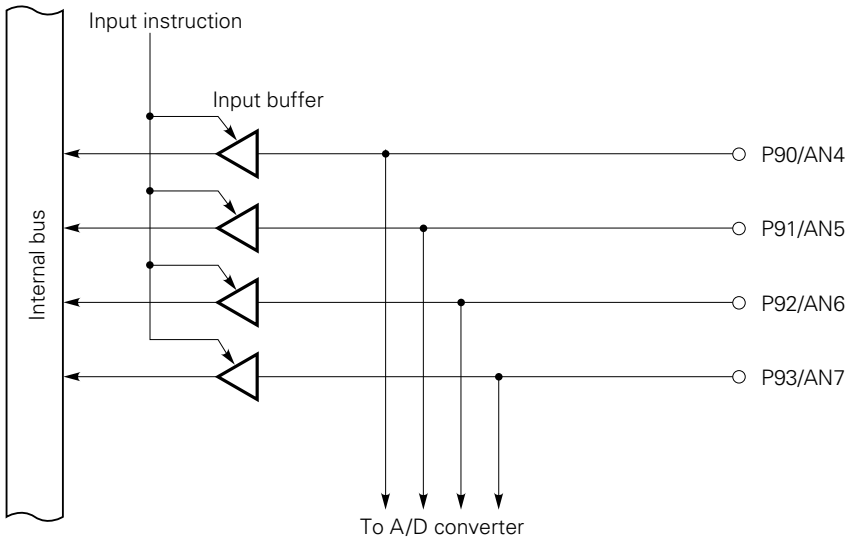
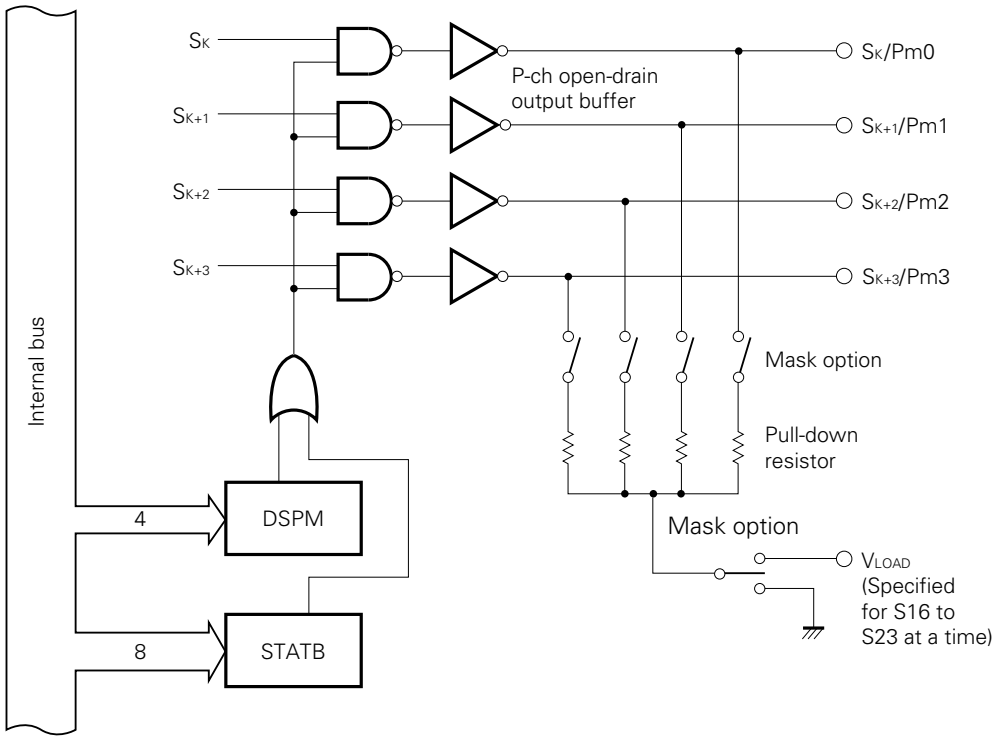


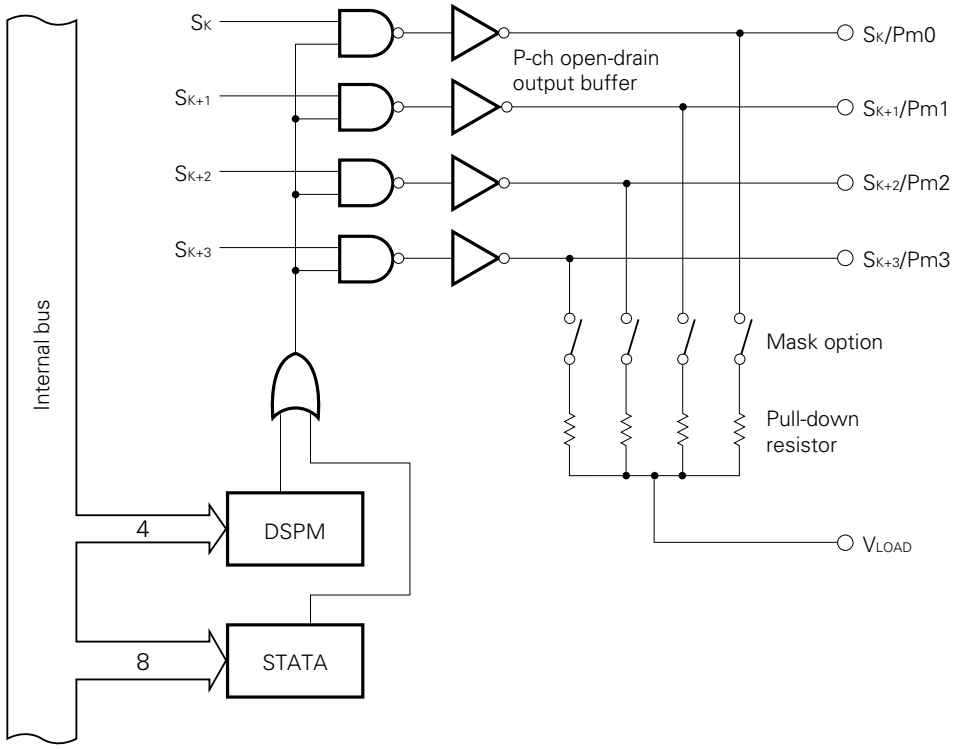
**Fig. 4-8 Configuration of Ports 10 and 11**




**Remarks 1.** Port 10: K = 16, m = 10

**2.** Port 11: K = 20, m = 11

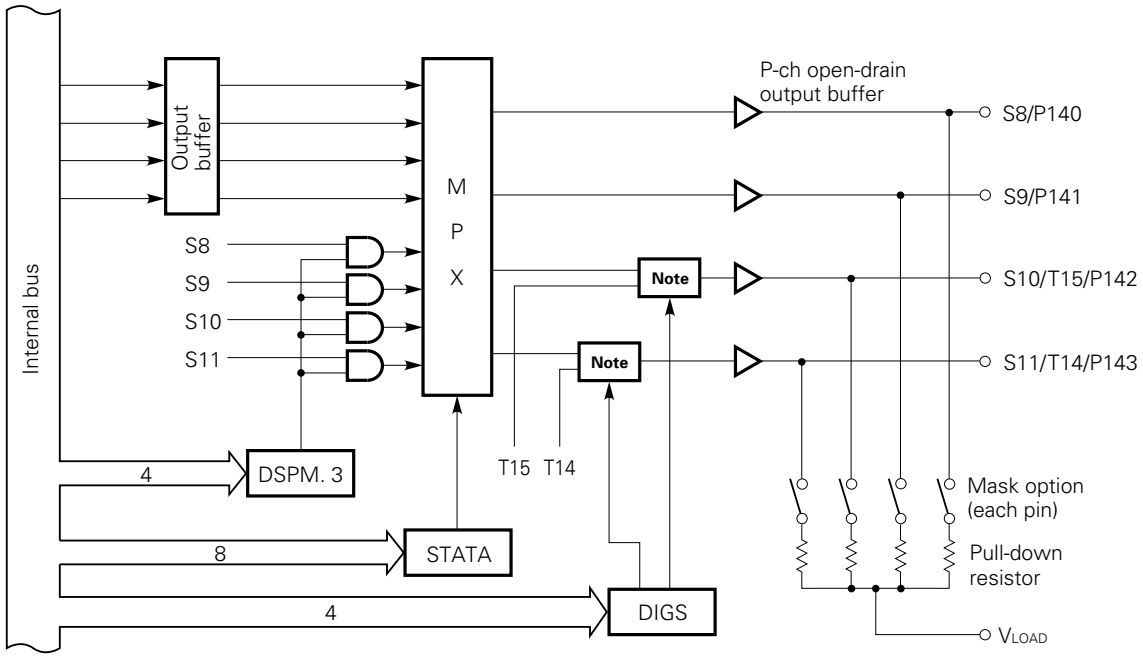
**Fig. 4-9 Configuration of Ports 12 and 13**




**Remarks 1.** Port 12: K = 0, m = 12

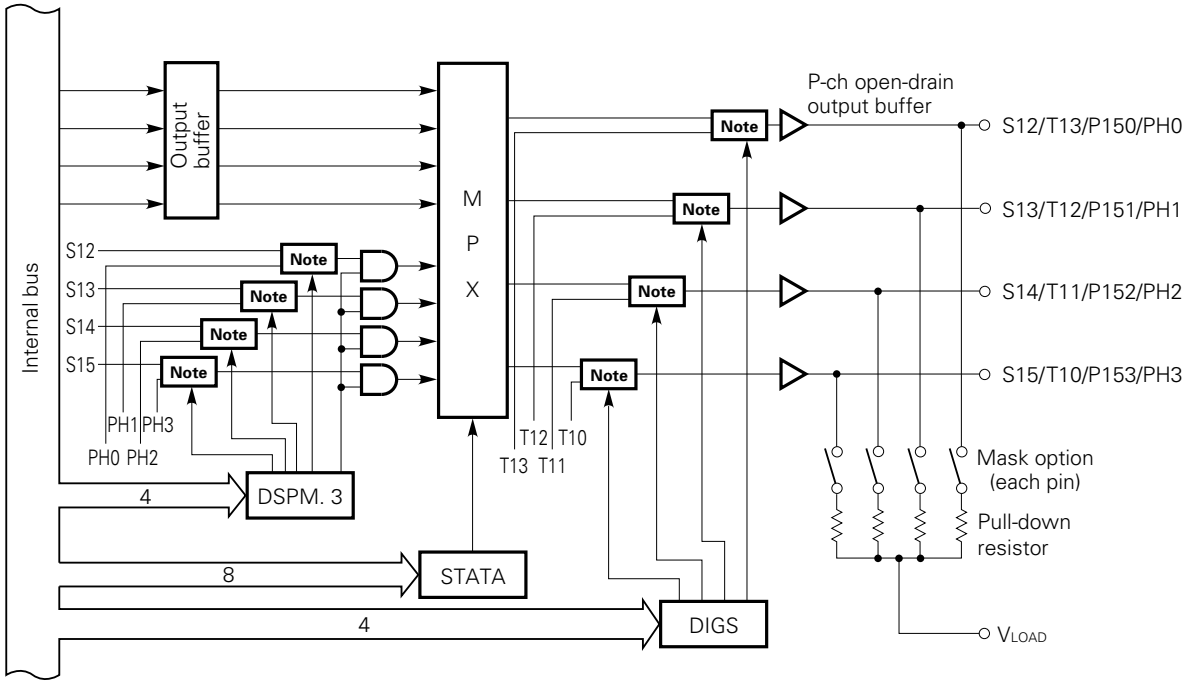
**2.** Port 13: K = 4, m = 13

**Fig. 4-10 Configuration of Port 14**




**Note** Selector

**Fig. 4-11 Configuration of Ports 15 and H**




**Note** Selector

**Fig. 4-12 Formats of the Port Mode Registers**

**Port mode register group A**

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FE8H | PM63 | PM62 | PM61 | PM60 | PM33 | PM32 | PM31 | PM30 | PMGA |

| Symbol | PM3n, PM6n | P3n and P6n pin I/O specification (n = 0-3) |
|---|---|---|
| PMGA | 0 | Input mode (output buffers set off) |
| | 1 | Output mode (output buffers set on) |

**Port mode register group B**

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FECH | PM7 | —— | PM5 | PM4 | —— | PM2 | —— | —— | PMGB |

| Symbol | PMn | Port n I/O specification (n = 2, 4, 5, and 7) |
|---|---|---|
| PMGB | 0 | Input mode (output buffers set off) |
| | 1 | Output mode (output buffers set on) |

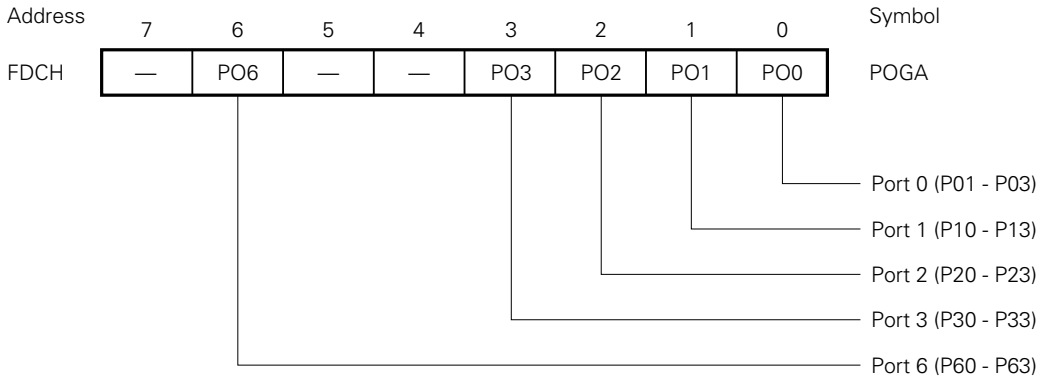
**Remark** ——: Don't care

**(4) Pull-up resistor register group A (POGA)**

Pull-up resistor register group A is a register to specify an internal pull-up register to each port pin of ports 0 to 3 and port 6 (excluding P00). Fig. 4-13 shows the format of this register.

When a pull-up resistor is to be contained, set 1 to an associated bit, and when a pull-up resistor is not to be contained, set 0.

**Fig. 4-13 Format of the Register Group A Specifying the Use of Pull-Up Resistors**




**Caution For mask option, ports 4 and 5 can contain pull-up resistors, and port 7 and ports 10 to 15 can contain pull-down resistors bit by bit.**

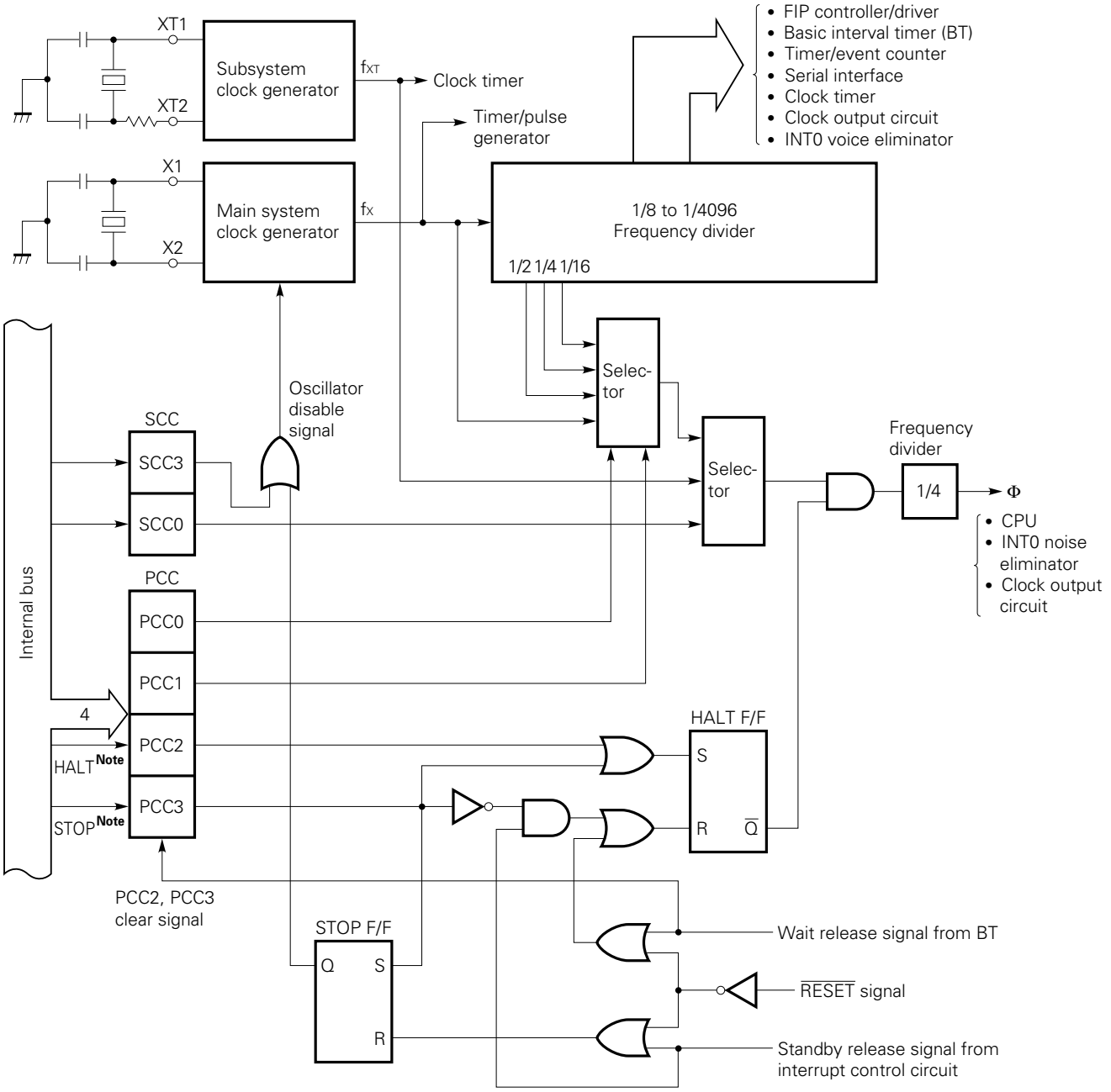
**Remark** ——: Don't care

## 4.2 CLOCK GENERATOR

### (1) Configuration of the clock generator

The clock generator supplies various clock signals to the CPU and peripheral hardware. Fig. 4-14 shows the configuration of the clock generator.

**Fig. 4-14 Block Diagram of the Clock Generator**




**Note** Instruction execution

**Remarks 1.** fX : Main system clock frequency

**2.** fXT: Subsystem clock frequency

**3.** Φ = CPU clock

**4.** PCC: Processor clock control register

**5.** SCC: System clock control register

**6.** One clock cycle (tCY) of the CPU clock (Φ) is equal to one machine cycle of an instruction. See **Chapter 11** for details of tCY. ★

**(2) Functions of the clock generator**

The clock generator generates the clock signals listed below, and controls the standby mode and other CPU operation modes.

- Main system clock: $f_X$
- Subsystem clock: $f_{XT}$
- CPU clock: Φ
- Clocks for peripheral hardware

The operation of the clock generator is determined by the processor clock control register (PCC) and system clock control register (SCC). The clock generator functions and operates as described below.

(a) A $\overline{\text{RESET}}$ input selects the lowest-speed mode (10.7 μs at 6.0 MHz)[Note 1] for the main system clock. (PCC = 0, SCC = 0)

(b) When the main system clock is selected, the PCC can be set to select one of four CPU clocks (0.67 μs, 1.33 μs, 2.67 μs, and 10.7 μs at 6.0 MHz)[Note 2].

(c) When the main system clock is selected, the two standby modes, STOP mode and HALT mode, are available.

(d) The SCC can be set to select the subsystem clock for very low-speed, low-current operation (122 μs at 32.768 kHz).

(e) When the subsystem clock is selected, main system clock generation can be stopped with the SCC. In addition, the HALT mode can be used, but the STOP mode cannot be used. (Subsystem clock generation cannot be stopped.)

(f) Clocks for peripheral hardware are produced by dividing the main system clock signal. Only to the watch timer, the subsystem clock can be directly supplied to continue the clock function.

(g) When the subsystem clock is selected, the watch timer can operate normally, but other hardware cannot be used because they operate with the main system clock.

**Notes 1.** 15.3 μs at 4.19 MHz

**2.** 0.95 μs, 1.91 μs, 3.82 μs, 15.3 μs at 4.19 MHz

**(3) Processor clock control register (PCC)**

The PCC is a 4-bit register for selecting CPU clock Φ with the low-order two bits and for selecting a CPU operation mode with the high-order two bits. (See **Fig. 4-15**.)

When bit 3 or bit 2 is set to 1, the standby mode is set. When this is released by the standby release signal, these bits are automatically cleared to return to the normal operation mode. (See **Chapter 6** for detailed information.)

A 4-bit memory manipulation instruction is used to set the low-order two bits of the PCC. (The high-order two bits are set to 0.)

Bit 3 and bit 2 are set to 1 using the STOP instruction and HALT instruction, respectively.

The STOP instruction and HALT instruction can be executed regardless of MBE setting.

A CPU clock can be selected only when the main system clock is used for operation. When the subsystem clock is selected for operation, the low-order two bits of the PCC are invalidated, and $f_{XT}/4$ is automatically set. The STOP instruction can be executed only when the main system clock is used for operation.

The generation of a $\overline{\text{RESET}}$ signal clears the PCC to 0.

**Fig. 4-15 Format of the Processor Clock Control Register**

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FB3H | PCC3 | PCC2 | PCC1 | PCC0 | PCC |

**CPU clock selection bit**

**(Operation with $f_X$ = 6.0 MHz)**

| | | SCC = 0 ( ) indicates $f_X$ = 6.0 MHz | | SCC = 1 ( ) indicates $f_{XT}$ = 32.768 kHz | |
|---|---|---|---|---|---|
| | | CPU clock frequency | 1 machine cycle | CPU clock frequency | 1 machine cycle |
| 0 | 0 | Φ = $f_X$/64 (93.7 kHz) | 10.7 μs | Φ = $f_{XT}$/4 (8.192 kHz) | 122 μs |
| 0 | 1 | Φ = $f_X$/16 (375 kHz) | 2.67 μs | Not to be set | |
| 1 | 0 | Φ = $f_X$/8 (750 kHz) | 1.33 μs | Φ = $f_{XT}$/4 (8.192 kHz) | 122 μs |
| 1 | 1 | Φ = $f_X$/4 (1.5 MHz) | 0.67 μs | | |

**(Operation with $f_X$ = 4.19 MHz)**

| | | SCC = 0 ( ) indicates $f_X$ = 4.19 MHz | | SCC = 1 ( ) indicates $f_{XT}$ = 32.768 kHz | |
|---|---|---|---|---|---|
| | | CPU clock frequency | 1 machine cycle | CPU clock frequency | 1 machine cycle |
| 0 | 0 | Φ = $f_X$/64 (65.5 kHz) | 15.3 μs | Φ = $f_{XT}$/4 (8.192 kHz) | 122 μs |
| 0 | 1 | Φ = $f_X$/16 (262 kHz) | 3.82 μs | Not to be set | |
| 1 | 0 | Φ = $f_X$/8 (524 kHz) | 1.91 μs | Φ = $f_{XT}$/4 (8.192 kHz) | 122 μs |
| 1 | 1 | Φ = $f_X$/4 (1.05 MHz) | 0.95 μs | | |

**Remarks 1.** $f_X$ : Output frequency from the main syem clock oscillator
**2.** $f_{XT}$: Output frequency from the subsyem clock oscillator

**CPU operation mode control bits**

| | | |
|---|---|---|
| 0 | 0 | Normal operation mode |
| 0 | 1 | HALT mode |
| 1 | 0 | STOP mode |
| 1 | 1 | Not to be set |

**(4) System clock control register (SCC)**

The SCC is a 4-bit register for selecting CPU clock Φ with the least significant bit and for controlling the termination of main system clock generation with the most significant bit. (See **Fig. 4-16**.)

SCC.0 and SCC.3 are located at the same data memory address, but both bits cannot be changed at the same time. Accordingly, SCC.0 and SCC.3 are set using bit manipulation instructions. SCC.0 and SCC.3 can be manipulated regardless of MBE setting.

Main system clock generation can be terminated by setting SCC.3 only when the subsystem clock is used for operation. The STOP instruction must be used for generation termination when the main system clock is used for operation.

A $\overline{\text{RESET}}$ input clears the SCC to 0.

**Fig. 4-16 Format of the System Clock Control Register**

Address: FB7H

Symbol: SCC

| SCC3 | — | — | SCC0 |
|---|---|---|---|

| SCC3 | SCC0 | System clock selection | Main system clock operation |
|---|---|---|---|
| 0 | 0 | Main system clock | Can oscillate |
| 0 | 1 | Subsystem clock | |
| 1 | 0 | Not to be set | |
| 1 | 1 | Subsystem clock | Oscillation stopped |

**Cautions 1. A time period of up to 1/$f_{XT}$ is needed to change the system clock. This means that to terminate main system clock generation, SCC.3 must be set when the machine cycles indicated in Table 4-2 or more have elapsed after the clock is switched from the main system clock to the subsystem clock.**

**2. When the main system clock is used for operation, setting SCC.3 to stop clock generation does not enter the normal STOP mode.**

**3. When SCC.3 is set to 1, the X1 input pin is connected to $V_{SS}$ (GND electric potential) to prevent leakage in the crystal oscillator. When an external clock is used as the main system clock, never set SCC.3 to 1.**

**4. When the four bits of PCC are set to 0001B (Φ = $f_X$/16), do not set SCC.0 to 1. Before switching the main system clock to the subsystem clock, be sure to manipulate the PCC bits so other than 0001B is set. When the system operates on the subsystem clock, the PCC bits must also be other than 0001B.**

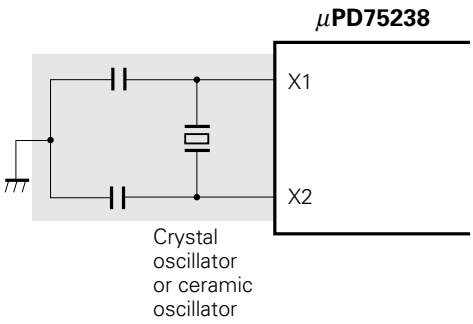
**(5) System clock oscillator**

The main system clock oscillator operates with a crystal (6.0 MHz standard) or ceramic resonator connected to the X1 and X2 pins.

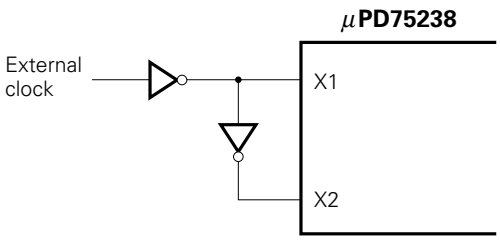
An external clock can also be input.

**Fig. 4-17 External Circuitry for the Main System Clock Oscillator**

**(a) Crystal/ceramic oscillation**




**(b) External clock**

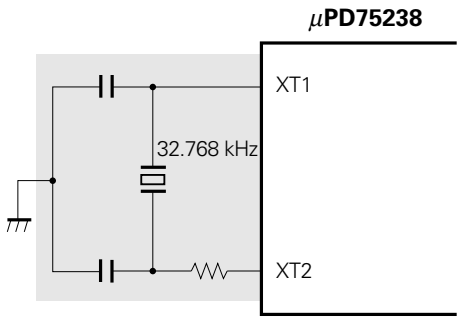



**Caution When an external clock is used, the STOP mode cannot be set. This is because the X1 pin is connected to $V_{SS}$ in the STOP mode.**

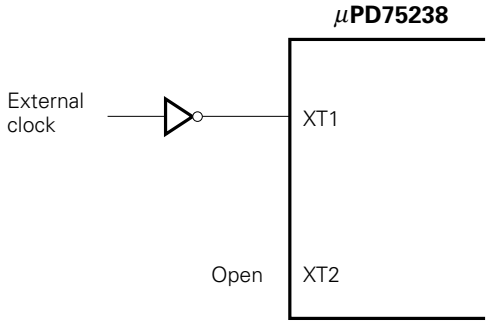
The subsystem clock oscillator operates with a crystal resonator (32.768 kHz standard) connected to the XT1 and XT2 pins.
An external clock can also be input.

**Fig. 4-18 External Circuitry for the Subsystem Clock Oscillator**

**(a) Crystal oscillation**




**(b) External clock**




**Caution When the main system clock or subsystem clock oscillator is used, conform to the following guidelines when wiring at the shaded portions of Fig. 4-17 and 4-18 to eliminate the influence of the wiring capacity.**

- **The wiring must be as short as possible.**
- **Other signal lines must not run in these areas. Any line carrying a high fluctuating current must be kept away as far as possible.**
- **The grounding point of the capacitor of the oscillator must have the same potential as that of $V_{SS}$. It must not be grounded to ground patterns carrying a large current.**
- **No signal must be taken from the oscillator.**

**When the subsystem clock is used, pay special attention to its wiring; the subsystem clock oscillator has low amplification to minimize current consumption and is more likely to malfunction due to noise than the main system clock oscillator.**

**(6) Time required to change the system clock and CPU clock**

The system clock and CPU clock can be changed by using the least significant bit of the SCC and the low-order two bits of the PCC. This switching is not performed immediately after the contents of the registers are rewritten, but the system operates with the previous clock for some machine cycles. Accordingly, after this time period, the STOP instruction must be executed or SCC.3 must be set to 1 to terminate main system clock generation.

**Table 4-2 Maximum Time Required to Change the System Clock and CPU Clock**

| Setting before switching | | | Setting after switching | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCC | PCC | PCC | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 | SCC0 | PCC1 | PCC0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | × | × |
| 0 | 0 | 0 | | | | 1 machine cycle | | | 1 machine cycle | | | 1 machine cycle | | | $f_X$ /64$f_{XT}$ machine cycles (3 machine cycles) | | |
| | 0 | 1 | 4 machine cycles | | | | | | 4 machine cycles | | | 4 machine cycles | | | Not to be set | | |
| | 1 | 0 | 8 machine cycles | | | 8 machine cycles | | | | | | 8 machine cycles | | | $f_X$/8$f_{XT}$ machine cycles (23 machine cycles) | | |
| | 1 | 1 | 16 machine cycles | | | 16 machine cycles | | | 16 machine cycles | | | | | | $f_X$/4$f_{XT}$ machine cycles (46 machine cycles) | | |
| 1 | × | × | 1 machine cycle | | | Not to be set | | | 1 machine cycle | | | 1 machine cycle | | | | | |

**Remarks 1.** CPU clock Φ is supplied to the CPU in the μPD75238. The reciprocal of this frequency is a minimum instruction time (defined as one machine cycle in this manual).

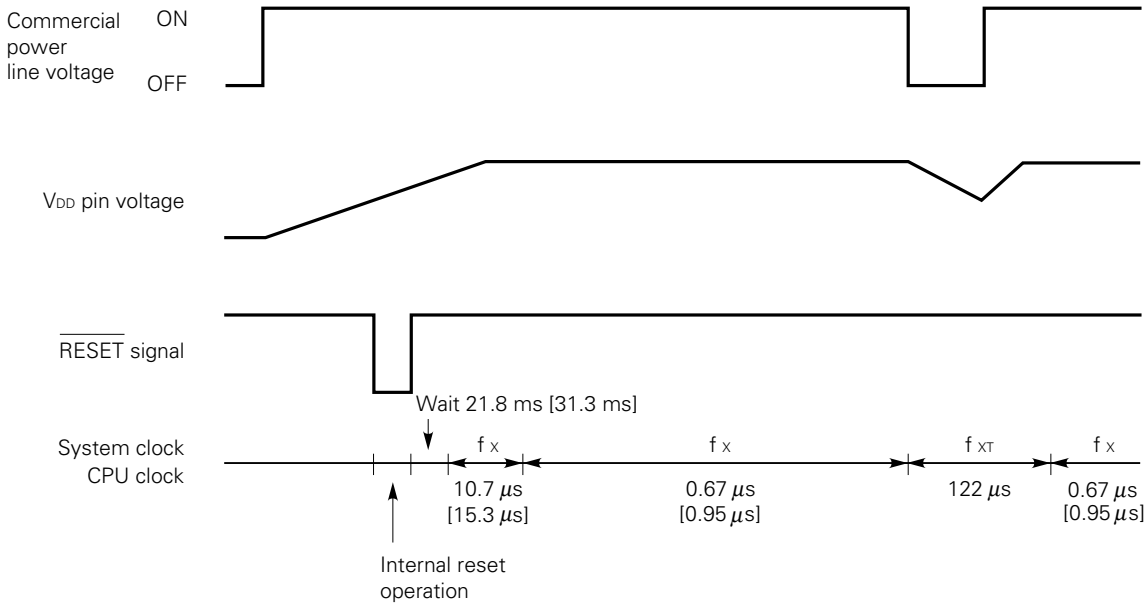
**2.** Time enclosed in parentheses is required when $f_X$ = 6.0 MHz and $f_{XT}$ = 32.768 kHz.

**Caution When the four bits of PCC are set to 0001B (Φ = $f_X$/16), do not set SCC.0 to 1. Before switching the main system clock to the subsystem clock, be sure to manipulate the PCC bits so other than 0001B is set. When the system operates on the subsystem clock, the PCC bits must also be other than 0001B.**

**(7) Procedure for changing the system clock and CPU clock**

The procedure for changing the system clock and CPU clock is explained using Fig. 4-19.

**Fig. 4-19 Changing the System Clock and CPU Clock**




**Remark** The values not enclosed in square brackets are for $f_X$ = 6.0 MHz and $f_{XT}$ = 32.768 kHz; the values enclosed in square brackets for $f_X$ = 4.19 MHz.

① A $\overline{\text{RESET}}$ input starts CPU operation at the lowest speed of the main system clock (10.7 μs at 6.0 MHz)**Note 1** after a wait time (21.8 ms at 6.0 MHz)**Note 2** for stable oscillation.

② The PCC is rewritten for highest-speed operation after a time elapse which is sufficient for the voltage on the $V_{DD}$ pin to be high enough for highest-speed operation.

③ The removal of commercial power is detected using, for example, an interrupt input (INT4 is useful), then SCC.0 is set to operate with the subsystem clock. (In this case, the start of subsystem clock generation must be confirmed beforehand.) After a time (32 machine cycles) required to switch to the subsystem clock elapses, SCC.3 is set to terminate main system clock generation.

④ After detecting the input of commercial power by using an interrupt, SCC.3 is cleared to start main system clock generation. After a time required for stable generation, SCC.0 is cleared to operate at highest speed.

**Notes 1.** 15.3 μs at 4.19 MHz

**2.** 31.3 ms at 4.19 MHz

### 4.3 CLOCK OUTPUT CIRCUIT

**(1) Configuration of the clock output circuit**

Fig. 4-20 shows the configuration of the clock output circuit.

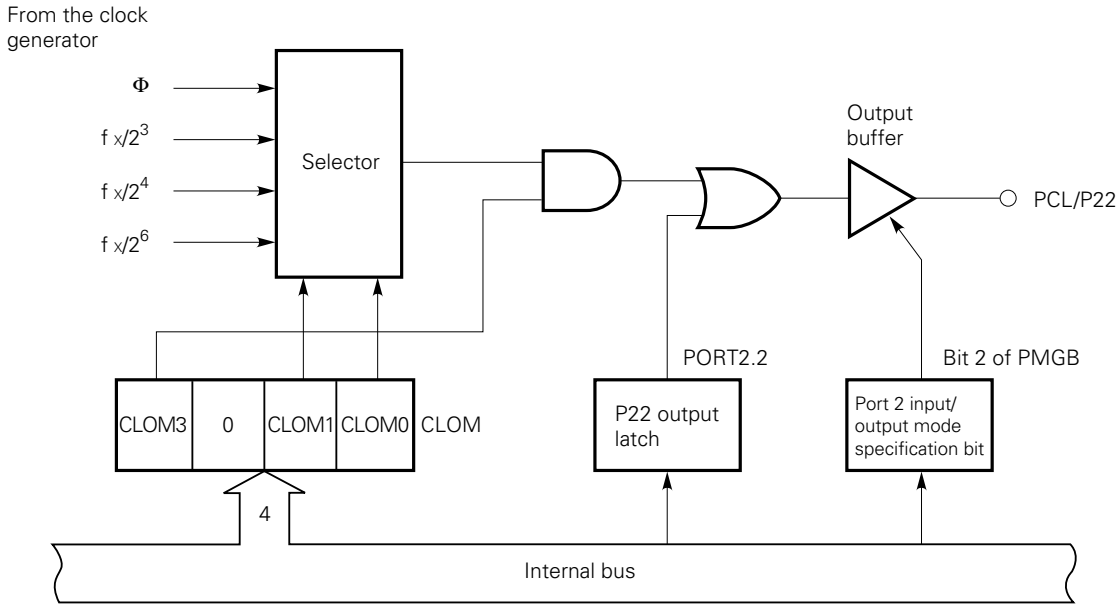
**(2) Functions of the clock output circuit**

The clock output circuit outputs a clock pulse signal on the P22/PCL pin for remote control or for supplying clock pulses to a peripheral LSI device.

The procedure for outputting a clock pulse signal is as follows:

(a) Select a clock output frequency, and disable clock output.
(b) Write a 0 in the P22 output latch.
(c) Set the output mode for port 2.
(d) Enable clock output.

**Fig. 4-20 Configuration of the Clock Output Circuit**




**Remark** The clock output circuit is designed so that pulses with short widths do not appear in enabling or disabling clock output.

**(3) Clock output mode register (CLOM)**

The CLOM is a 4-bit register to control clock output.

The CLOM is set with a 4-bit memory manipulation instruction. No read operation is allowed on this register.

**Example** CPU clock Φ is output on the PCL/P22 pin.

```
SEL   MB15        ; Or CLR1 MBE
MOV   A, #1000B
MOV   CLOM, A
```

A $\overline{\text{RESET}}$ input clears the CLOM to 0, disabling clock output.

**Fig. 4-21 Format of the Clock Output Mode Register**

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FD0H | CLOM3 | 0 | CLOM1 | CLOM0 | CLOM |

**Clock output frequency selection bit**

**(Frequency when $f_X$ = 6.0 MHz)**

| | | |
|---|---|---|
| 0 | 0 | Φ Output [Note] (1.50 MHz, 750 kHz, 375 kHz, 93.7 kHz) |
| 0 | 1 | Output $f_X/2^3$ (750 kHz) |
| 1 | 0 | Output $f_X/2^4$ (375 kHz) |
| 1 | 1 | Output $f_X/2^6$ (93.7 kHz) |

**(Frequency when $f_X$ = 4.19 MHz)**

| | | |
|---|---|---|
| 0 | 0 | Φ Output [Note] (1.05 MHz, 524 kHz, 262 kHz, 65.5 kHz) |
| 0 | 1 | Output $f_X/2^3$ (524 kHz) |
| 1 | 0 | Output $f_X/2^4$ (262 kHz) |
| 1 | 1 | Output $f_X/2^6$ (65.5 kHz) |

**Note** Φ is the CPU clock supply selected by PCC.

**Clock output enable/disable bit**

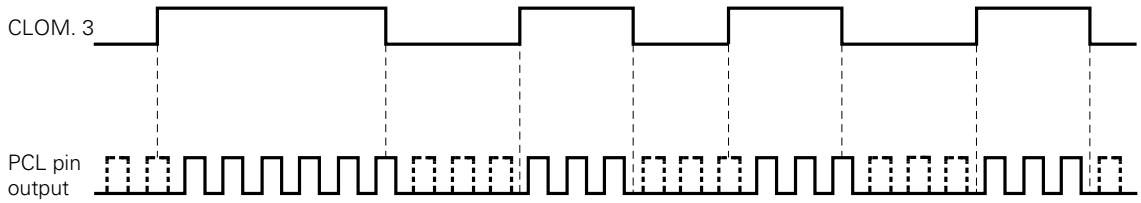
| | |
|---|---|
| 0 | Output disable |
| 1 | Output enable |

**Caution Be sure to write a 0 in bit 2 of the CLOM.**

**(4) Application to remote control output**

The clock output function of the μPD75238 is applicable to remote control output. The frequency of the carrier for remote control output is selected by the clock frequency select bit of the clock output mode register. Pulse output is enabled or disabled by controlling the clock output enable/disable bit by software. The clock output circuit is designed so that pulses with short widths do not appear in enabling or disabling clock output.

**Fig. 4-22 Application to Remote Control Output**



### 4.4 BASIC INTERVAL TIMER

**(1) Configuration of the basic interval timer**

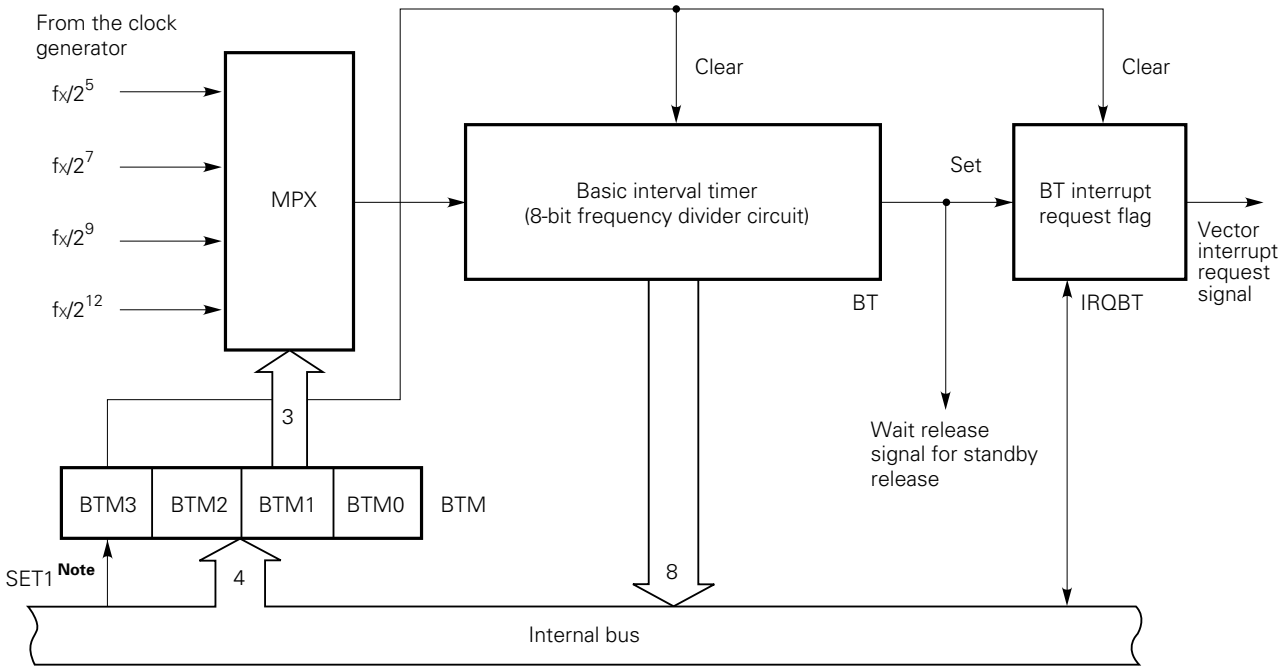
Fig. 4-23 shows the configuration of the basic interval timer.

**(2) Basic interval timer functions**

The basic interval timer provides the following four functions:

(a) Reference time generation (four time intervals)
(b) Application of watchdog timer for detecting program crashes
(c) Selection of a wait time for releasing the standby mode, and counting
(d) Reading the count value

**Fig. 4-23 Configuration of the Basic Interval Timer**




**Note** Instruction execution

### (3) Basic interval timer mode register (BTM)

BTM is a 4-bit register that controls operation of the basic interval timer.

The BTM contents are set by using a 4-bit memory manipulation instruction.

Bit 3 can be independently set using a bit manipulation instruction.

When bit 3 is set to 1, the contents of the basic interval timer are cleared, and the basic interval timer interrupt request flag (IRQBT) is also cleared (to start the basic interval timer).

A $\overline{\text{RESET}}$ input clears the contents to 0, and the longest interrupt request signal generation interval time is set.

**Fig. 4-24 Format of the Basic Interval Timer Mode Register**

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| F85H | BTM3 | BTM2 | BTM1 | BTM0 | BTM |

**(Frequency when $f_X$ = 6.0 MHz)**

| | | | Input clock specification | Interrupt interval time (wait time for releasing standby) |
|---|---|---|---|---|
| 0 | 0 | 0 | $f_X/2^{12}$ (1.46 kHz) | $2^{20}/f_X$ (175 ms) |
| 0 | 1 | 1 | $f_X/2^{9}$ (11.7 kHz) | $2^{17}/f_X$ (21.8 ms) |
| 1 | 0 | 1 | $f_X/2^{7}$ (46.9 kHz) | $2^{15}/f_X$ (5.46 ms) |
| 1 | 1 | 1 | $f_X/2^{5}$ (188 kHz) | $2^{13}/f_X$ (1.37 ms) |
| Other setting | | | Not to be set | – |

**(Frequency when $f_X$ = 4.19 MHz)**

| | | | Input clock specification | Interrupt interval time (wait time for releasing standby) |
|---|---|---|---|---|
| 0 | 0 | 0 | $f_X/2^{12}$ (1.02 kHz) | $2^{20}/f_X$ (250 ms) |
| 0 | 1 | 1 | $f_X/2^{9}$ (8.18 kHz) | $2^{17}/f_X$ (31.3 ms) |
| 1 | 0 | 1 | $f_X/2^{7}$ (32.768 kHz) | $2^{15}/f_X$ (7.82 ms) |
| 1 | 1 | 1 | $f_X/2^{5}$ (131 kHz) | $2^{13}/f_X$ (1.95 ms) |
| Other setting | | | Not to be set | – |

**Basic interval timer start control bit**

When "1" is written to this bit, the basic interval timer operation starts (the counter and the interrupt request flag are cleared).
When the operation starts, this bit is automatically reset to 0.

**(4) Operation of the basic interval timer**

The basic interval timer (BT) is always incremented by the clock supplied from the clock generator, and when it overflows, the interrupt request flag (IRQBT) is set. The count operation of BT cannot be stopped. One of four interrupt generation intervals can be selected by setting BTM. (See **Fig. 4-24**.)

The basic interval timer and the interrupt request flag can be cleared by setting bit 3 of BTM to 1 (instruction for starting as an interval timer).

The count status can be read by using an 8-bit manipulation instruction. No data can be loaded to the timer.

**Caution When reading the count value of the basic interval timer, execute a read instruction twice so that unstable data which has been counted will not be read. If the two read values are reasonable, use the second one as the result. If the two read values are far apart, retry from the beginning.**

To allow the system clock to stabilize after releasing the STOP mode, a wait function is available which stops the operation of the CPU until the basic interval timer overflows.

The wait time after a $\overline{\text{RESET}}$ input is fixed. On the other hand, a wait time can be selected by setting BTM when releasing the STOP mode with an interrupt occurrence. In this case, the wait times are the same as the interval times shown in Fig. 4-24.

BTM must be set before the STOP mode is set. (For details, see **Chapter 6**.)

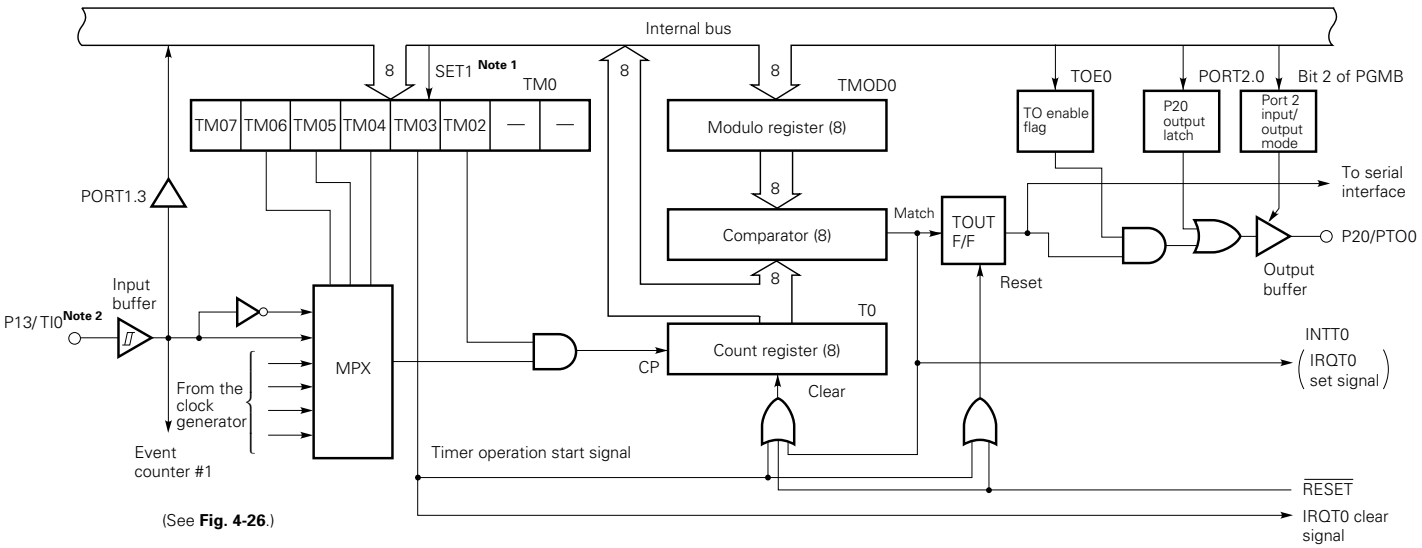
## 4.5 TIMER/EVENT COUNTER

**(1) Functions of the timer/event counter**

The timer/event counter has the following functions.

(a) Programmable interval timer operation
(b) Output of a square wave at a given frequency to the PTO0 pin
(c) Event counter operation
(d) Frequency divider operation that divides TI0 pin input by N and outputs the result to the PTO0 pin
(e) Supply of serial shift clock signal to a serial interface circuit
(f) Function of reading the state of counting

**Fig. 4-25 Block Diagram of the Timer/Event Counter**




(See **Fig. 4-26**.)

**Notes 1.** Instruction execution

**2.** The P13/TI0 pin is an external event pulse input pin shared between timer/event counter and event counter.

**(2) Timer/event counter mode register (TM0) and timer/event counter output enable flag (TOE0)**

The timer/event counter mode register (TM0) is an 8-bit register for controlling the timer/event counter. It is set by an 8-bit memory manipulation instruction.

Fig. 4-27 shows the format of the timer/event counter mode register.

Bit 3 is the timer start bit, and can be set independently of the other bits. Bit 3 is automatically reset to 0 when the timer starts operation.

A $\overline{\text{RESET}}$ input clears all the bits of the TM0 to 0.

The timer/event counter output enable flag (TOE0) enables or disables output of the timer out F/F (TOUT F/F) status to the PTO0 pin.

Fig. 4-26 shows the format of the timer/event counter output enable flag.

The timer out F/F (TOUT F/F) can be inverted by a match signal sent out from the comparator. The timer out F/F is reset when an instruction sets bit 3 of the TM0.

A $\overline{\text{RESET}}$ input clears the TOE0 and TOUT F/F to 0.

**Fig. 4-26 Format of the Timer/Event Counter Output Enable Flag**

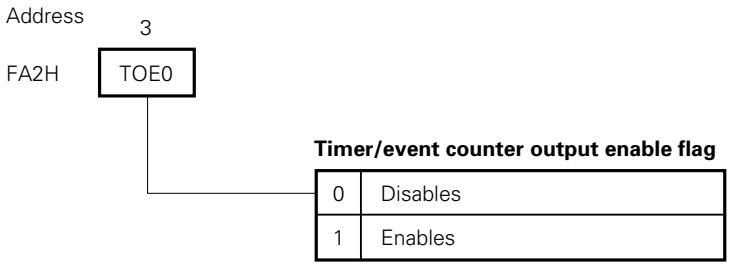


**Fig. 4-27 Format of the Timer/Event Counter Mode Register**

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FA0H | — | TM06 | TM05 | TM04 | TM03 | TM02 | — | — | TM0 |

**Operation mode**

| | Count operation |
|---|---|
| 0 | Halts (retains the contents of counting) |
| 1 | Count operation |

**Timer start specification bit**

When "1" is written to this bit, the counter and the IRQT0 flag are cleared. Count operation starts if bit 2 has been set to 1.

**Count pulse (CP) select bit**
**(Frequency when $f_X$ = 6.0 MHz)**

| TM06 | TM05 | TM04 | Count pulse (CP) |
|---|---|---|---|
| 0 | 0 | 0 | TI0 input rising edge |
| 0 | 0 | 1 | TI0 input falling edge |
| 1 | 0 | 0 | $f_X/2^{10}$ (5.86 kHz) |
| 1 | 0 | 1 | $f_X/2^8$ (23.4 kHz) |
| 1 | 1 | 0 | $f_X/2^6$ (93.8 kHz) |
| 1 | 1 | 1 | $f_X/2^4$ (375 kHz) |
| Other setting | | | Not to be set |

**(Frequency when $f_X$ = 4.19 MHz)**

| TM06 | TM05 | TM04 | Count pulse (CP) |
|---|---|---|---|
| 0 | 0 | 0 | TI0 input rising edge |
| 0 | 0 | 1 | TI0 input falling edge |
| 1 | 0 | 0 | $f_X/2^{10}$ (4.09 kHz) |
| 1 | 0 | 1 | $f_X/2^8$ (16.4 kHz) |
| 1 | 1 | 0 | $f_X/2^6$ (65.5 kHz) |
| 1 | 1 | 1 | $f_X/2^4$ (262 kHz) |
| Other setting | | | Not to be set |

### (3) Operation mode of the timer/event counter

The timer/event counter operates in the count operation disable mode or in the count operation mode, depending on the setting of the mode register.

The following operations are possible, regardless of the setting of the mode register:

(i) P13/TI0 pin signal input and test
(ii) Output of the timer out F/F status to the PTO0
(iii) Setting of the modulo register (TMOD0)
(iv) Reading from the count register (T0)
(v) Setting, clearing, and testing of the interrupt request flag (IRQT0)

#### (a) Count operation disable mode

This mode is set when bit 2 of TM0 is set to 0. In this mode, count operation is not performed because count pulse (CP) supply to the count register is stopped.

#### (b) Count operation mode

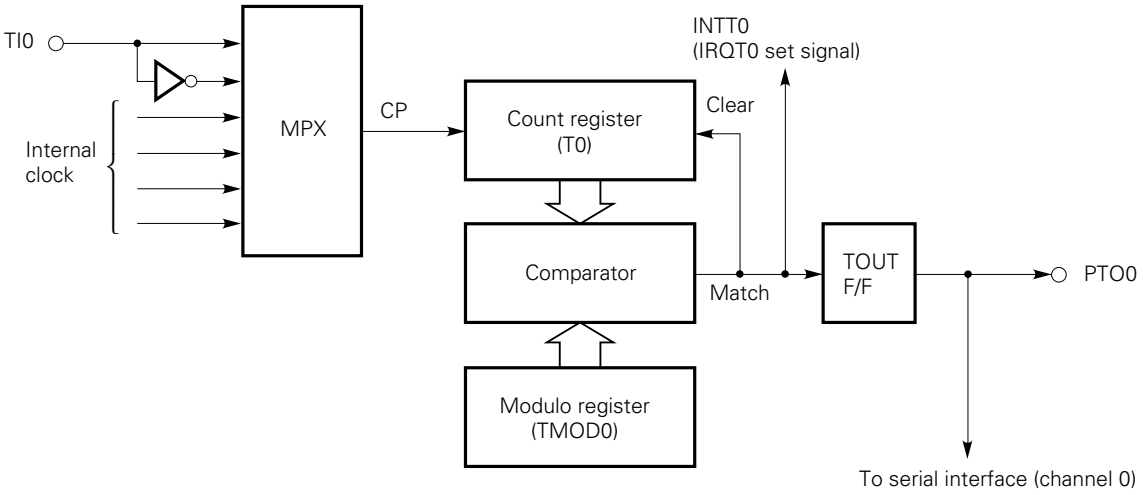
This mode is set when bit 2 of TM0 is set to 1. In this mode, a count pulse signal selected with bits 4 to 6 is supplied to the count register for count operation as shown in Fig. 4-28.

Timer operation is usually started in the following steps:

① A count value is set in the modulo register (TMOD0).
② An operation mode, count clock, and start instruction are set in the mode register (TM0).

An 8-bit data transfer instruction is used to set the modulo register.

**Fig. 4-28 Operation in the Count Operation Mode**



**(4) Time setting in the timer/event counter**

[Timer set time] (period) is [value of the modulo register + 1] divided by [count pulse frequency] selected by the timer mode register.

$$T(sec) = \frac{(n + 1)}{f_{CP}} = (n + 1) \cdot (resolution)$$

T(sec) : Timer set time (seconds)
$f_{CP}$ (Hz) : Count pulse frequency (Hz)
n : Value in the modulo register ($n \neq 0$)

Once the timer is set, the interrupt request signal (IRQT0) is generated at set time intervals. Table 4-3 indicates the resolution and maximum set time (set when FFH is set in the modulo register) of the timer/event counter for each count pulse signal.

**Table 4-3 Resolution and Maximum Set Time**

**(When fx = 6.0 MHz)**

| Mode register | | | Timer channel 0 | |
|---|---|---|---|---|
| TM06 | TM05 | TM04 | Resolution | Maximum set time |
| 1 | 0 | 0 | 171 μs | 43.7 ms |
| 1 | 0 | 1 | 42.7 μs | 10.9 ms |
| 1 | 1 | 0 | 10.7 μs | 2.73 ms |
| 1 | 1 | 1 | 2.67 μs | 683 μs |

**(When fx = 4.19 MHz)**

| Mode register | | | Timer channel 0 | |
|---|---|---|---|---|
| TM06 | TM05 | TM04 | Resolution | Maximum set time |
| 1 | 0 | 0 | 244 μs | 62.5 ms |
| 1 | 0 | 1 | 61.1 μs | 15.6 ms |
| 1 | 1 | 0 | 15.3 μs | 3.91 ms |
| 1 | 1 | 1 | 3.81 μs | 977 μs |

## 4.6 CLOCK TIMER

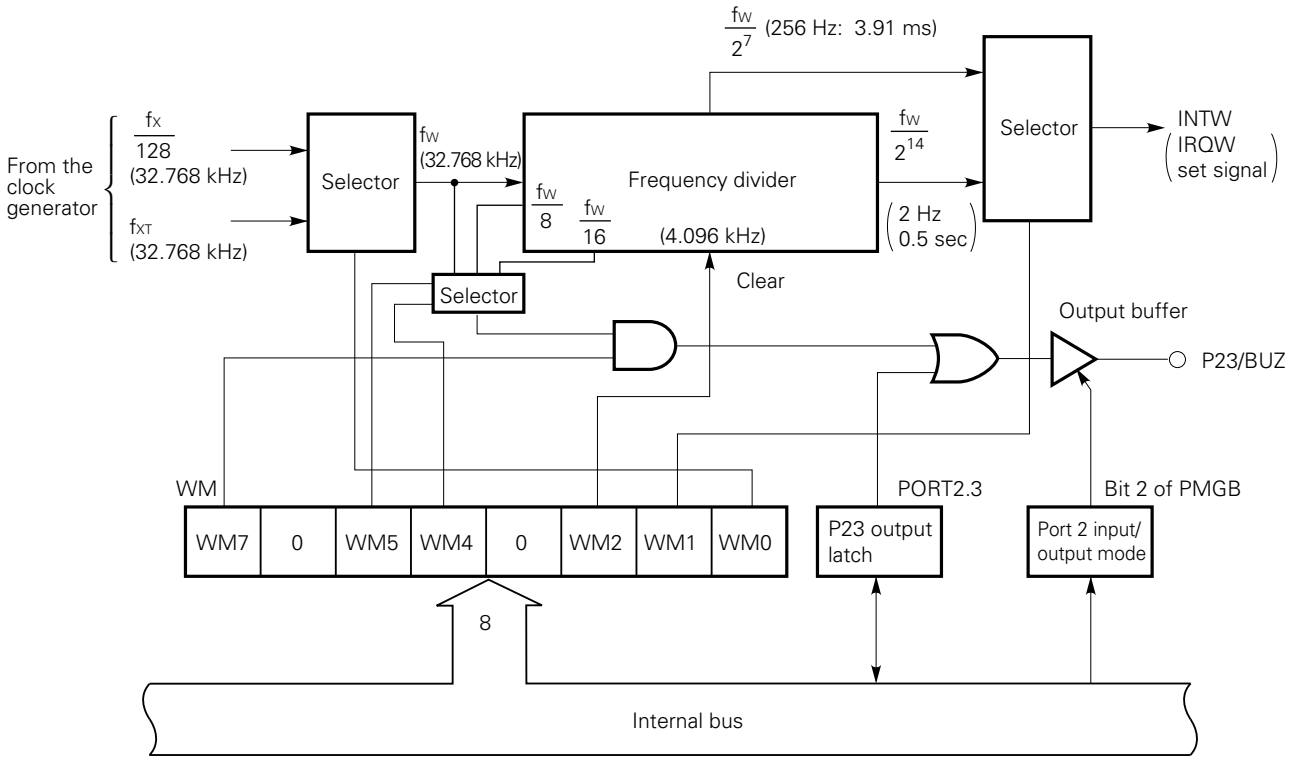
### (1) Clock timer

The μPD75238 contains one channel for a clock timer. Fig. 4-29 shows the configuration of the timer.

### (2) Clock timer functions

(a) The clock timer sets the test flag (IRQW) every 0.5 seconds.
The standby mode can be released with IRQW.
(b) Either the main system clock (4.19 MHz) or subsystem clock (32.768 kHz) can produce 0.5-second intervals.
(c) The fast-forward mode produces an interval 128 times faster (3.91 ms), which is useful for program debugging and testing.
(d) A fixed frequency (2.048, 4.096, or 32.768 kHz) can be output to the P23/BUZ pin, so that it can be used for sounding the buzzer and system clock frequency trimming.
(e) The frequency divider can be cleared, so the clock can start from zero seconds.

**Fig. 4-29 Block Diagram of the Clock Timer**




**Remark** The values in parentheses are for fX = 4.194304 MHz and fXT = 32.768 kHz

**Caution When the main system clock operates at 6.0 MHz, a time interval of 0.5 s cannot be produced. Before producing this time interval, the main system clock must be changed to the subsystem clock.**

### (3) Watch mode register (WM)

The watch mode register (WM) is an 8-bit register that controls the clock timer, and that is set with an 8-bit memory manipulation instruction. Fig. 4-30 shows the format.

An 8-bit memory manipulation instruction is used to set the watch mode register. A $\overline{\text{RESET}}$ input clears all bits to 0.

**Fig. 4-30 Format of the Clock Mode Register**

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| F98H | WM7 | 0 | WM5 | WM4 | 0 | WM2 | WM1 | WM0 | WM |

**Count clock ($f_W$) selection bit**

| | | |
|---|---|---|
| WM0 | 0 | Selects divided system clock output: $\frac{f_X}{128}$ |
| | 1 | Selects subsystem clock: $f_{XT}$ |

**Operation mode selection bit**

| | | |
|---|---|---|
| WM1 | 0 | Normal clock mode ( $\frac{f_W}{2^{14}}$ : sets IRQW at 0.5 s) |
| | 1 | Advanced clock mode ( $\frac{f_W}{2^7}$ : sets IRQW at 3.91 ms) |

**Clock operation enable/disable bit**

| | | |
|---|---|---|
| WM2 | 0 | Disables clock operation (clears the frequency dividing circuit) |
| | 1 | Enables clock operation |

**BUZ output frequency selection bit**

| WM5 | WM4 | BUZ output frequency |
|---|---|---|
| 0 | 0 | $f_W/2^4$ (2.048 kHz) |
| 0 | 1 | $f_W/2^3$ (4.096 kHz)**Note** |
| 1 | 0 | Not to be set |
| 1 | 1 | $f_W$ (32.768 kHz)**Note** |

**Note** Not supported by the IE-75000-R.

**BUZ output enable/disable bit**

| | | |
|---|---|---|
| WM7 | 0 | Disables BUZ output |
| | 1 | Enables BUZ output |

### 4.7 TIMER/PULSE GENERATOR

**(1) Timer/pulse generator functions**

The μPD75238 contains one channel for a timer/pulse generator that can be used as a timer or a pulse generator. It has the following functions:

**(a) Functions available when the timer/pulse generator is used in the timer mode**

- 8-bit interval timer operation using one of five clock sources (occurrence of IRQTPG)
- Square wave output to the PPO pin

**(b) Functions available when the timer/pulse generator is used in the PWM pulse generation mode**

- PWM pulse output to the PPO pin with an accuracy of 14 bits (applicable for electronic tuning when used as an D/A converter)
- Generation of interrupts at regular intervals ($2^{15}$/fx = 5.46 ms at 6.0 MHz)[Note]

**Note** 7.81 ms at 4.19 MHz

If pulse output is unnecessary, the PPO pin can be used as a 1-bit output port.

**Caution If the timer/pulse generator is operating when the STOP mode is set, it may malfunction. So the timer/pulse generator must be disabled with the mode register in advance.**

**(2) Timer/pulse generator mode register (TPGM)**

The timer/pulse generator mode register (TPGM) is an 8-bit register that controls operation of the timer/pulse generator. Fig. 4-31 shows the format of the register.

TPGM is set with an 8-bit memory manipulation instruction.

Bit 3 enables or disables the transfer (reloading) of the timer/pulse generator modulo register (MODH and MODL) contents to the modulo latch. Bit 3 can be manipulated independently of the other bits.

By setting TPGM1 to 0, timer/pulse generator operation can be stopped to decrease current consumption.

A $\overline{\text{RESET}}$ input clears all bits to 0.

**Fig. 4-31 Format of Timer/Pulse Generator Mode Register**

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| F90H | TPGM7 | — | TPGM5 | TPGM4 | TPGM3 | 0 | TPGM1 | TPGM0 | TPGM |

**Timer/pulse generator operation mode selection bit**

| | | |
|---|---|---|
| TPGM0 | 0 | Select PWM pulse generation mode |
| | 1 | Select timer mode |

**Timer/pulse generator operation enable/disable bit**

| | | |
|---|---|---|
| TPGM1 | 0 | Disable timer/pulse generator operation |
| | 1 | Enable timer/pulse generator operation |

**Modulo register reload enable/disable bit**

| | | |
|---|---|---|
| TPGM3 | 0 | Disable reloading of modulo register |
| | 1 | Enable reloading of modulo register |

**PPO output latch data**

| | | |
|---|---|---|
| TPGM4 | 0 | Output 0 to PPO output latch |
| | 1 | Output 1 to PPO output latch |

**PPO pin output selection bit static/pulse**

| | | |
|---|---|---|
| TPGM5 | 0 | Static output on PPO pin |
| | 1 | Pulse output (square wave/PWM) on PPO pin |

**PPO pin output enable/disable bit**

| | | |
|---|---|---|
| TPGM7 | 0 | Disable output on PPO pin (high-impedance) |
| | 1 | Enable output on PPO pin |

**(3) Configuration and operation when the timer/pulse generator is used in the timer mode**

Fig. 4-32 shows the configuration when the timer/pulse generator is used in the timer mode.

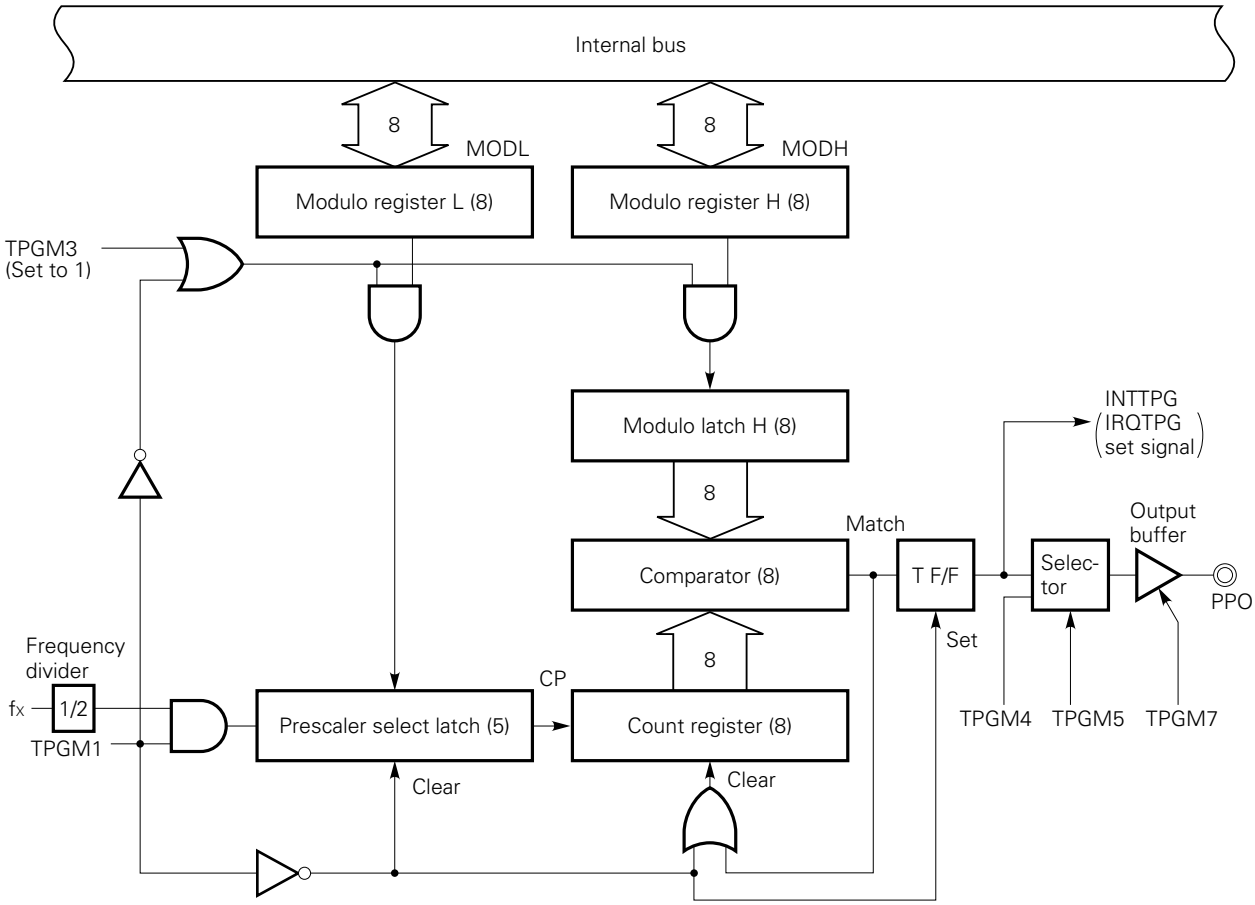
The timer mode is selected by setting bit 0 of TPGM to 1. In the timer mode, TPGM3 must be set to 1, allowing a modulo register to be reloaded at any time.

In the timer mode, a prescaler is selected with the modulo register L (MODL), and a frequency or interrupt interval value is set in the modulo register H (MODH). The timer starts when the TPGM1 is set to 1.

Fig. 4-33 shows the operation timing for the MODH setting, and Table 4-4 shows the setting of a frequency or interrupt interval.

The output to the PPO pin can be switched between the square wave output and static output. To output a square wave, set TPGM5 and TPGM7 to 1.

**Fig. 4-32 Block Diagram of the Timer/Pulse Generator (Timer Mode)**




**Caution When the timer operating in the timer operation mode is stopped, IRQTPG may be set because T F/F is set. So, the timer must be stopped with an interrupt being disabled, then IRQTPG must be cleared.**

**Fig. 4-33 Timer Mode Operation Timing**

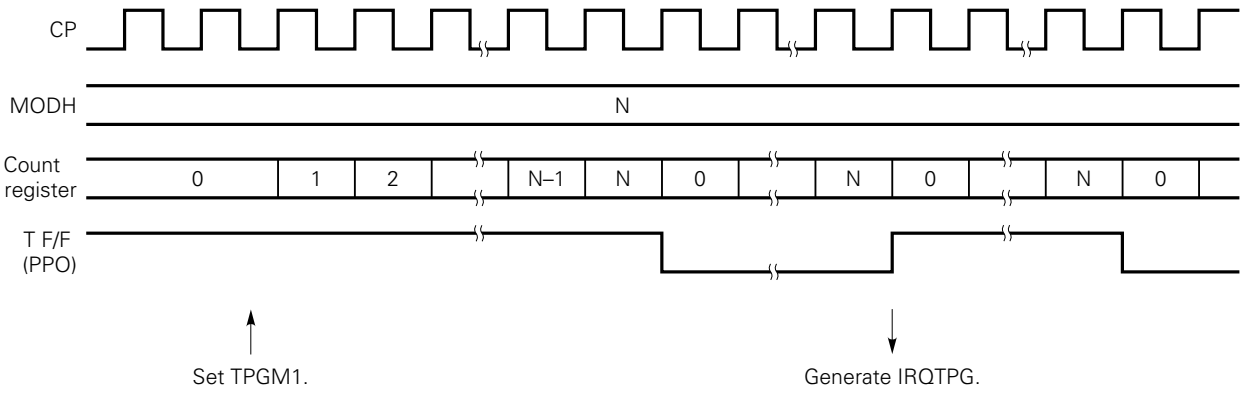



**Table 4-4 Modulo Register Settings**

**(When fx = 6.0 MHz)**

| MODL bits 2-6 | | | | | Interrupt generation interval (fx = 6.0 MHz) | Square wave output frequency (fx = 6.0 MHz) |
|---|---|---|---|---|---|---|
| 6 | 5 | 4 | 3 | 2 | | |
| 0 | 0 | 0 | 0 | 1 | 256 (N+1)/fx = 85.3 μs - 10.9 ms | fx/256 (N+1) = 91.6 Hz - 11.7 kHz |
| 0 | 0 | 0 | 1 | 0 | 128 (N+1)/fx = 42.7 μs - 5.45 ms | fx/128 (N+1) = 183 Hz - 23.4 kHz |
| 0 | 0 | 1 | 0 | 0 | 64 (N+1)/fx = 21.3 μs - 2.73 ms | fx/64 (N+1) = 366 Hz - 46.9 kHz |
| 0 | 1 | 0 | 0 | 0 | 32 (N+1)/fx = 10.7 μs - 1.37 ms | fx/32 (N+1) = 732 Hz - 93.8 kHz |
| 1 | 0 | 0 | 0 | 0 | 16 (N+1)/fx = 5.33 μs - 683 μs | fx/16 (N+1) = 1465 Hz - 188 kHz |

**(When fx = 4.19 MHz)**

| MODL bits 2-6 | | | | | Interrupt generation interval (fx = 4.19 MHz) | Square wave output frequency (fx = 4.19 MHz) |
|---|---|---|---|---|---|---|
| 6 | 5 | 4 | 3 | 2 | | |
| 0 | 0 | 0 | 0 | 1 | 256 (N+1)/fx = 122 μs - 15.6 ms | fx/256 (N+1) =64 Hz - 8 kHz |
| 0 | 0 | 0 | 1 | 0 | 128 (N+1)/fx = 61.0 μs - 7.81 ms | fx/128 (N+1) = 128 Hz - 16 kHz |
| 0 | 0 | 1 | 0 | 0 | 64 (N+1)/fx = 30.5 μs - 3.91 ms | fx/64 (N+1) = 256 Hz - 32 kHz |
| 0 | 1 | 0 | 0 | 0 | 32 (N+1)/fx = 15.3 μs - 1.95 ms | fx/32 (N+1) = 512 Hz - 65 kHz |
| 1 | 0 | 0 | 0 | 0 | 16 (N+1)/fx = 7.63 μs - 977 μs | fx/16 (N+1) = 1024 Hz - 131 kHz |

**Cautions 1. A value other than the above cannot be set in MODL. Bits 0, 1, and 7 must be set to 0.**

**2. N is the set value of MODH. 0 must not be set for N.**
**Be sure to set a value from 1 to 255 for N.**

**(4) Configuration and operation when the timer/pulse generator is used in the PWM pulse generation mode**

Fig. 4-34 shows the configuration when the timer/pulse generator is used in the PWM pulse generation mode.

The PWM pulse generation mode is selected by setting TPGM0 to 0. TPGM5 and TPGM7 are set to 1 to enable pulse output. In the PWM mode, the PWM pulse signal can be output on the PPO pin, and IRQTPG can be set at intervals of a fixed time period ($2^{15}$/fx = 5.46 ms at 6.0 MHz)**Note 1**.

PWM pulses output by the μPD75238 are active-low and have an accuracy of 14 bits. This pulse signal is applicable for electronic tuning and control of a DC motor when it is integrated by an external low-pass filter and is converted to analog voltage. (See **Fig. 4-35**.)

The PWM pulse signal is generated by combining the basic period determined by $2^{10}$/fx (171 μs at 6.0 MHz)**Note 2** and the secondary period by $2^{15}$/fx (5.46 ms at 6.0 MHz)**Note 1** so that the time constant of the external low-pass filter can be decreased.

The low-level width of a PWM pulse depends on the 14-bit modulo latch value. The upper 8 bits of the modulo latch are sent from the 8 bits of MODH, and the lower 6 bits of the latch are sent from the upper 6 bits of MODL.

When the PWM pulse signal is converted to analog form, the voltage level of the analog output is obtained as follows:

$$V_{AN} = V_{ref} \times \frac{\text{Value of modulo latch}}{2^{14}}$$

$V_{ref}$: Reference voltage of external switching circuitry

To prevent an incorrect PWM pulse from being output by unstable modulo latch data being rewritten, the μPD75238 allows correct data to be written in MODH and MODL beforehand with 8-bit manipulation instructions, then in the 14-bit data which is to be transferred to the modulo latch at one time. This transfer operation is referred to as reloading, and it is controlled by TPGM3.

**Cautions 1. If the modulo register H (MODH) is set to 0, the PWM pulse generator cannot function normally. So be sure to set MODH to a value from 1 to 255.**

**2. If the lower 2 bits of the modulo register L (MODL) are read, the read result is unpredictable.**

**3. If the modulo latch is changed in a shorter period than the PWM pulse basic period $2^{10}$/fx (171 μs at 6.0 MHz)Note 2, PWM pulses do not change.**

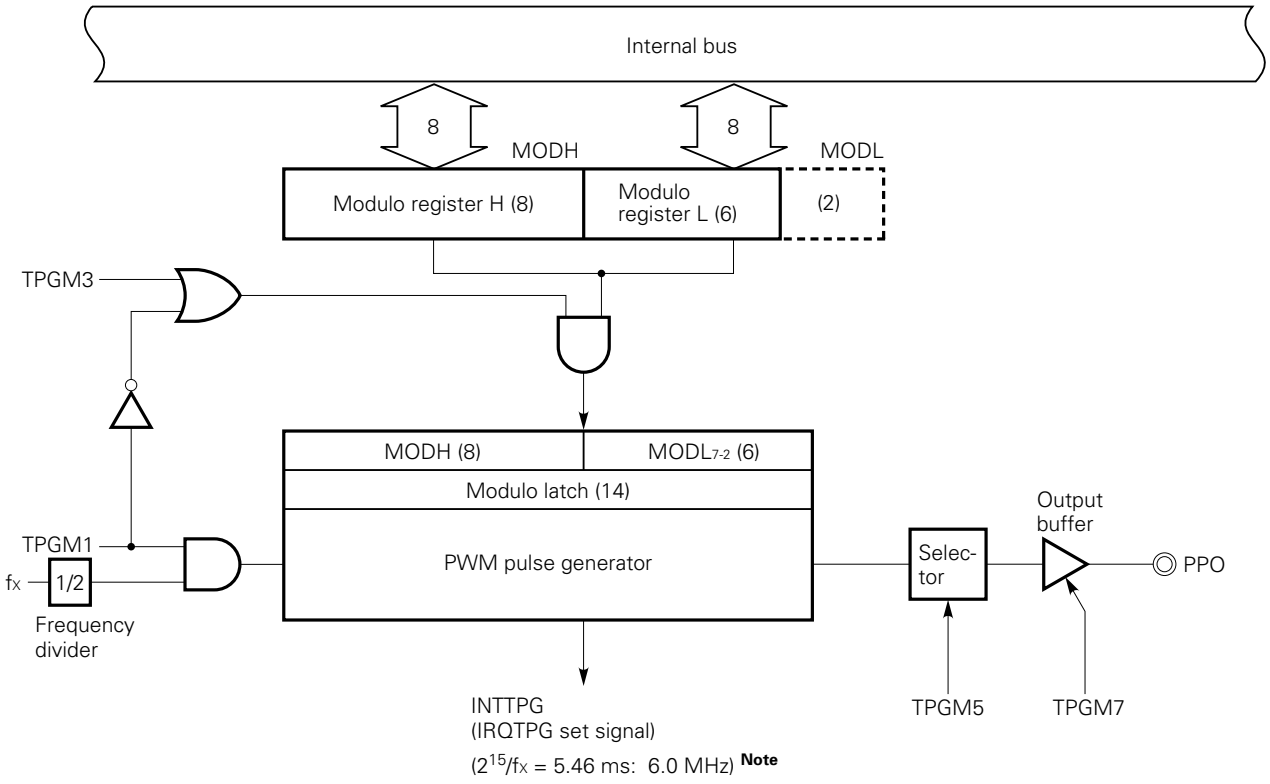
**Notes 1.** 7.81 ms at 4.19 MHz

**2.** 244 μs at 4.19 MHz

**(5) Static output to the PPO pin**

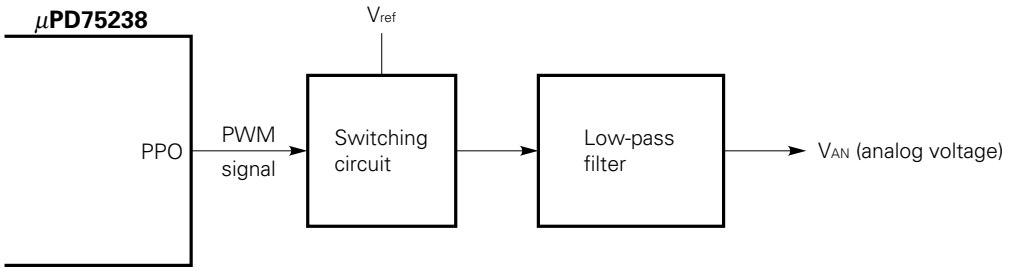
When pulse output is unnecessary, the PPO pin can be used as normal static output. In this case, the output data is set in TPGM4 with TPGM5 being set to 0 and TPGM7 to 1.

**Fig. 4-34 Block Diagram of the Timer/Pulse Generator (PWM Pulse Generation Mode)**




**Note** 7.81 ms at 4.19 MHz

**Fig. 4-35 Sample Configuration of D/A Conversion Using μPD75238**

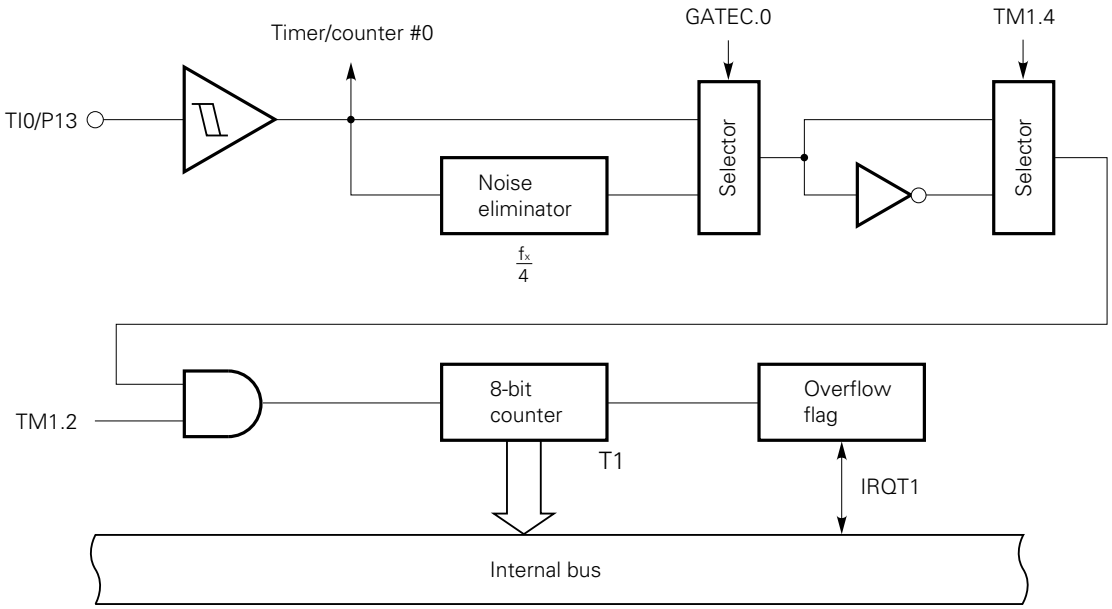


## 4.8 EVENT COUNTER

### (1) Configuration of the event counter

The event counter of the μPD75238 has a noise eliminator. Fig. 4-36 shows the configuration of the counter.

**Fig. 4-36 Block Diagram of the Event Counter**




**Caution The TI0/P13 pin is an external event pulse input pin shared between timer/event counter #0 and event counter #1.**

### (2) Event counter functions

The event counter provides the following functions:

(a) Event counter operation
(b) Function of reading the state of counting
(c) Count pulse edge specification
(d) Noise elimination function

### (3) Event counter mode register

The event counter mode register (TM1) is an 8-bit register that controls the event counter. Fig. 4-37 shows the format of the register.

The TM1 is set with an 8-bit memory manipulation instruction. Bit 3 is an event counter start bit, and can be set independently of the other bits. Bit 3 is automatically reset to 0 when the timer starts operation.

**Fig. 4-37 Format of the Event Counter Mode Register**

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FA8H | 0 | 0 | 0 | TM14 | TM13 | TM12 | 0 | 0 | TM1 |

**Event count operation enable/disable bit**

| | | |
|---|---|---|
| TM12 | 0 | Disables count operation (count value retained) |
| | 1 | Enables count operation |

**Event count start instruction bit**

| | |
|---|---|
| TM13 | When 1 is written, the counter and the IRQ1 flag are cleared.<br>IF TM12 is set to 1, count opertaion starts. |

**Count pulse edge specification**

| | | |
|---|---|---|
| TM14 | 0 | TI0 input rising edge |
| | 1 | TI0 input falling edge |

### (4) Overflow flag (IRQT1)

The overflow flag is set to 1 when the event counter (IRQT1) overflows. The flag is cleared to 0 by a count operation start instruction.

### (5) Event counter control register (GATEC)

This register specifies sampling by sampling clock (fx/4). A noise eliminator eliminates pulses narrower than two sampling clock cycles (8/fx) as noise and accepts pulses wider than as interrupt signals.

Fig. 4-38 shows the format of the GATEC.

**Fig. 4-38 Format of the Event Counter Control Register**

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| FABH | 0 | 0 | 0 | GATEC0 | GATEC |

| | |
|---|---|
| 0 | No sampling |
| 1 | Sampling at fx/4 |

### 4.9 SERIAL INTERFACE

The μPD75238 has two channels of clock synchronous 8-bit serial interface: Channel 0 and channel 1. Table 4-5 lists the differences between channel 0 and channel 1.

**Table 4-5 Differences between Channel 0 and Channel 1**

| Serial transfer mode, function | | Channel 0 | Channel 1 |
|---|---|---|---|
| 3-wire serial I/O | Clock selection | $f_X/2^4$, $f_X/2^3$, TOUT F/F, external clock | $f_X/2^4$, $f_X/2^3$, external clock |
| | Transfer method | Start bit switchable: MSB/LSB | Start bit: MSB |
| | Transfer end flag | Serial transfer end interrupt request flag (IRQCSI0) | Serial transfer end flag (EOT) |
| 2-wire serial I/O | | Available | Not available |
| Serial bus interface | | | |

**(1) Serial interface (channel 0) functions**

The serial interface (channel 0) of the μPD75238 has following four different modes.
The functions of the four modes are outlined below.

- **Operation halt mode**

  This mode is used when serial transfer is not performed. This mode reduces power consumption.

- **Three-wire serial I/O mode**

  In this mode, 8-bit data is transferred through three lines: Serial clock ($\overline{SCK0}$), serial output (SO0), and serial input (SI0).

  The three-wire serial I/O mode allows full-duplex transmission, so data transfer can be performed at higher speed.

  The user can choose 8-bit data transfer starting with the MSB or LSB, so devices starting with either the MSB or LSB can be connected.

  The three-wire serial I/O mode enables connections to be made with the 75X series, 78K series, and many other types of peripheral I/O devices.

- **Two-wire serial I/O mode**

  In this mode, 8-bit data is transferred through two lines: Serial clock ($\overline{SCK0}$) and serial data bus (SB0 or SB1). By controlling output levels on the two lines by software, communication with multiple devices is enabled.

  The output levels of $\overline{SCK0}$ and SB0 (or SB1) can be controlled by software, so the user can match an arbitrary transfer format. This means that a line that has been required for handshaking to connect multiple lines can be eliminated for more efficient I/O port utilization.

- **Serial bus interface (SBI) mode**

  In this mode, communication with multiple devices can be performed using two lines: Serial clock ($\overline{SCK0}$) and serial data bus (SB0 or SB1).

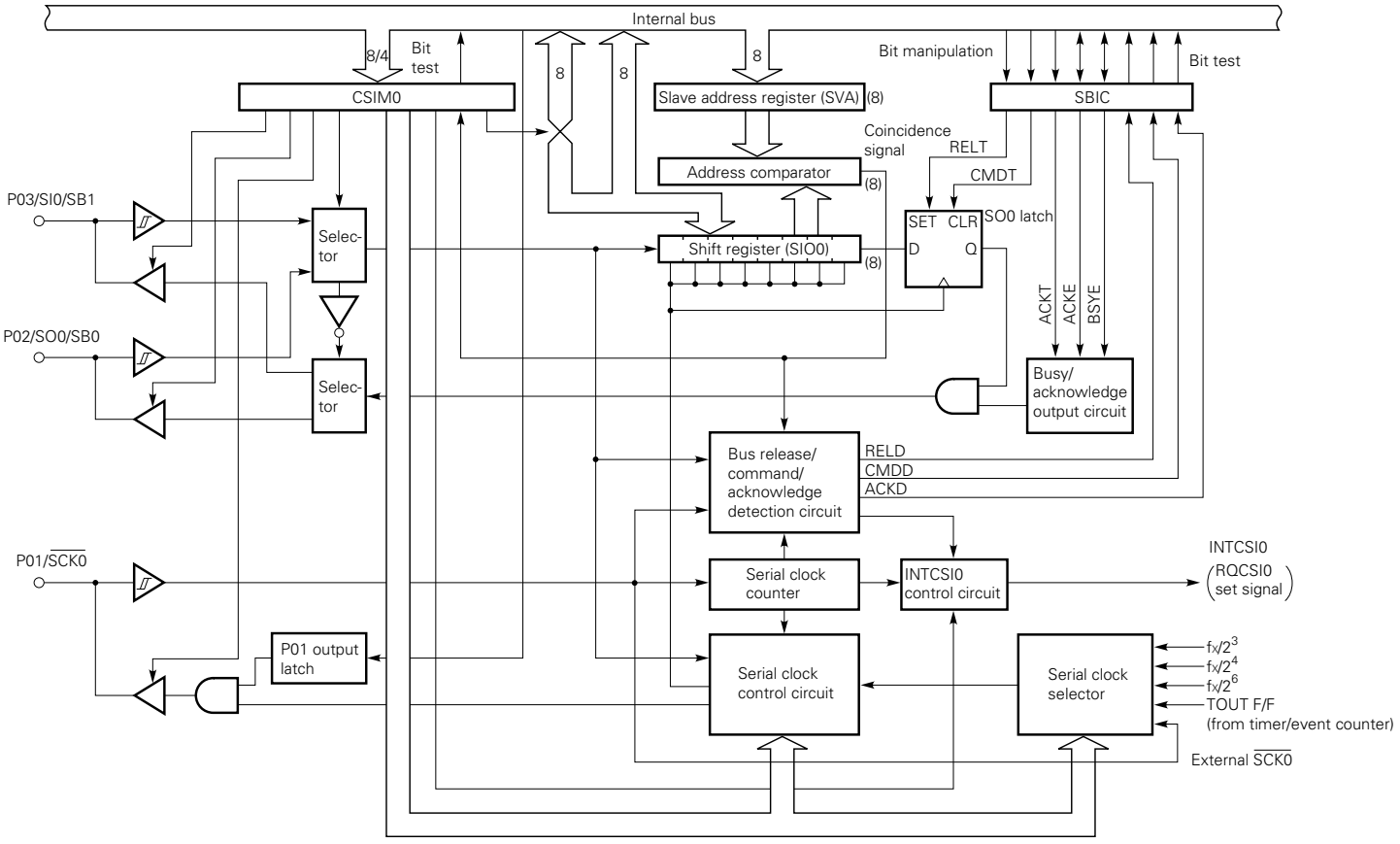
  This mode conforms to the NEC serial bus format.

  In this mode, the sender can output, on the serial data bus, an address for selecting a device subject to serial communication, commands directed to the remote device, and data. The receiver can identify an address, commands, and data from received data by hardware. This function enables more efficient I/O port utilization as in the case of the two-wire serial I/O mode. In addition, this function can simplify the serial interface control portion of an application program.

**(2) Configuration of serial interface (channel 0)**

Fig. 4-39 shows the block diagram of the serial interface (channel 0).

**Fig. 4-39 Block Diagram of the Serial Interface (Channel 0)**



### (3) Functions of serial interface (channel 0) registers

#### (a) Serial operation mode register 0 (CSIM0)

Fig. 4-40 shows the format of serial operation mode register 0 (CSIM0).

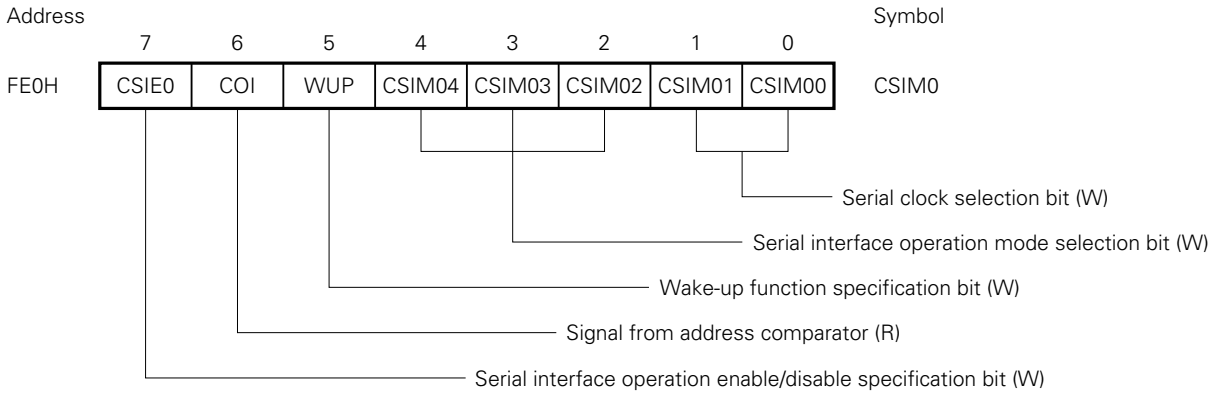
CSIM0 is an 8-bit register which specifies a serial interface (channel 0) operation mode, serial clock, wake-up function, and so forth.

CSIM0 is manipulated using an 8-bit memory manipulation instruction. The higher three bits can be manipulated bit by bit. Each bit can be manipulated using its name.

Each bit may or may not allow read and/or write operation. (See **Fig. 4-40**.) Bit 6 allows bit test operation only; any data written to this bit is invalid.

A $\overline{\text{RESET}}$ input clears all bits to 0.

**Fig. 4-40 Format of Serial Operation Mode Register 0 (CSIM0) (1/3)**




**Remark** (R) : Read only
(W) : Write only

**Fig. 4-40 Format of Serial Operation Mode Register 0 (CSIM0) (2/3)**

**Serial clock selection bit (W)**

| CSIM01 | CSIM00 | Serial clock | | | $\overline{SCK0}$ pin mode |
|---|---|---|---|---|---|
| | | 3-wire serial I/O mode | SBI mode | 2-wire serial I/O mode | |
| 0 | 0 | External clock applied to $\overline{SCK0}$ pin | | | Input |
| 0 | 1 | Time/event counter output (T0) | | | Output |
| 1 | 0 | $f_X/2^4$ (262 kHz or 375 kHz)**Note** | | $f_X/2^6$ (65.5 kHz or 93.8 kHz)**Note** | |
| 1 | 1 | $f_X/2^3$ (524 kHz or 750 kHz)**Note** | | | |

**Note** The values in parentheses are for fx = 4.19 MHz or 6.0 MHz.

**Serial interface operation mode selection bit (W)**

| CSIM04 | CSIM03 | CSIM02 | Operation mode | Bit sequence for shift register 0 | SO0 pin function | SI0 pin function |
|---|---|---|---|---|---|---|
| × | 0 | 0 | 3-wire serial I/O mode | $SIO0_{7-0} \leftrightarrow XA$ (Transfer starting with MSB) | SO0/P02 (CMOS output) | SI0/P03 (Input) |
| | | 1 | | $SIO0_{0-7} \leftrightarrow XA$ (Transfer starting with LSB) | | |
| 0 | 1 | 0 | SBI mode | $SIO0_{7-0} \leftrightarrow XA$ (Transfer starting with MSB) | SB0/P02 (N-ch open-drain input/output) | P03 input |
| 1 | | | | | P02 input | SB1/P03 (N-ch open-drain input/output) |
| 0 | 1 | 1 | 2-wire serial I/O mode | $SIO0_{7-0} \leftrightarrow XA$ (Transfer starting with MSB) | SB0/P02 (N-ch open-drain input/output) | P03 input |
| 1 | | | | | P02 input | SB1/P03 (N-ch open-drain input/output) |

**Remark** ×: Don't care

**Wake-up function specification bit (W)**

| WUP | | |
|---|---|---|
| | 0 | Sets IRQCSI0 each time serial transfer is completed in each mode. |
| | 1 | Used in the SBI mode only to set IRQCSI0 only when an address received after bus release matches the data in the slave address register (wake-up state). SB0/SB1 goes to high-impedance state. |

**Caution When WUP = 1 is set during $\overline{BUSY}$ signal output, $\overline{BUSY}$ is not released. In the SBI mode, the $\overline{BUSY}$ signal is output until the next falling edge of the serial clock ($\overline{SCK0}$) appears after release of $\overline{BUSY}$ is directed. Before setting WUP = 1, be sure to confirm that the SB0 (or SB1) pin is high after releasing $\overline{BUSY}$.**

**Fig. 4-40 Format of Serial Operation Mode Register 0 (CSIM0) (3/3)**

**Signal from address comparator (R)**

| COI[Note] | Condition for being cleared (COI = 0) | Condition for being set (COI = 1) |
|---|---|---|
| | When the slave address register (SVA) does not match the data of the shift register | When the slave address register (SVA) matches the data of the shift register |

**Note** COI can be read only before serial transfer is started or after serial transfer is completed. An undefined value may be read during transfer.
COI data written by an 8-bit manipulation instruction is ignored.

**Serial interface operation enable/disable specification bit (W)**

| | | Shift register 0 operation | Serial clock counter | IRQCSI0 flag | SO0/SB0, SI0/SB1 pin |
|---|---|---|---|---|---|
| CSIE0 | 0 | Shift operation disabled | Cleared | Held | Used only for port 0 |
| | 1 | Shift operation enabled | Count operation | Can be set. | Used in each mode as well as for port 0 |

**Remarks 1.** Each mode can be selected by setting CSIE0, CSIM03, and CSIM02.

| CSIE0 | CSIM03 | CSIM02 | Operation mode |
|---|---|---|---|
| 0 | × | × | Operation halt mode |
| 1 | 0 | × | Three-wire serial I/O mode |
| 1 | 1 | 0 | SBI mode |
| 1 | 1 | 1 | Two-wire serial I/O mode |

**2.** The P01/$\overline{\text{SCK0}}$ pin assumes the following state according to the setting of CSIE0, CSIM01, and CSIM00:

| CSIE0 | CSIM01 | CSIM00 | P01/$\overline{\text{SCK0}}$ pin state |
|---|---|---|---|
| 0 | 0 | 0 | Input port |
| 1 | 0 | 0 | High impedance |
| 0 | 0 | 1 | High level output |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 1 | 0 | 1 | Serial clock output (High level output) |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | |

**Remarks 3.** When clearing CSIE0 during serial transfer, use the following procedure:

① Disable interrupts by clearing the interrupt enable flag.
② Clear CSIE0.
③ Clear the interrupt request flag.

**Examples 1.** $f_X/2^4$ is selected as the serial clock, serial interrupt IRQCSI0, is generated each time serial transfer is completed, and serial transfer is performed in the SBI mode with the SB0 pin used as the serial data bus.

```
SEL     MB15            ; or CLR1  MBE
MOV     XA, #10001010B
MOV     CSIM0, XA       ; CSIM0 ← 10001010B
```

**2.** Serial transfer dependent on the contents of CSIM0 is enabled.

```
SEL     MB15            ; or CLR1  MBE
SET1    CSIE0
```

**(b) Serial bus interface control register (SBIC)**

Fig. 4-41 shows the format of the serial bus interface control register (SBIC).

SBIC is an 8-bit register consisting of bits for controlling the serial bus and flags for indicating the states of input data from the serial bus. SBIC is used mainly in the SBI mode.

SBIC is manipulated using a bit manipulation instruction. SBIC cannot be manipulated using a 4-bit or 8-bit memory manipulation instruction.

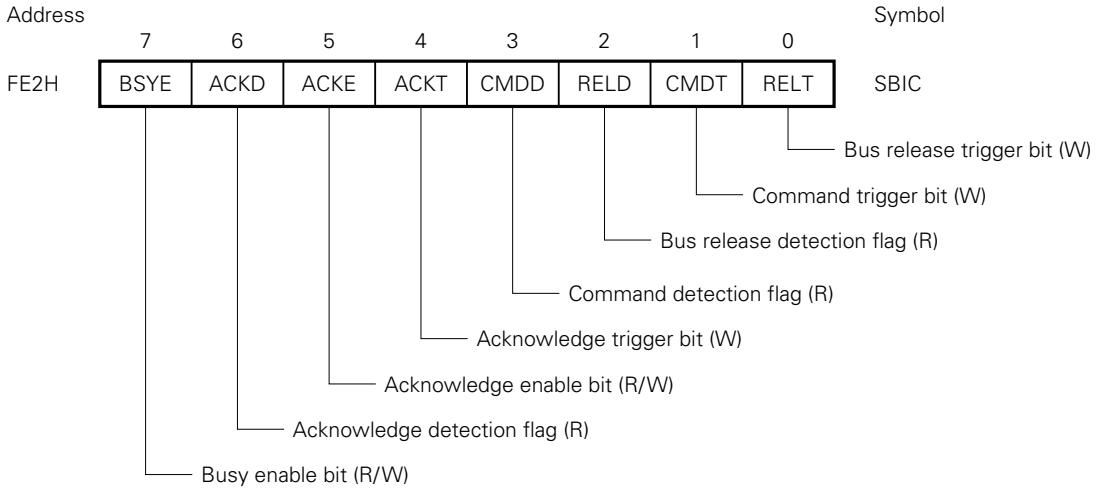
Each bit may or may not allow read and/or write operation. (See **Fig. 4-41**.)

A $\overline{\text{RESET}}$ input clears all bits to 0.

**Caution Only the following bits can be used in the three-wire and two-wire serial I/O modes:**

- **Bus release trigger bit (RELT): Sets the SO0 latch.**
- **Command trigger bit (CMDT): Clears the SO0 latch.**

**Fig. 4-41 Format of Serial Bus Interface Control Register (SBIC) (1/3)**




**Remarks 1.** (R) : Read only

**2.** (W) : Write only

**3.** (R/W): Read/write

**Fig. 4-41 Format of Serial Bus Interface Control Register (SBIC) (2/3)**

**Bus release trigger bit (W)**

| | |
|---|---|
| RELT | Control bit for bus release signal (REL) trigger output.<br>By setting RELT = 1, the SO0 latch is set to 1. Then the RELT bit is automatically cleared to 0. |

**Caution Never clear SB0 (or SB1) during serial transfer. Be sure to clear SB0 (or SB1) before or after serial transfer.**

**Command trigger bit (W)**

| | |
|---|---|
| CMDT | Control bit for command signal (CMD) trigger output.<br>By setting CMDT = 1, the SO0 latch is cleared to 0. Then the CMDT bit is automatically cleared to 0. |

**Caution Never clear SB0 (or SB1) during serial transfer. Be sure to clear SB0 (or SB1) before or after serial transfer.**

**Bus release detection flag (R)**

| | Condition for being cleared (RELD = 0) | Condition for being set (RELD = 1) |
|---|---|---|
| RELD | ① The transfer start instruction is executed.<br>② The $\overline{\text{RESET}}$ signal is entered.<br>③ CSIE0 = 0 (See Fig. 4-40.)<br>④ SVA does not match SIO0 when an address is received. | The bus release signal (REL) is detected. |

**Command detection flag (R)**

| | Condition for being cleared (CMDD = 0) | Condition for being set (CMDD = 1) |
|---|---|---|
| CMDD | ① The transfer start instruction is executed.<br>② The bus release signal (REL) is detected.<br>③ The $\overline{\text{RESET}}$ signal is entered.<br>④ CSIE0 = 0 (See Fig. 4-40.) | The command signal (CMD) is detected. |

**Acknowledge trigger bit (W)**

| | |
|---|---|
| ACKT | When set after transfer, $\overline{\text{ACK}}$ is output in phase with the next $\overline{\text{SCK0}}$. After $\overline{\text{ACK}}$ signal output, this bit is automatically cleared to 0. |

**Cautions 1. Never set ACKT before or during serial transfer.**
**2. ACKT cannot be cleared by software.**
**3. Before setting ACKT, set ACKE = 0.**

**Acknowledge enable bit (R/W)**

| | | | |
|---|---|---|---|
| ACKE | 0 | Disables automatic output of the acknowledge signal ($\overline{\text{ACK}}$). (Output by ACKT is possible.) | |
| | 1 | When set before transfer | $\overline{\text{ACK}}$ is output in phase with the 9th clock of $\overline{\text{SCK0}}$. |
| | | When set after transfer | $\overline{\text{ACK}}$ is output in phase with $\overline{\text{SCK0}}$ immediately following set instruction execution. |

**Fig. 4-41 Format of Serial Bus Interface Control Register (SBIC) (3/3)**

**Acknowledge detection flag (R)**

| ACKD | Condition for being cleared (ACKD = 0) | Condition for being set (ACKD = 1) |
|---|---|---|
| | ① The transfer start instruction is executed.<br>② The $\overline{\text{RESET}}$ signal is entered. | The acknowledge signal ($\overline{\text{ACK}}$) is detected (in phase with the rising edge of $\overline{\text{SCK0}}$). |

**Busy enable bit (R/W)**

| BSYE | | |
|---|---|---|
| | 0 | ① The busy signal is automatically disabled.<br>② Busy signal output is stopped in phase with the falling edge of $\overline{\text{SCK0}}$ immediately after clear instruction execution. |
| | 1 | The busy signal is output after the acknowledge signal in phase with the falling edge of $\overline{\text{SCK0}}$. |

**Examples 1.** A command signal is output.

```
SEL     MB15      ; or CLR1  MBE
SET1    CMDT
```

**2.** RELD and CMDD are tested to identify the types of received data and the types of processing accordingly.

By setting WUP = 1, this interrupt routine is processed only when an address match is found.

```
        SEL     MB15
        SKF     RELD      ; RELD test
        BR      !ADRS
        SKT     CMDD      ; CMDD test
        BR      !DATA
CMD  :  ·····             ; Command analysis
DATA :  ·····             ; Data processing
ADRS :  ·····             ; Address decode
```

**(c) Shift register (SIO0)**

Fig. 4-42 shows the configuration of peripheral hardware of shift register 0. SIO0 is an 8-bit register which performs parallel-serial conversion and serial transfer (shift) operation in phase with the serial clock.

Serial transfer is started by writing data to SIO0.

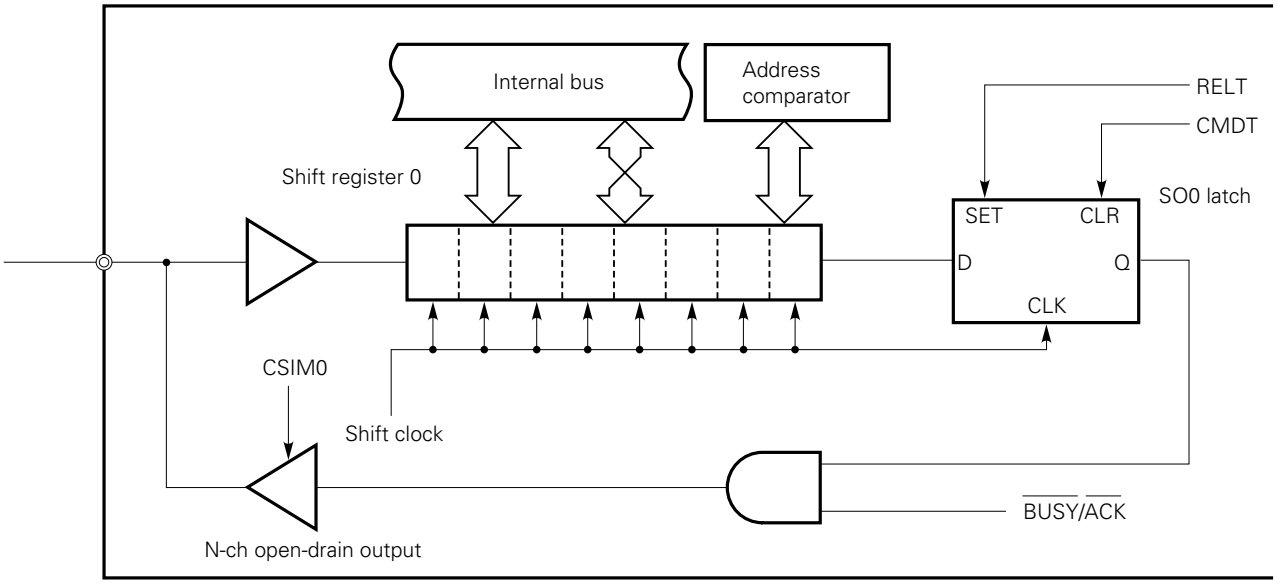
In send operation, data written to SIO0 is output on the serial output (SO0) or serial data bus (SB0/SB1). In receive operation, data is read from the serial input (SI0) or SB0/SB1 into SIO0.

Data can be read from or written to SIO0 by using an 8-bit manipulation instruction.

When the $\overline{\text{RESET}}$ signal is entered during operation, the value of SIO0 is undefined. When the $\overline{\text{RESET}}$ signal is entered in the standby mode, the value of SIO0 is preserved.

Shift operation is stopped after 8-bit send or receive operation is completed.

**Fig. 4-42 Peripheral Hardware of Shift Register 0**




The timing for reading SIO0 and start of serial transfer (writing to SIO0) is as follows:

- When the serial interface operation enable/disable bit (CSIE0) = 1. However, the case where CSIE0 is set to 1 after data is written to the shift register is excluded.
- When the serial clock is masked after 8-bit serial transfer
- $\overline{\text{SCK0}}$ is high.

When reading from or writing to SIO0, make sure that $\overline{\text{SCK0}}$ is high.

In the two-wire serial I/O mode and SBI mode, the pins specified for the data bus are used for both input and output. Because the configuration of output pins is N-ch open-drain, write FFH in SIO0 for devices that are to receive data.

**(d) Slave address register (SVA)**

The slave address register (SVA) has the two functions described below.

SVA is manipulated using an 8-bit manipulation instruction. SVA allows only write operation.

When a $\overline{\text{RESET}}$ is entered, the value of SVA is undefined. However, the value of SVA is preserved when the $\overline{\text{RESET}}$ is entered in the standby mode.

- **Slave address detection**

  **[In the SBI mode]**

  SVA is used when the μPD75238 is connected as a slave device to the serial bus. SVA is an 8-bit register for a slave to set its slave address (number assigned to it). The master outputs a slave address to the connected slaves to select a particular slave. Two data values (a slave address output from the master and the value of SVA) are compared with each other by the address comparator. If a match is found, the slave is selected.

  At this time, bit 6 (COI) of serial operation mode register 0 (CSIM0) is set to 1.

  **Cautions 1. Slave selection or nonselection state is detected by detecting a match for a slave address received after bus release (in the state of RELD = 1).**

  **For this match detection, an address match interrupt (IRQCSI0) generated when WUP is set to 1 is usually used. So detect selection/nonselection state by slave address when WUP is set to 1.**

  **2. When detecting selection/nonselection state without using an interrupt when WUP is 0, do not use the address match detection method. Instead, use transfer of commands set in advance in a program.**

- **Error detection**

  **[In the two-wire serial I/O mode or SBI mode]**

  SVA detects an error in either of the following cases:

  - When addresses, commands, or data is transferred with the μPD75238 operating as the master
  - When data is transferred with the μPD75238 operating as a slave

**(4) Signals**

Table 4-6 lists signals. Fig. 4-43 to 4-48 show operations of signals and flags.

**Table 4-6 Various Signals Used in the SBI Mode (1/2)**

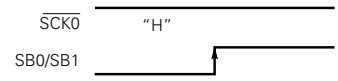
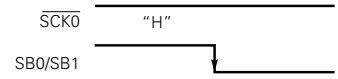
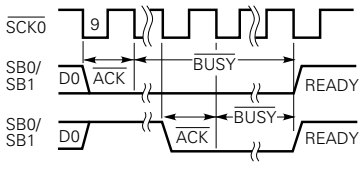
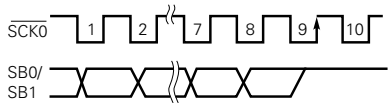
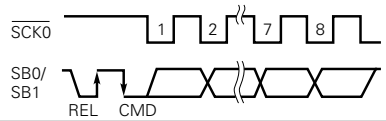
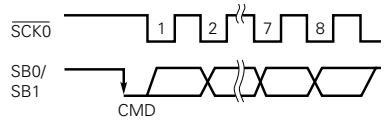
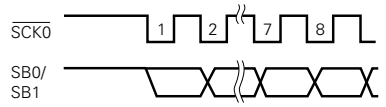
| Signal name | Output device | Definition | Timing chart | Condition for output | Flag operation | Meaning of signal |
|---|---|---|---|---|---|---|
| Bus release signal (REL) | Master | Rising edge of SB0/SB1 when $\overline{SCK0}$ = 1 |  | • RELT is set. | • RELD is set.<br>• CMDD is cleared. | Indicates that CMD signal follows and data sent is address data. |
| Command signal (CMD) | Master | Falling edge of SB0/SB1 when $\overline{SCK0}$ = 1 |  | • CMDT is set. | • CMDD is set. | i) Data sent after REL signal output is address.<br>ii) Data sent, with REL signal not being output, is command. |
| Acknowledge signal ($\overline{ACK}$) | Master/ slave | Low level signal output on SB0/SB1 during one $\overline{SCK0}$ clock cycle after serial receive operation is completed | [Synchronous busy signal output] | # ACKE = 1<br>$ ACKT is set. | • ACKD is set. | Indicates completion of receive operation. |
| Busy signal ($\overline{BUSY}$) | Slave | [Synchronous busy signal]<br>Low level signal output on SB0/SB1 after acknowledge signal |  | • BSYE = 1 | – | Indicates that serial receive operation is disabled because processing is in progress. |
| Ready signal (READY) | Slave | High level signal output on SB0/SB1 before serial transfer is started or after serial transfer is completed |  | # BSYE = 0<br>$ Execution of instruction to write data to SIO0 (direction to start transfer) | – | Indicates that serial receive operation is enabled. |

**Table 4-6 Various Signals Used in the SBI Mode (2/2)**

| Signal name | Output device | Definition | Timing chart | Condition for output | Flag operation | Meaning of signal |
|---|---|---|---|---|---|---|
| Serial clock ($\overline{SCK0}$) | Master | Synchronous clock for outputting address/ command/data, $\overline{ACK}$ signal, synchronous $\overline{BUSY}$ signal, and so on. Address/command/data is output during first 8 clock cycles. |  | Execution of instruction to write data to SIO0 when CSIE0 = 1 (direction to start serial transfer)Note 2 | IRQCSI0 is set (on rising edge of ninth clock)Note 1 | Timing of signal output on serial data bus |
| Address (A7 - A0) | Master | 8-bit data transferred in phase with $\overline{SCK0}$ after REL signal and CMD signal output |  | | | Address of slave device on serial bus |
| Command (C7 - C0) | Master | 8-bit data transferred in phase with $\overline{SCK0}$ after only CMD signal is output, with REL signal not being output |  | | | Directions and messages to slave device |
| Data (D7 - D0) | Master/ slave | 8-bit data transferred in phase with $\overline{SCK0}$, with neither REL signal nor CMD signal being output |  | | | Value processed by slave or master device |

**Notes 1.** When WUP = 0, IRQCSI0 is always set on the ninth rising edge of $\overline{SCK0}$.
When WUP = 1, IRQCSI0 is set on the ninth rising edge of $\overline{SCK0}$ only if a received address matches the value of the slave address register (SVA).

**2.** If the $\overline{BUSY}$ state is present, data transfer is started after the READY state is set.

**Fig. 4-43 Operations of RELT, CMDT, RELD, and CMDD (Master)**

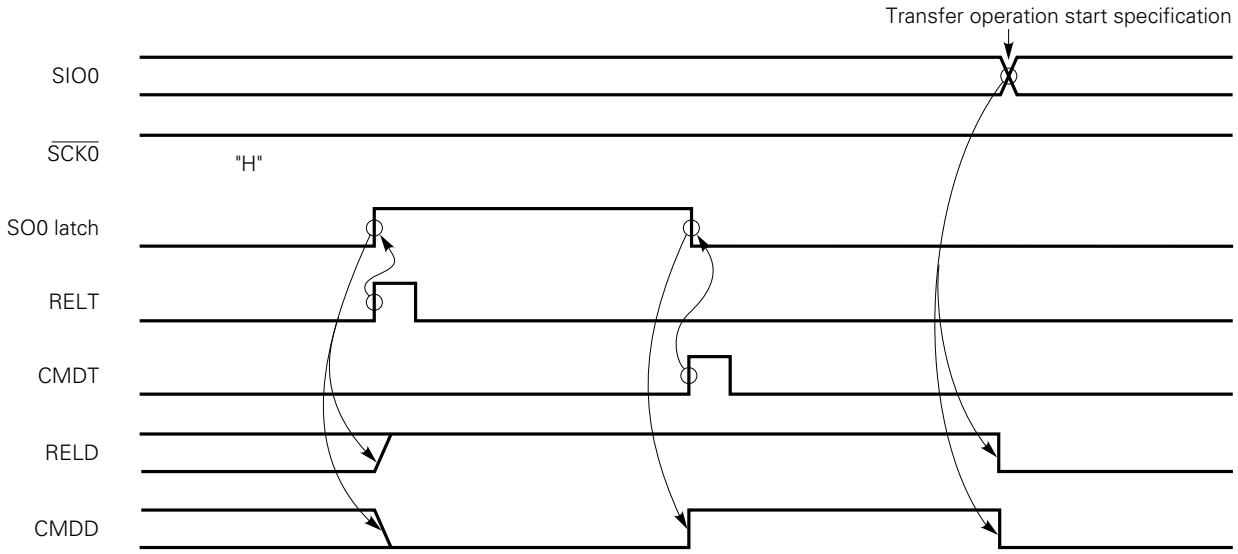



**Fig. 4-44 Operations of RELT, CMDT, RELD, and CMDD (Slave)**

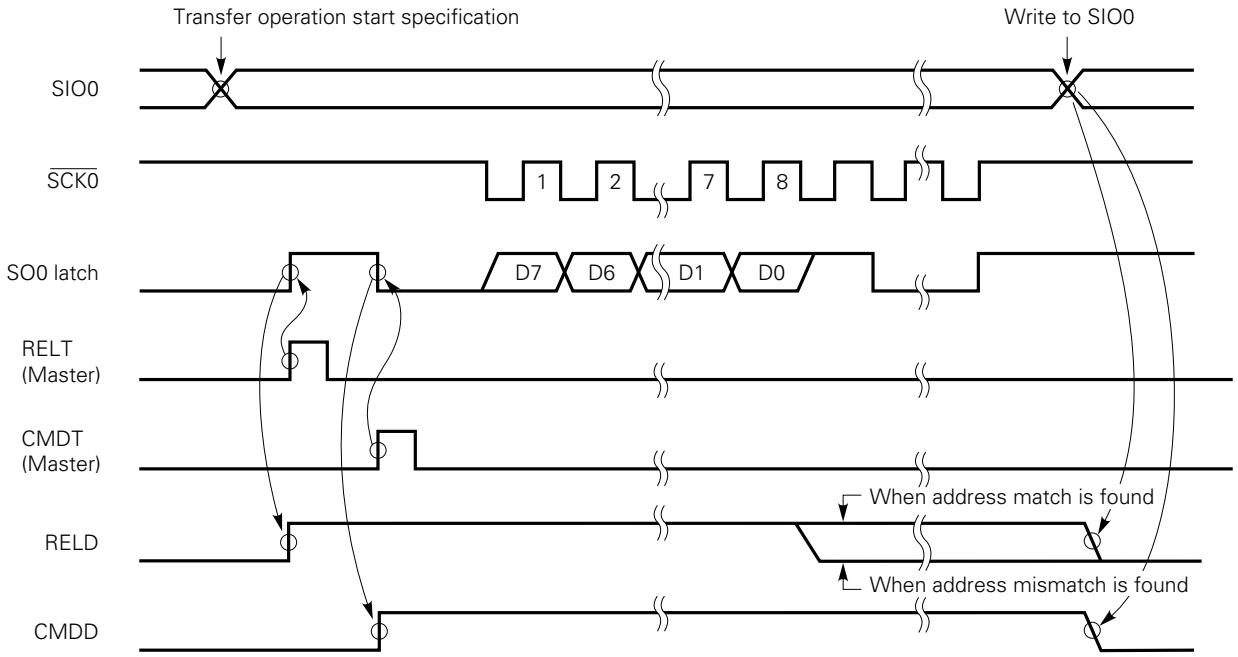
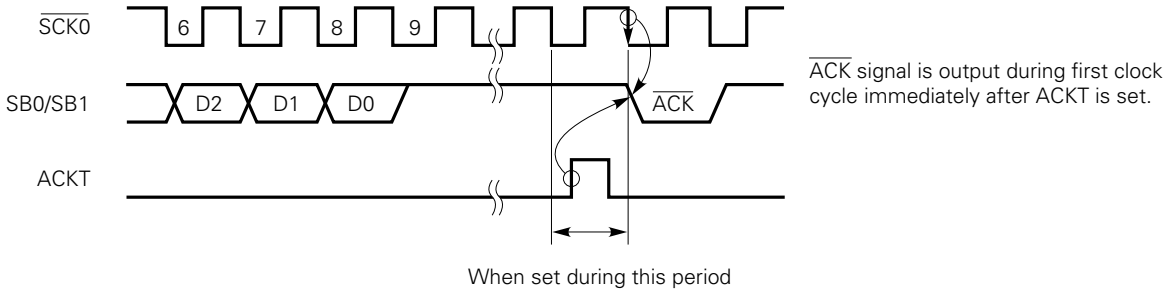



**Fig. 4-45 Operation of ACKT**

Set after transfer completion.

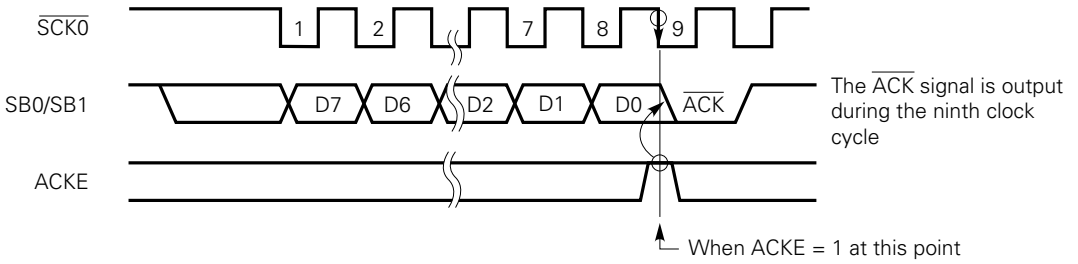



ACK signal is output during first clock cycle immediately after ACKT is set.

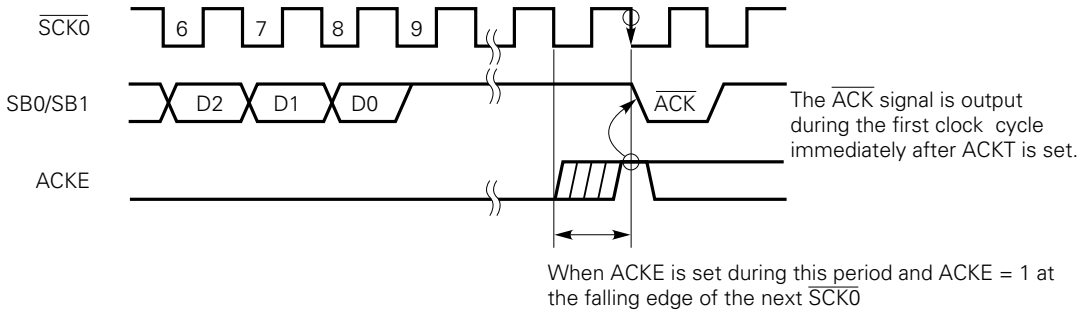
**Caution Do not set the ACKT until the transfer is completed.**

**Fig. 4-46 Operation of ACKE**

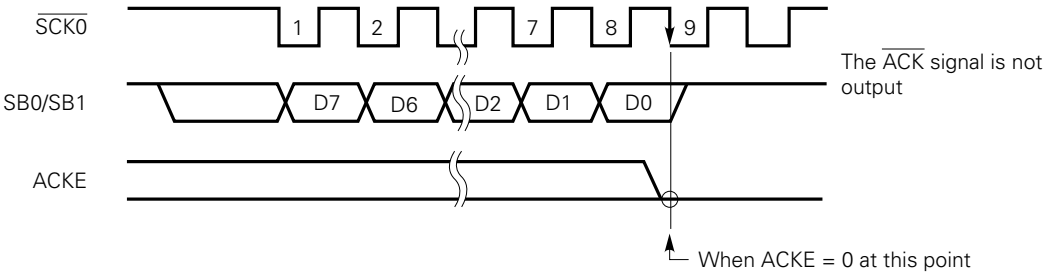
**(a) When ACKE = 1 at time of transfer operation completion**




**(b) When ACKE is set after transfer operation completion**




**(c) When ACKE = 0 at time of transfer operation completion**




**(d) When ACKE = 1 period is too short**

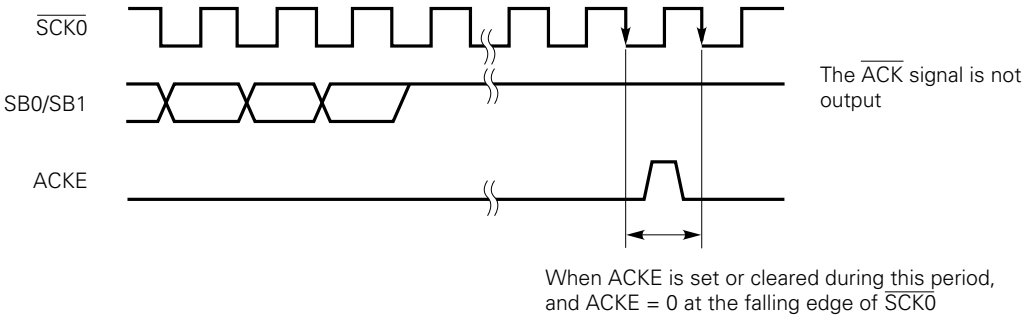
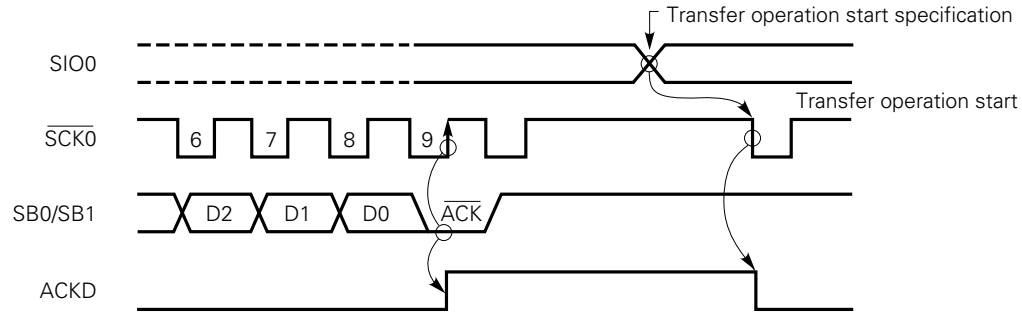
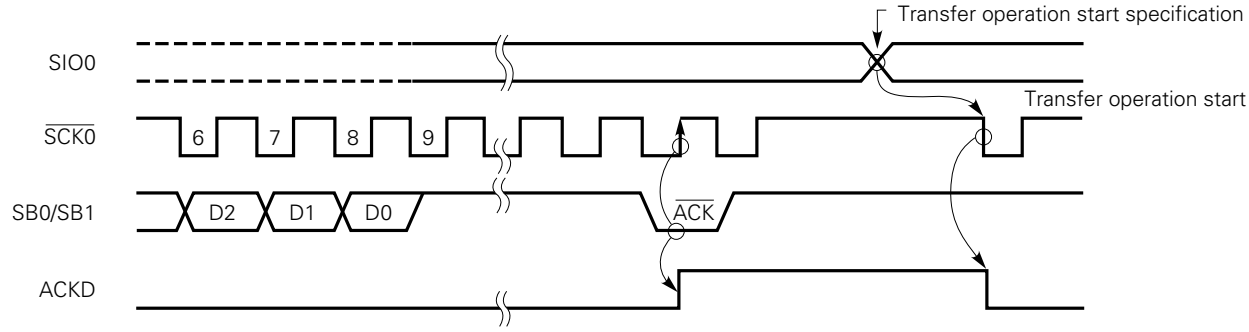


**Fig. 4-47 Operation of ACKD**

**(a) When $\overline{ACK}$ signal is output during ninth $\overline{SCK0}$ clock**




**(b) When $\overline{ACK}$ signal is output after ninth $\overline{SCK0}$ clock**




**(c) Clear timing for case where start of transfer is directed during $\overline{BUSY}$**

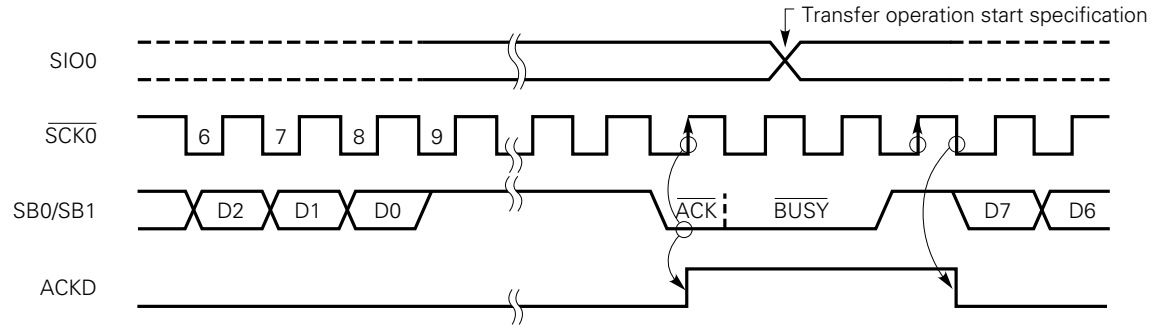
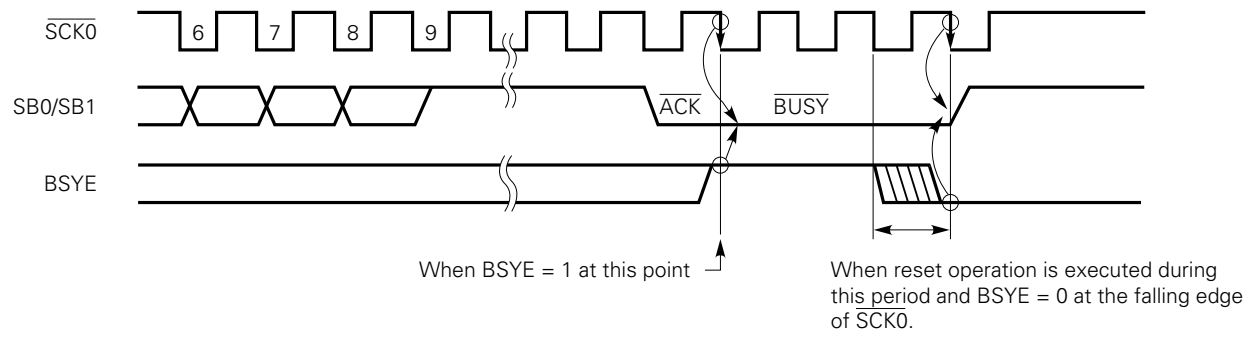



**Fig. 4-48 Operation of BSYE**



### (5) Serial interface (channel 0) operation

#### (a) Operation halt mode

The operation halt mode is used when serial transfer is not performed. This mode reduces power consumption.

The shift register 0 does not perform shift operation in this mode, so the shift register can be used as a normal 8-bit register.

A $\overline{\text{RESET}}$ input sets the operation halt mode. The P02/SO0/SB0 pin and P03/SI0/SB1 pin function as input-only port pins. The P01/$\overline{\text{SCK0}}$ pin can be used as an input port pin by setting the serial operation mode register 0.

#### (b) Three-wire serial I/O mode operations

The three-wire serial I/O mode is compatible with other modes used in the 75X series and 78K series. Communication is performed using three lines: Serial clock ($\overline{\text{SCK0}}$), serial output (SO0), and serial input (SI0).

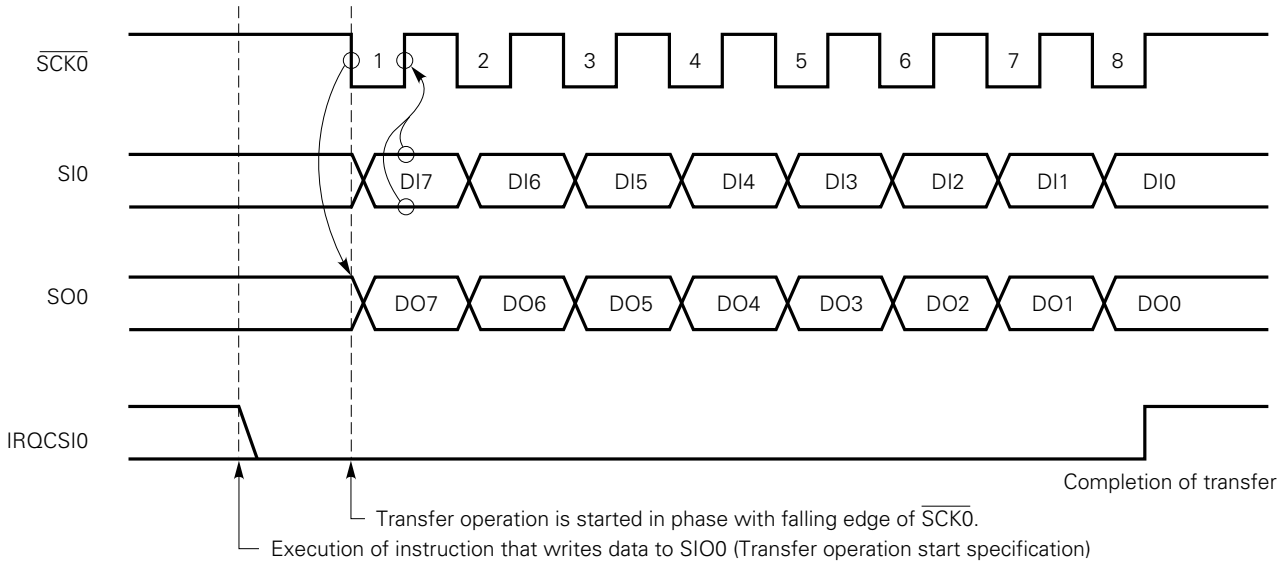
##### (i) Communication operation

The three-wire serial I/O mode transfers data, with eight bits as one block. Data is transferred bit by bit in phase with the serial clock.

The shift register performs shift operation on the falling edge of the serial clock ($\overline{\text{SCK0}}$). Send data is latched on the SO0 latch, and is output on the SO0 pin. Receive data applied to the SI0 pin is latched in the shift register 0 on the rising edge of $\overline{\text{SCK0}}$.

When eight bits have been transferred, shift register 0 operation automatically terminates setting the interrupt request flag (IRQCSI0).

**Fig. 4-49 Timing of Three-Wire Serial I/O Mode**

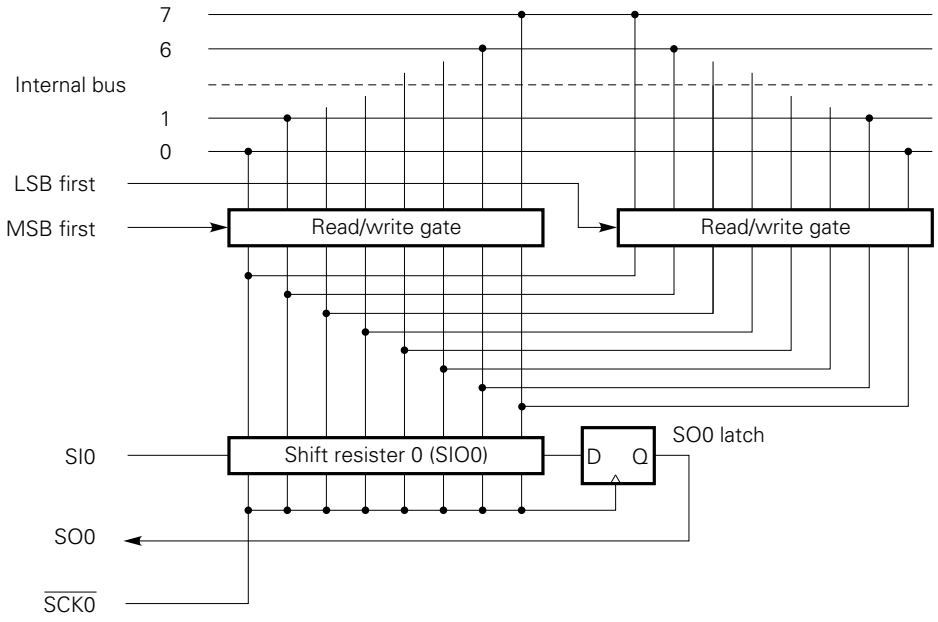


The SO0 pin becomes a CMOS output and outputs the state of the SO0 latch. So the output state of the SO0 pin can be manipulated by setting the RELT bit and CMDT bit.
However, this manipulation must not be performed during serial transfer operation.
The output state of the $\overline{SCK0}$ pin can be controlled by manipulating the P01 output latch in the output mode (internal system clock mode). (See **(7) in Section 4.9**.)

**(ii) Switching between MSB and LSB as the first transfer bit**

The three-wire serial I/O mode has a function that can switch between the MSB and LSB as the first bit of transfer.
Fig. 4-50 shows the configuration of shift register 0 (SIO0) and internal bus. As shown in Fig. 4-50, read or write operation can be performed by switching between the MSB and LSB.
This switching can be specified using bit 2 of serial operation mode register 0 (CSIM0).

**Fig. 4-50 Transfer Bit Switching Circuit**




The first bit is switched by changing the order of data bits written to shift register 0 (SIO0). The shift operation order of SIO0 is always the same.
Accordingly, the first bit must be switched between the MSB and LSB before writing data to the shift register 0.

**(c) Two-wire serial I/O mode**

The two-wire serial I/O mode can be made compatible with any communication format by programming. In this mode, communication is basically performed using two lines: Serial clock ($\overline{SCK0}$) and serial data input/output (SB0 or SB1).

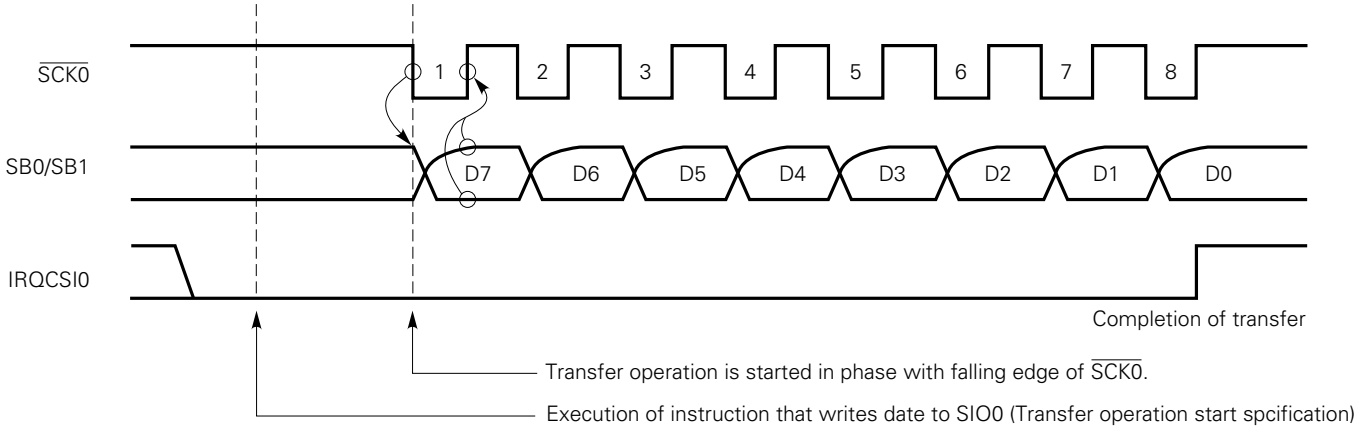
**(i) Communication operation**

The two-wire serial I/O mode transfers data, with eight bits as one block. Data is transferred bit by bit in phase with the serial clock.

The shift register 0 performs shift operation on the falling edge of the serial clock ($\overline{SCK0}$). Send data is latched on the SO0 latch, and is output on the SB0/P02 pin or SB1/P03 pin starting with the MSB. Receive data applied to the SB0 pin or SB1 pin is latched in the shift register 0 on the rising edge of $\overline{SCK0}$.

When eight bits have been transferred, shift register 0 operation automatically terminates setting the interrupt request flag (IRQCSI0).

**Fig. 4-51 Timing of Two-Wire Serial I/O Mode**




The SB0 or SB1 pin becomes an N-ch open-drain I/O when specified as the serial data bus, so the voltage level on that pin must be pulled up externally.

The state of the SO0 latch is output on the SB0 or SB1 pin, so the SB0 or SB1 pin output states can be controlled by setting the RELT or CMDT bit.

However, this operation must not be performed during serial transfer operation.

The output state of the $\overline{SCK0}$ pin can be controlled by manipulating the P01 output latch in the output mode (internal system clock mode). (See **(7) in Section 4.9**.)

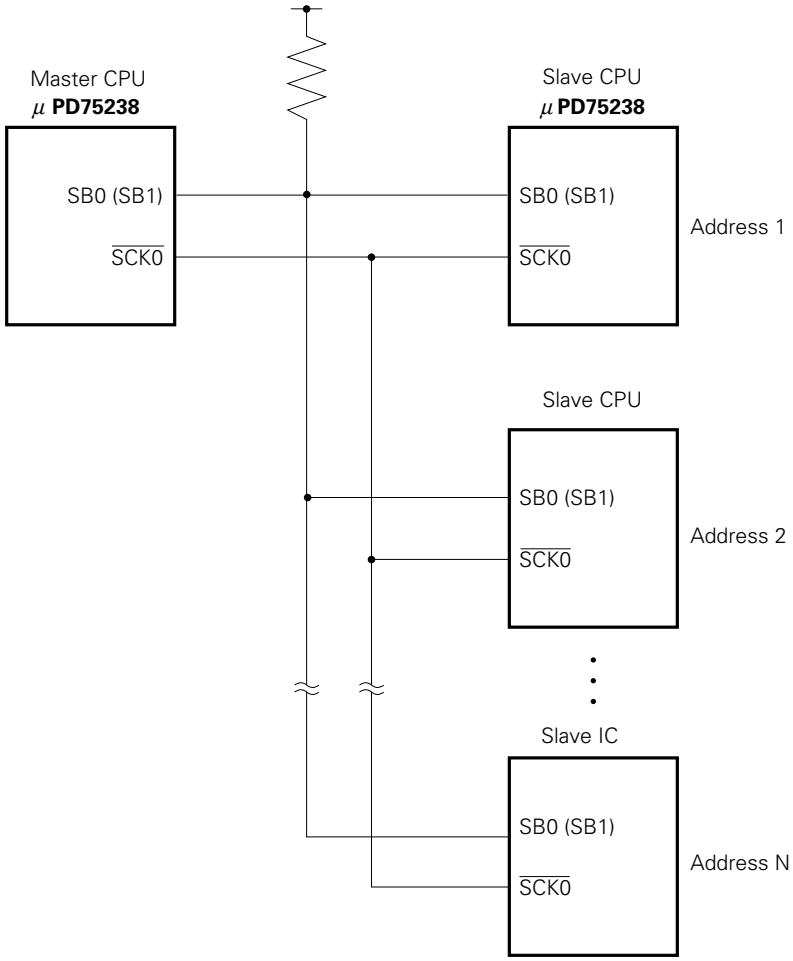
**(d) SBI mode operation**

The SBI (serial bus interface) is a high-speed serial interface that conforms to the NEC serial bus format.

To allow communication with multiple devices on a single-master and high-speed serial bus using two signal lines, the SBI has a bus configuration function added to the clock synchronous serial I/O method. So the SBI can reduce ports and wires on boards when multiple microcomputers and peripheral ICs are used to configure a serial bus.

Fig. 4-52 is an example of the SBI system configuration.

**Fig. 4-52 Example of SBI System Configuration**




**Cautions 1. In the SBI mode, the serial data bus pin SB0 (or SB1) is an open-drain output. So the serial data bus line is placed in the wired OR state. A pull-up resistor is required for the serial data bus line.**

**2. To switch between the master and slave, a pull-up resistor is required also for the serial clock line ($\overline{SCK0}$), because $\overline{SCK0}$ input/output switching is performed between the master and slave asynchronously.**

**(i) SBI functions**

- Address/command/data identification function
  Serial data is classified into three types: Address, command, and data.

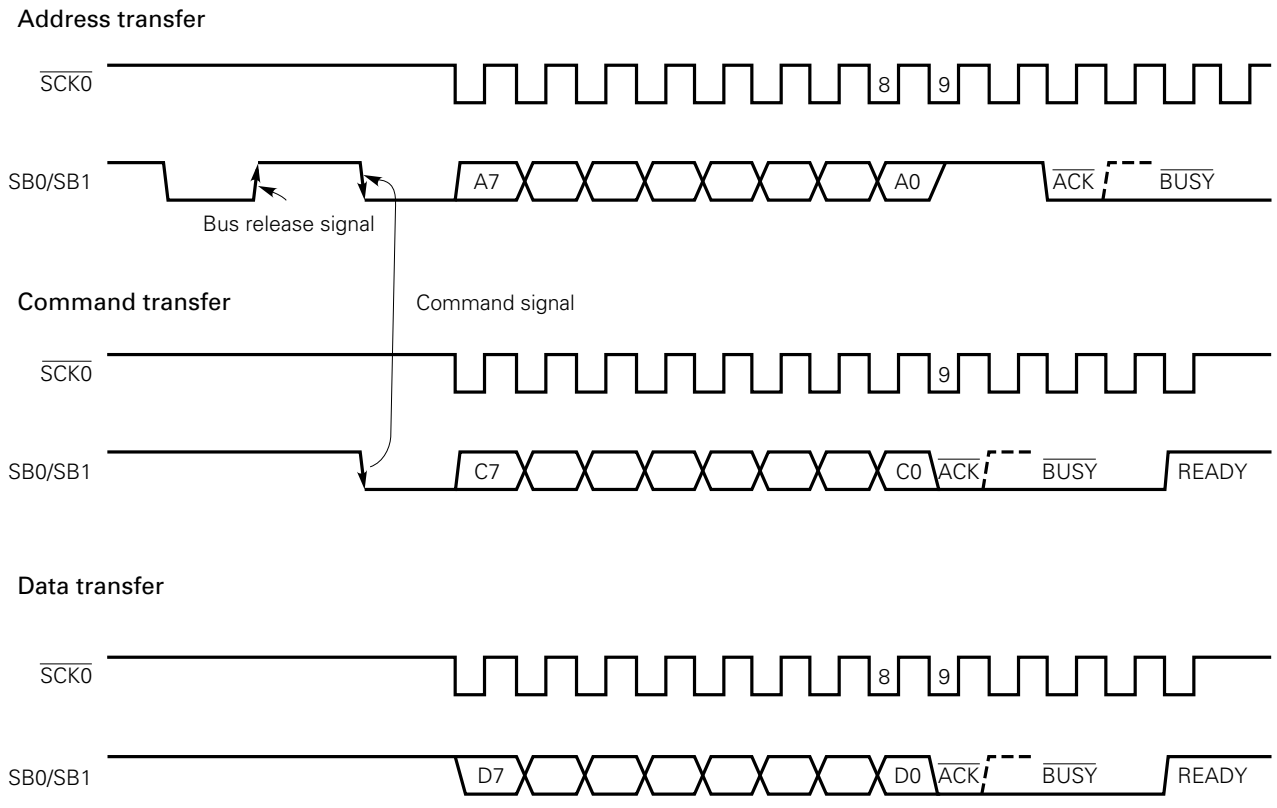
- Address-based chip select function
  The master selects a chip by address transfer.

- Wake-up function
  A slave can easily check address reception (for chip select identification) with the wake-up function. This function can be set or released by software.
  When the wake-up function is set, an interrupt (IRQCSI0) is generated when a match address is received. For this reason, in communication with multiple devices, a CPU other than a selected slave can operate independently of serial communication.

- Acknowledge signal ($\overline{ACK}$) control function
  The acknowledge signal, which is used to confirm the reception of serial data, can be controlled.

- Busy signal ($\overline{BUSY}$) control function
  The busy signal, which is used to post the busy state of a slave, can be controlled.

**Fig. 4-53 Timing of SBI Transfer**



**(ii) Communication operation**

In the SBI mode, the master usually selects a slave device to communicate with from multiple devices by outputting the address of the slave in the serial bus.

After selecting a device to communicate with, the master exchanges commands and data with the slave device, thus establishing serial communication.

Fig. 4-54 to 4-57 show the timing charts of data communication operations.

In the SBI mode, the shift register 0 performs shift operation on the falling edge of the serial clock ($\overline{SCK0}$). Send data is held on the SO0 latch, and is output on the SB0/P02 or SB1/P03 pin starting with the MSB. Receive data applied to the SB0 (or SB1) pin is latched in the shift register 0 on the rising edge of $\overline{SCK0}$.

**Fig. 4-54 Address Transfer Operation from Master Device to Slave Device (WUP = 1)**

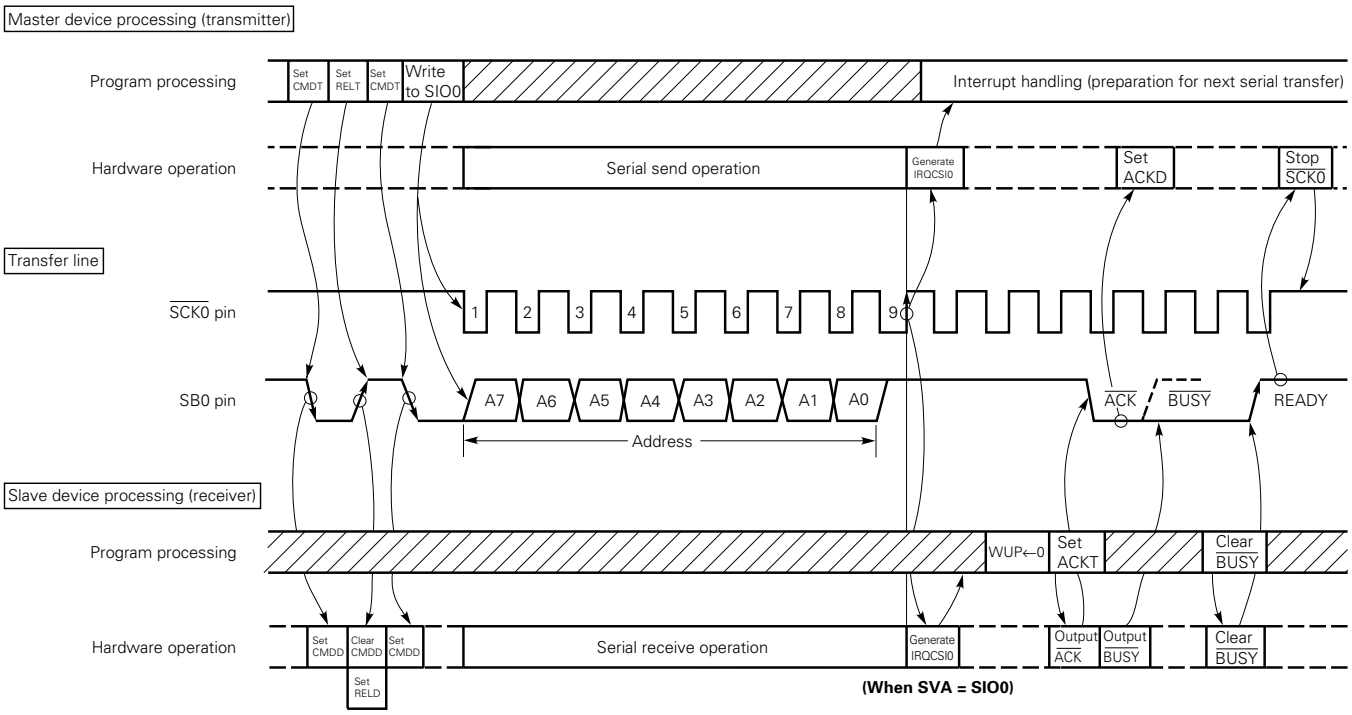


**Fig. 4-55 Command Transfer Operation from Master Device to Slave Device**

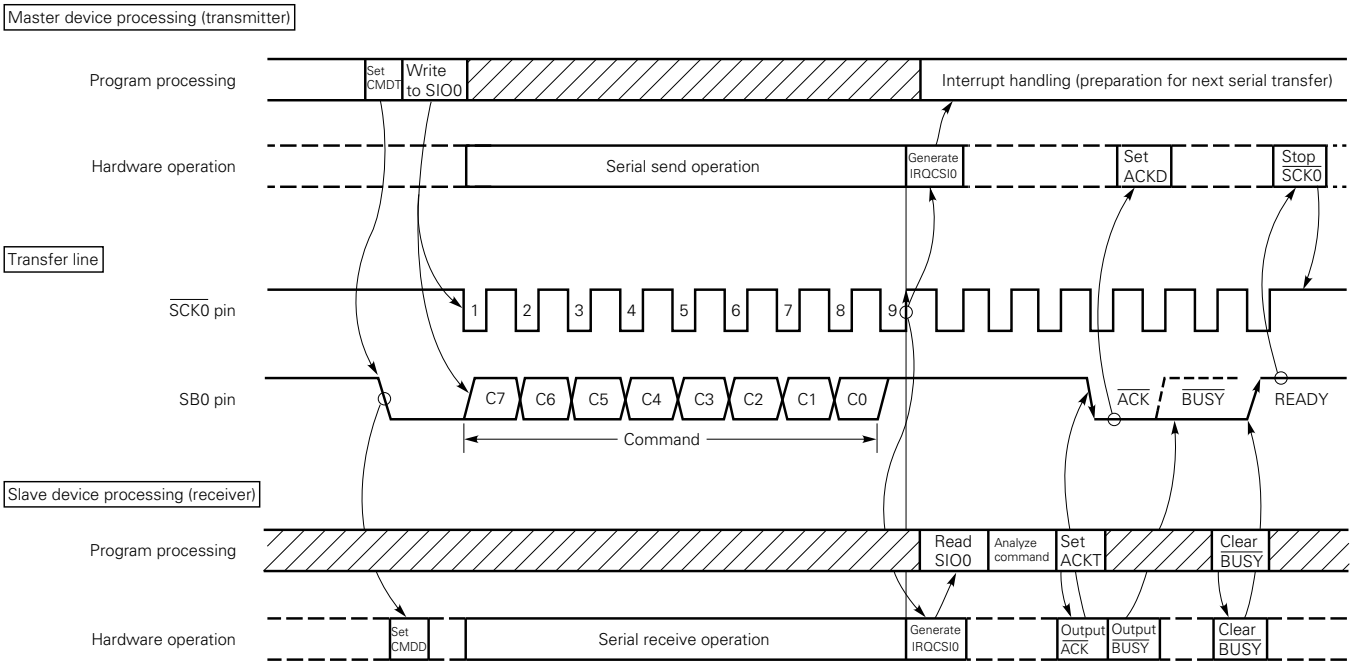


**Fig. 4-56 Data Transfer Operation from Master Device to Slave Device**

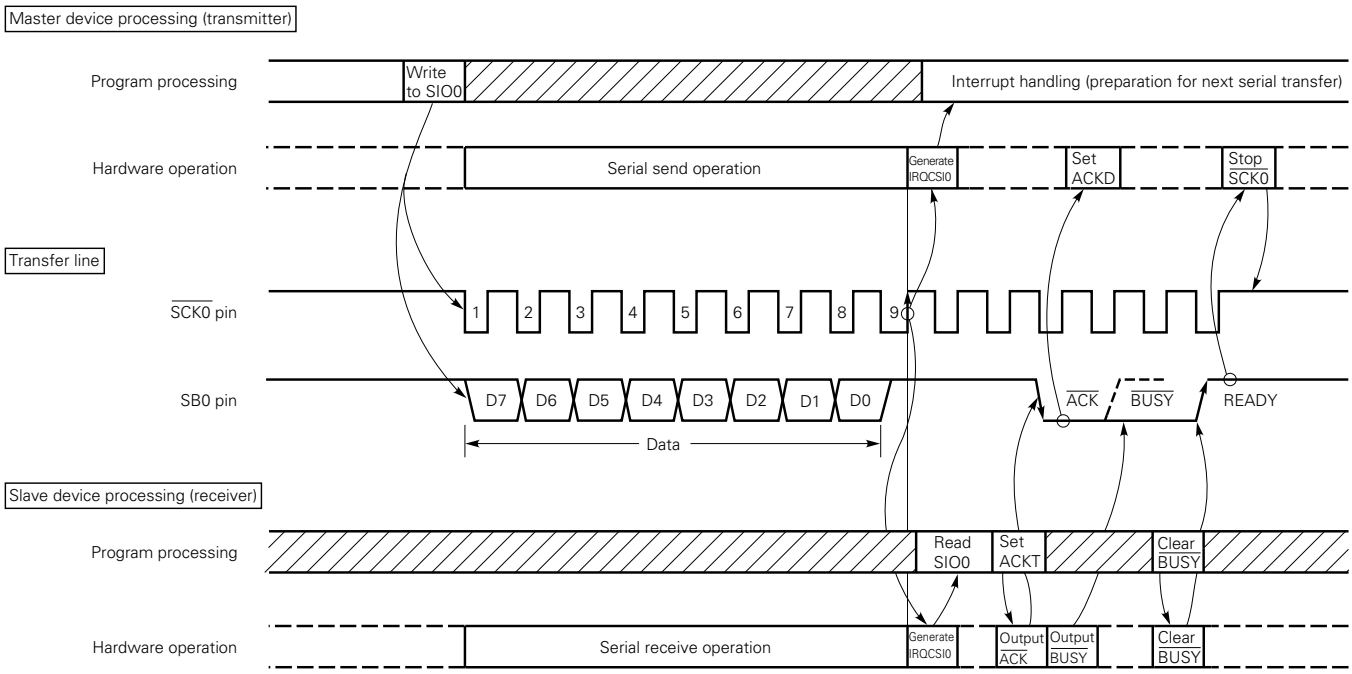
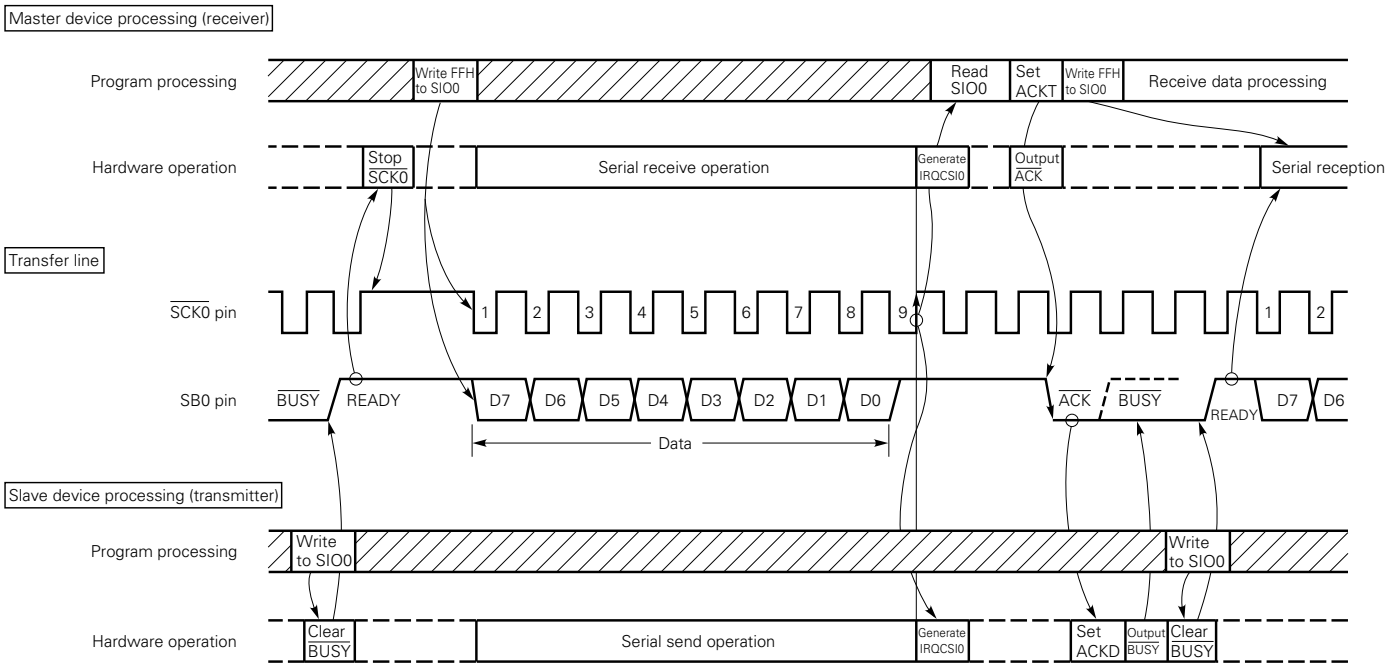


**Fig. 4-57 Data Transfer Operation from Master Device to Slave Device ★**



**(6) Transfer start in each mode**

In each of the three-wire serial I/O, two-wire serial I/O, and SBI modes, serial transfer is started by writing transfer data in shift register 0 (SIO0).

However, the following two conditions must be satisfied:

- The serial interface operation enable/disable bit (CSIE0) is set to 1.
- The internal serial clock is not operating after 8-bit serial transfer, or $\overline{SCK0}$ is high.

**Caution Transfer operation cannot be started by setting CSIE0 to 1 after writing data to the shift register 0.**

When eight bits have been transferred, serial transfer automatically terminates setting the interrupt request flag (IRQCSI0).

**[In the two-wire serial I/O mode]**

**Caution The N-ch transistor needs to be turned off when data is received. So FFH must be written to SIO0 beforehand.**

**[In the SBI mode]**

**Cautions 1. The N-ch transistor needs to be turned off when data is received. So FFH must be written to SIO0 beforehand.**
**However, when the wake-up function specification bit (WUP) is set to 1, the N-ch transistor is always off. So FFH need not be written to SIO0 beforehand for reception.**

**2. If data is written to SIO0 when the slave is busy, the data is not lost.**
**Transfer operation is started when the busy state is released and input to SB0 (or SB1) goes high.**

**Example** When RAM data specified by the HL register is transferred to SIO0, SIO0 data is loaded into the accumulator at the same time, and serial transfer is started.

```
MOV   XA, @HL     ; Extracts send data from RAM
SEL   MB15        ; Or CLR1  MBE
XCH   XA, SIO0    ; Exchanges send data with receive data and starts transfer
```

**(7) Manipulation of $\overline{SCK0}$ pin output**

The $\overline{SCK0}$/P01 pin has a built-in output latch, so that this pin allows static output by software manipulation in addition to normal serial clock output.

The number of $\overline{SCK0}$s can be software-set arbitrarily by manipulating the P01 output latch. (The SO0/SB0/SB1 pin is controlled by manipulating the RELT and CMDT bits of SBIC.)

The procedure for manipulating $\overline{SCK0}$/P01 pin output is explained below.

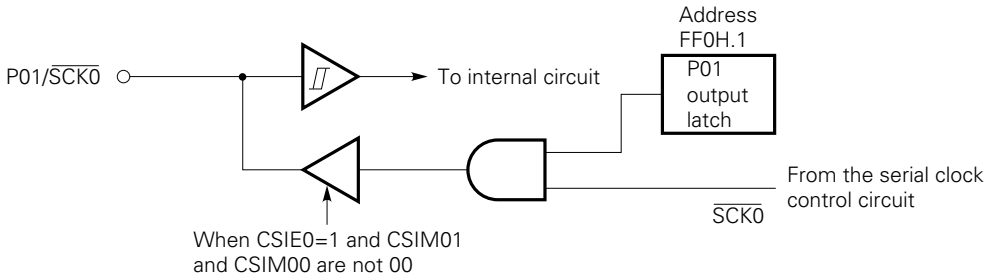
① Set serial operation mode register 0 (CSIM0) ($\overline{SCK0}$ pin: output mode, serial operation: enabled). When serial transfer operation is halted, $\overline{SCK0}$ from the serial clock control circuit is set to 1.

② Manipulate the P01 output latch by using a bit manipulation instruction.

**Example** To output one clock cycle on the $\overline{SCK0}$/P01 pin by software

```
SEL     MB15            ; or CLR1  MBE
MOV     XA, #10000011B  ; SCK0 (fx/2^3), output mode
MOV     CSIM0, XA
CLR1    0FF0H.1         ; SCK0/P01 ← 0
SET1    0FF0H.1         ; SCK0/P01 ← 1
```

**Fig. 4-58 $\overline{SCK0}$/P01 Pin Circuit Configuration**




The P01 output latch is mapped to bit 1 of address FF0H. A $\overline{RESET}$ signal sets the P01 output latch to 1.

**Cautions 1. During normal serial transfer operation, the P01 output latch must be set to 1.**

**2. The P01 output latch cannot be addressed by specifying PORT0.1 (as described below). The address of the latch (0FF0H.1) must be coded in the operand of an instruction directly. However, MBE = 0 (or MBE = 1, MBS = 15) must be specified before the instruction is executed.**

```
CLR1  PORT0.1   }
SET1  PORT0.1   }  Not allowed

CLR1  0FF0H.1   }
SET1  0FF0H.1   }  Allowed
```

**(8) Serial interface (channel 1) functions**

The serial interface (channel 1) of the μPD75238 has following two modes.
The functions of the two modes are outlined below.

- **Operation halt mode**

  This mode is used when serial transfer is not performed. This mode reduces power consumption.

- **Three-wire serial I/O mode**

  8-bit data transfer is performed using three lines: Serial clock ($\overline{SCK1}$), serial output (SO1), and serial input (SI1).

  The three-wire serial I/O mode allows full-duplex transmission, so data transfer can be performed at higher speed.

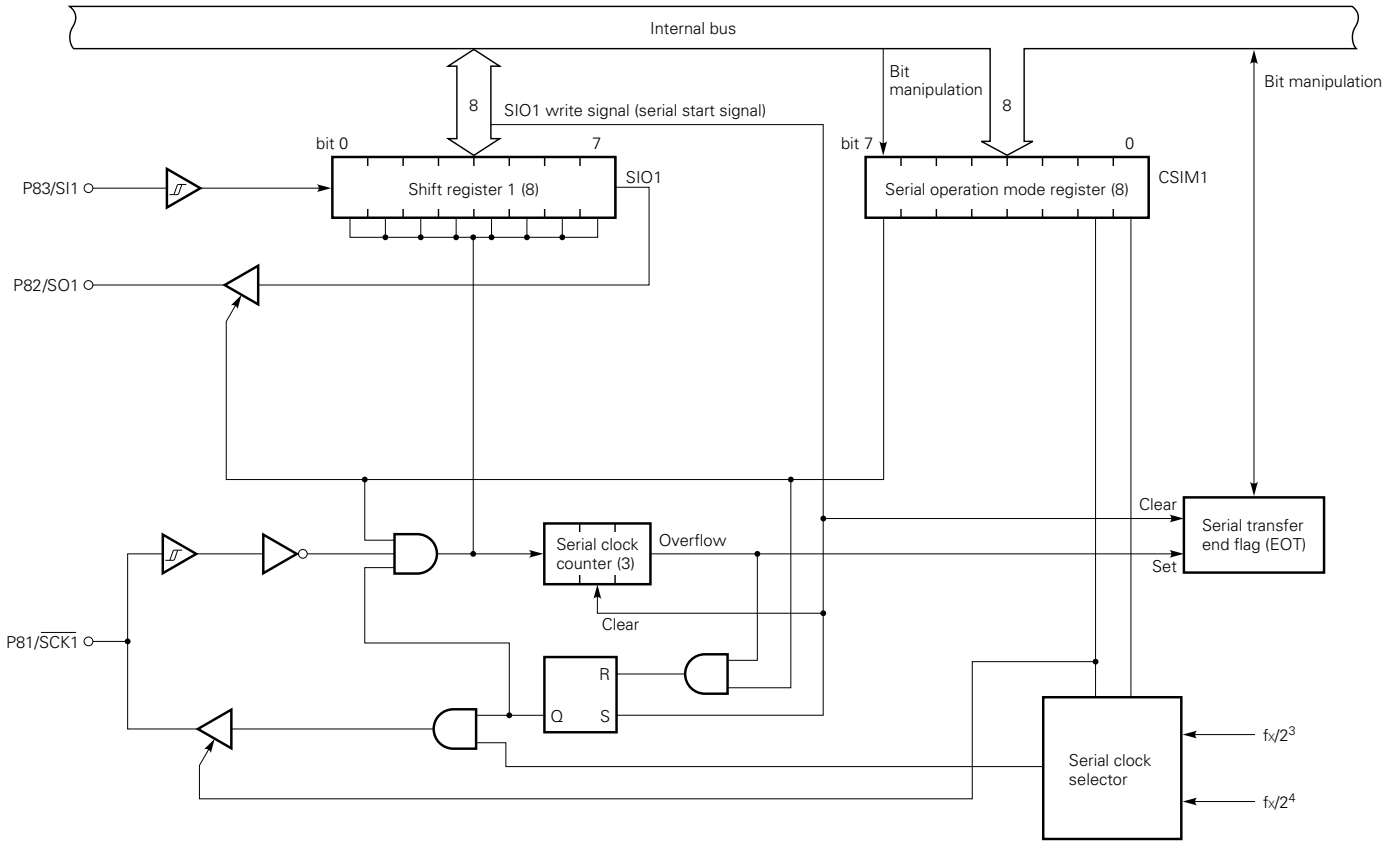
★ Eight-bit data transfer always starts the MSB.

  The three-wire serial I/O mode enables connections to be made with the 75X series, 78K series, and many other types of peripheral I/O devices.

**(9) Serial interface (channel 1) configuration**

Fig. 4-59 shows the block diagram of the serial interface (channel 1).

**Fig. 4-59 Block Diagram of the Serial Interface (Channel 1)**



### (10) Functions of serial interface (channel 1) registers

#### (a) Serial operation mode register 1 (CSIM1)

Fig. 4-60 shows the format of serial operation mode register 1 (CSIM1).

CSIM1 is an 8-bit register which specifies a serial interface (channel 1) operation mode and serial clock.

CSIM1 is manipulated using an 8-bit memory manipulation instruction. Only the high-order one bit can be manipulated independently. Each bit can be manipulated using its name.

A $\overline{\text{RESET}}$ input clears all bits to 0.

**Fig. 4-60 Format of Serial Operation Mode Register 1**

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FC8H | CSIE1 | 0 | 0 | 0 | 0 | 0 | CSIM11 | CSIM10 | CSIM1 |

**Serial clock selection bit (W)**

| CSIM11 | CSIM10 | Serial clock (3-wire serial I/O mode) | $\overline{\text{SCK1}}$ pin mode |
|---|---|---|---|
| 0 | 0 | External clock applied to $\overline{\text{SCK1}}$ pin | Input |
| 0 | 1 | Not to be set | — |
| 1 | 0 | $f_X/2^4$ (262 kHz or 375 kHz)**Note** | Output |
| 1 | 1 | $f_X/2^3$ (524 kHz or 750 kHz)**Note** | |

**Note** The values in parentheses are for fx = 4.19 MHz or 6.0 MHz.

**Serial interface operation enable/disable specification bit (W)**

| | | Shift register 1 | Serial clock counter | IRQCSI flag | SO1, SI1 pin |
|---|---|---|---|---|---|
| ★ CSIE1 | 0 | Shift operation disabled | Cleared | Held | Used only for port 8 |
| ★ | 1 | Shift operation enabled | Count operation | Can be set. | Port 8 is shared with serial function |

**Caution Be sure to write 0 in bits 2 to 6 of the serial operation mode register.**

**(b) Shift register 1 (SIO1)**

SIO1 is an 8-bit register which performs parallel-serial conversion and serial transfer (shift) operation in phase with the serial clock.

Serial transfer is started by writing data to SIO1.

In send operation, data written to SIO1 is output on the serial output (SO1). In receive operation, data is read from the serial input (SI1) into SIO1.

Data can be read from or written to SIO1 using an 8-bit manipulation instruction.

When the $\overline{\text{RESET}}$ signal is entered during operation, the value of SIO1 is undefined. When the $\overline{\text{RESET}}$ signal is entered in the standby mode, the value of SIO1 is preserved.

Shift operation is stopped after 8-bit send or receive operation is completed.

The timing for reading SIO1 and start of serial transfer (writing to SIO1) is as follows:

- When the serial interface operation enable/disable bit (CSIE1) is set to 1. However, the case where CSIE1 is set to 1 after data is written to the shift register is excluded.
- When the serial clock is masked after 8-bit serial transfer
- When $\overline{\text{SCK1}}$ is high

### (11) Serial interface (channel 1) operation

#### (a) Operation halt mode

The operation halt mode is used when serial transfer is not performed. This mode reduces power consumption.

Shift register 1 does not perform shift operation in this mode, so the shift register can be used as a normal 8-bit register.

A $\overline{\text{RESET}}$ input sets the operation halt mode.

The P82/SO1 pin and P83/SI1 pin are fixed to function for input ports. The P81/$\overline{\text{SCK1}}$ pin can be used as an input port pin by setting serial operation mode register 1.

#### (b) Three-wire serial I/O mode operations

The three-wire serial I/O mode is compatible with other modes used in the 75X series and 78K series. Communication is performed using three lines: Serial clock ($\overline{\text{SCK1}}$), serial output (SO1), and serial input (SI1).

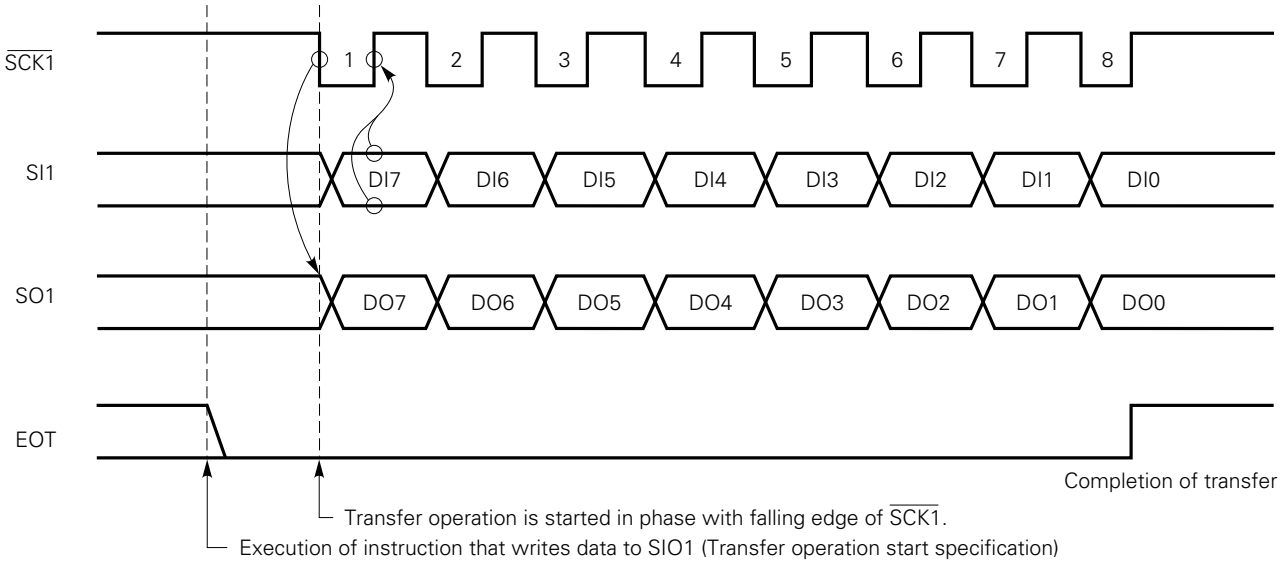
The three-wire serial I/O mode transfers data with eight bits as one block. Data is transferred bit by bit in phase with the serial clock.

Shift register 1 performs shift operation on the falling edge of the serial clock ($\overline{\text{SCK1}}$). Send data is latched on the SO1 latch, and is output on the SO1 pin.

Receive data applied to the SI1 pin is latched in the shift register 1 on the rising edge of $\overline{\text{SCK1}}$.

When eight bits have been transferred, operation of shift register 1 automatically terminates setting the serial transfer end flag (EOT).

**Fig. 4-61 Timing of the Three-Wire Serial I/O Mode**



### 4.10 A/D CONVERTER

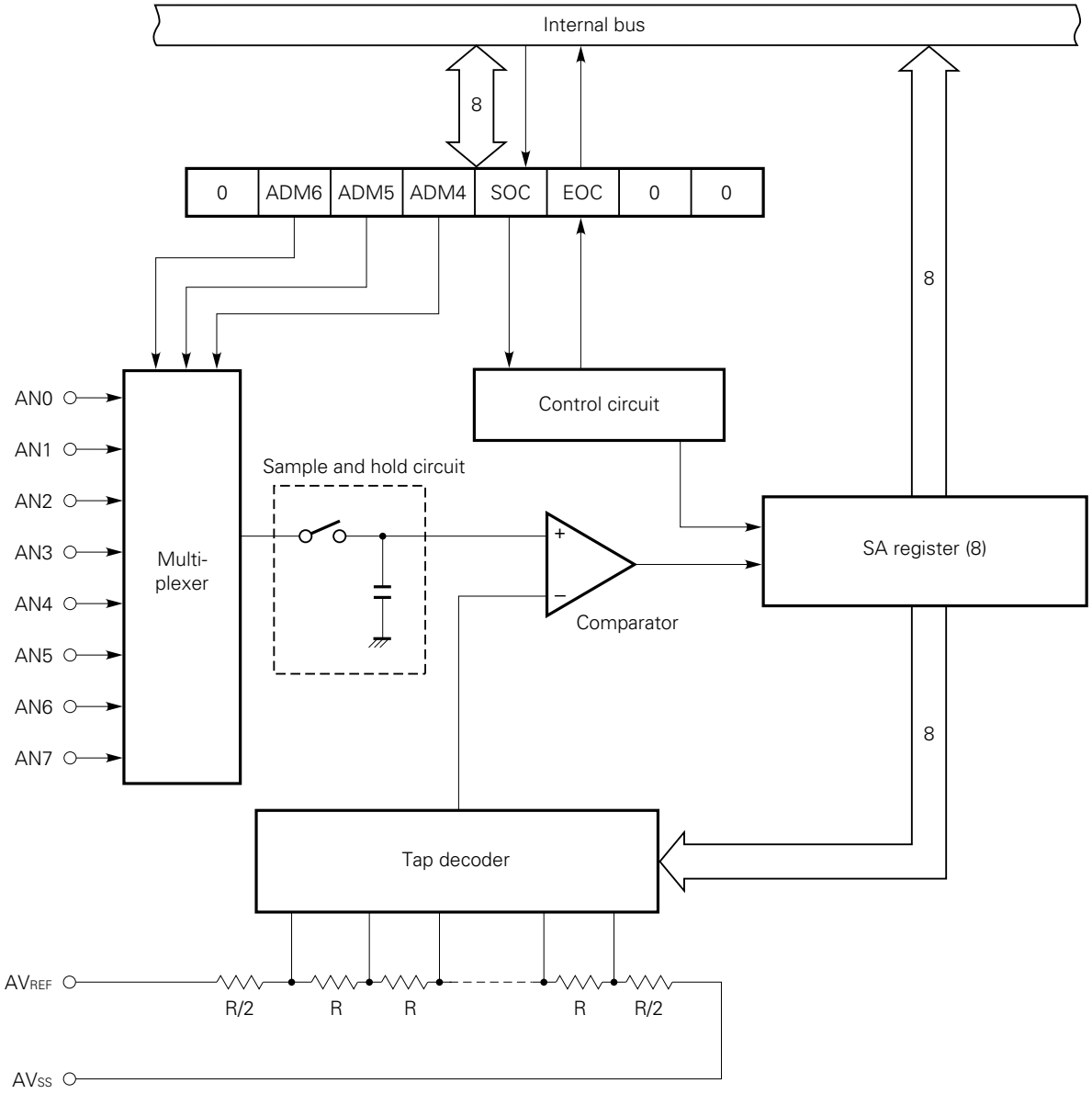
The μPD75238 contains an 8-bit analog/digital (A/D) converter that has eight analog input channels (AN0 to AN7).

The A/D converter employs the successive-approximation method.

#### (1) Configuration of the A/D converter

Fig. 4-62 shows the configuration of the A/D converter.

**Fig. 4-62 Block Diagram of the A/D Converter**



**(2) Pins of the A/D converter**

**(a) AN0 to AN7**

AN0 to AN7 are the input pins for eight analog signal channels. Analog signals subject to A/D conversion are applied to these pins.

The A/D converter contains a sample-and-hold circuit, and analog input voltages are internally maintained during A/D conversion.

**(b) $AV_{REF}$, $AV_{SS}$**

A reference voltage for the A/D converter is applied to these pins.

By using an applied voltage across $AV_{REF}$ and $AV_{SS}$, signals applied to AN0 to AN7 are converted to digital signals.

$AV_{SS}$ must be always $V_{SS}$.

★ **(c) $AV_{DD}$**

This is the supply voltage pin for the A/D converter. When the A/D converter is not used or is in the standby mode, the potential of the $AV_{DD}$ pin must be equal to that of the $V_{DD}$ pin.

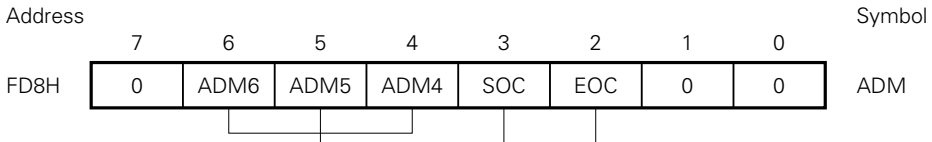
**(3) A/D conversion mode register**

The A/D conversion mode register (ADM) is an 8-bit register used to select analog input channels, direct the start of conversion, and detect the completion of conversion. (See **Fig. 4-63**.)

ADM is set with an 8-bit manipulation instruction. The conversion completion detection flag (EOC) in bit 2 and the conversion start direction bit (SOC) in bit 3 can be manipulated individually.

A $\overline{\text{RESET}}$ input initializes ADM to 04H. That is, only EOC is set to 1, with all bits cleared to 0.

**Fig. 4-63 Format of the A/D Conversion Mode Register**




**End of conversion flag**

| | | |
|---|---|---|
| EOC | 0 | Conversion under way |
| | 1 | Conversion completed |

**Start of conversion bit**

| | |
|---|---|
| SOC | Setting this bit starts conversion.<br>After conversion is started, the bit is reset automatically. |

**Analog channel selection bit**

| ADM6 | ADM5 | ADM4 | Analog channel |
|---|---|---|---|
| 0 | 0 | 0 | AN0 |
| 0 | 0 | 1 | AN1 |
| 0 | 1 | 0 | AN2 |
| 0 | 1 | 1 | AN3 |
| 1 | 0 | 0 | AN4 |
| 1 | 0 | 1 | AN5 |
| 1 | 1 | 0 | AN6 |
| 1 | 1 | 1 | AN7 |

**Caution A/D conversion is started a maximum of $2^4/f_X$ seconds (2.67 μs at 6.0 MHz)[Note] after SOC is set. (See (5) in Section 4.10.)**

**Note** 3.81 μs at 4.19 MHz

**(4) SA register (SA)**

The SA register (successive approximation register) is an 8-bit register to hold the result of A/D conversion in successive approximation.

SA is read with an 8-bit manipulation instruction. No data can be written to SA by software.

A $\overline{\text{RESET}}$ input sets SA to 7FH.

**(5) A/D converter operation**

Analog input signals subject to A/D conversion are specified by setting bits 6, 5, and 4 in the A/D conversion mode register (ADM6, ADM5, and ADM4).

A/D conversion is started by setting bit 3 (SOC) of ADM to 1. After that, SOC is automatically cleared to 0. A/D conversion is performed by hardware using the successive-approximation method. The resultant 8-bit data is loaded into the SA register. Upon completion of A/D conversion, bit 2 (EOC) of ADM is set to 1.

Fig. 4-64 shows the timing chart of A/D conversion.

The A/D converter is used as follows:

① Select analog input channels (by setting ADM6, ADM5 and ADM4).
② Direct the start of A/D conversion (by setting SOC)3
③ Wait for the completion of A4D conversion (wait for EOC to be set or wait using a software timer).
④ Read the result of A/D conversion (read the SA register).

**Cautions 1. ① and ② above can be performed at the same time.**

**2. There is a delay of up to $2^4/f_X$ seconds (2.67 μs at 6.0 MHz)[Note] from the setting of SOC to the clearing of EOC after A/D conversion is started. EOC must be tested when a time indicated in Table 4-7 has elapsed after the setting of SOC. Table 4-7 also indicates A/D conversion times.**

**Note** 3.81 μs at 4.19 MHz

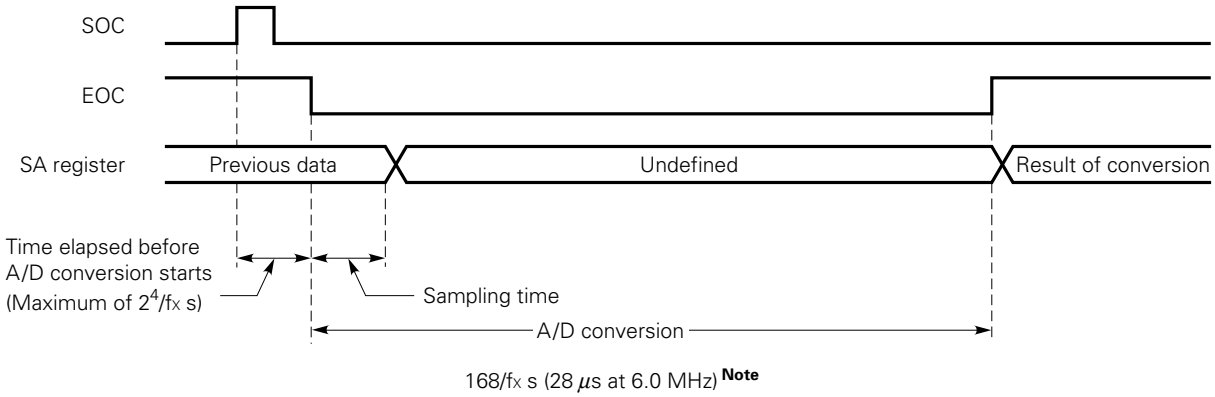
**Table 4-7 Setting of SCC and PCC**

| Setting values of SCC, PCC | | | | A/D conversion time | Wait time from SOC setting to EOC test | Wait time from SOC setting to A/D conversion completion |
|---|---|---|---|---|---|---|
| SCC3 | SCC0 | PCC1 | PCC0 | | | |
| 0 | 0 | 0 | 0 | 168/fx (28.0 μs at 6.0 MHz)[Note] | Waiting not required | 3 machine cycles |
| | | 1 | 0 | | 2 machine cycles | 21 machine cycles |
| | | 1 | 1 | | 4 machine cycles | 42 machine cycles |
| 0 | 1 | × | × | | Waiting not required | Waiting not required |
| 1 | × | × | × | Conversion stopped | — | — |

**Note** 40.1 μs at 4.19 MHz

**Remark** ×: Don't care

**Fig. 4-64 Timing Chart of A/D Conversion**




**Note** 40.1 μs at 4.19 MHz

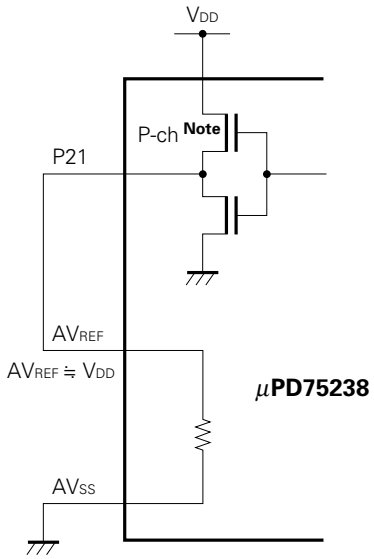
**(6) Notes on the standby mode**

The A/D converter operates with the main system clock. So its operation stops in the STOP mode, or when the subsystem clock is used, in the HALT mode. A current flows through the $AV_{REF}$ pin even when the A/D converter is stopped, so that the current must be stopped to reduce overall system power consumption. Since the P21 pin has a higher drive capability than the other ports, it can supply voltage to the $AV_{REF}$ pin directly.

In this case, however, the actual $AV_{REF}$ voltage does not provide precision. This means that the value resulting from conversion does not provide precision and can be used only for relative comparison. In the standby mode, outputting a low on the P21 can reduce power consumption.

In the standby mode, the potential of the $AV_{DD}$ pin must be equal to that of the $V_{DD}$ pin.

**Fig. 4-65 Reducing Power Consumption in the Standby Mode**




**Note** The drive capability of P-ch is higher than that of other ports.

### (7) Other notes on use

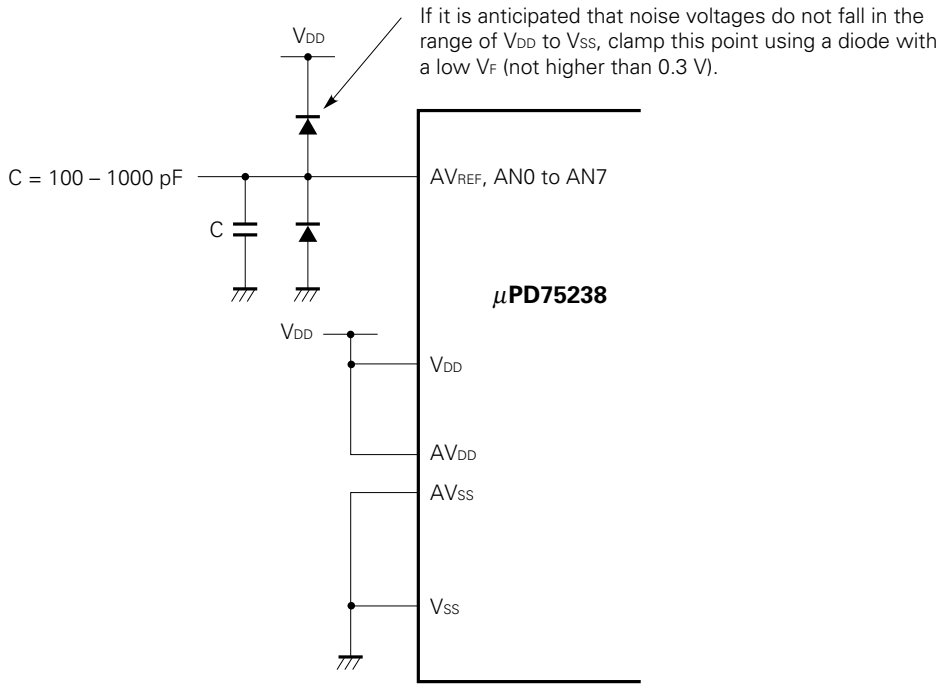
#### (a) AN0 to AN7 input range

Specified voltages must be applied to AN0 to AN7 inputs. If a voltage higher than $V_{DD}$ or lower than $V_{SS}$ is applied even when the maximum absolute rating is not exceeded, the conversion result for an associated channel becomes unpredictable. In addition, the conversion results for other channels may be affected.

★ #### (b) Noise protection

To maintain 8-bit resolution, the user should pay attention to noise that may be applied to the $AV_{REF}$, and AN0 to AN7 pins. Noise adversely affects operation to a greater extent when the analog input source has a higher output impedance. As shown in Fig. 4-66, a capacitor should be externally connected.

**Fig. 4-66 Analog Input Pin Connection**




#### (c) AN4/P90 to AN7/P93 pins

The analog input pins (AN4 to AN7) are also used for an input port (port 9).

When any of AN4 to AN7 is selected for A/D conversion, no input instruction must be executed for port 9 during A/D conversion. Otherwise, the accuracy of conversion may deteriorate.

If a digital pulse signal is applied to a pin adjacent to a pin being used for A/D conversion, an expected A/D conversion value may not be obtained because of coupling noise.

So no digital pulse signal should be applied to the adjacent pin being used for A/D conversion.

★ #### (d) $AV_{DD}$ pins

When the A/D converter is not used or is in the standby mode, the potential of the $AV_{DD}$ pin must be equal to that of the $V_{DD}$ pin.

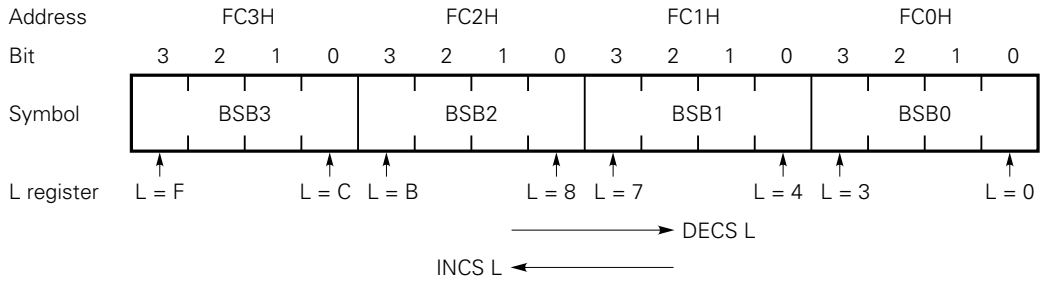
### 4.11 BIT SEQUENTIAL BUFFER: 16 BITS

The bit sequential buffer (BSB0 to BSB3) is special data memory for bit manipulations.

The buffer allows bit manipulations to be performed very easily by sequentially changing address and bit specifications. So the buffer is particularly useful in processing long data bit by bit.

This data memory consists of 16 bits, and allows pmem.@L addressing with a bit manipulation instruction and also allows indirect bit specification using the L register. In this case, only by incrementing or decrementing the L register in a program loop, the bit to be manipulated can be sequentially shifted for continued processing.

**Fig. 4-67 Format of the Bit Sequential Buffer**




**Remark** In pmem.@L addressing, bit specification is shifted according to the L register. With pmem.@L addressing, bit sequential buffer can be manipulated at any time regardless of MBE/MBS specification.

Data can also be manipulated using direct addressing. The buffer can be used for applications such as continuous 1-bit data input or output operations by combining direct 1-bit, 4-bit, and 8-bit addressing with pmem.@L addressing. In 8-bit manipulation, the higher eight bits or lower eight bits can be manipulated by specifying BSB0 or BSB2.

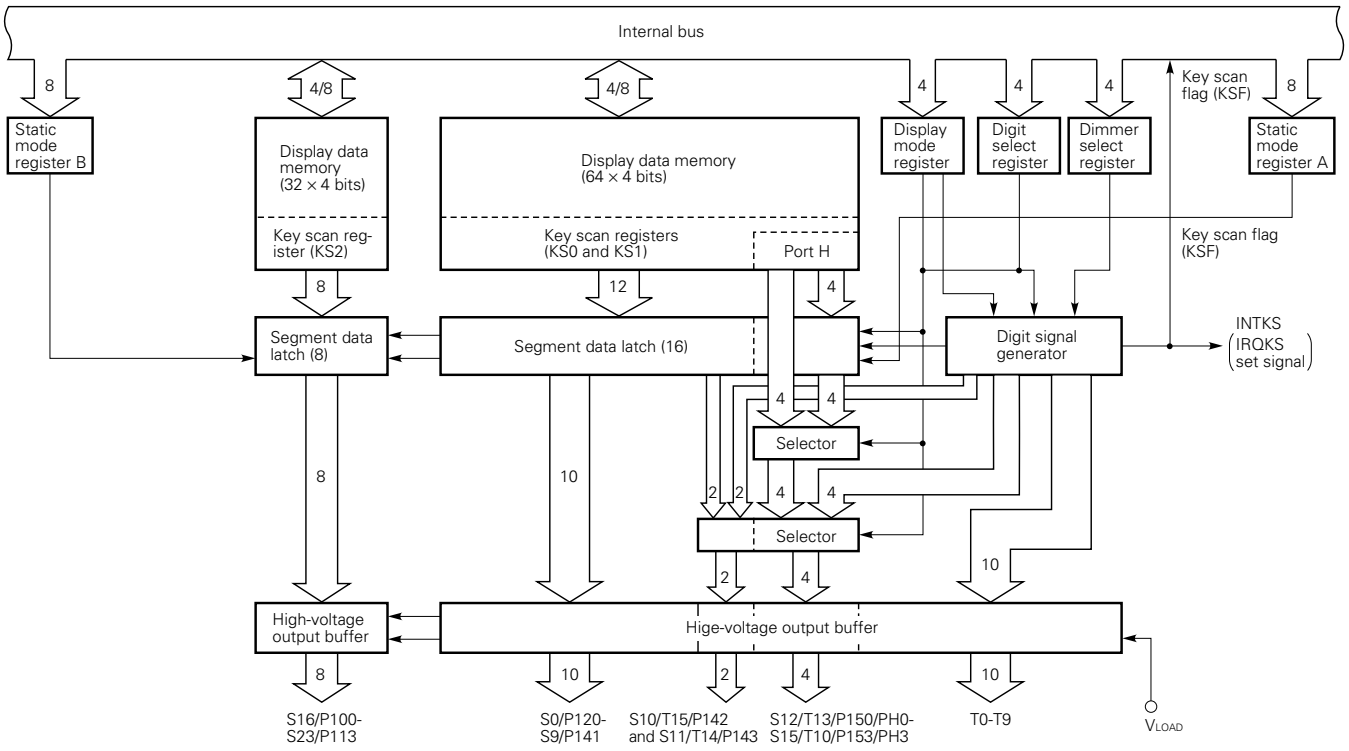
### 4.12 FIP CONTROLLER/DRIVER

**(1) Configuration of the FIP controller/driver**

The μPD75238 contains a display controller that reads the contents of display data memory by DMA operation and generates digit and segment signals automatically. It also contains a high-voltage output buffer that can directly drive a fluorescent indicator lamp (FIP). Fig. 4-68 shows the configuration of the FIP controller/driver.

**Caution The FIP controller/driver can operate only when the high-speed or medium-speed (PCC = 0011B or 0010B) is set for the main system clock (SCC.0 = 0). With the other clocks or in the standby mode, the FIP controller/driver may malfunction. Disable FIP controller operation (DSPM.3 = 0) before entering into the standby mode.**

**Fig. 4-68 Block Diagram of the FIP Controller/Driver**



**(2) FIP controller/driver functions**

The FIP controller/driver contained in the μPD75238 has the following functions:

(a) The FIP controller/driver automatically reads display data and generates a segment signal (DMA operation) and a digit signal.

(b) An FIP having 9 to 24 segments and 9 to 16 digits can be controlled with the display mode register (DSPM), digit select register (DIGS), static mode register A (STATA), and static mode register B (STATB). (Up to 34 display outputs are allowed.)

(c) Any outputs not used for dynamic display can be used as static outputs or output ports.

(d) The dimmer function provides eight levels of intensity.

(e) Hardware that makes key scan application possible
- An interrupt (IRQKS) is caused for key scanning (detection of key scan timing).
- Key scan data can be output on a segment output with key scan buffers (KS0, KS1, and KS2).

(f) A high-voltage output pin (40 V) is provided which can directly drive the FIP.
- Segment output pins (S0 to S9, S16 to S23): $V_{OD}$ = 40 V, $I_{OD}$ = 3 mA
- Digit output pins (T0 to T15): $V_{OD}$ = 40 V, $I_{OD}$ = 15 mA

(g) Mask options for display output pins
- A pull-down resistor to $V_{LOAD}$ can be incorporated bit by bit for T0 to T9 and S0 to S15.
- A pull-down resistor to $V_{LOAD}$ or $V_{SS}$ can be incorporated bit by bit for S16 to S23. $V_{LOAD}$ or $V_{SS}$ must be selected in 8-bit units.

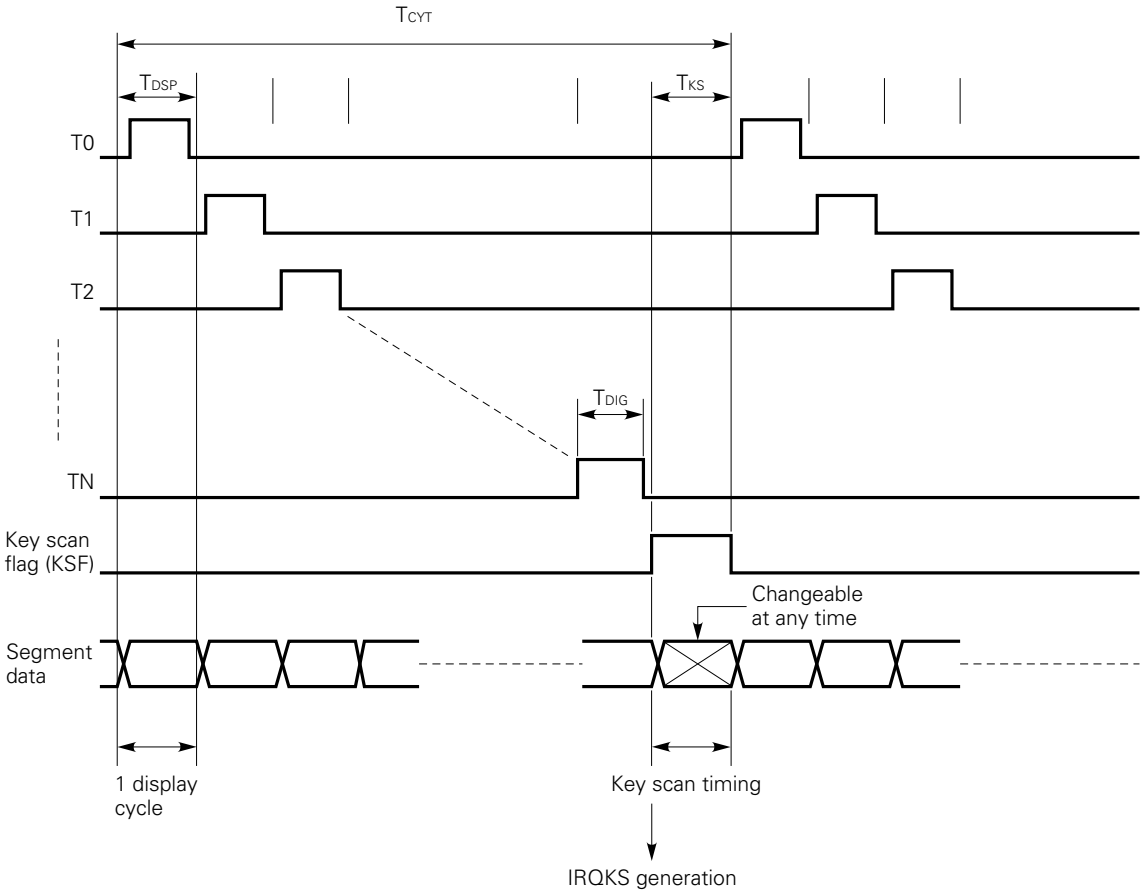
**(3) Difference of the display output function between the μPD75238, μPD75216A, and μPD75217**

Table 4-8 shows the difference of the display output function between the μPD75238, μPD75216A, and μPD75217.

**Table 4-8 Difference of the Display Output Function between the μPD75238, μPD75216A, and μPD75217**

| | μPD75238 | μPD75216A, μPD75217 |
|---|---|---|
| High-voltage output display | FIP output total : 34<br>Segment output : 9 - 24<br>Digit output : 9 - 16 | FIP output total : 26<br>Segment output : 9 - 16<br>Digit output : 9 - 16 |
| Display data area | 1A0H-1FFH | 1C0H-1FFH |
| Output dual-function pin | S0-S23 (Port 10-port 15) | S12-S15 (Port H) |
| Key scan register | KS0-KS2 | KS0, KS1 |

**Fig. 4-69 FIP Controller Operation Timing**




N : Value set in the digit select register

$T_{DSP}$: One display cycle (1024/fx: 171 μs at 6.0 MHz[Note 1] or 2048/fx: 341 μs at 6.0 MHz[Note 2])

$T_{CYT}$: Display cycle ($T_{CYT} = T_{DSP} \times (N + 2)$)

$T_{DIG}$: Digit signal pulse width, which can be selected from eight levels with the dimmer select register

**Notes 1.** 244 μs at 4.19 MHz

**2.** 489 μs at 4.19 MHz

**(4) Display mode register (DSPM)**

The display mode register (DSPM) is a 4-bit register for enable/disable setting for the display operation and for specifying the number of display segments. Fig. 4-70 shows the format of the register.

The display mode register is set with a 4-bit memory manipulation instruction.

Before a standby mode (STOP mode or HALT mode) can be set or the subsystem clock ($f_{XT}$) can be used for operation, display must be disabled by setting DSPM.3 to 0.

A $\overline{\text{RESET}}$ input clears all bits to 0.

**Fig. 4-70 Display Mode Register Format**

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| F88H | DSPM3 | DSPM2 | DSPM1 | DSPM0 | DSPM |

**Bit for specifying the number of display segments**

| DSPM2 | DSPM1 | DSPM0 | Number of display segments |
|---|---|---|---|
| 0 | 0 | 0 | 9 segments (+ 8 segments) |
| 0 | 0 | 1 | 10 segments (+ 8 segments) |
| 0 | 1 | 0 | 11 segments (+ 8 segments) |
| 0 | 1 | 1 | 12 segments (+ 8 segments) |
| 1 | 0 | 0 | 13 segments (+ 8 segments) |
| 1 | 0 | 1 | 14 segments (+ 8 segments) |
| 1 | 1 | 0 | 15 segments (+ 8 segments) |
| 1 | 1 | 1 | 16 segments (+ 8 segments) |

**Remark** Segments in parentheses are added when pins S16 to S23 are specified as dynamic mode in STATB.

**Display operation enable/disable bit**

| | | |
|---|---|---|
| DSPM3 | 0 | Display disabled |
| | 1 | Display enabled |

**(5) Digit select register (DIGS)**

The digit select register (DIGS) is a 4-bit register that specifies the number of display digits. Fig. 4-71 shows the format of the register.

DIGS is set with a 4-bit memory manipulation instruction. This register enables the number of display digits to be selected from 9 to 16 digits. No other values can be selected.

A $\overline{\text{RESET}}$ input initializes the register to 1000B so that 9-digit display is selected.

**Fig. 4-71 Digit Select Register Format**

| Address | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|
| F8AH | DIGS3 | DIGS2 | DIGS1 | DIGS0 | DIGS |

**Caution Do not set a value from 0 to 7 for N.**

| DIGS0-3 set value | Number of display digits |
|---|---|
| N (= 8 to 15) | N + 1 |

**(6) Dimmer select register (DIMS)**

The dimmer select register (DIMS) is a 4-bit register for specifying a digit signal cut width to prevent display light leakage and to use the dimmer function (for intensity control). In addition, the register is used to select a display cycle ($T_{DSP}$).

Fig. 4-72 shows the format of DIMS.

DIMS is set with a 4-bit memory manipulation instruction.

A display cycle of 341 μs at 6.0 MHz[Note 1] is usually selected by setting 1 in DIMS.0 for reduced light leakage. However, as the number of display digits increases, the display cycle becomes closer to the frequency of commercial power, causing the display to fricker. In this case, a display cycle of 171 μs at 6.0 MHz[Note 2] should be selected. If light leakage occurs in this case, adjust cutting width of the digit single by setting DIMS.1, DIMS.2, and DIMS.3.

A $\overline{\text{RESET}}$ input clears all bits to 0.

**Notes 1.** 489 μs at 4.19 MHz

**2.** 244 μs at 4.19 MHz

**Fig. 4-72 Dimmer Select Register Format**

| Address | | | | | Symbol |
|---|---|---|---|---|---|
| F89H | DIMS3 | DIMS2 | DIMS1 | DIMS0 | DIMS |

**Display cycle specification bit**

| | | |
|---|---|---|
| DIMS0 | 0 | A display cycle is $\frac{1024}{f_X}$. (One cycle = 171μs/6.0 MHz) **Note 1** |
| | 1 | A display cycle is $\frac{2048}{f_X}$. (One cycle = 341μs/6.0 MHz) **Note 2** |

**Digit signal cut width specification bit**

| DIMS3 | DIMS2 | DIMS1 | Digit signal cut width |
|---|---|---|---|
| 0 | 0 | 0 | 1/16 |
| 0 | 0 | 1 | 2/16 |
| 0 | 1 | 0 | 4/16 |
| 0 | 1 | 1 | 6/16 |
| 1 | 0 | 0 | 8/16 |
| 1 | 0 | 1 | 10/16 |
| 1 | 1 | 0 | 12/16 |
| 1 | 1 | 1 | 14/16 |

**Notes 1.** 244 μs at 4.19 MHz

**2.** 489 μs at 4.19 MHz

### (7) Static mode registers

A static mode registers are registers used to specify static output/dynamic output of the segment output pins.

There are two static mode registers: static mode register A and static mode register B. Fig. 4-73 and Fig. 4-74 show their formats.

These registers are set with an 8-bit manipulation instruction. A $\overline{\text{RESET}}$ input clears all bits to 0.

#### (a) Static mode register A (STATA)

Static mode register A (STATA) is a register to specify static output/dynamic output of the S0/P120 to S15/P153/T10/PH3 pins.

**Fig. 4-73 Static Mode Register A (STATA)**

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FD6H | 0 | 0 | 0 | 0 | STATA3 | STATA2 | STATA1 | STATA0 | STATA |

**S0 to S15 pins Static output/dynamic output selection bit**

| STATA3 | STATA2 | STATA1 | STATA0 | Output status of S0 to S15 pins |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | S0 to S15 become dynamic outputs. The number of segments and the number of digits are set by DSPM and DIGS. |
| 1 | 1 | 1 | 1 | S0 to S15 become static outputs. Static data is output by issuing an output instruction for ports 12 to 15. The outputs are independent of the value of DSPM.3. |

**Caution Part of the S0 to S15 pins cannot be set as dynamic outputs while the other pins are set as static outputs.**

**(b) Static mode register B (STATB)**

Static mode register B (STATB) is a register to specify static output/dynamic output of the S16/P100 to S23/P113 pins.

**Fig. 4-74 Static Mode Register B (STATB)**

| Address | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Symbol |
|---|---|---|---|---|---|---|---|---|---|
| FD4H | 0 | 0 | STATB5 | STATB4 | 0 | 0 | 0 | 0 | STATB |

**S16 to S23 pins Static output/dynamic output selection bit**

| STATB5 | STATB4 | Output status of S16 to S23 pins |
|---|---|---|
| 0 | 0 | S16 to S23 become dynamic outputs. Output depends on the contents of 1A0H to 1BDH. |
| 1 | 1 | S16 to S23 become static outputs. Static data is output by issuing an output instruction for ports 10 and 11. The outputs are independent of the value of DSPM.3. |

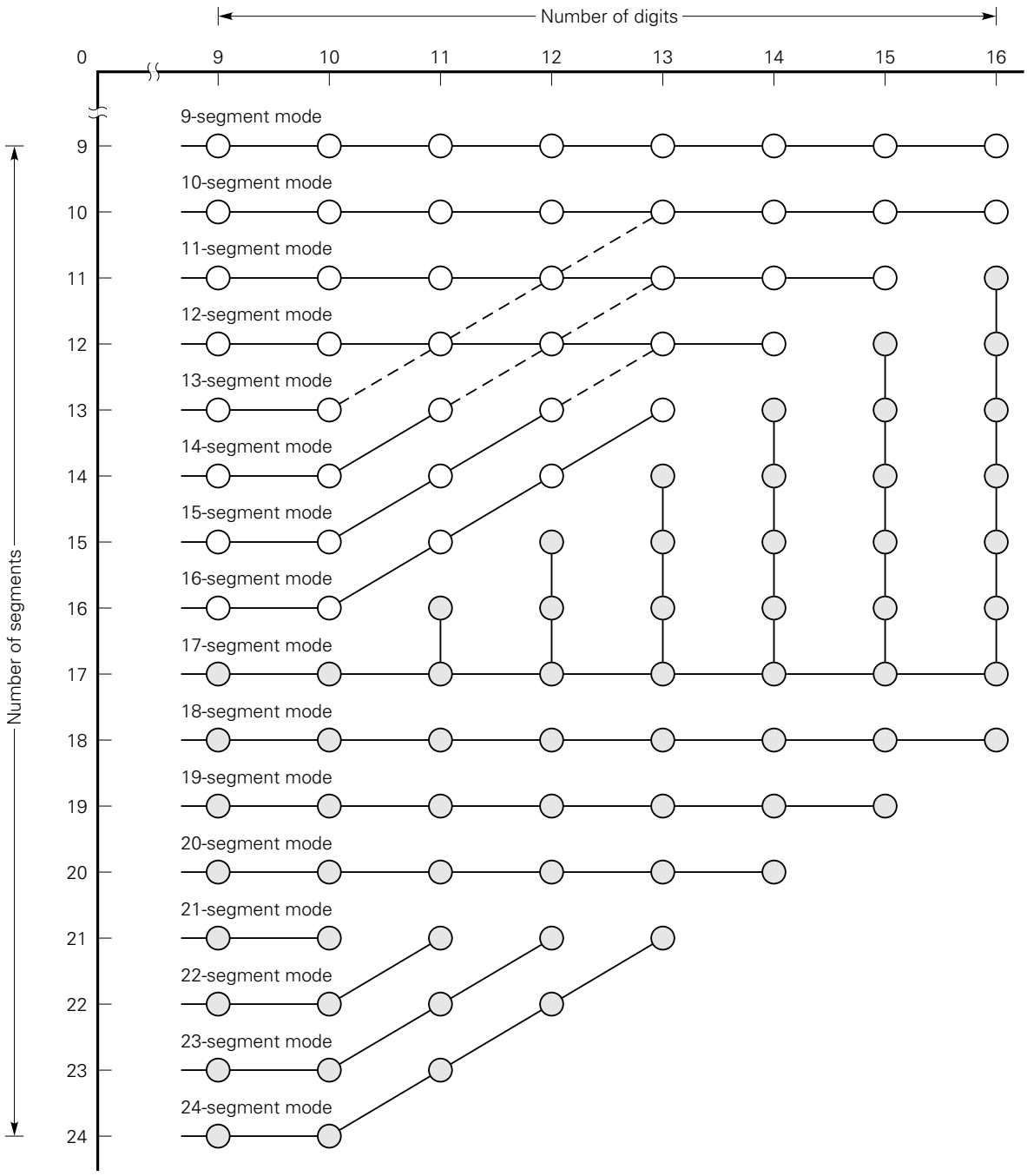
**Caution Part of the S16 to S23 pins cannot be set as dynamic outputs while the other pins are set as static outputs.**

**(8) Selection of display mode**

The number of segments and number of digits which can be displayed by the internal FIP controller/driver are determined by display modes.

Fig. 4-75 shows the selection of display mode.

**Fig. 4-75 Selection of Display Mode**




**Remark** The shaded circles indicate extended modes derived from the μPD75216A and μPD75217.

**(9) Display data memory**

Display data memory stores segment data for display. It is mapped to addresses 1A0H to 1FFH in data memory. The display controller automatically reads display data (DMA operation). Area not used for display can be used as normal data memory.

Display data is manipulated with data memory manipulation instructions in 1-, 4-, and 8-bit units. With 8-bit manipulation instructions, only even-numbered addresses can be specified.

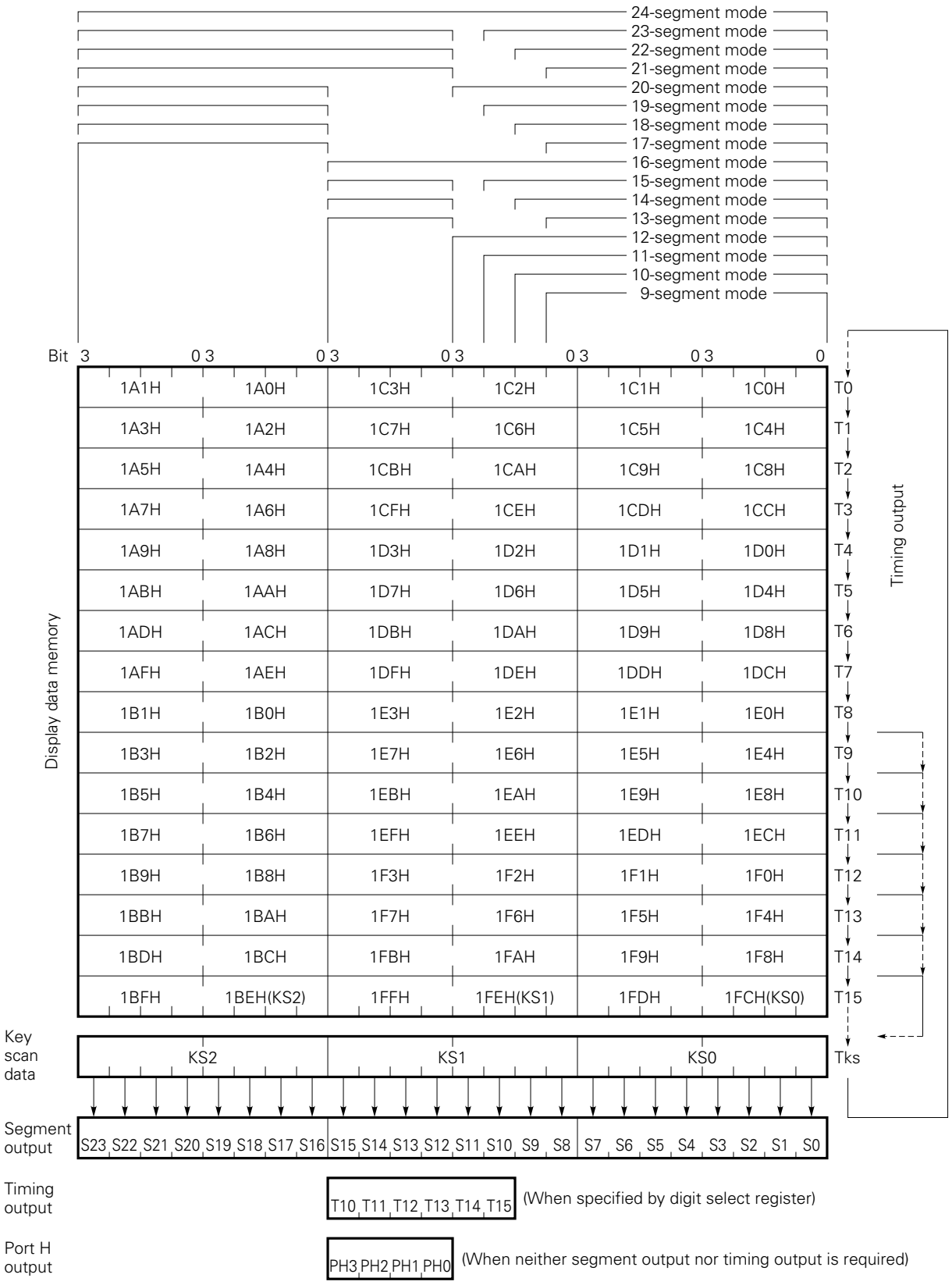
Display data memory locations at 1FCH to 1FFH, 1BEH, and 1BFH are also used as key scan registers KS0, KS1, and KS2.

**Table 4-9 Data Memory Locations Also Used as Key Scan Registers**

| Key scan register | Data memory locations |
|---|---|
| KS0 | 1FCH, 1FDH |
| KS1 | 1FEH, 1FFH |
| KS2 | 1BEH, 1BFH |

**Caution Special care must be paid when a program developed for the μPD75238 is transported to the μPD75216A or the μPD75217. This is because the μPD75216A and the μPD75217 allow up to 16 display segments, and do not contain data memory at the addresses (1A0H + 4n, 1A1H + 4n).**

**Fig. 4-76 Correspondence between Display Data Memory and Segment Output**



**(10) Key scan registers (KS0, KS1, and KS2)**

The key scan registers (KS0, KS1, and KS2) set segment output data on the key scan timing that is mapped to display data memory locations (1FCH, 1FDH, 1FEH, 1FFH, 1BEH, and 1BFH).

KS0, KS1, and KS2 are 8-bit registers, which are usually manipulated with an 8-bit manipulation instruction. (One-bit or 4-bit manipulation is also possible for the lower 4 bits.)

The data set in KS0, KS1, and KS2 is output on the segment output pins on the key scan timing. During the key scan timing, the rewriting of KS0, KS1, and KS2 is immediately reflected in segment output data, which can be used for key scanning.

**(11) Key scan flag (KSF)**

The key scan flag (KSF) is set to 1 on the key scan timing, and is automatically reset to 0 on any other timing. KSF is mapped to bit 3 at address F8AH, and can be tested on a single-bit basis. KSF cannot be written to. Testing this flag can determine whether the key scan timing is present, so that the validity of key input data can be determined.

## 5. INTERRUPT FUNCTION

The μPD75238 has eight interrupt sources and can handle multiple interrupts with a priority.

The μPD75238 is also provided with two test sources. One test source, INT2, is set by edge detection testable input.

**Table 5-1 Interrupt Sources**

| Interrupt source | | In/out | Priority[Note 1] | Vector interrupt request signal (vector table address) |
|---|---|---|---|---|
| INTBT | (Reference time interval signal from basic interval timer) | In | 1 | VRQ1 (0002H) |
| INT4 | (Detection of rising or falling edge) | Out | | |
| INT0 | (Rising/falling edge detection specification) | Out | 2 | VRQ2 (0004H) |
| INT1 | | Out | 3 | VRQ3 (0006H) |
| INTCSI0 | (Serial data transfer completion signal) | In | 4 | VRQ4 (0008H) |
| INTT0 | (Match signal from timer enter/counter 0) | In | 5 | VRQ5 (000AH) |
| INTTPG | (Match signal from timer/pulse generator) | In | 6 | VRQ6 (000CH) |
| INTKS | (Key scan timing signal from display controller) | In | 7 | VRQ7 (000EH) |
| INT2[Note 2] | (Detection of rising edge ) | Out | Testable input signal (Sets IRQ2 and IRQW.) | |
| INTW[Note 2] | (Signal from clock timer) | In | | |

**Notes 1.** The priority is used when two or more interrupt requests are issued at a time.

**2.** INT2 and INTW are test sources. Like interrupt sources, these test sources are controlled by an interrupt enable flag. But, they do not cause vector interrupts.

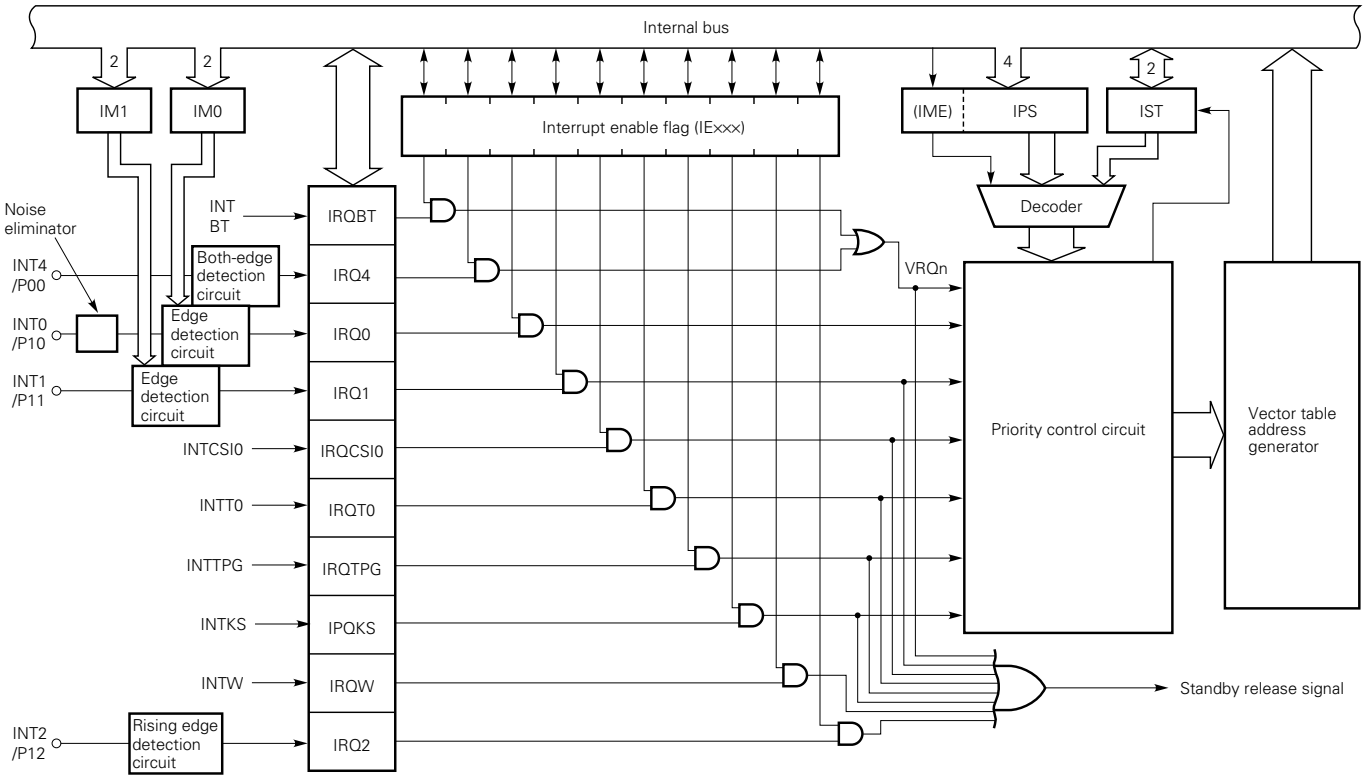
The following functions are provided for the interrupt control circuit of the μPD75238.

(a) Vector interrupt function under hardware control which can determine whether to accept an interrupt by an interrupt enable flag (IE×××) and the interrupt master enable flag (IME)
(b) Any interrupt start address can be set.
(c) Multiple interrupt function which can specify the priority by the interrupt priority specification register (IPS)
(d) Test function of an interrupt request flag (IRQ×××)
(The software can confirm that an interrupt occurred.)
(e) Release of the standby mode (Interrupts released by an interrupt enable flag can be selected.)

### 5.1 CONFIGURATION OF THE INTERRUPT CONTROL CIRCUIT

The interrupt control circuit of the μPD75238 is configured as shown in Fig. 5-1. Each hardware item is mapped in the data memory space.

**Fig. 5-1 Block Diagram of Interrupt Control Circuit**



## 5.2 HARDWARE OF THE INTERRUPT CONTROL CIRCUIT

### (1) Interrupt request flag and interrupt enable flag

There are ten interrupt request flags (IRQ×××), listed below, each corresponding to a particular interrupt or test source; there are 8 interrupt and 2 test sources. ★

| | |
|---|---|
| INT0 interrupt request flag (IRQ0) | Serial interface interrupt request flag (IRQCSI0) |
| INT1 interrupt request flag (IRQ1) | Timer/event counter interrupt request flag (IRQT0) |
| INT2 interrupt request flag (IRQ2) | Timer/pulse generator interrupt request flag (IRQTPG) |
| INT4 interrupt request flag (IRQ4) | Key scan interrupt request flag (IRQKS) |
| BT interrupt request flag (IRQBT) | Clock timer interrupt request flag (IRQW) |

The interrupt request flag is set to 1 when an interrupt request is issued, and is automatically cleared to 0 when the CPU is interrupted. Since the IRQBT and IRQ4 share the vector address, the clear operation varies. (See **Section 5.5**.)

There are ten interrupt enable flags (IE×××), listed below, each corresponding to a particular interrupt request flag. ★

| | |
|---|---|
| INT0 interrupt enable flag (IE0) | Serial interface interrupt enable flag (IECSI0) |
| INT1 interrupt enable flag (IE1) | Timer/event counter interrupt enable flag (IET0) |
| INT2 interrupt enable flag (IE2) | Timer/pulse generator interrupt enable flag (IETPG) |
| INT4 interrupt enable flag (IE4) | Key scan interrupt enable flag (IEKS) |
| BT interrupt enable flag (IEBT) | Clock timer interrupt enable flag (IEW) |

When an interrupt request flag is set, the interrupt enable flag corresponding to that interrupt request flag enables the request interrupt. When an interrupt request flag is cleared, the interrupt enable flag corresponding to that interrupt request flag disables the interrupt.

When an interrupt request flag is set and its corresponding interrupt enable flag enables the requested interrupt, a vector interrupt request (VRQn) is issued. This signal is also used for releasing the standby mode.

The interrupt request flags and interrupt enable flags are manipulated with bit manipulating instructions and 4-bit memory manipulation instructions. When a bit manipulation instruction is used, the flags can always be manipulated directly irrespective of the MBE setting. The interrupt enable flags are manipulated with EI IE××× and DI IE××× instructions. An SKTCLR instruction is normally used to test an interrupt request flag.

When an interrupt request flag is set with an instruction, a vector interrupt is executed irrespective of whether an interrupt occurs.

A $\overline{\text{RESET}}$ input clears an interrupt request flag and its corresponding interrupt enable flag to 0 and all interrupts are disabled.

**Table 5-2 Set Signals of Interrupt Request Flags**

| Interrupt request flag | Set signal of interrupt request flag | Interrupt enable flag |
|---|---|---|
| IRQBT | Set by a reference time interval signal from the basic interval timer. | IEBT |
| IRQ4 | Set by a detected rising or falling edge of an INT4/P00 pin input signal. | IE4 |
| IRQ0 | Set by a detected edge of an INT0/P10 pin input signal. The detection edge is specified by the INT0 mode register (IM0). | IE0 |
| IRQ1 | Set by a detected edge of an INT1/P11 pin input signal. The detection edge is specified by the INT1 mode register (IM1). | IE1 |
| IRQCSI0 | Set by a serial data transfer completion signal for the serial interface. | IECSI0 |
| IRQT0 | Set by a match signal from timer/event counter 0. | IET0 |
| IRQTPG | Set by a match signal from the timer/pulse generator. | IETPG |
| IRQKS | Set by a key scan timing signal from the display controller. | IEKS |
| IRQW | Set by a signal from the clock timer. | IEW |
| IRQ2 | Set by a detected rising edge of an INT2/P12 pin input signal. | IE2 |

★ **(2) Noise eliminator and edge detection mode register**

As shown in Fig. 5-2 and Fig. 5-3, the INT0, INT1, and INT2 pins are configured as external interrupt input pins that enable detection edge selection.

In addition, INT0 is provided with a noise elimination function based on a sampling clock. Basically, the noise eliminator eliminates pulses narrower than two sampling clock cycles[Note] as noise. However, it may accept pulses wider than one sampling clock cycle as interrupt signals depending on the sampling timing. It surely accepts pulses wider than two sampling clock cycles as interrupt signals.

INT0 has two sampling clocks Φ and $f_X/64$, either of which can be selected according to bit 3 (IM03) of the edge detection mode register (see **Fig. 5-4**).

The IRQ2 is set by detecting a rising edge of the INT2 pin input.

The edge detection mode registers (IM0 and IM1) used to select a detection edge have the format shown in Fig. 5-4. A 4-bit memory manipulation instruction is used to set IM0 or IM1. A $\overline{\text{RESET}}$ input clears all bits to 0, and a rising edge is selected for INT0, INT1, and INT2.

**Note** When a sampling clock is Φ, two sampling clock cycles are $2t_{CY}$.
When a sampling clock is $f_X/64$, two sampling clock cycles are $128/f_X$.

**Cautions 1. Since the INT0 pin input is sampled with a clock, INT0 does not operate in a standby mode.**
**2. When INT0/P10 is used as a port, pulses input from INT0/P10 go through the noise eliminator. So the input pulses must be wider than two sampling clock cycles.**

**Fig. 5-2 Configuration of INT0 and INT1**

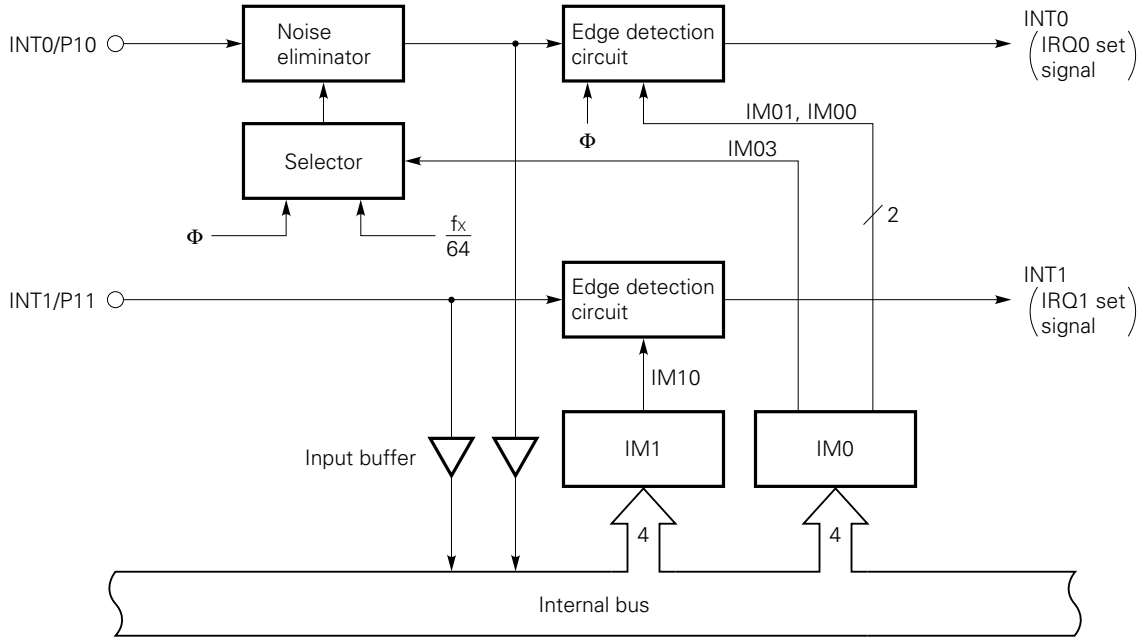



**Fig. 5-3 Configuration of INT2**

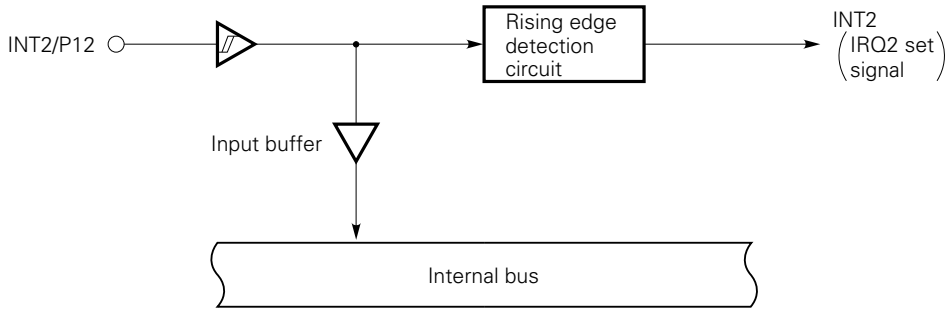
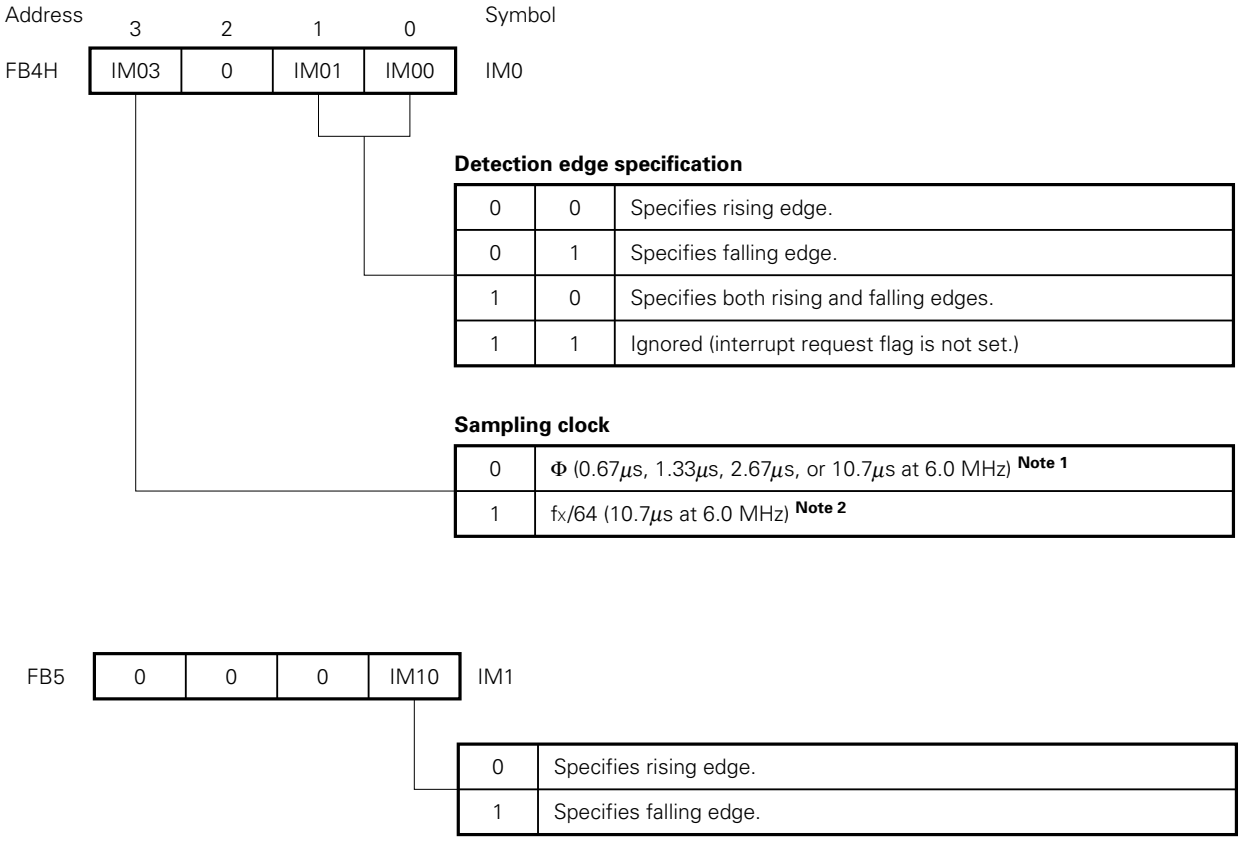


**Fig. 5-4 Format of Edge Detection Mode Registers**




**Notes 1.** 0.95 μs, 1.91 μs, 3.82 μs, or 15.3 μs at 4.19 MHz

**2.** 15.3 μs at 4.19 MHz

**Caution Since changing the edge detection mode register may set an interrupt request flag, disable the interrupts before changing the edge detection mode register. Then clear the interrupt request flag with a CLR1 instruction and enable the interrupts. When fx/64 is selected as a sampling clock pulse in changing IM0, wait for 16 machine cycles after changing the mode register and clear the interrupt request flag.**

### (3) Interrupt priority specification register (IPS)

The interrupt priority specification register specifies an interrupt with a higher priority from multiple interrupts using the low-order three bits.

Bit 3, interrupt master enable flag (IME), specifies whether to disable all interrupts.

The IPS is set with a 4-bit memory manipulation instruction. Bit 3 is set with an EI instruction and reset with a DI instruction.

When changing the low-order three bits of the IPS, interrupts must be disabled (IME = 0) beforehand.

A $\overline{\text{RESET}}$ input clears all bits to 0.

**Fig. 5-5 Interrupt Priority Specification Register**

| Address | | | | | Symbol |
|---|---|---|---|---|---|
| | 3 | 2 | 1 | 0 | |
| FB2H | IPS3 | IPS2 | IPS1 | IPS0 | IPS |

**High-order interrupt selection**

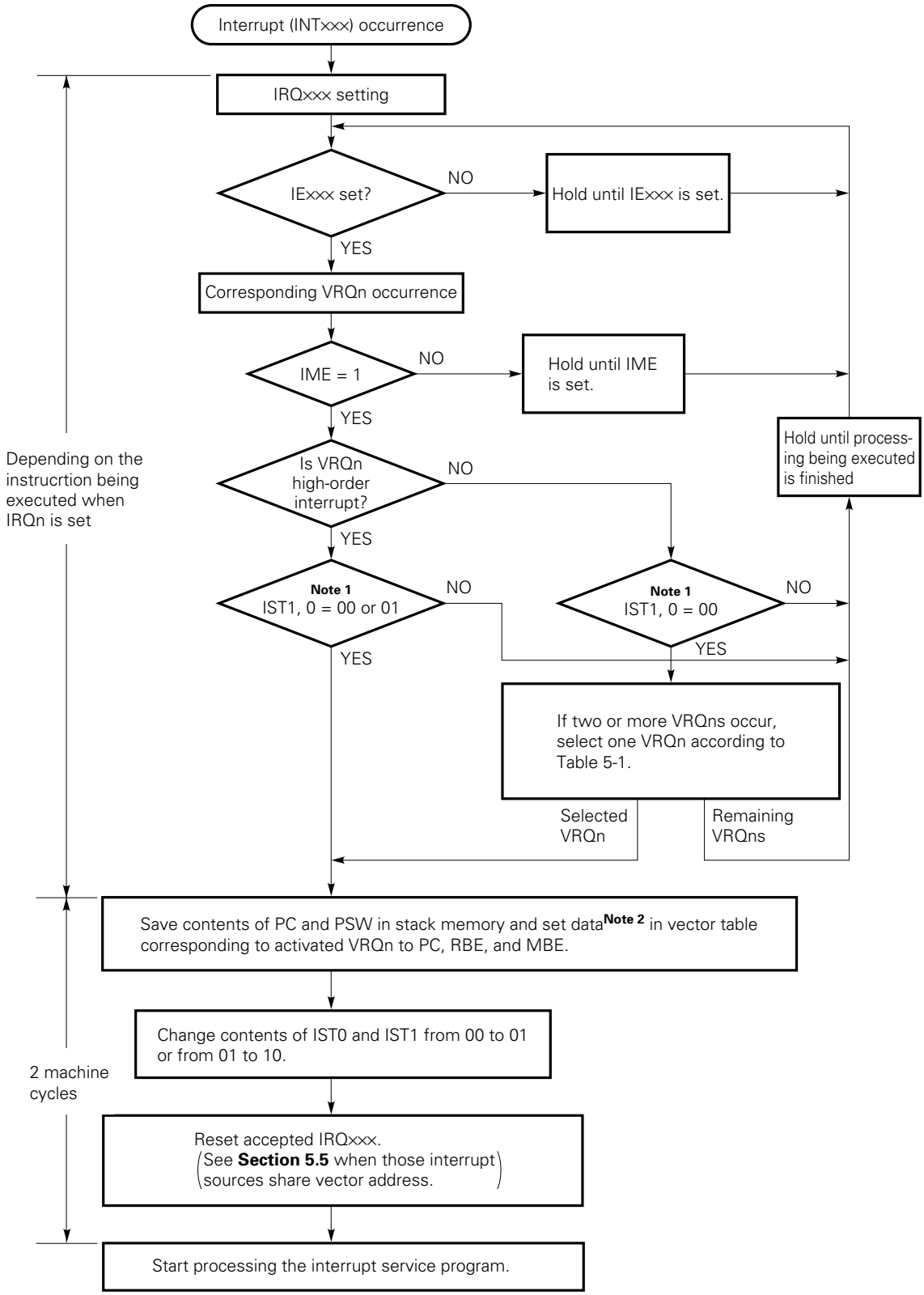
| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | All low-order interrupt | |
| 0 | 0 | 1 | VRQ1 (INTBT/INT4) | The listed vector interrupts are treated as high-order interrupts. |
| 0 | 1 | 0 | VRQ2 (INT0) | |
| 0 | 1 | 1 | VRQ3 (INT1) | |
| 1 | 0 | 0 | VRQ4 (INTCSI0) | |
| 1 | 0 | 1 | VRQ5 (INTT0) | |
| 1 | 1 | 0 | VRQ6 (INTTPG) | |
| 1 | 1 | 1 | VRQ7 (INTKS) | |

**Interrupt master enable flag (IME)**

| | |
|---|---|
| 0 | All interrupts are disabled and no vector interrupt is activated. |
| 1 | The interrupt enable flag corresponding to an interrupt request flag controls interrupt enabling/disabling. |

### 5.3 INTERRUPT SEQUENCE

The following flowchart shows the sequence of an interrupt.




**Notes 1.** IST1 and IST0: Interrupt status flags (Bits 3 and 2 of PSW. See **Table 5-3**.)

**2.** Each vector table must store the start address of the interrupt service program and the set values of the MBE and RBE at the start of an interrupt.

## 5.4 MULTIPLE INTERRUPT PROCESSING CONTROL

The μPD75238 can handle multiple interrupts by either of the following methods.

### (1) Multiple interrupt processing by a high-order interrupt

In this method, the μPD75238 selects an interrupt source among multiple interrupt sources, enabling double interrupt processing.

That is, the high-order interrupt specified by the interrupt priority specification register (IPS) is enabled when the processing status is 0 or 1. Other interrupts (interrupts lower than the specified high-order interrupt) are enabled only when the status is 0. (See **Fig. 5-6** and **Table 5-3**.)

**Fig. 5-6 Multiple Interrupt Processing by a High-Order Interrupt**

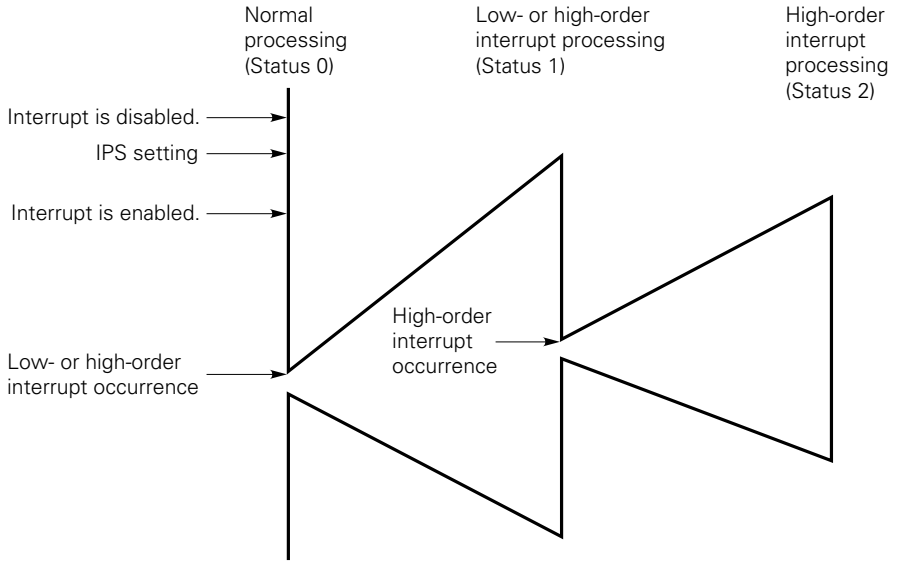



**Table 5-3 Interrupt Processing Statuses of IST1 and IST0**

| IST1 | IST0 | Processing status | CPU operation | Interrupts that can be accepted | After acceptance | |
|---|---|---|---|---|---|---|
| | | | | | IST1 | IST0 |
| 0 | 0 | Status 0 | Is processing the normal program. | All | 0 | 1 |
| 0 | 1 | Status 1 | Is processing a low- or high-order interrupt. | Only high-order interrupts | 1 | 0 |
| 1 | 0 | Status 2 | Is processing a high-order interrupt. | None | – | – |
| 1 | 1 | This status is disabled. | | | | |

IST1 and IST0 are saved with the remaining PSW in the stack memory when an interrupt is accepted and the status of IST0 and IST1 changes to a status one level higher. When an RETI instruction is executed, the former values of IST0 and IST1 are returned.

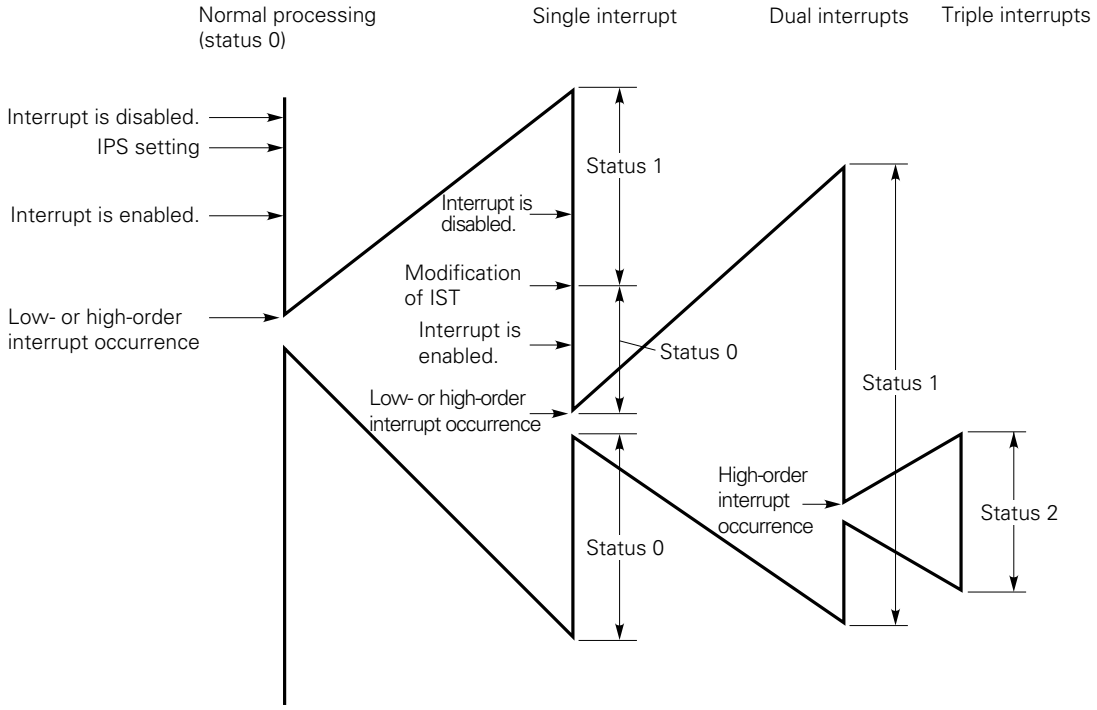
**(2) Multiple interrupt processing by changing the interrupt status flags**

As shown in Table 5-3, changing the interrupt status flags with the program causes multiple interrupts to be enabled. That is, when the interrupt processing program changes both IST1 and IST0 to 0 (status 0), multiple interrupt processing is enabled.

This method is used when two or more interrupts are to be enabled at a time or when the processing of three or more interrupts is to be performed.

When changing IST1 and IST0, interrupts must be disabled beforehand with a DI instruction.

**Fig. 5-7 Multiple Interrupt Processing by Changing the Interrupt Status Flags**



### 5.5 VECTOR ADDRESS SHARE INTERRUPT PROCESSING

Since interrupt sources INTBT and INT4 share the vector table, the following two cases must be considered.

**(1) When using only one interrupt source**

The interrupt enable flag corresponding to the required interrupt source of the two interrupt sources sharing the vector table is set and the other interrupt enable flag is cleared. In this case, the enabled interrupt source (IE××× = 1) issues an interrupt request. If this request is accepted, the corresponding interrupt request flag is reset. (The same operation as that of interrupts not sharing a vector address)

**(2) When using both interrupt sources**

The interrupt enable flags corresponding to the two interrupt sources are set. In this case, the logical and of the interrupt request flags corresponding to the two interrupt sources is an interrupt request.

Even if one or both of the interrupt request flags are set and an interrupt request is accepted, neither of the interrupt request flags is reset.

The interrupt service routine must therefore judge which interrupt source caused an interrupt. This is done by executing a DI instruction at the beginning of the interrupt service routine and checking the interrupt request flags with an SKTCLR instruction.

## 6. STANDBY FUNCTION

To reduce the power consumption when the program is in the wait state, the μPD75238 has two standby modes, STOP and HALT.

### 6.1 SETTING OF STANDBY MODES AND OPERATION STATUSES

**Table 6-1 Operation Statuses in the Standby Mode**

| | | STOP mode | HALT mode |
|---|---|---|---|
| Instruction for setting | | STOP instruction | HALT instruction |
| System clock at setting | | This mode can be set only when the main system clock is used. | This mode can be set when either the main system clock or the subsystem clock is used. |
| Operation status | Clock generator | Only the main system clock is stopped. | Only CPU clock (Φ) is stopped (with oscillation continued). |
| | Serial interface (Channel 0) | Operation is possible only when external $\overline{SCK0}$ input is selected for the serial clock. | Operation is possible only when the main system clock operates or external $\overline{SCK0}$ is used. |
| | Serial interface (Channel 1) | Operation is possible only when external $\overline{SCK1}$ input is selected for the serial clock. | Operation is possible only when the main system clock operates. |
| | Basic interval timer | Operation is stopped. | Operation is continued (to set IRQBT at reference time intervals). |
| | Timer/event counter | Operation is possible only when TI0 pin input is selected for the count clock. | Operation is possible. |
| | Watch timer | Operation is possible only when $f_{XT}$ is selected for the count clock. | Operation is possible. |
| | Timer/pulse generator | Operation is stopped. | Operation is possible only when the main system clock operates. |
| | Event counter | Operation is stopped. | Operation is possible only when the main system clock operates. |
| | A/D converter | Operation is stopped. | Operation is possible only when the main system clock operates. |
| | FIP controller/ driver | Operation is disabled. (The display off mode is selected before setting.) | |
| | External interrupt | INT0 is disabled. INT1, INT2, and INT4 are enabled. | |
| | CPU | Operation is stopped. | |
| Release signal | | Interrupt request signals sent out from hardware, which are enabled by interrupt enable flags, or $\overline{RESET}$ input. | |

A STOP instruction is used to set the STOP mode, and a HALT instruction is used to set the HALT mode. (A STOP instruction sets bit 3 of PCC, and a HALT instruction sets bit 2 of PCC.)

When changing a CPU operation clock pulse with the low-order two bits of PCC, a time lag may occur from the time when PCC is rewritten to the time when the CPU clock signal is changed as indicated in Table 4-1. When changing an operation clock pulse before the standby mode or a CPU clock signal after the standby mode is released, it is necessary to rewrite PCC and set the standby mode after the number of machine cycles required to change the CPU clock pulse elapses.

In a standby mode, the contents of all registers and data memory that are stopped during the standby mode, including general registers, flags, mode registers, and output latches, are retained.

**Cautions 1. When the STOP mode is set, the X1 input is internally connected to $V_{SS}$ (GND potential) to suppress leakage at the crystal oscillator circuitry. This means that the STOP mode cannot be used with a system that uses an external clock.**

**2. An interrupt request signal is used to release a standby mode.**
**This means that if an interrupt source whose interrupt request flag and interrupt enable flag are both set exists, the initiated standby mode is released immediately after it is set. When the STOP mode is set, therefore, the μPD75238 enters the HALT mode immediately after the STOP instruction is executed, then returns to the operation mode after the wait time specified by the BTM register has elapsed.**

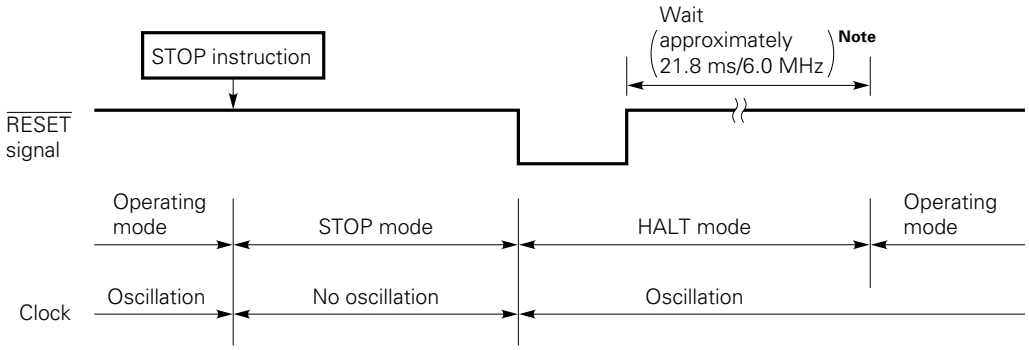
### 6.2 RELEASE OF THE STANDBY MODES

The STOP mode and HALT mode are released by a $\overline{\text{RESET}}$ input or the generation of an interrupt request signal[Note] that is enabled with the interrupt enable flag. Fig. 6-1 shows how the STOP and HALT modes are released.

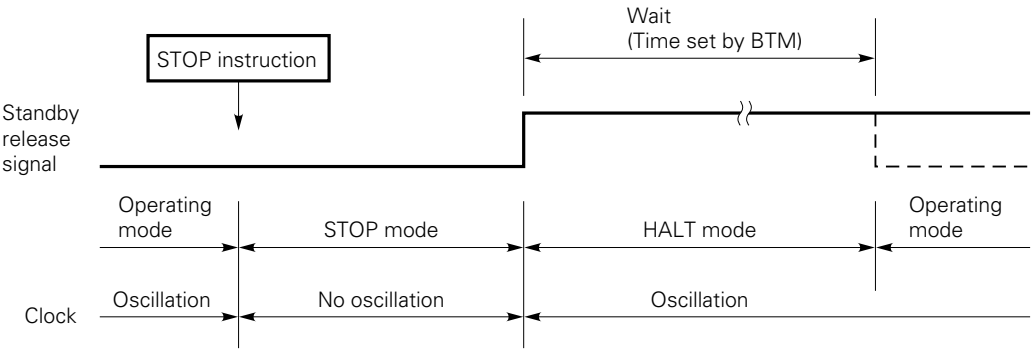
**Note** INT0 to INT2 are excluded.

**Fig. 6-1 Standby Mode Release Operation**

**(a) Release of the STOP mode by $\overline{\text{RESET}}$ input**

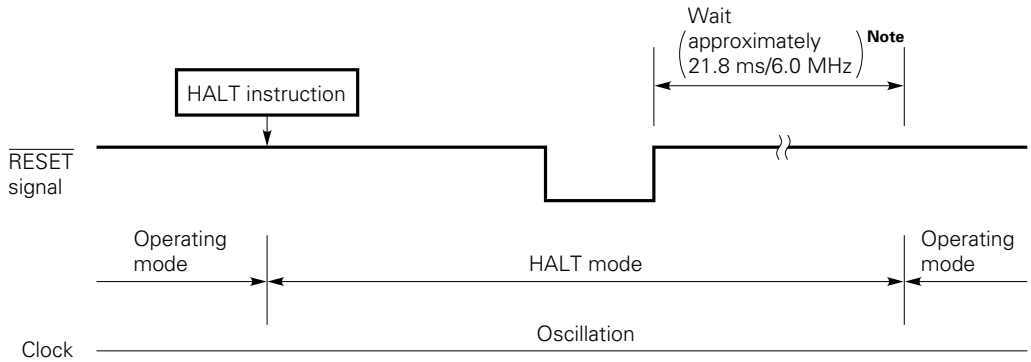



**(b) Release of the STOP mode by the occurrence of an interrupt**



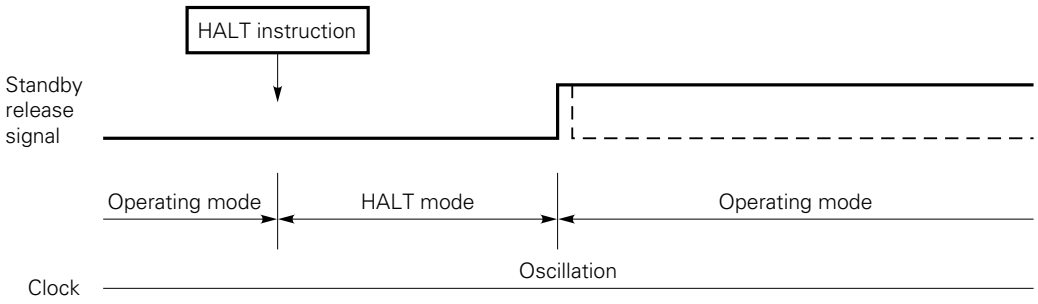

**Remark** The dashed line indicates the case where the interrupt request that releases the standby mode is accepted (IME = 1).

**(c) Release of the HALT mode by $\overline{\text{RESET}}$ input**



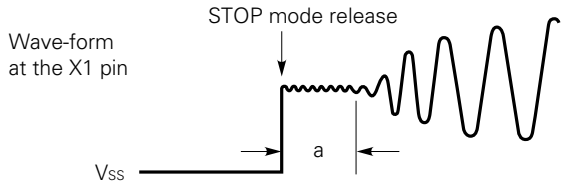

**Note** 31.3 ms at 4.19 MHz.

**(d) Release of the HALT mode by the occurrence of an interrupt**




**Remark** The dashed line indicates the case where the interrupt request that releases the standby mode is accepted (IME = 1).

The wait times used when the STOP mode is released do not include a time (a in the figure below) required before clock generation is started following the release of the STOP mode, regardless of whether the STOP mode is released by $\overline{\text{RESET}}$ signal input or the generation of an interrupt.




When the STOP mode is released by the occurrence of an interrupt, a wait time is determined by BTM. (See **Table 6-2**.)

**Table 6-2 Selection of a Wait Time with BTM**

**(When fx = 6.0 MHz)**

| BTM3 | BTM2 | BTM1 | BTM0 | Wait time[Note]. ( ) indicates the value for fx = 6.0 MHz |
|---|---|---|---|---|
| – | 0 | 0 | 0 | Approx. $2^{20}$/fx (Approx. 175 ms) |
| – | 0 | 1 | 1 | Approx. $2^{17}$/fx (Approx. 21.8 ms) |
| – | 1 | 0 | 1 | Approx. $2^{15}$/fx (Approx. 5.46 ms) |
| – | 1 | 1 | 1 | Approx. $2^{13}$/fx (Approx. 1.37 ms) |
| Other than above | | | | Use prohibited |

**(When fx = 4.19 MHz)**

| BTM3 | BTM2 | BTM1 | BTM0 | Wait time[Note]. ( ) indicates the value for fx = 4.19 MHz |
|---|---|---|---|---|
| – | 0 | 0 | 0 | Approx. $2^{20}$/fx (Approx. 250 ms) |
| – | 0 | 1 | 1 | Approx. $2^{17}$/fx (Approx. 31.3 ms) |
| – | 1 | 0 | 1 | Approx. $2^{15}$/fx (Approx. 7.82 ms) |
| – | 1 | 1 | 1 | Approx. $2^{13}$/fx (Approx. 1.95 ms) |
| Other than above | | | | Use prohibited |

**Note** This time does not include the time from the release of the STOP mode to the start of oscillation.

### 6.3 OPERATION AFTER A STANDBY MODE IS RELEASED

(1) If a standby mode is released by a $\overline{\text{RESET}}$ input, normal reset operation is performed.

(2) If a standby mode is released by the occurrence of an interrupt request, bit 3 of the IPS (IME) determines whether to perform a vectored interrupt when the CPU resumes instruction execution.

**(a) When IME = 0**

After the standby mode is released, execution of an instruction (NOP instruction) is restarted immediately after the instruction which set the standby mode.

The interrupt request flag is held.

**(b) When IME = 1**

After the standby mode is released, two instructions are executed, then a vector interrupt is caused. However, when the standby mode is released by INTW or INT2 (input of a testable signal), no vector interrupt is caused and the same processing as (a) above is performed.

## 7. RESET FUNCTION

Fig. 7-1 shows the configuration of the reset ($\overline{RES}$) signal generator.

**Fig. 7-1 Reset Signal Generator**

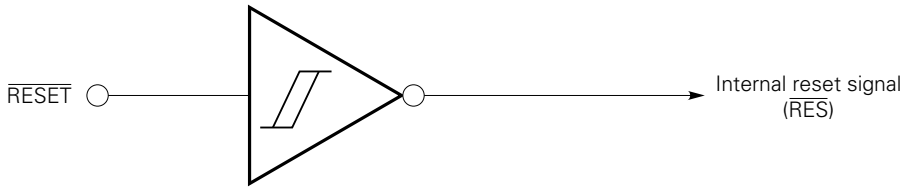
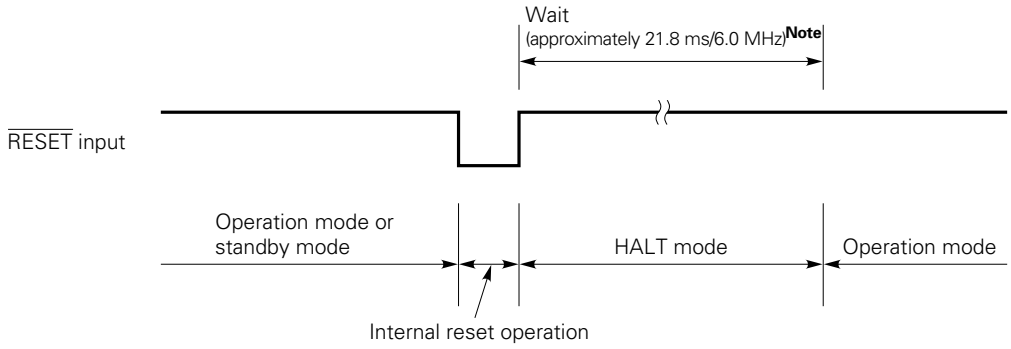



Fig. 7-2 shows reset operation.
The output buffer is set to off at the time of a $\overline{RESET}$ input.
After a reset, the hardware is initialized as indicated in Table 7-1.

**Fig. 7-2 Reset Operation by $\overline{RESET}$ Input**




**Note** 31.3 ms at 4.19 MHz.

After a reset, the hardware is initialized as indicated in Table 7-1.

**Table 7-1 Statuses of the Hardware after a Reset (1/2)**

| Hardware | | $\overline{\text{RESET}}$ input in a standby mode | $\overline{\text{RESET}}$ input during operation |
|---|---|---|---|
| Program counter (PC) | | Low-order 6 bits at address 0000H in program memory are set in PC bits 13 to 8, and the data at address 0001H are set in PC bits 7 to 0. | Low-order 6 bits at address 0000H in program memory are set in PC bits 13 to 8, and the data at address 0001H are set in PC bits 7 to 0. |
| PSW | Carry flag (CY) | Held | Undefined |
| | Skip flags (SK0 to SK2) | 0 | 0 |
| | Interrupt status flags (IST1, IST2) | 0 | 0 |
| | Bank enable flags (MBE, RBE) | Bit 6 at address 0000H in program memory is set in RBE, and bit 7 is set in MBE. | Bit 6 at address 0000H in program memory is set in RBE, and bit 7 is set in MBE. |
| Data memory (RAM) | | Held | Undefined |
| General registers (X, A, H, L, D, E, B, C) | | Held | Undefined |
| Bank select register (MBS, RBS) | | 0, 0 | 0, 0 |
| Stack pointer (SP) | | Undefined | Undefined |
| Stack bank select register (SBS) | | Undefined | Undefined |
| Basic interval timer | Counter (BT) | Undefined | Undefined |
| | Mode register (BTM) | 0 | 0 |
| Timer/event counter | Counter (T0) | 0 | 0 |
| | Modulo register (TMOD0) | FFH | FFH |
| | Mode register (TM0) | 0 | 0 |
| | TOE0, TOUT F/F | 0, 0 | 0, 0 |
| Watch timer | Mode register (WM) | 0 | 0 |
| Timer/pulse generator | Modulo registers (MODH, MODL) | Held | Held |
| | Mode register (TPGM) | 0 | 0 |
| Event counter | Counter (T1) | 0 | 0 |
| | Mode register (TM1) | 0 | 0 |
| | Gate control register (GATEC) | 0 | 0 |
| Serial interface (Channel 0) | Shift register (SIO0) | Held | Undefined |
| | Operation mode register (CSIM0) | 0 | 0 |
| | SBI control register (SBIC) | 0 | 0 |
| | Slave address register (SVA) | Held | Undefined |
| | P01/$\overline{\text{SCK0}}$ output latch | 1 | 1 |
| Serial interface (Channel 1) | Shift register (SIO1) | Held | Undefined |
| | Operation mode register (CSIM1) | 0 | 0 |
| | Serial transfer end flag (EOT) | 0 | 0 |

**Table 7-1 Statuses of the Hardware after a Reset (2/2)**

| Hardware | | $\overline{\text{RESET}}$ input in a standby mode | $\overline{\text{RESET}}$ input during operation |
|---|---|---|---|
| A/D converter | Mode register (ADM), EOC | 04H (EOC = 1) | 04H (EOC = 1) |
| | SA register | Undefined | Undefined |
| Bit sequential buffers (BSB0 to BSB3) | | Held | Undefined |
| FIP controller/ driver | Mode register (DSPM) | 0 | 0 |
| | Dimmer select register (DIMS) | 0 | 0 |
| | Digit select register (DIGS) | 8H | 8H |
| | Display data memory | Held | Held |
| | Output buffer | Off | Off |
| | Static mode registers (STATA, STATB) | 0, 0 | 0, 0 |
| Clock generator, clock output circuit | Processor clock control register (PCC) | 0 | 0 |
| | System clock control register (SCC) | 0 | 0 |
| | Clock output mode register (CLOM) | 0 | 0 |
| Interrupt | Interrupt request flag (IRQ×××) | Reset | Reset |
| | Interrupt enable flag (IE×××) | 0 | 0 |
| | Interrupt master enable flag (IME) | 0 | 0 |
| | INT0 and INT1 mode registers (IM0, IM1) | 0, 0 | 0, 0 |
| Digital ports | Output buffer (Ports 2 to 7) | Off | Off |
| | Output latch (Ports 2 to 7) | Clear | Clear |
| | I/O mode registers (PMGA, PMGB) | 0 | 0 |
| | Pull-up resistor specification register (POGA) | 0 | 0 |
| Ports 10 to 15 | Output buffer | Off | Off |
| | Output latch | 0 | 0 |
| Port H | Output latch | Held | Undefined |

# 8. INSTRUCTION SET

## 8.1 μPD75238 INSTRUCTIONS

### (1) GETI instruction

The GETI instruction references a two-byte table in the program memory and performs the following three types of operations. This single-byte instruction is very useful in reducing the number of program steps.

(a) A subroutine call is made to a 16-KB space (0000H to 3FFFH), regarding data in a table as the call address of a call instruction.
(b) A branch is made to a 16-KB space (0000H to 3FFFH), regarding data in a table as the branch address of a branch instruction.
(c) Data in a table is executed as a double-byte instruction excluding BRCB and the CALLF instructions.
(d) Data in a table is executed as the instruction code of two single-byte instructions.

As shown in Fig. 3-2, the tables to be referenced by a GETI instruction are located at addresses 0020H to 007FH in the program memory. That is, data can be set in up to 48 tables.
When describing a table address as an operand, describe an even address.

**Cautions 1. A two-byte instruction which can be referenced by a GETI instruction must be a two-machine-cycle instruction.**

**2. When referencing two single-byte instructions with a GETI instruction, only the combinations listed in the table below are valid.**

| Instruction of first byte | Instruction of second byte |
|---|---|
| MOV A,@HL<br>MOV @HL,A<br>XCH A,@HL | INCS L<br>DECS L<br>INCS H<br>DECS H<br>INCS HL |
| MOV A,@DE<br>XCH A,@DE | INCS E<br>DECS E<br>INCS D<br>DECS D<br>INCS DE |
| MOV A,@DL<br>XCH A,@DL | INCS L<br>DECS L<br>INCS D<br>DECS D |

**3. Branch and subroutine instructions can be referenced by the GETI instruction only when the addresses of the destination and subroutine call are in the 16K-byte space (0000H to 3FFFH). A branch or subroutine instruction to an address from 4000H to 7F7FH cannot be referenced by the GETI instruction.**

Since PC does not increment the counter during execution of a GETI instruction, control returns to the address next to the GETI instruction after the execution of the GETI instruction.

When the instruction before a GETI instruction has the skip function, the GETI instruction is skipped in the same way as for other single-byte instructions. When the instruction referenced by a GETI instruction has the skip function, the instructions after the GETI instruction are skipped.

When a string effect instruction is referenced by a GETI instruction, the following results are obtained.

- When the group of the string effect instruction before the GETI instruction is the same as that of the instruction referenced by the GETI instruction, the effect of the string effect instruction is canceled and the referenced instruction is not skipped.
- When the group of the instruction after the GETI instruction is the same as that of the instruction referenced by the GETI instruction, the effect of the string effect instruction caused by the referenced instruction is valid and the instructions after the referenced instruction are skipped.

**(2) Bit manipulation instructions**

The μPD75238 is provided with bit test instructions, bit transfer instructions, and bit Boolean instructions (AND, OR, and XOR) in addition to normal bit manipulation instructions (set instruction and clear instruction). Manipulation bits are specified by bit manipulation addressing.

There are three types of bit manipulation addressing. The table below lists the bits manipulated by each addressing.

| Addressing | Specifiable peripheral hardware | Specifiable bit address range |
|---|---|---|
| fmem.bit | RBE/MBE/IST1, IST0/IE×××/IRQ××× | FB0H to FBFH |
| | Port 0 to port 6 | FF0H to FFFH |
| pmem.@L | Port 0 and port 4 | FC0H to FFFH |
| @H+mem.bit | All the peripheral hardware (bit-manipulatable) | All the bits of the memory bank specified by MB (bit-manipulatable) |

(××× : 0, 1, 2, 4, BT, T0, TPG, CSI0, KS, W
MB = MBE• MBS)

**(3) String effect instructions**

When two or more instructions in the same group (group A or B) are placed at two or more string effect addresses, the instruction placed at the start point of the string effect instructions is executed. After that, each string effect instruction is executed as an NOP instruction.

Group A: MOV A, #n4, MOV XA, #n8
Group B: MOV HL, #n8

**(4) Base conversion instruction**

The μPD75238 is provided with base conversion instructions to convert the results of addition and subtraction of 4-bit data to a base-n number.

When a base-m number is to be obtained, the following combinations of instructions are used for adjustment.

- For addition
  ADDS A, #16-m
  ADDC A, @HL
  ADDS A, #m
- For subtraction
  SUBC A, @HL
  ADDS A, #m

The result of adding or subtracting the contents of the accumulator and the memory addressed by the HL register pair is converted to a base-m number. However, for subtraction, the complement of the obtained result (base-m number) is set in the accumulator. An overflow or underflow is reflected in the carry flag. (In the above combinations, the skip function of the ADDS A, #m instruction is prohibited.)

## 8.2 INSTRUCTION SET AND ITS OPERATION

### (1) Representation format and description method of operands

An operand is described in the operand field of each instruction according to the description method corresponding to the operand representation format of the instruction. Refer to the assembler specifications for details. When two or more elements are described in the description method field, select one of them. Uppercase letters, a plus sign (+), and a minus sign (-) are keywords, so they can be used without alteration.

Specify an appropriate numeric value or label for immediate data.

The symbols shown in format diagrams in Chapters 3 to 5 can be used as a label instead of mem, fmem, pmem, and bit. However, there are some restrictions on usable labels for fmem and pmem. (See **(2)** in **Section 8.1**.)

| Representation format | Description method |
|---|---|
| reg | X, A, B, C, D, E, H, L |
| reg1 | X, B, C, D, E, H, L |
| rp | XA, BC, DE, HL |
| rp1 | BC, DE, HL |
| rp2 | BC, DE |
| rp' | XA, BC, DE, HL, XA', BC', DE', HL' |
| rp'1 | BC, DE, HL, XA', BC', DE', HL' |
| rpa | HL, HL+, HL-, DE, DL |
| rpa1 | DE, DL |
| n4 | 4-bit immediate data or label |
| n8 | 8-bit immediate data or label |
| mem | 8-bit immediate data or label[Note] |
| bit | 2-bit immediate data or label |
| fmem | FB0H-FBFH/FF0H-FFFH immediate data or label |
| pmem | FC0H-FFFH immediate data or label |
| addr1 | 0000H-7F7FH immediate data or label |
| addr | 0000H-3F7FH immediate data or label |
| caddr | 12-bit immediate data or label |
| faddr | 11-bit immediate data or label |
| taddr | 20H-7FH immediate data (bit 0 = 0) or label |
| PORTn | PORT0-PORT15 |
| IE××× | IEBT, IECSI0, IET0, IETPG, IE0, IE1, IE2, IEKS, IEW, IE4 |
| RBn | RB0-RB3 |
| MBn | MB0, MB1, MB2, MB3, MB15 |

**Note** Only even addresses can be specified for 8-bit data processing.

**(2) Legend**

| | | |
|---|---|---|
| A | : | A register, 4-bit accumulator |
| B | : | B register, 4-bit accumulator |
| C | : | C register, 4-bit accumulator |
| D | : | D register, 4-bit accumulator |
| E | : | E register, 4-bit accumulator |
| H | : | H register, 4-bit accumulator |
| L | : | L register, 4-bit accumulator |
| X | : | X register, 4-bit accumulator |
| XA | : | Register pair (XA), 8-bit accumulator |
| BC | : | Register pair (BC), 8-bit accumulator |
| DE | : | Register pair (DE), 8-bit accumulator |
| HL | : | Register pair (HL), 8-bit accumulator |
| XA′ | : | Extended register pair (XA′) |
| BC′ | : | Extended register pair (BC′) |
| DE′ | : | Extended register pair (DE′) |
| HL′ | : | Extended register pair (HL′) |
| PC | : | Program counter |
| SP | : | Stack pointer |
| SBS | : | Stack bank select register |
| CY | : | Carry flag, bit accumulator |
| PSW | : | Program status word |
| MBE | : | Memory bank enable flag |
| RBE | : | Register bank enable flag |
| PORTn | : | Port n (n = 0 to 15) |
| IME | : | Interrupt master enable flag |
| IPS | : | Interrupt priority specification register |
| IE×××| : | Interrupt enable flag |
| RBS | : | Register bank select register |
| MBS | : | Memory bank select register |
| PCC | : | Processor clock control register |
| . | : | Address/bit delimiter |
| (××) | : | Contents addressed by ×× |
| ××H | : | Hexadecimal data |

### (3) Explanation of the symbols in the addressing area field

| | | |
|---|---|---|
| *1 | MB = MBE•MBS<br>(MBS = 0, 1, 2, 3, or 15) | Data memory addressing |
| *2 | MB = 0 | |
| *3 | MBE = 0: MB = 0 (00H-7FH)<br>MB = 15 (80H-FFH)<br>MBE = 1: MB = MBS (MBS = 0, 1, 2, 3, or 15) | |
| *4 | MB = 15, fmem = FB0H-FBFH or<br>FF0H-FFFH | |
| *5 | MB = 15, pmem = FC0H-FFFH | |
| *6 | addr = 0000H-3FFFH | Program memory addressing |
| *7 | addr = (Current PC) -15 to (Current PC) -1 or<br>(Current PC) +2 to (Current PC) +16 | |
| *8 | caddr = 0000H-0FFFH ($PC_{14,13,12}$ = 00B) or<br>1000H-1FFFH ($PC_{14,13,12}$ = 01B) or<br>2000H-2FFFH ($PC_{14,13,12}$ = 10B) or<br>3000H-3FFFH ($PC_{14,13,12}$ = 11B) or<br>4000H-4FFFH ($PC_{14,13,12}$ = 100B) or<br>5000H-5FFFH ($PC_{14,13,12}$ = 101B) or<br>6000H-6FFFH ($PC_{14,13,12}$ = 110B) or<br>7000H-7F7FH ($PC_{14,13,12}$ = 111B) | |
| *9 | faddr = 0000H-07FFH | |
| *10 | taddr = 0020H-007FH | |
| *11 | addr1 = 0000H-7F7FH | |

**Remarks 1.** MB indicates an accessible memory bank.

**2.** For *2, MB is always 0 irrespective of MBE and MBS.

**3.** For *4 and *5, MB is always 15 irrespective of MBE and MBS.

**4.** *6 to *11 indicate each addressable area.

### (4) Explanation of the machine cycle column

S represents the number of machine cycles required when a skip instruction with the skip function performs a skip operation. S assumes one of the following values:

- When no skip operation is performed : S = 0
- When a 1-byte instruction or 2-byte instruction is skipped: S = 1
- When a 3-byte instruction is skipped : S = 2

**Caution The GETI instruction is skipped in one machine cycle.**

One machine cycle is equal to one cycle (= $t_{CY}$) of the CPU clock (Φ), and five types of times are available for selection according to the PCC and SCC settings. (See **(3)** in **Section 4.2**.)

| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Transfer | MOV | A,#n4 | 1 | 1 | A ← n4 | | String effect A |
| | | reg1,#n4 | 2 | 2 | reg1 ← n4 | | |
| | | XA,#n8 | 2 | 2 | XA ← n8 | | String effect A |
| | | HL,#n8 | 2 | 2 | HL ← n8 | | String effect B |
| | | rp2,#n8 | 2 | 2 | rp2 ← n8 | | |
| | | A,@HL | 1 | 1 | A ← (HL) | *1 | |
| | | A,@HL+ | 1 | 2 + S | A ← (HL), then L ← L + 1 | *1 | L = 0 |
| | | A,@HL- | 1 | 2 + S | A ← (HL), then L ← L - 1 | *1 | L = FH |
| | | A,@rpa1 | 1 | 1 | A ← (rpa1) | *2 | |
| | | XA,@HL | 2 | 2 | XA ← (HL) | *1 | |
| | | @HL,A | 1 | 1 | (HL) ← A | *1 | |
| | | @HL,XA | 2 | 2 | (HL) ← XA | *1 | |
| | | A,mem | 2 | 2 | A ← (mem) | *3 | |
| | | XA,mem | 2 | 2 | XA ← (mem) | *3 | |
| | | mem,A | 2 | 2 | (mem) ← A | *3 | |
| | | mem,XA | 2 | 2 | (mem) ← XA | *3 | |
| | | A,reg | 2 | 2 | A ← reg | | |
| | | XA,rp' | 2 | 2 | XA ← rp' | | |
| | | reg1,A | 2 | 2 | reg1 ← A | | |
| | | rp'1,XA | 2 | 2 | rp'1 ← XA | | |
| | XCH | A,@HL | 1 | 1 | A ↔ (HL) | *1 | |
| | | A,@HL+ | 1 | 2 + S | A ↔ (HL), then L ← L + 1 | *1 | L = 0 |
| | | A,@HL- | 1 | 2 + S | A ↔ (HL), then L ← L - 1 | *1 | L = FH |
| | | A,@rpa1 | 1 | 1 | A ↔ (rpa1) | *2 | |
| | | XA,@HL | 2 | 2 | XA ↔ (HL) | *1 | |
| | | A,mem | 2 | 2 | A ↔ (mem) | *3 | |
| | | XA,mem | 2 | 2 | XA ↔ (mem) | *3 | |
| | | A,reg1 | 1 | 1 | A ↔ reg1 | | |
| | | XA,rp' | 2 | 2 | XA ↔ rp' | | |
| Table reference | MOVT | XA,@PCDE | 1 | 3 | XA ← $(PC_{14-8}+DE)_{ROM}$ | | |
| | | XA,@PCXA | 1 | 3 | XA ← $(PC_{14-8}+XA)_{ROM}$ | | |
| | | XA, @BCDE | 1 | 3 | XA ← $(BCDE)_{ROM}$ | *11 | |
| | | XA, @BCXA | 1 | 3 | XA ← $(BCXA)_{ROM}$ | *11 | |

| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Bit transfer | MOV1 | CY,fmem.bit | 2 | 2 | CY ← (fmem.bit) | *4 | |
| | | CY,pmem.@L | 2 | 2 | CY ← ($pmem_{7-2}$+$L_{3-2}$.bit($L_{1-0}$)) | *5 | |
| | | CY,@H+mem.bit | 2 | 2 | CY ← (H+$mem_{3-0}$.bit) | *1 | |
| | | fmem.bit,CY | 2 | 2 | (fmem.bit) ← CY | *4 | |
| | | pmem.@L,CY | 2 | 2 | ($pmem_{7-2}$+$L_{3-2}$.bit($L_{1-0}$)) ← CY | *5 | |
| | | @H+mem.bit,CY | 2 | 2 | (H+$mem_{3-0}$.bit) ← CY | *1 | |
| Arithmetic/logical | ADDS | A,#n4 | 1 | 1 + S | A ← A + n4 | | carry |
| | | XA,#n8 | 2 | 2 + S | XA ← XA + n8 | | carry |
| | | A,@HL | 1 | 1 + S | A ← A + (HL) | *1 | carry |
| | | XA,rp′ | 2 | 2 + S | XA ← XA + rp′ | | carry |
| | | rp′1,XA | 2 | 2 + S | rp′1 ← rp′1 + XA | | carry |
| | ADDC | A,@HL | 1 | 1 | A,CY ← A + (HL) + CY | *1 | |
| | | XA,rp′ | 2 | 2 | XA,CY ← XA + rp′ + CY | | |
| | | rp′1,XA | 2 | 2 | rp′1,CY ← rp′1 + XA + CY | | |
| | SUBS | A,@HL | 1 | 1 + S | A ← A - (HL) | *1 | borrow |
| | | XA,rp′ | 2 | 2 + S | XA ← XA - rp′ | | borrow |
| | | rp′1,XA | 2 | 2 + S | rp′1 ← rp′1 - XA | | borrow |
| | SUBC | A,@HL | 1 | 1 | A,CY ← A - (HL) - CY | *1 | |
| | | XA,rp′ | 2 | 2 | XA,CY ← XA - rp′ - CY | | |
| | | rp′1,XA | 2 | 2 | rp′1,CY ← rp′1 - XA - CY | | |
| | AND | A,#n4 | 2 | 2 | A ← A ∧ n4 | | |
| | | A,@HL | 1 | 1 | A ← A ∧ (HL) | *1 | |
| | | XA,rp′ | 2 | 2 | XA ← XA ∧ rp′ | | |
| | | rp′1,XA | 2 | 2 | rp′1 ← rp′1 ∧ XA | | |
| | OR | A,#n4 | 2 | 2 | A ← A ∨ n4 | | |
| | | A,@HL | 1 | 1 | A ← A ∨ (HL) | *1 | |
| | | XA,rp′ | 2 | 2 | XA ← XA ∨ rp′ | | |
| | | rp′1,XA | 2 | 2 | rp′1 ← rp′1 ∨ XA | | |
| | XOR | A,#n4 | 2 | 2 | A ← A ∀ n4 | | |
| | | A,@HL | 1 | 1 | A ← A ∀ (HL) | *1 | |
| | | XA,rp′ | 2 | 2 | XA ← XA ∀ rp′ | | |
| | | rp′1,XA | 2 | 2 | rp′1 ← rp′1 ∀ XA | | |
| Accumulator manipulation | RORC | A | 1 | 1 | CY ← $A_0$,$A_3$ ← CY, $An_{-1}$ ← An | | |
| | NOT | A | 2 | 2 | A ← $\overline{A}$ | | |

| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Increment/ decrement | INCS | reg | 1 | 1 + S | reg ← reg + 1 | | reg = 0 |
| | | rp1 | 1 | 1 + S | rp1 ← rp1 + 1 | | rp1 = 00H |
| | | @HL | 2 | 2 + S | (HL) ← (HL) + 1 | *1 | (HL) = 0 |
| | | mem | 2 | 2 + S | (mem) ← (mem) + 1 | *3 | (mem) = 0 |
| | DECS | reg | 1 | 1 + S | reg ← reg - 1 | | reg = FH |
| | | rp′ | 2 | 2 + S | rp′ ← rp′ - 1 | | rp′ = FFH |
| Comparison | SKE | reg,#n4 | 2 | 2 + S | Skip if reg = n4 | | reg = n4 |
| | | @HL,#n4 | 2 | 2 + S | Skip if (HL) = n4 | *1 | (HL) = n4 |
| | | A,@HL | 1 | 1 + S | Skip if A = (HL) | *1 | A = (HL) |
| | | XA,@HL | 2 | 2 + S | Skip if XA = (HL) | *1 | XA = (HL) |
| | | A,reg | 2 | 2 + S | Skip if A = reg | | A = reg |
| | | XA,rp′ | 2 | 2 + S | Skip if XA = rp′ | | XA = rp′ |
| Carry flag manipulation | SET1 | CY | 1 | 1 | CY ← 1 | | |
| | CLR1 | CY | 1 | 1 | CY ← 0 | | |
| | SKT | CY | 1 | 1 + S | Skip if CY = 1 | | CY = 1 |
| | NOT1 | CY | 1 | 1 | CY ← $\overline{CY}$ | | |

| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Memory bit manipulation | SET1 | mem.bit | 2 | 2 | (mem.bit) ← 1 | *3 | |
| | | fmem.bit | 2 | 2 | (fmem.bit) ← 1 | *4 | |
| | | pmem.@L | 2 | 2 | (pmem$_{7-2}$+L$_{3-2}$.bit(L$_{1-0}$)) ← 1 | *5 | |
| | | @H+mem.bit | 2 | 2 | (H+mem$_{3-0}$.bit) ← 1 | *1 | |
| | CLR1 | mem.bit | 2 | 2 | (mem.bit) ← 0 | *3 | |
| | | fmem.bit | 2 | 2 | (fmem.bit) ← 0 | *4 | |
| | | pmem.@L | 2 | 2 | (pmem$_{7-2}$+L$_{3-2}$.bit(L$_{1-0}$)) ← 0 | *5 | |
| | | @H+mem.bit | 2 | 2 | (H+mem$_{3-0}$.bit) ← 0 | *1 | |
| | SKT | mem.bit | 2 | 2 + S | Skip if (mem.bit) = 1 | *3 | (mem.bit) = 1 |
| | | fmem.bit | 2 | 2 + S | Skip if (fmem.bit) = 1 | *4 | (fmem.bit) = 1 |
| | | pmem.@L | 2 | 2 + S | Skip if (pmem$_{7-2}$+L$_{3-2}$.bit(L$_{1-0}$)) = 1 | *5 | (pmem.@L) = 1 |
| | | @H+mem.bit | 2 | 2 + S | Skip if (H+mem$_{3-0}$.bit) = 1 | *1 | (@H+mem.bit) = 1 |
| | SKF | mem.bit | 2 | 2 + S | Skip if (mem.bit) = 0 | *3 | (mem.bit) = 0 |
| | | fmem.bit | 2 | 2 + S | Skip if (fmem.bit) = 0 | *4 | (fmem.bit) = 0 |
| | | pmem.@L | 2 | 2 + S | Skip if (pmem$_{7-2}$+L$_{3-2}$.bit(L$_{1-0}$)) = 0 | *5 | (pmem.@L) = 0 |
| | | @H+mem.bit | 2 | 2 + S | Skip if (H+mem$_{3-0}$.bit) = 0 | *1 | (@H+mem.bit) = 0 |
| | SKTCLR | fmem.bit | 2 | 2 + S | Skip if (fmem.bit) = 1 and clear | *4 | (fmem.bit) = 1 |
| | | pmem.@L | 2 | 2 + S | Skip if (pmem$_{7-2}$ + L$_{3-2}$.bit(L$_{1-0}$)) = 1 and clear | *5 | (pmem.@L) = 1 |
| | | @H+mem.bit | 2 | 2 + S | Skip if (H+mem$_{3-0}$.bit) = 1 and clear | *1 | (@H+mem.bit) = 1 |
| | AND1 | CY,fmem.bit | 2 | 2 | CY ← CY ∧ (fmem.bit) | *4 | |
| | | CY,pmem.@L | 2 | 2 | CY ← CY ∧ (pmem$_{7-2}$+L$_{3-2}$.bit(L$_{1-0}$)) | *5 | |
| | | CY,@H+mem.bit | 2 | 2 | CY ← CY ∧ (H+mem$_{3-0}$.bit) | *1 | |
| | OR1 | CY,fmem.bit | 2 | 2 | CY ← CY ∨ (fmem.bit) | *4 | |
| | | CY,pmem.@L | 2 | 2 | CY ← CY ∨ (pmem$_{7-2}$+L$_{3-2}$.bit(L$_{1-0}$)) | *5 | |
| | | CY,@H+mem.bit | 2 | 2 | CY ← CY ∨ (H+mem$_{3-0}$.bit) | *1 | |
| | XOR1 | CY,fmem.bit | 2 | 2 | CY ← CY ∀ (fmem.bit) | *4 | |
| | | CY,pmem.@L | 2 | 2 | CY ← CY ∀ (pmem$_{7-2}$+L$_{3-2}$.bit(L$_{1-0}$)) | *5 | |
| | | CY,@H+mem.bit | 2 | 2 | CY ← CY ∀ (H+mem$_{3-0}$.bit) | *1 | |
| Branch | BR | addr | — | — | PC$_{14-0}$ ← addr1 (The assembler selects an appropriate instruction from the BR !addr, BRA !addr1, BRCB !caddr, and BR $addr1 instructions.) | *11 | |
| | | $addr 1 | 1 | 2 | PC$_{14-0}$ ← addr1 | *7 | |
| | | !addr | 3 | 3 | PC$_{14}$ ← 0, PC$_{13-0}$ ← !addr | *6 | |
| | | PCDE | 2 | 3 | PC$_{14-0}$ ← PC$_{14-8}$ + DE | | |
| | | PCXA | 2 | 3 | PC$_{14-0}$ ← PC$_{14-8}$ + XA | | |
| | | BCDE | 2 | 3 | PC$_{14-0}$ ← BCDE | | |
| | | BCXA | 2 | 3 | PC$_{14-0}$ ← BCXA | | |
| | BRA | !addr1 | 3 | 3 | PC$_{14-0}$ ← !addr1 | *11 | |
| | BRCB | !caddr | 2 | 2 | PC$_{14-0}$ ← PC$_{14,13,12}$ + caddr$_{11-0}$ | *8 | |

★

| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Subroutine stack control | CALL | !addr | 3 | 4 | (SP-5)(SP-6)(SP-3)(SP-4) ← $PC_{14-0}$<br>(SP-2) ← ×, ×, MBE, RBE<br>$PC_{14}$ ← 0, $PC_{13-0}$ ← addr, SP ← SP-6 | *6 | |
| | CALLA | !addr1 | 3 | 3 | (SP-5)(SP-6)(SP-3)(SP-4) ← $PC_{14-0}$<br>(SP-2) ← ×, ×, MBE, RBE<br>$PC_{14-0}$ ← addr1, SP ← SP-6 | *11 | |
| | CALLF | !faddr | 2 | 3 | (SP-5)(SP-6)(SP-3)(SP-4) ← $PC_{14-0}$<br>(SP-2) ← ×, ×, MBE, RBE<br>$PC_{14-0}$ ← 0000, faddr, SP ← SP-6 | *9 | |
| | RET | | 1 | 3 | ×, ×, MBE, RBE ← (SP+4)<br>$PC_{14-0}$ ← (SP+1)(SP)(SP+3)(SP+2)<br>SP ← SP+6 | | |
| | RETS | | 1 | 3 + S | ×, ×, MBE, RBE ← (SP+4)<br>$PC_{14-0}$ ← (SP+1) (SP)(SP+3)(SP+2)<br>SP ← SP+6<br>then skip unconditionally | | Unconditionally |
| | RETI | | 1 | 3 | ×, $PC_{14,13,12}$ ← (SP+1)<br>$PC_{11-0}$ ← (SP)(SP+3)(SP+2)<br>PSW ← (SP+4)(SP+5), SP ← SP+6 | | |
| | PUSH | rp | 1 | 1 | (SP-1)(SP-2) ← rp, SP ← SP-2 | | |
| | | BS | 2 | 2 | (SP-1) ← MBS, (SP-2) ← RBS, SP ← SP-2 | | |
| | POP | rp | 1 | 1 | rp ← (SP+1)(SP), SP ← SP+2 | | |
| | | BS | 2 | 2 | MBS ← (SP+1), RBS ← (SP), SP ← SP+2 | | |
| Interrupt control | EI | | 2 | 2 | IME(IPS.3) ← 1 | | |
| | | IE×××| 2 | 2 | IE××× ← 1 | | |
| | DI | | 2 | 2 | IME(IPS.3) ← 0 | | |
| | | IE××× | 2 | 2 | IE××× ← 0 | | |
| I/O | IN[Note] | A,PORTn | 2 | 2 | A ← PORTn | | |
| | | XA,PORTn | 2 | 2 | XA ← $PORTn_{+1}$, PORTn | | |
| | OUT[Note] | PORTn,A | 2 | 2 | PORTn ← A | | |
| | | PORTn,XA | 2 | 2 | $PORTn_{+1}$, PORTn ← XA | | |
| CPU control | HALT | | 2 | 2 | Set HALT Mode (PCC.2 ← 1) | | |
| | STOP | | 2 | 2 | Set STOP Mode (PCC.3 ← 1) | | |
| | NOP | | 1 | 1 | No Operation | | |

**Note** MBE = 0, or MBE = 1 and MBS = 15 must be set when an IN/OUT instruction is executed.

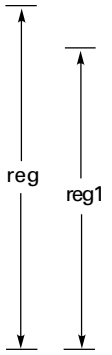
| Instruction | Mnemonic | Operand | Number of bytes | Machine cycle | Operation | Addressing area | Skip condition |
|---|---|---|---|---|---|---|---|
| Special | SEL | RBn | 2 | 2 | RBS ← n (n=0-3) | | |
| | | MBn | 2 | 2 | MBS ← n (n=0,1,2,3,15) | | |
| | GETI[Note] | taddr | 1 | 3 | • For a TBR instruction<br>$PC_{13-0} \leftarrow (taddr)_{5-0} + (taddr+1)$<br>$PC_{14} \leftarrow 0$ | *10 | ★ |
| | | | | 4 | • For a TCALL instruction<br>(SP-5)(SP-6)(SP-3)(SP-4) ← $PC_{14-0}$<br>(SP-2) ← ×, ×, MBE, RBE<br>$PC_{13-0} \leftarrow (taddr)_{5-0} + (taddr+1)$<br>SP ← SP-6 $PC_{14} \leftarrow 0$ | | |
| | | | | 3 | • For an instruction other than TBR and TCALL<br>Executes the instruction in (taddr)(taddr+1). | | Depends upon the referenced instruction. |

**Note** The TBR and TCALL instructions are table definition assembler pseudo instructions of the GETI instructions.

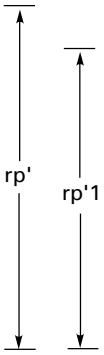
## 8.3 INSTRUCTION CODES OF EACH INSTRUCTION

### (1) Explanations of the symbols for the instruction codes

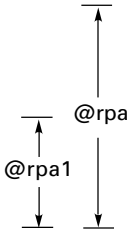
| $R_2$ | $R_1$ | $R_0$ | reg |
|---|---|---|---|
| 0 | 0 | 0 | A |
| 0 | 0 | 1 | X |
| 0 | 1 | 0 | L |
| 0 | 1 | 1 | H |
| 1 | 0 | 0 | E |
| 1 | 0 | 1 | D |
| 1 | 1 | 0 | C |
| 1 | 1 | 1 | B |



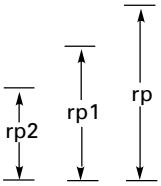

| $P_2$ | $P_1$ | $P_0$ | reg-pair |
|---|---|---|---|
| 0 | 0 | 0 | XA |
| 0 | 0 | 1 | XA' |
| 0 | 1 | 0 | HL |
| 0 | 1 | 1 | HL' |
| 1 | 0 | 0 | DE |
| 1 | 0 | 1 | DE' |
| 1 | 1 | 0 | BC |
| 1 | 1 | 1 | BC' |




| $Q_2$ | $Q_1$ | $Q_0$ | addressing |
|---|---|---|---|
| 0 | 0 | 1 | @HL |
| 0 | 1 | 0 | @HL+ |
| 0 | 1 | 1 | @HL– |
| 1 | 0 | 0 | @DE |
| 1 | 0 | 1 | @DL |




| $P_2$ | $P_1$ | reg-pair |
|---|---|---|
| 0 | 0 | XA |
| 0 | 1 | HL |
| 1 | 0 | DE |
| 1 | 1 | BC |




| $N_5$ | $N_2$ | $N_1$ | $N_0$ | IE××× |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | IEBT |
| 0 | 0 | 1 | 0 | IEW |
| 0 | 0 | 1 | 1 | IETPG |
| 0 | 1 | 0 | 0 | IET0 |
| 0 | 1 | 0 | 1 | IECSI0 |
| 0 | 1 | 1 | 0 | IE0 |
| 0 | 1 | 1 | 1 | IE2 |
| 1 | 0 | 0 | 0 | IE4 |
| 1 | 0 | 1 | 1 | IEKS |
| 1 | 1 | 1 | 0 | IE1 |

In : Immediate data for n4 or n8
Dn: Immediate data for mem
Bn: Immediate data for bit
Nn: Immediate data for n or IE×××
Tn: Immediate data for taddr × 1/2
An: Immediate data for the relative address distance (2 to 16) for the branch destination address minus one
Sn: Immediate data for the ones complement of the relative address distance (15 to 1) for the branch destination address

**(2) Bit manipulation addressing instruction codes**

*1 in the operand field indicates that there are three types of bit manipulation addressing, fmem.bit, pmem.@L, and @H+mem.bit.

The table below lists the second byte *2 of an instruction code corresponding to the above addressing.

| *1 | Second byte of instruction code | | | | | | | | Accessible bits |
|---|---|---|---|---|---|---|---|---|---|
| fmem.bit | 1 | 0 | $B_1$ | $B_0$ | $F_3$ | $F_2$ | $F_1$ | $F_0$ | FB0H - FBFH manipulatable bits |
| | 1 | 1 | $B_1$ | $B_0$ | $F_3$ | $F_2$ | $F_1$ | $F_0$ | FF0H - FFFH manipulatable bits |
| pmem.@L | 0 | 1 | 0 | 0 | $G_3$ | $G_2$ | $G_1$ | $G_0$ | FC0H-FFFH manipulatable bits |
| @H+mem.bit | 0 | 0 | $B_1$ | $B_0$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | Manipulatable bits of accessible memory bank |

Bn: Immediate data for bit
Fn: Immediate data for fmem (Low-order four bits of address)
Gn: Immediate data for pmem (Bits 2 to 5 of address)
Dn: Immediate data for mem (Low-order four bits of address)

| Instruction | Mnemonic | Operand | Instruction code | | |
|---|---|---|---|---|---|
| | | | $B_1$ | $B_2$ | $B_3$ |
| Transfer | MOV | A, #n4 | 0 1 1 1 $I_3$ $I_2$ $I_1$ $I_0$ | | |
| | | reg1, #n4 | 1 0 0 1 1 0 1 0 | $I_3$ $I_2$ $I_1$ $I_0$ 1 $R_2$ $R_1$ $R_0$ | |
| | | rp, #n8 | 1 0 0 0 1 $P_2$ $P_1$ 1 | $I_7$ $I_6$ $I_5$ $I_4$ $I_3$ $I_2$ $I_1$ $I_0$ | |
| | | A, @rpa | 1 1 1 0 0 $Q_2$ $Q_1$ $Q_0$ | | |
| | | XA, @HL | 1 0 1 0 1 0 1 0 | 0 0 0 1 1 0 0 0 | |
| | | @HL, A | 1 1 1 0 1 0 0 0 | | |
| | | @HL, XA | 1 0 1 0 1 0 1 0 | 0 0 0 1 0 0 0 0 | |
| | | A, mem | 1 0 1 0 0 0 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | XA, mem | 1 0 1 0 0 0 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ 0 | |
| | | mem, A | 1 0 0 1 0 0 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | mem, XA | 1 0 0 1 0 0 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ 0 | |
| | | A, reg | 1 0 0 1 1 0 0 1 | 0 1 1 1 1 $R_2$ $R_1$ $R_0$ | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 0 1 0 1 1 $P_2$ $P_1$ $P_0$ | |
| | | reg1, A | 1 0 0 1 1 0 0 1 | 0 1 1 1 0 $R_2$ $R_1$ $R_0$ | |
| | | rp'1, XA | 1 0 1 0 1 0 1 0 | 0 1 0 1 0 $P_2$ $P_1$ $P_0$ | |
| | XCH | A, @rpa | 1 1 1 0 1 $Q_2$ $Q_1$ $Q_0$ | | |
| | | XA, @HL | 1 0 1 0 1 0 1 0 | 0 0 0 1 0 0 0 1 | |
| | | A, mem | 1 0 1 1 0 0 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | XA, mem | 1 0 1 1 0 0 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ 0 | |
| | | A, reg1 | 1 1 0 1 1 $R_2$ $R_1$ $R_0$ | | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 0 1 0 0 0 $P_2$ $P_1$ $P_0$ | |
| Table reference | MOVT | XA, @PCDE | 1 1 0 1 0 1 0 0 | | |
| | | XA, @PCXA | 1 1 0 1 0 0 0 0 | | |
| | | XA, @BCDE | 1 1 0 1 0 1 0 1 | | |
| | | XA, @BCXA | 1 1 0 1 0 0 0 1 | | |
| Bit transfer | MOV1 | CY, [*1] | 1 0 1 1 1 1 0 1 | *2 | |
| | | [*1], CY | 1 0 0 1 1 0 1 1 | *2 | |

| Instruction | Mnemonic | Operand | Instruction code | | |
|---|---|---|---|---|---|
| | | | $B_1$ | $B_2$ | $B_3$ |
| Arithmetic/logical | ADDS | A, #n4 | 0 1 1 0 $I_3$ $I_2$ $I_1$ $I_0$ | | |
| | | XA, #n8 | 1 0 1 1 1 0 0 1 | $I_7$ $I_6$ $I_5$ $I_4$ $I_3$ $I_2$ $I_1$ $I_0$ | |
| | | A, @HL | 1 1 0 1 0 0 1 0 | | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 1 1 0 0 1 $P_2$ $P_1$ $P_0$ | |
| | | rp'1, XA | 1 0 1 0 1 0 1 0 | 1 1 0 0 0 $P_2$ $P_1$ $P_0$ | |
| | ADDC | A, @HL | 1 0 1 0 1 0 0 1 | | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 1 1 0 1 1 $P_2$ $P_1$ $P_0$ | |
| | | rp'1, XA | 1 0 1 0 1 0 1 0 | 1 1 0 1 0 $P_2$ $P_1$ $P_0$ | |
| | SUBS | A, @HL | 1 0 1 0 1 0 0 0 | | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 1 1 1 0 1 $P_2$ $P_1$ $P_0$ | |
| | | rp'1, XA | 1 0 1 0 1 0 1 0 | 1 1 1 0 0 $P_2$ $P_1$ $P_0$ | |
| | SUBC | A, @HL | 1 0 1 1 1 0 0 0 | | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 1 1 1 1 1 $P_2$ $P_1$ $P_0$ | |
| | | rp'1, XA | 1 0 1 0 1 0 1 0 | 1 1 1 1 0 $P_2$ $P_1$ $P_0$ | |
| | AND | A, #n4 | 1 0 0 1 1 0 0 1 | 0 0 1 1 $I_3$ $I_2$ $I_1$ $I_0$ | |
| | | A, @HL | 1 0 0 1 0 0 0 0 | | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 1 0 0 1 1 $P_2$ $P_1$ $P_0$ | |
| | | rp'1, XA | 1 0 1 0 1 0 1 0 | 1 0 0 1 0 $P_2$ $P_1$ $P_0$ | |
| | OR | A, #n4 | 1 0 0 1 1 0 0 1 | 0 1 0 0 $I_3$ $I_2$ $I_1$ $I_0$ | |
| | | A, @HL | 1 0 1 0 0 0 0 0 | | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 1 0 1 0 1 $P_2$ $P_1$ $P_0$ | |
| | | rp'1, XA | 1 0 1 0 1 0 1 0 | 1 0 1 0 0 $P_2$ $P_1$ $P_0$ | |
| | XOR | A, #n4 | 1 0 0 1 1 0 0 1 | 0 1 0 1 $I_3$ $I_2$ $I_1$ $I_0$ | |
| | | A, @HL | 1 0 1 1 0 0 0 0 | | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 1 0 1 1 1 $P_2$ $P_1$ $P_0$ | |
| | | rp'1, XA | 1 0 1 0 1 0 1 0 | 1 0 1 1 0 $P_2$ $P_1$ $P_0$ | |
| Accumulator manipulation | RORC | A | 1 0 0 1 1 0 0 0 | | |
| | NOT | A | 1 0 0 1 1 0 0 1 | 0 1 0 1 1 1 1 1 | |

| Instruction | Mnemonic | Operand | Instruction code | | |
|---|---|---|---|---|---|
| | | | $B_1$ | $B_2$ | $B_3$ |
| Increment/decrement | INCS | reg | 1 1 0 0 0 $R_2$ $R_1$ $R_0$ | | |
| | | rp1 | 1 0 0 0 1 $P_2$ $P_1$ 0 | | |
| | | @HL | 1 0 0 1 1 0 0 1 | 0 0 0 0 0 0 1 0 | |
| | | mem | 1 0 0 0 0 0 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | DECS | reg | 1 1 0 0 1 $R_2$ $R_1$ $R_0$ | | |
| | | rp' | 1 0 1 0 1 0 1 0 | 0 1 1 0 1 $P_2$ $P_1$ $P_0$ | |
| Comparison | SKE | reg, #n4 | 1 0 0 1 1 0 1 0 | $I_3$ $I_2$ $I_1$ $I_0$ 0 $R_2$ $R_1$ $R_0$ | |
| | | @HL, #n4 | 1 0 0 1 1 0 0 1 | 0 1 1 0 $I_3$ $I_2$ $I_1$ $I_0$ | |
| | | A, @HL | 1 0 0 0 0 0 0 0 | | |
| | | XA, @HL | 1 0 1 0 1 0 1 0 | 0 0 0 1 1 0 0 1 | |
| | | A, reg | 1 0 0 1 1 0 0 1 | 0 0 0 0 1 $R_2$ $R_1$ $R_0$ | |
| | | XA, rp' | 1 0 1 0 1 0 1 0 | 0 1 0 0 1 $P_2$ $P_1$ $P_0$ | |
| Carry flag manipulation | SET1 | CY | 1 1 1 0 0 1 1 1 | | |
| | CLR1 | CY | 1 1 1 0 0 1 1 0 | | |
| | SKT | CY | 1 1 0 1 0 1 1 1 | | |
| | NOT1 | CY | 1 1 0 1 0 1 1 0 | | |
| Memory bit manipulation | SET1 | mem.bit | 1 0 $B_1$ $B_0$ 0 1 0 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | [*1] | 1 0 0 1 1 1 0 1 | *2 | |
| | CLR1 | mem.bit | 1 0 $B_1$ $B_0$ 0 1 0 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | [*1] | 1 0 0 1 1 1 0 0 | *2 | |
| | SKT | mem.bit | 1 0 $B_1$ $B_0$ 0 1 1 1 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | [*1] | 1 0 1 1 1 1 1 1 | *2 | |
| | SKF | mem.bit | 1 0 $B_1$ $B_0$ 0 1 1 0 | $D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$ | |
| | | [*1] | 1 0 1 1 1 1 1 0 | *2 | |
| | SKTCLR | [*1] | 1 0 0 1 1 1 1 1 | *2 | |
| | AND1 | CY, [*1] | 1 0 1 0 1 1 0 0 | *2 | |
| | OR1 | CY, [*1] | 1 0 1 0 1 1 1 0 | *2 | |
| | XOR1 | CY, [*1] | 1 0 1 1 1 1 0 0 | *2 | |

| Instruction | Mnemonic | Operand | Instruction code | | |
|---|---|---|---|---|---|
| | | | $B_1$ | $B_2$ | $B_3$ |
| Branch | BR | !addr | 1 0 1 0 1 0 1 1 | 0 0 ← | addr → |
| | | $addr1 (+16) to (+2) | 0 0 0 0 $A_3$ $A_2$ $A_1$ $A_0$ | | |
| | | (-1) to (-15) | 1 1 1 1 $S_3$ $S_2$ $S_1$ $S_0$ | | |
| | | PCDE | 1 0 0 1 1 0 0 1 | 0 0 0 0 0 1 0 0 | |
| | | PCXA | 1 0 0 1 1 0 0 1 | 0 0 0 0 0 0 0 0 | |
| | | BCDE | 1 0 0 1 1 0 0 1 | 0 0 0 0 0 1 0 1 | |
| | | BCXA | 1 0 0 1 1 0 0 1 | 0 0 0 0 0 0 0 1 | |
| | BRA | !addr1 | 1 0 1 1 1 0 1 0 | 0 ← | addr1 → |
| | BRCB | !caddr | 0 1 0 1 ← | caddr → | |
| Subroutine stack control | CALL | !addr | 1 0 1 0 1 0 1 1 | 0 1 ← | addr → |
| | CALLA | !addr1 | 1 0 1 1 1 0 1 1 | 0 ← | addr1 → |
| | CALLF | !faddr | 0 1 0 0 0 ← | faddr → | |
| | RET | | 1 1 1 0 1 1 1 0 | | |
| | RETS | | 1 1 1 0 0 0 0 0 | | |
| | RETI | | 1 1 1 0 1 1 1 1 | | |
| | PUSH | rp | 0 1 0 0 1 $P_2$ $P_1$ 1 | | |
| | | BS | 1 0 0 1 1 0 0 1 | 0 0 0 0 0 1 1 1 | |
| | POP | rp | 0 1 0 0 1 $P_2$ $P_1$ 0 | | |
| | | BS | 1 0 0 1 1 0 0 1 | 0 0 0 0 0 1 1 0 | |
| I/O | IN | A, PORTn | 1 0 1 0 0 0 1 1 | 1 1 1 1 $N_3$ $N_2$ $N_1$ $N_0$ | |
| | | XA, PORTn | 1 0 1 0 0 0 1 0 | 1 1 1 1 $N_3$ $N_2$ $N_1$ $N_0$ | |
| | OUT | PORTn, A | 1 0 0 1 0 0 1 1 | 1 1 1 1 $N_3$ $N_2$ $N_1$ $N_0$ | |
| | | PORTn, XA | 1 0 0 1 0 0 1 0 | 1 1 1 1 $N_3$ $N_2$ $N_1$ $N_0$ | |
| Interrupt control | EI | | 1 0 0 1 1 1 0 1 | 1 0 1 1 0 0 1 0 | |
| | | IE××× | 1 0 0 1 1 1 0 1 | 1 0 $N_5$ 1 1 $N_2$ $N_1$ $N_0$ | |
| | DI | | 1 0 0 1 1 1 0 0 | 1 0 1 1 0 0 1 0 | |
| | | IE××× | 1 0 0 1 1 1 0 0 | 1 0 $N_5$ 1 1 $N_2$ $N_1$ $N_0$ | |
| CPU control | HALT | | 1 0 0 1 1 1 0 1 | 1 0 1 0 0 0 1 1 | |
| | STOP | | 1 0 0 1 1 1 0 1 | 1 0 1 1 0 0 1 1 | |
| | NOP | | 0 1 1 0 0 0 0 0 | | |
| Special | SEL | RBn | 1 0 0 1 1 0 0 1 | 0 0 1 0 0 0 $N_1$ $N_0$ | |
| | | MBn | 1 0 0 1 1 0 0 1 | 0 0 0 1 $N_3$ $N_2$ $N_1$ $N_0$ | |
| | GETI | taddr | 0 0 $T_5$ $T_4$ $T_3$ $T_2$ $T_1$ $T_0$ | | |

★

## 9. SPECIFICATION OF MASK OPTIONS

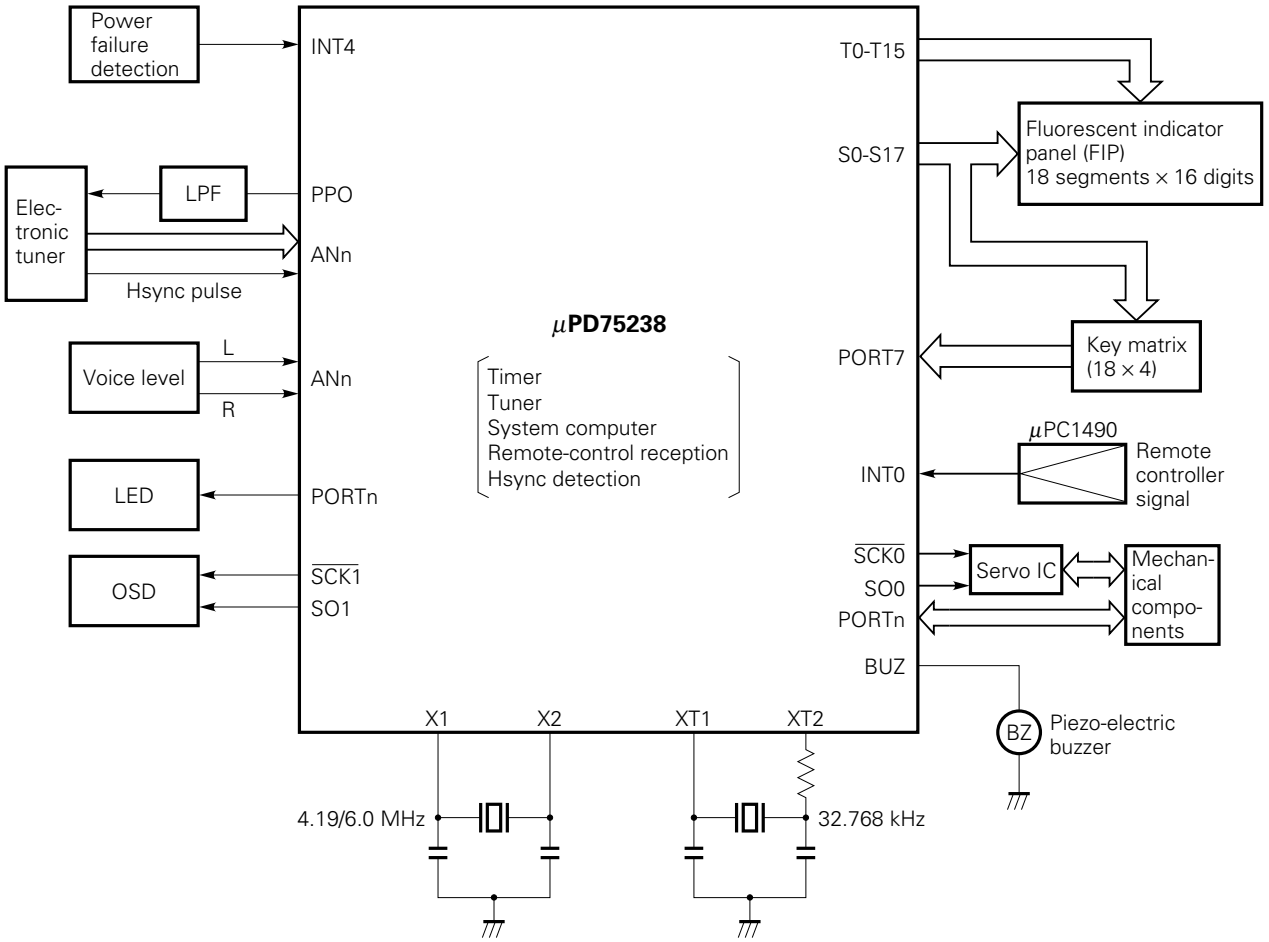
The μPD75238 provides the following mask options, which enable specifying whether to incorporate the elements below:

| Pin | Mask option |
|---|---|
| P40-P43 | Allows pull-up resistor to be contained bit by bit. |
| P50-P53 | |
| P70-P73 | Allows pull-down resistor to be contained bit by bit. |
| S0/P120-S3/P123 | Allows pull-down resistor to be contained to $V_{LOAD}$ bit by bit. |
| S4/P130-S7/P133 | |
| S8/P140, S9/P141 | |
| S10/T15/P142, S11/T14/P143 | |
| S12/T13/P150/PH0-S15/T10/P153/PH3 | |
| S16/P100-S19/P103 | Allows pull-down resistor[Note] to be contained to $V_{LOAD}$ or $V_{SS}$ bit by bit. |
| S20/P110-S23/P113 | |
| XT1, XT2 | Allows feed-back resistor of the subsystem clock oscillator to be deleted. |

**Note** Select whether to incorporate the pull-down resistors in $V_{LOAD}$ or $V_{SS}$ in unit of eight bits.

**Caution In systems without using subsystem clock, power consumption in STOP mode can be more reduced by deleting feed-back resistor of the oscillator.**

## 10. APPLICATION BLOCK DIAGRAM




**Remark** LPF : Low pass filter
OSD : On screen display
Hsync: Horizontal synchronous

## 11. ELECTRICAL CHARACTERISTICS

**ABSOLUTE MAXIMUM RATINGS** ($T_a$ = 25 °C)

| Parameter | Symbol | Conditions | | Rating | Unit |
|---|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | | | -0.3 to +7.0 | V |
| | $V_{LOAD}$ | | | $V_{DD}$ - 40 to $V_{DD}$ + 0.3 | V |
| Input voltage | $V_{I1}$ | Ports other than ports 4 and 5 | | -0.3 to $V_{DD}$ + 0.3 | V |
| | $V_{I2}$ | Ports 4 and 5 | Built-in pull-up resistor | -0.3 to $V_{DD}$ + 0.3 | V |
| | | | Open drain | -0.3 to +11 | V |
| Output voltage | $V_O$ | Pins other than display output pins | | -0.3 to $V_{DD}$ + 0.3 | V |
| | $V_{OD}$ | Display output pins | | $V_{DD}$ - 40 to $V_{DD}$ + 0.3 | V |
| High-level output current | $I_{OH}$ | One of pins other than display output pins | | -15 | mA |
| | | One pin of S0 to S9 and S16 to S23 | | -15 | mA |
| | | One pin of T0 to T15 | | -30 | mA |
| | | All pins other than display output pins | | -30 | mA |
| | | All display output pins | | -120 | mA |
| Low-level output current | $I_{OL}$ | Each pin | Peak value | 30 | mA |
| | | | rms | 15 | mA |
| | | Total of all pins of ports 0, 2, 3, and 4 | Peak value | 100 | mA |
| | | | rms | 60 | mA |
| | | Total of all pins of ports 5 to 8 | Peak value | 100 | mA |
| | | | rms | 60 | mA |
| Total power dismission | $P_T$ | Plastic QFP | ($T_a$ = -40 to +70 °C) | 700 | mW |
| | | | ($T_a$ = -40 to +85 °C) | 510 | mW |
| Operating temperature | $T_{opt}$ | | | -40 to +85 | °C |
| Storage temperature | $T_{stg}$ | | | -65 to +150 | °C |

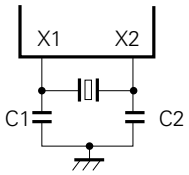
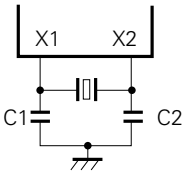
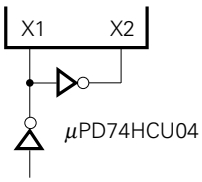
**RANGE OF SUPPLY VOLTAGE** ($T_a$ = -40 to +85 °C)

| Parameter | Min. | Max. | Unit | Conditions |
|---|---|---|---|---|
| CPU[Note 1] | [Note 2] | 6.0 | V | |
| Display controller | 4.5 | 6.0 | V | |
| Timer/pulse generator | 4.5 | 6.0 | V | |
| Hardware other than the above[Note 1] | 2.7 | 6.0 | V | |

**Notes 1.** The CPU does not include the system clock oscillator, the display controller, and the timer/pulse generator.

**2.** The range of the supply voltage at which the CPU can operate varies according to the cycle time. See the item of AC characteristics.

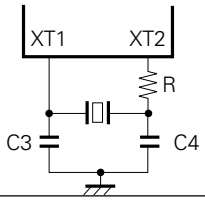
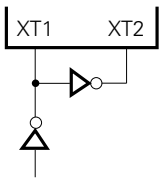
**CHARACTERISTICS OF THE MAIN SYSTEM CLOCK OSCILLATOR** ($T_a$ = -40 to +85 °C, $V_{DD}$ = 2.7 to 6.0 V)

| Resonator | Recommended constant | Parameter | Min. | Typ. | Max. | Unit | Conditions |
|---|---|---|---|---|---|---|---|
| Ceramic resonator |  | Oscillator frequency ($f_X$)**Note 1** | 2.0 | | 6.2 | MHz | $V_{DD}$ = oscillation voltage range |
| | | Oscillation stability time**Note 2** | | | 4 | ms | After $V_{DD}$ reaches Min. of the oscillation voltage range |
| Crystal resonator |  | Oscillator frequency ($f_X$)**Note 1** | 2.0 | 4.19 | 6.2 | MHz | |
| | | Oscillation stability time**Note 2** | | | 10 | ms | $V_{DD}$ = 4.5 to 6.0 V |
| | | | | | 30 | ms | |
| External clock |  | X1 input frequency ($f_X$)**Note 1** | 2.0 | | 6.2 | MHz | |
| | | X1 input high/low level width ($t_{XH}$, $t_{XL}$) | 81 | | 250 | ns | |

**Notes 1.** The oscillator frequency and input frequency indicate only the oscillator characteristics. See the item of AC characteristics for the instruction execution time.

**2.** The oscillation stability time means the time required for the oscillation to stabilize after $V_{DD}$ is applied or after the STOP mode is released.

**CHARACTERISTICS OF THE SUBSYSTEM CLOCK OSCILLATOR** ($T_a$ = -40 to +85 °C, $V_{DD}$ = 2.7 to 6.0 V)

| Resonator | Recommended constant | Parameter | Min. | Typ. | Max. | Unit | Conditions |
|---|---|---|---|---|---|---|---|
| Crystal resonator |  | Oscillator frequency ($f_{XT}$)**Note 1** | 32 | 32.768 | 35 | kHz | |
| | | Oscillation stability time**Note 2** | | 1.0 | 2 | s | $V_{DD}$ = 4.5 to 6.0 V |
| | | | | | 10 | s | |
| External clock |  | XT1 input frequency ($f_{XT}$)**Note 1** | 32 | | 100 | kHz | |
| | | XT1 input high/low level width ($t_{XTH}$, $t_{XTL}$) | 5 | | 15 | μs | |

**Notes 1.** The oscillator frequency and input frequency indicate only the oscillator characteristics. See the item of AC characteristics for the instruction execution time.

**2.** The oscillation stability time means the time required for the oscillation to stabilize after $V_{DD}$ is applied or after the STOP mode is released.

**CAPACITANCE** ($T_a$ = 25 °C, $V_{DD}$ = 0 V)

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions |
|---|---|---|---|---|---|---|
| Input capacitance | $C_I$ | | | 15 | pF | f = 1 MHz<br>0 V for pins other than pins to be measured |
| Output capacitance (Other than display output) | $C_O$ | | | 15 | pF | |
| I/O capacitance | $C_{IO}$ | | | 15 | pF | |
| Output capacitance (Display output) | $C_O$ | | | 35 | pF | |

**DC CHARACTERISTICS** ($T_a$ = -40 to +85 °C, $V_{DD}$ = 2.7 to 6.0 V)

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | | |
|---|---|---|---|---|---|---|---|---|
| High-level input voltage | $V_{IH1}$ | 0.7$V_{DD}$ | | $V_{DD}$ | V | Other than ports 0, 1, 4, 5, and 7, $\overline{RESET}$, P81, P83, X1, X2, and XT1 | | |
| | $V_{IH2}$ | 0.8$V_{DD}$ | | $V_{DD}$ | V | Ports 0 and 1, $\overline{RESET}$, P81, and P83 | | |
| | $V_{IH3}$ | $V_{DD}$ - 0.4 | | $V_{DD}$ | V | X1, X2, and XT1 | | |
| | $V_{IH4}$ | 0.65$V_{DD}$ | | $V_{DD}$ | V | Port 7 | $V_{DD}$ = 4.5 to 6.0 V | |
| | | 0.7$V_{DD}$ | | $V_{DD}$ | V | | | |
| | $V_{IH5}$ | 0.7$V_{DD}$ | | $V_{DD}$ | V | Ports 4 and 5 | Built-in pull-up resistor | |
| | | 0.7$V_{DD}$ | | 10 | V | | Open drain | |
| Low-level input voltage | $V_{IL1}$ | 0 | | 0.3$V_{DD}$ | V | Other than ports 0 and 1, $\overline{RESET}$, P81, P83, X1, X2, and XT1 | | |
| | $V_{IL2}$ | 0 | | 0.2$V_{DD}$ | V | Ports 0 and 1, $\overline{RESET}$, P81, and P83 | | |
| | $V_{IL3}$ | 0 | | 0.4 | V | X1, X2, and XT1 | | |
| High-level output voltage | $V_{OH}$ | $V_{DD}$ - 1.0 | | | V | All output pins (excl. ports 4 and 5, and P03) | $V_{DD}$ = 4.5 to 6.0 V | $I_{OH}$ = -1 mA |
| | | $V_{DD}$ - 0.5 | | | V | | $V_{DD}$ = 2.7 to 6.0 V | $I_{OH}$ = -100 μA |
| Low-level output voltage | $V_{OL}$ | | 0.4 | 2.0 | V | Ports 3, 4, and 5 | $V_{DD}$ = 4.5 to 6.0 V | $I_{OL}$ = 15 mA |
| | | | | 0.4 | V | All output pins | $V_{DD}$ = 4.5 to 6.0 V | $I_{OL}$ = 1.6 mA |
| | | | | 0.5 | V | | $V_{DD}$ = 2.7 to 6.0 V | $I_{OL}$ = 400 μA |
| | | | | 0.2$V_{DD}$ | V | SB0 and SB1 | Open drain pull-up resistor: 1 kΩ or more | |
| High-level input leakage current | $I_{LIH1}$ | | | 3 | μA | Other than X1, X2, XT1, Ports 4 and 5 | $V_{IN}$ = $V_{DD}$ | |
| | $I_{LIH2}$ | | | 20 | μA | X1, X2, and XT1 | | |
| | $I_{LIH3}$ | | | 20 | μA | Ports 4 and 5 | $V_{IN}$ = 10 V (open drain) | |
| Low-level input leakage current | $I_{LIL1}$ | | | -3 | μA | Other than X1, X2, and XT1 | $V_{IN}$ = 0 V | |
| | $I_{LIL2}$ | | | -20 | μA | X1, X2, and XT1 | | |
| High-level output leakage current | $I_{LOH1}$ | | | 3 | μA | Other than ports 4 and 5 | $V_{OUT}$ = $V_{DD}$ | |
| | $I_{LOH2}$ | | | 20 | μA | Ports 4 and 5 | $V_{OUT}$ = 10 V (open drain) | |
| Low-level output leakage current | $I_{LOL1}$ | | | -3 | μA | Other than display output | $V_{OUT}$ = 0 V | |
| | $I_{LOL2}$ | | | -10 | μA | Display output | $V_{OUT}$ = $V_{LOAD}$ = $V_{DD}$ - 35 V | |
| Display output current | $I_{OD}$ | -3 | -5.5 | | mA | S0 to S9, S16 to S23 | $V_{DD}$ = 4.5 to 6.0 V | |
| | | -15 | -22 | | mA | T0 to T15 | $V_{OD}$ = $V_{DD}$ - 2 V | |
| Built-in pull-down resistor (mask option) | $R_{P7}$ | 20 | 80 | 200 | kΩ | Port 7 $V_{IN}$ = $V_{DD}$ | $V_{DD}$ = 4.5 to 6.0 V | |
| | | 20 | | 1000 | kΩ | | | |
| | $R_L$ | 25 | 50 | 135 | kΩ | Display output | $V_{OD}$ - $V_{LOAD}$ = 35 V | |
| Built-in pull-up resistor | $R_{V1}$ | 15 | 40 | 80 | kΩ | Ports 0, 1, 2, 3, and 6 (excl. P00) $V_{IN}$ = 0 V | $V_{DD}$ = 5 V ±10 % | |
| | | 30 | | 300 | kΩ | | $V_{DD}$ = 3 V ±10 % | |
| | $R_{V2}$ | 15 | 40 | 70 | kΩ | Ports 4 and 5 $V_{OUT}$ = $V_{DD}$ - 2.0 V | $V_{DD}$ = 5 V ±10 % | |
| | | 10 | | 60 | kΩ | | $V_{DD}$ = 3 V ±10 % | |

**DC CHARACTERISTICS** ($T_a$ = -40 to +85 °C, $V_{DD}$ = 2.7 to 6.0 V)

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | | |
|---|---|---|---|---|---|---|---|---|
| Power supply current **Note 1** | $I_{DD1}$ | | 4.5 | 13.5 | mA | 6.0 MHz crystal resonance C1 = C2 = 22 pF**Note 4** | Operation mode | $V_{DD}$ = 5 V ±10 %**Note 2** |
| | | | 0.6 | 1.8 | mA | | | $V_{DD}$ = 3 V ±10 %**Note 3** |
| | $I_{DD2}$ | | 600 | 1800 | μA | | HALT mode | $V_{DD}$ = 5 V ±10 % |
| | | | 200 | 600 | μA | | | $V_{DD}$ = 3 V ±10 % |
| | $I_{DD1}$ | | 3 | 9 | mA | 4.19 MHz crystal resonance C1 = C2 = 22 pF**Note 4** | Operation mode | $V_{DD}$ = 5 V ±10 %**Note 2** |
| | | | 0.5 | 1.5 | mA | | | $V_{DD}$ = 3 V ±10 %**Note 3** |
| | $I_{DD2}$ | | 600 | 1800 | μA | | HALT mode | $V_{DD}$ = 5 V ±10 % |
| | | | 200 | 600 | μA | | | $V_{DD}$ = 3 V ±10 % |
| | $I_{DD3}$ | | 40 | 120 | μA | 32 kHz**Note 5** crystal resonance | Operation mode | $V_{DD}$ = 3 V ±10 % |
| | $I_{DD4}$ | | 5 | 15 | μA | | HALT mode | $V_{DD}$ = 3 V ±10 % |
| | $I_{DD5}$ | | 0.5 | 20 | μA | XT1 = 0 V STOP mode | $V_{DD}$ = 5 V ±10 % | |
| | | | 0.3 | 10 | μA | | $V_{DD}$ = 3 V ±10 % | $T_a$ = 25 °C |
| | | | | 5 | μA | | | |

**Notes 1.** This current excludes the current which flows through the built-in pull-down or pull-up resistors.

**2.** Value when the processor clock control register (PCC) is set to 0011 and the μPD75238 is operated in the high-speed mode

**3.** Value when the PCC is set to 0000 and the μPD75238 is operated in the low-speed mode

**4.** This value applies also when the subsystem clock oscillates.

**5.** This value applies when the system clock control resistor (SCC) is set to 1001 to stop the main system clock pulse and to start the subsystem clock pulse.

**A/D CONVERTER CHARACTERISTICS** ($T_a$ = -40 to +85 °C, $V_{DD}$ = 2.7 to 6.0 V, $AV_{SS}$ = $V_{SS}$ = 0 V, $AV_{DD}$ = $V_{DD}$)

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | |
|---|---|---|---|---|---|---|---|
| Resolution | | 8 | 8 | 8 | bit | | |
| Absolute accuracy**Note 1** | | | | ±1.5 | LSB | 2.5 V ≤ $AV_{REF}$ ≤ $AV_{DD}$ | -10 ≤ $T_a$ ≤ +85 °C ★ |
| | | | | ±2.0 | | | -40 ≤ $T_a$ < -10 °C |
| Conversion time | $t_{CONV}$ | | | 168/$f_X$ | μs | **Note 2** | |
| Sampling time | $t_{SAMP}$ | | | 44/$f_X$ | μs | **Note 3** | |
| Analog input voltage | $V_{IAN}$ | $AV_{SS}$ | | $AV_{REF}$ | V | | |
| Analog input impedance | $R_{AN}$ | | 1000 | | MΩ | | |
| $AV_{REF}$ current | $I_{AREF}$ | | 1.0 | 2.0 | mA | | |

**Notes 1.** Absolute accuracy excluding quantization error (±1/2 LSB)

**2.** Time from the execution of a conversion start instruction till EOC = 1 (28.0 μs at $f_X$ = 6.0 MHz, 40.1 μs at $f_X$ = 4.19 MHz)

**3.** Time from the execution of a conversion start instruction till the end of sampling (7.33 μs at $f_X$ = 6.0 MHz, 10.5 μs at $f_X$ = 4.19 MHz)

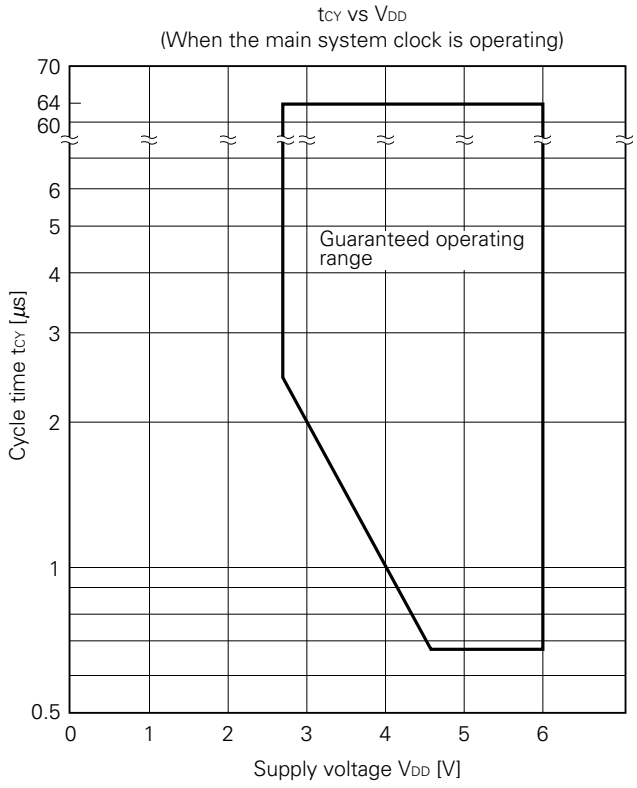
**AC CHARACTERISTICS** ($T_a$ = -40 to +85 °C, $V_{DD}$ = 2.7 to 6.0 V)

**(1) Basic operation**

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions |
|---|---|---|---|---|---|---|
| CPU clock cycle time (minimum instruction execution time)**Note 1** | $t_{CY}$ | 0.67 | | 64 | μs | Operated by main system clock pulse; $V_{DD}$ = 4.5 to 6.0 V |
| | | 2.6 | | 64 | μs | Operated by main system clock pulse |
| | | 114 | 122 | 125 | μs | Operated by subsystem clock pulse |
| TI0 input frequency | $f_{TI}$ | 0 | | 1 | MHz | $V_{DD}$ = 4.5 to 6.0 V |
| | | 0 | | 275 | kHz | |
| TI0 input high/low level width | $t_{TIH}$, $t_{TIL}$ | 0.48 | | | μs | $V_{DD}$ = 4.5 to 6.0 V |
| | | 1.8 | | | μs | |
| Interrupt input high/low level width | $t_{INTH}$, $t_{INTL}$ | **Note 2** | | | μs | INT0 |
| | | 10 | | | μs | INT1, INT2, and INT4 |
| $\overline{\text{RESET}}$ low level width | $t_{RSL}$ | 10 | | | μs | |

**Notes 1.** The cycle time of CPU clock (Φ) depends on the connected resonator frequency, the system clock control register (SCC), and the processor clock control register (PCC). The figure on the next page shows the cycle time $t_{CY}$ characteristics for the supply voltage $V_{DD}$ during main system clock operation.

**2.** This value becomes $2t_{CY}$ or $128/f_X$ according to the setting of the interrupt mode register (IM0).



### (2) Serial transfer operation

#### (a) Two-wire and three-wire serial I/O modes ($\overline{SCK}$ ... Internal clock output):

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | |
|---|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY1}$ | 1340 | | | ns | $V_{DD}$ = 4.5 to 6.0 V | $f_X$ = 6.0 MHz |
| | | 1600 | | | ns | | $f_X$ = 4.19 MHz |
| | | 2680 | | | ns | | $f_X$ = 6.0 MHz |
| | | 3800 | | | ns | | $f_X$ = 4.19 MHz |
| $\overline{SCK}$ high/low level width | $t_{KL1}$ | ($t_{KCY}$/2) – 50 | | | ns | $V_{DD}$ = 4.5 to 6.0 V | |
| | $t_{KH1}$ | ($t_{KCY}$/2) – 150 | | | ns | | |
| SI setup time (referred to $\overline{SCK}$↑) | $t_{SIK1}$ | 150 | | | ns | | |
| SI hold time (referred to $\overline{SCK}$↑) | $t_{KSI1}$ | 400 | | | ns | | |
| $\overline{SCK}$↓ → SO output delay time | $t_{KSO1}$ | | | 250 | ns | $R_L$ = 1 kΩ, $C_L$ = 100 pF[Note] | $V_{DD}$ = 4.5 to 6.0 V |
| | | | | 1000 | ns | | |

**Note** $R_L$ and $C_L$ are the resistance and capacitance of the SO output line load respectively.

#### (b) Two-wire and three-wire serial I/O modes ($\overline{SCK}$ ... External clock input):

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | |
|---|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY2}$ | 800 | | | ns | $V_{DD}$ = 4.5 to 6.0 V | |
| | | 3200 | | | ns | | |
| $\overline{SCK}$ high/low level width | $t_{KL2}$ | 400 | | | ns | $V_{DD}$ = 4.5 to 6.0 V | |
| | $t_{KH2}$ | 1600 | | | ns | | |
| SI setup time (referred to $\overline{SCK}$↑) | $t_{SIK2}$ | 100 | | | ns | | |
| SI hold time (referred to $\overline{SCK}$↑) | $t_{KSI2}$ | 400 | | | ns | | |
| $\overline{SCK}$↓ → SO output delay time | $t_{KSO2}$ | | | 300 | ns | $R_L$ = 1 kΩ, $C_L$ = 100 pF[Note] | $V_{DD}$ = 4.5 to 6.0 V |
| | | | | 1000 | ns | | |

**Note** $R_L$ and $C_L$ are the resistance and capacitance of the SO output line load respectively.

**(c) SBI mode ($\overline{SCK}$ ... Internal clock output (master)):**

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | |
|---|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY3}$ | 1340 | | | ns | $V_{DD}$ = 4.5 to 6.0 V | $f_X$ = 6.0 MHz |
| | | 1600 | | | ns | | $f_X$ = 4.19 MHz |
| | | 2680 | | | ns | | $f_X$ = 6.0 MHz |
| | | 3800 | | | ns | | $f_X$ = 4.19 MHz |
| $\overline{SCK}$ high/low level width | $t_{KL3}$ | $t_{KCY}$/2 - 50 | | | ns | $V_{DD}$ = 4.5 to 6.0 V | |
| | $t_{KH3}$ | $t_{KCY}$/2 - 150 | | | ns | | |
| SB0/SB1 setup time (referred to $\overline{SCK}$↑) | $t_{SIK3}$ | 150 | | | ns | | |
| SB0/SB1 hold time (referred to $\overline{SCK}$↑) | $t_{KSI3}$ | $t_{KCY}$/2 | | | ns | | |
| $\overline{SCK}$↓ → SB0/SB1 output delay time | $t_{KSO3}$ | 0 | | 250 | ns | $R_L$ = 1 kΩ, $C_L$ = 100 pF[Note] | $V_{DD}$ = 4.5 to 6.0 V |
| | | 0 | | 1000 | ns | | |
| $\overline{SCK}$↑ → SB0/SB1↓ | $t_{KSB}$ | $t_{KCY}$ | | | ns | | |
| SB0/SB1↓ → $\overline{SCK}$ | $t_{SBK}$ | $t_{KCY}$ | | | ns | | |
| SB0/SB1 low level width | $t_{SBL}$ | $t_{KCY}$ | | | ns | | |
| SB0/SB1 high level width | $t_{SBH}$ | $t_{KCY}$ | | | ns | | |

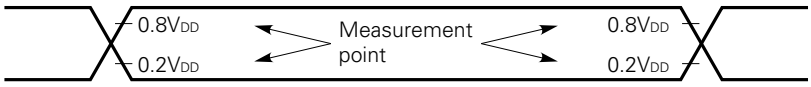
**Note** $R_L$ and $C_L$ are the resistance and capacitance of the SO output line load respectively.

**(d) SBI mode ($\overline{SCK}$ ... External clock input (slave)):**

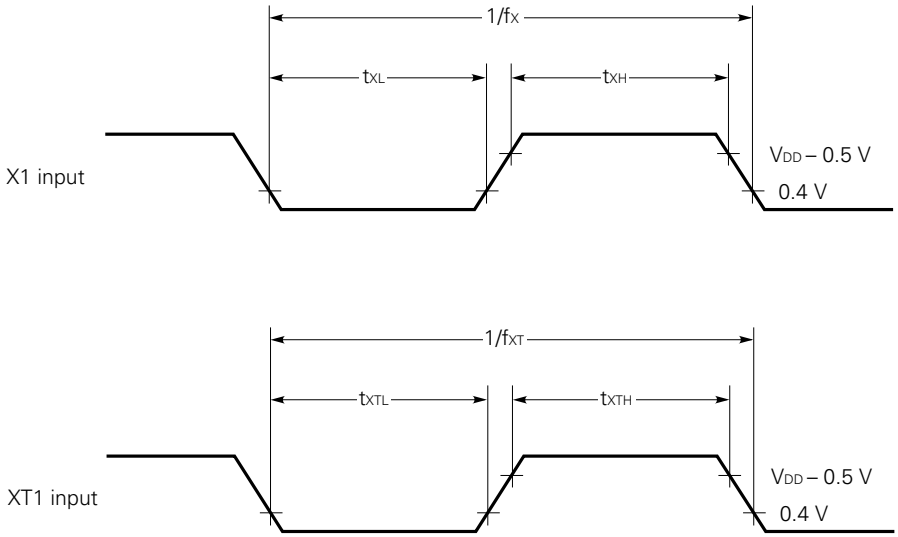
| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions | |
|---|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY4}$ | 800 | | | ns | $V_{DD}$ = 4.5 to 6.0 V | |
| | | 3200 | | | ns | | |
| $\overline{SCK}$ high/low level width | $t_{KL4}$ | 400 | | | ns | $V_{DD}$ = 4.5 to 6.0 V | |
| | $t_{KH4}$ | 1600 | | | ns | | |
| SB0/SB1 setup time (referred to $\overline{SCK}$↑) | $t_{SIK4}$ | 100 | | | ns | | |
| SB0/SB1 hold time (referred to $\overline{SCK}$↑) | $t_{KSI4}$ | $t_{KCY}$/2 | | | ns | | |
| $\overline{SCK}$↓ → SB0/SB1 output delay time | $t_{KSO4}$ | 0 | | 300 | ns | $R_L$ = 1 kΩ, $C_L$ = 100 pF[Note] | $V_{DD}$ = 4.5 to 6.0 V |
| | | 0 | | 1000 | ns | | |
| $\overline{SCK}$↑ → SB0/SB1↓ | $t_{KSB}$ | $t_{KCY}$ | | | ns | | |
| SB0/SB1↓ → $\overline{SCK}$ ↓ | $t_{SBK}$ | $t_{KCY}$ | | | ns | | |
| SB0/SB1 low level width | $t_{SBL}$ | $t_{KCY}$ | | | ns | | |
| SB0/SB1 high level width | $t_{SBH}$ | $t_{KCY}$ | | | ns | | |

**Note** $R_L$ and $C_L$ are the resistance and capacitance of the SO output line load respectively.

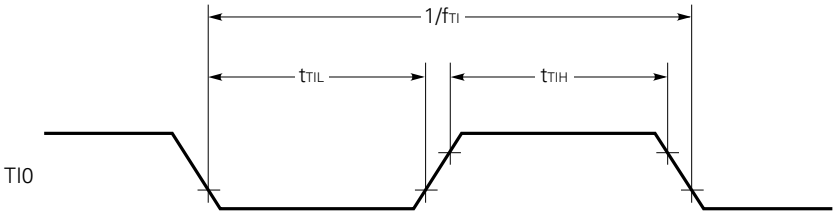
### AC Timing Measurement Points (Excluding X1 and XT1 Inputs)




### Clock Timing

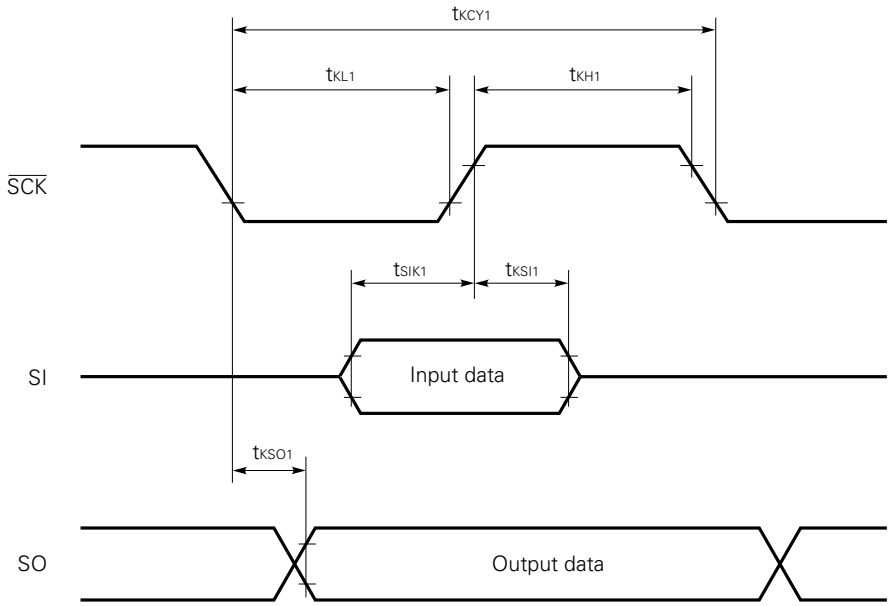



### TI0 Timing

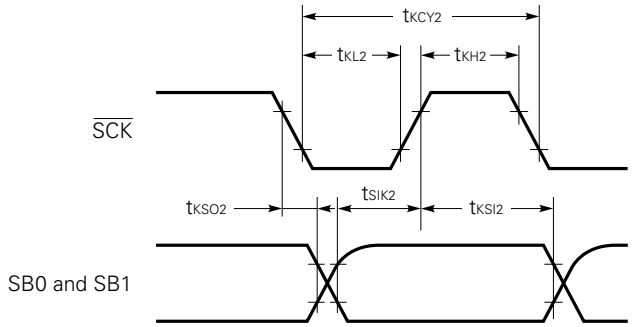


### Serial Transfer Timing

**Three-wire serial I/O mode:**

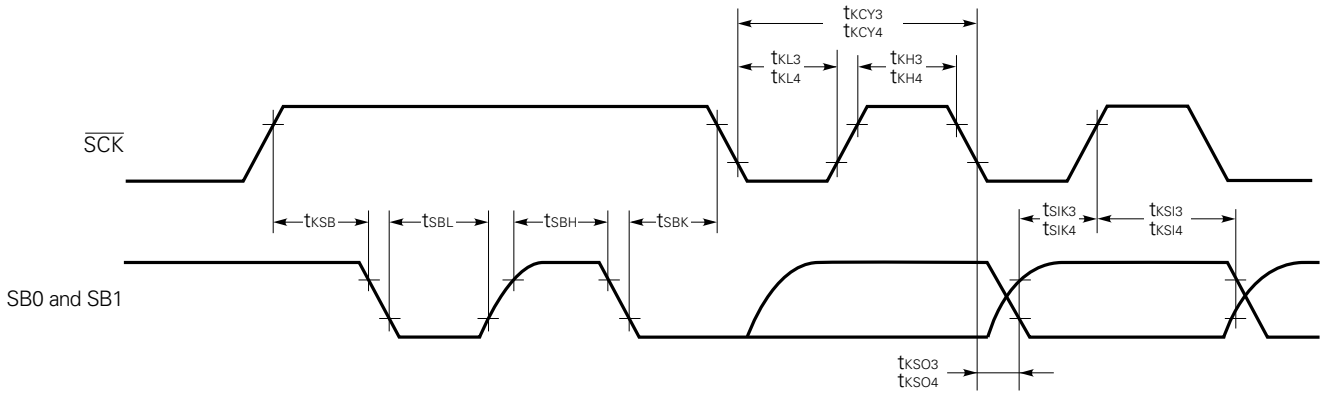



**Two-wire serial I/O mode:**

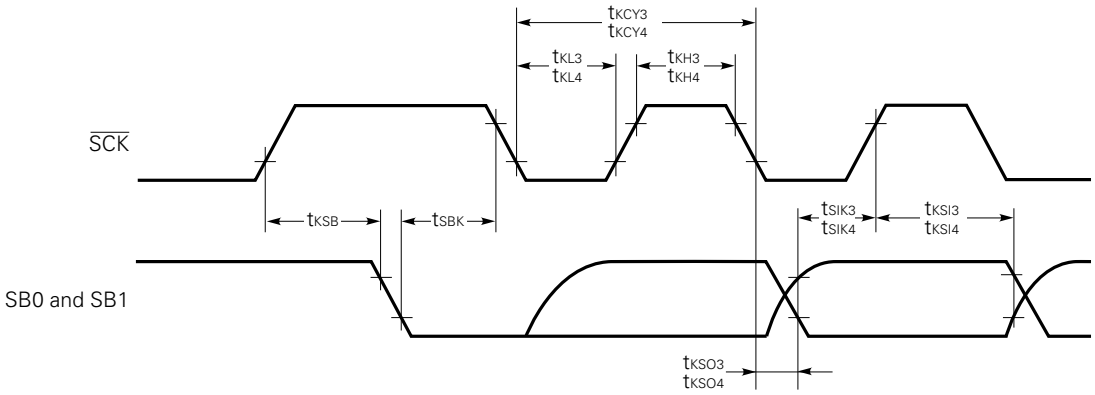


### Serial Transfer Timing

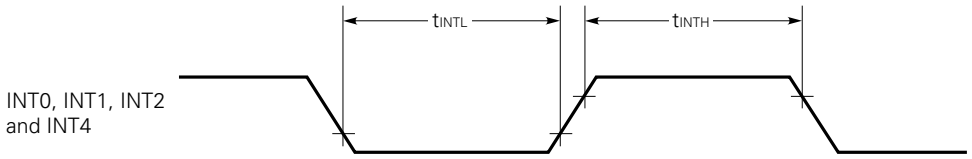
**Bus release signal transfer:**



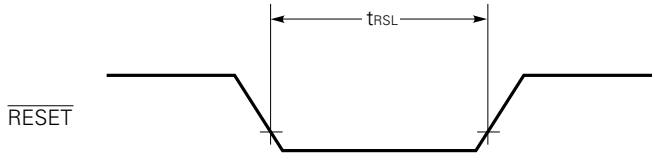

**Command signal transfer:**




### Interrupt Input Timing




### RESET Input Timing



## DATA HOLD CHARACTERISTICS BY LOW SUPPLY VOLTAGE IN DATA MEMORY STOP MODE

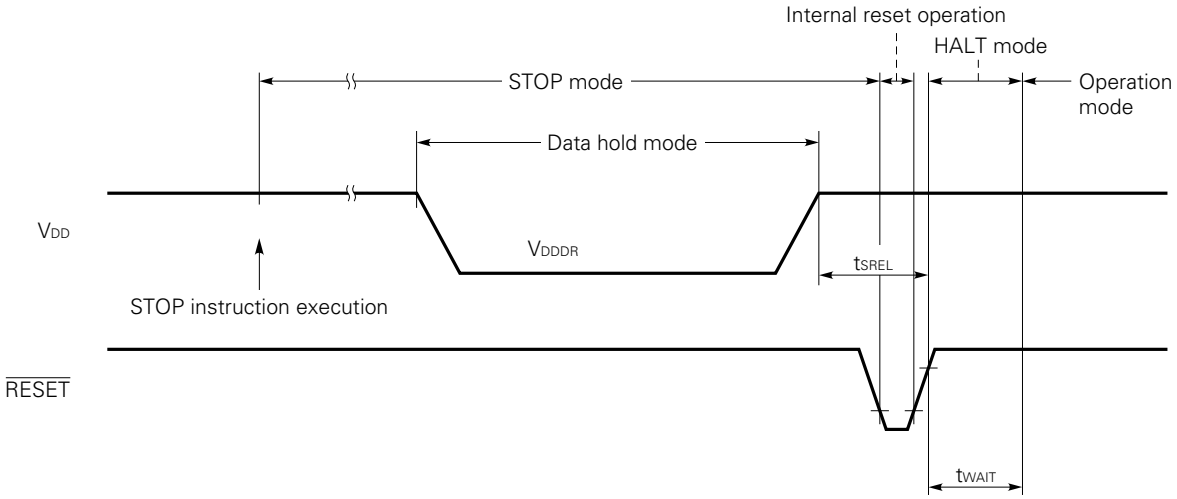
($T_a$ = -40 to +85 °C)

| Parameter | Symbol | Min. | Typ. | Max. | Unit | Conditions |
|---|---|---|---|---|---|---|
| Data hold supply voltage | $V_{DDDR}$ | 2.0 | | 6.0 | V | |
| Data hold supply current[Note 1] | $I_{DDDR}$ | | 0.1 | 10 | μA | $V_{DDDR}$ = 2.0 V |
| Release signal setting time | $t_{SREL}$ | 0 | | | μs | |
| Oscillation stability wait time[Note 2] | $t_{WAIT}$ | | $2^{17}/f_X$ | | ms | Release by $\overline{\text{RESET}}$ |
| | | | Note 3 | | ms | Release by interrupt request |

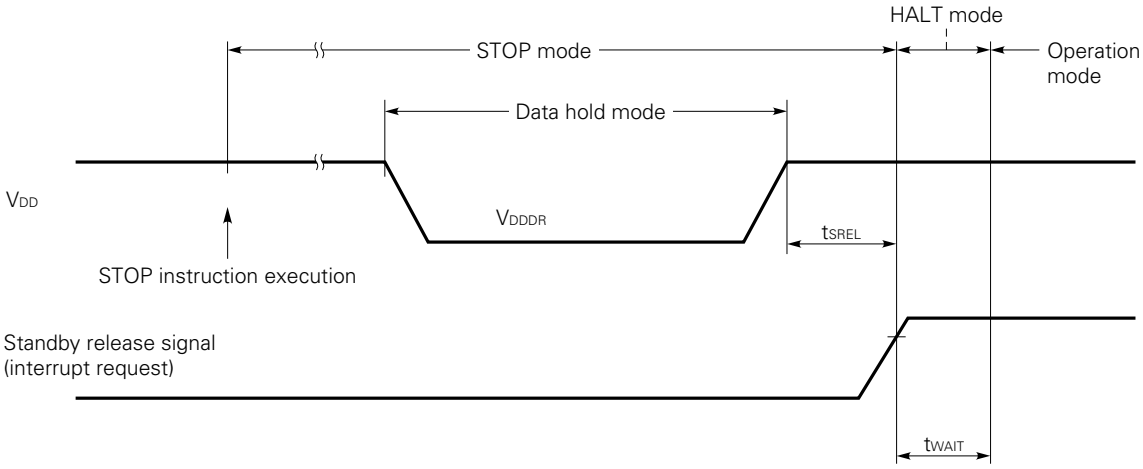
**Notes 1.** Excluding the current which flows through the built-in pull-up or pull-down resistors
**2.** CPU operation stop time for preventing unstable operation at the beginning of oscillation
**3.** This value depends on the settings of the basic interval timer mode register (BTM) shown below.

| BTM3 | BTM2 | BTM1 | BTM0 | Wait time | |
|---|---|---|---|---|---|
| | | | | $f_X$ = 6.0 MHz | $f_X$ = 4.19 MHz |
| — | 0 | 0 | 0 | $2^{20}/f_X$ (approx. 175 ms) | $2^{20}/f_X$ (approx. 250 ms) |
| — | 0 | 1 | 1 | $2^{17}/f_X$ (approx. 21.8 ms) | $2^{17}/f_X$ (approx. 31.3 ms) |
| — | 1 | 0 | 1 | $2^{15}/f_X$ (approx. 5.46 ms) | $2^{15}/f_X$ (approx. 7.82 ms) |
| — | 1 | 1 | 1 | $2^{13}/f_X$ (approx. 1.37 ms) | $2^{13}/f_X$ (approx. 1.95 ms) |

### Data Hold Timing (STOP Mode Release by $\overline{\text{RESET}}$)

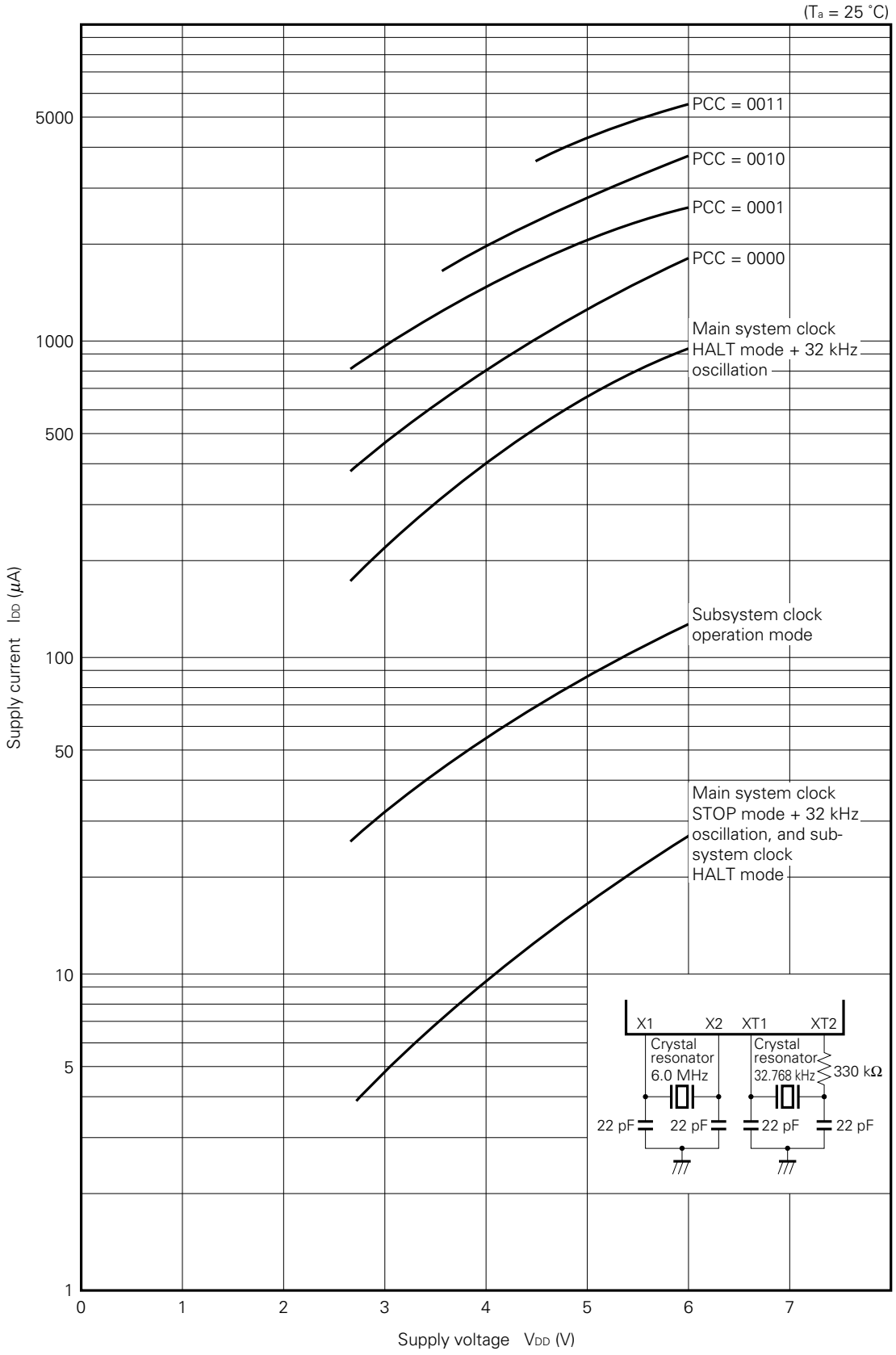



### Data Hold Timing (Standby Release Signal: STOP Mode Release by Interrupt Signal)

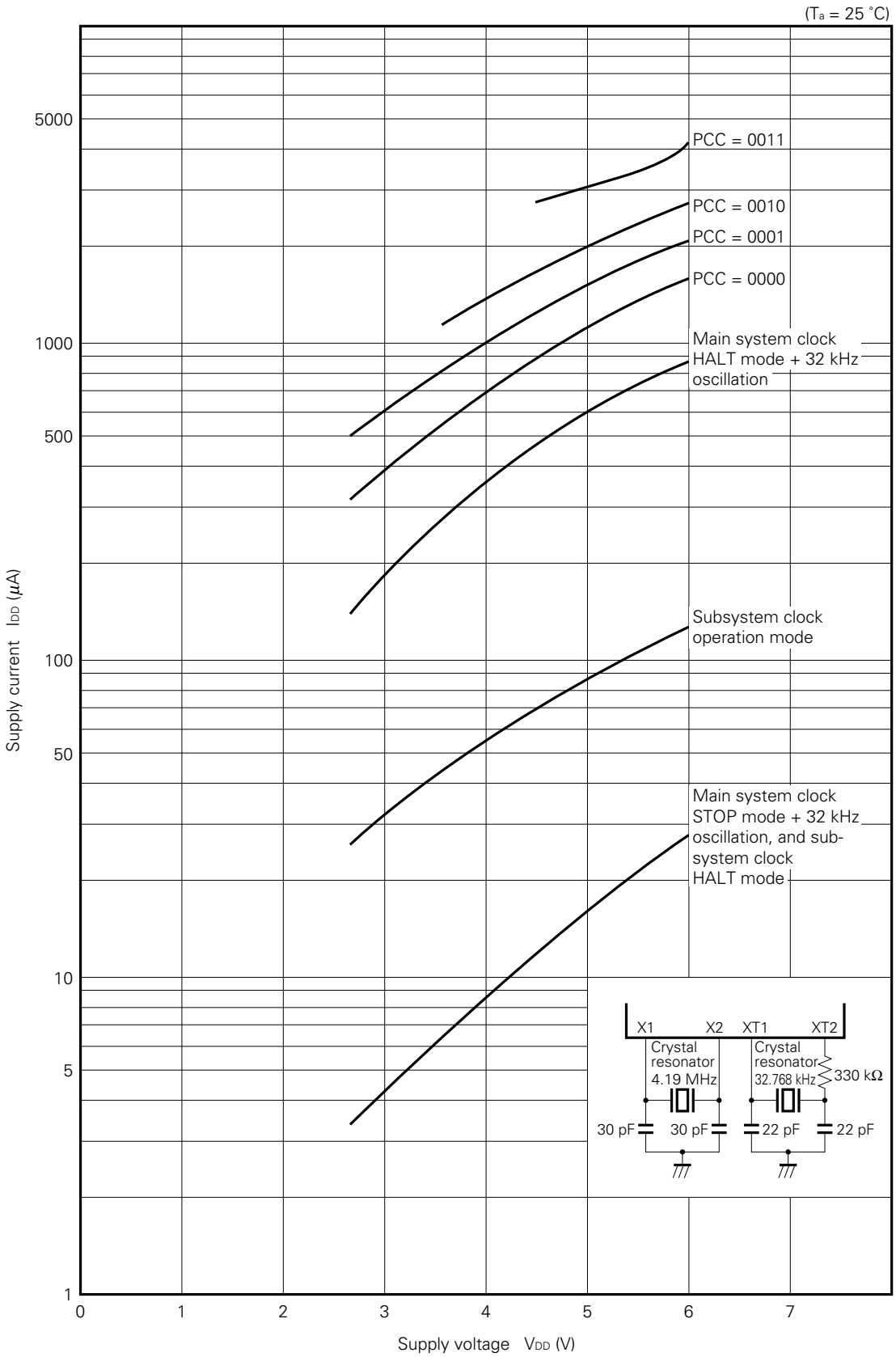


## 12. CHARACTERISTIC CURVES (FOR REFERENCE) ★

**$I_{DD}$ vs $V_{DD}$ (Main System Clock: 6.0 MHz)**

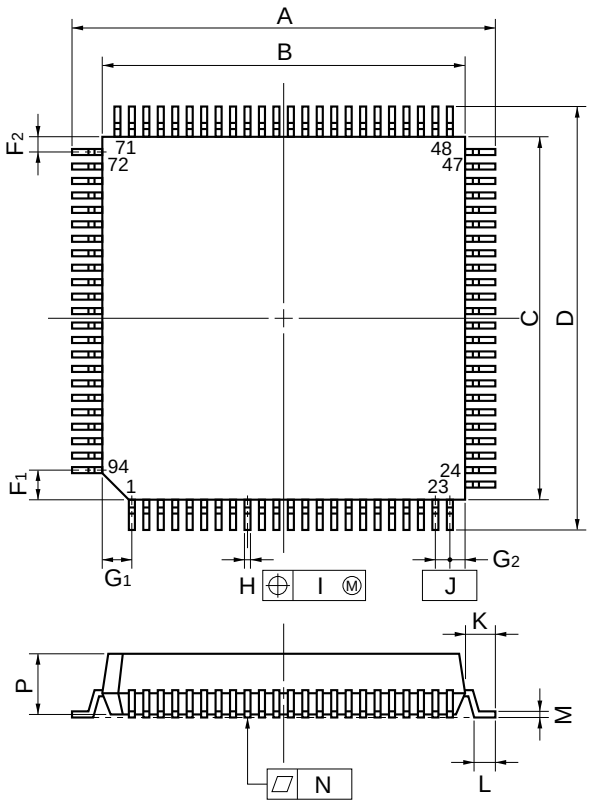


**$I_{DD}$ vs $V_{DD}$ (Main System Clock: 4.19 MHz)**

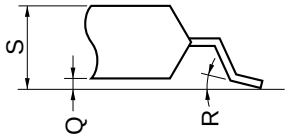


## 13. PACKAGE DIMENSIONS

### 94 PIN PLASTIC QFP (□20)




detail of lead end




**NOTE**

Each lead centerline is located within 0.15 mm (0.006 inch) of its true position (T.P.) at maximum material condition.

| ITEM | MILLIMETERS | INCHES |
|---|---|---|
| A | 23.2±0.4 | $0.913^{+0.017}_{-0.016}$ |
| B | 20.0±0.2 | $0.787^{+0.009}_{-0.008}$ |
| C | 20.0±0.2 | $0.787^{+0.009}_{-0.008}$ |
| D | 23.2±0.4 | $0.913^{+0.017}_{-0.016}$ |
| $F_1$ | 1.6 | 0.063 |
| $F_2$ | 0.8 | 0.031 |
| $G_1$ | 1.6 | 0.063 |
| $G_2$ | 0.8 | 0.031 |
| H | 0.35±0.10 | $0.014^{+0.004}_{-0.005}$ |
| I | 0.15 | 0.006 |
| J | 0.8 (T.P.) | 0.031 (T.P.) |
| K | 1.6±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 | 3.7 | 0.146 |
| Q | 0.1±0.1 | 0.004±0.004 |
| R | 5°±5° | 5°±5° |
| S | 4.0 MAX. | 0.158 MAX. |

**S94GJ-80-5BG-3**

## ★ 14. RECOMMENDED SOLDERING CONDITIONS

The following conditions must be met when soldering the μPD75238.

Please consult with our sales offices in case other soldering process is used, or in case soldering is done under different conditions.

**Table 14-1 Recommended Soldering Conditions**

| Part number | Package | Symbol |
|---|---|---|
| μPD75238GJ-×××-5BG | 94-pin plastic QFP | WS60-107-1<br>IR30-107-1<br>VP15-107-1<br>Partial heating method |

**Table 14-2 Soldering Conditions**

| Symbol | Soldering process | Soldering conditions |
|---|---|---|
| WS60-107-1 | Wave soldering | Temperature in the soldering vessel: 260 °C or below<br>Soldering time: 10 seconds or less<br>Number of soldering processes: 1<br>Exposure limit: 7 days[Note]<br>(10 hours of pre-baking is required at 125 °C afterward.)<br>Pre-heating temperature: 120 °C max.<br>(package surface temperature) |
| IR30-107-1 | Infrared ray reflow | Peak package's surface temperature: 230 °C<br>Reflow time: 30 seconds or less (at 210 °C or higher)<br>Number of reflow processes: 1<br>Exposure limit: 7 days[Note]<br>(10 hours of pre-baking is required at 125 °C afterward.) |
| VP15-107-1 | VPS | Peak package's surface temperature: 215 °C<br>Reflow time: 40 seconds or less (at 200 °C or higher)<br>Number of reflow processes: 1<br>Exposure limit: 7 days[Note]<br>(10 hours of pre-baking is required at 125 °C afterward.) |
| Partial heating method | Partial heating method | Terminal temperature: 300 °C or below<br>Flow time: 3 seconds or less (one side per device) |

**Note** Exposure limit before soldering after dry-pack package is opened. Storage conditions: 25 °C and relative humidity at 65 % or less.

**Caution Do not apply more than a single process at a time, except for "Partial heating method."**

**Remark** For more details, refer to our document "SMT MANUAL" (IEI-1207).

## APPENDIX A μPD75238 SERIES PRODUCT FUNCTION LIST

| Item \ Product | | μPD75217 | μPD75236 | μPD75237 | μPD75238 | μPD75P238 |
|---|---|---|---|---|---|---|
| ROM | | 24448 × 8 | 16256 × 8 | 24448 × 8 | 32640 × 8 | |
| RAM | | 768 × 4 | | 1024 × 4 | | |
| Instruction cycle | When main system clock is selected | 0.95 μs/1.91 μs/15.3 μs (at 4.19 MHz) | 0.95 μs/1.91 μs/3.82 μs/15.3 μs (at 4.19 MHz) | 0.67 μs/1.33 μs/2.67 μs/10.7 μs (at 6.0 MHz) | | |
| | When sub-system clock is selected | 122 μs (at 32.768 kHz) | | | | |
| I/O lines (including FIP dual-function pins and excluding FIP-exclusive pins) | Total number of I/O lines | 33 | 64 | | | |
| | Number of input lines | 8 | 16 | | | |
| | Number of I/O lines | 20: Eight lines for driving LED | 24: 12 lines for driving LED | | | |
| | Number of output lines | 5 | 24 | | | |
| A/D converter | | None | 8 : 8-bit resolution | | | |
| FIP controller/driver | High-voltage output | 26: 40 V (max.) | 34: 40 V (max.) | | | |
| | Number of segments | 9 to 16 | 9 to 24 | | | |
| | Number of digits | 9 to 16 | | | | |
| Timer | | Four channels | Five channels | | | |
| Serial interface | | One channel: 3-wire system | Two channels { SBI/3-wire system; 3-wire system } | | | |
| Number of interrupt sources | | 10 | 11 | | | |
| Range of operating temperature | | -40 to +85 °C | | | | -40 to +70 °C ★ |
| Operating power voltage | | 2.7 to 6.0 V | | | | |
| Package | | 64-pin plastic shrink DIP<br>64-pin plastic QFP | 94-pin plastic QFP | | | 94-pin plastic QFP<br>94-pin ceramic LCC with a window |

## ★ APPENDIX B DEVELOPMENT TOOLS

The following development tools are provided for developing a system which employs the μPD75238.

### Language processor

| | | | | |
|---|---|---|---|---|
| RA75X relocatable assembler | Host machine | OS | Distribution media | Part number |
| | PC-9800 series | MS-DOS™ Ver. 3.10 to Ver. 3.30C | 3.5-inch 2HD | μS5A13RA75X |
| | | | 5-inch 2HD | μS5A10RA75X |
| | IBM PC series | PC DOS™ (Ver. 3.1) | 5-inch 2HC | μS7B10RA75X |

### PROM programming tools

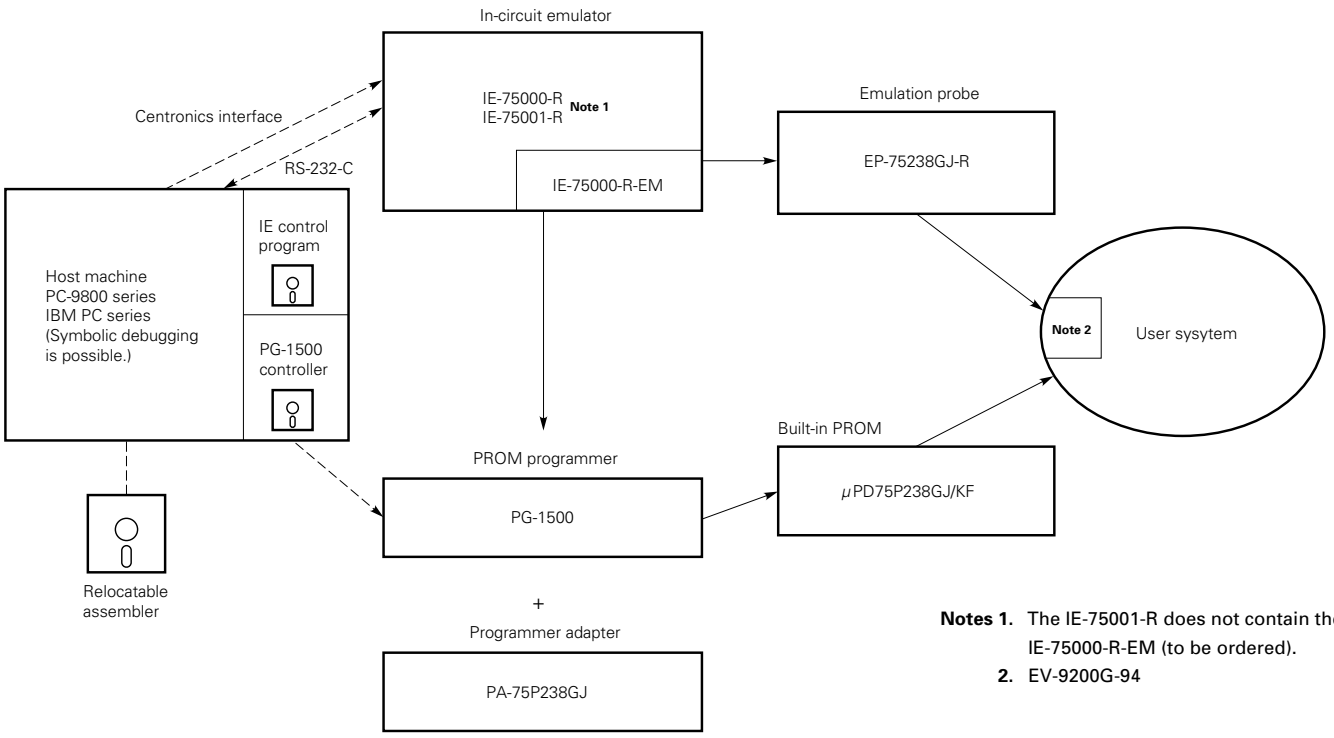
| | | | | |
|---|---|---|---|---|
| Hardware | PG-1500 | A PROM programmer with which programs can be written into PROMs through keyboards or by remote control, when connected with the accessory board and optional programmer adapter. Products programmable with the PG-1500 are commonly-used PROMs (256K-bit to 1M-bit) and single chip microcomputers containing a PROM. | | |
| | PA-75P238GJ | A PROM programmer adapter for the μPD75P238. This adapter is connected to the PG-1500. | | |
| Software | PG-1500 controller | This program enables the host machine to control the PG-1500 through the serial and parallel interfaces. | | |
| | | Host machine | OS | Distribution media | Part number |
| | | PC-9800 series | MS-DOS Ver. 3.10 to Ver. 3.30C | 3.5-inch 2HD | μS5A13PG1500 |
| | | | | 5-inch 2HD | μS5A10PG1500 |
| | | IBM PC series | PC DOS (Ver. 3.1) | 5-inch 2HC | μS7B10PG1500 |

**Debugging tools**

<table>
<tr><td rowspan="6">Hardware</td><td>IE-75000-RNote</td><td colspan="4">The IE-75000-R is an in-circuit emulator available for the 75X series. This emulator is used together with the emulation probe to develop application systems of the μPD75238. For efficient debugging, the emulator is connected to the host machine and PROM programmer.</td></tr>
<tr><td>IE-75000-R-EM</td><td colspan="4">The IE-75000-R-EM is an emulation board for the IE-75000-R and IE-75001-R. The IE-75000-R contains the emulation board. The emulation board is used together with the IE-75000-R or IE-75001-R to evaluate the μPD75238.</td></tr>
<tr><td>IE-75001-R</td><td colspan="4">The IE-75001-R is an in-circuit emulator available for the 75X series. This emulator is used together with the IE-75000-R-EM emulation board (option) and emulation probe to develop application systems of the μPD75238. For efficient debugging, the emulator is connected to the host machine and PROM programmer.</td></tr>
<tr><td>EP-75238GJ-R</td><td colspan="4" rowspan="2">The EP-75238GJ-R is an emulation probe for the μPD75238 (94-pin plastic QFP). The emulation probe is connected to the IE-75000-R or IE-75001-R when it is used. A 94-pin conversion socket, the EV-9200G-94, attached to the probe facilitates the connection of the prove with the user system.</td></tr>
<tr><td>EV-9200G-94</td></tr>
<tr><td rowspan="4">Software</td><td rowspan="4">IE control program</td><td colspan="4">This program enables the host machine to control the IE-75000-R or IE-75001-R through the RS-232-C interface.</td></tr>
<tr><td rowspan="2">Host machine</td><td rowspan="2">OS</td><td rowspan="2">Distribution media</td><td rowspan="2">Part number</td></tr>
<tr></tr>
<tr><td>PC-9800 series</td><td rowspan="2">MS-DOS<br>(Ver. 3.10<br>to<br>Ver. 3.30C)</td><td>3.5-inch 2HD</td><td>μS5A13IE75X</td></tr>
<tr><td>IBM PC series</td><td>PC DOS<br>(Ver. 3.1)</td><td>5-inch 2HD</td><td>μS5A10IE75X</td></tr>
<tr><td></td><td></td><td></td><td>5-inch 2HC</td><td>μS7B10IE75X</td></tr>
</table>

**Note** Maintenance service only

**Development tool configuration**




**Notes 1.** The IE-75001-R does not contain the IE-75000-R-EM (to be ordered).

**2.** EV-9200G-94

[MEMO]

**[MEMO]**



M4 92.6